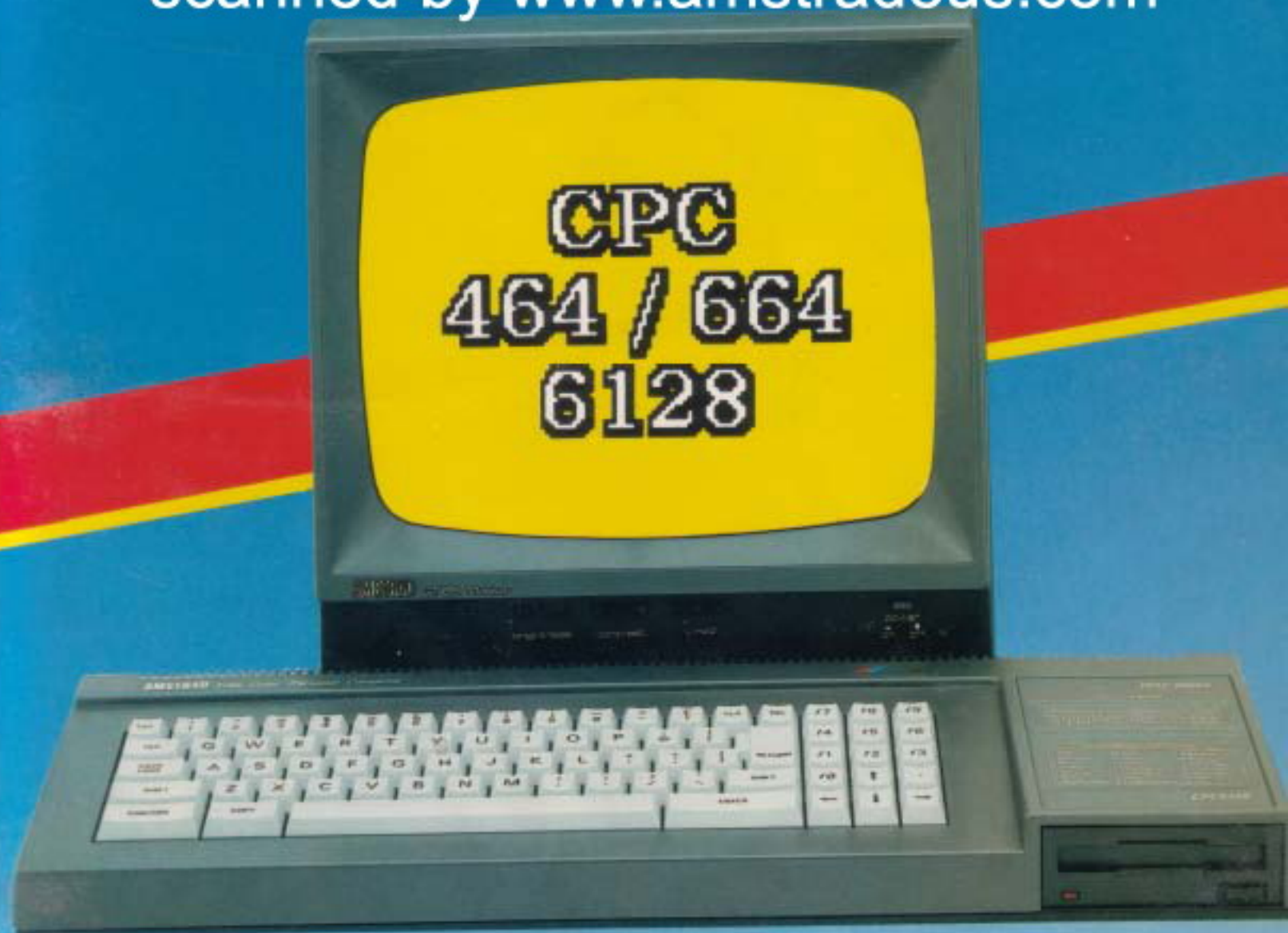


Comment exploiter
toutes les ressources
et augmenter les
performances de votre

AMSTRAD

scanned by www.amstradeus.com



9/0

Table des matières

9/1 Savoir programmer

9/2 Moniteur : Assembleur - désassembleur - debugger

9/2.1 Le désassembleur

9/2.2 L'Assembleur

- I. Programme de saisie de texte
- II. Compilateur
 - A. Définitions
 - B. Rappels
- III. Entrées/Sorties sur disquettes

9/2.3 Le debugger

- I. Lecture d'un programme objet
- II. Sauvegarde d'un programme objet ou d'une zone mémoire
- III. Exécution d'un programme
- IV. Affichage du contenu d'une mémoire
- V. Affichage d'une zone mémoire : fonction DUMP
- VI. Retour au BASIC

9/3 Jeux d'esprit

9/3.1 Le jeu du taquin

9/3.2 Renversé

9/3.3 Tours de Hanoi

9/3.4 Jeu des allumettes

9/3.5 Awari

9/3.6 Jeu du Simon

9/4 Mathématiques

9/4.1 Nom d'un jour de la semaine

9/4.2 Calendrier perpétuel

9/4.3 Biorythmes

- 9/5 Gestion de fichier**
- 9/6 Jeux d'aventures**
 - I. Création de l'univers de jeu
- 9/6.1 Analyse syntaxique d'une phrase**
- 9/6.2 Fonction LOCATE-INPUT**
- 9/6.3 Fonction HELP**
- 9/6.4 Création de jeux d'aventures**
- 9/6.5 Exécution de jeux d'aventures**
- 9/7 Jeux d'Arcade**
- 9/7.1 Casse briques**
- 9/7.2 Bataille navale**
- 9/7.3 Danger piranhas**
- 9/8 Utilitaires**
- 9/8.1 Copie d'écran graphique**
 - 9/8.1.1 Turbo copie d'écran graphique
- 9/8.2 Commande PIP en Basic**
- 9/8.3 Transformation du clavier QWERTY en clavier AZERTY sous CP/M Plus**
- 9/8.4 Checksum, vérificateur de données**
- 9/8.5 Dump hexadécimal et ASCII**
 - 9/8.5.1 Programme de Dump en Basic
 - 9/8.5.2 Programme de Dump en Assembleur
- 9/8.6 Récupération d'un fichier effacé par la commande IERA**
- 9/8.7 Défilement d'un message alphanumérique sur l'écran**
- 9/8.8 Driver d'imprimante, DMP 2000**
 - I. Élément de base sur la définition de caractères graphiques
 - II. Définition de caractères graphiques
 - III. Impression de textes en utilisant un driver d'imprimante
 - IV. Utilisation du driver DMP 2000 sur d'autres ordinateurs que les Amstrad CPC
- 9/8.9 Instruction CAT évoluée**
- 9/8.10 Edition et modification des secteurs d'une disquette**
- 9/8.11 CAPS LOCK Interactif**
- 9/8.12 Protection écran (*screen saver*)**
- 9/8.13 Chargeur hexadécimal**

9/8.14 Formattage des listings

- I. Formattage des listings en Assembleur
- II. Formattage des listings en Basic
- III. Formattage des listings en Turbo-Pascal

9/9 Programmes divers**9/9.1 Générateur de signaux morse**

- I. Reconnaître une touche frappée
- II. Inhibition de l'affichage écran
- III. Interruption du fonctionnement pour traitement

9/9.2 Filtrage de fichiers ASCII**9/9.3 Transformez votre Amstrad CPC + DMP 2000 en machine à écrire****9/10 Gestion familiale****9/10.1 Gestion de compte bancaire**

- I. But du programme
- II. Mode d'emploi
 - A. Création du fichier
 - B. Utiliser la gestion de compte
 - C. Le menu
- III. Extension
- IV. Le programme
 - A. COMPTBIN
 - B. OUVCOMPT
 - C. COMPTE 1
 - D. COMPTE 2

9/10.2 Gestion de logiciels

- I. Présentation
- II. Organisation matérielle et logiciel
- III. Le programme
- IV. Le principe du tri
- V. Modification du programme

9/11 Traitement de texte**9/11.1 Mise en œuvre d'utilitaires****9/11.2 Le traitement de texte Weka****9/11.2.1 Fonctions élémentaires**

- I. Saisie, affichage et mémorisation de caractères sur l'écran en pleine page
- II. Gestion du curseur (haut, bas, droit et gauche)
- III. Commandes de scrolling bas et haut d'une ligne d'écran
- IV. Touches CLR et DEL pour l'effacement d'un caractère

9/11.2.2 Premier jeu de fonctions évoluées

I. Déplacements rapides

II. Résumé des fonctions disponibles

9/11.2.3 Second jeu de fonctions évoluées

9/12 Correcteurs orthographiques

9/12.1 Correcteur orthographique de base

9/1

Savoir programmer

Les programmes présentés dans la partie 9 suivent la méthode de programmation hiérarchisée/structurée développée au chapitre 1.5 de la partie 4.

Reportez-vous à ce chapitre pour avoir plus de précisions à ce sujet.

Comme vous pourrez le voir dans la suite, tous les programmes comportent un programme principal, de niveau hiérarchique le plus élevé, situé en début de programme. Ce programme principal fait appel à un ou plusieurs sous-programmes de niveau hiérarchique juste inférieur, qui, eux mêmes font appel à un ou plusieurs sous-programmes de niveau hiérarchique juste inférieur.

Les niveaux hiérarchiques sont séparés par des lignes de remarque où apparaissent des signes égale, comme suit :

1200 REM=====

A l'intérieur d'un même niveau, les grands groupes fonctionnels sont séparés par des lignes de remarque où apparaissent des tirets comme suit :

4320 REM-----

9/2

Moniteur : Assembleur Désassembleur Debugger

Nous allons créer un assembleur/désassembleur/debugger. Cet utilitaire est à rapprocher du chapitre 2.6 de la partie 4 (Assembleurs existants). Vous trouverez dans la suite la description détaillée de la façon dont est écrit un assembleur Z80.

9/2.1

Le désassembleur

Qu'est-ce qu'un désassembleur ?

Comme son nom l'indique, c'est un programme qui transforme le code hexadécimal exécutable situé en mémoire en mnémoniques assembleurs. Par opposition, un programme d'assemblage ou assembleur transformera les codes mnémoniques en codes hexadécimaux exécutables.

Analyse du problème :

Comment procéder pour convertir les codes hexadécimaux en mnémoniques ?

Le micro-processeur Z80 possède de nombreux modes d'adressage. Pour clarifier les choses, établissons un tableau qui donne les codes mnémoniques correspondant à chaque code hexadécimal sur 8 bits.

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
NOP	LD BC,NN	LD (BC),A	INC BC	INC B	DEC B	LD B,n	RLCA	EX AF,AF'	ADD HL,BC	LD A,(BC)	DEC BC	INC C	DEC C	LD C,n	RRCA
DJNZ DEPL	LD DE,NN	LD (DE),A	INC DE	INC D	DEC D	LD D,n	RL A	JR DEPL	ADD HL,DE	LD A,(DE)	DEC DE	INC E	DEC E	LD E,n	RRA
JR NZ,DEPL	LD HL,mn	LD (mn),HL	INC HL	INC H	DEC H	LD H,n	DAA	JR Z,DEPL	ADD HL,HL	LD HL,(mn)	DEC HL	INC L	DEC L	LD L,n	CPL
JR NC,DEPL	LD SP,mn	LD (mn),A	INC SP	INC (HL)	DEC (HL)	LD (HL),n	SCF	JR C,DEPL	ADD HL,SP	LD A,(NN)	DEC SP	INC A	DEC A	LD A,n	CCF
LD B,B	LD B,C	LD B,D	LD B,E	LD B,H	LD B,L	LD B,(HL)	LD B,A	LD C,B	LD C,C	LD C,D	LD C,E	LD C,H	LD C,L	LD C,(HL)	LD C,A
LD D,B	LD D,C	LD D,D	LD D,E	LD D,H	LD D,L	LD D,(HL)	LD D,A	LD E,B	LD E,C	LD E,D	LD E,E	LD E,H	LD E,L	LD E,(HL)	LD E,A
LD H,B	LD H,C	LD H,D	LD H,E	LD H,H	LD H,L	LD H,(HL)	LD H,A	LD L,B	LD L,C	LD L,D	LD L,E	LD L,H	LD L,L	LD L,(HL)	LD L,A
LD (HL),B	LD (HL),C	LD (HL),D	LD (HL),E	LD (HL),H	LD (HL),L	HALT	LD (HL),A	LD A,B	LD A,C	LD A,D	LD A,E	LD A,H	LD A,L	LD A,(HL)	LD A,A
ADD A,B	ADD A,C	ADD A,D	ADD A,E	ADD A,H	ADD A,L	ADD A,(HL)	ADD A,A	ADC A,B	ADC A,C	ADC A,D	ADC A,E	ADC A,H	ADC A,L	ADC A,(HL)	ADC A,A
SUB B	SUB C	SUB D	SUB E	SUB H	SUB L	SUB (HL)	SUB A	SBC A,B	SBC A,C	SBC A,D	SBC A,E	SBC A,H	SBC A,L	SBC A,(HL)	SBC A,A
AND B	AND C	AND D	AND E	AND H	AND L	AND (HL)	AND A	XOR B	XOR C	XOR D	XOR E	XOR H	XOR L	XOR (HL)	XOR A
OR B	OR C	OR D	OR E	OR H	OR L	OR (HL)	OR A	CP B	CP C	CP D	CP E	CP H	CP L	CP (HL)	CP A
RET NZ	POP BC	JP NZ,ADR	JP ADR	CALL NZ,ADR	PUSH BC	ADD A,n	RST 00	RET Z	RET	JP Z,ADR	3	CALL Z,ADR	CALL ADR	ADC A,n	RST 08
RET NC	POP DE	JP NC,ADR	OUT (N),A	CALL NC,ADR	PUSH DE	SUB n	RST 10H	RET C	EXX	JP C,ADR	IN A,(IN)	CALL C,ADR	1	SBC A,n	RST 18H
RET PO	POP HL	JP PO,ADR	EX (SP),HL	CALL PO,ADR	PUSH HL	AND n	RST 20H	RET PE	JP (HL)	JP PE,ADR	EX DE,HL	CALL PE,ADR	4	XOR n	RST 28H
RET P	POP AF	JP P,ADR	DI	CALL P,ADR	PUSH AF	OR n	RST 30H	RET M	LD SP,HL	JP M,ADR	EI	CALL M,ADR	2	CP n	RST 38H

- Une *instruction* peut être codée sur 1, 2, 3 ou 4 octets.

Codage sur un octet :

Nous avons, par exemple :

51	LD	D,C
96	SUB	(HL)

Codage sur deux octets :

Nous avons, par exemple :

20 dd	JR	NZ,dd
ED 68	LD	L,(C)

Codage sur trois octets :

Nous avons, par exemple :

DD 86 dd	ADD	A,(IX + dd)
E2 aa aa	JP	PO,aaaa

Codage sur quatre octets :

Nous avons, par exemple :

DD CB dd 7E	BIT	7,(IX + dd)
FD 36 dd nn	LD	(IY + dd),nn

- Les *mnémoniques* « simples » peuvent être codés de trois manières différentes :

1) Codage sur un octet :

Par exemple :

00	NOP
----	-----

Il suffit dans ce cas d'afficher le mnémonique.

2) Codage sur deux octets :

Par exemple :

06 nn	LD B,nn
-------	---------

Dans ce cas, il faut afficher la « **racine** » du mnémonique (le début du code) suivie de l'octet lu en mémoire.

3) Codage sur trois octets :

Par exemple :

CDnnnn	CALL nnnn
--------	-----------

Dans ce cas, il faut afficher la « **racine** » du mnémonique (le début du code) suivie des 2 octets lus en mémoire.

En ce qui concerne les mnémoniques qualifiés de « simples » dans ce qui précède, les références aux contenus sont toujours implicites (sur un registre), comme par exemple LD H,(HL), et ne font jamais appel à la mémoire, aux exceptions près suivantes :

LD	(nnnn),HL	Cas n° 5
LD	HL,(nnnn)	Cas n° 6
LD	(nnnn),A	Cas n° 7
LD	A,(nnnn)	Cas n° 8
OUT	(nnnn),A	Cas n° 9
IN	A,(nnnn)	Cas n° 10

Chacune de ces instructions fera l'objet d'un traitement particulier comme indiqué ci-dessus (cas n° 5 à cas n° 10).

Cette distinction est faite, car, dans le programme désassembleur, l'affichage du mnémonique se fera en trois temps pour ces cas particuliers :

- début du mnémonique (appelé « racine » dans ce qui précède),
- octet ou octets lu(s) en mémoire,
- fin du mnémonique.

Dans un mnémonique « simple », l'affichage se fait de la manière suivante :

- mnémonique,
- éventuellement adresse ou déplacement lu en mémoire.

Les codes CB, DD, ED et FD sont particulièrement complexes. Ils représentent chacun plusieurs dizaines de mnémoniques en fonction des codes hexadécimaux qui les suivent. Nous leur consacrerons un traitement spécial :

1 pour DD, 2 pour FD, 3 pour CB et 4 pour ED.

La liste des mnémoniques possibles pour ces 4 codes spéciaux est la suivante :

1^{er} op - code = DD

8E dd	ADC	A, (IX + dd)
86 dd	ADD	A, (IX + dd)
09	ADD	IX, BC
19	ADD	IX, DE
29	ADD	IX, IX
39	ADD	IX, SP
A6 dd	AND	(IX + dd)
CB dd 46	BIT	0, (IX + dd)
CB dd 4E	BIT	1, (IX + dd)
CB dd 56	BIT	2, (IX + dd)
CB dd 5E	BIT	3, (IX + dd)
CB dd 66	BIT	4, (IX + dd)
CB dd 6E	BIT	5, (IX + dd)
CB dd 76	BIT	6, (IX + dd)
CB dd 7E	BIT	7, (IX + dd)
BE dd	CP	(IX + dd)
35 dd	DEC	(IX + dd)
2B	DEC	IX
E3	EX	(SP), IX
34 dd	INC	(IX + dd)
23	INC	IX
E9	JP	(IX)
77 dd	LD	(IX + dd), A

70 dd	LD	(IX + dd), B
71 dd	LD	(IX + dd), C
72 dd	LD	(IX + dd), D
73 dd	LD	(IX + dd), E
74 dd	LD	(IX + dd), H
75 dd	LD	(IX + dd), L
36 dd nn	LD	(IX + dd), n
22 nn nn	LD	(nn), IX
7E dd	LD	A, (IX + dd)
46 dd	LD	B, (IX + dd)
4E dd	LD	C, (IX + dd)
56 dd	LD	D, (IX + dd)
5E dd	LD	E, (IX + dd)
66 dd	LD	H, (IX + dd)
2A nn nn	LD	IX, (nn)
21 nn nn	LD	IX, nn
6E dd	LD	L, (IX + dd)
F9	LD	SP, IX
B6 dd	OR	(IX + dd)
E1	POP	IX
E5	PUSH	IX
CB dd 80	RES	0, (IX + dd)
CB dd 8E	RES	1, (IX + dd)
CB dd 96	RES	2, (IX + dd)
CB dd 9E	RES	3, (IX + dd)
CB dd A6	RES	4, (IX + dd)
CB dd AE	RES	5, (IX + dd)
CB dd B6	RES	6, (IX + dd)
CB dd BE	RES	7, (IX + dd)
CB dd 16	RL	(IX + dd)
CB dd 06	RLC	(IX + dd)
CB dd 1E	RR	(IX + dd)
CB dd 0E	RRC	(IX + dd)
9E dd	SBC	A, (IX + dd)
CB dd C6	SET	0, (IX + dd)
CB dd CE	SET	1, (IX + dd)
CB dd D6	SET	2, (IX + dd)
CB dd DE	SET	3, (IX + dd)
CB dd E6	SET	4, (IX + dd)
CB dd EE	SET	5, (IX + dd)
CB dd F6	SET	6, (IX + dd)
CB dd FE	SET	7, (IX + dd)
CB dd 26	SLA	(IX + dd)
CB dd 2E	SRA	(IX + dd)
CB dd 3E	SRL	(IX + dd)
96 dd	SUB	(IX + dd)
AE dd	XOR	(IX + dd)

1^{er} op - code = FD

8E dd	ADC	A, (IY + dd)
86 dd	ADD	A, (IY + dd)
09	ADD	IY, BC
19	ADD	IY, DE
29	ADD	IY, IX
39	ADD	IY, SP
A6 dd	AND	(IY + dd)
CB dd 46	BIT	0, (IY + dd)
CB dd 4E	BIT	1, (IY + dd)
CB dd 56	BIT	2, (IY + dd)
CB dd 5E	BIT	3, (IY + dd)
CB dd 66	BIT	4, (IY + dd)
CB dd 6E	BIT	5, (IY + dd)
CB dd 76	BIT	6, (IY + dd)
CB dd 7E	BIT	7, (IY + dd)
BE dd	CP	(IY + dd)
35 dd	DEC	(IY + dd)
2B	DEC	IY
E3	EX	(SP), IY
34 dd	INC	(IY + dd)
23	INC	IY
E9	JP	(IY)
77 dd	LD	(IY + dd), A
70 dd	LD	(IY + dd), B
71 dd	LD	(IY + dd), C
72 dd	LD	(IY + dd), D
73 dd	LD	(IY + dd), E
74 dd	LD	(IY + dd), H
75 dd	LD	(IY + dd), L
38 dd nn	LD	(IY + dd), n
22 nn nn	LD	(nn), IY
7E dd	LD	A, (IY + dd)
46 dd	LD	B, (IY + dd)
4E dd	LD	C, (IY + dd)
56 dd	LD	D, (IY + dd)
5E dd	LD	E, (IY + dd)
66 dd	LD	H, (IY + dd)
2A nn nn	LD	IY, (nn)
21 nn nn	LD	IY, nn
6E dd	LD	L, (IY + dd)
F9	LD	SP, IY
B6 dd	OR	(IY + dd)
E1	POP	IY
E5	PUSH	IY
CB dd 86	RES	0, (IY + dd)
CB dd 8E	RES	1, (IY + dd)
CB dd 96	RES	2, (IY + dd)
CB dd 9E	RES	3, (IY + dd)
CB dd A6	RES	4, (IY + dd)

CB dd AE	RES	5, (IY + dd)
CB dd B6	RES	6, (IY + dd)
CB dd BE	RES	7, (IY + dd)
CB dd 16	RL	(IY + dd)
CB dd 06	RLC	(IY + dd)
CB dd 1E	RR	(IY + dd)
CB dd 0E	RRC	(IY + dd)
9E dd	SBC	A, (IY + dd)
CB dd C6	SET	0, (IY + dd)
CB dd CE	SET	1, (IY + dd)
CB dd D6	SET	2, (IY + dd)
CB dd DE	SET	3, (IY + dd)
CB dd E6	SET	4, (IY + dd)
CB dd EE	SET	5, (IY + dd)
CB dd F6	SET	6, (IY + dd)
CB dd FE	SET	7, (IY + dd)
CB dd 26	SLA	(IY + dd)
CB dd 2E	SRA	(IY + dd)
CB dd 3E	SRL	(IY + dd)
96 dd	SUB	(IY + dd)
AE dd	XOR	(IY + dd)

1 ^{er} op - code = CB

46	BIT	0, (HL)
47	BIT	0, A
40	BIT	0, B
41	BIT	0, C
42	BIT	0, D
43	BIT	0, E
44	BIT	0, H
45	BIT	0, L
4E	BIT	1, (HL)
4F	BIT	1, A
48	BIT	1, B
49	BIT	1, C
4A	BIT	1, D
4B	BIT	1, E
4C	BIT	1, H
4D	BIT	1, L
56	BIT	2, (HL)
57	BIT	2, A
50	BIT	2, B
51	BIT	2, C
52	BIT	2, D
53	BIT	2, E
54	BIT	2, H
55	BIT	2, L
5E	BIT	3, (HL)

5F	BIT	3,A
58	BIT	3,B
59	BIT	3,C
5A	BIT	3,D
5B	BIT	3,E
5C	BIT	3,H
5D	BIT	3,L
66	BIT	4,(HL)
67	BIT	4,A
60	BIT	4,B
61	BIT	4,C
62	BIT	4,D
63	BIT	4,E
64	BIT	4,H
65	BIT	4,L
6E	BIT	5,(HL)
6F	BIT	5,A
68	BIT	5,B
69	BIT	5,C
6A	BIT	5,D
6B	BIT	5,E
6C	BIT	5,H
6D	BIT	5,L
76	BIT	6,(HL)
77	BIT	6,A
70	BIT	6,B
71	BIT	6,C
72	BIT	6,D
73	BIT	6,E
74	BIT	6,H
75	BIT	6,L
7E	BIT	7,(HL)
7F	BIT	7,A
78	BIT	7,B
79	BIT	7,C
7A	BIT	7,D
7B	BIT	7,E
7C	BIT	7,H
7D	BIT	7,L
86	RES	0,(HL)
87	RES	0,A
80	RES	0,B
81	RES	0,C
82	RES	0,D
83	RES	0,E
84	RES	0,H
85	RES	0,L
8E	RES	1,(HL)

8F	RES	1,A
88	RES	1,B
89	RES	1,C
8A	RES	1,D
8B	RES	1,E
8C	RES	1,H
8D	RES	1,L
96	RES	2,(HL)
97	RES	2,A
90	RES	2,B
91	RES	2,C
92	RES	2,D
93	RES	2,E
94	RES	2,H
95	RES	2,L
9E	RES	3,(HL)
9F	RES	3,A
98	RES	3,B
99	RES	3,C
9A	RES	3,D
9B	RES	3,E
9C	RES	3,H
9D	RES	3,L
A6	RES	4,(HL)
A7	RES	4,A
A0	RES	4,B
A1	RES	4,C
A2	RES	4,D
A3	RES	4,E
A4	RES	4,H
A5	RES	4,L
AE	RES	5,(HL)
AF	RES	5,A
A8	RES	5,B
A9	RES	5,C
AA	RES	5,D
AB	RES	5,E
AC	RES	5,H
AD	RES	5,L
B6	RES	6,(HL)
B7	RES	6,A
B0	RES	6,B
B1	RES	6,C
B2	RES	6,D
B3	RES	6,E
B4	RES	6,H
B5	RES	6,L
BE	RES	7,(HL)
BF	RES	7,A
B8	RES	7,B

B9	RES	7,C
BA	RES	7,D
BB	RES	7,E
BC	RES	7,H
BD	RES	7,L
16	RL	(HL)
17	RL	A
10	RL	B
11	RL	C
12	RL	D
13	RL	E
14	RL	H
15	RL	L
06	RLC	(HL)
07	RLC	A
00	RLC	B
01	RLC	C
02	RLC	D
03	RLC	E
04	RLC	H
05	RLC	L
1E	RR	(HL)
1F	RR	A
18	RR	B
19	RR	C
1A	RR	D
1B	RR	E
1C	RR	H
1D	RR	L
0E	RRC	(HL)
0F	RRC	A
08	RRC	B
09	RRC	C
0A	RRC	D
0B	RRC	E
0C	RRC	H
0D	RRC	L
C6	SET	0,(HL)
C7	SET	0,A
C0	SET	0,B
C1	SET	0,C
C2	SET	0,D
C3	SET	0,E
C4	SET	0,H
C5	SET	0,L
CE	SET	1,(HL)
CF	SET	1,A
C8	SET	1,B
C9	SET	1,C
CA	SET	1,D

CB	SET	1,E
CC	SET	1,H
CD	SET	1,L
D6	SET	2,(HL)
D7	SET	2,A
D0	SET	2,B
D1	SET	2,C
D2	SET	2,D
D3	SET	2,E
D4	SET	2,H
D5	SET	2,L
DE	SET	3,(HL)
DF	SET	3,A
D8	SET	3,B
D9	SET	3,C
DA	SET	3,D
DB	SET	3,E
DC	SET	3,H
DD	SET	3,L
E6	SET	4,(HL)
E7	SET	4,A
E0	SET	4,B
E1	SET	4,C
E2	SET	4,D
E3	SET	4,E
E4	SET	4,H
E5	SET	4,L
EE	SET	5,(HL)
EF	SET	5,A
E8	SET	5,B
E9	SET	5,C
EA	SET	5,D
EB	SET	5,E
EC	SET	5,H
ED	SET	5,L
F6	SET	6,(HL)
F7	SET	6,A
F0	SET	6,B
F1	SET	6,C
F2	SET	6,D
F3	SET	6,E
F4	SET	6,H
F5	SET	6,L
FE	SET	7,(HL)
FF	SET	7,A
F8	SET	7,B
F9	SET	7,C
FA	SET	7,D
FB	SET	7,E
FC	SET	7,H
FD	SET	7,L

26	SLA	(HL)
27	SLA	A
20	SLA	B
21	SLA	C
22	SLA	D
23	SLA	E
24	SLA	H
25	SLA	L
2E	SRA	(HL)
2F	SRA	A
28	SRA	B
29	SRA	C
2A	SRA	D
2B	SRA	E
2C	SRA	H
2D	SRA	L
3E	SRL	(HL)
3F	SRL	A
38	SRL	B
39	SRL	C
3A	SRL	D
3B	SRL	E
3C	SRL	H
3D	SRL	L

1 ^{er} op - code = ED

4A	ADC	HL, BC
5A	ADC	HL, DE
6A	ADC	HL, HL
7A	ADC	HL, SP
A9	CPD	
B9	CP	
	DR	
A1	CPI	
B1	CPIR	
46	IM	0
56	IM	1
5E	IM	2
78	IN	A,(C)
40	IN	B,(C)
48	IN	C, (C)
50	IN	D,(C)
58	IN	E,(C)
60	IN	H,(C)
68	IN	L,(C)
AA	IND	
BA	INDR	
A2	INI	
B2	INIR	

43	nn nn	LD	(nn), BC
53	nn nn	LD	(nn), DE
73		LD	(nn), SP
57		LD	A, I
5F		LD	A, R
4B	nn nn	LD	BC, (nn)
5B	nn nn	LD	DE, (nn)
47		LD	I, A
4F		LD	R, A
7B	nn nn	LD	SP, (nn)
A8		LDD	
B8		LDDR	
A0		LDI	
B0		LDIR	
44		NEG	
BB		OTDR	
B3		OTIR	
79		OUT	(C), A
41		OUT	(C), B
49		OUT	(C), C
51		OUT	(C), D
59		OUT	(C), E
61		OUT	(C), H
69		OUT	(C), L
AB		OUTD	
A3		OUTI	
4D		RETI	
45		RETN	
6F		RLD	
67		RRD	
42		SBC	HL, BC
52		SBC	HL, DE
62		SBC	HL, HL
72		SBC	HL, SP

Remarques :

Dans un op-code,

→ n représente un nombre sur 8 bits (2 digit)

→ dd représente un déplacement sur 8 bits (2 digit)

→ nn représente un nombre sur 16 bits (4 digit)

→ quand des codes n, dd ou nn apparaissent dans le code opératoire et pas dans le code hexadécimal correspondant, ils suivent le dernier code hexadécimal donné.

Par exemple :

LD BC, (nn) est codé : ED 4B nn nn

Les données de type 1 et 2 peuvent être divisées en trois sous-types en fonction du nombre d'octets supplémentaires nécessaires pour le codage complet.

1°) Un octet supplémentaire.

Par exemple, DD 09 pour ADD IX,BC.

2°) Deux octets supplémentaires.

Par exemple, DD 8E dd pour ADC A,(IX + dd).

3°) Quatre octets supplémentaires.

Par exemple, DD CB dd EE pour SET 5,(IX + dd).

Les données du type 3 sont toutes adressées sur deux octets. Par exemple, CB D6 pour SET 2,(HL).

Les données du type 4 peuvent être divisées en deux sous-types selon le nombre d'octets supplémentaires nécessaires pour le codage complet.

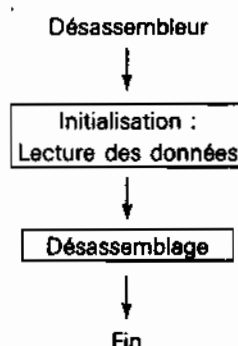
1°) Un octet supplémentaire.

Par exemple, ED 4A pour ADC HL,BC.

2°) Trois octets supplémentaires.

Par exemple, ED 43 nn nn pour LD (nnnn),BC.

Le programme que nous allons réaliser aura la structure suivante :



La phase d'initialisation mettra les codes opératoires du Z80 dans des tableaux pour faciliter leur manipulation et augmenter la vitesse d'affichage à l'écran. Nous distinguerons :

- a) les mnémoniques accessibles directement,
- b) les mnémoniques de type 1 (1^{er} octet = DD),
- c) les mnémoniques de type 2 (1^{er} octet = FD),
- d) les mnémoniques de type 3 (1^{er} octet = CB),
- e) les mnémoniques de type 4 (1^{er} octet = ED).

Les mnémoniques seront codés dans des lignes de DATA comme suit.

- a) Les mnémoniques accédés directement seront codés sur deux octets :

Longueur codage	Mnémonique
↓	↓
dans le tableau L	dans le tableau C\$

b et c) Les mnémoniques de type 1 ou 2 seront définies de trois manières différentes selon le nombre d'octets nécessaires pour qu'ils soient entièrement codés :

1 octet :

Code hexa de l'octet supplémentaire	Mnémonique
---	------------

2 octets :

Code hexa du 1 ^{er} octet supplémentaire	Début du mnémonique	Fin du mnémonique
---	------------------------	----------------------

4 octets :

Code hexa du 1 ^{er} octet supplémentaire	Code hexa du 3 ^e octet supplémentaire	Début du mnémonique	Fin du mnémonique
---	--	------------------------	----------------------

d) Les mnémoniques du type 3 sont définies par deux données :

Code hexa supplémentaire	Mnémonique
-----------------------------	------------

e) Les mnémoniques du type 4 sont définies de deux manières différentes selon que le code-op. est en une ou deux parties :

— en une partie :

Code hexa supplémentaire	Mnémonique
-----------------------------	------------

— en deux parties :

Code hexa supplémentaire	Début du mnémonique	Fin du mnémonique
-----------------------------	------------------------	----------------------

La phase désassemblage consiste à :

- lire le code hexadécimal en mémoire,
- voir s'il s'agit d'un code spécial :
- dans ce cas, activer un sous-programme de traitement spécial,

- dans le cas contraire, afficher le mnémonique correspondant.

Les mnémoniques spéciaux sont repérés dans le programme dans la partie « codes opératoires et données d'adressage » par un mnémonique numérique. Ce nombre définit le numéro du traitement spécial à effectuer.

Le programme BASIC du désassembleur est le suivant :

```

10 GOSUB 3000 'Initialisation
20 GOSUB 4000 'Désassemblage
30 END

1000 REM =====
1010 REM          Codes-opérateurs et données d'adressage
1020 REM          Codes accessibles directement
1030 REM =====
1040 DATA 1,NOP,3,"LD BC,",1,"LD (BC),A",1,INC BC,1,INC B,1,DEC B,2,"LD B,",1,RL
CA
1050 DATA 1,"EX AF,AF",1,"ADD HL,BC",1,"LD A,(BC)",1,DEC BC,1,INC C,1,DEC C,2,"
LD C,",1,PRCA
1060 DATA 3,DJNZ,3,"LD DE,",1,"LD (DE),A",1,INC DE,1,INC D,1,DEC D,2,"LD D,",1,
RL A
1070 DATA 2,JR,1,"ADD HL,DE",1,"LD A,(DE)",1,DEC DE,1,INC E,1,DEC E,2,"LD E,",1,
RRA
1080 DATA 2,"JR NZ,",3,"LD HL,",3,"5",1,INC HL,1,INC H,1,DEC H,2,"LD H,",1,DAA
1090 DATA 2,"JR Z,",1,"ADD HL,HL",3,"6",1,DEC HL,1,INC L,1,DEC L,2,"LD L,",1,CPL
1100 DATA 2,"JR NC,",3,"LD SP,",3,"7",1,INC SP,1,INC(HL),1,DEC(HL),2,"LD (HL)",1,
SCF
1110 DATA 2,"JR C,",1,"ADD HL,SP",3,"8",1,DEC SP,1,INC A,1,DEC A,2,"LD A,",1,CCF
1120 DATA 1,"LD B,B",1,"LD B,C",1,"LD B,D",1,"LD B,E",1,"LD B,H",1,"LD B,L",1,"L
D B,(HL)",1,"LD B,A"
1130 DATA 1,"LD C,B",1,"LD C,C",1,"LD C,D",1,"LD C,E",1,"LD C,H",1,"LD C,L",1,"L
D C,(HL)",1,"LD C,A"
1140 DATA 1,"LD D,B",1,"LD D,C",1,"LD D,D",1,"LD D,E",1,"LD D,H",1,"LD D,L",1,"L
D D,(HL)",1,"LD D,A"
1150 DATA 1,"LD E,B",1,"LD E,C",1,"LD E,D",1,"LD E,E",1,"LD E,H",1,"LD E,L",1,"L
D E,(HL)",1,"LD E,A"
1160 DATA 1,"LD H,B",1,"LD H,C",1,"LD H,D",1,"LD H,E",1,"LD H,H",1,"LD H,L",1,"L
D H,(HL)",1,"LD H,A"
1170 DATA 1,"LD L,B",1,"LD L,C",1,"LD L,D",1,"LD L,E",1,"LD L,H",1,"LD L,L",1,"L
D L,(HL)",1,"LD L,A"
1180 DATA 1,"LD (HL),B",1,"LD (HL),C",1,"LD (HL),D",1,"LD (HL),E",1,"LD (HL),H",
1,"LD (HL),L",1,HALT,1,"LD (HL),A"

```

```

119D DATA 1,"LD A,B",1,"LD A,C",1,"LD A,D",1,"LD A,E",1,"LD A,H",1,"LD A,L",1,"LD
D A,(HL)",1,"LD A,A"

120D DATA 1,"ADD A,B",1,"ADD A,C",1,"ADD A,D",1,"ADD A,E",1,"ADD A,H",1,"ADD A,L
",1,"ADD A,(HL)",1,"ADD A,A"

121D DATA 1,"ADC A,B",1,"ADC A,C",1,"ADC A,D",1,"ADC A,E",1,"ADC A,H",1,"ADC A,L
",1,"ADC A,(HL)",1,"ADC A,A"

122D DATA 1,SUB B,1,SUB C,1,SUB D,1,SUB E,1,SUB H,1,SUB L,1,SUB (HL),1,SUB A

123D DATA 1,"SBC A,B",1,"SBC A,C",1,"SBC A,D",1,"SBC A,E",1,"SBC A,H",1,"SBC A,L
",1,"SBC A,(HL)",1,"SBC A,A"

124D DATA 1,AND B,1,AND C,1,AND D,1,AND E,1,AND H,1,AND L,1,AND (HL),1,AND A

125D DATA 1,XOR B,1,XOR C,1,XOR D,1,XOR E,1,XOR H,1,XOR L,1,XOR (HL),1,XOR A

126D DATA 1,OR B,1,OR C,1,OR D,1,OR E,1,OR H,1,OR L,1,OR (HL),1,OR A

127D DATA 1,CP B,1,CP C,1,CP D,1,CP E,1,CP H,1,CP L,1,CP (HL),1,CP A

128D DATA 1,RET NZ,1,POP BC,2,"JP NZ,",3,"JP ",3,"CALL NZ,",1,PUSH BC,2,"ADD A,"
,1,RST 0D

129D DATA 1,RET Z,1,RET,3,"JP Z,",255,3,3,"CALL Z,",3,CALL ,2,"ADC A,",1,RST 0B

130D DATA 1,RET NC,1,POP DE,3,"JP NC,",2,"9",3,"CALL NC,",1,PUSH DE,2,SUB ,1,RST
10H

131D DATA 1,RET C,1,EXX,3,"JP C,",2,"10",3,"CALL C,",255,1,2,"SBC A,",1,RST 18H

132D DATA 1,RET PO,1,POP HL,3,"JP PO,",1,"EX (SP),HL",3,"CALL PO,",1,"PUSH HL",2
,AND ,1,RST 20H

133D DATA 1,RET PE,1,JP (HL),3,"JP PE,",1,"EX DE,HL",3,"CALL PE,",255,4,2,XOR ,1
,RST 28H

134D DATA 1,RET P,1,POP AF,3,"JP P,",1,DI,3,"CALL P,",1,PUSH AF,2,OR ,1,RST 30H

135D DATA 1,RET M,1,"LD SP,HL",3,"JP M,",1,EI,3,"CALL M,",255,2,2,CP ,1,RST 38H

136D REM -----

137D REM                                     Codes de type 1

138D REM -----

139D DATA 9,"ADD IX,BC",&19,"ADD IX,DE",&29,"ADD IX,IX",&39,"ADD IX,SP",&2B,DEC
IX,&E3,"EX (SP),IX",&23,INC IX,&E9,JP (IX),&E1,POP I
X,&E5,PUSH IX

140D DATA &BE,"ADC A,(IX+)",&86,"ADD A,(IX+)",&A6,AND (IX+),&BE,CP (IX+),&35
,DEC (IX+),&34,INC (IX+),&77,LD (IX+),A,&7D,LD
(IX+),B"

141D DATA &71,LD (IX+),C,&72,LD (IX+),D,&73,LD (IX+),E,&74,LD (IX+),H
,&75,LD (IX+),L,&7E,"LD A,(IX+)",&46,"LD B,(IX+
",&56,"LD C,(IX+)"

```



```

1420 DATA &5E,"LD D,(IX+,),&5E,"LD E,(IX+,),&66,"LD H,(IX+,),&6E,"LD L,(IX+,
),&B6,OR (IX+,),&9E,"SBC A,(IX+,),&96,SUB (IX+,),&A

E,XOR (IX+,)

1430 DATA &CB,&46,"BIT 0,(IX+,),&CB,&4E,"BIT 1,(IX+,),&CB,&56,"BIT 2,(IX+,),&
CB,&5E,"BIT 3,(IX+,),&CB,&66,"BIT 4,(IX+,),&CB,&6E

,"BIT 5,(IX+,)

1440 DATA &CB,&76,"BIT 6,(IX+,),&CB,&7E,"BIT 7,(IX+,),&CB,&86,"RES 0,(IX+,),&
CB,&8E,"RES 1,(IX+,),&CB,&96,"RES 2,(IX+,),&CB,&9E

,"RES 3,(IX+,)

1450 DATA &CB,&A6,"RES 4,(IX+,),&CB,&AE,"RES 5,(IX+,),&CB,&B6,"RES 6,(IX+,),&
CB,&BE,"RES 7,(IX+,),&CB,&16,RL (IX+,),&CB,&6,RLC (

IX+,)

1460 DATA &CB,&1E,RR (IX+,),&CB,&E,RRC (IX+,),&CB,&C6,"SET 0,(IX+,),&CB,&CE,"SE
T 1,(IX+,),&CB,&D6,"SET 2,(IX+,),&CB,&DE,"SET 3,(I
X+,),&CB,&E6,"SET 4,(IX+,)

1470 DATA &CB,&EE,"SET 5,(IX+,),&CB,&F6,"SET 6,(IX+,),&CB,&FE,"SET 7,(IX+,),&
CB,&26,SLA (IX+,),&CB,&2E,SRA (IX+,)

1480 REM -----
1490 REM                               Codes de type 2
1500 REM -----
1510 DATA &8E,"ADC A,(IX+,),&86,"ADD A,(IX+,),&9,"ADD IY,BC",&19,"ADD IY,DE",&29,"A
DD IY,IX",&39,"ADD IY,SP",&2B,DEC IY,&E3,"EX (SP),IY

",&23,INC IY,&E9,JP (IY)

1520 DATA &F9,"LD SP,IY",&E1,POP IY,&E5,PUSH IY

1530 DATA &8E,"ADC A,(IX+,),&86,"ADD A,(IX+,),&A6,"AND (IX+,),&BE,CP (IX+,),&
35,DEC (IX+,),&34,INC (IX+,),&77,LD (IX+,),A",&7D,L

D (IX+,),B"

1540 DATA &71,LD (IX+,),C",&72,LD (IX+,),D",&73,LD (IX+,),E",&74,LD (IX+,),H
",&75,LD (IX+,),L"

1550 DATA &CB,&86,"RES 0,(IX+,),&CB,&8E,"RES 1,(IX+,),&CB,&96,"RES 2,(IX+,),&
CB,&9E,"RES 3,(IX+,),&CB,&A6,"RES 4,(IX+,),&CB,&AE

,"RES 5,(IX+,)

1560 DATA &CB,&B6,"RES 6,(IX+,),&CB,&BE,"RES 7,(IX+,),&CB,&16,RL (IX+,),&CB,&6,
RLC (IX+,),&CB,&1E,&RR (IX+,),&CB,&E,&RRC (IX+,),&CB

,&C6,"SET 0,(IX+,)

1570 DATA &CB,&CE,"SET 1,(IX+,),&CB,&D6,"SET 2,(IX+,),&CB,&DE,"SET 3,(IX+,),&
CB,&E6,"SET 4,(IX+,),&CB,&EE,"SET 5,(IX+,),&CB,&F6

,"SET 6,(IX+,),&CB,&FF,"SET 7,(IX+,)

```

```

1580 DATA &CB,&26,SLA (IY+,&CB,&2E,SRA (IY+,&CB,&3E,SRL (IY+,&
1590 REM -----
1600 REM                                     Codes de type 3
1610 REM -----
1620 DATA &46,"BIT 0,(HL)",&47,"BIT 0,A",&48,"BIT 0,B",&49,"BIT 0,C",&50,"BIT 0,
D",&51,"BIT 0,E",&52,"BIT 0,H",&53,"BIT 0,L"
1630 DATA &4E,"BIT 1,(HL)",&4F,"BIT 1,A",&50,"BIT 1,B",&51,"BIT 1,C",&52,"BIT 1,
D",&53,"BIT 1,E",&54,"BIT 1,H",&55,"BIT 1,L"
1640 DATA &56,"BIT 2,(HL)",&57,"BIT 2,A",&58,"BIT 2,B",&59,"BIT 2,C",&60,"BIT 2,
D",&61,"BIT 2,E",&62,"BIT 2,H",&63,"BIT 2,L"
1650 DATA &5E,"BIT 3,(HL)",&5F,"BIT 3,A",&60,"BIT 3,B",&61,"BIT 3,C",&62,"BIT 3,
D",&63,"BIT 3,E",&64,"BIT 3,H",&65,"BIT 3,L"
1660 DATA &66,"BIT 4,(HL)",&67,"BIT 4,A",&68,"BIT 4,B",&69,"BIT 4,C",&70,"BIT 4,
D",&71,"BIT 4,E",&72,"BIT 4,H",&73,"BIT 4,L"
1670 DATA &6E,"BIT 5,(HL)",&6F,"BIT 5,A",&70,"BIT 5,B",&71,"BIT 5,C",&72,"BIT 5,
D",&73,"BIT 5,E",&74,"BIT 5,H",&75,"BIT 5,L"
1680 DATA &76,"BIT 6,(HL)",&77,"BIT 6,A",&78,"BIT 6,B",&79,"BIT 6,C",&80,"BIT 6,
D",&81,"BIT 6,E",&82,"BIT 6,H",&83,"BIT 6,L"
1690 DATA &7E,"BIT 7,(HL)",&7F,"BIT 7,A",&80,"BIT 7,B",&81,"BIT 7,C",&82,"BIT 7,
D",&83,"BIT 7,E",&84,"BIT 7,H",&85,"BIT 7,L"
1700 DATA &86,"RES 0,(HL)",&87,"RES 0,A",&88,"RES 0,B",&89,"RES 0,C",&90,"RES 0,
D",&91,"RES 0,E",&92,"RES 0,H",&93,"RES 0,L"
1710 DATA &8E,"RES 1,(HL)",&8F,"RES 1,A",&90,"RES 1,B",&91,"RES 1,C",&92,"RES 1,
D",&93,"RES 1,E",&94,"RES 1,H",&95,"RES 1,L"
1720 DATA &96,"RES 2,(HL)",&97,"RES 2,A",&98,"RES 2,B",&99,"RES 2,C",&100,"RES 2,
D",&101,"RES 2,E",&102,"RES 2,H",&103,"RES 2,L"
1730 DATA &9E,"RES 3,(HL)",&9F,"RES 3,A",&100,"RES 3,B",&101,"RES 3,C",&102,"RES 3,
D",&103,"RES 3,E",&104,"RES 3,H",&105,"RES 3,L"
1740 DATA &A6,"RES 4,(HL)",&A7,"RES 4,A",&A8,"RES 4,B",&A9,"RES 4,C",&AA,"RES 4,
D",&AB,"RES 4,E",&AC,"RES 4,H",&AD,"RES 4,L"
1750 DATA &AE,"RES 5,(HL)",&AF,"RES 5,A",&B0,"RES 5,B",&B1,"RES 5,C",&B2,"RES 5,
D",&B3,"RES 5,E",&B4,"RES 5,H",&B5,"RES 5,L"
1760 DATA &B6,"RES 6,(HL)",&B7,"RES 6,A",&B8,"RES 6,B",&B9,"RES 6,C",&BA,"RES 6,
D",&BB,"RES 6,E",&BC,"RES 6,H",&BD,"RES 6,L"
1770 DATA &BE,"RES 7,(HL)",&BF,"RES 7,A",&C0,"RES 7,B",&C1,"RES 7,C",&C2,"RES 7,
D",&C3,"RES 7,E",&C4,"RES 7,H",&C5,"RES 7,L"
1780 DATA &16,RL (HL),&17,RL A,&18,RL B,&19,RL C,&20,RL D,&21,RL E,&22,RL H,&23,
RL L
1790 DATA &6,RLC (HL),&7,RLC A,&8,RLC B,&9,RLC C,&10,RLC D,&11,RLC E,&12,RLC H,&13,
RLC L
1800 DATA &1E,RR (HL),&1F,RR A,&20,RR B,&21,RR C,&22,RR D,&23,RR E,&24,RR H,&25,
RR L

```

1810 DATA &E, RRC (HL), &F, RRC A, &8, RRC B, &9, RRC C, &A, RRC D, &B, RRC E, &C, RRC H, &D, RRC L

1820 DATA &C6, "SET 0, (HL)", &C7, "SET 0, A", &C0, "SET 0, B", &C1, "SET 0, C", &C2, "SET 0, D", &C3, "SET 0, E", &C4, "SET 0, H", &C5, "SET 0, L"

1830 DATA &CE, "SET 1, (HL)", &CF, "SET 1, A", &CB, "SET 1, B", &C9, "SET 1, C", &CA, "SET 1, D", &CB, "SET 1, E", &CC, "SET 1, H", &CD, "SET 1, L"

1840 DATA &D6, "SET 2, (HL)", &D7, "SET 2, A", &D0, "SET 2, B", &D1, "SET 2, C", &D2, "SET 2, D", &D3, "SET 2, E", &D4, "SET 2, H", &D5, "SET 2, L"

1850 DATA &DE, "SET 3, (HL)", &DF, "SET 3, A", &DB, "SET 3, B", &D9, "SET 3, C", &DA, "SET 3, D", &DB, "SET 3, E", &DC, "SET 3, H", &DD, "SET 3, L"

1860 DATA &E6, "SET 4, (HL)", &E7, "SET 4, A", &E0, "SET 4, B", &E1, "SET 4, C", &E2, "SET 4, D", &E3, "SET 4, E", &E4, "SET 4, H", &E5, "SET 4, L"

1870 DATA &EE, "SET 5, (HL)", &EF, "SET 5, A", &EB, "SET 5, B", &E9, "SET 5, C", &EA, "SET 5, D", &EB, "SET 5, E", &EC, "SET 5, H", &ED, "SET 5, L"

1880 DATA &F6, "SET 6, (HL)", &F7, "SET 6, A", &F0, "SET 6, B", &F1, "SET 6, C", &F2, "SET 6, D", &F3, "SET 6, E", &F4, "SET 6, H", &F5, "SET 6, L"

1890 DATA &FE, "SET 7, (HL)", &FF, "SET 7, A", &FB, "SET 7, B", &F9, "SET 7, C", &FA, "SET 7, D", &FB, "SET 7, E", &FC, "SET 7, H", &FD, "SET 7, L"

1900 DATA &26, SLA (HL), &27, SLA A, &20, SLA B, &21, SLA C, &22, SLA D, &23, SLA E, &24, SLA H, &25, SLA L

1910 DATA &2E, SRA (HL), &2F, SRA A, &28, SRA B, &29, SRA C, &2A, SRA D, &2B, SRA E, &2C, SRA H, &2D, SRA L

1920 DATA &3E, SRL (HL), &3F, SRL A, &38, SRL B, &39, SRL C, &3A, SRL D, &3B, SRL E, &3C, SRL H, &3D, SRL L

1930 REM -----

1940 REM Codes de type 4

1950 REM -----

1960 DATA &4A, "ADC HL, BC", &5A, "ADC HL, DE", &6A, "ADC HL, HL", &7A, "ADC HL, SP", &A9, CP D, &B9, &CPDR, &A1, CPI, &B1, CPIR, &46, IM 0, &56, IM 1, &5E, I

M 2

1970 DATA &78, "IN A, (C)", &40, "IN B, (C)", &48, "IN C, (C)", &50, "IN D, (C)", &58, "IN E, (C)", &60, "IN H, (C)", &68, "IN L, (C)"

1980 DATA &AA, IND, &BA, INDR, &A2, INI, &B2, INIR, &57, "LD A, I", &5F, "LD A, R"

1990 DATA &47, "LD I, A", &4F, "LD R, A", &AB, LDD, &BB, LDDR, &AD, LDI, &BD, LDIR, &44, NEG, &B B, OTDR

2000 DATA &B3, OTIR, &79, "OUT (C), A", &41, "OUT (C), B", &49, "OUT (C), C", &51, "OUT (C), D", &59, "OUT (C), E", &61, "OUT (C), H", &69, "OUT (C), L"

2010 DATA &AB, OUT D, &A3, OUTI, &4D, RETI, &45, RETN, &6F, RLD, &67, RRD, &42, "SBC HL, BC", &52, "SBC HL, DE", &62, "SBC HL, HL", &72, "SBC HL, SP"

2020 DATA &43, LD (,), BC, &53, LD (,), SP, &4B, "LD BC, (,)", &5B, "LD DE, (,)", &7B, "LD D SP, (,)"

```
3000 REM =====
3010 REM                      Initialisation
3020 REM =====
3030 DIM L(256),C$(256) 'Longueur et Code operatoire
3040 FOR I=0 TO 255
3050   READ L(I),C$(I)
3060 NEXT I
3070 REM -----
3080 DIM D1(10),D2(24),D4(30),D5(30),D6(248),D7(50),D8(5),E1(13),E2(13),E4(23),E
5(23)
3090 DIM C1$(10),C2$(24),C3$(24),C4$(30),C5$(30),C6$(248),C7$(50),C8$(5),C9$(5),
D1$(13),D2$(13),D3$(13),D4$(23),D5$(23)
3100 REM Lecture des donnees speciales de type 1
3110 FOR I=1 TO 10
3120   READ D1(I):READ C1$(I)
3130 NEXT I
3140 FOR I=1 TO 24
3150   READ D2(I):READ C2$(I):READ C3$(I)
3160 NEXT I
3170 FOR I=1 TO 30
3180   READ D4(I):READ D5(I):READ C4$(I):READ C5$(I)
3190 NEXT I
3200 REM -----
3210 REM Lecture des donnees speciales de type 2
3220 FOR I=1 TO 13
3230   READ E1(I):READ D1$(I)
3240 NEXT I
3250 FOR I=1 TO 13
3260   READ E2(I):READ D2$(I):READ D3$(I)
3270 NEXT I
3280 FOR I=1 TO 23
3290   READ E4(I):READ E5(I):READ D4$(I):READ D5$(I)
3300 NEXT I
```

[illegible]

```

5030 ON VAL(C$(A)) GOTO 5050,5280,5530,5640,5780,5790,5800,5810,5820,5830
5040 REM -----
5050 REM                      Traitement special No 1
5060 REM -----
5070 ADR=ADR+1:P=PEEK(ADR)
5080 IF P=&36 THEN PRINT"LD (IX+";HEX$(PEEK(ADR+1));";",";HEX$(PEEK(ADR+2)):ADR=
ADR+3:GOTO 5260
5090 IF P=&22 THEN PRINT"LD (";V=PEEK(ADR+1)+PEEK(ADR+2)*256:PRINT"#";HEX$(V);"
),IX":ADR=ADR+3:GOTO 5260
5100 IF P=&2A THEN PRINT"LD IX,(";V=PEEK(ADR+1)+PEEK(ADR+2)*256:PRINT"#";HEX$(V)
);")":ADR=ADR+3:GOTO 5260
5110 IF P=&21 THEN PRINT"LD IX,(";V=PEEK(ADR+1)+PEEK(ADR+2)*256:PRINT"#";HEX$(V):
ADR=ADR+3:GOTO 5260
5120 T=0 'Indicateur de code trouve
5130 FOR I=1 TO 10
5140   IF P=D1(I) THEN T=I
5150 NEXT I
5160 IF T<>0 THEN PRINT C1$(T):GOTO 5260
5170 FOR I=1 TO 24
5180   IF P=D2(I) THEN T=I
5190 NEXT I
5200 IF T<>0 THEN PRINT C2$(T);HEX$(PEEK(ADR+1));C3$(T):ADR=ADR+2:GOTO 5260
5210 FOR I=1 TO 30
5220   IF P=D4(I) AND PEEK(ADR+2)=D5(I) THEN T=I
5230 NEXT I
5240 IF T<>0 THEN PRINT C4$(T);HEX$(PEEK(ADR+1));C5$(T):ADR=ADR+3:GOTO 5260
5250 PRINT HEX$(PEEK(ADR))
5260 GOTO 4050
5270 REM -----
5280 REM                      Traitement special No 2
5290 REM -----
5300 ADR=ADR+1:P=PEEK(ADR)
5310 IF P=&36 THEN PRINT"LD (IY+";HEX$(PEEK(ADR+1));";",";HEX$(PEEK(ADR+2)):ADR=
ADR+3:GOTO 5490

```

```
5320 IF P=&22 THEN PRINT"LD (";V=PEEK(ADR+1)+PEEK(ADR+2)*256:PRINT"*";HEX$(V);"  
)";IY":ADR=ADR+3:GOTO 5490  
  
5330 IF P=&2A THEN PRINT"LD IY, (";V=PEEK(ADR+1)+PEEK(ADR+2)*256:PRINT"*";HEX$(V)  
;")";ADR=ADR+3:GOTO 5490  
  
5340 IF P=&21 THEN PRINT"LD IY, ";V=PEEK(ADR+1)+PEEK(ADR+2)*256:PRINT"*";HEX$(V):  
ADR=ADR+3:GOTO 5490  
  
5350 T=0 'Indicateur de code trouve  
  
5360 FOR I=1 TO 13  
  
5370 IF P=E1(I) THEN T=I  
  
5380 NEXT I  
  
5390 IF T<>0 THEN PRINT D1$(T):GOTO 5490  
  
5400 FOR I=1 TO 13  
  
5410 IF P=E2(I) THEN T=I  
  
5420 NEXT I  
  
5430 IF T<>0 THEN PRINT D2$(T);HEX$(PEEK(ADR+1));D3$(T):ADR=ADR+2:GOTO 5490  
  
5440 FOR I=1 TO 23  
  
5450 IF P=E4(I) AND PEEK(ADR+2)=E5(I) THEN T=I  
  
5460 NEXT I  
  
5470 IF T<>0 THEN PRINT D4$(T);HEX$(PEEK(ADR+1));D5$(T):ADR=ADR+3:GOTO 5490  
  
5480 PRINT HEX$(PEEK(ADR))  
  
5490 GOTO 4050  
  
5500 REM -----  
  
5510 REM Traitement special No 3  
  
5520 REM -----  
  
5530 ADR=ADR+1:P=PEEK(ADR):T=0 'Initialisation  
  
5540 FOR I=1 TO 248  
  
5550 IF P=D6(I) THEN T=I  
  
5560 NEXT I  
  
5570 IF T<>0 THEN PRINT C6$(T):GOTO 5590  
  
5580 PRINT HEX$(PEEK(T))  
  
5590 GOTO 4050  
  
5600 REM -----  
  
5610 REM Traitement special No 4  
  
5620 REM -----
```

```

5630 REM Traitement special No 4
5640 ADR=ADR+1:P=PEEK(ADR):T=0 'Initialisation
5650 FOR I=1 TO 50
5660   IF P=D7(I) THEN T=I
5670 NEXT I
5680 IF T<>0 THEN PRINT C7$(T):ADR=ADR+1:GOTO 5740
5690 FOR I=1 TO 5
5700   IF P=D8(I) THEN T=I
5710 NEXT I
5720 IF T<>0 THEN PRINT C8$(I):V=PEEK(ADR+1)+PEEK(ADR+2)*256:PRINT HEX$(V):C9$(
T):ADR=ADR+3:GOTO 5740
5730 PRINT HEX$(P)
5740 GOTO 4050
5750 REM -----
5760 REM          Traitements speciaux No 5 a 10
5770 REM -----
5780 PRINT"LD (";ADR=ADR+1:A=PEEK(ADR):ADR=ADR+1:A=A+PEEK(ADR)*255:ADR=ADR+1:PR
INT"#" ;HEX$(A);"),HL":GOTO 4050
5790 PRINT"LD HL, (";ADR=ADR+1:A=PEEK(ADR):ADR=ADR+1:A=A+PEEK(ADR)*255:ADR=ADR+1
:PRINT"#" ;HEX$(A);")":GOTO 4050
5800 PRINT"LD (";ADR=ADR+1:A=PEEK(ADR):ADR=ADR+1:A=A+PEEK(ADR)*255:ADR=ADR+1:PR
INT"#" ;HEX$(A);"),A":GOTO 4050
5810 PRINT"LD A, (";ADR=ADR+1:A=PEEK(ADR):ADR=ADR+1:A=A+PEEK(ADR)*255:ADR=ADR+1:
PRINT"#" ;HEX$(A);")":GOTO 4050
5820 PRINT"OUT (";ADR=ADR+1:A=PEEK(ADR):ADR=ADR+1:A=A+PEEK(ADR)*255:ADR=ADR+1:P
RINT"#" ;HEX$(A);"),A":GOTO 4050
5830 PRINT"IN A, (";ADR=ADR+1:A=PEEK(ADR):ADR=ADR+1:A=A+PEEK(ADR)*255:ADR=ADR+1:
PRINT"#" ;HEX$(A);")":GOTO 4050

```

Lignes 10 à 30 : Programme principal.

Lignes 1000 à 2020 : Données.

Lignes 3000 à 3440 : Initialisation.

Lignes 4000 à 4140 : Désassemblage.

Lignes 5000 à 5830 : Traitements spéciaux.

9/2.2

L'assembleur

Qu'est-ce qu'un assembleur ?

C'est un programme qui transforme des codes mnémoniques en codes hexadécimaux exécutables par l'ordinateur.

Un Assembleur classique est composé :

- d'un programme de traitement de textes pour saisir le programme (un programme écrit en codes mnémoniques sera appelé programme source par la suite).
- du compilateur lui-même qui fait la conversion : texte → code exécutable.

Nous allons étudier ici un Assembleur complet qui possède :

- un programme de saisie de texte en mode ligne,
- un compilateur mono-passe sans gestion d'étiquettes.

Cette étude précède une autre, beaucoup plus vaste, qui sera faite par la suite et qui consistera à créer un compilateur de langage évolué du genre BASIC ou PASCAL.

I. Programme de saisie de texte

Vous qui utilisez l'Assembleur couramment (sinon, reportez-vous au chapitre 2 de la partie 4) n'êtes pas sans savoir que ce langage peut rendre de grands services. Le BASIC qui fonctionne sur les CPC est très performant, et, combiné avec quelques routines écrites en Assembleur (quand il y a un problème de vitesse d'exécution), vous arriverez sans mal à traiter la plupart des problèmes qui pourront se présenter.

Pour ne pas avoir à manipuler des codes en hexadécimal, un langage de plus haut niveau a été créé. Il s'agit du *langage d'assemblage*. Ce langage est formé de mots-clés longs de deux à quatre lettres, qui sont à rapprocher des mots équivalents en Anglais. Reportez-vous en au Chapitre 2.3 Partie 4 pour avoir la liste de ces mots-clés.

Pour qu'un programme tapé dans ce langage soit exécutable, il faudra le convertir en codes machines. C'est le rôle du compilateur Assembleur.

Mais il faudra aussi le saisir. Pour cela, nous allons développer un programme de saisie ultra-simple qui comprend les fonctions élémentaires suivantes :

- Saisie de texte, ligne par ligne,
- Affichage sur écran d'une partie ou de la totalité du texte saisi,
- Impression du texte entré sur une imprimante,
- Suppression d'une ligne de texte,
- Insertion d'une ligne de texte.

1°) Saisie de texte

Nous allons stocker les lignes entrées dans le tableau A\$. Il est dimensionné à 200 lignes (Ligne 1030 dans le programme). Si cela ne vous convient pas, vous pouvez augmenter cette dimension dans les limites de la place mémoire RAM disponible. Il vous faudra également modifier de la même manière le tableau GC\$ qui est dimensionné sur la même ligne. Le tableau GC\$ contient le code hexadécimal généré par le compilateur.

Chaque ligne est précédée d'un numéro de ligne et sera repérée grâce à ce numéro :

- si elle doit être effacée,
- si l'on doit insérer une ligne avant ou après,
- pour sortir des listings sur imprimante, etc.

Cet éditeur de ligne vous offre la possibilité d'utiliser la touche DEL pour corriger une éventuelle erreur de saisie sur la ligne courante.

Si la ligne sur laquelle vous voulez écrire n'est pas vierge, le message

« Ligne occupée »

« Etes-vous sur (O/N) »

apparaîtra à l'écran. Si vous répondez O, la ligne qui portait le même numéro que la ligne courante sera remplacée par la ligne courante.

2°) Affichage sur l'écran du texte saisi

Vous pourrez lister sur l'écran une ou plusieurs lignes, en donnant les numéros de première et dernière ligne à lister.

Quand cette option est sélectionnée, le message suivant apparaît à l'écran :

de ---- entrez le premier numero de ligne,

a ---- entrez le dernier numero de ligne.

3°) Impression du texte entré sur une imprimante

Cette option est comparable à la précédente, à ceci près que le texte sera imprimé et non pas affiché sur l'écran.

Il vous faut donner les premier et dernier numéro de ligne à lister en répondant aux questions suivantes :

Impression a partir de ---- entrez le premier numero de ligne
jusqu'a ---- entrez le dernier numero de ligne

4°) Suppression d'une ligne de texte

La ligne à supprimer est repérée par son numéro de ligne. Vous devez indiquer ce numéro après le message :

Ligne a supprimer ---- entrez le numéro de ligne à supprimer

La ligne concernée est alors affichée à l'écran, précédée de son numéro de ligne.

La question « **Etes-vous sur (O/N)** » vous permet d'éviter de supprimer une ligne par erreur : Si vous avez sélectionné une mauvaise ligne, répondez N, et la ligne ne sera pas effacée.

5°) Insertion d'une ligne de texte

Une ligne peut être insérée n'importe où entre le premier numéro de ligne et le dernier numéro de ligne.

Sélectionnez l'option correspondante dans le menu (option 3), puis donnez le numéro de la ligne après laquelle il faudra insérer la ligne que vous allez saisir. Pour cela, répondez à la question suivante :

Insertion apres la ligne ---- en donnant le numero de la ligne
après lequel doit se faire l'insertion

Le programme de saisie de texte est inséré dans un programme complet d'édition/compilation/sauvegarde. Il est accessible à partir d'un menu général (souvent appelé SHELL dans la littérature informatique), et occupe les lignes 3000 à 3920 dans le listing général :

Lignes 3000 à 3180 : Menu de l'éditeur de lignes,

Lignes 3190 à 3280 : Liste sur écran,

Lignes 3290 à 3450 : Suppression d'une ligne,

Lignes 3460 à 3640 : Insertion d'une ligne de programme,

Lignes 3650 à 3810 : Edition du programme sur l'écran,

Lignes 3820 à 3920 : Impression du programme entré sur une imprimante.

II. Compilateur

A. DEFINITION

Un abus de langage est souvent fait par les informaticiens pressés.

Pour eux, le terme « Assembleur » désigne aussi bien :

- le langage d'assemblage composé de codes-opérateurs,
- le compilateur qui transforme les codes opérateurs en binaire.

Pour éviter toute confusion, nous emploierons les termes suivants :

Assembleur : Compilateur qui transforme les codes opérateurs en langage exécutable par le microprocesseur.

Langage d'assemblage : Langage composé de codes opérateurs. Ce langage a été créé pour faciliter la programmation (il est, en effet, plus facile de manipuler des mots-clés que des nombres, même s'ils sont exprimés en hexadécimal !).

Avant de détailler le fonctionnement de l'Assembleur que nous allons développer, il convient de rappeler quelques notions fondamentales à propos des compilateurs.

B. RAPPELS

Comme nous l'avons vu au Chapitre 2.1, un compilateur est un programme qui permet de transformer les mots-clés d'un langage en codes exécutables par le microprocesseur, donc en binaire. Le microprocesseur utilisé dans les CPC est un Z80 qui manipule des données de 8 bits. Pour plus de commodité, nous dirons que le travail du compilateur consistera à traduire des mots-clés en données hexadécimales codées sur 8 bits.

Nous venons de soulever un des problèmes majeurs qui apparaît lorsque l'on s'attaque à l'écriture d'un compilateur, à savoir l'analyse des phrases entrées. Cette recherche de mots-clés dans les phrases entrées est souvent appelée *analyse syntaxique*.

Le problème est simple quand il s'agit d'un langage pauvre comme l'Assembleur. Mais il se complique très vite lorsque l'on désire créer un compilateur de langage évolué. Un tel compilateur sera étudié ultérieurement.

ANALYSE DES FONCTIONS D'UN COMPILATEUR DE LANGAGE D'ASSEMBLAGE

Cette analyse va être faite de la manière suivante.

Nous allons considérer que le compilateur est une boîte noire dans laquelle arrivent des informations (entrées), de laquelle sortent des informations (sorties). Nous allons étudier successivement :

- 1) Les entrées,
- 2) Les sorties,
- 3) Le traitement à appliquer aux entrées pour les transformer en sorties désirées.

1 - ENTREES

Si nous analysons le langage d'assemblage, nous voyons qu'un mot-clé se décompose en deux parties :

- une instruction (par exemple, ADD, LD, XOR, etc.),
- une opérande (par exemple, IX, HL ou L ou encore (BC), A).

L'instruction permet de définir le type d'opération qui va être faite. Alors que l'opérande définit le ou les registres, l'adresse de la ou des mémoires qui vont participer à cette opération. L'analyse syntaxique portera donc sur deux entités :

- l'instruction que nous appellerons « 1^{er} Op-code » par la suite,
- l'opérande que nous appellerons « 2^e Op-code » par la suite.

A chaque 1^{er} Op-code peut correspondre 1 ou plusieurs 2^e Op-code.

Par exemple, pour le 1^{er} Op-code EX, il existe cinq 2^e Op-code qui sont :

(SP),HL

(SP),IX

(SP),IY

AF,AF'

DE,HL

pour former les instructions complètes :

EX (SP),HL

EX (SP), IX

EX (SP),IY

EX AF,AF'

EX DE,HL

Par contre, le 1^{er} Op-code CPD ne possède qu'un deuxième Op-code de longueur nulle : En effet, l'instruction CPD se suffit à elle-même, et il n'est pas nécessaire d'indiquer sur quel(s) registre(s) ont porté les manipulations. Dans ce cas, on parle d'adressage implicite (Voir Chapitre 2.2 Partie 4, Les modes d'adressage).

Il apparaît donc le besoin de donner le nombre de 2^e Op-code pour chaque 1^{er} Op-code.

C'est la démarche qui a été employée pour définir les entités manipulées par le compilateur. Sur le listing BASIC, vous pourrez remarquer le codage de ces trois types de données :

Lignes 7000 à 7040 : Premier Op-code.

Lignes 8000 à 8030 : Nombre de 2^e Op-code pour chaque 1^{er} Op-code.

Lignes 9000 à 9390 : Seconds Op-codes.

Les données manipulées par le compilateur sont maintenant définies. Reste à décrire l'ensemble des codes qui pourront être générés par le compilateur.

2 - SORTIES

A chaque couple (1^{er} Op-code, 2^e Op-code) correspond(ent) un ou plusieurs codes à générer. Ayant défini l'ensemble des 2^e Op-code pour chaque 1^{er} Op-code, il suffit d'associer le(s) code(s) à générer à chaque 2^e Op-code pour passer en revue tous les codes possibles. Les codes générés occupent les lignes 10000 à 10320 du listing général. Ils sont définis de la manière suivante :

- un premier nombre donne le nombre de code(s) généré(s),
- les codes générés suivent cette première donnée.

Remarque :

Un « XX » dans un code généré indique que ce code sera contenu de manière implicite dans la phrase à compiler, et sera remplacé à la compilation.

3 - TRAITEMENTS

Les entrées (1^{er} Op-code, 2^e Op-code) et les sorties (Codes générés) sont maintenant définis. Il reste à décrire le traitement à appliquer aux entrées pour les transformer en les sorties désirées.

• 1^{re} opération

Il est nécessaire d'identifier le 1^{er} Op-code dans chaque phrase. Ce 1^{er} Op-code pourra être :

- soit le code d'implantation en mémoire « ORG »,
- soit un des 67 Op-codes du Z80.

Le code ORG (Origine) définit l'adresse à laquelle sera implanté le programme en mémoire RAM pour y être exécuté. Ce code est une directive d'assemblage : il ne génère aucun code exécutable, mais sert simplement au compilateur à planter le code généré à la bonne place en mémoire.

La détection du code « ORG » est faite ligne 4080.

La détection d'un des Op-code du Z80 est faite entre les lignes 4090 et 4140.

Si le premier Op-code n'est ni « ORG » ni un des Op-codes du Z80, une erreur est générée :

« Erreur ligne XXXX : Op-code inconnu »

• 2^e opération

Le premier Op-code identifié, il faut voir si l'association 1^{er} Op-code/2^e Op-code est correcte. Pour cela, nous calculons le déplacement à effectuer

dans la liste des 2^e Op-codes pour se situer sur le premier couple (1^{er} Op-code/2^e Op-code) validé. Ce calcul est fait entre les lignes 4160 et 4180.

Certains 2^e Op-codes contiennent une valeur numérique qui sera incorporée dans le code généré. Ces codes sont identifiés dans un sous-programme qui est appelé en ligne 4190.

Le sous-programme d'identification consiste à extraire la ou les donnée(s) numérique(s) présent(es) dans le 2^e Op-code et à la (les) stocker dans une (des) variable(s).

- 3^e opération

- Enfin, le code est généré entre les lignes 4200 et 4260. Il est affiché sur l'écran ou envoyé vers l'imprimante en ligne 4260. Pour repasser au menu général, appuyez sur une touche (ligne 4280).

III. Entrées / Sorties sur disquette

Pour clore ce programme de compilation, deux sous-programmes ont été développés :

— Le premier permet de sauvegarder un programme source (le listing en langage d'assemblage) ou un programme objet (le code généré qui pourra être exécuté par la suite). Ce sous-programme occupe les lignes 6000 à 6210.

— Le second permet de charger en mémoire un programme source qui a été sauvegardé par le premier sous-programme. Il occupe les lignes 5000 à 5110.

Les diverses parties du listing sont les suivantes :

```
10 GOSUB 1000 'Lecture des donnees
20 GOSUB 2010 'Menu principal: SHELL
30 END

1000 REM =====
1010 REM Lecture des donnees
1020 REM =====
1030 DIM T1$(67),T(67),T2$(696),CB$(696),A$(200),GC$(200)
1040 FOR I=1 TO 67
1050   READ T1$(I) 'Lecture des 1ers Op-codes
1060 NEXT I
1070 FOR I=1 TO 67
1080   READ T(I) 'Lecture du nombre de 2eme Op-code pour chaque Op-code
```

```
1090 NEXT I
1100 FOR I=1 TO 67
1110   FOR J=1 TO T(I)
1120     K=K+1:READ T2$(K) 'Lecture des seconds Op-Codes
1130   NEXT J
1140 NEXT I
1150 FOR I=1 TO 696
1160   READ A:A$=CHR$(A+48)
1170   FOR J=1 TO A
1180     READ B$:A$=A$+B$
1190   NEXT J
1200   CG$(I)=A$ '1 Code genere complet
1210 NEXT I
1220 RETURN

2000 REM =====
2010 REM Shell
2020 REM =====
2030 MODE 2
2040 PRINT"Voulez-vous : "
2050 PRINT"1) Entrer dans l'editeur de lignes"
2060 PRINT"2) Compiler un programme source present en memoire"
2070 PRINT"3) Charger en memoire un programme source"
2080 PRINT"4) Sauvegarder sur disquette un programme source ou objet"
2090 PRINT"5) Sortir de l'assembleur"
2100 PRINT:INPUT "Votre choix ";C
2110 IF C<1 OR C>5 THEN PRINT CHR$(7):GOTO 2010
2120 ON C GOSUB 3010,4010,5010,6010,30
2130 GOTO 2010
2140 RETURN

3000 REM =====
3010 REM Editeur de lignes
3020 REM =====
```



```
3030 MODE 2
3040 I=0
3050 REM Shell
3060 REM
3070 PRINT"1) Listage sur ecran"
3080 PRINT"2) Suppression d'une ligne"
3090 PRINT"3) Insertion d'une ligne"
3100 PRINT"4) Edition a l'ecran"
3110 PRINT"5) Ecriture du listing sur imprimante"
3120 PRINT"6) Sortie de l'editeur":PRINT
3130 INPUT"Votre choix";C
3140 IF C<1 OR C>6 THEN 3160
3150 IF C=6 THEN RETURN
3160 ON C GOSUB 3200,3300,3470,3660,3830,3180
3170 GOTO 3050
3180 RETURN
3190 REM -----
3200 REM Liste sur ecran
3210 REM -----
3220 INPUT"de";DE:INPUT"a";A
3230 '
3240 FOR I=DE TO A
3250   PRINT I,A$(I)
3260 NEXT I
3270 '
3280 RETURN
3290 REM -----
3300 REM Supprime une ligne
3310 REM -----
3320 INPUT"Ligne a supprimer";L
3330 IF L>MX OR L=0 THEN 3450
3340 PRINT L;A$(L)
```

```
3350 INPUT"Etes-vous sur (O/N)";R$
3360 IF R$<>"O" THEN 3450
3370 '-----
3380 'Suppression
3390 '-----
3400 FOR I=L TO MX
3410   A$(I)=A$(I+1)
3420 NEXT I
3430 A$(MX)="";MX=MX-1
3440 '
3450 RETURN
3460 REM -----
3470 REM Insere une ligne
3480 REM -----
3490 INPUT"Insertion apres la ligne";L
3500 IF L>MX OR L=0 THEN 3640
3510 PRINT:PRINT L+1;:B$=""
3520 A$=INKEY$:IF A$="" THEN 3520
3530 IF ASC(A$)=13 THEN 3560
3540 IF ASC(A$)=127 THEN PRINT CHR$(8);" ";CHR$(8);:B$=LEFT$(B$,LEN(B$)-1):GOTO 3520
3550 PRINT A$;:B$=B$+A$:GOTO 3520
3560 '-----
3570 'Insertion
3580 '-----
3590 FOR I=MX TO L+2 STEP -1
3600   A$(I)=A$(I-1)
3610 NEXT I
3620 A$(L+1)=B$:MX=MX+1
3630 '
3640 PRINT:RETURN
```

```
3650 REM -----
3660 REM Edition sur l'ecran
3670 REM -----
3680 INPUT "Ligne";L
3690 IF L>=MX THEN 3720
3700 PRINT:PRINT "Ligne occupee":INPUT "Etes-vous sur (O/N)";R$
3710 IF R$="N" THEN 3810
3720 E=1:I=L
3730 A$(I)="":PRINT:PRINT I;
3740 A$=INKEY$:IF A$="" THEN 3740
3750 IF ASC(A$)=13 THEN 3780
3760 IF ASC(A$)=127 THEN PRINT CHR$(8);" ";CHR$(8);:A$(I)=LEFT$(A$(I),LEN(A$(I))-1):GOTO 3740
3770 PRINT A$;:A$(I)=A$(I)+A$:GOTO 3740
3780 IF LEN(A$(I))=0 THEN 3810
3790 I=I+1:IF I>MX THEN MX=I
3800 L=L+1:GOTO 3690
3810 PRINT:RETURN
3820 REM -----
3830 REM Impression
3840 REM -----
3850 INPUT "Impression a partir de";DE
3860 INPUT "jusqu'a";A
3870 IF A>MX THEN A=MX
3880 IF DE=0 THEN DE=1
3890 FOR I=DE TO A
3900   PRINT#B,I;": ";A$(I)
3910 NEXT I
3920 RETURN
4000 REM =====
4010 REM Compilation du programme en memoire
4020 REM =====
```

```

4030 INPUT "Sortie sur imprimante ? (O/N)";R$
4040 IF R$="O" THEN DD=8 ELSE DD=0
4050 FOR I=1 TO MX-1
4060   PR=0:K=0:M=0 'Initialisation
4070   CC$=A$(I) 'Ligne courante
4080   II=INSTR(1,CC$,"ORG");IF II<>0 THEN PRINT #DD,TAB(17);CC$:ADR=VAL(MID$(CC$,II+4,LEN(CC$)-II)):SADR=ADR:GOTO 4270
4090   FOR J=1 TO 67
4100     IF INSTR(1,CC$,T1$(J))<>0 THEN K=J 'No Op-code
4110   NEXT J
4120   IF K=0 THEN PRINT "Erreur ligne ";I;": Op-Code inconnu":GOTO 4270
4130   '
4140   ' A ce niveau, un Op-Code est trouve
4150   '
4160   FOR L=1 TO K-1
4170     PR=PR+T(L) '1er indice de la 2eme partie de l'Op-Code
4180   NEXT L
4190   GOSUB 4320 'Test code a generation non directe
4200   FOR L=PR+1 TO PR+T(K)
4210     IF INSTR(LEN(T1$(K))+1,CC$,T2$(L))<>0 OR T2$(L)="" THEN M=L
4220   NEXT L
4230   IF M=0 THEN PRINT "Erreur ligne ";I;": 2eme partie de l'Op-code incorrecte ":GOTO 4270
4240   C$=CG$(M):IF V1$<>"" AND V2$="" THEN P=INSTR(1,CG$(M),"XX"):C$=MID$(CG$(M),1,P-1)+V1$
4250   IF V1$<>"" AND V2$<>"" THEN P1=INSTR(1,CG$(M),"XX"):C$=MID$(CG$(M),1,P1-1)+V1$+V2$
4260   GC$(I)=C$:PRINT #DD,HEX$(ADR);"  "RIGHT$(C$,LEN(C$)-1);SPC(11-LEN(GC$(I)));A$(I):ADR=ADR+VAL(LEFT$(GC$(I),1))
4270 NEXT I
4280 PRINT:PRINT "Appuyez sur une touche"
4290 A$=INKEY$:IF A$="" THEN 4290
4300 RETURN

```

```

8000 REM =====
8010 REM Nombre de 2eme Op-code pour chaque Op-code
8020 REM =====
8030 DATA 15,23,11,80,9,1,11,1,1,1,1,1,1,16,1,1,1,5,1,1,3,8,16,1,1,1,1,12,5,132,
1,1,1,1,1,1,11,1,1,8,1,1,6,6,80,9,1,1,10,1,10,1,1,10
,1,10,1,1,8,15,1,80,10,10,10,11,11
9000 REM =====
9010 REM Second Op-code
9020 REM =====
9030 DATA "A,(HL)","A,(IX+d1)","A,(IY+d1)","A,A","A,B","A,C","A,D","A,E","A,H","
A,L","A,d1","HL,BC","HL,DE","HL,HL","HL,SP"
9040 DATA "A,(HL)","A,(IX+d1)","A,(IY+d1)","A,A","A,B","A,C","A,D","A,E","A,H","
A,L","A,d1","HL,BC","HL,DE","HL,HL","HL,SP","IX,BC",
"IX,DE","IX,IX","IX,SP"
9050 DATA "IY,BC","IY,DE","IY,IY","IY,SP"
9060 DATA (HL),(IX+d1),(IY+d1),A,B,C,D,E,H,L,d1,"O,(HL)","O,(IX+d1)","O,(IY+d1)",
"O,A","O,B","O,C","O,D","O,E","O,H","O,L"
9070 DATA "1,(HL)","1,(IX+d1)","1,(IY+d1)","1,A","1,B","1,C","1,D","1,E","1,H","
1,L"
9080 DATA "2,(HL)","2,(IX+d1)","2,(IY+d1)","2,A","2,B","2,C","2,D","2,E","2,H","
2,L"
9090 DATA "3,(HL)","3,(IX+d1)","3,(IY+d1)","3,A","3,B","3,C","3,D","3,E","3,H","
3,L"
9100 DATA "4,(HL)","4,(IX+d1)","4,(IY+d1)","4,A","4,B","4,C","4,D","4,E","4,H","
4,L"
9110 DATA "5,(HL)","5,(IX+d1)","5,(IY+d1)","5,A","5,B","5,C","5,D","5,E","5,H","
5,L"
9120 DATA "6,(HL)","6,(IX+d1)","6,(IY+d1)","6,A","6,B","6,C","6,D","6,E","6,H","
6,L"
9130 DATA "7,(HL)","7,(IX+d1)","7,(IY+d1)","7,A","7,B","7,C","7,D","7,E","7,H","
7,L"
9140 DATA "C,d2","M,d2","NC,d2","d2","NZ,d2","P,d2","PE,d2","PO,d2","Z,d2",,(HL),
,(IX+d1),(IY+d1),A,B,C,D,E,H,L,d1,,,,,
9150 DATA (HL),(IX+d1),(IY+d1),A,B,BC,C,D,DE,E,H,HL,IX,IY,L,SP,,d1,,"(SP),HL","(
SP),IX","(SP),IY","AF,AF'",,"DE,HL",,,O,1,2,"A,(C)",,
A,(d1)","B,(C)"
9160 DATA "C,(C)","D,(C)","E,(C)","H,(C)","L,(C)",(HL),(IX+d1),(IY+d1),A,B,BC,C,
D,DE,E,H,HL,IX,IY,L,SP,,,,(HL),(IX),(IY),"C,d2"

```

```

9170 DATA "M,d2","NC,d2",d2,"NZ,d2","P,d2","PE,d2","PO,d2","Z,d2","C,d1",d1,"NC,
d1","NZ,d1","Z,d1","(BC),A","(DE),A","(HL),A","(HL),
B","(HL),C"

9180 DATA "(HL),D","(HL),E","(HL),H","(HL),L","(HL),d1","(IX+d1),A","(IX+d1),B",
"(IX+d1),C","(IX+d1),D","(IX+d1),E","(IX+d1),H","(IX
+d1),L","(IX+d1),d1"

9190 DATA "(IX+d1),A","(IX+d1),B","(IX+d1),C","(IX+d1),D","(IX+d1),E","(IX+d1),H",
"(IX+d1),L","(IX+d1),d1"

9200 DATA "(d2),A","(d2),BC","(d2),DE","(d2),HL","(d2),IX","(d2),IY","(d2),SP",
A,(BC),"A,(DE)","A,(HL)","A,(IX+d1)","A,(IX+d1)"

9210 DATA "A,(d2)","A,A","A,B","A,C","A,D","A,E","A,H","A,I","A,L","A,d1","A,R",
"B,(HL)","B,(IX+d1)","B,(IX+d1)","B,A","B,B","B,C",
B,D,"B,E","B,H","B,L","B,d1"

9220 DATA "BC,(d2)","BC,d2","C,(HL)","C,(IX+d1)","C,(IX+d1)","C,A","C,B","C,C",
C,D,"C,E","C,H","C,L","C,d1","D,(HL)","D,(IX+d1)",
D,(IX+d1)"

9230 DATA "D,A","D,B","D,C","D,D","D,E","D,H","D,L","D,d1","DE,(d2)","DE,d2","E,
(HL)","E,(IX+d1)","E,(IX+d1)","E,A","E,B","E,C","E,D",
"E,E","E,H","E,L","E,d1"

9240 DATA "H,(HL)","H,(IX+d1)","H,(IX+d1)","H,A","H,B","H,C","H,D","H,E","H,H",
H,L,"H,d1","HL,(d2)","HL,d2","I,A","IX,(d2)","IX,d2",
"IX,(d2)","IX,d2"

9250 DATA "L,(HL)","L,(IX+d1)","L,(IX+d1)","L,A","L,B","L,C","L,D","L,E","L,H",
L,L,"L,d1","R,A","SP,(d2)","SP,HL","SP,IX","SP,IY",
"SP,d2",,,,,,

9260 DATA (HL),(IX+d1),(IX+d1),A,B,C,D,E,H,L,d1,,,"(C),A","(C),B","(C),C","(C),D",
,"(C),E","(C),H","(C),L","(d1),A",,AF,BC,DE,HL,IX,
IY

9270 DATA AF,BC,DE,HL,IX,IY,"O,(HL)","O,(IX+d1)","O,(IX+d1)","O,A","O,B","O,C",
O,D,"O,E","O,H","O,L","1,(HL)","1,(IX+d1)","1,(IX+d1)",
1),"1,A","1,B","1,C"

9280 DATA "1,D","1,E","1,H","1,L","2,(HL)","2,(IX+d1)","2,(IX+d1)","2,A","2,B",
2,C,"2,D","2,E","2,H","2,L"

9290 DATA "3,(HL)","3,(IX+d1)","3,(IX+d1)","3,A","3,B","3,C","3,D","3,E","3,H",
3,L,"4,(HL)","4,(IX+d1)","4,(IX+d1)","4,A","4,B","4",
,C,"4,D","4,E","4,H","4,L"

9300 DATA "5,(HL)","5,(IX+d1)","5,(IX+d1)","5,A","5,B","5,C","5,D","5,E","5,H",
5,L,"6,(HL)","6,(IX+d1)","6,(IX+d1)","6,A","6,B","6

```

,C", "6,D", "6,E", "6,H", "6,L"

9310 DATA "7,(HL)", "7,(IX+d1)", "7,(IY+d1)", "7,A", "7,B", "7,C", "7,D", "7,E", "7,H", "7,L", "C,M,NC,NZ,P,PE,PO,Z,,(HL),(IX+d1),(IY+d1),A,B

,C,D,E,H,L,

9320 DATA (HL),(IX+d1),(IY+d1),A,B,C,D,E,H,L,,(HL),(IX+d1),(IY+d1),A,B,C,D,E,H,L,,(HL),(IX+d1),(IY+d1),A,B,C,D,E,H,L,,

9330 DATA 0,8,10 H,18 H,20 H,28 H,30 H,38 H,"A,(HL)", "A,(IX+d1)", "A,(IY+d1)", "A,A", "A,B"

9340 DATA "A,C", "A,D", "A,E", "A,H", "A,L", "A,d1", "HL,BC", "HL,DE", "HL,HL", "HL,SP", "O,(HL)", "O,(IX+d1)", "O,(IY+d1)", "O,A", "O,B", "O,C", "

O,D", "O,E", "O,H", "O,L"

9350 DATA "1,(HL)", "1,(IX+d1)", "1,(IY+d1)", "1,A", "1,B", "1,C", "1,D", "1,E", "1,H", "1,L", "2,(HL)", "2,(IX+d1)", "2,(IY+d1)", "2,A", "2,B", "2

,C", "2,D", "2,E", "2,H", "2,L"

9360 DATA "3,(HL)", "3,(IX+d1)", "3,(IY+d1)", "3,A", "3,B", "3,C", "3,D", "3,E", "3,H", "3,L", "4,(HL)", "4,(IX+d1)", "4,(IY+d1)", "4,A", "4,B", "4

,C", "4,D", "4,E", "4,H", "4,L"

9370 DATA "5,(HL)", "5,(IX+d1)", "5,(IY+d1)", "5,A", "5,B", "5,C", "5,D", "5,E", "5,H", "5,L", "6,(HL)", "6,(IX+d1)", "6,(IY+d1)", "6,A", "6,B", "6

,C", "6,D", "6,E", "6,H", "6,L"

9380 DATA "7,(HL)", "7,(IX+d1)", "7,(IY+d1)", "7,A", "7,B", "7,C", "7,D", "7,E", "7,H", "7,L", (HL),(IX+d1),(IY+d1),A,B,C,D,E,H,L,(HL),(IX+d1)

, (IY+d1), A,B,C,D,E,H,L

9390 DATA (HL),(IX+d1),(IY+d1),A,B,C,D,E,H,L,(HL),(IX+d1),(IY+d1),A,B,C,D,E,H,L,d1,(HL),(IX+d1),(IY+d1),A,B,C,D,E,H,L,d1

10000 REM =====

10010 REM Code -genre

10020 REM =====

10030 DATA 1,8E,3,DD,8E,XX,3,FD,8E,XX,1,8F,1,88,1,89,1,8A,1,8B,1,8C,1,8D,2,CE,XX,2,ED,4A,2,ED,5A,2,ED,6A,2,ED,7A,1,86,3,DD,86,XX,3,F

D,86,XX

10040 DATA 1,87,1,80,1,81,1,82,1,83,1,84,1,85,2,C6,XX,1,09,1,19,1,29,1,39,2,DD,09,2,DD,19,2,DD,29,2,DD,39,2,FD,09,2,FD,19,2,FD,29,2,

FD,39,1,A6,3,DD,A6,XX

10050 DATA 3,FD,A6,XX,1,A7,1,A0,1,A1,1,A2,1,A3,1,A4,1,A5,2,E6,XX,2,CB,46,4,DD,CB,XX,46,4,FD,CB,XX,46,2,CB,47,2,CB,40,2,CB,41,2,CB,42

,2,CB,43,2,CB,44

10060 DATA 2,CB,45,2,CB,4E,4,DD,CB,XX,4E,4,FD,CB,XX,4E

10070 DATA 2,CB,4F,2,CB,48,2,CB,49,2,CB,4A,2,CB,4B,2,CB,4C,2,CB,4D,2,CB,56

10080 DATA 4,DD,CB,XX,56,4,FD,CB,XX,56,2,CB,57,2,CB,50,2,CB,51,2,CB,52,2,CB,53,2,
CB,54,2,CB,55,2,CB,5E,4,DD,CB,XX,5E,4,FD,CB,XX,5E,2

,CB,5F,2,CB,58,2,CB,59,2,CB,5A,2,CB,5B,2,CB,5C,2,CB,5D,2,CB,66,4,DD,CB,XX,66,4,F
D,CB,XX,66

10090 DATA 2,CB,67,2,CB,60,2,CB,61,2,CB,62,2,CB,63,2,CB,64,2,CB,65,2,CB,6E,4,DD,
CB,XX,6E,4,FD,CB,XX,6E,2,CB,6F,2,CB,68,2,CB,69,2,CB,

6A,2,CB,6B,2,CB,6C,2,CB,6D,2,CB,76,4,DD,CB,XX,76,4,FD,CB,XX,76,2,CB,77,2,CB,70,2,
CB,71,2,CB,72,2,CB,73,2,CB,74

10100 DATA 2,CB,75,2,CB,7E,4,DD,CB,XX,7E,4,FD,CB,XX,7E,2,CB,7F,2,CB,78,2,CB,79,2,
CB,7A,2,CB,7B,2,CB,7C,2,CB,7D,3,DC,XX,XX,3,FC,XX,XX

,3,D4,XX,XX,3,CD,XX,XX,3,C4,XX,XX,3,F4,XX,XX,3,ED,XX,XX,3,E4,XX,XX,3,CC,XX,XX,1,
3F,1,BE,3,DD,BC,XX,3,FD,BE,XX,1,BF

10110 DATA 1,B8,1,B9,1,BA,1,BB,1,BC,1,BD,2,FE,XX,2,ED,A9,2,ED,B9,2,ED,A1,2,ED,B1,
1,2F,1,27,1,35,3,DD,35,XX,3,FD,35,XX,1,3D,1,05,1,0B

,1,0D,1,15,1,1B,1,1D,1,25,1,2B,2,DD,2B,2,FD,2B,1,2D,1,3B,1,F3,2,10,XX,1,FB,1,E3,
2,DD,E3,2,FD,E3,1,0B,1,EB

10120 DATA 1,D9,1,76,2,ED,46,2,ED,56,2,ED,5E,2,ED,78,2,DB,XX,2,ED,40,2,ED,48,2,E
D,50,2,ED,58,2,ED,60,2,ED,68,1,34,3,DD,34,XX,3,FD,34

,XX,1,3C,1,04,1,03,1,0C,1,14,1,13,1,1C,1,24,1,23,2,DD,23,2,FD,23,1,2C,1,33,2,ED,
AA,2,ED,BA,2,ED,A2,2,ED,B2

10130 DATA 1,E9,2,DD,E9,2,FD,E9,3,DA,XX,XX,3,FA,XX,XX,3,D2,XX,XX,3,C3,XX,XX,3,C2,
XX,XX,3,F2,XX,XX,3,EA,XX,XX,3,E2,XX,XX,3,CA,XX,XX,2

,38,XX,2,18,XX,2,30,XX,2,20,XX,2,28,XX,1,02,1,12,1,77,1,70,1,71,1,72,1,73,1,74,1,
75,2,36,XX,3,DD,77,XX,3,DD,70,XX,3,DD,71,X

10140 DATA 3,DD,72,XX,3,DD,73,XX,3,DD,74,XX,3,DD,75,XX,4,DD,36,XX,XX,3,FD,77,XX,
3,FD,70,XX,3,FD,71,XX,3,FD,72,XX,3,FD,73,XX,3,FD,74,

XX,3,FD,75,XX,4,FD,36,XX,XX,3,32,XX,XX,4,ED,43,XX,XX,4,ED,53,XX,XX,3,22,XX,XX,4,
DD,22,XX,XX,4,FD,22,XX,XX,4,ED,73,XX,XX,1,0

10150 DATA 1,1A,1,7E,3,DD,7E,XX,3,FD,7E,XX,3,3A,XX,XX,1,7F,1,78,1,79,1,7A,1,7B,1,
7C,2,ED,57,1,7D,2,3E,XX,2,ED,5F,1,46,3,DD,46,XX,3,F

D,46,XX,1,47,1,40,1,41,1,42,1,43,1,44,1,45,2,06,XX,4,ED,4B,XX,XX,3,01,XX,XX,1,4E,
3,DD,4E,XX,3,FD,4E,XX,1,4F,1,48,1,49,1,4A

10160 DATA 1,4B,1,4C,1,4D,2,0E,XX,1,56,3,DD,56,XX,3,FD,56,XX,1,57,1,50,1,51,1,52,
1,53,1,54,1,55,2,16,XX,4,ED,5B,XX,XX,3,11,XX,XX,1,5

E,3,DD,5E,XX,3,FD,5E,XX,1,5F,1,58,1,59,1,5A,1,5B,1,5C,1,5D,2,1E,XX,1,66,3,DD,66,
XX,3,FD,66,XX,1,67,1,60,1,61,1,62,1,63,1,64

10170 DATA 1,65,2,26,XX,3,2A,XX,XX,3,21,XX,XX,2,ED,47,4,DD,2A,XX,XX,4,DD,21,XX,X
X,4,FD,2A,XX,XX,4,FD,21,XX,XX,1,6E,3,DD,6E,XX,3,FD,6

E,XX,1,6F,1,68,1,69,1,6A,1,6B,1,6C,1,6D,2,2E,XX,2,ED,4F,4,ED,7B,XX,XX,1,F9,2,DD,
F9,2,FD,F9,3,31,XX,XX,2,ED,AB

10180 DATA 2,ED,B8,2,ED,A0,2,ED,80,2,ED,44,1,00,1,B6,3,DD,B6,XX,3,FD,B6,XX,1,B7,
1,B0,1,B1,1,B2,1,B3,1,B4,1,B5,2,F6,XX,2,ED,BB,2,ED,B

3,2,ED,79,2,ED,41,2,ED,49,2,ED,51,2,ED,59,2,ED,61,2,ED,69,2,D3,XX,2,ED,AB,2,ED,A
3,1,F1,1,C1,1,D1,1,E1,2,DD,E1,2,FD,E1,1,F5


```

10190 DATA 1,C5,1,D5,1,E5,2,DD,E5,2,FD,E5,2,CB,86,4,DD,CB,XX,86,4,FD,CB,XX,86,2,
CB,87,2,CB,80,2,CB,81,2,CB,82,2,CB,83,2,CB,84,2,CB,8
5,2,CB,8E,4,DD,CB,XX,8E,4,FD,CB,XX,8E,2,CB,8F,2,CB,88,2,CB,89,2,CB,8A,2,CB,8B,2,
CB,8C,2,CB,8D,2,CB,96,4,DD,CB,XX,96

10200 DATA 4,FD,CB,XX,96,2,CB,97,2,CB,90,2,CB,91,2,CB,92,2,CB,93,2,CB,94,2,CB,95
,2,CB,9E,4,DD,CB,XX,9E,4,FD,CB,XX,9E,2,CB,9F,2,CB,98
,2,CB,99,2,CB,9A,2,CB,9B,2,CB,9C,2,CB,9D,2,CB,A6,4,DD,CB,XX,A6,4,FD,CB,XX,A6,2,C
B,A7

10210 DATA 2,CB,A0,2,CB,A1,2,CB,A2,2,CB,A3,2,CB,A4,2,CB,A5

10220 DATA 2,CB,AE,4,DD,CB,XX,AE,4,DD,CB,XX,AE,2,CB,AF,2,CB,AB,2,CB,A9,2,CB,AA,2
,CB,AB,2,CB,AC,2,CB,AD,2,CB,B6,4,DD,CB,XX,B6,4,FD,CB
,XX,B6,2,CB,B7,2,CB,B0,2,CB,B1,2,CB,B2,2,CB,B3,2,CB,B4,2,CB,B5,2,CB,BE,4,DD,CB,X
X,BE,4,FD,CB,XX,BE,2,CB,BF,2,CB,B8,2,CB,B9

10230 DATA 2,CB,BA,2,CB,BB,2,CB,BC,2,CB,BD,1,C9,1,D8,1,FB,1,DO,1,CO,1,FO,1,EB,1,
EO,1,C8,2,ED,4D,2,ED,4S,2,CB,16,4,DD,CB,XX,16,4,FD,C
B,XX,16,2,CB,17,2,CB,10,2,CB,11,2,CB,12,2,CB,13,2,CB,14,2,CB,15,1,17,2,CB,06,4,D
D,CB,XX,06,4,FD,CB,XX,06,2,CB,07,2,CB,00

10240 DATA 2,CB,01,2,CB,02,2,CB,03,2,CB,04,2,CB,05,1,07,2,ED,6F,2,CB,1E,4,DD,CB,
XX,1E,4,FD,CB,XX,1E,2,CB,1F,2,CB,18,2,CB,19,2,CB,1A,
2,CB,1B,2,CB,1C,2,CB,1D,1,1F,2,CB,0E,4,DD,CB,XX,0E,4,FD,CB,XX,0E,2,CB,0F,2,CB,08
,2,CB,09,2,CB,0A,2,CB,0B,2,CB,0C,2,CB,0D

10250 DATA 1,0F

10260 DATA 2,ED,67,1,C7,1,CF,1,D7,1,DF,1,E7,1,EF,1,F7,1,FF,1,9E,3,DD,9E,XX,3,FD,
9E,XX,1,9F,1,98,1,99,1,9A,1,9B,1,9C,1,9D,2,DE,XX,2,E
D,42,2,ED,52,2,ED,62,2,ED,72,1,37,2,CB,C6,4,DD,CB,XX,C6,4,FD,CB,XX,C6,2,CB,C7,2,
CB,C0,2,CB,C1,2,CB,C2,2,CB,C3,2,CB,C4

10270 DATA 2,CB,C5,2,CB,CE,4,DD,CB,XX,CE,4,FD,CB,XX,CE,2,CB,CF,2,CB,C8,2,CB,C9,2
,CB,CA,2,CB,CB,2,CB,CC,2,CB,CD,2,CB,D6

10280 DATA 4,DD,CB,XX,D6,4,FD,CB,XX,D6,2,CB,D7,2,CB,D0,2,CB,D1,2,CB,D2,2,CB,D3,2
,CB,D4,2,CB,D5,2,CB,DE,4,DD,CB,XX,DE,4,FD,CB,XX,DE,2
,CB,DF,2,CB,D8,2,CB,D9,2,CB,DA,2,CB,DC,2,CB,DD,2,CB,E6,4,DD,CB,XX,E6,4,FD,CB,XX,
E6,2,CB,E7,2,CB,E0,2,CB,E1,2,CB,E2,2,CB,E3

10290 DATA 2,CB,E4,2,CB,E5,2,CB,EE,4,DD,CB,XX,EE,4,FD,CB,XX,EE,2,CB,EF

10300 DATA 2,CB,E8,2,CB,E9,2,CB,EA,2,CB,EB,2,CB,EB,2,CB,EC,2,CB,ED,2,CB,F6,4,DD,
CB,XX,F6,4,FD,CB,XX,F6,2,CB,F7,2,CB,FO,2,CB,F1,2,CB,
F2,2,CB,F3,2,CB,F4,2,CB,F5,2,CB,FE,4,DD,CB,XX,FE,4,FD,CB,XX,FE,2,CB,FF

10310 DATA 2,CB,F8,2,CB,F9,2,CB,FA,2,CB,FB,2,CB,FC,2,CB,FD,2,CB,26,4,DD,CB,XX,26
,4,FD,CB,XX,26,2,CB,27,2,CB,20,2,CB,21,2,CB,22,2,CB,
23,2,CB,24,2,CB,25,2,CB,2E,4,DD,CB,XX,2E,4,FD,CB,XX,2E,2,CB,2F,2,CB,28,2,CB,29,2
,CB,2A,2,CB,2B,2,CB,2C,2,CB,2D,2,CB,3E

10320 DATA 4,DD,CB,XX,3E,4,FD,CB,XX,3E,2,CB,3F,2,CB,38,2,CB,39,2,CB,3A,2,CB,3B,2
,CB,3C,2,CB,3D,1,96,3,DD,96,XX,3,FD,96,XX,1,97,1,90,
1,91,1,92,1,93,1,94,1,95,2,D6,XX,1,AE,3,DD,AE,XX,3,FD,AE,XX,1,AF,1,AB,1,A9,1,AA,
1,AB,1,AC,1,AD,2,EE,XX

```

- Lignes 10 à 30 : Programme principal
- Lignes 1000 à 1220 : Lecture des données et mémorisation
- Ligne 1050 : 1^{er} Op-code
 - Ligne 1080 : Nombre de 2^e Op-code
 - Ligne 1120 : 2^e Op-code
 - Ligne 1200 : Code généré
- Lignes 2000 à 2140 : Menu général
- Lignes 3000 à 3920 : Editeur de lignes
- Lignes 3000 à 3180 : Menu
 - Lignes 3190 à 3280 : Liste sur écran
 - Lignes 3290 à 3450 : Suppression ligne
 - Lignes 3460 à 3640 : Insertion ligne
 - Lignes 3650 à 3810 : Edition à l'écran
 - Lignes 3820 à 3920 : Impression
- Lignes 4000 à 4510 : Compilation du source
- Lignes 4120 à 4230 : Messages d'erreur
 - Lignes 4310 à 4510 : Extraction des données numériques.
- Lignes 5000 à 5110 : Chargement en mémoire d'un programme source
- Lignes 6000 à 6210 : Sauvegarde sur disquette d'un programme source ou objet
- Lignes 7000 à 10320 : Données
- Lignes 7000 à 7040 : 1^{er} Op-code
 - Lignes 8000 à 8030 : Nombre de 2^e Op-code
 - Lignes 9000 à 9390 : 2^e Op-code
 - Lignes 10000 à 10320 : Code généré

Par la suite, nous verrons comment exécuter un programme en Assembleur issu du compilateur que nous venons d'étudier. Ce programme pourra être implanté en mémoire ou dans un programme BASIC. Nous verrons également comment créer un « DEBUGGER ». Ce programme facilitera la mise au point de vos programmes écrits en Assembleur.

9/2.3

Le debugger

Qu'est-ce qu'un debugger ?

Ce mot vient du mot anglais « bug » qui signifie erreur en français. Un debugger est un outil qui permet de faciliter la mise au point des programmes écrits en assembleur. C'est un complément indispensable aux programmes précédents : désassembleur et assembleur.

Analyse du problème

Nous allons créer un debugger écrit dans le langage assembleur Z80 le moins complexe mais le plus complet possible. Les fonctions minimum d'un tel programme sont :

- Lecture d'un programme source (en code exécutable) par exemple créé par le programme assembleur décrit en 9/2.2 ;
- Sauvegarde sur disquette d'un programme source éventuellement modifié, ou d'une zone mémoire ;
- Exécution d'un programme ou d'une partie de programme à partir d'une adresse déterminée et jusqu'à une adresse déterminée puis affichage des registres lorsque l'adresse finale est atteinte ;
- Affichage et éventuellement modification d'une case mémoire ;
- Affichage d'une zone mémoire.

Les fonctions que nous venons d'énumérer sont accessibles par un menu affiché en début de programme et après chaque commande. Une fonction est activée en tapant la lettre du menu correspondante (par exemple L pour Lecture, X pour exécution, etc.). Passez en mode CAPS LOCK pour que le debugger fonctionne correctement. Si vous tapez une option que le debugger ne comprend pas, le menu est réaffiché et aucune autre action n'est effectuée.

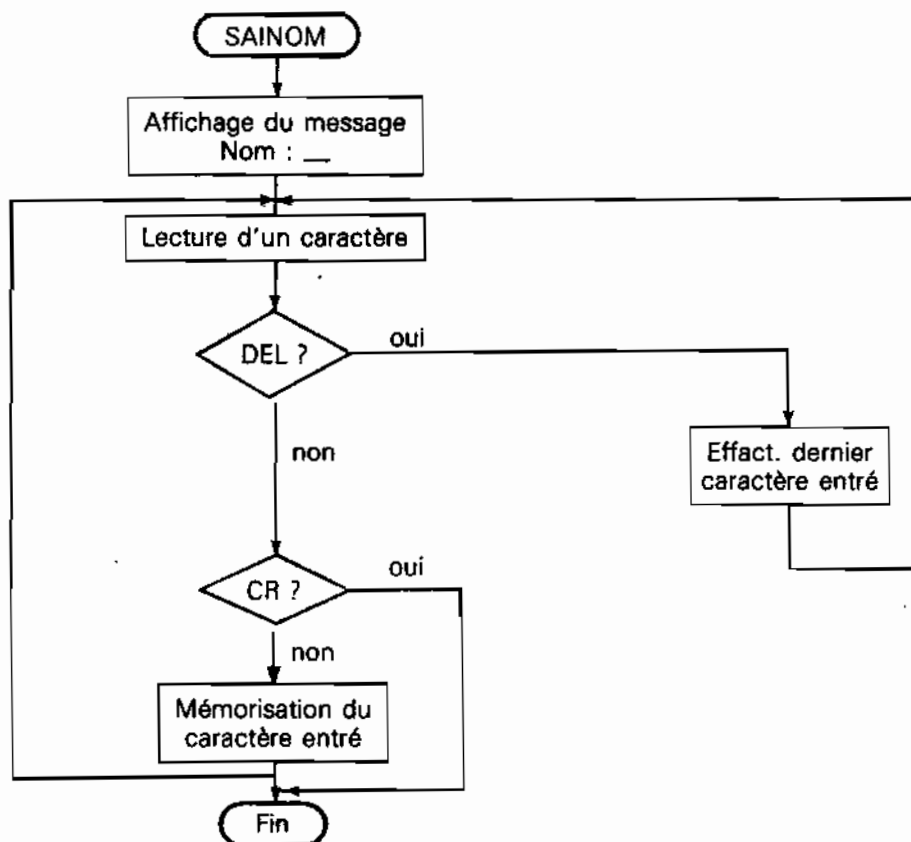
Les fonctions du debugger sont développées dans le programme assembleur qui suit. Nous allons les examiner en détail.

I. Lecture d'un programme objet

Les actions réalisées par cette option sont les suivantes :

- saisie du nom du programme à lire :

Pour cela, nous avons créé une routine de lecture (SAINOM) qui permet de lire une chaîne alphanumérique quelconque de longueur maximum 12 caractères qui se termine par le caractère CR (Carriage Return). La chaîne est stockée dans un buffer de lecture (BUFF). La routine de lecture suit la logique suivante :



- ouverture du fichier :

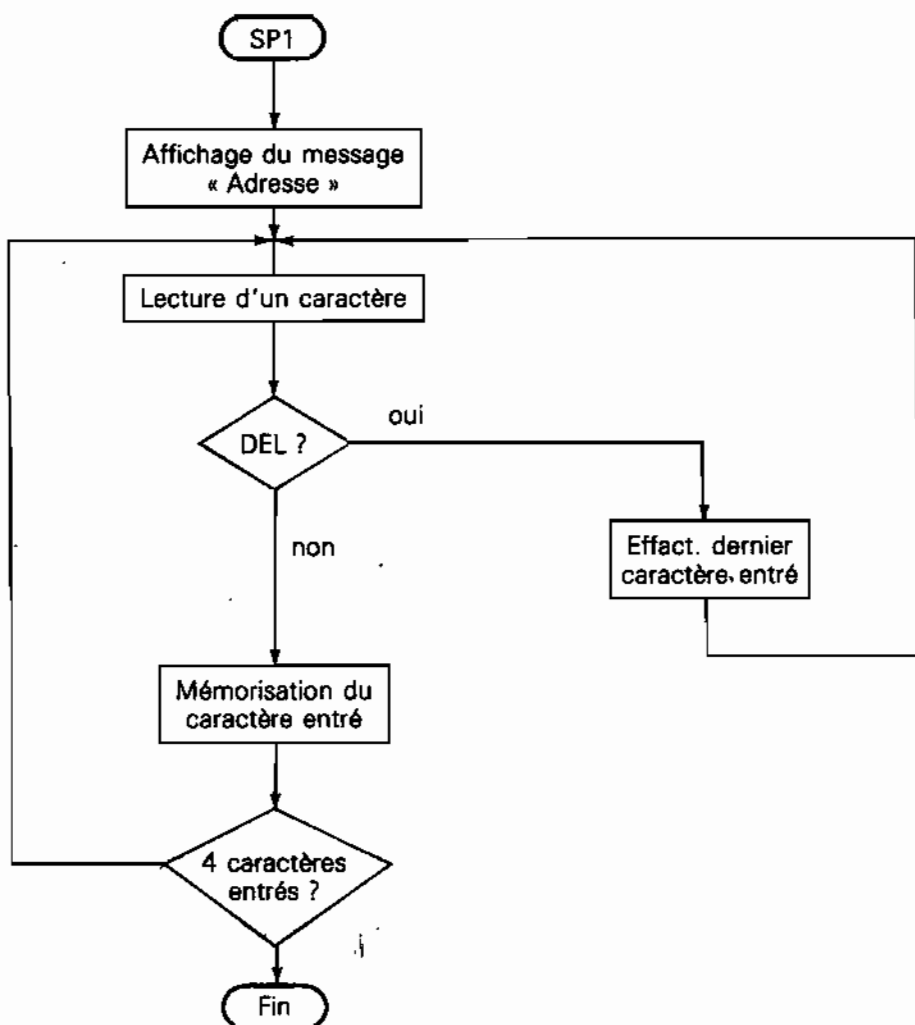
L'ouverture du fichier se fait en utilisant la macro du Firmware qui se trouve à l'adresse #BC77 et qui a pour nom CAS IN OPEN. A noter que cette macro est utilisable sur les trois CPC (464, 664 et 6128).

- lecture de l'adresse d'implantation :

Cette lecture est précédée de l'affichage du message "A implanter à l'adresse : __".

La lecture de l'adresse d'implantation est faite en utilisant une autre routine que SAINOM pour rendre automatique le retour chariot en fin de saisie. En effet, le nombre de caractères saisis est invariablement quatre. Le

code ASCII de chaque caractère est stocké séquentiellement dans la zone BUFF. La logique de la routine SP1 créée à cet effet est la suivante :



— Lecture du fichier ouvert en lecture :

La lecture du fichier se fait en utilisant la macro du Firmware située à l'adresse #BC83 qui a pour nom CAS IN DIRECT. A noter que cette routine est identique sur les trois systèmes CPC. Sur les modèles 664 et 6128, la lecture se fait cependant sur disquette.

— Fermeture du fichier ouvert en lecture :

La fermeture du fichier se fait en utilisant la macro du Firmware située à l'adresse #BC7A qui a pour nom CAS IN CLOSE. A noter que cette routine est identique sur les trois systèmes CPC.

Comme dans toutes les options du menu, les registres AF, BC, DE et HL sont mis en pile au début de la routine et dépilés en fin de routine.

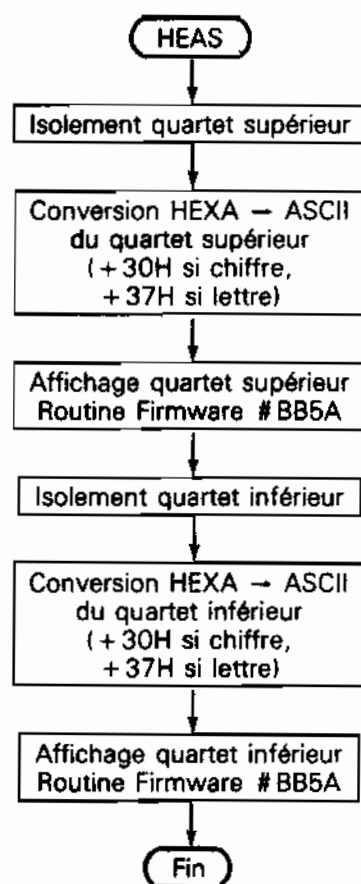
II. Sauvegarde d'un programme objet ou d'une zone mémoire

- Les registres AF, BC, DE et HL sont sauvegardés en début de routine ;
- Le nom du fichier cassette ou disquette est demandé en utilisant la même routine (SAINOM) que pour l'option de lecture précédemment décrite ;
- Le fichier est ouvert en écriture en utilisant la macro du Firmware située en #BC8C (CAS OUT OPEN) ;
- Les adresses de début et de fin de sauvegarde sont demandées en utilisant le sous-programme de saisie SP1 ;
- Les données correspondantes sont alors envoyées vers le lecteur de cassettes ou de disquettes en utilisant la macro du Firmware située en #BC98 (CAS OUT DIRECT) ;
- Le fichier ouvert en écriture est enfin fermé en utilisant la macro du Firmware située en #BC8F (CAS OUT CLOSE).
- Les registres AF, BC, DE et HL sont dépilés et le sous-programme rend la main au menu.

III. Exécution d'un programme

- Les registres AF, BC, DE et HL sont sauvegardés en début de routine ;
 - Les adresses de début et de fin d'exécution sont lues en utilisant la routine SP1. Ces adresses sont sauvegardées dans les variables ADDEB et ADFIN ;
 - Les trois octets de code commençant à l'adresse de fin d'exécution sont sauvegardés dans une zone buffer (BSTOP) ;
 - Le code JP 83D0H (#C3, #D0, #83) est implanté dans cette zone. Ce code permet d'activer la routine d'affichage du contenu des registres. Voyons en détail son fonctionnement :
- Une routine de nom ALPHA affiche toutes les constantes texte (nom des registres) sur l'écran ;
 - Une routine de nom AVAL affiche le contenu des registres. Pour cela, une routine d'affichage spécifique a été développée. Elle a pour nom HEAS. Le quartet supérieur est séparé du quartet inférieur, converti en ASCII, et affiché par une routine Firmware.

L'organigramme suivant décrit l'enchaînement des opérations de conversion et d'affichage :



— La routine d'affichage des registres rend la main à la routine EXEC qui restitue le code sauvegardé dans BSTOP, dépile les registres et rend la main au menu.

IV. Affichage du contenu d'une mémoire

- Les registres AF, BC, DE et HL sont sauvegardés en début de routine ;
- L'adresse de la mémoire à afficher est saisie grâce à la routine SP1 décrite précédemment ;
- Le contenu de cette mémoire est alors converti en ASCII et affiché par la routine HEAS ;
- La routine SAISIE est alors activée. Elle permet de lire la nouvelle valeur qui sera stockée dans la mémoire sélectionnée. Retapez la même valeur si vous ne voulez pas la modifier ;

- La routine ASCHEX convertit (ASCII->HEXA) les caractères ASCII entrés au clavier acquis dans la routine SAISIE. Le code HEXA résultant de la conversion est alors stocké dans la mémoire sélectionnée ;
- Les registres empilés en début de fonction sont dépilés et le contrôle est redonné au programme de menu.

V. Affichage d'une zone mémoire : fonction DUMP

- Les registres AF, BC, DE et HL sont sauvegardés en début de routine ;
- L'adresse de début de DUMP est saisie grâce à la routine SP1 décrite précédemment ;
- L'affichage des mémoires commence à cette adresse et respecte le format suivant :

Adresse	Val	Val	Val	Val	Val	Val	Val	Val
Adresse	Val	Val	Val	Val	Val	Val	Val	Val
Adresse	Val	Val	Val	Val	Val	Val	Val	Val
Adresse	Val	Val	Val	Val	Val	Val	Val	Val
Adresse	Val	Val	Val	Val	Val	Val	Val	Val
Adresse	Val	Val	Val	Val	Val	Val	Val	Val
Adresse	Val	Val	Val	Val	Val	Val	Val	Val
Adresse	Val	Val	Val	Val	Val	Val	Val	Val
Adresse	Val	Val	Val	Val	Val	Val	Val	Val
Adresse	Val	Val	Val	Val	Val	Val	Val	Val

- Les registres empilés en début de fonction sont dépilés et le contrôle est redonné au programme de menu.

VI. Retour au BASIC

Le programme debugger étant activé sous BASIC, il suffit d'une instruction RET pour y retourner. C'est ce que réalise l'option B du menu.

Saisie du programme

Deux méthodes sont possibles :

- Saisie en assembleur ;
- Saisie en BASIC.

1^{re} méthode

Vous pouvez utiliser l'assembleur décrit dans cet ouvrage ou un autre assembleur pour entrer le programme « debugger ». Voici le listing commenté du programme en page 7.


```

1          ORG 8000H
2          LOAD 8000H
3          PRINT: EQU 0BB5AH          ;TXT OUTPUT
4          SUP:   EQU 62              ;Signe superieur
5          READ:  EQU 0BB06H          ;KM WAIT CHAR
6          CURS:  EQU 95              ;Caract. curseur
7          BS:    EQU 8               ;Caractere Back-Space
8          CR:    EQU 13              ;Carriage Return
9          LF:    EQU 10              ;Line Feed
10         BLANC: EQU 32               ;Caract. ESPACE
11         DEL:   EQU 127              ;Caract. DElete
12         OPEN:  EQU 0BC77H          ;CAS IN OPEN
13         DREAD: EQU 0BC83H          ;CAS IN DIRECT
14         CLOSE: EQU 0BC7AH          ;CAS IN CLOSE
15         LECT:  EQU B2E0H            ;Routine externe
16         ECR:   EQU 835DH            ;Routine externe
17         ;=====
18         ;PROGRAMME PRINCIPAL
19         ;=====
20         BP:    EQU *                ;Boucle principale
21 8000 CDD480    CALL MENU              ;Affichage menu
22 8003 CDEB80    CALL CMD               ;Saisie d'une commande
23 8006 FE4C      CP 76                  ;L?
24 8008 CAE082    JP Z,LECT              ;Lecture
25 800B FE45      CP 69                  ;E?
26 800D CA5D83    JP Z,ECR               ;Ecriture
27 8010 FE58      CP 88                  ;X?
28 8012 CA0F81    JP Z,EXEC              ;Execution
29 8015 FE4D      CP 77                  ;M?
30 8017 CABF81    JP Z,MEM               ;Memoire
31 801A FE44      CP 68                  ;D?
32 801C CA7081    JP Z,DUMP              ;Dump.

```

```

33 801F FE42          CP    66          ;B?
34 8021 CA49B2        JP    Z,BASIC     ;Retour au BASIC
35 8024 18DA          JR    BP          ;Boucle principale
36 8026 00            NOP
37
38          BUFF:      EQU    $
39          ;-----
40          ;Definition de buffers
41          ;-----
42          DS    12          ;12 caract au maximum
43          MAX:      DS    1
44          ADDEB:     DS    2          ;Adresse de debut
45          ADFIN:     DS    2          ;Adresse de fin
46          BSTOP:     DS    3          ;Buffer stop prog.
47
48          DATA:     EQU    $
49          ;-----
50          ;Donnees alphanum. a afficher
51          ;-----
52 803B 44656275        DB    68,101,98,117,103,
53 803F 67676572
53 8043 205A3830        DB    32,90,56,48,13,10,
53 8047 0D0A4C29
53 804B 6563
54 804D 74757265        DB    116,117,114,101,32
54 8051 20202020
54 8055 20
55 8056 20452963        DB    32,69,41,99,114,10
55 805A 72697475
55 805E 72
56 805F 650D0A65        DB    101,13,10,101,88,4
56 8063 58296578
56 8067 63

```

```

57 806B 7574696F          DB  117,116,105,111,11
57 806C 6E202020
58 8070 4D29656D          DB  77,41,101,109,111,
58 8074 6F697265
59 8078 0D0A4429          DB  13,10,68,41,117,10
59 807C 756D7020
59 8080 20
60 8081 20202020          DB  32,32,32,32,32,32,
60 8085 20202042
60 8089 29
61 808A 61736963          DB  97,115,105,99,13,1
61 808E 0D0AFF
62
63
64          DATA2:      EQU  $              ;Message "Adresse :_"
65 8091 41647265          DB  65,100,114,101,115
65 8095 73736520
66 8099 3A5FFF           DB  58,95,0FFH
67
68          DATA3:      EQU  $              ;Message "a partir
69                      ;de 1'"
70 809C 61207061          DB  97,32,112,97,114,1
70 80A0 72746972
71 80A4 20646520          DB  32,100,101,32,108,
71 80AB 6C27FF
72
73          DATA4:      EQU  $              ;Message "jusqu'a 1'"
74 80AB 0D0A             DB  13,10
75 80AD 6A757371          DB  106,117,115,113,11
75 80B1 75276120
76 80B5 6C27FF           DB  108,39,0FFH
77

```

```

78          DATA5:      EQU  *                      ;Message "Nom :_"
79 80B8 0D0A4E6F          DB  13,10,7B,111,109
79 80BC 6D
80 80BD 203A5FFF          DB  32,58,95,0FFH
81
82          DATA6:      EQU  *                      ;Message "A implanter
83                      ;a 1'"
84 80C1 0D0A4120          DB  13,10,65,32,105,10
84 80C5 696D
85 80C7 706C616E          DB  112,108,97,110,116
85 80CB 74
86 80CC 65722061          DB  101,114,32,97,32,1
86 80D0 206C
87 80D2 27FF              DB  39,0FFH
88
89          ;-----
90          MENU:         EQU  *
91          ;-----
92          ;Entree: Donnees situees en DATA
93          ;Sortie: Aucun registre modifie
94          ;-----
95 80D4 213BB0              LD   HL,DATA
96 80D7 CDD880              CALL AFALPH                ;Affichage menu
97 80DA C9                  RET
98
99
00          AFALPH:       EQU  *
01          ;-----
02          ;Entree : Zone d'aff dans HL
03          ;Sortie : Aucun registre modifie
04          ;-----

```

```

105 80DB E5          PUSH HL
106 80DC F5          PUSH AF
107                ME1: EQU $                ;Boucle d'affichage
108 80DD 7E          LD  A,(HL)
109 80DE FEFF        CP  OFFH
110 80E0 2B06        JR  Z,ME2                ;Fin d'affichage
111 80E2 CD5ABB       CALL PRINT              ;Aff. caractere
112 80E5 23          INC HL                  ;Donnee suivante
113 80E6 1BF5        JR  ME1
114                ME2: EQU $
115 80E8 F1          POP  AF
116 80E9 E1          POP  HL
117 80EA C9          RET
118
119                CMD: EQU $
120                ;-----
121                ;Saisie d'une commande
122                ;Entree: Aucune
123                ;Sortie: Tous registres intacts
124                ;-----
125 80EB 3E3E        LD  A,SUP
126 80ED CD5ABB       CALL PRINT              ;Affichage ">"
127 80F0 3E5F        LD  A,CURS
128 80F2 CD5ABB       CALL PRINT              ;Affichage curseur
129 80F5 CD06BB       CALL READ              ;Lecture d'un caract.
130 80F8 F5          PUSH AF
131 80F9 F5          PUSH AF
132 80FA 3E0B        LD  A,BS
133 80FC CD5ABB       CALL PRINT              ;Affichage back-space
134 80FF F1          POP  AF
135 B100 CD5ABB       CALL PRINT              ;Affichage caractere
136 B103 3E0D        LD  A,CR

```

```

137 8105 CD5ABB          CALL PRINT
138 8108 3E0A           LD  A,LF
139 810A CD5ABB          CALL PRINT          ;Affiche CR+LF
140 810D F1             POP  AF
141 810E C9             RET
142
143          EXEC:      EQU  $
144          ;-----
145          ;Execution d'un programme
146          ;-----
147 810F F5             PUSH AF
148 8110 C5             PUSH BC
149 8111 D5             PUSH DE
150 8112 E5             PUSH HL          ;Sauvegarde des registres
151 8113 219C80         LD  HL,DATA3
152 8116 CDBB80         CALL AFALPH
153 8119 CD9782         CALL SP1          ;Saisie de l'À de depart
154 811C 223480         LD  (ADDEB),HL    ;Sauv. À debut
155 811F 21AB80         LD  HL,DATA4
156 8122 CDBB80         CALL AFALPH
157 8125 CD9782         CALL SP1          ;Saisie de l'À de fin
158 8128 223680         LD  (ADFIN),HL    ;Sauvegarde À fin
159
160          X3:        EQU  $          ;Execution
161 812B 3E0D           LD  A,CR
162 812D CD5ABB          CALL PRINT
163 8130 3E0A           LD  A,LF
164 8132 CD5ABB          CALL PRINT          ;Passage a la ligne
165 8135 2A3680         LD  HL,(ADFIN)
166 8138 7E            LD  A,(HL)
167 8139 323880         LD  (BSTOP),A
168 813C 23            INC  HL

```

```

169 813D 7E          LD  A,(HL)
170 813E 323980      LD  (BSTOP+1),A
171 8141 23          INC  HL
172 8142 7E          LD  A,(HL)
173 8143 323A80      LD  (BSTOP+2),A      ;Sauvegarde code
174 8146 2A3680      LD  HL,(ADFIN)
175 8149 3EC3        LD  A,0C3H
176 814B 77          LD  (HL),A
177 814C 3ED0        LD  A,0D0H
178 814E 23          INC  HL
179 814F 77          LD  (HL),A
180 8150 3E93        LD  A,83H      ;JP 83D0H
181 8152 23          INC  HL
182 8153 77          LD  (HL),A      ;Debranch en 9000H
183 8154 2A3480      LD  HL,(ADDEB)
184 8157 E9          JP  (HL)      ;Execution programme
185
186                ARET:      EQU  $      ;Adresse de retour
187 8158 2A3680      LD  HL,(ADFIN)
188 815B 3A3880      LD  A,(BSTOP)
189 815E 77          LD  (HL),A
190 815F 3A3980      LD  A,(BSTOP+1)
191 8162 23          INC  HL
192 8163 77          LD  (HL),A
193 8164 3A3A80      LD  A,(BSTOP+2)
194 8167 23          INC  HL
195 8168 77          LD  (HL),A      ;Restitution code
196 8169 E1          POP  HL
197 816A D1          POP  DE
198 816B C1          POP  BC
199 816C F1          POP  AF      ;Restitution des registre
200 816D C30080      JP  BP      ;Retour boucle principale
201

```

```

202          DUMP:      EQU  $
203          ;-----
204          ;Affichage d'une zone memoire
205          ;-----
206 8170 F5          PUSH AF
207 8171 C5          PUSH BC
208 8172 D5          PUSH DE
209 8173 E5          PUSH HL          ;Sauvegarde des registres
210 8174 CD97B2      CALL SP1          ;Saisie de l'adresse
211 8177 1600        LD  D,0
212 8179 3E0D        DU1: LD  A,CR
213 817B CD5ABB      CALL PRINT
214 817E 3E0A        LD  A,LF
215 8180 CD5ABB      CALL PRINT
216 8183 2A29B0      LD  HL,(BUFF+2)  ;à 1e memoire
217 8186 7C          LD  A,H
218 8187 CD66B2      CALL HEAS
219 818A 7D          LD  A,L
220 818B CD66B2      CALL HEAS          ;Affichage adresse
221 818E 010000      LD  BC,0          ;Compteur colonnes
222 8191 3E20        DU2: LD  A,BLANC
223 8193 CD5ABB      CALL PRINT          ;Affichage ESPACE
224 8196 2A29B0      LD  HL,(BUFF+2)
225 8199 09          ADD HL,BC          ;à (mem) a afficher
226 819A 7E          LD  A,(HL)
227 819B CD66B2      CALL HEAS          ;Aff. memoire
228 819E 03          INC BC          ;Memoire suivante
229 819F 79          LD  A,C
230 81A0 FE08        CP  B
231 81A2 20ED        JR  NZ,DU2          ;Boucle sur colonne
232 81A4 23          INC HL
233 81A5 2229B0      LD  (BUFF),HL      ;Sauv prochaine à

```



```

234 81A8 14          INC  D
235 81A9 7A          LD   A,D
236 81AA FE0A        CP   10
237 81AC 20CB        JR   NZ,DU1          ;Boucle sur ligne
238 81AE 3E0D        LD   A,CR
239 81B0 CD5ABB      CALL PRINT
240 81B3 3E0A        LD   A,LF
241 81B5 CD5ABB      CALL PRINT          ;Passage a la ligne
242 81B8 E1          POP  HL
243 81B9 D1          POP  DE
244 81BA C1          POP  BC
245 81BB F1          POP  AF          ;Restitution des registre
246 81BC C30080      JP   BP          ;Retour boucle principale
247
248                MEM:      EQU  $
249                ;-----
250                ;Lecture et/ou modification d'une
251                ;memoire
252                ;-----
253 81BF F5          PUSH AF
254 81C0 C5          PUSH BC
255 81C1 D5          PUSH DE
256 81C2 E5          PUSH HL          ;Sauvegarde des registres 11
257 81C3 CD9782      CALL SP1          ;Saisie de l'adresse
258 81C6 7E          LD   A,(HL)
259 81C7 CD6682      CALL HEAS          ;Affichage (memoire)
260 81CA 3E20        LD   A,BLANC
261 81CC CD5ABB      CALL PRINT
262 81CF 3E3F        LD   A,CURS
263 81D1 CD5ABB      CALL PRINT
264
265 81D4 3E02        LD   A,2

```

```

266 81D6 323380      LD    (MAX),A           ;2 caracteres a saisir
267 81D9 CD0182      CALL SAISIE
268 81DC 212780      LD    HL,BUFF
269 81DF CD4A82      CALL ASCHEX           ;Conv. ASCII->HEXA
270 81E2 2A2980      LD    HL,(BUFF+2)
271 81E5 77          LD    (HL),A           ;Modif memoire
272 81E6 3E08        LD    A,BS
273 81E8 CD5ABB      CALL PRINT
274 81EB 3E20        LD    A,BLANC
275 81ED CD5ABB      CALL PRINT
276 81F0 3E0D        LD    A,CR
277 81F2 CD5ABB      CALL PRINT
278 81F5 3E0A        LD    A,LF
279 81F7 CD5ABB      CALL PRINT           ;Passage a la ligne
280 81FA E1          POP    HL
281 81FB D1          POP    DE
282 81FC C1          POP    BC
283 81FD F1          POP    AF           ;Retitution des registres
284 81FE C30080      JP     BP           ;Retour boucle principale
285
286
287                SAISIE:    EQU    $
288                ;-----
289                ;Saisie de caracteres alphanum.
290                ;Entree : (MAX)=nbre de caract
291                ;Sortie : aucun registre ecrase
292                ;-----
293 8201 3A3380      LD    A,(MAX)
294 8204 57          LD    D,A           ;D=Nbre de caract a lire
295 8205 010000      LD    BC,0         ;Index dans le buffer
296 8208 212780      S1:    LD    HL,BUFF      ;Buffer de lecture
297 820B CD06BB      CALL READ           ;1 caractere

```

298 820E FE7F	CP DEL	;DELeTe ?
299 8210 2B17	JR Z,S2	;Traitement special
300 8212 F5	PUSH AF	
301 8213 3E08	LD A,BS	
302 8215 CD5ABB	CALL PRINT	
303 8218 F1	POP AF	
304 8219 CD5ABB	CALL PRINT	
305 821C 09	ADD HL,BC	
306 821D 77	LD (HL),A	;Sauvegarde
307 821E 03	INC BC	
308 821F 3E5F	LD A,CURS	
309 8221 CD5ABB	CALL PRINT	
310 8224 79	LD A,C	
311 8225 BA	CP D	
312 8226 20E0	JR NZ,S1	;Suite de la saisie
313 8228 C9	RET	
314 8229 79 S2:	LD A,C	
315 822A B7	OR A	
316 822B 2BDB	JR Z,S1	;DEL non accepte
317 822D 0B	DEC BC	
318 822E 3E08	LD A,BS	
319 8230 CD5ABB	CALL PRINT	;Retour en arriere
320 8233 3E20	LD A,BLANC	
321 8235 CD5ABB	CALL PRINT	;Effacement caractere
322 823B 3E08	LD A,BS	
323 823A CD5ABB	CALL PRINT	
324 823D 3E08	LD A,BS	
325 823F CD5ABB	CALL PRINT	;2x retour en arriere
326 8242 3E5F	LD A,CURS	
327 8244 CD5ABB	CALL PRINT	;Affichage curseur
328 8247 1B8F	JR S1	
329		

```

330          BASIC:      EQU  $
331 8249 C9              RET
332          ASCHEX:     EQU  $
333          ;-----
334          ;Conversion ASCII->HEXA
335          ;Entree : HL pointe sur le data
336          ;Sortie : A contient la conversion
337          ;          Autres registres intacts
338          ;-----
339 824A C5              PUSH BC                ;Sauvegarde
340 824B 7E              LD  A,(HL)
341 824C FE40            CP   40H
342 824E 3B02            JR   C,A51
343 8250 D607            SUB  7
344 8252 D630  A51:      SUB  30H                ;Conversion MSQ
345 8254 17              RLA
346 8255 17              RLA
347 8256 17              RLA
348 8257 17              RLA                ;et passage en poids fort
349 8258 47              LD   B,A                ;Sauvegarde
350 8259 23              INC  HL
351 825A 7E              LD   A,(HL)
352 825B FE40            CP   40H
353 825D 3B02            JR   C,A52
354 825F D607            SUB  7
355 8261 D630  A52:      SUB  30H                ;Conversion MSQ
356 8263 B0              OR   B                ;Octet complet
357 8264 C1              POP  BC                ;Restitution de BC
358 8265 C9              RET
359
360
361          HEAS:       EQU  $

```

```

362      ;-----
363      ;Routine de conversion HEXA->ASCII
364      ;et d'affichage du resultat
365      ;-----
366      ;Entree: A=nombre a convertir
367      ;Sortie: A efface
368      ;-----
369 8266 F5          PUSH AF          ;Sauvegarde de A
370 8267 E6F0        AND  OF0H        ;Isole MSQ
371 8269 1F          RRA
372 826A 1F          RRA
373 826B 1F          RRA
374 826C 1F          RRA          ;Isole MSQ
375 826D FE0A        CP   10          ;Chiffre?
376 826F 3802        JR   C,HEAS1     ;Oui
377 8271 C607        ADD  A,7         ;Decalage pour une lettre
378      HEAS1:      EQU  $
379 8273 C630        ADD  A,30H        ;Conversion
380 8275 CD5ABB       CALL PRINT       ;Affichage MSQ
381 8278 F1          POP  AF          ;A retrouve
382 8279 E60F        AND  OFH
383 827B CB17        RL   A
384 827D CB17        RL   A
385 827F CB17        RL   A
386 8281 CB17        RL   A          ;Isole LSQ
387 8283 CB07        RLC  A
388 8285 CB07        RLC  A
389 8287 CB07        RLC  A
390 8289 CB07        RLC  A          ;LSQ a droite
391 828B FE0A        CP   10          ;Chiffre?
392 828D 3802        JR   C,HEAS2     ;Oui
393 828F C607        ADD  A,7

```

```

394          HEAS2:      EQU  *
395 8291 C630          ADD  A,30H          ;Conversion
396 8293 CD5ABB        CALL PRINT        ;Affichage LSO
397 8296 C9           RET
398
399
400          ;-----
401          ; ZONE SOUS PROGRAMMES
402          ;-----
403          SP1:         EQU  *          ;Saisie d'une adresse
404 8297 219180         LD   HL,DATA2
405 829A CDD880         CALL AFALPH        ;Affichage message
406 829D 3E04          LD   A,4
407 829F 323380         LD   (MAX),A      ;4 caract a saisir
408 82A2 CD0182         CALL SAISIE        ;Saisie
409 82A5 3E08          LD   A,85
410 82A7 CD5ABB        CALL PRINT
411 82AA 3E20          LD   A,BLANC
412 82AC CD5ABB        CALL PRINT
413 82AF 212780         LD   HL,BUFF
414 82B2 CD4AB2        CALL ASCHEX        ;Conversion ASCII->HEX
415 82B5 322780         LD   (BUFF),A     ;Sauvegarde LSB
416 82B8 212980         LD   HL,BUFF+2
417 82BB CD4AB2        CALL ASCHEX
418 82BE 6F           LD   L,A
419 82BF 3A2780         LD   A,(BUFF)
420 82C2 67           LD   H,A            ;HL=> memoire
421 82C3 222980         LD   (BUFF+2),HL  ;Sauvegarde &
422 82C6 C9           RET
423          END

```

```

1          ORG 82E0H
2          LOAD 82E0H
3
4          BUFF:    EQU 8027H
5          OPEN:    EQU 0BC77H          ;Ouv. fichier
6          DREAD:   EQU 0BC83H          ;Lecture disque
7          CLOSE:   EQU 0BC7AH          ;Ferm. fichier
8          DATA6:  EQU 80C1H           ;Message externe
9          AFALPH:  EQU 80DBH           ;Routine externe
10         SP1:     EQU 8297H           ;Routine externe
11         CR:      EQU 13              ;Carriage Return
12         LF:      EQU 10              ;Line Feed
13         PRINT:   EQU 0BB5AH          ;TXT OUTPUT
14         BP:      EQU 8000H           ;Boucle principale
15         DATA5:  EQU 80BBH           ;Message externe
16         READ:    EQU 0BB06H          ;KM WAIT CHAR
17         DEL:     EQU 127              ;DELeTe
18         BS:      EQU 8               ;Back Space
19         CURS:    EQU 95              ;Caractere curseur
20         BLANC:   EQU 32              ;Caractere espace
21         WOPEN:   EQU 0BC8CH          ;CAS OUT OPEN
22         WRITE:   EQU 0BC98H          ;CAS OUT DIRECT
23         WCLOSE:  EQU 0BC8FH          ;CAS OUT CLOSE
24         DATA3:  EQU 809CH           ;Message externe
25         ADDEB:   EQU 8034H           ;à debut exec.
26         ADFIN:   EQU 8036H           ;à fin exec.
27         DATA4:  EQU 80ABH           ;Message externe
28
29         LECT:    EQU $
30         ;-----
31         ;Lecture d'un programme objet
32         ;-----

```

```

33 82E0 F5          PUSH AF
34 82E1 C5          PUSH BC
35 82E2 D5          PUSH DE
36 82E3 E5          PUSH HL          ;Sauvegarde des registres
37 82E4 CD1183      CALL SAINOM      ;Saisie du nom
38 82E7 41          LD   B,C         ;Longueur du nom
39 82E8 1100C0      LD   DE,0C000H   ;Buffer de 2 KO
40 82EB 212780      LD   HL,BUFF     ;Zone du nom
41 82EE CD77BC      CALL OPEN        ;Ouverture fichier
42 82F1 21C180      LD   HL,DATA6
43 82F4 CDD880      CALL AFALPH      ;à d'implantation
44 82F7 CD9782      CALL SP1
45 82FA 3E0D        LD   A,CR
46 82FC CD5ABB      CALL PRINT
47 82FF 3E0A        LD   A,LF
48 8301 CD5ABB      CALL PRINT        ;CR+LF
49                ;HL contient l'à d'implantation
50 8304 CD83BC      CALL DREAD        ;Lecture disque
51 8307 CD7ABC      CALL CLOSE        ;Fermeture fichier
52 830A E1          POP  HL
53 830B D1          POP  DE
54 830C C1          POP  BC
55 830D F1          POP  AF          ;Restitution des registre
56 830E C30080      JP   BP          ;Retour boucle principale
57
58                SAINOM:   EQU  $          ;Saisie du nom
59                ;-----
60                ;Saisie de caracteres alphanum.
61                ;se terminant par un CR dans une
62                ;limite de 12 caracteres
63                ;-----
64 8311 21B880      LD   HL,DATA5

```



```

65 8314 CDD8B0          CALL AFALPH          ;Saisie du nom
66
67 8317 010000          LD   BC,0
68 831A 2127B0  L1:      LD   HL,BUFF          ;Buffer de lecture
69 831D CD06BB          CALL READ
70 8320 FE7F            CP   DEL
71 8322 2818            JR   Z,L2              ;Traitement special
72 8324 FE0D            CP   CR
73 8326 2834            JR   Z,L3              ;Fin de saisie
74 8328 F5             PUSH AF
75 8329 3E08            LD   A,BS
76 832B CD5ABB          CALL PRINT
77 832E F1             POP  AF
78 832F CD5ABB          CALL PRINT
79 8332 09             ADD  HL,BC
80 8333 77             LD   (HL),A          ;Sauvegarde
81 8334 03             INC  BC
82 8335 3E5F            LD   A,CURS
83 8337 CD5ABB          CALL PRINT
84 833A 18DE            JR   L1
85 833C 79             LD   A,C  L2:
86 833D B7             OR   A
87 833E 28DA            JR   Z,L1
88 8340 0B             DEC  BC
89 8341 3E08            LD   A,BS
90 8343 CD5ABB          CALL PRINT          ;Retour en arriere
91 8346 3E20            LD   A,BLANC
92 8348 CD5ABB          CALL PRINT          ;Effacement caractere
93 834B 3E08            LD   A,BS
94 834D CD5ABB          CALL PRINT
95 8350 3E08            LD   A,BS
96 8352 CD5ABB          CALL PRINT          ;2x retour en arriere

```

```

97 8355 3E5F          LD  A,CURB
98 8357 CD5ABB        CALL PRINT          ;Affichage curseur
99 835A 18BE          JR   L1
100 835C C9          L3:  RET              ;Fin de saisie
101
102
103                  ECR:      EQU  $
104                  ;-----
105                  ;Sauvegarde d'1 programme objet
106                  ;-----
107 835D F5          PUSH AF
108 835E C5          PUSH BC
109 835F D5          PUSH DE
110 8360 E5          PUSH HL          ;Sauvegarde des registres
111 8361 CD11B3       CALL SAINOM      ;Saisie du nom
112 8364 41          LD   B,C          ;Longueur nom
113 8365 1100C0       LD   DE,0C000H   ;Buffer 2 KO
114 8368 2127B0       LD   HL,BUFF     ;Zone du nom
115 836B CD8CBC       CALL WOPEN
116
117 836E 3E0D        LD   A,CR
118 8370 CD5ABB        CALL PRINT
119 8373 3E0A        LD   A,LF
120 8375 CD5ABB        CALL PRINT      ;CR+LF
121 8378 219CB0       LD   HL,DATA3
122 837B CDD8B0       CALL AFALPH
123 837E CD97B2       CALL SP1
124 8381 2234B0       LD   (ADDEB),HL   ;Sauv à debut
125 8384 21ABB0       LD   HL,DATA4
126 8387 CDD8B0       CALL AFALPH
127 838A CD97B2       CALL SP1
128 838D 2236B0       LD   (ADFIN),HL   ;Sauv à fin

```

129 8390 2A36B0	LD	HL, (ADFIN)	
130 8393 ED5B3480	LD	DE, (ADDEB)	
131 8397 37	SCF		
132 8398 3F	CCF		;Carry a zero
133 8399 ED52	SBC	HL, DE	
134 839B 223680	LD	(ADFIN), HL	;Sauv longueur
135			
136 839E 2A3480	LD	HL, (ADDEB)	
137 83A1 ED5B3680	LD	DE, (ADFIN)	
138 83A5 ED4B3480	LD	BC, (ADDEB)	
139 83A9 3E2A	LD	A, 2AH	;Fichier binaire
140 83AB CD98BC	CALL	WRITE	
141 83AE CDBFBC	CALL	WCLOSE	
142 83B1 3E0D	LD	A, CR	
143 83B3 CD5ABB	CALL	PRINT	
144 83B6 3E0A	LD	A, LF	
145 83B8 CD5ABB	CALL	PRINT	;Passage a la ligne
146 83BB E1	POP	HL	
147 83BC D1	POP	DE	
148 83BD C1	POP	BC	
149 83BE F1	POP	AF	;Restitution des registre
150 83BF C30080	JP	BP	;Retour boucle principale
151	END		

```

1          ORG 83D0H
2          LOAD 83D0H
3          ;
4          BLANC: EQU 32          ;Code HEXA espace
5          PRINT: EQU 0BB5AH      ;TXT OUTPUT
6          CR: EQU 13             ;Carriage Return
7          LF: EQU 10             ;Line Feed
8          ARET: EQU 8158H        ;Adresse de retour
9          ADFIN: EQU 8036H       ;A fin execution
10
11         AREG: EQU *
12         ;=====
13         ;AFFICHAGE DES REGISTRES
14         ;=====
15 83D0 CD1184          CALL ALPHA          ;Affichage alphanum.
16 83D3 CD2484          CALL AVAL           ;Affichage des valeurs
17 83D6 C35881          JP ARET             ;Retour au prog principal
18 83D9 C9             RET
19         ;=====
20
21
22
23
24         DATA: EQU *
25         ;-----
26         ;Donnees alphanum. a afficher
27         ;-----
28 83DA 0D0A          DB 13,
29 83DC 20484C20       DB 20H,48H,4CH,20H
30 83E0 20204445       DB 20H,20H,44H,45H
31 83E4 20202042       DB 20H,20H,20H,42H
32 83E8 43202020       DB 43H,20H,20H,20H

```

```

33 B3EC 41462020      DB  41H,46H,20H,20H
34 B3F0 2052490A      DB  20H,52H,49H,0AH
35 B3F4 0A0A0D20      DB  0AH,0AH,0DH,20H
36 B3FB 49582020      DB  49H,58H,20H,20H
37 B3FC 20495920      DB  20H,49H,59H,20H
38 B400 20205350      DB  20H,20H,53H,50H
39 B404 20202050      DB  20H,20H,20H,50H
40 B408 430B0B0D      DB  43H,0BH,0BH,0DH
41 B40C FF           DB  0FFH

42                ;-----
43                ;Zone buffer
44                ;-----
45 BUFF:          DS   2
46 SPC:           DS   2
47                ;
48 ALPHA:         EQU  *
49                ;-----
50                ;Affichage de texte
51                ;Entree : donnees situees en DATA
52                ;Sortie : aucun registre efface
53                ;-----
54 B411 E5          PUSH HL
55 B412 F5          PUSH AF                ;Sauvegardes
56 B413 21DAB3      LD   HL,DATA          ;HL pointe su le data
57                AF1: EQU  *              ;Boucle d'affichage
58 B416 7E          LD   A,(HL)
59 B417 FEFF        CP   0FFH
60 B419 2B06        JR   Z,AF2            ;Fin d'affichage
61 B41B CD5ABB      CALL PRINT            ;Affichage
62 B41E 23          INC  HL
63 B41F 1BF5        JR   AF1
64                AF2: EQU  *

```

```
65 8421 F1          POP  AF
66 8422 E1          POP  HL          ;Restitution registres
67 8423 C9          RET
68
69
70          AVAL:    EQU  $
71          ;-----
72          ;Aff. des valeurs des registres
73          ;Entree : Aucune
74          ;Sortie : Aucun registre ecrase
75          ;-----
76 8424 E5          PUSH HL
77 8425 F5          PUSH AF
78 8426 F5          PUSH AF          ;Sauvegarde registres
79 8427 7C          LD   A,H
80 8428 CD1385       CALL HEAS
81 842B 7D          LD   A,L
82 842C CD1385       CALL HEAS          ;Affichage HL
83 842F 3E20        LD   A,BLANC
84 8431 CD5ABB       CALL PRINT          ;Affichage espace
85 8434 7A          LD   A,D
86 8435 CD1385       CALL HEAS
87 8438 7B          LD   A,E
88 8439 CD1385       CALL HEAS          ;Affichage DE
89 843C 3E20        LD   A,BLANC
90 843E CD5ABB       CALL PRINT          ;Affichage espace
91 8441 7B          LD   A,B
92 8442 CD1385       CALL HEAS
93 8445 79          LD   A,C
94 8446 CD1385       CALL HEAS          ;Affichage BC
95 8449 3E20        LD   A,BLANC
96 844B CD5ABB       CALL PRINT          ;Affichage espace
```

97 844E E1	POP HL	
98 844F 7C	LD A,H	
99 8450 CD1385	CALL HEAS	
100 8453 7D	LD A,L	
101 8454 CD1385	CALL HEAS	;Affichage AF
102 8457 3E20	LD A,BLANC	
103 8459 CD5ABB	CALL PRINT	;Affichage espace
104 845C ED5F	LD A,R	
105 845E CD1385	CALL HEAS	
106 8461 ED57	LD A,I	
107 8463 CD1385	CALL HEAS	;Affichage RI
108		
109 8466 3E0D	LD A,CR	
110 8468 CD5ABB	CALL PRINT	
111 846B 3E0A	LD A,LF	
112 846D CD5ABB	CALL PRINT	;CR+LF
113 8470 D9	EXX	
114 8471 7C	LD A,H	
115 8472 D9	EXX	
116 8473 CD1385	CALL HEAS	
117 8476 D9	EXX	
118 8477 7D	LD A,L	
119 847B D9	EXX	
120 8479 CD1385	CALL HEAS	;Affichage HL'
121 847C 3E20	LD A,BLANC	
122 847E CD5ABB	CALL PRINT	;Affichage espace
123 8481 D9	EXX	
124 8482 7A	LD A,D	
125 8483 D9	EXX	
126 8484 CD1385	CALL HEAS	
127 8487 D9	EXX	
128 848B 7B	LD A,E	

```
129 8489 D9          EXX
130 848A CD1385       CALL HEAS          ;Affichage DE'
131 848D 3E20        LD  A,BLANC
132 848F CD5ABB       CALL PRINT        ;Affichage espace
133 8492 D9          EXX
134 8493 7B          LD  A,B
135 8494 D9          EXX
136 8495 CD1385       CALL HEAS
137 8498 D9          EXX
138 8499 79          LD  A,C
139 849A D9          EXX
140 849B CD1385       CALL HEAS          ;Affichage BC'
141 849E 3E20        LD  A,BLANC
142 84A0 CD5ABB       CALL PRINT        ;Affichage espace
143 84A3 0B          DB  0B             ;EX AF,AF'
144 84A4 E5          PUSH HL
145 84A5 F5          PUSH AF
146 84A6 CD1385       CALL HEAS          ;Affichage A'
147 84A9 E1          POP  HL
148 84AA 7D          LD  A,L
149 84AB CD1385       CALL HEAS          ;Affichage F'
150 84AE 0B          DB  0B             ;EX AF,AF'
151 84AF E1          POP  HL
152 84B0 3E0D        LD  A,CR
153 84B2 CD5ABB       CALL PRINT
154 84B5 3E0A        LD  A,LF
155 84B7 CD5ABB       CALL PRINT
156 84BA 3E0A        LD  A,LF
157 84BC CD5ABB       CALL PRINT        ;CR+LF+LF
158 84BF DD220B84     LD  (BUFF),IX
159 84C3 2A0B84       LD  HL,(BUFF)
160 84C6 7C          LD  A,H
```


161 84C7 CD1385	CALL HEAS	
162 84CA 7D	LD A,L	
163 84CB CD1385	CALL HEAS	;Affichage IX
164 84CE 3E20	LD A,BLANC	
165 84D0 CD5ABB	CALL PRINT	;Affichage espace
166 84D3 FD220D84	LD (BUFF),IY	
167 84D7 2A0D84	LD HL,(BUFF)	
168 84DA 7C	LD A,H	
169 84DB CD1385	CALL HEAS	
170 84DE 7D	LD A,L	
171 84DF CD1385	CALL HEAS	;Affichage IY
172 84E2 3E20	LD A,BLANC	
173 84E4 CD5ABB	CALL PRINT	;Affichage espace
174 84E7 ED730D84	LD (BUFF),SP	
175 84EB 2A0D84	LD HL,(BUFF)	
176 84EE 7C	LD A,H	
177 84EF CD1385	CALL HEAS	
178 84F2 7D	LD A,L	
179 84F3 CD1385	CALL HEAS	;Affichage SP
180 84F6 3E20	LD A,BLANC	
181 84F8 CD5ABB	CALL PRINT	;Affichage espace
182 84FB 2A36B0	LD HL,(ADFIN)	
183 84FE 7C	LD A,H	
184 84FF CD1385	CALL HEAS	
185 8502 7D	LD A,L	
186 8503 CD1385	CALL HEAS	;Affichage PC
187 8506 3E0D	LD A,CR	
188 8508 CD5ABB	CALL PRINT	
189 850B 3E0A	LD A,LF	
190 850D CD5ABB	CALL PRINT	;CR+LF
191 8510 F1	POP AF	
192 8511 E1	POP HL	

```

193 8512 C9          RET
194
195
196
197          HEAS:      EQU  *
198          ;-----
199          ;Conversion HEXA -> ASCII et
200          ;affichage du resultat
201          ;-----
202          ;Entree: A=Nombre a convertir
203          ;Sortie: A efface
204          ;-----
205 8513 F5          PUSH AF          ;Sauvegarde de A
206 8514 E6F0        AND  OF0H        ;Isole MSQ
207 8516 1F          RRA
208 8517 1F          RRA
209 8518 1F          RRA
210 8519 1F          RRA          ;Isole MSQ
211 851A FE0A        CP   10          ;Chiffre ?
212 851C 3B02        JR   C,HEAS1     ;Oui
213 851E C607        ADD  A,7         ;Decalage pour une lettre
214          HEAS1:    EQU  *
215 8520 C630        ADD  A,30H        ;Conversion
216 8522 CD5AB8      CALL PRINT       ;Affichage MSQ
217 8525 F1          POP  AF          ;A retrouve
218 8526 E60F        AND  OFH        ;Isole LSQ
219 8528 CB17        RL   A
220 852A CB17        RL   A
221 852C CB17        RL   A
222 852E CB17        RL   A          ;Isole,LSQ
223 8530 CB07        RLC  A
224 8532 CB07        RLC  A

```

```

225 8534 CB07          RLC  A
226 8536 CB07          RLC  A                ;LSQ a droite
227 8538 FE0A          CP   10                ;Chiffre ?
228 853A 3B02          JR   C,HEAS2           ;Dui
229 853C C607          ADD  A,7
230                   HEAS2: EQU  *
231 853E C630          ADD  A,30H              ;Conversion
232 8540 CD5AB8        CALL PRINT              ;Affichage
233 8543 C9           RET
234                   END

```

Une fois le programme saisi et assemblé, vous sauvegarderez la zone mémoire comprise entre #8000 et #8543. Le programme BASIC suivant permettra d'activer le debugger :

```

10 MEMORY &5000
20 LOAD"DEBUG.BIN"
30 CALL &8000

```

2° méthode

Si vous préférez saisir le programme debugger sous BASIC, il vous faudra procéder en deux étapes :

1) Saisie d'un programme qui contient les codes HEXA du programme debugger et qui fabrique le programme binaire DEBUG.BIN (Cf. programme page suivante).

2) Activation du debugger par le même programme BASIC que celui décrit page suivante.

E. A. 3B. 2

1120 DATA C6,7,C6,30,CD,5A,BB,F1,E6,F,CB
 ,17,CB,17,CB,17,CB,17,CB,7,CB,7,CB,7,CB,
 7,FE,A,38,2,C6,7,C6,30,CD,5A,BB,C9,21,91
 ,80,CD,DB,80,3E,4,32,33,80,CD,1,82,3E,8,
 CD,5A,BB

```

1130 DATA 3E,20,CD,5A,BB,21,27,80,CD,4A,
      B2,32,27,80,21,29,80,CD,4A,82,6F,3A,27,B
      0,67,22,29,80,C9,0,0,0,0,0,0,0,0,0,0,0,
      0,0,0,0,0,0,0,0,0,0,0,0,0,0,F5,C5,D5,E5,C
      D,11,83,41

```

```

1140 DATA 11,0,CO,21,27,80,CD,77,BC,21,C
1,80,CD,DB,80,CD,97,82,3E,D,CD,5A,BB,3E,
A

```

1150 DATA CD,5A,BB,CD,83,BC,CD,7A,BC,E1,
D1,C1,F1,C3,0,80,21,BB,80,CD,DB,80,1,0,0
,21,27,80,CD,6,BB,FE,7F,28,18,FE,D,28,34
,F5,3E,8,CD,5A,BB,F1,CD,5A,BB,9,77,3,3E,
5F,CD,5A

1160 DATA BB,18,DE,79,B7,28,DA,B,3E,8,CD
 ,5A,BB,3E,20,CD,5A,BB,3E,8,CD,5A,BB,3E,8
 ,CD,5A,BB,3E,5F,CD,5A,BB,18,BE,C9,F5,C5,
 D5,E5,CD,11,83,41,11,0,C0,21,27,80,CD,8C
 ,BC,3E,D

1170 DATA CD,5A,BB,3E,A,CD,5A,BB,21,9C,B
0,CD,DB,80,CD,97,82,22,34,80,21,AB,80,CD
,DB,80,CD,97,82,22,36,80,2A,36,80,ED,5B,
34,80,37,3F,ED,52,22,36,80,2A,34,80,ED,5
B,36,80,ED

```
1180 DATA 4B,34,80,3E,2A,CD,98,BC,CD,8F,  
      BC,3E,D,CD,5A,BB,3E,A,CD,5A,BB,E1,D1,C1,  
      F1,C3,0,80,0,0,0,0,0,0,0,0,0,0,0,0,  
      CD,11,B4,CD,24,B4,C3,58,B1,C9,D,A,20,48,
```

40, 20, 20

1190 DATA 20,44,45,20,20,20,42,43,20,20,
20,41,46,20,20,20,52,49,A,A,A,D,20,49,58
,20,20,20,49,59,20,20

1200 DATA 20,53,50,20,20,20,50,43,B,B,D,
FF,0,0,0,0,E5,F5,21,DA,83,7E,FE,FF,28,6,
CD,5A,BB, 23,18,F5,F1,E1,C9,E5,F5,F5,7C,
CD,13,85,7D,CD,13,85,3E,20,CD,5A,BB,7A,C
D,13,85,7B

1210 DATA CD,13,85,3E,20,CD,5A,BB,78,CD,
13,85,79,CD,13,85,3E,20,CD,5A,BB,E1,7C,C
D,13,85,7D,CD,13,85,3E,20,CD,5A,BB,ED,S
F,CD,13,85,ED,57,CD,13,85,3E,D,CD,5A,BB,
3E,A,CD

1220 DATA 5A, BB, D9, 7C, D9, CD, 13, 85, D9, 7D,
D9, CD, 13, 85, 3E, 20, CD, 5A, BB, D9, 7A, D9, CD, 1
3, 85, D9, 7B, D9, CD, 13, 85, 3E, 20, CD, 5A, BB, D9
, 7B, D9, CD, 13, 85, D9, 79, D9, CD, 13, 85, 3E, 20,
CD, 5A, BB, 8

1230 DATA E5,F5,CD,13,85,E1,7D,CD,13,85,
B,E1,3E,D,CD,5A,8B,3E,A,CD,5A,8B,3E,A,CD
,5A,8B,DD,22,D,84,2A,D,84,7C,CD,13,85,7D
,CD,13,85,3E,20,CD,5A,8B,FD,22,D,84,2A,D
,84,7C,CD

1240 DATA 13,85,7D,CD,13,85,3E,20,CD,5A,
BB,ED,73,D,84,2A,D,84,7C,CD,13,85,7D,CD,
13,85,3E,20,CD,5A,BB,2A,36,80,7C,CD,13

1250 DATA B5,7D,CD,13,85,3E,D,CD,5A,BB,3
E,A,CD,5A,BB,F1,E1,C9,F5,E6,F0,1F,1F,1F,
1F,FE,A,3B,2,C6,7,C6,30,CD,5A,BB,F1,E6,F
,CB,17,CB,17,CB,17,CB,17,CB,7,CB,7,CB,7,
CB,7,FE,A

1260 DATA 3B,2,C6,7,C6,30,CD,5A,BB,C9

9/3

Jeux d'esprit

9/3.1

Jeu du taquin

Ce jeu a été inventé aux Etats-Unis le siècle dernier.

Quinze plaquettes se trouvent disposées sur un taquin de seize cases. La case vide permet de déplacer les plaquettes. Le but du jeu est d'obtenir le plus rapidement possible le taquin (classé par ordre alphabétique) suivant :

A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	

Le programme principal fait appel à deux sous-programmes :

1°) *Initialisation* :

— Mise en mémoire du tableau à reconstituer,

- Tirage aléatoire du taquin de départ,
- Définition de caractères graphiques pour l'affichage du taquin.

2°) Jeu :

- Affichage du taquin,
- Saisie d'une commande :

- Cette commande peut être une des quatre touches-flèches pour indiquer le sens du déplacement. L'appui sur une autre touche actionne un signal sonore et ne provoque aucune action.
- Le taquin courant est comparé au taquin à obtenir. Si quinze des seize plaquettes sont identiques, le jeu est fini.

Le programme écrit en BASIC est le suivant :

```

1000 GOSUB 2010 'Mélange des lettres
1010 GOSUB 3000 'Jeu
1020 END

2000 REM =====
2010 REM Initialisation et mélange des lettres du taquin de départ
2020 REM =====
2030 DIM T(16),T1$(4,4),T2$(4,4) 'Tableaux de travail (T et T1)
2040 REM 'et à reconstituer (T2)
2050 '
2060 REM Tableau à reconstituer
2070 '
2080 T2$(1,1)="A":T2$(1,2)="B":T2$(1,3)="C":T2$(1,4)="D"
2090 T2$(2,1)="E":T2$(2,2)="F":T2$(2,3)="G":T2$(2,4)="H"
2100 T2$(3,1)="I":T2$(3,2)="J":T2$(3,3)="K":T2$(3,4)="L"
2110 T2$(4,1)="M":T2$(4,2)="N":T2$(4,3)="O":T2$(4,4)=" "
2120 '
2130 REM Tirage aléatoire du taquin de départ
2140 '
2150 FOR I=1 TO 15
2160 A=65+INT(RND(1)*15) 'Tirage aléatoire entre 65 et 79
2170 N=0 'Indicateur de "tirage déjà fait"
2180 FOR J=1 TO I

```

```

2190     IF T(J)=A THEN N=1
2200     NEXT J
2210     IF N=1 THEN 2160 'Tirage deja fait
2220     T(I)=A 'Memorisation du tirage
2230     NEXT I
2240     FOR I=1 TO 4
2250         FOR J=1 TO 4
2260             T1$(I,J)=CHR$(T((I-1)*4+J))
2270         NEXT J
2280     NEXT I
2290 '
2300 'Definition du tableau affiche a l'ecran
2310 '
2320 A$=CHR$(150)+CHR$(154)+CHR$(158)+CHR$(154)+CHR$(158)+CHR$(154)+CHR$(158)+CHR$(154)+CHR$(156)
2330 B$=CHR$(151)+CHR$(154)+CHR$(159)+CHR$(154)+CHR$(159)+CHR$(154)+CHR$(159)+CHR$(154)+CHR$(157)
2340 C$=CHR$(147)+CHR$(154)+CHR$(155)+CHR$(154)+CHR$(155)+CHR$(154)+CHR$(155)+CHR$(154)+CHR$(153)
2350 D$=CHR$(149)+" "+CHR$(149)+" "+CHR$(149)+" "+CHR$(149)+" "+CHR$(149)
2360 '
2370 L=4:C=4 'Ligne et colonne de la case vide
2380 '
2390 RETURN
3000 REM =====
3010 REM                                     Jeu
3020 REM =====
3030 MODE 1:PRINT SPACE$(16);"JEU DU TAQUIN"
3040 GOSUB 3140 'Affichage du taquin
3050 GOSUB 3300 'Entree d'une commande
3060 IF FIN=0 THEN 3040
3070 RETURN

```



```
3100 REM -----
3110 REM           Affichage du taquin
3120 REM -----
3130 CLS
3140 LOCATE 16,10:PRINT A$:LOCATE 16,12:PRINT B$:LOCATE 16,14:PRINT B$:LOCATE 16
,16:PRINT B$:LOCATE 16,18:PRINT C$
3150 LOCATE 16,11:PRINT D$:LOCATE 16,13:PRINT D$:LOCATE 16,15:PRINT D$:LOCATE 16
,17:PRINT D$
3160 FOR I=1 TO 4
3170   FOR J=1 TO 4
3180     LOCATE 15+J*2,9+2*I:PRINT T1$(I,J)
3190   NEXT J
3200 NEXT I
3210 RETURN
3300 REM -----
3310 REM           Saisie d'une commande
3320 REM -----
3330 X$=INKEY$:IF X$="" THEN 3330 'Attente de l'appui sur une touche
3340 A=ASC(X$)
3350 IF A>243 OR A<240 THEN SOUND 1,100,30:GOTO 3300
3360 ON A-239 GOSUB 3510,3610,3710,3810
3370 NE=0:FIN=0
3380 FOR I=1 TO 4
3390   FOR J=1 TO 4
3400     IF T1$(I,J)=T2$(I,J) THEN NE=NE+1
3410   NEXT J
3420 NEXT I
3430 IF NE=15 THEN FIN=1
3440 RETURN
3500 REM -----
3510 'Deplacement vers le haut
3520 IF L=4 THEN SOUND 1,100,30:GOTO 3300 'Commande refusee
3530 X$=T1$(L+1,C):T1$(L,C)=X$:T1$(L+1,C)=" ":L=L+1
3540 RETURN
```

```
3600 REM -----
3610 'Deplacement vers le bas
3620 IF L=1 THEN SOUND 1,100,30:GOTO 3300 'Commande refusee
3630 X#=T1$(L-1,C):T1$(L,C)=X#:T1$(L-1,C)=" " :L=L-1
3640 RETURN
3700 REM -----
3710 'Deplacement vers la gauche
3720 IF C=4 THEN SOUND 1,100,30:GOTO 3300 'Commande refusee
3730 X#=T1$(L,C+1):T1$(L,C)=X#:T1$(L,C+1)=" " :C=C+1
3740 RETURN
3800 REM -----
3810 'Deplacement vers la droite
3820 IF C=1 THEN SOUND 1,100,30:GOTO 3300 'Commande refusee
3830 X#=T1$(L,C-1):T1$(L,C)=X#:T1$(L,C-1)=" " :C=C-1
3840 RETURN
```

Lignes 1000 à 1020 : Programme principal.

Lignes 2000 à 2390 : Initialisation.

Lignes 3000 à 3070 : Jeu.

Lignes 3100 à 3210 : Affichage du taquin.

Lignes 3300 à 3440 : Saisie d'une commande au clavier.

Lignes 3500 à 3840 : Calculs relatifs à chaque commande.

9/3.2

Renversé

Ce jeu de réflexion demande beaucoup de pratique pour être utilisé de façon optimale.

Le principe du jeu est simple : n lettres (n compris entre 2 et 26) sont affichées à l'écran dans le désordre ; il faut les afficher dans l'ordre alphabétique en retournant successivement les x premières (x compris entre 2 et n). Le jeu se décompose en deux phases :

1°) *Initialisation* :

- Choix de la dimension du jeu,
- Tirage aléatoire des lettres.

2°) *Jeu* :

- Affichage des lettres,
- Prise en compte de l'action du joueur,
- Test sur la fin du jeu.

Le programme écrit en BASIC est le suivant :

```

1000 REM =====
1010 REM                      Jeu du RENVERSE
1020 REM =====
1030 GOSUB 2000 'Initialisation
1040 GOSUB 3000 'Jeu
1050 END

2000 REM =====
2010 REM                      Initialisation
2020 REM =====
2030 MODE 1:PRINT"JEU DU RENVERSE":PRINT
2040 DIM T$(26),T2$(26) 'Tableaux de jeu
2050 PRINT:INPUT"Dimension du jeu (entre 2 et 26) 1°ID

```

```

2060 '
2070 'Tirage aleatoire du Jeu
2080 '
2090 FOR I=1 TO D
2100   A=65+INT(RND(1)*D) 'Une lettre comprise entre A et chr$(65+D)
2110   N=0 'Indicateur de tirage deja fait
2120   FOR J=1 TO I
2130     IF T$(J)=CHR$(A) THEN N=1
2140   NEXT J
2150   IF N=1 THEN 2100 'Tirage deja fait
2160   T$(I)=CHR$(A)
2170 NEXT I
2180 RETURN

3000 REM =====
3010 REM                               Jeu
3020 REM =====
3030 '
3040 'Affichage du jeu
3050 '
3060 CLS:LOCATE 1,10
3070 FOR I=1 TO D
3080   PRINT T$(I); " ";
3090 NEXT I
3100 IF FIN=1 THEN LOCATE 1,16:PRINT "Jeu réussi en";C;"coups":RETURN
3110 '
3120 'Action du Joueur
3130 '
3140 C=C+1 'Nombre de coups + 1
3150 LOCATE 1,20:PRINT "Nombre a retourner"
3160 INPUT "(a partir de la gauche)";NR
3170 IF NR>D THEN SOUND 1,100,30:GOTO 3150
3180 '

```

```
3190 'Prise en compte de l'action
3200 '
3210 FOR I=1 TO NR
3220   T2$(I)=T$(NR+1-I)
3230 NEXT I
3240 FOR I=1 TO NR
3250   T$(I)=T2$(I)
3260 NEXT I
3270 '
3280 'Test de la fin du jeu
3290 '
3300 BP=0 'Nombre de lettre(s) bien placee(s)
3310 FOR I=1 TO D
3320   IF T$(I)=CHR$(64+I) THEN BP=BP+1
3330 NEXT I
3340 IF BP=D THEN FIN=1
3350 GOTO 3030 'Boucle de jeu
```

Lignes 1000 à 1050 : Programme principal.

Lignes 2000 à 2180 : Initialisation.

Lignes 3000 à 3350 : Jeu.

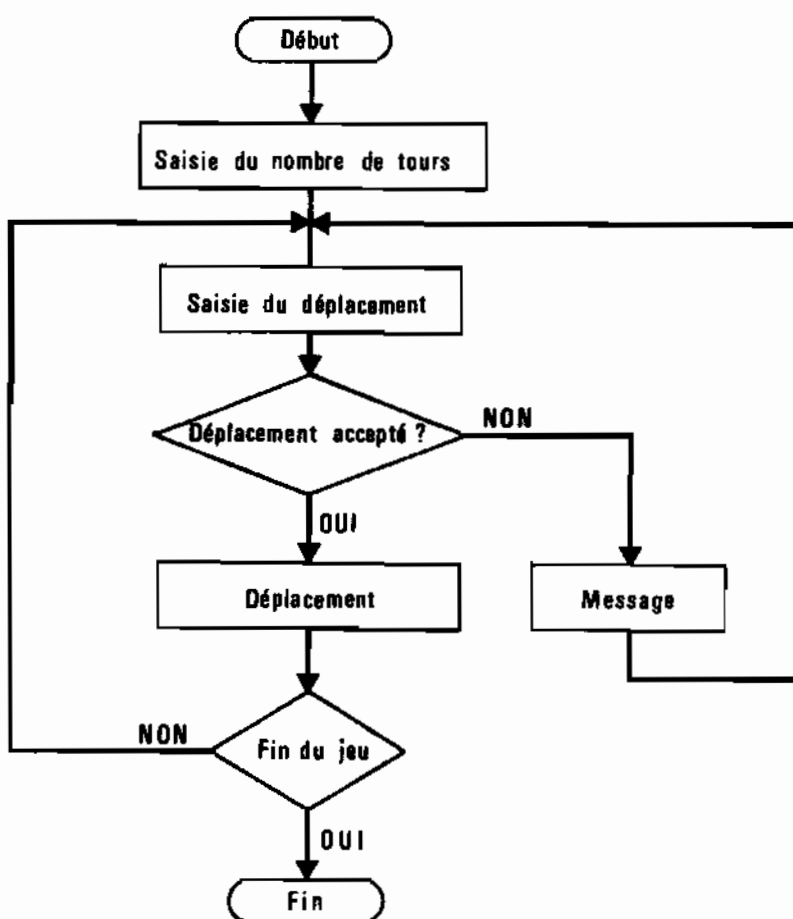
9/3.3

Tours de Hanoï

Ce jeu de réflexion met en œuvre des tours de diamètres différents empilées dans l'ordre le plus petit diamètre en haut, le plus grand en bas.

Ces tours sont empilées dans une des trois zones possibles. Le but du jeu consiste à reconstituer l'empilement de départ sur l'emplacement opposé en un minimum de coups. Le nombre de tours au départ est choisi par le joueur.

L'organigramme du programme est le suivant :



```

1000 MODE 1:PRINT"Tours de Hanoi"
1010 '=====
1020 'Initialisation
1030 '=====
1040 INPUT "Nombre de disques (entre 3 et 7)";ND
1050 IF ND>7 OR ND<3 THEN GOTO 1040
1060 '
1070 FOR J=0 TO 3
1080   FOR I=0 TO 8
1090     T(I,J)=0
1100   NEXT I
1110 NEXT J
1120 FOR I=1 TO ND
1130   T(I,1)=I
1140 NEXT I
1150 '=====
1160 'Partie
1170 '=====
1180 CLS
1190 FOR J=1 TO 3
1200   FOR I=1 TO ND
1210     IF T(I,J)=0 THEN 1270
1220     A=(J-1)*95+50-5*(8+T(I,J)-ND)
1230     B=150+12*(ND-I+1)
1240     C=(J-1)*95+60+5*(8+T(I,J)-ND)
1250     D=B+12
1260     MOVE A,B:DRAW C,B:DRAW C,D:DRAW A,D:DRAW A,B
1270   NEXT I
1280 NEXT J
1290 '
1300 INPUT"Tour de depart ";TD
1310 INPUT"Tour d'arrivee ";TA
1320 '
1330 IF TA>3 OR TD>3 OR TA<1 OR TD<1 THEN OK=1:GOTO 1420
1340 I=ND
1350 IF T(I,TA)<>0 THEN I=I-1:GOTO 1350
1360 J=ND
1370 IF T(J,TD)<>0 THEN J=J-1:GOTO 1370
1380 J=J+1:OK=1
1390 IF I=ND AND J<=ND THEN OK=-1:GOTO 1420
1400 IF J>ND THEN OK=1:GOTO 1420
1410 IF T(I+1,TA)>T(J,TD) THEN OK=-1
1420 IF OK=1 THEN PRINT"Deplacement non accepte":GOTO 1300
1430 '
1440 T(I,TA)=T(J,TD):T(J,TD)=0
1450 '
1460 BP=0
1470 FOR I=1 TO ND
1480   IF T(I,3)=I THEN BP=BP+1
1490 NEXT I
1500 NC=NC+1
1510 IF BP<>ND THEN 1180
1520 '=====
1530 'Fin de partie
1540 '=====
1550 PRINT"Tours de Hanoi deplacees en ";NC;"coups"

```

Lignes 1000 à 1140 : Initialisation des tours dans leur position initiale.

Lignes 1150 à 1510 : Jeu

Saisie du déplacement : Lignes 1300 et 1310

Test de cohérence : Lignes 1330 à 1410

Comptabilisation
du déplacement : Lignes 1460 à 1500

Lignes 1520 à 1550 : Affichage des résultats

9/3.4

Jeu des allumettes

Vous jouez contre l'ordinateur. Un certain nombre d'allumettes (choisi par vous) se trouve sur une table. Vous avez le droit d'en retirer une, deux ou trois. L'ordinateur de même. Celui qui retire la dernière allumette a gagné.

```

1000 CLS:PRINT"JEU DES ALLUMETTES"
1010 PRINT:PRINT"Celui qui retire la ou les dernieres a gagne."
1020 '=====
1030 'Instructions de debut de jeu
1040 '=====
1050 PRINT"Les prelevements possibles sont 1, 2 ou 3 allumettes."
1060 PRINT:INPUT"Nombre d'allumettes au depart ";N
1070 T(1)=1:T(2)=2:T(3)=3:T(4)=1:T(5)=1:T(6)=2:T(7)=3
1080 '=====
1090 'Jeu
1100 '=====
1110 INPUT"Combien en prenez-vous ";U
1120 IF U<1 OR U>3 THEN PRINT"Impossible: entre 1 et 3 !":GOTO 1110
1130 N=N-U
1140 IF N=0 THEN G=1:GOTO 1230
1150 PRINT"Il en reste ";N
1160 IF N>7 THEN T=INT(RND*3)+1 ELSE T=T(N)
1170 PRINT"J'en prends "T
1180 N=N-T
1190 PRINT"Il en reste ";N
1200 IF N=0 THEN G=0:GOTO 1230
1210 GOTO 1110 'Boucle de jeu
1220 '
1230 IF G=0 THEN PRINT"J'ai gagne" ELSE PRINT"Vous avez gagne"
1240 END

```

Lignes 1000 à 1070 : Initialisation, instructions du jeu

Lignes 1080 à 1240 : Jeu

Saisie du nombre d'allumettes
à déplacer :

Ligne 1110

Test de cohérence :

Ligne 1120

Jeu de l'ordinateur :

Ligne 1160

9/3.5

Awari

Ce jeu de réflexion se joue contre l'ordinateur. Le principe du jeu est simple : un circuit sépare l'écran en deux camps. Le joueur se trouve en bas, et l'ordinateur en haut. Les nombres présents sur les côtés indiquent les scores joueur (à droite) et ordinateur (à gauche). Au départ, toutes les cases sont à 3. Grâce aux touches-flèches, choisissez un des nombres de votre camp, puis appuyez sur la lettre « T ». Cette case passe à 0. Si son contenu était n, les n cases situées en aval (dans une rotation de sens inverse des aiguilles d'une montre) sont incrémentées de 1.

Pour gagner, il faut :

- annuler toutes les cases de son camp,
- avoir une case total supérieure à celle de l'ordinateur.

```

1000 '=====
1010 ' Présentation du jeu
1020 '=====
1030 MODE 2:PRINT"AWARI"
1040 PRINT:PRINT"Le but du jeu est d'annuler le nombre de points dans son camp"
1050 PRINT"en tenant compte des regles suivantes:"
1060 PRINT"1) Jouer une case qui contient n pions provoque n additions"
1070 PRINT"   d'un pion sur les n cases situees apres cette case (dans le"
1080 PRINT"   sens inverse des aiguilles d'une montre)"
1090 PRINT"2) Si la derniere addition tombe sur une case total, a droite ou a ga
uche,"
1100 PRINT"le joueur rejoue"
1110 PRINT:PRINT"Appuyez sur une touche quelconque pour commencer a jouer"
1120 A$=INKEY$:IF A$="" THEN 1120
1130 '-----
1140 'Initialisation du jeu
1150 '-----
1160 'Affichage du jeu
1170 '-----
1180 MODE 1
1190 FOR I=0 TO 1
1200   FOR J=0 TO 5
1210     LOCATE 9+J*4,12+I*6:PRINT"3"
1220   NEXT J
1230 NEXT I
1240 LOCATE 5,15:PRINT"0":LOCATE 33,15:PRINT"0"
1250 '
1260 DIM T(14)
1270 FOR I=1 TO 14
1280   T(I)=3
1290 NEXT I
1300 T(1)=0:T(8)=0

```

```

1310 -----
1320 'Derolement du jeu
1330 '-----
1340 GOSUB 1430 'Joueur
1350 IF FIN=1 THEN GOSUB 2410
1360 IF BIS=1 THEN 1340
1370 GOSUB 1880 'Amstrad
1380 IF BIS=1 THEN 1370
1390 IF FIN=0 THEN 1320
1400 GOSUB 2410
1410 END
1420 '-----
1430 'Jeu du joueur
1440 '-----
1450 LOCATE 9,20:PRINT CHR$(196):PX=0:BIS=0:SO=0
1460 LOCATE 1,5:PRINT"A vous de jouer"
1470 LOCATE 1,22:PRINT"Servez-vous des touches fleches droite"
1480 PRINT"et gauche pour positionner la case"
1490 PRINT"desiree. Appuyez sur T quand vous "
1500 PRINT"serez positionne."
1510 B$=INKEY$:IF B$="" THEN 1510
1520 B=ASC(B$)
1530 IF B=243 THEN GOSUB 1590 'Vers la droite
1540 IF B=242 THEN GOSUB 1670 'Vers la gauche
1550 IF B$="T" THEN GOSUB 1760
1560 IF B$="T" THEN RETURN
1570 GOTO 1510
1580 '.....
1590 'Deplacement vers la droite
1600 '.....
1610 IF PX=5 THEN RETURN
1620 LOCATE 9+PX*4,20:PRINT" "
1630 PX=PX+1
1640 LOCATE 9+PX*4,20:PRINT CHR$(196)
1650 B=0
1660 RETURN :
1670 '.....
1680 'Deplacement vers la gauche
1690 '.....
1700 IF PX=0 THEN RETURN
1710 LOCATE 9+PX*4,20:PRINT" "
1720 PX=PX-1
1730 LOCATE 9+PX*4,20:PRINT CHR$(196)
1740 B=0
1750 RETURN
1760 '
1770 LOCATE 1,5:PRINT SPACE$(20)
1780 LOCATE 1,20:PRINT SPACE$(235)
1790 '.....
1800 'Calcul et affichage du coup joue
1810 '.....
1820 JEU=0:GOSUB 2140:SO=0
1830 FOR I=2 TO 7
1840   SO=SO+T(I)
1850 NEXT I
1860 IF SO=0 THEN FIN=1:GA=1
1870 RETURN
1880 '-----
1890 'Jeu Amstrad
1900 '-----
1910 MX=-100:BIS=0:SO=0
1920 '

```

```

1930 FOR I=9 TO 14
1940   C=0:D=0
1950   IF T(I)=0 THEN 2020
1960   FOR J=1 TO T(I)
1970     D=I+J
1980     IF D<15 AND D>8 THEN C=C-1
1990     IF D>1 AND D<8 THEN C=C+1
2000   NEXT J
2010   IF C>MX THEN MX=C:CASE=I
2020 NEXT I
2030 PX=CASE-2
2040 '
2050 LOCATE 1,5:PRINT"Je joue en "
2060 JEU=1:GOSUB 2140
2070 LOCATE 1,5:PRINT SPACE$(20):S0=0
2080 '
2090 FOR I=9 TO 14
2100   S0=S0+T(I)
2110 NEXT I
2120 IF S0=0 THEN FIN=1:GA=0
2130 RETURN
2140 '.....
2150 'Calcul et affichage du coup joue
2160 '.....
2170 A=PX+2:IF T(A)=0 THEN RETURN
2180 FOR I=1 TO T(A)
2190   IF A+I<=14 THEN T(A+I)=T(A+I)+1
2200   IF A+I>14 THEN T(A+I-14)=T(A+I-14)+1
2210 NEXT I
2220 IF A+I=9 OR A+I=16 THEN BIS=1
2230 T(A)=0
2240 IF JEU=1 THEN LOCATE 9+(14-A)*4,10:PRINT CHR$(198):FOR I=1 TO 500:NEXT
2250 '
2260 FOR I=0 TO 5
2270   LOCATE 8+I*4,12:PRINT T(14-I);
2280 NEXT I
2290 '
2300 FOR I=0 TO 5
2310   LOCATE 8+I*4,18:PRINT T(I+2);
2320 NEXT I
2330 '
2340 LOCATE 4,15:PRINT T(1)
2350 '
2360 B=INT(T(8)/10)
2370 LOCATE 32,15:PRINT T(8)
2380 '
2390 IF JEU=1 THEN LOCATE 9+(14-A)*4,10:PRINT" "
2400 RETURN
2410 '-----
2420 'Partie terminee
2430 '-----
2440 CLS
2450 PRINT"Partie terminee"
2460 PRINT:PRINT:PRINT
2470 IF GA=1 THEN PRINT"Bravo, vous avez gagne " ELSE PRINT"Vous avez perdu "
2480 IF T(1)>T(8) THEN PRINT "et vous etes battu sur le nombre de points "; ELSE
PRINT"et vous gagnez au nombre de points ";
2490 PRINT "(";T(1);"contre";T(8);")"

```

- Lignes 1000 à 1120 : Présentation du jeu et rappel des règles
- Lignes 1150 à 1300 : Affichage du parcours initial
- Lignes 1310 à 1410 : Programme principal qui oriente vers les sous-programmes joueur et ordinateur
- Lignes 1420 à 1570 : Action du joueur
- Lignes 1580 à 1660 : Déplacement du curseur vers la droite
- Lignes 1670 à 1780 : Déplacement du curseur vers la gauche
- Lignes 1790 à 1870 : Détermine si la partie est gagnée
- Lignes 1880 à 2130 : Jeu de l'ordinateur
- Lignes 2140 à 2400 : Répercussions du jeu de l'ordinateur sur le parcours et calcul si la partie est terminée
- Lignes 2410 à 2490 : Commentaires de fin de partie

9/3.6

Jeu du Simon

Introduction

Ce jeu fait appel à votre mémoire à court terme. Un symbole lumineux apparaît sur l'écran. La position de ce symbole par rapport au centre de l'écran désigne une des touches flèches. Appuyez sur cette touche pour le reproduire. Deux symboles lumineux sont maintenant affichés. Le premier est celui qui est apparu lors du premier affichage. Le second est aléatoire. Le nombre de symboles augmente chaque fois que vous arrivez à les restituer dans le bon ordre avec les touches flèches. La partie prend fin lorsque les touches restituées ne correspondent pas à celles affichées par le programme.

Comment utiliser le programme

Quel que soit le langage utilisé, lancez le programme et appuyez successivement sur les touches flèches correspondant aux symboles lumineux affichés sur l'écran.

Le programme en détails

VERSION BASIC

Le listing du programme est le suivant :

```

1000 REM -----
1010 REM Jeu du Simon
1020 REM -----
1030 '
1040 GOSUB 2000 'Presentation du jeu
1050 GOSUB 3000 'Initialisation du jeu
1060 GOSUB 4000 'Deroulement du jeu
1070 GOSUB 7000 'Affichage du score
1080 END
1090 '
2000 REM -----
2010 REM Presentation du jeu
2020 REM -----
2030 '
2040 CLS
2050 PRINT "Simon"
2060 PRINT "-----"
2070 PRINT
2080 PRINT "L'ordinateur va afficher une suite de symboles l
umineux."
2090 PRINT "Vous devrez restituer les symboles dans le bon o
rdre en"
2100 PRINT "utilisant les touches fleches du clavier."
2110 PRINT
2120 PRINT "Toute bonne reponse augmente d'un le nombre de sy
mboles"
2130 PRINT "affiches au prochain coup. Toute mauvaise reponse
annule"
2140 PRINT "la partie."
2150 PRINT
2160 PRINT "Appuyez sur une touche pour commencer a jouer..."
2170 a$=INKEY$:IF a$="" THEN I=I+1:GOTO 2170
2180 RETURN
2190 '
3000 REM -----
3010 REM Initialisation des variables du jeu
3020 REM -----
3030 '
3035 DIM T(50),TU(50)
3040 St1$=CHR$(127)+CHR$(127)+CHR$(127) 'le tiers d'une cas
e
3050 St2$=" " 'chaîne d'effacement
3060 N=0 'Nombre de tirages
3070 RETURN
3080 '
4000 '-----
4010 ' Procedure principale de jeu
4020 '-----
4030 '
4040 RANDOMIZE i

```

```

4050 Fin=0
4060 WHILE Fin=0
4070   CLS
4080   N=N+1 'Nombre de tirages + 1
4090   A=INT(RND(1)*4)+1
4100   T(N)=A
4110   '
4120   'Affichage des delimitateurs
4130   LOCATE 40,12
4140   PRINT CHR$(244)
4150   LOCATE 39,13
4160   PRINT CHR$(247);" ";CHR$(246)
4170   LOCATE 40,14
4180   PRINT CHR$(245)
4190   '
4200   'Affichage des touches en memoire
4210   FOR I=1 TO N
4220     A=T(I)
4230     GOSUB 5000 'Affichage d'une des cases
4240     GOSUB 6000 'Temporisation
4250   NEXT I
4260   '
4270   LOCATE 1,20
4280   PRINT "Retapez la sequence apparue sur l'ecran..."
4290   FOR i=1 TO n
4300     PRINT"*";
4310   NEXT i
4320   PRINT
4330   '
4340   'Lecture des touches tapees par le joueur
4350   FOR i=1 TO n
4360     a$=INKEY$:IF a$="" THEN 4360
4370     IF ASC(a$)=240 THEN A=1:GOTO 4410
4380     IF ASC(a$)=243 THEN A=2:GOTO 4410
4390     IF ASC(a$)=241 THEN A=3:GOTO 4410
4400     IF ASC(a$)=242 THEN A=4
4410     tu(i)=a 'Memorisation de la touche tapee
4420     GOSUB 5000 'Affichage de la touche choisie
4430     LOCATE i+1,22 'Repositionnement du curseur
4440   NEXT i
4450   '
4460   'Comparaison
4470   Fin=0
4480   FOR i=1 TO N
4490     IF T(i) <> TU(i) THEN fin=1
4500   NEXT i
4510 WEND
4520 RETURN
4530 '
5000 REM -----
5010 REM Affichage des cases sur l'ecran
5020 REM -----
5030 '
5040 ON a GOSUB 5100,5200,5300,5400
5050 RETURN
5060 '
5100 LOCATE 39,9:PRINT St1$

```

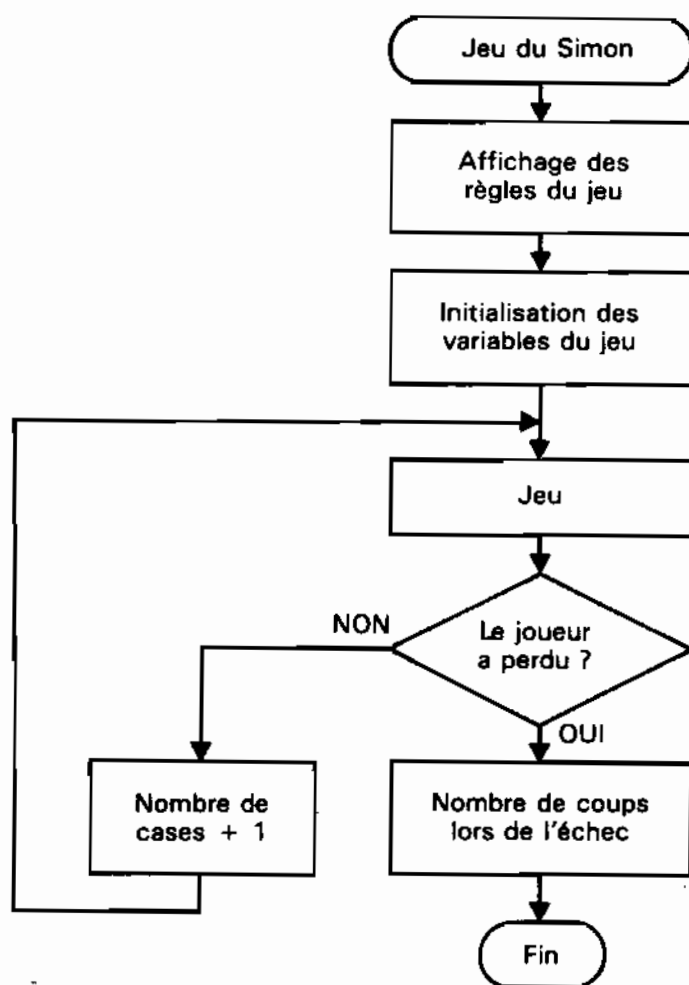


```

5110 LOCATE 39,10:PRINT St1$
5120 LOCATE 39,11:PRINT St1$
5130 GOSUB 6000 'Temporisation
5140 LOCATE 39,9:PRINT St2$
5150 LOCATE 39,10:PRINT St2$
5160 LOCATE 39,11:PRINT St2$
5170 RETURN
5180 '
5200 LOCATE 42,12:PRINT St1$
5210 LOCATE 42,13:PRINT St1$
5220 LOCATE 42,14:PRINT St1$
5230 GOSUB 6000 'Temporisation
5240 LOCATE 42,12:PRINT St2$
5250 LOCATE 42,13:PRINT St2$
5260 LOCATE 42,14:PRINT St2$
5270 RETURN
5280 '
5300 LOCATE 39,15:PRINT St1$
5310 LOCATE 39,16:PRINT St1$
5320 LOCATE 39,17:PRINT St1$
5330 GOSUB 6000 'Temporisation
5340 LOCATE 39,15:PRINT St2$
5350 LOCATE 39,16:PRINT St2$
5360 LOCATE 39,17:PRINT St2$
5370 RETURN
5380 '
5400 LOCATE 36,12:PRINT St1$
5410 LOCATE 36,13:PRINT St1$
5420 LOCATE 36,14:PRINT St1$
5430 GOSUB 6000 'Temporisation
5440 LOCATE 36,12:PRINT St2$
5450 LOCATE 36,13:PRINT St2$
5460 LOCATE 36,14:PRINT St2$
5470 RETURN
5480 '
6000 REM -----
6010 REM Temporisation entre deux affichages
6020 REM -----
6030 '
6040 FOR L=1 TO 200
6050 NEXT L
6060 RETURN
6070 '
7000 REM -----
7010 REM Le joueur a fait une erreur ou plus
7020 REM La partie est finie.
7030 REM -----
7040 '
7050 LOCATE 1,23
7060 PRINT "Vous avez perdu au ";N;"eme coup."
7070 RETURN

```

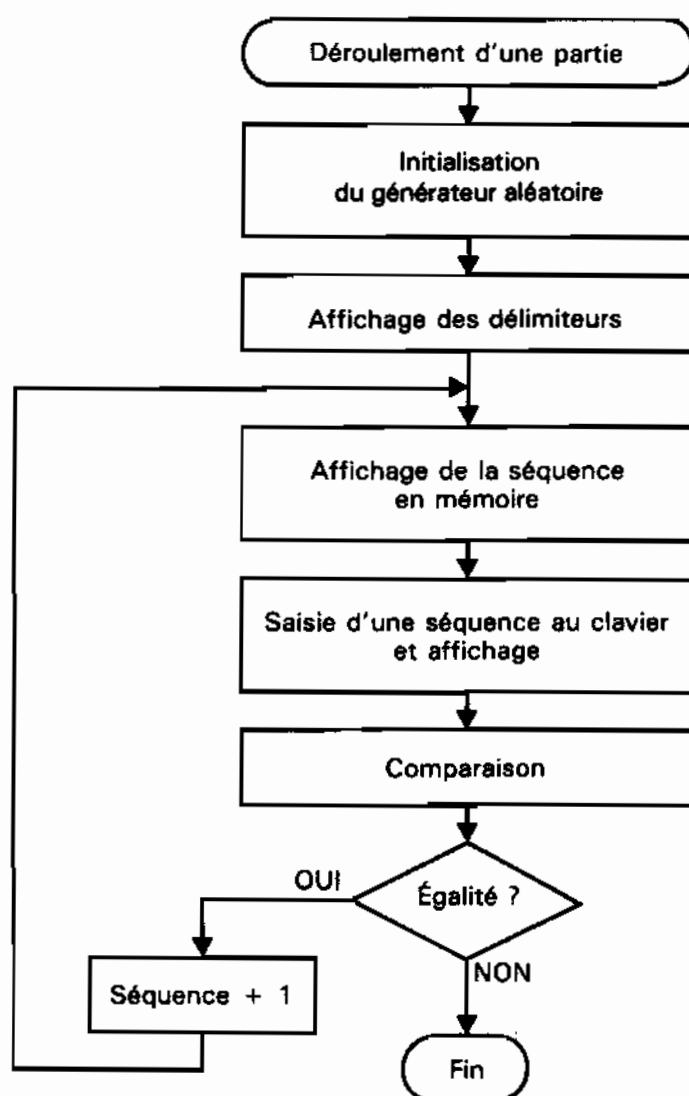
La logique du programme principal apparaît dans l'ordinogramme suivant :



Le sous-programme de présentation qui occupe les lignes 2000 à 2180 informe le joueur du but et de la règle du jeu.

Le sous-programme d'initialisation (lignes 3000 à 3070) consiste en la définition des tableaux de jeu et le prépositionnement des variables utilisées par le sous-programme principal.

Le déroulement d'une partie se fait selon la logique de l'ordinogramme ci-dessous :



La case lumineuse affichée sur l'écran est choisie aléatoirement grâce à la formule suivante :

INT (RND(1)*4) + 1

Les cases sont constituées de 9 caractères de codes ASCII 127 accolés trois par trois. Leur affichage, puis leur effacement se fait à l'aide d'une suite d'instructions LOCATE/PRINT dans le sous-programme situé entre les lignes 5000 et 5470 :

LOCATE 39,9:PRINT St1\$ 'Motif affiché

...

LOCATE 39,11:PRINT St2\$ 'Chaîne d'effacement

L'acquisition de la séquence tapée par le joueur se fait à l'aide d'une boucle de lecture sur l'instruction **INKEY\$**:

a\$ = INKEY\$:IF a\$ = " " then 4360

La variable **Fin** est initialisée à un si la séquence entrée par le joueur est différente de la séquence visualisée :

Fin = 0

FOR i = 1 TO N

IF T(i) <> TU(i) THEN Fin = 1

NEXT i

Le temps d'affichage des cases lumineuses est fixé à l'aide d'une boucle **FOR** vide :

FOR L = 1 TO 200

NEXT L

Enfin, le nombre de cases affichées lorsque le joueur a commis une erreur est affiché à l'aide du sous-programme situé entre les lignes 7000 et 7070.

VERSION TURBO-PASCAL

Le listing du programme est le suivant :

```

Program Simon;
(=====)
( JEU DU SIMON )
(=====)

Const Up = 240;           ( Touche vers le haut           )
      Down = 241;         ( Touche vers le bas           )
      Left = 242;         ( Touche vers la gauche        )
      Right = 243;        ( Touche vers la droite       )

Var N,                    ( Nombre de cases a afficher   )
    A,                    ( Variable intermediaire     )
    L,                    ( Index de boucles                 )
    K,                    ( Index de boucles                 )
    J,                    ( Index de boucles                 )
    I : Integer;          ( Index de boucles                 )
    St1,                  ( Motif du Simon              )
    St2 : String[2];      ( Effacement du Simon        )
    T,                    ( Memorisation des touches       )
    TU : Array[1..50] of Integer; ( Touches tapées par le joueur )
    Fin : Boolean;        ( Indicateur de fin de jeu    )
    Ch : Char;

Procedure Presentation;
(-----)
( Presentation du jeu )
(-----)

begin
  ClrScr;
  Writeln('Simon');
  Writeln('-----');
  Writeln;
  Writeln('L'ordinateur va afficher une suite de symboles lumineux. ');
  Writeln('Vous devrez restituer les symboles dans le bon ordre. ');
  Writeln('Utilisant les touches flèches du clavier. ');
  Writeln;
  Writeln('Toute bonne reponse augmente d'un le nombre de symboles ');
  Writeln('affiches au prochain coup. Toute mauvaise reponse annule ');
  Writeln('la partie. ');
  Writeln;
  Write('Appuyez sur une touche pour commencer a jouer... ');
  While not Keypressed do
    Read(Kbd,Ch);
end;

```

```
Procedure Tempo;  
{-----}  
{ Temporisation entre deux affichages }  
{-----}  
  
begin  
  For L:=1 to 10000 do;  
end;  
  
Procedure Affiche(Var A:Integer);  
{-----}  
{ Affichage des cases sur l'ecran }  
{-----}  
  
begin  
  Case A of  
    1 : begin  
      ( En haut )  
      GotoXY(39,9);  
      Write(St1);  
      GotoXY(39,10);  
      Write(St1);  
      GotoXY(39,11);  
      Write(St1);  
      Tempo;  
      GotoXY(39,9);  
      Write(St2);  
      GotoXY(39,10);  
      Write(St2);  
      GotoXY(39,11);  
      Write(St2);  
    end;  
  
    2 : begin  
      ( A droite )  
      GotoXY(42,12);  
      Write(St1);  
      GotoXY(42,13);  
      Write(St1);  
      GotoXY(42,14);  
      Write(St1);  
      Tempo;  
      GotoXY(42,12);  
      Write(St2);  
      GotoXY(42,13);  
      Write(St2);  
      GotoXY(42,14);  
      Write(St2);  
    end;  
  
    3 : begin  
      ( En bas )  
      GotoXY(39,15);  
      Write(St1);  
      GotoXY(39,16);  
      Write(St1);  
      GotoXY(39,17);  
      Write(St1);
```

```

    Tempo;
    GotoXY(39,15);
    Write(St2);
    GotoXY(39,16);
    Write(St2);
    GotoXY(39,17);
    Write(St2);
end;

```

```

4 : begin                                { A gauche }
    GotoXY(36,12);
    Write(St1);
    GotoXY(36,13);
    Write(St1);
    GotoXY(36,14);
    Write(St1);
    Tempo;
    GotoXY(36,12);
    Write(St2);
    GotoXY(36,13);
    Write(St2);
    GotoXY(36,14);
    Write(St2);
end;
end;
end;

```

```

Procedure Initialisation;
{-----}
{ Initialisation des variables du jeu }
{-----}

```

```

begin
    St1:=chr(127)+chr(127)+chr(127); { le tiers d'une case }
    St2:='';                          { chaine d'effacement }
    N:=0;                             { Nombre de tirages }
end;

```

```

Procedure Jeu;
{-----}
{ Procedure principale de jeu }
{-----}

```

```

begin
    Randomize;
    Repeat
        Clrscr;
        N:=N+1; { Nombre de tirages + 1 }
        A:=Round(Int(Random * 4))+1;
        T[N]:=A;

        { Affichage des delimitateurs }
        GotoXY(40,12);
        Write(chr(244));
    
```

```

GotoXY(39,13);
Write(chr(247),' ',chr(246));
GotoXY(40,14);
Write(chr(245));

{ Affichage des touches en memoire }
For I:=1 to N do
begin
  A:=T[I];
  Affiche(A); { Affichage d'une des cases }
  Tempo;
end;

GotoXY(1,20);
Writeln('Retapez la sequence apparue sur l''ecran ...');
For I:=1 to N do
  Write('*');
Writeln;

{ Lecture des touches tapees par le joueur }
For I:=1 to N do
begin
  While not Keypressed do;
  Read(Kbd,Ch);
  Case Ord(Ch) Of
    Up      : A:=1;
    Down    : A:=3;
    Left    : A:=4;
    Right   : A:=2;
  end;
  TUI:=A; { Memorisation de la touche tapee }
  Affiche(A); { Affichage de la touche choisie }
  GotoXY(I+1,22); { Repositionnement du curseur }
end;

{ Comparaison }
Fini:=False;
For I:=1 to N do
  If T[I] <> TUI then Fini:=True;

until Fini; { Le joueur a perdu }
end;

Procedure Fini;
{-----}
{ Le joueur a fait une erreur ou plus }
{ La partie est finie. }
{-----}

begin
  GotoXY(1,23);
  Writeln('Vous avez perdu au ',N,'eme coup. ');
end;

```



```

                                {=====}
                                { PROGRAMME PRINCIPAL }
                                {=====}

begin
  Presentation;                { Presentation du jeu      }
  Initialisation;              { Initialisation du jeu  }
  Jeu;                          { Deroulement d'une partie }
  Fini;                         { Affichage du score    }
end.

```

Le découpage hiérarchique de ce programme est similaire à celui du programme Basic au détail près suivant : les sous-programmes Basic sont remplacés par des procédures Turbo-Pascal.

Remarquez en particulier dans ce programme :

- la nécessité d'utiliser une boucle de temporisation bien plus longue pour obtenir le même résultat qu'en Basic : **For L:= 1 to 10000 do;**
- l'utilisation judicieuse d'une instruction **CASE** pour afficher les cases sur l'écran.

VERSION ASSEMBLEUR

Le listing du programme est le suivant :

```

1                                ORG  9000H
2                                LOAD 9000H
3 9000 C38E94                    JP   SIMON                ;Execution
4                                ;
5                                ;=====
6                                ; ZONE DES SOUS-PROGRAMMES
7                                ;=====
8                                ;
9                                ;-----
10                               ;Effacement de l'ecran
11                               ;-----
12                               CLS:          EQU  $
13 9003 3E02                      LD   A,2

```

```

14 9005 CD0EBC          CALL SETMODE          ;MODE 2
15 9008 C9              RET
16                      ;
17                      ;-----
18                      ; Deroulement d'une partie
19                      ;-----
20                      JEU:          EQU $
21 9009 CD0E90          CALL CLS                ;Effacement de l'ecran
22 900C 3A7E92          LD  A,(N)
23 900F 3C              INC  A
24 9010 327E92          LD  (N),A              ;Nombre de tirages + 1
25 9013 ED5F            LD  A,R              ;Nombre aleatoire
26 9015 E60F            AND  0FH
27 9017 FE04            CP   4
28 9019 380C            JR   C,UN
29 901B FE08            CP   8
30 901D 380C            JR   C,DEUX
31 901F FE0C            CP   12
32 9021 380C            JR   C,TROIS
33 9023 3E04            LD  A,4
34 9025 180A            JR   SUIJEU1
35
36                      UN:          EQU $
37 9027 3E01            LD  A,1
38 9029 1806            JR   SUIJEU1
39                      DEUX:        EQU $
40 902B 3E02            LD  A,2
41 902D 1802            JR   SUIJEU1
42                      TROIS:       EQU $

```

```

3 902F 3E03          LD  A,3
4
5          SUIJEU1:   EQU  $
6 9031 327F92        LD  (SA),A          ;Sauvegarde de A
7 9034 218292        LD  HL,T           ;@ du tabl des sequences
8 9037 1600          LD  D,0
9 9039 3A7E92        LD  A,(N),
0 903C 5F            LD  E,A
1 903D 19            ADD  HL,DE
2 903E 3A7F92        LD  A,(SA)
3 9041 77            LD  (HL),A
4
5          ; Affichage des delimitateurs
6 9042 3E28          LD  A,40
7 9044 CD6FBB        CALL SETCOL
8 9047 3E0C          LD  A,12
9 9049 CD72BB        CALL SETROW
0 904C 3EF4          LD  A,244
1 904E CD5ABB        CALL PRINT
2 9051 3E27          LD  A,39
3 9053 CD6FBB        CALL SETCOL
4 9056 3E0D          LD  A,13
5 9058 CD72BB        CALL SETROW
6 905B 3EF7          LD  A,247
7 905D CD5ABB        CALL PRINT
8 9060 3E20          LD  A," "
9 9062 CD5ABB        CALL PRINT
0 9065 3EF6          LD  A,246
1 9067 CD5ABB        CALL PRINT

```

```

72 906A 3E2B          LD    A,40
73 906C CD6FBB        CALL SETCOL
74 906F 3E0E          LD    A,14
75 9071 CD72BB        CALL SETROW
76 9074 3EF5          LD    A,245
77 9076 CD5ABB        CALL PRINT
78
79                    ;Affichage des touches en memoire
80 9079 218292        LD    HL,T
81 907C 3A7E92        LD    A,(N)
82 907F 47            LD    B,A
83 9080 3E01          LD    A,1
84                    BISAF: EQU $
85 9082 E5            PUSH HL
86 9083 F5            PUSH AF
87 9084 1600          LD    D,0
88 9086 5F            LD    E,A
89 9087 19            ADD  HL,DE
90 9088 7E            LD    A,(HL)
91 9089 328192        LD    (CAA),A          ;Carre a afficher
92 908C F1            POP  AF
93 908D E1            POP  HL
94 908E CD1E91        CALL AFTOUCH
95 9091 CD2092        CALL TEMPO
96 9094 B8            CP    B
97 9095 2803          JR    Z,AF2
98 9097 3C            INC  A
99 9098 18E8          JR    BISAF

```

```

1          AF2:      EQU  $
2 909A 211F94      LD  HL,MES1
3 909D 110114      LD  DE,1401H
4 90A0 CD4092      CALL AFFICH          ;Retapez la seq...
5
6 90A3 3A7E92      LD  A,(N)          ;Nombre de tirages
7 90A6 47          LD  B,A
8 90A7 3E01        LD  A,1
9          BISAF2:   EQU  $
10 90A9 F5         PUSH AF
11 90AA 3E2A        LD  A,"*"
12 90AC CD5ABB      CALL PRINT
13 90AF F1         POP  AF
14 90B0 B8         CP   B
15 90B1 2B03        JR   Z,AF3
16 90B3 3C         INC  A
17 90B4 1BF3        JR   BISAF2
18
19          AF3:      EQU  $
20 90B6 21A092      LD  HL,TU          ;Tableau utilisateur
21 90B9 3A7E92      LD  A,(N)          ;Nombre de tirages
22 90BC 47          LD  B,A
23 90BD 3E01        LD  A,1
24          AF4:      EQU  $
25 90BF 327F92      LD  (SA),A
26          TOUREF:   EQU  $
27 90C2 CD06BB      CALL WAIT
28 90C5 FEF0        CP   240
29 90C7 2B0E        JR   Z,TUP

```

130 90C9 FEF1		CP	241	
131 90CB 280E		JR	Z,TDOWN	
132 90CD FEF2		CP	242	
133 90CF 280E		JR	Z,TLEFT	
134 90D1 FEF3		CP	243	
135 90D3 280E		JR	Z,TRIGHT	
136 90D5 18EB		JR	TOUREF	;Touche refusee
137				
138	TUP:	EQU	\$	
139 90D7 3E01		LD	A,1	
140 90D9 180A		JR	AF5	
141	TDOWN:	EQU	\$	
142 90DB 3E03		LD	A,3	
143 90DD 1806		JR	AF5	
144	TLEFT:	EQU	\$	
145 90DF 3E04		LD	A,4	
146 90E1 1802		JR	AF5	
147	TRIGHT:	EQU	\$	
148 90E3 3E02		LD	A,2	
149				
150	AF5:	EQU	\$	
151 90E5 328092		LD	(MEMCAR),A	
152 90E8 328192		LD	(CAA),A	
153 90EB CD1E91		CALL	AFTOUCH	
154 90EE CD2092		CALL	TEMPO	
155 90F1 3A7F92		LD	A,(SA)	
156 90F4 1600		LD	D,0	
157 90F6 5F		LD	E,A	
158 90F7 E5		PUSH	HL	

```

9 90F8 19          ADD  HL,DE
0 90F9 3A0092      LD   A,(MEMCAR)
1 90FC 77          LD   (HL),A
2 90FD E1          POP  HL
3
4 90FE 3A7F92      LD   A,(SA)
5 9101 B8          CP   B
6 9102 2016        JR   NZ,ENCORE
7
8                ; Comparaison de T et TU
9 9104 218292      LD   HL,T
0 9107 11A092      LD   DE,TU
1 910A 23          INC  HL
2 910B 13          INC  DE
3 910C 3A7E92      LD   A,(N)
4 910F 47          LD   B,A
5                BC1: EQU  $
6 9110 1A          LD   A,(DE)
7 9111 BE          CP   (HL)
8 9112 C0          RET  NZ                ; Fin du jeu
9 9113 13          INC  DE
0 9114 23          INC  HL
1 9115 10F9        DJNZ BC1
2 9117 C30990      JP   JEU                ; Poursuite du jeu
3
4                ENCORE: EQU  $
5 911A 3C          INC  A
6 911B C3BF90      JP   AF4
7

```

```

188          AFTOUCH:    EQU  $
189 911E E5              PUSH HL
190 911F C5              PUSH BC
191 9120 F5              PUSH AF
192 9121 3A8192          LD  A,(CAA)          ;Carre a afficher
193 9124 FE01            CP   1
194 9126 2845            JR   Z,AFT1
195 9128 FE02            CP   2
196 912A 287C            JR   Z,AFT2
197 912C FE03            CP   3
198 912E CAE391          JP   Z,AFT3
199
200          AFT4:        EQU  $          ;Vers la droite
201 9131 218694          LD   HL,ST1
202 9134 11220C          LD   DE,0C22H
203 9137 CD4092          CALL AFFICH
204 913A 218694          LD   HL,ST1
205 913D 11220D          LD   DE,0D22H
206 9140 CD4092          CALL AFFICH
207 9143 218694          LD   HL,ST1
208 9146 11220E          LD   DE,0E22H
209 9149 CD4092          CALL AFFICH
210 914C CD2092          CALL TEMPO
211 914F 218A94          LD   HL,ST2
212 9152 11220C          LD   DE,0C22H
213 9155 CD4092          CALL AFFICH
214 9158 218A94          LD   HL,ST2
215 915B 11220D          LD   DE,0D22H
216 915E CD4092          CALL AFFICH

```



```
9161 218A94      LD  HL,ST2
9164 11220E      LD  DE,0E22H
9167 CD4092      CALL AFFICH
916A C31C92      JP   FINAFFI
```

AFT1: EQU \$;Vers le haut

```
916D 218694      LD  HL,ST1
9170 112709      LD  DE,927H
9173 CD4092      CALL AFFICH
9176 218694      LD  HL,ST1
9179 11270A      LD  DE,0A27H
917C CD4092      CALL AFFICH
917F 218694      LD  HL,ST1
9182 11270B      LD  DE,0B27H
9185 CD4092      CALL AFFICH
9188 CD2092      CALL TEMPO
918B 218A94      LD  HL,ST2
918E 112709      LD  DE,927H
9191 CD4092      CALL AFFICH
9194 218A94      LD  HL,ST2
9197 11270A      LD  DE,0A27H
919A CD4092      CALL AFFICH
919D 218A94      LD  HL,ST2
91A0 11270B      LD  DE,0B27H
91A3 CD4092      CALL AFFICH
91A6 1874        JR   FINAFFI
```

AFT2: EQU \$;Vers le bas

```
91AB 218694      LD  HL,ST1
```

246	91AB	112A0C	LD	DE,0C2AH	
247	91AE	CD4092	CALL	AFFICH	
248	91B1	218694	LD	HL,ST1	
249	91B4	112A0D	LD	DE,0D2AH	
250	91B7	CD4092	CALL	AFFICH	
251	91BA	218694	LD	HL,ST1	
252	91BD	112A0E	LD	DE,0E2AH	
253	91C0	CD4092	CALL	AFFICH	
254	91C3	CD2092	CALL	TEMPO	
255	91C6	218A94	LD	HL,ST2	
256	91C9	112A0C	LD	DE,0C2AH	
257	91CC	CD4092	CALL	AFFICH	
258	91CF	218A94	LD	HL,ST2	
259	91D2	112A0D	LD	DE,0D2AH	
260	91D5	CD4092	CALL	AFFICH	
261	91D8	218A94	LD	HL,ST2	
262	91DB	112A0E	LD	DE,0E2AH	
263	91DE	CD4092	CALL	AFFICH	
264	91E1	1B39	JR	FINAFFI	
265					
266		AFT3:	EQU	*	;Vers la gauche
267	91E3	218694	LD	HL,ST1	
268	91E6	11270F	LD	DE,0F27H	
269	91E9	CD4092	CALL	AFFICH	
270	91EC	218694	LD	HL,ST1	
271	91EF	112710	LD	DE,1027H	
272	91F2	CD4092	CALL	AFFICH	
273	91F5	218694	LD	HL,ST1	
274	91F8	112711	LD	DE,1127H	

```

91FB CD4092      CALL AFFICH
91FE CD2092      CALL TEMPO
9201 218A94      LD   HL,ST2
9204 11270F      LD   DE,0F27H
9207 CD4092      CALL AFFICH
920A 218A94      LD   HL,ST2
920D 112710      LD   DE,1027H
9210 CD4092      CALL AFFICH
9213 218A94      LD   HL,ST2
9216 112711      LD   DE,1127H
9219 CD4092      CALL AFFICH

```

```

          FINAFFI: EQU  $
921C F1          POP  AF
921D C1          POP  BC
921E E1          POP  HL
921F C9          RET

```

```

          TEMPO:   EQU  $
9220 E5          PUSH HL
9221 C5          PUSH BC
9222 F5          PUSH AF
9223 212C01      LD   HL,300

```

```

          BIST:    EQU  $
9226 0600      LD   B,0
9228 10FE      BT:   DJNZ BT
922A 2B        DEC  HL
922B 7C        LD   A,H
922C B5        OR   L

```

```

304 922D 20F7          JR    NZ,BIST
305 922F F1           POP    AF
306 9230 C1           POP    BC
307 9231 E1           POP    HL
308 9232 C9           RET
309                   ;
310                   ;-----
311                   ; Commentaire de fin de partie
312                   ;-----
313 COMMENT: EQU $
314 9233 CD0390        CALL CLS
315 9236 214D94        LD     HL,MES2
316 9239 110101        LD     DE,101H
317 923C CD4092        CALL AFFICH
318 923F C9           RET
319                   ;
320                   ;-----
321                   ;Affichage d'un texte a l'ecran
322                   ;-----
323                   ;Entree: HL=pointeur memoire
324                   ;      DE=ligne/colonne affich
325                   ;-----
326 AFFICH: EQU $
327 9240 E5           PUSH HL
328 9241 F5           PUSH AF
329 9242 7A           LD     A,D
330 9243 CD72BB        CALL SETROW          ;Init ligne
331 9246 7B           LD     A,E

```

```

332 9247 CD6FBB      CALL SETCOL           ;Initi colonne
333 924A F1          POP AF
334 924B E1          POP HL
335 924C E5          PUSH HL
336 924D F5          PUSH AF
337                AT1:      EQU $           ;Boucle d'affichage
338 924E 7E          LD A,(HL)
339 924F FEFF        CP 0FFH               ;Delimiteur ?
340 9251 2B06        JR Z,AT2              ;Oui => Fin d'affichage
341 9253 CD5ABB      CALL PRINT            ;Affichage caractere
342 9256 23          INC HL
343 9257 1BF5        JR AT1
344                AT2:      EQU $
345 9259 F1          POP AF
346 925A E1          POP HL
347 925B C9          RET
348                ;
349                ;
350                ;=====
351                ; ZONE DES EQU
352                ;=====
353                ;
354                CR:      EQU 13           ;Code CR
355                LF:      EQU 10           ;Code LF
356                CRLF:    EQU 13           ;Code CR
357                BS:      EQU 127          ;Code BS
358                WAIT:    EQU 0BB06H       ;KM WAIT CHAR
359                PRINT:    EQU 0BB5AH      ;TXT OUTPUT
360                TXTOUT:   EQU 0BB5DH      ;TXT WR CHAR

```

```

361      SETCOL:      EQU 0BB6FH          ;TXT SET COLUMN
362      SETROW:      EQU 0BB72H          ;TXT SET ROW
363      SETMODE:     EQU 0BC0EH          ;TXT SET MODE
364      ;
365      ;=====
366      ; ZONE DES DS
367      ;=====
368      ;
369      PX:           DS 1                ;Position en X
370      PY:           DS 1                ;Position en Y
371      PDX:          DS 1                ;Pos depart en X
372      BUF:          DS 30              ;Buffer de lecture
373      CAR:          DS 1                ;Stockage temporaire
374      N:            DS 1                ; Nombre de tirages
375      SA:           DS 1                ;Sauvegarde de A
376      MEMCAR:       DS 1                ;Memo car tape
377      CAA:          DS 1                ;Carre a afficher
378      T:            DS 30              ;Tableau de jeu
379      TU:           DS 30              ;Tableau utilisateur
380
381
382      REGLE:        EQU $
383 92BE 53494D4F      DB "SIMON",CR,LF,"---
383 92C2 4E0D0A2D
383 92C6 2D2D2D2D
383 92CA 0D0A0A
384 92CD 4C276F72      DB "L'ordinateur va a
384 92D1 64696E61
384 92D5 74657572
384 92D9 20766120

```

384 92DD 61666669
384 92E1 63686572
384 92E5 20756E65
385 92E9 20737569 DB " suite de symbole
385 92ED 74652064
385 92F1 65207379
385 92F5 6D626F6C
385 92F9 6573206C
385 92FD 756D696E
385 9301 6575782E
386 9305 0D0A566F DB CR,LF,"Vous devrez
386 9309 75732064
386 930D 65767265
386 9311 7A207265
386 9315 73746974
386 9319 75657220
386 931D 6C
387 931E 65732073 DB "es symboles dans
387 9322 796D626F
387 9326 6C657320
387 932A 64616E73
387 932E 206C6520
387 9332 626F6E20
387 9336 6F726472
387 933A 65
388 933B 20656E0D DB " en",CR,LF,"utili
388 933F 0A757469
388 9343 6C697361
388 9347 6E74206C
388 934B 65732074

388 934F 6F75	
389 9351 63686573	DB "ches fleches du c
389 9355 20666C65	
389 9359 63686573	
389 935D 20647520	
389 9361 636C6176	
389 9365 6965722E	
389 9369 0D	
390 936A 0A0A546F	DB LF,LF,"Toute bonne
390 936E 75746520	
390 9372 626F6E6E	
390 9376 65207265	
390 937A 706F6E73	
390 937E 65206175	
390 9382 67	
391 9383 6D656E74	DB "mente d'un le nom
391 9387 65206427	
391 938B 756E206C	
391 938F 65206E6F	
391 9393 6D627265	
391 9397 20646520	
391 939B 73796D62	
391 939F 6F	
392 93A0 6C65730D	DB "les",CR,LF,"affic
392 93A4 0A616666	
392 93A8 69636865	
392 93AC 73206175	
392 93B0 2070726F	
392 93B4 6368	
393 93B6 61696E20	DB "ain coup. Toute m


```

393 93BA 636F7570
393 93BE 2E20546F
393 93C2 75746520
393 93C6 6D617576
393 93CA 61697365
393 93CE 20726570
393 93D2 6F
394 93D3 6E736520      DB  "nse annule",CR,LF
394 93D7 616E6E75
394 93DB 6C650D0A
394 93DF 6C612070
394 93E3 61727469
394 93E7 652E
395 93E9 0D0A0A41      DB  CR,LF,LF,"Appuyez
395 93ED 70707579
395 93F1 657A2073
395 93F5 75722075
395 93F9 6E652074
395 93FD 6F7563
396 9400 68652070      DB  "he pour commencer
396 9404 6F757220
396 9408 636F6D6D
396 940C 656E6365
396 9410 72206120
396 9414 6A6F7565
396 9418 722E2E2E
397 941C 0D0AFF      DB  CR,LF,0FFH
398
399                      MES1:    EQU  *
400 941F 52657461      DB  "Retapez : a sequen

```

```

400 9423 70657620
400 9427 6C612073
400 942B 65717565
400 942F 6E636520
400 9433 61707061
400 9437 72756520
400 943B 73
401 943C 7572206C      DB  "ur l'ecran ...",C
401 9440 27656372
401 9444 616E202E
401 9448 2E2E0D0A
401 944C FF
402
403      MES2:      EQU  $
404 944D 4C612073      DB  "La sequence entre
404 9451 65717565
404 9455 6E636520
404 9459 656E7472
404 945D 6565206E
404 9461 6520636F
404 9465 72726573
404 9469 70
405 946A 6F6E6420      DB  "ond pas a celle a
405 946E 70617320
405 9472 61206365
405 9476 6C6C6520
405 947A 61666669
405 947E 63686565
405 9482 2E2E2E

```

```

406 9485 FF          DB    0FFH
407
408                ST1:    EQU  $
409 9486 7F7F7FFF      DB    127,127,127,0FFH
410
411                ST2:    EQU  $
412 948A 202020FF      DB    32,32,32,0FFH
413                ;
414                ; =====
415                ; PROGRAMME PRINCIPAL
416                ; =====
417                SIMON:   EQU  $
418 948E CD0390          CALL CLS                ;Effacement d'ecran
419 9491 21BE92          LD    HL,REGLE
420 9494 110101          LD    DE,101H
421 9497 CD4092          CALL AFFICH            ;Regle du jeu
422 949A CD06BB          CALL WAIT             ;Attente d'un caractere
423 949D AF             XOR    A
424 949E 327E92          LD    (N),A           ;Nombre de tirages
425                ;.....
426                ; Deroulement d'une partie
427                ;.....
428 94A1 CD0990          CALL JEU
429                ;.....
430                ; Commentaire de fin de partie
431                ;.....
432 94A4 CD3392          CALL COMMENT
433 94A7 C9             RET
434                END

```

Remarquez la similitude de sa structure avec le programme Turbo-Pascal correspondant.

La possibilité de le comparer à un programme Basic et Turbo-Pascal strictement équivalent devrait vous faire progresser dans l'apprentissage de ce langage.

Si vous désirez utiliser un chargeur Basic plutôt qu'un long programme source Assembleur, voici les codes à entrer :

```

1000 REM =====
1010 REM JEU DU SIMON
1020 REM =====
1030
1040 MEMORY &4000
1050 FOR I=&9000 TO &94A7
1060   READ A$
1070   A$="&" + A$
1080   POKE I,VAL(A$)
1090 NEXT I
1100 CALL &9000
1110 END
1120
1130 -----
1140 DATA C3,8E,94,3E,2,CD,E,BC,C9,CD,3,90,3A,7E,92,3C
1150 DATA 32,7E,92,ED,5F,E6,F,FE,4,38,C,FE,8,38,C,FE
1160 DATA C,38,C,3E,4,18,A,3E,1,18,6,3E,2,18,2,3E
1170 DATA 3,32,7F,92,21,82,92,16,0,3A,7E,92,5F,19,3A,7F
1180 DATA 92,77,3E,28,CD,6F,BB,3E,C,CD,72,BB,3E,F4,CD,5A
1190 DATA BB,3E,27,CD,6F,BB,3E,D,CD,72,BB,3E,F7,CD,5A,BB
1200 DATA 3E,20,CD,5A,BB,3E,F6,CD,5A,BB,3E,28,CD,6F,BB,3E
1210 DATA E,CD,72,BB,3E,F5,CD,5A,BB,21,82,92,3A,7E,92,47
1220 DATA 3E,1,E5,F5,16,0,5F,19,7E,32,81,92,F1,E1,CD,1E
1230 DATA 91,CD,20,92,BB,28,3,3C,18,E8,21,1F,94,11,1,14
1240 DATA CD,40,92,3A,7E,92,47,3E,1,F5,3E,2A,CD,5A,BB,F1
1250 DATA BB,28,3,3C,18,F3,21,A0,92,3A,7E,92,47,3E,1,32
1260 DATA 7F,92,CD,6,BB,FE,F0,28,E,FE,F1,28,E,FE,F2,28
1270 DATA E,FE,F3,28,E,18,EB,3E,1,18,A,3E,3,18,6,3E
1280 DATA 4,18,2,3E,2,32,80,92,32,81,92,CD,1E,91,CD,20
1290 DATA 92,3A,7F,92,16,0,5F,E5,19,3A,80,92,77,E1,3A,7F
1300 DATA 92,BB,20,16,21,82,92,11,A0,92,23,13,3A,7E,92,47
1310 DATA 1A,BE,C0,13,23,10,F9,C3,9,90,3C,C3,BF,90,E5,C5
1320 DATA F5,3A,81,92,FE,1,28,45,FE,2,28,7C,FE,3,CA,E3
1330 DATA 91,21,86,94,11,22,C,CD,40,92,21,86,94,11,22,D
1340 DATA CD,40,92,21,86,94,11,22,E,CD,40,92,CD,20,92,21
1350 DATA 8A,94,11,22,C,CD,40,92,21,8A,94,11,22,D,CD,40
1360 DATA 92,21,8A,94,11,22,E,CD,40,92,C3,1C,92,21,86,94
1370 DATA 11,27,9,CD,40,92,21,86,94,11,27,A,CD,40,92,21
1380 DATA 86,94,11,27,B,CD,40,92,CD,20,92,21,8A,94,11,27
1390 DATA 9,CD,40,92,21,8A,94,11,27,A,CD,40,92,21,8A,94
1400 DATA 11,27,B,CD,40,92,18,74,21,86,94,11,2A,C,CD,40
1410 DATA 92,21,86,94,11,2A,D,CD,40,92,21,86,94,11,2A,E
1420 DATA CD,40,92,CD,20,92,21,8A,94,11,2A,C,CD,40,92,21
1430 DATA 8A,94,11,2A,D,CD,40,92,21,8A,94,11,2A,E,CD,40
1440 DATA 92,18,39,21,86,94,11,27,F,CD,40,92,21,86,94,11
1450 DATA 27,10,CD,40,92,21,86,94,11,27,11,CD,40,92,CD,20
1460 DATA 92,21,8A,94,11,27,F,CD,40,92,21,8A,94,11,27,10
1470 DATA CD,40,92,21,8A,94,11,27,11,CD,40,92,F1,C1,E1,C9
1480 DATA E5,C5,F5,21,2C,1,6,0,10,FE,28,7C,B5,20,F7,F1

```

Partie 9 : Programmes

```

1490 DATA C1,E1,C9,CD,3,90,21,4D,94,11,1,1,CD,40,92,C9
1500 DATA ES,F5,7A,CD,72,BB,7B,CD,6F,BB,F1,E1,ES,F5,FE
1510 DATA FF,28,6,CD,5A,BB,23,18,F5,F1,E1,C9,68,65,D,A
1520 DATA 6C,65,20,6E,6F,6D,62,72,65,20,64,65,20,6C,65,74
1530 DATA 74,72,65,73,20,64,65,73,20,6D,6F,74,73,20,71,75
1540 DATA 69,20,6C,61,20,63,6F,6D,70,6F,73,65,6E,74,2E,D
1550 DATA A,56,6F,75,73,20,64,65,76,65,7A,20,70,72,6F,70
1560 DATA 6F,73,65,72,20,64,65,73,20,6C,65,74,74,72,65,73
1570 DATA 20,64,65,20,6C,27,61,6C,70,68,61,62,65,74,53,49
1580 DATA 4D,4F,4E,D,A,2D,2D,2D,2D,2D,D,A,A,4C,27,6F
1590 DATA 72,64,69,6E,61,74,65,75,72,20,76,61,20,61,66,66
1600 DATA 69,63,68,65,72,20,75,6E,65,20,73,75,69,74,65,20
1610 DATA 64,65,20,73,79,62,62,6F,6C,65,73,20,6C,75,6D,69
1620 DATA 6E,65,75,78,2E,D,A,56,6F,75,73,20,64,65,76,72
1630 DATA 65,7A,20,72,65,73,74,69,74,75,65,72,20,6C,65,73
1640 DATA 20,73,79,6D,62,6F,6C,65,73,20,64,61,6E,73,20,6C
1650 DATA 65,20,62,6F,6E,20,6F,72,64,72,65,20,65,6E,D,A
1660 DATA 75,74,69,6C,69,73,61,6E,74,20,6C,65,73,20,74,6F
1670 DATA 75,63,68,65,73,20,66,6C,65,63,68,65,73,20,64,75
1680 DATA 20,63,6C,61,76,69,65,72,2E,D,A,A,54,6F,75,74
1690 DATA 65,20,62,6F,6C,6E,65,20,72,65,70,6F,6E,73,65,20
1700 DATA 61,75,67,6D,65,6E,74,65,20,64,27,75,6E,20,6C,65
1710 DATA 20,6E,6F,6D,62,72,65,20,64,65,20,73,79,6D,62,6F
1720 DATA 6C,65,73,D,A,61,66,66,69,63,68,65,73,20,61,75
1730 DATA 20,70,72,6F,63,68,61,69,6E,20,63,6F,75,70,2E,20
1740 DATA 54,6F,75,74,65,20,6D,61,75,76,61,69,73,65,20,72
1750 DATA 65,70,6F,6E,73,65,20,61,6E,6E,75,6C,65,D,A,6C
1760 DATA 61,20,70,61,72,74,69,65,2E,D,A,A,41,70,70,75
1770 DATA 79,65,7A,20,73,75,72,20,75,6E,65,20,74,6F,75,63
1780 DATA 68,65,20,70,6F,75,72,20,63,6F,6D,6D,65,6E,63,65
1790 DATA 72,20,61,20,6A,6F,75,65,72,2E,2E,2E,D,A,FF,52
1800 DATA 65,74,61,70,65,7A,20,6C,61,20,73,65,71,75,65,6E
1810 DATA 63,65,20,61,70,70,61,72,75,65,20,73,75,72,20,6C
1820 DATA 27,65,63,72,61,6E,20,2E,2E,2E,D,A,FF,4C,61,20
1830 DATA 73,65,71,75,65,6E,63,65,20,65,6E,74,72,65,65,20
1840 DATA 6E,65,20,63,6F,72,72,65,73,70,6F,6E,64,20,70,61
1850 DATA 73,20,61,23,63,65,6C,6C,65,20,61,66,66,69,63,68
1860 DATA 65,65,2E,2E,2E,FF,7F,7F,7F,FF,20,20,20,FF,CD,3
1870 DATA 90,21,BE,92,11,1,1,CD,40,92,CD,6,BB,AF,32,7E
1880 DATA 92,CD,9,90,CD,33,92,C9,0,0,0,0,0,0,0,0

```

et les codes de checksum correspondant :

```

72 18 AA 11 B 7B FB EA 2E 2E A6 84 9 3A 55 B3 C4 33 B 2A 60 8D 63 22 F7 D 2 3D 6
A 9F 55 EB 43 2A 6C 4F F3 C5 C7 9 BE DB 3E 7E E7 18
E2 34 88 50 E5 F 4A 11 6 DB DA DB BF 9E 24 B3 EF 1B 20 2F 2D E1 C1 22 29 9F 5 A6
57

```

Pour installer et exécuter le programme Assembleur, il suffit d'exécuter le programme Basic. Les codes hexadécimaux sont POKÉS en mémoire à partir de l'adresse &9000.

Lorsque l'installation a été faite par le programme Basic, le programme Assembleur peut à nouveau être exécuté par une instruction **CALL &9000**.

9/4

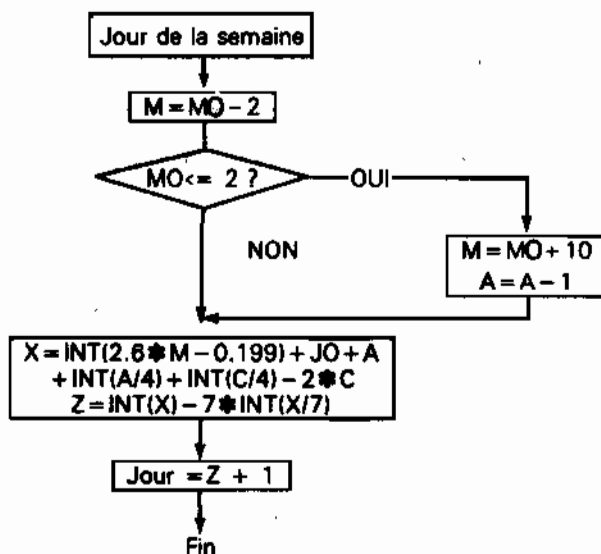
Mathématiques

9/4.1

Nom d'un jour de la semaine

Plusieurs formules permettent de connaître le nom d'un jour de la semaine. Nous utilisons ici celle de GAUSS.

Si JO est le jour, MO le mois, C les deux premiers chiffres de l'année et A les deux derniers chiffres de l'année, le numéro du jour est obtenu par :



Le programme écrit en BASIC est le suivant :

```

1000 REM =====
1010 REM                      Nom d'un jour de la semaine
1020 REM =====
1030 GOSUB 2000 'Initialisation
1040 GOSUB 3000 'Calcul
1050 END

2000 REM =====
2010 REM                      Initialisation
2020 REM =====
2030 FOR I=1 TO 7
2040   READ J$(I)
2050 NEXT I
2060 DATA Dimanche,Lundi,Mardi,Mercredi,Jedi,Vendredi,Samedi
2070 RETURN

3000 REM =====
3010 REM                      Saisie du jour et calcul
3020 REM =====
3030 MODE 1:PRINT" CALCUL DU NOM D'UN JOUR DE LA SEMAINE"
3040 LOCATE 1,5
3050 INPUT "Jour (Entre 1 et 31)   " : J$
3060 INPUT "Mois (Entre 1 et 12)    " : M$
3070 INPUT "Annee (sur 4 chiffres) " : AN$
3080 M=M$-2:IF M<=2 THEN M=M$+10
3090 C=VAL(LEFT$(AN$,2)):A=VAL(RIGHT$(AN$,2))
3100 IF M<=2 THEN A=A-1
3110 X=INT(2.6*M-0.199)+J$+A+INT(A/4)+INT(C/4)-2*C
3120 Z=INT(X)-7*INT(X/7)
3130 PRINT:PRINT"Le";J$;"/";M$;"/ "AN$;" est un "J$(Z+1)
3140 RETURN

```

Lignes 1000 à 1050 : Programme principal.

Lignes 2000 à 2070 : Initialisation.

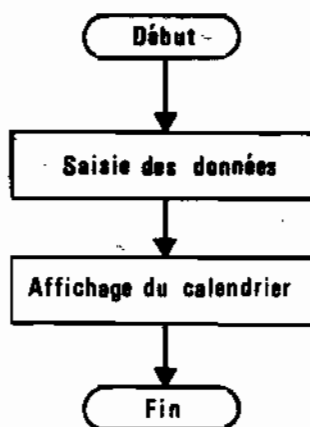
Lignes 3000 à 3140 : Saisie du jour et calculs.

9/4.2

Calendrier perpétuel

La formule de Gauss dont nous avons parlé en 9/4.1 est reprise dans ce programme pour nous permettre de sortir sur imprimante le calendrier complet d'une année quelconque.

L'algorithme qui illustre la structure du programme est très simple :



```

1000 CLS:PRINT"Trace de calendrier"
1010 '=====
1020 'Nom des mois et nombre de jours pour chaque mois
1030 '=====
1040 DIM MO$(12),MX(12)
1050 FOR I=1 TO 12
1060   READ MO$(I)
1070 NEXT I
1080 '
1090 FOR I=1 TO 12
1100   READ MX(I)
1110 NEXT I
1120 '
1130 DATA Janvier,Fevrier,Mars,Avril,Mai,Juin,Juillet,Aout,Septembre,Octobre,Nov
embre,Decembre
1140 DATA 31,28,31,30,31,30,31,31,30,31,30,31
1150 '=====
1160 'Trace du calendrier
1170 '=====
1180 INPUT "Année du calendrier";AN$
1190 PRINT"Connectez l'imprimante puis appuyez sur une touche quelconque"
1200 AS=INKEY$:IF AS="" THEN 1200
1210 '
  
```

```

1220 MOIS=1
1230 PRINT#8,SPACE$(32);"CALENDRIER ";AN$
1240 PRINT#8:PRINT#8:PRINT#8
1250
1260 FOR N=1 TO 6
1270   GOSUB 1360 'Calcul des jours J1 et J2
1280   FOR L=1 TO 40
1290     ON L GOSUB 1610,1790,1850,1910,2060,2060,2060,2060,2060,1850
1300     PRINT#8,L;I$
1310   NEXT L
1320   PRINT#8:PRINT#8
1330   MOIS=MOIS+2
1340 NEXT N
1350 END
1360 '=====
1370 'Calcul des premiers jours J(1) et J(2) des mois a afficher
1380 '=====
1390 AN=VAL(AN$):SEC=0
1400 R$=RIGHT$(AN$,1):L$=LEFT$(AN$,2)
1410 IF VAL(R$)=0 THEN S=1 ELSE S=0
1420 BX=0
1430 IF S=0 AND INT(AN/4)=AN/4 THEN BX=1
1440 IF S=1 AND INT(VAL(L$)/4)=VAL(L$)/4 THEN BX=1
1450 IF BX=1 THEN MX(2)=29 ELSE MX(2)=28
1460
1470 JOUR=1
1480 FOR I=1 TO 2
1490   M=MOIS-2:IF MOIS<=2 THEN M=MOIS+10
1500   C=VAL(LEFT$(AN$,2)):A=VAL(RIGHT$(AN$,2))
1510   IF MOIS<=2 THEN A=A-1
1520   X=INT(2.6*M-0.199)+JOUR+A+INT(A/4)+INT(C/4)-2*C
1530   J(I)=INT(X)-7*INT(X/7)+1
1540   MOIS=MOIS+1
1550 NEXT I
1560 MOIS=MOIS-2
1570 RETURN
1580 '=====
1590 'Calcul des lignes a afficher
1600 '=====
1610
1620 'Affichage de la premiere ligne
1630 '-----
1640 L1$=""
1650 FOR K=1 TO 2
1660   LL=LEN(MO$(MOIS))+5
1670   L1=INT((24-LL)/2)
1680   FOR I=1 TO L1
1690     L1$=L1$+" "
1700   NEXT I
1710   L1$=L1$+MO$(MOIS)+" "+AN$
1720   FOR I=1 TO 24-L1-LL
1730     L1$=L1$+" "
1740   NEXT I
1750   IF K=1 THEN L1$=L1$+" " MOIS=MOIS+1
1760 NEXT K
1770 MOIS=MOIS-1
1780 RETURN
1790 '-----
1800 'Affichage de la seconde ligne
1810 '-----
1820 IN$=" : D L M M J V S : "
1830 L1$="" "+IN$+" "+IN$
1840 RETURN

```

```

1850 '-----
1860 'Affichage de la troisieme ligne
1870 '-----
1880 IN$="-----"
1890 LI$="      "+IN$+"      "+IN$
1900 RETURN
1910 '-----
1920 'Affichage de la quatrieme ligne
1930 '-----
1940 SE$="      ";LI$="      ";
1950 FOR I=1 TO 7
1960   IF I<J(1) THEN LI$=LI$+SE$ ELSE LI$=LI$+STR$(I-J(1)+1)+" "
1970 NEXT I
1980 JC(1)=I-J(1)+1;LI$=LI$+" ";
1990 FOR I=1 TO 7
2000   IF I<J(2) THEN LI$=LI$+SE$ ELSE LI$=LI$+STR$(I-J(2)+1)+" "
2010 NEXT I
2020 LI$=LI$+" ";
2030 JC(2)=I-J(2)+1
2040 RETURN
2050 '-----
2060 'Affichage d'une ligne superieure a 4
2070 '-----
2080 LI$="      ";
2090 FOR I=JC(1) TO JC(1)+6
2100   IF I>MX(MOIS) THEN 2130
2110   LI$=LI$+STR$(I)
2120   IF I<=9 THEN LI$=LI$+" "
2130 NEXT I
2140 LL=LEN(LI$);IF LL<28 THEN FOR K=LL TO 27:LI$=LI$+" ":NEXT K
2150 JC(1)=I;LI$=LI$+" ";
2160 FOR I=JC(2) TO JC(2)+6
2170   IF I>MX(MOIS+1) THEN 2200
2180   LI$=LI$+STR$(I)
2190   IF I<=9 THEN LI$=LI$+" "
2200 NEXT I
2210 LL=LEN(LI$);IF LL<70 THEN FOR K=LL TO 69:LI$=LI$+" ":NEXT K
2220 JC(2)=I;LI$=LI$+" ";
2230 RETURN

```

Lignes 1000 à 1140 : Initialisation, saisie des données

Lignes 1150 à 1350 : Tracé du calendrier

Lignes 1360 à 1570 : Calcul des premiers jours des deux mois à afficher sur une même ligne

Lignes 1580 à 2230 : Sous-programmes de calcul des données à afficher

Partie 9 : Programmes

-----Janvier 1987-----						
: D	L	M	M	J	V	S :
:	:	:	:	1	2	3 :
:	4	5	6	7	8	9 10 :
:	11	12	13	14	15	16 17 :
:	18	19	20	21	22	23 24 :
:	25	26	27	28	29	30 31 :
:	:	:	:	:	:	:

-----Février 1987-----						
: D	L	M	M	J	V	S :
:	1	2	3	4	5	6 7 :
:	8	9	10	11	12	13 14 :
:	15	16	17	18	19	20 21 :
:	22	23	24	25	26	27 28 :
:	:	:	:	:	:	:
:	:	:	:	:	:	:

-----Mars 1987-----						
: D	L	M	M	J	V	S :
:	1	2	3	4	5	6 7 :
:	8	9	10	11	12	13 14 :
:	15	16	17	18	19	20 21 :
:	22	23	24	25	26	27 28 :
:	29	30	31	:	:	:
:	:	:	:	:	:	:

-----Avril 1987-----						
: D	L	M	M	J	V	S :
:	:	:	:	1	2	3 4 :
:	5	6	7	8	9	10 11 :
:	12	13	14	15	16	17 18 :
:	19	20	21	22	23	24 25 :
:	26	27	28	29	30	:
:	:	:	:	:	:	:

-----Mai 1987-----						
: D	L	M	M	J	V	S :
:	:	:	:	:	1	2 :
:	3	4	5	6	7	8 9 :
:	10	11	12	13	14	15 16 :
:	17	18	19	20	21	22 23 :
:	24	25	26	27	28	29 30 :
:	31	:	:	:	:	:

-----Juin 1987-----						
: D	L	M	M	J	V	S :
:	:	:	:	1	2	3 4 :
:	5	6	7	8	9	10 11 :
:	12	13	14	15	16	17 18 :
:	19	20	21	22	23	24 25 :
:	26	27	28	29	30	:
:	:	:	:	:	:	:

-----Juillet 1987-----						
: D	L	M	M	J	V	S :
:	:	:	:	1	2	3 4 :
:	5	6	7	8	9	10 11 :
:	12	13	14	15	16	17 18 :
:	19	20	21	22	23	24 25 :
:	26	27	28	29	30	31 :
:	:	:	:	:	:	:

-----Août 1987-----						
: D	L	M	M	J	V	S :
:	:	:	:	:	:	1 :
:	2	3	4	5	6	7 8 :
:	9	10	11	12	13	14 15 :
:	16	17	18	19	20	21 22 :
:	23	24	25	26	27	28 29 :
:	30	31	:	:	:	:

-----Septembre 1987-----						
: D	L	M	M	J	V	S :
:	:	:	:	1	2	3 4 5 :
:	6	7	8	9	10	11 12 :
:	13	14	15	16	17	18 19 :
:	20	21	22	23	24	25 26 :
:	27	28	29	30	:	:
:	:	:	:	:	:	:

-----Octobre 1987-----						
: D	L	M	M	J	V	S :
:	:	:	:	:	1	2 3 :
:	4	5	6	7	8	9 10 :
:	11	12	13	14	15	16 17 :
:	18	19	20	21	22	23 24 :
:	25	26	27	28	29	30 31 :
:	:	:	:	:	:	:

-----Novembre 1987-----						
: D	L	M	M	J	V	S :
:	1	2	3	4	5	6 7 :
:	8	9	10	11	12	13 14 :
:	15	16	17	18	19	20 21 :
:	22	23	24	25	26	27 28 :
:	29	30	:	:	:	:
:	:	:	:	:	:	:

-----Decembre 1987-----						
: D	L	M	M	J	V	S :
:	:	:	:	1	2	3 4 5 :
:	6	7	8	9	10	11 12 :
:	13	14	15	16	17	18 19 :
:	20	21	22	23	24	25 26 :
:	27	28	29	30	31	:
:	:	:	:	:	:	:

9/4.3

Biorythmes

Des études statistiques ont permis de mettre en évidence trois cycles de vie spécifiques à chaque individu. Il s'agit des cycles :

- physique (23 jours),
- émotionnel (28 jours),
- intellectuel (31 jours).

Ces cycles sinusoïdaux ont une origine commune le jour de la naissance de l'intéressé. Ils indiquent une période positive si, pour un jour donné, l'ordonnée est positive, une période négative si l'ordonnée est négative et une période critique si l'ordonnée est nulle (barricadez-vous chez vous car tout peut arriver ces jours-là !).

```

1000 MODE 1:PRINT"Biorythmes"
1010 '=====
1020 'Initialisation
1030 '=====
1040 DIM T(12)
1050 FOR I=1 TO 12
1060   READ T(I)
1070 NEXT I
1080 DATA 31,28,31,30,31,30,31,31,30,31,30,31
1090 '=====
1100 'Entree des donnees
1110 '=====
1120 INPUT"Jour de naissance ";JN
1130 INPUT"Mois de naissance ";MN
1140 INPUT"Annee de naissance ";AN
1150 PRINT:INPUT"Mois du trace ";MB
1160 INPUT"Annee du trace ";AB
1170 '=====
1180 'Trace des biorythmes
1190 '=====
1200 '
1210 '-----
1220 'Nombre de jours ecoules depuis la naissance
1230 '-----
1240 IF (AN/4)=INT(AN/4) THEN IN=1 ELSE IN=0
1250 IF IN=1 AND MN=2 THEN FIN=29 ELSE FIN=T(MN)
1260 FOR I=JN TO FIN
1270   TT=TT+1
1280 NEXT I
1290 FOR I=MN+1 TO 12
1300   IF IN=1 AND I=2 THEN TT=TT+29 ELSE TT=TT+T(I)
1310 NEXT I
1320 '

```

```

1330 FOR I=2 TO AB-AN
1340   B=AN+I-1
1350   IF B/4=INT(B/4) THEN TT=TT+1
1360   TT=TT+365
1370 NEXT I
1380 '
1390 IF AB/4=INT(AB/4) THEN BI=1 ELSE BI=0
1400 FOR I=1 TO MB-1
1410   IF BI=1 AND I=2 THEN TT=TT+29 ELSE TT=TT+T(I)
1420 NEXT I
1430 '-----
1440 'Affichage
1450 '-----
1460 CLS
1470 FOR I=1 TO T(MB)
1480   A=SIN(2*PI*(TT+I)/23)
1490   LOCATE I+3,11-INT(9*A):PRINT"*"
1500   C=SIN(2*PI*(TT+I)/33)
1510   LOCATE I+3,11-INT(9*C):PRINT"+"
1520   B=SIN(2*PI*(TT+I)/28)
1530   LOCATE I+3,11-INT(9*B):PRINT"."
1540 NEXT I
1550 FOR I=1 TO 7
1560   READ X,Y,D
1570   LOCATE X,Y:PRINT D
1580 NEXT I
1590 DATA 3,21,1,7,21,5,12,21,10,17,21,15,22,21,20,27,21,25,32,21,30
1600 '
1610 'Axes
1620 '
1630 LOCATE 4,11:PRINT"-----"
1640 FOR I=1 TO 20
1650   LOCATE 3,I+1:PRINT"Δ"
1660 NEXT I
1670 '=====
1680 'Legende
1690 '=====
1700 LOCATE 5,22:PRINT"* = Physique  . = Affectif"
1710 LOCATE 5,23:PRINT"+ = Intellectuel"

```

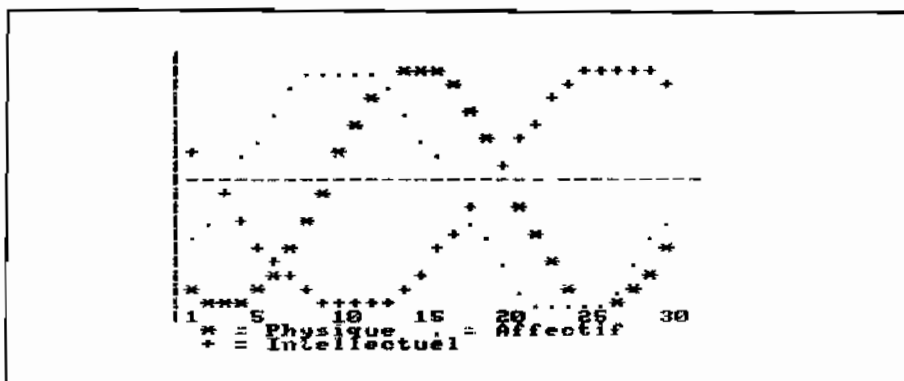
Lignes 1000 à 1080 : Initialisation

Lignes 1090 à 1160 : Saisie des données pour tracer le biorythme

Lignes 1210 à 1420 : Calculs pour initialiser le biorythme courant

Lignes 1430 à 1710 : Affichage du biorythme

Exemple de biorythme issu du présent programme.



9/5

Gestion de fichiers

Nous allons réaliser un programme de gestion de fichiers classique dont les possibilités seront :

- création de structure,
- création de fiche,
- visualisation de l'ensemble des fiches,
- impression de l'ensemble des fiches,
- modification d'une fiche,
- suppression d'une fiche,
- lecture d'un fichier sur disquette,
- sauvegarde d'un fichier sur disquette.

Ce programme fonctionne sur les CPC 664 et 6128, et avec de légères modifications sur CPC 464 qui sont :

1110 PRINT"7) Lire un fichier sur cassette"

1120 PRINT"8) Sauvegarder un fichier sur cassette"

5010 REM Lecture d'un fichier sur cassette

5030 CLS:PRINT"LECTURE D'UN FICHIER SUR CASSETTE":PRINT

5050 OPENIN "I" + N\$

5510 REM Sauvegarde d'un fichier sur cassette

5530 CLS:PRINT"SAUVEGARDE D'UN FICHIER SUR CASSETTE":PRINT

5550 OPENOUT "I" + N\$

La création de structure consiste à définir le nombre d'articles par fiche et le libellé de chaque article.

Une fiche créée sera identifiée par la suite par son numéro de création.

L'ensemble des fiches peut être visualisé sur l'écran, fiche par fiche.

Appuyez sur une touche quelconque pour passer à la fiche suivante.

Pour la sortie sur imprimante, les fiches sont listées les unes derrière les autres, et un message de fin d'impression apparaît à l'écran avant le retour au menu général quand l'impression est finie.

Une fiche peut être modifiée. Son ancien contenu apparaît sur l'écran après avoir donné son numéro d'identification, et le curseur se positionne automatiquement au début de chaque article pour effectuer la modification. Si vous tapez <cr> sur un article, il est annulé.

Une fiche identifiée par son numéro de création peut être supprimée.

Les fichiers lus ou sauvegardés sur disquette ont la structure suivante :

AF Nombre de libellés

Libellé 1		Libellé n
-----------	-------	-----------

NF Nombre de fiches

Fiche 1		Fiche n
---------	-------	---------

Le programme écrit en BASIC est le suivant :

```

1000 REM =====
1010 REM          Programme principal
1020 REM =====
1030 CLS:PRINT"GESTION DE FICHIERS"
1040 PRINT:PRINT"Voulez-vous : "
1050 PRINT"1) Créer une structure de fichier"
1060 PRINT"2) Créer une fiche"
1070 PRINT"3) Visualiser l'ensemble des fiches"
1080 PRINT"4) Imprimer l'ensemble des fiches"
1090 PRINT"5) Modifier une fiche"
1100 PRINT"6) Supprimer une fiche"
1110 PRINT"7) Lire un fichier sur disquette"
1120 PRINT"8) sauvegarder un fichier sur disquette"
1130 PRINT"9) Sortir du programme"
1140 PRINT:INPUT "Votre choix " : C
1150 IF C<1 OR C>9 THEN SOUND 1,100,100:GOTO 1000

```



```
1160 ON C GOSUB 2030,2530,3030,3530,4030,4530,5030,5530,6030
1170 IF FIN=0 THEN GOTO 1000 'Boucle sur le menu
1180 END

2000 REM =====
2010 REM          Creation de structure
2020 REM =====
2030 CLS:PRINT"CREATION DE STRUCTURE":PRINT
2040 INPUT"Donnez le nombre d'articles par fiche : ";AF
2050 PRINT:PRINT"Entrez le libelle de chaque article : "
2060 FOR I=1 TO AF
2070   PRINT"Libelle ";I;" : ";INPUT L$(I)
2080 NEXT I
2090 PRINT:PRINT"La structure est memorisee":FOR I=1 TO 500:NEXT
2100 RETURN

2500 REM =====
2510 REM          Creation d'une fiche
2520 REM =====
2530 CLS:PRINT "CREATION DE FICHE":PRINT
2540 INPUT "Numero de fiche";NF
2550 IF NF>NM THEN NM=NF
2560 FOR I=1 TO AF
2570   PRINT L$(I);:INPUT A$(NF,I)
2580 NEXT I
2590 PRINT:PRINT"Fiche en memoire"
2600 INPUT"Une autre saisie (O/N) : ";A$
2610 A$=UPPER$(A$)
2620 IF A$<>"O" AND A$<>"N" THEN SOUND 1,100,100:GOTO 2600
2630 IF A$="O" THEN 2530
2640 RETURN

3000 REM =====
3010 REM          Visualisation des fiches
3020 REM =====
```

```
3030 CLS:PRINT"VISUALISATION DES FICHES":PRINT
3040 FOR I=1 TO NM
3050   PRINT "Fiche No";I
3060   FOR J=1 TO AF
3070     PRINT L$(J);" : ";A$(I,J)
3080   NEXT J
3090   PRINT:PRINT"Appuyez sur une touche pour visualiser la fiche suivante"
3100   A$=INKEY$:IF A$="" THEN 3100
3110   CLS:PRINT:PRINT
3120 NEXT I
3130 PRINT" Toutes les fiches ont ete visualisees." :FOR I=1 TO 500:NEXT I
3140 RETURN

3500 REM =====
3510 REM           Impression des fiches
3520 REM =====
3530 CLS:PRINT"IMPRESSION DE L'ENSEMBLE DES FICHES":PRINT
3540 FOR I=1 TO NM
3550   PRINT #8,"Fiche No";I
3560   FOR J=1 TO AF
3570     PRINT #8,L$(J);" : ";A$(I,J)
3580   NEXT J
3590   PRINT #8
3600 NEXT I
3610 PRINT" Toutes les fiches ont ete imprimees." :FOR I=1 TO 500:NEXT I
3620 RETURN

4000 REM =====
4010 REM           Modification d'une fiche
4020 REM =====
4030 CLS:PRINT"MODIFICATION D'UNE FICHE":PRINT
4040 INPUT"Numero de la fiche a modifier ";N
4050 IF N>NM THEN PRINT"Cette fiche n'est pas en memoire":SOUND 1,100,100:FOR I=
1 TO 500:NEXT I:GOTO 4030
```

```
4060 FOR I=1 TO AF
4070   PRINT L$(I); " : ";A$(N,I)
4080 NEXT I
4090 LOCATE 1,4
4100 FOR I=1 TO AF
4110   PRINT L$(I); " : ";INPUT A$(N,I)
4120 NEXT I
4130 PRINT:PRINT"Fiche en memoire."
4140 PRINT:INPUT"Une autre modification (O/N) ";A$
4150 A$=UPPER$(A$)
4160 IF A$<>"O" AND A$<>"N" THEN SOUND 1,100,100:GOTO 4140
4170 IF A$="O" THEN 4030
4180 RETURN

4500 REM =====
4510 REM           Suppression d'une fiche
4520 REM =====
4530 CLS:PRINT"SUPPRESSION D'UNE FICHE":PRINT
4540 INPUT"Numero de fiche a supprimer";N
4550 IF N>NM THEN SOUND 1,100,100:PRINT"Cette fiche n'est pas en memoire":FOR I=
1 TO 500:NEXT I:GOTO 4500
4560 FOR I=1 TO AF
4570   PRINT L$(I); " : ";A$(N,I)
4580 NEXT I
4590 PRINT:INPUT "Etes-vous sur (O/N) ";A$
4600 A$=UPPER$(A$)
4610 IF A$<>"O" AND A$<>"N" THEN SOUND 1,100,100:GOTO 4530
4620 IF A$="N" THEN 4710
4630 FOR I=1 TO AF
4640   A$(N,I)=" "
4650 NEXT I
4660 PRINT:PRINT"Fiche supprimee"
4670 PRINT:INPUT"Une autre suppression (O/N) ";A$
4680 A$=UPPER$(A$)
```

```
4690 IF A$<>"0" AND A$<>"N" THEN SOUND 1,100,100:GOTO 4530
4700 IF A$="0" THEN 4500
4710 RETURN

5000 REM =====
5010 REM          Lecture d'un fichier sur disquette
5020 REM =====
5030 CLS:PRINT"LECTURE D'UN FICHIER SUR DISQUETTE":PRINT
5040 INPUT "Nom du fichier a lire ";N$
5050 OPENIN N$
5060 INPUT #9,AF "Nombre de libelles
5070 FOR I=1 TO AF
5080   INPUT #9,L$(I)
5090 NEXT I
5100 INPUT #9,NF "Nombre de fiches
5110 NM=NF "Nombre de fiches en memoire
5120 FOR I=1 TO NF
5130   FOR J=1 TO AF
5140     INPUT #9,A$(I,J)
5150   NEXT J
5160 NEXT I
5170 CLOSEIN
5180 PRINT:PRINT"Fichier en memoire":FOR I=1 TO 500:NEXT I
5190 RETURN

5500 REM =====
5510 REM          Sauvegarde d'un fichier sur disquette
5520 REM =====
5530 CLS:PRINT"SAUEGARDE D'UN FICHIER SUR DISQUETTE":PRINT
5540 INPUT "Nom de la sauvegarde ";N$
5550 OPENOUT N$
5560 PRINT #9,AF "Nombre de libelles
5570 FOR I=1 TO AF
5580   PRINT #9,L$(I) "Ecriture des libelles
```

```
5590 NEXT I
5600 PRINT #9,NF 'Nombre de fiches
5610 FOR I=1 TO NF
5620   FOR J=1 TO AF
5630     PRINT #9,A$(I,J) 'Ecriture des fiches
5640   NEXT J
5650 NEXT I
5660 CLOSEOUT
5670 PRINT:PRINT"Fichier sauvegarde sous le nom "I$;FOR I=1 TO 500:NEXT I
5680 RETURN
6000 REM =====
6010 REM           Sortie du programme
6020 REM =====
6030 CLS:PRINT"SORTIE DU PROGRAMME":PRINT
6040 INPUT"Etes-vous sur (O/N) : "I$
6050 A$=UPPER$(A$)
6060 IF A$<>"O" AND A$<>"N" THEN SOUND 1,100,100:GOTO 6040
6070 IF A$="O" THEN FIN=1
6080 RETURN
```

Lignes 1000 à 1180 : Programme principal : Affichage du menu et orientation vers un sous-programme spécifique en fonction de l'option choisie.

Lignes 2000 à 2100 : Création d'une structure.

Lignes 2500 à 2640 : Création d'une fiche.

Lignes 3000 à 3140 : Visualisation des fiches.

Lignes 3500 à 3620 : Impression des fiches.

Lignes 4000 à 4180 : Modification d'une fiche.

Lignes 4500 à 4710 : Suppression d'une fiche.

Lignes 5000 à 5190 : Lecture d'un fichier sur disquette.

Lignes 5500 à 5680 : Sauvegarde sur disquette du fichier en mémoire.

Lignes 6000 à 6080 : Fin du programme.

9/6

Jeux d'aventures

Si vous aimez les jeux d'aventures ou de rôles, ce chapitre vous concerne. En effet, nous allons étudier une série d'outils qui vous permettront de créer vos propres jeux d'aventures. Nous ferons très souvent allusion aux parties 5 et 6 de l'ouvrage. En effet, des programmes qui nous permettront de construire notre générateur de jeux d'aventures y sont étudiés.

Pour mieux cerner la façon dont va être étudié le générateur d'aventures, posons-nous la question suivante : « Qu'est-ce qu'un jeu d'aventures ? ».

Eh bien, un jeu d'aventures découle d'un scénario qui met en scène un ou plusieurs personnages évoluant dans un ou plusieurs décors. Le joueur participe à l'aventure et choisit son évolution.

Généralement, un seul chemin conduit à l'aboutissement de l'aventure, et plusieurs centaines, voire plusieurs milliers existent... d'où un temps plus ou moins long pour résoudre l'aventure.

Nous voyons donc que, pour écrire un jeu d'aventures, il faudra rédiger un scénario qui décrira :

- l'univers de jeu,
- les possibilités d'intervention du joueur et des actions qui en découlent.

Dans ce chapitre, nous allons voir comment :

- créer l'univers de jeu,
- définir les actions découlant des commandes du joueur,
- agrémenter le jeu d'une musique exécutée sous interruptions,
- créer des jeux et les exécuter.

Remarque

Parfois, il existe une confusion entre les appellations « jeux d'aventures » et « jeux de rôles ».

Les jeux de rôles sont, en fait, des jeux d'aventures dans lesquels le joueur se voit attribuer des caractéristiques et un rôle bien particulier (d'où le nom de jeu de rôle).

I. Création de l'univers de jeu

Au chapitre 10.1 partie 5 nous avons étudié un programme de tracé plein écran qui nous permettait de créer des écrans graphiques en MODE 1. Pour nous limiter dans la consommation en mémoire de tels écrans (16 kilo-octets par écran), nous allons construire une restriction de l'écran normal. L'écran pourra occuper 6 ou 10 kilo-octets.

Entrez les couleurs du dessin et sa taille : petit (6 kilo-octets) ou grand (10 kilo-octets). Le clavier est utilisé de la même manière que pour le programme de tracé en plein écran décrit au chapitre 10.1 partie 5 : (pavé numérique pour le déplacement point par point, touches numériques correspondantes en haut du clavier texte pour le déplacement en « AUTO-REPEAT »).

Le programme de dessin est le suivant :

```

1000 REM *****
1010 REM CONSTITUTION D'ECRANS POUR JEUX D'AVENTURES
1020 REM *****
1030
1040 MEMORY &4000
1050 'INTERFACE AVEC MBG ET ABG
1060 FOR I=0 TO 28:READ A:POKE &301B+I,A:NEXT I
1070 DATA &1,&64,&0,&11,&C4,&0,&26,&A,&2E,&A,&3E,&0,&CD,&35,&30,&C9
1080 DATA &1,&64,&0,&11,&C4,&0,&21,&0,&60,&CD,&A6,&30,&C9
1090 '-----
1100 'MBG EN 3035H
1110 FOR I=0 TO 112:READ A:POKE &3035+I,A:NEXT I
1120 DATA &1B,&B,&73,&20,&5B,&20,&61,&75,&20,&64,&65,&80,&61,&ED,&43,&37,&30,&ED
,&53,&39,&30,&32,&3D,&30,&7C,&32,&3B,&30,&7D,&32,&3C
,&30,&CD,&3,&B9,&21,&0,&60,&11,&94,&2,&3A,&3D,&30,&47,&B7,&2B,&4,&ED,&5A,&10,&FC
,&3A,&3B,&30,&77,&23,&22,&3E,&30,&ED,&5B
1130 DATA &37,&30,&2A,&39,&30,&CD,&1D,&BC,&22,&40,&30,&ED,&5B,&3E,&30,&3A,&3B,&3
0,&47,&7E,&12,&23,&13,&10,&FA,&2A,&40,&30,&CD,&26,&B
C,&22,&40,&30,&3A,&3C,&30,&3D,&32,&3C,&30,&20,&E4,&3E,&CF,&12,&ED,&53,&F0,&9C,&C
9
1140 '-----
1150 'ABG EN 30A6H
1160 FOR I=0 TO 72:READ A:POKE &30A6+I,A:NEXT

```

```

1170 DATA &1B,&A,&A,&3B,&D,&A,&4F,&52,&47,&20,&39,&43,&ED,&43,&AB,&30,&ED,&53,&A
A,&30,&7E,&32,&AE,&30,&23,&22,&AC,&30,&CD,&3,&B9,&ED
,&5B,&AB,&30,&2A,&AA,&30,&CD,&1D,&BC,&22,&B0,&30,&ED,&5B,&AC,&30
1180 DATA &3A,&AE,&30,&47,&1A,&77,&23,&13,&10,&FA,&2A,&B0,&30,&CD,&26,&BC,&22,&B
0,&30,&1A,&FE,&CF,&20,&EB,&C9
1190 -----
1200
1210 'Initialisation
1220 '
1230 STYLO=0 'Stylo leve
1240 BORDER 0:INK 0,0:INK 1,10:X=0:Y=0
1250 MODE 1
1260 INPUT"Affichage monochrome (O/N) ";R$:R$=UPPER$(R$)
1270 IF R$<>"O" AND R$<>"N" THEN 1250
1280 IF R$="O" THEN NBCOUL=1 ELSE NBCOUL=3
1290 PRINT
1300 FOR I=0 TO NBCOUL
1310 PRINT"INK";I," (0 A 26) ";:INPUT A:INK I,A
1320 NEXT I
1330 PRINT:PRINT "Taille ecran (1=Petit, 2=Grand) ?"
1340 A$=INKEY$:IF A$="" THEN 1340
1350 TE=ASC(A$)-48
1360 IF TE<>1 AND TE<>2 THEN SOUND 1,100,20:GOTO 1340
1370 IF TE=2 THEN POKE &3019,50:POKE &301F,57:POKE &3021,148:POKE &3029,50 ELSE
POKE &3019,75:POKE &301F,44:POKE &3021,124:POKE &302
9,75 'Taille de l'ecran a sauvegarder
1380 PRINT :INPUT"Chargement d'un ecran (O/N) ";R$:R$=UPPER$(R$)
1390 IF R$<>"O" AND R$<>"N" THEN 1380
1400 IF R$="O" THEN PRINT:INPUT"Nom de l'ecran ";ECR$:LOAD ECR$,&6000:CLS:CALL &
3028:GOTO 1420
1410 CLS
1420 IF TE=1 THEN X=152:Y=152:X1=X:Y1=Y:X2=488:Y2=388:PLOT 150,150,1:DRAWR 340,0
:DRAWR 0,240:DRAWR -340,0:DRAWR 0,-240
1430 IF TE=2 THEN X=102:Y=102:X1=X:Y1=Y:X2=538:Y2=388:PLOT 100,100,1:DRAWR 440,0
:DRAWR 0,290:DRAWR -440,0:DRAWR 0,-290

```



```
1440 PLOT X,Y,2
1450 '
1460 'Boucle principale
1470 '
1480 A$=INKEY$:IF A$="" THEN 1480 'Attente action
1490 A=ASC(A$)
1500 IF A<58 AND A>48 THEN ON A-48 GOTO 1510,1520,1530,1540,1440,1550,1560,1570,
1580 ELSE 1590
1510 IF X>X1 AND Y>Y1 THEN PLOT X,Y,STYLO:Y=Y-2:X=X-2:GOTO 1440 ELSE 1620 'En ba
s a gauche
1520 IF Y>Y1 THEN PLOT X,Y,STYLO:Y=Y-2:GOTO 1440 ELSE 1620 'Vers le as
1530 IF X<X2 AND Y>Y1 THEN PLOT X,Y,STYLO:Y=Y-2:X=X+2:GOTO 1440 ELSE 1620 'En ba
s a droite
1540 IF X>X1 THEN PLOT X,Y,STYLO:X=X-2:GOTO 1440 ELSE 1620 'Vers la gauche
1550 IF X<X2 THEN PLOT X,Y,STYLO:X=X+2:GOTO 1440 ELSE 1620 'Vers la droite
1560 IF X>X1 AND Y<Y2 THEN PLOT X,Y,STYLO:Y=Y+2:X=X-2:GOTO 1440 ELSE 1620 'En ha
ut a gauche
1570 IF Y<Y2 THEN PLOT X,Y,STYLO:Y=Y+2:GOTO 1440 ELSE 1620 'Vers le haut
1580 IF X<X2 AND Y<Y2 THEN PLOT X,Y,STYLO:Y=Y+2:X=X+2:GOTO 1440 ELSE 1620 'En ha
ut a droite
1590 IF A=95 OR (A>32 AND A<33+NBCOUL) THEN 1640 'Changement de stylo
1600 IF A=13 THEN 1690 'Fin de trace
1610 GOTO 1480 'Boucle de trace
1620 SOUND 1,100,20:GOTO 1440
1630 '-----
1640 REM Changement de couleur stylo
1650 '
1660 IF A=95 THEN STYLO=0 ELSE STYLO=A-32
1670 GOTO 1440
1680 '-----
1690 REM Fin de trace
1700 '
1710 PLOT X,Y,0:CALL &3018 'Sauvegarde ecran
1720 CLS
```

```
1730 LOCATE 1,10:INPUT"Sauvegarde magnetique (O/N) ";R$:R$=UPPER$(R$)
1740 IF R$<>"O" AND R$<>"N" THEN 1730
1750 IF R$="N" THEN CLS:CALL &3028:GOTO 1440 'Restitution ecran
1760 LOCATE 1,12:INPUT"Nom de l'ecran ";N$
1770 LL=PEEK(&9CF0)+PEEK(&9CF1)*256-&6000+1
1780 SAVE N$,B,&6000,LL 'Sauvegarde
1790 LOCATE 1,14:INPUT"Poursuite (O/N) ";R$:R$=UPPER$(R$)
1800 IF R$="O" THEN CLS:CALL &3028:GOTO 1440 'Restitution ecran
1810 END
```

Lignes 1050 à 1180 : Chargement des sous-programmes ASSEMBLEUR

Lignes 1200 à 1400 : Initialisation du programme

Lignes 1420 à 1430 : Tracé des limites de l'écran

Lignes 1450 à 1610 : Gestion du curseur

Lignes 1660 à 1670 : Changement de la couleur du tracé

Lignes 1710 à 1810 : Fin du tracé avec
sauvegarde d'écran (Ligne 1780)
et/ou Retour au tracé (Ligne 1800).

Remarques :

Les fichiers écran générés par ce programme sont affichables grâce au sous-programme « ABG », contrairement aux fichiers écran générés par le programme de dessin en plein écran étudié au chapitre 10.1 partie 5 qui étaient affichables par la commande LOAD "ECRAN",&C000.

Les sous-programmes Assembleur utilisés ont été décrits au chapitre 10.2 partie 5.

Il s'agit de :

- MBG (Mémorisation de blocs graphiques)
- ABG (Affichage de blocs graphiques)
- Interface BASIC/MBG-ABG

9/6.1

Analyse syntaxique d'une phrase

Nous avons vu comment créer les écrans d'un jeu d'aventures. De nombreuses étapes restent encore à franchir avant de pouvoir exécuter notre premier jeu.

Une des actions les plus importantes effectuée par un jeu d'aventures consiste à analyser les ordres entrés par le ou les joueurs.

Tous les jeux d'aventures capables d'interpréter un ordre ou une suite d'ordres entrés en toutes lettres au clavier mettent en œuvre un analyseur syntaxique. Comme son nom l'indique, un analyseur syntaxique est un programme qui analyse la syntaxe d'une phrase. Dans les cas conventionnels, le programme extrait des mots clés de la phrase à analyser, et vérifie si ces derniers peuvent être associés dans la salle actuelle.

Exemple :

La liste des mots clés est :

(PRENDRE, CASSER, AVANCER, NORD, SUD, EST, OUEST, BATON, CHANDELLE)

Le joueur entre la phrase :

« PRENDRE LE BATON QUI SE TROUVE SUR LA PIERRE »

Les mots clés identifiés sont **PRENDRE** et **BATON**.

Si, dans la salle où se trouve le joueur, le cas a été prévu, le programme agira en conséquence. Par exemple, il pourra répondre **« TU L'AS MAINTENANT DANS LES MAINS »**, et mémoriser que le joueur a un bâton en main pour la suite du jeu.

L'analyse syntaxique étudiée dans ce paragraphe portera sur les phrases ayant la structure suivante :

VERBE + SUJET

Il y aura, dans un premier temps, extraction des mots clés VERBE et SUJET ; dans un deuxième temps, analyse de la corrélation entre VERBE et SUJET en tenant compte de la salle où se trouve le joueur ; et enfin, dans un troisième temps, l'action découlant de la reconnaissance nulle, partielle ou totale de l'ordre entré par le joueur.

Phase 1 : Extraction de mots clés

Problème à résoudre

Soit une phrase de longueur n , et soit un mot clé de longueur p . Comment déterminer si le mot clé fait partie de la phrase ?

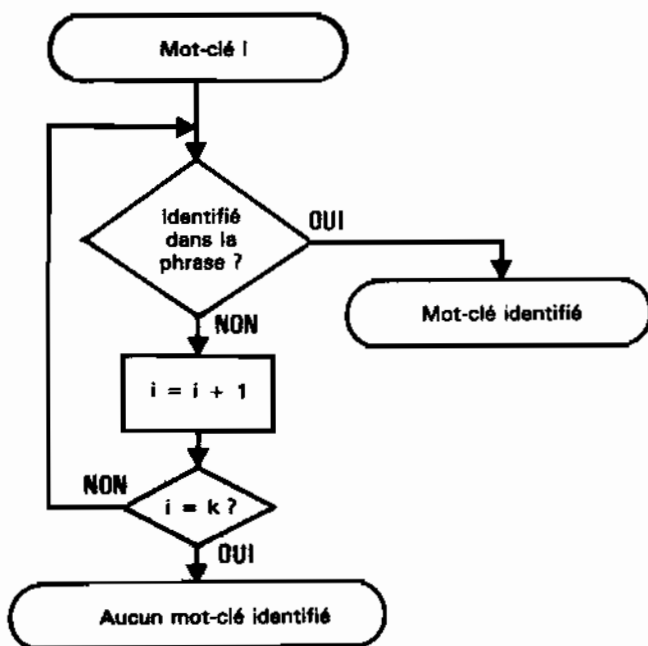
Résolution du problème

Parcourir la phrase de la première lettre à la $(n-p)$ ème lettre. Extraire à chaque fois un bloc de p lettres, et le comparer au mot clé. S'il y a similitude, c'est que le mot clé fait partie de la phrase.

Corrolaire

Si k mots clés sont définis, il faudra faire l'opération qui vient d'être décrite jusqu'à ce qu'un mot clé soit rencontré ou jusqu'à ce que les k mots clés aient été passés en revue.

Ce corrolaire peut être représenté comme suit :



Le problème que nous venons d'exposer peut être résolu par un programme Basic si le nombre de mots clés est faible, mais nous avons préféré présenter une version Assembleur de ce programme pour vous permettre de créer des jeux d'aventures complexes dans lesquels le temps d'analyse des phrases entrées par le joueur est très court.

Le listing du programme est le suivant :

```

1          ;
2          ;Analyse d'une phrase
3          ;Recherche de mots-cle
4          ;Entree: SCE=Pointeur Source
5          ;      TXT=Point. phrase entree
6          ;Sortie: NOCLE=No de mot-cle ou FF
7          ;
8          ;-----
9          ORG 9000H
10         LOAD 9000H
11         ;
12         NOCLE: DS 1          ;No Mot-cle
13         MOTCLE: DS 1        ;Nbre de Mots-cle
14         SCE: DS 2           ;Pointeur Source
15         TXT: DS 2           ;Pointeur phrase
16         STXT: DS 2          ;Sauvegarde TXT
17         ;
18 9008 2A0490 LD HL,(TXT)
19 900B 220690 LD (STXT),HL    ;Sauvegarde
20 900E 3E01 LD A,1
21 9010 320090 LD (NOCLE),A
22 9013 2A0290 LD HL,(SCE)
23 9016 7E LD A,(HL)          ;Nb de mots-cle
24 9017 320190 LD (MOTCLE),A  ;Sauvegarde
25 901A 23 INC HL
26         BX: EQU $
27 901B 220290 LD (SCE),HL    ;Mot cle No n

```

Partie 9 : Programmes

```

28 901E CD6590      CALL RECLEN      ;Rech LEN(Mot-cle)
29 9021 0E00        LD    C,0        ;Rech en debut de phrase
30                PO:    EQU    $
31 9023 D5          PUSH BC          ;Sauv.garde
32 9024 2A0290      LD    HL,(SCE)
33 9027 ED5B0490    LD    DE,(TXT)
34 902B CD7090      CALL CMPARE      ;Comparaison
35 902E B7          OR    A
36 902F C1          POP  BC
37 9030 C8          RET  Z           ;Mot-cle identifie
38 9031 0C          INC  C
39 9032 79          LD    A,C
40 9033 FE28        CP    40
41 9035 2809        JR    Z,P2      ;Fin de comparaison
42 9037 2A0490      LD    HL,(TXT)
43 903A 23          INC  HL
44 903B 220490      LD    (TXT),HL   ;Prochain test
45 903E 18E3        JR    P0
46                P2:    EQU    $
47 9040 3A0190      LD    A,(MOTCLE)
48 9043 3D          DEC  A
49 9044 320190      LD    (MOTCLE),A
50 9047 2816        JR    Z,P3
51 9049 2A0690      LD    HL,(STXT)
52 904C 220490      LD    (TXT),HL   ;Restitution pointeur
53 904F 2A0290      LD    HL,(SCE)
54 9052 CD6590      CALL RECLEN
55 9055 23          INC  HL
56 9056 3A0090      LD    A,(MOTCLE)
57 9059 3C          INC  A

```

```

58 905A 320090      LD    (NOCLE),A
59 905D 18BC        JR     BX
60                P3:    EQU    $
61 905F 3EFF        LD     A,OFFH      ;Aucun mot-cle identifie
62 9061 320090      LD     (NOCLE),A
63 9064 C9          RET
64                ;-----
65                ;Recherche de longueur de mot-cle
66                ;-----
67                ;Entree: HL=@ Mot-Cle
68                ;Sortie: B =Longueur du mot-cle
69                ;-----
70                RECLN:  EQU    $
71 9065 0600        LD     B,0
72 9067 05          DEC    B
73                R1:    EQU    $
74 9068 04          INC    B
75 9069 7E          LD     A,(HL)
76 906A FEFF        CP     OFFH
77 906C 08          RET     Z          ;B=LEN(Mot-cle)
78 906D 23          INC    HL
79 906E 18FB        JR     R1
80                ;-----
81                ;Comparaison de chaines ASCII
82                ;-----
83                ;Entree: HL=@ Source
84                ;      DE=@ Texte a comparer
85                ;      B =Longueur comparaison
86                ;Sortie: A =0 Si comparaison OK

```

```

87          ;          A =1 Sinon
88          ;-----
89          CMPARE:    EQU  $
90          C0:        EQU  $
91 9070 1A            LD   A,(DE)
92 9071 BE            CP   (HL)
93 9072 2006          JR   NZ,C1          ;Difference
94 9074 23            INC  HL
95 9075 13            INC  DE
96 9076 10FB          DJNZ C0          ;Boucle de comparaison
97 907B 1B03          JR   C2          ;Comparaison OK
98          C1:        EQU  $          ;Comparaison ratee
99 907A 3E01          LD   A,1
100 907C C9           RET
101          C2:        EQU  $          ;Comparaison reussie
102 907D AF           XOR  A
103 907E C9           RET
104          END

```

Décomposition du programme

Initialisation	Lignes 18 à 25
Recherche de la longueur du mot clé n	Ligne 28
Test d'identification du mot clé n	Lignes 29 à 45
Passage au prochain mot clé	Lignes 47 à 63
Recherche de la longueur d'un mot clé	Lignes 71 à 79
Comparaison mot clé/portion de phrase	Lignes 89 à 103

**Phase 2 : Identification d'un couple VERBE/SUJET
et recherche d'une corrélation entre eux dans la salle courante**

Prenons l'exemple suivant :

Supposons que l'ensemble des mots clés VERBE soit le suivant :

(PRENDRE, VERSER)

et que l'ensemble des mots clés SUJETS soit le suivant :

(LAMPE, PO, BATON)

La représentation mémoire adoptée par la suite pour ces deux ensembles sera la suivante :

```
02/50/52/15/1E/44/52/45/FF/56/45/52/53/45/52/FF
  P R E N D R E      V E R S E R
```

2 Verbes Terminateurs

```
03/1C/41/4D/50/45/FF/50/4F/FF/42/41/54/4F/4E/FF
  L A M P E      P O      B A T O N
```

3 Sujets Terminateurs

Supposons que le jeu soit constitué de trois scènes (Ces scènes ont été créées avec le générateur d'écrans présenté en Partie 9, chapitre 6). Supposons que le nom de ces scènes soit E1, E2 et E3.

La représentation mémoire de ces scènes sera :

```
03/65/31/FF/65/32/FF/65/33/FF
  E 1      E 2      E 3
```

3 Scènes Terminateurs

Dans un jeu d'aventures classique, l'identification d'un ordre comme « **PRENDRE LA LAMPE** » par exemple, peut produire :

- un changement de scène ;
- une réponse de l'ordinateur ;
- la modification du nombre de « pointe de vie » du joueur ;
- la fin du jeu.

Supposons que les conditions suivantes aient été définies :

Salle	Verbe	Sujet	Changement de pièce	Réponse	Points	Fin
1	1	1	0	1	138	0
2	1	3	3	2	128	1
2	1	2	0	3	158	0

Soit l'ensemble des réponses suivant :

(TU L'AS DANS LES MAINS, UN SCORPION ETAIT CACHE DESSOUS, CA FAIT DU BIEN)

Analysons la signification du tableau ci-dessus :

- *Ligne 1* : Si le joueur se trouve en salle 1 et qu'il donne l'ordre (VERBE = 1, SUJET = 1), « **PRENDRE LAMPE** », aucun changement de salle n'est effectué.
L'ordinateur affiche (réponse 1) « **TU L'AS DANS LES MAINS** », le nombre de points de vie augmente de 10 (128 + 10), et la partie n'est pas finie (FIN = 0).
- *Ligne 2* : Si le joueur se trouve en salle 2 et qu'il donne l'ordre (VERBE = 1, SUJET = 3) « **PRENDRE BATON** », il y a passage en salle 3, l'ordinateur affiche « **UN SCORPION ETAIT CACHE DESSOUS** », le nombre de points de vie est inchangé (128), et la partie est terminée (FIN = 1).
- *Ligne 3* : Si le joueur se trouve en salle 2 et qu'il donne l'ordre (VERBE = 1, SUJET = 2) « **PRENDRE PO** », aucun changement de salle n'est effectué (Changt. = 0).
L'ordinateur affiche « **CA FAIT DU BIEN** », le nombre de points de vie augmente de 30 (128 + 30) et la partie peut se poursuivre (FIN = 0).

Remarque :

L'information « Points » (de vie) est un nombre signé sur 8 bits : par convention, 128 correspond à un accroissement nul du nombre de points de vie, $128 + n$ correspond à un accroissement de n du nombre de points de vie, et $128 - n$ à une diminution de n du nombre de points de vie.

Reprenons l'exemple précédent et voyons quel sera son codage en mémoire.

Rappel : Scènes (E1, E2, E3)
 Verbes (PRENDRE, VERSER)
 Sujets (LAMPE, PO, BATON)
 Réponses (TU L'AS DANS LES MAINS, UN SCORPION
 ETAIT CACHE DESSOUS, CA FAIT DU BIEN)

Codage mémoire

03/65/31/FF/65/32/FF/65/33/FF
 E 1 E 2 E 3

02/50/52/45/4E/44/52/45/FF/56/45/52/53/45/52/FF
 P R E N D R E V E R S E R

03/40/41/4D/50/45/FF/50/4F/FF/42/41/54/4F/4E/FF
 L A M P E P O B A T O N

03/54/55/20/40/27/41/53/20/44/41/4E/53/20/40/45/53/20/4D/41/47/4E/53/FF
 T U L ' A S D A N S L E S M A I N S

55/4E/20/53/43/4F/52/40/49/4F/4E/20/45/54/41/49/54/20/43/11/13/1P/15
 U N S C O R P I O N E T A I T C A C H E

20/44/45/53/53/4F/55/53/FF/43/41/20/46/41/49/54/20/44/55/20
 D E S S O U S C A F A I T D U

42/49/45/4E/FF
 D I E N

03/01/01/01/00/01/8A/00/02/01/03/03/02/80/01/02/01/02/00/03/9E/00/FF
 C S V S C R F S V S C R P F S V S C R P F
 o a e u h e t i a e u h e t i a e u h e t i
 n l r j g p s n l r j g p s n l r j g p s n
 d l b e t l b e t l b e t
 e e t e e t e e t

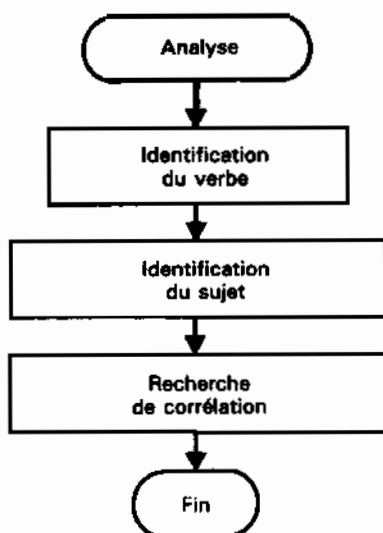
Maintenant qu'une convention est adoptée pour le codage en mémoire des divers éléments, voyons comment va s'organiser le programme d'analyse de la phrase.

1^{er} temps : Recherche de verbe ;

2^e temps : Recherche de sujet ;

3^e temps : Recherche de corrélation (Salle, Verbe, Sujet).

Cette décomposition du programme est clairement représentée par l'organigramme suivant :



La recherche de corrélation se fait comme suit :

- 1) Parcours de la table des conditions jusqu'à tomber sur une condition opérant sur la salle courante ;
- 2) Même démarche qu'au 1), mais concernant le verbe ;
- 3) Même démarche qu'au 1) mais concernant le sujet ;
- 4) Extraction des actions à effectuer (Changement de salle, réponse, modification du nombre de points de vie, fin du jeu).

Le listing du programme est le suivant :

```

1      ; Recherche de Verbe, Sujet et
2      ; d'une relation entre eux en
3      ; fonction de la salle courante
4      ; Entree: SALLE=No salle courante
5      ;      COND =@ Debut conditions
6      ;      SUJET=@ Debut sujets
7      ;      VERBE=@ Debut verbes
8      ; Sortie: CH=No de salle ou FF
9      ;      RE=No de reponse ou FF
10     ;      PT=Nbre de points ou FF
  
```

```

11          ;          FI=1 si fin de partie
12          ;----->
13          ORG 907FH
14          LOAD 907FH
15          SALLE:    DS 1          ;No de salle
16          NCOND:    DS 1          ;Nombre de conditions
17          COND:     DS 2          ;@ Debut conditions
18          VERBE:    DS 2          ;@ Debut verbes
19          SUJET:    DS 2          ;@ Debut sujets
20          SU:       DS 1          ;Sauv. Sujet
21          VE:       DS 1          ;Sauv. Verbe
22          CH:       DS 1          ;No de salle
23          RE:       DS 1          ;No de reponse
24          PT:       DS 1          ;Nombre de points
25          FI:       DS 1          ;Indic. fin de partie
26          NOCLE:    EQU 9000H     ;No aut-cle
27          SCE:      EQU 9002H     ;Pointeur Source
28          TXT:      EQU 9004H     ;Pointeur Phrase
29          TEXTE:    EQU 4300H     ;Debut phrase
30          RECHER:   EQU 9008H     ;Pt entree Rech.
31          ;-----
32          ;RAZ Zone de Sortie
33 908D 218990      LD HL,CH
34 9090 AF          XOR A
35 9091 0604        LD B,4
36          RAZ:     EQU $
37 9093 77          LD (HL),A
38 9094 23          INC HL
39 9095 10FC        DJNZ RAZ

```

```

40      ; -----
41 9097 2A8390      LD    HL,(VERBE)
42 909A 220290      LD    (SCE),HL      ;Debut des verbes
43 909D 210043      LD    HL,TEXTE
44 90A0 220490      LD    (TXT),HL      ;Phrase utilisateur
45 90A3 CD0890      CALL RECHER      ;Recherche Verbe
46 90A6 3A0090      LD    A,(NOCLE)
47 90A9 32B890      LD    (VE),A      ;Sauvegarde du verbe
48 90AC 2A8590      LD    HL,(SUJET)
49 90AF 220290      LD    (SCE),HL      ;Debut des Sujets
50 90B2 210043      LD    HL,TEXTE
51 90B5 220490      LD    (TXT),HL      ;Phrase Utilisateur
52 90B8 CD0890      CALL RECHER      ;Recherche Sujet
53 90BB 3A0090      LD    A,(NOCLE)
54 90BE 32B790      LD    (SU),A      ;Sauvegarde du sujet
55      ; -----
56      ;Recherche de conditions
57 90C1 2A8190      LD    HL,(COND)
58 90C4 7E          LD    A,(HL)
59 90C5 32B090      LD    (NCOND),A      ;Nombre de conditions
60      R2:      EQU    $      ;Boucle de recherche
61 90C8 3A7F90      LD    A,(SALLE)
62 90CB 23          INC    HL
63 90CC BE          CP     (HL)      ;Salle courante ?
64 90CD 2024        JR     NZ,R3      ;Non
65 90CF 23          INC    HL
66 90D0 3AB890      LD    A,(VE)
67 90D3 BE          CP     (HL)      ;Verbe courant ?

```

```

68 90D4 201E          JR    NZ,R4          ; Non
69 90D6 23            INC    HL
70 90D7 3AB790         LD     A,(SU)
71 90DA BE             CP     (HL)          ; Sujet courant ?
72 90DB 2018          JR     NZ,R5          ; Non
73                    ; A ce pt, un contexte est trouve
74 90DD 23            INC    HL
75 90DE 7E            LD     A,(HL)
76 90DF 32B990         LD     (CH),A        ; Changement de salle
77 90E2 23            INC    HL
78 90E3 7E            LD     A,(HL)
79 90E4 32BA90         LD     (RE),A        ; Reponse
80 90E7 23            INC    HL
81 90E8 7E            LD     A,(HL)
82 90E9 32BB90         LD     (PT),A        ; Points
83 90EC 23            INC    HL
84 90ED 7E            LD     A,(HL)
85 90EE 32BC90         LD     (FI),A        ; Fin de partie
86 90F1 180F          JR     FIN            ; Fin du S/P
87 90F3 23            R3:    INC    HL
88 90F4 23            R4:    INC    HL
89 90F5 23            R5:    INC    HL
90 90F6 23            INC    HL
91 90F7 23            INC    HL
92 90F8 23            INC    HL
93 90F9 3AB090         LD     A,(NCOND)
94 90FC 3D            DEC     A
95 90FD 32B090         LD     (NCOND),A     ; Nombre de conditions
96 9100 20C6          JR     NZ,R2          ; Boucle de recherche
97                    ;
98                    FIN:    EQU    $
99 9102 C9            RET
100                   END

```

Décomposition du programme

Initialisation	Lignes 33 à 39
Recherche de verbe	Lignes 41 à 47
Recherche de sujet	Lignes 48 à 54
Recherche de conditions	Lignes 57 à 99

9/6.2

Fonction LOCATE-INPUT

Le générateur de jeux d'aventures que nous étudierons dans un des prochains compléments utilise une fonction Assembleur bien utile appelée **LOCATE-INPUT**. Cette fonction positionne le curseur à un endroit quelconque sur l'écran (équivalent à la fonction Basic **LOCATE**), et acquiert une chaîne de caractères (équivalent à la fonction Basic **INPUT**).

Pour recréer la fonction **LOCATE**, nous allons utiliser plusieurs macros du firmware :

- **TXT CUR ENABLE** pour déclarer l'existence du curseur,
- **TXT CUR ON** pour faire apparaître le curseur sur l'écran,
- **TXT SET CUR** pour positionner le curseur sur l'écran.

Pour recréer la fonction **INPUT**, nous allons également utiliser plusieurs macros du firmware :

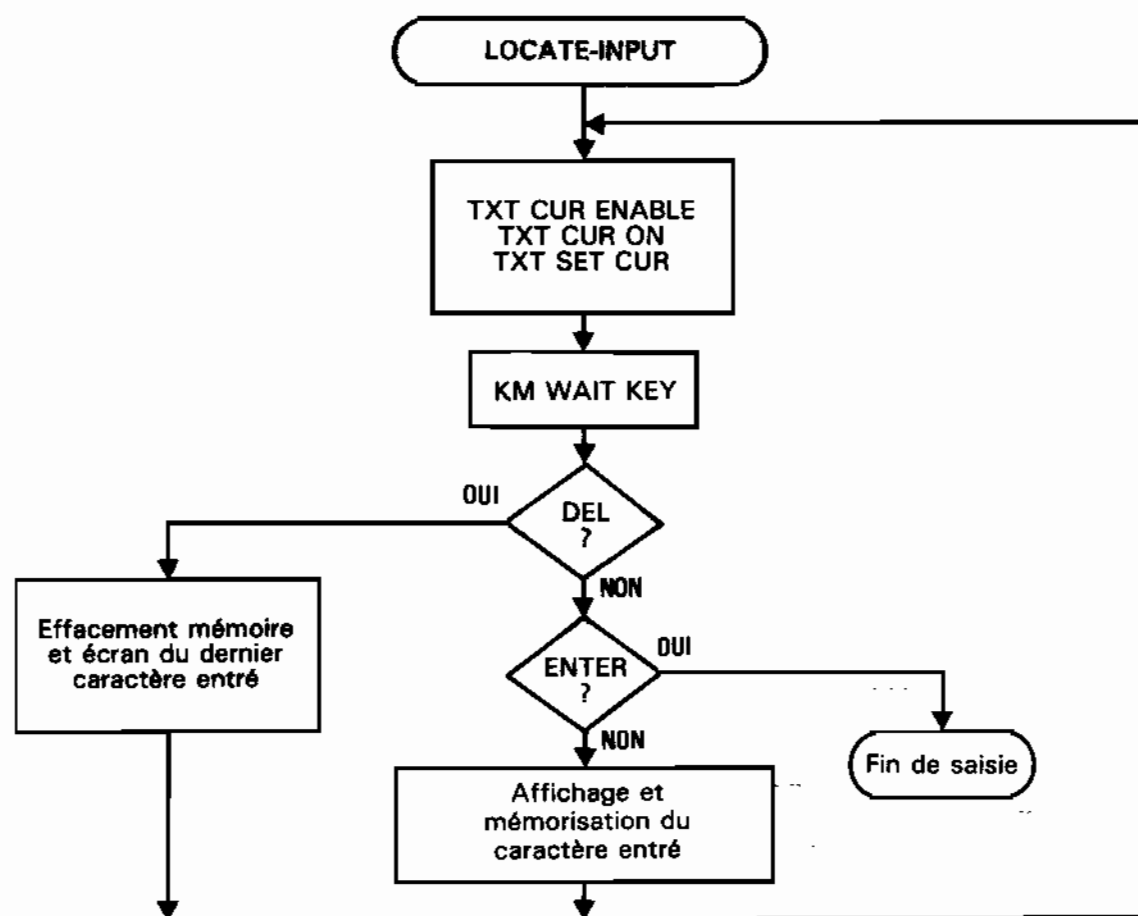
- **KM WAIT KEY** pour acquérir un caractère au clavier,
- **TXT WR CHAR** pour afficher un caractère sur l'écran.

Reportez-vous en Partie 4, chapitre 2.7, pages 5 et 12 pour avoir plus de détails sur ces macros.

Pour acquérir une suite de caractères avec la fonction **LOCATE-INPUT**, les étapes suivantes doivent être respectées :

- validation de l'affichage du curseur ;
- positionnement du curseur ;
- boucle sur la macro **KM WAIT KEY** jusqu'à ce que la touche **<Enter>** soit pressée ;
- affichage des caractères entrés sur l'écran grâce à la macro **TXT WR CHAR** ;
- stockage des caractères en mémoire en gérant un pointeur. Pour permettre une plus grande souplesse, la touche **DEL** est prise en compte. Elle permet d'effacer le caractère qui se trouve à gauche du curseur.

Ces actions sont résumées dans l'organigramme suivant :



Le listing du programme Assembleur est le suivant :

```

1      ;Fonction LOCATE+INPUT
2      ;-----
3      ORG 933FH
4      LOAD 933FH
5      ; Macros FIRMWARE
6      SETCUR: EQU 0BB75H      ;TXT SET CURSOR
7      TXTOUT: EQU 0BB5DH      ;TXT WR CHAR
8      WAITK: EQU 0BB18H      ;KM WAIT KEY
9      TCUREN: EQU 0BB7BH      ;TXT CUR ENABLE
10     TCURON: EQU 0BB81H      ;TXT CUR ON
11     TCUROF: EQU 0BB84H      ;TXT CUR OFF
12     ;
13     ;EQU Clavier
14     RET: EQU 13              ;RETurn
15     DEL: EQU 7FH             ;DELete
16     BLANC: EQU 32            ;Espace
17     CTRLH: EQU 8             ;CTRL+H
18     ;
19     ;Reservation de zones
20     SAVHL: DS 2              ;Sauvegarde reg. HL
21     SAV2: DS 2               ;2eme Sauv. reg. HL
22     SAVA: DS 1               ;Sauvegarde reg. A
23     MAX: DS 1                ;Long. max INPUT
24     ;-----
25     ;
26     ;Fonction LOCATE+INPUT avec
27     ;gestion de la touche DEL
28     ;
29     ;Entree: H =Colonne du LOCATE
30     ;          L =Ligne du LOCATE

```

```

31      ;      A =Nb Max de caracteres
32      ;      DE=Adresse de stockage
33      ;
34      ;Point d'entree:LOCINP
35      ;
36      LOCINP:      EQU      $      ;LOCATE+INPUT
37 9345 324493      LD      (MAX),A      ;Lgr Max
38 9348 223F93      LD      (SAVHL),HL      ;Sauv Colonne/Ligne
39 934B CD7BBB      CALL TCUREN      ;Cursor Enable
40 934E CD81BB      CALL TCURDN      ;Cursor DN
41 9351 CD75BB      CALL SETCUR      ;LOCATE
42 9354 0600      LD      B,0      ;Init pointeur texte
43 9356 2A3F93      LD      HL,(SAVHL)      ;Restit. pos curseur
44      IO:      EQU      $
45 9359 CD18BB      CALL WAITK      ;KM WAIT KEY
46 935C FE7F      CP      DEL      ;Est-ce un DEL ?
47 935E 2826      JR      Z,I1      ;Oui
48 9360 FE0D      CP      RET      ;Est-ce un RET ?
49 9362 284F      JR      Z,I3      ;Oui
50 9364 FE08      CP      CTRLH      ;Est-ce un CTRL+H ?
51 9366 2855      JR      Z,I5      ;Oui
52 9368 324393      LD      (SAVA),A      ;Ni DEL, ni RET
53      III:      EQU      $
54 936B 3A4493      LD      A,(MAX)
55 936E B8      CP      B
56 936F 38EB      JR      C,I0      ;Pas possible d'ecrire
57 9371 C5      PUSH BC
58 9372 D5      PUSH DE
59 9373 E5      PUSH HL
60 9374 3A4393      LD      A,(SAVA)
61 9377 CD5DBB      CALL TXTOUT      ;Ecriture
62 937A E1      POP      HL

```

63 937B D1	POP DE	
64 937C C1	POP BC	
65 937D 04	INC B	
66 937E 24	INC H	;Pos curseur + 1
67 937F 3A4393	LD A,(SAVA)	;Caractere tape
68 9382 12	LD (DE),A	;Memorisation
69 9383 13	INC DE	;Memoire suivante
70 9384 18D3	JR IO	;Boucle de saisie
71 I1:	EQU #	;Traitement du DEL
72 9386 224193	LD (SAV2),HL	;Sauv pos curseur
73 9389 D5	PUSH DE	
74 938A EB	EX DE,HL	
75 938B 2A3F93	LD HL,(SAVHL)	
76 938E 7A	LD A,D	
77 938F BC	CP H	
78 9390 2828	JR Z,I4	;DEL Impossible
79 9392 D1	POP DE	;DEL Possible
80 9393 05	DEC B	
81 9394 2A4193	LD HL,(SAV2)	;Pos curseur
82 9397 25	DEC H	
83 9398 224193	LD (SAV2),HL	
84 939B CD75BB	CALL SETCUR	;LOCATE
85 939E C5	PUSH BC	
86 939F D5	PUSH DE	
87 93A0 3E20	LD A,BLANC	;Code "Blanc"
88 93A2 CD5DBB	CALL TXTOUT	;Ecriture
89 93A5 D1	POP DE	
90 93A6 C1	POP BC	
91 93A7 2A4193	LD HL,(SAV2)	;Restit @ Curseur
92 93AA CD75BB	CALL SETCUR	
93 93AD 2A4193	LD HL,(SAV2)	;Restit @ Curseur
94 93B0 1B	DEC DE	;Pointeur memoire - 1

```

1          ;Activation de INPUT et
2          ;analyse de la reponse utilisateur
3          ;-----
4          ORB  91A0H
5          LOAD 91A0H
6          SABAS: DS  1          ;No Salle BASIC
7          SALLE: EQU 907FH      ;Salle ds ANALYS
8          INPUT: EQU 9345H      ;Input ASM
9          ANALYS: EQU 908DH     ;Anal. syntaxiq.
10         ;-----
11 91A1 211701      LD  HL,117H      ;Colonne/Ligne
12 91A4 3E26        LD  A,38        ;Nbre max caracteres
13 91A6 110043      LD  DE,4300H    ;@ Stockage reponse
14 91A9 CD4593      CALL INPUT
15 91AC 3AA091      LD  A,(SABAS)    ;Salle BASIC
16 91AF 327F90      LD  (SALLE),A
17 91B2 CDBD90      CALL ANALYS      ;Analyse reponse
18 91B5 C9          RET            ;Retour au BASIC
19                END

```

- Ligne 11 : Positionnement de la colonne et de la ligne de début de saisie dans le registre HL.
- Ligne 12 : Taille maximale de la chaîne saisie dans le registre A.
- Ligne 13 : Adresse de stockage de la réponse dans le registre DE.
- Ligne 14 : LOCATE-INPUT.
- Ligne 17 : Analyse syntaxique de la chaîne entrée.

9/6.3

Fonction HELP

La fonction **HELP** peut rendre de grands services aux joueurs de jeux d'aventures : elle donne la liste des mots clés « compris » par la machine.

Pour accéder à cette liste, il suffit d'appuyer simultanément sur les touches **Ctrl** et **H**. Les touches flèches vers le haut et vers le bas font défiler les mots clés dans une fenêtre de texte située en bas de l'écran.

La fenêtre texte est définie grâce à la macro **TEXT WIN ENABLE** (Cf. point d'entrée **OBB66H** du Firmware) puis effacée grâce à la macro **TEXT CLEAR WIN** (Cf. point d'entrée **OBB6CH** du Firmware) [voir Partie 4, Chap. 2.7, pages 13 et 14].

Les trois premiers mots clés sont affichés grâce aux macros **TEXT SET-CURSOR** et **TEXT OUTPUT** (Cf. points d'entrées **OBB75H** et **OBB5AH** du Firmware) [Voir Partie 4, Chap. 2.7 pages 12 et 14].

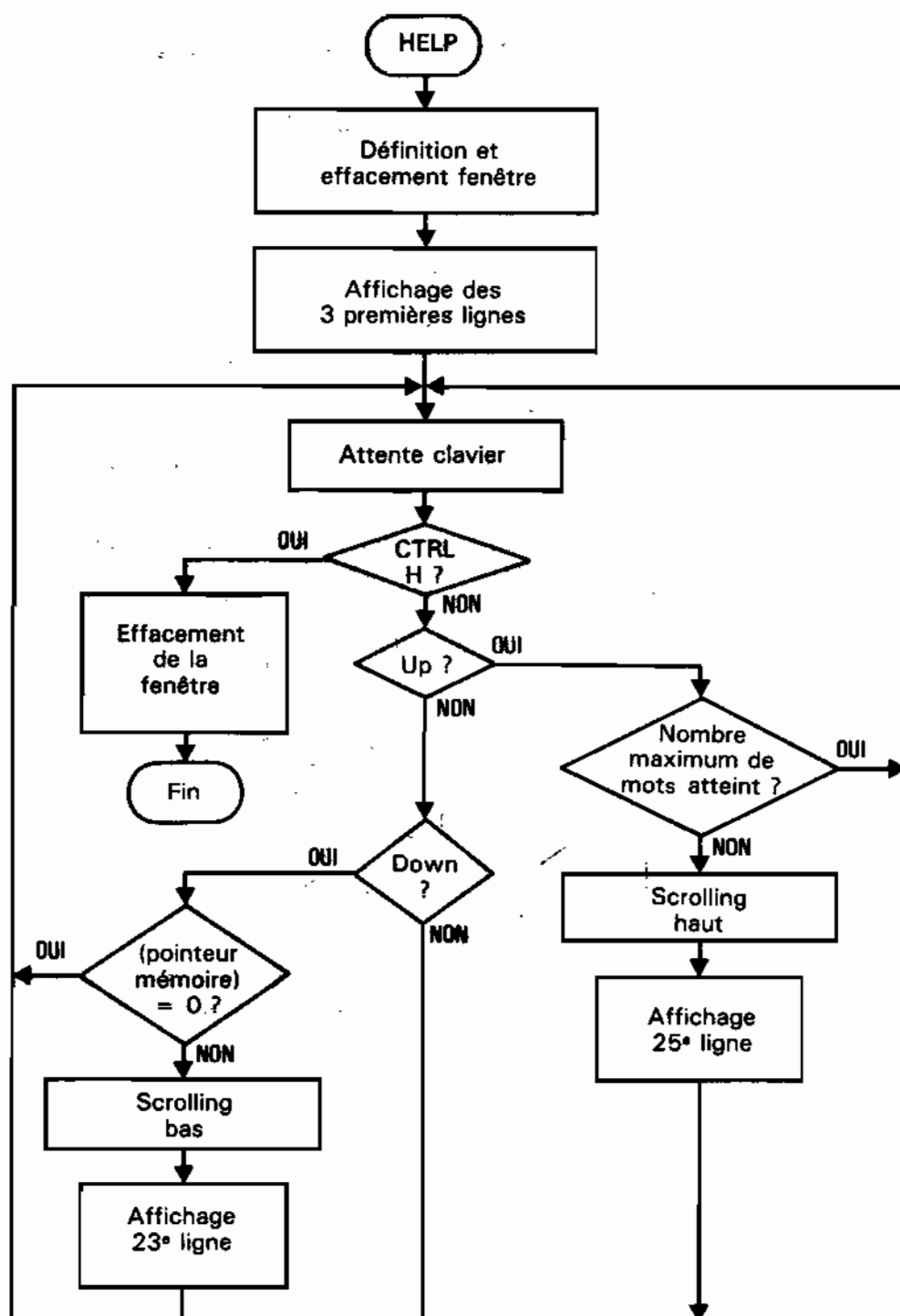
Le programme se met alors en attente d'une action au clavier :

- si la séquence **Ctrl H** est reconnue, la fenêtre **HELP** est effacée, et le contrôle est redonné au programme principal ;
- si la touche flèche vers le haut est reconnue, et s'il reste des mots clés à lister, un scrolling vers le haut est exécuté ;
- si la touche flèche vers le bas est reconnue, et si le premier mot clé ne se trouve pas sur la première ligne de la fenêtre, un scrolling vers le bas est exécuté.

Les autres touches du clavier n'ont aucun effet.

Le scrolling est obtenu grâce à la macro **ROLL**. Pour plus de détails, reportez-vous au point d'entrée **OBC50H** du Firmware (voir Partie 4, Chap. 2.7 page 38).

Les actions effectuées par le programme correspondent à la logique de l'ordinogramme suivant :



Le listing du programme Assembleur est le suivant :

```

1          ;Fonction HELP
2          ;I/O: Aucune,
3          ;Pt d'entree: HELP
4          ;-----
5          ORG 920CH
6          LOAD 920CH
7          ADCOUR: DS 2
8          L1: DS 1
9          SS: DS 2
10         CUROFF: EQU 0BB84H ;TXT CUR OFF
11         SETCUR: EQU 0BB75H ;TXT SET CURSOR
12         TXTOUT: EQU 0BB5AH ;TXT OUTPUT
13         WAITCH: EQU 0BB06H ;TXT WAIT CHAR
14         DEFWIN: EQU 0BB66H ;Def. Fenetre
15         EFFWIN: EQU 0BB6CH ;Eff. Fenetre
16         ROLL: EQU 0BC50H ;SCR SW ROLL
17         ;-----
18         HELP: EQU $
19 9211 211600 LD HL,16H
20 9214 111827 LD DE,2718H
21 9217 CD66BB CALL DEFWIN ;Definition Fenetre
22 921A CD6CBB CALL EFFWIN ;Effacement Fenetre
23 921D CD84BB CALL CUROFF
24 9220 CD6CBB CALL EFFWIN ;Effacement Fenetre
25 9223 210080 LD HL,8000H
26 9226 220C92 LD (ADCUR),HL ;à debut verbes
27 9229 211701 LD HL,117H
28 922C CD75BB CALL SETCUR
29 922F 2A0C92 LD HL,(ADCUR)
30 9232 CDFE92 CALL AF251 ;Affichage ligne 1

```

```

31 9235 2A0C92      LD  HL,(ADCOUR)
32 9238 23          INC  HL
33 9239 220C92      LD  (ADCOUR),HL
34 923C 220F92      LD  (SS),HL
35 923F 211801      LD  HL,118H
36 9242 CD75BB      CALL SETCUR
37 9245 2A0C92      LD  HL,(ADCOUR)
38 9248 CDFE92      CALL AF251          ;Affichage ligne 2
39 924B 2A0C92      LD  HL,(ADCOUR)
40 924E 23          INC  HL
41 924F 220C92      LD  (ADCOUR),HL
42 9252 211901      LD  HL,119H
43 9255 CD75BB      CALL SETCUR
44 9258 2A0C92      LD  HL,(ADCOUR)
45 925B CDFE92      CALL AF251          ;Affichage ligne 3
46                  ;
47 925E 3E00        LD  A,0
48 9260 320E92      LD  (L1),A
49 9263 2600        LD  H,0
50 9265 1627        LD  D,39
51 9267 2E16        LD  L,22
52 9269 1E19        LD  E,25
53 926B D5          PUSH DE
54 926C E5          PUSH HL
55 926D 2A0F92      LD  HL,(SS)
56 9270 220C92      LD  (ADCOUR),HL
57                  AA1:  EQU  *
58 9273 CD06BB      CALL WAITCH          ;Attente clavier
59 9276 FE08        CP   8
60 9278 2865        JR   Z,AA5          ;CTRL H (Fin du HELP)

```

```

61 927A FEF0          CP    OF0H
62 927C 2B06          JR     Z,AA2          ; ->
63 927E FEF1          CP    OF1H
64 9280 2B2D          JR     Z,AA3          ; <-
65 9282 1BEF          JR     AA1          ;What ?
66                    ;
67                    AA2:    EQU    $          ;Gestion touche UP
68 9284 3A0E92        LD     A,(L1)
69 9287 FE0A          CP     10          ;Nombre max de mots
70 9289 2BEB          JR     Z,AA1
71 928B 3C           INC     A
72 928C 320E92        LD     (L1),A
73 928F 0601          LD     B,1
74 9291 E1           POP     HL
75 9292 D1           POP     DE
76 9293 3E00          LD     A,0
77 9295 D5           PUSH    DE
78 9296 E5           PUSH    HL
79 9297 CB30BC        CALL    ROLL
80 929A CDE592        CALL    VESU
81 929D CDE592        CALL    VESU
82 92A0 CDF592        CALL    AFVE25
83 92A3 CD1C93        CALL    VEPR
84 92A6 CD1C93        CALL    VEPR
85 92A9 23           INC     HL
86 92AA 220C92        LD     (ADCOUR),HL
87 92AD 1BC4          JR     AA1
88                    ;
89                    AA3:    EQU    $          ;Gestion touche DOWN
90 92AF 3A0E92        LD     A,(L1)

```

91	92B2	FE00	CP	0	
92	92B4	28BD	JR	Z,AA1	
93	92B6	3D	DEC	A	
94	92B7	320E92	LD	(L1),A	
95	92BA	0600	LD	B,0	
96	92BC	E1	POP	HL	
97	92BD	D1	POP	DE	
98	92BE	3E00	LD	A,0	
99	92C0	D5	PUSH	DE	
100	92C1	E5	PUSH	HL	
101	92C2	CD50BC	CALL	ROLL	
102	92C5	2A0C92	LD	HL,(ADCOUR)	
103	92C8	2B	DEC	HL	
104	92C9	220C92	LD	(ADCOUR),HL	
105	92CC	CD1C93	CALL	VEPR	
106	92CF	CD1C93	CALL	VEPR	
107	92D2	23	INC	HL	
108	92D3	220C92	LD	(ADCOUR),HL	
109	92D6	CD1193	CALL	AFVE23	
110	92D9	23	INC	HL	
111	92DA	220C92	LD	(ADCOUR),HL	
112	92DD	1894	JR	AA1	
113		,			
114		AA5:	EQU	*	;Bastion touche CTRL+H
115	92DF	D1	POP	DE	
116	92E0	E1	POP	HL	
117	92E1	CD6CBB	CALL	EFFWIN	
118	92E4	C9	RET		;Retour au prog appelant
119		,			
120		VESU:	EQU	*	;Verbe suivant

```

121 92E5 2A0C92      LD    HL,(ADCOUR)
122                  VESU1: EQU  $
123 92E8 7E          LD    A,(HL)
124 92E9 FEFF        CP    OFFH
125 92EB 2B03        JR    Z,VESU2
126 92ED 23          INC   HL
127 92EE 1BF8        JR    VESU1
128                  VESU2: EQU  $
129 92F0 23          INC   HL
130 92F1 220C92      LD    (ADCOUR),HL
131 92F4 C9          RET
132                  ;
133                  AFVE25: EQU  $                ;Affichage ligne 25
134 92F5 210301      LD    HL,103H
135 92F8 CD75BB      CALL SETCUR
136 92FB 2A0C92      LD    HL,(ADCOUR)
137                  AF251: EQU  $
138 92FE 7E          LD    A,(HL)
139 92FF FEFF        CP    OFFH
140 9301 2B0A        JR    Z,AF252
141 9303 FE20        CP    32                ;Si <32 (blanc)
142 9305 3B03        JR    C,AFBU            ;Caract. non affiche
143 9307 CD5ABB      CALL TXTOUT            ;Aff. d'un caractere
144                  AFSU: EQU  $
145 930A 23          INC   HL
146 930B 1BF1        JR    AF251
147                  AF252: EQU  $
148 930D 220C92      LD    (ADCOUR),HL
149 9310 C9          RET
150                  ;

```

```

151          AFVE23: EQU $ ;Affichage ligne 23
152 9311 210101 LD HL,101H
153 9314 CD75BB CALL SETCUR
154 9317 2A0C92 LD HL,(ADCOUR)
155 931A 18E2 JR AF251
156          ;
157          VEPR: EQU $ ;Verbe precedent
158 931C 2A0C92 LD HL,(ADCOUR)
159 931F 2B DEC HL
160          VEPR1: EQU $
161 9320 7E LD A,(HL)
162 9321 FEFF CP OFFH
163 9323 2803 JR Z,VEPR2
164 9325 2B DEC HL
165 9326 18F8 JR VEPR1
166          VEPR2: EQU $
167 9328 220C92 LD (ADCOUR),HL
168 932B C9 RET
169          END

```

- Lignes 19 à 21 : Définition de la fenêtre.
- Ligne 24 : Effacement de la fenêtre.
- Lignes 25 à 56 : Affichage des trois premières lignes.
- Ligne 58 : Attente d'une action au clavier.
- Ligne 60 : Ctrl H.
- Ligne 62 : Up.
- Ligne 64 : Down.
- Lignes 67 à 87 : Gestion du Up.
- Lignes 89 à 112 : Gestion du Down.
- Lignes 114 à 118 : Gestion du Ctrl H.
- Lignes 120 à 131 : Modification du pointeur de mot clé DOWN.
- Lignes 134 à 149 : Affichage en ligne 25.
- Lignes 151 à 155 : Affichage en ligne 23.
- Lignes 157 à 168 : Modification du pointeur de mot clé UP.

Les données affichées par le **HELP** vont être implantées par défaut en &H8000. Cette adresse peut être modifiée ligne 25. Le premier octet

(d'adresse &H8000 dans notre cas) doit contenir la valeur zéro. Les mots à afficher doivent être séparés par le caractère séparateur &HFF :

00 Do1 FF Do2 FF ... DoN FF

où Do1 à DoN représentent les mots clés à afficher.

Enfin, le nombre de scrolling (c'est-à-dire le nombre de données moins trois) doit être initialisé ligne 69. Il est égal à 10 dans notre exemple.

Il est également possible de modifier l'adresse d'implantation et le nombre de scrolling sans modifier le code source du listing Assembleur en utilisant trois instructions Basic Poke :

```
POKE &H9224,LSB
POKE &H9225,MSB
POKE &H9288,SC
```

où LSB est le poids faible de l'adresse d'implantation des mots clés,
MSB est le poids fort de l'adresse d'implantation des mots clés,
SC est le nombre de scrolling. Ce nombre est égal au nombre de mots clés moins trois : par exemple 4 pour 7 mots clés.

Le programme Assembleur qui vient d'être décrit peut également être exécuté en utilisant le chargeur Basic suivant :

```
1000 FOR I=&9211 TO &932B
1010 READ A$
1020 A=VAL("&" + A$)
1030 POKE I,A
1040 NEXT I
1060 -----
1070 DATA 21,16,0,11,18,27,CD,66,BB,CD,6C,BB,CD,84,BB,CD
1080 DATA 6C,BB,21,0,80,22,C,92,21,17,1,CD,75,BB,2A,C
1090 DATA 92,CD,FE,92,2A,C,92,23,22,C,92,22,F,92,21,18
1100 DATA 1,CD,75,BB,2A,C,92,CD,FE,92,2A,C,92,23,22,C
1110 DATA 92,21,19,1,CD,75,BB,2A,C,92,CD,FE,92,3E,0,32
1120 DATA E,92,26,0,16,27,2E,16,1E,19,D5,E5,2A,F,92,22
1130 DATA C,92,CD,6,BB,FE,8,28,65,FE,F0,28,6,FE,F1,28
1140 DATA 2D,18,EF,3A,E,92,FE,A,28,E8,3C,32,E,92,6,1
1150 DATA E1,D1,3E,0,D5,E5,CD,50,BC,CD,E5,92,CD,E5,92,CD
1160 DATA F5,92,CD,1C,93,CD,1C,93,23,22,C,92,18,C4,3A,E
1170 DATA 92,FE,0,28,BD,3D,32,E,92,6,0,E1,D1,3E,0,D5
1180 DATA E5,CD,50,BC,2A,C,92,2B,22,C,92,CD,1C,93,CD,1C
1190 DATA 93,23,22,C,92,CD,11,93,23,22,C,92,18,94,D1,E1
1200 DATA CD,6C,BB,C9,2A,C,92,7E,FE,FF,28,3,23,18,F8,23
1210 DATA 22,C,92,C9,21,3,1,CD,75,BB,2A,C,92,7E,FE,FF
1220 DATA 28,A,FE,20,38,3,CD,5A,BB,23,18,F1,22,C,92,C9
1230 DATA 21,1,1,CD,75,BB,2A,C,92,18,E2,2A,C,92,2B,7E
1240 DATA FE,FF,28,3,2B,18,F8,22,C,92,C9,0,0,0,0,0
```

Les données de checksum de ce programme sont les suivantes :

49 F8 9B 42 65 29 F9 40 E2 8C 55 DC 2E 88 F4 28 58 F0

Reportez-vous à la Partie 9, chapitre 8.4, pour vérifier la cohérence des données entrées dans les lignes de DATA du chargeur.

9/6.4

Création de jeux d'aventures

Le programme présenté ici permet de créer de façon simple des jeux d'aventures dans lesquels il y aura plusieurs tableaux (aucune limite quant à leur nombre, si ce n'est la place mémoire disponible), et où le joueur pourra dialoguer avec l'ordinateur pour ramasser des objets, avancer dans une direction ou effectuer une autre action quelconque (les diverses actions possibles sont définies dans ce programme).

Une analyse syntaxique de mots-clés du type « VERBE SUJET » (voir Partie 9, chapitre 6.1) est faite et permet de définir l'action à effectuer en fonction du couple de mots-clés identifiés.

Les points de vie sont incrémentés ou décrémentés en fonction des actions du joueur.

Une musique peut éventuellement être exécutée sous interruptions en parallèle avec le jeu (voir Partie 6, chapitre 5.1).

Enfin, une fenêtre déroulante peut permettre au joueur de visualiser quels sont les mots-clés compris par la machine (Voir Partie 9, chapitre 6.2).

Utilisation du programme

Dès le début du programme, un menu est affiché sur l'écran. Il comporte les options suivantes :

- charger une aventure existante ;
- modifier ou créer le nom des salles de jeu ;
- modifier ou créer les mots-clés (Verbes et Sujets) du jeu ;
- modifier ou créer les réponses que donnera l'ordinateur suite aux actions du joueur ;
- définir les actions découlant de l'association SALLE-VERBE-SUJET décryptée par l'analyseur de syntaxe ;
- créer le fichier Aventure exécutable par le programme d'exécution d'aventure défini à la Partie 9, chapitre 6.4.

Lors de la définition des actions, 5^e option du menu ci-dessus, il vous faudra entrer :

- le numéro de la salle ;
- le numéro du verbe ;
- le numéro du sujet ;
- le code action correspondant.

Ce code peut comporter une association quelconque des quatre commandes suivantes :

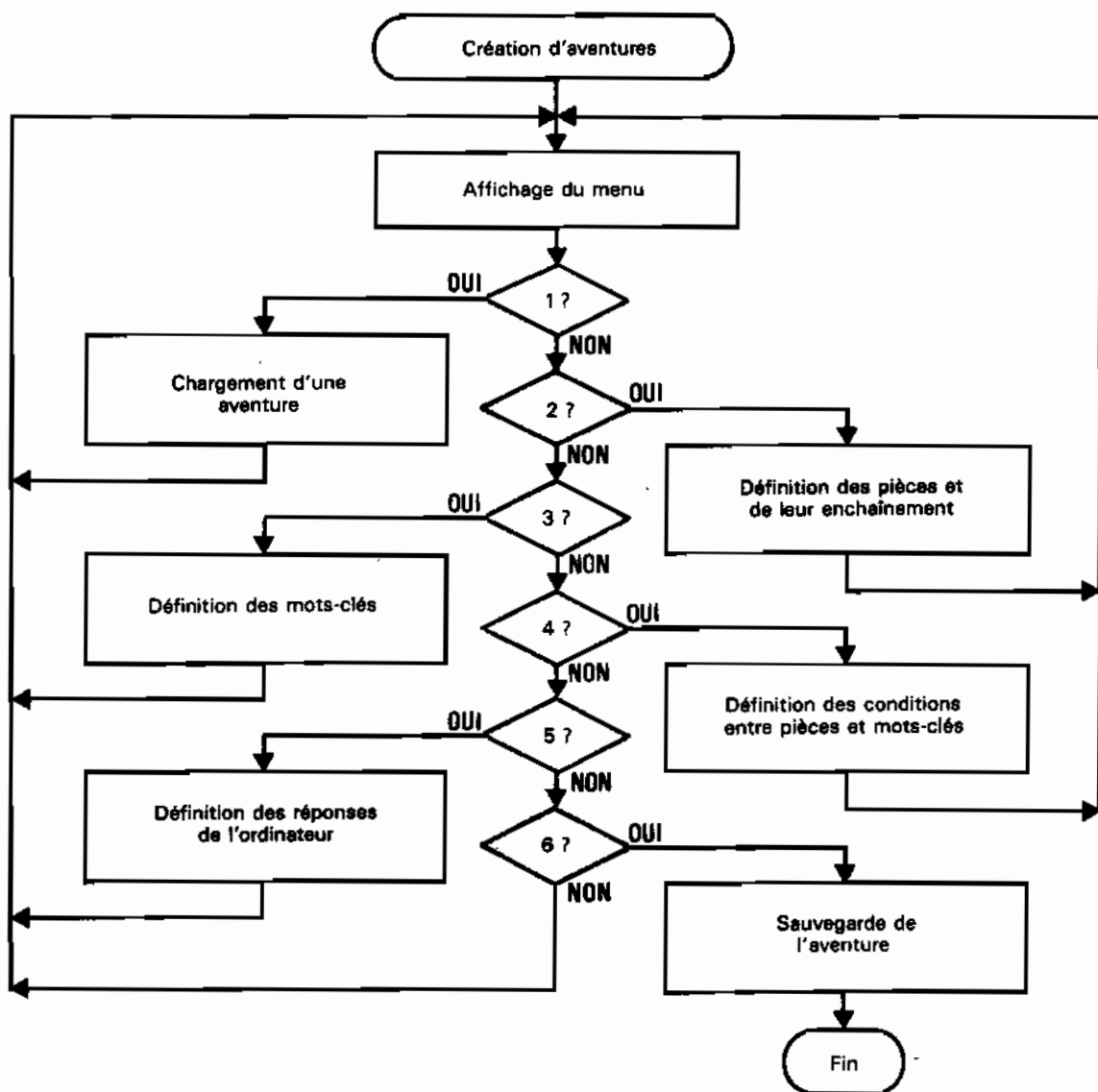
- changement de salle (*Exemple* : C4 fait passer en salle 4) ;
- réponse de l'ordinateur (*Exemple* : R2 fait afficher la réponse numéro 2) ;
- modification des points du joueur (*Exemple* : P10 fait augmenter le nombre de points du joueur de 10, et P-20 fait diminuer de 20 ce même nombre) ;
- fin du jeu. La partie est terminée (F).

Remarques :

- maximum de 80 caractères,
- écrits de la façon suivante : Lettre,O

Par exemple, l'action changement de salle, passage en salle 4 et fin de jeu sera codée : C4ROPOF.

La logique du programme est la suivante :



Le listing du programme est le suivant :

```

1000 MEMORY &5F00
1010 REM CREATION D'AVENTURES DE TYPE 1
1020 DIM LAB$(100,4),N(4)
1030 '=====
1040 REM Menu
1050 '
1060 INK 0,0:BORDER 0:INK 1,10:INK 2,3:INK 3,1,3:MODE 1:PEN
3
1070 PRINT"      CREATION D'AVENTURES TYPE 1":PEN 1
1080 LOCATE 13,8:PRINT"1- Chargement"
1090 LOCATE 13,10:PRINT"2- Salles"
1100 LOCATE 13,12:PRINT"3- Mots-cles"
1110 LOCATE 13,14:PRINT"4- Reponses"
1120 LOCATE 13,16:PRINT"5- Conditions"
1130 LOCATE 13,18:PRINT"6- Fin"
1140 A$=INKEY$:IF A$="" THEN 1140
1150 A=ASC(A$)
1160 IF A>54 OR A<49 THEN SOUND 1,100,30 :GOTO 1140
1170 ON A-48 GOSUB 2000,3000,4000,5000,6000,7000
1180 GOTO 1060 'Boucle sur menu
2000 '=====
2010 'CHARGEMENT D'UNE AVENTURE
2020 '
2030 CLS:PEN 3:PRINT" CHARGEMENT D'UNE AVENTURE":PEN 1
2040 WINDOW 1,40,5,25
2050 INPUT"Nom de l'aventure ";N$
2060 LOAD N$,&6000 'Chargement
2070 AD=&6000 'Adresse de depart du fichier aventure
2080 SD=PEEK(AD):AD=AD+2 'Salle de depart
2090 IF PEEK(AD)=&FF THEN T$="":AD=AD+1 ELSE GOSUB 2320:T$=X
$
2100 IF PEEK(AD)=&FF THEN MUS$="":AD=AD+1 ELSE GOSUB 2320:MU
S$=X$
2110 HELP=PEEK(AD):AD=AD+2
2120 FOR I=1 TO 4
2130   N(I)=PEEK(AD):AD=AD+1
2140   FOR J=1 TO N(I)
2150     GOSUB 2320:LAB$(J,I)=X$
2160   NEXT J
2170 NEXT I
2180 NR=PEEK(AD)
2190 FOR I=1 TO NR
2200   FOR J=1 TO 3
2210     AD=AD+1:CAR(I,J)=PEEK(AD)
2220   NEXT J
2230   X$="":AD=AD+1:P=PEEK(AD):IF P<>0 THEN X$=X$+"C"+RIGHT
$(STR$(P),LEN(STR$(P))-1)
2240   AD=AD+1:P=PEEK(AD):IF P<>0 THEN X$=X$+"R"+RIGHT$(STR$
(P),LEN(STR$(P))-1)

```

```

2250 AD=AD+1:P=PEEK(AD):IF P=0 THEN 2280
2260 X$=X$+"P":LL=LEN(STR$(P-128))
2270 IF P-128<0 THEN X$=X$+RIGHT$(STR$(P-128),LL) ELSE X$=
X$+RIGHT$(STR$(P-128),LL-1)
2280 AD=AD+1:P=PEEK(AD):IF P<>0 THEN X$=X$+"F"
2290 C$(I)=X$
2300 NEXT I
2310 RETURN
2320 'Conversion ASCII -> Chaîne de caractères
2330 X$=""
2340 IF PEEK(AD)<>255 THEN X$=X$+CHR$(PEEK(AD)):AD=AD+1:GOTO
2340 ELSE AD=AD+1:RETURN
3000 '=====
3010 'ENCHAÎNEMENT DES PIÈCES
3020 '
3030 CLS:PEN 3:PRINT " ENCHAÎNEMENT DES PIÈCES":PEN 1:CA
=1
3040 WINDOW 1,40,5,25
3050 IF CA=1 THEN PR$="Donnez le nom des ECRANS"
3060 IF CA=2 THEN PR$="Donnez le nom des VERBES"
3070 IF CA=3 THEN PR$="Donnez le nom des SUJETS"
3080 IF CA=4 THEN PR$="Entrez les réponses"
3090 IF N(CA)<>0 THEN 3110 'Ecrans déjà existant
3100 PRINT PR$;"et terminez la saisie par ENTER.":PRINT:GOTO
3530
3110 PEN 2:IF CA=1 THEN PRINT"ECRANS";
3120 IF CA=2 THEN PRINT"VERBES";
3130 IF CA=3 THEN PRINT"SUJETS";
3140 IF CA=4 THEN PRINT"REPONSES";
3150 PRINT" déjà en mémoire":PEN 1
3160 PRINT"Voulez-vous 1)Modifier"
3170 PRINT" 2)Ajouter"
3180 PRINT" 3)Supprimer des labels "
3190 A$=INKEY$:IF A$="" THEN 3190
3200 R=ASC(A$)-48
3210 IF ASC(A$)=13 THEN RETURN 'Retour au menu
3220 IF R>3 OR R<1 THEN SOUND 1,100,30:GOTO 3190
3230 ON R GOTO 3240,3330,3410
3240 'Modification de labels
3250 '
3260 PRINT:INPUT"L)iste,NO DE LABEL OU ENTER ";A$:PRINT
3270 IF LEN(A$)=0 THEN CLS:GOTO 3050
3280 IF A$="L" OR A$="1" THEN FOR I=1 TO N(CA):PRINT I;LAB$(
I,CA):NEXT I:GOTO 3260
3290 A=VAL(A$):IF A=0 THEN SOUND 1,100,30:GOTO 3260
3300 IF A>N(CA) THEN SOUND 1,100,10:GOTO 3260
3310 PRINT:PRINT A;:INPUT A$:LAB$(A,CA)=A$
3320 GOTO 3260
3330 'Ajout de labels
3340 '
3350 PRINT:INPUT"L)iste OU ENTER ";A$:PRINT

```

```

3360 IF A$="L" OR A$="I" THEN FOR I=1 TO N(CA):PRINT I;LAB$(
I,CA):NEXT:GOTO 3350
3370 IF LEN(A$)<>0 THEN SOUND 1,100,30:GOTO 3350
3380 N(CA)=N(CA)+1:PRINT N(CA);:INPUT A$:LAB$(N(CA),CA)=A$
3390 IF LEN(A$)=0 THEN N(CA)=N(CA)-1:CLS:GOTO 3050
3400 GOTO 3380
3410 'Suppression de labels
3420 '
3430 PRINT:INPUT"L)iste,No label ou ENTER ";A$:PRINT
3440 IF LEN(A$)=0 THEN CLS:GOTO 3050
3450 IF A$="L" OR A$="I" THEN FOR I=1 TO N(CA):PRINT I;LAB$(
I,CA):NEXT:GOTO 3430
3460 A=VAL(A$):IF A=0 THEN SOUND 1,100,10:GOTO 3410
3470 IF A>N(CA) THEN SOUND 1,100,10:GOTO 3410
3480 PRINT A;LAB$(A,CA);" ";
3490 INPUT"OK (O/N) ";R$:R$=UPPER$(R$)
3500 IF R$<>"O" AND R$<>"N" THEN SOUND 1,100,10:GOTO 3490
3510 IF R$="O" THEN FOR I=A TO N(CA):LAB$(I,CA)=LAB$(I+1,CA)
:NEXT:N(CA)=N(CA)-1:GOTO 3260
3520 GOTO 3430
3530 'Saisie de labels
3540 '
3550 N(CA)=0 'initialisation
3560 N(CA)=N(CA)+1:PRINT N(CA);:INPUT A$:LAB$(N(CA),CA)=A$
3570 IF LEN(A$)=0 THEN N(CA)=N(CA)-1:RETURN
3580 GOTO 3560
4000 '=====
4010 'DEFINITION DES MOTS-CLES
4020 '
4030 CLS:PEN 3:PRINT"          MOTS CLE":PEN 1
4040 WINDOW 1,40,5,25
4050 PRINT"V)erbes ou S)ujets "
4060 A$=INKEY$:IF A$="" THEN 4060
4070 A$=UPPER$(A$)
4080 IF A$<>"V" AND A$<>"S" THEN SOUND 1,100,30:GOTO 4060
4090 IF A$="V" THEN CA=2 ELSE CA=3
4100 GOTO 3040
5000 '=====
5010 'CONDITIONS ENTREES SALLES ORDRES ET MOTS-CLES
5020 '
5030 CLS:PEN 3:PRINT"  CONDITIONS LIANT SALLES ET MOTS-CLES"
:PEN 1
5040 WINDOW 1,40,5,25
5050 CLS:PRINT:PRINT"Voulez-vous 1)Lister les regles"
5060 PRINT"          2)Creer d'autres regles"
5070 PRINT"          3)Modifier une regle"
5080 A$=INKEY$:IF A$="" THEN 5080
5090 IF ASC(A$)=13 THEN RETURN
5100 A=VAL(A$):IF A<1 OR A>3 THEN SOUND 1,100,30:GOTO 5080
5110 ON A GOTO 5120,5230,5340
5120 'Listage des regles existantes
5130 IF NR=0 THEN PEN 3:LOCATE 10,10:PRINT"Aucune regle en m
emoire":PEN 1

```

```

5140 IF NR=0 THEN A$=INKEY$:IF A$="" THEN 5140
5150 FOR I=1 TO NR
5160   CLS:PRINT"Salle : ";LAB$(CAR(I,1),1)
5170   PRINT"Verbe : ";LAB$(CAR(I,2),2)
5180   PRINT"Sujet : ";LAB$(CAR(I,3),3)
5190   PRINT"Action: ";C$(I)
5200   A$=INKEY$:IF A$="" THEN 5200
5210 NEXT I
5220 GOTO 5050
5230 'Creation de regles
5240 NR=NR+1 'Nombre de relations+1
5250 LOCATE 1,7:INPUT "Salle ";CAR(NR,1)
5260 IF CAR(NR,1)=0 THEN NR=NR-1:GOTO 5050 'Sortie
5270 INPUT "Verbe ";CAR(NR,2)
5280 INPUT "Sujet ";CAR(NR,3)
5290 PRINT:PRINT:PRINT"(C=Changt salle,R=Reponse,P=Points)"
5300 PRINT"(F=Fin de partie)"
5310 PRINT"(Ex:C3P-10 =>Passage salle 3 et"
5320 PRINT"          nombre de points-10 )"
5330 LOCATE 1,10:INPUT"Action ";C$(NR):GOTO 5050
5340 'Modification d'une regle
5350 INPUT "No de la regle ";N
5351 IF N=0 THEN 5050
5360 IF N>NR THEN PRINT"Regle inexistante":GOTO 5200 'Retour
    au menu
5370 LOCATE 1,8:PRINT"Salle ";CAR(N,1)
5380 PRINT"Verbe ";CAR(N,2)
5390 PRINT"Sujet ";CAR(N,3)
5400 PRINT"Action ";C$(N)
5410 LOCATE 12,8 :INPUT CAR(N,1)
5420 LOCATE 12,9 :INPUT CAR(N,2)
5430 LOCATE 12,10:INPUT CAR(N,3)
5440 LOCATE 18,11:INPUT C$(N)
5450 GOTO 5050
6000 '=====
6010 'REPONSES DE LA MACHINE
6020 '
6030 CLS:PEN 3:PRINT"          REPONSES DE L'ORDINATEUR"
6040 CA=4:PEN 1:GOTO 3040
7000 '=====
7010 'SORTIE DU GENERATEUR D'AVENTURES
7020 '
7030 CLS:PEN 3:PRINT"          SAUVEGARDE DE L'AVENTURE":PEN 1

7040 WINDOW 1,40,5,25
7050 CLS:PRINT:PRINT"Avant de sauvegarder le fichier aventur
erepondez aux questions suivantes:":PRINT
7060 PEN 2:PRINT"Salle de depart (1 a";N(1);") ";:PEN 1
7070 IF SD<>0 THEN PRINT SD
7080 LOCATE 25,5:INPUT SD$:PRINT
7090 IF SD$="" AND SD=0 THEN RETURN
7100 IF SD$="" THEN 7120 ELSE SD=VAL(SD$)

```

```

7110 IF SD<1 OR SD>N(1) THEN SOUND 1,100,30:GOTO 7060
7120 IF T$<>"" THEN CLS:PEN 2:PRINT"Texte en debut d'aventur
e:":PEN 1:PRINT T$:PRINT:INPUT X$:IF X$="" THEN 7190 ELSE T$
=X$:GOTO 7190
7130 PEN 2:CLS:PRINT"Texte en debut d'aventure (O/N) ":PEN
1:INPUT R$:R$=UPPER$(R$):PRINT
7140 IF R$="" THEN RETURN
7150 IF R$<>"O" AND R$<>"N" THEN SOUND 1,100,30:GOTO 7130
7160 IF R$="N" THEN 7200
7170 PEN 2:PRINT"Entrez le texte:":PEN 1
7180 INPUT T$:PRINT
7190 IF MUS$<>"" THEN CLS:PEN 2:PRINT"Nom du fichier musical
:":PEN 1:PRINT MUS$:PRINT:INPUT X$:IF X$="" THEN 7250 ELSE M
US$=X$:GOTO 7250
7200 PEN 2:CLS:PRINT"Musique pendant l'aventure (O/N) ":PEN
1:INPUT R$:R$=UPPER$(R$):PRINT
7210 IF R$="" THEN RETURN
7220 IF R$<>"O" AND R$<>"N" THEN SOUND 1,100,30:GOTO 7200
7230 IF R$="N" THEN 7250 'Pas de musique
7240 PEN 2:PRINT"Nom du fichier musical ":PEN 1:INPUT MUS$:
PRINT
7250 PEN 2:CLS:PRINT"HELP(CTRL H) disponible (O/N) ":PEN 1:
INPUT R$:R$=UPPER$(R$)
7260 IF R$="" THEN RETURN
7270 IF R$<>"O" AND R$<>"N" THEN SOUND 1,100,30:GOTO 7250
7280 IF R$="O" THEN HELP=1 ELSE HELP=0
7290 'Constitution du fichier aventure
7300 '
7310 AD=&6000 'Debut du fichier aventure
7311 FOR I=&6000 TO &6A00:POKE I,0:NEXT 'RAZ fichier aventur
e precedent
7320 POKE AD,SD:AD=AD+1:POKE AD,&FF:AD=AD+1 'Salle de depart
7330 IF T$="" THEN POKE AD,&FF ELSE FOR I=1 TO LEN(T$):POKE
AD,ASC(MID$(T$,I,1)):AD=AD+1:NEXT:POKE AD,&FF 'Texte de debu
t d'aventure
7340 AD=AD+1:IF MUS$="" THEN POKE AD,&FF ELSE FOR I=1 TO LEN
(MUS$):POKE AD,ASC(MID$(MUS$,I,1)):AD=AD+1:NEXT:POKE AD,&FF
'Fichier musical
7350 AD=AD+1:POKE AD,HELP:AD=AD+1:POKE AD,&FF 'Fonction HELP

7360 FOR J=1 TO 4
7370   AD=AD+1:POKE AD,N(J)
7380   FOR I=1 TO N(J)
7390     FOR K=1 TO LEN(LAB$(I,J)):AD=AD+1:POKE AD,ASC(MID$(
LAB$(I,J),K,1)):NEXT
7400   AD=AD+1:POKE AD,&FF
7410   NEXT I
7420 NEXT J
7430 'Extraction des conditions et actions
7440 AD=AD+1:POKE AD,NR 'Nombre de regles
7450 FOR I=1 TO NR
7460   FOR F=1 TO 3
7470     AD=AD+1:POKE AD,CAR(I,F)
7480   NEXT F

```

```

7490 A=99:AD=AD+1:B=LEN(C$(I))
7500 FOR J=1 TO B
7510 IF UPPER$(MID$(C$(I),J,1))="C" THEN A=VAL(RIGHT$(C$(I),B-J)):POKE AD,A 'Changements de salle
7520 NEXT J
7530 IF AD=99 THEN POKE AD,&FF
7540 A=99:AD=AD+1
7550 FOR J=1 TO LEN(C$(I))
7560 IF UPPER$(MID$(C$(I),J,1))="R" THEN A=VAL(RIGHT$(C$(I),B-J)):POKE AD,A 'Reponse
7570 NEXT J
7580 IF AD=99 THEN POKE AD,&FF
7590 A=99:AD=AD+1
7600 FOR J=1 TO LEN(C$(I))
7610 IF UPPER$(MID$(C$(I),J,1))="P" THEN A=VAL(RIGHT$(C$(I),B-J))+128:POKE AD,A 'Gestion des points
7620 NEXT J
7630 IF AD=99 THEN POKE AD,&FF
7640 A=99:AD=AD+1
7650 FOR J=1 TO LEN(C$(I))
7660 IF UPPER$(MID$(C$(I),J,1))="F" THEN POKE AD,1 'Fin de partie ou non
7670 NEXT J
7680 IF AD=99 THEN POKE AD,0
7690 NEXT I
7700 AD=AD+1:POKE AD,&FF 'Terminateur conditions
7710 '
7720 CLS:PRINT"Puis-je proceder a la sauvegarde"
7730 INPUT"de l'aventure (O/N) ";R$:R$=UPPER$(R$)
7740 IF R$<>"O" AND R$<>"N" THEN SOUND 1,100,30:GOTO 7720
7750 IF R$="N" THEN 1060
7760 PEN 2:PRINT:INPUT"Nom de l'aventure ";N$
7770 LL=AD-&6000+1
7780 SAVE N$,B,&6000,LL
7790 GOTO 1060

```

- Lignes 1060 à 1180 : Menu.
- Lignes 2030 à 2340 : Chargement d'une aventure en &H6000.
- Lignes 3000 à 3580 : Définition des salles de jeu.
- Lignes 4030 à 4100 : Définition des mots-clés.
- Lignes 5030 à 5450 : Conditions liant salles et mots-clés.
- Lignes 6030 à 6040 : Réponses de l'ordinateur.
- Lignes 7030 à 7790 : Fin d'utilisation du générateur d'aventures.
- Lignes 7030 à 7280 : Questions sur le début du jeu.
- Lignes 7290 à 7700 : Compilation de l'aventure.
- Lignes 7730 à 7780 : Sauvegarde de l'aventure.

9/7

Jeux d'Arcade

Ces jeux ont remplacé le « baby-foot » de nos jeunes années. Seul le flipper semble survivre au raz-de-marée de ces nouveaux venus.

Transformez votre Amstrad en une véritable console de jeu.

Utilisez, améliorez les programmes de ce chapitre et envoyez-nous vos œuvres originales.

9/7.1

Casse briques

Le classique jeu de café est repris dans ce programme. Un mur de briques de 4 briques d'épaisseur occupe la partie haute de l'écran. Une raquette peut être déplacée en utilisant les touches \ et Z. Vous disposez de 5 balles pour arriver à un score minimum de 1 300 points. Votre score est affiché en fin de partie.

```

1000 'Casse briques
1010 '=====
1020 GOSUB 1060 'Initialisation du jeu
1030 GOSUB 2150 'fin de la partie
1040 END
1050 '=====
1060 'Initialisation
1070 '=====
1080 MODE 1:ON BREAK GOSUB 2210
1090 FOR I=0 TO 8:READ A:POKE &9000+I,A:NEXT I
1100 DATA &CD,&60,&BB,&32,&10,&90,&C9
1110 SPEED KEY 1,1
1120 '-----
1130 'Definition des briques et de la raquette
1140 '-----
1150 SYMBOL AFTER 129
1160 SYMBOL 129,73,54,73,54,73,54,73,54
1170 SYMBOL 130,0,0,0,0,0,255,255,0
1180 '-----
1190 'Affichage des briques
1200 '-----
1210 CLS
1220 FOR I=1 TO 4
1230   FOR K=1 TO 38
1240     PRINT CHR$(129);
1250   NEXT K
1260   PRINT
1270 NEXT I
1280 '-----
1290 'Affichage de la raquette
1300 '-----
1310 XR=1:YR=20
1320 LOCATE XR,YR:PRINT CHR$(130);CHR$(130);CHR$(130)
1330 '
1340 XB=3:YB=19
1350 VX=INT(RND(1)*3)-1:VY=-1
1360 '-----
1370 'Debut de la partie
1380 '-----
1390 PRINT"                Balle 1":BA=1:PRINT
1400 PRINT"Appuyez sur une touche pour commencer"
1410 A$=INKEY$:IF A$="" THEN 1410
1420 LOCATE 1,23:PRINT SPACE$(38)
1430 PRINT"                SCORE : ";SC
1440 '-----
1450 'Deroulement du jeu
1460 '-----
1470 GOSUB 1500
1480 B$=INKEY$:IF B$<>"" THEN GOSUB 1850
1490 IF FIN=1 THEN RETURN ELSE 1470
1500 '-----
1510 'Action sur la balle
1520 '-----
1530 LOCATE XB,YB:PRINT" "
1540 IF (YB+VY)<=0 THEN VY=1:VX=INT(RND(1)*3)-1
1550 IF XB+VX<1 THEN VX=1:SOUND 1,100,10
1560 IF XB+VX=39 THEN VX=-1:SOUND 1,100,10
1570 LOCATE XB+VX,YB+VY:CALL &9000:A=PEEK(&9010):IF A=129 THEN GOSUB 1710
1580 IF A=130 THEN GOSUB 1650
1590 IF YB+VY>20 THEN GOSUB 2040
1600 IF FIN=1 THEN RETURN
1610 IF XB+VX<1 THEN VX=1:SOUND 1,100,10
1620 IF XB+VX=39 THEN VX=-1:SOUND 1,100,10

```

```

1630 XB=XB+VX:YB=YB+VY:LOCATE XB,YB:PRINT"o"
1640 RETURN
1650 '-----
1660 'Renvoi de la balle
1670 '-----
1680 VY=-1:VX=INT(RND(1)*3)-1
1690 SOUND 1,200,10
1700 RETURN
1710 '-----
1720 'Demolition brique
1730 '-----
1740 LOCATE XB+VX,YB+VY:PRINT" "
1750 SC=SC+10*(5-YB-VY):LOCATE 14,24:PRINT"SCORE : ";SC
1760 VY=1:VX=INT(RND(1)*3)-1
1770 IF SC>1300 THEN FIN=1
1780 SOUND 1,100,10
1790 RETURN
1800 '-----
1810 'Balle perdue
1820 '-----
1830 FIN=1
1840 RETURN
1850 '-----
1860 'Action du joueur
1870 '-----
1880 IF UPPER$(B$)="Z" THEN GOSUB 1910
1890 IF B$="c" THEN GOSUB 1970
1900 RETURN
1910 '-----
1920 'Deplacement de la raquette <-
1930 '-----
1940 IF XR=1 THEN RETURN
1950 LOCATE XR+1,YR:PRINT"  ":XR=XR-2:LOCATE XR,YR:PRINT CHR$(130);CHR$(130)
1960 RETURN
1970 '-----
1980 'Deplacement de la raquette ->
1990 '-----
2000 IF XR=37 THEN RETURN
2010 LOCATE XR,YR:PRINT"  ":XR=XR+2:LOCATE XR+1,YR:PRINT CHR$(130);CHR$(130)
2020 RETURN
2030 '-----
2040 REM Perte de la balle
2050 '-----
2060 BA=BA+1:IF BA=6 THEN FIN=1:GOTO 2130 ELSE FIN=0
2070 LOCATE 1,21:PRINT"      Balle";BA:PRINT
2080 LOCATE 1,23:PRINT"Appuyez sur une touche pour commencer"
2090 A$=INKEY$:IF A$="" THEN 2090
2100 LOCATE 1,23:PRINT SPACE$(38);
2110 XB=3:YB=19:VY=-1
2120 PRINT:PRINT:PRINT
2130 RETURN
2140 '-----
2150 REM Fin du jeu
2160 '-----
2170 CLS
2180 PRINT"Score";SC
2190 IF BA=1 OR BA=2 THEN PRINT "Bravo"
2200 SPEED KEY 10,1
2210 RETURN

```

Lignes 1000 à 1040 : Programme principal

Lignes 1050 à 1350 : Initialisation (définition des caractères graphiques, et affichage du terrain de jeu)

Lignes 1360 à 1430 : Début d'une partie

Lignes 1440 à 1490 : Déroulement d'une partie

Lignes 1500 à 1840 : Déplacement de la balle en fonction des obstacles rencontrés

Lignes 1850 à 2020 : Action du joueur

Lignes 2030 à 2130 : Prise en compte de la perte d'une balle

Lignes 2140 à 2210 : Fin du jeu

Un programme en assembleur est inséré dans le listing pour permettre de lire le code d'un caractère n'importe où sur l'écran. Pour cela, nous faisons appel à la macro du firmware « TXD RD CHAR » qui a son point d'entrée en &BB60. Consultez la Partie 4 Chap. 2.7 pour avoir plus de détails sur le fonctionnement de cette routine. Le code ASCII du caractère est placé en &9010, et lu en BASIC par l'ordre PEEK.

1	ORG 9000H	
2	LOAD 9000H	
3	9000 CD60BB	CALL 0BB60H ;Input char
4	9003 321090	LD (9010H),A ;Sauvegarde lecture
5	9006 C9	RET
6	END	

9/7.2

Bataille navale

Qui n'a jamais joué à la bataille navale ?

Voici une version informatisée de ce célèbre jeu de hasard dans laquelle vous jouez contre l'ordinateur.

Dans cette version simplifiée, chaque joueur dispose de six navires évoluant sur une grille carrée de neuf cases de côté.

Chaque navire occupe une case sur la grille.

	A	B	C	D	E	F	G	H	I
1				x					x
2		x							
3			x						
4									
5						x			
6				x					
7									
8									
9									

La grille de jeu avec les six navires positionnés.

Le nombre de torpilles à la disposition de chaque joueur est illimité. A chaque tour de jeu, une torpille est envoyée vers une case identifiée par ses coordonnées (lettre puis chiffre). Par exemple A4.

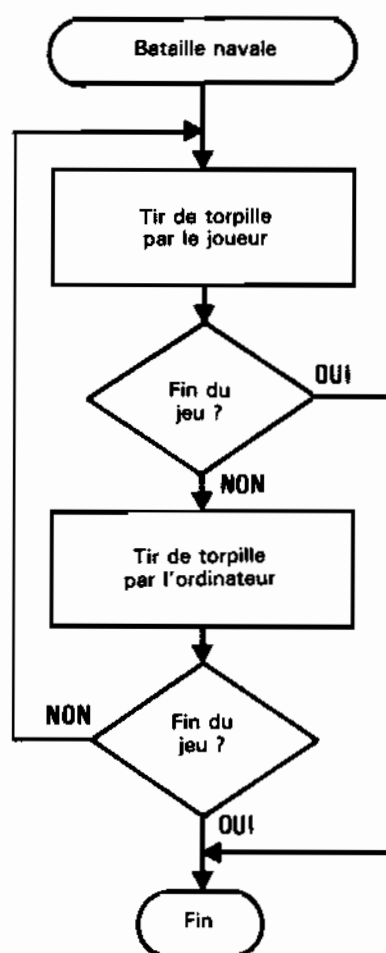
Avant de commencer une partie, le joueur doit dessiner sur papier sa grille de jeu comportant six navires.

Les tirs de torpilles du joueur sont visualisés en permanence sur la grille de jeu. La convention suivante est adoptée :

- o désigne un « loupé »,
- x désigne un « coulé ».

La partie se termine à l'avantage du joueur s'il arrive à couler les six navires de l'ordinateur avant que ce dernier n'ait fait l'opération inverse.

Le programme obéit à la logique de l'ordinogramme suivant :



Pour tous les programmeurs Basic qui désirent se mettre au Turbo-Pascal, nous avons développé ce programme dans les deux langages.

La version Basic du programme

```

1000 '=====
1010 ' BATAILLE NAVALE
1020 '=====
1030 'Programme principal
1040 '=====
1050 '
1060 GOSUB 1150 'Presentation
1070 GOSUB 1440 'Deroulement d'une partie
1080 GOSUB 2540 'Fin de la partie
1090 IF bis THEN 1070 'Nouvelle partie
1100 END
1110 '-----
1120 ' Presentation
1130 '-----
1140 '
1150 MODE 1
1160 PRINT"          Bataille navale"
1170 PRINT"          -----"
1180 PRINT
1190 PRINT"Dans ce jeu, vous devez decouvrir des"
1200 PRINT"navires ennemis qui sont positionnes"
1210 PRINT"sur une grille de 9 cases sur 9,et ce"
1220 PRINT"avant que l'ennemi ne decouvre les"
1230 PRINT"votres."
1240 PRINT
1250 PRINT"Chaque joueur possede 6 navires"
1260 PRINT"occupant chacun une position sur la"
1270 PRINT"grille."
1280 PRINT
1290 PRINT"Pour decouvrir les navires ennemis,"
1300 PRINT"vous disposez de torpilles que vous"
1310 PRINT"pouvez lancer sur une case quelconque"
1320 PRINT"de la grille."
1330 PRINT
1340 INK 2,2,3:PEN 2
1350 PRINT"Pour commencer, appuyez sur une touche"
1360 PRINT"quelconque..."
1370 PEN 1
1380 a$=INKEY$:IF a$="" THEN 1380
1390 RETURN
1400 '-----
1410 ' Deroulement d'une partie
1420 '-----
1430 '
1440 GOSUB 1510 'Initialisation
1450 GOSUB 1970 'Partie
1460 RETURN

```

```
1470 '-----
1480 ' Initialisation d'une partie
1490 '-----
1500 '
1510 CLS
1520 PRINT"Dessinez une grille carree de 9 cases"
1530 PRINT"de cote. Placez six navires d'une case"
1540 PRINT"sur cette grille."
1550 PRINT
1560 PEN 2
1570 PRINT"Tapez sur une touche pour commencer"
1580 PRINT"a jouer..."
1590 PEN 1
1600 '
1610 ' Initialisation de l'ordinateur
1620 '
1630 FOR i=1 TO 6
1640   x=INT(RND(1)*9)+1
1650   y=INT(RND(1)*9)+1
1660   IF o(x,y)=0 THEN o(x,y)=1 ELSE i=i-1
1670 NEXT i
1680 '
1690 FOR i=1 TO 9
1700   FOR j=1 TO 9
1710     essai(i,j)=0
1720   NEXT j
1730 NEXT i
1740 '
1750 EF$=SPACE$(120)
1760 '
1770 a$=INKEY$:IF a$="" THEN 1770
1780 '
1790 ' Affichage de la grille de tir
1800 '
1810 CLS
1820 PRINT"          Bataille navale"
1830 PRINT"          -----"
1840 LOCATE 1,5
1850 PRINT"Votre grille de tir:"
1860 PRINT
1870 PRINT"          A B C D E F G H I"
1880 PRINT
1890 FOR i=1 TO 9
1900   PRINT"    ";i
1910 NEXT i
1920 RETURN
1930 '-----
1940 ' Deroulement d'une partie
1950 '-----
1960 '
1970 GOSUB 2030 'Jeu de l'utilisateur
```



```

1980 IF fin THEN RETURN
1990 GOSUB 2380 'Jeu de l'ordinateur
2000 IF fin THEN RETURN
2010 GOTO 1970
2020 '-----
2030 '  Jeu de l'utilisateur
2040 '-----
2050 '
2060 LOCATE 1,20:PRINT EF$
2070 LOCATE 1,20:INPUT"Votre jeu ";j$
2080 j$=UPPER$(j$)
2090 a=ASC(LEFT$(j$,1))-64
2100 b=ASC(RIGHT$(j$,1))-48
2110 IF o(b,a)=0 THEN GOSUB 2190 'A l'eau
2120 IF o(b,a)=1 THEN GOSUB 2270 'Touche-coule
2130 LOCATE 1,22:PEN 2
2140 PRINT"appuyez sur une touche..."
2150 PEN 1
2160 a$=INKEY$:IF a$="" THEN 2160
2170 RETURN
2180 '-----
2190 'A l'eau
2200 '-----
2210 '
2220 LOCATE 1,21:PRINT"A l'eau"
2230 LOCATE 2*a+6,b+8
2240 PRINT"o"
2250 RETURN
2260 '-----
2270 'Touche-coule
2280 '-----
2290 '
2300 LOCATE 1,21:PRINT"Coule !"
2310 LOCATE 2*a+6,b+8
2320 PRINT"x"
2330 o(b,a)=2 'ce coup ne peut etre rejoue
2340 cj=cj+1
2350 IF cj=6 THEN fin=-1
2360 RETURN
2370 '-----
2380 '  Jeu de l'ordinateur
2390 '-----
2400 '
2410 x=INT(RND(1)*9)+1
2420 y=INT(RND(1)*9)+1
2430 IF essai(x,y)<>0 THEN 2380
2440 essai(x,y)=1
2450 a$=CHR$(y+64)+CHR$(x+48)
2460 LOCATE 1,20:PRINT ef$
2470 LOCATE 1,20:PRINT"L'ordinateur joue en ";a$ ,

```

```
2480 PRINT"Entrez le resultat du tir."
2490 INPUT"(1 si a l'eau, 2 si coule):";a
2500 IF a=2 THEN co=co+1
2510 IF co=6 THEN fin=-1
2520 RETURN
2530 '-----
2540 'Fin de la partie
2550 '-----
2560 '
2570 CLS
2580 IF co=6 THEN PRINT"Pas de chance."
2590 IF cj=6 THEN PRINT"J'espere que vous n'avez pas triche !"
2600 PRINT:PRINT
2610 INPUT"Une autre partie (O/N) ";r$
2620 r$=UPPER$(r$)
2630 IF r$="O" THEN bis=-1 ELSE bis=0
2640 RETURN
```

Lignes 1060 à 1100 : Programme principal
Lignes 1150 à 1390 : Présentation du jeu
Lignes 1440 à 1460 : Activation d'une partie
Lignes 1510 à 2020 : Initialisation d'une partie
Positionnement des navires de l'ordinateur
ligne 1660
Initialisation de la table des essais de tir ligne
1710
Affichage de la grille de tir lignes 1870 à
1910
Lignes 1970 à 2010 : Déroulement d'une partie
Lignes 2060 à 2170 : Essai du joueur
Tir pour rien lignes 2220 à 2250
Bateau coulé lignes 2300 à 2360
Lignes 2410 à 2520 : Jeu de l'ordinateur
Lignes 2570 à 2640 : Affichage des résultats

La version Turbo-Pascal du programme

```
Program BatNav;
{-----}
{Bataille Navale}
{-----}

Var Ch : Char;
    Ef : String[200];
    Essai,
    O : Array[1..9,1..9] of Integer;
    Js : String[2];
    co,
    Cj,
    a,b,
    X,Y,
    I,J: Integer;
    Bis,
    Fin: Boolean;

Procedure Presentation;
{-----}
{Presentation du jeu}
{-----}
begin
    ClrScr;
    Writeln('          Bataille Navale');
    Writeln('          -----');
    Writeln;
    Writeln('Dans ce jeu, vous devez decouvrir des');
    Writeln('navires ennemis qui sont positionnes');
    Writeln('sur une grille de 9 cases sur 9, et ce');
    Writeln('avant que l''ennemi ne decouvre les');
    Writeln('votres.');
```

```

Procedure Init;
{-----}
{Initialisation des variables du jeu}
{-----}
begin
  ClrScr;
  Writeln('Dessinez une grille carree de 9 cases');
  Writeln('de cote. Placez 6 navires d''une case');
  Writeln('sur cette grille');
  Writeln;
  Writeln('Tapez une touche pour commencer. ');
  While not Keypressed Do;
  Read(Kbd,Ch);
  { Init }
  co:=0; cj:=0;
  Fin:=False;
  For i:=1 to 9 do
    For j:=1 to 9 do
      begin
        Essai[i,j]:=0;
        o[i,j]:=0;
      end;

  For I:=1 To 6 do
    begin
      x:=round(random*9)+1;
      y:=round(random*9)+1;
      if o[x,y]=0 then o[x,y]:=1 else o[x,y]:=0;
    end;
    Ef:='';
    Ef:=Concat(Ef,Ef,Ef,Ef,Ef,Ef,Ef,Ef,Ef,Ef);
    ClrScr;
    Writeln('          Bataille Navale');
    Writeln('          -----');
    GotoXY(1,5);
    Writeln('Votre grille de tir');
    Writeln;
    Writeln('          A B C D E F G H I');
    Writeln;
    For I:=1 to 9 do
      Writeln('          ',I);
  end;

Procedure Plouf;
{-----}
{Tir pour rien}
{-----}
begin
  GotoXY(1,21); Write('A l''eau');
  GotoXY(2*a+6,b+8); Write('o');
end;

```

```

Procedure Coule;
{-----}
{Touche-Coule}
{-----}
begin
  GotoXY(1,21); Write('Coule');
  GotoXY(2*a+6,b+8); Write('x');
  o[b,a]:=2; {ce coup ne peut etre rejoue}
  cj:=cj+1;
  If cj=6 then fin:=True;
end;

Procedure Utilisateur;
{-----}
{Jeu de l'utilisateur}
{-----}
begin
  GotoXY(1,20); Write(Ef);
  GotoXY(1,20); Write('Votre jeu : ');
  Readln(Js);
  a:=ord(copy(Js,1,1))-64;
  b:=ord(copy(Js,2,1))-48;
  If o[b,a]=0 then Plouf;
  If o[b,a]=1 then Coule;
  GotoXY(1,22); Writeln('Appuyez sur une touche...');
  While not Keypressed Do;
    Read(Kbd,Ch);
end;

Procedure Ordinateur;
{-----}
{Jeu de l'ordinateur}
{-----}
begin
  Repeat
    x:=round(random*9)+1;
    y:=round(random*9)+1;
  until Essai[x,y]=0;
  Essai[x,y]:=1;
  Js:=Chr(y+64)+Chr(x+48);
  GotoXY(1,20); Write(Ef);
  GotoXY(1,20); Writeln('L'ordinateur joue en ',Js);
  Writeln('Entrez le resultat du tir. ');
  Write('(1 si a l'eau, 2 si coule) : ');
  Readln(i);
  If i=2 then co:=co+1;
  If co=6 then Fin:=True;
end;

```

```
Procedure Partie;  
{-----}  
{Deroulement d'une partie}  
{-----}  
begin  
  Repeat  
    Utilisateur;  
    If Not Fin Then Ordinateur;  
  until Fin;  
end;  
  
Procedure Jeu;  
{-----}  
{Jeu complet}  
{-----}  
begin  
  Init;  
  Partie;  
end;  
  
Procedure Conclusion;  
{-----}  
{Fin d'une partie}  
{-----}  
begin  
  ClrScr;  
  If co=6 then Writeln('Pas de chance.');  If cj=6 then Writeln('J''espere que vous n''avez pas triche.');  Writeln; Writeln;  
  Write('Une autre partie (O/N) : ');  
  Readln(Ch);  
  If (Ch='O') or (Ch='o') then Bis:=True else Bis:=False  
end;  
  
begin  
{-----}  
{Programme principal}  
{-----}  
  Repeat  
    Presentation;  
    Jeu;  
    Conclusion;  
  until Not Bis;  
end.
```

9/7.3

Danger piranhas

Un touriste imprudent se baigne dans un lac infesté de piranhas. Pour leur échapper, il doit faire preuve de tactique et d'intelligence.

La règle du jeu est la suivante :

— Le baigneur peut se déplacer d'une case dans toutes les directions en utilisant les touches de fonction F1 à F7 de la manière suivante :

F7 Déplacement vers le haut et la gauche

F8 Déplacement vers le haut

F9 Déplacement vers le haut et la droite

F4 Déplacement vers la gauche

F5 Aucun déplacement

F6 Déplacement vers la droite

F1 Déplacement vers le bas et la gauche

F2 Déplacement vers le bas

F3 Déplacement vers le bas et la droite.

— Après un déplacement du baigneur, chaque piranha peut se déplacer d'une case. Ces déplacements sont guidés par l'ordinateur.

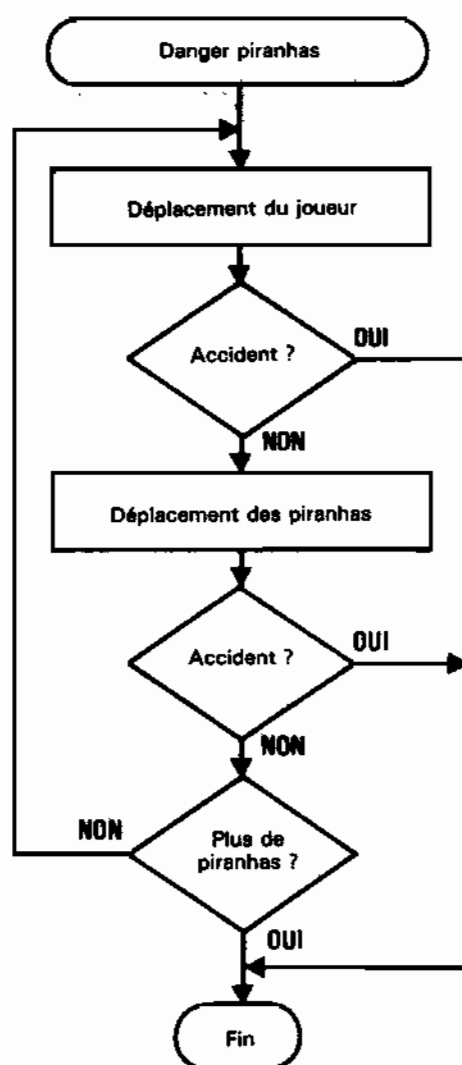
— Les piranhas sont complètement affamés. Aussi s'attaquent-ils aux rochers qui se trouvent sur leur passage. Suite à quoi ils meurent d'indigestion. Un bip sonore indique la disparition d'un piranha.

Les piranhas sont repérés par des caractères « + », les rochers par des caractères « * » et le joueur par le caractère « O ».

Le jeu d'apparence complexe se décompose en deux tâches simples :

- déplacement du joueur,
- déplacement des piranhas,

Ces deux tâches s'enchaînent comme le montre l'ordinogramme suivant :



Déplacement du joueur

Les touches de fonction **F1** à **F9** permettent de déplacer le baigneur d'une case dans tous les sens comme indiqué précédemment.

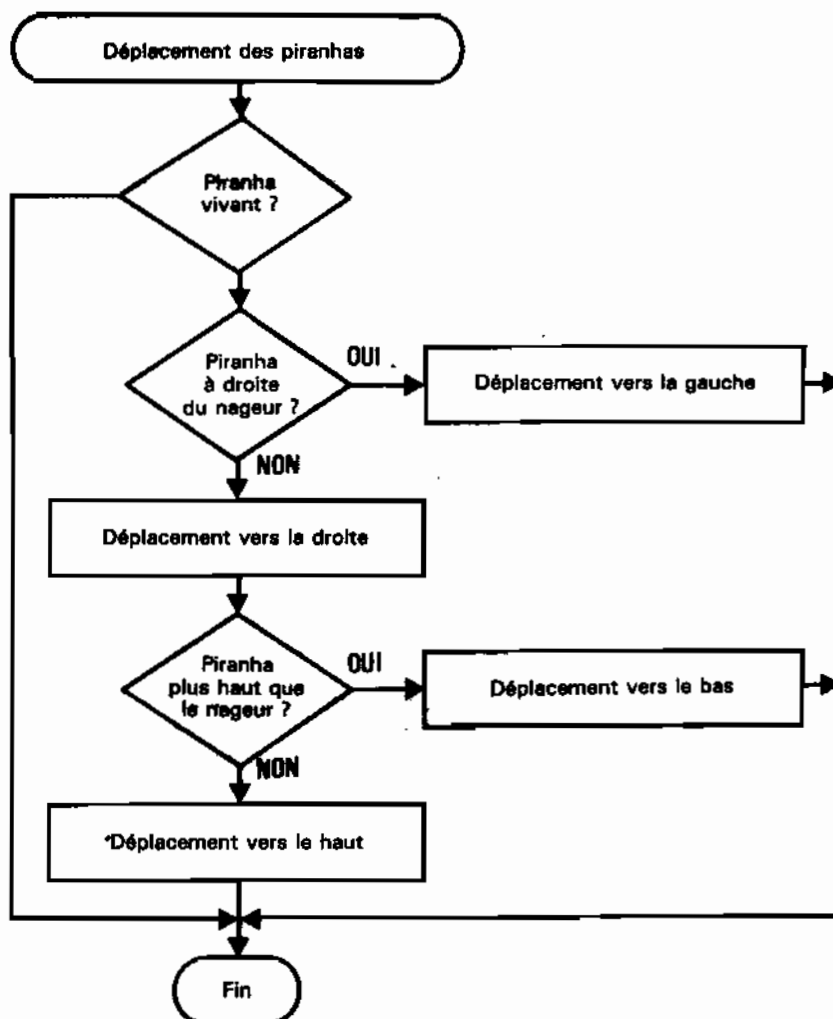
Si une autre touche est pressée, le baigneur ne se déplace pas, mais chaque piranha avance d'une case.

Si le baigneur percute la côte (extrémités de l'écran), les piranhas avancent également chacun d'une case.

Si le joueur percute un rocher ou un piranha, sa baignade s'arrête là...

Déplacement des piranhas

Les piranhas se déplacent d'une case vers le joueur à chaque tour en obéissant à la logique de l'ordinogramme suivant :



Si un piranha rencontre un rocher, il l'avale et meurt d'indigestion. Cette situation est signalée par un bip sonore.

Le joueur est sauvé s'il oblige les huit piranhas à avaler un rocher. En revanche, il périt dès qu'il est rejoint par un piranha.

Le listing du programme est le suivant :

```

1000 ' =====
1010 ' DANGER PIRANHAS
1020 ' =====
1030 ' Programme principal
1040 ' =====
1050 '
1060 GOSUB 1110 'Initialisation
1070 GOSUB 1600 'Partie
1080 GOSUB 2280 'Fin du jeu
1090 END
1100 '
1110 ' -----
1120 ' Instructions de jeu
1130 ' -----
1140 '
1150 MODE 1
1160 PRINT"          DANGER-PIRANHAS"
1170 PRINT"          -----"
1180 PRINT
1190 PRINT"Vous etes dans un lac infeste de"
1200 PRINT"piranhas. Les * representent des"
1210 PRINT"rochers, les + des piranhas et 0"
1220 PRINT"le joueur."
1230 PRINT
1240 PRINT"Les touches de fonction F1 a F9"
1250 PRINT"vous permettent de vous deplacer"
1260 PRINT"dans toutes les directions."
1270 PRINT
1280 INK 2,2,3:PEN 2
1290 PRINT"Appuyez sur une touche pour commencer."
1300 PEN 1
1310 a$=INKEY$:IF a$="" THEN 1310
1320 ' -----
1330 ' Initialisation
1340 ' -----
1350 '
1360 CLS
1370 DIM x(10),y(10) 'Tableaux de travail
1380 FOR i=1 TO 8
1390   xp=INT(RND(1)*38)+1
1400   yp=INT(RND(1)*24)+1
1410   LOCATE xp,yp
1420   a$=COPYCHR$(#0)
1430   IF a$<>" " THEN i=i-1 ELSE PRINT "+":x(i)=xp:y(i)=yp
1440 NEXT i
1450 FOR i=1 TO 30 'Nombre de rochers
1460   xr=INT(RND(1)*38)+1
1470   yr=INT(RND(1)*24)+1
1480   LOCATE xr,yr
1490   a$=COPYCHR$(#0)
1500   IF a$<>" " THEN i=i-1 ELSE PRINT "*"
1510 NEXT i
1520 xn=INT(RND(1)*38)+1
1530 yn=INT(RND(1)*24)+1
1540 LOCATE xn,yn
1550 a$=COPYCHR$(#0)
1560 IF a$<>" " THEN 1520
1570 PRINT"0"

```

```

1580 np=8 'Nombre de piranhas
1590 RETURN
1600 '-----
1610 ' Deroulement du jeu
1620 '-----
1630 '
1640 GOSUB 1680 'Utilisateur
1650 IF NOT fin THEN GOSUB 1900 'Ordinateur
1660 IF NOT fin THEN 1600 'Suite du jeu
1670 RETURN
1680 '-----
1690 ' Jeu de l'utilisateur
1700 '-----
1710 '
1720 a$=INKEY$:IF a$="" THEN 1720 'Attente d'une action
1730 LOCATE xn,yn:PRINT" ":sx=xn:sy=yn
1740 a=ASC(a$)
1750 IF a=55 THEN xn=xn-1:yn=yn-1:GOTO 1840 'Vers le haut et la gauche
1760 IF a=56 THEN yn=yn-1:GOTO 1840 'Vers le haut
1770 IF a=57 THEN xn=xn+1:yn=yn-1:GOTO 1840 'Vers le haut et la droite
1780 IF a=52 THEN xn=xn-1:GOTO 1840 'Vers la gauche
1790 IF a=53 THEN 1840 'Sur place
1800 IF a=54 THEN xn=xn+1:GOTO 1840 'Vers la droite
1810 IF a=49 THEN xn=xn-1:yn=yn+1:GOTO 1840 'Vers le bas et la gauche
1820 IF a=50 THEN yn=yn+1:GOTO 1840 'Vers le bas
1830 IF a=51 THEN xn=xn+1:yn=yn+1 'Vers le bas et la droite
1840 IF (xn=39) OR (xn<1) OR (yn=25) OR (yn<1) THEN xn=sx:yn=sy:LOCATE xn,yn:PRINT "Q":RETURN
1850 LOCATE xn,yn
1860 a$=COPYCHR$(#0)
1870 IF a$<>" " THEN GOSUB 2210:RETURN 'Fin de la partie
1880 PRINT"Q"
1890 RETURN
1900 '-----
1910 ' Deplacement piranha
1920 '-----
1930 '
1940 FOR i=1 TO 8
1950 IF x(i)=99 THEN i=i+1:IF i=9 THEN RETURN ELSE 1950
1960 LOCATE x(i),y(i):PRINT" "
1970 IF x(i)+y(i)=0 THEN i=i+1:GOTO 1950
1980 dx=xn-x(i)
1990 IF dx>=1 THEN x(i)=x(i)+1
2000 IF dx<=-1 THEN x(i)=x(i)-1
2010 dy=yn-y(i)
2020 IF dy>=1 THEN y(i)=y(i)+1
2030 IF dy<=-1 THEN y(i)=y(i)-1
2040 LOCATE x(i),y(i): a$=COPYCHR$(#0):a=ASC(a$)
2050 IF (a=42) OR (a=43) THEN GOSUB 2100:i=i+1:IF i=9 THEN RETURN ELSE 1950'dans un rocher
2060 IF a=79 THEN GOSUB 2210:i=i+1:IF i=9 THEN RETURN ELSE 1950 'dans le jeu
2070 LOCATE x(i),y(i):PRINT"+"
2080 NEXT i
2090 RETURN
2100 '-----
2110 ' rencontre avec un rocher
2120 '-----
2130 '
2140 SOUND 1,100

```

```

2150 LOCATE x(i),y(i)
2160 PRINT" "
2170 x(i)=99
2180 np=np-1
2190 IF np=0 THEN fin=-1
2200 RETURN
2210 '-----
2220 ' rencontre avec le joueur
2230 '-----
2240 '
2250 SOUND 1,200
2260 fin=-1
2270 RETURN
2280 '-----
2290 ' Fin de la partie
2300 '-----
2310 '
2320 FOR i=1 TO 10
2330   INK 0,i
2340   FOR j=1 TO 20:NEXT j
2350 NEXT i
2360 INK 0,1:CLS
2370 a=0
2380 FOR i=1 TO 8
2390   a=a+x(i)
2400 NEXT i
2410 IF a=792 THEN PRINT"Bien joue..."
2420 IF a<>792 THEN PRINT"Il ne reste plus rien de vous..."
2430 fin=-1
2440 RETURN

```

- Lignes 1060 à 1090 : Programme principal
- Lignes 1150 à 1310 : Affichage des règles du jeu
- Lignes 1360 à 1590 : Initialisation du jeu
 - Affichage des piranhas ligne 1430
 - Nombre de piranhas ligne 1380
 - Nombre de rochers ligne 1450
 - Affichage du baigneur ligne 1570
- Lignes 1640 à 1670 : Déroulement d'une partie
- Lignes 1720 à 1890 : Jeu du baigneur
 - Déplacement ligne 1750 à 1830
 - Interdiction de déplacement ligne 1840
- Lignes 1940 à 2090 : Déplacement des piranhas
- Lignes 2140 à 2200 : Un piranha a ingurgité un rocher
- Lignes 2250 à 2270 : Un piranha a rencontré le baigneur
- Lignes 2320 à 2440 : Affichage des résultats

Si le jeu vous semble trop difficile, augmentez le nombre de rochers ligne 1450 (par exemple 100).

Les piranhas attaquent

Faisant suite à la version Basic, nous vous proposons maintenant ses versions Turbo-Pascal et Assembleur. Cette triple écriture devrait vous faire progresser rapidement dans l'apprentissage des langages Turbo-Pascal et Assembleur. En effet, les programmes sont autant que possible équivalents.

VERSION TURBO-PASCAL

Le listing du programme est le suivant :

```
Program Piranhas;
{*****}
{ JEU DES PIRANHAS }
{*****}

Var
  X,
  Y   : Array[1..10] of Byte;
  T   : Array[1..80,1..24] of Byte;
  A,
  SX,
  SY,
  XN,
  YN,
  XP,
  YP,
  XR,
  YR,
  J,
  I   : Byte;
  Gagne,
  Fin  : Boolean;
  Ch   : Char;
  DX,
  DY   : Integer;

Procedure Regles;
{ Affichage des regles du jeu }

begin
  ClrScr;
  Writeln('Les piranhas attaquent');
  Writeln('-----');
  Writeln;
  Writeln('Vous vous baignez dans un lac infeste de piranhas.');

Writeln(' - les * representent des rochers,');



Writeln(' - les + representent des piranhas,');



Writeln(' - le 0 vous materialise dans le lac.');



Writeln;


```

```

Writeln('Les touches de fonction vous permettent de vous');
Writeln('deplacer d'une case dans toutes les directions. ');
Writeln('Chacun de vos déplacements entraine un déplacement de');
Writeln('chaque piranha. ');

Writeln('Le but du jeu est d'éliminer les piranhas en les faisant');
Writeln('heurter des rochers. ');
Writeln;

Write('Appuyez sur une touche ... ');
While not KeyPressed do;
Read(Kbd,Ch);
d;

Procédure Initialisation;
Initialisation des variables du jeu )

begin
For I:=1 to 80 do
  For J:=1 to 24 do
    T[I,J]:=32;      { Initialisation du tableau de jeu }

Fin:=False;
Gagne:=False;
Randomize;

ClrScr;
{ Piranhas }
For I:=1 to 8 do
begin
  XP:=Round(Random * 70)+1;
  YP:=Round(Random * 24)+1;
  A:=T[XP,YP];
  If A<>32 then I:=I-1
    else begin
      T[XP,YP]:=Ord('+');
      GotoXY(XP,YP);
      Write('+');
      X[I]:=XP;
      Y[I]:=YP;
    end;
end;

{ Rochers }
I:=1;
Repeat
  XR:=Round(Random * 70) + 1;
  YR:=Round(Random * 24) + 1;
  A:=T[XR,YR];
  If A=32 then begin
    T[XR,YR]:=Ord('*');
    GotoXY(XR,YR);
    Write('*');
    I:=I+1;
  end;
until I=60;

```

```

{ Nageur }
Repeat
  XN:=Round(Random * 78) + 1;
  YN:=Round(Random * 24) + 1;
until T[XN,YN]=32;
T[XN,YN]:=Ord('O');
GotoXY(XN,YN);
Write('O');
end;

Procédure Joueur;
{ Déplacement du joueur }

begin
  Repeat
    While not KeyPressed do;
      Read(Kbd,Ch);
until (Ord(Ch)<=27) and (Ord(Ch)>=49);
T[XN,YN]:=32;
GotoXY(XN,YN);
Write(' ');
SX:=XN;
SY:=YN;      { Sauvegarde }

Case Ord(Ch) of
  55 : { Vers le haut et la gauche }
    begin
      XN:=XN-1;
      YN:=YN-1;
    end;

  56 : { Vers le haut }
    YN:=YN-1;

  57 : { Vers le haut et la droite }
    begin
      XN:=XN+1;
      YN:=YN-1;
    end;

  52 : { Vers la gauche }
    XN:=XN-1;

  54 : { Vers la droite }
    XN:=XN+1;

  49 : { Vers le bas et la gauche }
    begin
      XN:=XN-1;
      YN:=YN+1;
    end;

  50 : { Vers le bas }
    YN:=YN+1;

```

```

51 : { Vers le bas et la droite }
    begin
        XN:=XN+1;
        YN:=YN+1;
    end;
end;

If (XN=79) or (XN<1) or (YN=25) or (YN<1) then
begin
    XN:=SX;
    YN:=SY;
    T[XN,YN]:=Ord('O');
    GotoXY(XN,YN);
    Write('O');
end
else
If T[XN,YN] <> 32 then Fin:=True
    else begin
        T[XN,YN]:=ord('O');
        GotoXY(XN,YN);
        Write('O');
    end;
end;

nd;

Procédure Piranha;
Déplacement des piranhas )

begin
    For I:=1 to 8 do
    begin
        If X[I] <> 99 then
        begin
            GotoXY(X[I],Y[I]);
            Write(' ');
            T[X[I],Y[I]]:=32;
            DX:=XN-X[I];
            If (DX >= 1) and (X[I] < 79) then X[I]:=X[I]+1;
            If (DX <= -1) and (X[I] > 1) then X[I]:=X[I]-1;
            DY:=YN-Y[I];
            If (DY >= 1) and (Y[I] < 24) then Y[I]:=Y[I]+1;
            If (DY <= -1) and (Y[I] > 1) then Y[I]:=Y[I]-1;

            Case T[X[I],Y[I]] of
                42 : begin
                    T[X[I],Y[I]]:=32;
                    GotoXY(X[I],Y[I]);
                    Write(' ');
                    X[I]:=99;
                end;
                79 : Fin:=True;
            else begin
                    T[X[I],Y[I]]:=43;
                    GotoXY(X[I],Y[I]);
                    Write('+');
                end;
            end;
        end;
    end;
end;

```



```

    end;
end;

Procedure Jeu;
{ Procédure principale }

begin
  Repeat
    Joueur;           { Déplacement du joueur   }
    Piranha;          { Déplacement des piranhas }
    J:=0;
    For I:=1 to 8 do
      If X[I]=99 then J:=J+1;
    If J=8 then begin
      Fin:=True;
      Gagne:=True;
    end;
  until Fin;
end;

Procedure Fini;
{ Commentaire sur la partie }

begin
  clrScr;
  If Gagne then Writeln('Bien joué.')
  else Writeln('Il ne reste que vos os...');
end;

{=====}
{ PROGRAMME PRINCIPAL }
{=====}

begin
  Regles;           { Affichage des règles du jeu }
  Initialisation;  { Initialisation du jeu   }
  Jeu;              { Déroulement d'une partie }
  Fini;             { Message de fin de partie }
end.

```

La logique du programme est la même que celle du programme Basic équivalent.

Remarquez dans le programme Turbo-Pascal :

- l'utilisation de l'instruction CASE pour tester les touches tapées par l'utilisateur ;
- l'utilisation du tableau d'écran T qui permet de mémoriser en permanence les objets présents sur l'écran : T: ARRAY [1..80,61..24] of Byte ;

VERSION ASSEMBLEUR

Le listing du programme est le suivant :

```

1                ORG 9000H
2                LOAD 9000H
3 9000 036A9D      JP  PIRANHA           ;Execution
4                ;
5                ;=====
6                ; ZONE DES SOUS-PROGRAMMES
7                ;=====
8                ;
9                ;-----
10               ;Effacement de l'ecran
11               ;-----
12 CLS:           EQU $
13 9003 3E02      LD  A,2
14 9005 CD0EBC    CALL SETMODE          ;MODE 2
15 9008 C9       RET
16               ;
17               ;-----
18               ; Positionnement dans le tableau T
19               ;-----
20               ; Entree: B=Y, C=X
21               ; Sortie: HL pointe sur T[X,Y]
22               ;      A=T[X,Y]
23               ;-----
24               ;
25 POSIT:         EQU $
26 9009 215000    LD  HL,80

```

```

27 900C 115000      LD    DE,00
28 900F 7B          LD    A,B
29                BIN1: EQU  $
30 9010 3D          DEC   A
31 9011 2803        JR    Z,BIN5
32 9013 19          ADD   HL,DE
33 9014 18FA        JR    BIN4
34                BIN5: EQU  $
35 9016 79          LD    A,C
36 9017 1600        LD    D,0
37 9019 5F          LD    E,A
38 901A 19          ADD   HL,DE
39 901B 11B995      LD    DE,T
40 901E 19          ADD   HL,DE
41 901F 2B          DEC   HL           ;HL=T[XP,YP]
42 9020 7E          LD    A,(HL)
43 9021 C9          RET
44
45
46                ;
47                ;-----
48                ; Initialisation des variables
49                ;-----
50                ;
51                INIT: EQU  $
52                ;Initialisation du tableau de jeu
53 9022 21B995      LD    HL,T           ;Adresse tableau
54 9025 018007      LD    BC,1920       ;Longueur
55                BIN1: EQU  $

```

```

56 9028 3E20          LD  A,32
57 902A 77            LD  (HL),A
58 902B 0B            DEC  BC
59 902C 23            INC  HL
60 902D 7B            LD  A,B
61 902E B1            OR   C
62 902F 20F7          JR   NZ,BIN1
63
64 9031 AF            XOR  A
65 9032 32B795         LD  (FIN),A
66 9035 32B895         LD  (GAGNE),A
67
68 9038 CD0390         CALL 2LS
69 903B 3E01          LD  A,1
70 903D 32B095         LD  (N),A           ;Compteur
71
72                    ;Tirage aleatoire des piranhas
73                    BIN2:    EQU  $
74 9040 ED5F          LD  A,R
75 9042 FE4E          CP   7B
76 9044 30FA          JR   NC,BIN2
77 9046 3C            INC  A
78 9047 32B195         LD  (XP),A           ;XP aleatoire
79                    BIN3:    EQU  $
80 904A ED5F          LD  A,R
81 904C FE17          CP   23
82 904E 30FA          JR   NC,BIN3
83 9050 3C            INC  A
84 9051 32B295         LD  (YP),A           ;YP aleatoire

```

```

85
86 9054 3AB295      LD  A,(YP)
87 9057 47          LD  B,A
88 9058 3AD195      LD  A,(XP)
89 905B 4F          LD  C,A
90 905C CD0990      CALL POSIT
91
92 905F FE20        CP   32
93 9061 20DD        JR   NZ,BIN2           ;Mauvaise case
94 9063 3E2B        LD  A,"+"
95 9065 77          LD  (HL),A             ;Memorisation
96
97 9066 3AD195      LD  A,(XP)
98 9069 CD6FBB      CALL SETCOL
99 906C 3AB295      LD  A,(YP)
100 906F CD72BB     CALL SETROW
101 9072 3E2B        LD  A,"+"
102 9074 CD5ABB      CALL PRINT
103
104 9077 21579D      LD  HL,X
105 907A 1600        LD  D,0
106 907C 3AB095      LD  A,(N)
107 907F 3D         DEC  A
108 9080 5F          LD  E,A
109 9081 19          ADD  HL,DE
110 9082 3AB195      LD  A,(XP)
111 9085 77          LD  (HL),A             ;X[I]=XP
112
113 9086 215F9D      LD  HL,Y

```

```

114 9089 1600      LD    D,0
115 908B 3AB095     LD    A,(N)
116 908E 3D        DEC   A
117 908F 5F        LD    E,A
118 9090 19        ADD   HL,DE
119 9091 3AB295     LD    A,(YP)
120 9094 77        LD    (HL),A          ;YCII=YP
121
122 9095 3AB095     LD    A,(N)
123 9098 3C        INC   A
124 9099 32B095     LD    (N),A
125 909C FE09      CP    9
126 909E 20A0      JR    NZ,BIN2        ;Nouvel affichage
127
128 90A0 3E01      LD    A,1
129 90A2 32B095     LD    (N),A
130
131                ;Tirage aleatoire des rochers
132                BIN6:    EQU    $
133 90A5 ED5F      LD    A,R
134 90A7 FE4E      CP    78
135 90A9 30FA      JR    NC,BIN6
136 90AB 3C        INC   A
137 90AC 32B395     LD    (XR),A          ;XR aleatoire
138                BIN7:    EQU    $
139 90AF ED5F      LD    A,R
140 90B1 FE17      CP    23
141 90B3 30FA      JR    NC,BIN7
142 90B5 3C        INC   A

```

```

143 90B6 32B495      LD   (YR),A           ;YR aleatoire
144
145 90B9 3AB495      LD   A,(YR)
146 90BC 47          LD   B,A
147 90BD 3AB395      LD   A,(XR)
148 90C0 4F          LD   C,A
149 90C1 CD0990      CALL POSIT
150
151 90C4 FE20        CP   32
152 90C6 20DD        JR   NZ,BIN6         ;Mauvaise case
153 90C8 3E2A        LD   A,"*"
154 90CA 77          LD   (HL),A          ;Memorisation
155
156 90CB 3AB395      LD   A,(XR)
157 90CE CD6FBB      CALL SETCOL
158 90D1 3AB495      LD   A,(YR)
159 90D4 CD72BB      CALL SETROW
160 90D7 3E2A        LD   A,"*"
161 90D9 CD5ABB      CALL PRINT
162
163 90DC 3AB095      LD   A,(N)
164 90DF 3C          INC   A
165 90E0 32B095      LD   (N),A
166 90E3 FE1E        CP   30
167 90E5 20BE        JR   NZ,BIN6         ;Nouvel affichage
168
169                  ;Tirage aleatoire du baigneur
170                  BIN10: EQU $
171 90E7 ED5F        LD   A,R

```

```

72 90E9 FE4E          CP    78
73 90EB 30FA          JR    NC,BIN10
74 90ED 3C            INC    A
75 90EE 32B595        LD     (XN),A          ;XN aleatoire
76                   BIN11: EQU    $
77 90F1 ED5F          LD     A,R
78 90F3 FE17          CP     23
79 90F5 30FA          JR    NC,BIN11
80 90F7 3C            INC    A
81 90F8 32B695        LD     (YN),A          ;YN aleatoire
82
83 90FB 3AB695        LD     A,(YN)
84 90FE 47            LD     B,A
85 90FF 3AB595        LD     A,(XN)
86 9102 4F            LD     C,A
87 9103 CD0990        CALL   POSIT
88
89 9106 FE20          CP     32
90 9108 20DD          JR     NZ,BIN10          ;Mauvaise case
91 910A 3E4F          LD     A,"0"
92 910C 77            LD     (HL),A          ;Memorisation
93
94 910D 3AB595        LD     A,(XN)
95 9110 CD6FBB        CALL   SETCOL
96 9113 3AB695        LD     A,(YN)
97 9116 CD72BB        CALL   SETROW
98 9119 3E4F          LD     A,"0"
99 911B CD5ABB        CALL   PRINT
00 911E C9            RET

```



```

202      ; -----
203      ; Jeu du joueur
204      ; -----
205      ;
206      JOUEUR:      EQU  $
207  911F CD06BB      CALL WAIT
208  9122 FE31        CP    49
209  9124 38F9        JR    C,JOUEUR
210  9126 FE3A        CP    58
211  9128 30F5        JR    NC,JOUEUR      ;FN KEY seules OK
212  912A 32699D      LD    (CH),A
213
214  912D 3AB695      LD    A,(YN)
215  9130 47          LD    B,A
216  9131 3AB595      LD    A,(XN)
217  9134 4F          LD    C,A
218  9135 CD0990      CALL POSIT
219  9138 3E20        LD    A,32
220  913A 77          LD    (HL),A      ;TCXN,YN]=32
221
222  913E 3AB595      LD    A,(XN)
223  913E CD6FBB      CALL SETCOL
224  9141 3AB695      LD    A,(YN)
225  9144 CD72BB      CALL SETROW
226  9147 3E20        LD    A," "
227  9149 CD5ABB      CALL PRINT
228
229  914C 3AB595      LD    A,(XN)
230  914F 32679D      LD    (SX),A

```

31 9152 3AB695	LD	A, (YN)	
32 9155 32689D	LD	(SY), A	; Sauvegarde
33			
34 9158 3A699D	LD	A, (CH)	
35 915B FE37	CP	55	
36 915D 282C	JR	Z, HAUGAU	
37 915F FE38	CP	56	
38 9161 2838	JR	Z, HAU	
39 9163 FE39	CP	57	
40 9165 283D	JR	Z, HAUDRO	
41 9167 FE34	CP	52	
42 9169 2849	JR	Z, GAU	
43 916B FE35	CP	53	
44 916D 286E	JR	Z, FINDEP	
45 916F FE36	CP	54	
46 9171 284A	JR	Z, DRO	
47 9173 FE31	CP	49	
48 9175 284F	JR	Z, BASGAU	
49 9177 FE32	CP	50	
50 9179 285B	JR	Z, BAS	
51			
52	DASDRO:	EQU	\$
53 917B 3AB595	LD	A, (XN)	
54 917E 3C	INC	A	
55 917F 32B595	LD	(XN), A	
56 9182 3AB695	LD	A, (YN)	
57 9185 3C	INC	A	
58 9186 32B695	LD	(YN), A	
59 9189 1B52	JR	FINDEP	

260

261	HAUGAU:	EQU	\$
262 918B 3AB595		LD	A, (XN)
263 918E 3D		DEC	A
264 918F 32B595		LD	(XN), A
265 9192 3AB695		LD	A, (YN)
266 9195 3D		DEC	A
267 9196 32B695		LD	(YN), A
268 9199 1B42		JR	FINDEP

269

270	HAU:	EQU	\$
271 919B 3AB695		LD	A, (YN)
272 919E 3D		DEC	A
273 919F 32B695		LD	(YN), A
274 91A2 1B39		JR	FINDEP

275

276	HAUDRO:	EQU	\$
277 91A4 3AB595		LD	A, (XN)
278 91A7 3C		INC	A
279 91AB 32B595		LD	(XN), A
280 91AB 3AB695		LD	A, (YN)
281 91AE 3D		DEC	A
282 91AF 32B695		LD	(YN), A
283 91B2 1B29		JR	FINDEP

284

285	GALJ:	EQU	\$
286 91B4 3AB595		LD	A, (XN)
287 91B7 3D		DEC	A
288 91B8 32B595		LD	(XN), A

```
289 91BB 1820      JR    FINDEP
290
291                DRO:    EQU    $
292 91BD 3AB595      LD     A, (XN)
293 91C0 3C          INC    A
294 91C1 32B595      LD     (XN), A
295 91C4 1817        JR     FINDEP
296
297                BASGAU:  EQU    $
298 91C6 3AB595      LD     A, (XN)
299 91C9 3D          DEC    A
300 91CA 32B595      LD     (XN), A
301 91CD 3AB695      LD     A, (YN)
302 91D0 3C          INC    A
303 91D1 32B695      LD     (YN), A
304 91D4 1807        JR     FINDEP
305
306                DAS:    EQU    $
307 91D6 3AB695      LD     A, (YN)
308 91D9 3C          INC    A
309 91DA 32B695      LD     (YN), A
310
311                FINDEP:  EQU    $
312 91DD 3AB595      LD     A, (XN)
313 91E0 FE4F        CP     79
314 91E2 2832        JR     Z, INAC
315 91E4 FE00        CP     0
316 91E6 282E        JR     Z, INAC
317 91EB 3AB695      LD     A, (YN)
```

318 91EB FE19	CP	25	
319 91ED 2827	JR	Z, INAC	
320 91EF FE00	CP	0	
321 91F1 2823	JR	Z, INAC	
322			
323 91F3 3AB695	LD	A, (YN)	
324 91F6 47	LD	B, A	
325 91F7 3AB595	LD	A, (XN)	
326 91FA 4F	LD	C, A	
327 91FB CD0990	CALL	POSIT	
328 91FE FE20	CP	32	
329 9200 2040	JR	NZ, FINITO	
330 9202 364F	LD	(HL), "0"	
331 9204 3AB595	LD	A, (XN)	
332 9207 CD6FBB	CALL	SETCOL	
333 920A 3AB695	LD	A, (YN)	
334 920D CD72BB	CALL	SETROW	
335 9210 3E4F	LD	A, "0"	
336 9212 CD5ABB	CALL	PRINT	
337 9215 C9	RET		
338			
339	INAC:	EQU \$	;Position refusee
340 9216 3A679D	LD	A, (SX)	
341 9219 32B595	LD	(XN), A	
342 921C 3A689D	LD	A, (SY)	
343 921F 32B695	LD	(YN), A	
344 9222 3AB695	LD	A, (YN)	
345 9225 47	LD	B, A	
346 9226 3AB595	LD	A, (XN)	

```

347 9229 4F          LD    C,A
348 922A CD0990      CALL POSIT
349 922D 3E4F        LD    A,"0"
350 922F 77          LD    (HL),A
351 9230 3AB595      LD    A,(XN)
352 9233 CD6FBB      CALL SETCOL
353 9236 3AB695      LD    A,(YN)
354 9239 CD72BB      CALL SETROW
355 923C 3E4F        LD    A,"0"
356 923E CD5ABB      CALL PRINT
357 9241 C9          RET
358
359                FINITO: EQU $
360 9242 3E01        LD    A,1
361 9244 32B795      LD    (FIN),A
362 9247 C9          RET
363                ;
364                ; -----
365                ; Jeu des piranhas
366                ; -----
367                ;
368                PIRAN: EQU $
369 9248 AF          XOR    A
370 9249 32B095      LD    (N),A
371
372                PIRAN2: EQU $                ;Boucle principale
373 924C 3AB095      LD    A,(N)
374 924F 21579D      LD    HL,X
375 9252 1600        LD    D,0

```

376 9254 5F	LD E,A
377 9255 19	ADD HL,DE
378 9256 7E	LD A,(HL)
379 9257 FE63	CP 99
380 9259 CAFD92	JP Z,AUTPIR
381 925C 32B195	LD (XP),A
382 925F 215F9D	LD HL,Y
383 9262 1600	LD D,0
384 9264 3AB095	LD A,(N)
385 9267 5F	LD E,A
386 9268 19	ADD HL,DE
387 9269 7E	LD A,(HL)
388 926A 32B29C	LD (YP),A
389 926D 3AB195	LD A,(XP)
390 9270 CD6FBB	CALL SETCOL
391 9273 3AB295	LD A,(YP)
392 9276 CD72BB	CALL SETROW
393 9279 3E20	LD A," "
394 927B CDSABB	CALL PRINT
395	
396 927E 3AB595	LD A,(XN)
397 9281 47	LD B,A
398 9282 3AB195	LD A,(XP)
399 9285 BB	CP B
400 9286 3820	JR C,XINF
401 9288 2808	JR Z,CALCY
402 928A FE02	CP 2
403 928C 3804	JR C,CALCY
404 928E 3D	DEC A

```
405 928F 32B195      LD    (XP),A
406
407                  CALCY:    EQU    $                      ;Calcul en Y
408 9292 3AB695      LD    A,(YN)
409 9295 47          LD    B,A
410 9296 3AB295      LD    A,(YP)
411 9299 BB          CP    B
412 929A 3B16        JR    C,YINF
413 929C 2B1C        JR    Z,CALCS
414 929E FE02        CP    2
415 92A0 3B1B        JR    C,CALCS
416 92A2 3D          DEC    A
417 92A3 32B295      LD    (YP),A
418 92A6 1B12        JR    CALCS
419
420                  XINF:    EQU    $
421 92AB FE4F        CP    79
422 92AA 30E6        JR    NC,CALCY
423 92AC 3C          INC    A
424 92AD 32B195      LD    (XP),A
425 92B0 18E0        JR    CALCY
426
427                  YINF:    EQU    $
428 92B2 FE1B        CP    24
429 92B4 3004        JR    NC,CALCS
430 92B6 3C          INC    A
431 92B7 32B295      LD    (YP),A
432
433                  CALCS:    EQU    $
```



```

434 92BA 3AB295      LD    A,(YP)
435 92BD 47          LD    B,A
436 92BE 3AB195      LD    A,(XP)
437 92C1 4F          LD    C,A
438 92C2 CD0990      CALL POSIT
439 92C5 FE2A        CP    42
440 92C7 2841        JR    Z,ASTER
441 92C9 FE4F        CP    79
442 92CB 2860        JR    Z,BAIGN
443 92CD 3E2B        LD    A,43
444 92CF 77          LD    (HL),A
445 92D0 3AB195      LD    A,(XP)
446 92D3 CD6FBB      CALL SETCOL
447 92D6 3AB295      LD    A,(YP)
448 92D9 CD72BB      CALL SETROW
449 92DC 3E2B        LD    A,"+"
450 92DE CD5ABB      CALL PRINT
451
452                FINPIR: EQU $
453 92E1 21579D      LD    HL,X
454 92E4 3AB095      LD    A,(N)
455 92E7 1600        LD    D,0
456 92E9 5F          LD    E,A
457 92EA 19          ADD   HL,DE
458 92EB 3AB195      LD    A,(XP)
459 92EE 77          LD    (HL),A
460 92EF 215F9D      LD    HL,Y
461 92F2 3AB095      LD    A,(N)
462 92F5 1600        LD    D,0

```

```

463 92F7 5F          LD    E,A
464 92F8 19          ADD   HL,DE
465 92F9 3AB295       LD    A,(YP)
466 92FC 77          LD    (HL),A
467                AUTPIR: EQU  $
468 92FD 3AB095       LD    A,(N)
469 9300 3C          INC   A
470 9301 32B095       LD    (10),A
471 9304 FE08        CP    B
472 9306 C24C92       JP    NZ,PIRAN2
473 9309 C9          RET
474
475                ASTER: EQU  $                ;Rencontre d'un *
476 930A 3E20        LD    A,32
477 930C 77          LD    (HL),A
478 930D 3AB195       LD    A,(XP)
479 9310 CD6FBB       CALL SETCOL
480 9313 3AB295       LD    A,(YP)
481 9316 CD72BB       CALL SETROW
482 9319 3E20        LD    A," "
483 931B CD5ABB       CALL PRINT
484 931E 3AB095       LD    A,(N)
485 9321 1600        LD    D,0
486 9323 5F          LD    E,A
487 9324 21579D       LD    HL,X
488 9327 19          ADD   HL,DE
489 9328 3E63        LD    A,99                ;Piranha mort
490 932A 77          LD    (HL),A
491 932D 1800        JR    AUTPIR

```

```

492
493          BAIGN:      EQU  $                      ;Rencontre baigneur
494 932D 3E01          LD  A,1
495 932F 32B795        LD  (FIN),A
496 9332 1BAD          JR  FINPIR
497
498 9334 09            RET
499          ;
500          ;-----
501          ; Deroulement d'une partie
502          ;-----
503          JEU:        EQU  $
504 9335 CD1F91          CALL JOUEUR
505 9338 CD4892          CALL PIRAN
506
507 933B 3E08          LD  A,8
508 933D 47            LD  B,A
509 933E AF            XOR  A
510 933F 4F            LD  C,A
511 9340 21579D        LD  HL,X
512
513          BOUMORT:    EQU  $
514 9343 7E            LD  A,(HL)
515 9344 FE63          CP  99
516 9346 280F          JR  Z,PIMORT
517          BOU2:      EQU  $
518 9348 23            INC  HL
519 9349 10F8          DJNZ BOUMORT
520

```

```

521 934B 79          LD  A,C
522 934C FE0B        CP  B
523 934E 2B0A        JR  Z,FINGA
524 9350 3AB795      LD  A,(FIN)
525 9353 B7          OR  A
526 9354 2BDF        JR  Z,JEU
527 9356 C9          RET
528
529                PIMORT: EQU $
530 9357 0C          INC  C
531 9358 1BEE        JR  BOU2
532
533                FINGA:  EQU $
534 935A 3E01        LD  A,1
535 935C 32B795      LD  (FIN),A
536 935F 32B895      LD  (GAGNE),A
537 9362 C9          RET
538
539                ;
540                ;-----
541                ; Commentaire de fin de partie
542                ;-----
543                COMMENT: EQU $
544 9363 CD0390        CALL CLS
545 9366 3AB895      LD  A,(GAGNE)
546 9369 B7          OR  A
547 936A 2B0A        JR  Z,FINMORT
548 936C 21B895      LD  HL,MES1
549 936F 110101      LD  DE,101H

```

```

550 9372 CD8093          CALL AFFICH          ; Bien joue
551 9375 C9             RET
552                     FINMORT: EQU $
553 9376 219695          LD HL,MES2
554 9379 110101          LD DE,101H
555 937C CD8093          CALL AFFICH          ; Mort
556 937F C9             RET
557                     ;
558                     ;-----
559                     ;Affichage d'un texte a l'ecran
560                     ;-----
561                     ;Entree: HL=pointeur memoire
562                     ;      DE=ligne/colonne affich
563                     ;-----
564                     AFFICH: EQU $
565 9380 E5             PUSH HL
566 9381 F5             PUSH AF
567 9382 7A             LD A,D
568 9383 CD72BB          CALL SETROW          ; Init ligne
569 9386 7B             LD A,E
570 9387 CD6FBB          CALL SETCOL          ; Init colonne
571 938A F1             POP AF
572 938B E1             POP HL
573 938C E5             PUSH HL
574 938D F5             PUSH AF
575                     AT1: EQU $              ; Boucle d'affichage
576 938E 7E             LD A,(HL)
577 938F FEFF           CP 0FFH              ; Delimiteur ?
578 9391 2806           JR Z,AT2              ; Oui => Fin d'affichage

```

```

579 9393 CD5ABB          CALL PRINT          ;Affichage caractere
580 9396 23              INC HL
581 9397 18F5            JR AT1
582                      AT2: EQU $
583 9399 F1              POP AF
584 939A E1              POP HL
585 939B C9              RET
586                      ;
587                      ;=====
588                      ; ZONE DES EQU
589                      ;=====
590                      ;
591                      CR: EQU 13              ;Code CR
592                      LF: EQU 10              ;Code LF
593                      CRLF: EQU 13           ;Code CR
594                      BS: EQU 127             ;Code BS
595                      WAIT: EQU 0BB06H        ;KM WAIT CHAR
596                      PRINT: EQU 0BB5AH       ;TXT OUTPUT
597                      TXTOUT: EQU 0BB5DH      ;TXT WR CHAR
598                      SETCOL: EQU 0BB6FH      ;TXT SET COLUMN
599                      SETROW: EQU 0BB72H      ;TXT SET ROW
600                      SETMODE: EQU 0BC0EH     ;TXT SET MODE
601                      ;
602                      ;=====
603                      ; ZONE DES DS
604                      ;=====
605                      ;
606                      REGLE: EQU $
607 939C 4C657320          DB "Les piranhas atta

```

```

607 93A0 70697261
607 93A4 6E686173
607 93A8 20617474
607 93AC 61717565
607 93B0 6E740D2A
608 93B4 2D2D2D2D DB  "-----
608 93B8 2D2D2D2D
608 93BC 2D2D2D2D
608 93C0 2D2D2D2D
608 93C4 2D2D2D2D
608 93C8 2D2D0D0A
609 93CC 0A566F75 DB  LF,"Vous vous baig
609 93D0 7320766F
609 93D4 75732062
609 93D8 6169676E
609 93DC 657A2064
609 93E0 616E7320
609 93E4 756E20
610 93E7 6C616320 DB  "lac infecte de pi
610 93EB 696E6665
610 93EF 63746520
610 93F3 64652070
610 93F7 6972616E
610 93FB 6861732E
610 93FF 0D
611 9400 0A DB  LF
612 9401 202D206C DB  " - les * represen
612 9405 6573202A
612 9409 20726570

```

612 940D 72657365

612 9411 6E74656E

612 9415 74656E74

612 9419 20646573

612 941D 20

613 941E 726F6368 DB "rochers, ",CR,LF,"

613 9422 6572732C

613 9426 0D0A202D

613 942A 206C6573

613 942E 202B2072

613 9432 6570

614 9434 72657365 DB "resentent des pir

614 9438 6E74656E

614 943C 74206465

614 9440 73207069

614 9444 72616E68

614 9448 61732C0D

614 944C 0A

615 944D 202D206C DB " - le O vous mate

615 9451 65204F20

615 9455 766F7573

615 9459 206D6174

615 945D 65726961

615 9461 6C697365

615 9465 2064616E

615 9469 73

616 946A 206C6520 DB " le lac.",CR,LF,L

616 946E 6C61632E

616 9472 0D0A0A4C


```

616 9476 65732074
616 947A 6F756368
617 947E 65732064      DB  "es de fonction vo
617 9482 6520666F
617 9486 6E637469
617 948A 6F6E2076
617 948E 6F757320
617 9492 7065726D
617 9496 65747465
617 949A 6E
618 949B 74206465      DB  "t de vous",CR,LF,
618 949F 20766F75
618 94A3 730D0A64
618 94A7 65706C61
618 94AB 63657220
618 94AF 6427
619 94B1 756E6520      DB  "une case dans tou
619 94B5 63617365
619 94B9 2064616E
619 94BD 7320746F
619 94C1 75746573
619 94C5 206C6573
619 94C9 20646972
619 94CD 65
620 94CE 6374696F      DB  "ctions.",CR,LF,"C
620 94D2 6E732E0D
620 94D6 0A436861
620 94DA 63756E20
620 94DE 64652076

```

520 94E2 6F73

521 94E4 20646570 DB " déplacements ent

521 94E8 6C616365

521 94EC 6D656E74

521 94F0 7320656E

521 94F4 74726169

521 94F8 6E652075

521 94FC 6E206465

521 9500 70

522 9501 6C616365 DB "lacement de",CR,L

522 9505 6D656E74

522 9509 2064650D

522 950D 0A636861

522 9511 71756520

522 9515 7069

523 9517 72616E68 DB "ranha.",CR,LF,LF,

523 951B 612E0D0A

523 951F 0A4C6520

523 9523 62757420

523 9527 6475206A

524 952B 65752065 DB "eu est d'eliminer

524 952F 73742064

524 9533 27656C69

524 9537 6D696E65

524 953B 72206C65

524 953F 73207069

524 9543 72616E68

524 9547 61

525 954B 7320656E DB "s en les faisant

```

625 954C 206C6573
625 9550 20666169
625 9554 73616E74
625 9558 20686575
625 955C 72746572
625 9560 20646573
625 9564 20
626 9565 726F6368      DB  "rochers.",CR,LF,L
626 9569 6572732E
626 956D 0D0A0A41
626 9571 70707579
626 9575 657A2073
627 9579 75722075      DB  "ur une touche ...
627 957D 6E652074
627 9581 6F756368
627 9585 65202E2E
627 9589 2EFF
628
629          MES1:      EQU  $
630 958B 4269656E      DB  "Bien joue.",0FFH
630 95CF 206A6F75
630 9593 652EFF
631
632          MES2:      EQU  $
633 9596 496C206E      DB  "Il ne reste que v
633 959A 65207265
633 959E 73746520
633 95A2 71756520
633 95A6 766F7320

```

633 95AA 6F732E2E

633 95AE 2E

634 95AF FF DB 0FFH

635

636 N: DS 1 ;index boucles

637 XP: DS 1 ;Abs piranha

638 YP: DS 1 ;Ord piranha

639 XR: DS 1 ;Abs rocher

640 YR: DS 1 ;Ord rocher

641 XN: DS 1 ;Abs nageur

642 YN: DS 1 ;Ord nageur

643 FIN: DS 1 ;Fin de partie

644 GAGNE: DS 1 ;Gagnant

645 T: DS 1950 ;Ecran

646 X: DS 8 ;Abs piranha

647 Y: DS 8 ;Ord piranha

648 SX: DS 1 ;Sauvegarde X

649 SY: DS 1 ;Sauvegarde Y

650 CH: DS 1 ;Caractere clavier

651

652

;

653 ;=====

654 ; PROGRAMME PRINCIPAL

655 ;=====

656 PIRANHA: EQU \$

657 9D6A CD0390 CALL CLS

658 9D6D 219C93 LD HL,REGLE

659 9D70 110101 LD DE,101H

```

660 9D73 CD8093          CALL AFFICH          ;Regle du jeu
661 9D76 CD06BB          CALL WAIT            ;Attente d'un caractere
662                      ;.....
663                      ; Initialisation des variables
664                      ;.....
665 9D79 CD2290          CALL INIT
666                      ;.....
667                      ; Deroulement d'une partie
668                      ;.....
669 9D7C CD3593          CALL JEU
670                      ;.....
671                      ; Commentaire de fin de partie
672                      ;.....
673 9D7F CD6393          CALL COMMENT
674 9D82 C9              RET
675                      END

```

Remarquez l'utilisation du registre R pour réaliser l'équivalent de l'instruction RND du Basic ou Random du Turbo-Pascal.

Pour bien comprendre le fonctionnement du programme Assembleur, reportez-vous aussi souvent que nécessaire au programme Turbo-Pascal qui suit exactement la même logique.

Si vous désirez utiliser un chargeur Basic plutôt qu'un long programme source Assembleur, voici les codes à entrer :

```

1000 REM =====
1010 REM JEU DES PIRANHAS
1020 REM =====
1030 '
1040 MEMORY &4000
1050 FOR I=&9000 TO &95AF
1060   READ A$
1070   A$="&"+A$
1080   POKE I,VAL(A$)
1090 NEXT I
1100 '
1110 FOR I=&9D6A TO &9D82
1120   READ A$
1130   A$="&"+A$
1140   POKE I,VAL(A$)
1150 NEXT I
1160 CALL &9000
1170 END
1180 '-----
1190 DATA C3,6A,9D,3E,2,CD,E,BC,C9,21,50,0,11,50,0,78
1200 DATA 3D,28,3,19,18,FA,79,16,0,5F,19,11,B9,95,19,2B
1210 DATA 7E,C9,21,B9,95,1,80,7,3E,27,77,B,23,78,B1,20
1220 DATA F7,AF,32,B7,95,32,B8,95,CD,3,90,3E,1,32,80,95
1230 DATA ED,5F,FE,4E,30,FA,3C,32,B1,95,ED,5F,FE,17,30,FA
1240 DATA 3C,32,B2,95,3A,B2,95,47,3A,B1,95,4F,CD,9,90,FE
1250 DATA 20,20,DD,3E,2B,77,3A,B1,95,CD,6F,BB,3A,B2,95,CD
1260 DATA 72,BB,3E,2B,CD,5A,BB,21,57,9D,16,0,3A,B0,95,3D
1270 DATA 5F,19,3A,B1,95,77,21,5F,9D,16,0,3A,B0,95,3D,5F
1280 DATA 19,3A,B2,95,77,3A,B0,95,3C,32,B0,95,FE,9,20,A0
1290 DATA 3E,1,32,B0,95,ED,5F,FE,4E,30,FA,3C,32,B3,95,ED
1300 DATA 5F,FE,17,30,FA,3C,32,B4,95,3A,B4,95,47,3A,B3,95
1310 DATA 4F,CD,9,90,FE,20,20,DD,3E,2A,77,3A,B3,95,CD,6F
1320 DATA BB,3A,B4,95,CD,72,BB,3E,2A,CD,5A,BB,3A,B0,95,3C
1330 DATA 32,B0,95,FE,1E,20,BE,ED,5F,FE,4E,30,FA,3C,32,B5
1340 DATA 95,ED,5F,FE,17,30,FA,3C,32,B6,95,3A,B6,95,47,3A
1350 DATA B5,95,4F,CD,9,90,FE,20,20,DD,3E,4F,77,3A,B5,95
1360 DATA CD,6F,BB,3A,B6,95,CD,72,BB,3E,4F,CD,5A,BB,C9,CD
1370 DATA 6,BB,FE,31,38,F9,FE,37,30,F5,32,69,9D,3A,B6,95
1380 DATA 47,3A,B5,95,4F,CD,9,90,3E,20,77,3A,B5,95,CD,6F
1390 DATA BB,3A,B6,95,CD,72,BB,3E,20,CD,5A,BB,3A,B5,95,32
1400 DATA 67,9D,3A,B6,95,32,68,9D,3A,69,9D,FE,37,28,2C,FE
1410 DATA 38,28,38,FE,39,28,3D,FE,34,28,49,FE,35,28,6E,FE
1420 DATA 36,28,4A,FE,31,28,4F,FE,32,28,58,3A,B5,95,3C,32
1430 DATA B5,95,3A,B6,95,3C,32,B6,95,18,52,3A,B5,95,3D,32
1440 DATA B5,95,3A,B6,95,3D,32,B6,95,18,42,3A,B6,95,3D,32
1450 DATA B6,95,18,39,3A,B5,95,3C,32,B5,95,3A,B6,95,3D,32
1460 DATA B5,95,18,29,3A,B5,95,3D,32,B5,95,18,20,3A,B5,95
1470 DATA 3C,32,B5,95,18,17,3A,B5,95,3D,32,B5,95,3A,B6,95
1480 DATA 3C,32,B6,95,18,7,3A,B6,95,3C,32,B6,95,3A,B5,95
1490 DATA FE,4F,28,32,FE,0,28,2E,3A,B6,95,FE,19,28,27,FE
1500 DATA 0,28,23,3A,B6,95,47,3A,B5,95,4F,CD,9,90,FE,20
1510 DATA 20,40,36,4F,3A,B5,95,CD,6F,BB,3A,B6,95,CD,72,BB
1520 DATA 3E,4F,CD,5A,BB,C9,3A,67,9D,32,B5,95,3A,68,9D,32
1530 DATA B6,95,3A,B6,95,47,3A,B5,95,4F,CD,9,90,3E,4F,77
1540 DATA 3A,B5,95,CD,6F,BB,3A,B6,95,CD,72,BB,3E,4F,CD,5A
1550 DATA BB,C9,3E,1,32,B7,95,C9,AF,32,B0,95,3A,B0,95,21
1560 DATA 57,9D,16,0,5F,19,7E,FE,63,CA,FD,92,32,B1,95,21
1570 DATA 5F,9D,16,0,3A,B0,95,5F,19,7E,32,B2,95,3A,B1,95
1580 DATA CD,6F,BB,3A,B2,95,CD,72,BB,3E,20,CD,5A,BB,3A,B5

```

```

1590 DATA 95,47,3A,B1,95,BB,38,20,2B,8,FE,2,38,4,3D,32
1600 DATA B1,95,3A,B6,95,47,3A,B2,95,BB,38,16,28,1C,FE,2
1610 DATA 38,18,3D,32,12,95,18,12,FE,4F,30,E6,3C,32,B1,95
1620 DATA 18,E0,FE,18,30,4,3C,32,B2,95,3A,B2,95,47,3A,B1
1630 DATA 95,4F,CD,9,90,FE,2A,28,41,FE,4F,20,60,3E,2B,77
1640 DATA 3A,B1,95,CD,6F,BB,3A,B2,95,CD,72,BB,3E,2B,CD,5A
1650 DATA BB,21,57,9D,3A,B0,95,16,0,5F,19,3A,B1,95,77,21
1660 DATA 5F,9D,3A,B0,95,16,0,5F,19,3A,B2,95,77,3A,B0,95
1670 DATA 3C,32,B0,95,FE,8,C2,4C,92,C9,3E,20,77,3A,B1,95
1680 DATA CD,6F,BB,3A,B2,95,CD,72,BB,3E,20,CD,5A,BB,3A,B0
1690 DATA 95,16,0,5F,21,57,9D,19,3E,63,77,18,D0,3E,1,32
1700 DATA B7,95,18,AD,C9,CD,1F,91,CD,48,92,3E,8,47,AF,4F
1710 DATA 21,57,9D,7E,FE,63,28,F,23,10,F8,79,FE,8,28,A
1720 DATA 3A,B7,95,B7,28,DF,C9,C,18,EE,3E,1,32,B7,95,32
1730 DATA B8,95,C9,CD,3,90,3A,B8,95,B7,28,A,21,B8,95,11
1740 DATA 1,1,CD,80,93,C9,21,96,95,11,1,1,CD,80,93,C9
1750 DATA E5,F5,7A,CD,72,BB,7B,CD,6F,BB,F1,E1,E5,F5,7E,FE
1760 DATA FF,28,6,CD,5A,BB,23,18,F5,F1,E1,C9,4C,65,73,20
1770 DATA 70,69,72,61,6E,68,61,73,20,61,74,74,61,71,75,65
1780 DATA 6E,74,D,A,2D,2D,2D,2D,2D,2D,2D,2D,2D,2D,2D,2D
1790 DATA 2D,2D,2D,2D,2D,2D,2D,2D,2D,2D,D,A,56,6F,75
1800 DATA 73,20,76,6F,75,73,20,62,61,69,67,6E,65,7A,20,64
1810 DATA 61,6E,73,20,75,6E,20,6C,61,63,20,69,6E,66,65,63
1820 DATA 74,65,20,64,65,20,70,69,72,61,6E,68,61,73,2E,D
1830 DATA A,20,2D,20,6C,65,73,20,2A,20,72,65,70,72,65,73
1840 DATA 65,6E,74,65,6E,74,65,6E,74,20,64,65,73,20,72,6F
1850 DATA 63,68,65,72,73,2C,D,A,20,2D,20,6C,65,73,20,2B
1860 DATA 20,72,65,70,72,65,73,65,6E,74,65,6E,74,20,64,65
1870 DATA 73,20,70,69,72,61,6E,68,61,73,2C,D,A,20,2D,20
1880 DATA 6C,65,20,4F,20,76,6F,75,73,20,6D,61,74,65,72,69
1890 DATA 61,6C,69,73,65,20,64,61,6E,73,20,6C,65,20,6C,61
1900 DATA 63,2E,D,A,A,4C,65,73,20,74,6F,75,63,68,65,73
1910 DATA 20,64,65,20,66,6F,6E,63,74,69,6F,6E,20,76,6F,75
1920 DATA 73,20,70,65,72,6D,65,74,74,65,6E,74,20,64,65,20
1930 DATA 76,6F,75,73,D,A,64,65,70,6C,61,63,65,72,20,64
1940 DATA 27,75,6E,65,20,63,61,73,65,20,64,61,6E,73,20,74
1950 DATA 6F,75,74,65,73,20,6C,65,73,20,64,69,72,65,63,74
1960 DATA 69,6F,6E,73,2E,D,A,43,68,61,63,75,6E,20,64,65
1970 DATA 20,76,6F,73,20,64,65,70,6C,61,63,65,6D,65,6E,74
1980 DATA 73,20,65,6E,74,72,61,69,6E,65,20,75,6E,20,64,65
1990 DATA 70,6C,61,63,65,6D,65,6E,74,20,64,65,D,A,63,68
2000 DATA 61,71,75,65,20,70,69,72,61,6E,68,61,2E,D,A,A
2010 DATA 4C,65,20,62,75,74,20,64,75,20,6A,65,75,20,65,73
2020 DATA 74,20,64,27,65,6C,69,6D,69,6E,65,72,20,6C,65,73
2030 DATA 20,70,69,72,61,6E,68,61,73,20,65,6E,20,6C,65,73
2040 DATA 20,66,61,69,73,61,6E,74,20,68,65,75,72,74,65,72
2050 DATA 20,64,65,73,20,72,6F,63,68,65,72,73,2E,D,A,A
2060 DATA 41,70,70,75,79,65,7A,20,73,75,72,20,75,6E,65,20
2070 DATA 74,6F,75,63,68,65,20,2E,2E,2E,FF,42,69,65,6E,20
2080 DATA 6A,6F,75,65,2E,FF,49,6C,20,4E,65,20,72,65,73,74
2090 DATA C5,20,71,75,65,20,76,6F,73,20,6F,73,2E,2E,2E,FF
2100 DATA CD,3,90,21,9C,93,11,1,1,CD,80,93,CD,6,BB,CD
2110 DATA 22,90,CD,35,93,CD,63,93,C9,0,0,0,0,0,0,0

```

et les codes de checksum correspondant :

```

D9 41 BF C0 A B7 C9 65 C2 11 23 A8 74 45 5E E6 A9 84 43 1C 38 8E A4 FB EB DD D2
8B AF A0 EA 74 E6 6A 5B B6 D7 5A 86 A9 4C E3 4D B0
96 8A FA 86 7E A4 AD 90 D 15 3F B9 F3 26 71 18 20 E9 BF 78 BA 3B 58 2E 9D D4 B7
F5 E8 E9 AD 8A 35 3E 20 DA 89 3 76 DD D2 2B C5 F5 D4
6C D8 5 D7

```

Pour installer et exécuter le programme Assembleur, il suffit d'exécuter le programme Basic. Les codes hexadécimaux sont POKés en mémoire à partir de l'adresse &9000.

Lorsque l'installation a été faite par le programme Basic, le programme Assembleur peut à nouveau être exécuté par une instruction CALL &9000.

Si vous désirez modifier le nombre de piranhas et/ou rochers présents sur l'écran au début de la partie, procédez comme suit :

- dans la procédure Initialisation, modifiez les indices maximum des boucles { Piranhas } et/ou { Rochers },
- piranhas : modifiez la constante de la ligne 125,
- rochers : modifiez la constante de la ligne 166.

9/8

Utilitaires

Ce chapitre est consacré aux « programmes-outils ». Ceux-ci peuvent améliorer les performances de votre AMSTRAD en lui ajoutant de nouvelles fonctions comme la copie d'écran graphique par exemple ou en vous facilitant la tâche dans divers domaines, programmation, gestion des disquettes, etc.

9/8.1

Copie d'écran graphique

Ce programme s'adresse à tous les possesseurs d'imprimantes DMP-1, DMP-2000 ou équivalentes. Il permet de faire une copie d'écran graphique, en Basic.

Le principe de recopie d'écran est le suivant :

- 1) Initialisation de l'imprimante,
- 2) Définition du saut entre deux lignes,
- 3) Pour chaque ligne, grâce à la fonction TEST, les pixels allumés (d'une couleur autre que celle du fond de l'écran) sont mémorisés,
- 4) Les données mémorisées sont ensuite envoyées à l'imprimante.

```

3000 '-----
3010 '-----Initialisations-----
3020 '-----
3030 PRINT#8,CHR$(27);"a":PRINT#8,CHR$(27);"1"
3040 ORIGIN 0,0:DIM Z(320):VE=400:i=0
3050 '-----
3060 '-----Calcul d'une ligne graphique-----
3070 '-----
3080 FOR H=0 TO 639 STEP 2
3090   C$=""
3100   FOR X=0 TO 12 STEP 2
3110     IF TEST(H,VE-X)=0 THEN C$=C$+"0" ELSE C$=C$+"1"
3120     NEXT X
3130     I=I+1:Z(I)=VAL("&X"+C$)
3140   NEXT H
3150   GOSUB 3220 'Trace
3160   VE=VE-14
3170   I=0:IF VE>0 THEN 3080
3180 END
3190 '-----
3200 '-----Trace d'une ligne graphique-----
3210 '-----
3220 A=0:AF=0:A(0)=1:A(1)=100:A(2)=100:A(3)=120
3230 FOR J=1 TO 3
3240   A=A+A(J-1):AF=AF+A(J)
3250   PRINT#8,CHR$(27);"K";CHR$(A(J));CHR$(0);
3260   FOR K=A TO AF
3270     PRINT#8,CHR$(Z(K));
3280   NEXT K
3290 NEXT J
3300 PRINT#8
3310 RETURN

```

Lignes 3000 à 3040 : Initialisation de l'imprimante et définition de l'écart entre deux lignes

Lignes 3050 à 3180 : Identification et mémorisation des pixels « allumés » sur une ligne

Lignes 3190 à 3310 : Affichage d'une ligne graphique

9/8.1.1

Turbo copie d'écran graphique

Nous vous proposons une autre version du programme développé en Partie 9, chapitre 8.1, dans laquelle la boucle de calcul des éléments graphiques à imprimer est effectuée en Assembleur.

Si vous avez utilisé le programme Basic de copie d'écran graphique, vous avez pu voir à quel point son exécution est lente.

La partie de programme incriminée se trouve entre les lignes 3080 et 3140. Dans ces lignes sont identifiés les pixels dont la couleur est différente de la couleur de fond. Pour une ligne élémentaire, ces pixels sont au nombre de 320. Une ligne d'impression est constituée de 7 pixels les uns au-dessous des autres comme le montre le schéma suivant :

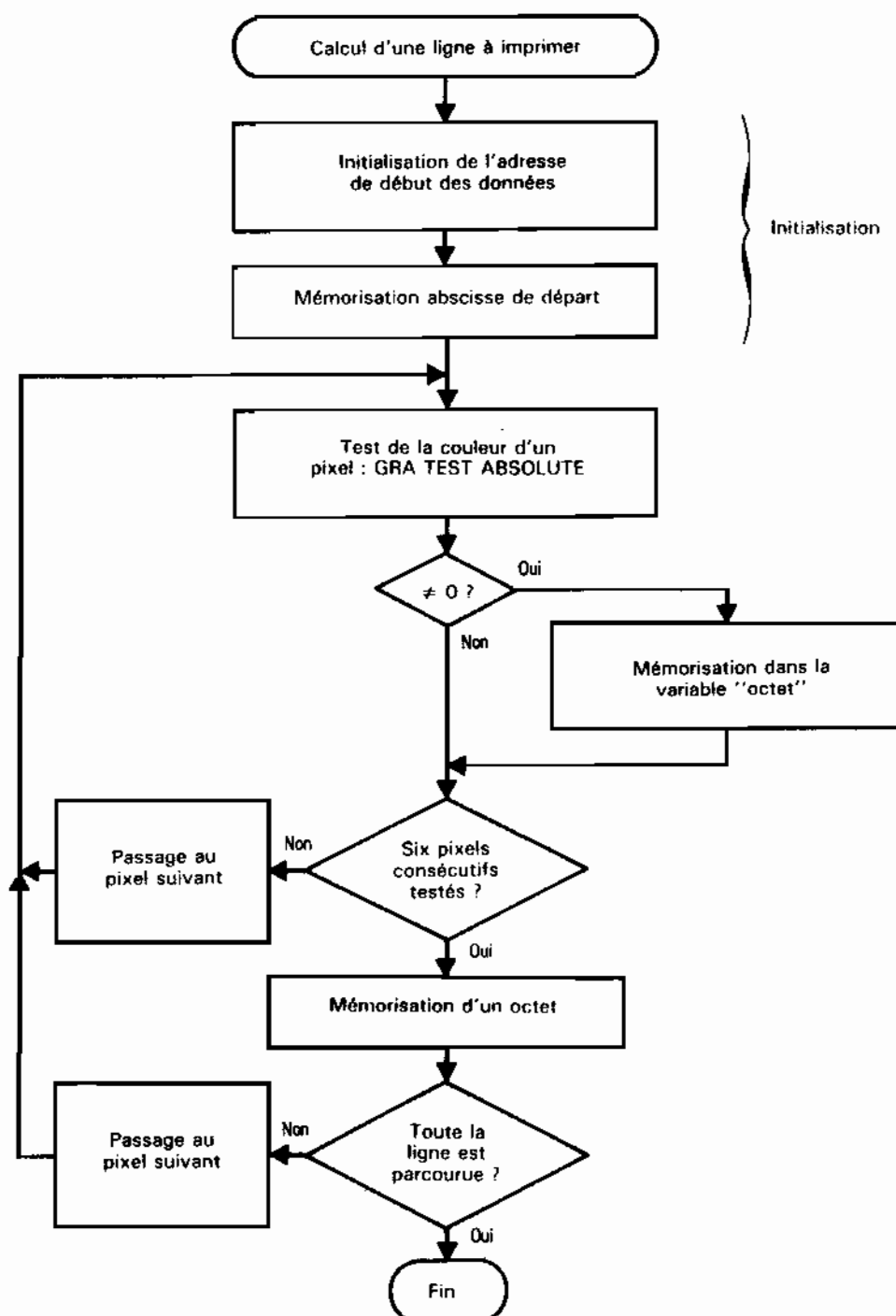
Octets

Poids 1 3 5 7 9 11

64						
32						
16						
8						
4						
2						
1						

Pour chaque ligne de 7 pixels, le nombre de points testés est donc de $320 \times 7 = 2240$. Cela explique le long temps de calcul (en Basic) entre deux lignes consécutives à imprimer. Grâce au sous-programme Assembleur qui suit, le temps de calcul pour chaque ligne est inférieur à une seconde...

La logique du sous-programme Assembleur est la suivante :



Le sous-programme Assembleur est le suivant :

```

1          ORG 9000H
2          LOAD 9000H
3 9000 216990      LD HL,ADTAB
4 9003 226590      LD (PTRTAB),HL      ;@ init tableau
5 9006 11FEFF      LD DE,OFFFEH      ;Abcisse
6          B0:     EQU $
7 9009 13          INC DE
8 900A 13          INC DE      ;X+2
9 900B 2A5C90      LD HL,(ADVE)      ;@ Variable VE
10 900E AF         XOR A
11 900F 326790      LD (POIDS),A      ;Init Poids a 0
12 9012 326890      LD (OCTET),A      ;Init Octet a 0
13          B1:     EQU $
14 9015 2B         DEC HL
15 9016 2B         DEC HL      ;Y-2
16 9017 D5        PUSH DE
17 9018 E5        PUSH HL      ;Sauvegarde registres
18 9019 CDF0BB     CALL GRATEST      ;Couleur pixel
19 901C E1        POP HL
20 901D D1        POP DE      ;Restitution registres
21 901E B7        OR A
22 901F 2816      JR Z,B2      ;Pixel a 0
23          ; Pixel a 1
24 9021 E5        PUSH HL
25 9022 D5        PUSH DE
26 9023 215E90      LD HL,ADRP01
27 9026 3A6790      LD A,(POIDS)
28 9029 5F        LD E,A
29 902A 1600      LD D,0
30 902C 19        ADD HL,DE

```

```

31 902D 46          LD  B,(HL)          ;Poids courant
32 902E 3A6890      LD  A,(OCTET)       ;Octet courant
33 9031 B0          OR   B              ;Modification octet coura
34 9032 326890      LD  (OCTET),A
35 9035 D1          POP  DE
36 9036 E1          POP  HL

37                ;

38                B2:      EQU  $

39 9037 3A6790      LD  A,(POIDS)

40 903A FE06        CP   6

41 903C 2B06        JR   Z,B3          ;1 caract complet

42 903E 3C          INC  A

43 903F 326790      LD  (POIDS),A

44 9042 18D1        JR   B1

45                ;

46                B3:      EQU  $

47 9044 E5          PUSH HL

48 9045 2A6590      LD  HL,(PTRTAB)

49 9048 3A6890      LD  A,(OCTET)

50 904B 77          LD  (HL),A          ;Memo tableau

51 904C 23          INC  HL

52 904D 226590      LD  (PTRTAB),HL

53 9050 E1          POP  HL

54 9051 7A          LD  A,D

55 9052 FE02        CP   2

56 9054 20B3        JR   NZ,B0

57 9056 79          LD  A,E

58 9057 FE80        CP   80H

59 9059 20AE        JR   NZ,B0          ;Pas fini

60 905B C9          RET

```

```

61          ;-----
62          ;Zone des EQU
63          ;-----
64          GRATEST:    EQU    0BBF0H          ;GRA TEST ABS
65          ;-----
66          ;Zone de communication
67          ;-----
68          ADVE:       DS     2              ;e var VE
69          ADRPDI:     EQU    $              ;Poids aiguilles
70 905E 4020100B        DB     64,32,16,8,4,2,1
70 9062 040201
71          PTRTAB:     DS     2              ;Pointeur ADTAB
72          POIDS:      DS     1              ;Poids courant
73          OCTET:      DS     1              ;Val octet courant
74          ADTAB:      DS     320            ;Tableau ligne
75          END

```

Pour éviter d'entrer les codes opératoires, voici un chargeur Basic qui fabrique le fichier binaire « hcbn » à partir de données hexadécimales définies en DATA:

```

1000 '-----
1010 ' Constitution du fichier de donnees
1020 '-----
1030 FOR I=&9000 TO &906F
1040   READ A$ 'Lecture d'une donnee
1050   A=VAL("&"+A$) 'Conversion numerique de la donnee
1060   POKE I,A 'Mise en memoire de la donnee
1070 NEXT I
1080 SAVE "hcbn",B,&9000,&70 'Sauvegarde du fichier binaire
1090 '-----
1100 ' Zone de donnees
1110 '-----
1120 DATA 21,69,90,22,65,90,11,FE,FF,13,13,2A,5C,90,AF,32
1130 DATA 67,90,32,68,90,2B,2B,D5,E5,CD,FO,BB,E1,D1,B7,2B
1140 DATA 16,E5,D5,21,5E,90,3A,67,90,5F,16,0,19,4E,3A,6B
1150 DATA 90,B0,32,68,90,D1,E1,3A,67,90,FE,6,28,6,3C,32
1160 DATA 67,90,18,D1,E5,2A,65,90,3A,68,90,77,23,22,65,90
1170 DATA E1,7A,FE,2,20,B3,7B,FE,80,20,AE,C9,0,0,40,20
1180 DATA 10,8,4,2,1,0,0,0,0,0,0,0,0,0,0,0,0

```

Utilisez le programme de checksum donné en Partie 9 chapitre 8.4 pour vérifier que les lignes de DATA entrées sont bien correctes (une seule erreur peut « planter » l'ordinateur, voire endommager la disquette qui se trouve dans le lecteur...).

Les données affichées par le programme de checksum doivent être les suivantes :

62 43 8B F3 CD 25 1F

Le programme de copie d'écran est le suivant :

```

1000 '-----
1010 ' Hard-Copy graphique rapide DMP 2000
1020 '-----
1030 '-----
1040 ' Initialisation
1050 '-----
1060 WIDTH 255 'Pas de retour charriot automatique
1070 MEMORY &9000 : LOAD "hcbn" 'Chargement du S/P Assembleur
1080 PRINT#8,CHR$(27);"@":PRINT#8,CHR$(27);"1"
1090 ORIGIN 0,0 : ve%=401
1100 '-----
1110 ' Programme principal
1120 '-----
1130 a=@ve% : amsb=PEEK(a+1) : alsb=PEEK(a) 'Interface avec le S/P Assembleur
1140 POKE &905C,alsb : POKE &905D,amsb
1150 CALL &9000 'Calcul d'une ligne graphique
1160 GOSUB 1230 'Trace
1170 ve%=ve%-14
1180 IF ve%>0 THEN 1130 'Passage a la prochaine colonne
1190 END
1200 '-----
1210 ' Sous-programme d'impression
1220 '-----
1230 a=0:af=0:a(0)=1:a(1)=100:a(2)=100:a(3)=120
1240 FOR j=1 TO 3
1250   a=a+a(j-1):af=af+a(j)
1260   PRINT#8,CHR$(27);"K";CHR$(a(j));CHR$(0);
1270   FOR k=a TO af
1280     PRINT#8,CHR$(PEEK(&9069+k-1));
1290   NEXT k
1300 NEXT j
1310 PRINT#8
1320 RETURN

```

Remarques :

- ligne 1060, l'utilisation de la commande **WIDTH** permet de spécifier que les passages à la ligne sur l'imprimante ne se font pas de manière automatique.
- ligne 1070, chargement du fichier binaire « hcbn ».
- ligne 1130, calcul de la valeur de la variable **ve%**.
- ligne 1140, stockage de la valeur de **ve%** en &905C et &905D pour le sous-programme Assembleur.

9/8.2

Commande PIP en Basic

Dans le but d'illustrer les ordres Basic concernant les fichiers (OPENIN, OPENOUT, PRINT #, INPUT #, CLOSEIN, CLOSEOUT), et la mise en place d'une routine de traitement des erreurs, nous vous proposons un petit programme totalement écrit en Basic. L'objectif est de faire une copie d'un fichier texte en le renommant. Pour l'utiliser, donnez le nom du fichier texte à copier (fichier source) et le nom du fichier texte où sera faite la copie (fichier destination). Si le fichier texte n'est pas présent sur la disquette, un message d'erreur est affiché, et le programme s'arrête. Dans le cas contraire, les données sont lues dans le fichier texte source, jusqu'à ce qu'une erreur de type « accès à une donnée après la fin de fichier » soit détectée. Lorsque cette erreur est détectée, le pavé de traitement des erreurs est activé (ligne 5000). Celui-ci ferme le fichier source, et lance l'exécution du bloc d'écriture du fichier destination (ligne 1100).

```
1000 REM Simulation de la commande CPM
1010 REM PIP      sur les fichiers texte
1020 REM =====
1030 DIM a$(400) 'Nombre maximum de lignes
1032 ON ERROR GOTO 5000
1033 CLS:PRINT"Copie de fichiers texte sur "
1034 PRINT"la meme disquette"
1035 PRINT:INPUT "Nom du fichier source: ",N1$
1036 INPUT"Nom du fichier destination: ",N2$
1037 '-----
1038 ' Lecture fichier source
1039 '-----
1040 OPENIN n1$
```

```
1050 FOR i=1 TO 1000
1060   INPUT#9,a$(i)
1070 NEXT
1090 '-----
1091 'Ecriture fichier destination
1092 '-----
1100 OPENOUT n2$
1110 FOR i=1 TO 259
1120   PRINT#9,a$(i)
1130 NEXT i
1140 CLOSEOUT
1201 END
1210 '-----
1211 ' Gestion des erreurs
1212 '-----
5000 IF I=0 THEN 5030 'Fichier source absent
5001 MEMO=I-1 'Memorisation Longueur fichier
5010 CLOSEIN
5020 GOTO 1100 'Reprise execution
5030 RESUME 5031 'Fin d'execution si fichier absent
5031 END
```

Lignes 1000 à 1036 : Présentation et entrée des données
Lignes 1040 à 1070 : Lecture du fichier source
Lignes 1100 à 1201 : Ecriture du fichier destination
Lignes 5000 à 5031 : Traitement des erreurs.

9/8.3

Transformation du clavier QWERTY en clavier AZERTY sous CP/M Plus

Certains de nos lecteurs sont peut-être habitués à travailler avec un clavier AZERTY, et possèdent un AMSTRAD CPC muni d'un clavier QWERTY.

Une commande de CP/M plus permet de reconfigurer certaines touches du clavier. Il s'agit de la commande **SETKEYS**. Elle est utilisée avec un fichier de données qui contient la redéfinition des touches désirées dans un code bien particulier.

Le fichier de données est un fichier texte qui peut être créé avec n'importe quel éditeur de textes (par exemple Amlettre, Wordstar, etc.) de lignes (comme par exemple EDLIN), ou même par la commande PIP en faisant par exemple :

PIP FICH.EX = CON: <cr>

Cette commande entrée, tapez le texte correspondant au fichier de redéfinition. Terminez la saisie en tapant simultanément sur les touches **Ctrl** et **Z**.

Les lignes du fichier de données contiennent, dans l'ordre :

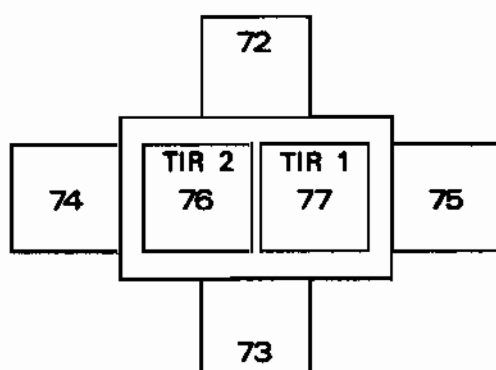
- le numéro de la touche à redéfinir ;
- un état shift (S pour SHIFT, C pour CONTROL, N pour NOTHING) qui précise si la touche est redéfinie lorsqu'une des touches CONTROL ou SHIFT est pressée ou quand aucune de ces deux touches n'est pressée ;
- le ou les caractères de redéfinition, entre guillemets.

Pour avoir plus de détails à ce sujet, reportez-vous à la Partie 3, chap. 4 page 23 : l'ordre SETKEYS est commenté.

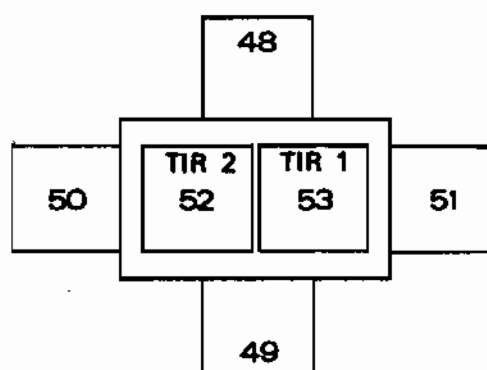
Le numéro de chaque touche du clavier est défini ci-dessous :

66	64	65	57	56	49	48	41	40	33	32	25	24	16	79	10	11	3
68	67	69	58	50	51	43	42	35	34	27	28	17	18		20	12	4
70	69	60	61	53	52	44	45	37	36	29	26	19			13	14	5
21		71	63	62	55	54	46	38	39	31	30	22	21		15	0	7
23		9					47						6		8	2	1

MANETTE DE JEU 0



MANETTE DE JEU 1



Dans le cas qui nous intéresse, les touches suivantes sont à redéfinir :

Touche à redéfinir	Position	Touche redéfinie
64	N	&
64	S	1
65	N	é
65	S	2
57	N	''
57	S	3
56	N	'
56	S	4
49	N	(
49	S	5
48	N	\$
48	S	6
41	N	è
41	S	7

Touche à redéfinir	Position	Touche redéfinie
40	N	!
40	S	8
33	N	ç
33	S	9
32	N	à
32	S	0
25	N	)
25	S	°
67	N	a
67	S	A
59	N	z
59	S	Z
69	N	q
69	S	Q
71	N	w
71	S	W
29	N	m
29	S	M
28	N	u
28	S	%
38	N	,
38	S	?
39	N	;
39	S	:
31	N	.
31	S	/
30	N	=
30	S	+

Le listing du fichier texte de redéfinition est donc le suivant :

64 N "&"	25 S "^'##2' "
64 S "1"	67 N "a"
65 N "^'##E1' "	67 S "A"
65 S "2"	59 N "z"
57 N "^'##22' "	59 S "Z"
57 S "3"	69 N "q"
56 N "r"	69 S "Q"
56 S "4"	71 N "w"
49 N "("	71 S "W"
49 S "5"	29 N "m"
48 N "^'##A6' "	29 S "M"
48 S "6"	28 N "^'##EE' "
41 N "^'##EB' "	28 S "%" "
41 S "7"	38 N ", " "
40 N "1"	38 S "? " "
40 S "8"	39 N "; " "
33 N "^'##f\$' "	39 S ". " "
33 S "9"	31 N ": " "
32 N "^'##EA' "	31 S "/" "
32 S "0"	30 N "=" "
25 N "> " "	30 S "+" "

Supposons que vous ayez appelé le fichier texte de redéfinition « MON-CLAV.IER », pour pouvoir l'exécuter, il vous faudra :

- passer sous CP/M en tapant « !CPM », puis,
- taper « SETKEYS MONCLAV.IER ».

Nota : Le signe ! (code ASCII 124) s'obtient sur les CPC en tapant SHIFTΛ. Sur claviers AZERTY, la frappe de ces touches renvoie le caractère ! à l'écran.

Les codes ESCape de redéfinition sont issus de la table suivante qui donne le jeu de caractères disponibles sous CP/M plus et leur code d'accès sous CP/M plus :

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	∞	-	+	Δ	⊗	×	÷	∴	π	!	Σ	—	—	±	△	Ω
1	α	β	γ	δ	ε	θ	λ	μ	π	p	σ	t	∅	χ	μ	ω
2		!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7		p	r	s	t	u	v	w	x	y	z	{		}	~	0
8	■	▬	▮	▮	▮	▮	▮	▮	▮	▮	▮	▮	▮	▮	▮	▮
9	●	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
A	ā	ō	°	£	©	π	§	+	14	12	34	«	»	Pt	¿	¡
B	£	ç	..	'	^	/..	18	38	58	78	B	O	•	Y	®	™
C	Á	É	Í	Ó	Ú	Â	Ê	Î	Ô	Û	Ä	È	Ì	Ò	Ù	ÿ
D	Ä	È	Ï	Ö	Ü	Ç	Æ	À	Ø	Ñ	Ã	Õ	≥	≤	≠	≈
E	á	é	í	ó	ú	â	ê	î	ô	û	ä	è	ì	ò	ù	ÿ
F	ä	ë	ï	ö	ü	ç	æ	a	ø	ñ	ã	õ	⇒	⇐	⇔	≡

Caractères disponibles sous CP/M plus

9/8.4

Checksum, vérificateur de données

Nous vous donnons ici le listing d'un petit programme (dit de checksum) qui vous permettra d'être sûr que les programmes assembleurs sont conformes à ceux présentés dans votre ouvrage « Comment exploiter toutes les ressources et augmenter les performances de votre Amstrad ». Précisons d'abord que cet utilitaire ne concerne que les programmes assembleur qui sont incorporés dans des programmes Basic sous forme de DATA.

Les programmes de ce type seront désormais accompagnés de valeurs de checksum destinées à vérifier leur cohérence.

Mais qu'est-ce qu'un checksum ? Encore un mot anglais pour désigner une opération assez étonnante de dépistage des erreurs.

De nombreuses formes de checksum existent. Pour celui qui nous concerne, la valeur de vérification est obtenue en additionnant modulo 255 les données du programme 16 par 16. Une valeur de vérification existe donc pour 16 codes entrés. Pour vérifier la validité d'un programme qui contient un grand nombre de données numériques, utilisez ce programme et vérifiez que les données de checksum sont correctes.

Le listing de cet utilitaire bien précieux est le suivant. N'hésitez pas à le taper : il est court et il peut vous éviter bien des erreurs, et ... bien des courriers si vous êtes friand d'Assembleur !

```
50000 '-----
50010 '----- CHECKSUM POUR VERIFIER DES CODES HEXA DE -----
50020 '----- PROGRAMMES ASSEMBLEUR ENTRES SOUS BASIC -----
50030 '-----
50040 MODE 2
50050 PRINT"CHECKSUM AMSTRAD":PRINT"-----":PRINT
50060 INPUT "Nombre de donnees a verifier : ";ND
50070 PRINT "Resultat sur 1) Ecran"
50080 INPUT "2) Imprimante : ";per
```



```
50090 IF PER<>1 AND PER<>2 THEN 50070
50100 IF PER=2 THEN PER=8 'Imprimante
50110 RESTORE
50120 LOCATE #1,1,10
50130 FOR I=1 TO ND STEP 16
50140     K=0 'Initialisation du compteur de checksum
50150     FOR J=0 TO 15
50160         READ A$
50170         A=VAL("&" + A$)
50180         K=(K+A) MOD 255
50190     NEXT J
50200     PRINT #PER, HEX$(K); " ";
50210 NEXT I
50220 IF PER=8 THEN PRINT#8 'Fin d'impression
```

A titre d'exemple, nous donnons le petit programme suivant :

```
100 '-----
101 'Programme test pour le checksum
110 '-----
120 DATA 10,20,30,40,50,60,70,80,90,A0,B0,C0,D0,E0,F0,01
130 DATA 02,03,04,05,06,07,08,09,09,0A,0B,0C,0D,0E,0F,10
```

Entrez le programme ci-dessus. Supposons que vous ayez appelé le programme de checksum « CHECK » ; tapez « MERGE CHECK », puis « RUN 50 000 ». Vous verrez alors apparaître les données suivantes si vous avez bien tapé le listing test:

88 90

9/8.5

Dump hexadécimal et ASCII

Dans le but de démystifier et d'apprendre par des exemples concrets les langages Basic, Assembleur et Pascal, la plupart des programmes présentés seront développés, dans la mesure où cela offre un intérêt, pour deux ou trois de ces langages. Le premier de ces programmes est un Dump. Il est développé en Basic et en Assembleur.

La version Basic est très rapide à écrire, mais fonctionne assez lentement. Par contre, la version Assembleur fonctionne beaucoup plus rapidement, mais demande beaucoup plus de temps pour être entrée au clavier.

Définition du Dump

Comme beaucoup de termes en informatique, le terme « dump » vient de l'anglais et signifie littéralement « décharger » ou « déposer ». Nous traduirons cela par dérouler. Un dump de mémoire consiste donc à vider la mémoire, ou en d'autres termes à lire son contenu. Le programme que nous présentons demande l'adresse de départ et l'adresse de fin de dump. Il passe alors automatiquement en mode 2 et affiche le dump, 16 octets par 16 selon le format suivant :

Adresse 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 ASCII

Adresse représente l'adresse du premier octet affiché.

01 à 16 représentent la valeur de 16 octets consécutifs dont le premier occupe l'adresse spécifiée en début de ligne.

ASCII représente la conversion ASCII de chaque octet. Lorsque cette conversion ASCII n'est pas affichable sous la forme d'un caractère, un point décimal (.) est affiché.

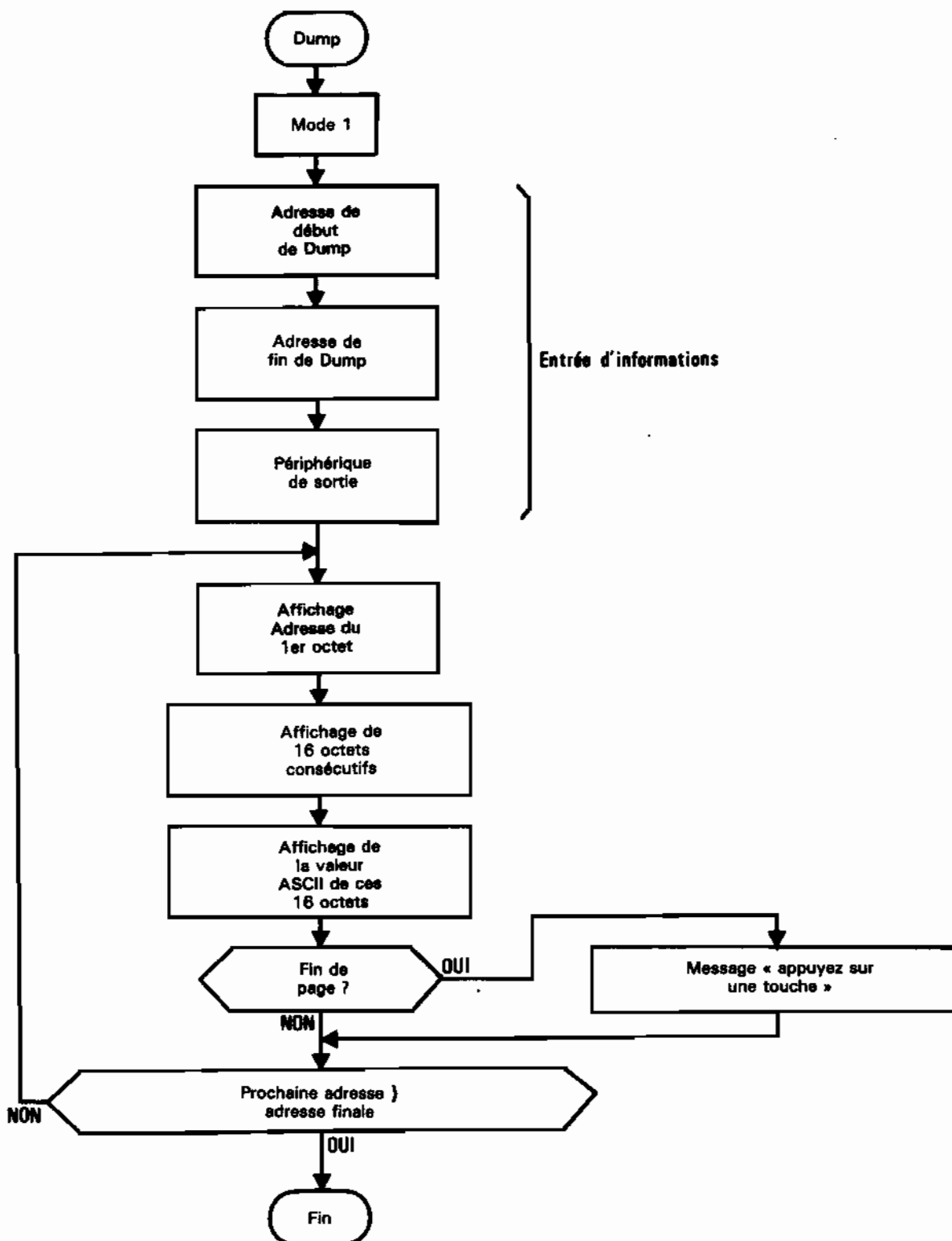
9/8.5.1

Programme de Dump en Basic

Analysons la structure du programme de Dump Basic.

- 1) Passage en MODE 1.
- 2) Entrée des informations concernant le Dump :
 - adresse de début de Dump (en décimal ou hexa. [précédé de &]) ;
 - adresse de fin de Dump (en décimal ou hexa. [précédé de &]) ;
 - choix de la sortie (écran ou imprimante).
- 3) Affichage/impression des données lues en mémoire :
 - adresse du premier octet ;
 - 16 octets consécutifs ;
 - valeurs ASCII de ces 16 octets ;
 - test de fin d'affichage/impression ;
 - poursuite éventuelle de l'affichage/impression.

Ce qui se traduit par l'organigramme suivant :



Le listing du programme est le suivant :

```
1000 REM -----
1010 REM -                DUMP Hexadecimal et ASCII                -
1020 REM -----
1030 '
1040 '=====
1050 ' Entree des donnees
1060 '=====
1070 MODE 1
1080 PRINT "Utilitaire de DUMP"
1090 LOCATE 1,4:INPUT"A partir de : ";AD
1100 IF AD>2^16-1 THEN 1090
1110 LOCATE 1,6:INPUT"jusqu'a : ";AF
1120 IF AF>2^16-1 THEN 1110
1130 LOCATE 1,8: PRINT "1) Affichage sur ecran,"
1140 PRINT "2) impression sur imprimante."
1150 INPUT "Votre choix : ";CH
1160 IF CH<>1 AND CH<>2 THEN 1130
1170 IF CH=1 THEN PER=1 ELSE PER=8
1180 '
2000 '=====
2010 ' Affichage des donnees lues
2020 '=====
2030 MODE 2
2040 IF PER=8 THEN LOCATE 25,13:PRINT"Impression en cours..."
2050 FOR I=AD TO AF STEP 16
2060   LIGNE=LIGNE+1 'Indicateur de ligne d'affichage
2070   AS$="" 'Initialisation de la ligne des codes ASCII
2080   IF I<2^4 THEN AD$="000"+HEX$(I):GOTO 2110
2090   IF I<2^8 THEN AD$="00"+HEX$(I):GOTO 2110
2100   IF I<2^12 THEN AD$="0"+HEX$(I) ELSE AD$=HEX$(I)
2110   PRINT #PER,AD$;" "; 'Affichage de la premiere adresse
```

```

2120   FOR J=0 TO 15
2130     A=PEEK(I+J)
2140     B$=CHR$(A)
2150     IF A<32 THEN B$="."
2160     IF (A>128 AND PER=B) THEN B$="."
2170     AS$=AS$+B$
2180     IF A<16 THEN A$="0"+HEX$(A) ELSE A$=HEX$(A)
2190     PRINT #PER,A$;" ";
2200   NEXT J
2210   PRINT #PER,"      ";AS$
2220   IF (LIGNE<>23 OR I>=AF) OR (PER=8) THEN 2260
2230   LOCATE 10,25:PRINT "Appuyez sur une touche pour voir la suite"
2240   a$=INKEY$:IF a$="" THEN 2240
2250   CLS #PER:LOCATE 1,1:LIGNE=0 'A nouveau 1ere ligne d'affichage
2260 NEXT I
2270 IF PER=1 THEN LOCATE 1,24 ELSE CLS

```

L'exécution du programme en demandant la sortie des données sur imprimante pour un Dump entre #0001 et #00C5 (par exemple) donne le résultat suivant :

0001	B9 7F ED 49 C3 7B 05 C3 8A B9 C3 84 B9 C5 C9 C3	..I.é.....
0011	1D BA C3 17 BA D5 C9 C3 C7 B9 C3 B9 B9 E9 00 C3	
0021	C6 BA C3 C1 B9 00 00 C3 35 BA 00 ED 49 D9 FB C7	5...I...
0031	D9 21 2B 00 71 18 08 C3 41 B9 C9 00 00 00 00 CA	..!+.q...A.....
0041	00 00 00 00 00 00 9C 04 00 00 00 20 A9 20 19 19	
0051	2C 19 0D 01 BF 22 49 6D 70 72 65 73 73 69 6F 6E	,...."Impression
0061	20 65 6E 20 63 6F 75 72 73 2E 2E 2E 22 00 00 00	en cours..."...
0071	00 00 00 D3 EF 03 00 00 41 D3 F4 FF 03 2B 0D 00	A....(..
0081	00 C1 29 20 00 00 00 00 00 00 00 00 00 00 00	..)
0091	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00A1	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00B1	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00C1	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	

9/8.5.2

Programme de Dump en Assembleur

Partons de la logique développée pour le Dump Basic mais détaillons-la de façon à ne laisser passer aucun point obscur.

Le Dump peut se décomposer en deux grandes parties (entrée des données et affichage du résultat). Deux sous-programmes ont été créés à cet effet. Ils ont pour noms respectifs « ENTREE » et « DUMP ». Le programme principal qui occupe les adresses #9000 à #9006 se contente d'appeler ces deux sous-programmes.

Entrée des données

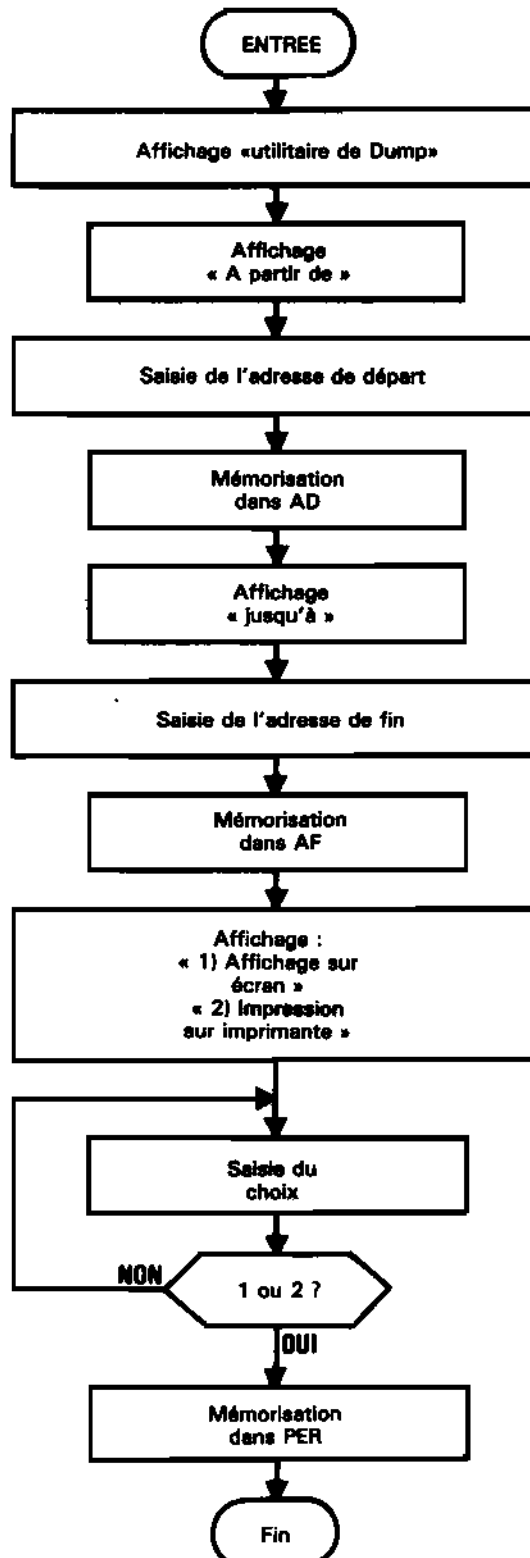
Affichage de messages

Plusieurs messages vont être affichés dans ce sous-programme. Pour faciliter les choses, nous utiliserons la routine TST Output du Firmware (en #BB5A) qui a l'avantage de préserver l'état de tous les indicateurs et de tous les registres. Cette routine affiche un caractère à la position courante du curseur. Pour afficher un texte, il faut donc réaliser une boucle sur cette routine. De plus, pour savoir que le texte à afficher est fini, il faut insérer un terminateur en fin de texte. Nous avons choisi le caractère de code ASCII 0 (nul).

Lecture de données au clavier

Le programme d'entrée des données va lire plusieurs données alphanumériques au clavier. Pour réaliser la lecture d'un caractère, nous utilisons la routine KM WAIT CHAR (en #BB06). Pour lire une chaîne alphanumérique, nous devons donc faire une boucle sur cette routine. Connaissant la longueur de la chaîne à saisir, la fin de la saisie se fera automatiquement.

Les routines d'affichage et de lectures mises en place, la structure du module d'entrée des données sera la suivante :



Affichage des données lues en mémoire

Initialisation

Pour augmenter le nombre de données par ligne (dans le cas où le résultat est sorti sur écran), le mode d'affichage choisi est le mode 80 colonnes (MODE 2). Ce mode est activé grâce à la fonction SCR SET MODE (#BC0E) du FIRMWARE.

Si la sortie se fait sur imprimante, le message « Impression en cours » est affiché sur l'écran, toujours grâce à la routine TXT OUTPUT.

L'adresse de départ de DUMP est mémorisée dans la variable mot « I1 ». Le numéro de la ligne affichée est mémorisé dans la variable octet « LIGNE ». Enfin, le buffer « BUFF » destiné à recevoir plusieurs données durant l'affichage est initialisé. Notez à ce sujet l'utilisation de l'ordre Assembleur « DJNZ » ligne 62 qui permet de réaliser des opérations répétitives très aisément.

Affichage du DUMP

L'affichage se réalise en trois étapes :

- adresse ;
- données octets ;
- valeurs ASCII correspondantes.

Les lignes affichées/imprimées ont la structure suivante :

Adresse 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 ASCII

• Affichage de l'adresse

L'adresse est affichée/imprimée en utilisant une routine dont le but est de :

- convertir une donnée hexadécimale codée sur un octet en deux données ASCII,
- afficher ou imprimer ces deux données ASCII.

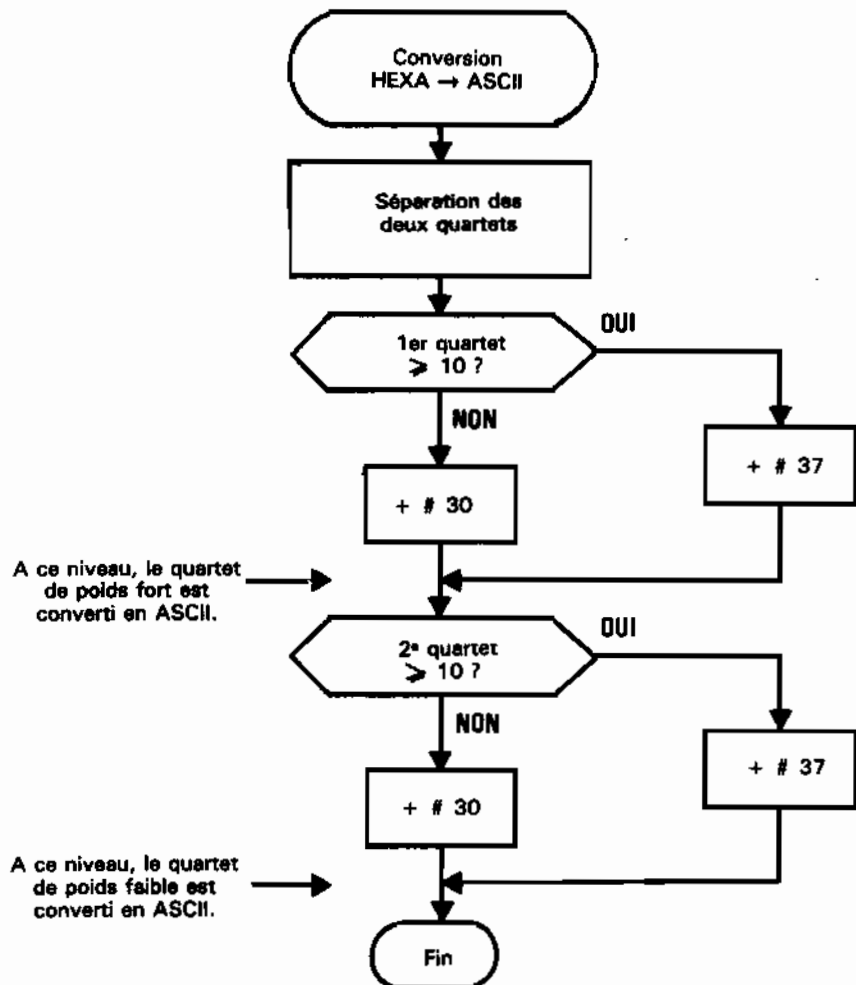
Une telle routine est nécessaire car l'affichage ou l'impression ne peut se faire qu'à travers des caractères ASCII.

La conversion HEXA → ASCII est faite en séparant l'octet de poids fort de l'octet de poids faible (chacun donnera un caractère ASCII). Chacun des deux quartets ainsi obtenus est comparé au chiffre 10. S'il est supérieur ou égal, le nombre #37 est ajouté à quartet ; sinon, le nombre #30 est ajouté au quartet. Ce qui se traduit par l'organigramme page suivante.

• Affichage des données octets

Chaque octet est affiché/imprimé en utilisant la routine de conversion/affichage décrite ci-dessus. A noter à ce sujet que deux routines ont

été développées. La première (de nom ECRECR) convertit un octet en deux caractères ASCII et affiche ces caractères sur l'écran). La seconde (de nom ECRIMP) convertit un octet en deux caractères ASCII et envoie ces deux caractères sur l'imprimante.



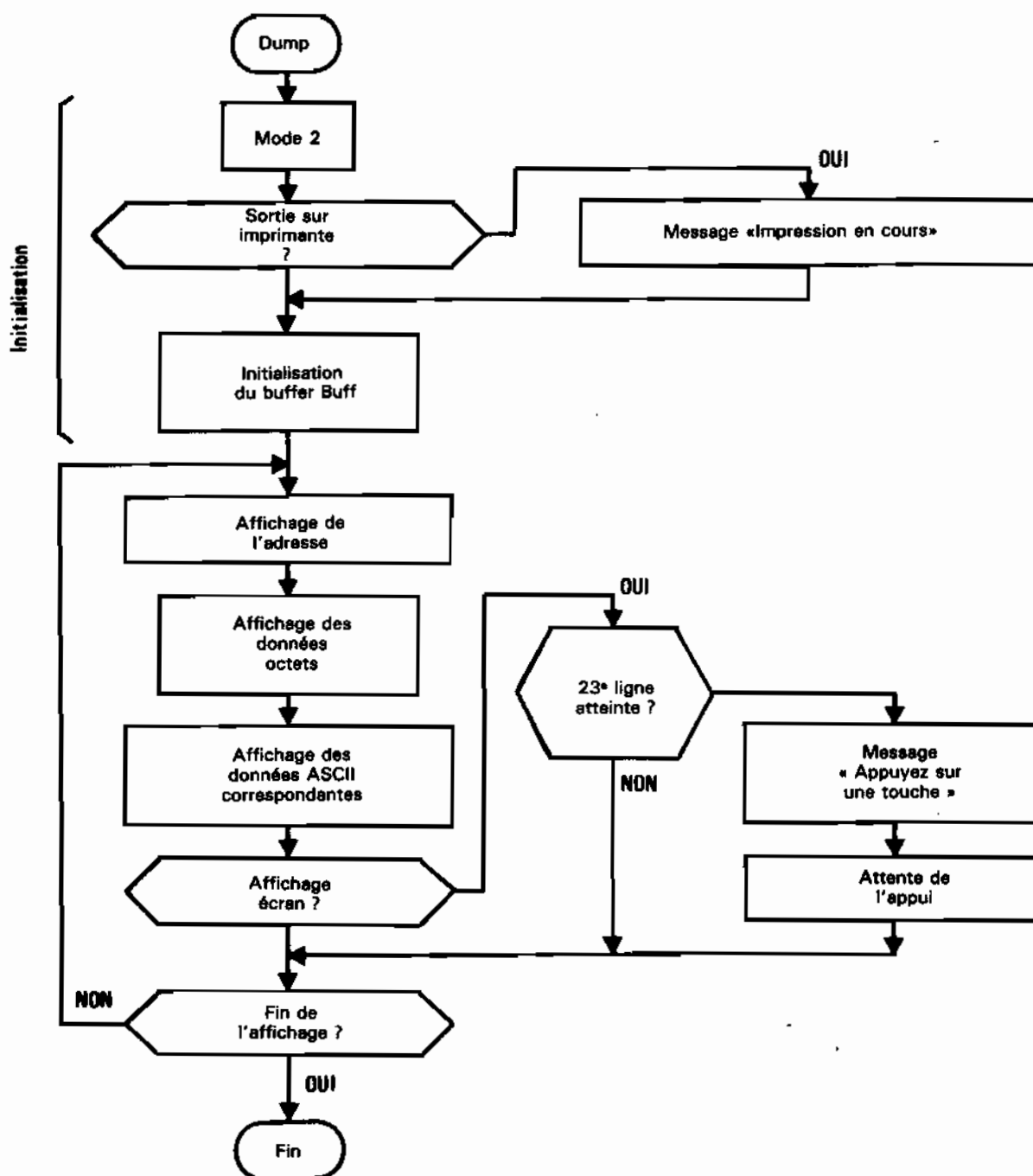
- Affichage des données ASCII correspondantes

Les données ASCII affichables et imprimables sur des imprimantes classiques ont des codes compris entre 32 et 128. Tous les autres codes sont systématiquement ignorés par le programme de Dump et remplacés par un point décimal (.).

Une fois les données ASCII affichées, la ligne de Dump suivante est amorcée :

- dans le cas d'un affichage écran, si la 23^e ligne n'a pas été atteinte ;
- dans tous les cas, si toutes les données n'ont pas été affichées.

Les différents tests et actions cités ci-dessus s'enchaînent comme le montre l'organigramme suivant :



Le programme assembleur a été saisi sous « ZEN ». Les instructions LOAD et END ne sont pas forcément nécessaires avec d'autres assembleurs. Le listing du programme est le suivant :

```

1          ORG 9000H
2          LOAD 9000H
3          ;-----
4          ;Zone des constantes du programme
5          ;-----
6          SETMODE:    EQU 0B00EH          ; SCR SET MODE
7          WRCHAR:     EQU 0BB5AH          ;TXT OUTPUT
8          WCHAR:      EQU 0BB06H          ; KM WAIT CHAR
9          PRCHAR:     EQU 0BD2BH          ;MC PRINT CHAR
10         BS:         EQU 8               ;Back Space
11         DEL:        EQU 127             ;Caractere DELeTe
12         CR:         EQU 0DH             ;Code Carriage Return
13         LF:         EQU 0AH             ;Code Line Feed
14         ;
15         ;-----
16         ;Programme principal
17         ;-----
18         ;
19 9000 CD4F91          CALL ENTREE          ;des donnees
20 9003 CD0790          CALL DUMP           ;Affichage du DUMP
21 9006 C9             RET
22         ;
23         ;
24         ;-----
25         ;Zone des sous-programmes
26         ;-----
27         ;
28         ;-----

```

```

29          ;DUMP: Affichage/Impression du
30          ;dump.
31          ;-----
32          DUMP:      EQU  $
33 9007 3E02          LD  A,2
34 9009 CD0EBC       CALL SETMODE
35 900C AF           XOR  A
36 900D 329A93       LD   (LIGNE),A      ;Initialisation
37 9010 3A9593       LD   A,(PER)
38 9013 FE01         CP   1
39 9015 2B15         JR   Z,DU1          ;Pas de message
40 9017 3E0A         LD   A,LF
41 9019 CD5ABB       CALL WRCHAR        ;Line feed
42 901C 3E0A         LD   A,LF
43 901E CD5ABB       CALL WRCHAR        ;Line Feed
44 9021 3E0A         LD   A,LF
45 9023 CD5ABB       CALL WRCHAR        ;line Feed
46 9026 21F992       LD   ,HL,TEX4
47 9029 CD1892       CALL AFFICHE       ;Affiche texte
48          ;
49          DU1:      EQU  $
50 902C 2A9193       LD   HL,(AD)
51 902F 229693       LD   (I1),HL      ;Pointeur en memoire
52          DU1P:    EQU  $           ;Boucle principale
53 9032 3A9A93       LD   A,(LIGNE)
54 9035 3C           INC  A
55 9036 329A93       LD   (LIGNE),A    ;Inc. cpt. ligne
56 9039 217C93       LD   HL,BUFF
57 903C 3E00         LD   A,0
58 903E 0610         LD   B,16         ;Longueur buffer
59          DU2:      EQU  $           ;Boucle d'effacement
60 9040 77           LD   (HL),A

```

Partie 9 : Programmes

```

61 9041 23          INC  HL
62 9042 10FC        DJNZ  DU2
63                  ;
64 9044 3A9593       LD   A,(PER)
65 9047 FE01         CP   1
66 9049 2812         JR   Z,DU4          ;Ecran
67 904B 2A9693       LD   HL,(I1)       ;Adresse a ecrire
68 904E 7C          LD   A,H            ;Poids fort à
69 904F CDCA91       CALL ECRIMP        ;Emission -> PRN
70 9052 7D          LD   A,L            ;Poids faible à
71 9053 CDCA91       CALL ECRIMP        ;Emission -> PRN
72 9056 3E20         LD   A,32
73 905B CD2BBD       CALL PRCHAR
74 905E 1810         JR   DU5
75                  DU4: EQU  $          ;Affichage adresse
76 905D 2A9593       LD   HL,(I1)       ;Adresse a afficher
77 9060 7C          LD   A,H            ;Poids fort à
78 9061 CDF191       CALL ECRECR        ;Emission -> Ecran
79 9064 7D          LD   A,L            ;Poids faible à
80 9065 CDF191       CALL ECRECR        ;Emission -> Ecran
81 9068 3E20         LD   A,32
82 906A CD5A3B       CALL WRCHAR
83                  DU5: EQU  $
84                  ,
85 906D AF          XOR  A
86 906E 329693       LD   (I2),A        ;Initialisation
87 9071 329993       LD   (I2+1),A      ;pointeur de ligne
88 9074 ED5B9893     LD   DE,(I2)
89                  DU6: EQU  $
90 907B 2A9693       LD   HL,(I1)
91 907B 19          ADD  HL,DE          ;Adresse donnee
92 907C 3A9593       LD   A,(PER)

```

```

90 907F FE01      CP      1
94 9081 2B0B      JR      Z,DU7          ;Affichage ecran
95 9083 7E        LD      A,(HL)        ;Impression
96 9084 CDCAD1     CALL    ECRIMP        ;sur PRN
97 9087 3E20      LD      A,32
98 9089 CD2BBD     CALL    PRCHAR        ;Espace sur PRN
99 908C 1B09      JR      DUB
100              DU7:    EQU      $          ;Affichage ecran
101 908E 7F        LD      A,(HL)
102 908F CDF191     CALL    ECRECR        ;Affichage
103 9092 3E20      LD      A,32
104 9094 CD5ABB     CALL    WRCHAR        ;Espace sur ecran
105              DUB:    EQU      $
106 9097 13        INC     DE
107 909B 7B        LD      A,E
108 9099 FE10      CP      16
109 909B 20DB      JR      NZ,DUG        ;Boucle de ligne
110 909D 1E00      LD      E,0
111              DU9:    EQU      $
112 909F 3A9593     LD      A,(PER)
113 90A2 FE01      CP      1
114 90A4 2B0C      JR      Z,DU9C        ;Ecran
115 90A6 3E20      LD      A,32
116 90A8 CD2BBD     CALL    PRCHAR
117 90AB 3E20      LD      A,32
118 90AD CD2BBD     CALL    PRCHAR
119 90B0 1B0A      JR      DU9B
120              DU9C:   EQU      $
121 90B2 3E20      LD      A,32
122 90B4 CD5ABB     CALL    WRCHAR
123 90B7 3E20      LD      A,32
124 90B9 CD5ABB     CALL    WRCHAR

```

Partie 9 : Programmes

```

125          DU9B:    EQU    $
126 90BC 2A9693      LD     HL, (I1)
127 90BF 19          ADD    HL, DE
128 90C0 7E          LD     A, (HL)
129 90C1 FE20         CP     32
130 90C3 380E        JR     C, DU11
131 90C5 FE90         CP     128
132 90C7 3807        JR     C, DU10
133 90C9 3A9593      LD     A, (PER)
134 90CC FE02         CP     2
135 90CE 2803        JR     Z, DU11
136          DU10:    EQU    $
137 90D0 7E          LD     A, (HL)
138 90D1 1B02        JR     DU12
139          DU11:    EQU    $
140 90D3 3E2E        LD     A, ". "
141          DU12:    EQU    $
142 90D5 47          LD     B, A
143 90D6 3A9593      LD     A, (PER)
144 90D9 FE01         CP     1
145 90DB 2806        JR     Z, DU13
146 90DD 78          LD     A, B                ;Vers Imprimante
147 90DE CD28BD      CALL    PRCHAR            ;Impression
148 90E1 1B04        JR     DU14
149          DU13:    EQU    $
150 90E3 78          LD     A, B                ;Vers ecran
151 90E4 CD5ABB      CALL    WRCHAR            ;Affichage
152          DU14:    EQU    $
153 90E7 13          INC     DE
154 90E8 78          LD     A, E
155 90E9 FE10         CP     16
156 90EB 20CF        JR     NZ, DU9B

```



```

157 90ED 3A9593      LD    A,(PER)
158 90F0 FE01      CP    1
159 90F2 280C      JR    Z,DU15
160 90F4 3E0D      LD    A,CR
161 90F6 CD2BBD     CALL PRCHAR
162 90F9 3E0A      LD    A,LF
163 90FB CD2BBD     CALL PRCHAR
164 90FE 180A      JR    DU16
165                DU15: EQU $
166 9100 3E0D      LD    A,CR
167 9102 CD5ABB     CALL WRCHAR
168 9105 3E0A      LD    A,LF
169 9107 CD5ABB     CALL WRCHAR
170                DU16: EQU $
171                ;
172 910A ED5B9393    LD    DE,(AFIN)
173 910E 37        SCF
174 910F 3F        CCF
175 9110 ED52      SBC    HL,DE
176 9112 CB7C      BIT    7,H
177 9114 CA4E91     JP    Z,DU3P          ;Fin de DUMP
178 9117 3A9593    LD    A,(PER)
179 911A FE02      CP    2
180 911C CA4191     JP    Z,DU2P
181 911F 3A9A93    LD    A,(LIGNE)
182 9122 FE17      CP    23
183 9124 C24191     JP    NZ,DU2P
184                ;
185 9127 3E0A      LD    A,LF
186 9129 CD5ABB     CALL WRCHAR          ; Line Feed
187 912C 210D93    LD    HL,TEXT
188 912F CD1892     CALL AFFICHE

```

Partie 9 : Programmes

```

189 9132 CD06BB      CALL WCHAR      ; "INKEY"
190 9135 3E02        LD    A,2
191 9137 CD0EBC      CALL SETMODE
192 913A AF          XOR    A
193 913B 329A93      LD     (LIGNE),A      ; Init cpt ligne
194 913E C33290      JP     DU1P          ; Boucle principale
195                DU2P:    EQU    $
196 9141 2A9693      LD     HL,(I1)
197 9144 111000      LD     DE,16
198 9147 19          ADD    HL,DE
199 914B 229693      LD     (I1),HL
200 914B C33290      JP     DU1P          ; Retour BP
201 914E C9          DUGP:    RET
202                ;-----
203                ;ENTREE: Saisie des donnees du
204                ;dump.
205                ;-----
206                ENTREE:    EQU    $
207 914F 3E01        LD     A,1
208 9151 CD0EBC      CALL SETMODE      ; MODE 1
209 9154 21B592      LD     HL,TEX1
210 9157 CD1892      CALL AFFICHE      ; Affichage texte 1
211 915A 21CA92      LD     HL,SOUS
212 915D CD1892      CALL AFFICHE      ; Affichage souligne
213 9160 21DF92      LD     HL,TEX2
214 9163 CD1892      CALL AFFICHE      ; Affichage texte 2
215 9166 3E04        LD     A,4
216 916B CD2192      CALL SAISIE      ; Saisie de 4 chiffres
217 916B CD6792      CALL CAH          ; Conversion ASCII->Hexa
218 916E 229193      LD     (AD),HL      ; Memo à depart
219 9171 3E0D        LD     A,OR
220 9173 CD5ABB      CALL WRCHAR      ; Affiche Carr. Return

```

```

221 9176 3E0A      LD   A,LF
222 9178 CD5ABB      CALL WRCHAR      ;Affiche Line Feed
223 917B 21EE92      LD   HL,TEX3
224 917E CD1892      CALL AFFICHE      ;Affichage texte 3
225 9181 3E04      LD   A,4
226 9183 CD2192      CALL SAISIE      ;Saisie de 4 chiffres
227 9186 CD6792      CALL CAH      ;Conversion ASCII->Hex.
228 9189 229393      LD   (AFIN),HL      ;Sauvegarde
229 918C 3E0D      LD   A,CR
230 918E CD5ABB      CALL WRCHAR      ;Affiche Carr. Return
231 9191 3E0A      LD   A,LF
232 9193 CD5ABB      CALL WRCHAR      ;Affiche Line Feed
233          PRO:    EQU   $      ;Boucle saisie periph
234 9196 213793      LD   HL,TEX6
235 9199 CD1892      CALL AFFICHE      ;Affichage texte 6
236 919C 214F93      LD   HL,TEX7
237 919F CD1892      CALL AFFICHE      ;Affichage texte 7
238 91A2 216D93      LD   HL,TEX8
239 91A5 CD1892      CALL AFFICHE      ;Affichage texte 8
240 91A8 3E01      LD   A,1
241 91AA CD2192      CALL SAISIE      ;Saisie du choix
242 91AD 3E0D      LD   A,CR
243 91AF CD5ABB      CALL WRCHAR      ;Carriage Return
244 91B2 3E0A      LD   A,LF
245 91B4 CD5ABB      CALL WRCHAR      ;Line Feed
246 91B7 3A7C93      LD   A,(BUFF)      ;Choix
247 91BA D630      SUB   30H      ;Conv. ASCII->Hexa
248 91BC FE01      CP   1
249 91BE 2806      JR   Z,PR1      ;
250 91C0 FE02      CP   2
251 91C2 2802      JR   Z,PR1      ;
252 91C4 1BD0      JR   PRO

```

Partie 9 : Programmes

```

253          PR1:      EQU  $          ;
254 91C6 329593          LD  (PER),A      ;Memorisation periph.
255 91C9 C9              RET
256          ;
257          ;
258          ;
259          ;-----
260          ;ECRIMP: Conversion HEXA->ASCII
261          ;des 2 caract. contenus dans A et
262          ;envoi des ces codes sur PRN
263          ;-----
264          ECRIMP:     EQU  $
265 91CA 47              LD   B,A          ;Sauvegarde
266 91CB CB3F           SRL  A
267 91CD CB3F           SRL  A
268 91CF CB3F           SRL  A
269 91D1 CB3F           SRL  A          ;Poids fort isole
270 91D3 FE0A           CP   10
271 91D5 3804           JR   C,EI1        ;Chiffre
272 91D7 C637           ADD  A,37H        ;Conv. ASCII lettre
273 91D9 1802           JR   EI2        ;Impression
274          EI1:      EQU  $
275 91DB C630           ADD  A,30H        ;Conv. ASCII chiffre
276          EI2:      EQU  $
277 91DD CD2BBD         CALL PRCHAR      ;Emission caractere
278 91E0 7B             LD   A,B          ;Restitution sauvegarde
279 91E1 E60F           AND  0FH        ; Poids faible isole
280 91E3 FE0A           CP   10
281 91E5 3804           JR   C,EI3        ;Chiffre
282 91E7 C637           ADD  A,37H        ;Conv. ASCII lettre
283 91E9 1802           JR   EI4        ;Impression
284          EI3:      EQU  $

```

```

285 91EB C630          ADD  A,30H          ;Conv. ASCII chiffre
286                  EI4:      EQU  $
287 91ED CD2BBD        CALL PRCHAR        ;Emission caractere
288 91F0 C9            RET
289                  ;
290                  ;
291                  ;
292                  ;-----
293                  ;ECRECR: Conversion des 2 caract.
294                  ;contenus dans A en ASCII et
295                  ;affichage sur l'ecran
296                  ;-----
297                  ECRECR:      EQU  $
298 91F1 47             LD   B,A            ;Sauvegarde
299 91F2 CB3F           SRL  A
300 91F4 CB3F           SRL  A
301 91F6 CB3F           SRL  A
302 91F8 CB3F           SRL  A            ;Octet fort isole
303 91FA FE0A          CP   10
304 91FC 3804           JR   C,EE1         ;Chiffre
305 91FE C637          ADD  A,37H          ;Conv. ASCII lettre
306 9200 1802           JR   EE2         ;Affichage
307                  EE1:      EQU  $
308 9202 CC7C          ADD  A,30H          ;Conv. ASCII chiffre
309                  EE2:      EQU  $
310 9204 CD5ABB        CALL WRCHAR        ;Affichage
311 9207 7B            LD   A,B
312 9208 E60F          AND  0FH          ;Octet faible isole
313 920A FE0A          CP   10
314 920C 3804           JR   C,EE3         ;Chiffre
315 920E C637          ADD  A,37H          ;Conv. ASCII lettre
316 9210 1802           JR   EE4         ;Affichage

```

```

017          EE3:      EQU  $
018 0212 0630          ADD  A,30H          ;Conv. ASCII chiffre
019          EE4:      EQU  $
020 0214 0D5ABB        CALL WRCHAR        ;Affichage
021 0217 09           RET
022          ;-----
023          ;AFFICHE: Affichage de donnees
024          ;alphanumeriques. Termineur: 0
025          ;Entree: HL = Adresse 1er caract.
026          ;-----
027          AFFICHE:   EQU  $
028 0218 7C           LD   A,(HL)
029 0219 B7           OR   A
030 021A 08           RET  Z          ;Fin d'affichage
031 021B 0D5ABB        CALL WRCHAR
032 021E 23           INC  HL
033 021F 18F7          JR   AFFICHE
034          ;
035          ;
036          ;-----
037          ;SAISIE: saisie de caracteres,
038          ;stockage dans le buffer buff.
039          ;Entree: A = Nombre de caracteres
040          ;-----
041          SAISIE:     EQU  $
042 0221 329093         LD   (S1),A      ;Sauvegarde Nb caract.
043 0224 217C9D         LD   HL,BUFF    ;Zone de memorisation
044 0227 47           LD   B,A          ;Nombre de caract. a lire
045          SAI2:      EQU  $
046 0228 3E5F          LD   A,"_"
047 022A 0D5ABB        CALL WRCHAR      ;Affich. curseur
048 022D 3E0B          LD   A,BS

```

```

349 922F CD5ABB      CALL WRCHAR      ;Curseur en arriere
350 9232 CD06BB      CALL WCHAR      ;Lecture 1 caractere
351 9235 FE7F        CP    DEL      ;Caractere DEL ?
352 9237 2023        JR    NZ,SAI3   ;Non
353 9239 3A9093      LD    A,(S1)
354 923C B8          CP    B
355 923D 28E9        JR    Z,SAI2    ;Aucune action
356                  ;Touche DEL interdite
357 923F 3E20        LD    A,32
358 9241 CD5ABB      CALL WRCHAR      ;Effacement curseur
359 9244 3E08        LD    A,BS
360 9246 CD5ABB      CALL WRCHAR      ;Curseur en arriere
361 9249 3E08        LD    A,BS
362 924B CD5ABB      CALL WRCHAR      ;Curseur en arriere
363 924E 3E5F        LD    A,"_"
364 9250 CD5ABB      CALL WRCHAR      ;Curseur
365 9253 3E08        LD    A,BS
366 9255 CD5ABB      CALL WRCHAR      ;Curseur en arriere
367 9258 2B          DEC    HL
368 9259 04          INC    B      ;Reajustement des pointeu
369 925A 180C        JR    SAI2      ;Poursuite de la saisie
370                  SAI0: EQU    $
371 925C CD5ABB      CALL WRCHAR      ;Affichage caractere
372 925F 77          LD    (HL),A
373 9260 05          DEC    B
374 9261 78          LD    A,B
375 9262 B7          OR     A
376 9263 0B          RET    Z      ;Fin de saisie
377 9264 23          INC    HL      ;Memoire suivante
378 9265 18C1        JR    SAI2      ;Boucle de saisie
379                  ;

```

```

380      ; -----
381      ;CAH: Conversion Hexa  -> ASCII
382      ;des 4 premiers caracteres du
383      ;buffer BUFF. Resultat dans HL
384      ;-----
385      CAH:      EQU  $
386  9267 3A7C93      LD  A,(BUFF)      ;1er octet
387  926A FE3A      CP   3AH
388  926C 3804      JR   C,CAH1      ;Chiffre
389  926E D637      SUB  37H
390  9270 1802      JR   CAH2
391      CAH1:      EQU  $
392  9272 D630      SUB  30H
393      CAH2:      EQU  $
394  9274 CB27      SLA  A
395  9276 CB27      SLA  A
396  9278 CB27      SLA  A
397  927A CB27      SLA  A      ;* 16
398  927C 6F      LD   L,A      ;Sauvegarde 1ere conv.
399  927D 3A7D93      LD  A,(BUFF+1)      ;2eme octet
400  9280 FE3A      CP   3AH
401  9282 3804      JR   C,CAH3      ;Chiffre
402  9284 D637      SUB  37H
403  9286 1802      JR   CAH4
404      CAH3:      EQU  $
405  9288 D630      SUB  30H
406      CAH4:      EQU  $
407  928A 85      ADD  A,L      ;Addition au poids fort
408  928B 6F      LD   L,A      ;Sauvegarde
409  928C 3A7E93      LD  A,(BUFF+2)      ;3eme octet
410  928F FE3A      CP   3AH
411  9291 3804      JR   C,CAH5      ;Chiffre

```



```

412 9293 D637          SUB  37H
413 9295 1B02          JR   CAH6
414                   CAH5: EQU  $
415 9297 D630          SUB  30H
416                   CAH6: EQU  $
417 9299 CB27          SLA  A
418 929B CB27          SLA  A
419 929D CB27          SLA  A
420 929F CB27          SLA  A           ;* 16
421 92A1 67           LD   H,A
422 92A2 3A7F93       LD   A,(BUFF+3)   ;4eme octet
423 92A5 FE3A         CP   3AH
424 92A7 3B04         JR   C,CAH7       ;Chiffre
425 92A9 D637          SUB  37H
426 92AB 1B02          JR   CAH8
427                   CAH7: EQU  $
428 92AD D630          SUB  30H
429                   CAH8: EQU  $
430 92AF B4           ADD  A,H           ;Addition poids fort
431 92B0 67           LD   H,A           ;HL=Conversion 16 bits
432 92B1 EB           EX   DE,HL
433 92B2 63           LD   H,E
434 92B3 6A           LD   L,D
435 92B4 C9           RET
436                   ;
437                   ;
438                   ;-----
439                   ; Zone des chaines du programme
440                   ;-----
441 92B5 55746960 TEX1: DB   "Utilitaire d. "
441 92B9 69746169
441 92BD 72652064

```

```
441 92C1 6520
442 92C3 44554D50          DB  "DUMP",ODH,0AH,0
442 92C7 0D0A00
443 92CA 2D2D2D2D SOU1:    DB  "-----"
443 92CE 2D2D2D2D
443 92D2 2D2D2D2D
443 92D6 2D2D
444 92DB 2D2D2D2D          DB  "-----",ODH,0AH,0
444 92DC 0D0A00
445 92DF 41207061 TEX2:    DB  "A partir de : ",0
445 92E3 72746972
445 92E7 20646520
445 92EB 3A2000
446 92EE 6A757371 TEX3:    DB  "jusqu'a : ",0
446 92F2 75276120
446 92F6 3A2000
447 92F9 496D7072 TEX4:    DB  "Impression en "
447 92FD 65737369
447 9301 6F6E2065
447 9305 6E20
448 9307 636F7572          DB  "cours",0
448 930B 7300
449 930D 41707075 TEX5:    DB  "Appuyez sur une "
449 9311 79657A20
449 9315 73757220
449 9319 756E6520
450 931D 746F7563          DB  "touche pour voir
450 9321 6B652070
450 9325 6F757220
450 9329 766F6972
450 932D 20
451 932E 6C612073          DB  "la suite",0
```

```

451 9332 75697465
451 9336 00
452 9337 31294166 TEX6:      DB   "1)Affichage sur "
452 933B 66696368
452 933F 61676520
452 9343 70757220
453 9347 65637261      DB   "ecran",ODH,0AH,0
453 934B 6E0D0A00
454 934F 3229496D TEX7:      DB   "2)Impression sur
454 9353 70726572
454 9357 73696F6E
454 935B 20737572
454 935F 20
455 9360 696D7072      DB   "imprimante",ODH,0
455 9364 696D616E
455 9368 74650D0A
455 936C 00
456 936D 566F7472 TEX8:      DB   "Votre choix : ",0
456 9371 65206368
456 9375 6F697820
456 9379 3A2000
457      ;
458      BUFF:      DS    20      ;Zone buffer
459      S1:      DS    1      ;Sauvegarde Nb caract
460      AD:      DS    2      ;Sauvegarde à depart
461      AFIN:      DS    2      ;Sauvegarde à fin
462      PER:      DS    1      ;Sauvegarde periph
463      I1:      DS    2      ;Index
464      I2:      DS    2      ;Index
465      LIGNE:      DS    1      ;Compteur de ligne
466      END

```

Ci-dessous, nous donnons un exemple d'impression obtenu entre les adresses #6000 et #610F.

```

6000 4F 52 47 20 39 30 30 30 4B 0D 4C 4F 41 44 20 39   ORG 9000H.LOAD 9
6010 30 30 30 4B 0D 3B 2D 2D 2D 2D 2D 2D 2D 2D 2D   000H.;-----
6020 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D   -----
6030 2D 2D 2D 2D 2D 2D 2D 0D 3B 5A 6F 6E 65 20 64 65   -----.;Zone de
6040 73 20 63 6F 6E 73 74 61 6E 74 65 73 20 64 75 20   s constantes du
6050 70 72 6F 67 72 61 6D 6D 65 0D 3B 2D 2D 2D 2D 2D   programme.;-----
6060 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D   -----
6070 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 0D 53 45 54   -----;SET
6080 4D 4F 44 45 3A 20 45 51 55 20 30 42 43 30 45 4B   MODE: EQU 0BC0EH
6090 20 3B 20 53 43 52 20 53 45 54 20 4D 4F 44 45 0D   ; SCR SET MODE.
60A0 57 52 43 4B 41 52 3A 20 45 51 55 20 30 42 42 35   WRCHAR: EQU 0BB5
60B0 41 4B 20 3B 54 5B 54 20 4F 55 54 50 55 54 0D 57   AH ;TXT OUTPUT.W
60C0 43 4B 41 52 3A 20 45 51 55 20 30 42 42 30 36 4B   CHAR: EQU 0BB06H
60D0 20 3B 20 4B 4D 20 57 41 49 54 20 43 4B 41 52 0D   ; KM WAIT CHAR.
60E0 50 52 43 4B 41 52 3A 20 45 51 55 20 30 42 44 32   PRCHAR: EQU 0BD2
60F0 42 4B 20 3B 4D 43 20 50 52 49 4E 54 20 43 4B 41   BH ;MC PRINT CHA
6100 52 0D 42 53 3A 20 45 51 55 20 3B 20 3B 42 61 63   R.BS: EQU B ;Bac

```

Dump sur imprimante avec le programme ASM de Dump de 6000 à 6100

Il est également possible (pour aller plus vite) de saisir les codes hexadécimaux correspondant au programme assembleur sous BASIC.

Le listing du programme est alors le suivant :

```

1000 FOR I=&9000 TO &937B
1010   READ A$ 'Lecture 1 donnée
1020   B$="&"+A$
1030   B=VAL(B$)
1040   POKE I,B 'Mise en mémoire
1050 NEXT I
1060 END

```

```

1070 '-----
1080 'Codes operatoires
1090 '-----
1100 DATA CD,4F,91,CD,07,90,C9,3E,02,CD,0E,BC,AF,32,9A,93
1110 DATA 3A,95,93,FE,01,28,15,3E,0A,CD,5A,BB,3E,0A,CD,5A
1120 DATA BB,3E,0A,CD,5A,BB,21,F9,92,CD,18,92,2A,91,93,22
1130 DATA 96,93,3A,9A,93,3C,32,9A,93,21,7C,93,3E,00,06,10
1140 DATA 77,23,10,FC,3A,95,93,FE,01,28,12,2A,96,93,7C,CD
1150 DATA CA,91,7D,CD,CA,91,3E,20,CD,2B,BD,18,10,2A,96,93
1160 DATA 7C,CD,F1,91,7D,CD,F1,91,3E,20,CD,5A,BB,AF,32,9B
1170 DATA 93,32,99,93,ED,5B,9B,93,2A,96,93,19,3A,95,93,FE
1180 DATA 01,28,0B,7E,CD,CA,91,3E,20,CD,2B,BD,18,09,7E,CD
1190 DATA F1,91,3E,20,CD,5A,BB,13,7B,FE,10,20,0B,1E,00,3A
1200 DATA 95,93,FE,01,28,0C,3E,20,CD,2B,BD,3E,20,CD,2B,BD
1210 DATA 18,0A,3E,20,CD,5A,BB,3E,20,CD,5A,BB,2A,96,93,19
1220 DATA 7E,FE,20,3B,0E,FE,80,3B,07,3A,95,93,FE,02,28,03
1230 DATA 7E,1B,02,3E,2E,47,3A,95,93,FE,01,28,06,7B,CD,2B
1240 DATA BD,18,04,7B,CD,5A,BB,13,7B,FE,10,20,CF,3A,95,93
1250 DATA FE,01,28,0C,3E,0D,CD,2B,BD,3E,0A,CD,2B,BD,18,0A
1260 DATA 3E,0D,CD,5A,BB,3E,0A,CD,5A,BB,ED,5B,93,93,37,3F
1270 DATA ED,52,CB,7C,CA,4E,91,3A,95,93,FE,02,CA,41,91,3A
1280 DATA 9A,93,FE,17,C2,41,91,3E,0A,CD,5A,BB,21,0D,93,CD
1290 DATA 18,92,CD,06,BB,3E,02,CD,0E,BC,AF,32,9A,93,C3,32
1300 DATA 90,2A,96,93,11,10,00,19,22,96,93,C3,32,90,C9,3E
1310 DATA 01,CD,0E,BC,21,B5,92,CD,18,92,21,CA,92,CD,18,92
1320 DATA 21,DF,92,CD,18,92,3E,04,CD,21,92,CD,67,92,22,91
1330 DATA 93,3E,0D,CD,5A,BB,3E,0A,CD,5A,BB,21,EE,92,CD,1B
1340 DATA 92,3E,04,CD,21,92,CD,67,92,22,93,93,3E,0D,CD,5A
1350 DATA BB,3E,0A,CD,5A,BB,21,37,93,CD,18,92,21,4F,93,CD
1360 DATA 18,92,21,6D,93,CD,18,92,3E,01,CD,21,92,3E,0D,CD
1370 DATA 5A,BB,3E,0A,CD,5A,BB,3A,7C,93,D6,D0,FE,01,28,05
1380 DATA FE,02,28,02,1B,D0,32,95,93,C9,47,CB,3F,CB,3F,CB

```

Partie 9 : Programmes

1390 DATA 3F, CB, 3F, FE, 0A, 3B, 04, C6, 37, 1B, 02, C6, 30, CD, 2B, BD
1400 DATA 7B, E6, 0F, FE, 0A, 3B, 04, C6, 37, 1B, 02, C6, 30, CD, 2B, BD
1410 DATA C9, 47, CB, 3F, CB, 3F, CB, 3F, CB, 3F, FE, 0A, 3B, 04, C6, 37
1420 DATA 1B, 02, C6, 30, CD, 5A, BB, 7B, E6, 0F, FE, 0A, 3B, 04, C6, 37
1430 DATA 1B, 02, C6, 30, CD, 5A, BB, C9, 7E, B7, CB, CD, 5A, BB, 2B, 1B
1440 DATA F7, 32, 90, 93, 21, 7C, 93, 47, 3E, 5F, CD, 5A, BB, 3E, 0B, CD
1450 DATA 5A, BB, CD, 06, BB, FE, 7F, 20, 23, 3A, 90, 93, BB, CB, E9, 3E
1460 DATA 20, CD, 5A, BB, 3E, 0B, CD, 5A, BB, 3E, 0B, CD, 5A, BB, 3E, 5F
1470 DATA CD, 5A, BB, 3E, 0B, CD, 5A, BB, 2B, 04, 1B, CC, CD, 5A, BB, 77
1480 DATA 05, 7B, B7, CB, 2B, 1B, C1, 3A, 7C, 93, FE, 3A, 3B, 04, D6, 37
1490 DATA 1B, 02, D6, 30, CB, 27, CB, 27, CB, 27, CB, 27, 6F, 3A, 7D, 93
1500 DATA FE, 3A, 3B, 04, D6, 37, 1B, 02, D6, 30, B5, 6F, 3A, 7E, 93, FE
1510 DATA 3A, 3B, 04, D6, 37, 1B, 02, D6, 30, CB, 27, CB, 27, CB, 27, CB
1520 DATA 27, 67, 3A, 7F, 93, FE, 3A, 3B, 04, D6, 37, 1B, 02, D6, 30, B5
1530 DATA 67, EB, 63, 6A, C9, 55, 74, 69, 6C, 69, 74, 61, 69, 72, 65, 70
1540 DATA 64, 65, 20, 44, 55, 4D, 50, 0D, 0A, 00, 2D, 2D, 2D, 2D, 2D, 2D
1550 DATA 2D, 2D, 2D, 2D, 2D, 2D, 2D, 2D, 2D, 2D, 2D, 2D, 0D, 0A, 00, 41
1560 DATA 20, 70, 61, 72, 74, 69, 72, 20, 64, 65, 20, 3A, 20, 00, 6A, 75
1570 DATA 73, 71, 75, 27, 61, 20, 3A, 20, 00, 49, 6D, 70, 72, 65, 73, 70
1580 DATA 69, 6F, 6E, 20, 65, 6F, 20, 63, 6F, 75, 72, 73, 00, 41, 70, 70
1590 DATA 75, 79, 65, 7A, 20, 73, 75, 72, 20, 75, 6E, 65, 20, 74, 6F, 75
1600 DATA 63, 6B, 65, 20, 70, 6F, 75, 72, 20, 76, 6F, 69, 72, 20, 6C, 61
1610 DATA 20, 73, 75, 69, 74, 65, 00, 31, 29, 41, 66, 66, 69, 63, 6B, 61
1620 DATA 67, 65, 20, 73, 75, 72, 20, 65, 63, 72, 61, 6E, 0D, 0A, 00, 32
1630 DATA 29, 49, 6D, 70, 72, 65, 73, 73, 69, 6F, 6E, 20, 73, 75, 72, 20
1640 DATA 69, 6D, 70, 72, 69, 6D, 61, 6E, 74, 65, 0D, 0A, 00, 5b, 6F, 74
1650 DATA 72, 65, 20, 63, 6B, 6F, 69, 7B, 20, 3A, 20, 00, 00, 00, 00, 00

Si vous décidez d'entrer le programme assembleur sous Basic, vérifiez les codes entrés grâce au programme de checksum (voir Partie 9, chap. 8.4).

Pour cela, tapez « MERGE » suivi du nom sous lequel vous avez sauvegardé le programme de checksum. Exécutez le programme de checksum en tapant « RUN 50000 ». Les données de vérification sont les suivantes :

C6 3D 7F B4 E3 95 59 38 5F B7 87 14 32 4F 27 57 42 6F 95 19 F9
72 4B 77 DA 1E 1F C1 62 55 79 80 A6 DC 5C CE F5 7D C8 A7 E4
4A 5 2B 47 76 F8 43 AB 2D E8 4B BC F1 8B 8F

Si une ou plusieurs des données de vérification ne correspondent pas avec celles données ci-dessus, vérifiez la ligne correspondante.

Remarque :

Plusieurs données ont été rajoutées en fin de listing afin d'assurer la compatibilité avec le programme de checksum.

9/8.6

Récupération d'un fichier effacé par la commande IERA

Vous connaissez sans doute l'instruction **IERA**, "**nomfic.ext**" du Basic Amstrad qui vous permet d'effacer un fichier sur une disquette. Lorsqu'une telle commande est activée, le fichier spécifié n'est pas supprimé physiquement de la disquette: il est seulement marqué « absent » dans le catalogue, et, de ce fait, il n'apparaît plus lorsque vous demandez le répertoire de la disquette.

Partant de cette remarque, il vient tout de suite à l'esprit qu'un fichier supprimé par la commande IERA peut être restitué en effaçant la marque « absent » du répertoire. C'est effectivement le cas si aucune opération d'écriture sur la disquette n'a été faite depuis l'effacement. Effectivement, les commandes d'écriture sur disquettes sont autorisées à écrire sur les fichiers marqués « absent » (sinon, à quoi la commande IERA servirait-elle ?).

Voyons en détails la façon de procéder pour récupérer un fichier effacé.

Définitions

- On appelle *secteur* un certain nombre d'octets consécutifs situés sur la disquette (512 octets pour Amstrad CPC). Un secteur est la quantité minimale d'informations accédée à chaque lecture sur disquette ou disque dur.
- Sur CPC, un secteur est divisé en quatre zones de taille égale appelées *enregistrements* (chaque zone fait donc 128 octets).
- Un *bloc* est un ensemble de secteurs qui occupent une taille de 1 Koctets ou 2 Koctets. Les blocs permettent une gestion plus aisée des fichiers de taille importante. La taille des blocs est fixée sous CP/M.

Dans le répertoire figurent les numéros des blocs occupés par chaque fichier.

Sachant que :

- la taille standard d'un bloc sous CP/M est de 1 Koctets,
- chaque fichier occupe au minimum un bloc,

il est facile de conclure que chaque fichier, même s'il ne contient qu'un caractère, occupera une taille minimale de 1 Koctet sur la disquette.

Structure détaillée du catalogue

Une des fonctions fondamentales d'un système d'exploitation, quel qu'il soit et quelles que soient ses origines, est de faciliter la gestion des fichiers sur les supports de sauvegarde de masse (lecteurs de disquettes ou de disques durs). Pour ce faire, les systèmes d'exploitation font appel à un répertoire (souvent appelé catalogue sur les ordinateurs CPC) qui contient les informations nécessaires pour retrouver rapidement les fichiers sur le support mémoire. Ces informations sont les suivantes :

- nom des fichiers,
- emplacement des fichiers sur le support.

Examinons en détails les données enregistrées dans le répertoire pour chaque fichier. Par la suite, nous appellerons *entrées* l'ensemble des données permettant d'accéder à un fichier dans le répertoire.

Chaque entrée comporte 32 octets dont voici la signification :

00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F

Octet 00 : Numéro de USER du fichier (00 à 0F), ou indicateur de fichier « absent » (effacé par une commande IERA). Lorsqu'un fichier a été effacé, cet octet contient la valeur #E5.

Octets 01 à 08 : Nom du fichier, complété par des caractères « Espace » (#20) si nécessaire (si la longueur du nom est inférieure à 8 caractères).

Octets 09 à 0B : Nom de l'extension.

Octet 0C : Numéro de l'extension (Cf. Octet 0F).

Octets 0D à 0E : 0

Octet 0F : Nombre d'enregistrements du fichier.
Si cet octet vaut #80 (128), une « extension » suit car une entrée ne peut référencer que 16 Koctets (effectivement, une entrée ne peut comporter que 16 numéros de blocs, donc 16 Koctets. (Cf. Octets 10 à 1F.)

Octets 10 à 1F : Numéros de blocs occupés par le fichier.

Précisons encore que la position du répertoire dépend du format de la disquette. Une disquette peut être formatée en « Système », « Data » ou « Ibm ».

— si la disquette est au format système, le catalogue se trouvera sur la piste 2, secteurs 65 à 69 ;

— si la disquette est au format data, le catalogue se trouvera sur la piste 0, secteurs 193 à 197 ;

— si la disquette est au format Ibm, le catalogue se trouvera sur la piste 1, secteurs 1 à 5.

Ainsi l'octet 0 d'une entrée contiendra la valeur #E5 si ce fichier a été effacé, ou un numéro d'USER. Pour qu'un tel fichier soit à nouveau accessible, il suffira de charger l'octet 0 de l'entrée correspondante avec la valeur 0.

Utilisation des instructions « KL FIND COMMAND », « READ SECTOR » et « WRITE SECTOR »

Instruction « READ SECTOR » (&84) du lecteur de disquettes

Une des instructions cachées du lecteur de disquette est « READ SECTOR ».

Cette instruction permet d'accéder à un secteur quelconque de la disquette. Nous allons l'utiliser pour lire la valeur d'une entrée dans le répertoire.

Instruction « WRITE SECTOR » (&85) du lecteur de disquettes

Une autre instruction cachée du lecteur de disquettes est « WRITE SECTOR ».

Nous allons l'utiliser pour réécrire l'entrée lue par READ SECTOR dans le répertoire disquette, en ayant pris le soin de rectifier la valeur contenue dans le premier octet de l'entrée.

Accès aux instructions « READ SECTOR » et « WRITE SECTOR » à travers la macro instruction KL FIND COMMAND

L'instruction « KL FIND COMMAND » du Firmware est située à l'adresse #BCD4. Elle permet de trouver l'adresse d'une instruction de type RSX ou d'une commande externe de la ROM basse de l'Amstrad.

Elle permet, en outre, de trouver les adresses des commandes de lecture et d'écriture « READ SECTOR » et « WRITE SECTOR ». L'instruction « KL FIND COMMAND » combinée à un RESTART 24 permettra d'activer la lecture ou l'écriture d'un secteur de la disquette.

Interfaçage de la routine KL FIND COMMAND

L'adresse en mémoire où se trouve le nom de la commande à accéder doit être fournie en entrée de cette routine dans le registre HL.

En sortie, si la commande indiquée est trouvée :

- l'indicateur « Carry » est positionné à un ;
- le registre C contient l'adresse de ROM SELECT ;
- le registre HL contient l'adresse de la routine.

Si la commande indiquée n'est pas trouvée :

- l'indicateur « Carry » est positionné à zéro ;
- les registres C et HL contiennent des valeurs sans signification.

Dans tous les cas :

- les registres A, B et DE sont effacés ;
- les autres registres sont intacts.

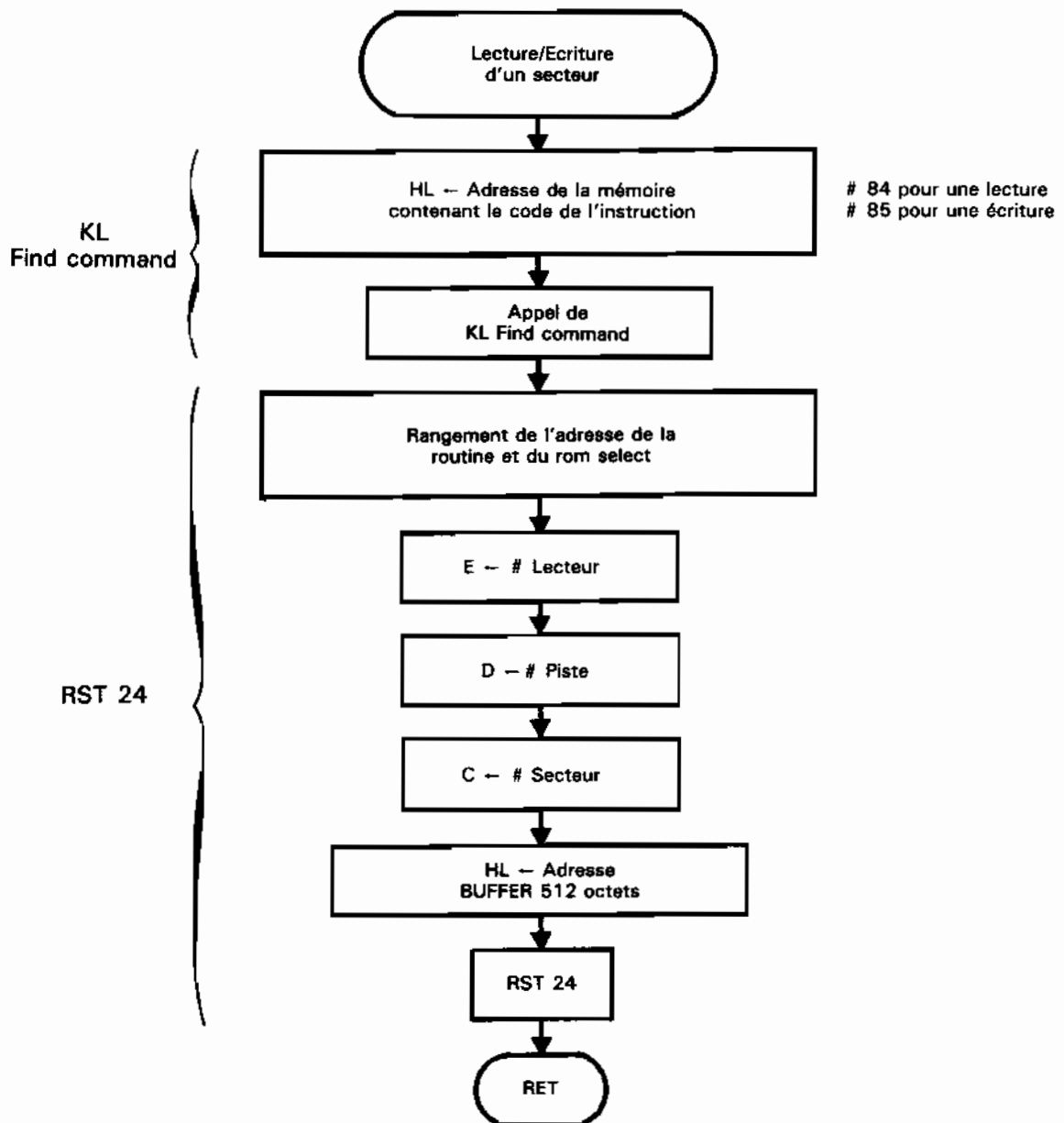
Interfaçage du RESTART 24

Les registres C, D, E et HL sont utilisés en entrée. Ils doivent contenir les informations suivantes :

- E = Numéro du lecteur de disquettes,
- D = Numéro de la piste à accéder,
- C = Numéro du secteur à accéder,
- HL = Adresse d'un buffer de 512 octets nécessaire à la lecture ou à l'écriture d'un secteur.

Nous pouvons maintenant réaliser assez facilement des programmes en assembleur de lecture et d'écriture d'un secteur de la disquette.

Ces deux programmes respecteront la logique suivante :



Algorithme : Lecture / Ecriture d'un secteur sur disquette

Lecture d'un secteur sur disquette

```

1          ORG 8000H
2          LOAD 8000H
3          ;
4          ;-----
5          ; Definition des constantes
6          ;-----
7          ;
8          FIND:      EQU 0BCD4H          ;KL FIND COMMAND
9          ;
10         ;-----
11         ; Definition des variables
12         ;-----
13         ;
14         AD:         DS 3
15         BUFF:       DS 512             ;Buffer lect/ecrit
16         MAX:        DS 1
17         DRIVE:      DS 1               ;Nom du drive
18         PISTE:      DS 1               ;Numero de piste
19         SECT:       DS 1               ;Numero de secteur
20 8207 84  LIT:       DP 84H             ;Instr. lecture
21 8208 85  ECRIT:     DB 85H             ;Instr. ecriture
22         ;
23         ;-----
24         ; Lecture d'un secteur
25         ;-----
26         ;Entree: Lecteur, piste et secteur
27         ;Sortie: Secteur lu dans BUFF
28         ;-----
29         ;
30         LECT:       EQU $              ;Point d'entree

```

31 8209 210782	LD HL,LIT	;Lecture disque
32 820C CDD4BC	CALL FIND	;KL FIND COMMAND
33 820F 220080	LD (AD),HL	
34 8212 79	LD A,C	
35 8213 320280	LD (AD+2),A	
36 8216 3A0482	LD A,(DRIVE)	
37 8219 5F	LD E,A	;No de drive
38 821A 3A0582	LD A,(PISTE)	
39 821D 57	LD D,A	;No de piste
40 821E 3A0682	LD A,(SECT)	
41 8221 4F	LD C,A	;No de secteur
42 8222 210380	LD HL,BUFF	
43 8225 DF	RST 24	;Activ. instruction en R0
44 8226 0080	DW AD	
45 8228 C9	RET	
46	END	

Ecriture d'un secteur sur disquette

```

1          ORG 8000H
2          LOAD 8000H
3          ;
4          ;-----
5          ; Definition des constantes
6          ;-----
7          ;
8          FIND:      EQU 0BCD4H          ;KL FIND COMMAND
9          ;
10         ;-----
11         ; Definition des variables
12         ;-----
13         ;
14         AD:         DS 3
15         BUFF:       DS 512             ;Buffer lect/ecrit
16         MAX:        DS 1
17         DRIVE:      DS 1              ;Nom du drive
18         PISTE:      DS 1              ;Numero de piste
19         SECT:       DS 1              ;Numero de secteur
20 8207 84    LIT:      DB 84H           ;Instr. lecture
21 8208 85    ECRIT:    DB 85H           ;Instr. ecriture
22         ;
23         ;-----
24         ; Ecriture d'un secteur
25         ;-----
26         ;Entree: BUFF = buffer a ecrire
27         ;Sortie: Ecriture BUFF sur disq.
28         ;-----
29         ;

```

```

30          ECR:      EQU  $              ;Point d'entree
31 8209 210882      LD   HL,ECRIT          ;Ecriture disque
32 820C CDD4BC      CALL FIND              ;KL FIND COMMAND
33 820F 220080      LD   (AD),HL
34 8212 79          LD   A,C
35 8213 320280      LD   (AD+2),A
36 8216 3A0482      LD   A,(DRIVE)
37 8219 5F          LD   E,A              ;No de drive
38 821A 3A0582      LD   A,(PISTE)
39 821D 57          LD   D,A              ;No de piste
40 821E 3A0682      LD   A,(SECT)
41 8221 4F          LD   C,A              ;No de secteur
42 8222 210380      LD   HL,BUFF
43 8225 DF          RST  24              ;Activ. instruction en Rf
44 8226 0080      DW   AD
45 8228 C9          RET
46          END

```

Ces programmes élémentaires vont être agrémentés d'une interface utilisateur qui permettra d'entrer le nom et l'extension du fichier effacé à retrouver. Ce fichier sera cherché sur la disquette spécifiée, et le premier octet de son entrée dans le répertoire sera modifié si nécessaire. Si le fichier spécifié n'est pas trouvé dans le répertoire, le message 'Fichier non trouvé' sera affiché à l'écran.

Le programme de restitution est présenté dans deux versions : totalement assembleur et basic plus assembleur.

Interface utilisateur

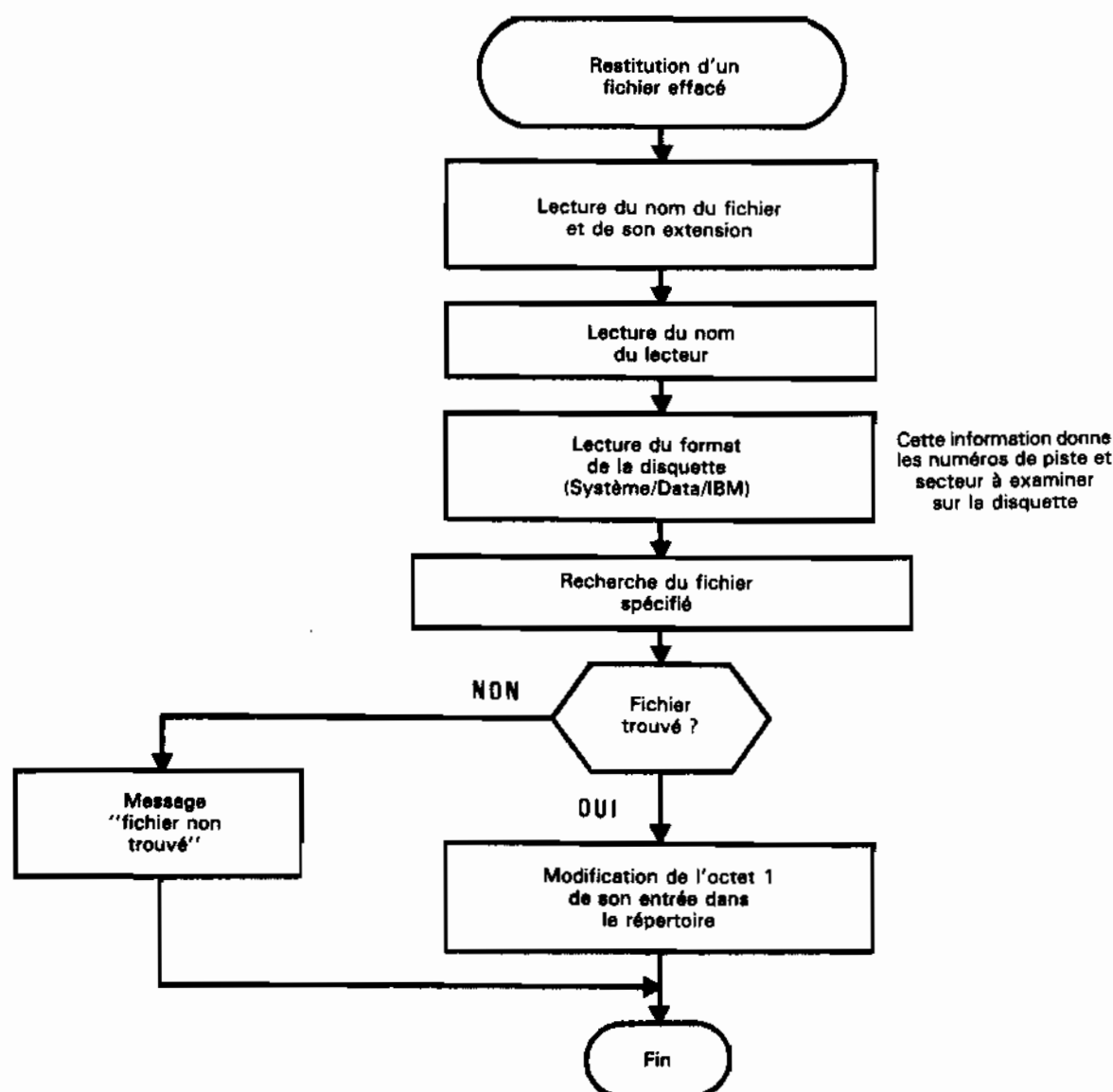
Cette interface consiste à lire :

- le nom et l'extension du fichier à retrouver ;
- le nom du lecteur de disquette (A ou B) ;
- le format de la disquette (Système, donnée ou lbn).

Le fichier est recherché dans la totalité du catalogue. Si le fichier spécifié est rencontré dans le catalogue, le premier octet de l'entrée correspondante est comparé à #E5. S'il est égal à cette valeur, il est mis à 0 pour être accessible sous un numéro d'USER quelconque.

Si le fichier spécifié n'est pas trouvé dans le catalogue ou s'il est trouvé mais avec le premier octet de son entrée différent de #E5, le message 'Fichier non trouvé' est affiché et le répertoire est laissé intact. Ces diverses actions sont résumées dans l'algorithme suivant.

Restitution d'un fichier effacé



Programme de restitution de fichiers en version Basic

• Les ordres « **KL FIND COMMAND** », « **READ SECTOR** » et « **WRITE SECTOR** » ne peuvent pas être utilisés facilement en Basic, c'est pourquoi nous allons développer un petit programme en assembleur pour lire et écrire un secteur quelconque de la disquette.

```

1          ORG 9000H
2          LOAD 9000H
3          ;
4          ;-----
5          ; Definition des constantes
6          ;-----
7          ;
8          FIND:      EQU 0BCD4H          ;KL FIND COMMAND
9          ;
10         ;-----
11         ; Definition des variables
12         ;-----
13         ;
14         AD:        DS 3
15         BUFF:      DS 512              ;Buffer lect/ecrit
16         MAX:       DS 1
17         DRIVE:     DS 1                ;Nom du drive
18         PISTE:     DS 1                ;Numero de piste
19         SECT:      DS 1                ;Numero de secteur
20         ;
21         ORG 9217H
22         LOAD 9217H
23 9217 84          LIT:      DB 84H          ;Instr. lecture
24 9218 85          ECRIT:    DB 85H          ;Instr. ecriture
25         ;
26         ORG 9375H
27         LOAD 9375H

```

```

28      ;
29      ;-----
30      ; Ecriture d'un secteur
31      ;-----
32      ;Entree: BUFF = buffer a ecrire
33      ;Sortie: Ecriture BUFF sur disq.
34      ;-----
35      ;
36      FCR:      EQU  $      ;Point d'entree
37 9375 211B92      LD  HL,ECRIT      ;Ecriture disque
38 937B CDD4BC      CALL FIND      ;KL FIND COMMAND
39 937B 220090      LD  (AD),HL
40 937E 79          LD  A,C
41 937F 320290      LD  (AD+2),A
42 9382 3A0492      LD  A,(DRIVE)
43 9385 5F          LD  E,A      ;No de drive
44 9386 3A0592      LD  A,(PISTE)
45 9389 57          LD  D,A      ;No de piste
46 938A 3A0692      LD  A,(SECT)
47 938D 4F          LD  C,A      ;No de secteur
48 938E 210390      LD  HL,BUFF
49 9391 DF          RST  24      ;Activ. instruction en R0
50 9392 0090      DW  AD
51 9394 09          RET
52      ;
53      ;-----
54      ; Lecture d'un secteur
55      ;-----
56      ;Entree: PISTE et SECTeur a lire
57      ;Sortie: secteur lu dans BUFF
58      ;-----

```

```
59          ;
60          LECT:      EQU  $           ;Point d'entree
61 9395 211792        LD  HL,LIT        ;Lecture disque
62 9398 CDD4BC        CALL FIND         ;KL FIND COMMAND
63 939B 220090        LD  (AD),HL
64 939E 79           LD  A,C
65 939F 320290        LD  (AD+2),A
66 93A2 3A0492        LD  A,(DRIVE)
67 93A5 5F           LD  E,A           ;No de drive
68 93A6 3A0592        LD  A,(PISTE)
69 93A9 57           LD  D,A           ;No de piste
70 93AA 3A0692        LD  A,(SECT)
71 93AD 4F           LD  C,A           ;No de secteur
72 93AE 210390        LD  HL,BUFF
73 93B1 DF           RST  24           ;Activ. instruction en RD
74 93B2 0090        DW  AD
75 93B4 C9           RET
76                END
```

Ce programme en Assembleur est en effet une concaténation des deux précédents. Il est implanté en #9375 pour être identique au programme de restitution écrit totalement en Assembleur présenté par la suite.

- Le programme Basic de restitution est relativement court, mais assez lent, c'est pourquoi on pourra référer une version assembleur.

```

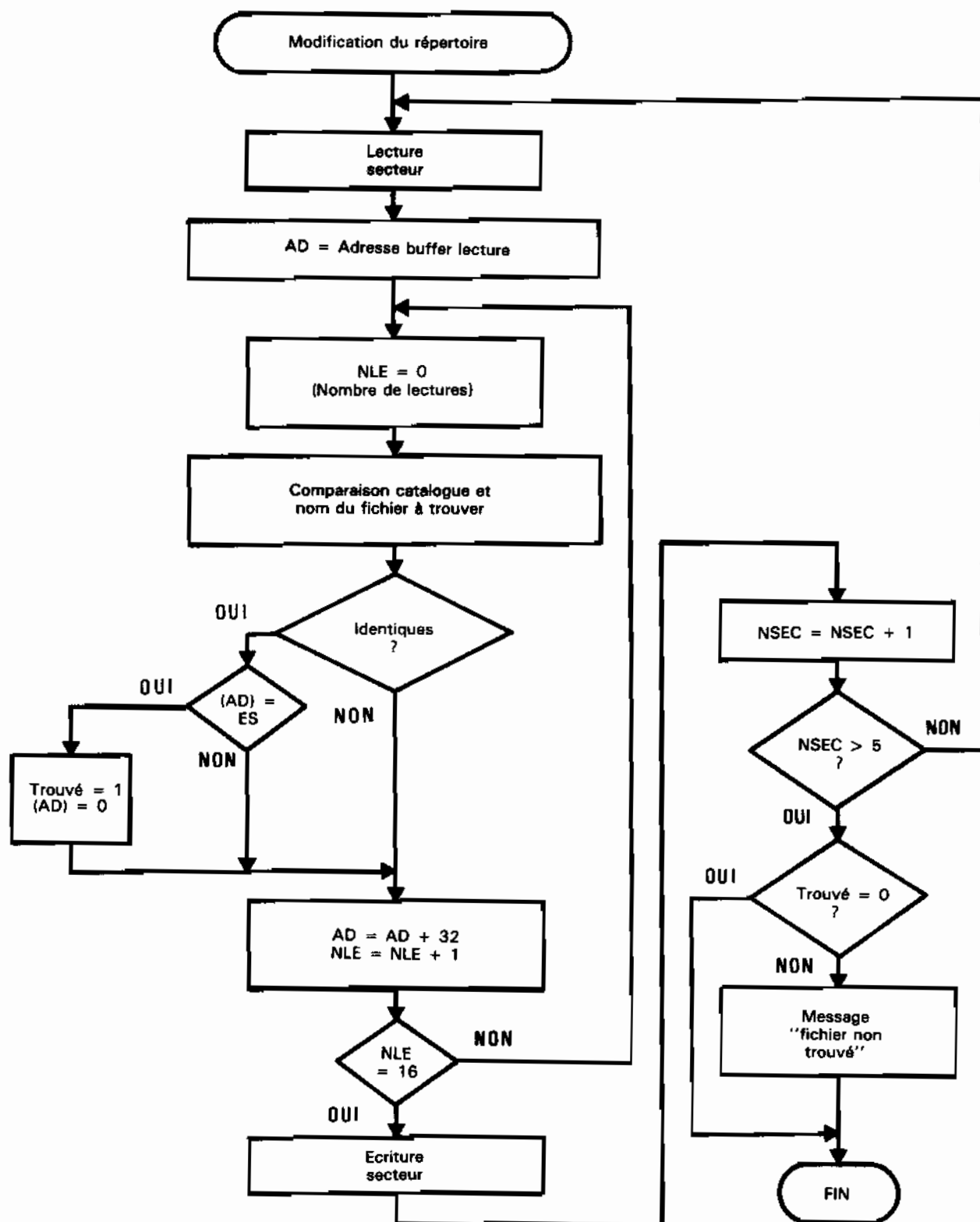
1000 REM Restitution d'un fichier efface
1010 REM -----
1020 MEMORY &BFFF 'Pour eviter un ecrasement par le Basic
1030 FOR I=&9375 TO &93B4
1040   READ A$
1050   A$="&"+A$
1060   POKE I,VAL(A$)
1070 NEXT I
1080 DATA 21,1B,92,CD,D4,BC,22,0,90,79,32,02,90,3A,4,92
1090 DATA 5F,3A,5,92,57,3A,6,92,4F,21,3,90,DF,0,90,C9
1100 DATA 21,17,92,CD,D4,BC,22,0,90,79,32,2,90,3A,4,92
1110 DATA 5F,3A,5,92,57,3A,6,92,4F,21,3,90,DF,0,90,C9
1120 POKE &9217,&84 'Ordre de lecture disque
1130 POKE &9218,&85 'Ordre d'écriture disque
1140 CLS
1150 PRINT "Restitution d'un fichier efface"
1160 PRINT "-----"
1170 PRINT:INPUT"Nom du fichier : ";nf$
1180 PRINT:INPUT"Extension : ";ex$
1190 PRINT:INPUT"Lecteur A ou B : ";le$
1200 le$=UPPER$(le$)
1210 IF le$<>"A" AND le$<>"B" THEN SOUND 1,100,30:GOTO 1190
1220 PRINT:INPUT"Format Systeme Data ou Ibm : ";fo$
1230 fo$=UPPER$(fo$)
1240 IF fo$<>"S" AND fo$<>"D" AND fo$<>"I" THEN SOUND 1,100,30:GOTO 1220
1250
1260 ll=LEN(nf$)

```

```
1270 IF l1<8 THEN FOR i=l1+1 TO 8 : nf$=nf$+" " : NEXT i
1280 nf$=nf$+ex$
1290 l1=LEN(ex$)
1300 IF l1<3 THEN FOR i=l1+1 TO 3 : nf$=nf$+" " : NEXT i
1310 '
1320 IF le$="A" THEN POKE &9204,0 ELSE POKE &9204,1
1330 IF fo$="S" THEN POKE &9205,2:POKE &9206,&41
1340 IF fo$="D" THEN POKE &9205,0:POKE &9206,&C1
1350 IF fo$="I" THEN POKE &9205,1:POKE &9206,1
1360 '
1370 nsec=1 'Nombre de secteurs lus
1380 CALL &9395 'Lecture
1390 ad=&9003 'Adresse de lecture nom
1400 nle=0 'Nombre de lectures
1410 lu$=""
1420 FOR i=1 TO l1
1430   ter=PEEK(ad+i)
1440   ter$=CHR$(ter)
1450   lu$=lu$+ter$
1460 NEXT i
1470 IF (lu$=nf$) AND (PEEK(ad)=&E5) THEN trouve=1:POKE ad,0
1480 ad=ad+32 'Passage a l'entree suivante
1490 nle=nle+1 'Nombre de lectures + 1
1500 IF nle<>16 THEN 1410 'Lecture prochaine entree
1510 CALL &9375 'Ecriture
1520 nsec=nsec+1 'Nombre de secteurs lus + 1
1530 IF nsec>5 THEN 1550
1540 POKE &9206,PEEK(&9206)+1:GOTO 1380 'Boucle de lecture
1550 '
1560 IF trouve=0 THEN PRINT:PRINT"Fichier non trouve"
1570 END
```

- Lignes 1000 à 1130 : Implantation du programme Assembleur en mémoire.
- Lignes 1140 à 1250 : Entrée des données pour retrouver le fichier.
- Lignes 1260 à 1310 : Constitution du nom du fichier à retrouver.
- Lignes 1320 à 1350 : Mémorisation des données entrées pour l'interface avec le programme Assembleur.
- Lignes 1370 à 1540 : Lecture, modification éventuelle et écriture des entrées du répertoire.
- Ligne 1560 : Affichage du résultat de la recherche.

Le détail des lignes 1370 à 1540 est donné dans l'algorithme suivant :



Programme de restitution de fichiers en version Assembleur

Le programme Assembleur de restitution reprend les opérations détaillées dans l'algorithme précédent.

A noter que l'affichage du texte est fait par la routine AFALPH, et la saisie formatée de texte par la routine SAISIE. La routine SAISIE demande le nombre maximal de caractères à saisir, et fait un retour chariot automatique lorsque ce nombre est atteint. Une routine d'émission d'un beep sonore (BEEP) est également utilisée. (Voir Partie 6, chap. 5.2).

Les routines de lecture et d'écriture d'un secteur sont implantées aux mêmes adresses que dans le programme Basic.

```

1                ORG 9000H
2                LOAD 9000H
3                ;-----
4                ;Recuperation d'un fichier efface
5                ;-----
6                ;
7                ;-----
8                ;Definition des constantes
9                ;-----
10               ;
11               SOUND:    EQU 0BD34H           ;MC SOUND REG
12               FIND:     EQU 0BCD4H           ;KL FIND COMMAND
13               PRINT:    EQU 0BB5AH           ;TXT OUTPUT
14               READ:     EQU 0BB06H           ; KM WAIT CHAR
15               MODE:     EQU 0BC0EH           ;SCR SET MODE
16               DEL:      EQU 127              ;Caractere DElete
17               BS:       EQU 8                ;Caractere Back Space
18               CR:       EQU 13              ;Carriage Return
19               LF:       EQU 10              ;Line Feed
20               CURS:     EQU 95              ;Caractere curseur
21               BLANC:    EQU 32              ;Caractere espace
22               ;

```

```

23      ; -----
24      ;Definition des variables
25      ; -----
26      ;
27      AD:          DS      3              ;Sauv. sortie de FIND
28      BUFF:        DS      512           ;Buffer ecriture
29      MAX:          DS      1
30      DRIVE:        DS      1            ;Nom du drive
31      PISTE:        DS      1            ;Numero de piste
32      SECT:         DS      1            ;Numero de secteur
33      PBUF:         DS      2            ;Pointeur sur un buffer
34      NOMLEC:       DS      1            ;Nom du lecteur
35      FORDIS:       DS      1            ;Format Disquette
36      NOM:          DS      12           ;Nom Fichier+extension
37 9217 84      LIT:          DB      84H   ;Instr. lecture
38 9218 85      ECRIT:       DB      85H   ;Instr. ecriture
39      ALEC:         DS      2            ;@ lecture ds buffer
40      NLE:          DS      1            ;Nbre d'entrees lues
41      TROUVE:       DS      1            ;Fichier trouve
42      NSEC:         DS      1            ;Nbre de secteurs lus
43      ;
44      ; -----
45      ;Definition des messages
46      ; -----
47      ;
48      MES1:         EQU      $
49 921E 52657374      DB      "Restitution d'un
49 9222 69747574
49 9226 696F6E20
49 922A 6427756E
49 922E 20666963

```

```

49 9232 68696572
50 9236 20656666          DB  " efface",CR,LF
50 923A 6163650D
50 923E 0A
51 923F 2D2D2D2D          DB  "-----"
51 9243 2D2D2D2D
51 9247 2D2D2D2D
51 924B 2D2D2D2D
51 924F 2D2D2D2D
51 9253 2D2D2D2D
52 9257 2D2D2D2D          DB  "-----",CR,LF,LF
52 925B 2D2D2D0D
52 925F 0A0AFF
53          ;-----
54          MES2:          EQU  $
55 9262 4E6F6D20          DB  "Nom du fichier :"
55 9266 64752066
55 926A 69636869
55 926E 6572203A
56 9272 5FFF              DB  CURS,OFFH
57          ;-----
58          MES2B:         EQU  $
59 9274 45787465          DB  "Extension :"
59 9278 6E73696F
59 927C 6E203A
60 927F 5FFF              DB  CURS,OFFH
61          ;-----
62          MES3:          EQU  $
63 9281 4C656374          DB  "Lecteur A ou B :"
63 9285 65757220
63 9289 41206F75

```

```

63 928D 2042203A
64 9291 5FFF                                DB    CURS,OFFH
65                                          ; -----
66          MES4:          EQU    $
67 9293 466F726D                                DB    "Format Systeme, D
67 9297 61742053
67 929B 79737465
67 929F 6D652C20
67 92A3 44617461
68 92A7 206F7520                                DB    " ou Ibm : ",CURS,0
68 92AB 49626D20
68 92AF 3A5FFF
69                                          ; -----
70          MES5:          EQU    $
71 92B2 46696368                                DB    "Fichier non trouv
71 92B6 69657220
71 92BA 6E6F6E20
71 92BE 74726F75
71 92C2 7665
72 92C4 FF                                DB    OFFH
73                                          ; -----
74          ALALI:          EQU    $
75 92C5 0D0A0AFF                                DB    CR,LF,LF,OFFH
76                                          ; -----
77                                          ;
78                                          ; -----
79                                          ; Sous-programmes
80                                          ; -----
81                                          ;
82                                          ; -----
83                                          ; Initialisation
84                                          ; -----

```

```

85      ;
86      INIT:      EQU  $              ;Point d'entree
87 92C9 210B92      LD  HL,NOM
88 92CC 3E20          LD  A,20H        ;Blanc
89 92CE 060C          LD  B,12
90      IO:      EQU  $
91 92D0 77            LD  (HL),A
92 92D1 23            INC  HL
93 92D2 10FC          DJNZ IO          ;RAZ Buffer Nom
94      ;
95 92D4 AF            XOR  A
96 92D5 321D92        LD  (NSEC),A     ;Nbre secteurs lus
97 92D8 3E01          LD  A,1
98 92DA 321C92        LD  (TROUVE),A   ;Non trouve
99 92DD 321B92        LD  (NLE),A     ;Nbre entrees lues
100 92E0 C9           RET
101      ;
102      ;-----
103      ;Emission d'un BEEP sonore
104      ;-----
105      ;Entree: aucune
106      ;Sortie: AF, BC et HL effaces
107      ;-----
108      ;
109      BEEP:      EQU  $              ;Point d'entree
110 92E1 3E00          LD  A,0
111 92E3 0E00          LD  C,0
112 92E5 CD34BD        CALL SOUND
113 92E8 3E01          LD  A,1
114 92EA 0E01          LD  C,1
115 92EC CD34BD        CALL SOUND      ;Frequence

```

```

116 92EF 3E08          LD  A,8
117 92F1 0E06          LD  C,6
118 92F3 CD34BD        CALL SOUND           ; Amplitude
119 92F6 3E07          LD  A,7
120 92F8 0E08          LD  C,8
121 92FA CD34BD        CALL SOUND           ; Validation registre A
122                   ;
123 92FD 21FFFF        LD  HL,0FFFFH
124                   BOU:  EQU  $
125 9300 2B            DEC  HL
126 9301 7C            LD  A,H
127 9302 B5            OR   L
128 9303 20FB          JR   NZ,BOU
129                   ;
130 9305 3E08          LD  A,8
131 9307 0E00          LD  C,0
132 9309 CD34BD        CALL SOUND           ; Amplitude 0
133 930C C9            RET
134                   ;
135                   ;-----
136                   ;Affichage d'un texte alphanum.
137                   ;-----
138                   ;Entree: @ de depart dans HL
139                   ;Sortie: aucun registre modifie
140                   ;-----
141                   ;
142                   AFALPH: EQU  $           ;point d'entree
143 930D E5            PUSH HL
144 930E F5            PUSH AF
145                   ME1:  EQU  $           ;Boucle d'affichage
146 930F 7E            LD  A,(HL)

```

```

147 9310 FEFF          CP    0FFH
148 9312 2806          JR     Z,ME2          ;Fin d'affichage
149 9314 CD5ABB          CALL PRINT          ;Af. caractere
150 9317 23            INC    HL            ;Caractere suivant
151 9318 18F5          JR     ME1
152                    ME2:    EQU    $
153 931A F1            POP    AF
154 931B E1            POP    HL
155 931C C9            RET
156                    ;
157                    ;-----
158                    ;Saisie de caracteres
159                    ;-----
160                    ;Entree: (MAX)=Nbre de caracteres
161                    ;Sortie: Aucun registre ecrase
162                    ;-----
163                    ;
164                    SAISIE:    EQU    $          ;Point d'entree
165 931D 3A0392          LD     A,(MAX)
166 9320 57            LD     D,A          ;Nbre de caract. a lire
167 9321 010000          LD     BC,0          ;Index dans le buffer
168                    S1:    EQU    $
169 9324 2A0792          LD     HL,(PBUF)      ;Buffer de lecture
170 9327 CD06BB          CALL READ          ;Lecture 1 caractere
171 932A FE0D          CP     CR          ;Carriage Return ?
172 932C 283C          JR     Z,S3          ;Oui=> Fin de saisie
173 932E FE7F          CP     DEL          ;DElete ?
174 9330 2818          JR     Z,S2          ;Oui => Traitement DEL
175 9332 F5            PUSH AF
176 9333 3E08          LD     A,BS
177 9335 CD5ABB          CALL PRINT

```

```

178 9338 F1          POP  AF
179 9339 CD5ABB      CALL PRINT
180 933C 09          ADD  HL,BC
181 933D 77          LD   (HL),A          ;Sauvegarde
182 933E 03          INC  BC
183 933F 3E5F        LD   A,CURS
184 9341 CD5ABB      CALL PRINT
185 9344 79          LD   A,C
186 9345 BA          CP   D
187 9346 20DC        JR   NZ,S1          ;Suite de la saisie
188 9348 1820        JR   S3            ;Fin de saisie
189                S2:  EQU  $
190 934A 79          LD   A,C
191 934B B7          OR   A
192 934C 28D6        JR   Z,S1          ;DEL non accepte
193 934E 0B          DEC  BC
194 934F 3E0B        LD   A,BS
195 9351 CD5ABB      CALL PRINT          ;Retour en arriere
196 9354 3E20        LD   A,BLANC
197 9356 CD5ABB      CALL PRINT          ;Effacement caractere
198 9359 3E0B        LD   A,BS
199 935B CD5ABB      CALL PRINT          ;Retour en arriere
200 935E 3E0B        LD   A,BS
201 9360 CD5ABB      CALL PRINT          ;Retour en arriere
202 9363 3E5F        LD   A,CURS
203 9365 CD5ABB      CALL PRINT          ;Affichage curseur
204 9368 18BA        JR   S1
205                S3:  EQU  $
206 936A 3E0B        LD   A,BS
207 936C CD5ABB      CALL PRINT          ;Retour en arriere
208 936F 3E20        LD   A,BLANC

```



```

209 9371 CD5ABB          CALL PRINT          ;Effacement caract.
210 9374 C9              RET
211                      ;
212                      ;-----
213                      ; Ecriture d'un secteur
214                      ;-----
215                      ;Entree:
216                      ;Sortie:
217                      ;-----
218                      ;
219 ECR:                  EQU  $              ;Point d'entree
220 9375 211892           LD  HL,ECRIT        ;Ecriture disque
221 9378 CDD4BC           CALL FIND           ;KL FIND COMMAND
222 937B 220090           LD  (AD),HL
223 937E 79              LD  A,C
224 937F 320290           LD  (AD+2),A
225 9382 3A0492           LD  A,(DRIVE)
226 9385 5F              LD  E,A              ;No de drive
227 9386 3A0592           LD  A,(PISTE)
228 9389 57              LD  D,A              ;No de piste
229 938A 3A0692           LD  A,(SECT)
230 938D 4F              LD  C,A              ;No de secteur
231 938E 210390           LD  HL,BUFF
232 9391 DF              RST  24              ;Activ. instruction en RO
233 9392 0090            DW  AD
234 9394 C9              RET
235                      ;
236                      ;-----
237                      ; Lecture d'un secteur
238                      ;-----
239                      ;Entree:

```

```

240          ;Sortie:
241          ;-----
242          ;
243          LECT:      EQU  $           ;Point d'entree
244 9395 211792      LD  HL,LIT         ;Lecture disque
245 9398 CDD4BC      CALL FIND          ;KL FIND COMMAND
246 939B 220090      LD  (AD),HL
247 939E 79         LD  A,C
248 939F 320290      LD  (AD+2),A
249 93A2 3A0492      LD  A,(DRIVE)
250 93A5 5F         LD  E,A           ;No de drive
251 93A6 3A0592      LD  A,(PISTE)
252 93A9 57         LD  D,A           ;No de piste
253 93AA 3A0692      LD  A,(SECT)
254 93AD 4F         LD  C,A           ;No de secteur
255 93AE 210390      LD  HL,BUFF
256 93B1 DF         RST  24           ;Activ. instruction en R0
257 93B2 0090      DW  AD
258 93B4 C9         RET
259          ;
260          ;=====
261          ; PROGRAMME PRINCIPAL
262          ;=====
263          ORG 8000H
264          LOAD 8000H
265 8000 CDC992      CALL INIT          ;des variables
266 8003 3E01      LD  A,1
267 8005 CD0EBC      CALL MODE          ;Mode 1
268 8008 211E92      LD  HL,MES1
269 800B CD0D93      CALL AFALPH        ;Affich 1er message
270 800E 216292      LD  HL,MES2

```

Partie 9 : Programmes

```

271 8011 CD0D93      CALL AFALPH      ;Aff 2eme message
272 8014 3E08        LD   A,8
273 8016 320392      LD   (MAX),A      ;Saisie sur 8 car. max
274 8019 210B92      LD   HL,NOM
275 801C 220792      LD   (PBUF),HL    ;Buffer de lecture
276 801F CD1D93      CALL SAISIE      ;Saisie du nom
277 8022 21C592      LD   HL,ALALI
278 8025 CD0D93      CALL AFALPH      ;A la ligne
279 8028 217492      LD   HL,MES2B
280 802B CD0D93      CALL AFALPH      ;Aff. message 2 bis
281 802E 3E03        LD   A,3
282 8030 320392      LD   (MAX),A      ;3 caract. maxi
283 8033 211392      LD   HL,NOM+8
284 8036 220792      LD   (PBUF),HL
285 8039 CD1D93      CALL SAISIE      ;Saisie extension
286                PPO: EQU $
287 803C 21C592      LD   HL,ALALI
288 803F CD0D93      CALL AFALPH      ;A la ligne
289 8042 218192      LD   HL,MES3
290 8045 CD0D93      CALL AFALPH      ;Aff 3eme message
291 8048 3E01        LD   A,1
292 804A 320392      LD   (MAX),A      ;Saisie 1 caractere
293 804D 210992      LD   HL,NOMLEC
294 8050 220792      LD   (PBUF),HL
295 8053 CD1D93      CALL SAISIE      ;Lecture nom lecteur
296 8056 3A0992      LD   A,(NOMLEC)
297 8059 E6DF        AND   0DFH      ;Mise en majuscule
298 805B FE41        CP   'A'
299 805D 2B09        JR   Z,PP1
300 805F FE42        CP   'B'
301 8061 2B05        JR   Z,PP1      ;OK

```

```

302 B063 CDE192      CALL BEEP                ;Emission d'un son
303 B066 18D4         JR PP0                    ;Saisie a nouveau
304                  PP1: EQU $
305 B068 21C592      LD HL,ALALI
306 B06B CD0D93      CALL AFALPH                ;A la ligne
307 B06E 219392      LD HL,MES4
308 B071 CD0D93      CALL AFALPH                ;Aff 4eme message
309 B074 3E01         LD A,1
310 B076 320392      LD (MAX),A                ;Saisie 1 caractere
311 B079 210A92      LD HL,FORDIS
312 B07C 220792      LD (PBUF),HL
313 B07F CD1D93      CALL SAISIE                ;Lecture format disk
314 B082 3A0A92      LD A,(FORDIS)
315 B085 E6DF        AND ODFH                  ;Mise en majuscule
316 B087 FE53        CP 'S'
317 B089 280D        JR Z,PP2
318 B08B FE44        CP 'D'
319 B08D 2809        JR Z,PP2
320 B08F FE49        CP 'I'
321 B091 2805        JR Z,PP2                  ;OK
322 B093 CDE192      CALL BEEP                ;Emission d'un son
323 B096 18D0        JR PP1                    ;Saisie a nouveau
324                  PP2: EQU $
325                  ;
326 B098 3A0992      LD A,(NOMLEC)
327 B09B E6DF        AND ODFH                  ;Mise en majuscule
328 B09D FE41        CP 'A'
329 B09F 2003        JR NZ,PP3
330 B0A1 AF          XOR A
331 B0A2 1802        JR PP4
332 B0A4 3E01        PP3: LD A,1

```

Partie 9 : Programmes

```

333 80A6 320492  PP4:      LD      (DRIVE),A          ;Nom du drive
334              ;
335 80A9 3A0A92      LD      A,(FORDIS)
336 80AC E6DF        AND      0DFH                  ;Mise en majuscule
337 80AE FE53        CP       'S'
338 80B0 280E        JR       Z,PP5
339 80B2 FE44        CP       'D'
340 80B4 2816        JR       Z,PP6
341 80B6 3E01        LD      A,1
342 80B8 320592      LD      (PISTE),A
343 80BB 320692      LD      (SECT),A
344 80BE 1815        JR       PP6B
345 80C0 3E02  PP5:      LD      A,2
346 80C2 320592      LD      (PISTE),A
347 80C5 3E41        LD      A,41H
348 80C7 320692      LD      (SECT),A
349 80CA 1809        JR       PP6B
350 80CC AF  PP6:      XOR      A
351 80CD 320592      LD      (PISTE),A
352 80D0 3E01        LD      A,0C1H
353 80D2 320692      LD      (SECT),A
354              PP6B:   EQU      $
355 80D5 3E01        LD      A,1
356 80D7 321D92      LD      (NSEC),A              ;RAZ nbre sect lus
357              PP7:   EQU      $
358 80DA CD9593      CALL    LECT                  ;Lecture secteur
359 80DD 210390      LD      HL,BUFF              ;@ buff lect
360 80E0 221992      LD      (ALEC),HL            ;@ de lecture
361 80E3 3E01        LD      A,1
362 80E5 321B92      LD      (NLE),A
363              ;

```

```

364          PP8:      EQU  $              ; Comparaison
365 80EB 2A1992      LD   HL, (ALEC)
366 80EB 23          INC  HL
367 80EC 110B92      LD   DE, NOM
368 80EF 060B      LD   B, 11              ; 11 caract a comparer
369          PP8B:     EQU  $
370 80F1 1A          LD   A, (DE)
371 80F2 BE          CP   (HL)
372 80F3 2015        JR   NZ, PP10
373 80F5 23          INC  HL
374 80F6 13          INC  DE
375 80F7 05          DEC  B
376 80F8 20F7        JR   NZ, PP8B        ; Boucle compar.
377          ; Comparaison reussie
378 80FA 2A1992      LD   HL, (ALEC)
379 80FD 7E          LD   A, (HL)
380 80FE FEE5        CP   0E5H
381 8100 200B        JR   NZ, PP10
382 8102 AF          XOR  A
383 8103 321C92      LD   (TROUVE), A      ; Fichier trouve
384 8106 2A1992      LD   HL, (ALEC)
385 8109 77          LD   (HL), A          ; Restitution
386          ;
387          PP10:     EQU  $              ; Comparaison NOK
388 810A 2A1992      LD   HL, (ALEC)
389 810D 112000      LD   DE, 32
390 8110 19          ADD  HL, DE
391 8111 221992      LD   (ALEC), HL
392 8114 3A1B92      LD   A, (NLE)        ; Nbre entrees
393 8117 3C          INC  A
394 811B 321B92      LD   (NLE), A        ; Nbre entrees + 1

```

```
395 811B FE11          CP    17
396 811D 20C9          JR     NZ,PP8
397 811F CD7593        CALL ECR          ;Ecriture secteur
398 8122 3A0692        LD     A,(SECT)
399 8125 3C            INC    A
400 8126 320692        LD     (SECT),A    ;Prochain secteur
401 8129 3A1D92        LD     A,(NSEC)
402 812C 3C            INC    A
403 812D 321D92        LD     (NSEC),A    ;Nbre secteurs+1
404 8130 FF05          CP     5
405 8132 20A6          JR     NZ,PP7
406 8134 3A1C92        LD     A,(TROUVE)
407 8137 B7            OR     A
408 8138 280C          JR     Z,PP11      ;Fichier pas trouve
409 813A 21C592        LD     HL,ALALI
410 813D CD0D93        CALL AFALPH      ;A la ligne
411 8140 21B292        LD     HL,MES5
412 8143 CD0D93        CALL AFALPH      ;Fichier non trouve
413                    PP11: EQU $
414 8146 C9            RET
415                    END
```

Chargeur Basic du programme Assembleur de restitution

Si vous ne voulez pas entrer le listing assembleur précédent, tout en conservant la vitesse d'exécution de l'assembleur, vous pourrez entrer le chargeur que nous présentons ici.

```

1000 REM Recuperation d'un fichier efface par IERA
1010 REM =====
1020 REM Mise en memoire du programme assembleur
1030 REM -----
1040 FOR I=&920E TO &93BD
1050     READ A$
1060     A$="&"+A$
1070     POKE I,VAL(A$)
1080 NEXT I
1090 FOR I=&8000 TO &814F
1100     READ a$
1110     A$="&"+A$
1120     POKE I,VAL(A$)
1130 NEXT I
1140 REM -----
1150 REM Appel au programme en &8000
1160 REM -----
1170 CALL &8000
1180 END

2000 REM =====
2010 REM Variables et sous-programmes
2020 REM =====
2030 DATA 0,0,0,0,0,0,0,0,0,0,B4,B5,0,0,0,0,0
2040 DATA 52,65,73,74,69,74,75,74,69,6F,6E,20,64,27,75,6E
2050 DATA 20,66,69,63,68,69,65,72,20,65,66,66,61,63,65,D
2060 DATA A,2D,2D,2D,2D,2D,2D,2D,2D,2D,2D,2D,2D,2D,2D
2070 DATA 2D,2D,2D,2D,2D,2D,2D,2D,2D,2D,2D,2D,2D,2D,2D

```


Partie 9 : Programmes

```
2080 DATA D,A,A,FF,4E,6F,6D,20,64,75,20,66,69,63,68,69
2090 DATA 65,72,20,3A,5F,FF,45,7B,74,65,6E,73,69,6F,6E,20
2100 DATA 3A,5F,FF,4C,65,63,74,65,75,72,20,41,20,6F,75,20
2110 DATA 42,20,3A,5F,FF,46,6F,72,6D,61,74,20,53,79,73,74
2120 DATA 65,6D,65,2C,20,44,61,74,61,20,6F,75,20,49,62,6D
2130 DATA 20,3A,5F,FF,46,69,63,68,69,65,72,20,6E,6F,6E,20
2140 DATA 74,72,6F,75,76,65,FF,D,A,A,FF,21,B,92,3E,20
2150 DATA 6,C,77,23,10,FC,AF,32,1D,92,3E,1,32,1C,92,32
2160 DATA 1B,92,C9,3E,0,E,0,CD,34,BD,3E,1,F,1,CD,34
2170 DATA BD,3E,8,E,6,CD,34,BD,3E,7,E,8,CD,34,BD,21
2180 DATA FF,FF,2B,7C,B5,20,FB,3E,8,E,0,CD,34,BD,C9,E5
2190 DATA F5,7E,FE,FF,28,6,CD,5A,BB,23,18,F5,F1,E1,C9,3A
2200 DATA 3,92,57,1,0,0,2A,7,92,CD,6,BB,FE,D,28,3C
2210 DATA FE,7F,28,18,F5,3E,8,CD,5A,BB,F1,CD,5A,BB,9,77
2220 DATA 3,3E,5F,CD,5A,BB,79,BA,20,DC,18,20,79,B7,2B,D6
2230 DATA B,3E,8,CD,5A,BB,3E,20,CD,5A,BB,3E,8,CD,5A,BB
2240 DATA 3E,8,CD,5A,BB,3E,5F,CD,5A,BB,18,BA,3E,8,CD,5A
2250 DATA BB,3E,20,CD,5A,BB,C9,21,18,92,CD,D4,BC,22,0,90
2260 DATA 79,32,2,90,3A,4,92,5F,3A,5,92,57,3A,6,92,4F
2270 DATA 21,3,90,DF,0,90,C9,21,17,92,CD,D4,BC,22,0,90
2280 DATA 79,32,2,90,3A,4,92,5F,3A,5,92,57,3A,6,92,4F
2290 DATA 21,3,90,DF,0,90,C9,0,0,0,0,0,0,0,0,0,0
3000 REM =====
3010 REM Programme principal
3020 REM =====
3030 DATA CD,C9,92,3E,1,CD,E,BC,21,1E,92,CD,D,93,21,62
3040 DATA 92,CD,D,93,3E,8,32,3,92,21,B,92,22,7,92,CD
3050 DATA 1D,93,21,C5,92,CD,D,93,21,74,92,CD,D,93,3E,3
3060 DATA 32,3,92,21,13,92,22,7,92,CD,1D,93,21,C5,92,CD
3070 DATA D,93,21,81,92,CD,D,93,3E,1,32,3,92,21,9,92
3080 DATA 22,7,92,CD,1D,93,3A,9,92,E6,DF,FE,41,2B,9,FE
```

```

3090 DATA 42,28,5,CD,E1,92,1B,D4,21,C5,92,CD,D,93,21,93
3100 DATA 92,CD,D,93,3E,1,32,3,92,21,A,92,22,7,92,CD
3110 DATA 1D,93,3A,A,92,E6,DF,FE,53,2B,D,FE,44,28,9,FE
3120 DATA 49,28,5,CD,E1,92,1B,D0,3A,9,92,E6,DF,FE,41,20
3130 DATA 3,AF,1B,2,3E,1,32,4,92,3A,A,92,E6,DF,FE,53
3140 DATA 2B,E,FE,44,28,16,3E,1,32,5,92,32,6,92,1B,15
3150 DATA 3E,2,32,5,92,3E,41,32,6,92,1B,9,AF,32,5,92
3160 DATA 3E,C1,32,6,92,3E,1,32,1D,92,CD,95,93,21,3,90
3170 DATA 22,19,92,3E,1,32,1B,92,2A,19,92,23,11,B,92,6
3180 DATA B,1A,BE,20,15,23,13,5,20,F7,2A,19,92,7E,FE,E5
3190 DATA 20,8,AF,32,1C,92,2A,19,92,77,2A,19,92,11,20,0
3200 DATA 19,22,19,92,3A,1B,92,3C,32,1B,92,FE,11,20,C9,CD
3210 DATA 75,93,3A,6,92,3C,32,6,92,3A,1D,92,3C,32,1D,92
3220 DATA FE,5,20,66,3A,1C,92,B7,2B,C,21,C5,92,CD,D,93
3230 DATA 21,B2,92,CD,D,93,C9,0,0,0,0,0,0,0,0,0

```

Pour être sûr que le programme fonctionnera au premier essai, nous vous donnons la liste des codes de vérification à utiliser avec le programme de checksum (Voir Partie 9, chap. 8.4).

```

0A 3E 86 AF D2 6B 72 F6 3C 3E 03 E5 9D D3 14 3D 8E B1 35 1E A1 EC
A5 B9 CB B9 EE C5 57 70 10 08 47 3B 4F 49 9E C4 B8 EE 97 9A A5 0D
B2 EA 87 9E

```

9/8.7

Défilement d'un message alphanumérique sur l'écran

Ce programme va être présenté dans deux versions : Basic et Assembleur. Il vous sera également possible d'entrer les données hexadécimales correspondant aux codes opératoires sous Basic.

Version Basic

Un message alphanumérique quelconque peut être affiché n'importe où sur l'écran, en mode 0, 1 ou 2. Pour cela, il faut placer le message à afficher dans la variable M\$. Les coordonnées absolues d'affichage du premier caractère sur l'écran sont placées dans les variables X (abscisse) et Y (ordonnée). La vitesse de défilement est enfin indiquée dans la variable V. L'affichage est déclenché en faisant un « GOSUB 1000 », comme le montre l'exemple suivant :

```
10 MODE 2
20 V=80
30 M$="Texte quelconque pour tester la routine "
40 X=10:Y=15
50 CLS
60 GOSUB 1000
70 END
```

Le décalage se fait en trois étapes :

- isoler la première lettre de la chaîne ;
- copier les caractères suivants dans la chaîne de départ ;
- placer le caractère isolé en fin de chaîne.

Cette manipulation est faite lignes 1100 et 1110.

La boucle d'attente paramétrée se fait ligne 1120. Elle consiste à faire une boucle FOR NEXT dont la longueur dépend de la vitesse définie dans la variable V.

Le listing du programme est le suivant :

```

1000 '-----
1010 '----- MESSAGE DEFILANT -----
1020 '-----
1030 '---Entree:V = Vitesse (entre 0 et 100) ---
1040 '--- M$ = Message ---
1050 '--- X = Abscisse 1ere lettre du message ---
1060 '--- Y = Ordonnee 1ere lettre du message ---
1070 '-----
1080 LOCATE X,Y
1090 PRINT M$
1100 I$=MID$(M$,1,1):J$=MID$(M$,2,LEN(M$)-1)
1110 M$=J$+I$
1120 FOR I=1 TO (100-V)*3:NEXT I
1130 A$=INKEY$
1140 IF A$="" THEN 1080
1150 RETURN

```

Version Assembleur

Même si la vitesse d'exécution d'un programme écrit en Assembleur n'est plus à démontrer, vous serez peut-être surpris par le programme qui suit.

Il reprend les concepts exposés pour le programme de défilement Basic. Trois routines du FIRMWARE sont utilisées pour faciliter l'écriture du programme. Il s'agit de :

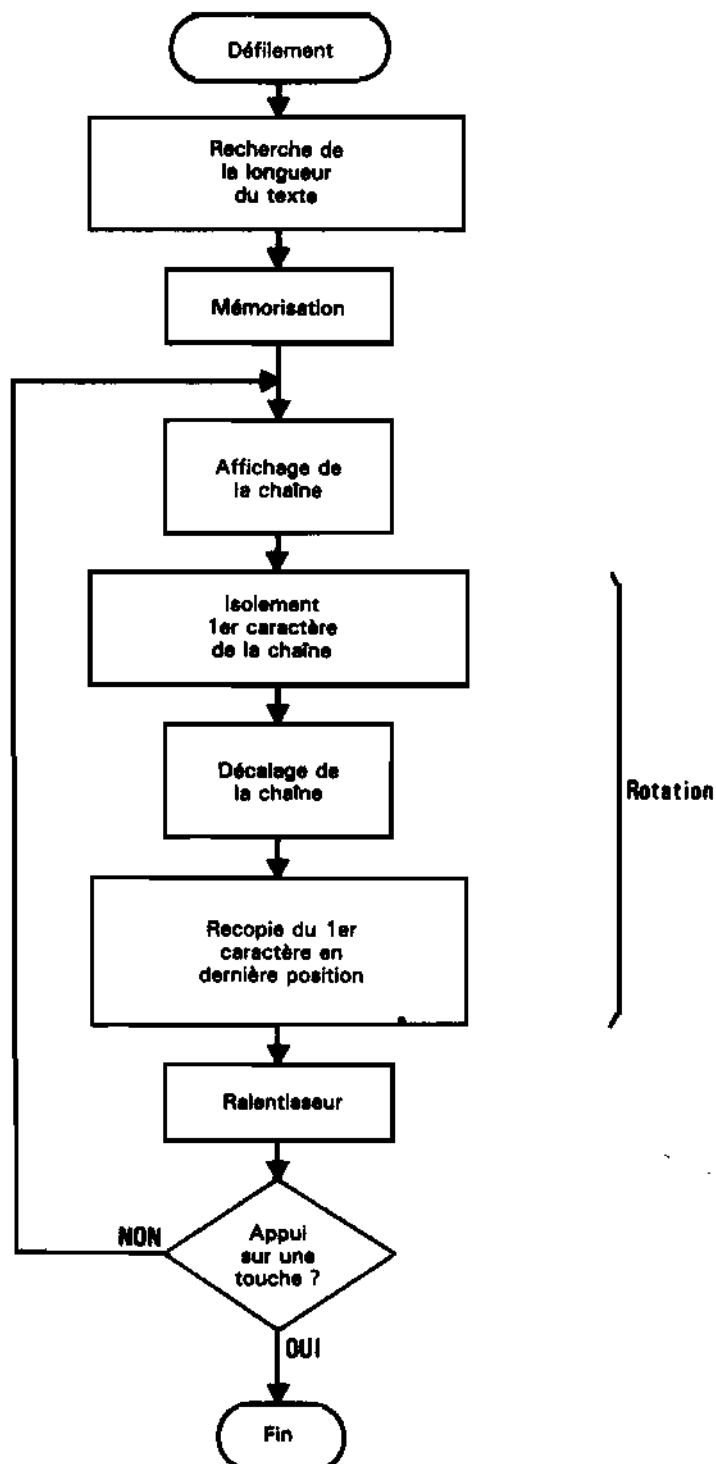
— TXT SET CURSOR (#BB75) qui positionne le curseur à un endroit quelconque de l'écran ;

- TXT OUTPUT qui affiche un caractère à la position courante du curseur ;
- KM READ KEY qui lit le code d'une éventuelle touche actionnée sur le clavier.

Le programme se décompose en cinq parties :

- recherche de la longueur du texte à afficher. Précisons à ce sujet que le texte à afficher doit être suivi du caractère terminateur « nul » de code ASCII 0 ;
- affichage de la chaîne jusqu'au terminateur ;
- rotation de la chaîne,
- ralentisseur. Il consiste en une boucle qui utilise l'ordre DJNZ. Cette boucle contient 6 ordres NOP qui n'ont aucune action particulière, si ce n'est demander un cycle machine pour être exécutés.
- test, si une touche a été actionnée. Si oui, arrêt du programme et retour à l'appelant.

Ces diverses actions se retrouvent dans l'organigramme ci-dessous :



Le listing Assembleur est le suivant :

```

1          ORG 9000H
2          LOAD 9000H
3          ;-----
4          ;Defilement d'un message sur
5          ;l'ecran.
6          ;-----
7          ;Entree: A=Vitesse (0 a 255)
8          ;      H=Position en X
9          ;      L=Position en Y
10         ;-----
11         ;
12         ;
13         ;-----
14         ;Zone des constantes du programme
15         ;-----
16         SETCUR:    EQU 0BB75H          ;TXT SET CURSOR
17         WRCHAR:    EQU 0BB5AH          ;TXT OUTPUT
18         RKEY:      EQU 0BB1BH          ;KM READ KEY
19         ;
20         ;
21 9000 326D90        LD    (VIT),A        ;Vitesse
22 9003 226790        LD    (POS),HL       ;Sauvegarde
23 9006 ED536990      LD    (TEXTE),DE     ;Sauvegarde à texte
24 900A EB           EX    DE,HL
25         ;
26         ;Recherche de la longueur de TEXTE
27         ;
28         BP:        EQU $               ;Boucle rech longueur
29 900B 7E           LD    A,(HL)
30 900C B7           OR    A
31 900D 23           INC   HL

```

Partie 9 : Programmes

```

32 900E 20FB          JR    NZ,BP
33 9010 ED5B6990      LD    DE,(TEXTE)
34 9014 37            SCF
35 9015 3F            CCF
36 9016 ED52          SRC    HL,DE
37 9018 2B            DEC    HL
38 9019 7D            LD     A,L
39 901A 326B90        LD     (LEN),A          ;Longueur chaine
40                    ;
41                    MB:    EQU    $
42 901D 2A6790        LD     HL,(PDS)
43 9020 CD75BB        CALL SETCUR          ;Locate
44 9023 2A6990        LD     HL,(TEXTE)
45                    BP0:    EQU    $          ;Bourle d'affichage
46 9026 7E            LD     A,(HL)
47 9027 B7            OR     A
48 9028 2B06          JR     Z,BP1
49 902A CD5ABB        CALL WRCHAR          ;Affichage
50 902D 23            INC     HL
51 902E 1BF6          JR     BP0
52                    ;
53                    BP1:    EQU    $          ;Rotation chaine
54 9030 2A6990        LD     HL,(TEXTE)
55 9033 7E            LD     A,(HL)          ;iere lettre
56 9034 326C90        LD     (SAUV),A      ;Sauvegarde
57 9037 3A6B90        LD     A,(LEN)
58 903A 47            LD     B,A          ;Longueur decalage
59 903B 05            DEC     B
60 903C ED5B6990      LD     DE,(TEXTE)
61 9040 1B            DEC     DE
62                    BP2:    EQU    $
63 9041 23            INC     HL

```



```

64 9042 13          INC  DE          ;Caract suivant
65 9043 7E          LD   A,(HL)
66 9044 12          LD   (DE),A
67 9045 10FA        DJNZ BP2          ;Boucle de deplact
68 9047 3A6C90      LD   A,(SAUV)
69 904A 77          LD   (HL),A
70 904B CD5490      CALL WAIT          ;Ralentisseur
71 904E CD1BBB      CALL RKEY          ;Test appui clavier
72 9051 30CA        JR   NC,MB        ;Pas d'appui => boucle
73 9053 C9          RET
74                  ;
75                  ;Ralentisseur
76                  ;
77                  WAIT: EQU $
78 9054 3A6D90      LD   A,(VIT)      ;Vitesse
79                  W0: EQU $          ;Boucle ralentisseur
80 9057 3D          DEC  A
81 9058 2001        JR   NZ,W1
82 905A C9          RET
83                  W1: EQU $
84 905B 0600        LD   B,0
85                  W2: EQU $
86 905D 00          NOP
87 905E 00          NOP
88 905F 00          NOP
89 9060 00          NOP
90 9061 00          NOP
91 9062 00          NOP          ;Instructions NOP pour att
92 9063 10FB        DJNZ W2
93 9065 1BF0        JP   W0
94                  ;-----
95                  ;Zone des variables du programme
96                  ;-----

```

Partie 9 : Programmes

```

97          POS:      DS    2                ;Position du curseur
98          TEXTE:    DS    2                ;à debut de texte
99          LEN:      DS    1                ;Longueur chaine
100         SAUV:     DS    1                ;Sauveg. caractere
101         VIT:      DS    1                ;Vitesse d'affichage
102                                     END

```

Pour faciliter l'interfaçage, un programme Basic peut appeler ce programme Assembleur de la façon suivante :

```

100 MODE 2
110 FOR I=&8000 TO &800B:READ A:POKE I,A:NEXT
120 DATA &21,10,10,&11,0,&81,&3E,&70,&CD,&0,&90,&C9
130 A$="Exemple de message..."
140 FOR I=1 TO LEN(A$)
150   B$=MID$(A$,I,1)
160   B=ASC(B$) 'Code ASCII de chaque lettre
170   POKE &8100+I-1,B 'Mise en memoire
180 NEXT I
190 POKE &8100+I-1,0 'Code terminateur

```

L'abscisse correspond à la deuxième donnée, l'ordonnée à la troisième donnée.

Les données de la ligne 120 correspondent au programme Assembleur suivant :

```

1          ORG    8000H
2          LOAD  8000H
3 8000 210A0A    LD    HL,0A0AH                ;Position curseur
4 8003 1100B1    LD    DE,8100H                ;Position memoire
5 8006 3E70      LD    A,70H                  ;Vitesse d'affichage
6 8008 CD0090    CALL 9000H
7 800B C9        RET
8              END

```

Il est également possible de saisir les codes hexadécimaux correspondant au programme Assembleur dans un programme Basic. Le listing du programme est alors le suivant :

```

1000 'Défilement d'un message sur l'écran
1010 '=====
1020 FOR I=&9000 TO &9066
1030   READ A$
1040   B$="&"+A$
1050   POKE I,VAL(B$)
1060 NEXT I
1070 '-----
1080 'Données hexadécimales
1090 '-----
1100 DATA 32,6D,90,23,67,90,ED,53,69,90,EB,7E,B7,23,20,FB
1110 DATA ED,5B,69,90,37,3F,ED,52,2B,7D,32,6B,90,2A,67,90
1120 DATA CD,75,BB,2A,69,90,7E,B7,2B,06,CD,5A,BB,23,1B,F6
1130 DATA 2A,69,90,7E,32,6C,90,3A,6B,90,47,05,ED,5B,69,90
1140 DATA 1B,23,17,7E,12,10,FA,3A,6C,90,77,CD,54,90,CD,1B
1150 DATA BB,30,CA,C9,3A,6D,90,3D,20,01,C9,06,00,00,00,00
1160 DATA 00,00,00,10,FB,1B,F0,00,00,00,00,00,00,00,00

```

Si vous décidez d'entrer les codes du programme assembleur sous Basic, vérifiez les codes entrés grâce au programme de checksum (voir Partie 9, chap. 8.4).

Pour cela, tapez « MERGE » suivi du nom sous lequel vous avez sauvegardé le programme de checksum. Exécutez le programme de checksum en tapant « RUN 50000 ». Les données de vérification sont les suivantes :

E6 F2 9D 97 37 E6 12

Si une ou plusieurs des données de vérification ne correspondent pas avec celles données ci-dessus, vérifiez la ligne correspondante.

Remarque :

Plusieurs données nulles ont été rajoutées en fin de listing afin d'assurer la compatibilité avec le programme de checksum.

9/8.8

Driver d'imprimante DMP 2000

L'imprimante DMP 2000 dispose d'un jeu de caractères réduit qui est fonction du pays où elle est utilisée. Les micro-switchs DS1-1, DS1-2 et DS1-3 permettent de spécifier le pays comme le montre le tableau suivant :

Pays	DS1-1	DS1-2	DS1-3
USA	ON	ON	ON
GB	OFF	OFF	ON
France	OFF	ON	ON
Allemagne	ON	OFF	ON
Danemark	ON	ON	OFF
Suède	OFF	ON	OFF
Italie	ON	OFF	OFF
Espagne	OFF	OFF	OFF

En fonction du type de port Centronics (7 ou 8 bits) utilisé par l'ordinateur, les caractères imprimables seront les suivants (les exemples qui suivent ont été faits sur un IBM-PC pour illustrer l'impression Centronics 8 bits et sur un Amstrad CPC pour illustrer l'impression Centronics 7 bits).

Plusieurs commandes ESCape destinées à l'imprimante permettent d'imprimer des caractères qui n'appartiennent pas au jeu de caractères standard et qui sont définis au gré de l'utilisateur.

Voyons comment créer et utiliser des caractères définis par l'utilisateur :

- Éléments de base sur la définition de caractères graphiques.
- Définition de caractères graphiques avec un utilitaire écrit en Basic. Ces caractères sont stockés dans un fichier binaire à l'intention du driver d'imprimante.
- Impression de textes en utilisant un driver d'imprimante.

Le driver est écrit en Basic et en Assembleur. Il permet d'imprimer tout texte ASCII composé de caractères de codes compris entre 0 et 255 (comme par exemple un texte issu du traitement de texte Pocket Wordstar).

CPC 7 bits
(France)

```

1 ! 22 " 23 # 24 $ 25 % 26 & 27 ' 28 (
9 ) 2A * 2B + 2C , 2D - 2E . 2F / 30 0
1 1 32 2 33 3 34 4 35 5 36 6 37 7 38 8
9 9 3A : 3B ; 3C < 3D = 3E > 3F ? 40 à
1 A 42 B 43 C 44 D 45 E 46 F 47 G 48 H
9 I 4A J 4B K 4C L 4D M 4E N 4F O 50 P
1 Q 52 R 53 S 54 T 55 U 56 V 57 W 58 X
9 Y 5A Z 5B ^ 5C _ 5D ` 5E ~ 5F " 60 '
1 a 62 b 63 c 64 d 65 e 66 f 67 g 68 h
9 i 6A j 6B k 6C l 6D m 6E n 6F o 70 p
1 q 72 r 73 s 74 t 75 u 76 v 77 w 78 x
9 y 7A z 7B ù 7C ù 7D ù 7E " 7F 80
1 ç A2 é A3 # A4 $ A5 % A6 & A7 ' AB (
9 ) AA * AB ù AC é AD " AE $ AF " BO 0
1 1 B2 2 B3 3 B4 4 B5 5 B6 6 B7 7 B8 8
9 9 BA : BB ; BC < BD = BE > BF ? CO à
1 A C2 B C3 C C4 D C5 E C6 F C7 G CB H
9 I CA J CB K CC L CD M CE N CF O DO P
1 Q D2 R D3 S D4 T D5 U D6 V D7 W DB X
9 Y DA Z DB * DC ç DD $ DE ^ DF _ EO '
1 a E2 b E3 c E4 d E5 e E6 f E7 g E8 h
9 i EA j EB k EC l ED m EE n EF o FO p
1 q F2 r F3 s F4 t F5 u F6 v F7 w FB x
9 y FA z FB é FC ù FD è FE " FF 100

```

Centronics 8 bits
(France)

```

21 ! 22 " 23 # 24 $ 25 % 26 & 27 ' 28 (
29 ) 2A * 2B + 2C , 2D - 2E . 2F / 30 0
31 1 32 2 33 3 34 4 35 5 36 6 37 7 38 8
39 9 3A : 3B ; 3C < 3D = 3E > 3F ? 40 à
41 A 42 B 43 C 44 D 45 E 46 F 47 G 48 H
49 I 4A J 4B K 4C L 4D M 4E N 4F O 50 P
51 Q 52 R 53 S 54 T 55 U 56 V 57 W 58 X
59 Y 5A Z 5B ^ 5C _ 5D ` 5E ~ 5F " 60 '
61 a 62 b 63 c 64 d 65 e 66 f 67 g 68 h
69 i 6A j 6B k 6C l 6D m 6E n 6F o 70 p
71 q 72 r 73 s 74 t 75 u 76 v 77 w 78 x
79 y 7A z 7B ù 7C ù 7D ù 7E " 7F 80
A1 / A2 " A3 # A4 $ A5 % A6 & A7 ' AB (
A9 ) AA * AB + AC , AD - AE . AF / BO 0
B1 1 B2 2 B3 3 B4 4 B5 5 B6 6 B7 7 B8 8
B9 9 BA : BB ; BC < BD = BE > BF ? CO à
C1 A C2 B C3 C C4 D C5 E C6 F C7 G CB H
C9 I CA J CB K CC L CD M CE N CF O DO P
D1 Q D2 R D3 S D4 T D5 U D6 V D7 W DB X
D9 Y DA Z DB * DC ç DD $ DE ^ DF _ EO '
E1 a E2 b E3 c E4 d E5 e E6 f E7 g E8 h
E9 i EA j EB k EC l ED m EE n EF o FO p
F1 q F2 r F3 s F4 t F5 u F6 v F7 w FB x
F9 y FA z FB é FC ù FD è FE " FF 100

```

CPC 7 bits
(Allemagne)

```

1 ! 22 " 23 # 24 $ 25 % 26 & 27 ' 28 (
9 ) 2A * 2B + 2C , 2D - 2E . 2F / 30 0
1 1 32 2 33 3 34 4 35 5 36 6 37 7 38 8
9 9 3A : 3B ; 3C < 3D = 3E > 3F ? 40 $
1 A 42 B 43 C 44 D 45 E 46 F 47 G 48 H
9 I 4A J 4B K 4C L 4D M 4E N 4F O 50 P
1 Q 52 R 53 S 54 T 55 U 56 V 57 W 58 X
9 Y 5A Z 5B ^ 5C _ 5D ` 5E ~ 5F " 60 '
1 a 62 b 63 c 64 d 65 e 66 f 67 g 68 h
9 i 6A j 6B k 6C l 6D m 6E n 6F o 70 p
1 q 72 r 73 s 74 t 75 u 76 v 77 w 78 x
9 y 7A z 7B ù 7C ù 7D ù 7E " 7F 80
1 ç A2 ä A3 # A4 $ A5 % A6 & A7 ' AB (
9 ) AA * AB ù AC ü AD ß AE ü AF ä BO 0
1 1 B2 2 B3 3 B4 4 B5 5 B6 6 B7 7 B8 8
9 9 BA : BB ; BC < BD = BE > BF ? CO $
1 A C2 B C3 C C4 D C5 E C6 F C7 G CB H
9 I CA J CB K CC L CD M CE N CF O DO P
1 Q D2 R D3 S D4 T D5 U D6 V D7 W DB X
9 Y DA Z DB ä DC ö DD ü DE ^ DF _ EO '
1 a E2 b E3 c E4 d E5 e E6 f E7 g E8 h
9 i EA j EB k EC l ED m EE n EF o FO p
1 q F2 r F3 s F4 t F5 u F6 v F7 w FB x
9 y FA z FB ä FC ö FD ü FE ß FF 100

```

Centronics 8 bits
(Allemagne)

```

21 ! 22 " 23 # 24 $ 25 % 26 & 27 ' 28 (
29 ) 2A * 2B + 2C , 2D - 2E . 2F / 30 0
31 1 32 2 33 3 34 4 35 5 36 6 37 7 38 8
39 9 3A : 3B ; 3C < 3D = 3E > 3F ? 40 $
41 A 42 B 43 C 44 D 45 E 46 F 47 G 48 H
49 I 4A J 4B K 4C L 4D M 4E N 4F O 50 P
51 Q 52 R 53 S 54 T 55 U 56 V 57 W 58 X
59 Y 5A Z 5B ^ 5C _ 5D ` 5E ~ 5F " 60 '
61 a 62 b 63 c 64 d 65 e 66 f 67 g 68 h
69 i 6A j 6B k 6C l 6D m 6E n 6F o 70 p
71 q 72 r 73 s 74 t 75 u 76 v 77 w 78 x
79 y 7A z 7B ù 7C ù 7D ù 7E " 7F 80
A1 / A2 " A3 # A4 $ A5 % A6 & A7 ' AB (
A9 ) AA * AB + AC , AD - AE . AF / BO 0
B1 1 B2 2 B3 3 B4 4 B5 5 B6 6 B7 7 B8 8
B9 9 BA : BB ; BC < BD = BE > BF ? CO $
C1 A C2 B C3 C C4 D C5 E C6 F C7 G CB H
C9 I CA J CB K CC L CD M CE N CF O DO P
D1 Q D2 R D3 S D4 T D5 U D6 V D7 W DB X
D9 Y DA Z DB ä DC ö DD ü DE ^ DF _ EO '
E1 a E2 b E3 c E4 d E5 e E6 f E7 g E8 h
E9 i EA j EB k EC l ED m EE n EF o FO p
F1 q F2 r F3 s F4 t F5 u F6 v F7 w FB x
F9 y FA z FB ä FC ö FD ü FE ß FF 100

```

CPC 7 bits

(Etats-Unis)

```

21 ! 22 " 23 # 24 $ 25 % 26 & 27 ' 28 (
29 ) 2A * 2B + 2C , 2D - 2E . 2F / 30 0
31 1 32 2 33 3 34 4 35 5 36 6 37 7 38 8
39 9 3A : 3B ; 3C < 3D = 3E > 3F ? 40 @
41 A 42 B 43 C 44 D 45 E 46 F 47 G 48 H
49 I 4A J 4B K 4C L 4D M 4E N 4F O 50 P
51 Q 52 R 53 S 54 T 55 U 56 V 57 W 58 X
59 Y 5A Z 5B [ 5C \ 5D ] 5E ^ 5F _ 60 `
61 a 62 b 63 c 64 d 65 e 66 f 67 g 68 h
69 i 6A j 6B k 6C l 6D m 6E n 6F o 70 p
71 q 72 r 73 s 74 t 75 u 76 v 77 w 78 x
79 y 7A z 7B { 7C | 7D } 7E ~ 7F 80
A1 \ A2 [ A3 # A4 $ A5 % A6 & A7 ' AB (
A9 ) AA * AB + AC , AD - AE . AF / B0 0
B1 1 B2 2 B3 3 B4 4 B5 5 B6 6 B7 7 B8 8
B9 9 BA : BB ; BC < BD = BE > BF ? C0 @
C1 A C2 B C3 C C4 D C5 E C6 F C7 G C8 H
C9 I CA J CB K CC L CD M CE N CF O DO P
D1 Q D2 R D3 S D4 T D5 U D6 V D7 W DB X
D9 Y DA Z DB [ DC \ DD ] DE ^ DF _ EO `
E1 a E2 b E3 c E4 d E5 e E6 f E7 g EB h
E9 i EA j EB k EC l ED m EE n EF o FO p
F1 q F2 r F3 s F4 t F5 u F6 v F7 w FB x
F9 y FA z FB { FC | FD } FE ~ FF 100

```

Centronics 8 bits

(Etats-Unis)

```

21 ! 22 " 23 # 24 $ 25 % 26 & 27 ' 28 (
29 ) 2A * 2B + 2C , 2D - 2E . 2F / 30 0
31 1 32 2 33 3 34 4 35 5 36 6 37 7 38 8
39 9 3A : 3B ; 3C < 3D = 3E > 3F ? 40 @
41 A 42 B 43 C 44 D 45 E 46 F 47 G 48 H
49 I 4A J 4B K 4C L 4D M 4E N 4F O 50 P
51 Q 52 R 53 S 54 T 55 U 56 V 57 W 58 X
59 Y 5A Z 5B [ 5C \ 5D ] 5E ^ 5F _ 60 `
61 a 62 b 63 c 64 d 65 e 66 f 67 g 68 h
69 i 6A j 6B k 6C l 6D m 6E n 6F o 70 p
71 q 72 r 73 s 74 t 75 u 76 v 77 w 78 x
79 y 7A z 7B { 7C | 7D } 7E ~ 7F 80
A1 \ A2 [ A3 # A4 $ A5 % A6 & A7 ' AB (
A9 ) AA * AB + AC , AD - AE . AF / B0 0
B1 1 B2 2 B3 3 B4 4 B5 5 B6 6 B7 7 B8 8
B9 9 BA : BB ; BC < BD = BE > BF ? C0 @
C1 A C2 B C3 C C4 D C5 E C6 F C7 G C8 H
C9 I CA J CB K CC L CD M CE N CF O DO P
D1 Q D2 R D3 S D4 T D5 U D6 V D7 W DB X
D9 Y DA Z DB [ DC \ DD ] DE ^ DF _ EO `
E1 a E2 b E3 c E4 d E5 e E6 f E7 g EB h
E9 i EA j EB k EC l ED m EE n EF o FO p
F1 q F2 r F3 s F4 t F5 u F6 v F7 w FB x
F9 y FA z FB { FC | FD } FE ~ FF 100

```

CPC 7 bits
(Grande Bretagne)

```

21 ! 22 " 23 £ 24 $ 25 % 26 & 27 ' 28 (
29 ) 2A * 2B + 2C , 2D - 2E . 2F / 30 0
31 1 32 2 33 3 34 4 35 5 36 6 37 7 38 8
39 9 3A : 3B ; 3C < 3D = 3E > 3F ? 40 @
41 A 42 B 43 C 44 D 45 E 46 F 47 G 48 H
49 I 4A J 4B K 4C L 4D M 4E N 4F O 50 P
51 Q 52 R 53 S 54 T 55 U 56 V 57 W 58 X
59 Y 5A Z 5B [ 5C \ 5D ] 5E ^ 5F _ 60 `
61 a 62 b 63 c 64 d 65 e 66 f 67 g 68 h
69 i 6A j 6B k 6C l 6D m 6E n 6F o 70 p
71 q 72 r 73 s 74 t 75 u 76 v 77 w 78 x
79 y 7A z 7B { 7C | 7D } 7E ~ 7F 80
A1 \ A2 [ A3 £ A4 $ A5 % A6 & A7 ' AB (
A9 ) AA * AB + AC , AD - AE . AF / B0 0
B1 1 B2 2 B3 3 B4 4 B5 5 B6 6 B7 7 B8 8
B9 9 BA : BB ; BC < BD = BE > BF ? C0 @
C1 A C2 B C3 C C4 D C5 E C6 F C7 G C8 H
C9 I CA J CB K CC L CD M CE N CF O DO P
D1 Q D2 R D3 S D4 T D5 U D6 V D7 W DB X
D9 Y DA Z DB [ DC \ DD ] DE ^ DF _ EO `
E1 a E2 b E3 c E4 d E5 e E6 f E7 g EB h
E9 i EA j EB k EC l ED m EE n EF o FO p
F1 q F2 r F3 s F4 t F5 u F6 v F7 w FB x
F9 y FA z FB { FC | FD } FE ~ FF 100

```

Centronics 8 bits
(Grande Bretagne)

```

21 ! 22 " 23 £ 24 $ 25 % 26 & 27 ' 28 (
29 ) 2A * 2B + 2C , 2D - 2E . 2F / 30 0
31 1 32 2 33 3 34 4 35 5 36 6 37 7 38 8
39 9 3A : 3B ; 3C < 3D = 3E > 3F ? 40 @
41 A 42 B 43 C 44 D 45 E 46 F 47 G 48 H
49 I 4A J 4B K 4C L 4D M 4E N 4F O 50 P
51 Q 52 R 53 S 54 T 55 U 56 V 57 W 58 X
59 Y 5A Z 5B [ 5C \ 5D ] 5E ^ 5F _ 60 `
61 a 62 b 63 c 64 d 65 e 66 f 67 g 68 h
69 i 6A j 6B k 6C l 6D m 6E n 6F o 70 p
71 q 72 r 73 s 74 t 75 u 76 v 77 w 78 x
79 y 7A z 7B { 7C | 7D } 7E ~ 7F 80
A1 \ A2 [ A3 £ A4 $ A5 % A6 & A7 ' AB (
A9 ) AA * AB + AC , AD - AE . AF / B0 0
B1 1 B2 2 B3 3 B4 4 B5 5 B6 6 B7 7 B8 8
B9 9 BA : BB ; BC < BD = BE > BF ? C0 @
C1 A C2 B C3 C C4 D C5 E C6 F C7 G C8 H
C9 I CA J CB K CC L CD M CE N CF O DO P
D1 Q D2 R D3 S D4 T D5 U D6 V D7 W DB X
D9 Y DA Z DB [ DC \ DD ] DE ^ DF _ EO `
E1 a E2 b E3 c E4 d E5 e E6 f E7 g EB h
E9 i EA j EB k EC l ED m EE n EF o FO p
F1 q F2 r F3 s F4 t F5 u F6 v F7 w FB x
F9 y FA z FB { FC | FD } FE ~ FF 100

```

Partie 9 : Programmes

CPC 7 bits

(Italie)

```

21 ! 22 " 23 # 24 $ 25 % 26 & 27 ' 28 (
29 ) 2A * 2B + 2C , 2D - 2E . 2F / 30 0
31 1 32 2 33 3 34 4 35 5 36 6 37 7 38 8
39 9 3A : 3B ; 3C < 3D = 3E > 3F ? 40 @
41 A 42 B 43 C 44 D 45 E 46 F 47 G 48 H
49 I 4A J 4B K 4C L 4D M 4E N 4F O 50 P
51 Q 52 R 53 S 54 T 55 U 56 V 57 W 58 X
59 Y 5A Z 5B * 5C \ 5D ^ 5E ^ 5F _ 60 `
61 a 62 b 63 c 64 d 65 e 66 f 67 g 68 h
69 i 6A j 6B k 6C l 6D m 6E n 6F o 70 p
71 q 72 r 73 s 74 t 75 u 76 v 77 w 78 x
79 y 7A z 7B à 7C b 7D è 7E ì 7F 80
81 \ 82 à 83 # 84 $ 85 % 86 & 87 ' 88 (
89 ) 8A * 8B + 8C , 8D - 8E . 8F / 90 0
91 1 92 2 93 3 94 4 95 5 96 6 97 7 98 8
99 9 9A : 9B ; 9C < 9D = 9E > 9F ? 100 @
1 A 2 B 3 C 4 D 5 E 6 F 7 G 8 H
9 I 10 J 11 K 12 L 13 M 14 N 15 O 16 P
17 Q 18 R 19 S 20 T 21 U 22 V 23 W 24 X
25 Y 26 Z 27 DB * 28 DC \ 29 DD ^ 30 DE ^ 31 DF _ 32 EO `
33 a 34 b 35 c 36 d 37 e 38 f 39 g 40 h
41 i 42 j 43 k 44 l 45 m 46 n 47 o 48 p
49 q 50 r 51 s 52 t 53 u 54 v 55 w 56 x
57 y 58 z 59 à 60 b 61 d 62 è 63 ì 64 100

```

Centronics 8 bits

(Italie)

```

21 ! 22 " 23 # 24 $ 25 % 26 & 27 ' 28 (
29 ) 2A * 2B + 2C , 2D - 2E . 2F / 30 0
31 1 32 2 33 3 34 4 35 5 36 6 37 7 38 8
39 9 3A : 3B ; 3C < 3D = 3E > 3F ? 40 @
41 A 42 B 43 C 44 D 45 E 46 F 47 G 48 H
49 I 4A J 4B K 4C L 4D M 4E N 4F O 50 P
51 Q 52 R 53 S 54 T 55 U 56 V 57 W 58 X
59 Y 5A Z 5B * 5C \ 5D ^ 5E ^ 5F _ 60 `
61 a 62 b 63 c 64 d 65 e 66 f 67 g 68 h
69 i 6A j 6B k 6C l 6D m 6E n 6F o 70 p
71 q 72 r 73 s 74 t 75 u 76 v 77 w 78 x
79 y 7A z 7B à 7C b 7D è 7E ì 7F 80
81 / 82 " 83 # 84 $ 85 % 86 & 87 ' 88 (
89 ) 8A * 8B + 8C , 8D - 8E . 8F / 90 0
91 1 92 2 93 3 94 4 95 5 96 6 97 7 98 8
99 9 9A : 9B ; 9C < 9D = 9E > 9F ? 100 @
1 A 2 B 3 C 4 D 5 E 6 F 7 G 8 H
9 I 10 J 11 K 12 L 13 M 14 N 15 O 16 P
17 Q 18 R 19 S 20 T 21 U 22 V 23 W 24 X
25 Y 26 Z 27 DB * 28 DC \ 29 DD ^ 30 DE ^ 31 DF _ 32 EO `
33 a 34 b 35 c 36 d 37 e 38 f 39 g 40 h
41 i 42 j 43 k 44 l 45 m 46 n 47 o 48 p
49 q 50 r 51 s 52 t 53 u 54 v 55 w 56 x
57 y 58 z 59 à 60 b 61 d 62 è 63 ì 64 100

```

CPC 7 bits

(Espagne)

```

21 ! 22 " 23 # 24 $ 25 % 26 & 27 ' 28 (
29 ) 2A * 2B + 2C , 2D - 2E . 2F / 30 0
31 1 32 2 33 3 34 4 35 5 36 6 37 7 38 8
39 9 3A : 3B ; 3C < 3D = 3E > 3F ? 40 @
41 A 42 B 43 C 44 D 45 E 46 F 47 G 48 H
49 I 4A J 4B K 4C L 4D M 4E N 4F O 50 P
51 Q 52 R 53 S 54 T 55 U 56 V 57 W 58 X
59 Y 5A Z 5B ; 5C Ñ 5D ¿ 5E ^ 5F _ 60 `
61 a 62 b 63 c 64 d 65 e 66 f 67 g 68 h
69 i 6A j 6B k 6C l 6D m 6E n 6F o 70 p
71 q 72 r 73 s 74 t 75 u 76 v 77 w 78 x
79 y 7A z 7B " 7C Ñ 7D } 7E ~ 7F 80
81 Ñ 82 " 83 # 84 $ 85 % 86 & 87 ' 88 (
89 ) 8A * 8B + 8C , 8D - 8E . 8F / 90 0
91 1 92 2 93 3 94 4 95 5 96 6 97 7 98 8
99 9 9A : 9B ; 9C < 9D = 9E > 9F ? 100 @
1 A 2 B 3 C 4 D 5 E 6 F 7 G 8 H
9 I 10 J 11 K 12 L 13 M 14 N 15 O 16 P
17 Q 18 R 19 S 20 T 21 U 22 V 23 W 24 X
25 Y 26 Z 27 DB / 28 DC Ñ 29 DD ¿ 30 DE ^ 31 DF _ 32 EO `
33 a 34 b 35 c 36 d 37 e 38 f 39 g 40 h
41 i 42 j 43 k 44 l 45 m 46 n 47 o 48 p
49 q 50 r 51 s 52 t 53 u 54 v 55 w 56 x
57 y 58 z 59 à 60 Ñ 61 FD } 62 FE ~ 63 FF 100

```

Centronics 8 bits

(Espagne)

```

21 ! 22 " 23 # 24 $ 25 % 26 & 27 ' 28 (
29 ) 2A * 2B + 2C , 2D - 2E . 2F / 30 0
31 1 32 2 33 3 34 4 35 5 36 6 37 7 38 8
39 9 3A : 3B ; 3C < 3D = 3E > 3F ? 40 @
41 A 42 B 43 C 44 D 45 E 46 F 47 G 48 H
49 I 4A J 4B K 4C L 4D M 4E N 4F O 50 P
51 Q 52 R 53 S 54 T 55 U 56 V 57 W 58 X
59 Y 5A Z 5B ; 5C Ñ 5D ¿ 5E ^ 5F _ 60 `
61 a 62 b 63 c 64 d 65 e 66 f 67 g 68 h
69 i 6A j 6B k 6C l 6D m 6E n 6F o 70 p
71 q 72 r 73 s 74 t 75 u 76 v 77 w 78 x
79 y 7A z 7B " 7C Ñ 7D } 7E ~ 7F 80
81 / 82 " 83 # 84 $ 85 % 86 & 87 ' 88 (
89 ) 8A * 8B + 8C , 8D - 8E . 8F / 90 0
91 1 92 2 93 3 94 4 95 5 96 6 97 7 98 8
99 9 9A : 9B ; 9C < 9D = 9E > 9F ? 100 @
1 A 2 B 3 C 4 D 5 E 6 F 7 G 8 H
9 I 10 J 11 K 12 L 13 M 14 N 15 O 16 P
17 Q 18 R 19 S 20 T 21 U 22 V 23 W 24 X
25 Y 26 Z 27 DB / 28 DC Ñ 29 DD ¿ 30 DE ^ 31 DF _ 32 EO `
33 a 34 b 35 c 36 d 37 e 38 f 39 g 40 h
41 i 42 j 43 k 44 l 45 m 46 n 47 o 48 p
49 q 50 r 51 s 52 t 53 u 54 v 55 w 56 x
57 y 58 z 59 à 60 Ñ 61 FD } 62 FE ~ 63 FF 100

```

CPC 7 bits (Danemark)

```

21 ! 22 " 23 # 24 $ 25 % 26 & 27 ' 28 (
29 ) 2A * 2B + 2C , 2D - 2E . 2F / 30 0
31 1 32 2 33 3 34 4 35 5 36 6 37 7 38 8
39 9 3A : 3B ; 3C < 3D = 3E > 3F ? 40 @
41 A 42 B 43 C 44 D 45 E 46 F 47 G 48 H
49 I 4A J 4B K 4C L 4D M 4E N 4F O 50 P
51 Q 52 R 53 S 54 T 55 U 56 V 57 W 58 X
59 Y 5A Z 5B [ 5C \ 5D ^ 5E _ 60 `
61 a 62 b 63 c 64 d 65 e 66 f 67 g 68 h
69 i 6A j 6B k 6C l 6D m 6E n 6F o 70 p
71 q 72 r 73 s 74 t 75 u 76 v 77 w 78 x
79 y 7A z 7B [ 7C \ 7D ^ 7E _ 7F 80
A1 0 A2 1 A3 2 A4 3 A5 4 A6 5 A7 6 A8 7
A9 8 AA * AB + AC , AD - AE . AF / B0 0
B1 1 B2 2 B3 3 B4 4 B5 5 B6 6 B7 7 B8 8
B9 9 BA : BB ; BC < BD = BE > BF ? C0 @
C1 A C2 B C3 C C4 D C5 E C6 F C7 G C8 H
C9 I CA J CB K CC L CD M CE N CF O DO P
D1 Q D2 R D3 S D4 T D5 U D6 V D7 W DB X
D9 Y DA Z DB [ DC \ DD ^ DE _ DF ` EO `
E1 a E2 b E3 c E4 d E5 e E6 f E7 g EB h
E9 i EA j EB k EC l ED m EE n EF o FO p
F1 q F2 r F3 s F4 t F5 u F6 v F7 w FB x
F9 y FA z FB [ FC \ FD ^ FE _ FF 100

```

Centronics 8 bits (Danemark)

```

21 ! 22 " 23 # 24 $ 25 % 26 & 27 ' 28 (
29 ) 2A * 2B + 2C , 2D - 2E . 2F / 30 0
31 1 32 2 33 3 34 4 35 5 36 6 37 7 38 8
39 9 3A : 3B ; 3C < 3D = 3E > 3F ? 40 @
41 A 42 B 43 C 44 D 45 E 46 F 47 G 48 H
49 I 4A J 4B K 4C L 4D M 4E N 4F O 50 P
51 Q 52 R 53 S 54 T 55 U 56 V 57 W 58 X
59 Y 5A Z 5B [ 5C \ 5D ^ 5E _ 60 `
61 a 62 b 63 c 64 d 65 e 66 f 67 g 68 h
69 i 6A j 6B k 6C l 6D m 6E n 6F o 70 p
71 q 72 r 73 s 74 t 75 u 76 v 77 w 78 x
79 y 7A z 7B [ 7C \ 7D ^ 7E _ 7F 80
A1 / A2 " A3 # A4 $ A5 % A6 & A7 ' AB (
A9 ) AA * AB + AC , AD - AE . AF / B0 0
B1 1 B2 2 B3 3 B4 4 B5 5 B6 6 B7 7 B8 8
B9 9 BA : BB ; BC < BD = BE > BF ? C0 @
C1 A C2 B C3 C C4 D C5 E C6 F C7 G C8 H
C9 I CA J CB K CC L CD M CE N CF O DO P
D1 Q D2 R D3 S D4 T D5 U D6 V D7 W DB X
D9 Y DA Z DB [ DC \ DD ^ DE _ DF ` EO `
E1 a E2 b E3 c E4 d E5 e E6 f E7 g EB h
E9 i EA j EB k EC l ED m EE n EF o FO p
F1 q F2 r F3 s F4 t F5 u F6 v F7 w FB x
F9 y FA z FB [ FC \ FD ^ FE _ FF 100

```

CPC 7 bits (Suède)

```

21 ! 22 " 23 # 24 $ 25 % 26 & 27 ' 28 (
29 ) 2A * 2B + 2C , 2D - 2E . 2F / 30 0
31 1 32 2 33 3 34 4 35 5 36 6 37 7 38 8
39 9 3A : 3B ; 3C < 3D = 3E > 3F ? 40 @
41 A 42 B 43 C 44 D 45 E 46 F 47 G 48 H
49 I 4A J 4B K 4C L 4D M 4E N 4F O 50 P
51 Q 52 R 53 S 54 T 55 U 56 V 57 W 58 X
59 Y 5A Z 5B [ 5C \ 5D ^ 5E _ 60 `
61 a 62 b 63 c 64 d 65 e 66 f 67 g 68 h
69 i 6A j 6B k 6C l 6D m 6E n 6F o 70 p
71 q 72 r 73 s 74 t 75 u 76 v 77 w 78 x
79 y 7A z 7B [ 7C \ 7D ^ 7E _ 7F 80
A1 0 A2 1 A3 2 A4 3 A5 4 A6 5 A7 6 A8 7
A9 8 AA * AB + AC , AD - AE . AF / B0 0
B1 1 B2 2 B3 3 B4 4 B5 5 B6 6 B7 7 B8 8
B9 9 BA : BB ; BC < BD = BE > BF ? C0 @
C1 A C2 B C3 C C4 D C5 E C6 F C7 G C8 H
C9 I CA J CB K CC L CD M CE N CF O DO P
D1 Q D2 R D3 S D4 T D5 U D6 V D7 W DB X
D9 Y DA Z DB [ DC \ DD ^ DE _ DF ` EO `
E1 a E2 b E3 c E4 d E5 e E6 f E7 g EB h
E9 i EA j EB k EC l ED m EE n EF o FO p
F1 q F2 r F3 s F4 t F5 u F6 v F7 w FB x
F9 y FA z FB [ FC \ FD ^ FE _ FF 100

```

Centronics 8 bits (Suède)

```

21 ! 22 " 23 # 24 $ 25 % 26 & 27 ' 28 (
29 ) 2A * 2B + 2C , 2D - 2E . 2F / 30 0
31 1 32 2 33 3 34 4 35 5 36 6 37 7 38 8
39 9 3A : 3B ; 3C < 3D = 3E > 3F ? 40 @
41 A 42 B 43 C 44 D 45 E 46 F 47 G 48 H
49 I 4A J 4B K 4C L 4D M 4E N 4F O 50 P
51 Q 52 R 53 S 54 T 55 U 56 V 57 W 58 X
59 Y 5A Z 5B [ 5C \ 5D ^ 5E _ 60 `
61 a 62 b 63 c 64 d 65 e 66 f 67 g 68 h
69 i 6A j 6B k 6C l 6D m 6E n 6F o 70 p
71 q 72 r 73 s 74 t 75 u 76 v 77 w 78 x
79 y 7A z 7B [ 7C \ 7D ^ 7E _ 7F 80
A1 / A2 " A3 # A4 $ A5 % A6 & A7 ' AB (
A9 ) AA * AB + AC , AD - AE . AF / B0 0
B1 1 B2 2 B3 3 B4 4 B5 5 B6 6 B7 7 B8 8
B9 9 BA : BB ; BC < BD = BE > BF ? C0 @
C1 A C2 B C3 C C4 D C5 E C6 F C7 G C8 H
C9 I CA J CB K CC L CD M CE N CF O DO P
D1 Q D2 R D3 S D4 T D5 U D6 V D7 W DB X
D9 Y DA Z DB [ DC \ DD ^ DE _ DF ` EO `
E1 a E2 b E3 c E4 d E5 e E6 f E7 g EB h
E9 i EA j EB k EC l ED m EE n EF o FO p
F1 q F2 r F3 s F4 t F5 u F6 v F7 w FB x
F9 y FA z FB [ FC \ FD ^ FE _ FF 100

```


I. Éléments de base sur la définition de caractères graphiques

Quatre commandes ESCape permettent d'envoyer des caractères graphiques à l'imprimante DMP 2000. Sur les ordinateurs qui possèdent un port Centronics 7 bits (comme les CPC), ces caractères sont inscrits dans une grille de 7 lignes sur 6 colonnes organisée comme suit :

Poids	1	3	5	7	9	11	<— Octet
64							
32							
16							
8							
4							
2							
1							

Chaque colonne est codée sur un octet de valeur comprise entre 0 et 127. Par exemple, le caractère suivant sera défini comme suit :

64		x	x	x	x	
32	x					x
16	x					x
8	x					x
4		x	x	x	x	
2						
1	x	x	x	x	x	x

Codage du caractère

Colonne 1 : $1 + 8 + 16 + 32 = 57$

Colonne 2 : $1 + 4 + 64 = 69$

Colonne 3 : $1 + 4 + 64 = 69$

Colonne 4 : $1 + 4 + 64 = 69$

Colonne 5 : $1 + 4 + 64 = 69$

Colonne 6 : $1 + 8 + 16 + 32 = 57$

L'impression de caractères graphiques peut se faire dans l'un des modes suivants :

- Simple densité : ESC K n1 n2
(où n1 est le nombre de colonnes à imprimer et n2 vaut 0)

- Double densité : ESC L n1 n2
- Double densité double vitesse : ESC Z n1 n2
- Quadruple densité : ESC Z n1 n2

Le caractère défini ci-dessus sera imprimé grâce aux commandes Basic suivantes :

Simple densité

```
10 PRINT #8,CHR$(27);"K";CHR$(12);CHR$(0);
20 PRINT #8,CHR$(57);CHR$(0);CHR$(69);CHR$(0);CHR$(69);CHR$(0);CHR$(69);CHR$(0);
CHR$(69);CHR$(0);CHR$(57);CHR$(0)
```

Double densité

```
10 PRINT #8,CHR$(27);"L";CHR$(12);CHR$(0);
20 PRINT #8,CHR$(57);CHR$(0);CHR$(69);CHR$(0);CHR$(69);CHR$(0);CHR$(69);CHR$(0);
CHR$(69);CHR$(0);CHR$(57);CHR$(0)
```

Double densité double vitesse

```
10 PRINT #8,CHR$(27);"Y";CHR$(12);CHR$(0);
20 PRINT #8,CHR$(57);CHR$(0);CHR$(69);CHR$(0);CHR$(69);CHR$(0);CHR$(69);CHR$(0);
CHR$(69);CHR$(0);CHR$(57);CHR$(0)
```

Quadruple densité

```
10 PRINT #8,CHR$(27);"Z";CHR$(12);CHR$(0);
20 PRINT #8,CHR$(57);CHR$(0);CHR$(69);CHR$(0);CHR$(69);CHR$(0);CHR$(69);CHR$(0);
CHR$(69);CHR$(0);CHR$(57);CHR$(0)
```

Sur les ordinateurs qui possèdent un port Centronics 8 bits (comme les PC et compatibles par exemple), ces caractères sont inscrits dans une grille de 8 lignes sur 6 colonnes organisée comme suit :

Poids	1	3	5	7	9	11	< — Octet
128							
64							
32							
16							
8							
4							
2							
1							

Chaque colonne est codée sur un octet de valeur comprise entre 0 et 255.

II. Définition de caractères graphiques

Il est assez aisé de définir de nouveaux caractères graphiques. Afin de rendre automatique cette définition, nous vous présentons un petit utilitaire écrit en Basic. Grâce à lui, vous pourrez définir sans aucun calcul vos caractères en positionnant ou en effaçant des repères dans une grille de 7 lignes sur 6 colonnes.

Lorsque vous lancez le programme, vous devez entrer dans un premier temps le code ASCII du caractère qui va être redéfini. Par exemple, si le caractère « Accolade ouvrante » de code ASCII 123 ne vous convient pas, répondez 123 à cette question.

Une grille 7 lignes sur 6 colonnes apparaît. Sous cette grille se trouvent les instructions d'utilisation :

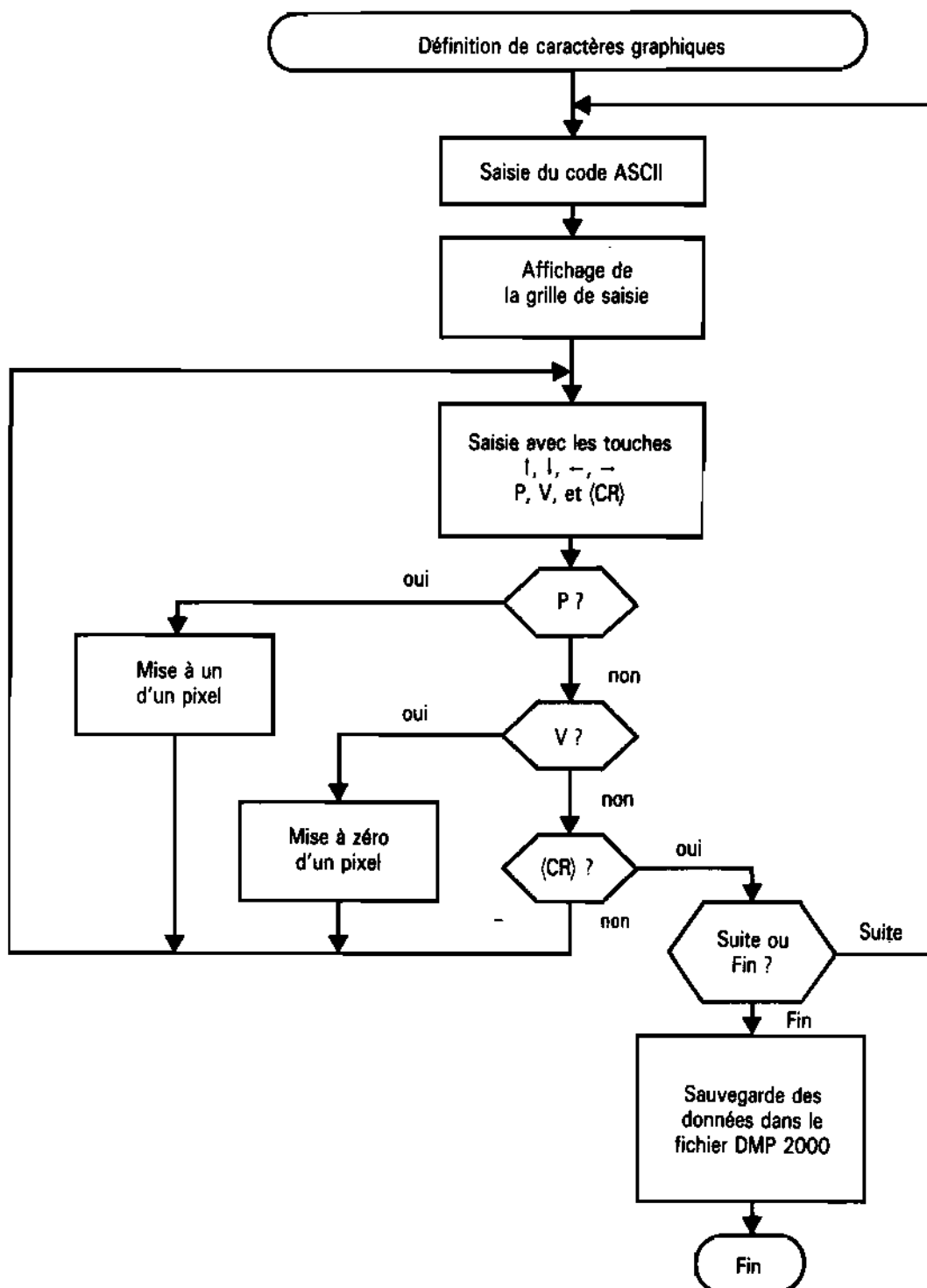
Utilisez :

- les touches flèches pour déplacer le curseur dans la grille ;
- la touche **P** pour positionner un point ;
- la touche **V** pour effacer un point.

Appuyez sur **<Enter>** lorsque la définition est terminée.

Le programme vous demande alors si vous désirez redéfinir d'autres caractères. Répondez **O** (oui) tant que vous aurez des caractères à redéfinir et **N** (non) lorsque vous aurez défini le dernier caractère. Le fichier DMP2000.BIN est alors créé. Ce fichier est utilisé par le driver d'imprimante. Il contient toutes les données correspondant aux caractères que vous avez redéfinis. Lorsqu'un texte sera imprimé à travers le driver, tous les caractères dont les codes ASCII sont redéfinis dans le fichier DMP2000 seront imprimés en tant que caractères graphiques.

La logique du programme de définition de caractères graphiques est la suivante :



Le programme de définition de caractères est le suivant :

```

1000 CLS
1010 DIM TX(6,7,50),TCA%(50)
1020 K=0
1030 MODE 2
1040 PRINT "Definition de caracteres graphiques DMP 2000:"
1050 PRINT "-----"
1060 PRINT:INPUT"Code ASCII du caractere : ";CA%
1070 K=K+1:TCA%(K)=CA%
1080 CLS
1090 FOR I=1 TO 7
1100   PRINT". . . . ."
1110 NEXT I
1120 LOCATE 1,10:PRINT"Utilisez les touches fleches pour vous deplacer,"
1130 PRINT"P pour allumer un point, et V pour l'eteindre."
1140 I=1 : J=1
1150 A%=INKEY% : IF A%="" THEN 1150
1160 A=ASC(A%)
1170 IF A=242 THEN 2010
1180 IF A=243 THEN 3010
1190 IF A=240 THEN 4010
1200 IF A=241 THEN 5010
1210 IF UPPER$(A%)="P" THEN LOCATE I*2-1,J:PRINT"X";:TX(I,J,K)=1
1220 IF UPPER$(A%)="V" THEN LOCATE I*2-1,J:PRINT".":TX(I,J,K)=0
1230 IF A=13 THEN 6030
1240 GOTO 1150
2000 REM -----
2010 REM Deplacement vers la gauche
2020 REM -----
2030 IF I=1 THEN 1150
2040 I=I-1
2050 GOTO 1150
3000 REM -----
3010 REM Deplacement vers la droite
3020 REM -----
3030 IF I=6 THEN 1150
3040 I=I+1
3050 GOTO 1150
4000 REM -----
4010 REM Deplacement vers le haut
4020 REM -----
4030 IF J=1 THEN 1150
4040 J=J-1
4050 GOTO 1150
5000 REM -----
5010 REM Deplacement vers le bas
5020 REM -----
5030 IF J=7 THEN 1150
5040 J=J+1
5050 GOTO 1150
6000 REM -----
6010 REM Suite ou fin de definition
6020 REM -----
6030 LOCATE 1,14
6040 INPUT "Definition d'un autre caractere (O/N) : ";R%
6050 IF UPPER$(R%)="O" THEN 1030
7000 REM -----
7010 REM Sauvegarde des donnees
7020 REM -----
7030 FOR I=1 TO K
7040   POKE &70B0+I-1,TCA%(I)
7050 NEXT I
7060 POKE &70B0+I-1,255 'Terminateur
7070 FOR I=1 TO K
7080   FOR J=1 TO 6
7090     VA=0
7100     FOR L=1 TO 7
7110       VA=VA+TX(I,J,L)*2^(7-J)
7120       POKE &71B0+(L-1)*12+(I-1)*2,VA
7130       POKE &71B0+(L-1)*12+(I-1)*2+1,0

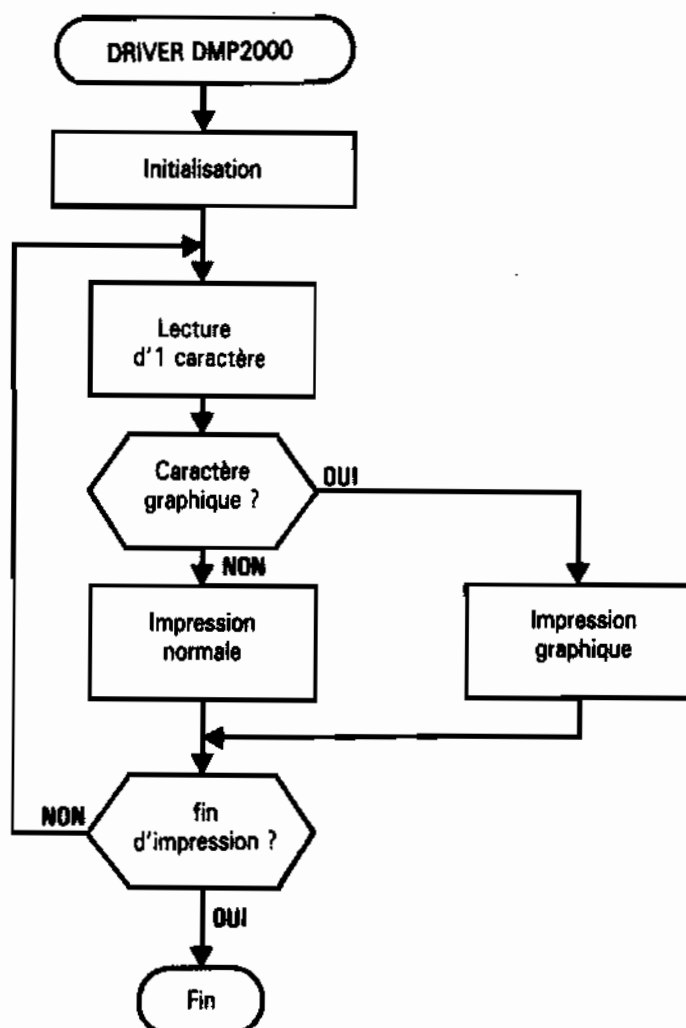
```

(programme de définition de caractère suite)

```
7140     NEXT J
7150     NEXT I
7160     NEXT L
7170     L=255+K*12+1 'Calcul de la longueur des données
7180     SAVE"DMP2000",B,&7080,L
```

III. Impression de textes en utilisant un driver d'imprimante

Les caractères graphiques définis à l'aide de l'utilitaire précédent se trouvent maintenant dans le fichier DMP2000.BIN. Pour imprimer un texte ASCII en envoyant des données correctes à l'imprimante (codes ASCII des caractères non redéfinis et séquence ESCape pour les caractères redéfinis), le driver doit analyser chaque caractère à imprimer, comme le montre l'ordinogramme suivant :



Comme le calcul et l'émission des caractères doivent se dérouler assez rapidement (à une vitesse supérieure ou égale à celle de l'impression), le driver ne peut pas être totalement écrit en Basic. Nous avons développé la partie d'analyse des caractères et d'émission de codes vers l'imprimante en Assembleur. Ce court programme utilise la macro MC PRINT CHAR du Firmware des CPC. Reportez-vous en Partie 4 chapitre 27 page 62 pour avoir plus de détails à ce sujet.

Le programme Assembleur est le suivant :

```

1                      ORG 7000H
2                      LOAD 7000H
3                      ;-----
4                      ; Initialisation
5                      ;-----
6 7000 ED5BFE6F          LD DE,(IMPL)          ;à implantation
7 7004 3AFD6F           LD A,(LGR)
8 7007 4F              LD C,A                ;Longueur ligne
9 700B 0C              INC C
10                     ;-----
11                     ;Boucle principale
12                     ;-----
13 BOU1:               EQU $
14 7009 3E01           LD A,1
15 700B 327F70         LD (POS),A            ;Pos du caract
16 700E 21B070         LD HL,GRAPH           ;Codes ASCII car graph
17 7011 1A             LD A,(DE)            ;Caractere courant
18 7012 47             LD B,A
19 7013 0D             DEC C
20 7014 2B5A           JR Z,FINLIG           ;Fin de ligne
21                     ;
22 BOU2:               EQU $
23 7016 7E             LD A,(HL)
24 7017 FEFF           CP OFFH
25 7019 2B0D           JR Z,IMPNOT           ;Impression normale

```

```

26 701B BB          CP    B
27 701C 2B13        JR    Z,IMPSPE          ;Impression speciale
28 701E 23          INC    HL                ;Car graph suivant
29 701F 3A7F70      LD     A,(POS)
30 7022 3C          INC     A
31 7023 327F70      LD     (POS),A
32 7026 1BEE        JR     BOU2              ;Boucle
33                  ;-----
34                  ;Impression normale
35                  ;-----
36  IMPNOR:         EQU    $
37 7028 7B          LD     A,B
38 7029 CD2BB0      CALL  MCPRINT            ;Emission d'i caract
39 702C 30FA        JR     NC,IMPNOR
40 702E 13          INC     DE                ;Caractere suivant
41 702F 1BDB        JR     BOU1
42                  ;-----
43                  ;Impression speciale
44                  ;-----
45  IMPSPE:         EQU    $
46 7031 21B071      LD     HL,DGRAPH         ;à Donnees graph
47 7034 C5          PUSH   BC
48 7035 010C00      LD     BC,12
49  SPE0:           EQU    $
50 7038 3A7F70      LD     A,(POS)
51 703B 3D          DEC     A
52 703C 327F70      LD     (POS),A
53 703F 2B03        JR     Z,SPE1            ;Plus d'incrementation
54 7041 09          ADD     HL,BC
55 7042 1BF4        JR     SPE0              ;Boucle d'incrementation
56  SPE1:           EQU    $

```


Partie 9 : Programmes

```

57 7044 3E1B          LD  A,27
58 7046 CD2BBD        CALL MCPRINT          ;Emission ESC
59 7049 30F9          JR   NC,SPE1
60                   SPE4: EQU  $
61 704B 3E4C          LD  A,"L"
62 704D CD2BBD        CALL MCPRINT          ;Emission "L"
63 7050 30F9          JR   NC,SPE4
64                   SPE5: EQU  $
65 7052 3E0C          LD  A,12
66 7054 CD2BBD        CALL MCPRINT          ;Emission ASC 12
67 7057 30F9          JR   NC,SPE5
68                   SPE6: EQU  $
69 7059 3E00          LD  A,0
70 705B CD2BBD        CALL MCPRINT
71 705E 30F9          JR   NC,SPE6
72                   ;
73 7060 060C          LD  B,12
74                   ;Emission de 12 donnees graphiques
75                   SPE2: EQU  $
76 7062 7E           LD  A,(HL)
77 7063 CD2BBD        CALL MCPRINT
78 7066 30FA          JR   NC,SPE2
79 7068 23           INC  HL
80 7069 05           DEC  B
81 706A 20F6          JR   NZ,SPE2
82 706C C1           POP  BC
83 706D 13           INC  DE          ;Donnee suivante
84 706E 1899          JR   BDU1
85                   ;
86                   FINLIG: EQU  $
87 7070 3E0D          LD  A,CR
88 7072 CD2BBD        CALL MCPRINT

```

```

89 7075 30F9          JR    NC,FINLIG
90                   BIS:   EQU    *
91 7077 3E0A          LD     A,LF
92 7079 CD2BBD        CALL MCPRINT
93 707C 30F9          JR     NC,BIS
94 707E C9            RET

95
96                   ;-----
97                   ; ZONE DES EQU, des DB et DS
98                   ;-----
99                   CR:     EQU    13           ;Carriage Return
100                  LF:     EQU    10           ;Line Feed
101                  LGR:     EQU    6FFDH        ;Longueur ligne
102                  IMPL:     EQU    6FFEH        ;à Implant ligne
103                  MCPRINT: EQU    0BD2BH        ;MC PRINT CHAR
104                  ;
105                  POS:     DS     1           ;Position dans GRAPH
106
107                  ;=====
108                  ;Codes ASCII des caract graphiques
109                  ;=====
110                  ORG     7080H
111                  LOAD    7080H
112                  GRAPH:   EQU    *
113
114                  ;=====
115                  ; Paquets de 12 donnees graphiques
116                  ;=====
117                  ORG     7180H
118                  LOAD    7180H
119                  DGRAPH:  EQU    *           ;Donnees corresp.
120                  END

```

Ce programme est incorporé dans un programme Basic dont voici le listing :

```

1000 REM DRIVER D'IMPRIMANTE DMP 2000
1010 REM =====
1020 FOR I=&7000 TO &707F
1030   READ A$ 'Lecture d'une donnee
1040   A$="&"+A$
1050   POKE I,VAL(A$) 'Memorisation
1060 NEXT I
1070 '
1080 LOAD"DMP2000" 'Chargement des donnees graphiques
1090 '
1100 CLS
1110 PRINT "Driver d'imprimante DMP 2000"
1120 PRINT "-----"
1130 INPUT"Nom du fichier a imprimer ";f$
1140 OPENIN f$
1150 LINE INPUT #9,a$ 'Lecture d'une ligne
1160 A=AA$ 'Adresse d'implantation de la variable
1170 POKE &6FFD,PEEK(A) 'Longueur de la ligne a imprimer
1180 POKE &6FFE,PEEK(A+1) 'LSB Adresse de debut de ligne
1190 POKE &6FFF,PEEK(A+2) 'MSB Adresse de debut de ligne
1200 CALL &7000 'Lancement du driver
1210 IF EOF=0 THEN 1150 'Poursuite du traitement des lignes
1220 CLOSEIN 'Fermeture du fichier texte a imprimer
1230 '
1240 '-----
1250 'Driver d'impression Assembleur
1260 '-----
1270 DATA ED,5B,FE,6F,3A,FD,6F,4F,C,3E,1,32,7F,70,21,80
1280 DATA 70,1A,47,D,28,5A,7E,FE,FF,28,D,B8,2B,13,23,3A
1290 DATA 7F,70,3C,32,7F,70,18,EE,78,CD,2B,BD,30,FA,13,18
1300 DATA DB,21,80,71,C5,1,C,0,3A,7F,70,3D,32,7F,70,28
1310 DATA 3,9,18,F4,3E,1B,CD,2B,BD,30,F9,3E,4C,CD,2B,BD
1320 DATA 30,F9,3E,C,CD,2B,BD,30,F9,3E,0,CD,2B,BD,30,F9
1330 DATA 6,C,7E,CD,2B,BD,30,FA,23,5,20,F6,C1,13,18,99
1340 DATA 3E,D,CD,2B,BD,30,F9,3E,A,CD,2B,BD,30,F9,C9,0

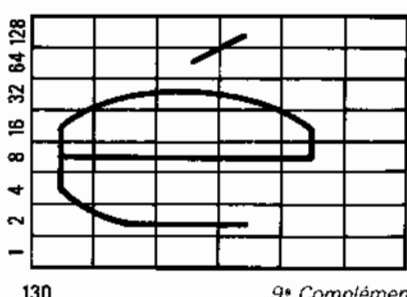
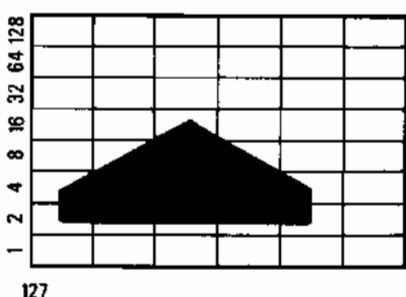
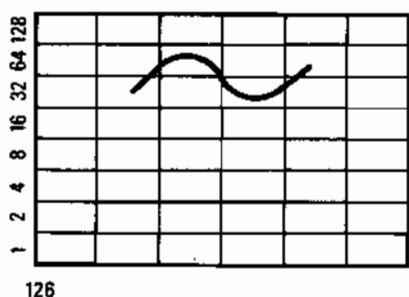
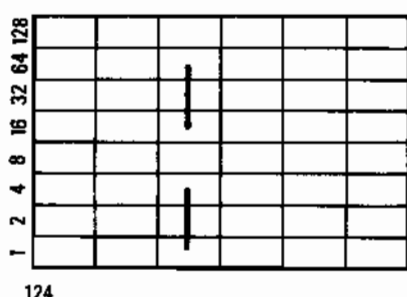
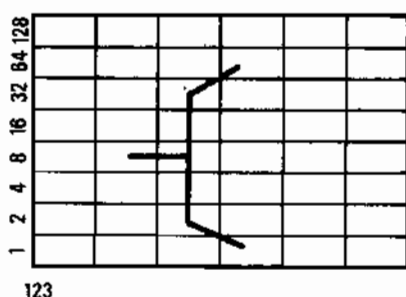
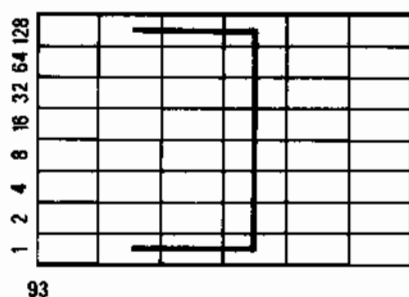
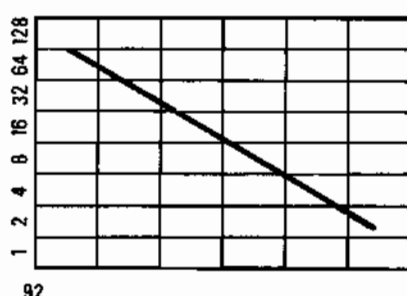
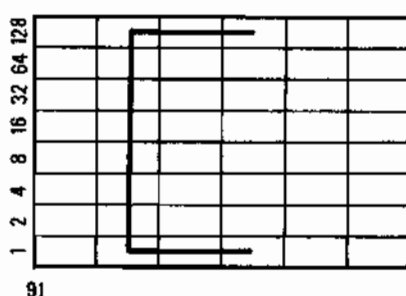
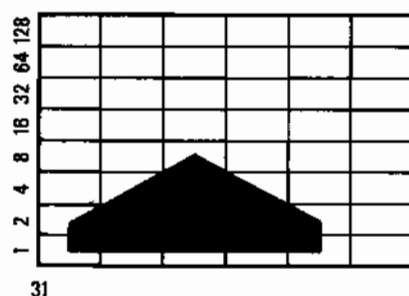
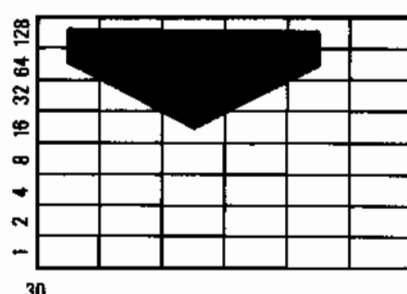
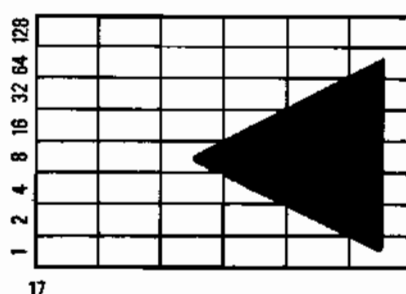
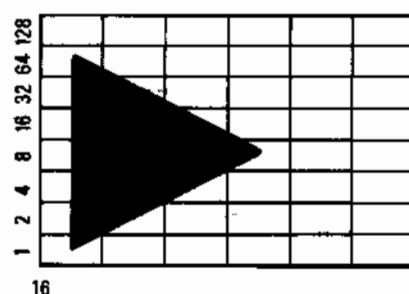
```

Les données de vérification des codes Assembleur à utiliser avec le programme de Checksum décrit en Partie 9, chapitre 8.4 sont les suivantes :

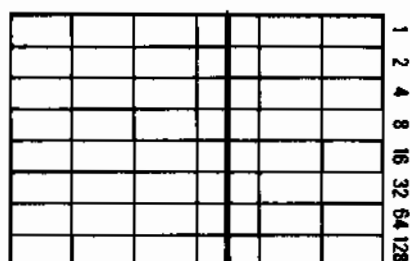
BD 65 DA 70 94 74 38 1F

IV. Utilisation du driver DMP2000 sur d'autres ordinateurs que les Amstrad CPC

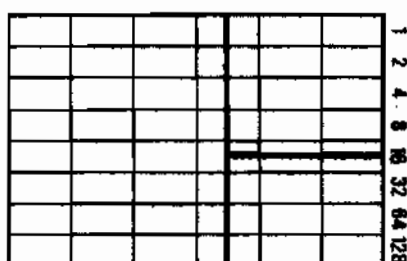
Le driver que nous venons de présenter est utilisable à de petites modifications près sur des ordinateurs autres que l'Amstrad CPC. A titre d'exemple, voici le même driver écrit en Turbo Basic sur un IBM PC. Les données qui figurent en fin de listing correspondent aux caractères graphiques suivants :



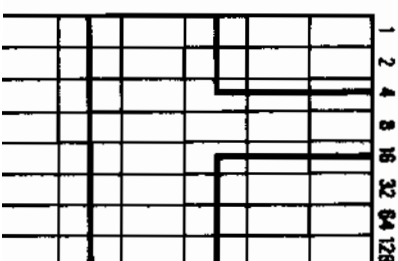
179



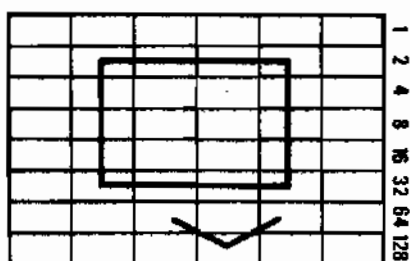
180



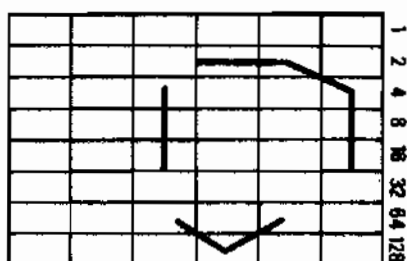
185



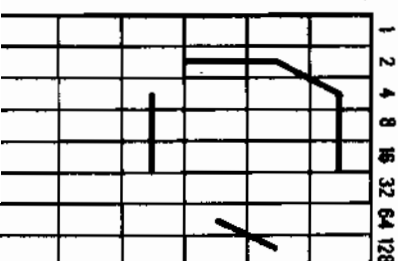
147



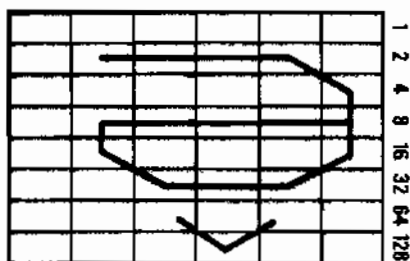
150



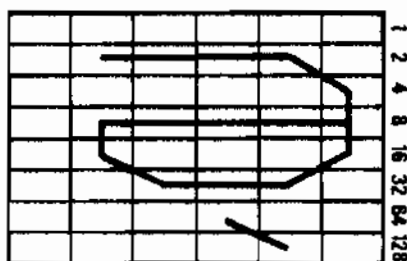
151



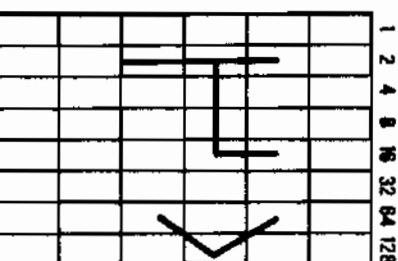
136



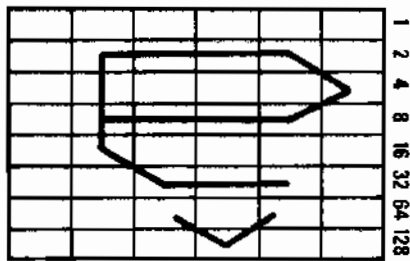
138



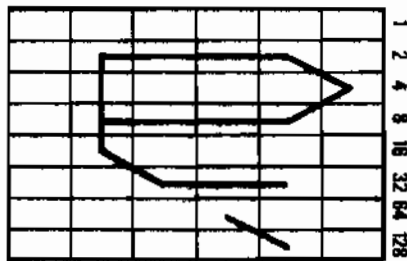
140



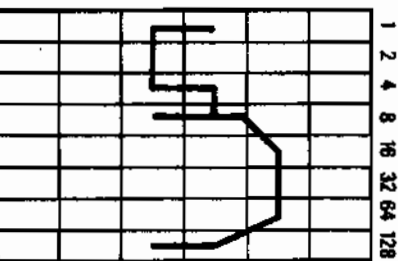
131

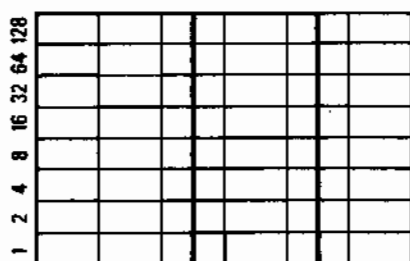


133

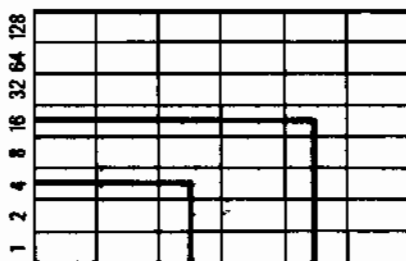


135

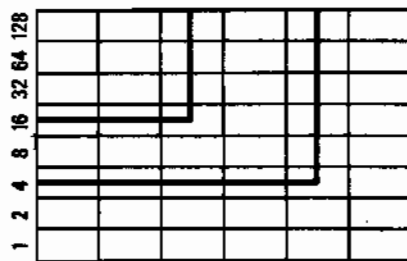




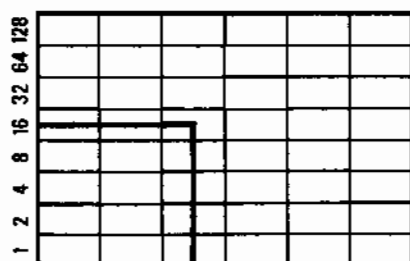
186



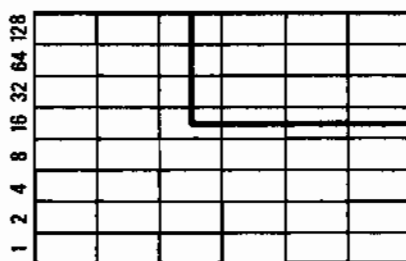
187



188



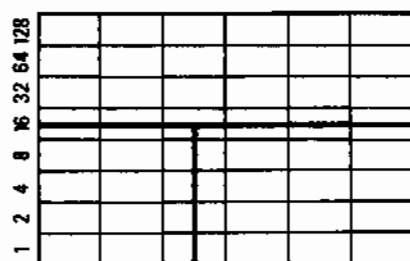
191



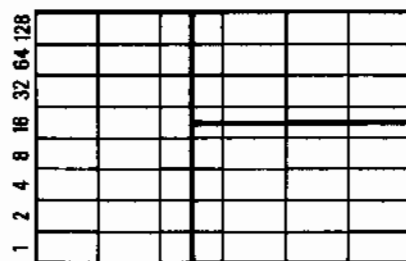
192



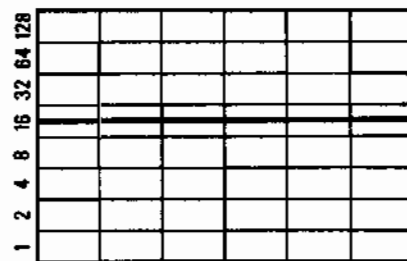
193



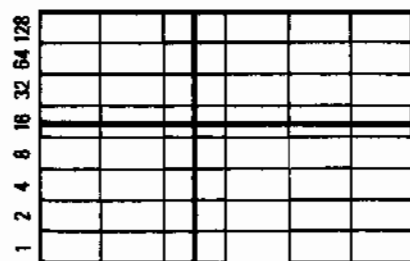
194



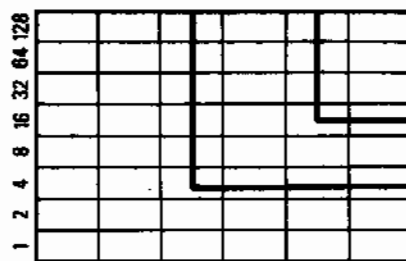
195



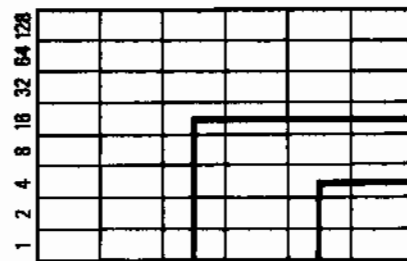
196



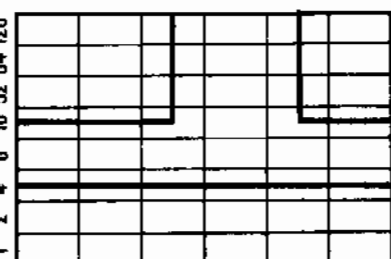
197



200



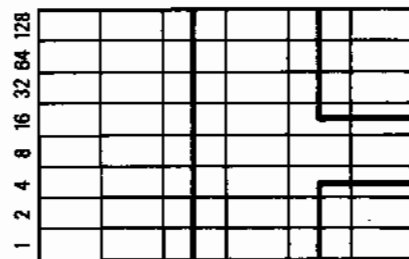
201



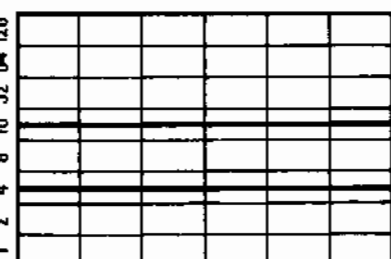
202



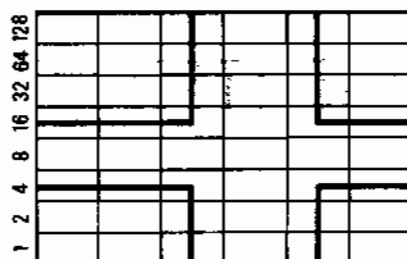
203



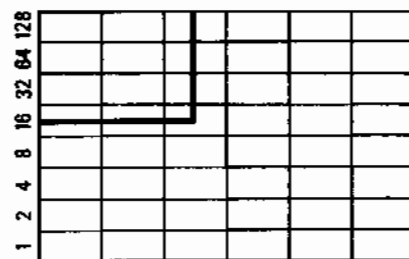
204



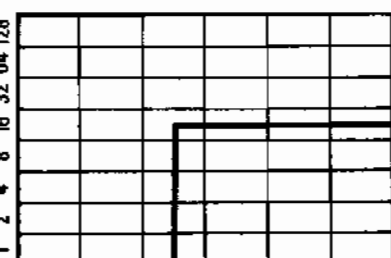
205



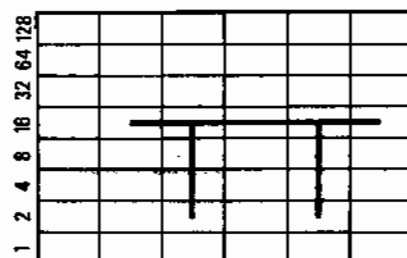
206



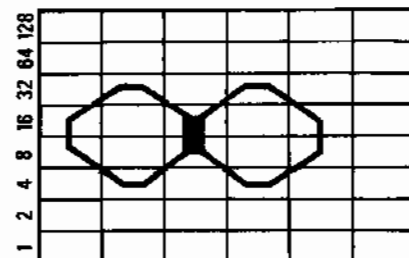
217



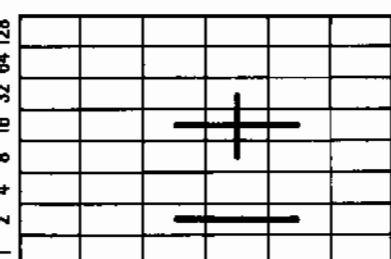
218



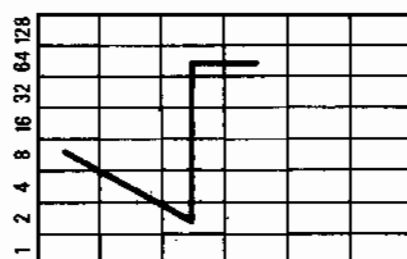
227



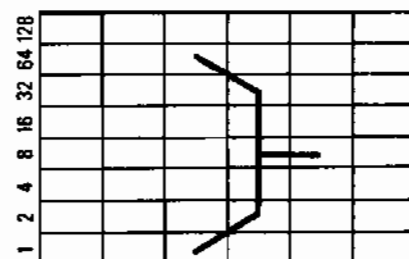
236



241



251



125

Le listing du programme est le suivant :

```
width "lpt1:",2000
DIM TT$(50,12), CC(48)

FOR I=1 TO 48
  READ CC(I)
NEXT I

FOR I=1 TO 50
  FOR J=1 TO 12
    READ TT$(I,J)
  NEXT J
NEXT I

CLS
INPUT "Fichier à imprimer : ";N$
? "Impression de ";N$;" en cours..."
OPEN N$ FOR INPUT AS #1

DO
  LINE INPUT #1,A$
  FOR I%=1 TO LEN(A$)
    b$=MID$(A$,I%,1)
    b=asc(b$)
    car=0
    FOR J%=1 TO 48
      IF CC(J%)=b THEN car=J%
    NEXT J%
    IF car<>0 THEN
      lprint chr$(27);"L";chr$(12);chr$(0);
      for j%=1 to 12
        x$=tt$(car,j%):x=val("&H"+x$):lprint chr$(x);
      next j%
      SELECT CASE b
        CASE 218,180,179,191,194,195,197:
          LPRINT chr$(27);"J";chr$(22);
          LPRINT chr$(8);
          lprint chr$(27);"L";chr$(12);chr$(0);
          for j%=1 to 12
            x$=tt$(49,j%):x=val("&H"+x$):lprint chr$(x);
          next j%
          LPRINT chr$(27);"j";chr$(22);
        CASE 203,204,206,185,187,186,201:
          LPRINT chr$(27);"J";chr$(22);
          LPRINT chr$(8);
          lprint chr$(27);"L";chr$(12);chr$(0);
          for j%=1 to 12
            x$=tt$(50,j%):x=val("&H"+x$):lprint chr$(x);
          next j%
          LPRINT chr$(27);"j";chr$(22);
      END SELECT
    ELSE
      lprint b$;
    END IF
  NEXT I%
  lprint
LOOP until EOF(1)
close
END
```


Codes ASCII des données redéfinies

```
DATA 16,17,30,31,91,92,93,123,124,125,126,127
DATA 130,131,133,135,136,138,140,147,150,151,179,180
DATA 185,186,187,188,191,192,193,194,195,196,197,200
DATA 201,202,203,204,205,206,217,218,227,236,241,251
```

Redéfinition des caractères imprimés

DATA 7F,0,3E,0,1C,0,8,0,0,0,0,0	DATA 0,0,0,0,FF,0,0,0,FF,0,0,0
DATA 0,0,0,0,8,0,1C,0,3E,0,7F,0	DATA 14,0,14,0,17,0,10,0,1F,0,0,0
DATA C0,0,E0,0,F0,0,E0,0,C0,0,0,0	DATA 14,0,14,0,F4,0,4,0,FC,0,0,0
DATA 3,0,7,0,F,0,7,0,3,0,0,0	DATA 10,0,10,0,1F,0,0,0,0,0,0,0
DATA 0,0,FF,0,81,0,81,0,0,0,0,0	DATA 0,0,0,0,F0,0,10,0,10,0,10,0
DATA 0,0,20,0,10,0,8,0,4,0,0,0	DATA 10,0,10,0,F0,0,10,0,10,0,10,0
DATA 0,0,81,0,81,0,FF,0,0,0,0,0	DATA 10,0,10,0,1F,0,10,0,10,0,10,0
DATA 0,0,0,8,0,36,0,41,0,0,0,0	DATA 0,0,0,0,FF,0,10,0,10,0,10,0
DATA 0,0,0,0,77,0,0,0,0,0,0,0	DATA 10,0,10,0,10,0,10,0,10,0,10,0
DATA 0,0,0,41,0,36,0,8,0,0,0,0	DATA 10,0,10,0,FF,0,10,0,10,0,10,0
DATA 0,0,20,0,40,0,20,0,40,0,0,0	DATA 0,0,0,0,FC,0,4,0,F4,0,14,0
DATA 3,0,5,0,9,0,5,0,3,0,0,0	DATA 0,0,0,0,1F,0,10,0,17,0,14,0
DATA 1C,0,2A,0,6A,0,AA,0,1B,0,0,0	DATA 14,0,14,0,F4,0,4,0,F4,0,14,0
DATA 0,4,0,6A,0,AA,0,6A,0,1E,2,0	DATA 14,0,14,0,17,0,10,0,17,0,14,0
DATA 0,4,0,AA,0,6A,0,2A,0,1E,2,0	DATA 0,0,0,0,FF,0,0,0,F7,0,14,0
DATA 0,0,70,0,8D,0,8B,0,0,0,0,0	DATA 14,0,14,0,14,0,14,0,14,0,14,0
DATA 1C,0,6A,0,AA,0,6A,0,1A,0,0,0	DATA 14,0,14,0,F7,0,0,0,F7,0,14,0
DATA 1C,0,AA,0,6A,0,2A,0,1B,0,0,0	DATA 10,0,10,0,F0,0,0,0,0,0,0,0
DATA 0,0,52,0,9E,0,42,0,0,0,0,0	DATA 0,0,0,0,1F,0,10,0,10,0,10,0
DATA 0,1C,62,0,A2,0,62,0,22,1C,0,0	DATA 0,0,10,0,1E,0,10,0,1E,0,10,0
DATA 1C,0,42,0,82,0,40,1C,2,0,0,0	DATA 18,0,24,0,18,0,24,0,18,0,0,0
DATA 3C,0,82,0,42,0,0,3E,2,0,0,0	DATA 0,0,0,0,12,0,3A,0,12,0,0,0
DATA 0,0,0,0,FF,0,0,0,0,0,0,0	DATA 8,0,4,0,FE,0,80,0,0,0,0,0
DATA 8,0,8,0,FF,0,0,0,0,0,0,0	DATA 0,0,0,0,FB,0,0,0,0,0,0,0
DATA 14,0,14,0,F7,0,0,0,FF,0,0,0	DATA 0,0,0,0,FB,0,0,0,FB,0,0,0

Si vous possédez un autre ordinateur qui fonctionne sous Basic, les modifications nécessaires pour adapter le driver devraient être minimales. En utilisant un driver approprié, il est donc possible d'utiliser l'imprimante DMP 2000 sur un micro-ordinateur quelconque muni d'un port Centronics 7 ou 8 bits...

9/8.9

Instruction CAT évoluée

Aux vues des insuffisances de la commande Basic CAT (listage de répertoires), nous avons créé une extension de cette commande permettant :

- le tri par ordre alphabétique, inverse alphabétique, par taille de fichiers et par extension,
- la sortie du catalogue trié sur l'écran, sur une imprimante ou dans un fichier.

Le programme développé ici est écrit en Basic et en Assembleur.

La syntaxe générale d'une commande CAT évoluée est la suivante :

<Specification fichier>, <Type de tri>, <Organe de sortie>

- **<Specification fichier>**

permet de définir le ou les fichiers sur le(s)quel(s) porte(nt) la commande. Cette spécification est du type **<Nom> . <Extension>** avec les restrictions suivantes :

- | | |
|--------------------------|--|
| <Nom> | est un nom de fichier CP/M valide comportant entre 1 et 8 caractères (y compris le joker *). |
| <Extension> | est une extension CP/M valide comportant obligatoirement 3 caractères (même espace). |

- **<Type de tri>** est une des lettres suivantes, indifféremment majuscule ou minuscule :

- | | |
|----------|-------------------------------------|
| A | - - - tri alphabétique, |
| I | - - - tri inverse alphabétique, |
| T | - - - tri par taille de fichier, |
| E | - - - tri par extension de fichier. |

- **<Organe de sortie>** est une des lettres suivantes, indifféremment majuscule ou minuscule :

- | | |
|----------|---------------------------------|
| E | - - - sortie écran, |
| I | - - - sortie imprimante, |
| F | - - - écriture dans un fichier. |

Lorsque l'option F apparaît dans la commande, elle doit être suivie d'un nom de fichier CP/M valide (1 à 8 caractères pour le radical et 0 à 3 caractères pour l'extension).

Exemples :

Visualisation de tous les fichiers par ordre alphabétique : *.* ,A,E

```
CAT *.* ,A,E
3DEX          BIN          1 KOctet(s).
ADJ           BAS          1 KOctet(s).
ADJ1          BIN          17 KOctet(s).
ADJANI2       BIN          1 KOctet(s).
AF2D          2 KOctet(s).
AF3D          2 KOctet(s).
AS2           2 KOctet(s).
ASM           2 KOctet(s).
BIN2          BIN          17 KOctet(s).
COMPAC2       BAS          4 KOctet(s).
DAGRAPH       BIN          17 KOctet(s).
DEBUG         BAS          5 KOctet(s).
DECOM2        2 KOctet(s).
DEFIN         1 KOctet(s).
ESSS          BIN          1 KOctet(s).
F1            BIN          2 KOctet(s).
GEFOR         BAS          3 KOctet(s).
INIT          BAS          1 KOctet(s).
MAXSUI        4 KOctet(s).
MUSE          BIN          1 KOctet(s).
P2            BIN          5 KOctet(s).
PRES          BIN          17 KOctet(s).
REG           4 KOctet(s).
STAT          DOC          1 KOctet(s).
SUPERCOM      BAS          4 KOctet(s).
T2            BIN          6 KOctet(s).
TAF2D         1 KOctet(s).
X             8 KOctet(s).
X2            3 KOctet(s).
ZARKA         BAS          3 KOctet(s).
```

Impression des fichiers dont le préfixe commence par AD, par extension de fichier : AD*.* ,E,I

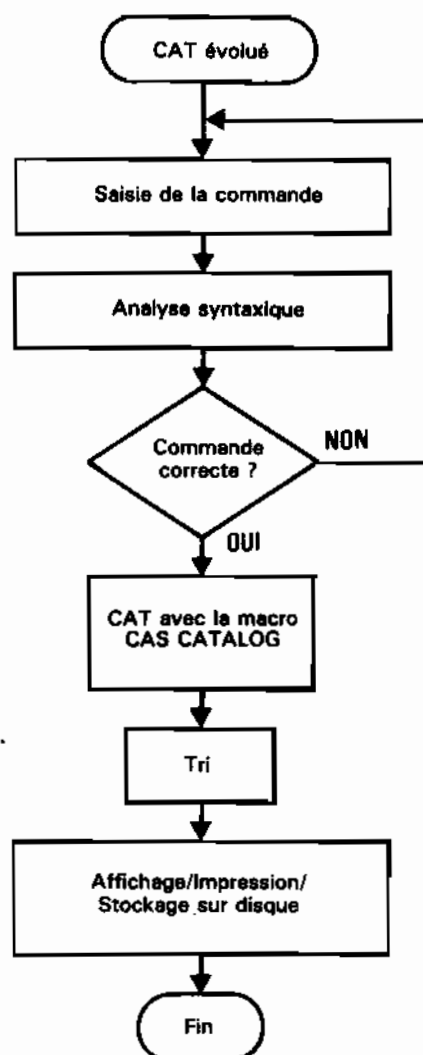
```
CAT AD*.* ,E,I
ADJ           BAS          1 KOctet(s).
ADJ1          BIN          17 KOctet(s).
ADJANI2       BIN          1 KOctet(s).
```

Stockage sur disque des fichiers d'extension nulle, par taille de fichier,
dans le fichier stat.doc : *. ,T,F,stat.doc

CAT *. ,T,F,STAT.DOC	
DEFIN	1 KOctet(s).
TAF2D	1 KOctet(s).
AF2D	2 KOctet(s).
AF3D	2 KOctet(s).
AS2	2 KOctet(s).
ASM	2 KOctet(s).
DECOM2	2 KOctet(s).
X2	3 KOctet(s).
MAXSUI	4 KOctet(s).
REG	4 KOctet(s).
X	8 KOctet(s).

Analyse du programme

La logique du programme respecte l'ordinogramme suivant :



L'analyse syntaxique s'assure que la commande entrée pourra être traitée par le logiciel :

- spécification de fichier correcte,
- type de tri correct,
- organe de sortie correct,
- spécification de fichier correcte pour l'organe de sortie F.

Si un des points listé ci-dessus n'est pas conforme au format attendu, un message d'erreur est affiché, la commande entrée est ignorée, et le programme réexécuté pour saisir une nouvelle commande.



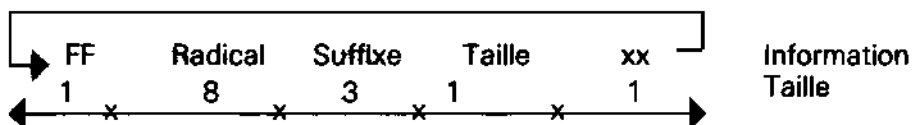
Le programme Basic utilise une routine du Firmware : **CAS CATALOG**. Cette routine permet de lire le catalogue d'une disquette.

```

1                ORG 9000H
2                LOAD 9000H
3                ; -----
4                CASCAT: EQU 0BC9BH                ; CAS CATALOG
5                ; -----
6 9000 110080                LD DE,8000H                ; à Buffer
7 9003 CD9BBC                CALL CASCAT                ; Catalogue
8 9006 C9                RET
9                END

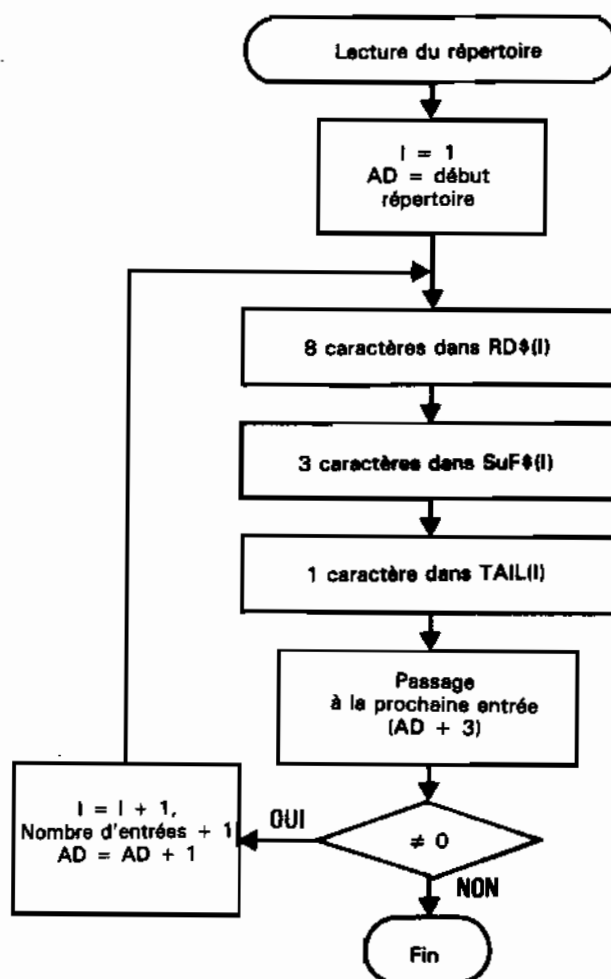
```

Ce catalogue est affiché sur l'écran et stocké en mémoire RAM à une adresse quelconque selon le format suivant :



Le délimiteur &HFF est utilisé entre deux entrées de fichiers et le délimiteur &H00 marque la fin de la dernière entrée dans le répertoire.

La lecture des nom, suffixe et taille des fichiers d'un répertoire suit les étapes de l'ordinogramme suivant :



Le listing du programme est le suivant :

```

1000 '=====
1010 'Instruction CAT evoluee
1020 '=====
1030 '-----
1040 ' Initialisations
1050 '-----
1060 DIM RD$(30),SUF$(30),TAIL(30),TP(30)
1070 DIM TT1$(30),TT2$(30),TT3(30)
1080 '
1090 FOR I=&9000 TO &9006
1100   READ A$
1110   A=VAL("&H"+A$)
1120   POKE I,A

```



```

1680 INK 0,0:PEN 0
1690 CALL &9000 'Appel du S/P Assembleur
1700 CLS:PEN 1
1710 PRINT"Tri en cours...":PRINT
1720 '
1730 '-----
1740 'Classement des entrees
1750 '-----
1760 AD=&8000
1770 AD=AD+1
1780 I=1:NB=1 'Initialisation
1790 FOR J=1 TO 8
1800   RD$(I)=RD$(I)+CHR$(PEEK(AD))
1810   AD=AD+1
1820 NEXT J
1830 FOR J=1 TO 3
1840   SUF$(I)=SUF$(I)+CHR$(PEEK(AD))
1850   AD=AD+1
1860 NEXT J
1870 TAIL(I)=PEEK(AD)
1880 AD=AD+2 'Passage a l'entree suivante
1890 IF PEEK(AD)<>0 THEN I=I+1:AD=AD+1:NB=NB+1:GOTO 1790
1900 '
1910 '-----
1920 'Toutes les entrees sont memorisees
1930 '-----
1940 IF C3$="A" THEN 2580
1950 '
1960 '-----
1970 'Classement par ordre inverse alphabetique
1980 '-----
1990 IF C3$<>"I" THEN 2150
2000 FOR I=1 TO NB
2010   TT1$(I)=RD$(NB-I+1)
2020   TT2$(I)=SUF$(NB-I+1)
2030   TT3(I)=TAIL(NB-I+1)
2040 NEXT I
2050 FOR I=1 TO NB
2060   RD$(I)=TT1$(I)
2070   SUF$(I)=TT2$(I)
2080   TAIL(I)=TT3(I)
2090 NEXT I
2100 GOTO 2580
2110 '
2120 '-----
2130 'Classement par taille
2140 '-----
2150 IF C3$<>"T" THEN 2390
2160 FOR I=1 TO NB
2170   TP(I)=I
2180 NEXT I
2190 V=0
2200 FOR I=1 TO NB-1
2210   IF TAIL(I)<=TAIL(I+1) THEN 2240
2220   A=TAIL(I+1):TAIL(I+1)=TAIL(I):TAIL(I)=A:V=1
2230   A=TP(I+1):TP(I+1)=TP(I):TP(I)=A

```

```

2240 NEXT I
2250 IF V=1 THEN 2190
2260 FOR I=1 TO NB
2270   TT1$(I)=RD$(TP(I))
2280   TT2$(I)=SUF$(TP(I))
2290 NEXT I
2300 FOR I=1 TO NB
2310   RD$(I)=TT1$(I)
2320   SUF$(I)=TT2$(I)
2330 NEXT I
2340 GOTO 2580
2350 '
2360 '-----
2370 'Classement par extension
2380 '-----
2390 FOR I=1 TO NB
2400   TP(I)=I
2410 NEXT I
2420 V=0
2430 FOR I=1 TO NB-1
2440   IF SUF$(I)<=SUF$(I+1) THEN 2470
2450   A$=SUF$(I+1):SUF$(I+1)=SUF$(I):SUF$(I)=A$:V=1
2460   A=TP(I+1):TP(I+1)=TP(I):TP(I)=A
2470 NEXT I
2480 IF V=1 THEN 2420
2490 FOR I=1 TO NB
2500   TT1$(I)=RD$(TP(I))
2510   TT3(I)=TAIL(TP(I))
2520 NEXT I
2530 FOR I=1 TO NB
2540   RD$(I)=TT1$(I)
2550   TAIL(I)=TT3(I)
2560 NEXT I
2570 '
2580 '-----
2590 'Toutes les entrees sont classees
2600 '-----
2610 CLS
2620 IF C4$<>"F" THEN 2650
2630 A$=C5$+"."+C6$:OPENOUT A$
2640 A$="CAT "+C$:PRINT#9,A$
2650 IF C4$="E" THEN PRINT"CAT "+C$
2660 IF C4$="I" THEN PRINT#8,"CAT "+C$
2670 FOR I=1 TO NB
2680   OK=0
2690   IF INSTR(C1$,"*")<>0 AND LEFT$(C1$,1)<>"*" THEN 2720
2700   IF INSTR(C1$,"*")<>0 AND LEFT$(C1$,1)="*" THEN OK=1:GOTO 2750
2710   IF C1$=RD$(I) THEN OK=1:GOTO 2750 ELSE OK=0:GOTO 2820
2720   P=INSTR(C1$,"*")
2730   IF LEFT$(C1$,P-1)=LEFT$(RD$(I),P-1) THEN OK=1
2740   '
2750   IF OK<>1 THEN 2860
2760   IF INSTR(C2$,"*")<>0 AND LEFT$(C2$,1)<>"*" THEN 2790
2770   IF INSTR(C2$,"*")<>0 AND LEFT$(C2$,1)="*" THEN OK=2:GOTO 2820
2780   IF C2$=SUF$(I) THEN OK=2:GOTO 2820 ELSE OK=0:GOTO 2820
2790   P=INSTR(C2$,"*")

```

```

2800 IF LEFT$(C2$,P-1)=LEFT$(SUF$(I),P-1) THEN OK=2
2810 '
2820 IF OK<>2 THEN 2860
2830 IF C4$="E" THEN PRINT RD$(I),SUF$(I),TAIL(I); "KOctet(s). "
2840 IF C4$="I" THEN PRINT#8,RD$(I),SUF$(I),TAIL(I); "KOctet(s). "
2850 IF C4$="F" THEN PRINT#9,RD$(I),SUF$(I),TAIL(I); "KOctet(s). "
2860 NEXT I
2870 IF C4$="F" THEN CLOSEOUT
2880 END
2890 '-----
2900 'Erreur pendant l'analyse syntaxique
2910 '-----
2920 IF E=1 THEN PRINT"Point decimal attendu"
2930 IF E=2 THEN PRINT"Longueur prefixe incorrecte"
2940 IF E=3 THEN PRINT"Virgule attendue"
2950 IF E=4 THEN PRINT"Longueur suffixe incorrecte"
2960 IF E=5 THEN PRINT"A,I,T ou E attendu"
2970 IF E=6 THEN PRINT"Virgule attendue"
2980 IF E=7 THEN PRINT"E, I ou F attendu"
2990 IF E=8 THEN PRINT"Virgule attendue"
3000 IF E=9 THEN PRINT"Point decimal attendu"
3010 IF E=10 THEN PRINT"Longueur prefixe incorrecte"
3020 IF E=11 THEN PRINT"Longueur suffixe incorrecte"
3030 PRINT:PRINT"Appuyez sur une touche..."
3040 A$=INKEY$:IF A$="" THEN 3040
3050 RUN

```

- Lignes 1060 et 1070... Déclaration des tableaux de travail
- Lignes 1090 à 1150... Mise en mémoire du programme
Assembleur
- Lignes 1200 à 1250... Saisie de la chaîne de commande CAT
- Lignes 1300 à 1590... Analyse syntaxique
Extraction :
 - préfixe (ligne 1330),
 - suffixe (ligne 1380),
 - type de tri (ligne 1440),
 - organe de sortie (ligne 1480),
 - préfixe option T (ligne 1560),
 - suffixe option T (ligne 1590),
- Lignes 1680 à 1710... Lecture du répertoire en utilisant la macro
CAS CATALOG du firmware
- Lignes 1760 à 1890... Mémorisation dans les tableaux des don-
nées du répertoire (noms, suffixes et tailles)
- Ligne 1940... Aucune opération si tri alphabétique
- Lignes 1990 à 2100... Tri par ordre inverse alphabétique
- Lignes 2150 à 2340... Tri par taille de fichier
- Lignes 2390 à 2560... Tri par extension
- Lignes 2610 à 2880... Affichage, impression ou stockage sur dis-
que du résultat du tri
- Lignes 2920 à 3050... Erreur pendant l'analyse syntaxique

9/8.10

Edition et modification des secteurs d'une disquette

Suite aux programmes de lecture et écriture de secteurs de disquettes, nous avons développé un éditeur de secteurs élémentaire.

Grâce à lui, vous pouvez :

- visualiser un des secteurs de la disquette (zone de 512 octets consécutifs). Rappelons que chaque disquette est divisée en neuf secteurs numérotés de 1 à 9, et en 40 pistes numérotées de 0 à 39,
- modifier un octet particulier d'un des secteurs de la disquette.

Ce programme est écrit en Basic et en Assembleur.

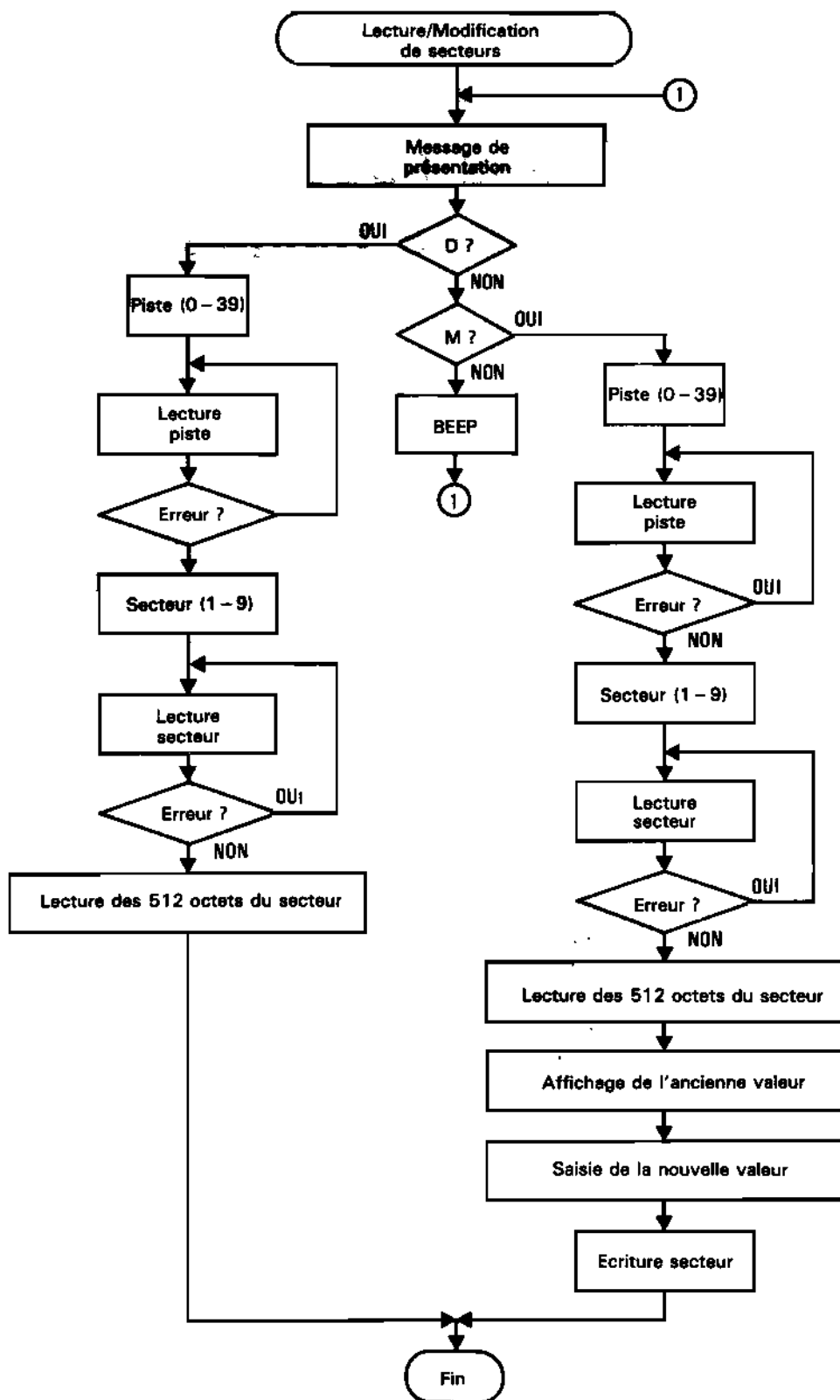
La partie Basic réalise :

- l'affichage des secteurs sur deux pages,
- l'attente entre l'affichage des deux pages.

La partie Assembleur réalise :

- l'acquisition du choix utilisateur et des données correspondantes :
- pour une lecture de secteur : numéro de piste et de secteur,
- pour une modification d'octet, numéro de piste et de secteur, déplacement de l'octet à modifier par rapport au début du secteur et acquisition de la nouvelle valeur de cet octet,
- la lecture du secteur demandé,
- éventuellement l'écriture du secteur qui a été modifié.

La logique du sous-programme Assembleur est la suivante :



Remarques :

Les sous-programmes utilisés sont :

- BEEP pour émettre un bip sonore lorsque l'utilisation a effectué une action interdite,
- AFALPH pour afficher un texte alphanumérique sur l'écran. Le texte à afficher est suivi du terminateur OFFH,
- SAISIE pour saisir un ou plusieurs caractères au clavier. Le code ASCII de ces caractères est stocké dans un buffer paramétrable d'adresse (PBUF),
- ECR pour écrire un secteur de disquette,
- LECT pour lire un secteur de disquette.

Le programme Assembleur est le suivant :

```

                ORG 9000H
                LOAD 9000H

;-----
;Lecture/Modification de secteurs
;sur une disquette
;-----
;
;-----
;Definition des constantes
;-----
;
SOUND:         EQU 0BD34H           ;MC SOUND REG
FIND:          EQU 0BCD4H           ;KL FIND COMMAND
PRINT:         EQU 0BB5AH           ;TXT OUTPUT
READ:          EQU 0BB06H           ;KM WAIT CHAR
MODE:          EQU 0BC0EH           ;SCR SET MODE
CR:            EQU 13               ;Carriage Return
LF:            EQU 10               ;Line Feed
DEL:           EQU 127              ;127 ;DELete
BS:            EQU 8                ;Back Space
CURS:          EQU 95               ;Curseur
BLANC:         EQU 32               ;Espace
;
;-----
;Definition des variables
;-----
BD:            DS 3                 ;Buffer de donnees
AD:            DS 3                 ;Sauv sortie de FIND
MAX:           DS 1                 ;Nbre de caract a lire
PBUF:          DS 2                 ;Pointeur de buffer

```

```

31 9009 85      ECRIT:      DB      85H          ; Instr. Ecriture
32 900A 84      LIT:       DB      84H          ; Instr. Lecture
33              DRIVE:     DS      1            ; Numero du drive
34              PISTE:      DS      1            ; Numero de piste
35              SECT:       DS      1            ; Numero de secteur
36              BUFF:       DS      512         ; Buffer lect/ecrit
37              CHOIX:      DS      1            ; Choix dans menu
38              OFFSET:     DS      2            ; @ octet a modifier
39              CAM:        DS      1            ; Caractere a modifier
40              ;
41              ;-----
42              ;Definition des messages
43              ;-----
44              MES1:       EQU      $
45 9212 54617065          DB      "Tapez D pour visu
45 9216 7A204420
45 921A 706F7572
45 921E 20766973
45 9222 75616C69
45 9226 736572
46 9229 20756E20          DB      " un secteur,"
46 922D 73656374
46 9231 6575722C
47 9235 0D0A             DB      CR,LF
48 9237 6574204D          DB      "et M pour modifie
48 923B 20706F75
48 923F 72206D6F
48 9243 64696669
48 9247 65722075
48 924B 6E20

```



```

9 924D 6F637465          DB  "octet particulier
9 9251 74207061
9 9255 72746963
9 9259 756C6965
9 925D 722E
0 925F 0D0A0AFF          DB  CR,LF,LF,OFFH
1
;-----
2          MES2:          EQU  $
3 9263 50697374          DB  "Piste (0-39) : "
3 9267 65202830
3 926B 2D333929
3 926F 203A20
4 9272 FF                DB  OFFH
5
;-----
6          MES3:          EQU  $
7 9273 53656374          DB  "Secteur (1-9) : "
7 9277 65757220
7 927B 28312D39
7 927F 29203A20
8 9283 FF                DB  OFFH
9
;-----
0          MES4:          EQU  $
1 9284 4F637465          DB  "Octet (0-511) : "
1 9288 74202830
1 928C 2D353131
1 9290 29203A20
2 9294 FF                DB  OFFH
3
;-----
4          MES5:          EQU  $
5 9295 416E6369          DB  "Ancienne valeur :

```

```

65 9299 656E6E65
65 929D 2076616C
65 92A1 65757220
65 92A5 3A2026
66 92A8 FF                      DB    OFFH
67                                ;-----
68                                MES6:    EQU    $
69 92A9 4E6F7576                DB    "Nouvelle valeur :
69 92AD 656C6C65
69 92B1 2076616C
69 92B5 65757220
69 92B9 3A202620
70 92BD FF                      DB    OFFH
71                                ;-----
72                                ALALI:    EQU    $
73 92BE 0D0AFF                  DB    CR,LF,OFFH
74                                ;-----
75                                ;
76                                ;-----
77                                ;ZONE DES SOUS-PROGRAMMES
78                                ;-----
79                                ;
80                                ;-----
81                                ;Emission d'un BEEP sonore
82                                ;-----
83                                ;Entree: aucune
84                                ;Sortie: AF, BC et HL effaces
85                                ;-----
86                                ;
87                                BEEP:      EQU    $                      ;Point d'entree

```

```

88 92C1 3E00          LD  A,0
89 92C3 0E00          LD  C,0
90 92C5 CD34BD        CALL SOUND
91 92C8 3E01          LD  A,1
92 92CA 0E01          LD  C,1
93 92CC CD34BD        CALL SOUND          ;Frequence
94 92CF 3E08          LD  A,8
95 92D1 0E06          LD  C,6
96 92D3 CD34BD        CALL SOUND          ;Amplitude
97 92D6 3E07          LD  A,7
98 92D8 0E08          LD  C,8
99 92DA CD34BD        CALL SOUND          ;Validation registre A
00                   ;
01 92DD 21FFFF        LD  HL,OFFFHH
02                   BOU: EQU  $
03 92E0 2B            DEC  HL
04 92E1 7C            LD   A,H
05 92E2 B5            OR   L
06 92E3 20FB          JR   NZ,BOU
07                   ;
08 92E5 3E08          LD  A,8
09 92E7 0E00          LD  C,0
10 92E9 CD34BD        CALL SOUND          ;Amplitude 0
11 92EC C9            RET
12                   ;
13                   ;-----
14                   ;Affichage d'un texte alphanum.
15                   ;-----
16                   ;Entree: @ de depart dans HL
17                   ;Sortie: aucun registre modifie

```

```

118      ;-----
119      ;
120      AFALPH:    EQU  $                ;point d'entree
121  92ED E5                PUSH HL
122  92EE F5                PUSH AF
123      ME1:      EQU  $                ;Boucle d'affichage
124  92EF 7E                LD   A,(HL)
125  92F0 FEFF             CP   OFFH
126  92F2 2B06             JR   Z,ME2                ;Fin d'affichage
127  92F4 CD5ABB           CALL PRINT                ;Af. caractere
128  92F7 23                INC  HL                ;Caractere suivant
129  92F8 1BF5             JR   ME1
130      ME2:      EQU  $
131  92FA F1                POP  AF
132  92FB E1                POP  HL
133  92FC C9                RET
134      ;
135      ;-----
136      ;Saisie de caracteres
137      ;-----
138      ;Entree: (MAX)=Nbre de caracteres
139      ;Sortie: Aucun registre ecrase
140      ;-----
141      ;
142      SAISIE:    EQU  $                ;Point d'entree
143  92FD 3A0690           LD   A,(MAX)
144  9300 57                LD   D,A                ;Nbre de caract. a lire
145  9301 010000           LD   BC,0                ;Index dans le buffer
146      S1:      EQU  $
147  9304 2A0790           LD   HL,(PBUF)            ;Buffer de lecture

```

Partie 9 : Programmes

```

48 9307 CD06BB      CALL READ          ;Lecture 1 caractere
49 930A FE0D        CP    CR            ;Carriage Return ?
50 930C 2B3C        JR    Z,S3          ;Oui=> Fin de saisie
51 930E FE7F        CP    DEL           ;DELeTe ?
52 9310 2B1B        JR    Z,S2          ;Oui => Traitement DEL
53 9312 F5          PUSH AF
54 9313 3E06        LD    A,BS
55 9315 CD5ABB      CALL PRINT
56 931B F1          POP  AF
57 9319 CD5ABB      CALL PRINT
58 931C 09          ADD  HL,BC
59 931D 77          LD    (HL),A        ;Sauvegarde
60 931E 03          INC  BC
61 931F 3E5F        LD    A,CURS
62 9321 CD5ABB      CALL PRINT
63 9324 79          LD    A,C
64 9325 BA          CP    D
65 9326 20DC        JR    NZ,S1         ;Suite de la saisie
66 932B 1B20        JR    S3           ;Fin de saisie
67                S2: EQU  $
68 932A 79          LD    A,C
69 932B B7          OR    A
70 932C 2BDA        JR    Z,S1         ;DEL non accepte
71 932E 0B          DEC  BC
72 932F 3E0B        LD    A,BS
73 9331 CD5ABB      CALL PRINT          ;Retour en arriere
74 9334 3E20        LD    A,BLANK
75 9336 CD5ABB      CALL PRINT          ;Effacement caractere
76 9339 3E0B        LD    A,BS
77 933B CD5ABB      CALL PRINT          ;Retour en arriere (6)

```

```

178 933E 3E0B          LD  A,BS
179 9340 CD5ABB        CALL PRINT          ;Retour en arriere
180 9343 3E5F          LD  A,CURS
181 9345 CD5ABB        CALL PRINT          ;Affichage curseur
182 9348 18BA          JR   S1
183                   S3: EQU  $
184 934A 3E0B          LD  A,BS
185 934C CD5ABB        CALL PRINT          ;Retour en arriere
186 934F 3E20          LD  A,BLANC
187 9351 CD5ABB        CALL PRINT          ;Effacement caract.
188 9354 C9           RET
189                   ;
190                   ;-----
191                   ; Ecriture d'un secteur
192                   ;-----
193                   ;Entree:
194                   ;Sortie:
195                   ;-----
196                   ;
197                   ECR: EQU  $          ;Point d'entree
198 9355 210990         LD  HL,ECRIT      ;Ecriture disque
199 9358 CDD4BC        CALL FIND          ;KL FIND COMMAND
200 935B 220390         LD  (AD),HL
201 935E 79            LD  A,C
202 935F 320590         LD  (AD+2),A
203 9362 3A0B90         LD  A,(DRIVE)
204 9365 5F            LD  E,A          ;No de drive
205 9366 3A0C90         LD  A,(PISTE)
206 9369 57            LD  D,A          ;No de piste
207 936A 3A0D90         LD  A,(SECT)

```

```

08 936D 4F          LD    C,A          ;No de secteur
09 936E 210E90      LD    HL,BUFF
10 9371 DF          RST    24          ;Activ. instruction en R0
11 9372 0390        DW    AD
12 9374 C9          RET
13                ;
14                ;-----
15                ; Lecture d'un secteur
16                ;-----
17                ;Entree:
18                ;Sortie:
19                ;-----
20                ;
21                LECT:    EQU    $          ;Point d'entree
22 9375 210A90      LD    HL,LIT          ;Lecture disque
23 9378 CDD4BC      CALL FIND          ;KL FIND COMMAND
24 937B 220390      LD    (AD),HL
25 937E 79          LD    A,C
26 937F 320590      LD    (AD+2),A
27 9382 3A0B90      LD    A,(DRIVE)
28 9385 5F          LD    E,A          ;No de drive
29 9386 3A0C90      LD    A,(PISTE)
30 9389 57          LD    D,A          ;No de piste
31 938A 3A0D90      LD    A,(SECT)
32 938D 4F          LD    C,A          ;No de secteur
33 938E 210E90      LD    HL,BUFF
34 9391 DF          RST    24          ;Activ. instruction en R0
35 9392 0390        DW    AD
36 9394 C9          RET
37                ;

```

```

238      ; =====
239      ; PROGRAMME PRINCIPAL
240      ; =====
241      ORG 8000H
242      LOAD 8000H
243 8000 3E02      LD A,2
244 8002 CD0EBC    CALL MODE      ;Mode 2
245 8005 211292    LD HL,MES1
246 8008 CDED92    CALL AFALPH     ;Aff 1er message
247      ;
248      BIS:      EQU $
249 800B CD06BB    CALL READ       ;Lecture choix
250 800E F620      OR 20H          ;Mise en minuscule
251 8010 320E92    LD (CHOIX),A    ;Memorisation
252 8013 FE64      CP 64H          ;D ?
253 8015 2B09      JR Z,DUMP       ;Affichage 1 secteur
254 8017 FE6D      CP 6DH          ;M ?
255 8019 2B05      JR Z,DUMP       ;Modif 1 octet
256 801B CDC192    CALL BEEP       ;Reponse incorrecte
257 801E 18EB      JR BIS         ;Resaisie
258      ;
259      DUMP:     EQU $
260 8020 21BE92    LD HL,ALALI
261 8023 CDED92    CALL AFALPH     ;Passage a la ligne
262 8026 216392    LD HL,MES2
263 8029 CDED92    CALL AFALPH     ;Aff 2eme message
264 802C 210090    LD HL,BD
265 802F 3EFF      LD A,OFFH
266 8031 77        LD (HL),A
267 8032 23        INC HL

```



```

58 8033 77          LD  (HL),A          ;Init zone de saisie
59 8034 3E02        LD  A,2
70 8036 320690       LD  (MAX),A        ;Nbre de caract a lire
71 8039 210090       LD  HL,BD
72 803C 220790       LD  (PBUF),HL      ;Zone de lecture
73 803F CDFD92       CALL SAISIE        ;Saisie Piste
74 8042 3A0190       LD  A,(BD+1)
75 8045 FEFF        CP  OFFH
76 8047 2B20        JR  Z,UNCAR
77 8049 3A0090       LD  A,(BD)
78 804C E60F        AND  0FH           ;Extraction chiffre
79 804E 47          LD  B,A
80 804F 3E0A        LD  A,10
81 8051 4F          LD  C,A
82 8052 AF          XOR  A
83                PPB1: EQU $
84 8053 81          ADD  A,C
85 8054 10FD        DJNZ PPB1
86 8056 47          LD  B,A
87 8057 3A0190       LD  A,(BD+1)
88 805A E60F        AND  0FH           ;Extraction chiffre
89 805C 80          ADD  A,B           ;A = Nombre tape
90 805D 320C90       LD  (PISTE),A      ;Memorisation
91 8060 FE2B        CP  40
92 8062 3B0D        JR  C,OK1          ;Piste OK
93 8064 CDC192       CALL BEEP
94 8067 1B87        JR  DUMP
95                ;
96                UNCAR: EQU $
97 8069 3A0090       LD  A,(BD)

```

```

298 806C E60F          AND    0FH          ;Extraction chiffre
299 806E 320C90        LD      (PISTE),A    ;Memorisation
300                    OK1:    EQU    $
301 8071 21BE92        LD      HL,ALALI
302 8074 CD0ED92       CALL AFALPH          ;A la ligne
303 8077 217392        LD      HL,MES3
304 807A CD0ED92       CALL AFALPH          ;Aff message 3
305 807D CD06BB        CALL READ           ;Lecture 1 caract
306 8080 CD5ABB        CALL PRINT          ;Affichage caractere
307 8083 E60F          AND    0FH          ;Extraction chiffre
308 8085 C640          ADD    A,64          ;Ajustement secteur
309 8087 320D90        LD      (SECT),A     ;Memo isation
310 808A FE4A          CP      74
311 808C 3805          JR      C,OK2        ;Secteur OK
312 808E CDC192       CALL BEEP
313 8091 18DE          JR      OK1          ;Resaisie du secteur
314                    OK2:    EQU    $
315 8093 3A0E92        LD      A,(CHOIX)
316 8096 FE64          CP      64H          ; D?
317 8098 2802          JR      Z,CY3        ;Poursuite du DUMP
318 809A 1808          JR      MODIF        ;Modification d'un octet
319                    OK3:    EQU    $
320 809C AF            XOR     A
321 809D 320B90        LD      (DRIVE),A    ;Drive A
322 80A0 CD7593       CALL LECT             ;Lecture secteur
323 80A3 C9           RET
324                    ;
325                    ;
326                    MODIF:    EQU    $    ;Modification 1 octet
327 80A4 21BE92        LD      HL,ALALI

```

28 80A7 CDED92	CALL AFALPH	;A la ligne
29 80AA 218492	LD HL,MES4	
30 80AD CDED92	CALL AFALPH	;Aff 4eme message
31 80B0 210090	LD HL,BD	
32 80B3 3EFF	LD A,OFFH	
33 80B5 77	LD (HL),A	
34 80B6 23	INC HL	
35 80B7 77	LD (HL),A	
36 80B8 23	INC HL	
37 80B9 77	LD (HL),A	;Init zone de saisie
38 80BA 3E03	LD A,3	
39 80BC 320690	LD (MAX),A	;Nb de caract a lire
40 80BF 210090	LD HL,BD	
41 80C2 220790	LD (PBUF),HL	;Zone de lecture
42 80C5 CDFD92	CALL SAISIE	;Lecture octet a modif
43 80C8 3A0290	LD A,(B0+2)	
44 80CB FEFF	CP OFFH	
45 80CD 2837	JR Z,UNZER	;Nb sur 2 chiffres
46 80CF 3A0090	LD A,(BD)	
47 80D2 E60F	AND 0FH	
48 80D4 47	LD B,A	
49 80D5 210000	LD HL,0	
50 80D8 116400	LD DE,100	
51	BM1: EQU \$	
52 80DB 19	ADD HL,DE	
53 80DC 10FD	DJNZ BM1	
54 80DE 3A0190	LD A,(BD+1)	
55 80E1 FEFF	CP OFFH	
56 80E3 280C	JR Z,BM2	
57 80E5 E60F	AND 0FH	

```

358 80E7 B7          OR    A
359 80E8 2807        JR     Z,BM2
360 80EA 47          LD     B,A
361 80EB 110A00       LD     DE,10
362                BM10: EQU  $
363 80EE 19          ADD    HL,DE
364 80EF 10FD        DJNZ   BM10
365                BM2:  EQU  $
366 80F1 3A0290       LD     A,(BD+2)
367 80F4 FEFF        CP     OFFH
368 80F6 283E        JR     Z,FINCON
369 80F8 E60F        AND    OFH
370 80FA B7          OR     A
371 80FB 2839        JR     Z,FINCON
372 80FD 47          LD     B,A
373 80FE 110100       LD     DE,1
374                BM20: EQU  $
375 8101 19          ADD    HL,DE
376 8102 10FD        DJNZ   BM20
377 8104 1830        JR     FINCON          ;Fin de conversion
378                ;
379                UNZER: EQU  $
380 8106 3A0190       LD     A,(BD+1)
381 8109 FEFF        CP     OFFH
382 810B 2821        JR     Z,DEUZER       ;Deux zeros
383 810D 3A0090       LD     A,(BD)
384 8110 E60F        AND    OFH
385 8112 47          LD     B,A
386 8113 210000       LD     HL,0
387 8116 110A00       LD     DE,10

```

```

388          UZ1:      EQU  $
389 8119 19          ADD  HL,DE
390 811A 10FD        DJNZ  UZ1
391 811C 3A0190      LD   A,(BD+1)
392 811F FEFF        CP   OFFH
393 8121 2B13        JR   Z,FINCON
394 8123 E60F        AND  OFH
395 8125 47          LD   B,A
396 8126 110100      LD   DE,1
397          UZ10:     EQU  $
398 8129 19          ADD  HL,DE
399 812A 10FD        DJNZ  UZ10
400 812C 1B0B        JR   FINCON          ;Fin de conversion
401          ;
402          DEUZER:    EQU  $
403 812E 3A0090      LD   A,(BD)
404 8131 E60F        AND  OFH
405 8133 2600        LD   H,0
406 8135 6F          LD   I,A          ;Fin de conversion
407          ;
408          FINCON:     EQU  $
409 8136 220F92      LD   (OFFSET),HL      ;Memorisation
410 8139 AF          XOR  A
411 813A 320B90      LD   (DRIVE),A      ;Drive A
412 813D CD7593      CALL LECT          ;Lecture secteur
413          ;
414 8140 21BE92      LD   HL,ALALI
415 8143 CD0E92      CALL AFALPH
416 8146 21BE92      LD   HL,ALALI
417 8149 CD0E92      CALL AFALPH          ;A la ligne

```

```

418 814C 219592      LD    HL,MES5
419 814F CDED92      CALL AFALPH      ;Aff 5e message
420 8152 210E90      LD    HL,BUFF      ;@ debut buffer
421 8155 ED5B0F92     LD    DE,(OFFSET)
422 8159 1?          ADD    HL,DE
423 815A 7E          LD    A,(HL)
424 815B 321192      LD    (CAM),A      ;Caract a modifier
425 815E CB3F        SRL    A
426 8160 CB3F        SRL    A
427 8162 CB3F        SRL    A
428 8164 CB3F        SRL    A
429 8166 FE0A        CP     10
430 8168 3802        JR     C,EE1
431 816A C607        ADD    A,7
432                EE1:    EQU    $
433 816C C630        ADD    A,30H      ;Conversion ASCII
434 816E CD5ABB        CALL PRINT      ;Affichage
435 8171 3A1192      LD    A,(CAM)
436 8174 E60F        AND    0FH
437 8176 FE0A        CP     10
438 8178 3802        JR     C,EE2
439 817A C607        ADD    A,7
440                EE2:    EQU    $
441 817C C630        ADD    A,30H      ;Conversion ASCII
442 817E CD5ABB        CALL PRINT      ;Affichage (octet)
443                ;
444 8181 21BE92      LD    HL,ALAL1
445 8184 CDED92      CALL AFALPH      ;A la ligne
446 8187 21A992      LD    HL,MES6
447 818A CDED92      CALL AFALPH      ;Aff 6e message

```

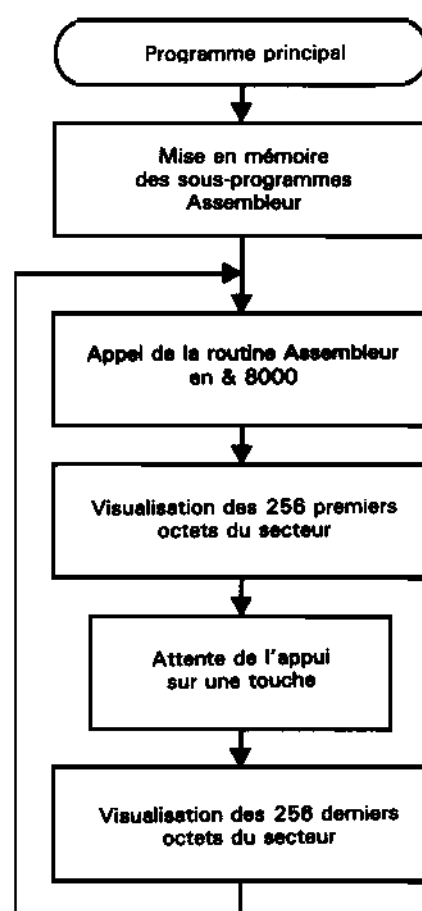
```

148 818D 210090      LD   HL,BD
149 8190 3EFF        LD   A,OFFH
150 8192 77          LD   (HL),A
151 8193 23          INC  HL
152 8194 77          LD   (HL),A
153 8195 3E02        LD   A,2
154 8197 320690      LD   (MAX),A      ;2 chiffres max
155 819A 210090      LD   HL,BD
156 819D 220790      LD   (PBUF),HL
157 81A0 CDFD92      CALL SAISIE
158                  ;
159 81A3 3A0190      LD   A,(BD+1)
160 81A6 FEFF        CP   OFFH
161 81A8 2B22        JR   Z,USC      ;Un seul chiffre
162 81AA 3A0090      LD   A,(BD)
163 81AD FE3A        CP   3AH
164 81AF 3B02        JR   C,CAH1
165 81B1 D607        SUB  7
166                  CAH1:  EQU  #
167 81B3 D630        SUB  30H      ;Conversion Hexa
168 81B5 47          LD   B,A
169 81B8 0E10        LD   C,16
170 81BB 3E00        LD   A,0
171                  B2C:  EQU  #
172 81BA 81          ADD  A,C
173 81BB 10FD        DJNZ B2C
174 81BD 4F          LD   C,A
175 81BE 3A0190      LD   A,(BD+1)
176 81C1 FE3A        CP   3AH
177 81C3 3B02        JR   C,CAH2

```

```
478 81C5 D607          SUB  7
479                  CAH2: EQU  $
480 81C7 D630          SUB  30H          ;Conversion Hexa
481 81C9 81          ADD  A,C
482 81CA 1803          JR   FINC2        ;Fin de conversion
483                  USC:   EQU  $
484 81CC 3A0090        LD   A, (BD)
485                  FINC2: EQU  $
486 81CF 210E90        LD   HL,BUFF
487 81D2 ED5B0F92      LD   DE, (OFFSET)
488 81D6 19          ADD  HL,DE
489 81D7 77          LD   (HL),A
490 81DB CD5593        CALL ECR          ;Ecriture secteur
491 81DB C9          RET                ;Retour au S/P Basic
492                  END
```


La logique du sous-programme Basic est la suivante :



Pour éviter d'entrer les codes opératoires de ce programme, le programme Basic reprend les valeurs hexadécimales correspondantes et les places en mémoire aux emplacements nécessaires.

La première exécution du programme se fait en tapant simplement **run**. Cela a pour effet de placer tous les codes du programme Assembleur en mémoire. Pour gagner du temps lors des prochaines exécutions du programme, vous pourrez taper **run 2000**.

Le programme Basic est le suivant :

```

1000 REM =====
1010 REM Lecture/Modification de secteurs de disquette
1020 REM =====
1030 '
1040 '
1050 REM -----
1060 REM Lecture des donnees Assembleur et mise en memoire
1070 REM -----
1080 FOR I=&9212 TO &9394 'Donnees et sous-programmes
1090   READ A$
1100   A=VAL("&" + A$)
1110   POKE I,A
1120 NEXT I
1130 '
1140 FOR I=&8000 TO &81DB 'Programme principal
1150   READ A$
1160   A=VAL("&" + A$)
1170   POKE I,A
1180 NEXT I
1190 '
1200 POKE &9009,&85 : POKE &900A,&84 'Donnees complementaires
1210 '
2000 '-----
2010 ' Execution du programme
2020 '-----
2030 CALL &8000 'Lecture d'un secteur
2040 PISTE=PEEK(&900C) : SECTEUR=PEEK(&900D)-64
2050 CLS : PRINT "          Visualisation de la piste";PISTE;"secteur";SECTEUR
2060 PRINT:PRINT
2070 AD=0 'Deplacement dans le secteur
2080 FOR I=&900E TO &910D STEP 16
2090   A$=HEX$(AD) : AD = AD + 16
2100   IF LEN(A$)=1 THEN A$="00" + A$
2110   IF LEN(A$)=2 THEN A$="0" + A$
2120   PRINT A$;" ";
2130   FOR J=0 TO 15
2140     A$=HEX$(PEEK(I+J))
2150     IF LEN(A$)=1 THEN A$="0" + A$
2160     PRINT A$;" ";
2170   NEXT J
2180   PRINT " ";
2190   FOR J=0 TO 15
2200     A=PEEK(I+J)
2210     IF A<32 THEN PRINT "."; ELSE PRINT CHR$(A);
2220   NEXT J
2230   PRINT
2240   NEXT I
2250   PRINT:PRINT " Appuyez sur une touche pour visualiser les 256 octets suivant
s ..."
2260   A$=INKEY$ : IF A$="" THEN 2260
2270   CLS : PRINT "          Visualisation de la piste";PISTE;"secteur";SECTEUR
2280   PRINT:PRINT
2290   FOR I=&910E TO &920D STEP 16
2300     A$=HEX$(AD) : AD = AD + 16
2310     IF LEN(A$)=1 THEN A$="00" + A$
2320     IF LEN(A$)=2 THEN A$="0" + A$
2330     PRINT A$;" ";
2340     FOR J=0 TO 15
2350       A$=HEX$(PEEK(I+J))
2360       IF LEN(A$)=1 THEN A$="0" + A$
2370       PRINT A$;" ";

```

```

380 NEXT J
390 PRINT " ";
400 FOR J=0 TO 15
410 A=PEEK(I+J)
420 IF A<32 THEN PRINT". "; ELSE PRINT CHR$(A);
430 NEXT J
440 PRINT
450 NEXT I
460 PRINT:PRINT" Appuyez sur une touche pour revenir au menu ..."
470 A$=INKEY$: IF A$="" THEN 2470
480 RUN 2000
490
000 REM -----
010 REM Messages et sous-programmes
020 REM -----
030 DATA 54,61,70,65,7A,20,44,20,70,6F,75,72,20,76,69,73
040 DATA 75,61,6C,69,73,65,72,20,75,6E,20,73,65,63,74,65
050 DATA 75,72,2C,D,A,65,74,20,4D,20,70,6F,75,72,20,6D
060 DATA 6F,64,69,66,69,65,72,20,75,6E,20,6F,63,74,65,74
070 DATA 20,70,61,72,74,69,63,75,6C,69,65,7C,2E,D,A,A
080 DATA FF,50,69,73,74,65,20,28,30,2D,33,39,29,20,3A,20
090 DATA FF,53,65,63,74,65,75,72,20,28,31,2D,39,29,20,3A
100 DATA 20,FF,4F,63,74,65,74,20,28,30,2D,35,31,31,29,20
110 DATA 3A,20,FF,41,6E,63,69,65,6C,6E,65,20,76,61,6C,65
120 DATA 75,72,20,3A,20,26,FF,4E,6F,75,74,65,6C,6C,65,20
130 DATA 76,61,6C,65,75,72,20,3A,20,26,20,FF,D,A,FF,3E
140 DATA 0,E,0,CD,34,BD,3E,1,E,1,CD,34,BD,3E,8,E
150 DATA 6,CD,34,BD,3E,7,E,8,CD,34,BD,21,FF,FF,2B,7C
160 DATA B5,20,FB,3E,8,C,0,CD,34,BD,C9,E5,F5,7E,FE,FF
170 DATA 28,6,CD,5A,BB,23,18,F5,F1,E1,C9,3A,6,90,57,1
180 DATA 0,0,2A,7,90,CD,6,BB,FE,D,28,3C,FE,7F,28,18
190 DATA F5,3E,8,CD,5A,BB,F1,CD,5A,BB,9,77,3,3E,5F,CD
200 DATA 5A,BE,79,BA,20,DC,18,20,79,B7,2B,D6,B,3E,8,CD
210 DATA 5A,BB,3E,20,CD,5A,BB,3E,8,CD,5A,BB,3E,8,CD,5A
220 DATA BD,3E,5F,CD,5A,BD,18,BA,3E,8,CD,5A,BB,3E,20,CD
230 DATA 5A,BB,C9,21,9,90,CD,D4,BC,22,3,90,79,32,5,90
240 DATA 3A,B,90,5F,3A,C,90,57,3A,D,90,4F,21,C,90,DF
250 DATA 3,90,C9,21,A,90,CD,D4,BC,22,3,90,79,32,5,90
260 DATA 3A,B,90,5F,3A,C,90,57,3A,D,90,4F,21,E,90,DF
270 DATA 3,90,C9,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
000 REM -----
010 REM Programme principal
020 REM -----
030 DATA 3E,2,CD,E,BC,21,12,92,CD,ED,92,CD,6,BB,F6,20
040 DATA 32,E,92,FE,64,28,9,FE,6D,28,5,CD,C1,92,18,EB
050 DATA 21,BE,92,CD,ED,92,21,63,92,CD,ED,92,21,0,90,3E
060 DATA FF,77,23,77,3C,2,32,6,90,21,0,90,22,7,90,CD
070 DATA FD,92,3A,1,90,FE,FF,28,20,3A,0,90,E6,F,47,3E
080 DATA A,4F,AF,81,10,FD,47,3A,1,90,E6,F,30,32,C,90
090 DATA FE,28,38,D,CD,C1,92,18,B7,3A,0,90,E6,F,32,C
100 DATA 90,21,BE,92,CD,ED,92,21,73,92,CD,ED,92,CD,6,BB
110 DATA CD,5A,BB,E6,F,C6,40,32,D,90,FE,4A,38,5,CD,C1
120 DATA 92,18,DE,3A,E,92,FE,64,28,2,18,8,AF,32,B,90
130 DATA CD,75,93,C9,21,BE,92,CD,ED,92,21,84,92,CD,ED,92
140 DATA 21,0,90,3E,FF,77,23,77,23,77,3E,3,32,6,90,21
150 DATA 0,90,22,7,90,CD,FD,92,3A,2,90,FE,FF,28,37,3A
160 DATA 0,90,E6,F,47,21,0,0,11,64,0,19,10,FD,3A,1
170 DATA 90,FE,FF,28,C,E6,F,B7,28,7,47,11,A,0,19,10
180 DATA FD,3A,2,90,FE,FF,28,3E,E6,F,B7,28,39,47,11,1
190 DATA 0,19,10,FD,18,30,3A,1,90,FE,FF,28,21,3A,0,90
200 DATA E6,F,47,21,0,0,11,A,0,19,10,FD,3A,1,90,FE
210 DATA FF,28,13,E6,F,47,11,1,0,19,10,FD,18,8,3A,0
220 DATA 90,E6,F,26,0,6F,22,F,92,AF,32,B,90,CD,75,93

```

```

4230 DATA 21,BE,92,CD,ED,92,21,BE,92,CD,ED,92,21,95,92,CD
4240 DATA ED,92,21,E,90,ED,5B,F,92,19,7E,32,11,92,CB,3F
4250 DATA CB,3F,CB,3F,CB,3F,FE,A,38,2,C6,7,C6,30,CD,5A
4260 DATA BB,3A,11,92,E6,F,FE,A,38,2,C6,7,C6,30,CD,5A
4270 DATA BB,21,BE,92,CD,ED,92,21,AB,92,CD,ED,92,21,0,90
4280 DATA 3C,FF,77,23,77,3E,0,32,6,90,21,0,90,22,7,90
4290 DATA CD,FD,92,3A,1,90,FE,FF,23,22,3A,0,90,FF,3A,3D
4300 DATA 2,06,7,06,30,47,E,1C,3E,0,81,10,FD,4F,3A,1
4310 DATA 90,FF,3A,38,2,06,7,06,30,81,18,3,3A,0,90,21
4320 DATA E,90,FD,5B,F,92,19,77,CD,55,23,CD,0,0,0,0

```

Utilisez le programme de checksum pour vérifier que les lignes de DATA entrées sont bien correctes (une seule erreur peut « planter » l'ordinateur, voire endommager la disquette qui se trouve dans le lecteur).

Les données affichées par le programme de checksum doivent être les suivantes :

```

C5 32 E7 2A 18 BC 41 A7 48 F5 A7 30 A9 9 A 80 E4 CE F0 66
F0 2A 6F 2A 5D 93 27 16 54 E9 F0 5D 56 C6 8F E7 C7 E C6 2C
98 4E 6B C 34 98 A3 51 BF D9 C4 AF A4 71 9A

```

Voici un exemple de listage de la piste 3 secteur 5 obtenu avec la disquette insérée dans le lecteur A :

Visualisation de la piste 3 secteur 4

```

000 F6 04 A1 20 0D 00 00 6D EC F1 16 20 EB 20 9E 20 ... ..1... ..
010 0D 00 00 E9 EF 0D 00 00 6D EC F4 0F 20 EC 20 16 .....1....
020 20 01 20 03 00 00 6E E6 EF 03 00 00 4E E6 F4 22 . ...n....."
030 20 22 20 01 20 B0 20 0D 00 00 E9 00 16 00 00 05 " . . ....
040 03 00 00 6E E6 EF 03 00 00 6E E6 F4 03 00 00 65 ...n.....n.....e
050 F8 00 14 00 0A 05 0D 00 00 6D EC EF FF 0E 28 03 .....1....(
060 00 00 65 F8 29 03 3E 00 14 05 A1 20 0D 00 00 6C ...e.).>.... ..1
070 EC F1 11 20 EB 20 9E 20 0D 00 00 E9 EF 0D 00 00 ... ..
080 6D EC F4 0F 20 EC 20 11 20 01 20 03 00 00 6E E6 1... .. ..n.
090 EF 03 00 00 6E E6 F4 22 20 22 20 01 20 B0 20 0D .....n.. " " . .
0A0 00 00 E9 00 07 00 1E 05 01 00 00 25 00 22 05 A1 .....%.(..
0B0 20 03 00 00 6D F5 EF 22 41 22 20 EB 20 BE 20 1C ...1.."A" . .
0C0 04 92 20 0E 20 01 97 20 BE 20 1C 04 92 20 0F 00 .., .. .., .., ..
0D0 21 00 32 05 A1 20 03 00 00 66 EF EF 22 53 22 20 $.2.. ..f.."S"
0E0 EB 20 BE 20 1C 05 92 20 10 03 BE 20 1C 06 72 2C . . .., .., ..,
0F0 1C 41 00 00 21 00 3C 05 A1 20 03 00 00 66 EF EF .A..$.<... ..f..

```

Appuyez sur une touche pour visualiser les 256 octets suivants ...

Visualisation de la piste 3 secteur 4

00	22	44	22	20	EB	20	BE	20	1C	05	92	2C	0E	01	BE	20	"D",
10	1C	06	92	2C	1C	01	00	00	22	00	46	05	A1	20	03	00	...,"F.. ..
20	00	66	EF	EF	22	49	22	20	EB	20	DE	20	1C	05	92	2C	.f.."I",
30	0F	01	BE	20	1C	06	92	2C	0F	09	07	00	50	05	01	C0	,P...
40	00	27	00	5A	05	0D	00	00	6E	73	65	E3	EF	0F	20	01	..Z....nse... .
50	C0	4E	6F	6D	62	72	65	20	64	65	20	73	65	63	74	65	.Nombre de secte
60	75	72	73	20	6C	75	73	00	14	00	64	05	83	20	1C	95	urs lus...d.. ..
70	93	20	01	C0	4C	65	63	74	75	72	65	00	28	00	6E	05	. ..Lecture.(.n.
80	0D	00	00	61	E4	EF	1C	03	90	20	01	C0	41	64	72	65	...a..... ..Adre
90	73	73	65	20	64	65	20	6C	65	63	74	75	72	65	20	6E	ssa de lecture n
A0	6F	6D	20	00	22	00	78	05	0D	00	00	6E	6C	E5	EF	0E	om ."x....nl...
B0	20	01	C0	4E	6F	6D	62	72	65	20	64	65	20	6C	65	63	..Nombre de lec
C0	74	75	72	65	73	00	0D	00	82	05	03	00	00	6C	F5	EF	tures.....l..
D0	22	22	00	13	00	8C	05	9E	20	0D	00	00	E9	EF	0F	20	".....
E0	EC	20	19	0B	20	00	1C	00	96	05	20	20	0D	00	00	74	t
F0	65	F2	EF	FF	12	28	0D	00	00	61	E4	F4	0D	00	00	E9	e....(....a.....

Appuyez sur une touche pour revenir au menu ...

9/8.11

CAPS LOCK interactif

Qui n'a pas, un jour, été confronté à l'angoissante question : « Suis-je en caractère MAJUSCULE ou minuscule ? », lors de la frappe d'un listing, ou de l'utilisation d'un logiciel Basic.

L'idéal est de posséder un voyant **CAPS LOCK** comme sur beaucoup de claviers de type professionnel, mais la modification du clavier demande une certaine habitude dans le maniement du fer à souder, et la possession d'un matériel adéquat, sans compter la perte de la garantie si vous avez acheté votre CPC récemment.

Nous nous sommes donc attachés à la possibilité de reconnaître cet état dans les abîmes des adresses de la mémoire RAM des CPC 664 et 6128. Vous pourrez faire de même avec un CPC 464 en sachant que les adresses mémoires que nous vous donnons ci-dessous ne sont pas forcément identiques, mais doivent certainement être très proches.

Présentation de CAPS LOCK

Après quelques Pokes appropriés aux environs de la zone d'adresses &B6xx, et quelques désagréments — vous pouvez effectuer des Pokes partout dans la mémoire de votre CPC sans aucun danger pour celui-ci, mais nous vous conseillons d'avoir sauvegardé vos programmes si vous en aviez entré un, et de retirer toute disquette ou cassette de votre lecteur — nous avons découvert quelques fonctions intéressantes, dont la possibilité sur CPC 664 et CPC 6128 de forcer le mode majuscule. En effet, l'adresse mémoire &B632 contient l'état **CAPS LOCK** et permet de forcer celui-ci.

Essayez l'instruction : **POKE &B632, &FF**

puis frappez sur les touches de caractères alphabétiques : les caractères sont affichés en majuscule.

Le retour à l'état initial est effectué par l'instruction :

POKE &B632, &00.

Cet état peut, de plus, être lu grâce à l'instruction :

X = PEEK (&B632)

qui place la valeur &00 (ou zéro) dans la variable X lorsque vous êtes en mode minuscule, et la valeur &FF (ou 255) lorsque la touche **CAPS LOCK** a été activée.

Nous pouvons ainsi, dans un logiciel **basic**, forcer le mode majuscule pour éviter d'avoir recours à l'instruction : **A\$ = UPPER\$ (A\$)** qui transforme le contenu d'une variable alphanumérique en caractères majuscules.

Mais nous pouvons tirer parti, plus efficacement, de cette fonctionnalité en l'utilisant dans un programme **Basic**, pour afficher l'état de l'activité **CAPS LOCK**.

Utilisation basic de l'adresse mémoire &B632

Nous vous proposons, ci-après, un programme qui permettra l'affichage dans le bord supérieur droit de l'état **CAPS LOCK** de votre CPC.

EXPOSE DU PROBLEME

La méthode pour reconnaître cet état étant expliquée ci-dessus, il nous reste à en décrire la mise en œuvre.

Il nous faut tester régulièrement le contenu de l'adresse mémoire &B632. Ce test est impensable avec l'instruction : **X = INP(&B632)** et le traitement de ce test, ou l'appel à un sous-programme de traitement, toutes les deux ou trois lignes **Basic**, nous gênerions de la place mémoire.

Une possibilité intéressante du **Basic Locomotive** est celle d'intégrer, dans un programme **basic**, des interruptions logicielles grâce à l'instruction :

EVERY <delai> [, <N° de chronomètre>] GOSUB < N° ligne> delai est le temps au bout duquel le sous-programme est appelé une nouvelle fois, et exprimé en multiples de 50 millisecondes.

Nous allons donc placer dès le début de notre programme **Basic** cette instruction, qui permettra de l'interrompre régulièrement, et d'effectuer un sous-programme permettant le traitement du test de l'activité **CAPS LOCK** (on se reportera avec intérêt à la description et l'exemple donné sur cette instruction, à la Partie 4 chapitre 1.2 page 54) ; nous placerons ce sous-programme en fin de programme, par exemple en lignes 60000 et suivantes.

Nous choisirons, par exemple, d'effectuer le traitement toutes les 2 secondes, afin de ne pas trop ralentir le programme en cours. La valeur de **delai** sera donc égale à :

$$\text{delai} = 2 / (50 * 10^{-3}) = 40$$

Ce qui nous amène à la ligne d'instruction :

EVERY 40,2 GOSUB 60000.

Ainsi, toutes les deux secondes, si **CAPS LOCK** est active, nous viendrons écrire grâce à l'instruction **LOCATE** en haut et à droite de l'écran.

Deux autres problèmes subsistent tout de même :

1/ Le curseur texte doit revenir à sa position initiale, quelle que soit celle-ci suite à cet affichage.

2/ Les coordonnées d'affichage ne sont pas identiques selon le mode d'affichage choisi.

Le premier problème est résolu grâce à une autre adresse mémoire, dans laquelle est enregistrée la position du curseur texte : **&B726** pour les CPC 664 et CPC 6128.

Il nous suffira donc de sauvegarder temporairement le contenu de cette adresse dès le début de la routine, et de le restituer quand la routine sera exécutée. — Vous pouvez essayer, par quelques POKEs hasardeux (POKE **&B726,xx**) de modifier directement la position du curseur de texte, en mode direct.

Le deuxième problème nous préoccupe surtout si nous ne connaissons pas par avance le mode graphique dans lequel notre routine va opérer. Heureusement, la RAM de votre CPC renferme encore ce précieux renseignement à l'adresse hexadécimale **&B7C3**.

Nous lirons donc le contenu de cette adresse par l'instruction :

Variable = PEEK (&B7C3)

— Vous pouvez de même essayer la commande :

PRINT PEEK (&B7C3) en mode direct, et selon les trois modes écran choisis.

Une autre particularité de cette adresse est qu'elle permet de modifier le mode courant au niveau de l'affichage. Si, par exemple, vous vous trouvez en mode 0 et que vous forcez l'affichage en mode 1 par la commande : **POKE &B7C3, 1** vous obtiendrez alors un style d'affichage quelque peu bizarre, que certains concepteurs de logiciels de jeux utilisent quelquefois.

Afin de ne pas trop nous écarter du sujet qui nous préoccupe, nous vous laisserons le soin d'expérimenter cette particularité.

APPLICATION AU PROGRAMME

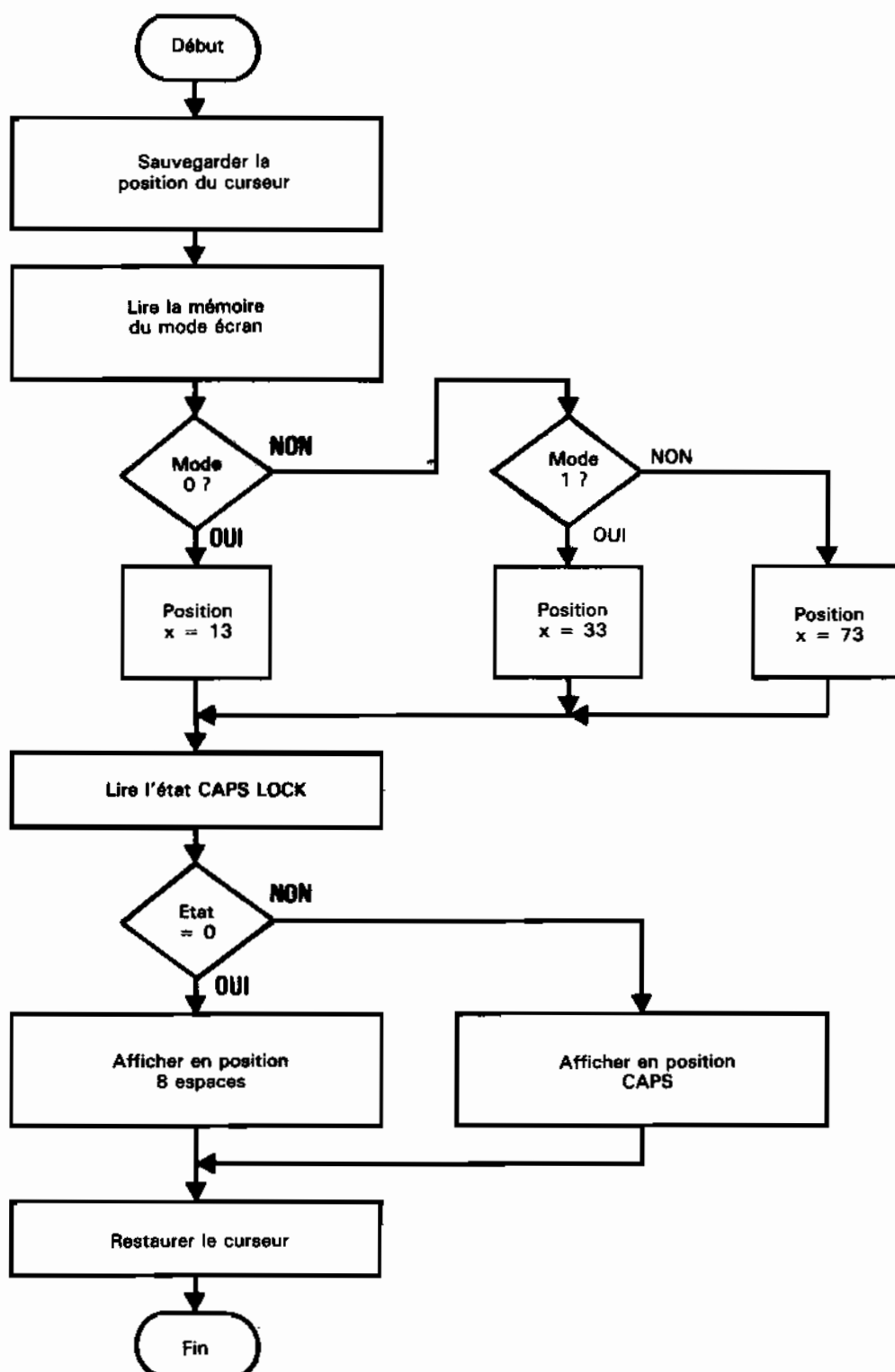
Les différentes positions que nous avons choisies sont ainsi, selon le mode :

— MODE 0 → LOCATE 13,1

— MODE 1 → LOCATE 33,1

— MODE 2 → LOCATE 73,1

Nous obtiendrons l'algorithme de fonctionnement suivant :



La traduction en basic de cet algorithme donne le programme suivant :

```

1 EVERY 40,2 GOSUB 60000
.
.
.
100 REM *** INSEREZ ENTRE LES LIGNES ***
110 REM ***      1 ET 60000      ***
120 REM *** DE VOTRE PROGRAMME QUI ***
130 REM *** FONCTIONNERA SOUS ***
140 REM *** INTERRUPTION LOGICIELLE ***
.
.
.
.
60000 REM *** ROUTINE CAPS LOCK ***
60005 REM *****
60010 TXTcurseur = PEEK(&B726):REM sauve
garde de la position curseur
60020 SCRmode = PEEK(&B7C3):REM lecture
du mode ecran
60030 KMcaps = PEEK(&B632):REM lecture d
e l'etat CAPS LOCK
60040 IF SCRmode = 0 THEN X = 13 : GOTO
60070:REM mode 1?
60050 IF SCRmode = 1 THEN X = 33 : GOTO
60070:REM mode 2?
60060 X = 73:REM alors mode 2
60070 IF KMcaps = 0 THEN 60110:REM test
CAPS LOCK
60080 LOCATE X,1:REM positionne
60090 PRINT CHR$(&8F);CHR$(&8F);"CAPS";C
HR$(&8F);CHR$(&8F)
60100 GOTO 60130
60110 LOCATE X,1:REM positionne
60120 PRINT SPACE$(8):REM efface CAPS
60130 POKE &B726,TXTcurseur:REM restitue
l'ancienne position curseur
60140 RETURN:REM fin d'interruption
60150 REM *****

```

Vous remarquerez les variables utilisées :

- TXTcurseur : curseur texte - TeXTe curseur
- SCRmode : mode écran — SCReen mode
- KMcaps : touche CAPS LOCK — Key Manager caps
- X : variable de position dans l'axe X de l'écran et veillerez à ne pas les utiliser dans le programme que vous insérerez entre les lignes 1 et 60000.

Le numéro de chronomètre choisi en ligne 1 est le chronomètre 2 qui vous laisse le choix d'utiliser les chronomètres 0 et 1, qui sont plus prioritaires, ainsi que le chronomètre 3, moins prioritaire. Ce numéro n'étant pas important, vous pouvez le modifier à votre gré.

Vous pouvez aussi faire varier le temps entre deux appels du sous-programme, mais en sachant que si vous le diminuez fortement, vous risquez de ralentir considérablement le programme à exécuter, et, dans le cas contraire, vous ne verriez pas apparaître le message **CAPS** assez souvent, notamment en cas de « scrolling » de l'écran, ou d'effacement.

UTILISATION DU SOUS-PROGRAMME

Nous vous conseillons de sauvegarder le sous-programme tel quel, sans les lignes 100 à 140, sur disquette de préférence, et en ASCII par l'instruction : **SAVE "CAPS.ASC", A.**

Vous pourrez ensuite l'intégrer à n'importe quel programme basic que vous possédez, et le charger dans la mémoire de votre CPC, grâce à la commande :

MERGE "CAPS.ASC" ou CHAIN MERGE "CAPS.ASC"

LIMITES DU SOUS-PROGRAMME EN BASIC

Ce type de sous-programme fonctionnant en interruption dite *interruption logicielle* est particulièrement adapté lorsque le programme est exécuté, mais non lorsque votre Amstrad attend une commande, comme, par exemple, lorsque vous frappez un listing. La solution est alors d'entrer au cœur même du fonctionnement de votre CPC et d'en détourner certaines de ses tâches, comme nous allons vous l'expliquer ci-après.

L'interruption matérielle de l'Amstrad

Contrairement à ce que vous êtes en droit de croire, le moniteur Basic faisant parfaitement illusion, le microprocesseur (le Z 80) est continuellement en train de travailler : grâce aux composants qui l'entourent et aux programmes en ROMs, il effectue entre autre la remise à jour du compteur **TIME** tous les 300^e de seconde, la scrutation du clavier tous les 50^e de seconde, et une somme inimaginable d'exécutions de routines internes au système d'exploitation centrale et à l'interpréteur Basic ; seul le contrôleur de visualisation **CRTC 6845** se permet d'interrompre le Z 80 (pour les plus connaisseurs d'entre vous : en positionnant la broche **WAIT**) pour remettre à jour l'écran.

Pour effectuer ces multiples opérations, le microprocesseur est interrompu par un signal, provenant du composant **GATE ARRAY** (composant développé spécialement par Amstrad pour les CPC), tous les 300^e de seconde, et appliqué sur sa broche **IRQ**.

Ce signal force le Z 80 à se brancher à l'adresse (en mémoire RAM) hexadécimale &0038 — l'interruption est nommée **RST 7 (ReSTart 7)** (voir Partie 4, chapitre 2.8, page 4).

Par quelques POKEs à partir de cette adresse et les suivantes ou le désassemblage suivant :

0030	RST	0	C7	-
0031	EXX		D9	-
0032	LD	HL,&002B	212B00	!+-
0035	LD	(HL),C	71	q
0036	JR	&0040	1808	--
0038	JP	&B941	C341B9	-A-
003B	RET		C9	-
003C	NOP		00	-
003D	NOP		00	-
003E	NOP		00	-
003F	NOP		00	-

Adresse Code Opérations Code hexadécimal des codes opérations Code ASCII s'il y a lieu

Si vous ne possédez pas de désassembleur, vous pouvez obtenir les valeurs hexadécimales par l'instruction :

PRINT HEX\$(PEEK(&0038))... et suivantes.

A ces adresses, nous trouvons un saut à l'adresse &B941 (JP B941, codé en hexadécimal : C3 41 B9).

A partir de cette dernière adresse est effectué le traitement de l'interruption IRQ, dont voici une partie désassemblée :

B941	DI		F3	-
B942	EX	AF,AF	08	-
B943	JR	C,&B978	3833	B3
B945	EXX		D9	-
B946	LD	A,C	79	y
B947	SCF		37	7
B948	EI		FB	-
B949	EX	AF,AF	08	-
B94A	DI		F3	-
B94B	PUSH	AF	F5	-
B94C	RES	2,C	CB91	--
B94E	OUT	(C),C	ED49	-I
B950	CALL	&00B1	CDB100	----
B953	OR	A,A	B7	-
B954	EX	AF,AF	08	-
B955	LD	C,A	4F	0

B956	LD	B, &7F	067F	
B958	LD	A, (&B831)	3A31B8	: 1-
B95B	OR	A, A	B7	-
B95C	JR	Z, &B972	2814	(-
B95E	JP	M, &B972	FA72B9	-r-

Dans cette routine sont, entre autres, sauvegardés les registres du Z 80, et effectuée la commutation de la ROM BIOS du système d'exploitation, située entre l'adresse &0000 et &3FFF à la place de la RAM (on se reportera à l'organisation mémoire des CPCs (Partie 2 chapitre 1 page 3).

La commutation effectuée, un sous-programme à l'adresse &00B1 est appelé dans cette ROM, par l'instruction située à l'adresse &B950 : **CALL &00B1** (à peu près équivalent du GOSUB en Basic).

Comme cette instruction se trouve en RAM, il est possible de modifier l'adresse à laquelle elle saute, pour y mettre l'adresse d'une de nos routines en langage machine ; l'important étant d'effectuer à la fin de celle-ci un saut direct à l'adresse de la ROM.

Ces explications commençant à devenir ardues et impliquant des connaissances en langage machine, nous vous conseillons de vous reporter à la Partie, chapitre 2 traitant du langage d'assemblage, et aux instructions du Z 80.

Rassurez-vous tout de même, si vous ne possédez pas d'assembleur, un listing basic permettant de charger la routine en code machine vous sera proposé à la fin du chapitre.

Dérouter l'interruption

Pour dérouter la routine d'interruption, nous allons donc écrire aux adresses &B951 et &B952 (*attention* : ces adresses ne sont valables que pour les CPC 664 et CPC 6128) le numéro de l'adresse où débutera notre routine de traitement. Il serait, par ailleurs, préférable auparavant, d'inhiber toute nouvelle interruption lors de l'installation, et de les autoriser à nouveau à la sortie de la routine. Vous trouverez l'ordinogramme général de cette routine en page 12.

Une fois l'adresse installée, au maximum un 300^e de seconde plus tard, un branchement à la nouvelle adresse sera effectué, vous comprendrez donc qu'il est nécessaire d'avoir chargé auparavant en mémoire notre sous-programme de détection et d'affichage de l'état **CAPS LOCK**.

La routine Assembleur de détection CAPS LOCK

PREPARATION DU RETOUR EN FIN DE ROUTINE

A la fin de la routine, il nous faut retourner à l'adresse que nous avons détournée, c'est-à-dire en &00B1, et pour cela, deux styles de programmes s'ouvrent à nous :

- Une façon très brutale de retourner à cette adresse est d'y effectuer un branchement inconditionnel appelé **Jump**, par l'instruction suivante : **JP &00B1** ;
- Une façon plus astucieuse puisqu'elle nous permet de terminer la routine de la même façon qu'une routine machine classique, c'est-à-dire par l'instruction **RET** (de **RETurn**), ce qui nous semble préférable.

Il faut savoir que l'instruction **RET** prend la dernière adresse contenue dans la pile (pile repérée par un registre du Z 80 appelé **SP**, de **Stack Pointer** = pointeur de pile) et la place dans le compteur programme, c'est-à-dire qu'elle effectue un branchement à cette adresse. (Lorsque l'on appelle un sous-programme en Assembleur, c'est l'adresse de retour qui est mise dans la pile — on dit aussi empilée — donc, lors d'un **RET**, on retourne à cette adresse.)

Nous allons ainsi empiler l'adresse &00B1, mais les instructions du microprocesseur Z 80 ne permettent pas de placer directement dans la pile, par contre les registres de travail sont empilables.

Pour cela, il nous faut charger un registre 16 bits (par exemple **HL** avec cette adresse, et empiler **HL**, grâce aux instructions qui suivent :

```
LD HL, &00B1  
PUSH HL.
```

L'ALGORITHME

Nous avons en partie retenu la forme de l'algorithme proposée plus haut pour réfléchir au programme assembleur.

En revanche, contrairement à l'instruction d'interruption **EVERY** du Basic, l'interruption **IRQ** n'est pas modifiable logiquement sans causer de graves perturbations (mais non destructives, rassurez-vous) dans le fonctionnement de votre CPC. Nous n'allons pas laisser le CPC afficher 300 fois par secondes (c'est-à-dire environ toutes les 3.3 millisecondes) le message **CAPS LOCK**, ce serait une perte de temps inutile, qui retarderait le fonctionnement de l'ordinateur, et absurde, car personne n'est capable de modifier l'état de cette touche aussi rapidement.

Nous utiliserons un compteur pour l'affichage, qui sera réalisé toutes les &FF fois (contenu maximal du compteur), c'est-à-dire toutes les $((15 \times 16) + 15) \times 3,3 = 841,5$ ms, proche de 1 seconde.

Notre programme devenant de plus en plus complexe, il est nécessaire de le structurer, donc d'en écrire d'abord l'algorithme que nous vous proposons ci-dessous :

DEBUT

Installer la routine de détournement de l'interruption matérielle $\overline{IRQ}$

FIN

DEBUT

CO

Routine détournée par le traitement précédent.
C'est elle qui sera appelée régulièrement lors d'une interruption matérielle $\overline{IRQ}$

FINCO

Charger l'adresse de retour dans la pile

Tester l'état de **CAPS LOCK**

CO

Cet état est contenu dans une adresse en RAM

FINCO

SI **CAPS LOCK** est activée

ALORS

Décrémenter le compteur d'affichage

CO

Afin de régénérer régulièrement l'affichage de **CAPS LOCK**
(environ toutes les secondes)

FINCO

SI le compteur d'affichage a atteint la valeur 1

ALORS

Positionner à 1 le flag signalant l'affichage de **CAPS LOCK**

Afficher sur l'écran le message **CAPS**

CO

Afficher sera effectué par un sous-programme

FINCO

FINSI

SINON

Tester l'état du flag signalant l'état de l'affichage de **CAPS LOCK**

Si le flag est positionné à 1

CO

flag à 1 si **CAPS LOCK** est affiché

FINCOALORS

Annuler le flag d'affichage

Effacer le message **CAPS**

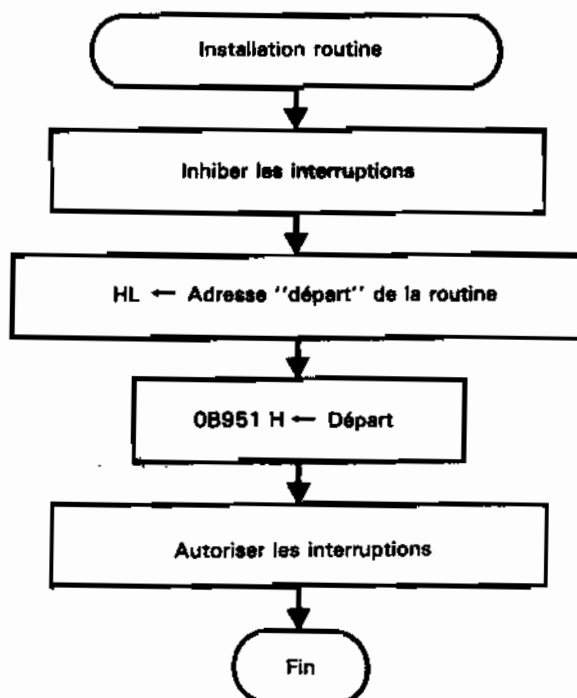
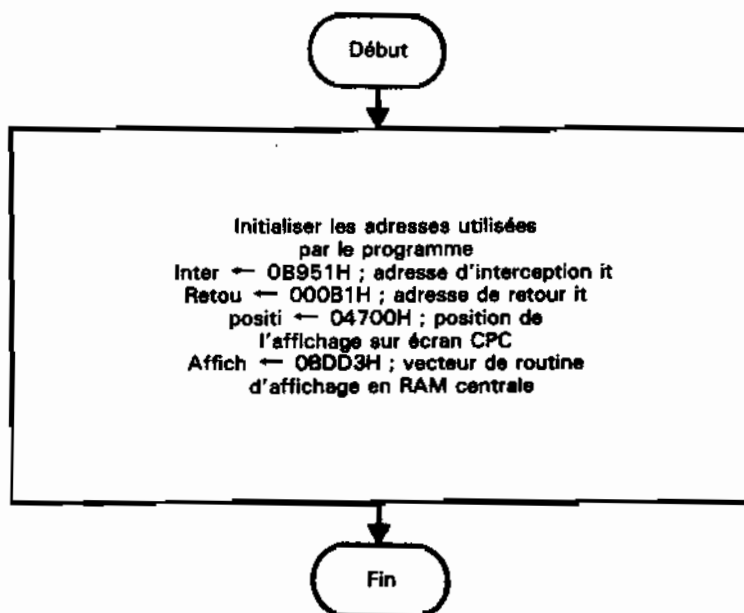
FINSIFINSIFINCO

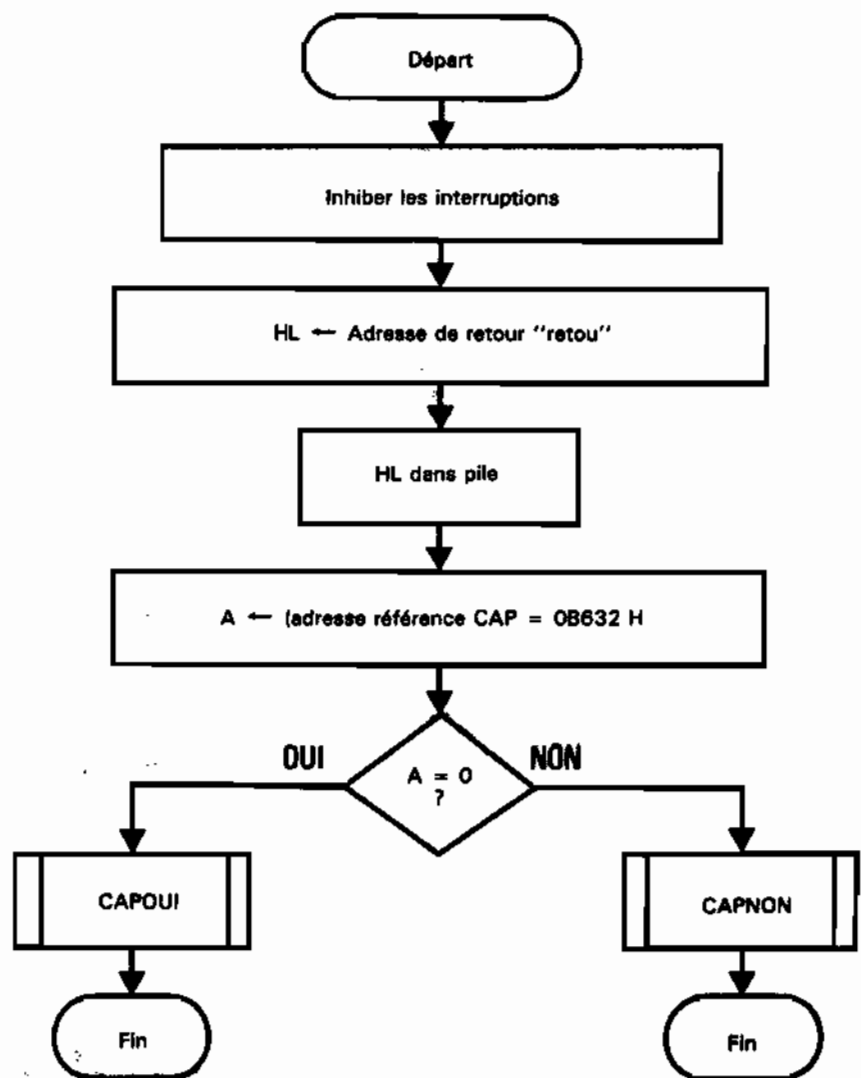
Retour à la suite du traitement de la routine d'interruption

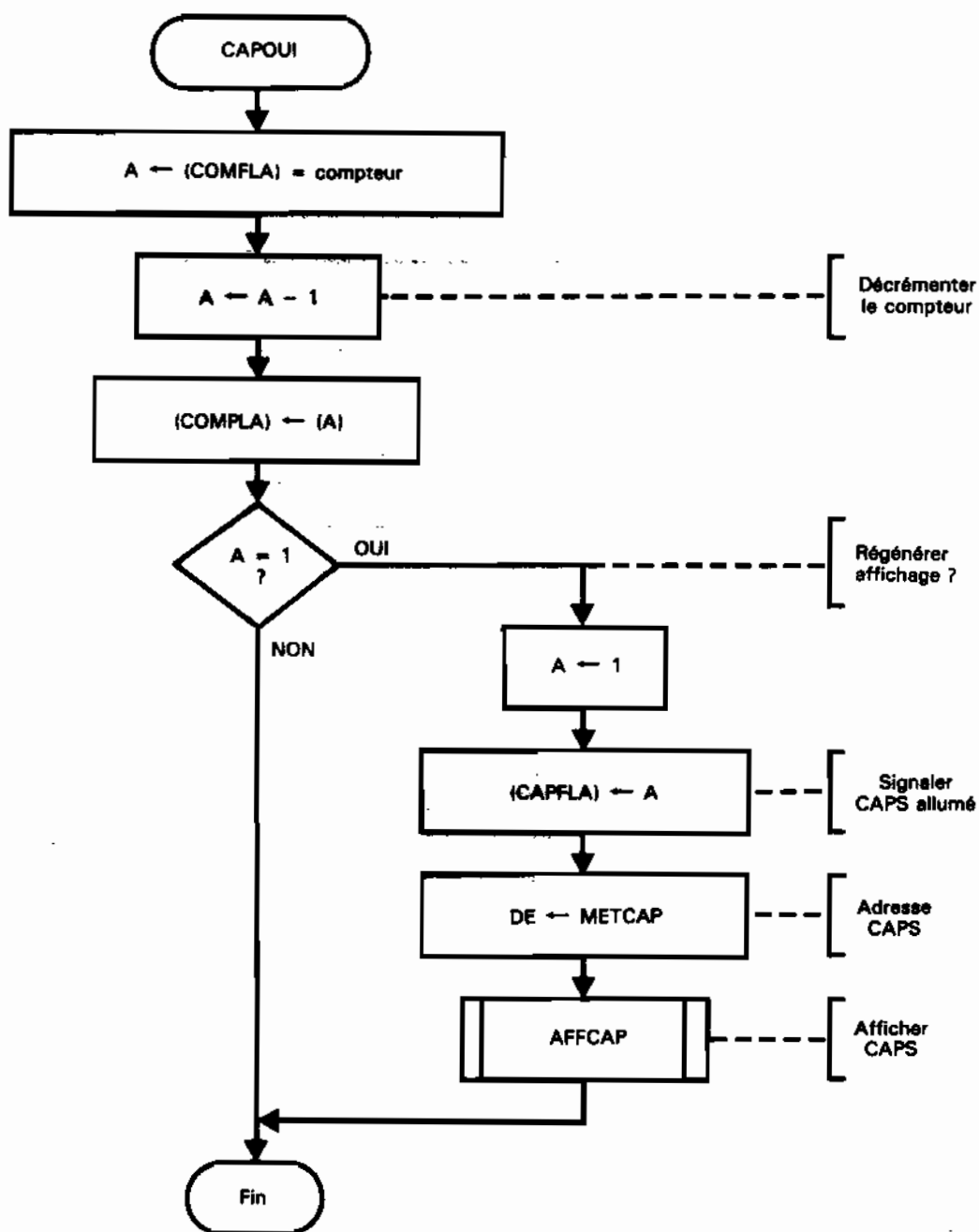
FINCO**L'ORDINOGRAMME**

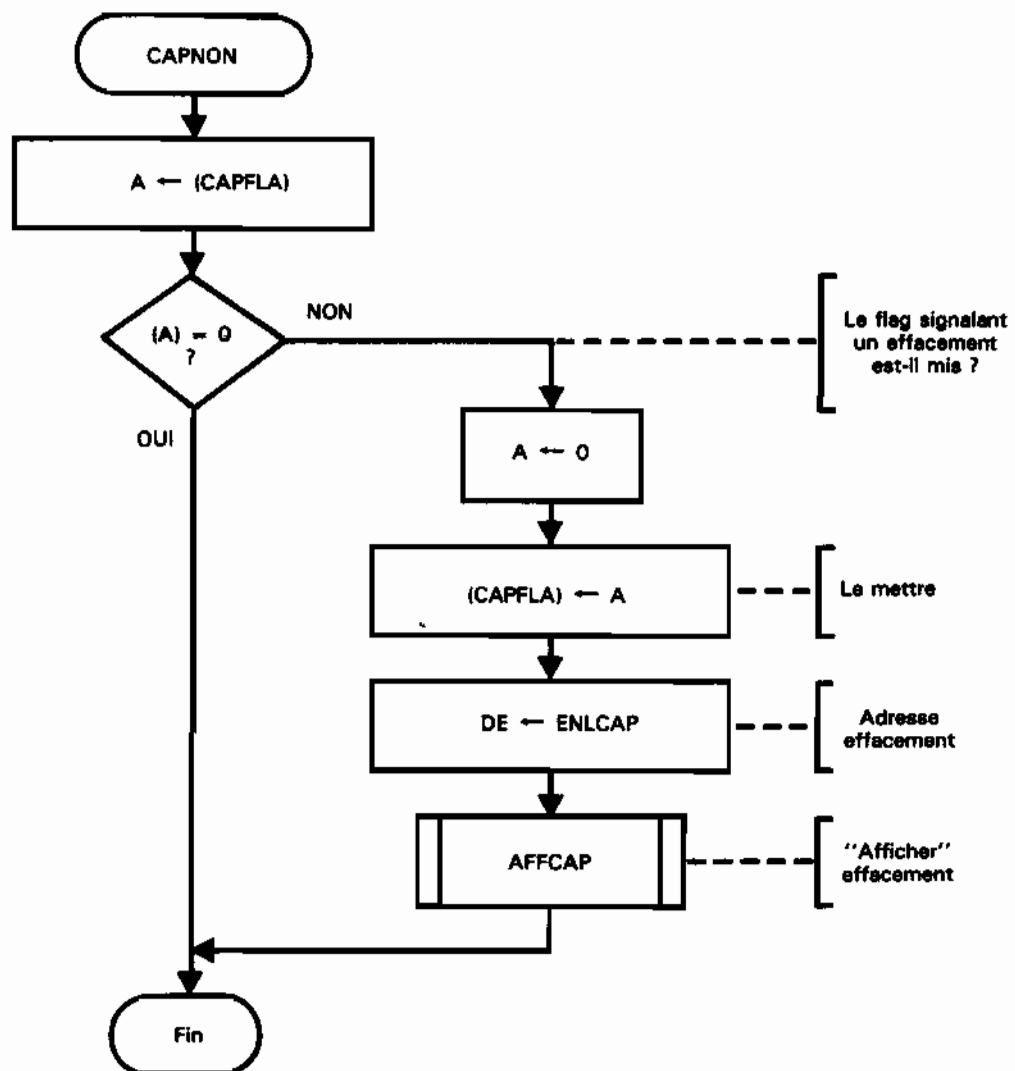
Nous vous proposons l'ordinogramme (page suivante), qui est la représentation graphique de l'algorithme précédent, faisant apparaître le langage spécifique au microprocesseur utilisé.

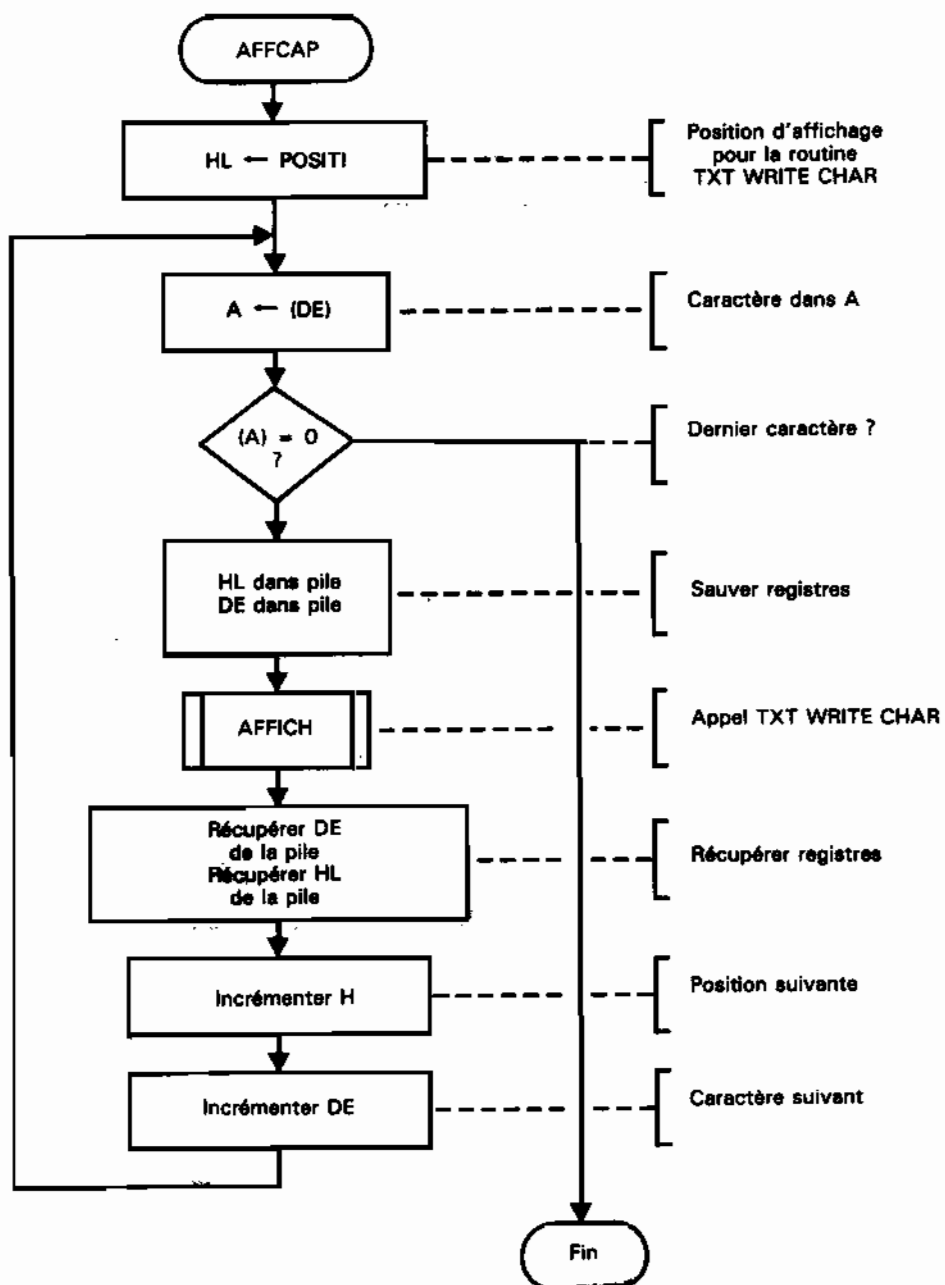
On remarquera dans cet ordinogramme l'utilisation de symboles de sous-programme (les rectangles avec deux doubles barres verticales), qui renvoient, soit à un sous-programme (dans le cas de l'affichage : **AFFCAP**) ou à un autre programme, pour éclaircir la présentation (dans le cas des traitements **CAPOUI** et **CAPNON**).











LE PROGRAMME ASSEMBLEUR

Nous vous livrons ainsi le programme Assembleur qui découle de l'ordonnogramme précédent.

Les non-initiés noteront la signification des colonnes de gauche à droite :

— les numéros de lignes (de 1 à 147) qui servent uniquement de repères lors de l'écriture du programme ;

- les adresses (de A000 à A067) qui sont générées à la compilation du programme, grâce à l'origine fixée par l'ordre origine : **ORG 0A000H** ;
- les codes machine exécutables (codes hexadécimaux) qui sont générés à la compilation à partir des mnémoniques ;
- les étiquettes (suivies du signe :) ;
- les mnémoniques (LD A,xx : CP xx ; JR yyyy ; ...) qui sont les instructions symbolisées de l'assembleur ;
- les commentaires qui sont précédés du caractère ; et qui servent à donner des précisions sur les opérations effectuées (équivalent de l'instruction REM du Basic). Nous tenons, et espérons que vous ferez de même, à faire apparaître ces commentaires, car un programme en Assembleur est difficilement lisible sans commentaires, si vous désirez le réétudier ultérieurement.

```

1          ;PROGRAMME D'IMPLANTATION
2          ;D'UNE ROUTINE DETOURNANT
3          ;LA ROUTINE DE TRAITEMENT
4          ;D'INTERRUPTION (IT) IRQ
5          ;*****
6          ;
7          ;
8          ;TABLE DE REFERENCE
9          ;*****
10         ;
11         ;ADRESSE ROUTINE DETOURNEE
12         INTER: EQU 0B951H
13         ;ADRESSE RETOUR EN ROM INFERIEURE
14         RETOU: EQU 000B1H
15         ;POSITION D'AFFICHAGE SUR L'ECRAN
16         POSITI: EQU 04700H
17         ;VECTEUR D'AFFICHAGE EN RAM
18         ;(TXT WRITE CHAR)
19         AFFICH: EQU 0BDD3H
20         ;ADRESSE DE REFERENCE ACTIVITE CAP
21         MEMCAP: EQU 0B632H
22         ;*****
23         ;
24         ;
25         ;ADRESSE DE REFERENCE D'ASSEMBLAGE
26         ;*****
27         ;
28         ORG 0A000H
29         LOAD 0A000H
30         ;*****
31         ;
32         ;
33         ;ROUTINE D'INSTALLATION POUR
34         ;DETOURNEMENT DE LA ROUTINE
35         ;DE TRAITEMENT D'INTERRUPTION
36         ;*****
37         ;
38 A000 F3          DI          ;INHIBE LES INTERRUPTIONS
39 A001 2109A0      LD HL,DEPART ;CHARGE L'ADRESSE DE
40 A004 2251B9      LD (INTER),HL ;LA ROUTINE D'IT
41 A007 FB          EI          ;AUTORISE LES INTERRUPTIONS
42 A008 C9          RET         ;ROUTINE INSTALLEE -> FIN

```

```

43 ; *****
44 ;
45 ;
46 ; PROGRAMME DE TEST DE L'ETAT
47 ; CAPS LOCK ET AFFICHAGE EN
48 ; CONSEQUENCE EN INTERRUPTION
49 ; *****
50 ;
; ROM E DEPART:; PREPARATION AU RETOUR EN
52 ; ROM EN FIN DE TRAITEMENT
53 A009 F3 DI ; INHIBITION DES INTERRUPT
54 A00A 21B100 LD HL, RETOU ; SAUVEGARDE DANS PILE
55 A00D E5 PUSH HL ; DE L'ADRESSE DE RETOUR E
56 ; ROM AFIN DE TERMINER SUR 'RET'
57 ;
58 A00E 3A32B6 DEBUT: LD A, (MEMCAP) ; ETAT CAPSLOCK
59 A011 FE00 CP 00 ; ACTIVE EN RAM?
60 A013 2B18 JR Z, CAPNON ; SAUT SI NON ACTIVE...
61 ;
LD6B, ( CAPQUI:; CAPS LOCK A ETE ACTIVE
63 A015 3A66A0 LD A, (COMFLA) ; COMPTEUR AFFICHAGE
64 A018 3D DEC A ; EST DECREMENTE
65 A019 3266A0 LD (COMFLA), A ; ET A NOUVEAU SAUVE
66 A01C FE01 CP 01H ; EST-IL EGAL A 1
67 A01E 200B JR NZ, CAPMIS ; NON -> PAS AFFICHAGE
68 A020 3E01 LD A, 01H ; CHARGER LE FLAG CAPS POU
69 A022 3265A0 LD (CAPFLA), A ; SIGNALER L'AFFICHAGE
70 ; DE CAPS LOCK EFFECTUE
71 A025 1153A0 LD DE, METCAP ; CHARGER DE AVEC
72 ; L'ADRESSE DE DEBUT DU MESSAGE
73 ; CAPS LOCK ENFONCE
74 A02B CD41A0 CALL AFFCAP ; AFFICHER MESSAGE
75 A02B 1B3A CAPMIS: JR RETOUR ; SAUT A FIN
76 ;
77 ; CAPS LOCK NON ACTIVE EN RAM
78 A02D 3A65A0 CAPNON: LD A, (CAPFLA) ; CHARGE FLAG
79 A030 FE00 CP 00H ; FLAG DEJA EFFACE?
80 A032 2B0B JR Z, CAPEFF ; SI OUI -> SAUT
81 A034 3E00 LD A, 00 ; SI NON -> POSITIONNER
82 A036 3265A0 LD (CAPFLA), A ; A ZERO LE FLAG
83 A039 115CA0 LD DE, ENLCAP ; CHARGER DE AVEC
84 ; L'ADRESSE DE DEBUT DU MESSAGE
85 ; 'ESPACES' D'EFFACEMENT
86 A03C CD41A0 CALL AFFCAP ; AFFICHER MESSAGE
87 A03F 1B26 CAPEFF: JR RETOUR ; SAUT A FIN
88 ;
89 ;
90 ; AFFICHAGE D'UN MESSAGE DONT
91 ; L'ADRESSE DU PREMIER CARACTERE
92 ; EST EN (DE) ET LA POSITION
93 ; CHARGEE DANS HL
94 ;
95 A041 210047 AFFCAP: LD HL, POSITI ; POSITION ECR.
96 A044 1A AFFCA1: LD A, (DE) ; CHARGER CARACTER
97 A045 FE00 CP 00 ; FIN DU MESSAGE?
98 A047 CB RET Z ; SI OUI -> RETOUR
99 A048 E5 PUSH HL ; SAUVER HL
100 A049 D5 PUSH DE ; SAUVER DE
101 A04A CDD3BD CALL AFFICH ; EXECUTER LA ROUTINE
102 ; D'AFFICHAGE EN ROM, DONT LE
103 ; VECTEUR EST EN RAM
104 A04D D1 POP DE ; RECUPERER DE

```

```

105 A04E E1          POP HL          ; RECUPERER HL
106 A04F 24          INC H           ; INCREMENTER POUR POSITIO
107                ; SUIVANTE
108 A050 13          INC DE           ; INCREMENTER DE POUR CARA
109                ; TERE SUIVANT A AFFICHER
110 A051 18F1        JR AFFCA1        ; RECOMMENCER EN AFFCA1
111                ;
112                ;
113                ; MESSAGE 'CAPS' + VIDEO INVERSE
114                ; *****
115                ;
116 A053 8F8F4341    METCAP:          DEFB 08FH,08FH,'CAPS'
116 A057 5053
117 A059 8F8F00          DEFB 08FH,08FH,00
118                ;
119                ;
120                ; MESSAGE ESPACES POUR PERMETTRE
121                ; L'EFFACEMENT DU MESSAGE 'CAPS'
122                ; *****
123                ;
124 A05C 20202020    ENLCAP:          DEFB 020H,020H,020H,020
125 A060 20202020          DEFB 020H,020H,020H,020
125 A064 00
126                ;
127                ;
128                ; FLAG D'ETAT DE L'AFFICHAGE 'CAP'
129                ; *****
130                ;
131 A065 00          CAPFLA:          DEFB 00
132                ;
133                ;
134                ; COMPTEUR DU TEMPS DE REAFFICHAGE
135                ; *****
136                ;
137 A066 00          COMFLA:          DEFB 00H
138                ;
139                ;
140                ; FIN -> RETOUR A LA SUITE DE LA
141                ; ROUTINE D'IT SITUEE EN ROM
142                ; *****
143                ;
144 A067 C9          RETOUR:          RET
145                ;
146                ;
147                END                ; FIN DE LISTE ASSEMBLEUR

```

Après compilation et sauvegarde, vous pourrez utiliser ce programme à partir du Basic en frappant :

MEMORY &9FFF : LOAD "CAP.BIN" : CALL &A000

Note :

Certains assembleurs, tel l'Assembleur DEVPAC présenté Partie 4, chapitre 2.6, préconisent le caractère # en tant que préfixe des données hexadécimales, le nôtre préconisant le 0 (zéro) plus le caractère H en suffixe.

LE CHARGEUR BASIC DES CODES MACHINES

Pour ceux d'entre vous qui sont intéressés par l'utilisation de cette routine et qui ne possèdent pas d'Assembleur, nous vous donnons ci-dessous le chargeur Basic :

```

10 REM ***  INSTALLATION EN BASIC  ***
20 REM ***  DE LA ROUTINE MACHINE  ***
30 REM ***  DE TRAITEMENT CAPS LOCK ***
40 REM *****
50 FOR ADRESSE = &A000 TO &A067:REM adre
  sses de chargement
60 READ DONNEE$:REM lecture des codes ma
  chines
70 DONNEE=VAL("&"+DONNEE$):REM transform
  ation hexa
80 SOMME = SOMME + DONNEE:REM somme des
  codes machines
90 POKE ADRESSE,DONNEE:REM chargement ef
  fectif d'un code
100 NEXT:REM code suivant
110 READ CONTROLE:REM lecture de la somm
  e de controle
120 IF CONTROLE = SOMME THEN 140:REM ver
  ification
130 MODE 2:PRINT "ERREUR DANS LES LIGNES
  DE DATAS":LIST:REM erreur
140 MODE 2:PRINT "SAUVEGARDE PAR ":REM
  chargement ok
150 PRINT "SAVE "+CHR$(34)+"CAP.BIN"+CHR
  $(34)+",B,&A000,&A068-&A000":REM recomme
  ndations d'utilisation
160 PRINT:PRINT
170 PRINT"UTILISATION PAR : "
180 PRINT"MEMORY &9FFF : LOAD "+CHR$(34)
  +"CAP.BIN"+CHR$(34)+",&A000 : CALL &A000
  "
190 REM *****
200 REM *** codes operations a charger *
  **
210 DATA F3,21,09,A0,22,51,B9,FB
220 DATA C9,F3,21,B1,00,E5,3A,32
230 DATA B6,FE,00,28,18,3A,66,A0
240 DATA 3D,32,66,A0
250 DATA FE,01,20,0B,3E,01,32,65
260 DATA A0,11,53,A0,CD,41,A0,18
270 DATA 3A,3A,65,A0,FE,00,28,0B
280 DATA 3E,00,32,65,A0,11,5C,A0
290 DATA CD,41,A0,18,26,21,00,47
300 DATA 1A,FE,00,C8,E5,D5,CD,D3

```

```
310 DATA BD,D1,E1,24,13,18,F1,8F
320 DATA 8F,43,41,50,53,8F,8F,00
330 DATA 20,20,20,20,20,20,20,20
340 DATA 00,00,00,C9
350 DATA 10091
360 REM *****
```

Les contenus des lignes de DATAs correspondent aux codes hexadécimaux résultant de la compilation.

Nous vous conseillons, avant toute utilisation de ce programme de le sauvegarder sur disquette.

Lors du lancement, si une erreur de frappe des DATAs a été commise, le programme vous redonne le listing complet pour corriger.

Suite au succès de chargement des codes machines, le programme vous indique la méthode de sauvegarde de la routine ainsi créée (l'instruction **MEMORY &9FFF** sert à protéger le sous-programme d'un éventuel débordement d'un programme ou des variables Basic sur cette zone d'adresses).

9/8.12

Protection écran (Screen saver)

Votre frénétique activité de programmeur est parfois interrompue par une sonnerie de téléphone inopportune, ou la sonnette de la porte d'entrée.

Ce dérangement peut parfois se prolonger plus longtemps que vous ne l'auriez désiré, et vous faire oublier votre ordinateur.

Hors, laisser une page écran fixe sur la visu n'est pas sans conséquence dommageable à long terme sur la durée de vie de votre écran vidéo.

Ce problème peut apparaître aussi chez nos lecteurs passionnés de calculs mathématiques ou astronomiques, longs, qui laissent l'écran du CPC allumé sur une page fixe durant l'attente du résultat.

La solution idéale serait de couper automatiquement l'alimentation du moniteur vidéo au bout de quelques minutes sans frappe sur le clavier. Cette solution s'avère compliquée et probablement coûteuse, sans compter les problèmes de fiabilité, de matériel électronique, d'approvisionnement des composants, et le risque d'électrocution dû à la manipulation près des organes électroniques de la THT (Très Haute Tension) appliquée sur le tube vidéo. La perte de la garantie est aussi à craindre si votre matériel est récent.

La solution la moins dangereuse et la plus économique est une solution logicielle telle que celle que nous vous proposons.

A défaut d'éteindre complètement l'écran, nous allons inhiber l'affichage des caractères sur celui-ci. Cette solution est acceptable, dans la mesure où ce sont les pixels allumés qui risquent de brûler le tube cathodique.

Nous aurions pu provoquer un CLS de l'écran, mais il faut que celui-ci restitue intégralement son contenu dès la frappe d'une touche, lors du retour de l'utilisateur.

Nos problèmes sont donc les suivants :

- reconnaître la frappe ou non d'une touche sur le clavier ;
- inhiber l'affichage sur la visu ;
- effectuer ces différentes opérations sans perturber un programme en cours ou les manipulations éventuelles de l'utilisateur (frappe d'un listing, par exemple),

— restituer l'affichage lors de la frappe d'une touche.

La résolution de ces problèmes vous fera, tout au long de ce chapitre, découvrir de nouvelles possibilités, qui vous permettront de sophistication vos propres programmes.

I. Reconnaître une touche frappée

Afin de reconnaître si une touche a été frappée par l'utilisateur, nous pouvons, dans un programme Basic, utiliser l'instruction :

A\$ = INKEY\$

mais, lors de l'utilisation en mode commande (ou direct) de l'AMSTRAD, cette instruction n'a aucun effet.

Penchons-nous donc sur le fonctionnement du CPC en mode commande.

ADRESSES DE GESTIONS DES TOUCHES FRAPPÉES

Il faut savoir que toutes les 3,3 millisecondes (tous les 1/300^e de seconde exactement), le CPC est interrompu afin d'effectuer des routines essentielles à son fonctionnement (ré-actualisation de la variable TIME, gestion de la table d'événements, actualisation du compteur de gestion clavier, ...) et, entre autres, la scrutation du clavier (grâce au compteur précédemment nommé) tous les 1/50^e de seconde (toutes les 20 ms).

La scrutation du clavier remet à jour une zone de mémoire située, pour les CPC 664 et CPC 6128, entre les adresses &B63F et &B648 ainsi qu'entre les adresses &B649 et &B652.

Ces deux zones d'adresses sont en fait complémentaires au sens binaire du terme : c'est-à-dire que le contenu de l'adresse &B63F est le complément binaire du contenu de l'adresse &B649, le contenu de l'adresse &B640 le complément binaire du contenu de &B64A, ... etc.

Initialement, quand aucune touche n'est enfoncée, le contenu des adresses &B63F à &B648 est fixé à &FF (255 décimal), donc celui des adresses &B649 à &B652 fixé à 00.

Nous vous proposons ci-après un programme permettant d'explorer plus en détail le contenu de ces adresses.

PROGRAMME EXPÉRIMENTAL

```
10 MODE 2
20 PRINT "ADRESSE", "HEXA", "DECIMAL", "BINAIRE"
30 FOR ADRESSE = &B63F TO &B648
40 PRINT HEX$(ADRESSE);SPACE$(3);"";HEX$(PEEK(ADRESSE));SPACE$(3);PEEK(ADRESSE);SPACE$(4);USING"#####";VAL(BIN$(PEEK(ADRESSE)))
50 NEXT ADRESSE
```

```

60 PRINT STRING$(40," - "):REM separation
70 PRINT "ADRESSE", "HEXA", "DECIMAL", "BINAIRE"
80 FOR ADRESSE = &B649 TO &B652
90 PRINT HEX$(ADRESSE);SPACE$(3);";";HEX$(PEEK(ADRESSE));SPACE$(3);PEEK(ADRESSE);SPACE$(4);USING"#####";VAL(BIN$(PEEK(ADRESSE)))
100 NEXT ADRESSE
110 PRINT CHR$(30):REM retour en haut de l'écran
120 GOTO 20
130 END

```

Ce petit programme permet de lire et d'afficher en permanence le contenu des adresses précédemment citées.

Vous pourrez ainsi appuyer sur une touche de votre choix et visualiser la modification du contenu d'une adresse spécifique. L'appui sur une touche devra être suffisamment long pour permettre au Basic d'afficher les contenus.

Vous trouverez, par exemple, les valeurs suivantes pour les touches :

- DEL : &60 à l'adresse &B652, &7F à l'adresse &B648
- f6 : &10 à l'adresse &B649, &EF à l'adresse &B63F
- ESPACE : &80 à l'adresse &B64E, &7F à l'adresse &B644.

II. Inhibition de l'affichage écran

LE CONTROLEUR VIDÉO

Pour modifier l'affichage sur l'écran, il nous faut connaître comment sont organisées les entrailles du CPC, et notamment nous renseigner sur la gestion de l'écran.

Le contrôle de l'affichage sur l'Amstrad a été dédié au VGA (*Vidéo Gate Array*, composant spécifique conçu par AMSTRAD), et surtout au composant 6845 conçu par MOTOROLA initialement pour la famille 68XX, mais adaptable facilement à d'autres microprocesseurs tel le Z80. Nous vous conseillons de vous reporter avec attention à la description qui en est faite à la Partie 2, chapitre 3.2.

Ce composant contient 19 registres (le registre AR et les 18 registres R0 à R17), auxquels on accède en écriture par deux adresses de ports (au lieu de 19). Cette astuce est possible grâce au registre AR, placé à une adresse de port, qui sert de pointeur de registre, dans lequel on place d'abord le numéro du registre de gestion vidéo à atteindre, l'autre adresse du port étant celle du registre pointé.

Les constructeurs de l'AMSTRAD ont placé le composant 6845 aux adresses &BCXX (pour le registre AR) et &BDXX (pour écrire dans le registre dont le numéro est pointé par le contenu de AR). Chacun des X des adresses peut être n'importe quelle valeur hexadécimale comprise entre &0 et &F, ceci est dû à un décodage partiel des composants sur les CPCs.

Pour simplifier notre étude, nous prendrons l'adresse &BC00 pour le registre AR et l'adresse &BD00 pour le registre pointé par le contenu de AR. L'adressage d'un port s'effectue différemment de l'adressage d'une case mémoire. Par exemple en Basic, on écrit dans la case mémoire en RAM &adresse grâce à l'instruction :

POKE &adresse,&valeur

et on lit par :

A = PEEK(&adresse)

Remarque :

Le caractère & indique que la valeur est en hexadécimal, mais elle peut aussi être décimale, on enlève cette fois-ci le caractère &.

Par contre pour lire ou écrire sur un port, on utilisera les instructions **INP** (pour lire) et **OUT** (pour écrire) :

OUT &adresse,&valeur et A = INP (&adresse)

Nous vous proposons d'expérimenter le programme ci-dessous qui accède au registre de largeur de synchronisation horizontale :

```

10  MODE 1
20  PRINT "SI VOUS APPUYEZ SUR UNE TOUCHE"
30  PRINT "JE DETRUIS VOTRE AMSTRAD"
60  CLEAR INPUT
70  A$ = INKEYS
80  IF A$ = " " THEN 70
90  OUT &BC00,3:REM accès au registre
100 OUT &BD00,1:REM modification de la synchronisation
110 END

```

Suivez le conseil, et, pour tout remettre dans l'ordre, frappez en aveugle la commande :

OUT &BD00,&8E

Vous pouvez ainsi essayer, en vous aidant des renseignements fournis Partie 2, chapitre 3.2, différentes combinaisons sur les registres du contrôleur Vidéo. Rassurez-vous sur les effets produits, ils ne peuvent en aucun cas endommager votre AMSTRAD. Si vous vous retrouvez bloqué suite à certaines commandes, un RESET par l'appui simultané sur <SHIFT> <CONTROL> <ESC> vous redonnera la configuration initiale.

L'INITIALISATION DU CONTROLEUR VIDÉO

Lors de l'initialisation, l'AMSTRAD effectue un saut à l'adresse 0000. A cette adresse est effectuée une commutation sur la ROM du système d'exploitation et un saut à l'adresse &0591H. A partir de là, sont effectuées les différentes initialisations des composants de l'unité centrale, et notamment à partir de l'adresse &05B1H l'initialisation du contrôleur Vidéo.

Nous vous donnons ci-dessous un désassemblage commenté de cette initialisation :

```

05B1    21E505    LD HL,05E5; pointeur fin table CRTC
05B4    010FBC    LD BC,BC0F; adresse registre (BCXX)
05B7    ED49      OUT(C),C ; adresse registre dans AR
05B9    2B        DEC HL ; pointe valeur registre suivant
05BA    7E        LD A,(HL) ; charge valeur
05BB    04        INC B ; position BDXX
05BC    ED79      OUT(C),A ; charge registre
05BE    05        DEC B ; position BCXX (reg. AR)
05BF    0D        DEC C ; position registre suivant
05C0    F2B705    JP P,05B7 ; saut si pas dernier registre traité
05C3    1820      JR 05E5 ; saut pour suite init.

```

...
;**** TABLE CRTC ****

```

05D5    3F        ; fréquence de synchro horizontale → R0
05D6    28        ; nombre de caractères par ligne → R1
05D7    2E        ; délai de synchro horizontale → R2
05D8    8E        ; largeur de synchronisation horizontale → R3
05D9    1F        ; nombre de lignes par écran → R4
05DA    06        ; ajustage du nombre de lignes → R5
05DB    19        ; nombre de lignes affichées → R6
05DC    1B        ; synchronisation verticale → R7
05DD    00        ; mode → R8
05DE    07        ; nombre de lignes par caractère → R9
05DF    00        ; début de curseur → R10
05E0    00        ; fin de curseur → R11
05E1    30        ; poids fort adresse RAM écran
05E2    30        ; poids faible adresse RAM écran
05E3    C0        ; poids fort adresse position curseur
05E4    00        ; poids faible adresse position curseur

```

INHIBITION DE LA VISUALISATION DES CARACTÈRES

Il nous faut maintenant déterminer comment inhiber la visualisation de l'affichage sur la visu sans modifier le contenu de la mémoire écran.

Le registre du CRTC 6845 qui nous intéresse est le registre R1 contrôlant le nombre de caractères affichés par ligne. Celui-ci est, à la mise sous tension, chargé avec la valeur 40 (&28 hexadécimal), et ce quel que soit le mode de travail choisi pour l'écran (20, 40 ou 80 colonnes).

Si nous nous amusons sous Basic à modifier le nombre contenu par ce registre, nous modifierons ainsi le nombre de caractères visualisables, la commande sera :

```

OUT &BC00,1
OUT &BD00,nb. caractères (inférieur à 40)

```

Le petit programme suivant vous permettra d'effectuer une petite animation agréable, qui peut être utilisée lors de modification d'une page d'un menu, par exemple :

```

10 REM AFFICHAGE PAGE ECRAN 1
20 REM ATTENTE CHOIX SUR MENU
30 GOSUB 60000:REM ANIMATION NON VISUALISATION PAGE
40 REM TRAITEMENT DU CHOIX ET AFFICHAGE NOUVELLE PAGE
50 GOSUB 60060:REM ANIMATION VISUALISATION PAGE
60 REM SUITE...

.....
60000 REM ANIMATION NON VISUALISATION
60010 OUT &BC00,1
60020 FOR J = 40 TO 0 STEP -1
60030 OUT &BD00,J
60040 NEXT J
60050 RETURN
60060 REM ANIMATION VISUALISATION
60070 OUT &BC00,1
60080 FOR J = 0 TO 40
60090 OUT %BD00,J
60100 NEXT J
60110 RETURN

```

Vous pouvez visualiser cette animation en mode direct par les commandes :

GOSUB 60000 : GOSUB 60060 <RETURN>

On remarquera, lors de cette animation, que les caractères supplémentaires de la ligne originale sont reportés sur la ligne suivante ; de plus en mode 0, c'est un demi-caractère qui est reporté, tandis qu'en mode 1, deux caractères sont reportés ; la ligne suivante commençant avec le nombre de caractères reportés.

Pour résoudre notre problème, il nous suffira d'inhiber l'écran à l'aide de l'instruction :

```

OUT &BC00,1
OUT &BD00,0

```

et de le valider à nouveau, lors de la frappe sur une touche par :

```

OUT &BC00,1
OUT &BD00,40

```

III. Interruption du fonctionnement pour traitement

Dans ce paragraphe, nous allons étudier la façon de détecter l'appui sur une touche afin de protéger notre visu.

Cette gestion ne doit tout d'abord perturber aucun programme Basic en cours, nous pourrions utiliser pour cela la gestion sous interruption logicielle telle que décrite assez précisément dans le chapitre concernant l'affichage d'un CAPS LOCK interactif (voir Partie 9, chapitre 8. 11).

INTERRUPTION LOGICIELLE

Pour ceux qui désireraient essayer ce traitement, nous vous donnons ci-après l'algorithme permettant d'écrire le programme à insérer. La lecture du chapitre précédemment cité vous y aidera aussi.

- DEBUT
 - Initialiser une variable de comptage
 - Installer le chronomètre d'interruption logicielle
- FIN
- DEBUT
 - CO
 - routine de traitement à saisir en fin de programme
 - FINCO
 - Initialiser une variable témoin à zéro
 - Initialiser la variable début de tableau « touche frappée » à &B63F
 - REPETER
 - Lire le contenu de la variable « touche frappée »
 - CO
 - par l'instruction **PEEK**
 - FINCO
 - SI le contenu est différent de &FF (=255)
 - ALORS
 - Placer 1 dans la variable témoin
 - FINSI
 - Incrémenter la variable touche frappée
 - JUSQU'A ce que le contenu de la variable touche frappée soit supérieur à la dernière adresse du tableau touche frappée
 - SI le contenu de la variable témoin est différent de 1
 - ALORS
 - Incrémenter le contenu de la variable de comptage
 - SI le contenu de la variable de comptage est supérieur à une valeur déterminée Y
 - CO
 - Y sera calculée par la formule :
Temps d'attente = Y x Délai x 50 ms
 - FINCO
 - ALORS
 - Inhiber l'affichage
 - FINSI
 - SINON
 - Réinitialiser la variable de comptage
 - Autoriser l'affichage
 - FINSI
 - FIN

Ce type d'interruption ne fonctionne cependant que lors de l'exécution d'un programme Basic, ce qui n'est pas des plus intéressants, surtout lorsque c'est durant la frappe de votre listing que vous êtes interrompu !

Nous devons donc penser à un autre type d'interruption, qui est exécutée quelle que soit la façon dont vous faites travailler votre AMSTRAD (en mode direct ou en mode programme Basic).

INTERRUPTION MATÉRIELLE

Nous allons donc utiliser une interruption qui est générée toutes les 3.3 millisecondes par le composant GATE ARRAY (on se reportera au paragraphe concernant les interruptions logicielles dans le chapitre traitant de la touche CAPS LOCK, Partie 9, chapitre 8.11).

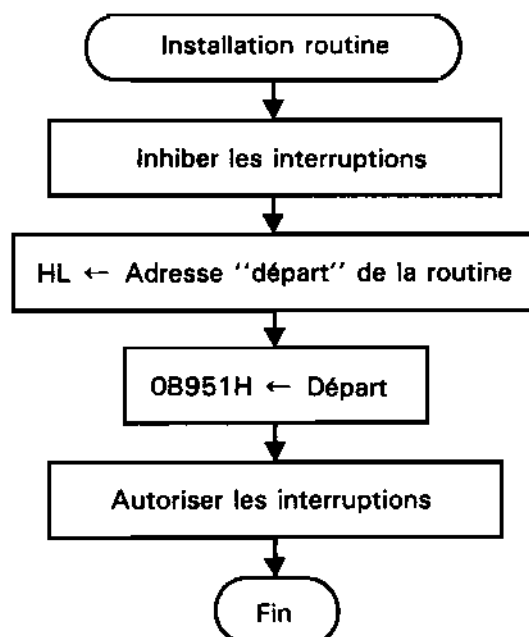
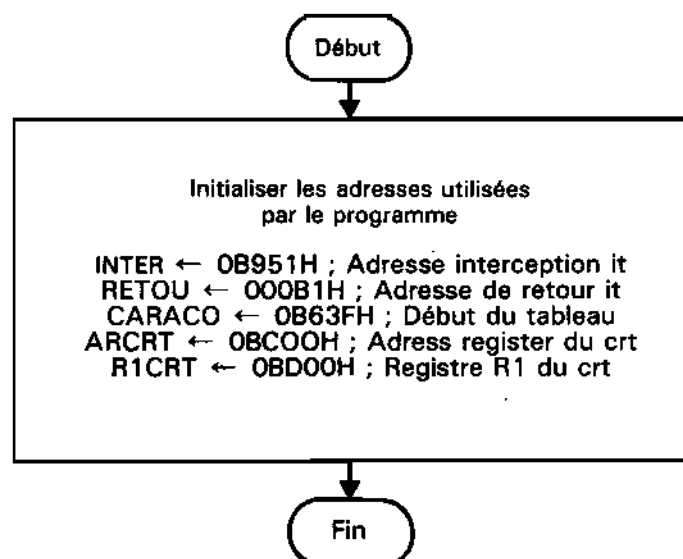
Rappelons que cette interruption est placée sur la broche $\overline{\text{IRQ}}$ du microprocesseur Z80, et qu'elle l'oblige à effectuer une instruction dénommée **RST 8**, qui effectue un branchement à l'adresse &B941 pour les CPC 664 et CPC 6128 (à l'adresse &B939 pour le CPC 464).

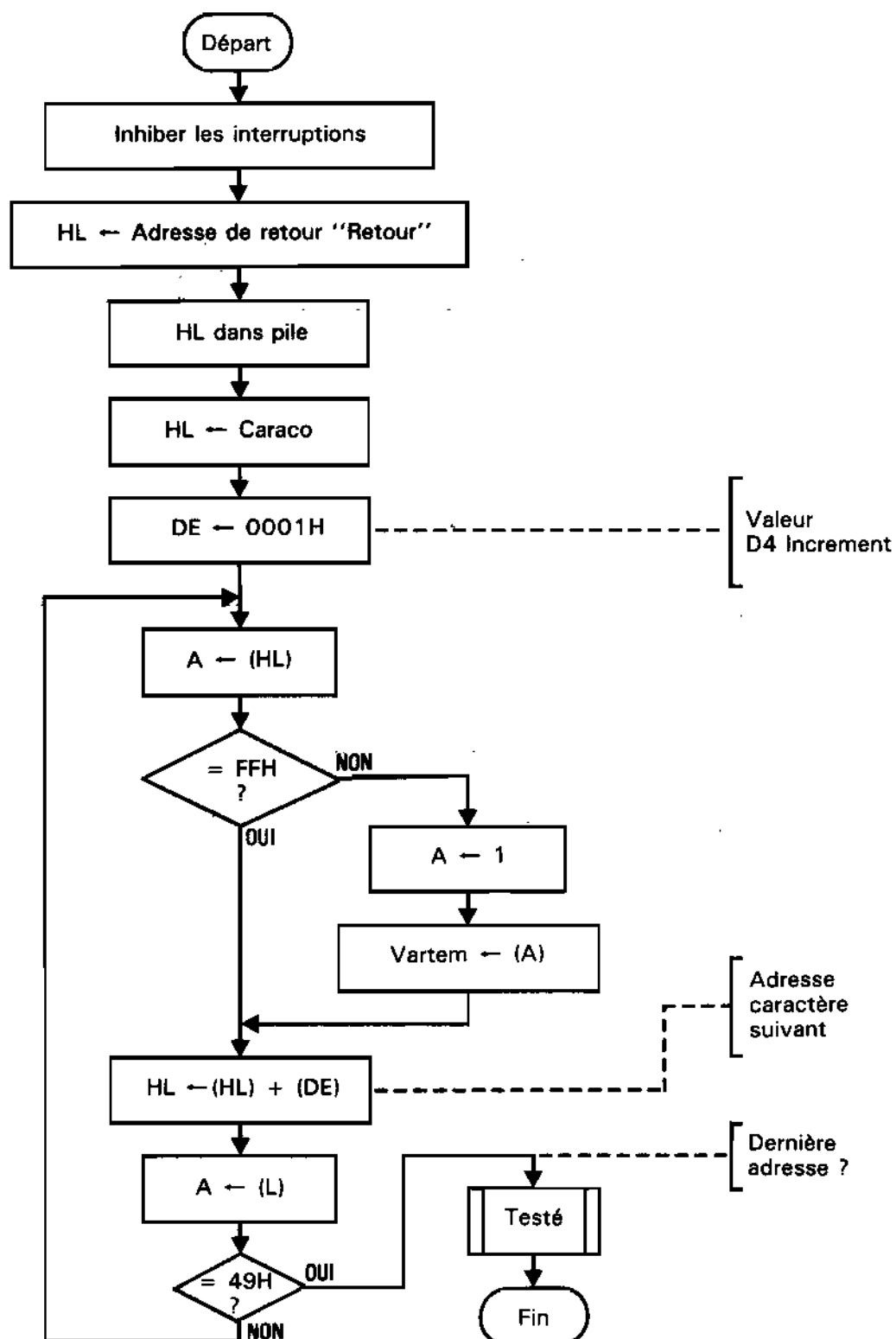
Nous pouvons détourner cette routine en plaçant une adresse différente de &00B1 aux adresses &B951 et &B952 (&B949 et &B94A pour le CPC 464).

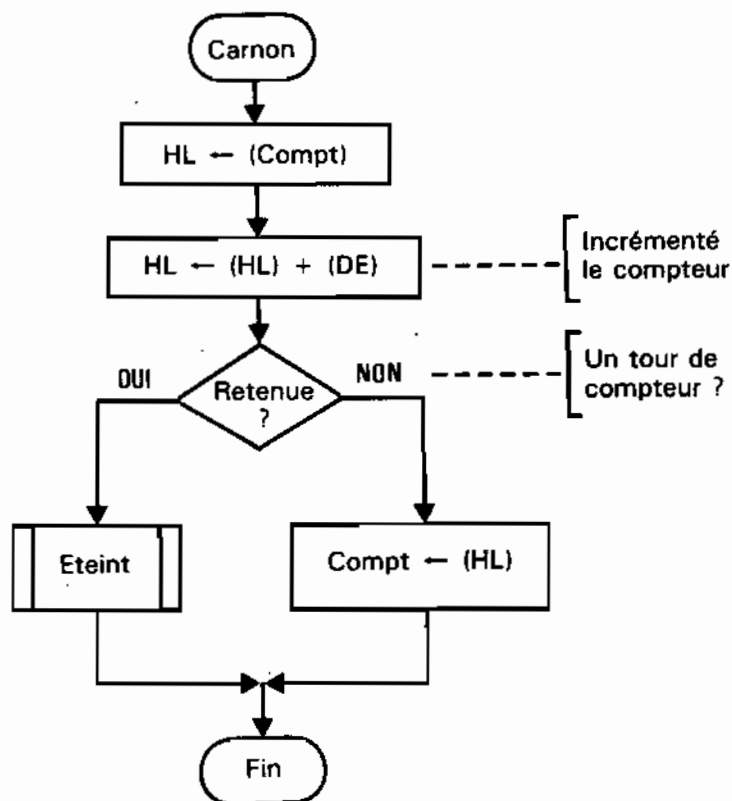
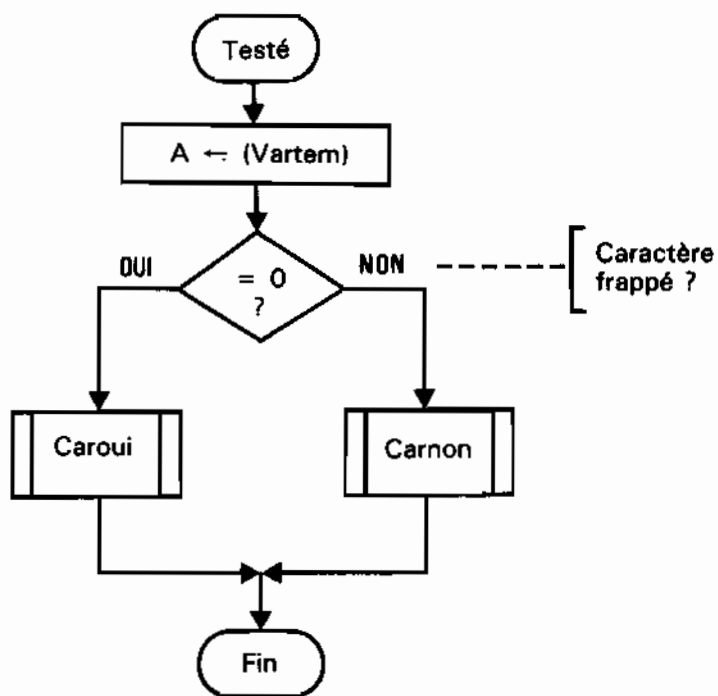
Suite à ce détournement, il nous faudra bien sûr retourner à l'adresse &00B1, faute de quoi l'AMSTRAD n'y retrouverait plus ses petits, au mieux effectuerait un splendide RESET.

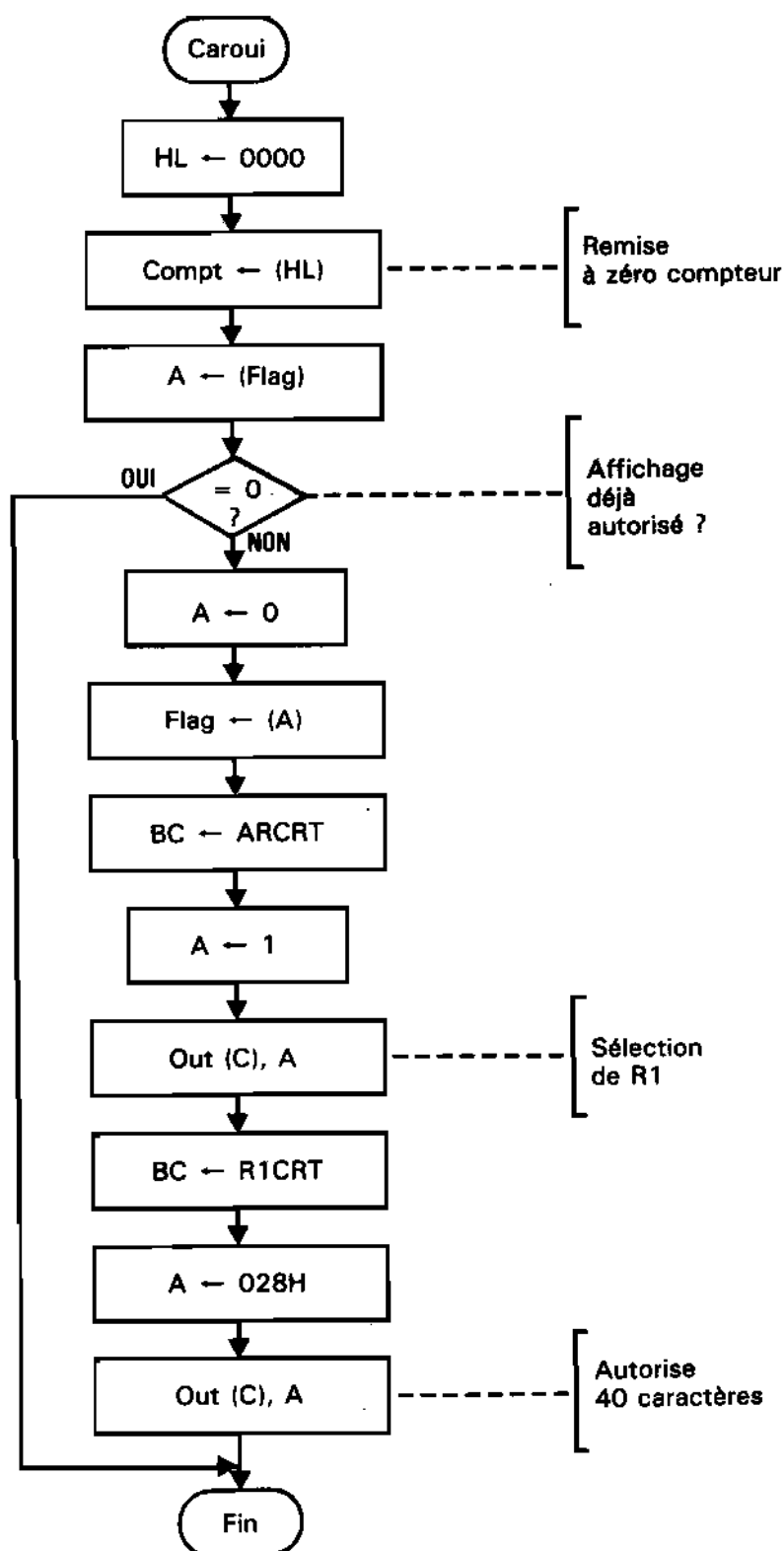
L'ORDINOGRAMME

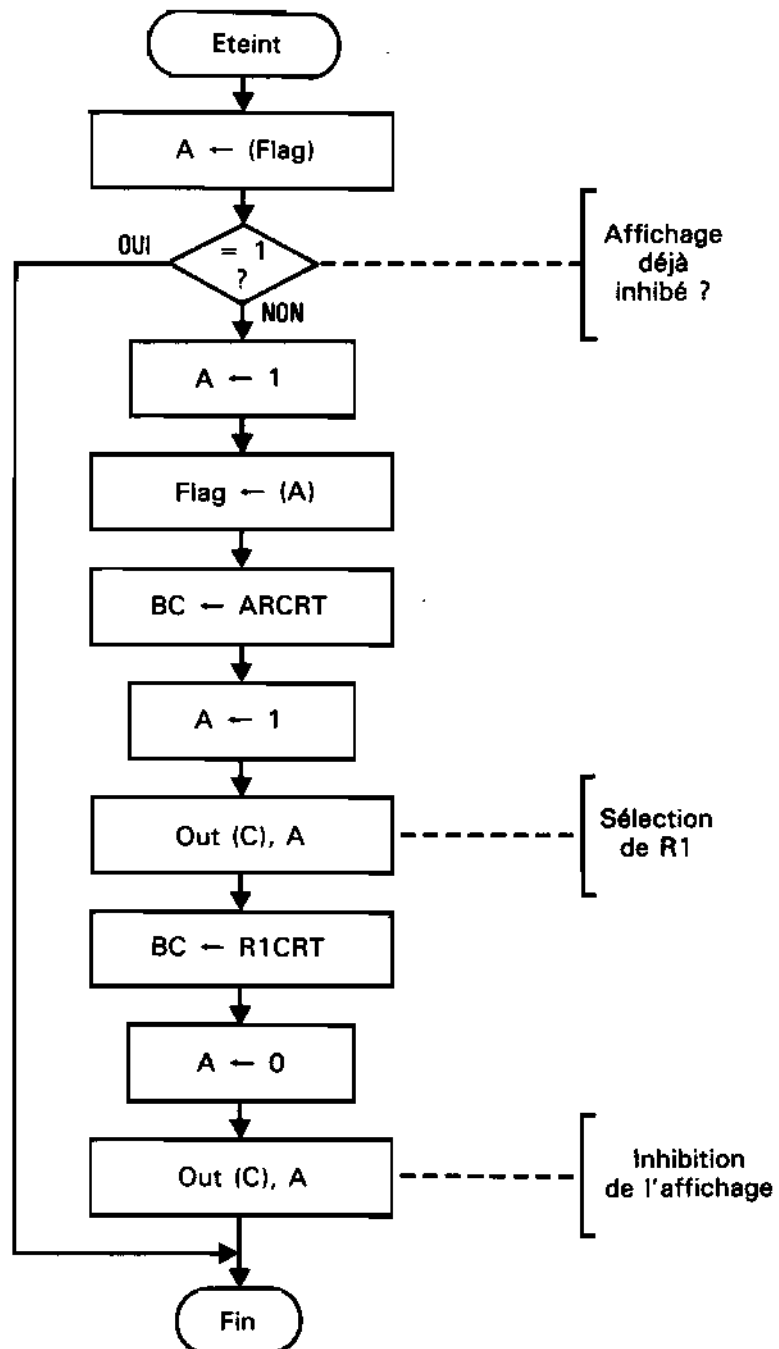
En tenant compte de ces précisions, nous pouvons adapter l'algorithme précédent au langage machine pour proposer l'ordinogramme suivant :











LE PROGRAMME ASSEMBLEUR

Grace à l'ordinogramme précédent, nous allons déduire le programme Assembleur Z80 suivant.

Ce programme devra peut-être subir quelques modifications dans la notation des valeurs hexadécimales (certains Assembleurs utilisent la notation # en tant que préfixe, les autres utilisent le 0 (zéro) en préfixe et le H en suffixe) — dans tous les cas, reportez-vous à la notice fournie avec celui-ci.

```

1      ;*** PROGRAMME ASSEMBLEUR ***
2      ;*** DE PROTECTION ECRAN ***
3      ;*** PAR INHIBITION DE ***
4      ;*** DE L'AFFICHAGE ***
5      ;*****
6      ;
7      ;
8      ;DEFINITION DES ADRESSES UTILES
9      ;*****
10     ;
11     ;ADRESSE DETOURNEMENT INTERRUPTION
12     INTER:      EQU 0B951H
13     ;
14     ;ADRESSE DE RETOUR D'INTERRUPTION
15     RETOU:      EQU 000B1H
16     ;
17     ;ADRESSE DEBUT TABLEAU TOUC.FRAP.
18     CARACO:     EQU 0B63FH
19     ;
20     ;REGISTER ADRESS DU CRT
21     ARCRT:      EQU 0B000H
22     ;
23     ;REGISTRE R1 DU CRT
24     RICRT:      EQU 0B000H
25     ;
26     ;*****
27     ;
28     ;DIRECTIVE DE L'ORIGINE
29     ;D'ASSEMBLAGE
30     ;
31     ORG 0A000H
32     LOAD 0A000H
33     ;
34     ;*****
35     ;
36     ;
37     ;ROUTINE D'INSTALLATION POUR
38     ;DETOURNEMENT DE LA ROUTINE
39     ;DE TRAITEMENT D'INTERRUPTION
40     ;*****
41     ;
42     A000 F3      DI                      ;INHIBITION INTERRUPTION
43     A001 2109A0   LD HL,DEPART          ;ADRESSE DEPART ROUTIN
44     A004 2251B9   LD (INTER),HL        ;A LA PLACE DE INTER
45     A007 FB      EI
46     A008 C9      RET                    ;FIN D'INSTALLATION

```



```

47 ;*****
48 ;
49 ;
;ROM E DEPART:;PREPARATION AU RETOUR EN
51 ;ROM EN FIN DE TRAITEMENT
52 A009 F3 DI ;INHIBITION INTERRUPTION
53 A00A 21B100 LD HL,RETOU ;ADRESSE DE RETOUR
54 A00D E5 PUSH HL ;DANS LA PILE
55 ;
X0B6A DEBUT:
57 A00E AF XOR A ;PLACE 00H
58 A00F 3278A0 LD (VARTEM),A ;DANS VARIABLE TEST
59 A012 213FB6 LD HL,CARACO ;POINTE PREMIERE
60 ;ADRESSE DU TABLEAU
61 A015 110100 LD DE,000001H ;VALEUR D'INCREMENT
62 ;
63 A018 7E LOOP1: LD A,(HL) ;CHARGE POUR TEST
64 A019 F1FF CP OFFH ;DU CONTENU ADRESSE
65 A01B 2805 JR Z,LOOP2 ;TOUCHE FRAPPEE ?
66 A01D 3E01 LD A,01H
67 A01F 3278A0 LD (VARTEM),A
68 A022 19 LOOP2: ADD HL,DE ;INCREMENTE ADRESSE
69 A023 7D LD A,L
70 A024 FE49 CP 049H ;DERNIERE ADRESSE?
71 A026 20F0 JR NZ,LOOP1 ;SAUT SI NON
72 ;
73 A028 3A78A0 TESTE: LD A,(VARTEM) ;CHARGE A
74 A02B FE00 CP 00H ;AVEC VARIABLE INDIQUANT
75 A02D 2822 JR Z,CARNON ;TOUCHE FRAPPEE
76 ;
77 A02F 210000 CARQUI: LD HL,0000 ;RE-INITIALISE
78 A032 227AA0 LD (COMPT),HL ;LE COMPTEUR
79 A035 3A79A0 LD A,(FLAG) ;CHARGE A AVEC LE
80 A038 FE00 CP 00H ;FLAG POUR TESTER SI
81 A03A 2840 JR Z,FIN ;L'ECRAN EST DEJA ETEINT
82 A03C 3E00 LD A,00H ;SINON
83 A03E 3279A0 LD (FLAG),A ;POSITIONNER LE FLAG
84 A041 0103BC LD BC,ARCRT ;PREPARER PORT ARCRT
85 A044 3E01 LD A,01H ;POUR
86 A046 ED79 OUT (C),A ;SELECTIONNER RICRT
87 A048 0100BD LD BC,RICRT ;PREPARER PORT RICRT
88 A04B 3E28 LD A,028H ;POUR
89 A04D ED79 OUT (C),A ;AUTORISER AFFICHAGE
90 A04F 182B JR FIN ;SAUT A FIN
91 ;
92 A051 2A7AA0 CARNON: LD HL,(COMPT) ;CHARGER
93 A054 19 ADD HL,DE ;COMPTEUR POUR INCREMENTE
94 A055 3805 JR C,ETEINT ;SAUT SI UN TOUR
95 A057 227AA0 LD (COMPT),HL ;SINON SAUVER COMPTEU
96 A05A 1820 JR FIN ;SAUT A FIN
97 ;
98 A05C 3A79A0 ETEINT: LD A,(FLAG) ;CHARGE FLAG
99 A05F FE01 CP 01H ;POUR
100 A061 2819 JR Z,FIN ;POUR TEST SI DEJA ETEINT
101 A063 3E01 LD A,01H ;SINON SIGNALER EXTINCTIO
102 A065 3279A0 LD (FLAG),A ;DANS LE FLAG
103 A068 0100BC LD BC,ARCRT ;PREPARER PORT ARCRT
104 A06B 3E01 LD A,01H ;POUR

```

```

105 A06D ED79      OUT  (C),A          ;SELECTIONNER R1CRT
106 A06F 0100BD    LD   BC,R1CRT      ;PREPARER PORT R1CRT
107 A072 3E00      LD   A,00H         ;POUR
108 A074 ED79      OUT  (C),A          ;INHIBER L'AFFICHAGE
109 A076 1B04      JR   FIN           ;SAUT A FIN
110                ;
111                ;
112                ;ADRESSES VARIABLES UTILISEE
113                ;*****
114                ;
115 A078 00      VARTEM:  DEFB 00H
116 A079 00      FLAG:   DEFB 00H
117 A07A 0000    COMPT:  DEFB 00H,00H
118                ;*****
119                ;
120                ;
121 A07C C9      FIN:     RET            ;FIN DE TRAITEMENT
122                ;
123                END

```

Après compilation et sauvegarde, vous pourrez utiliser ce programme à partir du Basic en frappant :

MEMORY &9FFF : LOAD « EFFECR.BIN » : CALL &A000

si vous avez dénommé le programme sauvegarde : EFFECR.BIN

LE CHARGEUR BASIC DES CODES MACHINES

Pour ceux d'entre vous qui sont intéressés par l'utilisation de cette routine et qui ne possèdent pas d'Assembleur, nous vous donnons ci-dessous le chargeur Basic.

```

10 REM ***  INSTALLATION EN BASIC  ***
20 REM ***  DE LA ROUTINE MACHINE  ***
30 REM ***  DE TRAITEMENT CAPS LOCK ***
40 REM *****
50 FOR ADRESSE = &A000 TO &A07C:REM adre
sses de chargement
60 READ DONNEE$:REM .lecture des codes ma
chines
70 DONNEE=VAL("&"+DONNEE$):REM transform
ation hexa
80 SOMME = SOMME + DONNEE:REM somme des
codes machines
90 POKE ADRESSE,DONNEE:REM chargement ef
fectif d'un code

```

```

100 NEXT:REM code suivant
110 READ CONTROLE:REM lecture de la somm
e de controle
120 IF CONTROLE = SOMME THEN 140:REM ver
ification
130 MODE 2:PRINT "ERREUR DANS LES LIGNES
DE DATAs":LIST:REM erreur
140 MODE 2:PRINT "SAUVEGARDE PAR :":REM
chargement ok
150 PRINT "SAVE "+CHR$(34)+"EFFECCR.BIN"+
CHR$(34)+",B,&A000,&A07C-&A000":REM reco
mmendations d'utilisation
160 PRINT:PRINT
170 PRINT"UTILISATION PAR : "
180 PRINT"MEMORY &9FFF : LOAD "+CHR$(34)
+"EFFECCR.BIN"+CHR$(34)+",&A000 : CALL &A
000"
190 REM *****
200 REM *** codes operations a charger *
**
210 DATA F3,21,09,A0,22,51,B9,FB
220 DATA C9,F3,21,B1,00,E5,AF,32
230 DATA 78,A0,21,3F,B6,11,01,00
240 DATA 7E,FE,FF,28,05,3E,01,32
250 DATA 78,A0,19,7D,FE,49,20,F0
260 DATA 3A,78,A0,FE,00,28,22,21
270 DATA 00,00,22,7A,A0,3A,79,A0
280 DATA FE,00,28,40,3E,00,32,79
290 DATA A0,01,00,BC,3E,01,ED,79
300 DATA 01,00,BD,3E,28,ED,79,18
310 DATA 2B,2A,7A,A0,19,38,05,22
320 DATA 7A,A0,18,20,3A,79,A0,FE
330 DATA 01,28,19,3E,01,32,79,A0
340 DATA 01,00,BC,3E,01,ED,79,01
350 DATA 00,BD,3E,00,ED,79,18,04
354 DATA 00,00,00,00,C9
355 DATA 11218
360 REM *****

```

Les contenus des lignes de DATAs correspondent aux codes hexadé-
cimaux résultant de la compilation.

Nous vous conseillons, avant toute utilisation de ce programme, de le
sauvegarder sur disquette.

Lors du lancement, si une erreur de frappe des DATAs a été commise,
le programme vous redonne le listing complet pour corriger.

Suite au succès de chargement des codes machines, le programme vous indique la méthode de sauvegarde de la routine ainsi créée (l'instruction **MEMORY &9FFF** sert à protéger le sous-programme d'un éventuel débordement d'un programme ou des variables Basic sur cette zone d'adresses).

Surtout, n'oubliez pas le **CALL &A000** après chargement du code machine, sinon la routine ne sera pas initialisée.

ESSAIS PRÉALABLES

La routine permet d'inhiber l'affichage sur l'écran de votre CPC au bout d'un certain temps déterminé par le compteur « COMPT ».

Ce compteur effectue un comptage complet sur 2 octets. Le temps au bout duquel l'affichage sera inhibé est calculé par la formule :

$$\begin{aligned}
 \text{Temps} &= (\text{COMPT}) \times 3.3 \text{ ms} \\
 &= (\&FFFF) \times 3.3 \text{ ms} \\
 &= (15 \times 16 \times 16 \times 16 + 15 \times 16 \times 16 + 15 \times 16 + 15) \times 3.3 \text{ ms} \\
 &= 216,3 \text{ secondes} \\
 &= 3 \text{ minutes et } 16 \text{ secondes environ}
 \end{aligned}$$

Si nous voulons, lors des essais de cette routine, ne pas attendre le temps complet, nous vous proposons le petit programme Basic suivant qui initialise le compteur à une valeur relativement élevée (&D000), et visualise sa progression.

Attention, dès que vous appuyez sur une touche, le compteur est réinitialisé à zéro.

```

10 FOR I = 1 TO 500 : REM attente fin appui sur
20 NEXT I : REM la touche <RETURN>
30 POKE &A07B,&E0 : REM octet de poids fort du compteur
40 POKE &A07A,&00 : REM octet de poids faible du compteur
50 PRINT HEX$(PEEK(&a07B)); : affiche octet fort
60 PRINT HEX$(PEEK(&A07A)); : REM affiche octet faible
70 PRINT SPACE$(3);CHR$(13) : REM retour position initial
80 GOTO 50
90 END

```

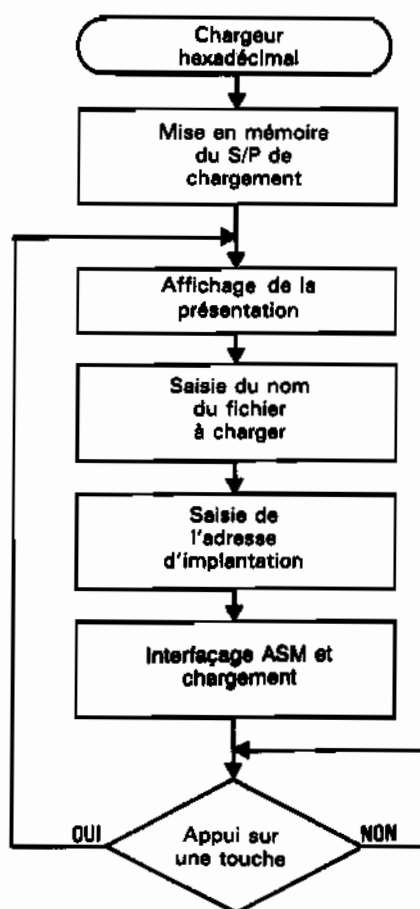
9/8.13

Chargeur hexadécimal

Ce petit utilitaire écrit en Assembleur vous permet de charger un programme ou des données à n'importe quel endroit en mémoire RAM (chose impossible sous Basic: essayez par exemple de charger des données en &H100 !! Vous verrez apparaître le message **MEMORY FULL** et le chargement ne s'effectuera pas).

Le chargeur hexadécimal présenté ici est appelé sous Basic. Il vous demande le nom du fichier à charger, puis son implantation en mémoire. Le programme (ou les données) correspondant est alors chargé.

Sa logique de programmation est la suivante :



Le listing du programme est le suivant :

```

1000 REM Chargeur de programmes ou fichiers
1010 '
1020 '-----
1030 'Programme Assembleur de chargement
1040 '
1050 FOR I=&9000 TO &9014:READ A:POKE I,A:NEXT
1060 DATA &6,&0,&21,&15,&90,&11,&0,&A6,&CD,&77,&BC,&21,&0,&C
0,&CD,&83,&BC,&CD,&7A,&BC,&C9
1070 '-----
1080 INK C,0:INK 1,10:BORDER 0:MODE 1:PEN 3:PRINT"          Ch
argeur de programmes":PEN 1
1090 LOCATE 1,10:INPUT"Nom du fichier ";N$
1100 PRINT:INPUT"Implantation memoire ";IM
1110 IF IM<0 THEN IM=IM+2^16
1120 '-----
1130 A=LEN(N$):POKE &9001,A 'Longueur du nom
1140 MSB=INT(IM/256):LSB=IM-MSB*256 'Poids Fort et Faible à
Implantation
1150 POKE &900C,LSB:POKE &900D,MSB 'à implantation du Fichie
r
1160 FOR I=0 TO A-1
1170   POKE &9015+I,ASC(MID$(N$,I+1,1)) 'Nom du prog a charg
er
1180 NEXT I
1190 '-----
1200 CALL &9000 'Chargement
1201 PRINT:PRINT"Appuyez sur une touche"
1202 A$=INKEY$:IF A$="" THEN 1202
1210 GOTO 1080 'Autre chargement
1220 END

```

- Lignes 1050 à 1060 : Chargement du programme assembleur.
- Lignes 1080 à 1110 : Présentation.
- Lignes 1130 à 1180 : Interfaçage avec l'assembleur.
- Lignes 1200 : Chargement.
- Lignes 1210 : Poursuite sur un autre chargement.

Le programme de chargement est écrit en Assembleur et fait appel à plusieurs macros du firmware :

CAS IN OPEN	→ Ouverture d'un fichier
CAS IN DIRECT	→ Lecture de données dans un fichier
CAS IN CLOSE	→ Fermeture d'un fichier

Rapportez-vous aux points d'entrées OBC77H, OBC83H et OBC7AH pour avoir plus de détails sur ces macros.

Son listing est le suivant :

```

1          ;
2          ; Chargeur HEXA
3          ; Entree interfacee par BASIC:
4          ; Lgr du nom du fichier a charger,
5          ; et a Stockage Fichier.
6          ;
7          ORG 9000H
8          LOAD 9000H
9          OPEN:      EQU 0BC77H          ; CAS IN OPEN
10         DIRECT:    EQU 0BC83H          ; CAS IN DIRECT
11         CLOSE:     EQU 0BC7AH          ; CAS IN CLOSE
12         ; -----
13 9000 0600          LD B,0              ; Longueur Nom Fichier
14 9002 211590        LD HL,AFFICH        ; a Nom Fichier
15 9005 1100A6        LD DE,0A600H        ; a Buffer 2KO
16 9008 CD77BC        CALL OPEN            ; Ouverture fichier
17         ;
18 900B 2100C0        LD HL,0C000H        ; a Stockage Fichier
19 900E CDB3BC        CALL DIRECT          ; Chargement Fichier
20         ;
21 9011 CD7ABC        CALL CLOSE          ; Fermeture Fichier
22 9014 C9            RET
23         AFFICH:    EQU $              ; a Nom Fichier
24         END

```

Le fichier est ouvert (ligne 16) en ayant pris soin de déclarer une zone tampon de 2 KOctets nécessaire au système pour le chargement (ligne 15).

Ensuite, le fichier est chargé (ligne 19) en ayant au préalable précisé son adresse d'implantation (ligne 18).

Enfin, le fichier est fermé (ligne 21).

Remarque :

Ce programme peut s'avérer très utile dans le cas où une grande partie de la mémoire est occupée (par exemple par un programme de jeu). En effet, le Basic détruit les données situées trop bas ou trop haut en mémoire. Pour pallier à ce problème, le chargeur ne doit plus être activé par un **RUN BASIC** mais par un **CALL**. Il doit être constitué d'une suite de modules identiques au programme de chargement décrit ci-dessus (un module par fichier à charger).

9/8.14

Formattage des listings

Il est toujours désagréable d'imprimer des listings sur plusieurs pages sans effectuer des sauts de page. En effet, une ligne se trouve la plupart du temps imprimée entre deux pages et les informations qui s'y trouvent sont difficiles à lire. Pour éviter ce genre de problème, nous vous proposons un programme utilitaire écrit dans les trois langages du CPC (Assembleur, Basic et Turbo-Pascal) qui vous permettra d'effectuer des sauts de page paramétrables.

Ces programmes ne fonctionnent correctement que sur des fichiers purement ASCII. Ces fichiers peuvent être des fichiers de données créés par vos propres programmes, des fichiers textes issus d'un traitement de texte, ou encore des programmes Basic qui ont été sauvegardés en ASCII grâce à l'option A (SAVE "MONPROG",A par exemple).

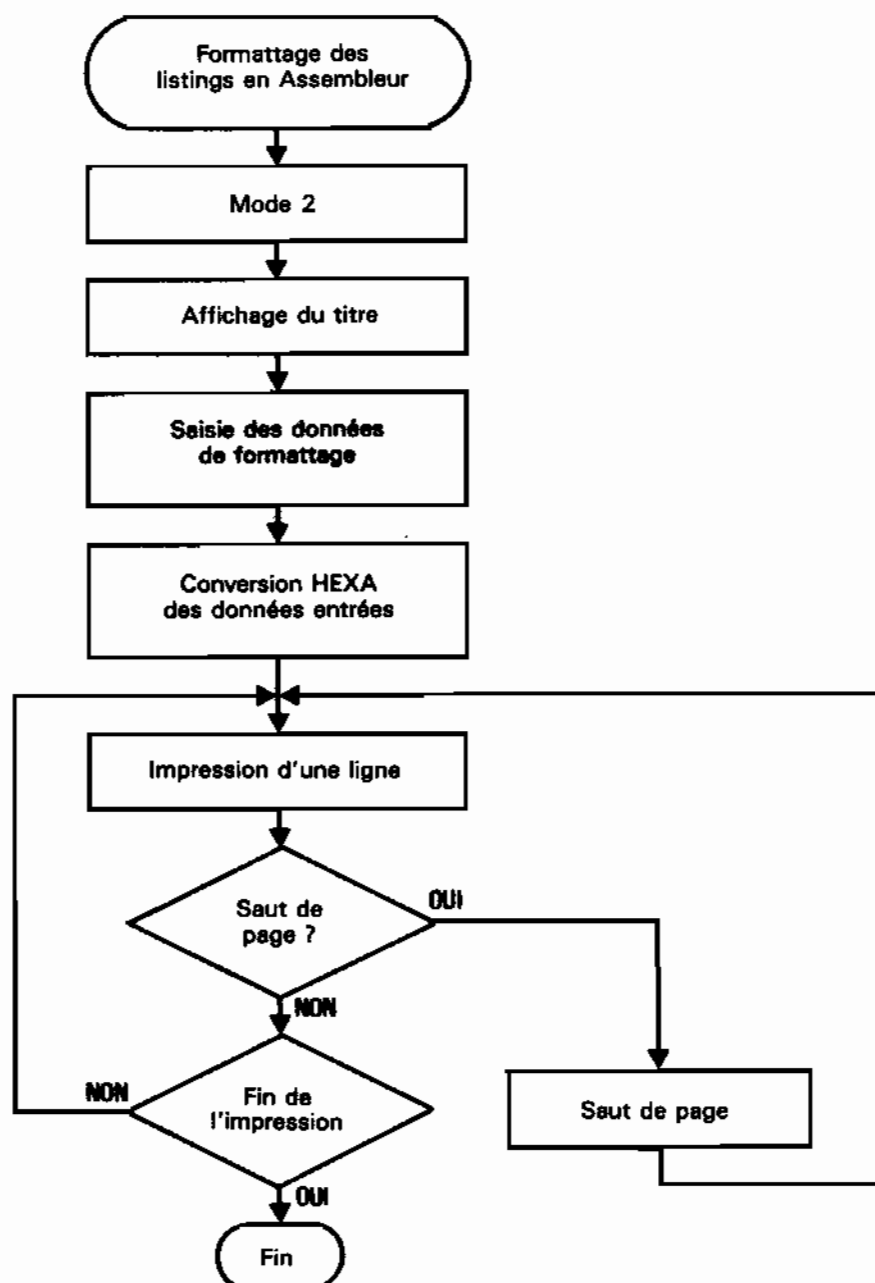
Dans le programme écrit en Assembleur, le nombre de lignes imprimées pour chaque page et le nombre de lignes total pour chaque page sont laissés au libre choix de l'utilisateur.

Dans les programmes écrits en Basic et en Turbo-Pascal, il est également possible de paramétrer la longueur des lignes imprimées.

I. Formattage des listings en Assembleur

Ce programme est de loin le plus complexe des trois, mais il représente un bon exercice de programmation pour tous ceux qui désirent apprendre l'Assembleur. De plus, il montre que les sous-programmes (affichage et saisie d'un texte alphanumérique, conversion ASCII en hexadécimal) développés dans les précédents compléments peuvent être repris sans aucune modification.

La logique du programme est la suivante :



La technique utilisée pour effectuer des sauts de page consiste à comptabiliser le nombre de lignes imprimées depuis le début de la page courante et à insérer des bas et haut de page aux bons moments.

Le listing du programme de formatage Assembleur est le suivant :

```

1          ORG 3000H
2          LOAD 3000H
3          ;=====
4          ; FORMATAGE DES LISTINGS
5          ;=====
6          ;
7 3000 3E02          LD  A,2
8 3002 CD0EBC        CALL MODE          ;Passage en mode 2
9 3005 217131        LD  HL,MES1
10 3008 CDD630        CALL AFALPH        ;Affichage du titre
11          ;
12          ;-----
13          ; Lecture du nom du programme
14          ; a imprimer
15          ;-----
16          ;
17 300B 219D31        LD  HL,MES2
18 300E CDD630        CALL AFALPH        ;Affich. 2eme message
19 3011 3E0C          LD  A,12          ;Saisie sur 12 caract
20 3013 325A31        LD  (MAX),A
21 3016 215D31        LD  HL,BUFNOM
22 3019 225B31        LD  (PBUF),HL    ;Buffer de saisie
23 301C CDE630        CALL SAISIE
24 301F 79           LD  A,C
25 3020 326D31        LD  (LNOM),A     ;Longueur du nom
26 3023 216632        LD  HL,ALALI
27 3026 CDD630        CALL AFALPH        ;Passage a la ligne

```

```

28      ;
29      ;-----
30      ; Lecture du nombre de lignes
31      ; par feuille de papier
32      ;-----
33      ;
34 3029 21C431      LD    HL,MES3
35 302C CDD630      CALL AFALPH      ;Affich 3eme message
36 302F 3E02      LD    A,2
37 3031 325A31      LD    (MAX),A      ;2 Chiffres au maximum
38 3034 216B31      LD    HL,NLF
39 3037 225B31      LD    (PBUF),HL      ;Buffer de saisie
40 303A CDE630      CALL SAISIE
41 303D 216632      LD    HL,ALALI
42 3040 CDD630      CALL AFALPH      ;A la ligne
43      ;
44      ;-----
45      ; Lecture du nombre de lignes
46      ; imprimees par feuille
47      ;-----
48      ;
49 3043 21EE31      LD    HL,MES4
50 3046 CDD630      CALL AFALPH      ;Affich. 4eme message
51 3049 3E02      LD    A,2
52 304B 325A31      LD    (MAX),A      ;2 caract maxi
53 304E 216931      LD    HL,NLI
54 3051 225B31      LD    (PBUF),HL      ;Buffer de saisie
55 3054 CDE630      CALL SAISIE
56 3057 216632      LD    HL,ALALI
57 305A CDD630      CALL AFALPH      ;A la ligne
58      ;

```

```

59      ;-----
60      ; Impression
61      ;-----
62      ;
63 305D 3A6D31      LD    A,(LNOM)      ;Lgr nom fichier
64 3060 47          LD    B,A
65 3061 215D31      LD    HL,BUFNOM    ;@ Nom fichier
66 3064 110068      LD    DE,B2K      ;Buffer 2 KO
67 3067 CD77BC      CALL OPEN
68 306A 210080      LD    HL,DEBFIC    ;@ stockage fichier
69      ;
70      BIS:      EQU $
71 306D CD80BC      CALL INCHAR      ;Lect 1 caractere
72 3070 77          LD    (HL),A
73 3071 23          INC    HL
74 3072 CD89BC      CALL TESTEOF      ;Fin de fichier ?
75 3075 38F6        JR     C,BIS      ;Bouclage en lecture
76 3077 361A        LD    (HL),1AH    ;Termineur
77 3079 CD7ABC      CALL CLOSE      ;Fermeture fichier
78      ;
79      ; Conversions ASCII -> Hexa
80      ;
81 307C 216931      LD    HL,NLI
82 307F CD3E31      CALL ASCHEX
83 3082 326E31      LD    (HNLI),A    ;Nbre lign a imp
84 3085 216B31      LD    HL,NLF
85 3088 CD3E31      CALL ASCHEX
86 308B 326F31      LD    (HNLH),A    ;Nbre ligne/page
87      ;
88 308E 3A6E31      LD    A,(HNLI)
89 3091 47          LD    B,A

```

```

90 3092 3A6F31      LD   A,(HNLH)
91 3095 90          SUB  B
92 3096 327031      LD   (NSL),A      ;Nbre sauts de ligne
93                ;
94 3099 47          LD   B,A      ;Nbre sauts de ligne
95 309A 3A6E31      LD   A,(HNL1)
96 309D 4F          LD   C,A      ;Nbre ligne imp
97 309E 1600        LD   D,0      ;Index de ligne
98 30A0 2100B0      LD   HL,DEBFIC ;Debut de fichier
99                SUIIMP: EQU $
100 30A3 7E          LD   A,(HL)
101 30A4 FE1A        CP   1AH      ;Terminateur ?
102 30A6 2B0F        JR   Z,FIN     ;Oui
103 30A8 FE0A        CP   LF       ; A la ligne ?
104 30AA 2B0C        JR   Z,LIPLUN  ;Ligne + 1
105                SUI2IMP: EQU $
106                BBUS1: EQU $
107 30AC CD2EBD      CALL BUSY
108 30AF 3BFB        JR   C,BBUS1   ;Boucle si occupe
109 30B1 CD2BBD      CALL PRCHAR    ;Impression
110                SUI3IMP: EQU $
111 30B4 23          INC  HL      ;Prochain caractere
112 30B5 1BEC        JR   SUIIMP    ;Suite impression
113                FIN: EQU $
114 30B7 C9          RET
115                LIPLUN: EQU $
116 30B8 14          INC  D
117 30B9 7A          LD   A,D
118 30BA B9          CP   C
119 30BB 3B15        JR   C,SUI4IMP
120                ;Sauts de ligne
121 30BD 3E0A        LD   A,LF

```

```

122          SAULIG:      EQU  $
123          BBUS2:      EQU  $

124 30BF CD2EBD          CALL BUSY
125 30C2 38FB           JR   C,BBUS2          ;Boucle si occupe
126 30C4 CD2BBD          CALL PRCHAR          ;Impression
127 30C7 05             DEC  B
128 30C8 20F5           JR   NZ,SAULIG
129 30CA 3A7031          LD   A,(NSL)
130 30CD 47             LD   B,A              ;Restitution
131 30CE 1600           LD   D,0
132 30D0 18E2           JR   SUI3IMP
133          SUI4IMP:    EQU  $
134 30D2 3E0A           LD   A,LF            ;Restitution du LF
135 30D4 18D6           JR   SUI2IMP

136          ;
137          ;
138          ;=====
139          ; ZONE DES SOUS-PROGRAMMES
140          ;=====
141          ;
142          ;-----
143          ; Affichage d'un texte alphanum.
144          ;-----
145          ;Entree: @ de depart dans HL
146          ;Sortie: aucun registre ecrase
147          ;-----

148          AFALPH:     EQU  $              ;Point d'entree
149 30D6 E5             PUSH HL
150 30D7 F5             PUSH AF
151          ME1:        EQU  $              ;Boucle d'affichage
152 30D8 7E             LD   A,(HL)
153 30D9 FEFF           CP   0FFH

```

```

154 30DB 2806          JR    Z,ME2          ;Fin d'affichage
155 30DD CD5ABB        CALL PRINT          ;Af. caractere
156 30E0 23           INC    HL            ;Caractere suivant
157 30E1 18F5          JR     ME1
158                   ME2:    EQU    $
159 30E3 F1            POP    AF
160 30E4 E1            POP    HL
161 30E5 C9            RET
162                   ;
163                   ;-----
164                   ; Saisie de caracteres
165                   ;-----
166                   ;Entree: (MAX)=Nbre de caracteres
167                   ;Sortie: Aucun registre ecrase
168                   ;-----
169                   ;
170                   SAISIE:    EQU    $
171 30E6 3A5A31         LD     A,(MAX)
172 30E9 57             LD     D,A          ;Nombre de caract. a lire
173 30EA 010000         LD     BC,0        ;Index dans le buffer
174                   S1:      EQU    $
175 30ED 2A5B31         LD     HL,(PBUF)    ;Buffer de lecture
176 30F0 CD06BB        CALL READ          ;Lecture 1 caractere
177 30F3 FE0D          CP     CR          ;Carriage Return ?
178 30F5 283C          JR     Z,S3        ;Oui => Fin de saisie
179 30F7 FE7F          CP     DEL         ;DElete ?
180 30F9 2818          JR     Z,S2        ;Oui => Traitement DEL
181 30FB F5            PUSH AF
182 30FC 3E00          LD     A,BS
183 30FE CD5ABB        CALL PRINT
184 3101 F1            POP    AF
185 3102 CD5ABB        CALL PRINT

```



```

186 3105 09          ADD  HL,BC
187 3106 77          LD   (HL),A          ;Sauvegarde
188 3107 03          INC  BC
189 3108 3E5F        LD   A,CURS
190 310A CD5ABB      CALL PRINT
191 310D 79          LD   A,C
192 310E BA          CP   D
193 310F 20DC        JR   NZ,S1          ;Suite de la saisie
194 3111 1B20        JR   S3            ;Fin de saisie
195                S2: EQU  $
196 3113 79          LD   A,C
197 3114 B7          OR   A
198 3115 2B06        JR   Z,S1          ;DEL non accepte
199 3117 0B          DEC  BC
200 3118 3E0B        LD   A,BS
201 311A CD5ABB      CALL PRINT          ;Retour en arriere
202 311D 3E20        LD   A,BLANC
203 311F CD5ABB      CALL PRINT          ;Effacement caractere
204 3122 3E0B        LD   A,BS
205 3124 CD5ABB      CALL PRINT          ;Retour en arriere
206 3127 3E0B        LD   A,BS
207 3129 CD5ABB      CALL PRINT          ;Retour en arriere
208 312C 3E5F        LD   A,CURS
209 312E CD5ABB      CALL PRINT          ;Affichage curseur
210 3131 1BBA        JR   S1
211                S3: EQU  $
212 3133 3E0B        LD   A,BS
213 3135 CD5ABB      CALL PRINT          ;Retour en arriere
214 3138 3E20        LD   A,BLANC
215 313A CD5ABB      CALL PRINT          ;Effacement caract.
216 313D C9          RET
217                ;

```

```

218      ;-----
219      ; Conversion ASCII -> HEXA
220      ;-----
221      ;Entree: HL=@ des car a convertir
222      ;Sortie: A=Conversion
223      ;-----
224      ;
225      ASCHEX:      EQU    $
226 313E C5          PUSH BC                      ;Sauvegarde
227 313F 7E          LD     A,(HL)
228 3140 FE40        CP     40H
229 3142 3802        JR     C,AS1
230 3144 D607        SUB    7
231      AS1:        EQU    $
232 3146 D630        SUB    30H                      ;Conversion MSQ
233 3148 17          RLA
234 3149 17          RLA
235 314A 17          RLA
236 314B 17          RLA                      ;et passage en poids fort
237 314C 47          LD     B,A                      ;Sauvegarde
238 314D 23          INC    HL
239 314E 7E          LD     A,(HL)
240 314F FE40        CP     40H
241 3151 3802        JR     C,AS2
242 3153 D607        SUB    7
243      AS2:        EQU    $
244 3155 D630        SUB    30H                      ;Conversion MSQ
245 3157 B0          OR     B                      ;Octet complet
246 3158 C1          POP    BC                      ;Restitution de BC
247 3159 C9          RET

```

```

248      ; -----
249      ; ZONE DES CONSTANTES
250      ; -----
251      ;
252      CR:      EQU   13           ; Carriage Return
253      LF:      EQU   10           ; Line Feed
254      BS:      EQU   8            ; Back Space
255      DEL:     EQU   127          ; DElete
256      BLANC:   EQU   32           ; Espace
257      CURS:    EQU   95           ; Curseur
258      DEBFI:   EQU   8000H        ; Debut Fichier
259      B2K:     EQU   6800H        ; Buffer 2 KO
260      PRINT:   EQU   0BB5AH       ; TXT OUTPUT
261      READ:    EQU   0BB06H       ; KM WAIT CHAR
262      MODE:    EQU   0BC0EH       ; SCR SET MODE
263      OPEN:    EQU   0BC77H       ; CAS IN OPEN
264      CLOSE:   EQU   0BC7AH       ; CAS IN CLOSE
265      TESTEOF: EQU   0BC89H       ; CAS TEST EOF
266      INCHAR:  EQU   0BC80H       ; CAS IN CHAR
267      PRCHAR:  EQU   0BD2BH       ; MC PRINT CHAR
268      BUSY:    EQU   0BD2EH       ; MC BUSY PRINTER
269      ; -----
270      ; ZONE DES VARIABLES
271      ; -----
272      ;
273      MAX:     DS    1             ; Nbre max de caract.
274      PBUF:    DS    2             ; Pointeur sur un buffer
275      BUFNOM:  DS    12            ; Nom fich a lister
276      NLI:     DS    2             ; Nbre lignes imprimees
277      NLF:     DS    2             ; Nbre lignes/feuille
278      LNDM:    DS    1             ; Lgr nom fichier

```

```

279      HNL1:      DS      1      ;Ligne impr en hexa
280      HNLH:      DS      1      ;Lign/feuille en hexa
281      NSL:       DS      1      ;Nbre de sauts de ligne
282      ;
283      ;-----
284      ; ZONE DES MESSAGES
285      ;-----
286      MES1:      EQU      $
287 3171 496D7072      DB      "Impression format
287 3175 65737369
287 3179 6F6E2066
287 317D 6F726D61
287 3181 7465650D
287 3185 0A
288 3186 2D2D2D2D      DB      "-----
288 318A 2D2D2D2D
288 318E 2D2D2D2D
288 3192 2D2D2D2D
288 3196 2D2D2D0D
288 319A 0A0A
289 319C FF      DB      0FFH      ;Terminateur
290      ;-----
291      MES2:      EQU      $
292 319D 456E7472      DB      "Entrez le nom du
292 31A1 657A206C
292 31A5 65206E6F
292 31A9 6D206475
292 31AD 2070726F
292 31B1 6772616D
292 31B5 6D6520
293 31B8 61206C69      DB      "a lister : "
293 31BC 73746572

```

```

293 31C0 203A20
294 31C3 FF                      DB  .0FFH                      ;Terminateur
295                               ;-----
296                               MES3:      EQU  $
297 31C4 456E7472                  DB  "Entrez le nombre
297 31C8 657A206C
297 31CC 65206E6F
297 31D0 6D627265
297 31D4 20646520
297 31D8 6C69676E
297 31DC 657320
298 31DF 70617220                  DB  "par feuille : "
298 31E3 66657569
298 31E7 6C6C6520
298 31EB 3A20
299 31ED FF                      DB  .0FFH                      ;Terminateur
300                               ;-----
301                               MES4:      EQU  $
302 31EE 456E7472                  DB  "Entrez le nombre
302 31F2 657A206C
302 31F6 65206E6F
302 31FA 6D627265
302 31FE 20646520
302 3202 6C69676E
302 3206 657320
303 3209 696D7072                  DB  "imprimees par feu
303 320D 696D6565
303 3211 73207061
303 3215 72206665
303 3219 75696C6C
303 321D 65203A20

```

```

304 3221 FF                      DB    0FFH                      ;Terminateur
305                               ;-----
306                               MES5:    EQU    $
307 3222 456E7472                DB    "Entrez le nombre
307 3226 657A206C
307 322A 65206E6F
307 322E 6D627265
307 3232 206465
308 3235 20636172                DB    " caracteres par l
308 3239 61637465
308 323D 72657320
308 3241 70617220
308 3245 6C69676E
308 3249 65203A20
309 324D FF                      DB    0FFH                      ;Terminateur
310                               ;-----
311                               MES6:    EQU    $
312 324E 496D7072                DB    "Impression en cou
312 3252 65737369
312 3256 6F6E2065
312 325A 6E20636F
312 325E 75727320
312 3262 2E2E2E
313 3265 FF                      DB    0FFH                      ;Terminateur
314                               ;-----
315                               ALALI:    EQU    $
316 3266 0D0A0AFF                DB    CR,LF,LF,0FFH
317                               ;-----
318                               END

```

- Lignes 7 et 8 : Passage en mode 2.
- Lignes 9 et 10 : Affichage du titre.
- Lignes 12 à 27 : Affichage d'un texte et saisie du nom du fichier à imprimer.
- Lignes 29 à 42 : Affichage d'un texte et saisie du nombre de lignes par feuille de papier.
- Lignes 44 à 57 : Affichage d'un texte et saisie du nombre de lignes imprimées par feuille de papier.
- Lignes 63 à 77 : Mise en mémoire du fichier à imprimer.
- Lignes 78 à 86 : Conversion Hexa des données ASCII saisies.
- Lignes 88 à 135 : Impression.
- Lignes 142 à 161 : Affichage d'un texte alphanumérique.
- Lignes 163 à 216 : Saisie d'une chaîne alphanumérique.
- Lignes 218 à 247 : Conversion de deux caractères codés en ASCII en un octet codé en hexadécimal.

Comme toujours, voici la version chargeur Basic du programme Assembleur :

```

1000 '=====
1010 'CHARGEUR HEXADECIMAL DU
1020 'FORMATEUR DE LISTINGS
1030 '=====
1040 '
1050 FOR i=&3000 TO &3269
1060   READ a$
1070   POKE i,VAL("&h"+a$)
1080 NEXT i
1090 CALL &3000
1100 END
1110 '
1120 DATA 3E,2,CD,E,BC,21,71,31,CD,D6,30,21,9D,31,CD,D6
1130 DATA 30,3E,C,32,5A,31,21,5D,31,22,5B,31,CD,E6,30,79
1140 DATA 32,6D,31,21,66,32,CD,D6,30,21,C4,31,CD,D6,30,3E
1150 DATA 2,32,5A,31,21,6B,31,22,5B,31,CD,E6,30,21,66,32
1160 DATA CD,D6,30,21,EE,31,CD,D6,30,3E,2,32,5A,31,21,69
1170 DATA 31,22,5B,31,CD,E6,30,21,66,32,CD,D6,30,3A,6D,31
1180 DATA 47,21,5D,31,11,0,68,CD,77,BC,21,0,80,CD,80,BC
1190 DATA 77,23,CD,89,BC,38,F6,36,1A,CD,7A,BC,21,69,31,CD
1200 DATA 3E,31,32,6E,31,21,6B,31,CD,3E,31,32,6F,31,3A,6E
1210 DATA 31,47,3A,6F,31,90,32,70,31,47,3A,6E,31,4F,16,0
1220 DATA 21,0,80,7E,FE,1A,28,F,FE,A,28,C,CD,2E,BD,38
1230 DATA FB,CD,2B,BD,23,18,EC,C9,14,7A,B9,38,15,3E,A,CD
1240 DATA 2E,BD,38,FB,CD,2B,BD,5,20,F5,3A,70,31,47,16,0
1250 DATA 18,E2,3E,A,18,D6,E5,F5,7E,FE,FF,28,6,CD,5A,BB
1260 DATA 23,18,F5,F1,E1,C9,3A,5A,31,57,1,0,0,2A,5B,31
1270 DATA CD,6,BB,FE,D,28,3C,FE,7F,28,18,F5,3E,8,CD,5A
1280 DATA BB,F1,CD,5A,BB,9,77,3,3E,5F,CD,5A,BB,79,BA,20
1290 DATA DC,18,20,79,B7,28,D6,B,3E,8,CD,5A,BB,3E,20,CD
1300 DATA 5A,BB,3E,8,CD,5A,BB,3E,8,CD,5A,BB,3E,5F,CD,5A
1310 DATA BB,18,BA,3E,8,CD,5A,BB,3E,20,CD,5A,BB,C9,C5,7E
1320 DATA FE,40,38,2,D6,7,D6,30,17,17,17,17,47,23,7E,FE

```

```

1330 DATA 40,38,2,D6,7,D6,30,B0,C1,C9,2C,4C,46,2C,4C,46
1340 DATA 2C,30,46,46,48,D,A,3B,2D,2D,2D,2D,2D,2D,2D
1350 DATA 2D,49,6D,70,72,65,73,73,69,6F,6E,20,66,6F,72,6D
1360 DATA 61,74,65,65,D,A,2D,2D,2D,2D,2D,2D,2D,2D,2D
1370 DATA 2D,2D,2D,2D,2D,2D,2D,2D,2D,D,A,A,FF,45,6E,74
1380 DATA 72,65,7A,20,6C,65,20,6E,6F,6D,20,64,75,20,70,72
1390 DATA 6F,67,72,61,6D,6D,65,20,61,20,6C,69,73,74,65,72
1400 DATA 20,3A,20,FF,45,6E,74,72,65,7A,20,6C,65,20,6E,6F
1410 DATA 6D,62,72,65,20,64,65,20,6C,69,67,6E,65,73,20,70
1420 DATA 61,72,20,66,65,75,69,6C,6C,65,20,3A,20,FF,45,6E
1430 DATA 74,72,65,7A,20,6C,65,20,6E,6F,6D,62,72,65,20,64
1440 DATA 65,20,6C,69,67,6E,65,73,20,69,6D,70,72,69,6D,65
1450 DATA 65,73,20,70,61,72,20,66,65,75,69,6C,6C,65,20,3A
1460 DATA 20,FF,45,6E,74,72,65,7A,20,6C,65,20,6E,6F,6D,62
1470 DATA 72,65,20,64,65,20,63,61,72,61,63,74,65,72,65,73
1480 DATA 20,70,61,72,20,6C,69,67,6E,65,20,3A,20,FF,49,6D
1490 DATA 70,72,65,73,73,69,6F,6E,20,65,6E,20,63,6F,75,72
1500 DATA 73,20,2E,2E,2E,FF,D,A,A,FF,0,0,0,0,0,0

```

et les données de checksum correspondantes :

```

6 F4 89 CA 73 2C 1F BC B7 3E 9F 50 2B 9D A3 23 EA A6 30 9 A2 19 EC 30 7B DF AC 2
2 E4 C6 B E2 20 A0 5A 3 C6 45 3F

```

II. Formattage des listings en Basic

Comme le programme en Basic est bien plus simple que le précédent, nous y avons ajouté une fonctionnalité : la possibilité de définir le nombre de caractères par ligne. Ainsi, un saut de page pourra éventuellement se produire en plein milieu d'une ligne, et le découpage éventuel des lignes sera également comptabilisé pour effectuer les sauts de page aux bons moments.

A titre d'exemple, voici le résultat du programme appliqué aux 11 premières lignes de son propre listing :

— Voir listing page 17 —



FORMATAGE DES LISTINGS VERSION BASIC

Nom du listing à imprimer : FORBAS
 Nombre de lignes par feuille de papier : 6
 Nombre de lignes imprimées par feuille : 4
 Nombre de caractères par ligne : 20

Impression en cours ...

```
1000 '=====
=====
=====
1010 ' FORMATAGE DES
```

LISTINGS VERSION BA
 SIC

```
1020 '=====
=====
```

```
=====
1030 '
1040 MODE 2
1050 PRINT "FORMATAG
```

E DES LISTINGS VERSI
 ON BASIC"

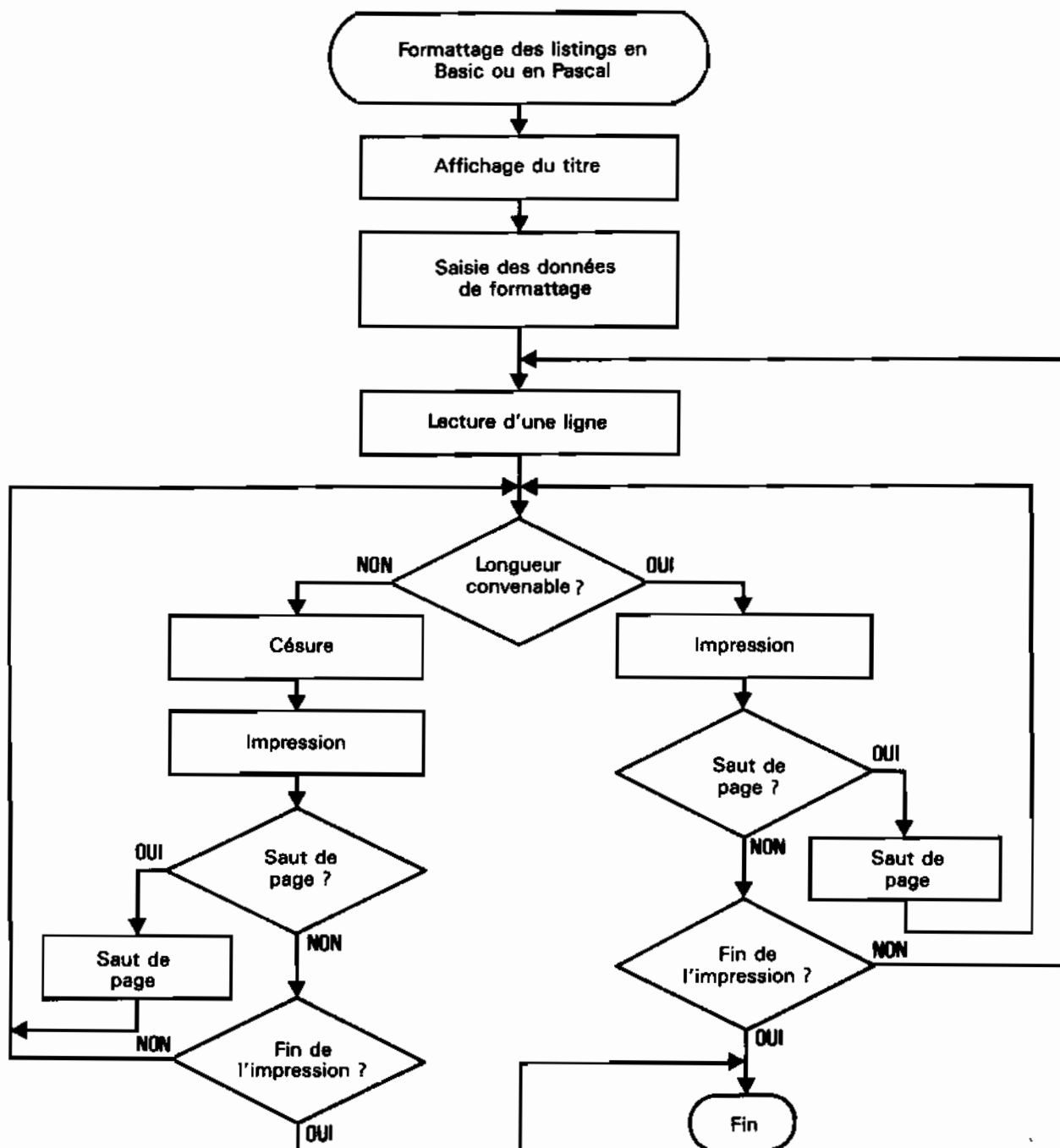
```
1060 PRINT "-----
-----
```

```
-----"
1070 PRINT
1080 INPUT "Nom du l
isting à imprimer :
```

```
",nlf
1090 INPUT "Nombre d
e lignes par feuille
de papier : ",nlf
```

```
1100 INPUT "Nombre d
e lignes imprimées p
ar feuille : ",nlf
```

La logique du programme apparaît dans l'ordinogramme suivant :



Rappelons une fois encore que ce programme ne s'applique qu'aux fichiers ASCII. Il est donc impossible de lister un programme Basic sauvegardé sous sa forme habituelle.

Le listing du programme de formatage Basic est le suivant :

```

1000 '=====
1010 ' FORMATAGE DES LISTINGS VERSION BASIC
1020 '=====
1030 '
1040 MODE 2
1050 PRINT "FORMATAGE DES LISTINGS VERSION BASIC"
1060 PRINT "-----"
1070 PRINT
1080 INPUT "Nom du listing a imprimer : ",nl$
1090 INPUT "Nombre de lignes par feuille de papier : ",nlf
1100 INPUT "Nombre de lignes imprimees par feuille : ",nli
1110 INPUT "Nombre de caracteres par ligne : ",ncl
1120 '
1130 hp=INT((nlf-nli)/2) 'Haut de page
1140 bp = nlf-nli-hp 'Bas de page
1150 '
1160 PRINT
1170 PRINT "Impression en cours ..."
1180 '
1190 OPENIN nl$
1200 li=0 'Initialisation
1210 '
1220 FOR i=1 TO hp
1230 PRINT #8 ' Impression du haut de page
1240 NEXT i
1250 '
1260 LINE INPUT #9,a$
1270 IF LEN(a$)<=ncl THEN 1360 'La ligne a une longueur convenable
1280 '
1290 a1$=MID$(a$,1,ncl) 'Premiere partie de la ligne
1300 a2$=MID$(a$,ncl+1,LEN(a$)-ncl) 'Seconde partie de la ligne
1310 li=li+1 'Index de ligne + 1
1320 PRINT #8,a1$ 'Impression de la 1ere partie de la ligne
1330 IF li = nli THEN li=0:FOR i=1 TO bp+hp:PRINT #8:NEXT i
1340 GOTO 1270
1350 '
1360 PRINT #8,a2$ 'Impression de la ligne
1370 li=li+1 'Index de ligne
1380 '
1390 IF EOF THEN 1490 'Fin d'impression
1400 IF li < nli THEN 1260 'L'impression n'est pas finie
1410 li=0
1420 '
1430 FOR i=1 TO bp
1440 PRINT #8 ' Impression du bas de page
1450 NEXT i
1460 '
1470 IF NOT EOF THEN 1220 ELSE 1490
1480 '
1490 CLOSEIN
1500 END

```

- Lignes 1040 à 1110 : Saisie des paramètres de formatage.
- Lignes 1130 et 1140 : Définition des haut et bas de page.
- Lignes 1220 à 1240 : Impression du premier haut de page.
- Ligne 1270 : Test de la longueur de la ligne courante.
- Lignes 1290 à 1340 : Césure d'une ligne trop longue et impression partielle.
- Ligne 1360 : Impression d'une ligne de longueur convenable.
- Lignes 1390 et 1470 : Test de fin d'impression.
- Lignes 1430 à 1450 : Impression d'un bas de page.
- Lignes 1490 à 1500 : Fermeture du fichier et fin du programme.

III. Formattage des listings en Turbo-Pascal

La structure du programme Turbo-Pascal correspond au même ordonnogramme que celle du programme Basic précédent. Cependant, notez que les fonctions essentielles (saut de ligne, saisie, calcul des haut et bas de page, impression) sont réparties en autant de procédures que nécessaire. Le programme principal est très court. Il se contente d'activer séquentiellement les trois procédures principales : Saisie, Calcul et Impression.

Le listing du programme de formatage Turbo-Pascal est le suivant :

```

Program Formatage;
{-----}
{ Formatage des listings version Turbo Pascal }
{-----}

Var I,LI,NLF,NLI,NCL,BP,HP : Integer;
    A,A1 : String[120];
    NL : String[16];
    F : Text;

Procedure Saut_Ligne(NS : Integer);
{-----}
{ Saut d'une ou de plusieurs lignes }
{-----}
begin
    For I:=1 to NS do
        Writeln(Lst);
end;

Procedure Saisie;
{-----}
{ Saisie des donnees de formatage }
{-----}
begin
    ClrScr;
    Writeln('FORMATAGE DES LISTINGS VERSION PASCAL');
    Writeln('=====');
    Writeln;
    Write('Nom du listing a imprimer : ');
    Readln(NL);
    Write('Nombre de lignes par feuille de papier : ');
    Readln(NLF);
    Write('Nombre de lignes imprimees par feuille : ');
    Readln(NLI);
    Write('Nombre de caracteres par ligne : ');
    Readln(NCL);
end;

Procedure Calcul;
{-----}
{ Calculs relatifs au formatage }
{-----}
begin
    HP:=Round((NLF-NLI)/2);    {Haut de page}
    BP:=NLF-NLI-HP;
    writeln(hp,bp);
end;

```

```

Procédure Impression;
{-----}
{ Impression du fichier }
{-----}
begin
  Writeln;
  Writeln('Impression en cours ...');
  Assign(F,NL);
  Reset(F);
  LI:=0; {Initialisation}
  Saut_Ligne(HP); { Impression du haut de page }
  Repeat
    Readln(F,A);
    If Length(a) <= NCL then
      begin
        if (not EOF(F)) and (LI < NLI) then
          begin
            Writeln(Lst,a); { Impression de la ligne }
            LI:=LI+1;      { Index de ligne + 1 }
          end;
        if LI >= NLI then
          begin
            LI:=0;
            Saut_Ligne(BP+HP);
          end;
        end
      else
        begin
          Repeat
            a1:=Copy(a,1,NCL);
            a:=Copy(a,NCL+1,Length(a)-NCL); { Première partie }
            LI:=LI+1; { Seconde partie }
            Writeln(Lst,a1); { Index de ligne }
            If LI=NLI then
              begin
                LI:=0;
                Saut_Ligne(BP+HP);
              end;
            until Length(a)<=NCL;
            If Length(a)<>0 then
              begin
                Writeln(Lst,a);
                If LI+1 = NLI then
                  begin
                    LI:=0;
                    Saut_Ligne(BP+HP);
                  end
                else LI:=LI+1;
              end;
            end;
          end;
        end;
      until eof(F);
    end;

    {=====}
    { PROGRAMME PRINCIPAL }
    {=====}

begin
  Saisie; { des données de formatage }
  Calcul; { relatifs au formatage }
  Impression; { du fichier }
end.

```

9/9

Programmes divers

9/9.1

Générateur de signaux morse

Survivant des temps héroïques des radiocommunications, le code Morse est aujourd'hui encore très utilisé aux côtés des techniques les plus modernes de transmission numérique.

Il s'agit en effet du moyen le plus simple permettant l'échange de messages relativement fiables dans les plus mauvaises conditions.

L'informatique peut remplacer l'opérateur pour les tâches d'émission et, dans une certaine mesure, de réception. Elle peut aussi l'aider à apprendre le morse, qui figure au programme des principaux examens de radio-amateur.

Le logiciel qui va être publié ici est capable de transformer un mot frappé au clavier, en une suite de tonalités représentant les « points » et les « traits » du code de la figure 1.

Ces signaux audio peuvent évidemment être directement émis par tout émetteur « phonie », mais sont surtout utiles à des fins pédagogiques.

Il ne suffit pas, en effet, d'apprendre par cœur le tableau de l'alphabet pour pouvoir prétendre « connaître » le morse. L'opérateur expérimenté est capable de reconnaître « au vol » non seulement chaque lettre ou signe, mais aussi des mots entiers (et pas seulement le fameux « SOS » !)

C'est seulement de cette façon qu'il est possible de ne pas perdre le fil d'une transmission rapide...

A •—	N —•	1 •—
B —•••	O ———	2 ••—
C —•—•	P •—•	3 •••—
D —••	Q —•—•	4 ••••—
E •	R •—•	5 •••••
F ••—•	S •••	6 —••••
G ——•	T —	7 ——•••
H ••••	U ••—	8 ———••
I ••	V •••—	9 ———•
J •—	W •—	0 ———
K —•—	X —••—	
L •—••	Y —•—	
M ——	Z —••	
Point •—•—•— Virgule —••—		

Fig. 1

Le programme de la figure 2 peut être un fort bon professeur, toujours disponible pour « traduire » les mots de votre choix : commencez donc par des mots courants et courts (le fameux CQ-CQ-CQ-CQ des télégraphistes, par exemple), et vous arriverez progressivement à comprendre une partie de ces avalanches de points et de traits que l'on rencontre partout en ondes courtes ! Le cœur du logiciel n'occupe que quelques lignes de BASIC, numérotées de 10 à 510 : le reste du programme est une longue suite d'instructions « IF-THEN » contenant l'équivalent morse de toutes les lettres de l'alphabet. Grâce à l'éditeur de lignes de l'AMSTRAD, chaque ligne peut être rapidement obtenue par simple modification de la précédente.

Lorsque vous serez arrivé à « Z », vous pourrez compléter cette liste par les chiffres, en vous inspirant de la figure 1, et même ajouter les signes de ponctuation si le cœur vous en dit ! Tout est réglé dans ce logiciel pour que le son émis ressemble d'aussi près que possible à une « vraie » transmission. Il est toutefois facile de modifier les instructions SOUND pour obtenir des sons plus aigus ou plus graves, ou pour transformer la cadence de « manipulation ».

Dans cette éventualité, on retiendra qu'un trait doit durer trois fois plus longtemps qu'un point, et qu'un espace entre point et trait doit avoir la durée d'un point, tandis que deux lettres d'un même mot doivent être séparées par la durée d'un trait.

A titre indicatif, précisons aussi que l'espace entre deux mots consécutifs est en principe de sept fois la durée d'un point. Toute modification de la durée du son programmé dans SOUND devrait donc être accompagnée d'une correction des lignes 180 et 500.

Ces normes précises permettent un bon « décodage » des signaux morse non seulement par les opérateurs « humains », mais aussi par les équipements informatiques destinés à la réception automatique.

Il existe en effet des interfaces et des logiciels qui, adaptés à un micro-ordinateur AMSTRAD ou autre, permettent une interprétation automatique des signaux morse reçus.

Même les logiciels les plus élaborés (des milliers d'octets de langage machine) se révèlent assez décevants en présence de signaux morse émis « à la main » : aussi expérimenté soit-il, un opérateur est difficilement capable de respecter une cadence de manipulation parfaitement constante, tandis que même les meilleurs logiciels ont du mal à suivre ces variations.

Les choses se passent beaucoup mieux lorsque c'est un ordinateur qui manipule, en respectant scrupuleusement la cadence.

Seulement, lorsqu'il s'agit de faire communiquer deux ordinateurs entre eux par radio, il existe aujourd'hui des procédés autrement plus performants que le morse (RTTY, radio-paquets, AMTOR, etc.).

Quoi qu'il en soit, votre AMSTRAD est désormais capable de produire d'excellents signaux morse, que vous êtes évidemment libres d'utiliser à votre convenance, amis lecteurs...

```

10 CLS
20 PRINT "mot a transmettre?"
30 INPUT b$
40 CLS
60 FOR f=1 TO LEN(b$)
70 c$=MID$(b$,f,1)
80 GOSUB 1000
90 a$=a$+d$
100 NEXT f
110 CLS:PRINT b$
120 FOR f=1 TO LEN(a$)
130 z$=MID$(a$,f,1)
140 IF z$="\" THEN GOSUB 500
150 IF z$="\" THEN 190
160 IF z$="." THEN SOUND 1,100,8,12
170 IF z$="-" THEN SOUND 1,100,24,12
175 IF z$="~" THEN GOSUB 500
180 FOR g=1 TO 100:NEXT g
190 NEXT f
200 RUN
500 FOR g=1 TO 300:NEXT g
510 RETURN
1000 IF c$=" " THEN d$=""
1001 IF c$="a" THEN d$=".-"

```

Fig. 2

```
1002 IF C$="b" THEN d$="...."
1003 IF C$="c" THEN d$="...-"
1004 IF C$="d" THEN d$="..--"
1005 IF C$="e" THEN d$=".-.-"
1006 IF C$="f" THEN d$="..-.-"
1007 IF C$="g" THEN d$="---."
1008 IF C$="h" THEN d$="...."
1009 IF C$="i" THEN d$="...-"
1010 IF C$="j" THEN d$="..-.-"
1011 IF C$="k" THEN d$="..--"
1012 IF C$="l" THEN d$=".-.-"
1013 IF C$="m" THEN d$="---."
1014 IF C$="n" THEN d$="..-.-"
1015 IF C$="o" THEN d$="...."
1016 IF C$="p" THEN d$="...-."
1017 IF C$="q" THEN d$="---.-"
1018 IF C$="r" THEN d$="..-.-"
1019 IF C$="s" THEN d$="...-"
1020 IF C$="t" THEN d$="..--"
1021 IF C$="u" THEN d$="..-.-"
1022 IF C$="v" THEN d$="...-.-"
1023 IF C$="w" THEN d$="..-.-"
1024 IF C$="x" THEN d$="..-.-"
1025 IF C$="y" THEN d$="..-.-"
1026 IF C$="z" THEN d$="---.-"
1990 d$=d$+"\"
2000 RETURN
```

Fig. 2 (suite)

9/9.2

Filtrage de fichiers ASCII

Si vous travaillez avec un traitement de texte évolué sur votre CPC (*Pocket Wordstar* par exemple), vous connaissez très certainement une de ses fonctions les plus évoluées : « Recherche et Remplace ». Cette fonction recherche une chaîne alphanumérique dans le texte en mémoire et la remplace par une autre chaîne alphanumérique.

Si les modifications à apporter aux fichiers texte que vous utilisez sont nombreuses et/ou systématiques, vous préférerez certainement passer par le programme utilitaire que nous présentons ici. Ce programme permet de rechercher et de remplacer automatiquement un nombre quelconque de chaînes alphanumériques. La seule limite est la taille de mémoire RAM disponible.

Afin d'automatiser les remplacements, vous devez dans un premier temps créer un fichier de directives qui contient le libellé des chaînes à remplacer et le libellé des chaînes qui les remplacent.

Pour ce faire, il vous suffit de connaître :

- le nombre de chaînes à remplacer,
- leur libellé et celui des chaînes qui les remplacent, et de donner un nom au fichier dans lequel ces données vont être stockées.

Ces données apparaissent dans le programme Basic ci-dessous :

```

1000 '=====
1010 '   CONSTITUTION D'UN FICHIER DE DIRECTIVES
1020 '=====
1030 '
1040 DIM o$(100) 'Chaines recherchees
1050 DIM d$(100) 'Chaines rempalcantes
1060 '
1070 FOR i=1 TO 10
1080   READ o$(i) 'Lecture des chaines a remplacer'
1090 NEXT i
1100 '
1110 FOR i=1 TO 10
1120   READ d$(i) 'Lecture des chaines rempalcantes

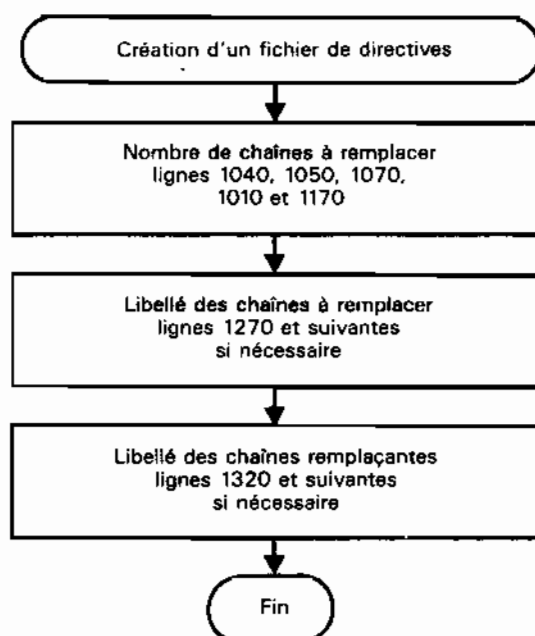
```

```

1130 NEXT i
1140 '
1150 fr$="regle.dat" 'Nom du fichier de regles
1160 OPENOUT fr$
1170 FOR i=1 TO 10
1180   PRINT #9,o$(i) 'Sauvegarde des chaines a remplacer
1190   PRINT #9,d$(i) 'Sauvegarde des chaines remplaçantes
1200 NEXT i
1210 CLOSEOUT
1220 '
1230 '-----
1240 'Chaines a remplacer
1250 '-----
1260 '
1270 DATA a,b,c,d,e,f,g,h,i,1,2,30
1280 '-----
1290 'Chaines remplaçantes
1300 '-----
1310 '
1320 DATA A,B,C,D,E,F,G,H,un,deux,trente

```

L'ordinogramme suivant vous indique le numéro de lignes où reporter les données.

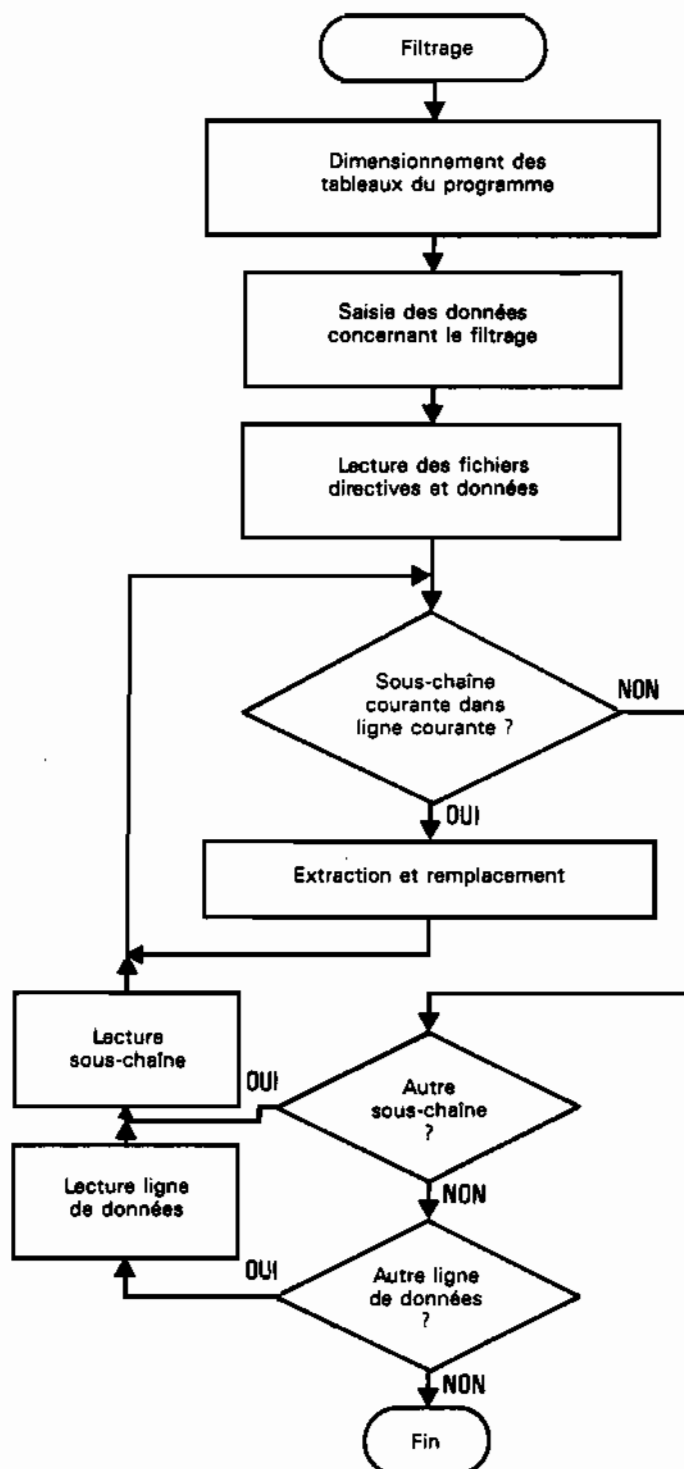


Cette première étape achevée, il vous suffit d'exécuter le programme de filtrage, et d'entrer :

- le nom du fichier de directives ;
- le nom du fichier à filtrer ;
- le nom du fichier résultant.

Quelques instants après avoir entré ces informations, le message « **Filtrage en cours** » apparaît sur l'écran. Ce message est suivi de l'affichage d'un ou de plusieurs points décimaux. Chaque point correspond au traitement d'une ligne dans le fichier à filtrer. Lorsque le filtrage est terminé, le message « **Filtrage terminé** » est affiché sur l'écran.

La logique du programme de filtrage est la suivante :



Le listing du programme de filtrage est le suivant :

```

1000 '=====
1010 ' FILTRAGE DE FICHIERS ASCII
1020 '=====
1030 '
1040 '-----
1050 'Programme principal
1060 '-----
1070 '
1080 GOSUB 2000 'Initialisation
1090 GOSUB 3000 'Entree des donnees
1100 GOSUB 4000 'Conversion
1110 END
2000 '-----
2010 ' Initialisation
2020 '-----
2030 '
2040 DIM o$(100) 'Chaines origine
2050 DIM d$(100) 'Chaines destination
2060 DIM f$(100) 'lignes a convertir
2070 '
2080 RETURN
3000 '-----
3010 'Entree des donnees
3020 '-----
3030 '
3040 ON ERROR GOTO 10000
3050 MODE 2
3060 PRINT"Filtrage de fichiers ASCII"
3070 PRINT"-----"
3080 PRINT:PRINT:PRINT
3090 INPUT"Nom du fichiers de directives : ";fd$
3100 a=1:OPENIN fd$:CLOSEIN
3110 INPUT "Nom du fichier a filtrer : ";ff$
3120 a=2:OPENIN ff$:CLOSEIN
3130 INPUT "Nom du fichier resultat : ";fr$
3140 RETURN
4000 '-----
4010 ' Conversion
4020 '-----
4030 '
4040 a=3
4050 OPENIN fd$
4060 i=1
4070 WHILE NOT EOF
4080 INPUT #9,o$(i) 'Chaines a convertir
4090 INPUT #9,d$(i) 'Conversions
4100 i=i+1
4110 WEND
4120 Nbre=i-1 'Nombre de regles de conversion
4130 CLOSEIN
4140 '
4150 OPENIN ff$
4160 i=1 'Initialisation
4170 WHILE NOT EOF
4180 INPUT #9,f$(i) 'Lignes a filtrer
4190 i=i+1
4200 WEND
4210 nbli=i-1 'Nombre de lignes a traiter

```

```

4220 CLOSEIN
4230
4240
4250 PRINT:PRINT"Filtrage en cours ..."
4260 OPENOUT fr$ "Ouverture du fichier resultat"
4270 FOR i=1 TO nbli
4280   FOR j=1 TO nbre
4290     bis=1
4300     WHILE bis=1
4310       p=INSTR(f$(i),o$(j))
4320       IF p=0 THEN 4420 'Fin de la recherche
4330       ta$=f$(i) 'Memorisation tampon
4340       lf=LEN(f$(i)) 'Longueur de la chaîne origine
4350       ld=LEN(d$(j)) 'Longueur de la chaîne remplaçante
4360       lo=LEN(o$(j)) 'Longueur de la chaîne à remplacer
4370       f$(i)=" " 'Mise à vide de la chaîne origine
4380       IF p<>1 THEN f$(i)=MID$(ta$,1,p-1)
4390       f$(i)=f$(i)+d$(j) 'Modification
4400       f$(i)=f$(i)+MID$(ta$,p+lo,lf-p-lo+1) 'Extraction 2eme partie
4410       GOTO 4430 'Suite de la recherche
4420       bis=0
4430     WEND
4440   NEXT j
4450   PRINT #9,f$(i)
4460   PRINT". "; 'pour patienter
4470 NEXT i
4480 CLOSEOUT 'Fermeture du fichier resultat
4490 PRINT:PRINT"Filtrage termine"
4500 RETURN
10000 '-----
10010 ' Routine d'erreur
10020 '-----
10030 PRINT:PRINT"Fichier inexistant"
10040 IF a=1 THEN RESUME 3090
10050 IF a=2 THEN RESUME 3110

```

- Lignes 1000 à 1110 : Programme principal.
- Lignes 2000 à 2080 : Dimensionnement des tableaux.
- Lignes 3000 à 3140 : Saisie des données concernant le filtrage.
- Lignes 4000 à 4500 : Filtrage.
- Lignes 4070 à 4130 : Lecture des chaînes à convertir.
- Lignes 4150 à 4220 : Lecture des chaînes de conversion.
- Lignes 4270 à 4470 : Algorithme de filtrage.
- Lignes 4310 : Recherche d'une chaîne dans la ligne courante.
- Ligne 4380 : Isolement 1^{re} partie de la chaîne.
- Ligne 4390 : Conversion.
- Ligne 4400 : Isolement 2^e partie de la chaîne.
- Lignes 10000 à 10050 : Routine d'erreur activée si le fichier de directives ou le fichier à filtrer n'existent pas sur le disque courant.

9/10

Gestion familiale

Ce chapitre regroupe les programmes utilitaires de gestion familiale. Comptabilité familiale, gestion d'une bibliothèque ou carnet d'adresses, tous ces programmes qui facilitent la vie se retrouveront ici.

9/10.1

Gestion de compte bancaire

I. But du programme

Le but de ce programme est de permettre la mise à jour d'un compte bancaire, postal ou même d'un compte épargne. Ce qui permet à l'utilisateur de limiter le nombre de feuillets décrivant les opérations concernant la tenue du compte ainsi que les risques d'erreurs dans les calculs.

Ce programme se devait :

- d'être simple d'utilisation ;
- d'avoir une présentation claire et similaire à celle d'un cahier de tenue de compte ;

- de posséder un minimum d'informations concernant les fonctions utilisées afin d'éviter de se référer trop souvent au mode d'emploi ;
- de limiter les erreurs de saisie (les erreurs de calcul étant impossibles).

Le premier point est résolu par l'utilisation d'un minimum de touches, et notamment par l'utilisation de trois touches « fonction » du clavier, ~~plus trois touches~~ significatives.

La présentation retenue est similaire à celle que l'on peut trouver à la fin des carnets de Compte Chèque Postal, ressemblant aussi à celle de la majorité des relevés envoyés par les établissements bancaires.

Pour résoudre le troisième point évoqué, le programme affiche des messages concernant les touches utilisables dans les différents menus, accompagnées de leur signification.

Des demandes de confirmation permettront de minimiser les risques d'erreurs à la saisie.

II. Mode d'emploi

A. CRÉATION DU FICHIER

Après avoir enregistré les différentes parties du programme, (voir chapitre IV), vous allez d'abord créer votre fichier de tenue de compte, ainsi que le programme de lancement.

Pour cela, lancez le programme intitulé « OUVCOMPT.BAS ».

L'écran suivant s'affichera alors :

*** OUVERTURE D'UN COMPTE ***	
Date :→.....	
Centre..... Numéro c/c.....	
Nom :	
Adresse :	
Ville :	
Nom du 1er fichier :....	
Avoir :	

La flèche vous indique la zone d'écriture dans laquelle vous vous trouvez. Le nombre maximum de caractères à frapper est de 9 pour la rubrique « Date », 12 pour « Centre », 15 pour « Numéro c/c », 28 pour « Nom »,

deux fois 28 pour « Adresse », 28 pour « Ville », 4 pour « Nom du 1^{er} fichier » et 11 caractères pour la zone « Avoir ».

Vous répondez dans chacune des zones et validez par la touche <RETURN> si besoin. Il vous sera demandé ensuite le type de compte à créer (3 caractères maximum), puis une confirmation. Moyennant quelques accès disquette et un peu de patience, vous verrez apparaître l'écran suivant, récapitulant les renseignements concernant votre compte :

```
*** Gestion d'un Compte Courant ***

Centre.      Numero c/o.
PARIS        43 07 60 50
Editions HEKA
12, cour St Eloi
75012 PARIS

    mis a jour le :17 11 87
    par :BANME1
Fichier en cours :BANME1
```

Vous pouvez, si vous le désirez, arrêter à partir de maintenant toute opération et reprendre plus tard.

En demandant le catalogue de la disquette, vous vous apercevrez que quatre fichiers et un programme basic ont été créés (« XXXNOM. », « CTAMP. », « INTXXX. », « VARXXX. », « XXX.BAS » ;

XXX représentant le type de compte et NOM le nom donné au fichier).

XXXNOM. est le fichier contenant les opérations effectuées sur le compte.

CTAMP. est un fichier temporaire de variables.

INTXXX. contient toutes les références du compte.

VARXXX. contient les renseignements relatifs aux fichiers utilisés par le compte, le nom du fichier en cours, ainsi que l'avoir disponible et la date de la dernière opération.

XXX.BAS est le programme qui vous permettra d'effectuer votre gestion.

B. UTILISER LA GESTION DE COMPTE

Dorénavant, pour utiliser le programme de gestion de compte, il vous suffira de frapper : RUN « XXX.BAS ».

Après quelques instants, on retrouvera l'écran décrit précédemment. En appuyant sur une touche quelconque, vous verrez apparaître le menu suivant :

***** Gestion d'un Compte Courant *****

- possibilités -

- 1 : Répertoire
- 2 : Mise à jour
- 3 : Ouverture-Fermeture d'un fichier
- 4 : Travail sur fichier fermé
- 5 : Catalogue fichiers
- 6 : Sauvegarde

C. LE MENU

• Répertoire

Appuyez sur la touche <1>. Ce choix vous permet de visualiser toutes les transactions effectuées sur le fichier en cours et de rechercher une transaction particulière.

La touche <f1> vous permet de consulter une page du fichier, après avoir donné son numéro.

page No : 1 Autre page 0/N,+1,-1 : ?

Date	Numero	Objet	Credit	Debit	Avoir	Reste le
12 11 87		Initialisation	12233.35		12233.35	
13 11 87	12233	Mise cheque 1	1000.00		13233.35	
14 11 87	12234	Cheque 1		1541.00	11692.35	
15 11 87	12235	Cheque 2		200.25	11492.10	
16 11 87	12236	Cheque 3		2003.00	9489.10	
17 11 87	12237	Rest. Socu.	125.07		9614.17	17 11 87
18 11 87	12238	Cheque 4		5000.15	4614.02	
19 11 87	12239	Carte Bleue		250.00	4364.02	
20 11 87	125	Page Novembre	5000.00		9364.02	

Vous pouvez accéder aux pages suivantes par la touche <+>, et aux précédentes par la touche <->, à moins que vous ne préfériez lire une toute autre page en frappant sur <0>.

Si vous avez frappé **<N>**, vous retournerez au choix précédent. Vous pouvez alors rechercher une opération particulière en frappant **<f2>**, puis en donnant le numéro du chèque.

La touche **** vous renvoie au menu général.

- *Mise à jour*

Cette option vous permet de mettre à jour votre fichier en insérant toute nouvelle opération, ou en modifiant la date de traitement de l'opération par la banque.

— **<f1>** (insertion) vous invite à entrer les différentes coordonnées d'un chèque par exemple (ou toute autre transaction). Si vous ne désirez rien inscrire dans la case pointée, frappez **<RETURN>**.

Une confirmation vous sera demandée à chaque transaction.

— **<f2>** vous permettra d'entrer les dates auxquelles ont été effectivement effectuées les transactions (ce sont celles apparaissant sur votre relevé). Vous frappez donc dans ce menu la touche **<f0>** correspondante au choix de la date de la transaction. Après avoir entré cette date, celle-ci s'affiche en haut à droite de l'écran jusqu'à ce que vous la modifiez une nouvelle fois. Vous pouvez maintenant affecter cette date à tous les numéros que vous rechercherez grâce à **<f1>**, puis que vous validerez grâce à **<f3>**. Vous pouvez, entre-temps, accéder à toutes les pages écrites, pour vérification (touche **<f2>**).

- *Ouverture-Fermeture d'un fichier*

Cette option vous permet d'ouvrir un nouveau fichier, par exemple pour passer d'une année à l'autre, ou lorsque la place disponible en mémoire devient insuffisante.

- *Travail sur fichier fermé*

Vous pouvez, par ce choix reprendre un ancien fichier pour y effectuer des modifications, de la même façon que pour un fichier en cours.

- *Catalogue fichiers*

Ce choix vous affiche les fichiers utilisés depuis l'ouverture du compte.

- *Sauvegarde*

La sauvegarde est à effectuer avant de quitter le programme. Ce choix vous renseigne tout d'abord sur la place disponible en mémoire (vous pouvez fermer le fichier si vous la jugez insuffisante). Il vous indique ensuite l'avoir présenté par votre relevé bancaire. Après confirmation vous pouvez sauvegarder votre fichier, une première fois sur la disquette de travail, puis par mesure de sécurité, sur une seconde disquette.

III. Extension

Il vous reste certainement un traitement de texte caché dans le fond d'un tiroir. Voilà deux bonnes raisons de l'utiliser : vous pouvez imprimer un fichier complet de cette manière, ou encore effectuer des corrections dans celui-ci.

La visualisation du fichier au traitement de texte vous affichera un écran de ce type :

14

```
"03 12 87", "46554", "Cheque Banque", -750,2853.72, ""
"03 12 87", "OSN 4", "Frais Prof.", 2000,3603.72, ""
"30 11 87", "OSN 3", "Carte Bleu", -500,1603.72, ""
"30 11 87", "15454", "Cheque 6", -6000,2103.72, ""
"29 11 87", "12355", "Cheque 5", -1054.5,8103.72, ""
"29 11 87", "125", "Paye Novembre", 5000,9158.22, ""
"27 11 87", "OSN 2", "Carte Bleue", -256.85,4158.22, ""
"19 11 87", "122557", "Cheque 4", -5000.15,4415.07, ""
"16 11 87", "154459", "Rbst. Secu.", 125.87,9415.22, ""
"16 11 87", "122556", "Cheque 3", -2203,9289.35, "17 11 87"
"16 11 87", "122555", "Cheque 2", -200.25,11492.35, ""
"15 11 87", "122554", "Cheque 1", -1541,11692.6, ""
"13 11 87", "152544", "Remise cheque 1", 1000.25,13233.6, ""
"12 11 87", "", "Initialisation.", 12233.35,12233.35, ""
```

14 représente le nombre d'opérations effectuées.

On retrouve dans l'ordre : « Date », « Numéro », « Objet », « Débit » ou « Crédit », « Avoir », « Rentré le », la dernière opération effectuée se trouvant dans le haut du fichier. La quatrième donnée d'une ligne est précédée d'un signe « - » si c'est un débit.

Vous pouvez aussi reprendre votre fichier pour toute autre utilisation, par exemple pour faire une étude statistique de vos dépenses.

IV. Le programme

A. COMPTBIN

Après avoir sauvegardé le programme COMPTBIN vous le lancez, puis vous frappez

**MEMORY &A59A <RETURN>
SAVE "CCPBIN.BIN", B, &A59B, &DE <RETURN>**

Le programme en code machine ainsi créé sera utilisé pour dessiner le cadre de la feuille de calcul afin de gagner du temps à l'affichage.

Programme COMPTBIN

```

5 s=0
10 FOR i=0 TO 220
20 READ a$
30 a=VAL("&"+a$)
35 s=s+a
40 POKE &A59B+i,a
50 NEXT
60 IF S<>23961 THEN PRINT"ERREUR DANS LE
S DATA$....":STOP
100 DATA 00,00,4C,00,9C,01,1C,01,74,01
110 DATA D4,02,2C,02,7F,03,07,0C,12,18
120 DATA 1D,26,2C,32,37,3E,43,47,50,FF
130 DATA 44,61,74,65,4E,75,6D,65,72,6F
140 DATA 4F,62,6A,65,74,43,72,65,64,69
150 DATA 74,44,65,62,69,74,41,76,6F,69
160 DATA 72,52,65,6E,74,72,65,20,6C,65
170 DATA 11,00,00,21,75,01,D5,CD,EA,BB
180 DATA CD,5A,A6,D1,D5,21,03,00,CD,EA
190 DATA BB,CD,5A,A6,D1,21,5B,01,CD,EA
200 DATA BB,CD,5A,A6,DD,21,AA,A5,FD,21
210 DATA B9,A5,DD,7E,00,FE,FF,28,1F,DD
220 DATA 66,00,2E,03,E5,CD,75,BB,FD,7E
230 DATA 00,CD,5A,BB,E1,FD,23,24,DD,7E
240 DATA 01,BC,20,EA,DD,23,DD,23,18,DA
250 DATA 01,00,00,DD,21,9A,A5,C5,DD,5E
260 DATA 01,DD,56,00,21,03,00,CD,EA,BB
270 DATA 11,00,00,21,72,01,CD,F9,BB,C1
280 DATA DD,23,DD,23,0C,3E,08,B9,20,DF
290 DATA C9,11,7F,02,21,00,00,CD,f9,BB
300 DATA C9,26,4F,2E,03,E5,CD,75,BB,3E
310 DATA 20,CD,5A,BB,E1,25,20,F1,18,BA
320 DATA C9

```

B. OUVCOMPT

C'est le programme qui vous permettra de créer les différents fichiers nécessaires à l'utilisation de votre gestion de compte.

Programme OUVCOMPT

```

10 a$="  ":IF RIGHT$(a$,1)=" " THEN a$=
LEFT$(a$,2)
15 RUN 50
20 OPENOUT "ctamp":WRITE #9,a$:CLOSEOUT
25 RUN"compte1
30 WINDOW #0,40,40,25,25:KEY 159,"delete
15"+CHR$(13)+"delete 40- "+CHR$(13)+"got
o 34"+CHR$(13):KEY DEF 68,1,159
32 PRINT#1,"Appuyer sur la touche [TAB]"
:END
34 PRINT#1,:WINDOW#0,1,40,1,25:a$=a$+".b
as":SAVE a$:RUN a$
40 REM ouverture d'un compte
50 MODE 1:LOCATE 6,3:PRINT"*** OUVERTURE
D'UN COMPTE ***":LOCATE 25,8:PRINT"Date
:.....":LOCATE 10,11:PRINT"Centre..
.... Numero c/c.....":LOCATE 5,15:PRINT"
Nom :":LOCATE 1,16:PRINT"Adresse :":LOCA
TE 3,18:PRINT"Ville : "
60 LOCATE 2,21:PRINT"Nom du 1er fichier
:...."
70 LOCATE 2,23:PRINT"Avoir : "
80 point=8:col=31:lg3=9:GOSUB 360:date$(
1)=n$:dmaj$=n$
90 point=13:col=10:lg3=12:GOSUB 360:cent
re$=n$
100 col=23:lg3=15:GOSUB 360:cc$=n$
110 point=15:col=10:lg3=28:GOSUB 360:nom
$=n$
120 point=16:GOSUB 360:inti1$=n$
130 point=17:GOSUB 360:inti2$=n$
140 point=18:GOSUB 360:vil$=n$
150 point=21:col=22:lg3=4:GOSUB 360:nfic
h$=n$
160 point=23:col=10:lg3=11:GOSUB 360:new
a(1)=VAL(n$):OPE(1)=newa(1):Difrent=newa
(1)
170 OBJ$(1)="Initialisation.":ind=1:QSN=
0:num$(1)="":rent$(1)="":nofich=1:majp$=
""
180 WINDOW #1,1,40,25,25:PRINT#1,"Ok :?"
:GOSUB 520:IF a$<>"0" AND a$<>"o" THEN R
UN 50

```



```

190 PRINT#1,"1: CCP - 2: CL - 3: CE
- 4: autre":GOSUB 520
200 ON VAL(a$) GOTO 210,220,230,240:GOTO
190
210 a$="CCP":GOTO 250
220 a$="CL":GOTO 250
230 a$="CE":GOTO 250
240 INPUT#1, "Autre : (3 lettres)";a$:a$
=UPPER$(a$):IF LEN(a$)>3 OR LEN(a$)<1 T
HEN 240 ELSE 250
250 REM SAUVEGARDE VARIABLES CCP
255 POKE &17A,ASC(LEFT$(a$,1)):IF LEN(a$
)=3 THEN POKE &17B,ASC(RIGHT$(LEFT$(a$,2
),1)):POKE &17C,ASC(RIGHT$(a$,1)) ELSE P
OKE &17B,ASC(RIGHT$(a$,1))
258 nfich$=a$+nfich$:catal$=LEFT$(nfich$
+" ",8)
260 var$="var"+a$:OPENOUT var$:WRITE #9,
ind,osn,nfich$,catal$,difrent,dmaj$,majp
$:CLOSEOUT
270 OPENOUT "int"+a$:WRITE #9,centre$,cc
$,nom$,inti1$,inti2$,vil$:CLOSEOUT
280 REM sauvegarde de la fiche
290 OPENOUT nfich$
300 WRITE #9,ind
310 FOR x=1 TO ind
320 WRITE #9,date$(x),num$(x),obj$(x),op
e$(x),newa$(x),rent$(x)
330 NEXT x
340 CLOSEOUT
350 GOTO 30
360 REM ENTREE D'UN MESSAGE LG3 EN POINT
370 NT=0
380 N$=""
390 LOCATE COL,POINT:PRINT CHR$(243):LOC
ATE COL,POINT
400 A$=INKEY$:IF A$="" THEN 400
410 IF A$=CHR$(13) THEN LOCATE COL+NT,PO
INT:PRINT" ":GOTO 510
420 IF A$<> CHR$(27) THEN 460
430 IF NT=0 THEN 400
440 NT=NT+1:N$=LEFT$(N$,NT):LOCATE COL+N
T,POINT:PRINT CHR$(243);" ":LOCATE COL+N
T,POINT
450 GOTO 400
460 NT=NT+1:PRINT A$;
470 IF NT<>LG3 THEN PRINT CHR$(243)
480 LOCATE COL+NT,POINT
490 N$=N$+A$
500 IF NT<LG3 THEN 400
510 RETURN
520 A$=INKEY$:IF A$="" THEN 520 ELSE RET 5* Complément
URN

```

C. COMPTE 1

Ce programme procède à l'initialisation et affiche le tableau initial récapitulant l'intitulé du compte.

Programme COMPTE 1

```
10 'tableau initial CCP
20 BORDER 0:INK 0,0:INK 1,0:INK 2,0
30 MODE 1
40 PEN 1
50 LOCATE 3,3:PRINT"*** Gestion d'un Com
pte Courant ***"
60 INK 1,27
65 SYMBOL 255,&80,&80,&80,&80,&80,&80,&80
0,&80
70 OPENIN "ctamp":INPUT #9,a$:CLOSEIN
101 OPENIN "int"+a$:INPUT#9,centre$,cc$,
nom$,inti1$,inti2$,vil$:CLOSEIN
102 OPENIN "var"+a$:INPUT #9,ind,OSN,nfi
ch$,CATAL$,difrent,dmaj$,majp$:CLOSEIN
130 REM affichage intitule-compte
140 PEN 2
150 LOCATE 1,11:PRINT"Centre.      Numer
o c/c."
160 LOCATE 1,13:PRINT centre$:LOCATE 14,
13:PRINT cc$
170 LOCATE 1,15:PRINT nom$:PRINT inti1$
180 IF inti2$="" THEN 190 ELSE PRINT int
i2$
190 PRINT vil$
200 PRINT:PRINT:PRINT"    mis a jour le :
";dmaj$
210 PRINT,:PRINT"par : "majp$
220 PRINT"Fichier en cours :":PRINT nfi
ch$
230 INK 2,27
235 MEMORY &A59A
236 LOAD"ccpbin.bin
240 RUN"compte2
```

D. COMPTE 2

C'est le programme principal de la gestion de compte.

Programme COMPTE 2

```

10 REM gestion d'un ccp ou compte bancaire
20 ' *****
30 CLEAR
70 DATA 9,9,15,10,10,10,
80 DATA 1,11,21,37,48,71
90 DATA Repertoire,Mise a jour,Ouverture
-Fermeture d'un fichier,Travail sur fichier
ferme,Catalogue fichiers,Sauvegarde
100 OPENIN "ctamp":INPUT#9,i$:CLOSEIN:OPENIN
"var"+i$:INPUT #9,ind,OSN,nfich$,CATAL$,difrent,dmaj$,majp$:CLOSEIN:GOSUB 550:PRINT:GOSUB 1905
110 BORDER 0:INK 0,0:INK 1,0:INK 2,0:MODE 1:PEN 1:GOSUB 1450:INK 1,27:PEN 2:LOCATE 12,9:PRINT"- possibilites -":RESTORE
90:LOCATE 1,12:FOR x=1 TO 6:READ x$:PRINT x;" : ";x$:PRINT:NEXT:INK 2,27
130 GOSUB 1865:IF VAL(j$)>6 THEN 130 ELSE ON VAL(j$) GOSUB 190,580,1470,1590,1710,1730:GOTO 110
140 REM sauvegarde de la fiche
150 GOSUB 1850
160 OPENOUT k$:WRITE #9,ind:FOR x=ind TO 1 STEP -1:WRITE #9,date$(x),num$(x),obj$(x),ope(x),newa(x),rent$(x):NEXT x:CLOSEOUT:RETURN
180 REM repertoire
190 GOSUB 2000
230 PRINT#1,"[F1] : lecture d'une page
- [F2] : recherche d'un numero - [DEL]"
240 GOSUB 1860
250 IF J$="1" THEN 380
260 IF j$=CHR$(&7F) THEN RETURN
270 IF J$<>"2" THEN 240
280 REM recherche d'un numero.
290 point=4
300 IF point=24 THEN GOSUB 8030:GOTO 290
310 GOSUB 1370
320 IF trouve=0 THEN 360
330 REM numero trouve
340 PEN 1:point=point+1:ind1=x:GOSUB 1190:CALL &A631

```

```

350 REM un autre numero
360 PRINT#1,"Recherche d'un autre numero
(O/N) :?":GOSUB 1860:IF j$="o" OR j$="O"
" THEN 300 ELSE GOSUB 8020:GOTO 230
370 REM lecture de page
380 PRINT CHR$(22)CHR$(1)
390 page=1+INT(ind/20):IF page=1+ind/20
THEN page=page-1
400 PRINT#1,page;:PRINT#1," page";:IF pa
ge>1 THEN PRINT#1,"s";
410 PRINT#1," ecrite";:IF page>1 THEN PR
INT#1,"s"
420 LOCATE 40,1:PRINT"No de Page :":LOCA
TE 52,1:INPUT pag1:pag1=FIX(pag1)
430 point=5:GOSUB 8020
440 IF pag1>page OR pag1<=0 THEN PRINT#
1,:GOTO 400
450 IF pag1*20-19<=ind AND ind<=pag1*20
THEN x=ind ELSE x=pag1*20
460 FOR ind1=pag1*20-19 TO x:GOSUB 1190:
point=point+1:NEXT ind1:CALL &A631
480 PRINT#1,:PRINT#1,"page No :";pag1;"
";"Autre page O/N,+1,-1 :?":INK 1,27
490 GOSUB 1860:IF UPPER$(j$)="O" THEN 39
0
500 IF j$="--" THEN pag1=pag1-1:IF pag1<=
0 THEN pag1=pag1+1:GOTO 490 ELSE 430
510 IF j$<>"": THEN GOSUB 8020:GOTO 230
ELSE IF pag1<page THEN pag1=pag1+1:GOTO
430 ELSE 490
540 REM lecture de la fiche
550 OPENIN nfich$:INPUT#9,ind:GOSUB 1910
:FOR x=ind TO 1 STEP -1:INPUT #9,date$(x
),num$(x),obj$(x),ope(x),newa(x),rent$(x
):NEXT x:CLOSEIN:RETURN
570 REM mise a jour
580 GOSUB 2000
590 PRINT#1,"[F1] : INSERTION :?","[F2]
: MODIFICATION :?"," -DEL-"
600 GOSUB 1865:IF j$=CHR$(&7F) THEN RETU
RN ELSE ON ASC(j$)-48 GOTO 870,640:GOTO
600
635 ' modification
640 GOSUB 8020
660 DAT$="[DEL]"
665 point=5
670 TROUVE=0
690 IF point=25 THEN GOSUB 8020:GOTO 665
700 PRINT#1,"[F0]: DATE - [F1]: NUME
RO - [F2]: LIST - [F3]: VALIDE
";DAT$

```

```

720 GOSUB 1865: IF J$=CHR$(&7F) AND TAF=1
  THEN RETURN
730 IF J$=CHR$(&7F) AND TAF=0 THEN 590
740 ON VAL(j$)+1 GOTO 750,765,770,820:GO
TO 720
750 INPUT#1,"DATE : ";DAT$: IF LEN(DAT$)>
9 THEN 750 ELSE 700
765 GOSUB 1370: IF trouve=0 THEN 700 ELSE
  IND1=X:GOSUB 1180: POINT=POINT+1:GOTO 69
0
770 GOSUB 8020:GOSUB 230:GOSUB 8020:GOTO
665
820 IF TROUVE<>1 THEN 690 ELSE IF LEN(re
nt$(ind1))=0 THEN difrent=difrent+ope(in
d1)
840 RENT$(IND1)=DAT$:POINT=POINT-1:GOSUB
1190:POINT=POINT+1:PRINT#1,"MEME OPERAT
ION (O/N) :?":GOSUB 1860:IF UPPER$(j$)=
"O" THEN 670 ELSE IF taf THEN 700 ELSE 5
90
865 'INSERTION
870 PRINT CHR$(22)CHR$(0):GOSUB 8020
890 PEN 1
900 IND1=IND:POINT=5:GOSUB 1190
910 CLS#1:LOCATE 31,1:PRINT"*** Insertio
n ***"
920 POINT=POINT+1:IND=IND+1:IND1=IND:RES
TORE 70:FOR X=1 TO 6:READ F(X):NEXT X:RE
STORE 80:FOR X=1 TO 6:READ G(X):NEXT X:F
OR X=1 TO 6 :LG3=F(X):COL=G(X):GOSUB 123
0:T$(X)=N$
930 IF X<> 4 THEN 950
940 LOCATE 37,POINT:PRINT" ":IF VAL(T$(4
))<>0 THEN LOCATE 37,point:PRINT USING
"#####.##";VAL(T$(4))
950 IF X<>5 THEN 1000
960 LOCATE 48,POINT:PRINT" ";IF VAL(T$(
5))<>0 THEN PRINT USING "#####.##";VA
L(T$(5))
970 ope(ind)=VAL(t$(4))-VAL(t$(5)):newa(
ind)=newa(ind-1)+ope(ind):LOCATE 60,poin
t:PRINT USING "#####.##";newa(ind):dat
e$(ind)=t$(1)
980 t$(2)=UPPER$(t$(2)):IF T$(2)="" THEN
  OSN=OSN+1:NUM$(IND)="OSN"+STR$(OSN) ELS
E num$(ind)=t$(2)
990 obj$(ind)=t$(3)
1000 NEXT X
1010 rent$(ind)=t$(6):CALL &A631:PRINT#1
," Ok (O/N) :?":GOSUB 1860:IF UPPER$(j$)
="O" THEN 1080

```

```

1040 REM pas ok
1050 IF num$(ind)<>t$(2) THEN osn=osn-1
1060 PEN 2:GOSUB 1190:PEN 1:LOCATE 1,point:PRINT CHR$(255):ind=ind-1:point=point-1:GOTO 1090
1070 REM ok
1080 IF rent$(ind)<>" THEN difrent=difrent+ope(ind)
1090 PRINT#1,"une autre ligne (O/N) :?":GOSUB 1860:IF UPPER$(j$)<>"O" THEN 590 ELSE IF point=24 THEN GOSUB 8020:point=4
1110 GOTO 910
1120 REM dessin du cadre ecran complet
1130 PEN 1:CALL &ASE1:INK 1,27
1180 REM AFFICHE EN POINT DATE$(IND1)...RENT$(IND1)
1190 LOCATE 1,POINT:PRINT DATE$(IND1):LOCATE 11,POINT:PRINT NUM$(IND1):LOCATE 21,POINT:PRINT OBJ$(IND1):IF OPE(IND1)<0 THEN LOCATE 48,POINT:PRINT USING "#####.##";ABS(ope(ind1))
1200 IF ope(ind1)>=0 THEN LOCATE 37,point:PRINT USING "#####.##";ABS(ope(ind1))
1210 LOCATE 60,POINT:PRINT USING "#####.##"; NEWA(IND1):LOCATE 71,POINT:PRINT RENT$(IND1):RETURN
1220 REM ENTREE D'UN MESSAGE LG3 EN POINT
1230 NT=0:N$=""
1240 LOCATE COL,POINT:PRINT CHR$(243):LOCATE COL,POINT
1250 GOSUB 1860
1260 IF j$=CHR$(13) THEN LOCATE COL+NT,POINT:PRINT " ":RETURN
1270 IF j$<>CHR$(&7F) THEN 1310
1280 IF NT=0 THEN 1250
1290 NT=NT-1:N$=LEFT$(N$,NT):LOCATE COL+NT,POINT:PRINT CHR$(243);" ":LOCATE COL+NT,POINT:GOTO 1250
1310 NT=NT+1:PRINT j$;IF NT<>LG3 THEN PRINT CHR$(243)
1320 LOCATE COL+NT,POINT:N$=N$+j$:IF NT<LG3 THEN 1250 ELSE RETURN
1360 'recherche de num$
1370 trouve=0
1380 INPUT#1," No a rechercher :";j$:nrech$=UPPER$(j$)
1390 IF LEN(nrech$)>9 THEN 1380
1400 FOR x=ind TO 1 STEP -1

```

```

1410 IF nrech$=num$(x) THEN trouve=1:RET
URN ELSE NEXT x
1420 PRINT#1, nrech$;" ne fait pas parti
e du fichier !!! ( appuyer sur une to
uche )"
1430 GOSUB 1860:PEN 1:RETURN
1440 REM ECRAN INITIAL
1450 MODE 1:LOCATE 5,3:PRINT"*** Gestion
d'un Compte Courant ***":RETURN
1460 REM FERMETURE-OUVERTURE
1470 MODE 1:LOCATE 7,5:PRINT"*** Fermetu
re - Ouverture ***":LOCATE 1,12:PRINT"Nom
du nouveau fichier :";i$;". ....":PRINT
,PRINT" date : .....":PRINT,PRINT
" Avoir :";USING "#####.##";newa
(ind):
1480 point=12:col=25+LEN(I$):lg3=5:GOSUB
1220:c$=i$+LEFT$(n$+" ",5):GOSUB 17
80:IF x THEN LOCATE 1,16:PRINT"Fichier
existant...[ENTER]":GOSUB 1860:RETURN
1500 COL=25:point=13:lg3=9:GOSUB 1230:d$
=n$:LOCATE 1,17:PRINT"Ok (O/N) ? ":GOSUB
1860:LOCATE 1,16:IF UPPER$(j$)<>"O" THE
N RETURN
1540 ind=ind+1:newa(ind)=newa(ind-1):dat
e$(ind)=d$:obj$(ind)="Fermeture":k$=nfic
h$:GOSUB 1880
1550 OSN=1:newa(1)=newa(ind):num$(1)=""
:rent$(1)="" :ind=1:date$(1)=d$:nfich$=c$
:catal$=catal$+nfich$:ope(1)=newa(1):k$=n
fich$:GOSUB 1880
1560 PRINT:PRINT"Nouveau fichier :";nfic
h$
1570 PRINT:GOTO 1905
1580 REM Travail sur un autre fichier.
1590 GOSUB 2010:LOCATE 1,10:PRINT"Fichie
r en cours sauvegarde (O/N) ?":GOSUB 186
0:IF UPPER$(j$)<>"O" THEN RETURN
1620 GOSUB 2010:LOCATE 1,10:INPUT"Nom du
fichier :";n$:c$=LEFT$(n$+" ",8)
1630 GOSUB 1780:IF x<>1 THEN PRINT:PRINT
"Fichier inexistant...[ENTER]":GOSUB 186
0:IF taf THEN 1660 ELSE RETURN
1640 taf=1:k$=nfich$:nfich$=c$:ERASE ope
,newa,date$,num$,obj$,rent$:GOSUB 550
1650 GOSUB 1990:GOSUB 660
1660 GOSUB 2010:LOCATE 1,10:PRINT"1 : Sa
uvegarde du fichier":PRINT"2 : Retour au
menu principal":PRINT"3 : autre fichier
":PRINT"4 : ";nfich$

```

```

1670 GOSUB 1865:ON VAL(j$) GOTO 1680,10,
1620,1650:GOTO 1670
1680 GOSUB 2010:GOSUB 1980:PRINT:PRINT"O
/K :?":GOSUB 1860:IF UPPER$(j$)="O" THEN
j$=nfich$:nfich$=k$:k$=j$:GOSUB 1880:j$
=k$:k$=nfich$:nfich$=j$
1690 GOTO 1660
1700 REM catalogue
1710 MODE 1:LOCATE 12,5:PRINT"*** catalo
gue ***":LOCATE 1,12:FOR x=1 TO LEN(cata
l$)/8:PRINT RIGHT$(LEFT$(catal$,8*x),8),
:NEXT x:PRINT:PRINT:GOTO 1905
1720 REM SAUVEGARDE
1730 GOSUB 1450:GOSUB 1980:LOCATE 1,14:P
RINT"O/K :?":GOSUB 1860:IF UPPER$(j$)<>"
O" THEN RETURN
1750 LOCATE 1,14:PRINT" Mise a jour le
:.....":LOCATE 14,16:PRINT"par :....
:.....":point=14:col=19:lg3=9:GOSUB
1230:dmaj$=n$:point=16:col=19:lg3=15:GO
SUB 1230:majp$=n$:k$=nfich$:PRINT:GOTO 1
880
1770 REM fichier deja existant
1780 x=0:IF LEFT$(catal$,8)=c$ THEN x=1:
RETURN
1790 FOR nt=2 TO LEN(catal$)/8
1800 IF c$=RIGHT$(LEFT$(catal$,x*8),8) T
HEN x=1 ELSE NEXT
1810 RETURN
1840 REM SAUVEGARDE VARIABLES CCP
1850 OPENOUT "var"+i$:WRITE #9,ind,osn,n
fich$,catal$,difrent,dmaj$,majp$:CLOSEOU
T:RETURN
1855 'test appui sur une touche
1860 j$=INKEY$:IF j$="" THEN 1860 ELSE
RETURN
1865 GOSUB 1860:IF ASC(j$)<48 THEN 1865
ELSE RETURN
1870 'sauvegarde
1880 PRINT:PRINT"Inserer -la 1ere disque
tte...[ENTER].":GOSUB 1860:GOSUB 150:IER
A,"*.bak":PRINT" -la 2nd disquet
te...[ENTER].":GOSUB 1860:GOSUB 150:IERA
,"*.bak":PRINT:PRINT"Sauvegarde effectue
e..."
1905 PRINT" (appuyer sur une touche)":G
OSUB 1860:RETURN

```



```

1910 x=50+ind: DIM ope(x): DIM newa(x): DIM
    date$(x): DIM num$(x): DIM obj$(x): DIM re
    nt$(x): RETURN
1975 'parametres
1980 PRINT: PRINT "Nombre d'octets disponi
    bles : "; FRE(""): PRINT: PRINT "L'avoir indi
    que sur le dernier extrait de compte do
    it etre : "; PRINT USING "#####.##"; difr
    ent; PRINT " frs. ": RETURN
1990 'passage en mode 2 et affichage du
    cadre
2000 MODE 2: WINDOW #1,1,80,1,1: WINDOW #2
    ,1,80,5,24: INK 1,0: PEN 1: CALL &A5E1: INK
    1,27: RETURN
2010 MODE 1: LOCATE 3,5: PRINT "*** Travail
    sur un autre fichier ***": RETURN
8000 STOP
8020 'efface texte dans le cadre
8030 INK 1,0: CLS#2: CALL &A631: INK 1,27: R
    ETURN

```

Remarque :

Les différents programmes sont à saisir tels quels surtout le programme OUVCOMPT, dans lequel, même une ligne de REM au tout début perturbera le fonctionnement (en effet la ligne 255, par quelques POKEs appropriés, vient écrire le nom que vous avez choisi pour le programme de gestion (CCP, CE, ...) dans la variable a\$ placée en ligne 10. Vérifiez aussi que vous frappez le nombre d'espaces attendus dans le listing, et méfiez-vous des caractères zéro et O majuscule.

Nous rappelons aussi que cette gestion de compte bancaire fonctionne avec une unité de disquette, l'utilisation d'un lecteur de cassette, par sa lenteur et son accès séquentiel, étant inconcevable pour ce genre de programme.

V. Utilitaires en complément au programme

A. ETAT DU COMPTE

Lorsque vous désirez connaître uniquement l'état de votre compte pour un achat éventuel, il est parfois agaçant de devoir attendre que le logiciel recharge tout le fichier, puis d'attendre l'affichage de différents menus avant de connaître le crédit restant (si vous n'êtes pas à découvert !).

Aussi, nous vous proposons le petit programme dénommé ETAT qui vient rechercher dans VARXXX. (créé par OUVCOMPT) puis dans le dernier fichier de travail, les données nécessaires à la connaissance du dernier avoir.

Ce programme, une fois saisi, se lancera tout simplement par :

RUN «ETAT»

qui vous demandera le nom de votre gestion (CCP, CE, ...) puis vous affichera dans l'instant qui suit l'état de votre compte.

```

10 REM *** etat du compte ***
20 MODE 1
30 LOCATE 3,3
40 PRINT"*** Gestion d'un Compte Courant
   ***"
50 LOCATE 1,12
60 INPUT"Intitule du compte :";i$
70 i$=UPPER$(i$)
80 OPENIN "var"+i$
90 INPUT #9,ind,OSN,nfich$,CATAL$,difren
   t
100 CLOSEIN
110 OPENIN nfich$
120 INPUT #9,date$,num$,obj$,ope,newa
130 CLOSEIN
140 LOCATE 1,15
150 PRINT"Etat du compte : ";USING "
   #####.##";newa;;PRINT" Fra."
160 LOCATE 1,17
170 PRINT"Avoir indique par"
180 PRINT" le dernier releve : ";USING "
   #####.##";difrent;;PRINT" Fra"
190 IF INKEY$="" THEN 190 ELSE CALL 0

```

Le programme état

B. IMPRESSION D'UNE PAGE

Si vous possédez une imprimante mais aucun traitement de texte, nous vous proposons ci-après de modifier le programme afin d'imprimer en cas de besoin une page du fichier de tenue de compte.

Tout d'abord, nous allons modifier la ligne 480 proposant le choix d'impression P :

```

480 PRINT # 1, :PRINT # 1, «page No :»; pag1; «      »; «Autre page
   O/N, +1, -1, P :?» :INK 1,27

```

et ajouter la ligne 505 permettant le branchement à la routine de traitement du choix :

```

505 IF j$ = «P» or j$ = «p» then GOSUB 9000

```

9/10.2

Gestion de logiciels

I - Présentation

Votre bibliothèque de logiciels s'enrichit de jour en jour, et le nombre de disquettes que vous possédez frôle la centaine... Quoi de plus exaspérant que de devoir lire le catalogue de 30 disquettes avant de retrouver le logiciel « Trucmuch », que vous avez écrit il y a un mois, pour en faire une démonstration à un ami !

Une seule solution : un répertoire de logiciels. Sur papier ? A quoi bon posséder un outil de programmation tel que l'Amstrad ? Vous pourrez créer ce répertoire grâce à dBase II, mais le prix de revient serait relativement élevé pour la maigre utilisation que vous feriez face à la puissance de ce logiciel. Une autre solution, la bonne : suivre les conseils qui suivent afin d'écrire soi-même sa gestion de logiciel.

Nous vous proposons donc un programme qui vous permettra de gérer votre stock de logiciels avec quelques similitudes envers dBase II, et de retrouver du premier coup un programme parmi vos disquettes. Nous vous proposerons ensuite la méthode d'adaptation à pratiquement tout ce que vous désirez classer.

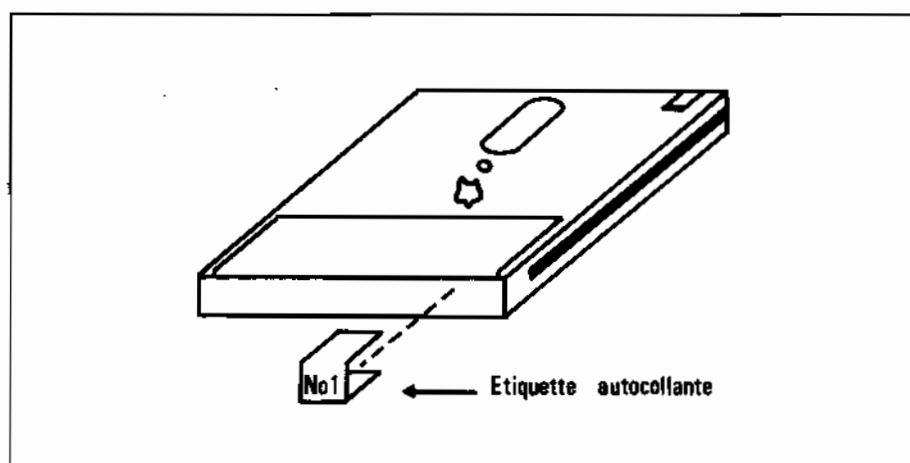
Il nous permettra, de plus, d'étudier une méthode de tri intéressante : le tri par SHELL-METZNER.

Ce programme développé sur Amstrad CPC6128 tourne sur CPC664 et s'adapte sans problème sur CPC464 muni d'un lecteur de disquette (rapelons que la gestion de fichiers se justifie surtout par l'utilisation d'un lecteur de disquette, le lecteur de cassettes, de par sa fiabilité et sa relative lenteur, ne permettant pas un chargement ou une sauvegarde rapide des fichiers gérés).

II - Organisation matérielle et logiciel

LE SUPPORT : LES DISQUETTES

Si vous voulez retrouver rapidement une disquette : une seule solution : le marquage. Vous serez libre du marquage, mais nous vous proposons une méthode simple et rapide : la numérotation algébrique, sur le côté supérieur de la disquette.



Numérotation de disquette.

Vous pouvez maintenant ranger vos disquettes sans leur boîtier, dans une boîte prévue à cet effet (ne craignez pas de vous passer de vos boîtiers, les disquettes sont protégées par une lumière qui s'ouvre uniquement lors de l'insertion dans le lecteur).

ORGANISATION DU LOGICIEL

Les caractéristiques d'un logiciel stocké

Il nous faut d'abord définir les caractéristiques à retenir pour chaque logiciel, afin de définir les différentes zones de chacun.

Les variables seront définies en même temps :

DESI\$: désignation, ou nom, du logiciel

CODE\$: codage personnel

REMA\$: remarque concernant le logiciel

DK\$: référence de la disquette

NBKO\$: taille du logiciel

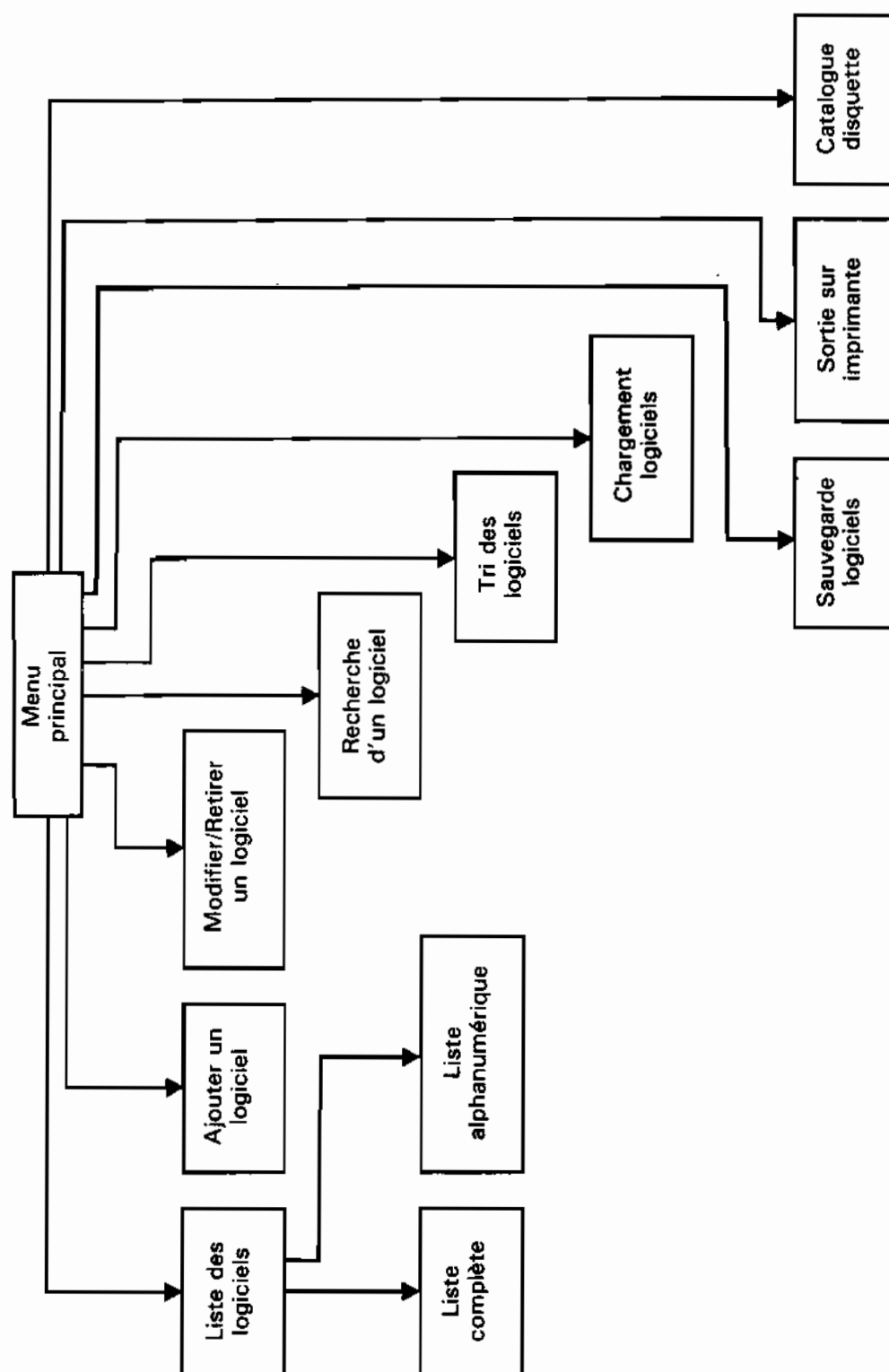
Le menu

Nous vous proposons un menu de neuf options, organisées selon le schéma suivant :

- Liste des logiciels qui sera décomposée en deux options :
 - liste entière
 - liste à partir d'une position alphabétique.
- Ajouter un logiciel.
- Corriger/Supprimer un logiciel (si vous désirez supprimer un logiciel il vous suffira d'entrer le caractère ! à la place de la désignation — le logiciel sera sélectionné par son numéro d'ordre).

- Recherche d'un logiciel par caractéristique.
- Tri alphabétique des logiciels.
- Chargement de la liste.
- Sauvegarde de la liste.
- Sortie sur imprimante.
- Catalogue disquette.

Toutes ces opérations étant définies, nous pouvons dire que le programme fonctionnera sous une structure de type arborescente.



Vous retrouverez ce menu dans la copie d'écran suivante :

PROGRAMME GESTION DE LOGICIELS

MEMOIRE LIBRE : 24

NB DE LOGICIEL : 0

MENU

- (1) LISTE DES LOGICIELS
 - (2) AJOUTER UN LOGICIEL
 - (3) CORRIGER/ELIMINER UN LOGICIEL
 - (4) RECHERCHE PAR CARACTERISTIQUE
 - (5) TRI DES DONNEE
 - (6) CHARGEMENT DU FICHIER LOGICIELS ..
 - (7) SAUVEGARDE DU FICHIER LOGICIELS ..
 - (8) SORTIE SUR IMPRIMANTE
 - (9) CATALOGUE DISQUETTE
- > 6

Copie d'écran menu.

Nous imposerons de plus un retour au menu principal dès qu'une erreur de saisie surviendra.

La saisie ou la correction sera présentée de la même façon que celle de dBase II, c'est-à-dire par champs à remplir, telle la copie d'écran donnée ci-dessous :

ENTRER LE NO DU LOGICIEL A CORRIGER : ? 5

DESIGNATION : CCP..... LOGICIEL No : 5

CODE : 001 PLACE LIBRE : 20 NO

REMARQUE : GESTION DE COMPTES.....

NOMBRE DE NO : 120

DISQUETTE No : 1.....

Les champs de saisie.

Vous pourrez aussi garder une trace écrite de tous vos logiciels, pour vérifier ou tout simplement pour donner à un de vos amis qui pourra vous demander une copie d'un logiciel particulier, après avoir choisi sans semer la zizanie dans le classement de vos disquettes.

	Designation	Cod	Remarques	Ko	No dk
* 1	* AWARI	*	* BASIC	* 2	* 6
* 2	* BIORITM	*	* BASIC	* 5	* 2
* 3	* CALEND	*	* WEKA	* 6	* 3
* 4	* CASSBRIK	*	* BASIC	* 3	* 1
* 5	* CCP	*	* GESTION DE COMPTE	* 12	* 1
* 6	* CHAUFFAG	*	* REGULATION CHAUFFAGE	* 3	* 8
* 7	* DUMP	*	* BASIC WEKA	* 2	* 8
* 8	* HANOI	*	* WEKA	* 6	* 8
* 9	* MORSE	*	* WEKA	* 1	* 6
* 10	* OTHELLO	*	* JEU	* 33	* 6
* 11	* PIP	*	* PIP BASIC WEKA	* 2	* 3
* 12	* PROGRAM	*	* PROGRAMMEUR MEMOIRE	* 3	* 12
* 13	* RECEP232	*	* RECEP. FICHER RS232	* 1	* 4
* 14	* SONNERIE	*	* DETECTEUR SONNERIE	* 1	* 7
* 15	* TAQUIN	*	* WEKA	* 4	* 8
* 16	* WEKASERV	*	* SERVEUR	* 27	* 2

Exemple de liste de logiciels sur imprimante.

III - Le programme

Vous trouverez ci-après le listing du programme qui ne devrait poser aucun problème à la saisie, si ce n'est de ne pas confondre les caractères zéro et O majuscule.

Dans un premier temps, nous vous conseillons de ne pas taper la ligne 110 qui branche le programme à une routine de traitement d'erreur. Vous essaieriez votre programme (après l'avoir sauvegardé) et, après avoir corrigé les éventuelles erreurs de frappe (*Syntax Error*), vérifié sur un fichier d'essai que tout fonctionne correctement, vous pourriez insérer cette ligne et sauvegarder de nouveau le programme.


```

10 REM *****
20 REM ***** GESTION DE LOGICIELS *****
30 REM *****
40 OPENOUT"essai."
50 CLOSEOUT
60 MODE 2
70 INK 0,0:INK 1,26:PEN 1:PAPER 0:BORDER
  0
80 DIM desi$(500),code$(500),rema$(500),
  dk$(500),NBK0$(500)
90 sursauv=1
100 REM ***** MENU *****
110 ON ERROR GOTO 2940
120 CLS
130 LOCATE 1,2:PRINT"PROGRAMME GESTION D
  E LOGICIELS"
140 LOCATE 50,2:PRINT"MEMOIRE LIBRE : "
150 LOCATE 67,2:PRINT(INT(FRE(0)/1024))
160 LOCATE 50,4:PRINT"NB DE LOGICIEL : "
170 FOR I=1 TO 500
180 IF DESI$(I)="" THEN GOTO 200
190 NEXT I
200 LOCATE 67,4:PRINT I-1:tri=(i-1)
210 LOCATE 28,5:PRINT"          MENU "
220 LOCATE 22,7:PRINT"( 1 ) LISTE DES LO
  GICIELS ....."
230 LOCATE 22,9:PRINT"( 2 ) AJOUTER UN L
  OGICIEL ....."
240 LOCATE 22,11:PRINT"( 3 ) CORRIGER/EL
  IMINER UN LOGICIEL ...."
250 LOCATE 22,13:PRINT"( 4 ) RECHERCHE P
  AR CARACTERISTIQUE ...."
260 LOCATE 22,15:PRINT"( 5 ) TRI DES DON
  NEES....."
270 LOCATE 22,17:PRINT"( 6 ) CHARGEMENT
  DU FICHIER LOGICIELS .."
280 LOCATE 22,19:PRINT"( 7 ) SAUVEGARDE
  DU FICHIER LOGICIELS .."
290 LOCATE 22,21:PRINT"( 8 ) SORTIE SUR
  IMPRIMANTE ....."
300 LOCATE 22,23:PRINT"( 9 ) CATALOGUE D
  ISQUETTE ....."
310 CLEAR INPUT
320 A$=INKEY$
330 LOCATE 10,2:PRINT "          "
340 LOCATE 11,2:PRINT CHR$(24);"GESTION"
  ;CHR$(24):IF A$="" THEN 320
350 LOCATE 28,25:PRINT "-----> ";A$
360 IF A$="+" OR A$="&" OR A$="-" OR A$=
  "." THEN GOTO 100
370 A=VAL(A$)

```

```

380 IF A<1 OR A>9 THEN 100
390 ON a GOTO 400,880,1170,1640,2710,1920,2230,2400,2520
400 REM ***** LISTE DES LOGICIELS *****
410 CLS
420 LOCATE 28,5:PRINT"                                MENU LISTE"
430 LOCATE 22,7:PRINT"( 1 ) LISTE ENTIERE ....."
440 LOCATE 22,9:PRINT"( 2 ) LISTE PAR LETTRE ALPHANUMERIQUE .."
450 CLEAR INPUT
460 A$=INKEY$:IF A$="" THEN 460
470 LOCATE 28,25:PRINT "-----> ";A$
480 IF A$="+" OR A$="&" OR A$="-" OR A$="." THEN GOTO 100
490 A=VAL(A$)
500 IF A<1 OR A>2 THEN 100
510 ON a GOTO 520,680
520 REM ***** LISTE ENTIERE *****
530 CLS:GOSUB 2690:A=1
540 FOR I=1 TO 500
550 IF DESI$(I)="" THEN PRINT "*****
***** FIN *****
*****":CALL &
BB06:GOTO 10
0
560 IF A=21 THEN PRINT"                                Ta
per <enter> pour suite, <S> pour stopper
"
570 IF A=21 THEN u$=INKEY$:IF u$="s" OR
u$="S" THEN 100 ELSE IF u$<>CHR$(13) THE
N 570 ELSE A=1:CLS:GOSUB 2690
580 LOCATE 2,A+4:PRINT"*"
590 LOCATE 4,A+4:PRINT DESI$(I)
600 LOCATE 28,A+4:PRINT"*"
610 LOCATE 30,A+4:PRINT NBKD$(I)
620 LOCATE 48,A+4:PRINT"*"
630 LOCATE 50,A+4:PRINT DK$(I)
640 LOCATE 73,A+4:PRINT"*"
650 LOCATE 75,A+4:PRINT I
660 A=A+1
670 NEXT I
680 REM ***** LISTE ALPHANUMERIQUE **
*****
690 CLS:INPUT"ENTREZ LA OU LES LETTRES D
U LOGICIEL A RECHERCHER : ",L$
700 L$=UPPER$(L$)
710 CLS:GOSUB 2690

```

```
720 A=1
730 FOR I=1 TO 500
740 IF DESI$(I)="" THEN PRINT "*****
***** FIN *****
*****":CALL &
BB06:GOTO 10
0
750 IF A=21 THEN CALL &BB06:A=1:CLS:GOSU
B 2690
760 IF LEFT$(DESI$(I),LEN(L$)) <> L$ THE
N 860
770 LOCATE 2,A+4:PRINT"*"
780 LOCATE 4,A+4:PRINT DESI$(I)
790 LOCATE 28,A+4:PRINT"*"
800 LOCATE 30,A+4:PRINT NBKO$(I)
810 LOCATE 48,A+4:PRINT"*"
820 LOCATE 50,A+4:PRINT DK$(I)
830 LOCATE 73,A+4:PRINT"*"
840 LOCATE 75,A+4:PRINT I
850 A=A+1
860 NEXT I
870 CALL &BB06:GOTO 100
880 REM ***** AJOUT D'UN LOGICIEL *
*****
890 sursauv=0
900 FOR I=1 TO 500:IF DESI$(I)="" THEN G
OTO 910 ELSE NEXT I
910 CLS
920 LOCATE 30,3:PRINT"AJOUTER UN LOGICIE
L"
930 GOSUB 2540'GRILLE 1
940 LOCATE 72,7:PRINT I
950 LOCATE 72,10:X=INT((FRE(0)/1000)):PR
INT X;" KO"
960 LOCATE 25,7:LINE INPUT "",DESI$(I)
970 DESI$(I)=UPPER$(DESI$(I))
980 IF DESI$(I)="" THEN GOTO 100
990 IF LEN(DESI$(I))>24 THEN DESI$(I)=""
:GOTO 2630
1000 LOCATE 25,10:LINE INPUT "",CODE$(I)
1010 CODE$(I)=UPPER$(CODE$(I))
1020 IF LEN(CODE$(I))>3 THEN DESI$(I)=""
:GOTO 2660
1030 LOCATE 25,13:LINE INPUT "",REMA$(I)
1040 REMA$(I)=UPPER$(REMA$(I))
1050 IF LEN(REMA$(I))>24 THEN DESI$(I)=""
:GOTO 2630
1060 LOCATE 25,16:LINE INPUT "",NBKO$(I)
1070 NBKO$(I)=UPPER$(NBKO$(I))
1080 IF LEN(NBKO$(I))>3 THEN DESI$(I)=""
```

```

:GOTO 2660
1090 LOCATE 25,19:LINE INPUT "",DK$(I)
1100 DK$(I)=UPPER$(DK$(I))
1110 IF LEN(DK$(I))>24 THEN DESI$(I)="":
GOTO 2630
1120 LOCATE 18,23:PRINT"VALIDATION OUI =
>*ENTERS, NON => *NS"
1130 A$=INKEY$
1140 IF A$=CHR$(13) THEN I=I+1:GOTO 910
1150 IF A$="N" OR A$="n" THEN DESI$(I)="
":a$="ANNULATION":GOSUB 2920:GOTO 910
1160 LOCATE 18,23:PRINT"
":GOTO 1120
1170 REM ***** CORRECTION / ANNULAT
ION D'UN LOGICIEL *****
1180 sur sauve
1190 CLS :INPUT "ENTRER LE NO DU LOGICIE
L A CORRIGER : ";NUM
1200 GOSUB 2540:GOSUB 1210:GOTO 1280
1210 LOCATE 72,7:PRINT NUM
1220 LOCATE 72,10:X=INT((FRE(0)/1024)):P
RINT X;" KO"
1230 LOCATE 25,7:PRINT DESI$(NUM)
1240 LOCATE 25,10:PRINT CODE$(NUM)
1250 LOCATE 25,13:PRINT REMA$(NUM)
1260 LOCATE 25,16:PRINT NBKO$(NUM)
1270 LOCATE 25,19:PRINT DK$(NUM):RETURN
1280 D$=DESI$(NUM)
1290 C$=CODE$(NUM)
1300 R$=REMA$(NUM)
1310 N$=NBKO$(NUM)
1320 NK$=NBKO$(NUM)
1330 DK$=DK$(NUM)
1340 'CORRECTION EFFECTIVE
1350 LOCATE 25,7:INPUT "",DESI$(NUM)
1360 DESI$(NUM)=UPPER$(DESI$(NUM))
1370 IF DESI$(NUM)="" THEN DESI$(NUM)=D$
1380 IF DESI$(NUM)="!" THEN GOTO 1600
1390 IF LEN(DESI$(NUM))>24 THEN GOTO 263
0
1400 LOCATE 25,10:LINE INPUT "",CODE$(NU
M)
1410 CODE$(NUM)=UPPER$(CODE$(NUM))
1420 IF CODE$(NUM)="" THEN CODE$(NUM)=C$
1430 IF LEN(CODE$(NUM))>3 THEN CODE$(NUM
)="" :GOTO 2660
1440 LOCATE 25,13:LINE INPUT "",REMA$(NU
M)
1450 REMA$(NUM)=UPPER$(REMA$(NUM))
1460 IF REMA$(NUM)="" THEN REMA$(NUM)=R$
1470 IF LEN(REMA$(NUM))>24 THEN REMA$(NU

```

```

M="":GOTO 2630
1480 LOCATE 25,16:LINE INPUT "",NBK$(NUM)
1490 NBK$(NUM)=UPPER$(NBK$(NUM))
1500 IF NBK$(NUM)="" THEN NBK$(NUM)=NK$
1510 IF LEN(NBK$(NUM))>3 THEN NBK$(NUM)="" :GOTO 2660
1520 LOCATE 25,19:LINE INPUT "",DK$(NUM)
1530 DK$(NUM)=UPPER$(DK$(NUM))
1540 IF DK$(NUM)="" THEN DK$(NUM)=DK$
1550 IF LEN(DK$(NUM))>24 THEN DK$(NUM)="" :GOTO 2630
1560 CLS:LOCATE 30,3:PRINT"VERIFICATION"
1570 GOSUB 2540:GOSUB 1210
1580 CALL &B806::GOTO 100
1590 REM *** SUPPRESSION D'UN LOGICIEL **
*
1600 FOR J = NUM+1 TO I
1610 DESI$(J-1) = DESI$(J)
1620 NEXT J
1630 GOTO 100
1640 REM ***** CARACTERISTIQUES D'UN LOGICIEL *****
1650 CLS:LOCATE 30,3:PRINT"RECHERCHE PAR
:"
1660 LOCATE 22,9:PRINT"( 1 ) LA DESIGNATION ....."
1670 LOCATE 22,11:PRINT"( 2 ) LE NUMERO ....."
1680 LOCATE 22,13:PRINT"( 3 ) LE NOMBRE DE KD ....."
1690 LOCATE 22,15:PRINT"( 4 ) RETOUR AU MENU PRINCIPAL ....."
1700 FOR I=1 TO 100:A$=INKEY$:NEXT I
1710 A$=INKEY$:IF A$="" THEN 1710
1720 LOCATE 28,17:PRINT "-----> ";A$
1730 IF A$="+" OR A$="&" OR A$="-" OR A$="," THEN GOTO 1640
1740 A=VAL(A$)
1750 IF A<1 OR A>4 THEN 1640
1760 ON A GOTO 1770,1840,1870,100
1770 REM ***** RECHERCHE PAR DESIGNATION *****
1780 CLS:INPUT "ENTRER LE NOM DU LOGICIEL A TROUVER : ",RECHERCHE$
1790 RECHERCHE$=UPPER$(RECHERCHE$)
1800 IF RECHERCHE$="" THEN 100
1810 FOR I=1 TO 500
1820 IF RECHERCHE$=DESI$(I) THEN NUM=I:G

```

```

OSUB 2540:GOSUB 1210:CALL &BB06
1830 NEXT I:GOTO 100
1840 CLS:INPUT "ENTRER LE NUMERO DU LOGI
CIEL A TROUVER : ",RECHERCHE$
1850 I=VAL(RECHERCHE$)
1860 NUM=I:GOSUB 2540:GOSUB 1210:CALL &B
B06:GOTO 100
1870 CLS:INPUT "ENTRER LE NOMBRE DE KO :
",RECHERCHE$
1880 FOR I=1 TO 500
1890 IF RECHERCHE$=NBKO$(I) THEN NUM=I:G
OSUB 2540:GOSUB 1210:CALL &BB06
1900 NEXT I
1910 GOTO 100
1920 REM***** CHARGEMENT DU FICHIER
*****
1930 IF SURSAUV=0 THEN GOSUB 2190
1940 IF SURSAUV=1 THEN 2030 ELSE IF UPPE
R$(REP$) <> "0" THEN GOTO 100
1950 FOR I=1 TO 500
1960 DESI$(I)=" "
1970 CODE$(I)=" "
1980 REMA$(I)=" "
1990 NBKO$(I)=" "
2000 DK$(I)=" "
2010 NEXT I
2020 PRINT FRE("")
2030 CLS
2040 PRINT"NOM DU FICHIER A ";CHR$(24);"
CHARGER";CHR$(24)
2050 LOCATE 54,1:LINE INPUT"",REP$
2060 IF REP$="" THEN 100
2070 CLS:LOCATE 10,10:PRINT"CHARGEMENT E
N COURS."
2080 OPENIN REP$
2090 FOR I=1 TO 500:LOCATE 20,20:PRINT i
2100 INPUT#9,DESI$(I):IF DESI$(I)=" " THE
N 2160
2110 INPUT#9,CODE$(I)
2120 INPUT#9,REMA$(I)
2130 INPUT#9,DK$(I)
2140 INPUT#9,NBKO$(I)
2150 NEXT I
2160 CLOSEIN
2170 SURSAUV=1
2180 GOTO 100
2190 CLS:PRINT"FICHIER EN MEMOIRE MODIFI
E!"
2200 PRINT:PRINT"EFFACE LE FICHIER?(O/N)
"

```

```
2210 LINE INPUT "",REP$
2220 RETURN
2230 REM ***** SAUVEGARDE DE FICHIER
*****
2240 CLS
2250 PRINT"NOM DU FICHIER A ";CHR$(24);"
SAUVEGARDER";CHR$(24)
2260 LOCATE 54,1:LINE INPUT "",REP$
2270 IF REP$="" THEN 100
2280 CLS:LOCATE 10,10:PRINT"SAUVEGARDE E
N COURS."
2290 OPENOUT REP$
2300 FOR I=1 TO 500:LOCATE 20,20:PRINT i
2310 WRITE#9,DESI$(I)
2320 WRITE#9,CODE$(I)
2330 WRITE#9,REMA$(I)
2340 WRITE#9,DK$(I)
2350 WRITE#9,NBKO$(I):IF DESI$(I)="" THE
N 2370
2360 NEXT I
2370 CLOSEOUT
2380 SURSAUV = 1
2390 GOTO 100
2400 REM ***** SORTIE IMPRIMANTE *
*****
2410 CLS:LOCATE 10,10:PRINT"SORTIE SUR I
MPRIMANTE EN COURS."
2420 REP = INP(&F500)
2430 IF REP <> 30 THEN 3000
2440 PRINT#8,CHR$(27);CHR$(64);CHR$(27);
CHR$(15)
2450 PRINT #8,TAB(10);"Designation";TAB
(36);"Cod";TAB (42);"Remarques";TAB (69)
;"Ko";TAB(75);"No dk":PRINT #8
2460 FOR I=1 TO 500
2470 IF DESI$(I)="" THEN LOCATE 15,12:PR
INT"FIN D'IMPRESSION ...":FOR XX=1 TO 20
00:NEXT XX:PRINT #8,CHR$(12):GOTO 100
2480 PRINT #8,TAB(1);" * ";TAB(2);i;TAB(7)
" * ";TAB(10);DESI$(I);TAB (33);" * ";CO
DE$(I);TAB (39);" * ";REMA$(I);TAB (66);
" * ";NBKO$(
I);TAB(72);" * ";DK$(I)
2490 NEXT I
2500 PRINT#8, CHR$(12);CHR$(27);CHR$(64)
2510 GOTO 100
2520 '***** CATALOGUE
DISQUETTE *****
2530 MODE 2:CAT:CALL &BBO6:GOTO 100
2540 '***** GRILLE ECR
```

```

AN *****
2550 LOCATE 3,7:PRINT"DESIGNATION .....
      : ";CHR$(24);".....":
PRINT CHR$(24)
2560 LOCATE 3,10:PRINT"CODE .....
      . : ";CHR$(24);"...":PRINT CHR$(24)
2570 LOCATE 3,13:PRINT"REMARQUE .....
      . : ";CHR$(24);".....":
      :PRINT CHR$(24)
2580 LOCATE 3,16:PRINT"NOMBRE DE KO ....
      . : ";CHR$(24);"...":PRINT CHR$(24)
2590 LOCATE 3,19:PRINT"DISQUETTE No ....
      . : ";CHR$(24);".....":
      :PRINT CHR$(24)
2600 LOCATE 50,7:PRINT"LOGICIEL No .....
      . : ";CHR$(24);".":PRINT CHR$(24)
2610 LOCATE 50,10:PRINT"PLACE LIBRE ....
      .. : ";CHR$(24);".":PRINT CHR$(24)
2620 RETURN
2630 '***** LONGUEUR D
ES DONNEES TROP GRANDES *****
2640 CLS:PRINT"***** TROP LONG,
PAS PLUS DE 24 CARACTERES !!! *****
****"
2650 CALL &BB06:GOTO 100
2660 '***** LONGUEUR D
ES DONNEES TROP GRANDES *****
2670 CLS:PRINT"***** TROP LONG,
PAS PLUS DE 3 CARACTERES !!! *****
***"
2680 CALL &BB06:GOTO 100
2690 '***** ENTETE *****
2700 LOCATE 1,1:PRINT"LISTE DES LOGICIEL
S":LOCATE 3,3:PRINT "DESIGNATION":LOCATE
30,3:PRINT "NOMBRE DE K.O":LOCATE 50,3:
PRINT "No DE
LA DISQUETTE":RETURN
2710 '***** tri
par shell-metznar *****
2720 SURSAUV=0
2730 m=tri
2740 garde=tri
2750 CLS:LOCATE 10,10:PRINT"TRI EN COURS
, PATIENCE ... "
2760 p=m
2770 p=INT(p/2)
2780 IF p<1 THEN GOTO 100
2790 deb=1:fin=m-p
2800 r=deb
2810 c=r+p
2820 IF c<garde THEN LOCATE 20,20:PRINT
c:garde=c

```



```

2830 IF desi$(r)<=desi$(c) GOTO 2890
2840 aa$=desi$(r):desi$(r)=desi$(c):desi$(c)=aa$
2850 sauv2$=code$(r):sauve3$=rema$(r):sauve4$=dk$(r):sauve5$=nbko$(r)
2860 code$(r)=code$(c):rema$(r)=rema$(c):dk$(r)=dk$(c):nbko$(r)=nbko$(c)
2870 code$(c)=sauv2$:rema$(c)=sauve3$:dk$(c)=sauve4$:nbko$(c)=sauve5$
2880 r=r-p:IF r>0 GOTO 2810
2890 deb=deb+1:IF deb>fin THEN GOTO 2770
      ELSE GOTO 2800
2900 GOTO 100
2910 REM ***** ANNULATION SONORE *****
2920 FOR i=1 TO LEN(a$):PRINT MID$(a$,i,1);CHR$(143);CHR$(8);:SOUND 1,20,3,15,,10:FOR x=1 TO 30:NEXT:NEXT:PRINT " ":RETURN
2930 '*****
2940 REM ***** ERREUR SUR L'UNITE DE DISQUETTE *****
2950 PRINT:PRINT:LOCATE 30,20
2960 PRINT CHR$(7);CHR$(24);"ERREUR DISQUETTE !!!";CHR$(24)
2970 FOR I=1 TO 2000 : NEXT I
2980 PRINT ERR;ERL
2990 RESUME 100
3000 REM ***** TESTE LA PRESENCE DE L'IMPRIMANTE SUR LE PORT D'ADRESSE &F500 (ON LINE) *****
3010 PRINT:PRINT:PRINT
3020 PRINT CHR$(7);CHR$(24);"IMPRIMANTE NON CONNECTEE";CHR$(24);" !!!"
3030 PRINT CHR$(7);CHR$(24);"      OU PAS DE PAPIER      ";CHR$(24);" !!!"
3040 FOR I = 1 TO 2000:NEXT I
3050 GOTO 100

```

- Lignes 40 et 50 : ces lignes permettent de réserver le buffer de l'unité de disquette.
- Ligne 80 : réserve un tableau possible de 500 logiciels à gérer (si vous n'en avez pas assez, faites donc un nettoyage par le vide des logiciels que vous n'utilisez certainement plus).
- Ligne 90 : création d'une variable de vérification de la sauvegarde d'un fichier logiciel avant le chargement d'un nouveau.
- Lignes 100 à 350 : affichage du MENU et acquisition à la volée du choix.
- Ligne 110 : permet d'appeler un sous-programme de traitement d'erreur sur l'unité de disquette (mauvais format du nom de fichier, fichier inexistant, oubli de disquette...).
- Lignes 360 à 390 : testent la validité du choix et effectuent le branchement.
- Lignes 400 à 470 : affichage du menu LISTE et attente du choix LISTE ENTIERE ou LISTE ALPHANUMERIQUE : pour effectuer un choix par liste alphanumérique, il vous suffira de frapper la ou les premières lettres du ou des logiciels recherchés et le programme gèrera la recherche.
- Lignes 520 à 670 : affichage de la liste complète des logiciels, avec possibilité de sortir de la liste à la fin de l'affichage d'une page (vous vous laisserez guider par les instructions).
- Lignes 680 à 870 : recherche par ordre alphanumérique (les logiciels dont le nom commence par un chiffre sont aussi pris en compte).
- Lignes 880 à 1160 : cette partie de programme vous propose un écran composé de champs que vous remplirez un à un (la validation d'un champ s'effectue par la touche <RETURN>). Lorsque vous n'aurez plus de logiciels à insérer, il vous suffira de valider par <RETURN> sur le champ DESIGNATION qui restera vide.
- Lignes 1170 à 1630 : permettent la modification d'une caractéristique sur un logiciel. Si vous entrez le caractère ! sur le champ DESIGNATION, le logiciel sera irrémédiablement éliminé du fichier. Par contre lors de la modification d'une caractéristique, le simple fait de frapper <RETURN> sur le champ où se trouve le curseur laisse ce champ inchangé. Si vous voulez effectuer une modification, il faut le faire sur tout le champ.
- Lignes 1640 à 1910 : cette portion de programme permet d'effectuer des recherches selon trois caractéristiques :
 - la désignation,
 - le numéro d'ordre,
 - la capacité du logiciel (exemple : si vous voulez optimiser les contenus de vos disquettes, et qu'il vous reste 15 K-octets sur une disquette, vous pouvez rechercher tous les logiciels ayant cette taille).

— Lignes 1920 à 2220 : permettent de charger un de vos fichiers logiciels en vous demandant le nom (vous pourrez séparer le fichier utilitaire du fichier jeu, nous ne saurons que trop vous le recommander). Une vérification est auparavant effectuée sur la variable **SURSAUV**, afin de savoir si le logiciel précédemment en mémoire a été modifié, a été sauvegardé. Si ce n'est pas le cas, le logiciel vous demande confirmation (saut à la routine placée en 2190 à 2220) de l'effacement du fichier actuellement en mémoire.

Par contre si le fichier doit être effacé, la place mémoire libérée doit être récupérée pour être affichée (ceci est fait habituellement par le programme et n'a pas besoin d'être géré par le programmeur, sauf dans notre cas car nous affichons la mémoire restant), ce qu'effectue la ligne 2020 par l'instruction **PRINT FRE (" ")**.

— Lignes 2230 à 2390 : sauvegarde du fichier en demandant à l'utilisateur le nom.

— Lignes 2400 à 2510 : possibilité d'imprimer le fichier sur imprimante. De plus, lorsque vous effectuez une sortie sur imprimante, et que celle-ci n'est pas connectée, il est très déconcertant d'attendre, et être obligé d'effectuer un « Break ». Le programme teste donc en ligne 2430 si l'imprimante est bien connectée par lecture en ligne 2420 de l'état du port Centronics placé à l'adresse &F500 (si le contenu de ce port est différent de 30 décimal alors l'imprimante ne répond pas présente — bouton « ON LINE » non appuyé — imprimante non allumée ou non connectée...). Si tout est correct, l'impression est réalisée en caractères condensés (CHR\$(27);CHR\$(15) ligne 2440), après réinitialisation de l'imprimante (CHR\$(27);CHR\$(64)). En fin d'impression, l'imprimante est de nouveau ré-initialisée pour une utilisation sur un autre logiciel par exemple (ligne 2500).

— Lignes 2520 à 2530 : affichage du catalogue disquette sans avoir à retourner sous Basic.

— Lignes 2540 à 2620 : ce sous-programme permet d'afficher à l'écran les champs de caractéristiques des logiciels utilisés par les choix **CORRECTION** et **AJOUTER** (appels en lignes 930 et 1200).

— Lignes 2630 à 2650 : traitement d'une erreur d'entrée sur un champ de 24 caractères.

— Lignes 2660 à 2680 : traitement d'une erreur d'entrée sur un champ de trois caractères.

— Lignes 2690 et 2700 : affichage de l'en-tête de présentation lors de l'affichage d'une liste de logiciels.

— Lignes 2710 à 2900 : tri des logiciels par ordre alphanumérique sur la désignation. Le principe de ce tri sera expliqué au paragraphe suivant.

— Lignes 2910 et 2920 : annulation sonore d'une entrée non confirmée (appel en ligne 1150).

— Lignes 2930 à 2990 : traitement de l'erreur sur l'unité de disquette par affichage en vidéo-inversée, et retour au menu (par l'instruction **<RESUME 100>**).

— Lignes 3000 à 3050 : traitement de la non-présence de l'imprimante et retour au menu.

Remarque :

Vous aurez certainement aperçu l'instruction `CALL &BB06`, qui effectue un branchement à un vecteur de la RAM (placé à cette adresse). Ce vecteur envoie le programme à une routine qui attend la frappe d'une touche sur le clavier. Dès que l'on appuie sur une touche, le Basic a de nouveau la main et continue son traitement. Cette routine n'affecte aucune variable du programme et peut être utilisée sans risque dans n'importe quel programme nécessitant un arrêt provisoire, redémarrant par l'appui d'une touche. Vous pouvez essayer cette commande en mode direct, vous verrez alors le curseur disparaître et ne réapparaître que lorsque vous appuierez sur une touche quelconque.

IV - Le principe de tri

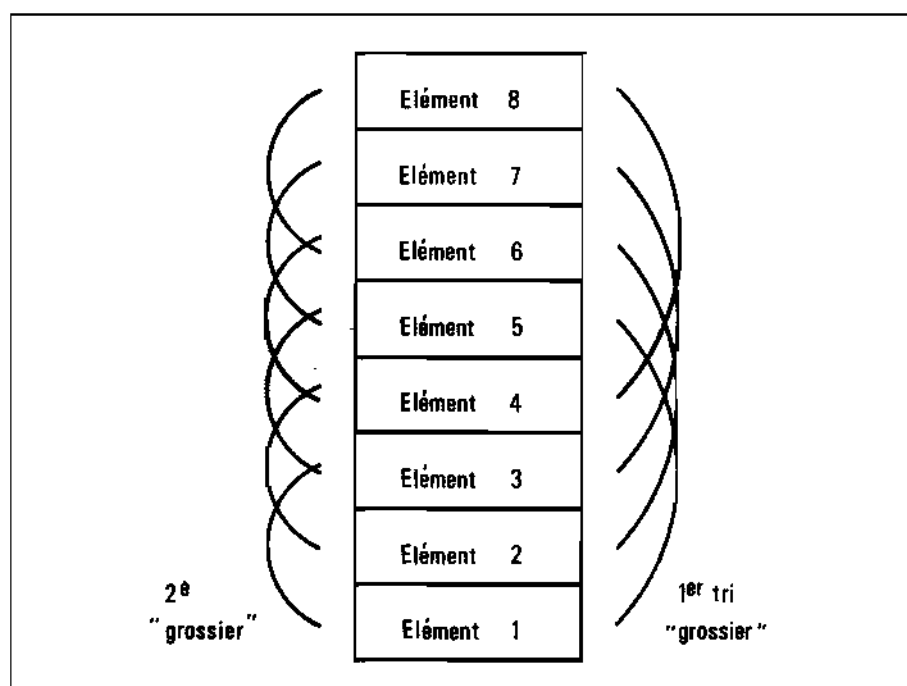
Ce sous-programme est celui du tri par la méthode de Shell améliorée.

Le principe du tri de Shell consiste à diviser une première fois l'ensemble à trier en deux, puis à comparer les éléments distants de la moitié du nombre d'éléments jusqu'à atteindre le dernier élément de la liste, et à les permuter si le cas se présente (selon un tri croissant ou décroissant).

Quand les éléments ont été une première fois triés grossièrement de cette façon, on divise de nouveau le nombre d'éléments et on effectue le tri des éléments distants de ce nouveau nombre. Et ainsi de suite jusqu'à ne plus pouvoir diviser l'ensemble des éléments à trier.

Si le nombre trouvé lors de la division par deux ne tombe pas juste, on prendra la valeur directement inférieure comme distance des éléments à trier.

Prenons exemple sur un ensemble de huit éléments que l'on veut classer par ordre croissant de bas en haut.



Exemple de tri « Shell-Metzner » sur 8 éléments.

Une première division par deux nous donne 4 comme distance entre les éléments. Nous comparerons d'abord l'élément 1 et l'élément 5 ($1 + 4 = 5$). Si l'élément 5 est inférieur à l'élément 1, ils seront permutés. Puis on compare l'élément 2 et l'élément 6, une permutation sera effectuée si nécessaire. On fera de même pour les éléments 3 et 7 ainsi que pour les éléments 4 et 8.

Ensuite la valeur 4 est divisée en deux, ce qui va permettre de trier les éléments distants de 2.

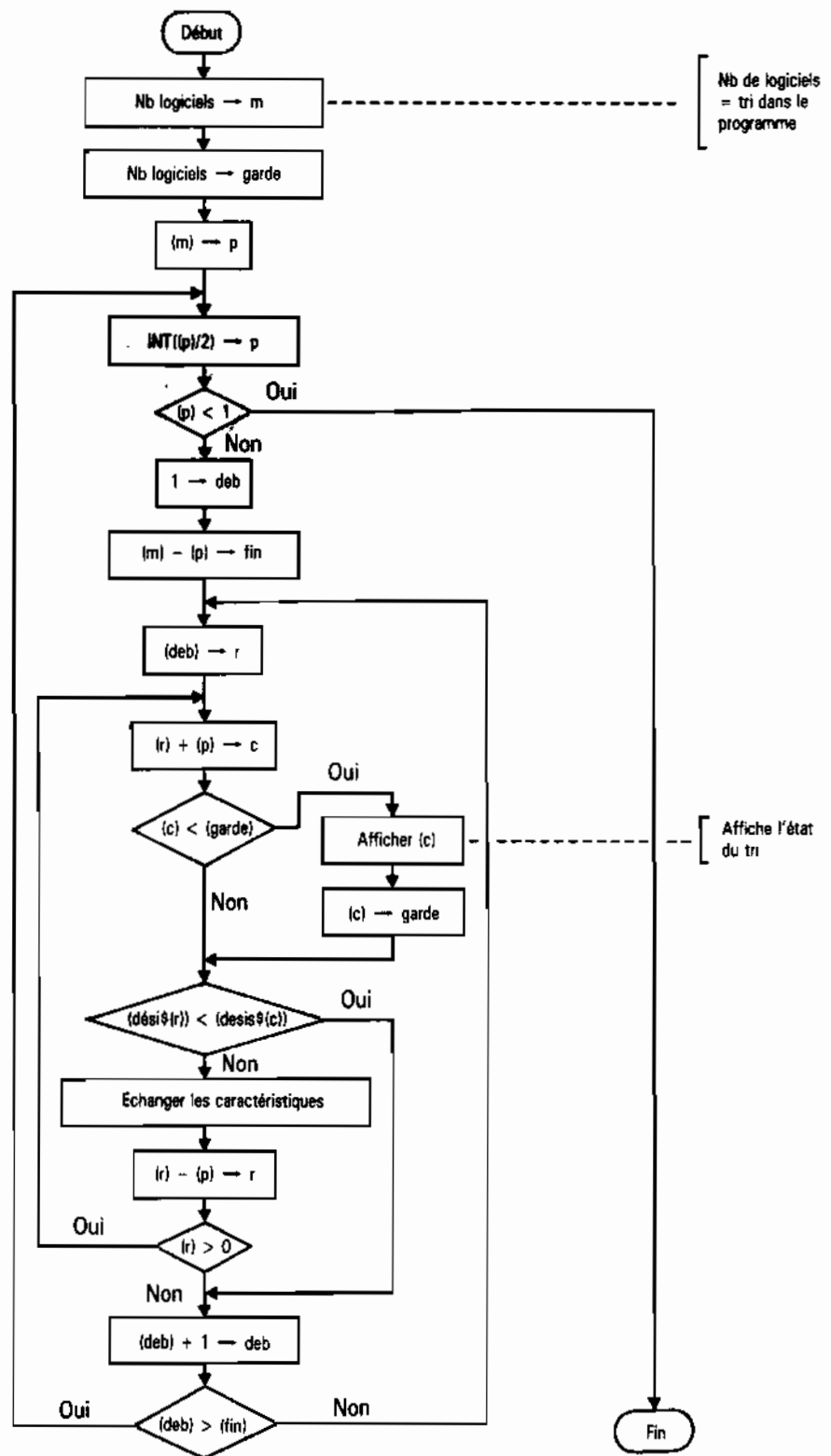
L'élément 1 est comparé avec l'élément 3, permuté si nécessaire, puis l'élément 2 et l'élément 4, ... etc.

Une nouvelle division par deux du précédent écart nous donne la distance UN (dernier écart possible) des éléments à trier.

Ce dernier tri effectué, tous les éléments de l'ensemble seront triés comme il l'était désiré.

Cette méthode de tri est de loin beaucoup plus rapide que la méthode du tri classique dit à bulle, que l'on trouvera décrite dans le chapitre concernant les applications du lecteur de disquette Vortex 5 pouces 1/4 (Voir Partie 8, chapitre 6.1).

L'algorithme du programme de tri utilisé dans la gestion de logiciels vous est donné ci-dessous :



Nous vous signalons que la notation **(m)** signifie que l'on utilise le contenu de la variable **m** pour effectuer le traitement.

De plus le signe $\rightarrow$ est le signe d'affectation, c'est-à-dire, dans l'exemple **(m) - (p) $\rightarrow$ fin**, que l'on affecte à la variable **fin** la valeur de la différence des contenus de **m** et de **p**. Cette formulation s'écrit plus simplement en Basic, mais moins rigoureusement, de la façon suivante : **fin = m - p**.

Vous retrouverez le programme transcrivant cet algorithme en Basic entre les lignes 2730 et 2890 (voir pages 14 et 15).

— La ligne 2750 n'a pas d'intérêt majeur dans le tri, ainsi que la ligne 2820 qui permet d'afficher l'état du tri (la distance entre les éléments triés), qui permet de faire patienter l'utilisateur en le rassurant sur un non-ratage du tri.

— Les lignes 2840 à 2870 assurent les permutations des éléments ne répondant pas au critère de comparaison de la ligne 2830.

V. Modification du programme

Nous vous proposons par exemple de modifier le programme pour qu'il convienne à la gestion de votre discothèque ou de votre bibliothèque.

Votre imagination aidant, vous pourrez élaborer les caractéristiques intéressantes de vos livres ou disques (nom de l'auteur, titre, nombre de pages...), pour remplir la liste décrite au paragraphe sur les caractéristiques d'un logiciel stocké.

Pour modifier le programme, il suffira de remplacer les caractéristiques, et le mot **LOGICIEL** par les noms que vous aurez choisis dans les lignes suivantes :

— 13	-	160	-	220 à 280	-	690	-	920	-
— 1190	-	1680	-	1780	-	1840	-	1870	-
— 2450	-	2550 à 2600	-	2700	-		-		-

9/11

Traitement de texte

Nous vous présentons un logiciel de traitement de texte pleine page très perfectionné. Ce logiciel est totalement écrit en assembleur. Vous découvrirez petit à petit comment fonctionne le cœur du traitement de texte (saisie de texte sur un écran en pleine page avec une gestion de scrolling haut, bas, droite et gauche, modes remplacement et insertion, gestion de blocs, insertion/écriture d'une partie du texte, entrée sur disquette, gestion d'un spooler d'imprimante, etc.).

9/11.1

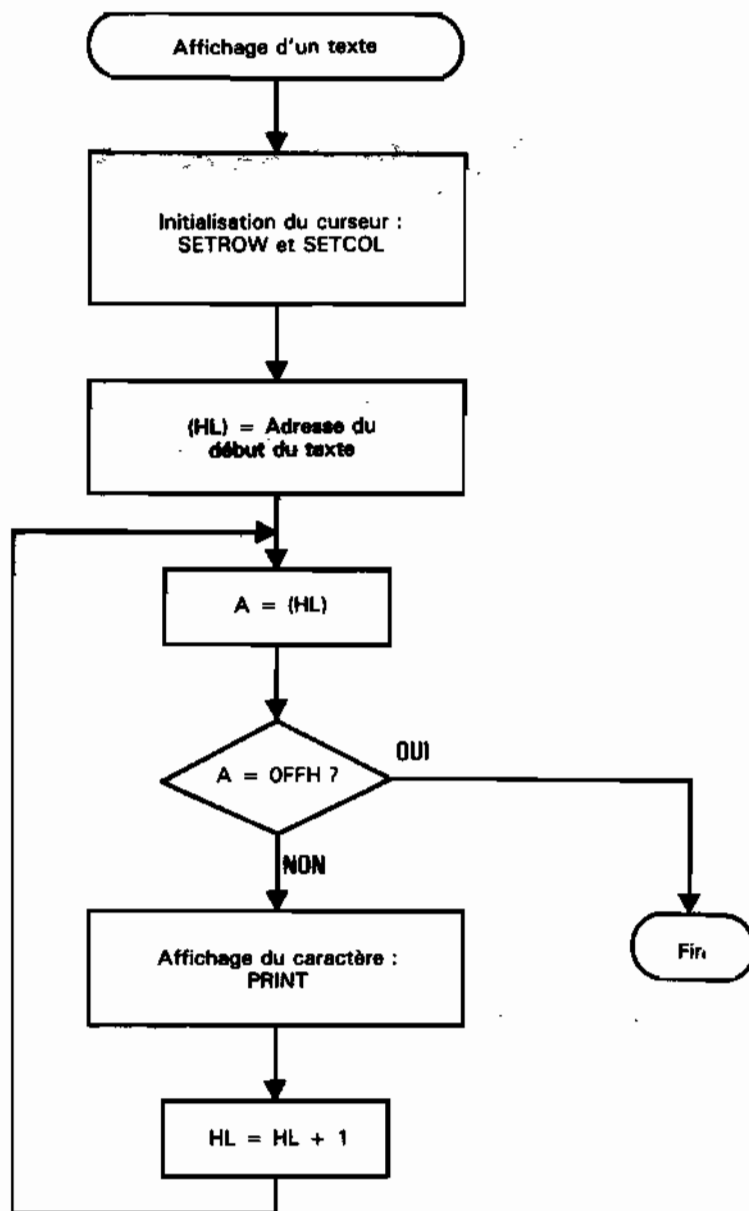
Mise en œuvre d'utilitaires

Avant de traiter le logiciel de traitement de texte à proprement parler, nous allons mettre en œuvre plusieurs utilitaires d'ordre général.

- Le premier utilitaire permet d'afficher un texte alphanumérique à un endroit quelconque sur l'écran.

Cet utilitaire servira à afficher tous les messages d'information (comme par exemple les messages indiquant à l'utilisateur d'entrer un nom de fichier ou une chaîne à rechercher dans le texte), et la ligne d'état en bas de l'écran.

La logique mise en œuvre pour réaliser cet utilitaire est la suivante :



Les routines du Firmware utilisées par cet utilitaire sont les suivantes :

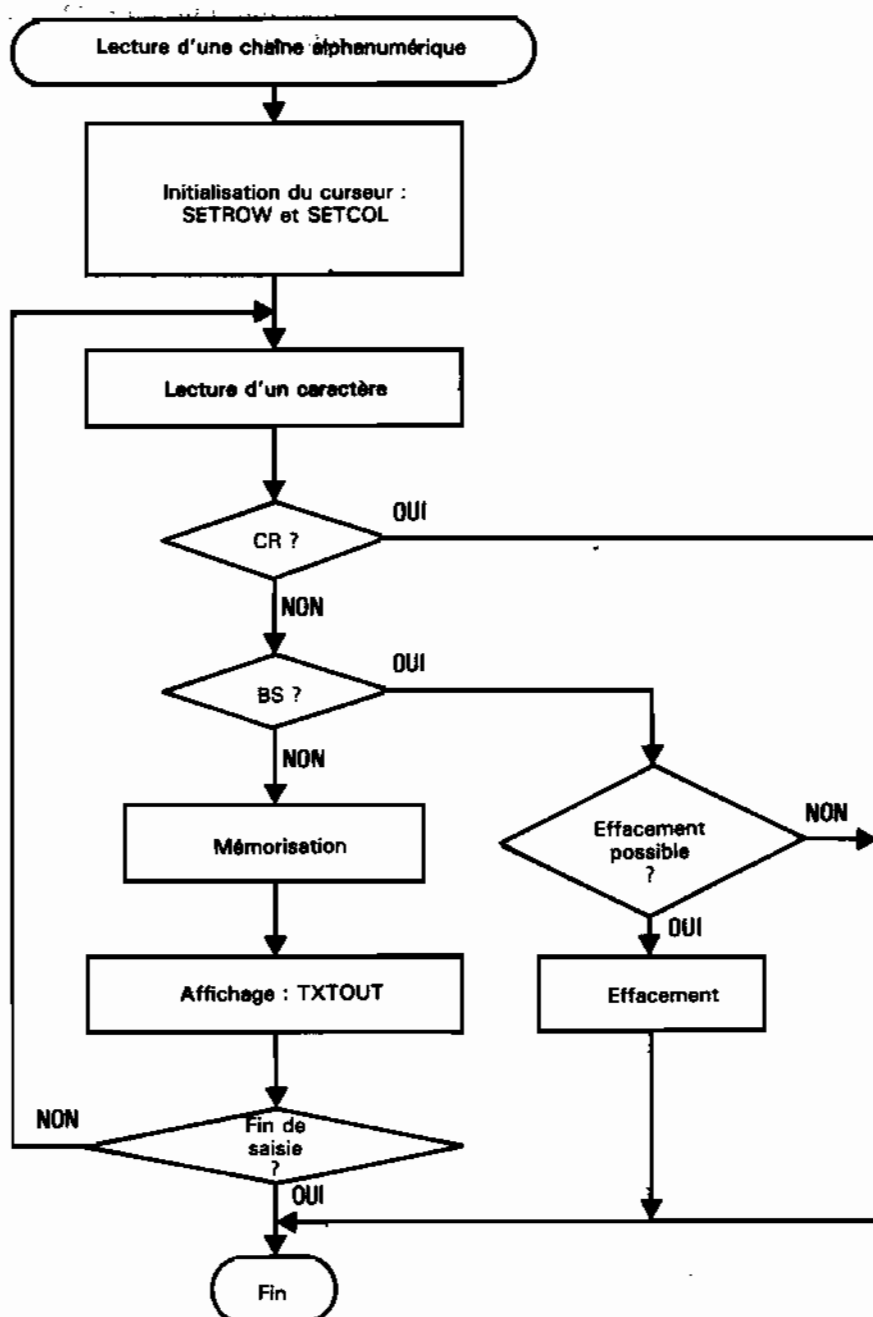
- 0BB72H: TXT SET ROW
(Positionnement de la ligne du curseur)
- 0BB6FH: TXT SET COLUMN
(Positionnement de la colonne du curseur)

Reportez-vous à la Partie 4, chapitre 2.7 page 78, pour avoir des détails à ce sujet.

- Le second utilitaire permet de lire une chaîne de caractères tapée au clavier. La longueur de la chaîne est paramétrable, et la touche < Del > (effacement du caractère à gauche du curseur) est prise en compte.

Cet utilitaire servira à lire toutes les données entrées par l'utilisateur suite aux messages d'information affichés par le traitement de texte (comme par exemple les messages indiquant à l'utilisateur d'entrer le nom d'une chaîne à rechercher dans le texte).

La logique mise en œuvre pour réaliser cet utilitaire est la suivante :



Les routines du Firmware utilisées par cet utilitaire sont les suivantes :

- 0BB72H: TXT SET ROW
(Positionnement de la ligne du curseur)
- 0BB6FH: TXT SET COLUMN
(Positionnement de la colonne du curseur)
- 0BB06H: KM WAIT CHAR
(Lecture d'un caractère au clavier)
- 0BB5AH: TXT OUTPUT
(Affichage d'un caractère, y compris le caractère de contrôle)
- 0BB5DH: TXT WR CHAR
(Affichage d'un caractère, sauf caractère de contrôle)

Pour montrer comment fonctionnent ces deux utilitaires, nous avons réalisé une courte démonstration (appelée « PROGRAMME PRINCIPAL » dans le programme ci-dessous).

Elle consiste à afficher un texte sur l'écran en ligne 1, colonne 1, et à saisir un texte de longueur comprise entre 1 et 6 caractères. Le texte entré se trouve dans le buffer « BUF » à partir de l'adresse &H908C.

Le listing Assembleur de ces deux utilitaires est le suivant :

```

1                                ORG 9000H
2                                LOAD 9000H
3 9000 C3AB90                    JP EXEC ;Execution
4                                FIN: EQU $
5 9003 00                        NOP
6                                ;
7                                ;-----
8                                ; Affichage d'un texte à l'ecran
9                                ;-----
10                               ;Entree: HL=pointeur memoire
11                               ; DE=Ligne,Colonne
12                               ;-----
13                               .AFFICH: EQU $
14 9004 E5                        PUSH HL
15 9005 F5                        PUSH AF

```

```

16 9006 7A          LD  A,D
17 9007 CD72BB      CALL SETROW          ;Init ligne
18 900A 7B          LD  A,E
19 900B CD6FBB      CALL SETCOL          ;Init colonne
20 900E F1          POP  AF
21 900F E1          POP  HL
22 9010 E5          PUSH HL
23 9011 F5          PUSH AF          ;Init et sauveg registre
24                AT1: EQU  $          ;Boucle d'affichage
25 9012 7E          LD  A,(HL)
26 9013 FEFF        CP   OFFH          ;Delimiteur?
27 9015 2B06        JR   Z,AT2          ;Oui=>Fin affichage
28 9017 CD5ABB      CALL PRINT          ;Aff caractere
29 901A 23          INC  HL
30 901B 1BF5        JR   AT1
31                AT2: EQU  $
32 901D F1          POP  AF
33 901E E1          POP  HL
34 901F C9          RET
35                ;
36                ;
37                ;-----
38                ; Lecture de n caracteres
39                ;-----
40                ;Entree: B=Nbre de caract a lire
41                ;      HL=Pointeur buffer
42                ;      DE=Ligne,Colonne
43                ;Sortie: Buf=buffer de lecture
44                ;-----
45                LECTN: EQU  $
46 9020 3AB990      LD  A,(PX)

```

```

47 9023 328B90      LD    (PDX),A          ;Pos depart en X
48 9026 E5          PUSH HL
49 9027 7A          LD    A,D
50 9028 CD72BB      CALL SETROW          ;Init ligne
51 902B 7B          LD    A,E
52 902C CD6FBB      CALL SETCOL          ;Init colonne
53 902F E1          POP    HL
54 9030 21BC90      LD    HL,BUF          ;Adresse du buffer
55                LO:    EQU    4          ;Boucle principale
56 9033 CD06BB      CALL WAIT.          ;Lect 1 caractere
57 9036 FE0D        CP    CRLF
58 903B 2B26        JR    Z,L1
59 903A FE7F        CP    BS
60 903C 2B23        JR    Z,L2
61                ; Caractere normal
62 903E E5          PUSH HL          ;Sauveg HL
63 903F 32AA90      LD    (CAR),A      ;Memo du caractere
64 9042 3AB990      LD    A,(PX)
65 9045 5F          LD    E,A
66 9046 1600        LD    D,0          ;DE=Depl dans buffer
67 904B 1B          DEC    DE
68 9049 19          ADD    HL,DE
69 904A 3AAA90      LD    A,(CAR)
70 904D 77          LD    (HL),A      ;Stockage caract
71 904E C5          PUSH BC
72 904F CD5DBB      CALL TXTOUI        ;Affichage caractere
73 9052 C1          POP    BC
74 9053 E1          POP    HL          ;Restit registres
75 9054 3AB990      LD    A,(PX)
76 9057 3C          INC    A

```

```

77 9058 328990      LD    (FX),A
78 905B 05           DEC    B                ;Nb caract a saisir-1
79 905C 78           LD    A,B
80 905D B7           OR     A
81 905E 20D3         JR     NZ,L0            ;On continue
82                L1:   EQU    $                ;Sortie du S/P
83 9060 C9           RET
84                L2:   EQU    $                ;
85 9061 3ABE90      LD    A,(PDX)
86 9064 C601         ADD    A,1
87 9066 4F           LD    C,A
88 9067 3A8990      LD    A,(PX)
89 906A B9           CP     C
90 906B 38C6         JR     C,L0            ;BS impossible
91 906D 3A8990      LD    A,(PX)
92 9070 3D           DEC    A
93 9071 328990      LD    (PX),A
94 9074 C5           PUSH   BC
95 9075 E5           PUSH   HL            ;BC,HL detruits par SETCOL
96 9076 CD6FBB      CALL   SETCOL
97 9079 3E2D         LD    A,2DH          ;Caract -
98 907B CD5DBB      CALL   TXTOUT        ;Affich caract -
99 907E 3A8990      LD    A,(PX)
100 9081 CD6FBB      CALL   SETCOL        ;Repos colonne
101 9084 E1          POP     HL
102 9085 C1          POP     BC            ;Restit registres
103 9086 04          INC     B
104 9087 18AA        JR     L0            ;Suite boucle
105                ;-----
106                ;Zone des EQU
107                ;-----

```

```

108      CRLF:      EQU   13              ;Code CR
109      BS:        EQU   127            ;Code BS
110      ;
111      WAIT:       EQU   0BB06H         ;KM WAIT CHAR
112      PRINT:      EQU   0BB5AH         ;TXT OUTPUT
113      TXTOUT:     EQU   0BB5DH         ;TXT WR CHAR
114      SETCOL:     EQU   0BB6FH         ;TXT SET COLUMN
115      SETROW:     EQU   0BB72H         ;TXT SET ROW
116      ;
117      ;-----
118      ;Zone des DS
119      ;-----
120      PX:         DS    1              ;Position en X
121      PY:         DS    1              ;Position en Y
122      PDX:        DS    1              ;Pod depart en X
123      BUF:        DS    30            ;Buffer de lecture
124      CAR:        DS    1              ;Stock tempo 1 caract
125      ;
126      ;-----
127      ; PROGRAMME PRINCIPAL
128      ;-----
129      EXECL:      EQU   %
130 90AB 110101      LD    DE,101H        ;Ligne,Colonne
131 90AE 21C990      LD    HL,TEXIE       ;Pointeur texte a aff
132 90B1 CD0490      CALL AFFICH
133      ;
134 90B4 3E16        LD    A,22
135 90B6 32B990      LD    (PX),A         ;Abcisse saisie
136 90B9 3E05        LD    A,5
137 90BB 32BA90      LD    (PY),A         ;Colonne saisie
138 90BE 0606        LD    B,6            ;Lgr texte saisi

```

```

139 90C0 111601      LD    DE,116H      ;Ligne,Colonne
140 90C3 CD2090      CALL LECTN      ;Lecture
141 90C6 C30390      JP     FIN
142                  ;
143                  ;
144 90C9 456E7472 TEXTE: DB    "Entrez 6 caracter
144 90CD 657A2036
144 90D1 20636172
144 90D5 61637465
144 90D9 7265733A
145 90DD 202D2D2D      DB    "  ----"
145 90E1 2D2D2D
146 90E4 FF          DB    0FFH
147                  END

```


Le chargeur Basic de ce même programme est le suivant :

```

1000 REM -----
1010 REM Boucle de lecture des donnees et memorisation
1020 REM -----
1030 FOR I=&4000 TO &90E4
1040   READ A$
1050   A=VAL("&H"+A$)
1060   POKE I,A
1070 NEXT I
1080 REM -----
1090 REM Appel des S/F Assembleur
1100 REM -----
1110 MODE 1
1120 CALL &9000
1130 END
1140 REM -----
1150 REM Donnees
1160 REM -----
1170 DATA C3,AB,90,C9,E5,F5,7A,CD,72,BB,7B,CD,6F,BB,F1,E1
1180 DATA E5,F5,7E,FE,FF,2B,6,CD,5A,BB,23,1B,F5,F1,E1
1190 DATA C9,3A,B9,90,32,BB,90,E5,7A,CD,72,BB,7B,CD,6F,BB
1200 DATA E1,21,8C,90,CD,6,BB,FE,D,2B,26,FE,7F,2B,23,E5
1210 DATA 32,AA,90,3A,B9,90,5F,16,0,1B,19,3A,AA,90,77,C5
1220 DATA CD,5D, BB,C1,E1,3A,B9,90,3C,32,B9,90,5,7B,B7,20
1230 DATA D3,C9,3A,BB,90,C6,1,4F,3A,B9,90,B9,3B, C6,3A,B9
1240 DATA 90,3D,32,B9,90,C5,E5,CD,6F,BB,3E,2D,CD,5D,BB,3A
1250 DATA B9,90,CD,6F,BB,E1,C1,4,1B,AA,0,0,0,0,0,0
1260 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1270 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,11,1,1,21
1280 DATA C9,90,CD,4,90,3E,16,32,B9,90,3E,5,32,BA,90,6
1290 DATA 6,11,16,1,CD,20,90,C3,3,90,45, 6E,74,72,65,7A
1300 DATA 20,36,20,63,61,72,61,63,74,65,72,65,73,3A,20,2D
1310 DATA 2D,2D,2D,2D,2D,FF,0,0,0,0,0,0,0,0,0,0

```

Utilisez ce chargeur plutôt que le code assembleur présenté pour passer moins de temps à saisir le programme. Les performances des deux versions du programme sont totalement identiques puisque le chargeur Basic se contente de placer le code binaire du programme Assembleur en mémoire et d'exécuter le code binaire.

Vous avez sans doute remarqué que chaque ligne de DATA contient 16 codes hexadécimaux. Cela n'est pas un hasard. De cette manière, vous pourrez vérifier rapidement que les codes que vous avez tapés sont bien conformes à ceux du livre. Pour cela, utilisez le programme de checksum (voir Partie 9, chap. 8.4) et comparez le résultat de son exécution avec les données suivantes :

64 3A 55 0A B9 C2 98 44 F3 00 FD 30 98 2C B4

9/11.2

Le traitement de texte Weka

Nous avons vu comment créer un traitement de texte élémentaire en turbo Pascal dans la Partie 4, chapitre 4.5.4. Ce traitement de texte aurait pu être amélioré dans de larges mesures si la mémoire des CPC était plus étendue. Comme ce n'est pas le cas, nous allons voir comment créer un traitement de texte pleine page totalement écrit en Assembleur.

9/11.2.1

Fonctions élémentaires

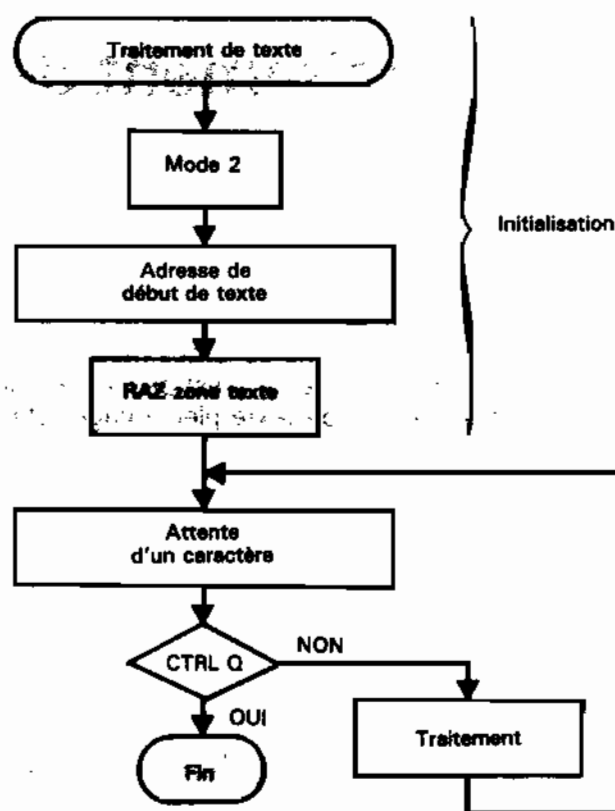
I - Saisie, affichage et mémorisation de caractères sur l'écran en pleine page

Le programme est implanté en &8000. Il débute par une série d'initialisations :

- Passage en mode 2 grâce à la macro **SCR SET MODE** du Firmware ;
- Mémorisation de l'adresse d'implantation du début du texte dans la variable **DEBTEX** ;
- Initialisation de la zone de stockage du texte à 32D (caractère espace).

Ces initialisations sont suivies d'une boucle d'attente principale dans laquelle le programme acquiert les caractères tapés au clavier. S'il s'agit de caractères affichables, il les affiche à la position courante du curseur et les mémorise. S'il s'agit de caractères de contrôle valides, il active le sous-programme de traitement correspondant. S'il s'agit de l'appui simultané sur **Ctrl Q**, le programme est avorté.

La logique apparaît clairement dans l'ordinogramme suivant :

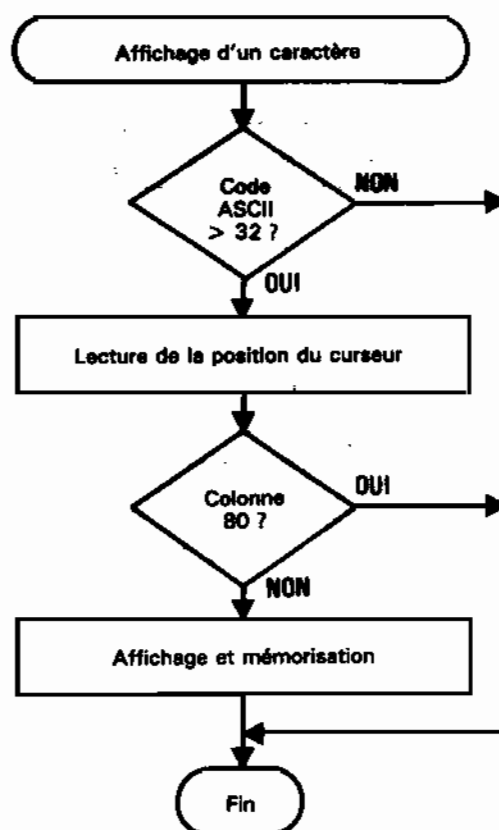


Examinons en détails les diverses actions effectuées par le programme.

- Lorsque le caractère tapé a un code ASCII supérieur ou égal à 32, il est affiché à la position courante du curseur.
- L'appui sur la touche <Enter> provoque un passage à la ligne suivante, éventuellement suivi d'un scrolling vers le haut de l'écran.
- L'appui sur une des touches flèche provoque le déplacement contrôlé du curseur dans la direction voulue, éventuellement suivi d'un scrolling vers le haut ou vers le bas.
- L'appui simultané sur Ctrl 2 et Z provoque un scrolling de l'écran vers le haut.
- L'appui simultané sur Ctrl et W provoque un scrolling de l'écran vers le bas.
- La touche permet d'effacer le caractère à gauche du curseur et déplace les caractères qui suivent d'une position vers la gauche.
- La touche <Clr> permet d'effacer le caractère courant et déplace les caractères qui suivent d'une position vers la gauche.

Affichage d'un caractère

Tout caractère de code ASCII supérieur ou égal à 32 est affiché et mémorisé selon la logique de l'ordinogramme suivant :



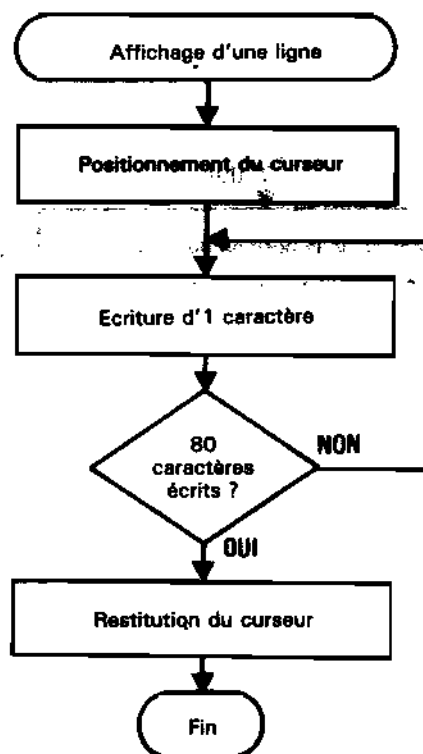
Le sous-programme chargé de cette tâche a pour nom AFFICH.

Affichage d'une ligne

L'affichage d'une ligne consiste :

- au positionnement du curseur sur l'écran,
- en l'affichage de 80 caractères consécutifs à partir de la position initiale du curseur.

Ces actions s'enchaînent comme suit :



Le sous-programme chargé de cette tâche a pour nom WRILIG.

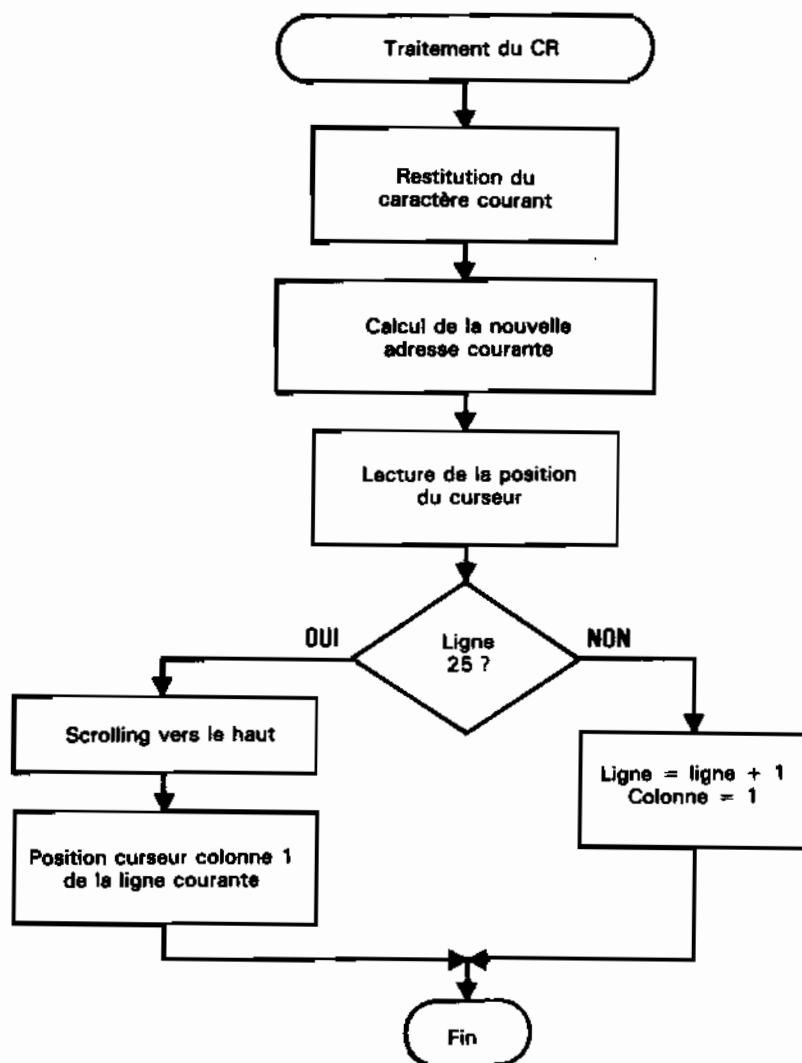
Appui sur la touche <Enter>

Lorsque l'utilisateur appuie sur la touche <Enter>, le caractère courant (qui est affiché en inverse vidéo) est réaffiché avec un attribut normal. L'adresse courante du début de la nouvelle ligne est calculée.

Si le curseur se trouve sur la ligne 25, un scrolling haut est effectué et le curseur est déplacé au début de la nouvelle ligne 25.

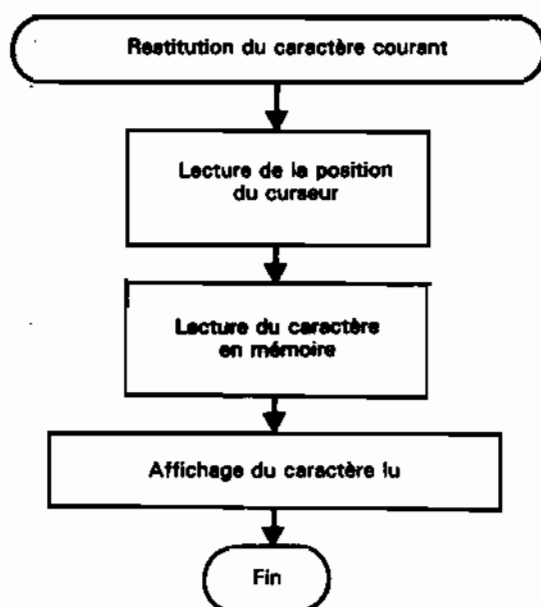
Dans le cas contraire, le curseur est simplement déplacé au début de la ligne suivante.

Ces actions s'enchaînent comme suit :



Le sous-programme chargé de cette tâche a pour nom TCR.

TCR fait appel à un sous-programme de restitution de caractère qui permet d'afficher le caractère courant avec un attribut d'affichage normal. Ce sous-programme a pour nom RESTIT. La logique qu'il met en œuvre est la suivante :



II - Gestion du curseur (haut, bas, droit et gauche)

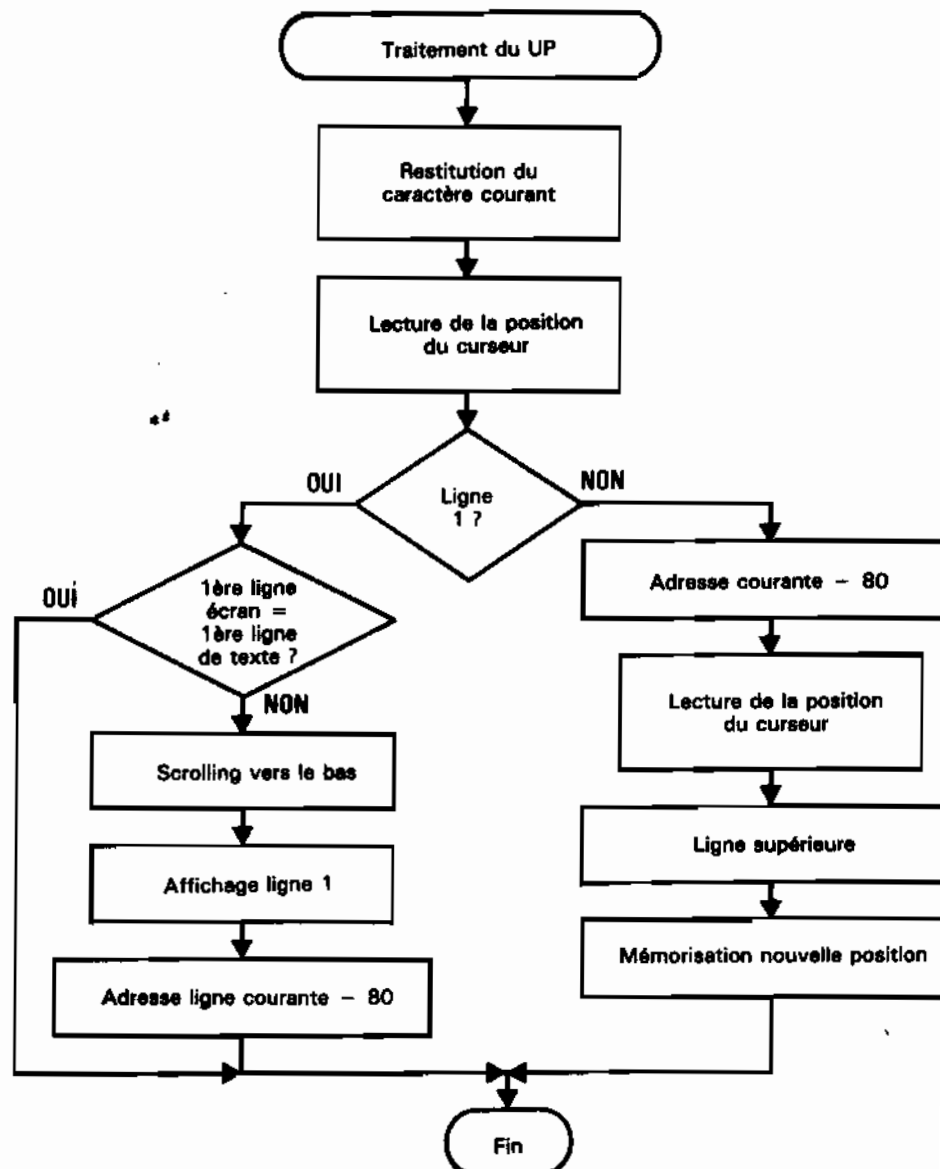
Appui sur UP

Lorsque l'utilisateur appuie sur la touche flèche **UP**, le caractère courant est restitué avec un attribut normal (pas inverse vidéo).

Si le curseur était positionné sur la première ligne de texte, aucune action n'est effectuée.

Dans le cas contraire, un scrolling d'une ligne vers le bas est effectué si nécessaire et la nouvelle position du curseur est mémorisée.

Ces actions s'enchaînent comme suit :



Le sous-programme chargé de cette tâche a pour nom TUP.

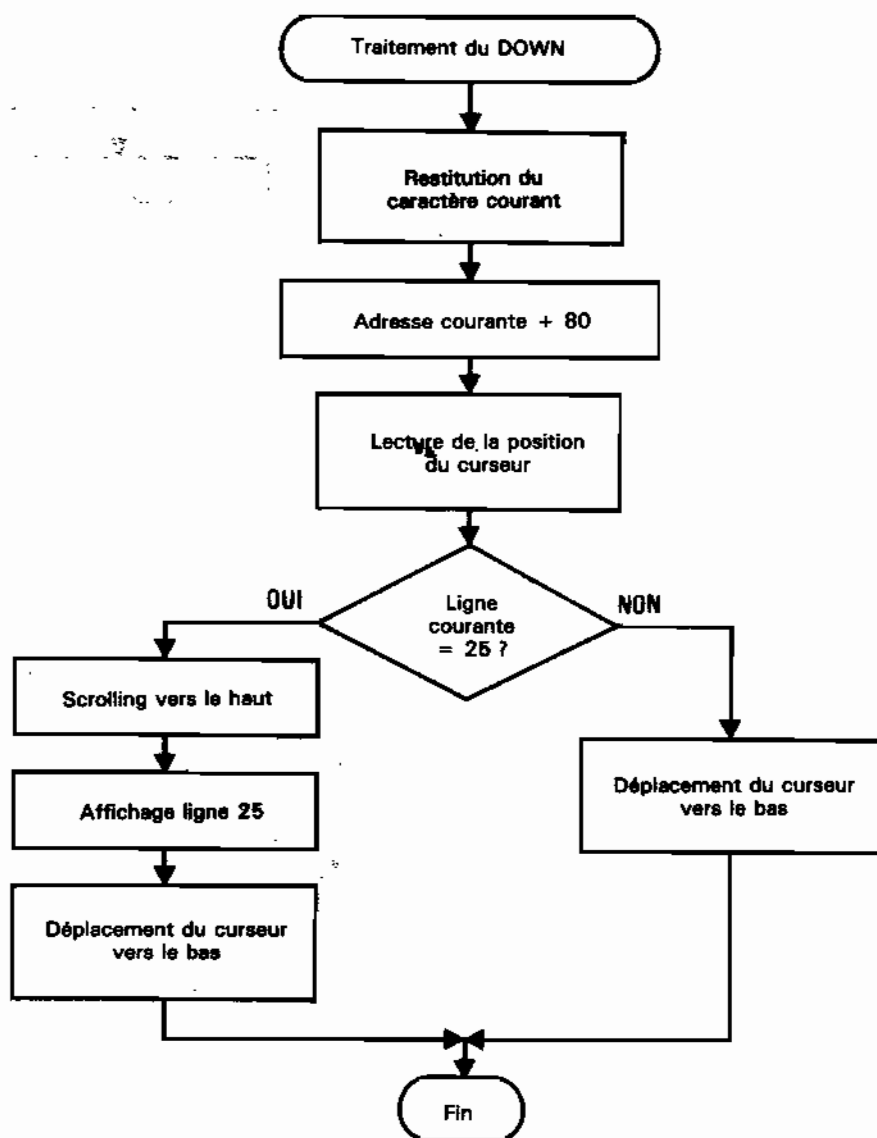
Appui sur DOWN

Lorsque l'utilisateur appuie sur la touche flèche **DOWN**, le caractère courant est restitué avec un attribut normal (pas inverse vidéo).

Si le curseur était positionné sur la dernière ligne de l'écran, un scrolling d'une ligne vers le haut est effectué, et la nouvelle ligne 25 est affichée.

Si la ligne courante n'est pas la 25^e, le curseur est simplement déplacé d'une position vers le bas.

Ces actions s'enchaînent comme suit :



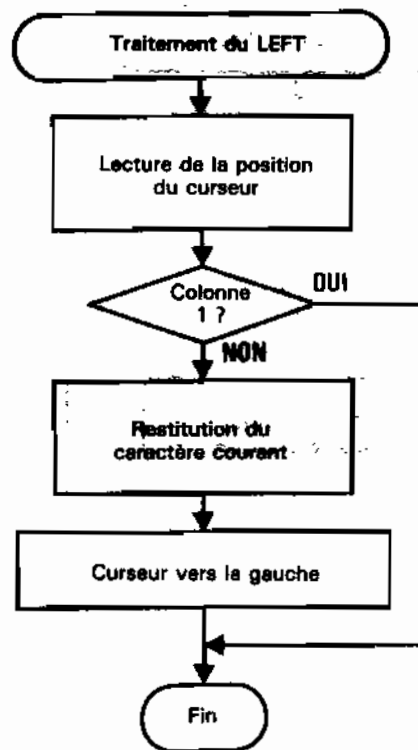
Le sous-programme chargé de cette tâche a pour nom TDOWN.

Appui sur LEFT

Lorsque la touche flèche **LEFT** est pressée, le curseur se déplace d'une position vers la gauche dans le cas où sa position colonne courante est différente de 1.

Dans le cas contraire, aucune action n'est effectuée.

Ces actions s'enchaînent comme suit :



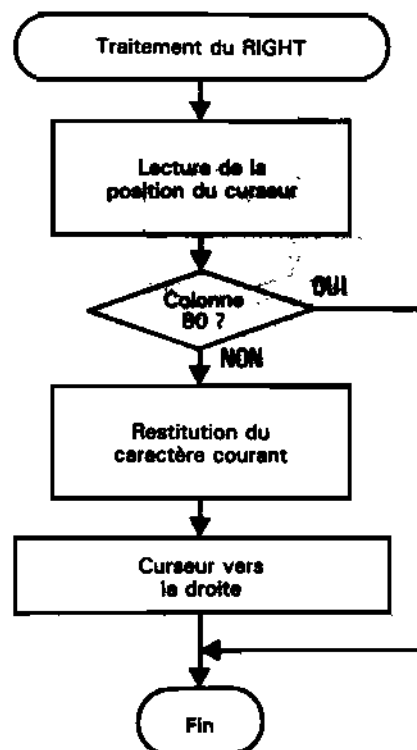
Le sous-programme chargé de cette tâche a pour nom TLEFT.

Appui sur RIGHT

Lorsque la touche flèche **RIGHT** est pressée, le curseur se déplace d'une position vers la droite dans le cas où sa position colonne courante est différente de 80.

Dans le cas contraire, aucune action n'est effectuée.

Ces actions s'enchaînent comme suit :



Le sous-programme chargé de cette tâche a pour nom TRIGHT.

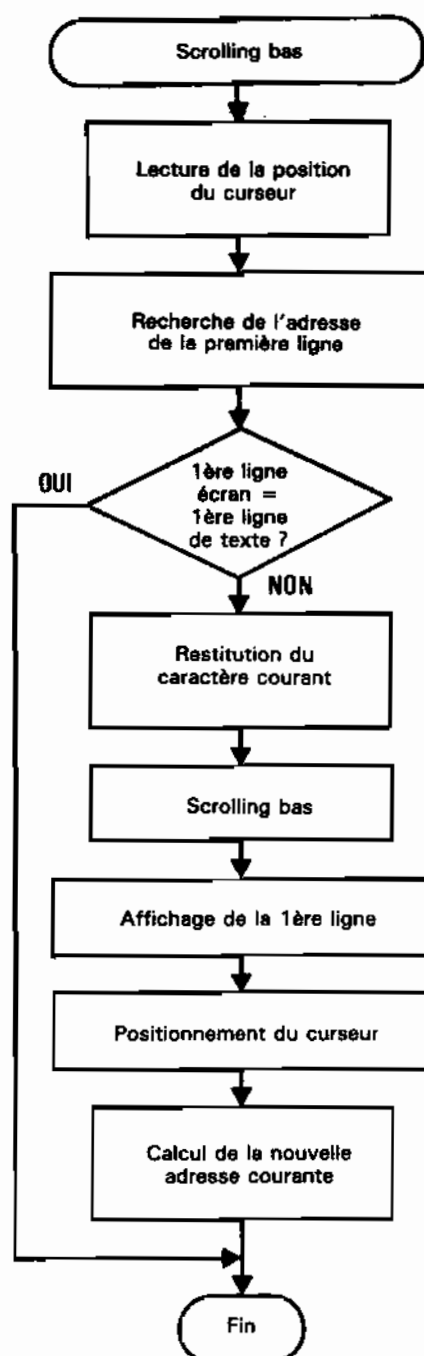
III - Commandes de scrolling bas et haut d'une ligne d'écran

Scrolling bas

Lorsque les touches **Ctrl** et **W** sont pressées simultanément, l'écran tout entier est déplacé vers le bas d'une ligne (si cela est possible). Pour ce faire, le programme calcule l'adresse du premier caractère de la ligne supérieure de l'écran. Si cette adresse est celle du début du texte, le scrolling ne peut être effectué.

Dans le cas contraire, le caractère courant est restitué (attribut d'affichage normal), un scrolling vers le bas est effectué et une nouvelle première ligne est affichée.

Ces actions s'enchaînent comme suit :

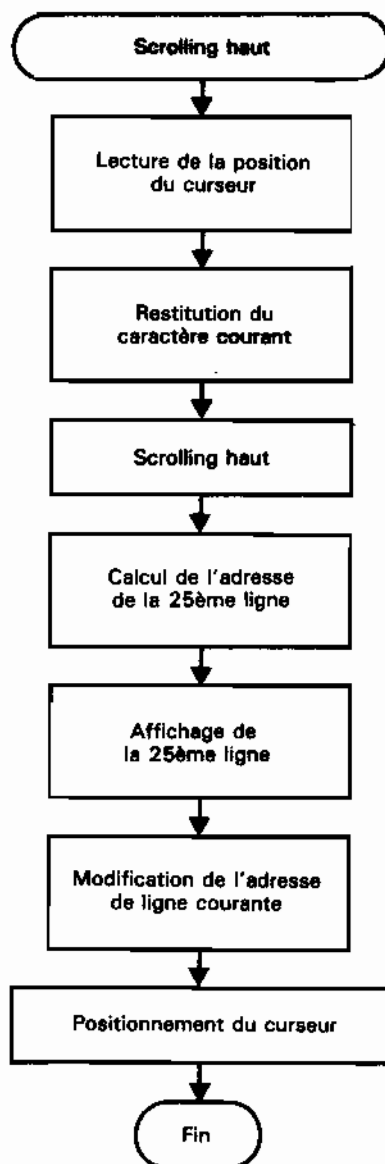


Le sous-programme chargé de cette tâche a pour nom BSCROL.

Scrolling haut

Un scrolling vers le haut est toujours possible. Il s'effectue lorsque l'utilisateur appuie simultanément sur les touches **Ctrl** et **Z**. Le caractère courant est restitué avec un attribut d'affichage normal, l'adresse de début de 25^e ligne est calculée et la 25^e ligne affichée.

Ces actions s'enchaînent comme suit :



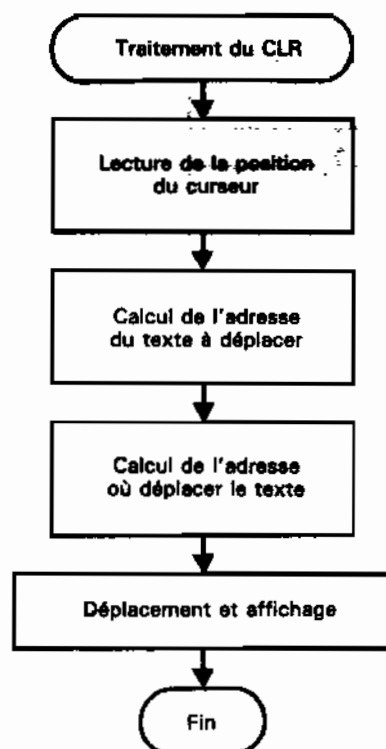
Le sous-programme chargé de cette tâche a pour nom HSCROL.

IV - Touches CLR et DEL pour l'effacement d'un caractère

Appui sur la touche <Clr>

Lorsque l'utilisateur appuie sur la touche <Clr>, le caractère courant est effacé et les caractères qui suivent sont déplacés d'une position vers la gauche. Pour ce faire, la puissante instruction **LDI** est utilisée pour déplacer un bloc mémoire et la ligne est réaffichée.

Ces actions s'enchaînent comme suit :

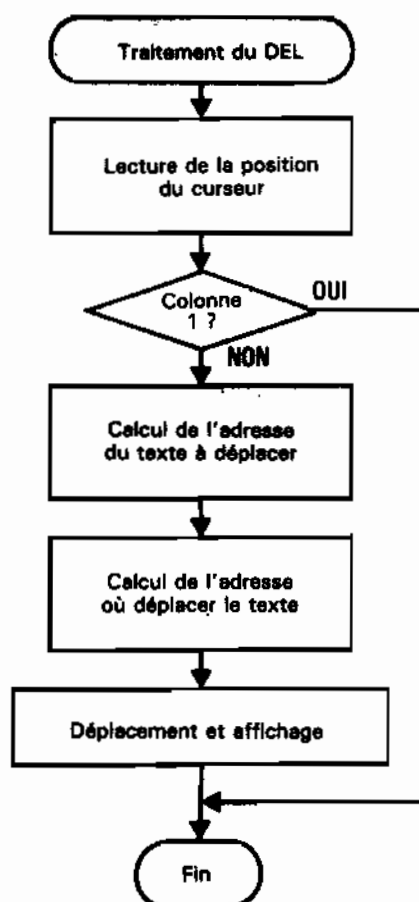


Le sous-programme chargé de cette tâche a pour nom TCLR.

Appui sur la touche

Lorsque l'utilisateur appuie sur la touche , le caractère courant et les caractères qui suivent sont déplacés d'une position vers la gauche. Pour ce faire, la puissante instruction LDI est également utilisée pour déplacer un bloc mémoire et la ligne modifiée est réaffichée.

Ces actions s'enchaînent comme suit :



Le sous-programme chargé de cette tâche a pour nom TDEL.

Le sous-programme Assembleur a été écrit sous ZEN. En voici le listing :

```

1                ORG 8000H
2                LOAD 8000H
3                ; =====
4                ; Initialisation
5                ; =====
6 8000 3E02                LD  A,2
7 8002 CD0EBC                CALL MODE                ; Passage en MODE 2
8 8005 210090                LD  HL,DEBTEX                ; Debut du texte
9 8008 22B0B2                LD  (ADRLIB),HL                ; à ligne depart
10               ;
11 800B 210090                LD  HL,9000H                ; Init memoire travail
12 800E 010010                LD  BC,1000H
13 8011 1620                LD  D,32
14               BOUINI:    EQU  $                ; Boucle d'init
15 8013 72                LD  (HL),D
16 8014 0B                DEC  BC
17 8015 23                INC  HL
18 8016 7B                LD  A,B
19 8017 B1                OR   C
20 8018 20F9                JR   NZ,BOUINI
21               ;
22               ; =====
23               ; Acquisition d'un caractere
24               ; et action correspondante
25               ; =====
26               ACQCAR:    EQU  $
27 801A CD8ABB                CALL PLCUR                ; Affiche curseur
28 801D CD06BB                CALL WAIT                ; Attente frappe 1 caract
29 8020 FE0D                CP   CR
30 8022 CA55B0                JP   Z,TCR                ; Traitement CR

```

```

31 8025 FEF0          CP    UP
32 8027 CA9A80        JP    Z,TUP          ;Traitement UP
33 802A FEF1          CP    DOWN
34 802C CADE80        JP    Z,TDOWN        ;Traitement DOWN
35 802F FEF2          CP    LEFT
36 8031 CA1181        JP    Z,TLEFT        ;Traitement LEFT
37 8034 FEF3          CP    RIGHT
38 8036 CA2881        JP    Z,TRIGHT       ;Traitement RIGHT
39 8039 FE1A          CP    CTRLZ
40 803B CA9D81        JP    Z,HSCROL      ;Scrolling haut
41 803E FE17          CP    CTRLW
42 8040 CA3781        JP    Z,BSCROL      ;Scrolling bas
43 8043 FE10          CP    CLR
44 8045 CAE881        JP    Z,TCLR        ;Traitement CLR
45 8048 FE7F          CP    DEL
46 804A CA0582        JP    Z,TDEL        ;Effect 1 caract
47 804D FE11          CP    CTRLQ
48 804F C8            RET    Z            ;Fin du programme
49 8050 C32F82        JP    AFFICH        ;Affiche un caract
50 8053 18C5          JR     ACQCAR        ;Boucle principale
51                  ; =====
52                  ; ZONE DES TRAITEMENTS
53                  ; =====
54                  ;
55                  ; -----
56                  ; Traitement du CR
57                  ; -----
58                  TCR:    EQU    $
59 8055 CD7082        CALL RESTIT
60 805B 2A8082        LD     HL,(ADRLIG)

```

```

61 805B 115000      LD    DE,80
62 805E 19          ADD   HL,DE
63 805F 2280B2      LD    (ADRLIB),HL      ;Nlle à courante
64 8062 CD78BB      CALL  GETCUR
65 8065 7D          LD    A,L
66 8066 FE19        CP     25              ;Si ligne 25
67 806B 280C        JR     Z,TCR1          ;Trait special
68 806A CD78BB      CALL  GETCUR
69 806D 2601        LD    H,1
70 806F 2C          INC    L
71 8070 CD75BB      CALL  SETCUR
72 8073 C31AB0      JP     ACQCAR
73                TCR1:    EQU    $
74 8076 0601        LD    B,1
75 807B AF          XOR    A
76 8079 CD4DBC      CALL  HWROLL           ;Scrolling haut
77 807C CD52B2      CALL  WRILIB          ;Affich nlle ligne
78 807F CD78BB      CALL  GETCUR
79 8082 2601        LD    H,1
80 8084 CD75BB      CALL  SETCUR           ;Posit. curseur
81 8087 C31AB0      JP     ACQCAR
82                ;
83                ;-----
84                ; Traitement du UP
85                ;-----
86                TUP:    EQU    $
87 808A CD70B2      CALL  RESTIT           ;Restit car courant
88 808D CD78BB      CALL  GETCUR
89 8090 7D          LD    A,L              ;Colonne
90 8091 FE01        CP     1

```

91	8093	2816	JR	Z,TUP1	
92	8095	2A8082	LD	HL,(ADRLIG)	
93	8098	115000	LD	DE,80	
94	809B	ED52	SBC	HL,DE	
95	809D	228082	LD	(ADRLIG),HL	
96	80A0	CD78BB	CALL	GETCUR	
97	80A3	2D	DEC	L	
98	80A4	25	DEC	H	
99	80A5	CD75BB	CALL	SETCUR	
100	80AB	C31A80	JP	ACQCAR	
101		TUP1:	EQU	*	
102	80AB	210090	LD	HL,DEBTEX	;Debut du texte
103	80AE	ED5B8082	LD	DE,(ADRLIG)	
104	80B2	ED52	SBC	HL,DE	
105	80B4	200A	JR	NZ,TUP2	
106	80B6	CD78BB	CALL	GETCUR	
107	80B9	25	DEC	H	
108	80BA	CD75BB	CALL	SETCUR	
109	80BD	C31A80	JP	ACQCAR	
110		TUP2:	EQU	*	
111	80C0	0600	LD	B,0	
112	80C2	AF	XOR	A	
113	80C3	CD4DBC	CALL	HWROLL	;Scroll
114	80C6	2A8082	LD	HL,(ADRLIG)	
115	80C9	115000	LD	DE,80	
116	80CC	ED52	SBC	HL,DE	
117	80CE	228082	LD	(ADRLIG),HL	;Nile à debut ligne
118	80D1	CD5282	CALL	WRILIG	
119	80D4	CD78BB	CALL	GETCUR	
120	80D7	25	DEC	H	

```

121 80D8 CD75BB          CALL SETCUR
122 80DB C31AB0          JP   ACQCAR
123                      ;
124                      ;-----
125                      ; Traitement du DOWN
126                      ;-----
127                      TDOWN: EQU $
128 80DE CD70B2          CALL RESTIT          ;Restit car courant
129 80E1 2A80B2          LD   HL,(ADRLIG)
130 80E4 115000          LD   DE,B0
131 80E7 19              ADD  HL,DE
132 80EB 2280B2          LD   (ADRLIG),HL
133 80EB CD78BB          CALL GETCUR
134 80EE 7D              LD   A,L          ;Colonne
135 80EF FE19            CP   25
136 80F1 2B0B            JR   Z,TD01
137 80F3 CD78BB          CALL GETCUR
138 80F6 25              DEC  H
139 80F7 2C              INC  L
140 80F8 CD75BB          CALL SETCUR
141 80FB C31AB0          JP   ACQCAR
142                      TD01: EQU $
143 80FE 0601            LD   B,1
144 8100 AF              XOR  A
145 8101 CD4DBC          CALL HWROLL
146 8104 CD52B2          CALL WRILIG
147 8107 CD78BB          CALL GETCUR
148 810A 25              DEC  H
149 810B CD75BB          CALL SETCUR
150 810E C31AB0          JP   ACQCAR

```

```

151      ;
152      ;-----
153      ; Traitement du LEFT
154      ;-----
155      TLEFT:      EQU  $
156  8111 CD78BB      CALL GETCUR
157  8114 7C          LD  A,H
158  8115 FE01        CP   1
159  8117 CA1AB0      JP   Z,ACQCAR
160  811A CD70B2      CALL RESTIT      ;Restit caract courant
161  811D CD78BB      CALL GETCUR
162  8120 25          DEC  H
163  8121 25          DEC  H
164  8122 CD75BB      CALL SETCUR
165  8125 C31AB0      JP   ACQCAR
166      ;
167      TRIGHT:     EQU  $
168  8128 CD78BB      CALL GETCUR
169  812B 7C          LD  A,H
170  812C FE50        CP   80
171  812E CA1AB0      JP   Z,ACQCAR
172  8131 CD70B2      CALL RESTIT      ;Restit car courant
173  8134 C31AB0      JP   ACQCAR
174      ;
175      ;-----
176      ; Scrolling bas
177      ;-----
178      BSCROL:      EQU  $
179  8137 CD78BB      CALL GETCUR
180  813A 2207B2      LD   (BVCUR),HL

```

181 813D 45	LD	B,L	
182 813E 2A80B2	LD	HL,(ADRLIB)	
183 8141 115000	LD	DE,80	
184 8144 37	SCF		
185 8145 3F	CCF		
186	BSC1:	EQU	*
187 8146 ED52	SBC	HL,DE	
188 8148 10FC	DJNZ	BSC1	
189 814A 19	ADD	HL,DE	
190 814B 228382	LD	(SAUVHL),HL	;à la ligne
191 814E 110090	LD	DE,DEBTEX	
192 8151 ED52	SBC	HL,DE	
193 8153 7C	LD	A,H	
194 8154 B5	OR	L	
195 8155 CA1A80	JP	Z,ACQCAR	;Scrolling impossible
196 8158 CD78BB	CALL	GETCUR	
197 815B 0600	LD	B,0	
198 815D 4C	LD	C,H	
199 815E 2A80B2	LD	HL,(ADRLIB)	
200 8161 09	ADD	HL,BC	
201 8162 2B	DEC	HL	
202 8163 7E	LD	A,(HL)	
203 8164 CD5ABB	CALL	WRCHAR	;Ecr caract lu
204 8167 0600	LD	B,0	
205 8169 AF	XOR	A	
206 816A CD4DBC	CALL	HWROLL	;Scrolling
207 816D 210101	LD	HL,101H	
208 8170 CD75BB	CALL	SETCUR	
209 8173 2A80B2	LD	HL,(ADRLIB)	
210 8176 228582	LD	(SAUV2),HL	;Sauv à ligne

```

211 8179 2A8382      LD    HL,(SAUVHL)
212 817C 115000      LD    DE,80
213 817F ED52        SBC   HL,DE
214 8181 228082      LD    (ADRLIG),HL
215 8184 CD5282      CALL  WRILIG
216 8187 2A8782      LD    HL,(SAVCUR)
217 818A CD75BB      CALL  SETCUR
218 818D 2A8582      LD    HL,(SAUV2)
219 8190 115000      LD    DE,80
220 8193 37          SCF
221 8194 3F          CCF
222 8195 ED52        SBC   HL,DE
223 8197 228082      LD    (ADRLIG),HL
224 819A C31A80      JP    ACQCAR

225                  ;
226                  ;-----
227                  ; Scrolling haut
228                  ;-----
229                  HSCROL:    EQU . $
230 819D CD78BB      CALL  GETCUR
231 81A0 228782      LD    (SAVCUR),HL      ;Sauv à curseur
232 81A3 0600      LD    B,0
233 81A5 4C          LD    C,H
234 81A6 2A8082      LD    HL,(ADRLIG)
235 81A9 09          ADD   HL,BC
236 81AA 2B          DEC   HL
237 81AB 7E          LD    A,(HL)
238 81AC CD5ABB      CALL  WRCHAR      ;Restit car au curseur
239 81AF 0601      LD    B,1
240 81B1 AF          XOR   A

```



```

241 81B2 CD4DBC      CALL HWROLL
242 81B5 211901      LD   HL,119H
243 81B8 CD75BB      CALL SETCUR
244 81BB 2A80B2      LD   HL,(ADRLIB)
245 81BE 2285B2      LD   (SAUV2),HL      ;Sauv à ligne
246 81C1 ED5B87B2    LD   DE,(SAVCUR)
247 81C5 3E1A       LD   A,26
248 81C7 93         SUB  E
249 81C8 47         LD   B,A
250 81C9 115000      LD   DE,80
251                HSC1: EQU  $
252 81CC 19         ADD  HL,DE
253 81CD 10FD       DJNZ HSC1
254 81CF 22B0B2      LD   (ADRLIB),HL
255 81D2 CD52B2      CALL WRILIB      ;Aff dernière ligne
256 81D5 2A85B2      LD   HL,(SAUV2)
257 81D8 115000      LD   DE,80
258 81DB 19         ADD  HL,DE
259 81DC 22B0B2      LD   (ADRLIB),HL      ;Restit à ligne
260 81DF 2A87B2      LD   HL,(SAVCUR)
261 81E2 CD75BB      CALL SETCUR      ;Restit à curseur
262 81E5 C31A80      JP   ACBCAR
263                ;
264                ;-----
265                ; Traitement du CLR
266                ;-----
267                TCLR: EQU  $
268 81E8 CD78BB      CALL GETCUR
269 81EB 3E50       LD   A,80
270 81ED 94         SUB  H

```

```

271 81EE 47          LD    B,A
272 81EF 04          INC   B
273 81F0 1600        LD    D,0
274 81F2 5C          LD    E,H
275 81F3 2A80B2      LD    HL,(ADRLIG)
276 81F6 19          ADD   HL,DE
277 81F7 2B          DEC   HL
278 81F8 54          LD    D,H
279 81F9 5D          LD    E,L
280 81FA 23          INC   HL
281                  TCLR1: EQU  $
282 81FB EDA0        LDI
283 81FD 10FC        DJNZ  TCLR1
284 81FF CD52B2      CALL  WRILIG          ;Aff nle ligne
285 8202 C31A80      JP    ACQCAR
286                  ;
287                  ;-----
288                  $ Traitement du DEL
289                  ;-----
290                  TDEL:   EQU  $
291 8205 CD78BB      CALL  GETCUR
292 8208 7C          LD    A,H
293 8209 FE01        CP    1
294 820B CA1A80      JP    Z,ACQCAR
295 820E 3E50        LD    A,80
296 8210 94          SUB   H
297 8211 47          LD    B,A
298 8212 04          INC   B
299 8213 1600        LD    D,0
300 8215 5C          LD    E,H

```

```

301 8216 2A80B2      LD    HL,(ADRLIG)
302 8219 19          ADD    HL,DE
303 821A 2B          DEC    HL
304 821B 54          LD     D,H
305 821C 5D          LD     E,L
306 821D 1B          DEC    DE
307                TDEL1: EQU  $
308 821E EDA0        LDI
309 8220 10FC        DJNZ  TDEL1
310 8222 CD52B2      CALL  WRILIG
311 8225 CD78BB      CALL  GETCUR
312 8228 25          DEC    H
313 8229 CD75BB      CALL  SETCUR
314 822C C31AB0      JP     ACQCAR
315                ;
316                ;-----
317                ; Affichage d'un caractere
318                ;-----
319                AFFICH: EQU  $
320 822F FE20        CP     32
321 8231 DA1AB0      JP     C,ACQCAR      ;Caract non affichable
322 8234 3282B2      LD     (SAUVA),A    ; Sauv caract tape
323 8237 CD78BB      CALL  GETCUR      ;Position curseur
324 823A 4C          LD     C,H          ;Colonne curseur
325 823B 7C          LD     A,H
326 823C FE50        CP     80
327 823E CA1AB0      JP     Z,ACQCAR
328 8241 0600        LD     B,0
329 8243 2A80B2      LD     HL,(ADRLIG)  ;Adresse ligne
330 8246 09          ADD    HL,BC

```

```

331 8247 2B          DEC  HL
332 8248 3A82B2      LD   A,(SAUVA)
333 824B 77          LD   (HL),A          ;Sauvegarde caract
334 824C CD5ABB      CALL WRCHAR          ;Affiche caract
335 824F C31A80      JP   ACQCAR

336                  ;
337                  ;-----
338                  ; Affichage d'une ligne
339                  ;-----

340                  WRILIG:  EQU  $
341 8252 CD78BB      CALL GETCUR
342 8255 228382      LD   (SAUVHL),HL
343 8258 2601        LD   H,1
344 825A CD75BB      CALL SETCUR
345 825D 0650        LD   B,80
346 825F 2A8082      LD   HL,(ADRLIG)

347                  BOUWRI:  EQU  $
348 8262 7E          LD   A,(HL)
349 8263 CD5ABB      CALL WRCHAR
350 8266 23          INC  HL
351 8267 10F9        DJNZ BOUWRI
352 8269 2A8382      LD   HL,(SAUVHL)
353 826C CD75BB      CALL SETCUR
354 826F C9          RET

355                  ;
356                  ;-----
357                  ;Restitution du caractere courant
358                  ;-----

359                  RESTIT:  EQU  $
360 8270 CD78BB      CALL GETCUR

```

```

361 8273 4C          LD    C,H
362 8274 0600        LD    B,0
363 8276 2A80B2      LD    HL,(ADR16)
364 8279 09          ADD   HL,BC
365 827A 2B          DEC   HL
366 827B 7E          LD    A,(HL)
367 827C CD5ABB      CALL  WRCHAR
368 827F C9          RET

369                ;
370                ; =====
371                ;      ZONE DES EQU
372                ; =====
373      DEBTEX:      EQU   9000H          ;Debut du texte
374      WAIT:        EQU   0BB06H        ;KM WAIT CHAR
375      WRCHAR:      EQU   0BB5AH        ;TXT OUTPUT
376      MODE:        EQU   0BC0EH        ;SCR SET MODE
377      GETCUR:      EQU   0BB7BH        ;TXT SET CURSOR
378      PLCUR:       EQU   0BB8AH        ;PLACE CURSOR
379      SETCUR:      EQU   0BB75H        ;TXT SET CURSOR
380      HWROLL:      EQU   0BC4DH        ;SCR HW ROLL
381      CURDIS:      EQU   0BB7EH        ;CUR DISABLE
382      CURON:       EQU   0BB81H        ;TXT CUR ON
383      CR:          EQU   13            ;Carriage Return
384      UP:          EQU   240
385      DOWN:        EQU   241
386      LEFT:        EQU   242
387      RIGHT:       EQU   243
388      CTRLA:       EQU   01H
389      CTRLF:       EQU   06H
390      CTRLZ:       EQU   1AH

```

```

391          CTRLW:      EQU   17H
392          CTRLC:      EQU   03H
393          CTRLR:      EQU   12H
394          CLR:         EQU   16
395          DEL:         EQU   127
396          CTRLT:      EQU   14H
397          CTRLV:      EQU   19H
398          CTRLV:      EQU   16H
399          CTRLQ:      EQU   11H
400          SPACE:      EQU   32
401          ;
402          ; *****
403          ; ZONE DES BUFFERS
404          ; *****
405          ADRLIG:      DS     2           ; à debut de ligne
406          SAUVA:       DS     1           ; Sauvegarde registre A
407          SAUVHL:     DS     2           ; Sauvegarde reg HL
408          SAUVZ:       DS     2           ; Sauvegarde 2 octets
409          SAVCUR:     DS     2           ; Sauv coord curseur
410          ;
411          ;
412          END

```

Si vous ne désirez pas entrer les nombreuses lignes de code de ce programme, vous pouvez entrer le chargeur Basic suivant :

```

1000 REM -----
1010 REM Chargeur Basic du traitement de texte
1020 REM -----
1030 FOR I=&B000 TO &B27F
1040   READ A$
1050   A$=" "+A$
1060   POKE I,VAL(A$)
1070 NEXT I
1080 CALL &B000
1090 END
1100 REM -----
2000 DATA 3E,2,CD,E,BC,21,0,90,22,80,82,21,0,90,1,0
2010 DATA 10,16,20,72,B,23,78,B1,20,F9,CD,BA,BB,CD,6,BB
2020 DATA FE,D,CA,55,80,F2,F0,CA,8A,80,FE,F1,CA,DE,80,FE
2030 DATA F2,CA,11,81,FE,F3,CA,2B,81,FE,1A,CA,9D,81,FE,17
2040 DATA CA,37,81,FE,10,CA,EB,81,FE,7F,CA,5,82,FE,11,CB
2050 DATA C3,2F,82,18,C5,CD,70,82,2A,80,82,11,50,0,19,22
2060 DATA 80,82,CD,78,BB,7D,FE,19,2B,C,CD,78,BB,26,1,2C
2070 DATA CD,75,BB,C3,1A,80,6,1,AF,CD,4D,BC,CD,52,82,CD
2080 DATA 78,BB,26,1,CD,75,BB,C3,1A,80,CD,70,82,CD,78,BB
2090 DATA 7D,FE,1,28,16,2A,80,82,11,50,0,ED,52,22,80,82
2100 DATA CD,78,BB,2D,25,CD,75,BB,C3,1A,80,21,0,90,ED,5B
2110 DATA 80,82,ED,52,20,A,CD,78,BB,25,CD,75,BB,C3,1A,80
2120 DATA 6,0,AF,CD,4D,BC,2A,80,82,11,50,0,ED,52,22,80
2130 DATA 82,CD,52,82,CD,78,BB,25,CD,75,BB,C3,1A,80,CD,70
2140 DATA 82,2A,80,82,11,50,0,19,22,80,82,CD,78,BB,7D,FE
2150 DATA 19,28,B,CD,78,BB,25,2C,CD,75,BB,C3,1A,80,6,1
2160 DATA AF,CD,4D,BC,CD,52,82,CD,78,BB,25,CD,75,BB,C3,1A
2170 DATA 80,CD,78,BB,7C,FE,1,CA,1A,80,CD,70,82,CD,78,BB

```

```

2180 DATA 25,25,CD,75,BB,C3,1A,80,CD,78,BB,7C,FE,50,CA,1A
2190 DATA 80,CD,70,82,C3,1A,80,CD,78,BB,22,87,82,45,2A,80
2200 DATA 82,11,50,0,37,3F,ED,52,10,FC,19,22,83,82,11,0
2210 DATA 90,ED,52,7C,B5,CA,1A,80,CD,78,BB,6,0,4C,2A,80
2220 DATA 82,9,2B,7E,CD,5A,BB,6,0,AF,CD,4D,BC,21,1,1
2230 DATA CD,75,BB,2A,80,82,22,85,82,2A,83,82,11,50,0,ED
2240 DATA 52,22,80,82,CD,52,82,2A,87,82,CD,75,BB,2A,85,82
2250 DATA 11,50,0,37,3F,ED,52,22,80,82,C3,1A,80,CD,78,BB
2260 DATA 22,87,82,6,0,4C,2A,80,82,9,2B,7E,CD,5A,BB,6
2270 DATA 1,AF,CD,4D,BC,21,19,1,CD,75,BB,2A,80,82,22,85
2280 DATA 82,ED,5B,87,82,3E,1A,93,47,11,50,0,19,10,FD,22
2290 DATA 80,82,CD,52,82,2A,85,82,11,50,0,19,22,80,82,2A
2300 DATA 87,82,CD,75,BB,C3,1A,80,CD,78,BB,3E,50,94,47,4
2310 DATA 16,0,5C,2A,80,82,19,2B,54,5D,23,ED,A0,10,FC,CD
2320 DATA 52,82,C3,1A,80,CD,78,BB,7C,FE,1,CA,1A,80,3E,50
2330 DATA 94,47,4,16,0,5C,2A,80,82,19,2B,54,5D,1B,ED,A0
2340 DATA 10,FC,CD,52,82,CD,78,BB,25,CD,75,BB,C3,1A,80,FE
2350 DATA 20,DA,1A,80,32,82,82,CD,78,BB,4C,7C,FE,50,CA,1A
2360 DATA 80,6,0,2A,80,82,9,2B,3A,82,82,77,CD,5A,BB,C3
2370 DATA 1A,80,CD,78,BB,22,83,82,26,1,CD,75,BB,6,50,2A
2380 DATA 80,82,7E,CD,5A,BB,23,10,F9,2A,83,82,CD,75,BB,C9
2390 DATA CD,78,BB,4C,6,0,2A,80,82,9,2B,7E,CD,5A,BB,C9
    
```

Assurez-vous que les données entrées sont correctes grâce au programme de checksum qui doit donner le résultat suivant :

**62 CE 8C D0 71 DD 24 5C 7B AF AC F1 FE E7 CD 4 2E 27 5A BD
F9 67 C9 D5 7F 9D 48 97 B3 A1 D7 22 A5 1F 33 CB 46 6B 8B E1**

Si une des données affichées par le programme de checksum n'est pas identique à celles indiquées ci-dessus (supposons qu'il s'agisse de la donnée n), la nième ligne de données contient très certainement une ou plusieurs fautes de frappe...

9/11.2.2

Premier jeu de fonctions évoluées

Dans ce paragraphe, nous vous proposons quatre fonctions destinées à améliorer le fonctionnement du traitement de texte Weka étudié à la Partie 9, chapitre 11.2. En voici la liste :

- déplacement rapide du curseur vers le début du prochain mot de la ligne courante ;
- déplacement rapide du curseur vers le début du mot précédent de la ligne courante ;
- déplacement rapide en fin de ligne courante ;
- déplacement rapide au début de la ligne courante.

Ces quatre fonctions sont exécutées par des sous-programmes écrits en Assembleur. Elles se logent immédiatement après la fin du programme de base : adresse &H8289.

Examinons en détail les diverses actions effectuées par chacune des nouvelles fonctions.

I. Déplacements rapides

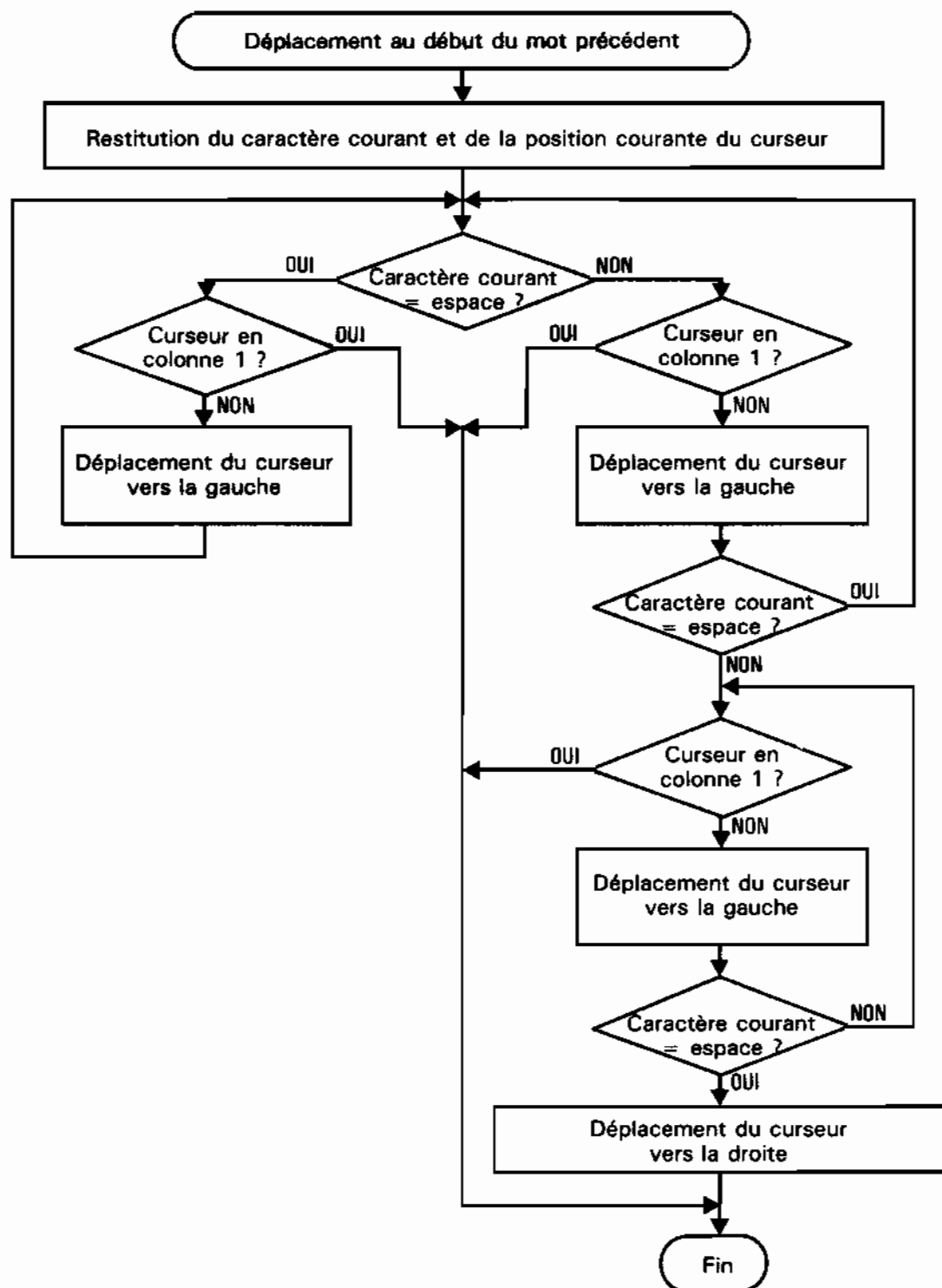
DÉPLACEMENT RAPIDE A DROITE

Lorsque l'utilisateur appuie simultanément sur les touches **Ctrl** et **A**, le curseur se déplace sur la première lettre du mot qui suit le mot courant.

Si le curseur se trouve en fin de ligne, aucun déplacement n'est effectué.

Si aucun mot ne se trouve à droite du curseur, celui-ci se déplace en fin de ligne courante.

Ces actions obéissent à la logique suivante :



Le sous-programme correspondant à cette fonction a pour nom TMDRO.

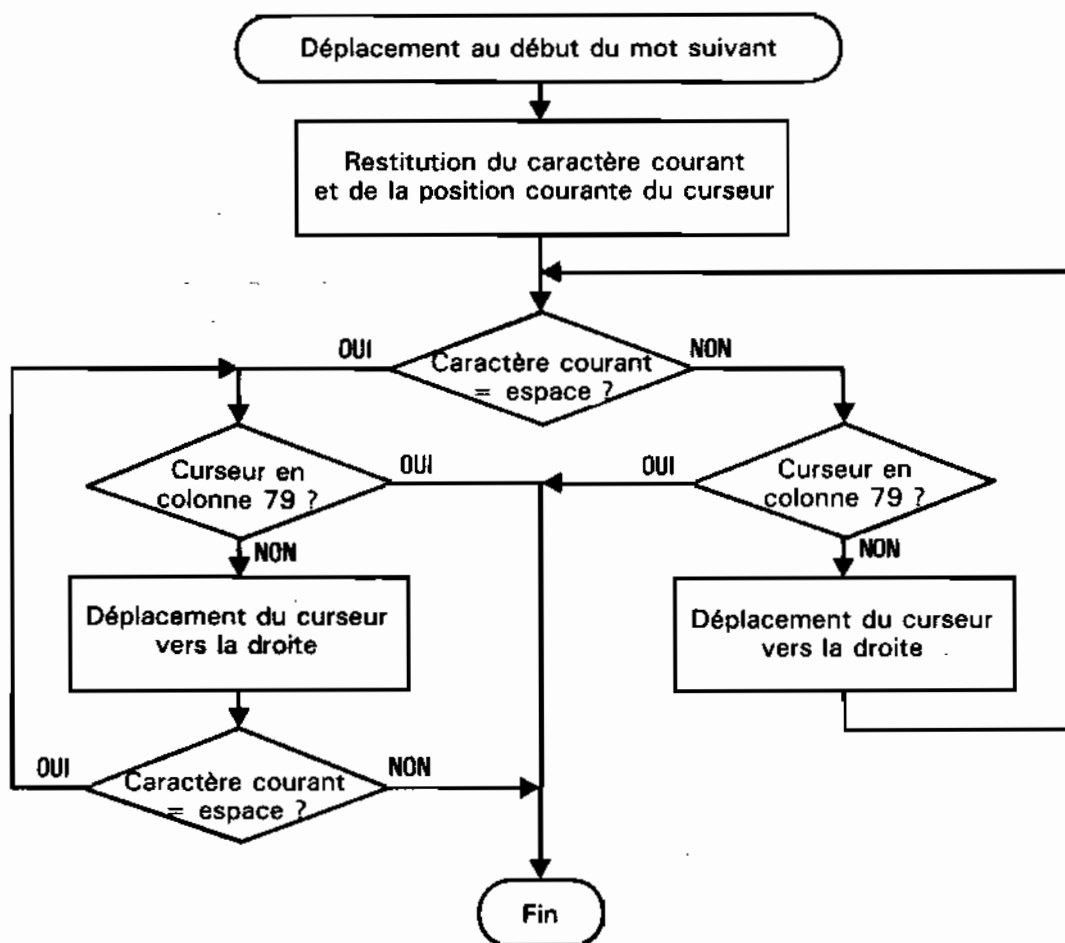
DÉPLACEMENT RAPIDE A GAUCHE

Lorsque l'utilisateur appuie simultanément sur les touches **Ctrl** et **F**, le curseur se déplace sur la première lettre du mot qui précède le mot courant.

Si le curseur se trouve en début de ligne, aucun déplacement n'est effectué.

Si aucun mot ne se trouve à gauche du curseur, celui-ci se déplace en début de ligne courante.

Ces actions obéissent à la logique suivante :

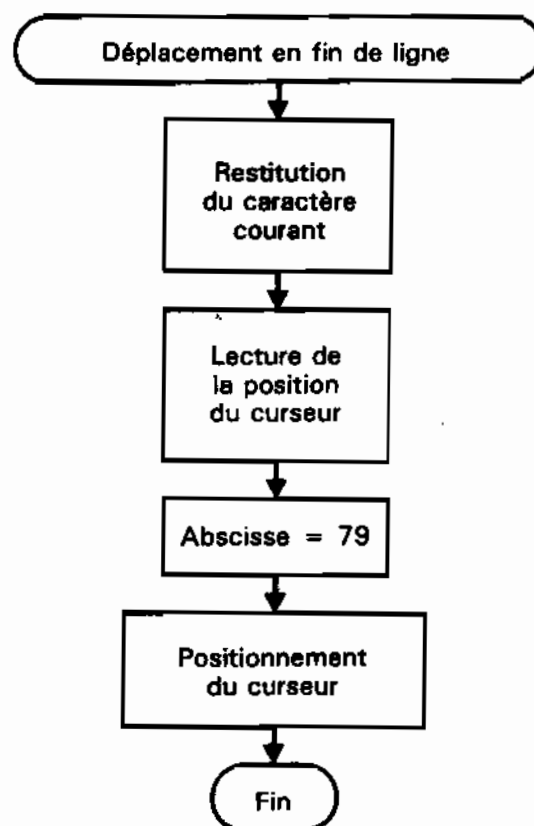


Le sous-programme correspondant à cette fonction a pour nom **TMGAU**.

DÉPLACEMENT EN FIN DE LIGNE

Lorsque l'utilisateur appuie simultanément sur les touches **Ctrl** et **Right**, le curseur se déplace sur le dernier caractère de la ligne courante.

Cette action met en œuvre des instructions qui obéissent à la logique suivante :

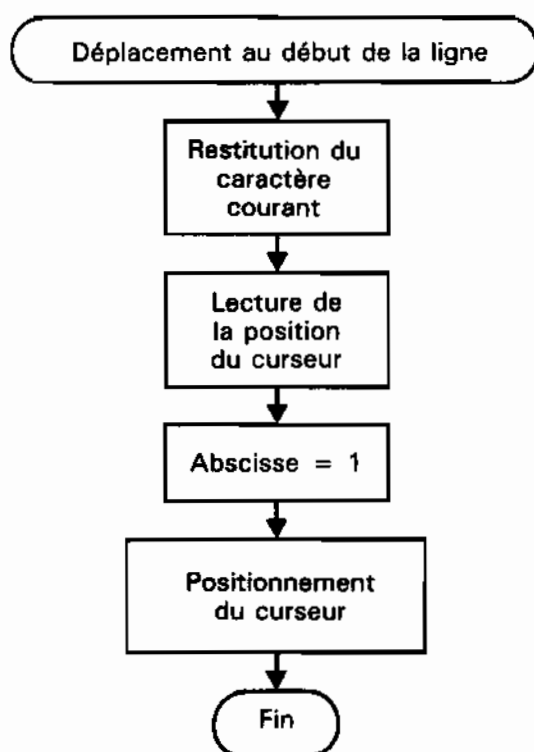


Le sous-programme correspondant à cette fonction a pour nom TFINLI.

DÉPLACEMENT AU DÉBUT DE LA LIGNE

Lorsque l'utilisateur appuie simultanément sur les touches **Ctrl** et **Left**, le curseur se déplace sur le premier caractère de la ligne courante.

Cette action met en œuvre des instructions qui obéissent à la logique suivante :



Le sous-programme correspondant à cette fonction a pour nom TDEBLI.

Le programme Assembleur qui contient ces quatre fonctions a été écrit sous ZEN. En voici le listing :

```

1                ORG 82B9H
2                LOAD 82B9H
3                ;-----
4                ;Suite 1 Traitement de texte WEKA
5                ;-----
6                ;Fonctions rajoutees:
7                ;Ctrl A : Deplact mot a gauche
8                ;Ctrl F : Deplact mot a droite
9                ;Ctrl Right : Fin de la ligne
    
```

```

10          ;Ctrl Left : Debut de la ligne
11          ;-----
12          ;
13 8289 FE01          CP    CTRLA
14 828B CAA3B2        JP    Z,TMGAU          ;Debut mot a gauche
15 828E FE06          CP    CTRLF
16 8290 CA0B83        JP    Z,TMDRO          ;Debut mot a droite
17 8293 FEFB          CP    CTRLRI
18 8295 CA45B3        JP    Z,TFINLI          ;Fin de la ligne
19 8298 FEFA          CP    CTRLLE
20 829A CA54B3        JP    Z,TDEBLI          ;Debut de la ligne
21 829D C32FB2        JP    AFFICH          ;Affichage caractere
22 82A0 C31AB0        JP    ACQCAR          ;Boucle principale
23          ;-----
24          ;
25          ;-----
26          ;Deplact debut mot a gauche
27          ;-----
28          TMGAU:      EQU $
29 82A3 CD70B2        CALL RESTIT          ;Caract courant
30 82A6 CD78BB        CALL GETCUR
31 82A9 25            DEC    H
32 82AA CD75BB        CALL SETCUR          ;Restit pas courante
33          GAUS0:      EQU $
34 82AD CD60BB        CALL RDCHAR          ;Lect car courant
35 82B0 FE20          CP    SPACE          ;Blanc ?
36 82B2 2B1C          JR     Z,GAU2          ;Oui
37 82B4 CD78BB        CALL GETCUR
38 82B7 7C            LD     A,H
39 82B8 FE01          CP     1
40 82BA 2B3C          JR     Z,GAU4          ;Plus de deplact

```

```

41 82BC 25          DEC H
42 82BD CD75BB      CALL SETCUR
43 82CD CD60BB      CALL RDCHAR
44 82C3 FE20        CP SPACE
45 82C5 2802        JR Z,GAU0
46 82C7 1815        JR GAU3
47                GAU0: EQU $
48 82C9 CD60BB      CALL RDCHAR          ;Lect car courant
49 82CC FE20        CP SPACE            ;Blanc ?
50 82CE 200E        JR NZ,GAU3          ;Non
51                GAU2: EQU $
52 82D0 CD78BB      CALL GETCUR
53 82D3 7C          LD A,H
54 82D4 FE01        CP 1
55 82D6 2820        JR Z,GAU4            ;Plus de deplact
56 82D8 25          DEC H
57 82D9 CD75BB      CALL SETCUR
58 82DC 18EB        JR GAU0              ;Boucle
59                GAU3: EQU $
60 82DE CD78BB      CALL GETCUR
61 82E1 7C          LD A,H
62 82E2 FE01        CP 1
63 82E4 2812        JR Z,GAU4            ;Plus de deplact
64 82E6 25          DEC H
65 82E7 CD75BB      CALL SETCUR
66 82EA CD60BB      CALL RDCHAR
67 82ED FE20        CP SPACE            ;Blanc ?
68 82EF 20ED        JR NZ,GAU3          ;Non
69 82F1 CD78BB      CALL GETCUR
70 82F4 24          INC H

```

```

71 82F5 CD75BB      CALL SETCUR          ;Ajustement curseur
72                GAU4:    EQU $
73 82F8 C31A80      JP  ACQCAR          ;Fin du traitement
74                GAU1:    EQU $
75 82FB CD60BB      CALL RDCHAR
76 82FE FE01        CP  1
77 8300 28F6        JR  Z,GAU4          ;Plus de deplact
78 8302 CD78BB      CALL SETCUR
79 8305 25          DEC  H
80 8306 CD75BB      CALL SETCUR
81 8309 18BE        JR  GAU0          ;Boucle de recherche
82                ;
83                ;-----
84                ;Deplact debut mot a droite
85                ;-----
86                TMDRO:    EQU $
87 830B CD70B2      CALL RESTIT          ;Restit car courant
88 830E CD78BB      CALL GETCUR          ;Position curseur
89 8311 25          DEC  H
90 8312 CD75BB      CALL SETCUR          ;Restit pos curseur
91                DROI0:    EQU $
92 8315 CD60BB      CALL RDCHAR          ;Lecture car courant
93 8318 FE20        CP  SPACE          ;Blanc?
94 831A 2015        JR  NZ,DROI1        ;Non
95                DROI2:    EQU $
96 831C CD78BB      CALL SETCUR
97 831F 7C          LD  A,H
98 8320 FE4F        CP  79
99 8322 281E        JR  Z,DROI3        ;Fin de deplact
100 8324 24         INC  H
101 8325 CD75BB      CALL SETCUR

```



```

102 832B CD60BB          CALL RDCHAR
103 832B FE20            CP    SPACE
104 832D 28ED            JR    Z,DRO12          ;Boucle
105 832F 1811            JR    DRO13          ;Fin de deplact
106                      ;
107                      DRO11: EQU $
108 8331 CD78BB          CALL GETCUR
109 8334 7C              LD    A,H
110 8335 FE4F            CP    79              ;Fin de ligne ?
111 8337 2809            JR    Z,DRO13          ;Oui
112 8339 CD78BB          CALL GETCUR
113 833C 24              INC    H
114 833D CD75BB          CALL SETCUR
115 8340 18D3            JR    DRO10          ;Boucle
116                      DRO13: EQU $
117 8342 C31A80          JP    ACQCAR          ;Fin du traitement
118                      ;
119                      ;-----
120                      ;Deplact fin de la ligne
121                      ;-----
122                      TFINLI: EQU $
123 8345 CD7082          CALL RESTIT          ;Caract courant
124 8348 CD78BB          CALL GETCUR
125 834B 3E4F            LD    A,79
126 834D 67              LD    H,A
127 834E CD75BB          CALL SETCUR          ;Curs en fin de ligne
128 8351 C31A80          JP    ACQCAR          ;Fin de traitement
129                      ;
130                      ;-----
131                      ;Deplact debut de la ligne
132                      ;-----

```

```

133          TDEBLI:      EQU    $
134 8354 CD70B2          CALL RESTIT          ;Caract courant
135 8357 CD78BB          CALL GETCUR
136 835A 3E01           LD     A,1
137 835C 67             LD     H,A
138 835D CD75BB          CALL SETCUR          ;Position curseur
139 8360 C31A80          JP     ACQCAR        ;Fin traitement

140          ;
141          ;-----
142          ;     ZONE DES EQU
143          ;-----
144          ;
145  GETCUR:      EQU    0BB7BH          ;TXT GET CURSOR
146  SETCUR:      EQU    0BB75H          ;TXT SET CURSOR
147  RDCHAR:      EQU    0BB60H          ;TXT RD CHAR
148  RESTIT:      EQU    8270H          ;Rest car cour
149  CTRLA:       EQU    01H
150  CTRLF:       EQU    06H
151  CTRLRI:      EQU    251
152  CTRLLE:      EQU    250
153  SPACE:       EQU    32
154          ;
155  AFFICH:      EQU    822FH          ;Ref 1er prog
156  ACQCAR:      EQU    801AH          ;Ref 1er prog
157          ;
158          ;
159          END

```

Comme toujours, voici la version « Chargeur Basic » correspondant au programme Assembleur listé ci-dessus :

```

1000 ' =====
1010 ' Suite 1 de l'editeur de texte WEKA
1020 ' =====
1030 '
1040 ' -----
1050 ' Memorisation des S/P Asm
1060 ' -----
1070 FOR I=&B2B9 TO &B360
1080   READ A$
1090   A=VAL("&"+A$)
1100   POKE I,A
1110 NEXT I
1120 '
1130 ' -----
1140 POKE &B051,&B9 : POKE &B052,&B2 'Modif du 1er programme
1150 ' -----
1160 CALL &B000 'Execution
1170 END
1180 '
1190 ' -----
1200 'Donnees des S/P Assembleur
1210 ' -----
1220 DATA FE,1,CA,A3,B2,FE,6,CA,B,B3,FE,FB,CA,45,B3,FE
1230 DATA FA,CA,54,B3,C3,2F,B2,B3,1A,B0,CD,70,B2,CD,7B,BB
1240 DATA 25,CD,75,BB,CD,60,BB,FE,20,2B,1C,CD,7B,BB,7C,FE
1250 DATA 1,2B,3C,25,CD,75,BB,CD,60,BB,FE,20,2B,2,1B,15
1260 DATA CD,60,BB,FE,20,20,E,CD,7B,BB,7C,FE,1,2B,20,25
1270 DATA CD,75,BB,1B,EB,CD,7B,BB,7C,FE,1,2B,12,25,CD,75
1280 DATA BB,CD,60,BB,FE,20,20,ED,CD,7B,BB,24,CD,75,BB,C3
1290 DATA 1A,B0,CD,60,BB,FE,1,2B,F6,CD,7B,BB,25,CD,75,BB
1300 DATA 1B,BE,CD,70,B2,CD,7B,BB,25,CD,75,BB,CD,60,BB,FE
1310 DATA 20,20,15,CD,7B,BB,7C,FE,4F,2B,1E,24,CD,75,BB,CD
1320 DATA 40,BB,FE,20,2B,ED,1B,11,CD,7B,BB,7C,FE,4F,2B,9
1330 DATA CD,7B,BB,24,CD,75,BB,1B,D3,C3,1A,B0,CD,70,B2,CD
1340 DATA 7B,BB,3E,4F,67,CD,75,BB,C3,1A,B0,CD,70,B2,CD,7B
1350 DATA BB,3E,1,67,CD,75,BB,C3,1A,B0,0,0,0,0,0,0

```

Assurez-vous que les données entrées sont correctes grâce au programme de checksum qui doit donner le résultat suivant :

DC 34 EE E9 23 24 BB C9 A6 59 7B FD BD BF

Si une des données affichées par le programme de checksum n'est pas identique à celle de même rang dans la liste ci-dessus, la ligne de DATA correspondante contient une ou plusieurs erreurs de frappe...

Pour implanter les quatre fonctions dont nous venons de parler, il faut procéder de la manière suivante :

— si vous utilisez un **Assembleur** :

- entrez les codes opératoires donnés dans le premier listing,
- implantez le programme de base en mémoire,
- placez les valeurs &H89 et &H82 dans les octets d'adresse &H8051 et &H8052, ceci pour effectuer un débranchement dans la boucle principale,
- activez le programme en tapant, sous **Basic** Call &8000.

— si vous préférez utiliser le chargeur **Basic** :

- chargez en mémoire le premier chargeur (Partie 9, Chap. 11.2.1 page 30),
- supprimez la ligne 1080,
- sauvegardez puis exécutez le chargeur ainsi modifié,
- chargez en mémoire le second chargeur (voir Partie 9, Chap. 11.2.2 page 11),
- exécutez ce second chargeur. Vous vous trouvez dans le traitement de texte.

II - Résumé des fonctions disponibles

Les commandes désormais disponibles dans le traitement de texte Weka sont les suivantes :

- touches flèches pour déplacer le curseur dans toutes les directions,
- **Carriage Return** pour passer à la ligne,
- **Ctrl Z** pour effectuer un scrolling d'une ligne vers le haut,
- **Ctrl W** pour effectuer un scrolling d'une ligne vers le bas,
- **CLR** pour effacer le caractère courant,
- **DEL** pour effacer le caractère à gauche du caractère courant,
- **Ctrl A** pour déplacer le curseur sur la première lettre du mot qui se trouve à droite du mot courant,
- **Ctrl F** pour déplacer le curseur sur la première lettre du mot qui se trouve à gauche du mot courant,
- **Ctrl Right** pour déplacer le curseur sur le dernier caractère de la ligne courante,
- **Ctrl Left** pour déplacer le curseur sur le premier caractère de la ligne courante,
- **Ctrl Q** pour quitter le traitement de texte et retourner sous **Basic**.

9/12

Correcteurs orthographiques

Ce chapitre analyse plusieurs correcteurs orthographiques, destinés à corriger automatiquement des fichiers texte issus, par exemple, d'un traitement de texte.

Le premier de ces programmes est assez élémentaire, mais très rapide car écrit en Assembleur.

9/12.1

Correcteur orthographique de base

Avant d'entrer dans les détails du programme, voyons comment fonctionnent la plupart des correcteurs orthographiques. Les plus simples comparent chaque mot du texte à corriger avec un dictionnaire. Lorsqu'un mot a une orthographe voisine (à une ou deux lettres près, par exemple) d'un des mots du dictionnaire, il est remplacé par ce dernier, éventuellement après confirmation. Bien entendu, ce type de correcteur ne tient pas compte des accords puisqu'il se contente de comparer la similitude des mots du texte et des mots du dictionnaire. Pour l'instant, nous nous limiterons à ce type élémentaire de correcteur. Les mots du dictionnaire devront donc être invariables.

Rassurez-vous, nous verrons très bientôt un correcteur d'un type plus évolué qui permet d'entrer des mots quelconques dans le dictionnaire...

Ce premier programme n'a qu'un but pédagogique. Grâce à lui, vous comprendrez la philosophie générale des correcteurs orthographiques, et vous serez capable de modifier un des correcteurs présentés dans ce livre ou de créer votre propre correcteur.

Le dictionnaire

Il est composé d'un ensemble de mots (en caractères majuscules dans ce programme) séparés les uns des autres par des caractères séparateurs (espace dans notre cas). Un caractère spécial, dit terminateur (OFFH dans notre cas) est placé après le dernier caractère du dernier mot du dictionnaire.

Supposons que les mots du dictionnaire soient les suivants :
chat, mère, miche, appétit.

Ils seront codés comme suit dans le dictionnaire :

CHAT<20H>MERE<20H>MICHEL<20H>APPETIT<FFH>

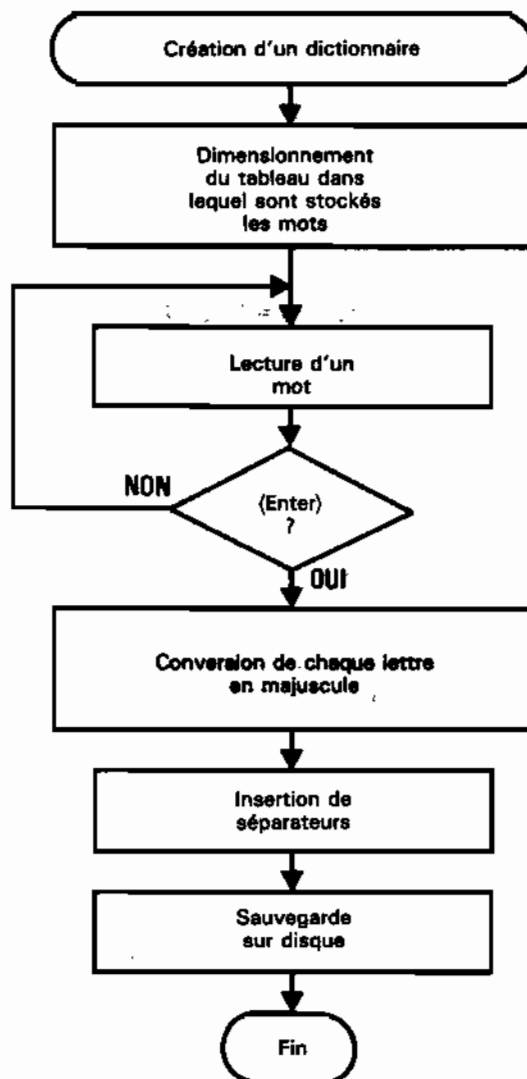
Le programme suivant permet de créer un ou plusieurs fichiers dictionnaire qui respectent les conventions énoncées ci-dessus :

```

1000 MODE 2
1010 PRINT"SAISIE DES MOTS DU DICTIONNAIRE"
1020 PRINT"-----"
1030 PRINT
1040 DIM A$(100) 'Nombre de mots
1050 I=1
1060 INPUT A$(I)
1070 IF A$(I)<>" THEN I=I+1:IF I<>11 THEN 1060
1080 '-----
1090 'Conversion
1100 '-----
1110 I=1:AD=&8000
1120 FOR J=1 TO LEN(A$(I))
1130   A=ASC(MID$(A$(I),J)) AND &DF 'Mise en majuscule
1140   POKE AD,A
1150   AD=AD+1
1160 NEXT J
1170 POKE AD,32 'Separateur
1180 AD=AD+1
1190 I=I+1:IF A$(I)<>" THEN 1120
1200 AD=AD-1:POKE AD,&FF
1210 '-----
1220 'Stockage
1230 '-----
1240 PRINT
1250 INPUT "Nom du dictionnaire : ";N$
1260 SAVE N$,B,&8000,AD-&8000+1
1270 END

```

Le fonctionnement du programme respecte les étapes suivantes :



Le texte à analyser

Il est composé d'un ensemble de mots (en caractères majuscules dans ce programme) répartis sur une ou plusieurs lignes. Deux caractères séparateurs sont insérés entre chaque ligne de texte. Il s'agit des caractères CR (ASCII 13) et LF (ASCII 10). Aucun caractère terminateur ne doit être inséré après le dernier CR LF du fichier, ceci afin de conserver la compatibilité avec la plupart des traitements de texte. Supposons que le texte suivant doive être analysé :

Le chat de la mère michel
mange avec appétit

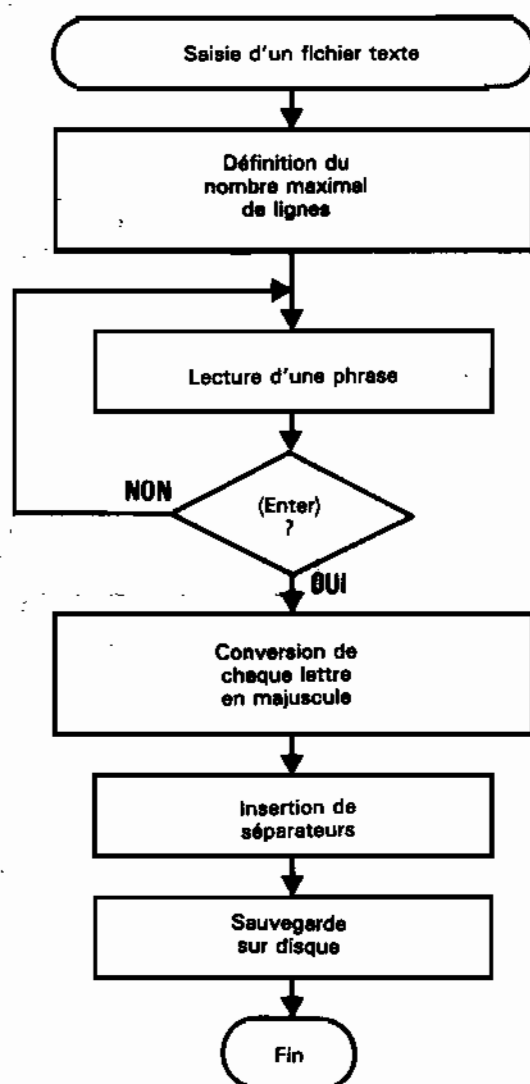
Ce texte sera codé comme suit :

LE CHAT DE LA MERE MICHEL<13><10>
MANGE AVEC APPETIT<13><10>

Le programme suivant permet de créer un ou plusieurs fichiers texte qui respectent les conventions énoncées ci-dessus :

```
1000 MODE 2
1010 PRINT "SAISIE DE PHRASES POUR L'ANALYSEUR DE SYNTAXE"
1020 PRINT "-----"
1030 PRINT
1040 DIM A$(10) 'Nombre de lignes
1050 I=1
1060 INPUT A$(I)
1070 IF A$(I)<>" " THEN I=I+1: IF I>11 THEN 1060
1080 '-----
1090 'Conversion
1100 '-----
1110 I=1: AD=&B000
1120 FOR J=1 TO LEN(A$(I))
1130   A=ASC(MID$(A$(I),J))AND &DF 'Mise en majuscule
1135   IF A=0 THEN A=32 'Espace
1140   POKE AD,A
1150   AD=AD+1
1160 NEXT J
1170 POKE AD,13: AD=AD+1: POKE AD,10: AD=AD+1 'Fin de ligne
1180 I=I+1: IF A$(I)<>" " THEN 1120
1190 '-----
1200 'Stockage
1210 '-----
1220 PRINT
1230 INPUT "Nom du fichier : ";N$
1240 SAVE N$,B,&B000,AD-&B000
1250 END
```


Le fonctionnement du programme respecte les étapes suivantes :



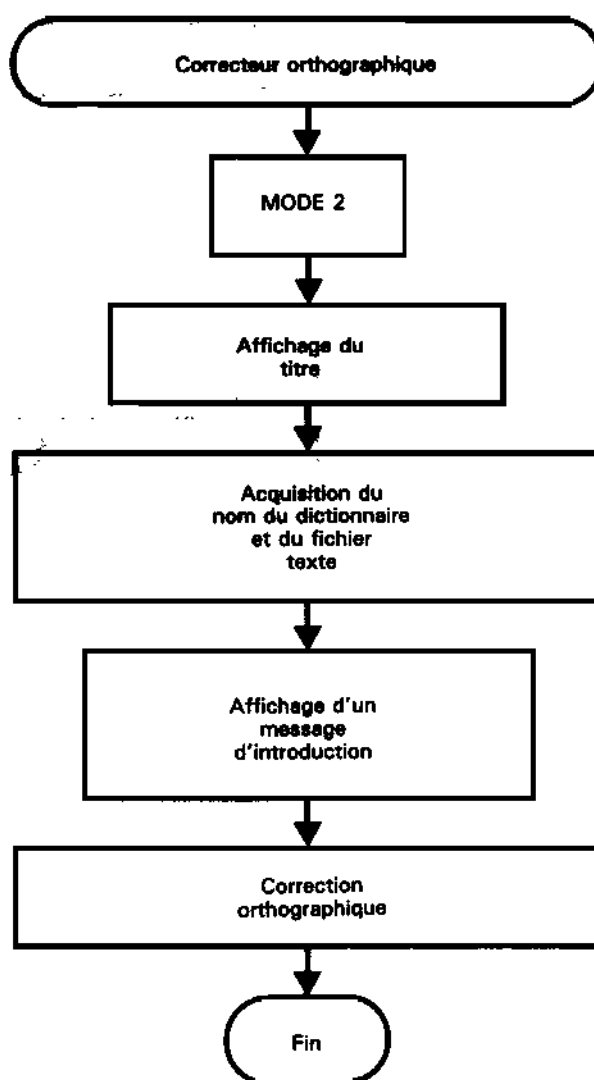
Le programme

Comme nous l'avons dit précédemment, le correcteur compare chaque mot du texte à corriger avec chaque mot du dictionnaire. Ceci représente un travail considérable si le texte à corriger et/ou si le dictionnaire est de grande taille.

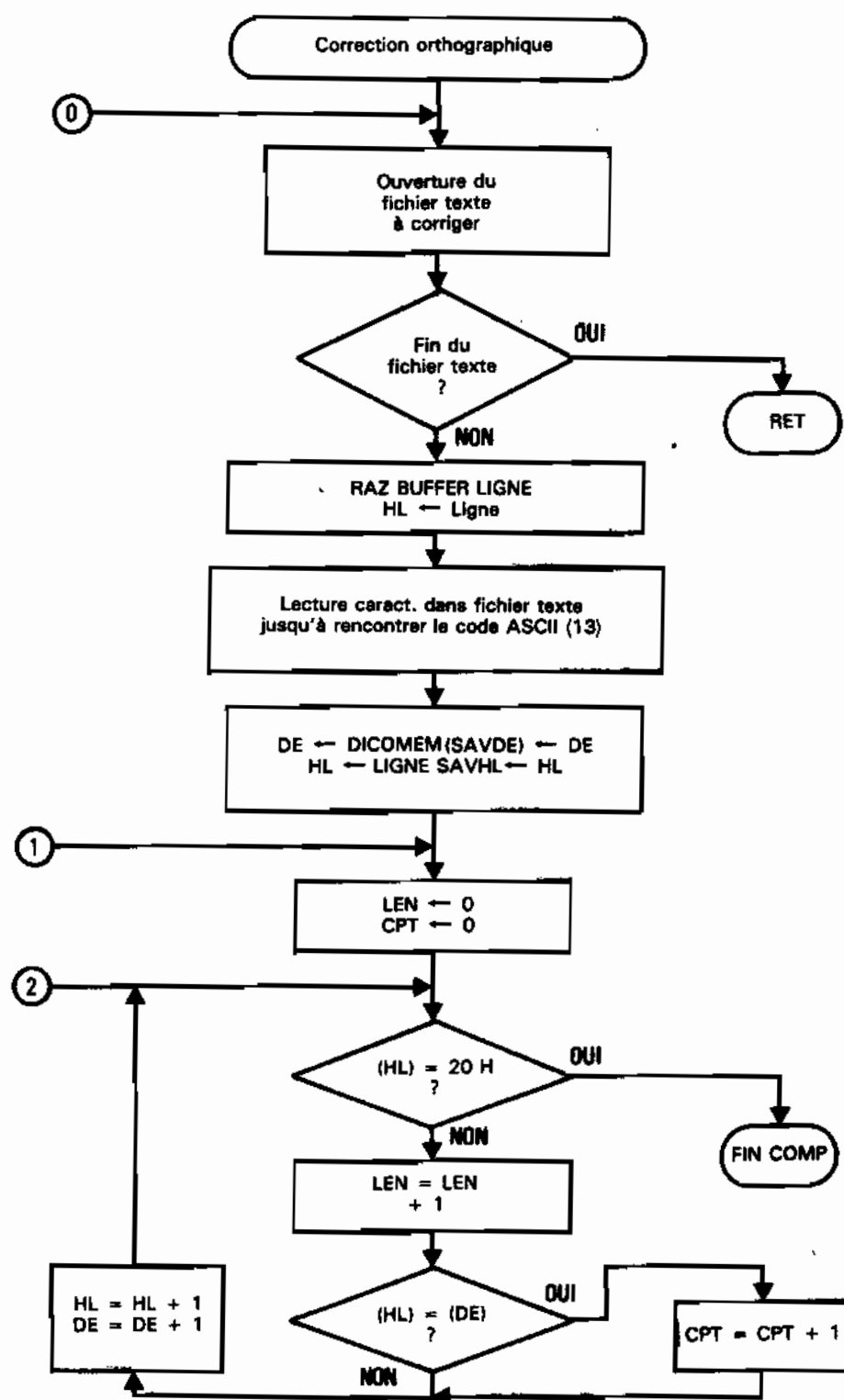
C'est pourquoi nous avons développé le correcteur dans un langage rapide : l'Assembleur.

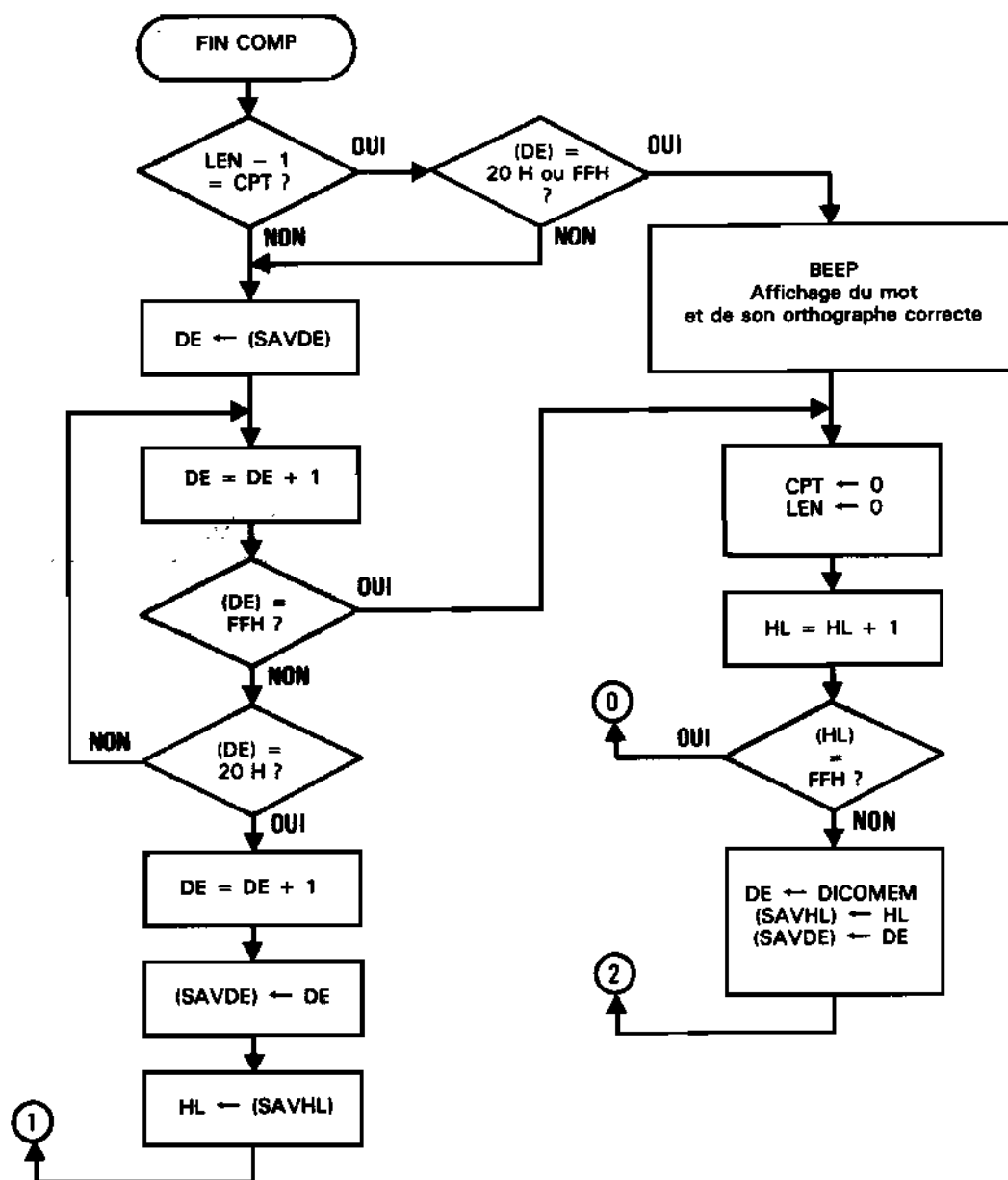
Avant de pouvoir l'utiliser, vous devez avoir créé un fichier dictionnaire et un fichier texte stockés sur disque, sous forme binaire, à l'aide des deux programmes précédents.

La logique générale du correcteur obéit à l'ordinogramme suivant :



Voici le détail de la case Correction dans l'ordinogramme précédent :





Le listing du correcteur orthographique est le suivant :

```

1          ORG 9000H
2          LOAD 9000H
3 9000 C30691          JP DEBUT          ;Debut du programme
4          ;
5          ;-----
6          ;ZONE DE DECLARATIONS
7          ;-----
8          ;
9          SOUND:      EQU 0BD34H          ;MC SOUND REG
10         PRINT:      EQU 0BB5AH          ;TXT OUTPUT
11         READ:       EQU 0BB06H          ;KM WAIT CHAR
12         MODE:       EQU 0BC0EH          ;SCR SET MODE
13         OPEN:       EQU 0BC77H          ;CAS IN OPEN
14         CLOSE:      EQU 0BC7AH          ;CAS IN CLOSE
15         CHAR:       EQU 0BC80H          ;CAS IN CHAR
16         DIRECT:     EQU 0BC83H          ;CAS IN DIRECT
17         TESTEOF:    EQU 0BC89H          ;CAS TEST EOF
18         OUTOPEN:    EQU 0BC8CH          ;CAS OUT OPEN
19         OUTCLOS:    EQU 0BC8FH          ;CAS OUT CLOSE
20         OUTCHAR:    EQU 0BC95H          ;CAS OUT CHAR
21         DEL:        EQU 127             ;Caractere DELETE
22         BS:         EQU 8               ;Caractere Back Space
23         CR:         EQU 13              ;Carriage Return
24         LF:         EQU 10              ;Line Feed
25         CURS:       EQU 95              ;Caractere curseur
26         BLANC:      EQU 32              ;Caractere espace
27         BUF2K:      EQU 5000H          ;Buffer lecture
28         DICOMEM:    EQU 8000H          ;@ du dico
29         LIGNE:      EQU 07FBOH          ;ligne de texte

```

```

30      ;
31      MAX:      DS      1              ;Nbre max de caracteres
32      PBUF:      DS      2              ;Pointeur de buffer
33      NOM:      DS      12             ;Nom fichier+extension
34      DICO:      DS      12             ;Dictionnaire
35      LEN:      DS      1              ;longueur mot
36      CPT:      DS      1              ;Compteur de similitude
37      SAVDE:     DS      2              ;Sauveg registre DE
38      SAVHL:     DS      2              ;Sauveg registre HL
39      DETEMP:    DS      2              ;Sauv tempor DE
40      BUFLOT:    EQU     $
41                      DS      15
42      ;
43      ;-----
44      ;Definition des messages
45      ;-----
46      MES1:      EQU     $
47 9035 416E616C          DB      "Analyseur syntaxi
47 9039 79736575
47 903D 72207379
47 9041 6E746178
47 9045 69717565
47 9049 20
48 904A 64652066          DB      "de fichiers texte
48 904E 69636869
48 9052 65727320
48 9056 74657874
48 905A 65
49 905B 0D0A              DB      CR,LF
50 905D 2D2D2D2D          DB      "-----
50 9061 2D2D2D2D

```

```

50 9065 2D2D2D2D
50 9069 2D2D2D2D
50 906D 2D2D2D2D
51 9071 2D2D2D2D      DB  "-----
51 9075 2D2D2D2D
51 9079 2D2D2D2D
51 907D 2D2D2D2D
51 9081 2D2D
52 9083 0D0A0AFF      DB  CR,LF,LF,OFFH
53      ;-----
54      MES2:      EQU  $
55 9087 4E6F6D20      DB  "Nom du fichier te
55 908B 64752066
55 908F 69636869
55 9093 65722074
55 9097 65787465
56 909B 20612076      DB  " a verifier : ",0
56 909F 65726966
56 90A3 69657220
56 90A7 3A20FF
57      ;-----
58      MES3:      EQU  $
59 90AA 4E6F6D20      DB  "Nom du dictionnai
59 90AE 64752064
59 90B2 69637469
59 90B6 6F6E6E61
59 90BA 69726520
59 90BE 3A20FF
60      ;-----
61      MES4:      EQU  $
62 90C1 4C697374      DB  "Liste des erreurs

```

```

62 90C5 65206465
62 90C9 73206572
62 90CD 72657572
62 90D1 7320736F
62 90D5 757320
63 90DB 6C612066          DB  "la forme :",CR,LF
63 90DC 6F726D65
63 90E0 203A0D0A
64 90E4 20204D6F          DB  " Mot incorrect,M
64 90EB 7420696E
64 90EC 636F7272
64 90F0 6563742C
64 90F4 4D6F7420
64 90F8 636F7272
64 90FC 656374
65 90FF 0D0A0AFF          DB  CR,LF,LF,0FFH
66          ; -----
67          ALALI:      EQU  $
68 9103 0D0AFF          DB  CR,LF,0FFH
69          ;
70          ; -----
71          ;Acquisition du nom du fichier
72          ;et ouverture
73          ; -----
74          ;
75          DEBUT:      EQU  $                      ;Point d'entree
76          ;Init. des zones des noms
77          ;
78 9106 210690          LD   HL,NOM
79 9109 AF              XOR  A
80 910A 061B          LD   B,24                      ;Nbre d'octets a init

```



```

81          BINI1:      EQU  $
82 910C 77             LD   (HL),A
83 910D 23             INC  HL
84 910E 10FC          DJNZ BINI1
85                   ;
86 9110 3E02           LD   A,2
87 9112 CDOEBC         CALL MODE           ;Mode 2
88 9115 213590         LD   HL,MES1
89 9118 CDD892         CALL AFALPH         ;Affichage titre
90 911B 210391         LD   HL,ALALI
91 911E CDD892         CALL AFALPH         ;Saut de ligne
92 9121 218790         LD   HL,MES2
93 9124 CDD892         CALL AFALPH         ;Affichage question 1
94 9127 3E0C           LD   A,12
95 9129 320390         LD   (MAX),A       ;Saisie sur 12 car. max
96 912C 210690         LD   HL,NOM
97 912F 220490         LD   (PBUF),HL     ;Buffer de lecture
98 9132 CDE892         CALL SAISIE        ;Saisie nom du fichier
99 9135 210391         LD   HL,ALALI
100 9138 CDD892        CALL AFALPH        ;Passage a la ligne
101 913B 21AA90        LD   HL,MES3
102 913E CDD892        CALL AFALPH        ;Nom du dictionnaire
103 9141 3E0C          LD   A,12
104 9143 320390        LD   (MAX),A       ;Saisie sur 12 car. max
105 9146 211290        LD   HL,DICO
106 9149 220490        LD   (PBUF),HL     ;Buffer de lecture
107 914C CDE892        CALL SAISIE
108                   ;
109                   ;Affichage d'un message avant
110                   ;la liste des erreurs
111                   ;

```

```

112 914F 3E02          LD   A,2
113 9151 CD0EBC        CALL MODE
114 9154 21C190        LD   HL,MES4
115 9157 CDD892        CALL AFALPH
116                   ;
117 915A 0600          LD   B,0
118 915C AF            XOR   A
119 915D 211290        LD   HL,DICO
120                   BOURE1: EQU  $
121 9160 23            INC   HL
122 9161 04            INC   B
123 9162 7E            LD   A,(HL)
124 9163 B7            OR    A
125 9164 20FA          JR    NZ,BOURE1          ;Rech de longueur
126                   ;
127 9166 211290        LD   HL,DICO
128 9169 110050        LD   DE,BUF2K
129 916C CD77BC        CALL OPEN              ;Ouverture dico
130 916F 2100B0        LD   HL,DICOMEM
131 9172 CDB3BC        CALL DIRECT
132 9175 CD7ABC        CALL CLOSE
133                   ;
134 9178 0600          LD   B,0
135 917A AF            XOR   A
136 917B 210690        LD   HL,NOM
137                   BOURE2: EQU  $
138 917E 23            INC   HL
139 917F 7E            LD   A,(HL)
140 9180 04            INC   B
141 9181 B7            OR    A
142 9182 20FA          JR    NZ,BOURE2          ;Rech lgr du nom

```

```

143 9184 210690          LD   HL,NOM
144 9187 110050          LD   DE,BUF2K
145 918A CD77BC          CALL OPEN                ;Ouverture fichier
146                      ;
147                      ;-----
148                      ;Lecture de la prochaine ligne
149                      ;si elle existe
150                      ;-----
151                      ;
152          BOUO:         EQU   $
153 918D CD89BC          CALL TESTEOF                ;Du fichier texte
154 9190 D2A692          JP    NC,FIN                ;Fin du traitement
155 9193 21B07F          LD   HL,LIGNE
156 9196 3E00          LD   A,0
157 9198 0650          LD   B,80
158          RAZBUF:      EQU   $
159 919A 77            LD   (HL),A
160 919B 23            INC  HL
161 919C 10FC          DJNZ RAZBUF                ;RAZ Buffer Ligne
162                      ;
163 919E 21B07F          LD   HL,LIGNE
164          BISCAR:      EQU   $
165 91A1 CDB0BC          CALL CHAR
166 91A4 77            LD   (HL),A
167 91A5 23            INC  HL
168 91A6 FE0D          CP   13
169 91AB 20F7          JR   NZ,BISCAR
170 91AA CDB0BC          CALL CHAR                ;Pos sur dernier car
171 91AD 3E20          LD   A,20H
172 91AF 2B            DEC  HL
173 91B0 77            LD   (HL),A

```

```

174 91B1 3EFF          LD  A,OFFH
175 91B3 23            INC  HL
176 91B4 77            LD   (HL),A          ;Terminateur 20,FF
177                    ;
178                    ;-----
179                    ;Recherche d'un mot dans le
180                    ;dictionnaire
181                    ;-----
182                    ;
183                    BOU1:      EQU  $
184 91B5 1100B0          LD  DE,DICOMEN
185 91B8 ED532090        LD  (SAVDE),DE
186 91BC 21B07F          LD  HL,LIGNE
187 91BF 222290          LD  (SAVHL),HL
188                    BOU11:    EQU  $
189 91C2 AF              XOR  A
190 91C3 321E90          LD  (LEN),A
191 91C6 321F90          LD  (CPT),A
192                    BOU2:    EQU  $
193 91C9 1A              LD  A,(DE)
194 91CA 47              LD  B,A
195 91CB 7E              LD  A,(HL)
196 91CC FE20            CP   20H
197 91CE 2B1C            JR   Z,FINCOMP      ;Analyse
198 91D0 FEFF            CP   OFFH
199 91D2 2B1B            JR   Z,FINCOMP
200 91D4 3A1E90          LD  A,(LEN)
201 91D7 3C              INC  A
202 91D8 321E90          LD  (LEN),A          ;LEN+1
203 91DB 7E              LD  A,(HL)
204 91DC BB              CP   B

```

205 91DD 2804		JR	Z,PLUSUN	
206	BOU3:	EQU	\$	
207 91DF 23		INC	HL	
208 91E0 13		INC	DE	
209 91E1 18E6		JR	BOU2	
210	PLUSUN:	EQU	\$	
211 91E3 3A1F90		LD	A, (CPT)	
212 91E6 3C		INC	A	
213 91E7 321F90		LD	(CPT),A	
214 91EA 18F3		JR	BOU3	
215	;			
216	FINCOMP:	EQU	\$	
217 91EC 3A1E90		LD	A, (LEN)	
218 91EF 3D		DEC	A	
219 91F0 47		LD	B,A	
220 91F1 3A1F90		LD	A, (CPT)	
221 91F4 B8		CP	B	
222 91F5 CA2C92		JP	Z,LITIGE	;Mot incorrect
223	BOU7:	EQU	\$	
224 91F8 ED5B2090		LD	DE, (SAVDE)	
225	BOU6:	EQU	\$	
226 91FC 13		INC	DE	
227 91FD 1A		LD	A, (DE)	
228 91FE FEFF		CP	OFFH	
229 9200 280E		JR	Z,BOU4	
230 9202 FE20		CP	20H	
231 9204 20F6		JR	NZ,BOU6	
232	BOU5:	EQU	\$	
233 9206 13		INC	DE	
234 9207 ED532090		LD	(SAVDE),DE	;Proch mot dans dico
235 920B 2A2290		LD	HL, (SAVHL)	

```

236 920E 18B2      JR    BOU11
237                BOU4: EQU  $
238 9210 AF        XOR   A
239 9211 321F90     LD    (CPT),A
240 9214 321E90     LD    (LEN),A
241 9217 23         INC   HL
242 9218 7E         LD    A,(HL)
243 9219 FEFF       CP    OFFH
244 921B 280C       JR    Z,FINLIGNE
245 921D 110080     LD    DE,DICOMEM
246 9220 ED532090   LD    (SAVDE),DE
247 9224 222290     LD    (SAVHL),HL
248 9227 18A0       JR    BOU2
249                FINLIGNE: EQU  $
250 9229 C38D91     JP    BOU0                ;Lecture ligne suivante
251                LITIGE: EQU  $
252 922C 1A         LD    A,(DE)
253 922D FE20       CP    20H
254 922F 2806       JR    Z,LITSUI
255 9231 FEFF       CP    OFFH
256 9233 2802       JR    Z,LITSUI
257 9235 18C1       JR    BOU7
258                LITSUI: EQU  $
259 9237 222290     LD    (SAVHL),HL
260 923A ED532490   LD    (DETEMP),DE
261 923E CDAA92     CALL  BEEP
262                ;Affichage mots
263 9241 012690     LD    BC,BUFMDT
264                AMOTO: EQU  $
265 9244 2B         DEC   HL
266 9245 7E         LD    A,(HL)

```

267 9246 FE20	CP	20H	
268 9248 2807	JR	Z,AMOT1	
269 924A 7D	LD	A,L	
270 924B FEB0	CP	0B0H	
271 924D 2803	JR	Z,AMOT3	
272 924F 18F3	JR	AMOT0	
273	AMOT1:	EQU	\$
274 9251 23	INC	HL	
275	AMOT3:	EQU	\$
276 9252 7E	LD	A,(HL)	
277	AMOT4:	EQU	\$
278 9253 02	LD	(BC),A	
279 9254 23	INC	HL	
280 9255 03	INC	BC	
281 9256 7E	LD	A,(HL)	
282 9257 FE20	CP	20H	
283 9259 20F8	JR	NZ,AMOT4	
284 925B 3EFF	LD	A,0FFH	
285 925D 02	LD	(BC),A	
286 925E 212690	LD	HL,BUFMOT	
287 9261 CDD892	CALL	AFALPH	
288	;		
289 9264 3E2C	LD	A,44	;Separateur
290 9266 CD5ABB	CALL	PRINT	;Affichage virgule
291	;		
292 9269 012690	LD	BC,BUFMOT	
293	AMOT00:	EQU	\$
294 926C 1B	DEC	DE	
295 926D 1A	LD	A,(DE)	
296 926E FE20	CP	20H	
297 9270 280B	JR	Z,AMOT10	
298 9272 7A	LD	A,D	

299 9273 FE80		CP	80H
300 9275 2004		JR	NZ,AMDSUI
301 9277 7B		LD	A,E
302 9278 B7		OR	A
303 9279 2B03		JR	Z,AMDT30
304	AMDSUI:	EQU	\$
305 927B 18EF		JR	AMDT00
306	AMDT10:	EQU	\$
307 927D 13		INC	DE
308	AMDT30:	EQU	\$
309 927E 1A		LD	A,(DE)
310	AMDT40:	EQU	\$
311 927F 02		LD	(BC),A
312 9280 13		INC	DE
313 9281 03		INC	BC
314 9282 1A		LD	A,(DE)
315 9283 FE20		CP	20H
316 9285 2B06		JR	Z,AMDT50
317 9287 FEFF		CP	OFFH
318 9289 2B02		JR	Z,AMDT50
319 928B 18F2		JR	AMDT40
320	AMDT50:	EQU	\$
321 928D 3EFF		LD	A,OFFH
322 928F 02		LD	(BC),A
323 9290 212690		LD	HL,BUFMDT
324 9293 CDD892		CALL	AFALPH
325 9296 210391		LD	HL,ALALI
326 9299 CDD892		CALL	AFALPH
327			
328 929C 2A2290		LD	HL,(SAVHL)


```

329 929F ED5B2490      LD    DE, (DETEMP)
330 92A3 C31092        JP    BOU4          ;Suite traitement
331                    ;
332                    FIN:      EQU    $          ;Fin du programme
333 92A6 CD7ABC         CALL CLOSE      ;Ferm fichier texte
334 92A9 C9            RET
335                    ;
336                    ;-----
337                    ;Emission d'un BEEP sonore
338                    ;-----
339                    ;Entree: aucune
340                    ;Sortie: AF, BC et HL effaces
341                    ;-----
342                    ;
343                    BEEP:      EQU    $          ;Point d'entree
344 92AA E5             PUSH HL
345 92AB 3E00           LD     A,0
346 92AD 0E00           LD     C,0
347 92AF CD34BD         CALL SOUND
348 92B2 3E01           LD     A,1
349 92B4 0E01           LD     C,1
350 92B6 CD34BD         CALL SOUND          ;Frequence
351 92B9 3E08           LD     A,8
352 92BB 0E06           LD     C,6
353 92BD CD34BD         CALL SOUND          ;Amplitude
354 92C0 3E07           LD     A,7
355 92C2 0E08           LD     C,8
356 92C4 CD34BD         CALL SOUND          ;Validation registre A
357                    ;
358 92C7 21FFFF         LD     HL,0FFFFH
359                    BOU:      EQU    $

```

```

360 92CA 2B          DEC  HL
361 92CB 7C          LD   A,H
362 92CC B5          OR   L
363 92CD 20FB        JR   NZ,BOU
364                  ;
365 92CF 3E08        LD   A,8
366 92D1 0E00        LD   C,0
367 92D3 CD34BD      CALL SOUND          ;Fin du BEEP
368 92D6 E1          POP  HL
369 92D7 C9          RET
370                  ;
371                  ;-----
372                  ;Affichage d'un texte alphanum.
373                  ;-----
374                  ;Entree: @ de depart dans HL
375                  ;Sortie: aucun registre modifie
376                  ;-----
377                  ;
378  AFALPH:          EQU  $              ;Point d'entree
379 92D8 E5          PUSH HL
380 92D9 F5          PUSH AF
381  ME1:            EQU  $              ;Boucle d'affichage
382 92DA 7E          LD   A,(HL)
383 92DB FEFF        CP   OFFH
384 92DD 2806        JR   Z,ME2          ;Fin d'affichage
385 92DF CD5ABB      CALL PRINT          ;Affichage caractere
386 92E2 23          INC  HL            ;Caractere suivant
387 92E3 18F5        JR   ME1
388  ME2:            EQU  $
389 92E5 F1          POP  AF
390 92E6 E1          POP  HL

```

```

391 92E7 C9                      RET
392                               ;
393                               ;-----
394                               ;Saisie de caracteres
395                               ;-----
396                               ;Entree: (MAX)=Nbre de caracteres
397                               ;Sortie: Aucun registre ecrase
398                               ;-----
399                               ;
400      SAISIE:      EQU  $                      ;Point d'entree
401 92EB 3A0390      LD  A,(MAX)
402 92EB 57          LD  D,A                      ;Nbre de caract. a lire
403 92EC 010000      LD  BC,0                    ;Index dans le buffer
404      S1:         EQU  $
405 92EF 2A0490      LD  HL,(PBUF)                ;Buffer de lecture
406 92F2 CD06BB      CALL READ                    ;Lecture 1 caractere
407 92F5 FE0D      CP   CR                      ;Carriage Return ?
408 92F7 2B3C      JR   Z,S3                    ;Oui => fin de saisie
409 92F9 FE7F      CP   DEL                     ;DElete
410 92FB 2B18      JR   Z,S2                    ;Oui => Traitement DEL
411 92FD F5          PUSH AF
412 92FE 3E08      LD  A,BS
413 9300 CD5ABB      CALL PRINT
414 9303 F1          POP  AF
415 9304 CD5ABB      CALL PRINT
416 9307 09          ADD  HL,BC
417 9308 77          LD   (HL),A                ;Sauvegarde
418 9309 03          INC  BC
419 930A 3E5F      LD  A,CURS
420 930C CD5ABB      CALL PRINT
421 930F 79          LD  A,E

```

```

422 9310 BA          CP    D
423 9311 20DC        JR    NZ,S1          ;Suite de la saisie
424 9313 1820        JR    S3            ;Fin de saisie
425                S2:    EQU    *
426 9315 79          LD    A,C
427 9316 B7          OR    A
428 9317 28D6        JR    Z,S1          ;DEL non accepte
429 9319 0B          DEC    BC
430 931A 3E0B        LD    A,BS
431 931C CD5ABB      CALL PRINT          ;Back Space
432 931F 3E20        LD    A,BLANC
433 9321 CD5ABB      CALL PRINT          ;Effacement caractere
434 9324 3E0B        LD    A,BS
435 9326 CD5ABB      CALL PRINT          ;Back Space
436 9329 3E0B        LD    A,BS
437 932B CD5ABB      CALL PRINT          ;Back Space
438 932E 3E5F        LD    A,CURS
439 9330 CD5ABB      CALL PRINT          ;Affichage curseur
440 9333 18BA        JR    S1
441                S3:    EQU    *
442 9335 3E0B        LD    A,BS
443 9337 CD5ABB      CALL PRINT          ;Back Space
444 933A 3E20        LD    A,BLANC
445 933C CD5ABB      CALL PRINT          ;Effacement caractere
446 933F C9          RET
447                ;
448                ;
449                END

```

- Lignes 9 à 29 : Déclaration de constantes.
- Lignes 31 à 41 : Déclaration de variables et buffers.
- Lignes 43 à 69 : Divers messages affichés par le programme.
- Lignes 70 à 107 : Acquisition du nom des fichiers texte et dictionnaire.
- Lignes 109 à 115 : Affichage du texte précédent la liste des erreurs.
- Lignes 117 à 145 : Ouverture et lecture des fichiers dictionnaire et (partiellement) texte.
- Lignes 147 à 176 : Lecture d'une ligne dans le fichier texte.
- Lignes 178 à 330 : Recherche d'erreurs dans le texte.
- Lignes 332 à 334 : Fermeture du fichier texte et retour au basic.
- Lignes 336 à 369 : Utilitaire d'émission d'un beep sonore lorsqu'une erreur est détectée.
- Lignes 371 à 391 : Utilitaire d'affichage d'un texte alphanumérique.
- Lignes 393 à 446 : Utilitaire de lecture d'un texte alphanumérique délimité.

Le correcteur est activé sous basic par l'instruction **CALL &9000**. Supposons que le dictionnaire contienne les mots suivants :

CHAT, MERE, MICHEL, APPETIT

Supposons que le texte à corriger soit le suivant :

**LE CHAT DE LA LERE MACHEL
MANGE AVEC APPITIT**

Le correcteur orthographique détectera trois erreurs, affichera les mots erronés suivis de leur syntaxe exacte :

**LERE,MERE
MACHEL,MICHEL
APPITIT,APPETIT**

Comme toujours, voici la version chargeur Basic du programme Assembleur ci-dessus :

```

1000 '=====
1010 ' CHARGEUR HEXA DU CORRECTEUR ORTHOGRAPHIQUE
1020 '=====
1030 FOR I=&9000 TO &933F
1040   READ A$
1050   A=VAL("&" + A$)
1060   POKE I,A
1070 NEXT I
1080 CALL &9000
1090 END
1100 DATA C3,6,91,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1110 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

```

```

1120 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1130 DATA 0,0,0,0,0,41,6E,61,6C,79,73,65,75,72,20,73
1140 DATA 79,6E,74,61,78,69,71,75,65,20,64,65,20,66,69,63
1150 DATA 68,69,65,72,73,20,74,65,78,74,65,D,A,2D,2D,2D
1160 DATA 2D,2D,2D,2D,2D,2D,2D,2D,2D,2D,2D,2D,2D,2D,2D
1170 DATA 2D,2D,2D,2D,2D,2D,2D,2D,2D,2D,2D,2D,2D,2D,2D
1180 DATA 2D,2D,2D,D,A,A,FF,4E,6F,6D,20,64,75,20,66,69
1190 DATA 63,68,69,65,72,20,74,65,78,74,65,20,61,20,76,65
1200 DATA 72,69,66,69,65,72,20,3A,20,FF,4E,6F,6D,20,64,75
1210 DATA 20,64,69,63,74,69,6F,6E,6E,61,69,72,65,20,3A,20
1220 DATA FF,4C,69,73,74,65,20,64,65,73,20,65,72,72,65,75
1230 DATA 72,73,20,73,6F,75,73,20,6C,61,20,66,6F,72,6D,65
1240 DATA 20,3A,D,A,20,2D,4D,6F,74,20,69,6E,63,6F,72,72
1250 DATA 65,63,74,2C,4D,6F,74,20,63,6F,72,72,65,63,74,D
1260 DATA A,A,FF,D,A,FF,21,6,90,AF,6,18,77,23,10,FC
1270 DATA 3E,2,CD,E,BC,21,35,90,CD,DB,92,21,3,91,CD,DB
1280 DATA 92,21,87,90,CD,DB,92,3E,C,32,3,90,21,6,90,22
1290 DATA 4,90,CD,EB,92,21,3,91,CD,DB,92,21,AA,90,CD,DB
1300 DATA 92,3E,C,32,3,90,21,12,90,22,4,90,CD,EB,92,3E
1310 DATA 2,CD,E,BC,21,C1,90,CD,DB,92,6,0,AF,21,12,90
1320 DATA 23,4,7E,B7,20,FA,21,12,90,11,0,50,CD,77,BC,21
1330 DATA 0,80,CD,B3,BC,CD,7A,BC,6,0,AF,21,6,90,23,7E
1340 DATA 4,B7,20,FA,21,6,90,11,0,50,CD,77,BC,CD,B9,BC
1350 DATA D2,A6,92,21,80,7F,3E,0,6,50,77,23,10,FC,21,80
1360 DATA 7F,CD,80,BC,77,23,FE,D,20,F7,CD,80,BC,3E,20,2B
1370 DATA 77,3E,FF,23,77,11,0,80,ED,53,20,90,21,80,7F,22
1380 DATA 22,90,AF,32,1E,90,32,1F,90,1A,47,7E,FE,20,28,1C
1390 DATA FE,FF,28,18,3A,1E,90,3C,32,1E,90,7E,B8,28,4,23
1400 DATA 13,18,E6,3A,1F,90,3C,32,1F,90,18,F3,3A,1E,90,3D
1410 DATA 47,3A,1F,90,B8,CA,2C,92,ED,5B,20,90,13,1A,FE,FF
1420 DATA 28,E,FE,20,20,F6,13,ED,53,20,90,2A,22,90,18,B2
1430 DATA AF,32,1F,90,32,1E,90,23,7E,FE,FF,28,C,11,0,80
1440 DATA ED,53,20,90,22,22,90,18,A0,C3,8D,91,1A,FE,20,28
1450 DATA 6,FE,FF,28,2,18,C1,22,22,90,ED,53,24,90,CD,AA
1460 DATA 92,1,26,90,2B,7E,FE,20,28,7,7D,FE,80,28,3,18
1470 DATA F3,23,7E,2,23,3,7E,FE,20,20,F8,3E,FF,2,21,26
1480 DATA 90,CD,DB,92,3E,2C,CD,5A,BB,1,26,90,1B,1A,FE,20
1490 DATA 28,B,7A,FE,80,20,4,7B,B7,28,3,18,EF,13,1A,2
1500 DATA 13,3,1A,FE,20,28,6,FE,FF,28,2,18,F2,3E,FF,2
1510 DATA 21,26,90,CD,DB,92,21,3,91,CD,DB,92,2A,22,90,ED
1520 DATA 5B,24,90,C3,10,92,CD,7A,BC,C9,E5,3E,0,E,0,CD
1530 DATA 34,BD,3E,1,E,1,CD,34,BD,3E,8,E,6,CD,34,BD
1540 DATA 3E,7,E,8,CD,34,BD,21,FF,FF,2B,7C,B5,20,FB,3E
1550 DATA 8,E,0,CD,34,BD,E1,C9,E5,F5,7E,FE,FF,28,6,CD
1560 DATA 5A,BB,23,18,F5,F1,E1,C9,3A,3,90,57,1,0,0,2A
1570 DATA 4,90,CD,6,BB,FE,D,28,3C,FE,7F,28,18,F5,3E,8
1580 DATA CD,5A,BB,F1,CD,5A,BB,9,77,3,3E,5F,CD,5A,BB,79
1590 DATA BA,20,DC,18,20,79,B7,28,D6,B,3E,8,CD,5A,BB,3E
1600 DATA 20,CD,5A,BB,3E,8,CD,5A,BB,3E,8,CD,5A,BB,3E,5F
1610 DATA CD,14,BB,18,BA,3E,8,CD,5A,BB,3E,20,CD,5A,BB,C9

```

Les données de checksum correspondantes sont les suivantes :

```

5B 0 0 4B 29 8 D2 D2 BD D6 23 9B A5 FA 92 BC 5B 55 EE CF A4 C0 C0 A2 6 6B DD 4
6B CB 4C 99 19 D8 C3 4C B2 FB 24 E6 F1 CA 45 1A F3
D6 35 8F 38 93 F5 EC

```

Comme pour le programme Assembleur, vous devez avoir créé un fichier dictionnaire et un fichier texte stockés sur disque, sous forme binaire, à l'aide des deux programmes précédents avant de pouvoir utiliser le chargeur Basic.