MISE EN MARCHE DU SYSTÈME SUR T07

UTILISEZ LA FACE ROUGE DE LA CASSETTE

Connectez le micro-ordinateur :
— à votre téléviseur ;
— au lecteur enregistreur de programmes.

Mettez la cartouche BASIC dans son logement.

Mettez sous tension :
— le téléviseur ;
— le T07. Le témoin lumineux rouge s'allume. Vous avez à l'écran le « menu » initial.

Si le programme utilise le crayon optique, appuyez sur la touche 3 du clavier et réglez le crayon. (Si le crayon optique ne réagit pas, agissez sur le niveau de luminosité de votre téléviseur).

Introduisez la cassette (Face ROUGE) dans votre lecteur de programmes. Rembobinez la cassette. Mettez le compteur à zéro.

Appuyez sur la touche ▶ du lecteur pour le mettre en mode « lecture ».

Pour charger le programme, tapez la touche 2 du clavier, ou pointez l'écran avec le lecteur optique.

MISE EN MARCHE DU SYSTÈME SUR M05

UTILISEZ LA FACE VERTE DE LA CASSETTE

Connectez le micro-ordinateur :
— à votre téléviseur ;
— au lecteur enregistreur de programmes.

Mettez sous tension. Vous avez à l'écran :

```
M05 BASIC 1.Ø
(C) Microsoft 1984
OK
—
```

Introduisez la cassette (Face VERTE) dans votre lecteur de programmes.

Rembobinez la cassette. Mettez le compteur à zéro.

Appuyez sur la touche ▶ du lecteur pour le mettre en mode « lecture ».

Pour charger le programme, tapez au clavier RUN « CASS: » puis appuyez sur la touche ENTREE.

MICROPROCESSEUR

J. FRANÇOIS

SOMMAIRE

(*) Pour chaque type d'instructions, on trouvera un complément relatif à des instructions non mentionnées mais utilisées dans les microprocesseurs actuels.

PRÉSENTATION GÉNÉRALE

OBJECTIFS.

Ce logiciel est destiné à toute personne voulant découvrir les principes internes de fonctionnement d'un micro-ordinateur et, par là-même, trouver réponse à la question : "Qu'y-a-t-il à l'intérieur d'un ordinateur ?".

Les objectifs du logiciel "MICROPROCESSEUR" sont les suivants :

— comprendre le rôle et le fonctionnement des principaux éléments composant l'architecture des microprocesseurs ;

— s'initier aux principes du langage machine et du langage assembleur ;

— s'exercer aux différents modes de représentation des nombres et des adresses ;

— visualiser, de façon claire et simple, les relations entre un programme (que vous écrirez) et les opérations élémentaires effectuées par le microprocesseur (*).

Ce logiciel constitue aussi une excellente initiation pour tous ceux qui souhaiteront découvrir par la suite le langage machine et l'assembleur.

PRINCIPES.

Le logiciel MICROPROCESSEUR comprend deux parties :

— *un module d'initiation* permettant de comprendre le fonctionnement d'un microprocesseur et de se familiariser avec les bases du langage machine ;

— *un module d'exercice,* véritable simulation d'un microprocesseur 6 bits, permettant de tester et de modifier des programmes écrits en langage machine.

L'initiation et la programmation se font à partir d'une structure simplifiée de microprocesseur 6 bits (c'est-à-dire utilisant des "mots" formés de 6 éléments binaires).

Il est conseillé à l'utilisateur de faire entièrement la première partie avant de passer à la seconde.

(*) La définition des principaux termes en italique est rappelée dans le lexique en fin de manuel.

GÉNÉRALITÉS.

La manipulation du logiciel se fait uniquement à l'aide du clavier.
Au début du programme, le menu suivant apparaît :

```
1 INITIATION
2 EXERCICE
```

Choisissez votre module en tapant **1** ou **2**.

Pour interrompre un module ou en changer, appuyez sur la touche **RAZ**. Le menu suivant apparaît :

```
1 RECOMMENCER
2 AUTRE MODULE
Ø ARRÊT
```

Vous pouvez ainsi :

— reprendre le module dans sa totalité (tapez **1**) ;

— lancer automatiquement le chargement de l'autre module (tapez **2**) ;

— arrêter le programme (tapez **0**).

LES SYMBOLES.

Dans la partie gauche de l'écran, réservée aux texte, vous verrez souvent apparaître en bas à droite, l'un des symboles suivants : ☞, <>.

☞ vous invite à passer à la suite. Il suffit pour cela d'appuyer sur la touche **ENTRÉE**.

<> vous permet d'effectuer le choix entre :

— recommencer la page que vous venz d'étudier. Pour cela, appuyez sur la touche ← ;

— passer à la page suivante, en appuyant sur la touche **ENTRÉE**.

CONTENU.

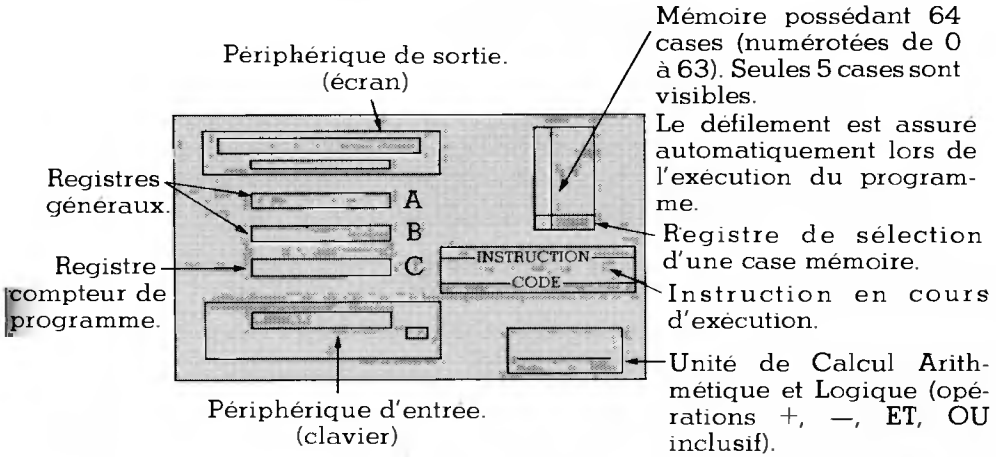
Le module d'initiation visualise et explique, de façon progressive, le rôle et le fonctionnement des principaux constituants du microprocesseur de l'unité centrale :

— les mémoires ;

— les registres : registre d'adresse, registres généraux (A et B), registre compteur de programme (C) ;

— l'Unité Arithmétique et Logique (U.A.L.) ;

— les périphériques (écran et clavier).



EXERCICES D'APPLICATION.

Ils illustrent, par l'exemple, les opérations effectuées par le microprocesseur, lors de l'exécution de quelques instructions machine, que vous compléterez :

1. INSTRUCTIONS D'ENTRÉE/SORTIE (ENTRE ., SORT .) :

ENTRE A : tapez sur une touche du clavier. Le caractère apparaît en clair et sous forme codée binaire alphanumérique, puis est transféré dans le registre A.

SORT A : inversement, cette instruction permet l'affichage du caractère à l'écran, à partir de son code binaire alphanumérique contenu dans le registre A.

2. TRANSFERT DE DONNÉES :

MET A EN . . : taper l'instruction **M** **A** puis un nombre compris entre Ø et 63, puis validez par la touche **ENTRÉE**.

Remarque : lorsque des exercices d'application vous sont proposés, ou lorsque vous taperez vous-mêmes vos propres programmes, vous vous apercevrez qu'il suffit de taper la première lettre de l'instruction (ou les deux ou trois premières lettres pour les instructions du type SORT, SIN, SIZ, SIP) pour voir apparaître sur l'écran le mot ou toute l'instruction en clair.

Exemple : si vous tapez sur la touche **H**, vous obtenez à l'écran l'instruction HALTE.

A = (?) B = (?) : taper une des deux instructions en remplaçant "?" par un nombre compris entre Ø et 63.

3. CALCULS :

A = A + B ou B = A + B : ces opérations s'effectuent sur des nombres codés en binaire. Additionnez les valeurs contenues dans les registres A et B et tenant compte des principes de calcul en binaire.

+	Ø	1
Ø	Ø	1
1	1	Ø

ET	Ø	1
Ø	Ø	Ø
1	Ø	1

OU	Ø	1
Ø	Ø	1
1	1	1

Attention : en binaire, 1 + 1 = Ø sans oublier la retenue de 1.

4. INSTRUCTIONS DE BRANCHEMENT :

Les branchements peuvent être conditionnels ou inconditionnels :

— *VA EN XX* est exécuté si X est compris entre Ø et 63.

— *SIZ VA EN XX* est exécuté uniquement si le résultat de l'opération précédente est Ø.

5. PROGRAMME D'APPLICATION :

Ce programme permet de suivre, pas à pas, la logique de l'enchaînement des instructions.

Le programme est chargé en mémoire à partir de l'adresse Ø.

— L'instruction *SORT A* affiche en clair à l'écran le caractère dont le code binaire alphanumérique est contenu dans le registre A.

— L'instruction *DEC B* permet de diminuer d'une unité le contenu du registre B (instruction équivalente à B = B — 1).

— L'intruction *COMP B-11* effectue la comparaison entre le contenu du registre B et la valeur décimale 11. Contrairement à l'opération de soustraction, le registre B n'est pas affecté par le résultat (la comparaison est effectuée dans l'U.A.L.).

— *SIP VA EN 2* (lire SI P : SI Positif) est une instruction conditionnelle. En fonction du résultat de la dernière opération effectuée (COMP B-11), le branchement sera effectué en adresse 2 (si B-11 > Ø) ou non.

— *HALTE* est l'instruction de fin de programme.

En fin d'exécution de programme et à la demande de l'ordinateur, tapez **O** (oui) ou **N** (non) selon votre choix :

O pour recommencer le programme d'application.

N pour passer au module de simulation.

LE MODULE D'EXERCICE DE SIMULATION

Il s'agit maintenant d'écrire votre propre programme en langage machine.

MENU.

Le module d'exercice vous permet de travailler selon 4 modes différents :

P (pour programmation).
M (pour modification de programme).
L (pour lister le programme).
E (pour exécution du programme).
F (pour retour au menu ou arrêt de programme).

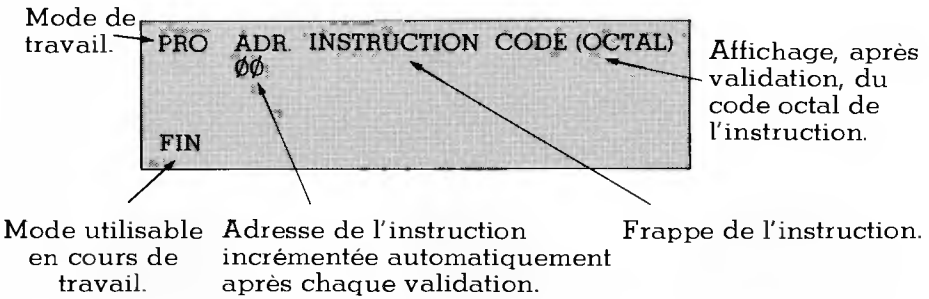
Choisissez votre mode de travail en tapant la lettre correspondante.

A tout moment, en particulier en fin de programmation ou de modification, vous pouvez revenir au menu proposé en tapant **F** pour fin, puis en validant par **ENTRÉE**.

MODES DE TRAVAIL.

1. LE MODE PROGRAMMATION :

Il est utilisé pour introduire un nouveau programme. A la fin de chaque ligne d'instruction que vous aurez tapée, appuyez sur la touche **ENTRÉE** pour valider votre commande et pour passer à la ligne suivante.



Le code octal de l'instruction correspond à son écriture en base 8 (voir liste des instructions en page ..).

Pour les instructions courtes (un mot de 6 bits), le code octal est formé d'un nombre compris entre Ø et 37.

Exemple : ENTRE A a pour code octal Ø4.

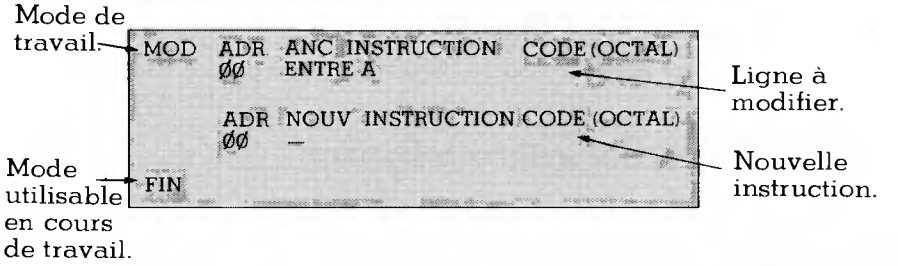
Pour les instructions longues, le code octal est formé de deux nombres :

- le premier correspond au code de l'instruction (de 4Ø à 77);
- le deuxième correspond à l'adresse ou la donnée qui suit l'instruction.

Exemples : VA EN 22 a pour code octal 4Ø26.
A = 45 a pour code octal 5Ø55.

2. LE MODE MODIFICATION :

Il est utilisé pour modifier un programme existant. Dans ce mode, l'ordinateur affiche la première instruction du programme.



Pour sélectionner l'instruction à modifier, taper ↓ ou ↑ jusqu'à obtenir la ligne comme précédemment, puis taper sur la touche **EFF**.

Pour insérer une instruction dans le programme, sélectionner la ligne immédiatement suivante grâce aux touches ↓ et ↑, puis taper sur la touche **INS**; votre ligne est insérée et vous pouvez taper la nouvelle instruction.

Remarque : insertion et suppression entraînent une renumérotation automatique des adresses.

3. LE MODE LISTE :

Ce mode est utilisé pour obtenir la liste complète des instructions de votre programme. Pour assurer le défilement de votre programme, utiliser les touches ↓ et ↑. En pratique, lister un programme permet de retrouver un numéro de ligne.

4. LE MODE EXÉCUTION :

En mode exécution, l'ordinateur affiche l'état des registres et des mémoires, visualise les transferts de données ainsi que les opérations effectuées à chaque pas de programme. A la fin de chaque pas, le symbole ☞ apparaît en bas à droite de l'écran. Taper ENTRÉE permet de continuer.

L'exécution peut être effectuée rapidement (R) ou lentement (L). Dans ce dernier cas, des explications sont données sur l'opération en cours, dans la partie inférieure de l'écran.

Il est possible de changer la vitesse d'exécution à la fin de chaque pas de programme, en tapant L ou R.

UTILISATION DES INSTRUCTIONS.

Les instructions sont au nombre de 64. Vous pouvez consulter leur liste détaillée à la fin de ce manuel, ou à l'écran en appuyant sur la touche RAZ.

On distingue :

— *les instructions courtes* (n° Ø à 31) n'utilisant qu'un mot de 6 bits, donc qu'une seule case mémoire.

— *les instructions longues* (n° 32 à 63) utilisant deux mots de 6 bits donc deux cases mémoire. Le premier mot définie l'instruction, le deuxième une adresse ou une donnée.

C'est pourquoi le numéro de ligne est augmenté automatiquement de 1 ou de 2.

Exemple :
```
ØØ   A = 12
Ø2   ENTRE B
Ø3   B = B — 5
Ø5   A = A + B
Ø6   HALTE
```


En mode programmation ou modification, chaque instruction peut être écrite :

— soit en clair (ex. MET A EN 22) : langage assembleur ;

— soit en utilisant son code binaire, octal ou décimal. Pour les instructions utilisant deux mots, vous aurez donc besoin de deux lignes d'instruction (code de l'instruction puis code de la donnée ou de l'adresse) : langage machine.

Attention : a priori, l'ordinateur suppose que vous entrez les nombres en base 1Ø (code décimal). Les conventions suivantes ont donc été prises :

— pour entrer un nombre en binaire, tapez % avant son code binaire ;

— pour utiliser le code octal (base 8), tapez d'abord la lettre O (et non le chiffre Ø) suivi du code ;

— pour entrer directement le code alphanumérique d'une lettre ou d'un symbole, tapez " suivi du caractère.

Exemple : A = B sera codé %ØØ1ØØØ en binaire, Ø1Ø en octal, 8 en décimal. Le nombre décimal 22 pourra être codé en binaire : %Ø1Ø11Ø, en octal Ø26.

Quel que soit le code utilisé pour représenter les instructions, il est compris entre Ø et 63 pour les codes décimal et binaire, Ø et 77 pour le code octal.

Attention : le microprocesseur 6 bits ne possède que 64 cases mémoire (numérotées de Ø à 63) ; vous ne pourrez pas écrire de programmes plus longs.

Une fois l'instruction tapée, elle est validée par la touche ENTRÉE ; le code octal de l'instruction s'affiche alors à droite de celle-ci.

LISTE DES INSTRUCTIONS

Le tableau répertorié en page 28 donne le jeu complet des instructions de notre microprocesseur 6 bits, avec leur numéro (ou code décimal), leurs codes octal et binaire et leurs caractéristiques.

On peut regrouper ces instructions en quatre catégories :

LES INSTRUCTIONS D'ENTRÉE/SORTIE.

Il y en a 4 : 2 d'entrées (clavier) et 2 de sorties (écran).

Chacune d'elles utilise l'un ou l'autre des registres généraux (A ou B). Lors de l'exécution des instructions d'entrée/sortie, c'est toujours le code alphanumérique qui est pris en compte (voir tableau en fin de manuel) :

— en entrée, chaque touche frappée au clavier est convertie en code alphanumérique. Par exemple, lors de l'exécution de l'instruction ENTRE A, si vous appuyez sur la touche **0**, le registre A ne contiendra pas cette valeur Ø mais le code alphanumérique de la touche Ø soit 16 (code décimal) ;

— en sortie, inversement, le code alphanumérique sélectionne le caractère affiché.

Compléments :

Dans la plupart des microprocesseurs, les caractères sont codés selon le code ASCII qui utilise au moins 7 bits. Le codage sur 8 bits permet, de plus, de coder des caractères spéciaux (graphiques par exemple).

Les microprocesseurs peuvent posséder plusieurs périphériques d'entrée (clavier, manette, crayon optique) et de sortie (écran, imprimante, générateur de sons). Les instructions d'entrée/sortie devront alors nécessairement spécifier le numéro du "port d'entrée sortie" sur lequel est branché le périphérique que l'on veut utiliser.

LES INSTRUCTIONS DE TRANSFERT.

Il y en a 14. Chacune d'elles doit préciser la source d'information et sa destination. Elles se différencient par leur mode d'adressage, c'est-à-dire la façon dont est calculée l'adresse de la case mémoire à sélectionner.

TYPE DE TRANSFERT	MODE D'ADRESSAGE	DÉFINITION	SYNTAXE EXEMPLES D'INSTR.
De registre à registre.	Inhérent.	Seuls les registres généraux sont utilisés.	A = B B = A
De mémoire à registre.	Indexé.	L'adresse est définie par un autre registre général.	L'adresse est mise entre parenthèses. A = (B) B = (A)
	Immédiat.	L'adresse est définie par le registre C. La donnee transférée suit immédiatement le code de l'instruction.	A = XX B = XX
	Direct.	L'adresse est définie par l'instruction elle-même. L'adresse de la donnée à transferer suit le code instruct.	L'adresse est mise entre parenthèses. A = (XX) B = (XX)
De registre à mémoire (opération inverse de la précédente).	Indexé.	L'adresse est définie par un autre registre général	L'adresse est mise entre parenthèses. (A) = B (B) = A
	Immédiat.	L'adresse est définie par le mot qui suit l'instruction.	MET A EN XX NET B EN XX
De mémoire à mémoire.	La source est définie par adressage immédiat (la donnée suit immédiatement le code de l'instruction). La destination est définie par adressage indexé (son adresse est contenue dans un registre général).		2 instructions : (A) = XX (B) = XX

Compléments :

D'autres modes d'adressage sont généralement accessibles sur les microprocesseurs existants :

— dans l'adressage relatif (ou indexé relatif), l'adresse est calculée en additionnant ou en soustrayant une certaine valeur au contenu du registre;

— dans l'adressage indirect, le calcul d'adressage se fait en deux temps.

1°) L'adresse définie par l'instruction en mode direct ou relatif sélectionne une case mémoire.

2°) Cette case mémoire est lue mais n'est pas transférée vers la destination : elle contient, non plus la donnée à transférer, mais uniquement son adresse. Il faut donc aller chercher la donnée à l'adresse indiquée.

Le microprocesseur effectue donc les opérations suivantes :

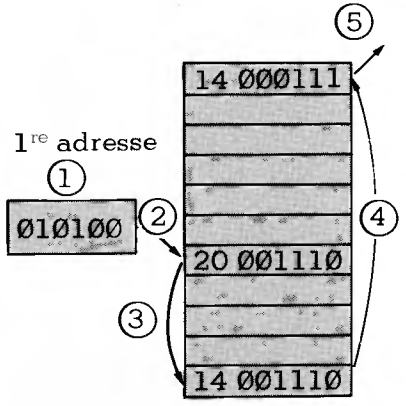
① : calcul de la première adresse selon le mode direct ou relatif.

② : sélection de la case mémoire correspondante.

③ : transfert du contenu de la case mémoire vers le registre d'adresse (2e adresse).

④ : sélection de la case mémoire correspondante.

⑤ : transfert de son contenu vers la destination.



LES INSTRUCTIONS DE CALCUL (opérations arithmétiques et logiques).

1. OPÉRATIONS ENTRE DEUX DONNÉES :

Il y en a 34 en tout. Les opérations sont effectuées entre un registre général et :

— soit un autre autre registre général (exemple : A = A + B);

— soit une case mémoire dont l'adresse est déterminée par l'un des trois modes d'adressage définis précédemment : indexé (type A = A + (B)), immédiat (type : A = A + XX), direct (type : A = A + (XX)).

Dans la plupart des cas, le résultat de l'opération est transféré dans le premier registre.

Quatre types d'opérations peuvent être effectuées sur ces données : l'addition, la soustraction et les opérations logiques ET et OU inclusif. Les instructions de comparaison (COMP...) sont équivalentes à des soustractions mais le résultat n'est pas transféré dans le premier registre.

2. LES OPÉRATIONS DE CALCUL UNAIRE :

Elles ne concernent qu'un seul registre.

— Incrémentation : le registre A (ou B) est augmenté de une unité : **INC A** (ou INC B), instruction équivalente à l'opération A = A + 1 (ou B = B + 1).

— Décrémentation : le registre A (ou B) est diminué de une unité : **DEC A** (ou DEC B), instruction équivalente à l'opération A = A — 1 (ou B = B — 1).

— Rotation du registre A à gauche **(ROTAG)** ou à droite **(ROTAD)** : ces instructions décalent les bits du registre A soit vers la gauche, soit vers la droite; le premier bit étant transféré dans le dernier (ROTAG) ou inversement (ROTAD).

Exemple : Si A = Ø11ØØ1,

— après rotation à gauche, A devient : 11ØØ1Ø;

— après rotation à droite, A devient : 1Ø11ØØ.

Ainsi peut-on effectuer des multiplications par 2 des nombres de Ø à 31 (ROTAG) et des divisions par 2 (ROTAD).

— Négation logique du registre A : **NON A.**
Cette opération inverse la valeur de tous les bits de registre A. Elle est équivalente à l'opération A = A — 63.

Exemple : Si A = Ø11ØØ1 alors NON A = 1ØØ11Ø.

3. COMPLÉMENTS :

Les microprocesseurs permettent généralement d'effectuer d'autres opérations arithmétiques et logiques telles que: multiplications; opérations OU exclusif, NI; décalage et tests de bits; etc.

Chaque exécution de calcul modifie le contenu d'un registre (appelé registre d'état), dans lequel sont consignées des informations portant sur le signe du résultat, l'existence d'un dépassement de capacité, l'apparition d'une interruption, etc.

Ce registre d'état peut lui-même être transféré ou testé par des instructions de saut conditionnel.

LES INSTRUCTIONS DE PROGRAMMATION.

Outre l'instruction de fin de programme (HALTE), les instructions de programmation permettent d'effectuer des branchements (ou sauts) conditionnels ou inconditionnels. Ces instructions modifient le contenu du registre C (compteur de programme) qui indique normalement l'adresse de l'instruction suivante à exécuter.

Dans le saut inconditionnel (VA EN XX), le deuxième mot (XX) qui suit immédiatement le code instruction est transféré dans le registre C. Le microprocesseur ira donc lire la prochaine instruction à l'adresse XX.

Dans les sauts conditionnels (SIP VA EN XX, SIZ VA EN XX, SIN VA EN XX), ce transfert n'est effectué que si la condition spécifiée est remplie. Cette condition porte sur le résultat de la dernière instruction de calcul effectuée :

SIP = Si le résultat est positif ou nul.
SIZ = Si le résultat est zéro.
SIN = Si le résultat est négatif.

Si la condition spécifiée n'est pas remplie, le registre C est simplement et automatiquement augmenté de 2, c'est-à-dire qu'il contient l'adresse de l'instruction immédiatement suivante.

Compléments :

L'adresse à laquelle doit s'effectuer le saut peut être défini selon d'autres modes d'adressage que le mode direct ici utilisé. Il existe ainsi des branchements relatifs, indexés, indirects; chacun d'entre eux pouvant être conditionnels ou inconditionnels.

Il faut mentionner ici deux types d'instructions de programmation, qui joient un rôle important dans la pratique : ce sont les instructions de branchement à sous-programmes et les instructions de retour.

Lors d'un branchement à un sous-programme, le microprocesseur sauvegarde (avant d'effectuer un saut) le contenu du registre du programme principal (généralement dans les adresses les plus élevées).

Lorsque le sous-programme est terminé, le microprocesseur rencontre l'instruction de retour. Celle-ci permet de rétablir le contenu initial du registre du programme principal. Le microprocesseur continue alors d'exécuter les instructions en séquence. C'est exactement ce qui se passe lorsque l'on interrompt momentanément la lecture d'un livre (action principale) pour téléphoner à un ami.

QUELQUES EXEMPLES DE PROGRAMMES

1. UN PROGRAMME QUI SE REPRODUIT LUI-MÊME :

Adresse	Instruction
ØØ	A = (B)
Ø1	B = B + 8
Ø3	(B) = A
Ø4	B = B — 7
Ø6	VA EN Ø

Les 8 mots qui composent ce programme vont être recopiés intégralement dans toute la mémoire. Comme la mémoire comporte 64 cases le programme sera donc recopié 8 fois. Ensuite la mémoire étant circulaire (c'est-à-dire que l'adresse 64 correspond à l'adresse Ø) le programme est recopié sur lui-même. Il ne peut donc pas s'arrêter (taper **F** pour arrêter le programme).

2. UN PROGRAMME QUI AFFICHE A L'ÉCRAN UN MOT ENTRÉ AU CLAVIER LETTRE A LETTRE :

Pour indiquer que le mot est terminé, entrer au clavier le caractère espace.

Adresse	Instruction
ØØ	B = 23
Ø2	ENTRE A
Ø3	(B) = A
Ø4	INC B
Ø5	COMP A — Ø
Ø7	SIZ VA EN 11
Ø9	VA EN 2
11	B = 23
13	A = (B)
14	COMP A — Ø
16	SIZ VA EN 22
18	SORT A
19	INC B
2Ø	VA EN 13
22	HALTE

Remarques :

- Adresse 2 : lorsque l'on entre un caractère au clavier, c'est le code alphanumérique de celui-ci qui est transféré dans le registre A (voir tableau en fin de manuel).
- Chaque caractère entré est stocké dans la mémoire à partir de l'adresse 23 (première adresse libre rencontrée après la mise en mémoire du programme).
- Dès que le caractère espace est rencontré (votre mot est donc terminé), les mémoires préremplies sont "vidées", de façon à faire apparaître à l'écran, dans l'ordre de frappe au clavier, les caractères correspondants au code qu'elles contiennent.

3. UN PROGRAMME QUI CALCULE LA SOMME DE DEUX NOMBRES D'UN CHIFFRE ET QUI AFFICHE LE RÉSULTAT SUR L'ÉCRAN :

Adresse	Instruction
ØØ	ENTRE A
Ø1	ENTRE B
Ø2	B = B — "Ø"
Ø4	A = A + B
Ø5	B = "Ø"
Ø7	COMP A — 26
Ø9	SIN VA EN 14
11	INC B
12	A = A — 1Ø
14	SORT B
15	SORT A
16	HALTE

Remarques :

- Adresse 4 : A correspond à la somme du code alphanumérique du premier caractère rentré et la valeur décimale du deuxième nombre rentré.
- Adresse 7 : on compare A au code alphanumérique 26 :

— Si A < 26 alors le résultat est sur un chiffre et est contenu dans le registre A (B = "O");

— Si A > 26 alors le résultat est sur deux chiffres :

- on incrémente le contenu de B, correspondant au chiffre des dizaines : B = 1;

- on diminue A de 1Ø (c'est-à-dire de la dizaine affectée à B) correspondant au chiffre des unités.

4. UN PROGRAMME QUI CALCULE LE PRODUIT DE DEUX NOMBRES D'UN CHIFFRE :

Ce programme utilise simplement l'addition répétée : a x b = (a + a + ... + a) b fois.

Adresse	Instruction
ØØ	ENTRE A
Ø1	ENTRE B
Ø2	A = A — "Ø"
Ø4	B = B — "Ø"
Ø6	MET A EN 43
Ø8	A = "Ø"
1Ø	MET A EN 44
12	MET A EN 45
14	COMP B — Ø
16	SIZ VA EN 38
18	DEC B
19	A = A + (43)
21	COMP A — 26
23	SIN VA EN 14
25	A = A — 1Ø
27	MET A EN 44
29	A = (45)
31	INC A
32	MET A EN 45
34	A = (44)
36	VA EN 14
38	B = (45)
4Ø	SORT B
41	SORT A
42	HALTE

LEXIQUE

ADRESSE Comme les habitants d'une ville, chaque case mémoire possède une adresse qu'il est indispensable de connaître pour pouvoir lire ou modifier son contenu. Il existe différentes manières de définir une adresse (voir modes d'adressage). L'adresse de la case mémoire sélectionnée est stockée dans le registre d'adresse.

Selon les capacités de la mémoire de l'ordinateur, l'adresse peut être un nombre plus ou moins grand, codé sur 6, 8, 16 voire 32 bits (donnant lieu à 2^6, 2^8, 2^{16} ou 2^{32} cases mémoire).

ARCHITECTURE C'est l'ensemble des éléments qui composent l'unité centrale de l'ordinateur (notamment les différents registres, la ou les unités de calcul, les mémoires, les interfaces...).

BINAIRE Le code binaire est la représentation des nombres en base 2. On utilise seulement deux symboles, notés généralement Ø et 1.

— Le nombre le plus grand que l'on puisse écrire en binaire dépend du nombre de bits utilisés. Le bit dit de "poids faible" (le plus à droite) donne le nombre d'unitée, le deuxière bit donne le nombre de paires d'unitée, le troisième bit le nombre de paquets de 4 unités, etc.

— Ainsi avec 6 bits, on peut écrire tous les nombres de Ø (ØØØØØØ) à 63 (111111); avec 1Ø bits tous les nombres de Ø à 1Ø23, etc.

BITS Le bit est l'unité élémentaire d'information. Un bit peut contenir un seul des deux symboles utilisés dans le code binaire : Ø ou 1.

CODE ALPHANUMÉRIQUE Un ordinateur ne peut manipuler que des nombres codés en binaire. Il ne sait pas ce qu'est une lettre ou un signe de ponctuation. Pour converser avec lui, on convient d'un code interne, c'est-à-dire d'une façon de numéroter les différentes lettres ou symboles utilisables, qui soit compréhensible par l'ordinateur. Le code habituellement utilisé est le code ASCII (voir tableau en annexe).

— Ainsi, lorsque nous appuyons sur une touche de clavier, l'ordinateur reçoit en fait le code qui a été attribué à ce caractère.

— Inversement, ce même code envoyé vers un périphérique de sortie (écran ou imprimante) produira l'affichage ou l'impression du caractère correspondant.

COMPTEUR ORDINAL (OU COMPTEUR DE PROGRAMME)

Ce registre particulier contient l'adresse de la prochaine instruction à exécuter. Normalement, il est automatiquement augmenté du nombre de mots formés par l'instruction en cours d'exécution, de telle sorte que la prochaine instruction soit celle immédiatement suivante, en séquence dans le programme.

Mais les instructions de branchement peuvent modifier ce registre, en remplaçant son contenu par une adresse différente. Dans les branchements à sous-programme, le contenu de ce registre est sauvegardé en mémoire avant modification (généralement dans les adresses les plus élevées), de telle sorte qu'après exécution du sous-programme, cette valeur soit réintroduite dans le compteur de programme et que l'exécution continue en séquence.

DÉCIMAL

Le code décimal est la manière usuelle de représenter les nombres, en base 10. On utilise ici les chiffres de Ø à 9. Le chiffre le plus à droite désigne le nombre d'unités, le deuxième chiffre le nombre des dizaines, etc.

HEXADÉCIMAL

C'est le mode de représentation des nombres en base 16. Comme le code héxadécimal exige l'utilisation de 16 symboles, on a ajouté aux 10 chiffres (Ø à 9), les lettres suivantes : A (pour 10), B (pour 11), C (pour 12), D (pour 13), E (pour 14), F (pour 15).

- A partir d'une représentation binaire, la valeur d'un nombre en héxadécimal s'obtient facilement en regroupant les bits par groupes de 4, à partir de la droite. A l'intérieur de chaque groupe, on calcule la valeur (comprise entre Ø et 15) et on la remplace par le chiffre ou la lettre correspondant en héxadécimal.

- Inversement, pour passer du code héxadécimal au code binaire, on remplace chiffre ou lettre par le groupe de 4 bits correspondant à sa valeur binaire.

Exemples : Le nombre 21 en base 1Ø, s'écrit en binaire 1Ø1Ø1 ce qui donne en héxadécimal 1 5.

Le nombre 29 en base 1Ø, s'écrit en binaire Ø111Ø1 ce qui donne en héxadécimal 1 D.

HORLOGE

L'unité centrale possède une horloge qui permet de synchroniser les différentes opérations effectuées par le microprocesseur : lecture de l'instruction, décodage, avance du compteur de programme, exécution, etc.

INTERFACES

L'unité centrale communique avec l'extérieur (c'est-à-dire avec ses périphériques) par l'intermédiaire d'interfaces. Ces interfaces sont numérotées, certaines fonctionnent en entrée seulement (ex. : clavier), d'autres en sortie uniquement (ex. : écran, imprimante), d'autres encore peuvent fonctionner alternativement en entrée et en sortie (ex. : lecteurs de cassettes et de disquettes).

INTERRUPTION

L'unité centrale et les périphériques ne travaillent généralement pas à la même vitesse. L'utilisateur doit pouvoir, lui aussi, travailler à son propre rythme.

Dans ces conditions, le microprocesseur doit "patienter" (par exemple, en effectuant une boucle à vide), jusqu'à ce que le périphérique ait terminé sa tâche ou que l'utilisateur ait tapé sa commande. Le microprocesseur reçoit alors un message d'interruption qui lui signifie que l'attente est terminée et qu'il peut exécuter la suite du programme.

LANGAGES

En informatique, le mot "langage" désigne un ensemble de caractères et de symboles (ou vocabulaire) et de règles de syntaxe (ou grammaire) qui permet à l'utilisateur de "converser" avec l'ordinateur, en vue de lui faire exécuter un ensemble d'instructions (ou programme). Il existe aujourd'hui un tel nombre de langages informatiques qu'il est impossible d'en donner ici une liste exhaustive.

On peut les répartir en trois grandes catégories : les langages machine et assembleur, les langages compilés et les langages interprétés.

LES LANGAGES MACHINE ET ASSEMBLEUR

— Le langage machine est, en fait, le seul que peut "comprendre" le microprocesseur. Ainsi, quelque soit le langage utilisé, un programme devra toujours être traduit, à un moment ou à un autre, en langage machine avant d'être exécuté.

— Le vocabulaire du langage machine est réduit à la seule liste des codes d'instruction du microprocesseur et sa seule règle de syntaxe consiste à imposer une longueur donnée à chaque type d'instruction (1 mot, 2 mots ou plus).

Le langage assembleur est très proche du langage machine : les codes d'instruction sont remplacés par une expression mnémonique qui rappelle en clair l'opération effectuée par le microprocesseur (exemple : ENTRE A remplace le code Ø Ø Ø 1Ø Ø). Les adresses et les données peuvent également être remplacées par des noms ou écrites dans différents modes de représentation (binaire, octal, etc.).

— Un programme écrit en langage machine ou assembleur sera très rapide à exécuter mais, en général, très long à écrire (on doit en effet le décomposer en un grand nombre d'opérations élémentaires).

— De plus, ces langages sont particuliers à chaque microprocesseur puisqu'ils dépendent du jeu d'instructions de celui-ci. Un programme machine ou assembleur écrit pour un ordinateur ne pourra donc être utilisé que sur des ordinateurs de même type.

LES LANGAGES COMPILÉS

— Ils possèdent, au contraire, un vocabulaire et une syntaxe indépendants du processeur et de son jeu d'instructions. Ils ont été définis à partir des opérations et des fonctions les plus couramment utilisées et possèdent une syntaxe plus riche, permettant de commander des opérations complexes dans une seule ligne de programme.

— A la suite du premier langage compilé, le FORTRAN, destine aux calculs numériques et scientifiques, sont apparus de nombreux autres langages plus spécifiques (COBOL pour la gestion, ALGOL pour les calculs mathématiques), mieux structurés (PASCAL) ou réalisant une synthèse des avantages de plusieurs langages (PL1, ADA).

— Avant d'être exécuté, un programme écrit dans un de ces langages, doit d'abord être compilé, c'est-à-dire traduit dans le langage machine spécifique de l'ordinateur. Cette traduction est réalisée par un programme particulier, propre à chaque machine et à chaque langage : le compilateur. Lors de cette phase, certaines erreurs de programmation peuvent être détectées par le compilateur.

— Le résultat de la compilation est une suite d'instructions écrites en langage machine réalisant toutes les opérations indiquées dans le programme d'origine (ou programme source).

— En général, cette suite d'instructions doit être "assemblée", avant d'être exécutable, avec d'autre modules ou sous-programmes qui ont été écrits et compilés séparément. Cet assemblage est réalisé par l'éditeur de liens.

— Un programme écrit en langage compilé sera très rapide à exécuter et très facile à écrire. Cependant, sa mise au point est relativement longue : chaque erreur détectée nécessite sa correction, une nouvelle compilation, l'édition des liens et la vérification pendant l'exécution.

LES LANGAGES INTERPRÊTÉS

— Ils possèdent, comme les langages compilés, un vocabulaire et une syntaxe très riches et indépendants du jeu d'instructions de chaque machine. Cette fois-ci, c'est immédiatement avant son exécution que chaque ligne du programme est traduite par l'interpréteur en une suite d'instructions machine.

— On élimine ainsi les phases de compilation et d'édition de liens, et la mise au point des programmes est beaucoup plus rapide.

— Par contre, l'exécution sera évidemment ralentie par le temps de traduction.

— Certains langages existent sous forme interprétée et compilée. Le BASIC est le plus connu des langages interprétés et sa facilité d'emploi est légendaire.

— D'autres interpréteurs ont été introduits dans le cadre des recherches en Intelligence Artificielle : LISP, PROLOG.

MÉMOIRES

Avec le processeur, la mémoire est le principal élément d'un ordinateur. Il existe plusieurs types de mémoires caractérisés par leur mode d'accès, leur technologie, leur capacité et leur vitesse d'accès.

— La mémoire centrale d'un ordinateur présente généralement une capacité limitée (quelques milliers à quelques millions d'octets) et une grande vitesse d'accès (de l'ordre du millionième de seconde). Chaque élément de la mémoire possède une adresse. Le processeur envoie cette adresse à la mémoire centrale, pour lire ou modifier son contenu. On distingue :

— Les mémoires "mortes" (ROM : Read Only Memory) qui ne peuvent être que lues par le microprocesseur : on ne peut donc modifier leur contenu. Elles contiennent des informations ou des programmes qui doivent être protégés des erreurs de manipulation. Les cartouches des ordinateurs T07, T070 et M05 contiennent des mémoires mortes. Ainsi, l'interpréteur BASIC, inscrit dans une ROM, ne pourra être modifié ou effacé par une erreur de programmation.

— Les mémoires "vives" (RAM : Random Access Memory) qui, au contraire, peuvent aussi bien être lues qu'écrites. Leur contenu pourra donc être modifié à loisir par l'utilisateur.

— En complément, les ordinateurs disposent également de mémoires de masse : ce sont des mémoires de très grande capacité (quelques centaines de kilooctets à quelques centaines de mégaoctets) mais d'accès relativement lent. Elles utilisent un support magnétique (disquette, disque, bande magnétique) le plus souvent interchangeable, ce qui permet de constituer une véritable bibliothèque de programmes et de données.

— Les disques et les disquettes ont un coût plus élevé et une capacité plus réduite que les bandes magnétiques. Par contre, ils présentent l'avantage d'avoir un temps d'accès plus rapide et surtout aléatoire : toute information stockée peut être immédiatement retrouvée en quelques millionièmes de seconde.

— Au contraire, les bandes magnétiques sont à accès séquentiel : il faut lire toutes les informations préalablement stockées avant d'accéder à celle que l'on recherche. D'où un temps d'accès relativement long (quelques secondes à quelques minutes).

MICRO-PROCESSEUR

Le microprocesseur est la partie "intelligente" du micro-ordinateur. C'est lui qui gère la mémoire centrale, échange des informations avec les périphériques, effectue les calculs, etc.

● Le mot "microprocesseur" est réservé aux processeurs réalisés sur un seul circuit intégré (la puce). Chaque microprocesseur possède une architecture et un jeu d'instructions qui lui sont propres.

MODES D'ADRESSAGE

Toute opération de mémorisation, d'entrée-sortie, de programmation implique la sélection d'une adresse. C'est pourquoi le calcul des adresses est une opération fondamentale en programmation. Pour la simplifier et la rendre plus rapide, la plupart des microprocesseurs proposent différents modes d'adressage, c'est-à-dire différentes façons de calculer une adresse :

— L'adressage inhérent n'intéresse que les registres (voir ce mot), la sélection du ou des registres étant entièrement définie par le code d'instruction.

— L'adressage immédiat sélectionne la case mémoire qui suit exactement celle du code d'instruction. L'adresse est donc entièrement définie par le contenu du registre compteur de programme.

— L'adressage direct utilise une définition explicite de l'adresse immédiatement après le code d'instruction. Cette définition peut être réduite à une partie de l'adresse complète (le reste étant défini par le registre de pages) ou étendue (complète).

— L'adressage indexé utilise le contenu d'un des registres du microprocesseur. On peut le considérer comme un adressage inhérent indirect.

— Dans l'adressage indexé relatif, l'adresse réelle est obtenue par addition ou soustraction à l'un des registres, d'une quantité contenue soit dans un autre registre, soit dans la case mémoire suivant immédiatement le registre d'instruction.

— L'adressage indirect nécessite le calcul préalable d'une première adresse, définie par l'un des modes précédents. Cette première adresse sélectionne une case mémoire qui contient l'adresse réelle.

MOT

Le mot est la quantité d'informations échangée en une seule fois entre le microprocesseur et la mémoire centrale. C'est aussi la quantité d'informations contenue dans les registres généraux. Les microprocesseurs les plus courants utilisent des mots de 8, 16, voire 32 bits soit 1,2 ou 4 octets.

OCTET

L'octet est la quantité d'informations définie par 8 bits. Ne pas confondre octet et octal (mode de représentation des nombres en base 8).

— Un octet peut contenir un nombre compris entre Ø et 255 (code décimal), Ø et 377 (code octal), Ø et FF (code hexadécimal), un caractère quelconque (code ASCII).

— Un kilooctet (abréviation ko) contient 1Ø Ø Ø octets soit 8Ø Ø Ø bits. Un mégaoctet (Mo) contient un million d'octets.

PÉRIPHÉRIQUE

Un périphérique est un appareil utilisé par l'ordinateur pour échanger des informations avec l'extérieur. Il permet :

— soit d'entrer des informations dans l'ordinateur (clavier, manette, crayon optique),

— soit de sortir des informations de l'ordinateur et de les présenter sous forme convenable à l'utilisateur (écran, imprimante, générateur de sons),

— soit de lire ou stocker des informations sur un support magnétique (disque, disquette).

REGISTRES

Les registres sont des mémoires particulières intégrées au microprocesseur, dont le rôle est, en général, de stocker des résultats intermédiaires. Leur temps d'accès est extrêmement court mais leur nombre est réduit.

Certains registres ont des fonctions plus spécialisées :

— le registre compteur de programme contient l'adresse de l'instruction suivante à exécuter,

— le registre d'état contient des informations relatives à la dernière opération effectuée : signe, dépassement de capacité, interruption, etc,

— les registres d'index sont essentiellement utilisés dans le calcul des adresses indexées relatives.

UNITÉ ARITHMÉTIQUE ET LOGIQUE

C'est la partie du microprocesseur spécialisée dans le calcul. Le microprocesseur y transfert les données contenues soit dans un registre soit dans une case mémoire et lui indique le type d'opération à effectuer : addition, soustraction, multiplication, opérations logiques ET-OU-NON, etc.

UNITÉ CENTRALE

L'unité centrale est composée du ou des microprocesseurs, des mémoires centrales, des circuits d'interfaces, de l'horloge et des circuits de décompte du temps. Un ordinateur complet comprendra, outre son unité centrale, des périphériques, des mémoires de masse, des circuits d'alimentation.

TABLEAU DES INSTRUCTIONS

INSTRUCTIONS COURTES (1 mot de 6 bits)

N°	Octal	Binaire	Instruction	Type	Mode d'adressage
0	00	000000	HALTE	programmation	inhérent
1	01	000001	NON A	calcul	« «
2	02	000010	(A) = B	transfert	indexé
3	03	000011	(B) = A	« «	« «
4	04	000100	ENTRE A	entrée/sortie	inhérent
5	05	000101	ENTRE B	« «	« «
6	06	000110	SORT A	« «	« «
7	07	000111	SORT B	« «	« «
8	10	001000	A = B	transfert	« «
9	11	001001	B = A	« «	« «
10	12	001010	A = (B)	« «	indexé
11	13	001011	B = (A)	« «	« «
12	14	001100	ROTAG	calcul	inhérent
13	15	001101	ROTAD	« «	« «
14	16	001110	COMP A-B	« «	« «
15	17	001111	COMP B-A	« «	« «
16	20	010000	A = A + B	« «	« «
17	21	010001	B = B + A	« «	« «
18	22	010010	A = A + (B)	« «	indexé
19	23	010011	B = B + (A)	« «	« «
20	24	010100	A = A-B	« «	inhérent
21	25	010101	B = B-A	« «	« «
22	26	010110	A = A-(B)	« «	indexé
23	27	010111	B = B-(A)	« «	« «
24	30	011000	A = A ET B	« «	inhérent
25	31	011001	B = B ET A	« «	« «
26	32	011010	A = A OU B	« «	« «
27	33	011011	B = B OU A	« «	« «
28	34	011100	INC A	« «	« «
29	35	011101	INC B	« «	« «
30	36	011110	DEC A	« «	« «
31	37	011111	DEC B	« «	« «

INSTRUCTIONS LONGUES

(2 mots de 6 bits) : le deuxième mot contient la valeur XX.

N°	Octal	Binaire	Instruction	Type	Mode d'adressage
32	40	100000	VA EN XX	programmation	direct
33	41	100001	SIN VA EN XX	« «	« «
34	42	100010	SIZ VA EN XX	« «	« «
35	43	100011	SIP VA EN XX	« «	« «
36	44	100100	(A) = XX	transfert	indexé
37	45	100101	(B) = XX	« «	« «
38	46	100110	MET A EN XX	« «	direct
39	47	100111	MET B EN XX	« «	« «
40	50	101000	A = XX	« «	immédiat
41	51	101001	B = XX	« «	« «
42	52	101010	A = (XX)	« «	direct
43	53	101011	B = (XX)	« «	« «
44	54	101100	COMP A - XX	calcul	immédiat
45	55	101101	COMP B - XX	« «	« «
46	56	101110	COMP A-(XX)	« «	direct
47	57	101111	COMP B-(XX)	« «	« «
48	60	110000	A = A + XX	« «	immédiat
49	61	110001	B = B + XX	« «	« «
50	62	110010	A = A + (XX)	« «	direct
51	63	110011	B = B + (XX)	« «	« «
52	64	110100	A = A - XX	« «	immédiat
53	65	110101	B = B - XX	« «	« «
54	66	110110	A = A - (XX)	« «	direct
55	67	110111	B = B - (XX)	« «	« «
56	70	111000	A = A ET XX	« «	immédiat
57	71	111001	B = B ET XX	« «	« «
58	72	111010	A = A OU XX	« «	« «
59	73	111011	B = B OU XX	« «	« «
60	74	111100	A = A ET (XX)	« «	direct
61	75	111101	B = B ET (XX)	« «	« «
62	76	111110	A = A OU (XX)	« «	« «
63	77	111111	B = B OU (XX)	« «	« «

CODE ALPHANUMÉRIQUE

Caractères	Code 6 bits
Espace	Ø
!	1
"	2
#	3
$	4
%	5
&	6
'	7
(	8
)	9
*	1Ø
+	11
,	12
—	13
.	14
/	15
0	16
1	17
2	18
3	19
4	2Ø
5	21
6	22
7	23
8	24
9	25
:	26
;	27
<	28
=	29
>	3Ø
?	31
φ	32

Caractères	Code 6 bits
A	33
B	34
C	35
D	36
E	37
F	38
G	39
H	4Ø
I	41
J	42
K	43
L	44
M	45
N	46
O	47
P	48
Q	49
R	5Ø
S	51
T	52
U	53
V	54
W	55
X	56
Y	57
Z	58
[	59
\	6Ø
]	61
↑	62
—	63

Pour pouvoir coder l'ensemble des caractères usuels sur 6 bits, nous avons adopté le code 6 bits ci-dessus, égal au code ASCII diminué de 32 unités. Les caractères de contrôle et les caractères minuscules ne peuvent pas être représentés dans ce code. La liste complète du code ASCII est indiqué dans votre manuel BASIC.

ANNEXE

"MICROPROCESSEUR" est enregistré une fois sur chaque face de la cassette. L'enregistrement de la face A est compatible avec le système TO 7/70, celui de la face B avec le système MO 5.

Lorsque le joueur sera familiarisé avec "MICROPROCESSEUR", il pourra relever le numéro du compteur du lecteur de programme, lors du chargement de chaque module, et compléter l'un des tableaux ci-dessous.

	N° du compteur
EN-TÊTE	.....
INITIATION	.....
EXERCICE DE SIMULATION	.....

Pour retrouver directement le jeu, positionner la bande sur le numéro relevé sur le lecteur. Charger le programme en fonction de votre micro-ordinateur, comme cela est indiqué au chapitre "Mise en marche du système".

ATTENTION !
pour le TO 7/70 : FACE ROUGE
pour le MO 5 : FACE VERTE

CONSEILS D'UTILISATION DES LOGICIELS

IL EST IMPÉRATIVEMENT RECOMMANDÉ DANS L'UTILISATION :

● du magnétophone ;
— d'éviter de passer de l'avance rapide au retour rapide et vice versa sans passer par le stop ;
— de nettoyer de temps en temps avec un coton imbibé d'alcool (à 90 °C), ou avec une cassette autonettoyante, la tête magnétique, les galets d'entraînement et les guides-bandes de votre magnétophone ;

● de la cassette :
— de ne pas toucher la bande avec les doigts ;
— de protéger de la poussière en rangeant la cassette dans sa boîte ;
— d'éviter les hautes températures, l'humidité et le voisinage avec les champs magnétiques.

CONDITIONS DE GARANTIE

1. Conditions générales

De cette cassette, cartouche ou disquette protégée par copyright, toute reproduction directe ou indirecte par quelque moyen électronique, électrique, magnétique, optique, laser, acoustique ou toutes autres technologies similaires existantes ou à venir est strictement interdite sous peine de poursuites.

2. Conditions de garantie.

Cette garantie couvre les défauts de fabrication des composants physiques de la cassette, de la cartouche ou de la disquette, et les erreurs éventuelles de duplication des programmes.

ÉCHANGE STANDARD DU LOGICIEL CHEZ VOTRE REVENDEUR :

— gratuitement pendant un an à compter de la date d'acquisition pour les cassettes, les disquettes et les cartouches.