

# raconte-moi...

## LA MICRO-INFORMATIQUE

**L'** Assembleur est sans conteste le langage le plus adéquat pour tous ceux qui veulent tirer le meilleur parti de leur système. Par rapport au Basic, tout programme écrit en Assembleur est nettement plus performant du point de vue vitesse d'exécution, tout en occupant une zone mémoire moindre. Des gains de plusieurs dizaines sont des ordres de grandeurs tout à fait courants. Ce sont là des avantages très appréciables, sinon indispensables, dans des applications comme la commande d'organes d'un automate, ou le traitement de données graphiques (jeux vidéo, entre autres). Le second point, tout aussi important, réside dans son aspect didactique. Le langage Assembleur est « proche » du microprocesseur et, de ce fait, spécifique du composant choisi. Dès lors qu'il s'agit du Z-80, le microprocesseur 8 bits le plus répandu sur le marché de la micro-informatique, ce n'est plus une limitation. Comme la rose a ses épines, le lan-

Nous avons décrit dans cette rubrique le Microprofessor 1 (voir Led n° 3, décembre 1982). Devant le succès de ce produit, et dans un même esprit pédagogique, la société Multitech a développé le Microprofessor-1 PLUS. Le MPF-1 PLUS se programme en hexadécimal (code machine), mais surtout en Assembleur (mnémonique Z-80), langage le plus efficace à l'échelle du microprocesseur. L'option Basic (ROM 8 Koctets, type 2764) donne une dimension supplémentaire à ce nouveau micro-ordinateur.

gage Assembleur a ses contraintes. L'apprentissage de l'Assembleur est plus long que celui du Basic. Cependant le MPF-1 PLUS est astucieuse-

ment conçu pour en atténuer les difficultés, grâce notamment à deux programmes résidents particulièrement performants, le Moniteur et l'Editeur. Les résultats obtenus récompenseront largement les utilisateurs qui auront consenti à cet effort.

L'investissement se révélera d'autant plus payant qu'existe la ROM Basic, offrant ainsi la possibilité à chacun de programmer son application dans le langage de son choix. La dextérité (stimulée par la curiosité) venant avec la pratique, l'utilisation conjointe des deux langages conduira à des réalisations très performantes. Le Basic est réservé pour bâtir l'ossature du programme tandis que les sous-routines sont exécutées en Assembleur. Ainsi, simplicité, efficacité et rapidité confèrent à ce système peu onéreux des performances dignes de rivaliser avec les micro-ordinateurs haut de gamme.

Les caractéristiques du MPF-1 PLUS sont résumées dans le tableau 1 de la figure 1. La figure 2 présente le MPF-1 PLUS.

### CARACTERISTIQUES DU MPF-1 PLUS

CPU	Microprocesseur <b>Z80</b> haute performance, comportant un répertoire de 158 instructions de base. Compatibilité avec les programmes écrits en 8080 ou 8085 (code objet).
RAM	4 K.Octets. Possibilité de sauvegarde du contenu avec RAM CMOS (alimentation par piles).
ROM	8 K.Octets (type 2764) pour le <b>Moniteur</b> comportant 27 commandes. Programmes résidents: <b>Editeur, Assembleur « 2 Passes », Assembleur par ligne, Initialisation</b> , etc. Sous-routines adressables : 44.
CLAVIER	Alphanumérique (QWERTY), 49 touches mécaniques avec « bip » de contrôle.
VISUALISATION	Affichage sur 20 caractères, digits 16 segments.
INTERFACE K7	Vitesse 165 bits/seconde avec CHECK SUM. Recherche automatique d'un fichier par son indicatif.
OPTIONS	ROM « BASIC » 8 K.Octets (type 2764). Désassembleur en mnémoniques Z80 avec l'imprimante.

Fig. 1. tableau 1

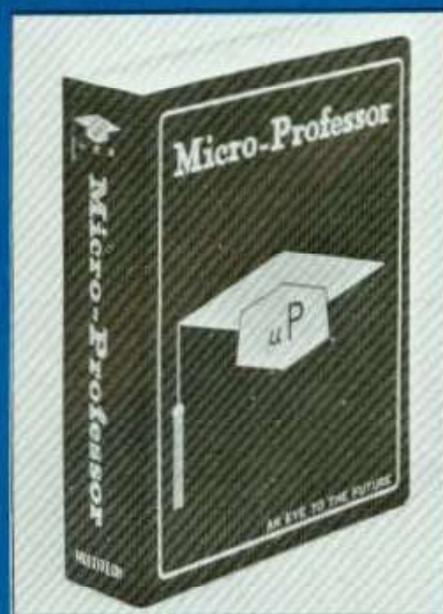


Fig. 2 : Présentation du MPF-1 PLUS.

# raconte-moi...

## LA MICRO-INFORMATIQUE ET L'AMATEUR

### LE MONITEUR

Le Moniteur est l'un des programmes résidents dans la ROM (8 Koctets). Il gère les diverses servitudes : initialisation à la mise sous tension, scrutation du clavier, affichage des caractères alphanumériques, échange des données avec le magnétocassette, etc.

Les différentes commandes du Moniteur sont présentées tableau II, fig. 3. L'accès à l'un des modes de fonctionnement, Editeur, Assembleur, Désassembleur ou interpréteur Basic, s'effectue en maintenant la touche CONTROL enfoncée et en sélectionnant le mode souhaité. A noter que CONTROL R et CONTROL C autorisent un retour respectivement, soit à l'Editeur, soit à l'Interpréteur sans initialisation préalable : les données contenues dans la mémoire sont intégralement conservées ainsi que les adresses des différents pointeurs.

En mode Moniteur, le chargement de données en code hexadécimal s'effectue au moyen du clavier ou d'une cassette. Le contenu de n'importe quel emplacement mémoire peut être visualisé et une ou plusieurs données supprimées ou insérées dans un programme.

Comme dans le MPF-1, le contenu de tous les registres (au nombre de 22 dans le Z-80) est accessible pour être lu et/ou modifié, y compris pendant le déroulement d'un programme.

Ceux-ci peuvent être exécutés « intégralement », en « pas à pas », c'est-à-dire instruction après instruction et avec un « point d'arrêt ».

Lorsque les RAM's utilisées sont du type CMOS, un jeu de quatre piles (placées au dessous de la carte) assurent automatiquement la sauvegarde des mémoires vives. Votre travail peut être ainsi interrompu inopi-

nément et repris ultérieurement : c'est un avantage fort appréciable, que le programme soit court ou long. Un petit interrupteur placé sur le dessus de la carte permet de mettre en œuvre ou non les piles, et ainsi de ne

les utiliser que lorsque vous le souhaitez.

Les quarante quatre routines du Moniteur sont adressables. Elles facilitent beaucoup le travail de programmation.

### MONITEUR « MPF-1 PLUS » COMMANDES

#### 1 - Fonctions générales

RESET	Introduit et initialise le Moniteur
CONTROL Q	Retour au Moniteur
CONTROL E	Introduit et initialise l'Editeur
CONTROL R	Retour à l'Editeur
CONTROL A	Introduit l'Assembleur « 2 Passes »
CONTROL L	Introduit l'Assembleur « 1 Passe »
CONTROL D	Introduit le Désassembleur
CONTROL B	Interpréteur Basic
CONTROL C	Retour au Basic
CONTROL J	Calcul de l'adresse relative

#### 2 - Fonctions « DATA »

F	Chargement des données
I	Insertion de données dans la mémoire
D	Suppression de données dans la mémoire
M	Visualise le contenu de la mémoire spécifiée
↓	Visualise le contenu des 4 octets suivants
↑	Visualise le contenu des 4 octets précédents
:	Modifie le contenu de la mémoire
/	Transfert d'un bloc de données

#### 3 - Fonctions « Registres »

R	Visualise le contenu des registres
↓	Visualise le contenu de la paire de registres suivante
↑	Visualise le contenu de la paire de registres précédente
:	Modifie le contenu du registre

#### 4 - Fonctions « Exécution »

G	Exécute le programme à l'adresse spécifiée
S	Exécute le programme en « pas à pas »
B	Introduction ou suppression d'un point d'arrêt

#### 5 - Fonctions « cassette »

L	Chargement de la mémoire à partir d'une bande
W	Ecriture de la bande à partir de la mémoire

**Problème :** Additionner les N premiers nombres entiers. Placer le résultat dans le registre B.

Fig. 3, tableau II

## DEVELOPPEMENT D'UN PROGRAMME

Avant d'examiner les étonnantes possibilités du MPF-1 PLUS, en Assembleur, retraçons les phases essentielles du développement d'un programme écrit dans ce langage.

La première étape consiste à bien définir la tâche à accomplir, et c'est vrai, quel que soit le mode de programmation employé. Elle est toujours suivie d'une phase d'analyse, de laquelle découle un ensemble d'ordinogrammes. Les uns, très généraux, constituent le squelette du programme, d'autres plus détaillés, font apparaître l'enchaînement des opérations permettant d'aboutir au résultat final.

La figure 4 présente un ensemble « ordinogramme ». Compte tenu de la simplicité du problème, un organigramme unique suffit amplement.

Une telle représentation est très explicite pour les utilisateurs, mais ne peut en aucun cas être acceptée ainsi par le micro-ordinateur, qui ne peut traiter que les informations binaires.

C'est pour favoriser le dialogue « homme-machine » que des langages, purement artificiels, ont été développés, comme l'Assembleur en ce qui nous concerne, mais il en existe bien d'autres, comme le Basic, que nous avons évoqué, ou le Fortran, le Cobol, le Forth, etc.

En Assembleur, pour pouvoir introduire notre programme, il faut le traduire en une séquence d'opérations. Chacune d'entre elles est une instruction. C'est une opération élémentaire exécutable par le microprocesseur. Pour favoriser le dialogue, tout en conservant un aspect compréhensif au programme, chaque instruction est représentée sous une forme condensée, sous le nom de code mnémotique.

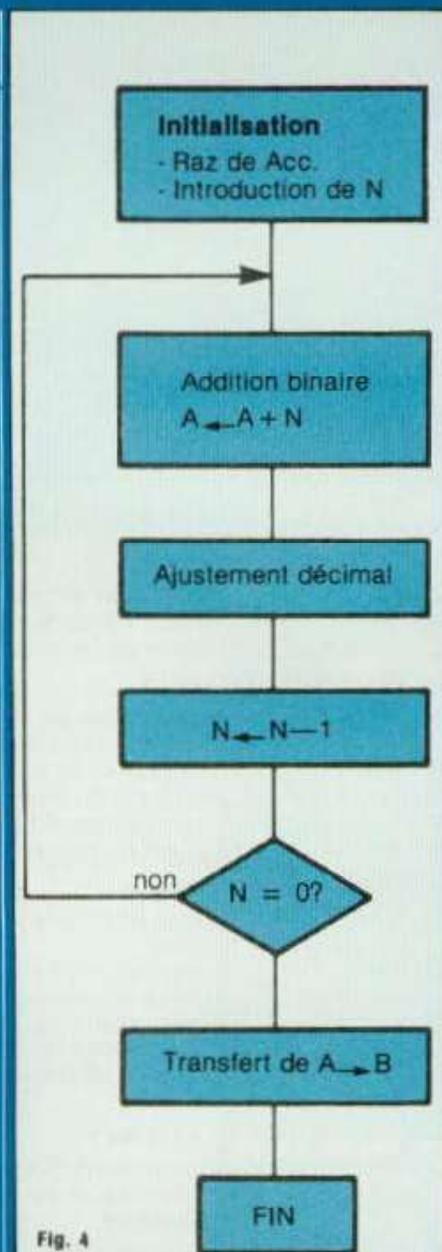


Fig. 4

Par exemple, « charger le registre accumulateur avec la quantité décimale 3 », devient :

Ld A, 3 (avec Ld = Load, charger)

ou bien encore : « additionner au contenu de l'accumulateur celui du registre B, et placer le résultat dans A », devient

ADD A, B (avec ADD = Addition)

C'est à ce niveau que le langage Assembleur établit le trait d'union entre l'utilisateur et le microprocesseur. Au moment de l'assemblage, qui peut être immédiat ou différé comme nous le verrons par la suite, le programme Assembleur traduit les différents codes mnémotiques en leurs données binaires équivalentes, si bien que Ld A,3 devient 0011 1110 0000 0011 et ADD A,B 1000 0000 ou exprimées en hexadécimal, 3E 03 et 80.

La suite ordonnée des instructions écrite en mnémotique constitue le « programme source », tandis que la suite des octets qui en résulte après assemblage est le « code objet ».

Le répertoire du Z-80, un des plus riches parmi les microprocesseurs 8 bits, ne comporte pas moins de 158 instructions de base qui peuvent donner naissance à 696 codes différents. Si, par inadvertance, vous introduisez un mnémotique incomplet ou incorrect du point de vue syntaxe, il ne sera pas accepté. Ainsi, au lieu de rentrer Ld, HL, FB 00H, vous introduisez Ld, LH, FB 00H, le code erreur apparaît sur la visualisation.

## L'ASSEMBLEUR 1 PASSE

Lorsque l'utilisateur choisit de travailler en mode Assembleur une passe (control L), chaque instruction (après « FIN DE LIGNE ») est traduite en binaire (et visualisée en hexadécimal). Le ou les codes hexadécimaux correspondants sont rangés automatiquement les uns derrière les autres. Chaque octet (2 codes hexa) occupant un emplacement de la mémoire RAM. Dans ce mode de fonctionnement l'Assem-

# raconte-moi...

## LA MICRO-INFORMATIQUE ET L'AMATEUR

bleur joue essentiellement le rôle de traducteur mais suffisamment « fûté » pour n'accepter que des instructions valides, ce qui garantit à l'utilisateur la disposition d'un programme machine correct sur le plan grammatical.

C'est certainement le mode de fonctionnement le mieux adapté pour tous ceux qui veulent s'initier au langage Assembleur. Il présente un aspect didactique des plus performants dans la mesure, d'une part où la traduction code mnémotique en code objet est immédiate, et d'autre part toute instruction mal libellée est automatiquement rejetée. L'Assembleur 1 passe présente un second avantage : celui de laisser disponible la totalité de l'espace mémoire pour l'application, ce qui est particulièrement séduisant dans le cas des programmes très longs.

### L'ASSEMBLEUR 2 PASSES

Un logiciel Assembleur digne de ce nom, effectue non seulement la conversion du programme source en programme « objet », mais aussi gère l'espace mémoire, c'est l'adressage symbolique ; de plus il autorise l'emploi des « commentaires » pour améliorer la lisibilité des programmes ainsi que des pseudo-instructions.

La fonction traduction est similaire à ce que nous avons décrit pour l'assemblage ligne par ligne. Rappelons que toute erreur de syntaxe est automatiquement détectée. Alors que dans le premier mode de fonctionnement un mnémotique incorrect était immédiatement signalé (un ? apparaît sur la visu) dans ce second mode, l'erreur n'est indiquée qu'au cours de la première phase de l'assemblage, c'est-à-dire après l'introduction du texte complet. Chaque ligne « inacceptable » est d'une part repérée par son numéro de lignes (numérotation automatique) et

d'autre part la raison du refus codifiée. Par exemple « I » indique un code opératoire illégal, « U » un symbole indéfini, ou bien « E » une donnée hors limite. Le MPF-1 PLUS dispose de sept messages d'erreurs qui facilitent d'autant la correction du texte d'origine. Toutes les erreurs sont visualisées sur l'affichage et peuvent être listées simultanément sur l'imprimante. Il ne reste plus qu'à retourner au mode Edition (CONTROL R), appeler les différentes lignes (G numéro de lignes) détectées erronées, et les corriger.

Il n'est de bon Assembleur sans bon Editeur de texte. Ce dernier logiciel joue deux rôles essentiels : le pre-

mier est de faciliter l'introduction du texte initial qui constitue le programme « source ». C'est au cours de cette phase que chaque ligne de programme est identifiée par un numéro d'ordre, ce qui rend la recherche aisée par la suite.

Le second rôle, et non le moindre, est de permettre d'apporter les modifications nécessaires au texte, soit après la première phase d'assemblage (erreur de syntaxe) soit au cours de la mise au point (erreur de logique). Un programme se déroule rarement correctement dès la première fois. Aussi, il importe de disposer de beaucoup de souplesse pour y apporter les modifications nécessaires.

### EDITEUR « MPF-1 PLUS » COMMANDES

#### 1 - Fonctions générales

CONTROL E Introduit et initialise l'Editeur  
CONTROL R Retour à l'Editeur  
(CONTROL) Q Retour au Moniteur

#### 2 - Fonctions « Edition »

D Suppression de la ligne  
I Insertion d'une ligne  
P « n » Impression de « n » lignes — n = 1 par défaut  
R/nom du fichier/ Lecture d'un fichier à partir d'une K7  
W/nom du fichier/ Enregistrement d'un fichier  
Z Edition du programme source

#### 3 - Fonctions « pointeur »

B Pointe et affiche la dernière ligne du programme source  
G « n » Pointe et affiche la n<sup>ème</sup> ligne  
L Affiche le numéro de ligne  
N « n » Déplacement de « n » lignes (vers le bas)  
U « n » Déplacement de « n » lignes (vers le haut)  
T Pointe et affiche la première ligne du programme source

#### 4 - « Chaînes de caractères »

C/ / / Changement d'une chaîne de caractères  
F/ / Recherche et positionnement sur une chaîne de caractères.

#### 5 - Autres fonctions

S Imprimer les adresses limites de l'espace mémoire réservé à l'Editeur et l'espace réellement utilisé  
X Arrêt ou mise en service de l'imprimante  
CR Visualise la ligne suivante.

Fig. 5, tableau III

## D'ELECTRONIQUE

Un coup d'œil sur le tableau III, figure 5, permet de découvrir les facilités particulièrement étonnantes ainsi mises à la disposition de l'utilisateur. Six commandes « pointeurs » permettent de se rendre à n'importe quelle ligne du programme, de se déplacer à partir de là soit vers le haut, soit vers le bas et de connaître le numéro de n'importe quelle ligne (commande L). La première ligne (commande T : Top) et la dernière ligne (commande B : Bottom) sont directement accessibles.

Deux autres commandes, particulièrement intéressantes, portent sur des « chaînes de caractères ». La première permet de changer (commande C) un ensemble de mots dans une ligne d'instructions sans avoir à la retaper entièrement. La seconde permet de se rendre directement à une ligne, et d'en identifier le numéro, en ne désignant qu'un caractère qui peut être par exemple une étiquette.

Toujours dans le but de clarifier les programmes, l'adressage symbolique évite à l'utilisateur d'indiquer les adresses réelles des emplacements de la mémoire. Au lieu de désigner une case par une quantité hexadécimale (2 octets), on utilise un symbole alphanumérique ayant autant que possible une signification avec la fonction.

Par exemple : l'emplacement qui contient le résultat d'une opération peut s'appeler RES (RES = Résultat). Ceci est aussi valable pour l'appel d'une sous-routine.

Ainsi : CALL DIV au lieu de CALL FC 00 ou Ld (RES), A au lieu de Ld (18 3E), A.

Pour effectuer des sauts de programmes, dont bien souvent l'adresse exacte n'est pas déterminée, il suffit de placer dans le champ adresse (première zone d'une instruction en langage Assembleur) une étiquette. Le calcul du déplacement ou la déter-

mination de l'adresse réelle s'effectue automatiquement au moment de l'assemblage.

D'autre part l'intérêt de l'adressage symbolique est très grand dans la phase de mise au point. L'utilisateur peut sans problème insérer et/ou supprimer des lignes d'instructions sans avoir à recalculer les adresses de saut : l'Assembleur s'en charge lui-même.

Enfin, le MPF-1 PLUS peut interpréter sept pseudo-instructions parmi les plus courantes. Ce sont en fait des directives d'assemblage, comprises par l'Assembleur et non traduites comme les codes mnémoniques. Citons ainsi ORG suivi de l'adresse à partir de laquelle l'Assembleur rangera les octets du code objet, et END

qui indique la fin du texte. D'autres commandes telles que EQU affectent la valeur réelle d'une étiquette, ou DEFB, ou DEFM qui attribuent soit une donnée à un octet ou une chaîne de caractères à une étiquette.

Le MPF-1 PLUS permet d'aborder les microprocesseurs et la micro-informatique d'une manière particulièrement attrayante. Chacun peut aussi choisir le langage qui lui semble le plus approprié à son application, et si l'on veut améliorer les performances du micro-ordinateur, il y a tout loisir de combiner l'un et l'autre. Nous présenterons le Basic dans cette rubrique dans un prochain numéro, et peut-être aussi d'autres langages.

**Philippe Duquesne**

