NOUVELLE SERIE N° 7

L'ABC DE LA MICRO-INFORMATIQUE

LE JEU D'INSTRUCTIONS DU 8088

Nous abordons aujourd'hui « la dernière ligne droite » avec la présentation des instructions de chaînes et de contrôle du processeur qui termineront ce descriptif du jeu d'instructions du 8088.

QU'EST-CE QU'UNE CHAINE

En langage 8088, une chaîne est une suite d'octets ou de mots contenus en mémoire. La signification de ces octets ou mots est sans importance et ils peuvent tout aussi bien représenter des nombres en binaire pur ou signé que des caractères ASCII. Le terme chaîne ne doit donc pas vous faire croire, par analogie avec certaines appellations utilisées pour les langages évolués, qu'il ne s'agit que de blocs de caractères.

Ces chaînes peuvent avoir une taille quelconque comprise entre 1 octet et 64 Ko, c'est-à-dire un segment complet.

Attention ! ces notions sont propres au 8088 et ne correspondent à aucune règle, notion ou terminologie générale.

Pour manipuler ces chaînes, le 8088 propose un certain nombre d'instructions que nous allons voir dans un instant ; instructions qui ont toutes en commun de ne pas utiliser d'opérande dans leur écriture. De ce fait, le 8088 fait certaines hypothèses que vous devez absolument respecter. Ce sont les suivantes :

– la chaîne dite source se trouve obligatoirement dans le segment DS et sa position dans ce segment est donnée par le contenu du registre SI ;

– la chaîne dite destination se trouve obligatoirement dans le segment ES et sa position dans ce segment est pointée par le contenu du registre DI.

D'autre part, comme les instructions que nous allons voir sont spécialement prévues pour le traitement des chaînes, l'incrémentation ou la décrémentation des pointeurs SI ou DI est réalisée automatiquement par les instructions. Le choix incrémentation/décrémentation est réalisé au moyen du bit DF du registre d'état, qu'il faudra donc correctement positionner au préalable avec un STD ou CLD, que nous verrons dons un instant. Si ce bit est mis à 1, il y a incrémentation.

Dernier point important à signaler, les chaînes peuvent être, comme nous venons de le voir, des octets ou des mots de 16 bits. Les instructions savent tenir compte de cela et travaillent sur des octets si leurs mnémoniques se terminent par un B (comme Byte ou octet) ou sur des mots si leurs mnémoniques se terminent par W (comme Word ou mot). L'incrémentation ou la décrémentation évoquée ci-avant sait tenir compte de cela ; elle est donc d'une unité pour des instructions « B » ou de deux unités pour des instructions « W ». Comme il peut être délicat pour le programmeur de définir à priori une instruction « B » ou « W », les assembleurs évolués permettent d'écrire les mnémoniques sans suffixe ; le choix étant réalisé par leurs soins au moment de l'assemblage en fonction des définitions des chaînes source et destination.

LES PREFIXES DE REPETITION

Comme toutes les autres instructions, autres que celles de boucle bien sûr, les instructions de choînes ne sont exécutées qu'une fois. Cependant, comme elles font souvent partie de structures répétitives (comparaison de deux blocs de mémoire par exemple), il est possible de les faire précéder d'un préfixe de répétition.

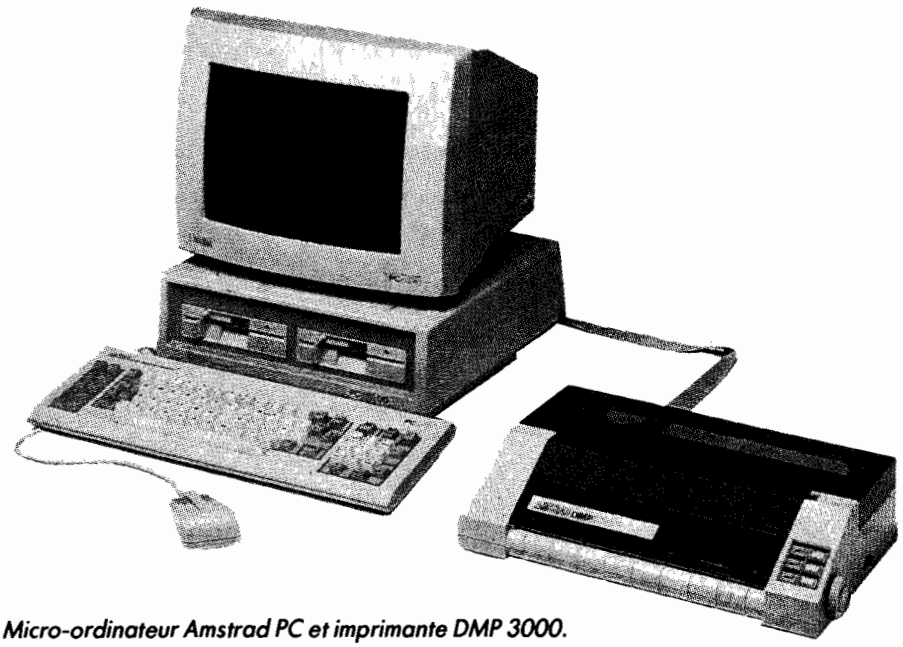
Ce dernier se note REP, se place devant le mnémonique de l'instruction dont il est séparé par un espace et utilise le registre CX comme compteur. Il fait tout simplement répéter l'exécution de l'instruction tant que CX n'est pas arrivé à 0. Par exemple, il y a équivalence, au point de vue programme, entre :

```
MOV CX, 100
REP MOVSW
```

et

```
MOV CX, 100
boucle : MOVSW
LOOP boucle
```

La deuxième forme est cependant d'exécution nettement moins rapide que la première, et il est donc préférable, pour les instructions de chaînes, d'utiliser le préfixe REP plutôt qu'une structure de boucle classique.
Le préfixe REP est pratique, mais s'avère parfois assez limité du fait de l'impossibilité qu'il y a d'arrêter la répétition en fonction d'une condition extérieure (autre que l'arrivée à 0 de CX bien sûr). Pour ce faire, deux autres préfixes vous sont offerts : ce sont REPE (appelé aussi REPZ) et REPNE (appelé aussi REPNZ).
Le préfixe REPE répète une instruction de chaîne tant que CX est différent de 0 et tant que l'indicateur ZF du registre d'état du 8088 est à 1.
Le préfixe REPNE répète quant à lui l'instruction de chaîne tant que CX est différent de 0 et tant que l'indicateur ZF du registre d'état du 8088 est à 0.
Comme cet indicateur est positionné en fonction de la valeur de la donnée manipulée par le 8088, il est possible de faire cesser une répétition d'instruction de chaîne en fonction de ce qu'elle rencontre dans la chaîne ; nous allons en voir un exemple dans quelques instants.



Micro-ordinateur Amstrad PC et imprimante DMP 3000.

DEPLACEMENT D'UNE CHAINE

L'instruction « de base » s'appelle MOVS avec ses deux variantes MOVSB pour des manipulations d'octets ou MOVSW pour des manipulations de mots.
Elle déplace la chaîne repérée par DS:SI à l'adresse repérée par ES:DI. Il faut donc, avant toute utilisation de cette instruction, positionner convenablement les contenus de ces divers registres. Comme pour l'instruction MOV ordinaire, la chaîne source n'est pas affectée et reste identique à elle-même sauf, bien sûr, s'il y a recouvrement destination-source.
Cette instruction permet très facilement de réaliser un sous-programme de recopie d'une zone mémoire dans une autre puisqu'il suffit d'écrire :

```
MOV CX,XXX où XXX est le nombre d'octets à déplacer
MOV AX,SEGS, où SEGS est le segment « source »
MOV DS,AX
MOV AX,SEGD où SEGD est le segment « destination »
MOV ES,AX
MOV SI,SOURCE où SOURCE est l'adresse de début dans le segment
MOV DI,DESTIN où DESTIN est l'adresse de début dans le segment
CLD pour incrémentation de DI et SI
REP MOVSB pour un mouvement octet par octet.
```

Comme vous pouvez le constater, la partie « active » du programme se résume à une ligne, tout le reste n'étant que de l'initialisation. L'exécution est donc relativement rapide puisqu'une seule instruction est concernée : MOVSB.
Si un transfert par mot de 16 bits est possible, il est évidemment indiqué d'utiliser un MOVSW qui fait encore gagner du temps puisqu'elle traite deux octets à la fois.

LECTURE/ ECRITURE D'UNE CHAINE

Deux instructions « de base » sont proposées : LODS pour la lecture et STOS pour l'écriture avec, bien évidemment, leurs versions pour octets ou mots : LODSB, LODSW, STOSB et STOSW.
L'instruction LODS lit le mot ou l'octet pointé par DS:SI et le place dans AL (octet) ou dans AX (mot). Le registre SI est évidemment incrémenté ou décrémenté automatiquement suite à l'exécution de l'instruction et est donc prêt pour la lecture suivante.
L'instruction STOS est la réciproque de LODS et place le mot contenu dans AX ou l'octet contenu dans AL à l'adresse pointée par ES:DI. DI bénéficie bien évidemment de l'incrémentation ou décrémentation automatique comme pour toutes les autres instructions de chaînes.
Un sous-programme d'initialisation d'une zone mémoire avec un octet déterminé peut ainsi s'écrire :

```
CLD
MOV DI,200
MOV CX,100
MOV AL,XX où XX est l'octet d'initialisation
REP STOSB
```

Il a pour effet de placer l'octet XX dans la mémoire commençant en 200 du segment DS et pour une longueur de 256 octets (100 en hexadécimal placé dans le compteur CX).

COMPARAISON DE CHAINES

Une instruction de « base » de mnémonique CMPS et ses deux dérivées CMPSB et CMPSW permettent de comparer automatiquement deux chaînes. Elles fonctionnent comme l'instruction CMP classique en soustrayant les opérandes, mais sans en affecter aucune puisqu'elles se bornent à positionner les indicateurs du registre d'état du

8088 en fonction du résultat obtenu.
Attention ! Cette instruction fonctionne « à l'envers » du CMP classique. Ce dernier fait en effet destination-source alors qu'ici on fait source-destination (merci Intel !). Il est inutile de vous dire que cela génère quelques erreurs lors des premières écritures de programmes car on oublie très souvent ce détail.
Il est très facile, avec CMPS, d'écrire un sous-programme qui compare deux chaînes en mémoire et qui indique l'adresse du premier élément différent puisqu'il suffit de faire :

```
CLD
MOV CX,100
REPE CMPSB
```

Dans ce cas, nous comparons le bloc de 256 octets (le 100 chargé dans CX) pointé par DS:SI avec celui pointé par ES:DI. S'ils sont identiques, le programme se termine avec l'indicateur ZF positianné à 1. Si au moins un octet est différent, le programme se termine avec ZF mis à 0, et DI-1 et SI-1 pointent sur le premier octet différent.

RECHERCHE D'UN ELEMENT DANS UNE CHAINE

Dernière instruction de chaîne, SCAS et ses dérivées SCASB et SCASW permet de rechercher un élément donné dans une chaîne pointée par ES:DI. L'élement à rechercher doit être placé dans AL (octet) ou dans AX (mot) avant l'utilisation de l'instruction qui ne se conçoit évidemment qu'avec un suffixe de répétition ; en effet, on voit mal comment elle pourrait balayer la chaîne à la recherche de la donnée si tel n'était pas le cas.
Pratiquement, cette instruction soustrait la donnée à rechercher de celle lue dans la chaîne et positionne en conséquence les indicateurs du registre d'état du 8088. Comme pour CMPS, ni la donnée contenue dans AL ou AX, ni celles constituant la chaîne ne sont affectées.
Le petit sous-programme suivant vérifie si un octet particulier se trouve dans la chaîne pointée par ES:DI

```
MOV CX,100
MOV AL,XX où XX est l'octet à rechercher
REPNE SCASB
JCXZ pasXX
```

Si l'octet XX ne se trouve pas dans la chaîne, le programme sautera à l'adresse pasXX du fait du JCXZ et du positionnement du bit Z résultant du SCASB qui n'a pas trouvé notre octet.

LES INTERRUPTIONS LOGICIELLES

Tout microprocesseur qui se respecte dispose de possibilités d'interruption ; ces dernières pouvant être logicielles ou matérielles. L'interruption matérielle étant plus facile à appréhender, nous allons nous permettre une petite digression à son sujet d'autant que les micro-ordinateurs style IBM PC et compatibles en font un assez large usage.
Une interruption est un événement extérieur qui peut se produire à n'importe quel moment par rapport à l'exécution d'un programme en cours. Cette interruption doit être prise en compte très rapidement, doit pouvoir déclencher l'exécution d'un programme particulier, mais ne doit pas trop perturber le programme qu'elle a interrompu de façon à ce que celui-ci puisse reprendre normalement lorsque le programme propre à l'interruption est terminé.
Un exemple simple est celui de la gestion des feux tricolores d'une ville grâce à un micro-ordinateur. Ce dernier exécute en permanence le programme de contrôle « de routine ». Si une ambulance doit traverser la ville, un opérateur déclenche une interruption qui fait alors immédiatement exécuter un programme de mise au vert des feux nécessaires. Lorsque ce programme est terminé, il faut évidemment revenir au programme principal de façon correcte afin de ne pas générer une immense pagaille !

Pour que cela soit possible, tous les microprocesseurs utilisent le même principe : la sauvegarde sur la pile. Chronologiquement, une interruption produit donc les phénomènes suivants :

– le microprocesseur termine l'exécution de l'instruction en cours (dans le cas d'une interruption matérielle bien sûr, puisque l'instant d'arrivée de celle-ci est imprévisible) ;
– il sauvegarde sur la pile la valeur du PC (l'adresse de l'instruction qui devrait normalement suivre donc) ;
– il sauvegarde ensuite, toujours sur la pile, un certain nombre de registres internes. Ce nombre dépend du type de microprocesseur utilisé. Dans le cas du 8088, seul le registre d'état est sauvegardé. Dans le cas du 6809 de Motorola par exemple, tous les registres peuvent être sauvegardés ;
– il charge alors le PC avec une valeur lue à un emplacement particulier en mémoire. Cet emplacement s'appelle la table des vecteurs d'interruption. C'est là que sont rangées les adresses de début de tous les programmes d'interruptions susceptibles d'être activés ;
– du fait de ce chargement, il exécute alors le programme d'interruption ;
– lorsque ce dernier est terminé, une instruction spéciale, qui doit impérativement être utilisée, assure le retour au programme principal en rechargeant les registres du micro avec les valeurs qui avaient été sauvegardées sur la pile.

Si un microprocesseur donné dispose de plusieurs entrées d'interruptions matérielles, il dispose nécessairement de plusieurs vecteurs d'interruptions qui leur correspondent. De même, si un microprocesseur dispose de plusieurs sources d'interruptions logicielles, il doit aussi offrir les vecteurs correspondants.
Dans le cas du 8088, il est possible de gérer jusqu'à 256 interruptions logicielles différentes grâce à l'instruction INT qui doit être suivie d'un numéro qui est un entier non signé codé sur un octet.
Cette instruction fait traiter l'interruption dont le vecteur est retrouvé dans la table des vecteurs d'interruptions à un décalage, par rapport au début de cette dernière, égal à 4 fois le numéro de l'interruption choisie.
Pour revenir du programme d'interruption, il faut impérativement terminer ce dernier par une instruction IRET qui récupère sur la pile le contenu du registre d'état et du compteur de programme (CS et IP en fait) qui y avaient été sauvegardés.

Il est également possible de terminer un programme d'interruption par un RETF 2 qui récupère la valeur de CS et IP (l'adresse de retour donc) mais qui ignore le registre d'état. Cette façon de faire permet de récupérer des informations, dans le registre d'état, sur le déroulement du programme d'interruption, ce qui peut être utile dans certains cas.

LES INSTRUCTIONS DIVERSES

Nous avons classé dans ce paragraphe les instructions qu'il était difficile de faire figurer ailleurs. On y trouve en particulier toutes celles de prépositionnement des bits du registre d'état dont le rôle est fort simple puisque :

- CLC met CF à 0 ;
- STC met CF à 1 ;
- CMC complémente CF ;
- CLD met DF à 0 (voir instructions de chaînes ci-avant) ;
- STD met DF à 1 (idem) ;
- CLI met IF à 0 et fait ignorer au 8088 les interruptions masquables ;
- STI met IF à 1 et autorise ainsi la prise en compte des interruptions masquables ;
- PUSHF et POPF qui permettent de sauvegarder sur la pile et de récupérer le registre d'état ;
- LAHF qui recopie l'octet du registre d'état dans AH.
- SAHF qui recopie le registre AH dans l'octet bas du registre d'état.

Ces instructions sont d'emploi particulièrement simple et ne nécessitent pas d'autres développements.
L'instruction HLT permet de synchroniser le 8088 avec un événement extérieur. Elle arrête l'exécution du programme à son niveau et celui-ci ne peut plus reprendre que si l'on fait un RESET (c'est un peu brutal !) ou si une interruption a lieu (et si elle n'est pas masquée par le bit IF bien sûr). Dans ce cas, l'interruption n'est pas traitée mais le programme reprend l'exécution après le HLT.
L'instruction WAIT, comme son nom l'indique, fait attendre le processeur (un peu à la manière de HLT), mais ici c'est sa ligne TEST qui compte et qui doit passer à 0 pour que le programme reprenne. Toutefois, si une interruption se produit pendant un WAIT, elle est traitée et l'attente reprend ensuite.
ESC permet d'inclure dans un programme 8088 des instructions destinées à un coprocesseur. En effet, ce microprocesseur présente la particularité de pouvoir être associé à un coprocesseur (le 8087) ou processeur spécialisé. Dans le cas du 8087, il s'agit d'un microprocesseur spécialement programmé pour faire des calculs scientifiques. Nous ne détaillerons pas plus l'utilisation de ESC qui nécessite la connaissance du coprocesseur pour pouvoir être utilisée.
L'instruction LOCK permet quant à elle de bloquer les accès au bus du 8088 pendant la durée de l'instruction qui la suit. C'est très utile dans les applications multi-microprocesseurs (où plusieurs microprocesseurs se partagent le même bus) pour rendre une instruction importante ininterruptible.
Enfin, nous avons gardé l'instruction la plus complexe pour terminer cette présentation puisqu'il s'agit de NOP. Comme son nom l'indique, cette instruction ne fait rien (No OPeration). Elle a donc tout simplement pour but de consommer du temps (trois cycles machine), ce qui est utile dans les boucles de temporisation.

CONCLUSION

Nous en avons terminé avec cette présentation, un peu longue il est vrai, du jeu d'instructions du 8088. Nous pourrons, le mois prochain, parler un peu d'assembleur et des notions qui s'y rapportent, ce qui vous permettra d'écrire vos premiers programmes et de faire quelques manipulations simples si vous avez la chance d'avoir un IBM PC ou compatible sous la main.

C. TAVERNIER