Deuxième partie

Autres fonctions d'un microprocesseur

Chapitre 7

La mémoire segmentée du i8086

Comme nous l'avons déjà dit, les différences entre le `i8085` et le `i8086` sont de deux ordres :

- Les registres ont une capacité de 16 bits au lieu de 8 bits.
- La capacité mémoire (maximum) du `i8086` a été portée de 64 Ko à 1 Mo, c'est-à-dire qu'il y a vingt broches de sortie au lieu de 16. Une **adresse physique** est donc maintenant comprise entre `00000h` et `FFFFFh`.

Cependant ce microprocesseur intermédiaire (entre le *i8085* et le `i80286`) était toujours muni d'un bus d'adresse de 16 bits là où il en aurait fallu 20. Les concepteurs du `i8086` ont donc jonglé en inventant la notion de *mémoire segmentée*. Puisque ce microprocesseur (ou, plus exactement, une version affaiblie, le `i8088`) fut utilisé dans la première version des micro-ordinateurs IBM, l'étude de cette mémoire segmentée avait été très étudiée dans les années 1980. Elle n'a évidemment plus beaucoup d'importance de nos jours.

7.1 Mémoire segmentée du i8086

7.1.1 Notion de segment de mémoire

Problème.- Un petit problème se pose au vu de la capacité mémoire et du bus d'adresse de 16 bits. En effet :

Comment désigner 2^{20} éléments mémoire avec des registres d'un mot, c'est-à-dire de deux octets ou 16 bits ?

Il faut utiliser plus d'un mot pour désigner l'adresse.

Première solution.- On pourrait imaginer qu'un premier mot contienne les seize premiers bits de l'adresse et un second mot les quatre derniers bits. Ce ne fut pas le choix d'Intel, qui a préféré **segmenter** la mémoire.

Méthode du segment et du décalage.- Intel a découpé la mémoire en **segments** de 64 Ko. Le choix de la capacité d'une unité de mémoire appelée *segment* permet d'assurer la compatibilité avec le `i8085`.

Une adresse sera décomposée en deux parties : l'adresse d'un **segment**, c'est-à-dire d'une partie de la mémoire de 64 Ko, et un **décalage** (en anglais *offset*), qui permet de repérer l'emplacement de l'octet à l'intérieur de ce segment.

Le décalage, emplacement à l'intérieur d'un segment de 64 Ko, est compris entre `0000h` et `FFFFh`.

Entrelacement des segments.- Dans 1 Mo, il y a de la place pour seize segments disjoints de 64 Ko. Intel a décidé de ne pas utiliser des segments disjoints mais de permettre qu'ils empiètent les uns sur les autres. Un segment peut commencer à n'importe quelle adresse divisible par seize : l''adresse du début d'un segment est comprise entre `0000h` et `FFF0h`, exprimée par un entier divisible par seize et donc se terminant par `0h`.

7.1.2 Désignation d'un élément de mémoire segmentée

Adresse physique et adresse logique.- Nous avons donc besoin de deux registres pour désigner une **adresse physique** de 20 bits : un registre de 16 bits pour le décalage et un registre d'au moins 4 bits pour le segment. Au vu du choix d'Intel, on a également 16 bits pour désigner un segment.

L'**adresse logique** est la donnée d'une adresse de segment et d'un décalage. L'adresse physique est décodée à partir de l'adresse logique grâce à un circuit combinatoire. Une adresse logique s'écrit symboliquement sous la forme :

`AS:AO`

où `AS` est l'adresse du segment et `AO` le décalage (0 pour *Offset*).

On permet à `AO` de prendre toutes les valeurs entre `0000h` et `FFFFh`, ce qui est normal. On permet également à `AS` de prendre toutes ces valeurs, alors que `0h` à `Fh` est en théorie suffisant. L'adresse physique `AP` est alors donnée par :

`AP = AS` $\times$ `16 + AO.`

Exemple.- L'adresse logique :

2300:85F2

représente l'adresse physique :

```
23000h + 85F2h = 2B5F2h,
```

c'est-à-dire que l'on a multiplié par seize l'adresse du segment et qu'on lui a ajouté le décalage.

Remarque.- Une adresse physique ne correspond pas à un seul couple `AS:AO`, puisqu'il y a entrelacement des segments.

7.1.3 Chargement depuis la mémoire

Syntaxe.- Pour placer le contenu de l'emplacement mémoire d'adresse logique `AS:AO` dans un registre d'un octet, on place l'adresse de segment `AS` dans le registre `DS` et on utilise l'instruction :

```
mov registre, [AO]
```

où `AO` est le décalage en hexadécimal.

Exemple.- Pour obtenir la valeur de l'emplacement mémoire `210:1004h` on peut utiliser le programme suivant :

```
C:>debug
-a
249C:0100 mov bx,0210
249C:0103 mov ds,bx
249C:0105 mov al,[1004]
249C:0108 int 3
249C:0109
-g
AX=0005 BX=0210 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0210 ES=249C SS=249C CS=249C IP=0108 NV UP EI PL NZ NA PO NC
249C:0108 CC INT 3
-q
```

sans moyen de vérifier le résultat pour l'instant.

Remarque.- On a utilisé le registre `AX` pour initialiser `DS` car on ne peut pas utiliser l'adressage immédiat pour les registres de segment.

7.1.4 Stockage en mémoire

Syntaxe.- Pour placer le contenu d'un registre (de capacité un octet) dans l'emplacement mémoire d'adresse logique `AS:AO`, on place l'adresse de segment `AS` dans `DS` et on utilise l'instruction :

```
mov [AO], registre
```

Exemple.- Pour placer la valeur `80h` dans l'emplacement mémoire `210:1004h` on peut utiliser le programme suivant :

```
C:>debug
-a
249C:0100 mov bx, 0210
249C:0103 mov ds, bx
249C:0105 mov al, 80
249C:0107 mov [1004], al
```

```
249C:010A mov cl, [1004]
249C:010E int 3
249C:010F
-g
AX=0080 BX=0210 CX=0080 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0210 ES=249C SS=249C CS=249C IP=010E NV UP EI PL NZ NA PO NC
249C:010E CC INT 3
-q
```

On a vérifié de plus que l'opération a bien été effectuée en plaçant le contenu de cet élément mémoire dans le registre `CL`.

7.2 Sauts inter-segmentaires

Introduction.- Lorsqu'on passe du modèle plat au modèle segmenté, lors d'un saut (inconditionnel ou conditionnel), il faut préciser à la fois le segment et le décalage.

Exemple 1.- Au démarrage un microprocesseur Intel `8086` initialise le registre `cs` à `FFFFh` et le pointeur d'instruction `ip` à 0. Sur un compatible PC la mémoire centrale commençant à l'adresse `FFF0h` est de la mémoire ROM contenant le BIOS permettant de démarrer l'ordinateur.

Vérifions ce qu'il en est sur notre ordinateur :

```
C:>debug
-r cs
CS 1568
:ffff
-r ip
IP 0100
:0
-r
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1568  ES=1568  SS=1568  CS=FFFF  IP=0000   NV UP EI PL NZ NA PO NC
FFFF:0000 EA5BE000F0    JMP     F000:E05B
-d cs:0
FFFF:0000  EA 5B E0 00 F0 30 32 2F-31 38 2F 30 35 00 FC F9   .[...02/18/05...
FFFF:0010  34 12 00 00 00 00 00 00-00 00 00 00 00 00 00 00   4...............
FFFF:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
FFFF:0030  70 00 2E 8E 06 30 00 BF-7F 01 B9 02 00 AB 47 47   p....0........GG
FFFF:0040  E2 FB CB 56 50 51 52 57-55 1E 06 53 8B EC 8B 76   ...VPQRWU..S...v
FFFF:0050  12 2E 8E 1E 30 00 8B 44-02 A2 22 00 88 26 08 01   ....0..D.."..&..
FFFF:0060  8B 34 C4 1E 18 00 26 8A-47 01 26 8A 67 0D 26 8B   .4....&.G.&.g.&.
FFFF:0070  4F 12 26 8B 57 14 97 26-8A 47 02 2E 3A 04 73 2C   O.&.W..&.G..:.s,
-q
```

Nous allons revenir sur les cinq premiers octets. Remarquons que les huit octets suivants contiennent la date de révision du BIOS, que nous pouvons lire en clair dans la partie ASCII.

Lorsqu'on lit les registres, la dernière ligne désassemble la première instruction, constituée de cinq octets `EA5BE000F0h`. Sa version symbolique est donc :

```
JMP F000:E05B
```

qui nous montre la façon d'indiquer les sauts inter-segmentaires.

Remarque.- Sous MS-DOS, on peut faire exécuter le programme. On obtient l'analogue d'un démarrage, appelé *démarrage à froid*. Sous Windows, ce démarrage à froid n'est pas permis à partir d'une fenêtre MS-DOS. Sous Windows

XP, par exemple, ceci conduit à un processus qui prend presque tout le temps machine.

Syntaxe.- Un saut inter-segmentaire sera exécuté par l'instruction en langage symbolique :

```
jxx SSSS:DDDD
```

où `jxx` est un saut inconditionnel ou l'un des nombreux sauts inconditionnels, `SSSS` l'adresse de segment et `DDDD` le décalage.

Exemple 2.- Écrivons un programme qui utilise deux sauts inconditionnels inter-segmentaires. Pour des raisons de compatibilité avec une fenêtre MS-DOS nous utiliserons la syntaxe des sauts inter-segmentaires mais en restant dans le même segment :

```
C:>debug
-a
15BA:0100 mov ax,1
15BA:0103 jmp 15BA:200
15BA:0106 add ax,3
15BA:0109 int3
15BA:010A
-a 200
15BA:0200 add ax,2
15BA:0203 jmp 15BA:0106
15BA:0206
-g

AX=0006  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=15BA  ES=15BA  SS=15BA  CS=15BA  IP=0109   NV UP EI PL NZ NA PE NC
15BA:0109 CC            INT     3
-q
```

Le contenu du registre `ax` est 6, la somme de 1, 2 et 3. Les sauts ont donc bien été effectués.