Chapitre 23

Dernières instructions

Nous allons passer en revue dans ce chapitre les instructions du 8086 que nous n'avons pas encore rencontrées.

23.1 Adressage indexé

Avec l'**adressage indexé** l'instruction spécifie à la fois un registre d'index et une constante, appelée **déplacement** (*displacement* en anglais). Le contenu du registre d'index est ajouté à la constante, la somme constituant l'adresse de l'opérande.

Cette technique d'adressage permet d'accéder successivement aux éléments d'un bloc, appelé **table** dans ce contexte : la constante est l'adresse du début de la table et le registre d'index est utilisé pour accéder successivement aux différents éléments de la table. Ceci requiert évidemment la disponibilité d'instructions d'incrémentation et de décrémentation du registre d'index.

Combiné avec les adressages précédents, ceci donne une grande variété de formation de l'**adresse effective** AE dans un segment. Cette adresse est le résultat de l'addition du déplacement, qui peut être codé sur un mot (entier naturel) ou un octet (entier relatif), d'un registre de base (BP ou BX) et d'un registre index (SI ou DI), chacun de ces trois termes étant facultatif, ce qui donne 16 possibilités comme le montre le tableau suivant :

Adresse effective	par rapport au segment
DEP	DS
$[BP]$	interdit
$[BX]$	DS
$[SI]$	DS
$[DI]$	DS
$[BP]$ + DEP	SS
$[BX]$ + DEP	DS
$[SI]$ + DEP	DS
$[DI]$ + DEP	DS
$[BP] + [SI]$	SS
$[BP] + [DI]$	SS
$[BX] + [SI]$	DS
$[BX] + [DI]$	DS
$[BP] + [SI]$ + DEP	SS
$[BP] + [DI]$ + DEP	SS
$[BX] + [SI]$ + DEP	DS
$[BX] + [DI]$ + DEP	DS

La désignation de l'adresse effective s'effectue grâce à l'octet maintes fois rencontré dans les instructions :

```
| mod reg r/m |
```

mais pas encore décrit complètement. Nous avons vu que `reg` désigne le registre (ou le premier registre) de l'instruction. Les termes `mod` et `r/m` permettent de spécifier l'adresse effective conformément au tableau suivant :

mod = 00	
r/m	AE
000	$[BX] + [SI]$
001	$[BX] + [DI]$
010	$[BP] + [SI]$
011	$[BP] + [DI]$
100	$[SI]$
101	$[DI]$
110	DEP (16 bits)
111	$[BX]$

mod = 01 ou 10			
r/m	AE	mod	
		01	10
000	$[BX] + [SI] +$		
001	$[BX] + [DI] +$	DEP	DEP
010	$[BP] + [SI] +$		
011	$[BP] + [DI] +$	8 bits	16 bits
100	$[SI] +$		
101	$[DI] +$		
110	$[BP] +$		
111	$[BX] +$		

mod = 11 : r/m = reg

23.2 Instructions d'échange

Langage symbolique.- Le 8086 dispose, outre les instructions de transfert simple des données (à savoir MOV), d'instructions qui échangent entre eux les contenus de deux emplacements. Ces instructions sont représentées par :

```
XCHG but, source
```

pour l'anglais *eXCHanGe*. Bien entendu, pour cette instruction, il n'est pas très utile de distinguer entre but et source puisque les deux opérandes jouent un rôle symétrique.

Langage machine.- L'échange peut avoir lieu entre registres ou registre et case mémoire :

- 1^o) Un échange entre l'accumulateur AX et un registre (16 bits) est codé sur un octet par :

```
| 100 10reg |
```

- 2^o) Sinon, l'échange est codé sur deux à quatre octets par :

```
| 1000 011w | mod reg r/m |         |         |
```

23.3 Pas d'opération

Langage symbolique.- L'instruction :

```
NOP
```

(pour l'anglais *No OPeration*) ne fait rien.

Intérêt.- Cette instruction peut être utilisée pour ralentir volontairement un programme. Elle peut également être utilisée lors du développement de programmes pour réserver de la place.

Langage machine.- L'instruction NOP est codée sur un octet par `1001 0000b`, soit 90h.

23.4 Arrêt du programme

Langage symbolique.- L'instruction :

```
HALT
```

interrompt les opérations du microprocesseur jusqu'à ce qu'une réinitialisation ou une interruption se produisent.

Langage machine.- L'instruction HALT est codée sur un octet par `| 1111 0100 |`, soit F4h.

23.5 Tables de traduction

Introduction.- Une opération courante sur les chaînes de caractères consiste à remplacer un caractère par un autre suivant une **table de traduction**. Les concepteurs du 8086 ont implémenté une instruction pour faciliter cette traduction.

Langage symbolique.- L'instruction :

```
XLAT
```

(pour l'anglais *tranSLATe*) remplace le contenu actuel du registre AL par l'octet d'un tableau de 256 octets d'indice spécifié par AL. L'adresse du début de la table est indiquée par le couple de registres DS :BX.

Remarque.- L'instruction XLAT est équivalente à l'instruction *illégale* suivante :

```
MOV AL,[BX + AL]
```

Langage machine.- L'instruction XLAT est codée sur un octet par `| 1101 0111 |`, soit D7h.

23.6 A

AAA
AAD
AAM
AAS
CBW
CWD
ESC
INTO
JCXZ
LAHF
LDS
LEA
LES
LOCK
SAHF
TEST
WAIT

23.7 Historique

L'adressage indexé n'existe pas sur le 8080. Il a été ajouté sur le Z80 en même temps que deux registres d'index, dénommés IX et IY.