Chapitre 5

Structures de contrôle

Nous avons vu, lors de l'initiation à la programmation, que ce qui faisait toute la puissance d'un ordinateur, et la différence avec une calculatrice, étaient ce qu'on appelle les *structures de contrôle*. Nous allons voir quelles sont les types de structures de contrôle sur un microprocesseur et comment les mettre en pratique sur le microprocesseur `i8086`.

5.1 Étude générale

Il y a trois types de structures de contrôle sur un microprocesseur : le *séquencement*, bien sûr, le *saut inconditionnel* et les *sauts conditionnels* :

- Le **séquencement** est la structure par défaut et donc n'exige rien de particulier du point de vue du programmeur. Les instructions sont exécutées l'une parès l'autre, dans l'ordre de la programmation, sauf dans le cas d'une **instruction de saut**.
- Un **saut inconditionnel** indique d'aller à telle instruction. Il faut repérer cette instruction, ce qui est fait bien entendu par son numéro (c'est-à-dire son adresse). L'utilisation d'un saut inconditionnel seul a peu d'intérêt puisque soit on saute une instruction une fois pour toute, soit on génère une boucle infinie.
- Un **saut conditionnel** indique d'aller à telle instruction lorsque telle situation est réalisée.

5.2 Cas du modèle plat

Nous avons vu que Intel a segmenté la mémoire. par conséquent il faut distinguer les **sauts intra-segmentaires**, dont l'adresse se à laquelle il faut aller se trouve dans le même segment que l'adresse actuelle, des **sauts inter-segmentaires**. Le premier type de sauts correspond au modèle plat de la mémoire. C'est celui-ci que nous allons abordé dans ce chapitre. Nous étudierons le cas des sauts inter-segmentaires au chapitre 7.

5.2.1 Saut inconditionnel

Syntaxe.- Le format d'un saut inconditionnel est :

```
jmp adresse
```

où `adresse` est un nombre en hexadécimal. Le mnémonyme `jmp` provient évidemment de *to jump*, c'est-à-dire *sauter* en anglais.

Exemple.- Écrivons un programme qui additionne 1, 2 et 3 et place le résultat dans le registre `AX`. Effectuons cependant un saut inconditionnel avant l'addition de 3, ce qui a pour conséquence que cette dernière n'est pas prise en compte :

```
C:>debug
-a
249C:0100 mov ax,1
249C:0103 add ax,2
249C:0106 jmp 10B
249C:0108 add ax,3
249C:010B int3
249C:010C
-g
AX=003 BX=0000 CX=0000 DX=0002 SP=FFEE BP=0000 SI=0000 DI=0000
DS=249C ES=249C SS=249C CS=249C IP=010B NV UP EI PL NZ NA PE NC
249C:010B CC INT 3
-q
```

Remarque.- L'écriture en langage machine, et même en langage symbolique de `debug`, est difficile car il faut déterminer l'adresse adéquate.

5.2.2 Sauts conditionnels

5.2.2.1 Affectation des indicateurs

Introduction.- Les conditions prises en compte lors des sauts conditionnels sont les valeurs du registre des indicateurs. Il faut donc pouvoir initialiser celui-ci ou, tout au moins, savoir ce qui fait changer son contenu. Comme nous l'avons déjà dit, on ne peut pas changer directement son contenu ; celui-ci est revu après le déroulement de chaque instruction et, en particulier pour ce qui nous intéresse, après chaque instruction arithmétique.

Opérations arithmétiques et registre des indicateurs.- Indiquons les conséquences des opérations arithmétiques sur chacun des bits du registre des indicateurs :

- l'**indicateur de retenue CF**, comme nous l'avons vu, représente le neuvième (respectivement dix-septième) bit d'une addition ou d'une soustraction de deux nombres de huit (respectivement seize) bits ;
- l'**indicateur de parité PF** est positionné lorsque le résultat d'une opération sur 8 ou 16 bits présente un nombre pair de bits égaux à 1 ;

- l'**indicateur de retenue auxiliaire AF** fonctionne comme `CF` mais il réagit aux opérations sur un demi-octet ; il est utilisé avec le format de nombres appelé BCD, que nous verrons plus tard ;
- l'**indicateur de zéro ZF** est positionné lorsque le résultat d'une opération sur 8 ou 16 bits donne zéro ;
- l'**indicateur de signe SF** est positionné lorsque le résultat d'une opération sur 8 ou 16 bits est négatif ;
- l'**indicateur de débordement OF** intervient lorsqu'on effectue des calculs sur des entiers relatifs (*nombres signés* dans le jargon informatique) et lorsque le résultat d'une opération sur 8 ou 16 bits dépasse le domaine autorisé.

5.2.2.2 L'instruction de comparaison

Les opérations arithmétiques détruisent le contenu de la destination. Lorsque cela n'est pas souhaitable on utilise souvent l'instruction :

```
cmp destination, source
```

qui positionne le registre des indicateurs comme le ferait la soustraction `sub` mais sans changer la valeur de destination.

5.2.2.3 Les instructions de sauts conditionnels

Le tableau 5.1 donne la liste des différentes instructions de sauts conditionnels avec leurs mnémonymes, l'explication et la condition de saut.

Remarquons que certaines instructions de saut ont des noms différents mais effectuent la même chose, par exemple `je` et `jz`.

5.2.2.4 Un exemple d'utilisation

Nous avons déjà dit que, sur certains microprocesseurs, la multiplication de deux entiers naturels n'est pas implémentée. On peut alors programmer cette multiplication. Un algorithme naïf pour cela (pas très efficace) consiste à additionner à lui-même le multiplicateur le nombre de fois désigné par le multiplicande.

On place le multiplicateur dans `AX` et le multiplicande dans `BX`, le résultat se trouvear dans `CX`. On initialise `CX` à zéro. Si `BX` est non nul, on ajoute `AX` à `CX` et on décrémente `BX`. On recommence jusqu'à ce que `BX` soit nul.

Ceci conduit au programme suivant pour multiplier 5 par 3 :

```
C:>debug
-a
249C:0100 mov ax,5
249C:0103 mov bx,3
249C:0106 mov cx,0
249C:0109 cmp bx,0
249C:010C jz 113
249C:010E add cx,ax
249C:0110 dec bx
249C:0111 jmp 10C
249C:0113 int3
249C:0114
-g
AX=0005 BX=0000 CX=000F DX=0002 SP=FFEE BP=0000 SI=0000 DI=0000
DS=249C ES=249C SS=249C CS=249C IP=0113 NV UP EI PL ZR NA PE NC
249C:0113 CC INT 3
-q
```

mnémonyme	condition de saut	test sur les indicateurs
Tests généraux		
je	equal	zf = 1
jz	zero	zf = 1
jne	not equal	zf = 0
jnz	not zero	zf = 0
Opérations sur les nombres non signés		
ja	above	(cf or zf) = 0
jnbe	not below or equal	(cf or zf) = 0
jae	above or equal	cf = 0
jnb	not below	cf = 0
jb	below	cf = 1
jnae	not above or equal	cf = 1
jc	carry	cf = 1
jbe	below or equal	(cf or zf) = 1
jna	not above	(cf or zf) = 1
Opérations sur les nombres signés		
jg	greater	zf = 0 and sf = of
jnle	not les than or equal	zf = 0 and sf = of
jge	greater or equal	sf = of
jnl	not less	sf = of
jl	less	$sf \neq of$
jnge	not greater or equal	$sf \neq of$
jle	less or equal	(zf = 1) or ($sf \neq of$)
jng	not greater	(zf = 1) or ($sf \neq of$)
Autres opérations		
jo	overflow	of = 1
jno	not overflow	of = 0
jp	parity	pf = 1
jpe	parity even	pf = 1
jnp	not parity	pf = 0
jpo	odd parity	pf = 0
js	sign	sf = 1
jns	no sign	sf = 0

TAB. 5.1 – *Les sauts conditionnels*

qui donne bien le résultat dans CX, à savoir Fh ou 15.