

# Microprocesseurs (MIC)

## Chap. ? : Les modes d'adressage

# Adresser les données

De nombreuses instructions manipulent des données

- ▶ transferts mémoire-registres : **MOV AX, 3**
- ▶ calculs : **ADD EAX, EBX**
- ▶ tests : **BT AX, 2**
- ▶ ...

Il faut pouvoir indiquer où se trouve / où mettre chaque donnée

C'est ce qu'on appelle les **modes d'adressage**

# Modes de base

**Immédiat** la donnée est directement dans l'instruction

ex : **MOV AX**, 42

**Registre** la donnée est (doit être placée) dans un registre

ex : **MOV** AX, BX

**Direct** la donnée est (doit être placée) à l'adresse donnée dans l'instruction

ex : **MOV AX**, [0xB8A0]

ex : **MOV** [0xB8A2], AX

# Modes de base – Piège

Attention à ne pas confondre **une adresse** et **ce qu'elle contient**

```
MOV EAX, 0xB8A0 ; immédiat. EAX reçoit la valeur 0xB8A0.  
MOV EAX, [0xB8A0] ; direct. EAX reçoit la valeur (4 bytes) à l'adresse 0xB8A0.  
MOV AX, [0xB8A0] ; direct. AX reçoit la valeur (2 bytes) à l'adresse 0xB8A0.  
MOV AL, [0xB8A0] ; direct. AL reçoit la valeur (1 byte) à l'adresse 0xB8A0.
```

D'ailleurs

```
MOV AL, 0xB8A0 ; Ne compile pas ! AL trop petit pour y mettre la valeur  
MOV 0xB8A0, AL ; Ne compile pas ! Pas d'immédiat à gauche.
```

# Modes de base – Autre piège

Un **label** est un nom symbolique pour une **adresse**

```
MOV EAX, brol ; immédiat. On met dans EAX l'adresse brol.  
MOV EAX, [brol] ; direct. EAX reçoit la valeur (4 bytes) à l'adresse brol.  
MOV AX, [brol] ; direct. AX reçoit la valeur (2 bytes) à l'adresse brol.  
MOV AL, [brol] ; direct. AL reçoit la valeur (1 byte) à l'adresse brol.
```

D'ailleurs

```
MOV [brol], AX ; on met le contenu de AX à l'adresse brol  
MOV brol, AX ; Ne compile pas ! On ne peut pas modifier un label.
```

# Modes de base – Utilisation

Ces trois modes permettent de traduire les instructions simples des langages de haut niveau.

Par exemple

```
b ← a + 2
```

pourrait se traduire<sup>1</sup>

```
MOV AX, [a] ; registre , direct  
ADD AX, 2   ; registre , immédiat  
MOV [b], AX ; direct , registre
```

---

1. Valable pour des variables globales. En pratique, a et b sont probablement des variables locales ; elles seront dès lors stockées dans une pile, ce qui modifie un peu les instructions.

# RISC vs CISC

	RISC	CISC
Signification	Reduced Instruction Set Computer	Complex Instruction Set Computer
Modes d'adressage	Peu	Beaucoup
Processeur	Moins complexe	Plus complexe

Les architectures **CISC** permettent de coder plus facilement des instructions de haut niveau comme

```
maStructure.unChamp <- 1  
tab[3] <- 2  
tab[3].unChamp <- 3
```

# Adressage indirect (registre)

La donnée est à une adresse donnée par un registre

Exemple :

```
MOV EBX, 0xB8A0 ; EBX contient l'adresse donnée  
MOV EAX, [EBX] ; on met dans EAX la donnée se trouvant à l'adresse 0xB8A0
```

Utile pour traduire les instructions manipulant les  
**pointeurs / références**

Exemple : Pour traduire  $\text{objet1} \leftarrow \text{objet2}$  (copie des références), on pourrait avoir

```
MOV EAX, [objet2] ; l'objet référencé par objet2  
MOV [objet1], EAX ; et si on ne met pas les crochets ?
```



# Adressage indirect avec déplacement

Adresse obtenue en ajoutant un déplacement à une adresse dans un registre

Exemple :

```
MOV AX, [EBX + 4] ; AX reçoit la donnée se trouvant à l'adresse  
                  ; donnée par (le contenu de) EBX + 4
```

Utile pour traduire les instructions manipulant les  
**structures**

Exemple : Pour traduire `maStructure.unChamp <- 1`, on  
pourrait avoir

```
MOV EAX, maStructure ; pas de crochet ici  
MOV [EAX + 8], 1      ; Le '8' dépend des champs précédents dans la structure
```

# Adressage indirect indexé

Le déplacement est lui aussi dans un registre. On y applique un facteur multiplicatif.

Exemple :

```
MOV AX, [EBX + 4 * ECX] ; AX reçoit la donnée se trouvant à l'adresse  
; donnée par (le contenu de) EBX + 4 * (le contenu de) ECX
```

Utile pour traduire les instructions manipulant les **tableaux**

Exemple : Pour traduire `tab[3] <- 1`, on pourrait avoir

```
MOV EAX, tab  
MOV EBX, 3  
MOV [EAX + 4 * EBX], 1 ; Le '4' est la taille d'une case.  
; on comprend mieux que les tableaux commencent à 0 dans de nombreux langages
```

Nous n'avons pas tout dit :

- ▶ Les modes portent parfois d'autres noms
- ▶ Il existe encore d'autres modes d'adressages
- ▶ Chaque processeur dispose d'un sous-ensemble des modes possibles
- ▶ Chaque assembleur dispose de sa propre syntaxe pour ces modes d'adressages