

Les modes d'adressages du processeur 9S12

Module Info 2

Sebastien.Kramm@univ-rouen.fr

IUT GEII Rouen

2013-2014

- Un programme doit traiter des données.
- A l'échelle d'un processeur, ces données sont des **octets** ou des **mots** (16 bit, " *Word* ").

- Un programme doit traiter des données.
- A l'échelle d'un processeur, ces données sont des **octets** ou des **mots** (16 bit, " *Word* ").
- Le terme **mode d'adressage** désigne la façon dont la donnée est fournie à l'instruction
- Les valeurs peuvent être :
 - données directement dans le programme (fixées à l'assemblage)
⇒ mode d'adressage **immédiat**,

- Un programme doit traiter des données.
 - A l'échelle d'un processeur, ces données sont des **octets** ou des **mots** (16 bit, " *Word* ").
 - Le terme **mode d'adressage** désigne la façon dont la donnée est fournie à l'instruction
 - Les valeurs peuvent être :
 - données directement dans le programme (fixées à l'assemblage)
⇒ mode d'adressage **immédiat**,
 - stockées dans la mémoire
 - en RAM : variables, ou constantes initialisée par le prog.
 - en ROM : constantes.
- ⇒ mode d'adressage **étendu** ou **indexé**.

- Sont désignés par un nom
 - Inhérent (*Inherent*)
 - Immédiat (*Immediate*)
 - Etendu (*Extended*)
 - Indexé (*Indexed*)
 - Relatif (*Relative*)
- Liés à chaque famille de processeur

Mode d'adressage **inhérent**

- Certaines instructions n'ont pas besoin d'opérande.
- Exemples :
 - `inx` : _____
 - `dey` : _____
 - `aba` : _____
 - `coma` : _____
- On parle alors de mode d'adressage "inhérent".

Mode d'adressage **immédiat**

- La valeur est donnée directement dans le programme.
- Symbolisé dans le source par le symbole '#'
- Exemples :
 - `ldx #2` : _____
 - `adda #45` : _____
 - `cmpb #toto` : _____

Mode d'adressage **immédiat**

- La valeur est donnée directement dans le programme.
- Symbolisé dans le source par le symbole '#'
- Exemples :

- `ldx #2` : _____

- `adda #45` : _____

- `cmpb #toto` : _____

Avec, par exemple :

```
toto equ 4
```

Mode d'adressage **immédiat**

- La valeur est donnée directement dans le programme.
- Symbolisé dans le source par le symbole '#'
- Exemples :

- `ldx #2` : _____

- `adda #45` : _____

- `cmpb #toto` : _____

Avec, par exemple :

```
toto equ 4
```

- Attention : il est impossible "d'écrire dans le programme".
⇒ `staa #2` : interdit !

- Symbolisé par aucun signe particulier.
- Exemples :
 - `ldaa $1234` \Rightarrow charge dans A la valeur contenue à l'adresse _____
 - `staa $1234` \Rightarrow sauvegarde la valeur de A à l'adresse _____
 - `addb compteur` \Rightarrow additionne B avec la valeur de la variable 'compteur'
(soit la valeur contenue à l'**adresse** correspondant au symbole compteur)
 - `cpx compteur` \Rightarrow compare le registre X avec la valeur de la variable 16 bits compteur
(avec la valeur 16 bits située à l'adresse compteur).

Cas particulier des branchements

- Instruction de branchement \Rightarrow déroutent l'exécution vers un autre endroit du programme.

Cas particulier des branchements

- Instruction de branchement \Rightarrow déroutent l'exécution vers un autre endroit du programme.
- Deux types :
 - **Conditionnel** : le déroutement ne s'effectue que **si** une certaine condition est remplie.
Exemples : `bne labas` (*Branch if Not Equal*)
 - **Inconditionnel** : le déroutement est systématique.
Exemples :
`bra etiquette` (*Branch Always*)
`bsr monsp` (*Branch to Sub Routine*)

Cas particulier des branchements

- Instruction de branchement \Rightarrow déroutent l'exécution vers un autre endroit du programme.
- Deux types :
 - **Conditionnel** : le déroutement ne s'effectue que **si** une certaine condition est remplie.
Exemples : `bne labas` (*Branch if Not Equal*)
 - **Inconditionnel** : le déroutement est systématique.
Exemples :
`bra etiquette` (*Branch Always*)
`bsr monsp` (*Branch to Sub Routine*)
- Ces instructions utilisent toujours le mode d'adressage **relatif**.

- La valeur donnée avec le code opérateur correspond au **déplacement** (signé) par rapport à l'adresse courante.

Mode d'adressage **relatif**

- La valeur donnée avec le code opératoire correspond au **déplacement** (signé) par rapport à l'adresse courante.
- Exemple : soit un programme en \$1000 :

```
1      1000 20 02  debut bra labas
2      1002 86 01          ldaa #1 ;autre chose ...
3      1004 18 06  labas aba
```

Mode d'adressage **relatif**

- La valeur donnée avec le code opératoire correspond au **déplacement** (signé) par rapport à l'adresse courante.
- Exemple : soit un programme en \$1000 :

```
1      1000 20 02  debut bra labas
2      1002 86 01           ldaa #1 ;autre chose ...
3      1004 18 06  labas aba
```

- L'adresse labas est en \$1004, le déroulement normal du programme se ferait en \$1002 \Rightarrow Le déplacement est de + 2.
- En pratique, l'assembleur fait le calcul du déplacement pour nous.

Mode d'adressage **indexé**

- Utilité : permet d'accéder à des tables de valeurs.
- Principe : on utilise un registre d'index, préalablement initialisé, qui pointe sur la table (équivalent à un pointeur en C).

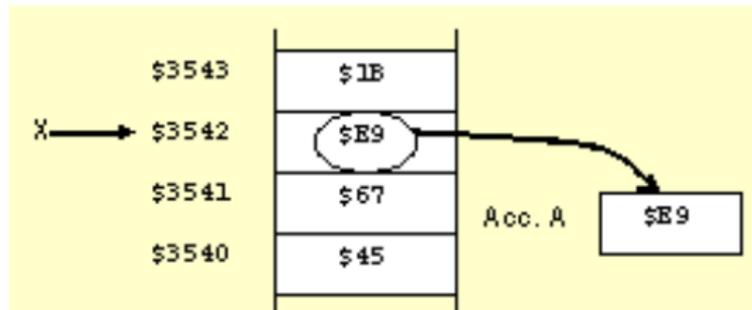
Mode d'adressage **indexé**

- Utilité : permet d'accéder à des tables de valeurs.
- Principe : on utilise un registre d'index, préalablement initialisé, qui pointe sur la table (équivalent à un pointeur en C).
- Symbolisé par la présence de " ,x" ou " ,y" dans le champ opérande.

Mode d'adressage **indexé**

- Utilité : permet d'accéder à des tables de valeurs.
- Principe : on utilise un registre d'index, préalablement initialisé, qui pointe sur la table (équivalent à un pointeur en C).
- Symbolisé par la présence de " ,x" ou " ,y" dans le champ opérande.
- Exemple :

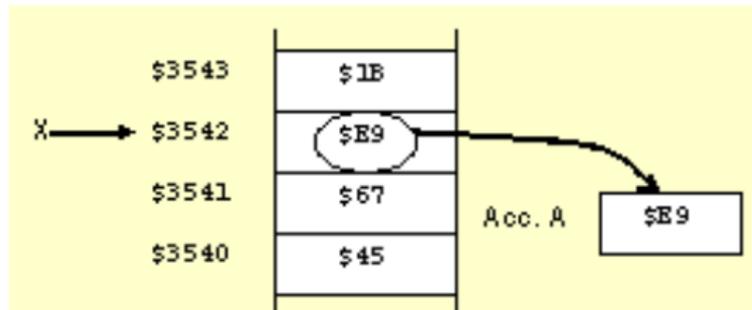
```
1  ldx  # $3542
2  ldaa 0,x
```



Mode d'adressage **indexé**

- Utilité : permet d'accéder à des tables de valeurs.
- Principe : on utilise un registre d'index, préalablement initialisé, qui pointe sur la table (équivalent à un pointeur en C).
- Symbolisé par la présence de " ,x" ou " ,y" dans le champ opérande.
- Exemple :

```
1  ldx  # $3542
2  ldaa 0,x
```

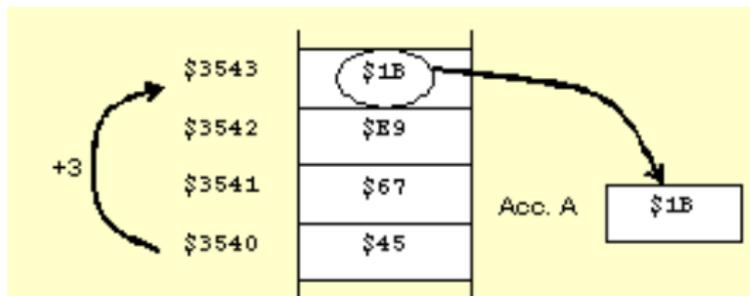


- On pourra accéder aux valeurs au dessus et en dessous en incrémentant (**inx** ou **iny**) et décrémentant (**dex** ou **dey**) l'index

Mode d'adressage **indexé**

- On peut lui ajouter un offset optionnel (valeur constante, ou l'un des 2 accumulateurs)
- Exemple :

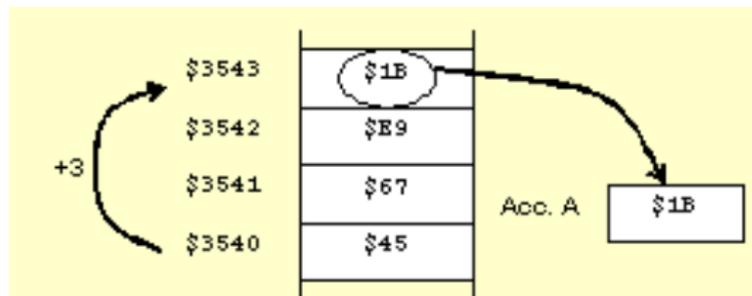
```
1  ldx  #3540
2  ldaa 3,x
```



Mode d'adressage **indexé**

- On peut lui ajouter un offset optionnel (valeur constante, ou l'un des 2 accumulateurs)
- Exemple :

```
1  ldx  # $3540  
2  ldaa 3,x
```



- On pourra utiliser ce mode avec d'autres instructions, par exemple :
 - staa 0,x : sauvegarde la valeur de A à l'adresse indiquée par X.
 - cmpb 3,y : compare la valeur de B avec la valeur se trouvant à l'adresse indiquée par Y, plus 3.
- Remarque : l'offset est limité à 255.

Mode d'adressage indexé : application

- Accès à une table de 10 valeurs, située à l'adresse TABLE

```
1  DEBUT  ldx  #TABLE    ; initialisation index
2  encore ldaa 0,x      ; lecture valeur
3          bsr  TRAITEMENT ; appel sous prog.
4          inx          ; incrémentation X
5          cpx  #TABLE+10 ; est-on arrivé au bout ?
6          bne  encore   ; si non, on continue
7          ...           ; la suite ...
```

Mode d'adressage indexé : application

- Accès à une table de 10 valeurs, située à l'adresse TABLE

```
1  DEBUT  ldx  #TABLE    ; initialisation index
2  encore ldaa 0,x      ; lecture valeur
3          bsr  TRAITEMENT ; appel sous prog.
4          inx          ; incrémentation X
5          cpx  #TABLE+10 ; est-on arrivé au bout ?
6          bne  encore   ; si non, on continue
7          ...           ; la suite ...
```

- Equivalent à :

```
1  char *p;
2  for( p = &table; p < table+10; p++ )
3      traitement( *p );
```

(Rappel : en C le type char correspond à un octet.)

- Lors de la lecture d'une table, on pourra supprimer l'instruction d'incrément / décrémentation du registre d'index (X ou Y)
- Exemples :
 - `ldaa 0,x+` (post-incrémentation)
 - `stab 5,+x` (pré-incrémentation)
 - `ldaa 50,x-` (post-décrémentation)
 - `stab 19,-x` (pré-décrémentation)

Adressage Indirect - Indexé

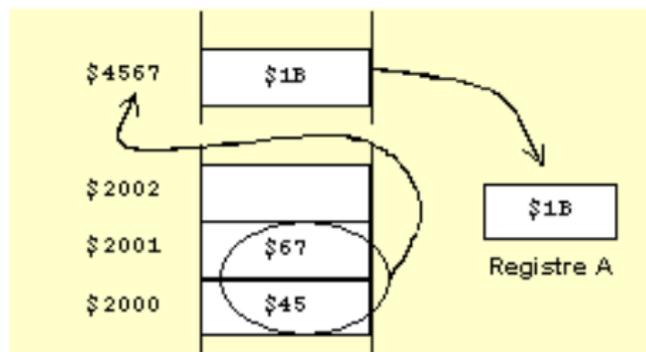
- Rarement utilisé au niveau assembleur, plus destiné aux langages évolués (compilateur).
- La valeur de l'index indique l'adresse de la valeur.
- Permet d'implémenter des pointeurs.
- Exemple :

Adressage Indirect - Indexé

- Rarement utilisé au niveau assembleur, plus destiné aux langages évolués (compilateur).
- La valeur de l'index indique l'adresse de la valeur.
- Permet d'implémenter des pointeurs.
- Exemple :

```
1   ldx  # $2000
2   ldaa [0,x]
```

⇒ Charge A avec la valeur qui se trouve à l'adresse qui se trouve en \$2000.



Instruction MOVE

- Copie d'une valeur (octet ou mot) d'un endroit à un autre.
- Deux modes d'adressages à considérer : source et destination.
- Deux instructions : `movb` (*Byte*) - `movw` (*Word*).
- Syntaxe : `movb src,dst`

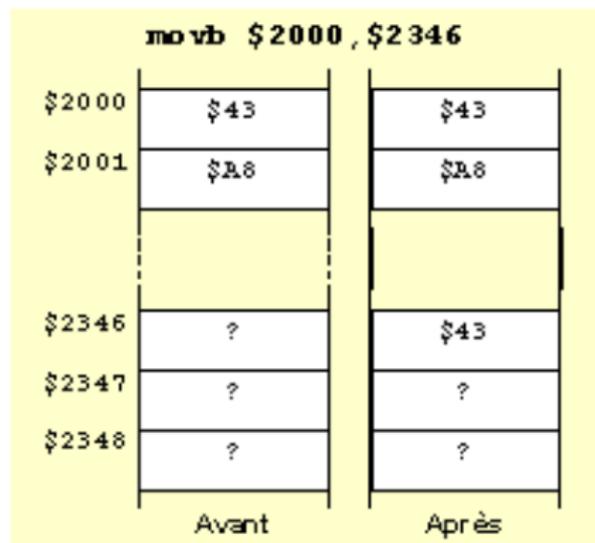
Instruction MOVE

- Copie d'une valeur (octet ou mot) d'un endroit à un autre.
- Deux modes d'adressages à considérer : source et destination.
- Deux instructions : `movb` (*Byte*) - `movw` (*Word*).
- Syntaxe : `movb src,dst`
- Exemples :

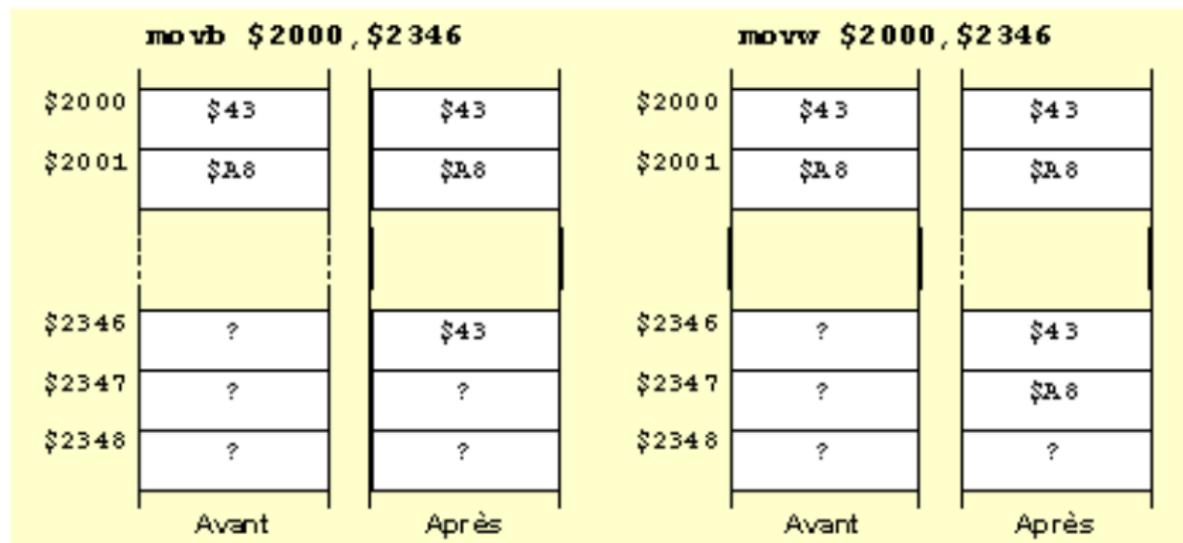
1	<code>movb #12,\$3456</code>	<code>movb #symb1,symb2</code>
2	<code>movw 12,\$3456</code>	<code>movw symb1,symb2</code>
3	<code>movb 0,x,\$3456</code>	<code>movb 0,x,symb2</code>
4	<code>movw \$3456,0,x</code>	<code>movw symb1,0,x</code>

- Remarque : le 2ème terme ne peut pas être en immédiat !
`movb 12,#3456` : Interdit !

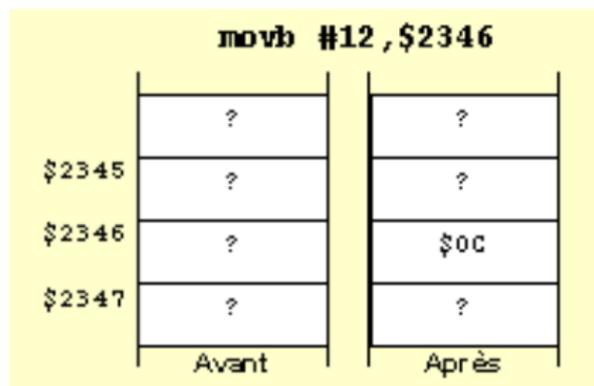
Instruction MOVE : exemples



Instruction MOVE : exemples

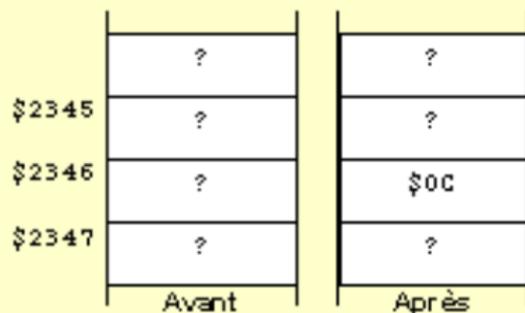


Instruction MOVE : exemples

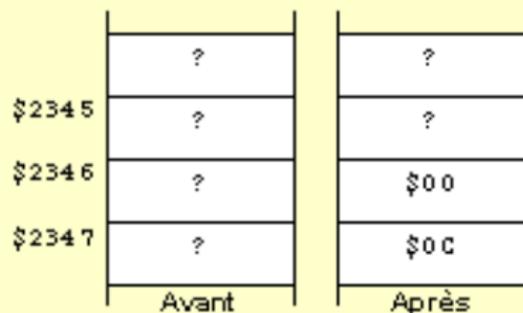


Instruction MOVE : exemples

movb #12,\$2346



movw #12,\$2346



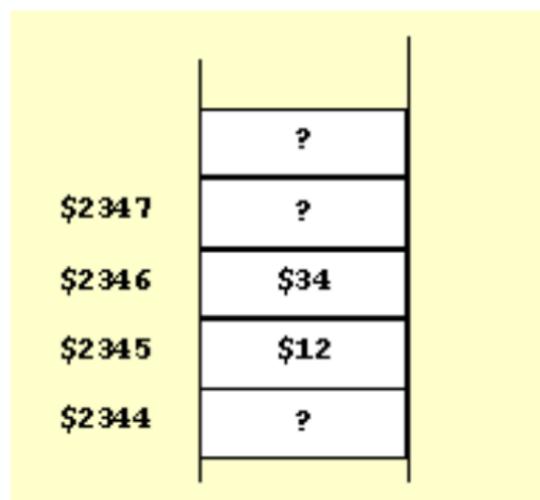
Remarque : taille de la donnée

- La donnée peut-être un octet ou un mot 16 bits.
- MAIS : la mémoire est organisée en octets !
- Deux conventions de stockage existent :
 - Big-Endian : poids **fort** à l'adresse la plus petite,
 - Little-Endian : poids **faible** à l'adresse la plus petite.

Remarque : taille de la donnée

- La donnée peut-être un octet ou un mot 16 bits.
- MAIS : la mémoire est organisée en octets !
- Deux conventions de stockage existent :
 - Big-Endian : poids **fort** à l'adresse la plus petite,
 - Little-Endian : poids **faible** à l'adresse la plus petite.

- Exemple en "Big Endian" :
Mémorisation de la valeur \$1234
(= 4660) à l'adresse \$2345 :



Manipulation de mots 16 bits

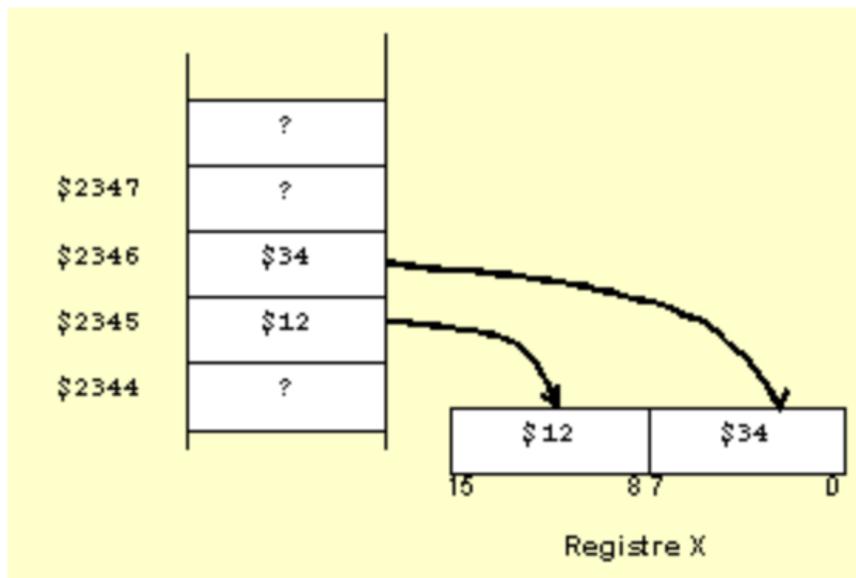
- Processeurs Freescale : **Big Endian**.
- Processeurs Intel : **Little Endian**.

Manipulation de mots 16 bits

- Processeurs Freescale : **Big Endian**.
- Processeurs Intel : **Little Endian**.
- Exemple (Freescale 9s12) : 1dx \$2345 \Rightarrow déplacement de 2 octets

Manipulation de mots 16 bits

- Processeurs Freescale : **Big Endian**.
- Processeurs Intel : **Little Endian**.
- Exemple (Freescale 9s12) : `ldx $2345` \Rightarrow déplacement de 2 octets



Exercices : instruction MOVE

Compléter le contenu de la mémoire après chaque instruction :

`movb $2001, $2004`

`movb #$1F, $2000`

`movw #$BD, $2000`

\$2000	\$43			
\$2001	\$A8			
\$2002	\$C9			
\$2003	\$0E			
\$2004	\$5A			
\$2005	\$88			

Exercice : chargement de registres

- A la suite des trois instructions précédentes, on exécute les lignes suivantes. Compléter le tableau.

Instruction	Registre modifié	Valeur (hexa)	Mode d'adressage
ldaa \$2001			
ldx \$2002			
ldy #\$2004			