Introduction à l'assembleur ARM:
utilisation de variables

```
bool arm = true;
```

GIF-1001 Ordinateurs: Structure et Applications
Jean-François Lalonde

Revenons à notre exemple d'addition

```
MOV R0, #0x1000   ; Adresse de la première valeur
LDR R1, [R0]      ; Lecture de la première valeur dans R0
ADD R0, R0, #4    ; Adresse de la deuxième valeur
LDR R2, [R0]      ; Lecture de la deuxième valeur dans  R1
ADD R1, R1, R2    ; R1 = R1 + R2
ADD R0, R0, #4    ; Adresse du résultat
STR R1, [R0]      ; Écriture du résultat en mémoire
```

- Pas pratique:
  - d'avoir à connaître l'adresse des valeurs et du résultat
  - d'avoir à utiliser deux instructions (MOV puis LDR) pour les charger
- Solution?
  - L'assembleur nous permet de **donner un nom** à des adresses mémoires: ce sont les variables!

Variables assignées

- Déclarer une variable assignée (initialisée)

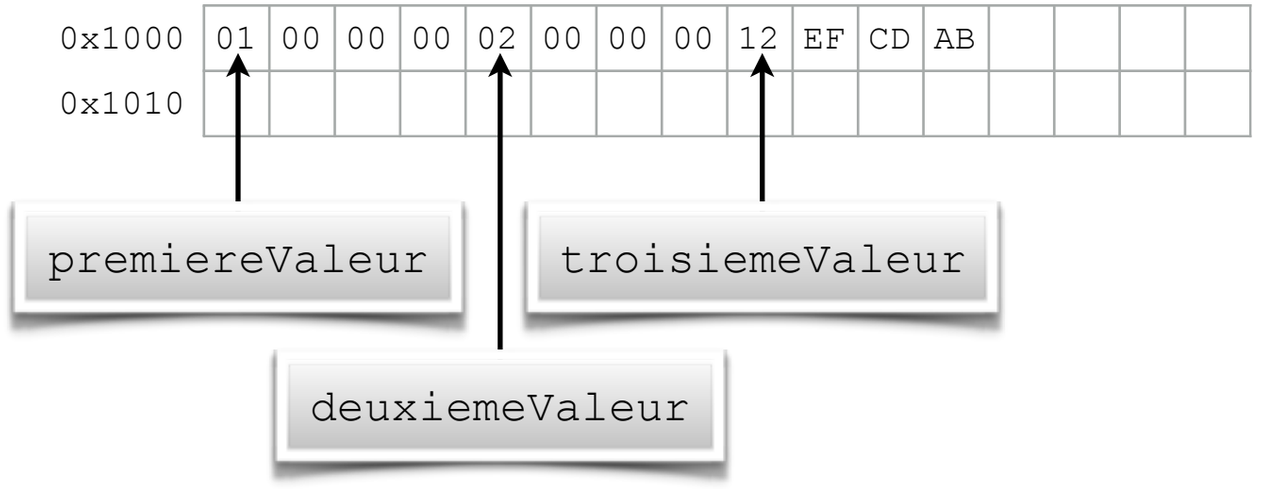
```
nom ASSIGNxx valeur
```

  - `nom`: le nom de la variable
  - `ASSIGNxx`: `xx` indique la taille en bits. Par exemple:
    - `ASSIGN8`: variable de 8 bits
    - `ASSIGN32`: variable de 32 bits
- `valeur`: la valeur de la variable
- exemple:

```
premiereValeur ASSIGN32 0x1
deuxiemeValeur ASSIGN32 0x2
```

Variables assignées

```
premiereValeur ASSIGN32 0x1
deuxiemeValeur ASSIGN32 0x2
troisiemeValeur ASSIGN32 0xABCDEF12
```



Accéder aux variables

- Charger la **valeur** d'une variable
  - « Variante » de LDR: `LDR Rd, nomDeLaVariable`

```
; Charger la valeur de premiereValeur dans R1
LDR R1, premiereValeur
```

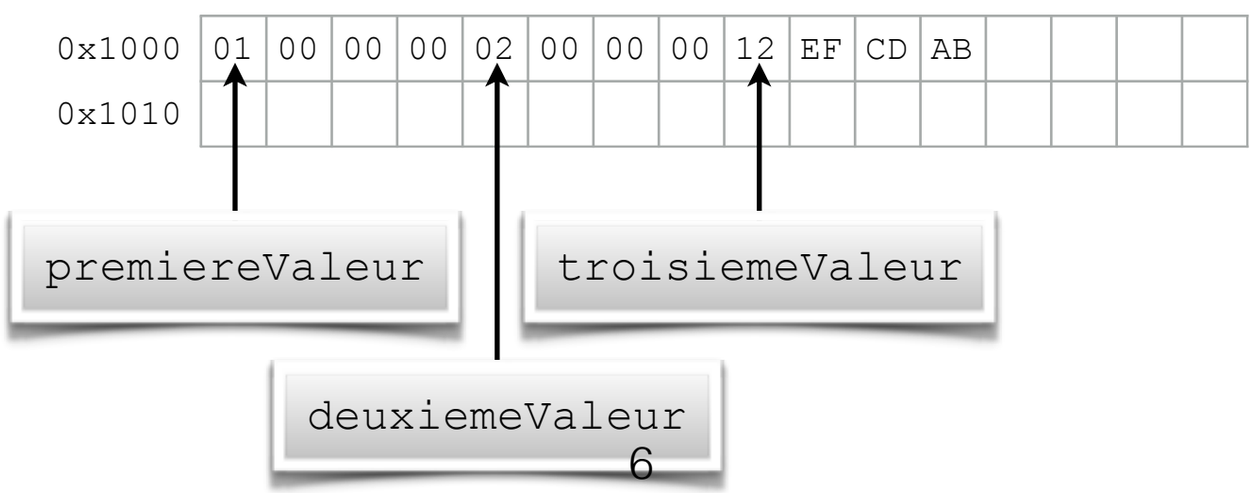
Accéder aux variables

- Charger la **valeur** d'une variable
  - « Variante » de LDR: `LDR Rd, nomDeLaVariable`

```
; Charger la valeur de premiereValeur dans R1
LDR R1, premiereValeur
```

Après l'exécution de cette instruction, R1 contiendra la valeur **0x1**.

```
premiereValeur ASSIGN32 0x1
deuxiemeValeur ASSIGN32 0x2
troisiemeValeur ASSIGN32 0xABCDEF12
```



Accéder aux variables

- Charger **l'adresse** d'une variable
  - « Variante » de LDR: `LDR Rd, =nomDeLaVariable`

```
; Charger l'adresse de premiereValeur dans R1
LDR R1, =premiereValeur
```

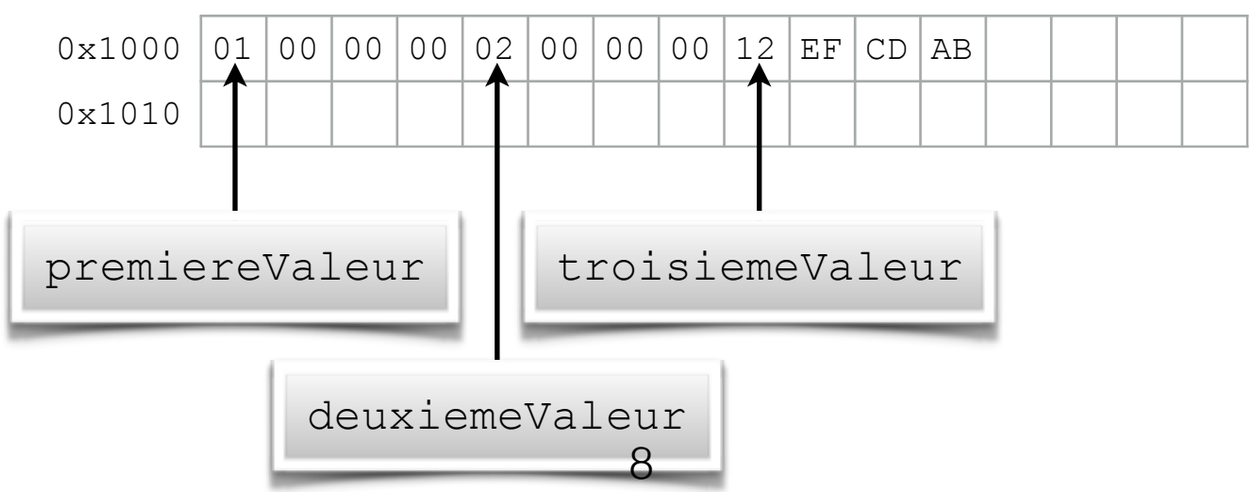
Accéder aux variables

- Charger **l'adresse** d'une variable
  - « Variante » de LDR: `LDR Rd, =nomDeLaVariable`

```
; Charger l'adresse de premiereValeur dans R1
LDR R1, =premiereValeur
```

Après l'exécution de cette instruction, R1 contiendra la valeur **0x1000**.

```
premiereValeur ASSIGN32 0x1
deuxiemeValeur ASSIGN32 0x2
troisiemeValeur ASSIGN32 0xABCDEF12
```



Variables allouées

- Allouer une variable (réserve de l'espace mais ne lui assigne pas de valeur):

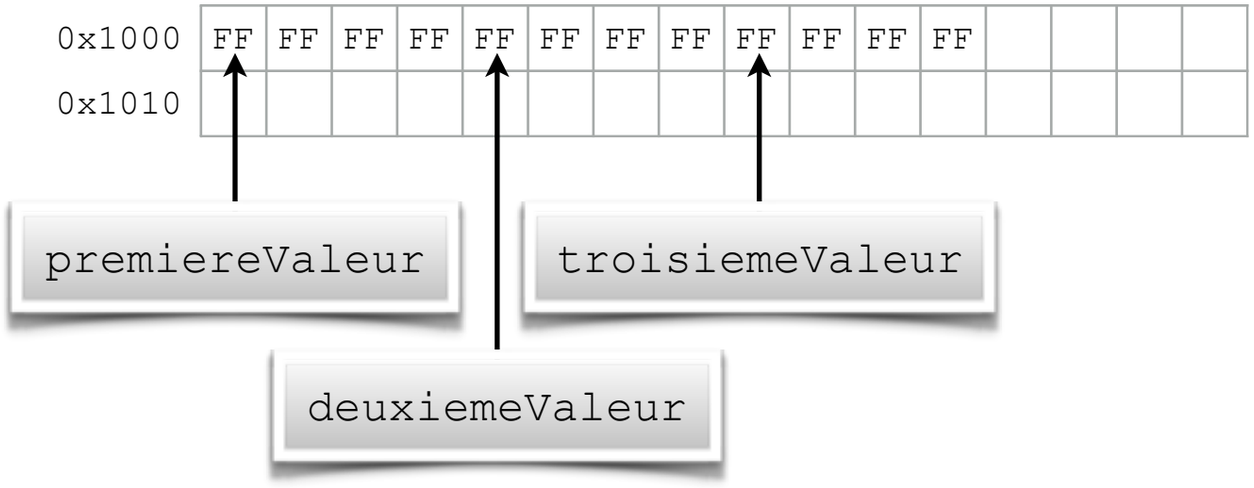
```
nom ALLOCxx nombre
```

  - `nom`: nom de la variable
  - `ALLOCxx`: `xx` indique la taille en bits. Par exemple:
    - `ALLOC8`: variable de 8 bits
    - `ALLOC32`: variable de 32 bits
- `nombre`: le nombre d'éléments à réserver
- exemple:

```
; Résultat (on ignore sa valeur a priori)
resultat ALLOC32  1
```

Variables allouées

```
premiereValeur ALLOC32 1
deuxiemeValeur ALLOC32 1
troisiemeValeur ALLOC32 1
```



Assignation vs allocation

- `nom ASSIGNxx valeur`
  - assigne une *valeur* à un espace mémoire
  - parallèle avec le standard du C:
    - c'est une *définition* (on définit une valeur précise).
- `nom ALLOCxx nombre`
  - alloue un *nombre* d'espaces mémoire *sans attribuer de valeur*.
  - parallèle avec le standard du C:
    - c'est une *déclaration* (on déclare une variable sans définir de valeur).

Tableaux

- Une variable peut indiquer *plusieurs* éléments

```
chaine ASSIGN8 0x48, 0x65, 0x6C, 0x6C, 0x6F, 0x00
tableau ASSIGN32 0x12, 0x23
tableauAlloue ALLOC32 3
```

- Ici:
  - `chaine` désigne un tableau assigné de 6 octets
  - `tableau` désigne un tableau assigné de (2 x 4 = 8) octets
  - `tableauAlloue` désigne un tableau alloué de (3 x 4 = 12) octets
- Le nom du tableau « pointe » vers le *premier* octet de ce tableau.

Tableaux

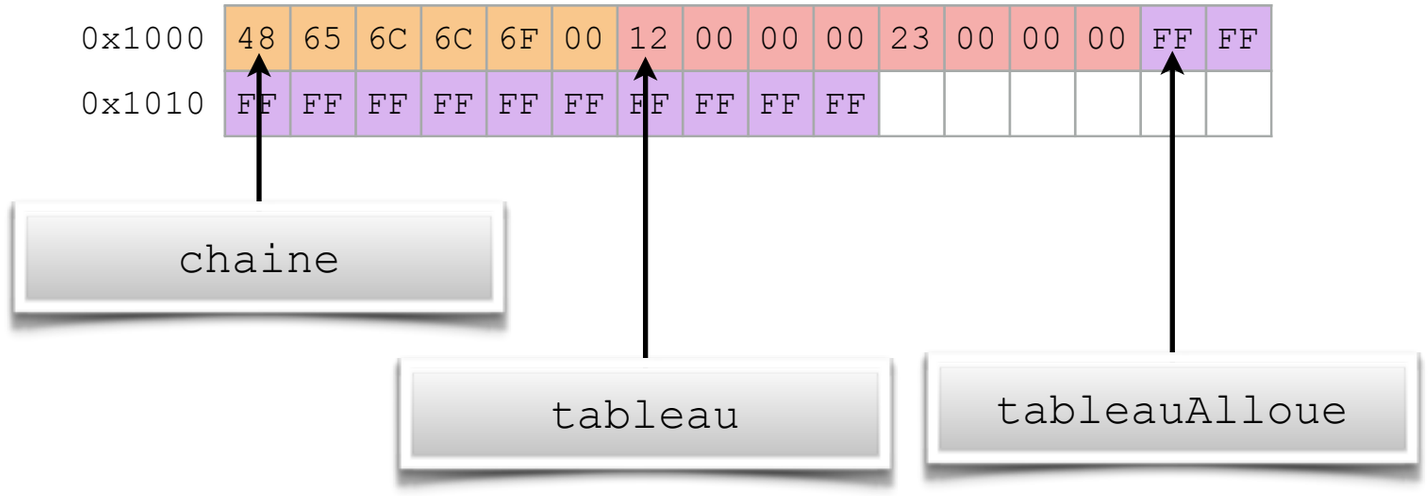
Attention! Chaque élément du tableau `chaine` possède **8 bits** car on a utilisé `ASSIGN8`.

```
chaine ASSIGN8 0x48, 0x65, 0x6C, 0x6C, 0x6F, 0x00
tableau ASSIGN32 0x12, 0x23
tableauAlloue ALLOC32 3
```

0x1000	48	65	6C	6C	6F	00	12	00	00	00	23	00	00	00	FF	FF
0x1010	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF						

Tableaux

```
chaine ASSIGN8 0x48, 0x65, 0x6C, 0x6C, 0x6F, 0x00
tableau ASSIGN32 0x12, 0x23
tableauAlloue ALLOC32 3
```



Exemple d'addition (avec variables)

- Comment représenter notre programme d'addition en utilisant des variables?

```
; Valeurs stockées en mémoire
premiereValeur ASSIGN32 0x1
deuxiemeValeur ASSIGN32 0x2
```

```
; Résultat (on ne connait pas sa valeur a priori)
resultat ALLOC32  1
```

```
LDR R1, premiereValeur  ; charger la valeur de premiereValeur
LDR R2, deuxiemeValeur  ; charger la valeur de deuxiemeValeur
ADD R1, R1, R2          ; R1 = R1 + R2
STR R1, resultat        ; stocker R1 dans resultat
```

Démonstration

(addition, avec variables)

Démonstration

(tableaux)