

aoÃ»t 24, 18 3:09

constantes_32_bits.latin.txt

Page 1/2

Chargement de constantes 32 bits dans un registre.

L'opérande droit de type immédiat des instructions de calcul et la partie droite d'une adresse des instructions load et store du ARM est de taille limitée (8 bits pour les instructions de calcul).

```
mov r1, #0x87654321 @ interdit : 0x98764321 pas codable sur 8 bits.
```

Première méthode : assembler l'entier par tranches de 8 bits :
c'est long !

```
mov r1, #0x87
mov r1, r1, lsl #8 @ décalage à gauche de 8 bits
or r1, r1, #0x65 @ or : ou bit à bit
mov r1, r1, lsl #8
or r1, r1, #0x43
mov r1, r1, lsl #8
or r1, r1, #0x21
```

Deuxième méthode : mettre la constante dans un .word et lire le contenu de ce mot avec ldr.

```
/* exemple : charger 0x87654321 dans r1 et 0xabcdef23 dans r2 */
```

```
.text
ldr r1, [pc, #DELTA1]
ldr r2, [pc, #DELTA2]
... suite des instructions ...

.data
.word 0x87654321
.word 0xabcdef23
```

$\begin{array}{cc} \text{DELTA1} & \text{DELTA2} \\ | & | \\ \cdot & \cdot \\ | & | \\ \cdot & \cdot \\ | & | \\ \text{v} & \text{v} \end{array}$

Problème : on ne maîtrise pas les adresses de text et data donc rien ne garantit que DELTA1 et DELTA2 soient codables sur 8 bits

Solution : caser le .word dans la section text, près de l'instruction ldr :
on est sur que DELTA1 et DELTA2 seront < 256.

On écrit ceci :

```
.text
ldr r1,= 0x87654321
ldr r2,= 0xabcdef23
... suite des instructions ...
```

```
.ltorg @ ltorg case les constantes des ldr reg,=
@ dans la section .text
```

L'assembleur le traduit automatiquement en :

```
ici1: ldr r1, cte1
ici2: ldr r2, cte2
... suite des instructions ...
```

```
cte1: .word 0x87654321 @ ltorg est expansé en une suite de .word
cte2: .word 0xabcdef23
```

et les ldr reg, adresse

aoÃ»t 24, 18 3:09

constantes_32_bits.latin.txt

Page 2/2

sont à leur tour traduits en :

```
ici1: ldr r1, [pc, #(cte1 - ici1 - 8)]
ici2: ldr r2, [pc, #(cte2 - ici2 - 8)]
```

Lors du calcul de l'adresse, pc contient l'adresse de l'instruction ldr (ici1) + 8 (pc en avance de 2 instructions lorsqu'on l'utilise comme opérande).
D'où DELTA1 = cte1 - ici1 - 8 et DELTA2 = cte2 - ici2 - 8