L'ASSEMBLEUR – PARTIE 1
Gestion des variables globales et affectation

Xavier Merrheim

Programmation de haut niveau

- Aujourd'hui la plupart des programmeurs programment dans des langages de haut niveau comme le langage C, le java ou le C++
- Le processeur utilise un langage très rudimentaire appelé assembleur
- Il faut traduire les programmes dans un langage de haut niveau en assembleur pour que le processeur puisse les exécuter.

Compilation

- Le compilateur va traduire un programme source en assembeur puis en programme exécutable.
- La compilation est une science extrêmement complexe.

De nombreux assembleurs

- Il existe autant d'assembleurs que de familles de processeurs : x86, arm, 68 000 ...
- Le langage assembleur est très complexe avec de nombreux détails extrêmement techniques.
- Il est difficile de programmer en assembleur

Pour étudier l'assembleur

- Il ne s'agit pas de faire de vous des programmeurs assemblleurs opérationnels.
- L'objectif est une bonne culture générale permettant de mieux comprendre l'informatique
- Nous allons notamment apprendre à traduire des programmes en C en assembleur ARM

Architecture cible

- Nous allons étudier l'assembleur ARM 32 bits
- Le processeur contient 16 registres de 32 bits notés r0,r1, r2, ..., r15
- A coté du processeur se trouve un RAM mémorisant des cases mémoires de 32 bits.
- Chaque case mémoire porte un numéro appelé adresse

Exemple 1

```
int a,b,c;
main()
{
a=10;
b=20;
c=a+b;
}
```

Un programme bien étrange

- Variables globales : cet exemple en comporte car il est plus facile de gérer des variables globales que locales.

- Le main n'est pas une fonction : c'est volontaire car les fonctions sont complexes à traduire
- Par d'entrée/sortie car les entrées/sorties sont des appels de fonctions.
- Ne faites pas cela en cours de C !!!

Traduction de l'exemple 1

```
L1:
.word a
.word b
.word c
.comm a,4,4
.comm b, 4,4
.comm c, 4,4
```

Traduction de l'exemple 1

```
mov r0,#10

ldr r1,L1
str r0,[r1]
mov r0,#20

ldr r1,L1+4
str r0,[r1]
ldr r0,L1
ldr r1,[r0]
ldr r0,L1+4
ldr r2,[r0]
add r1,r1,r2
ldr r0,L1+8
str r1,[r0]
```

mov

- mov r0,#10 : mets la constante 10 dans le registre r0 du processeur.

- Autre utilisation : mov r0,r1 : copie le contenu du registre r1 dans r0

ldr

- ldr r1,L1 : met dans r1 le contenu de la case mémoire L1 (lecture de la RAM)
- On peut ajouter une contante à l'adresse L1+4 ou L1+8
- ldr r1,[r0] met dans r1 le contenu de la case mémoire r0

Qui choisit la valeur de L1 ?

- Le système d'exploitation va charger le programme qui est sur le disque dur dans la RAM.

- A ce moment, il choisit la valeur de L1 et calcule toutes les adresses comme L1+4

str

- str r0,[r1] copie r0 dans la case mémoire numéro r1
- Il s'agit d'une écriture de la ram

Add et sub

- add r1,r2,r3 met dans r1 la somme de r2 et de r3
- Sub r1,r2,r3 : met dans r1 la valeur de r2-r3
- On peut écrire
  add r1,r1,#4
  sub r4,r5,#89

Différents modes d'adressage

- Par registre : r2
- Constante : #10
- Par adresse : L1
- Indirect [r0]
- Il y en a d'autres !

Exercice

Traduire en assembleur ARM le programme C

```
int a,b,c,d;
main()
{
a=10;b=65;
c=b-a+12;
d=c+b-98;
d=d+a-87;
}
```

Solution

```
L1:
.word a
.word b
.word c
.word d
.comm a,4,4
.comm b, 4,4
.comm c, 4,4
.comm d,4,4
```

```
mov r0,#10
ldr r1,L1
str r0,[r1]
mov r0,#65

ldr r1,L1+4
str r0,[r1]
ldr r0,L1+4
ldr r1,[r0]
ldr r0,L1
ldr r2,[r0]
sub r1,r1,r2
add r1,r1,#12
ldr r0,L1+8

str r1,[r0]
```

```
ldr r0,L1+8
ldr r1,[r0]
ldr r0,L1+4

ldr r2,[r0]
add r1,r1,r2
sub r1,r1,#98

ldr r0,L1+12
str r1,[r0]
ldr r0,L1+12
ldr r1,[r0]
ldr r2,L1
ldr r3,[r2]
add r1,r1,r3
sub r1,r1,#87
str r1,[r0]
```

QUESTIONNAIRE

Question 1

- Définir les directives assembleur ARM permmettant de définir les variables globales int u,v,x,y;

Question 2

- Dans les directives précédentes :
  à quelle adresse se trouve l'adresse de u, celle de v, celle de x et celle de y

Question 3

- Citez 2 utilisations de l'instruction assembleur mov en donnant 2 exemples détaillés.

Question 4

- Citez 2 utilisations de l'instruction assembleur ldr en donnant 2 exemples détaillés.

Question 5

- Citez une utilisation de l'instruction assembleur str en donnant un exemple détaillé.

Question 6

- L'instruction assembleur ldr permet-elle de lire ou d'écrire dans la RAM ?

Question 7

- L'instruction assembleur str permet-elle de lire ou d'écrire dans la RAM ?

Question 8

- Donnez 2 exemples d'utilisation de l'instruction assembleur sub.

Question 9

- Traduire en assembleur ARM le programme C suivant :

```
int u,v,w;
main()
{u=9;v=90;w=67;
w=w-u+v-99;
}
```

Question 10

- Traduire en assembleur ARM le programme C suivant :

```
int a,b,c,d,e,f;
main()
{a=1;b=2;c=90;d=80;e=6;
f=a+b+c+d+e
}
```

QUESTIONNAIRE CORRECTION

Question 1

- Définir les directives assembleur ARM permmettant de définir les variables globales int u,v,x,y;

```
L1:
.word u
.word v
.word x
.word y
.comm a,4,4
.comm b,4,4
.comm b,4,4
.comm b,4,4
```

Question 2

- Dans les directives précédentes :
  à quelle adresse se trouve l'adresse de u, celle de v, celle de x et celle de y

```
L1:
.word u
.word v
.word x
.word y
.comm a,4,4
.comm b,4,4
.comm b,4,4
.comm b,4,4
```

L'adresse de u se trouve à l'adresse L1
L'adresse de v se trouve à l'adresse L1+4
L'adresse de x se trouve à l'adresse L1+8
L'adresse de y se trouve à l'adresse L1+12

Question 3

- Citez 2 utilisations de l'instruction assembleur mov en donnant 2 exemples détaillés.

  mov r6,#45 ==> mets la constante 45 dans le registre r6

  mov r4,r2 ==> copie le contenu du registre r2 dans r4

Question 4

- Citez 2 utilisations de l'instruction assembleur ldr en donnant 2 exemples détaillés.

  ldr r2,L1+4 ==>mets dans r2 le contenu de la case mémoire L1+4

  ldr r4,[r6] ==>mets dans r6 le contenu de la case mémoire n°r6

Question 5

- Citez une utilisation de l'instruction assembleur str en donnant un exemple détaillé.

  str r2,[r0] écris dans la case mémoire n° r0 le contenu de r2

Question 6

- L'instruction assembleur ldr permet-elle de lire ou d'écrire dans la RAM ?
  C'est une lecture de la RAM

Question 7

- L'instruction assembleur str permet-elle de lire ou d'écrire dans la RAM ?
  C'est une écriture de la RAM

Question 8

- Donnez 2 exemples d'utilisation de l'instruction assembleur sub.

  sub r0,r2,r5 mets dans r0 le contenu de r2-r5

  sub r0,r6,#4 mets dans r0 le contenu de r6-4

Question 9

- Traduire en assembleur ARM le programme C suivant :

```
int u,v,w;
main()
{u=9;v=90;w=67;
w=w-u+v-99;
}
L1:
.word u
.word v
.word w
.comm u,4,4
.comm v,4,4
.comm w,4,4
mov r0,#9
ldr r1,L1
str r0,[r1]
mov r0,#90
ldr r1,L1+4
str r0,[r1]
mov r0,#67
ldr r1,L1+8
str r0,[r1]
ldr r0,L1+8
ldr r1, [r0]
ldr r2,L1
ldr r3,[r2]
sub r1,r1,r3
ldr r2,L1+4
ldr r3,[r2]
add r1,r1,r3
sub r1,r1,#99
str r1,[r0]
```

Question 10

- Traduire en assembleur ARM le programme C suivant :

```
int a,b,c,d,e,f;
main()
{a=1;b=2;c=90;d=80;e=6;
f=a+b+c+d+e
}
```

```
L1:
.word a
.word b
.word c
.word d
.word e
.word f
.comm a,4,4
.comm b,4,4
.comm c,4,4
.comm d,4,4
.comm e,4,4
.comm f,4,4
```

```
mov r0,#1
ldr r1,L1
str r0,[r1]
mov r0,#2
ldr r1,L1+4
str r0,[r1]
mov r0,#90
ldr r1,L1+8
str r0,[r1]
mov r0,#80
ldr r1,L1+12
str r0,[r1]
mov r0,#6
ldr r1,L1+16
str r0,[r1]
```

```
ldr r0,L1
ldr r1, [r0]
ldr r0,L1+4
ldr r2, [r0]
add r1,r1,r2
ldr r0,L1+8
ldr r2, [r0]
add r1,r1,r2
ldr r0,L1+12
ldr r2, [r0]
add r1,r1,r2
ldr r0,L1+16
ldr r2, [r0]
add r1,r1,r2
ldr r0,L1+20
str r1,[r0]
```