EL KANBI Asmae
HANI Pamella

Compte-rendu ALM

I. Questions d'analyse d'un programme assembleur généré par un compilateur

0) Ce programme ne donnera pas toujours les bonnes valeurs de Un. Lorsqu'on donne U0 = 2147483647, la valeur maximale prise par un entier en C, le programme ne fonctionne plus.

1) $@ correspond à la première dépendance, et $< à la première dépendance.

2) Lorsqu'on enlève **syracuse** du **all**, cela supprime syracuse de la compilation du programme. En effet, le fichier crée toujours le fichier syracuse.s ,mais dans la règle syracuse du Makefile, ce dernier ne reconnaît plus l'exécutable syracuse. En effet si on exécute la commande ./syracuse, on obtient le message d'erreur suivant:
`no such file or directory: ./syracuse`

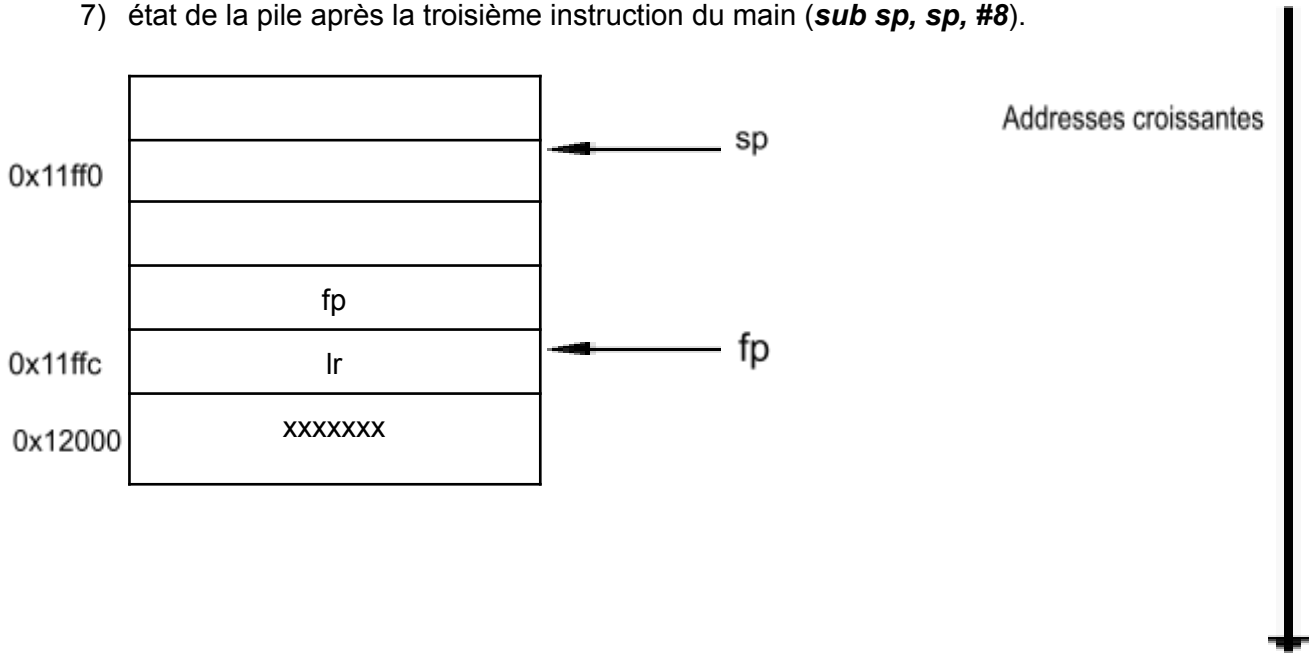
2) La variable globale de ce programme se nomme "x".

3) La variable globale x se trouve à l'étiquette .L8, et qui est ensuite stockée dans le registre r3, dans le main. La variable globale x est stockée à 0x100c0.

   L'étiquette .L8 se trouve à la fin du fichier, ligne 99.
   .comm x, 4, 4 va allouer 4 octets en mémoire dans la variable x, à une adresse multiple de 4.

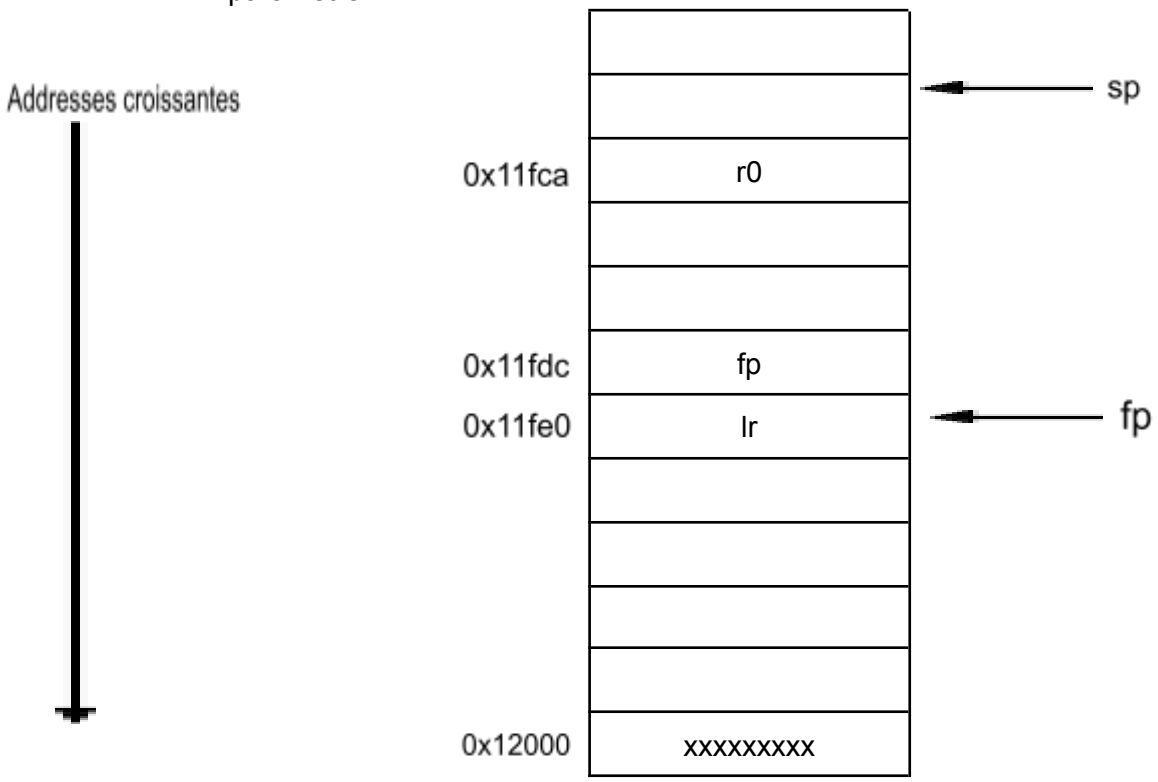
4) L'adresse de la variable globale x est stockée à l'adresse 0x100b4. on trouve cela en affichant toutes les adresses à partir de l'étiquette main avec la commande x /20xw 0x100b4 (adresse de l'étiquette main).
   Les valeurs des adresses en hexadécimal ou est stockée cette adresse sont 0xe50b0008, 0xeaffffffe, 0x000100c0, 0x0000000d.

5) Le début de la pile se situe à l'adresse 0x12000.
6) On stocke dans la pile, les registres fp et lr afin de revenir à l'instruction qui était juste après l'appel de la fonction syracuse, dans le main.
7) état de la pile après la troisième instruction du main (***sub sp, sp, #8***).



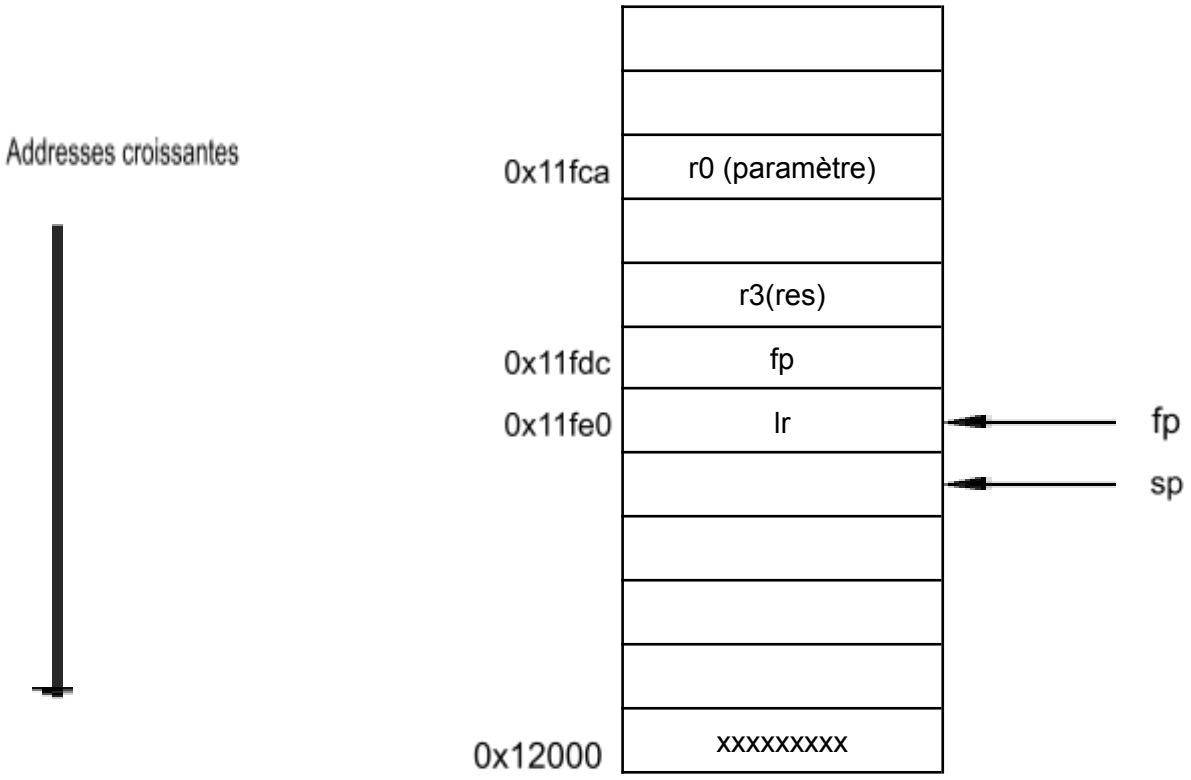
8) valeur de fp: 0x11ffc
   valeur de sp: 0x11ff0
9) Dans la fonction main, le paramètre de syracuse se situe dans r0.
10) str r0, [fp, #-16] permet de stocker la valeur du paramètre r0 dans sp.
    ldr r3, [fp, #-16] permet de lire la valeur présente à l'adresse de sp dans r3.

11) dessin de la pile en précisant l'emplacement (par rapport à ***fp***) de la sauvegarde du paramètre:



12) La variable locale ***res*** est stockée dans r3 dans syracuse. L'adresse de la variable locale res est à fp-8.

    Dessin de la pile montrant où est stockée la variable res:



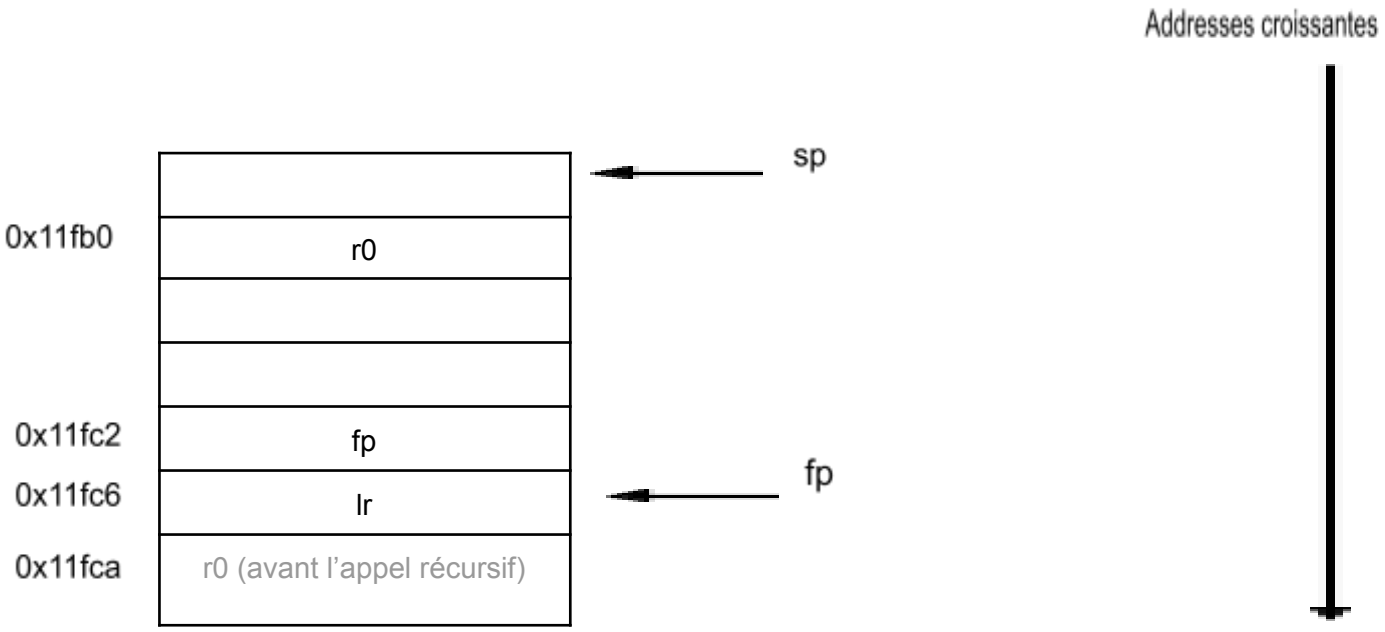
13) La valeur de retour de la fonction syracuse (le résultat) est stockée dans la pile.
14)

```
sub     sp, fp, #4
 @ sp needed
pop {fp, lr}
bx  lr
```

Les 3 dernières lignes de la fonction syracuse :
On place sp au-dessus de fp, on, puis on sort de la fonction (on met lr dans pc).

15) Au moment de l'instruction (bx lr), X=0x10070
16) L'instruction qui se trouve en mémoire à l'adresse X est "mov r0,r3".
17) La variable y du main est stockée dans le registre r0.
Adresse de la variable y: 0x100b4 (fp-8)
L'instruction qui effectue l'affectation y=syracuse(x) est "str r0, [fp, #-8]".

18) Une fois le retour de la fonction main est effectué, la valeur de pc est 0x100b8.
19) Dans le cas où le programme est exécuté sur le processeur de la machine, le registre pc ne contient rien car le programme contient une boucle infinie et l'adresse de la prochaine instruction n'existe pas.
20) Le registre lr contient l'adresse de retour de la fonction main. Sa valeur en hexadécimal est 0x100b4.
21)
Etat de la pile après le premier appel récursif de syracuse (après l'instruction ***sub sp, sp, #16***):



22) C'est une bonne pratique d'écrire ***if (0 == (a & 0x1))*** au lieu de ***if ((a & 0x1) == 0)*** car dans le cas où l'utilisateur fait une faute de frappe et écrit ***if (0 = (a & 0x1))*** le compilateur affichera immédiatement un message d'erreur .

Si on écrivait ***if ((a & 0x1) = 0)***, le programme ne s'exécute pas et affiche le message d'erreur suivant:

```
syracuseprint.c:19:16: error: lvalue required as left operand
of assignment
   19 |   if ((a & 0x1)=0)
```

En effet, le programme ne peut pas déduire si le côté gauche de l'instruction possède une adresse ou non.

23) Pour l'opération ***a & 0x1***, le programme prend ces 2 nombres et fait l'opération binaire AND bit à bit sur les 2 nombres. Le résultat du AND est égal à 1 si les deux nombres sont égales à 1.
Donc l'expression ***if (0 == (a & 0x1))*** évalue l'égalité obtenue entre 0 et le résultat de l'opération binaire a & 0x1. Cette expression renvoie un booléen.

Si on écrivait ***if (0 == (a && 0x1)),*** le programme renvoie une erreur de segmentation. En effet, l'opérateur & est un opérateur logique et binaire, car il fonctionne à la fois sur des données booléennes et binaires, tandis que l'opérateur && n'est qu'un opérateur logique. Il fonctionne donc uniquement sur un type de données booléen, ce qui n'est pas le cas ici.

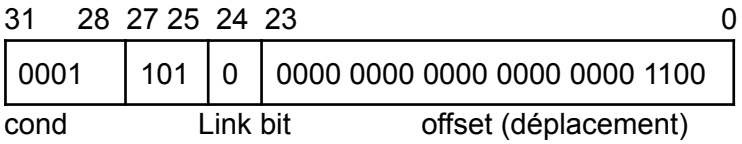
24) La valeur en hexadécimal des 4 octets codant l'instruction ***bne .L2*** est :
0x01 0x00 0x00 0x1a

Le champ **cond** contient les bits N Z C V, qui correspondent aux codes des conditions arithmétiques.

Dans cette instruction :

- N (résultat de l'opération est négatif) = 0
- Z (si tous les bits sont égaux à 0) = 0
- C (si une retenue a été produite) = 0
- V (dépassement de capacité) = 1

Codage de l'instruction ***bne .L2*** :



La valeur cond (codage de l'instruction bne) est donnée par les tableaux de la documentation ARM.

cond	Mnemonic extension	Meaning (integer)
0000	EQ	Equal
0001	NE	Not equal

La valeur offset (déplacement) correspond au décalage qu'il y a entre la valeur de pc au moment de l'instruction et l'adresse sur laquelle on souhaite aller (12 bits de décalage en l'occurrence).