

COMPTE RENDU TP ALM SOFT N°3

Anthony Geourjon - Clément Rouquier, Polytech Grenoble R1CM3 Gr.2

14/12/2015

Compilation

Note : Suite au mail que nous vous avons envoyé sur l'utilisation du compilateur `arm-elf-gcc`, vous nous aviez indiqué d'utiliser `arm-eabi-gcc`. Nous avons installé `arm-none-eabi-gcc` car c'était la seule version que nous avons trouvée. En comparant nos résultats avec d'autres étudiants, nous nous sommes aperçus que les résultats différaient quelque peu et que notre compte-rendu était donc valable seulement pour notre fichier `multrec.s`.

- 1.
2. L'algorithme de multiplication récursive est relativement simple. L'algorithme n'utilise que des additions et des soustractions ainsi que le fait que

$$x \times y = \sum_{k=1}^y x$$

L'algorithme est ici implémenté sous forme récursive, il pourrait l'être de manière itérative avec une boucle.

3. `simgcc -S -O0 multrec.c`
4. `simgcc -Wa,--gdwarf2,-L -o multrec multrec.s`
5.

```
anthony@geourjoa-desktop:~/TP_ALM_Soft/tp3$ armrun multrec
Donnez deux entiers positifs :
5
5
5 * 5 = 25
anthony@geourjoa-desktop:~/TP_ALM_Soft/tp3$
```

6. `simgdb multrec`

Exécution pas à pas

1. `(gdb) break main`
2. `(gdb) run`

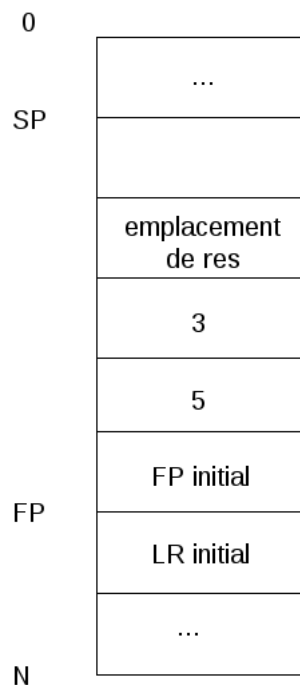
```

3. (gdb) info reg r13
r13: 0x1ffff0
(gdb) info reg r11
r11: 0x0 0
(gdb) info reg r12
r12: 0x1fffe8

```

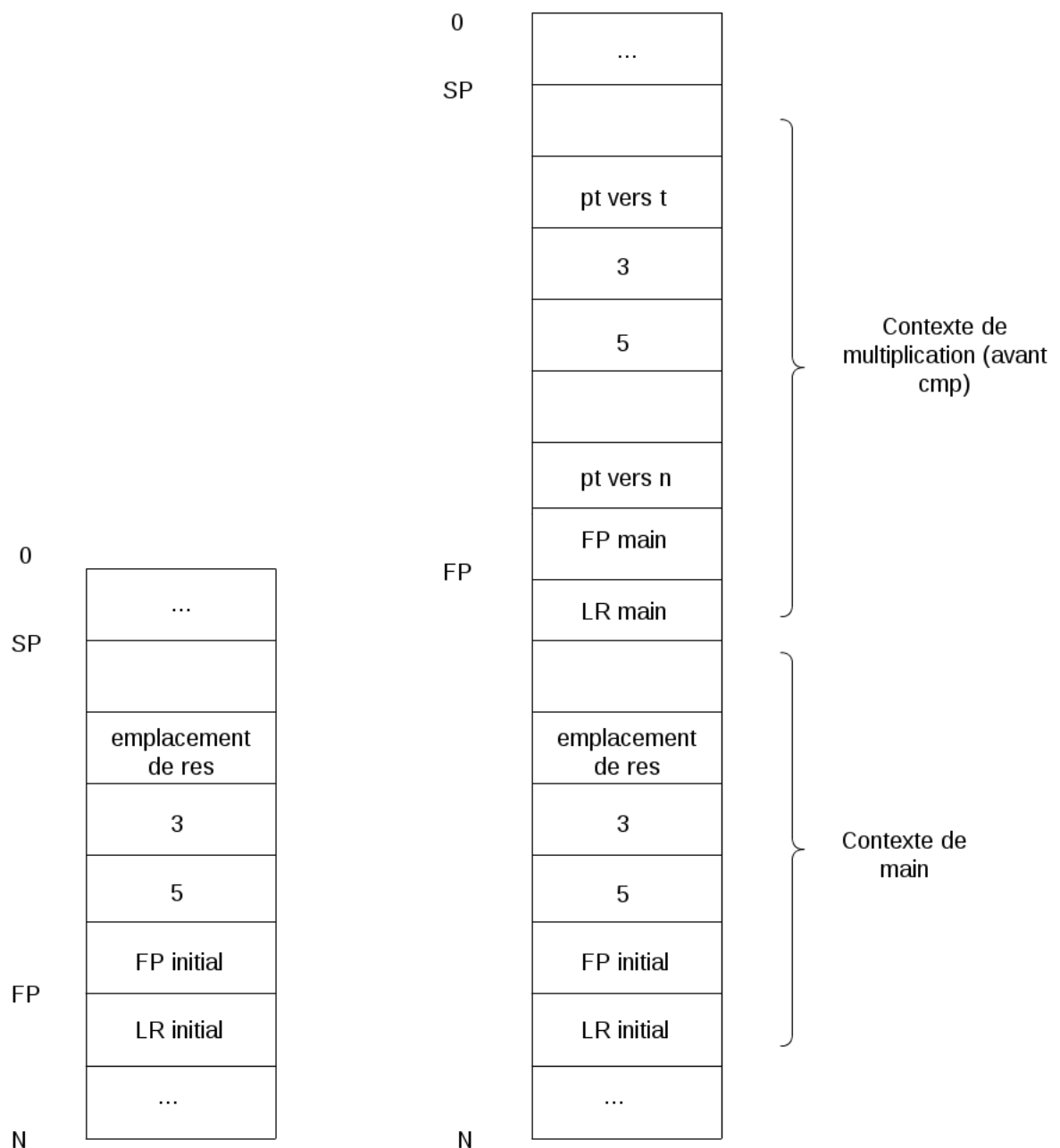
La pile est vide car le registre fp n'as pas encore été initialisé.

4. L'instruction `stmfd!, {fp,lr}` permet de sauvegarder l'adresse du contexte de la fonction précédente (le sommet de sa pile) ainsi que l'adresse de retour de la fonction (ici il n'y en a pas). Cette instruction est obligatoire si on ne veut pas compromettre l'exécution de la fonction appelante.
5. Avant l'appel l'adresse `fp - #12` contient une valeur inconnue. Après l'appel à `scanf` la case mémoire contient l'adresse vers le second résultat de `scanf`
6. Avant l'appel, on charge en mémoire des adresses dans `r1` et `r2`, puis après on recupère le résultat grâce à des `ldr`.
7. `n, t, res` sont rangés respectivement aux adresses `fp - #12`, `fp - #8`, `fp - #4`



8. Les paramètres sont passés grâce aux valeurs contenues dans `r0`, `r1` et à l'adresse contenue dans `r2`.

9.



10. f est contenue dans r2, a est initialement dans r2 et r est présent dans n
11. pc contient l'adresse de retour initialement contenue dans le registre lr. Il est noté que notre fichier d'assembleur compilé avec *simgcc* diffère légèrement du fichier *multrec.s* mais le résultat est le même. L'instruction `sub sp, fp, #4` permet de replacer le pointeur de pile au bon endroit (il avait été déplacé en au début de main) et de dépiler correctement.
12. Voir fichier *multrec.s*.