GIF-1001 Ordinateurs: Structure et Applications
Solutions: ARM — séquence d'exécution et branchements

---

1. À quoi sert le registre LR?

   Solution: Il sert à faire le lien entre une fonction qui appelle une autre fonction : le registre LR entrepose les adresses de retour (instruction BL).

2. Le registre SP est-il incrémenté ou décrémenté lors d'un PUSH? Lors d'un POP? Pourquoi varie-t-il toujours de 4?

   Solution: Habituellement, SP est décrémenté lors d'un PUSH parce que la pile est habituellement placée aux adresses supérieures de la RAM. Il varie de 4 parce que les mots et registres des microprocesseurs ARM ont 32 bits.

3. Écrivez une routine `Facto` de calcul de la factorielle d'un nombre entier en assembleur ARM. Vous routine doit accepter un paramètre en entrée dans `R0` (le nombre dont il faut calculer la factorielle) et retourner la sortie dans `R1` (la factorielle du nombre). N'oubliez pas que $0! = 1$. Vous n'avez pas à traiter le cas où une très grande valeur dans `R0` créerait un débordement.

   Solution:

   Décidons d'abord que `R0` contiendra le paramètre d'entrée et que `R1` contiendra le résultat (`R1 = R0!`). L'appel de la fonction se fera ainsi (`Res = N!`) :

   ```
     LDR R0, =N
     LDR R2, =Res
     LDR R0, [R0]
     BL Facto
     STR R1, [R2]
   ```

   La fonction pourrait être codée ainsi, avec une boucle qui multiple 1 par i à chaque itération pour i = 1 jusqu'à N :

   ```
   Facto
     PUSH {R3, R4, LR} ; Préservons l'environnement
     MOV R3, #1      ; R3 = i
     MOV R1, #1      ; R1 = i!
     CMP     R0, #0   ; Cas spécial: 0! = 1
     BEQ     FinFacto

   Boucle
   ```

```
  CMP R3, R0    ; i > N? Si oui, fin!
  BHI FinFacto  ; i > N? Si oui, fin!
  MOV R4, R1
  MUL R1, R4, R3
  ADD R3, R3, #1
  B Boucle

FinFacto
  POP {R3, R4, LR}
  BX LR
```

4. Comment fait-on pour passer des paramètres à une fonction?

Solution: Soit par registre (on assigne un registre à chaque paramètre), soit par la pile (les paramètres d'entrée sont mis sur la pile avant le BL, et les paramètres de sortie sont pris sur la pile après le retour de la fonction.

5. Qu'est-ce qu'une fonction qui préserve l'environnement?

Solution: Une fonction qui ne change pas les registres en dehors de ceux utilisés pour le passage de paramètres.

6. Réécrivez la question 3 en passant les paramètres de la fonction par la pile.

Solution: L'appel de la function se fera ainsi:

```
  LDR R0, =N
  LDR R2, =Res
  LDR R0, [R0]
  PUSH {R0} ; Le paramètre d'entrée, N, est mis sur la pile
  BL Facto
  POP {R1}   ; Le paramètre de sortie est récupéré de sur la pile
  STR R1, [R2]

Facto
  PUSH {R0, R3, R4, LR}  ; Fonction propre!
  LDR R0, [SP, #16]       ; Récupère N sur la pile
                          ; (#16 -> R0, R3, R4, LR empilés)
  MOV R3, #1              ; R3 = i
  MOV R1, #1              ; R1 = i!
  CMP     R0, #0          ; Cas spécial: 0! = 1
  BEQ     FinFacto
```

```
Boucle
  CMP R3, R0              ; i > N? Si oui, fin!
  BHI FinFacto            ; i > N? Si oui, fin!
  MOV R4, R1
  MUL R1, R4, R3
  ADD R3, R3, #1
  B Boucle

FinFacto
  STR R1, [SP, #16]        ; Met N! sur la pile
  POP {R0. R3, R4, LR}
  BX LR
```

7. Combien d'accès à la mémoire sont effectués lorsque l'on appelle une fonction? En d'autres mots, combien d'accès à la mémoire seront effectués pour lire et exécuter l'instruction `BL Mafonction`?

Solution: `BL Mafonction` requiert un seul accès mémoire, pour lire l'instruction.

8. Dites quels sont les avantages et les inconvénients de passer les paramètres d'une fonction par registre plutôt que par la pile. Dites également pourquoi passer les paramètres par variables globales est une mauvaise pratique.

Solution: Passer les paramètres par registres est beaucoup plus rapide et plus simple que par la pile (on évite des accès à la mémoire). Cependant, le nombre de registre est limité et il faudra mettre les registres sur la pile de toute façon si la fonction appelle une autre fonction avec des paramètres passés par les mêmes registres. Bref, les registres sont mieux, mais ils sont plus limités et n'offrent pas une solution universelle.

Les variables globales sont à éviter d'un point de vue maintenance et modularité. Par ailleurs, une fonction qui utilise des variables globales n'est pas ré-entrante, elle peut difficilement être récursive, les variables globales prennent de la mémoire qui pourraient être utilisée autrement...

9. Qu'est qu'un débordement de pile (stack overflow)? Que se produit-il lorsque la pile déborde?

Solution: Un débordement de pile survient lorsque le dessus de la pile dépasse la plage d'adresses réservée pour la pile (parce que plus de données sont empilées que dépilées par exemple). Lors d'un débordement de pile, le contenu de la mémoire après la pile est écrasé par les empilements: des variables changeront de valeur si la pile débordent sur des variables ou des instructions imprévues seront exécutées si la pile déborde sur des instructions.

10. Une fonction récursive (qui s'appelle elle-même) requiert souvent une grande pile. Expliquez pourquoi.

Solution: À tous les appels de fonction, plusieurs octets de piles sont utilisés pour emmagasiner l'adresse de retour, passer des paramètres, réserver de la mémoire pour les variables locales et/ou rendre la fonction propre. Cette mémoire est libérée seulement lors du retour de la fonction. Or, les fonctions récursives s'appellent avant de terminer. Il y aura donc plusieurs appels avant le premier retour de fonction qui libère de la mémoire de la pile.

11. Combien d'accès à la mémoire sont nécessaires afin de lire et d'exécuter l'instruction `PUSH {Rn}`? L'instruction PUSH existe-t-elle vraiment ou est-elle compilée comme une autre instruction?

Solution: L'instruction `PUSH {Rn}` est lue en un seul coup d'horloge et il faut un accès mémoire pour l'exécuter, c'est-à-dire pour écrire la valeur de `Rn` sur la pile. L'instruction PUSH est compilée comme un « Store Multiple » `STMFD SP!, {Rn}`.