Travaux Pratiques n°3
Architecture des ordinateurs.
—L3—

Ce sujet est à réaliser sur deux séances. Avant les deux séances du 18 Mai, vous enverrez une archive contenant les fichiers sources ainsi qu'un rapport au format pdf à votre chargé de td.

Arnaud Carayol : `arnaud.carayol@univ-mlv.fr`
Laurent Braud : `laurent.braud@univ-mlv.fr`

Vous donnerez à votre mail le sujet suivant:

`[Archi L3][TP3] Nom1--Nom2`

Le rapport doit répondre aux questions posées dans le sujet et peut éventuellement contenir des portions de code pour illustrer votre propos. Votre code doit être commenté sinon il ne sera pas lu. Si un fichier que vous rendez n'a pas le comportement attendu, cela doit être dit dans le rapport.

Les questions bonus sont facultatives.

Dans cette séance, nous allons voir diverses utilisations de la pile. En particulier nous verrons comment réaliser des fonctions en assembleur. Nous allons commencer par quelques approfondissements sur l'instruction `mov`.

1 L'instruction `mov` dans tous ses états

On considérons le code ci-dessous.

```
segment .bss
mem resb 4
segment .text
        global  asm_main
asm_main:
        enter   0,0               ; mise en place
        pusha
un:     mov eax,mem
        mov ebx,mem
        mov eax,[eax]
deux:   inc eax
        mov [ebx],al
        inc ebx
        mov [ebx],ah
        inc ebx
        mov [ebx],al
        inc ebx
        mov [ebx],ah
trois:  mov eax,[mem]

        popa
```

```
        mov     eax, 0            ; retour vers le C
        leave
        ret
```

Question 1 Quel est le rôle de la seconde ligne du programme ?

Question 2 Quelle est la valeur de `eax` après l'exécution de la ligne étiquetée `un` ?

Question 3 Que signifie l'instruction `mov eax,[eax]` ?

Question 4 Que vaut `eax` avant ligne étiquetée `deux` ?

Question 5 Que signifie l'instruction `mov [ebx],al` ?

Question 6 Quelle est la valeur de `eax` après l'exécution de la ligne étiquetée `trois` ?

2 Utilisation de la pile

Rappelons que la pile désigne une zone mémoire qui se trouve en dessous de l'adresse `0xBFFF FFFF`. En mode protégé 32 bits, on stocke uniquement dans la pile des double-mots. Au début du programme, l'adresse du haut de la pile est contenue dans le registre `esp`. La pile grandie vers le bas: plus on rajoute d'éléments dans la pile plus la valeur de `esp` diminue.

Le but de cette section est de se familiariser avec l'utilisation de la pile et en particulier avec les instructions `push` et `pop`.

Question 7 Donnez des suites d'instructions n'utilisant que `mov`, `add` et `sub` correspondant aux instructions `push eax` et `pop ebx`.

Question 8 Pour les instructions ci-dessous, donnez l'état de la pile et des registres `eax`,`ebx`et `esp`avant l'exécution de la ligne `un`,`deux`, `trois` et `quatre`. On supposera que `esp` vaut initialement `0xBFFF0000`. On donnera la pile comme un tableau de double-mots.

```
        mov eax,0
        mov ebx,0
        push dword 12
        push dword 13
        push dword 15
un :    pop eax
        pop ebx
        add eax,ebx
deux:   push eax
        push ebx
        mov dword [esp+8],9
trois:  pop eax
        pop ebx
        pop ebx
quatre:
```

Question 9 Écrivez une programme qui lit des entiers au clavier tant qu'ils sont différents de -1 et affiche la liste de nombre lu dans l'ordre inverse de leur lecture. Par exemple, si vous entrez:

```
2
5
6
-1
```

Le programme affichera `6 5 2`.

Astuce Utilisez la pile pour stocker les entiers lus.

Bonus 1 Réalisez un programme qui lit une suite d'entiers terminée par -1 et qui affiche cette liste triée dans l'ordre croissant.

3 Appel de fonction

Le but de cette section est devoir comment sont gérés les appels de fonctions en assembleur.

Rappel: L'instruction `call label` empile l'adresse de l'instruction suivante sur la pile et fait un saut vers `label`. L'intruction `ret` dépile l'adresse strocké en haut de la pile et effectue un saut à cette adresse.

Question 10 Que fait la fonction `swap` dans le programme `fun.asm` ? Quelle est la valeur des registres `eax`, `ebx`, `ecx` et `edx` avant l'instruction `popa` ?

Bonus 2 Donnez la valeur en haut de pile aprés l'exécution de l'instruction `call swap`.

Question 11 Expliquez le comportement du programme `failure.asm`.

Bonus 3 Quel est le comportement du programme `failure2.asm` ? Supprimez les instructions `call print_int` et `call print_nl`. Expliquez le comportement du programme et l'influence de la suppression de ces deux lignes.

Question 12 Réalisez une fonction qui affiche la chaîne (terminée par un octet de valeur 0) et dont l'adresse du premier octet est stockée dans `eax`. Vous utiliserez l'appel système `write` vu au premier TP. Complétez le squelette donné dans `sq_write.asm`.

Question 13 Modifiez la fonction précédente pour que la valeur des registres généraux reste inchangée avant et aprés l'appel à votre fonction.

Question 14 Réalisez une fonction qui calcule de manière récursive la somme des n premiers entiers. Le paramètre (i.e. l'entier n) sera passé en paramètre dans le registre `eax`et le résultat sera renvoyé dans le registre `eax`.

Voici le pseudo-code pour cette fonction:

```
sum(n) :=
si n=0 alors retourne 0
       sinon retourne n + sum(n-1)
```

Question 15 Réalisez une fonction qui calcule de manière récursive le n-ième terme F_n de la suite de Fibonacci. Rappelons que cette suite est définie par $F_0 = 0$, $F_1 = 1$ et pour $n \geq 2$, $F_n = F_{n-1} + F_{n-2}$. Le paramètre (i.e. l'entier n) sera passé en paramètre dans le registre `eax`et le résultat sera renvoyé dans le registre `eax`.