Travaux Pratiques n°3

Architecture des ordinateurs.

—IR 1—

Avant la séance de travaux pratiques suivante, 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`
Guillaume Blin : `guillaume.blin@univ-mlv.fr`

Vous donnerez à votre mail le sujet suivant:

`[Archi IR1][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.

Dans cette séance, nous allons voir diverse utilisation de la pile et en particulier comment faire des fonctions en assembleur.

1 Utilisation de la pile

Le but de cette section est de se familiariser avec l'utilisation de la pile et des instructions `push` et `pop`. Nous ne stockerons dans la pile que des double-mots (4 octects).

Question 1 Donnez des suites d'instructions n'utilisant que `mov`, `add` et `sub` correspondant aux instructions `push eax` et `pop ebx`.

Question 2 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 3 Ecrire 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.

2 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 4 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 5 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 6 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 7 Modifiez la fonction précédente pour que la valeur des registres généraux reste inchangée avant et aprés l'appel à votre fonction.

Question 8 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`.