UNIVERSITÉ PARIS-EST MARNE-LA-VALLÉE

Travaux Dirigés de Compilation n°3

Licence d'informatique

———

Prise en main de l'assembleur

Le but de ce TD est de prendre en main **nasm**, un assembleur pour processeurs x86. Vous allez compiler des programmes et faire des opérations arithmétiques sur les registres puis des boucles.
Ces exercices sont conçus pour les machines de l'IGM. Sur d'autres machines, il peut y avoir des différences dans l'accès aux registres (pas d'accès à **rax**, **rbx**...), les conventions d'appel de fonctions, les instructions, les conditions d'utilisation de la bibliothèque fournie, la compilation.

▸ Exercice 1. *Compiler du code en assembleur*

*Le fichier **utils.asm** est un module qui vous permet de faire des affichages. Le fichier **squelette.asm** contient le code minimal en assembleur pour créer un exécutable qui ne fait rien mais ne provoque pas d'erreur à l'exécution.*

1. *Créez un fichier objet **squelette.o** avec la commande*
   nasm -f elf64 -o squelette.o squelette.asm
   *et de même pour **utils.asm**. Créez un exécutable à partir de **utils.o** et **squelette.o** par la commande*
   gcc -o squelette squelette.o utils.o -nostartfiles -no-pie
   *Testez. Vérifiez la valeur de retour du programme par la commande **echo $?**.*
2. *Faites un **makefile** pour automatiser les opérations précédentes.*

▸ Exercice 2. *Registres, chevauchements, appels de fonction*

1. *Dans le programme **reg.asm**, à chaque commentaire dans le code, notez les valeurs des registres **rax**, **rbx**, **rcx** et **rdx**.*
2. *Ajoutez des appels à la fonction **print_registers** et testez pour vérifier vos résultats. Le module **utils.asm** qui vous est fourni affiche **rax** et **rbx** en hexadécimal, et **rcx** et **rdx** en décimal. Si vous préférez un autre affichage, vous pouvez modifier la valeur de **format_registers**.*

Dans les exercices qui suivent, pensez à utiliser **print_registers** pour tester votre code.

▸ Exercice 3. *Instructions conditionnelles, terminaison, valeur de retour*

1. *Faites un programme en **nasm** qui renvoie 1 si **rax** est strictement plus grand que **rbx** et -1 sinon. Mettez vous-même des valeurs dans **rax** et **rbx** pour tester votre code.*
2. *Modifiez le programme pour qu'il renvoie 0 si **rax** et **rbx** sont égaux.*

▸ **Exercice 4.** ***Boucles***

1. *Faites un programme en* ***nasm*** *qui incrémente* ***rax*** *de 1 indéfiniment en utilisant l'instruction* ***jmp****, sans condition d'arrêt.*
2. *Modifiez votre programme pour qu'il s'arrête quand* ***rax*** *vaut 100.*
3. *Modifiez votre programme pour que l'incrémentation se fasse par un incrément de 9.*
4. *Modifiez votre programme pour qu'il s'arrête quand* ***rax*** *est supérieur ou égal à 100.*

▸ **Exercice 5.** ***Segment de données et adresses***

1. *Dans le programme* ***seg-donnees.asm****, à chaque commentaire dans le code, notez les valeurs des registres* ***rax****,* ***rbx****,* ***rcx*** *et* ***rdx****.*

```
; seg-donnees.asm
section .data
table dw 1, 2, 6, 3, 4, 22, 10, 0
section .text
global _start
extern print_registers
_start:
    mov rsi, table
    mov rax, 0
    mov rbx, 0
    mov rcx, 0
    mov rdx, 0
    mov ax, [rsi]
    ; point 1
    mov ax, [rsi+2]
    ; point 2
    mov eax,0
    mov rdx, 5
    mov al, [rsi+rdx*2]
    ; point 3
    mov rax, 60
    mov rdi, 0
    syscall
```

2. *Ajoutez des appels à la fonction* ***print_registers*** *et testez pour vérifier vos résultats.*

▸ **Exercice 6.** ***Taille des opérandes***

1. *Faites un programme qui déclare un tableau comme* ***table*** *dans l'exercice 5 et qui intervertit les deux premiers éléments du tableau.*

2. *Faites un programme qui déclare un tableau* ***table*** *contenant une liste d'entiers strictement positifs, sauf le dernier qui est nul. Votre programme doit ensuite mettre successivement chaque entier strictement positif dans* ***rax*** *jusqu'à tomber sur 0.*
3. *Ajoutez du code qui parcourt le tableau de droite à gauche à partir du 0 en mettant les entiers dans* ***rax****.*
4. *Faites un programme qui place dans* ***rbx*** *le plus grand entier de la liste.*

▸ **Exercice 7.** ***Orientation little-endian***

Dans le fichier ***little_endian.asm****, quelle est la valeur de* ***rax*** *au point désigné par un commentaire dans le code ?*

```
; little_endian.asm
section .data
table dw 1, 2, 6, 3, 4, 22, 10, 0
section .text
global _start
extern print_registers
_start:
    mov rsi, table
    mov rax, 0
    mov rbx, 0
    mov rcx, 0
    mov rdx, 0
    mov eax, [rsi+2]
    ; valeur de rax ?
    mov rax, 60
    mov rdi, 0
    syscall
```