Université de Bordeaux
Licence STS

ARCHITECTURE DES ORDINATEURS

TP : **01**

Premiers pas en assembleur y86

L'architecture y86 est un modèle de processeur inspiré de l'architecture dite « x86 » d'Intel. Elle en constitue une version simplifiée, à visée pédagogique. Elle a été conçue par les les auteurs du cours *Computer Systems : A Programmer's Perspective* (`http://csapp.cs.cmu.edu/`), en tant qu'outil de travaux pratiques.

Cet outil a été largement adopté par la communauté mondiale des enseignants en architecture des ordinateurs. Cependant, comme ses technologies de mise en œuvre devenaient obsolètes (notamment, Tcl/Tk), des clones ont été écrits par des tiers, notamment en utilisant des technologies web. Un tel outil, plus simple d'utilisation, a été mis en ligne à l'université de Bordeaux.

Pour utiliser le simulateur web, ouvrez ce lien dans votre navigateur web :

`https://dept-info.labri.fr/ENSEIGNEMENT/archi/y86js_v2/`

Si vous souhaitez installer le simulateur dans sa version Tcl/Tk, suivez les instructions de la feuille d'aide correspondante, disponible sur la page du cours.

Rappels

- La machine possède huit registres dont les noms apparaissent en bas du panneau de contrôle. Pour l'instant, on utilisera seulement les quatre premiers : `eax`, `ebx`, `ecx` et `edx`. Leurs noms et l'ordre dans lequel ils apparaissent résultent de l'évolution historique des processeurs Intel.
- L'instruction `irmovl 10,%edx` place la valeur décimale `10` dans le registre `edx` . Décryptage : `i` = source immédiate, `r` = destination registre, `mov` = *move*, `l` = `long` (mode 32 bits).
- L'instruction `rrmovl %edx, %eax` copie la valeur du premier registre dans le second. Décryptage : `i` a été remplacé par `r` = source registre.

Exercice 1 : Instructions arithmétiques et logiques

Question 1

Écrivez un programme qui place les valeurs décimales `43` et `26` respectivement dans les registres `eax` et `ecx`, puis calcule la somme des valeurs de ces deux registres dans le registre `edx`, sans modifier les valeurs des deux autres registres.

Question 2

Même question que précédemment, mais en utilisant l'opération `andl` au lieu de `addl`. Quel en est le résultat ? Pourquoi ?

Question 3

Même question que précédemment, mais en utilisant l'opération `xorl`. Quel en est le résultat ? Pourquoi ?

Exercice 2 : Codes de condition

Les codes de condition sont les trois drapeaux `SF` (« *sign flag* »), `ZF` (« *zero flag* ») et `OF` (« *overflow flag* ») inclus dans l'état de la machine. Ces drapeaux sont mis à jour avec les résultats de chaque nouvelle opération arithmétique et logique (classe d'instructions « `6x` »).

Question 1

Écrivez des petits programmes qui, en plaçant des valeurs particulières dans les deux registres `eax` et `ecx`, et en utilisant une seule opération opération arithmétique (c'est-à-dire, seulement `addl` ou `subl`) entre ces deux registres, permettent de mettre à `0` ou à `1` chacun des bits d'état.

Question 2

Même question lorsque le registre `eax` contient la valeur `1` et qu'on utilise uniquement `subl`.

Question 3

Que se passe-t-il quand on effectue un `xorl` entre un registre et lui-même ? Pourquoi ? Quelle est la valeur des drapeaux d'état ? À quoi cette instruction peut-elle servir ?

Question 4

Que se passe-t-il quand on effectue un `andl` entre un registre et lui-même ? Pourquoi ? Quelle est la valeur des drapeaux d'état ? À quoi cette instruction peut-elle servir ?

Rappels

Une instruction peut être précédée d'une *étiquette*, par exemple :

```
etiq:   addl %eax,%ecx
```

après quoi l'instruction :

```
        jmp etiq
```

place dans le compteur ordinal `PC` l'adresse de l'instruction correspondante. On peut faire aussi des sauts conditionnels en remplaçant `jmp` par `je` (« *jump if equal* ») ou `jne` (« *jump if not equal* »). Dans ces deux cas, le saut est exécuté ou non suivant la valeur du code de condition `ZF`.

Exercice 3 : Test simple

Réalisez l'équivalent du code C suivant :

```
long a = 2, b = 3, c = 0;
if (b <= a)
  c = 1;
```

Exercice 4 : Test alternatif

Réalisez l'équivalent du code C suivant :

```
long a = 2, b = 3, c = 0;
if (b <= a)
  c = 1;
else
  c = 2;
```