

## ARCHITECTURE DES ORDINATEURS

### TP : 08

## ÉTUDE DÉTAILLÉE DE L'ARCHITECTURE Y86

---

### 1 Objectif

L'objectif de ce TP est d'observer le fonctionnement détaillé du processeur y86, pour en comprendre tous les détails, à l'aide du simulateur ssim que l'on a utilisé au début du semestre.

### 2 Ressources

Vous trouverez à l'adresse :

<http://dept-info.labri.fr/ENSEIGNEMENT/archi/CSAPP/seq.pdf>

une version résumée de l'architecture y86. En particulier, les transparents 4 et 5 décrivent de façon schématique l'ensemble de l'architecture, avec les noms des fils de commande. De même, vous trouverez à l'adresse :

<http://dept-info.labri.fr/ENSEIGNEMENT/archi/Seq/>

un résumé du code HCL qui décrit précisément quels choix de routage des fils sont faits en fonction des différentes instructions.

### 3 Étude préliminaire

Écrivez un programme mettant quelques valeurs dans des registres, puis effectuant `addl %eax,%ebx`, observez les valeurs affichées par le simulateur y86, regardez le code HCL pour le cas OPL (qui effectue le `addl`), constatez que cela effectue bien l'addition et range le résultat comme attendu.

Faites de même pour `iaddl 1,%eax`, puis `mrmovl 4(%eax),%ebx`, puis `rmmovl %ebx,4(%eax)`, puis `pushl %eax`, puis `popl %eax`.

### 4 Étude de l'exécution d'un programme

Lancez le simulateur et chargez le programme `y86-code/asum.yo`. Exécutez-le instruction par instruction, et vérifiez, pour chaque type d'instruction, que vous comprenez bien **tous** les détails qui s'affichent à l'écran, ainsi que le code HCL correspondant : `new_pc`, `valA`, `valB`, etc.

Par exemple, pourquoi, dans le code HCL, pour `mem_addr`, utilise-t-on `valE` pour `pushl`, et `valA` pour `popl` ? Étudiez également bien l'instruction `ret` : c'est en fait l'une des plus compliquées !

Où est-il décidé de ne pas enregistrer les « *condition codes* » pour toute opération autre qu'arithmétique ?

Observez la différence de comportement des instructions `je` et `jne` durant la boucle.