

Ajout d'instructions à y86

1 Préliminaire

Rappel : à l'adresse

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

vous trouverez une version résumée de l'architecture y86, notamment sur les transparents 4 et 5 un rappel de l'ensemble de l'architecture avec les noms des fils.

À l'adresse

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

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

2 Étude de `iaddl`

Observez les différences entre le traitement de `OPL` et `IOPL` .

3 Ajout de `jreg`

L'instruction `jmp` prend en paramètre une simple constante, à laquelle brancher. On se propose d'ajouter `jreg`, qui prend en paramètre un registre, dont la valeur donne l'adresse à laquelle brancher. Dans le fichier `sim/seq/seq-std.hcl`, ajoutez

```
intsig JREG 'I_JREG'
```

à la suite des autres déclaration de ce type. Implémentez `jreg` dans ce même fichier et testez-la (inspirez-vous de `jmp` bien sûr, la seule différence est que l'on branche à l'adresse lue depuis le registre plutôt qu'à la constante immédiate donnée par l'instruction. Oui, on pourra utiliser `valA` au niveau de `new_pc`).

Pensez bien à recompiler le simulateur (avec `make`) et le relancer, pour mettre à jour le processeur simulé.

4 Ajout de `jmem`

De la même façon, ajoutez et testez l'instruction `jmem`, qui prend en paramètre un emplacement mémoire (8(`%eax`) par exemple) auquel lire l'adresse à laquelle brancher. On pourra par exemple s'inspirer de `rrmovl` en plus de `jmp`.

5 Ajout de `leave`

À la fin des fonctions on utilise souvent la paire d'instructions `rrmovl %ebp, %esp ; popl %ebp`. X86 fournit une instruction `leave` qui effectue précisément ces deux instructions en une seule opération. Il se trouve que `leave` est implémentable en une seule opération avec notre simulateur, faites-le, et testez-la!