Université de Bordeaux
Licence STS

ARCHITECTURE DES ORDINATEURS

TP : **10**

Étude détaillée de l'architecture y86(seq)

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 `y86js`, version 2.

2 Ressources

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

`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.

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.

Exercice 1 : 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`), et constatez que cela effectue bien l'addition et range le résultat comme attendu.

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

Exercice 2 : Étude de l'exécution d'un programme

Chargez le programme `tp_10.ys` sur le simulateur `y86js_v2` et compilez-le.

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, concernant l'état du processeur et l'ensemble des signaux HCL correspondants,tels que : `icode`, `rA`, `valP`, `valB`, `aluA`, `valE`, `new_pc`, etc.

Exercice 3 : Étude du code HCL du chemin de données

Considérez maintenant le code HCL décrivant le chemin de données du processeur.

Question 1

Pourquoi, dans ce code HCL, pour le signal `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 !

Question 2

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

Question 3

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

Exercice 4 : Ajout de l'instrucion `decl`

L'instruction `decl %reg` décrémente un registre, c'est-à-dire lui retranche 1. Elle est équivalente à l'instruction `isubl 1,%reg`, mais est censée prendre moins de place en mémoire, afin de réduire la taille des programmes qui l'utilisent.

Comme vous pourrez le voir dans l'onglet « Jeu d'instructions », cette instruction existe déjà dans le jeu des instructions reconnues par le processeur. Pour autant, elle n'est pas encore utilisable.

Question 1

Dans le code de l'exercice précédent, remplacez l'instruction `isubl 1,%ecx` par `decl %ecx` et relancez l'assemblage. Observez le code objet généré. Quel est le format de cette instruction ?

Question 2

Lorsque vous exécutez le code pas à pas, que se passe-t-il lorsque le processeur rencontre l'instruction `decl` ?

Question 3

L'onglet « HCL » vous donne accès au code HCL actuellement utilisé pour décrire le chemin de données du processeur.

Faites en sorte que l'instruction `decl` soit reconnue comme valide, en modifiant, pour le moment, uniquement le code du signal `instr_valid`.

Relancez l'exécution du programme Y86. Comment le processeur se comporte-t-il maintenant ?

Question 4

Modifiez le code HCL des autres signaux de l'étage « Fetch » afin que l'instruction `decl` soit parfaitement reconnue. Pour le confirmer, relancez l'exécution du programme et étudiez l'état des signaux issus de l'étage « Fetch », afin de vérifier qu'ils sont bien conformes à ce que l'instruction doit produire.

Question 5

Ajoutez au code HCL toutes les informations nécessaires pour rendre l'instruction `decl` pleinement fonctionnelle.

Exercice 5 : De la place dans les opcodes (1)

Les opcodes sont des denrées rares, puisqu'il n'y en a que 16 en Y86. On peut donc vouloir « compacter » les opcodes afin de garder de la place pour de nouvelles instructions.

Question 1

Considérez les instructions `pushl` et `call`. Quelles sont leurs similarités ? Quelles sont leurs différences ?

Question 2

Afin de gagner de la place, on va replacer l'instruction `call` par l'instruction `ncall`, qui prendra le même opcode que `pushl`, mais avec un `ifun` égal à 1.

Dans le code de l'exercice précédent, remplacez l'instruction `call sum` par `ncall sum` et relancez l'assemblage. Observez le code objet généré.

Modifiez le code HCL des signaux de l'étage « Fetch » afin que l'instruction `ncall` soit parfaitement reconnue. Pour le confirmer, relancez l'exécution du programme et étudiez l'état des signaux issus de l'étage « Fetch », afin de vérifier qu'ils sont bien conformes à ce que l'instruction doit produire.

Question 3

Modifiez le code HCL des autres étages afin que les instructions `pushl` et `ncall` fonctionnent comme attendu. Toutes les références à l'opcode `CALL` doivent avoir disparu.

Exercice 6 : De la place dans les opcodes (2)

Mêmes questions pour l'instruction `nret`, qui factorise l'opcode de l'instruction `popl`.