Université Bordeaux 1 — U.F.R. Mathématiques et Informatique
Licence Informatique 2ème année — Année 2011 – 2012

ARCHITECTURE DES ORDINATEURS

Devoir Surveillé 2

1 heure 20 minutes
sans documents

N.B. : - Travaillez calmement. Ne bâclez pas vos réponses : il vaut mieux traiter correctement moins de questions que tout faire de travers.
- Les réponses aux questions doivent être argumentées et aussi concises que possible.
- Le barème est donné à titre indicatif.

Question 1 (13 points)

Les ordinateurs ne savent manipuler que des nombres binaires, alors que les humains expriment les nombres sous forme décimale. Lors d'un calcul interactif avec un ordinateur, il faut donc convertir les nombres décimaux en nombres binaires, puis à la fin du calcul binaire convertir le résultat en décimal, ce qui est très coûteux. Pour éviter cela, certains processeurs, utilisés par exemple dans les calculatrices, disposent d'instructions pour travailler directement avec des nombres entiers en base 10, en utilisant une représentation appelée « BCD », pour « *Binary-Coded Decimal* », soit en Français « *Décimal Codé en Binaire* ». L'idée du codage BCD est de coder chaque chiffre décimal sous forme binaire et de manipuler directement ce codage en machine, pour que chaque nombre ainsi calculé puisse se lire directement sous forme décimale. Quatre bits sont nécessaires au stockage des chiffres de 0 à 9. On peut donc représenter en BCD le nombre décimal $815_{(10)}$ sur deux octets, sous la forme $0000\ 1000\ 0001\ 0101_{(2)}$.

(1.1) (1 point)

Pour chacun des nombres décimaux suivants, donnez leur écriture binaire BCD, ainsi que l'écriture hexadécimale de leur codage BCD. Quel lien y a-t-il entre l'écriture décimale d'un nombre et son écriture hexadécimale en codage BCD ?

- $42_{(10)}$;
- $6502_{(10)}$.

On suppose que l'on dispose déjà d'un additionneur binaire classique sur 4 bits, avec une retenue entrante c_{in}, une retenue sortante c_{out}, deux groupes d'entrées, étiquetées de x_0 à x_3 et de y_0 à y_3 pour les deux nombres binaires x et y d'entrée à additionner (le bit 0 est le bit de poids le plus faible), et d'un groupe de sorties étiquetées de s_0 à s_3 pour le résultat. Dans la suite, cette fonction pourra être représentée schématiquement par une boîte A. On dispose en outre des portes logiques classiques.

(1.2) (2 points)

Construisez un tableau donnant, pour chaque chiffre décimal compris entre 0 et 9, les valeurs binaires codées sur 4 bits correspondant à :

- la représentation BCD de ce chiffre ;
- le résultat de l'ajout de 1 à cette représentation BCD, codée sous la forme de deux chiffres BCD (dizaine et unité) ;
- le résultat de l'ajout de 1 à cette représentation BCD, au moyen de l'additionneur binaire classique ;
- la différence entre les deux résultats précédents, interprétés tous les deux comme des nombres binaires sur 8 bits.

(1.3) (2 points)

Réalisez le même tableau que pour la question précédente, mais en ajoutant 3 au lieu de 1 aux chiffres de la première colonne.

(1.4) (2 points)

Par extension des résultats contenus dans les deux tableaux ci-dessus, quelles sont les valeurs que peut prendre la différence entre la somme BCD et la somme binaire de deux chiffres BCD quelconques ? Dans quel cas la différence entre les deux résultats est-elle non nulle, et que vaut-elle alors ?

(1.5) (3 points)

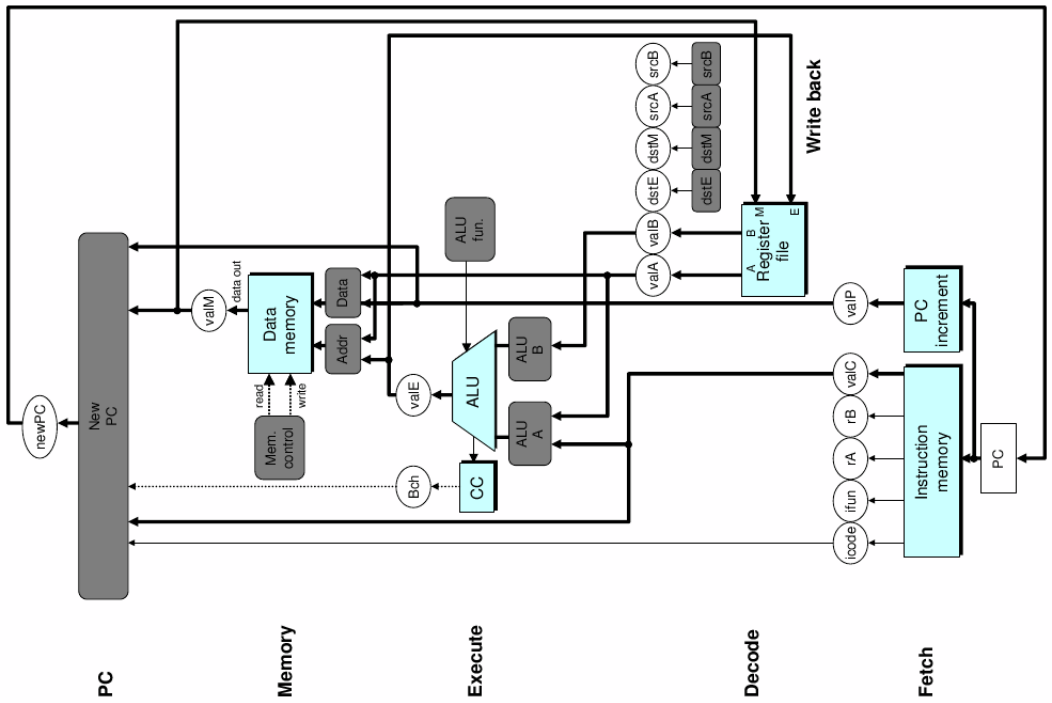
Donnez et justifiez la formule logique, puis faites le schéma, d'un circuit qui détecte quand la somme BCD calculée par un additionneur 4 bits classique est fausse (c'est-à-dire quand la différence étudiée à la question précédente est non nulle). Ce circuit a comme entrées la valeur s de la somme calculée par l'additionneur ainsi que la retenue sortante. Sa sortie e vaut 1 s'il y a erreur, et 0 sinon. Dans la suite, cette fonction pourra être représentée schématiquement par une boîte E.

(1.6) (3 points)

En utilisant, entre autres, le circuit de la question précédente et deux additionneurs binaires à 4 bits, dessinez un additionneur BCD qui prend en entrée deux chiffres BCD et une retenue d'entrée, et produit un chiffre BCD et une retenue de sortie.

Question 2 (7 points)

La figure ci-dessous est le schéma de l'architecture y86 séquentielle.



Le tableau ci-dessous représente le fonctionnement des différents étages lors de l'exécution des trois instructions « `rrmovl rA, rB` », « `popl rA` » et « `call valC` ».

Étage	rrmovl rA, rB	popl rA	call valC
Fetch	icode:ifun = M_1[PC] rA:rB = M_1[PC+1] valP = PC + 2	icode:ifun = M_1[PC] rA:rB = M_1[PC+1] valP = PC + 2	icode:ifun = M_1[PC] valC = M_4[PC+1] valP = PC + 5
Decode	valA = R[rA]	valA = R[%esp] valB = R[%esp]	valB = R[%esp]
Execute	ValE = 0 + valA	valE = valB + 4	valE = valB + (-4)
Memory		ValM = M_4[valA]	M_4[valE] = valP
Write back	R[rB] = valE	R[%esp] = valE R[rA] = valM	R[%esp] = valE
PC update	PC = valP	PC = valP	PC = valC

(2.1) (4 points)

En vous basant sur les informations précédentes, remplissez le tableau correspondant aux instructions « `rmmovl rA,D(rB)` », « `pushl rA` » et « `ret` ».

Le code ci-dessous est un fragment incomplet de code HCL correspondant à l'architecture y86 séquentielle.

```
int srcA = [
                icode in { RRMOVL, OPL } : rA;
                icode in { POPL } : RESP;
                1 : RNONE; # Don't need register
];

int srcB = [
                icode in { OPL, IOPL, MRMOVL } : rB;
                icode in { POPL, CALL } : RESP;
                1 : RNONE;  # Don't need register
];

int dstE = [
                icode in { RRMOVL, IRMOVL, OPL, IOPL } : rB;
                icode in { POPL, CALL } : RESP;
                1 : RNONE;  # Don't need register
];

int mem_addr = [
                icode in { CALL, MRMOVL } : valE;
                icode in { POPL } : valA;
                # Other instructions don't need address
];

## Select memory input data
int mem_data = [

                icode == CALL : valP;
                # Default: Don't write anything
];
```

(2.2) (3 points)

Complétez ce code afin de prendre en compte les trois instructions que vous venez de définir.