Université de Bordeaux
Licence Informatique 2ème année

U.F. Informatique
Année 2019 – 2020

ARCHITECTURE DES ORDINATEURS

Devoir Surveillé 1

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 (5 points)

On considère des entiers codés sur 16 bits, interprétés comme des entiers relatifs codés en complément à deux. Chacun de ces entiers peut (et doit) s'écrire avec quatre chiffres hexadécimaux (on omettra le préfixe « 0x »).
Soient les cinq entiers suivants :

M	N	P	Q	R
ABAD	1DEA	3456	DADA	789A

(1.1) (1 point)
Parmi ces 5 entiers, quels sont ceux qui sont négatifs ? Expliquez pourquoi.

(1.2) (1 point)
Classez ces entiers du plus petit au plus grand (par exemple : « M < N < P < Q < R », mais cette réponse choisie arbitrairement n'est probablement pas la bonne...). Justifiez la réponse sur votre copie.

(1.3) (1 point)
Calculez $P + Q$. Pour cela, effectuez l'addition en binaire directement sur la copie — une réponse brute ne rapportera aucun point. Y a-t-il débordement (*overflow*) ?

(1.4) (2 points)
Calculez $-R$ et $M - R$. Y a-t-il débordement (*overflow*) ? Comme pour la question précédente, le détail des calculs doit figurer sur la copie.

Question 2 (7 points)

On souhaite concevoir un circuit permettant de multiplier ensemble deux nombres à deux bits $\mathtt{A} = \mathtt{a}_1\mathtt{a}_0$ et $\mathtt{B} = \mathtt{b}_1\mathtt{b}_0$ en un produit `P`.

(2.1) (1 point)
Sur combien de bits le produit `P` sera-t-il codé au plus ? Justifiez votre réponse.

(2.2) (2 points)
Donnez les tables logiques, puis les fonctions logiques, d'un demi-additionneur à un bit, prenant en entrée deux valeurs `x` et `y` sur un bit, et fournissant la somme `s` et la retenue `c` codant `x + y`.
À partir de maintenant, et même si vous n'avez pas répondu aux questions précédentes, vous pouvez utiliser ce demi additionneur à un bit `HA`*, possédant les sorties* `HA:s(x,y)` *et* `HA:c(x,y)`*, et le comportement décrit ci-dessus.*

(2.3) (4 points)
Considérez la façon classique d'effectuer des multiplications, à savoir : calcul des produits partiels de A par chacun des chiffres de B, puis addition des résultats par colonnes.
Donnez les équations logiques des produits partiels de A par b_0 puis par b_1, puis, par ordre de poids croissant, chacune des équations logiques des valeurs p_i constituant P.

Question 3 (8 points)

Voici le texte d'un programme écrit en langage Y86. Tous les commentaires ayant malencontreusement été effacés, il vous faut partir de zéro pour comprendre ce que fait ce programme, en répondant aux questions ci-dessous.

Attention : les réponses qui, au lieu d'expliquer, paraphrasent simplement le code (telles que « on soustrait %edx à %ebx ») ne rapporteront aucun point.

```
0x000:              |          .pos    0
0x000: 308400020000 | main:    irmovl  pile,%esp
0x006: 308004010000 |          irmovl  t,%eax
0x00c: a008         |          pushl   %eax
0x00e: 8020000000   |          call    mystere
0x013: c08404000000 |          iaddl   4,%esp
0x019: 400800010000 |          rmmovl  %eax,n
0x01f: 10           |          halt
                    |
0x020: 502404000000 | mystere: mrmovl  4(%esp),%edx
0x026: 308000000000 |          irmovl  0,%eax
0x02c: 501200000000 | et1:     mrmovl  (%edx),%ecx
0x032: 6011         |          addl    %ecx,%ecx
0x034: 7350000000   |          je      et2
0x039: 401200000000 |          rmmovl  %ecx,(%edx)
0x03f: c08001000000 |          iaddl   1,%eax
0x045: c08204000000 |          iaddl   4,%edx
0x04b: 702c000000   |          jmp     et1
0x050: 90           | et2:     ret
                    |
0x100:              |          .pos    0x100
0x100: 00000000     | n:       .long   0
0x104: 02000000     | t:       .long   2
0x108: 04000000     |          .long   4
0x10c: 07000000     |          .long   7
0x110: 00000000     |          .long   0
0x114: 01000000     |          .long   1
                    |
0x200:              |          .pos    0x200
0x200: 00000000     | pile:    .long   0
```

(3.1) (1 point)

Quels sont les paramètres de la fonction `mystere` ? Expliquer quelle partie du code permet de répondre à cette question.

(3.2) (6 points)

Que calcule la fonction `mystere`, et que renvoie-t-elle ? Pour cela :

a). Pour chacun des registres utilisés au sein de cette fonction, donnez son nom puis, en une unique phrase synthétique, son rôle au sein de la fonction. Vous pourrez ensuite compléter cette phrase en précisant l'évolution du registre au cours du déroulement de la fonction ;

b). Indiquez à quoi sert, dans le cas de cette fonction, le test de l'adresse `0x034` ;

c). Indiquez à quoi sert, dans le cas de cette fonction, l'instruction d'adresse `0x045` ;

d). Indiquez à quelles modifications en mémoire centrale ce programme conduit, dans la zone comprise entre `0x100` et `0x117`.

(3.3) (1 point)

Si ce code avait été produit par un compilateur C, quel aurait été le prototype de la fonction `mystere` ?