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

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 (7 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
dead	beef	6502	42ab	8080

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

(1.2) (2 points)
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) (2 points)
Calculez $M + N$. 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 $-N$ et $M - N$. Comme pour la question précédente, le détail des calculs doit figurer sur la copie.

Question 2 (13 points)

La page suivante contient 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.

(2.1) (2 points)

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

(2.2) (1 point)

Quel est le rôle de l'instruction d'adresse `0x01d` ?

(2.3) (1 point)

Quel est le rôle de l'instruction d'adresse `0x023` ? Quel lien existe-t-il entre cette instruction et l'appel de la fonction `mystere` ?

(2.4) (1 point)

Aurait-on pu écrire de façon plus compacte en mémoire l'instruction d'adresse `0x038` ?

(2.5) (4 points)

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

- Expliquez le rôle de chacun des registres utilisés au sein de cette fonction et leur évolution au cours du déroulement de la fonction. Pour comprendre celle-ci, intéressez-vous en particulier aux registres %eax, %esi et %edx.
- Expliquez le rôle des étiquettes `et1`, `et2` et `et3`.

(2.6) (1 point)

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

(2.7) (2 points)

Expliquez le codage binaire de l'instruction d'adresse `0x032`. Vous pouvez pour cela vous aider de celui de l'adresse `0x069`.

(2.8) (1 point)

En vous basant sur le codage de certaines instructions (dites lesquelles), quels sont les numéros sur 4 bits correspondant aux registres %eax, %ecx et %esi ?

```
0x000:              |           .pos    0
0x000: 308400020000 | main:     irmovl  pile,%esp
0x006: 2045         |           rrmovl  %esp,%ebp
0x008: 308000010000 |           irmovl  m,%eax
0x00e: a008         |           pushl   %eax
0x010: 308008010000 |           irmovl  t,%eax
0x016: a008         |           pushl   %eax
0x018: 802a000000   |           call    mystere
0x01d: c08408000000 |           iaddl   8,%esp
0x023: 400804010000 |           rmmovl  %eax,n
0x029: 10           |           halt
                    |
0x02a: a058         | mystere:  pushl   %ebp
0x02c: 2045         |           rrmovl  %esp,%ebp
0x02e: a068         |           pushl   %esi
0x030: a038         |           pushl   %ebx
0x032: 506508000000 |           mrmovl  8(%ebp),%esi
0x038: 308000000000 |           irmovl  0,%eax
0x03e: 2002         |           rrmovl  %eax,%edx
0x040: 501600000000 | et1:      mrmovl  (%esi),%ecx
0x046: 6211         |           andl    %ecx,%ecx
0x048: 7369000000   |           je      et3
0x04d: 2013         |           rrmovl  %ecx,%ebx
0x04f: 6123         |           subl    %edx,%ebx
0x051: 7158000000   |           jle     et2
0x056: 2012         |           rrmovl  %ecx,%edx
0x058: c08001000000 | et2:      iaddl   1,%eax
0x05e: c08604000000 |           iaddl   4,%esi
0x064: 7040000000   |           jmp     et1
0x069: 50650c000000 | et3:      mrmovl  12(%ebp),%esi
0x06f: 402600000000 |           rmmovl  %edx,(%esi)
0x075: b038         |           popl    %ebx
0x077: b068         |           popl    %esi
0x079: b058         |           popl    %ebp
0x07b: 90           |           ret
                    |
0x100:              |           .pos    0x100
0x100: 00000000     | m:        .long   0
0x104: 00000000     | n:        .long   0
0x108: 23000000     | t:        .long   0x23
0x10c: 56000000     |           .long   0x56
0x110: 89000000     |           .long   0x89
0x114: 12000000     |           .long   0x12
0x118: 00000000     |           .long   0x00
                    |
0x200:              |           .pos    0x200
0x200: 00000000     | pile:     .long   0
```