Université Bordeaux 1 — U.F.R. Mathématiques et Informatique
Licence Informatique 2ème année — Année 2012 – 2013

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 (6 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
8632	7654	abba	f00d	07ea

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

(1.2) (1 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 $M + P$. 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 $-Q$ et $M - Q$. Comme pour la question précédente, le détail des calculs doit figurer sur la copie.

Question 2 (14 points)

Vous trouverez plus bas 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 :
- *pas de panique, inutile de commencer par lire en détail ce programme ;*
- *beaucoup de questions sont indépendantes ;*
- *les réponses qui, au lieu d'expliquer, paraphrasent simplement le code (telles que « on soustrait %edx à %ebx ») ne rapporteront aucun point.*

On va tout d'abord s'intéresser à la fonction principale.

(2.1) (1 point)
Combien de paramètres la fonction `map` semble-t-elle prendre ? Renvoie-t-elle une valeur ? Expliquer quelle(s) partie(s) du code permet(tent) de répondre à ces questions.

(2.2) (1 point)
Pourquoi l'instruction d'adresse `0x008` est-elle un `mrmovl` alors que l'instruction d'adresse `0x010` est un `irmovl` ? Donnez le prototype de la fonction `map`.

Intéressons-nous maintenant à la fonction `f`. Cette fonction appelle une fonction `mul`, de prototype `int mul (int a, int b)`, qui renvoie $a \times b$ comme son nom le laisse supposer. Le code de cette fonction n'est pas fourni.

(2.3) (1 point)

Quel est le rôle des instructions d'adresses `0x072` à `0x075` ?

(2.4) (1 point)

Combien de paramètres la fonction `f` semble-t-elle prendre ? Renvoie-t-elle une valeur ? Expliquer quelle(s) partie(s) du code permet(tent) de répondre à ces questions.

(2.5) (2 points)

Quel est le rôle de l'instruction d'adresse `0x07c` ? À quel mode d'adressage cela correspond-il ? Quel est donc le type du paramètre correspondant de la fonction `f` ?

(2.6) (1 point)

Donnez le prototype de la fonction `f` et expliquez ce qu'elle fait.

Intéressons-nous ensuite à la fonction `map`.

(2.7) (1 point)

Quel est le rôle des instructions d'adresses `0x02e` à `0x033` ? Pourquoi sont-elles présentes ici ?

(2.8) (1 point)

Quel(s) paramètre(s) la fonction `map` passe-t-elle successivement à la fonction `f` à chaque tour de boucle ? Comment ce paramètre évolue-t-il ?

(2.9) (1 point)

Combien de fois effectue-t-on la boucle de corps commençant en `map1` ?

(2.10) (2 points)

Sachant ce que fait la fonction `f`, que fait la fonction `map` ? Quelle sera la valeur numérique calculée par cette fonction dans cet exemple précis ?

Et pour finir...

(2.11) (2 points)

Écrivez le code de la fonction `mul`. Vous n'avez pas besoin de coder la version la plus optimisée possible ; une version fonctionnelle suffira.

```
0x000:              |          .pos    0
0x000: 308400020000 | main:    irmovl  pile,%esp
0x006: 2045         |          rrmovl  %esp,%ebp
0x008: 500804010000 |          mrmovl  n,%eax
0x00e: a008         |          pushl   %eax
0x010: 308008010000 |          irmovl  t,%eax
0x016: a008         |          pushl   %eax
0x018: 802a000000   |          call    map
0x01d: c08408000000 |          iaddl   8,%esp
0x023: 400800010000 |          rmmovl  %eax,r
0x029: 10           |          halt
                    |
0x02a: a058         | map:     pushl   %ebp
0x02c: 2045         |          rrmovl  %esp,%ebp
0x02e: a038         |          pushl   %ebx
0x030: a068         |          pushl   %esi
0x032: a078         |          pushl   %edi
0x034: 506508000000 |          mrmovl  8(%ebp),%esi
0x03a: 50750c000000 |          mrmovl  12(%ebp),%edi
0x040: 6300         |          xorl    %eax,%eax
0x042: c18701000000 | map1:    isubl   1,%edi
0x048: 7269000000   |          jl      map2
0x04d: 2003         |          rrmovl  %eax,%ebx
0x04f: a068         |          pushl   %esi
0x051: 8072000000   |          call    f
0x056: c08404000000 |          iaddl   4,%esp
0x05c: 6030         |          addl    %ebx,%eax
0x05e: c08604000000 |          iaddl   4,%esi
0x064: 7042000000   |          jmp     map1
0x069: b078         | map2:    popl    %edi
0x06b: b068         |          popl    %esi
0x06d: b038         |          popl    %ebx
0x06f: b058         |          popl    %ebp
0x071: 90           |          ret
```

```
0x072: a058         | f:       pushl   %ebp
0x074: 2045         |          rrmovl  %esp,%ebp
0x076: 502408000000 |          mrmovl  8(%esp),%edx
0x07c: 500200000000 |          mrmovl  (%edx),%eax
0x082: a008         |          pushl   %eax
0x084: a008         |          pushl   %eax
0x086: 8094000000   |          call    mul
0x08b: c08408000000 |          iaddl   8,%esp
0x091: b058         |          popl    %ebp
0x093: 90           |          ret
```

```
0x100:              |          .pos    0x100
0x100: 00000000     | r:       .long   0
0x104: 04000000     | n:       .long   4
0x108: 02000000     | t:       .long   2
0x10c: 03000000     |          .long   3
0x110: 01000000     |          .long   1
0x114: 06000000     |          .long   6
                    |
0x200:              |          .pos    0x200
0x200: 00000000     | pile:    .long   0
```