
ARCHITECTURE DES ORDINATEURS

Devoir Surveillé 1

1 heure et 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.

Exercice 1 (50 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
805F	45A2	3661	CAD1	E036

- (1.1) (10 points)
Parmi ces 5 entiers, quels sont ceux qui sont négatifs ? Expliquez pourquoi.
- (1.2) (10 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) (10 points)
Calculez $N + P$. 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) (20 points)
Calculez $-N$ et $M - N$. Y a-t-il débordement (*overflow*) ? Comme pour la question précédente, le détail des calculs doit figurer sur la copie.

Exercice 2 (150 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) (20 points)

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

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

(2.2) (10 points)

Quel est le rôle de l'instruction d'adresse `0x01b`? Quel lien existe-t-il entre cette instruction et l'appel de la fonction `mystere`?

(2.3) (20 points)

Quel est le rôle de l'instruction d'adresse `0x028`?

(2.4) (10 points)

Aurait-on pu écrire de façon plus compacte en mémoire l'instruction d'adresse `0x030`?

(2.5) (10 points)

Quelle est la valeur qui sera stockée à l'adresse `0x100` à la fin de l'exécution du programme? Justifiez votre réponse.

(2.6) (30 points)

Dans la fonction `mystere`, quelles sont les rôles respectifs des différents registres utilisés? Justifiez vos réponses.

(2.7) (10 points)

Dans la fonction `mystere`, quels sont les rôles respectifs des étiquettes `et1`, `et2` et `et3`?

(2.8) (10 points)

Dites, en une phrase, ce que renvoie la fonction `mystere`.

(2.9) (20 points)

Expliquez le codage binaire de l'instruction d'adresse `0x03c`. Vous pouvez pour cela vous aider de celui de l'adresse `0x05a`.

(2.10) (10 points)

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:	308004000000		irmovl	4,%eax
0x00c:	a008		pushl	%eax
0x00e:	308004010000		irmovl	t,%eax
0x014:	a008		pushl	%eax
0x016:	8028000000		call	mystere
0x01b:	c08408000000		iaddl	8,%esp
0x021:	400800010000		rmmovl	%eax,r
0x027:	10		halt	
0x028:	a068	mystere:	pushl	%esi
0x02a:	506408000000		mrmovl	8(%esp),%esi
0x030:	308000000000		irmovl	0,%eax
0x036:	50140c000000		mrmovl	12(%esp),%ecx
0x03c:	c18101000000	et1:	isubl	1,%ecx
0x042:	7265000000		j1	et3
0x047:	502600000000		mrmovl	(%esi),%edx
0x04d:	6102		subl	%eax,%edx
0x04f:	715a000000		jle	et2
0x054:	500600000000		mrmovl	(%esi),%eax
0x05a:	c08604000000	et2:	iaddl	4,%esi
0x060:	703c000000		jmp	et1
0x065:	b068	et3:	popl	%esi
0x067:	90		ret	
0x100:			.pos	0x100
0x100:	00000000	r:	.long	0
0x104:	23000000	t:	.long	0x23
0x108:	f9000000		.long	0xF9
0x10c:	56000000		.long	0x56
0x110:	12000000		.long	0x12
0x114:	ff000000		.long	0xFF
0x118:	7f000000		.long	0x7F
0x200:			.pos	0x200
0x200:	00000000	pile:	.long	0