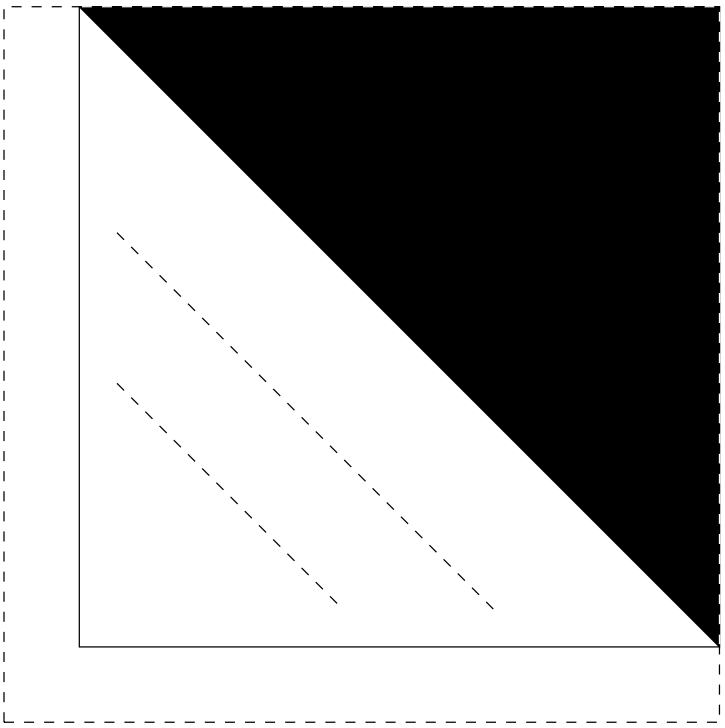
CC1 - LIF6 Printemps 2013
Durée prévue : 1h15

Aucun document autorisé. Calculatrices, téléphones et ordinateurs portables interdits.

N'oubliez pas de noter votre nom, votre prénom et votre numéro d'étudiant sous le cache rabattable.

Justifiez brièvement et précisément vos réponses, tout en respectant les emplacements prévus.

Le barème est donné à titre indicatif, et pourra être modifié.



Ne rien écrire dans ce cadre.
Partie 1 :
Partie 2 :
Partie 3 :
Partie 4 :

1 Mémoire cache (5 pts)

Un ordinateur dispose d'une mémoire centrale de 64 kio : les adresses sont codées sur 16 bits, et la taille de chaque case mémoire est de 1 o. Entre le processeur et la mémoire centrale est placée une petite mémoire cache directe, composée de 8 entrées pouvant chacune contenir 32 o. Pour savoir à quelle ligne de cache appartient une adresse mémoire, et savoir dans quelle entrée du cache sera rangée cette ligne, on décompose les adresses en trois champs :

INDICATEUR								LIGNE			OCTET				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

1. (0.5 points) Pourquoi le champ OCTET comporte-t-il 5 bits ? (justifiez)

   .....

   .....

2. (0.5 points) Pourquoi le champ LIGNE comporte-t-il 3 bits ? (justifiez)

   .....

   .....

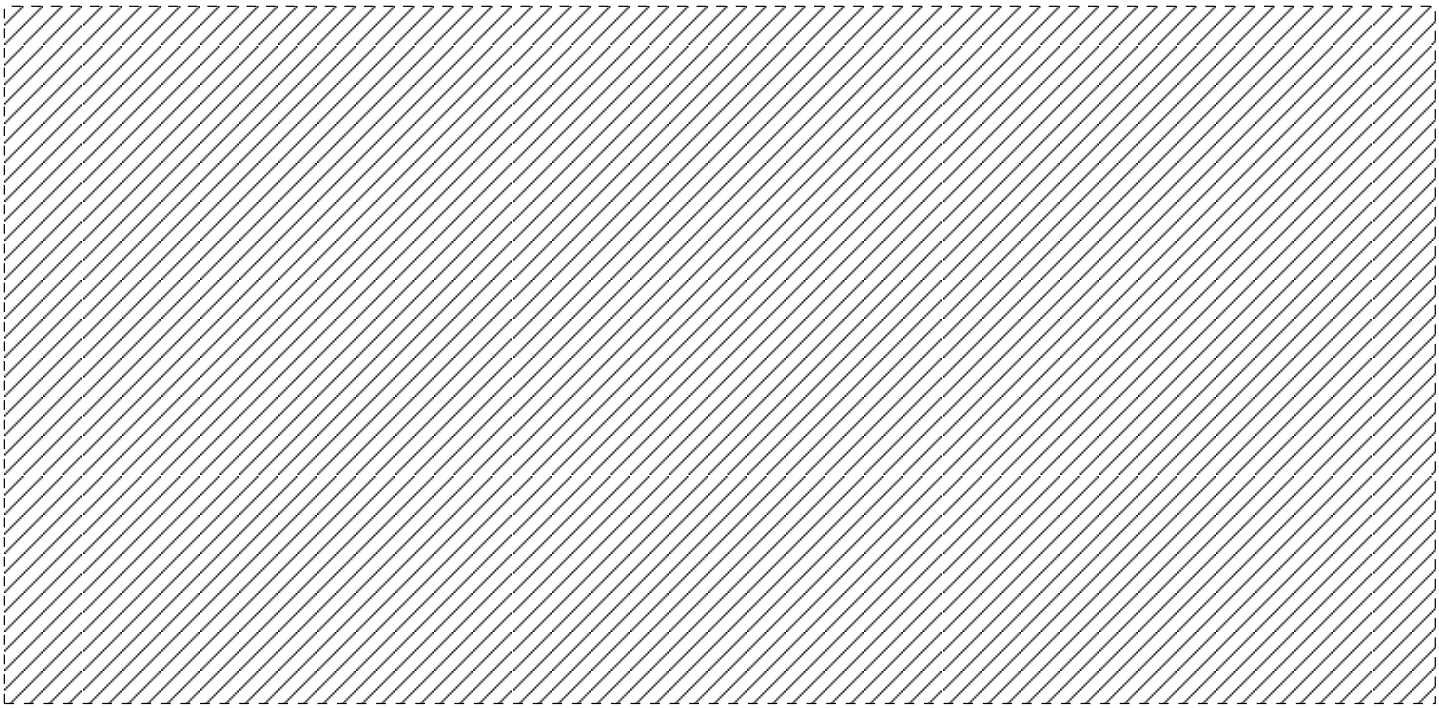
3. (0.5 points) En combien de lignes de caches se décompose la mémoire centrale ? (justifiez)

   .....

   .....

4. (2 points) Complétez le tableau suivant ; notez les numéros de lignes en indice de la lettre ℓ, et les numéros d'entrées en indice de la lettre e.

adresse (hexa.)	adresse (binaire)																ligne (décimal)	entrée (décimal)
$0BBF_H$																		
$0BC0_H$																		
$0BC1_H$																		
$05AE_H$																		
$05BF_H$																		
$05C0_H$																		



5. (1.5 points) Le processeur effectue successivement des accès aux adresses $0BBF_H$, $0BC0_H$, $0BC1_H$, $05AE_H$, $05BF_H$, $05C0_H$ (à celles-là seulement). Indiquez le contenu des entrées du cache après chaque accès (laissez vides les cases où on ne peut pas savoir), et signalez chaque défaut de cache (par une croix dans la dernière colonne).

adresse	entrées								défaut
	e_0	e_1	e_2	e_3	e_4	e_5	e_6	e_7	
$0BBF_H$									
$0BC0_H$									
$0BC1_H$									
$05AE_H$									
$05BF_H$									
$05C0_H$									

2 Traduction de programmes en langage machine (5 pts)

On se place sur un processeur disposant de 8 registres généraux 16 bits, notés dans le langage d'assemblage `r0`,...,`r7`. Les adresses sont codées sur 16 bits, et les cases de la mémoire centrale sont d'une taille de 1 o. Le jeu d'instruction du processeur comporte des instructions arithmétiques, dont

- `add rd, ra, rb` qui effectue `rd := ra + rb`, et qui a pour opcode $(0001001)_2$,
- `sub rd, ra, rb` qui effectue `rd := ra - rb`, et qui a pour opcode $(0001010)_2$,
- `mul rd, ra, rb` qui effectue `rd := ra * rb`, et qui a pour opcode $(0001011)_2$.

Les instructions arithmétiques ou logiques sont codées de la façon suivante :

opcode							`rd`			`ra`			`rb`		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Le jeu d'instruction comporte également deux instructions d'accès à la mémoire :

- `ld rd, @` qui charge le contenu de la case mémoire à l'adresse `@` dans `rd`, et qui a pour opcode $(10001)_2$,
- `st rs, @` qui range le contenu de `rs` dans la case mémoire à l'adresse `@`, et qui a pour opcode $(10010)_2$.

Ces instructions d'accès à la mémoire sont codées comme suit :

opcode					`rd/rs`			adresse `@`															
23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

On note respectivement `A`, `B` et `C` les variables se trouvant aux adresses $4AF0_H$, $4AF1_H$ et $4AF2_H$ de la mémoire.

1. (1.5 points) On considère le programme suivant écrit en langage d'assemblage :

```
ld r5, @4AF0    # ..........................................
ld r6, @4AF1    # ..........................................
mul r7, r6, r5  # ..........................................
mul r7, r7, r5  # ..........................................
st r7, @4AF2    # ..........................................
```

Commentez ce que fait chaque instruction, puis donnez la formule évaluée par le programme.

.....

2. (2 points) Traduisez chaque instruction en langage machine dans le tableau ci-dessous ;

instruction	codage binaire																							codage hexa.
`ld r5, @4AF0`																								
`ld r6, @4AF1`																								
`mul r7, r6, r5`																								
`mul r7, r7, r5`																								
`st r7, @4AF2`																								

3. (1.5 points) Donnez le programme en langage machine, en le plaçant en mémoire à partir de l'adresse $490A_H$.

adresse (hexa.)	contenu (hexa.)	adresse (hexa.)	contenu (hexa.)	adresse (hexa.)	contenu (hexa.)
$490A_H$		4910_H		4916_H	
$490B_H$		4911_H		4917_H	
$490C_H$		4912_H		4918_H	
$490D_H$		4913_H		4919_H	
$490E_H$		4914_H		$491A_H$	
$490F_H$		4915_H		$491B_H$	

3 Représentation de nombres à virgule flottante (5 pts)

On travaille sur une machine où les flottants binaires sont codés sur 12 bits. On considère le code :

$c =$

11	10	9	8	7	6	5	4	3	2	1	0
s	e_c				m_c						

Le nombre représenté est $f(c) = (-1)^s \times (1, m_c)_2 \times 2^{e(c)}$. Si $1 \leq N(e_c) \leq 14$, alors $e(c) = N(e_c) - 2^3$ (on ne se préoccupe pas du codage des sous-normaux ni des valeurs exceptionnelles ici).

1. (2 points) Quel nombre est représenté par le code **octal** $c = 7257_8$, si on le regarde comme le codage d'un flottant ? Donnez votre résultat sous la forme d'une représentation positionnelle à virgule en décimal.

.....

.....

.....

.....

.....

.....

.....

.....

.....

2. (1 point) Comment est représenté le nombre $(3, 2)_{10}$ en représentation positionnelle à virgule en binaire ?

.....

.....

.....

.....

.....

.....

.....

.....

.....

3. (2 points) Comment peut-on représenter le nombre $(3,2)_{10}$ dans le format flottant binaire décrit ci-dessus ? Vous effectuerez un arrondi au plus proche, et exprimerez votre résultat sous la forme d'un code **hexadécimal**.

4 Questions de cours (5 pts)

1. (2 points) Décrivez brièvement le modèle dit « de von Neumann ».

2. (1 point) On a vu en cours une définition possible pour le codage des entiers relatifs en complément à 2 sur p bits : rappelez cette définition.

3. (1 point) Définissez ce qu'est un multiplexeur 2 vers 1.