

# CC-TD-M – lundi 15/11/2010 – LIF6 Architecture matérielle et logicielle

Mioara Joldes – Nicolas Louvet

Aucun document autorisé. Les calculatrices, ordinateurs, téléphones et autres sont interdits.

Durée 1h30. Le barème est susceptible de modifications.

## I Langage machine

Un processeur 8 bits est doté d'un espace mémoire de 56 Kio ; à chaque adresse en mémoire centrale correspond une case de 1 octet. Le processeur dispose en outre d'un registre de travail sur 8 bits nommé ACC (pour « accumulateur »), permettant le stockage temporaire du résultat des opérations en arithmétique entière. Ses instructions sont de longueur fixe. Parmi les instructions figurent les suivantes (@ désigne une adresse en mémoire centrale) :

Instruction	Codage (héxa.)	Fonction réalisée
load @	A0@	Charge le contenu de la case mémoire d'adresse @ dans ACC
store @	A1@	Stocke le contenu du registre ACC dans la case mémoire d'adresse @
add @	B0@	Ajoute à ACC le contenu de la case mémoire d'adresse @, et place le résultat dans ACC. Les nombres sont manipulés comme des entiers naturels
mul @	B2@	Multiplie ACC par le contenu de la case mémoire d'adresse @, et place le résultat dans ACC.

**Q.I.1)** - Sur l'ordinateur considéré, combien d'octets sont nécessaires pour représenter chaque adresse de la mémoire centrale ? Justifiez votre réponse.

**Q.I.2)** - Quelle est la taille en octets du compteur ordinal C0 sur cet ordinateur ?

**Q.I.3)** - Combien d'octets occupe chacune des instructions décrites ? Justifiez votre réponse.

**Q.I.4)** - Écrire en assembleur un morceau de programme qui évalue l'expression suivante, en respectant le parenthésage :  $(a \times x + b) \times (c \times y + d)$ , puis range le résultat à l'adresse  $(0130)_H$ . On suppose l'association suivante entre les cases mémoires et les variables utilisées dans l'expression précédente.

case mémoire d'adresse	0131 <sub>H</sub>	0132 <sub>H</sub>	0133 <sub>H</sub>	0134 <sub>H</sub>	0135 <sub>H</sub>	0136 <sub>H</sub>
variable	$x$	$y$	$a$	$b$	$c$	$d$

Vous ne devez pas utiliser de case mémoire supplémentaire.

**Q.I.5)** - Donnez le codage en mémoire centrale des quatre premières instructions du programme de la question précédente, en supposant que la première instruction débute à l'adresse  $(0105)_H$ . Indiquez le contenu des cases mémoire en hexadécimal. Comme en TD, vous utiliserez la convention « gros-boutiste » pour le stockage des octets.

**Q.I.6)** - Représentez, après chaque instruction du programme, le contenu de C0, de ACC et de la case d'adresse  $(0132)_H$ . On suppose les affectations initiales suivantes pour les différentes variables

case mémoire d'adresse	0131 <sub>H</sub>	0132 <sub>H</sub>	0133 <sub>H</sub>	0134 <sub>H</sub>	0135 <sub>H</sub>	0136 <sub>H</sub>
variable	$x$	$y$	$a$	$b$	$c$	$d$
valeurs	$(4)_{10}$	$(8)_{10}$	$(4)_{10}$	$(36)_{10}$	$(53)_{10}$	$(17)_{10}$

Indiquez le contenu de C0 en hexadécimal, et ceux de ACC et de la case mémoire d'adresse  $(0130)_H$  en décimal. Donnez votre résultat sous forme d'un tableau.

## II Représentation à virgule flottante

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

$$c = \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|} \hline 11 & 10 & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ \hline s & & e_c & & & & & & m_c & & & \\ \hline \end{array}$$

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).

- Q.II.1)** - Quel nombre est représenté par le code octal  $c = 7045_8$ , si on le regarde comme le codage d'un flottant ?  
Donnez votre résultat sous la forme d'une représentation positionnelle à virgule en décimal.
- Q.II.2)** - Comment est représenté le nombre  $(0,2)_{10}$  représentation positionnelle à virgule en binaire ? Détaillez un peu vos calculs.
- Q.II.3)** - Comment peut-on représenter le nombre  $(0,2)_{10}$  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.

### III Imparité

On considère la fonction  $f : \mathbb{B}^3 \rightarrow \mathbb{B}$  définie de la manière suivante :

- $f(x_2, x_1, x_0) = 1$  si le mot  $x_2x_1x_0$  comprend un nombre impair de bits à 1,
- $f(x_2, x_1, x_0) = 0$  si le mot  $x_2x_1x_0$  comprend un nombre pair de bits à 1.

La fonction  $f$  est souvent appelée fonction *imparité*.

- Q.III.1)** - Dressez la table de vérité de  $f(x_2, x_1, x_0)$  en fonction de  $x_2, x_1, x_0$ .
- Q.III.2)** - Donnez une expression booléenne sous forme normale disjonctive pour  $f$ .
- Q.III.3)** - Exprimez  $f$  uniquement à l'aide de l'opérateur XOR (noté  $\oplus$ ).
- Q.III.4)** - Donnez un logigramme pour  $f$  en utilisant uniquement des portes XOR.

### IV Mémoire cache

On considère le programme C suivant :

```
int main(void) {
    int A[16], B[16], C[16];
    int i;
    for(i=0; i<16; i++) C[i] = A[i] + B[i];
    ...
}
```

On suppose que l'on compile et que l'on exécute ce programme sur un ordinateur qui ne dispose que d'un seul niveau de cache de données direct : ce cache comporte 4 entrées de 32 octets (on ne se préoccupe pas du chargement des instructions, ni de la variable  $i$  dans cet exercice). On note les lignes de cache  $\ell_0, \ell_1, \ell_2, \dots$ , et les entrées du cache  $e_0, e_1, e_2, e_3$ . Les entiers de type `int` sont d'une taille de 32 bits. Les tableaux sont stockés de manière contiguë en mémoire, dans l'ordre de leur déclaration.

- Q.IV.1)** - Combien d'entiers de type `int` peut contenir une ligne de cache ?
- Q.IV.2)** - Combien de lignes de caches consécutives occupe chacun des tableaux `A`, `B` et `C` ?
- Q.IV.3)** - En supposant que `A[0]` est aligné sur le début de la ligne de cache  $\ell_0$ , indiquez sur un schéma à quelles lignes de cache appartiennent les éléments de `A`, `B` et `C`. Indiquez également sur ce même schéma dans quelles entrées du cache seront rangés ces éléments lorsqu'ils seront accédés.
- Q.IV.4)** - Dans un tableau de la forme indiquée ci-dessous, indiquez quelle ligne de cache contient chaque entrée du cache à la fin de chaque itération de la boucle `for(i=0; i<16; i++) C[i] = A[i] + B[i]`; (Vous placerez un « ? » quand on ne peut pas savoir). Indiquez également le nombre de *cache miss* qui ont eu lieu au cours de l'itération. Quel est le nombre total de *cache miss* ?

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$e_0$																
$e_1$																
$e_2$																
$e_3$																
<i>cache miss</i>																