

CC-TD-M – lundi 21/03/2011 – LIF6 Architecture matérielle et logicielle

Mathilde Noual – Nicolas Louvet

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

Durée 1h30. Le barème est susceptible de modifications.

I Mémoire, transferts et petits calculs... (5 pts)

Justifiez vos réponses en détaillant vos calculs.

I.1 Circuits mémoires

Sur la carte mère d'un certain ordinateur, on trouve :

- un processeur 8 bits, ce qui implique que la mémoire est adressée octet par octet, et que le processeur peut lire et écrire des mots de 8 bits en mémoire centrale ;
- un bus système, dont le bus d'adresses est d'une largeur de 24 bits ;
- un certain nombre de circuits mémoire 8 bits, chacun d'une capacité de 512 Kio.

Q.I.1) - (0.5 pt) Quelle est la largeur, en nombre de bits, du bus de données ?

Q.I.2) - (1 pt) En supposant que le bus d'adresses ne sert à adresser que la mémoire centrale de l'ordinateur, combien de circuits mémoire sont présents sur la carte mère ?

I.2 Bande passante

On souhaite lire un film se trouvant sur un disque dur, connecter à l'ordinateur via un bus USB 1.1. Le film est non-compressé, et constitué d'une succession d'images de 1024×768 pixels en 256 couleurs. On suppose que le défilement des images se fait en 24 images par seconde. Une liaison USB 1.1 permet le transfert de données bit par bit à la fréquence de 12 MHz.

Q.I.3) - (1 pt) Quelle est la taille en octets d'une image du film ? ($768 = 3 \times 256$)

Q.I.4) - (1 pt) Quel est le débit (en Mio/s) requis pour le transfert du film du disque dur à la mémoire vidéo ?

Q.I.5) - (1 pt) Quelle est la bande passante en Mo/s du bus USB 1.1 envisagé pour la lecture du film ?

Q.I.6) - (0.5 pt) La lecture du film sera-t-elle possible ? Expliquez clairement votre réponse.

II Langage machine (9 pts)

Un processeur 8 bits est doté d'un espace mémoire de 64 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 codées sur 1 à 4 octets. 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 @	A2@	Stocke le contenu du registre ACC dans la case mémoire d'adresse @
add @	020@	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

Q.II.1) - (0.5 pt) Sur l'ordinateur considéré, combien d'adresses sont nécessaires pour représenter chaque adresse de la mémoire centrale ? Justifiez brièvement votre réponse.

Q.II.2) - (0.5 pt) Quelle est la taille en octets du compteur de programme PC sur cet ordinateur ?

Q.II.3) - (1 pt) Combien d'octets occupent chacune des instructions décrites ci-dessus ? Justifiez brièvement votre réponse, puis donnez vos résultats sous forme d'un tableau.

Q.II.4) - (1 pt) Écrire en assembleur un morceau de programme qui ajoute le contenu des cases mémoires d'adresses $(0130)_H$ et $(0131)_H$, puis range le résultat à l'adresse $(0132)_H$.

- Q.II.5)** - (2 pt) Représentez en mémoire centrale le morceau de programme précédent, en supposant que la première instruction débute à l'adresse $(0105)_H$. Indiquez le contenu des cases mémoire en hexadécimal.
- Q.II.6)** - (2 pt) Représentez, après chaque instruction du programme, le contenu de PC, de ACC et de la case d'adresse $(0132)_H$. Supposez que les cases mémoires d'adresses $(0130)_H$ et $(0131)_H$ contiennent initialement les valeurs $(88)_H$ et $(05)_H$ respectivement. Indiquez les contenus des registres et de la case mémoire en hexadécimal, et donnez votre résultat sous forme d'un tableau.
- Q.II.7)** - (1 pt) Écrire en assembleur un programme qui échange le contenu des cases mémoire d'adresses $(0130)_H$ et $(0131)_H$.

III Mémoire cache (6 pts)

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 contigüe en mémoire, dans l'ordre de leur déclaration.

- Q.III.1)** - (1 pt) Combien d'entiers de type `int` peut contenir une ligne de cache ?
- Q.III.2)** - (1 pt) Combien de lignes de caches consécutives occupe chacun des tableaux `A`, `B` et `C` ?
- Q.III.3)** - (2 pts) En supposant que `A[0]` est aligné sur le début de la ligne de cache ℓ_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.III.4)** - (2 pts) 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.

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>																