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 ?

8 bits.

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 ?

2^{24} adresses $\times$ 1o/adresse $= 2^{24}$ o.
2^{24} o/$(2^9 \cdot 2^{10})$ o $= 2^5 = 32$ circuits mémoire.

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 le taille en octets d'une image du film ? ($768 = 3 \times 256$)

Pour coder 256 couleurs, il faut 8 bits par pixel. L'image comporte $2^{10} \times 3 \times 2^8 = 3 \times 2^{18}$ pixels, d'où 3×2^{18} o par image.

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

Comme il faut transférer 24 images par seconde, le débit est de $3 \times 2^{18} \times 24$ o/s $= 3 \times 2^{18} \times 3 \times 2^3$ o/s $= 9 \times 2^{21}$ o/s $= 18$ Mio/s.

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

1 bit $\times$ $12 \cdot 10^6$/s $= 12/8 \times 10^6$ octet/s $= 1,5$ Mo/s

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

1.5 Mo/s $<$ 18 Mio/s

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. Parmis les instructions figurent les suivantes (`@` désigne une adresse en mémoire centrale) :

Instruction	Codage (héxa.)	Fonction réalisée
`load @`	*A*0@	Charge le contenu de la case mémoire d'adresse `@` dans `ACC`
`store @`	*A*2@	Stocke le contenu du registre `ACC` dans la case mémoire d'adresse `@`
`add @`	0206@	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.

La capacité de la mémoire est de 64 Kio = $2^6 \times 2^{10}$ o = 2^{16} o : à raison d'un octet par adresse, cela donne 2^{16} adresses, donc 16 bits par adresses, ou 2 octets. On supposera donc par la suite que les adresses sont codées sur 2 octets.

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

L'UCT doit pouvoir aller chercher ses instructions dans une mémoire dont les adresses sont codées sur 2 octets : on peut donc supposer que la taille de `PC` est de de 2 octets.

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

On détermine le nombre d'octet nécessaire pour le codage du code-op, d'après les codes en hexadécimal de l'énoncé. Comme chaque instruction prend comme opérande une adresse, on ajoute ensuite 2 octets.

Instruction	Codage (héxa.)	Taille
`load @`	*A*0@	3 o
`store @`	*A*2@	3 o
`add @`	0206@	4 o

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 $(0132)_H$.

`load` 0130_H
`add` 0131_H
`store` 0132_H

Q.II.5) - (2 pt) Représentez en mémoire centrale le morceau de programme précédent, en supposant que le première instruction débute à l'adresse $(0105)_H$. Indiquez le contenu des cases mémoire en hexadécimal.

Instruction	Adresse	Contenu
`load` 0130_H	0105_H	*A*0
	0106_H	01
	0107_H	30
`add` 0131_H	0108_H	02
	0109_H	06
	$010A_H$	01
	$010B_H$	31
`store` 0132_H	$010C_H$	*A*2
	$010D_H$	01
	$010E_H$	32

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 le 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.

Instant	contenu de `PC`	contenu de `ACC`	contenu de la case d'adresse $(0132)_H$
après `load` 0130_H	0108_H	88_H	inchangé
après `add` 0131_H	$010C_H$	05_H	inchangé
après `store` 0132_H	$010F_H$	inchangé	$8D_H$

Q.II.7) - (1 pt) Écrire en assembleur un programme qui échange le contenu des cases mémoire d'adresses $(0130)_H$ et $(0131)_H$.

Il est nécessaire d'utilisez une case mémoire pour effectuer un stockage intermédiaire. Utilisons la case d'adresse $(0132)_H$.

`load` 0130_H
`store` 0132_H `load` 0131_H
`store` 0130_H `load` 0132_H
`store` 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 $\ell_0, \ell_1, \ell_2, \ldots$, 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 ?

Un `int` occupe 4 o. Une ligne de cache de 32 o peut donc contenir 8 int.

Q.III.2) - (1 pt) Combien de lignes de caches consécutives occupe chacun des tableaux `A`, `B` et `C` ?

Chaque ligne de cache compte 8 int, donc un tableau de 16 int occupe 2 lignes de cache consécutives.

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 élément de `A`, `B` et `C`. Indiquez également sur ce même schéma dans quelles entrées du cache seront rangés ces élements lorsqu'ils seront accédés.

- `A[0....7]` appartiennent à ℓ_0, stockés dans e_0
- `A[8...15]` appartiennent à ℓ_1, stockés dans e_1
- `B[0....7]` appartiennent à ℓ_2, stockés dans e_2
- `B[8...15]` appartiennent à ℓ_3, stockés dans e_3
- `C[0....7]` appartiennent à ℓ_4, stockés dans e_0
- `C[8...15]` appartiennent à ℓ_5, stockés dans e_1

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																
cache miss																

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
e_0	l4	l4	l4	l4	l4	l4	l4	l4	l4	l4	l4	l4	l4	l4	l4	l4
e_1	?	?	?	?	?	?	?	?	l5	l5	l5	l5	l5	l5	l5	l5
e_2	l2	l2	l2	l2	l2	l2	l2	l2	l2	l2	l2	l2	l2	l2	l2	l2
e_3	?	?	?	?	?	?	?	?	l3	l3	l3	l3	l3	l3	l3	l3
cache miss	3	2	2	2	2	2	2	2	3	2	2	2	2	2	2	2