Examen Final

IFT-1215

14 avril 2014

Directives

— Documentation autorisée : une page manuscrite recto.
— Pas de calculatrice, téléphone, ou autre appareil électronique autorisé.
— Répondre sur le questionnaire dans l'espace libre qui suit chaque question. Utiliser le verso de la page si nécessaire.
— Ne faire que 5 des 6 questions (à choix).
— Chaque question vaut 5 points pour un total maximum de 25 points.
— Les questions ne sont pas placées par ordre de difficulté.

0 Nom et prénom (1 point de bonus)

Écrire son nom et prénom et son code permanent en haut de chaque page.

1 Circuits logiques

Mettre au point un compteur sur 2 bits. Le circuit a 2 entrées, une horloge CK et une ligne I, et deux sorties C_0 et C_1. Si l'entrée I a une valeur de 0 lors du coup d'horloge, C_0C_1 ne change pas, et si elle a une valeur de 1 lors du coup d'horloge le nombre binaire 2-bit représenté par C_0C_1 augmente de 1 (avec un *wraparound* de 11 vers 00).

1. Donner la table de vérité de la partie combinatoire du circuit (entrées I, C_0 et C_1 ; et sorties C_0 et C_1).
2. Dessiner le circuit avec ses portes logiques et ses deux registres.

2 LMC

Rappel : l'encodage des instructions LMC est comme suit

HLT	000	ADD	1xx	BR	6xx
IN	901	SUB	2xx	BRZ	7xx
OUT	902	STO	3xx	BRP	8xx
		LDA	5xx		

On veut créer le LMC-16, une version légèrement modernisée du LMC : au lieu de manipuler des mots de 3 chiffres décimaux, le LMC-16 manipule des mots binaires de 16bit.

1. Au lieu d'un chiffre décimal, combien de bits faut-il réserver dans l'encodage des instructions pour encoder le code de chaque instruction (i.e. l'*opcode*).
2. Combien de bits d'adressage peut-on donc utiliser ? Combien d'adresses (ou "tiroirs") le LMC-16 aura-t-il ?
3. Pourrait-on utiliser un bit de moins pour l'*opcode* ?
   Si oui, comment et quels seraient les inconvénients ?
   Si non, pourquoi, et quel serait l'avantage d'utiliser un bit de plus ?

Montrer l'encodage en langage machine LMC-16 (en binaire ou hexadécimal) du code assembleur suivant :

```
loop
        (LDA size)
        (SUB i)
        (BRZ loopDone)
        (BR loop)
loopDone
        (OUT)
        (HLT)

size    (DAT    10)
i       (DAT    0)
```

3 Mémoires caches

Un programme contient deux boucles imbriquées : une petite boucle interne et une boucle externe. La structure générale du programme est :

```
0:      ...
        ...
512:    ...
        ...
767:    BRP 512
        ...
1535:   BRP 0
        ...
2047:   HALT
```

Les adresses ci-dessus sont en décimal. Toutes les instructions sauf deux instructions de branchement (adresses 767 et 1535) sont exécutées successivement en commençant à l'adresse 0. La boucle externe est exécutée 2 fois, et à chaque itération, la boucle interne est exécutée 10 fois. Chaque adresse contient une instruction.

Le programme est exécuté sur un ordinateur avec une mémoire cache d'instructions de type *direct-mapped* (à correspondance directe) avec les paramètres suivants :

Taille de la mémoire totale	16 millions de mots
Taille de la mémoire cache	1024 mots
Taille de bloc (i.e. ligne de cache)	32 mots
Temps d'accès à la mémoire principale	100 ns
Temps d'accès au cache	1 ns

1. Déterminer le nombre de bits dans les champs ÉTIQUETTE, INDEX et OFFSET de l'adresse de la mémoire principale.
2. Calculer le temps total nécessaire pour la recherche (fetch) des instructions pendant l'exécution du programme.

   Le faire morceau par morceau : 0-511 au premier passage, 512-767 au premier passage, 512-767 pendant les 9 passages suivants, ...

4 Performance des disques

Soit un disque dur divisé en 1000 secteurs, 50000 cylindres et 6 plateaux et où chaque bloc contient 512 bytes. Le disque tourne à une vitesse de 5400tpm et son temps moyen de recherche (*seek time*) est de 10ms.

1. Quelle est la capacité totale du disque ?
2. Quel est le taux de transfert maximum en megabytes par seconde (MB/s) ?
3. Quelle est la latence minimum et maximum ? Latence moyenne ?
4. Soit un processus P1 qui lit séquentiellement un fichier de 100MB placé de manière optimale sur le disque. Combien de temps faudra-t-il approximativement pour lire tout le fichier ?
5. Soit un processus P2 qui lit un autre fichier dont les blocs, au lieu d'être bien contigus, sont répartis de manière complètement aléatoire sur l'ensemble du disque. Combien de temps faudra-t-il approximativement pour lire tout le fichier ?
6. Imaginons que le disque contient un cache et chaque fois qu'un bloc est lu, tous les blocs de la *piste* correspondante sont placés dans le cache. De plus, pendant que le disque est en train de chercher un bloc sur le disque, il peut répondre à d'autres requêtes (à conditions que le bloc correspondant soit dans le cache). Si P1 et P2 sont lancés au même moment, combien de temps leur faudra-t-il approximativement pour lire leur fichier respectif ?

5 Ordonnancement

Soit un système d'exploitation de style Unix utilisé pour une ordinateur de bureau. L'ordonnanceur utilise un système de priorités dynamiques, où chaque niveau de priorité a sa propre queue. Chaque niveau de priorité utilise un quantum de temps qui est le double de celui de la priorité antérieure et le noyau change la priorité des processus en observant leur comportement.

1. Donner les 2 plus importants critères de performance que l'ordonnanceur aimerait optimiser.
2. Quand l'ordonnanceur diminue-t-il la priorité d'un processus ?
   Quand l'ordonnanceur augmente-t-il la priorité d'un processus ?
3. Si tout se passe bien, quel genre de processus devrait obtenir une priorité élevée, et quel genre obtiendra une priorité basse ?
4. Soit un processus de simulation de qualité de l'air qui prend beaucoup de temps et de mémoire, et que l'utilisateur laisse tourner pendant qu'il développe frénétiquement la version suivante dans son éditeur préféré (Emacs, bien sûr). Ce processus de simulation utilise (très activement) deux fois plus de mémoire virtuelle que la mémoire physique disponible. Quel niveau de priorité le noyau va-t-il donner à ce processus et pourquoi ?
5. On ajoute un concept de *priorité statique souple* qui ne fixe pas un processus dans un niveau de priorité, mais qui doit influencer l'algorithme de priorité dynamique pour augmenter/diminuer la priorité dynamique que le noyau affectera au processus.
   Comment le noyau peut-il utiliser une telle priorité statique pour qu'elle aie l'effet désiré ?

6 Mémoire virtuelle

Soit un CPU avec un adressage (logique et physique) sur 32bits avec des pages de 4KB. La table des pages est organisée en hiérarchie de sous-tables, ou chaque sous-table tient dans une case (*frame*) de 4KB. Chaque entrée de la table occupe 32bits qui combinent le numéro de case avec quelques bits qui contrôlent les permissions d'accès.

1. Combien de niveaux y a-t-il dans cette table des pages hiérarchiques?
2. Mentionner 1 avantage et 1 inconvénient de l'utilisation de tables hiérarchiques plutôt qu'une table à un seul niveau.
3. Décomposer les 32bits d'une adresse pour montrer quels bits sont utilisés pour le déplacement (*offset*) et pour l'index de chaque niveau.
4. Décrire de manière concise et précise (i.e. une formule mathématique ou du pseudo-code) comment obtenir l'adresse physique qui correspond à une adresse logique. Utiliser par exemple une notation comme `mem[N]` pour lire le mot à l'adresse N et `M.case` pour extraire la partie "numéro de case" d'un mot N qui contient une entrée de table des pages.
5. Dans quelles conditions est-il préférable de lire la table des pages sans passer par le cache?