

INF1600 — TP1

Architecture du processeur

Giovanni Beltrame `giovanni.beltrame@polymtl.ca`
Erica Bugden `erica-2.bugden@polymtl.ca`

Polytechnique Montréal
Hiver 2017

Introduction

Vous commencez aujourd'hui le premier travail pratique sommatif du cours INF1600. Celui-ci vous permettra de réutiliser une partie de la matière théorique vue en INF1500 dans celle de ce cours d'architecture des micro-ordinateurs. Vous effectuerez également une série d'exercices sur l'architecture d'un processeur hypothétique. Enfin, ce TP vous permettra de vous familiariser davantage avec le logiciel Electric.

Remise

Voici les détails concernant la remise de ce travail pratique :

- **Méthode** : sur Moodle, une seule remise par équipe.
- **Echéance** :

Groupes B1	7 & 8 Février
Groupes B2	14 & 15 Février
- **Format** : un seul rapport PDF. Incluez les captures d'écran directement dans le rapport—les fichiers à l'extérieur de ce rapport ne seront pas corrigés. Incluez également une page titre où doivent figurer les noms et matricules des deux membres de l'équipe, votre groupe de laboratoire, le nom et le sigle du cours, la date de remise et le nom de l'École.
- **Langue écrite** : Français.
- **Distribution** : les deux membres de l'équipe recevront la même note.

Attention ! L'équipe de deux que vous formez pour ce TP sera définitive jusqu'au TP5. Il ne sera pas possible de changer d'équipe au cours de la session.

Barème

Contenu	Points du cours
Exercice 1	1,5
Exercice 2	1,5
Exercice 3	1
Exercice 4	2
Français écrit erroné	jusqu'à -0,6
Format de remise erroné (irrespect de l'arborescence ou des noms de fichiers demandés, fichiers superflus, mauvaise page de titre, etc.)	jusqu'à -0,5
Retard	-0,025 par heure

Exercice 1 Révision de logique et arithmétique numérique

Avant de vous lancer dans les exercices concernant l'architecture, on vous présente un échauffement vous permettant de vous (re)mettre à l'aise avec quelques notions élémentaires de logique et d'arithmétique numérique. Vous serez ensuite appelé à bien comprendre la notation RTN, notation qui est cruciale à la réussite de l'exercice 5, soit la simulation d'un processeur.

1. Pour les nombres entiers signés suivants, représentés en complément à deux sur le nombre de bits affichés, donnez la valeur décimale.

- (a) 11110101 (binaire)
- (b) 00101010 (binaire)
- (c) 4517 (octal)
- (d) CAFE (hexadécimal, 16 bits)
- (e) 10000000 (binaire)

2. Pour chacun des 4 numéros ci-dessous (on ne connaît pas la base de la représentation a-priori), indiquez les bases possibles :

ID	Numéros	BIN	OCT	DEC	HEX
(a)	5781				
(b)	10000000				
(c)	1600				
(d)	B747				
(e)	00000000				

3. Expliquez dans vos mots ce que fait la ligne suivante en langage C.

```
y = x & (3 << 4);
```

4. Pour les nombres entiers suivants écrits en base décimale, donnez la représentation au format binaire signé (complément à deux) 16-bit et en hexadécimal 16-bit.

- (a) -1234
- (b) 48879
- (c) -32

5. Effectuez les opérations arithmétiques suivantes sur 8 bits et indiquez s'il y a un débordement signé, en plus de fournir le résultat au format hexadécimal.

(a) $7C + 4F$ (b) $89 + 11$

6. En informatique, on est souvent confronté à utiliser une organisation différente des nombres multi-octets. Les deux façons courantes d'organiser les octets sont big-endian (par exemple Sun SPARC, Apple) et little-endian (x86).

Un nombre entier non signé de la longueur de 4 octets (par exemple, l'adresse du premier secteur d'une partition) est stocké dans les octets oc_2, oc_3, oc_4, oc_5 (remarque : le premier octet est oc_0) de la séquence d'octets suivante :

0861 C2BB 38A0 9EEC

Quelle est la valeur décimale de l'entier non signé en (a) big-endian et (b) little-endian ?

Exercice 2 Disque dur

Considérez un disque dur avec les caractéristiques suivantes :

Vitesse de rotation		5400 RPM
Zone	Pistes/Zone	Secteurs/Piste
1	624	792
2	1424	780
3	1680	760
4	1815	720

- a Calculez l'espace total sur le disque dur (512 B par secteur).
- b Calculez le taux de lecture moyenne.
- c Calculez le taux de lecture moyenne effective si le disque dure est connecté avec un bus PCIe de vitesse 4000 Mb/s.
- d Changeriez vous les résultats précédents si l'information sur le nombre de surfaces était disponible ?

Exercice 3 Description RTN

La notation RTN (*Register Transfer Notation*) est très importante en architecture de processeurs. Elle permet de décrire, dans un langage universel, les instructions et les opérations permettant ces instructions. On la retrouve, habituellement sous la forme enseignée en INF1600, dans la majorité des manuels de référence de microprocesseurs. Voyez, par exemple, cet extrait du manuel du Pentium d'Intel (instruction AAM) :

```
AAM - ASCII Adjust AX after Multiply Operation

regAL <- AL;
AH <- regAL / imm8;
AL <- regAL MOD imm8;

NOTE: imm8 has the value of the instruction's second byte.
```

Le langage RTN vient en deux saveurs : abstrait et concret. Vous aurez la possibilité d'écrire du RTN concret dans l'exercice 5. Dans cet exercice, vous devez plutôt traduire en RTN abstrait certaines instructions d'un processeur fictif. Notez les définitions suivantes :

```
op<4..0> := IR<31..27>
a<4..0> := IR<26..22>
b<4..0> := IR<21..17>
c<4..0> := IR<16..12>
k<16..0> := IR<16..0>
```

Ici, ceci signifie par exemple que la 'variable' **a** correspond à la plage des bits 26 à 22 (inclusivement) du registre d'instruction IR. On peut donc remplacer IR<26..22> par **a**, tout simplement. La plage **op** correspond au code d'opération.

Écrivez en notation RTN abstraite les instructions hypothétiques suivantes :

1. SUBMUL Ra, Rb, k

Cette instruction soustrait le contenu du registre numéro **b** du registre numéro **a** puis multiplie le tout par huit et met le résultat dans le registre numéro **a**. Préciser la valeur de la constante **k**. Le code d'opération de cette instruction est 5.

2. DECREM Ra, Rb

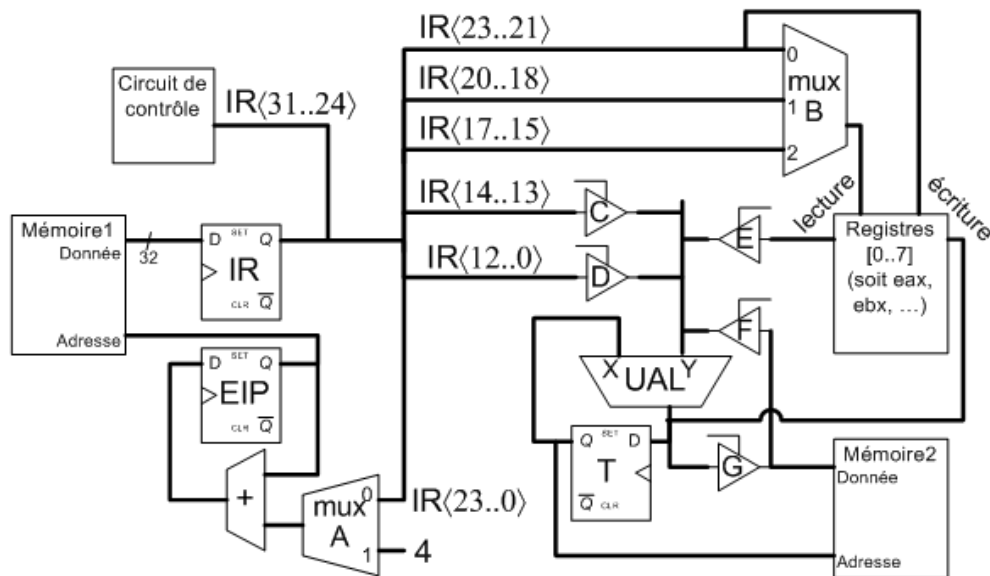
Cette instruction décrémente les registres numéro **a** et **b** et même temps. Le code d'opération de cette instruction est 13.

Voici un exemple pour vous mettre sur la bonne voie. Soit l'instruction ADD Ra, Rb, Rc qui effectue une addition entre les contenus des registres **a** et **b**, puis place le résultat dans le registre numéro **c** ; on suppose que son code d'opération est 7. La description RTN abstraite est :

```
(IR<31..27> = 7) -> R[IR<16..12>] <- (R[IR<26..22>] + R[IR<21..17>]);
```

Exercice 4 Architecture d'un microprocesseur

Soit la microarchitecture du processeur 32-bit simplifié suivant :



Les signaux de contrôle entre le circuit de contrôle et le reste ne sont pas indiqués, mais ils sont présents.

Le circuit de contrôle reçoit uniquement les bits 31 à 24 de l'instruction pour choisir quels signaux envoyer aux autres unités. Autrement dit, une seule instruction de haut-niveau donne lieu à une séquence de macro-instructions qui s'exécutent dans le circuit de contrôle.

Les signaux entrant par le haut de la boîte de registres sont en entrée des décodeurs pour sélectionner respectivement le registre en lecture et celui en écriture, tel que vu en cours, alors que les signaux à gauche et à droite sont respectivement la sortie de la valeur lue et l'entrée pour la valeur à écrire.

Le registre T est utilisé comme valeur temporaire pour effectuer des calculs avec l'unité arithmétique, sa sortie étant utilisée comme entrée X de l'UAL. Tel qu'avec l'architecture IA-32, le registre IR est le registre d'instructions et le registre EIP est le pointeur d'instruction.

L'opération effectuée par l'UAL est déterminée par le signal de contrôle qui lui est envoyé. L'UAL permet entre autre les opérations suivantes :

Valeur du signal de contrôle	Résultat de l'UAL
0x10	$X \ll Y$ (décalage vers la gauche du registre X par Y bits)
0x11	$X \gg Y$ (décalage vers la droite du registre X par Y bits)
0x0a	Y
0x4a	$X + Y$

Une réalisation complète (excluant le circuit de contrôle) de ce processeur, au niveau ‘portes logiques’, a été faite. Dans cet exercice, vous utiliserez le logiciel *Electric* pour simuler le comportement de ce processeur. Autrement dit, vous êtes le circuit de contrôle.

Lancement d'Electric

Suivez bien les prochaines étapes afin d'éviter tout problème de simulation. Depuis un terminal, créez un sous-répertoire dans votre répertoire personnel :

```
$ cd ~
$ mkdir inf1600_tp1
```

Depuis Moodle, téléchargez les fichiers nécessaires dans ce dossier nouvellement créé : `electric-8.05frb.jar`, `inf1600_tp1.zip` et `inf1600_tp1_config.zip`.

Dans le terminal précédemment utilisé, tapez la commande `ls` : vous devriez voir les 3 fichiers téléchargés :

```
$ cd inf1600_tp1
$ ls
```

Décompressez l'archive du TP :

```
$ unzip inf1600_tp1.zip
```

Décompressez également le fichier `inf1600_tp1_config.zip` dans la racine de votre répertoire utilisateur (ceci permettra que les signaux requis s'affichent automatiquement plutôt que vous ayez à les choisir vous-même) :

```
$ cd ~
$ unzip -o inf1600_tp1/inf1600_tp1_config.zip
```

Revenez dans le répertoire créé et lancez Electric (on vous conseille de les réécrire parce-que, parfois, la copie d'un document PDF au terminal ne fonctionne pas parfaitement) :

```
$ cd ~/inf1600_tp1
$ java -jar electric-8.05frb.jar &
```

Le simulateur d'Electric cherchera alors les fichiers `tp1mem1.txt` et `tp1mem2.txt` dans le répertoire courant. Sélectionnez le menu *File > Open Library...* (ou CTRL+O), puis ouvrez `testLogique.jelib`. Dans l'onglet *Explorer* de la fenêtre, choisissez `arch/archsch`. Pour voir le contenu d'un bloc, sélectionnez-le et appuyez sur CTRL+D (pour down). Pour en ressortir, appuyez sur CTRL+U (pour up). Vous pouvez ainsi naviguer dans l'architecture afin de bien comprendre son fonctionnement.

Sachez que cette architecture que vous voyez dans Electric est exactement la même que le schéma ci-haut. Il est conseillé de plutôt vous fier à ce dernier puisqu'il est plus simple, ne contenant pas tous les artéfacts du logiciel Electric.

Simulation d'une instruction avec Electric

Afin de simuler une instruction de haut-niveau avec Electric, suivez ces étapes :

1. Ecrivez votre instruction (le code hexadécimal sur 32 bits à l'adresse 0) dans le fichier `tp1mem1.txt`, qui contient la mémoire d'instructions, en tenant bien compte qu'il s'agit d'une architecture en *little-endian* ;
2. Suivez les étapes ci-haut afin de lancer Electric (Electric doit être relancé chaque fois que le fichier `tp1mem1.txt` est modifié) ;
3. Lancez une simulation en sélectionnant le menu *Tool > Simulation (Built-in) > ALS : Simulate current cell* ;
4. Dans la nouvelle fenêtre, sélectionnez le menu *Tool > Simulation (Built-in) > Restore Stimuli from Disk...* et choisissez le fichier `arch.vec` ;
5. Assurez-vous que votre instruction est bel et bien chargée dans le registre IR environ à 25 ns et que sa valeur est bonne ; dans le cas contraire, fermez Electric, modifiez `tp1mem1.txt` et refaites les étapes 1 à 5 ;

6. Déplacez le *main cursor* (longs traits verticaux) environ à la marque des 50 ns et, pour chaque signal de contrôle à modifier, appuyez sur G ou V pour modifier leur valeur ; modifiez ensuite les signaux au cycle suivant en alignant le curseur au prochain front descendant d'horloge, jusqu'à ce que le résultat attendu soit obtenu.

Notes à considérer

- Si vous utilisez un environnement *GNOME*, il se peut que, lorsqu'une fenêtre d'Electric est maximisée, les menus ne soient pas accessibles facilement ; minimisez alors la fenêtre et agrandissez-la manuellement sans la maximiser ;
- Utilisez *gedit* plutôt que *Kate* (tapez en console) ;
- Ne modifiez pas le fichier `tp1mem2.txt` ; celui-ci contient le contenu de la mémoire de données (Mémoire2) et il est déjà fixé ;
- Les combinaisons CTRL+4, CTRL+6, CTRL+2 et CTRL+8 du clavier numérique peuvent être utiles pour vous déplacer à l'intérieur de la fenêtre de simulation ; de même, CTRL+0 et CTRL+7 servent à zoomer en arrière et en avant ;
- les valeurs initiales de `r1` (ECX), `r2` (EDX) et `r3` (EBX) sont respectivement `0x137dd`, `0xb` et `0x1f` ;
- Lorsque vous prenez des captures d'écran, assurez-vous que le correcteur puisse voir facilement la valeur finale dans le registre affecté par l'instruction simulée ainsi que les différentes étapes des macro-instructions ; au besoin, effectuez un zoom avant et prenez plusieurs captures d'écran clairement identifiées en vous déplaçant chaque fois de gauche à droite ; des captures d'écran incompréhensibles apporteront le résultat 0 pour l'exercice.

Questions

1. Soit l'instruction :

```
r1 <- Memoire2 [r3] - r3
```

- a) Écrivez un encodage possible (en hexadécimal) de l'instruction. Inventez l'opcode au besoin.
- b) Écrivez le RTN concret des macro-instructions permettant d'exécuter l'instruction de haut-niveau avec la microarchitecture proposée.
- c) Sous forme de tableau : pour chaque micro instruction trouvée en b), écrivez la liste des valeurs des signaux de contrôle qui en permettent l'exécution. Les noms des signaux de contrôle sont A, B, C, D, E, F, G, UAL, `ecrireEIP`, `ecrireT` et `ecrireRegistre` (correspondant aux éléments de mêmes noms sur le circuit). À noter que le signal B est sur 2 bits et que l'UAL est sur 8 bits. Vous n'avez pas à effectuer la recherche d'instruction ; supposez qu'elle se trouve déjà dans le registre IR.
- d) Simulez l'instruction avec le logiciel Electric. Faites une ou plusieurs captures d'écran montrant clairement que le résultat de la simulation est correct, en justifiant pourquoi, et joignez-la dans le rapport suite à la question c) précédente.

2. Soit l'instruction :

```
r1 <- (Memoire2 [r3] - 0x23) >> r2
```


- a) Écrivez un encodage possible (en hexadécimal) de l'instruction. Inventez l'opcode au besoin.
- b) Écrivez le RTN concret des macro-instructions permettant d'exécuter l'instruction de haut-niveau avec la microarchitecture proposée.
- c) Sous forme de tableau : pour chaque micro instruction trouvée en b), écrivez la liste des valeurs des signaux de contrôle qui en permettent l'exécution. Les noms des signaux de contrôle sont **A**, **B**, **C**, **D**, **E**, **F**, **G**, **UAL**, **ecrireEIP**, **ecrireT** et **ecrireRegistre** (correspondant aux éléments de mêmes noms sur le circuit). À noter que le signal **B** est sur 2 bits et qu'**UAL** est sur 8 bits. Vous n'avez pas à effectuer la recherche d'instruction ; supposez qu'elle se trouve déjà dans le registre **IR**.
- d) Simulez l'instruction avec le logiciel Electric. Faites une ou plusieurs captures d'écran montrant clairement que le résultat de la simulation est correct, en justifiant pourquoi, et joignez-la dans le rapport suite à la question c) précédente.