INF1600 - TP 1

Samuel Rondeau
Pacôme Bondet

21 octobre 2014

Exercice 1 : révision de logique et arithmétique numérique

Avant de vous lancer dans les exercices concernant l'architecture, on vous présente un premier numéro vous permettant de vous (re)mettre à l'aise avec quelques notions élémentaires de logique et d'arithmétique numérique. Vous êtes 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. $(0110110)_2 = 2^5 + 2^4 + 2^2 + 2^1 = 32 + 16 + 4 + 2 = 54$

b. $(1011\ 1010)_2 = -(0100\ 0110)_2 = -(2^6 + 2^2 + 2^1) = -(64 + 4 + 2) = -70$

c. $(2021)_8 = (010\ 000\ 010\ 001)_2 = 2^{10} + 2^4 + 2^0 = 1024 + 16 + 1 = 1041$

d. $(\text{FACE})_{16} = (1111\ 1010\ 1100\ 1110)_2 = -(0000\ 0101\ 0011\ 0010)_2 = -(2^{10} + 2^8 + 2^5 + 2^4 + 2^2) = -1330$

e. $(1111\ 1111)_2 = -(0000\ 0001)_2 = -2^0 = -1$

2. Parmi les 4 représentations octales ci-dessous, certaines ne sont pas possibles. Lesquelles ?

CA73 et 958

3. Expliquez en vos mots ce que fait la ligne suivante en langage C et à quoi peut-elle servir.

$y = x \ \&\ (7 << 3)$

Cette instruction décale 7 de trois bits par la gauche, et sélectionne dans y les bits de x à la position où se trouvent les 1 dans la résultante de l'opération de décalage, les autres bits sont 0. Il s'agit ici d'un *masque* permettant de sélectionner certains bits d'intérêts de x. Dans le cas où $7 << 3 = (111000)_2$, $y = x_5x_4x_3000$ où x_5, x_4 et x_3 sont à la valeur des bits de position 5, 4 et 3 respectivement (où 0 = LSB) de x.

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

a. $546 = 2^9 + 2^5 + 2^1 = (0000\ 0010\ 0010\ 0010)_2$

b. $-1 = -2^0 = -(0000\ 0000\ 0000\ 0001)_2 = (1111\ 1111\ 1111\ 1111)_2$

c. $2014 = 2^{10} + 2^9 + 2^8 + 2^7 + 2^6 + 2^4 + 2^3 + 2^2 + 2^1 = (0000\ 0111\ 1101\ 1110)_2$

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. 29 + 89

En absence d'indications, assumons que les valeurs sont exprimées en système décimal.

$$\begin{array}{r} 0001\ 1101 \\ +\ 0101\ 1001 \\ \hline 0111\ 0110 \end{array}$$

Résultat = $(76)_{16}$. Il n'y a pas débordement.

Puis maintenant, supposons qu'elles sont en hexadécimal.

$$\begin{array}{r} 0010\ 1001 \\ +\ 1000\ 1001 \\ \hline 1011\ 0010 \end{array}$$

Résultat = $(B2)_{16}$. Il n'y a pas débordement.

b. AB + 63

En absence d'indications, assumons que les valeurs sont exprimées en système hexadécimal.

$$\begin{array}{r} 1010\ 1011 \\ +\ 0110\ 0011 \\ \hline 0000\ 1110 \end{array}$$

Résultat = $(0E)_{16}$. Il y a débordement. Un 1 en retenue ne peut être contenu dans 8 bits. Ainsi, la réponse devrait être $(10E)_{16}$.

Exercice 2 : disque dur

Soit un disque rigide comportant 3 plateaux (6 têtes) et étant divisé en 3 zones ayant chacune 25 000 cylindres. Les zones ont 1024, 512 et 256 secteurs par piste. Chaque secteur contient 512 octets. Le disque tourne à 7200 tours/minute. Les données sont placées dans l'ordre le plus commun (qui est dit vertical), soit selon l'algorithme :

```
Pour chaque cylindre du disque :
    Pour chaque piste du cylindre :
        Passer tous les secteurs d'une piste
```

Répondez aux questions suivantes.

1. Quelle est la capacité totale de ce disque (choisissez l'unité que vous voulez) ?

$$\begin{aligned}\text{Capacité} &= 3\ plateaux \times \frac{2\ surfaces}{plateau} \times \frac{25000\ pistes}{surface} \times \frac{1024\ secteurs}{piste} \times \frac{512\ octets}{secteur} \\ &+ 3\ plateaux \times \frac{2\ surfaces}{plateau} \times \frac{25000\ pistes}{surface} \times \frac{512\ secteurs}{piste} \times \frac{512\ octets}{secteur} \\ &+ 3\ plateaux \times \frac{2\ surfaces}{plateau} \times \frac{25000\ pistes}{surface} \times \frac{256\ secteurs}{piste} \times \frac{512\ octets}{secteur} \\ &= 3 \times 2 \times 25000 \times (1024 + 512 + 256) \times 512 \text{ octets} \\ &= 137\,625\,600\,000 \text{ B} = 134\,400\,000 \text{ kB} = 131\,250 \text{ MB} \approx 128 \text{ GB}\end{aligned}$$

2. Quel est son taux de transfert maximal soutenu en octets/seconde ?

Puisqu'on s'intéresse au taux maximal et non moyen pondéré, on s'intéresse au moment où une tête de lecture se trouve dans une zone de 1024 secteurs par piste. Il s'agit ici du taux pour une zone pour une surface.

$$\begin{aligned}\text{Taux} &= \frac{7200\ tours}{minute} \times \frac{1\ minute}{60\ secondes} \times \frac{1024\ secteurs}{tour} \times \frac{512\ octets}{secteur} \\ &= 7200 \div 60 \times 1024 \times 512 \text{ B/s} \\ &= 62\,914\,560 \text{ B/s}\end{aligned}$$

3. L'algorithme précédent est dit « vertical ». Suggérez un autre algorithme de parcours des données ayant un meilleur taux de transfert maximal soutenu. Quel serait ce taux (en octets/seconde) ?

   Nous proposons une technique qui réduit le nombre de transitions entre les plateaux.

```
Pour chaque surface du disque :
    Pour chaque piste d'une surface :
        Passer tous les secteurs d'une piste
```

Le taux de transfert est le même que l'algorithme vertical lorsqu'il s'agit de lire une piste, mais puisqu'il perd moins de temps sur différents plateaux, il est plus rapide dans l'ensemble. Pour calculer la valeur réelle, nous devons savoir le temps qu'il faut pour passer d'un plateau à l'autre et d'une piste à l'autre.
Cependant, nous pourrions également envisager, en mode lecture, l'activation de toutes les têtes de lecture en même temps.

```
Pour toutes les surfaces du disque :
    Pour chaque cylindre du disque :
        Passer tous les secteurs d'une piste
```

Ce qui donne un taux 6 fois plus rapide, donc $\approx$ 377 487 360 B/s.

Exercice 3 : ordinateur

Cette image représente un « Raspberry Pi », un ordinateur miniature de la taille d'une carte de crédit qui permet d'exécuter un système d'exploitation complet (comme Linux).

Pour en faire un vrai ordinateur utilisable, il manque les périphériques externes (écran, clavier, souris...). Cependant, tous les composants essentiels sont intégrés dans la puce noire au centre du Raspberry Pi, ce qui en fait un système sur puce (system on chip).

a. Quels sont ces composants essentiels ?

Le Raspberry Pi possède un processeur CPU/GPU intégrant la mémoire RAM, des ports entrées/sorties, comme des ports USB, des broches GPIO, et le nécessaire pour l'alimentation, l'affichage, etc. Il contient également un socle pour y insérer une carte SD pour la mémoire. Enfin, le tout est monté sur un circuit imprimé gérant les connexions entre les composants.

b. Sur un ordinateur de bureau, ces composants sont séparés et beaucoup plus gros. Que pouvez-vous dire sur les performances de la Raspberry Pi ?

Il est certain qu'un Raspberry Pi est moins puissant qu'un ordinateur moderne, mais il est pourtant très pratique. Il est très économique, facile à utiliser, permet d'effectuer des tâches conventionnelles, mais peut également exécuter des programmes conçus par son propriétaire. Puisqu'il contient plusieurs entrées et sorties, incluant des broches GPIO, plusieurs projets d'électronique ou de robotique peuvent être conçus en utilisant le Raspberry Pi comme carte mère et ses composants essentiels. Il s'agit donc d'un petit ordinateur parfait pour apprendre et s'amuser à budget.

Le nouveau modèle, Raspberry Pi B+, a tout de même des performances plus que satisfaisante. 512 MB de RAM, 4 ports USB 2.0, 40 broches GPIO, une meilleure consommation électrique, etc. Et avec une distribution Linux comme système d'exploitation qui consomme peu de ressources, on peut sans problème utiliser le Raspberry Pi comme ordinateur pour des utilisations quotidiennes, en plus de ses projets d'électroniques.

Exercice 4 : 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 Multiplicity
> 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 chance 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. Il s'agit du processeur décrit à la diapositive 36 du document 3-Architecture_ part1.pdf. Notez ces définitions supplémentaires :

op<4..0> := IR<31..27> c<4..0> := IR<16..12>
a<4..0> := IR<26..22> k<16..00> := IR<16..0>
b<4..0> := IR<21..17>

Ici, cela 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 éventuelles suivantes :

a. DECRINCR Rb, Ra :

(IR<31..27> = 14) → R[IR<21..17>] ← R[IR<21..17>] - 1 ; R[IR<26..22>] ← R[IR<26..22>] + 1 ;

b. XORk Ra, Rb, Rc, k :

(IR<31..27> = 13) → R[IR<16..12>] ← (R[IR<21..17>] << IR<16..0> ^ R[IR<26..22>])

Exercice 5 : architecture d'un microprocesseur

Soit l'instruction
r1 ← r2 + Mémoire2 [r3 + 0x11]

a. Écrivez un encodage possible (en hexadécimal) de l'instruction. Inventez l'opcode au besoin.

Inventons l'opcode BB. L'instruction est BB 011 001 010 00 0000000010001, donc BB 65 00 11, ou en *little-endian*, 11 00 65 BB.

b. Écrivez le RTN concret des macro-instructions permettant d'exécuter l'instruction de haut-niveau avec la microarchitecture proposée.

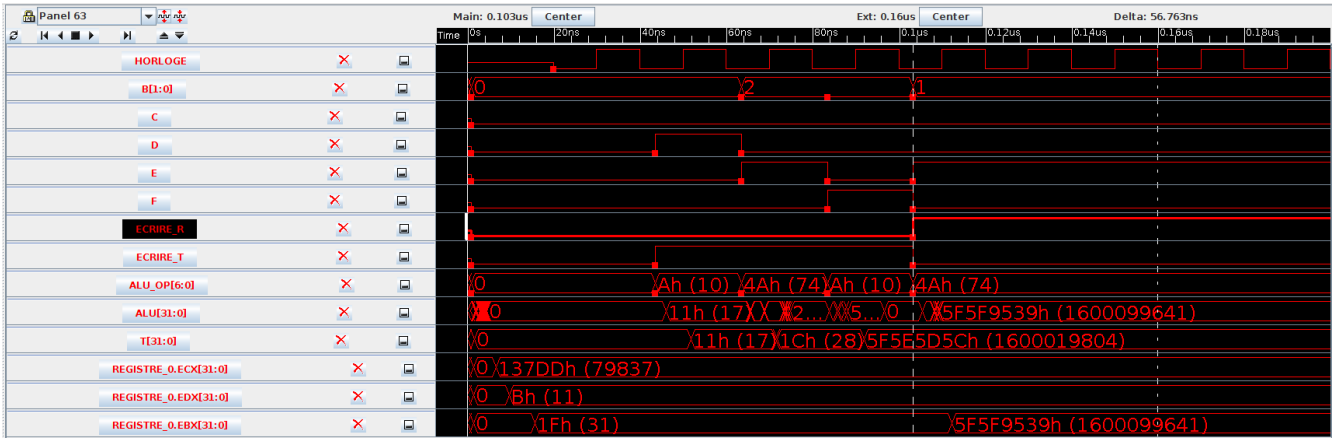
T ← IR<12..0> ;
T ← T + IR<17..15> ;
T ← MEM2[T] ;
R[IR<23..21>] ← T + R[IR<20..18>] ;

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.

Instruction	UAL	B	A	C	D	E	F	G	ecrireEIP	ecrireT	ecrireRegistre
T ← IR<12..0> ;	0x0A	xx	0	0	1	0	0	0	0	1	0
T ← T + IR<17..15> ;	0x4A	10	0	0	0	1	0	0	0	1	0
T ← MEM2[T] ;	0x0A	xx	0	0	0	0	1	0	0	1	0
R[IR<23..21>] ← T + R[IR<20..18>] ;	0x4A	01	0	0	0	1	0	0	0	0	1

d. Les noms des signaux de contrôle sont A, B, C, D, E, F, G, UAL, $ecrireEIP$ et $ecrireRegistre$ (correspondant aux éléments de mêmes noms sur le circuit). À noter que le signal B est sur 2 bits et que 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.

e. 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.



1. Notre algorithme commence à 45 ns. Le contenu de *ecx* (r2) est 79837. Le contenu de *edx* (r3) est 11. L'UAL contient la constante, 17.
2. Dans T se retrouve r3 + 0x11 = 11 + 17 = 28.
3. T vaut ensuite MEM2[28] = 5F5E5D5Ch = 1600019804.
4. Dans r1 on écrit 79837 + 1600019804 = 1600099641.

Soit l'instruction
r2 ← Mémoire2 [r1 + r3] << 0x7

f. Écrivez un encodage possible (en hexadécimal) de l'instruction. Inventez l'opcode au besoin.

Inventons l'opcode FA. L'instruction est FA 001 011 010 00 0000000000111, donc FA 2D 00 07, ou en *little-endian*, 07 00 2D FA.

g. Écrivez le RTN concret des macro-instructions permettant d'exécuter l'instruction de haut-niveau avec la microarchitecture proposée.

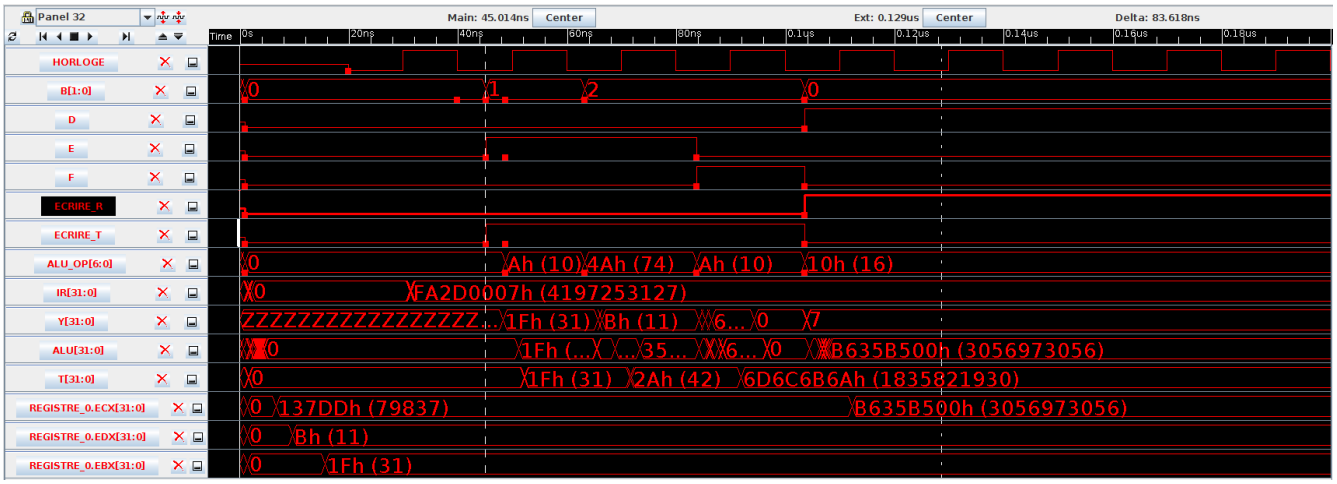
T ← R[IR<20..18>] ;
T ← T + R[IR<17..15>] ;
T ← MEM2[T] ;
R[IR<23..21>] ← T << IR<12..0> ;

h. 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.

Instruction	UAL	B	A	C	D	E	F	G	ecrireEIP	ecrireT	ecrireRegistre
T ← R[IR<20..18>] ;	0x0A	01	0	0	0	1	0	0	0	1	0
T ← T + R[IR<17..15>] ;	0x4A	10	0	0	0	1	0	0	0	1	0
T ← MEM2[T] ;	0x0A	xx	0	0	0	0	1	0	0	1	0
R[IR<23..21>] ← T << IR<12..0> ;	0x10	xx	0	0	1	0	0	0	0	0	1

i. Les noms des signaux de contrôle sont A, B, C, D, E, F, G, UAL, $ecrireEIP$ et $ecrireRegistre$ (correspondant aux éléments de mêmes noms sur le circuit). À noter que le signal B est sur 2 bits et que 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.

j. 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.



1. Notre algorithme commence à 45 ns. Le contenu de *ebx* (r1) est 31. Le contenu de *edx* (r3) est 11. Dans IR se trouve la constante, 7.
2. Dans T se retrouve r1 = 31.
3. T vaut ensuite 31 + r3 = 31 + 11 = 42.
4. T prend ensuite le contenu de MEM2[42] = 6D6B6C6Ah.
5. Dans r2 on écrit 6D6B6C6Ah << 7 = B635B500h, une fois converti en *little-endian*.