



**POLYTECHNIQUE
MONTRÉAL**

INF1600

Architecture de micro-ordinateurs

Laboratoire 2

Équipe 5

Soumis par:
Charles-Etienne Desormeaux - 1956442
Diab Khanafer - 1952548

Groupe: 05 - B1

Le 18 février 2018

Exercise 1

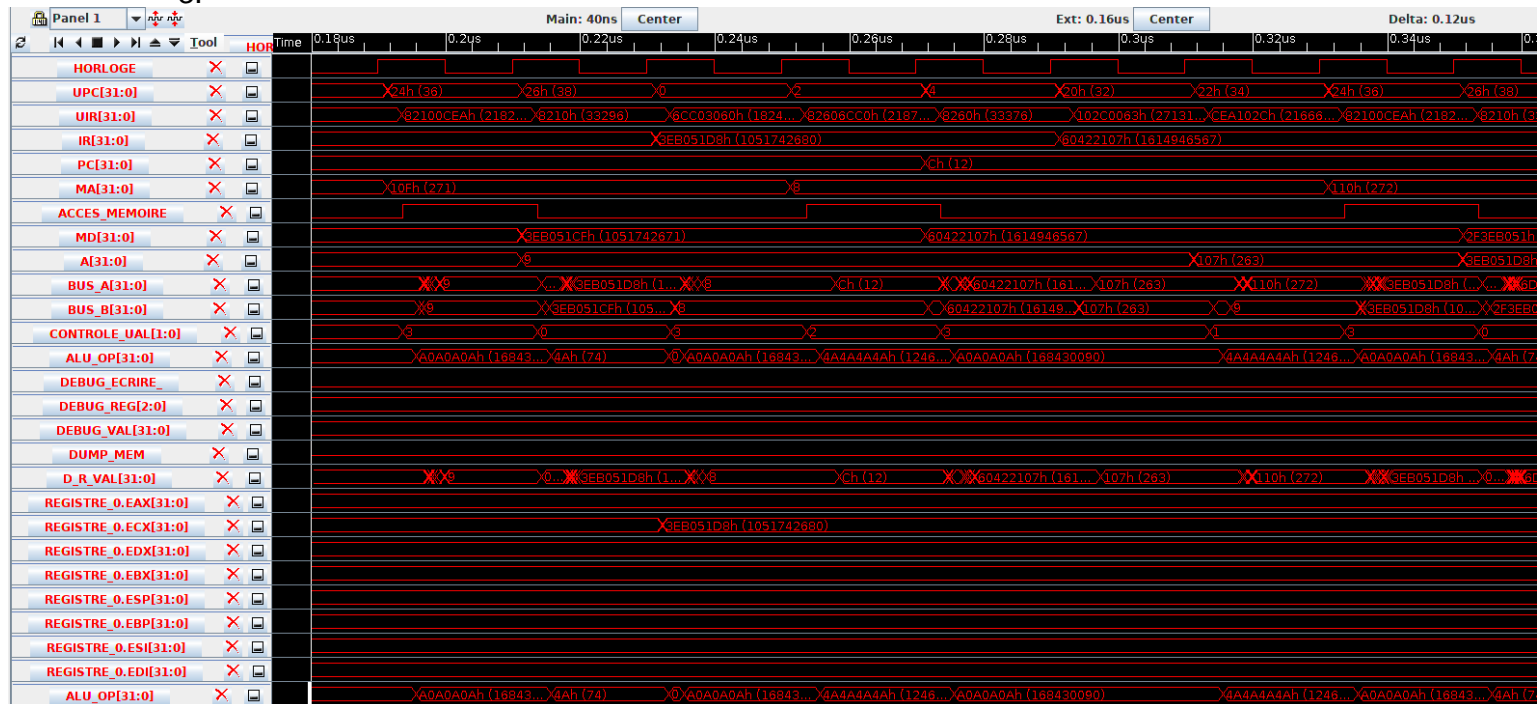
1.

RTN concret	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Hexa
MA <- PC;	0	0	1	1	0	0	0	0	0	1	1	0	0	0	0	0	0x3060
MD <- M[MA]: PC <- PC + 4;	0	1	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0x6CC0
IR <- MD	1	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0x8260

2.

RTN concret	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Hexa
A <- IR<11..0>	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	1	0x0063
MA <- R[IR<16..12>] + A	0	0	0	1	0	0	0	0	0	0	1	0	1	1	0	0	0x102C
MD <- M[MA] : A <- R[IR<21..17>];	0	0	0	0	1	1	0	0	1	1	1	0	1	0	1	0	0x0CEA
R[IR<26..22>] <- A oper MD;	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0x8210

3.



Comme demandé, la capture montre lorsque la valeur de PC est 0xC (12). Nous pouvons voir que la valeur du registre R[1], soit ECX dans la capture, est bel et bien sauvegarder après avoir fait l'additionner.

4.

Tableau de vérité de l'opération NAND

A	B	S
0	0	1
0	1	1
1	0	1
1	1	0

Le op[3:0] du bloc logique [op logique32] a comme ses trois bits de poids faible un 1 et comme bit fort un 0, tel que la table de vérité ci-haut. op[6] doit avoir la valeur de 0 pour que le circuit ne fasse pas l'opération and32, puisque op[6] agit comme «enable» de ce dernier, faisant en sorte que sa sortie soit toujours 0. op[5] a une valeur de 0 pour que add32 n'ait pas de retenue. Finalement, op[4] a une valeur de 0 pour que mux32bit, un multiplexeur, choisisse toujours de prendre en sortie l'opération add[31:0]. Donc, la valeur de op[6:0] est 0000 0111, soit 0x07.

5.

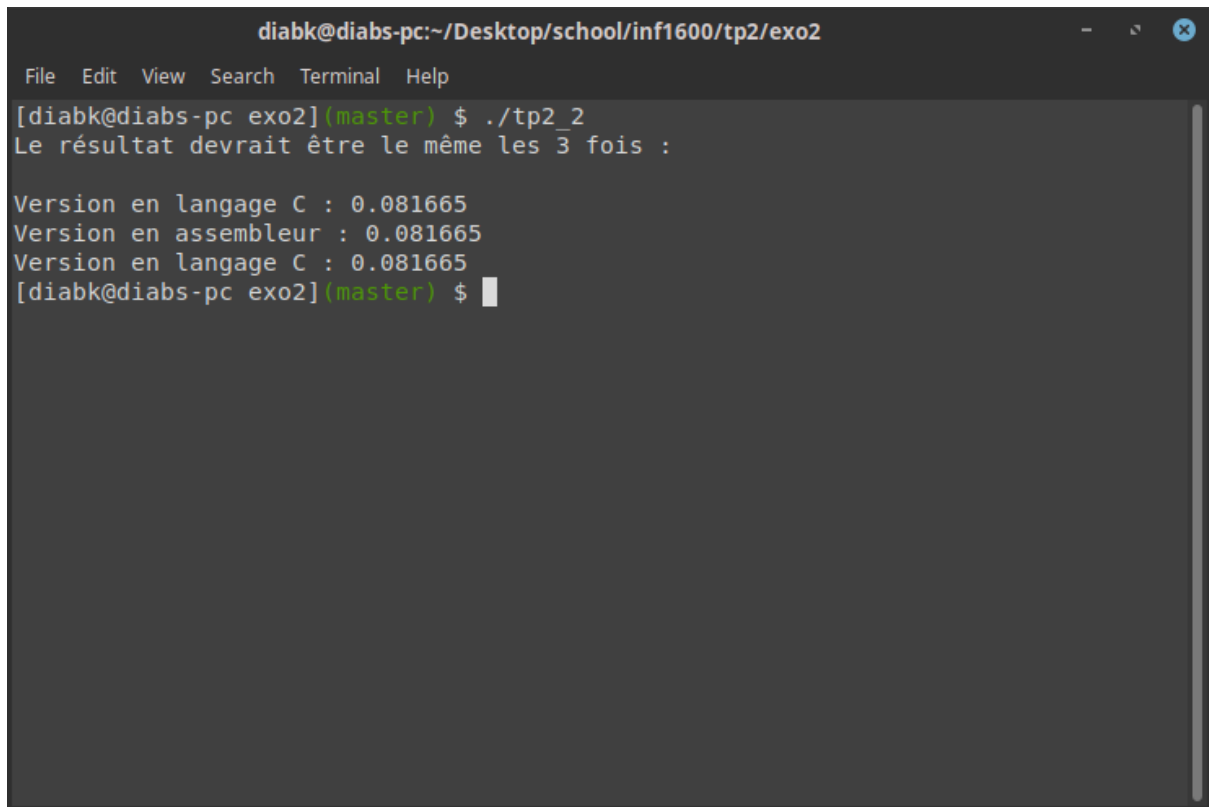
a) $0x55555555 = 0101\ 0101\ 0101\ 0101\ 010101010101$

Les deux derniers octets $0x5555$, soit $0101\ 010101010101$, définissent l'adresse du registre rc et $IR<11..0>$. Une instruction (sur 32 bits) qui aurait exactement le même effet d'exécution pourrait être $0x5000$ puisque ne nous connaissons pas ce que fait le code d'opération, nous pouvons assumer que l'instruction peut être commutative.

b) Avec deux bus, nous serons capables d'effectuer deux opérations par cycle d'horloge. Comme à l'exercice 2, nous avons effectué $MD \leftarrow M[MA]$ et $A \leftarrow R[IR<21..17>]$ sur le même cycle puisque les deux opérations sont faites sur deux bus différents, donc possible d'être fait en même temps. Les sorties se font sur un tandis que les entrées se font sur l'autre.

c) Dans le $tp1$, l'architecture du processeur avait deux emplacements ce qui lui permet des opérations arithmétiques/logiques de façon plus flexible.

Exercice 2



```
diabk@diabs-pc:~/Desktop/school/inf1600/tp2/exo2
File Edit View Search Terminal Help
[diabk@diabs-pc exo2](master) $ ./tp2_2
Le résultat devrait être le même les 3 fois :

Version en langage C : 0.081665
Version en assembleur : 0.081665
Version en langage C : 0.081665
[diabk@diabs-pc exo2](master) $
```

Comme on peut voir dans la capture, nous obtenons les mêmes résultats. Les fichiers sont dans le dossier exo2.

Exercice 3

Tous les fichiers sont dans le dossier exo3.