École Polytechnique de Montréal

Département de génie informatique et génie logiciel

INF1600

Architecture du processeur

Travail pratique #2

Soumis par

Bouis Constantin 1783438
Thimonier Alexandre 1782235

Section de Laboratoire : 02

Date de remise : 1 & 2 Mars
Session Hiver 2017

EXERCICE 1

1/

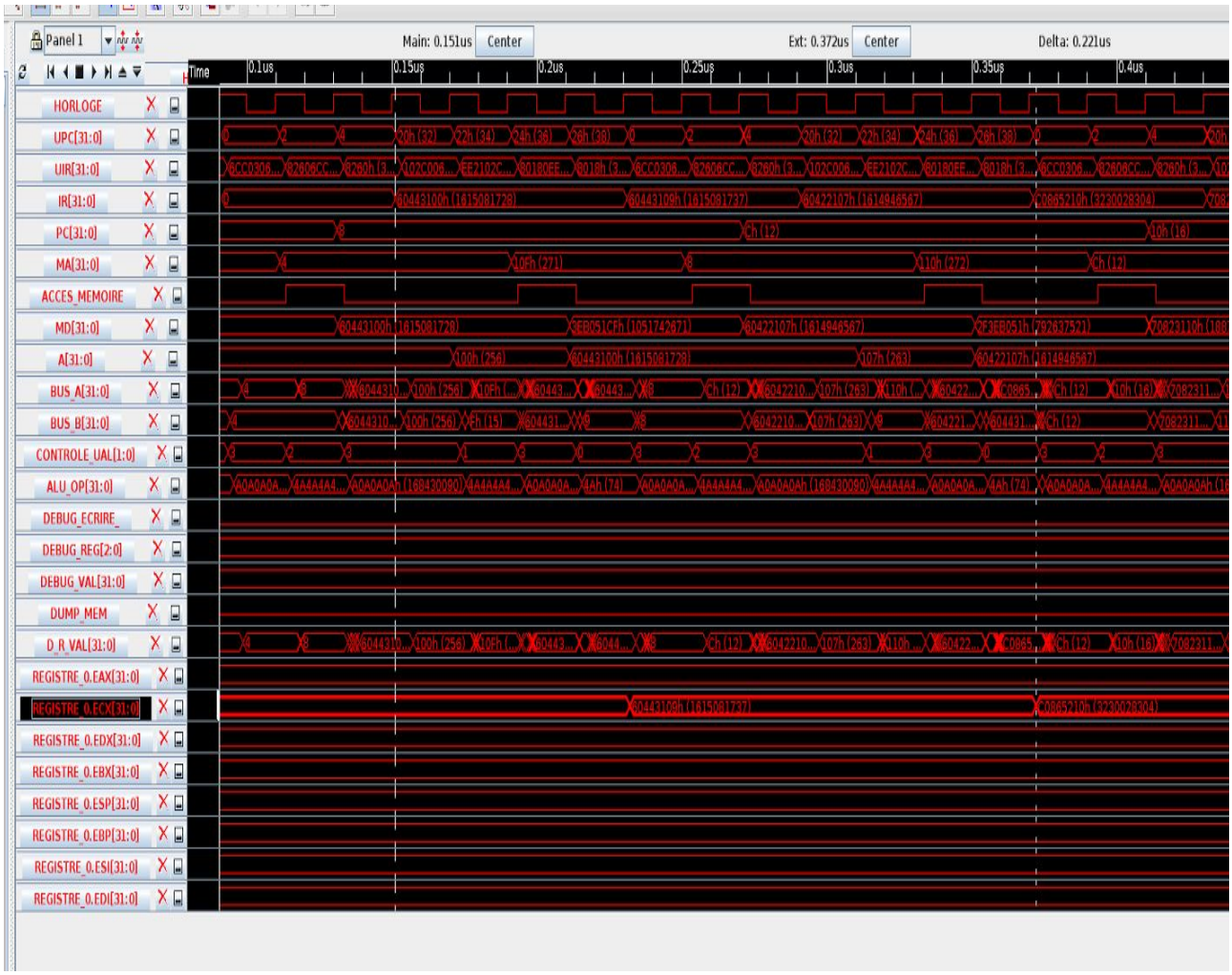
RTN concret	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	HEXA
MA<-PC;	0	0	1	1	0	0	0	0	0	1	1	0	0	0	0	0	0x3060
MD<-M[MA] : PC<-PC+4;	0	1	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0x6CC0
IR<-MD;	1	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0x8260

2/

(IR<31..27> = opcode) ->

R[IR<26..22>] <- R[IR<21..17>] oper M[R[IR<16..12>] + IR <11..0>] ;

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<-A+R[IR<16..12>] ;	0	0	0	1	0	0	0	0	0	0	1	0	1	1	0	0	0x102C
MD <- M[MA]: A<-MD;	0	0	0	0	1	1	1	0	1	1	1	0	0	0	1	0	0x0EE2
R[IR<26..22>] <-R[IR<21..17>] oper A	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0x8018



On peut remarquer dans notre simulation qu'au niveau du main cursor, nous avons la valeur de IR qui a pris la valeur de MD (=0x60443100), ainsi cela signifie que notre premier microprogramme à correctement fonctionné.

De plus, on peut constater qu'au niveau du exit cursor, nous avons la valeur de R[1] (ECX dans Electric) qui a pris la valeur convenable (=0xC0865210). On peut confirmer cela en regardant juste avant ce curseur que le microprogramme s'est rendu jusqu'à l'instruction "MD <- M[MA]: A<-MD;" en voyant que, A a pris la valeur de MD (=0x60422107), et que MD a pris sa valeur dans la mémoire a l'adresse de MA (=8).

4 /

Pour comprendre l'ual, il nous a fallu ouvrir chacun des blocs. On a commencé par l'opLogique32 et on a remarqué qu'il était composé de 32 mux 4 vers 1 , les 4 entrées correspondent à l'op[3:0] et la sélection des voies est faite par les bits An et Bn du n ieme mux. Ainsi il nous a été possible de créer une grande porte NAND seulement avec l'opLogique, nous pouvons utiliser l'opcode 0111 ce qui permettra de sortir la valeur 0 lorsque An et Bn seront à 1 sinon les autres cas donneront 1 : c'est ce qui crée la porte NAND.

En revenant dans le circuit de l'ual il nous faut maintenant transmettre les 32 bits de l'opLogique32 vers la sortie, pour cela nous utilisons seulement la voie du haut (du mux), l'op[4] du mux sera à 0. Ainsi l'op[0] du shift32 peut prendre n'importe quelle valeur(celle ci ne sera de toute facon pas selectionnée par le mux).
Il nous faut alors additionner à la sortie de l'opLogique la valeur 0 en mettant l'op[6] à la valeur 0, nous créons un bus de 32bits rempli de 0. Pour l'add32 nous choisissons d'utiliser la fonction d'addition en prenant l'op[5] = 0 . Nous venons de transmettre la sortie de l'opLogique vers la sortie créant une porte NAND.
l'opcode final sera donc 0000111 .

5/

a) Chaque 16bit donne des signaux de contrôle d'un cycle de processeur, donc 0x5555 donne des signaux de contrôle au processeur. Une autre instruction qui aurait le même effet serait : 0x12345555

b) Un avantage d'une architecture à 2 bus est que l'on peut gagner des cycles en utilisant les deux bus en même temps pour des instructions différentes sans qu'il n ' y ait de court circuit. Nous ne nous sommes pas servi de cet avantage dans notre microprogramme car les instructions ne nous l'ont pas permis.

c) Cette architecture n'est pas beaucoup plus flexible que celle du TP1. Il est vrai qu'il y a un accès à la mémoire par un bus séparé ce qui permet de gagner des cycles. En revanche, cette architecture nous limite par l'ual qui nous oblige à utiliser le bus A et le bus B en même temps si l'on souhaite passer une instruction par l'ual. Cette architecture est donc plus flexible que celle du TP1 mais demeure néanmoins assez limitée.

EXERCICE 2

Cet exercice est présent sous forme de fichier dans le dossier de ce TP.

EXERCICE 3

Cet exercice est présent sous forme de fichier dans le dossier de ce TP.