École Polytechnique de Montréal

Département de génie informatique et génie logiciel

INF1600

Architecture des micro-ordinateurs

Rapport du laboratoire #2

Soumis par

Édouard Hébert

Sid'Ahmed Lehah

Section de labo #2

Remise le 28 février 2017

Exercice 1

Question 1 : recherche d'instruction

Microcodes d'une recherche d'instruction

RTN	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

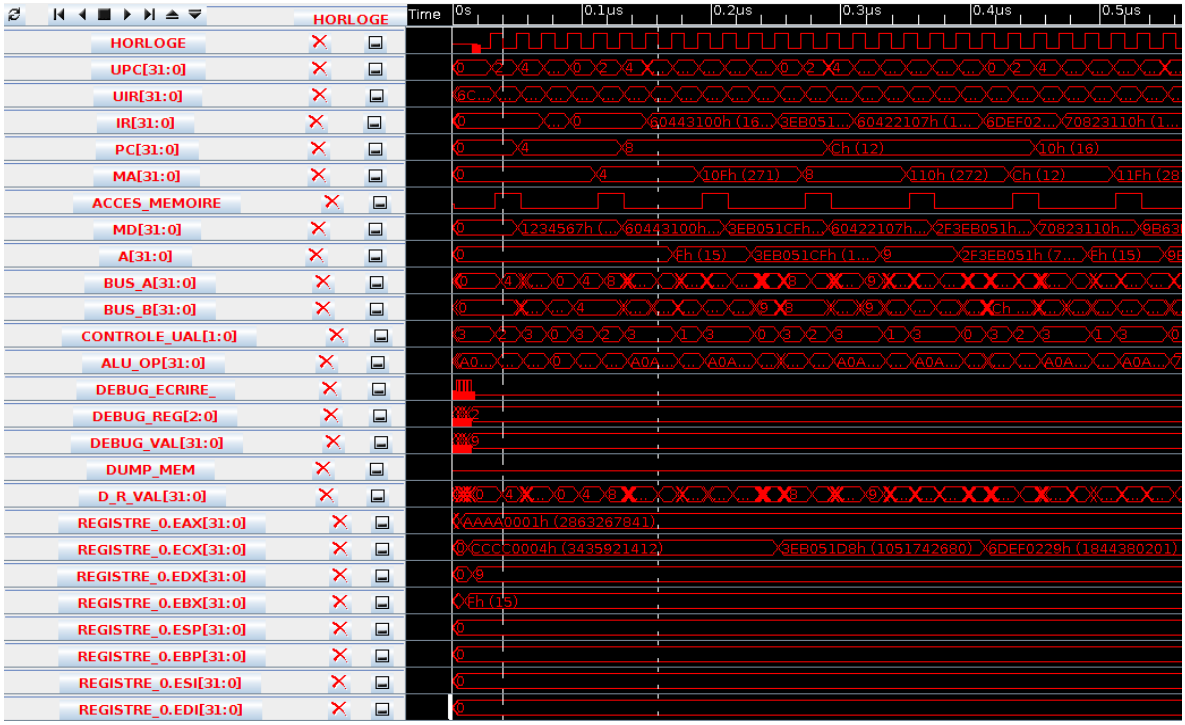
Question 2 : exécution d'une instruction générique

Microcodes d'une instruction générique

RTN	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	hexa
A <- R[IR<16..12>];	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	0	0x006E
MA <- A + IR<11..0>;	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0x1021
MD <- M[MA];	0	0	0	0	1	1	0	0	1	1	1	0	0	0	0	0	0x0CE0
A <- MD;	0	0	0	0	0	0	1	0	0	1	1	0	0	0	1	0	0x0262
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

Question 3 : simulation

Après avoir entré nos microcodes dans les fichiers appropriés, voici le résultat dans *Electric* :



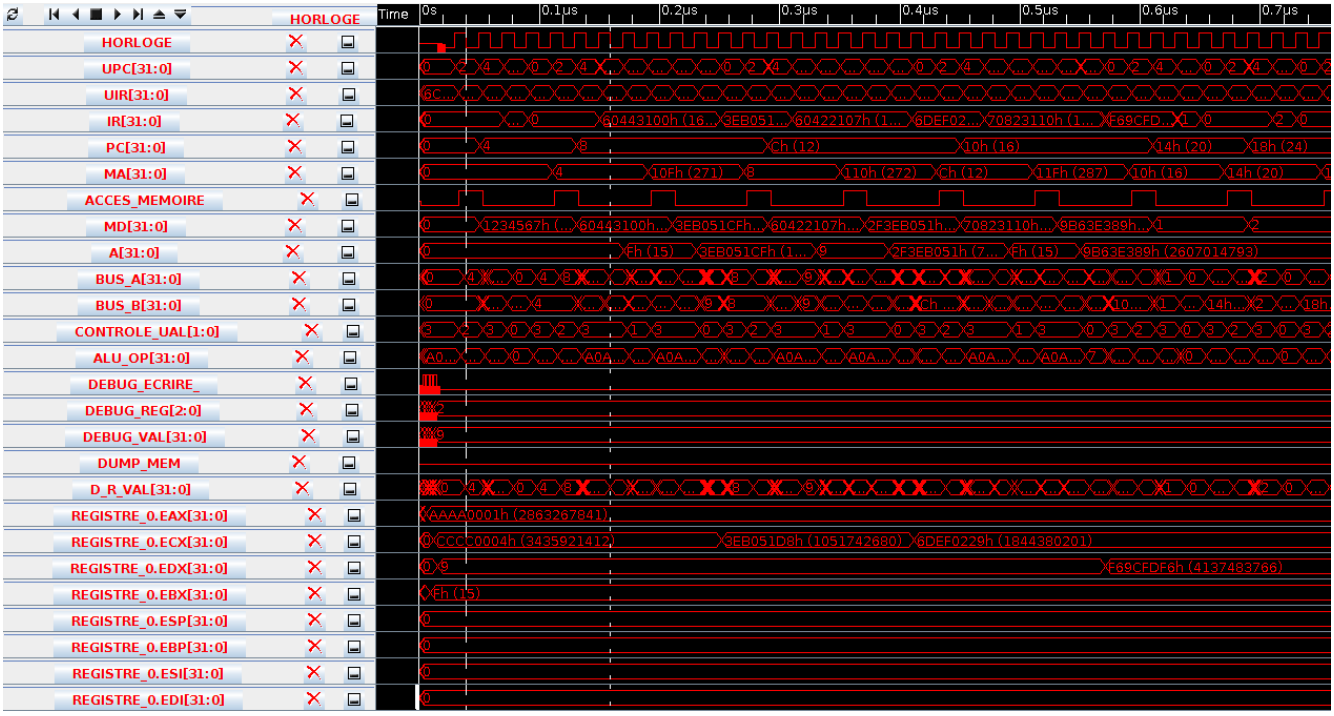
L'opération effectuée à PC = 4 est R[1] <= R[2] add M[R[3] + 0x100]. R[3] est 0xF, donc M[R[3] + 0x100] est 0x3EB051CF, que l'on peut voir dans MD un peu après 0.2µs. La valeur de R[2] est 0x9, et quand on l'additionne avec 0x3EB051CF, on obtient : 0x3EB051D8, que l'on peut voir dans R[2] (ECX). La prochaine opération est R[1] <= R[1] add M[R[2] + 0x107]. M[R[2] + 0x107] est M[0x110] = 0x2F3EB051. R[1] + M[0x110] = 0x3EB051D8 + 0x2F3EB051 = 0x6DEF0229, qui est bien la valeur suivante de R[1] (ECX).

Question 4 : opération NAND

La valeur de op[6..0] est séparée en les parties suivantes dans l'UAL : op[0], op[3..0], op[4], op[5] et op[6]. Nous voulons une opération NAND, donc il nous faut passer par le bloc logique32. Op[4] sera donc 0 pour sélectionner la sortie du bloc logique, et op[0] ne sera pas utilisé. Op[5] et op[6] servent à faire une addition à la sortie du bloc logique, ce dont nous ne voulons pas. Ces deux valeurs seront donc 0. Op[3..0] représente la table de vérité qui est passée au bloc logique. Cette table de vérité est représentée dans le tableau suivant :

A	B
1	1
0	1
1	0
0	0

Pour obtenir un NAND, il nous faut donc les valeurs 0, 1, 1, 1 dans op[3..0]. Ce qui nous donne op[6..0] = 0b0000111 = 0x07. Voici le résultat de la simulation de l'opération NAND après avoir rentré l'opcode dans les fichiers :



L'opération effectuée est R[2] <= R[1] NAND M[R[3] + 0x110].

R[2] <- R[1] NAND M[R[3] + 0x110]

R[2] <- 0x6DEF0229 NAND M[0x11F]

R[2] <- 0x6DEF0229 NAND 0x9B63E389

	0110	1101	1110	1111	0000	0010	0010	1001
NAND	1001	1011	0110	0011	1110	0011	1000	1001
	1111	0110	1001	1100	1111	1101	1111	0110

R[2] <- 0xF69CFDF6

Ce résultat est bien celui que l'on observe dans la simulation.

Question 5

a) L'instruction 0x05555555 donne les valeurs suivantes :

op = 0

ra = 0b10101

rb = 0b01010

rc = 0b10101

constante = 0b010101010101

Les deux derniers octets forment la constante ainsi que les 4 derniers bits de rc. Ils peuvent servir à n'importe quoi dépendamment de l'opcode. Pour cette instruction, l'opcode est 0, ce qui signifie qu'aucune opération n'est effectuée. Un exemple d'instruction ayant le même résultat est 0x07654321, puisque toutes les instruction avec l'opcode 0 ont le même effet.

b) Avec une architecture à deux bus, on peut rendre le code plus rapide en faisant des instructions en parallèle. Nous ne nous en sommes pas servis, car toutes nos instructions devaient prendre un cycle chaque. Nous aurions obtenu le même RTN avec une architecture à un bus.

c) L'architecture à deux bus est plus flexible, car sur le processeur étudié au TP1, on peut seulement accéder à la mémoire à partir de la bascule T située après l'UAL (l'équivalent de A). Sur le processeur du TP2, la mémoire peut être accédée directement à partir de la sortie de l'UAL sans affecter la valeur qui est stockée dans A.