INF1600 – TP2

Architecture à deux bus et   
introduction à l’assembleur IA-32

Richer Archambault : 1792473

Kevin Pantelakis : 1794745

Groupe : 02

Polytechnique Montréal

23 février 2016

Exercice 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 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| RTN concret | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | hexa |
| A<-R[rc]; | 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 | 1 | 0x1021 |
| MD <- M [MA]: A<-R[rb]; | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0x0CEA |
| R[ra] <-A oper MD; | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0x8210 |

1. Prise d’écran de la simulation  
   Comme on peut le constater sur la prise d’écran, le code pour réaliser l’instruction fonctionne bien. On voit bien que lorsque PC a une valeur de 0x0C ou (12)10, les micros instructions se déroulent bien notamment l’assignation du registre R[ra] avec l’opération inconnue entre A et MD qui se trouve dans le BUS B.

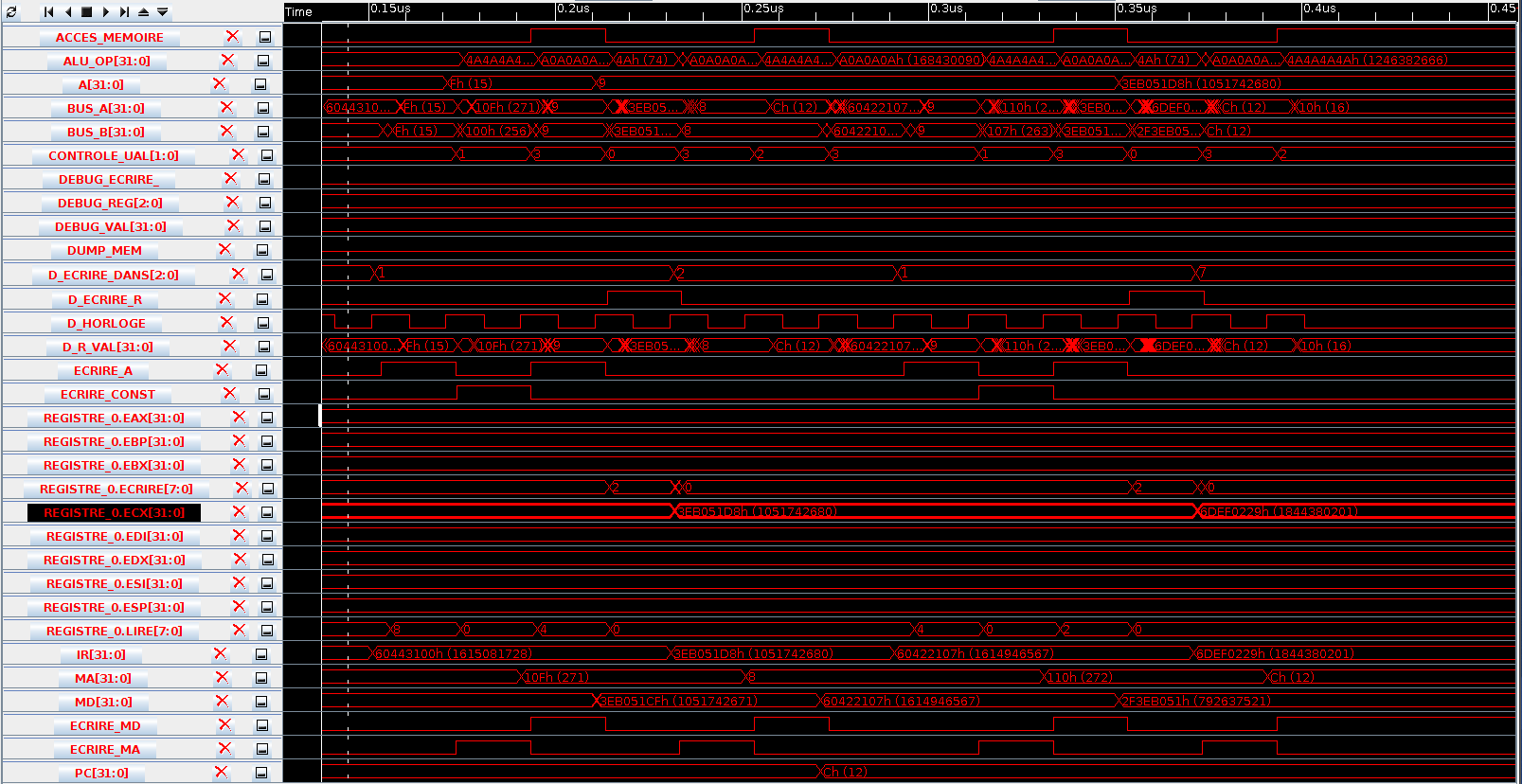


Figure : Prise d'écran de la simulation des micro instructions

1. La valeur hexadécimale à mettre dans l’opcode pour exécuter un XOR est : 0x06
2. Compréhension :
   1. Elle écrit dans PC, MA, MD, A et dans la banque de registre, l’addition des valeurs se trouvant dans A et le bus B
   2. Parce qu’on peut faire plus d’une instruction en même temps. Oui on s’en est servi lors de la 3ieme instruction.
   3. Oui il est aussi flexible sinon plus car il permet de faire plusieurs opérations différentes et plusieurs de celles-ci peuvent être exécuté en parallèle. De plus dans une architecture à deux BUS comme nous utilisons ici on peut lire et écrire dans la même micro instruction ce qui était impossible avec l’architecture du dernier laboratoire.

Exercice 2

Code assembleur dans le fichier exo2

Exercice 3

Code assembleur dans le fichier exo3