**Université du Québec à Montréal**

A2014

**Travail Pratique II**

**INF4170**

**Maxime Girard GIRM30058500**

Table des matières

[0](#_Toc403218957)

[Instruction BME (Branch Memory Equals) 3](#_Toc403218958)

[Processeur à un cycle 3](#_Toc403218959)

[Processeur multi-cycle 4](#_Toc403218960)

[Processeur “pipeline” 6](#_Toc403218961)

[Instruction EX (execute) 7](#_Toc403218962)

[Processeur multi-cycle 7](#_Toc403218963)

[Processeur en « pipeline » 8](#_Toc403218964)

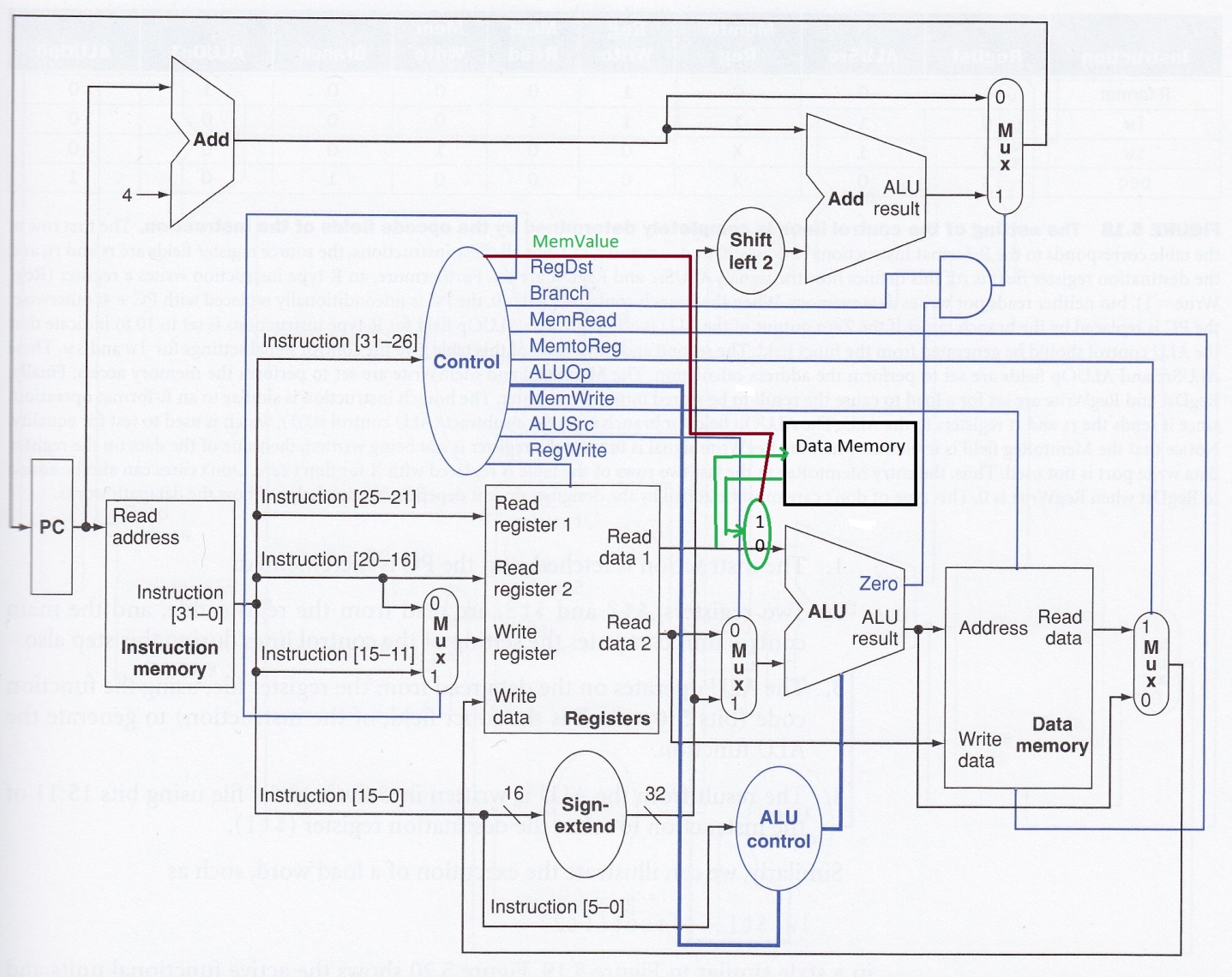
# Instruction BME (Branch Memory Equals)

## Processeur à un cycle

Ajout d’un composant “data memory” en haut à gauche de l’UAL et ajout d’un multiplexeur entre « read data 1 » du banc de registre et l’entré de l’UAL. Le multiplexeur décide entre envoyer la donnée de « read data 1 » directement dans l’UAL et envoyer la donnée reçu par le composant « data memory ».

Un bit de contrôle est ajouté « memValue » qui vaut 1 si l’instruction BME est appellé, 0 sinon. Ainsi si le bit « memValue » est activé la valeur du registre source 1 « data read 1 » sera envoyé dans le composant « data memory » afin d’y retrouver la valeur qui réside à l’adresse mémoire que détenait le registre source.

L’UAL est ainsi en mesure de procéder à une soustraction entre les deux valeur et activer le bit « zero » si les deux valeurs sont égales et le registre PC sera écraser par la valeur immediate de l’instruction.

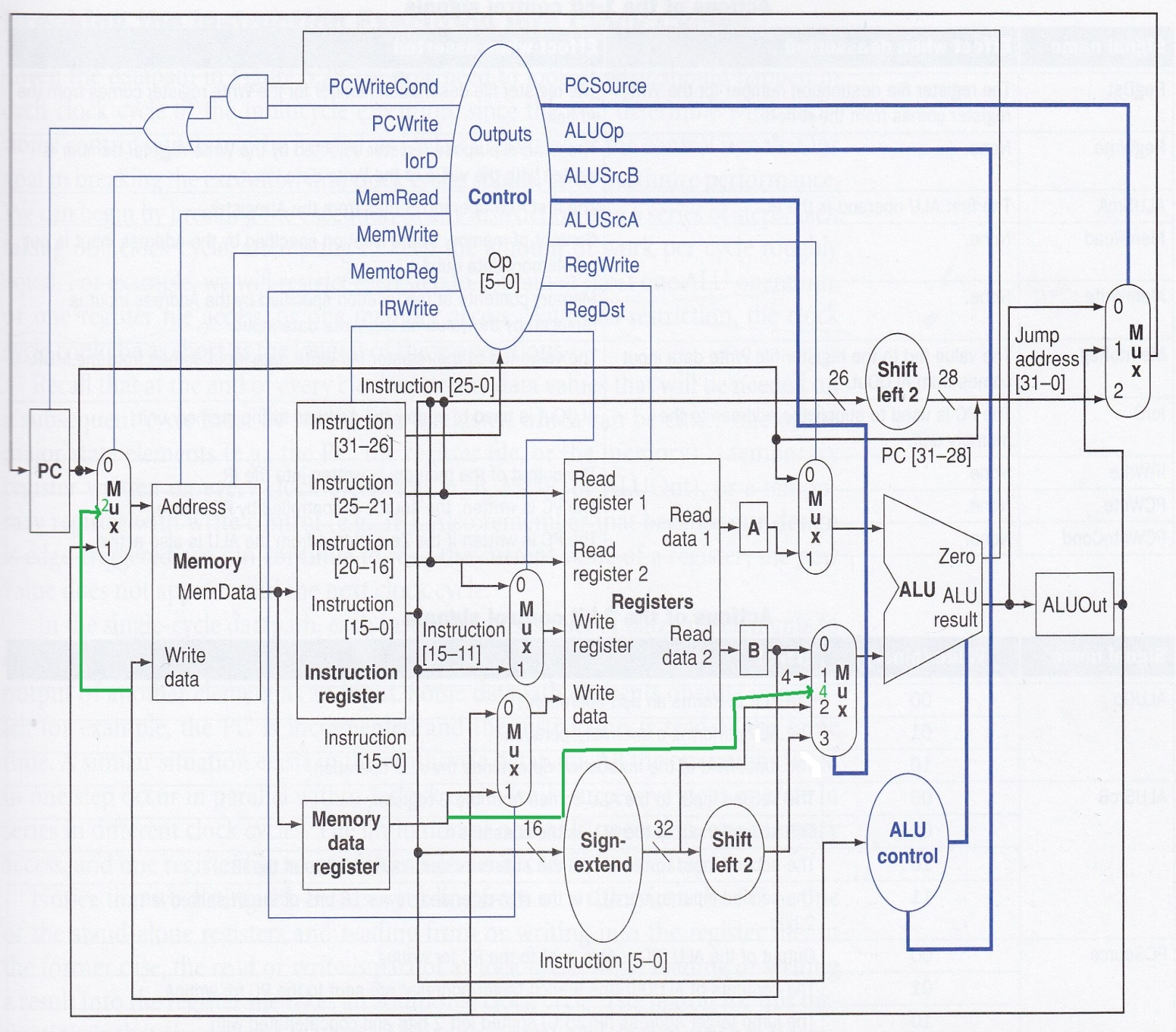


## Processeur multi-cycle

Ajout d’un registre de sauvegarde C près du registre de sauvegarde A, qui sauvegarde la valeur du registre à comparer dans l’UAL. Ajout d’un bit dans le signal de contrôle ALUSrcA. Ajout d’un bit dans le signal de contrôle ALUSrcB. Le multiplexeur qui reçoit ALUSrcB doit recevoir les donné stocké dans « memory data register » en plus.

Le multiplexeur qui reçoit PC doit avoir un bit de plus afin de pouvoir avoir le choix d’envoyer la valeur de l’adresse stocker dans le registre B au troisième cycle.

|  |  |  |
| --- | --- | --- |
| # de cycle | Action | Bits activés |
| 1 | L’adresse de l’instruction est retrouvé dans data memory et passé à instruction register afin d’y être décodé dans le prochain cycle.  PC + 4 est calculé. | ALUSrcB = 1 |
| 2 | L’instruction est décodé, l’opcode est envoyé au contrôle et on calcule le branch target. Les valeur des registres sources sont stocké dans les registres de sauvegarde A et B. | ALUSrcB = 3 |
| 3 | La valeur de B est envoyer à la mémoire pour que sa valeur soit chargé. La valeur d’ALUOut est toujours l’adresse mémoire du branchement. | ALUSrcB = 4,  ALUSrcA = 1,  lorD = 2,  MemRead = 1 |
| 4 | La valeur correspondant à l’adresse mémoire stocké dans B est retourner à L’UAL et la soustraction est faite, si le resultat est zero alors le bit zero est activé et on effectue le branchement avec l’adresse calculé stocké dans ALUOut en écrasant PC avec la valeur de ALUOut. | ALUSrcB = 4,  ALUSrcA = 1,  lorD = 2,  MemRead = 1,  PCSource = 1 |



## Processeur “pipeline”

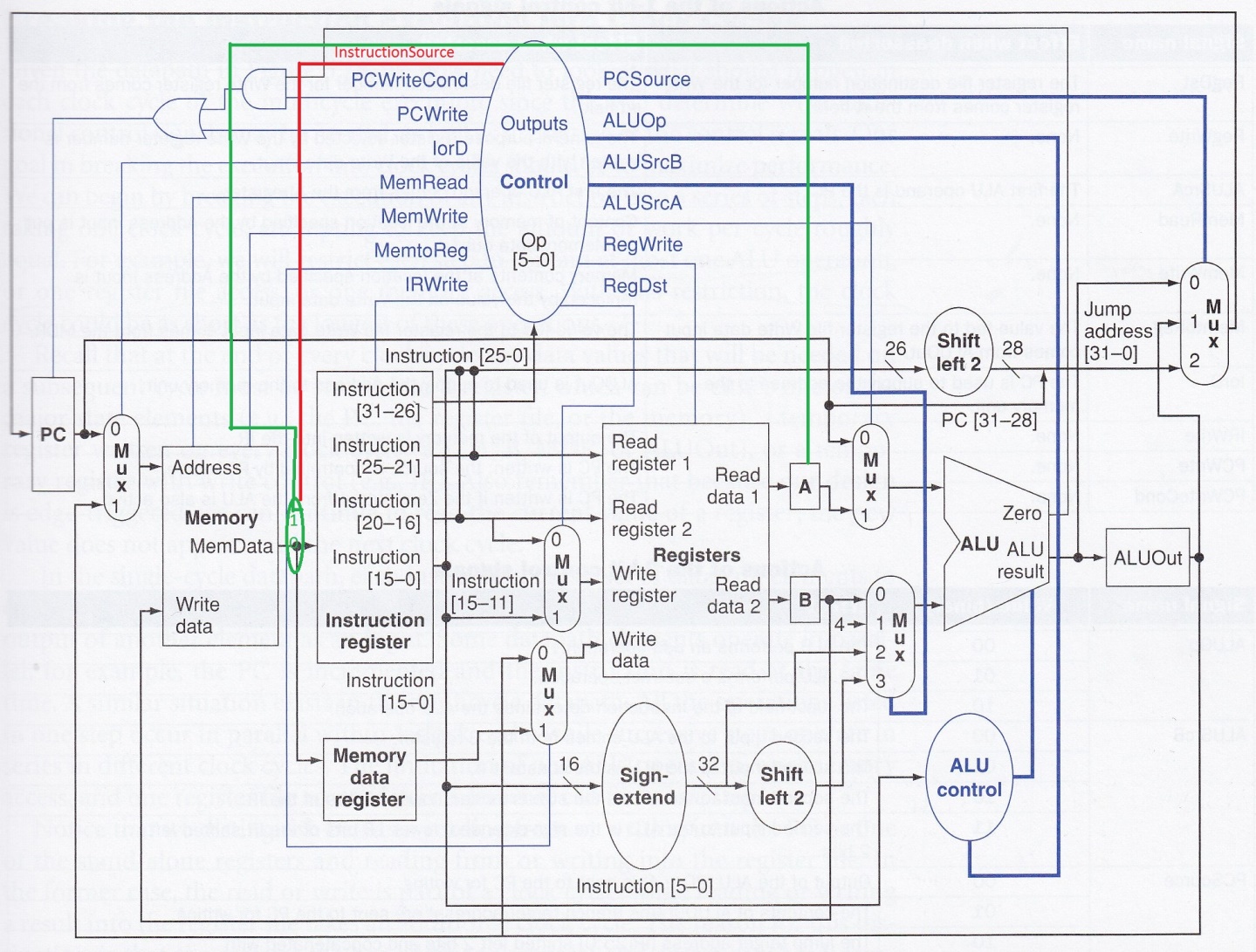
« not implemented yet »

# Instruction EX (execute)

## Processeur multi-cycle

Les deux premier cycles sont les même pour toutes les instructions, ceci dit, le registre A sauvegarder lors du deuxième cycle est renvoyé au composant Instruction Register et écrase le contenue si un nouveau bit de contrôle nommé « InstructionSource » est activé (ajout d’un multiplexeur).

Les deux cycle de base se répète et tout dépendant de l’instruction à exécuter ex $sx devrait prendre minimalement 5 cycle à s’éxécuter.



## Processeur en « pipeline »

« not implemented yet »