# 104-2 Computer Organization Pipeline Simulation Homework

## ➢ 作業要求

請同學實作課本 Pipeline 處理器之指令運作模擬程式。同學需實作下列指令：**「lw」、「add」、「addi」、「sub」、「and」、「or」、「ori」、「beq」**。並且能夠**偵測與處理「data hazard」、「hazard with load」以及「branch hazard」**。

## ➢ 程式的輸入與輸出

**輸入：**MIPS 指令轉換後的 32-bit 機器碼指令為程式的輸入，每題的 32-bit 機器碼指令放在作業資料夾中對應的「.txt」檔案，請同學們撰寫模擬程式將檔案讀入並處理。

**輸出：**根據連續輸入的 32-bit 機器碼指令，模擬其在每個 clock cycle 時，各個 pipeline registers 所儲存的值，並輸出寫入至對應文件中。其中「Instruction」和「Control signals」的結果以二進位表示，其餘皆以十進位表示。輸出文件格式請按照範例輸出（SampleOutput.txt）。「Control signals」的排列順序，與額外定義的「ALUop」請參考附錄。

請參考作業資料夾中的範例輸入「SampleInput.txt」，與其結果輸出「SampleOutput.txt」。其中範例輸入對應的 MIPS 指令如下：

```
lw      $1, 0x01($8)
add     $3, $0, $2
beq     $0, $2, 0x06    #(branch PC+4+4×6)
```

P.S. Branch 指令第三個欄位應為 label，這裡為了方便而表示成數字形式，非正確寫法。

## ➢ 題目說明

題目一共四題，每一題都是獨立的，不會相互影響。並請依照每一題輸入，將結果分別寫入到「genResult.txt」、「dataResult.txt」、「loadResult.txt」和「branchResult.txt」之中。以下是每題的解說。

1. 請同學的模擬程式從 "General.txt" 檔中讀出指令，其對應的 MIPS 指令為：

   ```
   or    $3, $5, $0
   add   $1, $4, $6
   and   $8, $7, $2
   ```

   將執行結果依照範例輸出格式寫入 " genResult.txt " 檔案中。

2. 請同學的模擬程式從 "Datahazard.txt" 檔中讀出指令，其對應的 MIPS 指令為：

   ```
   addi  $5, $2, 6
   sub   $6, $5, $7
   and   $4, $6, $5
   or    $3, $2, $5
   ```

   將執行結果依照範例輸出格式寫入 "dataResult.txt" 中。

3. 請同學的模擬程式從 "Lwhazard.txt" 檔中讀出指令，其對應的 MIPS 指令為：

   ```
   lw    $8, 0x03($5)
   and   $4, $8, $1
   ori   $3, $3, 5
   sub   $7, $8, $6
   ```

   將執行結果依照範例輸出格式寫入 "loadResult.txt" 中。

4. 請同學的模擬程式從 "Branchazard.txt" 檔中讀出指令，其對應的 MIPS 指令為：

   ```
   beq   $8, $2, 0x03   #(branch 至 and 指令)
   addi  $3, $6, 8
   sub   $4, $5, $1
   lw    $6, 0x00($0)
   and   $7, $2, $7
   ```

   將執行結果依照範例輸出格式寫入 "branchResult.txt" 中。

## ➢ 暫存器、指令記憶體與資料記憶體的初始化

**暫存器初始化為以下所示：**假設宣告一般暫存器$0 ~ $8，共 9 個暫存器。

| 暫存器編號 | $0 | $1 | $2 | $3 | $4 |
|---|---|---|---|---|---|
| 初始值 | 0 | 8 | 7 | 6 | 3 |
| 暫存器編號 | $5 | $6 | $7 | $8 | |
| 初始值 | 9 | 5 | 2 | 7 | |

**資料記憶體初始化為以下所示：**

| 記憶體位址 | 0 | 4 | 8 | 12 | 16 |
|---|---|---|---|---|---|
| 初始值 | 5 | 5 | 6 | 8 | 8 |

**指令記憶體的初始位址由「0」開始。**

## ➢ 附錄

「Control signals 順序表」

| | Execution/Address Calculation stage control lines | | | | Memory access stage control lines | | | Write-back stage control lines | |
|---|---|---|---|---|---|---|---|---|---|
| | Reg Dst | ALU Op1 | ALU Op0 | ALU Src | Branch | Mem Read | Mem Write | Reg write | Mem to Reg |
| R-type | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| lw | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| sw | X | 0 | 0 | 1 | 0 | 0 | 1 | 0 | X |
| beq | X | 0 | 1 | 0 | 1 | 0 | 0 | 0 | X |

其中控制訊號為 X 的部分輸出 0 或 1 皆可

「ALUop 定義表」

| | ALUop | | func | | | | | | ALU | ALUctr | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | bit<1> | bit<0> | bit<5> | bit<4> | bit<3> | bit<2> | bit<1> | bit<0> | Operation | bit<2> | bit<1> | bit<0> |
| I | 1 | 1 | x | x | x | x | x | x | Or | 0 | 0 | 1 |
| I | 0 | 0 | x | x | x | x | x | x | Add | 0 | 1 | 0 |
| I | 0 | 1 | x | x | x | x | x | x | Subtract | 1 | 1 | 0 |
| R | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | Add | 0 | 1 | 0 |
| R | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | Subtract | 1 | 1 | 0 |
| R | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | And | 0 | 0 | 0 |
| R | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | Or | 0 | 0 | 1 |
| R | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | Set on < | 1 | 1 | 1 |

紅色部分為給予 I-type 指令控制 ALU 做 Or 運算而新增的，其它定義為原本的並無更動