105-2 Computer Organization Pipeline Simulation Homework

➢ 作業要求

請同學以 C/C++/Java 程式語言實作課本 Pipeline 處理器之指令運作模擬程式。同學需實作下列指令：「**lw**」、「**sw**」、「**add**」、「**addi**」、「**sub**」、「**and**」、「**andi**」、「**or**」、「**slt**」、「**bne**」。並且能夠偵測與處理「**data hazard**」、「**hazard with load**」、「**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。

範例：

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

```
lw   $1, 0x03($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。

P.S. 後面的題目可能會包含前面的 hazard。

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：

```
lw      $9, 0x04($2)
addi    $5, $0, 9
or      $4, $8, $7
```

將執行結果依照範例輸出格式寫入" genResult.txt "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：

```
sub     $2, $3, $6
add     $9, $2, $6
sw      $5, 0x04($2)
slt     $3, $2, $9
```

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：

```
andi    $3, $1, 7
lw      $6, 0x06($9)
add     $4, $6, $5
and     $2, $8, $7
```

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"Branchhazard.txt"檔中讀出指令，其對應的 MIPS 指令為：

```
bne     $4, $5, 0x03    # (branch 至 or 指令)
addi    $4, $4, 1
sub     $8, $7, $1
lw      $9, 0x00($0)
or      $5, $1, $9
```

將執行結果依照範例輸出格式寫入"branchResult.txt"中。

➢ 暫存器、資料記憶體與指令記憶體的初始化

暫存器初始化為以下所示：假設宣告一般暫存器$0 ~ $9，共10個暫存器。

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

資料記憶體初始化為以下所示：

記憶體位址	0x00	0x04	0x08	0x0C	0x10
初始值	5	9	4	8	7

指令記憶體的初始位址由「0」開始。

➢ 附錄

「Control signals」

	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」

Type	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	And	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