Computer Organization, Fall 2013

Lab 3: Single Cycle CPU Ⅱ

Due: 2013/11/14 23:59:59

1. 作業目的

以 LAB2 為基礎，加上 memory unit(named the module DM in Simple_Single_CPU，並且能執行 R-type、I-type、branch 以及 jump 類型的指令，實作出一個完整的 single cycle CPU。

2. 作業要求(90%)

A. 請使用 ModelSim 為模擬平台。

B. 兩個人為一組，並在繳交的每個檔案開頭上以註解方式附上兩人學號、姓名。
上傳檔案為: 學號 1_學號 2.rar(檔名不要有中文)

C. Top module 的名稱和 IO ports 的命名請按照以下方式:
Top module: Simple_Single_CPU.v

```
Module Simple_Single_CPU(
    clk_i,    //system clock (input)
    rst_i     //reset signal (input)
);
```

為了培養良好的 coding style, 請務必遵守以下幾點:

I. 一個.v 檔內只有一個 module。

II. module name 和檔案名稱必須相同。

D. Reg_File[29]代表 stack point，請將 Reg_File[29]的值初始成 128，其餘設成 0。
Decoder 中要加入額外的 control signal 以控制各個 unit 的運作

- BranchType_o
- Jump_o
- MemRead_o
- MemWrite_o
- MemtoReg_o

E. Please name the Data_Memory module DM in Simple_Single_CPU.

F. **Basic instruction (60%)**

Lab2 instruction + mul、lw、sw、jump

Instruction	Op[31:26]	rs	rt	rd	shamt	func
mul	6’b000000	rs[25:21]	rt[20:16]	rd[15:11]	5’b00000	6’b011000
lw	6’b100011	rs[25:21]	rt[20:16]	imm[15:0]		
sw	6’b101011	rs[25:21]	rt[20:16]	imm[15:0]		
jump	6’b000010	Address[25:0]				

I. Mul

Reg[rd] ← Reg[rs]*Reg[rt]

II. lw

MemWrite 為 0，MemRead 為 1，RegWrite 為 1

Reg[rt] ← Mem[rs+imm]

III. sw

MemWrite 為 1，MemRead 為 0

Mem[rs+imm] ← Reg[rt]

IV. jump

Jump 為 1。

PC={PC[31:28]，address<<2}

G. Advanced instruction (30%)

Instruction	Op[31:26]	rs	rt	imm
bgt	6’b000111	rs[25:21]	rt[20:16]	imm[15:0]
bnez	6’b000101	rs[25:21]	5’b00000	imm[15:0]
bgez	6’b000001	rs[25:21]	5’b00001	imm[15:0]
lui	6’b001111	5’b00000	rt[20:16]	imm[15:0]

I. bgt (branch greater than)

Branch 為 1

if (rs > rt) then PC = PC + 4 + (sign_imm<<2)

II. bnez (branch non equal zero)

Branch 為 1

if (rs != 0) then PC = PC + 4 + (sign_imm<<2)

III. bgez (branch greater equal zero)

Branch 為 1

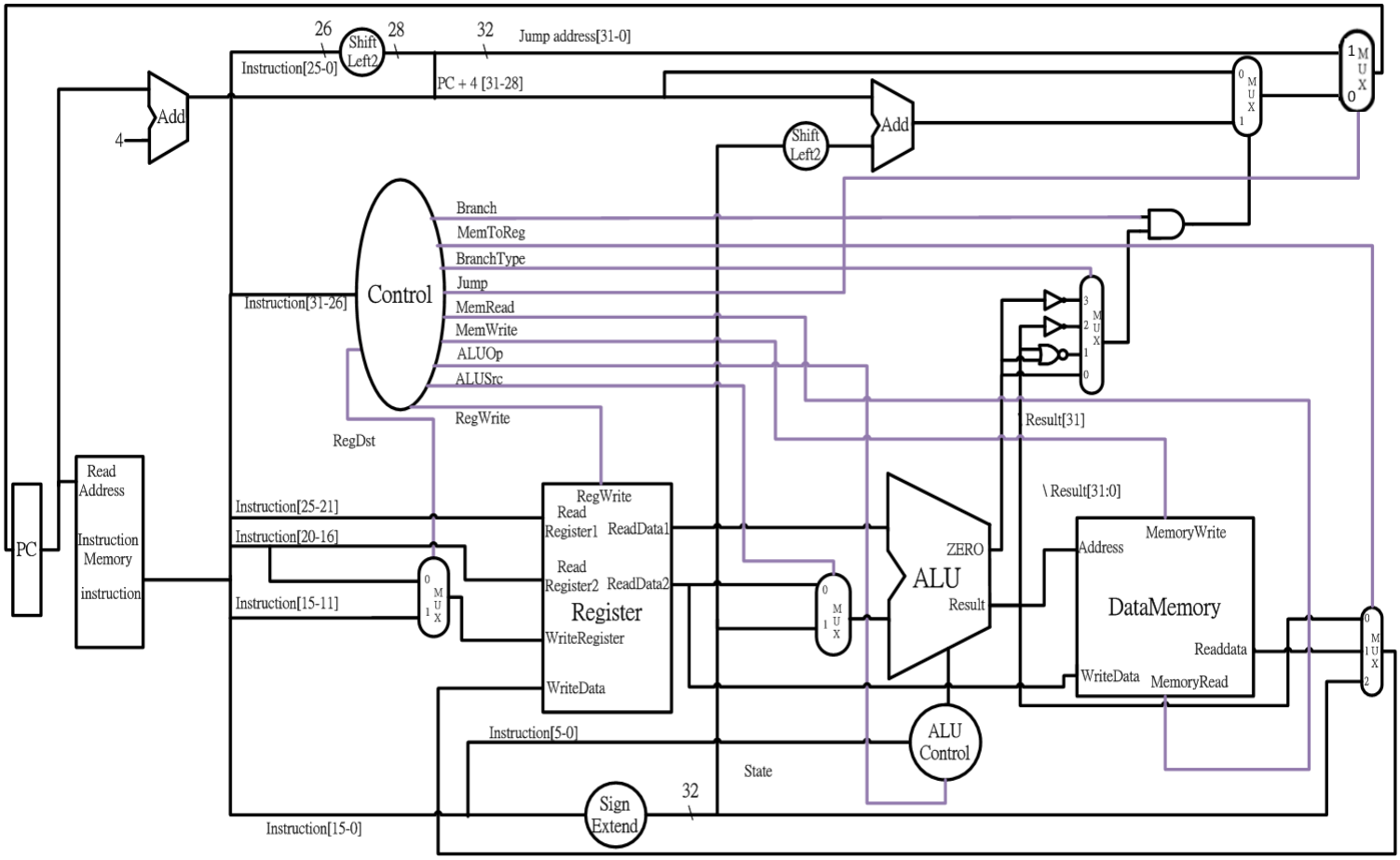
if (rs >= 0) then PC = PC + 4 + (sign_imm<<2)

IV. li (load unsigned immediate)

Reg[rt] ←unsigned_ imm

3. 基本架構圖

此次 lab 若需要用到額外的 control signal，請將新增的部分畫在架構圖上，並在報告中描述實作流程。



4. Bonus (20%)

A. Bonus1 (10%)

Instruction	op[31:26]	rs	rt	rd	shamt	func
jal	6'b000011	Address[25:0]				

jr	6'b000000	rs	0	0	0	6'b001000

I. jal (jump and link)

reg[31]=PC+4

PC={PC[31:28],address[25:0]<<2}

MIPS中，第31個register用來存function call時的return address。

Reg[31]儲存PC+4並且執行jump

II. jr (jump to the address in the register rs)

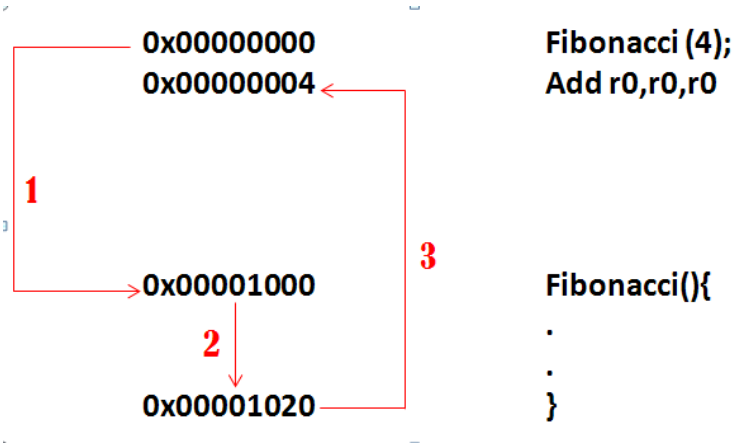
PC=reg[rs];

MIPS 裡，return 可用

jr r31

來跳回 jal 時存的 return address

III. 範例 ： 程式中執行 function call 時



若要執行 recursive function 必須配合 stack point 也就是 Reg_File[29]。先將 reg 用 sw 存入 memory 中，等 function call 結束以後再 lw 回來。

B. Bonus2 (10%)

請利用至今已完成的 MIPS instruction set 編寫一個簡單的 sorting 程式，在一開始先利用 lui, sw 指令將 Data_Memory 中的 memory[0] ~ memory[4] 分別 assign 成 4, 5, 3, 2, 1。預期在程式結束時看到 memory[0] ~ memory[4]的值由小到大排列。

同學可自行撰寫 assembly code 或利用 C code , MIPS 轉換的 tool 來幫忙，若你轉換成 16 進位需要將 Test_Bench.v 裡讀取 instruction memory 的$readmemb 改成 $readmemh，若你的程式超過 50 道 instruction 也請修改 END_COUNT(default : 50)。

5. 測試說明

A. Basic instruction (60%)
提供測資 CO_LAB3_test_data1.txt 供同學測試自己的設計，結果比對可參照 test1.txt。

B. Advanced instruction (30%)
CO_LAB3_test_data2.txt 延續前面的測資加上更多的指令供同學測試自己的設計，結果比對可參照 test2.txt。

C. Bonus 1(10%)
CO_LAB3_test_data3.txt是一個Fibonacci function，執行完畢r2代表最後的答案，詳細說明請看test3.txt。

D. Bonus 2(10%)
請參照助教提供的測資形式，繳交 CO_LAB3_bonus.txt 及 bonus.txt 兩個檔案。CO_LAB3_bonus.txt 中請提供要被執行指令的 machine code，bonus.txt 中請提供你所撰寫的 assembly code 並加入適當註解說明。

E. 請使用 Modelsim 作為模擬的平台，同學上傳的檔案必須保證能在 Modelsim 模擬時能輸出正確答案。

F. 同學可修改 Test_Bench.v 中讀取的指令檔案(default: CO_LAB3_test_data1.txt)，在 demo 時助教會有其他組測資來測試同學的設計。

6. 評分方式

A. **抄襲一律 0 分!**

B. 基本: 60 分, 進階: 30 分, 加分: 20 分。

C. 報告: 10 分 – 格式參考此次附檔，請務必說明分工情形。

D. 每遲交一天作業分數打 **9** 折**!**

E. 此次 Lab 需要 demo , demo 時間: 11/19 7:30 ~9:30 p.m. 、11/20 7:00~9:00 p.m.

(請於 11/4~6 至 EC447 門口填寫 demo 時間)

7. 繳交方式

請上傳至 E3 平台。

請將所有.v 檔(勿將整個專案一起壓進來)、bonus (.txt 檔)及報告放置於同資料夾下並壓縮

命名成：學號 1_學號 2.rar

格式錯誤將會斟酌扣分

8. Q&A

若對作業有任何疑問，請到課程網頁論壇 https://sites.google.com/site/nctuco/q-a-forum 發問