Computer Organization, Spring 2019

Lab 3: Single Cycle CPU II

Due : 2019/05/26

1. Goal

Based on Lab 2 (simple single-cycle CPU), add a memory unit to implement a complete single-cycle CPU which can run R-type, I-type and jump instructions.

2. Demands

a) Please use ModelSim as your HDL simulator.

b) "Data_Memory.v", and "TestBench.v" are supplied. Please use these modules to accomplish the design of your CPU.

c) You cannot create additional module (.v file) for your design.

d) Compress all *.v source files and report(*.pdf) as a zip file and name your submission with your Student ID (e.g., 0616001_0616002.zip).

e) Refer to Lab 2 for top module's name and IO ports.
Initialize the stack pointer (i.e., Reg_File[29]) to 128, and other registers to 0.
Decoder may add control signals:

-Branch_o

-Jump_o

-MemRead_o

-MemWrite_o

-MemtoReg_o

3. Requirement description

a) **Basic instruction:**

Lab 2 instruction + **lw、sw、beq、bne、j**

- Format:

R-type

Op[31:26]	Rs[25:21]	Rt[20:16]	Rd[15:11]	Shamt[10:6]	Func[5:0]

I-type

Op[31:26]	Rs[25:21]	Rt[20:16]	Immediate[15:0]

Jump

Op[31:26]	Address[25:0]

- Definition:

lw instruction :

memwrite is 0，memread is 1，regwrite is 1
Reg[rt] ← Mem[rs+imm]

sw instruction :

memwrite is 1，memread is 0
Mem[rs+imm] ← Reg[rt]

branch instruction :

branch is 1，and decide branch or not by do AND with the zero signal from ALU

beq:

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

bne:

if (rs!=rt) then PC=PC+4+ (sign_Imm<<2)

jump instruction:

jump is 1
PC={PC[31:28], address<<2}

Op field:

instruction	Op[31:26]
lw	6’b100001
sw	6’b100011
beq	6’b111011
bne	6’b100101
jump	6’b100010

Extend ALUOp from 2-bit to 3-bit: (You can modify this if necessary)

instruction	ALUOp
R-type	010
addi	100
lui	101
lw、sw	000
beq	001
bne	110
jump	x

b) **Advance set 1:**

Jal: jump and link

In MIPS, 31th register is used to save return address for function call Reg[31] save PC+4 and perform jump.

Reg[31]=PC+4
PC={PC[31:28], address[25:0]<<2}

Op[31:26]	Address[25:0]
6’b100111	Address[25:0]

Jr: jump to the address in the register rs

PC=reg[rs]
e.g.： In MIPS, return could be used by jr r31 to jump to return address from JAL.

Op[31:26]	Rs[25:21]	Rt[20:16]	Rd[15:11]	Shamt[10:6]	Func[5:0]
6’b111111	rs	0	0	0	6’b001000

c) **Advance set 2:**

blt (branch on less than): if(rs<rt) then branch

Op[31:26]	Rs[25:21]	Rt[20:16]	Immediate[15:0]
6'b100110	rs	rt	offset

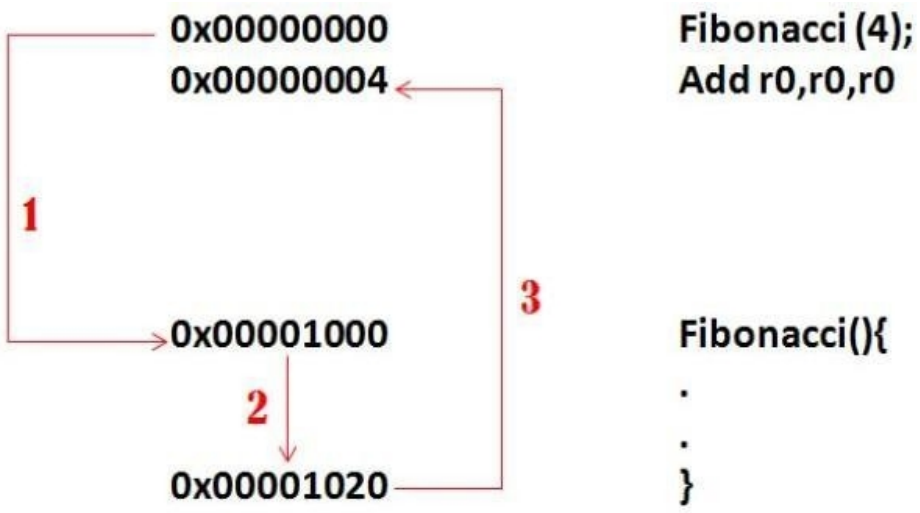
bnez (branch non equal zero): if(rs!=0) then branch (it is same as bne)

Op[31:26]	Rs[25:21]	Rt[20:16]	Immediate[15:0]
6'b101101	rs	0	offset

bgez (branch greater equal zero): if(rs>=0) then branch

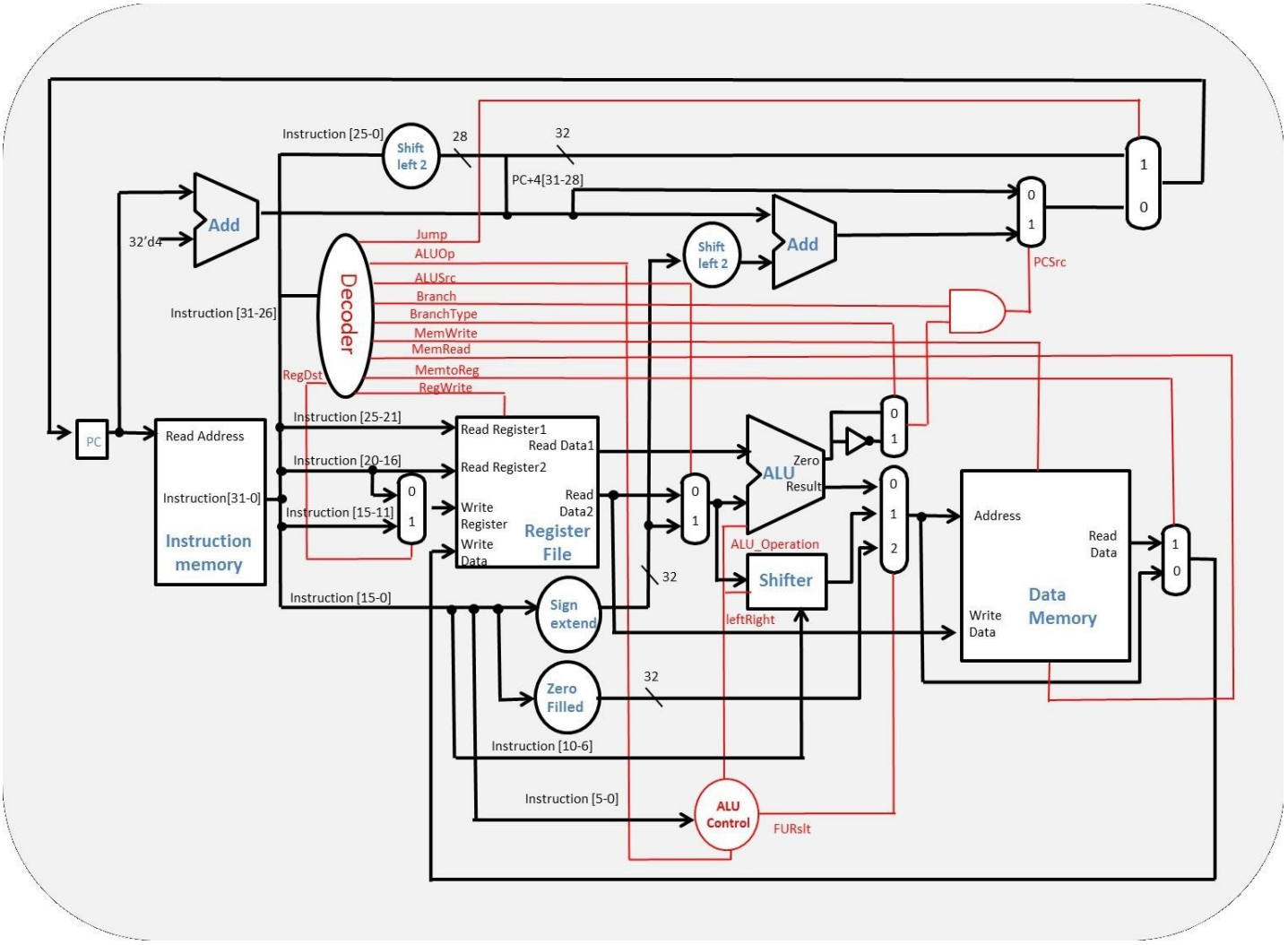
Op[31:26]	Rs[25:21]	Rt[20:16]	Immediate[15:0]
6'b110001	rs	1	offset

Example: when CPU executes function call:



if you want to execute recursive function, you must use the stack point (REGISTER_BANK [29]). First, store the register to memory and load back after function call has been finished.

4. Architecture Diagram



5. Test

Modify line 123 of TestBench.v to read different data.

CO_P3_test_data1.txt tests the basic instructions.

CO_P3_test_data2.txt tests the advanced set1.

CO_P3_test_data2_2.txt tests the advanced set2.

6. Grade

a) Total score: 100pts. COPY WILL GET A 0 POINT!

b) Instruction score: Total 80 pts

– basic instructions: 65 pts

– advanced set 1: 10 pts

– advanced set 2: 5 pts

c) Report: 20 pts – format is in CO_document and hand in as pdf format

d) Late submission: 10 points off per day

7. Q&A

If you have any question, just send email to TAs via new E3.