

Computer Organization, Spring 2019

Lab 4: Single Cycle CPU II

Due : 2019/05/26

1. Goal

Based on Lab 3 (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 **ModleSim** 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. Submit all *.v source files and report(pdf) on new e3. **Other form of file will get 0%.**
- D. Refer to Lab 3 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 3 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
 $\text{Reg}[\text{rt}] \leftarrow \text{Mem}[\text{rs} + \text{imm}]$

sw instruction :

memwrite is 1 , memread is 0
 $\text{Mem}[\text{rs} + \text{imm}] \leftarrow \text{Reg}[\text{rt}]$

branch instruction :

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

if (rs==rt) then $\text{PC} = \text{PC} + 4 + (\text{sign_Imm} \ll 2)$

bne:

if (rs!=rt) then $\text{PC} = \text{PC} + 4 + (\text{sign_Imm} \ll 2)$

Jump instruction :

jump is 1

$\text{PC} = \{\text{PC}[31:28], \text{address} \ll 2\}$

Op field:

instruction	Op[31:26]
lw	6'b011110
sw	6'b011100
beq	6'b000100
bne	6'b011010
jump	6'b011101

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'b011000	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'b000000	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'b011001	rs	rt	offset

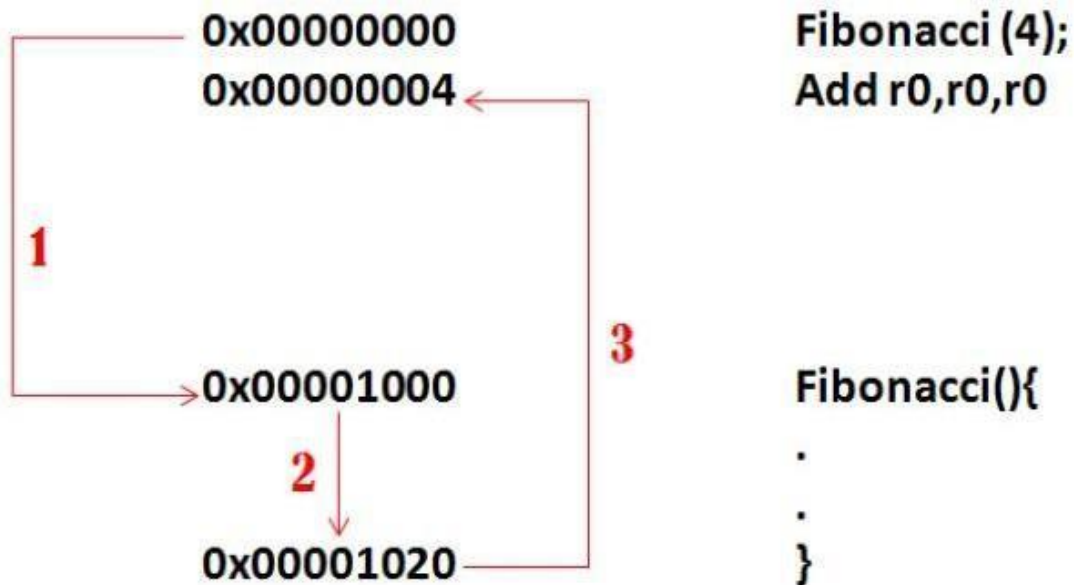
bnez (branch non equal zero): if($rs \neq 0$) then branch (it is same as bne)

Op[31:26]	Rs[25:21]	Rt[20:16]	Immediate[15:0]
6'b010010	rs	0	offset

bgez (branch greater equal zero): if($rs \geq 0$) then branch

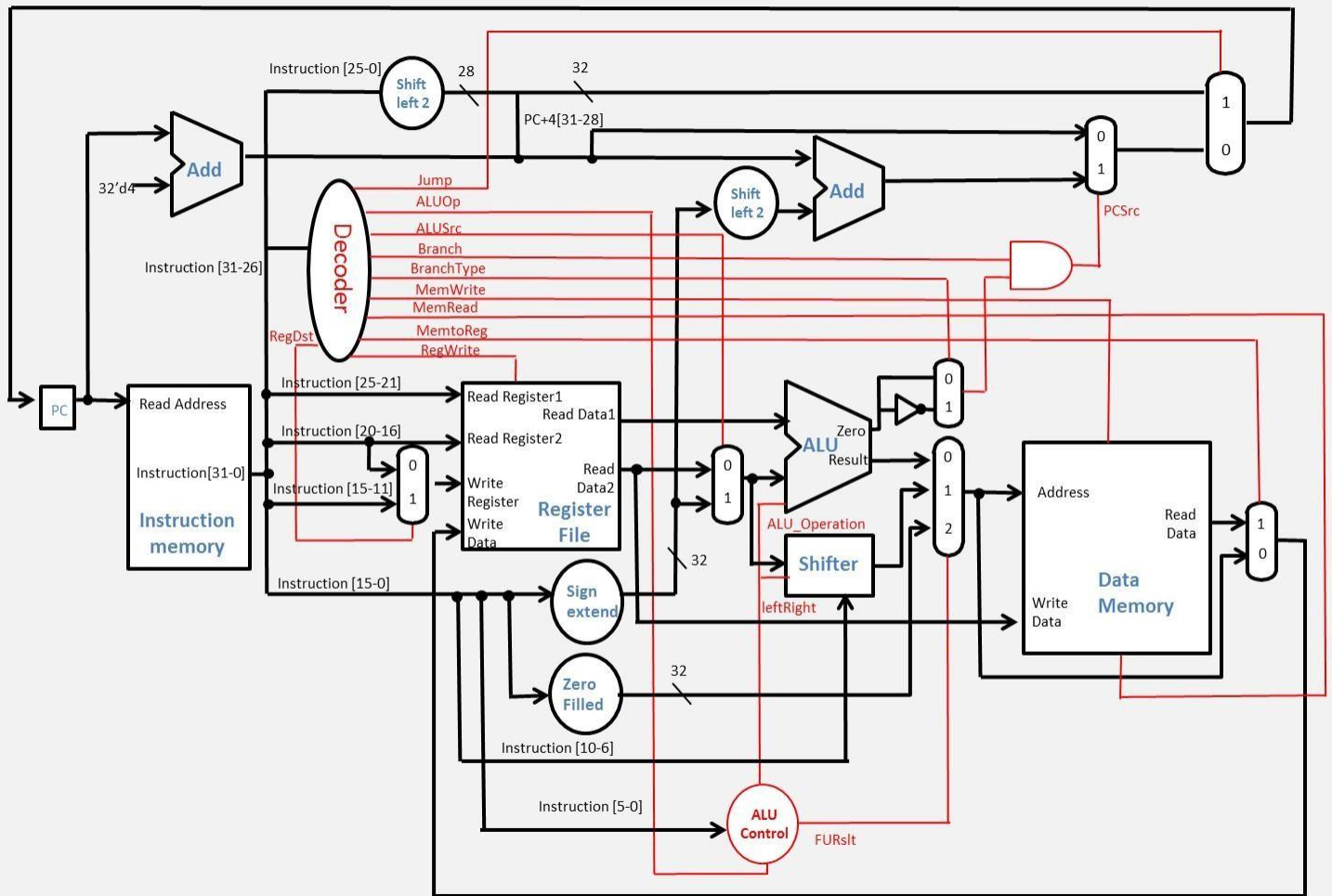
Op[31:26]	Rs[25:21]	Rt[20:16]	Immediate[15:0]
6'b001110	rs	1	offset

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.

6. Grade

- Total score: **120pts**. **COPY WILL GET A 0 POINT!**
- Instruction score: **Total 100 pts**
 - basic instructions: 75 pts
 - advanced set 1: 15 pts
 - advanced set 2: 10 pts
- Report: **20 pts** – format is in 改成你的學號.doc. (up to 2 pages)
請真的改成你的學號後再上傳