Computer Organization 2021

Lab 3: Single Cycle CPU

Due : 8/11 23:55

1. Goal

Based on reference (simple single-cycle CPU) that we give, 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 iverilog as your HDL simulator.

B. "Data_Memory.v", and "TestBench.v" are supplied. Please use these modules and modules in reference code to accomplish the design of your CPU. Specify in your report if you have any other files in your design.

C. Submit all *.v source files and report(pdf) on new e3.
Other form of file will get -10%.

D. Refer to reference code 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:

reference 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'b101100
sw	6'b101101
beq	6'b001010
bne	6'b001011
jump	6'b000010

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'b000011	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'b001110	rs	rt	offset

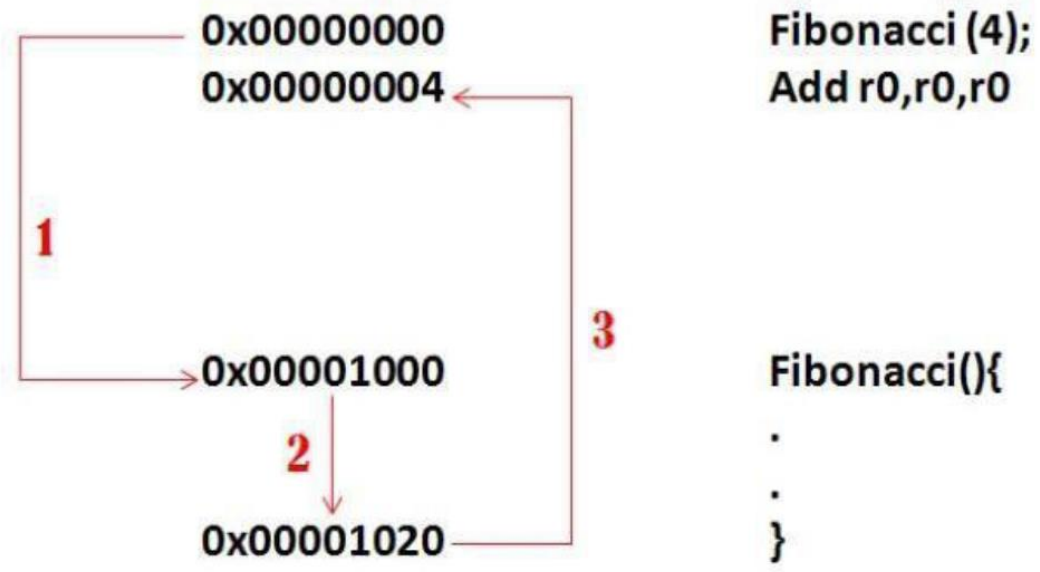
bnez (branch non equal zero): if(rs!=0) then branch (it is same as bne)

Op[31:26]	Rs[25:21]	Rt[20:16]	Immediate[15:0]
6'b001100	rs	000000	offset

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

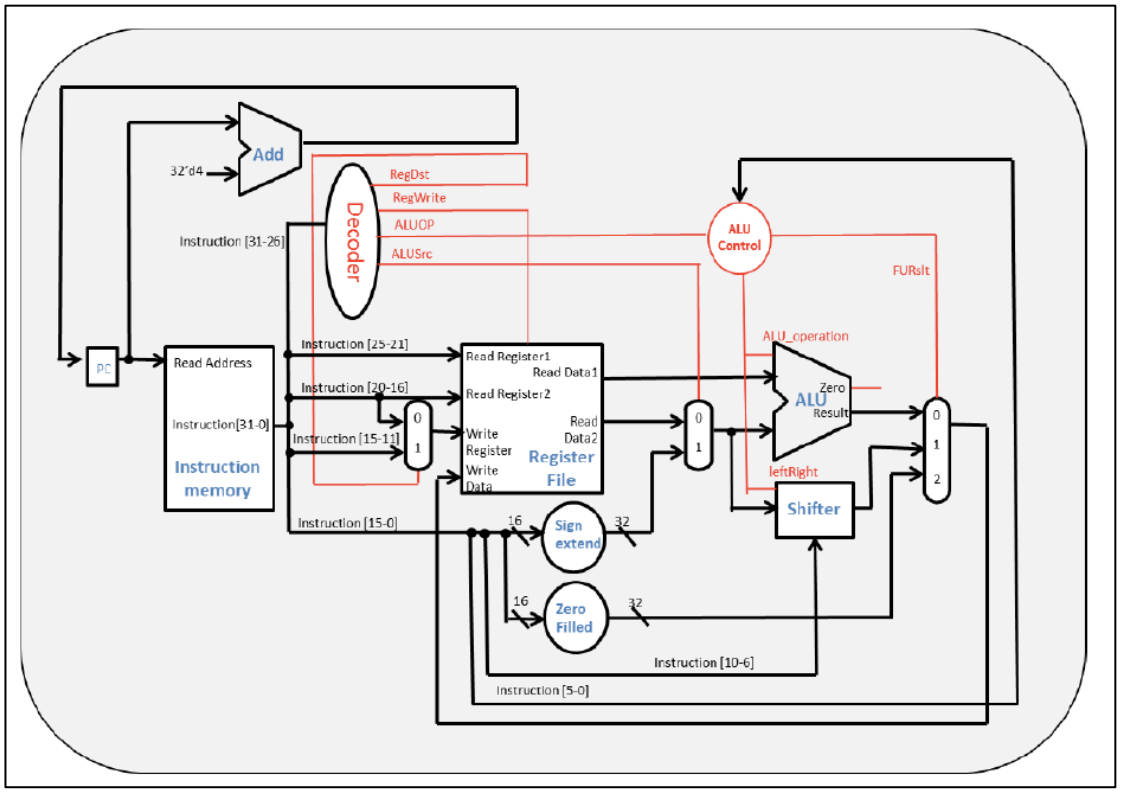
Op[31:26]	Rs[25:21]	Rt[20:16]	Immediate[15:0]
6'b001101	rs	000001	offset

Example: when CPU executes function call:



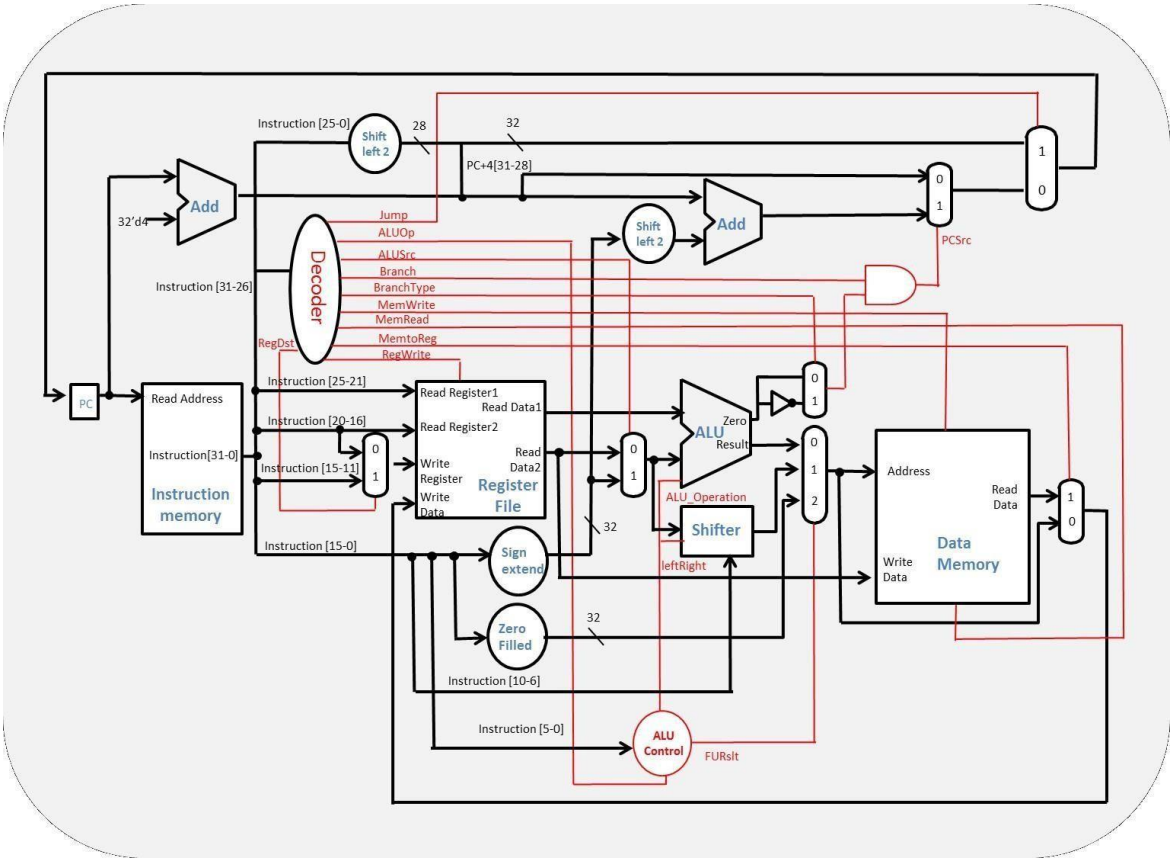
If you want to execute recursive function, you must use the stack point (REGISTER_BANK [29]). First, store the register to memory and load back after function call has been finished.

4. Reference architecture diagram



5. Architecture Diagram

You may modify the Simple_Single_CPU.v and add some modules to finish this architecture.

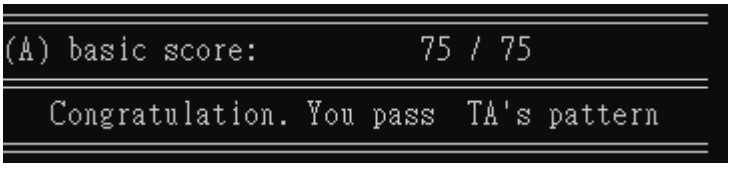


6. Test

Modify line 139 to 141 of TestBench.v to read different data.
Ex: $readmemb("CO_P3_test_data1.txt", cpu.IM.Instr_Mem)
$readmemb("CO_P3_test_data2.txt", cpu.IM.Instr_Mem)
CO_P3_test_data1.txt tests the basic instructions.
CO_P3_test_data2.txt tests the advanced set 1.
CO_P3_test_data2_2.txt test the advanced set 2.

After the simulation of TestBench, you will get the file CO_P3_result.txt. You can verify the result with dataX_result.txt.

If your design passes the test data, the following words would show in the terminal.



You can add more "`include" instructions if necessary.

```
`include "Adder.v"
`include "ALU.v"
`include "ALU_Ctrl.v"
`include "Data_Memory.v"
`include "Decoder.v"
`include "Instr_Memory.v"
`include "Mux2to1.v"
`include "Mux3to1.v"
`include "Program_Counter.v"
`include "Reg_File.v"
`include "Shifter.v"
`include "Sign_Extend.v"
`include "Simple_Single_CPU.v"
`include "Zero_Filled.v"
```

7. Grade

a. Total score: 120pts. COPY WILL GET A 0 POINT!

b. Instruction score: Total 100 pts
   basic instructions: 75 pts
   advanced set 1: 15 pts
   advanced set 2: 10 pts

c. Report: 20 pts – format is in StudentID_report.pdf.

8. Hand in your assignment

Please upload the assignment to the E3.

Put all *.v files and report(StudentID_report.pdf) into same compressed file.

(Use Lab3_StudentID.zip to be the name of your compressed file)

9. Q&A

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