Computer Organization, Spring 2017

Lab 3: Single Cycle CPU –Simple Edition

Due : 2017/04/27

1. Goal

Utilizing the ALU & Shifter in Lab2 to implement a simple single cycle CPU. CPU is the most important unit in computer system. Reading the document carefully and do the Lab, you will have elementary knowledge of CPU.

2. Demands

A. Please use ModleSim or Xilinx as your HDL simulator.

B. One person forms a group. Please use your student ID as your file name. (Ex. Lab3_0416001.zip) The type of compressed file must be "zip".
Other form of file will get -10%.

The assignment you upload on E3 must have the form of "Lab3_student ID.zip".

C. "Simple_Single_CPU.v", "Adder.v", "ALU.v", "ALU_Ctrl.v", "Decoder.v", "Instr_Memory.v", "Mux2to1.v", "Mux3to1.v", "Program_Counter.v", "Reg_File.v", "Shifter.v", "Sign_Extend.v", "Zero_Filled.v", and "TestBench.v" are supplied. Please use these modules to accomplish the design of your CPU. You may create additional module(.v file) for your design, if necessary.
In Lab3, the complete modules of "Instr_Memory.v", "Program_Counter.v", "Reg_File.v" are provided. You should accomplish other modules to make this CPU work.

D. Instruction set: the following instructions have to running in your designed CPU(80pts.).

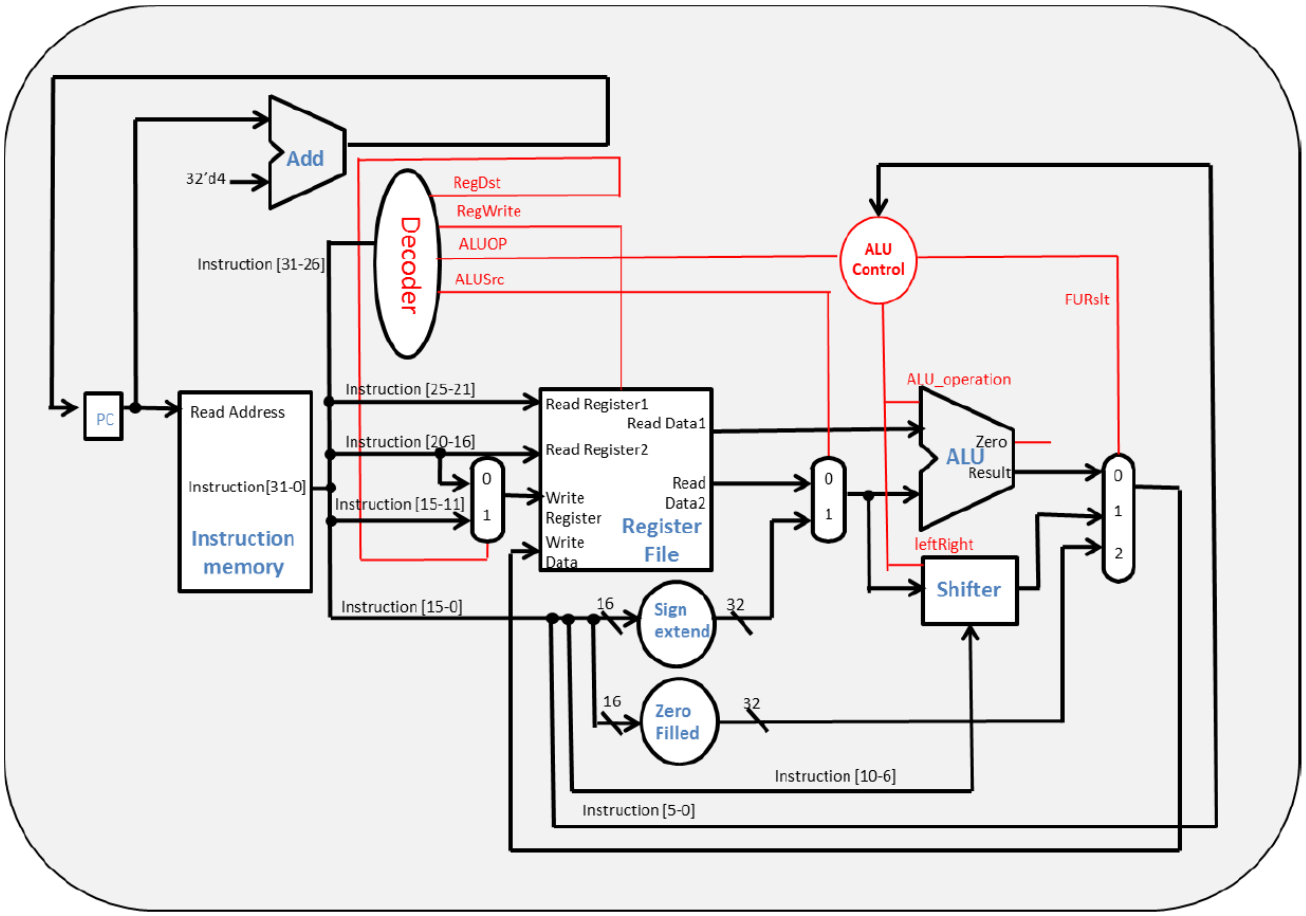
The following instructions have to be running in your designed CPU.

Instruction	Example	Meaning	Op field	shamt	Function field
ADD (addition)	add r1,r2,r3	r1 = r2+r3	0	x	32 (6'b100000)
SUB (subtraction)	sub r1,r2,r3	r1 = r2-r3	0	x	34 (6'b100010)
AND (logic and)	and r1,r2,r3	r1 = r2&r3	0	x	36 (6'b100100)
OR (logic or)	or r1,r2,r3	r1 = r2|r3	0	x	37 (6'b100101)
NOR (not or)	nor r1,r2,r3	r1 = ~(r2|r3)	0	x	39 (6'b100111)

SLT (set on less than)	slt r1,r2,r3	if (r2<r3) r1=1 else r1=0	0	x	42 (6'b101010)
SLL (shift left logic)	sll rd,rt,5	rd = rt<<5	0	5'd5	0 (6'b000000)
SRL (shift right logic)	srl rd,rt,5	rd = rt>>5	0	5'd5	2 (6'b000010)
ADDI (add immediate)	addi r1,r2,10	r1 = r2+10	8 (6'b001000)	x	x
LUI (load upper immediate)	lui r1,100	r1 = 100*2^16	15 (6'b001111)	x	x

Table 1: Instruction definition and example

3. Architecture Diagram



C. Bonus

Additional functions for ALU and shifter are described in the following subsection with bonus.

Implement sllv and srlv instructions. You can add new module or control signal in this advanced design.

Instruction	Example	Meaning	Op field	shamt	Function field
SLLV	sllv rd,rt,rs	rd = rt<<rs	0	x	4 (6'b000100)
SRLV	srlv rd,rt,rs	rd = rt>>rs	0	x	6 (6'b000110)

4. Test Bench

After you hand in your code, TA will use the same TestBench module and different test data to verify the correctness of your design of Simple_Single_CPU.

If you have attached additional modules for your design, do ensure that those modules would not affect our testing.

In Lab3, three test data (binary code), stored in "CO_P3_test_data1.txt"~ "CO_P3_test_data3.txt", are provided. The default test data is the first one. If you would like to use second test data, modify the module " TestBench.v" in line 75 as follows:

```
$readmemb("CO_P3_test_data2.txt", cpu.IM.Instr_Mem);
```

Use the similar way to check the third test data "CO_P3_test_data3.txt". You can design other test data by yourself to verify the correctness of your design.

The Assembly codes of the test data are given as follows:

CO_P3_test_data1.txt	CO_P3_test_data2.txt	CO_P3_test_data3.txt (for bonus)

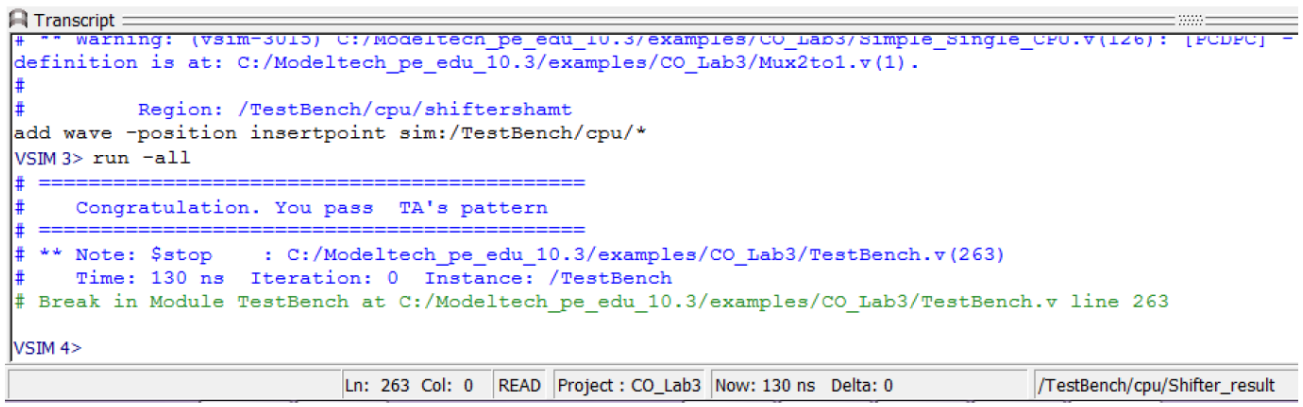
addi r1,r0,10
addi r2,r0,4
slt r3,r1,r2
add r4,r1,r2
sub r5,r1,r2
nor r5,r5,r0
	addi r6,r0,3
	addi r7,r0,14
	and r8,r6,r7
	or r9,r6,r7
	sll r10,r9,3
	lui r11,1
	srl r12,r11,5
		addi r1,r0,6
		addi r2,r0,3
		addi r6,r0,-5
		add r3,r1,r2
		sub r4,r2,r3
		nor r5,r6,r0
		sll r7,r5,2
		sllv r8,r7,r1
		srlv r9,r8,r5
		srl r10,r8,1

result	result	result
r0=0, r1=10, r2=4, r3=0, r4=14, r5=-7	r6=3, r7=14, r8=2, r9=15, r10=120, r11=65536, r12=2048	r0=0, r1=6, r2=3, r3=9, r4=-6, r5=4, r6=-5, r7=16, r8=1024, r9=64, r10=512

After the simulation of TestBench, you will get the file "CO_P3_result.txt".

You can verify the result.

If your design pass the test data, the following words would show in the Transcript windows.



5. Grade

a. Total score: 110pts. COPY WILL GET A 0 POINT!
b. Instruction score: 80 pts.
c. Report: 20pts –format is in CO_document.
d. Bonus: 10pts

6. Hand in your assignment

Please upload the assignment to the E3.

Put all of .v source files and report into same compressed file.

(Use your Lab3_student ID.zip to be the name of your compressed file)

7. Q&A

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

8. Notice

a. Use the modules provided to implement your Single Cycle CPU
b. Do not modify any existing code except the Input filename in TestBench.v
c. Write your code in "/*your code here*/"
d. In Lab3, the modules to be designed by students should be implemented as combination circuits. (Do not design as sequential circuits or Zero score will be given!)

9. Appendix

In lab3, you can use 32bits ALU.

Here is the example of 32bits ALU from textbook

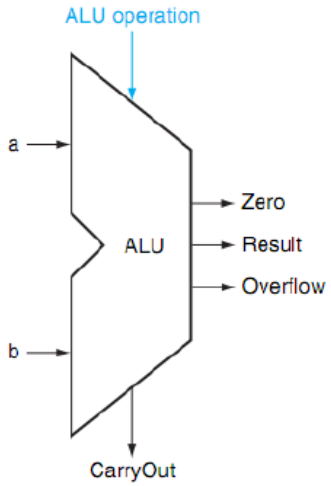


FIGURE C.5.14 The symbol commonly used to represent an ALU, as shown in Figure C.5.12. This symbol is also used to represent an adder, so it is normally labeled either with ALU or Adder.

```
module MIPSALU (ALUctl, A, B, ALUOut, Zero);
   input [3:0] ALUctl;
   input [31:0] A,B;
   output reg [31:0] ALUOut;
   output Zero:
   assign Zero = (ALUOut==0); //Zero is true if ALUOut is 0
   always @(ALUctl, A, B) begin //reevaluate if these change
      case (ALUctl)
         0: ALUOut <= A & B;
         1: ALUOut <= A | B;
         2: ALUOut <= A + B;
         6: ALUOut <= A - B;
         7: ALUOut <= A < B ? 1 : 0;
         12: ALUOut <= ~(A | B); // result is nor
         default: ALUOut <= 0;
      endcase
    end
endmodule
```

FIGURE C.5.15 A Verilog behavioral definition of a MIPS ALU.