Computer Organization, Spring 2015

Lab 2: Single Cycle CPU – Simple Edition

Due : 2015/04/27

1. Goal

Utilizing the ALU in Lab1 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 Xilinx ISE as your HDL simulator.

b. Please attach your names and student IDs as comments at the top of each file. PLEASE FOLLOW THE FOLLOWING RULE! Zip your folder and name it as "ID.zip" (e.g., 0216001.zip) before uploading to e3. Multiple submissions are accepted, and the version with the latest time stamp will be graded.

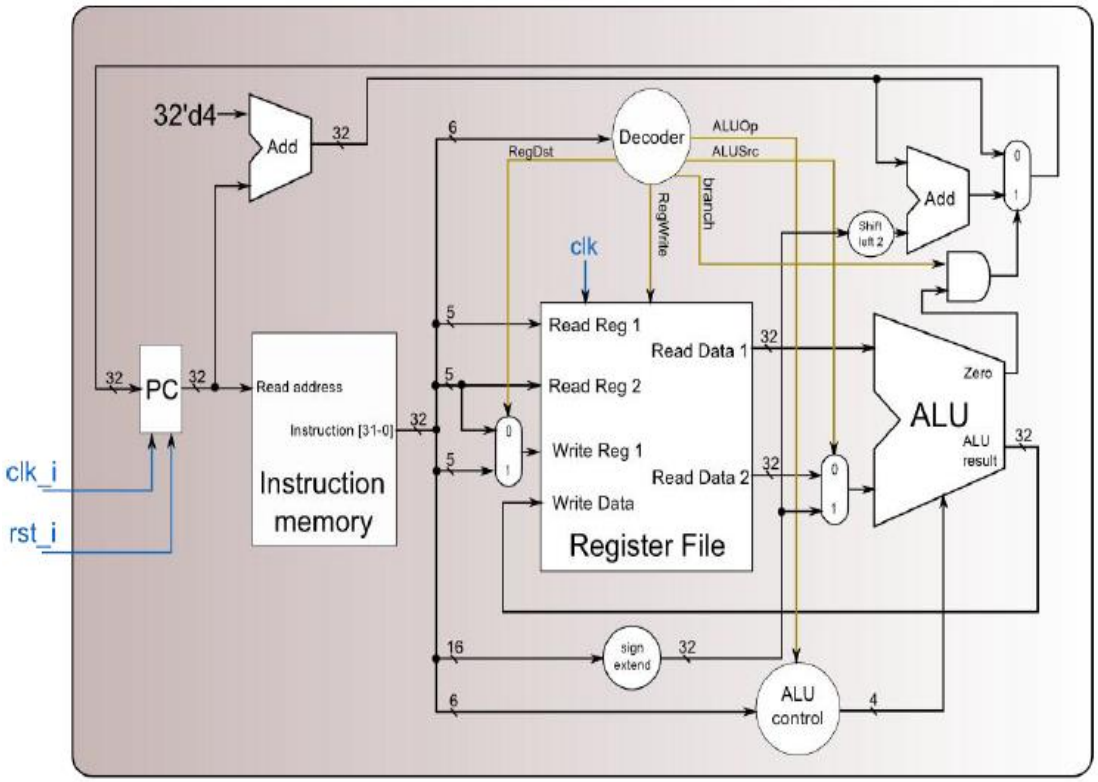
Note: You must be uploading the “.zip” file to e3.

c. Program Counter, Instruction Memory, Register File and Test Bench are supplied.

d. Instruction set: the following instructions have to running in your designed CPU (60 pts.).

Instruction	Example	Meaning	Op field	Function field
ADD (Addition)	add r1, r2, r3	r1 = r2 + r3	0	32(0x20)
ADDI (Add Immediate)	addi r1, r2, 100	r1 = r2 + 100	8	0
SUB (Subtraction)	sub r1, r2, r3	r1 = r2 - r3	0	34(0x22)
AND (Logic And)	and r1, r2, r3	r1 = r2 & r3	0	36(0x24)
OR (Logic Or)	or r1, r2, r3	r1 = r2 | r3	0	37(0x25)
SLT (Set on Less Than)	slt r1, r2, r3	if (r2 < r3) r1 = 1 else r1 = 0	0	42(0x2a)
SLTI (Set on Less Than Immediate)	slti r1, r2, 10	if(r2 < 10) r1 = 1 else r1 = 0	10(0xa)	0
BEQ (Branch On Equal)	beq r1, r2, 25	if (r1 == r2) go to PC+4+100	4	0

3. Architecture Diagram



Top module: Simple_Single_CPU

4. Advance Instructions (20 pts.)

Should modify the architecture of basic design above

I. ALUOp should be extended to 3bits to implement I-type instructions.

Original 2bits ALUOp from textbook：00->000, 01->001, 10->010

II. Encode shift L/R and LUI instruction by using unused ALU_ctrl.

Ex. ALU_ctrl=0 is AND, 1 is OR…, 0 1 2 6 7 &12 are used by basic

Instructions

Instruction	Example	Meaning	Op field	Function field
SRL(Shift Right Logic)	srl r1, r2, 10	r1 = r2 >> 10	0	2
SRLV(Shift Right Logic Vaiable)	srlv r1, r2, r3	r1 = r2 >> r3	0	6
LUI(Load Upper Immediate)	lui r1, 10	r1 = 10 * 2^16	15(0xf)	0
ORI(Or Immediate)	ori r1, r2, 100	r1 = 2 | 100	13(0xd)	0
BNE(Brench On Not Equal)	bne r1, r2, 30	if(r1!=r2) go to PC+4+120	5	0

To implement those advanced instructions, please note about the following formats

SRL Rd, Rt, shamt

Shift register Rt right by the distance indicated by immediate *shamt*

0	Rs	Rt	Rd	shamt	2

Rs is ignored for srl

SRLV Rd, Rt, Rs

Shift register Rt right by the distance indicated by the register Rs

0	Rs	Rt	Rd	shamt	6

Hint: ALU bypass Rt value and use a module to shift it left or right.

You can implement these by your design.

LUI Rt, Imm

0xf	0	Rt	Imm

ORI Rt, Rs, Imm

Put the logical OR of register Rs and the zero-extended immediate into register Rt

5. Test

There are 3 test patterns, CO_P2_test_data1.txt ~CO_P2_test_data3.txt. The default pattern is the first one. Please change the column 39 in the file "Instr_Memory.v" if you want to test the other cases.

$readmemb("CO_P2_test_data1.txt", Instr_Mem)

The following are the assembly code for the test patterns.

1	2	3

addi r1, r0, 13
addi r2, r0, 7
slti r3, r1, 10
beq r3, r0, 2
slt r4, r2, r1
and r5, r1, r4
sub r4, r1, r5
	addi r6, r0, -2
	addi r7, r0, 5
	or r8, r6, r7
	addi r9, r0, -1
	addi r6, r6, 2
	add r9, r9, r6
	beq r6, r0, -3
		ori r10, r0, 3
		lui r11, 10
		srl r11, r11, 8
		srlv r11, r11, r10
		addi r10, r10, -1
		bne r10, r0, -3

final result	final result	final result
r1 = 13, r2 = 7, r3 = 0 r4 = 13, r5 = 0.	r6 = 2, r7 = 5, r8 = -1 r9 = 1	r10 = 0 r11 = 40

The file "CO_P2_Result.txt" will be generated after executing the Testbench. Check your answer with it.

6. Grade

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

b. Basic score: 60 pts. Advance instructions: 20 pts.

c. Report: 20pts – format is in CO_document.

d. **Delay: 10%off/day**

7. 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 student ID to be the name of your compressed file)

8. Q&A

If you have any question, go to E3 or just send email to TAs.

9. Appendix

In lab2, you can use 32bits ALU to implement shift instruction and lui instruction.

Here is the example of 32bits ALU from textbook

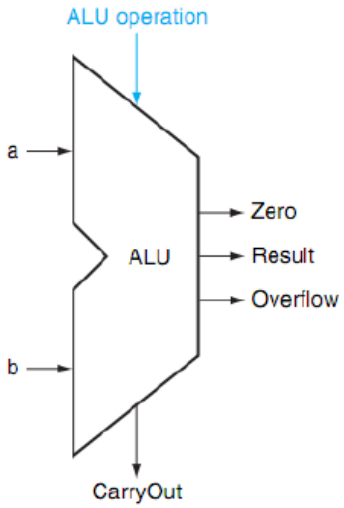


FIGURE C.5.14 The symbol commonly used to represent an ALU, as shown in Figure C.5.12. This symbol is also used to represent an adder, so it is normally labeled either with ALU or Adder.

```
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.