Computer Organization, spring 2016

Lab 2: Single Cycle CPU – Simple Edition

Due: 2016/04/25

1. Goal

Utilizing the ALU in Lab1 to implement a simple single cycle CPU. CPU is the most important unit in computer system. Read the document carefully and do the Lab, and you will have the elementary knowledge of CPU.

2. Requirement

a. Please use Xilinx as your HDL simulator. Xilinx ISE Design Suite 14.7 is used to evaluate.

b. Please attach student IDs as a comment at the top of each file.

c. PLEASE FOLLOW THE FOLLOWING RULE! Zip your folder and submit only one *.zip file. Name the *.zip file with your student IDs (e.g., 0316001_0316002.zip). Other filenames and formats such as *.rar and *.7z are NOT accepted! Multiple submissions are accepted, and the version with the latest time stamp will be graded. For the ease of grading, a team's submissions should be uploaded by the same person (so we don't have to check if your teammate submits a new/different version).

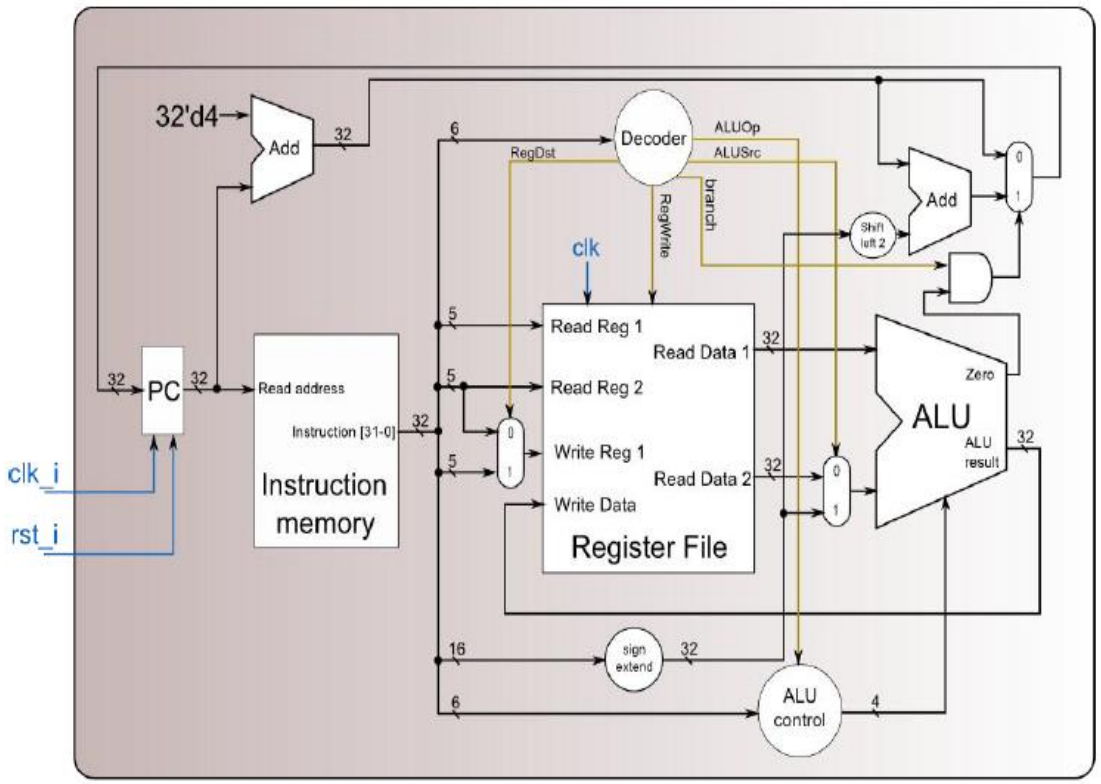
d. Program Counter, Instruction Memory, Register File and Test Bench are provided.

e. Instruction set: the following instructions are to run on your 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)
SLTIU(Set on Less Than Immediate unsigned)	sltiu r1,r2,10	if(r2 < 10) r1 = 1 else r1 = 0	9(0x9)	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.)

Modify the architecture of the 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 right 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
SRA(Shift Right Arithmetic)	sra r1,r2,10	r1 = r2 >>> 10	0	3
SRAV(Shift Right Arithmetic Variable	srav r1,r2,r3	r1 = r2 >>> r3	0	7
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(Branch 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

SRA Rd, Rt, shamt

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

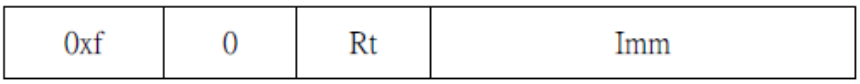
Rs is ignored for sra

SRAV Rd, Rt, Rs

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

Hint1: Be careful of using Verilog operator >>> directly in your code. To use this operator, you have to declare the variable as signed.

LUI 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
sltiu r3,r1,0xFFFF
beq r3,r0,1
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
		sra r11,r11,8
		srav r11,r11,r10
		addi r10,r10,-1
		bne r10,r0,-3

final result	final result	final result
r1 = 13, r2 = 7, r3 = 1 r4 = 12, r5 = 1	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 IDs to be the name of your compressed file)

One submission for one team

8. Q&A

For any questions regarding Lab 1, please contact 林滆晨 (miz1205@gmail.com) and 潘儀芳 (sa69mo@gmail.com), or ask/post your questions in the corresponding discussion forum!

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

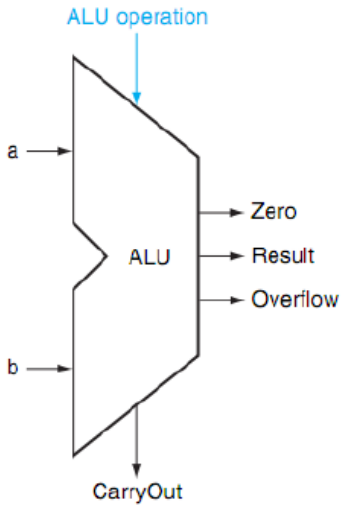


FIGURE C.5.14 The symbol commonly used to represent an ALU, as shown in Figure C.5.12. This symbol is also used to represent an adder, so it is normally labeled either with ALU or Adder.

```
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.