

# Computer Organization, spring 2017

## Lab3: Single Cycle CPU

**Due: 2017/05/12**

### 1. Goal

Based on Lab 2 (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. Requirement

- a. Please use Xilinx as your HDL simulator.
- 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., 0416001\_0416002.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. Top module's name and IO port reference Lab2

**Reg\_File[29] represents stack point. Initialize Reg\_File[29] to 128 while others to 0.**

You may add control signals to Decoder.

-Branch\_o

-Jump\_o -

MemRead\_o -

MemWrite\_o -

MemtoReg\_o

### 3. Requirement description:

#### 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  $PC = PC + 4 + (\text{sign\_Imm} < 2)$

### Jump instruction

jump is 1  $PC = \{PC[31:28], \text{address} < 2\}$

## 1. Code: (80%)

### Basic instruction:(50%)

Lab 2 instruction + mul、lw、sw、j

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]
-----------	---------------

instruction	Op[31:26]			
lw	6'b100011	Rs[25:21]	Rt[20:16]	Immediate[15:0]
sw	6'b101011	Rs[25:21]	Rt[20:16]	Immediate[15:0]
jump	6'b000010	Address[25:0]		

Mul is R-type instruction

0	rs	rt	rd	0	24
---	----	----	----	---	----

## Advance set 1:(10%)

instruction	op	rs	rt	rd	shamt	Func
jal	6'b000011	Address[25:0]				
jr	6'b000000	rs	0	0	0	6'b001000

### Jal: jump and link

In MIPS, the 31st register is used to save return address for function call  
 Reg[31] saves PC+4 and address for jump

Reg[31]=PC+4

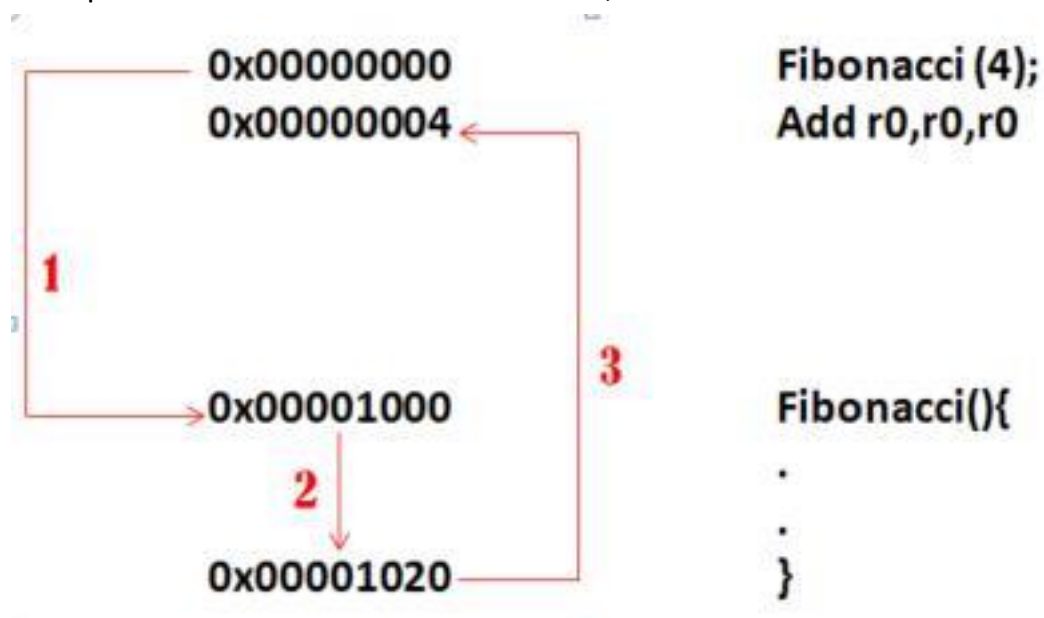
PC={PC[31:28],address[25:0]<<2}

### 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**

Example: when **CPU** executes function call,



if you want to execute recursive function, you must use the stack point ( **Reg\_File[29]** ).

First, store the register to memory and load back after function call has been finished.

The second testbench CO\_P3\_test\_data2.txt is the Fibonacci function. After it is done, r2 stores the final answer. Please refer to test2.txt

### Advance set 2:(20%)

ble(branch less equal than) → if( rs<=rt ) then branch

6	rs	rt	offset
---	----	----	--------

bnez(branch non equal zero) → if( rs!=0 ) then branch (It is same to bne)

5	rs	0	offset
---	----	---	--------

bltz(branch less than zero) if(rs<0) then branch

1	rs	0	offset
---	----	---	--------

li (load immediate)

You don't have to implement it, because it is similar to(and thus can be replaced by) addi.

0xf	0	rd	immediate
-----	---	----	-----------

## 2. Testbench

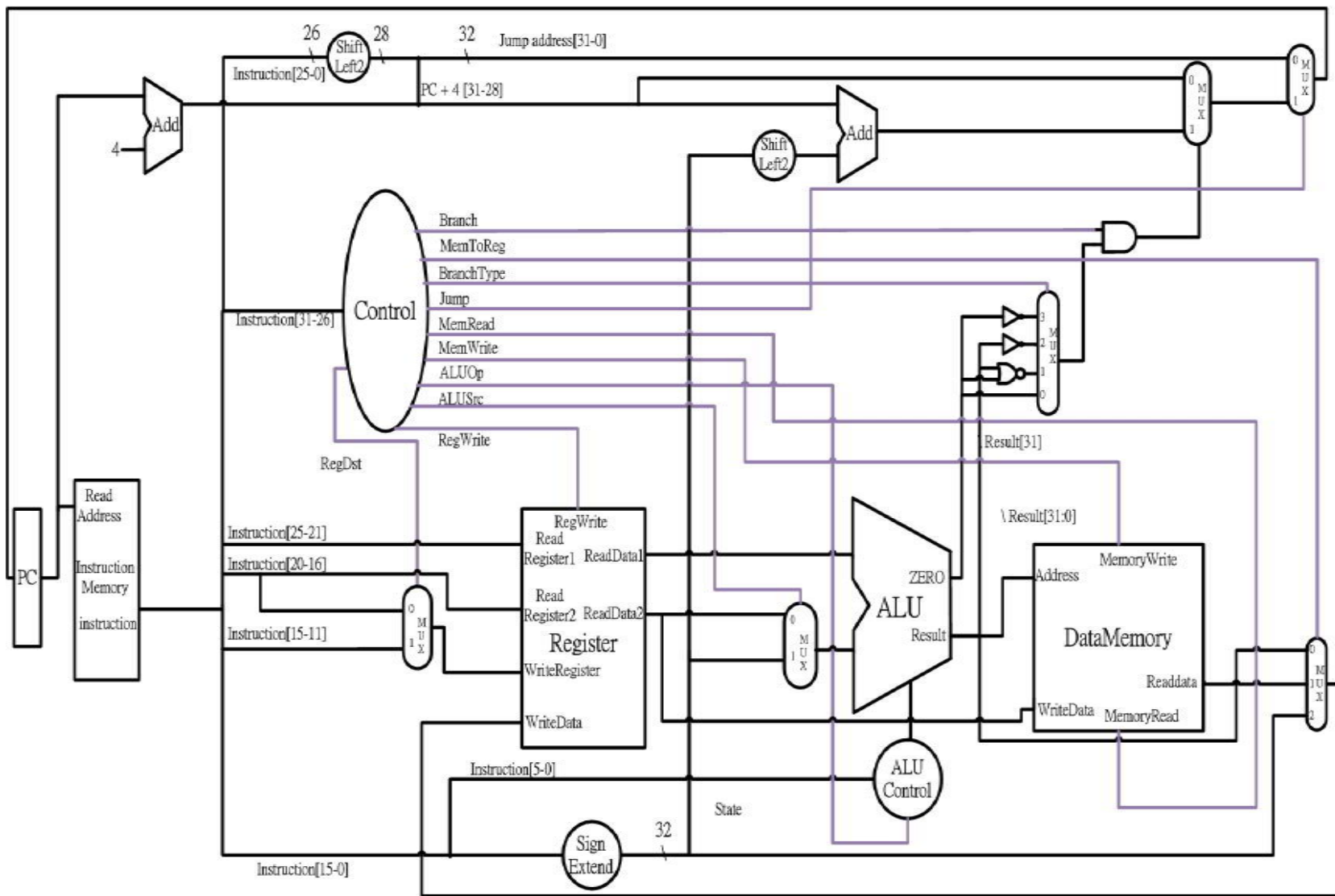
CO\_P3\_test\_data1.txt tests **the basic instructions (50%)** and CO\_P3\_test\_data2.txt tests **the advanced set 1 (10%)**.

Please refer to test1.txt and test2.txt for details. The following MIPS code is bubble sort. Please transform the MIPS code to machine code, store the machine code in CO\_P3\_test\_data3.txt and run it (for testing advance set 2 (20%)).

add	\$t0, \$0, \$0	sw	\$t2, 0(\$t0)
addi	\$t1, \$0, 10	sw	\$t3, 4(\$t0)
addi	\$t2, \$0, 13	li	\$t1, 1
mul	\$t3, \$t1, \$t1	no_swap:	
j	Jump	addi	\$t5, \$0, 4
bubble:		sub	\$t0, \$t0, \$t5
li	\$t0, 10	bltz	\$t0, next_turn
li	\$t1, 4	j	inner
mul	\$t4, \$t0, \$t1	next_turn:	
outer:		bnez	\$t1, outer
addi	\$t6, \$t0, 8	j	End
sub	\$t0, \$t4, \$t6	Jump:	
li	\$t1, 0	sub	\$t2, \$t2, \$t1
inner:		Loop:	
lw	\$t2, 4(\$t0)	add	\$t4, \$t3, \$t2
lw	\$t3, 0(\$t0)	beq	\$t1, \$t2, Loop
ble	\$t2, \$t3, no_swap	j	bubble
		End:	

### 3. Reference architecture

This lab might be used extra signal to control. Please draw the architecture you designed on your report.



#### 4. Grade

- a. Total score: 100 pts. **COPY WILL GET A 0 POINT!**
- b. Basic score: 50 pts. Advanceset1: 10 pts. Advanceset2: 20 pts
- c. Report: 20pts – format is in CO\_document.docx
- d. **Delay: 10%off/day**

#### 5. 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.**