

Computer Organization, Spring 2017

Lab 3: Single Cycle CPU

Due: 2017/5/11

1. Goal

In this Lab, we add memory unit to the CPU you created in Lab2 to implement a complete single cycle CPU which is able to run R-type, I-type and jump instructions,

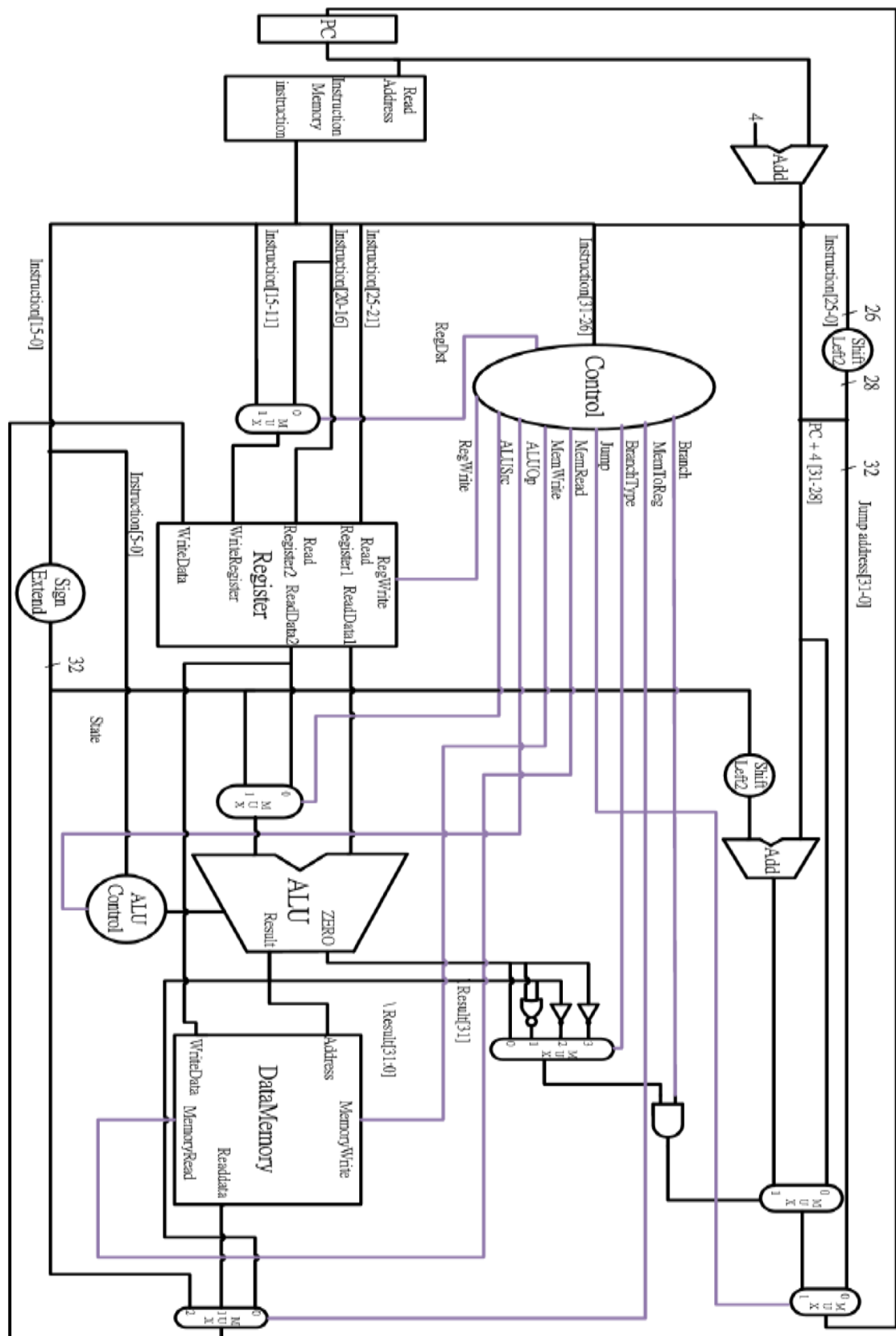
2. HW Requirement

- (1) Please use **Xilinx ISE** as your HDL simulator.
- (2) Please attach **your names** and **student IDs** as comment at the top of each file.
- (3) Please use the Testbench we provide you.
- (4) **PLEASE FOLLOW THE FOLLOWING RULE!**
 1. **Zip** your folder and submit ***.zip** file.
 2. **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!**
 3. A team's submissions **must** be uploaded **by the same person**.
 4. If one **violates** the rules above, **score** will be **deducted**.
- (5) Reg_File[29] represents stack point initialized to 128, others are 0.
You may add these control signals to decoder: Branch_o, Jump_o, MemRead_o, MemWrite_o, MemtoReg_o
- (6) Basic instruction set **(50%)**

All instructions in Lab2 and the following should be implemented.

Instruction ^o	Example ^o	Meaning ^o	Op field ^o	Function field ^o
LW(Load Word) ^o	lw r1, 12(r2) ^o	r1=MEM[r2+12] ^o	35 ^o	- ^o
SW(Save Word) ^o	sw r1, 12(r2) ^o	MEM[r2+12]=r1 ^o	43 ^o	- ^o
J(Jump) ^o	j target ^o	PC={PC[31:28], target<<2} ^o	2 ^o	- ^o
MUL(Multiply) ^o	mul r1, r2, r3 ^o	r1=r2*r3 ^o	0 ^o	24(0x18) ^o

3. Architecture diagram



4. Advanced Instructions 1 (10 pts)

Instruction ^o	Example ^o	Meaning ^o	Op field ^o	Function field ^o
JAL(Jump and Link) ^o	jal target ^o	see below ^o	3 ^o	- ^o
JR(Jump register) ^o	jr r1 ^o	see below ^o	0 ^o	8(0x8) ^o

JAL:

3 ^o	address ^o
----------------	----------------------

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

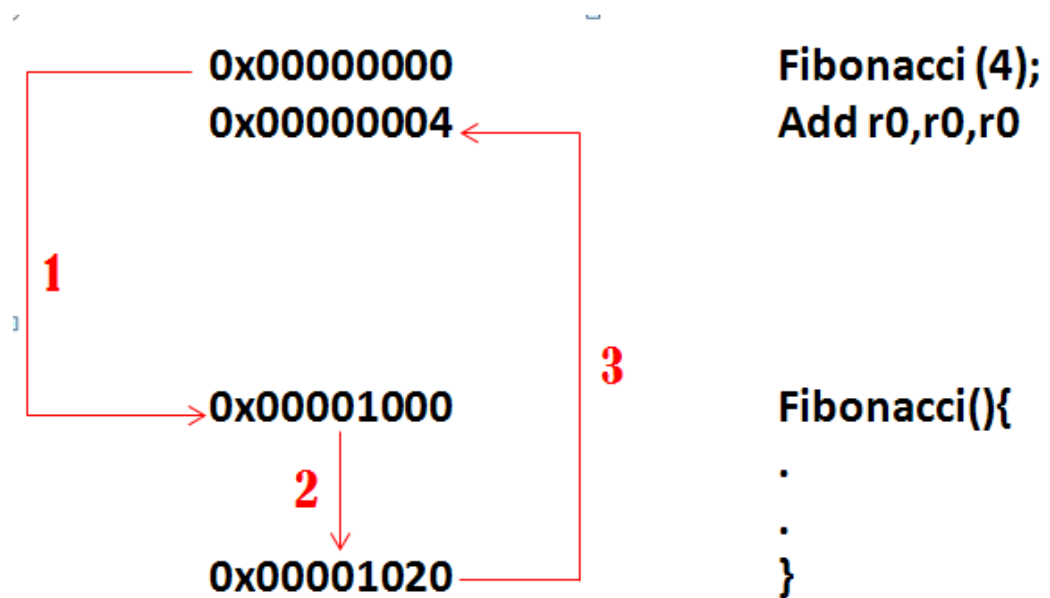
i.e., Reg[31]=PC+4; PC={PC[31:28], target<<2}

JR:

0 ^o	Rs ^o	- ^o	- ^o	- ^o	8 ^o
----------------	-----------------	----------------	----------------	----------------	----------------

In MIPS, return can be implemented by jr r31.

e.g., When CPU executes function call



If you want to execute recursive function, you can use the stack point (Reg[29]). Store the register to memory and load back after the function call is finished.

5. Advanced Instructions 2 (20 pts)

Instruction	Example	Meaning	Op field	Function field
BLE (Branch Less Than Equal)	ble r1, r2, 25	if($r1 \leq r2$) goto PC+100	7	-
BLT (Branch Less Than)	blt r1, r2, 25	if($r1 < r2$) goto PC+100	6	-
BNEZ (Branch Not Equal Zero)	bnez r1, 25	if($r1 \neq 0$) goto PC+100	5	-
LI (Load Immediate)	li r1, 1	$r1 = 1$	15	-

BLE:

7	Rs	Rt	offset
---	----	----	--------

BLT:

6	Rs	Rt	offset
---	----	----	--------

BNEZ:

5	Rs	0	offset
---	----	---	--------

LI:

15	0	Rd	immediate
----	---	----	-----------

6. Grade

- (1) Total: 100 points (plagiarism will get 0 point)
- (2) Document: 20 points
- (3) Late submission: 10 points off per day

7. Hand in

Please follow the rules! Zip your folder and name it as "ID1_ID2.zip" (e.g., 0416001_0416002.zip) before uploading to e3. Multiple submissions are accepted, and the version with the latest time stamp will be graded.

8. How to test

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↵	blt	\$t0, \$0, 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:↵	

CO_P2_test_data1.txt is for basic instruction and CO_P2_test_data2.txt is for advanced set 1. As for advanced set 2, please translate the bubble sort above to machine code, and test it on your CPU.

9. Q&A

For any questions regarding Lab 3, please contact 林濟晨 (miz1205@gmail.com) and 曾天鴻 (eric830303@gmail.com)