LAB # 9:
Follow the steps to test the Decode module of MIPS 32-bit Microprocessor using VHDL and implementation on FPGA

Objective

- To test the decode module that decodes the fetched instruction of a single-cycle MIPS 32-bit processor

Introduction

Testing the decode module is a lot more complicated than the previous module. There are two ways that could be done. The first is to test the fetch and decode modules independently by supplying appropriate signals to the input signals. The second method is to test two modules by connecting the decode module to the fetch module. You will use the second approach to ensure that your fetch module works correctly.

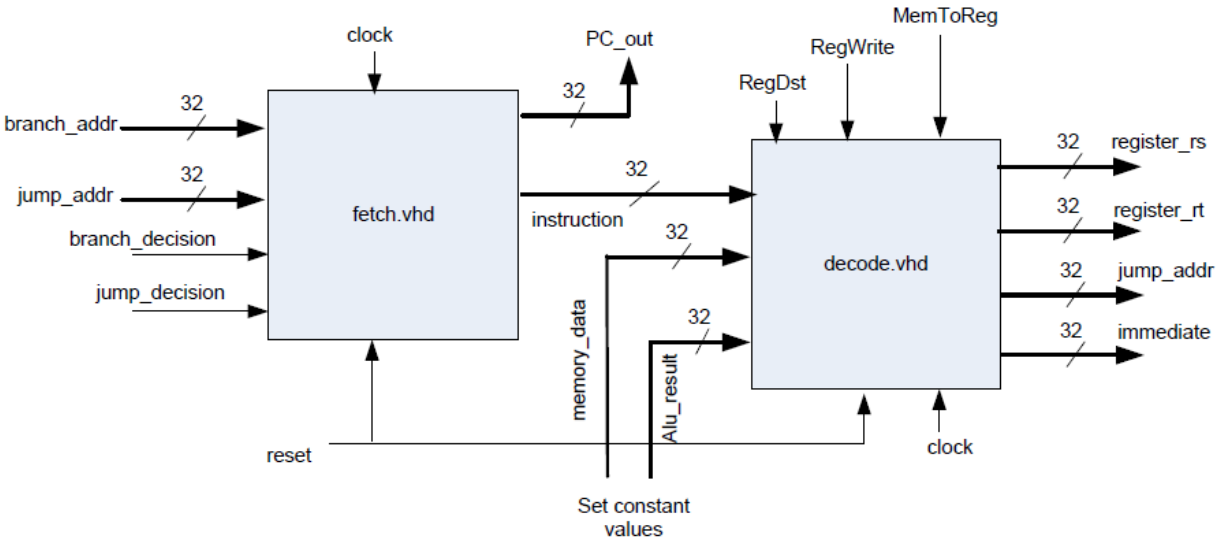
In-Lab Tasks

Lab Task 1: To test the functionality follow the steps

1) Modify the codes in the fetch module for testing
   (You may try different codes)

```
--initialize the instruction memory
variable mem: mem_array := (
X"8c220000",            --L: lw $2, 0($1) ; $2 <= mem[0+$1]
X"8c640001",            -- lw $4, 1($3) ; $4 <= mem[1+$3]
X"00622022",            -- sub $4, $3, $2 ; $4 <= $3 - $2
X"ac640000",            -- sw $4, 0($3) ; mem[0+$3] <=$4
X"1022fffb",            -- beq $1, $2, L ; if ($1=$2), branch_addr<=L
X"00612064",            -- and $4, $3, $1 ; $4 <= $3 and $4
X"08000000",            -- j L ; jump_addr <= L
X"00000000"); --
```

2) Make a port map with the ***fetch.vhd*** and ***decode.vhd*** module



3) Initialize the registers in ***decode.vhd*** as follows

```
if reset = '1' then
    reg0 <= x"00000000";
    reg1 <= x"11111111";
    reg2 <= x"22222222";
    reg3 <= x"33333333";
    reg4 <= x"44444444";
    reg5 <= x"55555555";
    reg6 <= x"66666666";
    reg7 <= x"77777777";
else
    ...
end if;
```

4) Display and switch setup

Note that there are four 32 bit outputs that must be verified. Use the four switches to control the selection of the outputs, i.e.

- SW0 =1 displays register_rs
- SW1 =1 displays register_rt
- SW2 =1 displays jump_addr
- SW3 =1 displays immediate

Connect the clock to a push button to create a slow manual clock.

5) RegDst, RegWrite, MemToReg control

At the top test module, RegDst, RegWrite and MemToReg signals are generated according to the instruction op code. It follows the following table. The complete version of this table can be found from the textbook

	R-type	lw	sw	beq
RegDst	1	0	x	x
RegWrite	1	1	0	0
MemToReg	0	1	x	x

```
--For the jump instruction,
if (opcode = jump) then
    jump_decision< = '1';
else
    jump_decision< = '0';
end if;
```

6) Instruction-by-instruction Verification

Consider the first instruction:

```
X"8c220000", --L: lw $2, 0($1) ; $2 <= mem[0+$1]
```

In this case, the source register is $1, the target register is $2, and the immediate is 0. Hence, you should see assuming you initialize the registers as in Step 3.

register_rs = x"11111111"
register_rt = x"22222222" -- disable storing to see the target reg value
immediate = x"00000000"
jump_addr = don't care

Consider another example which is an R-type instruction:

```
X"00622022", -- sub $4, $3, $2 ; $4 <= $3 - $2
```

In this case, since the sub instruction has a format "sub rd, rs, rt", you should expect to see the following:

register_rs = x"33333333"
register_rt = x"22222222"
immediate = don't care
jump_addr = don't care

Similarly, ***beq*** and ***j*** instructions can be verified. Create a list of expected output values of the decode module for each instruction and verify. Unfortunately, writing to registers cannot be tested easily unless we bring out the destination registers as another test output. One way of easy testing is to modify the reg_write process to replace the register_rt output with the register_rd signal and then you can use the same testing method as above to verify the decoding. Another way is to add one more interface signal register_rd as a part of the port and display it on the terminal. But you should remember that this interface has to be removed later because MIPS does not send this signal outside the decode module.

This lab is probably the most complex part of the MIPS-32 design. After this step is completed, the subsequent labs for MIPS-32 implementation will become easier.

7) Demo

Show the four outputs of the decode module using buttons, switches, and the Hyper terminal for each type of instruction.

Rubric for Lab Assessment

The student performance for the assigned task during the lab session was:			
Excellent	The student completed assigned tasks without any help from the instructor and showed the results appropriately.	4	
Good	The student completed assigned tasks with minimal help from the instructor and showed the results appropriately.	3	
Average	The student could not complete all assigned tasks and showed partial results.	2	
Worst	The student did not complete assigned tasks.	1	

Instructor Signature: _____ Date: _____