LAB # 12:
Follow the steps to reproduce the single cycle MIPS 32-bit microprocessor using VHDL

Objective

- To design the single-cycle MIPS 32-bit processor
- To test the R-Type instructions
- To test LW, SW and ADDI instructions
- To test BEQ instruction
- To test Jump instruction

Pre-Lab

Congratulations on your hard work and dedication to complete your own MIPS-32 processor implementation! Implementing a working processor is not a trivial task, but you have done it. You should be proud of yourself that you came this far, considering that you had no experience in either VHDL or processor design at the beginning of the class. In this lab you will complete the MIPS-32 processor design by adding all the modules together as shown in Figure 12.1

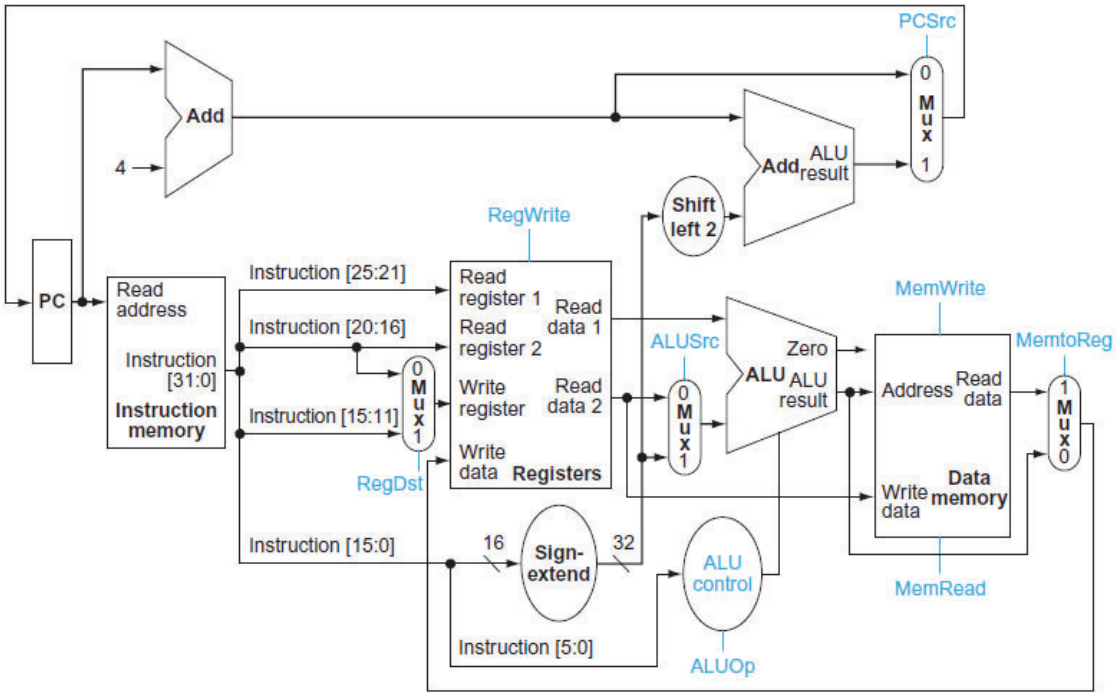

Figure 12.1 Complete MIPS microprocessor

Task 1: Make Module to Module connections

For all connections, a signal should be defined. This allows testing of the module-to-module functionality to be done conveniently by connecting them to LEDs and 7-segment displays.

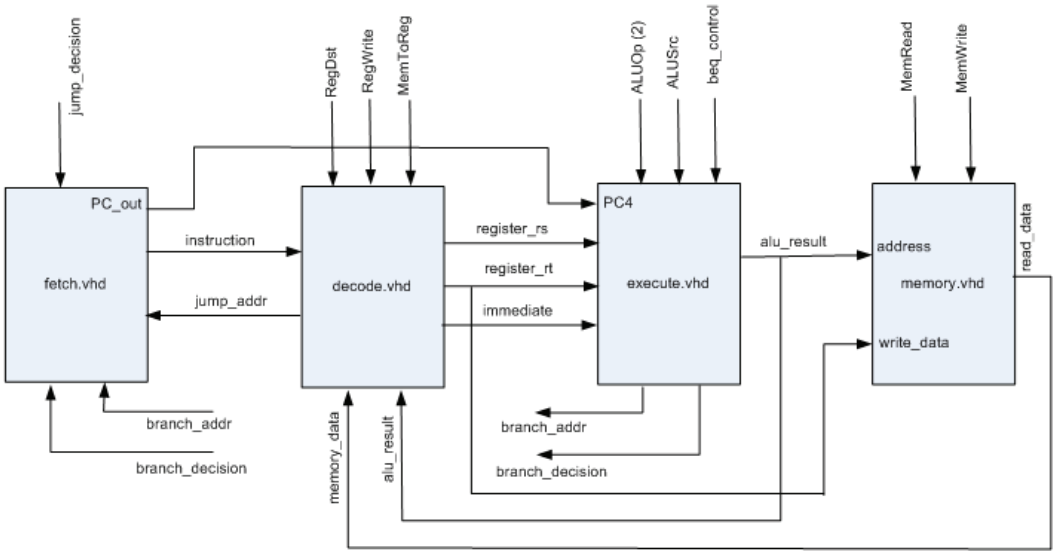

Figure 12.2 Module Connection

Task 2: Connections to switches, LEDs, and 7-segment

The Display unit will have 4 switches, eight 7-segment displays, a button (clock) and a 50MHz clock.

Switch status	To Display
0001	Displays **PC_out** from fetch.vhd
0010	Displays **register_rs** from decode.vhd
0011	Displays **register_rt** from decode.vhd
0100	Displays **Immediate** from decode.vhd
0101	Displays **alu_result** from execute.vhd
0110	Displays **read_data** from memory.vhd
0111	Display **branch_address**
Otherwise	Displays **Instruction** from fetch.vhd

In-Lab Tasks

We will be testing our MIPS 32-bit Single Cycle Microprocessor by completing the lab tasks incrementality (like we designed the Datapath). In order for the processor to work as a single cycle design, we need to incorporate some changes in our individual created modules. The single cycle ideology recommends that every instruction must complete in one clock cycle. To do that, do the following changes in already created modules

1) At every rising edge of clock, pick a new instruction from Instruction memory (fetch module)
2) The decode module has two processes, one for register read and other for register write. Register read process will be sensitive on instruction and reset only (no clock signal here), while the Register write process will write the values in registers at the end of clock signal, that is when clock'event and clock ='0'.
3) The Execute module should not be sensitive on clock, rather it should be sensitive on the values of register_rs, register_rt or immediate.
4) Memory process should also be exempted from clock sensitivity rather it should be sensitive on memory_address or memory signals.

Once the changes are incorporated, move on with Lab tasks.

Lab Task 1: Create the top-level entity

Create the wrapper file (MIPS.vhd) that has the following pins

Input

a) Clock
b) Button (will be used as clock, for testing only)
c) Reset switch

Outputs

a) Eight 7-segment displays

Lab Task 2: Test R-Type and ADDI functionality

This last lab is to test the MIPS machine codes as the final step. Update the instruction memory with the machine code of following instructions. Use green sheet to convert the assembly code into machine instructions

```
addi $1,$0,10
addi $2,$0, 20
add $3,$1,$2
sub $4,$2,$1
```

Lab Task 3: Test LW, SW commands

```
addi $1,$0,10
addi $2,$0, 20
add $3,$1,$2
sw $3,1($0)      --- Store at Data Memory location 1
lw $4,1($0)      --- Read the content of Memory location
```

Lab Task 4: Test BEQ and Jump instructions

```
      addi $1,$0,10
      addi $2,$0, 20
      add   $3,$1,$2
      beq   $1,$2,lblyes
      j lblend
lblyes:
      sub   $3,$1,$2
lblend:

      sw $3,1($0)      --- Store at Data Memory location 1
      lw $4,1($0)      --- Read the content of Memory location
```

Lab Task 5: Make module-to-module connections as shown

This last lab task is to test the MIPS machine codes as the final step. Write and test a MIPS assembly code that adds from 1 to 200 (C8hex) and then stores the result at a memory location on your own processor. This time, connect the clock to the system clock (50MHz) or a divided slower clock. In order to display the final result, use an infinitive loop of loads. This can be achieved through displaying the signal ***read_data*** to the 7-segment display output. The 7-segment display is used because it is a faster display technology.

Your assembly code should look like
MIPS code that computes 1+2+3+, …., + 200=20100=4e84hex.

```
L1: ….
beq   $1,$2, L2  --- "bne" can be implemented by beq and j combination
j     L1
L2:   [write your code here]
                      --- Assume that the addition result is in $5
      …

      sw    $5, 3($0)  --- Save the addition result at mem[3]
L3:   lw     $5, 3($0)
--- Create an infinitive loop to display the result stored in Memory
      j     L3               --- to the seven segment display.
```

Above method should allow you to test your processor's loop (branch) capability. One common mistake students make is not recognizing the instructions implemented. Remember that you only implemented lw, sw, rformat, beq, j, and addiu. Your program can use only these instructions. Another important feature you should test is the reset capability. Resetting your processor should produce the same result, that is, it should be able to repeat the execution of the addition program. This is because the processor starts from PC=0 when a reset is applied, and thus your code ends up executing the infinitive loop again.

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:** _____