**ECE 4250/ 7250: VHDL and Programmable Logic Devices  
Laboratory**

**Lab #8  
Lab Title: MizzouRisc Simulation with ModelSim**

**Group #2  
Group Names: Chris Smith, Benjarit Hotrabhavananda**

**Teaching Assistant Use Only:**

**Points Earned Reasons for Deduction**

**Pre-lab:**

**Post Lab report:**

**Demonstration:**

**Final Lab Grade:**

**Comments to students:**

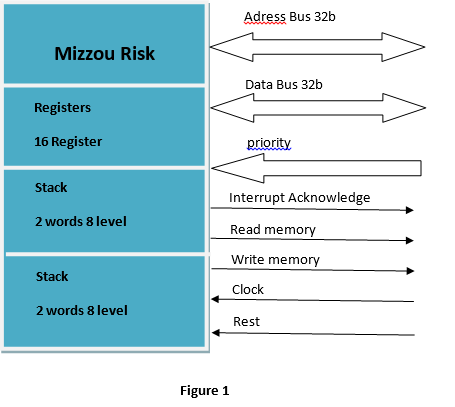
**Objective**

**The objective is to become familiar with MizzouRisc processor. In this lab, we learned to simulate the MizzouRisc by using ModelSim. We have programmed the processor with the machine language and add a memory to the processor to store data.**

**Introduction**

**Reduced instruction set computing, or RISC (pronounced 'risk'), is a CPU design strategy based on the insight that a simplified instruction set (as opposed to a complex set) provides higher performance when combined with a microprocessor architecture capable of executing those instructions using fewer microprocessor cycles per instruction. A computer based on this strategy is a *reduced instruction set computer*, also called *RISC*. The opposing architecture is called complex instruction set computing, i.e. CISC.**

**The Mizzou Risc architecture features an adress bus and data bus, each 32 bits wide. Externel control is provided to the processor by means of a 2-bit priority bus, clock, and reset lines. Figure 1 shows its architecture.**



**Implementation**

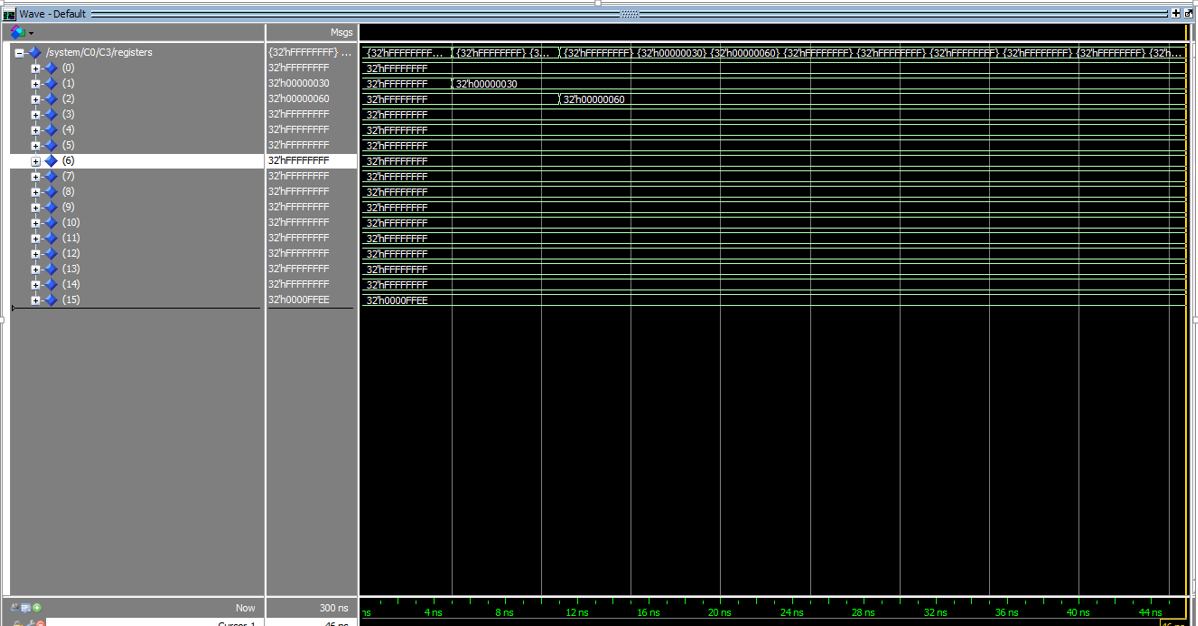
**1. Task1**

**We translate the following Assembly instructions into Machine language and put them into rom.vhd**

**load L#0030, R1**

**add R1, R1, R2**

**The results is in the figure below: R2 has #0060**

****

**Figure 2**

**2. Task2**

**We have created a new memory module to store data (ram.vhd), we wrote the program to generate the Fibonacci number(first10 numbers) and we did all the other changes which is needed such as adding ram\_select signal to the system and also a new component for mapping the ram.**

**Conclusion**

**We learned how to use the myzou risk and how can we do some changes to do what we need, for example we have created a memory module for read and write and also creating a rom which include all the assembly code of the instructions of the Fibonacci number and how to do some changes to run this program. We faced some errors but finally it has been run successfully.**

**Code**

**Rom.vhd for task1**

library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

use ieee.std\_logic\_unsigned.all;

use ieee.numeric\_std.all;

entity rom is

port (sel : in std\_logic;

address : in std\_logic\_vector(9 downto 2);

data : inout std\_logic\_vector(31 downto 0));

end rom;

architecture program1 of rom is

signal data\_out : std\_logic\_vector(31 downto 0);

begin

data <= data\_out when sel='1'

else "ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ";

process(address)

begin

CASE address(9 downto 2) IS

WHEN "00000000" => data\_out <= x"80000004";

WHEN "00000001" => data\_out <= x"81000002";

WHEN "00000010" => data\_out <= x"52100000";

WHEN "00000100" => data\_out <= x"00000000";

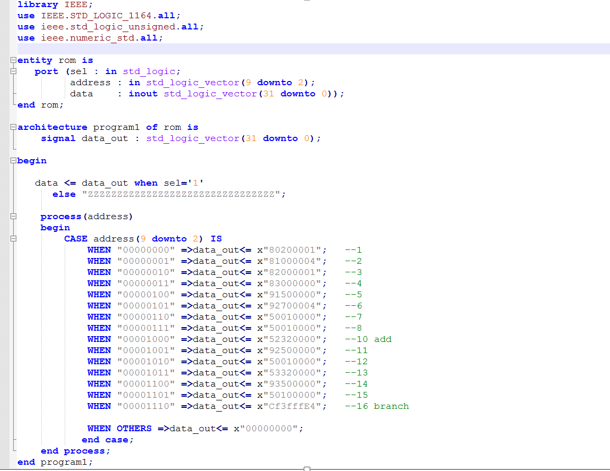
WHEN OTHERS => data\_out <= x"00000000";

end case;

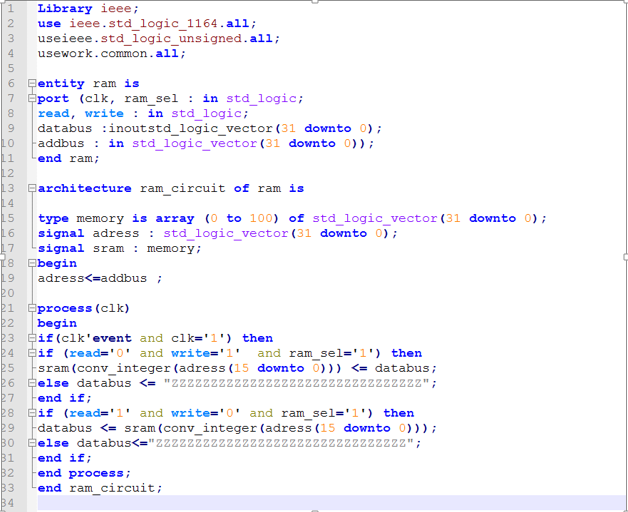
end process;

end program1;

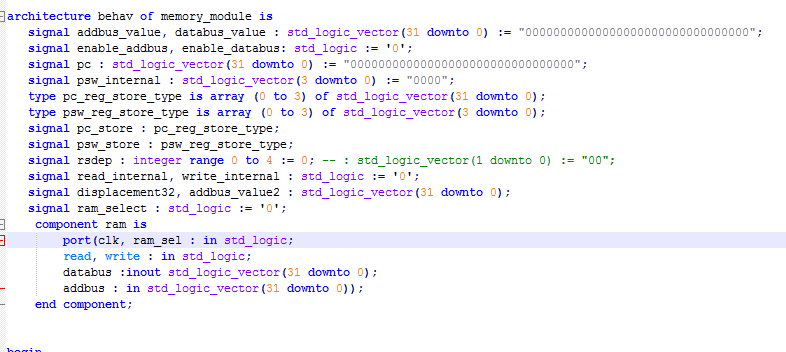
**Rom.vhd for task2**



**Ram.vhd**



**Code added to Memory\_module.vhd**



****