Computer Architecture Lab #4

Objectives

After this lab, the student should:

- Understand VHDL basics:
    - Declare New Types & use them.
    - Use memory in Modelsim

Requirements

Continue on your previous Lab, Let the Registers communicate with your memory.
use the last lab file to get your code up and running.
By the end of the implementation it should contain:

4 register components
A RAM component
Counter (it **decrements** by 1 each clock cycle, connected to the memory Address)

1. Connect Your Ram's data in and out to the bidirectional bus from last lab (You may need to add some try state buffers).
2. Connect Counter output to Your Ram Address
3. Draw the schematic diagram, think of the control signals needed
4. Modify **Ram to be generic** if needed and connect it with external signals.
5. **Initialize RAM with numbers starting from (700,699,698 ,…etc)**
6. Create a wave form simulation that would performs the following:
    - 6.1. reset all registers, and counter. (counter should be initialized with 10 upon reset)
    - 6.2. Load data from Memory M[10] to r0
    - 6.3. Load data from Memory M[9] to r1
    - 6.4. Transfer data from r0 to r3
    - 6.5. Store the data from r1 to Memory M[7]

Inputs (There are no outputs):

1. Decoders' selections & Enables (3 bits for each decoder, if the source decoder is disable, this means that the data source is the memory, also if the destination decoder is disabled this means that the destination is the memory)
2. one Clk
3. one reset

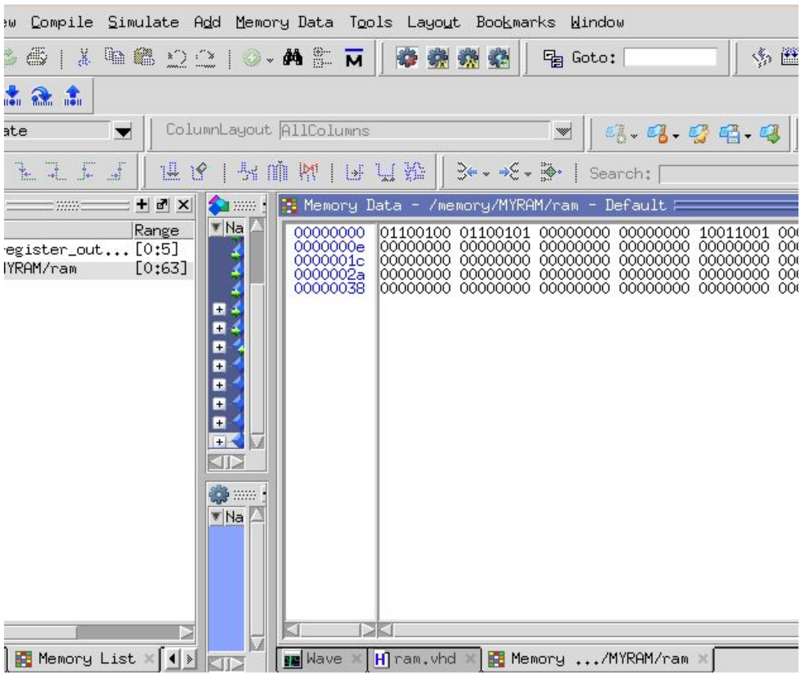
Guidelines

1. Be aware of **starting conditions (Initialization inputs & use "rst")** for your circuit to avoid going in an undefined state in case of contention on this inout bus.

2. If you want to initialize memory in your code you can use this type

```
SIGNAL ram : ram_type := (
  0     => X"C3",
  1     => X"38",
  2     => X"00",
 16#38# => X"C3",
 16#39# => X"00",
 16#3A# => X"00",
 OTHERS => X"FF"
);
```

3. You can initialize memory from UI too. From the view submenu, click on the "memory list", double click on the ram and you'll see a text area containing the values of RAM bytes, you can edit them in place if you right-click then choose edit. Another handy way is to export the file then import it again after you fill the whole bytes. To do so, right-click on the text-area again and choose export. Fill in the data of the exported file in whatever way you might want then save it in the same format and right-click again to import it. Voila!.
Note that you need to load it each time you start the simulation. You might need to add the



```
00000000 01100100 01100101 00000000 00000000 10011001 000
0000000e 00000000 00000000 00000000 00000000 00000000 000
0000001c 00000000 00000000 00000000 00000000 00000000 000
0000002a 00000000 00000000 00000000 00000000 00000000 000
00000038 00000000 00000000 00000000 00000000 00000000 000
```

loading of these data at the beginning of your simulation do file.
Note also that in real life we don't load RAM from files, it's just for simulation and learning purposes.

4. Sometimes it is advised to have 2 different clocks, one for the registers and one for the memory. To save clock cycles, make the 2 clocks reversed (e.g: one starts at 1 and the other starts at 0)

5. If you want to work from terminal (cmd) use the following commands

    a. vlib work #to create working library
    b. vcom *.vhd #to compile files
    c. vsim work.entityname(arch_name) #to run simulation for a certain entity
    or
    d. vsim -do test.do -logfile log #to run do files

Deliverables:

1. Schematic diagram using basic components (Decoder , Tri-State Buffers, Registers and other basic gates)
2. VHDL code.
3. Do file Simulation

Grading Criteria

Schematic
Implementation
Simulation

N.B. you will be graded for code neatness (indentations, variable names, …)

Good luck.