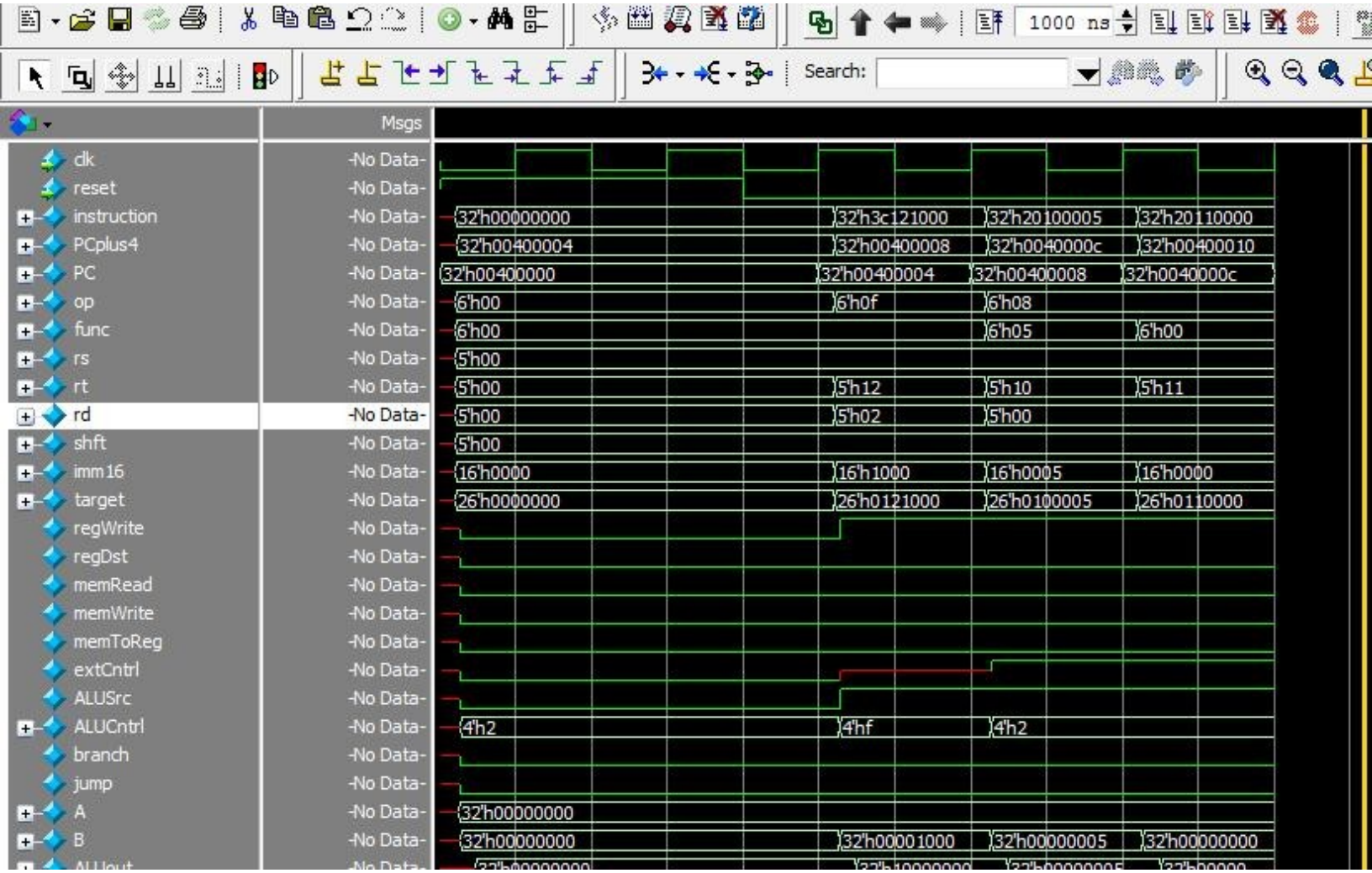
Project 2

1. Download P2-start.zip from Scholar and extract it to P2-start folder.
2. Open ModelSim and Select File->New->Project.
    a) In the dialog box that opens, type in the Project Name and select Project Location and Select ok.
    b) In the new dialog box which says “Add items to the Project”, click on Add Existing File->Browse and select all the files from P2-start folder. (**Important NOTE:** While adding files, ensure that you are adding files of all formats from the current directory)

       Select Copy to project directory. Select ok.

    c) Compile all the files. Select Compile->Compile All.
    d) Select Simulate->Start Simulation. Expand work->Select MIPS_TB. Select Ok.
    e) Now a new sim tab will open and here, right click on myMIPS->Add to->Wave->All items in region.
    f) A new waveform window should open. At the top, change 100ns box to say 1000ns and run it. You should be able to see waveform similar to the following screenshot



In the waveform window, check the signal instruction (on the left side) and see its waveform, we can observe instructions 00000000, 3C121000, 20100005, 20110000 represented as 32'h00000000, 32'h3C121000, etc. All the signal values –> PC+4 = 32'h400008, PC = 32'h400000, op = 6'h0F under 32'h3C121000 instruction are related to this instruction.

h) Open program.s with QtSpim.

```
[00400000]    00000000    nop
[00400004]    3c121000    lui $18, 4096
[00400008]    20100005    addi $16, $0, 5
[0040000c]    20110000    addi $17, $0, 0
```

We can see that program.txt contains the hex instruction format for the instructions in program.s. So in the waveform window, all the signal values under 32'h3C121000 are related to lui instruction. Similarly signals under 32'h20100005 are related to addi instruction.

3. What is actually happening?

When you select MIPS_TB after you simulate, the testbench runs and calls the following line

MIPS myMIPS(clk, reset);

Module MIPS is in mips.v file. Mips.v file instantiates all the modules of the single cycle data path. Some of the modules are already instantiated for you. For e.g, instruction memory, register file, PC, muxes etc. Some modules need to be instantiated depending on the instruction and so you will have to change to mips.v file. So, as all the modules are instantiated, all the signals will be instantiated. For e.g, if we take lui instruction which is 32'h3C121000, mips-decode decodes this and assigns values of rs, rt, op, func etc. accordingly to the following signals.

```
.instruction_in(instruction),
.op_out(op),
.func_out(func),
.rs_out(rs),
.rt_out(rt),
.rd_out(rd),
.shft_out(shft),
.imm16_out(imm16),
.target_out(target)
```

Similarly all the other signals are assigned values and we see the output waveform.

4. For this project, you have to implement add, sub, ori etc, lw, sw, beq, bne, j, jr etc (See Project2.doc).

What files to change?

a) Mips.v to include uninstantiated modules and any additional hardware as required.

b) Mips_control.v to add control signals to all the instructions (add, sub, lw, sw etc).

How to check the new instructions that you implemented?

a) Add your new instruction to program.s. For e.g add $s0, $s1, $s2. Open this new program.s with QtSpim.exe and get the hex instruction format and copy it into program.txt.

b) Now simulate your project in ModelSim and observe waveforms for your new instruction.

Things to note:

You will see red lines in the waveform window when the signal value is undefined or garbage.