Lab Worksheet 01

LWS01 LWS01

Part 1:

Main Objective:

1) To prepare a work environment for all Ensc 350 hardware design tasks.

Sub-Objectives:

1) To install Quartus Prime Lite.
2) To install ModelSim.
3) To understand the standard organisation strategy to be employed for all future design tasks in Ensc 350.
4) To learn how to configure necessary features of ModelSim.
5) To learn how to configure necessary features of Quartus.

Part 2:

Main Objective:

2) To prepare a work environment for all Ensc 350 system design tasks.

Sub-Objectives:

1) To download & install a simple RISC-V simulator.
2) To manually assemble instructions and verify the result.
3) To manually verify the behaviour of various RISC-V instructions.
4) To create simple RISC-V sub-routines.

Part 3:

Main Objective:

3) To become familiar with a typical workflow when performing one design iteration.

Sub-Objectives:

1) To create a VHDL description of an elementary combinational circuit.
2) To review VHDL statements for Behavioural, RTL & structural descriptions.
3) To use netlist viewers and the chip-planner to review synthesis results.

Lab Worksheet 01

Tasks: (Part 1) – refer to the document "LWS-01.x-SW-Install-Setup-350-1257.pdf"

Prepare your computer:

01) Install Quartus and ModelSim on you computer. (version 20.1.1, do not use the newer versions)
02) Verify that Quartus & ModelSim execute and that you have the correct versions.
03) Prepare a project filing hierarchy called "LWSTest". (*page 4 of reference document*)
04) Create project files.
05) Experiment with various settings to make your workflow efficient.

Tasks: (Part 2) – Learning RISC-V Instructions

Prepare your computer:

01) Install the ISA simulator on your computer.

- To download the simulator, there is a link on the Canvas Home page, "RISC-V Simulator.jar". The simulator is a Java Executable.
- You will need the Java Runtime Environment (**JRE**) – version 8 or higher installed on your computer to execute the simulator.
- JRE can be downloaded and installed from the **Oracle website**.

Practice Assembling RISC-V Instructions:

02) You will be allowed a printout of the document, "RISC-V(1.0)-reference.pdf" during exams.

- Using this document, assemble the following instructions.

```
        ADD     x27, x31, x0
        SRAIW   t3, x24, 0x1F
        OR      sp, t0, t1
        LB      x17, -1(t6)
        SH      x6, -2(t6)
again:  BNE     x0, x1, again
        LUI     x19, 0xBEEF1
        JAL     sp, again
```

- Check your results using the simulator.

Learn the Behaviour of "Execution Class" RISC-V Instructions:

03) Assume x2 = `0x FEED 0BAD A550 C0DE` and x3 = `0x 0015 DEAD BEEF F00D`.

- Predict the result in the destination register after execution of each of the following instructions.

```
Start:
        ADD     x4, x2, x3
        ADDW    x5, x2, x3
        SRAW    x6, x2, x3
        SLL     x7, x2, x3
        SLT     x8, x2, x3
        SLTU    x9, x2, x3
        LUI     x10, 0xC0DE5
```

Learn Register Initialization:

04) Initializing a register with a (word) 32-bit constant.

- Create a sequence of instructions to initialize register x2, with the value, SgnExt(`0x 89AB CDEF`).
- Do not use any other registers.
- Use only, LUI and "addition-type" instructions.
- Hint – look at the expansion of the (load Immediate) LI macro-instruction.

05) Repeat (04) , this time using only LUI and "bitwise-logic-type" instructions.

06) Initializing a register with a (double word) 64-bit constant.
Create a sequence of instructions to initialize register x2,
with the value, `0x 0123 4567 89AB CDEF`.
Do not use any other registers.
Use only, LUI , "shift-type" , and "addition-type" instructions.

07) Repeat (06) , this time using only LUI, "shift-type" , and "bitwise-logic-type" instructions.

Learn the Behaviour of "Memory Class" RISC-V Instructions:

08) The simulator places the default data segment (.data) beginning at memory address,
`0x 0000 0000 1001 0000.`
Select the "execute pane" to view/edit the contents of the data segment.

- x1 & x2 are initialized using the LI macro-instruction.
- Predict the contents of the data segment and registers after executing the instructions.
- Is this simulator BIG-endian or LITTLE-endian?
- Does this simulator allow misaligned memory access?

```
LI x1, 0xFEDCBA9876543210
LI x2, 0x0000000010010000

SD      x1, 1(x2)
LB      x3, 7(x2)
LBU     x4, 7(x2)
SW      x1, 16(x2)
LHU     x5, 18(x2)
SH      x1, 32(x2)
LHU     x6, 33(x2)
```

Writing Simple Loops using RISC-V Instructions:

09) Basic loops:
Write a program that calculates the Fibonacci number , F(n).
where F(0) = 0, F(1) = 1 and F(n+2) = F(n+1) + F(n) for all positive integers, n.

- Assume that n is a 64-bit unsigned number, initially loaded into x10.
- Place the computed result F(n) in register x11.
- Determine the largest value for n, such that $F(n) < 2^{64}$.

10) Simple algorithm:
Write a program that calculates square roots. $Y = \sqrt{X}$

- Assume that X is a 64-bit unsigned integer, initially loaded into x10.
- Place the computed result, Y in register x11.
- Assume that Y contains 32 integer bits and 32 fractional bits.
- Round the answer to the closest 32 bit integer.

Tasks: (Part 3)

Experiment with a various VHDL design methods:

01) Create a project filing hierarchy called "LWSmux"
02) On paper, draw a truth table for a 4-channel, 1-bit MUX.

03) Write separate Entity/Architecture pairs for primitive logic elements,
- 1-bit inverter,
- 2-input AND,
- 2-input OR

Implement these simple logic operations using (regular) signal assignments and IEEE logical operators.

04) Write a VHDL Entity that describes the circuit. Name the Entity, "MUX41"

05) Create a **structural** architecture for MUX41 using ONLY **instances of the primitive entities** created in (3).

06) Synthesise, then Place and Route.
- Synthesise and Fit the circuit for a Cyclone V FPGA.
- Review the resulting RTL netlist.
- Locate all elements using the chip planner and
- Review the individual equations for each LUT.

07) Create an **RTL** architecture for MUX41 using ONLY **selected signal assignment**.
08) Repeat (6), for this new architecture.

09) Create an **RTL** architecture for MUX41 using ONLY **conditional signal assignment**.
10) Repeat (6) for this new architecture.

11) Create an **behavioural** architecture for MUX41 using ONLY a VHDL **process** statement.
12) Repeat (6) for this new architecture.

13) Compare the results in (6), (8), (10) & (12). Write notes regarding your observations.

Lab Worksheet 01

LWS01 LWS01

Task ID	Show to TA during Lab Period for feedback and Lab Tick	**Demonstration** Required Signup for a demo time on Canvas	**Documentation** Required Canvas Submission	**Documentation** Required for feedback and Lab Tick
LWS-01-P1 (03)	☑	☒	☒	☒
LWS-01-P2 (01-08)	☒	☒	☒	☒
LWS-01-P2 (09, 10)	☑	☒	☒	☒
LWS-01-P3 (01-12)	☒	☒	☒	☒
LWS-01-P3 (13)	☑	☒	☒	☒