Computing and Communication Technology - FINAL PROJECT.

Your task in this project is to implement a simplified, single-cycle

MIPS CPU in Logisim.

The Project File

================

You will do your work in the CPU-PROJECT.circ file and, more specifically,

in the "CPU" circuit. The "main" circuit is a test driver, consisting of a

clock, instruction memory, and screen all connected to an incomplete CPU.

Your job will be to complete the CPU by editing the relevant subcircuit.

When you have completed the CPU, you will be able to load programs into the

instruction memory and run them.

Requirements & Testing

======================

Your only deliverable for this project is the completed CPU-PROJECT.circ file.

I provide a sample assembly program (jumpless.asm) that you can use to test your work. The program (jumpless.asm) is designed to contain no jump instructions, but it contains other basic instructions including unsigned addition/subtraction, lw/sw, slt, etc. If this program displays the correct output to the screen then your CPU can be considered correct.

This will require your processor to handle most types of instructions that we

have seen in this course. To see exactly which instructions

you must support, you can browse the jumpless.asm file.

Included in this archive is another sample assembly program (print129.asm). It

is more complex and contains various jump and branch instructions. To make this program (print129.asm) work, your CPU needs to be fully built.

If you want to earn extra credit, you can complete your CPU so that both of the programs display the correct output to the screen.

The Logisim-Diku Component Library

==================================

To aid you, I have included logisim-diku library published by a group at

the University of Copenhagen (https://github.com/andersbll/logisim-diku).

The library contains essential components, including

1) Instruction memory unit

2) 32-bit Register file

3) 32-bit MIPS ALU

4) Data memory unit.

The ALU and Register file follow the specifications described in lecture and

in the course text.

The Screen and Data Memory

==========================

To make testing more interesting and convenient, I have wired a Screen

component into the CPU so that the test programs can print out words and

numbers. I describe how this works below, but emphasize that this bears

little relationship to the mechanism underlying real-life I/O systems.

For information about general purpose I/O, you may read the relevant chapter

in your text book, as well as numerous sources on the internet.

For information about the silly little I/O mechanism we use for testing in this

project, examine the "Data Memory" subcircuit that is included in your

soon-to-be-completed CPU. This is a small wrapper around the data memory unit

provided in the logisim-diku component library.

The memory unit from the diku library takes a 20-bit address and can load or

store a 32-bit word at any address from 0 to 2^20-1. The wrapper takes a

32-bit address, rather than a 20-bit one. If the address is less than 2^20,

then each load or store operation interacts with RAM as normal. If the address

is 2^20 or greater, and the operation is "sw", then no store will occur, but

the bottom 7 bits of the input word will be sent to the screen, which will

interpret them as a character according to the ASCII standard.

Good Luck

=========

Have fun!