CSE140: Lab/HW #1
Machine Language

Overview

Since computer hardware can only communicate in 0's and 1's, our programs written in MIPS must be translated into machine code containing only 0's and 1's so that they can be executed. In this lab, we will review the conversions between MIPS and machine code.

Getting started

We will use MARS (our beloved MIPS simulator!) throughout this assignment. Feel free to read the attached documents (***MARS Tutorial.pdf, MARS features.pdf***) if you have not used MARS before or forgot about how to use it.

If you did not learn MIPS when you learned assembly language, please feel free to group with other students who have learned MIPS. You will pick it up easily with their help.

(Exercise) MIPS ↔ Machine Code

TPS (Think-Pair-Share) activity 1 Paired with the classmate(s) sitting next to you and perform the following tasks (You are allowed to form a group of three):

1. Record the name(s) of your partner(s).
2. Download ***"MIPS_Reference_Sheet"*** from CatCourses. We will need to refer to this sheet in order to complete all the exercises in this lab.
3. Load ***proc1.s*** in MARS and study the code.
4. After assembling the program, study the *Text Segment* window and see how your source code is translated into True Assembly Language (*Basic*) as well as machine code (*Code*).
5. In true assembly language, every single instruction can be translated into a machine instruction. How many bits does a machine instruction contain?

6. To utilize the limited number of bits efficiently, all machine instructions are categorized into different types (or formats). How many types are there? What are they? Give 2 operations for each type as examples.

7. Now, locate the instruction in ***line #7*** of ***proc1.s***. Let's translate this instruction into machine code.

    a. What instruction type is this? How many fields does this type of instruction have? What are the names of these fields?

    b. Refer to the **MIPS sheet**, what is the value of the ***opcode*** of this instruction in **Hex**? What register is ***rs***? What is the value of this register in **Hex**? What register is ***rt***? What is the value of this register in **Hex**? What is the value of ***immediate*** in **Hex**?

    c. Construct the machine code of ***line #7*** using the values obtained from **part b**. Write your answer in both **binary** and **Hex** formats. You can verify your answer with the *Code* column in *Text Segment* window.

8. Now, let's convert a machine code to a MIPS instruction. Locate address **0x00400010** from the *Text Segment* window.

    a. What is the machine code at this address in **Hex**? Convert this code into **binary**.

    b. From the binary version of this machine code. What is the instruction type? How can you tell? How many fields are there in this instruction type? What are the names of these fields?

    c. According to the binary machine code, what is the value of each field in **Hex**?

    d. Refer to the MISP sheet, what operation is this instruction? How can you tell? What is the mapping of the registers being used in this instruction?

    e. What is the final MIPS instruction? Is it the same as the *Source* column in the *Text Segment* window?

9. Now, let's take a look at ***line #17*** of ***proc1.s***.

    a. What format is this instruction?

    b. What are the values of ***opcode***, ***rs***, and ***rt*** of this instruction in **hex**?

    c. What is the name of the ***target*** label if it takes the branch? What is the address of this label in **hex**? (Hint: you can find it in the *Text Segment* window.)

   d. So, do we put this address as the value of the ***immediate field*** of the instruction? Why?

   e. How do we find the value of the ***immediate field***? What is this value?

   f. What is the machine code of this instruction in **binary** and **hex** formats? Does your answer match the Code column in the Text Segment window?

10. Finally, let's convert the j instruction in ***line #20***.

   a. What format is this instruction? How many fields are there in this format?

   b. What is the opcode of this instruction in **hex**?

   c. What ***label*** and ***address*** does this instruction jump to?

   d. How many bits can you use in the address field of the instruction? How can we "squeeze" the address into this field? What are the reasons behind this approach? What is the value of the address field in **binary**?

   e. What is the machine code of this instruction in **binary** and **hex**? Is it the same as what's in the *Code* column of the *Text Segment* window?

(Assignment 1, individual) Conversion in proc2.s

Convert the following line in ***proc2.s*** to machine code and then back to MIPS instructions:

Line #7

Line #14

Line #17

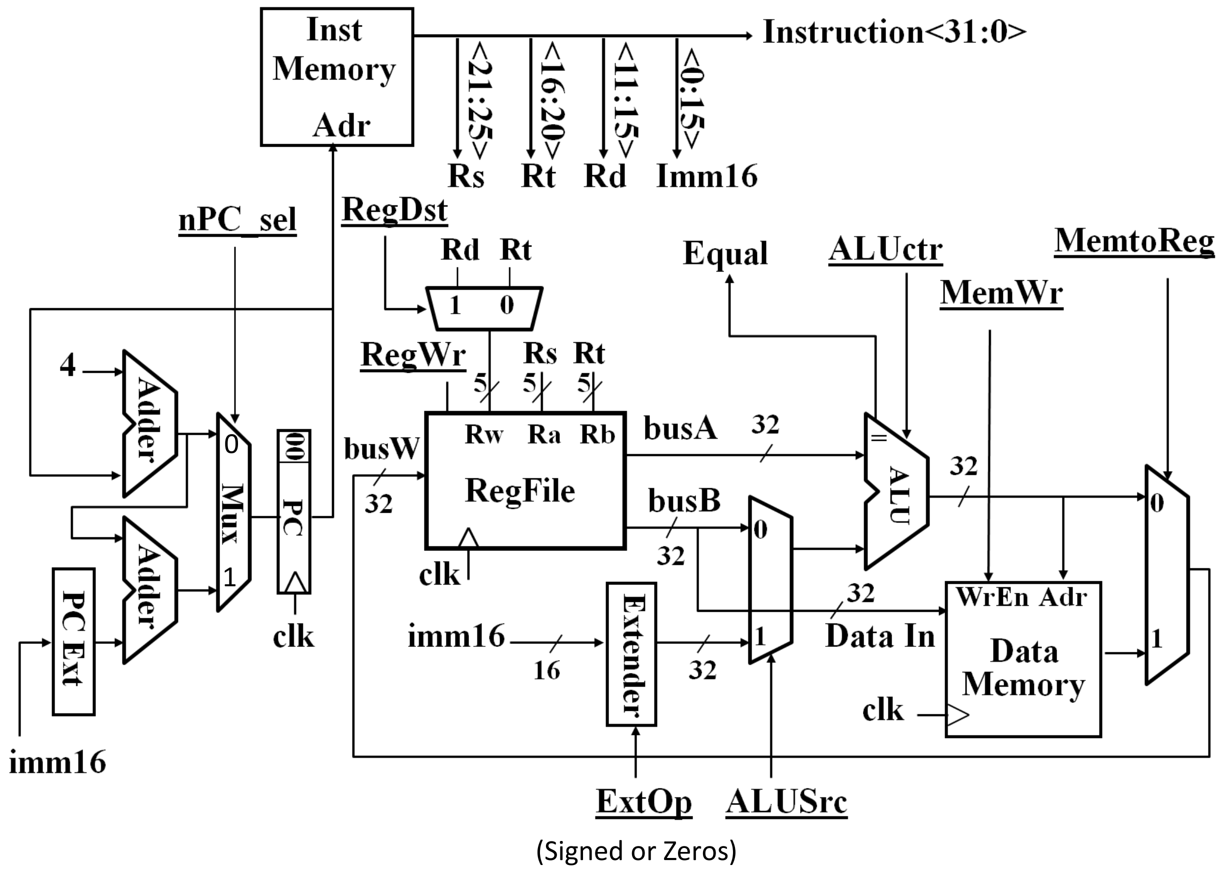
Line #20

You must show all the steps including values of the instruction fields in order to receive points.

Verify your answers with the *Text Segment* window.

(Assignment 2, individual) Datapath

Trace through the paths the execution of **SLTIU** instruction in this single cycle datapath design. Fill in the appropriate control signal values in the table. If there are any modifications needs to make it work, draw them in the diagram and create appropriate control values.



nPc_sel	ExtOp	ALUsrc	ALUctr	MemWr	MemtoReg	RegDst	RegWr

What to submit

When you are done with this lab assignments, you are ready to submit your work. Make sure you have included the following ***before*** you press Submit:

- Your answers to ***assignments***, ***TPS activities*** in a *text file*, and a list of Collaborators.
- Your assignment is closed **7 days after this lab is posted** (at 11:59pm).