Programming Project

Part 1
University of Nevada, Reno
CS 219 - Computer Organization
Instructor: Dwight Poplin
Teaching Assistant: Shawn Ray

Due before 11:59pm, January 28th 2022

Contents

1 Theory

In computer programming, instructions are broken down from higher level languages into more machine interpret-able forms. An example of such would be designing a computer program in *C* and once you compile the program it will be compiled into forms the machine can understand. In simple terms this will be described as assembly. Assembly language is one step above machine code, or binary.

Assembly language commands will follow the format of:

operation operand1 operand2

This is the basic form of the operations that will be discussed. The result of the operation between the two operands will be returned to the machine. HINT: The return of the resulting operation may not always happen behind the scenes.

2 Implementation Details

You are tasked with designing a machine language simulator. This project must be written in C or C++. If you elect to use C++ be careful with your objects. Your code should be versatile enough to be modified for future assignments with more than one operation. For part 1 you must only implement the **ADD** operation.You will be given a text file for input on this project; following the format discussed in the Theory section. An example operation will be in the form

ADD 0x12345 0x678

When the operation to be performed is addition and it is adding the two numbers 0x12345 and 0x678. These numbers will be hexadecimal format. Recall that the prefix of hexadecimal is always "$0x$" and this is not to be considered in any conversion process. You might find it useful to use the *uint32t* in C/C++ to store the number as an unsigned 32-bit number.
Using the provided text file as an example input. You will be tasked with creating a program that will read the command (ADD, but future projects may have more) and it will perform the addition between the two hexadecimal numbers. HINT: You may find it helpful to store this value in a variable, should you ever need to retrieve the information or store it in a simulated register. But this is not required for part 1. The result of the addition operation must be displayed to the terminal or build environment.
Your code should follow proper code commenting and indentation procedures.

Please refer to the documents on canvas for help on formatting.

3 Submission Requirements

For submission of this assignment, you are expected to at minimum submit your working code. If you elect to use a specific build environment please pick something simple to aid the compilation done by the TA. For example, you may use Visual studio, Clion or anything else. If you choose a specific build environment. It must be detailed in your *README* file, and must include any specific build files required to compile. Should you wish, you will provided a standard makefile on canvas, and a detailed how-to on how to use *Linux Make*. All submissions are expected by the due date, submission is done on canvas. No extensions will be given except as permitted by the course Syllabus.

4 Grading

30 points

30 points will be distributed based on working code. As detailed in this document.

30 points

30 points will be distributed based on properly documented code. As detailed in this document.

30 points

30 points will be distributed based on whether or not you solved the task you were given.
Did you properly read the commands from the text file?
Did you properly display the result?
Is the result correct?
Are you wrongly calculating values with the prefix?
Please Refer to implementation details.

10 points

10 points will be given for properly handling data types and hexadecimal numbers. Please refer to Implementation details.