Virginia Tech ECE 5484: Fundamentals of Computer Systems Summer 2020

Project 2 Assignment

Preface

Before starting this project, please be sure that you have completed all of the following activities.

- Read Chapter 4 of the textbook and view the associated online lectures to gain an understanding of the MARIE computer organization, instruction set, and assembly language programming constructs. Note that you can utilize any of the instructions in MARIE's full instruction set.
- Review the course syllabus and be sure that you understand grading and submission policies.
- Review the course schedule to understand the due dates for this and other assignments.
- Review the Graduate Honor System at https://graduateschool.vt.edu/academics/expectations/graduate-honor-system.html. Review the Graduate Honor System Constitution, especially Articles I (Sections 1, 2, and 3), V, VI, VII, VIII, and IX.
- This project should be completely your own work. You are welcome to discuss high-level aspects of the project with others and you are encouraged to the Project discussion area on the class website for this. Any questions concerning design details must be directed to the instructor or graduate teaching assist - This project should be completely your own work. You are welcome to discuss high-level aspects of the project with others and you are encouraged to the Project discussion area on the class website for this. Any questions concerning design details must be directed to the instructor or graduate teaching assist
- Download the MARIE simulator (MarieSim.jar), which allows you to observe the contents of registers and memory while running a program. The MARIE simulator is available for free at: http://www.jblearning.com/catalog/9781284123036/. It can be found on the Sample Materials tab on this page. It is required to complete this project assignment. Also, note that the Marie simulator requires Java version 5 or later to be installed on your computer. You will *not* be using the datapath or memory simulators that are also available at this site.

1. Introduction

The objective of this project is to reinforce your understanding of computer organization, instruction set architectures, and assembly language. You will accomplish this by writing, analyzing, and debugging an assembly language program for the MARIE processor.

You must:

i) design and write an assembly language program for the MARIE processor that inputs, transforms, stores, and then outputs a sequence of characters from the set A-Z;
ii) debug and test your program by simulating it using the MARIE simulator;
iii) document your work in a short report; and
iv) submit the report file (*.pdf), assembler source file (*.mas), assembler listing file (*.lst), and assembler executable file (*.mex).

2. The MARIE Simulator

The MARIE simulator is provided as a zip file containing Java archives (*.jar) files, documentation, and example source files. Unzip the file to a directory for use. Do the following to become familiar with the MARIE simulator.

- Read "A Quick Start Guide for the MARIE Machine Simulator Environment" (QuickGuide.pdf provided in the zip file).
- Depending on how comfortable you are with using the MARIE simulator after reading the quick start guide, you may also wish to read "A Guide to the MARIE Machine Simulator Environment" (MarieGuide.pdf provided in the zip file).

- Review the example assembly language source files and experiment with the MARIE simulator using these examples. The Ex4_3.mas and Ex4_4.mas examples are likely the most relevant to this assignment.

3. Design Specification

You are to design, write, test, and debug a MARIE assembly language program that inputs a sequence of characters from the set A-Z (capital letters only), stores each character in memory after it is transformed by the trivial ROT13 cipher, and then, after character input completes, outputs the transformed characters.

A template source code file (Project-2_Start.mas) is provided with this assignment. Edit this file to create a program that meets the program specifications. Note that the template includes instructions to initialize some working values that your program can use. The template also defines memory locations. You may add data memory locations. The program can be designed without additional data locations, but it may be necessary to do so for your design.

For full credit, your solution must perform the functions and satisfy the requirements specified below.

a) The first instruction of the program must be placed at location (address) 0x100 (100 hexadecimal) in MARIE's memory. This is accomplished by following the program template that is provided.
b) The constant data values (One, ChA, ChZ, ChPe, Val13, Start) should not be changed by the program. The program can load from these memory locations, but should not store to them.
c) Transformed input characters must be stored in successive memory locations beginning at location 0x200 (200 hexadecimal) as indicated in the program template. The program should store all transformed input characters before any characters are output.
d) The program should always initialize the values for Ptr in the working data memory and not rely on the values for these locations that are defined in the assembly source file. This initialization is done by the provided template file.
e) The program should work for any inputs 'A' through 'Z' and '.' (a period terminates input). In the interest of keeping the program simple, the program does not need to validate inputs.
f) When transformed characters are stored and when transformed characters are output, the program must use a loop and indirect addressing to access the values in the array of words. Note that variable Ptr is initialized in the template code and should be used in the loop. You may also define a Count variable to count the number of characters, but there are also correct designs that do not require a Count variable.

The program should operate as follows.

Input Phase:

1. A character (A-Z or '.') is input. MarieSim allows the user to input a single character that is read into the accumulator (AC) with an Input instruction.
2. If character '.' (period) is input, then the input phase ends and the output phase begins (step 5 below). (The period may be stored in memory to mark the end of the characters or the characters can be counted to determine how many transformed characters to output during the output phase.)
3. The character that is input is transformed using the trivial ROT13 cipher (see Section 5.1).
4. The transformed character is stored in the next location in the block of memory beginning at location Start. (Variable Ptr must be updated and indirect memory addressing must be used.)

Output Phase:

1. All transformed characters are output, beginning with the first character that was transformed. The '.' character is not to be output. (This will require a loop using variable Ptr and indirect addressing. Note that the number of characters to output will vary and the program must know when to stop the output by relying on a '.' or other special character in memory, counting the number of input characters during the input phase, or some other method.)

2. After all characters are output, the program halts by executing the HALT instruction.

Make sure you project has both distinct phases. You could, but don't have to, use the idea of "subroutines" or "functions" to handle each distinct phase.

4. Testing

Test and debug the program using the MARIE simulator (MarieSim.j Debug the program using the "Step" and "Breakpoint" features of simulator. You must test your program with the following two test cases.

Test 1: Input the eight-character sequence "VIRGINIA" followed '.' to terminate the input. Note that you need to input one character a time into MarieSim's ASCII Input area, with each character followed pressing the "Enter" key. The ROT13 value of each character ("IVETVAVN" should be displayed after the '.' character is input.

Test 2: Rerun the program in MarieSim, without reassembling or reloading, then input the four-character sequence "GRPU" followed '.' to terminate the input. Note the output.

When you create your source file within MarieSim (using the File > menu pick), use file name *lastname_firstname*_P2.mas, where "lastname" is your last or family name and "firstname" is your first given name. You can assemble your source file in the editor program. The assembly process creates a listing (*lastname_firstname*_P2.lst) and an executable (*lastname_firstname*_P2.mex). Load the executables file into simulator for execution.

5. Design Notes

5.1. The ROT13 Cipher

The ROT13 cipher (see http://en.wikipedia.org/wiki/ROT13) is an but trivial cipher that simply rotates the characters by 13 positions. For example, 'A' is transformed to 'N', 'Z' is transformed to 'M', and so on.

The Project-2_Start.mas source file includes a ROT13 subroutine *almost* performs this transformation. You need to fix one bug in subroutine.

5.2. Tips for Program Design and Debugging

Here are some suggestions to keep in mind as you design and implement your program.

- Design and test your program in an iterative manner, building from simple functionality to full functionality. For example, first write a program that inputs characters and just stores them in memory. Then, add the code to transform the characters using ROT13 before they are stored (and fix ROT13). Then complete the program by adding code to output the transformed characters.
- Study sample code in the textbook. For example, Example 4.4 (page 260) provides code that traverses and outputs a string. (Sound familiar?) Example 4.5 (page 260-261) shows code that calls a subroutine. The operand used with the `Skipcond` instruction is a source of frequent errors, so study the examples, such as Example 4.4 (page 260), to be sure you understand how to specify the operand for `Skipcond`.
- Correct solutions for the project require that about 25 to 30 instructions are added to the template. If you find that you are using significantly more instructions than this, you should reconsider your design and, as needed, consult with the GTA and/or instructor.
- When debugging your program, set a breakpoint in MarieSim to execute past the input operation and then single-step through the program to ensure the code is doing what you want it to do.
- When debugging and testing, be sure that "ASCII" is selected for the input and output windows in MarieSim.
- An "Instruction Set Cheat Sheet" can be displayed from the "Help" menu in the MARIE Assembler Code Editor.

6. Submission

6.1. Report

You must document the design, simulation, and outcomes in a brief written report. Your report should contain the following items.

- At the top of the first page of your report, include: your name (as recorded by the university); your email address; and the assignment name (e.g., "ECE 5484, Project 2"). Do *not* include your Virginia Tech ID number or your social security number.
- The body of the report must contain the following sections. Use section numbers and headings to organize your report.

Section 1 – Objectives: Provide a brief summary of the design objectives and general approach to the design.

Section 2 – Design Description: Describe the high-level operation the program. In particular, briefly describe initialization and operation of the loop.

Section 3 – Testing and Results: Give a summary of the two tests and associated outcomes that you obtained using the MARIE simulator. Pro images of one or more screen captures for Test 1 described in Section 4 above. Your screen capture(s) should show the final results in output area of MarieSim, and the contents of memory locations 0 through 0x20F from the memory display in MarieSim.

Section 4 – Conclusions: Briefly discuss the outcome of your de and any problems or aspects that do not work properly; what you lea by doing this project; and any experiences that were particularly or bad. Also, specify the approximate number of hours that you dev to the project. (The number of hours is just for the instructor to assess the suitability of this project assignment.)

Your writing should be well-organized, concise, and technical in nature. Your report should use complete, grammatically correct Eng sentences. Use section headings within the report that match the sec names listed above. Every figure and table should have a caption should be introduced in the body of the report.

6.2. Submission

Carefully follow these instructions when submitting your project.

- Create a single PDF file for your report. Name the PDF file *lastname_firstname*_P2.pdf, where *lastname* is your last or family name and *firstname* is your first or given name.
- Create the source file with file name *lastname_firstname*_P2.mas. When the source file is assembled, the assembly process will create an executable file (*lastname_firstname*_P2.mex) and a listing file (*lastname_firstname*_P2.lst). Save the *.mas, *.lst, and *.mex files for submission.
- Submit the four files (*.pdf, *.mas, *.lst, *.mex) in the Assignments area of the class website by no later than 11:55 p.m. on the due date.

7. Grading

The project will be evaluated based on the following criteria.

- Presentation (20 points)
  - Complete, clear, and well organized report
  - Mechanics (spelling, grammar, etc.)
- Technical Merit (30 points)
  - Discussion of design objectives
  - Discussion of program operation
  - Discussion of testing
  - Conclusions
- Correct operation (50 points)

- Correct operation as verified using the executable (*.mas) file

8. Seeking Assistance

This is *not* a team project. Your project should be completely your work. You are welcome to discuss high-level aspects of the project others, and you are encouraged to use the Canvas discussion forum this purpose. Any questions concerning design details must be dire to the instructor. Please refer to the Honor Code statement in syllabus.