EE371 - PROGRAMMING PROJECT #1, FALL 2021

Douglas V. Hall
Electrical and Computer Engineering Department
Portland State University
Portland, OR 97207

ARRAYS AND PROCEDURES

REFERENCE: Hall Chapters 3 and 4 along with class discussions.

IMPORTANT NOTE: Read ALL of the project description, the list of deliverables, and the required signed statement of individual effort before doing any work on the project.

Introduction:

This exercise will give you some experience using many of the basic techniques in ARM assembly language programming. Specifically, you will gain experience with a procedure call and return, passing parameters to procedures, processing arrays with pointers, auto-increment addressing, conditional branches, converting an algorithm to assembly language, and the program development cycle. This assignment will also give you some practice following the "Fast is Slow" rule and convincing yourself that it really works.

Problem Statement:

For this project, assume that a system is set up with a temperature sensor and an Analog to Digital converter to measure the Fahrenheit temperature every 1.5 hours in each 24 hour period. The A/D converter produces 8-bit binary values that represent the temperature in Fahrenheit. Temperature values range from 0 – 125 degrees Fahrenheit. The 8-bit binary values for the 16 temperatures taken each day are stored in an array of bytes named Fahrenheit_Temps. Your task will be to perform some calculations on the 16 temperature values in the array. (You will set up an array with 16 8-bit test values.)

Your two main tasks will be:

1. In your first version program you calculate the rounded average of the 16 temperature values and put the result in a named memory location.
2. When the average version is working correctly, you develop and call a procedure that determines the minimum and maximum temperatures for the day and returns these to the mainline to be put in named memory locations.

Procedure:

1. Don't think any programming at this point, just think how you would calculate the rounded average of 16 values with a pencil and paper.

2. Using the **standard program structures** in Chapter 3 write an algorithm for the program section that reads the values form the array and determines the average of the 16 byte-sized temperature values.
3. Using the Multo program in the text as an example, think about how you can set up required data structures and translate each step in your algorithm to one or more assembly language instructions, as shown in the Chapter 4 examples.

4. Bring up CCS and create a source program with your instructions and data sections. Note that to test your program, you will have to initialize the Fahrenheit_Temps array with 16 trial values for which you can predict the correct binary results. (If you need to declare an array that is longer than will fit on one line, you can simply put a dummy name on the next line and type more elements of the array on the next line.) Carefully determine reasonable test values to put in your Fahrenheit_Temps array, so you are testing a range of values. Make sure to put in the Header, .global, _start, etc. and load addresses into registers as in Multo.

5. As described in the ARM Tutorial, edit, assemble, link, run, test, and debug this first version of your assembly language program.

6. When this first version of your program works, save a copy, and use this copy to work on for the second version of your program.

7. Now think about how you would use the pencil and paper method to read through the array one byte at a time and determine the maximum value and then the minimum value. Using Standard Programming Structures, write an algorithm for a procedure that will determine the maximum and minimum temperatures.

8. To develop the assembly language for this version of the program, assume the mainline passes the starting address of the array and the length of the array to the procedure and the procedure passes back the maximum and minimum temperatures in registers.

9. Determine the assembly language instructions you can use to implement each step in the algorithm for the procedure. Use the examples from class and APCS rules in text to determine which registers to use for passing parameters to and from the procedure.

10. To your copy of the program from the first part of the project, add the needed mainline instructions needed to set up the stack, pass parameters to the procedure, call the procedure and write the returned values to named memory locations t. Also add the procedure instructions, including those to push and pop the used registers, to this copy from the first part.

11. As described in the ARM Tutorial, edit, assemble, link, run, test, and debug your assembly program. To start, use a breakpoint to determine if execution is getting to the procedure and the passed parameters are available. Then single step and run down through the MAX-MIN section of the program to see if the correct results are being produced. If so, you can single step back to the mainline and check if the results are returned and written to memory correctly.

12. When your complete program works correctly, print out copies of the final source code and the appropriate Debugger memory display(s) to demonstrate that the program works correctly.

Deliverables:

The golden rule of documentation is, "Create the documentation you would like to receive, if you were assigned the task of adding features to the product."

Your documentation for the project should include:

A. A design log that shows the steps you took, any problems you had, how you solved these problems, and the results at each step.
B. A clearly written algorithm /task list for the first version of the project and a clearly written algorithm /task list for the second version of the project.
C. A discussion of how you tested the program thoroughly so you can convince your boss that it is marketable. (Refer to Debugger printouts and specific cases.)
D. A printout of the .s file for your program with header and full comments.
E. **Labeled** printouts of Debugger Memory Windows that show the program works.
F. A signed statement that "I developed and wrote this program by myself with no help from anyone except the instructor and/or the T.A. and I did not provide help to anyone else"

(Any evidence of joint work will result in project grades of zeros for all parties involved.)

GRADING KEY FOR ECE 371 DESIGN PROJECT #1 Doug Hall FALL 2021

LOG (DETAILED AND "AS YOU GO"). 20 MAX ________

TASK LISTS/ALGORITHMS
(DETAILED AND COMPLETE). 20 MAX ________

WORKING PROGRAM 40 MAX ________

COMMENTS (CLEAR AND USEFUL)	**10 MAX**	________
OVERALL ORGANIZATION	**10 MAX**	________
TOTAL	**100 MAX**	________