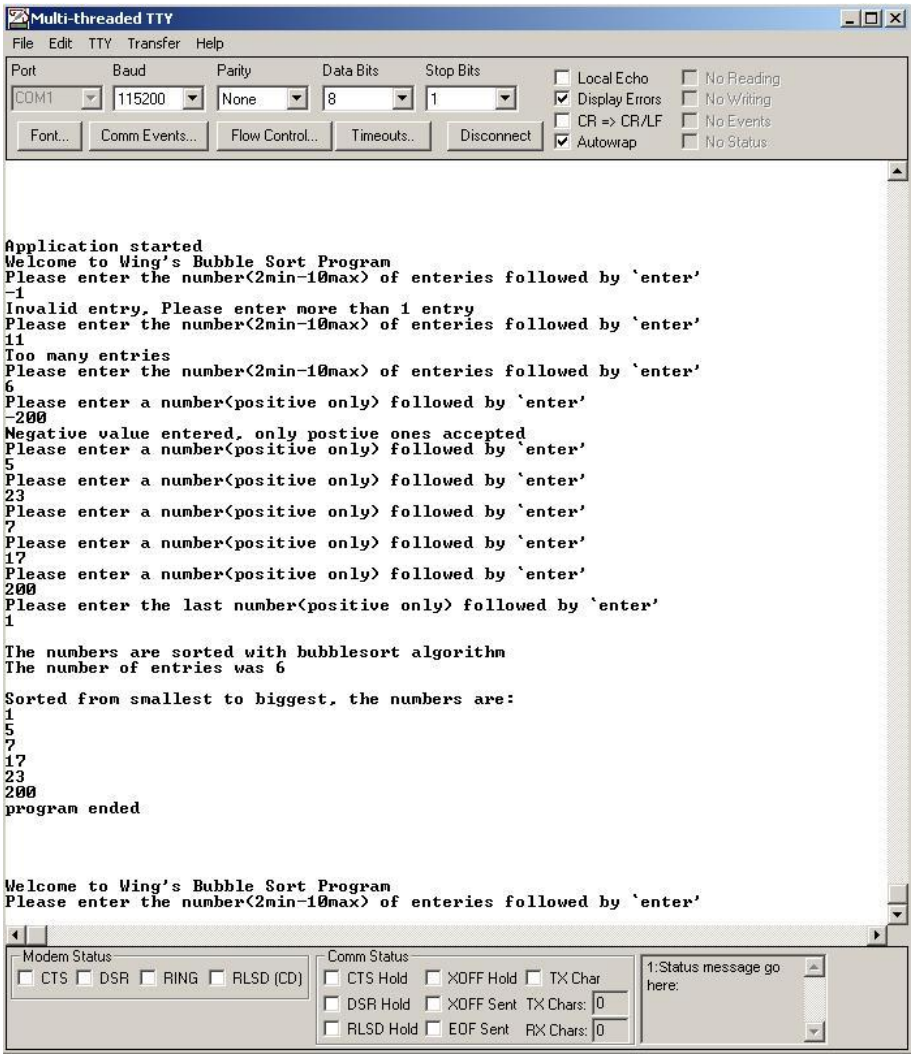
Lab 3: Introduction to Subroutines



Lab Dates

Refer to the schedule on the ECE212 Laboratories page for the latest schedule

Introduction

In this lab the students will be learning how to divide up task by writing subroutines for a bubble sort algorithm program. Good subroutines are expected to be completely contained pieces of code with one entry point and one exit point. In addition, either the caller (the main program) or the callee (the subroutine) must conserve register values to ensure proper execution of code. In ECE212, the callee method will always be used. Your subroutines cannot modify any registers except for registers designated for return values. You must save and restore any registers used in your subroutines.

Objectives:

1. To gain experience using the STACK(Push and Pop).
2. To gain experience in dividing up existing code into subroutines.
3. To gain experience in calling subroutines/functions.
4. To learn the basic parameter passing techniques.

Prelab and Preparation:

- Read the lab prior to coming to your lab section.
- Do the online prelab Quiz
- provide flowchart for each subroutine(3 in total)

Lab Work and Specifications

1. Download the template files
    - main.cpp
    - Lab3.s
    - Lab3a.s
    - Lab3b.s
    - Lab3c.s

Specifications

If you recall from the Lab1, the 'main.cpp' is the standard project template that is used to initialize and call standard functions/subroutines including our 'AssemblyProgram'. Do not modify any of the parameters in this file. 'Lab3.s' has been provided to call your subroutines. Do not modify any of the parameters in this file. Lab3a.s, Lab3b.s and Lab3c.s are provided for you to write your subroutines. Each subroutines has certain passed parameters and certain conditions. Carefull attention should be paid. A more detail description is provided below.

The requirements on each subroutine are indicated below.

1. WelcomePrompt

   In the Lab3a.s template file, you are to code a subroutine called *WelcomePrompt* that prompts the user to enter the number of entries for the bubble sort followed by the numbers for sorting from the keyboard. The number of entries(longword) will be passed out through the stack. The values will be stored starting at memory location 0x2300000. Each entry will occupy one long word(32bits). Certain restrictions have been placed on the number of entries and the size of the numbers. The number of entries entered by the user must be between **2 to 10**(min 2, max 10). The values entered for the bubble sort must be **positive numbers**. Negative numbers are to be rejected. If either condition is not met, your program should reprompt the user to enter a proper valid number. Your program should also flag when the last number is to be entered.

   Entry Condition = space allocated for the number of entries on stack(long word)

   Exit Condtion = number of entries on stack(long word)

   Example - Your program should look something like this:

   - Display a welcome string on startup.
     Welcome to Wing's Bubble Sort Program
   - Display a string prompting the user to enter the number of entries.
     Please enter the number(2min-10max) of entries followed by 'enter'
   - Display a string prompting the user to enter a number.
     Please enter a number(positive only)
   - Display a string prompting the user to enter the last number.
     Please enter the last number(positive only)
   - Display a string indicating invalid entry if an improper value was entered.
     Invalid entry, please enter proper value.

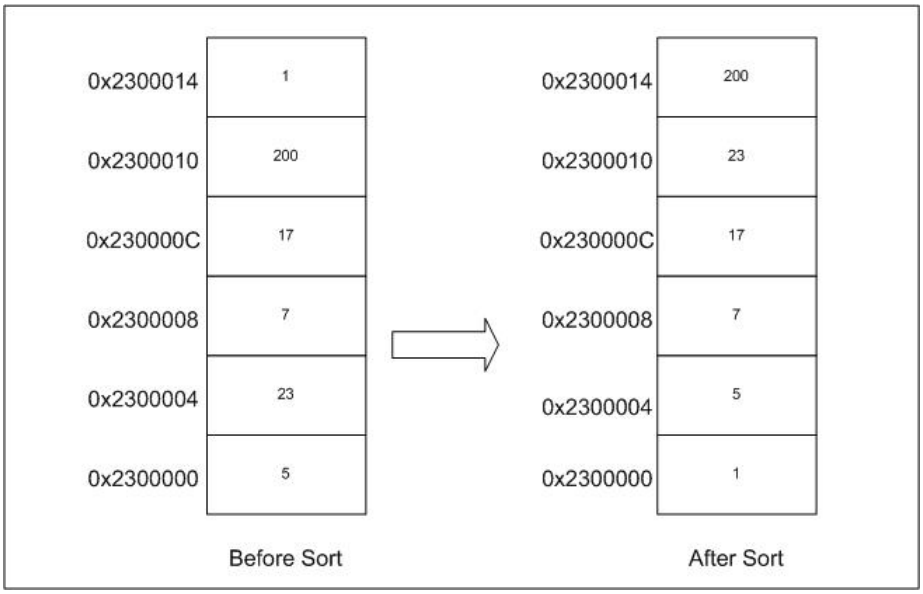
2. Sort

   In the Lab3b.s template file, you are to code a subroutine called *Sort* that peforms the bubble sort alogorith. The value of the number of entries(longword) is passed in through the stack. The memory location 0x2300000 where the entries are stored is provided in the address register A2.

   Entry Condition = Number of entries(long word) on stack.

   Exit Condtion = none

   The figure below illustrates an example of the sort with 6 entries stored starting at memory location 0x2300000.



3. Display

   In the Lab3c.s template file, you are to code a subroutine called *Display* that displays the numbers from lowest to highest. The value of the number of entries along with the memory location 0x2300000 where the entries are stored are passed through the stack.

   Entry Condition = number of entries(long word) and address of where the entries are stored(long word) on the stack

   Exit Condtion = none

Example - Your Display should look something like this:

- The number of entries should be indicated
  The number of entries was XX
- Display a string indicating the numbers are sorted from smallest to largest.
  Sorted from smallest to biggest, the numbers are:
- Display the numbers.
- Display a string ending the program.
  Program ended

Note: For each subroutine, only the information provided can be used. No assumptions can be made.*Use only Data registers D2-D7 and Address registers A2-A5

Provided Subroutines

1. iprintf

   Prints a string to the monitor. No carriage return or linefeed is invoked after calling this function.

   Entry Condition = address of string on the stack

   Exit Condition = address of string on the stack

   Example of pseduocode usage:
   Push StringLabel onto the stack
   Jump to iprint subroutine
   Clean up stack if necessary

2. cr

   A function that generates a carriage return and linefeed

   Entry Condition = None

   Exit Condition = None

   Example of pseduocode usage:
   Jump to cr subroutine
   Clean up stack if necessary

3. value

   A function that prints the number to the monitor with carriage return and linefeed.

   Entry Condition = decimal value on stack(long word)

   Exit Condition = decimal value on stack(long word)

   Example of pseduocode usage:
   Push Decimal value onto the stack
   Jump to value subroutine
   Clean up stack if necessary

4. getstring

   A function that prompts the user to enter a number from the keyboard. Valid entries are all numbers (negative and positive)

   Entry Condition = Nothing

   Exit Condition = Decimal number stored in Register D0 **(Store this value in another data register(D2-D7) for use)**

   Example of pseduocode usage:
   Jump to getstring subroutine
   Clean up stack if necessary

   Note: For simplicity, a check for a proper number entry has been provided. Invalid characters are rejected and cleared from the buffer and the user is prompted to re-enter a proper number.

Data Section

Strings are stored in the .data section in your template files. For example:

Welcome:
.string "This is the welcome string"
This will store the string label Welcome at a certain memory location. At that memory location, the string will be stored in hexadecimal. Each letter will occupy one memory block(1byte).

Note: Always leave a blank line after the last string defined in the .data section. If no blank line is present, an error will be generated when compiling the code.

Questions

1. Is it always necessary to implement either callee or caller preservation of registers when calling a subroutine. Why?

2. Is it always necessary to clean up the stack. Why?

3. If a proper check for the getstring function was not provided and you have access to the buffer, how would you check to see if a valid # was entered? A detailed description is sufficient. You do not need to implement this in your code.

Marking Scheme

Lab 3 is worth 25 % of the final lab mark.
Please view the Marking Sheet for this lab to ensure that you have completed all of the requirements of the lab. The Marking Sheet also provides a limited test suite in the demo section for you to think about. Make use of it!
The report is to follow the Report Writing Guidelines

Demo and Report
The report and demo due dates are given on the ECE212 Laboratories page. Note that you have one week from the dating of your prelab to complete the demo.

The reports must be handed in by 4 P.M. Do not be late.

Late submissions
Late submission penalties for the demo and report portions of the labs are given on the ECE212 Laboratories page