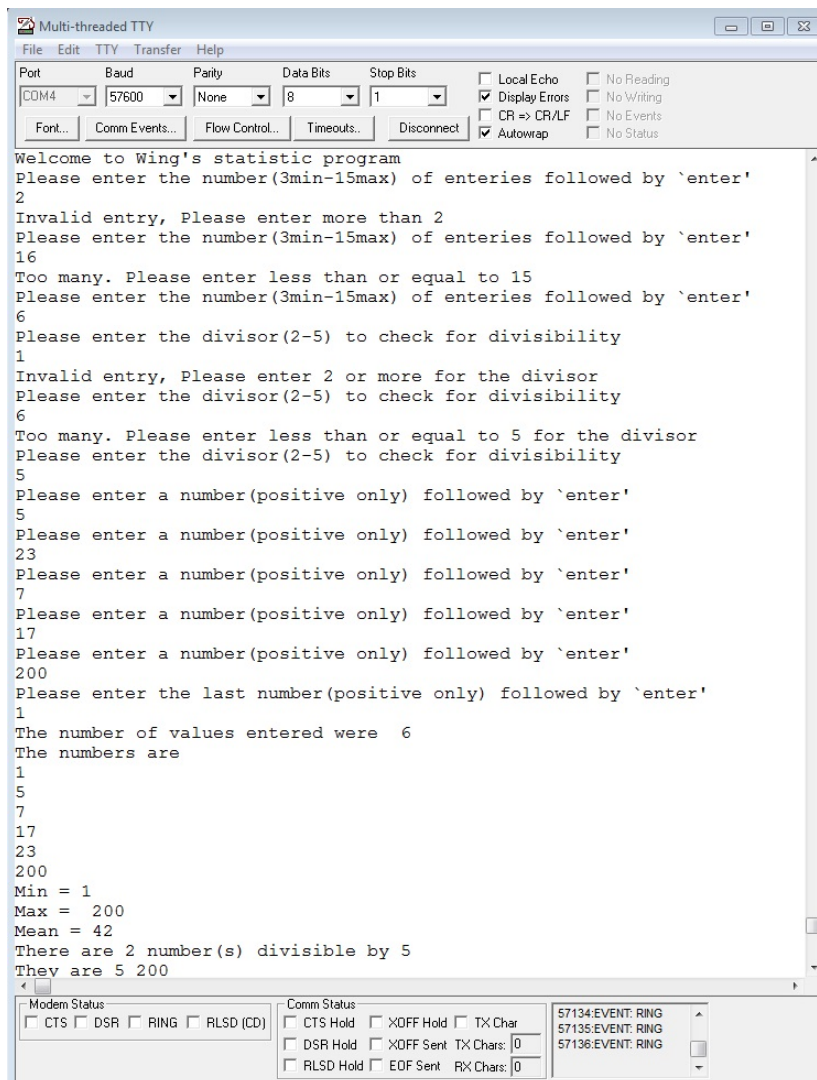


## Lab 3: Introduction to Subroutines



The screenshot shows a 'Multi-threaded TTY' window with a menu bar (File, Edit, TTY, Transfer, Help) and a configuration section. The configuration includes dropdowns for Port (COM4), Baud (57600), Parity (None), Data Bits (8), and Stop Bits (1). There are checkboxes for Local Echo, Display Errors, CR => CR/LF, Autowrap, No Reading, No Writing, No Events, and No Status. Below the configuration is a text area containing the following text:

```
Welcome to Wing's statistic program
Please enter the number(3min-15max) of enteries followed by `enter'
2
Invalid entry, Please enter more than 2
Please enter the number(3min-15max) of enteries followed by `enter'
16
Too many. Please enter less than or equal to 15
Please enter the number(3min-15max) of enteries followed by `enter'
6
Please enter the divisor(2-5) to check for divisibility
1
Invalid entry, Please enter 2 or more for the divisor
Please enter the divisor(2-5) to check for divisibility
6
Too many. Please enter less than or equal to 5 for the divisor
Please enter the divisor(2-5) to check for divisibility
5
Please enter a number(positive only) followed by `enter'
5
Please enter a number(positive only) followed by `enter'
23
Please enter a number(positive only) followed by `enter'
7
Please enter a number(positive only) followed by `enter'
17
Please enter a number(positive only) followed by `enter'
200
Please enter the last number(positive only) followed by `enter'
1
The number of values entered were 6
The numbers are
1
5
7
17
23
200
Min = 1
Max = 200
Mean = 42
There are 2 number(s) divisible by 5
They are 5 200
```

At the bottom of the window, there are sections for 'Modem Status' and 'Comm Status'. The 'Modem Status' section has checkboxes for CTS, DSR, RING, and RLSD (CD). The 'Comm Status' section has checkboxes for CTS Hold, XOFF Hold, TX Char, DSR Hold, XOFF Sent, TX Chars: 0, RLSD Hold, EOF Sent, and RX Chars: 0. On the right side of the 'Comm Status' section, there is a list of events: 57134:EVENT: RING, 57135:EVENT: RING, and 57136:EVENT: RING.

## 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 statistics 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 preserve 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.

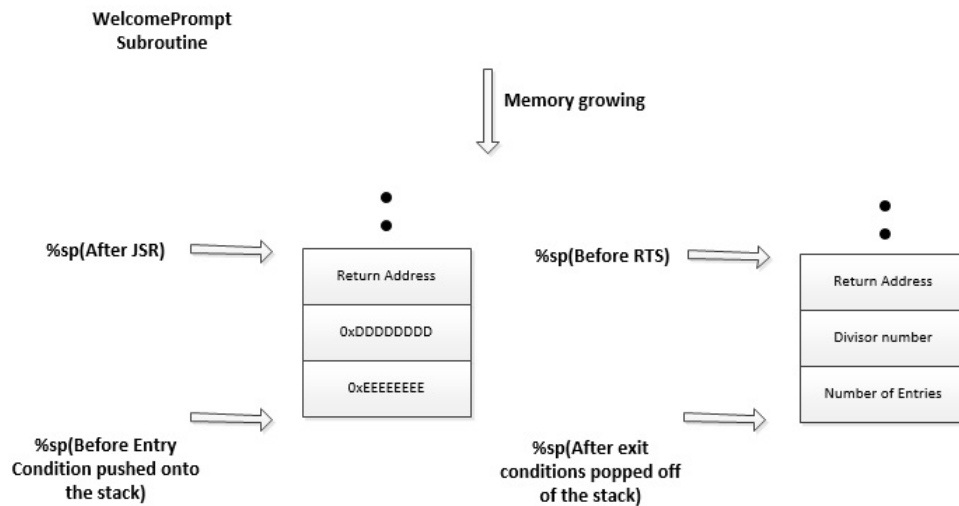
The requirements on each subroutine are indicated below.

### 1. WelcomePrompt

In the Lab3a.s template file, you are to code a subroutine called *WelcomePrompt* that prompts the user to enter numbers from the keyboard. The first number is the number of entries, followed by the divisor number and finally the values that will be analyzed statistically. The number of entries(longword) and divisor(longword) will be passed out through the stack. The values that are analyzed will be stored starting at memory location 0x43000000. Each entry will occupy one long word(32bits). Certain restrictions have been placed on the number of entries, divisor and the size of the numbers. The number of entries entered by the user must be between **3 to 15**(min 3, max 15). The divisor must be between **2 to 5**(min 2, max 5). The values entered that will be analyzed must be **positive numbers**. Negative numbers are to be rejected. If any of the conditions are 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.

**Stack Entry Condition = 1.space allocated for the number of entries on stack(long word) 2.space allocated for the divisor number on stack(long word)**

**Stack Exit Condtion = 1.number of entries on stack(long word) 2.divisor number on stack(long word)**



Example - Your program should look something like this:

- Display a welcome string on startup.  
*Welcome to Wing's Stats Program*
- Display a string prompting the user to enter the number of entries.  
*Please enter the number(3min-15max) of entries followed by 'enter'*
- Display a string prompting the user to enter the divisor number.  
*Please enter the divisor(2min-5max) 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.*

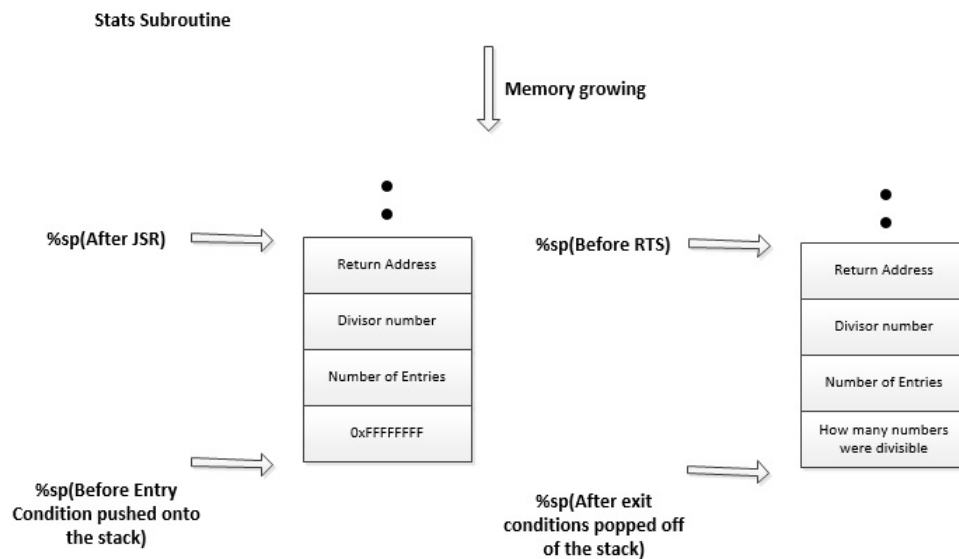
## 2. Stats

In the Lab3b.s template file, you are to code a subroutine called *Stats* that finds the min, max, mean, how many numbers were divisible by the divisor and what are they from the numbers entered. The results are to be stored starting at 0x43100000 with the exception of

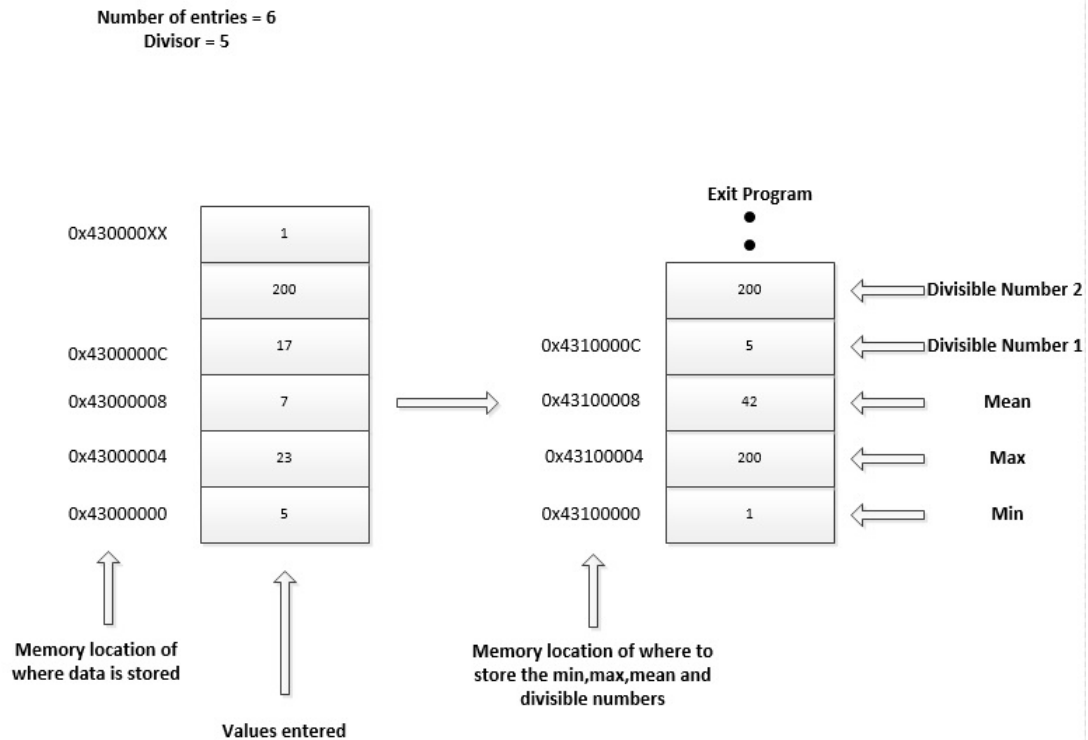
how many numbers were divisible by the divisor. That number is going to be passed out through the stack. Please refer to the figure below for the order in which they are stored. The number of entries(longword) and divisor are passed in through the stack. The memory location 0x43000000 and 0x43100000 are provided in the address register A2 and A3. Please use them.

**Stack Entry Condition** = 1.space allocated for how many numbers are divisible by the divisor(long word) 2.Number of entries(long word)on stack 3.Divisor number(long word) on stack.

**Stack Exit Condition** = 1.How many numbers are divisible by the divisor(long word) on stack 2.number of entries on stack(long word) 3.divisor number on stack(long word)



The figure below illustrates an example of the stats program with 6 entries stored starting at memory location 0x43000000.



**Min = Find the smallest number**

**Max = Find the largest number**

**Mean = Find the average. Round down if there is a fraction**

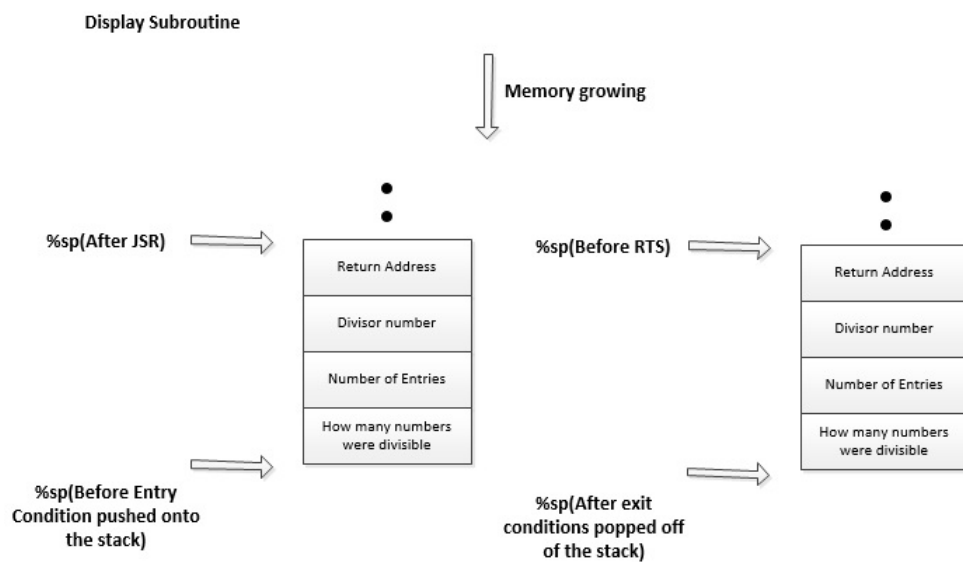
**Divisible = Find the numbers that are divisible by the divisor**

### 3. Display

In the Lab3c.s template file, you are to code a subroutine called *Display* that displays the min, max, mean, how many numbers were divisible by the divisor and what are they on MTTTY. In addition, please redisplay all the numbers that were entered. How many numbers were divisible by the divisor, the number of entries, and the divisor are passed in through the stack. The memory location 0x43000000 and 0x43100000 are provided in the address register A2 and A3. Please use them.

**Entry Condition** = 1. how many numbers were divisible by the divisor 2. number of entries(long word) 3. Divisor(longword)

**Exit Condition** = 1. how many numbers were divisible by the divisor 2. number of entries(long word) 3. Divisor(longword)



Example - Your Display should look something like this:

- The number of entries should be indicated  
*The number of entries was XX*
- Display the numbers.
- Display a separate string indicating the min,max,mean and median.  
*Min number =:XX*  
*Max number = XX*  
*Mean number = XX*  
*There are XX number(s) divisible by Y*
- Display a string ending the program.  
*Program ended*

**Note:** For each subroutine, only the information provided can be used. No assumptions can be made.

## 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

**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