CSE 379

University at Buffalo

Keming Kuang (kemingku)

Eric Li (eli9)

March 6, 2018

Contents

1 Division of work

This lab has mainly two components the read component and the write component. We decided to split the work based on these two components. Keming oversaw the read components namely read_string and read_character. Eric oversaw the write components namely output_string and output_character. As for the other subroutines, we worked on it together.

2 Description

2.1 Purpose of the program:

This program can take in 1-15 numbers between -9999 - +9999 and return the remainder and mean of the sum of the numbers. This is done by reading each number as a string which the program then converts back into their integer value, calculate the sum then through division find the mean and remainder of the numbers. The program then outputs the results in the form of a string.

2.2 Instructions:

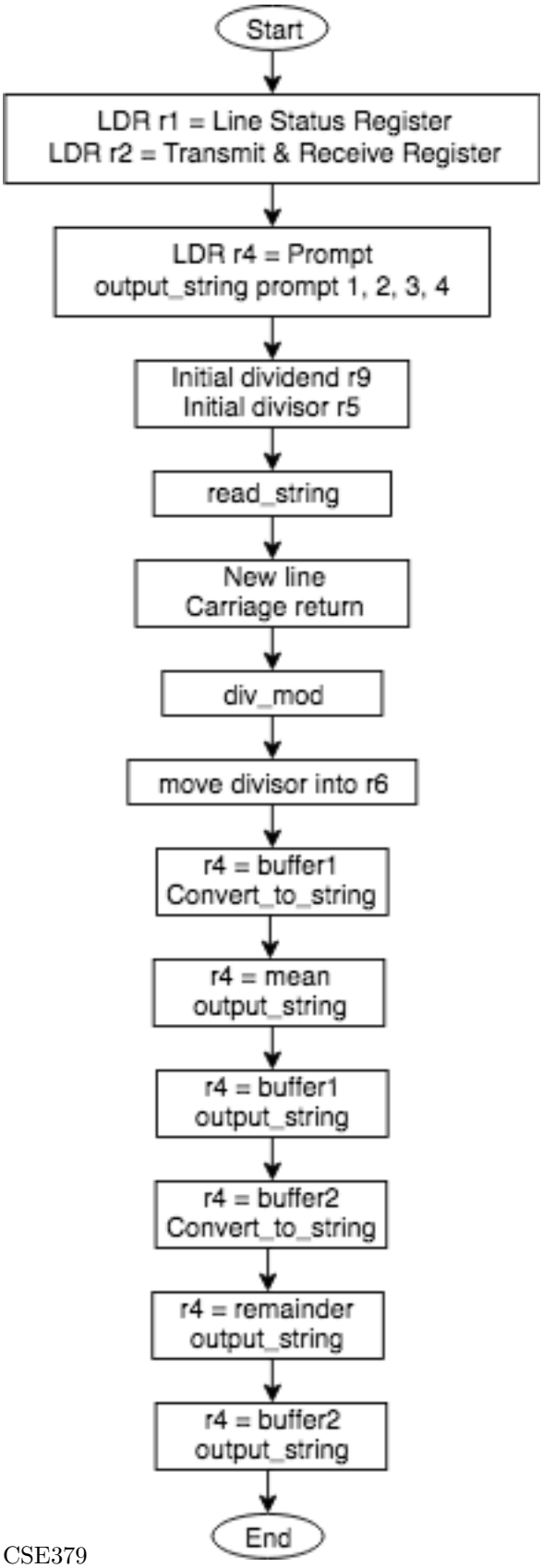
To use the program, enter 1 to 15 numbers from -9999 to +9999 including the positive and negative sign. After entering your number press the enter key before entering your next number. To signal to the program that all the numbers have been entered, enter the key q . After the key q has been entered the program will then proceed to calculate the mean and remainder of the sum of the numbers.

2.3 Debugging steps

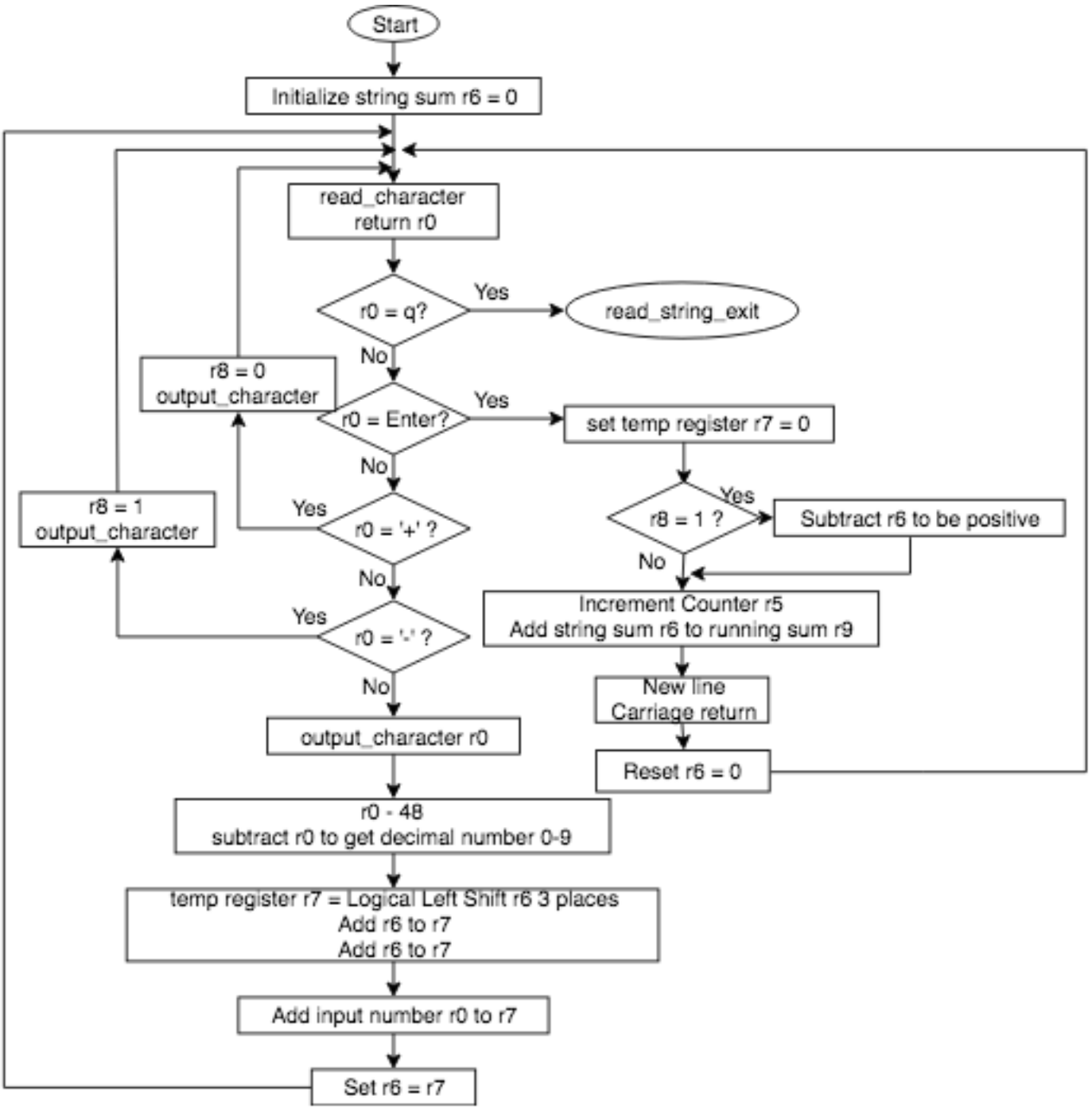
There were multiple debugging strategies we used to test and debug our code. One of the debugging strategies we constantly used was register value tracking. We passed into our subroutines parameters which results in an expected register value and through the simulation tool in keil, we looked for this expected value to test whether the subroutine functioned. Another strategy we frequently employed was to call subroutines that functioned after the subroutine we were debugging to test whether the subroutine in question exited properly. This debugging strategy was mainly used to give us a idea where bugs may have existed which we would then look deeper into via the simulation tool.

3 Flow Chart

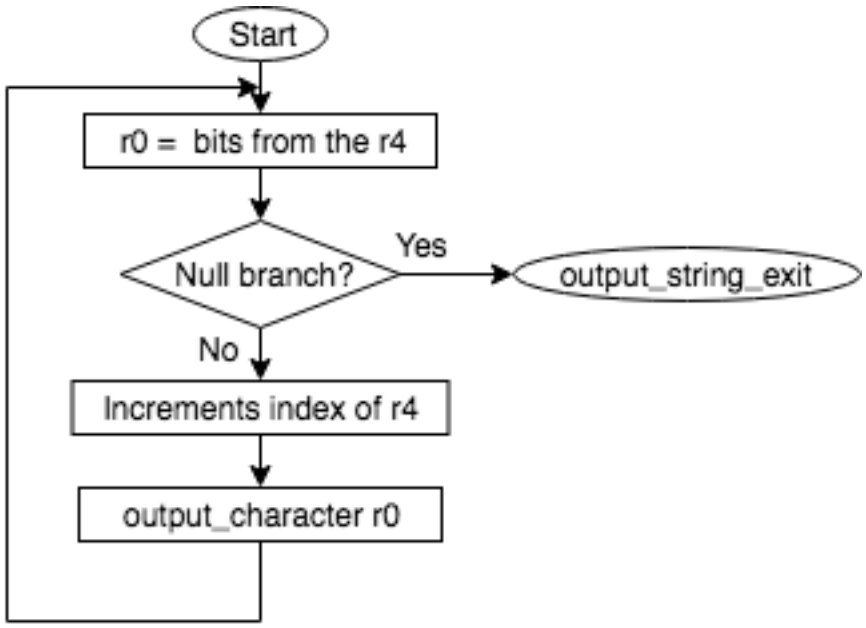
3.1 Main program



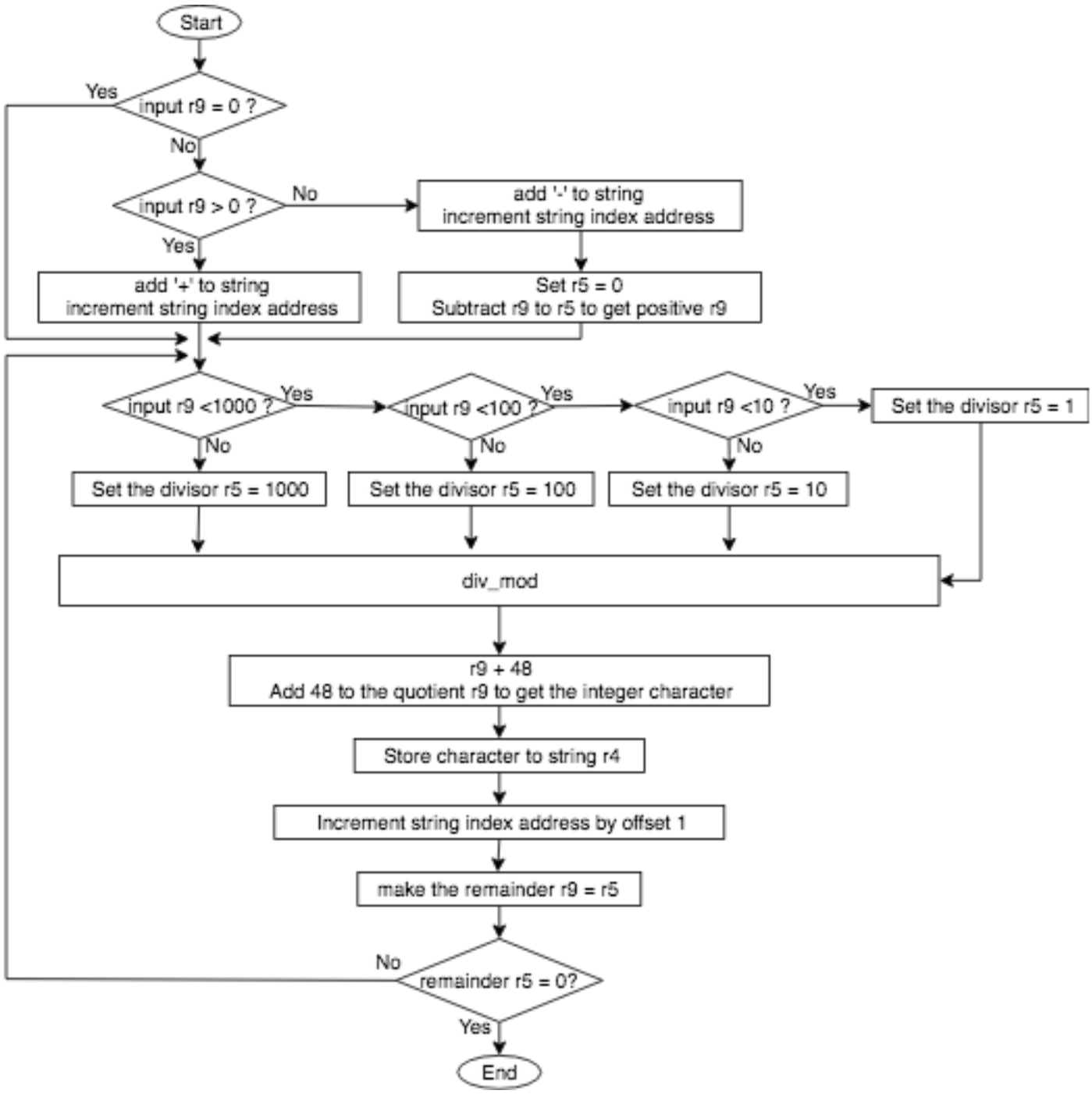
3.2 read_string



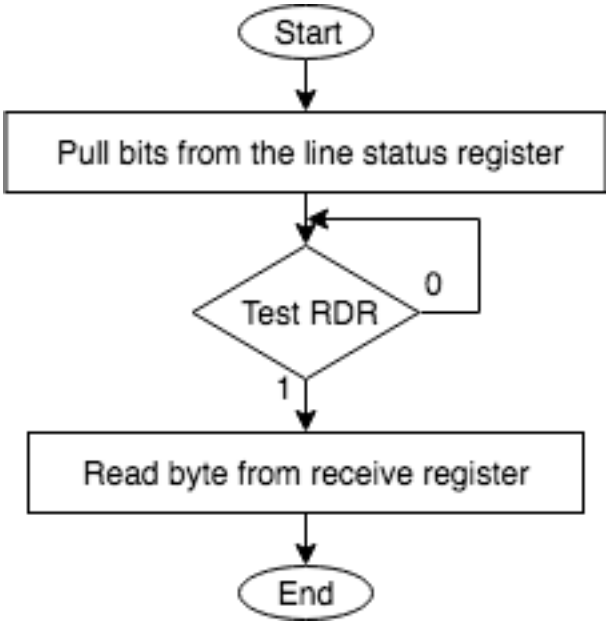
3.3 output_string



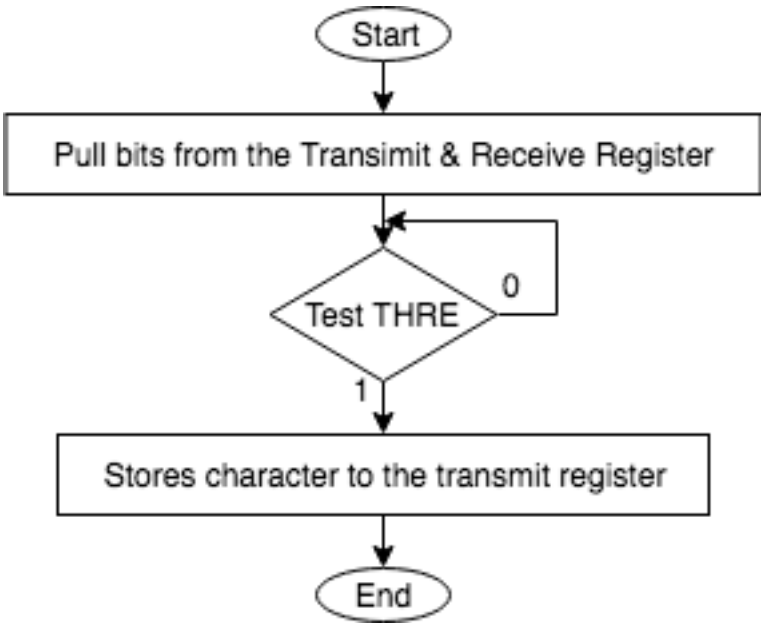
3.4 convert_to_string



3.5 read_character



3.6 output_character



4 Summary

4.1 uart init:

Initialize the user UART or use. uart_init will replace the function of the C function serial_init.

4.2 Main program:

The program is split into two sections, the main program, and the subroutines. The main programs first print out the prompt using output_string then, calls read_string which will read in the numbers and update running sum for the division process. Afterwards, the main program calls div_mod which will find the mean and remainder. Lastly, the main program calls convert_to_string to convert both integer values of the remainder and mean to strings which will then be printed with output_string.

4.3 Read string:

Read string functions on two main loops. The first loop ends when the character q is read. The second loops terminate when the character for carriage return is read. If neither is read, read strings tests to see if the character read is either a plus or minus sign r8 will then be updated to represent a positive or negative number based on the sign, 1 for positive 0 for negative. After each character is read in the character will be echoed back via output_character. The number read will be converted to its decimal value by multiplying the register r7 by 10 then adding the read character to r7. This multiplies the temporary sum by 10 before adding the new number which mimics powers of 10. This process and the process of reading character will repeat until the character for enter is read. After the character for enter is read the universal counter which will also be the divisor for the mean will be incremented by 1. This will then repeat until the character q is read.

4.4 Output string:

output_string is simply a loop that keeps on calling output_character until the null character is read which indicates the end of the string. output_string functions by loading each character of a string comparing it to null then calling output_character if the character is not null. If the character read is null, then it terminates.

4.5 Convert to string:

Convert_to_string functions by doing the reverse of what was done to convert a character to an integer. Convert_to_string in summary calls div_mod with divisors being powers of 10. The subroutine first, compares the number to determine whether it is positive or negative. It will first store the sign to the string buffer then, converts the number to positive if it is negative before proceeding with the conversion. To determine which power of 10 is used, convert_to_string compares the dividend to 1000, 100, and 10. If the dividend is greater than 1000 divisor becomes 1000 and so on. The routine will then call div_mod the quotient will be the character and the remainder will be the new dividend. This loop ends when the remainder is 0.

4.6 Read character:

Read character reads in a character from the receiving register when the RDR is 1. Read_character will infinitely loop unless the RDR is 1.

4.7 Output character:

Transmits the character to the transmit register when the THRE is 1. Output_character will infinitely loop until the THRE becomes 1.