Lab #3

CSE 379

Connor Lehr, cjlehr

Saahil Patel, saahilpa

Lab section R1

3/2/23

Table of Contents

Division of Work

Connor Lehr:

Created the following subroutines and flowcharts:

- read_charcter
- read_string
- int2String
- int_digit
- uart_init

Also set up documentation PDF and documented my contributions.

Saahil Patel:

Created the following subroutines and flow charts:

- output_character
- output_string
- string2int
- uart_init

Documented my contributions.

Program Overview

Lab 3 takes in two integers including 0 up to and including 999999 (no comma separation). It will then output the average of those 2 integers.

The user will be prompted to input the first integer. Once they press enter that integer will be recorded. The user will then be prompted for the second integer, which will again be recorded once they hit enter. After they've entered both integers the average of the two will be outputted.

steps:

0. Start running program
1. Directions displayed in putty:

   Type number and press enter (0-999999):

2. Enter first integer:

   Type number and press enter (0-999999): 25

3. Enter second integer:

   Type number and press enter (0-999999): 820

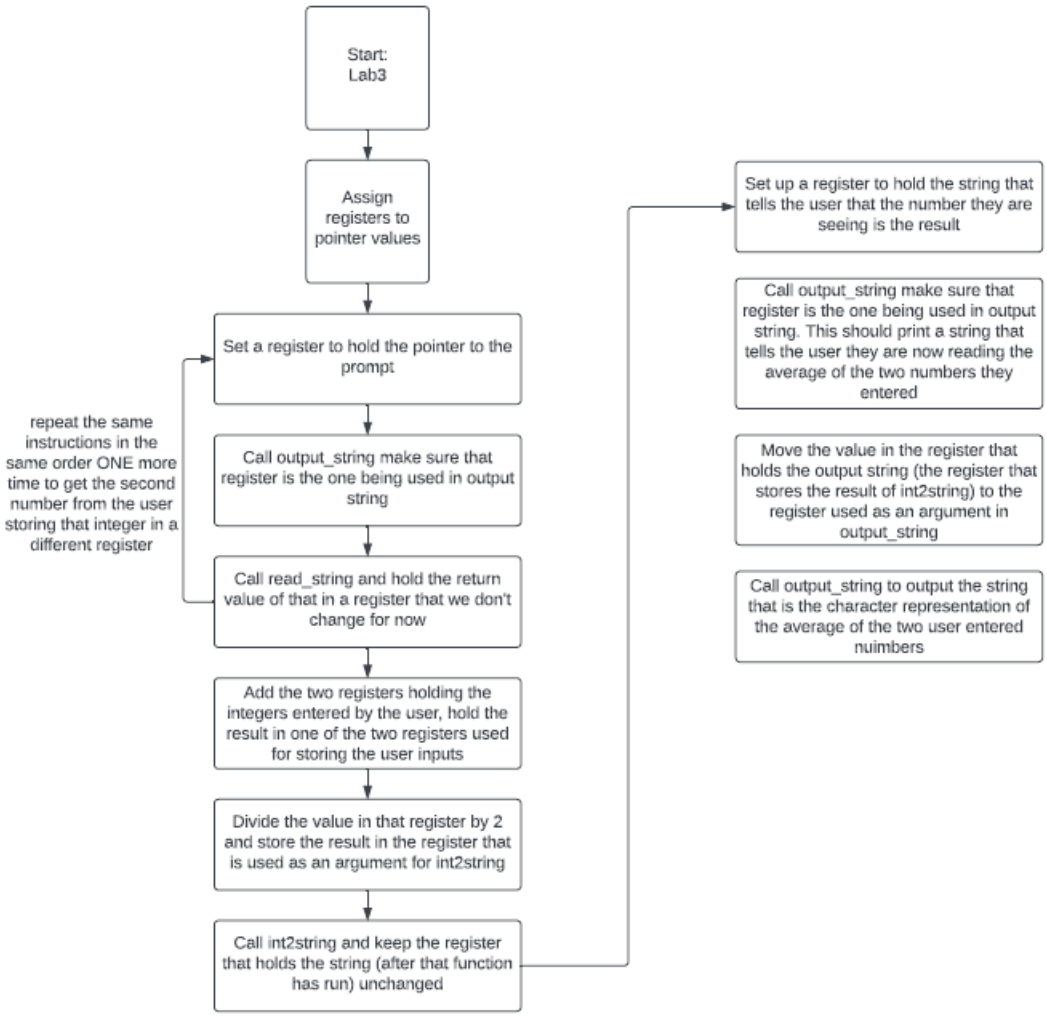
4. Average is returned:

   Average: 435

Program Summary

Program is ran is started with some strings stored in the data portion of the code. This includes 1 prompt to ask for numbers, 1 prompt to print out before the result (both of which are displayed in the step 1 and 2, and 4), and 2 place holder strings to store the integers typed into putty. It then sets up the UART to be able to send and receive characters with putty. After that it moves the prompts/strings into register 0 before necessary subroutine. R0 <- prompt then output_string, r0 stores place holder 1 then read_string, string2int, repeat steps to get next integer, add the 2 integers shift logically left once, int2string, r0 <- result prompt then output_string, r0<-actual result after int2string, output_string.

High Level Flowchart



Subroutine Description

Output_charcter:

Description: This subroutine will be used when outputting strings to the display. Takes a char makes sure its not equal to a null character and outputs it to the display.

Arguments: register holding a char

Return value: The char in the register is displayed on Putty

Read_string:

Description: This subroutine reads in the string entered into Putty. It then stores the NULL terminated string at r0

Arguments: UART.

Return value: The string inputted to the Putty, return register is r0

Output_string:

Description: This subroutine reads the string given as an argument character by character in a loop. If the character being read is not null then we assign a register to hold that char that register will be used in the output_character subroutine that will output the character. If the character is equal to null then we exit the function.

Argument: register containing the string to be displayed

Return value: the string is displayed to Putty

Read_character:

Description: This subroutine reads a character received by UART.

Arguments: UART.

Return value: character from UART, return register is r0

Int2string:

Description: This subroutine takes in an integer and converts it into a NULL terminated string. It then stores the string at r1.

Arguments: Integer to convert, input register is r0. Memory address of string, input register is r1.

Return value: String of integer, return register is r1

String2Int:

Description: This subroutine reads the string given as an argument character by character in a loop. If the character being read is not null then we do ascii math subtracting the ascii 0 from the character that is being read. We store that result in a register. We then enter a loop that does the same thing, however on every iteration after we subtract to get the integer value, we left shift the register holding the first digit and then we mask the current digit and use ORR to do a bitwise or between the left shifted register and the register holding the difference that we masked. This will enter the next digit into the register that is holding the value to be output.

If the character is equal to null on any compare, then we exit the function.

Argument: register containing the string to converted to int

Return value: register storing the integer representation of the string given

Lab3:

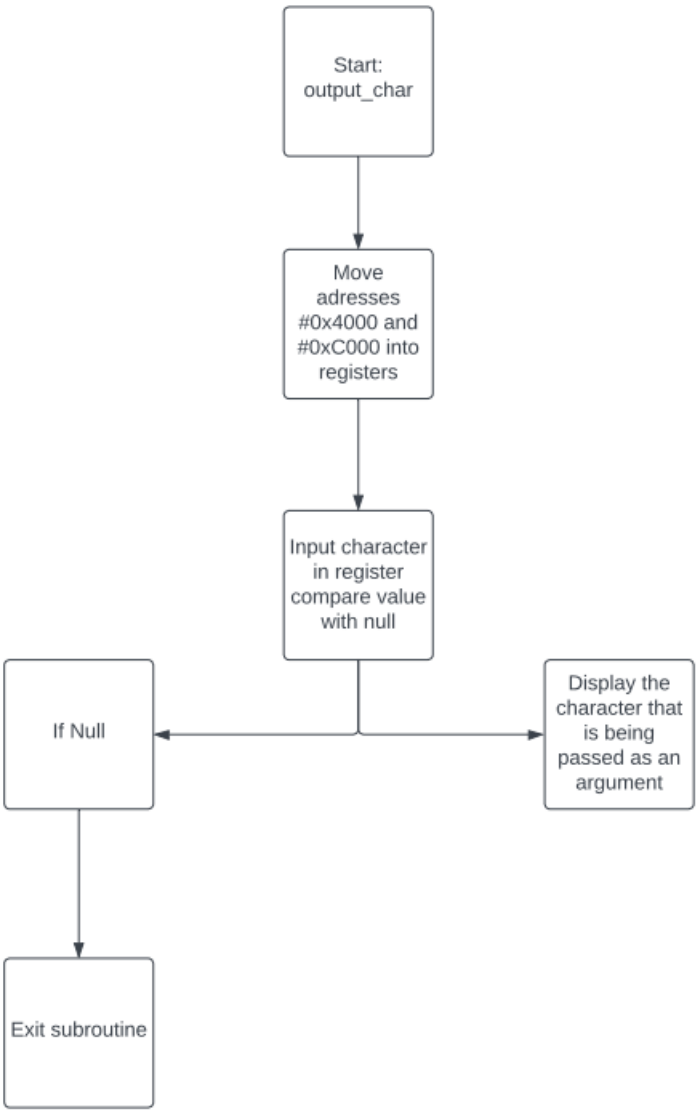
Description: This subroutine acts as the main function of the program here call on other subroutines. We also set up registers properly so that the argument register used in the function we are calling is assigned the correct value. We also assign 2 registers that hold the value of the numbers entered by the user as ints (string2int) . We compute the average of the two numbers in this subroutine.

Arguments: Pointers to prompt and result strings.

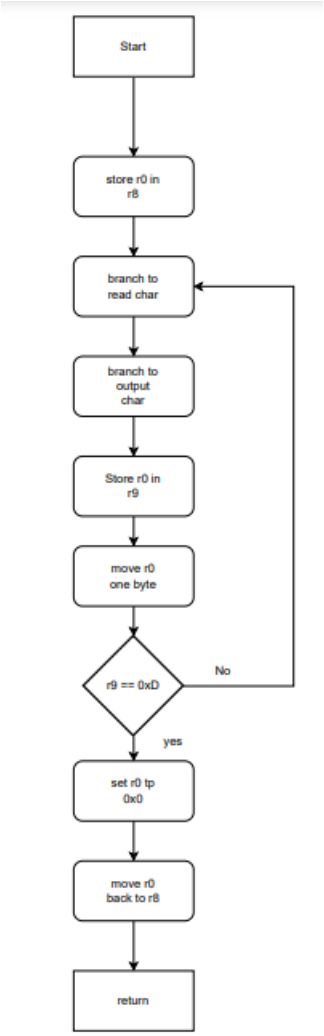
Output value: End output goal is the average of the two numbers entered by the user.

Subroutine Flowchart

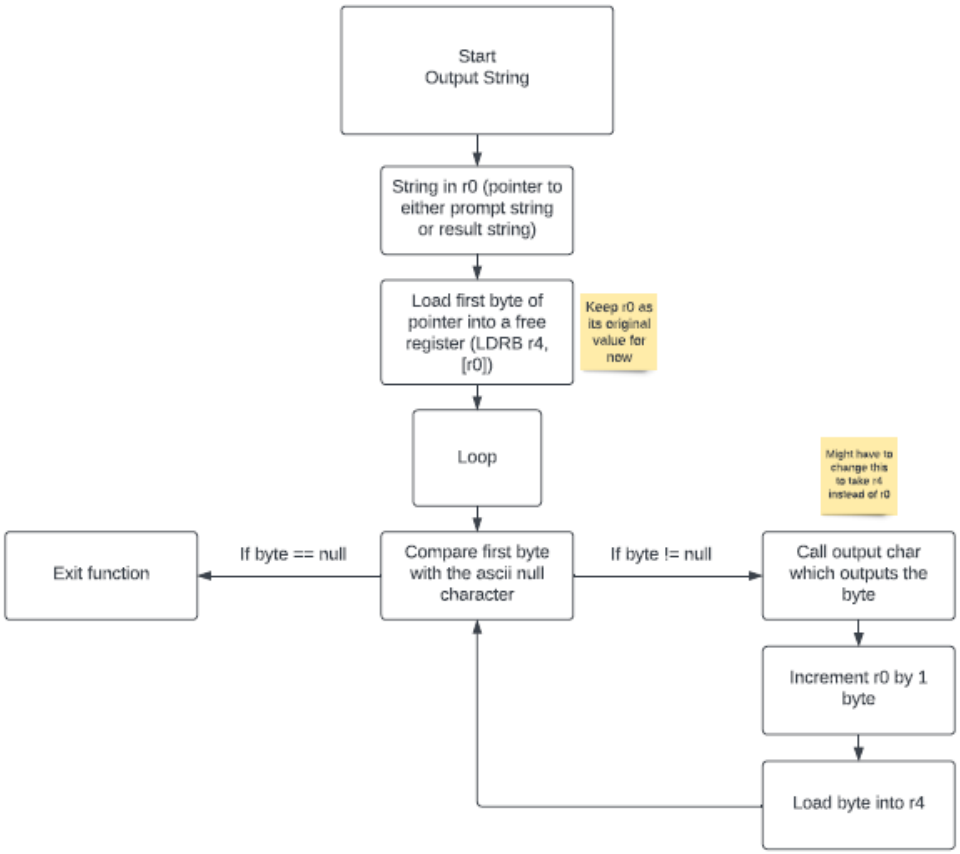
Output_charcter



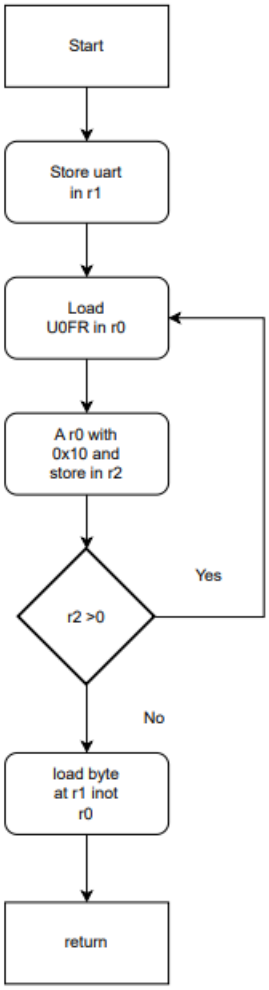
Read_string


Start
store r0 in r8
branch to read char
branch to output char
Store r0 in r9
move r0 one byte
r9 == 0xD
No
yes
set r0 tp 0x0
move r0 back to r8
return

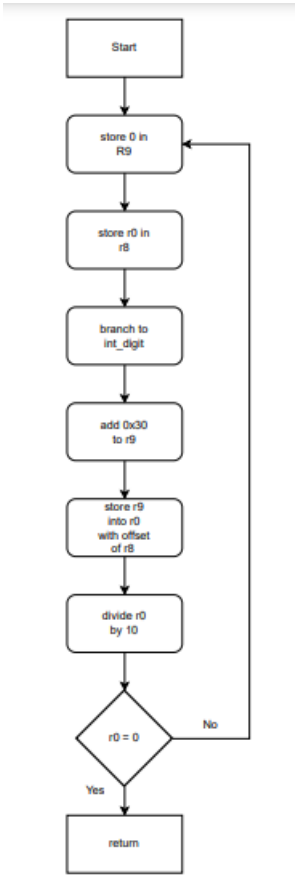
Output_string



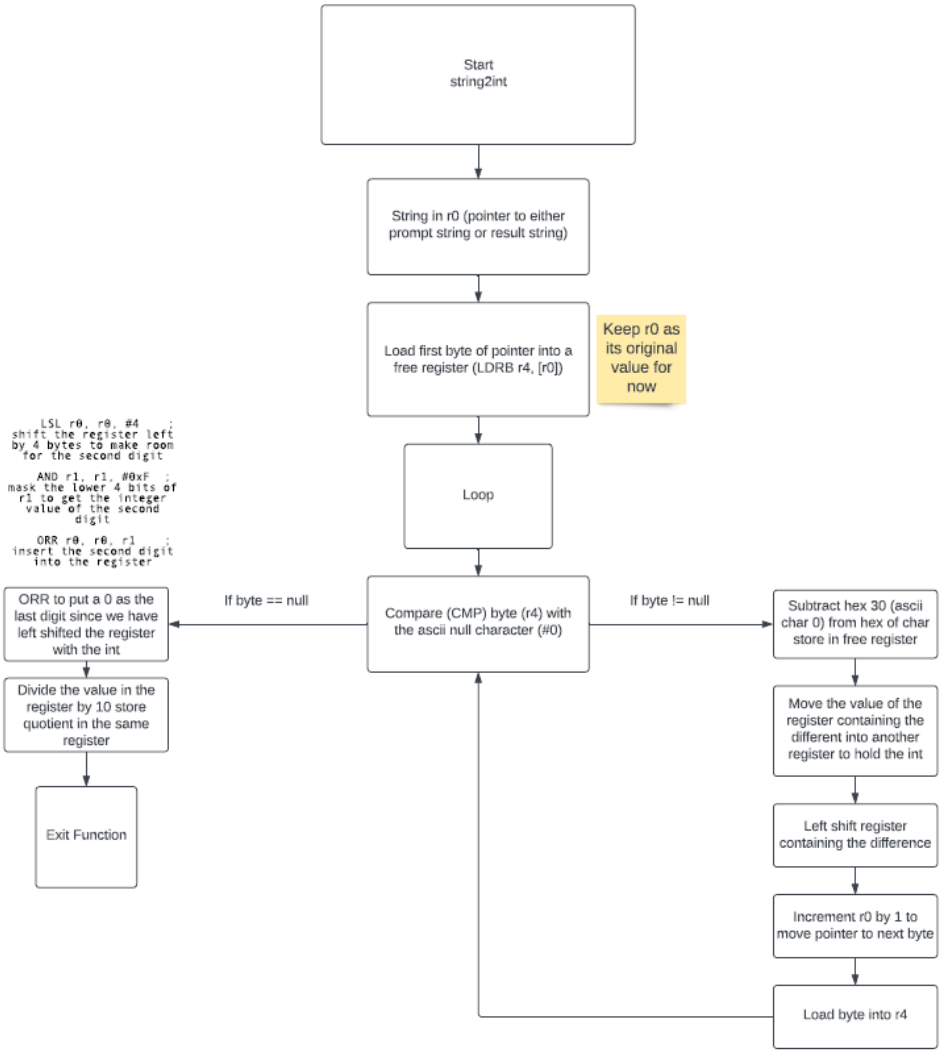
Read_character



Int2string


Start
store 0 in R9
store r0 in r8
branch to int_digit
add 0x30 to r9
store r9 into r0 with offset of r8
divide r0 by 10
r0 = 0
No
Yes
return

String2Int


Start
string2int
String in r0 (pointer to either prompt string or result string)
Load first byte of pointer into a free register (LDRB r4, [r0])
Keep r0 as its original value for now
Loop
Compare (CMP) byte (r4) with the ascii null character (#0)
If byte == null
If byte != null
ORR to put a 0 as the last digit since we have left shifted the register with the int
Divide the value in the register by 10 store quotient in the same register
Exit Function
Subtract hex 30 (ascii char 0) from hex of char store in free register
Move the value of the register containing the different into another register to hold the int
Left shift register containing the difference
Increment r0 by 1 to move pointer to next byte
Load byte into r4
LSL r0, r0, #4 ; shift the register left by 4 bytes to make room for the second digit
AND r1, r1, #0xF ; mask the lower 4 bits of r1 to get the integer value of the second digit
ORR r0, r0, r1 ; insert the second digit into the register

Flowchart Software

Draw.io on diagrams.net