Lab 3

CSE 379 LLB - Introduction to Microprocessors

Partner Name:Yicheng Luo, Xudong Liu

Partner Username: yluo25, xliu243

Lab Section: R4

Date: 02/20/2024

Division of Work

Yicheng Luo (yluo25):

Writing read_character, int2string, read_string, writeing document

Xiudong (xliu243):

Writing output_character, string2int, output_string, editing flowchart

Program Overview

Program Overview:

1. Start running the program
2. Connecting to Putty
3. Follow the instruction to input your dividend, press enter to finish your input.
4. Follow the instruction to input your divisor, press enter to finish your input.
5. At this point you should be able to see your quotient and remainder in putty.
6. According to the prompts of putty, press space and continue to enter a set of divisors and dividends, or press enter to exit the program.

Program Summary:

This project required the implementation of ARM assembly language subroutines to enable communication between a user interface (such as PuTTY) and an ARM processor. The goal is to create a program that allows the user to enter two signed integers (divisor and divisor) via PuTTY, perform division and modulus calculations on these numbers, and display the quotient and remainder to the user.

Subroutines to be developed include

uart_init: initializes the UART (Universal Asynchronous Receiver/Transmitter) for communication.
output_character: output character: transfers a character from the UART to PuTTY.
output_string: Transmit a NULL-terminated ASCII string for display in PuTTY.
read_character: Read the character received by UART: Read the character received by UART from PuTTY.
read_string: Reads the string entered in PuTTY and stores it in memory as a NULL-terminated ASCII string.
int2string: Converts an integer to a NULL-terminated ASCII string in memory.
string2int: Convert a NULL-terminated ASCII string to an integer.
div_and_mod: performs division and modulus calculations.

lab3: Top-level routine that handles user data input, performs calculations, and displays results. The user will enter two signed integers and press Enter to end the entry. The program will then calculate the quotient and remainder of the divisor after dividing the divisor by the divisor. The quotient will be displayed as a signed number and the remainder will always be positive. Input and output operations will be performed via PuTTY with appropriate prompts and instructions to guide the user through the process. The program will continue to run until the user decides to exit.

High Level Flowchart:

Subroutine Descriptions

Describe Each Subroutine

uart_init

output_character

read the single character from the previous register and output it

arguments char value in r0

return values: char value output to putty

output_string

read the result of int2string and output it

arguments: address in r0

return values: string value at address output to putty

read_character

read a single character into register

arguments: char value input from keyboard

return values: value stored in r0

read_string

read a string into a consistent memory block

arguments: enter end string input from keyboard

return value: Null(0x0) terminated string stored in address 0x2000 0000

int2string

convert the int in r0 into string

arguments: int value in r0

return values: address point to converged string in RAM

string2int

convert the string from the previous consistent memory block into signed int value

arguments: result of read_string, Address 0x2000 0000 which store the Null(0x0) terminated string

return values: converged int value in r0

div_and_mod:

divide two signed int values and get the remainder and quotient.

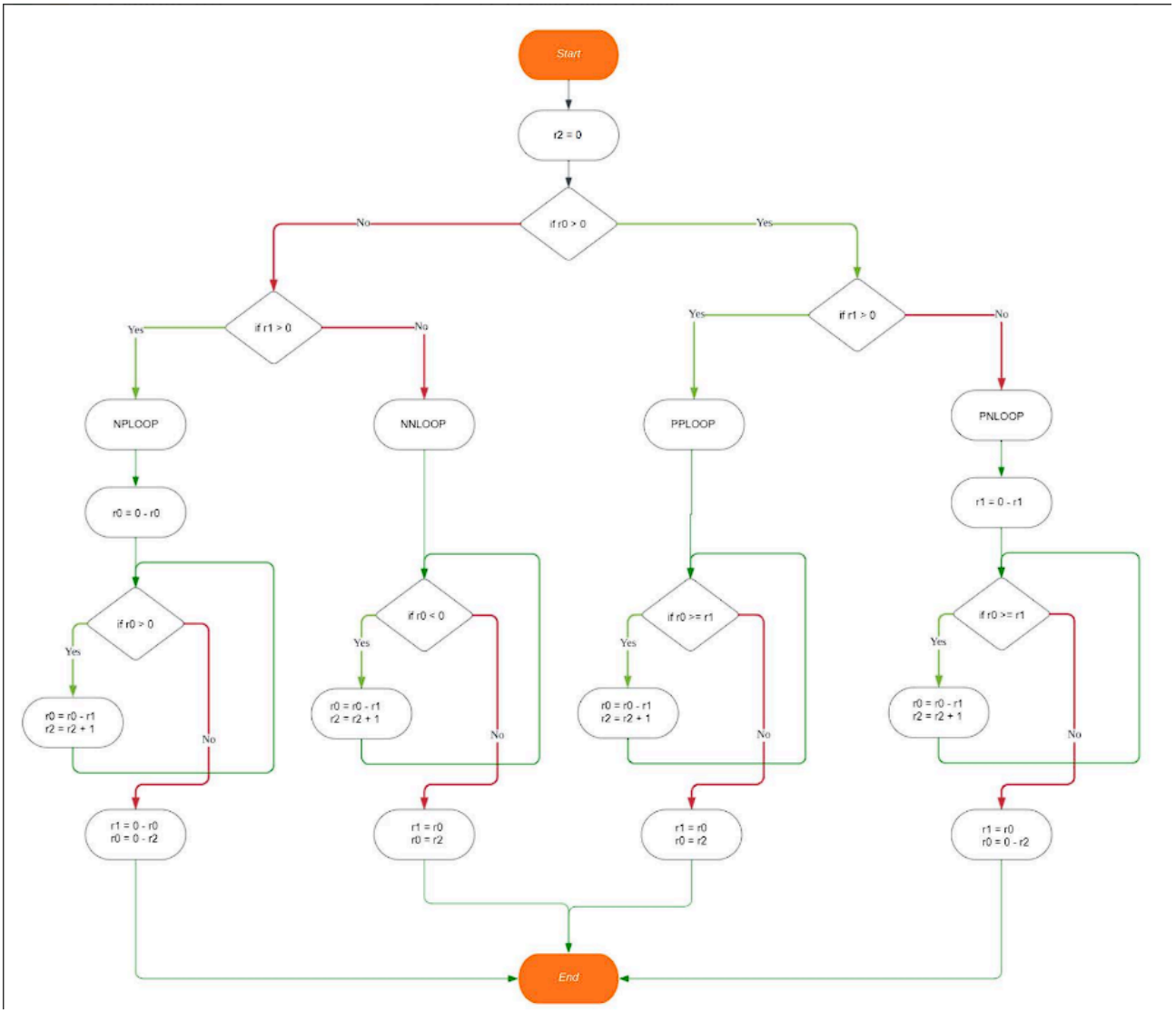
arguments:

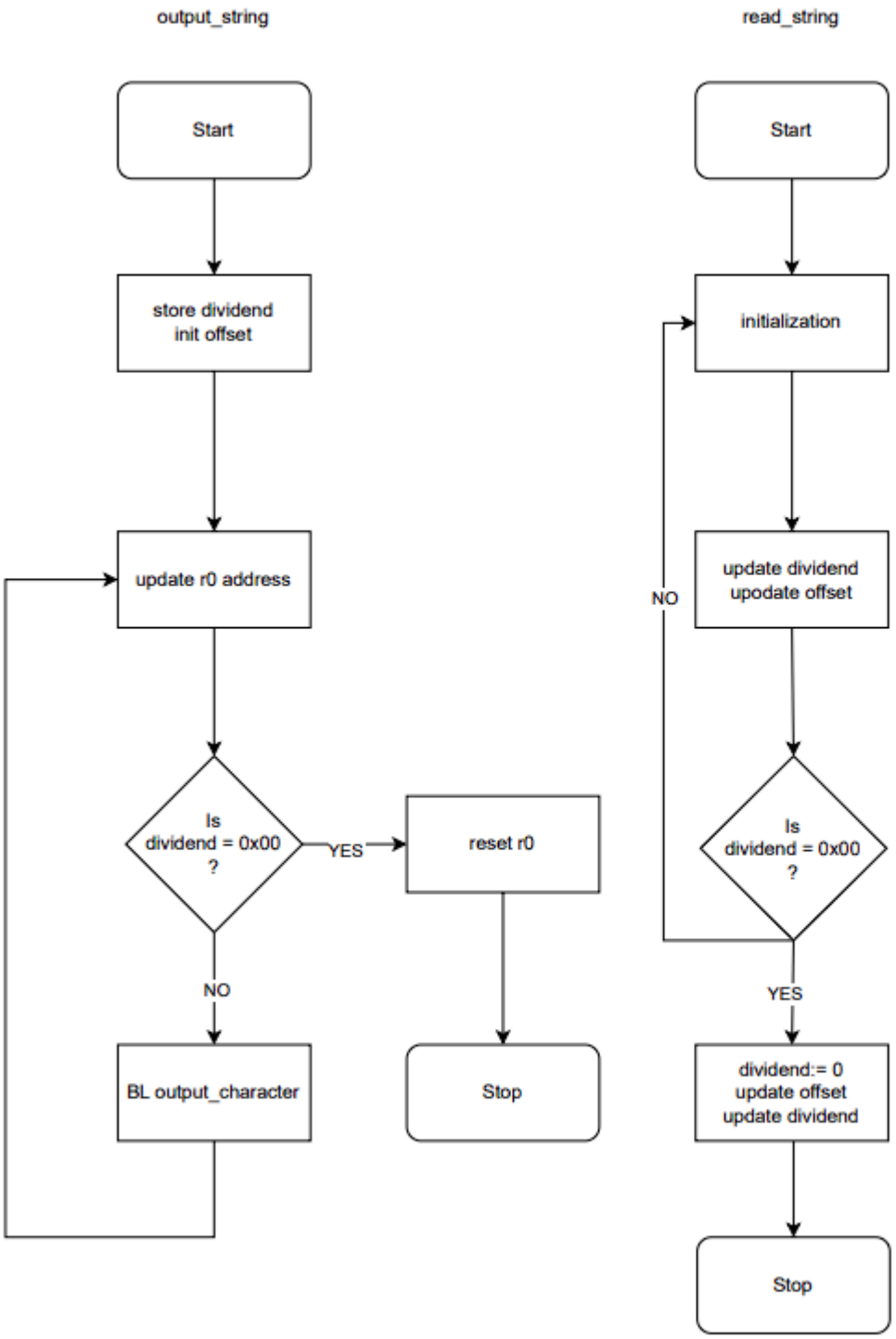
return values:

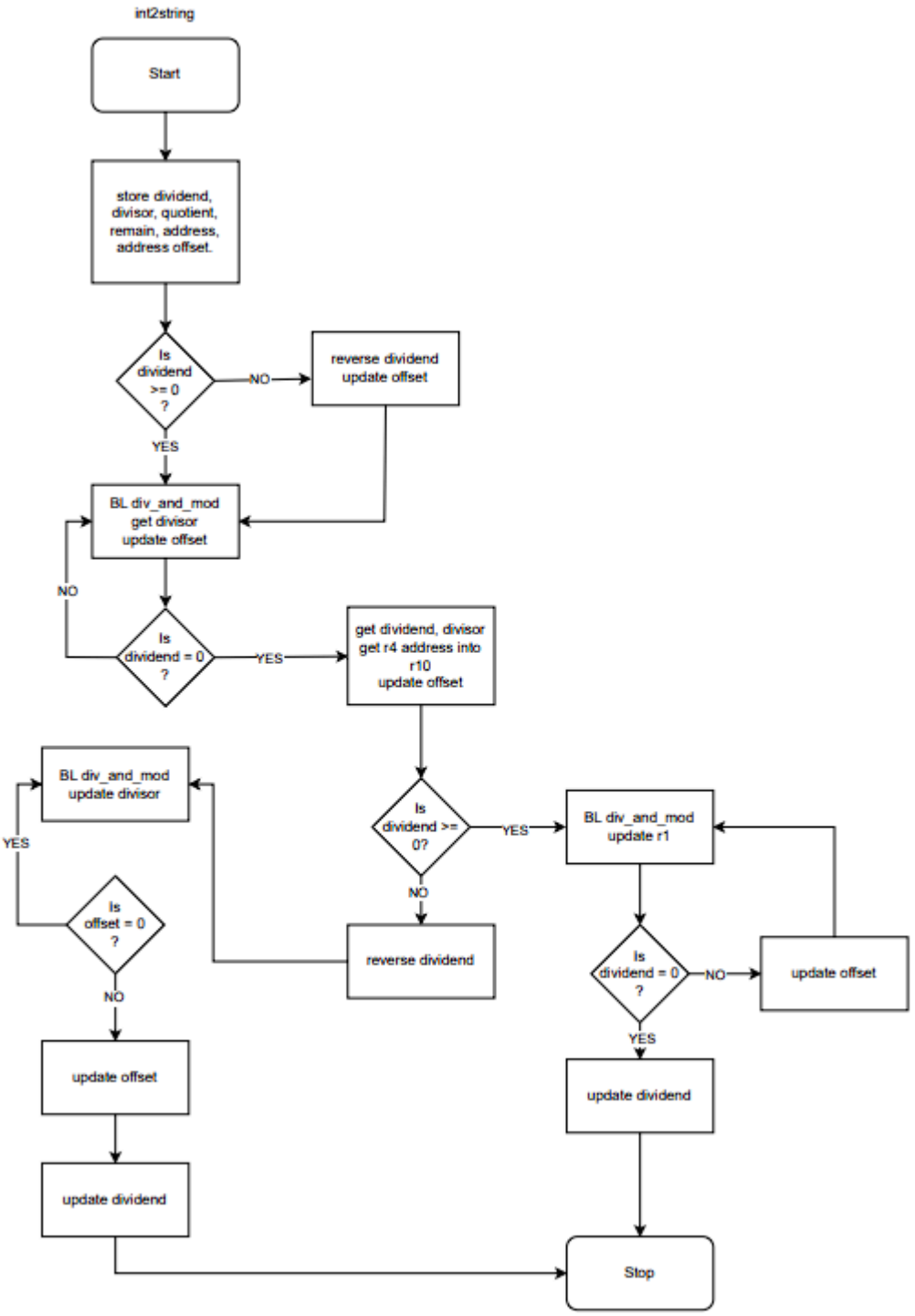
lab3

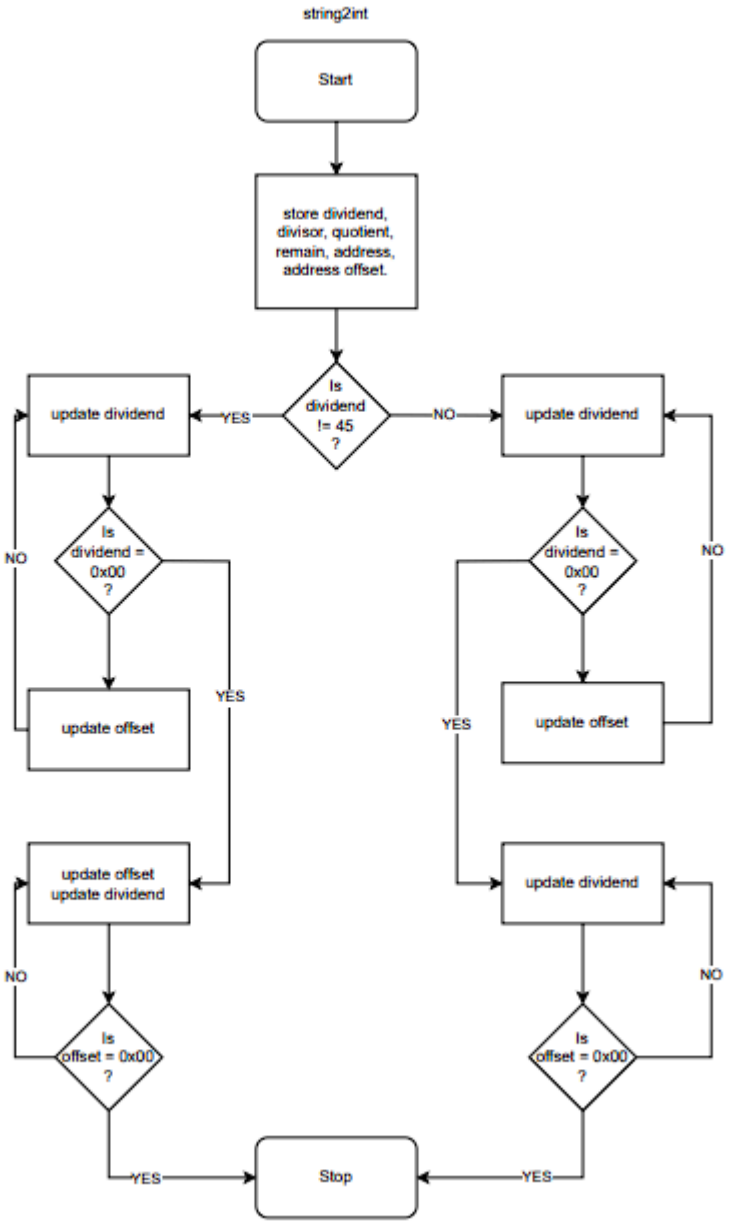
Subroutine Flowcharts

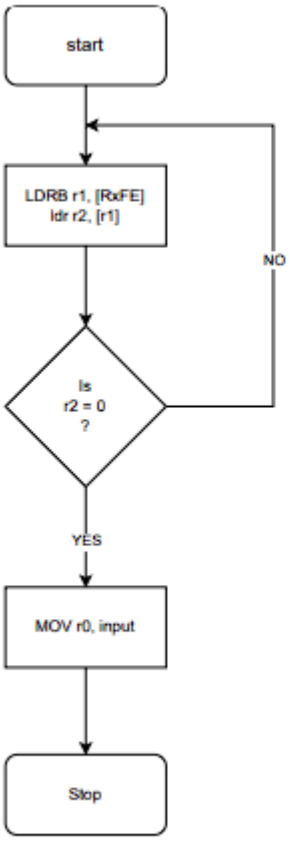
div_and_mod:

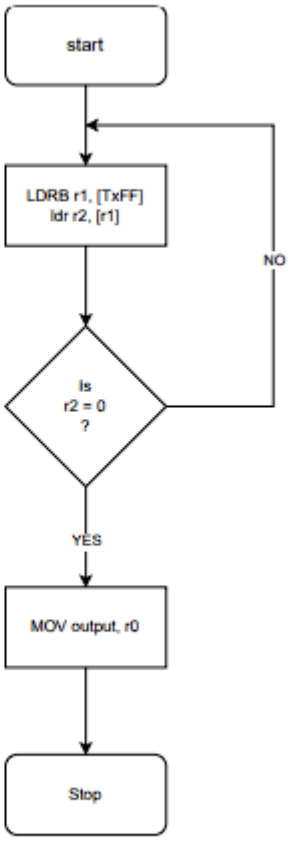


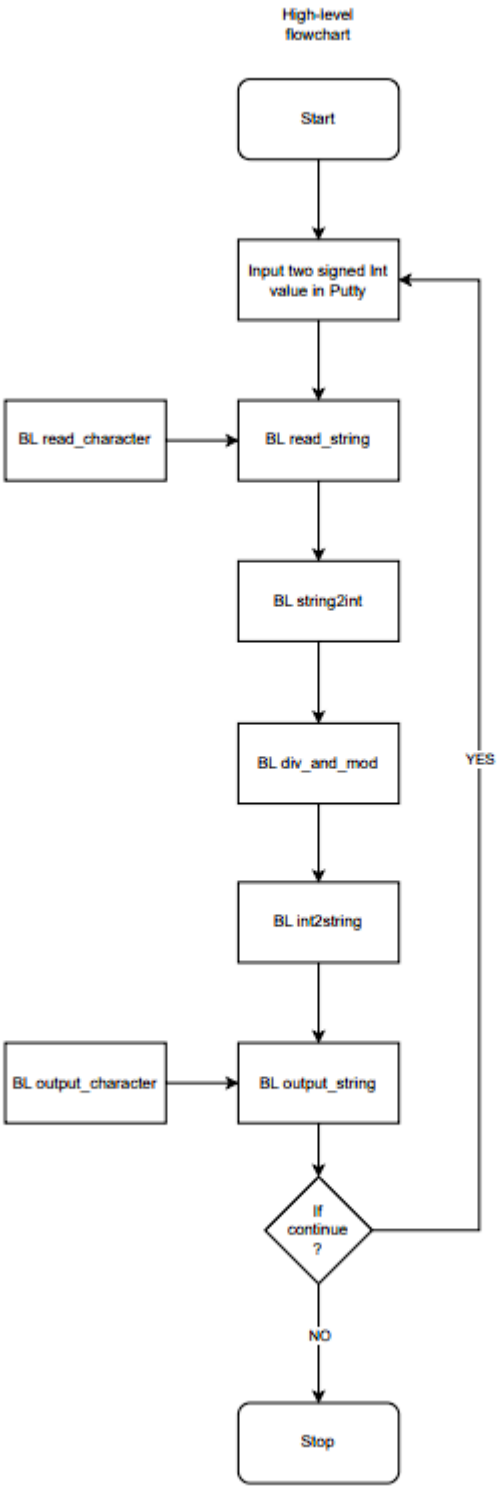

output_string
Start
store dividend
init offset
update r0 address
Is
dividend = 0x00
?
YES
reset r0
NO
BL output_character
Stop
read_string
Start
initialization
update dividend
upodate offset
NO
Is
dividend = 0x00
?
YES
dividend:= 0
update offset
update dividend
Stop


int2string
Start
store dividend, divisor, quotient, remain, address, address offset.
Is dividend >= 0 ?
NO
reverse dividend update offset
YES
BL div_and_mod get divisor update offset
NO
Is dividend = 0 ?
YES
get dividend, divisor get r4 address into r10 update offset
BL div_and_mod update divisor
Is dividend >= 0?
YES
BL div_and_mod update r1
YES
Is offset = 0 ?
NO
reverse dividend
Is dividend = 0 ?
NO
update offset
update offset
YES
update dividend
update dividend
Stop


string2int
Start
store dividend, divisor, quotient, remain, address, address offset.
Is dividend != 45 ?
YES
NO
update dividend
update dividend
Is dividend = 0x00 ?
Is dividend = 0x00 ?
NO
NO
YES
YES
update offset
update offset
update offset update dividend
update dividend
NO
NO
Is offset = 0x00 ?
Is offset = 0x00 ?
YES
YES
Stop


start
LDRB r1, [RxFE]
ldr r2, [r1]
Is
r2 = 0
?
NO
YES
MOV r0, input
Stop


start
LDRB r1, [TxFF]
ldr r2, [r1]
Is
r2 = 0
?
NO
YES
MOV output, r0
Stop


High-level
flowchart
Start
Input two signed Int
value in Putty
BL read_character
BL read_string
BL string2int
BL div_and_mod
BL int2string
BL output_character
BL output_string
If
continue
?
YES
NO
Stop