НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №1.2

з дисципліни“Архітектура комп’ютерів – 3”

Виконала:

студентка групи ІВ-81

ЗК ІВ-8105

Боровець Юлія

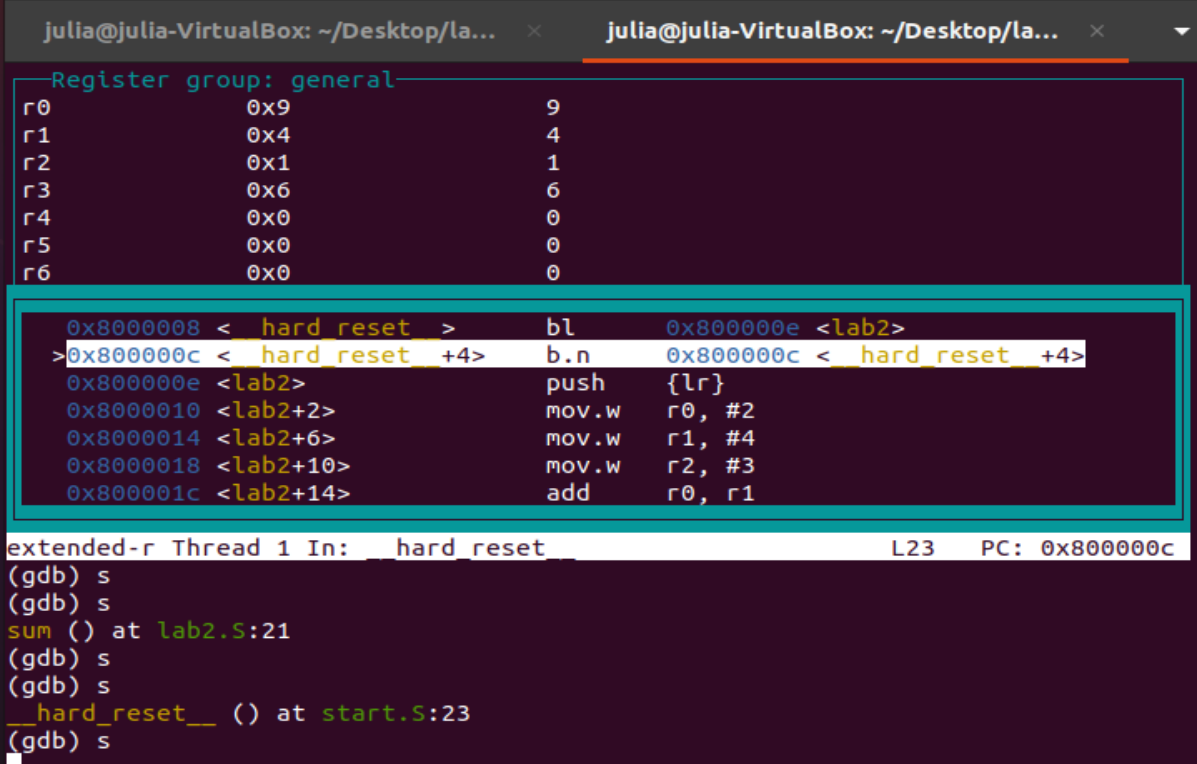
Київ 2021

**Тема :** основні інструкції 32-бітного ARM процесора для мікроконтролерів.

**Мета :** навчитися використовувати асемблерні інструкції ядра Cortex-M4,

працювати з процедурами і базово зрозуміти архітектуру ядра

**Скріншоти роботи:**



**Дані:**

a=2, b=4, c=3

(a+b)/2 + c! = 6/2+6=9

**Лістинг коду**

1. **Makefile**

SDK\_PREFIX?=arm-none-eabi-

CC = $(SDK\_PREFIX)gcc

LD = $(SDK\_PREFIX)ld

SIZE = $(SDK\_PREFIX)size

OBJCOPY = $(SDK\_PREFIX)objcopy

QEMU = qemu-system-gnuarmeclipse

BOARD ?= STM32F4-Discovery

MCU=STM32F407VG

TARGET=firmware

CPU\_CC=cortex-m4

TCP\_ADDR=1234

deps = \

start.S \

lscript.ld

all: target

target:

$(CC) -x assembler-with-cpp -c -O0 -g3 -mcpu=$(CPU\_CC) -Wall start.S -o start.o

$(CC) -x assembler-with-cpp -c -O0 -g3 -mcpu=$(CPU\_CC) -Wall lab2.S -o lab1.o

$(CC) start.o lab1.o -mcpu=$(CPU\_CC) -Wall --specs=nosys.specs -nostdlib -lgcc -T./lscript.ld -o $(TARGET).elf

$(OBJCOPY) -O binary -F elf32-littlearm $(TARGET).elf $(TARGET).bin

qemu:

$(QEMU) --verbose --verbose --board $(BOARD) --mcu $(MCU) -d unimp,guest\_errors --image $(TARGET).bin --semihosting-config enable=on,target=native -gdb tcp::$(TCP\_ADDR) -S

clean:

-rm \*.o

-rm \*.elf

-rm \*.bin

flash:

st-flash write $(TARGET).bin 0x08000000

1. **lscript.ld**

MEMORY

{

FLASH ( rx ) : ORIGIN = 0x08000000, LENGTH = 1M

RAM ( rxw ) : ORIGIN = 0x20000000, LENGTH = 128K

}

\_\_stack\_start = ORIGIN(RAM) + LENGTH(RAM);

1. **start.S**

.syntax unified

.cpu cortex-m4

//.fpu softvfp

.thumb

// Global memory locations.

.global vtable

.global \_\_hard\_reset\_\_

/\*

\* vector table

\*/

.type vtable, %object

.type \_\_hard\_reset\_\_, %function

vtable:

.word \_\_stack\_start

.word \_\_hard\_reset\_\_+1

.size vtable, .-vtable

\_\_hard\_reset\_\_:

// initialize stack here

// if not initialized yet

bl lab2

\_loop: b \_loop

.size \_\_hard\_reset\_\_, .-\_\_hard\_reset\_\_

1. **lab2.S**

.global lab2

.syntax unified

#define a #2

#define b #4

#define c #3

lab2:

push {lr}

mov r0, a

mov r1, b

mov r2, c

add r0, r1

lsr r0, r0, #1

mov r3, #1

factorial:

cmp r2, #1

beq sum

mul r3, r2

sub r2, #1

bne factorial

sum:

add r0, r3

pop {pc}

**Висновок**

Було створено програмний проект на мові асемблера та перевірено його виконання відлагоджувачем.

В результаті, отримано очікувані результати, що показано на скріншотах виконання програми.