

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-8101
Базова Лідія

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.

Варіант: 1

$$(a-b)*3 + 2^c$$

Скріншоти роботи:

The screenshot shows an ARM assembler interface. At the top, a table lists the values of general-purpose registers r0 through r5. Below this, a list of assembly instructions is displayed with their corresponding memory addresses. The instructions include a branch, a push, and two moves followed by a subtraction. At the bottom, the current thread and program counter (PC) are shown.

Register	Value
r0	0x2e
r1	0x0
r2	0x10
r3	0x2
r4	0x0
r5	0x0

Address	Instruction
0x8000008	< hard reset > bl 0x800000e <lab1>
0x800000c	< hard reset +4> b.n 0x800000c < hard reset +4>
0x800000e	<lab1> push {lr}
0x8000010	<lab1+2> mov.w r0, #16
0x8000014	<lab1+6> mov.w r1, #6
0x8000018	<lab1+10> sub.w r0, r0, r1

extended-r Thread 1 In: hard reset L23 PC: 0x800000c

Вхідні данні:

a = 16

b = 6

c = 4

результат = $(16-6)*3 + 2^c = 46$ (регістр r0)

Лістинг коду

1. lab1.S

```
.global lab1
.syntax unified
#define A #16
#define B #6
#define C #4
lab1:
    push {lr}
    // calculate
    mov r0, A
    mov r1, B
    SUB r0, r0, r1
    mov r1, #3
    MUL r0, r0, r1
```

```

mov r1, C
mov r2, #1
mov r3, #2

.L1:
    cmp r1, #0
    ble .L2
    mul r2, r2, r3

    sub r1, r1, #1
    b .L1

```

```

.L2:
    add r0, r0, r2
pop {pc}

```

2. lscript.ld

```

MEMORY
{
    FLASH ( rx )      : ORIGIN = 0x08000000, LENGTH = 1M
    RAM ( rxw )       : ORIGIN = 0x20000000, LENGTH = 128K
}

__stack_start = ORIGIN(RAM) + LENGTH(RAM);

```

3. 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 lab1.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
```

4. 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 lab1
    _loop: b _loop
    .size __hard_reset__, .-__hard_reset__
```

Висновок

Ми створили програмний проект на мові асемблера, що вирішує заданий математичний вираз, та перевірили його виконання відлагоджувачем. Отримали очікувані результати обчислень на регістрах, що показані на скріншотах виконання програми.