Міністерство освіти та науки України Національний технічний університет України «Київський політехнічний інститут ім. Ігоря Сікорського» Факультет інформатики і обчислювальної техніки Кафедра обчислювальної техніки

Лабораторна робота №2

з курсу «Архітектура комп'ютерів – 3»

Виконав:

студент 3 курсу ФІОТ

групи IB-81:

Бухтій О. В., 8106

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.

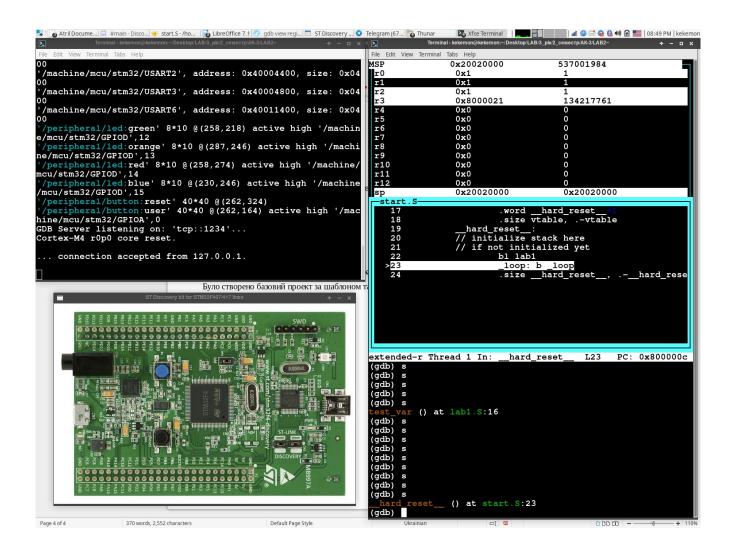
```
Bapiaht: 1 ( (a-b)*3 + 2^c ).
Лістинг:
---Makefile---:
PATH := ${PATH}:/opt/xPacks/gemu-arm/2.8.0-12/bin:\
/opt/gcc-arm-9.2-2019.12-x86 64-arm-none-eabi/bin
SDK PREFIX?=arm-none-eabi-
CC = \$(SDK\_PREFIX)gcc
LD = \$(SDK\_PREFIX)ld
SIZE = $(SDK_PREFIX)size
OBJCOPY = $(SDK PREFIX)objcopy
QEMU = qemu-system-gnuarmeclipse
BOARD ?= STM32F4-Discovery
MCU=STM32F407VG
TARGET=firmware
CPU_CC=cortex-m4
TCP_ADDR=1234
deps = \
     start.S\
     lscript.ld
all: target
target:
     $(CC) -x assembler-with-cpp -c -00 -g3 -mcpu=$(CPU CC) -Wall
start.S -o start.o
     $(CC) -x assembler-with-cpp -c -00 -g3 -mcpu=$(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) -0 binary -F elf32-littlearm $(TARGET).elf $
(TARGET).bin
qemu:
     $(QEMU) --verbose --verbose --board $(BOARD) --mcu $(MCU) -d
unimp, quest errors --image $(TARGET).bin --semihosting-config
enable=on, target=native -qdb tcp::$(TCP ADDR) -S
clean:
     -rm *.o
```

-rm *.elf
-rm *.bin

```
flash:
     st-flash write $(TARGET).bin 0x08000000
---lab1.s--- :
.global lab1
.syntax unified
#define A #1
#define B #1
#define C #1
lab1:
     push {lr} // (a-b)*3+(2^c) = 2
     mov r0, A
     mov r1, B
     mov r2, C
     bl test_var
     pop {pc}
test_var:
     push { r0, r1, r2, lr }
     sub r0, r1 //a-b
     mov r3, #3
     mul r0, r3 //*3
     mov r3, #2
     sub r3, #1
     lsl r2, r3, r2 //2^c
     add r0, r2// +
     pop { r0, r1, r2, r3, pc }
---lscript.ld---:
MEMORY
{
     FLASH ( rx ) : ORIGIN = 0x08000000, LENGTH = 1M
     RAM ( rxw ) : ORIGIN = 0x20000000, LENGTH = 128K
__stack_start = ORIGIN(RAM) + LENGTH(RAM);
---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__
```

Результат:



Висновки

Було створено програму яка рахує заданий вираз, її роботу продемонстровано в gdb. Було вивчено базові інструкції ассемблера для процесорів з архітекурою ARM.