НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота № 1.1
з дисципліни
“Архітектура комп’ютерів – 3”

Виконала:
студентка групи ІВ-81
ЗК ІВ-8101
Базова Лідія

Київ 2021

Тема: Створення мінімального програмного проекту на мові асемблера.

Мета: Створити мінімальний проект, перевірити виконання відлагоджувачем.

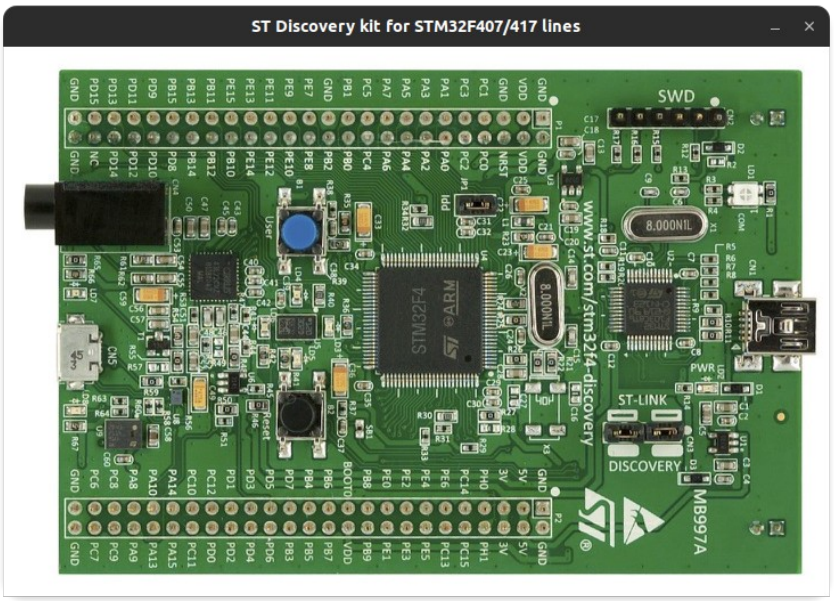
Скріншоти роботи:

make

```
lydia@lydia-Latitude:~/Documents/AK/project/lab1.1$ ls
AK_1.1_Bazova.odt  lscript.ld  makefile  MakeFile  start.S
lydia@lydia-Latitude:~/Documents/AK/project/lab1.1$ make
arm-none-eabi-gcc -x assembler-with-cpp -c -O0 -g3 -mcpu=cortex-m4  -Wall start.S -o start.o
arm-none-eabi-gcc start.o -mcpu=cortex-m4  -Wall --specs=nosys.specs -nostdlib -lgcc -T./lscript.ld -o firmware.elf
arm-none-eabi-objcopy -O binary -F elf32-littlearm  firmware.elf  firmware.bin
lydia@lydia-Latitude:~/Documents/AK/project/lab1.1$ ls
AK_1.1_Bazova.odt  firmware.bin  firmware.elf  lscript.ld  makefile  MakeFile  start.o  start.S
```

```
lydia@lydia-Latitude:~/Documents/AK/project/lab1.1$ qemu-system-gnuarmeclipse --verbose --verbose --board STM32F4-D
iscovery --mcu STM32F407VG -d unimp,guest_errors --image firmware.bin --semihosting-config enable=on,target=native
-s -S

xPack 64-bit QEMU v2.8.0-7 (qemu-system-gnuarmeclipse).
Board: 'STM32F4-Discovery' (ST Discovery kit for STM32F407/417 lines).
Board picture: '/home/lydia/opt/xPacks/qemu-arm/2.8.0-7/share/qemu/graphics/STM32F4-Discovery.jpg'.
Device file: '/home/lydia/opt/xPacks/qemu-arm/2.8.0-7/share/qemu/devices/STM32F40x-qemu.json'.
Device: 'STM32F407VG' (Cortex-M4 r0p0, MPU, 4 NVIC prio bits, 82 IRQs), Flash: 1024 kB, RAM: 128 kB.
Image: 'firmware.bin'.
Command line: (none).
Load     20 bytes at 0x08000000-0x08000013.
Cortex-M4 r0p0 core initialised.
'/machine/mcu/stm32/RCC', address: 0x40023800, size: 0x0400
'/machine/mcu/stm32/FLASH', address: 0x40023C00, size: 0x0400
'/machine/mcu/stm32/PWR', address: 0x40007000, size: 0x0400
'/machine/mcu/stm32/SYSCFG', address: 0x40013800, size: 0x0400
'/machine/mcu/stm32/EXTI', address: 0x40013C00, size: 0x0400
'/machine/mcu/stm32/GPIOA', address: 0x40020000, size: 0x0400
'/machine/mcu/stm32/GPIOB', address: 0x40020400, size: 0x0400
'/machine/mcu/stm32/GPIOC', address: 0x40020800, size: 0x0400
'/machine/mcu/stm32/GPIOD', address: 0x40020C00, size: 0x0400
'/machine/mcu/stm32/GPIOE', address: 0x40021000, size: 0x0400
'/machine/mcu/stm32/GPIOF', address: 0x40021400, size: 0x0400
'/machine/mcu/stm32/GPIOG', address: 0x40021800, size: 0x0400
'/machine/mcu/stm32/GPIOH', address: 0x40021C00, size: 0x0400
'/machine/mcu/stm32/GPIOI', address: 0x40022000, size: 0x0400
'/machine/mcu/stm32/USART1', address: 0x40011000, size: 0x0400
'/machine/mcu/stm32/USART2', address: 0x40004400, size: 0x0400
'/machine/mcu/stm32/USART3', address: 0x40004800, size: 0x0400
'/machine/mcu/stm32/USART6', address: 0x40011400, size: 0x0400
'/peripheral/led:green' 8*10 @(258,218) active high '/machine/mcu/stm32/GPIOD',12
'/peripheral/led:orange' 8*10 @(287,246) active high '/machine/mcu/stm32/GPIOD',13
'/peripheral/led:red' 8*10 @(258,274) active high '/machine/mcu/stm32/GPIOD',14
'/peripheral/led:blue' 8*10 @(230,246) active high '/machine/mcu/stm32/GPIOD',15
'/peripheral/button:reset' 40*40 @(262,324)
'/peripheral/button:user' 40*40 @(262,164) active high '/machine/mcu/stm32/GPIOA',0
GDB Server listening on: 'tcp::1234'...
Cortex-M4 r0p0 core reset.
```




```
lydia@lydia-Latitude:~/Documents/AK/project/lab1.1$ gdb-multiarch firmware.elf
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from firmware.elf...
(gdb) target extended-remote:1234
Remote debugging using :1234
__hard_reset__ () at start.S:18
18          ldr r0, =__stack_start
(gdb) step
19          mov sp, r0
(gdb) step
__hard_reset__ () at start.S:20
20          b __hard_reset__
(gdb) step
__hard_reset__ () at start.S:18
18          ldr r0, =__stack_start
(gdb)
```

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
       $(CC) start.o -mcpu=$(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
```

2. lscript.ld

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

3. start.S

```
.syntax unified
.cpu cortex-m4
//.fpu softvfp
.thumb

// Global memory locations.
.global vtable
.global reset_handler
/*
 * vector table
 */
.type vtable, %object
vtable:
    .word __stack_start
    .word __hard_reset__+1
    .size vtable, .-vtable
__hard_reset__:
    ldr r0, =__stack_start
    mov sp, r0
    b __hard_reset__
```

Висновок

Ми створили програмний проект на мові асемблера та перевірили його виконання відлагоджувачем. Отримали очікувані результати, що показані на скріншотах виконання програми.