

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КІЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

КАТЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Архітектура комп'ютерів

Лабораторна робота №1

«СТВОРЕННЯ ПРОГРАМНОГО ПРОЕКТУ НА МОВІ АСЕМБЛЕРА»

Виконав:
студент групи ІО-32
Крадожон М.

Перевірив(-ла):
Камплунов А.

Лабораторна робота №1

Відомості

Тема

СТВОРЕННЯ ПРОГРАМНОГО ПРОЕКТУ НА МОВІ АСЕМБЛЕРА

Мета

Створити мінімальний програмний проект на мові асемблер, перевірити виконання відлагоджувачем.

Хід роботи

Підготовка середовища

Dockerfile

```
FROM ubuntu:22.04

RUN apt-get update && apt-get install -y \
    gcc-arm-none-eabi \
    binutils-arm-none-eabi \
    libnewlib-arm-none-eabi \
    gdb-multiarch \
    stlink-tools \
    make \
    wget \
    tar \
    libx11-6 \
    libxext6 \
    libxrender1 \
    libxtst6 \
    libasound2 \
&& rm -rf /var/lib/apt/lists/*

WORKDIR /opt
RUN wget https://github.com/xpack-dev-tools/qemu-arm-xpack/releases/download/v7.2.0-1/xpack-qemu-arm-7.2.0-1-linux-x64.tar.gz \
    && tar xvf xpack-qemu-arm-7.2.0-1-linux-x64.tar.gz \
    && rm xpack-qemu-arm-7.2.0-1-linux-x64.tar.gz

ENV PATH="$PATH:/opt/xpack-qemu-arm-7.2.0-1/bin"

WORKDIR /app
```

Будуємо образ

```
maxim@x540ubr:~$ docker build -t stm32-kradozhon-io32 .
...
maxim@x540ubr:~$ docker run -it -v $(pwd):/app stm32-kradozhon-io32
```

Тестуємо образ

```
root@52cf8b845a48:/app# /opt/xpack-qemu-arm-7.2.0-1/bin/qemu-system-gnuarmeclipse --version

xPack QEMU emulator version 2.8.0 (v2.8.0-16-xpack-legacy)
Copyright (c) 2003-2016 Fabrice Bellard and the QEMU Project developers
```

Збірка проекту

start.S

```
.syntax unified
.cpu cortex-m4

.thumb

.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__
```

lscript.ld

```
/* linker script for stm32f1
* platforms
* Define the end of RAM and limit of stack memory
*/
MEMORY
{
    /* We mark flash memory as read-only, since that is where the program lives.
     STM32 chips map their flash memory to start at 0x08000000, and we have 32KB of
     flash memory available. */
    FLASH ( rx ) : ORIGIN = 0x08000000, LENGTH = 1M
    /* We mark the RAM as read/write, and as mentioned above it is 4KB long
     starting at address 0x20000000. */
    RAM ( rxw ) : ORIGIN = 0x20000000, LENGTH = 128K
}
__stack_start = ORIGIN(RAM) + LENGTH(RAM);
```

Збірка

```
root@52cf8b845a48:/app# arm-none-eabi-gcc -x assembler-with-cpp -c -O0 -g3 -mcpu=cortex-m4 -mthumb -Wall start.S -o start.o
root@52cf8b845a48:/app# arm-none-eabi-gcc start.o -mcpu=cortex-m4 -mthumb -Wall --specs=nosys.specs -nostdlib -lgcc -T./lscript.ld -o firmware.elf
root@52cf8b845a48:/app# arm-none-eabi-gcc start.o -mcpu=cortex-m4 -mthumb -Wall --specs=nosys.specs -nostdlib -lgcc -T./lscript.ld -o firmware.elf firmware.bin

root@52cf8b845a48:/app# qemu-system-gnuarmeclipse --verbose --verbose --board STM32F4-Discovery --mcu STM32F407VG -d unimp,guest_errors --image firmware.bin --semihosting -config enable=on,target=native -s -S -nographic

xPack 64-bit QEMU v2.8.0 (qemu-system-gnuarmeclipse).
Board: 'STM32F4-Discovery' (ST Discovery kit for STM32F407/417 lines).
Device file: '/opt/xpack-qemu-arm-7.2.0-1/share/legacy/qemu/devices/STM32F40x-qemu.json'.
Device: 'STM32F407VG' (Cortex-M4 r0p0, MPU, ITM, 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
GDB Server listening on: 'tcp::1234'...
```

```
QEMU 2.8.0 monitor - type 'help' for more information
(qemu) Cortex-M4 r0p0 core reset.
```

```
root@52cf8b845a48:/app# gdb-multiarch firmware.elf -ex="target extended-remote:1234"
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04.2) 12.1
Copyright (C) 2022 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:
<https://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...
```

```
Remote debugging using :1234
```

```
__hard_reset__ () at start.S:17
```

```
17      ldr r0, __stack_start
```

```
(gdb) step
```

```
18      mov sp, r0
```

```
(gdb)
```

```
__hard_reset__ () at start.S:19
```

```
19      b __hard_reset__
```

```
(gdb)
```

Створення прошивки

```
SDK_PREFIX = arm-none-eabi-
CC      = $(SDK_PREFIX)gcc
OBJCOPY = $(SDK_PREFIX)objcopy
QEMU    = qemu-system-gnuarmeclipse

BOARD  = STM32F4-Discovery
MCU    = STM32F407VG
TARGET = firmware

all: target

target:
    $(CC) -x assembler-with-cpp -c -O0 -g3 -mcpu=cortex-m4 -mthumb -Wall start.S -o start.o
    $(CC) start.o -mcpu=cortex-m4 -mthumb -Wall --specs=nosys.specs -nostdlib -lgcc -T./lscript.ld -o $(TARGET).elf
    $(OBJCOPY) -O binary $(TARGET).elf $(TARGET).bin

qemu:
    $(QEMU) --board $(BOARD) --mcu $(MCU) -d unimp,guest_errors --image $(TARGET).bin --semihosting-config enable=on,target=native -s -S -nographic

clean:
    rm -f *.o *.elf *.bin
```

```
root@52cf8b845a48:/app# make
arm-none-eabi-gcc -x assembler-with-cpp -c -O0 -g3 -mcpu=cortex-m4 -mthumb -Wall start.S -o start.o
arm-none-eabi-gcc start.o -mcpu=cortex-m4 -mthumb -Wall --specs=nosys.specs -nostdlib -lgcc -T./lscript.ld -o firmware.elf
arm-none-eabi-objcopy -O binary firmware.elf firmware.bin
root@52cf8b845a48:/app# make qemu
qemu-system-gnuarmeclipse --board STM32F4-Discovery --mcu STM32F407VG -d unimp,guest_errors --image firmware.bin --semihosting-config enable=on,target=native -s -S -nographic
QEMU 2.8.0 monitor - type 'help' for more information
(qemu)
```

Висновки

У ході виконання лабораторної роботи було успішно розгорнуто середовище розробки на базі **Docker-контейнера** з використанням крос-компілятора **arm-none-eabi-gcc** та емулятора **QEMU**. Нами було розроблено асемблерний файл **start.S**, що містить таблицю векторів виключень та обробник скидання для ініціалізації стека, а також скрипт лінкера **lscript.ld**, який визначає архітектуру пам'яті **FLASH** та **RAM**. Процес збірки та запуску проекту було автоматизовано за допомогою **Makefile**, що дозволило ефективно генерувати прошивку у форматі **.bin** та запускати її в емуляторі. Фінальна верифікація за допомогою відлагоджувача **GDB** та покрокового виконання команд підтвердила коректну ініціалізацію регістрів **r0** та **sp**, що свідчить про успішне досягнення мети роботи щодо створення та тестування мінімального програмного проекту для архітектури **Cortex-M4**.

Github

<https://github.com/TockePie/comp-arch-3/tree/main/lab1>