

МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Кафедра обчислювальної техніки

**Методичні вказівки**

# Архітектура ARM-CORTEX-M4 та теоретичні основи програмування для

# 32-бітних мікроконтролерів STM32F4

Розробники:

студент групи ІО-71 Муравйов І.П.

(посада, вчена ступінь та звання П.І.Б.)

Рецензент: доц. Клименко І. А.

Київ – 2019

*Метою створення цього файлу*

*було оновлення програми викладання дисципліни*

***“Архітектура комп’ютера”****. Зверніть увагу, що матеріал орієнтований на студентів, що вивчали “Вступ до ОС Linux” на першому курсі та “Системне Програмування — 1” на другому. Кожна лабораторна робота є логічним продовженням попередньої, отже всі лабораторні важливі.*

*Матеріали до циклу лабораторних робіт:*

[*https://github.com/Igor1101/CompArch2STM32*](https://github.com/Igor1101/CompArch2STM32)

*Теоретичні матеріали:*

[*https://drive.google.com/drive/folders/1QrUx5Zo0fMOKkudD9dEqX9VxrrUNg-lR*](https://drive.google.com/drive/folders/1QrUx5Zo0fMOKkudD9dEqX9VxrrUNg-lR)

Автор цього документу не несе відповідальності за успішно здані або не здані роботи.

**1. Теоретична інформація**

**1.1 Архітектура ARM**

Advanced RISC Machine (ARM) є сім’єю найбільш розповсюджених ядер у вбудованих системах в світі (станом на 2019 рік). Розроблене компанією Arm Holdings, що дає можливість іншим компаніям розробляти на основі цього сімейства процесори та системи на чіпі будь-якої складності.

Хоч повний опис архітектури і закритий(на нього можна купити ліцензію), все таки є достатньо відкритої інформації, що стосується саме ядра ARM процесора, щоби можна було вивчати сучасні RISC процесори, їх можливості, та деякі відкриті системи, в які вбудовані ці процесори.

Існує ціла низка спеціалізованих ARM мікроархітектур: ARM1-7, ARM7T, ARM10E, Cortex-A (32-bit), Neoverse та інші. Найбільш поширеними зараз є ARM-Cortex ядра.

Нижче приводиться таблиця застосування та особливостей різних ARM-Cortex сімейств.

|  |  |  |
| --- | --- | --- |
| Сімейство | Використання | Особливості |
| ARM-Cortex-M  M=Microcontroller Application | Мікроконтролери високої та низької швидкодії загального та спеціалізованого призначення, сигнальні процесори (DSP). | Найдешевше сімейство,  порівняно невисока швидкодія, низьке енергоспоживання,  маленький розмір ядра на кристалі, віртуальна пам’ять відсутня. |
| ARM-Cortex-R  R=Real-Time | Сигнальні процесори високої якості, системи на чипі(SOC),  мікроконтролери спеціального призначення. | Схоже на Cortex-M, але більш підходить для систем індустріального призначення,  найдорожче сімейство. |
| ARM-Cortex-A  A=Application | Системи на чипі(SOC),  мобільні процесори,  кластерні суперкомп’ютерні системи, процесори загального призначення | Висока швидкодія, високе енергоспоживання, існує як 32-бітна так і 64-бітна архітектура. |

Таблиця 1. Використання та особливості ARM-Cortex сімейств

В цьому курсі ми будемо вивчати саме **M** сімейство.

Сімейство ARM Cortex-M - це мікропроцесорні ядра ARM, призначені для використання в мікроконтролерах , ASIC , ASSP , FPGA і SoC . Ядра Cortex-M зазвичай використовуються як спеціальні чіпи мікроконтролера, але також "приховані" всередині мікросхем SoC як контролери управління потужністю, контролери вводу / виводу, системні контролери та інше. Cortex-M стали популярною заміною для 8-бітових мікросхем у додатках, які користуються 32-бітовими математичними операціями та заміною старих застарілих ядер ARM, таких як ARM7 та ARM9.

Нижче приведена таблиця використання та особливостей різних ARM-Cortex-M ядер.

|  |  |  |
| --- | --- | --- |
| Ядро\Архітектура | Використання | Особливості |
| Cortex-M0  ARMv6-M  \*\* | Заміна 8 та 16-бітних мікроконтролерів в старих системах. | Оптимізована для малих розмірів та використання в найменших чіпах в дешевих системах. |
| Cortex-M0+  ARMv6-M  \*  \*\* | Системи на батарейках. | Те ж саме, що і Cortex-M0, але оптимізована ще й для енергозбереження. |
| Cortex-M1  ARMv6-M  \* | FPGA. | Розроблене для того, щоб запускатися з програмованих інтегральних схем. |
| Cortex-M3  ARMv7-M  \* | Мікроконтролери загального призначення. | Оптимізоване для операційних систем реального часу(РТОС),  можлива підтримка ОС загального призначення (uClinux). |
| Cortex-M4  ARMv7E-M  \* | Заміна Cortex-M3. | Те ж саме, що і Cortex-M3, але більша швидкодія. |
| Cortex-M7  ARMv7E-M  \* | Мікроконтролери високої швидкодії. | Те ж саме, що і Cortex-M4, але більша швидкодія. |
| Cortex-M23  ARMv8-M  \*  \*\* | Заміна Cortex-M0+ | Те ж саме, що і Cortex-M0+, але більший набір інструкцій. |

\*- опціонально є блок захисту пам’яті (MPU)

\*\* - блок операцій з плаваючою комою (FPU) завжди відсутній

Таблиця 2. Використання та особливості ARM-Cortex-M ядер

Ми будемо вивчати Cortex-M4. Навчившись працювати з будь-якою мікроархітектурою можна легко перейти до іншої у рамках Cortex-M.

Існує декілька компаній, що розробляють мікроконтролери на Cortex-M4 архітектурі: Microchip(Atmel), TI, ST Microelectronics, Cypress Semiconductors, Silicon Labs, NXP, Nordic

Даний документ описує ST Microelectronics STM32F4XX архітектуру, що представлена цілою низкою мікроконтролерів (далі МК).

Рис 1. Мікроконтролери STM32F4XX

**1.2 Опис мікроконтролера МК STM32F407VG**

Ми будемо працювати з МК STM32F407VG, що має:

1. ядро Cortex-M4 з FPU на архітектурі ARMv7E-M;
2. адаптивний прискорювач реального часу загрузки програми з флеш пам’яті;
3. 1 МБ флеш пам’яті (FLASH);
4. 192 кб оперативної пам’яті (SRAM);
5. контролер зовнішньої статичної пам’яті;
6. паралельний порт виводу на екран 8080/6080;
7. годинник реального часу(RTC) на 32khz резонаторі;
8. блок харчування SRAM від батарейки;
9. контролер прямого доступу до пам’яті (DMA);
10. 17 таймеров загального призначення;
11. модуль програмування\відладки SWD\JTAG;
12. 140 портов вводу виводу з можливістю переривання;
13. інтерфейси:
14. 4 i2c порта,
15. 4 UART\USART порта,
16. 3 SPI шини,
17. 2 CAN шини
18. інтерфейс карти пам’яті SD (SDIO),
19. USB2.0 OTG FULL SPEED,
20. USB2.0 SLAVE,
21. 10/100 Ethernet MAC.

**1.2.1 Процесорне ядро**

Існує багато сімейств МК на ядрі Cortex-M4. В них багато відмінностей, але ядро та зв’язок ядра з периферією МК незмінне.

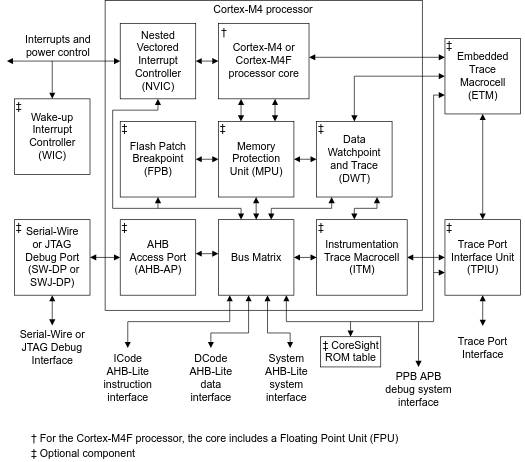


Рис 2. Структурна схема процесора(основної частини МК)

* NVIC - Nested Vectored Interrupt Controller(вкладений векторізований контролер переривань), що організовує внутрішні і зовнішні переривання;
* Serial Wire\JTAG — стандартний інтерфейс відладки;
* WIC — Wakeup Interrupt Controller (контролер запуску перериванням), що дозволяє

тримати МК виключеним поки не прийде зовнішнє переривання.

* MPU - Memory protection unit - підвищує надійність системи, визначаючи атрибути пам'яті для різних регіонів пам'яті.

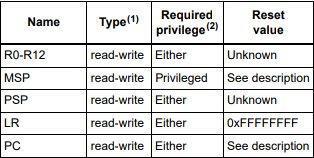


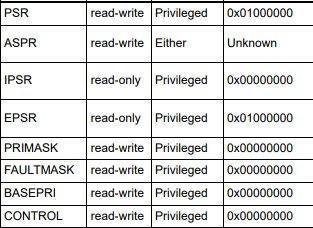
Рис 3. Модель програмування

**Привілейований (**Handler mode) – використовується в обробках переривань, виключень, завантажувачем, операційною системою. Всі інструкції та пам’ять доступні у такому режимі.

**Користувача** (Thread mode) – використовується для виконання програми користувача, це режим за замовчуванням.

Регістри ядра та АЛП





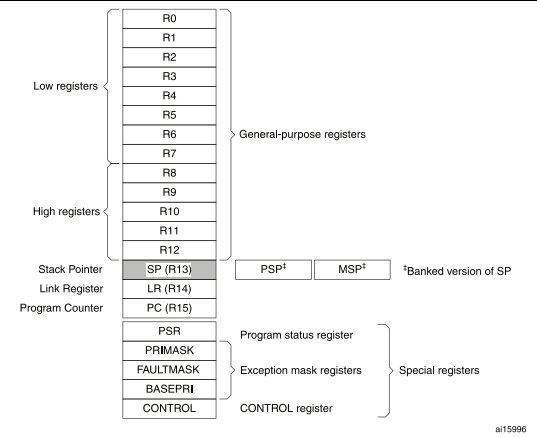


Рис 3. Регістри ядра та АЛП

* 32-бітні регістри загального призначення:R0-R12
* Вказівник стеку (Stack Pointer), R13
* В залежності від біту 1 в регістрі CONTROL:

bit[1]=1:

PSP: Process Stack Pointer

bit[1]=0:

MSP: Main Stack Pointer – за замовчуванням

* Link регістр (LR), R14 - зберігає інформацію про повернення для підпрограм, виклики функцій та винятки. Після сбросу процесор завантажує значення LR 0xFFFFFFFF.
* Програмний лічильник (PC), R15. Він містить поточну адресу програми. Після старту процесор завантажує PC на значення вектора RESET, яке знаходиться за адресою 0x00000004. Біт [0] значення завантажується в T-біт EPSR при скиданні процесора і повинен бути 1.

**2. Встановлення ПЗ**

1. Встановіть емулятор комп’ютерів qemu-system-gnuarmeclipse:

[https://xpack.github.io/qemu-arm/](https://xpack.github.io/qemu-arm/install/)

2. Завантажте архів з:

<https://github.com/xpack-dev-tools/qemu-arm-xpack/releases/>

та виконайте наступні команди:

>>> mkdir -p ~/opt

>>> cd ~/opt

>>> tar xvf ~/Downloads/xpack-qemu-arm-2.8.0-7-linux-x64.tgz

>>> chmod -R -w xPacks/qemu-arm/2.8.0-7

Перевірити виконання:

>>> ~/opt/xPacks/qemu-arm/2.8.0-7/bin/qemu-system-gnuarmeclipse --version

Результат успішного виконання:

xPack 64-bit QEMU emulator version 2.8.0-7 (v2.8.0-4-20190211-47-g109b69f49a-dirty)

Copyright (c) 2003-2016 Fabrice Bellard and the QEMU Project developers

3. Встановіть тулчейни:

1) arm-none-eabi-gcc

**Debian-based:**

>>> sudo apt-get install arm-none-eabi-gcc

*Або, якщо пакет не було знайдено:*

>>> sudo apt-get install gcc-arm-none-eabi

**Arch-based:**

>>> sudo pacman -S arm-none-eabi-gcc

2) arm-none-eabi-newlib

**Debian -based:**

>>> sudo apt-get install arm-none-eabi-newlib

*Або, якщо пакет не було знайдено:*

>>> sudo apt-get install libnewlib-arm-none-eabi

**Arch-based:**

>>> sudo pacman -S arm-none-eabi-newlib

3) arm-none-eabi-gdb

**Debian -based:**

>>> sudo apt-get install gdb-multiarch

*Або, якщо пакет не було знайдено:*

>>> sudo apt-get install arm-none-eabi-gdb

*Або, якщо пакет не було знайдено:*

>>> sudo apt-get install gdb-arm-none-eabi

**Arch-based:**

>>> sudo pacman -S arm-none-eabi-gdb

4) arm-none-eabi-binutils

**Debian -based:**

>>> sudo apt-get install arm-none-eabi-binutils

*Або, якщо пакет не було знайдено:*

>>> sudo apt-get install binutils-arm-none-eabi

**Arch-based:**

>>> sudo pacman -S arm-none-eabi-binutils

4. Встановіть утиліти:

1) stlink Firmware programmer for STM32 STLINK v1/v2 protocol

**Debian -based:**

Щоб встановити stlink-tools введіть в терміналі:

>>> sudo apt-get install stlink-tools

Якщо пакет не було знайдено, перейдіть за посиланням:

>>> <https://github.com/texane/stlink>

та виконайте інсталяцію згідно з вказівками **README.md** файлу**.**

*Альтернатива:*

вставьте в **/*etc/apt/sources.list*** *репозиторій*

>>> deb http://ftp.de.debian.org/debian sid main

Або командою:

>>> sudo apt-add-repository deb http://ftp.de.debian.org/debian sid main

**Arch-based:**

>>> sudo pacman -S stlink

2) make

**Debian -based:**

>>> sudo apt-get install make

**Arch-based:**

>>> sudo pacman -S make

Додаткова література:

1. PM0214 Programming manual STM32 Cortex®-M4 MCUs and MPUs programming manual - <https://www.st.com/content/ccc/resource/technical/document/programming_manual/6c/3a/cb/e7/e4/ea/44/9b/DM00046982.pdf/files/DM00046982.pdf/jcr:content/translations/en.DM00046982.pdf>