Chapters 1 and 3
ARM Processor Architecture

Embedded Systems with ARM Cortex-M

Updated: Monday, February 5, 2018
A Little about ARM – The company

- Originally Acorn RISC Machine (ARM)
- Later Advanced RISC Machine
- Then it became ARM Ltd owned by ARM Holdings (parent company)
- In 2016 SoftBank bought ARM for $31 billion

ARM:
- Develops the architecture and licenses it to other companies
- Other companies design their own products that implement one of those architectures—including systems-on-chips (SoC) and systems-on-modules (SoM) that incorporate memory, interfaces, radios, etc.
- It also designs cores that implement this instruction set and licenses these designs to a number of companies that incorporate those core designs into their own products.

ARM Processors:
- RISC based processors
- In 2010 alone, 6.1 billion ARM-based processor, representing 95% of smartphones, 35% of digital televisions and set-top boxes and 10% of mobile computers
- over 100 billion ARM processors produced as of 2017
- The most widely used instruction set architecture in terms of quantity produced

https://en.wikipedia.org/wiki/ARM_architecture
<table>
<thead>
<tr>
<th>Architecture</th>
<th>Core bit-width</th>
<th>ARM Holdings</th>
<th>Third-party</th>
<th>Profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARMv1</td>
<td>32</td>
<td>ARM1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ARMv2</td>
<td>32</td>
<td>ARM2, ARM250, ARM3</td>
<td>Amber, STORM Open Soft Core</td>
<td></td>
</tr>
<tr>
<td>ARMv3</td>
<td>32</td>
<td>ARM6, ARM7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ARMv4</td>
<td>32</td>
<td>ARM8</td>
<td>StrongARM, FA526, ZAP Open Source Processor Core</td>
<td></td>
</tr>
<tr>
<td>ARMv4T</td>
<td>32</td>
<td>ARM7TDMI, ARM9TDMI, SecurCore SC100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ARMv5TE</td>
<td>32</td>
<td>ARM7EJ, ARM9E, ARM10E</td>
<td>XScale, FA620TE, Ferocion, PJ1/Mohawk</td>
<td></td>
</tr>
<tr>
<td>ARMv6</td>
<td>32</td>
<td>ARM11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ARMv6-M</td>
<td>32</td>
<td>ARM Cortex-M0, ARM Cortex-M0+, ARM Cortex-M1, SecurCore SC000</td>
<td>Microcontroller</td>
<td></td>
</tr>
<tr>
<td>ARMv7-M</td>
<td>32</td>
<td>ARM Cortex-M3, SecurCore SC300</td>
<td>Microcontroller</td>
<td></td>
</tr>
<tr>
<td>ARMv7E-M</td>
<td>32</td>
<td>ARM Cortex-M4, ARM Cortex-M7</td>
<td>Microcontroller</td>
<td></td>
</tr>
<tr>
<td>ARMv8-M</td>
<td>32</td>
<td>ARM Cortex-M23, ARM Cortex-M33</td>
<td>Microcontroller</td>
<td></td>
</tr>
<tr>
<td>ARMv7-R</td>
<td>32</td>
<td>ARM Cortex-R4, ARM Cortex-R5, ARM Cortex-R7, ARM Cortex-R8</td>
<td>Real-time</td>
<td></td>
</tr>
<tr>
<td>ARMv8-R</td>
<td>32</td>
<td>ARM Cortex-R52</td>
<td>Real-time</td>
<td></td>
</tr>
<tr>
<td>ARMv7-A</td>
<td>32</td>
<td>ARM Cortex-A5, ARM Cortex-A7, ARM Cortex-A6, ARM Cortex-A9, ARM Cortex-A12, ARM Cortex-A15, ARM Cortex-A17</td>
<td>Application</td>
<td></td>
</tr>
<tr>
<td>ARMv8-A</td>
<td>32</td>
<td>ARM Cortex-A32</td>
<td>Application</td>
<td></td>
</tr>
<tr>
<td>ARMv8-A</td>
<td>64/32</td>
<td>ARM Cortex-A35, ARM Cortex-A53, ARM Cortex-A57, ARM Cortex-A72, ARM Cortex-A73</td>
<td>Application</td>
<td></td>
</tr>
<tr>
<td>ARMv8.1-A</td>
<td>64/32</td>
<td>TBA</td>
<td>Application</td>
<td></td>
</tr>
<tr>
<td>ARMv8.2-A</td>
<td>64/32</td>
<td>ARM Cortex-A55, ARM Cortex-A75</td>
<td>Application</td>
<td></td>
</tr>
<tr>
<td>ARMv8.3-A</td>
<td>64/32</td>
<td>TBA</td>
<td>Application</td>
<td></td>
</tr>
</tbody>
</table>

ARM Family and Architecture
ARM FAMILY TREE

- **Cortex-A9**
  - High performance
  - 32-bit CPU with enterprise class feature set

- **Cortex-A5**
  - ARMv7-A
  - Smallest and lowest power CPU

- **Cortex-R4**
  - Real-time standard

- **Cortex-M0**
  - Lowest cost
  - Lowest power
  - Highest energy efficiency

- **Cortex-A15**
  - ARMv7-A
  - High performance
  - 32-bit CPU with enterprise class feature set

- **Cortex-A17**
  - ARMv7-A
  - High performance
  - 32-bit CPU with lower power and smaller area

- **Cortex-A57**
  - ARMv8-A
  - Highest performance
  - 64/32-bit CPU

- **Cortex-A53**
  - ARMv8-A
  - High efficiency
  - 64/32-bit CPU

- **Cortex-R5**
  - Functional safety

- **Cortex-R7**
  - High performance
  - 4G modem and storage

- **Cortex-M0**
  - Performance efficiency

- **Cortex-M3**
  - Mainstream Control & DSP

- **Cortex-M4**
  - Mainstream Control & DSP

- **Cortex-M7**
  - Maximum Performance
  - Control & DSP

- **High Performance**
- **High Efficiency**
- **Real-time**
- **Control**
ARM Cortex Processors

- **ARM Cortex-A family:**
  - Applications processors
  - Support OS and high-performance applications
  - Such as Smartphones, Smart TV

- **ARM Cortex-R family:**
  - Real-time processors with high performance and high reliability
  - Support real-time processing and mission-critical control

- **ARM Cortex-M family:**
  - Microcontroller
  - Cost-sensitive, support SoC
• Cortex-M is a great trade-off between performance, cost, efficiency; used for IoT, various applications.
• Has on-chip peripherals
• Core is licensed by ARM
CORTEX-M: CORE + Peripherals

• Core
  • Memory
    • FLASH: Non-Volatile / Instruction memory
    • SRAM/DRAM: Volatile / data memory
  • Processor
    • ALU
    • Processor Control Unit (CPU)
  • Registers
    • Special Purpose Registers
    • General Purpose Registers
• Buses
  • Data Bus
  • Instruction Bus
  • Bus bridge to connect diff. buses
  • Advanced High-performance Bus (AHB)
  • Advanced Peripheral Bus (APB)
• GPIO

• Peripherals
  • ADC
  • LCD Controller
  • SPI
  • I2C
  • Etc.
Core Architecture

Von-Neumann

Instructions and data are stored in the same memory.

- Simple and inexpensive
- Access to data or instruction, one at a time

Harvard

Data and instructions are stored into separate memories.

- Faster
- More energy efficient
- Different bus sizes
Core Architecture

Von-Neumann

Instructions and data are stored in the same memory.

Harvard

Data and instructions are stored into separate memories.
ARM Simplified Block Diagram
System on Chip (SoC)

http://www.microdigitaled.com/ARM/ASM_ARM/PowerPoints/ARM_ASM_ppts.htm
ARM Cortex-M4 Organization (STM32L4)

Note that the Kit we are using Has an **STM32F401**
Memory

- Memory is arranged as a series of “locations”
  - Each location has a unique “address”
  - Each location holds a byte (*byte-addressable*)
  - e.g. the memory location at address \(0x080001B0\) contains the byte value \(0x70\), i.e., 112

- The number of locations in memory is limited
  - e.g. 4 GB of RAM
  - 1 Gigabyte (GB) = \(2^{30}\) bytes
  - \(2^{32}\) locations \(\rightarrow\) 4,294,967,296 locations!

- Values stored at each location can represent either program data or program instructions
  - e.g. the value \(0x70\) might be the code used to tell the processor to add two values together
Memory Mapping

• Answer the following questions:
  • What is the size of the EEPROM?
  • What is the size of the Flash?
  • Which Memory portion is non-volatile?
  • What does SRAM generally used for in an ARM core processor?
  • Where is 0x743 address?
  • Where is 0x1000AB address?
ARM Register and ALU

16 Processor Registers
13 for general purpose
3 for specific purpose
Processor Registers

- Fastest way to read and write
- Registers are within the processor chip
- A register stores 32-bit value
- Cortex M (STM32L) has
  - **R0-R12**: 13 general-purpose registers
  - **R13**: Stack pointer (Shadow of MSP or PSP)
  - **R14**: Link register (LR)
  - **R15**: Program counter (PC)
  - Special registers (xPSR, BASEPRI, PRIMASK, etc.) - more later

32 bits

Low Registers

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>R0</td>
<td>R1</td>
<td>R2</td>
</tr>
<tr>
<td>R3</td>
<td>R4</td>
<td>R5</td>
</tr>
<tr>
<td>R6</td>
<td>R7</td>
<td>R8</td>
</tr>
<tr>
<td>R9</td>
<td>R10</td>
<td>R11</td>
</tr>
<tr>
<td>R12</td>
<td>R13(SP)</td>
<td>R14(LR)</td>
</tr>
<tr>
<td>R15(PC)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

General Purpose Register

32 bits

High Registers

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>R13(MSP)</td>
<td>R13(PSP)</td>
</tr>
</tbody>
</table>

Special Purpose Register
Program Execution

- **Program Counter (PC)** is a register that holds the memory address of the next instruction to be fetched from the memory.

<table>
<thead>
<tr>
<th>Memory Address</th>
<th>Memory Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x080001B0</td>
<td>4770</td>
</tr>
<tr>
<td>0x080001B2</td>
<td>2000</td>
</tr>
<tr>
<td>0x080001B4</td>
<td>188B</td>
</tr>
<tr>
<td>0x080001AE</td>
<td>2201</td>
</tr>
<tr>
<td>0x080001AC</td>
<td>2100</td>
</tr>
</tbody>
</table>

**PC = 0x080001B0**

Instruction = 188B or 2000188B or 8B180020
Three-state pipeline: Fetch, Decode, Execution

- **Pipelining** allows hardware resources to be fully utilized
- One 32-bit instruction or **two 16-bit** instructions can be fetched.

1. Fetch instruction at PC address
2. Decode the instruction
3. Execute the instruction

Pipeline of 32-bit instructions
Loading Code and Data into Memory

Dissection of a C Program: Copying an Array
ARM Register and ALU
Machine codes are stored in memory
Fetch Instruction: pc = 0x08001AC
Decode Instruction: 2100 = MOVS r1, #0x00

Registers
r15 0x080001AC
r14
r13
r12
r11
r10
r9
r8
r7
r6
r5
r4
r3
r2
r1
r0

CPU

ALU

Memory

Data
Address

0xFFFFFFFF
4770 0x080001B4
2000 0x080001B2
188B 0x080001B0
2201 0x080001AE
2100 0x080001AC
0x00000000
Execute Instruction:
MOVS r1, #0x00
Fetch Next Instruction: \( pc = pc + 2 \)

Decode & Execute: \( 2201 = \text{MOV} \text{S r2, #0x01} \)
Fetch Next Instruction: pc = pc + 2
Decode & Execute: 188B = ADDS r3, r1, r2
Fetch Next Instruction: \( \text{pc} = \text{pc} + 2 \)
Decode & Execute: \( 2000 = \text{MOVS r0, \#0x00} \)
ARM Applications....
iPhone 5 Teardown

The A6 processor is the first Apple System-on-Chip (SoC) to use a custom design, based off the ARMv7 instruction set.

http://www.ifixit.com
The A8 processor is the first 64-bit ARM based SoC. It supports ARM A64, A32, and T32 instruction set.
iPhone 7 Teardown

A10 processor:
• 64-bit system on chip (SoC)
• ARMv8-A core
Apple Watch

• Apple S1 Processor
  • 32-bit ARMv7-A compatible
  • # of Cores: 1
  • CMOS Technology: 28 nm
  • L1 cache 32 KB data
  • L2 cache 256 KB
  • GPU PowerVR SGX543
Kindle HD Fire

Texas Instruments OMAP 4460 dual-core processor

http://www.ifixit.com
Fitbit Flex Teardown

STMicroelectronics 32L151C6 Ultra Low Power ARM Cortex M3 Microcontroller

Nordic Semiconductor nRF8001 Bluetooth Low Energy Connectivity IC

www.ifixit.com
Samsung Galaxy Gear

- STMicroelectronics STM32F401B ARM Cortex M4 MCU with 128KB Flash

source: ifixit.com
Pebble Smartwatch

- STMicroelectronics STM32F205RE ARM Cortex-M3 MCU, with a maximum speed of 120 MHz

source: ifixit.com
Oculus VR

- Facebook’s $2 Billion Acquisition Of Oculus in 2014
- ST Microelectronics STM32F072VB **ARM Cortex-M0** 32-bit RISC Core Microcontroller

*source: ifixit.com*
HTC Vive

STMicroelectronics
32F072R8 **ARM Cortex-M0**
Microcontroller

*source: ifixit.com*
Nest Learning Thermostat

- ST Microelectronics STM32L151VB ultra-low-power 32 MHz ARM Cortex-M3 MCU
Samsung Gear Fit Fitness Tracker

- STMicroelectronics STM32F439ZI 180 MHz, 32 bit ARM Cortex-M4 CPU

(source: ifixit.com)
A Little About STM32

- **STM32** is a family of 32-bit microcontroller integrated circuits by **STMicroelectronics**
- The STM32 chips are grouped into related series that are based around the same 32-bit ARM processor core, such as the Cortex-M7F, Cortex-M4F, Cortex-M3, Cortex-M0+, or Cortex-M0.
- Internally, each microcontroller consists of the processor core, static RAM memory, flash memory, debugging interface, and various peripherals.

https://en.wikipedia.org/wiki/STM32
<table>
<thead>
<tr>
<th>Product lines</th>
<th>FPU (MHz)</th>
<th>Flash (Kbytes)</th>
<th>RAM (KB)</th>
<th>Ethernet I/F</th>
<th>Camera I/F</th>
<th>SDRAM I/F</th>
<th>CAN I/F</th>
<th>CAN3 I/F</th>
<th>SPI</th>
<th>SPDIF RX</th>
<th>TFT LCD controller</th>
<th>NMI CSI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access lines</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STM32F461</td>
<td>84</td>
<td>128 to 512</td>
<td>Up to 96</td>
<td>Down to 128</td>
<td>Down to 10</td>
<td>Down to 3x3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STM32F410</td>
<td>100</td>
<td>84 to 128</td>
<td>32</td>
<td>Down to 80</td>
<td>Down to 6</td>
<td>Down to 2.55x to 2.57x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STM32F411</td>
<td>100</td>
<td>256 to 512</td>
<td>128</td>
<td>Down to 103</td>
<td>Down to 12</td>
<td>Down to 3.03x to 3.22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STM32F412</td>
<td>100</td>
<td>512 to 1024</td>
<td>256</td>
<td>Down to 112</td>
<td>Down to 18</td>
<td>Down to 3.65x to 3.661</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STM32F413</td>
<td>100</td>
<td>1024 to 1536</td>
<td>320</td>
<td>Down to 115</td>
<td>Down to 18</td>
<td>Down to 3.95x to 4.039</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes:
1. I / V min is specific package
2. The same devices are also found with embedded hardware crypto/hashing
3. Serial Audio Interface
4. Link Power Management
STM32 Nucleo Family

[Diagram showing the STM32 Nucleo Family with various models and their flash size categories]
<table>
<thead>
<tr>
<th></th>
<th>NUCLEO F401</th>
<th>Arduino UNO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microcontroller</td>
<td>STM32F401 32-bit</td>
<td>ATMEGA 328 8-bit</td>
</tr>
<tr>
<td>Family</td>
<td>ARM Cortex-M4</td>
<td>AVR</td>
</tr>
<tr>
<td>Clock Frequency</td>
<td>84 Mhz</td>
<td>16 Mhz</td>
</tr>
<tr>
<td>Flash memory</td>
<td>512 Kb</td>
<td>32 Kb</td>
</tr>
<tr>
<td>SRAM</td>
<td>96 Kb</td>
<td>2 K</td>
</tr>
<tr>
<td>EEPROM memory</td>
<td>-</td>
<td>1 Kb</td>
</tr>
<tr>
<td>PWM</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>Analog inputs</td>
<td>16</td>
<td>6</td>
</tr>
<tr>
<td>Digital Pin</td>
<td>47</td>
<td>14</td>
</tr>
<tr>
<td>I2C modules</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>USART modules</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>SPI modules</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Timer</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>Floating point UNIT</td>
<td>One</td>
<td>No</td>
</tr>
<tr>
<td>Maximum voltage supported</td>
<td>5V</td>
<td>5V</td>
</tr>
<tr>
<td>USB OTG</td>
<td>One</td>
<td>No</td>
</tr>
<tr>
<td>Dimensions</td>
<td>68mmx80mm</td>
<td>53mmx68mm</td>
</tr>
<tr>
<td>Price</td>
<td>€ 10</td>
<td>€ 20</td>
</tr>
</tbody>
</table>
Other ARM Chips

<table>
<thead>
<tr>
<th>Company</th>
<th>Device</th>
<th>Flash (K Bytes)</th>
<th>RAM (K Bytes)</th>
<th>I/O Pins</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atmel</td>
<td>AT91SAM7X512</td>
<td>512</td>
<td>128</td>
<td>62</td>
</tr>
<tr>
<td>NXP</td>
<td>LPC2367</td>
<td>512</td>
<td>58</td>
<td>70</td>
</tr>
<tr>
<td>ST</td>
<td>STR750FV2</td>
<td>256</td>
<td>16</td>
<td>72</td>
</tr>
<tr>
<td>TI</td>
<td>TMS470R1A256</td>
<td>256</td>
<td>12</td>
<td>49</td>
</tr>
<tr>
<td>Freescale</td>
<td>Mk10DX256VML7</td>
<td>256</td>
<td>64</td>
<td>74</td>
</tr>
</tbody>
</table>