The ARM (Advanced RISC Machine) is a 32-bit microcontroller created by a consortium of companies and manufactured in many different versions. It is widely used in modems, cell phones, personal audio, pagers, cameras, and many more embedded high end applications.

ARM is produced by many different companies and each adds its own variations. We will use the ARM Cortex M4 processor produced by STMicroelectronics (STM). The processor is an STM32F4VG07 mounted on a low cost Discovery board.

RISC – Reduced Instruction Set Computer.

RISC/CISC controversy and history

• In the 1970s as IC manufacturers were able to put more and more on a single chip, computer architects needed to decide what to do with the extra space.
• The prevailing idea was to create more complex instructions which are then executed in microcode. This reduces the use of program memory to fetch instructions and should run faster. This was called CISC for Complex Instruction Set Computer.
• A counter idea, created by IBM, was to use the extra space for things like more registers, cache memory, and numerical processing ability. This solution kept the instruction set very simple so that it took up little extra circuitry. This was called RISC and began about 1980.
• Many new RISC machines were built such as the SUN Sparc. The Intel x86 line represents an architecture that leans toward the CISC side.
• By the mid-1990s it was clear that the ideal machine was neither RISC nor CISC but something in between. Today most machines have features of both.

Generic RISC features:
• A large number of general purpose registers along with compiler technology to optimize register usage.
• A limited and simple instruction set
• An emphasis on optimizing the instruction pipeline
• Load and store architecture with simple addressing modes

Today, most CPU’s have some RISC-like features but pure RISC machines are uncommon. The ARM processor has its origins in 1983 from Acorn Computers Ltd. In 1990 Acorn spun off a new company called Advanced RISC Machines Ltd. which licenses its ARM core to other companies for manufacture. Many other companies now make versions of the ARM processor including Intel, Atmel, Texas Instruments,
FreeScale, and many others. The Cortex M-series is one of the more successful versions of the ARM processor.

**Feature Summary ARM Cortex M4 core**

- 32-bit RISC processor (32-bit data & address bus)
- Big and Little Endian operating modes
- High performance RISC
- Low power consumption
- Fully static operation
- Fast interrupt response for real-time applications
- Excellent high-level language support
- Simple but powerful instruction set
- Supports 8, 16, 32, and 64-bit data types (does not support bit data type)
- Support for floating point and DSP oriented operations

**The instruction set has just 56 instructions in 11 functional groups**

- Data move
- Memory access
- Stack operators
- Arithmetic operators
- Logical operators
- Shift and rotate instructions
- Extend instruction and reverse order instructions
- Flow control (branch and calls)
- Memory barrier instructions
- Exceptions related instructions
- Miscellaneous

Note that there is no division instruction in the assembly code – this has to be done piecemeal in software. Likewise, there are no I/O instructions – all I/O is memory mapped.

**Thumb instructions**

The ARM Cortex M4 is a 32-bit computer so one would expect that all of the instructions would be 32-bits long. But a 32-bit instruction is largely wasted on simple operations that make use of the UART, I²C, and bit manipulation so the ARM series of processors also uses *Thumb* instructions. Thumb instructions are 16-bit versions of 32-bit instructions that save considerable program space. A 32-bit ARM *ADD* instruction might look like this: ADD R1, R2, R3. This instruction adds R2 to R3 and puts the result in R1. The thumb version of this instruction is just ADD R1, R2 which add R1 to R2 and puts the result in R3. The result is a slightly less powerful instruction (in some cases) in half of the space. All ARM processors support the Thumb mode.

In some cases, 32-bit instructions are necessary so a third set of instructions called *Thumb2* instructions have been created. Thumb2 instructions are mostly 16-bit instructions with a few 32-bit instructions. The ARM Cortex M4 uses the Thumb2 instruction set. Of the 56 instructions, just 6 are 32-bit. All of the others are 16-bit.
**Data Types**
This processor supports bytes, half-words (16-bits), and full words (32-bits). Bits and nibbles are not supported but can be done with bit manipulation and the logical instructions. All data must begin on a 32-bit word boundary in memory. Instruction fetches are half-word aligned since they are mostly 16-bit.

![ARM Cortex M3/M4 Processor block diagram](image)

**Programmer's Model**

![Registors Set for ARM Cortex M4](image)
R0-R12 are called general purpose registers. For most of the thumb instruction set only the low registers, R0-R7 are available. The MOV instructions have access to all of the registers.

Register R13 is the stack pointer. The lower two bits of this register is always zero since it accesses the stack on 32-bit word boundaries.

Register R14 is called the link register. For quick access to subroutines the link register is effectively a 1 level stack. The return address of a call instruction is saved in this register. If a subroutine itself makes another call its return address then goes on the stack. One level subroutine calls are thus much faster.

Register R15 is the program counter. It is 32-bits long and there is a linear address space from 0 to $2^{32}-1$.

The CPU flags are stored in the program status register APSR.

<table>
<thead>
<tr>
<th>Table 4.2 Bit Fields in Program Status Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit</td>
</tr>
<tr>
<td>N</td>
</tr>
<tr>
<td>Z</td>
</tr>
<tr>
<td>C</td>
</tr>
<tr>
<td>V</td>
</tr>
<tr>
<td>Q</td>
</tr>
<tr>
<td>GE[3:0]</td>
</tr>
<tr>
<td>ICT/T</td>
</tr>
<tr>
<td>T</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Exception Number</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>Negative flag</td>
</tr>
<tr>
<td>Z</td>
<td>Zero flag</td>
</tr>
<tr>
<td>C</td>
<td>Carry (or NOT borrow) flag</td>
</tr>
<tr>
<td>V</td>
<td>Overflow flag</td>
</tr>
<tr>
<td>Q</td>
<td>Sticky saturation flag (not available in ARMv6-M)</td>
</tr>
<tr>
<td>GE[3:0]</td>
<td>Greater-Than or Equal flags for each byte lane (ARMv6-M only; not available in ARMv6-M or Cortex®-M3), Instruction status bit for conditional execution (not available in ARMv6-M).</td>
</tr>
<tr>
<td>ICT/T</td>
<td>Interrupt-Continuable Instruction (ICT) bits, IF-THEN TOT-THEN (not available in ARMv6-M).</td>
</tr>
<tr>
<td>T</td>
<td>Thumb state, always 1; trying to clear this bit will cause a fault exception. Indicates which exception the processor is handling.</td>
</tr>
</tbody>
</table>

**Flags for the M4 Processor**
Memory mapping

Memory Map for Cortex M4 Processor

**Code Region (0x00000000—0x1FFFFFFF)**

The size of the code region is 512 MB. It is primarily used to store program code, including the exception vector table, which is a part of the program image. It can also be used for data memory (connection to RAM).

**SRAM Region (0x20000000—0x3FFFFFFF)**

The SRAM region is located in the first 512 MB of the memory map. It is primarily used to store data, including stack. It can also be used to store program code. For example, in some cases you might want to copy program code from slow external memory to the SRAM and execute it from there. Despite the name given to this region (it is called “SRAM”), the actual memory devices being used could be SRAM, SDRAM, or some other type.

**Peripheral Region (0x40000000—0x5FFFFFFF)**

The peripheral region also has the size of 512 MB. It is primarily used for peripherals and can also be used for data storage. However, program execution is not allowed in the peripheral region. The peripherals connected to this memory region can be either the AHB-Lite peripheral or APB peripherals (via a bus bridge).

**RAM Region (0x60000000—0x9FFFFFFF)**

The RAM region consists of two 512 MB blocks, which results in a total of 1 GB of space. Both 512 MB memory blocks are primarily used to stored data, and in most cases the RAM region can be used as a 1 GB continuous memory space. The RAM region can also be used for program code execution. The only differences between the two halves of the RAM region is the memory attributes, which might cause differences in cache behavior if a system-level cache (level-2 cache) is used. Memory attributes will be covered in more detail later in this chapter.

**Device Region (0xA0000000—0xDFFFFFFF)**

The external device region consists of two 512 MB memory blocks, which results in a total of 1 GB of space. Both 512 MB memory blocks are primarily used for peripherals...
**STM32F407VG from STMicroelectronics on Discovery Board**

**FEATURES**

Processor

- ARM Cortex-M4 processor, running at 168 MHz.

Interrupts

- ARM Cortex-M4 built-in Nested Vectored Interrupt Controller (NVIC).
- Non-Maskable Interrupt (NMI).

Memory:

- 1024 kB on-chip flash programming memory.
- 192 kB SRAM.

Programming

- In-System Programming (ISP) and In-Application Programming (IAP) via on-chip bootloader software. Flash is programmed with a hex file using Keil µVision 5.

Digital peripherals:

- 82-GPIO pins can be used as edge and level sensitive interrupt sources.
- High-current output driver (25 mA) on one pin in or out.
- GPIO is 5 volt tolerant.

Timers

- Twelve 16-bit and two 32-bit general purpose counter/timers

Analog peripherals:

- Three 12-bit ADCs and two 12-bit DACs

Serial interfaces:

- Four USARTs plus two UARTs
- Three SPI channels and two I'C channels
- Two CAN bus interfaces
Clock generation:
- Crystal oscillator at 8 MHz
- PLL allows CPU operation at 168 MHz

Single power supply (1.8 V to 3.6 V).

LQFP 100-pin package.

**Figure 13. STM32F40x LQFP100 pinout**

100-pin LQFP package for STM32F407VG