ARM Cortex-M Architecture

1. Overview

- ARM Cortex-M is a **32-bit RISC** microcontroller processor widely used in embedded systems.
- It uses the **Harvard architecture**: separate buses for instructions and data, so instructions can be fetched and data accessed simultaneously, improving execution speed.
- It gives the high performance of a 32-bit processor while keeping the compact code size of 8-bit and 16-bit microcontrollers.

2. Major Components

2.1 Memory and Buses

Component	Function
Flash ROM	Stores program instructions permanently; instructions are fetched through the ICode bus
ICode bus (instruction bus)	Transfers instructions from Flash ROM to the CPU; used only for instruction fetching
System bus	Transfers data between processor, RAM and I/O peripherals; supports read and write
DCode bus (debug bus)	Used for debugging operations, program testing and debugging
AHB (Advanced High-performance Bus)	Available in Cortex-M4; high-speed communication for peripherals such as USB
PPB (Private Peripheral Bus)	Connects internal peripherals directly to the processor; used by components like the NVIC

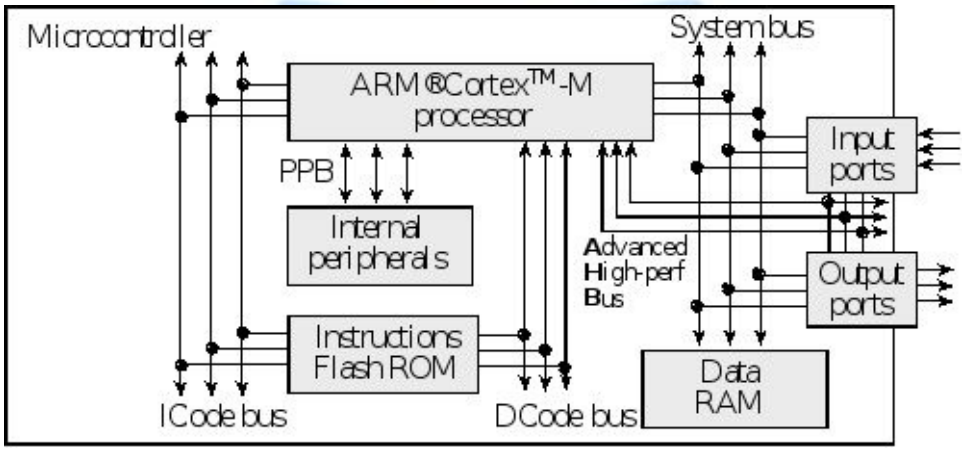

Figure 1.1 Harvard architecture of an ARM Cortex-M based microcontroller

2.2 Nested Vectored Interrupt Controller (NVIC)

- Manages all interrupts in the processor.
- Supports nested interrupts and priority-based interrupt handling.
- Reduces interrupt response time (low interrupt latency).

3. Registers in ARM Cortex-M

Registers are small, high-speed memory locations inside the processor used to store data, addresses and instructions temporarily.

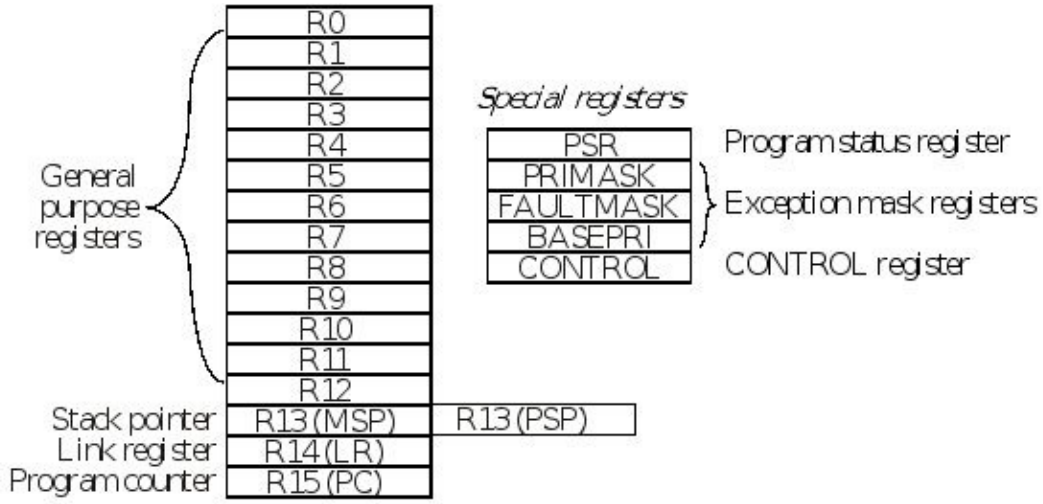

Figure 1.2 Register set of the Cortex-M processor

3.1 General Purpose Registers (R0–R12)

- 13 general-purpose registers, usable by the programmer during execution.
- Used to:
  - store data
  - store memory addresses
  - perform arithmetic and logical operations

3.2 Stack Pointer (R13 or SP)

- Always points to the **top of the stack**.
- The stack stores local variables, function parameters, return addresses and temporary data.

Stack pointer	Used by	Notes
Main Stack Pointer (MSP)	Operating system; interrupt handling	Mainly used in embedded systems; provides reliable operation
Process Stack Pointer (PSP)	User applications	Protects the OS if the user program crashes

Note: Only one stack pointer (MSP or PSP) is active at a time.

3.3 Link Register (R14 or LR)

- Stores the **return address** when a function is called; after the function finishes, the processor returns to this address.
- Also used during interrupts and exceptions.

3.4 Program Counter (R15 or PC)

- Stores the address of the **next instruction** to be executed.
- After fetching an instruction, the PC automatically increments by 2 or 4 bytes.

4. Procedure Call Standard (AAPCS)

ARM follows the ARM Architecture Procedure Call Standard (AAPCS) for function calls.

Register	Purpose
R0–R3	Function (input) arguments
R0	Return value

5. Program Status Registers

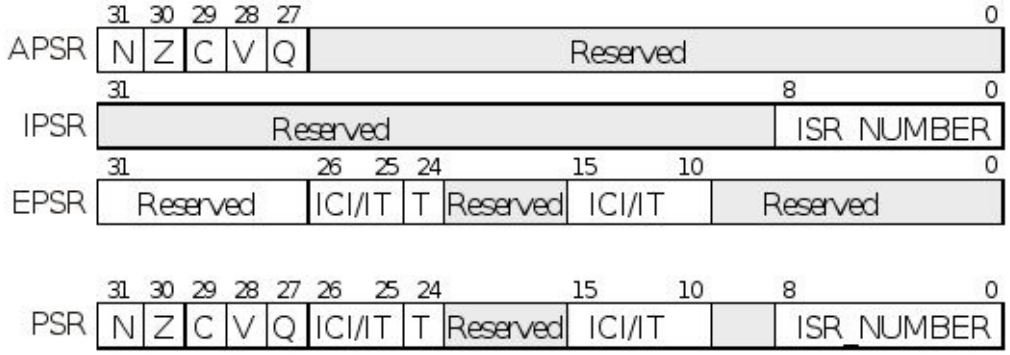

Figure 1.3 The program status register of the Cortex-M processor

5.1 Status Registers

Register	Function
APSR (Application PSR)	Stores the result of arithmetic and logical operations; contains the condition flags
IPSR (Interrupt PSR)	Indicates which interrupt is currently executing
EPSR (Execution PSR)	Stores execution state information; indicates the processor is executing Thumb instructions

The **PSR** is the combination of APSR, IPSR and EPSR.

5.2 Condition Flags in APSR

Flag	Meaning	Set when
N	Negative result	The result is negative
Z	Zero result	The result is zero
C	Carry generated	An unsigned carry/overflow occurs
V	Signed overflow	A signed overflow occurs
Q	Saturation occurred	A saturation operation has occurred

6. Special Registers

6.1 Thumb Bit (T) and ISR Number

- **T bit** is always 1: the processor executes Thumb instructions.
- **ISR number** indicates the interrupt currently being executed; 0 if no interrupt is active.

6.2 PRIMASK, FAULTMASK and BASEPRI

Register	Purpose	Bit value / behaviour
PRIMASK	Enable or disable interrupts	0 = interrupts enabled; 1 = most interrupts disabled
FAULTMASK	Enable or disable fault exceptions	0 = faults enabled; 1 = faults disabled
BASEPRI	Set the priority level of interrupts	Higher-priority interrupts allowed; lower-priority blocked

Note: The Non-Maskable Interrupt (NMI) cannot be disabled.

Example: if BASEPRI = 3, interrupts of priority 0, 1 and 2 are allowed; priority 3 and above are blocked until the current task finishes.

7. Operating Modes

The Cortex-M processor has two privilege levels and two execution modes.

7.1 Privilege Levels

Level	Access	Used by
Privileged	Full access to all processor resources: system timer, NVIC, system control registers	Operating system
Unprivileged	Limited access; cannot directly access system timer, interrupt controller or some system registers; improves security	User applications

7.2 CONTROL Register

The CONTROL register decides the processor privilege level and the stack pointer.

Bit	Name	Value 0	Value 1
Bit 0	Thread Mode Privilege Level (TPL)	Unprivileged mode	Privileged mode
Bit 1	Active Stack Pointer Selection (ASPSEL)	Main Stack Pointer (MSP)	Process Stack Pointer (PSP)

7.3 Stack Pointer Usage

- **MSP:** used by the OS and during interrupt handling; provides reliable system operation.
- **PSP:** used by user applications; keeps user programs separate from the OS and prevents OS failure if the user program crashes.

Note: In high-reliability systems, user programs use PSP and the operating system uses MSP.

7.4 Execution Modes

Feature	Thread mode	Handler mode
Also called	Foreground mode	Background mode
Executes	Main program / normal application code	Interrupt Service Routines (ISRs)
Entered	After reset (ISR_NUMBER = 0)	Automatically when an interrupt occurs
Stack pointer	MSP or PSP	Always MSP
Exit	–	Returns to Thread mode after the interrupt