# Embedded Systems – Fundamental Concepts

Date: 16–12–2025

Reference Books & Manuals:

- Mazidi – *AVR Microcontroller*
- Joseph Yiu – *The Definitive Guide to ARM Cortex-M3/M4*
- STM32 Reference Manual
- STM32F407G-DISC1 User Manual

---

## 1. Microprocessor (μP) vs Microcontroller (μC)

- A microprocessor is a general-purpose computing device that requires external memory and - peripherals.
- A microcontroller is designed for embedded systems and integrates CPU, memory, and peripherals on a single chip.
- Microprocessors provide flexibility but consume more power and space.
- Microcontrollers are compact, low-power, and cost-effective.

| **Feature** | **Microprocessor (μP)** | **Microcontroller (μC)** |
|---|---|---|
| Purpose | General-purpose computing | Dedicated embedded applications |
| CPU | ALU, registers | ALU, registers |
| Memory | External RAM & ROM | On-chip RAM & Flash |
| Peripherals | External | On-chip |
| Clock Speed | High | Low to medium |
| Power Consumption | High | Low |
| Cost | High | Low |
| Size | Large | Compact |
| Examples | x86, Cortex-A | AVR, PIC, Cortex-M |

---

`

`

`

`

`

`

## 2. Von Neumann vs Harvard Architecture

- The difference between these architectures lies in how memory is accessed.
- Von Neumann uses a single bus for both instruction and data.
- Harvard uses separate buses, enabling parallel access.
- This makes Harvard architecture faster.

| Feature | Von Neumann | Harvard |
|---|---|---|
| Address & Data Bus | Common | Separate |
| Instruction & Data Fetch | Not simultaneous | Simultaneous |
| Execution Speed | Slower | Faster |
| Cache | Unified | Separate I & D |
| Example | x86 | AVR, ARM (Modified Harvard) |

---

## 3. RISC vs CISC Architecture

- RISC and CISC differ in instruction complexity and execution style.
- CISC focuses on complex instructions to simplify compilers.
- RISC focuses on simple instructions for faster execution.
- Most embedded systems use RISC architecture.

| Feature | RISC | CISC |
|---|---|---|
| Instruction Set | Small | Large |
| Instruction Type | Micro-instructions | Macro + Micro |
| Instruction Length | Fixed | Variable |
| Execution Cycles | Mostly single cycle | Variable cycles |
| Registers | More | Fewer |
| Memory Access | Load/Store | Direct memory access |
| Compiler Design | Complex | Easier |
| Execution Method | Pipeline | Instruction Queue |
| Examples | AVR, ARM, PIC | 8086, 8051, x86 |

---

`

`

`

`

`

# 4. Memory-Mapped I/O vs I/O-Mapped I/O

- I/O devices must be accessed by the CPU using specific addressing methods.
- Memory-mapped I/O treats peripherals as memory locations.
- I/O-mapped I/O uses a separate address space.
- ARM processors mainly use memory-mapped I/O.

| Feature | Memory-Mapped I/O | I/O-Mapped I/O |
|---|---|---|
| Address Space | Same as memory | Separate |
| Instructions | MOV, LDR, STR | IN, OUT |
| Hardware Complexity | Simple | Complex |
| Example | ARM | x86 |

---

# 5. Instruction Pipeline vs Instruction Queue

- Instruction execution can be optimized using pipelines or queues.
- An instruction queue only stores instructions before execution.
- An instruction pipeline processes multiple instructions simultaneously.
- Pipelines improve instruction-level parallelism.

| Feature | Instruction Pipeline | Instruction Queue |
|---|---|---|
| Processing | Yes | No |
| Parallel Execution | Yes | No |
| Architecture | RISC | CISC |
| Examples | AVR, ARM, PIC | x86 |

---

# 6. Instruction Pipeline Stages

- Pipelining divides instruction execution into stages.
- Each stage works on a different instruction simultaneously.
- This improves overall CPU throughput.

| Pipeline Type | Stages | Examples |
|---|---|---|
| 2-Stage | Fetch → Execute | AVR, PIC |
| 3-Stage | Fetch → Decode → Execute | ARM7, Cortex-M3/M4 |

---

# 7. Pipeline Hazards

- Pipeline hazards occur when normal instruction flow is disturbed.
- They reduce performance by stalling or flushing the pipeline.
- Hazards are mainly classified into control, data, and structural hazards.

**Control Hazards**

- Due to jump or conditional jump, instructions already fetched into pipeline are of no use and hence
- pipeline need to be cleared and re-filled from new (jumped) address. Obviously next instruction is not effectively completed in single CPU cycle.
- To overcome this problem, CPU can use "branch prediction" i.e. CPU predicts whether condition in current jump instruction will be true or not. Depending on that it fetch the next instructions.
  Some advanced CPUs also use "brach speculation". In this next instructions are not only fetched but also processed and their result is stored temporarily. If prediction is correct, the result is utilized (for faster execution); otherwise the result is discarded.

**Data Hazards**

- For some instructions more CPU cycles are needed for the execution eg. mul. If next instruction execution depends on result of previous instruction, then pipeline is stalled (paused) until current instruction is completed. Due to this effective number of CPU cycles for execution of instruction will increase.

**Structural Hazards**

- Due to CPU design restrictions some instructions cannot be completed immediately. e.g. If CPU is designed to write only one result into regr and two results were produced in an instruction. In this case, pipeline is stalled to complete the current instruction

| Hazard Type | Cause | Effect |
|---|---|---|
| Control Hazard | Branch / Jump | Pipeline flush |
| Data Hazard | Data dependency | Pipeline stall |
| Structural Hazard | Hardware limitation | Pipeline stall |

---

# 8. AMBA – Advanced Microcontroller Bus Architecture

- AMBA is a standard bus architecture defined by ARM.
- It enables efficient communication between CPU and peripherals.
- Different AMBA buses are used based on performance needs.

| Bus Type | Speed | Complexity | Used For |
|---|---|---|---|
| AHB | High | Complex | CPU, RAM, Flash, DMA |
| APB | Low | Simple | UART, SPI, $I^2C$, GPIO |
| AXI | Very High | Very Complex | Cortex-A SoCs |

---

# Summary

This document explains:

- μP vs μC
- Memory architectures
- RISC vs CISC
- I/O mapping methods
- Instruction execution techniques
- Pipeline hazards
- AMBA bus architecture

**Embedded Systems and ARM Cortex-M beginners**.