# Org: Introduction to Microcomputer and Embedded Systems

Alex Chi

*Update: 2020-03-03*

## Contents



## 1 Von Neumann Architecture

Based on **Turing Machine**, aka. **Princeton Architecture**

1. Five components partitioning Input, Output, Memory, CPU (ALU, Control Unit)
   [Figure]
2. Three key concepts
   (a). Instruction and Data stored in single R/W memory
   (b). Contents of memory addressable by location, regardless of data type
   (c). Sequential execution
3. Harvard Architecture
   Split memory to read-only instruction memory and R/W data memory. Example: Arduino

4. Stored Program Concept

   Memory <–Address Bus–> CPU <–Control Bus / Data Bus–> I/O Device
5. How does CPU work
   - **Clock Generator** generates pulse.
   - **Instruction Pointer** points to next instruction. (PC in x86 or other arch)
   - **Instruction Register** stores currently executing instruction
   - **Instruction Decoder** decodes instruction
   - **ALU** does calculation.
   - **General Purpose Register** stores intermediate data.

   (a). CPU loads instruction from system memory (addressable by 8-bit or 1-byte)
      - Write instruction address to system address bus
      - Set system control bus to read from memory
      - Read instruction from memory from system data bus

   (b). CPU decodes instruction (64 -> `INC AX`)

   (c). ALU reads AX, plus 1 and write back

# 2 Microprocessor

It is

- CPU consisting of ALU, CU and Registers
- no RAM, ROM or I/O on chip
- e.g. Intel family

1. Moore's Law

   Transistor on IC doubles every two years.
2. Why I/O device uses interface

   Speed and information type varies, I/O devices are generally slower.
3. I/O devices are not included in MC

   They're part of Microcomputer System.
4. Microcomputer System

   Microcomputer, Peripheral I/O Devices, Software

# 3 Hardware

## 3.1 CPU

1. ALU
   add, subtract, multiply, divide, and, or, not
2. CU
   works under instructions, contains ID and PC.
3. Instruction Set
   All recognizable instructions by the ID. Classified as CISC and RISC.
4. CISC
   variable length, time, more instruction formats, upward compatible. Example: 80x86
5. RISC
   fixed-time, fixed-time, easy to pipeline, fewer formats. Example: PowerPC, MISC, ARM, RISC-V

## 3.2 Memory

```
CPU <--> Cache <--> Internal Memory <--> External Memory
```

1. Memory Hierarchy
   Cheaper is slower. Register > L1 L2 L3 Cache > RAM or ROM > Disk
2. Bit
   Binary digit, 0 or 1.
3. Byte
   8 bits (generally). Smallest addressable unit.
4. Nibble
   4 bits.
5. Word
   Maximum number of bits a CPU can process. Related to data bus.
6. Double Word
   2 * Word
7. K, M, G, T, P
   $10^{10}$, $10^{20}$, ...
8. Memory Module Organization
   e.g. bit extension: 4*8M 8-bit module $\Longrightarrow$ 8M 32-bit memory ; word extension: 4*8M 8-bit module $\Longrightarrow$ 16M 16-bit memory

## 3.3 Bus

1. Bus

   communication pathway connecting two or more device
2. System bus

   connects CPU, memory, I/O
3. Arbitration
   - Distributed protocols. e.g. CSMA/CD in Ethernet. In CS: Raft, Kademlia, Chord
   - Centralized scheme. Master/Slave. Slave passively waits for order.
4. Type

   Dedicated (physical dedication) / Multiplexed (e.g. time multiplexing)
5. Timing

   Synchronous (share global clock), Asynchronous (have own clock)
6. Examples

   (a). Single-bus structure

   AB, DB, CB connects all device.
   - pro: simple
   - con: poor performance in terms of throughput

   (b). Dual-bus structure

   Memory Bus, I/O Bus
   - pro: efficient when transferring data
   - con: information between memory and I/O device requires CPU intervention

   (c). Memory-central Dual-bus

   Memory connects to both I/O bus and memory bus

   I. Data Bus
   - bidirectional
   - width of data bus
     - the same as CPU register
     - maximum data being processed in one cycle
     - define word of computer

   II. Address Bus
   - one direction (cpu -> memory, cpu -> I/O)
   - if width is n, then $2^n$ addresses can be addressed
   - memory capacity = maximum addressable address * word size

   III. Control Bus
   - control each module and the use of data
   - command and timing information, e.g. memory r/w, IO r/w, bus request/grant

- two sets of uni-directional control signal, command: CPU -> memory, state: memory -> CPU
- I/O is from the processor's perspective

IV. Addressing Scheme

- Memory-mapped I/O (MMIO) e.g. ARM, RISC-V
  - One single address space for both I/O and memory
  - status and data registers are treated memory locations
  - use the same instruction as memory access
  - `CPU -> Address Decoder -> I/O or Memory`
- Isolated I/O
  - 2 address space for memory and I/O
  - different instruction for accessing
  - Example: 8086 assembly use `MOV` for memory and use `IN` `OUT` for I/O
  - Design
    - Dedicated address lines: output address directly from CPU pins
    - Add a M/IO pin to indicate $M/\overline{IO}$, where IO is low