Lecture 01:

Introduction to Computer Organization

(Don’t worry if you cannot fully understand the content. Keep your questions and you will find the answers in later lectures)

Jingwen Leng

Outline

- Why/what you will learn from this course
- Computer organization
- CPU
- Memory
- Bus
- I/O
- What is an embedded system?

What is Computer Science

- Computer science is the study of mathematical algorithms and processes that interact with data and that can be represented as data in the form of programs. It enables the use of algorithms to manipulate, store, and communicate digital information.
- A computer scientist studies the theory of computation and the practice of designing software systems.
  - Source: Wikipedia

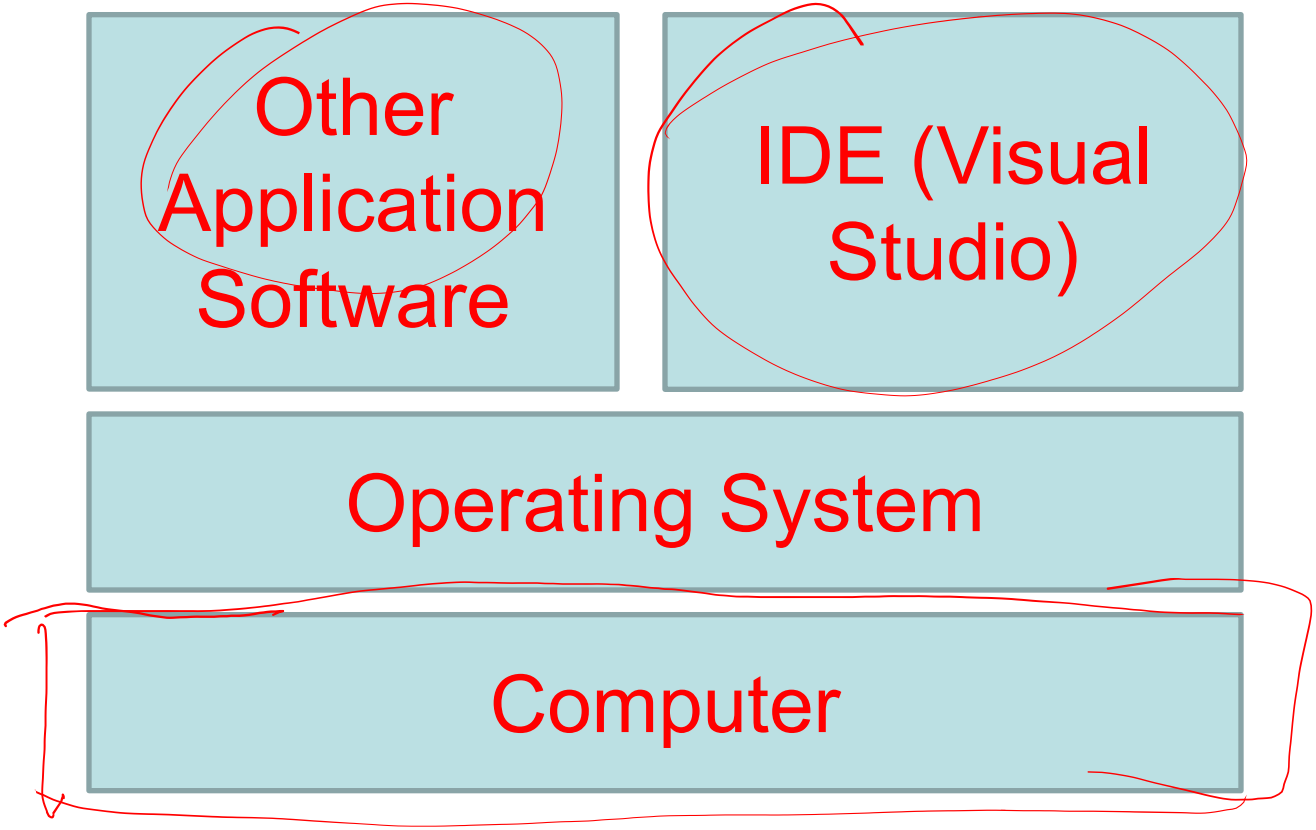
Abstractions in Computer Science

- "All problems in computer science can be solved by another level of indirection."
  - By David Wheeler
- Computer has many, many abstractions!
  - A Turing machine is a mathematical model of computation that defines an abstract machine, which manipulates symbols on a strip of tape according to a table of rules

Examples of Abstractions

- Abstraction in C programming language
  - Variables, data types, function call
- Abstraction in operating systems
  - Files: text files, executable files, doc/ppt files
  - Disk: different drives/directories
- Abstraction in WWW
  - Website, webpage

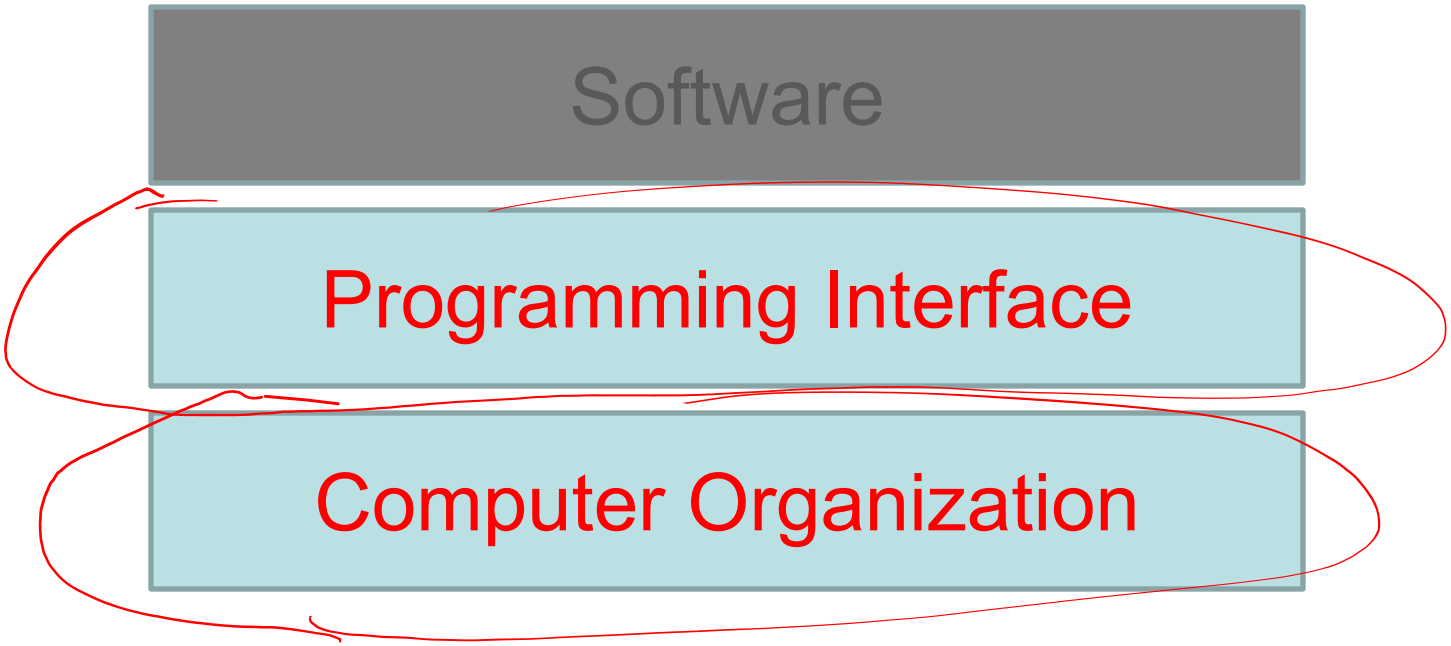
Computer Abstractions



Why Do We Learn This Course

- Understand the computer organization
- Understand how different computer components interact
- Understand the programming interface of computer
  - Different levels: assembly vs C

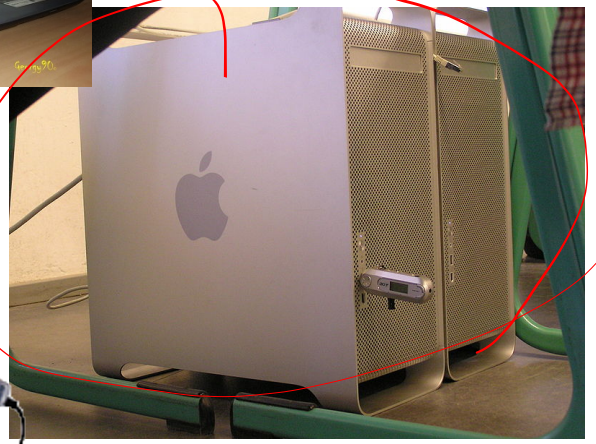
What this Course Teach

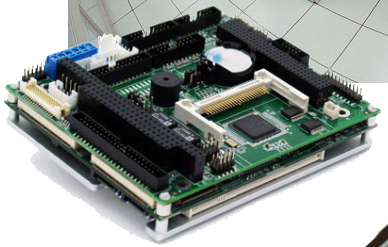


Outline

- Why/what you will learn from this course
- Computer organization
- CPU
- Memory
- Bus
- I/O
- What is an embedded system?

What is a computer?

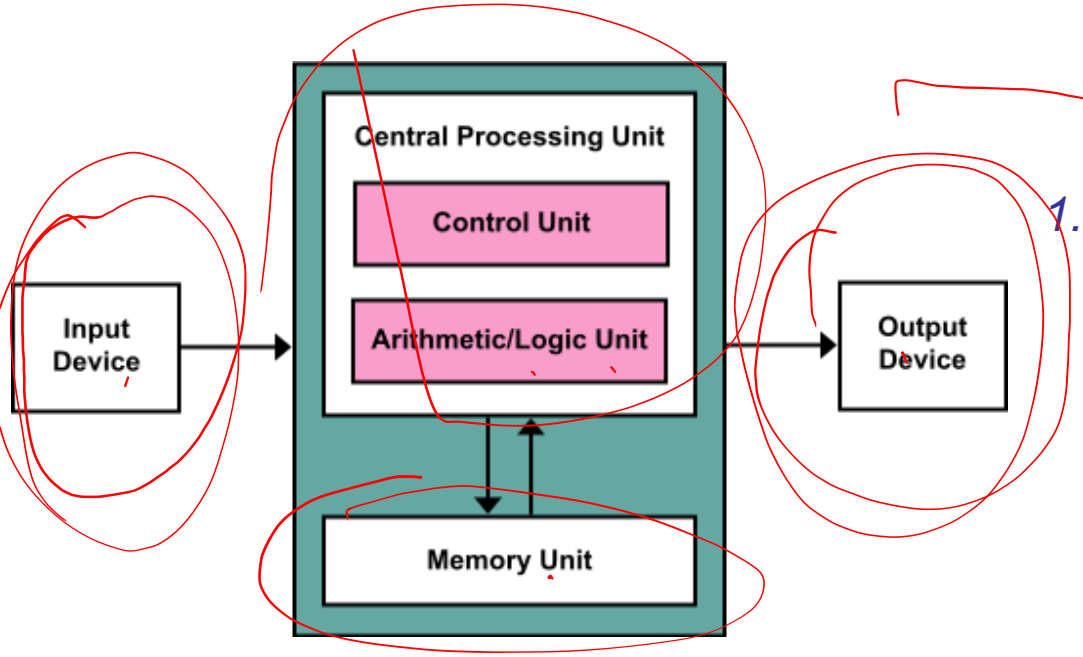




Computer Organization

Von Neumann Architecture

冯·诺伊曼架构



1.Five components partitioning

- Input
- Output
- Memory
- ALU
- Control unit

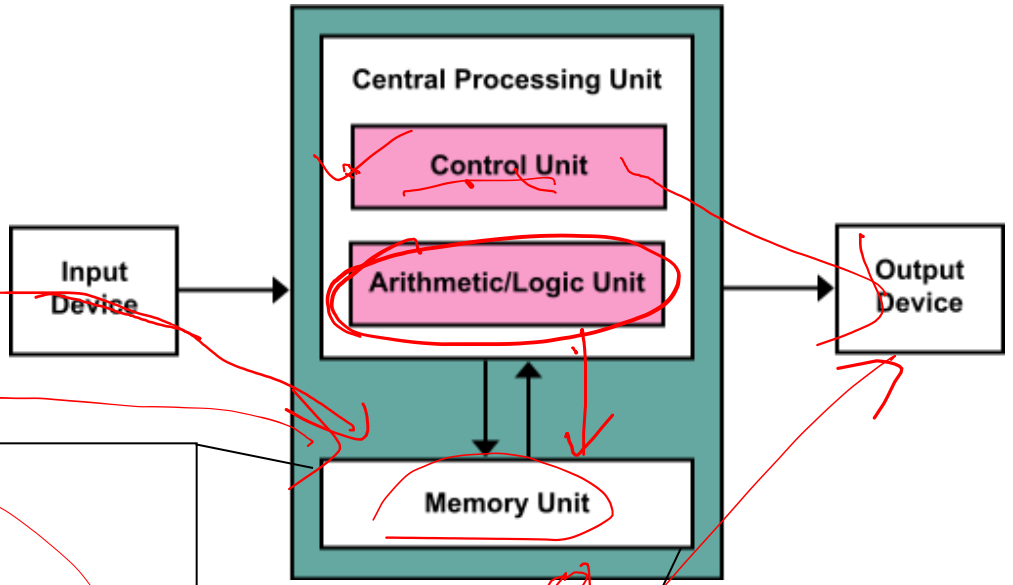
I/O devices

Memory

CPU

Question: how to program such a computer?

Programming the Computer from C Language



Code

Addition program in C

```c
#include <stdio.h>

int main()
{
  int x, y, z;

  printf("Enter two numbers to add\n");
  scanf("%d%d", &x, &y);

  z = x + y;

  printf("Sum of the numbers = %d\n", z);

  return 0;
}
```

Data

x

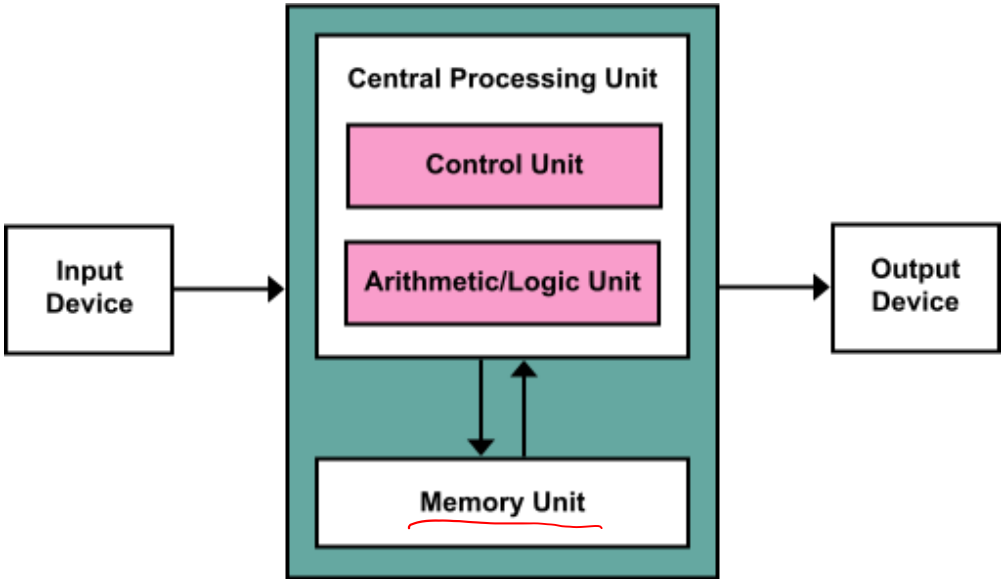
y

z

Computer Organization

Von Neumann Architecture

冯·诺伊曼架构



2. Three key abstractions:

- Data
- Instruction
- Sequential execution model

Question: what are the abstractions in C programming language?

Abstraction Comparison

- Von Neuman Architecture
  - Data
  - Instruction
  - Sequential execution model

- C programming language
  - Variable
  - Statement/code
  - Sequential execution model
- But more abstractions in C
  - Variable types, structs
  - Compound statement
  - Function calls

C programming language is just a higher level abstraction of Von Neuman architecture

We will teach you assembly programming so you will better understand high-level programming

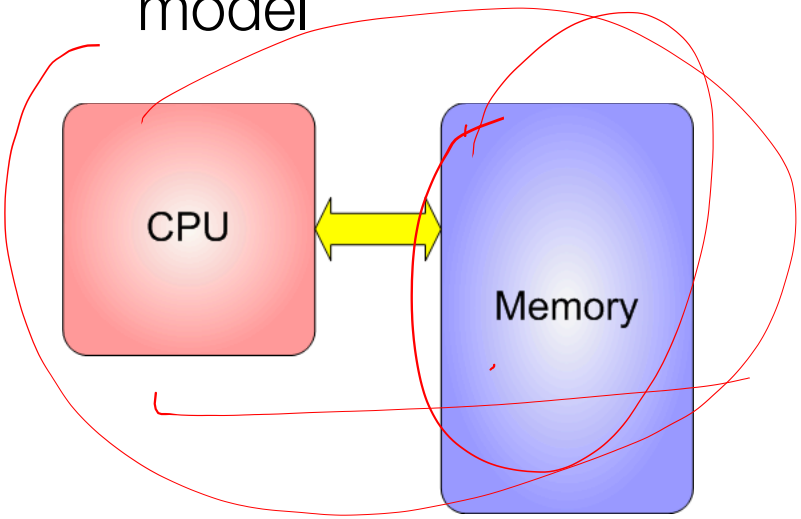
Interactions between the Components and Abstractions

- Abstraction
  - Data
  - Instruction
  - Sequential execution model
- Components
  - I/O
  - Memory
  - CPU

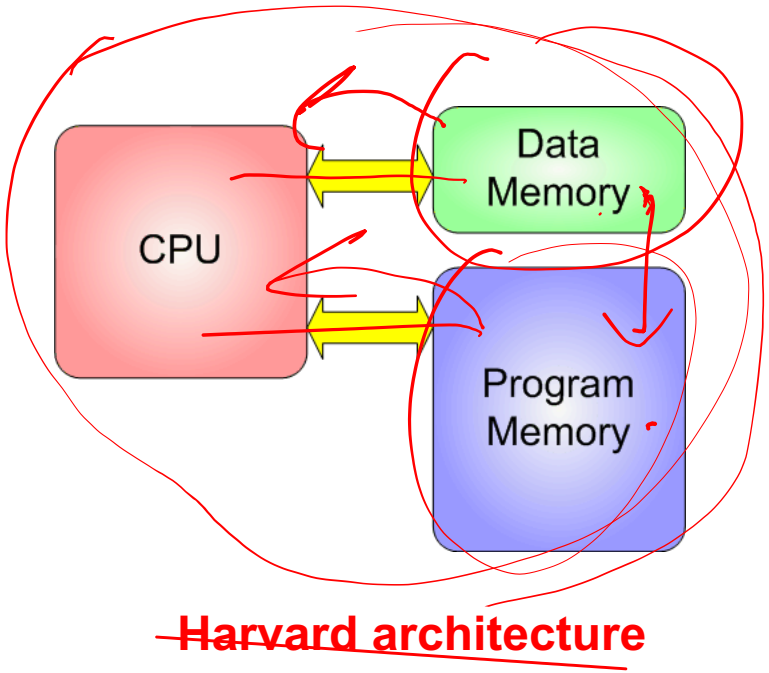
  - Data and instruction are stored in memory
  - CPU executes instruction in sequential order
  - Instruction can read and write memory
  - Instruction can perform arithmetic operations

Interactions between the Components and Abstractions

- Abstraction
  - Data
  - Instruction
  - Sequential execution model
- Components
  - I/O
  - Memory
  - CPU

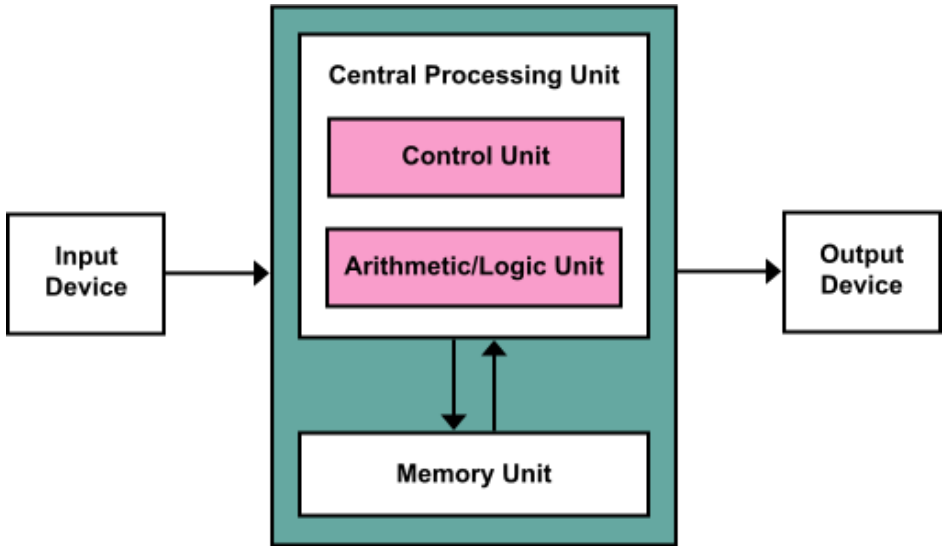


Von Neumann (a.k.a. Princeton) architecture



Harvard architecture

Von Neumann Architecture



1.Five components partitioning

- Input
- Output
- Memory
- ALU
- Control unit

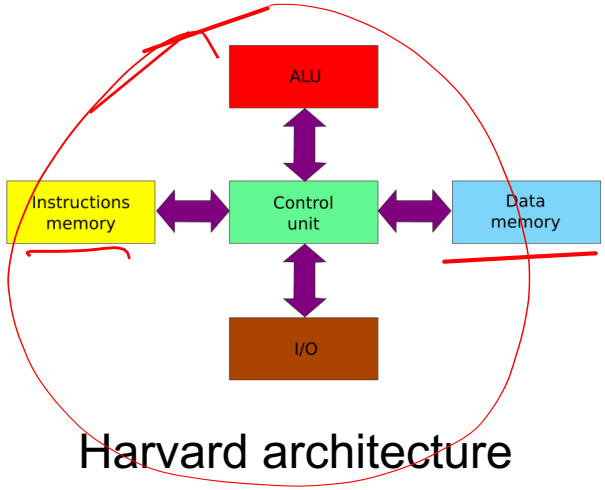
} I/O devices

Memory

} CPU

2. Three key concepts:

- Both instructions and data are stored in a **single** read-write memory
- The contents of memory are **addressable** by location, without regard to the type of data
- Execution occurs in a **sequential** fashion

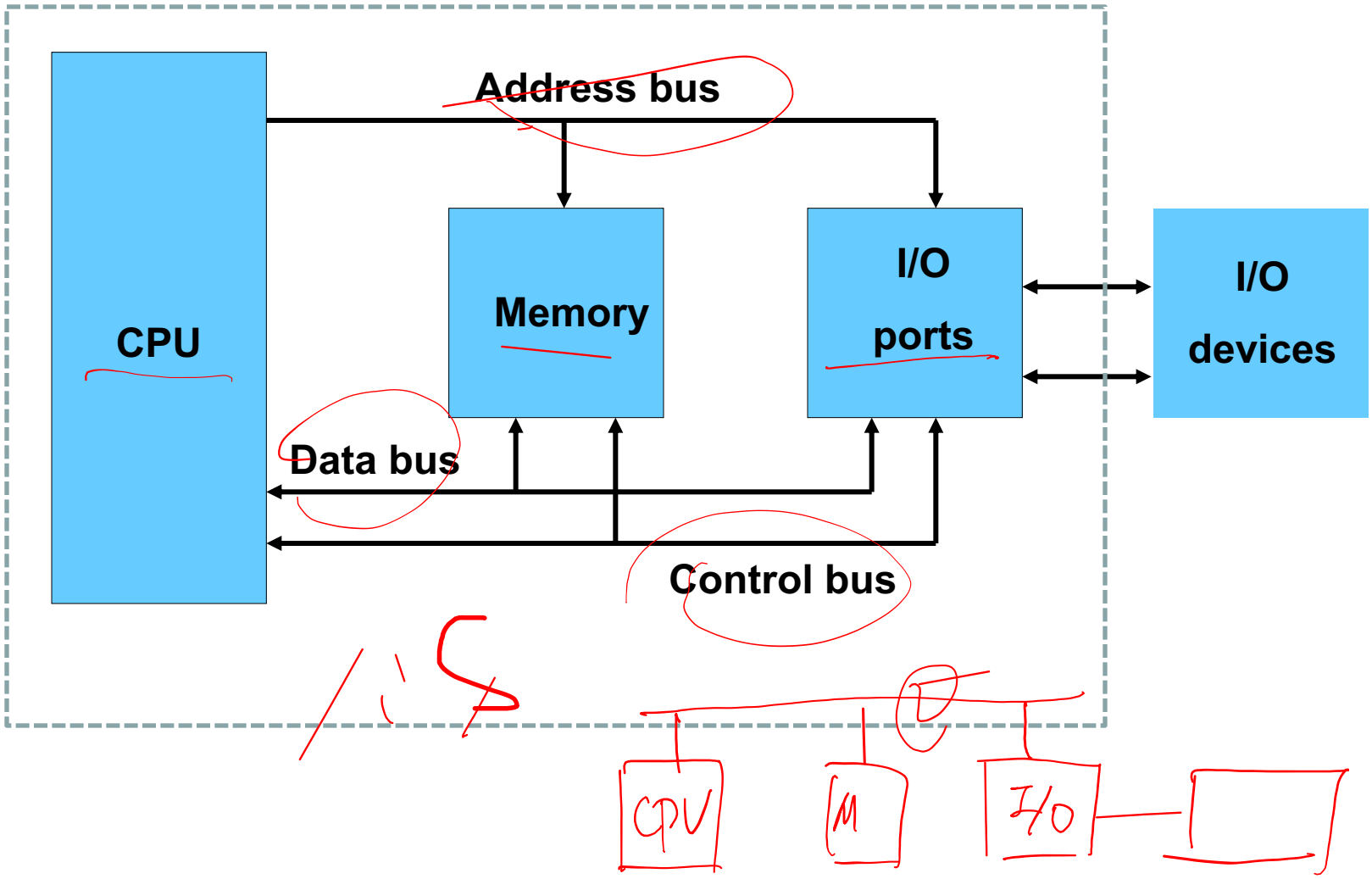


Harvard architecture

Microcomputer

- *CPU: processes information stored in the memory*
  - *Microprocessor*
- *Memory: stores both instructions and data*
  - ROM, RAM
- *Input/Output ports: provide a means of communicating with the CPU*
  - Connecting I/O devices, e.g., keyboard, monitor, tape, disk, printer and etc.
- *BUS: interconnecting all parts together*
  - Address bus
  - Data bus
  - Control bus

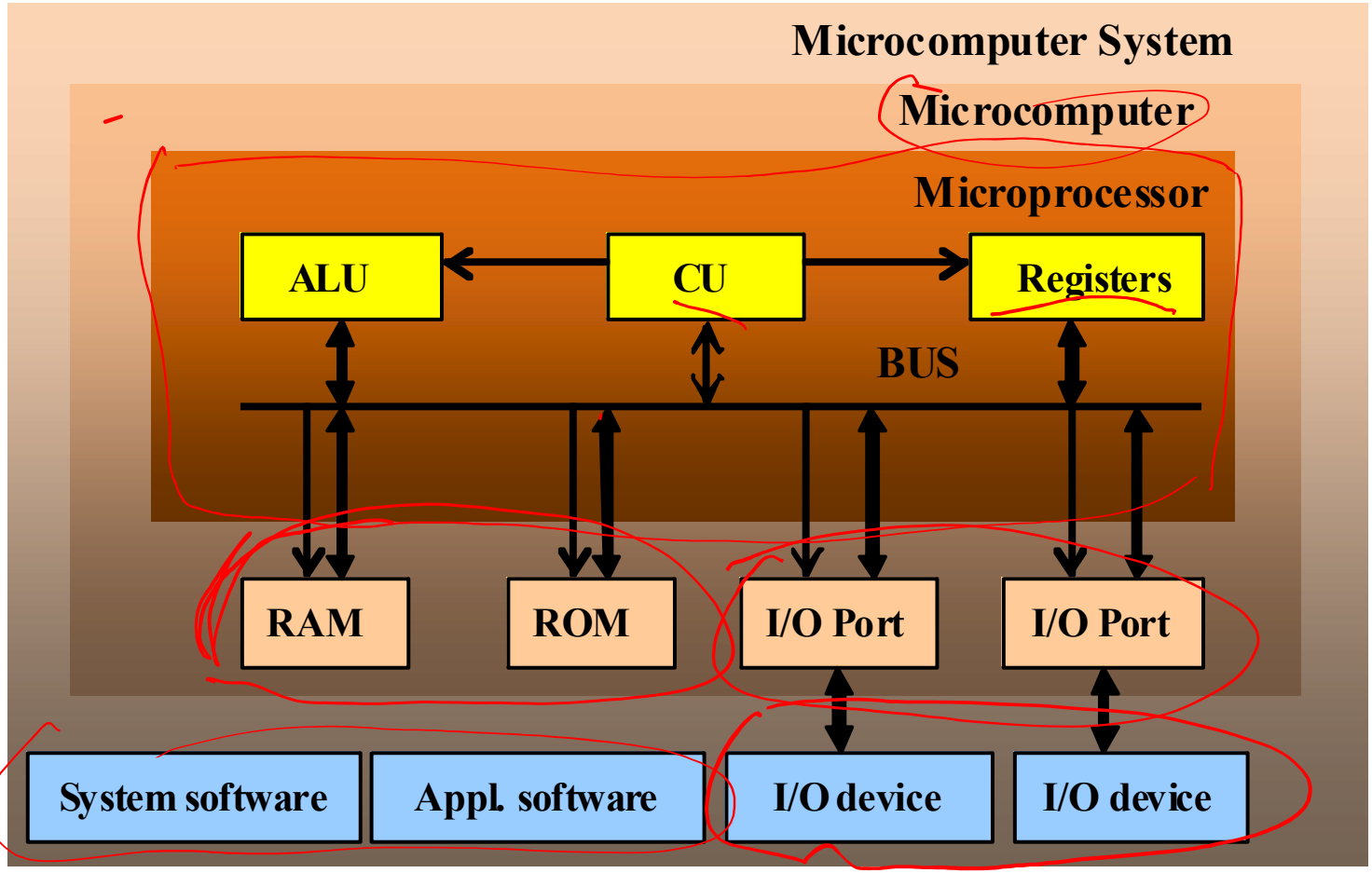
Microcomputer Structure



Microcomputer System

- Microcomputer
- Peripheral I/O devices
- Software
  - System software
    - e.g., OS, compilers, drivers
  - Application software
    - e.g. Word, MatLab, Media player, Latex…

Microcomputer System Structure

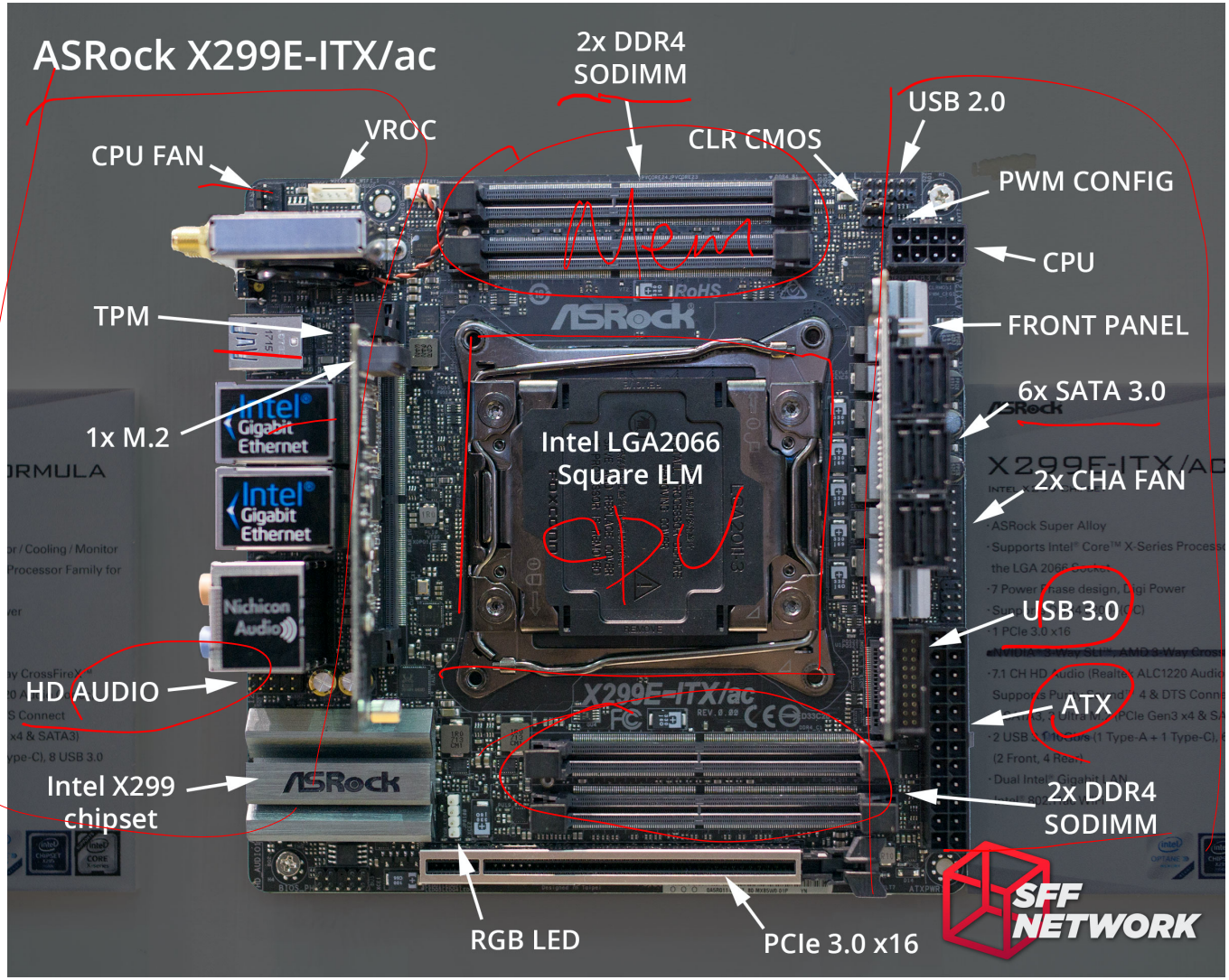


Microcomputer System Structure

Microcomputer system (μCS)

- Microcomputer (μC)
  - Microprocessor (μP)
    - ALU
    - CU
    - Registers
  - Memory
  - I/O ports
- I/O devices
- System software
- Application software

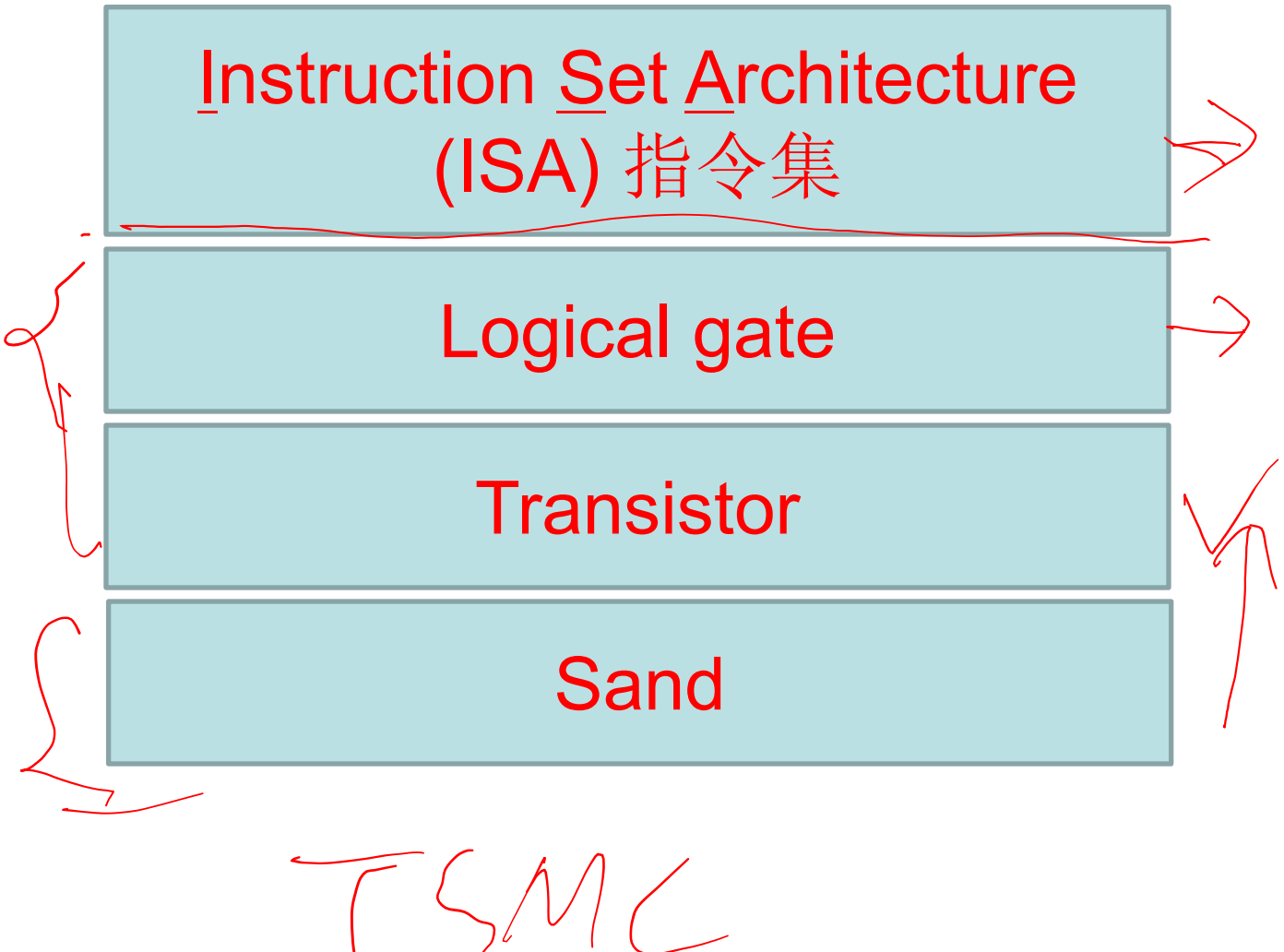
Quiz


ASRock X299E-ITX/ac
2x DDR4 SODIMM
USB 2.0
VROC
CPU FAN
CLR CMOS
PWM CONFIG
CPU
TPM
FRONT PANEL
6x SATA 3.0
1x M.2
Intel LGA2066 Square ILM
2x CHA FAN
USB 3.0
HD AUDIO
ATX
Intel X299 chipset
2x DDR4 SODIMM
RGB LED
PCIe 3.0 x16
SFF NETWORK

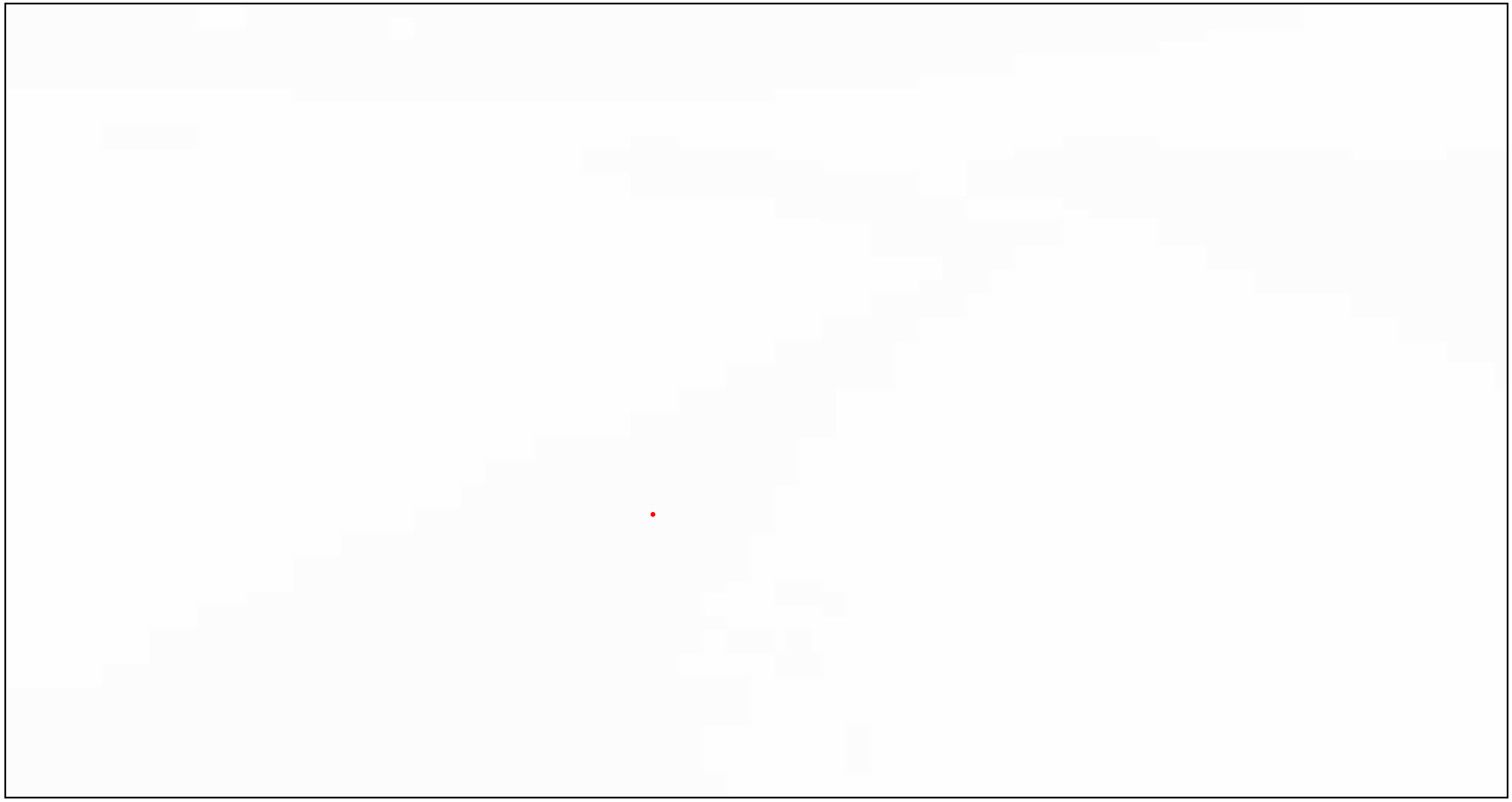
Outline

- Why/what you will learn from this course
- Computer organization
- CPU
- Memory
- Bus
- I/O
- What is an embedded system?

CPU/Microprocessor



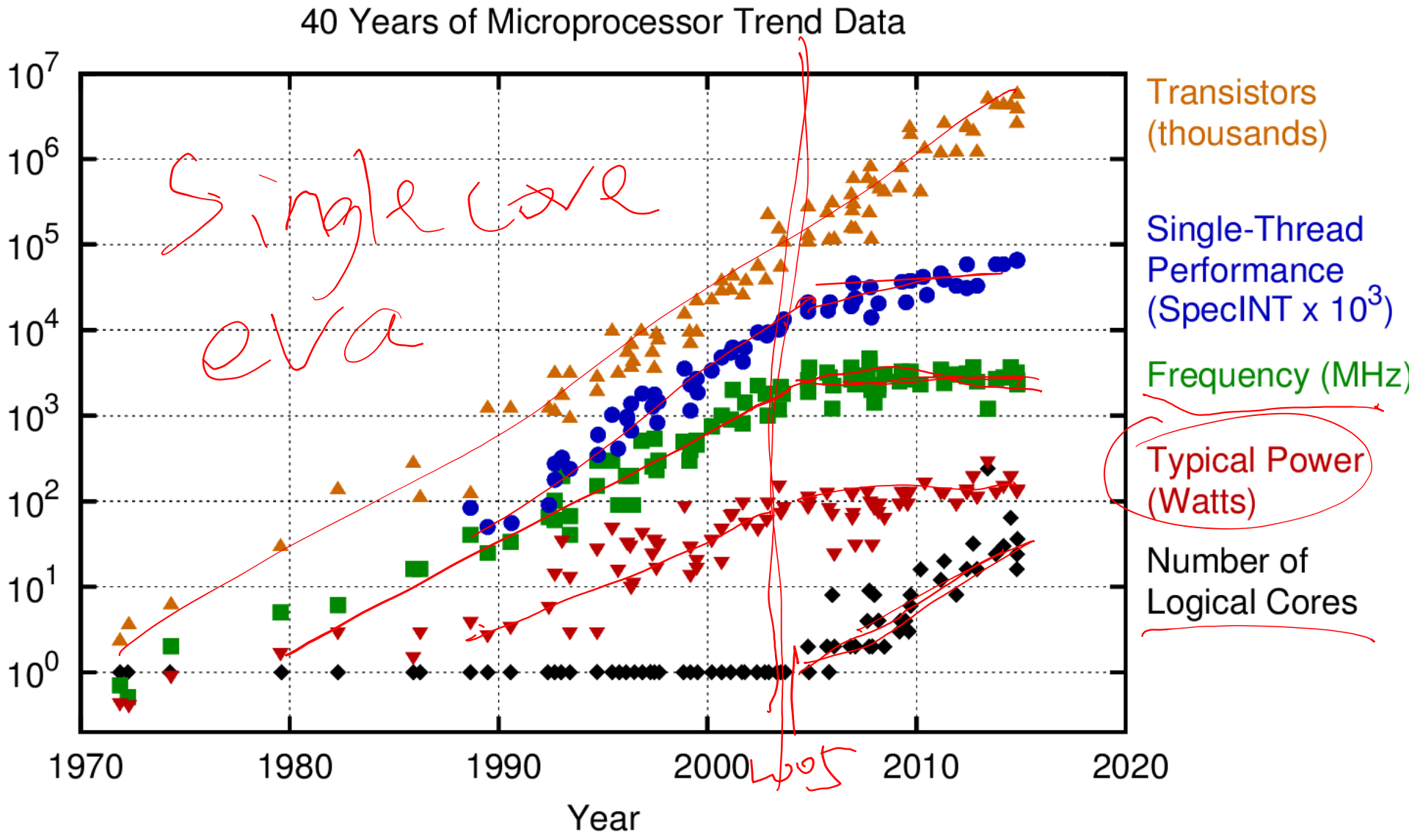
Inside a CPU



CPU/Microprocessor

- Key concept
  - Core 核心 (including ALU运算单元, CU控制单元, and register 寄存器)
  - Clock 时钟 (1 – 5 GHz, the CPU circuitry is made by digital circuits)
  - ISA/instruction set architecture/指令集 (Intel x86/ARM/PowerPC/MIPS/RISCV)

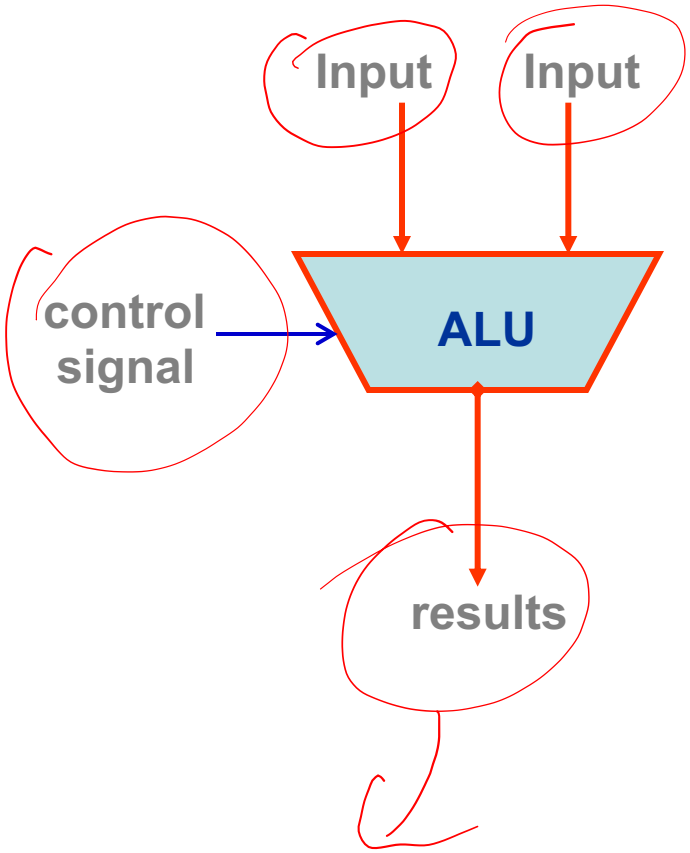
CPU Evolution



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2015 by K. Rupp

Hardware: CPU (1) - ALU

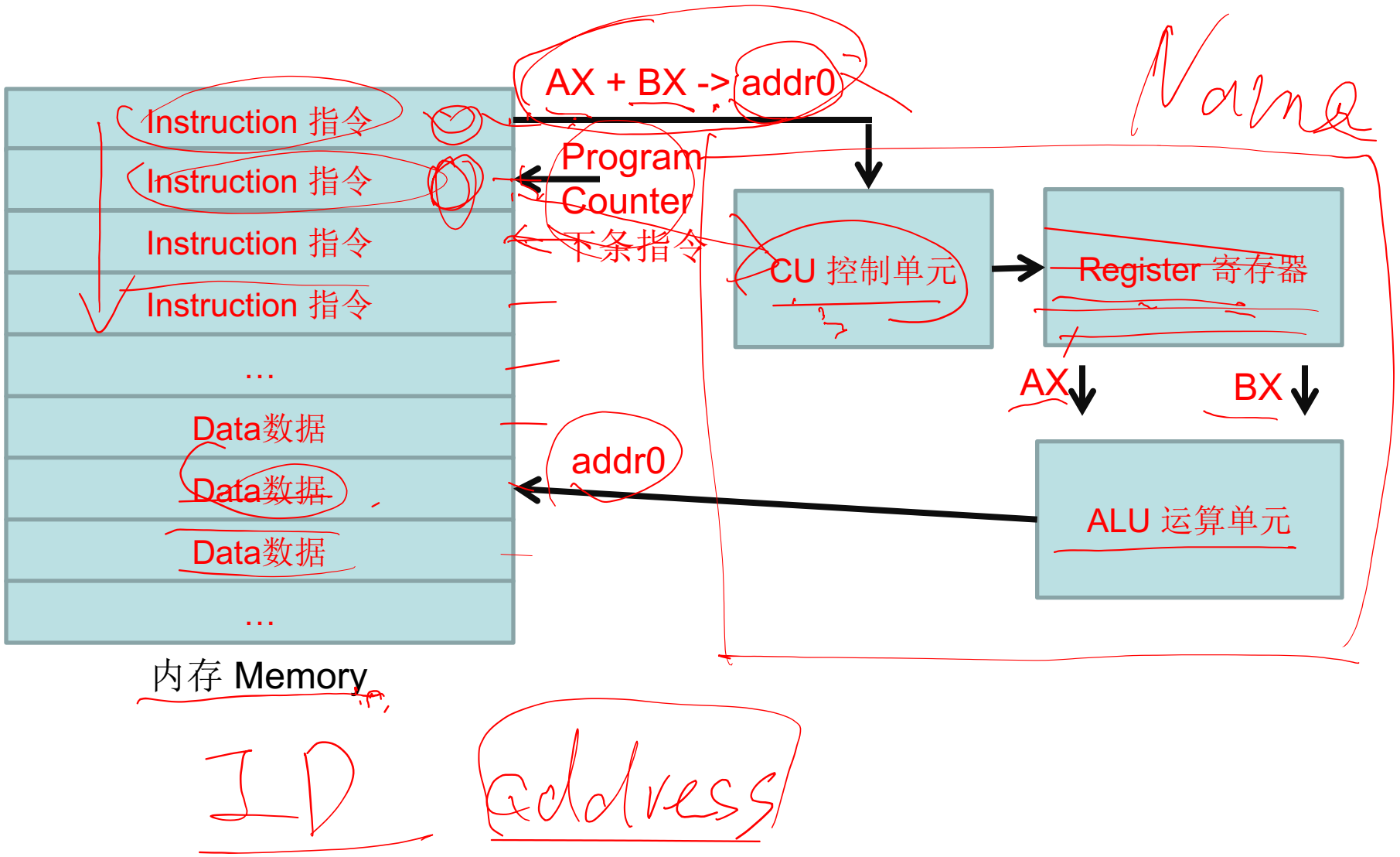
- Arithmetic Logic Unit (ALU)
  - Arithmetic functions: add, subtract, multiply and divide
  - Logic functions: AND, OR, and NOT
- ALU is a multifunctional calculator
  - What specific calculation will be taken depends on the particular control signal
- Two inputs
- Calculation result can be temporarily stored in one of the *registers*



Hardware: CPU (2) - CU

- Control Unit works under *instructions*
- An instruction is a pre-defined code which defines a specific operation, processing and exchanging information among CPU, memory and I/O devices.
- CU contains an *instructor decoder*
  - decodes an instruction and generates all control signals, coordinating all activities within the computer
- CU contains a *program counter*
  - points to the address of the next instruction to be executed

Hardware: CPU



Hardware: CPU (3) – Instruction Set

- The instruction set
  - All recognizable instructions by the instruction decoder

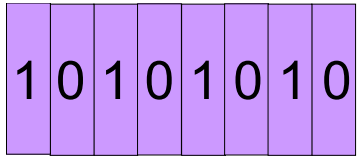
- CISC (Complex Instruction Set Computers)
  - Variable instruction length (1 word- *n* words)
  - Variable execution time of different format instructions
  - More instruction formats
  - Upwardly compatible (new instruction set contains earlier generation's instructions)
  - e.g., 80x86 family has more than 3000 instructions
- RISC (Reduced Instruction Set Computers)
  - Fixed size (1 word)
  - Fixed time for all instructions
  - Easy to pipeline the RISC instructions (fast)
  - Fewer formats (simple hardware, shorter design cycle)
  - e.g., PowerPC, MIPS, ARM, PIC's MCU

Outline

- Why/what you will learn from this course
- Computer organization
- CPU
- Memory
- Bus
- I/O
- What is an embedded system?

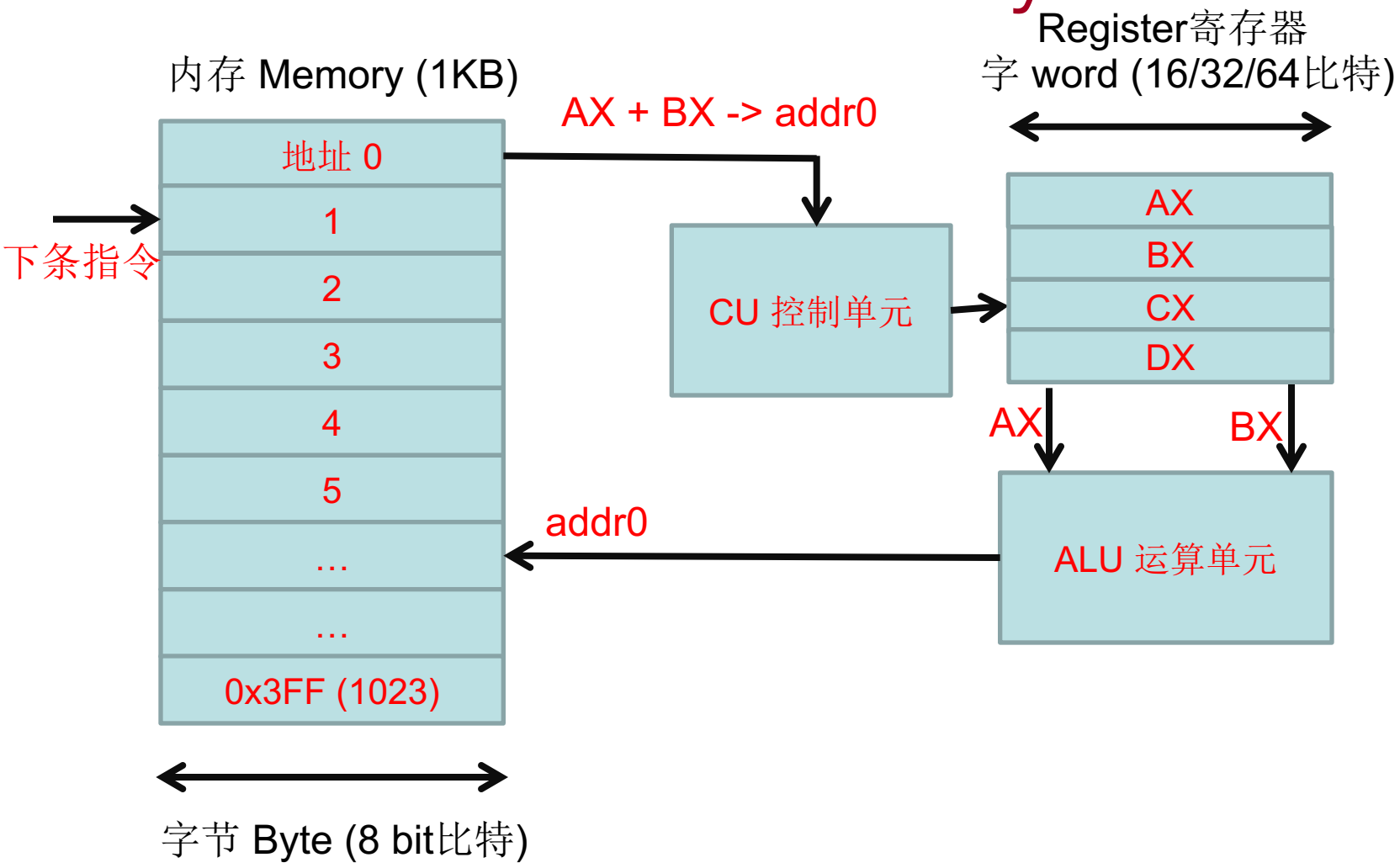
Hardware: Memory

- *Bit (b):* a binary digit that can have the value 0 or 1
- *Byte (B):* consists of 8 bits
  - smallest unit that can be addressed in microcomputers



- *Nibble:* is half a byte (4bits)
- *Word:* the number of bits that a CPU can process at one time
  - depends on the width of the CPU's registers and that of the data bus
    - e.g., if the width of the data bus is 16 bits, then a word is 16 bits; if the width of the data bus is 32 bits, then a word is 32 bits
- *Double word*
- Kilo (2^{10}/10^3), Mega(2^{20}/10^6), Giga(2^{30}/10^9), Tera(2^{40}/10^{12}), Peta(2^{50}/10^{15}), …

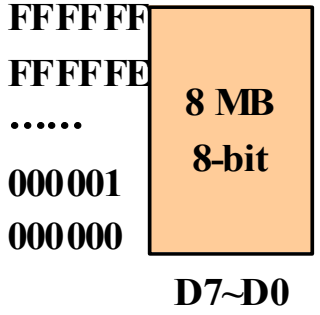
Hardware: Memory



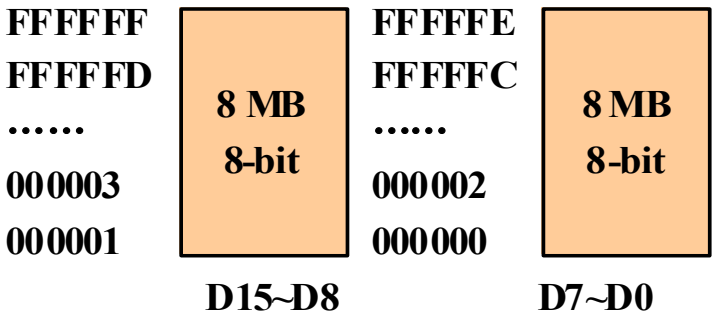
Memory Module Organization

- To organize a memory module:
  - If the module needs bigger **unit of transfer** than that of given memory chips, *bit extension*
  - If the module needs larger number of words than that of given memory chips, *word extension*

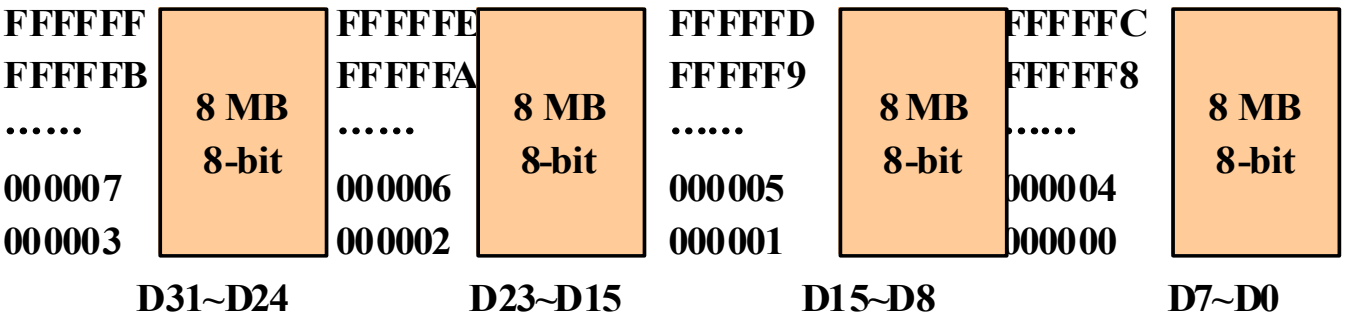
- 8-bit



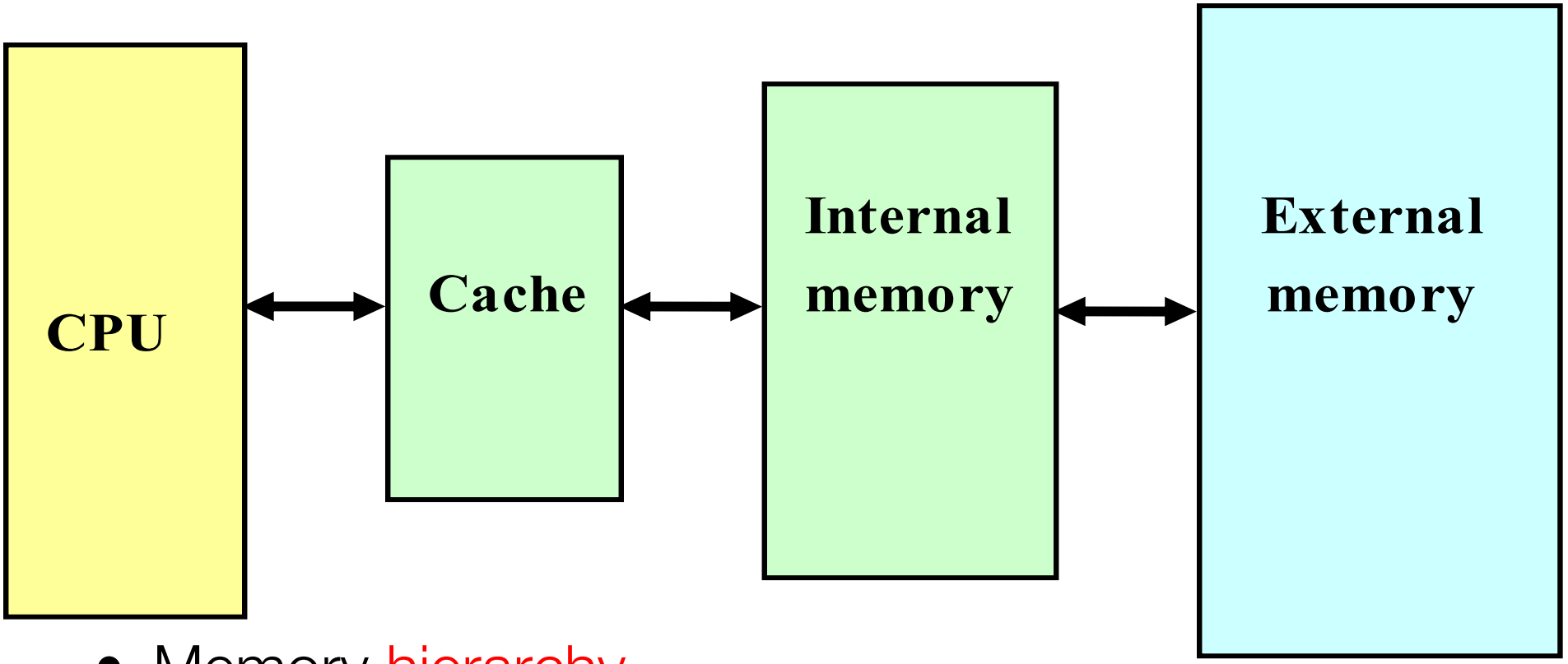
- 16-bit



- 32-bit



Hardware: Memory



- Memory hierarchy
  - *Cache*
  - *Primary memory: ROM, RAM*
  - *Secondary memory: magnetic disk, optical memory, tape, …*

Outline

- Why/what you will learn from this course
- Computer organization
- CPU
- Memory
- Bus
- I/O
- What is an embedded system?

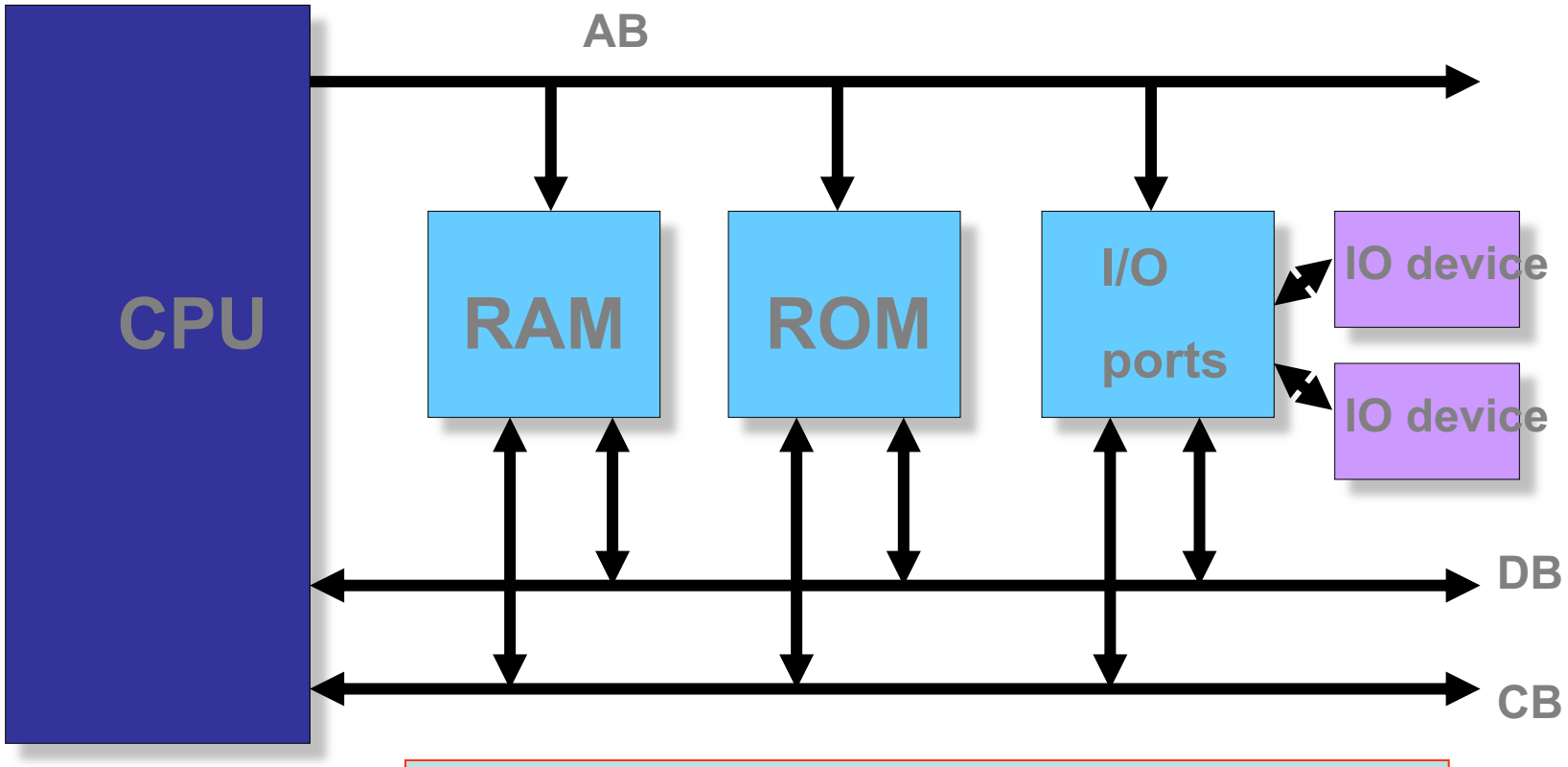
Hardware: Bus

- A bus is a communication pathway connecting two or more devices
  - A shared transmission medium: one device at a time
  - **System bus:** connects major computer components (processor, memory, I/O)
  - Arbitration:
    - Distributed protocols
      - e.g., CSMA/CD
    - Centralized scheme:
      - Master/Slave
        - » Master activates a bus
        - » Slave passively waits for command

Hardware: Bus

- Type
  - Dedicated (e.g., physical dedication)/Multiplexed (e.g., time multiplexing)
- Arbitration
  - Centralized: *bus controller* responsible for allocating time on a bus
  - Distributed: each module has access control logic and collaborate
- Timing
  - Synchronous: events on the bus is determined by a global clock, a single 1-0 transmission is referred to as a *bus cycle*
  - Asynchronous: devices have their own clocks and communicate before and after an event

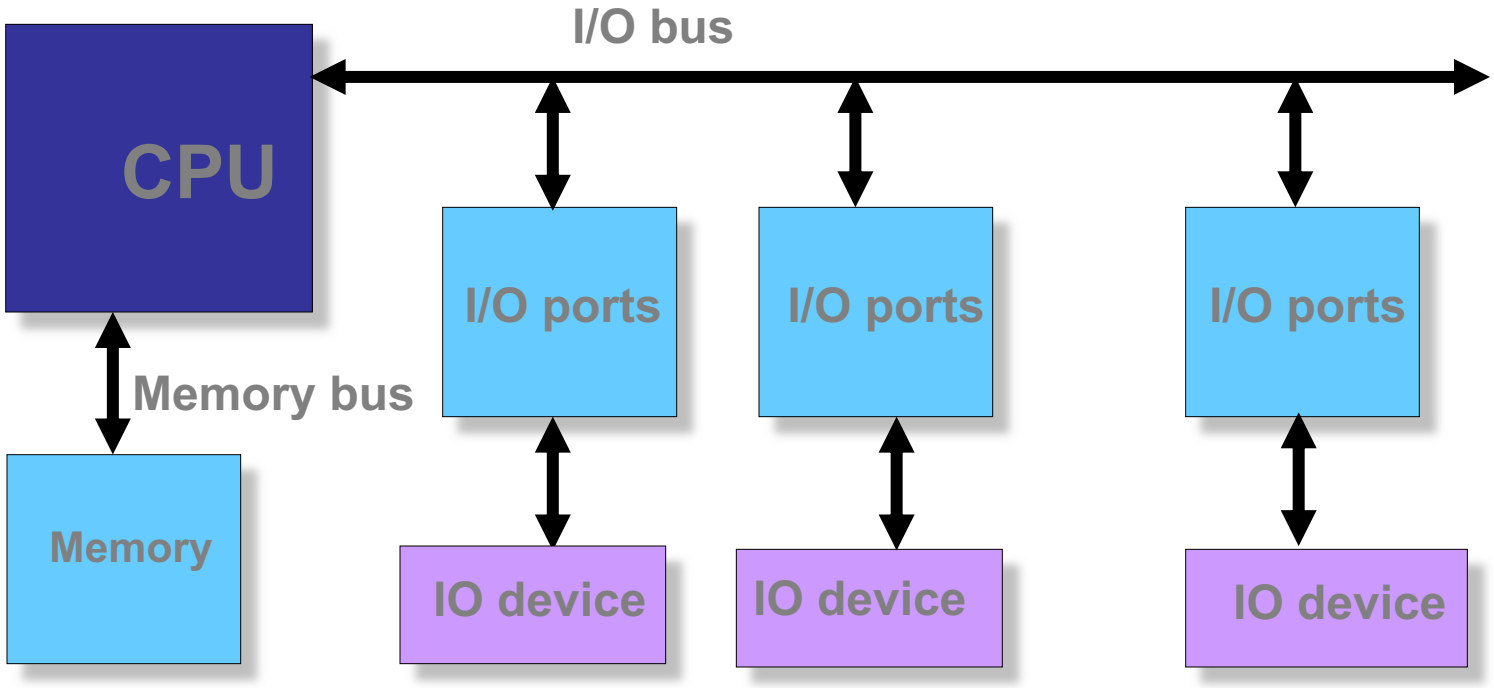
Single-bus Structure



A bus connects all modules

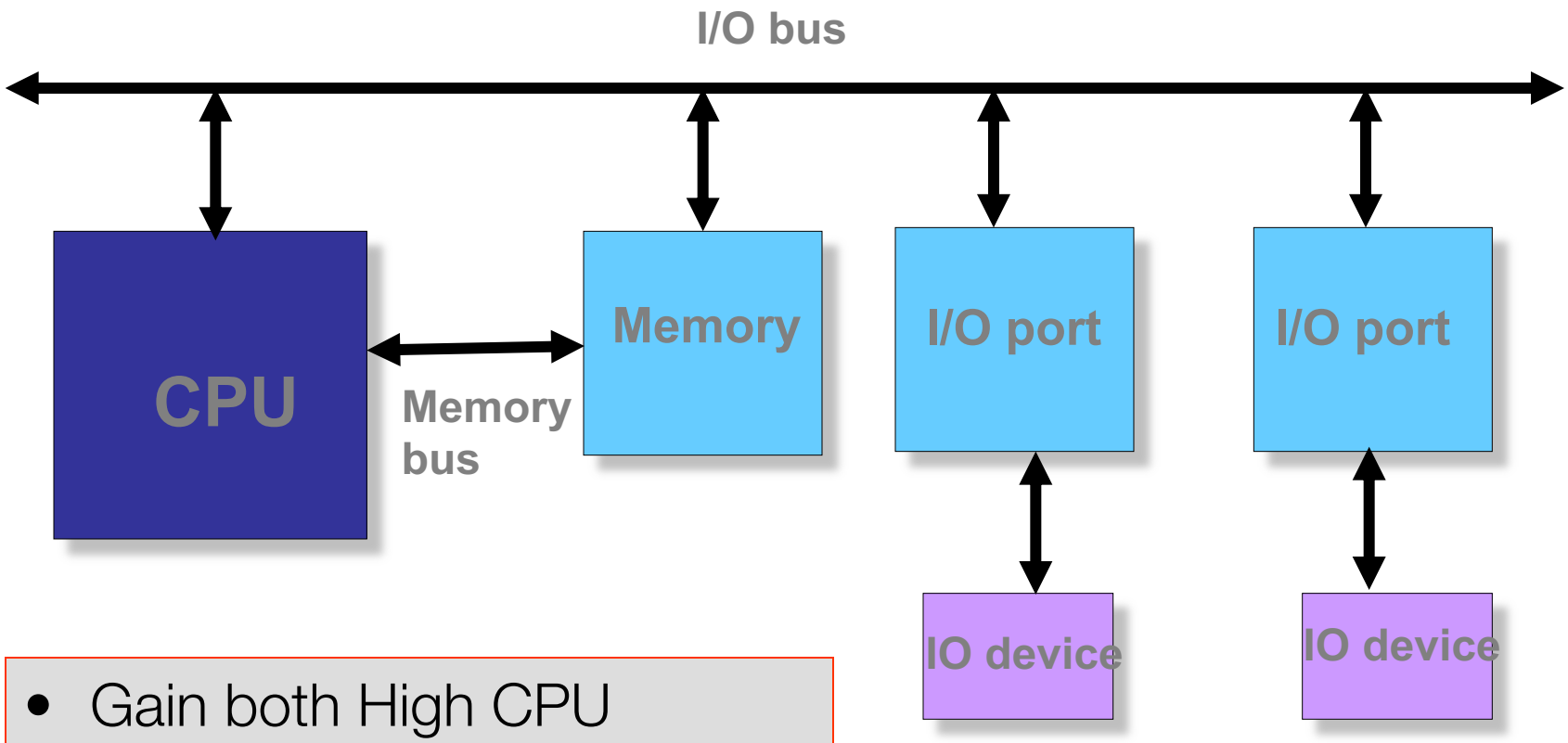
- pro: simple
- con: poor performance in terms of throughput

CPU-Central Dual-Bus Structure



- A dedicated bus between CPU and memory, and a dedicated bus between CPU and I/O devices
- Pros (Advantages): efficient in terms of data transfer
- Cons (Disadvantages): information between memory and I/O devices has to go through CPU. Therefore, poor CPU performance

Memory-Central Dual-Bus Structure



- Gain both High CPU performance and data transfer throughput

Hardware: BUS (1) – Data Bus

- Used to provide a path for moving data between system modules
- Bidirectional
  - CPU read: Memory (I/O device) -> CPU
  - CPU write: CPU -> Memory (I/O device)
- The width of data bus
  - is as wide as the registers of a CPU (i.e. the width of a *word*)
  - determines how much data the processor can read or write in one memory or I/O cycle
  - Which also defines a word of this computer

Hardware: BUS (2) - Address Bus

- Used to designate the source or destination of the data on the data bus that the processor intends to communicate with
- Unidirectional
  - CPU -> memory| I/O device
- The width of the address bus, *n*
  - determines the total number of memory locations addressable by a given CPU, which is 2^n
  - *e.g.,* 8086 has a 20-bit address bus which corresponds to 2^{20} addresses or 1M (1 Meg) addresses or memory locations;
  - *Pentium has 32-bit address bus, what is the size of its addressable memory?*
  - *How to calculate the capacity (size) of memory that a CPU can support then?*

Hardware: BUS (3) – Control Bus

- Used to control each module and the use of data and address buses
  - Command and timing information between modules
  - e.g., memory read/write, IO read/write, Bus request/grant
- Consists of two sets of unidirectional control signals
  - Command signal: CPU -> Memory (I/O device)
  - State signal: Memory (I/O device) -> CPU
- Input/Output is defined from the processor's point of view
  - e.g., when Memory (I/O device) Read is active, data is input to the processor

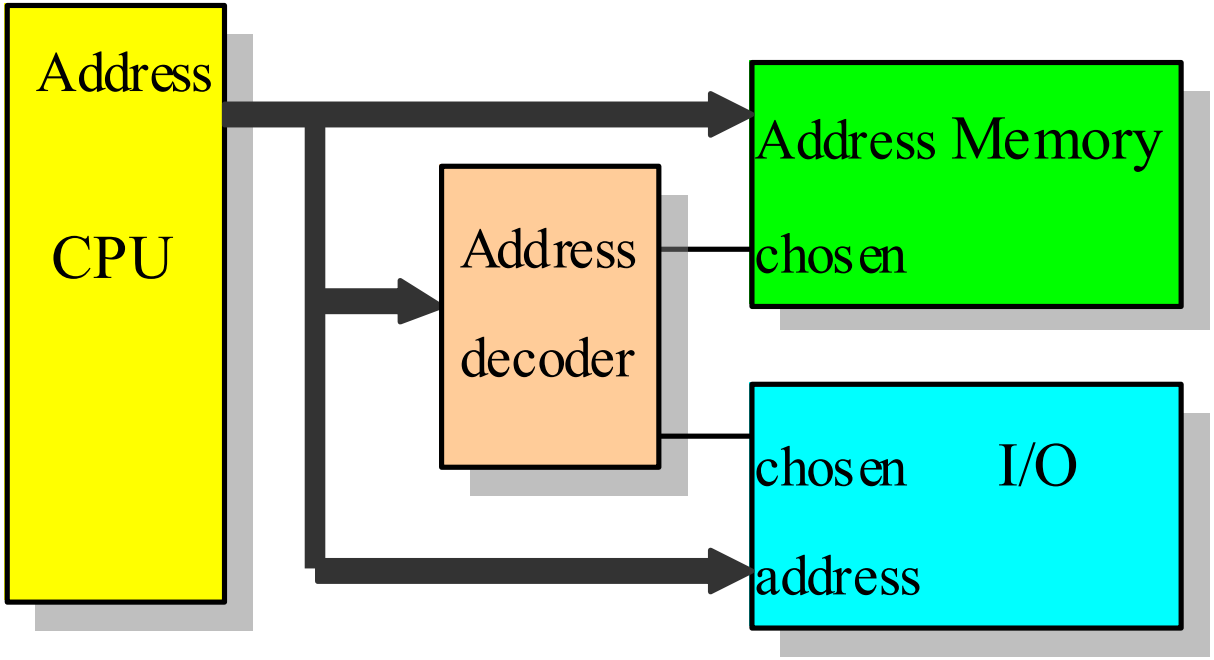
Outline

- Why/what you will learn from this course
- Computer organization
- CPU
- Memory
- Bus
- I/O
- What is an embedded system?

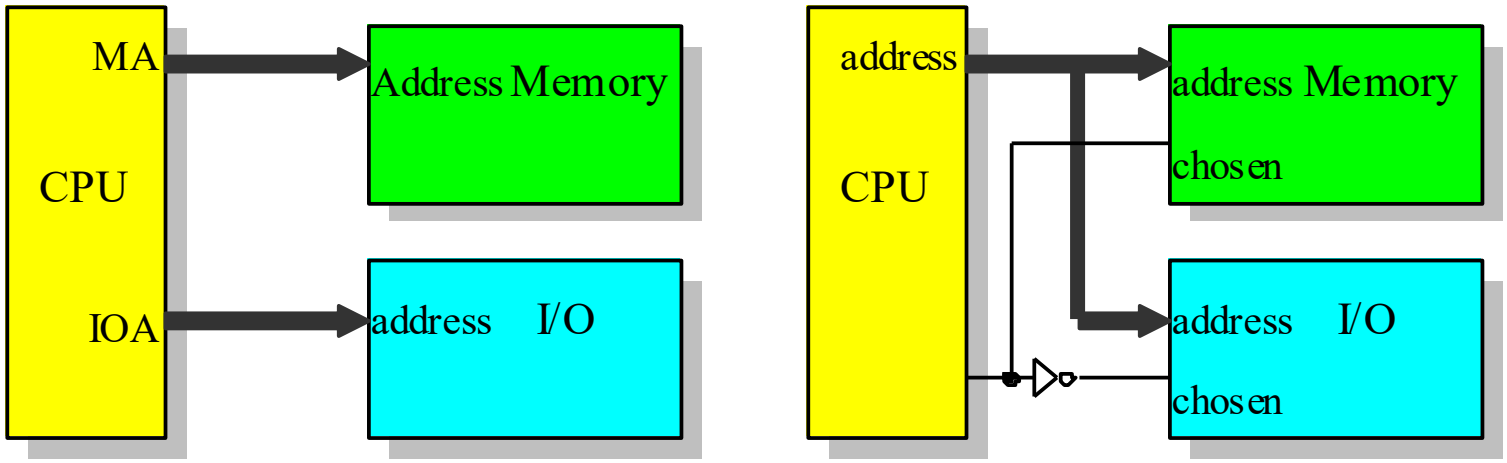
Addressing scheme to accessing memory and I/O modules

- Memory-mapped I/O
  - One single address space for both memory and I/O
  - Status and data registers of I/O modules are treated as memory locations
  - Using the same machine instructions to access both
- Isolated I/O
  - Two separate address spaces for memory and I/O modules
  - Using different sets of accessing instructions

Memory-mapped I/O



Isolated I/O



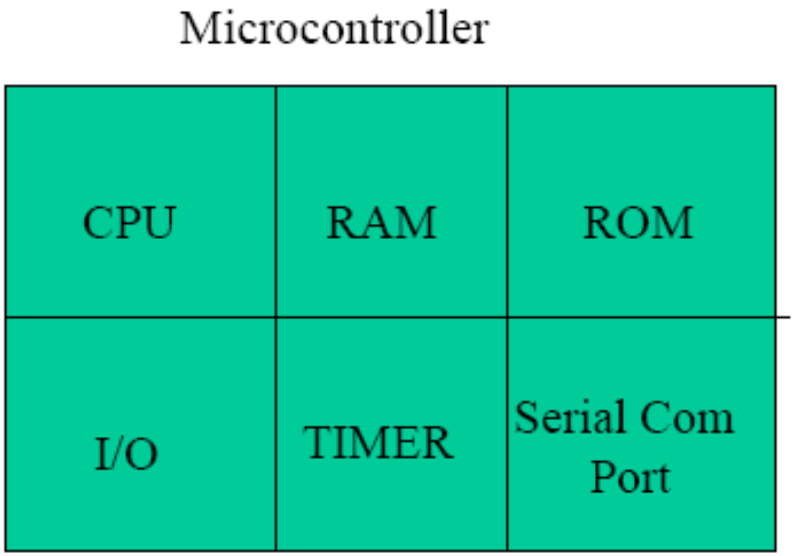
Dedicated address lines **Multiplexing address lines**

What is the essential difference between the memory-mapped and isolated I/O addressing schemes?

Microcontrollers (MCS)

A microcontroller has a CPU in addition to a fixed amount of RAM, ROM, I/O ports on one single chip

- Ideal for applications in which cost and space are critical
- Example: a TV remote control does not need the computing power of a 486



Embedded Systems

- An embedded system uses a *microcontroller* or a *microprocessor* to do one task and one task only
  - Example: toys, TV remote, keyless entry, etc.
- Using microcontrollers is cheap but sometimes inadequate for the task
- Microcontrollers differ in terms of their RAM,ROM, I/O sizes and type.
  - ROM (often used as program memory, like BIOS)
    - OTP (One Time-Programmable)
    - UV-ROM, EEPROM
    - Flash memory
  - RAM (can be used as both program mem and data mem)
    - SRAM(static RAM):cache
    - DRAM(Dynamic RAM): main memory
      - SDRAM (Synchrous DRAM)
      - DDR DRAM (Double Data Rate DRAM)
      - DDRII