

计算机组成原理复习笔记

计算机组成原理复习笔记

一、计算机系统概述

- 1.1 计算机发展历程
- 1.2 计算机系统层次结构
- 1.3 计算机的性能指标

二、数据的表示和运算

- 2.1 数制与编码
- 2.2 运算方法与运算电路
- 2.3 浮点数的表示与运算

三、存储系统

- 3.1 存储器概述
- 3.2 主存储器
- 3.3 主存储器与 CPU 的连接
- 3.4 外部存储器
- 3.5 高速缓冲存储器 (Cache)

四、指令系统

- 4.3 程序的机器级代码表示
- 4.4 CISC 和 RISC 的基本概念

五、中央处理器

六、总线

- 6.1 总线概述
- 6.2 总线事务和定时

七、输入/输出系统

一、计算机系统概述

- 计算机系统：硬件 + 软件

1.1 计算机发展历程

- 计算机硬件的发展

- 1946 年世界上第一台电子数字计算机：ENIAC

发展阶段	时间	逻辑元件
第一代	1946 – 1957	电子管
第二代	1958 – 1964	晶体管
第三代	1964 – 1971	中小规模集成电路
第四代	1972 – 至今	超大规模集成电路

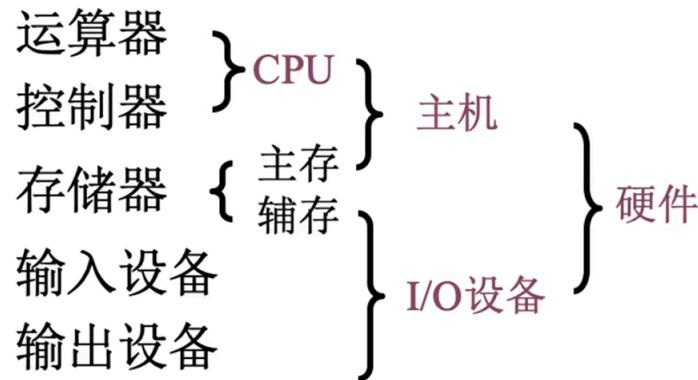
- 计算机软件的发展

- 计算机语言的发展：FORTRAN、PASCAL、C++、Java、Python
 - 操作系统的发展：Windows、UNIX、Linux

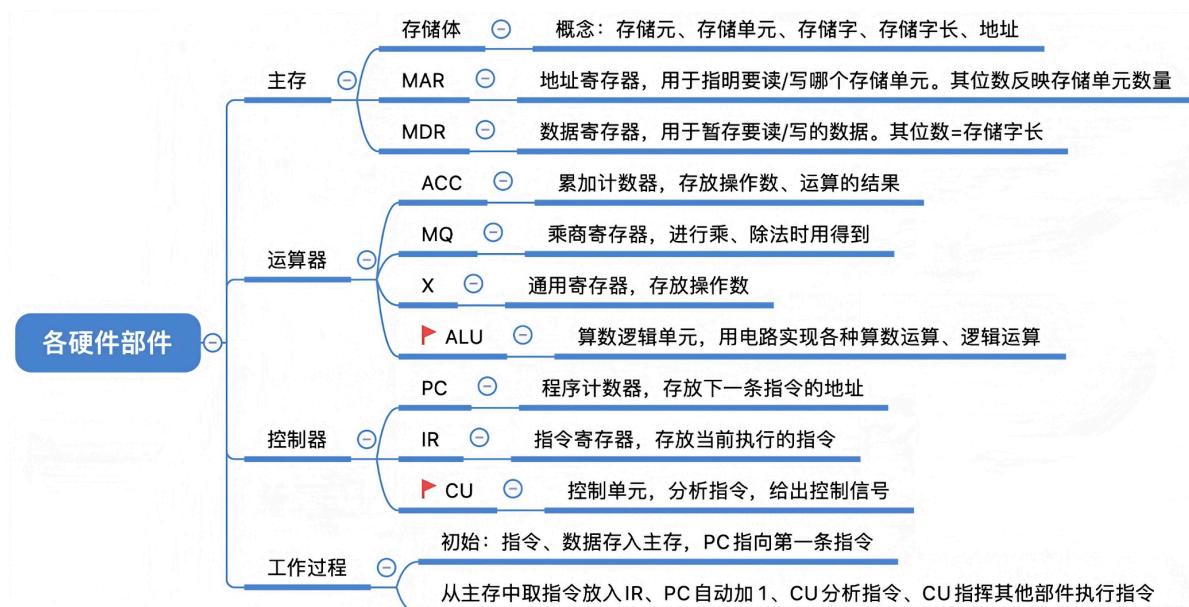
1.2 计算机系统层次结构

- 计算机硬件的基本组成

- 输入设备
- 输出设备
- 存储器：分为主存储器（内存）和辅存储器（外存，如硬盘等）
- 运算器
- 控制器



- 冯诺依曼结构：以运算器为中心
- 现代计算机结构：以存储器为中心
- 各硬件部件的工作原理
 - 各硬件部件
 - 主存：存储体、MAR、MDR
 - 运算器：ACC、MQ、X、ALU
 - 控制器：PC、IR、CU
 - 工作过程
 - 初始：指令、数据存入主存，PC 指向第一条指令
 - 从主存中取指令放入 IR，PC 自动加 1，CU 分析指令，CU 指挥其它部件执行指令



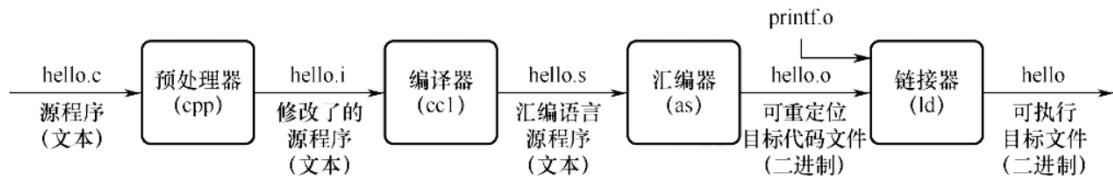
- 计算机软件

- **系统软件**: 用来管理整个计算机系统
 - 如: 操作系统、数据库管理系统、标准程序库、网络软件
- **应用软件**: 按任务需要编制成的各种程序
- **三种级别的语言**
 - **机器语言**: 二进制代码
 - **汇编语言**: 助记符
 - **高级语言**: 编译型语言 (C、C++)、解释型语言 (JavaScript、Python、Shell)

- **翻译程序**

- **汇编器**: 将汇编语言程序翻译成机器语言程序
- **解释器**: 将源程序中的语句按执行顺序逐条翻译成机器指令并立即执行
- **编译器**: 将高级语言程序翻译成汇编语言或机器语言程序

- **从源程序到可执行文件**



1.3 计算机的性能指标

- **机器字长**: 计算机进行一次整数运算所能处理的二进制数据的位数
 - 字长一般等于寄存器的位数或 ALU 的宽度
 - 32 位, 64 位机器的“32”“64”指的就是机器字长
- **数据通路带宽**: 数据总线一次所能并行传送信息的位数
- **主存容量**: 主存储器所能存储信息的最大容量
 - MAR 的位数反映了存储单元的个数
 - MDR 的位数反映了存储单元的字长
- **CPU 的性能指标**
 - **CPU 时钟周期**: CPU 中最小的时间单位
 - **主频 (CPU 时钟频率)**

$$\text{主频} = \frac{1}{\text{时钟周期}}$$

- **CPI**: 执行一条指令所需的时钟周期数
- **IPS**: 每秒执行多少条指令

$$\text{IPS} = \frac{\text{主频}}{\text{CPI}_{\text{平均}}}$$

- **FLOPS**: 每秒执行多少次浮点运算
- **CPU 执行时间**

$$\text{CPU 执行时间} = \frac{\text{CPU 时钟周期数}}{\text{主频}} = \frac{\text{指令条数} \times \text{CPI}}{\text{主频}}$$

- 上式表明, CPU 的性能取决于: 主频、CPI、指令条数

二、数据的表示和运算

2.1 数制与编码

- **进位计数法**

- 一个 r 进制数 $(K_n K_{n-1} \dots K_0 K_{-1} \dots K_{-m})$ 的数值可表示为:

$$K_n r^n + K_{n-1} r^{n-1} + \dots + K_0 r^0 + K_{-1} r^{-1} + \dots + K_{-m} r^{-m} = \sum_{i=n}^{-m} K_i r^i$$

- **定点数的编码表示**

- 定点数: 小数点的位置固定, 如 996.007
 - 浮点数: 小数点的位置不固定, 如 9.96007×10^2

- **无符号数**: 整个机器字长的全部二进制位均为数值位, 没有符号位

- **有符号数**

- **原码**: 用机器数的最高位表示数的符号, 其余各位表示数的绝对值

如, $x_1 = +1110, x_2 = -1110$, 字长为 8 位, 则其原码表示为

$$[x_1]_{\text{原}} = 00001110, [x_2]_{\text{原}} = 10001110$$

- **反码**: 正数的反码与原码相同, 负数的反码为原码将数值位全部取反

如, $x_1 = +10011, x_2 = -10011$, 字长为 8 位, 则其反码表示为

$$[x_1]_{\text{反}} = 00010011, [x_2]_{\text{反}} = 11101100$$

- **补码**: 正数的补码与原码相同, 负数的补码为反码 +1

对向量 $\vec{x} = [x_{w-1}, x_{w-2}, \dots, x_0]$,

$$B2T_w(\vec{x}) = -x_{w-1} 2^{w-1} + \sum_{i=0}^{w-2} x_i 2^i$$

如, $x_1 = +10011, x_2 = -10011$, 字长为 8 位, 则其补码表示为

$$[x_1]_{\text{补}} = 00010011, [x_2]_{\text{补}} = 11101101$$

- **移码**: 在补码的基础上, 将符号位取反

- **C 语言中的整数类型及类型转换**

- C 语言中的定点整数是用**补码**存储的

C声明		字节数	
有符号	无符号	32位	64位
[signed] char	unsigned char	1	1
short	unsigned short	2	2
int	unsigned	4	4
long	unsigned long	4	8
int32_t	uint32_t	4	4
int64_t	uint64_t	8	8
char *		4	8
float		4	4
double		8	8

- 强制类型转换

- 无符号数与有符号数：不改变数据内容，改变解释方式
- 长整数变短整数：高位截断，保留低位
- 短整数变长整数：符号扩展

- 零扩展和符号扩展

- 零扩展：适用于无符号整数，用0扩展高位
- 符号扩展：适用于带符号整数（补码），用符号位扩展高位

2.2 运算方法与运算电路

2.3 浮点数的表示与运算

- 浮点数的表示

$$V = (-1)^s \times M \times 2^E$$

- 符号： s 决定这数是负数 ($s = 1$) 还是正数 ($s = 0$)
 - 一个单独的符号位 s 直接编码符号 s
- 阶码： E 的作用是对浮点数加权，权重是 2 的 E 次幂
 - k 位的阶码字段 $\text{exp} = e_{k-1} \dots e_1 e_0$ 编码阶码 E
- 尾数： M 是一个二进制小数，范围是 $1 \sim 2 - \epsilon$ ，或者是 $0 \sim 1 - \epsilon$
 - n 位的小数字段 $\text{frac} = f_{n-1} \dots f_1 f_0$ 编码尾数 M

- 浮点数编码的值

- 情况 1：规格化的值：当 exp 的位模式既不全为 0，也不全为 1 时
 - $E = e - Bias, Bias = 2^{k-1} - 1, M = 1 + f$
- 情况 2：非规格化的值：当 exp 的位模式全为 0 时
 - $E = 1 - Bias, M = f$
- 情况 3：特殊值：当 exp 的位模式全为 1 时

- 小数位全为 0 时: $s = 0$ 时为 $+\infty$, $s = 1$ 时为 $-\infty$
- 小数位为非零时: 为 NaN
- IEEE 754 标准
 - 在单精度浮点格式 (C 语言中的 float) 中, s 、 exp 和 $frac$ 字段分别为 1 位、 $k = 8$ 位和 $n = 23$ 位, 得到一个 32 位的表示
 - 在双精度浮点格式 (C 语言中的 double) 中, s 、 exp 和 $frac$ 字段分别为 1 位、 $k = 11$ 位和 $n = 52$ 位, 得到一个 64 位的表示

1位	8位	23位
符号	阶码	尾数

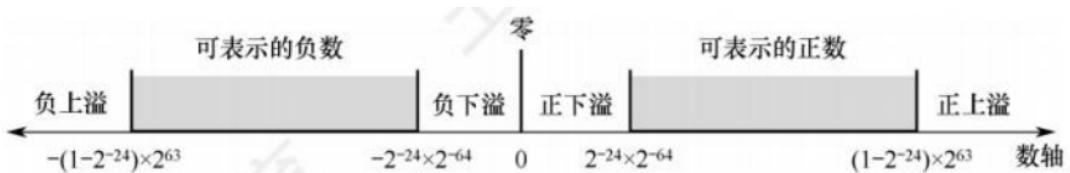
(a) 32位单精度格式

1位	11位	52位
符号	阶码	尾数

(b) 64位双精度格式

• 浮点数的表示范围

- 对于单精度浮点格式的规格化数, 范围是
 $[-(2 - 2^{-23}) \times 2^{127}, -2^{-126}] \cup [2^{-126}, (2 - 2^{-23}) \times 2^{127}]$
- 对于单精度浮点格式的非规格化数, 范围是
 $[-\frac{2^{23}-1}{2^{23}} \times 2^{-126}, -2^{-149}] \cup [2^{-149}, \frac{2^{23}-1}{2^{23}} \times 2^{-126}]$
- 当运算结果大于最大正数时称为**正上溢**, 小于绝对值最大负数时称为**负上溢**, 正上溢和负上溢统称**上溢**
 - 数据上溢时, 用 $+\infty$ 和 $-\infty$ 表示
- 当运算结果在 0 至最小正数之间时称为**正下溢**, 在 0 至绝对值最小负数之间时称为**负下溢**, 正下溢和负下溢统称**下溢**
 - 数据下溢时, 浮点数值趋近于 0, 计算机将其当做机器零处理



• 数据的大小端存储

- **大端法**: 先存储高位字节, 后存储低位字节
- **小端法**: 先存储低位字节, 后存储高位字节 (低位在低地址)

• 数据对齐

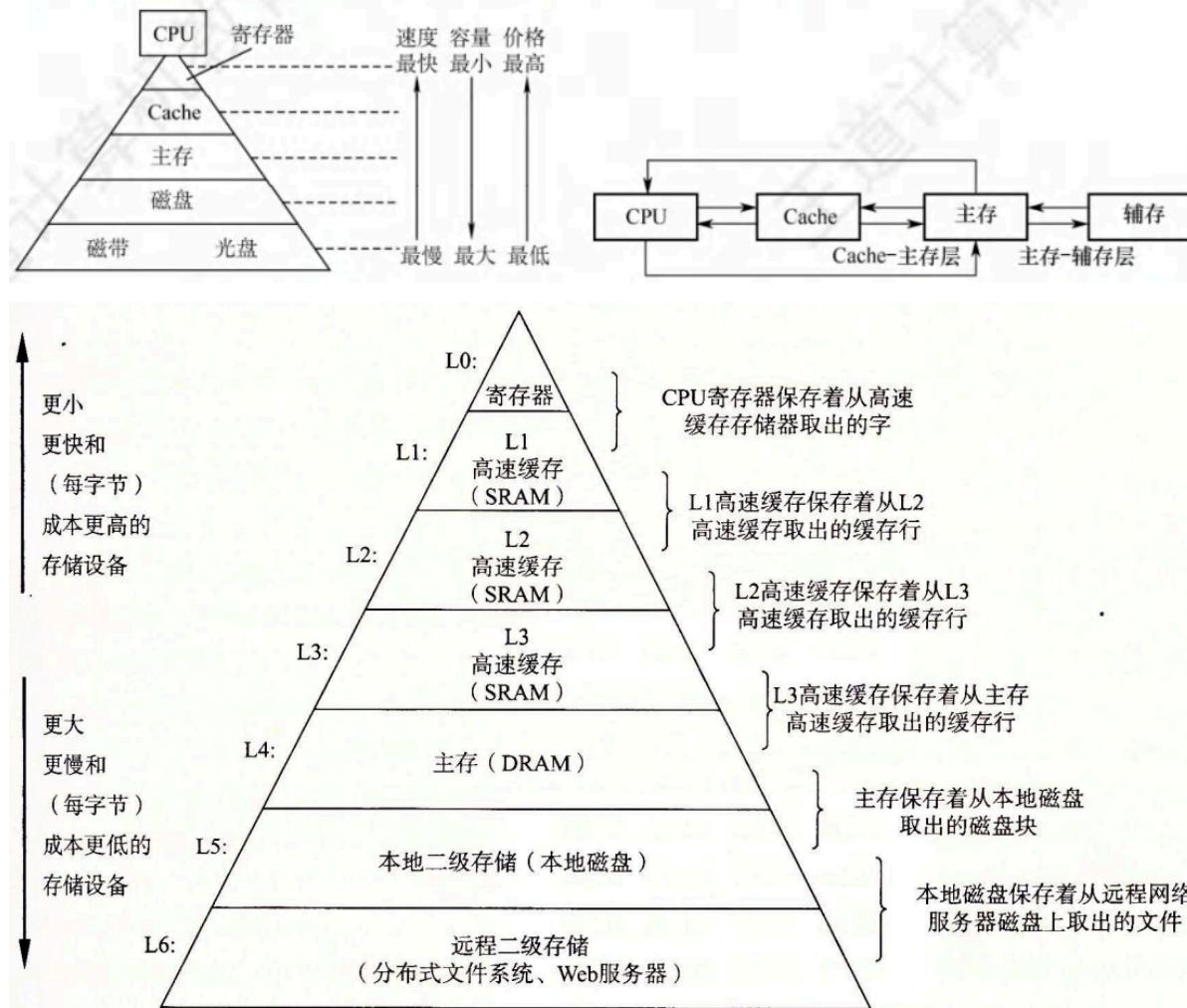
- 结构体的首地址必须是最大元素字节数的整数倍
- 结构体中每个数据的地址必须是自身字节数的整数倍
- 结构体的总体长度必须是最大元素字节数的整数倍

三、存储系统

3.1 存储器概述

- 存储器的分类 (按层次分类)

- 高速缓存 (Cache)
- 主存储器 (主存)
 - Cache 和主存可以直接被 CPU 读写
- 辅助存储器 (辅存、外存)



- 存储器的分类 (按介质分类)

- 半导体存储器：主存、Cache
- 磁表面存储器：磁盘、磁带
- 光存储器：光盘

- 存储器的分类 (按存取方式分类)

- 随机存储器 (RAM) : 读写任何一个存储单元所需时间都相同，与存储单元所在的物理位置无关
 - 如内存、Cache
- 只读存储器 (ROM) : 只能随机读出而不能写入
 - 如 CD-ROM
- 串行访问存储器: 读写某个存储单元所需时间与存储单元的物理位置有关

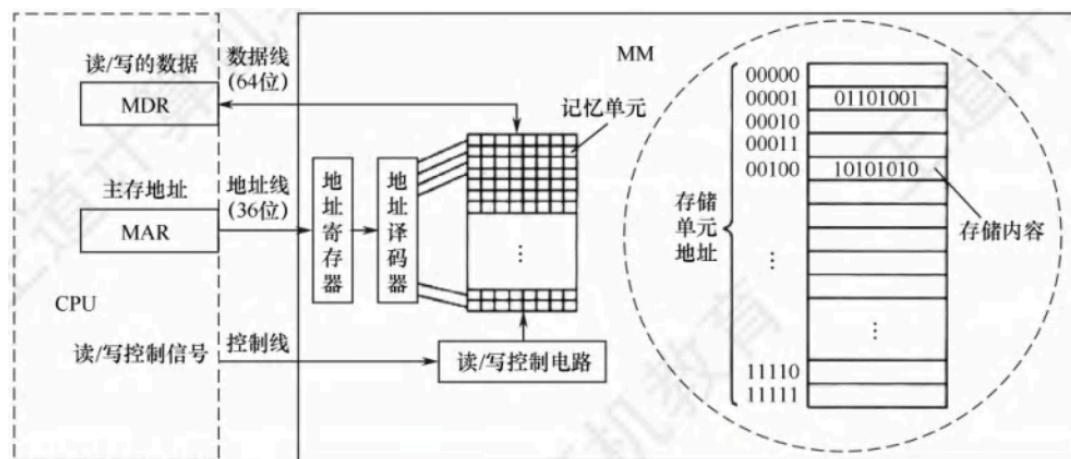
- 顺序存取存储器 (SAM)
- 直接存取存储器 (DAM)
- 相联存储器 (CAM) : 按照内容检索到存储位置进行读写
 - 如快表 (TLB)
- 存储器的分类 (按信息的可保存性分类)
 - 易失性存储器
 - 如 RAM
 - 非易失性存储器
 - 如 ROM, 磁表面存储器、光存储器

• 存储器的性能指标

- 存储容量: 存储字数 \times 字长
- 单位成本: 每位价格 = $\frac{\text{总成本}}{\text{总容量}}$
- 存储速度: 数据传输率 = $\frac{\text{数据的宽度}}{\text{存储周期}}$

3.2 主存储器

- 主存: 存储体、MAR、MDR
- 基本元件: MOS 管、电容



- SRAM 芯片和 DRAM 芯片
 - SRAM, 即静态 RAM, 用于 Cache, 读出后不需要重写, 运行速度快
 - DRAM, 即动态 RAM, 用于主存, 读出后需要重写, 运行速度慢

特点	类型	SRAM	DRAM
存储信息	触发器	电容	
破坏性读出	非	是	
需要刷新	不要	需要	
送行列地址	同时送	分两次送 (复用)	
运行速度	快	慢	
集成度	低	高	
存储成本	高	低	
主要用途	高速缓存	主机内存	

• ROM 芯片的分类

- 掩膜式只读存储器 (MROM)

- 一次可编程只读存储器 (PROM)
- 可擦除可编程只读存储器 (EPROM)
- 闪速存储器 (Flash Memory)
- 固态硬盘 (SSD)
 - 很多 ROM 芯片虽然名字是“只读存储器”，但很多 ROM 也可以写

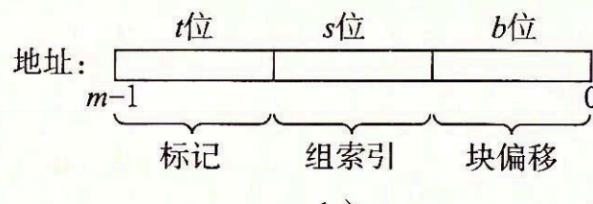
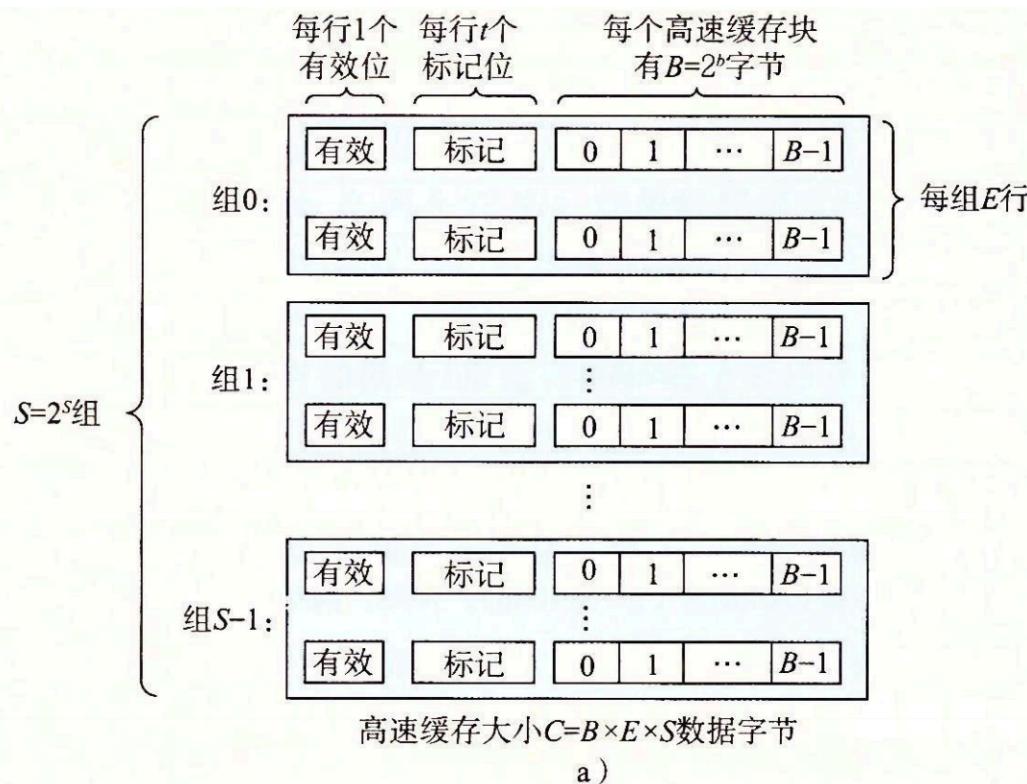
3.3 主存储器与 CPU 的连接

3.4 外部存储器

- 磁盘存储器
 - 组成：磁盘驱动器、磁盘控制器、盘片
 - 存储区域：磁头、柱面、扇区
 - 性能指标
 - 容量
 - 记录密度
 - 平均存取时间：寻道时间 + 旋转延迟时间 + 传输时间
 - 数据传输率
 - 磁盘地址：驱动器号 | 柱面号 | 盘面号 | 扇区号
 - 独立冗余磁盘阵列 (RAID)
- 固态硬盘 (SSD)
 - 原理：基于闪存技术 Flash Memory，属于电可擦除 ROM，即 EEPROM
 - 组成：闪存翻译层、存储介质
 - 与机械硬盘相比的特点：SSD 读写速度快，安静无噪音，但一个“块”被擦除次数过多（重复写一个块）可能会坏掉，而机械硬盘不会因为写的次数太多而坏掉

3.5 高速缓冲存储器 (Cache)

- Cache 工作原理：将某些主存块复制到 Cache 中，缓和 CPU 与主存之间的速度矛盾
- 局部性原理
 - 时间局部性：现在访问的地址，不久之后也很可能被再次访问
 - 空间局部性：现在访问的地址，其附近的地址也很可能即将被访问
- Cache 中存储的信息
 - 有效位 (0/1) + 标记位 + 整块数据
 - 标记位用于指明对应的内存块，不同的映射方式，标记的位数不同
 - 高速缓存的结构可以用元组 (S, E, B, m) 来表示，Cache 的大小 $C = S \times E \times B$



基本参数	
参数	描述
$S=2^s$	组数
E	每个组的行数
$B=2^b$	块大小（字节）
$m=\log_2(M)$	(主存)物理地址位数

衍生出来的量	
参数	描述
$M=2^m$	内存地址的最大数量
$s=\log_2(S)$	组索引位数量
$b=\log_2(B)$	块偏移位数量
$t=m-(s+b)$	标记位数量
$C=B \times E \times S$	不包括像有效位和标记位这样开销的高速缓存大小（字节）

• Cache 和主存的映射方式

- **全相联映射**: 主存块可以放到 Cache 的任意位置
 - **直接映射**: 每个组只有一行, 主存块只能放到特定的某个 Cache 行, 行号 = 主存块号 % 总行数
 - **组相连映射**: 主存块可以放到特定分组中的任意位置, 所属组号 = 主存块号 % 总组数

- Cache 替换算法

- 随机算法 (RAND)
- 先进先出算法 (FIFO)
- 最近最少使用算法 (LRU) ★: 将最久没有被访问过的主存块替换
- 最近不经常使用算法 (LFU) : 将被访问次数最少的主存块替换
- Cache 写策略 (Cache 一致性问题)
 - 写命中
 - 全写法: 当 CPU 对 Cache 写命中时, 必须把数据同时写入 Cache 和主存
 - 回写法: 当 CPU 对 Cache 写命中时, 只修改 Cache 的内容, 而不立即写入主存。只有当此块被换出时才写回主存
 - 写不命中
 - 写分配法: 当 CPU 对 Cache 写不命中时, 把主存中的块调入 Cache, 在 Cache 中修改
 - 非写分配法: 当 CPU 对 Cache 写不命中时, 只写入主存, 不调入 Cache
- 现代计算机通常采用多级 Cache 结构, 各级 Cache 间常采用“全写法 + 非写分配法”, Cache 和主存间常采用“回写法 + 写分配法”

四、指令系统

4.3 程序的机器级代码表示

- 汇编代码格式
 - AT&T 格式 (CSAPP 所使用的汇编代码格式)
 - 源操作数在左, 目的操作数在右
 - 主要适用于 Unix, Linux 操作系统
 - Intel 格式
 - 源操作数在右, 目的操作数在左
 - 主要适用于 Windows 操作系统

4.4 CISC 和 RISC 的基本概念

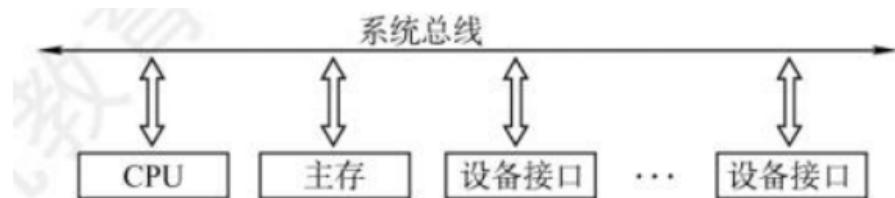
- 指令集架构
 - CISC (复杂指令集计算机) : 一条指令完成一个复杂的基本功能
 - 代表: Intel 芯片的 x86 架构
 - RISC (精简指令集计算机) : 一条指令完成一个基本动作, 多条指令完成一个复杂的基本功能
 - 代表: ARM 芯片的 ARM 架构

五、中央处理器

六、总线

6.1 总线概述

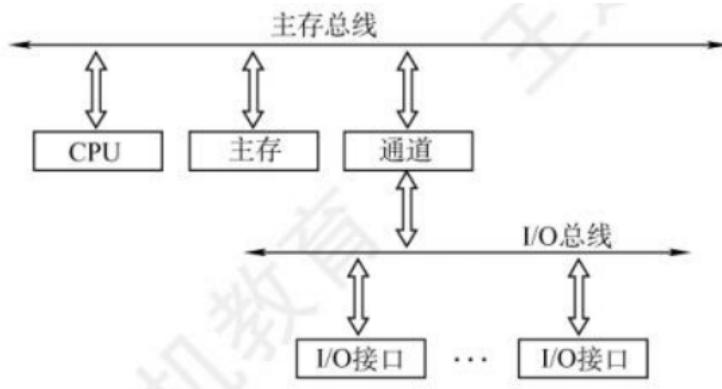
- **总线**: 一组能为多个部件分时共享的公共信息传送线路
- **总线的特点**
 - **分时**: 同一时刻只允许有一个部件向总线发送信息，如果系统中有多个部件，则它们只能分时地向总线发送信息
 - **共享**: 总线上可以挂接多个部件，各个部件之间互相交换的信息都可以通过这组线路分时共享
- **总线的分类**
 - 按数据传输方式分类
 - **串行总线**: 只有一条双向传输或两条单向传输的数据线，适合长距离通信，效率低于并行总线
 - **并行总线**: 有多条双向传输的数据线，适合近距离通信，效率高于串行总线
 - 按功能层次分类
 - **片内总线**: 芯片内部的总线，用于 CPU 芯片内部各寄存器之间及寄存器与 ALU 的连接
 - **系统总线 ***: 计算机系统内各功能部件之间相互连接的总线，按传输信息内容的不同，又可分为**数据总线**、**地址总线**、**控制总线**
 - **I/O 总线**
 - **通信总线**: 计算机系统之间或计算机系统与其他系统之间传输信息的总线
 - 按时序控制方式分类
 - 同步总线
 - 异步总线
- **系统总线的结构**
 - **单总线结构**
 - CPU、主存、I/O 设备都连接在**一组总线上**
 - 优点: 结构简单，成本低，易于接入新的设备
 - 缺点: 带宽低，负载重，多个部件只能争用唯一的总线，且不支持并行传送操作



单总线并不是指只有一根信号线，系统总线按传送信息的不同可以细分为地址总线、数据总线、控制总线

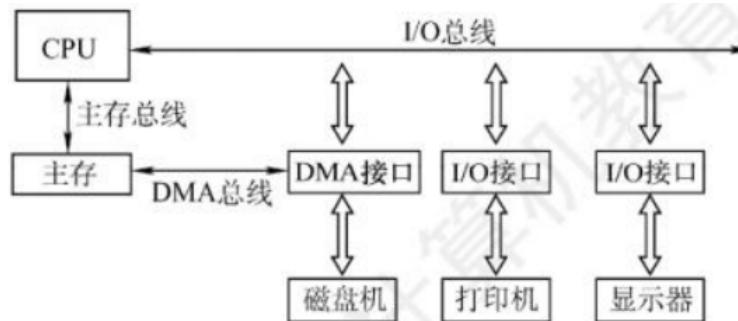
◦ **双总线结构**

- 有两条总线，一条是**主存总线**，用于 CPU、主存和通道之间进行数据传送；另一条是**I/O 总线**，用于多个外部设备与通道之间的数据传送



- **三总线结构**

- 在计算机系统各部件之间采用三条各自独立的总线来构成信息通路，分别是**主存总线**、**I/O 总线**和**直接内存访问 (DMA) 总线**



- **总线的性能指标**

- 总线带宽
- 总线宽度
- 总线工作频率
-

- **总线标准**: 国际上公布的互联各个模块的标准

- 系统总线标准: ISA、EISA、FBS
- 局部总线标准: VESA、PCI、PCIe
- 设备总线标准: USB、PCMCIA

6.2 总线事务和定时

- **总线事务**

- 请求阶段
- 仲裁阶段
- 寻址阶段
- 传输阶段
- 释放阶段

- **总线定时**: 总线在双方交换数据的过程中需要时间上配合关系的控制（实质是一种协议或规则）

- 同步定时方式
- 异步定时方式: 不互锁方式、半互锁方式、全互锁方式
- 半同步定时方式
- 分离式定时方式

七、输入/输出系统
