# 第一章 计算机系统概述

## 计算机发展历程（考纲已删除）

1. 第一代计算机：逻辑元件采用电子管；使用机器语言编程；主存容量小；大、慢、贵
2. 第二代计算机：逻辑元件采用晶体管；使用高级语言编程（面向过程编程），有了操作系统雏形
3. 第三代计算机：逻辑元件采用中小规模逻辑电路；半导体存储器；操作系统、高级语言发展迅速，开始有分时系统
4. 第四代计算机：逻辑元件采用大规模集成电路和超大规模集成电路；产生微处理器；有了并行、流水线、高速缓存、虚拟存储器概念
5. 微型计算机的发展以**微处理器**技术为标志。
6. 目前计算机所有信息任为二进制方式表示，是由物理器件性能决定。
7. 计算机**元件**的更新换代：
   1. 摩尔定律：集成电路上可容纳的晶体管数量
   2. 半导体存储器发展
   3. 微处理器的发展
8. 电子计算机可分为：电子模拟计算机（连续）和电子数字计算机（离散）
9. 数字计算机按用途可分为专用计算机和通用计算机
10. 通用计算机又可以分为：巨型机、大型机、中型机、小型机、微型机、单片机六类
11. 计算机按指令和数据流还可分为
    1. SISD：传统冯诺依曼体系结构
    2. SIMD：整列处理器和向量处理器系统
    3. MIMD：多处理器和多计算机系统
12. 计算机发展的两极化：一级是更微型，多用途；另一级是更巨型，高速
13. 语言发展：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 高级语言 | 编译程序 -> | 汇编语言 | 汇编器 -> | 机器语言 |
| JAVA | 本质是助记符 | 二进制代码 |
| A = X | LOAD A, 16 | 000010101 |

1. 机器语言是计算机硬件唯一能够直接执行的语言程序
2. 解释程序：当程序执行时才将源程序翻译成机器语言，且一次只能翻译一行语句，边翻译边执行。
3. 区分汇编和编译（高级语言经过编译程序编译成汇编语言）

## 计算机系统层次结构

1. 某一功能，其既可以用软件实现，又可以用硬件实现，则称为软硬件逻辑等效。
2. **冯诺依曼机的特点**：
3. 五大部件：运算器、存储器、控制器、输入设备、输出设备
4. 指令和数据以同等地位存储在存储器中，并且可以按地址寻访
5. 指令和数据都用二进制代码表示（由电子器件特性决定）
6. 指令由操作码和地址码组成
7. 指令在存储器内按顺序存放，通常指令是按顺序执行的，特定情况可改变顺序
8. 以运算器为中心，输入输出设备通过运算器与存储器传送数据
9. 现代计算机组织结构特点：以存储器为中心，使IO操作尽可能绕过CPU（I/O设备速度与CPU速度差距过大）

4. 计算机功能部件：

输入设备：键盘、鼠标、扫描仪

输出设备；显示器、打印机

**存储器**；由存储单元构成，工作方式为按存储单元的地址进行存储区

MAR存放访存地址；

MDR暂存要从存储器中读或写的信息（默认MAR\MDR在CPU中）

时序控制逻辑用于产生各种操作存储器的时序信号

还有译码器（为MAR编码）、驱动器（驱动存储体读写）

**运算器**：核心为算术逻辑单元ALU，包含若干通用寄存器

（**ACC**累加器, **MQ**乘商寄存器, **X**操作数寄存器, BR基址寄存器）

还有**PSW**程序状态寄存器（标志寄存器），可以判断是否溢出、结尾、非负等

**控制器**：由程序计数器PC、指令寄存器IR、控制单元CU组成，是计算机指挥中心

PC用于存放下一条指令的地址（自动加一）

IR用于存放当前指令，OP(IR)送到CU，Ad(IR)送到MAR取操作数

CU，分析IR中的操作码，发出各种命令序列，

发出的控制信号可以：控制修改PC、控制ALU执行的运算类型、控制对主存

进行是读还是写操作

5. 数据库系统和数据库管理系统：

**数据库管理系统（DMBS）**是位于用户和OS直接的一层数据管理软件，**是系统软件**

数据库系统是指计算机OS引入数据库后的系统，包好数据库管理系统、数据库、数据

库管理员、应用程序

6. 三个级别的语言

机器语言（二进制代码）、汇编语言（用单词缩写代替二进制指令代码）、高级语言

7. 计算机的工作过程：

把程序和数据装入主存、将源程序转化为可执行文件、从可执行文件的首地址开始逐条

执行指令

**从源程序到可执行文件**

在UNIX系统的GCC编译程序中：

① 预处理阶段：预处理器CPP将以字符#开头的命令进行预处理；输出一个.i的文件

② 编译阶段：编译器CCL对预处理活动的源程序进行编译，生成**汇编语言源程序**.s

③ 汇编阶段：汇编器AS 将.s文件**翻译成机器语言**；将指令打包成一个**可重定位目标**

**文件**；hello.o，是一种二进制文件

④ 链接阶段：连接器ld将多个可重定位目标文件和标准库函数合并为可执行文件；

最终生成的可执行文件保存在磁盘上

**指令执行过程**

① 取指令：根据PC取指令到IR。将PC的内容送至MAR送到地址线，同时控制器将

读信号送到读写信号线，主存根据地址线上的地址和读信号，从指定存储单元读出

指令，送到数据线上，MDR从数据线接受指令信息，并传到IR中

② 分析指令：指令译码并送出控制信号。控制器根据IR中的指令**操作码**，生成相应

的控制信号，送到不同的执行部件

③ 执行指令：以取数操作为例。将IR中的指令**地址码**送到MAR，MAR中的内容送到

地址线，同时控制器将读信号送到读写信号线，从主存中读出操作数送至MDR

8. 计算机系统的多级层次结构

|  |  |
| --- | --- |
| 高级语言层 | 面向用户，由各种高级语言编译程序支持和执行 |
| 汇编语言层 | 提供助记符（汇编语言）由汇编语言编译程序支持和执行 |
| 操作系统M2 | 由机器指令和广义指令组成的操作系统程序实现 |
| 传统机器M1（用机器语言的机器） | 由微程序解释机器指令系统 |
| 微程序机器M0（微指令系统） | 硬件层，由机器硬件直接执行微指令 |

9. 存储程序原理：将程序像数据一样存储到计算机主存，然后按其在存储器中的首地址

执行程序的第一条指令，以后就按该程序的规定顺序执行其他指令

## 计算机性能指标

1. **机器字长**：计算机进行一次定点整数运算所能处理的二进制数据的位数，通常与CPU

的寄存器位数、加法器有关。机器字长一般等于内部寄存器的大小，为字节的整数倍

2. 数据通路带宽：数据总线一次能并行传送信息的位数

3. 主存容量：主存所能存储信息的最大容量，通常以字节衡量，或者用字数（存储单元个

数）\* 字长（一个存储单元有多少比特位）；MAR的位数反映存储单元的个数（可寻址

范围的最大值）

4. 运算速度

① 吞吐量：单位时间内处理请求的数量（主要取决于主存的存取周期-数据/指令能多

块从内存取出或存入）

② 响应时间：从用户想计算机发送一个请求，到系统对该请求作出相应并给出结果的

总等待时间（包括了CPU时间和等待时间（用于IO、访内外存、OS开销等））

③ **主频**（CPU时钟频率）：机器主时钟频率，频率越高执行一个步骤耗时越短

④ CPU**时钟周期**（节拍）：CPU最小的时间单位，是主频的倒数，每个动作至少要一

个CPU时钟周期

**⑤ CPI：执行一条指令所需要的时钟周期数**

⑥ CPU执行时间：运行一个程序所花费的时间

⑦ MIPS：每秒执行多少百万条指令

⑧ FLOPS：每秒执行多少次浮点运算

一个MFLOPS（megaFLOPS）等于每秒（10^6）次的浮点运算，

一个GFLOPS（gigaFLOPS）等于每秒（10^9）次的浮点运算，

一个TFLOPS（teraFLOPS）等于每秒（10^12）次的浮点运算，

一个PFLOPS（petaFLOPS）等于每秒（10^15）次的浮点运算，

一个EFLOPS（exaFLOPS）等于每秒（10^18）次的浮点运算，

一个ZFLOPS（zettaFLOPS）等于每秒（10^21）次的浮点运算。

**（看门狗逃跑饿着）**

5. 系列机：使用相同基本指令系统的多个不同型号的计算机

兼容：运行在某种型号的计算机的硬件或软件也可以运行在其他型号的计算机

软件可移植性：软件经过少量修改或者不修改就能运行在另一个系列的计算机中

固件：将程序固定在ROM中组成的部件，是一种具有软件特性的硬件

# 第二章 数据的表示和运算

## 数制和编码

1. 二进制转为八进制：从整数低位开始，三个数画一根线，高位补零，小数部分低位补零

1111000010 . 01101->*00*1 111 000 010 . 011 01*0*

1 7 0 2 . 3 2

2. 二进制转为十六进制：从整数低位开始，四个数画一根线

1111000010 . 01101 -> *00*11 1100 0010 . 0110 1*000*

3 C 2 . 6 8

3. 八进制、十六进制转为二进制：每个数对应3/4个二进制位

4. 任意进制转为十进制：数码与权值相乘再相加

5. 十进制转为任意机制：

整数部分除以基数取余，到零结束

小数部分乘以基数取整，乘积为1.000或满足精度时结束

6. 不是每一个十进制小数都能用二进制小数表示，如0.3

但是每任意一个二进制小数都可以用十进制小数表示

~~7. BCD码：用四位二进制数来表示一个十进制数的编码方式，有6种冗余表示（删除）~~

~~①~~ **~~8421码（最常用）~~**~~：~~

~~有权码，权值分别为8 4 2 1；如8 -> 1000; 9 -> 1001~~

~~当两个数相加超过9（进位）时，要讲运算结果加6（0110）~~**~~修正~~**

~~② 余三码：无权码，在8421码的基础上加(0011)，每个数都“多余3”~~

~~③ 2421码：有权码，权值从高到低分别为 2 4 2 1；~~

~~特点是大于等于5的二进制数中最高位是1，小于5的最高位是0；~~

~~如 5 –> 1011~~

8. ASCII码：7位二进制编码，共128位字符

连续存放的顺序：A到Z，… ，a到z

如果按八位存放，左端添0

9. 汉字编码：输入编码（用于输入）、汉字内码（内部处理）、汉字字形码（输出）

区位码：两字节表示一个汉字；也即四位十进制数，前两位是区码，后两位是位码

国标码：将十进制的区位码转换为十六进制，再在每字节上加上20H；

两字节的最高位都是0（ASCII码的最高位也是0），为了区分中英文，所以

国标码最高位取1来区分

10. 校验码：能够发现或者能够自动纠正错位的数据彪马

码距：任意两个合法码字之间最少变化的二进制位数；（要变几位才能从合法变成合法）

码距越大，检错、纠错（码距不小于2）能力越强；且检错能力总是大于等于纠错能力

~~11. 三种常见校验码：~~

~~①~~ **~~奇偶校验码~~**~~：在原编码上加一根校验位，使整个数中的”1”的个数为奇/偶数个~~

~~码距为2，~~**~~只有检错无纠错功能~~**

~~②~~ **~~CRC循环冗余码~~**~~（N，K码）：~~

~~将信息码加K个零（生成多项次的最高次），再与生成多项式做模二除法，生成校~~

~~验码，附在信息码后；接收端利用生成多项式对接受到的数做模二除法~~

~~得到的余数是0，则无错，是010就是从右到左（包括校验码）第二个数出错~~

~~能否纠错取决于校验位的位数，(7,4)码就能纠错；~~**~~能发现并纠正一到多位错~~**

~~速度比海明码快，适合大量数据校验~~

~~模二除法：余数第一位为1，则商1，余数第一位为0，则商0~~

~~③~~ **~~海明码~~**~~：本质是多重奇偶校验码~~

* ~~确定海明码的位数~~

~~，n为信息位位数、k为校验位位数~~

* ~~确定校验位分布~~

~~第i个校验位放在第位上~~

* ~~分组~~

~~所以由第一个校验码检验的有：放在3,5,7 三个信息位~~

~~第二个校验码检验的有：放在3,6,7 三个信息位~~

~~第三个校验码检验的有：放在5,6,7 三个信息位~~

* ~~校验位取值~~

~~每组对应的信息位全部求异或（相同为0，不同为1）~~

* ~~校验~~

~~用校验位和所有参与形成该校验位的信息位进行异或生成~~

**~~具有一位纠错能力~~**

## 定点数的表示和运算

1. 无符号数和有符号数

2. 机器数的定点表示

定点小数：约定小数点的位置在符号位之后（隐含）；

x0（符号位）=0，x1到xn均为1取最大正数

x0（符号位）=1，x1到xn均为0取最小负数

定点整数：小数点位置在数值部分最低位之后（隐含）

x0（符号位）=0，x1到xn均为1取最大正数

x0（符号位）=0，x1到xn均为1取最小负数

3. **原码：**用机器数的最高位表示该数的符号，其余的各位表示数的绝对值

真值0有两种表示 +0：00000；-0：10000

**原码表示范围： （关于原点对称）**

**补码：**真值小于0时，除符号位按位取反再加1；正数补码等于原码

小数补码表示范围，比原码多表示一个-1：1。0000

整数补码表示范围，比原码多表示一个：1，0000

真值0只有一种表示方法 0.0000

缺点：不好比较大小

**反码：用于从原码求补码或者补码求原码的中间过渡**

对于正数数值位不变、对于负数数值位取反

真值0有两种表示 +0：0 0000；-0：1 1111

反码表示范围与原码一样

**移码：只表示正数，在真值X上加一个偏置 （字长为n+1）**

即

真值0表示唯一 :100…000（n个0）

一个真值的移码和补码仅差一个符号位

表示范围

移码保持了数据原有的大小，便于比较

**4. 算术移位（有符号数）**

正数无论左移还是右移都是添0

对于负数：原码添0；反码添1；补码左移添0，右移添1（左0右1）

因为补码从右到左第一个”1”开始，左边和反码相同，右边和原码相同

符号位不参与移动，右移空位补符号位，左移空位补0

**逻辑移位（无符号数）：全部添0**

循环左移ROL(Rotate Left)：移出的位不仅要进入CF，而且还要填补空出的位。

带进位的循环左移RCL：用原CF的值填补空出的位，移出的位再进入CF。

5. **补码加减法**：

①转换为x+y的形式；②计算[x]补+[y]补

补码的运算结果仍为补码

6. **溢出**：运算结果超过了数的表示范围

**上溢：大于机器所能表示的最大正数；下溢：小于机器所能表示的最小负数**

发生情况：两个符号相同的数相加或两个符号相异的数相减才可能产生溢出

判断方法：

① 一位符号位（用逻辑表达式来表达上面的情况）

② 双符号位

数据符号：00表示正号，11表示负号；

结果的符号位为01时，称为上溢；为10时，称为下溢

③ 采用一位符号位根据进位情况判断

**最高位进位 异或 次高位进位；结果为0即无溢出；结果为1即有溢出**

**就是符号位和数值位最高位的进位只有一个有，那就是有溢出**

7. 符号扩展

正数扩展：所有附加位填零

负数扩展：对于**补码整数添1，小数添0**；对于反码，全部用1填充

8. 原码定点数加减法

加法：符号位相同：做加法，绝对值相加，符号位不变

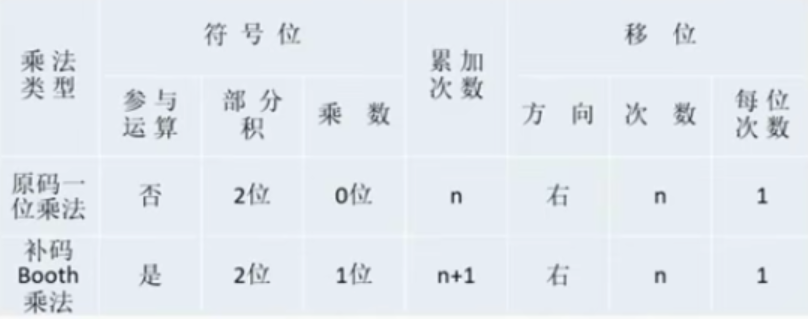
符号位不同：做减法，绝对值大的数减绝对值小的数，符号位与绝对值大的相同

减法：将减数符号位取反，再做加法

9. 定点数的乘法运算

原码一位乘法

补码一位乘法（booth算法）



10. 定点数的除法运算

原码除法运算（不恢复余数法）

①先算出x、y的绝对值、除数|y|绝对值的补码、负的除数的绝对值-|y|的补码

②被除数减去除数即 ；

若结果为正，商上1，左移，再减去除数

若结果为负，商上0，左移，再加上除数

补码除法运算

被除数和除数同号，则被除数减去除数；异号则被除数加上除数

余数和除数同号，商1，余数左移一位，减去除数

余数和除数异号，商0，余数左移一位，加上除数



**11. 有符号数和无符号数的转换**

**无符号数用真值存储（无正负号），有符号数一般用补码表示**

强制类型转换会保持每位不变，仅改变解释位的方式

**12. 不同字长整数之间的转换**

大字长转小字长：把多余的高位字长部分直接截断，低位直接赋值

短字长转长字长：对应位位值相等，**高位部分扩展为原数字的符号位**

char（8位）转int，高位补0

int和float型占4B，short型占2B，double型占8B

13. **大端方式**和**小端方式**存储：

大端方式：从最高位到最低位存储（数据的左边先存）

小端方式：从最低位到最高位存储（数据的右边先存）

阅读小端方式存储的机器代码，字节是按相反顺序显示的

14. 数据按**边界对齐**方式存储：

假设存储字长为32位，可按字节（8位）寻址，半字寻址（16位），字寻址（16）

**半字地址一定是2的整数倍、字地址一定是4的整数倍**

**无论是字节、半字还是字，均可一次访存取出**

（如果不按边界对齐，班子长或者字长的指令可能会存储在多个存储字中）

本质是一种空间换时间的方法——对齐方式取指令时间相同，可以适应指令流水

## 浮点数表示和运算

1. 浮点数的表示格式

阶符 + 阶码的数值部分 + 数符 + 尾数的数值部分

尾数的位数n反映了浮点数的精度

阶符合阶码的数值反映了浮点数的表示范围、小数点的实际位置

2. 规格化浮点数：使浮点数的尾数部分最高位是是1

左规：尾数算术左移一位、阶码减1

右规：尾数算术右移一位、阶码加一（通常发生在浮点数运算时尾数溢出，即双符号位

位01或10时）

规格化后的原码：正数0.1xxxxx；负数 1.1xxxxx

规格化后的补码：正数0.1xxxxx；负数 1.0xxxxx（补码要求符号位和第一位尾数不同）

3. 浮点数的加减运算：

① 对阶：小阶向大阶对齐的原则，**将阶码小的位数向右移**，阶加1

② 尾数求和：尾数按定点数加减规则运算

③ 规格化：尾数大于零时，补码规格化形式为00.1xxx；

小于零时，补码规格化形式为11.0xxx；

**规格化规定位数的绝对值应该≥1/R（R为基数），并≤1，当基数为4时，**

**尾数的绝对值应该大于等于1/4，尾数若用原码表示，则小数点后两位不**

**全为0位规格化数**

**对于基数为4的原码尾数，每右移两位，则阶码加1**

④ 舍入：

* 0舍1入法：右移时被移去的最高位为0则舍去，为1则在尾数末尾加1
* 恒置1法：

⑤ 溢出判断

4. **IEEE754标准**

第一位：尾数的数符（代表数的正负）

第二位：阶码（用**移码**表示，**偏置取127(不是128)**全1表示无穷，全0表示非数）

第三位：尾数数值位（用**原码**表示，同时隐含最高位1；xxxx表示1xxxx）



短浮点数（单精度、float型、32位、4B）

长浮点数（双精度、double型、64位、8B）

其他规定（短浮点数为例）：

* 阶码是0、尾数是0，则真值为0
* 阶码是0、尾数非0，则为非规格化的数
* 阶码全1、尾数非0，真值为‘NaN’（表示不是一个数，除以0时）
* 阶码全1、尾数为0，真值为正无穷（符号位为0）或负无穷（符号位为1）

**浮点数溢出：**

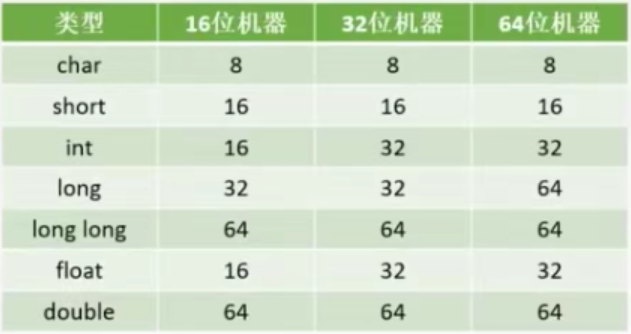
① 阶码上溢出：阶码超过了最大允许值，若结果为正数，置为正无穷

② 阶码下溢出：负指数比最小允许值还小，机器置为0

③ 尾数溢出：右规操作

④ 非规格化尾数：（最高位不是1），需要左规

5. 类型转换



**范围、精度从小到大**，转换过程没有损失：（通常是低精度转换为高精度）

long和double一起运算，会转换为double（int 和float同理）

char -> int -> long ->double

float -> double

**int和float型的转换**：

int 表示整数，有效数字是32位，float表示整数及浮点数，有效数字是23+1 24位

可以看到，float 有效数字少（精度低），但是可表示范围广

int -> float 可能损失精度（数太大就要用移位舍入）

float -> int 可能溢出（int无法表达float那么大的数），也可能损失精度（小数部分）

6. IEEE的表示范围

注意float型阶码表示的真值范围为1-127，给出阶码要减01111..11才为真值

其中阶码为0和阶码为1是预留位

最小正数：数符位取0，阶码取1（真值为1-127 = -126），尾数取全0

## 算术逻辑单元ALU

1. 运算器：承担了执行各种**算术和逻辑运算**的工作

由ALU（算术逻辑单元）、累加器、状态寄存器和通用寄存器组等组成

2. ALU的基本功能：四则运算、逻辑运算、移位、求补

3. 一位全加器（FA）是最基本的加法单元

逻辑表达式（三个输入、两个输出）



4. 串行加法器：只有一个全加器，数据逐位串行送入加法器中运算

操作数长n位，则要分n次进行加法，每次产生一位和，而且串行送回寄存器

进位触发器用来寄存进位信号，参与每次加法运算

优点：成本低、器件少（用于某系低速用运算器）、缺点：速度慢

5. **并行加法器：由n位（与机器字长相同）全加器组成，实现两个n位操作数相加**

虽然可以并行相加，但是每一位取决于前一位的进位，所以并行加法器的最长运算时间

由进位信号的传递时间决定的

**串行进位（行波进位）：**

n个全加器串联起来，每级进位直接依赖于前一级进位

位数越多延迟越长

**并行进位（先行进位）：**

所有的进位仅有最低位输入和其他位输入信号决定，不依赖相邻位的进位输入信号

进位快速，且与字长无关——CLA加法器

单级先行进位方式（组间串行、组内并行的进位方式）

多级先行进位方式（组间并行、组内并行的进位方式，组与组之间用CLA电路连接）

6. ALU的基本结构

有两个输入信号；控制信号（指令译码产生并接入，决定进行何种操作）；输出信号

# 第三章 存储系统

## 存储器概述

1. 存储器按**作用/层次**进行分类

- 主存储器（内存、**主存**），CPU可以**随机访问**；也可以和cache、辅存交换数据

- 辅助存储器（**辅存**、外存）；不能与CPU直接交换信息，属于外设

- 高速缓冲存储器（cache）；位于主存和CPU之间，cache的存取速度与CPU相匹

配，以便CPU能高速使用程序段和数据

2. 存储器按**存储介质**分类

- 磁表面存储器（磁盘、磁带）

- 半导体存储器（MOS、U盘）

- 光存储器（光盘）

**3. 按存取方式分类**

- **随机存储器（RAM）**：存储器内任何一个单元都可以**随机存取**，且与存取时间与物

理位置无关；RAM又分为**静态RAM**（触发器原理）、和**动态RAM**（电容充电原理）

**可以随机存取的不一定是随机存取存储器（ROM）**

**属于易失性存储器（断电信息即消失）**

- **只读存储器（ROM）**：存储器的内容只能随机读出而不能写入（ROM和RAM都是

**随机存取**），但是只读的概念已经没有保留，仍然保留了断电后内容保留的特性

**属于非易失性存储器**

CD-ROM不是ROM

- 串行访问存储器：需要**按物理位置的先后顺序寻址**，有顺序存取存储器**（磁带）**

和直接存取存储器**（磁盘）**

4. 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

- 破坏性读出（存储单元的信息被读出时，原存储信息会被破坏），必须接一个再生

的操作，恢复被破坏的信息

- 非破坏性读出

5. 存储器的性能指标

- 存储容量 = 存储字数（存储器地址空间大小） \* 存储单元容量

1B(Byte, 字节) = 8b(bit, 位)

- 单位成本 = 每位价格 = 总成本/ 总容量

- 存储速度：数据传输率 = 数据的宽度/ 存储周期

**存取时间**：从启动一次存储器操作到完成该操作所经历的总时间，分为读出时间和

写入时间

**存取周期（读写周期、访问周期）：两次独立访问存储器操作所需要最小时间间隔**

主存带宽（数据传输率）：每秒从主存进出信息的最大数量，单位为字或位每秒

**Cache系统效率：访问Cache的时间/平均访存时间**

**有效存取时间** = 平均存取时间

6. 相联存储器：把存储单元所存内容的某一部分（key）作为检索项，去检索该存储器；

所以这种存储器是**按内容或者地址进行寻址**，价格较为昂贵

应用：TLB、相联cache

7. **EPROM**是一组浮栅晶体管，被一个提供比电子电路中常用电压更高电压的电子器件分

别编程。一旦编程完成后，EPROM只能用强紫外线照射来擦除

**EEPROM**是指带电可擦可编程**只读存储器**。是一种掉电后数据不丢失的存储芯片。

EEPROM 可以在电脑上或专用设备上擦除已有信息，重新编程。一般用在即插即用

## 存储器的层次化结构

1. 多级存储系统体现在：“Cache---主存”层次和“主存---辅存”层次

2. Cache---主存：主要解决CPU和主存速度不匹配的问题

主存---辅存：主要解决主存容量问题

3. CPU---寄存器---Cache---主存---磁盘---磁带、光盘（速度、容量、价格关系）

## 半导体随机存储器

1. 主存储器由DRAM实现，靠处理器的那一层cache由SRAM实现（都是易失性存储器）

ROM则属于非易失性存储器

2. SRAM（静态随机存储器）：存储元用双稳态触发器（六晶体管MOS-6个逻辑电路）

属于非破坏性读出

特点：**存取速度最快**、集成度低、一般用来组成高速缓冲存储器

3. DRAM（动态随机存储器）：基本存储元为一个晶体管（集成度高、成本便宜）

DRAM采用**地址服用技术，地址线是原来 1/2**，**地址信号分行、列两次传送**

特点：**存取速度比SRAM慢但便宜、容易集成，一般用来组成大容量主存系统**

DRAM电容上的电荷只能维持1到2ms，需要刷新；**刷新周期**一般为2ms

4. 刷新方式：

**集中刷新：**在一个刷新周期内，利用一段固定时间，对所有存储器行进行刷新

这个过程中停止对存储器的读写操作（死区）

**分散刷新：**每行的刷新分散在各个工作周期中；存储器系统工作周期分为两部分

前半部分用于正常读写或保持、后半部分用于刷新某行

**无死区（就他没有）**；但是增加了系统的存取周期，降低了整机速度

**异步刷新：**将刷新请求分摊到一个周期中

缩短了死区、减少了刷新次数，根本上提升了整机的工作效率

**（仍有死时间）**

5. 刷新的注意点：

- **刷新对CPU透明，即刷新不依赖外部访问**

- 动态RAM刷新单位是**行**，也即刷新时**只需要行地址**

- SRAM和DRAM都是易失性存储器（断电后数据丢失）；但是需要**刷新的只有DRAM**

**6. SRAM 和DRAM比较**

|  |  |  |
| --- | --- | --- |
|  | SRAM | DRAM |
| 存储信息元件 | 触发器 | 电容 |
| 破坏性读出 | 不是 | 是 |
| 需要刷新 | 不需要 | 需要 |
| 送行列地址 | 一次送 | 分两次送（地址线复用） |
| 运行速度 | 快 | 慢 |
| 集成度 | 低 | 高 |
| 发热量 | 高 | 低 |
| 存储成本 | 高 | 低 |
| 用于 | **高速缓存** | 主机内存（ROM+DRAM） |

7. 只读存储器（ROM）

支持随机存取（ROM和RAM一样）；结构简单，位密度高；**非易失性存储器**

根据制造工艺可分为 MROM（掩模式只读存储器）、一次可编程只读存储器（PROM）

可擦除可编程只读存储器（EPROM）、**闪存**（Flash Memory）、固态硬盘（SSD）

8. 主存储器（MM）的基本组成

存储矩阵（存储体，由存储元件构成）

线：数据线、地址线（单向）、控制线（一或两根**读写控制线**、一根**片选线**）

**数据线的宽度和MDR宽度相同、地址线的宽度和MAR的宽度相同**

地址线的位数决定了主存地址空间的最大可寻址范围

数据线和地址线共同反映了存储体容量的大小

## 主存储器和CPU的连接

1. 单个存储芯片的容量有限，在字数或者字长方面与实际要求有差距

需要在**字**或者**位**上两方面进行扩充才能满足要求

**- 位扩展法（增加存储字长）**

将多个存储芯片的地址端、片选端和读写控制端相应并联；数据端分别引出

在某一时刻可以选中所有的芯片，所以片选信号CS需要连接所有芯片

8片8K\*1位的RAM芯片可以组成8K\*8位的存储器

**- 字扩展法（增加存储字数，位数不变）**

将芯片的地址线、数据线、读写控制线相应并联；而片选线来区分各个芯片

4片16K\*8位的RAM芯片可以组成64K\*8位存储器，其中地址线中的两根用来区

分不同的芯片（00 01 10 11对应不同四片）

- 字位同时扩展法

片选线先经过片选器、再分组并联

2. **片选和字选：**CPU对存储单元访问，首先需要选择存储芯片（片选）、再从选中的芯片

中选择相应的存储单元（字选）；片选信号可以由线选法和译码片选法产生

3. **线选法：**每次寻址时只能有一位有效，不允许多位有效；n个芯片就要n根线来区分

优点：不需要地址译码器、线路简单

缺点：**地址空间不连续（01直接到10了，没有00和11的地址空间）**，且地址线需要

分时为低电平，否则不能工作，**不能充分利用存储空间**，给资源造成浪费

4. **译码片选法：**通过地址译码器产生片选信号，n条线可以产生对应power(2,n)的信号

5. 存储器与CPU连接

① 选择合理的存储芯片

对芯片类型（ROM还是RAM）以及数量的选择

通常用ROM存放系统程序、标准子程序和常数；RAM为用户编程而设置

② 地址线的连接

通常将CPU地址线的低位和存储芯片的地址线相连；高位用来扩充芯片使用

③ 数据线的连接

CPU的数据线可能和芯片数据线数量不等；

相等时可以直接连接；不等就要对位进行扩充

④ 读写命令线的连接

只有一根时：通常高电平为读，低电平为写

有两根时，读为RD，写为WE，均为低电平有效（0有效）

⑤ 片选线的连接

哪一片芯片被选中取决于该存储芯片的片选控制端CS能否接受到来自CPU的片

选有效信号；其中片选有效信号与访存控制信号MREQ（低电平有效）相关

当CPU访问I/O时，MREQ为高，则不要求存储器工作

## 双端口RAM和多模块存储器

1. 为了**提高CPU访问存储器的速度**，可以采用**双端口存储器（空间并行）**、**多模块存储器**

**（时间并行）**等技术

2. **双端口RAM：**存储器左右有两个独立端口；**分别具有两组相互独立的地址线、数据线、**

**读写控制线，允许两个控制器同时异步访问存储单元**

出现错误的情况：

两个端口同时对一个地址单元写入数据（写入错误）

两个端口同时对同一地址操作，一个写，一个读（读出错误）

解决方法：

**设置忙信号BUSY**，BUSY为0时，判断逻辑会暂时关闭一个端口，该端口不能访问

**3. 多模块存储器（一个存储单元存储多个字）**

**单体多字存储器**：存储器中只有一个存储体，每个存储单元存储m个字，总线宽度也

为m个字，一次并行读出m个字，地址必须顺序排列并处于同一存储单元

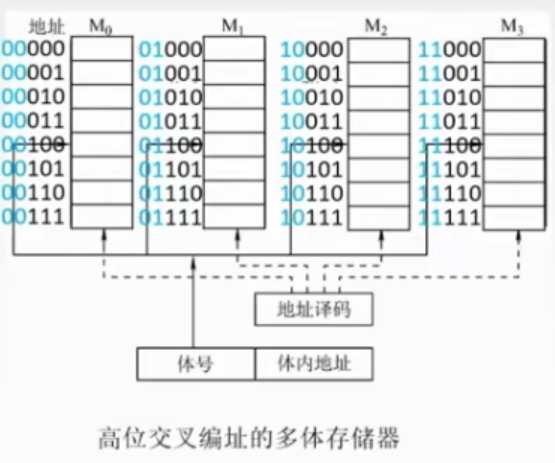
优点：一个存取周期可以取出m个指令，增大了带宽

缺点：指令和数据在内存中必须连续存放，才能同时读出

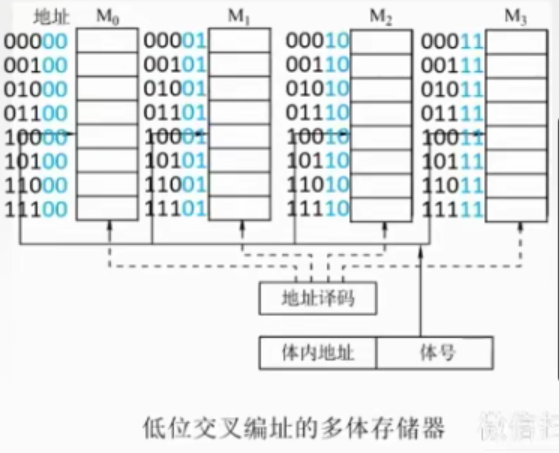
**多体并行存储器：**由多体模块组成，每个模块都有相同的容量和存取速度、独立的读写

控制电路、地址寄存器、数据寄存器；他们**可以并行工作、交叉工作**

- 高位交叉编址：高位地址表示体号、低位地址表示体内地址



- 低位交叉编址：**低位地址为体号，高位地址为体内地址**



- 两种编址方式对连续空间访问：



总线传输周期：每隔多久启动下一个模块（存储体）

交叉存储体的带宽：存储体的存取速率（字长/总线传输周期）

存储体的个数 大于等于 (存储周期/总线传输周期)

访存冲突：对于四体并行模块，给定的访存地址在相邻的四次访问中出现在同一个存

储模块内

## 高速缓冲存储器

1. 时间局部性：最近未来要用到的信息很可能是现在正在使用的信息（循环）

空间局部性：最近未来要用到的信息，很可能与现在正在使用的信息在存储空间上相邻

2. 对于二维数组的遍历，对数组进行顺序访问速度较快

a[0][0], a[0][1], a[0][2], a[0][3], a[1][0], a[1][1], a[1][2], a[1][3],……

3. **Cache：**通常由**SRAM构成**，Cache和主存都被划分成相等的块

命中：访存地址在cache中，将地址转换为Cache地址，直接对cache进行读操作

若执行的是写操作，还需要按照一定的写回策略（全写法、写回法）处理

不命中：访存，并把此字所在的块一次性地从主存调入Cache

若Cache已满，则用替换算法；

CPU与Cache之间的数据交换：以**字**为单位

Cache与主存之间的数据交换则以Cache**块**为单位

4. Cache命中率：CPU想要访问的东西在Cache中的比率，

与Cache的容量，块长有关，计算看书（126）

计算：

① Cache命中率

② Cache-主存系统平均访问时间

③ 效率

④ 带宽

5. **Cache和主存的映射方式**

- **直接映射**（对号入座）

在cache中的块号就是主存块号%cache总块数

直接映射的**地址结构**为：

标记（mod后第几个组） + Cache行号（组里的第几块） + 块内地址

CPU访存流程：先根据中间c位（cache行号），找到cache中的对应行，比较给

出的标记位和cache中的标记位是否相等（与要访问的是同一块），

如果相等则命中

**缺点**：冲突概率最高、空间利用率最低、即使有很多空但是不能利用

- **全相联映射**（空位随意放，不需要任何替换算法）

标记用来表示块取自主存中的哪一块

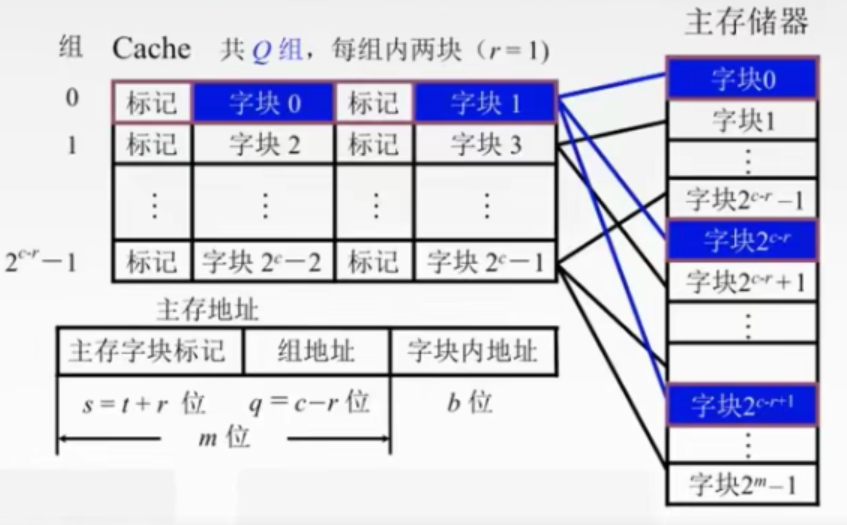
全相联映射的**地址结构**为

标记（块号，长度=主存块号长度） + 块内地址

**优点**：冲突概率低、灵活、空间利用率高、命中率高

缺点：对于标记的比较，速度很慢

- **组相联映射**（按号分组，组内随意放）



组相联映射地址结构：

标记 + 组号 + 块内地址；其中标记和组号合在一起为主存块号

CPU访问流程：

首先根据访存地址中间的组号找到对应的cache组，将组中所有的行的标记与

要访存地址的高位标记进行比较，有相等则为命中

6. Cache中的主存块替代算法

Cache中的空间被占满时，需要使用替换算法进行替换（对于直接映射，主存中的块对

cache中的块是唯一确定的，所以直接映射不用替换算法，直接顶替就行）

常用的算法：随机算法、先进先出、最不经常使用LFU、**最近最少使用LRU**

- 最近最少使用LRU：设置一个计数器

① 命中时，所命中的行的计数器清零，比这个值低的其他计数器加1，其余不变

② 未命中有空闲行，新装入的行的计数器置0，其余行全加1

③ 未命中且无空闲行，计数值最高的被淘汰，新装入的置0，其余全加1

特点：如果只有五行，那么计数值只会是0到4

- 最不经常使用算法：每行设置一个计数器，新行建立后置0，每访问一次被访问行

加1，替换时置换最小的

7. 抖动：对于LRU，访问序列为 1 2 3 4 5 1 2 3 4 5，而cache组只有4行式，命中率为0

**8. Cache写回策略**

**写命中处理方法：**

- **全写法：**CPU对**cache写命中**时，必须把数据**同时写入cache和主存**

**当某块需要替换时，不必把这块写回主存，直接用新块覆盖即可**

缺点：增加了访存次数、降低了cache效率

优点：随时能保存主存数据的正确性

（一般用于Cache – 主存层次——读写时间少，如Cache块置换）

**写缓冲：**为了**减少全写法直接写入主存的时间损耗**；CPU同时写数据到cache和写

缓冲，写缓冲在控制写入内存（可以**解决速度不匹配的问题**）

- **写回法：**当CPU对cache写命中时，只修改cache的内容，**换出时才写回主存**

优点：减少了访存次数，但存在不一致的隐患

**每个cache行必须设置一个脏位**，反映次块是否被修改过

（一般用于主存 – 外存层次——读写时间长，如页面置换）

**写未命中处理方法：**

- **写分配法：**加载主存中的块到cache，再更新cache中的块（**与写回法搭配使用**）

缺点：每次不命中都需要从主存读块

- **非写分配法：**只写入主存，不调块（**与全写法搭配**）

9. 采用指令Cache和数据Cache分离的主要目的：

Cache分为一级L1和二级L2。L1集成在CPU中，称为片内Cache。在L1中还分数据Cache和指令Cache。分别用来存放数据和执行这些数据的指令，**两个Cache可以同时被CPU访问，减少了争用Cache所造成的冲突**，提高了处理器效能。

可以保证不同的指令同时访存

## 虚拟存储器

1. 虚拟存储器：

将主存或者辅存地址空间统一编址，形成庞大的地址空间，可以自由编程而不必在意实

际存放地址；实际在主存单元存放的地址为实地址或者物理地址，编程的地址为虚地址

或者逻辑地址

过程：CPU使用虚拟地址时，由辅助硬件找出虚地址和实地址之间的映射关系，判断是

否在主存中；如果在主存，直接访问主存；如果不在主存，把该页或段调入主存

并访问

**实地址 = 物理页号 + 页内地址号**

**虚地址 = 虚拟页号 + 页内地址号**

**辅存地址 = 磁盘号 + 盘面号 + 磁道号 + 扇区号**

2. **页式虚拟存储器**

以页为基本单位，主存为实页，虚存为虚页

**过程**： ①页表基址寄存器存放了页表的首地址

②虚拟地址中的虚拟页号指出了页表中的页表项

③判断有效位、脏位、引用位进行不同区分操作

④若可以操作，读取页表项对应的物理页号

⑤得到物理页号，再加上虚地址中的页内地址，为实地址

脏位：页式虚拟存储使用回写策略，脏位用于判断是否需要协会磁盘

引用位：用于替换策略，如FIFO，LRU位等

有效位/装入位：表示对应页面是否在主存中

**优点**：页表长度固定，易于分配空间、调入方便

**缺点：页不是逻辑上独立实体，不好处理、保护、共享**；同时程序最后一页会浪费空间

3. 快表（TLB）

放在**主存中的页表为慢表**

放在**高速缓冲器的页表为快表**（是慢表中很少的一部分）

访问过程看书，OS里也有

4. 段式虚拟存储器

段是按逻辑结构划分的，每个段的长度不一样

段表中要给出各段的起始地址、段的长度

**优点**：段的分界由逻辑独立性，易于编译、管理、修改、保护

**缺点**：段长度可变，分配空间容易留下碎片

5. 段页式虚拟存储器

把程序按逻辑结构分段，每段再分成固定大小的页；程序的调入调出仍以页为基本传送

单位；每个程序对应一份段表，每段对应一个页表

虚地址为：段号 + 段内页号 + 页内地址

优点：兼备段式、页式虚拟存储的优点

缺点：地址变换过程需要查两次表（先查段表、再查页表），开销大

6. cache和虚拟存储器的比较

不同点：

① Cache主要是为了解决系统速度不匹配的问题，而虚拟存储器是为了提高主存容量

② Cache由硬件实现，对程序员、OS透明；而虚拟存储器由OS和硬件同时实现，

对系统程序员不透明，对应用程序员透明

③ 虚拟存储器在未命中时对系统性能影响较大

④ CPU与cache和主存都有直接访问的通路；而辅存和CPU没有直接通路

也就是说在cache未命中时，主存能和CPU直接通信

而虚拟存储器未命中时，只能现有硬盘调入主存，而不能从硬盘直接通信

**7. Cache缺失处理由硬件完成**

**缺页处理由软件完成（缺页异常处理程序）**

**TLB缺失既可以用硬件也可以用软件处理**

# 第四章 指令系统

## 指令格式

1. 指令：计算机执行某种操作的命令，是计算机执行某种操作的命令，**是计算机运行的最**

**小功能单位**；所有的指令构成指令系统，也即指令集

程序控制类指令：**用于改变程序的执行顺序**（条件转移、无条件转移、循环指令、子程

序调用和返回指令等）

2. 指令的基本格式：一般包含操作码字段、地址码字段

操作码：OP，识别指令、指出寻址方式、使用方法和操作数类型

地址码：AD，给出被操作的信息（指令或者数据）的地址

定长指令字结构：所有的指令长度相等（执行速度快、控制简单）

变长指令字结构：长度因功能而异 （增加指令译码和分析难度，控制器设计更加复杂）

3. 地址指令格式：

- **零地址指令**：只有操作码OP

使用零地址指令的两种情况：

① 不需要操作数的指令：空操作指令、停机指令、关中断指令等

② 运算类指令，参与运算的**两个操作数从堆栈中依次弹出**，送到运算器运算，再

将结果隐含压入栈中（仅用在堆栈计算机中）

- **一地址指令**：OP + A1

① OP（A1）-> A1：操作码含义可以是加1,、求反、求补等

② 另一个地址由ACC提供，运算地址也放在ACC中

（ACC）OP（A1）-> ACC

- **二地址指令**：OP + A1 + A2

目的操作数地址还用于保存本次运算结果的地址

- **三地址指令**：OP + A1 + A2 + A3

（A1）OP（A2）-> A3

完成三地址指令一共需要四次访存（取指、取两个操作数、存结果）

- **四地址指令**：OP + A1 + A2 + A3 + A4

（A1）OP（A2）-> A3 ;A4为下一条要执行的指令的地址

4. 扩展操作码（边长操作码）：

目的：为了保持指令字长度不变而增加指令数量

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | OP | A1 | A2 | A3 |
| 四位操作码 | 0000  0001  …  1110 | A1 | A2 | A3 |
| 八位操作码 | **1111** | 0000  0001  …  1110 | A2 | A3 |
| 十二位操作码 | **1111** | **1111** | 0000  0001  …  1110 | A3 |
| 十六位操作码 | **1111** | **1111** | **1111** | 0001  0002  …  1111 |

通常情况下，**对使用频率高的指令分配较短操作码，频率低的使用较长操作码**

可以减少指令的译码和分析时间

## 指令的寻址方式

1. 寻址方式：指寻找指令或者操作数有效地址的方式，也就是确定本条指令的数据地址

和下一条待执行指令地址的方法；分为指令寻址和数据寻址

2. 指令寻址

① 顺序寻址：PC + 1，生成下一条指令的地址

② 跳跃寻址：转移指令实现，跳跃的结果是当前的指令修改PC的值

**两种方式指令的地址都是由程序计数器PC给出**

3. 数据寻址：在指令中定义寻找数据的方式

指令格式为：操作码 + 寻址特征 + 形式地址A

- **隐含寻址：**

在指令中隐含操作数地址，如ACC对单指令格式来说就是隐含寻址

优点：**有利于缩短指令字长**

缺点：需要增加存储操作数或隐含地址的硬件

- **立即数寻址：**

给出的操作数不是地址，而是数本身（**立即数，补码方式存放**）

优点：不访问主存，执行时间最短

缺点：A的位数限制了立即数的范围

- **直接寻址：**

指令字中的形式地址A就是操作数的地址

优点：简单直接，指令在执行阶段仅访问一次主存

缺点：A的位数决定了操作数的寻址范围，操作数的地址不易修改

- **间接寻址：**

指令的地址字段给出的形式地址不是操作数的真正地址，而是操作数有效地址所在

的存储单元的地址（存放操作数地址的地址）EA = （A）

间接寻址可以是一次间接寻址，也可以是多次间接寻址

主存第一位是1时：取出的仍然不是操作数地址

直到第一位是0时，取出的就是操作数的地址，再找一次就是操作数

优点：可以扩大寻址范围

缺点：需要多次访存（一次间接寻址需要两次访存）

**易于完成子程序返回**

- **寄存器寻址：**

直接给出操作数所在的寄存器编号

优点：不需要访问主存，只用访问寄存器

缺点：寄存器昂贵

- **寄存器间接寻址：**

寄存器中存放的是操作数所在主存单元的地址（比一般间接寻址快、但也需要访存）

- **相对寻址：**

程序计数器PC的内容加上指令格式中的形式地址A形成操作数的有效地址

EA = (PC) + A；A是当前指令地址的位移量，补码表示，可正可负

广泛应用于转移指令

**用于控制程序的执行顺序、转移**

- **基址寻址：**

CPU中的基址寄存器（BR）的内容加上指令格式中的形式地址A形成的操作数的

有效地址 EA = （BR）+ A

其中基址寄存器可以是专用寄存器，也可以是通用寄存器；**是面向系统的**，内容由

OS确定，主要用于解决程序逻辑空间与存储器物理空间的无关性

优点：可以扩大寻址范围（基址寄存器的位数大于形式地址A的位数）、

**有利于多道程序设计、可用于编制浮动程序，但偏移量A的位数较短**

- **变址寻址：**

有效地址EA等于指令字中的形式地址A与变址寄存器IX的内容之和

EA = （IX）+ A；**变址寄存器是面向用户**的，主要用于处理**数组**问题，变址寄存器

的内容由用户设定，在**循环**过程中可以使A为数组首地址，不断改变IX的内容就

可编制循环程序遍历数组

**可以有效处理数组、循环问题**

- **堆栈寻址：**

堆栈是存储器或者专用寄存器组中的一块特定存储区，该存储区地址由特定寄存器

给出（堆栈指针SP）

寄存器堆栈：硬堆栈

主存中的一段区域作为堆栈：软堆栈

## CISC 和 RISC

1. 指令系统的发展

① 增强原有指令的功能，设置更为复杂的新指令：CISC（X86架构）

② 减少指令种类和简化指令功能，提高指令的执行速度：RISC（ARM、MIPS架构）

CISC：complex instruction set computer – **电脑、台式机**

RISC：reduced instruction set computer – **ARM 平板、手机（并行加流水线）**

**2. CISC主要特点：**

- 指令**数目**多

- **指令长度**不固定；**格式**多；**寻址方式**多

- 可以访存的指令不受限制

- 各种指令**频率**相差大（80%的语句仅用20%的指令）

- 指令**执行时间**相差大，大部分指令需要**多个时钟周期**才能完成

- 难以用优化编译的方式生成高效的目标代码程序

**3. RISC主要特点：**

- 选取了使用频率最高的指令，复杂指令由简单指令组合来实现

- **指令长度**固定；**格式**种类少；**寻址方式**种类少

- **只有Load、Store（取数存数指令）访存**，其余操作都在寄存器之间进行

- CPU中**通用寄存器数量**非常多

- 一定采用**流水线**方式，大部分指令在**一个时钟周期**完成

- 以**硬布线**为主，不用或少用微程序控制

- 特别重视编译优化工作，以减少程序执行时间

# 第五章 中央存储器

## CPU的功能和结构

1. 中央存储器由运算器和控制器组成
2. **CPU的功能**
   1. 指令控制：完成**取指令、分析指令、执行指令**的操作，即**程序的顺序控制**
   2. 操作控制：管理并产生每条指令的**操作信号**
   3. 时间控制：对各种操作加以时间控制，为每条指令按**时间顺序**提供控制信号
   4. 数据加工：**对数据进行算术、逻辑运算**
   5. 中断处理：计算机运行过程中出现的**异常情况、特殊请求**进行处理
3. 运算器：对数据进行加工和处理
   1. 算术逻辑单元：进行算术和逻辑运算
   2. 暂存寄存器：暂存从主存读来的数据，对程序员透明
   3. 累加寄存器：是一个通用寄存器，可以暂时存放ALU运算结果，作为加法输入端
   4. 通用寄存器组：存放操作数（源、中间、结果）和各种地址，对程序员可见
   5. **程序状态字寄存器**：各种状态信息（运算有无借位、溢出），参与微操作的形成
   6. 移位器：进行移位运算
   7. 计数器：控制乘除步骤的操作次数
4. 控制器：协调**控制**各部件执行程序的指令序列（**控制取、分析、执行指令**）
   1. 程序计数器：下一条指令在主存中的地址，可与MUX配合（决定指令执行顺序）
   2. 指令寄存器：保存当前正在执行的指令（IR）
   3. 指令译码器：仅对操作字段进行译码OP(X)
   4. 存储器地址寄存器：存放要访问的主存单元地址
   5. 存储器数据寄存器：存放从主存读出、写入的数据
   6. 时序系统：产生各种时序信号，由统一时钟（CLOCK）分频得到
   7. 微操作信号发生器：根据IR的指令、PSW的状态信息、时序信号，产生控制信号
5. 对**程序员透明的寄存器**有：MAR MDR IR
6. 对**程序员可见的寄存器**有：程序计数器PC、通用寄存器组、程序状态字寄存器PSW
7. 控制器的工作原理：根据指令的操作码、指令的微命令序列、条件信号形成各部件要用到的控制信号
8. CPU中的专用寄存器：PC、IR、MDR、MAR、PSW

9. CPU的位数 = CPU一次能处理数据的位数 = 数据总线线数

## 指令执行过程

1. **指令周期：**CPU从主存去除并执行一条指令的时间，不同指令的指令周期可能不同
2. **机器周期：**一条指令可以划分成多个机器周期，每个机器周期完成一个基本操作（如取指令、存储器写等），也称CPU周期，通常为一个总线事务访存或I/O时间（因为访问一次存储器的时间固定的，所以视内存中读取一个指令字的最短时间作为机器周期）
3. **时钟周期：**也称节拍或T周期，是CPU操作的基本单位

在一个时钟周期内，CPU仅完成一个最基本的动作

指令流水线每一条流水段时间单位对应一个时钟周期

1. **一个指令周期用若干机器周期表示，一个机器周期又由若干时钟周期（节拍）构成**
2. 完整的指令周期包括：**取指周期、间指周期、执行周期、中断周期**

取指周期：根据PC的值从主存取指令并放在IR里；

间指周期：根据IR中的地址位Ad(IR)取有效数地址；

执行周期：取操作数；

中断周期：保存程序断点

1. 为了区分不同的工作周期，CPU设置的4个标志触发器FE、IND、EX、INT，分别对应取指、间指、执行、中断周期，“1”状态表示有效，由1->FE等信号控制
2. 中断周期中的进栈操作是SP-1，计算机的堆栈都是向低地址增加
3. 取值周期的数据流：

|  |  |  |
| --- | --- | --- |
| 1 | PC -> MAR -> 地址总线 -> 主存 | (PC) -> MAR |
| 2 | CPU发送控制信号 -> 控制总线 ->主存 | 1 -> R |
| 3 | 主存 -> 数据总线 -> MDR -> IR（存放指令） | M(MAR)->MDR  (MDR) -> IR |
| 4 | CU发出读命令-> PC内容加一 | (PC) +1 -> PC |

（空指令只有取值周期）、取指操作是**控制器自动进行**的

1. 间指周期的数据流

|  |  |  |
| --- | --- | --- |
| 1 | AD(IR)或MDR -> MAR -> 地址总线 ->主存 |  |
| 2 | CU发出读命令 -> 控制总线 -> 主存 |  |
| 3 | 主存 -> 数据总线 -> MDR（存放有效地址） |  |
| 4 | 将有效地址送至指令的地址码字段（可能有这步） |  |

1. 执行周期的数据流

根据IR中 指令的操作码和操作数通过ALU产生不同结果，不同指令的执行周期操作不同，没有统一的数据流向

1. 中断周期的数据流

|  |  |  |
| --- | --- | --- |
| 1 | CU控制SP-1, SP->MAR->地址总线->主存 |  |
| 2 | CU发出写命令-> 控制总线-> 主存 | 1->W |
| 3 | PC-> MDR-> 数据总线-> 主存（保存断点至主存） | (PC) -> MDR |
| 4 | CU（中断服务程序的入口地址）-> PC |  |

完成工作：

保存断点（先修改SP栈顶指针再存入数据）、形成中断服务程序入口地址、硬件关中断

1. 指令执行方案有单指令周期、多指令周期、流水线方案。

**单指令周期：**所有指令选用相同的执行时间（固定的时钟周期）完成，指令之间串行，因此指令周期取决于执行时间最长的指令的执行时间

**多指令周期：**指令之间仍串行执行（下一条要在上一条执行完才可开始），但可为不同的指令分配不同时钟周期完成指令

**流水线方案：**指令之间并行，多条指令同时运行

## 数据通路的功能和基本结构

1. 数据通路：数据在功能部件之间传送的路径；

**即各个子系统通过数据总线形成的数据传送路径为数据通路**

1. 数据通路部件：路径上的部件，如ALU、通用寄存器、状态寄存器。数据通路描述了信息从何处开始，中间经过什么部件，最后传送到哪个寄存器。

数据通路由控制部件控制。

1. 数据通路的基本结构有1）CPU内部单总线方式。2）CPU内部三总线方式。3）专用数据通路方式 （**区分CPU内部总线和系统总线，内部总线是CPU内部数据交换的通路**）
2. CPU内部单总线：连接数据传送存在较多**冲突**现象，性能低

连接各个部件的总线只有一条，为单总线结构；如果有多条就是多总线结构

1. 专用数据通路方式：根据指令执行过程的数据和地址流动方向设置专门线路，性能高
2. 数据通路图中，各部件用大写字母表示，字母加in表示该部件允许输入控制信号，out表示该部件允许输出控制信号
3. 寄存器之间的数据传送

把PC的内容送至MAR：

|  |  |  |
| --- | --- | --- |
| 1 | (PC) -> Bus | PCout有效，PC内容送至总线 |
| 2 | Bus -> MAR | MARin有效，总线内容送至MAR |

1. 主存与CPU之间的数据传送

CPU从主存读取指令，再把指令送至IR寄存器

|  |  |  |
| --- | --- | --- |
| 1 | PC -> Bus -> MAR | PCout和MARin有效，指令地址送至MAR |
| 2 | 1 -> R | CU发出读命令（系统总线） |
| 3 | MEM(MAR) -> MDR | MDRinE有效，此时MDRinE为向系统总线允许输入信号（要区分系统总线） |
| 4 | MDR -> Bus -> IR | MDRout和IRin有效，指令存入IR |

1. 执行算数或逻辑运算

ALU本身是没有存储功能的组合电路，要执行家法运算，相加的两个数必须同时在ALU两个输入端同时有效；暂存器Y用于暂存一个操作数

现将一个操作数从主存中读入Y（ALU左端），另一个（通常保存在ACC中）送到ALU右端，结果暂存至Z

|  |  |  |
| --- | --- | --- |
| 1 | AD(IR) -> Bus -> MAR | MDRout和MARin有效，此时IR与MDR内容一样，但是IR通常没有out控制信号 |
| 2 | 1 -> R | CU发出读命令（经过控制总线） |
| 3 | MEM -> 数据线 -> MDR | 操作数从存储器经过数据总线存入MDR |
| 4 | MDR -> Bus -> Y | MDRout和Yin有效，操作数->Y |
| 5 | (ACC) + (Y) -> Z | ACCout和ALUin有效，CU向ALU发送加法命令，结果暂存至Z |
| 6 | Z -> ACC | Zout 和 ACCin有效，结果存至ACC |

## 控制器的功能和工作原理

1. 控制器：控制整个系统工作

2. 控制器与其他部件的主要连接关系：

- 运算器部件通过**数据总线**与内存、输入输出设备传送数据

- 输入输出设备通过**接口电路**与总线相连

- 内存储器、输入输出设备从地址总线接受地址信息，从个控制总线得到控制信号，

通过数据总线与其他部件传送数据

- 控制器部件从数据总线接受指令信息，送出指令地址到地址总线，还向各个部件 发出运行时所需要的控制信号

3. **控制器的主要功能（掌握）：**

**- 从主存中取出一条指令，并指出下一条指令在主存中的位置**

**- 对指令进行译码，产生相应的控制信号**

**- 指挥并控制CPU、主存、输入输出设备之间的数据流动方向**

4. 控制器的类别：分为**硬布线控制器**和**微程序控制器**。两类控制器的PC和IR是相同的；

但**确定和表示指令执行步骤的办法**和**给出各种各个部件的控制信号方案是不同的**

5. 硬布线控制器（了解）

基本原理：根据指令要求、当前时序、内外部状态，按**时间顺序**发送一系列微操作控制

信号；**由组合逻辑门电路（对应时序电路）和触发器构成**

CU**输入信号**来源：

① 经过**指令译码器译码产生的指令信息**

② 时序系统产生的**机器周期信号和节拍信号**（目的：使控制单元按一定的先后顺

序、一定的节奏发出咯咯控制信号）

③ 来自执行单元的反馈信息即**标志**，即依赖**CPU所处的状态**产生控制执行（比

如BAN指令-条件转移指令：需要判断结果是否为负进而决定到底转移与否）

④ 来自系统控制总线的控制信号（中断请求、DMA请求、INTR信号等）

CU**输出信号**：

① CPU内部控制信号

（寄存器之间的数据传输、PC的修改、控制ALU进行相应运算）

② 到控制总线的控制信号

到存储器：访问控制、读命令、写命令；

到IO设备：IO信号；

中断响应信号、总线响应信号

CPU的控制方式：

① 同步控制方式（选最长最繁琐的微操作作为标准，作为统一时钟信号）

优点：控制电路简单；缺点：速度慢

② 异步控制方式（各个部件按自身固有的速度工作，通过**应答**方式进行联络）

优点：运行速度快；缺点控制电路比较复杂

③ 联合控制方式：为各种不同的指令的微操作做实行大部分同步控制、小部分采

用异步控制的方法

硬布线控制单元设计步骤

① 列出微操作命令的操作时间表

② 进行微操作信号综合

③ 画出微操作命令的逻辑图

6. 微程序控制器

思想： 把**微操作信号代码化，使每条机器指令转化为一段微程序并存入一个专门的存储器中**（**控制存储器CM，在CPU内部，ROM**），微操作信号由微指令产生

微命令：（构成控制序列的最小单位）是控制部件向执行部件发出的各种控制命令

微操作：微操作是微命令的执行过程，与微操作一一对应（微操作是微命令的执行过程）

微命令的相容性与互斥性：相容性微命令指可以同一时间产生、执行的一系列微命令

微指令：若干微命令的集合

微地址：存放微指令的控制存储器（ROM）的单元地址

①操作控制字段（微操作码字段）：用于产生某一步操作需要的各种控制信号

②顺序控制字段（微地址码字段）：控制下一条要执行的微指令地址

微周期：从控制存储器中读取一条微指令并执行相应微操作所需要的时间

微程序：一个微程序是微指令的有序集合，一条指令的功能由一段微程序来实现

微程序控制器的**组成**：

① 控制存储器：核心部件，用于存放各个指令对应的微程序，ROM

② 微指令寄存器：CMDR，用于存放从CM中取出的微指令，位数与微指令字长相等

③ 微地址形成部件：用于产生初始微地址和后继微地址，以保证微指令的连续执行

④ 微地址寄存器：接受微地址形成部件送来的微地址，为CM读取微指令做准备

**若指令系统有n中机器指令，则控制存储器中的微程序数至少是n+1条，含一条公共**

**的取值微程序**

微程序控制器**工作过程**：

P229

微指令的编码方式：

直接编码方式：微指令中每一位都代表一个微命令；每个微指令用或者不用一个微

命令，只要将对应位设置0或者1；**后面接下地址**（下一条微指令

地址）

字段直接编码方式：互斥性微命令分在同一段内，相容性微命令分在不同段内

000..00表示保留，不操作；**后面接下地址**

**微指令的地址形成方式**：

① 下地址字段指出（断定方式）

② 根据**机器指令**的操作码经过微地址形成部件形成

微指令的格式：

① **水平型微指令：**一条水平型微指令定义并执行几种并行的基本操作

优点：微程序短执行**速度快**；缺点：**微指令长，编写微程序麻烦**

② **垂直型微指令：**一条垂直型微指令只能定义并执行一种基本操作

优点：**微指令短、简单规整**；缺点：

③ 混合型微指令：在垂直的基础上加了一些不复杂的并行操作

**7. 硬布线和微程序比较**

|  |  |  |
| --- | --- | --- |
|  | 微程序控制器 | 硬布线控制器 |
| 工作原理 | **微操作控制信号以微程序的形式存放在CM中**，执行时读出即可 | 微操作控制信号**由组合逻辑电路根据当前指令码、状态、时序，即时产生** |
| 执行速度 | 慢 | 快 |
| 规整性 | 较为规整 | 繁琐、不规整 |
| 应用场合 | CISC  需要指令修改 | RISC  需要高速 |
| 易扩充性 | 易于扩充 | 困难 |

## 指令流水线

1. 指令流水线：将指令的执行过程分成多个子过程，每个子过程并行执行

分为三个阶段：**取指**（根据PC内容到主存，取出指令送到IR）

**分析**（对指令操作译码，按照给定方式寻找指令中的有效地址并取数）

**执行**（根据操作码字段，完成指定功能）

多条指令在处理器中执行，可采用的方式：

① 顺序执行（串行，前一条指令执行完才启动下一条）：T = 3nt（n条指令）

② 一次重叠执行方式（执行和下一条取值重叠）：T = （1+2n）t

③ 二次重叠执行方式（分析、执行和下一条的取值、分析重叠）

2. 流水线方式的特点：

- 一个任务分为多个子任务，每个子任务由专门功能部件并行工作完成，缩短时间

- 每个功能段的部件都有一个缓冲存储器，用于保存本段结果给下一段流水段使用

- 流水线中的每一段时间尽量相等，否则会堵塞、断流

- 流水线处理的必须是连续任务

- 流水线需要有装入时间（第一个任务进入流水线到输出流水线的时间）和

排空时间（最后一个任务进入流水线到输出流水线的时间）

3. 流水线的分类

① 按照流水线使用级别不同

**部件功能**级流水线：

将复杂的算术逻辑运算组成流水线工作方式

如：将浮点加法操作分成求阶差、对阶、尾数相加、结果规格化四个子过程

**处理机**级流水线：

把**一条指令解释过程分成多个子过程（如取值、译码、执行、访存、写回）**

② 按完成的功能

单功能流水线：只能实现一种固定的专门功能流水线

多功能流水线：通过各段间不同连接方式可以同时/不同时实现多种功能的流水线

③ 按同一时间各段直接的连接方式

动态流水线：某些段在实现某种运算时，另一些段正在进行另一种运算

静态流水线：流水线的各段只能按同一种功能的连接方式工作

④ 是否有反馈信号

线性：无反馈信号；非线性：有反馈回路，适合线性递归运算

**4. 影响流水线的因素**

**① 数据冲突（数据冒险）：一条指令要用到上一条的结果**

解决方法：- 数据相关的指令**暂停**几个时钟周期（硬件阻塞或者软件插入空指令）

- 设置相关专用通路（**数据旁路技术**）

- **调整指令顺序**

**② 控制冲突（控制冒险）：执行转移、调用或者返回指令时修改PC导致断流**

解决方法：- 分支预测，尽早生成转移目标地址

- 预取转移成功和不成功两个控制流方向上的目标指令

流水线按序流动，一定不会发生读后写相关WAR和写后写相关WRW

只会出现没有等到上一条指令写入，则当前指令就去读的错误

流水线按序发射按序完成是指要求下一条指令的取指阶段和上一条指令的译码阶段并行，避免因上一条指令发生冲突而下一条指令先完成。

5. 流水线的性能指标（P245）

吞吐率：单位时间内流水线完成任务的数量

加速比：不使用流水线和使用流水线所用的时间之比

效率：完成n个任务占用的时空区的有效面积

6. 超标量流水线技术：每个时钟周期可并发多条独立指令，以并行的方式将两条或者多条

指令编译并执行（需要配置多个功能部件）

特点：不能调整指令的执行顺序，需要挖掘更多的指令并行性

7. 超流水线技术：一个时钟周期内一个功能部件使用多次

8. 超长指令字：将多条能并行操作的指令组合成一条具有多个操作码字段的超长指令字

# 第六章 总线

## 总线概述

1. 总线的定义：一组能为个部件**分时共享**的公共信息传送线路

**计算机使用总线结构便于增减外设、同时提高信息传输的并行性**

分时：同一时刻只允许由一个部件向总线发送信息，可能多个部件同时从总线上接受相同信息

共享：总线上可以挂多个部件用于交换信息

2. 总线设备：分为**主设备和从设备**

主设备：获得总线控制权的设备；

从设备：被主设备访问的设备，只能相应主设备发来的各种总线命令

3. 总线特性：机械特性（尺寸形状）、电气特性（传输方向、有效电平范围）

4. 猝发传输方式：一个总线周期内（CPU完成一次访问MEM或I/O端口操作所需要的时间）传输存储地址连续的多个数据自的总线传输方式

一次传输一个地址和一个数据的称为常规传输；

一次传输一个地址和 一批数据的称为**猝发传输**

5. 总线的分类（按功能划分）

- 片内总线：芯片内部的总线，连接CPU内部各个部件的公共连接线

- 系统总线：连接计算机系统内各个功能部件之间的总线

分为数据总线（双向）、地址总线（单向）、控制总线

- 通信总线：连接计算机与计算机之间的总线

6. **系统总线结构：**

- 单总线结构：CPU、内存、IO接口全部挂在一族总线上

缺点：负载重、带宽低；

优点：成本低、易于接入新的设备

- 双总线结构：CPU、主存和通道之间有一条主存总线；

另一条是I/O总线，用于多个外部设备和通道之间传输数据

- 三总线结构：主存总线、I/O总线、直接内存访问（DMA）总线

DMA总线用于在内存和高速外设之间直接传输数据

优点：提高了系统吞吐量、IO工作性能；

## ~~总线仲裁（已删）~~

~~1. 总线仲裁：为了解决多个主设备同时竞争总线控制权的问题~~

~~分为集中仲裁方式、分布仲裁方式~~

~~2， 集中仲裁方式：将所有总线请求集中起来，通过裁决算法决定~~

~~- 链式查询方式：~~

~~- 计数器定时查询：~~

~~- 独立请求方式：~~

~~3. 分布仲裁方式：不需要中央仲裁器，每个潜在的主模块都有自己的仲裁号和仲裁器~~

~~一有请求每个仲裁器就会将信号发在共享的仲裁总线上~~

~~仲裁器将从仲裁总线上得到的仲裁号判断它与自己的优先级，若低总线则不予反应~~

## 总线操作和定时

**在某个总线周期内，只能有一个主设备控制总线，选择一个从设备通信或者进行广播**

1. 总线传输的四个阶段

① 申请分配：要使用总线的设备提出申请，由总线仲裁机构决定使用权分配

② 寻址阶段：取得使用权的主模块通过总线发出要访问的从设备的地址和有关命令

用于启动本次传输的从模块

③ 传输阶段：主模块和从模块进行数据交换

④ 结束阶段：主模块信息从总线上删除，让出使用权

**2. 同步定时方式（同步通信）：**

发送方用时钟前沿发信号；接收方用时钟后沿判断，识别

优点：**传送速度快**，较高传输速率；总线控制逻辑简单

缺点：主从设备属于强制性同步；不能及时检验数据，可靠性差

**3. 异步定时方式：**

没有同一时钟、固定时间间隔，完全依靠双方握手信号来实现定时控制

优点：**能保证两个工作速度相差很大的部件或者设备可靠地进行信息交换**，自适应时间

缺点：速度慢、控制复杂

- 不互锁方式：主发“请求信号”，过一会就撤销；

从收到“请求信号”，发出“回答信号”，过一会就撤销

不存在互锁关系；优点速度快；缺点：发生故障无法发现

- 半互锁方式：主发“请求信号”，必须接到从设备的“回答信号”才会撤销请求信号

从发“回答信号”，不必获知主设备已经撤销请求信号，过一会自动撤

销回答信号；

不存在互锁关系；缺点：主设备可能不断请求

- 全互锁方式：主发出“请求”，必须收到“回答”才撤销请求信号

从发出“回答”，必须获知“请求”被撤销才撤销回答信号

缺点：速度最慢；优点：可靠性最强

## 总线标准

局部总线：传送活动图像的大量数据

设备总线：连接外部设备I/O总线

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ISA** | **系统总线** | **并行（16根）** | **IBM** |  |
| **EISA（扩展ISA）** | **系统总线** | **并行（32根）** |  |  |
| **PCI** | **局部总线** | **并行** | **实现外部设备互联**  **是声卡、显卡、网卡等常用接口** | **半同步总线，猝发式传输、** |
| **AGP** | **局部总线** | **并行** | **连接显卡(图形存储器)** |  |
| **PCI-Express** | **局部总线** | **串行** | **全面取代PCI和AGP** | **支持全双工、热拔插** |
| **USB** | **设备总线** | **串行** |  | **即插即用、热拔插** |
| **FSB、QPI** | **系统总线** | **串行** | **用于连接CPU与北桥** |  |
| **SATA** | **设备总线** | **串行** | **用于连接硬盘** |  |

视屏线传输标准：

VGA：传输模拟信号

CRT：阴极射线管

LCD：液晶显示器

DVI：传输数字信号

HDMI：视屏线，传输声音+图像（高速）

# 第七章 输入输出系统

## I/O设备基本概念

1. 接口：在各个外设和主机之间传输数据时各种协调工作的逻辑部件。

包括在传输过程中进行速度匹配、电平和格式转换等

2. I/O系统分为IO软件和IO硬件

I/O软件：包括**驱动程序、用户程序**等，

通常采用IO指令和通道指令实现CPU与I/O设备的信息交换

I/O硬件：**通过设备控制器来控制I/O具体动作；通过IO接口与主机相连**

3. I/O控制方式：

**① 程序查询方式：**CPU通过程序不断查询I/O设备是否已经做好准备（低速）

**② 程序中断方式：**I/O设备就绪并发出中断，CPU才给予反应（低速）

**③ DMA方式：**主存和I/O设备之间有一条直接数据通路（高速）

**④ 通道方式：**系统中又通道控制部件，每个通道都挂若干外设；通道就是小CPU

## 外部设备

1. 输入设备：键盘、鼠标；

输出设备：显示器（CRT LCD LED）、打印机；

外存储器：硬盘存储器、磁盘阵列（RAID）、光盘存储器（CD-ROM，CD-RW）

2. 磁盘性能指标：

磁盘容量：分为格式化容量（用户可以使用的容量）、非格式化容量（可以利用的

磁化单元总数）

记录密度：道密度（沿磁盘半径方向单位长度上的次倒数）

位密度（磁道单位长度上能记录的二进制代码位数）

面密度（道密度和位密度的乘积）

**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：单位时间内向主机传输数据字节数

**3. RAID（廉价冗余磁盘阵列）将多个独立的物理磁盘组成一个独立的逻辑盘**

物数据在多个物理盘上分割交叉存储、并行访问，有更好的存储性能、可靠性、安全性

① 同时使用**多个磁盘**，提高传输率

② 使用**多个磁盘并行存储**，提高数据吞吐量

③ 使用**镜像功能**，提高安全可靠性

④ 铜鼓**数据校验**，提供容错能力

RAID0 条带化（类似低位交叉多体编址），**无容错无纠错**

RAID1 存两份，**有容错能力无纠错能力**

RAID2-6 **有容错有纠错能力**

## I/O接口

**1. 接口实现的功能**

① 实现**主机和外设的通信联络控制**（解决配合问题，协调速度）

② 进行**地址译码、设备选择**

③ 实现**数据缓冲**（有专门的缓冲寄存器）

④ **信号格式的转换（电平、数据格式）**

⑤ **传送控制命令和状态信息**

2. I/O接口类型

- **并行接口**（**一个字或者一个字节**同时传输）、**串行接口**（**一位**一位传，USB）

这里说的数据传送方式是指接口和外设一侧

**主机和接口一侧总是并行传输的**

- 按主机访问I/O设备控制方式可分为程序查询接口、中断接口、DMA接口

- 根据灵活性可分为可编程接口、不可编程接口

3. **I/O端口**及其编址

I/O端口：接口电路中可被CPU直接访问的**寄存器**，有数据端口、状态端口、控制端口

CPU对数据端口执行读写操作，对状态端口读操作，对控制端口写操作

I/O端口的编址：

**统一编址**（存储器映射方式）：把I/O端口当做存储器单元进行地址分配

区分存储单元和I/O设备的方式：不同的地址码（访存指令可直接访问）

优点：不需要专门的输入输出指令，端口有较大编址空间

缺点：端口占用存储器地址，使内存容量变小

**独立编址**（I/O映射方式）：和主存地址空间是两个独立的地址空间

需要专门的输入输出指令来访问I/O端口

优点：清晰、便于理解

缺点：需要专门的I/O指令来访问I/O端口；增加了控制信号和控制复杂性

## I/O方式（第七章重点）

1. **程序查询方式：**

主机进行I/O操作，先发出寻味信号，读取设备的状态并根据状态（是否准备就绪）决

定下一步是传输数据（通常是**一个字**）还是继续等待

优点：简单、设备少

缺点：CPU有“**踏步现象**”、**CPU与I/O串行**工作，效率极低

2. **程序中断方式：**

程序中断：在执行现有的程序中，突然接受到某些急需处理的异常或者特殊请求，CPU

暂时中止先行程序，转而对该请求进行处理，处理完毕后返回断点

**程序中断的作用：**

- 实现CPU与I/O设备的并行工作

- 处理硬件故障和软件错误

- 实现人机交互，用户干预机器需要中断系统

- 实现多道程序、分时操作；多道程序的切换需要借助中断系统

- 实时处理

- 实现管态、目态的切换（软中断）

- 多处理器系统中各个处理器之间的信息交流和任务切换

3. 程序中断方式的工作流程：

① 中断请求

中断源向CPU发送中断请求信号（INTR触发器置1）

② 中断响应

响应中断的条件（用到中断判优）

③ 中断处理

中段隐指令；中断服务程序

**非屏蔽中断：**是一种硬件中断，**不受中断标志位IF影响，即使在IF=0的情况也会响应**

**可屏蔽中断：**也是硬件中断，受

其中**IF=0表示关中断**，在8088芯片**存放在PSW中、**

4. 中断判优：（中断相应次序）

硬件>软件; 非屏蔽>可屏蔽；DMA请求>I/O请求

高速>低速设备；输入设备>输出设备；实时设备>普通设备

中断判优可由硬件实现（硬件排队器），也可以由软件实现（查询程序）

访管指令的优先级最高

区分：中断响应次序、中断处理次序

中断响应次序是中断优先级（由硬件排队器决定）

中断处理次序是根据优先级以及中断屏蔽状况处理中断的次序（设置中断屏蔽标志）

5. **中断隐指令：（由硬件直接实现，目的是转去执行中断服务程序）**

① 关中断

② 保存断点（保存PC内容至堆栈）

③ 引出中断服务程序（**取出中断服务的入口地址并传送给PC**），

有软件查询、硬件向量（中断向量）的方式

中断向量：系统将全部中断向量集放在一块区域内，该存储区为中断向量表入口地

址表，CPU响应中断后，中断硬件会自动将中断向量地址传到CPU

**也就是说中断向量是中断服务程序的入口地址**

6. 中断处理流程：

① 关中断

② 保存断点（保存PC和PSW）

③ 引出中断服务程序（软件查询、硬件向量法，找中断向量地址）

④ 保存现场和屏蔽字

⑤ 开中断（实现中断嵌套）

⑥ 执行中断服务程序（中断请求的目的）

⑦ 关中断（恢复现场时需要关中断）

⑧ 恢复现场和屏蔽字（中断服务程序搞定）

⑨ 开中断、中断返回（中断服务程序最后一条指令是返回指令，无条件转移指令）

前三步是中断响应，由硬件完成

后面是第二部分，中断程序的执行

7. 多重中断

屏蔽字：每个中断源有一个屏蔽触发器，1表示屏蔽，0表示不屏蔽，所有屏蔽触发器

组合成一个屏蔽字寄存器，屏蔽字寄存器的内容为屏蔽字

1越多优先级越高，比如四个中断源，1111代表最高优先级，每个1对应四

个不同中断源

8. **DMA方式**

DMA方式：**完全由硬件进行成组信息传送的控制方式**，即在外设和内存之间有一条直

接数据通道，适合如磁盘机、磁带机等**大批量数据**的传送情况

**中断仅作用于故障、正常传送结束时**的处理

**DMA方式特点：**

① 使主存与CPU脱钩，主存既可以被CPU访问，也可直接与外设通信

② 数据块传送时，主存地址的确定、传送数据的计数由硬件电路完成

③ 主存中需要开设专用**缓冲区**，缓和速度差异

④ DMA传送速度快，CPU和外设**并行**工作，提高效率

⑤ DMA在传送开设时对程序要进行**预处理**（初始化），结束后要通过中断方式进

行**后处理**

**DMA控制器：**

**主要功能：**

- 传送前准备：

① 接受外设**（DMA控制器）发出的DMA请求**，并向CPU发出**总线请求**

② CPU响应总线请求，发出总线响应信号，接管总线控制权,进入DMA周期

**- 接管传送功能：**

③ 确定传送数据的主存单元地址长度，并自动修改主存地址计数和传送长度

（由**设备驱动程序设置传送参数**）

④ 规定传送方向，并发出读写控制信号，执行传送操作

- 结束后工作：

⑤ 向CPU报告DMA操作结束**（转入中断服务程序完成传输后的处理）**

**主要组成部件**

- 主存地址计数器：存放要交换数据的主存地址

- 传送长度计数器：记录数据的长度，计数溢出时即传送完毕

- 数据缓冲寄存器：暂存数据

- DMA请求触发器：每当I/O准备好数据，就给出一个控制信号使触发器置1

-- 控制/状态逻辑：控制整个过程，指定传送方向，与CPU信号保持协调同步

- 中断机构：每当一个数据块传输完后，触发一个中断请求

**9. DMA传送方式：**

① 停止CPU访问主存

② DMA与CPU交替访存（将CPU周期分为两个周期，硬件复杂）

③ 周期挪用（周期窃取）：看CPU在干嘛，决定接下来总线是给CPU还是给DMA

**优先级是DMA>CPU**

10. **DMA的传送过程**

① **预处理：**

首先CPU执行几条I/O指令，用来测试I/O设备状态，同时向DMA控制器传送寄

存器初值、传送方向，启动设备等，然后CPU继续回去干他的事

直到I/O设备准备好，I/O设备向DMA控制器发送**DMA请求**，再由DMA控制器

向CPU发送**总线请求（也可统称为DMA请求）**

② **数据传送：**

DMA数据传输可以以字或字节为单位，也可以块为单位

对于以块为单位的传送（硬盘），是**通过DMA控制器循环实现**的，不是通过CPU

**也可说数据传送阶段完全由DMA控制器控制**

③ **后处理：**

**DMA控制器向CPU发送中断请求**，CPU执行中断服务程序做处理

包括校验送入主存的数据是否正确、测试传送过程是否有错、是否继续使用DMA

传送其他数据

11. **DMA和中断方式的区别**

① 中断方式是程序的切换，需要保护和恢复现场

**DMA只需要预处理和后处理占用CPU，其他时候不占用CPU**

② 对中断请求的响应**只能发生在每条指令执行周期后**（中断周期）

对DMA请求的响应可以发生在**每个机器周期结束时均可**（只要CPU不占用总线）

③ 中断传送需要CPU干预

**DMA方式不需要CPU干预**，传输率非常高，适合高速设备

④ DMA请求**优先级**高于中断请求

⑤ 中断方式有对异常事件的处理能力，而DMA方式只能用于数据传送

⑥ 中断方式靠程序传送**（访存指令）**

**DMA方式靠硬件传送（DMA控制器）**

# 其他

1. MUX（数据选择器/多路选择器）

数据选择器(data selector) 根据给定的输入地址代码，从一组输入信号中选出指定的一个送至输出端的组合逻辑电路

**工作原理是：**给A1A0一组信号 比如1 0 那么就相当于给了他一个2进制数字2 也就相当于选通了D2这个输入端，这个时候 输出Y 输出的就是D2的信号；D2是什么，Y就输出什么