**目录**

第一章 系统概论 1

1.1 冯诺依曼计算机 1

1.2 计算机的硬件和软件 1

1.3 软件与硬件的逻辑等价性 2

第二章 运算方法和运算器 2

2.1 进位计数制及数制之间的转化 2

2.2 数的编码 2

2.3 各种字符的编码：ASCII码、BCD码 2

2.4 浮点数 3

2.5 IEEE754标准浮点格式 3

2.6 定点补码加、减运算 3

2.7 溢出的概念及判别 3

2.8 一位全加器、行波进位加法器、先行进位加法器：74181 3

2.9 原码一位乘法原理、补码一位乘法原理 3

2.10 定点运算器的结构 4

2.11 浮点数加减运算 5

2.12 浮点数乘、除运算 5

第三章 存储器 6

3.1 存储系统分级结构 6

3.2 存储器存储容量 6

3.3 半导体存储器的分类：RAM、ROM 7

3.4 DRAM的刷新 8

3.5 存储器的扩展 8

3.6 存储器的地址译码 10

3.7 双端口存储器、多模块交叉存储器 12

3.8 Cache的命中率 14

3.9 主存与cache地址映射方式 14

第四章 指令系统 16

4.1 指令的一般格式 16

4.2 操作码 16

4.3 寻址方式 16

4.4 操作码编码 20

4.5 机器指令格式 20

4.6 精简指令计算机系统（RISC）的特点 20

第五章 中央处理器 21

5.1 CPU的基本功能 21

5.2 CPU中的主要寄存器 21

5.3 指令周期流程图 22

5.4 微程序控制器 23

5.5 微指令的基本格式 24

5.6 微程序设计 24

5.7 流水CPU的时空图 25

5.8 水平型微指令 25

第六章 总线系统 26

6.1 总线带宽及计算 26

6.2 总线的仲裁 26

6.3 总线定时 26

6.4 I/O接口的定义及一般功能 26

第七章 外存和I/O接口 28

7.1 磁盘容量及数据传输率计算 28

第八章 输入输出系统 29

8.1 信息交换方式 29

8.2 设备编址 29

8.3 查询传送 30

8.4 中断 30

第九章 题型、分值 32

# 系统概论

## 冯诺依曼计算机

原理：二进制存储程序并按地址顺序执行（程序控制）

1. 计算机硬件有运算器、存储器、控制器、输入设备、输出设备五大基本部件组成。
2. 计算机内部采用二进制来表示指令和数据
3. 编好的程序和原始数据实现存入存储器，然后再启动计算机工作

## 计算机的硬件和软件

**计算机**

**硬件**

**软件**

**微型计算机**

**外围设备----**

**系统软件**

**程序设计语言**

**应用软件、软件包、数据库**

**微处理器**

**内存储器**

**输入/输出接口电路**

**系统总线**

**运算器**

**控制器**

**RAM**

**ROM**

**数据总线**

**地址总线**

**控制总线**

**键盘/显示器、打印机、磁带机、磁盘机、D/A、A/D转换器**

**监控程序、操作系统**

**汇编程序**

**解释程序**

**编译程序**

## 软件与硬件的逻辑等价性

1. 计算机系统以硬件为基础，通过配置软件扩充功能
2. 集成电子技术的发展，软件硬化是一种趋势

# 运算方法和运算器

## 进位计数制及数制之间的转化

## 数的编码

1. 定点表示和浮点表示，数的符号表示
2. 补码、求补：写出与该负数相对应的正数的补码，按位求反，末位加一
3. 移码：主要用于表示浮点数的阶码，移码的符号表示规律与补码的相反

## 各种字符的编码：ASCII码、BCD码



## 浮点数

N=R**e**×M，数符、阶码、尾数

考虑浮点数的运算，为了方便对阶，阶码通常采用移码形式（E）

## IEEE754标准浮点格式

|  |  |  |
| --- | --- | --- |
| MS | E1E2Em | M1M2Mn |

上式的实际值为：x=（－1）**s×**（1.M）×2E－K

## 定点补码加、减运算

[x]补+[y]补=[x+y]补、[x-y]补=[x]补-[y]补=[x]补+[-y]补

## 溢出的概念及判别

1. 双进位判别
2. 双符号位判别

## 一位全加器、行波进位加法器、先行进位加法器：74181

## 原码一位乘法原理、补码一位乘法原理

当两个用原码表示的数相乘时，可以用其数字部分直接相乘，得到乘积的数字部分，而乘积的符号取两个数符号的异或值即可。

补码一位乘法可按布斯（Booth）公式进行。

设有n位补码

[x]补 = xn-1 xn-2 … x1 x0

[y]补 = yn-1 yn-2 … y1 y0 y-1

其中，xn-1 和yn-1是[x]补和[y]补的符号位，y-1 是给[y]补 添加的一个附加位，且y-1 = 0，则布斯公式为



## 定点运算器的结构

单总线：同一时间内，只有一个操作数放在单总线上。为了把两个操作数输入到ALU，需要进行两次，而且要A、B两个缓冲寄存器。

例子：执行一个加法操作，需要将第一个操作数放入缓冲寄存器A中，再将第二个操作数放入缓冲寄存器B中，只有当两个操作数同时出现在ALU的两端，ALU才执行加法。当加法结果出现在总线上时，由于输入数已经保存在缓冲寄存器中，它并不会打扰输入数。然后由第三个传送动作，将加法的“和”选通到目的寄存器中。

结构特点——单路简单，但操作速度较慢。

双总线：只需要一次操作控制，可以将两个操作数同时加到ALU进行运算，并且马上就能得到运算结果。

例子：两条总线将两个数据分别送到ALU的输入端。专用寄存器分成两组，它们分别与一条总线交换数据。这样通用寄存器中的数就可以进入到任意一组专用寄存器中去。从而使数据传送更为灵活。

## 浮点数加减运算

1. 0操作数的检查：操作数中含有零时停止运算
2. 对阶：使阶码相同，即小数点对齐

原则：小阶向大阶看齐，右移位数等于阶差

1. 尾数加或减运算：对阶结束后，尾数的加/减运算与定点加减运算完全一样
2. 结果规格化

当尾数运算结果是01.\*或者10.\*，即两符号位不相等，表明运算结果的绝对值大于1，需进行右移1位，阶码加1，实现向右规格化；当尾数不是1.M时还须向左规格化

1. 舍入处理

在对阶或向右规格化时，尾数右移，对低位部分处理

简单的舍入方法：“0舍1入”和“恒置1”

在IEEE754标准中，舍入处理有4种：

1. 就近舍入
2. 朝0舍入
3. 朝+∞舍入
4. 朝-∞舍入

## 浮点数乘、除运算

乘法：阶码相加，尾数相乘

除法：阶码相减，尾数相除

定点补码加、减运算，溢出判别

浮点数加减运算过程P55例28

运算器核心部件：ALU

n×n的不带符号阵列乘法器需要\_\_n\*(n-1)\_个全加器和\_\_n\*n\_\_与门。P33

用带求补器的无符号阵列乘法器求乘积。P36例21

算前求补器输出为x、y的绝对值，算后求补器输出为x×y的补码，再根据补码求出其真值，并验证结果

# 存储器

## 存储系统分级结构

高速缓存cache、主存、辅存。

Cache：存取速度快、容量小，主要用于存取指令和数据。由高速小容量双极型半导体存储器组成。

主存：计算机运行期间存储大量程序和数据，一般由MOS半导体存储器组成

辅存：成本低，通常用来存储系统程序和大型数据文件及数据库。

主存—辅存层次：主要解决容量问题

Cache—主存层次：主要解决速度问题

## 存储器存储容量

存储器存储容量与地址、数据线个数有关

芯片的存储容量=存储单元数×存储单元的位数=2**M×N**

**M：芯片的地址线根数，N：芯片的数据线根数**

## 半导体存储器的分类：RAM、ROM

**半导体**

**存储器**

**只读存储器**

**（ROM）**

**随机存取存储器**

**（RAM）**

**静态RAM（SRAM）**

**动态RAM（DRAM）**

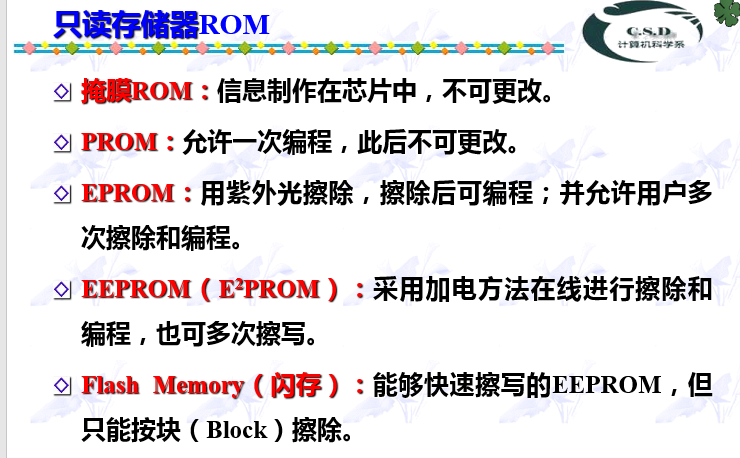
**非易失RAM（NVRAM）**

**掩膜式ROM**

**一次性可编程ROM（PROM）**

**紫外线擦除可编程ROM（EPROM）**

**电擦除可编程ROM（EEPROM）**



## DRAM的刷新

****

集中式刷新：刷新过程中，不能对存储器进行读写操作。所有行在每一个刷新周期中都被刷新。适用于高速存储器。（一半一半）

分散刷新：每一行的刷新插入到正常的读/写周期中。分散的刷新，缺点整个系统的速度降低。

在每个存取操作后绑定一个刷新操作。

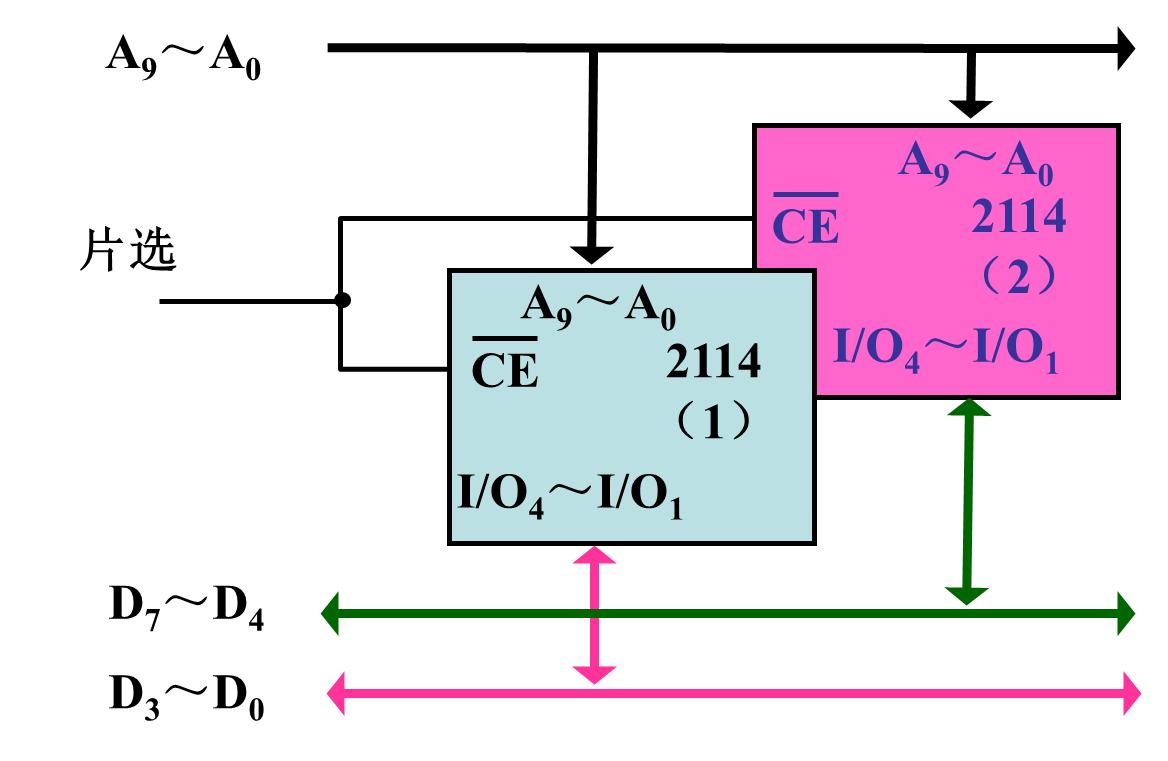
## 存储器的扩展

存储器芯片的容量总是有限的，容量越大，工艺越难，价高越高。因此总是利用一些芯片组合构成大容量存储器。称为扩展。

位扩展：用设计要求的存储容量/芯片的存储容量 等于 芯片数。

字数相同，字长不同

连接方式：控制线（片选线、读写控制线等）均分别并联，地址信号线分别并联，每个芯片的数据信号线分别与相应的数据总线相连。



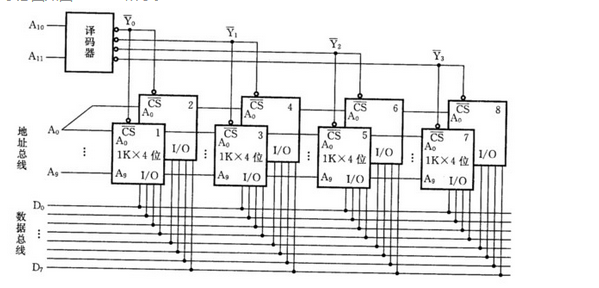
字扩展：给定的芯片容量较小，不满足设计要求的总存储容量，此时需要用到给定芯片扩展字数。字数不同，字长相同

读写控制信号均与系统总线的读写信号相连，数据分别并联与相应的数据总线相连，片内寻址的地址信号分别并联，高位地址信号译码产生每个芯片的片选信号。（CPU要实现对存储单元的访问，首先要选择存储芯片，即进行片选）

字位扩展：是前两种方式的结合，即当构成存储器的芯片容量的字数与存储器容量的字数不相同，且字长也不相同时，采用字位扩展。

具体方法，先进行位扩展，确定几个芯片一组；再进行字扩展，确定共几组。

例子用1K×4构成4K×8存储器



## 存储器的地址译码

片内地址译码是存储芯片上的地址线与系统地址总线的低位相连后，用于对片内存储单元的寻址。

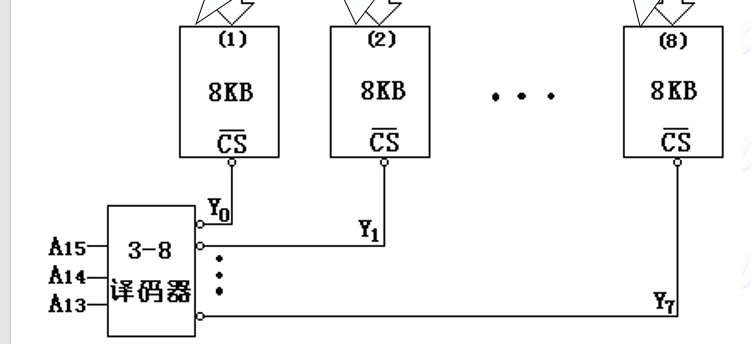
片选控制译码电路对高位地址进行译码后产生存储芯片的片选信号。(

方法①线选法：用高位地址线直接作为存储芯片的片选信号，每一根地址线选通一芯片。

优点，连线简单，片选控制无需专门的译码电路。

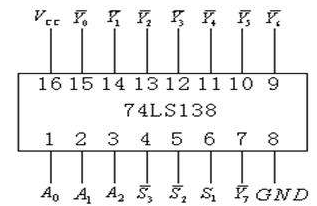
缺点，导致地址重叠；存储器地址分布不连续，可寻址范围减少

方法②全译码法：将高位地址总线全部经译码后作为各芯片的片选信号。



)

74LS138



A0~A2：地址输入端

SEA(E1)：选通端

/STB（/E2）、/STC（/E3）：选通端（低电平有效）

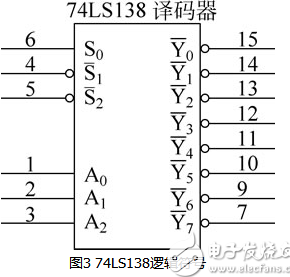
/Y0~/Y7：输出端（低电平有效）

VCC：电源正

GND：地

A0~A2对应Y0——Y7；A0，A1，A2以二进制形式输入，然后转换成十进制，对应相应Y的序号输出低电平，其他均为高电平；

原理：①当一个选通端（E1）为高电平，另两个选通端（（/E2））和（/E3））为低电平时，可将地址端（A0、A1、A2）的二进制编码在Y0至Y7对应的输出端以低电平译出。（即输出为Y0至Y7的非）比如：A2A1A0=110时，则Y6输出端输出低电平信号。



## 双端口存储器、多模块交叉存储器

双端口存储器(空间并行)：一个存储器具有两组相互独立的读写控制电路

访问地址不同，就不会发生冲突；通过两个端口访问同一个单元时，就会发生冲突，设置里BUSY标志信号，优先端口操作完毕，另一端的BUSY失效，端口开放。

应用：运算器中应用双端口存储器作为通用寄存器，能够快速提供双操作数；双端口存储器放在CPU和外围设备之间，通过共享的系统总线，增大信息吞吐量；多机系统中采用双端口存储器，作为CPU共享存储器，实现多CPU之间的通信。

多模块交叉存储器(时间并行):

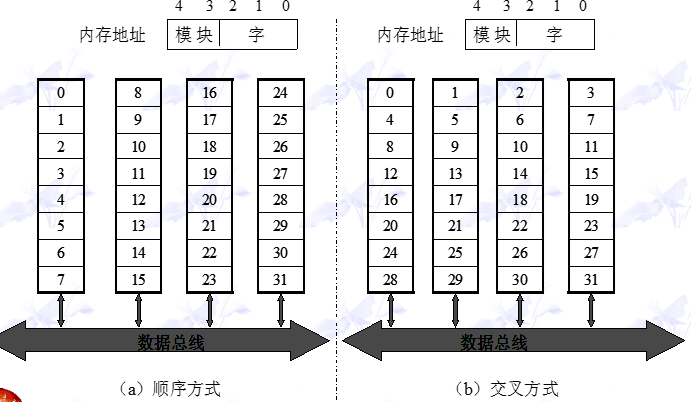
* 1. 顺序编址

在多模块组成的主存储器中，其单元地址按顺序对一个模块编完，再接着对另一个模块的进行编排，一直到所有模块排完。

特点：各模块串行工作，存储器带宽受限。增添模块扩容方便。

* 1. 交叉编址

可以对所有模块同时访问，加快存取速度。

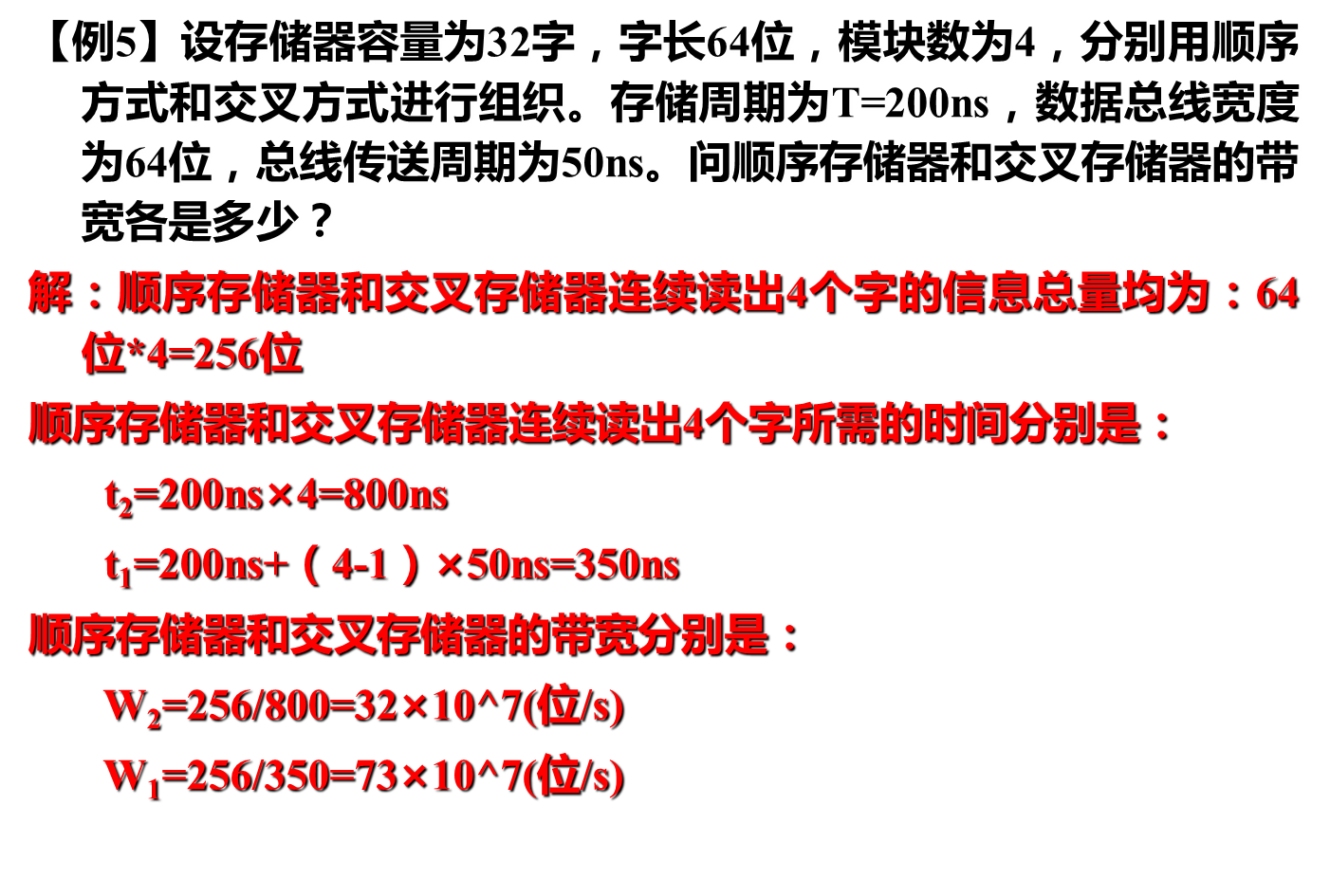


定量分析：

设模块字长等于数据总线宽度，模块存取一个字的周期为T，总线传送周期为t，那么，CPU可以每隔t时间启动下一个模块。

若设m=T/t，只要模块数M大于或等于m，连续读取m个模块的存储字所需的时间为：t1=T+(m-1)\*t

而顺序方式则需： t2=mT



## Cache的命中率

命中率与程序的行为，cache的容量，组织方式，块的大小有关。

命中率：CPU访问cache的次数占总访问次数的比例。

设NC表示cache完成存取的总次数，NM表示主存完成存取的总次数，h为命中率，则有：h=NC/(NC+NM)

若tC表示命中时的cache访问时间，tm表示未命中时的主存访问时间，则cache/主存系统的平均访问时间ta为：

ta=htc+（1-h）tm

目标：以较小的硬件代价使cache/主存系统的平均访问时间ta越接近tc越好，用e表示访问效率表示。

设tm/tc=r则有： e= tc / ta=1/(r+(1-r)h)

命中率h越接近于1越好，r值以5～10为宜，不宜太大。

## 主存与cache地址映射方式

全相联方式：主存的任意一块可以映象到Cache中的任意一块

主存的地址：

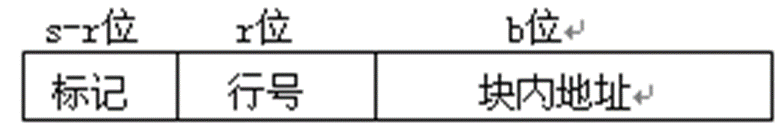


优点：命中率比较高，Cache存储空间利用率高。

缺点：访问相关存储器时，每次都要与全部内容比较，速度低，成本高，因而应用少。（一般用于小容量cache）

直接方式：一个主存块只能复制到cache的一个特定行中。

主存的地址：

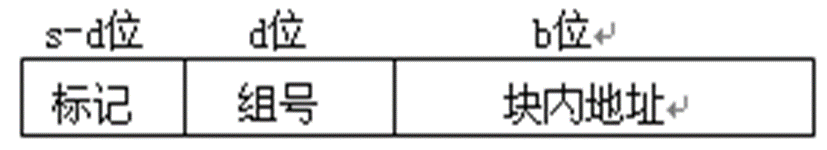


检索过程：根据地址中的r位行号找到cache的相应行，然后将该行中所存的标记部分与内存地址的高s-r位比较，判断是否命中。

特点：硬件实现简单，成本低。但每个主存块只有一个固定的行可以存放。当cache的行数较少时发生冲突的可能性很大，频繁的置换会使cache的效率下降。直接映射方式一般用于大容量的cache中。

组相联方式：组间直接映射，组内全相联映射。

主存的地址划分：



检索过程：根据地址中的d位组号找到cache的相应组，然后将该组中所有行的标记部分与内存地址的高s-d位同时比较，判断是否命中。

特点：组内有一定的灵活性，而且因组内行数的减少，比较的硬件电路比全相联方式简单些。

提高主存速度的具体措施：

使用快速存储器芯片、采用多体交叉存储器和高速缓冲存储器（CACHE）

# 指令系统

## 指令的一般格式

|  |  |
| --- | --- |
| **操作码（OPCODE）** | **地址码（ACODE）** |

## 操作码

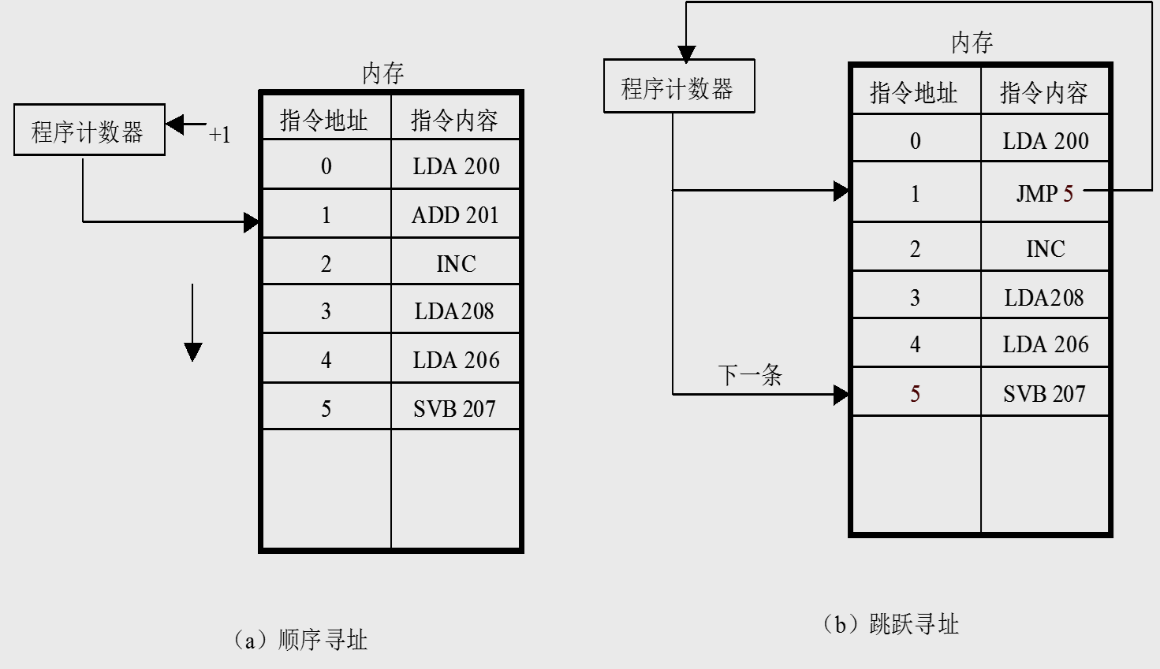
操作码的位数一般取决于指令系统的规模

## 寻址方式

指令寻址方式：

顺序寻址方式：按照指令在内存中的先后顺序，取一条执行一条。

跳跃寻址方式：跳跃寻址方式中要执行的指令的地址不是按程序顺序排列的下一条，而是跳到别的指令进行执行。采用跳跃寻址实现程序的转移或构成循环程序。

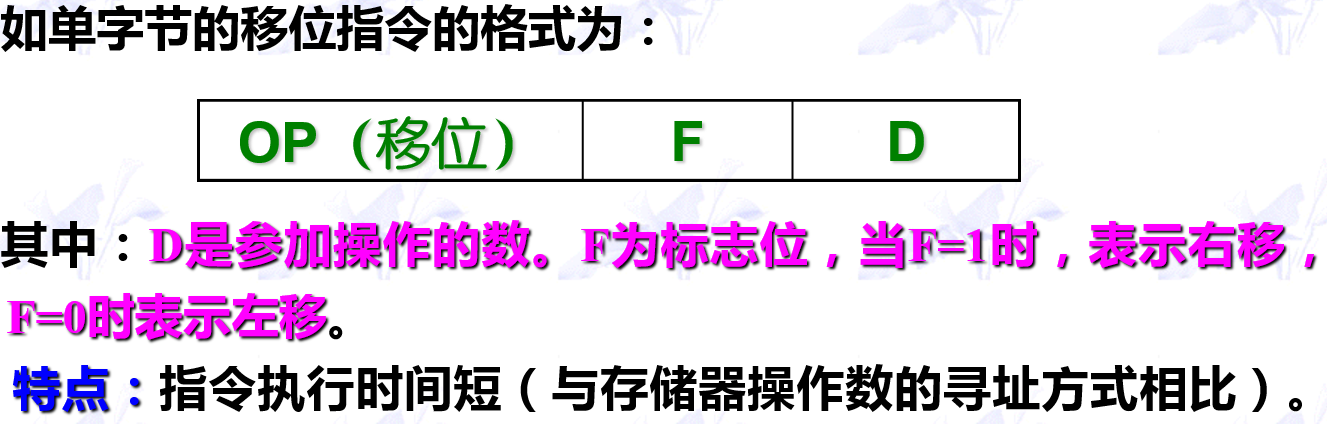


操作数的寻址方式：寻找操作数位置的方式

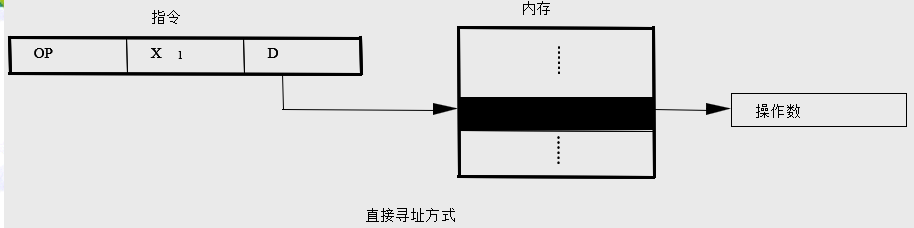
**机器中常采用的寻址方式**

立即寻址、寄存器寻址、直接寻址、间接寻址、寄存器间接寻址、相对寻址、基址寻址、变址寻址、隐含寻址、块寻址、段寻址

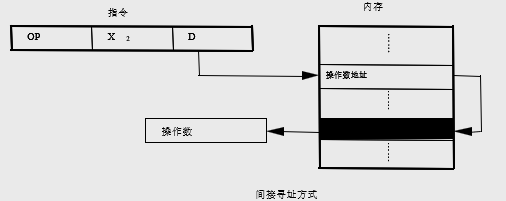
立即寻址：指令的地址码段不是操作数的地址，而是操作数本身。



直接寻址：指令的地址字段直接给出操作数在内存的地址。即形式地址就是有效地址。

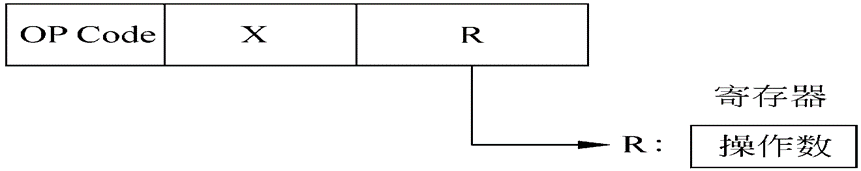


间接寻址：指令的地址码字段的形式地址D不是操作数真正的地址，而是操作数地址的位置（地址）

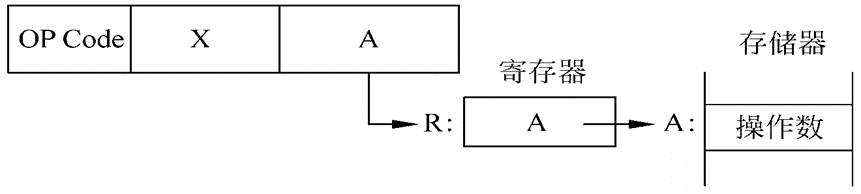


寄存器寻址和寄存器间接寻址：

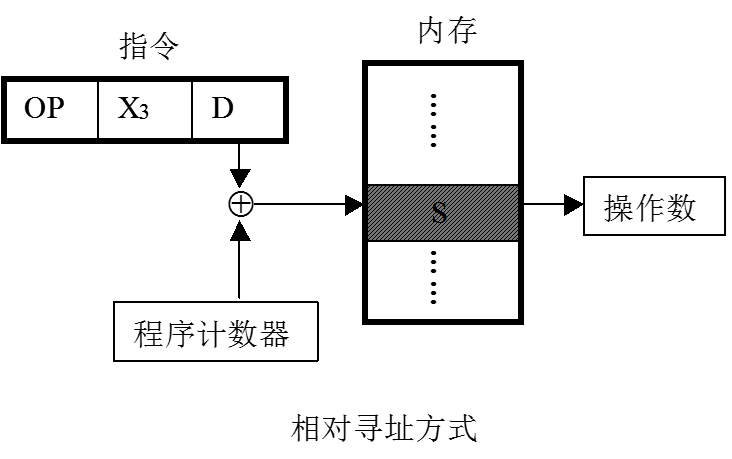
寄存器寻址：操作数在寄存器中时，指令的地址字段给出寄存器的编码。



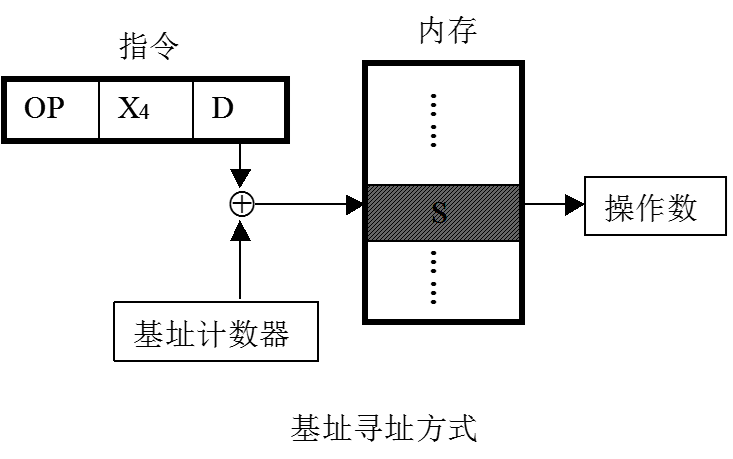
寄存器间接寻址：指令中的地址字段的寄存器中的内容不是实际的操作数，而是操作数（在内存中）的地址。



相对寻址：在程序计数器PC的内容上加上指令格式中的形式地址形成操作数的有效地址。（X3为寻址模式特征位）

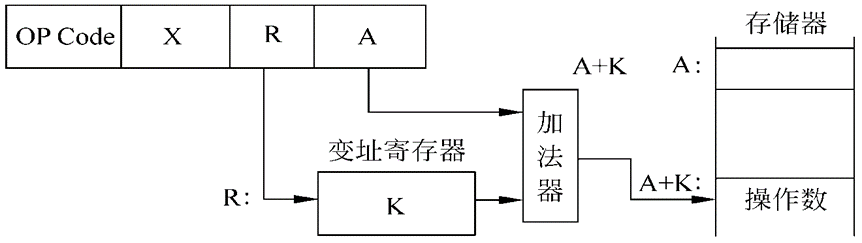


基址寻址：在基址寄存器的内容上加上指令格式中的形式地址形成操作数的有效地址。

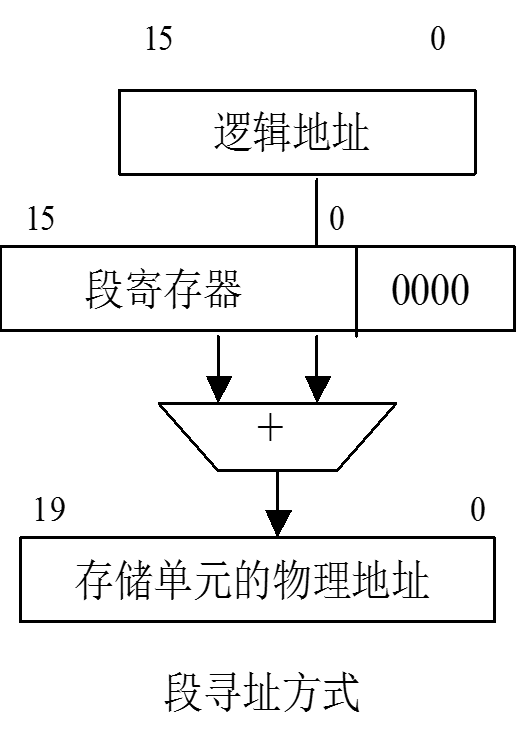


变址寻址：变址寻址与基址寻址相似，它是在变址寄存器上加上形式地址中给出的偏移量D形成有效地址。

基址寻址和变址寻址的有效地址的形成过程相似，实际应用中采用基址寻址扩大寻址能力（基址寄存器的位数比形式地址长），程序中改变形式地址；而变址寻址的应用是改变变址寄存器，使有效地址按变址寄存器的内容实现有规律的变化。

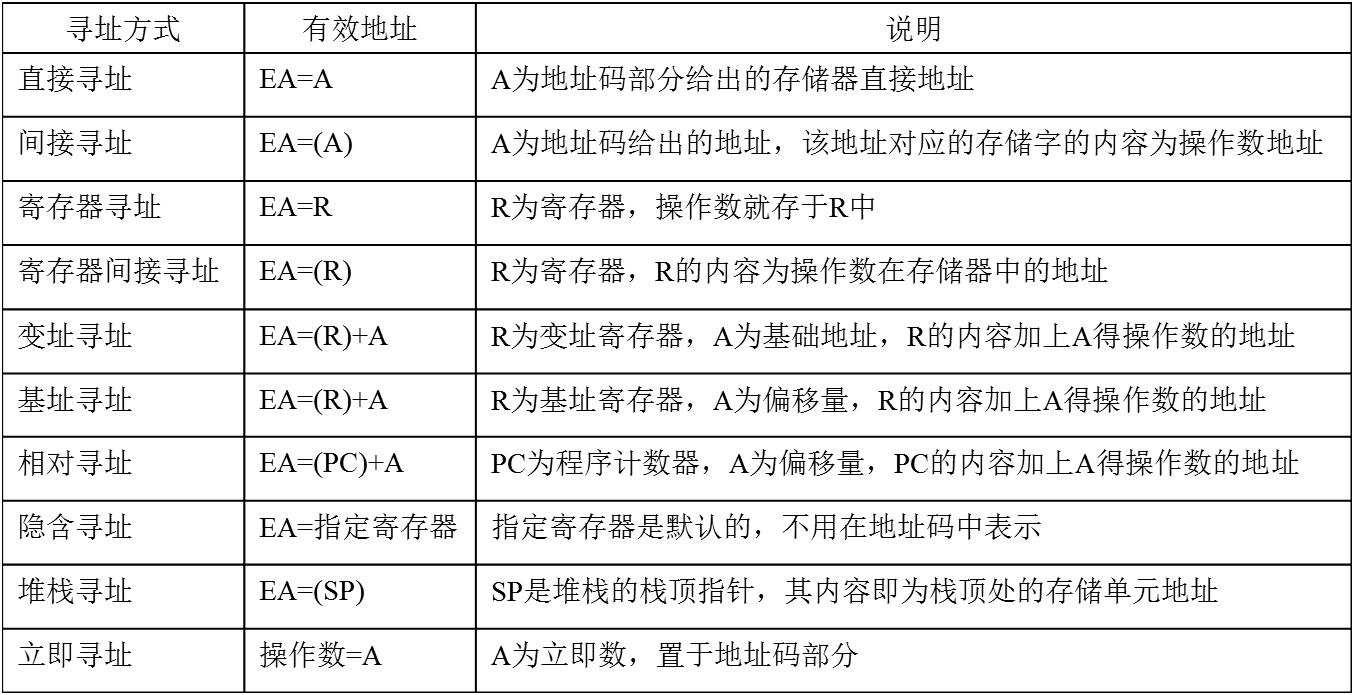


段寻址：存储器采用分段结构，形式地址就是逻辑地址，由段寄存器和逻辑地址组合形成要访问的存储器的物理地址。



隐含寻址：指令的操作数的位置没有明显给出，而是隐含的。

**总结：**



## 操作码编码

1. 定长编码

N≥log2m（m条指令，操作码长度为n）

1. 操作码扩展技术（课件第四章P52-57）Huffman编码

## 机器指令格式

用二进制代码表示的结构形式。

单字长指令：

双字长指令：

三字长指令：

## 精简指令计算机系统（RISC）的特点

1. 选取使用频率最高的一些简单指令；
2. 指令长度固定，指令格式种类少，寻址方式少；
3. 只有取数、存数指令访问存储器，其余指令的操作都在寄存器之间进行。

# 中央处理器

## CPU的基本功能

指令控制：程序的顺序控制，保证机器按顺序执行程序。（内置程计数器等部分）

操作控制：管理并产生完成的指令执行操作所需的控制信号，把这些信号送往相应的部件，控制这些部件按指令的要求进行动作。

时间控制：对各种操作实施时间上的定时，保证计算机有条不紊的自动工作。

数据加工： 对数据进行算术运算和逻辑运算处理

## CPU中的主要寄存器

DR：数据缓冲寄存器（用来暂时存放ALU的运算结果）

作用：作为ALU运算结果和通用寄存器之间信息传送中时间上的缓冲；补偿CPU和内存、外围设备之间在操作速度上的差别。

IR：指令寄存器（用来保存当前正在执行的一条指令）

Cache->（指令译码器）->IR

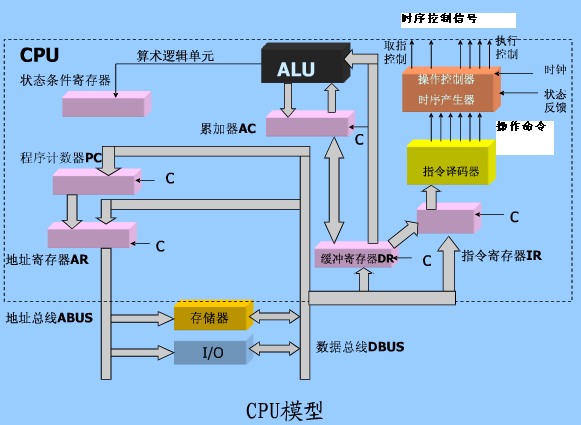
PC：程序计数器（保证程序能够连续执行下去，确定下一条指令的地址）又称为指令计数器

AR：数据地址寄存器（保存CPU所访问的数据cache存储器中单元的地址）因为要对存储器阵列进行地址译码

PSW：状态字寄存器（保存由算术指令和逻辑指令运算或测试结果建立的各种条件代码，除此之外，还保存中断和系统工作状态等信息）

累加器AC：暂时存放ALU运算的结果信息

通用寄存器R0~R3：为ALU提供一个工作区

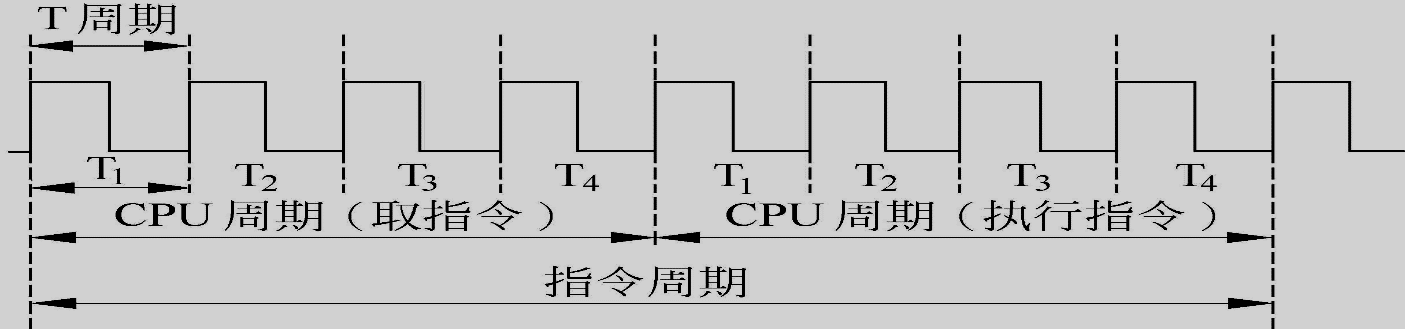
****

## 指令周期流程图

指令周期：取出并执行一条指令的时间（一般由若干个CPU周期组成）

CPU周期/机器周期：内存中读取一个指令字的最短时间

时钟周期：常称为节拍。（一个CPU周期又包含若干个时钟周期，它是处理操作的最基本单位）



## 微程序控制器

微程序控制器的基本组成：控制存储器、微指令寄存器和地址转移逻辑

微程序控制器的工作原理：

（1）将机器指令分解为基本的微命令序列，在制造CPU时固化在控制存储器CM中，执行一条机器指令时，CPU依次从CM中取出微指令产生微命令。

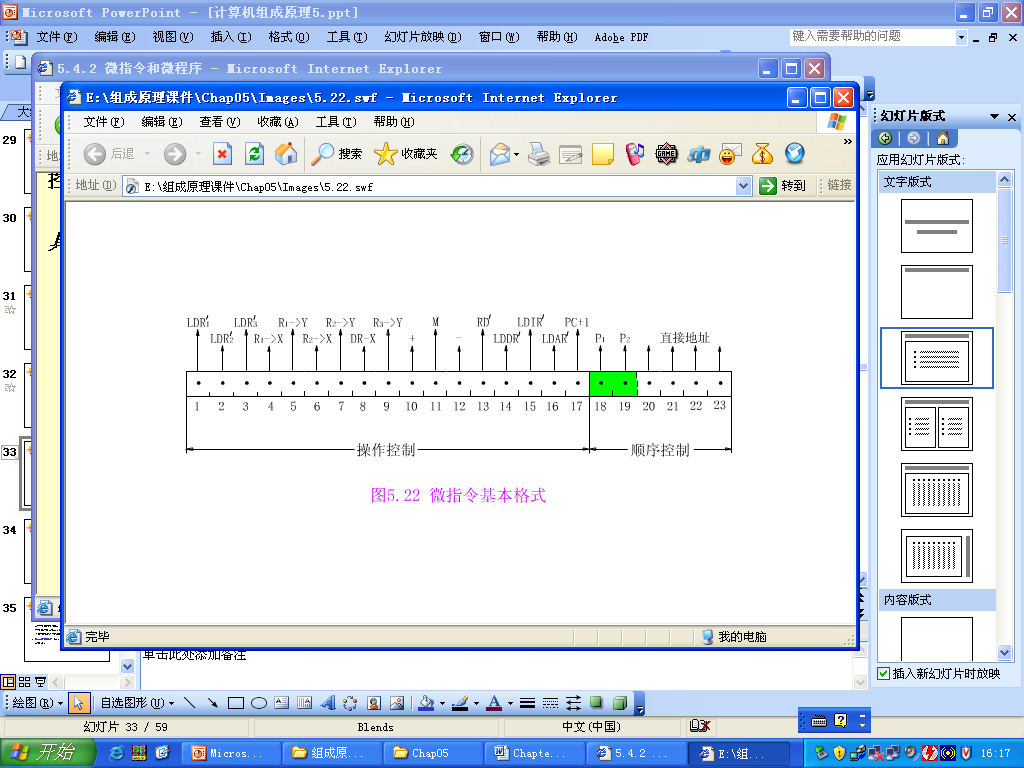
（2）一条微指令包含的微命令控制实现一步（一个节拍）操作，若干条微指令组成一小段微程序解释执行一条机器指令。

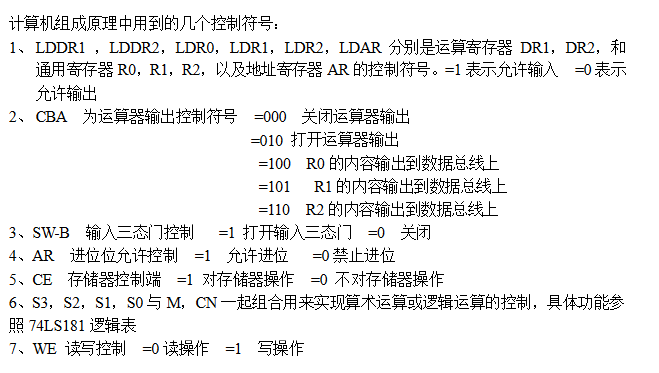
微操作的相容性和相斥性：

相容性的操作是指在同时或同一个CPU周期内可以并行执行的微操作。

相斥性是指不能在同时或同一个CPU周期内可以并行执行的微操作。

## 微指令的基本格式

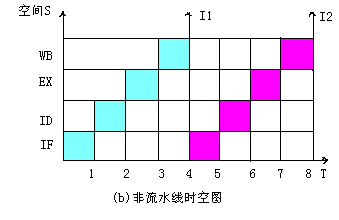
****

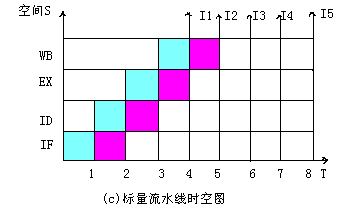


## 微程序设计

微命令编码：直接表示、小组译码、混合表示

## 流水CPU的时空图

****

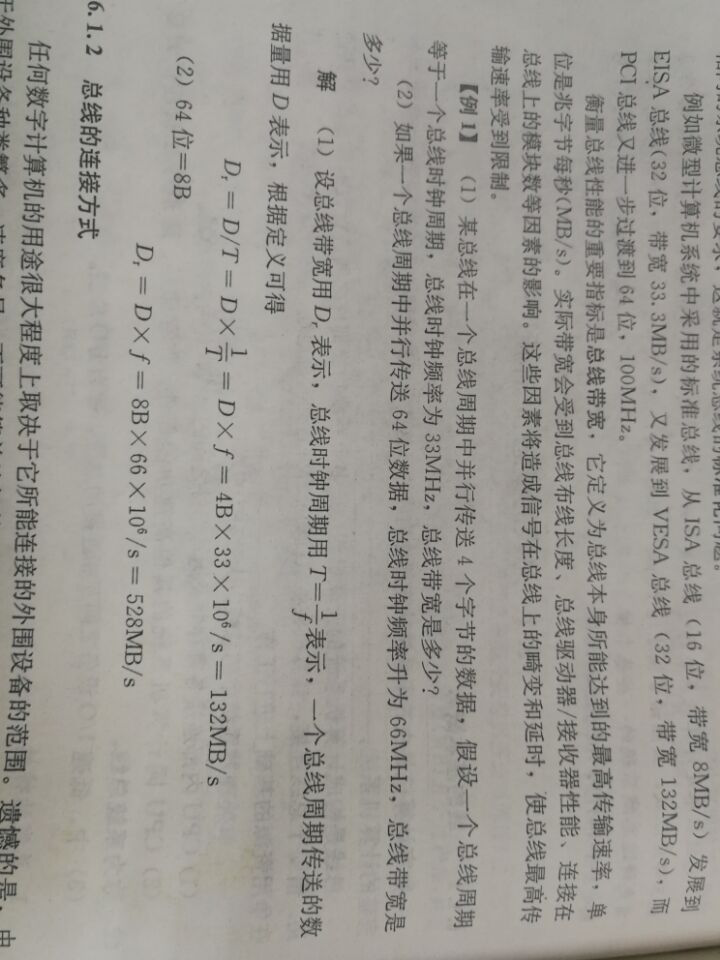
****

## 水平型微指令

水平型微指令是由控制字段，判别测试字段和下地址字段组成。

# 总线系统

## 总线带宽及计算



## 总线的仲裁

1. 集中式仲裁：链式查询方式、计数器定时查询方式、独立请求方式
2. 分布式仲裁：分布式仲裁是以优先级仲裁策略为基础

## 总线定时

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。这种关系称为总线定时。实质是一种协议或规则。

同步定时：事件出现在总线上的时刻由总线时钟信号来确定；

异步定时：采用应答方式进行总线传输控制。

## I/O接口的定义及一般功能

接口：I/O功能模块通常简称为I/O接口，也叫适配器。广义地讲，I/O接口是指CPU、主存、外围设备之间通过系统总线进行连接的标准化逻辑部件。

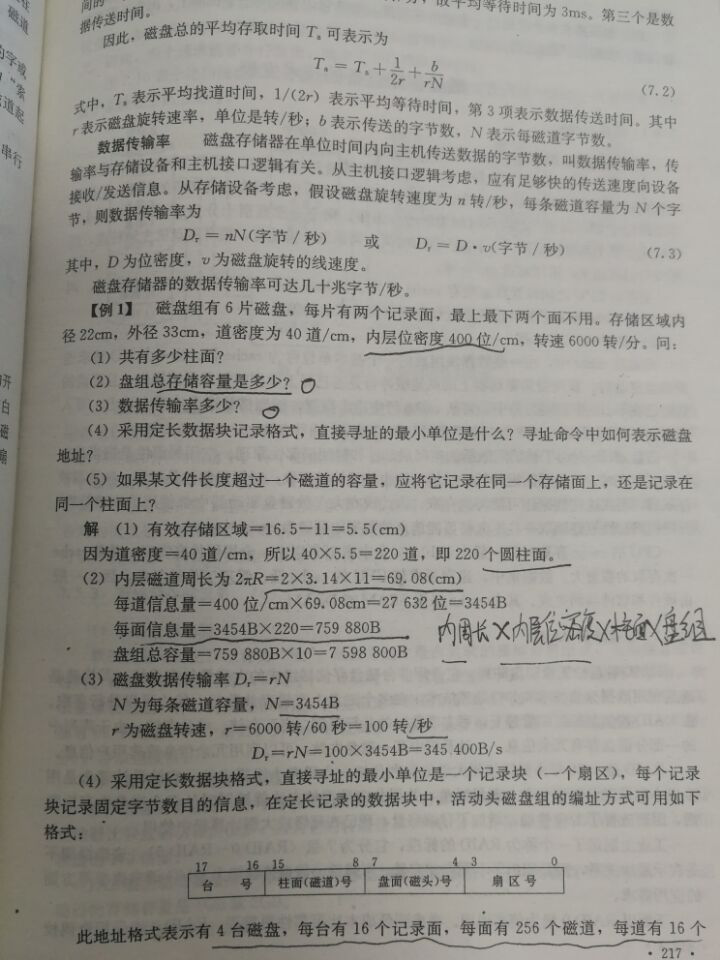
功能：

1. 控制，接口模块靠指令信息来控制外围设备的动作，如启动、关闭等等
2. 缓冲，接口模块在外围设备和计算机系统其他部件之间用作为一个缓冲器。补偿各种设备之间速度的差异。
3. 状态，接口模块可以完成任何要求的数据转换，例如并-串与串-并之间的转换
4. 整理，接口模块可以完成一些特别的功能，例如在需要时间可以修改字计数器或当前内存地址寄存器
5. 程序中断，每当外围设备向CPU请求某种动作时，接口模块即发生一个中断请求信号到CPU。

I/O接口模块分为串行数据接口和并行数据接口两大类

# 外存和I/O接口

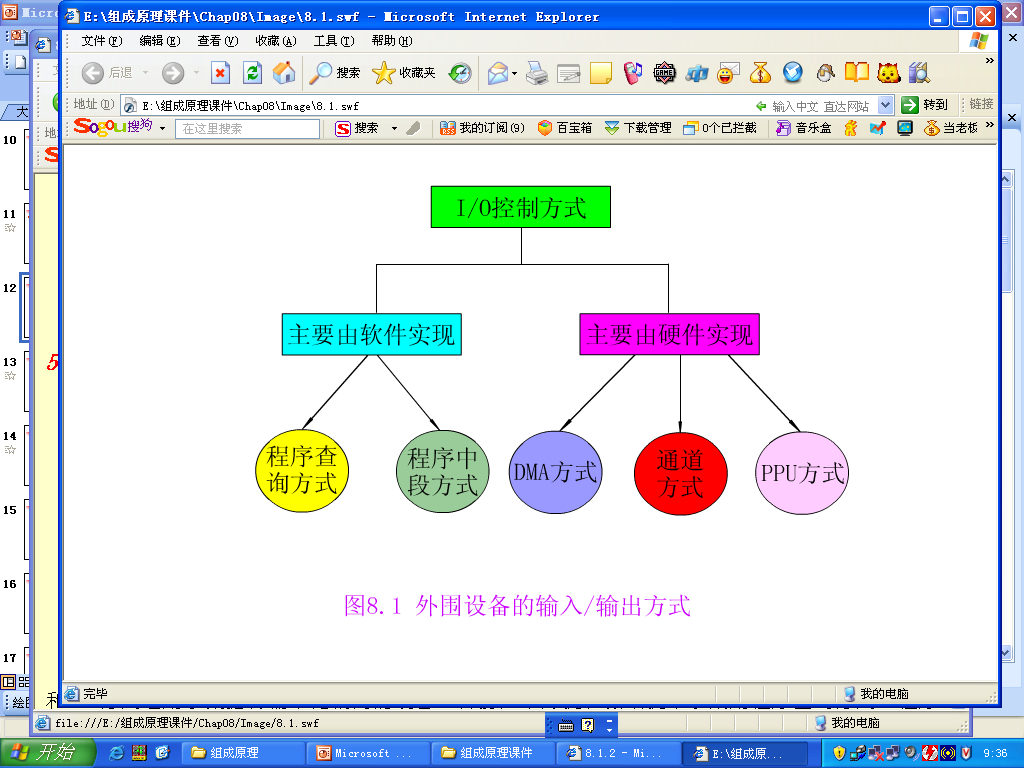
## 磁盘容量及数据传输率计算



# 输入输出系统

## 信息交换方式

无条件传送

****

## 设备编址

统一编址：是类似于访问内存，把外设的寄存器，当做一个内存地址来访问，从而可以像访问内存一样去访问外设，这种方式即是内存与外设统一的编址方式，称作统一编址。

独立编址：使用专用的CPU指令来进行访问特定的外设，称作独立编址。

## 查询传送

**CPU读I/O状态**

**检查状态**

**未准备就绪**

**出错**

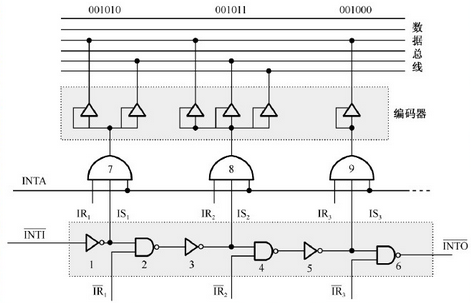
## 中断

中断的基本概念：中断是指计算机运行过程中，出现某些意外情况需主机干预时，机器能自动停止正在运行的程序并转入处理新情况的程序，处理完毕后又返回原被暂停的程序继续运行。

中断向量地址是中断服务程序入口地址

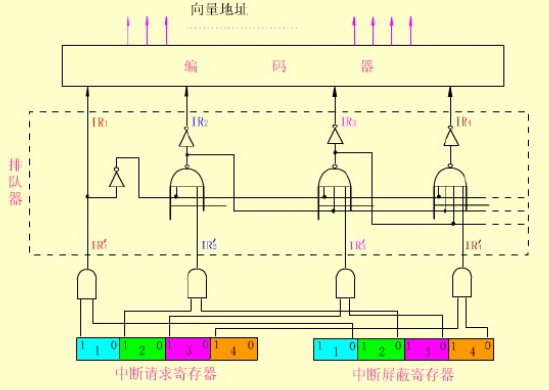
单级中断：所有的中断源都属于同一级，所有中断源触发器排成一行，其优先次序是离CPU近的优先权高。

请求方式：采用串行排队链法来实现具有公共请求线的中断源判优识别。



多级中断：指具有很多中断源，根据各中断事件的轻重缓急程度不同来划分若干级别，每一个中断级分配一个优先权。一般来说，优先权高的可以打断优先权低的中断服务程序，以程序嵌套方式进行工作。

请求方式：每一级均有一个根中断请求线送往CPU的中断优先级排队电路，对每一级赋予了不同的优先级。



对于多级中断，一个系统若有n级中断，在CPU就有n个中断请求触发器，总称为中断请求寄存器；与之对应有n个中断屏蔽触发器，总称为终端屏蔽寄存器。不同级中断源可以嵌套，同一级不同中断源的中断不能嵌套。

对于两种中断，都有使用中断堆栈保存现场信息。使用堆栈保存现场信息的好处是：控制逻辑简单，每一级中断不必再单独设置现场保护区，各级中断现场可按其顺序放在同一个栈里。

# 题型、分值

填空题、单项选择题、简答题、分析计算、分析设计，前一部分考核各章基本概念，占55分，其中简答题也需简单计算；分析计算、分析设计部分由4个小题组成，占45分。

认真看书复习，熟练掌握基本概念，对各章所讲的例题、习题，一定要真正理解并掌握解题步骤。