第一章 概述

# 存储程序概念

美籍匈牙利数学家**冯·诺依曼**等人在1946年6月提出**存储程序**概念。

**存储程序概念：**

* 计算机（指硬件）应由**运算器、存储器、控制器、输入设备和输出设备**五大基本部件组成；
* 计算机内部采用**二进制来表示指令和数据**；
* **将编好的程序和原始数据事先存入存储器中，然后再启动计算机工作**，这就是存储程序的基本含义。

**练习：**

**1、冯·诺依曼结构的特点是 存储程序 。**

**2、关于冯.诺依曼计算机，下列说法正确的是（ C ）**

A. 冯. 诺依曼计算机的程序和数据是靠输入设备送入计算机的寄存器保存的

B. 冯.诺依曼计算机工作时是由数据流驱动控制流工作的（控制流驱动数据流）

C. 冯.诺依曼计算机的基本特点可以用**“存储程序”和“程序控制”**高度概括

D. 随着计算机技术的发展，冯. 诺依曼计算机目前已经被淘汰

**3. 冯.诺依曼计算机的核心思想是（B），冯. 诺依曼计算机的工作特点是（C）**

（1）A. 采用二进制 B. 存储程序 C. 并行计算 D. 指令系统

（2） A. 堆栈操作 B. 存储器按内容访问

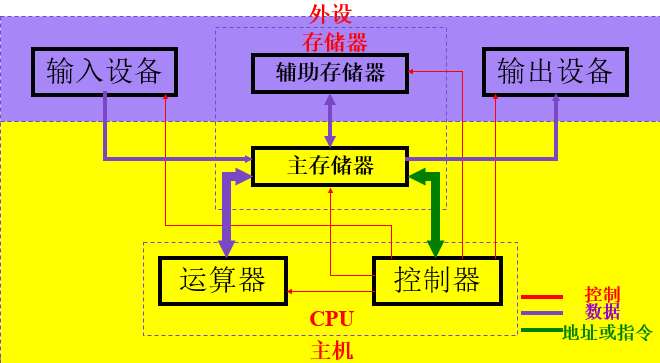
C. **按地址访问并顺序执行指令** D. 多指令流单数据流

# 计算机的硬件组成

**中央处理器（CPU）**：CPU = 运算器 + 控制器

**主机**：主机 = 中央处理器 + 主存储器

**外部设备**：除去主机以外的硬件装置（如输入设备、输出设备、辅助存储器等）



## 计算机的主要部件

**1.输入设备（常用的有键盘、鼠标、扫描仪等）**

输入设备的任务是把人们编好的程序和原始数据送到计算机中去，并且将它们转换成计算机内部所能识别和接受的信息方式。

**2.输出设备（常用的有显示器、打印机、绘图仪等）**

输出设备的任务是将计算机的处理结果以人或其他设备所能接受的形式送出计算机。

**3.存储器**

存储器是用来存放程序和数据的部件，它是一个记忆装置，也是计算机能够实现**“存储程序控制”的基础。**

* **主存储器**

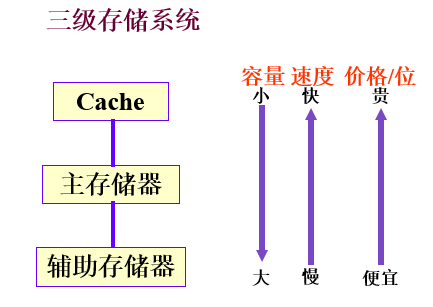
可由**CPU直接访问**，用来存放当前正在执行的程序和数据。

* **辅助存储器**

设置在主机外部，CPU**不能直接访问**，**用来存放暂时不参与运行的程序和数据，需要时再传送到主存。**

* **高速缓冲存储器（Cache）**

CPU**可以直接访问**，用来存放当前**正在执行的程序中的活跃部分（副本）**，以便快速地向CPU提供指令和数据。



**4.运算器**

运算器是对信息进行处理和运算的部件，经常进行的运算是算术运算和逻辑运算，因此运算器的核心是算术逻辑运算部件ALU。

运算器中有若干个寄存器（如累加寄存器、暂存器等）。

**5.控制器**

控制器是整个计算机的指挥中心。

控制器中主要包括时序控制信号形成部件和一些专用的寄存器。

**练习：**

**4. 计算机的存储系统是指（ D ）**

A. RAM B. ROM C. 主存储器 D. cache、主存储器、辅助存储器

**5. 叙述错误的是（ C ）**

A. 把数据从内存传输到硬盘叫写盘

B. 把源程序转换为目标程序的过程叫编译

C. 应用软件对操作系统没有任何要求

D. 计算机内部对数据的传输、存储和处理都是用二进制

**6. 计算机的存储单元中存储的内容是（ D ）**

A. 数据和指令 B. 只能是指令 C. 只能是数据 D. 数据或指令

**7. 微型计算机的运算器、控制器及内存的总称是（ C ）**

A. CPU B. ALU C. 主机 D. MPU

**8. 个人计算机属于（ D ）**

A. 巨型机 B. 中型机 C. 小型机 D. 微机

## 冯·诺依曼结构和哈佛结构的存储器设计思想

**1、冯·诺伊曼结构（普林斯顿结构）**

* 将**程序指令存储器和数据存储器合并在一起**
* 指令存储地址和数据存储地址指向同一个存储器的不同物理位置。



**2、哈佛结构**

* 将**程序指令存储和数据存储分开**
* 取指令 🡪 译码 🡪 数据地址 🡪 读取数据 🡪 执行
* 通常具有较高的执行效率。执行指令时可预取下条指令



哈弗结构至少有2组总线：**程序存储器（PM）的数据总线和地址总线，数据存储器（DM）的数据总线和地址总线**

# 计算机系统

**计算机系统=硬件系统+软件系统**

**PS：**

计算机的软件分为**系统软件和应用软件**。系统软件是为了计算机能正常、高效工件所配备的各种管理、监控和维护系统的程序及其有关资料。**系统软件主要包括如下几个方面：**

（1） **操作系统软件**，这是软件的核心

（2） **各种语言的解释程序和编译程序**（如BASIC语言解释程序等）

（3） **各种服务性程序**（如机器的调试、故障检查和诊断程序等）

（4） **各种数据库管理系统**（FoxPro等）

**练习：**

**10. 完整的计算机系统应包括（ D ）。**

A. 运算器、存储器、控制器 B. 外部设备和主机

C. 主机和实用程序 D. 配套的硬件设备和软件系统

**11. 下列软件中，不属于系统软件的是（ D ）**

A．编译软件 B．操作系统 C．数据库管理系统 D．C语言程序

**12. 某单位的人事档案管理程序属于（ B ）**

A. 工具软件 B. 应用软件 C. 系统软件 D. 字表处理软件

**13. 使用高级语言编写的程序称为（ A ）**

A. 源程序 B. 编辑程序 C. 编译程序 D. 连接程序

**14. 在微型计算机中，bit的中文含义是（ A ）**

A. 二进制位 B. 字 C. 字节 D. 双字

**15.现在主要采用 总线 结构作为微/小型计算机硬件之间的连接方式。**

# 计算机的主要性能指标

**1、机器字长**

**机器字长是指参与运算的数的基本位数，它是由加法器、寄存器、数据总线的位数决定的。**

在计算机中为了更灵活地表达和处理信息，许多计算机又以**字节（Byte）**为基本单位，一个字节等于8位二进制**位（bit）**。

**2、数据通路宽度**

**数据总线一次所能并行传送信息的位数，称为数据通路宽度。**它影响到信息的传送能力，从而影响计算机的有效处理速度。这里所说的数据通路宽度是指外部数据总线的宽度，它与CPU内部的数据总线宽度（内部寄存器的大小）有可能不同。

**3、主存容量**

**一个主存储器所能存储的全部信息量称为主存容量。**衡量主存容量单位有两种：

**① 字节数**。这类计算机称为**字节编址**的计算机。每1024个字节称为1K字节（210=1K），每1024K字节称为1M字节（220=1M），每1024M字节称为1G字节（230=1G）。

**② 字数×字长。**这类计算机称为**字编址**的计算机。如：4096×16表示存储器有4096个存储单元，每个存储单元字长为16位。

**常见题型：**

1. 若主存容量为16KB，按照字节编址，**则需要地址线14根，数据线8根**
2. 若主存容量为16KB，按照字编址，一个字为32位，**则需要地址线12根，数据线32根**
3. 若主存容量为16K，按照字编址，一个字为32位，**则需要地址线14根，数据线32根**

**4、运算速度**

**（1）主频：**（又称为**时钟频率**），表示在CPU内数字脉冲信号振荡的速度，即**1s中有多少个时钟周期。**

**（2）时钟周期：主频的倒数**

**（3）CPI（平均指令周期），每条指令执行所用的时钟周期数。**

**（4）IPC：每个时钟周期执行的指令数。即CPI的倒数。**

**（5）CPU执行的时间：**

**（6）MIPS：每秒执行多少百万条指令**

**MFLOPS：每秒执行多少百万次浮点运算**

**练习：**

**16. 系统总线中地址线的作用是（ C ）。**

A. 用于选择主存单元 B. 用于选择进行信息传输的设备

C. 用于指定主存单元和I/O设备接口电路的地址

D. 用于传送主存物理地址和逻辑地址

**17. 只有当程序执行时，才会将源程序翻译成机器语言，而且一次只能读取、翻译并执行程序中的一行语句，此程序称为（ C ） 。**

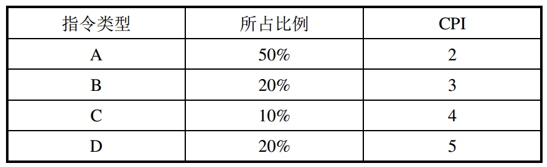
A. 目标程序 B. 编译程序 C. 解释程序 D. 汇编程序

**18. （2011）下列选项中，描述浮点数操作速度指标的是（ D ） 。**

A. MIPS B. CPI C. IPC D. MFLOPS

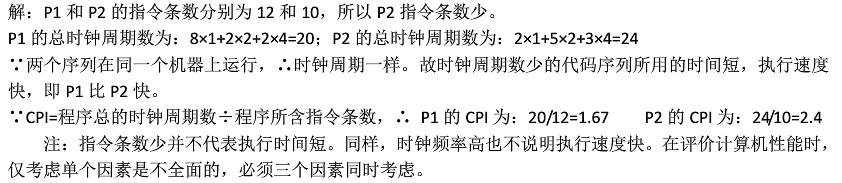
**19. （2013）某计算机主频为1.2GHz，其指令分为4类，它们在基准程序中所占比例及CPI如下表所示。该机的MIPS数是（ C ）。**

A. 100 B. 200 C. 400 D. 600

****

**19. C 解析：基准程序的CPI=2\*0.5+3\*0.2+4\*0.1+5\*0.2=3 ，计算机的主频为1.2GHz，为1200MHz，该机器的是MIPS为1200/3=400。**

**20. 假设计算机M的指令集中包含A、B、C三类指令，其中CPI分别为1、2、4。某个程序P在M上被编译成两个不同的目标代码序列P1和P2，P1所含A、B、C三类指令的条数分别为8、2、2 ，P2所含A、B、C三类指令的条数分别为2、5、3。请问：哪个代码序列指令条数少？哪个执行速度快？它们的CPI分别是多少？**

****

第二章 数据的机器层次表示

# 小知识点

时序逻辑电路的基本元器件是**触发器**；

组合逻辑电路的基本元器件是**门电路**。

由于计算机只能识别和处理二进制代码，所以字符必须按照一定的规则用一组二进制编码来表示。字符的编码的方式有很多种，现在用的最广泛的是美国国家信息交换标准字符码。（**ASCII码**）

# 计算机中的数值数据

在计算机中常用**后缀字母**来表示不同的数制。

**十进制数（D） 二进制数（B）**

**八进制数（Q） 十六进制数（H）**

# 无符号数和带符号数

**1、无符号数**

无符号数：就是整个机器字长的**全部二进制位均表示数值位**（没有符号位），相当于数的绝对值。

N1 =01001 //表示无符号数9

N2 =11001 //表示无符号数25

**对于字长为n+1位的无符号数的表示范围是 0～(2n+1-1)。**

例如：字长为8位，无符号数的表示范围是 0～255。

**2、带符号数**

带符号数，即**正、负数**。在日常生活中，我们用“+”、“-”号加绝对值来表示数值的大小，用这种形式表示的数值在计算机技术中称为“**真值**”。

机器数，就是将**真值的符号数码化**，约定二进制数的最高位为符号位，**“0”表示正号，“1”表示负号**，计算机能够识别和使用的表示形式。

N1 =01001 //表示带符号数+9

## 原码表示法

* 最简单的机器数表示法
* 用最高位表示符号位
  + **符号位为“0”表示该数为正**
  + **符号位为“1”表示该数为负**
* 数值部分与**真值**相同

**1.若真值为纯小数，原码形式为Xs.X1X2…Xn（Xs表示符号位）**

例1：X1=0.0110, X2=-0.0110 [X1]原=0.0110 , [X2]原=1.0110

**若表示纯小数，则能表示的范围：-(1-2-n) < [X]原 <1-2-n**

**2.若真值为纯整数，原码形式为XsX1X2…Xn（Xs表示符号位）**

例2： X1=1101, X2=-1101 [X1]原=0,1101 , [X2]原=1,1101

**（注意开区间）若表示纯整数，则能表示的范围：-2n < [X]原 <2n**

**在原码表示中，真值0有两种不同的表示形式：**

**[+0]原 =00000**  **[-0]原 =10000**

## 补码表示法

使符号位参加运算，从而简化加减法的规则；使减法运算转化成加法运算，从而简化机器的运算器电路。

**1.模和同余**

由于设备的原因，机器数是有字长限制的，不可能容纳无限大的任意数。当运算结果超出了机器的最大表示范围，就会发生溢出（丢失进位），此时所产生的溢出量称为模，用字母M表示。

**对钟表而言，M=12。假设：时钟停在8点，而现在正确的时间是6点，这时拨准时钟的方法有两种：**

时针倒着旋转2圈，等于分针正着旋转10圈。故有：**-2=10 (mod 12)** ，即 **-2和10同余。**

**2.补码表示**

* 符号位表示方法与原码相同
* 数值部分的表示与数的符号有关
  + **对于正数，数值部分与真值形式相同**
  + **对于负数，其数值部分为真值形式按位取反，且在最低位加1。**
* **若真值为纯小数，补码形式为Xs.X1X2…Xn（Xs表示符号位）**

例1：X1=0.0110 , X2=-0.0110 **[ X1]补=0.0110 , [X2]补=1.1010**

**若表示纯小数，则能表示的范围：-1=< [X]原 <1-2-n**

* **若真值为纯整数，补码形式为XsX1X2…Xn（Xs表示符号位）**

例2：X1=1101 , X2=-1101 **[X1]补=0,1101 , [X2]补=1,0011**

**若表示纯小数，则能表示的范围：-2n =< [X]原 <2n**

* **在补码表示中，真值0的表示形式是唯一的**

**[+0]补=[-0]补=00000**

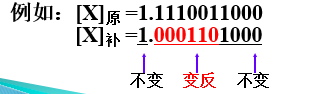
**3.由真值、原码转换为补码**

当X为正数时，[X]补=[X]原=X。

当X为负数时，由[X]原转换为[X]补的方法：

**①[X]原除掉符号位外的各位取反加“1”。**

**②自低位向高位，尾数的第一个“1”及其右部的“0”保持不变，左部的各位取反，符号位保持不变。**



## 移码表示法

**移码就是在真值X上加一个常数（偏置值），相当于X在数轴上向正方向平移了一段距离。移码也可称为增码或偏码。**

**[X]移=偏置值+X**

**字长n+1位定点整数的移码形式为X0X1X2…Xn**

最常见的移码的偏置值为2n。当字长8位时，偏置值为27。

例1：X=1011101

**[X]移=27+X=10000000+1011101=11011101**

例2：X=-1011101

**[X]移= 27 +X=10000000-1011101=00100011**

偏置值为2n的移码具有以下特点：

⑴ 在移码中，**最高位为“0”表示负数，最高位为“1”表示正数**。

⑵ 移码为全0时，它所对应的真值最小，为全1时，它所对应的真值最大。

⑶ **真值0在移码中的表示形式是唯一的**，即[+0]移=[-0]移=100…0。

⑷ 移码把真值映射到一个正数域，所以可将移码视为无符号数，直接按无符号数规则比较大小。

**练习：**

1. 四位机器内的数值代码1001，它所表示的十进制真值为（ **D** ）

A. 9 B. -1 C. -7 D. 以上三者均有可能

2. 在机器数（ **B** ）中，零的表示形式是唯一的。

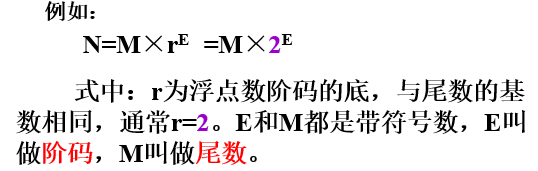
A.原码 B. 补码和移码 C.反码 D. 原码和反码

3. 设寄存器内容为11111111，若它等于+127，则为（ **D** ）

A. 原码 B. 反码 C. 补码 D. 移码

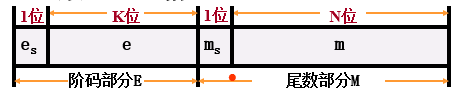
# 机器数的浮点表示法

小数点的位置根据需要而浮动，这就是浮点数



在大多数计算机中，尾数为纯小数，常用**原码或补码**表示；阶码为纯整数，常用**移码或补码**表示。

**浮点数的一般格式：**



**1.浮点数的阶码和尾数的含义**

尾数，表示这个数的精度

阶码，表示这个数的范围

虽然尾数和阶码不同，但是能表示的数的个数是一样的

1. **阶码小，表示数的范围小，故尾数就多，表示这个数的精度就很大**
2. **阶码大，表示数的范围大，故尾数就少，表示这个数的精度就很小**

**2.规格化的浮点数**

**（1）为什么要引入规格化的数？**

① 为了提高运算的精度

② 为了使一个浮点数的表示形式是唯一的

**（2）规格化浮点数的特点：**

规格化浮点数的位数M的绝对值应该在下列范围内：

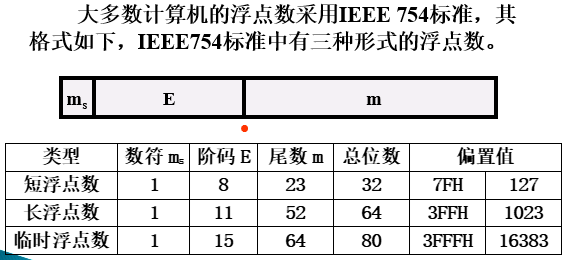
**1/r ≤|M| ＜ 1**

**如果r=2，则有1/2≤|M|＜1。**

**在尾数用原码表示时，规格化浮点数的尾数的最高数位总等于1。在尾数用补码表示时，规格化浮点数应满足尾数最高数位与符号位不同（ms⊕m1 =1），即当1/2≤M＜1时，应有0.1xx…x形式，当-1≤M＜-1/2时，应有1.0xx…x形式。**

**需要注意的是当M=-1/2，对于原码来说，是规格化数，而对于补码来说，不是规格化数；当M= -1时，对于原码来说，这将无法表示，而对于补码来说，这是一个规格化数。**

# IEEE754标准浮点数



**以短浮点数为例讨论浮点代码与其真值之间的关系**

* 最高位为数符位；其后是**8位阶码**，以2为底，**阶码的偏置值为127**；其余**23位是尾数**
* 为了使尾数部分能表示更多一位的有效值，IEEE754采用隐含尾数最高数位1（即这一位1不表示出来）的方法，因此尾数实际上是24位
* 注意，隐含的1是一位整数（即位权为20），在浮点格式中表示出来的23位尾数是纯小数，并用原码表示

**例1：将(100.25)10转换成短浮点数格式。**

**⑴ 十进制数→二进制数**

**(100.25)10=(1100100.01)2**

**⑵ 非规格化数→规格化数**

**1100100.01=1.10010001×26**

**⑶ 计算移码表示的阶码（偏置值＋阶码真值）**

**1111111+110=10000101**

**⑷ 以短浮点数格式存储该数。**

**符号位=0**

**阶码=10000101**

**尾数=10010001000000000000000**

**短浮点数代码为**

**0;100 0010 1;100 1000 1000 0000 0000 0000**

**表示为十六进制的代码：42C88000H。**

**例2：把短浮点数C1C90000H转换成为十进制数。**

**⑴ 十六进制→二进制形式，并分离出符号位、阶码和尾数。**

**C1C90000H=1;10000011;10010010000000000000000**

**⑵ 计算出阶码真值（移码－偏置值）**

**10000011-1111111=100**

**⑶ 以规格化二进制数形式写出此数**

**1.1001001×24**

**⑷ 写成非规格化二进制数形式**

**11001.001**

**⑸ 转换成十进制数，并加上符号位。**

**(11001.001)2=(25.125)10**

**所以，该浮点数=-25.125**

**当E和m均为0的时候，表示机器0**

# 数据校验码

## 奇偶校验码

* **奇校验──整个校验码（有效信息位和校验位）中“1”的个数为奇数。**
* **偶校验──整个校验码中“1”的个数为偶数。**

**练习：**

1. 假定下列字符码中有奇偶校验位，但没有数据错误，采用奇校验的字符码是（ **D** ）

A. 11001010 B. 11010111 C. 11001100 D. 11001011

## 海明校验码

**海明码编码规律（假设编成海明码为HmHm-1…H2H1）：**

**例如：设有一个8位的信息为10101100，试求汉明编码的生成和校验过程**

1. 编码生成

**冗余位数：假设校验位的位数为r位，设有效信息码组为k位，应有关系：**

**2r-1**≥**k+r**

D8 D7 D6 D5 P4 D4 D3 D2 P3 D1 P2 P1

P1= D1⊕D2⊕D4⊕D5⊕D7**(P1初值取1)**

P2= D1⊕D3⊕D4⊕D6⊕D7**(P2初值取1)**

P3= D2⊕D3⊕D4⊕D8**(P3初值取1)**

P4= D5⊕D6⊕D7⊕D8**(P4初值取0)**

（2）故可得到用二进制表示的汉明码为

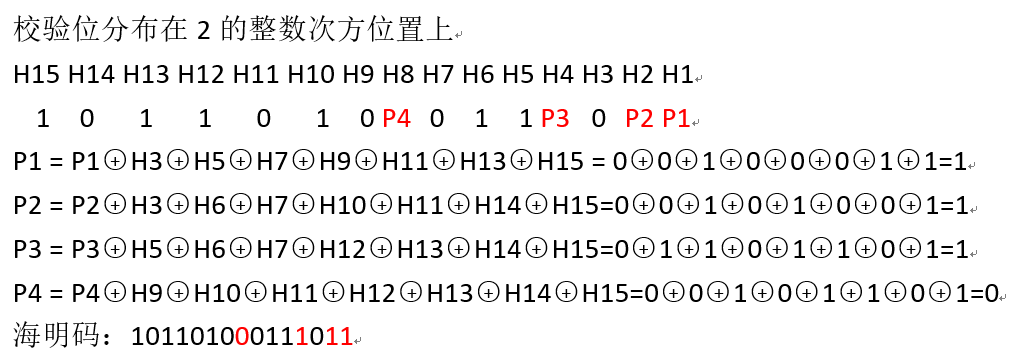
**101001101011**

**练习：**

1. （2013）用海明码对长度为8位的数据进行检/纠错时，若能纠正一位错，则校验位数至少是（ **C** ）

A. 2 B. 3 C. 4 D. 5

2. 求数据10110100110的海明码，可检/纠错一位错。



第三章 指令系统

# 指令格式

## 机器指令的基本格式



**操作码：指明操作的性质及功能。**

**地址码：指明操作数的地址，特殊情况下也可能直接给出操作数本身。**

## 地址码结构

**一条双操作数指令除操作码之外，还应包含以下信息：**

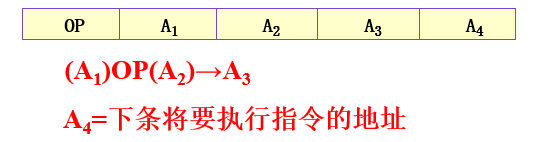
**第一操作数地址，用A1表示；**

**第二操作数地址，用A2表示；**

**操作结果存放地址，用A3表示；**

**下条将要执行指令的地址，用A4表示。**

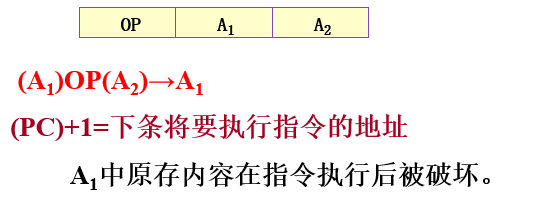
**1.四地址指令（执行一条四地址指令需5次访问主存。）**



**2.三地址指令（执行一条三地址指令需要4次访问内存）**



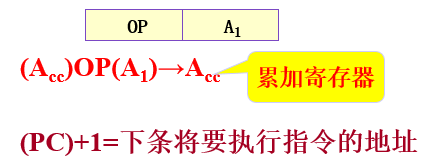
**3.二地址指令（执行一条二地址指令需要4次访问内存）**



**目的操作数地址**

**源操作数地址**

**4.一地址指令（执行一条二地址指令需要2次访问内存）**



**5.零地址指令**



操作数地址是隐含的。参加运算的操作数放在**堆栈的栈顶和次栈顶单元**中，运算结果也放在堆栈中。有关堆栈的概念将在稍后讨论。

## 扩展操作码法

**例如：设某机的指令长度为16位，操作码字段为4位，有三个4位的地址码字段，其格式为：**



如果按照定长编码的方法，4位操作码字段最多只能表示16条不同的三地址指令。



**练习：**

1. 一地址指令中另一个操作数存放在（ **累加器** ）

2. 零地址指令的操作数存放在（ **堆栈** ） 。

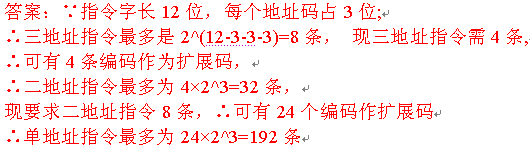
3. 下条指令的地址存放在（**程序计数器/PC** ）。

4. 某机器字长32位，指令单字长，指令系统中具有二地址指令、一地址指令和零地址指令各若干条，已知每个地址长12位，采用扩展操作码方式，问该指令系统中的二地址指令、一地址指令、零地址指令各最多能有多少条？

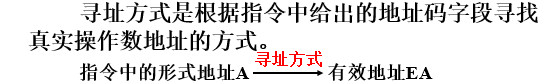
5. 指令字长为12位，每个地址码为3位，采用扩展操作码的方式，设计4条三地址指令、16条二地址指令、64条一地址指令和16条零地址指令。

给出一种操作码的扩展方案计算操作码的平均长度

6. 若某机为定长指令字结构，要求：三地址指令4条，二地址指令8条，单地址指令180条。设指令字长为12位.每个地址码长为3位。问能否以扩展操作码为其编码?如果其中单地址指令为254条呢?说明其理由。



# 寻址技术



**1.立即寻址**



指令中的操作码字段后面的部分不是通常意义上的操作数地址，而是操作数本身，也就是或数据就包含在指令中，只要取出指令，也就取出了可以立即使用的操作数。

特点：速度快；立即数大小受限

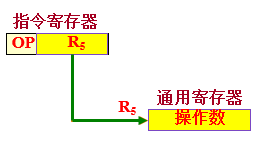
**2.寄存器寻址**

指令中地址码部分给出某一通用寄存器的编号，所指定的寄存器中存放着操作数。

两个明显的优点：

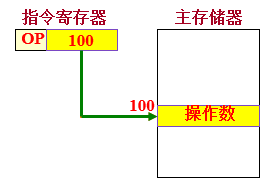
•从寄存器存取数据比主存快得多；

•由于寄存器的数量较少，其地址码字段比主存单元地址字段短得多。



**3.直接寻址**

指令中地址码字段**给出的地址A就是操作数的有效地址：EA=A**



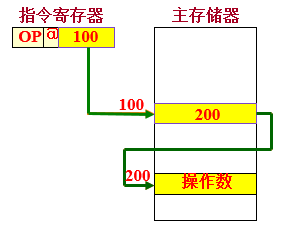
**4.间接寻址**

**指令中给出的地址A不是操作数的地址，而是存放操作数地址的地址。**

**EA=(A)**

通常在指令格式中划出一位**＠**作为标志位。

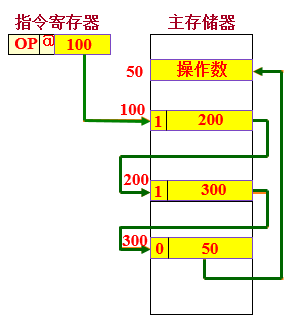
＠=0 直接寻址 ＠=1 间接寻址



除去一级间接寻址外，还有多级间接寻址。多级间接寻址为取得操作数需要多次访问主存，即使在找到操作数有效地址后，还需再访问一次主存才可得到真正的操作数。

**多级间接标志：**

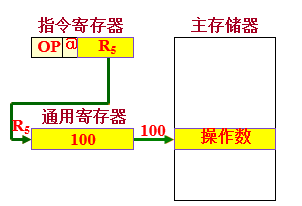
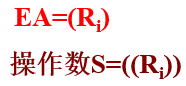
**0：找到有效地址** **1：继续间接寻址**



**5.寄存器间接寻址**

指令中的地址码给出某一通用寄存器的编号，被指定的寄存器中存放**操作数的有效地址**，而操作数则存放在**主存单元**中。

**这种寻址方式的指令较短，并且在取指后只需一次访存便可得到操作数。**

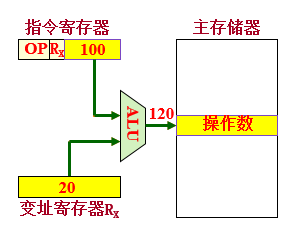


**6.变址寻址**

把指令给出的形式地址A与变址寄存器RX的内容**相加**，形成操作数有效地址：

**EA= (RX) +A** （**RX 的内容为变址值。**）

变址寻址是一种广泛采用的寻址方式，通常指令中的形式地址作为基准地址，而RX的内容作为修改量。在遇到需要频繁修改地址时，无须修改指令，只要修改变址值就可以了。

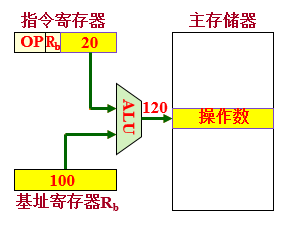


**7.基址寻址**

将基址寄存器Rb的内容与位移量D相加，形成操作数有效地址：

**EA=(Rb)+D**

**基址寄存器的内容称为基址值，指令的地址码字段是一个位移量，位移量可正可负。**

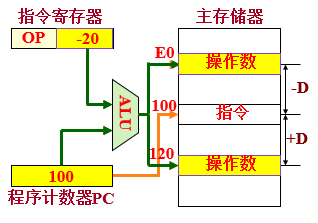
基址寻址和变址寻址在形成有效地址时所用的算法是相同的，而且在一些计算机中，这两种寻址方式都是由同样的硬件来实现的。

但这两种寻址方式**应用的场合**不同，变址寻址是面向**用户**的，用于访问字符串、向量和数组等成批数据；而基址寻址面向**系统**，主要用于逻辑地址和物理地址的变换，用以解决程序在主存中的再定位和扩大寻址空间等问题。在某些大型机中，基址寄存器只能由特权指令来管理，用户指令无权操作和修改。

**8.相对寻址**

相对寻址是基址寻址的一种变通，**由程序计数器PC提供基准地址**，即：

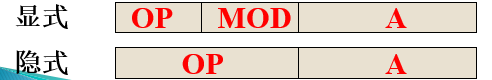
**EA=(PC)+D** 位移量指出的是操作数和现行指令之间的**相对位置**。



为了能区分出各种不同寻址方式，必须在指令中给出标识。标识的方式通常有两种：**显式和隐式**。

显式的方法就是在**指令中设置专门的寻址方式（MOD）字段**，用二进制代码来表明寻址方式类型。

隐式的方式是由**指令的操作码字段说明指令格式并隐含约定寻址方式**。



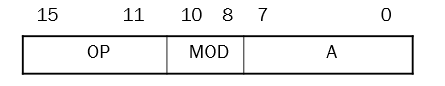
1.根据操作数所在位置，指出其寻址方式：

操作数在寄存器中，称为 **寄存器** 寻址方式；操作数地址在寄存器中，称为

**寄存器间接** 寻址方式；操作数在指令中，称为 **立即** 寻址方式；操作数地址在指令中，为 **直接** 寻址方式。操作数地址，为某一寄存器的内容与位移之和，则可以是 **基址寻址** 、 **变址寻址** 、 **相对寻址** 寻址方式。

2. 某计算机的字长为16位，存储器按字编址，访存指令格式如图所示：

其中，OP是操作码，MOD定义寻址方式，A为形式地址。设PC和Rx分别为程序计数器和变址寄存器，字长为16位，问：



1.该格式能定义多少种指令？

2.各种寻址方式的寻址范围是多少字？

3.写出各种寻址方式的有效地址EA的计算公式。

3. (2011)偏移寻址通过将某个寄存器内容与一个形式地址相加而生成有效地址。下列寻址方式中，不属于偏移寻址方式的是( **A** )

A.间接寻址 B.基址寻址 C.相对寻址 D.变址寻址

3. (2013)假设变址寄存器R的内容为1000H，指令中的形式地址为2000 H；地址1000H中的内容为2000H，地址2000H中的内容为3000H，地址3000H中的内容为4000H，则变址寻址方式下访问到的操作数是( **D** )

A. 1000H B. 2000H C. 3000H D. 4000 H

4. 指令系统中采用不同的寻址方式的目的主要是（ **B** ）

A. 实现存储程序和程序控制

B. 缩短指令长度，扩大寻址空间，提高编程灵活性

C. 可以直接访问外存

D. 提供扩展操作码的可能性并降低指令译码难度

# 堆栈与堆栈操作

**堆栈是一种按特定顺序进行存取的存储区，这种特定顺序可归结为“后进先出”（LIFO）或“先进后出”（FILO）。**

**堆栈主要用来暂存中断断点、子程序调用时的返回地址、状态标志及现场信息等，也可以用于子程序调用时的参数的传递。**

堆栈结构包括寄存器堆栈（**硬堆栈**）、存储器堆栈（**软堆栈**）

# 输入/输出类指令

**输入/输出（I/O）类指令用来实现主机与外部设备之间的信息交换，包括输入/输出数据、主机向外设发控制命令或外设向主机报告工作状态等。从广义的角度看，I/O指令可以归入数据传送类。**

**各种不同的计算机的I/O指令差别很大，通常有两种方式：**

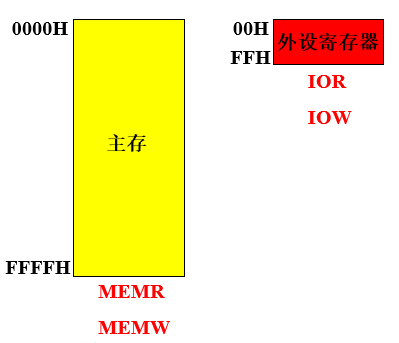
**独立编址方式**

**统一编址方式**

## 独立编址的I/O

所谓独立编址就是把外设端口和主存单元分别独立编址。指令系统中有专门的**IN/OUT**指令。以主机为基准，信息由外设传送到主机称为输入，反之称为输出。指令中需要给出外设端口地址。这些端口地址与主存地址无关，是另一个独立的地址空间。

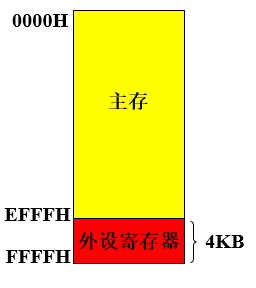
**输入/输出指令IN/OUT**



**访存指令load/store**

## 统一编址的I/O

**所谓统一编址就是把外设寄存器和主存单元统一编址。指令系统中没有专门的I/O指令，就用一般的数据传送类指令来实现I/O操作。**



1. 某个主存单元，按照（**A**）编址时，需要的地址码位数最少。

A. 字编址 B. 位编址 C. 字节编址 D. 以上选项一样多

2. 以下编址方式中外设有专用的I/O指令的是（ **C** ） 。

A.字节编址 B. 字编址 C.独立编址 D.统一编址

3. 某主存空间容量是1GB，字长32位，如果按照字编址，其地址码位数是（ **D** ）。

A.15 B. 30 C. 25 D. 28

4. 在统一编址方式下，下面的说法（**D**）是正确的。

A. 一个具体的地址只能对应输入输出设备。

B. 一个具体的地址只能对应内存单元。

C. 一个具体地址既可对应输入输出设备又可对应内存单元。

D. 只对应I/O设备或者对应内存单元。

5. 在独立编址方式下，下面的说法（**C**）是正确的。

A. 一个具体的地址只能对应输入输出设备。

B. 一个具体的地址只能对应内存单元。

C. 一个具体地址既可对应输入输出设备又可对应内存单元。

D. 只对应I/O设备或者对应内存单元。

6. 在独立编址方式下，存储单元和I/O设备是靠（**C**）来区分的。

A .不同的地址代码 B. 不同的地址总线

C. 不同的指令或不同的控制信号

D. 上述都不对

7. 在统一编址方式下，存储单元和I/O设备是靠（**A**）来区分的。

A. 不同的地址代码 B. 不同的地址总线

C. 不同的指令或不同的控制信号

D. 上述都不对

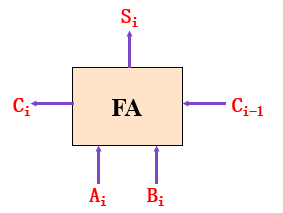
第四章 数值的机器运算

# 基本算术运算的实现

加法器是由全加器再配以其他必要的逻辑电路组成的。

**1.全加器（一个全加器可进行一位的加法运算）**

**基本的加法单元称为全加器，它要求三个输入量：操作数Ai和Bi、低位传来的进位Ci-1，并产生两个输出量：本位和Si、向高位的进位Ci。**



**全加器的逻辑表达式为**

**S*i*=A*i*⊕B*i*⊕C*i*-1** **C*i*=A*i*B*i*+(A*i*⊕B*i*)C*i*-1**

## 串行进位和并行进位

**进位产生函数用Gi表示**

**进位表达式**

**进位传递函数用Pi表示**

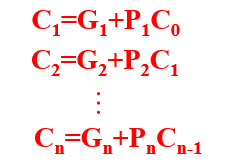
**Ci=AiBi+(Ai⊕Bi)Ci-1**

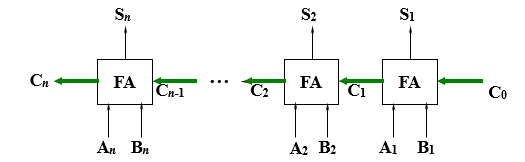
**Gi=AiBi（进位产生函数）的含义是：若本位的两个输入均为1，必然要向高位产生进位。**

**Pi=Ai⊕Bi（进位传递函数）的含义是：当两个输入中有一个为1，低位传来的进位Ci-1将超越本位向更高的位传送。**

**串行进位**

**把n个全加器串接起来，就可进行两个n位数的相加。串行进位又称行波进位，每一级进位直接依赖于前一级的进位，即进位信号是逐级形成的。**

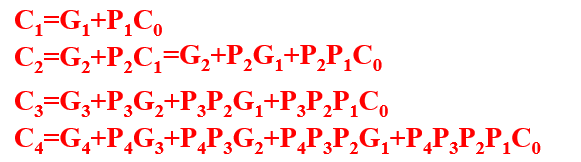




**串行进位链的总延迟时间与字长成正比。假定，将一级门的延迟时间定为ty，从上述公式中可看出，每形成一级进位的延迟时间为2ty。在字长为n位的情况下，若不考虑Gi、Pi的形成时间，从C0→Cn的最长延迟时间为2nty。**

**并行进位方式**

**并行进位又叫先行进位、同时进位，其特点是各级进位信号同时形成。**



# 定点加减运算

**补码加减运算**

**1.补码加法**

**两个补码表示的数相加，符号位参加运算，且两数和的补码等于两数补码之和，即** **[X+Y]补=[X]补+[Y]补**

**2.补码减法**

**根据补码加法公式可推出：** **[X-Y]补=[X+(-Y)]补=[X]补+[-Y]补**

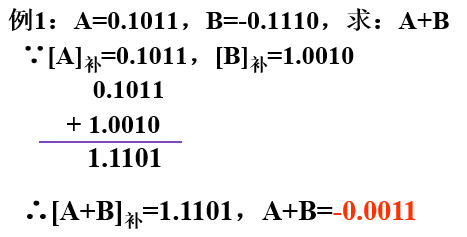
已知[Y]补求[-Y]补的方法是：将[Y]补连同符号位一起求反，末尾加“1”。

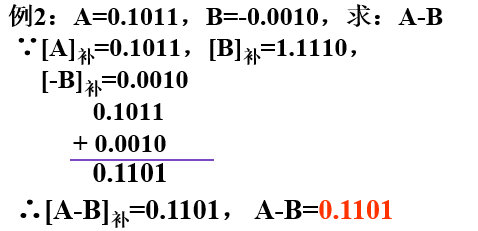
[-Y]补被称为[Y]补的**机器负数**，由[Y]补求[-Y]补的过程称为对[Y]补变补（求补），表示为：

**[-Y]补=[[Y]补]变补**

例1：Y=-0.0110 [Y]补=1.1010，[-Y]补=0.0110

例2：Y=0.0110 [Y]补=0.0110，[-Y]补=1.1010





# 补码的溢出判断与检测方法

**（1）采用一个符号位**

两正数相加，结果为负表明产生**正溢**；两负数相加，结果为正表明产生**负溢**。因此可得出采用一个符号位检测溢出的方法：

当Xs=Ys=0，Ss=1时，产生正溢。

当Xs=Ys=1，Ss=0时，产生负溢。

**（2）采用进位位**

两数运算时，产生的进位为Cs,C1C2…Cn，

其中：Cs为符号位产生的进位，C1为最高数值位产生的进位。

两正数相加，**当最高有效位产生进位（C1=1）而符号位不产生进位（Cs=0）时**，**发生正溢。**

两负数相加，**当最高有效位没有进位（C1=0）而符号位产生进位（Cs=1）时，发生负溢。**

**（3）采用变形补码（双符号位补码）**

双符号位的含义如下：

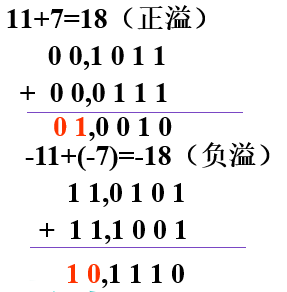
Ss1Ss2=00 结果为正数，无溢出

**Ss1Ss2=01 结果正溢**

**Ss1Ss2=10 结果负溢**

Ss1Ss2=11 结果为负数，无溢出

当两位符号位的值不一致时，表明产生溢出。



# 带符号数的移位和舍入操作

原码的移位规则：**无论是正数还是负数，在左移或右移的时候，符号位不变，空出位一律以‘0’补入。**

补码的移位规则：**对于正数**，在左移或右移的时候，符号位不变，**空出位一律以‘0’补入**。**对于负数来说，在左移的时候，以‘0’补入，在右移的时候，以‘1’补入。**

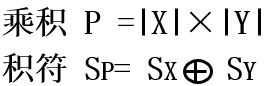
**带符号数的舍入操作：（了解）**

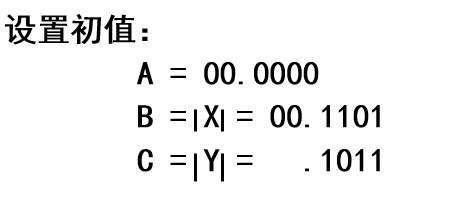
**1.恒舍（切断）2. 冯·诺依曼舍入法3. 下舍上入法4.查表舍入法**

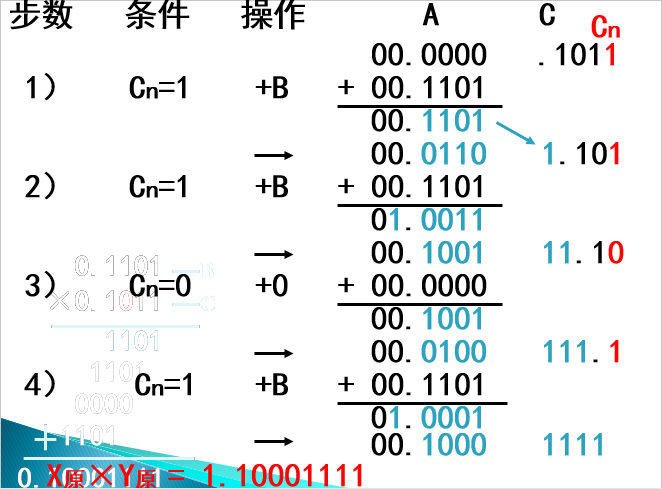
# 定点乘法运算

**原码一位乘法**

**例. 0.1101×1.1011**







**补码一位乘法**

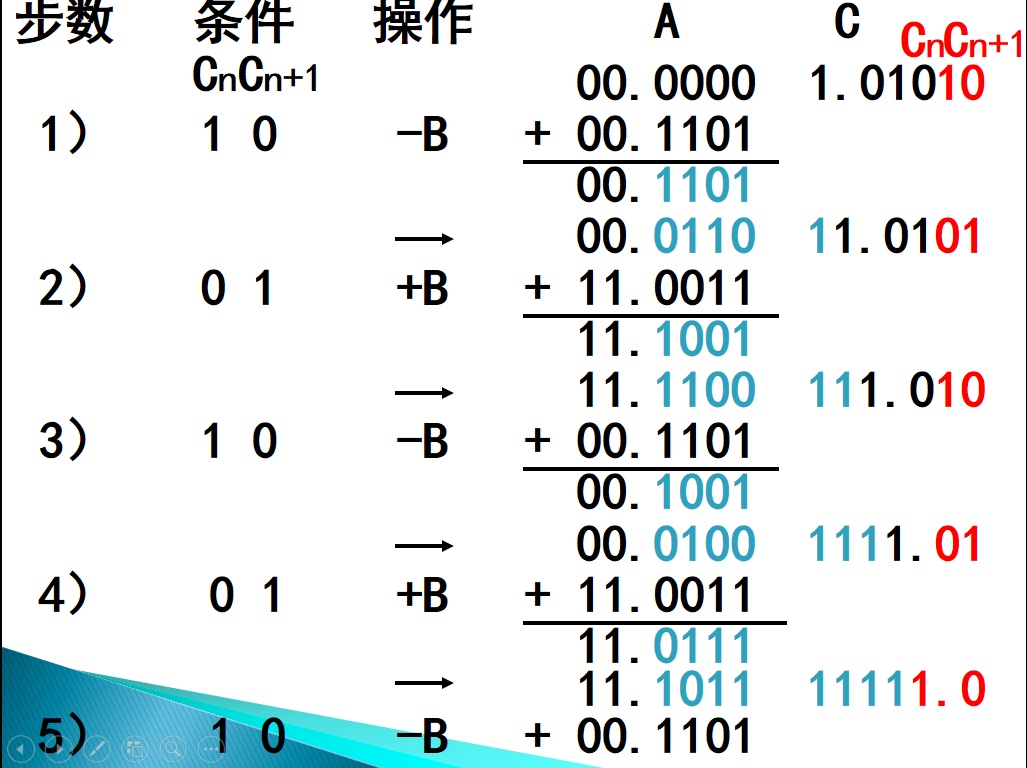


**运算实例：**

**X=-0.1101,Y=-0.1011,求(X\*Y)补。**

**初值：A=00.0000,B=X补=11.0011,**

**-B=(-X)补=00.1101,C =Y补=1.0101**



# 浮点加减运算

**(1)对阶**

**对阶的规则是：小阶向大阶看齐。**

**(2)尾数加/减**

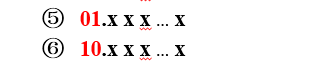
**对阶之后，就可以进行尾数加/减，即**

**MA±MB→MC**

**(3)尾数结果规格化**



**尾数每左移一位，阶码相应减1（EC-1→EC），直至成为规格化数为止。（左规可能需进行多次）只要满足下列条件：**



**尾数每右移一位，阶码相应加1（EC+1→EC）。 （右规最多进行一次）右规的条件如下：右规=Cs1⊕Cs2**

**(4)舍入**

**(5)溢出判断**

第五章 存储系统和结构

# 存储器分类

**(1)随机存取存储器RAM**

CPU可以对RAM单元的内容随机地读写访问。CPU对任何一个存储单元的读写时间是一样的，即存取时间是相同的。

**(2)只读存储器ROM**

ROM可以看作RAM的一种特殊方式，存储器的内容只能随机读出而不能写入。（在进行字位扩展的时候，需要进行注意数据线的方向，只能读入不能写入）

## 存储系统层次结构

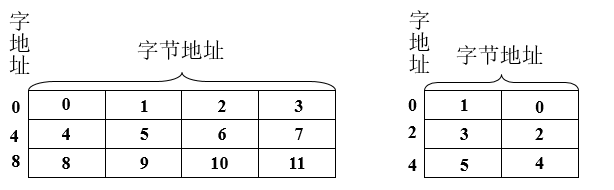
**Cache存储系统是为解决主存速度不足而提出来的。**



**虚拟存储系统是为解决主存容量不足而提出来的。**



# 大端方案和小端方案

假设一个字由四个字节组成，我们使用B3、B2、B1、B0来分别表示这四个字节，其中B3是字的最高有效字节，B0是最低有效字节。



图a是**大端方案**（高位地址作为字的地址），图b是**小端方案**（低位地址作为字的地址）

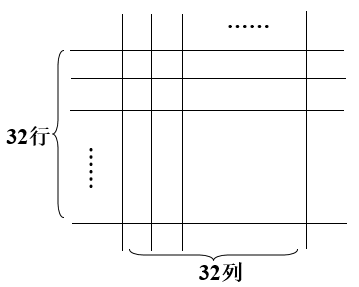
# 静态RAM和动态RAM

**静态RAM，即SRAM（Static RAM），其存储电路以双稳态触发器为基础；动态RAM，即DRAM（Dynamic RAM），其存储电路以电容为基础。**

例题：对于64K X 8，需要地址线多少根，对于静态RAM需要16根，对于动态RAM则需要8根，因为动态RAM有地址复用技术

## 动态RAM的刷新

例如，对具有1024个记忆单元（排列成32×32矩阵）的存储芯片进行刷新，刷新是按行进行的，且每刷新一行占用一个存取周期，存取周期为500ns（0.5 μs）。

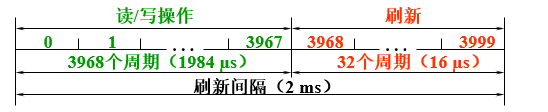


**(1)集中刷新方式**

**在允许的最大刷新间隔内，按照存储芯片容量的大小集中安排若干个刷新周期，刷新时停止读写操作。（这一段停止读写的时间称为死区）**

**刷新时间=存储体矩阵行数×刷新周期**

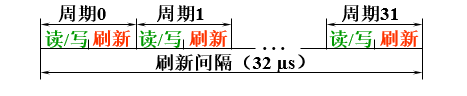
**这里刷新周期是指刷新一行所需要的时间（刷新是按行刷新的）**



死区时间：行数x刷新每行的时间  
死时间率：死区时间/总时间,总时间一般是2ms

**(2)分散刷新方式**

**分散刷新是指把刷新操作分散到每个存取周期内进行，此时系统的存取周期被分为两部分，前一部分时间进行读/写操作或保持，后一部分时间进行刷新操作。一个系统存取周期内刷新存储矩阵中的一行。**



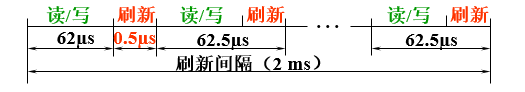
**缺点：尤其是当存储容量比较小的情况下，没有充分利用所允许的最大刷新间隔（2ms）。**

**(3)异步刷新方式**

**异步刷新方式可以看成前述两种方式的结合，它充分利用了最大刷新间隔时间，把刷新操作平均分配到整个最大刷新间隔时间内进行，故有：**

**相邻两行的刷新间隔=最大刷新间隔时间/行数**

对于32×32矩阵，在2ms内需要将32行刷新一遍，所以相邻两行的刷新时间间隔=2ms/32=**62.5 μs**，即每隔**62.5 μs安排一个刷新周期**，在刷新时封锁读/写。

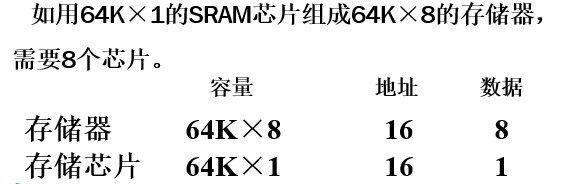


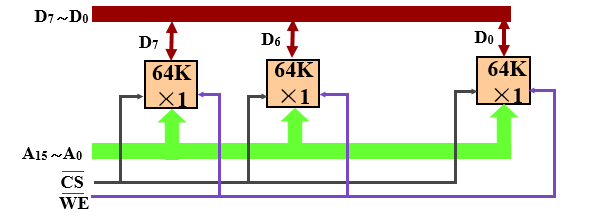
**异步刷新方式虽然也有死区。但比集中刷新方式的死区小得多，仅为0.5 μs。**

# 主存储器的连接与控制

## 位扩展

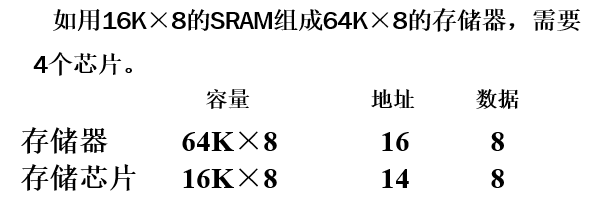
**位扩展指只在位数方向扩展（加大字长），而芯片的字数和存储器的字数是一致的。**

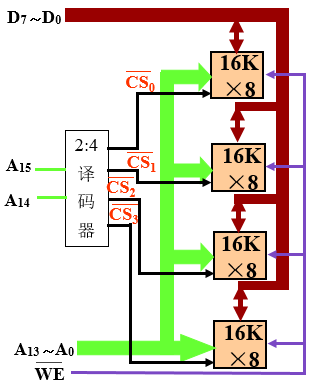




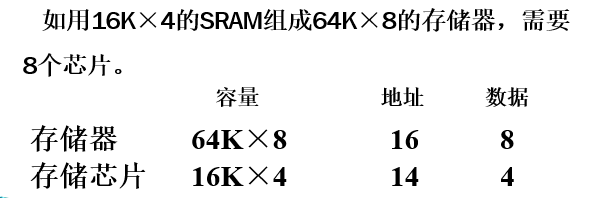
## 字扩展

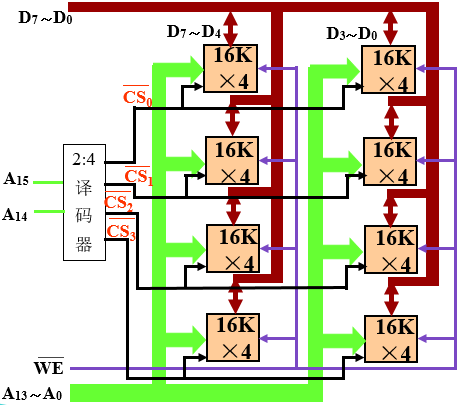
**字扩展是指仅在字数方向扩展，而位数不变。**





## 字和位同时扩展





**存储芯片的地址分配和片选**

* **线选法**
* **全译码法**
* **部分译码法。**

## 例题

**用2114（1K×4）SRAM芯片组成容量为4K×8的存储器。地址总线A15～A0（低）,双向数据总线D7～D0（低）,读/写信号线R/W。**

**给出芯片地址分配与片选逻辑,并画出M框图。**

