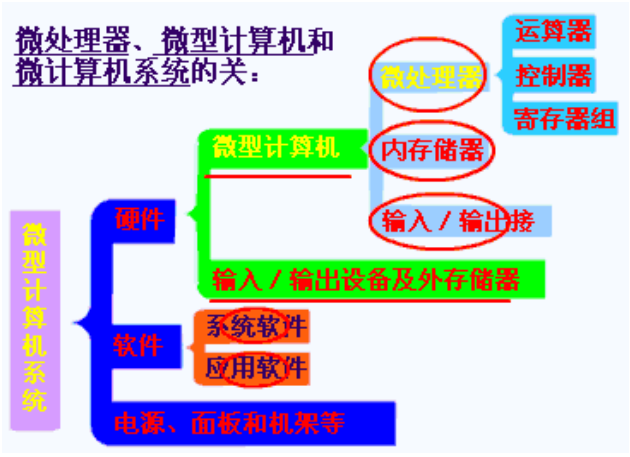
第一章　绪论

- 1.1 微机发展史
  - 1. 1946，*ENIAC*，第一台电子计算机(美，宾夕法尼亚大学，埃克特、莫克利)
    缺点：一、存储容量太小；二、用线路连接的方式来编排程序。
    冯·诺依曼、莫尔小组，EDVAC，奠定了计算机基础
    - 冯·诺依曼机
      1. 组成：运算器、控制器、存储器、输入设备、输出设备
      2. 基本原理：程序存储和控制
      3. 数值采用二进制计数
      发展：早期冯·诺依曼以CPU为中心，现在演变为以存储器为中心
  - 2. 1974，*Altair*，开创微机新时代
  - 3. 微机4代
    - 电子管　(1946～1956)
    - 晶体管　(1957～1963)
    - 中、小规模集成电路　(1964～1970)
    - 大、超大规模集成电路(1971～)
  - 4.
    - 大型机 → 中型机 → 小型机 → 微型机
    - 发展方向
      - 大型巨型化：性能
      - 小型微型化：成本
      - 网络、人工智能化
  - 5. 1995，摩尔定律
  - 6. 1971 CPU(Intel4004)问世：开创微机新纪元
  - 7. CPU按字长换代
    - 第一代4.8（1971）：
      - 1971：Intel4004, PMOS, 2300, <1MHz, 10μs～15μs
      - 1972：Intel8008
    - 第二代8（1974）：
      - 1974：Intel8080(NMOS, 4500, 2MHz, 1μs～2μs, 64KB, 单), Motorola MC6800
      - 1975：Zilog Z80
      - 1976：Intel 8085
    - 第三代16（1978）：
      - 1978：Intel 8086(NMOS, 29000万，5、8、10MHz, 0.5μs, 1MB)
      - 1979：Zilog Z8000, Mototola 6800
      - 1982：Intel 80826, Motorola 68010
    - 第四代32（1983）
      - 1983：Zilog Z80000
      - 1984：Motorola 68020
      - 1985：Intel 80386(CHMOS, 27.5万，16～33MHz，<0.1μs, 32, 40GB,)
      - 1989：Intel 80486(CHMOS, 120万,25/33/50MHz), Motorola 68040
    - 第五代
      - 1993：Pentium(CMOS, 310万,60MHz) 80586
      - 1995：Pentium Pro(550万,150/166/180/200MHz) 80686
      - 1997：Pentium Ⅱ(750万)
      - 1999：Pentium Ⅲ(950万)
      - 2000：Pentium 4(1.5GHz～3.6GHz)
      - 2006：*Core*(*SSSE*3)

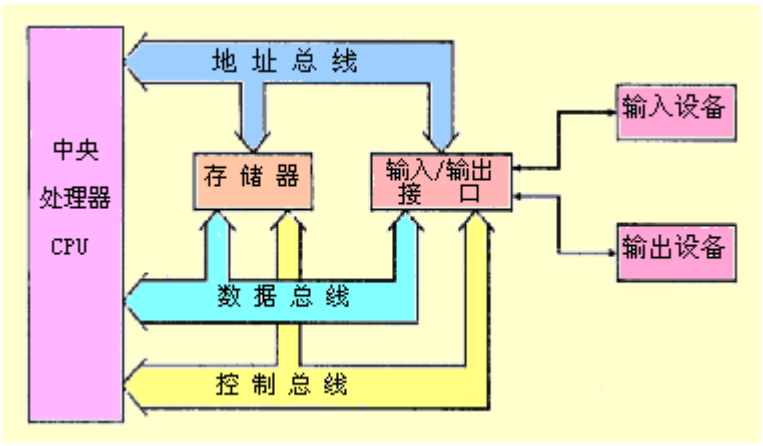
1.2微机系统
- 冯•诺依曼
  - 基本结构:CPU(运算器、控制器)、存储器、输入/输出设备
  - 工作原理：存储器存储程序控制的原理
  - 发展：早期冯•诺依曼以CPU为中心，现在演变为以存储器为中心



- 1.2.1微型计算机
  - 1 微机
    - 1. 微处理器
      - 核心，*CPU*(中央处理器)
      - 微处理器 / CPU =运算器+控制器
      - 微控制器 / 单片机 =运算器+控制器+存储器+I/O接口
      - 任务
        - 1.从存储器中取指令，指令译码
        - 2.简单的算术逻辑运算
        - 3.在处理器和存储器或*I/O*之间传送数据
        - 4.程序流向控制等
    - 2.存储器：分类
      - 随机存储器RAM
      - 只读存储器ROM

      存放程序和数据
    - 3.I/O接口：外部设备与CPU或存储器连接交换信息
    - 4. 系统总线
      - 连接CPU、存储器、I/O接口
      - 任务：CPU与存储器、I/O间传送地址、数据、控制信息
      - 分类
        - 数据总线
        - 地址总线
        - 控制总线
    - 5. I/O设备
      - 输入设备
        - 键盘、鼠标、扫描仪、数字化仪、
        - 条形码读入器、光笔、语音输入器
      - 输出设备：CRT显示器、打印机、绘图仪
      - I/O公用设备：软盘机，硬盘机，光盘机
  - 2 微机系统
    - 组成：（主体）微机、I/O设备、系统软件
    - 输入设备：键盘、鼠标、扫描仪、数字化仪、条形码读入器
    - 输出设备：CRT显示器、打印机、绘图仪
  - 3CPU
    - 算术逻辑部件ALU：算术、逻辑运算
    - 累加器和寄存器组
    - 指令指针寄存器IP（程序计数器）：指向要执行的下一条指令的偏移地址
    - 段寄存器：段地址与偏移地址组成20位物理地址对存储器寻址
    - 时序和控制部件
    - 内部总线
  - 4微机性能指标
    - 1主频2字长3内存容量4存取周期5运算速度6内核数目7高速缓冲

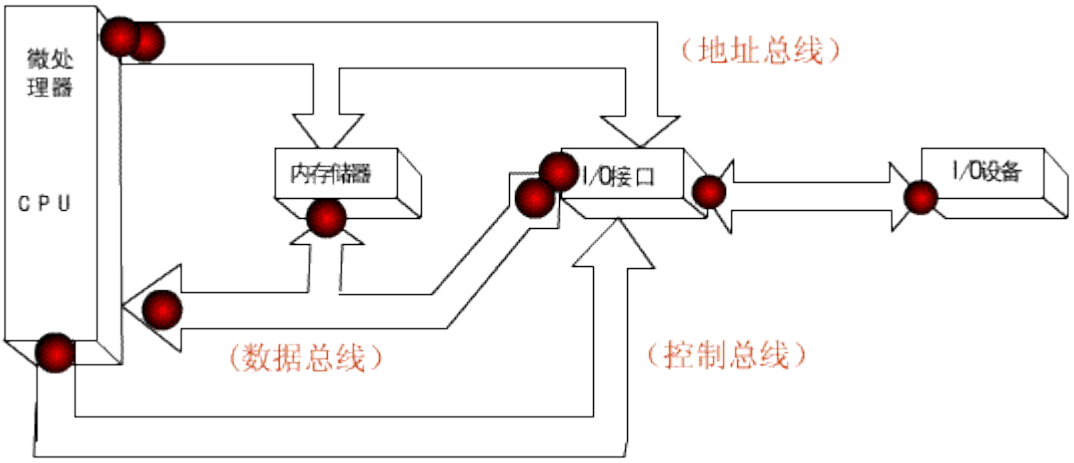
- 4.存储器
  - 内存
    - 半导体存储器
      - 分类
        - 随机RAM
          - 随机读写，断电存储内容消失
          - 分类
            - 动态DRAM
              - 特点：容量大速度慢
              - 存储单元
                - MOS电路　　内存条
                - 电容：漏电 → 信息消失 → 刷新
            - 静态SRAM
              - 特点：容量小速度高
              - 存储单元：触发器
                - 双极型电路
                - 或　　不要刷新
                - MOS电路
            - 非易失性随机读写存储器NVRAM
              - 断电不丢失数据
              - SRAM实时读写功能与EEPROM可靠非易失功能综合
        - 只读ROM
          - 只读不写，非易失性
          - 分类
            - 可编程PROM
              - 原理：专有设备烧断芯片内二极管而存储二进制信息
              - ⇒只能写一次，固化程序用
            - 可擦除可编程EPROM
            - EEPROM(电可擦除EPROM)
        - 高速缓冲存储器Cache：存储空间较小，速度较慢，位于CPU和主存之间
  - 外存
    - 磁记录：软盘、硬盘、磁带
    - 光盘存储器：
      - 1只读光盘
      - 2一次写入光盘 ⇒ 大容量、标准化、重量轻、易保存、寿命长
      - 3可改写光盘
    - Flash存储器：（优盘）将数据以电荷的形式储存在光栅电极上，集成度高，可靠性好，不易损坏



- 1.2.2 I/O接口
  - 作用：CPU（或存储器）与外设交换信息。
  - 主要接口芯片
    - 锁存器74LS373、缓冲器74LS245
    - 可编程中断控制器8259A：控制中断过程
    - 可编程技术/定时器8254：对DRAM定时刷新、扬声器定时发声等实时控制信号
    - 可编程并行接口芯片8255：CPU与外设间8位数据并行传送，打印机、CRT接口等
    - 可编程串行接口8251A：远距离传送数据，同步/异步数据收发器
    - 可编程DMA控制器8237A：大批量数据传送，DMA工作方式（外设直接与存储器交换数据）
    - A/D,D/A转换芯片：数字量与模拟量转换
- 1.2.3总线
  - 1.总线描述：各部件间传送信息的公共通道。
  - 2.总线标准特性
    - 1.物理特性：总线根数、插头、插座形状、引脚排列等
    - 2.功能特性：地址、数据、控制总线
    - 3.电器特性：IN/OUT
    - 4.时间特性：有效时间
  - 3.总线分类
    - 1内部总线
    - 2元件级总线
      - 1地址总线{单向/三态
      - 2数据总线{双向/三态
      - 3控制总线{复杂
    - 3系统总线
      - ISA
      - EISA
      - VESA
      - PCI
      - PCI Express
    - 4外部总线
      - USB：通行串行总线
      - IEEE 1394：火线接口
  - 4总线结构
    - 1单总线结构
    - 2面向CPU的双总线结构
    - 3面向主存储器的双总线结构

- 3.总线分类
  - 1.内部总线
    - 单总线：集成度、成品率提高
    - 双、三总线：加快数据传送
  - 2.元件级总线
    - 1.地址总线：
      - 是CPU向存储器或I/O端口传送地址的，是三态单向总线
      - 地址总线位数决定CPU可直接寻址的内存容量
        - 16位：地址总线20位，1MB
        - 32位：地址总线32位，4GB
    - 2.数据总线
      - CPU与存储器及外设交换数据，是三态双向总线
      - 8088：内部字长16位，外部数据总线8位 ⇒ 准16位微处理器
    - 3.控制总线
      - 传输控制信号，控制总线宽度根据系统需要而定，一般为8位
  - 3.系统总线
    - 定义：微处理器机箱内底板总线，连接插件板
    - 80x86
      - 1.ISA总线：工业标准体系结构总线，IBM,16位，80386，80486
      - 2.EISA总线：扩展工业体系结构总线,32位
      - 3.VESA总线：视频标准协会，全开放通用总线,32,80486
      - 4.PCI总线：外设互连局部总线，Intel,16/32位，Pentium，即插即用
      - 5.PCI Express总线：最新总线和接口标准，点对点串行连接
  - 4.外部总线
    - 描述：系统与系统、系统与外设间信息通路，数据传输方式：并行和串行
    - 1.USB总线
      - 通行串行总线（Universal Serial Bus）
      - 速度：480，15，1.5MB/s
      - 特点
        - 即插即用，无需单独供电系统，可支持127个外设同时连接到USB总线
        - 集中控制策略，占用资源少，通过4线串行电缆访问USB设备
    - 2.IEEE 1394
      - 火线接口（Fire Wire），苹果公司
      - 速度：400，800Mbps
      - 特点
        - 高速率、容许用户直接透过IEEE 1394接口编辑电子影像档案
        - 主要用于：数字成像和广播电视领域

- 4.总线结构
  - 1.单总线结构：内存和I/O接口，控制简单，易于扩充系统配置I/O设备，单数据传输量受限
  - 2.面向CPU的双总线结构：
    - CPU与主存、CPU与I/O设备分别设置一组总线
    - CPU与主存：存储总线；CPU与I/O设备：I/O总线
    - 优点：提高数据传送效率；缺点：降低了CPU工作效率
  - 3.面向主存的双总线结构：结合以上两种优点


微处
理器
CPU
内存储器
I/O接口
I/O设备
（地址总线）
（数据总线）
（控制总线）

- 1.3 计算机数据格式
  - 数制
    - 无符号
      - 1.数制
        - 二进制B
        - 八进制Q或O
        - 十进制D/省略
        - 十六进制H
        - $N=\pm\sum_{i=-m}^{n-1}K_i\times J^i$
      - 2.其它数制 → 十进制：按权相加
      - 3.十进制 → 其它数制：
        - 整数：基数去除，余数倒排
        - 小数：基数去乘，余数顺排
      - 二进制是本、是桥梁
  - 二进制信息编码
    - 二进制编码的十六进制：BCH
    - 二进制编码的 十 进制：BCD
      - 压　缩：4位1数
      - 非压缩：8位1数,0不可省
    - ASCII 码
      - ‘0 → 9’=30H → 39H
      - ‘a → z’=61H → 7AH
      - ‘A → Z’=41H → 5AH
    - 汉字编码
      - 国标码＝区码＋20H(高字节)＋位码＋20H(低字节)
      - 内码＝国标码＋8080H
  - 带符号数
    - $[x]_{原}$＝机器数，x＝真值
    - 原码
      - 整数：$[x]_{原}=\begin{cases}x & 0\le x<2^{n-1}\\ 2^{n-1}-x & -2^{n-1}<x\le 0\end{cases}$
      - 小数：$[x]_{原}=\begin{cases}x & 0\le x<1\\ 1-x & -1<x\le 0\end{cases}$
    - 反码
      - 整数：$[x]_{反}=\begin{cases}x & 0\le x<2^{n-1}\\ (2^n-1)+x & -2^{n-1}<x\le 0\end{cases}(\mathrm{mod}\ 2^n\text{-}1)$
      - 小数：$[x]_{反}=\begin{cases}x & 0\le x<1\\ (2-2^{-m})+x & -1<x\le 0\end{cases}(\mathrm{mod}\ 2\text{-}2^{-m})$
    - 补码
      - 整数：$[x]_{补}=\begin{cases}x & 0\le x<2^{n-1}\\ 2^n+x & -2^{n-1}<x\le 0\end{cases}(\mathrm{mod}\ 2^n)$
      - 小数：$[x]_{补}=\begin{cases}x & 0\le x<1\\ 2^n+x & -1<x\le 0\end{cases}(\mathrm{mod}\ 2)$
    - 结论
      - 1当x ≥ +0时，$[x]_{原}=[x]_{反}=[x]_{补}$
      - 2当x < 0时，符号位=1，$[x]_{补}=[x]_{反}+1$
      - 3＋0和－0
        - $[+0]_{原}=000...0$　$[-0]_{原}=100...0$
        - $[+0]_{反}=000...0$　$[-0]_{反}=111...1$
        - $[+0]_{补}=000...0=[-0]_{补}$
      - 4移码与补码数值位完全相同，符号位相反
    - 移码：$[x]_{移}=2^{n-1}+x$　$-2^{n-1}\le x\le 2^{n-1}$
  - 小数
    - 定点数
    - 浮点数
      - $N=R^{\pm j}\times(\pm x)$
      - 阶码、尾数都为原码
        - 最大正数$N_{max}=2^{(2^{e-1}-1)}(1-2^{-(m-1)})$
        - 最小负数$N_{min}=-2^{(2^{e-1}-1)}(1-2^{-(m-1)})$
        - 最大正数$N_{\quad}=2^{(2^{e-1}-1)}(1-2^{-(m-1)})$

- 算术运算
  - 定点±法
    - 原码
      - 1符号位不能参加运算
      - 2同号+/异号−
        - 符号：被+/−数相同
        - 数值：两数绝对值之和
        - 溢出：原码[-127,+127]，补码[-128,+127] OF
        - 判断溢出
          - 双符号位：10：上溢；01：下溢
          - 单符号位：$C_S \oplus C_P = 0$ 无溢出；$C_S \oplus C_P = 1$：$C_P = 1$ 上溢，$C_P = 0$ 下溢
      - 3异号+/同号−
        - 符号：绝对值大的那个数相同
        - 数值：两数之差的绝对值
    - 补码 $[x]_{补} \pm [y]_{补} = [x]_{补} \pm [y]_{补}$
  - 原码×

$[x]_{原} \times [y]_{原} = 01010B(+10D) \times 11011B(-10D)$（从低到高乘）

0000		部分积初始值
+1010	$y_1 = 1$	加上被乘数
1010	…	第一次部分积
01010	…	部分积右移
+1010	$y_2 = 1$	加上被乘数
11110	…	第二次部分积
011110	…	部分积右移
+0000	$y_3 = 0$	加0
011110	…	第三次部分积
0011110	…	部分积右移
+1010	$y_4 = 1$	加上被乘数
1101110	…	第三次部分积
01101110	…	部分积右移 → 乘积

  - 原码÷
    - 恢复余数法：被除数－除数，够减商1，不够减商0，且恢复余数，余数左移1位得新余数，在用新余数-除数，依次得到商的每一位
    - 加减交替法：被除数－除数，够减商1，余数左移1位−除数=商…余数；不够减商0，余数左移1位+除数=新余数

第二章　8086 系统结构

16位微处理器结构特点
- 1引脚功能复用：
- 2单总线、累加器结构：
- 3可控三态电路：
- 4总线分时复用：

8086：16 位、N－沟道、HMOS 工艺、双列直插式、40 个引脚、时钟频率 5/8/10MHZ、

16 根数据线 20 根地址总线、直接寻址空间为 2^{20}（1MB）

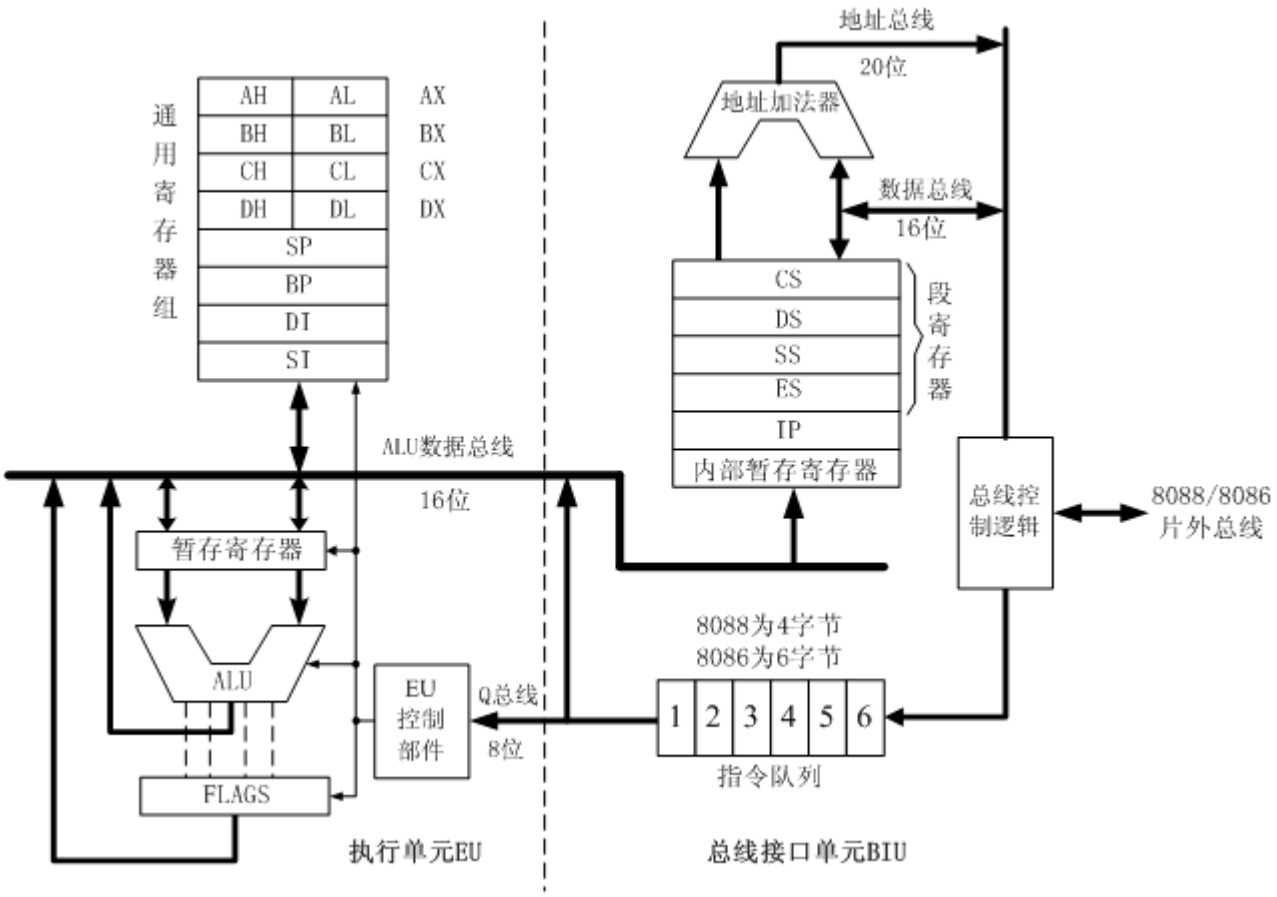
2.1 8086CPU 结构

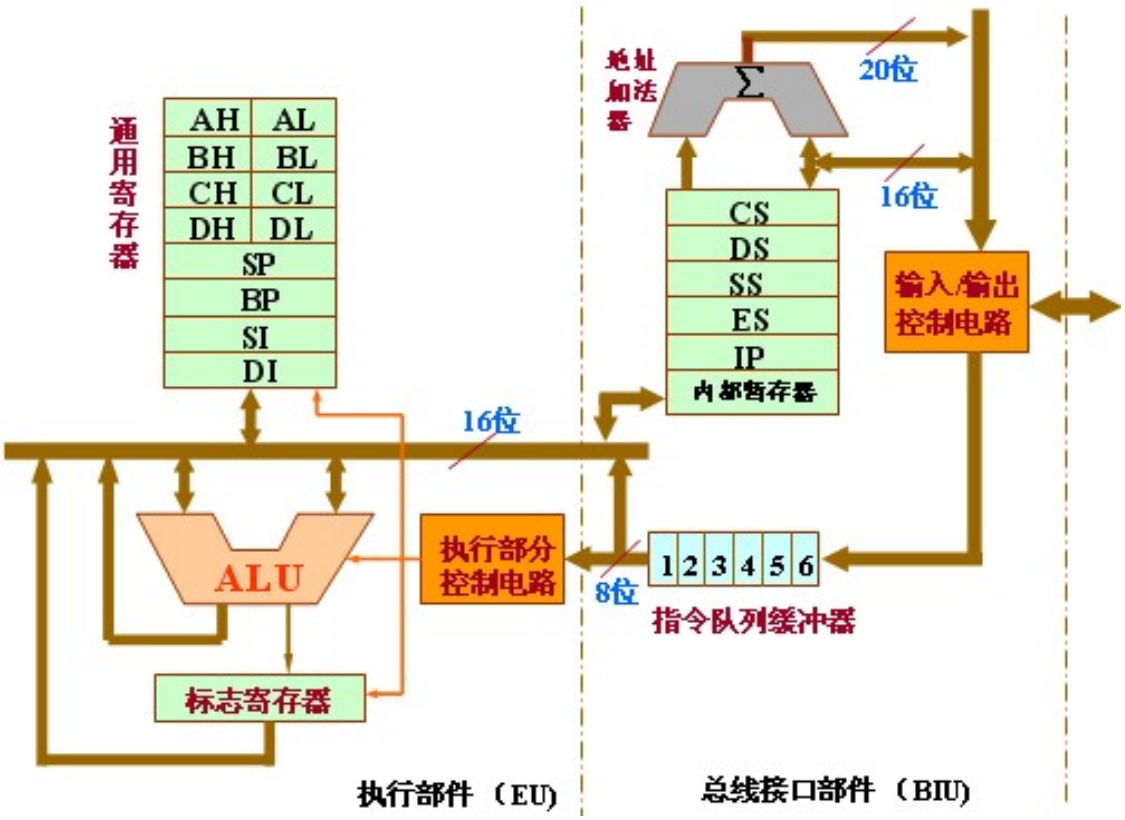
微机工作流程：取指令 $\xrightarrow{\text{存储器}}$ 取操作数 → 执行指令 → 送结果 。8 位机：串联执行。16 位机：并行操作。

构成：总线接口部件 BIU 和指令执行部件 EU

BIU：取指令、读操作数、送结果。EU：执行指令。

2.1.1 8086 CPU 的内部结构





- 8086CPU
  - BIU：
    - 描述：是8086CPU与外部(存储器和I/O端口)的接口，提供16位双向数据总线和20位地址总线，完成所以外部总线操作。
    - 功能：1地址形成2取指令3指令排队4读/写操作数5总线控制
    - 组成：
      1. 16位段地址寄存器{CS-代码段寄存器/DS-数据段寄存器/ES-附加段寄存器/SS-堆栈段寄存器
      2. 16位指令指针寄存器*IP*：
      3. 20位物理地址加法器：
      4. 6字节指令队列：
      5. 总线控制逻辑：
    - 工作过程
      - *CS* —最低位后面补4个0→ *IP* —地址加法器→ 20位物理地址 ⟶
      - 地址总线 —读信号RD→ 启动存储器 —取指令→ 送指令队列等待执行
  - EU：
    - 功能：指令译码和执行指令
    - 组成：
      1. 算术逻辑运算单元ALU：
      2. 标志寄存器flags：
      3. 寄存器组
         - 通用{AX/BX/CX/DX
         - 专有
           - 源变址寄存器SI
           - 目的变址寄存器DI
           - 堆栈指针寄存器SP
           - 基址指针寄存器BP
      4. *EU*控制器
    - *EU*工作过程：
  - 指令重叠执行技术{BIU、EU相互独立又相互配合，并行但不同步，
    加快了指令执行速度，提高总线传输速率

BIU：
- 描述：是8086CPU与外部(存储器和I/O端口)的接口，提供16位双向数据总线和20位地址总线，完成所以外部总线操作。
- 功能：1地址形成2取指令3指令排队4读/写操作数5总线控制
- 组成：
  1. $\frac{16位}{64K}$段地址寄存器
     - CS-代码段：指令(程序)代码
     - DS-数据段
     - ES-附加段
     - 操作数
       - 通常在DS中
       - 串指令中目的操作数必须在ES中
     - 00000H～FFFFFH，(2^5)16的倍数
     - SS-堆栈段：临时存放数据或地址；先进后出
  2. 16位指令指针寄存器IP
     - 功能：存放将要执行的下一条指令在现行CS中的偏移地址,由BIU自动修改
     - 8086程序不能直接访问IP，但可修改IP内容，如中断、程序转移等
  3. 20位物理地址加法器∑:16位逻辑地址 → 20位物理地址
  4. 6字节指令队列：存放6字节指令代码
  5. 总线控制逻辑：发出总线控制信号
- 工作过程：CS：IP $\xrightarrow{左移4位+16位偏移地址=\sum}$ 20位物理地址 → 地址总线 → 总线控制逻辑 $\xrightarrow{读信号\overline{RD}}$ 启动存储器 $\xrightarrow{取指令}$ 送

- *EU*：
  - 功能：指令译码和执行指令
  - 组成：
    - 1.算术逻辑运算单元ALU：
      - 完成8/16位二进制数运算
      - 16位暂存器可暂存参加运算的操作数
    - 2. 标志寄存器Flags：
      - 功能：存放ALU运算结果特征，后续指令控制条件
      - 状态标志
        - 运算结构状态特征，影响后面操作，常作为标志转移指令测试条件
        - 1CF
          - 进位标志位，最高位有进位或借位CF=1
          - STC置位；CLC复位；CMC取反
        - 2PF
          - 奇偶校验标志位，低8位
            - 偶数个1：PF=1
            - 奇数个1：PF=0
        - 3AF
          - 辅助进位标志位
            - 低4位向高4位进/借位
              - 有AF=1
              - 无AF=0
        - 4ZF
          - 全0标志位
        - 5SF
          - 符号标志位
            - 最高位=1，SF=1
            - 最高位=0，SF=0
        - 6OF
          - 溢出标志位
            - 有溢出OF=1，加减法复杂辨析
      - 控制标志
        - 控制CPU操作
        - 1TF
          - 单步标志位
        - 2IF
          - 中断标志位
            - IF=1，CPU响应可屏蔽中断
            - STI置位；CLI复位
        - 3DF
          - 方向标志位
            - 功能：用于串操作指令
              - DF=0：地址指针自动增量，CLD复位
              - DF=1：地址指针自动减量，STD置位
    - 3.寄存器组
      - 通用
        - AX：累加器
        - BX：
        - CX：
        - DX：
      - 专有
        - 源变址寄存器SI
        - 目的变址寄存器DI
        - 与DS联用寻址
          - 在串指令中
            - 源操作数偏移地址→SI；DS：SI
            - 目的源操作数偏移地址→DI；ES：DI
        - 堆栈指针寄存器SP：SP始终指向栈顶
        - 基址指针寄存器BP：间接寻址，与SS联用
    - 4.*EU*控制器：取指令控制和时序控制部件
  - *EU*工作过程：BIU指令队列→取指令译码→操作数偏移地址→BIU20物理地址→执行

存储器分段管理

- 分段类型
  1. 段与段紧密相邻
  2. 段与段部分重叠
  3. 段与段断开
  4. 段与段完全重叠：小程序
- 管理方法
  - 段起始地址是16的倍数，故将段地址左移4位即得20位物理段边界
  - 数操作
    - 寻址代码段：CS：IP
    - 寻址数据段：DS：BX/DI/SI
    - 寻址堆栈段：SS：SP/BP
  - 串操作
    - 源　数据区：DS：SI
    - 目的数据区：ES：DI
- 1MB物理空间
  - 最低端1KB(00000H ~ 003FFH)：中断向量(1*K* = 256×4(4 = 2偏移量 + 2段地址))
  - 最高端16B(FFFF0H ~ FFFFFH)：一条无条件转移指令，转至系统初始化程序(8086加电或复位CS:
- 逻辑地址：物理地址的虚拟化，程序设计用；16位
  物理地址=段地址（左移4位/×16/10H）：偏移地址；20位
  无符号整数，1：n
- 若上述地址相加遇到第20位有进位的情况，则进位可忽略

I/O接口/设备地址管理

- 方法
  1. 存储器统一编址：物理地址=段地址：偏移地址
  2. I/O独立　编址：物理地址=逻辑地址AD_{15} ~ AD_0
- M/$\overline{IO}$控制

逻辑地址来源			
操作类型	隐含段地址	替换段地址	偏移地址
取指令	CS	无	IP
堆栈操作	SS	无	SP
BP 为间址	SS	CS,DS,ES	有效地址 EA
存取变量	DS	CS,DS,ES	有效地址 EA
源字符串	DS	CS,DS,ES	SI
目的字符串	ES	无	DI

8086存储器分体结构

- 偶地址：低8位
- 奇地址：高8位
- 字节操作，只取一次
- 字操作
  1. 从偶地址开始只取一次：对准存放(对准字)
  2. 从奇地址开始先奇后偶两次：非对准字

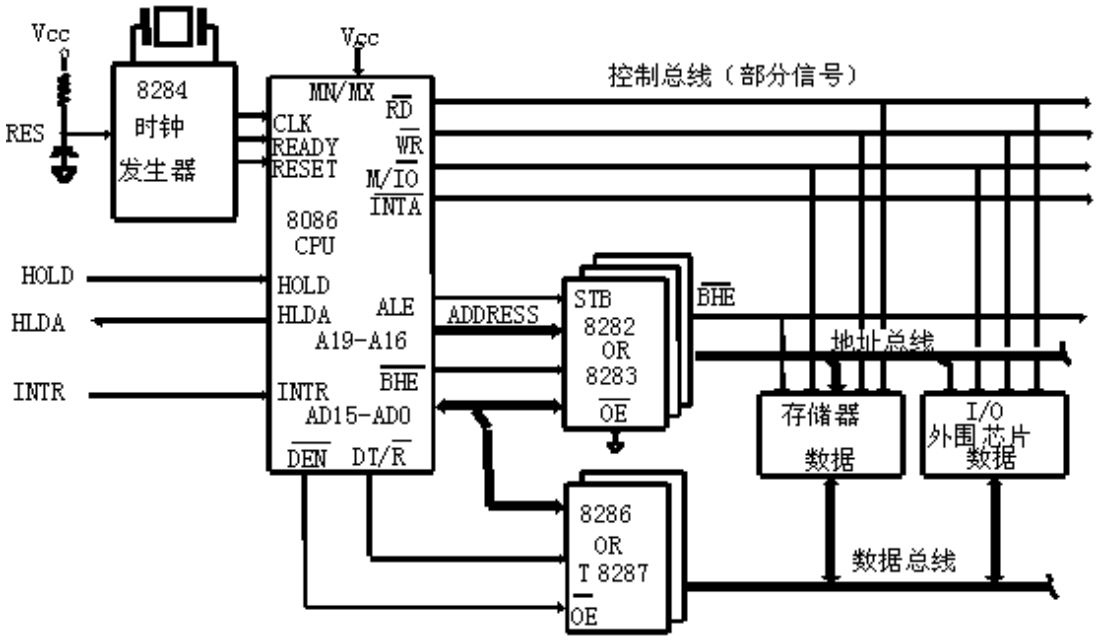
$\overline{BHE}$ 和 A_0 编码含义			
$\overline{BHE}$	A_0	操作	总线使用情况
0	0	从偶地址开始读/写一个字	AD_{15} ~ AD_0
0	1	从奇地址开始读/写一个字节	AD_{15} ~ AD_8
1	0	从偶地址开始读/写一个字节	AD_7 ~ AD_0
1	1		

0 1 1 0	从奇地址读/写一个字	AD_{15} ~ AD_8 AD_7 ~ AD_0

堆栈

堆栈
- 概念：在存储器中开辟一个区域，用来存放需要暂时保存的数据。
- 容量:≤64K，可在1MB空间任意浮动
- 段基址：SS；指针：SP
- 工作方式：先进后出，以字为单元操作
- 操作:$\frac{\text{PUSH}}{\text{SP-2}\rightarrow\text{SP}}/\frac{\text{POP}}{\text{SP+2}\rightarrow\text{SP}}$
- 应用：中断及子程序调用和数据暂时保存
- 地址增长方式：向上增长/由高到低：底高顶低

8086*CPU*引脚
- 地址线：20；数据线：16；40个引脚双列直插式封装，分时复用
- $MN/\overline{MX}(33)$
  - 最小模式：单机系统，所需控制信号全由8086(CPU)提供（兼容8080/8085）
  - 最大模式：多处理机系统，所需控制信号由总线控制器8288提供。IBM-PC
- 这些引脚构成CPU外部芯片总线
  1. 数据总线DB
  2. 地址总线AB $\rightarrow$ *CPU*通过这些总线和*M*、*I*/*O*交换数据
  3. 控制总线CB



最小模式的微计算机组成

最小模式引脚

1. $\frac{AD_{15} \sim AD_0\,(\text{Address Date Bus})}{\text{16位地址/数据总线}}$
   1. 双向、三态
   2. 地址：输出，T_1
      数据：双向输入/输出，$T_2 \sim T_4$
   3. 系统总线“保持响应”周期，浮空（高阻态）

8. $\frac{ALE\,(\text{Address}L\text{atch}E\text{nable})}{\text{地址锁存允许信号}}$ {三态，输出，不能浮空,8282/8283

9. $\frac{\overline{DEN}\,(D\text{ate}E\text{nable})}{\text{数据允许信号}}$ {三态，输出，在*DMA*工作方式时浮空,8286/8287

2. $\frac{A_{19}/S_6 \sim A_{16}S_3\,(\text{Address/Bus})}{\text{地址/状态线}}$
   1. 三态、输入、分时复用
   2. T_1：地址线（$A_{19} \sim A_{16}$）：与$A_{15} \sim A_0$构成20位物理地址（
      $T_2 \sim T_4$：状态线（$S_6 \sim S_3$）
      - S_6：保持“0”:8086连在总线上
      - S_5：中断允许标识状态
        - S_5=1，允许可屏蔽
        - S_5=0，禁止
      - S_4、S_3：指示当前正在使用的段寄存器
        - 00：ES
        - 01：SS
        - 10：CS，或不需要使用段寄存器（I
        - 11：DS

3。 $\frac{\overline{BHE}/S_7\,(BusHighEnable/Status)}{\text{高8位数据总线允许/状态信号}}$
   1. 三态、输出
   2. 在16位数据传送时，T_1：用$\overline{BHE}$指出高8位（$D_{15} \sim D_8$）数据总线上数据有效,AD_0指出低8位数据有效
   3. $T_2 \sim T_4$：S_7输出状态信息（=1）

4. $\frac{MN/\overline{MX}\,(Minimum/Maximun)}{\text{最小/最大工作模式}}$

5. $\overline{RD}\,(D\text{ead})$：三态，输出，$T_2.T_3.T_W$（数据：存储器 $\rightarrow$ *CPU*）

6. $\overline{WR}\,(W\text{rite})$：三态，输出，$T_2.T_3.T_W$（数据：*CPU* $\rightarrow$ 存储器）

7. $\frac{M/\overline{IO}\,(M\text{emory/Input and Output})}{\text{存储器或I/O端口控制信号}}$：三态，输出，$T_4 \sim T_4$

10. $\frac{DT/\overline{R}\,(DateTransmit/\text{Re}ceive)}{\text{数据发送/接收控制信号}}$：三态，输出
    - $DT/\overline{R}=1$，写：*CPU* $\rightarrow$ 存储器
    - $DT/\overline{R}=0$，读：存储器 $\rightarrow$ *CPU*

11. $\frac{READY\,(\text{ready})}{\text{准备就绪信号}}$

12.*RESET*（*reset*）：CS:FFFFH,其他：00H。FFFF0H

13. INTR(Interrupt Request) 可屏蔽中断请求信号
- $STI: IF = 1$
- $CLI: IF = 0$

14. $\overline{\text{INNA}}$(*I*nterrupt*A*cknowledge) 中断响应信号
- 第一次负脉冲：通知外设接口已响应它的中断请求
- 第二次负脉冲：通知外设将中断类型号输入考类型总线

15. NMI(Non-Maskable Interrupt Request) 不可屏蔽中断请求信号

16. $\overline{\text{TEST}}$(test)：测试信号

17. HOLD(Hold Request)：总线保持请求信号

18. *HLDA*(*H*old*A*cknowledge)：总线保持响应信号

19. CLK(Clock)：时钟信号

20. $V_{CC}(+5V), GND$

最大模式

1. $\overline{S_2} \sim \overline{S_0}$(*B*us*C*ycle*S*tatus) 总线周期状态信号
- 000：发中断响应信号。001：读I/0端口。010：写I/0端口。011：暂停
- 100：取指令。101：读存储器。110：写存储器。111：无源状态。

2. $\overline{\text{LOCK}}$(*Lock*) 总线封锁信号

3. $\overline{\text{RQ}}/\overline{GT_0}$、$\overline{\text{RQ}}/\overline{GT_1}$(*Q*equest / *G*rant) 总线请求信号输入/总线请求允许信号输出
- 00：无操作。01：从指令队列中取走第一个字节
- 10：队列已空。11：从指令队列中取走后续字节

4. QS_1、QS_0(*I*nstruction*Q*ueueStatus) 指令队列状态信号

8088与8086不同
1. 8088的指令队列长度为4个字节，指令队列中只要出现一个空闲字节时，BIU就会自动访问存储器，取指令来补充指令队列。
2. 8088CPU中BIU的总线控制电路与外部交换数据的总线宽度是8位，总线控制电路与专用寄存器之间的数据总线宽度也是8位 而EU的内部总线是16位，这样，对16位数的存储器读/写操作需要两个读/写周期才能完成。
3. 8088外部数据总线只有8条，所以分时复用的地址/数据总线为$\text{AD}_7 \sim AD_0$；而$\text{AD}_{15} \sim AD_8$成为仅传递地址信息的$\text{A}_{15} \sim A_8$.
4. 8088用IO/$\overline{\text{M}}$代替$\overline{\text{M}}$/IO，为了与8085兼容
5. 8088中只能进行8位数据传输，$\overline{\text{BHE}}$信号不需要了，改为$\overline{SS_0}$，与$DT/\overline{R}$和$IO/\overline{M}$一起决定最小模式中的总线周期操作

IO/$\overline{\text{M}}$	$DT/\overline{R}$	$\overline{SS_0}$	含义
0	0	0	取指令
0	0	1	读存储器
0	1	0	写存储器
0	1	1	无源状态
1	0	0	发中断响应信号
1	0	1	读 I/0 端口
1	1	0	写 I/0 端口
1	1	1	暂停

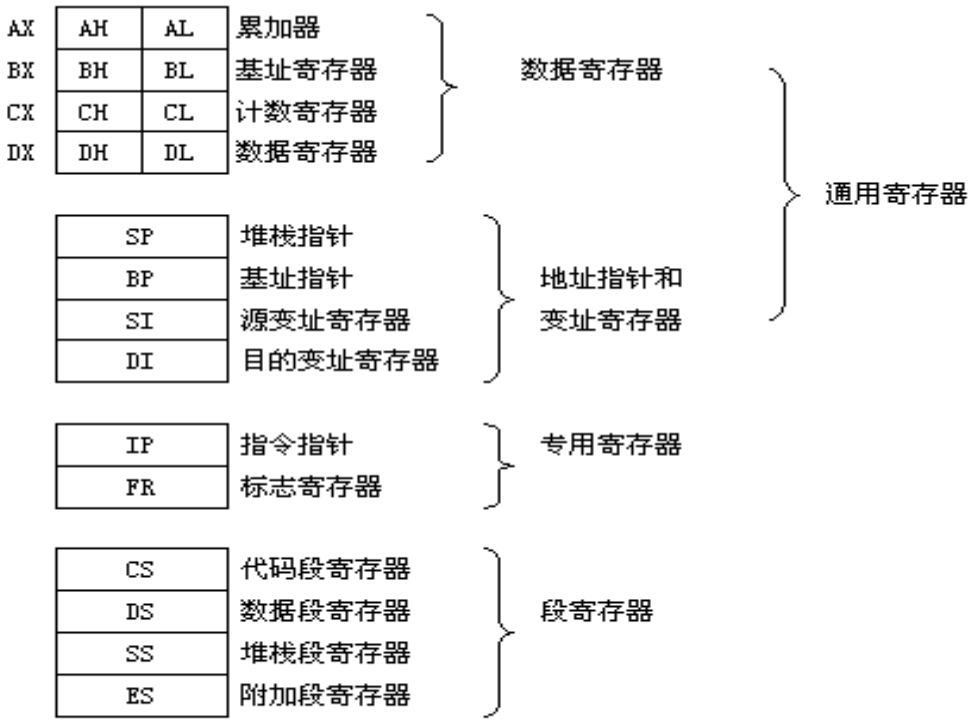


图 2．2　8086/8088 CPU 寄存器结构

第三章 8086　　寻址方式和指令系统

3.18086 的寻址方式

- 指令
  - 1.操作码：操作的性质
  - 2.操作数：操作的对象
    - 1.单操作数
    - 2.双操作数（必有一个寄存器）：操作码 目的操作数，源操作数　　MOV　AX ， CX
    - 3.无操作数
  - 包含在
    - 1.寄存器：指令执行速度最快，在CPU内部立即执行
    - 2.存储器：
      - 速度较慢(最复杂)：通过总线与CPU之间交换数
      - 必须先把一个偏移量送到BIU，计算出20位物理
    - 3.*I* / *O*端口地址
    - 4.立即数：寻址指令可直接从指令队列中取数，速度也较快
- 寻址方式：指令中说明操作数所在地址的方法。

寻址方式

1.立即寻址方式
- 1.操作数：直接包含在指令中，8或16位常数(立即数)。常用来给寄存器赋初值。
- 2.机器码：操作码 代码寄存器，立即数
  MOV AL, 26H(CX, 2A50H)
  ↖源操作数(A～F打头前加0)

2.寄存器寻址方式
- 1.操作数：包含在寄存器中。
- 2.机器码：操作码 寄存器(AX/BX/CX/DX/SI/DI/SP/BP或AH/AL/BH/BL/CH/CL/DH/DL)
  源操作数=目的操作数

3.直接寻址方式
- 1.有效地址EA：操作数的偏移地址；存储在代码段中指令的操作码之后。
  而该地址单元的数据总是存放在存储器中，必须先求出物理地址
- 2.物理地址：{操作数物理地址=16×DS+[EA]=10H×DS+[EA]
- 3.段超越前缀：MOV AX, ES:[EA]{操作数物理地址=16×ES+[EA]=10H×ES+[EA]
  MOV AX, DS:[EA]{操作数物理地址=16×DS+[EA]=10H×DS+[EA]
- 4.符号地址：或 MOV AX, AREA1 / MOV AX,[AREA1]
  AREA1 EQU 0867H
  MOV AX, AREA1
  允许段超越

4.寄存器间接寻址方式
- 1.寄存器(BX/BP//SI/DI)中的值不是操作数本身，而是操作数有效地址
- 2.物理地址=16×DS+BX
  或 =16×DS+SI
  或 =16×DS+DI
- 3.物理地址=16×SS+BP
- 4.MOV BX,[SI]
- 5.允许段超越前缀 MOV BX,DS:[BP] / MOV AX,ES:[SI]

5.寄存器相对寻址方式
- 1. EA：是一个基址或变址寄存器的内容与指令中指定的8或16位位移量之和。
- 2. MOV BX,COUNT[SI] / MOV BX,[COUNT+SI] 物理地址=16×DS+BX+COUNT
- 3.允许段超越前缀MOV DH,ES:ARRAY[SI]

6.基址变址寻址方式
- 1.操作数EA：一个基址寄存器(BX或BP)+一个变址寄存器(SI或DI)，寄存器均由指令指定
- 2.物理地址=16×DS+BX+SI
  或 =16×DS+BX+DI
  物理地址=16×SS+BP+SI
  或 =16×SS+BP+DI
- 3. MOV AX,[BX][SI] / MOV AX,[BX+SI] 物理地址=16×DS+BX+SI

7.相对基址变址寻址方式
- 1. 操作数EA：一个基址寄存器+一个变址寄存器+8或16位位移量
- 2.物理地址=16×DS+BX+SI+8或16位位移量
  或 =16×DS+BX+DI+8或16位位移量
  物理地址=16×SS+BP+SI+8或16位位移量
  或 =16×SS+BP+DI+8或16位位移量
- 3. MOV AX,MASK[BX][SI] / MOV AX,MASK[BX+SI] / MOV AX,[MASK+BX+SI] / MOV AX,200H[BX+SI]
  物理地址=16×DS+BX+SI+MASK

- [ ]规则
  - 1.立即数可以出现在[ ]内，表示直接地址,[2000H]
  - 2.只有BX/BP/SI/DI可以出现在[ ]内，单个或组合加，但BX/BP、SI/DI不允许同时出现
  - 3.6[BX][SI]=[BX+6][SI]=[BX+SI+6]
  - 4.[BP]：物理地址=16×SS+EA
    - DISP[BP+SI] ；EA=BP+SI+DISP
    - DISP[BP+DI] ；EA=BP+DI+DISP
    - DISP[BP] ；EA=BP+DISP

    $DISP = 0,8,16$
  - 5.[BP]：物理地址=16×DS+EA
    - [DISP] ；EA=DISP
    - DISP[BX+SI] ；EA=BX+SI+DISP
    - DISP[BX+DI] ；EA=BX+DI+DISP
    - DISP[BX] ；EA=BX+DISP
    - DISP[SI] ；EA=SI+DISP
    - DISP[DI] ；EA=DI+DISP

    $DISP = 0,8,16$
- 其它寻址方式
  - 1.隐含寻址
  - 2.I/O端口寻址
    - 1.直接：8位立即数，00～FFH,256个端口 IN AL, 63H
    - 2.间接：16位立即数，0000～FFFFH,2^{16}个端口 IN AL, 63H
  - 3.一条指令有几天寻址方式
  - 4.转移类指令寻址

8086 指令系统

- 8086指令系统
  - 1.数据传送指令（14）
    - 1.通用数据传送指令:MOV;PUSH;POP;XCHG;XLAT
    - 2.输入输出指令:IN;OUT
    - 3.地址目标传送指令:LEA;LDS;LES
    - 4.标志传送指令:LAHF;SAHF;PUSHF;POPF
  - 2.算术运算指令（20）
    - 1.加法指令:ADD;ADC;INC;AAA;DAA
    - 2.减法指令:SUB;SBB;DEC;NEG;CMP;AAS;DAS
    - 3.乘法指令:MUL;IMUL;AAM
    - 4.除法指令:DIV;IDIV;AAD;CBW;CWD
  - 3.逻辑运算和移位指令(12)
    - 1.逻辑运算：NOT;AND;OR;XOR;TEST
    - 2.算术逻辑移位：SHL/SAL;SHR;SAR
    - 3.循环移位：ROL;ROR;RCL;RCR
  - 4.字符串处理指令(5)
    - 1.MOVS
    - 2.CMPS
    - 3.SCAS
    - 4.LODS
    - 5.STOS
  - 5.控制转移指令(28)
    - 1.无条件移位和过程调用指令：JMP;CALL;RET
    - 2.条件转移:JZ/JE等10条指令；JA/JNBE等8条指令
    - 3.条件循环控制:LOOP;LOOPE/LOOPZ;LOOPNE/LOOPNZ;JCXZ
    - 4.中断：INT;INTO;IRET
  - 6.处理器控制指令()
    - 1.标志控制指令：CLC;CMC;STC;CLD;STD;CLI;STI
    - 2.外部同步指令:ESC;WAIT;LOCK
    - 3.停机指令和空操作指令:HLT;NOP

通用数据传送指令

1.MOV
- 格式：MOV 目的，源
- 注意：
  1. 目的 ≠ *CS* / 立即数/IP { 立即数 ⊂ *CS*
  2. 源 ≠ IP
  3. 两个操作数必须有一个为寄存器，但不能同时为段寄存器

2.PUSH
- 格式：PUSH 源
- 注意：
  1. 源 ≠ 立即数(字)
  2. 先进后出；SS:$SP \leftarrow SP-2; C_{max} = 64K, FFFEH$(偶) ~ $0000H$

3.POP
- 格式：POP 目的
- 注意：
  1. 目的 ≠ *CS* / 立即数(字)
  2. $SP \leftarrow SP+2;$

4.XCHG
- 格式：XCHG 目的，源(=PUSH+POP)
- 注意：
  1. 操作数 ≠ 段寄存器，立即数
  2. 寄存器 ↔ 寄存器；寄存器 ↔ 存储器

5.XLAT
- 格式：XLAT 转换表 或 XLAT 功能：将一个字节(256)从一种代码转换为另一种代码
- 步骤：

```
TABLE  DB 4OH,79H,24H,30H,19H
       DB 12H,02H,78H,00H,18H  ;七段数码表
       ⋮
       MOV  AL，5              ;AL ← 数字5的位移量
       MOV  BX,OFFSET TABLE    ;BX ← 表格首地址
       XLAT  TABLE             ；查表得AL=12H
```

数据传送指令

输入输出指令

功能：I/O端口 ↔ 累加器AX

1. IN
- 格式：
  1. IN AL/AX,端口地址 ； 00 ~ FFH
  2. MOV DX, 端口地址
     IN $\frac{AL}{AX}$,DX ;0000 ~ *FFFFH*

2.OUT
- 格式：
  1. OUT 端口地址,AL/AX ； 00 ~ FFH
  2. MOV DX,端口地址
     OUT DX,$\frac{AL}{AX}$;0000 ~ *FFFFH*

- 地址目标传送指令
  - 1.LEA
    - 格式：LEA 目的，源　功能：目的←源操作数偏移地址
    - 说明
      - 源：存储单元(偏移地址)
      - 目的：非段寄存器的16位寄存器
    - 与MOV区别
      - LEA BX,[SI] ；取地址
      - MOV BX,[SI] ；取内容
    - 与MOV联系
      - LEA BX,TABLE
      - MOV BX,OFFSET TABLE
    - MOV所不及
      - LEA BX,6[DI];数组单元
  - 2.LDS
    - 格式：LDS 目的，源　功能：目的←源操作数偏移地址
    - 说明
      - 源：4字节地址指针
      - 目的：寄存器(SI)(前2个)，隐含DS(后2个)
    - 实例
      - 设：DS=1200H,(12450H)=F346H,(12452H)=0A90H
      - LDS SI,[450H] ;SI=F346H,DS=0A90H
  - 3.LES
    - 格式：LES 目的，源　功能：目的←源操作数偏移地址
    - 说明
      - 源：4字节地址指针
      - 目的：寄存器(DI)(前2个)，隐含ES(后2个)
    - 实例
      - 设：DS=0100H,BX=0020H,(01020H)=0300H,(01022H)=0500H
      - LES DI,[BX] ;DI=0300H,ES=0500H

- 标志传送指令
  - 1.LAHF
    - 格式：LAHF
    - 功能：FLAGS→AH、兼容8080/8085
  - 2.SAHF
    - 格式：SAHF
    - 功能：AH→FLAGS
  - 3.PUSHF
    - 格式：PUSHF
    - FLAGS入栈
  - 4.POPF
    - 格式：POPF
    - FLAGS出栈

  3.PUSHF、4.POPF：常在子程序调用和中断服务程序的开头和结尾；对过程调用和发生中断的主程序状态(标志位)保护

算术运算指令

加法指令

1.ADD
- 格式：ADD 目的，源　　功能：目的←目的+源

2.ADC
- 格式：ADC 目的，源　　功能：目的←目的+源+CF

3.INC
- 格式：INC 目的　　功能：目的←目的+1
- 注意：INC BYTE PTR[BX]

4.AAA
- 格式：AAA　　功能：加法的ASCII码调整
- 注意：用于ADD/ADC对2个非压缩码或ASCII码运算之后
- 实例：

```
ADD  AL,BL  ;  0000 1001...9  ;BCD 9  '9'
              +0000 0101...5  ;BCD 5  '5'

AAA         ;  0000 1110...低4位>9
              +0000 0110...加6调整(AL=AL+6)
               0001 0100
              ∧0000 1111...清高4位
               0000 0100...AL=4
               CF=1,AF=1,AH=1(AH=AH+1)
结果为AX=0104H，表示非压缩十进制数14
```

5.DAA
- 格式：DAA　　功能：加法的十进制调整
- 注意：用于ADD/ADC对2个压缩BCD码之后，调整为正确的BCD码
- 实例：

```
ADD  AL,BL  ;  0011 1000...38  ;BCD 38
              +0001 0101...15  ;BCD 15

AAA         ;  0100 1101...低4位>9
              +0000 0110...加6调整(AL=AL+6)
               0101 0100...结果为AL=BCD 53,CF=0
```

- 实例：

```
ADD  AL,BL  ;  1000 1000...88  ;BCD 88
              +0100 1001...49  ;BCD 49

AAA         ;  1101 0001...AF=1
              +0000 0110...加6调整(AL=AL+6)
               1101 0111...高4位>9
              +0110 0000...加60调整(AL=AL+60)
               0011 0111...结果为AL=BCD 37,CF=1
```

减法指令

1.SUB
- 格式：SUB 目的，源　　功能：目的←目的−源

2.SBB
- 格式：SBB 目的，源　　功能：目的←目的−源−CF
- 适用场合：多字节减法

3.DEC
- 格式：DEC 目的　　功能：目的←目的−1
- 注意：DEC BYTE PTR[BX]

4.NEG
- 格式：NEG 目的　　功能：目的←0−目的

5.CMP
- 格式：CMP 目的，源　　功能：目的−源
- 注意：后常接JB条件跳转
- 适用场合：比较2个数大小且不破坏原操作数

6.AAS
- 格式：AAS　　功能：减法的ASCII码调整
- 注意：用于SUB/SBB对2个非压缩码十进制或ASCII码运算之后
- 实例：
  - SUB AL,CL ;
    - 0000 0011...3 ;BCD 3 '3'
    - −0000 1000...8 ;BCD 8 '8'
  - AAS ;
    - 1111 1011...低4位>9
    - −0000 0110...减6调整(AL=AL+6)
    - 0001 0101
    - ∧0000 1111...清高4位
    - 0000 0101...AL=5
    - 结果为AL=5,CF=1(AH=AH−1)

7.DAS
- 格式：DAS　　功能：减法的十进制调整
- 注意：用于SUB/SBB对2个压缩十进制之后
- 实例：
  - SUB AL,CL ;
    - 0101 0110...56 ;BCD 56
    - −1001 1000...15 ;BCD 98
  - DAS ;
    - 1011 1110...低4位>9
    - −0000 0110...减6调整(AL=AL−6)
    - 1011 1000...高4位>9
    - −0110 0000...减60H调整
    - 0101 1000...BCD 58
    - 结果为AL=BCD 58,CF=1,表示有借位

乘法指令

1. MUL 无符号
   - 格式：MUL 源　　功能：1.AX ← AL*源　源=8；2.(DX,AX) ← AX*源　源=16
   - 说明：源=寄存器/=定类型的存储单元/≠立即数
2. IMUL 带符号
   - 格式：IMUL 源　　功能：1.AX ← AL*源　源=8；2.(DX,AX) ← AX*源　源=16
   - 说明：源=寄存器/=定类型的存储单元/≠立即数
3. AAM ASCII
   - 格式：AAM　　调整：1.AH ← AL/10的商；2.AL ← AL/10的余数
   - 实例：

```
MOV  AL,09H   ;置初值BCD码
MOV  BL,06H
MUL  BL       ;AL ← 36H
AAM           ;调整得AH=05H(十位),AL=04H(个位)
              ；结果为AX=BCD 0504H
```

   - 实例：

```
MOV  AL,‘9’   ;置初值ASCII码
MOV  BL,‘6’
AND  AL,0FH   ;屏蔽高字节
AND  BL,0FH
MUL  BL       ;
AAM           ;调整得AH=05H(十位),AL=04H(个位)
              ；结果为AX=BCD 0504H
OR   AX,3030H ；结果调节成ASCII码，AX=3435H
```

除法指令

1. DIV 无符号
   - 格式：DIV 源（除数）
   - 功能：
     - 源=8
       - 被除数=16
         1. AL ← AX/源（字节）的商
         2. AH ← AX/源（字节）的余数
       - 被除数=8 { AL ← 被除数，AH=0
     - 源=16
       - 被除数=32
         1. AX ← (DX,AX)/源（字节）的商
         2. DX ← (DX,AX)/源（字节）的余数
       - 被除数=16 { AX ← 被除数，DX=0
   - 说明：需要MOV先赋值

2. IDIV 带符号
   - 格式：IDIV 源
   - 功能：
     - 源=8
       - 被除数=16
         1. AL ← AX/源（字节）的商
         2. AH ← AX/源（字节）的余数
       - 被除数=8 { AL ← 被除数，AH=0
     - 源=16
       - 被除数=32
         1. AX ← (DX,AX)/源（字节）的商
         2. DX ← (DX,AX)/源（字节）的余数
       - 被除数=16 { AX ← 被除数，DX=0
   - 说明：余数与被除数符号相同

3. AAD ASCII
   - 格式：AAD　　调整
     1. AL ← AH*10+AL
     2. AH ← 00
   - 实例：
     设AX=0307H(2个非压缩BCD码，即十进制数37)，BL=05H
     AAD ;
       在DIV之前将AX非压缩BCD码 → 二进制码
       03×10+7=37=25H, 并将AL ← 25H
     DIV BL ；结果为AL=7(商),AH=2(余数)
   - 实例：

```
编写程序，计算75÷6=12...3
FIRST   DB 06H              ;除数6
SECOND  DB 75H              ;被除数75(BCD)
THIRD   DB 2 DUP(0)         ;存商
FOUR    DB ?                ;存余数
        ⋮
        MOV AH,00H          ;第一个被除数高位清0
        MOV AL,SECOND       ;AL ← 被除数75
        AND  AL,0F0H        ;截取高4位
        MOV CL,04H          ;
        ROL  AL,CL          ;移至低4位
        DIV  FIRST          ;AX/06,即0007/06, 结果：AL ← 商1, AH ← 余数1
        MOV THIRD+1,AL      ;结果单元 ← 第一个商1
        MOV AL,SECOND       ;AL ← 被除数75
        AND  AL,0FH         ;截取低4位，AX=0105H
        AAD                 ;将AX中的内容0105H调整为0FH
        DIV  FIRST          ;0FH/6,结果：AL ← 商2，AH ← 余数3
        MOV THIRD,AL        ;THIRD单元 ← 第二个商2
        MOV FOUR ,AH        ;FOUR单 ← 元第二个余数3
```

4. CBW
   - 格式：CBW　　功能：字节 → 字
   - AH被扩充为AL中D_7的符号位

5. CWD
   - 格式：CWD　　功能：字 → 双字
   - DX被扩充为AX中D_{15}的符号位

```
MOV AL,11011010B   ;被除数-38
MOV CH,00000011B   ;除数+3
```

逻辑运算和位移指令-按位运算

- 逻辑运算指令
  - 1.NOT：格式：NOT 目的　　功能：目的←$\overline{\text{目的}}$
  - 2.AND：格式：AND 目的, 源　　功能：目的←目的∧源；适用性：屏蔽位
  - 3.OR：格式：OR 目的, 源　　功能：目的←目的∨源；适用性：保留位，置位1　BCD→ASCII：OR AX,3030H
  - 4.XOR：格式：XOR 目的, 源　　功能：目的←目的∀源；适用性：保留(0)与取反(1)
  - 5.TEST：格式：TEST 目的, 源　　功能：目的∧源；适用性：检测某些条件是否满足，但原操作数不变，后常跟条件转移指令JNZ
- 算术逻辑转移指令
  - 1.SAL 算术左移：格式：SAL 目的, 计数值
  - 2.SHL 逻辑左移：格式：SHL 目的, 计数值
    - 功能:2者完全相同，相当于×2
    - 说明:计数值=1，若计数值>1，则MOV CL，移位次数
  - 3.SHR 逻辑右移：格式：SHR 目的, 计数值
    - 功能：相当于÷2，但没有余数，最低位进入CF,最高位/符号位补0
    - 说明:计数值=1，若计数值>1，则MOV CL，移位次数
  - 4.SAR 算术右移：格式：SAR 目的, 计数值
    - 功能：相当于÷2，但没有余数，最低位进入CF,最高位/符号位不变
    - 说明:计数值=1，若计数值>1，则MOV CL，移位次数
- 循环位移指令
  - 1.ROL 循环左移：格式：ROL 目的, 计数值
  - 2.ROR 循环右移：格式：ROR 目的, 计数值
  - 3.RCL 通过进位CF位循环左移：格式：RCL 目的, 计数值
  - 4.RCR 通过进位CF位循环右移：格式：RCR 目的, 计数值

字符串处理指令

字符串指令

潜规则

1. 源串 {DS：SI，允许段前超越前缀
2. 目的串{ES：DI，不允许段前超越前缀
3. SI,DI会自动修改{字节：1；字：2
4. DF={0,递增，DS：SI指向源串首地址，CLD将DF清0；1,递减，DS：SI指向源串末地址，STD将DF置1
5. CX ← 要处理的字符串长度(字节或字数)

重复前缀：
- REP（无条件重复）{与MOVS连用，CX ≠ 0时重复传送，直至CX=0
- REPE/REPZ（相等/结果为0则重复）{与CMPS连用，ZF=1和CX ≠ 0时则重复比较，直至ZF=0或CX=0
- REPNE/REPNZ（不相等/结果非0则重复）{与SCAS连用，ZF=0和CX ≠ 0时则重复扫描，直至ZF=1或CX=0

MOVS

传送

格式：MOVS 目的串，源串

功能：
- MOV数据传送:存储器 → 寄存器 → 存储器，必须修改地址指针
- MOVS数据传送:存储器 → 存储器，自动修改地址指针，重复前缀REP批量传送

实例：把数据段中以SRC_MESS为偏移地址的一串字符“HELLO! ，传送到附加段NEW_LOC 开始的单元

```
DATA        SEGMENT              ;数据段
SRC_MESS    DB 'HELLO!'          ;源串
DATA        ENDS
EXTRA       SEGMENT              ;附加段
NEW_LOC     DB  6  DUP(?)        ;存放目的串
EXTRA       ENDS
CODE        SEGMENT              ;代码段
            ASSUME CS:CODE, DS:DATA, ES:EXTRA
START:      MOV   AX,DATA
            MOV   DS,AX          ;DS=数据段段址
            MOV   AX,EXTRA       ;
            MOV   ES,AX          ;ES=代码段段址
            LEA   SI,SRC_MESS    ;SI指向源串偏移地址
            LEA   DI,NEW_LOC     ;DI指向目的串偏移地址
            MOV   CX,6           ;CX做串长度计数器
            CLD                  ;DF=0地址递增
            REP   MOVSB          ;重复传送串中的各字节，直至CX=0
CODE        ENDS
            END   START
```

REP MOVSB=

```
AGAIN:  MOVS  NEW_LOC,SRC_MESS
        DEC   CX
        JNZ   AGAIN
```

CMPS 比较

格式：CMPS 目的串，源串

功能：比较2个字符串是否相同，且与JNE连用，结果只反映在标志位上

实例：比较PASSWORD与IN_WORD，执行以下指令或扬声器报警拒绝执行以下指令

```
DATA        SEGMENT              ;数据段
PASSWORD    DB '8086CPUI'        ;口令串
IN_WORD     DB '8086CPU'         ;从键盘输入的串
COUNT       EQU  8               ;串长度
            …
DATA        ENDS
CODE        SEGMENT              ;代码段
            ASSUME CS:CODE, DS:DATA, ES:DATA
            . . .
            MOV   AX,DATA
            MOV   DS,AX          ;DS=数据段段址
            MOV   ES,AX          ;ES=代码段段址
            LEA   SI,PASSWORD ;源串指针
            LEA   DI,IN_WORD  ;目的串指针
            MOV   CX,COUNT     ;CX做串长度计数器
            CLD                  ;DF=0地址递增
            REPZ    CMPSB        ;CX≠0且串相等时重复比较
             JNE   SOUND         ;若不相等，转发声程序
OK:         . . .                ;比完且相等，往下执行
            . . .
SOUND:      . . .                ;使PC机扬声器发声
            . . .
CODE        ENDS
```

REPZ CMPSB=

```
AGAIN:  CMPS   IN_WORD,PASSWORD
        DEC    CX
        JNZ    AGAIN
```

SCAS 扫描

格式：SCAS 目的串

功能：从AL(字节)或AX(字)内容(关键字)-目的串，结果反映在标志位上

实例：在某一字符串中搜寻是否有A
- 若有，记录搜索次数→BX
- 没有，BX清0

设字符串其实地址STRING的偏移地址为0，字符串长度为CX

```
        MOV   DI,OFFSET STRING   ;DI=字符串偏移地址
        MOV   CX,COUNT           ;CX=字符串长度
        MOV   AL,'A'             ;关键字A的ASCII码
        CLD                      ;DF=0地址递增
        REPNE SCASB              ;CX≠0(没查完)和ZF=0(不相等)时重复扫描
        JZ    FIND               ;若ZF=1，表示已搜到，转出
        MOV   DI,0               ;若ZF=0，表示没搜到，DI←0
FIND:   MOV   BX,DI              ;BX←搜索次数
        HLT                      ;停机
```

DI自动修改，正好=搜索次数

LODS 装入

格式：LODS 源串

功能：AL(字节)或AX(字)←源串([SI])，遵循潜规则4

STOS 存储

格式：STOS 目的串

功能：AL(字节)或AX(字)→目的串，遵循潜规则4

实例：

设BLOCK为数据段一块带符号数据段的首地址，要求将其中的正、负数分开

正数送到附加段中始址为PLUS_DATA的缓冲器，负数送到MINUS_DATA

```
START:  MOV   SI,OFFSET  BLOCK          ;SI为源串指针
        MOV   DI,OFFSET PLUS_DATA       ;DI为正数目的区指针
        MOV   BX,OFFSET MINUS_DATA;BX为负数目的区指针
        MOV  CX,COUNT                   ;循环次数
        CLD                             ;DF=0地址递增
GOON:   LODS  BLOCK                     ;AL←取源串的一个字节
        TEST  AL,80H                    ;是负数?
        JNZ   MINUS                     ;是，转MINUS
        STOSB                           ;非负数，将字节送正数区
        JMP   AGAIN                     ;处理下一个字节
MINUS:  XCHG  BX,DI                     ;交换正负数指针
        STOSB                           ;负数送入负数区
        XCHG  BX,DI                     ;恢复正负数指针
AGAIN:  DEC   CX                        ;次数--1
        JNZ   GOON                      ;未处理完，继续传送
        HLT                             ;停机
```

STOSB指令必须以SI为源指针，DI为目的指针，但存储负数时，负数指针在BX，因此要用XCHG

控制转移指令

1. 无条件转移、过程调用指令
   1. JMP 无条件转移
      - 格式：JMP 目的
      - 功能：
      - 分类
        1. NEAR 段内转移
        2. FAR 段间转移

控制转移指令				
类型	格式	功能	操作数属性	备注
过程调用 返回	CALL 过程名 RET	见下表		

类型	方式	格式	
段内调用	直接	CALL PROG_N ((IP+3)入栈,SP←SP-2) RET (IP←SP/SP+1, SP←SP+2)	
	间接	CALL BX SP←SP-2,IP 入栈，IP←EA CALL WORD PTR [BX+SI] (IP←SP/SP+1, SP←SP+2)	
段间调用	直接	CALL FAR PTR PROG_F (SP←SP-2,CS 入栈 SP←SP-2,IP+5 入栈) RET SP←SP+2,IP 出栈 SP←SP+2,CS 出栈	
	间接(存储单元)	CALL DWORD PTR [BX]存储器寻址：IP←前 2 个字节 CS←后 2 个字节 RET n n 表示 CPU 在弹出 CS：IP 后再弹出 n 个字节 IP←4+n n 为偶数	

类型	方式	寻址目标	实例
无条件转移	JMP 目的(标号)		
段内转移 IP 变	直接 直接	立即短转移(8 位)(-128--+127) 立即近转移(16 位)(-32768--+32767) 短：DISP=目标地址偏移量-IP 当前值 IP=IP+DISP 负数用补码表示 近：IP=IP+3+DISP	JMP SHORT PROG_S JMP (NEAR PTR) PROG_N(或 JMP 标号)
	间接 间接	寄存器(16 位) 存储器(16 位)	JMP BX JMP WORD PTR 5[BX](存储器寻址方式)
段间转移 CS 变 IP 变	直接 间接	立即转移(32 位) 存储器(32 位)	JMP FAR PTR PROG_F JMP DWORD PTR [DI] 直接：CS←PROG_F 所在段段地址 IP←PROG_F 段偏移量

					间接：4 个连续地址单元 前 2 个：IP 后 2 个：CS	
条件转移	直接标志	指令	测试条件	功能		
		JC JNC	CF=1 CF=0	有进位 无进位		转移
		JZ/JE JNZ/JNE	ZF=1 ZF=0	结果为 0/相等 不为 0/相等		
		JS JNS	SF=1 SF=0	符号为负 符号为正		
		JO JNO	OF=1 OF=0	溢出 无溢出		
		JP/JPE JNP/JPO	PF=1 PF=0	奇偶位为 1/为偶 奇偶位为 0/为奇		
	间接标志 通常位于 CMP 之后	无符号数比较测试	JA/JNBE CF∨ZF=0 JAE/JNB CF=0 JB/JNAE CF=1 JBE/JNA CF∨ZF=1		高于/不低于等于 高于等于/不低于 低于/不高于等于 低于等于/不高于	
		带符号数比较测试	JG/JNLE (SF∀OF)∨ZF=0 JGE/JNL SF∀OF =0 JL/JNGE SF∀OF =1 JLE/JNG (SF∀OF)∨ZF=1		大于/不小于等于 大于等于/不小于 小于/不大于等于 小于等于/不大于	
循环控制	LOOP		LOOP 短标号	DEC CX JNZ 标号		
	LOOPE/LOOPZ		LOOPE/LOOPZ 标号	CX≠0 和 ZF=1 循环 CX=0 或 ZF=0 退出		
	LOOPNE/LOOPNZ		LOOPNE/LOOPNZ 标号			
	JCXZ		JCXZ 标号	CX=0 跳转，不对 CX 自减		
中断指令	除法中断 类型 0 单步中断 类型 1 不可屏蔽中断 类型 2 断点中断 类型 3 溢出中断 类型 4	INT n INTO 溢出中断指令 IRET 中断返回指令				
处理控制指令						
标志操作指令		CLC CMC	CF←0			

	STC	**CF←$\overline{CF}$** **CF←1**	
	CLD **STD**	**DF←0** **DF←1**	**自增**
	CLI **STI**	**IF←0** **IF←1**	
外部同步指令	**ESC 换码** **WAIT 等待** **LOCK 封锁总线**		**ESC 外部操作数，源操作数（实现 8086 对 8087 协处理器控制）BUSY，$\overline{TEST}$** **跟在 ESC 后使用$\overline{TEST}$** **$\overline{LOCK}$**
停机和空操作指令	**HLT 停机** **NOP 空操作**		

第四章　汇编语言程序设计

- 汇编语言
  - 1.汇编语言
    - 特点
      - 1低级语言2机器语言符号表示3面向机器4将被淡化
      - 1机密联系硬件2运行速度快3节省内存4程序效率高5实用性强
      - 高级语言
        - 可移植性好，编译
  - 2.源程序
    - 用汇编语言或高级语言编写的程序 —汇编→ 目标程序(机器语言编写的程序)
    - 基本语句：指令
  - 3.MASM6.X
  - 4.程序格式
    - 1.段定义
      - 段名　*SEGMENT*
      - ……
      - 段名　*ENDS*
      - 分段
        - 代码段
        - 数据段
        - 堆栈段
    - 2.段分配
      - ASSUME　CS：段名，DS：段名，SS：段名，ES：段名
    - 过程调用
      - 过程名　PROC　FAR(远调用)　(NEAR(近调用，可省))
      - ……
      - 过程名　ENDP
    - 3.设置段地址
      - 代码段：不能设置，计算机自动分配
      - 堆栈段/数据段
        - MOV　AX,STACK　;送堆栈段地址
        - MOV　SS,AX　;SS ← STACK
        - MOV　AX,DATA　;送数据段地址
        - MOV　DS,AX　;DS ← DATA
    - 4.返回DOS
      - 一 开头
        - PUSH　DS　;DS(段地址)
        - MOV　AX,0　必须在堆栈段定义之后
        - PUSH　AX　;0(偏移地址)
      - 二 结尾
        - MOV　AX,4C00H　;AH ← 4CH
        - INT　21H
    - 5.全程序结束
      - *START*
      - ……
      - *END*


汇编语言
程序格式
MASM
语句
1.指令性语句
(机器指令)
格式
标号： 指令助记符 操作数，操作数 ；注释
↑ ↑
符号地址
（可省）
：指令名称(核心，不可省)
2.伪指令语句
指示性语句
格式
名字 伪指令指示符 操作数，操作数 ；注释
↑ ↑
符号地址
（可省）
指令名称
数据项
基本
1.常数
1.0C350必须的
2.字符串： ‘ABC’
2.变量
1.段值
SEGMENT
：所在段段基址
2.段内偏移量
OFFSET
：变量地址与段首地址之间地址偏移字节数
3.类型
TYPE
：变量元素包含的字节数，如DB等，类型属性
3.标号
1.段值
2.段内偏移量
3.类型：在转移指令中标号可转移的距离
距离属性
FAR，NEAR
表达式
表达式=运算对象+运算符
运算符
1算术运算符
运算对象和结构都是整数
2逻辑运算符
按位运算，只对常数运算，结构也是常数
3.关系运算符
操作数必须是数据或同一段内的两个地址
4
数值返回运算符
分析运算符
1.OFFSET
2. SEG
3. TYPE
4.LENGTH
5. SIZE
5
修改属性运算符
综合运算符
1.段操作符
2.PTR
3.THIS
4.SFORT
5.
HIGH
LOW
6其它运算符
( )：改变优先级
[ ]：表示地址表达式或多重变量下标
<>
结构专用

4 数值返回运算符 / 分析运算符

1.OFFSET
- 格式：OFFSET 变量或标　　功能：返回标量或变量的偏移地址号
- 实例：MOV BX,OFFSET DA1 ~ LEA BX, DA1

2. SEG
- 格式：SEG 变量或标号　　功能：取标量或变量的段地址
- 实例：
  - MOV AX,SEG M1
  - MOV DS，AX

3. TYPE
- 格式：TYPE 变量或标号　　功能：
  - 变量 / 类型属性 {DB=1，DW=2，DD=4，DQ=8
  - 标号 / 距离属性 {NEAR=-1[OFFH]，NEAR=-2[OFEH]
- 实例：MOV BX,OFFSET DA1

4.LENGTH
- 格式：LENGTH 变量　　功能：
  - DUP=所包含的单元数
  - 其它变量=1
- 实例：

5. SIZE
- 格式：OFFSET 变量　　功能：返回变量总字节数SIZE=LENGTH*TYPE
- 实例：

5 修改属性运算符 / 综合运算符

1.段操作符
- 格式：段前缀 变量或地址表达式　　功能：修改提供段基址的寄存器，CS, DS, ES, SS
- 实例：
  - MOV AX,ES:[BX]

2. PTR
- 格式：类型/距离 PTR 变量或标号　　功能：将PTR左边的类型属性赋给右边的变量或标号
- 实例：
  - N1 DB 15H,36H
  - L0：MOV AX,WORD PTR N1

3. THIS
- 格式：变量/标号 EQU THIS 类型/距离　　功能：
  - 将EQU THIS右边的类型/距离属性赋给左边的变量 / 标号
  - 该变量 / 标号的段基址和偏移地址与下一个存储单元地址相同
- 实例：
  - FIRST EQU THIS BYTE
  - TABLE DW 200 DUP(?)

4. SFORT
- 格式：SFORT 标号　　功能：说明转移类指令中转向地址的属性，限制在短转移范围内-128 -
- 实例：
  - L1： JMP SHORT L2
  - ⋮
  - L2： MOV AX,0

5. HIGH / LOW
- 格式：HIGH/LOW 变量或标号　　功能：字节分离运算符，HIGH/LOW分离出高位/低位字节
- 实例：
  - K1 EQU 0ABCDH
  - MOV AH,HIGH K1

类型	符号	名称	运算结果
1.算术运算符	+/- *//		可地址运算 地址运算无意义

	MOD SHL SHR	模除	余数
2.逻辑运算符	AND OR XOR NOT		
3.关系运算符	EQ NE LT LE GT GE	相等 不等 小于 小于等于 大于 大于等于	结果为真输出全“1” 0FF/OFFFF 结果为假输出全“0”
4.数值返回	OFFSET SEG TYPE LENGTH SIZE	返回偏移地址 返回段基址 返回元素字节数 返回变量单元数 返回变量总字节数	
5.修改属性	段寄存器名 PTR THIS HIGH LOW SHORT	段前缀 修改类型属性 指定类型/距离属性 分离高字节 分离低字节 短转移说明	
6.其它运算符	（ ） [ ] 。 < > MASK WIDTH	 记录位图 记录宽度	改变运算优先级 下标或间接寻址 连接结构与变量 修改变量 位图形 记录/字段位数

伪指令语句

1.数据定义语句
- 格式
  - 格式1：变量名 助记符 操作数，操作数… ；注释
  - 格式2：变量名 助记符 n DUP(操作数，操作数…) ；注释
- 功能：将操作数存入变量名指定的存储单元中，或只分不存
- 分类{DB=1，DW=2DD=4，DQ=8，DT=10每个操作数所占字节数
- 操作数：常数，字符串DB，变量？，标号，表达式

2.标号赋值语句
- EQU
  - 格式：符号名 EQU 表达式
  - 解除：PURGE 符号名
  - 不允许重复定义
- =｛格式：符号名=表达式 允许重复定义

3.段定义语句
- 格式：段名 SEGMENT 定位类型 组合类型 分类名
  - •逻辑段内容
  - 段名 ENDS

4.段分配语句
- 格式：ASSUME CS：段名，DS：段名，SS：段名，ES：段名
- 取消：ASSUME ES：NOTHING

5.过程定义语句
- 格式：过程名 PROC 属性 ；CALL指令调用过程，过程允许嵌套和递归调用，深度由堆栈决定
  - ；过程内容
  - RET N ；弹出值，可省，必须为正偶数
  - 过程名 ENDP

6.程序 开始/结束 语句
- 1.NAME 程序名
- 2.ORG 表达式
- 3.END 表达式

7.群定义语句 {

8.结构定义语句
- 1.结构定义
  - 格式：结构名 STRUC
  - (用DB，DW等定义结构中数据变量)
  - 结构名 ENDS
- 2.副本预置
  - 格式1：结构副本名 结构名〈元素值, 元素值•••〉 ；注释
  - 格式2：结构副本名 结构名 N DUP（〈元素值, 元素值•••〉） ；注释
  - 结构副本名•变量名
- 3.结构使用
  - 直接
  - 间接

9.记录定义语句

外部伪指令
- 格式：PUBLIC 名称，名称,••• ；注释
- EXTRN 名称：类型，名称：类型,••• ；注释

对准伪指令{格式：EVEN 功能：使下一语句的地址调整为偶地址

LABEL
- 格式：名称 LABEL 类型属性
- LABEL与变量连用：BYTE, WORD
- LABEL与标号连用：FAR，NEAR

DOS

BIOS{

- 程序设计方法
  - 好程序特点
    - 1程序结构模块化，易读易调试易维护
    - 2执行速度快
    - 3占用内存空间小
  - 设计源程序
    - 1分析问题 → 属性模型 → 算法
    - 2流程图
    - 3分配存储空间及工作单元
    - 4按流程图设计编写程序
    - 5静态检测，上机调试(分段调试好)
    - 6程序允许，结构分析
  - 顺序结构
  - 分支结构
    - 多分支
    - 跳转表实现多分支
      - 1根据表内地址分支：表地址=跳转表首地址+偏移地址
      - 2根据表内指令分支
      - 3根据关键字分支
  - 循环结构
    - 先执行，后判断：循环次数固定
    - 先判断，后执行：循环此时不固定
    - 多重循环

    循环计数判断可通过LOOP/LOOPE，也可用条件跳转语句
  - 子程序/过程结构
    - 定义：PROC•••ENDP 调用：CALL 返回：RET
    - 参数传递
      - 1寄存器：参数较少，速度快
      - 2存储器：参数较多，需事先在存储器中建立一个参数表
      - 3堆栈：参数较多，尤其在嵌套和递归调用情况下不易出错
    - 嵌套与递归

第五章　存储器

- 存储器
  - 外存
    - 软盘
    - 硬盘
    - 磁盘
    - → 磁记录存储器
    - 闪存盘 → 新型存储器
    - 光盘
  - 内存
    - RAM 随机存取
      - SRAM
      - DRAM
      - → 断电全失
    - ROM 只读
      - PROM
      - EPROM
      - EEPROM
      - Flash Memory
      - → 断电保留
      - → 只能读出已存储的内容，不能随意写入
      - → ROM中通常存储操作系统的程序或者用户固化的程序
    - 内存容量：大小受到地址总线位数的影响

- 存储器性能指标
  - 1.存储容量
    - 内存大小受地址总线位数限制
    - 基本存储单元：位(bit)
    - 计算单位：字节或字
    - SRAM最小，外存 → 海量存储器
  - 2.存取时间 → SRAM(Cache) < DRAM(内存条)
  - 3.功耗 → SRAM>DRAM； MOS<双极型
  - 4.可靠性 → Flash存储器
  - 5.价格 → SRAM最贵>DRAM

- RAM
  - SRAM
    - 1.构成
      - 1.存储矩阵
        - 字结构 → 读写一个字节的8位制作在一块芯片上:1K=128×8，7根地址线8根数据线1块芯片，封装时引线较
        - 位结构 → 读写一个字节的8位由8块芯片组成:1K=1024×1，10根地址线1根数据线8块芯片，封装时引线较少
      - 2.地址译码器
        - 功能：完成存储单元的选择
        - 方式
          - 线性译码
          - 复合译码
      - 3.控制逻辑
      - 4.三态数据缓冲器
    - 2.SRAM例子：CPU的Cache就是用SRAM构成的
  - DRAM
    - 1.原理：利用电容存储电荷来保存信息的
    - 2.主要特点：刷新和地址两次打入
    - 3.刷新
      - 1.定义：把写入到存储单元的数据进行读出，经过读放大器放大之后再写入以保存电荷上的信息
      - 2.按行刷新，每刷新一行的时间称为刷新周期
      - 3.刷新方式：集中刷新方式、分散刷新方式
      - 4.DRAM控制器
        - 1.地址多路器：CPU地址 → 行地址、列地址，实现DRAM芯片地址的两次打入
        - 2.刷新定时器
        - 3.刷新地址计数器
        - 4.仲裁电路
        - 5.定时发生器
    - 4.DRAM例子：内存条，通常所说的计算机内存就是指RAM存储器的容量

- 内存条
  - 1.SDRAM
  - 2.DDR SDRAM
  - 3.DDR2 SDRAM

	SDRAM	DDR SDRAM	DDR2 SDRAM
预读数据	1bit	2bit	4bit
数据传输率	1/CKL	2/CKL	4/CKL
工作电压	3.3V	2.5V	1.8V
封装类型	TSOP-Ⅱ 54pin	TSOP-Ⅱ 66pin	FFGA 60/64/68/84/92pin
模组标准	168pin DIMM	184pin DIMM	240pin DIMM

DRAM 控制器逻辑框图

- Cache
  - SRAM+常访问代码和数据
  - 结构
    - 全相联Cache
    - 直接映像Cache
    - 成组相联Cache
  - 构架
    - 1.读取结构
      - 1.旁视结构
        - 优点
          - 1.简单，成本低
          - 2.Cache和主存同时监视系统总线，数据请求的延迟低
        - 缺点
          - 1.系统总线占用率高
          - 2.其它系统主控器正在访问主存时，CPU不能访问Cache
      - 2.通视结构
        - 优点
          - 1.其它系统主控器访问主存时，CPU依然能访问Cache
          - 2.降低了CPU对系统总线请求次数
        - 缺点
          - 1.没有命中时，对主存的访问时间要比旁视的慢
          - 2.复杂
    - 2.写入策略
      - 1.通写法 CPU → Cache → 立即写入主存
      - 2.回写法 CPU → Cache → 空闲写入主存
    - 3.替换策略
      - 1.随机法
      - 2.先进先出法FIFO
      - 3.最近最少使用法LRU
  - 影响Cache性能因素
    - 1.Cache规模大小
    - 2.关联方式
    - 3.Cache行大小
    - 4.速度
    - 5.配置

- ROM
  - 1.掩膜型ROM
    - 用户不能对数据进行任何修改
  - 2.可编程ROM(PROM)
    - 只允许编程一次，不能修改
    - 组成：二极管矩阵组成
  - 3.可擦除可编程ROM(EPROM)
    - 紫外线擦除
    - 实现技术：浮栅雪崩式技术
    - 信息存储：电荷分布决定(浮置栅)
    - 典例：Intel 2764(8K×8)
  - 4.电可擦除可编程ROM(EEPROM)
    - 擦除：“电子隧道”
    - 优点
      - 1.在应用系统中能在线读写断电保存
      - 2.不需设置单独擦除操作，可在写入过程中自动擦除
      - 3.写入是数据可保存10年以上，可重复擦除/编程上万次
    - 分类
      - 1.并行接口
        - 特点
          - 1.读写方法简单，有字节写入方式和页写入方式
          - 2.速度快，功耗大，价格高，容量较大
        - 典例：NMC98C64A(8K×8)
      - 2. 串行接口
        - 特点
          - 1.读写方法复杂，占有系统信号线少
          - 2.速度慢，功耗小，价格低，体积小
        - 典例:24C64(8K×8)
  - 5.闪存Flash
    - 特点
      - 1. 高集成度(采用单管单元)
      - 2.寿命长，速度快
    - 编程方法
      - 1.沟道热电子注入(多线程重新数据)
      - 2.隧道效应:EEPROM(单线程重写数据)
    - 分类
      - 1.NOR：BIOS或固件、手机和手持装置等
      - 2.NAND：不适合做可编程ROM，很适合于存储卡

- CPU与M连接
  - 1.连接问题
    - 1.CPU总线的负载能力
    - 2.CPU的时序和M存取速度之间的配合
    - 3.M地址分配和片选
    - 4.控制信号的连接
  - 2.M地址选择
    - 1.M寻址两部分
      - 低地址线→M芯片：内寻址
      - 高地址线→译码器→线性组合→片选信号→片间寻址
    - 2.地址选择方法
      - 1.线性选择方式
      - 2.全译码选择方式
      - 3.部分译码选择方式
  - 3.M数据线连接
  - 4.M控制线连接

第六章　I/O 接口　和　总线

- I/O接口
  - 1.通信：JSJ与外设(I/O设备)间交换数据、状态、控制命令的过程
  - 2.通信问题
    - 1.速度不匹配
    - 2.信号电平不匹配
    - 3.信号格式不匹配
    - 4.时序不匹配
  - 3.接口电路功能
    - 1.设置数据缓冲以解决速度差异所带来的不协调问题
      - 输出：锁存器
      - 输入：缓冲器；寄存器与数据总线之间
    - 2.设置信号电平转换电路
      - 串行通信：MAX232和MAX233芯片
    - 3.设置信息转换逻辑以满足对各自格式的要求
    - 4.设置时序控制来同步CPU和外设的工作
    - 5.提供地址译码电路
    - 1数据缓冲与锁存
      - 输入缓冲、输出锁存是对IO接口的最基本要求。
    - 2电平变换与驱动
    - 3数据类型的变换
      - 并行：以字节/字为单位传送；速度快，短距离
      - 串行：逐位传送；速度慢，长距离
    - 4器件选择与地址译码
      - 片选信号$\overline{CS}$
    - 5命令译码与错误检测
      - 串行接口具有错误检测功能
      - 错误类型
        - 1传送错误
        - 2覆盖/重叠错误
        - 3帧格式错误
    - 6定时与控制
  - 4.I/O接口芯片
    - 缓冲器
      - 74LS244
        - 单向,8路(双4)
      - 74LS245
        - 双向,8路
    - 锁存器
      - 74LS373:8位寄存器+8位三态缓冲器

- I/O端口
  - 通信信息：分别进入的寄存器和它们的控制逻辑统称为I/O端口，但都通过数据总线传输信息
    1. 数据信息
    2. 状态信息 / 输入数据
    3. 控制信息 / 输出信息
  - 端口分类
    1. 数据端口：作用：数据缓冲
    2. 状态端口：作用：指示外设当前状态，每种状态用1位表示
       - 常用状态位
         1. 准备就绪位
            - 输入端口：1：端口的数据寄存器已经准备好数据，等待CPU来读取；0：数据被取走后，该位清零。
            - 输出端口：1：表输出端口数据寄存器已空，即上一个数据已被外设取走，可接；0：新数据到达
         2. 忙碌位：1：外设正在进行输出数据传送操作，暂时不允许CPU送新的数据来；0：本次数据传送完毕，该位清0，外设空闲，允许CPU送新数据来
         3. 错误位：1：数据传送过程出现错误
    3. 命令端口 / 控制端口
  - I/O端口寻址方式
    1. 存储器映像寻址方式
       - 优点：CPU的指令集中不必包含I/O操作指令，简化了指令系统的设计
       - 缺点：I/O端口占用了存储单元的地址空间
    2. I/O单独编址方式
       - 优点：I/O端口不占用内存空间, 速度快
       - 缺点：
         1. CPU指令集中必须有专门的IN/OUT指令，功能弱
         2. CPU还需提供能够区分访问内存和I/O的硬件引脚信号

- CPU与外设间数据传送方式
  - 软件CPU控制方式
    1. 程序控制方式
       1. 无条件传送：同步传送方式
       2. 条件传送：查询传送方式
    2. 中断方式：提高CPU利用率
  - 硬件控制方式
    3. DMA方式
       - 引入：提供传输速度
       - 关键：CPU从总线脱离，即总线浮空（高阻状态）

- 总线
  - 1.分类
    - 信息传送类型
      1. 地址总线
      2. 数据总线
      3. 控制总线
    - 规模用途场合
      1. 片级总线：实现CPU主板或其它插件版上的各种芯片间的互连
      2. 系统总线：或$\frac{\text{内总线}}{\text{板级总线}}$，实现微机中各插件版（部件）间连接，即微机总线
      3. 外部总线：通信总线，实现微机系统间或与其它电子仪器或设备间的通信
  - 2.标准
    - 1.标准系统总线
      - IBM PC机的62芯PC总线
      - PC/AT机的AT总线，即ISA总线
      - 在ISA基础上升级的32位EISA总线
      - PCI总线
    - 2.标准外部总线
      - IEEE-488总线
      - EIA RS-232总线
  - 3.周期
    - 定义：BIU通过总线访问（读/写）M或I/O口所需的时间。
    - 时钟周期T：CPU最小时间单位，时钟频率$f=1/T$
    - 总线周期=4时钟周期
    - 作用
      - T_1：BIU送出20位地址
      - T_2：BIU撤销20位地址
      - T_3：BIU与M或I/O传送数据
      - T_4：结束总线周期

第七章　微型计算机中断系统

- 中断概念
  - 作用：避免不断检测外部设备状态，提高CPU的效率
  - 概念
    - 1.中断源
      - 内部中断：程序预先安排的中断指令（INT n）/CPU运算错误
      - 外部中断：外设或协处理器向CPU发出的中断申请
    - 2.中断响应
    - 3.中断向量表
    - 4.中断优先级
    - 5.中断屏蔽：软件设置
  - 功能
    - 1.实现中断响应、中断服务、中断返回、中断屏蔽
    - 2.实现中断优先级排队
    - 3.实现中断嵌套
  - 分类
    - 1.外部中断/硬件中断
      - 1.不可屏蔽中断（NMI）
        - 1.系统板上的动态RAM有奇偶校验错误
        - 2.I/O通道的扩展板出现奇偶校验错误
        - 3.协处理器8087有中断请求
      - 2.可屏蔽中断（INTR）
        - 外设中断得到响应条件
          - 1.外设中断请求是否被屏蔽
          - 2.CPU是否允许响应中断(中断允许位IF=
            - 1允许，S
            - 0禁止，C
    - 2.内部中断/软件中断
      - 1.由中断指令INT引起的中断：类型2
      - 2.由CPU某些运算错误引起的中断
        - 1.除法错中断：类型0
        - 2.溢出中断：类型4OF=1，INTO
      - 不可屏蔽中断，不必判断IF=1
      - 3.由调试程序debug设置的中断
        - 1.单步中断：TF=1，类型1
        - 2.断点中断：类型3

- 可屏蔽中断处理过程
  - 过程
    - 1.中断请求
    - 2.中断响应
    - 3.保护现场　　书中有流程图
    - 4.转入中断服务子程序
    - 5.恢复现场和中断返回
  - CPU响应中断过程
    - 响应条件
      - 1.外设提出申请中断：中断接口
      - 2.本中断位未被屏蔽
      - 3.本中断优先级最高
      - 4.CPU允许中断
    - 工作
      - 1.从数据线上读取中断类型号，将其存入内部暂存区
      - 2.将标准寄存器flags的值入栈
      - 3.将flags中的中断允许标准IF和单步标准TF清0，以屏蔽外部其它中断请求，以避免CPU以单步方式执行中断处理子程序
      - 4.保护断点，将当前指令下面一条指令的段地址CS和指令指针IP的值入栈，使中断处理完毕后，能正确返回主程序继续执行。
      - 5.根据中断类型号到中断向量表中找到中断向量，转入相应中断服务子程序
      - 6.中断处理程序结束后，从堆栈中依次弹出IP、CS和flags，然后返回主程序处，继续执行原理的程序。
    - 中断向量表
      - 寻找中断源
        - 1.查询中断
        - 2.矢量中断
      - 1.中断向量表
        - 中断服务程序入口地址表
      - 2.中断向量设置
        - 1.指令来设置
        - 2.DOS调用设置
      - 3.中断类型号获取
        - 1.对于除法出错，单步中断，不可屏蔽中断NMI，断点中断，和溢出中断，CPU自动提供中断类型号0～4.
        - 2.对于用户自己确定的软件中断INT n，类型号由n确定
        - 3.对外部可屏蔽中断INTR，可用硬件电路设计产生中断类
        - 4.对外部可屏蔽中断INTR，可以用可编程中断控制器825获得中断类型号
    - 中断主程序编写方法
      - 1初始化
        - 1硬件中断向量
        - 2设置8259A的中断屏蔽寄存器的中断屏蔽位
        - 3设置CPU中断允许位标志IF(开中断STI)
      - 2 硬件(外设接口)和CPU自动完成
        - 1外部中断控制器接口向CPU INTR端发中断请求
        - 2当前指令执行完后，CPU发两个中断响应信号$\overline{INTA}$给外部中断控制器接口
        - 3CPU取中断类型号
        - 4CPU自动将当前flags、CS、IP内容入栈保护
        - 5清除IF、TF，禁止外部中断和单步中断
        - 6从中断向量表中取(4n)地址中内容→IP，去(4n+2)地址
        - 7转向中断服务子程序
      - 注意点
        - 1对重复前缀的指令(如REP MOVSB)作为一条指令处理。执行一条重和串指令即可响应中断，而不是把串操作完全执行完。
        - 2遇到开中断指令STI和中断返回指令IRET，要在这两条指令执行完再执行下一条指令才能响应中断。
        - 3CPU自动清除IF、TF位，使得CPU进入中断服务程序后，不允许再如果在中断服务程序中还允许外部中断进入，那么在中断服务程序

- 中断服务子程序
  - 结构形式
    - 1程序开始必须保护中断现场，PUSH入栈保护
    - 2若允许中断嵌套，则用STI指令来设置开中断IF=1
    - 3执行中断处理程序
    - 4CLI设置关中断，IF=0，禁止其它中断请求进入
    - 5送中断结束命令EOI，清除当前中断请求标志位
    - 6恢复中断现场，POP出栈恢复
    - 7中断返回IRET返回主程序IP、CS和flags

- 中断响应时序
  - 1.第一个$\overline{\text{INTA}}$：CPU通知外部中断控制器准备响应中断，外设应该准备好中断类型号
  - 2.第二个$\overline{\text{INTA}}$：CPU接受外部中断控制器接口发来的中断类型号
  - 说明
    - 1.
    - 2.
    - 3.
    - 4.
    - 5.

- 中断优先级
  - 前言
    - 1.中断服务前要开中断，以允许中断嵌套
    - 2.恢复现场前要关中断，使恢复现场不受干扰
    - 3.中断返回后自动开中断，以允许其它中断能被CPU响应
  - IBM PC优先级由高到低
    - 1.内中断(除法错，INTO，INT n)
    - 2.不可屏蔽中断(NMI)
    - 3.可屏蔽中断(INTR)
      - 方法
        - 1.软件查询中断优先级
          - 将外设中断请求信号“或”后送到CPU的INTR端
          - 查询次序决定外设优先级高低，先测试的中断源优
        - 2.硬件查询中断优先级--菊花链法
          - 每个外设对应接口上连接一个逻辑电路
          - 外设中断申请信号 → CPU，INTR → $\overline{\text{IN}}$
          - 越靠近CPU的外设接口优先级越高
        - 3.矢量中断优先级
          - 中断优先级控制器：如8259A
    - 4.单步中断

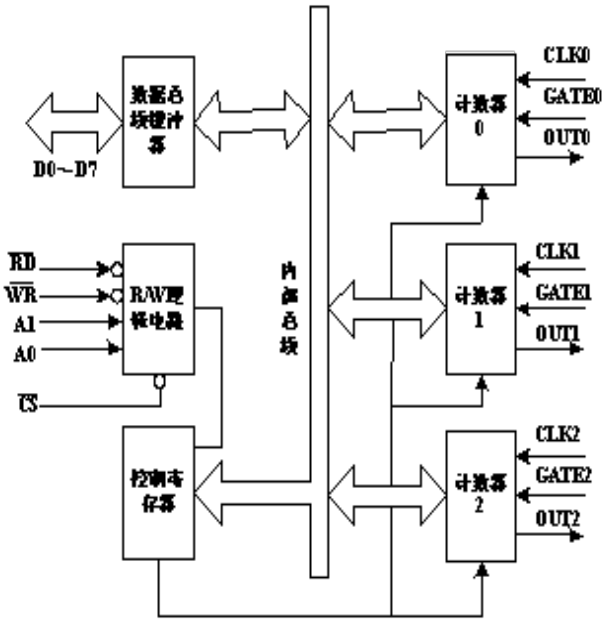
- 中断嵌套/多重中断
  - IBM PC没有规定中断嵌套深度，但受堆栈容量限制：保存嵌套的断点及寄存器
  - 说明
    - 1.
      - 主程序必须有开中断指令，是IF=1才能响应中断。
      - 中断服务子程序必须有STI开中断指令，实现中断嵌套。
    - 2.
      - 中断结束返回要有EOI中断结束命令，用来清除中断服务寄存器中的对应位，允许低级中断进入。
      - 最后有中断返回指令IRET
    - 3.
      - IF=1，能被中断优先级更高的中断源中断。
      - IF=0，即中断处理中提前发出了EOI命令，则清除了正在执行的中断服务，允许响应同级或低级中断。

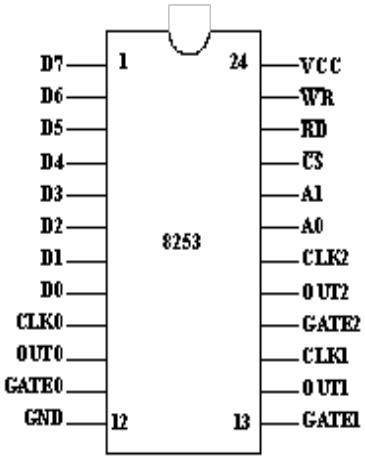
- 8259A
  - 1.功能和引脚
  - 2.内部结构
    - 1.数据总线缓冲器
    - 2.读写控制电路
    - 3.级联缓冲/比较器
    - 4.中断请求寄存器IRR
    - 5.中断屏蔽寄存器IMR
    - 6.优先级判别器PR
    - 7.中断服务寄存器ISR
    - 8.控制电路
  - 3.中断管理方式
    - 1.8259A编程结构
    - 2.优先级设置方式
      - 1全嵌套方式：固定级别0～7级，优先级 $IR_0 \rightarrow IR_7$
      - 2特殊全嵌套工作方式：实现了对同级中断请求的特殊嵌套，专门为多片8259A提供
      - 3优先级自动循环方式：优先级可改变，用操作命令子OCW_2中R，RL=10设置
      - 4优先级特殊循环方式：初始时最低优先级由程序规定，最高级也就确定了
    - 3.中断结束方式
      - 1普通EOI结束方式
      - 2特殊EOI结束方式
      - 3自动EOI结束方式
    - 4.循环优先级的循环方式
      - 1普通EOI循环方式
      - 2特殊EOI循环方式
      - 3自动EOI循环方式
    - 5.中断源屏蔽方式
      - 1普通屏蔽方式
      - 2特殊屏蔽方式
    - 6.中断请求引入方式
      - 1边沿触发方式
      - 2电平触发方式
      - 3中断查询方式
        - 1系统关中断
        - 2用OUT指令使CPU向8259A端口(偶地址)送OCW_3命令字
        - 3CPU用IN指令从端口（偶地址）读取8259A的查询字
  - 4.编程方法
    - 1初始化命令字
      - 功能
        - 1设定中断请求信号触发形式，高/上升沿触发
        - 2设定8259A工作方式，单片或级联
        - 3设定8259A中断类型号基值，即IR_0对应的中断类型号
        - 4设定优先级设置方式
        - 5设定中断处理结束时的结束操作方式
      - 1)ICW_1--芯片控制初始化命令字
      - 2)ICW_2--设置中断类型号初始化命令字
      - 3)ICW_3--标识主/从片初始化命令字
      - 4)ICW_4--方式控初始化命令字
    - 2操作字命令字
      - 功能：1决定中断屏蔽、优先级、中断结束方式等
      - 1)OCW_1--中断屏蔽操作操作字
      - 2)OCW_2--优先级循环方式 / 中断结束方式 操作字
      - 3)OCW_3--特殊屏蔽方式 / 查询方式 操作字
  - 5.中断级联
    - 1.注意点
    - 2.级联举例

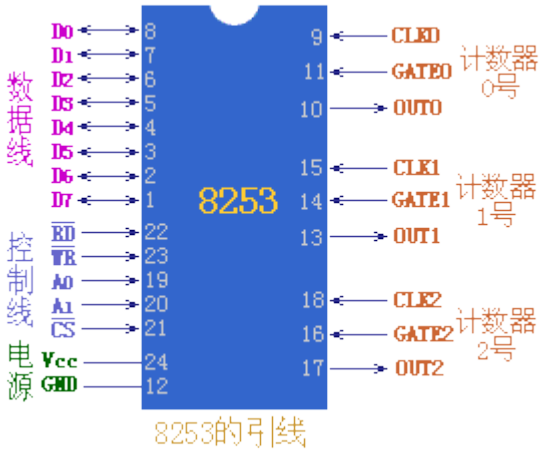
- 保护模式下的中断
  - 1中断源
    - 1中断
      - 1可屏蔽中断
      - 2不可屏蔽中断
    - 2异常
      - 1故障
      - 2陷阱
      - 3中止
    - 3中断类型码
    - 4中断优先级
  - 2中断处理
    - 1中断描述符
      - 1门描述符
      - 2中断描述符表IDT
      - 3中断描述表寄存器IDTR
    - 2获取中断服务子程序入口地址{123
    - 3中断处理过程
      - 1由CPU内部或外部获取中断源的中断类型号
      - 2将断点信息压入特殊级最高的核心堆栈进行保护并清除IF标志。
      - 3寻址中断服务子程序的入口地址
      - 4执行IRET指令使中断返回，恢复现场

第八章

可编程计数器/定时器 8253/8254 及其应用







8253的引线

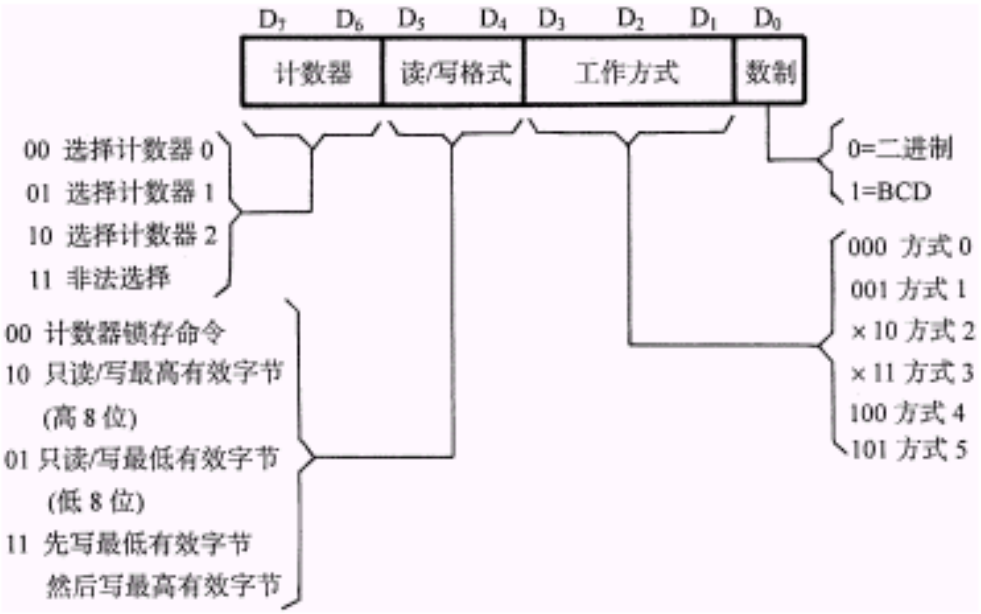


图 10-3　8253 的控制字

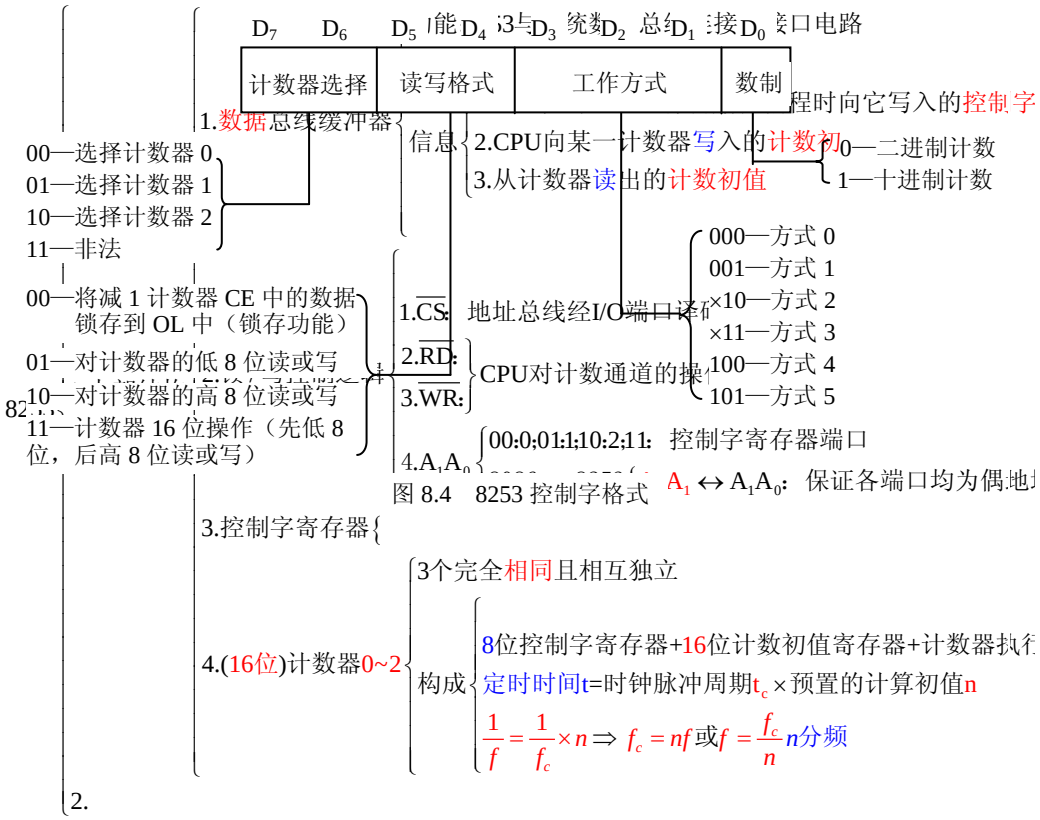


图 8.4　8253 控制字格式

2.

8253初始化

1.步骤

1.先写入控制字

2.后写入计数初值

8位：1条OUT指令

16位：2条OUT指令，先低8位后高8位；特别0

二进

BCD

2.实例

8253的3个计数器及控制字寄存器端口地址：3F0H、3F2H、3F4H、3F6H

```
初始化：MOV AL,00110111B   ;控制字
        MOV DX,3F6H        ;指向控制字寄存器端口
        OUT  DX,AL         ;送控制字
        MOV AL,34H         ;计数值低字节
        MOV DX,3F0H        ;指向计数器0端口
        OUT  DX,AL         ;先写入低字节
        MOV AL,12H         ;计数值高字节
        OUT  DX,AL         ;后写入低字节
```

读操作：D_5D_4=00，OUT → IN即可

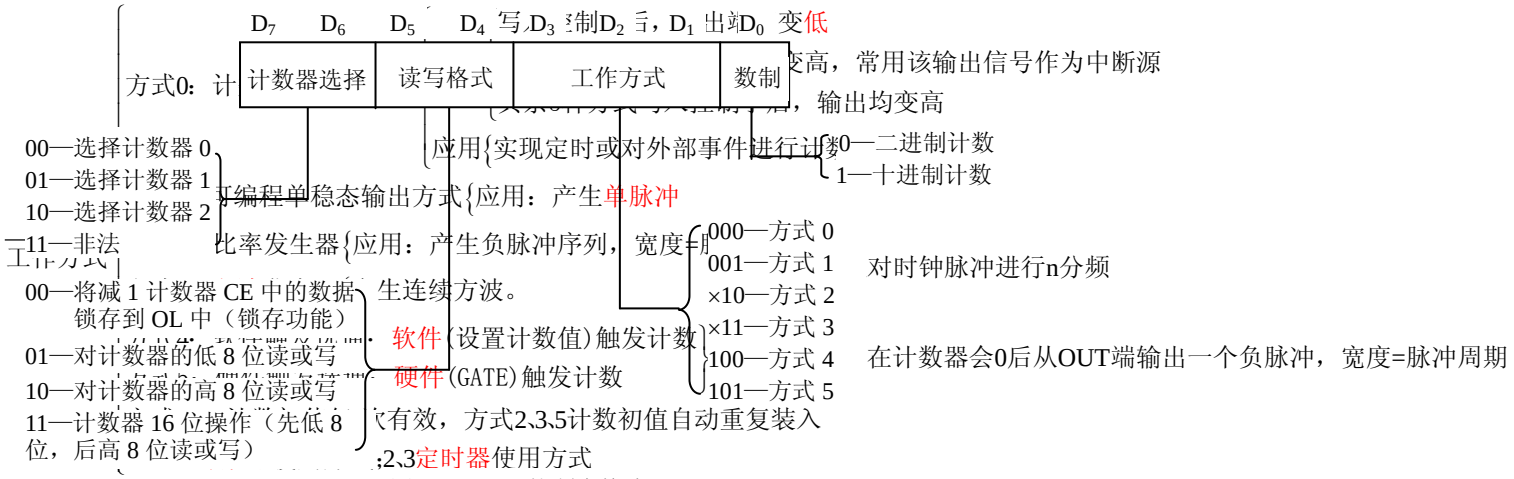


图 8.4　8253 控制字格式

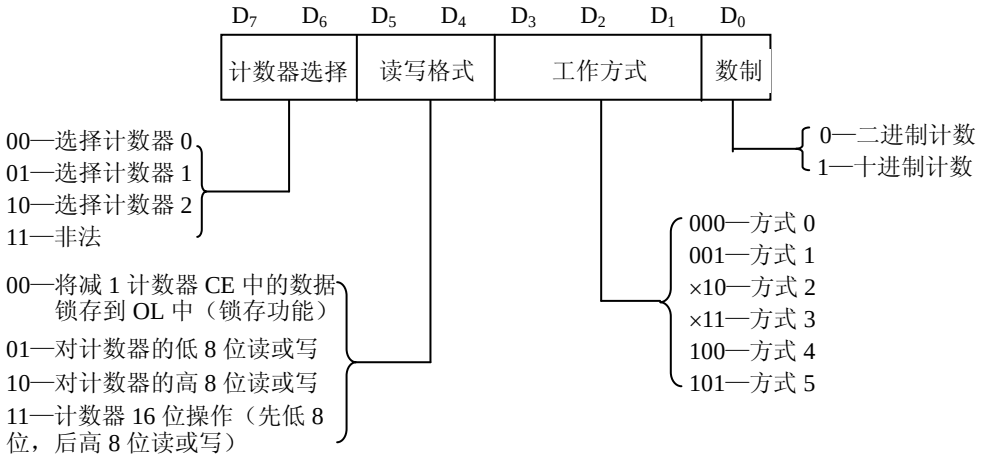


图 8.4　8253 控制字格式

计数/定时
- 1.软件方法
  - 子程序循环指令
  - 优点：简单，经济(节省硬件)
  - 缺点：CPU一直被占用，利用率低
- 2.硬件方法
  - 1.要求低：中小规模集成计数器或定时器(555)
  - 2. 要求高：大规模集成计数/定时器，可编程

第十章　串行通信和可编程接口器件

8251A

- 通信
  - 并行通信
    - 优点：速度快，适合近距离传送数据
    - 缺点：通信线多，成本高，可靠性随距离增加↑而下降↓
  - 串行通信
    - 缺点：速度慢，传输时间长
    - 优点：通信线少，成本低，兼容性广，适合远距离传送数据
- 串行通信
  - 数据传送方向
    - 1.单工：A → B，单方向传送
    - 2.半双工：A → B或B → A，双方向，但同一时间内只能单方向传送；对讲机
    - 3.双半工：A ⇄ B，2条不同的通路，同时间内可以同时传送、接收数据，如电话
  - 两种基本工作方式
    - 1.异步方式：起始位，停止位，传输速率低
    - 2.同步方式：同步字符，同步传送波特比>异步传送波特比
  - 传送速率
    - 波特比Bd：每秒传送数据的位数，如10位/字符×120字符/秒=1200波特
    - 同步传送波特比>异步传送波特比
  - 串行接口芯片硬件
    - 引入
      - 计算机数据传送方式：并行
      - 数据发送：并行输入，串行输出移位寄存器
      - 数据接收：串行输入，并行输出移位寄存器
    - 分类
      - UART
        - 通用异步收发器；如INS 8250
        - 组成
          - 1.接收器：串行码 → 并行码
          - 2.发送器：并行码 → 串行码
          - 3.控制器：接收CPU的控制信号并执行及输出状态、控制信息
      - USART：通用同步异步收发器；如Intel 8251A
  - 调制解调器MODEM
    - 引入
      - 使用串行接口芯片不适宜长距离传送
      - 电话线智能传送带宽为300Hz ~3000Hz
    - 调制：数字信号 → 音频信号
    - 解调：音频信号 → 数字信号
    - 数字调制
      - 幅度调制(幅移键控ASK)
        - 原理：用改变信号幅度方法表示数字信号0和1
        - 适用性：非常低速的反向通道传输，或与其它调制方法协用
      - 频率键移调制FSK
        - 原理：用频率信号表示数字信号0和1
        - 适用性：局限于以1200Bd的速率在两芯电话线上进行半双工通信
          或以1200Bd的速率在四芯电话线上进行全双工通信
      - 相位键移PSK：调相多路载波
        - 更高速率，较复杂
- 可编程串行通信接口芯片8259A

第十一章　A/D 和 D/A 转换

多路切换方法
- 1.外加多路模拟开关MUX
- 2.选用内部带多路开关的A/D转换器

采样量化编码
- 采样：
  - 按相等时间间隔t从电压信号上截取一个个离散的电压瞬时值，这些值有精确大小。
  - $f_s=1/\Delta$，t_0
  - 采样保持器：波动误差
- 量化：
  - 分层的其实电平就是采样的数字量。
  - 单位：q；
    - 量化单位越小，精度越高（采样率f_s越高，每秒内采样的点数接越多，采得的值就越接近原信号）。
    - $q=\frac{\text{电压量程}}{2^n}$，q=A/D输出数字量中最低位LSB=1时所对应的电压值，所以也称1LSB
  - 分层数：2^n(必须的)
  - n：表示A/D分辨率，表明A/D转换器能分辨最小量化信号的能力。
- 编码
  - 自然二进制码
  - 二进制小数形式
    - $N=d_1 2^{-1}+d_2 2^{-2}+\cdots+d_n 2^{-n}$($d_n$=1或0)、
    - 小数点不必表示出来
    - $V_X=V_R\times N$
  - 双极性二进制码

D/A
- 原理
  - $I_O=d_1I_1+d_2I_2+d_3I_3+d_4I_4=d_1\frac{V_R}{R}+d_2\frac{V_R}{2R}+d_3\frac{V_R}{4R}+d_4\frac{V_R}{8R}=\frac{2V_R}{R}(d_1 2^{-1}+d_2 2^{-2}+d_3 2^{-3}+d_4 2^{-4})$
  - $V_O=-I_f\bullet R_f$
  - $I_f=I_O$
- 性能
  - 1.输入数字量
  - 2.输出模拟量
    - $I_{O1}=\frac{V_R}{15k\Omega}\times\frac{\text{输入数字量}}{2^8}$
    - $I_{O1}=\frac{V_R}{15k\Omega}\times\frac{(2^8\text{-}1)-\text{输入数字量}}{2^8}$
  - 3.分辨率 $\Delta=\frac{FSR}{2^n}=q$
  - 4.精度5.建立时间
- 举例
  - AD7524
  - DAC0832 工作方式：1.直通方式2.单缓冲方式3.双缓冲方式
  - DAC1210

A/D
- 原理：逐次逼近法
- 举例
  - ADC0809
  - AD574A

软件概念及数据结构

软件发展

- 软件定义：计算机系统中所有计算机（程序 / 主体）以及开发、使用、维护程序所需的所有文档与数据的总称。
- 阶段：
  1. 机器语言(20世纪40年代中)
  2. 汇编语言(20世纪50年代初)
  3. 高级语言(20世纪50年代中)
  4. 操作系统(20世纪60年代中)
  5. 高性能软件(20世纪70年代中之后)
- 计算机系统分层结构：主机 → 外围设备 → 操作系统 → 语言处理程序及*DBMS*、实用程序 → 应用软件
- 软件开发技术：1. 结构化方法 / 面向数据流方法　2. 快速原型法　3. 面向对象方法 / 标志：Simula语言
- 信息处理方法：1. 数据处理 ⊃ 2. 信息处理 ⊃ 3. 知识处理 ⊃ 4. 智能处理

- 算法
  - 特性
    1. 有穷性：步骤有穷，时间有穷；与程序的根本区别
    2. 确定性：步骤含义确定
    3. 可行性：执行有限次可完成
    4. 输入：≥ 0
    5. 输出：≥ 1
  - 描述
  - 分析
    - 设计原则
      1. 正确性
      2. 可读性
      3. 健壮性
      4. 效率和存储空间需求
    - 时间复杂度 $T(n)=O(f(n))$
      - 有效算法：$f(n)=O(\log_2 n / n^k)$
      - 无效算法：$f(n)=O(a^n / n!)$
    - 空间复杂度

- 操作系统
  - 出现标志：多道程序系统和分时系统的出现。

- 数据
  - 数据：能被计算机识别、存储和处理的符号的集合。
  - 数据元素
    - 数据的基本单位。结构基本单位
    - 包括：编号、型号、合格证、废品、日期
  - 数据项：组成数据元素，是数据的最小单位。
  - 数据结构
    1. 逻辑结构
       1. 集合
       2. 线性结构
       3. 树形结构
       4. 图
    2. 存储结构 / 物理结构
       - 数据元素的表示
       - 关系的表示
         1. 顺序方式
            - 位置
         2. 链接方式
            - 结点通过指针链接
              - 数据域+指针域
         3. 索引方式
         4. 散列方式
    3. 数据运算 / 操作
       - 1. 插入 2. 删除 3. 更新 4. 查找 5. 排序
       - 算法实现方式
         1. 以调用数据结构中的若干基本运算来形成算法。
         2. 将包括基本运算在内的运算过程形成一个完整的算法。

- 线性表
  - 1°逻辑结构
    - $L=(a_1, a_2, \cdots a_{i-1}, a_i, a_{i+1}, \cdots a_n)$同一个线性表中的元素特性相同。
    - 逻辑结构：线性结构；数据元素之间关系：线性关系。
    - 顺序存储结构
      - $LOC(a_i)=LOC(a_1)+(i-1)L$
      - llist结构
        - 包括
          - 一维数组e
          - 整型表长n
        - 代表：lt
    - 运算
      - 1°插入
      - 2°删除

- 8255 可编程并行接口
  - 1° 外部引脚
    - 1° 与I/O设备连接的引脚：端口ABC
    - 2° 与CPU连接的引脚
      - 连接数据总线：D_7~D_0 ↔ D_7~D_0：传送数据/控制/状态字
      - 连接地址总线
        - 8088(8 位)A_1A_0 ↔ 8255A_1A_0
        - 8086(16位)A_2A_1 ↔ 8255A_1A_0(低8位 → 偶地址)
      - 连接控制总线
  - 2° 内部结构
    - 1° 数据端口
      - A口：8b输入锁存器+8b输出锁存器/缓冲器；只有A口能成为双向口
      - B口：8b输入缓冲器+8b输出锁存器/缓冲器；不能设为双向口
      - C口：8b输入缓冲器+8b输出锁存器/缓冲器；1°高4b+低4b/2°每b均可置/复位3°可为状态口
    - 2° A/B组控制
      - 功能：接收CPU(读/写控制逻辑)控制字，确定端口工作方式和输入/出方向；A组：A口+C口高4位/A组：B口+C口低4位
    - 3° 数据总线缓冲器
      - 1°性质：双向三态8位
      - 2°功能：8255与系统数据总线接口；CPU数据/控制字 ↔ 数据总线缓冲器 ↔ 外设数据/状态字
    - 4° 读/写控制逻辑
      - $\overline{CS}$：CPU地址总线高位 → I/O接口译码 → $\overline{CS}$
      - RESET：8255控制寄存器清0，ABC口都置成输入方式(避免复位有输出)
      - $\overline{RD}$：8255(数据/状态字) → CPU($\overline{RD}$)
      - $\overline{WR}$：CPU($\overline{WR}$)(数据/控制字) → 8255
      - A_1A_0：端口选择。CPU地址总线低位 → A_1A_0 = 00 → A;01 → B;10 → C;11 → 控制字寄存器
  - 3° 工作方式控制字
  - 4° 初始化编程
  - 5° 应用
  - 6°

- 8255 可编程并行接口
  - 3° 工作方式
    - 1• 方式0
      - 基本IO方式：ABC口均可工作
      - 适用性：不需要用应答信号的简单输入输出场合
      - 组合数:(2^4)4种独立端口；输入信号不能锁存，输出可锁存
      - C口＝输入＋输出，用IN/OUT时先屏蔽掉不需要的部分
    - 2• 方式1
      - 带中断请求 / 选通IO方式
        - C口不可工作，主要配合AB口提供联络信号
        - 剩余位作为I/O线用
    - 3• 方式2