微机小抄　@知乎 jujubur
说明：一级标题 二级标题 三级标题 四级标题
一级文字 二级文字 三级文字 四级文字 五级文字 六级文字
|| 表示语句之间的分隔

第一章

1.一些细碎的知识点

1）地址引脚数为 16 条，可以寻址2^{16}=64KB 的存储单元。2^{10}=1K，2^{20}=1M，2^{30}=1G；b↔位，B↔字节

2）有关数制

补码为原码取反+1，最高位 1 表示负数，0 表示正数。例：[-89]补=10100111 十六进制数后面要加 H，否则默认十进制数。

3）微处理器、微型计算机、微型计算机系统的组成

微处理器（微型计算机中用 CPU 表示）：由一片或几片大规模集成电路组成的具有运算和控制器功能的中央处理器部件。微型计算机：以微处理器为核心，配上存储器（内存条）、输入输出接口电路（接口卡如显卡、声卡）、及系统总线（总线插槽）所组成的计算机（又称主机）微型计算机系统：以微型计算机（主机）为中心，配以相应外围设备（鼠标、键盘、显示器）、电源和辅助电路（统称硬件系统：硬件系统分为：1 主机主机板控制着整个系统中各部件之间的指令流和数据流，从而实现对微机系统的监控与管理 2 输入设备键盘、鼠标、扫描仪、摄像头等 3 输出设备显示器、打印机等）以及指挥微型计算机工作的软件系统（软件系统包括：1 系统软件一组控制计算机系统并管理其资源的程序，功能：启动计算机，存储、加载和执行应用程序，对文件进行排序、检索，将程序语言翻译成机器语言等，包括：操作系统，语言编译器，数据库管理系统，网络与通信系统等。2 应用软件解决各种实际问题而编制的专用程序或软件，包括：文档处理(Office)，社交软件(QQ，微信)，多媒体软件）所构成的系统

4）典型微机硬件系统结构

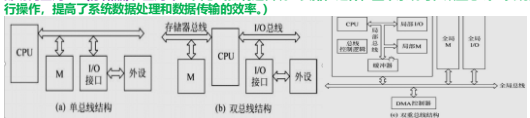
计算机的经典结构---冯·诺依曼结构（“存储程序式计算机”结构），该结构的特点：由运算器、控制器、存储器、输入设备和输出设备 5 个基本部分组成；程序和数据以二进制代码形式存放在存储器中，存放位置由地址指定，地址码也是二进制；控制器是根据存放在存储器中的指令序列（即程序）来工作的，并由一个程序计数器 PC（即指令地址计数器）控制指令的执行。控制器具有判断能力，能根据计算结果选择不同的动作流程。微处理器：微型计算机执行指令的核心部件，包含运算器和控制器。

存储器：存储信息的部件，存储当前正在使用的程序和数据。　I/O 设备和接口：微处理器和外部设备之间是通过 I/O 接口连接的。如 显示器的接口、硬盘的接口等。　系统总线：系统总线是连接微处理器和其他部件的通路，分为地址总线、数据总线和控制总线，分别用于传输地址、数据和控制信息。

5）总线

★CPU 通过总线读取指令，与内存、外设之间进行数据交换。在 CPU、内存和外设确定的情况下，总线速度是制约计算机整体性能的关键。从物理上来看，总线是一组传输公共信息的信号线的集合，即在计算机系统各部件之间传输地址、数据和控制信息的公共通道。在处理器内部的各功能部件之间，在处理器与高速缓冲器及内存之间、在处理器与外围设备之间等，都是通过总线连接的。

★根据总线结构组织方式的不同，可以将总线结构分为：单总线结构、双总线结构、双重总线结构（80X86 系统采用，局部总线：CPU 通过局部总线访问局部 M（存储器）和局部 I/O，全局总线：CPU 通过全局部访问全局 M（存储器）和全局 I/O 由总线控制逻辑统一安排，这样，整个系统可在双重总线上实现并行操作，提高了系统数据处理和数据传输的效率。）



系统总线的分类：控制总线：从处理器引出的若干信号线，CPU 通过它们与存储器或 IO 设备进行控制信息交换。CPU 向存储器或 IO 发送控制信号或时钟信号，存储器或 IO 向 CPU 发送状态信号、中断信号、总线请求信号等地址总线：传递地址信息的总线，CPU 输出将要访问的内存或 IO 地址，该总线为单向三态（三态：0,1，高阻，当其他主设备使用 AB 时，CPU 将其设为高阻）总线。数据总线：传递数据信息的总线，读：内存和 IO 到 CPU，写：CPU 到内存或 IO，双向三态总线。数据总线的宽度决定了 CPU 与其他器件进行一次数据传送时的最大位数

6）微处理器的组成：

以模型机为例，其组成：运算器（ALU):执行算术运算和逻辑操作。　控制器：指令译码，根据指令功能，产生一定时序的控制信号，组成包括: IR, ID, PLA 内部寄存器：用于存放指令、操作数和中间结果等。包括: 累加器 A、数据寄存器 DR、地址寄存器 AR、程序计数器 PC、标志寄存器 F、RA-寄存器阵列

算术逻辑部件（ALU）：执行算术和逻辑操作以及循环移位等。运算的两个操作数：1）I1:来自累加器 A 2）I2:来自内部数据总线：数据寄存器 DR(Data Register)中的内容，寄存器阵列 RA 中某个寄存器的内容，运算结果送回累加器 A 暂存。控制部件：(1）指令寄存器 IR(Instruction　Register) 存放从存储器取出的将要执行的指令。(2）指令译码器 ID(Instruction Decoder) 对指令寄存器 IR 中的指令进行译码，确定该指令应执行什么操作。(3）可编程逻辑阵列 PLA(Programmable Logic Array) 产生取指令和执行指令所需的各种微操作控制信号。内部寄存器：累加器 A：累加器用得最频繁。在进行算术逻辑运算时，具有双重功能：运算前，用来保存一个操作数；运算后，用来保存结果。数据寄存器 DR：数据寄存器 DR 用来暂存数据或指令。程序计数器 PC(Program Counter)：存放当前待取出指令在内存中的地址。通常，程序按顺序逐条执行，因此,PC 具有自动加 1 功能。地址寄存器 AR(Address Register)：　AR 用来存放正要取出的指令或操作数在内存中的地址。标志寄存器 F(Flag Register)：寄存执行指令时所产生的结果或状态的标志信号。寄存器阵列(register array, RA)也称为寄存器组(register stuff,RS)，它通常包括若干个通用寄存器和专用寄存器，不同的微处理器，其寄存器个数、功能不尽相同。

7）存储器概述

存储器：是微机中的存储和记忆部件，用来存放用二进制代码形式表示的数据和程序。字节（byte)：通常将 8 位二进制码作为一个字节。字（word)：通常将两个字节也就是 16 位称为一个字。

字长:表示计算机数据总线上一次能处理的信息的位数,即位长,由此定义计算机是多少位的,如 4 位机、8 位机、16 位机、32 位机、64 位机等。

★存储器的组成：存储体、地址译码器和控制电路组成。存储体存储数据和程序地址译码器接收从地址总线 AB 送来的地址码，经译码器译码选中相应的某个存储单元，以便从中读出（取出）信息或写入（存入）信息控制电路接收来自 CPU 的信号，控制存储器的读 / 写操作过程

8）输入输出接口概述：输入/输出接口(I/O 接口)：　是 CPU 与外设间的桥梁，完成信号转换、数据缓冲与 CPU 进行信号联络等工作

★接口的功能 1.提供驱动外设的电压或电流；　2.匹配 CPU 与外设之间的信号电平、信号类型、数据格式等；3.缓存发给外设的数据、控制命令和外设提供的运行状态信息。

9）微机的工作原理

工作原理（存储程序+程序控制)：计算机内采用二进制形式表示计算机中的指令和数据。程序和原始数据预先存入计算机主存储器中保存，执行程序时，控制器可以连续、自动、高速的从存储器中逐一取出指令并执行。微机工作过程：不断地取指令、执行指令的过程。

9）微机的执行过程

指令组成（操作码+操作数)

★由于 1 个存储单元只能存放 1 个字节，而指令根据其所含内容不同而有单字节、双字节、3 字节乃至最多 6 字节之分，因此在执行 1 条指令时,就可能要处理 1～6 个不等字节数目的代码信息，包括操作码、操作数或操作数的地址。

★取指+执行：1.假定程序已存放到内存中，计算机进入运行状态时，首先把第 1 条指令所在的地址赋给程序计数器 PC，然后微机进入取指阶段，2.在取指阶段，CPU 从内存中读出的内容比为指令，于是数据寄存器 DR 便把它送至指令寄存器 IR。3.指令译码器译码，控制器就发出相应的控制信号，CPU 根据控制信号执行相应操作。4.取指阶段结束后，微机就进入执行指令阶段，CPU 执行指令所规定的的具体操作。5.当一条指令执行完毕以后，就转入了下一条指令的取指阶段，这样周而复始地循环一直执行到程序中遇到暂停指令时方才结束。

第二章

1.带符号数的补码加减运算

★补码：X>0 时，[X]补=[X]原 X<0,[X]补=2^n+X

★对于二进制负数，求补码的方法是符号位保持不变，其他各位取反再加 1，正数补码和原码相同

★补码加减规则：[X1+X2]补=[X1]补+[X2]补　[X1-X2]补=[X1]补+[-X2]补

例：　X_1 =-0011，X_2 = 1011求$[X_1+X_2]_{补}$和 $[X_1-X_2]_{补}$

解：　$[X_1]_{补}$=11101，　$[X_2]_{补}$=01011, $[-X_2]_{补}$=10101

$[X_1+X_2]_{补}$=11101+01011= 01000

真值为：X_1+X_2=1000

```
  1 1 1 0 1
+ 0 1 0 1 1
-----------
1 0 1 0 0 0
↑ 丢弃
```

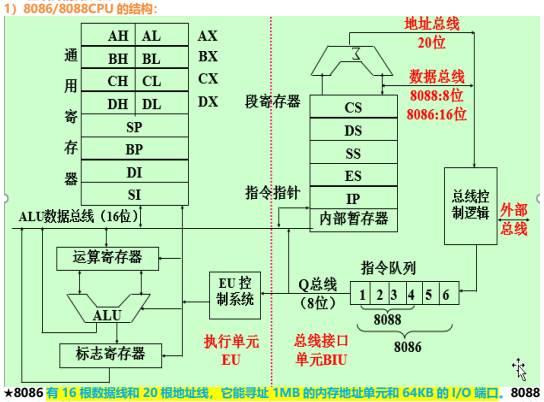
2.溢出的判断

D7 和 D6 进位情况不同则溢出。D7 的进位情况⊕（异或，不同为 1）D6 的进位情况=1 溢出，D7 的进位情况⊕D6 的进位情况=0 不溢出

第三章

1.一些细碎的知识点

1）8086/8088CPU 的结构：



★8086 有 16 根数据线和 20 根地址线，它能寻址 1MB 的内存地址单元和 64KB 的 I/O 端口。8088 有 20 根地址线，内部数据位宽为 16 位，但外部数据线只有 8 根。设计 8088 是为了和原有的 8 位外围接口芯片兼容。8086/8088 访问 IO 端口时，不用高 4 位地址线，所以能寻址2^{16}=64KB 的 IO 端口。

★8086/8088 分为总线接口单元（BIU）和执行单元（EU）两部分。指令队列、段寄存器、二十位物理地址的形成在总线接口单元中，指令译码、标志寄存器、存储器操作数偏移地址的计算在执行单元中。

★BIU：总线接口单元，根据执行单元的请求，负责 CPU 与存储器或 I/O 端口之间的数据传送 EU：执行单元，负责执行指令。BIU 和 EU 可以同时并行工作，提高了总线的传输效率和系统的执行速度。

2）BIU

①功能：

根据 EU 的命令完成总线操作：从存储器读取指令、送到指令队列，从存储器或 IO 接口读写数据。

②组成：

总线控制电路、指令队列、段寄存器（CS 代码段\DS 数据段\ES 附加数据段\SS 堆栈段）、指令指针 IP、地址加法器（由地址加法器将段寄存器（存放一个段在内存中起始地址（20 位）的高 16 位）的值和偏移地址（由 IP 提供或由 EU 按寻址方式计算出寻址单元的 16 位偏移地址）计算得到 20 位物理地址）

③操作过程（以存储器访问为例）：

1.首先通过地址加法器生成将要访问的存储器单元的物理地址；2.再从(往)物理地址对应的存储器单元读取(或写入)数据；3.如果是读取指令，BIU 从物理地址对应的存储器单元取到指令后将指令送入指令队列。

④指令队列

1.先入先出；　2.存满一条指令后就立即执行；　3.只要空出 2 个字节(8086)或 1 个字节(8088)，BIU 便自动执行取指操作，直到填满为止；　4.EU 执行完转移、调用和返回指令时，需要清空指令队列，并要求 BIU 从新地址取指令。

⑤分段和段寄存器

8086 有 20 根地址线，但内部寄存器只有 16 位，不能直接提供对 20 位地址的寻址信息。

如何实现对 20 位地址的寻址呢？　——采用“段加偏移”的技术将 1M(2^{20})字节的存储空间分段来管理，每个段的最大长度 64K(2^{16})字节。4 个段寄存器对应 4 种段：CS---代码段 DS---数据段 ES---附加数据段 SS---堆栈段

⑥地址加法器：

段寄存器存放一个段在内存中起始地址(20 位)的高 16 位;IP 提供 或 由 EU 按寻址方式计算出寻址单元的 16 位偏移地址（单元距离段首的地址偏移量），由地址加法器将段寄存器的值和偏移地址计算得到 20 位的物理地址（又称为实际地址）

例：假设 CS=5600H，IP=0020H，请计算指令的物理地址 PA。解：段寄存器值左移 4 位 = 段寄存器值*16，PA = CS*16+IP= 5600H*16+0020H= 56000H+0020H = 56020H

3）EU

①功能

负责指令的译码、执行并回写结果。此外还具有管理寄存器等功能。

②工作过程

EU 从 BIU 的指令队列里得到指令后，完成指令的译码、执行；如果执行指令时还需要从存储器或者外设存取操作数，便申请 BIU 访问存储器或者外设，并向 BIU 提供操作数的偏移地址。

③EU 的组成：

通用寄存器（AX\BX\CX\DX）、专用寄存器（SP 堆栈指针寄存器\BP 基址指针寄存器\SI 源变址寄存器\DI 目标变址寄存器）标志寄存器（16bit）、算术逻辑单元 ALU、控制电路

4）总线周期：

①周期

时钟周期：CPU 的时间基准，是其处理定时的最小单位，由主频决定。假设 8086 的主频为 5MHz，则一个时钟周期为 200ns 总线周期：CPU 通过总线进行一次操作所需的时间,即 CPU 通过总线接口单元(BIU)对存储器或 I/O 端口进行一次读/写操作所需的时间。指令周期：取出一条指令并执行这条指令所需的时间，是从取指令、指令译码到执行完所需的全部时间。通常 1 个指令周期包含若干个总线周期，而 1 个总线周期又包含若干个时钟周期。

②8086/8088 总线周期

在 8086/8088 中，一个基本的总线周期由 4 个时钟周期组成：T1、T2、T3、T4;如果内存或 IO 接口速度较慢，跟不上 CPU 的读/写速度，则需在 T3 和 T4 之间插入等待状态(Tw)。T3 时检测外部存储器或 I/O 接口送来的 READY 信号:低电平,插入 Tw; 高电平,进入 T4。每个 TW 时也要检测 READY 信号: 低电平,继续插入 Tw;高电平,进入 T4

③空闲周期：

只有 BIU 与内存或 I/O 端口交换数据，或者填充指令队列时，BIU 才执行总线周期。如果 CPU 既不需要填充指令队列，EU 也没有向 BIU 发出总线周期请求时，就进入空闲周期。空闲周期由一个或几个 TI 状态组成。空闲周期时,EU 可能仍在执行指令,做运算处理。

5）8086 的引脚信号和功能模式

①引脚信号

8086/8088 芯片共 40 个引脚，8086/8088 的信号线有多少？16 根数据线+20 根地址线+若干控制线+电源线+地线引脚不够用，怎么办？复用---同一引脚在不同的情况下做不同功能来用如何复用？分时复用，不同模式复用，不同方向复用

★8086/8088 的引脚信号可分为：地址/数据信号，地址/状态信号，控制信号和电源信号。

★地址/数据分时复用 AD0~AD15 地址/状态分时复用 A16/S3 ~ A19/S6

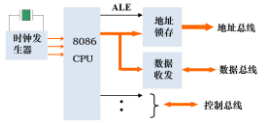
T1 地址，T3 数据

②功能模式

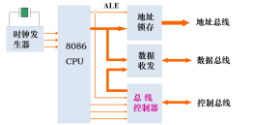
8086 有最大（多处理器模式，MN\MX 引脚输入低电平）和最小（单处理器模式，MN\MX 引脚输入高电平）两种工作方式，如果系统中除了 8086 还有 8087，则 8086 应工作于最大方式。最小模式下，控制信号全部由 8086 提供。

最小模式：控制信号较少，直接由 CPU 产生，地址线与数据线、状态线分时复用，需要地址锁存器将地址锁存下来，数据线连至内存及外设，负载重，需用数据收发器来驱动最大模式：控制信号较多，需要总线控制器和 CPU 一起输出更多控制信号

最小模式下的总线连接示意图



最大模式下的总线连接示意图



③8086 最小模式下的主要引脚信号

地址信号、数据信号和状态信号：AD_0—AD_{15} 低 16 位地址和低 16 位数据信号分时复用。传送地址信号时,为单向输出；传送数据信号时,为双向,分时复用。A_{16}/S_3 — A_{19}/S_6 高 4 位地址信号和状态信号分时复用。主要控制信号：RESET：复位信号 READY：准备好信号，外部输入 RD：读信号；WR：写信号；M/IO：为“低(0)”表示访问 I/O 接口，　为“高(1)”表示访问内存；ALE：地址锁存信号；DT/R：数据收发器的传送方向控制，1:发，0: 收；DEN：数据使能信号，低电平有效时，允许进行读/写操作；INTR、/INTA：外部可屏蔽中断请求/响应信号，高电平有效 NMI：外部不可屏蔽中断请求信号，上升沿有效 HOLD、/HDLA：　总线请求/响应信号，DMA 传送时会用到/BHE：总线高位使能---高 8 位数据总线允许，　8086 读写高 8 位数据时用到

★控制总线组合与四种基本的总线周期：

总线周期 \ 引脚	M/$\overline{IO}$	$\overline{WR}$	$\overline{RD}$
读存储器	高	高	低
写存储器	高	低	高
读I/O端口	低	高	低
写I/O端口	低	低	高

6）存储器组织

①存储空间

8086/8088 20 位地址→2^{20}=1MB 存储空间(00000~FFFFFH)，1MB 存储空间按字节组织，一个字节 → 一个地址单元

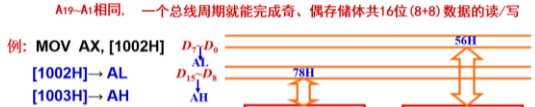
②数据存/取

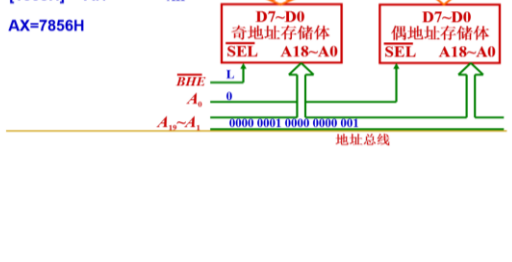
③存储器的连接

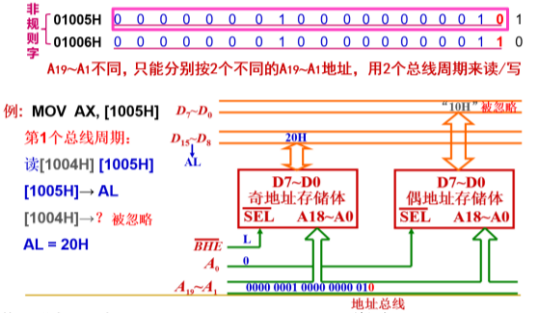
★8086 1MB 存储空间实际上分为 2 个 512KB 的存储体。偶地址存储体与低 8 位数据线(D7~D0)相连；奇地址存储体与高 8 位数据线(D15~D8)相连。

★读写偶地址字节：A0=0,/BHE=1，数据→D7~D0，一个总线周期完成读写。读写奇地址字节：A0=1,/BHE=0，数据→D15~D8，一个总线周期内完成读写。读写规则字：A0=0,/BHE=0,数据低 8 位→D7~D0，数据高 8 位→D15~D8，一个周期内完成读写。读写非规则字：需要两个总线周期。

★如果要在一个总线周期内读写 1 个字 (16 位) 数据，那么寄偶地址存储体就要能同时进行数据读写，即：奇偶地址存储体的片选/SEL 都要有效，从偶地址开始读，A0=0，读写一个字（含高、低 8 位），总线高位使能/BHL=L，读写数据时，奇偶地址存储体地址引脚上的地址 A19~A1 要相同







第 2 个总线周期：读[1006H][1007H],[1006H]→AH，[1007H]被忽略，AX=3020H

例题：假设下图是 8086 系统中存储器的部分单元，执行下列指令读取存储器中的数据，DS=0，请根据题目要求回答问题填空。

存储器

08010H	12
08011H	34
08012H	56
08013H	78
08014H	10
08015H	20
08016H	30
08017H	40

1) MOV AH, [8012H]
AH = 56H ，从存储器读取这个数据需要 1 个总线周期。
2) MOV AX, [8012H]
AX = 7856H ，从存储器读取这个数据需要 1 个总线周期。
3) MOV CL, [8015H]
CL = 20H ，从存储器读取这个数据需要 1 个总线周期。
4) MOV DX, [8015H]
DX = 3020H ，从存储器读取这个数据需要 2 个总线周期。

7）存储器的分段

8086 /8088 是 16 位微处理器，可以直接生成 16 位地址码，产生 2^{16} 个地址编码，只能寻址 64KB 的存储范围，　8086 /8088 有 20 位地址，1MB 内存，需要生成 20 位地址码，产生 2^{20} 个地址编码，寻址 1MB 的存储范围

①分段方法：

1MB 内存为若干逻辑段，段的长度：1B ~ 2^{16}(64K) B 段起始地址—由段寄存器设定，段寄存器值*16(左移 4 位)

②逻辑地址和物理地址

内存每个单元的地址在逻辑上都由两部分组成：段基址（段寄存器的值）16 位，指示存储单元所在的段在内存所处的区域，段内地址（偏移地址）　16 位，存储单元在段中的相对位置（与段中第 1 个单元的距离）例: 若 CS=2000H, IP=0600H，取指令的逻辑地址=2000H: 0600H; 例: 若 DS=3000H，MOV AL, [0800H],取数据的逻辑地址=3000H: 0800H

★物理地址：物理地址 = 段基址(段寄存器值)×16 + 偏移地址

★段加偏移寻址机制的好处：可以在偏移地址不改变的情况下，将段移动到存储器的任何区域；可以实现程序的重定位。即操作系统可根据系统当前内存情况，将同一程序（程序不变，指令、数据的移地址不变）加载到不同的地址。

8）堆栈

1.堆栈的特点（1）内存中一个特殊区域，用于存放暂时不用或需要保护的数据。常用于响应中断或子程序调用。(2）堆栈采用‘先进后出，后进先出’的存取方式。(3）栈顶是堆栈操作的唯一出口，堆栈指针 SP 指向栈顶。(4）堆栈有专门的数据传输指令：入栈指令‘PUSH’，出栈指令“POP”(5）堆栈指针 SP：执行 PUSH 指令时，SP←SP-2，执行 POP 指令，　SP←SP+2

例：已知 SS=1000H, SP=0100H 则堆栈段的起始地址=10000H 栈顶偏移地址(SP)=0100H。假设堆栈段最后一个单元(栈底)地址为 10200H，则：栈底偏移地址 =0200H

★堆栈的大小（长度）由 SP 的初值决定。

2.压栈：　寄存器的低字节→堆栈低地址，寄存器高字节→堆栈高地址

3.出栈：　堆栈低地址→寄存器的低字节，堆栈高地址→寄存器高字节

4.PUSH 指令，　堆栈对应的单元是目标操作数，POP 指令，堆栈对应的单元是源操作数

9）指令的分类

（一）无操作数指令如：NOP, HLT （二）单操作数指令 1.单操作数在寄存器中。如：NEG AX，2.单操作数在存储器中。如：INC BYTE PTR[1000H] （三）双操作数指令双操作数：源操作数、目标操作数，如:MOV BX，　AX ;右边 AX---源操作数，左边 BX---目标操作数 1.两个操作数都在寄存器中。如：MOV AX，　BX || ADD AX，　BX。2.两个操作数中一个在寄存器中，另一个是立即数：如：MOV AX, 1000H3.两个操作数中一个在存储器中，另一个是立即数。如：MOV WORD PTR [1000H], 1234H,DS:[1000H] ← 34H，　DS:[1001H] ← 12H。4.两个操作数中一个在寄存器中，另一个在存储器中；注：此种情况，存储器操作数的类型由寄存器操作数的类型来确定，如果寄存器操作数是字节型，则存储器操作数也是字节型；如果寄存器操作数是字类型，则存储器操作数也是字类型。如：MOV AL, [1000H] 存储器操作数是字节类型,在内存占 1 个字节单元，AL←DS:[1000H]。　MOV AX, [1000H]，存储器操作数是字类型,在内存占 2 个字节单元，AL←DS:[1000H]，AH←DS:[1001H]

10）寻址方式（看清楚问的是源操作数还是目标操作数的寻址方式）

如果是字操作数，要写两个单元的地址

①固定寻址，如 AAA

②立即数寻址，操作数直接包含在指令中

举例 MOV AL,15H||MOV AX,1234H

③寄存器寻址

对 16 位操作数寄存器可以为 8 个 16 位通用寄存器（AX\BX\CX\DX\SI\DI\SP\BP），对 8 位操作数寄存器只能为 AH\AL\BH\BL\CH\CL\DH\DL

举例：　MOV AX,SS || MOV SP,BP || MOV AX,BX

④存储器寻址

当执行单元 EU 需要读/写位于存储器的操作数时，应根据指令给出的寻址方式，由 EU 先计算出操作数的有效地址 EA(偏移地址)，并将它送给 BIU；同时请求 BIU 执行一个总线周期，BIU 将某个段寄存器的内容左移 4 位，加上 EU 送来的有效地址 EA 形成 20 位的物理地址，然后执行总线周期，读/写指令所需的操作数。有效地址 EA(偏移地址)的值要根据指令所采用的寻址方式计算得出。计算 EA 的通式为：

$$EA=\text{基址值}\left[\begin{matrix}BX\\BP\end{matrix}\right]+\text{变址值}\left[\begin{matrix}SI\\DI\end{matrix}\right]+\text{位移量}\left[\begin{matrix}0\text{位}\\8\text{位}\\16\text{位}\end{matrix}\right]$$

1）直接寻址：操作数的 EA 由指令直接给出。默认数据段 DS，其它数据段应在指令中用段前缀指出。这种寻址方式的指令执行速度较快，主要用于存取位于存储器中的简单变量。例: MOV AX, [1234H]；EA=1234H, 假设 DS=2000H, 则 PA(物理地址)=2000H+1234H = 3234H, 假设 [3234H]=11H, [3235H]=22H，则指令执行后，AL=11H，AH=22H

2）间接寻址：操作数在存储器中，存储单元的有效地址由寄存器指出。BX、SI、DI — 默认数据段 DS|| BP—默认堆栈段 SS，注意：间接寻址的地址寄存器只能是 BX、BP、SI、DI，不能是其它寄存器。指令中地址寄存器要加方括号，如：[BX]

根据所采用的地址寄存器的不同，间接寻址方式又可分为以下 3 种：

（1）基址寻址方式（BX,BP）操作数的有效地址由基址寄存器(BX 或 BP)的内容和指令中给出的地址位移量(0 位或 8 位或 16 位)之和来确定。EA = BX/BP+ 0 位/8 位/16 位移量

例: MOV AX，[BX]，假设 BX = 1122H，DS = 3000H， 则 PA = 30000H + 1122H = 31122H，30000H + 1123H = 31123H 假设 [31122H] = 34H，[31123H] = 56H，则指令执行后，AX = 5634H

例: 假设 BETA=8，DS=6000H， BX=5000H 则 MOV AL，[BX+8] || MOV AL，8[BX]; "8" 是偏移地址位移量，不是乘的倍数 || MOV AL，[BX+BETA] || MOV AL，BETA [BX] 注：上面 4 条指令是等价的。EA = 5000H+8= 5008H，PA = DS*16+5008H= 60000H+5008H= 65008H。假设 [65008H]=68H，执行后，AL=68H

（2）变址寻址方式（SI,DI）

变址寻址是指操作数的有效地址由变址寄存器(SI 或 DI)的内容与指令中给出的地址位移量(0 位、8 位或 16 位)之和来确定。EA （偏移地址）= SI/DI +0 位/8 位/16 位移量

例: MOV BETA[DI]，AX || MOV BX，[SI+BETA]

（3）基址加变址寻址方式（ BX/BP, SI/DI ）操作数的有效地址 EA 为以下三部分之和：基址寄存器(BX 或 BP)的值、变址寄存器(SI 或 DI)的值、指令中的地址位移量(0 位、8 位或 16 位)。EA=[基址 BX/BP]+[变址 SI/DI]+ 0/8/16 位偏移量。

例: MOV BX，[BX+SI] || MOV [BX][DI]，AX || MOV AX，8[BX][SI] || MOV AX，8[BX+SI] || MOV AX，[BX+SI+8] || MOV AX，ES:[BX+SI+10H] "ES:" ---段前缀，指定操作数在附加段，源操作数 PA = ES*16+ BX+SI+10H ，ES*16+ BX+SI+10H+1 || MOV AX，[BP+SI+20H] BP---操作数在堆栈段，源操作数 PA = SS*16+ BP+SI+20H ，SS*16+ BP+SI+20H+1

★存储器寻址小结：地址寄存器只能是 BX、BP、SI、DI，不能用其他寄存器。用 BX、SI、DI 时，默认操作数在数据段(DS)。用 BP 时，默认操作数在堆栈段(SS)。可用段前缀指定操作数在哪个段，如 "ES:" ---附加段。基址+变址只有四种组合：BX+SI, BX+DI, BP+SI, BP+DI，可以再加位移量。不能同时用两个基址寄存器，也不能同时用两个变址寄存器。

错误示例: MOV AX，[CX] || MOV BX，[AX] || MOV BX，[SP] || MOV AX，[SI+DI] || MOV DX，[BX+BP] || MOV DX，[SP+BX]

⑤段内寻址方式

1）串操作指令寻址方式：数据串(或字符串)指令中的操作数不使用正常的存储器寻址方式，而是采用隐含寻址方式。隐含规定：源串在数据段，操作数有效地址存放在源变址寄存器 SI(不能用其他地址寄存器)中；目标串在附加段，操作数有效地址存放在目标变址寄存器 DI(不能用其他地址寄存器)中。串操作指令中不出现 SI 或 DI。在重复串操作时,8086/8088 能自动修改 SI 和 DI 的内容，以便它们能指向后面的字节或字。

如：串操作指令： MOVSB, CMPSB, SCASB, LODSB, STOSB：隐含规定：源串 DS:［SI］; 目的串 ES:［DI］。

2）I/O 端口寻址 a. 直接端口寻址（8 位端口地址，0 到 255）MOV AL，20H || OUT 20H，AL || IN AL，21H b. 间接端口寻址（16 位端口地址，0 到 65535）MOV DX，3F8H || OUT DX，AL

3）转移类指令的寻址方式在 8086/8088 系统中,由于存储器采用分段结构，所以转移类指令有段内转移和段间转移之分。共有 5 种寻址方式：段内短转移、段内直接转移、段内间接转移、段间直接转移、段间间接转移。

1）"段加偏移"寻址机制的特性（好处）：--允许重定位

1 存储器采用分段管理后，段起始地址高 16 位被称为段地址，把它保存在 8086 内部的 16 位寄存器 CS、DS、SS、ES 中。2 段内的相对地址可用系统中的 16 位通用寄存器来存放，被称为偏移地址。3 实际上可用提据编程的需要来确定段的大小，它可以是 64K 字节范围内的任意多个字节。4 各段间可以连续、分开或者重叠。重定位（重定位就是把程序的逻辑地址空间变换成内存中的实际物理地址空间的过程），移植性强

3.标志位判断

8086 标志位有 9 位（注意指令对标志位是否有影响），见第四点的表格，分为 6 位状态标志和 3 位控制标志（TF\IF\DF）

CF: 进位标志(D0) CF 置 1，表示加法/减法运算时产生了进位/借位。此外，循环指令也会影响这一标志。OF: 溢出标志（D11 位）在有符号数进行加法或减法运算过程中产生溢出时，OF 被置为 1，指示运算结果超出了机器能够表示的范围。对于无符号数 的 操作，不考虑该标志。AF: 辅助进位标志(D4) 当加/减法运算时 D3 位向 D4 位有进位/借位，则 AF 置 1。BCD 码运算调整指令会判断 AF 。注意：不管是 8 位操作数，还是 16 位操作数运算，AF 都是判 D3 位向 D4 位是否有进位/借位。ZF: 零标志（D6 位） 表示算术或逻辑运算的结果是否为零。若当前运算结果为零，则 ZF 为 1；若当前运算结果不为零，则 ZF 为 0。SF: 符号标志（D7 位）表示算术或逻辑运算结果的算术符号，它和运算结果的最高位相同。当数据用补码表示时，负数的最高位为 1，所以符号标志指出了运算结果的正负。PF: 奇/偶标志(D2)如果运算结果的低 8 位中所含 1 的个数为偶数，则 PF 置 1，1 的个数为奇数时 PF 为 0。TF: 跟踪标志或称单步标志（D8 位）是为方便调试程序而设置的。如果 TF 置为 1，CPU 按照单步方式执行指令。IF: 中断标志（D9 位），对可屏蔽中断的控制标志。IF 为 1，表示 CPU 允许可屏蔽中断；IF 为 0，表示 CPU 不允许可屏蔽中断。指令： STI (IF 置 1)，CLI (IF 清 0) DF: 方向标志（D10 位）。控制串操作指令执行过程中地址的增量或减量。DF 为 0，在串操作过程中，地址会不断递增；DF 为 1，在串操作过程中，地址将不断递减。指令： STD（DF 置 1），CLD (DF 清 0）判断溢出：D7 的进位情况⊕（异或，不同为 1）D6 的进位情况=1 溢出，D7 的进位情况⊕D6 的进位情况=0 不溢出

4.指令对标志位影响（OF\SF\ZF\AF\PF\CF）

类 别	指 令 名 称	指令书写格式（助记符）	状态标志位 O	S	Z	A	P	C
加法	加法(字节/字)	ADD d,s	↑	↑	↑	↑	↑	↑
	带进位加法(字节/字)	ADC d,s	↑	↑	↑	↑	↑	↑
	加 1(字节/字)	INC d	↑	↑	↑	↑	↑	·
减法	减法(字节/字)	SUB d,s	↑	↑	↑	↑	↑	↑
	带借位减法(字节/字)	SBB d,s	↑	↑	↑	↑	↑	↑
	减 1(字节/字)	DEC d	↑	↑	↑	↑	↑	·
	取负	NEC d	↑	↑	↑	↑	↑	1
	比较	CMP d,s	↑	↑	↑	↑	↑	↑
乘法	不带符号乘法(字节/字)	MUL s	↑	×	×	×	×	↑
	带符号整数乘法(字节/字)	IMUL s	↑	×	×	×	×	↑
除法	不带符号除法(字节/字)	DIV s	×	×	×	×	×	×
	带符号整数除法(字节/字)	IDIV s	×	×	×	×	×	×
	字节转换成字	CBW	·	·	·	·	·	·
	字转换成双字	CWD	·	·	·	·	·	·
十进制调整	加法的 ASCII 码调整	AAA	×	×	×	↑	×	↑
	加法的十进制调整	DAA	×	↑	↑	↑	↑	↑
	减法的 ASCII 码调整	AAS	×	×	×	↑	×	↑
	减法的十进制调整	DAS	↑	↑	↑	↑	↑	↑
	乘法的 ASCII 码调整	AAM	×	↑	↑	×	↑	×
	除法的 ASCII 码调整	AAD	×	↑	↑	·	↑	×

类 别	指 令 名 称	指令书写格式（助记符）	状态标志位 O	S	Z	A	P	C
逻辑运算	"与"(字节/字)	AND d,s	0	↑	↑	×	↑	0
	"或"(字节/字)	OR d,s	0	↑	↑	×	↑	0
	"异或"(字节/字)	XOR d,s	0	↑	↑	×	↑	0
	"非"(字节/字)	NOT d	·	·	·	·	·	·
	测试(字节/字)	TEST d,s	0	↑	↑	×	↑	×
移位	算术左移(字节/字)	SAL d,count	↑	↑	↑	×	↑	↑
	算术右移(字节/字)	SAR d,count	↑	↑	↑	×	↑	↑
	逻辑左移(字节/字)	SHL d,count	↑	↑	↑	×	↑	↑
	逻辑右移(字节/字)	SHR d,count	↑	↑	↑	×	↑	↑
循环	循环左移(字节/字)	ROL d,count	↑	·	·	·	×	↑
	循环右移(字节/字)	ROR d,count	↑	·	·	·	×	↑
	带进位循环左移(字节/字)	RCL d,count	↑	·	·	·	×	↑
	带进位循环右移(字节/字)	RCR d,count	↑	·	·	·	×	↑

5.指令含义及其注意事项汇总（包括伪指令、各种运算符等相关内容）

A

ADD 目，源。目标操作数加源操作数，结果保留在目标操作数中。

目标操作数不允许是立即数。源和目标操作数不能同时为存储器操作数。影响状态标志位。

ADC 目，源。目标操作数加源操作数，并加 CF，结果保留在目标操作数中。

AAA：加法的 ASCII 调整

把 AL 中的和调整为非压缩的 BCD 格式，AH 加上调整产生的进位值（AF）

AAS：减法的 ASCII 调整

把 AL 中的差调整为非压缩的 BCD 格式，AH 减调整产生的借位值（AF）

AAM：乘法的 ASCII 调整

把 AL 中的积调整为两位非压缩的 BCD 码，高位在 AH，低位在 AL。AAM 实际上执行如下操作：将 AL 除以 10，商放到 AH，余数放到 AL。★8086 不允许采用压缩型十进制数乘法运算，如果是两个 ASCII 码数相乘，要先将其转换成非压缩型 BCD 码。用法：先执行非压缩 BCD 码乘法，然后执行 AAM。

AAD：除法的 ASCII 调整

将 AX 中的被除数（非压缩 BCD 码）转化为二进制数，放在 AL。实际执行操作：AX 中的被除数是两位非压缩 BCD 码，AH 乘 10，再加 AL，即转换为二进制数。

★AAD 用在 DIV 指令之前，先将 AX 中的被除数调整成二进制数，并存放在 AL 中，再用 DIV 指令作二进制数的除法。

AND 逻辑运算指令，与

ASSUME 伪指令

ASSUME 伪指令一般出现在代码段中，它用来建立段和段寄存器之间的关系。其格式为：ASSUME 段寄存器：段名[，段寄存器名：段名]例: ASSUME CS: CODE, DS: DATA , SS: STACK

C

CLD：将方向标志 DF 清零

CMP 目标，源。目标操作数减去源操作数，但不送回结果，只根据结果置标志位。

寻址方式的使用类似于 MOV。CMP 影响所有状态标志位。

CBW：有符号数的符号位扩展指令

当 AL 的最高有效位为 0 时，AH=0，当 AL 最高有效位为 1 时，AH=FFH。

CWD：有符号数的符号位扩展指令（这两个指令主要为了配合 IDIV 指令）

当 AX 的最高有效位为 0 时，DX=0；当 AX 的最高有效位为 1 时，DX=FFFFH

CMPSB\CMPSW（串比较指令）

用串源中 DS：SI 所指的串元素值减去目标串中 ES:DI 所指的串元素值。结果反应在标志位上，自动修改 SI、DI

需要用到重复前缀：REPZ\REPE。串操作指令执行后 CX=CX-1，若 CX≠0 且结果相等（ZF=1），则继续执行该串操作，否则顺序执行下一条指令。格式：REPE CMPSB

CALL 目标操作数

子程序调用指令，将当前的 IP 或 CS：IP 压栈，根据目标操作数修改 IP 或 CS：IP，完成转移。调用近过程时，仅将 IP 压栈，调用远过程时，将 CS：IP 压栈。

D

DAA：加法的十进制调整

把 AL 中的和调整为压缩的 BCD 格式，并将结果放在 AL。DAA 和 DAS 指令只能对 AL 中结果调整，因此多字节压缩 BCD 的加减法，只能从低字节开始，逐个运算。

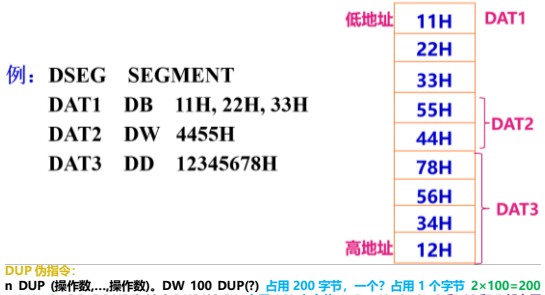
DAS：减法的十进制调整

把 AL 中的差调整为压缩的 BCD 格式，并将结果放在 AL。

DB\DW\DD\DQ\DT 分别为字节\字\双字\四字\五字。

DATA1 DW 12H，2+3，？。区分 DATA4 DB 'AB' 与 DATA5 DW 'AB' 前者把 A 存在低地址，后者把 A 存在高地址

★空格也占一个字节 THREE DB 'Please input password;' 占 22 个字节



DUP 伪指令：

n DUP (操作数,...,操作数)。DW 100 DUP(?) 占用 200 字节，一个？占用 1 个字节 2×100=200（DW×2）DD 5 DUP(?,12,3 DUP(12,5)) 占用 160 个字节 4×5×（2+3×2）？和 12 和 5 都占用 1 个字节。TWO DW TWO 占用 2 个字节

DIV 源：（无符号除法）

字节除法：AX/源的商→AL，字除法：(DX,AX)/源的商→AX，余数→DX

★若除数为 0 或 AL 中商大于 FFH（或 AX 中商大于 FFFFH），则 CPU 产生一个类型为 0 的内部中断。

★src 不能为立即数，对所有状态标志位均无定义。

E

END（结束指令）

整个源程序必须以 END 语句来结束，通知汇编程序停止汇编。END 后面的标号表示该程序执行时起始地址（详见 SEGMENT）

EQU：赋值伪指令

格式：名字 EQU 表达式 名字为任何有效的标识符，表达式为任何有效形式的操作数

=：它与 EQU 基本相似，起赋值作用。例: X=12

例 1: STRING DB "THIS IS MY COMPUTER!" // N EQU $-STRING 表示将字符串长度赋给 N（$表示当前存储单元地址）

例 2: FIRSTW EQU WORD PTR BYTES 将变量 BYTES 重新定义为字类型属性并赋予新变量名

例 3: A EQU ARRAY[BX][SI]基址加变址引用赋予符号名 A

例 4: COUNT EQU CX 为寄存器 CX 定义新的符号名 COUNT LD EQU MOV 为指令助记符 MOV 定义新的符号名 LD

★"EQU" 与 "=" 区别：1）使用 "＝" 定义的符号名可以被重新定义，使符号名具有新值。例:X=12，X=20 ；X 这个符号代表最新的值 20。2）习惯上 "＝" 主要用来定义符号常量。

I

IN 和 OUT

IN 累加器,端口号；OUT 端口号，累加器 (顺序不要弄反)。端口号可以用 8 位立即数直接给出，只限于在 AL 或 AX 与 I/O 端口之间传送信息，当端口号大于 255 时，必须使用间接端口寻址（端口号必须放在 DX 中）。

INT n （其中 n=0-255） 中断指令

IRET：中断返回

INTO：溢出中断，OF=1，则执行类型 4 的中断，OF=0 不做任何操作。

INC 目和 DEC 目，目标操作数加\减 1，结果保留在目标操作数中。

为针对无符号数的运算。对于间接寻址，必须用伪指令指定存储单元类型。INC WORD PTR [SI] 而 DEC SI 不用。INC 指令不影响 CF，影响其他 5 位状态标志位。

IMUL（有符号乘法），其他同 MUL

IDIV（有符号除法）

IDIV 的操作数、商、余数都要为有符号数，而且余数的符号与被除数的符号相同。

J

JZ（JUMP IF ZERO）即零标志位 ZF=1 就跳转。

JMP 无条件跳转指令。无条件跳转指令可转到内存中任何程序段

目标操作数可以为立即数、寄存器、存储器。

JLE\JNG（jump if less or equal,or not greater）

小于或等于，或者不大于则转移。

L

LEA（目标地址传送指令）

LEA reg src：取源操作数（源操作数必须为存储器操作数）的偏移地址送至目标操作数（一个指定的 16 位通用寄存器）

例: LEA BX,[SI]将 SI 指示的偏移地址（即 SI 的内容）装入 BX

例: MOV BX,[SI]将 SI 寻址的存储单元的数据装入 BX

LDS reg，src：指针送寄存器和 DS，[src]→reg，[src+2]→DS

例 LDS SI,[2480]执行后如图所示。

LES reg，src：指针送寄存器和 ES，[src]→reg，[src+2]→ES

LAHF：加载标志寄存器的低 8 位到 AH 寄存器

以上三个指令的目标操作数必须是寄存器。

LODSB\LODSW（串装入指令）

将源串中 DS：SI 所指的串元素复制到 AL（字节串）或 AX（字串）中，然后自动修改 SI。一般不使用重复前缀。

LOOP 标号 （循环指令）

执行动作，CX=CX-1，若 CX 不为零，则跳转至目标标号，否则结束循环

LOOPZ\LOOPE 标号

执行动作，CX=CX-1，若 CX 不为零，且 ZF=1（相等），则跳转至标号，否则结束循环。

LOOPNZ\LOOPNE 标号

执行动作，CX=CX-1，若 CX 不为零，且 ZF=0（不相等），则跳转至标号，否则结束循环。

LCXZ：CX=0，则转移。它不修改 CX 的值。

LENGTH：表示取所定义存储区的长度（指的是变量单元个数，与 SIZE 区分）

M

MOV（MOV 指令不影响标志位）

目标操作数不允许为立即数；源操作数和目标操作数不能同时为存储器操作数比如 MOV [BX],[DI]是错的一解决存储器与存储器之间的数据交换问题：MOV AX,[1000H]\\MOV [2000H]，AX；不能有指令指针 IP 比如 MOV [SI],IP 是错的;CS 段寄存器不能做 MOV 指令的目的操作数；目标操作数不能为立即数；传送类型要匹配比如 MOV BL,SI+2 是错的。立即数不能直接送到段寄存器比如 MOV DS,2000H 是错的（可以通过 AX 中转）；OFFSET 表示取偏移地址；目标操作数和源操作数不允许同时为段寄存器比如 MOV DS,CS 是错的。 MOV BX,OFFSET [SI]是错的，应改为 MOV BX,SI。汇编不能确定内存操作数的类型时，需要用 BYTE PTR or WORD PTR 指明类型。如 MOV WORD PTR [SI],12H

MUL 源 （无符号数乘法）

字节乘法（目标操作数必须是 AL）：AL 乘以源→AX。字乘法（目标操作数必须是 AX）：AX 乘以源→DX，AX。★源操作数不能为立即数。★除 CF 和 OF 外，对其他状态标志位无意义。

MOVSB\MOVSW：字节传送、字传送 (DS:SI→ES:DI)

★将源串的值送到相应的目标串单元中，同时修改指针 SI\DI。

★CLD，DF 清零，STD，DF 置 1

★重复前缀 REP：串操作指令执行后，CX=CX-1，判断 CX 是否为 0，若不为 0，则继续执行串操作指令，否则执行下一条指令。REP 不能单独使用，必须配合串操作指令。格式为 REP MOVSB

★MOVSB 修改指针每次加减 1，MOVSW 修改指针每次加减 2

N

NEG 目标操作数：将目标操作数当做有符号数，取负后送回目标。

实质上 NEG 指令相当于将目标操作数固定为 0 值的 SUB 指令。

NOT 源 逻辑非，所有位取反。不影响标志位。

O

OR 目，源。逻辑运算，取或

OFFSET：表示取其后变量或标号的偏移地址

ORG 伪指令：指出其后的程序段或数据块存放的起始偏移地址

ORG <表达式> 例: ORG 100H || DA1 DB 68H。则 DA1 的偏移地址为：100H

P

PUSH 和 POP

POP 指令的的目标操作数不允许是 CS 寄存器；堆栈的存取操作每次必须是 2 个字节，没有单字节的操作指令，进栈总是从高位地址向低位地址存放数据；物理地址由 SS 和 SP 或 SS 和 BP 决定，SS 是段地址；每执行一次进栈指令，SP-2，先减 2，再对此地址和高一位的地址进行操作。PUSH 和 POP 指令的操作数可以是存储器。POP 和 POSH 指令都不影响标志位。

ROL\ROR\RCL\RCR

ROL 不带进位的循环左移；ROR 不带进位的循环右移；RCL 带进位的循环左移；RCR 带进位的循环右移；移位位数超过 1 位，放入 CL

PUSHF\POPF：将标志寄存器压栈\将栈顶弹出至标志寄存器。

R

REPNZ\REPNE（两个完全等价）

串操作指令执行后 CX=CX-1，如果 CX≠0 且结果不相等（即 ZF=0）则继续执行该串操作指令，否则顺序执行下一条指令

格式：REPNE SCASB 或 REPNE SCASW（和 SCASB 或 SCASW 配合起来用）可以实现寻找关键字的功能

REPZ\REPE（串比较指令）（两个完全等价）

串操作指令执行后 CX=CX-1，若 CX≠0 且结果相等（ZF=1），则继续执行该串操作，否则顺序执行下一条指令。格式：REPE CMPSB

ROL\ROR\RCL\RCR 循环移位指令 格式：ROL 目，1 或 CL

ROL:不带进位的循环左移，移走的位送入 CF，送入空缺位 ROR：不带进位的循环右移 RCL：带进位的循环左移 RCR：带进位的循环右移。

REP（重复前缀）

串操作指令执行后，CX=CX-1，判断 CX 是否为 0，若不为 0，则继续执行串操作指令，否则执行下一条指令。

RET\RETF（返回 CALL 指令的后续指令）

返回指令，用栈中数据修改 IP/CS：IP 完成转移。RET 用于近过程的返回，仅修改 IP，RETF 用于远过程的返回，修改 CS：IP。

S

STD：将方向标志 DF 置 1

STI（IF 置 1）

STOSB\STOSW（串存储指令）

将 AL（字节串）或 AX（字串）的值复制到目标串 ES：DI 所指的单元处，然后自动修改 DI。可加重复前缀 REP

SCASB\SCASW（串检索指令）

用 AL（字节串）或 AX（字串）的值减去目标串中 ES：DI 所指的串元素值，结果反应在标志位上，然后自动修改 DI

SAHF：将 AH 寄存器传递给标志寄存器的低 8 位。

SUB 目，源（同 ADD，为减操作，具体看 ADD）

SBB 目，源（同 ADC，为减操作，具体看 ADC）

SAR 算术右移 若符号位（第一位为 1），则移动过程中保持符号位不变，右边位送入 CF

SHL\SAL 逻辑左移 ，SHL 左移的位送入 CF 寄存器中，右边补 0

SHR 逻辑右移 右移的位送入 CF 寄存器中，左边补 0

以上三条：算术右移是对带符号数进行右移，在移位过程中必须保持符号位不变。逻辑移位是对无符号数移位，总是用 0 来填补已空出的位置。允许的操作数类型：通用寄存器、存储器。对标志位的影响：移出位送至 CF，按照移位后的结果置/复位 SF\ZF\PF。若移位位数超过 1 位，移位位数放到 CL 寄存器。

SEG：表示取其后变量或标号的段地址

SIZE：表示取所定义存储区的字节数

SEGMENT 和 ENDS 伪指令：

段名 SEGMENT [定位类型] [组合类型] ['类别']

⋮

段名 ENDS

1）定位类型：表示该段对起始边界地址的要求，有 4 种定位类型：PARA（默认）(段起点从节边界开始，1 节=16 字节)起始地址 = XXXXXXXXXXXXXXXX0000，节型。BYTE (段起点从存储器任何地址开始)起始地址=XXXXXXXXXXXXXXXXX，字节型。WORD (段起点从偶地址开始)起始地址=XXXXXXXXXXXXXXXX0，字型。PAGE (段起点从页边界开始，1 页=256 字节)起始地址=XXXXXXXXXXX00000000，页型

2）组合类型：连接时与其它模块中的同名同类别段在满足定位类型的前提下具有的组合方式。有 6 种：NONE(默认)—— 不和其他段组合。PUBLIC—— 和其他同为 PUBLIC 组合类型的同名同类，别的段连接成一个段，运行时装入同一物理段中，使用同一段基址。STACK——和其他同为 STACK 组合类型的同名同类别的段连接成一个大堆栈段，由各模块共享。COMMON——和其他同为 COMMON 组合类型的同名同类别的段连接时，被重叠地放在一起。这样可使不同模块的变量、标号使用同一存储区，便于模块间通信。MEMORY——MEMORY 类型的段连接时被放在所有段的最后(高段地址)。

若有几个段都是 MEMORY 类型，则汇编时遇到的第一个段认为 MEMORY 类型，其他段为 COMMON 类型。AT 表达式——表达式的值即该段段基值(段寄存器值)，不能用来指定代码段段基值

3）类别：连接(LINK)时只将同类别的段进行组合。典型的类别有 'CODE' 、'STACK' 、'DATA' 等。例: 数据段：DATA SEGMENT PARA PUBLIC 'DATA' 代码段：CODE SEGMENT PARA PUBLIC 'CODE' 堆栈段：STACK SEGMENT PARA STACK 'STACK'

T

TEST 目，源，与运算，不保留结果，影响标志，用于跳转。

TYPE：取变量类型对应的字节数（DB:1,DW:2）或标号类型对应的数值（NEAR:-1,FAR=-2）

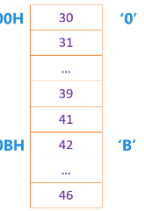
X

XCHG 目标，源

使两个操作数互相交换，其中一个操作数必须在寄存器中，另一个操作数可以在寄存器或存储器中。不允许使用段寄存器，不影响标志位。例如 SCHG [SI+BP],BX

XLAT（换码指令）

执行操作：AL←[DS*16+BX+AL]。使用方法：1.建立字节表格，字节表格的首地址放到 BX 中。2.把要转换的码存入 AL 寄存器。3.XLAT。例题：内存数据段有一张 16 进制数的 ASCII 码表，首地址为 2000H，如欲查出表中第 11 个代码，即十六进制数 B 的 ASCII 码。ASCII 表在 DS 段中，DS=4000H.MOV BX,2000H\\MOV AL，0BH\\XLAT 和 MOV BX,2000H\\MOV SI,0BH\\MOV AL,[BX][SI]等价



★存放位移量的是 8 位寄存器 AL，所建字节表格的长度不能超过 256 字节。

★XLAT 不影响标志位。

XOR 目，源 逻辑运算，取反，按位异或操作。有 1 的取反

第四章

1）程序设计语言概述

一、机器语言机器语言是由 0、1 二进制代码书写和存储的指令与数据。二、汇编语言汇编语言是用指令的助记符、符号地址、标号等书写程序的语言，简称符号语言。汇编语言依赖于特定计算机的结构和指令系统。三、高级语言高级语言是脱离具体机器的通用语言，不依赖于特定计算机的结构和指令系统。

2）8086/8088 汇编语言的基本语法

①汇编语言语句的种类和格式

1）语句的种类：在 8086/8088 汇编语言中，有三种基本语句：指令语句、伪指令语句和宏指令语句

指令语句是一种执行性语句，它在汇编时，汇编程序将为之产生对应的机器目标代码。

伪指令:是一种说明性语句，为汇编程序提供汇编所需相关信息，本身并不生成目标代码。其作用：定义变量；分配存储区定义逻辑段；指示程序开始和结束；定义过程等。

宏指令：是以某个宏名字定义的一段指令程序，在汇编时，凡有宏指令的地方都将用相应的指令序列的目标代码插入。宏指令语句是一般性指令语句的扩展。

2）语句格式（1）指令语句的格式 [标号：]{前缀}指令助记符[操作数表][；注释]。其中，[ ]表示可以任选的部分；操作数表是由逗号分隔开的多个操作数。(2) 伪指令语句的格式 [名字] 伪操作命令[操作数表][；注释] 注意：伪指令的名字后面没有冒号，这是伪指令语句和指令语句在格式上的主要区别。

②指令语句

1、标号：标号表示指令地址，是指令的符号地址。通常，"标号" 只在循环、转移和调用指令中使用。标号用一标识符定义，即以字母开头。由字母、数字、特殊字符组成的字符串表示。标号的最大长度一般不超过 31 个字符，除宏指令外，标号不能与保留字相同。2、前缀： 8086/8088 中有些特殊指令，它们常作为前缀同其它指令配合使用。例如：串操作指令的重复前缀（REP、REPE/REPZ、REPNE/REPNZ）以及总线封锁指令 LOCK 等，都是前缀。3、指令助记符：指令助记符主要为 8086/8088CPU 指令系统中的指令助记符，前面介绍的六大类指令。4、操作数： 8086、8088 汇编语言中的操作数有如下几种形式：(1) 立即数 (2) 寄存器操作数 (3) 存储器操作数 (4) I/O 端口操作数 5、各种寻址方式下操作数的表达式：(1) 常量与数值表达式 (2) 变量和地址表达式 (3) 运算符 IBM 宏汇编中有 5 种运算符,即:算术运算符、逻辑运算符、关系运算符、分析运算符、合成运算符

3）汇编程序的基本结构（编程举例）

①顺序结构程序

②分支结构程序：一般用条件转移指令实现

③循环结构程序：可使用 LOOP 指令或条件转移指令实现

第六章

1.一些细碎的知识点

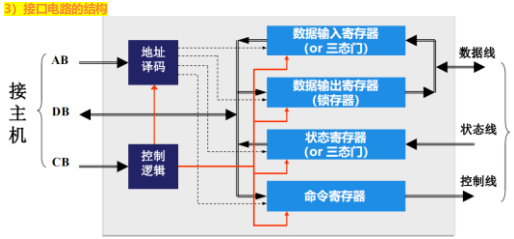
1）接口电路的作用：

电平转换、数据格式转换（串行和并行）、协调速度（输出锁存（输出数据时，一般都需要锁存器将要输出的数据保持一段时间，让外设有足够的时间读取数据）、输入缓冲（输入数据时，因外设输入数据的保持时间相对于 CPU 的接收速度来说较长，故输入数据通常不用加锁存器来锁存，而直接使用三态缓冲器与 CPU 数据总线相连即可））、总线隔离（三态门）

2）接口电路传送的信息：

数据信息（包括数字量（通常为 8 位二进制数或 ASCII 码）、模拟量、开关量）、状态信息（反应外设当前所处工作状态的信息，输入时，告知 CPU 输入设备数据是否准备好 (READY=1?) 输出时，告知 CPU 输出设备是否空闲（BUSY=0? ））、控制信息（用于控制外设的启动或停止）

3）接口电路的结构



★GLZ 复习提纲分类：数据输入缓冲器\输出寄存器、状态寄存器、命令控制寄存器、地址译码电路、控制逻辑（如果是中断方式，则包含中断控制逻辑）

★地址译码（用于接口内部个寄存器及芯片本身的寻址），数据寄存器（用于存放 CPU 与 IO 设备之间传送数据），状态寄存器（接口工作的状态标志，如 busy 供 CPU 查询），命令寄存器（存放控制字，设定接口功能，工作方式等）

4）CPU 与外设之间的数据传送的方式

①程序控制方式

无条件传送：要求外设总是处于准备好状态。优点：软件及接口硬件简单。缺点：只适用于简单外设，适应范围较窄如，开关、继电器、发光二极管、数码管等

查询工作方式：仅当条件满足时才能进行数据传送；每满足一次条件只能进行一次数据传送。适用场合：外设并不总是准备好，外设对传送速率和效率要求不高，CPU 不忙。工作条件：外设应提供设备状态信息，接口应具备状态端口

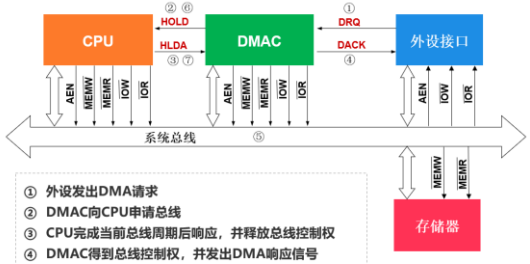
②中断方式传送

中断方式传送（接口电路复杂，但不用一直查询等待，可以提高 CPU 和系统中多个外设的工作效率，适用于 CPU 较忙，系统中外设较多的场合）。

中断是外设“主动”请求 CPU，准备传送数据的一种方式。当中断发生时，CPU 暂停其现行程序，转而执行中断处理程序，完成 I/O 工作；当中断处理完毕后，CPU 又返回到原来的任务。

③直接存储器存取（DMA）

DMA 方式是一种完全由硬件控制 I/O 交换的工作方式。在这种方式中，DMA 控制器从 CPU 完全接管对总线的控制，数据传送不经过 CPU，而是直接在内存和 I/O 设备之间传送。优点：数据传输由 DMA 硬件来控制，数据直接在内存和外设之间交换，可以达到很高的传输速率。如，计算机中硬盘和内存的数据传送就采用 DMA 方式。



① 外设发出DMA请求
② DMAC向CPU申请总线
③ CPU完成当前总线周期后响应，并释放总线控制权
④ DMAC得到总线控制权，并发出DMA响应信号
⑤ 由DMAC发控制信号，控制外设与存储器间的数据传送
⑥ 数据传送完后，DMAC撤销HOLD信号
⑦ CPU释放HLDA信号，并重新控制总线

④数据传送方式小结

	特点	优点	缺点	适用场合
查询	需要CPU执行指令来控制传送过程	传送数据经CPU中转(AL,AX)	接口电路和程序简单	CPU和外设的工作效率低 数据传输速率低	适用于CPU不忙、系统中外设较少的场合
中断	需要CPU执行指令来控制传送过程	传送数据经CPU中转(AL,AX)	CPU和外设工作效率高	接口电路和程序较复杂	适用于CPU较忙、系统中外设较多的场合 如键盘、打印机
DMA	无需CPU执行指令来控制传送过程，而是由DMAC(DMA控制器)来控制数据传送	传送数据不经CPU中转	数据传送速率高	接口电路和程序较复杂	适用于内存与高速外设之间（如硬盘）或者高速外设之间高速、大批量传送数据的场合

5）中断源的分类

外部中断（异常中断、软件中断）+**内部中断**（可屏蔽中断-INTR 中断、非屏蔽中断-NMI 中断）

6）CPU 响应中断及处理过程

1 关中断 CPU 响应中断后，在发出中断响应信号（/INTA）的同时，内部自动地由硬件实现关中断，以免在处理当前中断时又被新的中断源中断，破坏当前中断服务现场 **2 保护断点** CPU 响应中断后，立即将断点地址（被中断下一条指令的 PC 值）压栈保护，以备中断处理完毕后，CPU 能返回断点处继续运行主程序。**3 保护现场**保护标志位的状态是 CPU 自动完成的，保护寄存器值是在中断服务程序中用 PUSH 指令完成的 **4 给出中断入口(地址)，转入中断服务程序** 8086\8088 由中断源提供中断类型号，并根据中断类型号在中断向量表中取得中断服务程序的入口地址。**5 中断服务 6 恢复现场**把保存在堆栈中的有关寄存器的内容和标志位的状态弹出恢复。恢复寄存器值是在中断服务程序中用 POP 指令完成的 **7 中断返回**在中断服务程序的最后，要安排一条中断返回指令（IRET），将堆栈内保存的断点弹出至 PC，CPU 恢复至断点处继续运行，弹出恢复标志寄存器的值，同时，开中断以便 CPU 能响应新的中断请求。

7）内部中断的特点：

内部中断的中断类型号已由系统预先定义，如：除法出错（类型 0）INTO（类型 4）、单步中断（类型 1）、断点中断（类型 3）。或者中断类型号包含在指令代码中，如 INT N 指令，中断类型号 n 包含在指令中。除单步中断以外，所有内部中断都不能被屏蔽。所有内部中断都没有中断响应总线周期，这是因为内部中断不必通过查询外部来获得中断类型号。除了单步中断以外，所有内部中断的优先权都比外部中断的优先权高

8）8086\8088 系统的中断源、各类中断源获取类型码的方式。

1 外部中断

①可屏蔽中断 INTR（高电平有效）若 IF=0，CPU 关中断，不响应 INTR。若 IF=1，CPU 开中断，响应 INTR，并通过/INTA 引脚向产生 INTR 的设备接口（中断源）发回响应信号，启动中断过程。IF 位的状态可用指令 STI 使其置位，即开中断，也可用 CLI 指令使其复位，即关中断。外部可屏蔽中断是由 8259 在第 2 个中断响应（/INTA）周期送出中断类型码到数据总线③非屏蔽中断 NMI（上升沿有效）一旦此中断请求信号产生，不管标志位 IF 的状态如何，CPU 都响应它。非屏蔽中断源有三种（系统板上 RAM 的奇偶校验错、扩展槽中的 IO 通道错、协处理器 8087 的中断请求）NMI 比 INTR 的中断请求的级别高，因此，若在指令执行过程中，INTR 和 NMI 引脚上同时都有中断请求信号，则 CPU 将首先响应 NMI 引脚上的中断请求信号。NMI 的中断类型号预先定义为类型 2，所以 CPU 响应非屏蔽中断时，不要求外部向 CPU 提供中断类型号，CPU 在总线上也不发 INTA 信号。

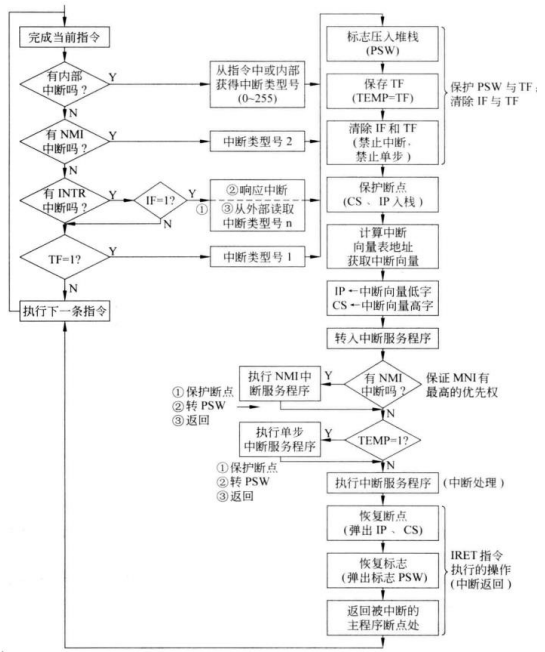
2 内部中断（由芯片自动提供中断类型码）

除法出错中断——类型 0（优先级最高），执行除法指令时，若除数为 0 或商超过范围，则立即产生类型为 0 的内部中断，CPU 转向除法出错的中断服务程序。单步中断—类型 1 标志位 TF 被置位(TF=1)时，8086\8088 处于单步工作方式，即 CPU 每执行完一条指令后就自动产生一个类型 1 的内部中断，程序控制将转入单步中断服务程序。CPU 响应单步中断后将自动把状态标志压入堆栈，然后清楚 TF 和 IF 标志位。由于在中断时 TF 位被保护起来了，中断返回时 TF 位又被重新恢复成 1，所以 CPU 在中断返回以后仍然处于单步工作方式（类型 2 为 NMI，见上一条）。断点中断—类型 3。8086\8088 指令系统中有一条设置程序断点的单字节中断指令（INT 3），执行该指令后就会产生一个中断类型为 3 的内部中断。主要用于软件调试中。溢出中断—类型 4 若上一条指令执行的结果使标志位置一（OF=1），则在执行溢出中断（INTO）指令时，将引起类型 4 的内部中断，CPU 就可以转入对溢出错误进行处理的中断服务程序。若 OF=0，则 INTO 指令执行空操作。INTO 指令常常紧跟在算术运算指令之后，以便在该指令执行产生溢出时，由 INTO 指令进行特殊的处理。用户定义的软件中断—类型 n,n 为中断类型码。软件中断（INT n）则是从指令的第 2 个字节中读得中断类型码。③外部可屏蔽中断（INTR）是由 8259 在第二个中断响应（/INTA）周期送出中断类型码到数据总线。

9）8086\8088 系统中断的优先级次序：

（优先级次序由低到高：单步类型 1（调试用，软件中断）、INTR 外设送入类型（可屏蔽中断，硬件中断）、NMI 类型 2（非屏蔽中断、硬件中断）、INTO 类型 4（溢出用，软件中断）、INT n 类型 n（内部检查用中断，软件中断）、除法类型 0（商大于被除数，软件中断））。**除了单步中断以外，所有内部中断的优先权都比外部中断的优先权高。**

10）中断处理流程图：实际上 NMI 是最优先处理的



11）可屏蔽中断 INTR 的过程：

首先，中断请求信号 INTR 由外部设备产生并送至 8086 的 INTR 引脚上（CPU 是否响应取决于 CPU 内部的 IF 标志，如果 IF 标志为 0，则 CPU 不会识别中断，当 IF=1 并出现 INTR 请求信号时，CPU 在完成正在执行的指令后，便开始响应中断）。

响应步骤：①CPU 读取中断类型号 n,CPU 发出 INTA 信号，外部中断接口电路把中断类型号通过数据总线送给 CPU②按先后顺序把 FLAG、CS 和 IP 的当前内容压入堆栈③清除 IF 和 TF 标志④查中断向量表，把 4×n+2 的字存储单元中的内容读入 CS 中，把 4×n 的字存储单元中内容读入 IP 中⑤转去执行中断服务城区

可屏蔽中断的响应时序：第 1 个 INTA 响应周期：通知外部响应其中断请求了。第二个 INTA 响应周期：通知外部送中断类型码到数据总线。

12）中断向量表（也称中断入口地址表）

★中断向量表存储在内存 00000H 到 003FFH 之间，占 1K 字节。中断向量表分成 256 组，每组两个字（即 4 个字节）组成一个双字长的指针（其高地址存放的是中断服务程序入口地址所在的代码段的 16 位段地址（CS）低地址字是中断服务程序入口地址的偏移值（IP））**CPU 通过 CS 和 IP 的值得到一个 20 位的地址**（16 位偏移地址左移 4 位+IP 值），**它就是中断服务程序的实际入口地址。**

★中断向量表分为 3 个部分：第一部分：类型 0-类型 4 专用中断，已由系统定义，不允许用户修改**第二部分：**类型 5-类型 31 系统备用中断，一般不允许用户作其他用途，其中许多中断已被系统开发使用，例如 21H 已用作系统功能调用的软中断。**第三部分：**类型 32-255，可供用户使用，这些中断可由用户定义为软中断，由 INT n 指令引入，也可以是通过 INTR 端直接引入硬件可屏蔽中断，使用时用户要自行设置中断向量。**例：**若某中断的类型号为 8，则这个中断向量在内存的起始地址为：8×4=32=20H，若中断向量如图所示，则 CS=1000，IP=0200，中断服务程序的入口地址为 10200

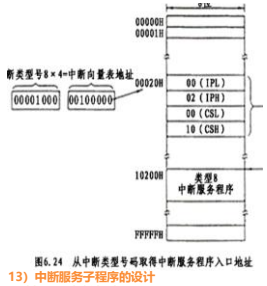


图6.24 从中断类型号码取得中断服务程序入口地址

13）中断服务子程序的设计

①选择一个中断类型号

如果是硬件中断，则要使用硬件决定的中断类型号；**如果是软件中断，**则可以在系统预留给用户的中断类型中选一个。（有时也可以用系统中已用的中断，重新编写中断服务程序）

②设置中断向量

法一：用数据传送指令设置：将中断服务子程序入口偏移地址放在物理地址为 4×n 的字单元中，将中断服务子程序入口段地址存放在物理地址为 4×n+2 的字单元中。

法二：采用 DOS 系统功能调用设置：向量号 21H，功能号 25H

③设计中断服务程序

2.中断总结

1）21H 向量号

01H、07H、08H

功能描述：从标准输入设备（如：键盘）读入一个字符，读中断在处理过程中一直处于等待状态直到有字符可读为止。**入口参数：**AH=01H 过滤掉控制字符，并回显；AH=07H，不过滤掉控制字符，不回显；AH=08H，过滤掉控制字符，不回显。**出口参数：**AL=输入字符的 ASCII 码

02H（显示字符）

功能描述：向标准输出设备（如：屏幕）输出一个字符。**入口参数：**AH=02H;DL=待输出字符的 ASCII 码。**出口参数：**无

06H（控制台输入/输出）

功能描述：控制台（如键盘、屏幕）输入\输出。**入口参数：**AH=06H,DL=输入\输出的功能选择，**出口参数：**若 DL=00-FEH，则此功能为输出，DL 为待输出字符的 ASCII 码，若 DL=0FFH，则此功能为输入，此时：若 ZF=1，则无字符可读，否则，AL=读入字符的 ASCII 码

08H（键盘单字符输入但不显示）

功能：等待键盘输入一字符，其 ASCII 码存放在 AL 中，但不显示。这种功能往往在设置密码时使用。假如键盘敲了数字“1”，那么，AL=31H，但屏幕上不显示数字“1”。

09H（显示字符串）

功能：在屏幕上显示字符串。**入口参数：**DS：DX 指向字符串的首地址 **注意点：**显示的字符串必须以‘$’结束；所显示的内容不应出现非可见的 ASCII 码；若考虑输出格式需要，在定义字符串后，加上回车和换行符。

例题：

例：(P140)

在显示器上显示“How are you?”,然后读一个字符，但不显示此字符，若读入字符是‘y’,则显示‘ok’。

```
D       SEGMENT
D1      DB   'How are you?', 0DH, 0AH,   '$'
D2      DB   'OK', 0DH, 0AH,   '$'
D       ENDS
C       SEGMENT
        ASSUME  CS: C,  DS: D
BG:     MOV     AX,     D
        MOV     DS,     AX
        MOV     DX,     OFFSET  D1
        MOV     AH,     9
        INT     21H             ；显示:How are you?
        MOV     AH,     8
        INT     21H     ；输入字符不回显
        CMP     AL,     'Y'
        JNE     NEXT
        LEA             DX,     D2  ；显示O K
        MOV     AH,     9
        INT     21H
NEXT:   MOV     AH,       4CH
        INT             21H
C       ENDS
        END     BG
```

0AH（字符串输入（回车一字符串输入结束符））

入口参数：DS：DX 指向输入缓冲区。输入缓冲区格式：

N1	N2	长度（N1-1）的缓冲区	0DH

N1：程序员设定的最大键入的字符串长度，包括回车

N2：实际键入的字符数，不包括回车，系统自动写入

25H（设置中断向量表）

功能描述：设置中断向量表。**入口参数：**AH=25H，AL=中断类型号，DS:DX=中断处理程序的入口地址。**出口参数：**无

例：从片 8259 的 IR3 的中断服务程序入口地址为 5000:1500，该从片的中断类型号为 50H-57H。编程将此中断的中断向量写入中断向量表中。

```
MOV AX,5000H (立即数不能直接送入段寄存器 DS 中) \\MOV DS,AX\\MOV DX,1500H\\MOV AL,53H（本题中 IR3 的中断类型号为 53H）\\MOV AH,25H\\INT 21H
```

2CH（读取系统时间）

功能描述：取系统时间。**入口参数：**AH=2CH。**出口参数：**CH=时，CL=分，DH=秒，DL=百分秒

2DH（设置时间）

入口参数：CH=小时（0~23），CL=分（0~59）DH=秒（0~59），DL=百分之一秒

4CH（返回 DOS）

功能描述：终止程序的执行，并可返回一个代码。**入口参数：**AH=4CH，AL=返回的代码。**出口参数：**无

2）10H 向量号（显示中断）

0（设置显示方式）

入口参数：AL=显示方式号（0—7）

2（设置光标位置）

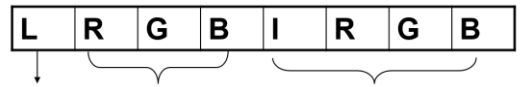
入口参数：BH=页号，DH=行号，DL=列号

例：将光标位置设置在屏幕第 10 行第 30 列。

```
MOV     BH,  0    ;第 0 页
MOV     DH,  10   ;第 10 行
MOV     DL,  30   ;第 30 列
MOV     AH,  2
INT     10H
```

6（屏幕上滚）

入口参数： AL = 上滚行数，AL=0 时，清除屏幕。CH、CL = 屏幕滚动区域左上角行、列号。DH、DL =屏幕滚动区域右下角行、列号



闪烁　　背景色　　字符色彩

例：红底白字的属性字节：01000111 =47H。蓝底黄字的属性字节：00010110 =16H

7（屏幕下滚）

入口参数：与功能 6 滚动方向相反，其他相同

9（在当前光标处写字符和属性）

入口参数：BH = 页号，AL = 显示字符的 ASCII 码，BL = 属性，CX = 重复显示次数

3）类型 0（内部中断，除法出错中断，优先级最高）

执行除法指令时，若除数为 0，或商超过范围，则立即产生类型为 0 的内部中断，CPU 转向出发出错的中断服务程序。

4）类型 1（单步中断）

标志位 TF 被置位（TF=1）时，8086\8088 处于单步工作方式，即 CPU 每执行完一条指令后，就自动地产生一个类型 1 的内部中断，程序控制将转入单步中断服务程序。CPU 响应单步中断后将自动把状态标志压入堆栈，然后清除 TF 和 TF 标志位。由于在中断时 TF 位被保护起来了，中断返回时 TF 位又被重新恢复（TF=1），所以 CPU 在中断返回以后仍然处于单步工作方式。除了单步中断外，所有内部中断都不能被屏蔽

5）类型 3（断点中断）

执行该指令以后就会产生一个中断类型为 3 的内部中断，CPU 将转向执行一个断点中断服务程序，以便进行一些特殊的处理，断点中断主要用于软件调试中，程序员可以用它在程序中设置一个程序断点。

6）类型 4（内部中断，溢出中断）

若上一条指令执行的结果使溢出标志位置 1（OF=1），则在执行溢出中断（INTO）指令时，将引起类型 4 的内部中断，CPU 就可以转入对溢出错误进行处理的中断服务程序。若 OF=0 时，则 INTO 指令执行空操作 INTO 指令常常紧跟在算术运算指令之后，以便在该指令执行产生溢出时进行特殊的处理

7）类型 INT 16（键盘中断）

①功能号：0

返回参数：AL=输入字符的 ASCII 码，AH = 键盘扫描码，(b7： 1—键按下，0—键松开）**应用：**键盘模拟电子琴（按下键发声，松开键不发声）

②功能号：1

返回参数：按过键，ZF=0，AL=输入字符的 ASCII 码;未按键，ZF=1;

③功能号：2

返回参数：AL=特殊功能键的状态

第七章

1.一些细碎的知识点

1）IO 端口

端口：接口电路中用于缓存数据、状态、及控制信息的部件，分为：数据端口、状态端口、控制端口

①IO 端口编址：

计算机系统中包含多个不同功能的接口电路。每个接口电路又可能包含 1 个或多个端口。

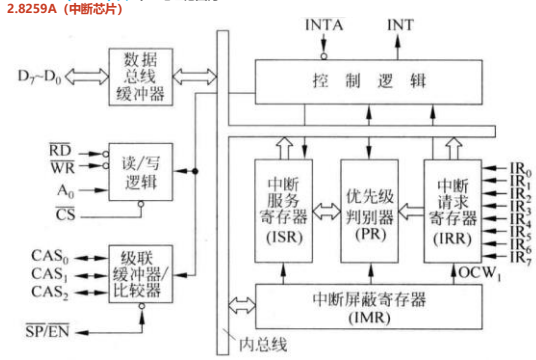
②寻址端口的方法：

先找到端口所在的接口电路芯片（片选），在该芯片上找到具体要访问的端口（片内地址），若接口中仅有一个端口，则找到芯片即找到端口，若接口中有多个端口，则找到芯片后需再找端口。每个端口地址=片选地址（高位地址）+片内地址。

③8086/8088 的 IO 端口编址

采用 I/O 独立编址方式；I/O 操作只使用 20 根地址线中的 16 根：A_{15}～A_0**；可寻址的 I/O 端口数为：**2^{16} = 64K (65536)个；**I/O 地址范围为：**0～0FFFFH

2.8259A（中断芯片）



1）一些细碎的知识点

①8259A 特性介绍

功能：8259A 是一个功能很强的中断扩充和多中断管理芯片，具有中断扩展、自动提供中断类型码、中断优先级裁决等中断管理功能。**可编程：**内部有多个寄存器以及功能部件都是可编程的，使用方便。**级联扩展：**单片可以连接 8 个中断请求源，将多个 8259 级联扩展，最多可以扩充到 64 级中断。**总结：**通过对 8259A 编程，可以设置中断触发方式、中断类型码、中断屏蔽方式、中断优先级方式、中断结束方式等。

②8259A 内部结构：

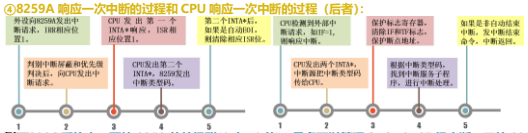
数据总线缓冲器（三态、双向、8 位寄存器）：数据线 D7-D0 与 CPU 数据总线连接，构成 CPU 与 8259A 之间信息传送的通道。**读写控制逻辑：**用来接收 CPU 系统总线的读写控制信号和端口地址选择信号，用于控制 8259A 内部寄存器的读写操作。**级联缓冲比较器：**8259A 可工作于单片或多片方式，级联缓冲/比较器提供多片 8259A 的管理和选择功能，其中一片为主片，其余为从片。**控制逻辑：**负责向片内各部件发送控制信号，向 CPU 发送中断请求信号 INT 和接收 CPU 回送的 INTA*信号，控制 8259A 进入中断管理状态。**中断请求寄存器：**IRR 是一个 8 位寄存器，用于记录外部中断请求。当 IRi 有请求时，IRR 中的相应位 Di 置 1。在中断响应信号有效时，Di 被清除。**中断屏蔽寄存器：**当 IMR 中的 Di 位为 1 时，对应的 IRi 请求被禁止；当 IMR 中的 Di 位为 0 时，则允许对应的中断请求进入。它可以由软件设置或清除。**优先权判决器：**对 IRR 中被未屏蔽的中断进行优先级比较，选出当前优先级最高级的中断请求。**中断服务寄存器：**记录 CPU 当前正在服务的中断标志。IRi 请求得到响应时，在 CPU 的第一个 INTA*将 ISR 中相应位置 1，ISR 复位由中断结束方式决定。

③8259A 的级联

单片可以连接 8 个中断请求源，可以将多个 8259 级联，最多扩充到 64 级中断。

优先级顺序：假设从片接主片 IR3，主片+从片的中断源优先级顺序为：IR0 主> IR1 主> IR2 主> IR0 从> IR1 从> IR2 从> IR3 从> IR4 从> IR5 从> IR6 从> IR7 从> IR4 主> IR5 主> IR6 主> IR7 主

④8259A 响应一次中断的过程和 CPU 响应一次中断的过程（后者）：



例：8086 系统中，两片 8259 芯片级联(1 主+1 从)，最多可以管理 8+8×1=15 级中断；三片 8259 级联(1 主+2 从)，最多可以管理 6+8×2=22 级中断；多片 8259A 级联(最多 1 主+8 从)，最多可以管理 0+8×8=64 级中断。

例：外部可屏蔽中断 INTR 的中断类型码是由 8259A 在中断响应的第 2 个总线周期送出的。

⑤8259A 的工作方式

1）中断触发方式：边沿触发方式：8259A 的 IRQI 引脚上出现上升沿信号表示有中断请求。电平触发方式：8259A 的 IRQi 引脚上出现高电平信号表示有中断请求；在第 1 个 INTA*结束前，IRi 必须保持高电平；另外，应注意及时撤除高电平，否则可能会引起不应该出现的第二次中断。

2）中断屏蔽方式：普通屏蔽方式：通过对中断屏蔽寄存器设定，实现对中断请求的屏蔽。中断屏蔽寄存器的每一位对应一个中断请求。1：屏蔽，0：开放。特殊屏蔽方式：提供了允许较低优先级的中断能够得到响应的特殊手段。假设当前正在处理 IR6，进入特殊屏蔽方式，这时除 IR6 和被 OCW1 屏蔽的中断，其余所有中断请求均能打断当前中断。

3）优先权方式：一般全嵌套方式：默认优先权管理方式，规定 IR0 最高，IR7 最低。即优先权 IR0>IR1>IR2>IR3>IR4>IR5>IR6>IR7 特殊全嵌套方式：与完全嵌套方式基本类似，区别在于允许同级的中断请求进入。通常主片设为特殊全嵌套方式，从片设为一般全嵌套方式自动循环方式：一个中断服务完成后，其优先级自动降为最低，与之相邻低一级中断请求源设为最高。开始时 IR0 优先级最高

例：原中断优先级次序为：IR0＞IR1 ＞ … ＞ IR7，对 IR3 中断服务完成后， IR4＞IR5＞ …＞IR2＞IR3 特殊循环方式与优先级自动循环方式相似，主要区别是可以通过编程设置开始的最低优先级。

例：初始设置 IR2 为最低优先级，则 IR3 为最高优先级。

主片+从片的中断源优先级排序：主片的中断源按优先级顺序排好后，主片上接从片的 IRi 用从片排好优先级顺序的 8 个中断源替换掉。例：两片 8259A 级联，从片接主片 IR3，主片的优先级方式为特殊全嵌套，从片的优先级方式为一般全嵌套，且主片、从片的所有中断源都未被屏蔽。问：1）主片+从片的中断源优先级顺序为：IR0 主＞ IR1 主＞ IR2 主＞ IR0 从＞ IR1 从＞ IR2 从＞ IR3 从＞ IR4 从＞ IR5 从＞ IR6 从＞ IR7 从＞ IR4 主＞ IR5 主＞ IR6 主＞ IR7 主 2）如果正在响应处理从片 IR5 的中断请求时，又有从片 IR3 的中断请求，那么，该中断请求是否会立即被响应？是否能实现中断嵌套？答：优先级 IR3 从＞ IR5 从所以， IR3 从中断请求会立即被响应，能实现中断嵌套。

4）中断结束方式：中断结束（EOI）：中断处理结束后须将 ISR 相应位清 0，以开放同级或低级的中断请求。自动中断结束：自动结束方式(AEOI)：CPU 响应中断请求时，在第二个 INTA 脉冲的后沿，由 8259A 自动将 ISR 的相应位清 0。普通的 EOI：CPU 向 8259A 发出普通 EOI 命令时，8259A 把所有正在服务的中断中优先权最高的 ISR 位复位。在中断服务程序返回之前，发 EOI 命令。特殊的 EOI CPU 向 8259A 发送一条特殊 EOI 命令，命令中指出了要清除哪个 ISR 位（用 3 位编码指定清除位）（适用于任意优先权方式）。

5）数据线连接方式：缓冲方式：8259A 的数据线连接总线驱动芯片，非缓冲方式：8259A 的数据线直接与 CPU 的数据总线相接。

例题：单片 8259A，一般全嵌套，IR1,IR3,IR6 引脚同时有中断请求，请问：1）IRR=？ 2）根据优先级方式，先响应那个中断请求？ 3）在 IR3 中断服务过程中又来了 IR0 中断请求，会被响应吗？ 4）如果响应了，ISR=？ 1）IRR= 01001010，2）优先级方式为一般全嵌套，IR1＞IR3＞IR6 所以，先响应 IR1，3）会被响应，因为 IR0 优先级比 IR3 高。4）ISR= 00001001

2）8259A 编程概要

★8259A 使用前要进行初始化（ICW1~ICW4），之后，初始化所设置的状态保持不变，但可根据实际情况对相关中断操作进行编程（OCW1~OCW3）

★ICW 须按顺序设置，OCW 可以不按顺序，任意设置。

3）8259A 的初始化

①初始化命令字：

◆设置顺序：ICW须按顺序设置

OCW可以不按顺序，任意设置

	A0	控制字（b7~b0）中 标志位
ICW1	0	b4=1
ICW2	1	
ICW3	1	
ICW4	1	
OCW1	1	
OCW2	0	b4=0，b3=0
OCW3	0	b4=0，b3=1

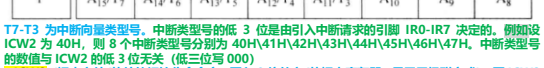
在编程过程中，必须按照确定顺序写向规定的端口地址写命令字，对寄存器进行设置。

指令举例 MOV AL,11H；OUT 0A0H, AL

★**ICW1**（芯片控制初始化命令字）写 ICW1 的标记为 A0=0，D4=1

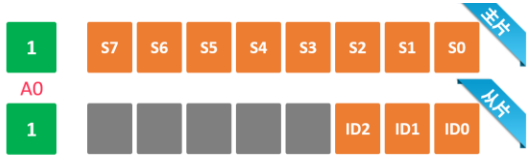
1(D4)特征位，表示现在设置的是 ICW1 而不是别的命令字 LTMI 若为 1，表示中断请求为电平触发方式；若为 0，则表示中断请求为边沿触发方式，且为上升沿触发，并保持高电平 SNGL1:表示单片 8259A 方式，0 表示多片 IC4：写 ICW4；0：不写 ICW4 。没有使用的位通常设置为 0

★**ICW2**（设置中断类型号的初始化命令字，该字写入 8 位的中断类型寄存器）标记为 A0=1



T7-T3 为中断向量类型号。中断类型号的低 3 位是由引入中断请求的引脚 IR0-IR7 决定的。例如设 ICW2 为 40H，则 8 个中断类型号分别为 40H\41H\42H\43H\44H\45H\46H\47H。中断类型号的数值与 ICW2 的低 3 位无关（低三位写 000）

★**ICW3**（标志主片/从片的初始化命令字，写入 8 位的主/从标志寄存器，只用于级联方式），写 ICW3 的标记为 A0=1。多片才需要写 ICW3



对于主 8259A（输入端 SP=1），置 1 位表示接了从片例如当 ICW3=F0 时，表示在 IR7\IR6\IR5\IR4 引脚上接有 8259A 从片，而 IR3\IR2\IR1\IR0 引脚上未接从片

对于从 8259A（输入端 SP=0），ID2～ID0 是从片接到主片 IRi 的标识码。

★**ICW4**（中断结束方式初始化命令字，写入 8 位的方式控制寄存器）标记为 A0=1。

D7D6D5=000 是 ICW4 的识别码;**D4（SFNM）**：SFNM=1 为特殊的全嵌套工作方式，SFNM=0 为一般的（非特殊的）全嵌套工作方式全嵌套方式中，中断请求按优先级 0-7 进行处理，0 级中断的优先级最高，只有更高级的中断请求到来时才进行嵌套；而特殊的嵌套方式允许同级的中断请求。如果对 8259A 进行初始化以后没有设置其他优先级方式，则 8259A 按全嵌套方式工作 **D3（BUF）**：BUF=1 为缓冲方式，8259A 的 SP\EN 端和总线驱动器（即数据总线缓冲器）的允许端相连，利用从 SP\EN 端输出的低电平，可以作为总线驱动器的启动信号；BUF=0 为非缓冲方式，8259A 直接连接 CPU 的数据总线 **D2（M/S）**：在缓冲方式下用于区别本片是主片还是从片，1 表示主片，0 表示从片。当 BUF=0 时，此位无意义 **D1（AEOI）** 若 AEOI=1，则设置自动结束中断方式，在 CPU 响应中断请求的过程中，当它向 8259A 发出的第 2 个 INTA 脉冲结束时，自动清除当前 ISR 中的对应位。在中断结束返回时，无须进行任何操作即自动结束中断。若 AEOI=0，则为非自动结束中断方式，它要求 CPU 发 EOI 命令后才能消除 ISR 中的对应位。**D0**：在 8086/8088 系统中，此位=1

②操作命令字

★**OCW1**（中断屏蔽命令字，OCW1 写入 IMR 寄存器，用来屏蔽中断请求）标记为 A0=1

A0

M7-M1 对应于 IMR 各位，Mi=1 表示该位中断被屏蔽，0 则表示允许中断。允许中断的引脚上的中断请求被允许进入 8259A 的下一级（即优先级判别器）

★**OCW2**（优先级循环方式和中断结束方式操作命令字）标记为 A0=0,D3=D4=0。

R\SL\EOI 三位：R：1 优先级自动循环方式（用于多个中断源其优先级相等的场合，此时按照 IR0-IR7 为高低顺序自动排列）；0 优先级非自动循环方式 **SL**1 特殊中断结束，L2-L0 有效（其编码对应的某位 Iri 为最低优先权）；0 一般中断结束，L2-L0 无效。**EOI**：EOI=1 时，使当前 ISR 中的对应位复位。**L2-L0（有效时）**：在特殊中断结束命令时，指出具体要清除 ISR 中哪一位；在特殊优先级循环方式命令时，指出循环开始时哪个中断优先级最低。

001 普通 EOI；011 特殊 EOI，按码复位；101EOI 优先级自动循环；100 设置优先级自动循环；000 清除优先级自动循环；111EOI 且按码优先级自动循环；110 按码优先级自动循环；010 无意义

★**OCW3**（多功能操作命令字，用于控制中断屏蔽、设置查询方式和读 8259A 内部寄存器）标记为 A0=0,D7=D4=0,D2=1

ESMM 位和为特殊的屏蔽模式允许位，SMM 为特殊的屏蔽模式位 00 无用；01 无用；10 清除特殊屏蔽（系统恢复原来的优先级工作方式）；11 设置特殊屏蔽（脱离当前的优先级方式，此时只要 CPU 内标志寄存器 IF=1，则 8259A 可以相应任何一级未被屏蔽的中断请求）**P（查询方式位）** P=1 将 8259A 设置成中断查询方式 **RR（读寄存器位）** RR=1，表示允许读 8259A 的状态（ISR 和 IRR 的内容）**RIS 位读** ISR 或 IRR 的选择位，必须与 RR 位配合使用，RR=1 时，若 RIS=0 表示读 IRR，若 RIS=1，表示读 ISR

③8259A 初始化步骤

★写 ICW1→写 ICW2→多片需写 ICW3→需要 ICW4 则写 ICW4

★一旦初始化以后，若要改变工作方式等，则必须重新进行初始化编程。

4）8259A 编程示例

①8259 初始化

主片：边沿触发、要 ICW4，中断类型码 18H~1FH
从片接主片 IR2
非缓冲方式、非自动结束中断方式、特殊全嵌套

```
MOV   AL,   000100001B
OUT   80H,  AL          ; ICW1
MOV   AL,   18H
OUT   81H,  AL          ; ICW2
MOV   AL,   00000100B
OUT   81H,  AL          ; ICW3
MOV   AL,   000100001B
OUT   81H,  AL          ; ICW4
```

从片：边沿触发、要 ICW4，中断类型码 30H~37H
从片接主片 IR2
非缓冲方式、非自动结束中断方式、一般全嵌套

```
MOV   AL,    00010001B
OUT   82H,   AL          ; ICW1
MOV   AL,    30H
OUT   83H,   AL          ; ICW2
MOV   AL,    00000010B
OUT   83H,   AL          ; ICW3
MOV   AL,    00000001B
OUT   83H,   AL          ; ICW4
```

②写中断屏蔽字：

```
;主片
IN    AL,    81H
AND   AL,    11010011B  ;开放主片 IR2 IR3 IR5
OUT   81H,   AL
;从片
IN    AL,    83H
AND   AL,    10101101B  ;开放从片 IR1 IR4 IR6
OUT   81H,   AL
```

③设置主片 IR3 中断向量：

```
CLI                     ;硬件中断设置，必须先关中断
MOV   AX,    SEG   M_INT3
MOV   DS,    AX
MOV   DX,    OFFSET  M_INT3
MOV   AL,    1BH     ;IR3 中断类型码=18H+3=1BH
MOV   AH,    25H
INT   21H
STI                     ;设置完，再开中断
```

④主片 IR3 中断服务程序：

```
M_INT3：  STI                 ; 开中断，可以实现中断嵌套
PUSH  AX
PUSH  BX      ; 保护现场
……
……                ; 中断处理
POP   BX
POP   AX      ; 恢复现场
MOV   AL, 20H
OUT   80H, AL       ; 发 EOI 命令给 8259 主片
IRET                ; 中断返回
```

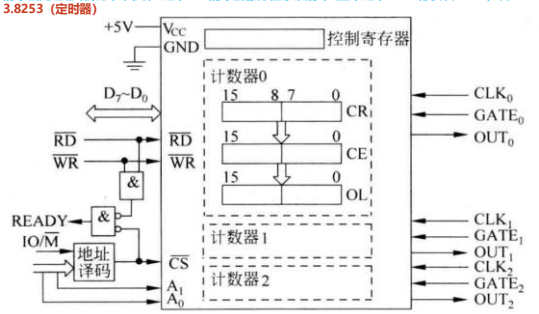
⑤从片 IR6 中断服务程序：

```
S_INT6：  STI                 ; 开中断，可以实现中断嵌套
PUSH  AX
PUSH  BX      ; 保护现场
……
……                ; 中断处理
POP   BX
POP   AX      ; 恢复现场
MOV   AL, 20H
OUT   82H, AL       ; 发 EOI 命令给 8259 从片
OUT   80H, AL       ; 发 EOI 命令给 8259 主片
IRET                ; 中断返回
```

注：从片 IR6 中断处理完成后，主片、从片 8259 都要发 EOI 命令。

因为从片的中断请求是接到主片的 IR2，所以，从片 IR6 中断处理结束时，对主片来说，其 IR2 的中断处理也结束了，因此，从片 IR6 的中断服务程序最后 ，主片从片 8259 都要发 EOI 命令。

3.8253（定时器）



1）一些细碎的知识点

①8253 功能介绍

★8253 是一种可编程的计数器/定时器接口芯片，最高计数频率为 2MHz，可用于产生各种定时波形，也可用于对外部事件计数。内部有三个独立的 16 位减一计数器，通过设置控制字，各计数器可以工作于 6 种工作方式。

★端口号的 A0A1 为 00、01、10 对应计数器为 0-2，11 对应控制字

★有 3 个独立结构完全相同的 16 位计数器和 1 位 8 位控制字寄存器。每个计数器内部可以分为计数初值寄存器（CR）、计数执行部件（CE）、输出锁存器（OL）

★8253 的初始化需要：写入控制字、写入计数初始值

★8253 采用 BCD 计数时，计数器初值为 0 时，计数次数达到最大

②计数方式小结

16位减一计数器

	二进制计数	BCD码计数
最大计数次数	2^{16}(65536)	10^4(10000)
最大计数次数对应的计数初值	0	0
计数值范围	1～2^{16}(65536)	1～10^4(10000)
计数次数对应的计数初值(例如，计数次数为1000)	1000	1000H

二进制计数(减一计数)：0000H-1=0FFFFH=2^{16}-1，再计(2^{16}-1) 个数，计数值减到0。初值0 对应的总计数次数=1+ (2^{16}-1) =2^{16}

BCD计数(减一计数)：0000H-1=9999H=10^4-1，再计(10^4-1) 个数，计数值减到0。初值0 对应的总计数次数=1+ (10^4-1) =10^4

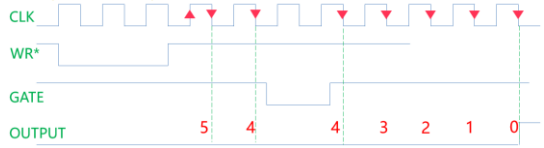
2）控制字

SC1	SC0	RW1	RW0	M2	M1	M0	BCD

SC1 和 SC0 用来选计数器，如 10 表示选计数器 2。RW1 和 RW0：00 对计数器进行锁存；01 只读\写低 8 位字节（高 8 位自动置零）；10 只读\写高 8 位字节；11 先读\写低 8 位字节，再读写高 8 位字节。M2\M1\M0：如 011 对应模式 3BCD：1 计数值为 BCD 码格式，0 计数值为二进制格式

3）工作方式

方式 0（计数结束产生中断）



★当写入控制字后，OUT 段输出低电平作为起始电平，在有两个负脉冲宽度的$\overline{WR}$信号的上升沿将初值写入计数初值寄存器（CR），待计数初值装入寄存器后，输出仍保持低电平。

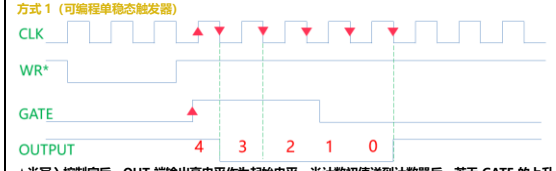
★若 GATE 的门控信号为高电平，当 CLK 端每来一个计数脉冲，计数器就做减一计数，当计数值减为 0 时，OUT 输出变为高电平。

★GATE 为计数控制门。方式 0 的计数过程是由门控信号 GATE 来控制暂停，即当 GATE=1 时，允许计数；而当 GATE=0 时，停止计数。所以计数过程中，GATE 端应保持高电平。

★软件启动，不自动重复计数，装入初值后 OUT 变低电平，计数结束后 OUT 输出高电平。

★举例：8253 端口地址范围是：40H~43H，使 1 号计数器工作在方式 0，计数初值为 0FF5H，二进制计数。写出 8253 的初始化程序。MOV AL,01110000B（写控制字，端口号的最后一位为控制字，选 1 号计数器，先读写低 8 位，再读写高 8 位，工作方式 0，二进制格式）\\OUT 43H,AL\\MOV AL,0F5H\\OUT 41H,AL\\MOV AL,0FH\\OUT 41H,AL（计数初值装入 41H 计数器 1，且先装低位再装高位）

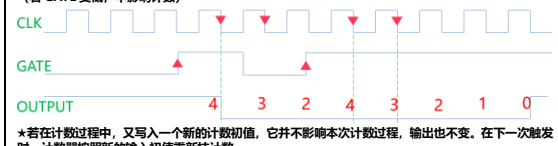
方式 1（可编程单稳态触发器）



★当写入控制字后，OUT 端输出高电平作为起始电平。当计数初值送到计数器后，若无 GATE 的上升沿，那么不管此时 GATE 输入的触发电平是高电平还是低电平，都不会开始减 1 计数。必须等到 GATE 端输入正跳变的触发脉冲时，计数过程才会开始。

★工作时，由 GATE 输入触发脉冲的上升沿使 OUT 变为低电平，每来一个计数脉冲，计数器做减一计数，当计数值减为 0 时，OUT 再变为高电平。OUT 端输出的单稳负脉冲的宽度为计数器的初值乘以 CLK 端输入脉冲周期。

★如果在计数器未减到 0 时，门控端 GATE 又来一个触发脉冲，则下一个时钟周期脉冲开始，计数器将从原有的初始值重新作减 1 计数，当减至 0 时，输出端又变为高电平。这样会使输出脉冲宽度延长。（若 GATE 变低，不影响计数）

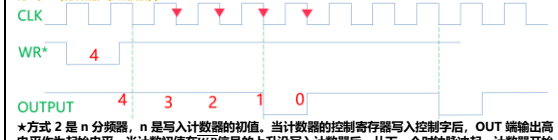


★若在计数过程中，又写入一个新的计数初值，它并不影响本次计数过程，输出也不变。在下一次触发时，计数器按照新的输入初值重新技计数。

★例：8253 时钟为 2MHz，使用 1 号计数器产生 500μs 的负脉冲。采用 BCD 计数。写出初始化程序。500/0.5 = 1000，则计数初值为 1000。

程序：MOV AL,01110011B\\OUT 43H,AL\\MOV AL,0H\\OUT 41H,AL\\MOV AL,10H\\OUT 41H,AL（采用 BCD 码计数，直接先低位后高位输出 1000 即可）

方式 2（分频器（矩形波））



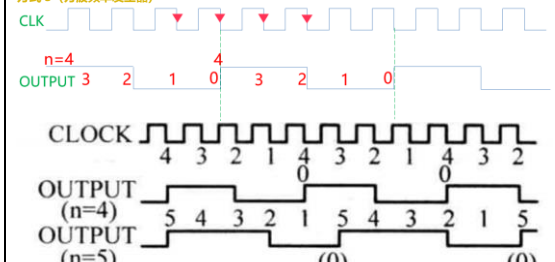
★方式 2 是 n 分频器，n 是写入计数器的初值。当计数器的控制寄存器写入控制字后，OUT 端输出高电平作为起始电平。当计数初值在$\overline{WR}$信号的上升沿写入计数器后，从下一个时钟脉冲起，计数器开始作减 1 计数。当减到 1 时，OUT 端输出将变为低电平。当计数端 CLK 输入 n 个计数脉冲后，在输出端 OUT 输出一个 n 分频脉冲，其正脉冲宽度为（n-1）个输入脉冲时钟周期，而负脉冲宽度只是一个输入脉冲时钟周期。GATE 用来控制计数，GATE=1，允许计数;GATE=0，停止计数。因此，可以用 GATE 来使计数器同步。方式 2 下，不但高电平的门控信号有效，上升跳变的门控信号也是有效的。

★方式 2 下可以自动重复计数。计数过程中修改初值不影响本轮计数过程，下一个计数周期按新的 n 值计数。

★例：8253 时钟为 2MHz 用 1 号计数器输出 40KHz 序列负脉冲，采用二进制计数。写出初始化程序。2000/40 = 50，计数初值为 50

初始程序：MOV AL,01010100B\\OUT 43H,AL\\MOV AL,50（不加 H 默认十进制，或者写成 32H 也可以（如果是 BCD 码计数，应该写 50H））\\OUT 41H,AL

方式 3（方波频率发生器）



★方式 3 输出为方波或者为近似对称的方波。当计数器的控制寄存器写入控制字后，OUT 端输出低电平作为起始电平。当计数器装入计数值 n，OUT 端输出变为高电平。如果当前门控信号 GATE 为高电平。当计数值 n 为偶数时：每当计数值减到 n/2 时，则 OUT 端由高电平变为低电平，并一直保持计数到 0，故输出的 n 分频波为方波；当 n 为奇数时：输出分频波的高电平宽度变为（n+1）/2 计数脉冲周期，低电平宽度为（n-1）/2 计数脉冲周期。

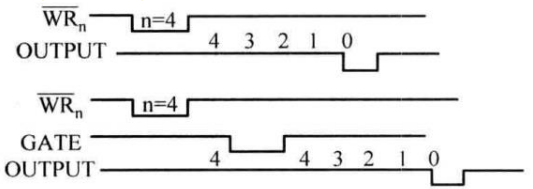
★计数中修改初值不影响本半轮计数过程，其余与方式 2 类似。

★例：8253 时钟为 2MHz，用 0 号计数器输出 2KHz 的方波，采用二进制计数。

写出初始化程序。2000/2 = 1000

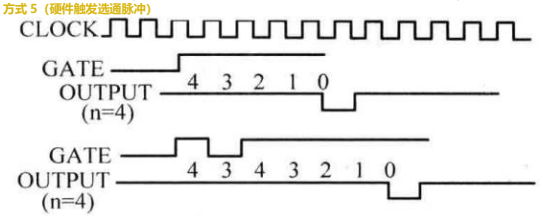
程序：MOV AL,00110110B\\OUT 43H,AL\\MOV AX,1000（采用二进制计数，1000 默认为十进制数）\\OUT 40H,AL\\MOV AL,AH\\OUT 40H,AL

方式 4（软件触发选通脉冲）



★计数器写入控制字后，输出的 OUT 信号变为高电平。当由软件触发写入初始值 n 经过 1 个时钟周期后，若 GATE=1，允许计数，则计数器在一个时钟脉冲之后开始做减一计数，当计数器减到 0 时，在 OUT 端输出一个宽度等于一个计数脉冲周期的负脉冲。若 GATE=0，则停止计数，只有在 GATE 恢复高电平之后才重新计数（即由 n=4 开始减 1 计数，直到减至 0 才发出一个选通负脉冲）

方式 5（硬件触发选通脉冲）



★方式 5 类似于方式 4，所不同的是 GATE 端输入信号的作用不同。由 GATE 输入触发脉冲，从其上升沿开始，计数器作减一计数，计数结束时，在 OUT 端输出一个宽度等于计数脉冲周期的负脉冲。在此方式中，计数器可重新触发。在任何时刻，当 GATE 触发脉冲上升沿到来时，将把计数初值重新送入计数器，然后开始计数过程。

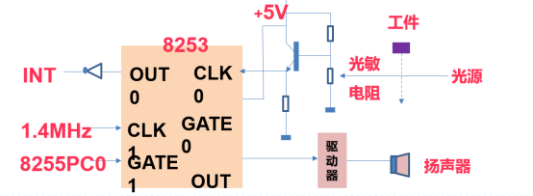
工作方式总结：

	自动重复计数	Gate信号	开始计数	输出信号	应用
方式0	否	H：允许计数 L：暂停计数 上升沿：继续计数	预置完初值	计数结束，输出由低电平跳变为高电平	计数结束产生中断请求信号
方式1	否	上升沿：启动计数 L：不影响计数	Gate信号上升沿	负脉冲 （计数期间一直为低）	产生宽度可设置的单脉冲
方式2	是	H：允许计数 L：暂停计数 上升沿：重新计数	预置完初值	连续的负脉冲序列 （每次计数的最后一个周期为低电平）	产生分频脉冲
方式3	是	H：允许计数 L：暂停计数 上升沿：重新计数	预置完初值	连续的方波信号	分频
方式4	否	H：允许计数 L：暂停计数 上升沿：重新计数	预置完初值	负脉冲 （计数值减为0后的一个周期为低电平）	产生一个周期的负脉冲选通信号
方式5	否	上升沿：启动计数 L：不影响计数	Gate信号上升沿	负脉冲 （计数值减为0后的一个周期为低电平）	产生一个周期的负脉冲选通信号

例：8253 六种工作方式中，（1）哪些方式是自动重复计数？（2）哪些方式不是自动重复计数？（3）什么方式能计数结束产生中断请求？（4）什么方式能做分频用？（5）什么方式可产生宽度可设置的负脉冲？（6）什么方式可产生一个时钟周期宽度的负脉冲？ 2.8253 的六种工作方式中，哪些方式是由门控信号（Gate）的上升沿启动计数的？哪些方式是预置完初值开始计数的？这些方式下，要正常计数，门控信号（Gate）应为____电平。**答案** 1.(1)方式 2、方式 3；(2)方式 0、方式 1、方式 4、方式 5；(3)方式 0；　(4)方式 3、方式 2　；(5)方式 1；(6)方式 4、方式 5　2.方式 1、方式 5；方式 0、方式 2、方式 3、方式 4；高电平

4）8253 编程示例 2：

下图是用8253监测的一个生产流水线示意图，每通过50个工件，扬声器响5秒钟，频率为2000Hz。



- 计数器0：方式2/方式3，每50个CLK0产生一个中断，控制字为：00010101B，初值只写低8位，BCD计数，初值为50H。
- 计数器1：工作于方式3（方波方式），控制字为：01110111B，BCD计数。　分频比=（1.4×10⁶）/2000=700，　初值=700H

BCD 计数，装入 50H 则表示十进制的 50

8253端口地址：40H~43H
8255端口地址：60H~63H

```
MOV   AL, 00H
OUT   63H, AL      ; 8255PC0清0, 8253GATE1为“低电平”
MOV   AL, 15H
OUT   43H, AL      ; 置计数器0的工作方式
MOV   AL, 50H
OUT   40H, AL      ; 装初值
MOV   AL, 77H
OUT   43H, AL      ; 置计数器1的工作方式
MOV   AL, 00H
OUT   41H, AL      ; 装初值，低8位
MOV   AL, 07H
OUT   41H, AL      ; 装初值，高8位
STI                ; 开中断
LOP: NOP
JMP   LOP          ; 进入循环等待中断

INTP  PROC
      STI
      PUSH  AX
      MOV   AL, 01H      ;开8255的PC0，8253CNT1开始计数
      OUT   63H, AL
      CALL  DLY5S        ;调5S延迟子程序
      MOV   AL, 00H      ;关8255的PC0，8253CNT1停止计数
      OUT   63H, AL
      MOV   AL, 20H
      OUT   20H, AL
      POP   AX
      IRET
INTP  ENDP
```

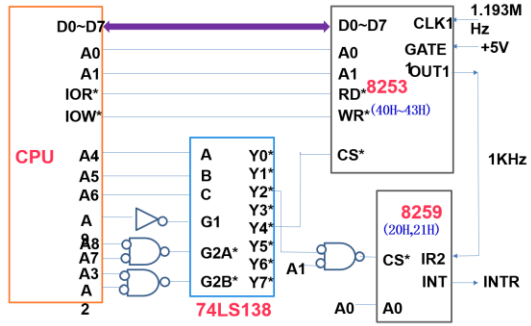
如何知道当前通过了多少个工件？

可以在中断服务子程序中设置一个变量记录中断次数，通过的工件数= 中断次数×50

5）8253 编程示例 3（本题为 8259A、8253,74138 的综合）（计数电路）

某 8086 系统中，8253 的口地址为 40h~43h，8259 的口地址为 20h~21h，中断类型码为 70h~77h。8253 计数器 1 的输入时钟为 1.193MHz，定时输出中断请求信号至 8259 的 IR2。在中断服务程序中对中断次数计数，存在 IR2_COUNT 单元中。 要求：1）设计硬件电路（分 1ms 和 1s 定时中断两种情况);2）编写 8253 初始化程序(分 1ms 和 1s 定时中断两种情况); 3）编程设置中断向量，设置中断屏蔽字；(IR2 的中断类型码为 72H) 4）编写中断服务程序(中断服务程序的入口标号是 IR2_SUB)。

硬 件 电 路（1ms定时—1KHz）



➢ 编程：

（1）8253初始化（1ms定时中断）：

通道1：方式3 (或方式2)，2进制计数

输入时钟1.193MHz，输出方波1kHz(周期1ms)

计数初值：1.193MHz / 1kHz = 1193

初始化程序：

```
MOV   AL,  01110110B
OUT   43H, AL
MOV   AX,  1193
OUT   41H, AL
MOV   AL,  AH
OUT   41H, AL
```

（2）设置中断向量：

```
CLI
MOV   AX,  SEG  IR2_SUB
MOV   DS,  AX
MOV   DX,  OFFSET  IR2_SUB
MOV   AL,  72H
MOV   AH,  25H
INT   21H
STI
```

（3）设置中断屏蔽字：

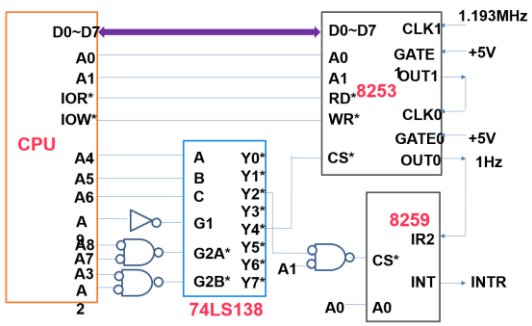
```
IN    AL,  21H
AND   AL,  11111011B
OUT   21H, AL
```

（4）中断服务程序：

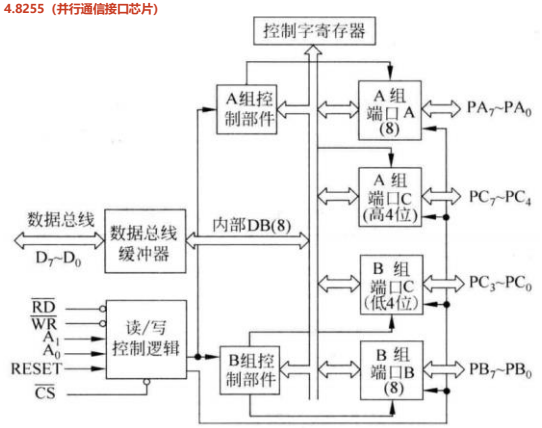
```
IR2_SUB:  STI
          PUSH  AX
          INC   WORD  PTR  IR2_COUNT
          MOV   AL,  20H
          OUT   20H, AL
          POP   AX
          IRET
```

若为 1s 定时，需要进行级联

硬 件 电 路（1s定时—1Hz）



4.8255（并行通信接口芯片）



1）一些细碎的知识点

①8255 功能介绍

★8255 是一种可编程的并行通信接口芯片，可用于 CPU 和外设之间进行并行数据传输。内部有三个 8 位的数据端口，有三种工作方式。

★端口号的 A0A1 为 00、01、10 分别表示读写 A、B、C 口；11 表示只写控制寄存器

②并行通信

指多位数据同时进行传送的方式，其特点是传输速度快。

③工作方式：

方式 0：基本的输入/输出方式　方式 1：选通的输入/输出方式　方式 2：选通的双向传输方式

A 口：可以工作在方式 0，方式 1，方式 2。三种 B 口：可工作在方式 0，方式 1 。两种 C 口：只能工作在方式 0 。一种无需设置工作方式，只要设置传送方向。C 口高四位和低四位可分开使用。

★方式 1、2：选通输入输出方式，可以中断方式传输，且 C 口会固定的引脚用作控制联络信号和中断请求信号。

★仅 A 口工作在方式 2 时，可以双向传输，A 口工作在方式 0、1 及 B、C 口只能单向传输

2）控制字

控制字分为端口的方式选择控制字（可使 8255 的 3 个数据端口工作在不同的方式）和 C 口的按位置位和复位控制字（可使 C 口的任意一位置位和复位）。

★控制字送入的端口为最后一个端口

①方式控制字

1	D6	D5	D4	D3	D2	D1	D0

D6\D5 设置 A 口的工作方式为 0\1\2;D4 设置端口 A 的输入 or 输出，1 为输入，0 为输出;D3 设置 PC7-PC4 输入 or 输出，1 为输入，0 为输出;D2 设置 B 口的工作方式，0 为工作方式 0,1 为工作方式 1（B 口只有两种工作方式）;D1 端口 B 输入 or 输出，1 为输入，0 为输出;D0 设置 PC3-PC0 输入 or 输出，1 为输入，0 为输出。C 口没有工作方式的定义，只有 IN 和 OUT 的定义。C 口高四位和低四位可以分开使用，也可联合使用。

例：8255 的端口地址为：60H~63H。假设 A 口工作在方式 0，输出，B 口工作在方式 1，输入。C 口高 4 位输入,低 4 位输出。方式控制字：1000 1110 初始化编程：MOV AL. 10001110 || OUT 63H,AL

②C 口的按位操作（置位和复位）

0	D6	D5	D4	D3	D2	D1	D0

D3\D2\D1 决定对 PCi 的哪一位进行操作；D0：1 表示置位，0 表示复位

★例：8255 的四个端口为：80H-83H，要对 PC7 置 1，对 PC3 置 0。对 PC7 置 1:0000 1111B，对 PC3 置零：0000 0110B。代码：MOV AL,0FH\\OUT 83H,AL\\MOV AL,06H\\OUT 83H,AL

3）工作方式

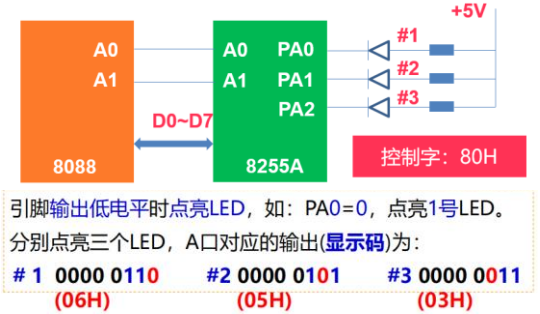
方式 0：基本的输入/输出方式，可用于无条件传送或查询式传送

★任何一个端口可以作为输入，也可以作为输出（不能同时输入输出）

★C 口的使用有四种情况：8 位输入，8 位输出，高四位输入低四位输出，高四位输出低四位输入

★各端口输入或输出总共有 16 种组合。

★例 1：用 8255A 控制三个发光二极管，依次循环点亮，端口地址为 340H-343H。



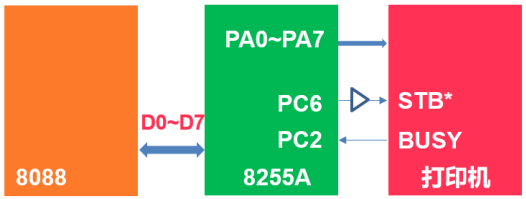
引脚输出低电平时点亮LED，如：PA0=0，点亮1号LED。

分别点亮三个LED，A口对应的输出(显示码)为：

1 0000 0110 (06H)　#2 0000 0101 (05H)　#3 0000 0011 (03H)

代码：LED DB 06 05 03\\.....\\MOV DX 343H\\MOV AL,80H\\OUT DX,AL（方式字设置为 A 口输出）\\LOP: MOV CX 3\\LEA BX,LED（把 LED 的偏移地址送至 BX）\\MOV DX,340H\\DON: MOV AL,[BX]\\OUT DX, AL\\CALL DELAY（这几行的目的为输出显示码）\\INC BX\\LOOP DON（在一次显示内循环，依次显示三个二极管）\\JMP LOP（循环上述操作）\\.....

★例 2。8255A 工作在方式 0，以查询方式向打印机传送数据，将缓冲区 BUF 开始的一串字符送打印机打印。8255 端口地址为 0D0H-0D3H。



PC2 连 BUSY，PC6 连 STB，工作时，先查询 BUSY 信号，如果 BUSY 信号=0，代表打印机不忙，打印一个字符。

代码：BUF DB “HELLO WORLD” \\……\\MOV AL,81H\\OUT 0D3H,AL（设置方式字）\\MOV AL 0DH（PC6 置 1）\\OUT 0D3H,AL\\LEA SI,BUF（将 BUF 的偏移地址送入 SI）\\MOV CX,11（空格也算字符）\\LOP: IN AL,0D2H（读 C 口，测试打印机的 BUSY 信号 PC2）\\AND AL,04H\\JNZ LOP（如果不是 0，则返回到 LOP 继续读 C 口信号）\\MOV AL,[SI]（数据送入 AL）\\OUT 0D0H,AL\\MOV AL,0CH（PC6 清零，STB 有效，打印机读取数据）\\OUT 0D3H,AL\\CALL DELAY\\INC AL（PC6 置 1，STB 无效）\\OUT 0D3H,AL\\INC SI（取下一字节）\\LOOP LOP

方式 1：选通的输入/输出方式。可用于查询方式传送或中断方式传送

★利用一组选通信号控制 A 口和 B 口的数据输入和输出

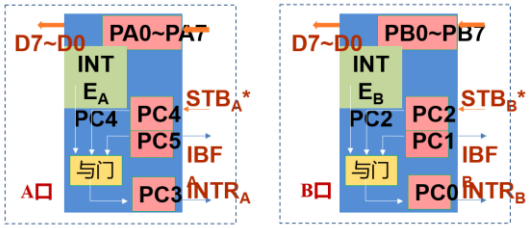
★A 口、B 口作输入或输出口，C 口的部分位用作选通控制信号

★A 口、B 口在作为输入和输出时的选通信号不同

★C 口的 8 位除用作选通信号外，其余位可工作于方式 0 下，作为输入或输出口。

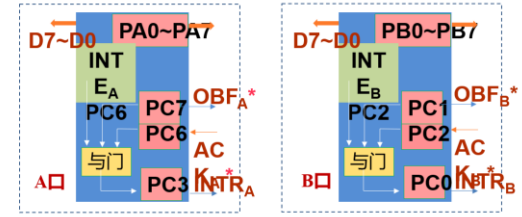
★A 口 B 口输入选通控制信号：

方式1：A口 B口 输入 选通控制信号



- STB*：选通信号，外设发出，外设送的数据锁存到8255的输入口。
- IBF：输入缓冲器满，8255发出，表示外设送的数据已进入输入口。
- INTR：中断请求信号，8255 发出，用来向CPU发出中断申请。
  STB、IBF、INTE有效时，8255自动发出INTR（接8259的IR）。
- INTE：中断允许控制信号，控制是否允许8255中断请求INTR发出。

方式1：A口 B口 输出 选通控制信号



- OBF*：输出缓冲器满，8255发出，数据已进入输出口，外设可取。
- ACK*：响应信号，外设发出，数据已被外设取走，可传送下一个数据。
- INTR：中断请求信号，8255 发出，用来向CPU发出中断申请。
  OBF 、ACK、INTE有效时，8255自动发出INTR（接8259的IR）。
- INTE：中断允许控制信号，控制是否允许8255中断请求INTR发出。

例 题 5

8255以中断方式连接打印机。A口方式1输出，将缓冲区BUF开始的11个字符打印。8255端口地址为 D0H~D3H。

PC7为输出缓冲满信号，通知打印机 — 控制字?

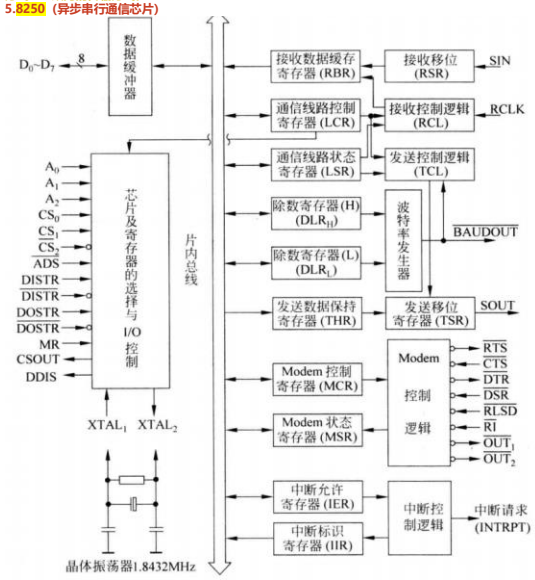
PC6作为打印机的响应信号

PC3为INTR信号。 — A0H

```
BUF   DB  'Hello world'
      ……
      MOV  AL, 0A0H
      OUT  0D3H, AL      ; 方式字设置
      MOV  AL, 0DH
      OUT  0D3H, AL      ; PC6(INTE)置1，开中断
      LEA  SI, BUF       ; 数据存储区设置
LOP:  MOV  CX, 11
      JCXZ LOP           ; 等待
INT_PRT  PROC FAR
      MOV  AL, [SI]
      OUT  0D0H, AL      ; A口输出打印数据
      INC  SI
      DEC  CX            ; 取下一个字节
      ……
      IRET
INT_PRT  ENDP
```

方式 2：双向的传输方式。

5.8250（异步串行通信芯片）



1）一些细碎的知识点

①8250 特性：

支持异步串行通信规程，发送时可自动插入起始位、停止位和奇偶校验位，接收时能自动去除。内部具有可编程的时钟产生电路，发送和接收都采用双缓冲结构。可编程 MODEM 的控制信号，有中断式收发功能。

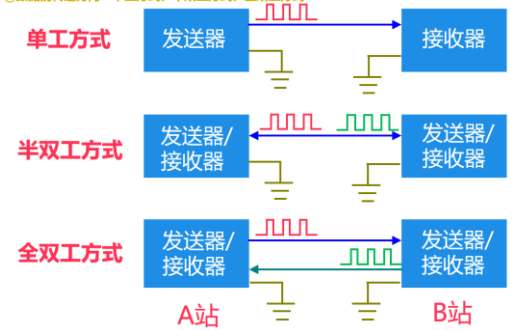
②8250 发送数据的工作过程

1.CPU 将数据发送给 8250 的 THR。2.THR 把数据送至 TSR，LSR 中‘THR 空’状态位置位。3.TSR 根据 LCR 规定的格式从低到高逐位发送数据。4.LSR 中‘THR 空’可产生中断，也可查询 LSR 的状态位。

③8250 接收送数据的工作过程

1.SIN 引脚上的串行数据逐位进入 RSR。2.RSR 根据 LCR 中规定接收一个完整数据并送至 RBR3.RBR 收到 RSR 的数据，将 LSR 中‘RBR 满’状态置位。4.LSR 中 ‘RBR 满’ 可产生中断，也可查询 LSR 的状态位。

④数据的传送方向：单工方式、半双工方式、全双工方式

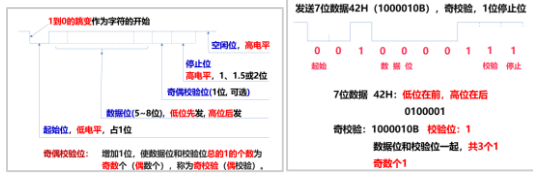


两台计算机之间的串口连接（简单的 3 线连接）：一台计算机的 TXD 接另一台计算机的 RXD，RXD 接另一台计算机的 TXD，另外，两台计算机的地线（GND）也相连接。

⑤串行通信的工作方式

异步工作方式：收发双方不要求时钟相同，传输速率双方约定。不发数据时处于高电平。有数据时，先发送一个低电平为起始位，然后发数据位、校验位、停止位。

串行异步通信的数据格式：（画图注意事项：低位先发、高位后发；看清几位数据位；注意起始位只有一个 0，没有 1）



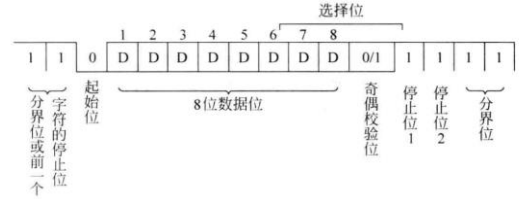
同步工作方式：收发双方必须用同一时钟协调，以确定数据中每一位的位置。先发送一到两个特殊同步字符，当收发同步后，连续发送一块数据。

例：假设传送数据为 65H，请给出 8250 SOUT 和串口 TXD 的信号波形(不用画图，用 0，1 表示高低电平)SOUT(TTL 0~5V)：0101001101。TXD(RS232 -12V~+12V):1010110010（TXD 和 SOUT 图形上是反的）

★SIN/SOUT TTL 电平：(0:0V，1:5V)

★串口 TXD\RXD RS232C 电平（0：+12V，1：-12V）

⑥异步数据传输格式：

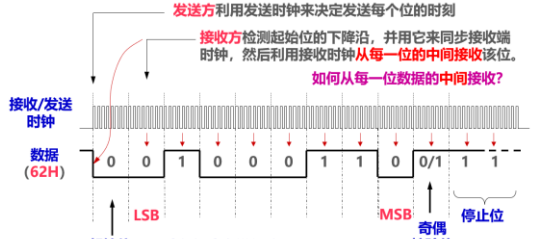


⑥波特率

每秒传送数据的位数，单位波特，Baud，用于表示数据传送的速率。

例：表示一个串行字符要 10 位，每秒传送 240 个字符。波特率=10 位/字符*240 字符/秒=2400（Baud）

⑦异步通信的时钟定时方法



接收时钟频率= n(波特率系数)*波特率，n一般为8,16

接收方从起始位的下降沿开始对接收时钟计数，计到n/2开始接收数据位，之后每计n个时钟接收一位数据。

⑧信号的调制与解调

调制：将数字信号转换成适于信道传输的模拟信号。调制方式：调幅、调频、调相

⑨通信协议：

协议：通信收/发双方必须共同遵守的基本通信规程。协议内容：收发双方的同步方式、传输速率、通信报文格式；传输控制步骤、及控制字符定义；差错检验方式、数据编码。

★串行异步通信的数据传送，先传送低位，再传送高位

⑩：8250 寄存器小结：

设置数据格式：线路控制寄存器 LCR(BASE+3)设置波特率：先将 LCR 的 DLAB 位置 1，再将分频系数的低 8 位、高 8 位分别写入除数寄存器的 BASE 和 BASE+1 两个端口。工作在查询方式时，可以查询线路状态寄存器 LSR(BASE+5)若要工作在中断方式，要先设置中断允许寄存器 IER(BASE+1)，在中断服务程序中查询中断识别寄存器 IIR(BASE+2)，从而完成数据收发。发送数据：写发送保持器 THR(BASE);接收数据：读接收缓冲器 RBR(BASE)。

2）各种寄存器及其控制字

线路控制寄存器（LCR）D7位	A2 A1 A0	被访问寄存器	地址(以首地址 BASE=3F8H为例)
0	0 0 0	接收缓冲器（读） 发送保持器（写）	3F8H（BASE）
1	0 0 0	除数寄存器（低字节）	3F8H(BASE)
0	0 0 1	中断允许寄存器(IER)	3F9H（BASE+1）
1	0 0 1	除数寄存器（高字节）	3F9H（BASE+1）
X	0 1 0	中断识别寄存器(IIR)	3FAH（BASE+2）
X	0 1 1	线路控制寄存器(LCR)	3FBH（BASE+3）
X	1 0 0	MODEM控制寄存器(MCR)	3FCH（BASE+4）
X	1 0 1	线路状态寄存器(LSR)	3FDH（BASE+5）
X	1 1 0	MODEM状态寄存器(MSR)	3FEH（BASE+6）

注：如果8250的端口地址有变化，只要将首地址BASE的值换成新的首地址，就可得到各寄存器的新端口地址。

①发送数据保持寄存器（THR）

发送数据时，CPU 将待发送的字符写入发送数据保持寄存器（THR）中，其中，第 0 位是串行发送的第 1 位数据。先由 8250 的硬件送入发送移位寄存器 TSR 中，在发送时钟的驱动下逐位将数据由 SOUT 引脚输出

②接受数据缓冲寄存器（RBR）

用于存放接收到的 1 个字符。当 8250 从 SIN 端接收到一个完整的字符后，会把该字符从接受移位寄存器送入 RBR 中，在 RBR 存放接收到的一个字符后，可由 CPU 将它读出，读出的数据只是一个字符帧中的数据部分，而起始位、奇偶校验位、停止位均被 8250 过滤掉。

③通信线路控制寄存器（LCR）(BASE+3)

设定了异步串行通信的数据格式

D7	D6	D5	D4	D3	D2	D1	D0

D7DLAB 标志位，0 允许访问接收、发送数据寄存器或中断允许寄存器，1 表示允许访问波特率因子寄存器；D6：0 正常操作，1 表示发空缺位，SOUT 强制为 0；D5\D4\D3：XX0 无校验位，001 奇校验，011 偶校验；101 奇校验并附加恒为 1,111 偶校验并附加恒为 0（附加位的作用是发送端把奇偶校验方式通过发送的信息告诉接收端，接收方将附加位分离出来可知发送方校验方式）；D2：0 表示 1 个停止位，1 表示 1.5 个停止位 (5 位数据) 或 2 个停止位 (6-8 位数据) D1\D0：00 表示 5 位数据，01 表示 6 位数据，以此类推

④波特率因子寄存器（第1寄存器）（分为 H 和 L）(BASE\BASE+1）PPT 上为除数寄存器（DLR）

8250 芯片规定当通信线路控制寄存器（LCR）写入 D7=1 时，接着对 I\O 口地址 3F8H、3F9H 可分别写入波特率因子的低字节和高字节，即写入除数寄存器 L 和除数寄存器 H 中，而波特率为 1.8432HZ/（波特率因子×16）如果要求发送波特率为 1200 波特，则波特率因子为 1.8432MHZ/1200×16=96

当线路控制寄存器 DLAB=1，表示触发写除数寄存器。

分频系数（和波特率因子是一个东西）=fs/(16×波特率)，8250 初始化时，必须将 16 位的分频系数分两次写入高低两个除数锁存器，地址为 A2A1A0 为 000 和 001。例：fs=1.8432MHz，波特率为 2400 波特。分频系数（除数寄存器的值）= 1.8432M/(16*2400) =48 =30H

⑤中断允许寄存器（IER）(BASE+1)

低 4 位允许 8250 设置 4 种类型的中断（将相应位置 1 即可），并通过 IRQ4 向 CPU 发中断请求，其低 4 位含义：D3=1：允许 Modem 状态改变中断；D2=1:允许接收数据出错中断；D1=1：允许发送保持器空中断；D0=1：允许接收缓冲器满中断

⑥中断标识寄存器（IRR）(BASE+2)

用来判断有无中断并判断是哪一类中断请求。IRR 的高 5 位恒为 0，只使用低 3 位作为 8250 的中断标识位。

0	0	0	0	0	D2	D1	D0

D2D1:00Modem 状态中断；01 发送保持寄存器控终端；10 接收缓冲寄存器满中断；11 线路状态中断。D0：0 有中断；1 无中断

⑦通信线路状态寄存器（LSR）(BASE+5)

用于向 CPU 提供有关 8250 数据传输的状态信息。

0	D6	D5	D4	D3	D2	D1	D0

D6：0 发送移位寄存器不空，1 发送移位寄存器空;D5：0 发送保持寄存器不空，1 发送保持寄存器空;D4：1 表示线路信号间断，这时接收的数据可能不正常;D3：1 接收到的数据停止位个数不正确;D2：奇偶校验位错标志，1 奇偶校验位与规定的奇偶校验不同，表示数据可能出错;D1：越限状态标志，1 表示接收数据寄存器中的前一个数据还未被 CPU 取走，而下一个数据已经到来，产生数据重叠出错;D0：1 表示 8250 已接收到一个有效的字符并将它放在接收数据缓冲器中，CPU 可以从 8250 的接收数据寄存器中读取，一旦读取后此位自动清 0。如果 D0=1 时，8250 又接收到一个新数据，就会替换前一个未取走的数据，8250 将产生一个重叠错误。测试 D5 用于发送，测试 D0 用于接收

⑧Modem 控制寄存器（MCR）(BASE+4)

用于设置联络线，以控制与调制解调器或数传机的接口信号。

0	0	0	D4	D3	D2	D1	D0

D4：0 表示 8250 正常工作，1 表示 8250 串行输出被回送，此时 SOUT 为高电平状态，SIN 将与外设分离，TSR 的数据由 8250 内部直接回送到 RSR 的输入端，形成“本地环”，同时 CTS、DSR、RI、RLSD 与外设相应线断开，而在 8250 内部分别与 RTS*、DTR*、OUT1、OUT2 连接，利用这个特点可以编程测试 8250 工作是否正常，从环回测试转到正常工作状态，必须对 8250 重新初始化;D3\D2：D3 为 1 表示使 OUT2 输出 0，D2 为 1 表示使 OUT1 输出 0;D1：1 表示 8250 的RTS输出为低电平，表示 8250 准备发送数据;D0：1 表示 8250 的DTR输出为低电平，表示 8250 准备接收数据

⑨Modem 状态寄存器（MSR）(BASE+6)

用在有 Modem 的系统中了解 Modem 控制线的当前状态。

RLSD：1 表示接收线检测到载波检出信号;**RI**：1 表示有振铃指示;**DSR**：1 表示数据就绪;**CTS**：1 表示发送数据

3）8250 的初始化编程

BASE+3 写线路控制寄存器（LCR）DLAB=1 设置波特率因子→**BASE** 写除数寄存器的低 8 位→**BASE+1** 写除数寄存器的高八位→**BASE+3** 写通信格式控制字(LCR)DLAB=0→**BASE+4** 写 Modem 控制字（MCR）→**BASE+1** 设置中断允许字（IER）

①设置工作方式：

无中断方式：MOV AL, 03H || MOV DX,3FCH || OUT DX, AL

中断方式：MOV AL, 0BH || MOV DX,3FCH || OUT DX, AL，设置 OUT2 位为 1， OUT2 * 信号有效，使能三态门，INT 信号可通过三态门向 CPU 申请中断。

循环自测：MOV AL, 13H || MOV DX,3FCH || OUT DX, AL

以首地址 BASE=3F8H 为例，则 3FCH 对应 MODEM 控制寄存器(MCR)的地址

②设置中断允许位：

禁止所有中断 MOV AL, 0H || MOV DX,3F9H || OUT DX, AL

允许所有中断 MOV AL, 0FH || MOV DX,3F9H || OUT DX, AL

允许收发中断 MOV AL, 03H || MOV DX,3F9H || OUT DX, AL

4）8250 通信编程

①查询方式（详见举例）

②中断方式

5）8250 编程举例：

假设 8250 的端口地址是 3F8H~3FEH

1. 初始化：

(1)设置波特率（假设波特率为 9600）

查表 7.6 或计算波特率因子: 1.8432M/(16*9600)=12

```
MOV   DX,    3FBH    ; LCR
MOV   AL,    80H
OUT   DX,    AL      ; D7(DLAB)←1
MOV   DX,    3F8H    ;设置除数低 8 位
MOV   AL,    12
OUT   DX,    AL
INC   DX             ;设置除数高 8 位
MOV   AL,    0
OUT   DX,    AL
```

(2) 设置串行通信数据格式

假设 数据格式为 8 位数据位，1 位停止位，奇校验。

```
MOV   AL,    00001011B
MOV   DX,    3FBH    ;LCR
OUT   DX,    AL
```

(3) 设置工作方式

正常通信，不用中断方式：

```
MOV   DX,    3FCH    ; MCR
MOV   AL,    3       ; OUT2=0
OUT   DX,    AL
```

正常通信，用中断方式：

```
MOV   DX,    3FCH    ; MCR
MOV   AL,    0BH     ; OUT2=1
OUT   DX,    AL
```

循环测试(自发自收)，不用中断方式：

```
MOV   DX,    3FCH    ; MCR
MOV   AL,    13H     ; LOOP=1
OUT   DX,    AL
```

循环测试(自发自收)，用中断方式：

```
MOV   DX,    3FCH    ; MCR
MOV   AL,    1BH     ; LOOP=1,  OUT2=1
OUT   DX,    AL
```

2、查询方式通信程序

发送程序：

查询 LSR(3FDH)中 D5 位，检查发送保持器是否空(D5=1?)

```
TR:   MOV   DX,   3FDH
      IN    AL,   DX     ;读取 LSR 的值
      TEST  AL,   20H    ;测试 D5 位
      JZ    TR           ;D5=0, 发送保持器不空，转 TR
      MOV   AL,   [SI]   ;D5=1, 发送保持器空, 发下一数据
      MOV   DX,   3F8H
      OUT   DX,   AL     ;发送下一个数据
```

接收程序：

查询 LSR(3FDH)中 D0 位，检查接收数据是否就绪（D0=1?）。

```
RE:   MOV   DX,   3FDH
      IN    AL,   DX     ;读取 LSR 的值
      TEST  AL,   01H    ;测试 D0 位
      JZ    RE           ; D0=0, 接收缓冲器不满, 转 RE
      MOV   DX,   3F8H   ; D0=1, 接收缓冲器满
      IN    AL,   DX     ;读数据
      MOV   [DI],  AL
```

3、用中断方式编程

1）设置 8259 的屏蔽字（OCW1），允许串口中断 IRQ4

```
IN    AL,   21H
AND   AL,   11101111B
OUT   21H,  AL
```

2）设置中断向量 IRQ4

PC 机 8259 的中断类型码：08H~0FH

串口中断 IRQ4 中断类型号: 08H+4 = 0CH

0CH×4=30H

中断向量 IP 值→[30H][31H]，CS 值→ [32H][33H]

假设中断服务程序入口处的标号为 “MY_INTR” 。

* 直接写内存方式设置中断向量

```
CLI
XOR   AX, AX
MOV   DS,  AX
MOV   AX,     OFFSET  MY_INTR
MOV   WORD  PTR [0030H], AX      ;送偏移地址
MOV   AX,     SEG  MY_INTR
MOV  WORD  PTR  [0032H], AX      ;送段基址
STI
```

* 采用 DOS 中断调用设置中断向量:

```
CLI
MOV   AX,    SEG  MY_INTR
MOV  DS,     AX
MOV  DX,     OFFSET  MY_INTR
MOV  AL,     0CH
MOV  AH,     25H
INT   21H
STI
```

3）设置 8250 的中断允许寄存器 IER (3F9H)

例如，允许发送与接收中断请求。

```
MOV   AL, 3
MOV   DX, 3F9H    ; IER
OUT   DX, AL
```

4）在中断服务程序中查询 8250 的中断识别寄存器 IIR (3FAH)

```
MY_INTR:  MOV   DX, 3FAH
          IN    AL,   DX       ;读 IIR
          TEST  AL,   02H      ;测试 D1 位(发送中断)
          JZ    RCVE           ;发送中断位=0,
                                                    ;则转 “RCVE” 处接收
SEND:     MOV   AL,   [SI]     ;否则，发送数据
          MOV   DX,   3F8H
          OUT   DX,   AL
          INC   SI
          JMP   MY_IRET
RCVE:     MOV   DX,   3F8H   ; 接收数据
          IN    AL,   DX
          MOV   [DI],  AL
          INC   DI
MY_IRET:  MOV   AL,  20H
          OUT   20H,  AL    ; 发 EOI 命令
          IRET                ; 中断返回
```

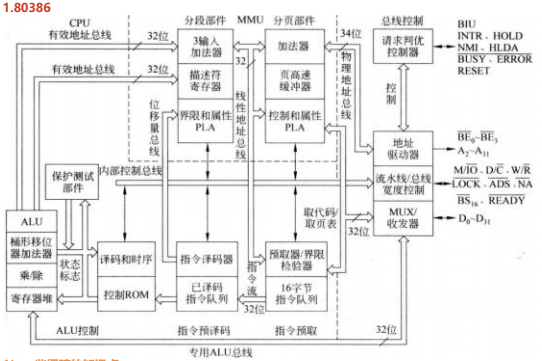
6)8250 编程应用举例

按**查询方式**编制一个发送与接收程序，能将键盘输入的每一个 ASCII 码字符发送出去，并显示在屏幕上，同时能把接收到的每一个字符也显示出来。假设：数据传输速率为 9600 波特，通信格式为 8 位数据位，1 位停止位，奇校验。

```
KEY:      MOV   DX,    3FBH   ; LCR D7 位置 1
          MOV   AL,    80h      ;准备设置波特率除数
          OUT   DX,    AL
          MOV   DX,    3F8H   ;设波特率除数,9600 波特
          MOV   AL,    12          ;低 8 位 (此处的 12 为十进制)
          OUT   DX,    AL
          INC   DX
          MOV   AL,    0           ;高 8 位 (设置除数寄存器高 8 位)
          OUT   DX,    AL
          MOV   AL, 0BH    ;8 位数据位，1 位停止位，奇校验
          MOV   DX, 3FBH
          OUT   DX,  AL     ;写 LCR，设通信格式
          MOV   AL, 03H     ;LOOP=0, 正常收发
          MOV   DX, 3FCH
          OUT   DX,  AL     ;写 MCR，设置工作方式
CHECK:    MOV   DX, 3FDH  ;读 LSR
          IN    AL,  DX
          TEST  AL, 1       ;查接收缓冲器是否满?
          JNZ   REV         ;若满，转 REV
          TEST  AL, 20H   ;查发送保持器是否空?
          JZ    CHECK     ;若不空，继续查
                          ; 发送数据
TR:       MOV   AH, 1           ;读键盘, 若有按键, AL=ASCII 码
          INT   16H             ;若无按键, ZF=1
          JZ    CHECK       ;无键按下, 转 CHECK
          MOV   DX, 3F8H   ;有键入，发送 ASCII 码
          OUT   DX, AL
          JMP   CHECK
REV:      MOV   DX, 3F8H
          IN    AL,  DX       ; 读入接收字符
          AND   AL, 7FH       ; 屏蔽 B7 位
          MOV   BX, 0041H   ; 第 0 页, 红底蓝字
          MOV   AH, 14
          INT   10H       ; 显示字符
          JMP   CHECK
```

第八章

1.80386



1）一些细碎的知识点

①80386 微处理器简介

★80386 是第一个全 32 位微处理器，其数据总线和内部数据通道都是 32 位，能灵活处理 8 位、16 位或 32 位 3 种数据类型。其地址总线也是 32 位，能寻址2^{32}字节（4GB）的物理存储空间。保护方式下，运用虚拟存储管理，能寻址2^{46}字节（64TB）虚拟存储空间（2（GDT\LDT）×2^{13}（GDT\LDT 表中最多有 2 的 13 次方个段描述符）×2^{32}（每个段最大长度为2^{32}B））。80386 的逻辑存储器采用分段结构，一个段最大可达 4GB

②六个基本单元

1. 总线接口单元（BIU） 2. 指令预取单元（CPU） 3. 指令译码部件（IDU） 4. 执行单元（EU）包含：控制部件（Control Unit），保护测试部件（Protection Test Unit）数据处理部件（Data Unit）
5. 段管理单元（SU） 6. 页管理单元（PU）。这 6 个单元既能各自独立操作，又能与其他部件并行工作。这样，既可以同时对不同指令进行操作，又可以同时对同一指令的不同部分并行操作。

③存储器的管理部件：段管理单元、页管理单元

★**段管理单元**：实现对逻辑地址空间的管理。把由指令指定的逻辑地址(虚拟地址)变换成线性地址。

★**页管理单元**：提供了对物理地址空间的管理。把由段管理单元所产生的线性地址再变换成物理地址。有了物理地址后，才能访问存储器或输入/输出端口。

★**分页单元**将一个段转换为多个页面，每页大小固定为 4KB(磁盘一个扇区的大小)。

★在内存和磁盘之间进行地址映射时，系统都是以页为单位把磁盘的程序和数据转存到内存中的一个相对应的地址区间的。

★**分页是可选择的，如果不使用分页，线性地址就是物理地址。**

④80386 的特点：

1. 80386 芯片在硬件结构上由 6 个逻辑单元组成。它们按流水线方式工作，运行速度可达到 4 MIPS。
2. 80386 有三种方式：实方式、保护方式、V86 方式 。3. 硬件支持分段管理和分页管理，易于实现虚拟存储系统。4. 硬件支持多任务，一条指令可以完成任务切换。5. 特权保护机制，4 级特权级：0 级最优先，其次为 1，2 和 3 级。一般 0、1 和 2 级用于操作系统程序，3 级用于用户程序。除特权级检查外，每条指令执行期间，CPU 还要进行访问类型、内存越限等保护特性检查。在用户之间、用户和操作系统之间形成严格的隔离保护。

⑤保护：

★**386 支持两个主要的保护类型**：一类是不同任务之间的保护，它是通过给每一任务分配不同的虚拟地址空间，而每一任务有各自不同的虚拟地址—物理地址的转换映射，因而可实现任务之间的完全隔离。另一类是同一任务内的保护，定义 4 种特权级（特权级 0、1、2、3），特权级 0 最高，特权级 3 最低。每个段都有对应的特权级，访问段时需遵循相应的特权保护规则。

★**特权级种类**：1、请求特权级——RPL， RPL 为选择子的最低 2 位。2、描述符特权级——DPL，DPL 为描述符中属性字节的 b6、b5 位，3、当前特权级——CPL （了解）CPL = 当前正在执行的代码段的 DPL。4、输入输出特权级——IOPL（了解）

2）80386 的内部寄存器

①通用寄存器（80386 有 8 个 32 位的通用寄存器）

8 个寄存器分别命名为：EAX\EBX\ECX\EDX\ESI\EDI\EBP\ESP。每个寄存器可用于存放数据或地址值。AX\BX\CX\DX\SI\DI\BP\SP 可作为 16 位寄存器单独使用，AH\AL\BH\BL\CH\CL\DH\DL 可作 8 位寄存器单独使用。

②段寄存器和段描述符高速缓存器（6 个 16 位的段寄存器）

除了有和 80386 相同的 CS\SS\DS\ES 这 4 个段寄存器之外，增加了 FS 和 GS 两个新的数据段寄存器。386 工作在实方式：16 位段寄存器里存放的还是段基址，段寄存器的使用、存储器物理地址的计算类似于 8086。386 工作在保护方式：16 位段寄存器不再直接存放段基址，而是存放一个选择子(selector)，通过它查表得到段描述符，进而得到段基地址、段界限、段属性。

③指令指针和标志寄存器

32 位的指令指针（EIP），32 位标志寄存器（EFLAGS）。实模式时采用 16 位的指令指针 IP（EIP 低 16 位）。EFLAGS 的低 12 位与 8086 标志寄存器 FLAGS 完全相同。高 20 位中设置了 5 位新的标志。

VM: 1--V86 方式, NT: 1-- 任务嵌套

④系统地址寄存器和系统段寄存器

★**GDT**：全局描述符表 **GDTR**（48 位，32 位线性基地址+16 位界限值）：全局描述符表寄存器 **IDT**：中断描述符表 **IDTR**（48 位，32 位线性基地址+16 位界限值）：中断描述符表寄存器 **LDT**：局部描述符表 **LDTR**（16 位选择子）：局部描述符表寄存器 **TSS**：任务状态段 **TR**（16 位选择子）：任务寄存器

★**GDTR 与 GDT（只有 1 个）的关系**：GDTR 的 32 位基地址为 GDT 的起始地址，16 位界限为 GDT 的最大偏移地址，表中最多有 $2^{16}/8=2^{13}=8192$ 个描述符**做题时十六进制记得加 H**

例：如果 GDTR 的内容为 0021000001FFh，请给出 GDT 表的起始地址、结束地址、表的长度。表中放了多少个描述符？第 5 个描述符的地址范围？答：GDTR = 0021 0000 01FFH，表起始地址 = 0021 0000H，表结束地址 = 0021 01FFH，表长 = 01FFH+1 = 0200H，描述符个数 = 0200H÷8 = 40H = 64，第 5 个描述符的地址范围：0021 0000H + 5*8 = 00210028H ~ 0021002FH

★**IDTR 与 IDT 的关系**：IDTR 的 32 位基地址为 IDT 的起始地址，16 位界限为最大偏移地址，386 只支持最多 256 个中断描述符。

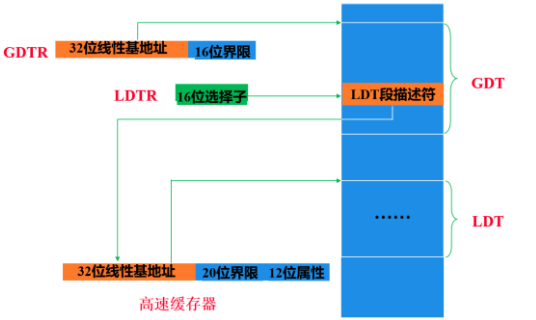
★**段选择子（放在段寄存器中）**：16 位段寄存器，其高 13 位是描述在描述符表中的索引号。

描述索引 n：指示所要读取的描述符在描述符表中的序号

TI：描述符表指示位。0：从 GDT（全局描述符表）中读取描述符，1：从 LDT（局部描述符表）中读取描述符。为加快查表和计算，CPU 里每个选择子有对应的描述符高速缓存

RPL：表示请求特权级。0-3 级，用于权限检查，以决定对该段能否进行访问。

★LDTR、**LDT** 与 **GDT** 的关系

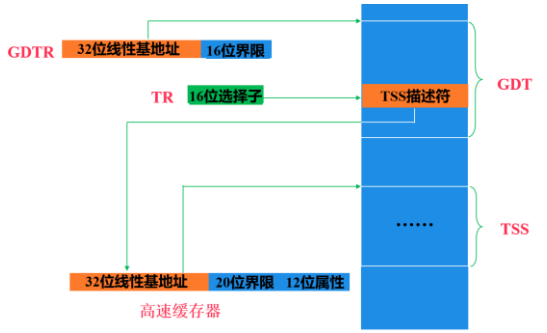


例：假设 GDT 的基地址为 01002000h，LDTR = 2108h，问 LDT 描述符的地址范围？

解：LDTR = 0010 0001 0000 1000 ，TI = 0, LDT 描述符在 GDT 中，索引 = 0010000100001，LDT 描述符的起始地址： **GDT 的基地址 + 索引×8**= 01002000h + 0010000100001 × 8= 01002000h + 2108H = 01004108h，LDT 描述符的地址范围（占 8 字节）： 01004108h ~ 01004140Fh

例：假设 GDT 的基地址为 00011000h，LDTR = 3208h，问 LDT 描述符的地址范围？答：LDTR = 0011001000001000 ，TI = 0, LDT 描述符在 GDT 中，索引 = 0011001000001，LDT 描述符的起始地址 = GDT 的基地址+索引×8= 00011000h + 0011001000001×8 = 00014208h，LDT 描述符的地址范围（占 8 字节）： 00014208h ~ 0001420Fh

★TR、TSS 与 **GDT** 的关系



⑤控制寄存器

32 位的控制寄存器 CR0、CR2、CR3 保存各种全局性状态。CR0 是最常用的一个控制寄存器，选择工作模式。CR3 为页目录基址寄存器，高 20 位为页目录表基址的高 20 位，存放页目录表的物理基地址。工作模式选择设置 PG\PE：00 代表实模式，01 代表分段不分页的保护模式，11 代表分段分页的保护模式

⑥调试寄存器组

⑦测试寄存器组

3）80386 的工作模式

①实地址模式（简称实模式）

当 80386 在刚加电启动或复位时，便由操作系统自动控制进入实模式。实模式主要是为 80386 进行初始化用的，为 80386 保护模式所需要的数据结构做好各种配置和准备。在实模式下，80386 类似于 8086 的体系结构，其寻址方式、中断处理方式等都与 8086 兼容。从编程角度看：可以用 32 位寄存器进行编程，加快了执行速度；增加了两个段寄存器 FS 和 GS，可以访问的段达到 6 个；新增指令可简化一些操作。

②保护虚拟地址模式（简称保护模式）

★**保护模式是 80386 最常用的、也是最具特色的工作模式。**通常在开机或复位后，先进入实模式完成初始化，然后就立即切换到保护模式。保护模式提供了多任务环境中的各种复杂功能以及对复杂存储器组织的管理机制： 1）多用户、多任务； 2）虚拟存储管理；3）保护机制：特权保护机制、不同任务之间的隔离(不同的虚拟存储空间)

★**虚拟存储空间**：80386 系统中，由于内存容量的限制，不可能将所有的段都放在内存中，而必须将大部分段放在外部磁盘上，待执行到相关程序段时再调入内存，将暂时不执行的程序段调出内存。程序员编写程序时，其程序存入磁盘。这样，从程序员的角度来看，系统中似乎有一个容量很大、速度也相当快的虚拟存储器；其实，它并不是真正物理上的内存。386 物理存储空间： 2^{32}=4GB ；虚拟存储空间有多大？ 2^{46}=64TB

★**保护模式下的地址转换**： 1）分段：逻辑地址转换为线性地址，逻辑地址由段寄存器和偏移地址组成。段寄存器中存放的不再是段基址，而是选择子，通过选择子来选择描述符表中的描述符，才能从中得到真正的段基地址，通过分段机制将段基地址和偏移地址相加即可得到线性地址；2）分页：线性地址 转换为物理地址，分页可选，如果不分页，物理地址=线性地址

★**保护模式下的分页**：磁盘分扇区，虚拟存储空间被划分成页，OS 把物理内存分为页，这些分页只是 OS 为了管理方便，在逻辑上划分的一种数据结构。虚拟内存的页可以和物理内存的页对应起来，产生映射关系。

★**虚拟页**：程序运行前，它的数据和代码是在可执行文件中，运行时需要把可执行文件加载到内存。硬盘上的文件逻辑上划分为 4K 的页（实际由加载器加载到内存），这就是虚拟页里面实际的东西。但是程序在运行时可能会申请内存，这时需新的虚拟页来映射。

★**虚拟页的 3 种状态**：已映射、未映射、未创建

★**存储器映射**：启动一个进程时，加载器完成文件映射的功能，所以执行文件又可称为映像文件。实际上加载器并不真正负责从磁盘加载数据到内存，因为和虚拟内存建立了映射关系，操作是虚拟内存自动进行的。正是有了存储器映射的存在，使得我们可以很方便的将程序和数据加载到内存中。

★**80386 的特权机制**：80386 为系统设置了 4 个特权等级，其取值由 0-3,0 为最高特权级。

★**保护模式下的特权规则**：1、高特权码段可访问低特权码段中的数据 2、低特权码段可调用高特权码段。

★**分段与分页**：分段：解决进程的虚拟地址空间的隔离，使程序员从复杂纷繁的存储器管理解脱，只需要关注程序设计，不用关心内存不够用。分页：解决物理内存的使用效率，与程序的执行密切相关，与程序员的设计无关。

★**段基地址由段寄存器间接获取**（还需要描述段长度、段属性（如访问权限等）），**偏移地址由指令中的寻址方式获取。**

③虚拟地址 8086 模式，简称 V86 模式

（1）使 80386 可以快速地执行多个 8086 的应用程序。（2）段寄存器的用法和实模式时相同。**（3）**存储器寻址空间为 1MB，可以使用分页机制，支持多任务。在多任务系统中，80386 可以使其中的一个或几个任务使用虚拟 8086 模式。此时，可以使一个任务所用的全部页面定位于某个物理地址空间，而另一个任务所用的页面定位于其他物理地址空间，这样，就把存储器虚拟化了，虚拟 8086 模式的名称即由此而来。**（4）**在虚拟 8086 模式中，程序是在最低特权级（3 级）上运行的，因此，80386 指令系统中的一些特权指令是不能使用的。

★**为了解决 80286 中不能在保护模式下运行 8086 / 8088 应用程序的问题，从 80386 开始，在保护模式中引入了虚拟 8086 工作模式（简称 V86 模式），操作系统可以并行执行 8086、80286 和 80386 的程序。V86 模式是让 80386 模拟 1MB 空间的寻址环境，但它并不仅限于 1MB 的存储空间，因为它可以同时支持几个 V86 环境。V86 模式下，多个 8086 实模式的应用软件可以同时运行。**

★**V86 模式的主要特点**：

1、V86 模式的寻址，在 V86 模式下，80386 对段寄存器的解释与处理方式和实模式一样。2、V86 模式下的保护功能，V86 模式具有 80386 的一些保护功能，但只是简单地把 V86 模式下任务的特权级一律降到最低级 3。3、V86 模式下的分页功能，为了使 80386 运行多个虚拟任务,地址空间大于 1 MB，需要启动分页机制，必须预先在保护模式下对 CR0 的 PG 位（位 31）置 1。V86 模式下分页时，线性地址的算法与实模式下完全相同。

④从保护模式下进入实模式

CPU 复位后进入实地址模式；CR0 的 PE 位置 1，可以使 CPU 从实地址模式切换到保护模式；EFLAG 的 VM 位置 1，可使 CPU 从保护模式转变到 V86 模式。实模式不能切换到 V86 模式。

4）80386 的存储管理机制

①简介

★80386 的存储器管理部件由分段单元和分页单元组成，实现存储器的段、页式管理。为了实现内存分段管理和保护功能，每个段都有一个描述符，段描述符中含有对应段的基地址、段界限和段的属性等信息，各种段描述符由操作系统建立，并组成描述符表。可以利用选择子查表

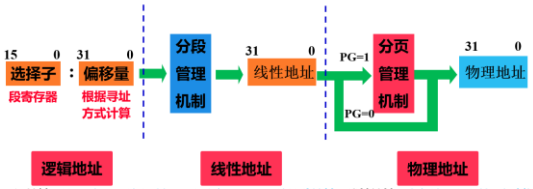
★80386 利用片内的存储管理单元（Memory Management Unit,简称 MMU ）来实现对存储器系统的两级管理：分段管理（逻辑地址→线性地址）分页管理（线性地址→物理地址）。在两级存储管理中，段的大小可以选择（以字节或页为单位），根据数据结构和代码模块的大小而定。另外，对每个段还可以赋予访问权和保护信息，以有效防止在多任务环境下各个任务对存储器的越权访问。

★**实模式：** 80386 的所有指令在实模式下都有效。其物理地址的形成与 8086 相同,可寻址的实地址空间只有 1MB，所有的段最大容量为 64KB。中断向量表仍设置在 00000H～003FFH 共计 1K 字节的存储区内。系统初始化区在 FFFFFFF0H～FFFFFFFFH 存储区内。

★**保护模式：** 保护模式下采用段页式存储管理机制。

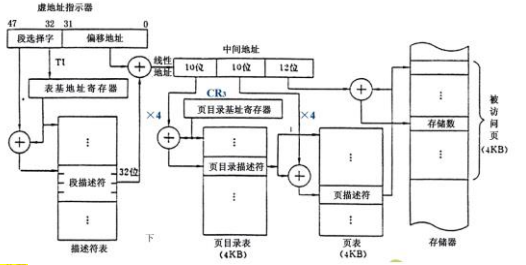
00	00
A2	10
00	00
00	FF

6
4
2
0



逻辑地址：编程时，16 位段选择子(段寄存器) + 32 位偏移地址。**线性地址**：由存储器分段管理机制按下式来计算：线性地址 = 32 位段基地址(查描述符得到) + 32 位偏移地址。**物理地址**：不分页时，物理地址 = 线性地址；分页时，由存储器分页管理机制转换：物理地址 = F(线性地址)

★寻址过程：



②分段

★为了实现分段管理，80386 把有关段的信息存放在一个称为段描述符（简称描述符, 8 个字节）的数据结构中，并把系统中所有的描述符按类组成一张张描述符表，以便硬件查找和识别。80386 共设置了 3 种描述符表:全局描述符表 GDT，局部描述符表 LDT，中断描述符表 IDT。**两种类型的段：**非系统段、系统段。**两种段描述符：**非系统段描述符，系统段描述符。**非系统段：**代码段、数据段、堆栈段。**系统段：** LDT、TSS，各种门--- 任务门，调用门，中断门等

★段描述符包括：段基地址（Base Address），规定了线性地址空间中段的起始地址。也可以把基地址看成是段内偏移量为 0 的线性地址。段的界限 (Limit)，表示在虚拟地址中，段内可使用的最大偏移量。段的属性（Attributes），包括该段是否可读出、写入以及段的特权级等，也叫访问权字节。

★描述符表：存放描述符的特殊段。最大 64KB，最多存放 8192 个描述符。(描述符表分为全局描述符表 GDT（Global Descriptor Table）(存放被系统所有任务共享的描述符，如 LDT\TSS 描述符，全局唯一)、局部描述符表 LDT (每任务独有 LDT，存放与该任务相关的描述符, 如代码段和数据段描述符) 和中断描述符表 IDT（存放描述中断程序入口地址有关门描述符，最多 256 个描述符，全局唯一）)

GDTR：用来定义全局描述符表在存储器中的位置。寄存器 GDTR 中存放的是 GDT 在内存中的基地址和其表长界限。(前 32 位为线性基地址，后 16 位为界限)

例：如果 GDTR 的内容为 0021000001FFH,请输出 GDT 表的起始地址、结束地址、表的长度，表中放了多少个描述符？第 5 个描述符的地址范围？起始地址：0021 0000H 结束地址：0021 01FFH 表的长度：01FFH+1=0200H，放了 40H 个描述符（除以 8），第 5 个描述符的地址范围为 0021 0028-0021 002F

LDTR：用来定义局部描述符表在存储器中的位置，此位置需要用一个 16 位选择子来指定，LDTR 就是容纳该选择子的 16 位寄存器。

例：假设 GDT 的基地址为 0100 2000h，LDTR=2108h，问 LDT 描述符的地址范围？

LDTR=0010 0001 0000 1000，易知 TI=0，表示描述符在 GDT 中。描述符索引 n=0010 0001 0000 1，LDT 描述符的起始地址：**GDT 的基地址+索引×8** 即 0100 2000H+2108H=0100 4108H（**标明是 16 进制数还是二进制数！**）LDT 描述符的地址范围（占 8 字节）：0100 4108H-0100 410FH

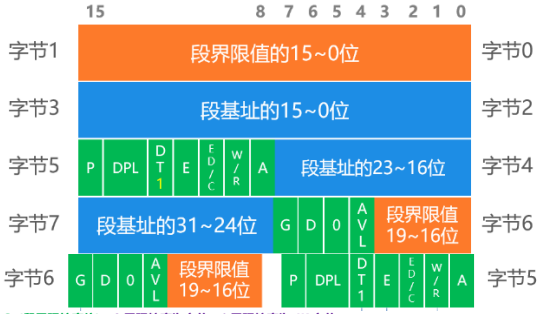
★分段单元实现对逻辑地址空间的管理，即把由指令指定的逻辑地址变换成线性地址。80386 在运行时，可以同时执行多任务操作。对每个任务来说，可以拥有多达 16K 段，因为每一段的最大空间可达 4GB，所以 80386 可为每个任务提供 64TB 的虚拟存储空间。

★有关特权的概念：

任务特权：任务当前特权级 CPL，由 CS 的最低两位来定；**选择子特权：**由选择子的 RPL 所确定；**描述符特权：**访问该描述符任务最低特权级；**有效特权级：**任务 EPL=max{RPL,CPL}，当 EPL<DPL 允许访问描述符。

★门：80386 允许低权限任务调用内核级代码，但是需要通过门调控。通过门调用可实现特权级的转变和任务切换。门描述符并不描述某种内存段，而是描述控制转移的入口点，即目标代码的门。

★存储段（非系统）描述符格式



G（段界限粒度位）：0 界限粒度为字节，1 界限粒度为 4K 字节

DT（表示描述符的类型）：1 存储段描述符，0 系统段描述符或门描述符

E（可执行位）：1 可执行，为代码段描述符，0 不可执行，为数据段描述符。

W\R（可读\写位）：对数据段：1 表示该段可读可写，0 表示可读不可写；对代码段：1 表示该段可执行可读，0 表示可执行不可读。

 P（段是否在内存）：1 该描述符所在描述段在内存，0 该描述符所描述段不在内存。

DPL（描述符特权级）：0-3 级

ED\C：对数据段：表示扩展方向位（0 向高，1 向低），对代码段：表示类型（1 依从，0 不依从）

A（访问位）:1 指示描述符被访问过，0 指示描述符未被访问过

例：请分析下面的描述符:

属性字节：B2h=1 01 10 0 1 0**DT（glzPPT 为 S）**=1：非系统段，E=0,ED=0 数据段，W=1 可写，DP1=1，P=1：在内存，A=0：未访问过。**段基址：**800A 00 00h，**界限** 03FFh，**段长：**3FFH+1=400H.段在内存的地址范围：800A 0000h-800A 03FFh

例：LDT 描述符：0000 8290 0000 FFFFh，从高位到低位排列，则 LDT 基址=0090 0000h，界限：0FFFFh

▲CS:

例：CS=1005h=0001 0000 0000 0101b，则代码段描述符的地址范围（占 8 字节）？TI=1，代码段描述符在 LDT 中，索引 0001 0000 0000 0，则代码段描述符的起始地址=LDT 起始地址+索引×8

▲存储段的段界：段界限决定段的寻址范围，它由描述符中 20 位段限长及 G 位得到。G=0 表示 32 位段限长的高 12 位为 0，其低 20 位是段描述符中 20 位段限，段最大寻址范围 1MB。G=1 表示段限长等于描述符中规定的 20 位限长左移 12 位+0FFFH

★LDT 描述符（属于系统段描述符，存放 LDT 的基地址、界限、属性等信息）(系统段包括局部描述符表 LDT、任务状态段 TSS 和各种门）(系统段描述符的特征为 DT=0)

例:属性字节 (字节 5)为 A2H=1010 0010B.则 DT=0 表示系统段，TYPE=2 为 LDT 描述符，DPL=1，P=1，描述符有效

▲门描述符：64 位的门描述符由 3 部分组成：16 位选择子、32 位偏移地址及 DPL

调用门描述符：用于特权级切换，存在与 GDT 或 LDT，其选择子指向代码，段偏移地址对应代码段中偏移量。

任务门描述符：在 GDT\LDT\IDT，用于任务切换，选择子指向 GDT 中的 TSS 描述符。偏移地址无意义

中断门和陷阱门描述符：存在于 IDT，中断处理程序入口，选择子指向代码段选择符、偏移地址指向代码段中。

★有关段描述符的大例题： 80386 工作在保护模式下,访问代码段、数据段,请回答下面问题:

1) GDTR=00A00000FFFFh，GDT 的起始地址 = ？ GDT 的结束地址 = ?答：GDT 的起始地址 = 00A00000h，GDT 的结束地址 = 00A00000h+FFFFh = 00A0FFFFh 2) LDTR = 2000h，LDT 描述符的地址范围（占 8 字节）：？答：LDTR = 2000h = 0010000000000000b，TI = 0，LDT 描述符在 GDT 中，索引 = 0010000000000，LDT 描述符的起始地址：GDT 起始地址＋索引×8= 00A00000h + 2000h= 00A02000h，LDT 描述符的地址范围(占 8 字节): 00A02000h ~ 00A02007h。3) LDT 描述符：0000,8290,0000,FFFFh，LDT 基址 = ？，界限 = ？答：LDT 基址 = 00900000h，界限 = 0FFFFh。4) CS = 1005h = 0001000000000101b，TI = 1，代码段描述符在 LDT 中，索引 = 0001000000000，代码段描述符的起始地址：LDT 起始地址＋索引×8= 00900000h + 1000h= 00901000h，代码段描述符的地址范围(占 8 字节):00901000h ~ 00901007h。5) 代码段描述符：000F,FA60,0000,FFFFh，代码段基址 = ？，界限 = ？分析属性：？答：代码段基址 = 00600000h，界限 = 0FFFFFh。属性：FAh = 11111010b 代码段，可读，DPL=3，在内存中，未访问过。6) 假设 EIP = 00012000h，CR0 的 PG 位= 0（不分页）则当前要取的指令在内存中的物理地址（起始的地址）=？答：当前要取的指令在内存中的物理地址（起始的地址）= 代码段基址＋偏移地址(EIP) = 00600000h＋00012000h = 00612000h。注：如果 CR0 的 PG 位= 1（分页），则上面求出的 00612000h 就不是物理地址，而是线性地址，需要进一步分页转换。

7）现在要访问一个数据段，数据段描述符为：0001,F220,0000,FFFFh，该数据段 基址 = ？ 界限 = ？，分析属性：？答：该数据段 基址 = 00200000h，界限 = 1FFFFh，属性：F2h = 11110010b 数据段，可写，DPL = 3，在内存，未访问过。8）如果在代码段中执行下面一段程序，程序中访问的数据段，描述符如 7)所示。

```
MOV AX, 1005H      ;访问数据段的选择子
MOV DS, AX
MOV EBX, 55667788H
MOV EAX, [EBX]     ;基址寻址
```

如果 CR0 的 PG 位= 0（不分页）那么，要读入 EAX 中的数据在内存的物理地址范围(4B)=?

答：要读入 EAX 中的数据在内存的物理地址范围(4B) = 数据段基址＋偏移地址= 00200000h＋55667788h = 55867788h ~ 5586778Bh

9）如果在代码段中执行下面一段程序，程序中访问的数据段，描述符如 7)所示。

```
MOV AX, 1005H      ;访问数据段的选择子
MOV DS, AX
MOV EBX, 55667788H
MOV ESI, 10000000H
MOV EAX, [EBX+ESI]  ;基址加变址寻址
```

如果 CR0 的 PG 位= 0（不分页），

那么，要读入 EAX 中的数据在内存的物理地址范围(4B)=?

答：要读入 EAX 中的数据在内存的物理地址范围(4B) = 数据段基址＋偏移地址= 00200000h＋(55667788h+10000000h) = 65867788h ~ 6586778Bh

③分页

★当 CR0：实模式下，PE 位置 1，可切换到保护模式；保护模式下，PG 位为 0，存储管理仅分段，不分页，PG 位置 1，启动分页机制，存储管理既分段又分页。分页管理的对象是 4KB 的存储块，称之为页。

★分页采用了两级表结构：页目录表和页表。分页机制实现两级地址转换：在较高一级，由页目录表中的页目录项(页目录描述符)映射页表；在低一级，由页表中的页表项(页描述符)映射页帧。页目录表、页表和页帧都是按页存放（4KB），**在内存的起始地址的低 12 位为 0。**

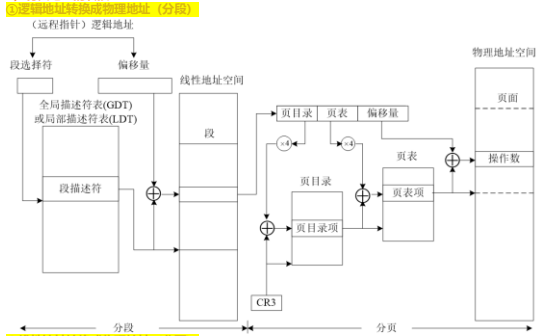
★分页单元提供了对物理地址空间的管理，它把由分段单元所产生的线性地址在换算成物理地址，并实现程序的重定位。

★有了物理地址后，总线接口单元就可以访问存储器或输入/输出端口。分页单元在将一个段转换为多个页面时，其页面大小可定义为 1B 到 4KB，但为了页定位的方便起见，将每一页的容量固定设为 4KB，它正好是磁盘按 4KB 大小来划分的一个扇区。通常，在内存和磁盘之间进行地址映射时，系统都是以页为单位把磁盘的程序和数据转存到内存中的一个相对应的地址区间的。

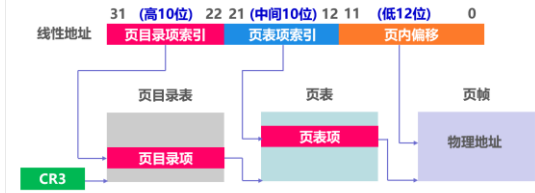
★页单元是可以选择的，如果不使用它，线性地址就是物理地址。

5）有关地址的转换

①逻辑地址转换成物理地址（分段）



②线性地址转换成物理地址（分页）



页目录基址寄存器

▲页目录项索引：线性地址高 10 位，查页目录表的索引。页表项索引：线性地址中间 10 位，查页表的索引。页内偏移量：线性地址低 12 位，页帧内的偏移地址

▲页目录项（页目录描述符）**和页表项**（页描述符）

页表/页 基地址(高20位)	AVL	00	D	A	00	U	W	P

D：1 出错，0 未出错。A：1 已访问过，0 未访问过。U：1 用户可用，0 用户不可用。W：1 表示页\页表可写，0 表示页\页表不可写（当页目录项和页表项的 U、W 位不一致时，选取最小的值，例如当页目录项的 UW 位为 11（用户可写），而页表项的 UW 为 10（用户可用，但只可读不可写），最终 UW 位为 10）。P：1 表示页\页表在内存，0 表示页\页表不在内存。AVL：程序员可使用的位。

▲80386 按页使用内存，每个页 4KB，索引该页的地址范围是：0-2^{12}-1。4GB 的线性地址空间可以分为2^{20}个页

▲页的基地址具有“XXXXX000H”的形式，因此其高 20 位被称为页码。页码左移 12 位可得页物理基地址。

▲页为 4K 字节且页边界是 4K 的倍数，所以把 32 位线性地址转换成 32 位物理地址的过程中，低 12 位地址保持不变。

▲两级页表：

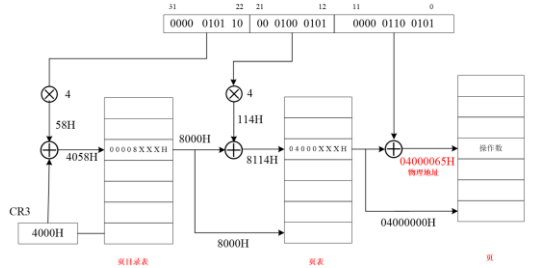
每表格 4KB，CR3 放页目录表基地址，页目录项放页表基地址，页表项放页基地址。系统为每个进程分配一个页目录表，只有被使用的页表才被分配到内存。

▲例：

页目录项索引号：线性地址域第 31-22 共 10 位，页目录中页目录项的索引。页表项索引号：线性地址域第 21-12 共 10 位，页表中页表项的索引。页内偏移量：线性地址域第 11-0 共 12 位，寻址页内地址。

例：

假设线性地址 = 05845065H，CR3 = 4000H，求转换的物理地址。

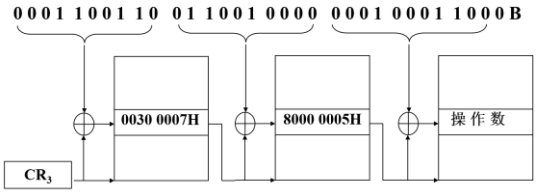


页目录表：00004000h~00004FFFh，页表：00008000h~00008FFFh，页帧：04000000~04000FFFh

例：

已知在80386系统中，**CR3 = 0010 0000H**，页目录项和页表项内容如下图中所示，假设分段转换得到的线性地址为 **1999 0118H**，请回答下面问题。

线性地址：



页目录表地址范围：0010 0000H~0010 0FFFH（页目录表的大小为 4KB，即 3 个 F）页目录项的地址范围：0010 0198H~0010 019BH（看线性地址前 10 位×4，页目录项占 32 比 特 4 字节）页表的地址范围：0030 0000H~0030 0FFFH（页表的大小为 4KB）页表项的地址范围：0030 0640H~0030 0643H（看线性地址中间 10 位×4，页表项占 32 比特 4 字节）页帧的地址范围：8000 0000H~8000 0FFFH（页帧的大小为 4KB）被访问操作数 (32 位) 的地址范围：8000 0118H~8000 011BH（被访问操作数的占 32 比特 4 字节）

③逻辑地址转换为线性地址

将段选择子指向的段描述符中 32 位基地址与 32 位偏移地址相加即得线性地址。

6）例题

例题 1

80386工作在保护方式下，GDTR=0020,0000,7FFFH,

CS=4002H, 代码段描述符和数据段描述符如下：

代码段描述符

00	03
DA	60
00	00
FF	FF

数据段描述符

00	01
F2	70
00	00
0F	FF

GDT 的起始地址为：0020 0000 结束地址为：0020 7FFF（GDT 前 32 位代表 GDT 线性基地址，后 16 位代表界限）代码段描述符的起始地址为：0020 4000（CS 代表代码段，DS 代表数据段）结束地址为：0020 4007 代码段的起始地址为：0060 0000（存储段描述符格式的蓝色部分）结束地址为：0063 FFFF（60+03,0000+FFFF）

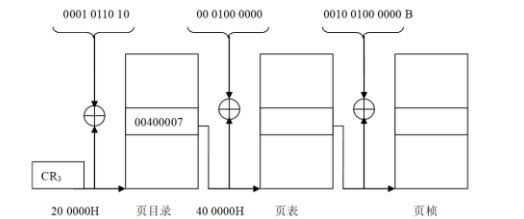
数据段的起始地址为：0070 0000 结束地址为：0071 0FFF

代码段：是否可读？可读，在内存中否？在内存，访问过否？未访问过

数据段：是否可写？可读，在内存中否？在内存，访问过否？未访问过

例题 2

已知在80386系统中，页目录起始地址为20 0000H，页表起始地址为40 0000H，页目录表内容如下图所示，已将线性地址1684 0240转换为物理地址7000 0240H。



CR3=0020 0000H 页表的偏移地址：0100H 页帧起始地址：7000 0000H 存放整个页目录表的内存地址范围：0020 0000-0020 0FFF（一个页 4KB）如果页表项所描述的页帧用户方式是只读，则该页表项内的 W\R=0

例题 3

MOV EAX,[EBX] 读入 EAX 中的数据在内存的物理地址首地址=数据段基址+偏移地址（即 EBX 的值）

其他细碎的知识点

1.IP 指示下一条要执行的指令地址

2.ASCII 码（十六进制）：ASCII 的 30-39 分别对应数字 0-9；ASCII 从 41 开始为 A，5A 为 Z（十六进制数 A 转换成 ASCII 形式的 A 需要加上 37）

★回车的 ASCII 码为 0DH，换行的 ASCII 码为 0AH。

注意事项

1、写数字记得表明进制。十六进制数如果是字母开头，前面要加个 0

2、涉及操作数地址时，如操作数为两字节或者四字节，需要将其占用的地址写全。不能仅写出最低位的首地址。