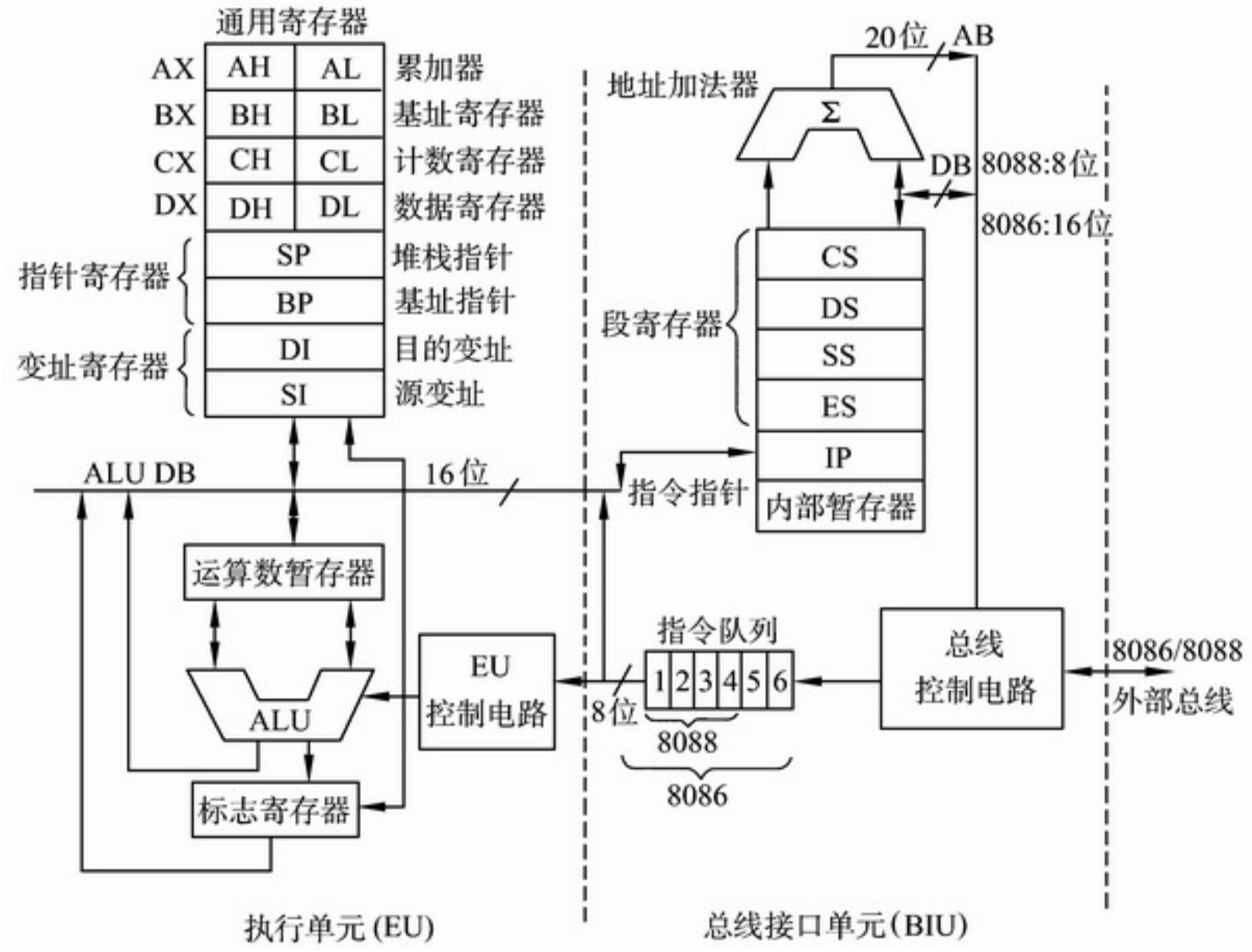
# 第2章 16位Intel 8086微处理器

- 1978年，Intel公司推出8086：
- 16位微处理器，兼容8085
- CISC结构
- 单一+5V供电
- 频率4.77MHz-10MHz
- 内部数据总线和外部数据总线都是16位，地址总线为20位，可最大寻址1MB的存储空间
- 双列直插式封装DIP，40个引脚
- 内存采用分段的管理方式。

- 1979年，推出成本更低的8088。
- 8088芯片最早于1981年用于IBM PC/XT。

# 2.1 8086微处理器内部结构

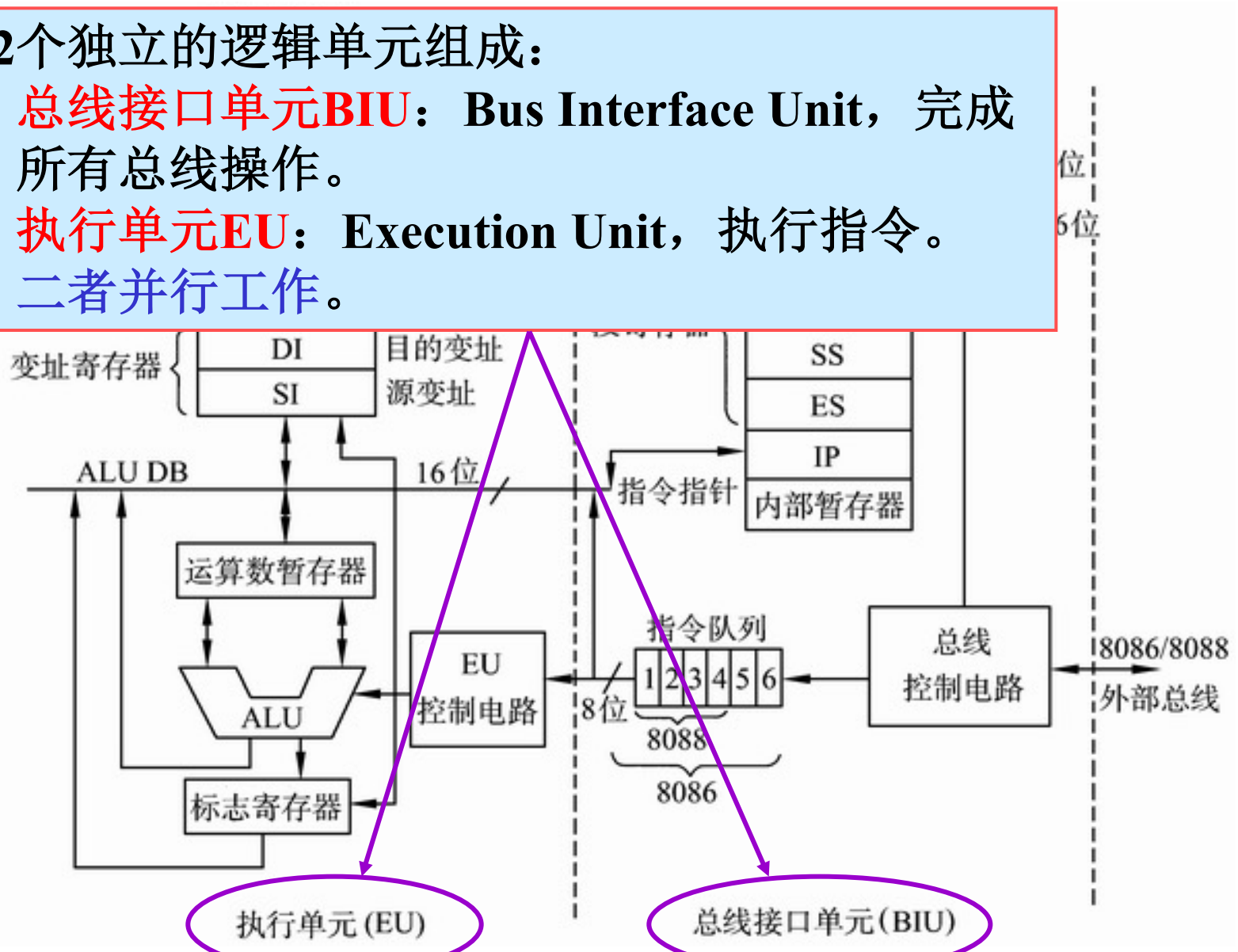


# 8086内部结构

由2个独立的逻辑单元组成：

1. 总线接口单元BIU：Bus Interface Unit，完成所有总线操作。
2. 执行单元EU：Execution Unit，执行指令。

➢ 二者并行工作。

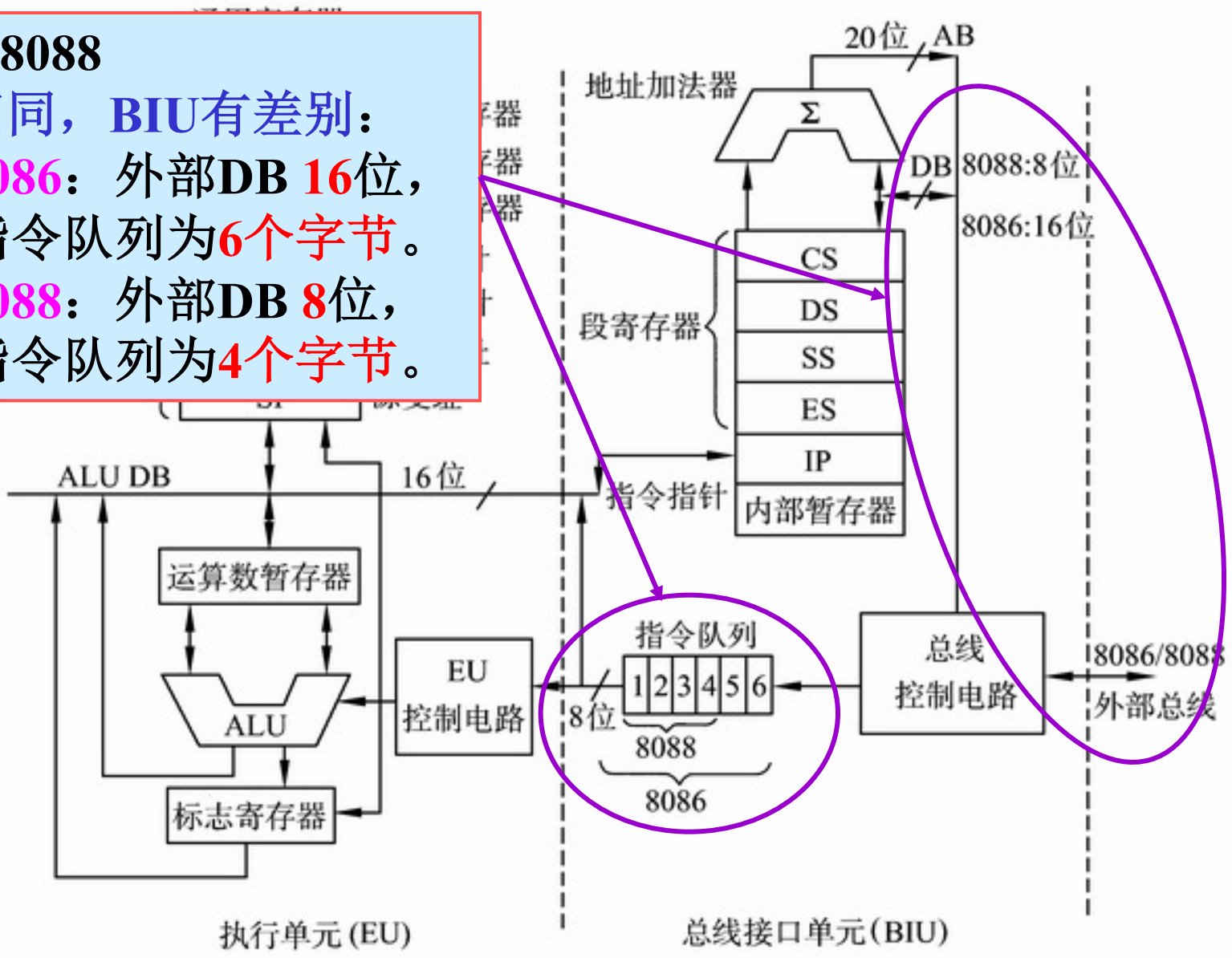


# 8086/8088差别

**8086/8088**

**EU相同，BIU有差别：**

1. **8086：外部DB 16位，指令队列为6个字节。**
2. **8088：外部DB 8位，指令队列为4个字节。**



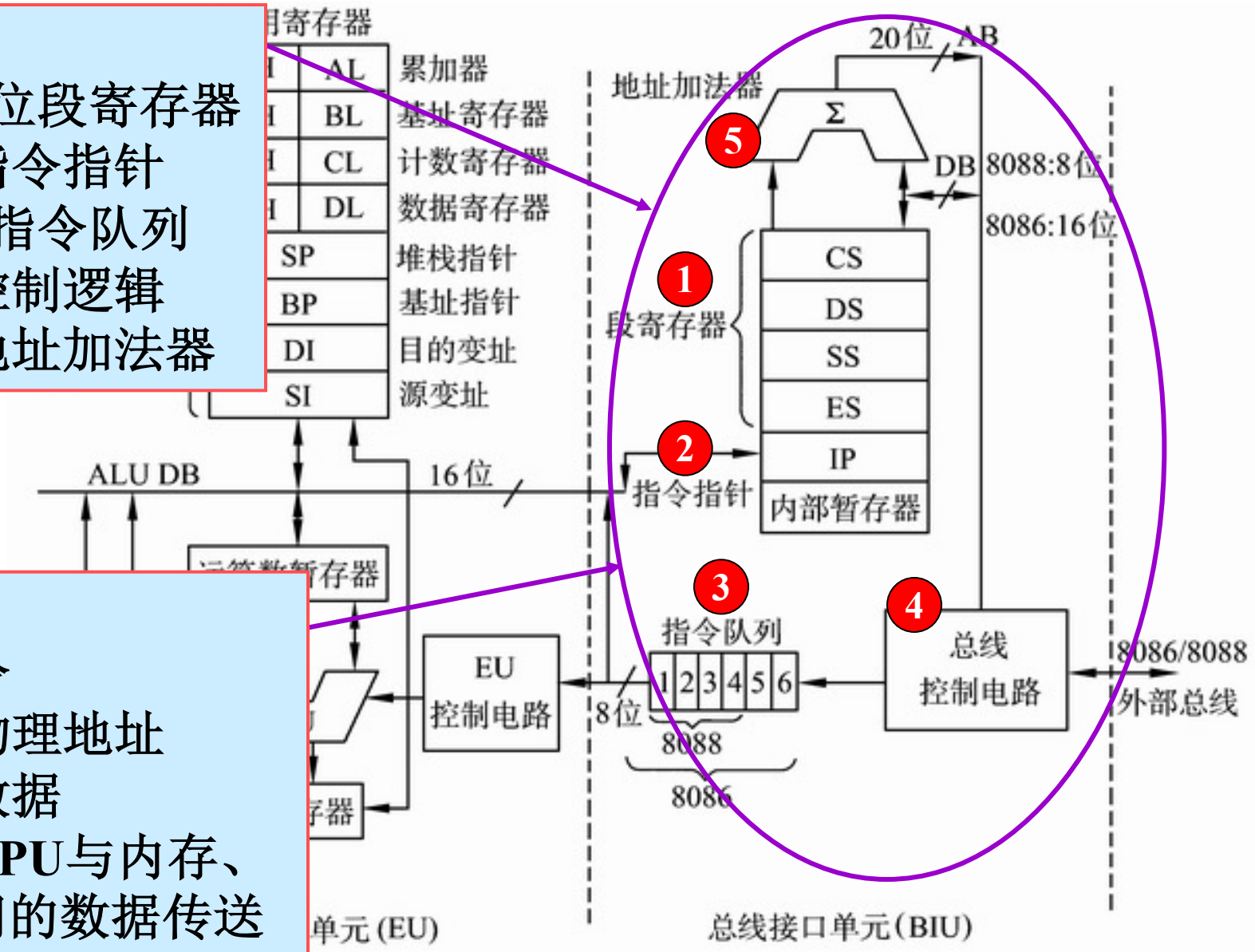
# 总线接口单元BIU

**BIU组成：**

1）4个16位段寄存器
2）16位指令指针
3）6字节指令队列
4）总线控制逻辑
5）20位地址加法器

**BIU功能：**

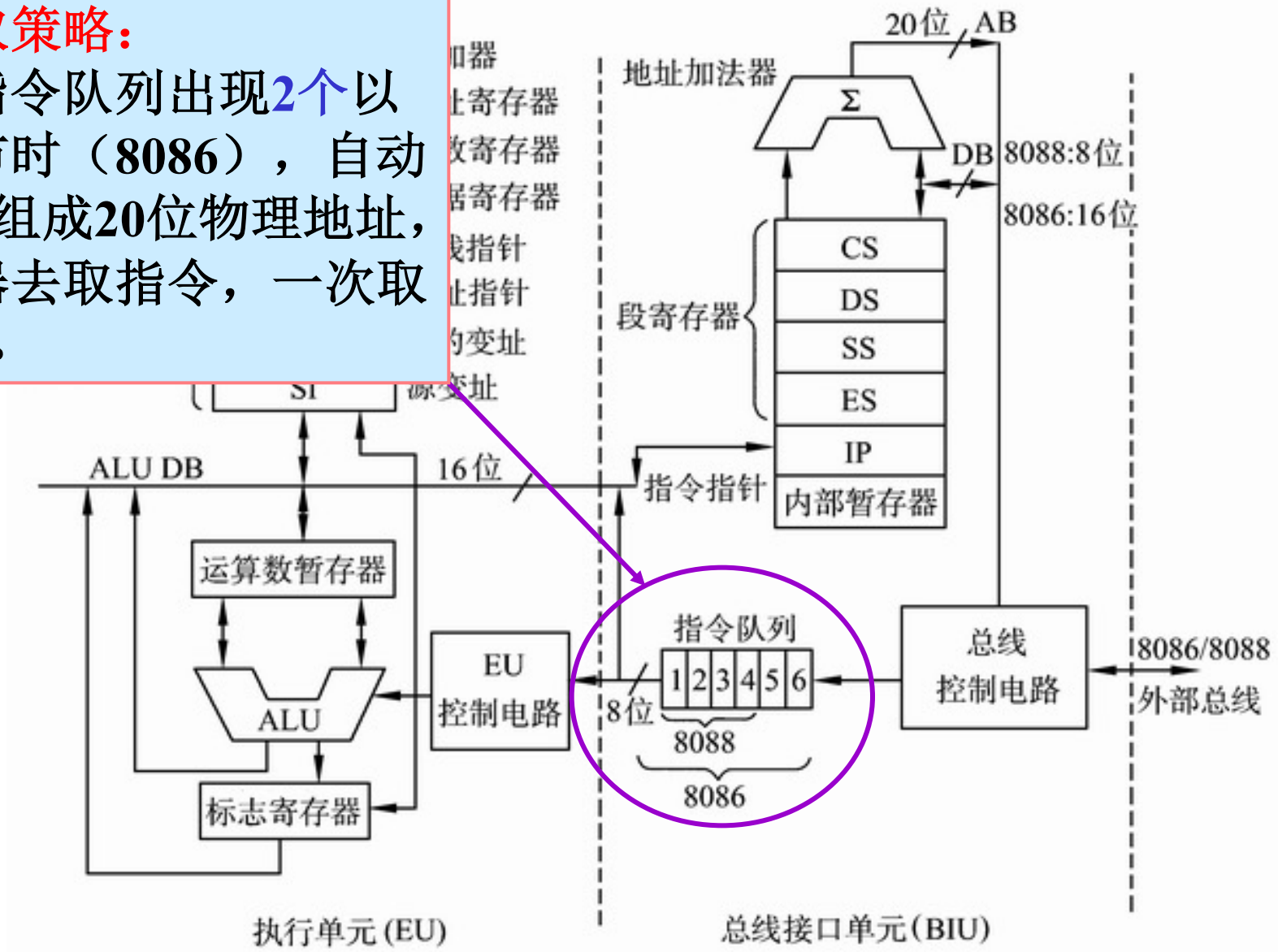
1）取指令
2）形成物理地址
3）传送数据
实现CPU与内存、I/O端口间的数据传送



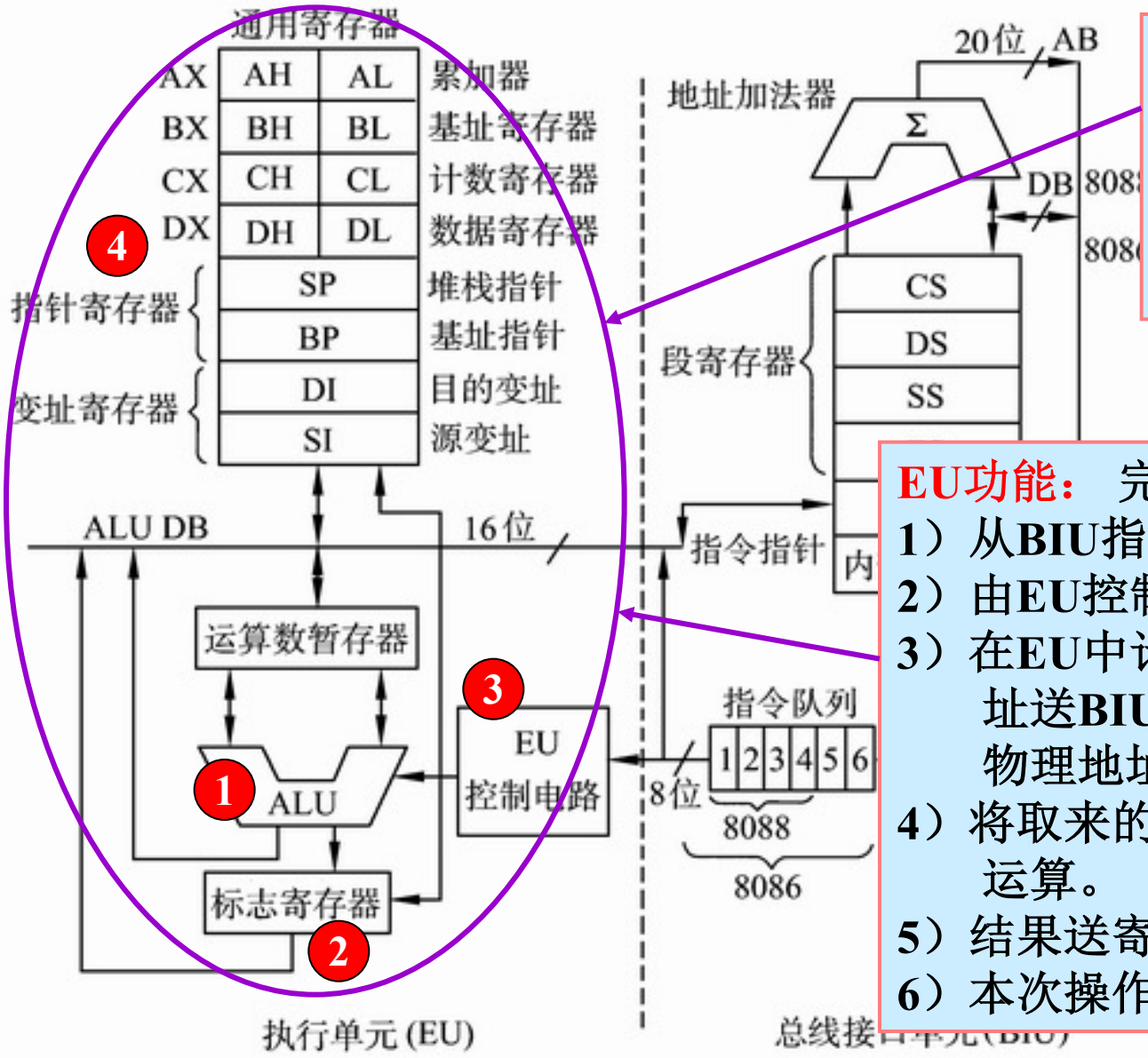
# 指令预取策略

**指令预取策略：**

当指令队列出现2个以上空字节时（8086），自动按CS:IP组成20位物理地址，到存储器去取指令，一次取2个字节。



# 执行单元EU




**EU组成：**

1）16位ALU
2）16位标志寄存器
3）EU控制电路
4）8个16位通用寄存器

**EU功能：** 完成指令所规定的操作

1）从BIU指令队列读取指令。
2）由EU控制电路译码分析。
3）在EU中计算操作数的16位偏移地址送BIU，由BIU的∑形成20位物理地址。
4）将取来的操作数送ALU进行指令运算。
5）结果送寄存器或送BIU放到内存
6）本次操作状态放到标志寄存器中

# 8086微处理器结构特点

**8086以前的处理器**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| CPU | 取指 1 | 执行 1 | 存结果 1 | 取指 2 | 执行 2 | 取指 3 | 取操作数 3 | 执行 3 |
| BUS | 忙 | 闲 | 忙 | 忙 | 闲 | 忙 | 忙 | 闲 |

**8086处理器**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| EU | | 执行 1 | 执行 2 | | 执行 3 | 执行 3 | 执行 4 | |
| BIU | 取指 1 | 取指 2 | 存结果 1 | 取指 3 | 取操作数 3 | 取指 4 | 存结果 3 | 取指 5 |
| BUS | 忙 | 忙 | 忙 | 忙 | 忙 | 忙 | 忙 | 忙 |

**8086内部由BIU、EU两个独立单元，可独立完成总线操作和执行指令的任务，即两个单元可重叠工作，总线利用率提高，降低存储器的存取速度要求，这种重叠技术以前只在大型机中使用。**

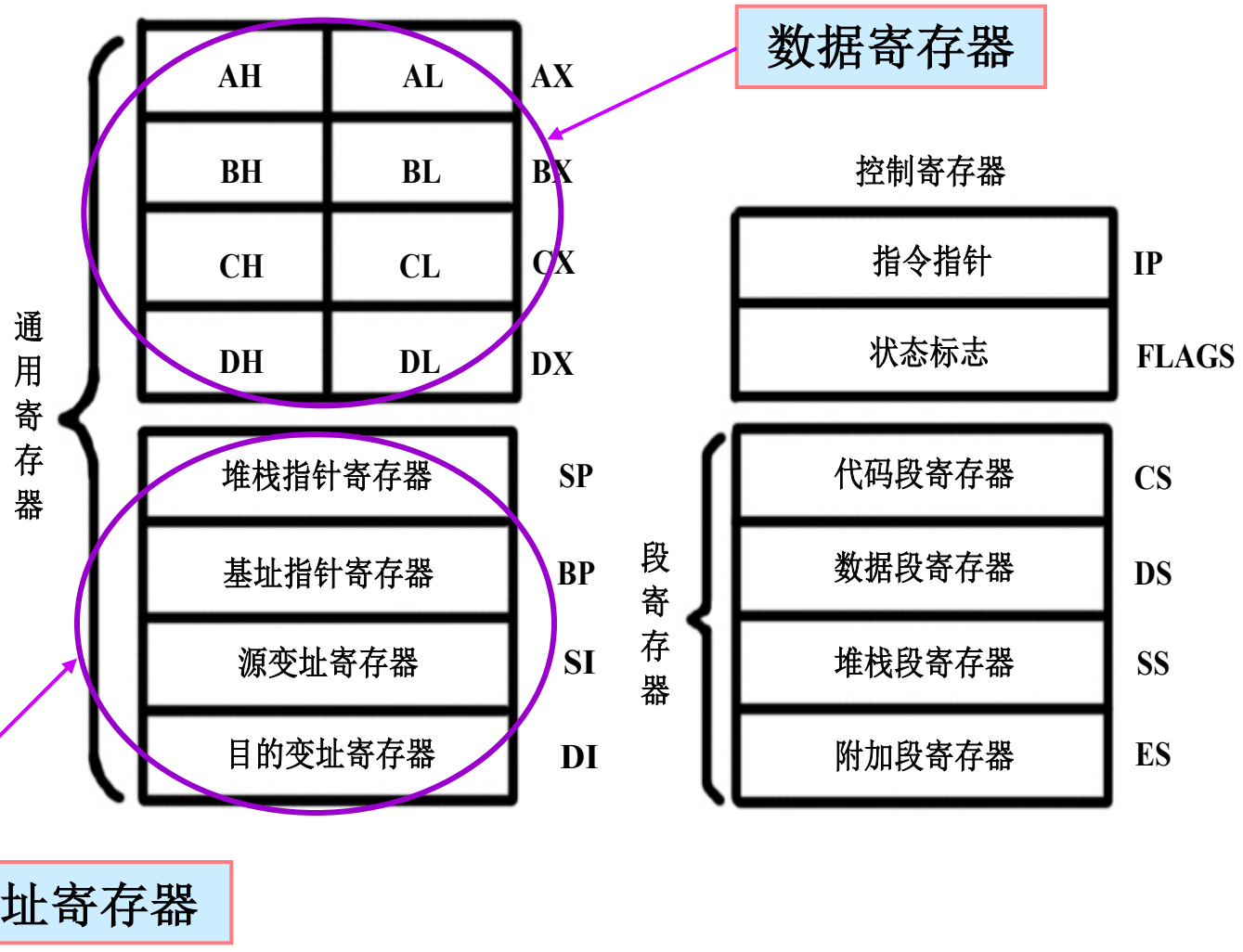
**8086结构特点：**

**1）由BIU、EU两个独立单元组成。**

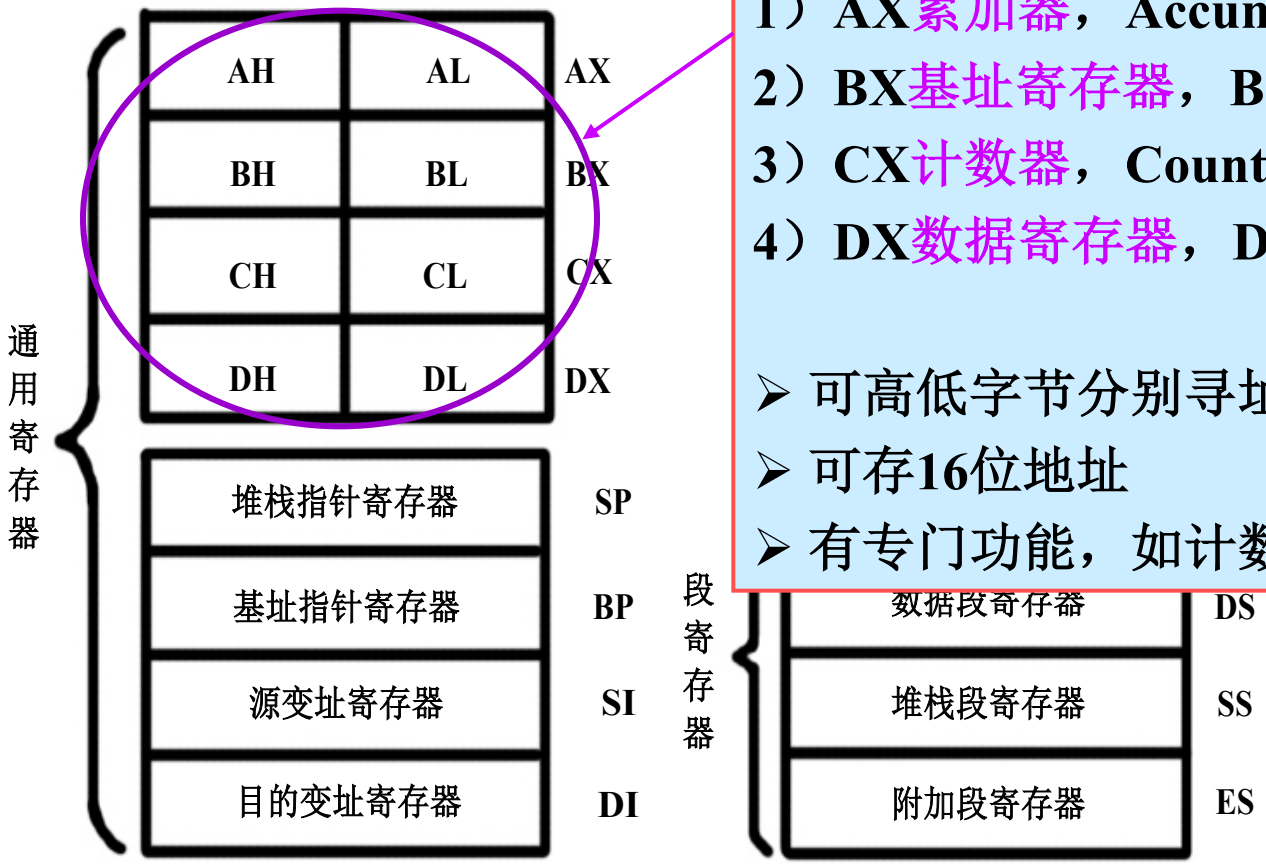
**2）取指和执行重叠。**

# 2.2 8086微处理器编程结构

**8086 CPU有14个16位寄存器**



# 数据寄存器




**数据寄存器：**存放操作数据及中间结果。

1）AX累加器，Accumulator Register

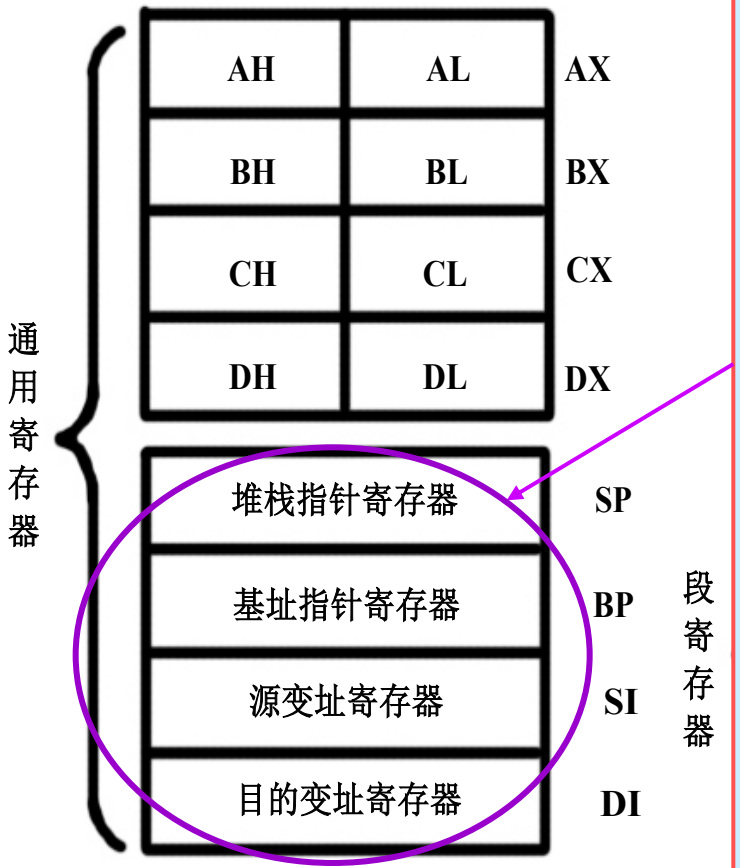
2）BX基址寄存器，Base Register

3）CX计数器，Counter

4）DX数据寄存器，DataRegister

- 可高低字节分别寻址
- 可存16位地址
- 有专门功能，如计数器、累加器等

# 指针及变址寄存器




**指针及变址寄存器：**存放逻辑地址的偏移量，是20位物理地址的组成部分

1） SP堆栈指针，Stack Pointer，指示在堆栈段中栈顶的位置，用于数据进栈、出栈的位置指向。

2） BP基址指针，Base Pointer，指示在堆栈段中一个数据区的基址位置，用于访问堆栈段中的某个数据。

3） SI源变址寄存器，Source Index，源串指针，指示在数据段中一个源串操作数的位置。

4） DI目的变址寄存器，Destination Index，目的串指针，指示在附加段中一个目的串操作数的位置。

➢ SI、DI用于数据串操作，且隐含使用。

# 段寄存器

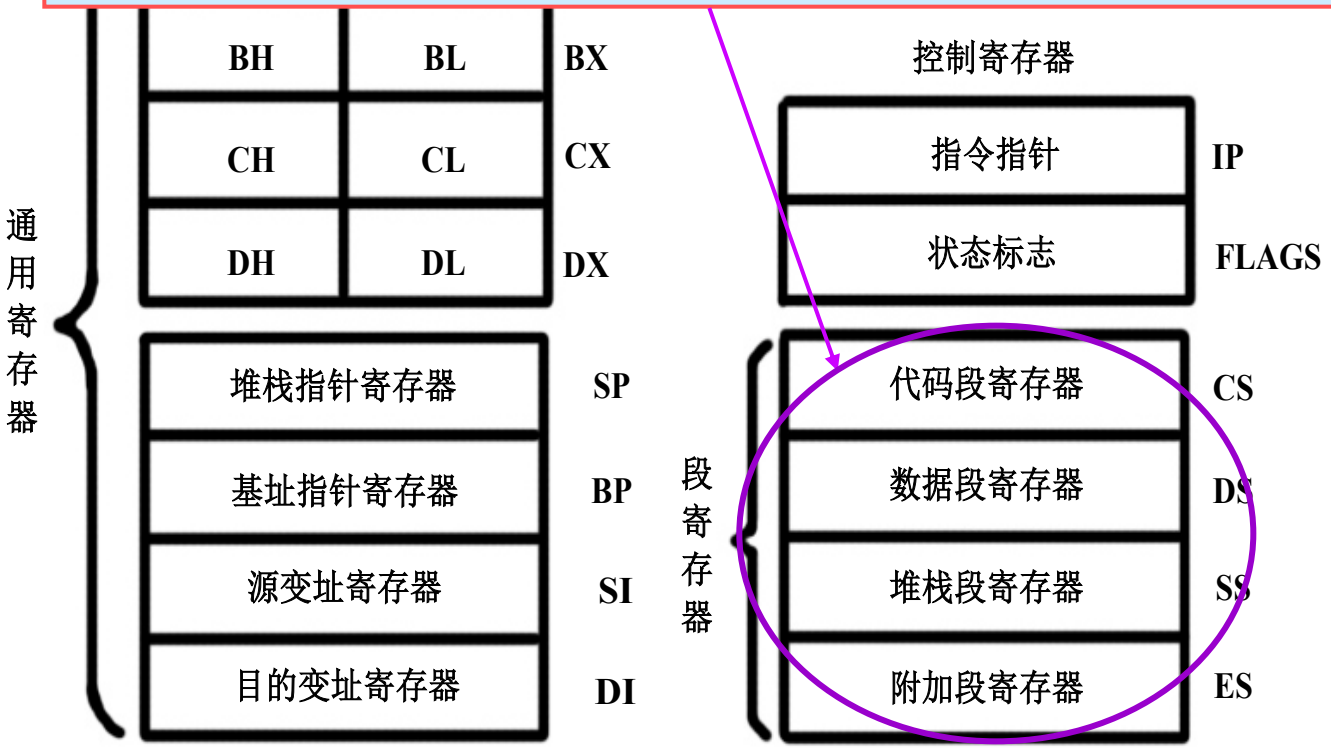
段寄存器：存放当前段的段起始地址

CS代码段寄存器，Code Segment，存放当前执行程序的段起始地址

SS堆栈段寄存器，Stack Segment，存放当前堆栈段的段起始地址

DS数据段寄存器，Data Segment，存放当前数据段的段起始地址

ES附加段寄存器，Extra Segment，存放当前附加段的段起始地址

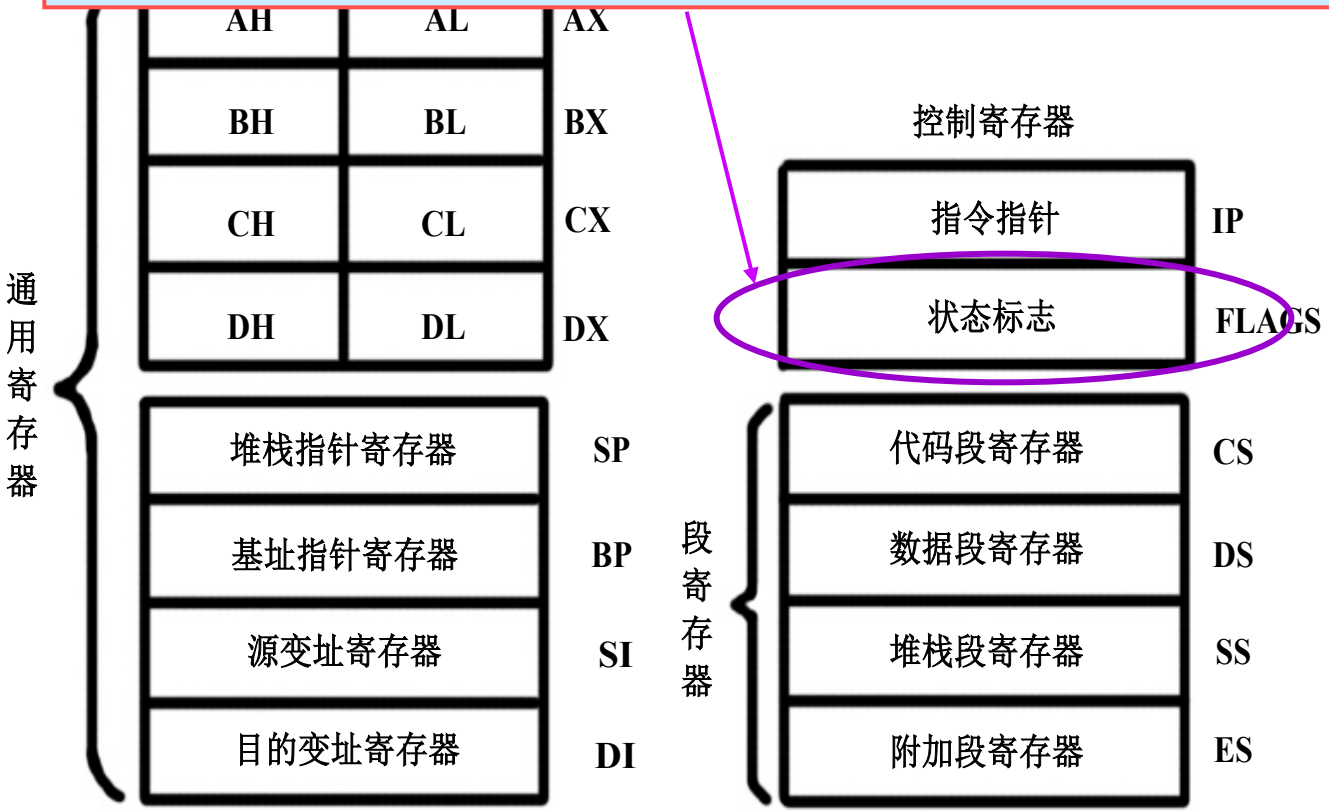


# 指令指针

- IP指令指针，Instruction Pointer，存放下一次要取出的指令的偏移地址
- CS:IP指示下一条要取出的指令的实际地址
- IP不能被程序直接存取，由BIU来修改，类似于程序计数器PC（Program Counter）

# 标志寄存器

**Flag标志寄存器：** 存放ALU状态和对CPU的控制状态
16位，定义9位，分2类：
1）状态标志：6位，CF、OF、ZF、SF、PF、AF，上次操作后ALU状态。
2）控制标志：3位， IF、DF、TF，人为设置，控制CPU操作。



# 标志寄存器中各位含义-状态位

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | OF | DF | IF | TF | SF | ZF | | AF | | PF | | CF |

1）CF进位，Carry Flag：结果有进位或借位，CF=1，否则CF=0。
2）PF奇偶，Parity Flag：低8位结果为偶数个1，PF=1，否则PF=0。
3）AF辅助进位，Auxiliary Carry Flag：D3有进位或借位，AF=1，否则AF=0。
4）ZF零标志，Zero Flag：结果为0，ZF=1，否则ZF=0。
5）SF符号位，Sign Flag：结果为正，SF=0，否则SF=1。
6）OF溢出标位，Overflow Flag：有符号数算术运算，结果超出其所能表示的数值范围，OF=1，否则OF=0。

例：35E5H+7832H=?　　CF,PF,AF,ZF,SF,OF=?

# 标志寄存器中各位含义-控制位

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | OF | DF | IF | TF | SF | ZF | | AF | | PF | | CF |

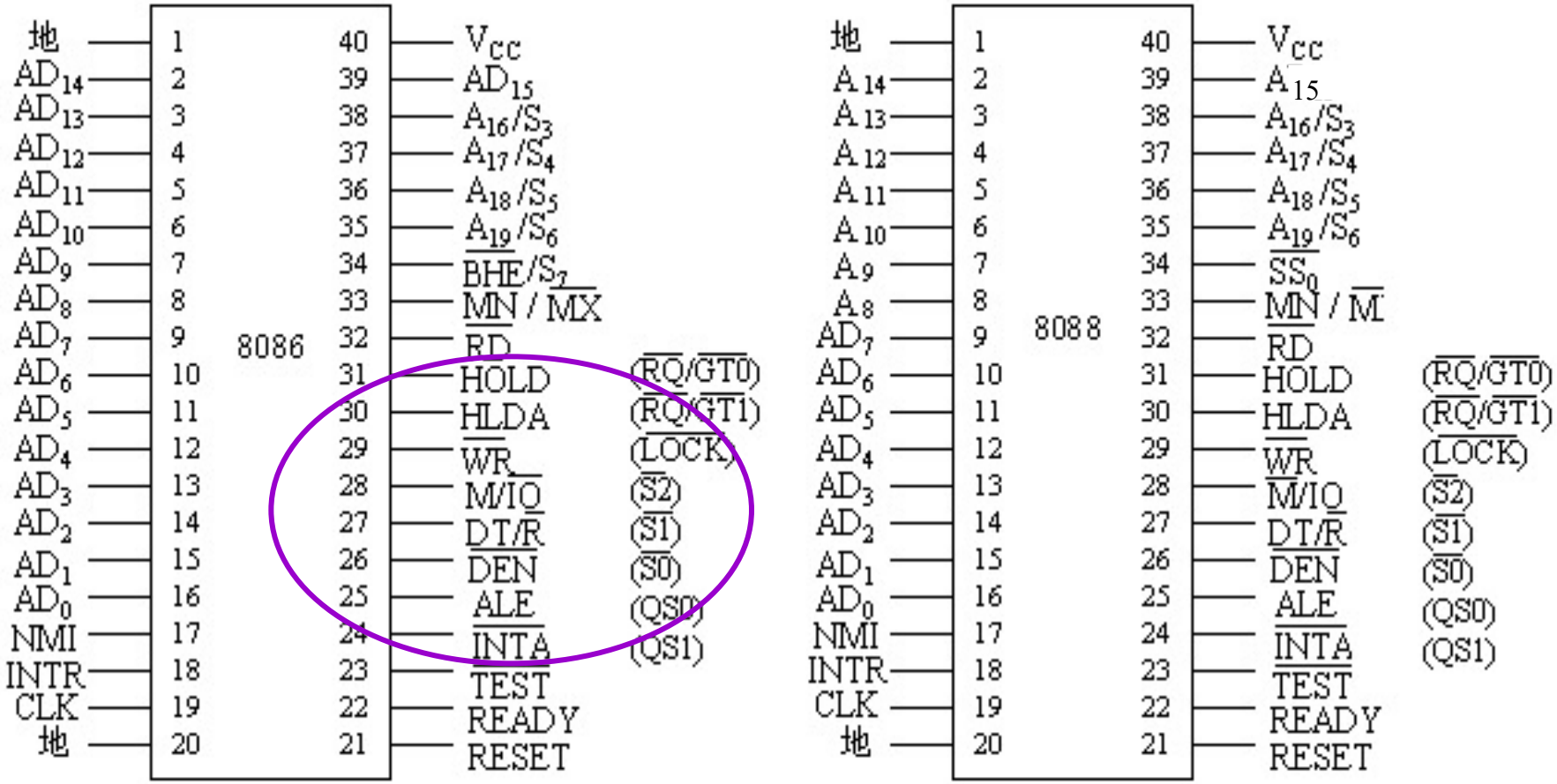
1）DF方向标志，Direction Flag：数据串操作地址修改方向控制，增址DF=0，减址DF=1。

2）IF中断允许，Interrupt Enable Flag：IF=1，允许可屏蔽中断（开中断），IF=0，关中断。

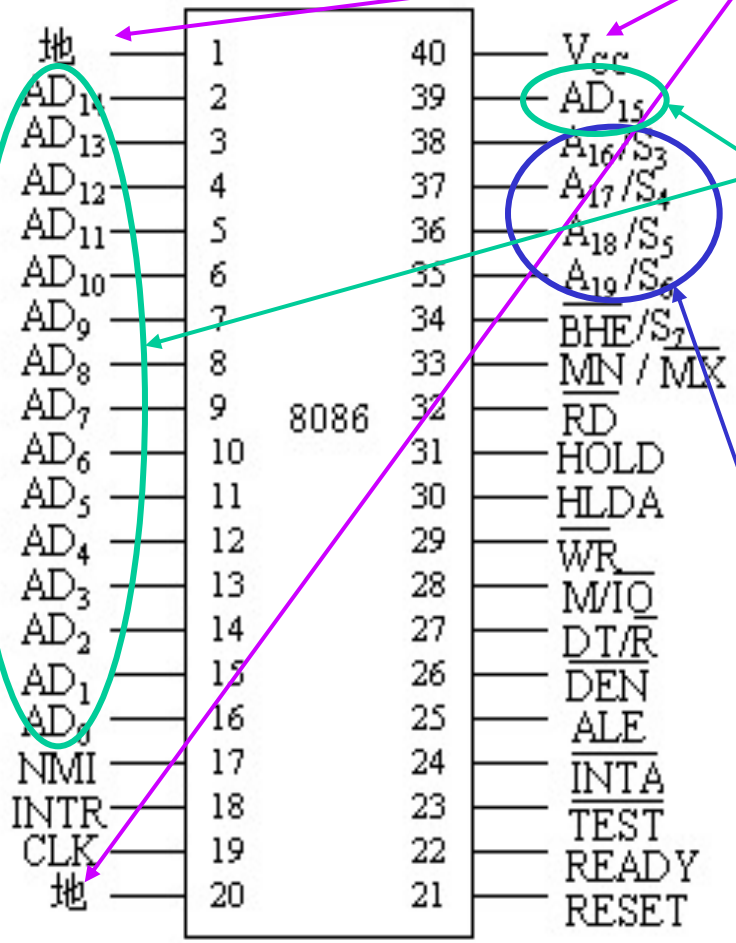
3）TF陷阱标志，Trap Flag：TF=1，单步方式，每执行一条指令自动产生一次类型为1的内部中断，使操作者可以逐条指令检查。 TF=0，正常。

# 2.3 8086微处理器外部结构

8086采用40引脚双列直插式封装。
8086有两种工作模式：最大模式、最小模式。
有8个引脚在不同的模式下功能有所不同。



# 2.3.1 两种模式下功能相同的引脚




（1）**VCC、GND：** 电源、地，单一+5V电源，2个接地。

（2）**AD15-AD0：** 地址/数据，分时输出低16位地址及数据信号。经地址锁存器输出对应的地址信号为A15-A0。

（3）**A19/S6-A16/S3：** 地址/状态，分时输出高4位地址及状态信息。

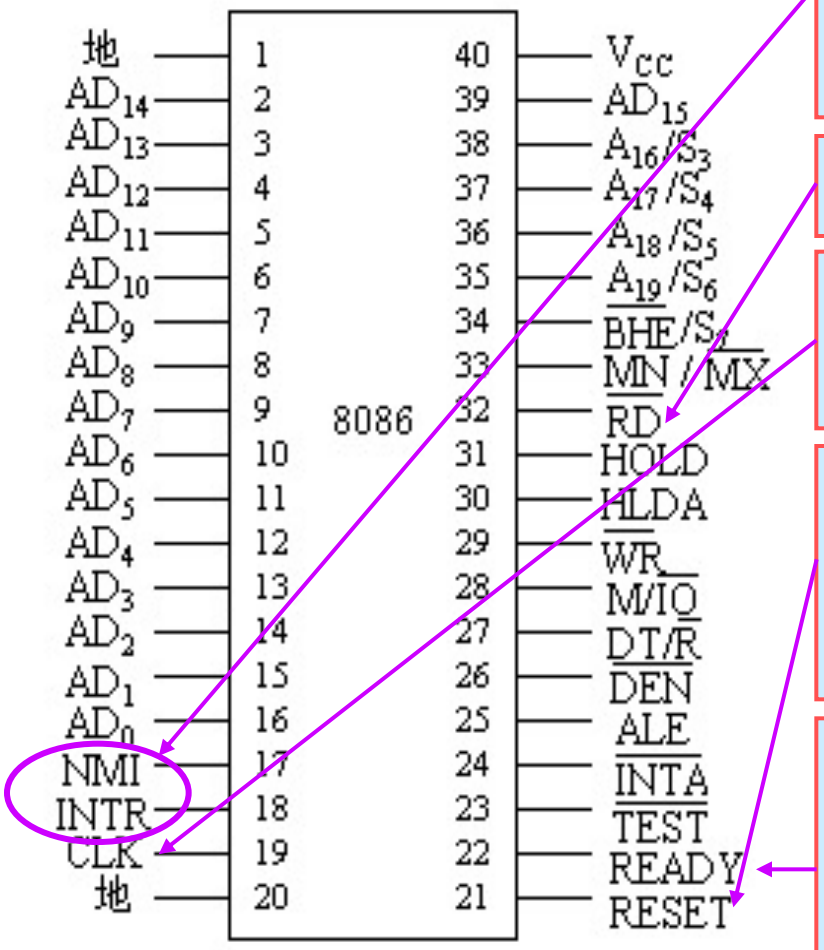
S6为0：CPU与总线连通

S5：等价于IF

S4、S3：使用哪个段寄存器

00=ES，01=SS，10=CS，11=DS

# 2.3.1 两种模式下功能相同的引脚




（4）**INTR**：可屏蔽中断请求信号，高电平有效；**NMI**：非屏蔽中断请求信号，上升沿有效。

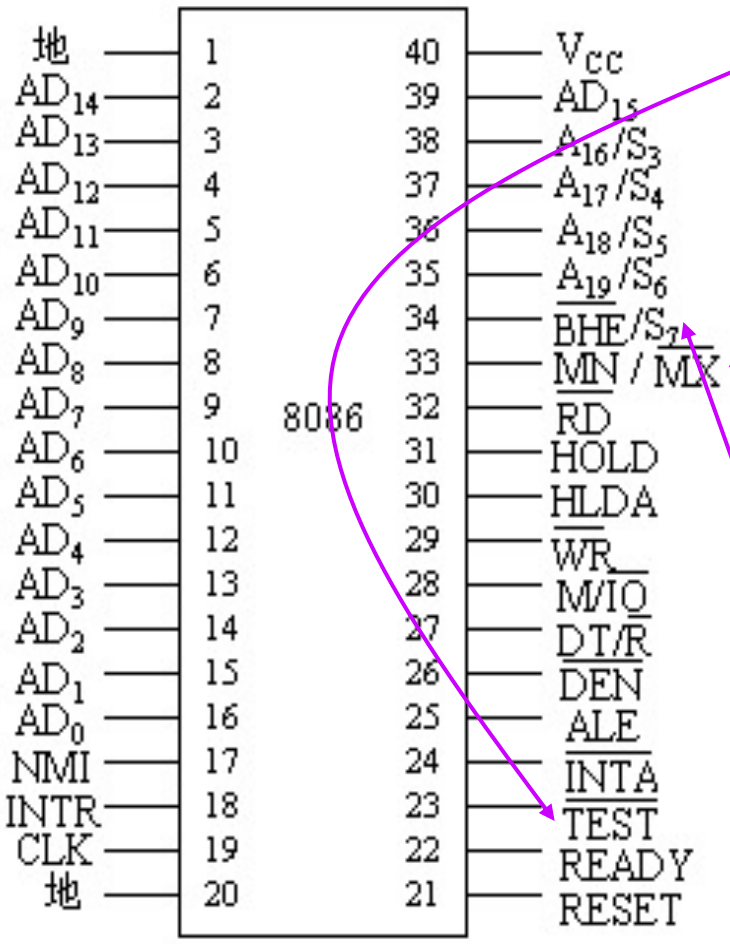
（5）$\overline{\text{RD}}$：读选通。

（6）**CLK**：时钟，占空比1/3，即1/3高电平、2/3低电平。

（7）**RESET**：复位，4个T周期，清0标志寄存器、IP、DS、SS、ES、指令队列，CS=FFFFH。

（8）**READY**：准备就绪，有效时表示主存或I/O接口准备好，可以进行数据传输。T3采样，若READY=0则插入TW。

# 2.3.1 两种模式下功能相同的引脚




（9）**TEST**：测试，由WAIT指令检查，使CPU与外部硬件同步，TEST=0继续，否则等待。

（10）**MN/MX**：最小/最大模式，接+5V最小模式，接地最大模式

（11）**BHE/S7**：高8位数据允许/状态，T1时BHE=0，AD15-AD8数据有效，S7未定义。

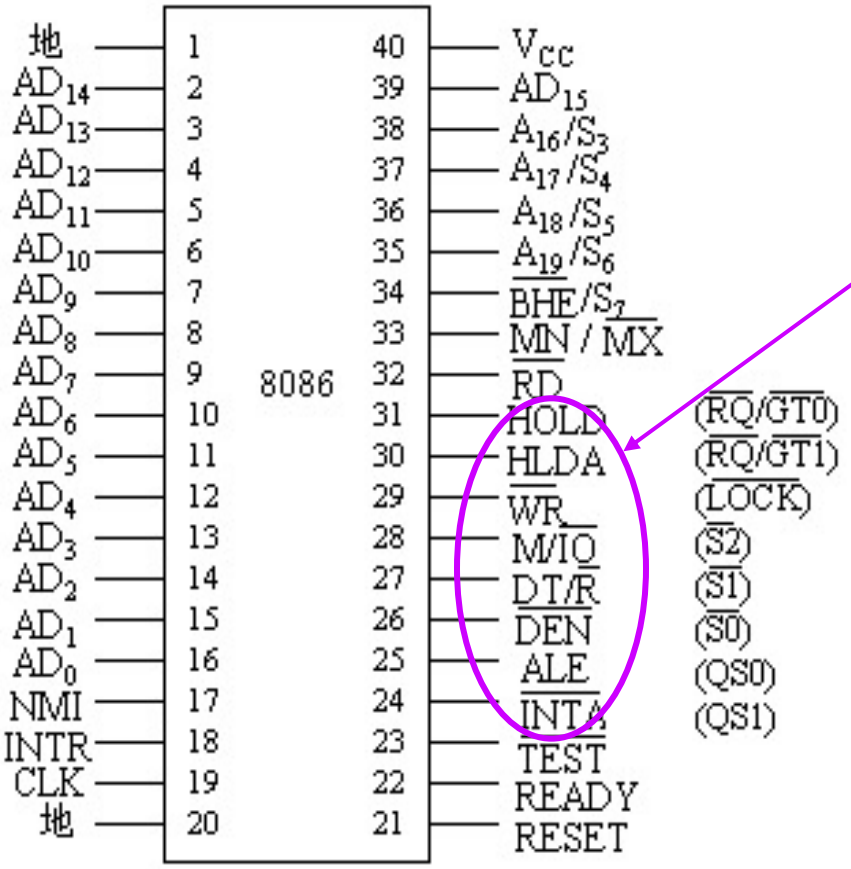
利用BHE信号和A0信号，可知系统当前的操作状态。

# 表2.3.1 BHE和A0的代码组合和对应的操作

| $\overline{BHE}$ 和A0的代码组合和对应的操作 | | | |
|---|---|---|---|
| $\overline{BHE}$ | A0 | 操作 | 所用数据引脚 |
| 0 | 0 | 从偶地址单元开始读/写一个字 | $AD_{15} \sim AD_0$ |
| 0 | 1 | 从奇地址单元或端口读/写一个字节 | $AD_{15} \sim AD_8$ |
| 1 | 0 | 从偶地址单元或端口读/写一个字节 | $AD_7 \sim AD_0$ |
| 1 | 1 | 无效 | -- |
| 0 | 1 | 从奇地址开始读/写一个字（在第一个总线周期将低8位数据送到AD15 ~AD8，下一个周期将高8位数据送到AD7 ~AD0） | $AD_{15} \sim AD_0$ |
| 1 | 0 | | |

2个周期

# 2.3.2 两种模式功能不同的引脚




**最小模式：**

**INTA：** 中断响应

**ALE：** 地址锁存允许

**DEN：** 数据允许

**DT/R：** 数据发送/接收

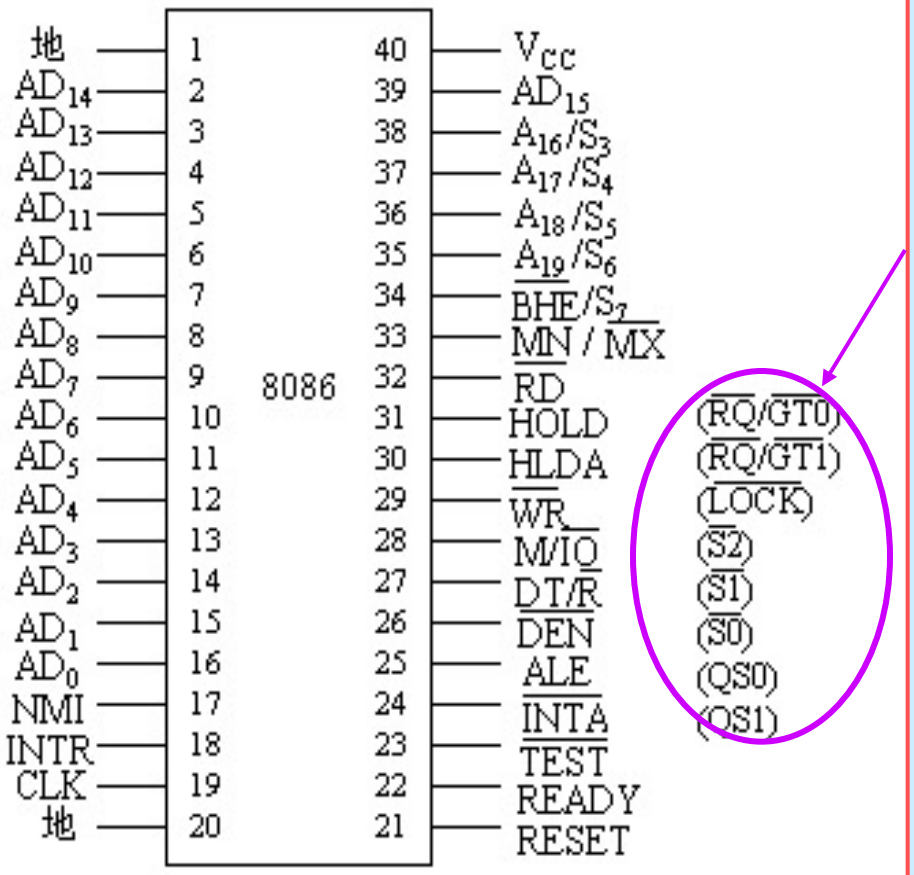
**M/IO：** 存储器/输入输出控制

**WR：** 写信号

**HOLD：** 总线请求

**HLDA：** 总线响应

# 2.3.2 两种模式功能不同的引脚




**最大模式：**

**QS1、QS0：** 指令队列状态，提供前一个T状态中指令队列的状态，允许外部跟踪CPU内部指令队列。

**$\overline{S2}$、$\overline{S1}$、$\overline{S0}$：** 总线周期状态，提供给8288。

**$\overline{LOCK}$：** 总线封锁，低电平时，别的设备不能获得总线控制权。由前缀指令“LOCK”产生。

**$\overline{RQ}/\overline{GT1}$、$\overline{RQ}/\overline{GT0}$：** 总线请求/允许，双向，$\overline{RQ}/\overline{GT1}$优先级高。

# 表2.3.4 S2、S1、S0的状态编码

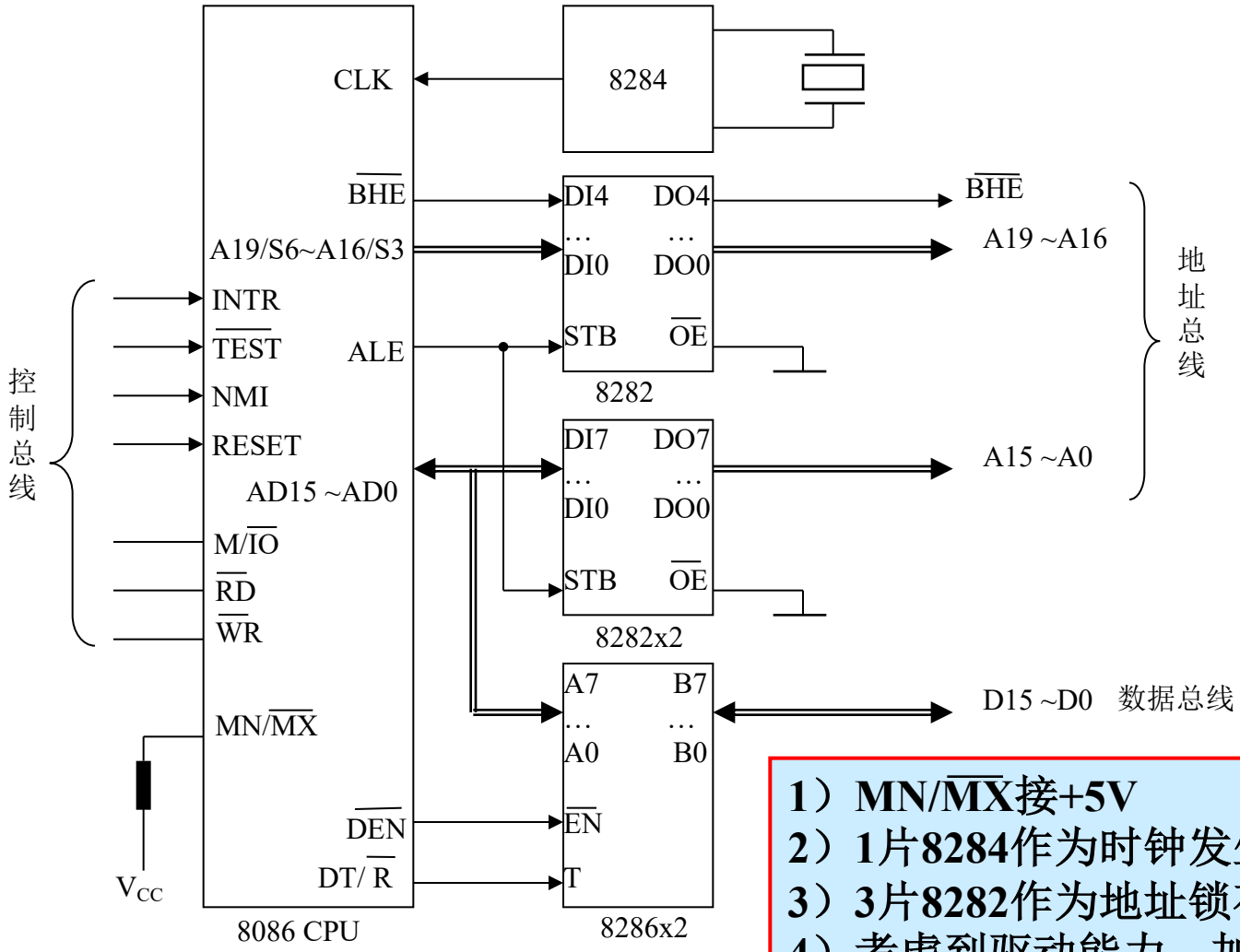
| $\overline{S2}$ | $\overline{S1}$ | $\overline{S0}$ | 操作 |
|---|---|---|---|
| 0 | 0 | 0 | 中断响应 |
| 0 | 0 | 1 | 读I/O端口 |
| 0 | 1 | 0 | 写I/O端口 |
| 0 | 1 | 1 | 暂停 |
| 1 | 0 | 0 | 取指 |
| 1 | 0 | 1 | 读存储器 |
| 1 | 1 | 0 | 写存储器 |
| 1 | 1 | 1 | 无作用 |

# 2.4 8086微处理器的两种组成模式

- 最小模式：单机系统，只有一个8086 CPU，所有控制信号由该CPU产生。系统中的控制电路可减到最小。

- 最大模式： 多机系统，有两个以上CPU，一个主8086 CPU，其他为协处理器（如8087、8089），控制信号由总线控制器8288产生。

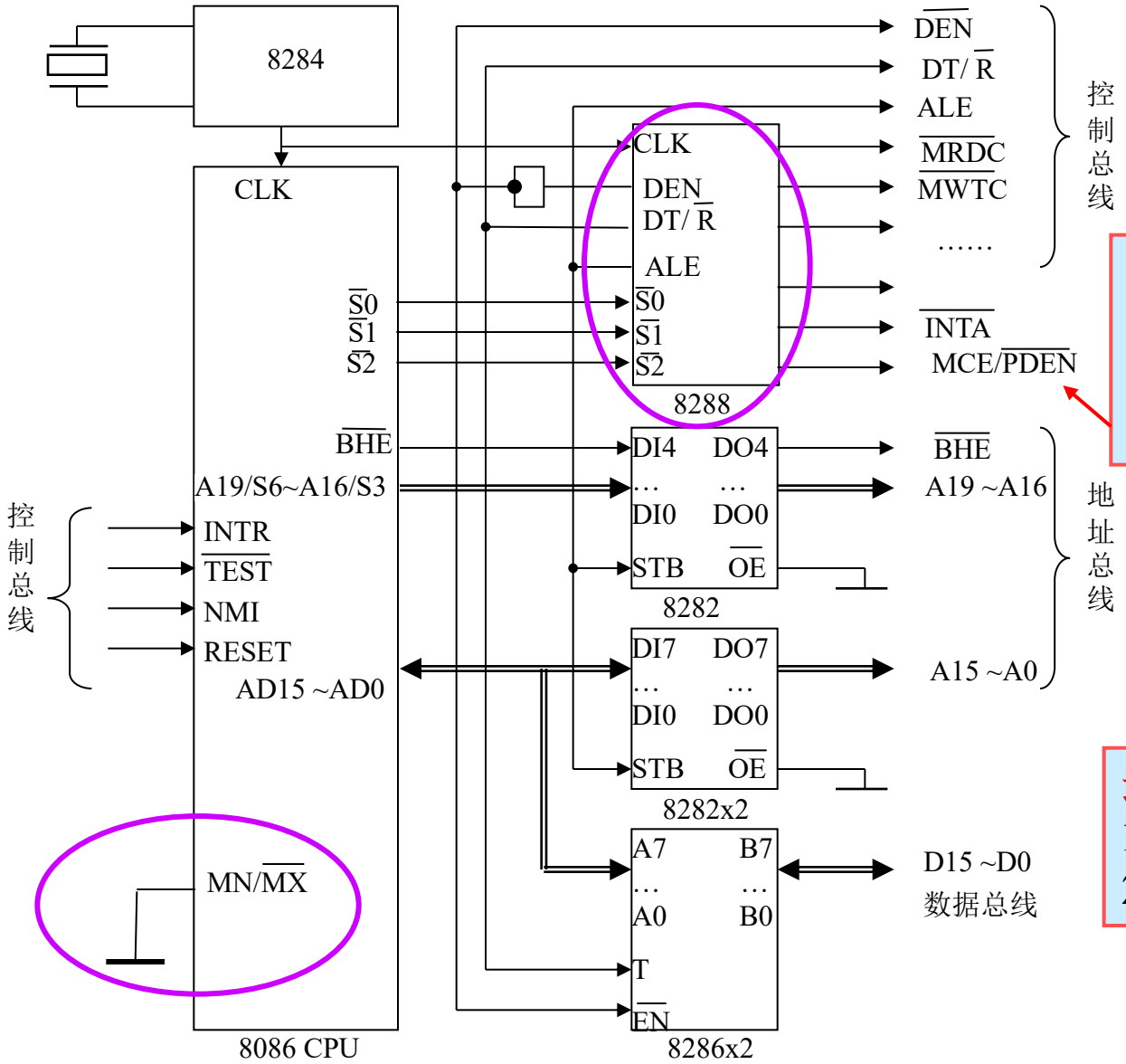
# 2.4.1 8086微处理器的最小模式




**硬件特点**

1）MN/$\overline{MX}$接+5V
2）1片8284作为时钟发生器
3）3片8282作为地址锁存器（或74LS373）
4）考虑到驱动能力，加2片8286作为总线驱动器（或74LS245）

# 2.4.2 8086微处理器的最大模式




**MCE：** 主控级联允许，
**IOB=0，** 用于系统总线方式
**PDEN：** 外围数据允许，
**IOB=1，** 用于I/O总线方式。

**最大模式与最小模式区别**
**1）MN/$\overline{MX}$接地**
**2）1片8288总线控制器**

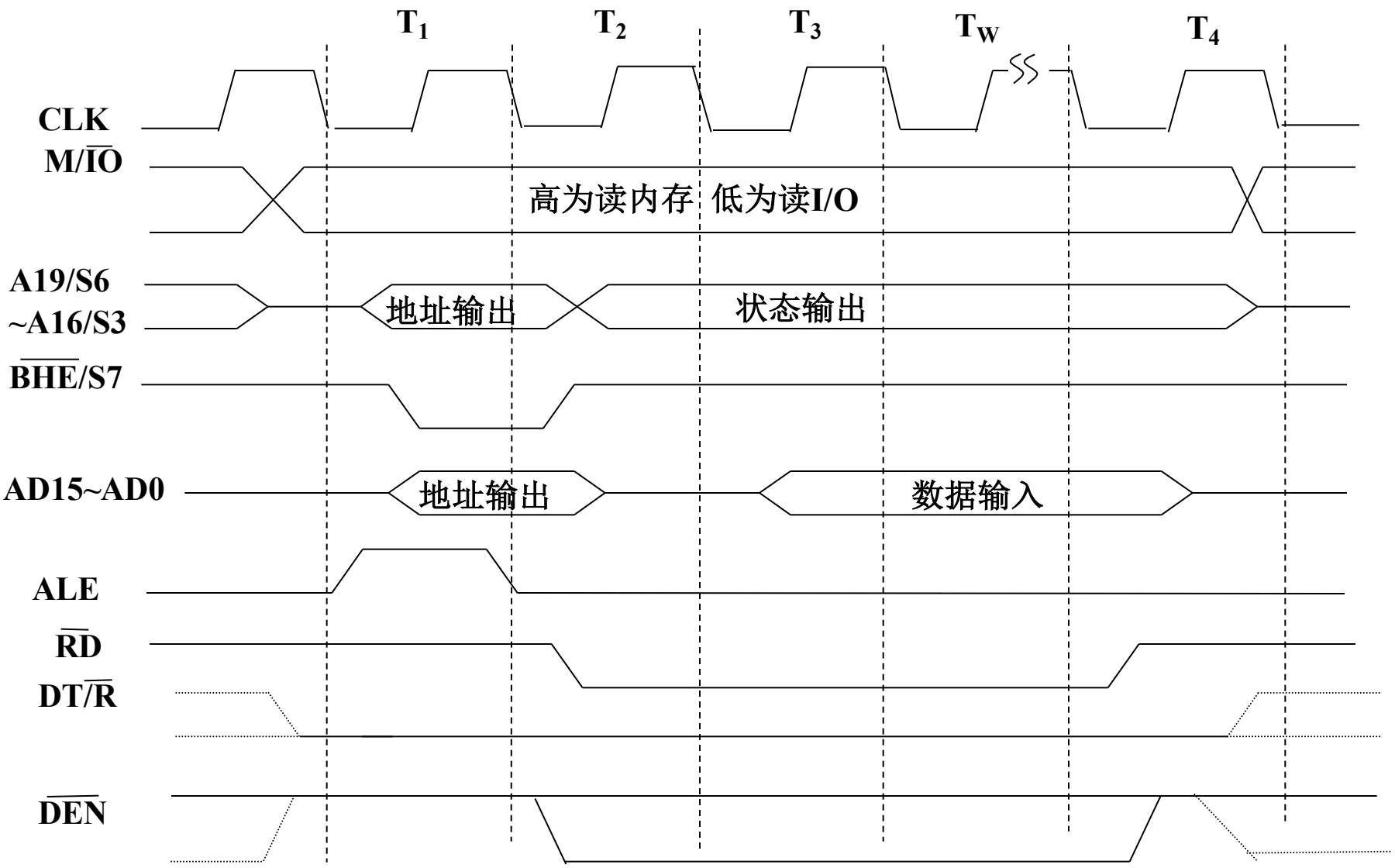
# 2.5 8086微处理器的总线周期

- 工作时序：微处理器按照一定的时序工作。
- 时钟周期：CPU主频每个时钟脉冲的持续时间。用一个“T”表示。
- 总线周期：CPU通过总线进行一次读或写的过程。总线读操作包括取指令、读存储器、读I/O接口，总线写操作包括写存储器、写I/O接口。还有一些特殊总线周期。
- 指令周期：执行一条指令所需要的时间。
- 总线请求响应：利用总线传输数据时，总线控制部件必须获得总线的控制权。总线请求是总线控制部件发出的占用总线的请求信号，总线响应是当前控制总线的部件或总线控制器在接收到总线请求后，给出的应答信号。
- 中断响应周期：中断响应是CPU接受中断请求后的处理过程。在响应中断时，CPU在当前指令结束后，插入两个总线周期，发出中断应答，并通过总线获取中断类型码。

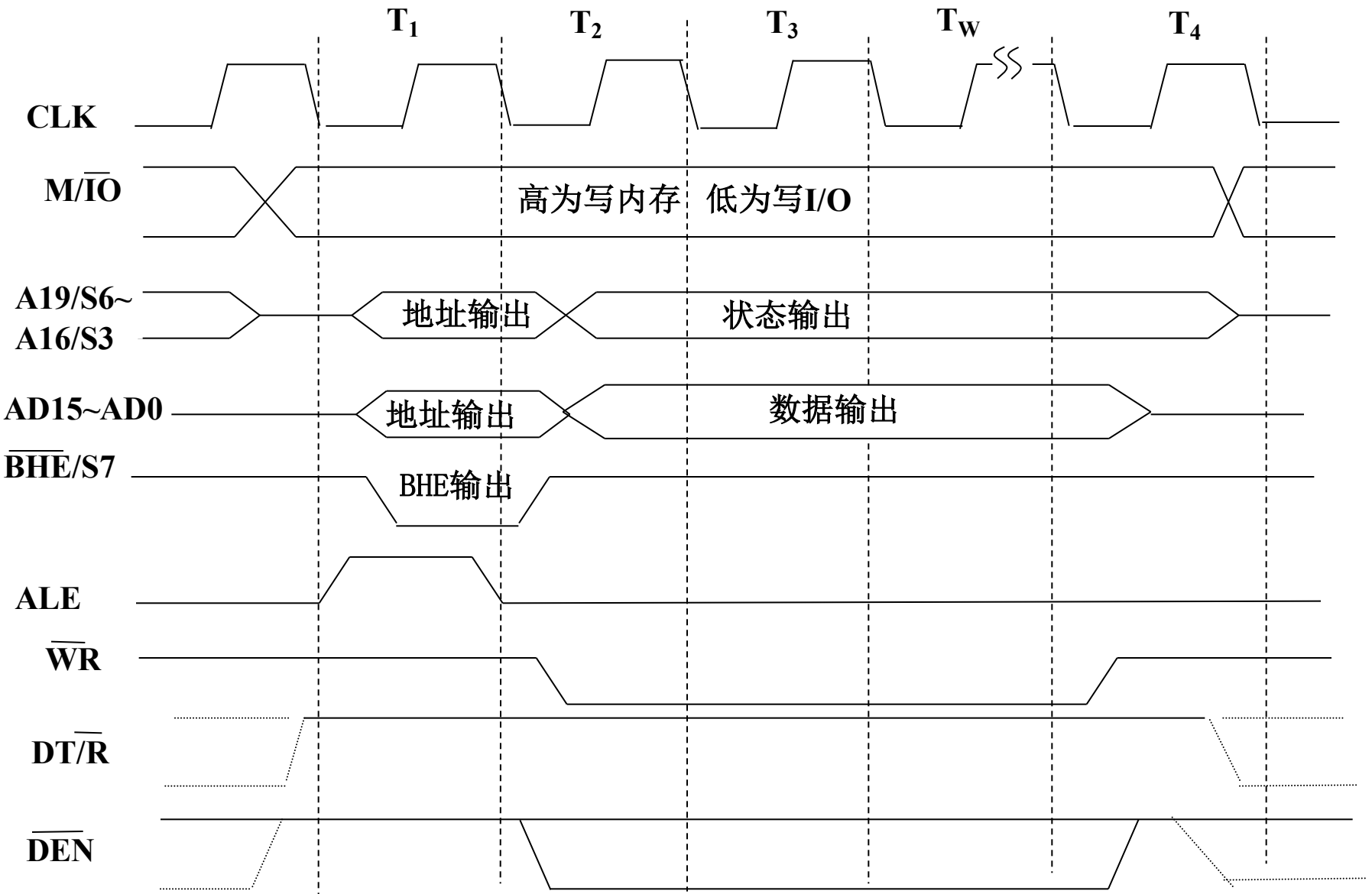
# 8086时钟状态

- 8086一个基本的总线周期由4个时钟周期组成。每个时钟称为T状态，用T1、T2、T3、和T4表示。
- T1状态： CPU发出地址。CPU将存储器地址或I/O端口的地址送上地址总线。
- T2状态：撤地址，发控制信号。进行读写准备，CPU撤销地址/数据、地址/状态复用线上的地址，地址/数据复用线浮置，地址/状态复用线输出状态，即复用信号在这个期间切换，读写控制信号有效。
- T3状态：地址/数据线上出现数据。写操作，CPU提供数据；读操作，等待存储器或I/O提供数据。检查READY信号，未就绪，插入TW。TW的操作与T3相同。
- T4状态：完成数据读/写，控制信号无效。结束总线操作。

# 8086总线读周期-最小模式

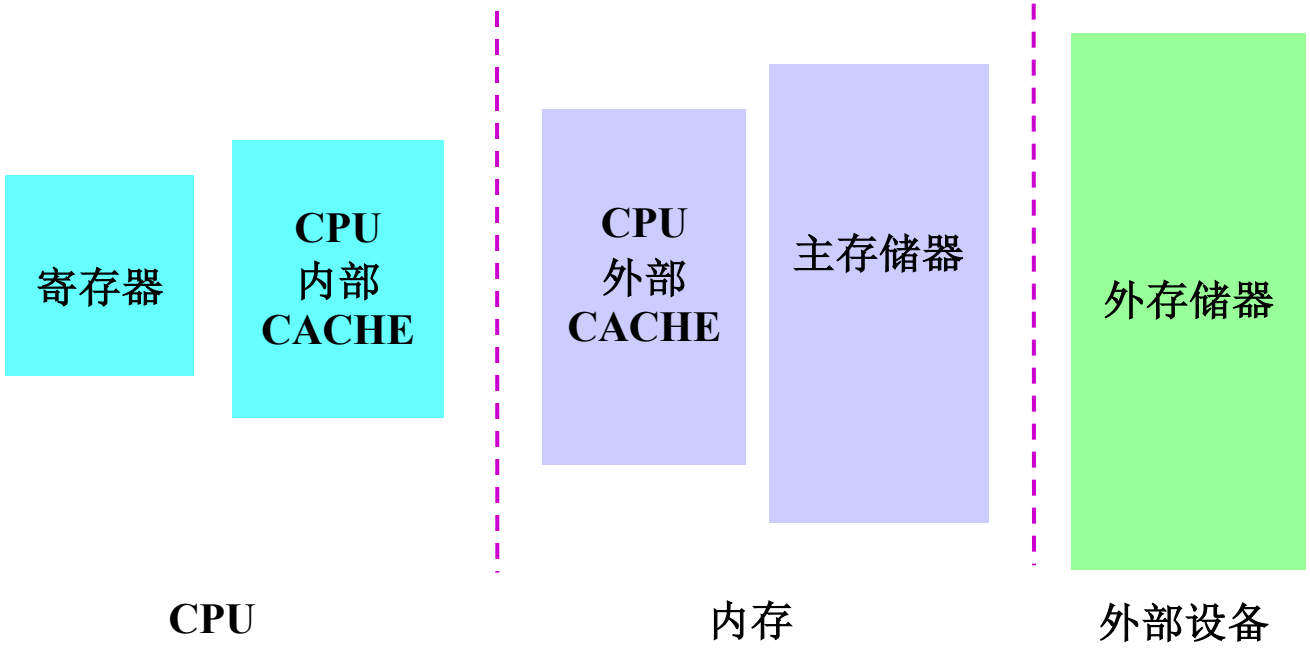


# 8086总线写周期-最小模式

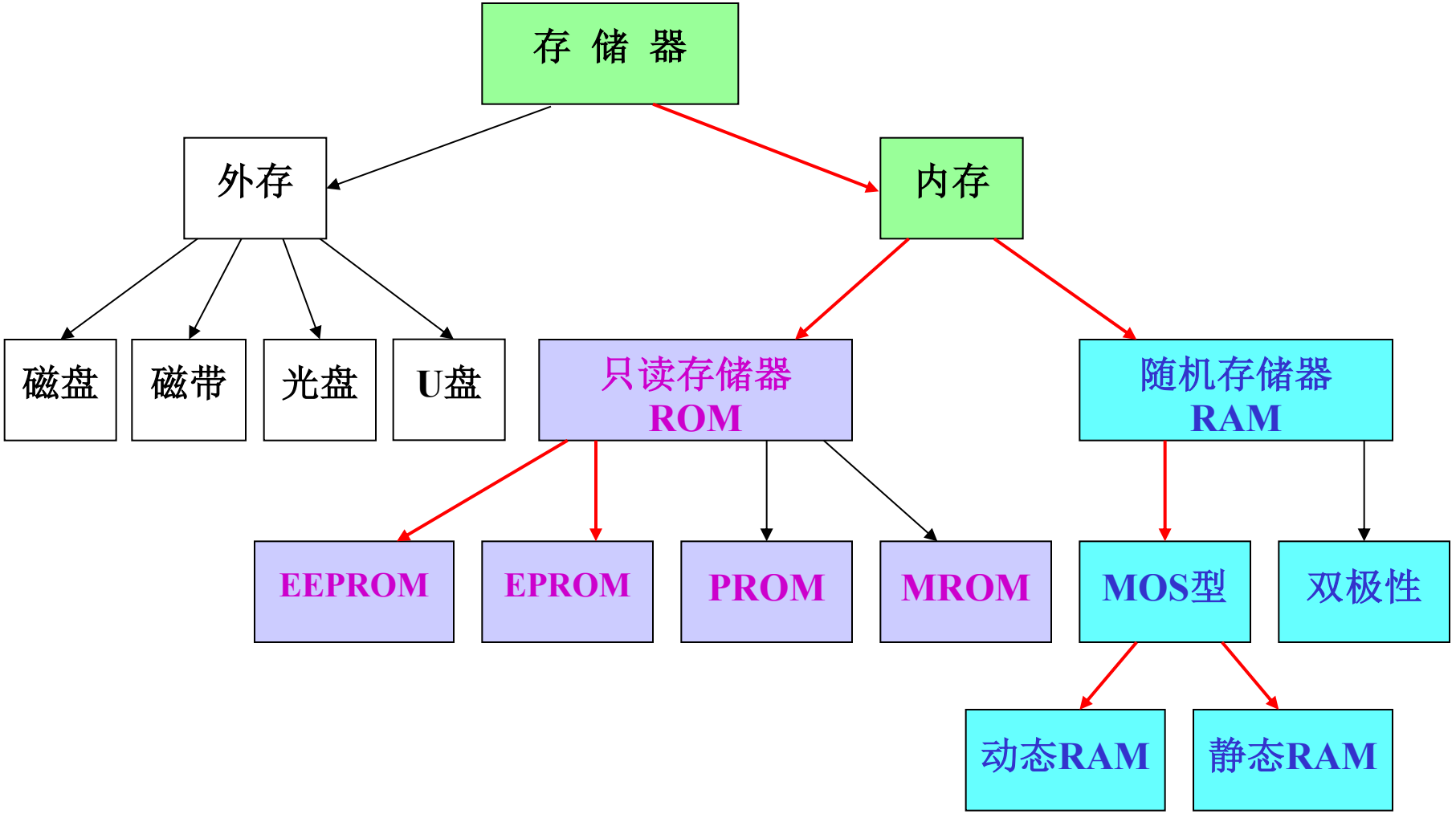


# 2.6 8086微处理器的存储器组织

- 微型计算机存储系统采用层次结构：高速缓存技术、虚拟存储技术。
- 从连接角度：靠近CPU的存储体速度快、成本高、存储容量小。
- 从CPU角度：访问速度接近最靠近的存储体，访问空间接近最远的存储体空间范围。



# 2.6.1 存储器概述


存 储 器
外存
内存
磁盘
磁带
光盘
U盘
只读存储器
ROM
随机存储器
RAM
EEPROM
EPROM
PROM
MROM
MOS型
双极性
动态RAM
静态RAM

# 2.6.2 存储器组织

- 地址空间：8086有20位地址线，按字节单元编址，直接寻址能力1M字节，地址范围为00000H-FFFFFH。每个存储单元对应一个20位的地址，这个直接通过地址线给出的地址称为物理地址。
- 8086支持字节数据、字（16位）数据的存取。
- 字存放在任意连续的两个单元中，低字节放在低地址中（小端方式），低字节对应的地址为这个字数据的地址。
- 对齐字：在偶数地址中开始存放。读写时，只需一个总线周期。
- 非对齐字：在奇数地址中开始存放。读写时，需两个总线周期。

# 2.6.3 存储器的分段管理

- 20位地址，1M空间，8086CPU内部寄存器都是16位，无法直接提供20位地址，故分段管理，每段64K，段内地址连续，各段独立。
- 为了寻址每一个段，应当指明段的起始位置，这个起始位置就放在段寄存器中。

- 而段寄存器是16位的，因而规定每个段的起始位置的实际地址应被16整除，即该地址低4位为0。这样，一个段20位起始地址的高16位就是该段的段地址，段内任一个单元的地址，用相对于段起始地址的偏移量表示，即段内偏移量，16位，64K。

# 逻辑地址与物理地址

- 每一个存储单元都有2类地址：
  - 物理地址：信息在存储器中实际存放的地址。地址通过地址线给出。
  - 逻辑地址：编程中使用的地址。
- 逻辑地址的格式：

  段地址：段内偏移量
- 段地址：决定该段第一个字节的位置。
- 段内偏移量：该储存单元相对于该段起点字节的距离。指令中称为有效地址EA。
- 物理地址计算：

  物理地址=段寄存器×16 + 段内偏移量

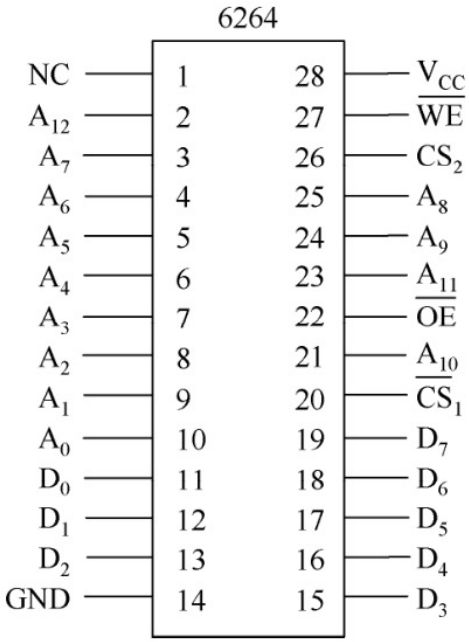
  （20位） （16位）（10H） （16位）
- 同一个物理地址可以用不同的逻辑地址表示。

# 段寄存器的使用约定

| 存储器操作类型 | 隐含段 | 超越段 | 段内偏移量 |
|---|---|---|---|
| 取指令 | CS | 无 | IP |
| 堆栈操作 | SS | 无 | SP |
| 数据变量 | DS | CS、SS、ES | 有效地址EA |
| 源串变量 | DS | CS、SS、ES | SI |
| 目的串变量 | ES | 无 | DI |
| 基址BP指针 | SS | CS、DS、ES | 有效地址EA |

# 2.6.4 典型的存储器芯片

- **1. 静态随机存储器（SRAM）**
- **SRAM特点：** 用触发器存储信息，电源掉电时信息丢失，属易失性半导体存储器，不需要刷新，速度快，功耗大。
- **Intel 6264：SRAM，容量8K×8位，28条引线，包括13根地址线、8根数据线、4根控制信号线。**




6264读写控制逻辑

| $\overline{CS_1}$ | $CS_2$ | $\overline{WE}$ | $\overline{OE}$ | $D_7 \sim D_0$ |
|---|---|---|---|---|
| 0 | 0 | × | × | 三态 |
| 0 | 1 | 0 | 1 | 写入 |
| 0 | 1 | 1 | 0 | 读出 |
| 1 | 0 | × | × | 三态 |
| 1 | 1 | × | × | 三态 |

图 2.6.3　6264 引脚结构及其读写控制逻辑

# 2. 可擦除可编程只读存储器（EPROM）

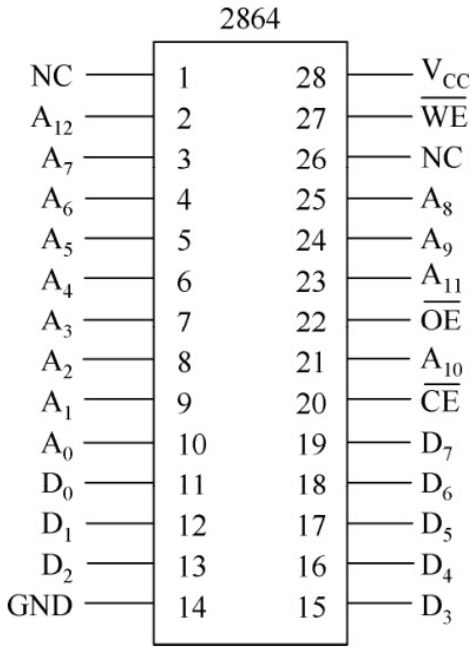
- **EPROM：** 可由用户进行编程、并可用紫外光擦除的ROM芯片。
- **典型的EPROM芯片：** 27系列。

| | | | |
|---|---|---|---|
| 2716 | 2K×8位 | 27512 | 64K×8位 |
| 2732 | 4K×8位 | 27010 | 128K×8位 |
| 2764 | 8K×8位 | 27020 | 256K×8位 |
| 27128 | 16K×8位 | 27040 | 512K×8位 |
| 27256 | 32K×8位 | 27080 | 1M×8位 |

- **2764：** 8K×8位，13根地址线A12～A0，8条数据线D7～D0，电源Vcc，编程电源Vpp，一个片选端nCE和一个输出允许控制端nOE。
- 一个编程控制端nPGM，编程时，该引脚需加50ms宽的负脉冲；正常读出时，该引脚应无效。
- **5种工作方式：** 读出、维持、编程、编程校验、编程禁止。

# 3. 电可擦除可编程只读存储器（EEPROM）

- **EEPROM：** 可用加电的方法在线擦除和编程，擦写次数大于1万次，数据可保存10年以上。
- EEPROM芯片提供不同的擦写手段：字节擦除、整片擦除、状态查询、页擦写等。
- 典型EEPROM芯片：28系列，如2816、2817、2864。




2864 控制逻辑

| 方式 | $\overline{CE}$ | $\overline{OE}$ | $\overline{WE}$ | $D_7 \sim D_0$ |
|---|---|---|---|---|
| 读出 | 0 | 0 | 1 | 读出 |
| 维持 | 1 | × | × | 三态 |
| 写入 | 0 | 1 | 负脉冲 | 写入 |
| 数据查询 | 0 | 0 | 1 | 读出 |

图 2.6.5　2864 引脚结构及其控制逻辑

# 2.6.5 微处理器与存储器芯片连接时应处理的问题

- **1．总线驱动能力**

  CPU的总线驱动能力是指可以直接驱动的标准门电路器件数量，通常CPU的直流负载是一个TTL器件，现在多为MOS器件，为容性负载，直流分量较小，可带20个。但若系统较大，需要驱动AB、DB、CB。

- **2．时序配合**

  CPU时序与存储器存取速度配合，可插入等待状态Tw。

- **3．数据线的连接**

  字节编址，存储体为8位，CPU数据线宽度不同，连接的存储体的个数也不同。

# 微处理器与存储器芯片连接时应处理的问题

- 4. 地址线的连接
- 问题：多个芯片连接，需要选择。

- CPU地址信号与存储器连接的选择有三种：
- （1）低位地址信号

CPU低位地址选择存储器体，即体选。每个存储体数据宽度一个字节，8位CPU连1个存储体，依次，16位→2个体， 32位→4个体， 64位→8个体。

- （2）中间位地址信号

CPU中间位地址用于片内译码，即字选。

- （3）高位地址信号

CPU高位地址选择存储器芯片，即片选，有三种方法，即全译码、部分译码、线选。

1）全译码法：高位地址线全部参加译码，地址连续。

2）部分译码法：高位地址线部分参加译码，地址不连续。

3）线选法：高位地址线单根地址线进行片选，地址不连续。

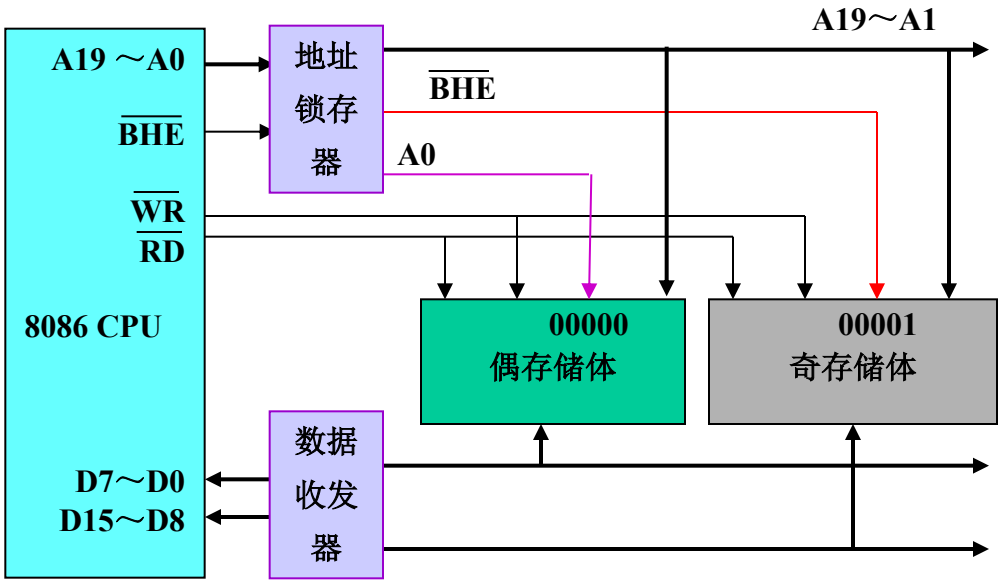
# 微处理器与存储器芯片连接时应处理的问题

- **5. 读写控制线的连接**
- 读写控制信号用于控制对存储器的读/写操作。当存储芯片的工作速度与CPU不匹配时，存储器芯片的接口电路就必须具有向CPU发等待命令的控制信号，以使CPU根据需要插入1个或几个等待周期。

- **6. ROM与RAM在存储器中的地址分配**
- RAM存放临时数据和当前的应用软件，非易失的ROM存放核心系统软件。80X86微处理机复位后从物理地址高端开始运行，所以总是在物理地址空间的高地址位置使用只读存储器ROM。

# 2.6.6 8086微处理器与存储器芯片的连接

- Intel 8086是典型的16位微处理器，但其可以访问字节数据单元。因此，要求存储器系统接口的设计能保证一次访问一个字，也能一次只访问一个字节。




接口关键：

（1）16位/8=2个存储体，偶体和奇体。

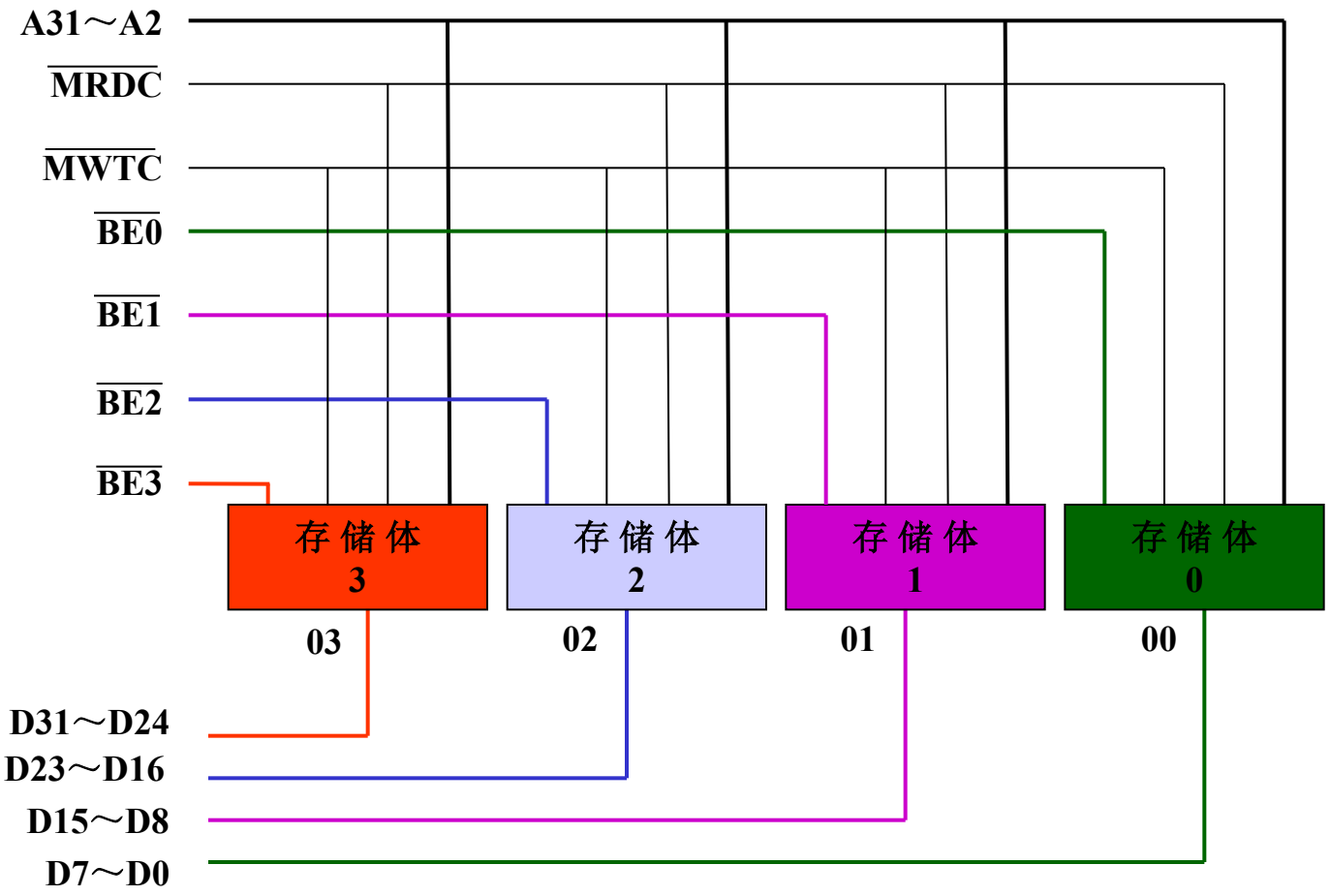
（2）偶体D7-D0，由A0=0选择。

（3）奇体D15-D8，由$\overline{BHE}$=0选择。

表2.6.1　存储体选择编码表

| $\overline{BHE}$ | A0 | 传送的字节 | 所用的数据引脚 |
|---|---|---|---|
| 0 | 0 | 从偶地址开始两个字节读/写 | D15～D0 |
| 0 | 1 | 奇地址的一个字节读/写 | D15～D8 |
| 1 | 0 | 偶地址的一个字节读/写 | D7～D0 |
| 1 | 1 | 不传送 | |

# 32位微型计算机系统中的主存储器接口

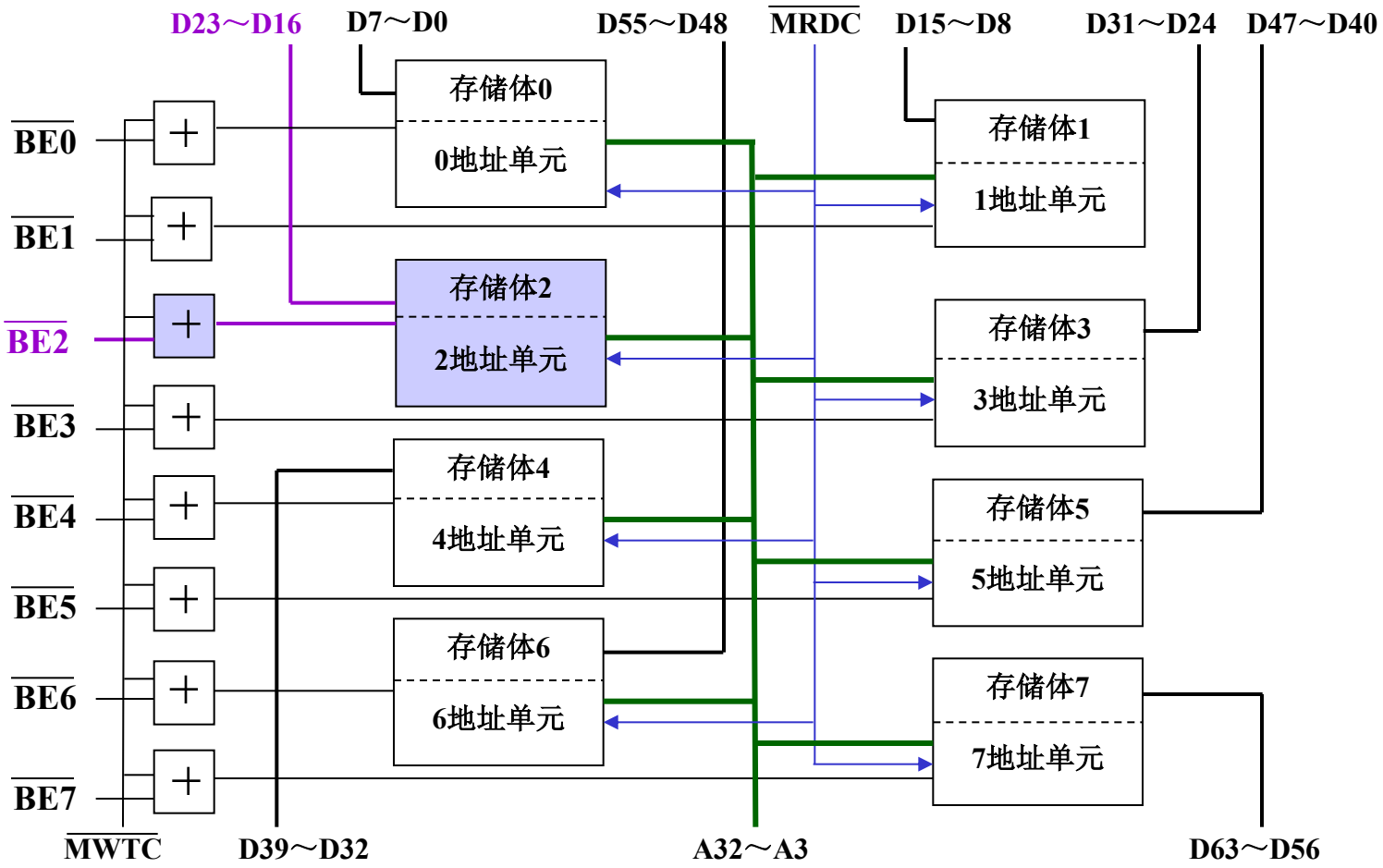



接口关键：（1）4个体，每体1G

（2）每体分别连相应的8位数据线

（3）每个体用$\overline{BE}$2-0对应选通

# 64位微型计算机系统中的主存储器接口



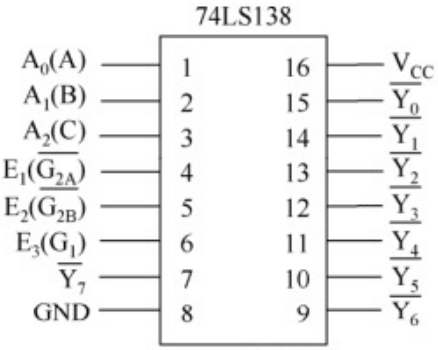

接口关键：（1）8个体，每体512M

（2）每体分别连相应的8位数据线

（3）每个体用$\overline{BE7\text{-}0}$对应选通，独立写选通

# 3/8译码器：74LSl38

- 在微型计算机系统中，CPU与存储器连接时一般采用译码器芯片作为高位地址产生片选信号的地址译码器。
- 常用的译码器之一是74LS138，这是一个3/8译码器芯片，有三个选择输入端，三个控制端，八个输出端。

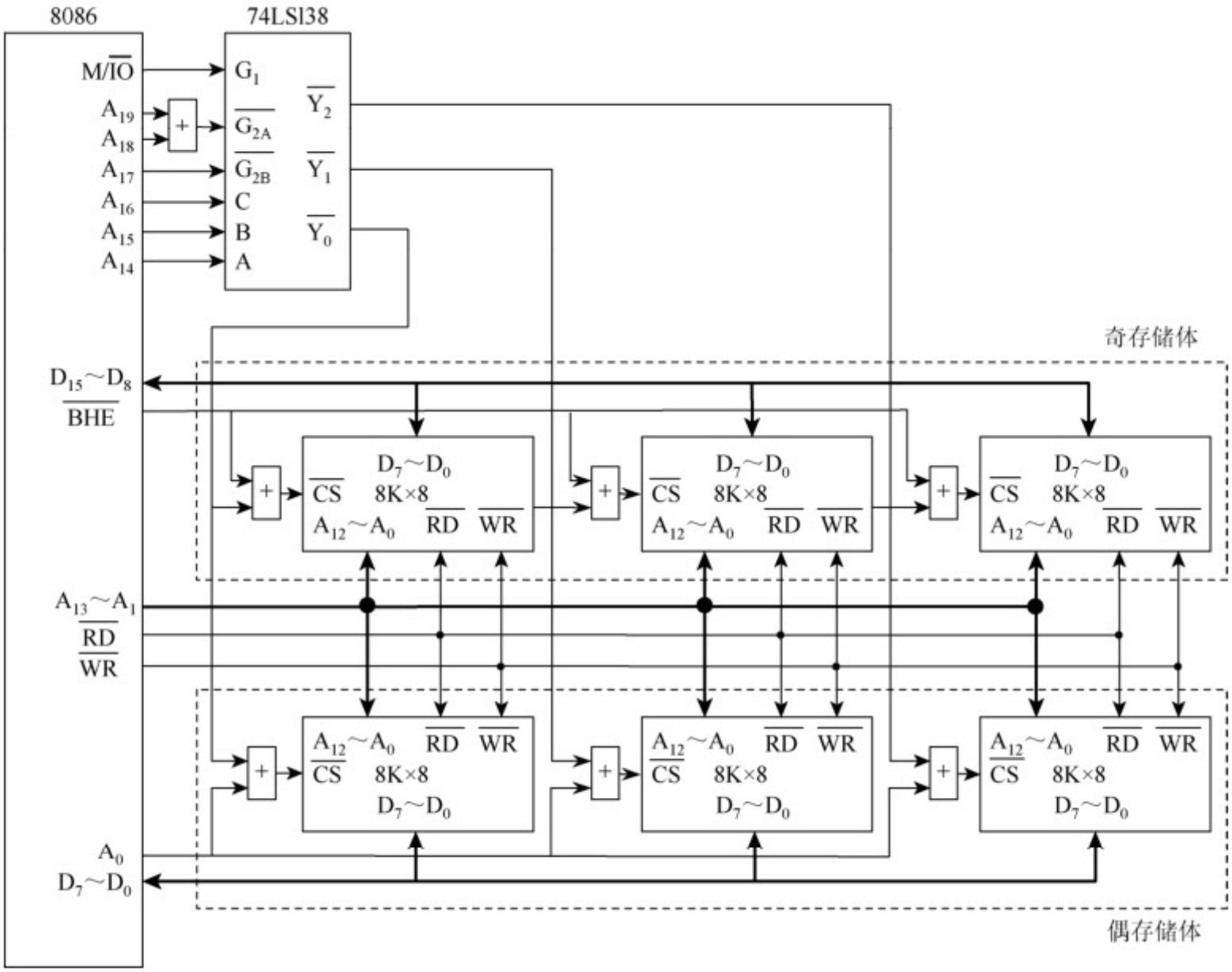



74LS138控制逻辑

| 输入 | | | | | | 输出 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 控制 | | | 选择 | | | | | | | | | | |
| $G_1$ | $\overline{G_{2A}}$ | $\overline{G_{2B}}$ | C | B | A | $\overline{Y_7}$ | $\overline{Y_6}$ | $\overline{Y_5}$ | $\overline{Y_4}$ | $\overline{Y_3}$ | $\overline{Y_2}$ | $\overline{Y_1}$ | $\overline{Y_0}$ |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 不是100组合 | | | × | × | × | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

图 2.6.8　74LS138 引脚结构及其控制逻辑

# 图2.6.7 8086微处理器存储器系统例



# 2.7 8086微处理器的I/O端口组织

- I/O 设备不能直接与CPU总线相连，需要通过I/O接口芯片。
- 每个I/O芯片都有一个端口或几个端口，一个端口往往对应芯片内部的一个寄存器或者一组寄存器。
- 每个端口分配一个地址，称为端口地址。
- 8086使用低16位地址线对I/O端口进行编址，最多允许有65536个8位的I/O端口，两个编号相邻的8位端口可以组合成一个16位端口。
- 8086采用独立的端口编址方式，指令系统中既有访问8位端口的输入/输出指令，也有访问16位端口的输入/输出指令。
- PC机使用A9-A0，有1024个端口。

# 2.8 8086微处理器的中断系统

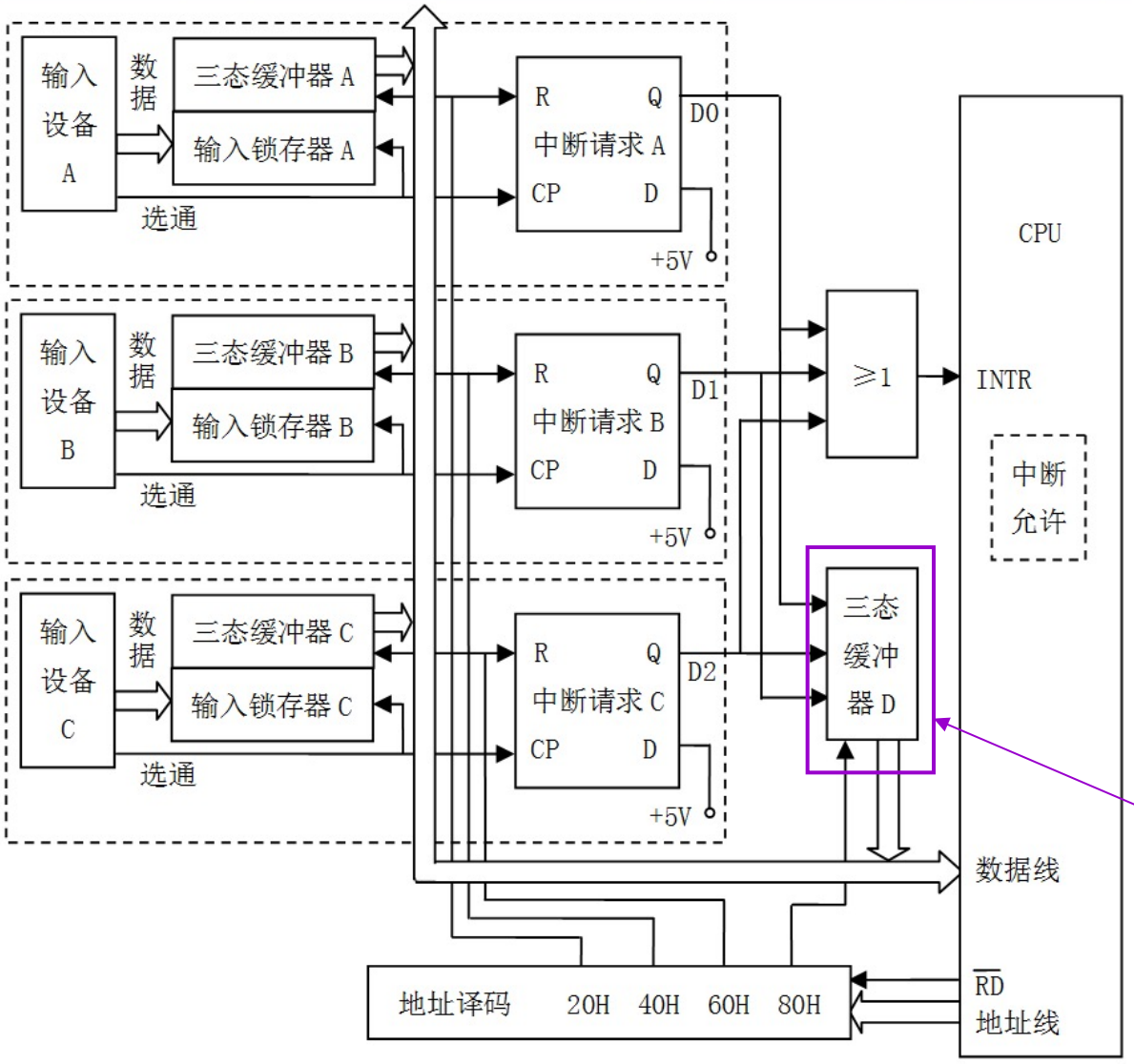
- **2.8.1 中断的基本概念与分类**

- **1. 中断和中断源**
- **中断：** 描述了一种CPU处理程序的过程。CPU在正常执行当前程序时，由某个事件引起CPU暂时停止正在执行的程序，进而转去执行请求CPU暂时停止的相应事件的服务程序，待该服务程序处理完毕后又返回继续执行被暂时停止的程序。
- **中断系统：** 实现中断功能的硬件和软件系统。
- **中断源：** 产生中断的来源。

# 2．中断优先级

- 出现情况：多个中断源同时请求中断。
- 解决方法：采用中断优先级排队。
- 中断优先排队方法：软件查询、硬件优先级排队。

# 软件查询确定优先级




- 在软件查询确定优先级方式中，询问的次序即为优先级的次序。
- 其缺点是在中断源较多的情况下，由询问到转至相应的中断服务程序的入口时间较长。

关键部件：中断状态缓冲器

# 硬件优先级排队电路确定优先级

- 菊花链法是硬件优先级排队电路确定优先级中的一个简单方法。

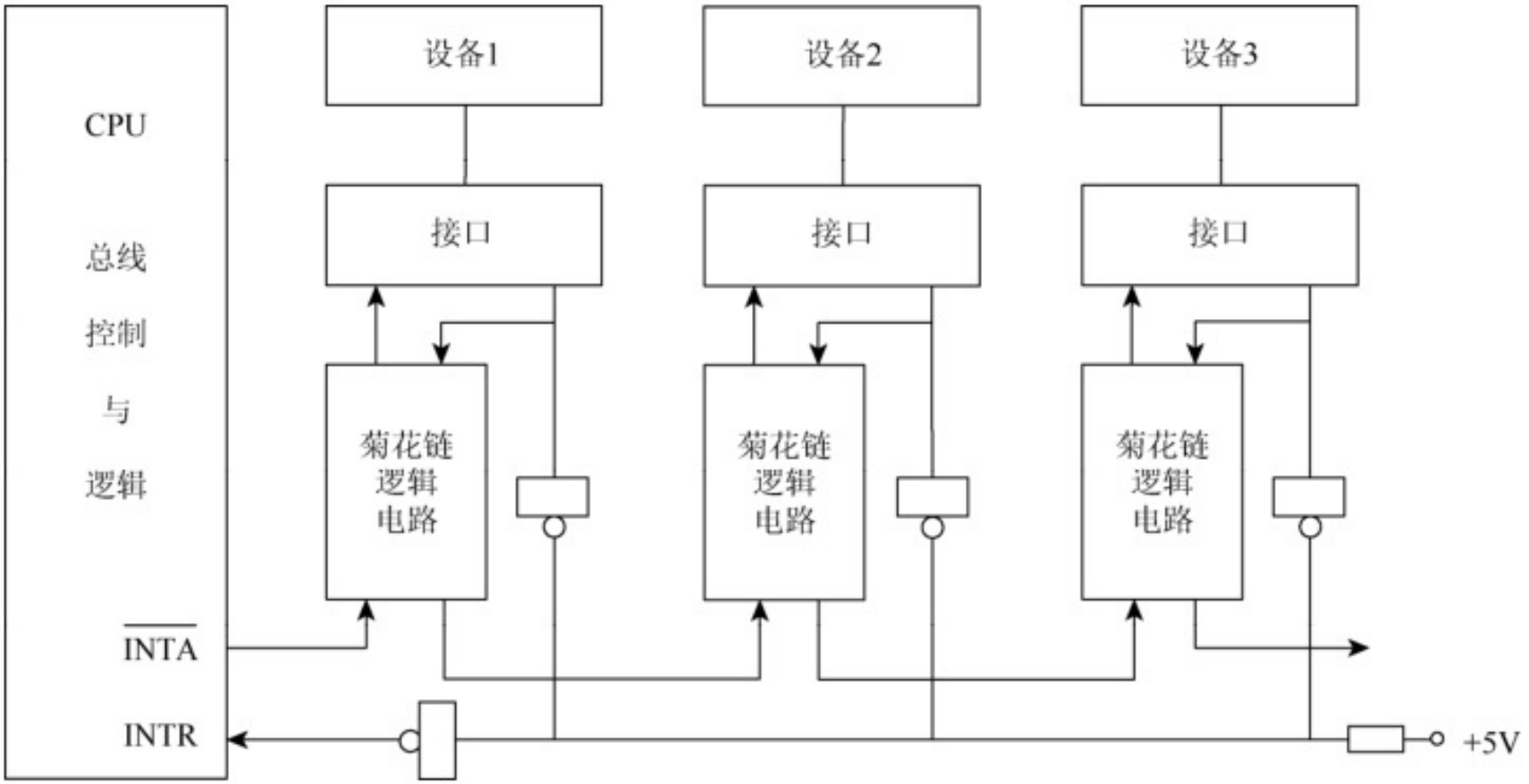



图 2.8.1　菊花链确定优先级的中断接口电路

# 图2.8.2 菊花链逻辑电路

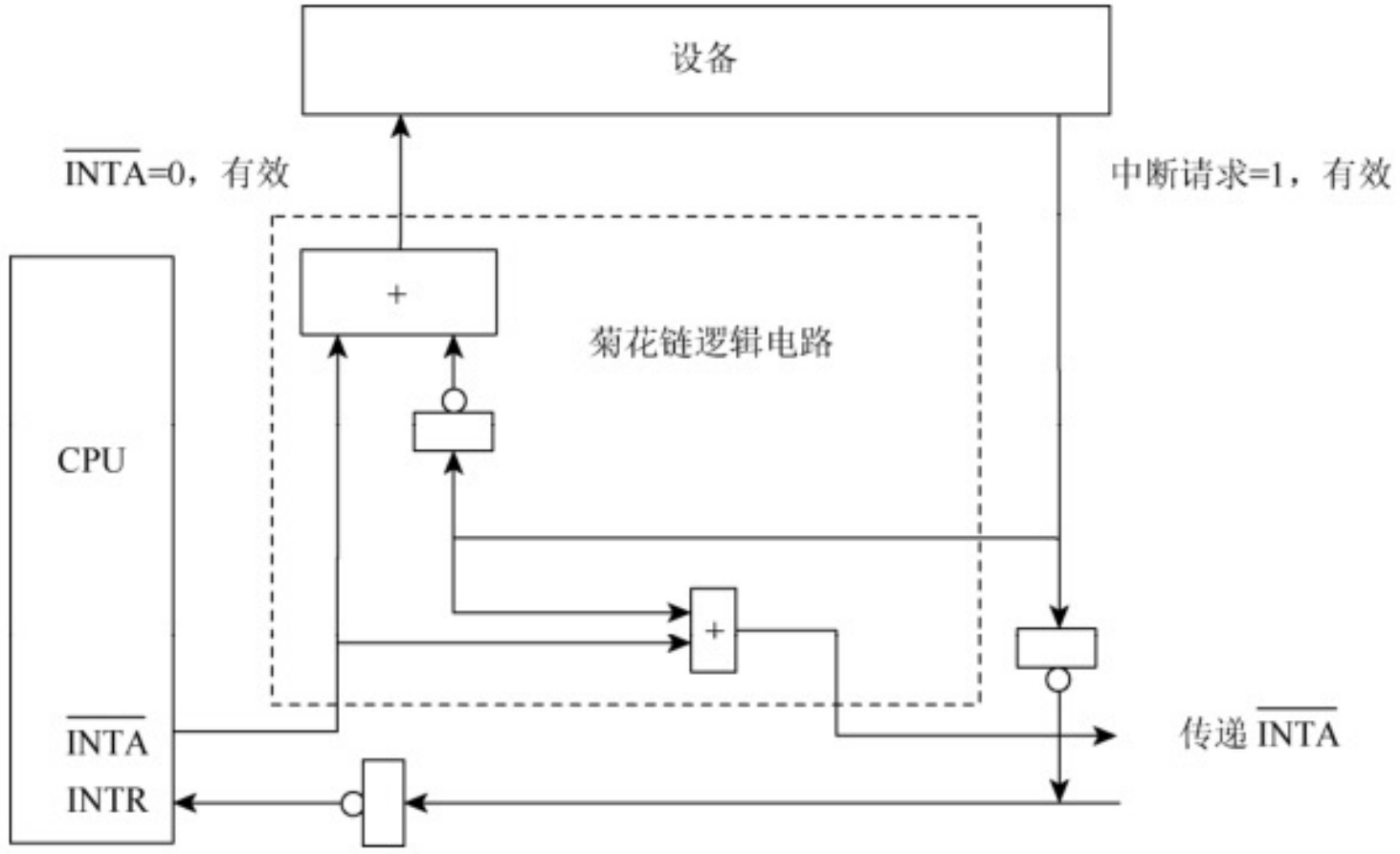



图 2.8.2　菊花链逻辑电路

# 3．CPU响应外部中断的条件

- CPU响应外部中断的条件：

  1）一条指令执行结束后。

  2）有中断请求。

  3）开中断。

# 4. 中断处理

- （1）关中断：CPU在响应中断后，发出中断响应信号，同时内部自动地关中断，以禁止接受其他的中断请求。
- （2）保护现场：为了使中断处理程序不影响主程序的运行，要把断点处的标志位的状态，推入堆栈保护起来。
- （3）保护断点：把断点处的程序计数器PC的内容压入堆栈，以备中断处理完后能正确地返回主程序断点。
- （4）形成中断服务程序入口地址：CPU要对中断请求进行处理，必须要找到相应的中断服务程序的入口地址。不同类型的微处理器的中断处理过程基本相似，在形成中断服务程序入口地址方式上有所区别。
- （5）转入中断服务程序：将中断服务程序入口地址送入程序计数器PC。
- 需要说明的是，上述过程是微处理器硬件自动完成的。

# 5. 中断服务程序

- （1）保护现场：中断服务程序像子程序一样，在执行过程中，要使用CPU中的寄存器。所以，就应该将中断服务程序中要使用的各个寄存器的内容保护起来，即保护现场。保护现场的程序段，一般放置在中断服务程序的开头，要保护多少个寄存器的内容，根据程序的需要而定。
- （2）中断服务：这是中断服务程序的核心部分，为中断源完成特定的功能。
- （3）恢复现场：在中断服务程序结束之前，应恢复现场，即把保护现场时所保存的各个寄存器的内容从堆栈弹出，送回原各个寄存器中。这样使原来被中断的程序能够正确地继续执行。否则，即使程序能够正确地返回被中断的断点，但是由于某些寄存器的内容已被破坏，不再是程序被中断时的状态，也会导致主程序产生错误的结果。恢复现场的程序段，应设置在中断服务程序的末尾，在返回指令之前。
- （4）开中断：在中断服务程序结束时，要开中断，以便CPU能响应新的中断请求。
- （5）返回断点：在中断服务程序结束的最后，要安排一条中断返回指令，将在中断处理中压入堆栈内的保护标志状态和保护断点的内容弹出，就可以返回断点执行主程序并能继续准备响应新的中断请求。

# 6. 中断的分类

- 8086可以处理256种不同类型的中断，每一种中断都规定一个唯一的中断类型编码。CPU根据中断类型编码来识别中断源。
- 8086 的中断分类：

  外部中断：外部中断源引起的中断，INTR，NMI。

  内部中断：由内部软件中断指令产生，或者在某些特定条件下，由CPU本身触发产生。
- 内部中断有两类：

  软件中断：中断指令INT n，溢出中断指令INTO，断点中断指令INT3。

  软件陷阱：除法错误中断，单步中断。

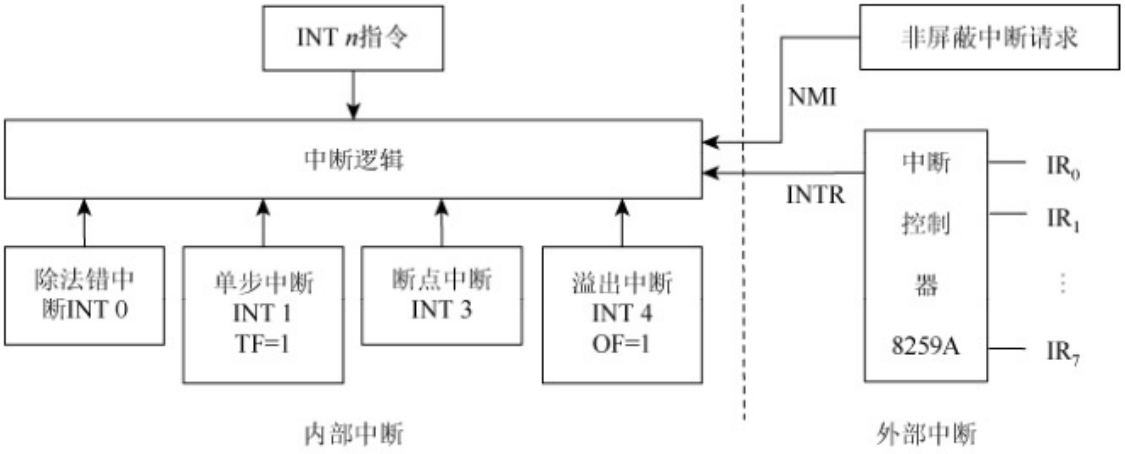



图 2.8.3　8086 微处理器的中断分类

# 2.8.2 中断向量表

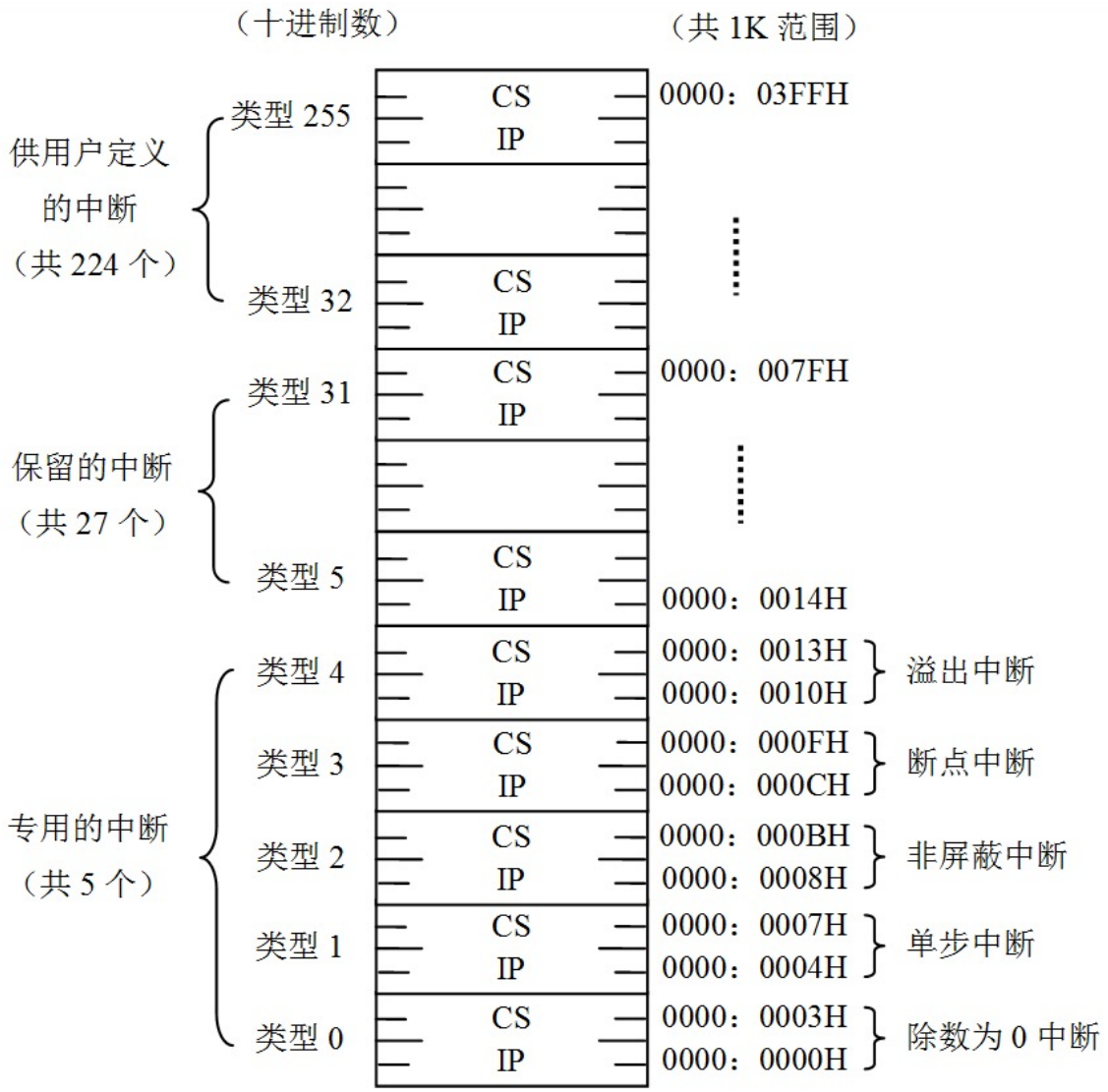



图 3.2.2 中断向量表

（1）**中断向量**：中断服务程序的入口地址。

（2）**中断向量表**：存放中断服务程序入口地址的表格。

（3）中断向量表位于存储器地址空间的低地址端，1K字节，地址为00000H～003FFH，每4个字节存放一个中断服务程序的入口地址。

（4）较高地址两个字节存放中断服务程序入口的段地址，较低地址的两个字节存放入口地址的段内偏移量，这4个单元的最低地址称为向量地址，其值为对应的中断类型码乘4。

（5）在实模式下，该表称为中断向量表。在保护模式下，该表称为中断描述符表。

# 2.8.3 可屏蔽中断

- **INTR引脚：高电平有效，当前的中断允许标志IF为1时，CPU在当前指令执行结束后，会响应INTR引脚的中断请求。**
- **可屏蔽中断请求信号是一个高电平信号，并且，INTR信号的高电平必须维持到CPU响应中断后才结束。**

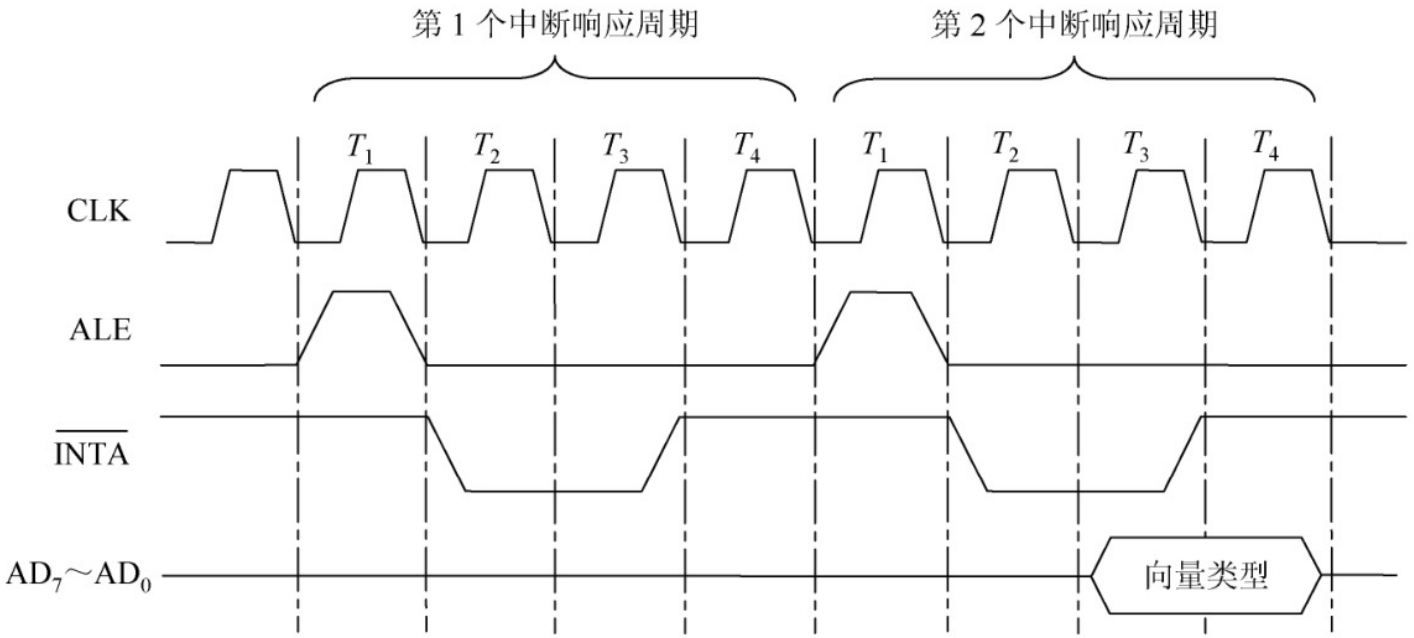



图 2.8.5　中断响应总线周期

# 2.8.4 非屏蔽中断

- 非屏蔽中断请求信号NMI用于重要事件处理，如电源掉电、存储器读写出错、总线奇偶位出错等。
- NMI中断：不受中断允许标志IF的影响。上升沿触发。中断类型码固定为2。
- 响应NMI请求时，CPU的动作和响应INTR请求时的动作基本相同，只有一个差别，就是在响应非屏蔽中断请求时，并不从外部设备读取中断类型码。这是因为从NMI进入的中断请求的中断类型码固定为2。
- 在实模式下，CPU直接从中断向量表中0000：0008H、0009H、000AH、000BH这4个单元内读取中断向量。CPU将0008H、0009H两个单元的内容装入IP，而将000AH、000BH两个单元的内容装入CS，于是就转入了非屏蔽中断服务程序的执行。

# 2.8.5 软件中断

- 软件中断包括INTO（溢出）、INT3（断点）、INT n等指令。
- 软件中断的特点：

（1）人为设置。

（2）中断类型码包含在指令中，所以不执行中断响应总线周期。

（3）中断返回地址指向软件中断指令的下一条指令。

# 2.8.6 中断处理过程

清除TF目的：避免进入中断处理程序后按单步执行。

再次查询NMI目的：使在中断响应到中断服务程序入口地址送入CS和IP期间的NMI中断请求能得到及时响应。

（1）若TEMP=1，则控制又传递给单步中断服务程序。当单步中断服务程序结束时，控制又返回原先的中断服务程序。

（2）若本次响应的中断是单步中断，则单步中断服务程序将执行两次，但不会产生一个死循环。

（3）在设置了TEMP=TF之后，CPU自动将陷阱标志位清除（是为了避免CPU以单步方式执行中断服务程序）。

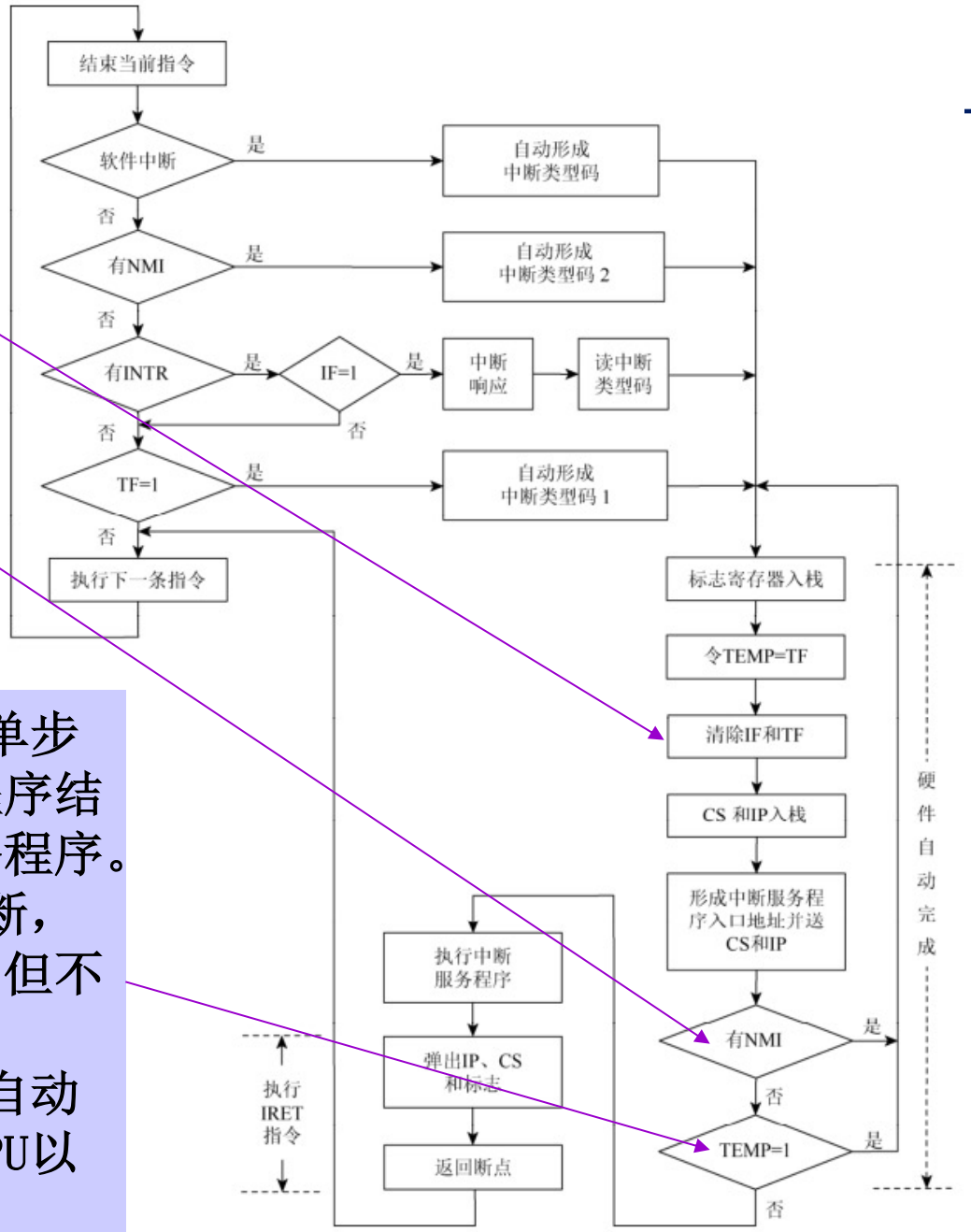
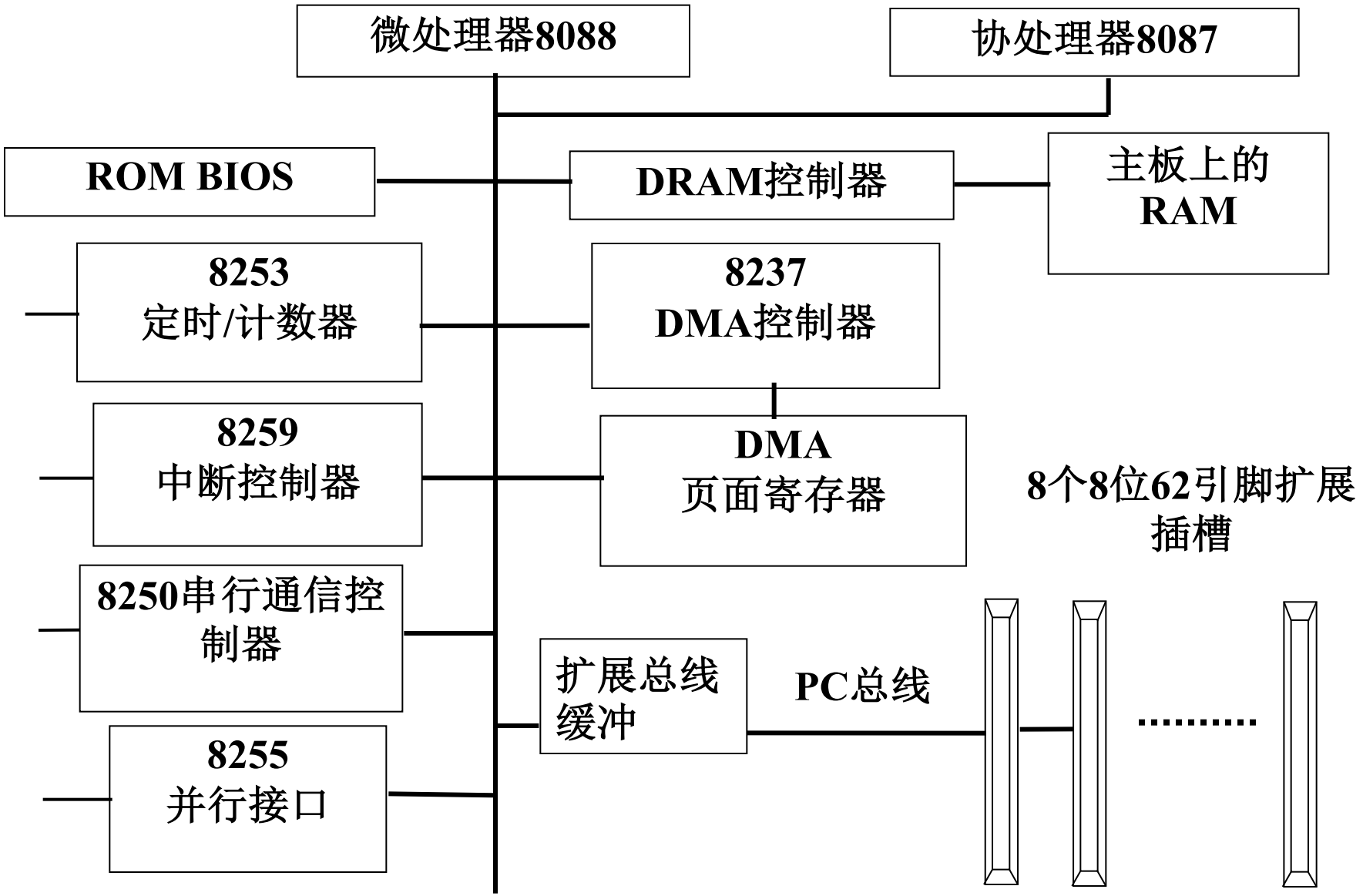



图 2.8.6　8086 微处理器对中断的基本处理过程

# PC/XT机的基本结构


微处理器8088
协处理器8087
ROM BIOS
DRAM控制器
主板上的
RAM
8253
定时/计数器
8237
DMA控制器
8259
中断控制器
DMA
页面寄存器
8个8位62引脚扩展
插槽
8250串行通信控
制器
扩展总线
缓冲
PC总线
8255
并行接口

# 结 束