《微机原理与接口》

第2章 微机原理(8088)

教师：苏曙光

华中科技大学软件学院

- 第二章 微机原理
  - 第1节 8088CPU的基本原理
  - 第2节 8088的运行（执行）环境
  - 第3节 数字电路、常用门和IC芯片
  - 第4节 8088CPU外部结构
  - 第5节 8088CPU总线时序
  - 第6节 IA-32发展历史

- 本章重点
  - 8088内部结构
  - 8088外部引脚；
  - 8088内部寄存器；
  - 8088的存储器组织；
  - 8088的工作时序
  - 数字电路回顾：常用门和IC芯片

第1节 8088CPU的基本原理

● 8086/8088 CPU

地	1	40	Vcc(+5V)
AD14	2	39	AD15
AD13	3	38	A16 / S3
AD12	4	37	A17 / S4
AD11	5	36	A18 / S5
AD10	6	35	A19 / S6
AD9	7	34	$\overline{BHE}$ / S7
AD8	8	33	MN / $\overline{MX}$
AD7	9	32	$\overline{RD}$
AD6	10	31	HOLD ($\overline{RQ}$/GT0)
AD5	11	30	HLDA ($\overline{RQ}$/GT1)
AD4	12	29	$\overline{WR}$ ($\overline{LOCK}$)
AD3	13	28	M / $\overline{IO}$ ($\overline{S2}$)
AD2	14	27	DT / $\overline{R}$ ($\overline{S1}$)
AD1	15	26	$\overline{DEN}$ ($\overline{S0}$)
AD0	16	25	ALE (QS0)
NMI	17	24	$\overline{INTA}$ (QS1)
INTR	18	23	$\overline{TEST}$
CLK	19	22	READY
地	20	21	RESET

8086

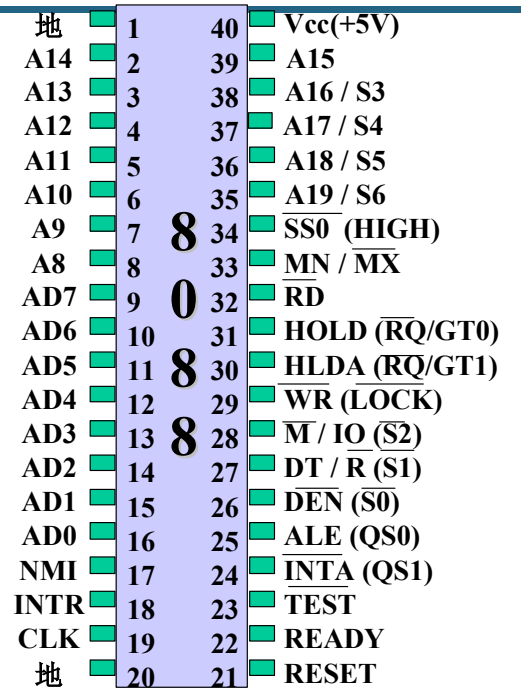
地	1	40	Vcc(+5V)
A14	2	39	A15
A13	3	38	A16 / S3
A12	4	37	A17 / S4
A11	5	36	A18 / S5
A10	6	35	A19 / S6
A9	7	34	$\overline{SS0}$ (HIGH)
A8	8	33	MN / $\overline{MX}$
AD7	9	32	$\overline{RD}$
AD6	10	31	HOLD ($\overline{RQ}$/GT0)
AD5	11	30	HLDA ($\overline{RQ}$/GT1)
AD4	12	29	$\overline{WR}$ ($\overline{LOCK}$)
AD3	13	28	$\overline{M}$ / IO ($\overline{S2}$)
AD2	14	27	DT / $\overline{R}$ ($\overline{S1}$)
AD1	15	26	$\overline{DEN}$ ($\overline{S0}$)
AD0	16	25	ALE (QS0)
NMI	17	24	$\overline{INTA}$ (QS1)
INTR	18	23	$\overline{TEST}$
CLK	19	22	READY
地	20	21	RESET

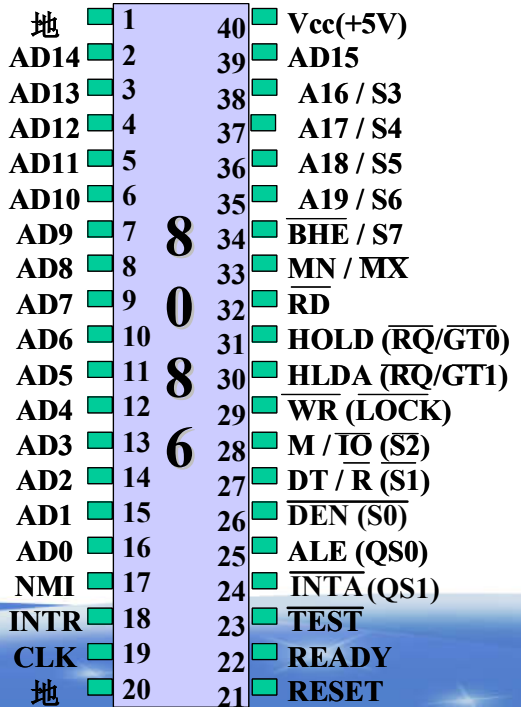
8088

8086/8088 CPU的特点

- 采用并行流水线工作方式
- 支持多处理器系统
- 片内无浮点运算部件，浮点运算由数学协处理器**8087**支持（也可用软件模拟）
  - 注：**80486DX**以后的**CPU**均将数学协处理器作为标准部件集成到**CPU**内部
- 对内存空间实行分段管理

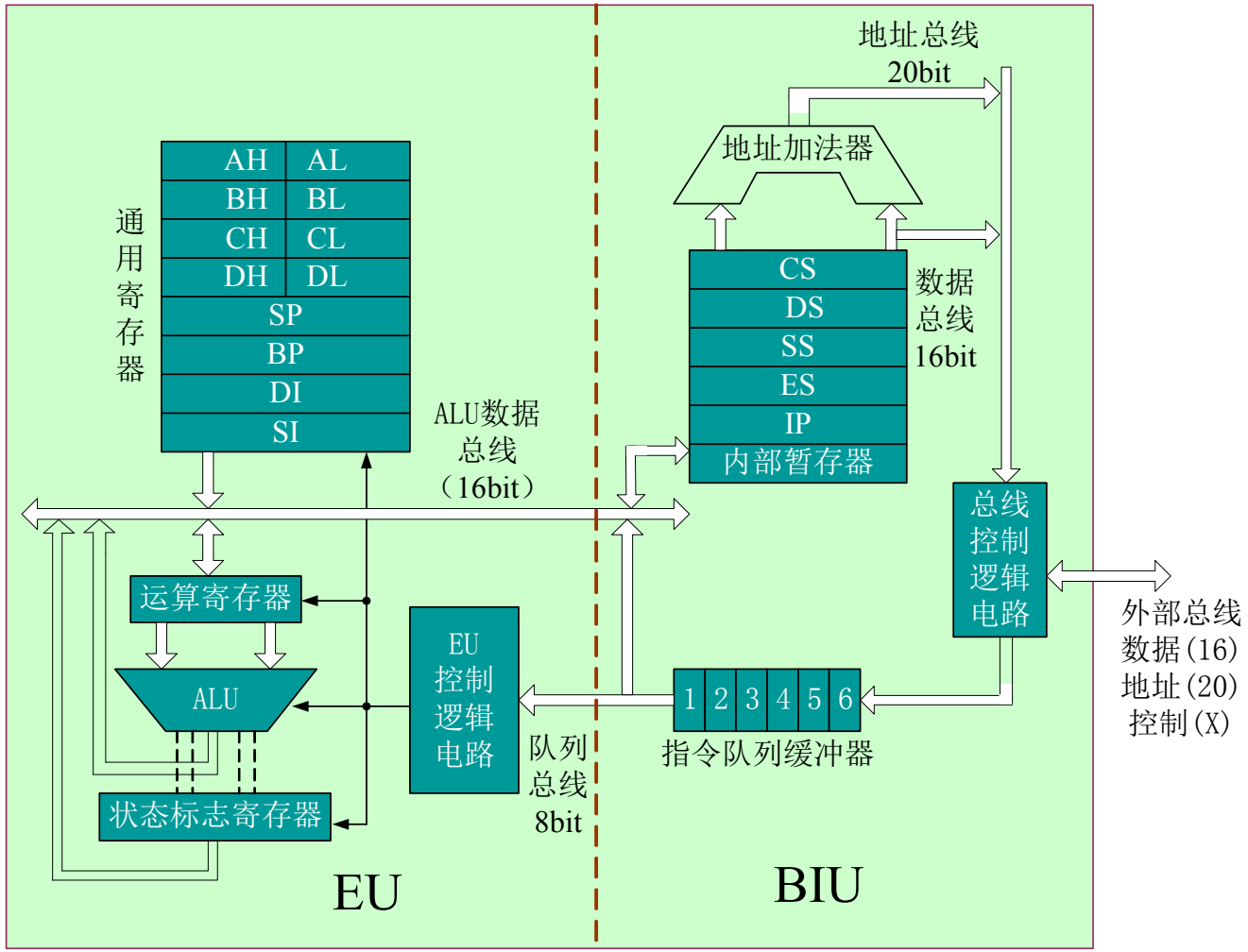
- 8086/8088 MPU
  - 相同点
    - ◆寄存器：16位；
    - ◆地址线：20根，1MB内存
  - 差异：数据总线，指令队列
    - ◆数据总线
      - □8086：16根
      - □8088： 8根 （准16位机）
    - ◆指令队列
      - □8086：6字节
      - □8088：4字节



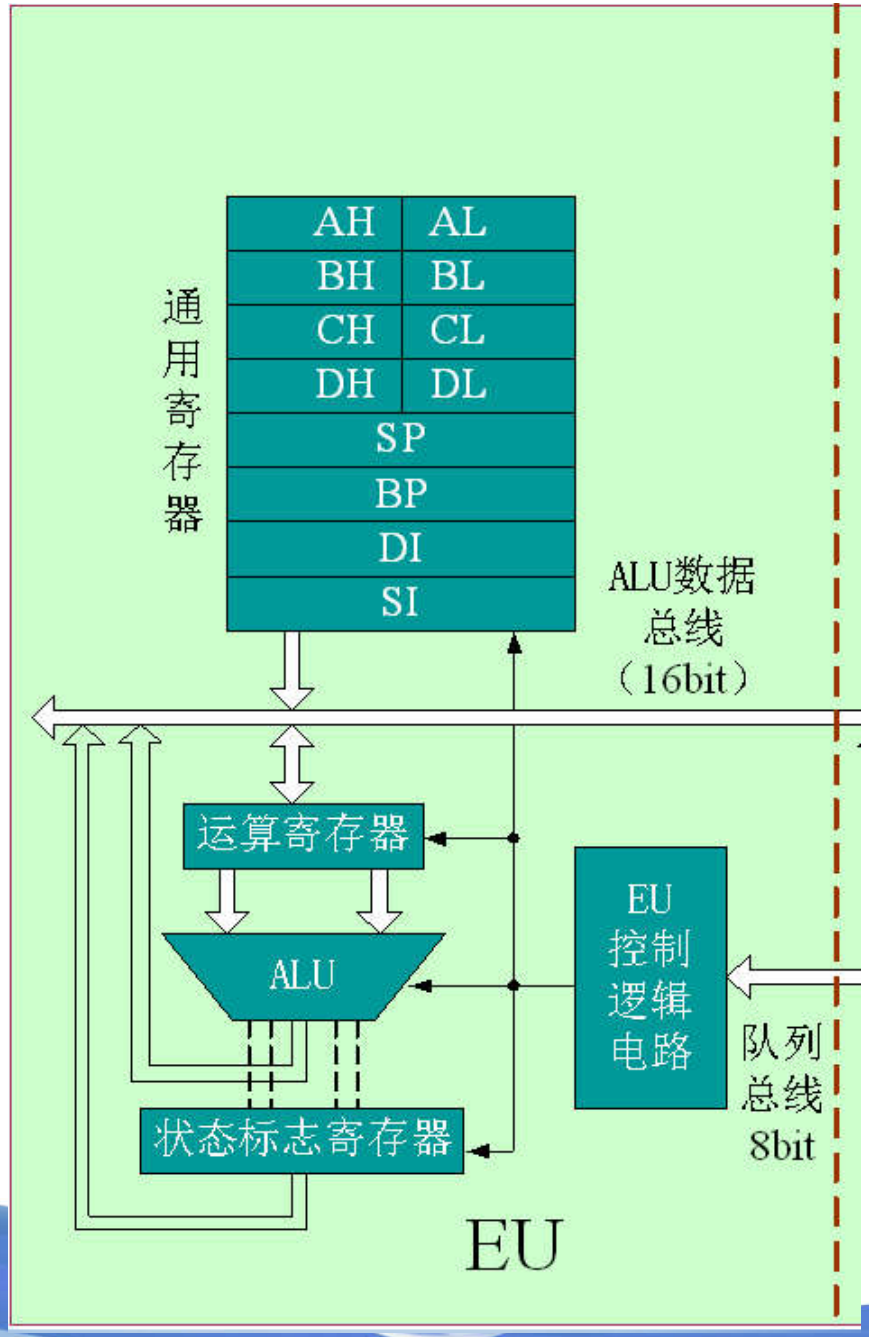


8088 内部结构：EU和BIU

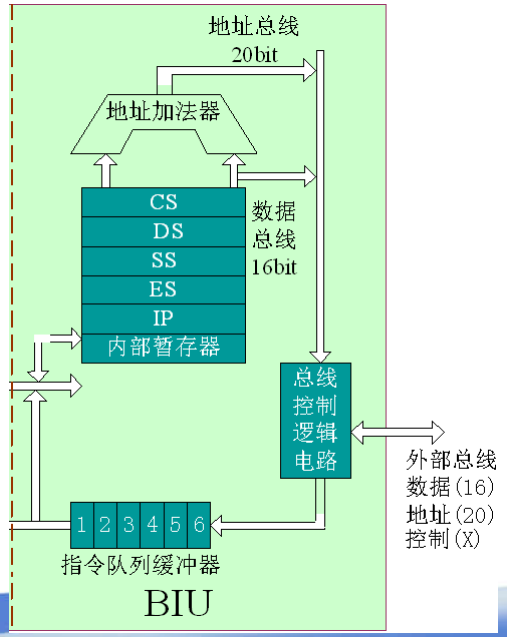
- **EU（Execute Unit，执行单元）：负责执行指令或运算**
- **BIU (Bus Interface Unit，总线接口单元）：负责读指令和数据**



EU功能和内部构成

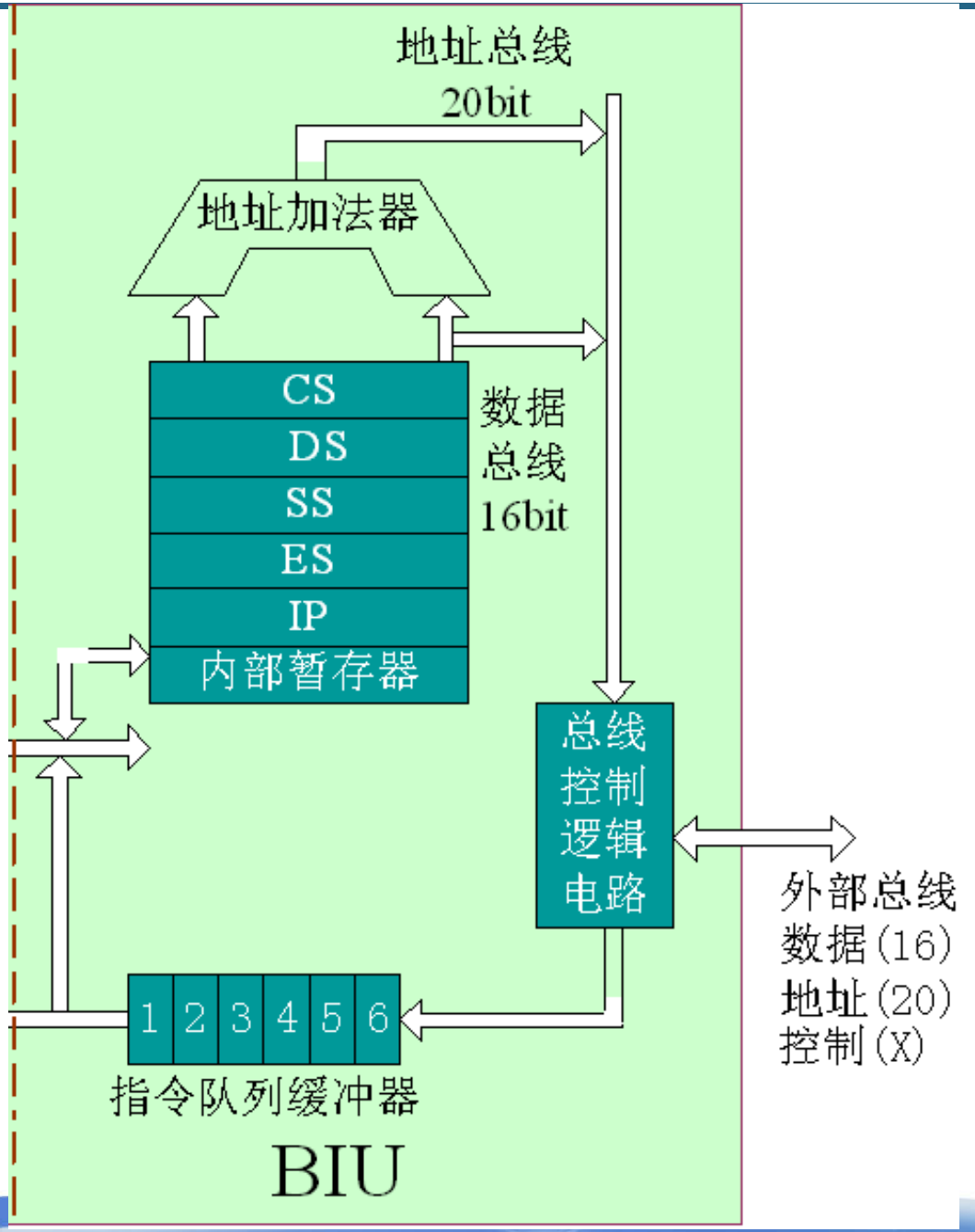


- 内部构成
  - 1）ALU：执行基本运算和处理.
  - 2）一组通用寄存器 + 标志寄存器
  - 3）EU控制系统：队列控制和时序控制
- 功能:负责执行指令或运算
  - 从指令队列中取指令代码，译码，在ALU中完成数据的运算，结果的特征保存在标志寄存器中。

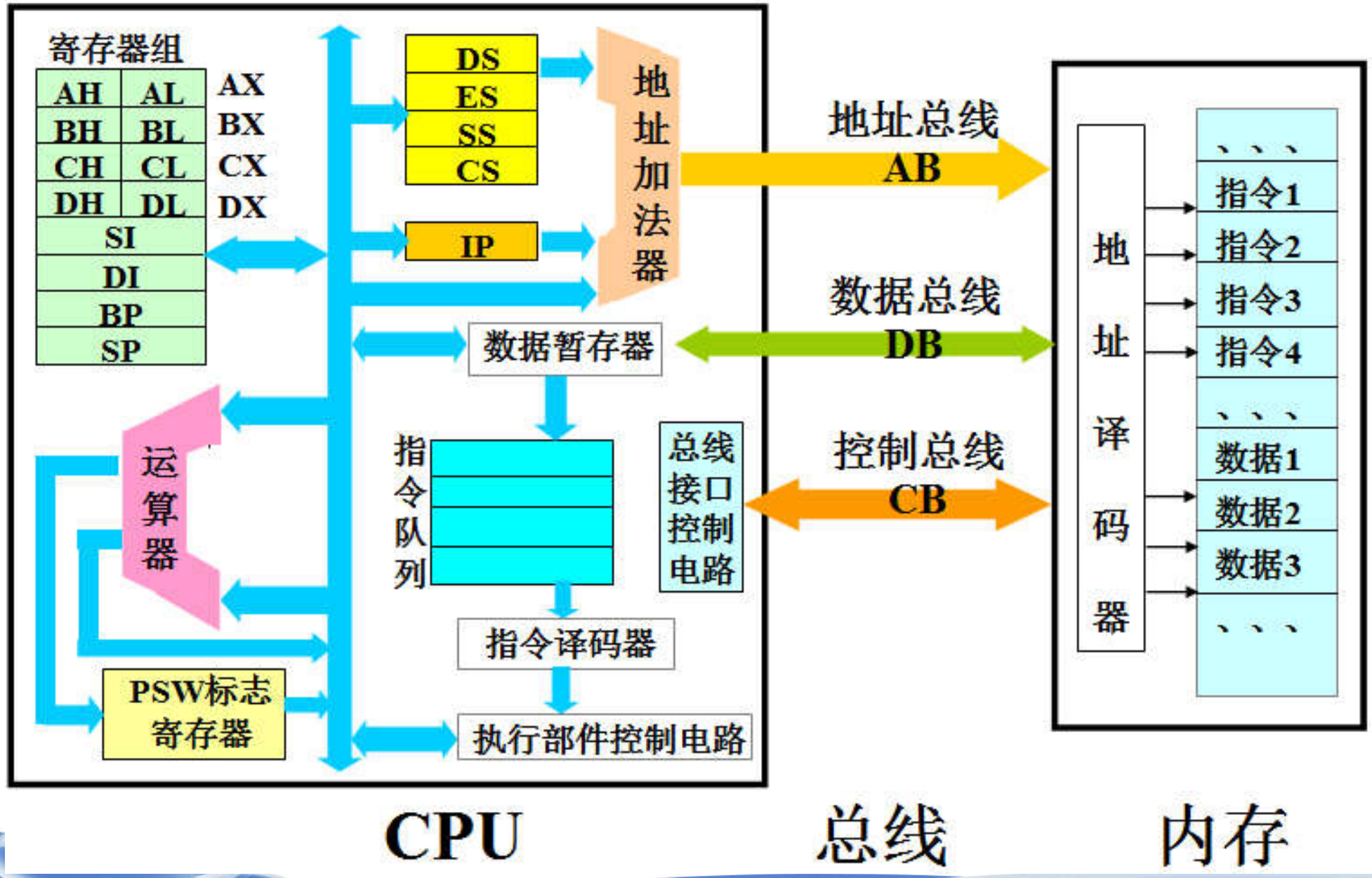


BIU功能和内部构成

- 构成
  - 1)四个段寄存器+ 指令指针IP
  - 2)地址加法器：段地扯和偏移地址相加，形成20位物理地址
  - 3)指令队列缓冲器： 6或4字节。
  - 4)总线控制逻辑：内外总线接口。
- 功能
  - 具有预取指令的功能
    - ◆执行指令的同时从内存取下一条或几条指令放在队列中。
  - 指令执行顺序
    - ◆顺序指令执行。
    - ◆执行转移指令后： 清除队列。从新地址取指并立即送往执行单元。



8088工作原理：取指令-执行指令不断循环



8088并行工作方式：流水线

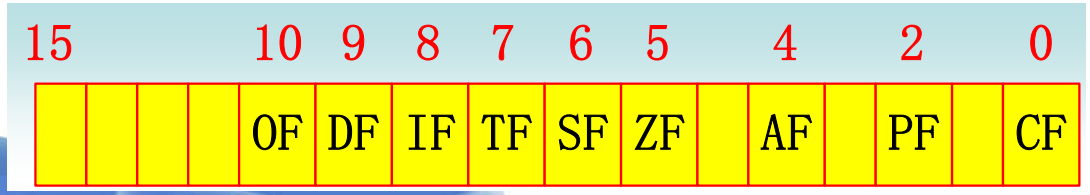
- 指令预取队列的存在使EU和BIU可同时工作
- 2级流水线

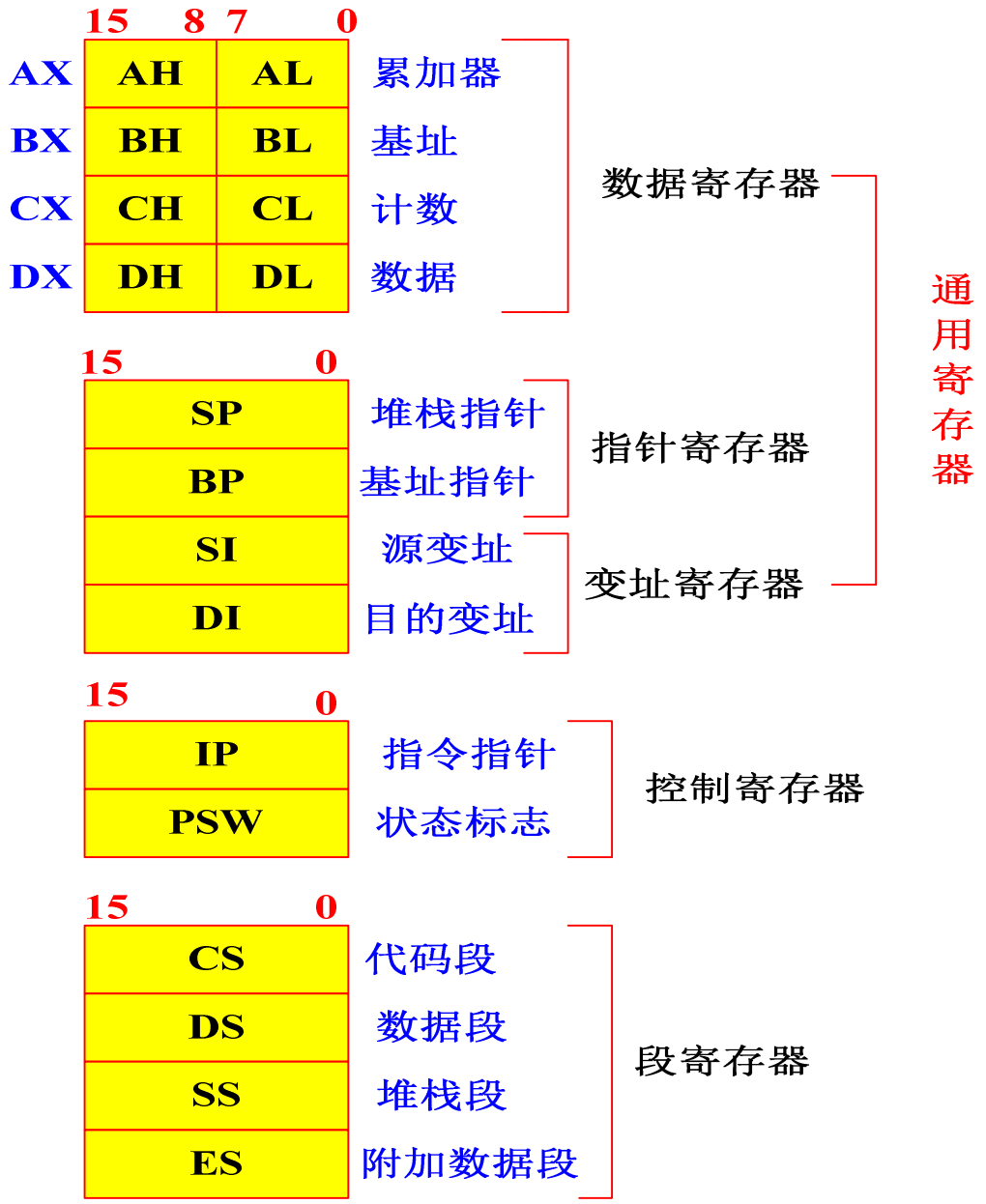
第2节 8088的运行（执行）环境

- 8088的运行（执行）环境
  - 寄存器
  - 内存空间
  - 堆栈（Stack）

14个基本寄存器

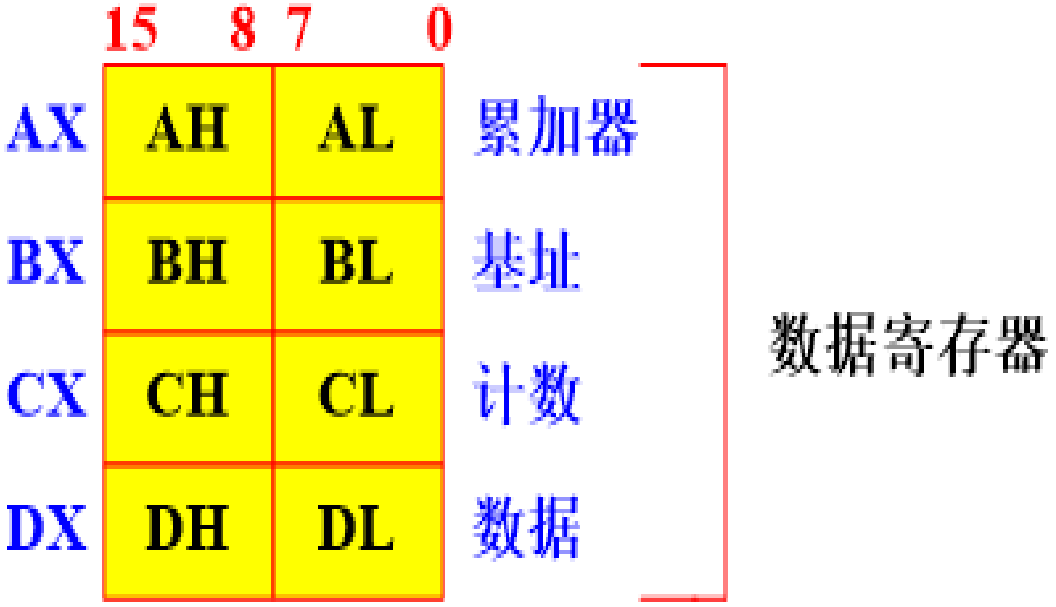
- 8个通用寄存器
  - General Registers
  - 数据和地址寄存器各4个
- 1个指令指针寄存器
  - IP: Instruction Point
- 1个状态标志寄存器
  - Flags Register;
- 4个段寄存器
  - Segment Registers
  - CS,DS,SS,ES





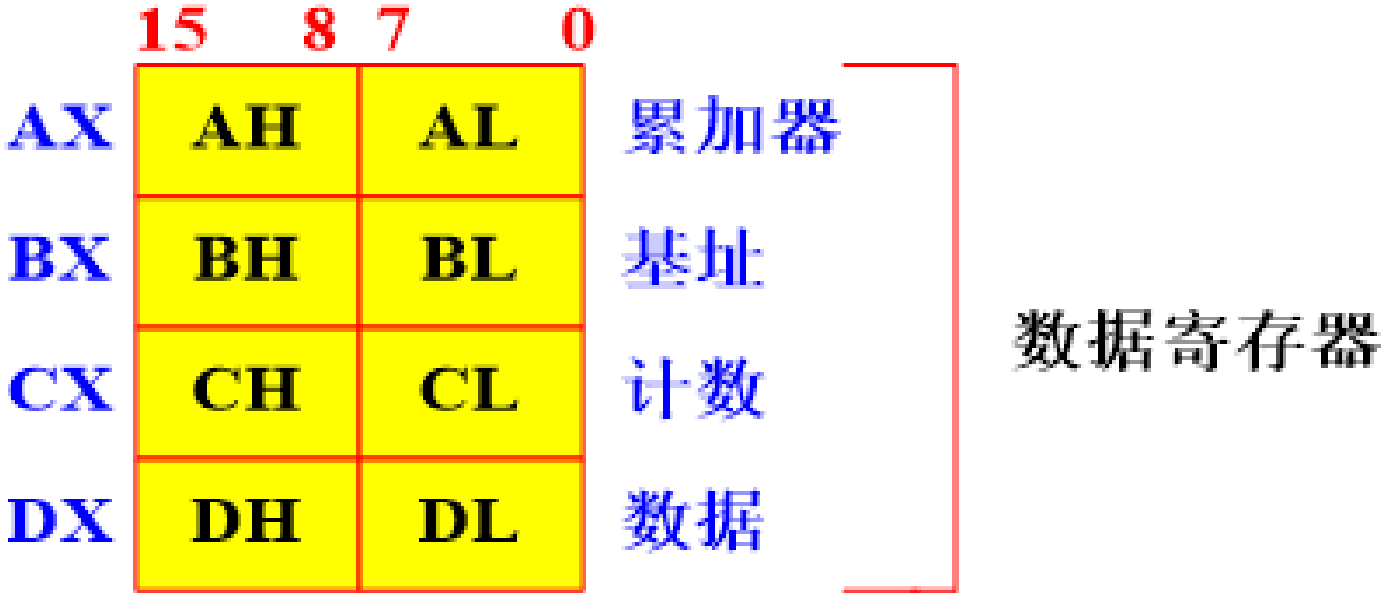
4个数据相关的寄存器：AX,BX,CX和DX

- 常用来存放参与运算的操作数或运算结果
- 16位数据寄存器，分为8个8位寄存器
  - AX： AH，AL
  - BX： BH，BL
  - CX： CH，CL
  - DX： DH，DL
- 8位可以单独操作位



4个数据相关的寄存器：AX,BX,CX和DX

- 习惯使用
  - ■AX：累加器。多用于存放运算结果及与外设信息等；
  - ■BX：基址寄存器。常用于存放内存地址；
  - ■CX：计数寄存器。循环或串操作存放循环或重复次数；
  - ■DX：数据寄存器。在32位乘除法运算存放高16位数；



4个地址相关的寄存器：SP,BP,SI,DI

- **SP,BP,SI,DI：段内寻址时存放偏移地址**

SP（stack pointer）——堆栈指针寄存器

用来指示栈顶的偏移地址，必须与SS段寄存器联合使用确定实际地址。

BP（base pointer）——基址指针寄存器

可以与SS寄存器联合使用来确定堆栈段中某一存储器单元地址。

SI——Source Index Register 源变址寄存器。

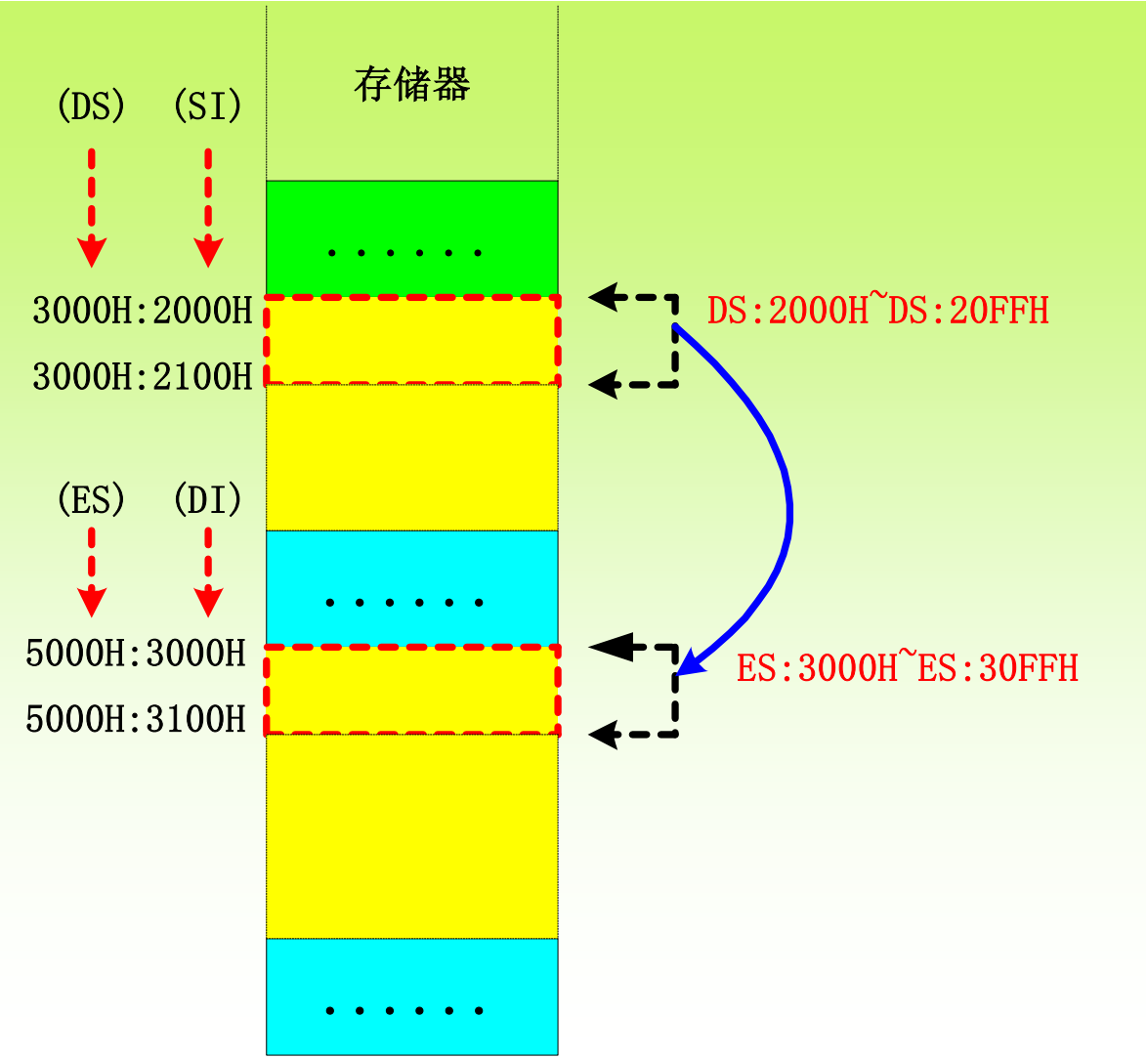
DI——Destination Index 目的变址寄存器。

- 一般与DS或ES联合使用，寻址数据段，具有自增1的功能。
  - 例：MOV AX，[SI]

4个地址相关的寄存器：SP,BP,SI,DI

- 串处理指令中**SI**、**DI**分别在数据段**DS**和附加段**ES**中寻址。
- 例：

```
……
MOV   SI，2000H
MOV   DI，3000H
MOV   CX，100H
CLD
……
MOVSB
……
```

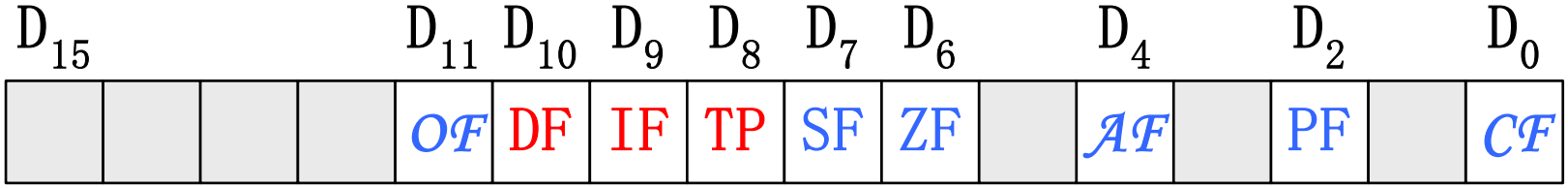


1个指令指针寄存器：IP

- 存储CPU将要执行的下一条指令的偏移地址；
- CPU在执行完一条指令之后，会自动将下一条指令的偏移地址存入到IP中。

1个状态标志寄存器：PSW

- 16位，包含9个标志位(6个状态位，3个控制位)



■状态位：由算术、逻辑运算结果结果自动设置

◆作为条件转移指令的转移控制条件

■控制位：由用户或程序直接或间接设置。

■状态位的例子

◆OF(Overflow Flag)，溢出标志位

□功能：标示符号数的运算结果是否溢出

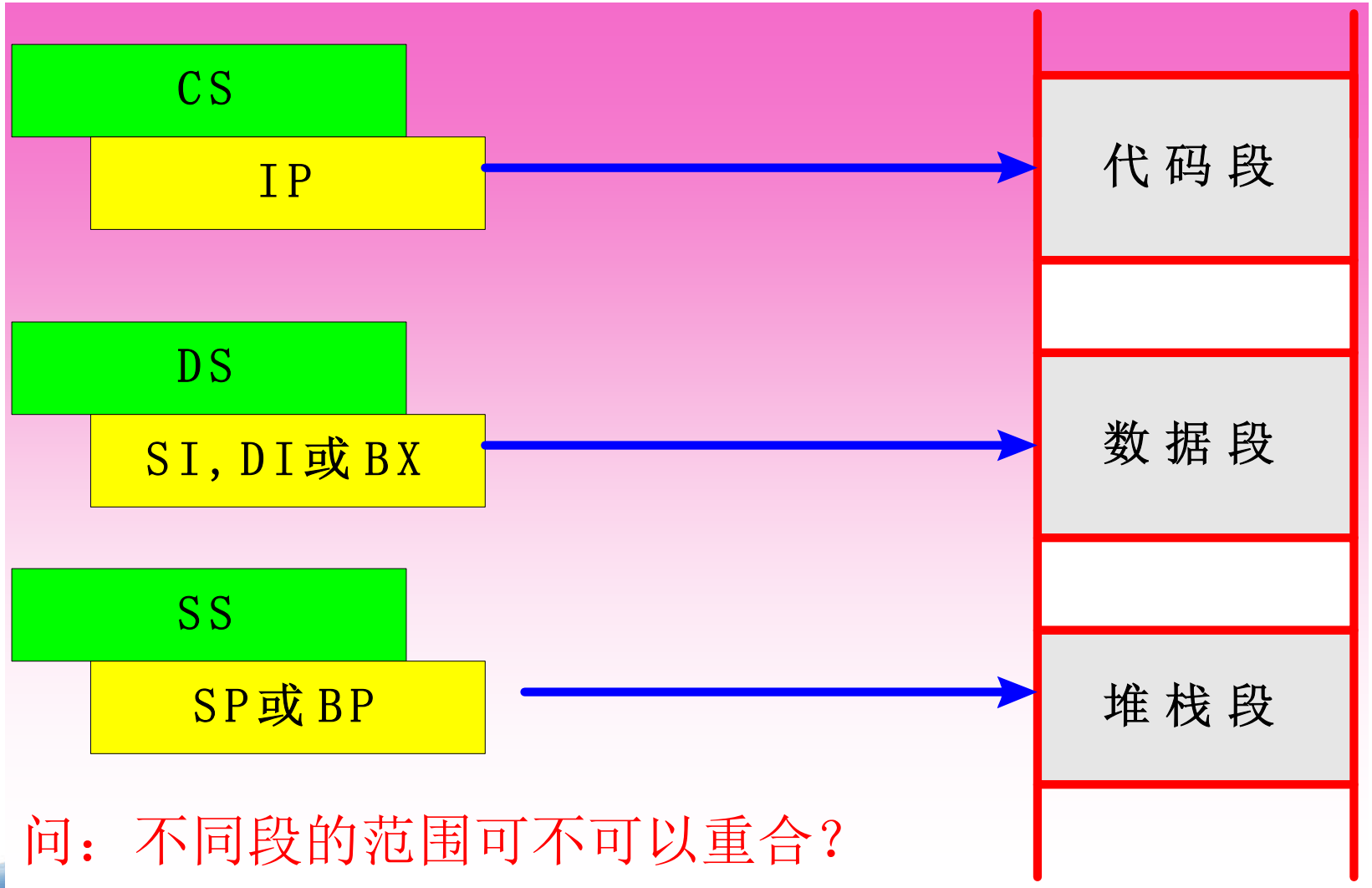
■控制位的例子

◆DF(Direction Flag)，方向位

□功能：用于控制字符串操作的地址步进方向

4个段寄存器：CS,DS,SS,ES

- 段寄存器和其他寄存器组合起来指向某个内存单元



问：不同段的范围可不可以重合？

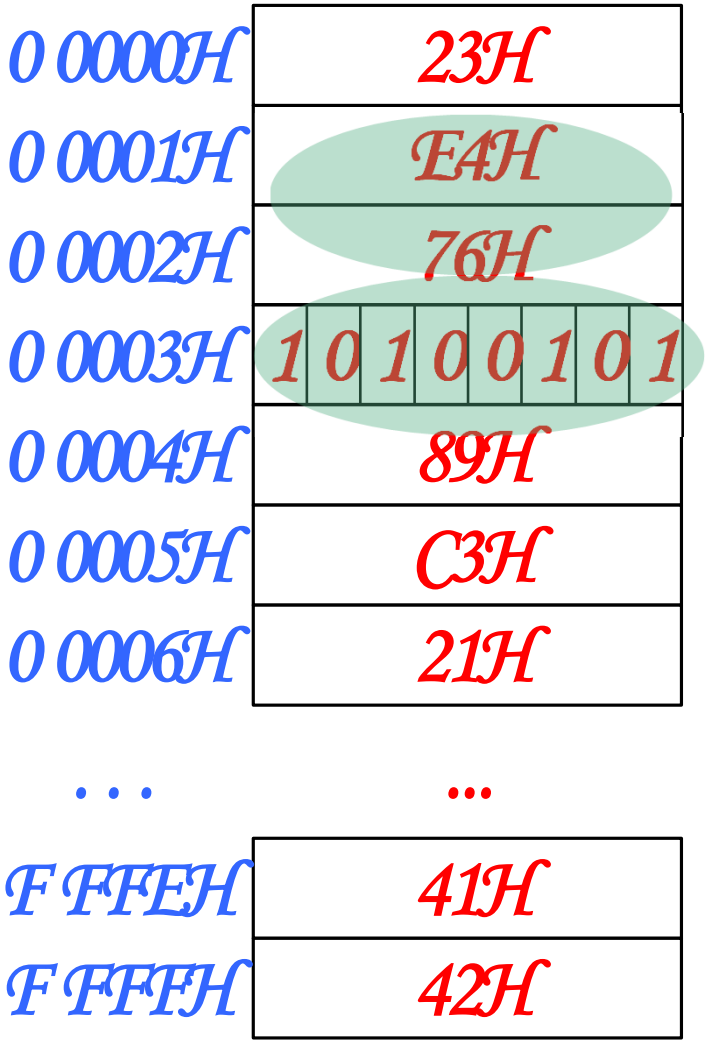
数据存放规律

- 字节数据
  - 一单元存放一个数

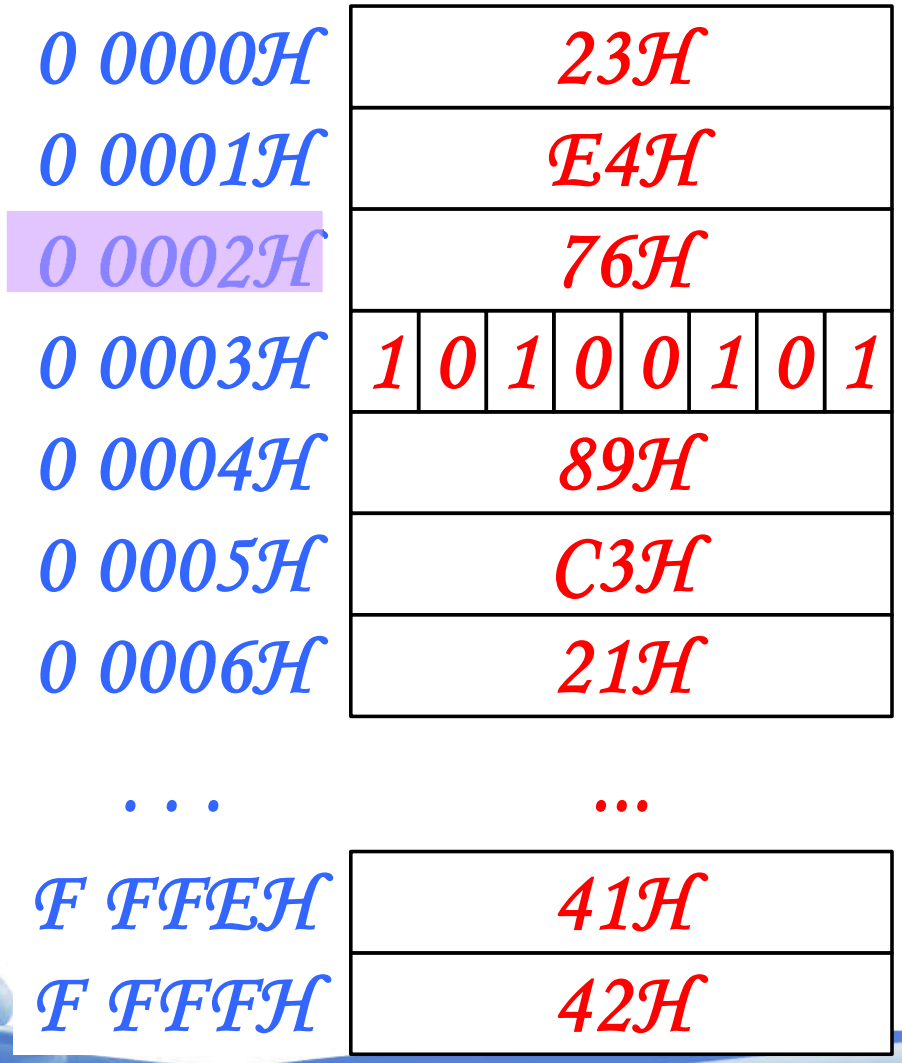
  例子：**E4H** 放在 **00001H**单元；
- 字数据：2个字节
  - 2单元：“低对低，高对高”
  - 字的地址：2 个单元中的低地址

  例子：**76E4H**放在**00001H**地址中
- 机器指令(机器码)：多个字节
  - 按字节顺序地址递增存放

  如：**MOV BX,AX ;89C3H，4H**单元
- 字符串：多个字节
  - 按字节顺序地址递增存放，同机器指令。

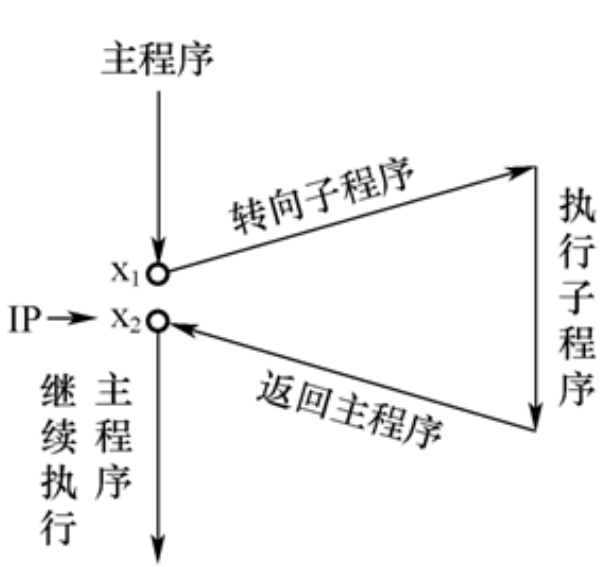


- 练习题
  - ■00002H单元存放的字节/ 字/ 双字节指令为多少？

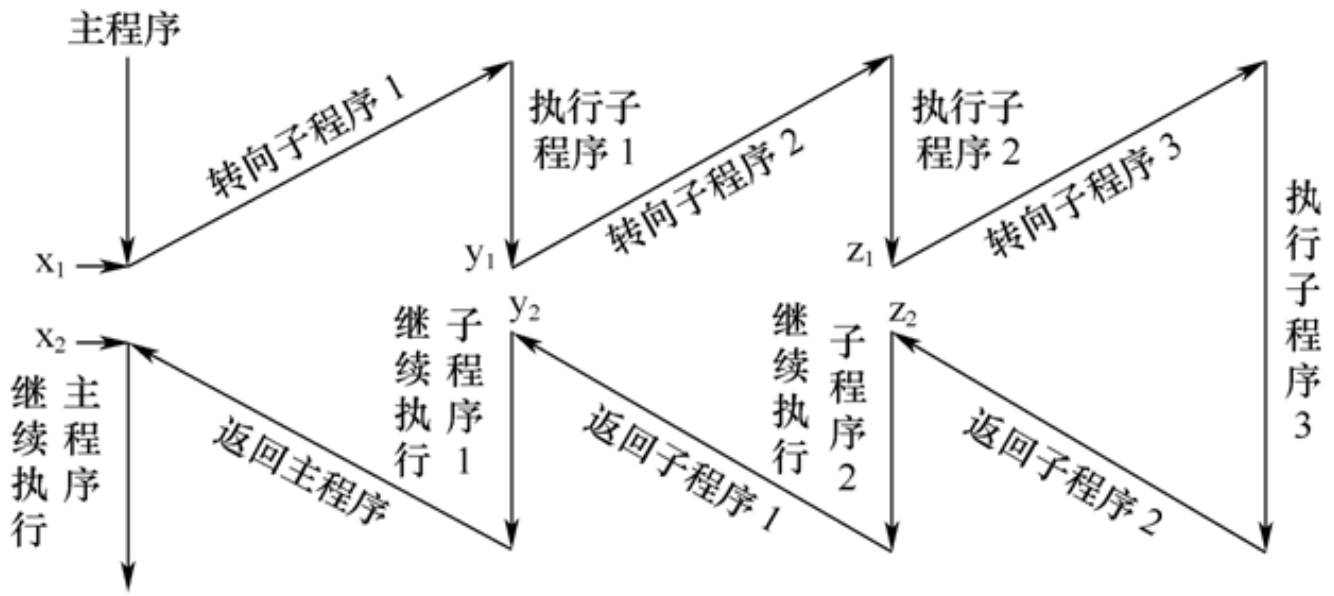


堆栈

- 例子：子程序调用的过程



(a)



(b)

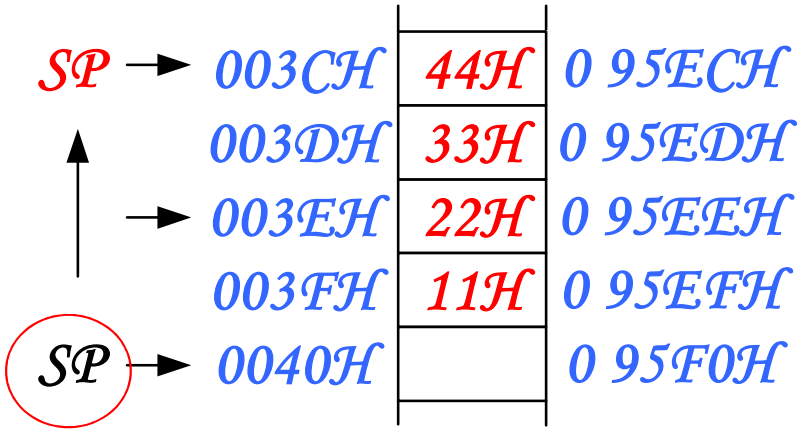
- 调用发生后，主程序在CPU中的运行环境被破坏。
- 调用返回时，必须恢复主程序之前的运行环境

- 堆栈（STACK）
  - 在子程序调用和中断服务时存储参数和现场数据；
  - 特殊内存
    - “后进先出”(LIFO)存储
    - 堆栈一端固定(栈底)，另一端活动(栈顶)，数据只允许从栈顶存取（进或出）
    - 栈指针：指示栈顶位置（Stack Poniter，SP）
  - 堆栈的伸展方向
    - 栈底的地址大，栈顶的地址小
  - 栈的操作（PC）
    - 入栈：将一个数存入栈顶，并改变SP （变小）
    - 出栈：从栈顶读出一个数据,并改变SP （变大）

- 入栈操作
  - **PUSH SRC；SRC** 代表寄存器或存储单元地址
  - 功能：将寄存器或存储单元中的一个字压入堆栈
  - 操作：
    - “先减后入”：**SP-1 → SP，** 字高位 → **[SP]**
      SP-1 → SP， 字低位 → **[SP]**
  - 结果：**SP-2,** 数据高对高，低对低存放。

- 入栈操作
  - **PUSH SRC；SRC** 代表寄存器或存储单元地址
  - 功能：将寄存器或存储单元中的一个字压入堆栈
  - 操作：
    - “先减后入”：**SP-1 → SP，** 字高位 → **[SP]**
      SP-1 → SP， 字低位 → **[SP]**
  - 结果：**SP-2,** 数据高对高，低对低存放。
  - 例：**AX=1122H，BX=3344H**
    SS=095BH，SP=0040H
    执行：**PUSH AX ;SP=** ？
    PUSH BX ;SP= ？

- 入栈操作
  - **PUSH SRC；SRC** 代表寄存器或存储单元地址
  - 功能：将寄存器或存储单元中的一个字压入堆栈
  - 操作：
    - “先减后入”：**SP-1 → SP，** 字高位 → **[SP]**
      SP-1 → SP， 字低位 → **[SP]**
  - 结果：**SP-2,** 数据高对高，低对低存放。
  - 例：**AX=1122H，BX=3344H**
    SS=095BH，SP=0040H
    执行：**PUSH AX ;SP=003EH**
    PUSH BX ;SP=003CH



- 出栈操作
  - **POP DST ；DST** 代表寄存器或存储单元地址
  - 功能：将栈顶一个字传送到寄存器或存储单元中
  - 操作
    - “先出后加”：**[SP]** → 字低位， **SP+1** → **SP**
      [SP] → 字高位， **SP+1** → **SP**
  - 结果：**SP+2**, 数据低对低,高对高存放
  - 例：上述前一例子中再执行：

POP CX ；CX=3344H, SP=003EH

POP DX ；DX=1122H, SP=0040H

- **Flag寄存器出/入栈**
  - ■命令格式

    PUSHF; F入栈， SP-2 →SP

    POPF ; F出栈， SP+2 → SP
  - ■功能： 保护和恢复状态标志寄存器**Flag**
- 注意：
  - ■栈操均以字为单位，下列指令均错：

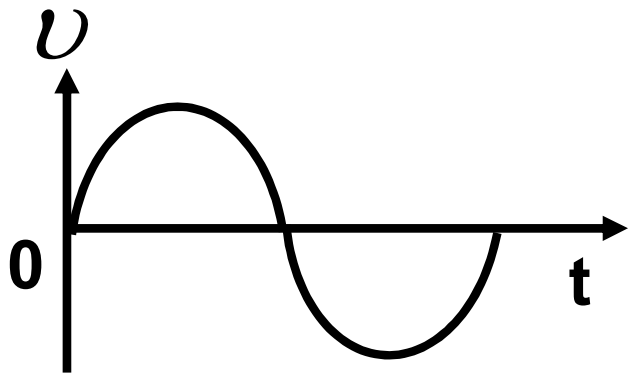
    PUSH AL

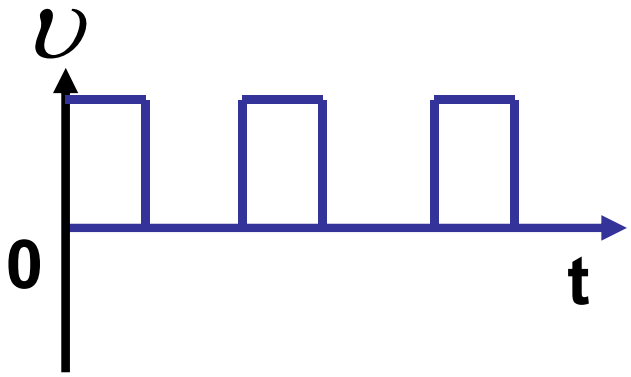
    POP DH
  - ■**PUSH**与**POP**成对，避免堆栈溢出或程序出错；
  - ■堆栈实为内存区，还可按数据区的方法对其操作。

第3节 数字电路、常用门和IC芯片

- 本节基本内容
  - 数字电路基本概念
  - 常用门电路
  - 常用IC芯片

- 电路中两类信号
  - ■模拟信号：在时间上和幅值上均连续的信号
  - ■数字信号：在时间上和幅值上均离散的信号





- 两类电路
  - ■模拟电路：处理模拟信号的电路
  - ■数字电路：处理数字信号的电路

数字电路中的基本概念

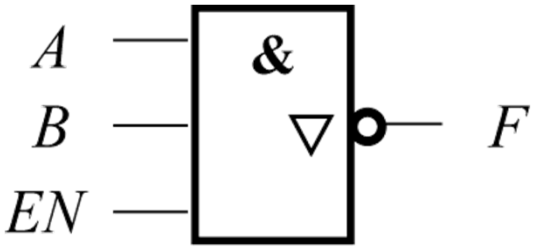
- 0和1：两个数字，高低两种电平，两种逻辑，…
- 电路中半导体管工作在开关状态
  - 二极管工作在导通态和截止态
  - 三极管工作在饱和态和截止态
- 逻辑运算
  - 基本逻辑运算
    - 与、或、非。
  - 复杂逻辑运算
    - 通过三种基本逻辑运算来实现。

复合逻辑运算

- 基本逻辑运算的复合叫做复合逻辑运算。而实现复合逻辑运算的电路叫复合逻辑门。
  - 与非门
  - 或非门
  - 异或门
  - 同或门

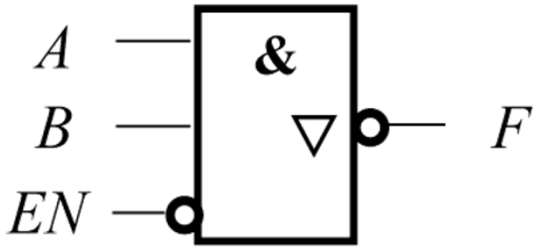
其它逻辑门：三态门

- 三态门（简称TS门，有▽符号）
  - ■0、1、Z（高阻状态或禁止态)。
- 在普通门上增加一个使能端（EN）
  - ■EN有效：门按原逻辑工作，输出0或1；
  - ■EN无效：门输出Z（高阻态）。
- 例：三态门的两个例子



EN =1 ： 使能

EN = 0 ： 失能，输出Z态



EN =0 ： 使能

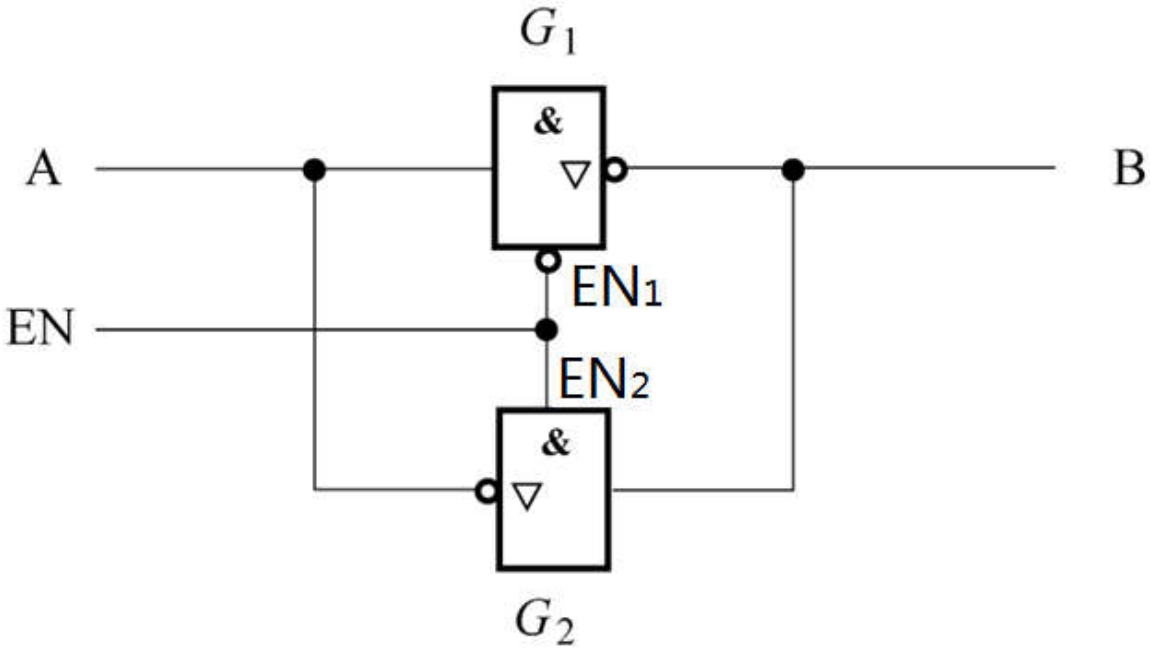
EN = 1 ： 失能，输出Z态

三态门的典型应用

- 数据传输方向控制
- 总线存取控制
- 模拟开关

数据传输方向控制

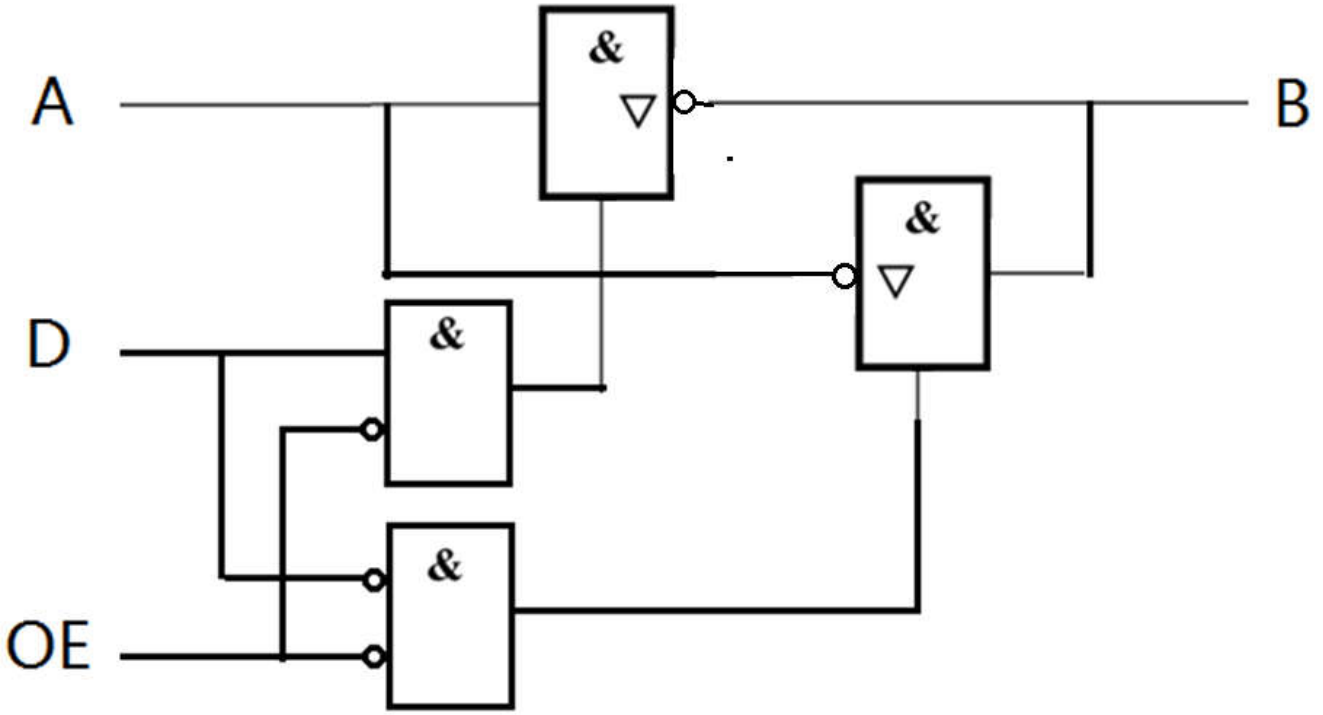
- A和B间数据传输方向可选，利用EN控制传输方向？



- 思考：
  - EN=1时，数据能从 B 端传到 A 端；
  - EN=0时，数据能从 A 端传到 B 端；

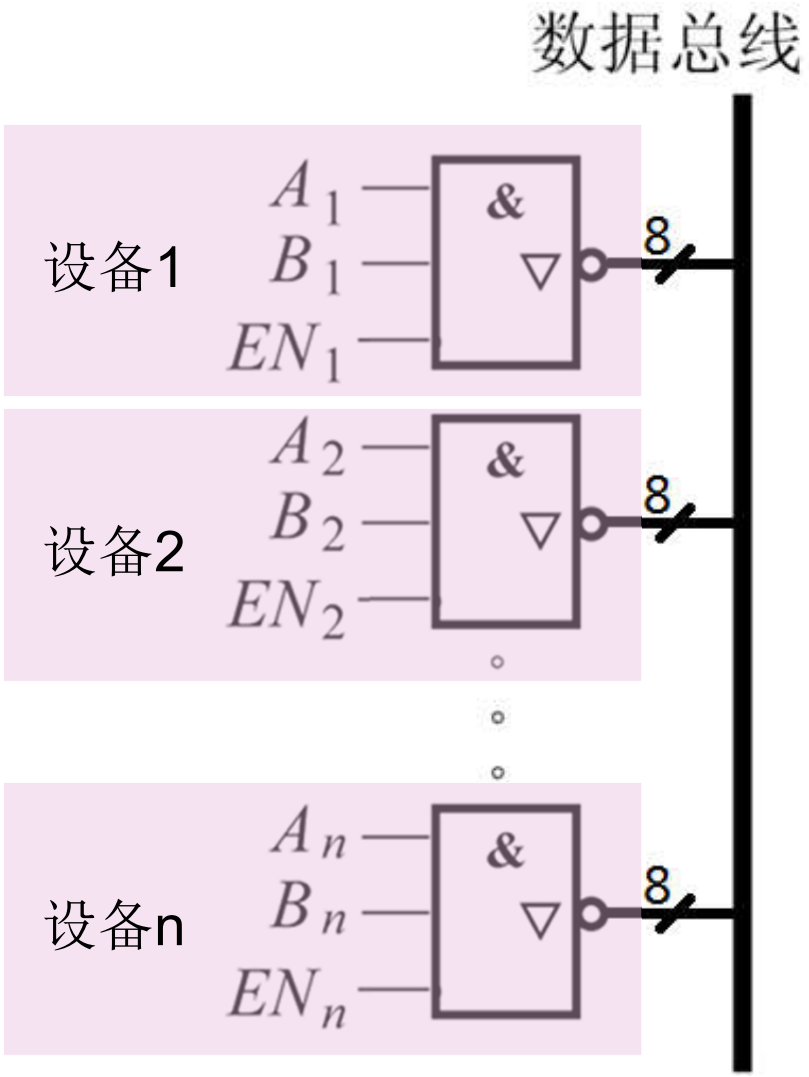
数据传输方向控制

- A和B之间传输方向可选：D控制方向,OE使能端。



- 思考：
  - OE = 1 时, A和B之间高阻
  - OE = 0 时, A和B之间连通
    - D = 0 时，数据能从 B 端传到 A 端；
    - D = 1 时，数据能从 A 端传到 B 端；

总线存取控制



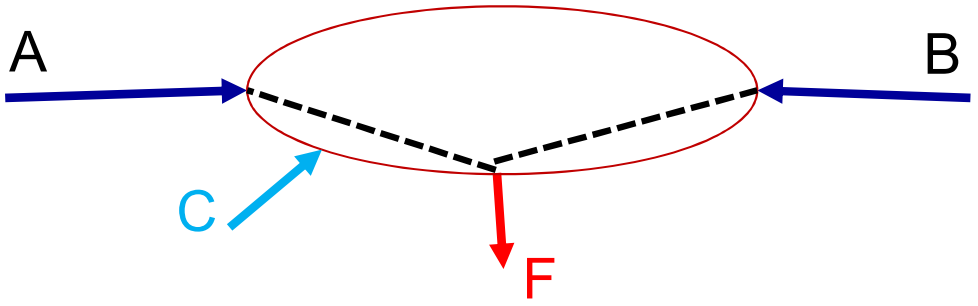
多个设备连在数据总线上，必须控制存取权限，任何时候只能让最多1个设备逻辑连接（占用）总线

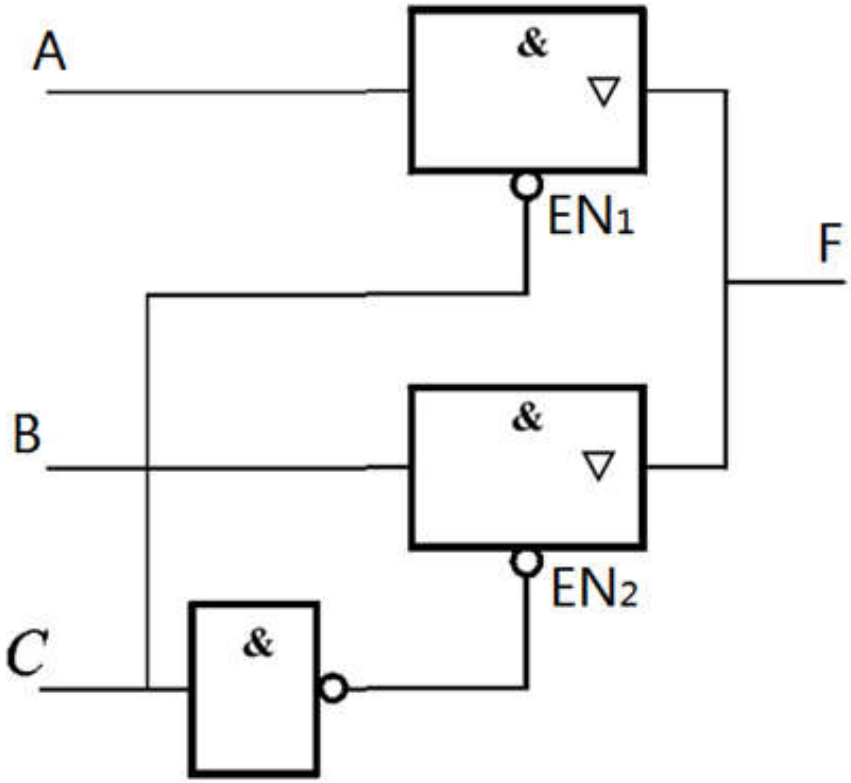
ENi ＝0： 设备i脱离总线（Z态）

ENi ＝1： 设备i逻辑连上总线

模拟开关

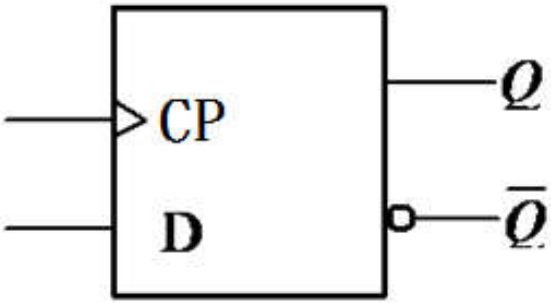
- 例子：单刀双掷开关
  - A,B输入，F=A或B？
- 思考
  - 若C=0，F = A还是=B?
  - 若C=1，F = A还是=B?
- 答案
  - C = 0
    - TG_1通，$F = A$；
  - C = 1
    - TG_2通，F = B。





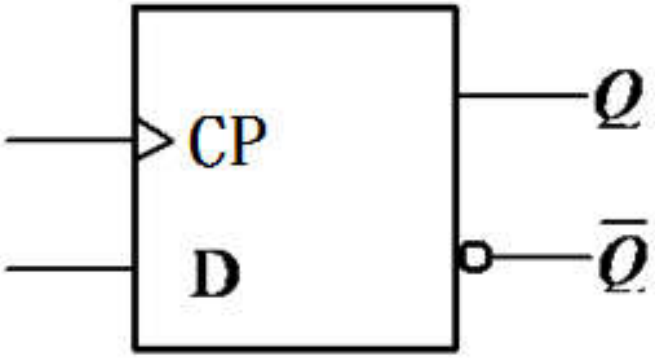
触发器——D触发器

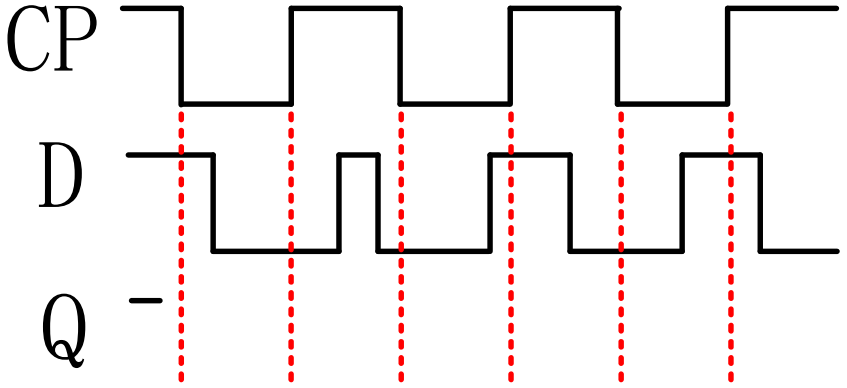
- 触发器的功能和特点
  - 能储存一位二进制信息的单元电路。
  - 用于信号保持
  - 用于导通开关
  - 特点：0-1双稳态电路



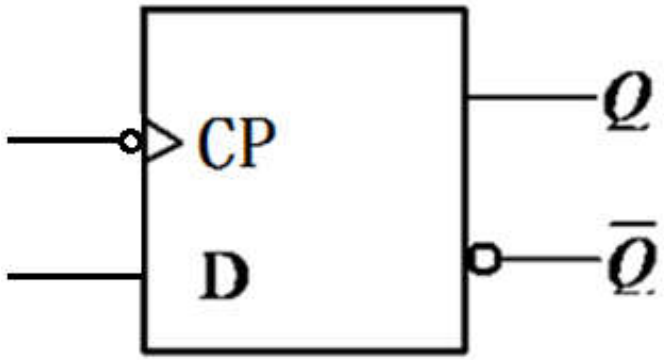
- 触发器与门的联系
  - 联系：触发器是在门电路的基础上引入反馈构成的。
  - 区别：门是组合电路，触发器是时序电路。
- 触发器的种类
  - 基本RS触发器、同步RS触发器、主从型JK触发器、维持阻塞型D触发器、T和T'触发器等。

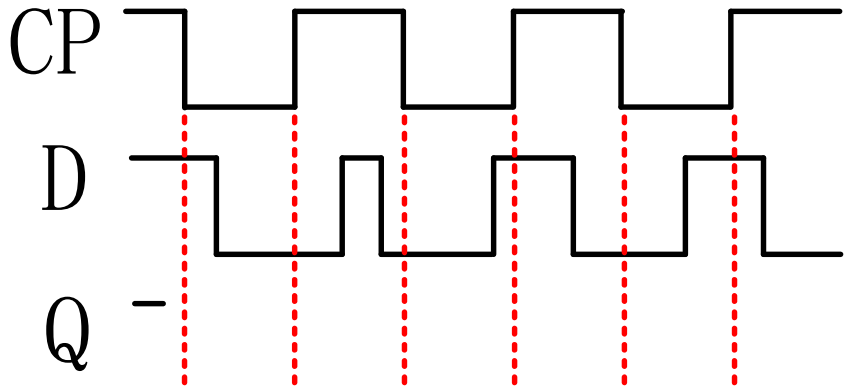
- 工作方式
  - CP上升沿锁存D (阻塞D)，上升沿后Q端保持不变。
  - 触发器在CP脉冲的上升沿产生状态变化:Q=D。而在上升沿后，D端信号变化对触发器输出状态没有影响。触发器的次态取决于CP脉冲上升沿时的D信号，



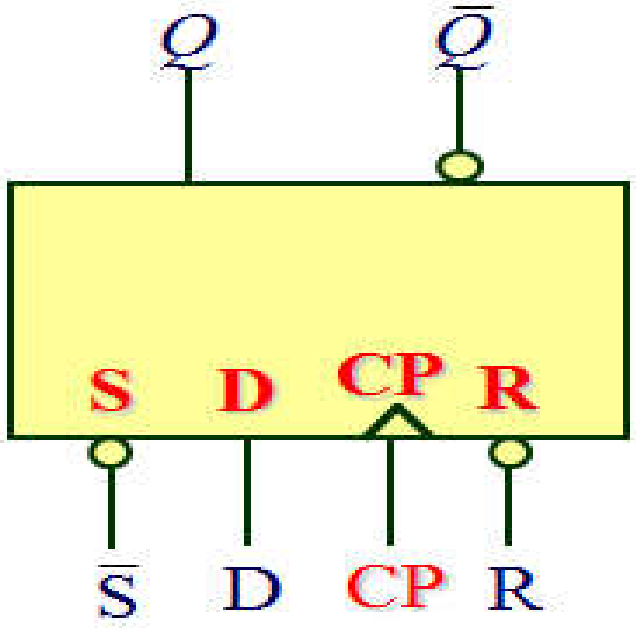


- **CP下降沿触发的触发器**

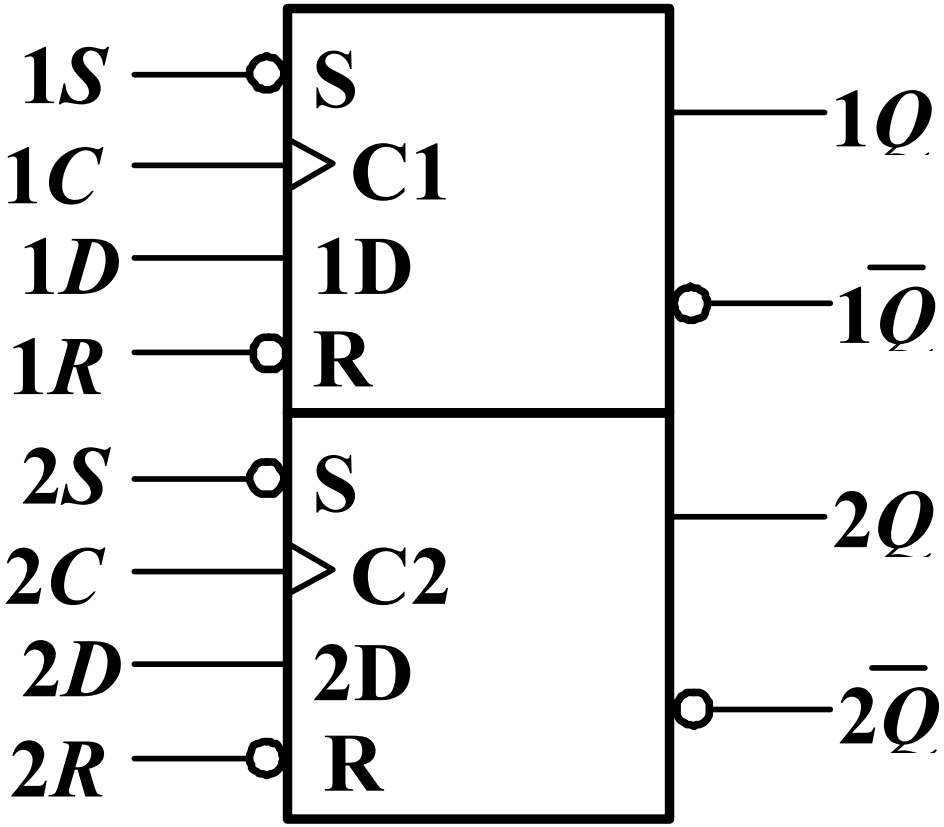




- 带清零和置1端的D触发器
  - R：置0端
  - S：置1端

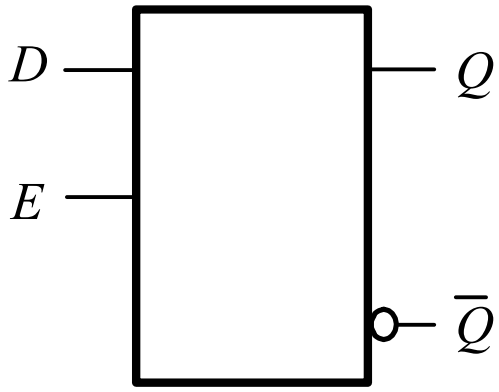


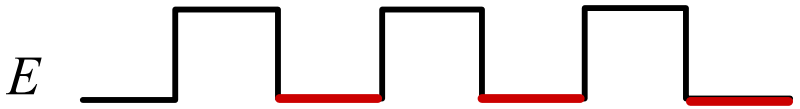
- D触发器集成电路
  - 74HC74
  - 2D触发器



锁存器——D锁存器

- 和触发器有类似的功能
  - 具有0和1两个稳状，能自行保持。
  - 能存储一位二进制码。
- 区别
  - 锁存器对电平敏感，触发器对边沿敏感





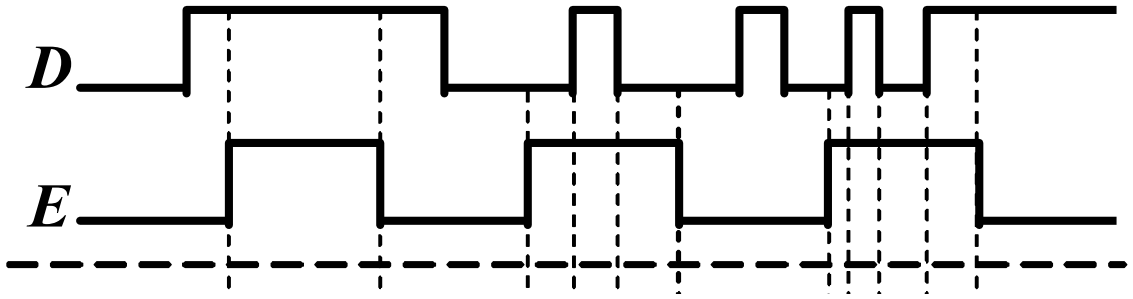
- *E*=1时 *Q* = *D*
- *E*=0时 *Q*不变

D锁存器的功能表

E	D	Q	$\overline{Q}$	功能
0	×	不变	不变	保持
1	0	0	1	置0
1	1	1	0	置1

E=1时 $Q = D$

E=0时 Q 不变

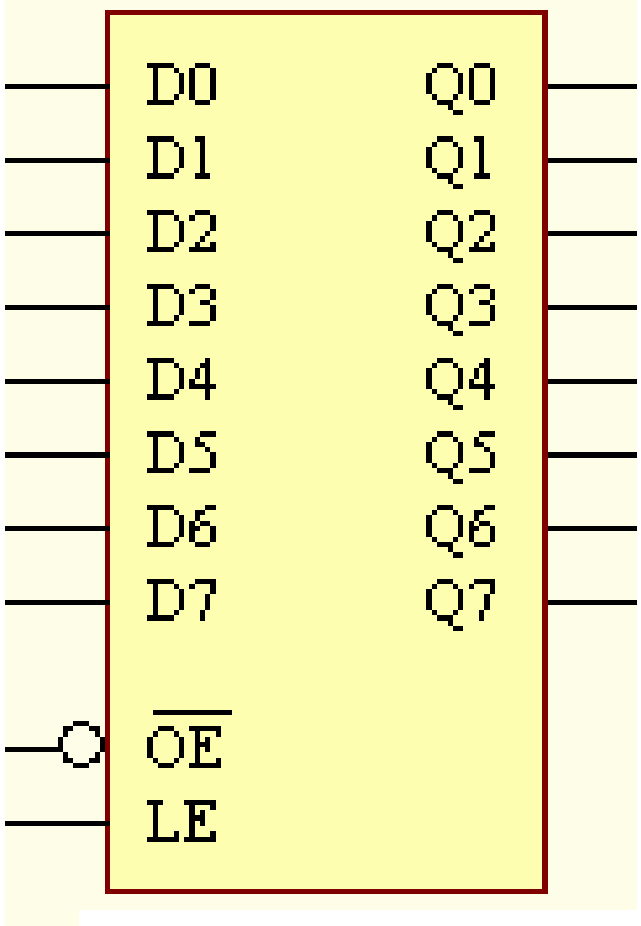


Q

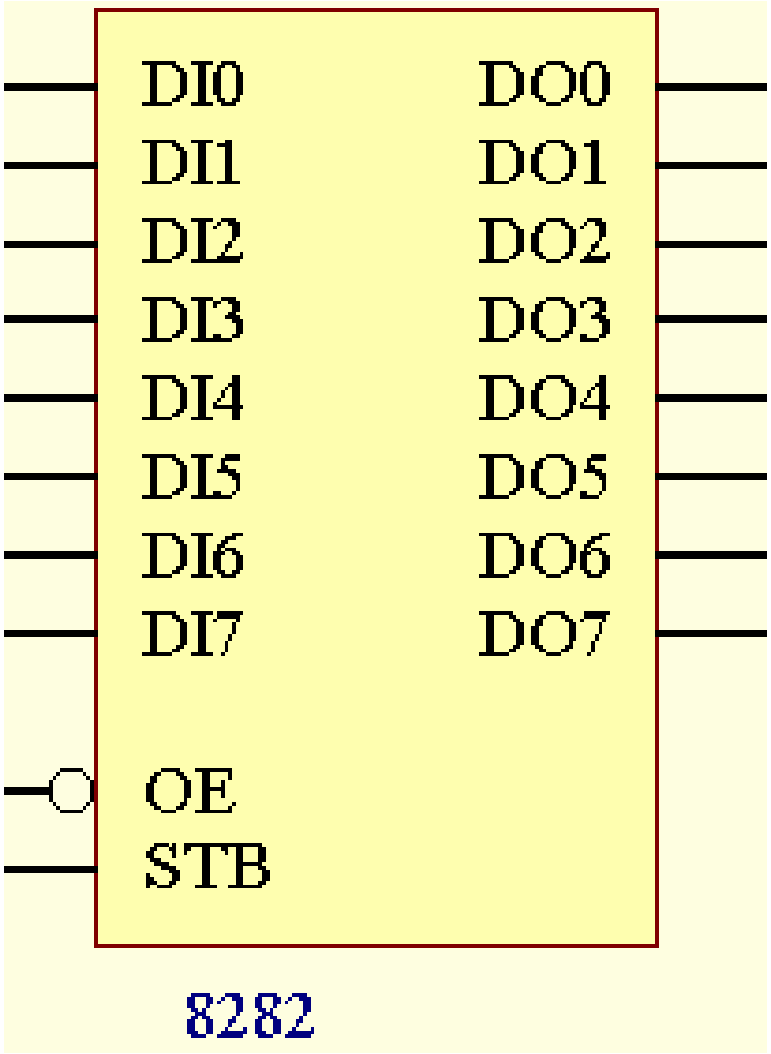
$\overline{Q}$

锁存器74HC373：8D三态锁存器

- *LE（Lock Enable)* *LE=1*：Q = D；*LE=0*：Q不变
- $\overline{OE}$（三态输出使能）：0输出使能，1输出失能



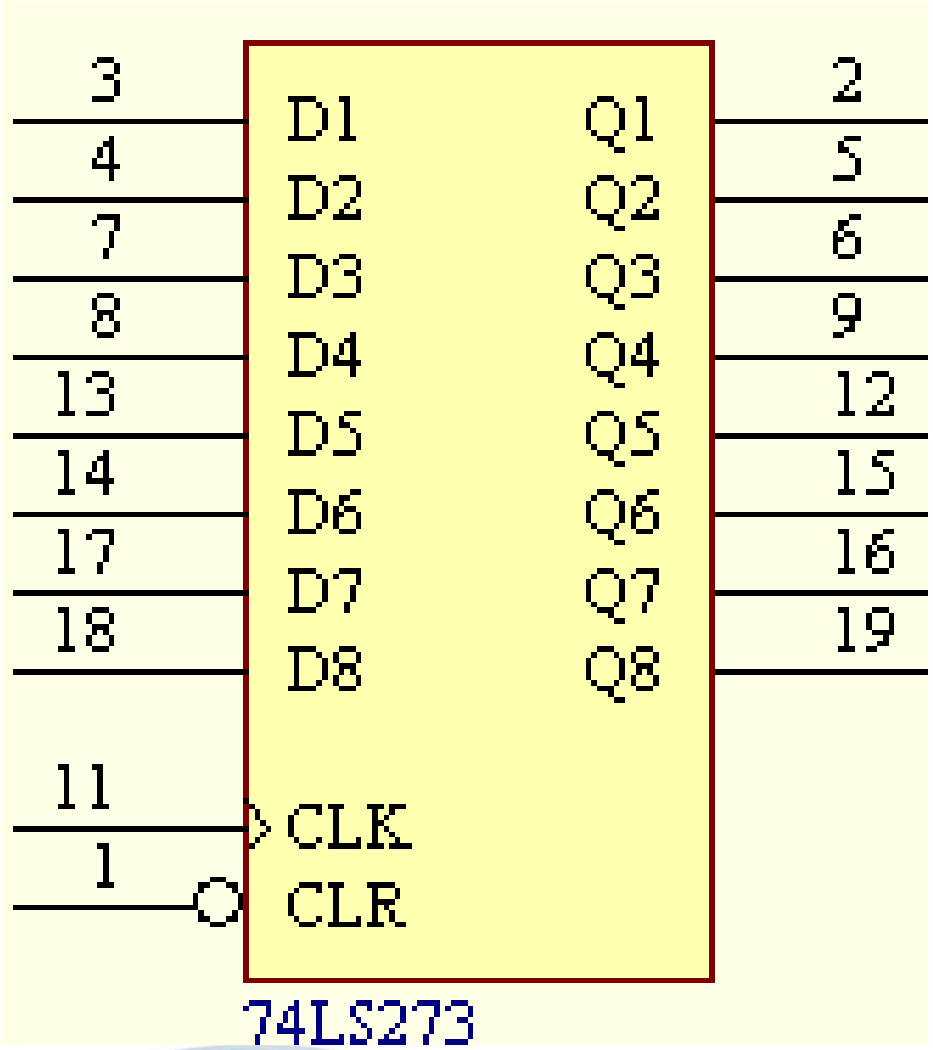
Intel 8282：三态锁存器



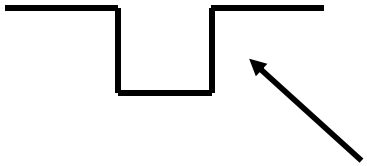
DI：输入；
DO：输出
STB：选通控制(等同373的LE)
OE ：输出使能

异步清零的锁存器：74LS273

D：输入 ； Q：输出　　CLK：控制　CLR: Clear

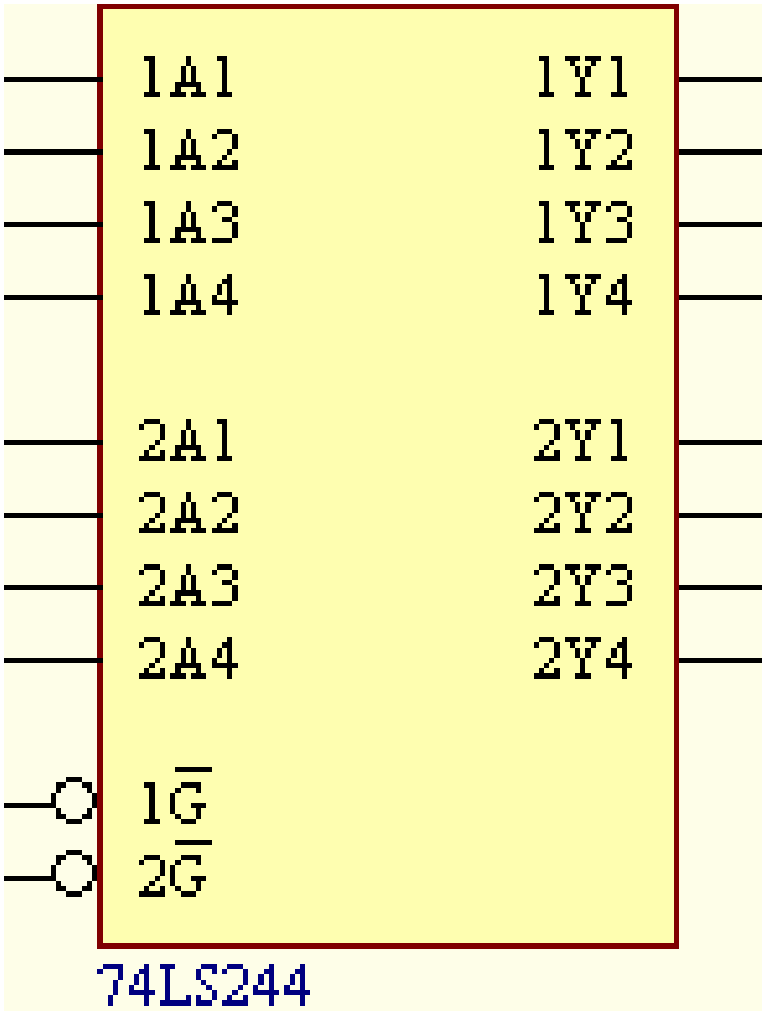


具有异步清零的
上升沿锁存器



负脉冲的上升沿

74LS244 ——缓冲器(实质是三态开关)



1$\overline{G}$ = 0: 1Y = 1A
1$\overline{G}$ = 1: 1Y = Z

2$\overline{G}$ = 0: 2Y = 2A
2$\overline{G}$ = 1: 2Y = Z

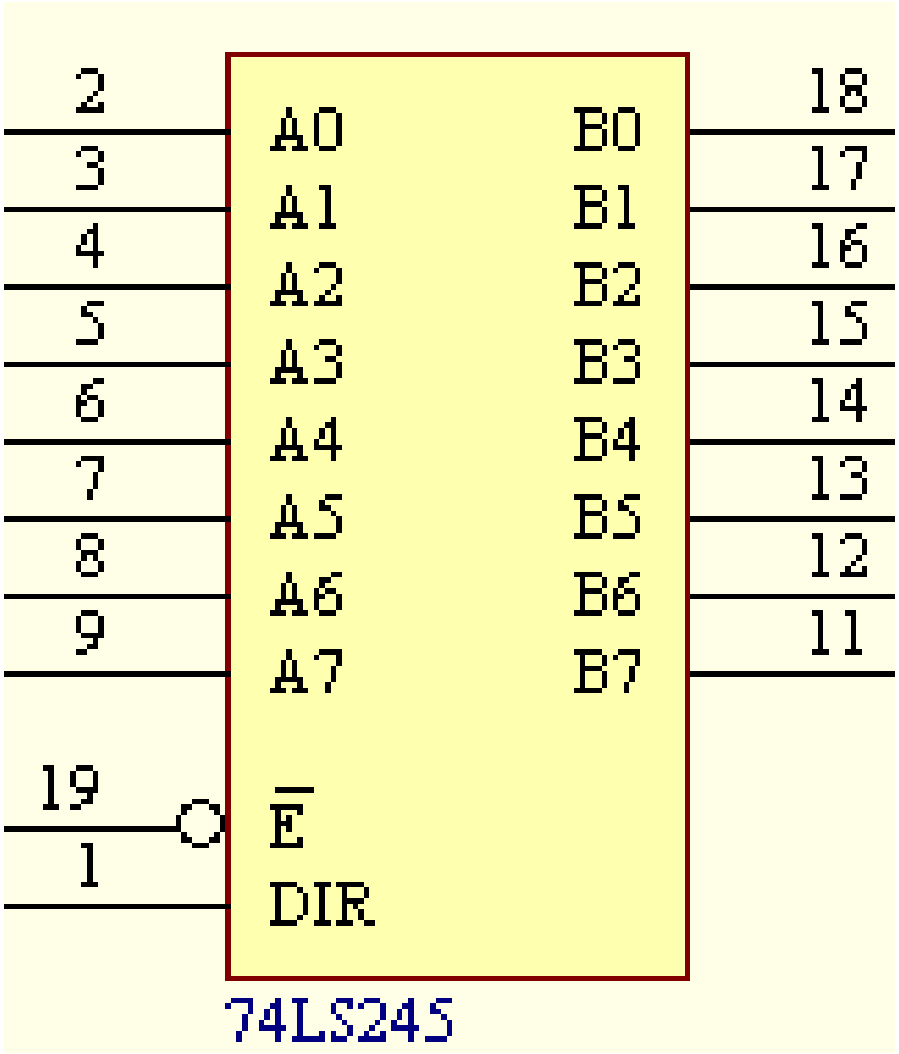
特点：

8位单向缓冲器

双**4**位分两组

输出与输入同相

74LS245 ——缓冲器(实质是三态开关)



8位双向缓冲器
控制端**E**低电平有效
输出与输入同相

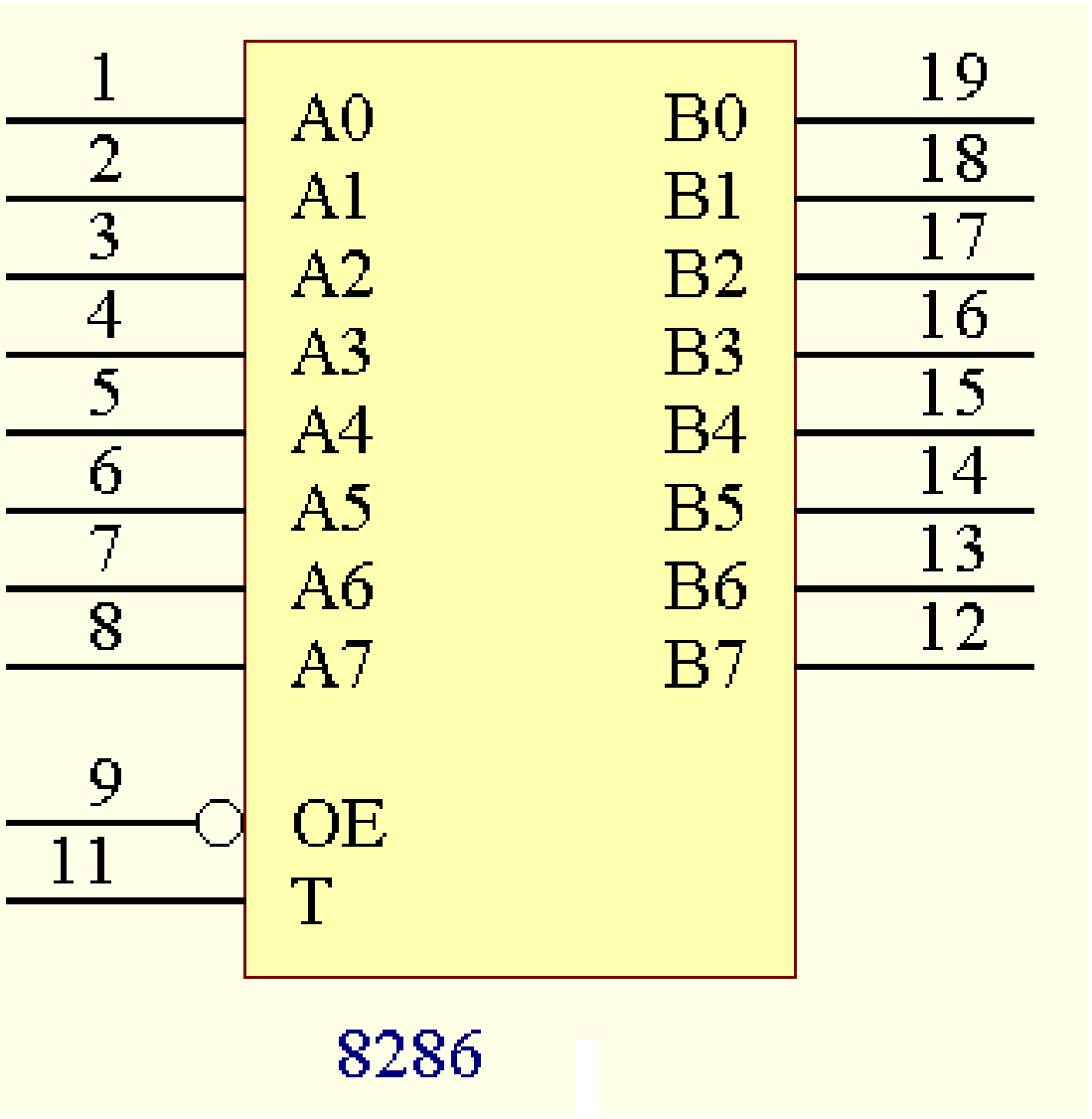
E：低电平导通

DIR：决定传输方向

DIR=1：A → B

DIR=0：B → A

缓冲器—— Intel 8286 (实质是三态开关)



8位双向缓冲器
控制端OE低电平有效
输出与输入同相

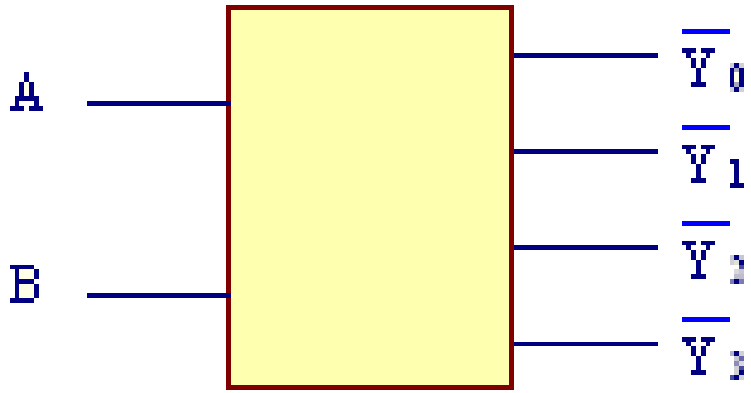
E：低电平导通
T：决定传输方向

DIR = 1：A → B
DIR = 0：B → A

数据收发器

基本芯片--译码器

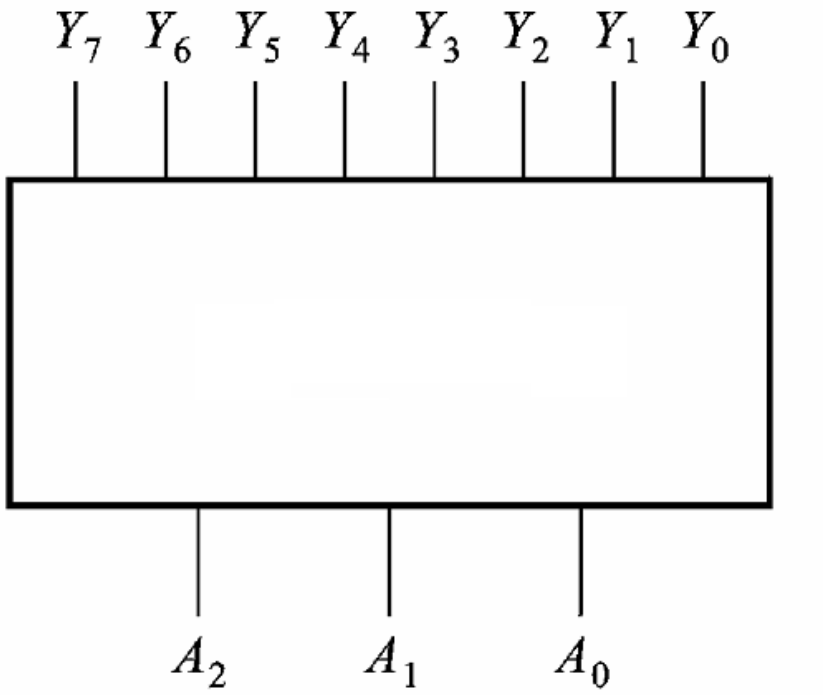
- 译码器
  - 功能：将某个二进制数据的含义“翻译”出来，指示唯一的某1个事件有效。
  - 结构：n位输入脚，2^n个输出脚（每脚对应1个事件）



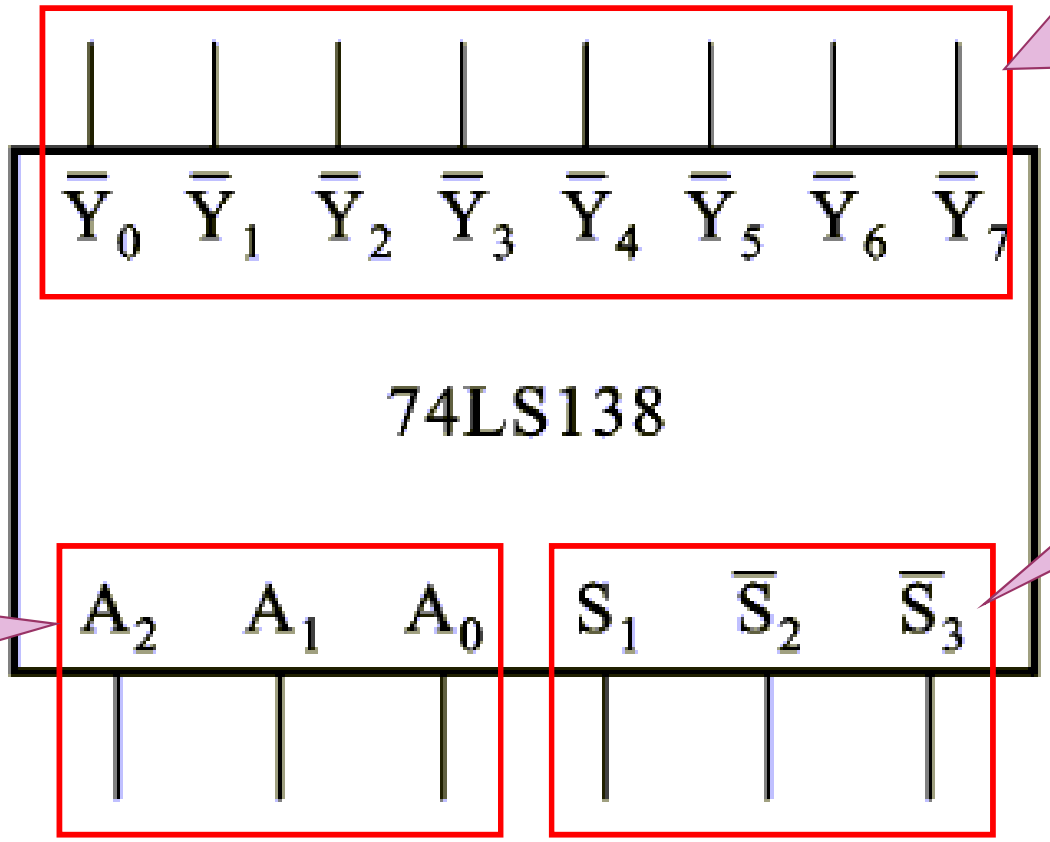
  - 特点：每次译码，仅唯一的1个输出引脚为有效电平
    - ◆输入：AB = [00, 01, 10, 11]
    - ◆输出：Y0Y1Y2Y3= 0111，1011，1101，1110

3-8译码器

- 结构：3个输入引脚，8个（ $=2^3$ ）输出引脚
- 功能：输入3位二进制代码$A_2A_1A_0$（000~111），在唯一的某个输出引脚上出现有效电平（低）。



74LS138 ——3-8译码器



低电平有效输出

3位二进制代码

使能端S

$$S = S_1 \cdot \overline{S_2} \cdot \overline{S_3}$$

- **S=1** 工作
- **S=0** 禁止（ 输出全**1**）

74LS138的逻辑符号

74LS138的功能表

输入			输出							
S_1	$\overline{S}_2+\overline{S}_3$	$A_2\ A_1\ A_0$	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$
×	1	× × ×	1	1	1	1	1	1	1	1
0	×	× × ×	1	1	1	1	1	1	1	1
1	0	0 0 0	0	1	1	1	1	1	1	1
1	0	0 0 1	1	0	1	1	1	1	1	1
1	0	0 1 0	1	1	0	1	1	1	1	1
1	0	0 1 1	1	1	1	0	1	1	1	1
1	0	1 0 0	1	1	1	1	0	1	1	1
1	0	1 0 1	1	1	1	1	1	0	1	1
1	0	1 1 0	1	1	1	1	1	1	0	1
1	0	1 1 1	1	1	1	1	1	1	1	0

禁止译码

译码工作

译中为0

高电平有效

低电平有效

常用集成门电路(TTL系列）

序号	型号	名称	主要功能
1	74LS00	四2输入与非门	
2	74LS02	四2输入或非门	
3	74LS04	六反相器	
4	74LS05	六反相器	OC门
5	74LS08	四2输入与门	
6	74LS10	三3输入与非门	
7	74LS14	六反相器	施密特触发
8	74LS20	双4输入与非门	
9	74LS21	双4输入与门	
10	74LS30	8输入与非门	
11	74LS32	四2输入或门	
12	74LS64	4-2-3-2输入与或非门	
13	74LS86	四2输入异或门	
14	74LS125	四总线缓冲器	三态输出

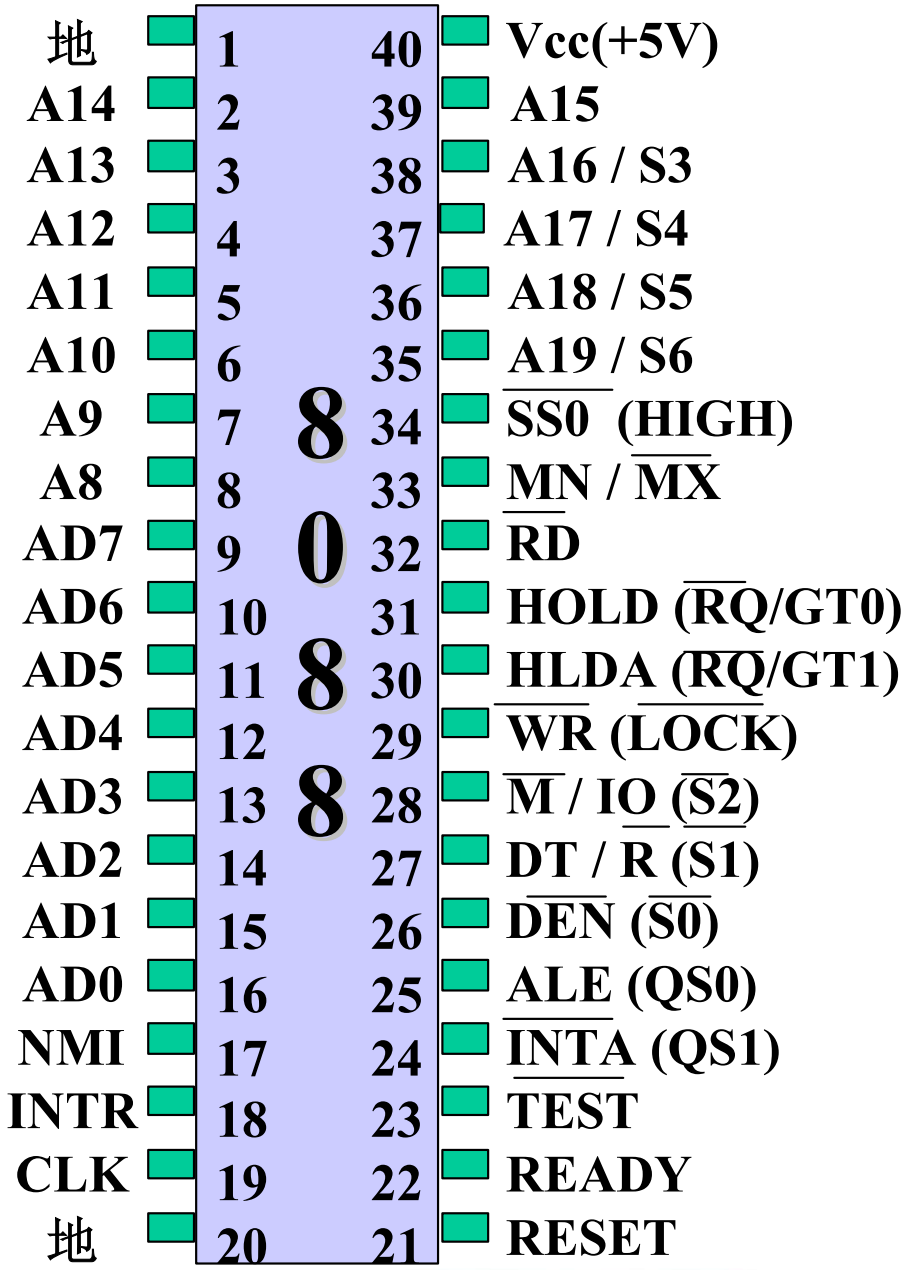
第4节 8088微处理的外部结构

8088的电气特性

- 电气特性
  - ■电源： 5V ± 10% 的条件下能够工作；
  - ■输入特性：
    低电平 0.8 V (0)
    高电平 2.0 V (1)
  - ■输出特性：
    低电平 0.45V (0)
    高电平 2.4 V (1)

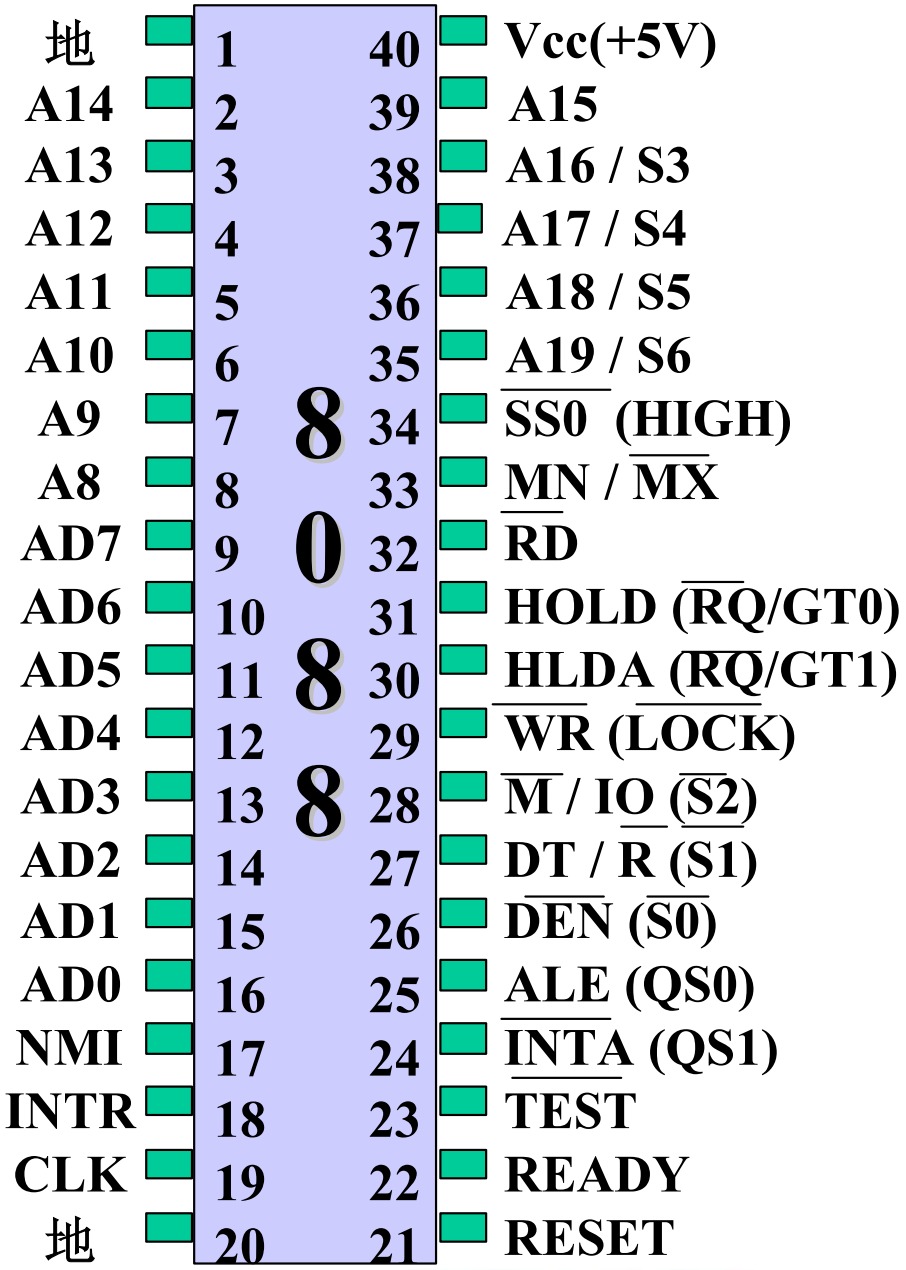
8088的外部引脚

- 外部引脚的属性
  - 引脚的功能
  - 是否复用
  - 信号的流向
  - 有效电平
  - 三态能力



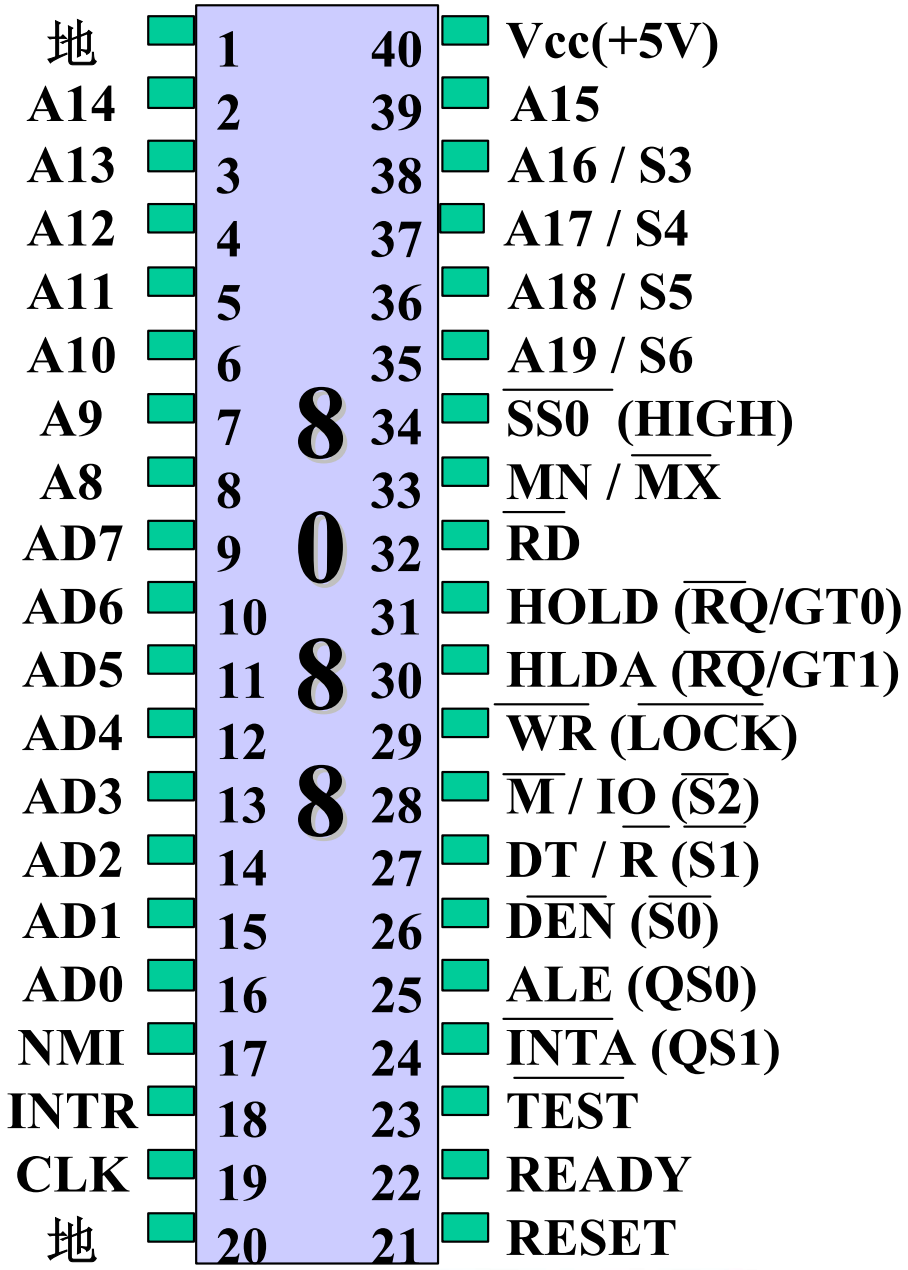
8088的外部引脚——辅助引脚：电源，时钟，复位

- 电源管脚
  - VCC(40)、GND(1/20)
  - 5V±10%
- 时钟管脚
  - CLK(19)
  - 外接4.77MHz外部时钟
- 复位管脚
  - RESET(21)
  - 正脉冲复位(4T)
    - ◆标志清0：Flag=0000H；
    - ◆DS,SS,ES复位为0H;
    - ◆CS:FFFFH ,IP复位为0H；
      - FFFF0H放第一条指令。



8088的外部引脚——功能引脚

- 读、写信号$\overline{RD}$、$\overline{WR}$
- 端口/存储器选择信号$\overline{M}$/IO
- 数据传送/接收信号DT/$\overline{R}$
- 数据线AD7～AD0
- 数据允许$\overline{DEN}$
- 地址线A19～A0
- 地址锁存ALE
- TEST
- READY
- $\overline{INTR}$
- $\overline{INTA}$
- NMI



8088的外部引脚

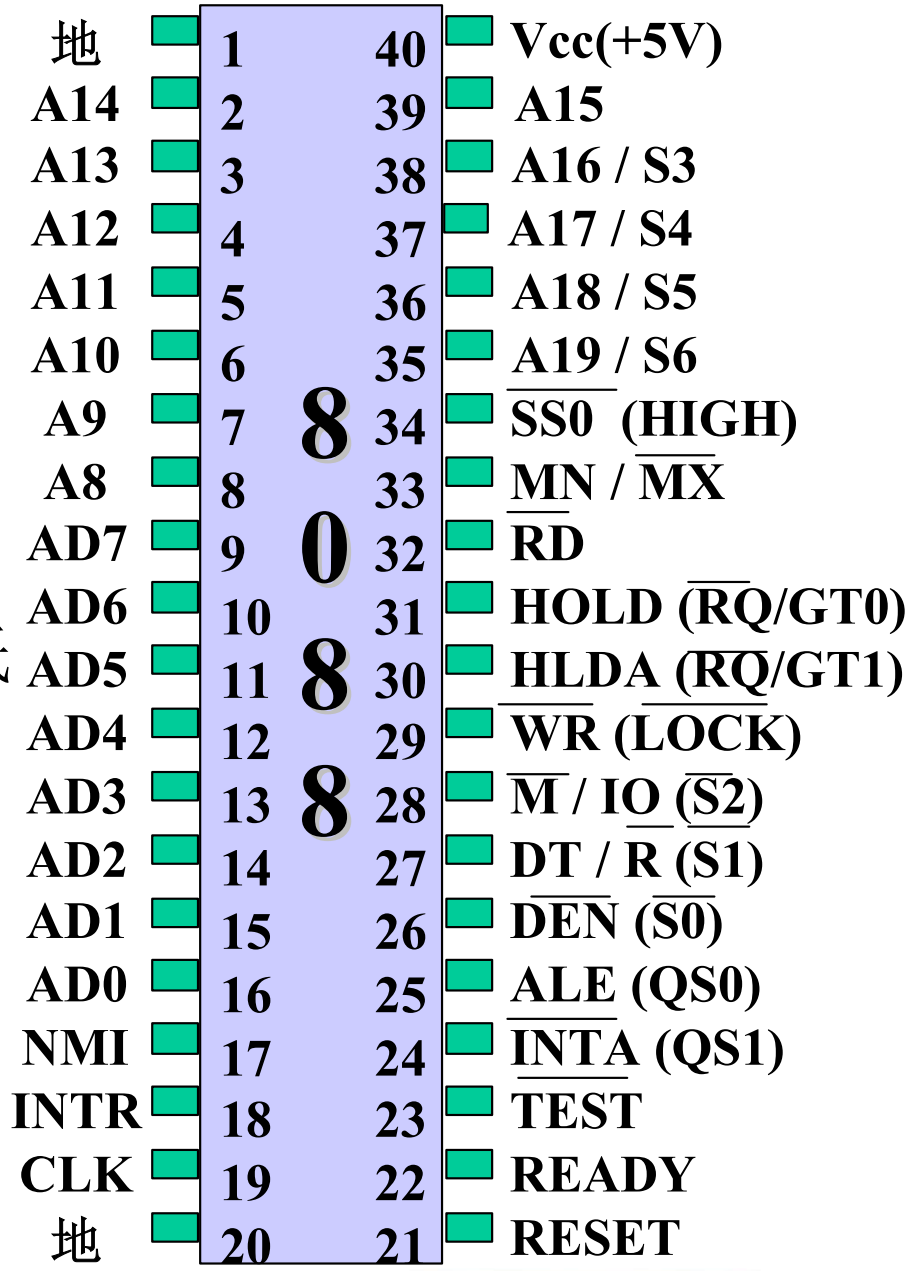
- $\overline{RD}$(32) ——读(输出，三态)
  - 读I/O端口或存储器时，$\overline{RD}$输出低电平；否则为高或高阻态。

```
IN      AL，25H      ；读I/O
MOV   AX，[BX]       ；读MEM
```

- $\overline{WR}$(29) ——写(输出，三态)
  - 写I/O端口或存储器时，$\overline{WR}$输出低电平；否则为高或高阻态。

```
OUT  25H，AX      ；写I/O
MOV [BX]，AX      ；写MEM
```

问题：执行：MOV　BX，AX ？



8088的外部引脚

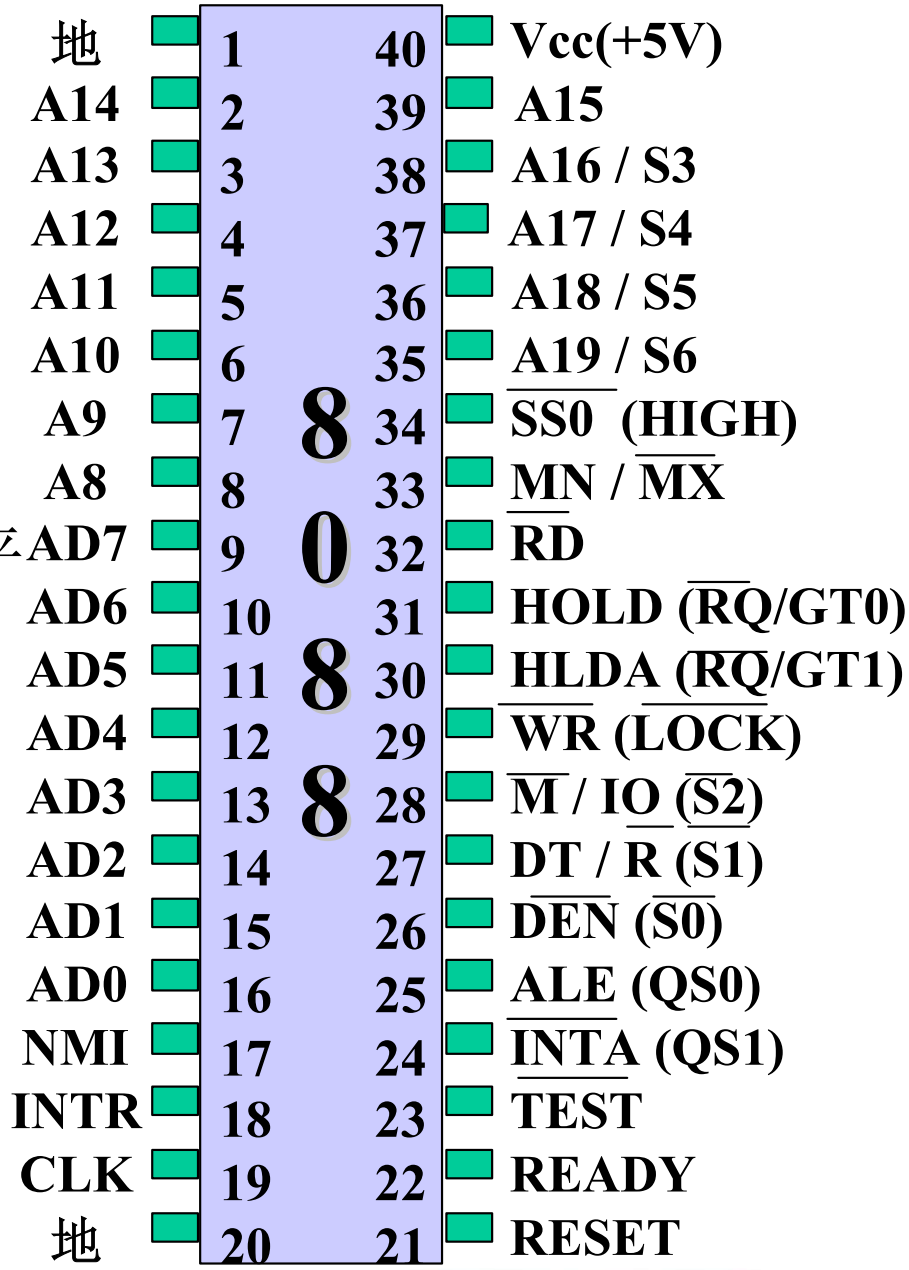
- IO/$\overline{M}$(28) ——端口/存储器选择(输出，三态)
  - 对端口读或写， IO/$\overline{M}$输出高电平

    IN　　AL，25H　；读I/O

    OUT　52H，AL　　；写I/O
  - 对存储器读或写，IO/$\overline{M}$输出低电平

    MOV　AX，[BX]　　；读MEM

    MOV　[BX]，AX　　；写MEM
  - 其它情况处于高阻状态



读写信号和IO/$\overline{M}$信号的组合

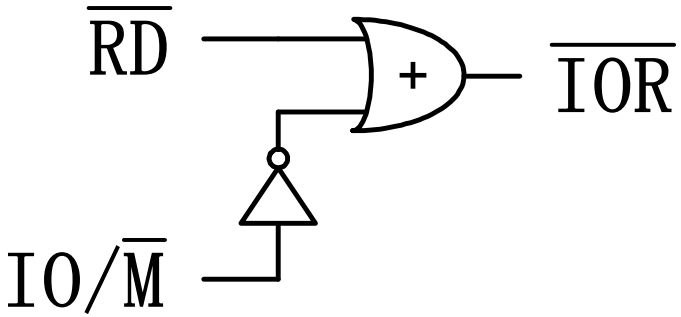
- 由IO/$\overline{M}$、$\overline{WR}$和$\overline{RD}$生成存储器读、存储器读写、I/O读、I/O写等4种信号（低电平有效）

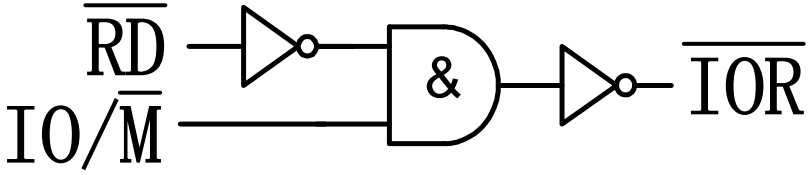
输出/低电平有效	IO/$\overline{M}$	$\overline{WR}$	$\overline{RD}$
存储器读(MEMR)	低	/	低
存储器写(MEMW)	低	低	/
I/O读(IOR)	高	/	低
I/O写(IOW)	高	低	/

读写信号和IO/$\overline{M}$信号的组合——实现IO读信号

- 通过RD、WR、IO/M得到IO读信号(IOR)

	IO/M	WR	RD
MEMR	低	/	低
MEMW	低	低	/
IOR	高	/	低
IOW	高	低	/

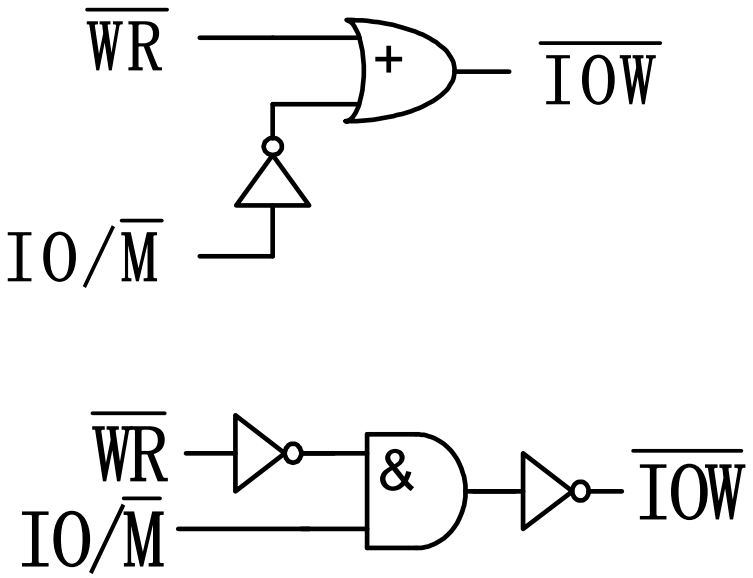




读写信号和IO/$\overline{M}$信号的组合——实现IO写信号

- 通过RD、WR、IO/M得到IO写信号(IOW)

	IO/M	WR	RD
MEMR	低	/	低
MEMW	低	低	/
IOR	高	/	低
IOW	高	低	/

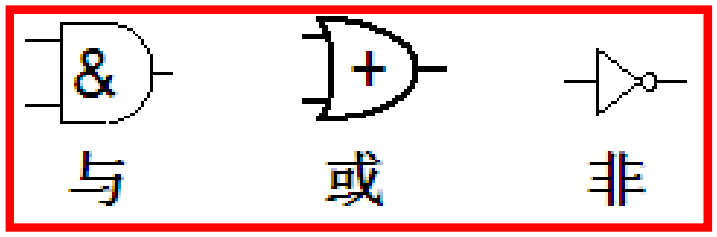


课堂练习——实现存储器读/写信号

- 通过RD、WR、IO/M得到存储器读/写信号(MEMR/MEMW)

	IO/M	WR	RD
MEMR	低	/	低
MEMW	低	低	/
IOR	高	/	低
IOW	高	低	/

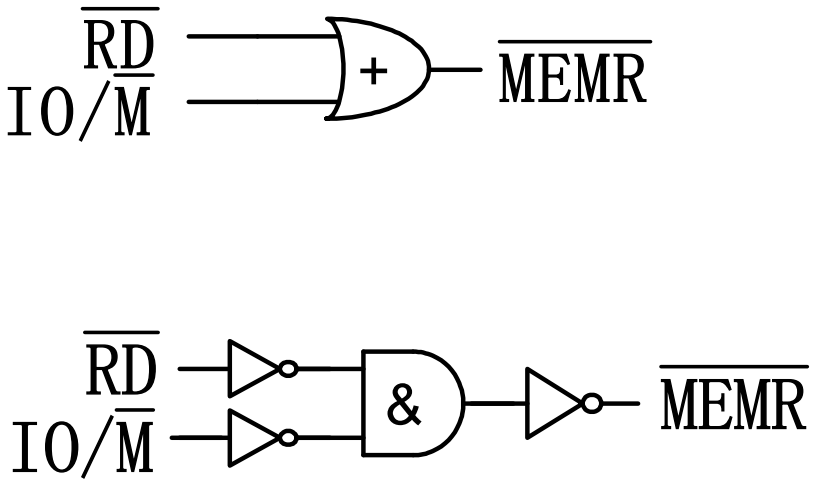
- 画出它们各自或和与两种形式的电路图



读写信号和IO/$\overline{M}$信号的组合——实现存储器读信号

- 通过RD、WR、IO/M得到存储器读信号

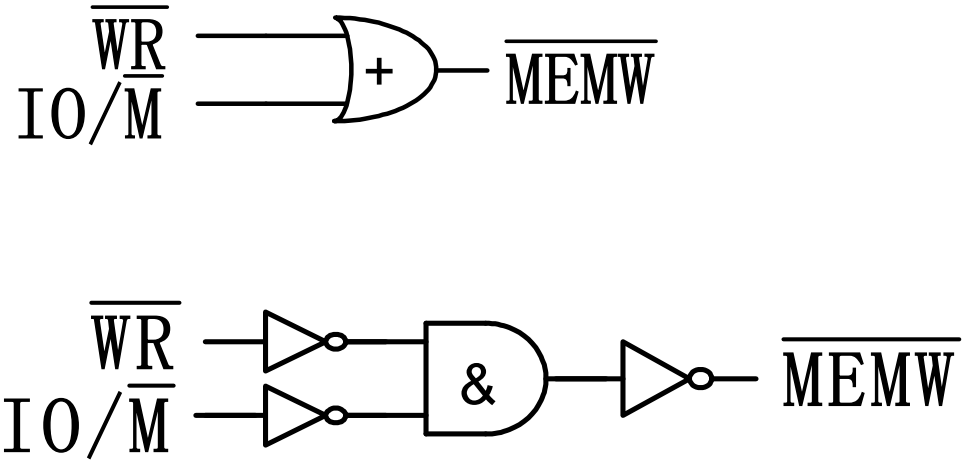
	IO/M	WR	RD
MEMR	低	/	低
MEMW	低	低	/
IOR	高	/	低
IOW	高	低	/



读写信号和IO/$\overline{M}$信号的组合——实现存储器写信号

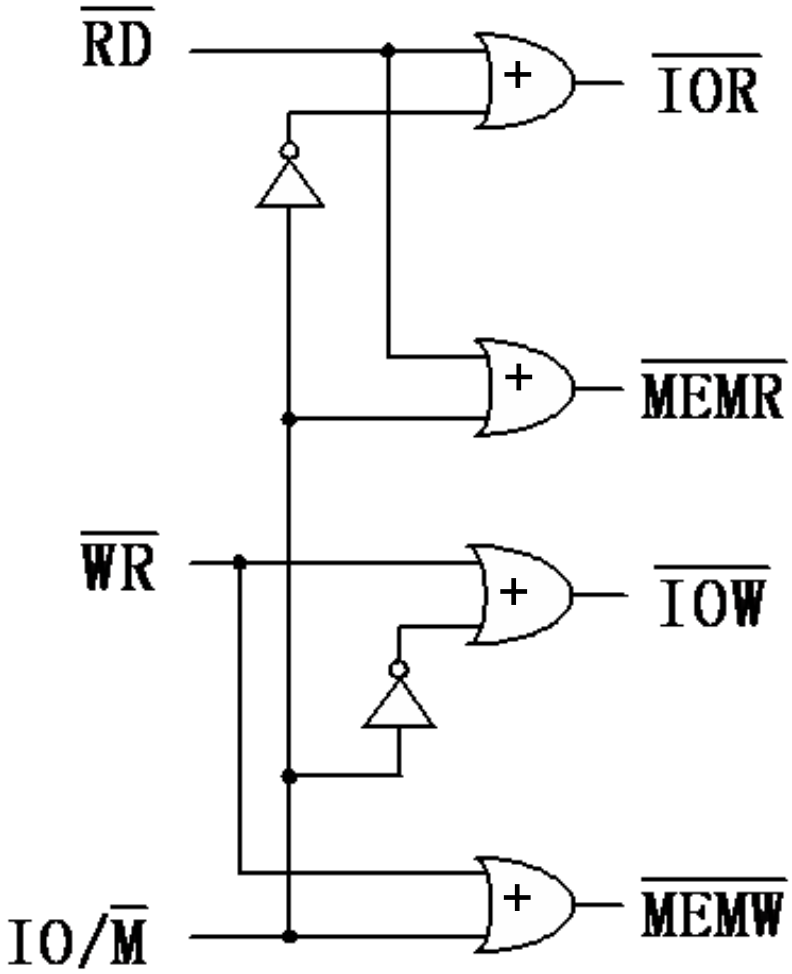
- 通过RD、WR、IO/M得到信号存储器写信号

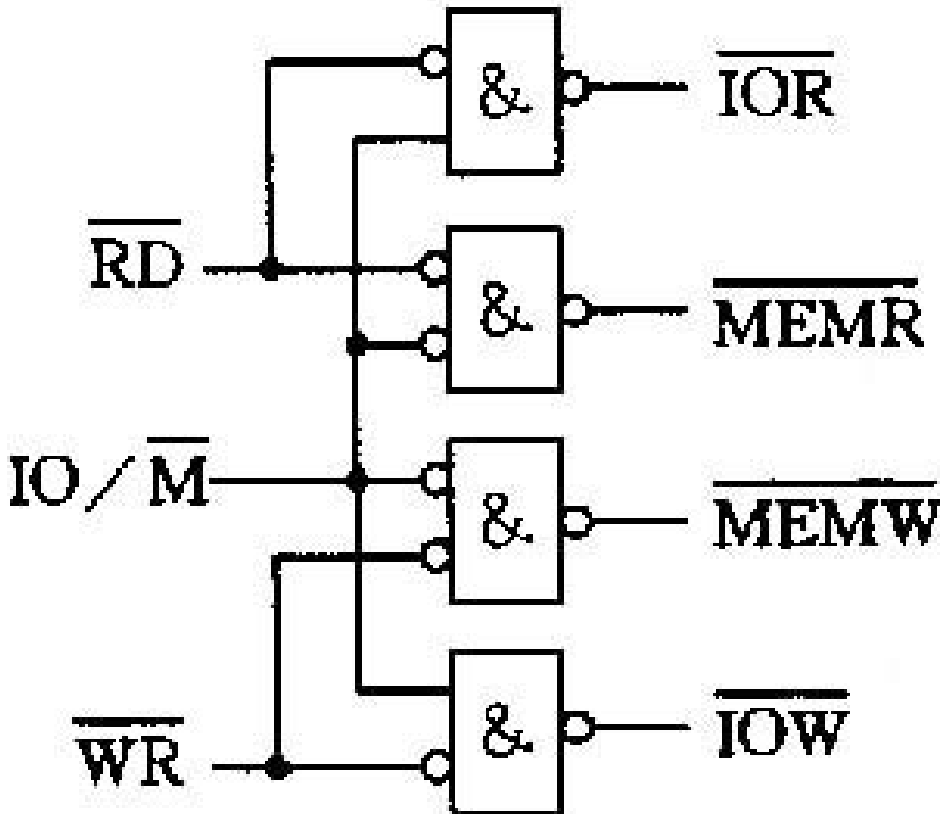
	IO/M	WR	RD
MEMR	低	/	低
MEMW	低	低	/
IOR	高	/	低
IOW	高	低	/



读写信号和IO/$\overline{M}$信号的组合——综合实现存储器和IO读写信号

- 通过单一电路同时实现存储器和I/O的读写4个信号





8088的外部引脚

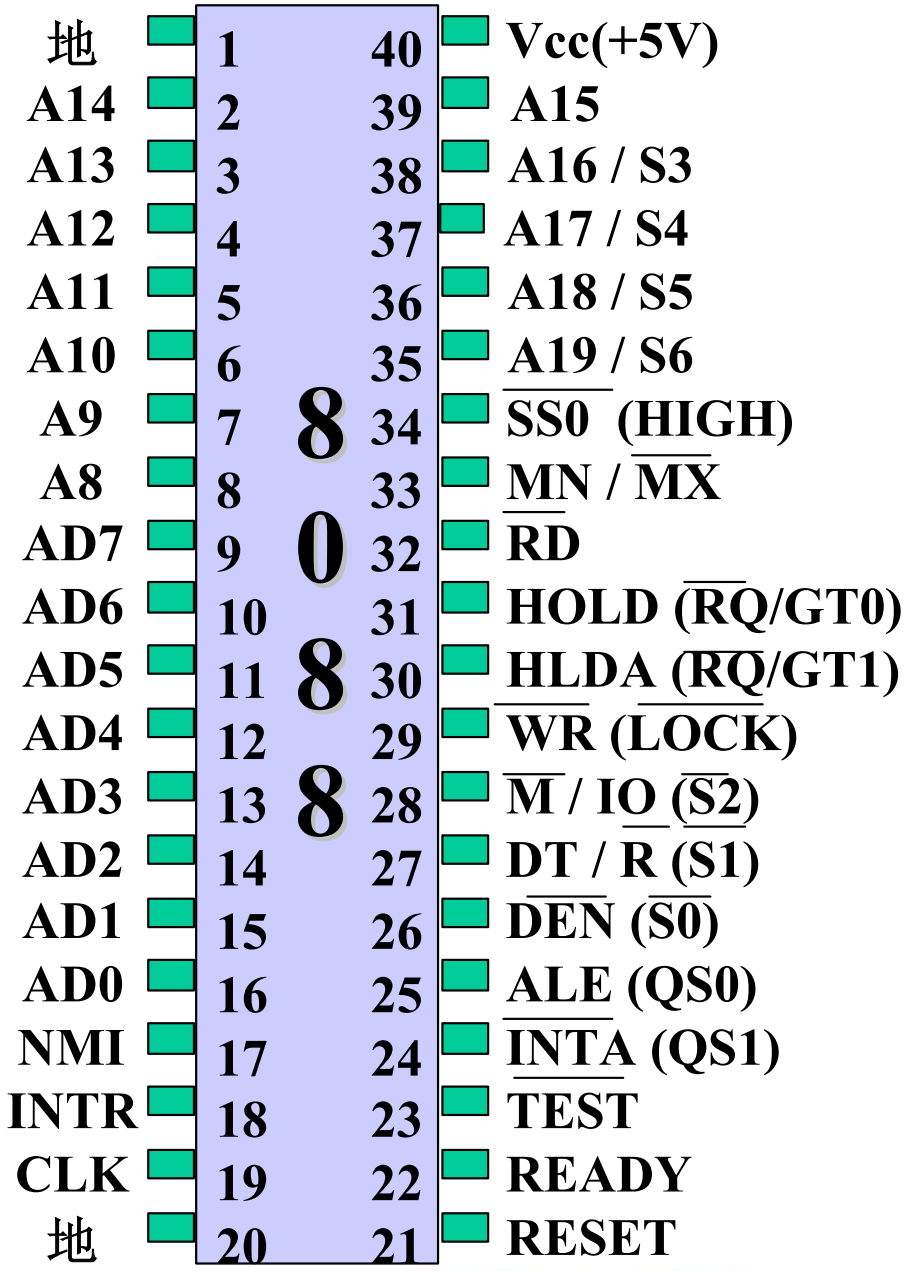
- DT/$\overline{R}$(27)
  - 数据传送/接收信号 (出，三态)
  - 向I/O或存储器传送数据，输出高

    OUT 25H，AL ；写I/O

    MOV [BX]，AX ；写MEM
  - 从I/O或存储器接收数据，输出低

    IN AL，25H； 读I/O

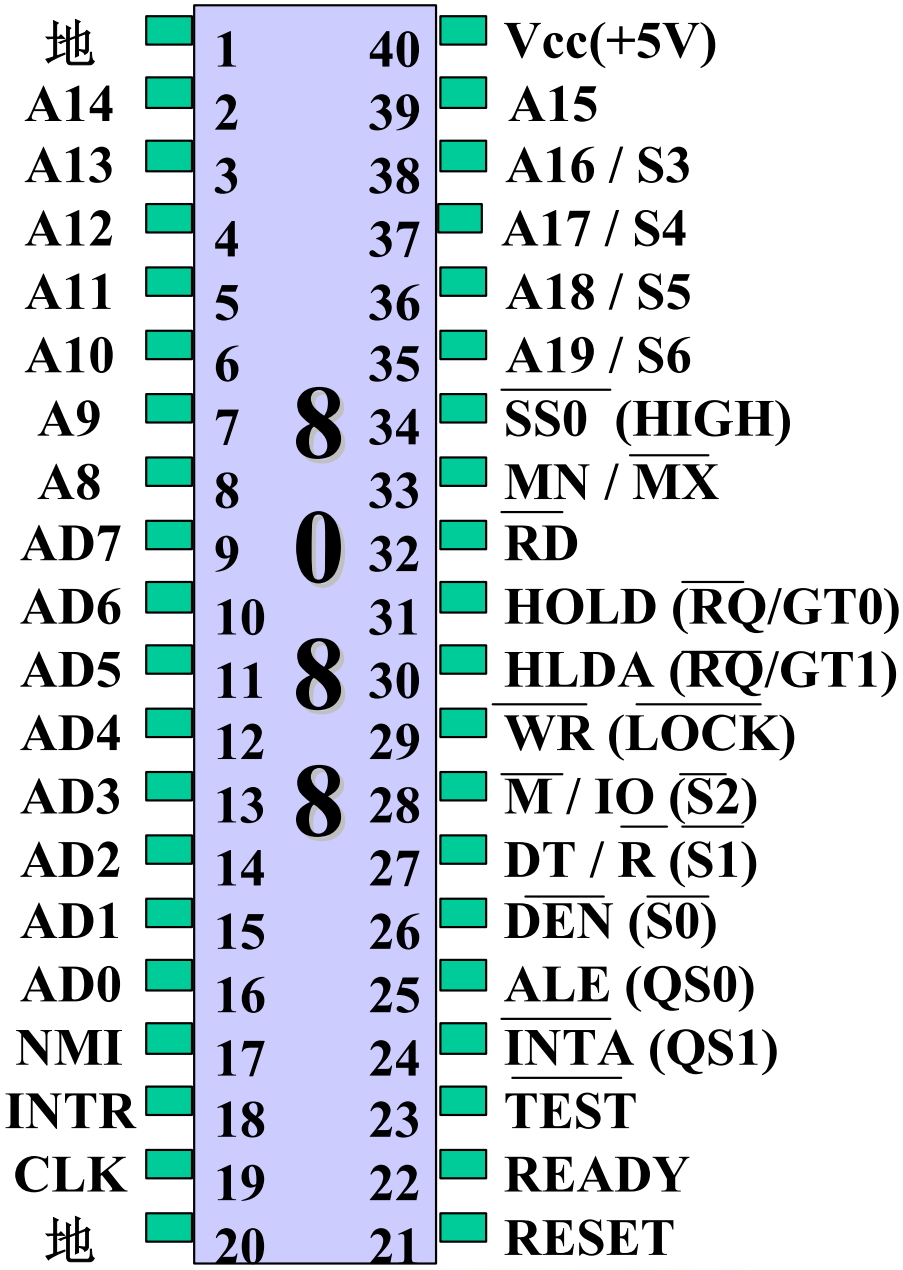
    MOV AX，[BX]； 读MEM
  - 其它情况下处于高阻状态



8088的外部引脚

- AD7～AD0(9～16)数据线(双，三态)
  - 读写I/O或存储器时出现相应数据

```
IN    AL，25H          ；读I/O
OUT 25H，AL            ；写I/O
MOV  BX，0000H
MOV  AL，[BX]          ；读MEM
MOV  [BX]，  32H       ；写MEM
```

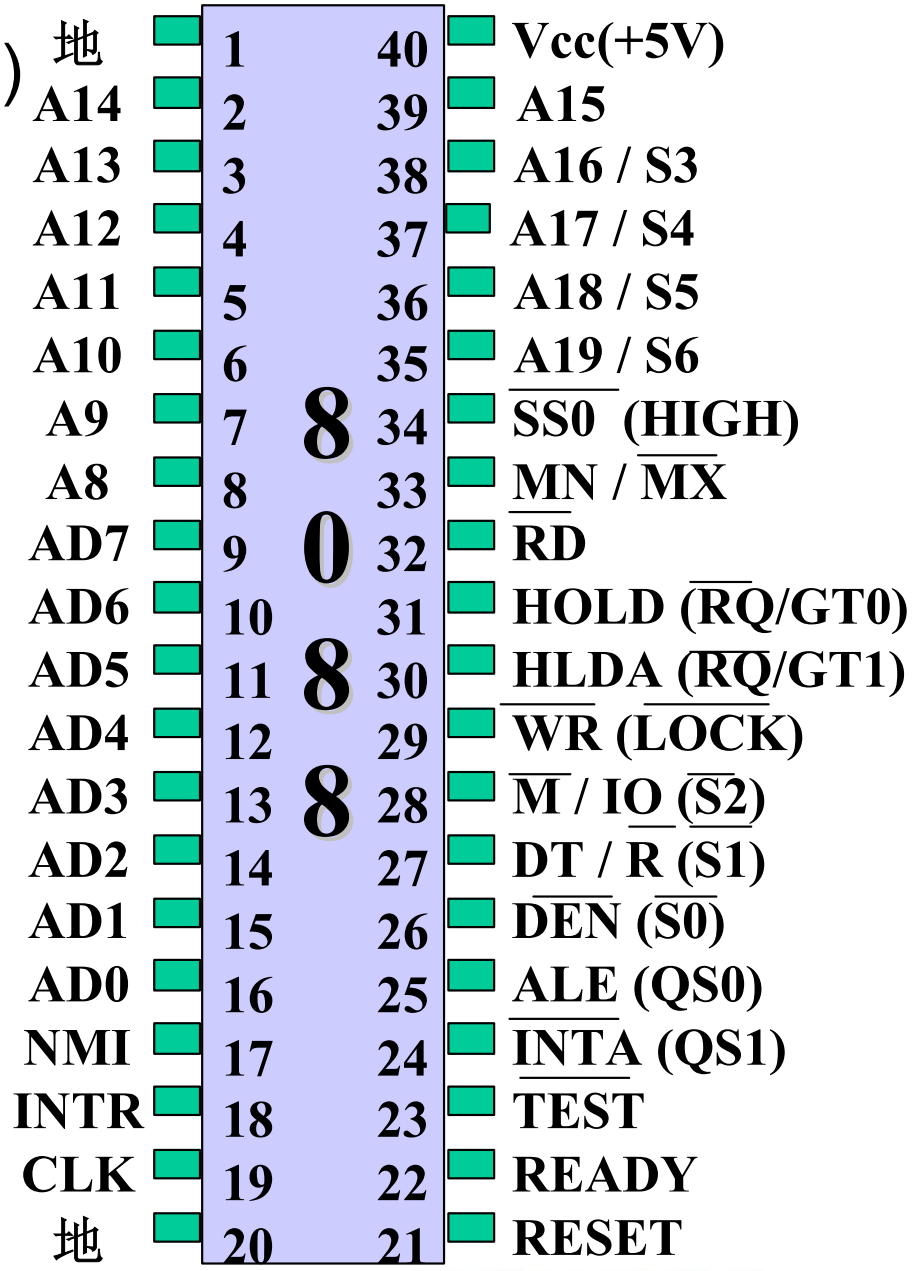


8088的外部引脚

- A19～A8,AD7~AD0地址线(出，三态)
  - 读写I/O或存储器，出现地址信息

```
IN    AL，25H      ；读I/O
OUT   25H，AL      ；写I/O
MOV   BX，0000H
MOV   AL，[BX]  ；读MEM
MOV  [BX]，32H  ；写MEM
```

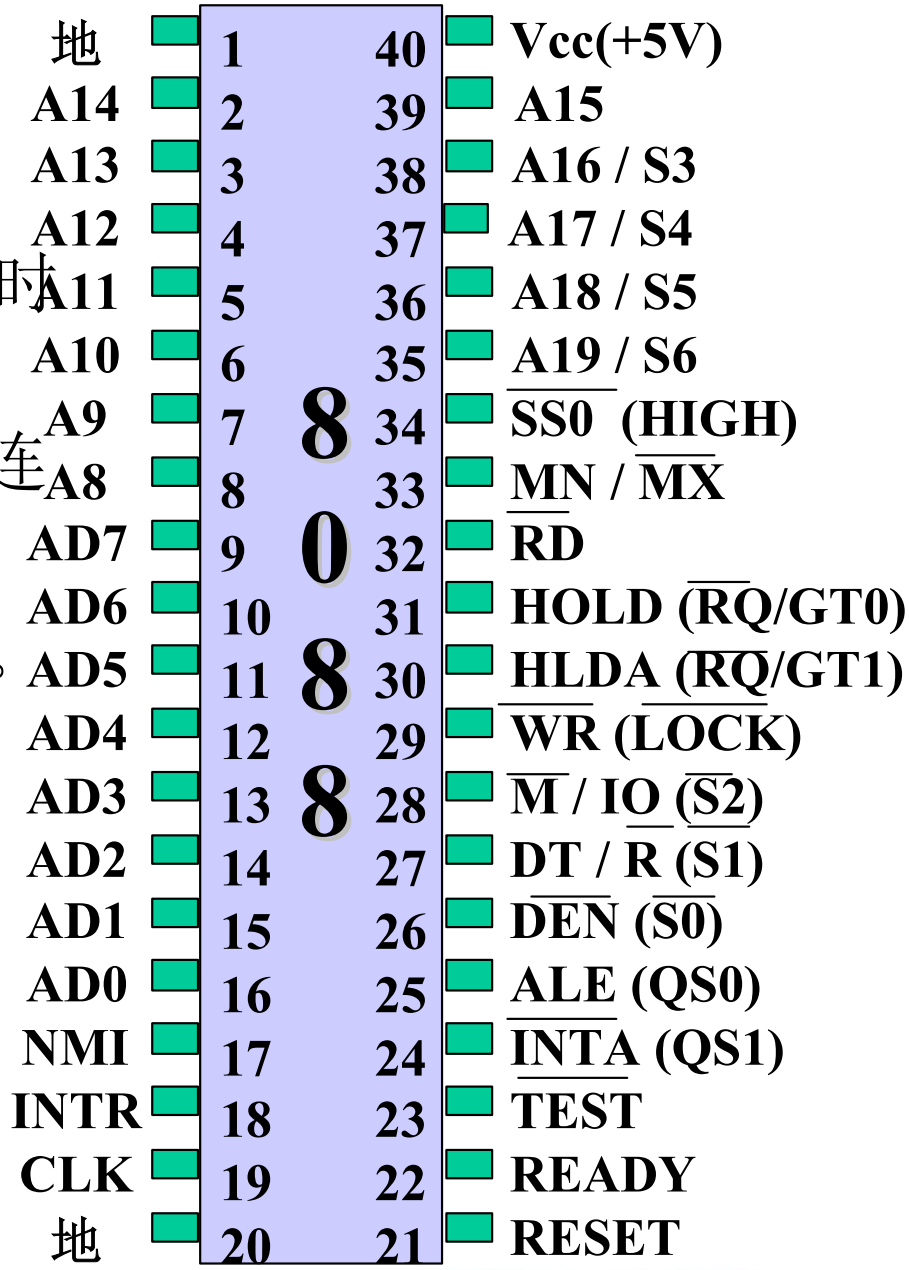
  - AD7～AD0数据和地址复用
    - 先传送地址，后传送数据。
    - 地址20位
    - 数据8位
- ALE(25)地址锁存信号(出，三态)
  - 当$A_{19\sim0}$出现地址，产生正脉冲。
  - 利用该脉冲锁存地址



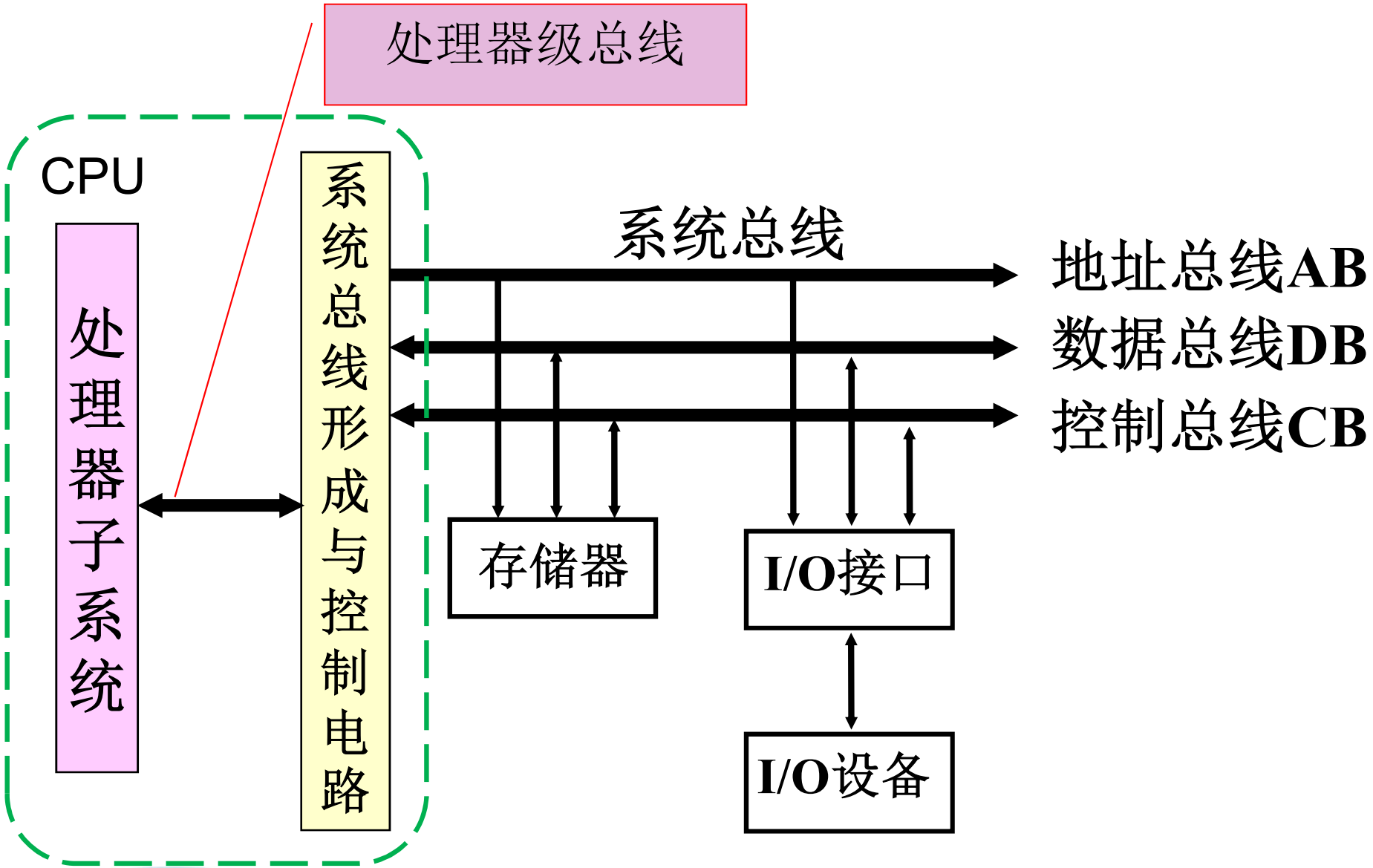
8088的外部引脚

- $\overline{DEN}$(26)数据允许 (出，三态)
  - 低电平：$AD_{7\sim0}$传送的是数据。
- A19/S6-A16/S3仅在T1输出地址,其他时间输出状态S_6-S_3
  - S6：恒低,表示8088当前与总线相连
  - S5：输出，PSW中断允许标志状态
  - S4,S3：当前正在使用的段寄存器。

S_4	S_3	
0	0	ES
0	1	SS
1	0	CS
1	1	DS



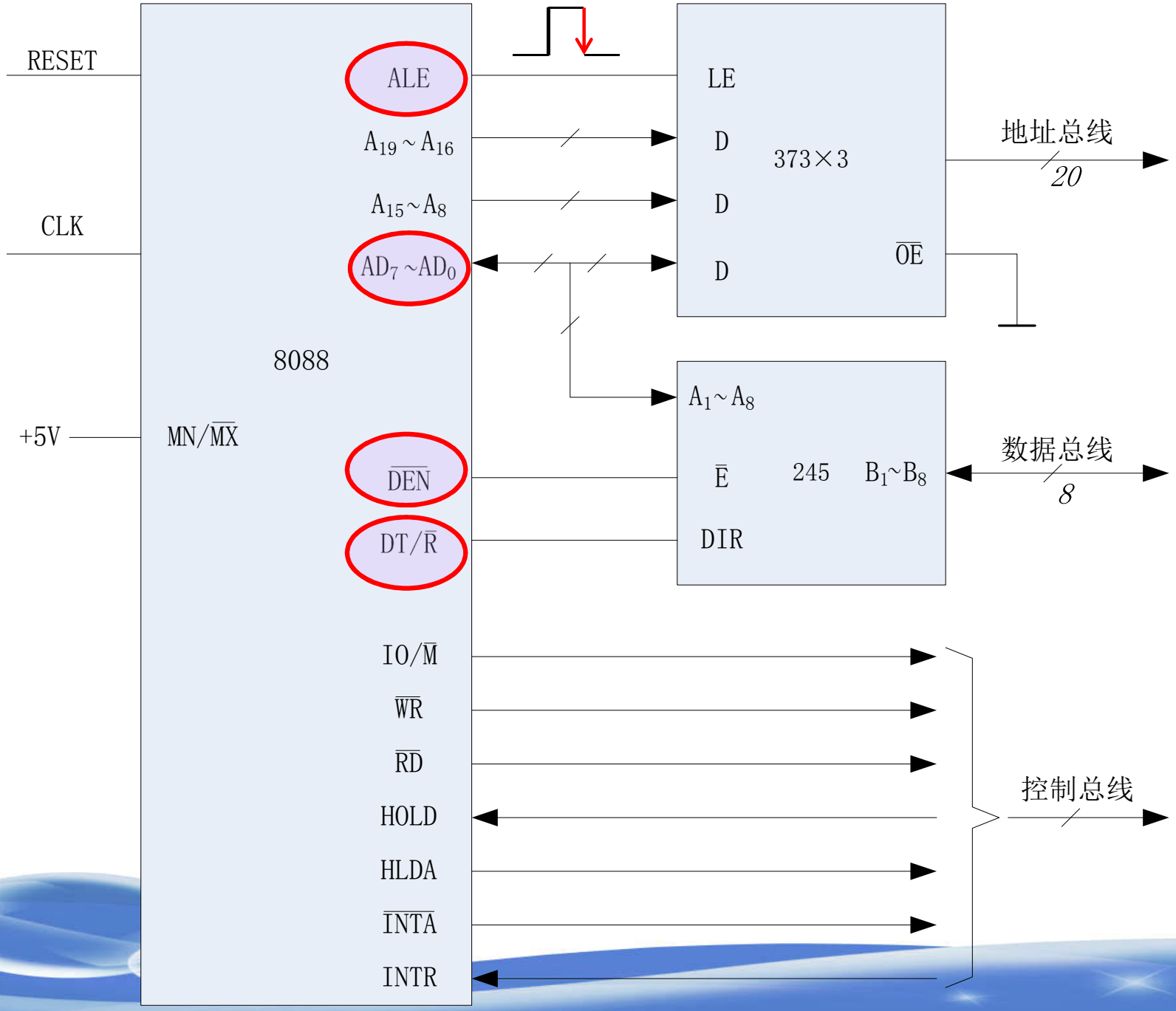
8088系统总线的形成



8088系统总线的形成

- 主要解决
  - 实现地址总线，数据总线和控制总线
    - 地址与数据的分离
    - 地址锁存

8088总线生成的典型实现电路

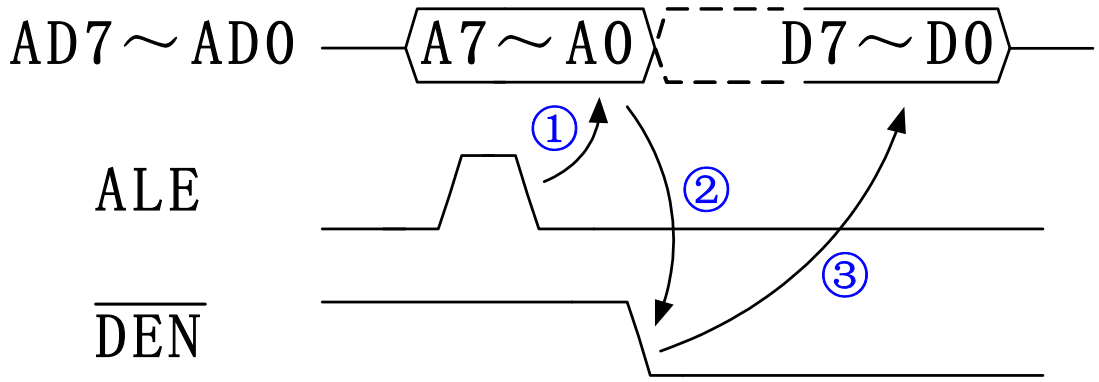


8088总线生成的典型实现电路

- 电路实现方案
  - 用3片8位锁存器（例74LS373）实现地址锁存。ALE为锁存控制信号，$\overline{OE}$=0输出地址；
  - 用1片双向缓冲器（三态门,例如74LS245）用作数据总线隔离，DT/$\overline{R}$控制方向，$\overline{DEN}$作为使能信号；
  - 控制信号由8088直接产生。

8088总线生成的典型实现电路

- $AD_{7\sim0}$解复用和地址锁存
  - ■先传送地址，后传送数据



①总线出现地址后，在ALE下降沿，地址被373锁存；

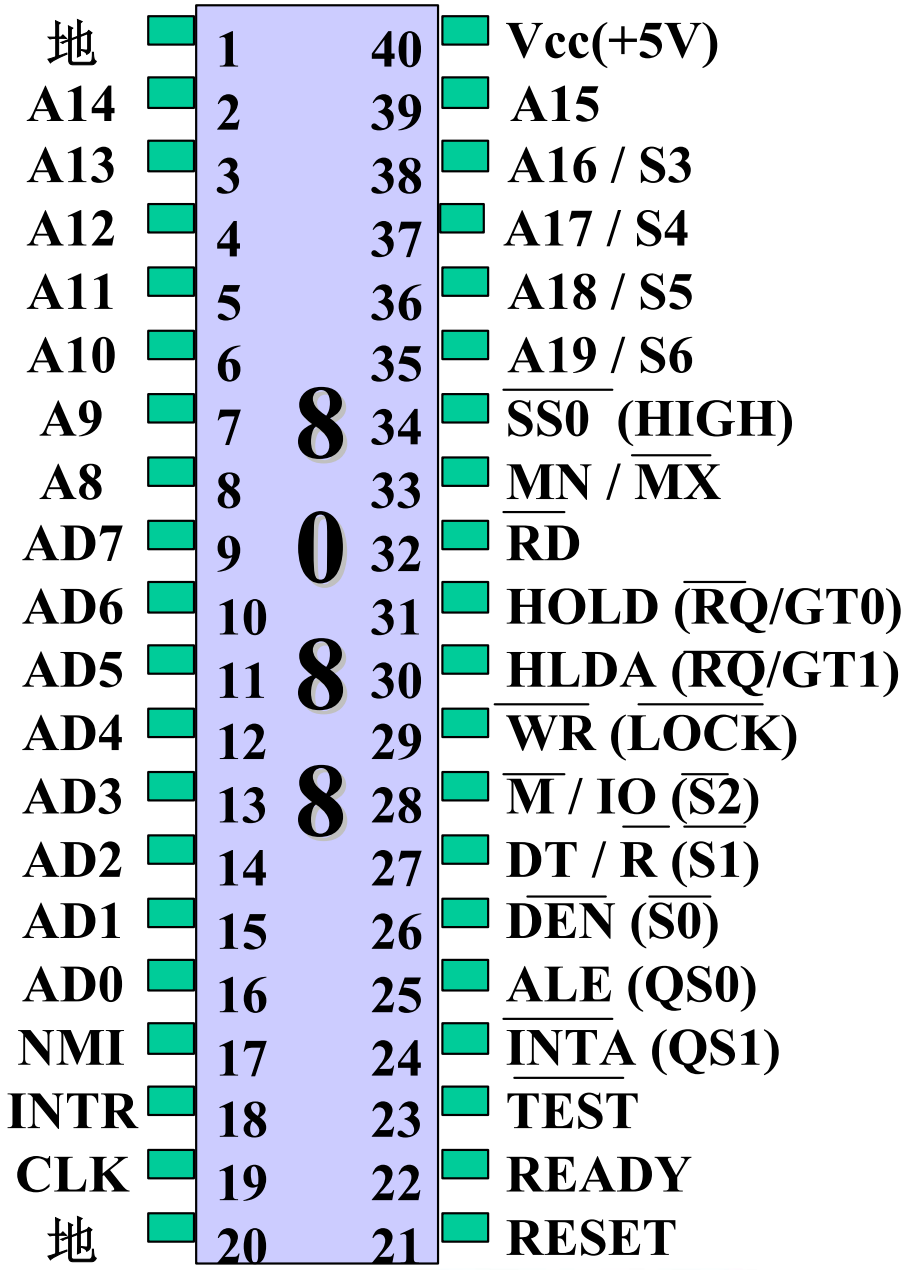
②总线出现地址时，$\overline{DEN}$高电平，245失能，数据总线被禁止。

③总线出现数据时，$\overline{DEN}$变低，245使能，数据总线开通。

- 数据传送方向的确定
  - ■数据输出时DT/$\overline{R}$高电平，245的DIR高电平：A ⟶ B
  - ■数据输出时DT/$\overline{R}$高电平，245的DIR高电平：B ⟵ A

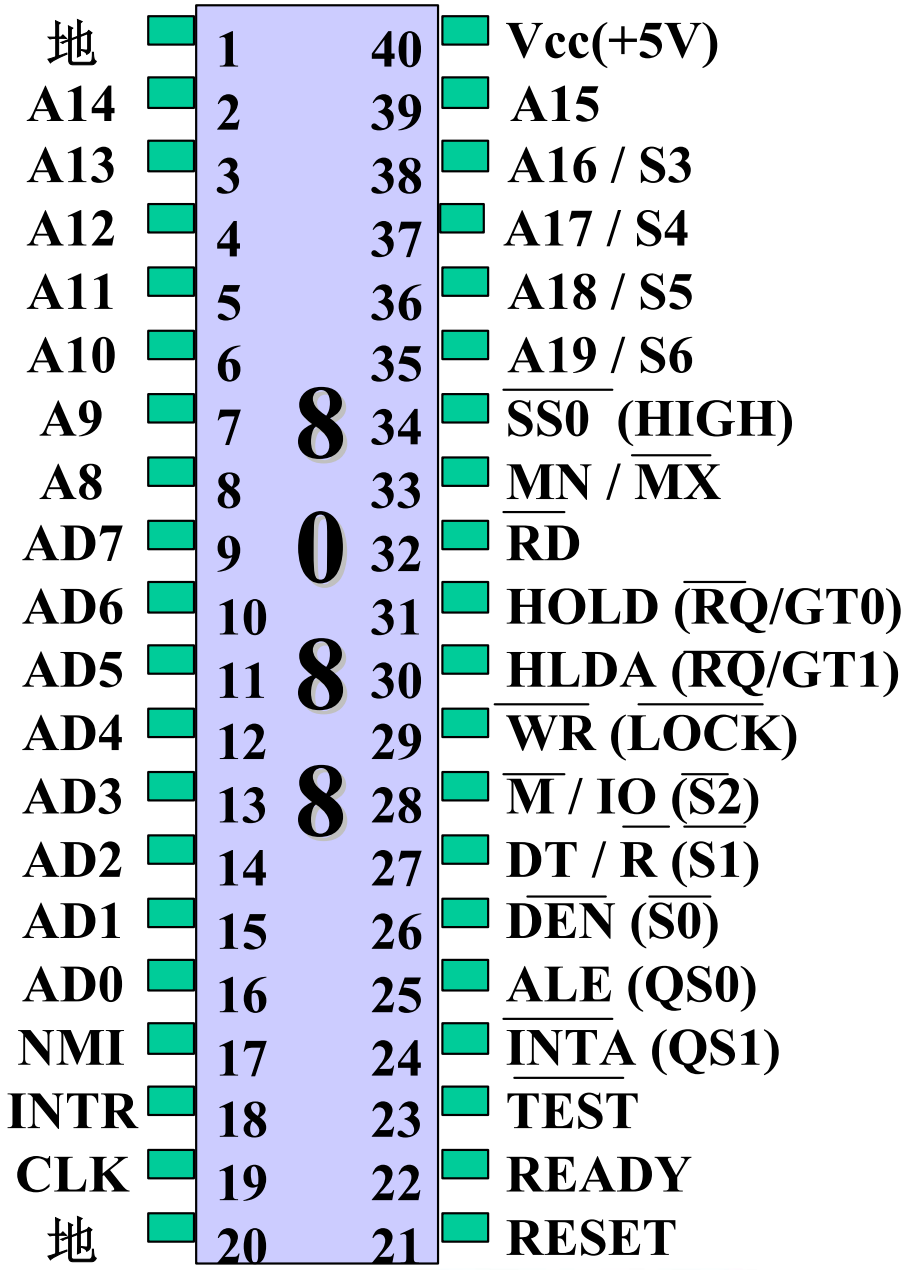
8088的外部引脚

- INTR(18, 中断请求信号, 入)
  - 高电平有效
  - 外设发来的可屏蔽中断请求信号。
- $\overline{\text{INTA}}$(24, 中断请应答信号, 出)
  - 低电平有效
  - 向外设回应的中断应答信号。
  - 连续2个负脉冲
- NMI(17, 非屏蔽中断请求信号, 入)
  - 非屏蔽中断请求信号
  - 高电平有效



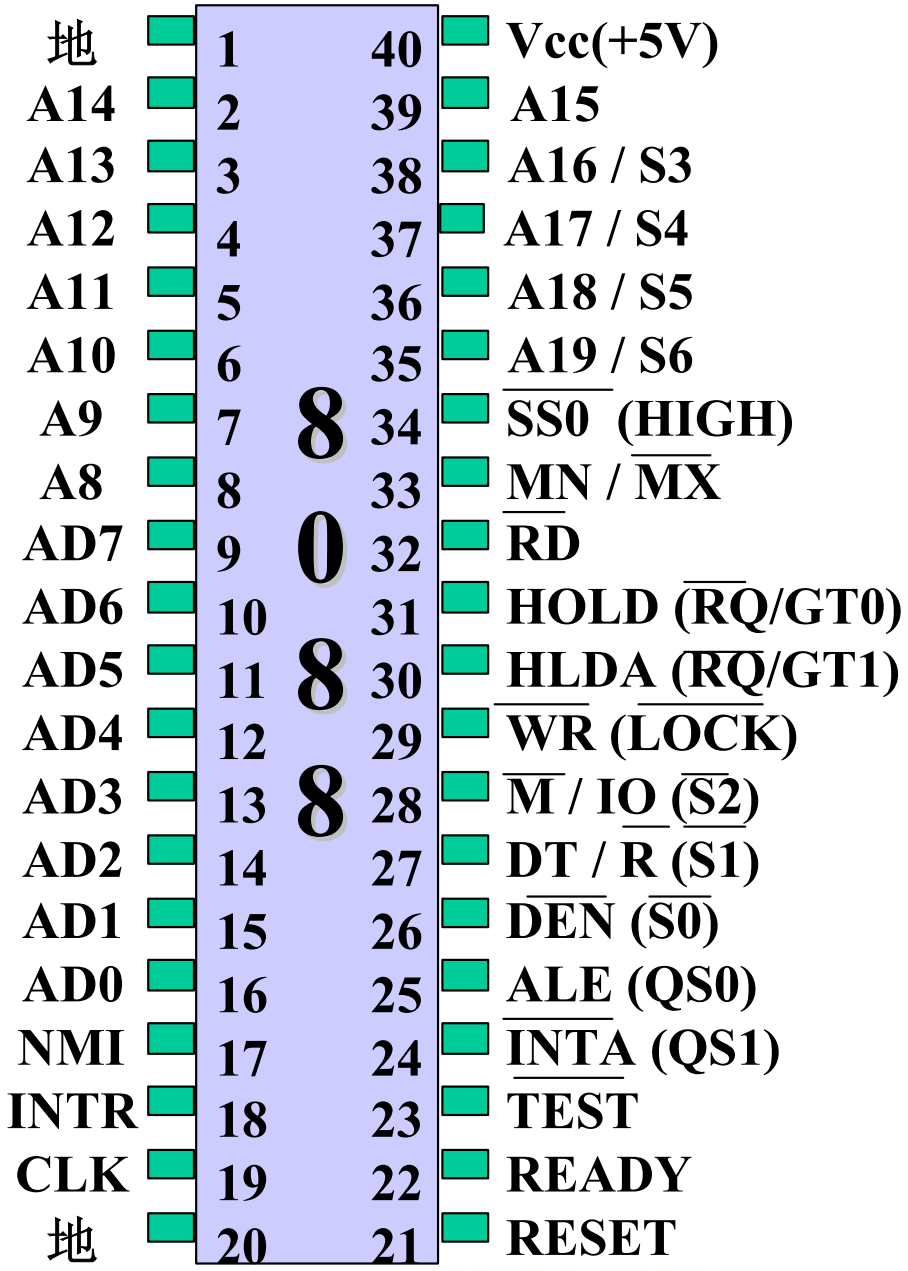
8088的外部引脚

- READY(22,准备就绪信号,入)
  - 高电平有效
  - 存储器或I/O口准备好时将该信号置为高电平。
- TEST(23, 测试信号,入)
  - 低电平有效
  - 执行WAIT指令时CPU监视TEST端，为低电平则执行WAIT后面的指令；为高时CPU空转。
  - 用来与外设同步。



- SS0(34, 系统状态, 出)

IO/$\overline{M}$	DT/$\overline{R}$	$\overline{SSO}$	Characteristics
1(HIGH)	0	0	Interrupt Acknowledge
1	0	1	Read I/O Port
1	1	0	Write I/O Port
1	1	1	Halt
0(LOW)	0	0	Code Access
0	0	1	Read Memory
0	1	0	Write Memory
0	1	1	Passive



8088的两种工作模式

- 最小模式
  - 构成小规模的应用系统——单处理器系统
  - 8088本身提供所有的系统总线信号
- 最大模式
  - 构成较大规模的应用系统——多处理器系统
    - 例如可以接入数值协处理器8087
  - 控制信号较多
    - 结合总线控制器8288形成系统总线号

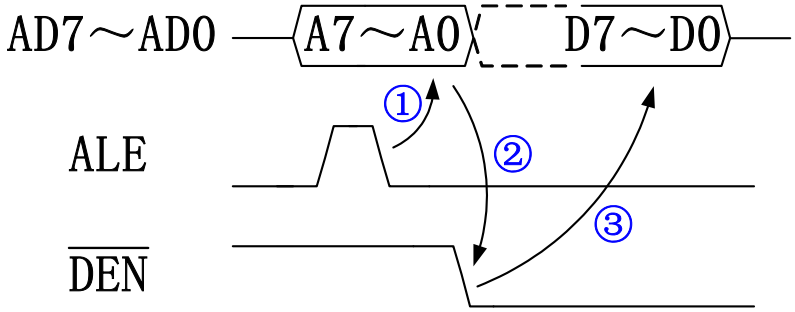
8088的两种工作模式

- 两种模式利用MN/$\overline{MX}$引脚区别
  - MN/$\overline{MX}$接高电平为最小模式
  - MN/$\overline{MX}$接低电平为最大模式
  - 硬件决定工作方式
- 两种模式内部操作没有区别
  - 本书：最小模式
  - IBM PC/XT：最大模式

第5节 8088处理器时序和总线周期

8088处理器的时序

- 时序概念（Timing）
  - 时序是指信号随时间和事件变化的顺序或规律。



  - 可用时序描述CPU通过总线对外实施的各种操作（总线操作）：
    - 存储器读操作
    - I/O读操作
    - 存储器写操作
    - I/O写操作
    - 中断响应操作
    - 总线请求及响应操作

和时序相关的几个概念

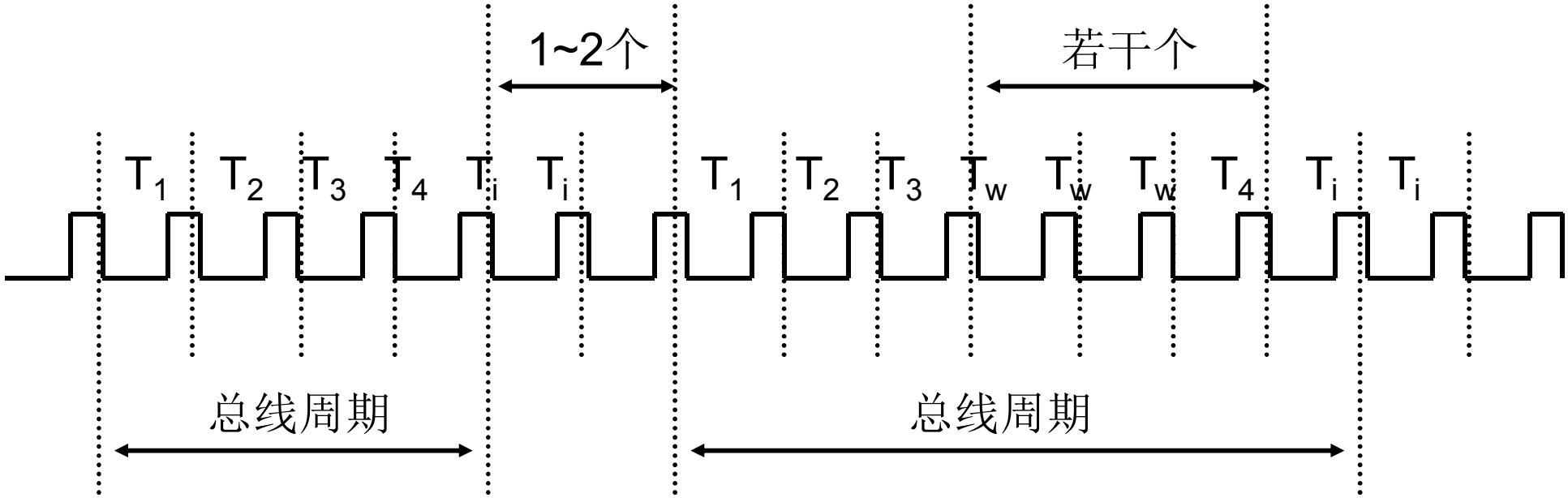
- 周期
  - ■时钟周期 | 总线周期 | 指令周期
- 时钟周期（Clock Cycle）
  - ■时钟的周期，时钟频率的倒数。 8088：4.77MHz
- 总线周期（ Bus Cycle ）
  - ■CPU通过总线与外部进行基本操作（一次数据交换）的过程
  - ■I/O读或写总线周期，存储器读或写总线周期,...
- 指令周期（ Instruction Cycle ）
  - ■取指、译码、读写操作数到执行完成的过程

指令周期 > 总线周期 > 时钟周期

- 总线周期（即总线操作）产生的例子
  - 指令取指阶段：存储器读总线周期(读取指令代码)
  - 源操作数是存储单元的指令：存储器读总线周期
  - 目的操作数是存储单元的指令：存储器写总线周期
  - 执行IN指令：I/O读总线周期
  - 执行OUT指令：I/O写总线周期
  - CPU响应可屏蔽中断：中断响应总线周期
- 空闲总线周期
  - CPU不执行任何存储单元或I/O操作，则执行空闲周期T_i（Idle）

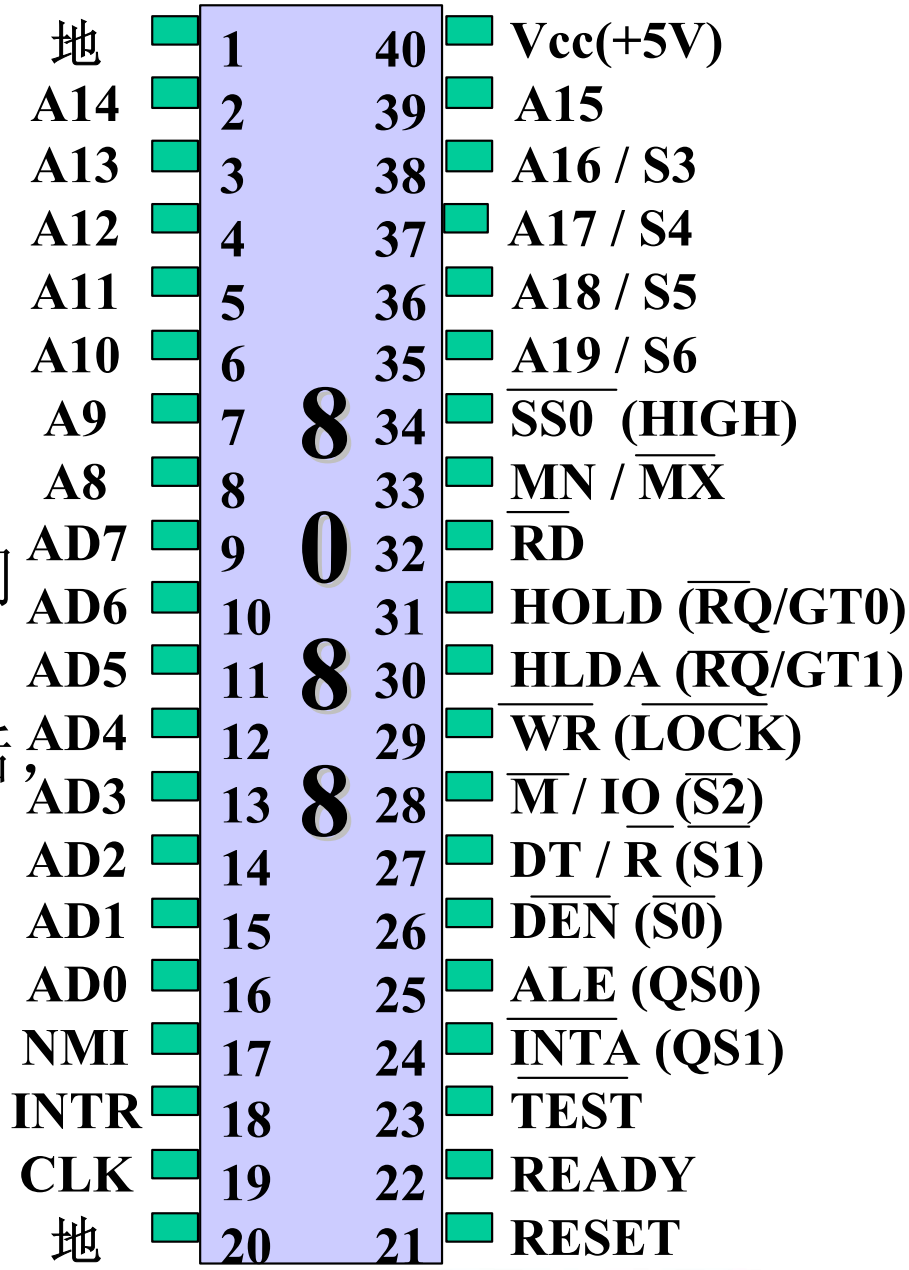
总线周期的构成

- (基本)总线周期需要4个时钟周期：T1、T2、T3和T4
  - 时钟周期也被称作“T状态” (T State)
  - 空闲周期T_i：空转，2个总线周期之间插入。
  - 等待周期Tw：用于延长总线周期，插入T3和T4之间

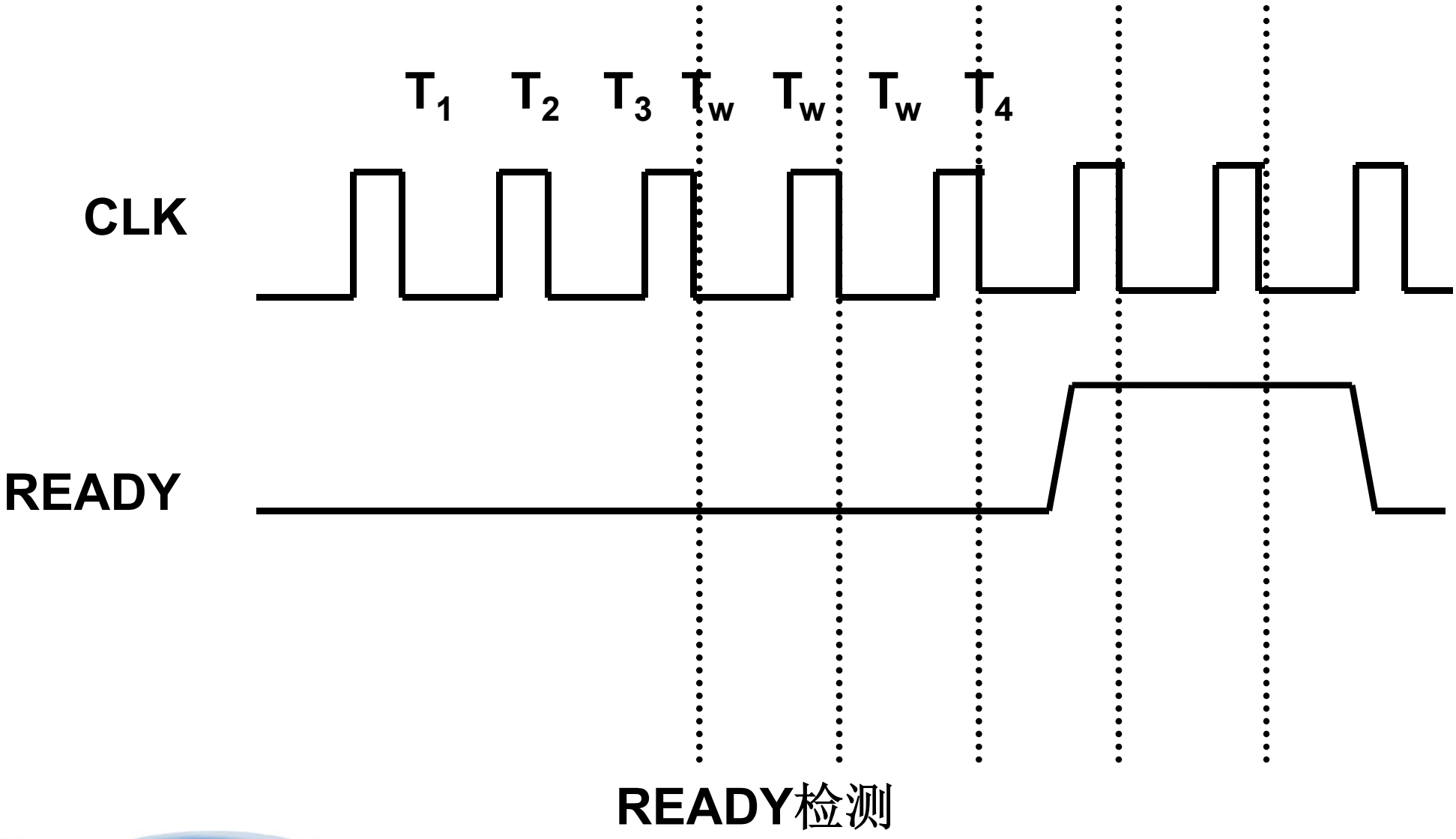


CPU和外设操作的同步：READY,等待周期Tw

- READY(22,准备就绪信号,入)
  - 高电平有效
  - MEM或I/O准备好时将该信号置高
  - CPU在T3期间采样READY信号
    - 1. T3期间检测READY是否有效？
    - 2. 如果READY无效，在T3和T4间插一个等效T3的Tw状态，转1
    - 3. 如果READY有效，执行完T3后，进入T4。

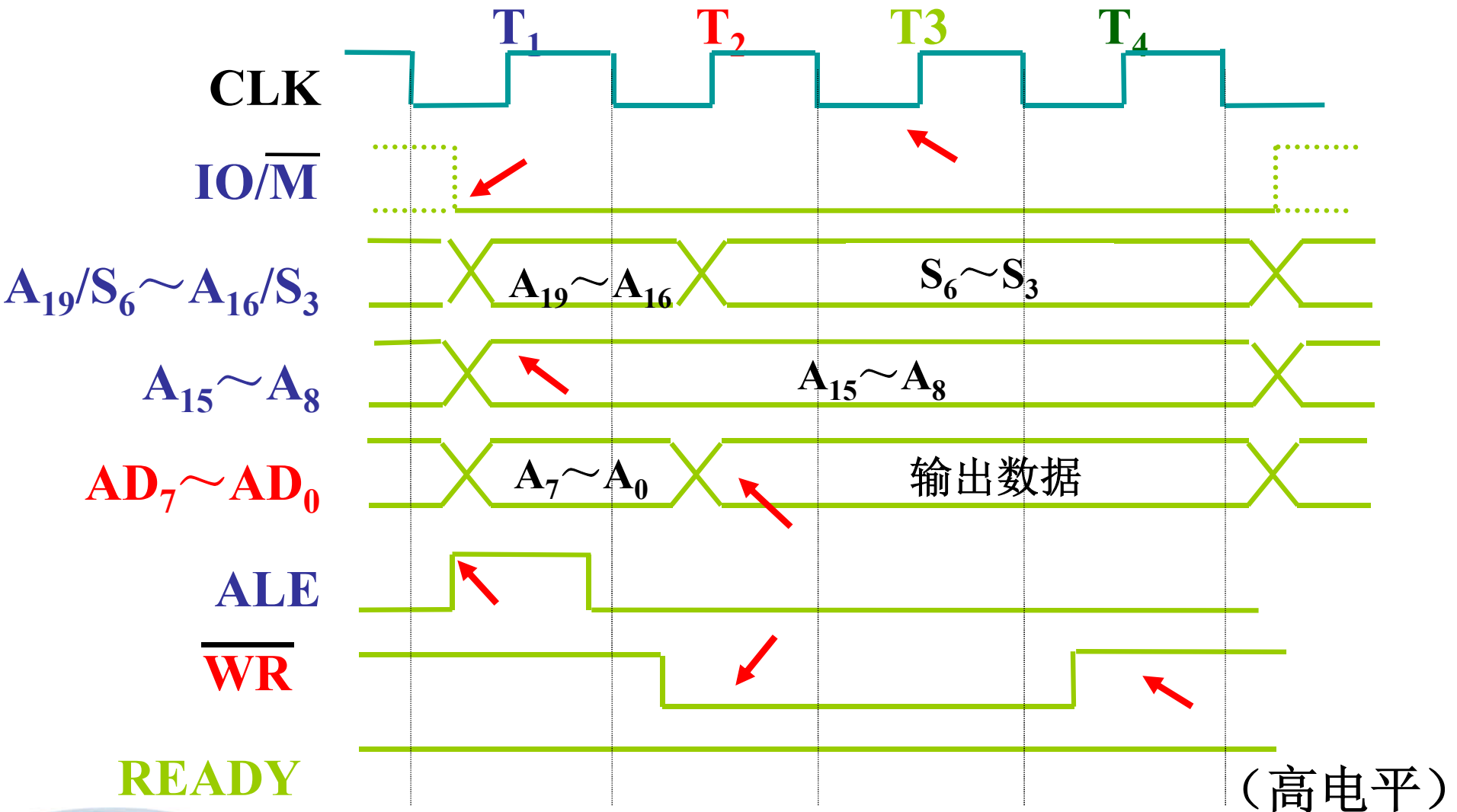


同步：T3和T4间插入等待状态Tw

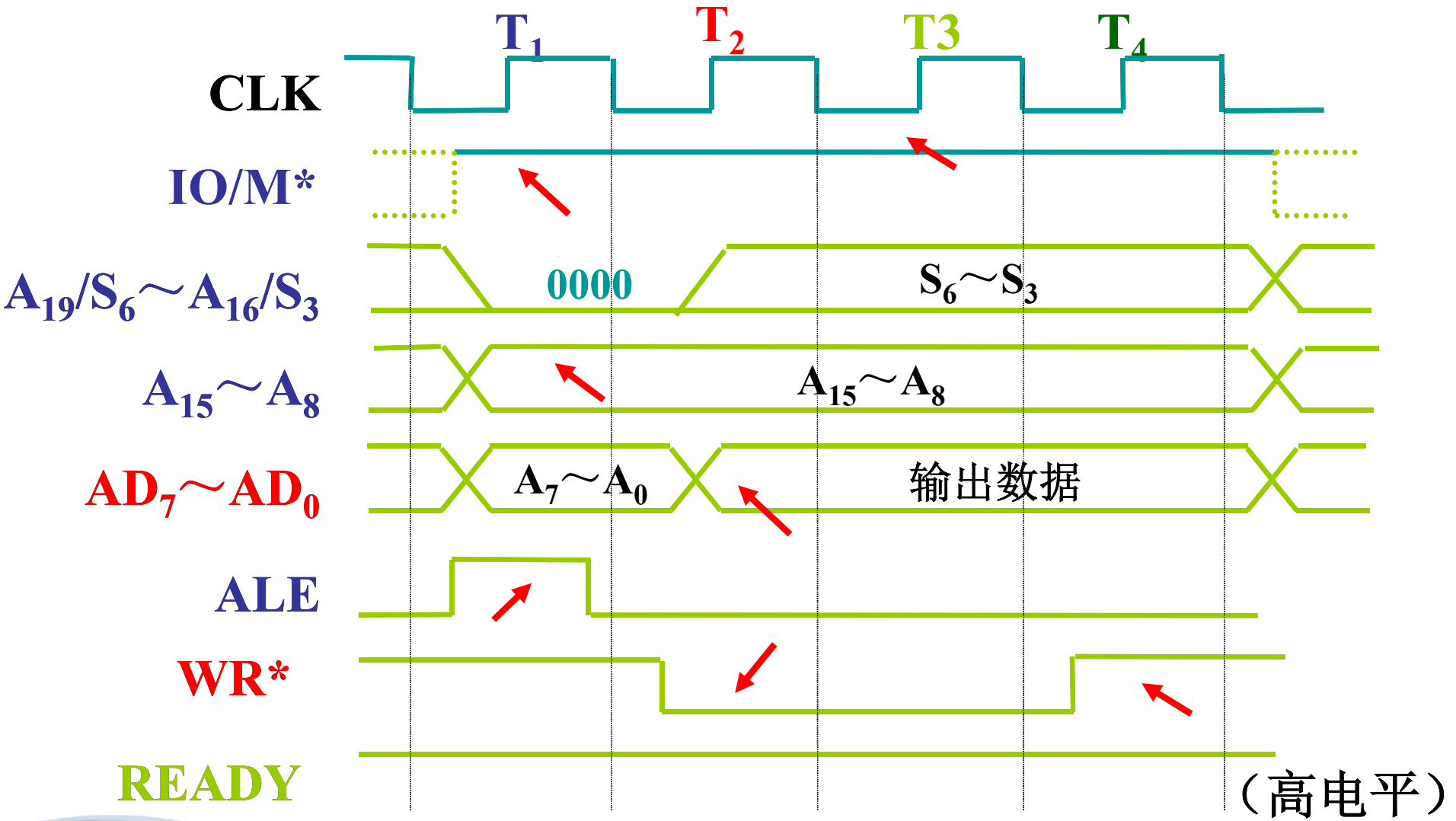


- 基本的总线周期
  - 存储器读
  - 存储器写
  - I/O端口读
  - I/O端口写
  - 中断响应

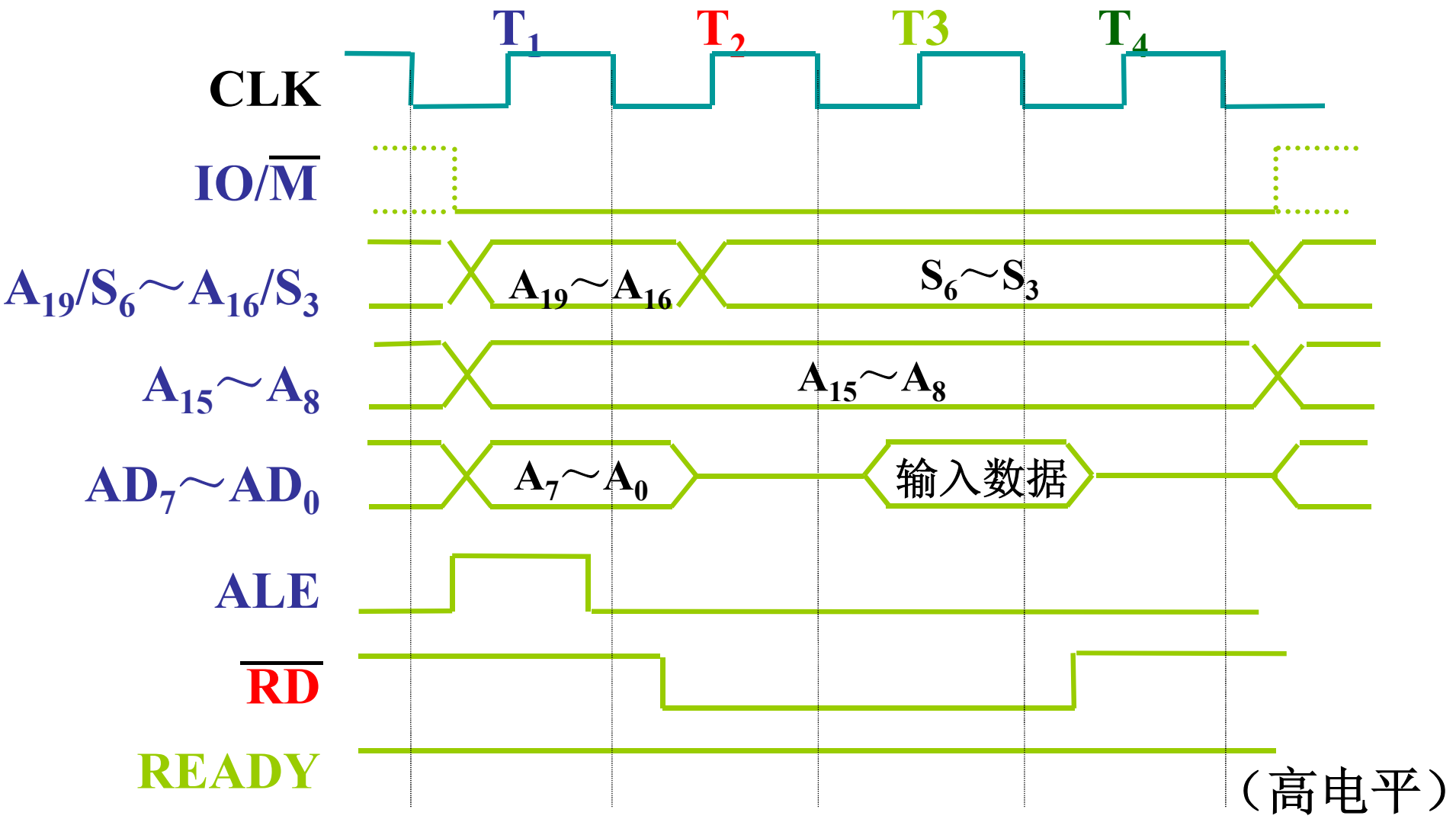
存储器写总线周期


T1
T2
T3
T4
CLK
IO/M
A19/S6～A16/S3
A19～A16
S6～S3
A15～A8
A15～A8
AD7～AD0
A7～A0
输出数据
ALE
WR
READY
（高电平）

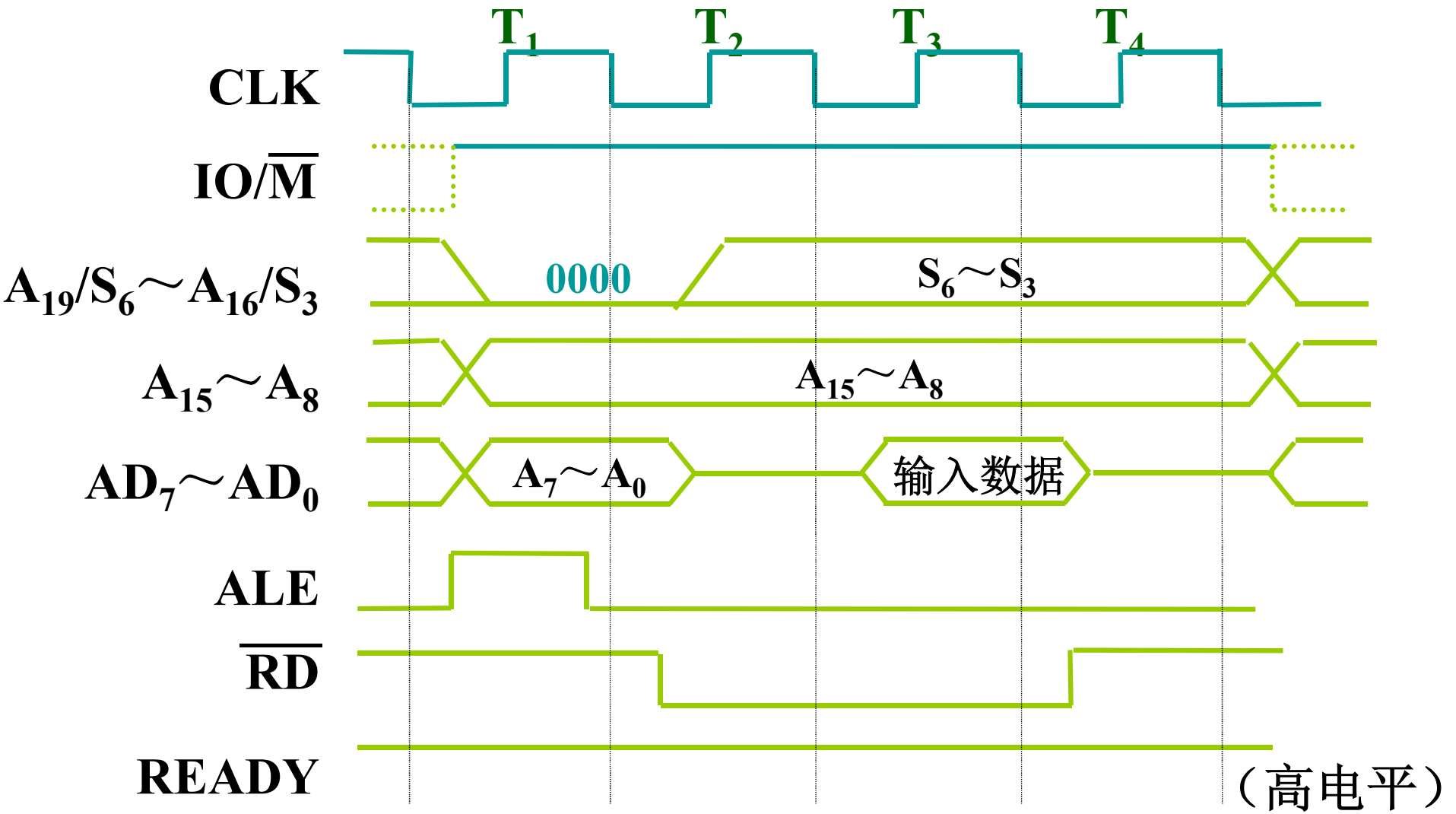
I/O写总线周期


T1
T2
T3
T4
CLK
IO/M*
A19/S6～A16/S3
0000
S6～S3
A15～A8
A15～A8
AD7～AD0
A7～A0
输出数据
ALE
WR*
READY
（高电平）

存储器读总线周期


T1
T2
T3
T4
CLK
IO/M
A19/S6～A16/S3
A19～A16
S6～S3
A15～A8
A15～A8
AD7～AD0
A7～A0
输入数据
ALE
RD
READY
（高电平）

I/O读总线周期


T1
T2
T3
T4
CLK
IO/M
A19/S6～A16/S3
0000
S6～S3
A15～A8
A15～A8
AD7～AD0
A7～A0
输入数据
ALE
RD
READY
（高电平）

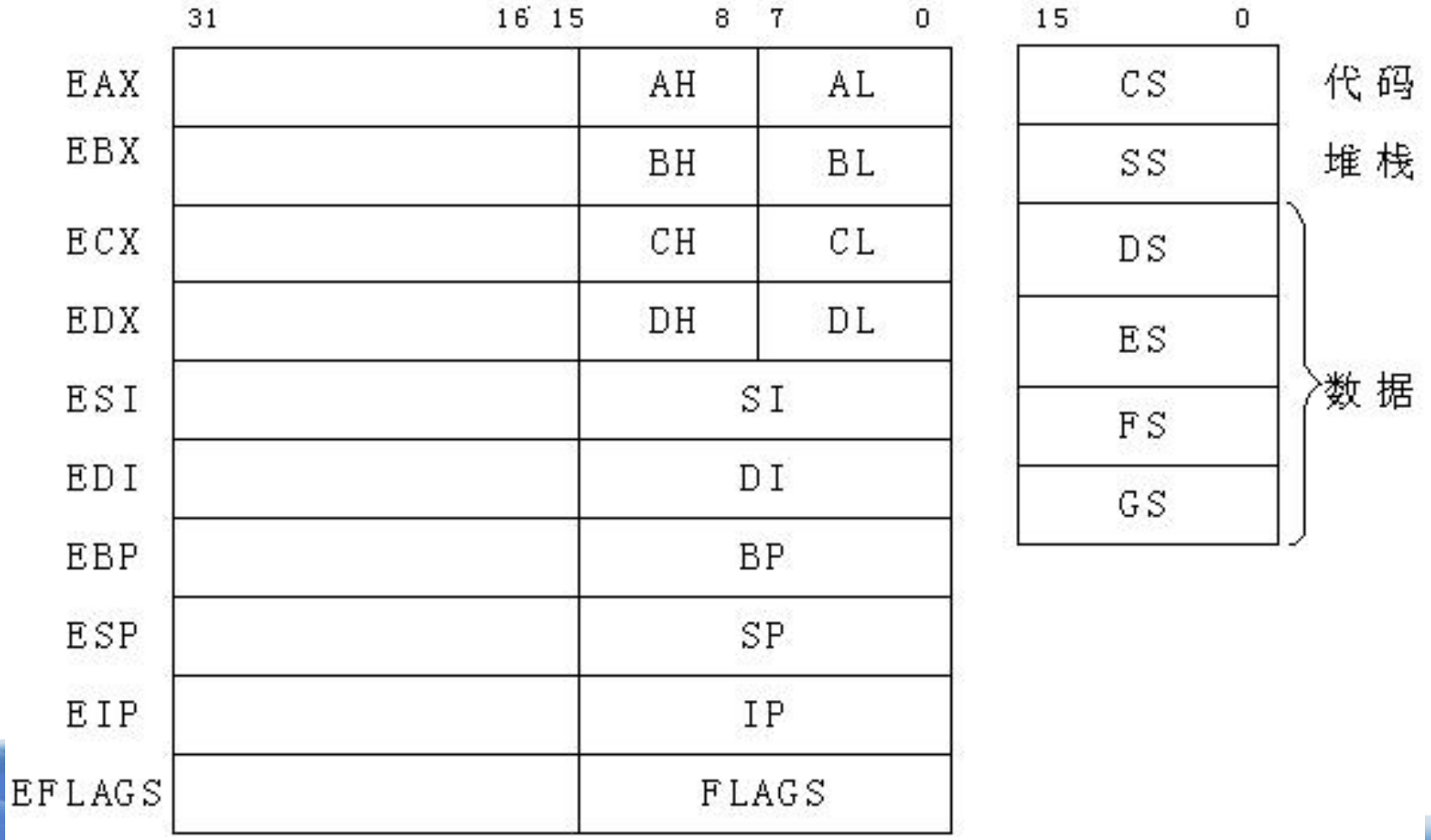
第6节 IA-32发展历史

IA-32的发展历史

- **IA-32微处理器是8086/8088的延伸**
  - 功能扩展
  - 性能扩展

功能扩展

- 1.从16位扩展为32位
  - 8086：16位，内存范围64K。
  - 80386 ： 32位，内存范围4G。(1985年)



功能扩展

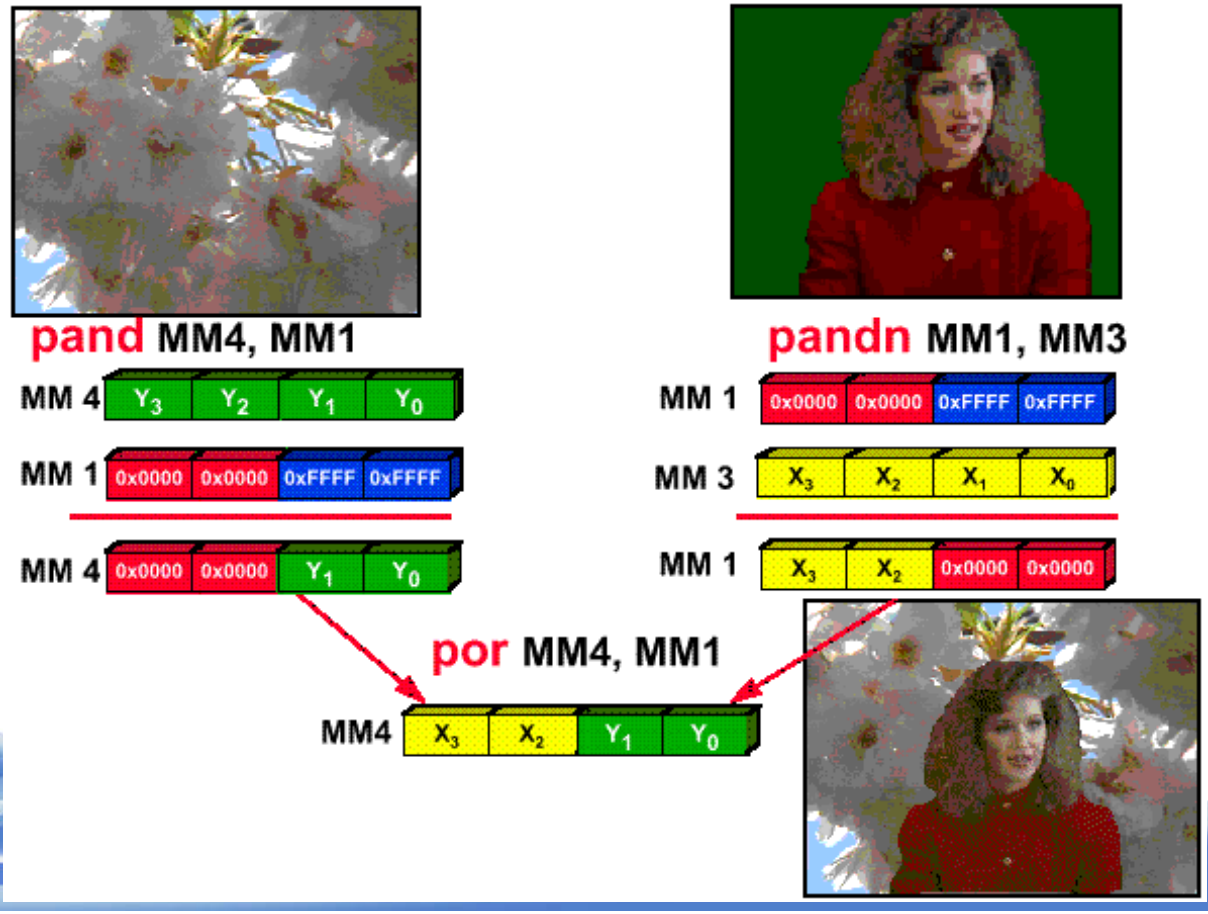
- 2.从实模式至保护模式
  - 从80286开始，在80386中真正完善保护模式。
  - 保护模式
    - 保护地址模式
    - 虚拟地址存储管理方式
    - 提供存储管理单元(MMU)
    - 增加支持多任务的指令。
    - 采用多级地址映射，把逻辑地址映射到物理地址。

功能扩展

- 3.浮点支持
  - 工程应用、图形处理、科学计算等
  - 自80486芯片开始集成x87（及其增强）浮点单元。

功能扩展

- 4.MMX技术
  - ■影像、语音、通信等处理：计算密集，矩阵运算
  - ■MMX增设57条指令，程序执行效率提高。
  - ■例：利用逻辑运算实现两幅图像的合成。

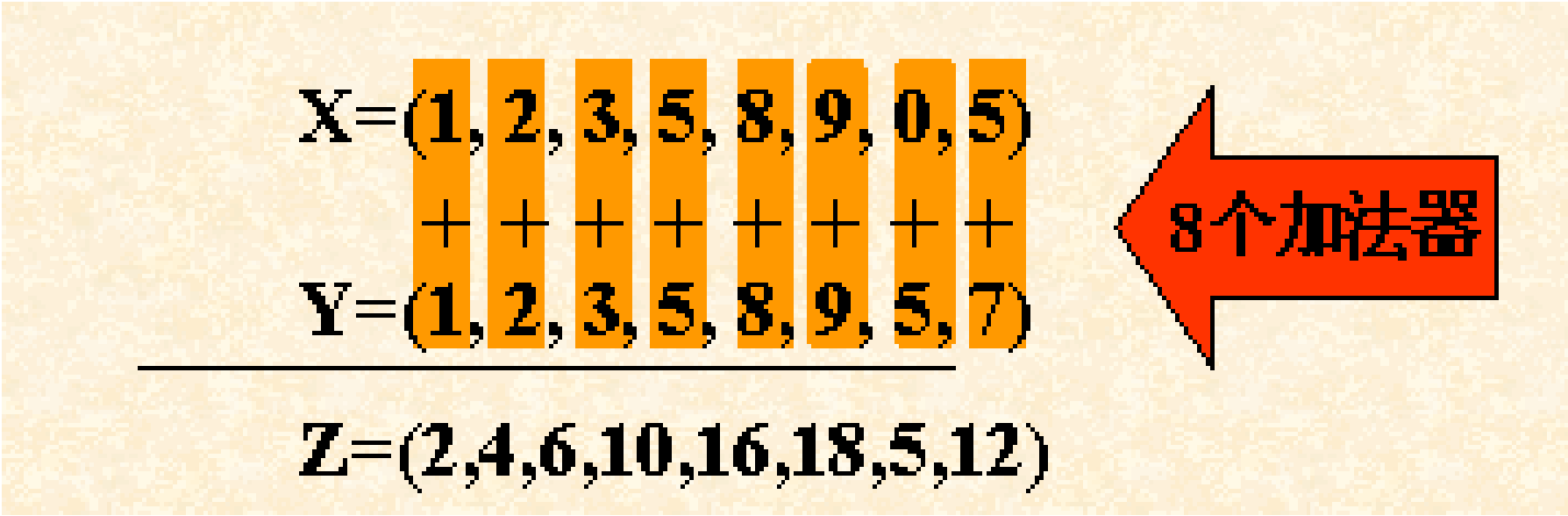


功能扩展

- **5.SIMD：** 单指令流多数据流

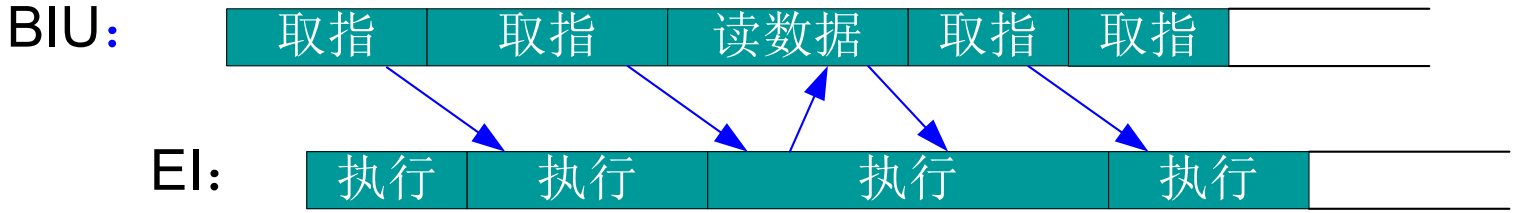
PADDB mm1, mm2

矢量相加，每项一个字节



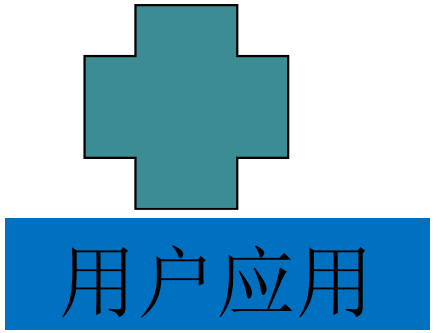
性能提高

- 1．流水线技术
  - 提高操作的并行性



- 2．片内缓存（Cache）
  - 程序局部性
  - 提高内存访问效率

当前处理器实现方式的演化




```
total = 0
for i = 1 to N  loop
    total += M[i]
end loop
```

通用处理器

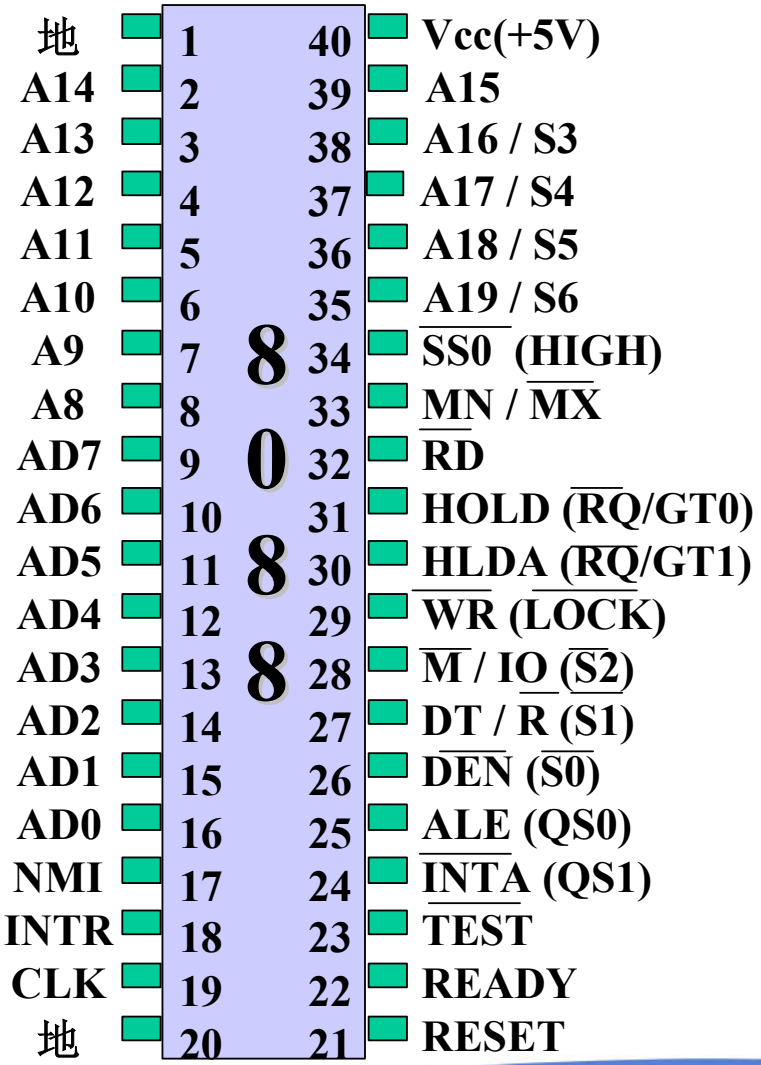
专用处理器

单用途处理器

- 典型的IA-32结构
  - 1.8088/8086
  - 2.80386
  - 3.奔腾（Pentium）
  - 4.Intel超线程处理器
  - 5.Intel多核处理器

1. 8088

- 8位CPU：寄存器16位；
- 20根地址线：寻址1MB内存；



2. 80386

- IA-32系列第一个32位处理器。
- 引入32位寄存器。
- 提供虚拟**8086**方式
  - 实模式 | 保护模式
  - 虚拟：在“保护模式”下提供“实模式”程序运行环境
- 能支持**4GB**内存。

3. 80486

- 第一次：指令执行采用流水线（**5**级）
- 第一次：引入缓存（**CACHE**）
- 第一次：引入 **FPU**（浮点处理单元）

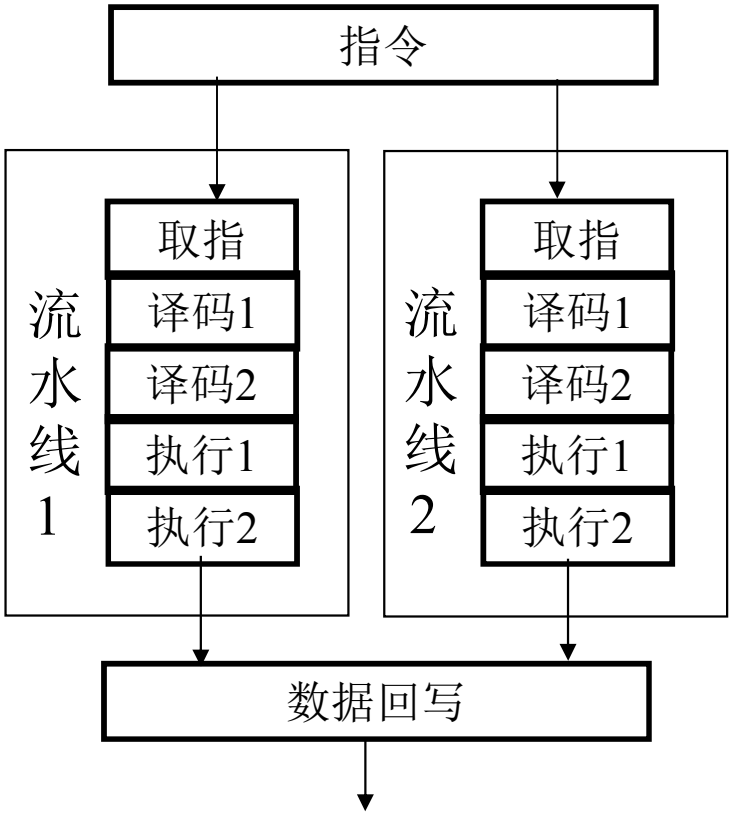
4. 奔腾（Pentium）（ 1995 ）

- 增加第二条流水线，达到超标量性能。
  - Pentium II （引入MMX）
  - Pentium II Xeon（至强）
  - Celeron（赛扬）
  - Pentium III（引进流SIMD扩展，即SSE）
  - Pentium III Xeon（至强）
  - Pentium 4（ 基于Intel NetBurst微结构）

超标量(Super Scalar)技术

- **超标量(Super Scalar)技术：**

  是RISC采用的有一种处理技术，是指在一个时钟周期内CPU可以执行一条以上的指令。它通过内装多条流水线来同时执行多个处理，其实质就是以空间换取时间。



5. Intel超线程处理器

- 超线程技术（ Hyper-Threading，简称“HT”）
  - HT利用特殊的硬件指令，把两个逻辑内核模拟成两个物理芯片，让单个处理器能支持线程级并行计算。
- 双核技术是CPU含两个硬核实现多线程。