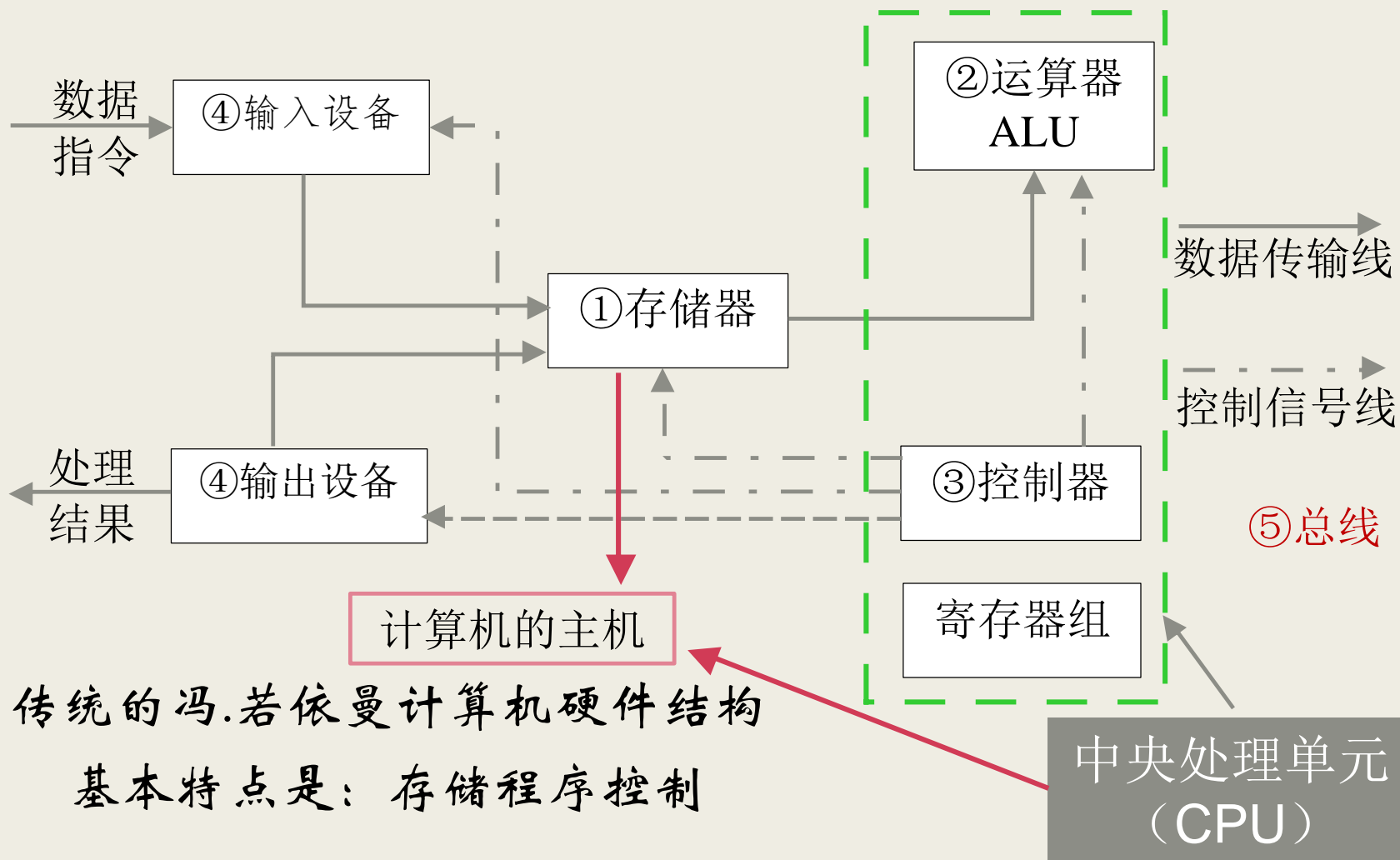


第2章

微处理器与系统结构（I）

微型计算机的基本硬件构成 (冯氏结构, 5大部分)



传统的冯·诺依曼计算机硬件结构

基本特点是：存储程序控制

由于制造工艺的原因，
微处理器的结构受3方面限制

引脚数限制

芯片面积限制

器件速度限制

16位微处理器基本结构具有
如下特点：

引脚功能复用

单总线、累加器
结构

可控三态电路

总线分时复用

Intel 8086

- 第一代超大规模集成电路，采用HMOS工艺制造，内含29万个晶体管。
- 有16根数据线和20根地址线，可寻址内存空间达1MB(2^{10} B)；可访问64K字节的输入/输出端口，16位数据总线与地址总线复用。
- 特点是采用并行流水线工作方式，通过设置指令预取队列实现；对内存空间实行分段管理；支持多处理器系统，可工作于最小和最大两种工作方式。



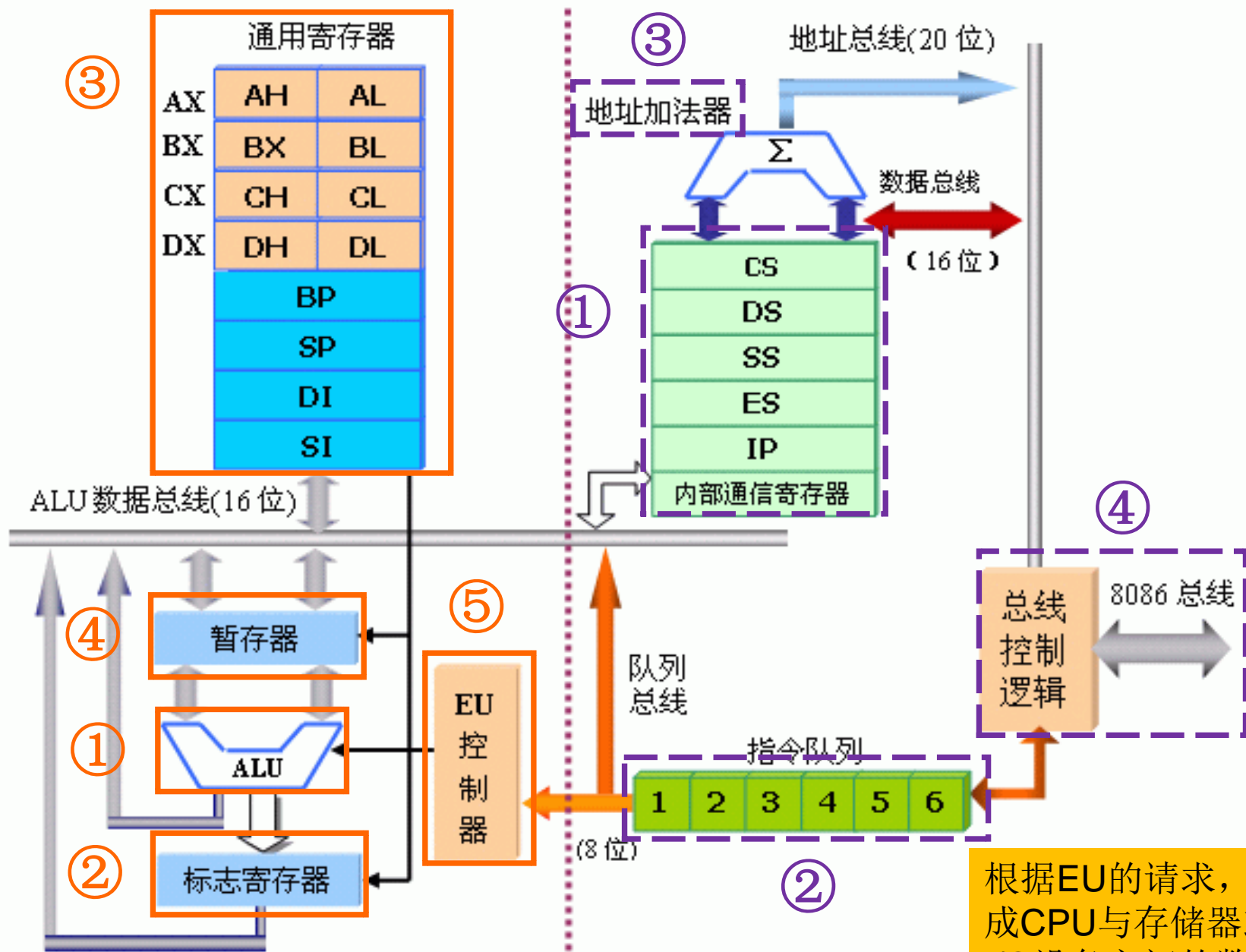
2.1 8086/8088 CPU的结构

2.1.1 8086/8088 CPU的内部结构

❖ 从功能上划分为两个独立的逻辑单元:

①总线接口部件BIU(Bus Interface Unit)

②执行部件EU(Execution Unit)



根据EU的请求，完成CPU与存储器或I/O设备之间的数据传送。

负责指令的译码和执行 指令执行单元(EU)

总线接口单元(BIU)

1、指令执行部件EU

计算机按照

取指令 → 指令译码 → 取操作数 → 执行指令 → 存放结果 的顺序进行操作；

EU可不断地从BIU指令队列缓冲器中取得指令并连续执行，省去访问存储器取指令所需时间；

如果指令执行过程中需要访问存储器存取数据时，只需将要访问的地址送给BIU，等待操作数到来后在继续执行；

EU无直接对外的接口，要译码的指令将从BIU的指令队列中获取；

BIU、EU两者并行操作，提高了CPU的运行效率

2、总线接口部件BIU

❖ 相关部件的功能：

- (1)指令队列缓冲器：存放6字节的指令代码，按“先进先出”的原则进行存取操作；
- (2)地址加法器和段寄存器：用于形成存储器的物理地址，完成从16位存储器逻辑地址到20位的物理存储器地址的转换运算；
- (3)指令指针寄存器IP：用于存放BIU要取的下一条指令的段内偏移地址；
- (4)总线控制电路与内部通信寄存器：前者用于产生外部总线操作时的相关控制信号，后者用于暂存BIU和EU之间交换的信息

BIU和EU的动作协调原则

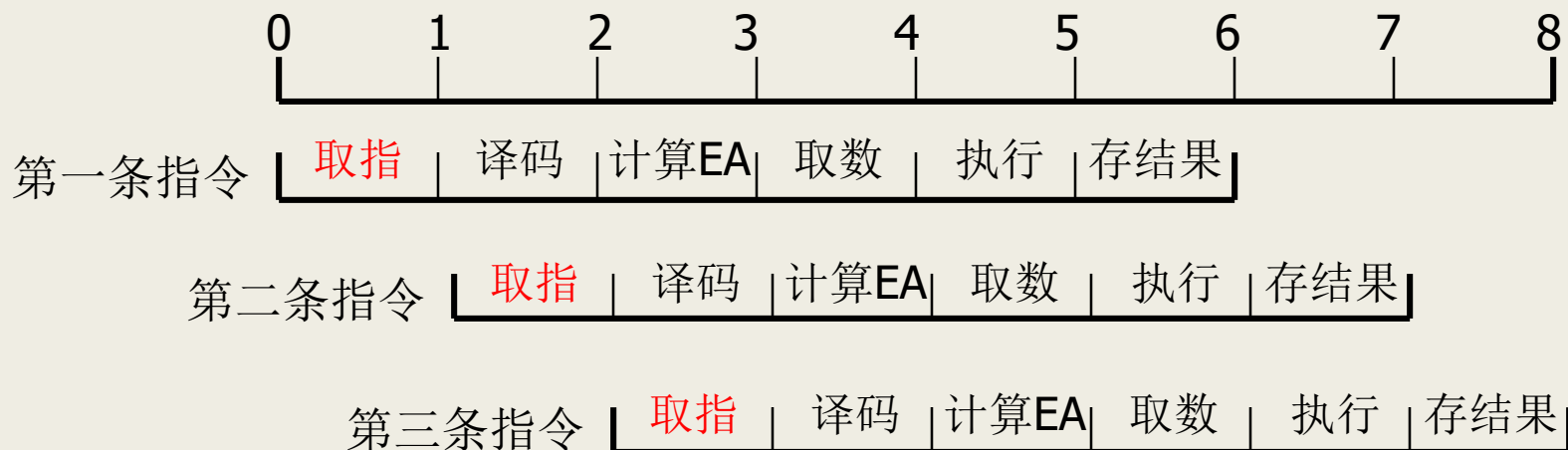
①每当8086的指令队列中有两个空字节，BIU就会自动把指令取到指令队列中。其取指的顺序是按指令在程序中出现的先后顺序；

BIU从CS中取出16位段地址，再从IP中取出16位偏移地址，经地址加法器形成20位地址，送上地址总线并找到该地址所在的内存单元，取出相关指令依次放入指令队列缓冲器中。

②每当EU准备执行一条指令时，它会从BIU部件的指令队列前部取出指令的代码，然后用几个时钟周期去执行指令。

- 在执行指令的过程中，如果必须访问存储器或者I/O端口，那么EU就会请求BIU，进入总线周期，完成访问内存或者I/O端口的操作；
 - 如果此时BIU正好处于空闲状态，会立即响应EU的总线请求。
 - 如BIU正将某个指令字节取到指令队列中，则BIU将首先完成这个取指令的总线周期，然后再去响应EU发出的访问总线的请求。
- ③当指令队列已满，且EU又没有总线访问请求时，BIU便进入空闲状态。
- ④在执行转移指令、调用指令和返回指令时，由于待执行指令的顺序发生了变化，则指令队列中已经装入的字节被自动消除，BIU会接着往指令队列装入转向的另一程序段中的指令代码。

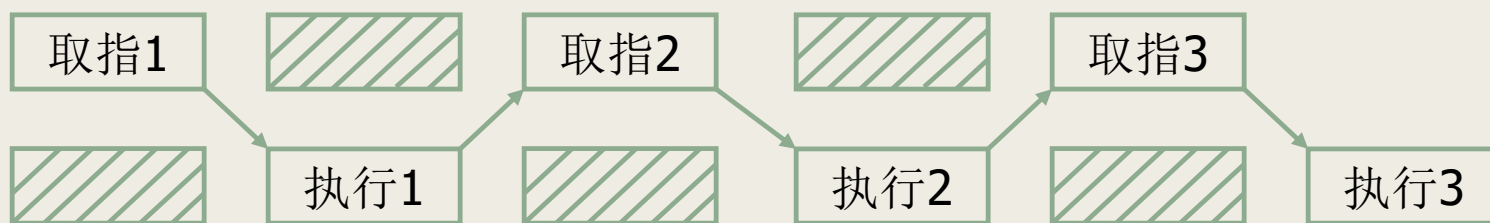
关于流水线计算机



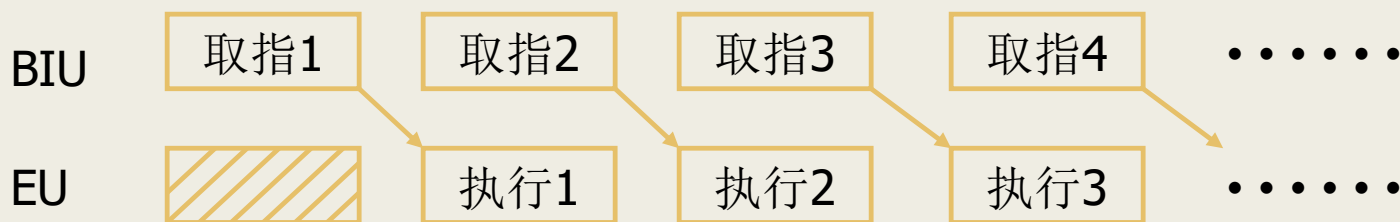
3条指令共需8个时间单位，即可全部执行完；
如果完全串行执行，则需 $3 \times 6 = 18$ 个时间单位。

显然，采用“流水线”技术可以显著提高计算机的处理速度。

早期的计算机将这两步采用先后轮流动作（串行），CPU效率较低。



在流水线方式下，BIU与EU同时动作（并行）完成指令周期，CPU效率高。



3、8086/8088CPU寄存器阵列(寄存器组)

CPU中大部分指令是在作数的预定功能；

寄存器用来暂时存放成运算过程的中间结果；

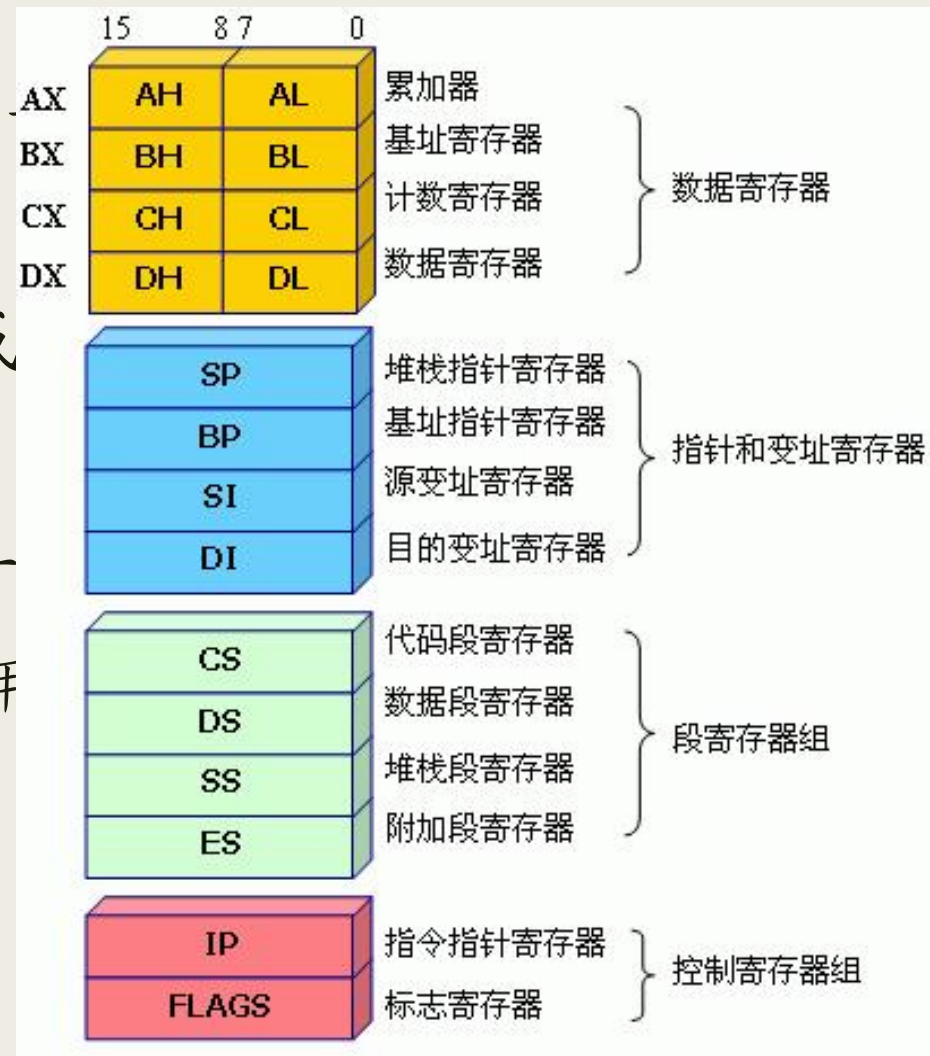
不必访问存储器，提升

14个16位的寄存器（用

1) 通用寄存器

2) 控制寄存器

3) 段寄存器



❖通用寄存器： 8个16位通用寄存器， 分成两组

①数据寄存器

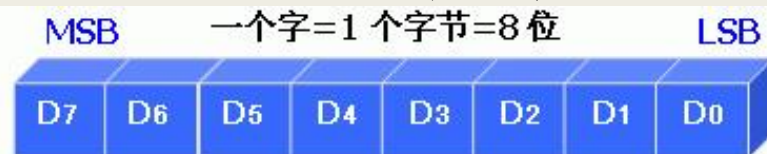
由AX、BX、CX、DX构成，可用来存放16位的数据或地址

也可把它们当作8个8位寄存器来使用，即把

每个通用寄存器的

低半部分为AL、BL

高半部分为AH、BH



②指针和变址寄存器

这组寄存器存放的内容是某一段内地址偏移量，用来形成操作数地址。主要左堆栈用于存取位于当前堆栈段中的数据。

SP和BP用于访问堆栈中的数据；SI和DI用于访问数据存储器中的数据

SP—堆栈指针寄存器
BP—基址指针寄存器

SI—源变址寄存器
DI—目的变址寄存器

SP—PUSH和POP指令由SP给出栈顶偏移地址，在内存寻址中，凡涉及到SP则默认的段寄存器为SS。

BP—指示现行堆栈段的一个数据区的“基址”偏移量，用它来访问堆栈段中的数据。

用来存放字符串操作时源与目的操作数段内的偏移地址。所以又称为源变址和目的变址寄存器。SI与DI在串操作时对应的段寄存器分别为DS与ES，在使用时不可混淆

通用寄存器的特定用法

寄存器	常用的操作功能
AX	字乘法，字除法，字I/O
AL	字节乘，字节除，字节I/O，十进制算术运算
AH	字节乘，字节除
BX	转移
CX	串操作，循环次数
CL	变量移位，循环控制
DX	字节乘，字节除，间接I/O
SP	堆栈操作
SI	数据串操作
DI	数据串操作

❖ 控制寄存器：2个

① 指令指针寄存器IP：16位

用来存放要取的下一条指令在当前代码段中的偏移地址，控制CPU的指令执行顺序；

和代码段寄存器CS一起可以确定当前所要取的指令的内存地址；

顺序执行程序时，CPU每取一个指令字节，IP自动加1，指向下一个要读取的字节；当IP单独改变时，会发生段内的程序转移；当CS和IP同时改变时，会产生段间的程序转移；

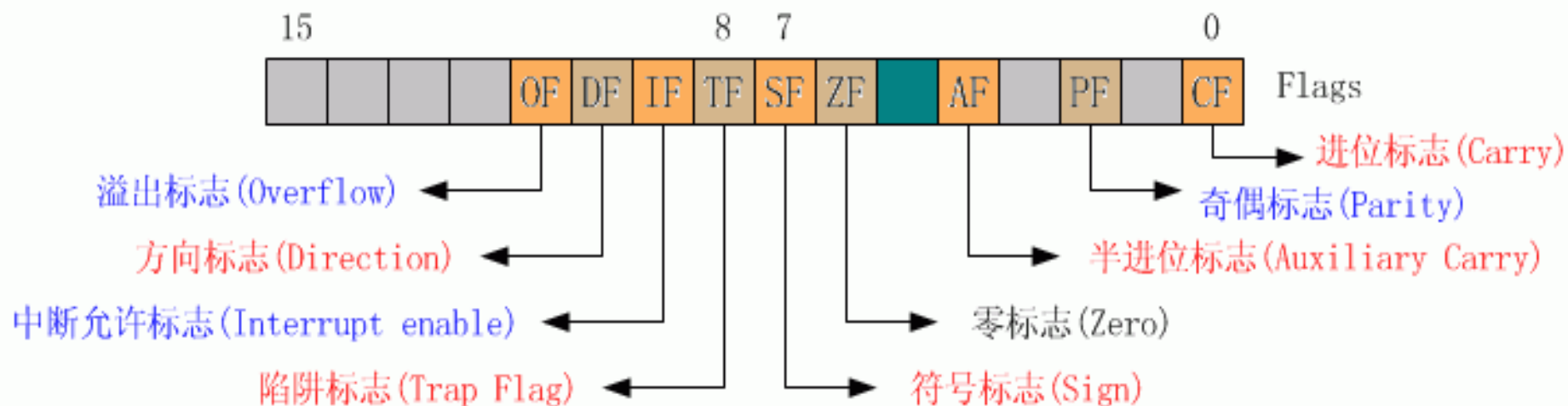
IP与PC又掠略有差别，
8086中IP要与CS配合才能形成真正的物理地址。用户不可以随意改变其值，只有遇到CALL、INT、JMP等指令时才可改变其值。
程序不能直接访问IP，在用DEBUG调试程序时可用调试命令改变其值。

②标志寄存器FR(处理器状态字)

16位寄存器，用了其中的9位，其它7位在8086/8088中无意义。

6位用作状态标志：OF、SF、ZF、AF、PF、CF

3位用作控制标志：DF、IF、TF



❖ 状态标志

用来反映EU执行算术或逻辑运算的结果特征，常作为条件转移类指令的测试条件，以控制程序的运行方向；

CPU在进行算术/逻辑运算时，根据操作结果自动将状态标志位置位(置1)或复位(清0)。

❖ 控制标志

用来控制CPU的工作方式或工作状态，一般由程序设置或由程序清除；

事先用指令设置，在程序执行时检测这些标志，用以控制程序的转向。

1) CF(Carry Flag) 进借位标志位

CF=1 表示本次运算中最高位(第7位或第15位)有进位(加法运算时)或有借位(减法运算时)。

CF=0 表示本次运算中最高位(第7位或第15位)无进位(加法运算时)或无借位(减法运算时)。

【例】（以8位运算为例）：

3AH + 7CH =

B6H

没有进位：CF=0

AAH + 7CH =

26H

有进位：CF=1

2) AF(Auxiliary Carry Flag) 辅助进借位标志位

AF=1 表示8位(或16位)运算结果(限使用AL寄存器)中低4位向高4位有进位(加法运算时)或有借位(减法运算时)，这个标志位只在BCD数运算中起作用。

【例】 $3AH + 7CH = B6H$

D_3 向前有进位：AF=1

这个标志主要由处理器内部使用，用于十进制算术运算的调整，用户一般不必关心。

3) OF(Overflow Flag) 溢出标志位

OF=1 表示算术运算结果产生溢出，否则OF= 0。
溢出标志位是根据操作数的符号及其变化情况设置的。

例如，加法运算时，两个操作数符号相同，而结果的符号与之相反，则OF= 1，否则OF= 0。

【例】

$$3AH + 7CH = B6H$$

产生溢出：OF=1

$$AAH + 7CH = 26H$$

没有溢出：OF=0

❖ 什么是溢出？

“溢出”针对有符号数

处理器内部以补码表示有符号数，范围：

$$-2^{n-1} \leq X \leq 2^{n-1} - 1$$

8位表示范围是： -128 ~ +127

16位表示范围是： -32768 ~ +32767

如果运算结果超出了这个范围，就是产生了溢出，溢出发生时，说明有符号数的运算结果不正确

❖ 溢出标志和进位标志的区别

溢出标志OF和进位标志CF是两个意义不同的标志

进位标志：无符号数运算结果是否超出范围，运算结果仍然正确；

溢出标志：有符号数运算结果是否超出范围，运算结果已经不正确。

❖ 溢出和进位的对比

【例1】 $3AH + 7CH = B6H$

无符号运算： $58 + 124 = 182$

不超范围，无进位

有符号运算： $58 + 124 = 182$

超范围，有溢出

【例2】 $AAH + 7CH = 26H$

无符号运算： $170 + 124 = 294$ ，

超范围，有进位

有符号运算： $-86 + 124 = 38$ ，

不超范围，无溢出

❖ 如何运用溢出和进位

处理器对两个操作数进行运算时，
根据无符号运算有无进位来设置进位标志CF；
根据有符号运算是否超出表示范围来设置溢出标志OF；

应该利用哪个标志，则由程序员来决定。

如果参加运算的操作数是无符号数，用户应该关心进位标志；

如果参加运算的操作数是有符号数，用户应该关心溢出标志。

❖ 如何判断CF标志和OF标志(1)

➤ 以8位加运算位例，判断CF标志和OF标志：

无符号运算：A8H + 67H =

$$168 + 103 = 271$$

有进位：CF = 1


有符号运算：A8H + 67H =

$$-88 + 103 = 15$$

有进有出：OF = 0

1010 1000
+ 0110 0111

1 0000 1111



有进有出/无进无出 不溢出

有进无出/无进有出 溢出

❖ 如何判断CF标志和OF标志(2)

➤ 以8位减运算为例，判断CF标志和OF标志：

无符号运算：A8H - 67H =

$$168 - 103 = 65$$

够减：CF = 0

有符号运算：A8H - 67H =

$$-88 - 103 = -192$$

无进有出：OF = 1

	1010 1000	-	0110 0111	
有	1010 1000	+	1001 1001	1010 1000
			+	1001 1001
			<hr/>	
			1	0100 0001

4) ZF(Zero Flag) 零标志位

ZF=1, 表示运算结果为0(各位全为0),
否则ZF=0。

【例】

$$3AH + 7CH = B6H$$

结果不是零: ZF = 0

$$86H + 7AH =$$

00H

结果是全零: ZF = 1

注意: ZF为1表示的结果是0

5) SF(Sign Flag) 符号标志位

SF=1，表示运算结果的最高位(第7位或第15位)为“1”，否则SF=0。

【例】

$$3AH + 7CH = B6H$$

最高位 $D_7=1$: SF = 1

$$86H + 7AH = 00H$$

最高位 $D_7=0$: SF = 0

有符号数利用最高有效位来表示它的符号。所以，运算结果的最高位与符号标志SF相一致。

6) PF(Parity Flag) 奇偶标志位

PF= 1，表示本次运算结果的低八位中有偶数个“1”；

PF= 0，表示有奇数个“1”。

【例】 $3AH + 7CH = B6H$

$= 10110110B$

结果中有5个1，是奇数，则 $PF = 0$

注意：PF标志仅反映最低8位中“1”的个数是偶或奇，即使是进行16位字操作。

7) DF(Direction Flag) 方向标志位

在串操作指令中，若DF=0，表示串操作指令执行后地址指针自动增量，串操作由低地址向高地址方向进行；

DF=1，表示地址指针自动减量，即串操作由高地址向低地址方向进行。

DF标志位可通过STD指令置位，也可通过CLD指令复位。

CLD 指令复位方向标志：DF=0

STD 指令置位方向标志：DF=1

8) IF(Interrupt Flag) 中断允许标志位

IF=1，表示允许CPU响应可屏蔽中断INTR。

IF标志可通过STI指令置位，也可通过CLI指令复位

设置IF=1，则允许中断；

设置IF=0，则禁止中断。

CLI 指令复位中断标志：IF=0

STI 指令置位中断标志：IF=1

9) TF(Trap Flag) 陷阱标志位

TF=1，表示控制CPU进入单步工作方式。

设置TF=0，处理器正常工作；

设置TF=1，处理器每执行一条指令就中断一次，中断编号为1（称单步中断）

TF 也被称为单步标志。

利用单步中断可对程序进行逐条指令的调试。这种逐条指令调试程序的方法就是单步调试。

1-低8位有偶数个1
0-低8位有奇数个1

D0

OF DF IF TF SF ZF AF PF CF

D15

方向标志

单步中断

符号标志

零标志

1-结果为0
0-结果不为0

半进借位标志

1-低4位向高4位有进、借位
0-低4位向高4位无进、借位

奇偶标志

1-有进、借位
0-无进、借位

进借位标志

溢出标志

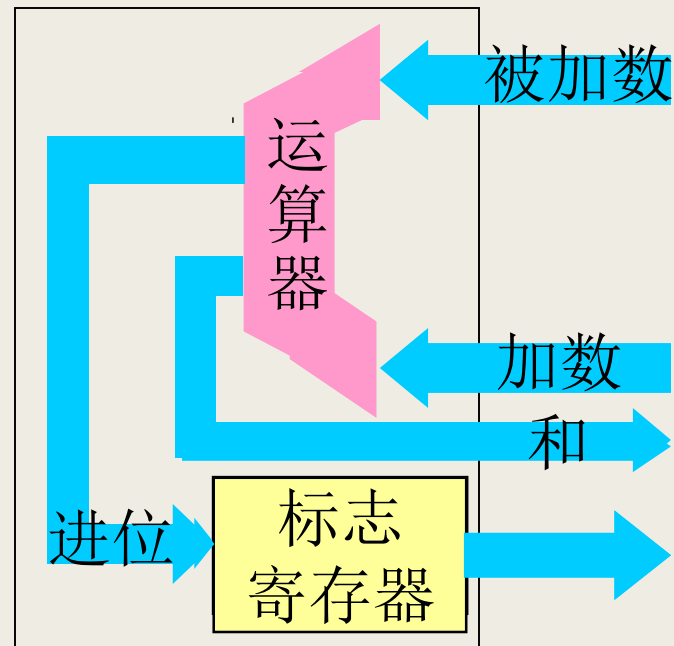
中断允许

【例】 8位二进制加法如下，给出各状态标志位的值

		1	0	1	1	0	1	0	1
+		1	0	0	0	1	1	1	1
进位	1		1	1	1	1	1	1	1
		0	1	0	0	0	1	0	0

被加数8
加数8位

和8位



最高位D7位产生进位:

CF = 1

D3位产生进位:

AF = 1

相加的结果为44H, 不为0:

ZF = 0

结果的最高位为0:

SF = 0

两负数相加结果为正, 溢出:

OF = 1

结果中有2个1, 偶数个1:

PF = 1

15	14			11	10	9	8	7	6		4		2		0
				OF	DF	IF	TF	SF	ZF		AF		PF		CF
				1				0	0		1		1		1

【练习1】给出各状态标志位的值

$$\begin{array}{r} 01011011\text{B} \\ + 01000100\text{B} \\ \hline 10011111\text{B} \end{array}$$

两个8位数相加，差是一个非0的负数：

SF = 1, ZF = 0;

结果中“1”的个数是6(偶数): PF = 1;

最高有效位无进位: CF = 0;

运算时低半字节向高半字节无进位: AF = 0;

正数 + 正数 = 负数，产生溢出: OF = 1

【练习2】给出各状态标志位的值

$$\begin{array}{r} 0110100001011010B \\ - 1000010010100010B \\ \hline 1110001110111000B \end{array}$$

两个16位数相减，差是一个非0的负数：

SF = 1, ZF = 0;

结果中低8位“1”的个数是4(偶数): PF = 1;

最高有效位有借位: CF = 1;

运算时低半字节向高半字节无借位: AF = 0;

正数 - 负数 = 负数，产生溢出: OF = 1

❖ 段寄存器：4个16位

8086 CPU可直接寻址1MB的存储器空间，直接寻址要20位地址码，内部寄存器都是16位，只能直接寻址64KB，采用分段技术来解决。

8086系列把整个存储空间分成许多逻辑段，每段容量不超过64KB。这些逻辑段在整个存储空间中可浮动。

段和段之间可以是连续的（整个存储空间分成16个逻辑段），也可以是分开的或重叠的。

8086微处理器共有4个16位段寄存器，存放每一个逻辑段的段起始地址。



代码段寄存器CS：存放当前代码段的段地址；

数据段寄存器DS：存放当前数据段的段地址；

堆栈段寄存器SS：存放当前堆栈段的段地址；

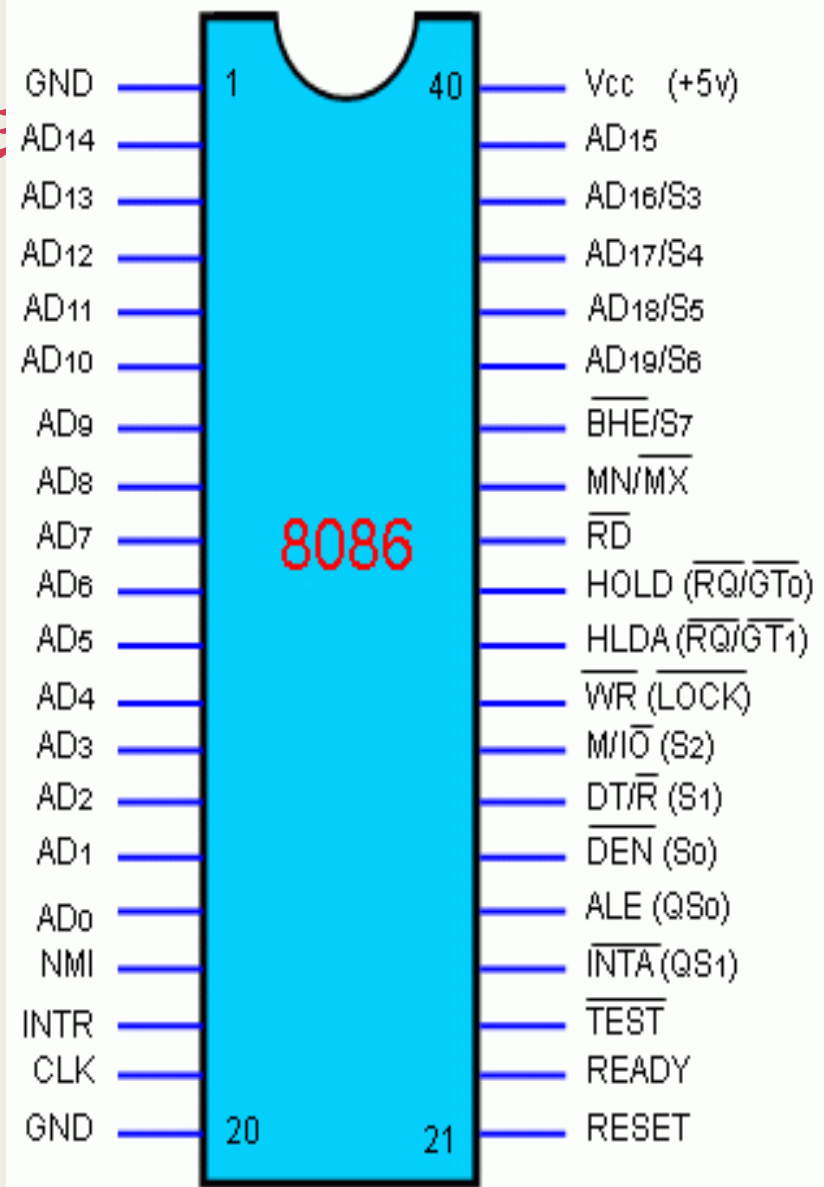
附加段寄存器ES：存放当前附加段的段地址；

有特定的现行段，彼此不能交换使用。

段寄存器的当前段：寻址空间中划出的64KB存储器块

2.1.2 8086/8088微处:

- ❖ 8086/8088 CPU具有40条引脚、双列直插式封装;
- ❖ 采用分时复用地址数据总线, 从而使8086/8088 CPU用40条引脚实现20位地址、16位数据、控制信号及状态信号的传输。



(a) 8086 CPU 封装外形

在理解和运用8086微处理器引脚时要注意：

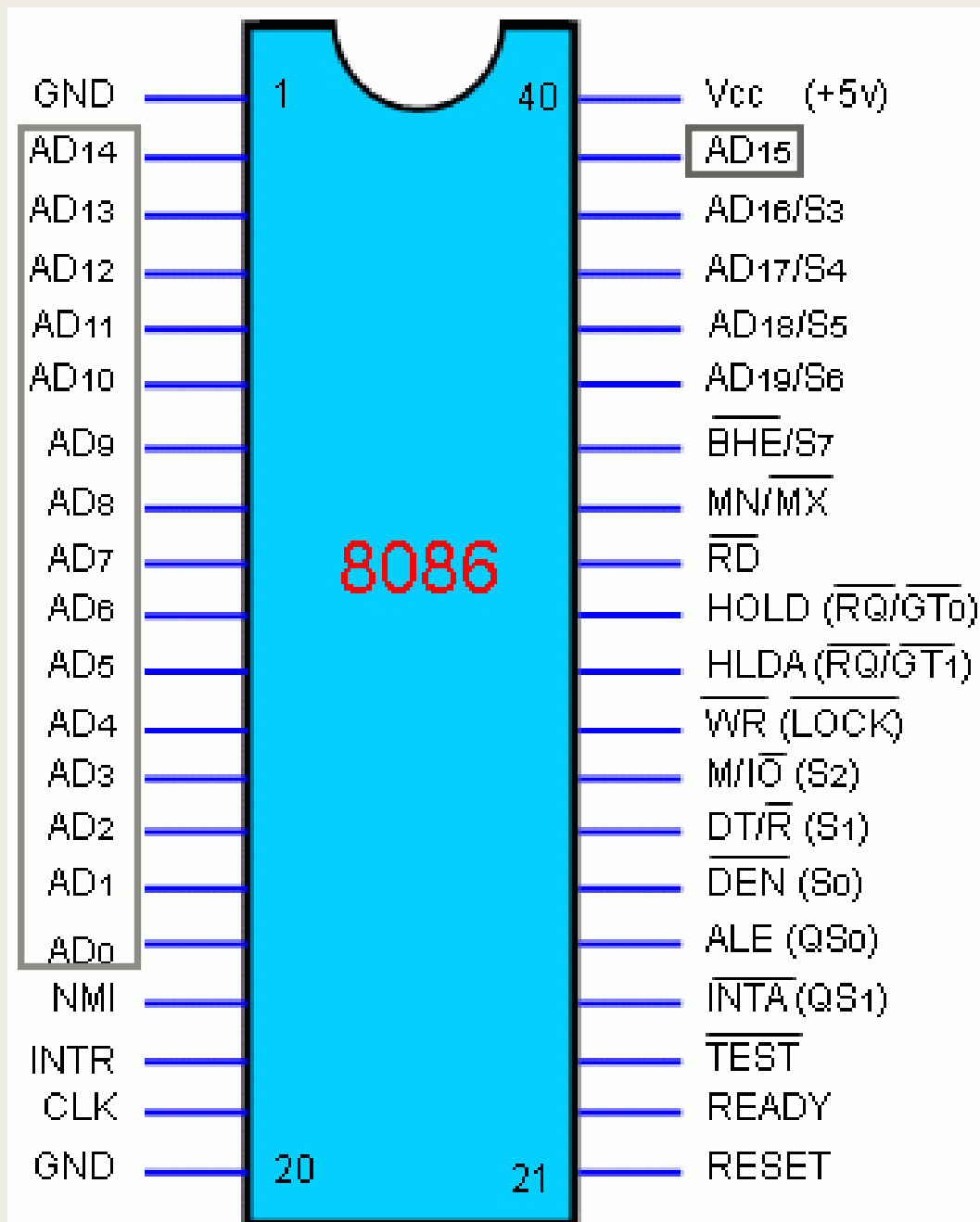
- 1) 每个引脚只传送一种特定信号；
- 2) 一个引脚电平的高低代表着不同的传递信号；
- 3) 当CPU工作于最大、最小不同模式时其引脚有着不同的名称和定义；
- 4) 地址和数据分时复用的引脚；
- 5) 某类特定引脚的输入和输出信号分别传送着不同的信息

5类

1) 地址/数据总线

$AD_{15} \sim AD_0$ (双向
传输信号、三态)

“三态”：是指除
“0” “1” 两种状态
外，还有一种悬浮（高
阻）状态，采用三态门
进行控制

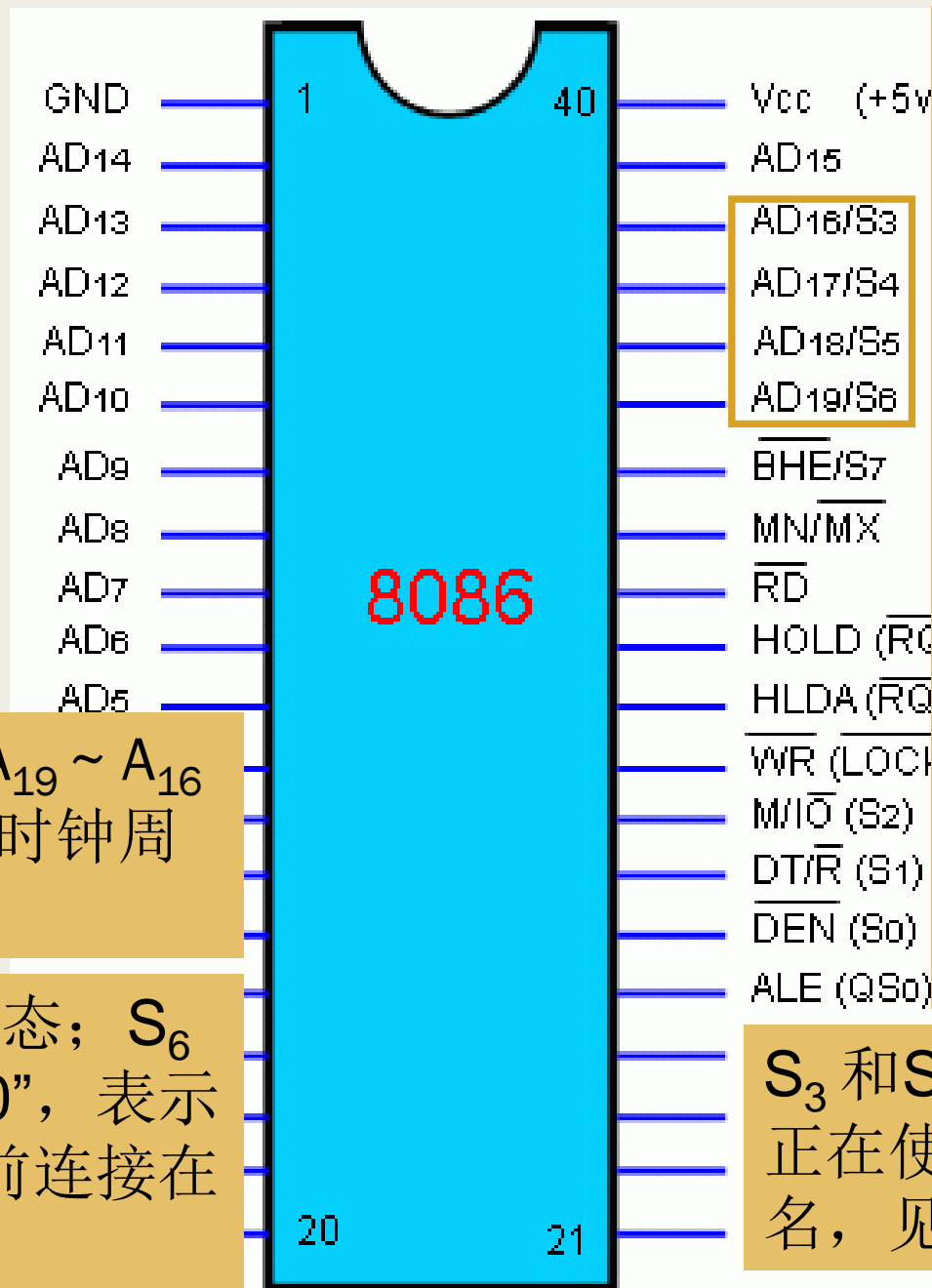


2) 地址/状态线

$A_{19}/S_6 \sim$
 A_{16}/S_3
(输出、三态)

访问I/O时不使用 $A_{19} \sim A_{16}$ 这4条线；在其他时钟周期输出状态信号。

S_5 表示IF的当前状态； S_6 始终输出低电平“0”，表示8086微处理器当前连接在总线上；



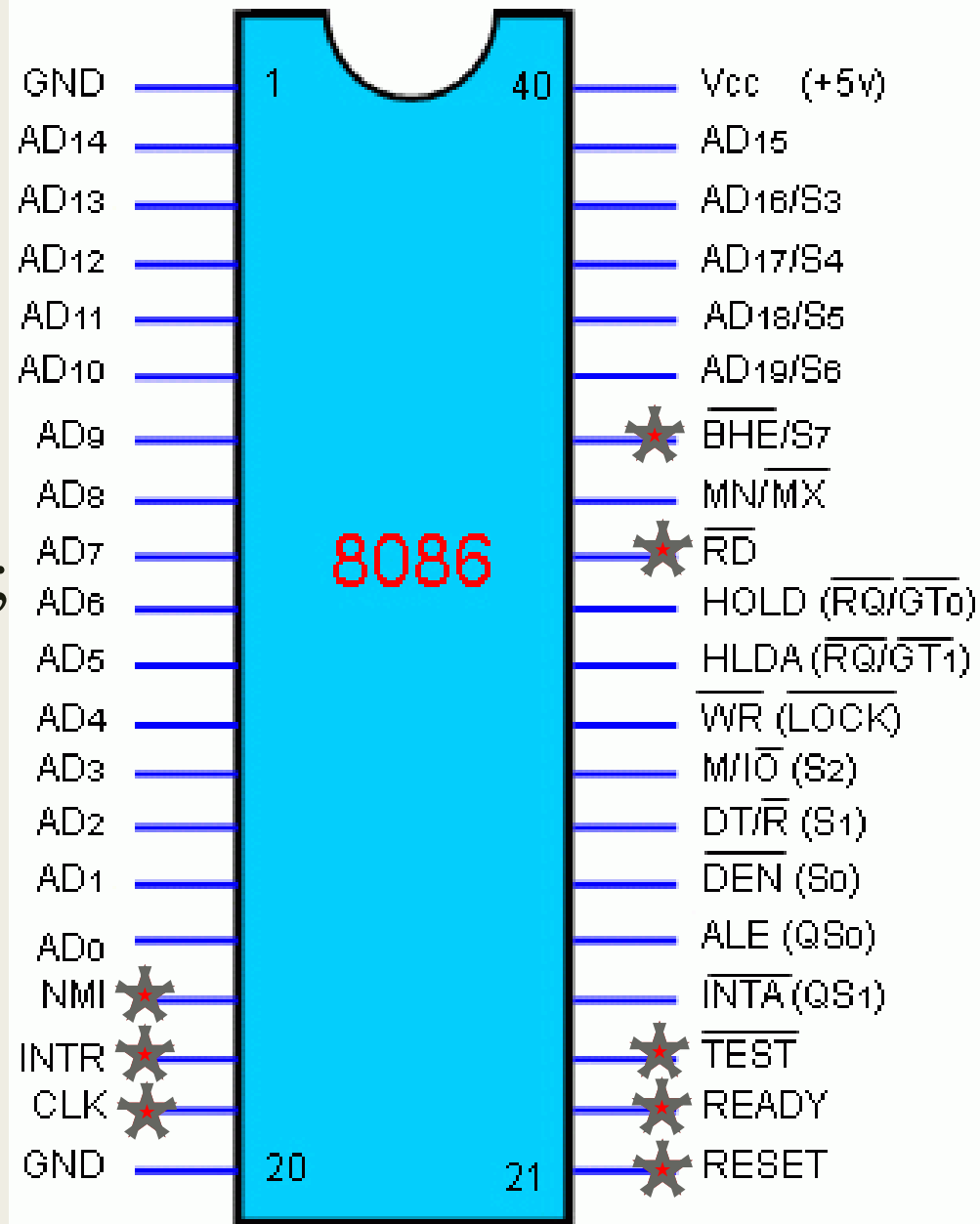
$S_6 \sim S_3$ 是状态信号，采用多路开关分时输出，在存储器操作的总周期第一个时钟周期输出20位地址的高4位 $A_{19} \sim A_{16}$ ，与 $AD_{15} \sim AD_0$ 组成20为地址信号

S_3 和 S_4 的组合表示正在使用的寄存器名，见表2.2

3) 控制总线

① 总线高字节允许/状态信号BHE/S7: BHE和地址线AD₀配合可用来产生存储体的选择信号;

② 读控制信号RD: 有效时表示CPU正在进行读存储器或读I/O端口的操作; 取决于M/IO信号;



4) 电源线 V_{CC} 和地线

GND

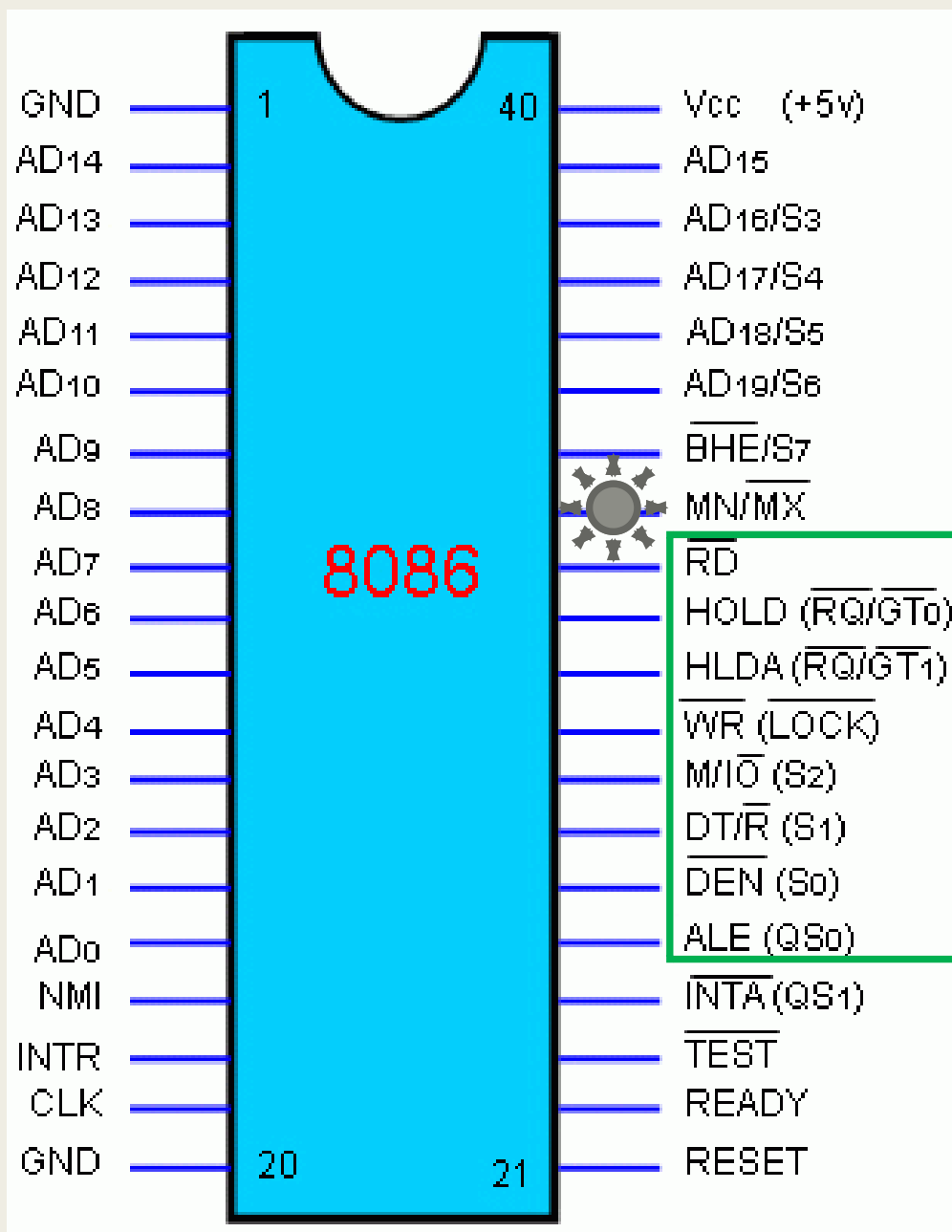
电压为 $+5 \times (1 \pm 10\%)V$,

两条地线GND均接地

5) 其他控制线: 24

~31引脚

这8条控制线的性能将根据8086处理器的最大/最小工作模式控制线MN/MX所处的工作状态而定。



8086/8088 CPU芯片可以在两种模式下工作，可通过第33条引脚MN/MX来控制：

- ❑ 最大模式（ $MN/MX = 0$ ）：8086 CPU的引脚MN/MX接地时，系统中含有两个或多个微处理器，其中一个主处理器就是8086/8088，另外的处理器称为协处理器。与8086匹配的①用于数值计算的协处理器8087，②用于输入/输出处理大量数据的协处理器8089；
- ❑ 最小模式（ $MN/MX = 1$ ）：8086 CPU的引脚MN/MX接+5V时，系统中只有一个微处理器，所有控制信号都是由自身提供，系统中的总线控制逻辑电路被减少到最少；
- ❑ 两种组成方式对CPU引脚24~31的定义不同。

2.1.3 总线与总线缓冲器

总线是为了将CPU内部的各部件连接成一个整体，实际上它是各部件传送信息的一组通信导线。

优点：

- 减少传输线的数目；
- 提高系统的可靠性；
- 增加系统的灵活性；
- 便于系统的标准化。

分为片内总线与片外总线：


1、片内总线

在微处理器内部各单元之间传送信息的总线

单总线、双总线、多总线

2、片外总线

系统总线一般是指片外总线，这种总线用来连接CPU与内存或I/O接口电路等部件。

- ✓ 地址总线
 - ✓ 数据总线
 - ✓ 控制总线
- 
- 三总线结构

均采用三态缓冲器总线电路。总线缓冲器的这种三态性，既保证了在任何时候，只允许此刻进行信息交换的设备占用总线，其他设备与总线“完全脱开，不会影响正常的信息传输，又为其他快速信息传输方式提供了必要的条件。

总线的三态性是现在所有的微处理器的共性。