第3章 8086/8088微处理器及其系统

在微处理器领域，Intel系列CPU产品一直占着主导地位。尽管8086/8088后续的80286、80386、80486以及Pentium系列CPU结构与功能已经发生很大的变化，但从基本概念与基本结构以及指令格式上来讲，它们仍然是经典的8086/8088 CPU的延续与提升。并且，其他系列流行的CPU（如AMD公司的6x86 MX/MⅡ等）也可以与80x86 CPU兼容。

本章着重介绍Intel 8086/8088微处理器及其指令系统；同时也简要介绍80286、80386、80486与Pentium系列微处理器的结构特点及其技术精髓。

3.1 8086/8088微处理器

8086是Intel系列的16位微处理器。在推出8086之后不久，Intel公司还推出了准16位微处理器8088。8088的内部寄存器、运算器以及内部数据总线与8086一样都是按16位设计的，但其外部数据总线只有8条。这样设计的目的主要是为了与Intel原有的8位外围接口芯片直接兼容。

3.1.1 8086/8088 CPU的内部结构

8086/8088 CPU的内部结构基本上是相似的，为了简化，在图3.1中只绘制了8086 CPU的内部功能结构框图。由图3.1可知，8086/8088 CPU内部可分为两个独立的功能单元，即总线接口单元（Bus Interface Unit，BIU）和执行单元（Execution Unit，EU）。

1. 总线接口单元

BIU是与总线连接的接口部件，其基本功能是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。在CPU取指令时，它从内存中取出指令送到指令队列缓冲器；而在执行指令时，它要与指定的内存单元或者I/O端口交换数据。

BIU内有4个16位段寄存器，即CS（代码段寄存器）、DS（数据段寄存器）、SS（堆栈段寄存器）和ES（附加段寄存器），16位指令指针IP，6字节指令队列缓冲器，20位地址加法器和总线控制电路。

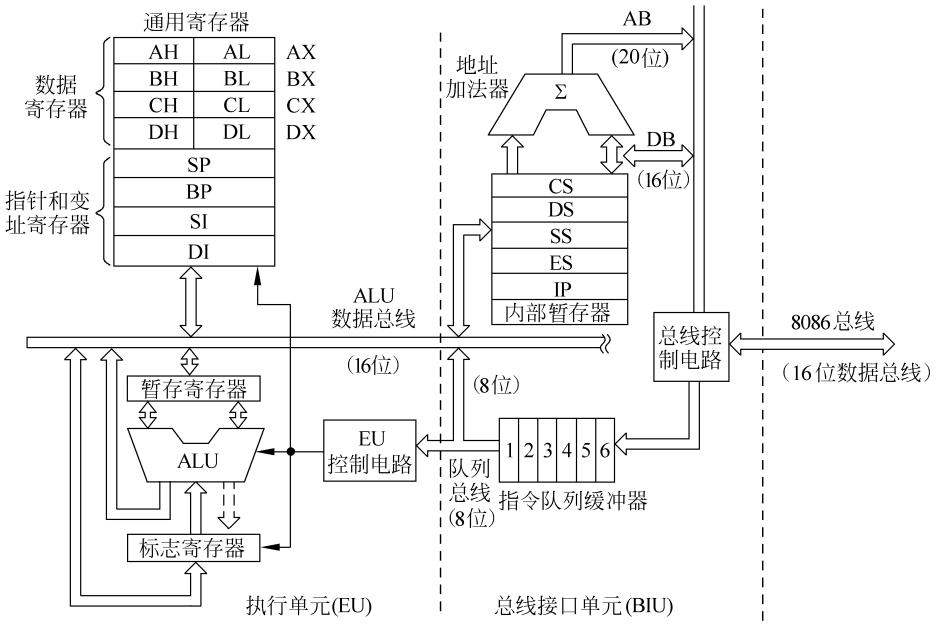


图 3.1　8086/8088 CPU 的内部功能结构框图

1) 指令队列缓冲器

8086 的指令队列由 6 个字节的寄存器组成,最多可存入 6 个字节的指令代码,而 8088 的指令队列只有 4 个字节。在 8086/8088 执行指令时,将从内存中取出 1 条或几条指令,依次放在指令队列中。它们采用"先进先出"的原则,按顺序存放,并按顺序取到 EU 中去执行。其操作将遵循下列原则。

(1) 取指令时,每当指令队列中存满 1 条指令后,EU 就立即开始执行。

(2) 当指令队列中空出 2 个(对 8086)或 1 个(对 8088)指令字节时,BIU 便自动执行取指操作,直到填满为止。

(3) EU 在执行指令的过程中,若 CPU 需要访问存储器或 I/O 端口,则 EU 自动请求 BIU 去完成访问操作。此时若 BIU 空闲,则会立即完成 EU 的请求;否则 BIU 首先将指令取至指令队列,再响应 EU 的请求。

(4) 当 EU 执行完转移、调用和返回指令时,则要清除指令队列缓冲器,并要求 BIU 从新的地址重新开始取指令,新取的第 1 条指令将直接经指令队列送到 EU 去执行,随后取来的指令将填入指令队列缓冲器。

2) 地址加法器和段寄存器

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? 这里采用了一种称为"段加偏移"的重要技术,即采用将可移位的 16 位段寄存器与 16 位偏移地址相加的办法,从而巧妙地解决了这一矛盾。具体地说,就是利用各段寄存器分别来存放确定各段的 20 位起始地址的高 16 位段地址信息,而由 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(又称为逻辑地址或偏移量),然后,将它与左移 4 位后的段寄存器的内容同时送到地址加法器进行相加,

最后形成一个 20 位的实际地址(又称为物理地址),以对存储单元寻址。图 3.2 所示为实际地址的产生过程。例如,要形成某指令码的实际地址,就要将 IP 的值与代码段寄存器(Code Segment,CS)左移 4 位后的内容相加。

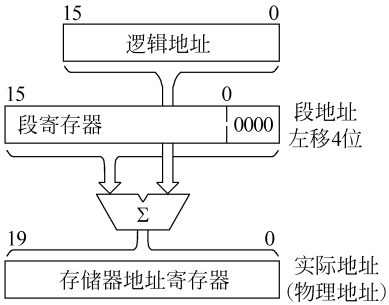


图 3.2 物理地址(实际地址)的产生过程

【例 3.1】 假设 CS=4000H,IP=0300H,则指令的物理地址 PA = 4000H × 16 + 0300H = 40300H。

3) 16 位指令指针

IP(Instruction Pointer)的功能与 8 位 CPU 中的 PC 类似。正常运行时,IP 中含有 BIU 要取的下一条指令(字节)的偏移地址。IP 在程序运行中能自动加 1 修正,使之指向要执行的下一条指令(字节)。有些指令(如转移、调用、中断和返回指令)能使 IP 值改变,或将 IP 值压进堆栈保存,或由堆栈弹出恢复原值。

2. 执行单元

EU 的功能是负责执行指令,执行的指令从 BIU 的指令队列中取得,执行指令的结果或执行指令所需要的数据,都由 EU 向 BIU 发出请求,再由 BIU 经总线控制电路对存储器或 I/O 端口存取。EU 由下列 5 个部分组成。

(1) 16 位算术逻辑单元(ALU):它可以用于进行算术、逻辑运算,也可以按指令的寻址方式计算出寻址单元的 16 位偏移量。

(2) 16 位标志寄存器 F:它用来反映 CPU 运算的状态特征或存放控制标志。

(3) 数据暂存寄存器:它协助 ALU 完成运算,暂存参加运算的数据。

(4) 通用寄存器组:它包括 4 个 16 位数据寄存器,即 AX、BX、CX、DX 和 4 个 16 位指针与变址寄存器,即 SP、BP 与 SI、DI。

(5) EU 控制电路:它是控制、定时与状态逻辑电路,接收从 BIU 中指令队列取来的指令,经过指令译码形成各种定时控制信号,对 EU 的各个部件实现特定的定时操作。

EU 中所有的寄存器和数据通道(除队列总线为 8 位外)都是 16 位的宽度,可实现数据的快速传送。

注意:由于 BIU 与 EU 分开独立设计,因此,在一般情况下,CPU 执行完一条指令后就可以立即执行下一条指令。16 位 CPU 这种并行重叠操作的特点,提高了总线的信息传输效率和整个系统的执行速度。

8088 CPU 的内部结构与 8086 的基本相似,只是 8088 的 BIU 中指令队列长度为 4 字节;8088 的 BIU 通过总线控制电路与外部交换数据的总线宽度是 8 位,总线控制电路与专用寄存器组之间的数据总线宽度也是 8 位。

3.1.2 8086/8088 的寄存器结构

对于微机应用系统的开发者来说,最重要的是掌握 CPU 的编程结构或程序设计模型。8086/8088 的内部寄存器编程结构如图 3.3 所示。它共有 13 个 16 位寄存器和 1 个

只用了 9 位的标志寄存器。其中,阴影部分与 8080/8085 CPU 相同。

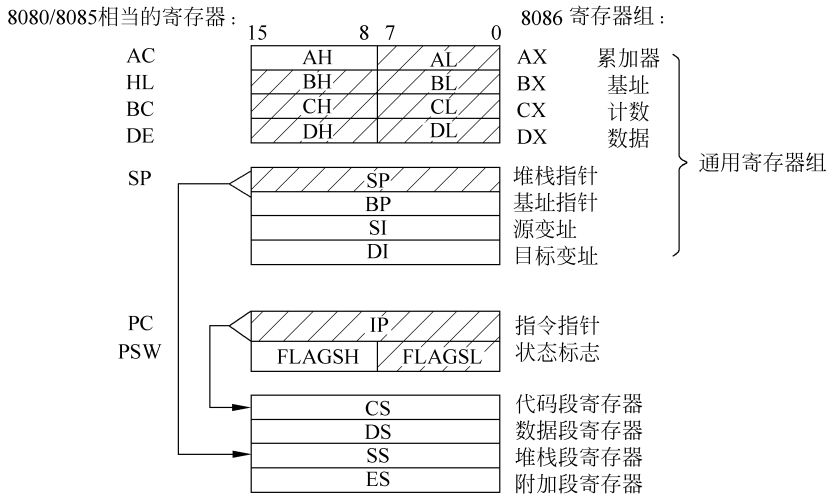


图 3.3　8086/8088 的编程结构

下面将根据寄存器的功能分别加以简要说明。

1. 通用寄存器

通用寄存器分为两组,即数据寄存器;指针寄存器和变址寄存器。

(1) 数据寄存器:执行单元(EU)中有 4 个 16 位数据寄存器,即 AX、BX、CX 和 DX。每个数据寄存器分为高字节 H 和低字节 L,它们均可作为 8 位数据寄存器独立寻址,独立使用。

在多数情况下,这些数据寄存器用在算术运算或逻辑运算指令中,用来进行算术逻辑运算。在有些指令中,它们则有特定的用途。这些寄存器在指令中的特定功能是被系统隐含使用的(如表 3.1 所示)。

表 3.1　数据寄存器的隐含使用

寄存器	操　作	寄存器	操　作
AX	字乘,字除,字 I/O	CL	多位移位和旋转
AL	字节乘,字节除,字节 I/O,转换,十进制运算	DX	字乘,字除,间接 I/O
AH	字节乘,字节除	SP	堆栈操作
BX	转换	SI	数据串操作
CX	数据串操作,循环	DI	数据串操作

(2) 指针寄存器和变址寄存器:指针寄存器是指堆栈指针寄存器(SP)和堆栈基址指针寄存器(BP),简称为 P 组。变址寄存器是指源变址寄存器(SI)和目的变址寄存器(DI),简称为 I 组。它们都是 16 位寄存器,一般用来存放偏移地址。

SP 和 BP 都用来指示存取位于当前堆栈段中的数据所在的地址,但 SP 和 BP 在使用上有区别。入栈(PUSH)和出栈(POP)指令是由 SP 给出栈顶的偏移地址,故称为堆栈指针寄存器。而 BP 则是存放位于堆栈段中的一个数据区基地址的偏移地址,故称为堆栈

基址指针寄存器。显然,由 SP 所指定的堆栈存储区的栈顶和由 BP 所指定的堆栈段中某一块数据区的首地址是两个不同的意思,不可混淆。

SI 和 DI 是存放当前数据段的偏移地址的。源操作数的偏移地址存放于 SI 中,所以 SI 称为源变址寄存器;目的操作数偏移地址存放于 DI 中,故 DI 称为目的变址寄存器。例如,在数据串操作指令中,被处理的数据串的偏移地址由 SI 给出,处理后的结果数据串的偏移地址则由 DI 给出。

2. 段寄存器

8086/8088 CPU 内部设计了 4 个 16 位的段寄存器,用这些段寄存器的内容作为段地址,再由段寄存器左移 4 位形成 20 位的段起始地址,它们通常被称为段基地址或段基址。再利用"段加偏移"技术,8086/8088 就有可能寻址 1MB 存储空间并将其分成为若干个逻辑段,使每个逻辑段的长度为 64KB(它由 16 位的偏移地址限定)。

注意:这些逻辑段可以通过修改段寄存器的内容被任意设置在整个 1MB 存储空间上下浮动。换句话说,逻辑段在存储器中定位以前,还不是微处理器可以真正寻址的实际内存地址,也正因为这样,通常人们就将未定位之前在程序中存在的地址叫做逻辑地址。这个概念对于后面将要讨论的程序"重定位"十分有用。

4 个 16 位段寄存器都可以被指令直接访问。其中,CS 用来存放程序当前使用的代码段的段地址,CPU 执行的指令将从代码段取得;SS 用来存放程序当前所使用的堆栈段的段地址,堆栈操作的数据就在堆栈段中;DS 用来存放程序当前使用的数据段的段地址,一般来说,程序所用的数据就存放在数据段中;ES 用来存放程序当前使用的附加段的段地址,也用来存放数据,但其典型用法是存放处理后的数据。

3. 标志寄存器

8086/8088 的 16 位标志寄存器 F 只用了其中的 9 位作为标志位,即 6 个状态标志位,3 个控制标志位。

如图 3.4 所示,低 8 位 FL 的 5 个标志与 8080/8085 的标志相同。

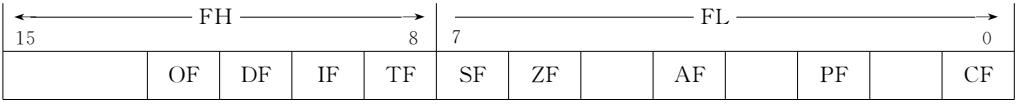


图 3.4　8086/8088 的标志寄存器

状态标志位用来反映算术或逻辑运算后结果的状态,以记录 CPU 的状态特征。下面介绍这 6 个标志位。

(1) CF(Carry Flag)进位标志:当执行一个加法或减法运算使最高位产生进位或借位时,则 CF 为 1;否则为 0。此外,循环指令也会影响它。

(2) PF(Parity Flag)奇偶性标志:当指令执行结果的低 8 位中含有偶数个"1"时,则 PF 为 1;否则为 0。此标志位用于微机中传送信息时,对产生的代码出错情况提供检测条件。此标志在现代程序设计中很少使用;现在,奇偶校验常常由数据通信设备完成,而不是由微处理器完成。

(3) AF(Auxiliary Carry Flag)辅助进位标志:当执行一个加法或减法运算使结果的

低字节的低 4 位向高 4 位(即 D_3 位向 D_4 位)进位或借位时,则 AF 为 1;否则为 0。DAA 和 DAS 指令测试这个特殊标志位,该标志一般用在 BCD 码运算中作为是否需要对 AL 寄存器进行十进制调整的依据。

(4) ZF(Zero Flag)零标志:零标志表示一个算术或逻辑操作的结果是否为 0。若当前的运算结果为 0,则 ZF 为 1;否则为 0。

(5) SF(Sign Flag)符号标志:符号标志保持算术或逻辑运算指令执行后结果的算术符号。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,正数的最高位为 0。

(6) OF(Overflow Flag)溢出标志:溢出标志用于判断在有符号数进行加法或减法运算时是否可能出现溢出。溢出将指示运算结果已超出机器能够表示的数值范围。当补码运算有溢出时,例如,用 8 位加法将 7FH(+127)加上 01H,结果为 80H(−128)。由于此结果已超出 8 位二进制补码所能表示的最大整数范围(+127),故此时 OF 标志为 1;否则为 0。

注意:对于无符号数的操作,将不考虑溢出标志。

控制标志位有 3 个,用来控制 CPU 的操作,由程序设置或清除。

(1) DF(Direction Flag)方向标志:它用来控制数据串操作指令的步进方向。若用 STD 指令将 DF 置 1,则数据串操作过程中地址会自动递减;若用 CLD 指令将 DF 清零,则数据串操作过程中地址会自动递增。地址的递增或递减由 DI 或 SI 变址寄存器来实现。

(2) IF(Interrupt Enable Flag)中断允许标志:它是控制可屏蔽中断的标志。若用 STI 指令将 IF 置 1,则表示允许 8086/8088 CPU 接收外部从其 INTR 引脚上发来的可屏蔽中断请求信号;若用 CLI 指令将 IF 清零,则禁止 CPU 接收外来的可屏蔽中断请求信号。IF 的状态不影响非屏蔽中断(NMI)请求,也不影响 CPU 响应内部的中断请求。

(3) TF(Trap Flag)跟踪(陷阱)标志:它是为调试程序方便而设置的。若将 TF 标志置为 1,则 CPU 处于单步工作方式;否则,将正常执行程序。

注意:在高型号微处理器中,跟踪(陷阱)标志能够激活芯片上的调试特性(调试程序,以便找到错误或故障),当 TF 标志为 1 时,则微处理器将根据调试寄存器和控制寄存器的指示中断程序流。

最后需要指出的是,8086/8088 所有上述标志位对 Intel 系列后续高型号微处理器的标志寄存器都是兼容的,只不过后者有些增强功能或者新增加了一些标志位而已。

3.1.3 总线周期

总线周期是微处理器操作时所依据的一个基准时间段,通常,它是指微处理器完成一次访问存储器或 I/O 端口操作所需的时间。

对于 8086/8088 CPU 来说,总线周期由 4 个时钟周期组成,这 4 个时钟周期也称为 T_1、T_2、T_3 与 T_4 四个状态;在每一个状态中,CPU 在操作时,总线所处的状态都不同。一般在 T_1 状态,CPU 往多路复用总线上发送寻址的地址信息,以选中某个被寻址的存储器

单元或端口地址；在 T_2 状态，CPU 从总线上撤销地址，为传送数据做准备；在 T_3 状态，多路总线的高 4 位继续提供状态信息，而其低 16 位（对 8086 CPU）或低 8 位（对 8088 CPU）上将出现由 CPU 读入或写出的数据；在 T_4 状态，CPU 采样数据总线，完成本次读或写操作，最后结束总线周期。

注意：不同的 CPU 在一个总线周期内所处的总线状态是不同的；而即使同一个 CPU，其读操作或写操作的具体状态也不相同。一般，在 $T_2 \sim T_4$，若是写操作，则 CPU 在此期间是先把输出数据送到总线上；若是读操作，则 CPU 在 $T_3 \sim T_4$ 期间将从总线上输入数据。T_2 时复用地址数据总线处于悬空状态，以便使 CPU 有一个缓冲时间把输出地址的写操作转换为输入数据的读操作。

此外，如果存储器或外设的速度较慢，不能及时地跟上 CPU 的速度时，存储器或外设就会通过 READY 信号线在 T_3 状态启动之前向 CPU 发一个“数据未准备好”信号，并且，CPU 会在 T_3 之后自动插入一个或多个等待状态 T_W，以等待存储器或外设准备好传送数据。只有在存储器或外设准备就绪时，它们才又通过 READY 的信号线向 CPU 发出一个有效的“准备好”信号，CPU 接收到这一信号后，才会自动脱离 T_W 状态而进入 T_4 状态。

总线周期只用于 CPU 取指和它同存储器或 I/O 端口交换数据；否则，总线接口单元 BIU 将不和总线“打交道”，即系统总线处于空闲状态，即执行空闲周期，这时，虽然 CPU 对总线进行空操作，但 CPU 内部的执行单元 EU 仍在进行操作，如逻辑运算单元 ALU 仍在进行运算，内部寄存器之间也在传送数据。

图 3.5 所示为一个典型的总线周期序列。

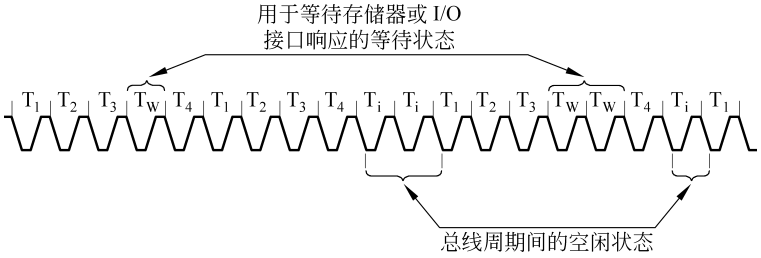


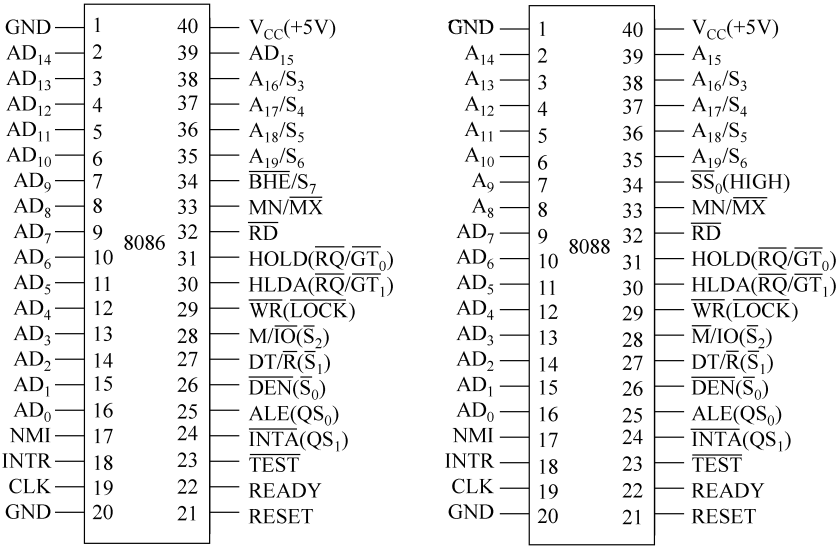
图 3.5　一个典型的总线周期序列

3.1.4　8086/8088 的引脚信号和功能

图 3.6 所示为 8086 和 8088 的引脚信号图。它们的 40 条引线按功能可分为以下 5 类。

1. 地址/数据总线

$AD_{15} \sim AD_0$ 是分时复用的存储器或端口的地址和数据总线。传送地址时为单向的三态输出，而传送数据时可为双向三态输入/输出。8086/8088 CPU 正是利用分时复用的方法才能使其用 40 条引脚实现 20 位地址、16 位数据及众多控制信号和状态信号的传输。在 8088 中，由于只能传输 8 位数据，因此，只有 $AD_7 \sim AD_0$ 八条地址/数据线，$A_{15} \sim A_8$ 只用来输出地址。



(a) 8086的引脚信号　　(b) 8088的引脚信号

图 3.6　8086/8088 的引脚信号(括号中为最大方式时的引脚名称)

作为复用引脚,在总线周期的 T_1 状态用来输出要寻址的存储器或 I/O 端口地址;在 T_2 状态浮置成高阻状态,为传输数据作准备;在 T_3 状态,用于传输数据;T_4 状态结束总线周期。当 CPU 响应中断以及与系统总线"保持响应"时,复用线都被浮置为高阻状态。

2. 地址/状态总线

地址/状态总线 $A_{19}/S_6 \sim A_{16}/S_3$ 为输出、三态总线,采用分时输出,即 T_1 状态输出地址的最高 4 位,$T_2 \sim T_4$ 状态输出状态信息。当访问存储器时,T_1 状态时输出的 $A_{19} \sim A_{16}$ 送到锁存器(8282)锁存,与 $AD_{15} \sim AD_0$ 组成 20 位的地址信号;而访问 I/O 端口时,不使用这 4 条引线,即 $A_{19} \sim A_{16}=0$。状态信息中的 S_6 为 0 用来指示 8086/8088 当前与总线相连,所以,在 $T_2 \sim T_4$ 状态,S_6 总等于 0,以表示 8086/8088 当前连在总线上。S_5 表明中断允许标志位 IF 的当前设置。S_4 和 S_3 用来指示当前正在使用哪个段寄存器,如表 3.2 所示。

表 3.2　S_4、S_3 的代码组合和对应的状态

S_4	S_3	状　　态
0	0	当前正在使用 ES
0	1	当前正在使用 SS
1	0	当前正在使用 CS 或未用任何段寄存器
1	1	当前正在使用 DS

当系统总线处于"保持响应"状态时,这些引线被浮置为高阻状态。

3. 控制总线

(1) $\overline{BHE}/S_7$:高 8 位数据总线允许/状态复用引脚,三态、输出。$\overline{BHE}$在总线周期的 T_1 状态时输出,S_7 在 $T_2 \sim T_4$ 时输出。在 8086 中,当$\overline{BHE}/S_7$ 引脚上输出$\overline{BHE}$信号时,

表示总线高 8 位 $AD_{15}\sim AD_8$ 上的数据有效。在 8088 中，第 34 引脚不是$\overline{BHE}/S_7$，而是被赋予另外的信号：在最小方式时，它为$\overline{SS_0}$，和 DT/$\overline{R}$、$\overline{M}$/IO 一起决定了 8088 当前总线周期的读/写动作；在最大方式时，它恒为高电平。S_7 在当前的 8086 芯片设计中未被赋予定义，暂作备用状态信号线。

(2) $\overline{RD}$：读控制信号，三态、输出。当$\overline{RD}=0$ 时，表示 CPU 执行存储器或 I/O 端口的读操作。是对内存单元还是对 I/O 端口读取数据，取决于 M/$\overline{IO}$(8086)或 $\overline{M}$/IO(8088)信号。在执行 DMA 操作时，$\overline{RD}$被浮空。

(3) READY："准备好"信号线，输入。该引脚接收被寻址的内存或 I/O 端口发给 CPU 的响应信号，高电平时表示内存或I/O 端口已准备就绪，CPU 可以进行数据传输。CPU 在 T_3 状态开始对 READY 信号采样。若检测到 READY 为低电平，表示内存或 I/O 端口尚未准备就绪，则 CPU 在 T_3 状态之后自动插入等待状态 T_W，直到 READY 变为高电平，内存或 I/O 端口已准备就绪，CPU 才可以进行数据传送。

(4) $\overline{TEST}$：等待测试输入信号，低电平有效。它用于多处理器系统中且只有在执行 WAIT 指令时才使用。当 CPU 执行 WAIT 指令时，它就进入空转的等待状态，并且每隔 5 个时钟周期对该线的输入进行一次测试；若$\overline{TEST}=1$，则 CPU 将停止取下一条指令而继续处于等待状态，重复执行 WAIT 指令，直至$\overline{TEST}=0$ 时，CPU 才结束 WAIT 指令的等待状态，继续执行下一条指令。等待期间允许外部中断。

(5) INTR：可屏蔽中断请求输入信号，高电平有效。它为高电平时，表示外设有中断请求，CPU 在每个指令周期的最后一个 T 状态采样此信号。若 IF=1，则 CPU 响应中断，并转去执行中断服务程序。若 IF=0(关中断)，则外设的中断请求被屏蔽，CPU 将不响应中断。

(6) NMI：非屏蔽中断请求输入信号，上升沿触发。此信号不受 IF 状态的影响，只要它一出现，CPU 就会在现行指令结束后引起中断。

(7) RESET：复位输入信号，高电平有效。通常，它与 8284A(时钟发生/驱动器)的复位输出端相连，8086/8088 要求复位脉冲宽度不得小于 4 个时钟周期，而初次接通电源时所引起的复位，则要求维持的高电平不能小于 50μs；复位后，CPU 的主程序流程恢复到启动时的循环待命初始状态，其内部寄存器状态如表 3.3 所示。在程序执行时，RESET 线保持低电平。

表 3.3　复位后内部寄存器的状态

内部寄存器	状　态	内部寄存器	状　态
标志寄存器	清除	SS	0000H
IP	0000H	ES	0000H
CS	FFFFH	指令队列缓冲器	清除
DS	0000H		

(8) CLK：系统时钟，输入。通常与 8284A 时钟发生器的时钟输出端 CLK 相连，该时钟信号的低/高之比常采用 2∶1(占空度为 1/3)。

4. 电源线和地线

电源线 V_{CC} 接入的电压为+5V±10%，有两条地线 GND，均应接地。

5. 其他控制线

这些控制线(24～31 引脚)的功能将根据方式控制线 MN/$\overline{MX}$所处的状态而确定。关于这些引脚在最小方式与最大方式下的具体功能差异,将在 3.2 节中给予详细说明。

由上述可知,8086/8088 CPU 引脚的主要特点是:数据总线和地址总线的低 16 位 AD_{15}～AD_0 或低 8 位 AD_7～AD_0 采用分时复用技术。还有一些引脚也具有两种功能,这由引脚 33(MN/$\overline{MX}$)来控制。当 MN/$\overline{MX}$=1 时,8086/8088 工作于最小方式(MN),在此方式下,全部控制信号由 CPU 本身提供。当 MN/$\overline{MX}$=0 时,8086/8088 工作于最大方式。这时,系统的控制信号由 8288 总线控制器提供,而不是由 8086/8088 直接提供。

3.2 8086/8088 系统的最小/最大工作方式

由 8086/8088 CPU 构成的微机系统,有最小方式和最大方式两种系统配置方式。

3.2.1 最小方式

8086 与 8088 构成的最小方式系统区别甚小,现以 8086 最小方式系统为例加以说明。

当 MN/$\overline{MX}$接电源电压时,系统工作于最小方式,即单处理器系统方式,它适合于较小规模的应用。8086 最小方式典型的系统结构如图 3.7 所示。它和 8 位微处理器系统类似,系统芯片可根据用户需要接入。图 3.7 中的 8284A 为时钟发生/驱动器,外接晶体

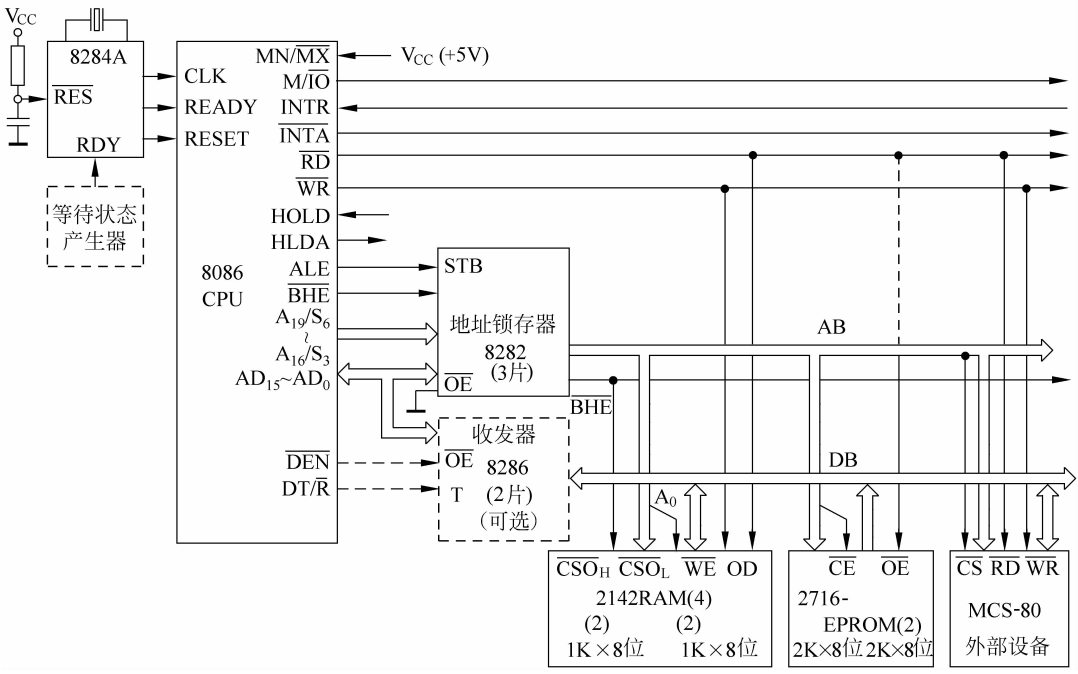


图 3.7 8086 最小方式的典型系统结构示意图

的基本振荡频率为15MHz，经8284A三分频后，送给CPU作系统时钟。8282为8位地址锁存器，当8086访问存储器时，在总线周期的T_1状态下发出地址信号，经8282锁存后的地址信号可以在访问存储器操作期间保持不变，为外部提供稳定的地址信号。由于8282是8位的锁存器芯片，而8086/8088有20位地址，加上还有$\overline{BHE}$与ALE信号连到8282片上，因此需要采用3片8282地址锁存器才能满足系统总线连接的需要。8286为具有三态输出的8位数据总线收发器，用于需要增加驱动能力的系统。在8086系统中，要用2片8286，而在8088系统中，只用1片8286即可。2142为1KB×4位的静态RAM，2716 EPROM为2KB×8位的可编程序只读存储器。8086/8088有20位地址信号线A_{19}～A_0，组成系统时将根据所使用的存储器的实际地址进行选用。

系统中还有一个等待状态产生电路，它为8284A的RDY端提供一个信号，经8284A同步后向CPU的READY线发送准备就绪信号，通知CPU数据传送已经完成，可以退出当前的总线周期。当READY=0时，CPU在T_3之后会自动插入T_W状态，以避免CPU与存储器或I/O设备进行数据交换时，因后者速度慢来不及完成读/写操作而丢失数据。

在最小方式下，第24～31脚的信号含义如下所示。

(1) $\overline{INTA}$中断响应信号(输出，低电平有效)：它表示8086/8088 CPU对外设中断请求INTR作出的响应信号。$\overline{INTA}$信号被设计为在相邻的两个总线周期中送出两个连续的负脉冲，第1个负脉冲是通知外设端口，它发出的中断请求已获允许；而在第2个负脉冲期间，外设端口(例如中断控制器)将往数据总线上发送一个中断类型码n，使CPU可以得到有关此中断的相应信息。

(2) ALE地址锁存信号(输出，高电平有效)：它是8086/8088 CPU提供给地址锁存器8282/8283的控制信号。ALE在T_1状态为高电平时，表示当前在地址/数据复用总线上输出的是有效地址，由地址锁存器把它作为锁存控制信号而将地址锁存其中。

注意：ALE信号总是接到锁存器的STB端。

(3) $\overline{DEN}$数据允许信号(输出，低电平有效，三态)：这是CPU提供给8286/8287数据总线收发器的三态控制信号，接到$\overline{OE}$端。该信号决定了是否允许数据通过数据总线收发器：当$\overline{DEN}$(即$\overline{OE}$)=1时，禁止收发器在收或发两个方向上传送数据；当$\overline{DEN}$(即$\overline{OE}$)=0时，才允许收发器传送数据。因此，总线收发器将$\overline{DEN}$作为数据收发的允许信号。在DMA方式时，$\overline{DEN}$被置为浮空。

(4) DT/$\overline{R}$数据收发信号(输出，三态)：它用于控制8286/8287的数据传送方向。当DT/$\overline{R}$=1时，表示CPU通过收发器发送数据；当DT/$\overline{R}$=0时，表示接收数据。在DMA方式时，它被置为浮空。

(5) M/$\overline{IO}$存储器/输入、输出控制信号(输出，三态)：它用于区分CPU当前是访问存储器还是访问输入/输出端口。高电平表示访问存储器，低电平表示访问输入/输出设备。在DMA方式时，它被置为浮空。

注意：8088 CPU的此引脚为$\overline{M}$/IO。

(6) $\overline{WR}$写信号(输出，低电平有效，三态)：当$\overline{WR}$=0时，表示CPU正在执行存储器或I/O写操作。在写周期中，$\overline{WR}$在T_2、T_3、T_W期间都有效。在DMA方式时，$\overline{WR}$被置为浮空。

(7) HOLD 总线保持请求信号(输入,高电平有效):它是系统中其他处理器部件(如 DMA——直接存储器存取控制器)用于向 CPU 发出要求占用总线的一个请求信号。当它为高电平时,表示其他处理器部件申请总线"保持"(即"持有"或"占有"),若 CPU 允许让出总线,则在当前总线周期的 T_4 状态从 HLDA 引脚发出应答信号,并暂停正常操作而放弃对总线的控制权。于是,其他处理器部件便获得对总线的控制权,以便完成所需要的操作(如 DMA 传送)。

(8) HLDA 总线保持响应信号(输出,高电平有效):当 HLDA 为有效电平时,表示 CPU 对其他处理部件的总线"保持"请求作出响应并正处于响应的状态,与此同时,所有带三态门的 CPU 引脚都置为浮空,从而让出总线。当其他处理器部件完成操作(如 DMA 传送)时,"保持"申请结束,CPU 便转向去执行下一个总线周期的操作。

3.2.2 最大方式

8086 与 8088 也都可以按最大方式来配置系统。当 MN/$\overline{MX}$线接地时,则系统工作于最大方式。图 3.8 所示为 8086 最大方式的典型系统结构。从图中可以看到,最大方式系统与最小方式系统的主要区别是:前者是一个多处理系统,必须外加有 8288 总线控制器,通过它对 CPU 发出的状态信息 $\overline{S}_2$、$\overline{S}_1$、$\overline{S}_0$ 进行不同的编码组合,就可以产生在最大方式系统中存储器和 I/O 端口以及锁存器 8282 与总线收发器 8286 所需要的多种控制信号。同时,在最大方式系统中,由于一般包含两个或多个处理器,这样就要解决主处理器和协处理器之间的通信以及对总线的争用共享问题。为此,在最大方式系统中配置了 8288 总线控制器,它使总线的控制功能更加完善。

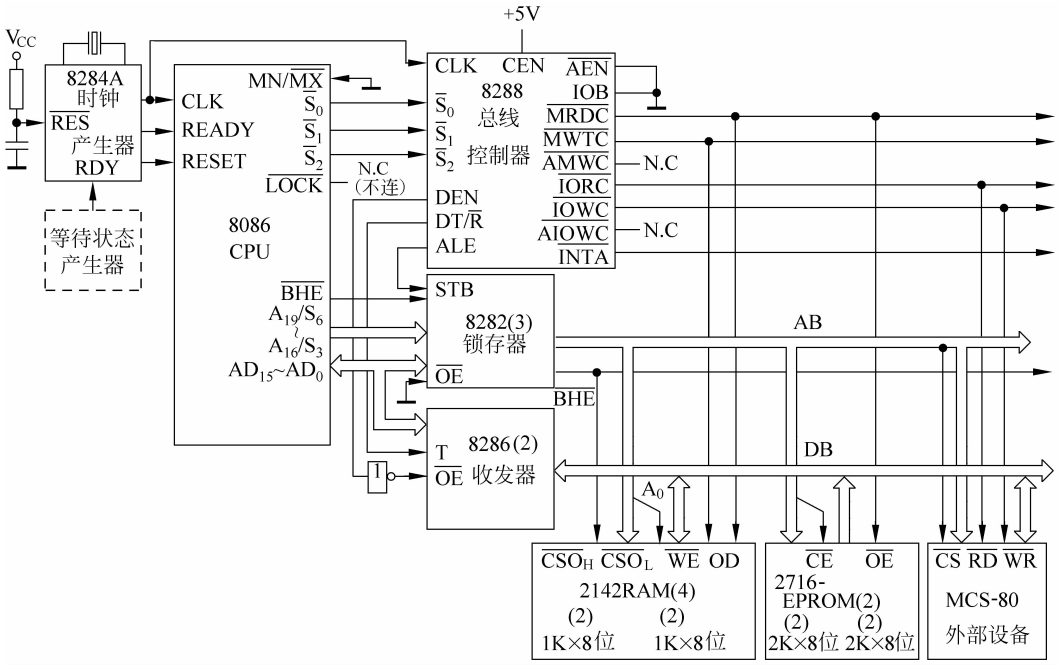


图 3.8 8086 最大方式的典型系统结构示意图

通过比较两种工作方式可以知道，在最小方式系统中，控制信号 M/$\overline{IO}$（或 $\overline{M}$/IO）、$\overline{WR}$、$\overline{INTA}$、ALE、DT/$\overline{R}$ 和$\overline{DEN}$是直接从 CPU 的第 24～29 脚送出的；而在最大方式系统中，则由状态信号 $\overline{S}_2$、$\overline{S}_1$、$\overline{S}_0$ 隐含了上面这些信息，使用 8288 后，系统就可以从 $\overline{S}_2$、$\overline{S}_1$、$\overline{S}_0$ 状态信息的组合中得到与这些控制信号功能相同的信息。$\overline{S}_2$、$\overline{S}_1$、$\overline{S}_0$ 和系统在当前总线周期中具体的操作过程之间的对应关系如表 3.4 所示。

表 3.4　$\overline{S}_2$、$\overline{S}_1$、$\overline{S}_0$ 的代码组合和对应的操作

$\overline{S}_2$	$\overline{S}_1$	$\overline{S}_0$	8288 产生的控制信号	操 作 状 态
0	0	0	$\overline{INTA}$	发中断响应信号
0	0	1	$\overline{IORC}$	读 I/O 端口
0	1	0	$\overline{IOWC}$、$\overline{AIOWC}$	写 I/O 端口
0	1	1	无	暂停
1	0	0	$\overline{MRDC}$	取指令
1	0	1	$\overline{MRDC}$	读内存
1	1	0	$\overline{MWTC}$、$\overline{AMWC}$	写内存
1	1	1	无	无源状态(CPU 无作用)

在表 3.4 中，前 7 种代码组合都对应了某一个总线操作过程，通常称为有源状态，它们处于前一个总线周期的 T_4 状态或本总线周期的 T_1、T_2 状态中，$\overline{S}_2$、$\overline{S}_1$、$\overline{S}_0$ 至少有一个信号为低电平。在总线周期的 T_3、T_W 状态并且 READY 信号为高电平时，$\overline{S}_2$、$\overline{S}_1$、$\overline{S}_0$ 都为高电平，此时，前一个总线操作过程就要结束，后一个新的总线周期尚未开始，通常称为无源状态。而在总线周期的最后一个状态即 T_4 状态，$\overline{S}_2$、$\overline{S}_1$、$\overline{S}_0$ 中任何一个或几个信号的改变，都意味着下一个新总线周期的开始。

此外，还有几个在最大方式下使用的专用引脚，其含义简要解释如下所示。

（1）QS_1、QS_0 指令队列状态信号（输出）：这两个信号的组合编码反映了 CPU 内部当前的指令队列状态，以便外部逻辑监视内部指令队列的执行过程。QS_1、QS_0 编码及其对应的含义如表 3.5 所示。

表 3.5　QS_1、QS_0 编码及其对应的含义

QS_1	QS_0	含　义
0	0	未从指令队列中取指令
0	1	从指令队列中取走第 1 个字节指令代码
1	0	指令队列已取空
1	1	从指令队列中取走后续字节指令代码

（2）$\overline{LOCK}$总线封锁信号（输出，低电平有效，三态）：当$\overline{LOCK}$输出低电平时，表示 CPU 独占对总线的主控权，并封锁系统中的其他总线主部件占用总线，这样，可以避免系统中多个处理主部件同时使用共享资源（如同时要求访问“内存”资源）而引起的冲突。

$\overline{LOCK}$信号由指令前缀 LOCK 产生，其有效时间是从 CPU 执行 LOCK 指令前缀开始直到下一条指令结束。在两个中断响应$\overline{INTA}$负脉冲期间也有效。在 DMA 时，$\overline{LOCK}$端被置为浮空。

(3) $\overline{RQ}/\overline{GT}_1$、$\overline{RQ}/\overline{GT}_0$ 总线请求信号输入/总线请求允许信号(双向,输出,低电平有效):在多处理系统中,当 8086/8088 CPU 以外的两个协处理器(如 8087 或 8089)需要占用总线时,就会用该信号线输出低电平表示要求占用总线;当 CPU 检测到有请求信号且总线处于允许状态时,则 CPU 的$\overline{RQ}/\overline{GT}$线输出低电平作为允许信号,再经协处理器检测出此允许信号后,便对总线进行占用。协处理器使用总线时,其输出的$\overline{RQ}/\overline{GT}$为高电平;待使用完毕,协处理器将$\overline{RQ}/\overline{GT}$线由高电平变为低电平(释放);当 CPU 检测到该释放信号后,又恢复对总线的主控权。$\overline{RQ}/\overline{GT}_1$ 和$\overline{RQ}/\overline{GT}_0$ 都是双向的,请求信号和允许信号在同一引线上传输,但方向相反。若总线信号同时出现在这两个引脚上时,$\overline{RQ}/\overline{GT}_0$ 的优先级高于$\overline{RQ}/\overline{GT}_1$。

在 8288 芯片上,还有几条控制信号线,如$\overline{MRDC}$(Memory Read Command)、$\overline{MWTC}$(Memory Write Command)、$\overline{IORC}$(I/O Read Command)、$\overline{IOWC}$(I/O Write Command)与$\overline{INTA}$等,它们分别是存储器与 I/O 的读/写命令以及中断响应信号。另外,还有$\overline{AMWC}$与$\overline{AIOWC}$两个输出信号,它们分别表示提前的写内存命令与提前的写 I/O 命令,其功能分别和$\overline{MWTC}$与$\overline{IOWC}$一样,只是它们由 8288 提前一个时钟周期发出信号,这样,一些较慢的存储器和外设将得到一个额外的时钟周期去执行写入操作。在使用 8288 时,连接在总线上的装置一般都用$\overline{MWTC}$和$\overline{IOWC}$,或者用$\overline{AMWC}$和$\overline{AIOWC}$,但不会同时使用这 4 种信号。另外,所有三态输出类型的控制线都可以被禁止,从而使它们均可以与系统总线断开。

3.3 8086/8088 的存储器

3.3.1 存储器组织

8086/8088 有 20 条地址线,可寻址 1MB 的存储空间。存储器仍按字节组织,每个字节只有唯一的一个地址。若存放的信息是 8 位的字节,则将按顺序存放;若存放的数为 1 个字,则将字的低位字节放在低地址中,高位字节放在高地址中;当存放的是双字形式(这种数一般作为指针),其低位字是被寻址地址的偏移量;高位字是被寻址地址所在的段地址。指令和数据(包括字节或字)在存储器中的存放位置如图 3.9 所示。对存放的字,其低位字节可以在奇数地址中开始存放,也可以在偶数地址中开始存放;前者称为非规则存放,这样存放的字称为非规则字;后者称为规则存放,这样存放的字称为规则字。对规则字的存取可在一个总线周期完成,非规则字的存取则需两个总线周期。这就是说,读或写一个以偶数为起始地

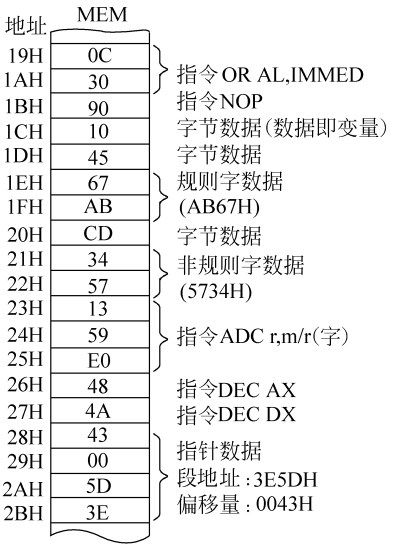


图 3.9 指令、数据在存储器中的存放位置

址的字的指令，只需访问一次存储器；而对于一个以奇数为起始地址的字的指令，就必须两次访问存储器中的两个偶数地址的字，忽略每个字中所不需要的那半个字，并对所需的两个半字进行字节调整。各种字节和字的读操作的例子如图 3.10 所示。

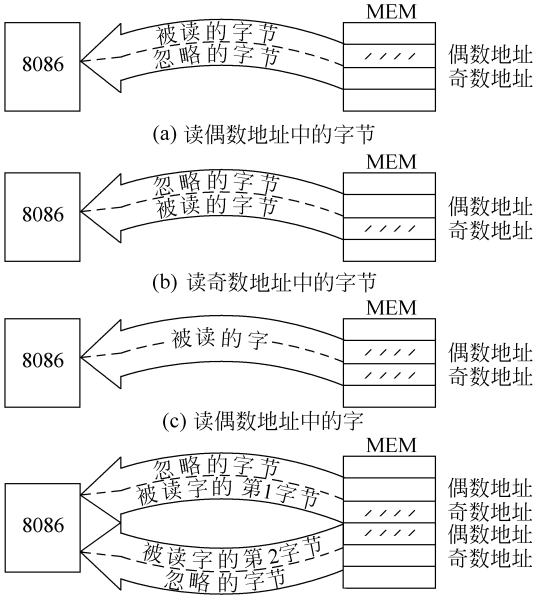


图 3.10　从 8086 存储器的偶数和奇数地址读字节和字

在 8086/8088 程序中，指令仅要求指出对某个字节或字进行访问，而对存储器访问的方式不必说明。无论执行哪种访问，都是由处理器自动识别的。

8086 的 1MB 存储空间实际上分为两个 512KB 的存储体，又称存储库，分别叫高位库和低位库，低位库与数据总线 $D_7 \sim D_0$ 相连，该库中每个地址为偶数地址；高位库与数据总线 $D_{15} \sim D_8$ 相连，该库中每个地址为奇数地址。地址总线 $A_{19} \sim A_1$ 可同时对高、低位库的存储单元寻址，A_0 或$\overline{BHE}$则用于库的选择，分别接到库选择端$\overline{SEL}$上，如图 3.11 所示。当 $A_0=0$，选择偶数地址的低位库；当$\overline{BHE}=0$ 时，选择奇数地址的高位库。利用 A_0

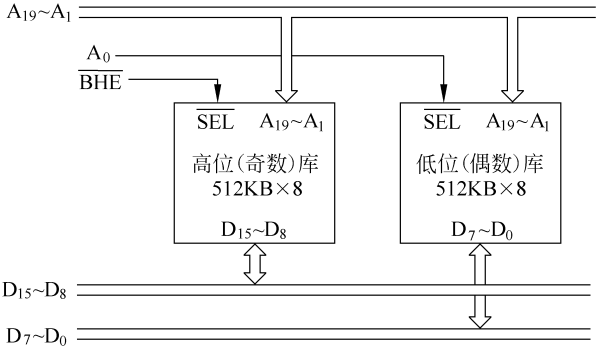


图 3.11　8086 存储器高、低位库的连接

或$\overline{BHE}$这两个控制信号可以实现对两个库进行读/写(即 16 位数据)操作,也可单独对其中的一个库进行读/写操作(即 8 位数据),如表 3.6 所示。

表 3.6　8086 存储器高、低位库的选择

$\overline{BHE}$	A_0	读/写的字节	$\overline{BHE}$	A_0	读/写的字节
0	0	同时读/写高、低两个字节	1	0	只读/写偶数地址的低位字节
0	1	只读/写奇数地址的高位字节	1	1	不传送

在 8088 系统中,可直接寻址的存储空间同样也为 1MB,但其存储器的结构与 8086 有所不同,它的 1MB 存储空间同属一个单一的存储体,即存储体为 1MB×8 位。它与总线之间的连接方式很简单,其 20 根地址线 A_{19}～A_0 与 8 根数据线分别同 8088 CPU 的对应地址线与数据线相连。8088 CPU 每访问 1 次存储器只读/写 1 字节信息,因此,在 8088 系统的存储器中不存在对准存放的概念,任何数据字都需要两次访问存储器才能完成读/写操作,故在 8088 系统中,程序运行速度比在 8086 系统中慢。

3.3.2　存储器的分段

8086/8088 CPU 的指令指针(IP)和堆栈指针(SP)都是 16 位,故只能直接寻址 64KB 的地址空间。而 8086/8088 有 20 根地址线,它允许寻址 1MB 的存储空间。如前所述,为了能寻址 1MB 的存储空间,引入了分段的概念。

在 8086/8088 系统中,1MB 存储空间可被分为若干逻辑段,其实际存储器中段的位置如图 3.12 所示。

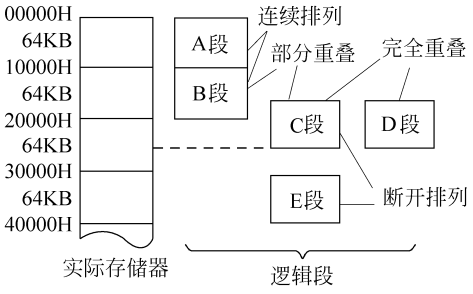


图 3.12　实际存储器中段的位置

由图 3.12 可知,每一段的大小,可能从 1 个字节开始任意递增,如 100 个字节、1000 个字节等,直至最多可包含 64KB 长的连续存储单元;每个段的 20 位起始地址(又叫段基址),是一个能被 16 整除的数(即最后 4 位为 0),它可以通过用软件在段寄存器中装入 16 位段地址来设置。

注意:段地址是 20 位段基址的前 16 位。

从图 3.12 中还可以看到内存中各个段所处位置之间的相互关系,即段和段之间可以是连续的、断开的、部分重叠的或完全重叠的。一个程序所用的具体存储空间可以为一个逻辑段,也可以为多个逻辑段。

由于段基址是由存放于段寄存器,即 CS、DS、SS 和 ES 中的 16 位段地址左移 4 位得来的,因此,程序可以从 4 个段寄存器给出的逻辑段中存取代码和数据。若要对别的段而不是当前可寻址的段中存取信息,则程序必须首先改变对应的段寄存器中段地址的内容,并将其设置成所要存取的段地址信息。

最后需要强调的是,段区的分配工作是由操作系统完成的;但是,系统允许程序员在必要时指定所需占用的内存区。

3.3.3 实际地址和逻辑地址

实际地址是指 CPU 对存储器进行访问时实际寻址所使用的地址，对 8086/8088 来说，它是用 20 位二进制数或 5 位十六进制数表示的地址。通常，实际地址又称为物理地址。

逻辑地址是指在程序和指令中表示的一种地址，它包括两部分，段地址和偏移地址。对 8086/8088 来说，前者是由 16 位段寄存器直接给出的 16 位地址；后者则是由指令寻址时的寄存器组合与位移量之和，它最终所给出的是一个 16 位的偏移量，表示所寻址的地址单元距离段起始地址之间的偏移字节的多少，故称为偏移地址（又简称为偏移量或偏移）。段地址和偏移地址都用无符号的 16 位二进制数或 4 位十六进制数表示。

对于 8086/8088 CPU 来说，由于其寄存器都是 16 位的体系结构，因此，程序中的指令不能直接使用 20 位的实际地址，而只能使用 16 位的逻辑地址。由逻辑地址计算实际地址的方法如图 3.2 所示。

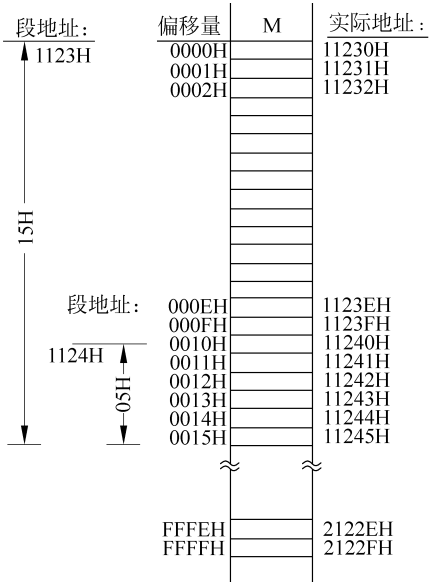


图 3.13 一个实际地址可对应多个逻辑地址

注意：一个实际地址可对应多个逻辑地址，如图 3.13 所示。图中的实际地址 11245H 可以从两个部分重叠的段中得到：一个段的段地址为 1123H，偏移地址为 15H，其实际地址为 (11230H＋15H)＝11245H；另一个段的段地址为 1124H，而偏移地址为 05H，其实际地址仍为 (11240H＋05H)＝11245H。由此可见，尽管两个段采用了不同的逻辑地址，但它们仍可获得同一个实际地址。段地址来源于 4 个段寄存器，偏移地址则来源于 IP、SP、BP、SI 和 DI。寻址时应该使用哪个寄存器或寄存器的组合，BIU 将根据执行操作的种类和要取得的数据类型来确定，如表 3.7 所示为逻辑地址源。

表 3.7 逻辑地址源

存储器操作涉及的类型	正常使用的段地址	可被使用的段地址	偏移地址
取指令	CS	无	IP
堆栈操作	SS	无	SP
变量(下面情况除外)	DS	CS,ES,SS	有效地址
源数据串	DS	CS,ES,SS	SI
目标数据串	ES	无	DI
作为堆栈基址寄存器使用的 BP	SS	CS,DS,ES	有效地址

注意：实际上，这些寻址操作都是由操作系统按默认的规则由 CPU 在执行指令时自动完成的。

3.3.4 堆栈

8086/8088 系统中的堆栈是用段定义语句在存储器中定义的一个堆栈段，与其他逻辑段一样，它可在 1MB 的存储空间中浮动。一个系统具有的堆栈数目不受限制，一个栈的深度最大为 64KB。

堆栈通过堆栈段寄存器(SS)和堆栈指针(SP)来寻址。SS 中记录的是其 16 位的段地址，它将确定堆栈段的段基址，而 SP 的 16 位偏移地址将指定当前栈顶，即指出从堆栈段的段基址到栈顶的偏移量；栈顶是堆栈操作的唯一出口，它是堆栈地址较小的一端。

若已知当前 SS=1050H，SP=0008H，AX=1234H，则 8086 系统中堆栈的入栈和出栈操作如图 3.14 所示。为了加快堆栈操作的速度，堆栈操作均以字为单位进行操作。

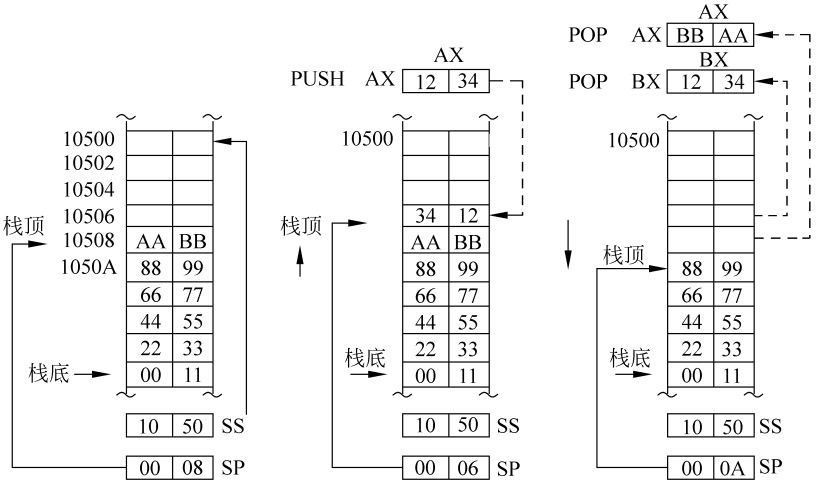


图 3.14 8086 系统的堆栈及其入栈、出栈操作

当执行 PUSH AX 指令时，是将 AX 中的数据 1234H 压入堆栈，该数据所存入的地址单元将由原栈顶地址 10508H 减 2 后的栈顶地址 10506H 给定。当执行 POP BX 指令时，将把当前堆栈中的数据 1234H 弹出并送到 BX，栈顶地址由 10506H 加 2 变为 10508H；再执行 POP AX 时，将把当前堆栈中的数据 BBAAH 送到 BX，则栈顶地址由 10508H 加 2 变为 1050AH。

3.3.5 “段加偏移”寻址机制允许重定位

如上所述，8086/8088 CPU 引入了分段技术，微处理器在寻址时利用了段基地址加偏移地址的原理，通常将这种寻址机制称为“段加偏移”寻址机制。

“段加偏移”寻址机制允许重定位(或再定位)是一种重要的特性。所谓重定位是指一个完整的程序块或数据块可以在存储器所允许的空间内任意浮动并定位到一个新的、可寻址的区域。在 8086 以前的 8 位微处理器中是没有这种特性的，而从 8086 引入分段概

念之后，由于段寄存器中的段地址可以由程序来重新设置，因而，在偏移地址不变的情况下，可以将整个存储器段移动到存储器系统内的任何区域而无需改变任何偏移地址。即“段加偏移”寻址机制可以实现程序的重定位。由此可以很容易想到，由于“段加偏移”寻址机制允许程序在存储器内重定位，因此，原来为 8086 在实模式下运行所编写的程序，在其后 80286 以上的高型号微处理器中，当系统由实模式转换为保护模式时也可以运行。这是因为，在从实模式转换为保护模式时，程序块本身的结构或指令序列都未改变，它们被完整地保留下来；而只不过在转换之后，段地址将会由系统重新设置，但偏移地址却没有改变。同样，数据块也被允许重定位，重定位的数据块也可以放在存储器的任何区域，且不需要修改就可以被程序引用。

由于“段加偏移”寻址机制允许程序和数据不需要进行任何修改，就能使它们重定位，这使应用具有一个很大的优点。因为，各种通用计算机系统的存储器结构不同，它们所包含的存储器区域也各不相同，但在应用中却要求软件和数据能够重定位；而“段加偏移”的寻址机制恰好具有允许重定位的特性，因此，这就给各种通用计算机系统在运行同一软件和数据时能够保持兼容性带来极大的方便。

例如，有一条指令位于距存储器中某段首（即段基地址）8 字节的位置，其偏移地址就是 8。当整个程序移到新的存储区，这个偏移地址 8 仍然指向距存储器中新的段首 8 个字节的位置。只是这时段寄存器的内容必须重新设置为程序所在的新存储段的起始地址。如果计算机系统没有重定位的特性，那么当一个程序在移动之前，就必须大范围地重写或更改，或者要为许多不同配置的计算机系统设计许多的程序文本，这不仅需要花费大量的时间，还可能会引起程序出错。

3.4 8086/8088 的指令系统

3.4.1 指令系统的特点及指令基本格式

8086 与 8088 的指令系统完全相同，它们是由 8 位的 8080/8085 指令系统扩展而来的，同时，它们又能在其后续的 80x86 系列的 CPU 上正确运行。因此，8086/8088 指令系统是 80x86 CPU 共同的基础。其主要特点如下：

（1）采用可变长指令，指令格式由 1～6 字节组成，比较复杂。

（2）寻址方式多样灵活，处理数据的能力比较强，可处理字节或字、带符号或无符号的二进制数据以及压缩型/非压缩型的十进制数据。

（3）有重复指令和乘除运算指令。扩充了条件转移、移位/循环指令。

（4）有软件中断功能和支持多处理器系统工作的指令。

指令格式是按指令系统的规范与要求精心设计的。了解指令格式有助于深入掌握指令代码的组成原理。指令的基本组成包括两部分，即操作码与操作数。8086/8088 的指令格式如图 3.15 所示。

其中，第 1、2 字节为基本字节，属操作码字段，B_1 给出操作码，B_2 给出寻址方式；第 3 字节 B_3 至第 6 字节 B_6 为操作数字段，将根据不同指令对地址位移量和/或立即数的设

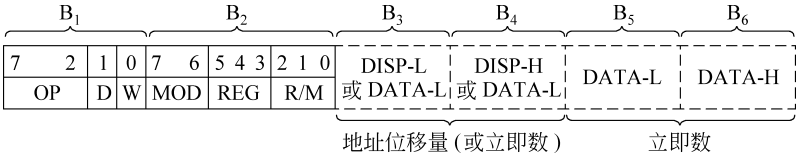


图 3.15　8086/8088 的指令格式

置做相应的安排。指令中的立即数(DATA)位于位移量(DISP)之后,均可为 8 位或 16 位。当为 16 位时,低位在前,高位在后。若指令中只有 8 位位移量(DISP 8),则 CPU 在计算有效地址(EA)时将自动用符号把它扩展为 16 位的双字节数,以保证计算不产生错误。若 B_3、B_4 有位移量,立即数就位于 B_5、B_6;否则就位于 B_3、B_4。

有关 8086/8088 指令的详细格式可参见附录 A。

3.4.2　寻址方式

CPU 的寻址方式就是根据指令功能所规定的操作码如何自动寻找相应的操作数的方式。8086/8088 的操作数可位于寄存器、存储器或 I/O 端口中。对位于存储器的操作数可采用多种不同方式进行寻址。8086/8088 不仅包含 8080/8085 的寻址方式,而且还有许多扩展。下面对 8086/8088 的寻址方式给予简要介绍。

1. 固定寻址

有些单字节指令其操作是规定 CPU 对某个固定的寄存器进行的,如加法的 ASCII 调整指令 AAA,规定被调整的数总是位于 AL 中。

该指令用来调整 AL 中的结果,此结果是把两个 ASCII 字符当作操作数相加后形成的。

2. 立即数寻址

操作数就在指令中,当执行指令时,CPU 直接从指令队列中取得该立即数,而不必执行总线周期。立即数可以是 8 位,也可以是 16 位;并规定只能是整数类型的源操作数。这种寻址主要用来给寄存器赋初值,指令执行速度快。

【例 3.2】

```
MOV AX,1680H                ; 将 1680H 送 AX,AH 中为 16H,AL 中为 80H
```

【例 3.3】

```
MOV AX,'AB'                 ; 将 ASCII 码'AB'在内存中的字内容 BA(4241H)送 AX
```

3. 寄存器寻址

操作数放在 CPU 的寄存器(如 AX、BX、CX 和 DX 等)中,而寄存器名在指令中指出。这种寻址的指令长度短,操作数就在 CPU 内部进行,不需要使用总线周期,所以,执行速度转快。

对 16 位操作数来说,寄存器可以为 8 个 16 位通用寄存器。而对 8 位操作数来说,寄存器只能为 AH、AL、BH、BL、CH、CL、DH、DL。在一条指令中,源操作数或/和目的操作数都可以采用寄存器寻址方式。