第3章

80x86 微处理器

Intel 80x86 系列 CPU 在微型计算机的发展过程中具有不可替代的地位。虽然目前高档微机系统已经普及，但 8086 CPU 作为主流微型计算机的基础，能够系统、全面地反映微型计算机系统的工作原理。本章以 8086 CPU 为例，介绍了微处理器的内部结构、外部基本引脚、工作方式、总线和时序，以及 8086 CPU 的存储器组织、中断系统等内容。为学习汇编语言程序设计和接口应用技术打下基础。

3.1 Intel 8086 微处理器

Intel 8086 微处理器是由美国 Intel 公司 1978 年推出的高性能的 16 位微处理器，是第三代微处理器的典型代表。它有 20 根地址总线，直接寻址能力达 1MB，具有 16 根数据总线，内部总线和 ALU 均为 16 位，可进行 8 位和 16 位操作。

Intel 8086 微处理器具有丰富的指令系统，采用多级中断技术、多重寻址方式、多重数据处理形式、段式存储器结构、硬件乘除法运算电路，增加了预取指令的队列寄存器等，一问世就显示出了强大的生命力，以它为核心组成的微机系统性能已达到当时中、高档小型计算机的水平。8086 的一个突出特点是多重处理能力，用 8086 CPU、8087 浮点运算器及 8089 I/O 处理器组成的多处理器系统，可大大提高其数据处理和输入/输出能力。另外，与 8086 配套的各种外围接口芯片非常丰富，用户可以方便地开发各种系统。

3.2 8086 的存储器组织

3.2.1 寻址空间和数据存储格式

1. 寻址空间

程序和数据存放在内存中，CPU 根据**地址**访问内存，找到需要的指令或数据。**寻址空间**就是指存储器地址允许的最大范围，即 CPU 能访问多大范围的地址。

计算机的寻址空间是由 CPU 地址总线的位数决定的。当存储器按字节编址时，若地址总线为 n 位，CPU 寻址范围是 2^n 字节。例如，8086 CPU 有地址总线 20 位，寻址能力为 $2^{20}=1$MB；80286 的地址总线为 24 位，CPU 的寻址能力为 $2^{24}=16$MB；20386 地址总线为 32 位，CPU 的寻址能力为 $2^{32}=4$GB。寻址范围的大小和内存的实际容量并不一定相等，如果地址总线位数不够，即使有很大的内存也无法完全访问。而对于当前主流的微处理器，其

寻址能力已远远超过实际的内存容量。

2. 8086 存储器的组织及寻址

8086 CPU 地址总线 20 位，寻址能力为 1MB，每字节用唯一的一个地址码标识。地址的范围为 0～$2^{20}-1$，用十进制表示为 0～1048575。但习惯上使用十六进制表示，即 00000H～FFFFFH。这种每字节对应一个地址的方式称为**“按字节编址”**，如图 3.1 所示。

十六进制地址	二进制地址	存储器
00000H	0000 0000 0000 0000 0000B	
00001H	0000 0000 0000 0000 0001B	
00002H	0000 0000 0000 0000 0010B	
00003H	0000 0000 0000 0000 0011B	
：	：	⋮
：	：	
FFFFDH	1111 1111 1111 1111 1101B	
FFFFEH	1111 1111 1111 1111 1110B	
FFFFFH	1111 1111 1111 1111 1111B	

图 3.1　存储空间的字节编址

8086 系统的存储空间虽然按照字节编址，但在实际编程时，一个变量可以是字节、字或双字类型。

（1）字节数据（BYTE）。字节数据 8 位，对应的地址可以是**偶地址**（地址的最低位 $A_0=0$），也可以是**奇地址**（$A_0=1$）。当 CPU 存取字节数据时，只需给出对应的实际地址即可。

（2）字数据（WORD）。

Intel 8086 是 16 位机，字长 16 位，每个字数据存放在两个连续的字节单元中。其中高 8 位存放在高地址字节（称为**高字节**），低 8 位存放在低地址字节（称为**低字节**），并规定将低字节的地址作为这个字的地址（**字地址**），如图 3.2 所示。若该字地址位于偶地址，即低字节地址为偶数，称为**规则字**，否则称为**非规则字**。

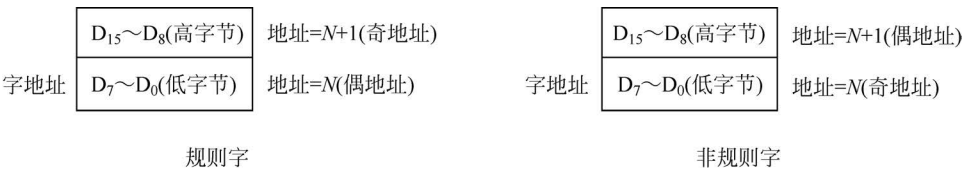


图 3.2　字数据

（3）双字数据（DOUBLE WORD）。

双字数据占用 4 个连续字节单元，并规定最低字节地址为双字的地址，如图 3.3 所示。

8086 系统将 1MB 的内存分为两个块，每个块的容量都是 512KB，如图 3.4 所示。其中和数据总线 D_{15}～D_8 相连的块称为**高位字节块**，它由所有的奇地址单元组成（对应双字数据的 D_{15}～D_8、D_{31}～D_{24} 位），也称为**奇地址块**；和数据总线 D_7～D_0 相连的块称为**低位字节块**，它由所有的偶地址单元组成（对应双字数据的 D_7～D_0、D_{23}～D_{16} 位），也称为**偶地址块**。

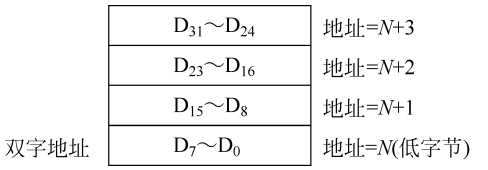


图 3.3 双字数据

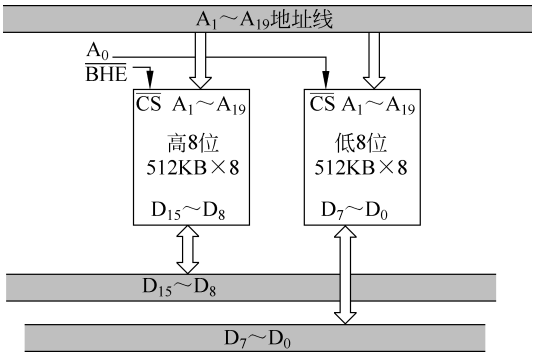


图 3.4 8086 系统的存储器结构

高位字节块利用 $\overline{BHE}$ 信号作为该块的选择信号；低位字节块利用地址线 $A_0=0$(低电平)作为该块的选择信号。对于每个块内的 512 字节，通过 $A_{19}\sim A_1$ 共 19 位地址线进行寻址。当 CPU 访问内存时，首先根据 $\overline{BHE}$ 和 A_0 信号配合判断字节所在的块，然后通过 $A_{19}\sim A_1$ 访问相应块中的存储单元。表 3.1 所示为 $\overline{BHE}$ 和 A_0 组合对应的存取操作。

表 3.1 $\overline{BHE}$ 和 A_0 的代码组合对应的存取操作

数据类型	$\overline{BHE}$	A_0	操　作	数　据
规则字	0	0	从偶地址开始读/写一个字	$D_{15}\sim D_0$
字节	0	1	从奇地址开始读/写一字节	$D_{15}\sim D_8$
	1	0	从偶地址开始读/写一字节	$D_7\sim D_0$
非规则字	0	1	从奇地址开始读写一个字(非规则字)，第一总线周	$D_{15}\sim D_8$
	1	0	期高 8 位数据有效，第二总线周期低 8 位数据有效	$D_7\sim D_0$
—	1	1	无效	—

当 $\overline{BHE}=0$、$A_0=0$ 时，高位字节块和低位字节块同时有效，8086 CPU 通过 $A_{19}\sim A_1$ 同时在两个块中各寻址一字节，高位字节块中的数据经数据线的高 8 位($D_{15}\sim D_8$)传送，低位字节块中的数据经数据线的低 8 位($D_7\sim D_0$)传送，完成一个规则字的存取操作。

当 $\overline{BHE}=0$、$A_0=1$ 时，高位字节块有效，通过 $A_{19}\sim A_1$ 在该块中寻址一字节，并经数据线的高 8 位($D_{15}\sim D_8$)传送。

当 $\overline{BHE}=1$、$A_0=0$ 时，低位字节块有效，通过 $A_{19}\sim A_1$ 在该块中寻址一字节，并经数据线的低 8 位($D_7\sim D_0$)传送。

而对于非规则字的存取操作，则需要两个总线周期才能完成：在第一个总线周期中，$\overline{BHE}=0$、$A_0=1$，存取高 8 位数据；在第二个总线周期中，$\overline{BHE}=1$、$A_0=0$，存储器地址加

1，访问低 8 位数据。

这里存取操作所需的 $\overline{BHE}$ 及 A_0 信号是由字操作指令给出的。

3.2.2　存储器的分段结构和物理地址的形成

1. 存储器的分段结构

8086 CPU 的 20 位地址总线，可直接寻址 1MB 存储器物理空间，其地址范围为 00000H～FFFFFH，与存储单元一一对应的 20 位地址，称为存储单元的**物理地址**。

但 8086CPU 内部寄存器均为 16 位，8086 指令中的地址码也只有 16 位，那么，利用 16 位的寄存器如何表示 20 位地址呢？为了解决这个问题，8086 存储器采用分段管理，将 1MB 的存储空间分成若干逻辑段，每个逻辑段长度≤64KB，并且规定段起始地址的低 4 位必须为 0。将段起始地址的高 16 位称为该段的**段地址**（或**段基地址**），段内存储单元相对于段起始地址偏移量称为当前段内的**偏移地址**（Offset Address），偏移地址也是 16 位。这样，任何一个内存单元的地址都可以用段地址和偏移地址来表示，其格式为：

```
段地址：偏移地址
```

这种地址表示的方式称为**逻辑地址**。例如，逻辑地址 7018H：5E7FH 表示段地址为 7018H，偏移地址为 5E7FH。可见，通过对内存的分段，将 20 位的物理地址，用 1 个 16 位的段地址和 1 个 16 位的偏移地址组成的逻辑地址表示，解决了 16 位 CPU 寻址 1MB 存储空间的问题。另外，内存分段也为程序的浮动分配创造了条件。

2. 物理地址的形成

根据逻辑地址，可以求出它对应的物理地址：

物理地址＝段地址×10H＋偏移地址

例如，逻辑地址 7018H：5E7FH 表示的物理地址是 7018H×10H＋5E7FH＝75FFFH。

物理地址的计算过程也可以表示为：

物理地址＝段地址×16＋偏移地址

8086CPU 有一个专门的 20 位地址加法器实现逻辑地址到物理地址的变换。当 CPU 寻址某个存储单元时，首先将段地址左移 4 位，再与 16 位偏移地址相加，从而形成 20 位的物理地址，如图 3.5 所示。

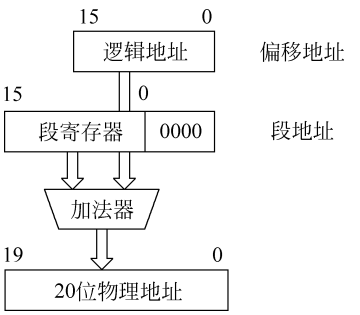


图 3.5　物理地址形成过程

3. 按信息特征分段存储

在存储器中存储的信息可分为程序指令、数据和计算机系统的状态等信息。为了寻址及操作的方便，8086 系统中，存储器空间根据信息特征分段存储。一般可将存储器划分为程序段、数据段、堆栈段和附加段。**程序段**中存储程序的指令代码；**数据段**和**附加段**中存储数据、中间结果和最后结果；**堆栈段**存储压入堆栈的数据或状态信息。在取指令时，CPU 自动选择代码段寄存器（CS）；堆栈操作时，CPU 自动选择堆栈段寄存器（SS）；每当存取操作数时，CPU 会自动选择数据段寄存器（DS）或附加段寄存器（ES）。

3.3 8086微处理器的内部结构

3.3.1 8086 CPU的内部结构

CPU的任务是执行存放在存储器中的指令序列,即取指令和执行指令。

8086 CPU内部由两大功能部件组成:**总线接口部件**(Bus Interface Unit,BIU)和**执行部件**(Execute Unit,EU),其结构框图如图3.6所示。

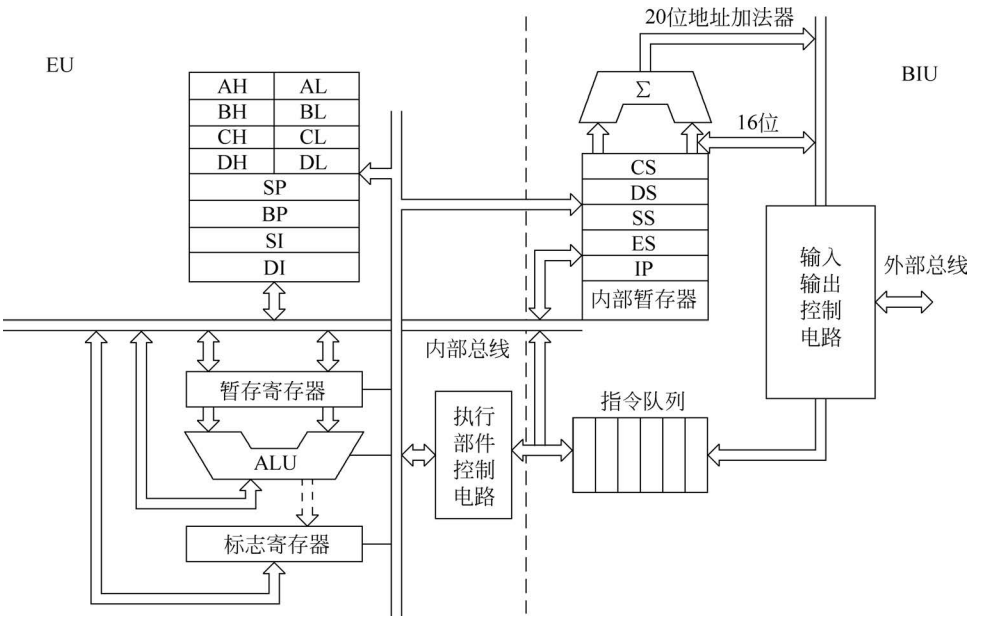


图3.6 Intel 8086 CPU逻辑结构框图

在执行指令的过程中,BIU和EU是既分工又合作的两个独立部件。BIU部件负责存取指令和数据,EU部件负责执行指令,它们的操作是并行的,如图3.7所示。

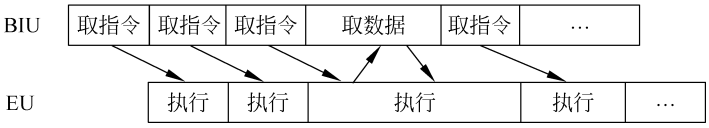


图3.7 Intel 8086执行指令过程

BIU和EU是各自独立工作的,在EU执行指令的同时,BIU可预取下面一条或几条指令。也就是说,一条指令在EU中执行的同时,BIU就可以提前取出下一条(或多条)指令,放在指令队列中排队。当CPU执行完当前指令后,就可立即执行存放在指令队列中的下一条指令。EU和BIU的并行操作提高了CPU和总线的利用率,加快了程序的运行速度。

BIU和EU的操作遵循下列原则。

(1) 每当8086 CPU指令队列中有两个空字节时,BIU就会自动寻找空闲的总线周期

进行预取指令操作，直到指令队列填满为止。

（2）当指令队列缓冲器中存有一条以上的指令时，EU 就立即开始执行。

（3）每当 EU 执行一条转移、调用或返回指令后，BIU 清空指令队列，并从转移后的当前地址取出指令送 EU 执行，实现程序段的转移；然后在新地址基础上再做预取指令操作。

1. 总线接口部件

总线接口部件 BIU 完成 CPU 与主存储器或 I/O 端口间的信息传送，它的主要功能如下。

（1）预取指令序列。BIU 会自动进行预取指令操作，并将从存储器中取出的指令按先后次序存入指令缓冲寄存器，以便 EU 按顺序执行这些指令。

（2）存取数据。在指令执行期间，BIU 配合 EU，从指定的内存单元或 I/O 端口中取出数据传送给执行单元，或者把执行单元的处理结果传送到指定的内存单元或 I/O 端口中。

（3）将访问主存的逻辑地址转换为实际的物理地址。

8086 的 BIU 由一个 20 位地址加法器、4 个 16 位段寄存器、一个 16 位指令指针 IP、一个 6 字节的指令队列缓冲器，以及总线控制逻辑电路等组成。

（1）地址加法器和段寄存器。地址加法器将 16 位的段寄存器内容左移 4 位，与 16 位偏移地址相加，形成 20 位的物理地址。

（2）指令指针 IP。16 位指令指针 IP 用来存放下一条将要执行的指令在代码段中的偏移地址。

（3）指令队列缓冲器。指令队列寄存器用来缓存 BIU 取出待执行的指令。该队列寄存器按“先进先出”的方式工作。

（4）总线控制逻辑。总线控制逻辑将 8086 CPU 的内部总线和系统总线相连，是 8086 CPU 与内存单元或 I/O 端口进行数据交换的必经之路。它包括 16 条数据总线、20 条地址总线和若干条控制总线，CPU 通过这些总线与外部取得联系，从而构成各种规模的 8086 微型计算机系统。

2. 执行部件

执行部件（EU）负责进行所有指令的解释和执行，同时管理 EU 中相关的寄存器。它的主要功能如下。

（1）从指令队列中取出指令代码，由 EU 控制器进行译码，然后控制各部件完成指令规定的操作。

（2）对操作数进行算术和逻辑运算，并将运算结果的特征状态存放在标志寄存器中。

（3）当需要与主存储器或 I/O 端口传送数据时，EU 向 BIU 发出命令，并提供要访问的内存地址或 I/O 端口地址及传送的数据。

EU 由一个 16 位的算术逻辑运算单元（ALU）、8 个 16 位通用寄存器、一个 16 位标志寄存器 FLAGS、一个数据暂存寄存器和 EU 控制电路组成。

（1）算术逻辑运算单元。算术逻辑运算单元是一个 16 位的运算器，可用于 8 位、16 位二进制算术和逻辑运算，也可计算内存地址的 16 位偏移量。

（2）通用寄存器组。通用寄存器组包括 4 个 16 位的数据寄存器 AX、BX、CX、DX 和 4 个 16 位指针与变址寄存器 SP、BP 与 SI、DI。

（3）标志寄存器。标志寄存器是一个 16 位的寄存器，用来反映 CPU 运算的状态特征

和存放某些控制标志。

(4) 数据暂存寄存器。数据暂存寄存器协助 ALU 完成运算,暂存参加运算的数据。

(5) EU 控制电路。EU 控制电路负责从 BIU 的指令队列缓冲器中取指令,并对指令译码,根据指令的要求向 EU 内部各部件发出控制命令,以完成各条指令规定的功能。

执行单元中的各部件通过 16 位的内部总线连接在一起,在内部实现快速数据传输。值得注意的是,这个内部总线与 CPU 外接的总线之间是隔离的,即这两个总线可以同时工作而互不干扰。EU 从 BIU 的指令队列缓冲器中取出指令并执行,由 BIU 通过外部总线从存储器中取得指令。

在指令执行过程中可能需要从存储器中存取数据,这时,EU 单元将 16 位有效地址提供给 BIU,在 BIU 中转换为 20 位的物理地址,送到外部总线进行寻址。

3.3.2 8086 CPU 的寄存器结构

8086 微处理器内部共有 14 个 16 位寄存器。这 14 个寄存器按其用途可分为数据寄存器、段寄存器、地址指针与变址寄存器、控制寄存器。

8086 CPU 内部寄存器如图 3.8 所示。

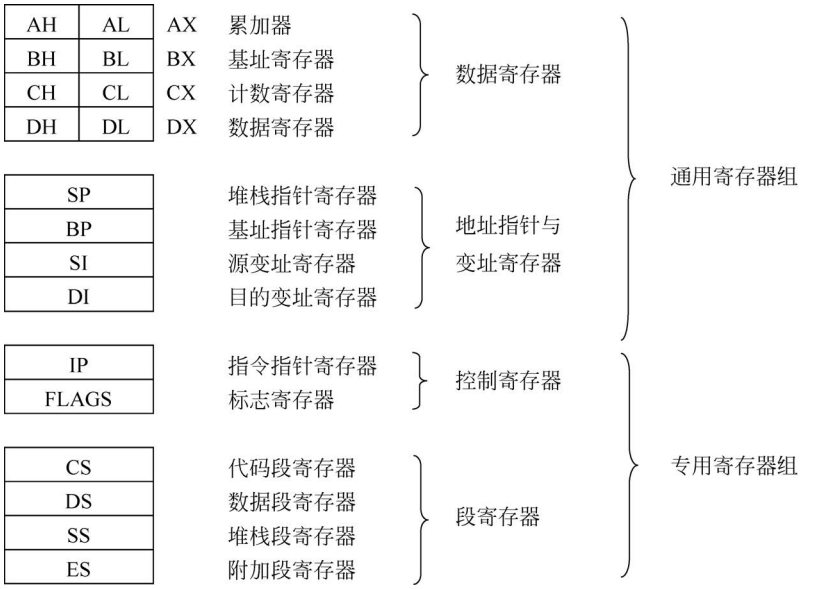


图 3.8 8086 CPU 内部寄存器

1. 段寄存器

在 8086 系统中,存储器是分段管理的,访问存储器的地址码由段地址和段内偏移地址两部分组成。段寄存器用来存放段地址,包括 4 个 16 位寄存器:代码段寄存器 CS、数据段寄存器 DS、堆栈段寄存器 SS 和附加段寄存器 ES。

(1) 代码段寄存器 CS(Code Segment):存放当前正在运行的程序所在段的段地址,段内的偏移地址则由 IP 提供。

(2) 数据段寄存器 DS(Data Segment):存放当前程序使用的数据所在段的段地址。

(3) 堆栈段寄存器 SS(Stack Segment)：存放当前堆栈段的段地址。堆栈是在存储器中开辟的、按照“后进先出”原则组织的一个特殊区域。主要用于子程序调用时保护现场和保存断点。

(4) 附加段寄存器 ES(Extra Segment)：存放当前程序使用附加段的段地址，附加段是一个附加的数据段，在执行串操作指令时，作为目的串地址使用。

2. 数据寄存器

数据寄存器用来暂时存放计算过程中所用到的操作数、结果或其他信息，包括累加器 AX、基址寄存器 BX、计数寄存器 CX 和数据寄存器 DX。这 4 个寄存器都是 16 位的，它们都可以以字(16 位)或字节(8 位)形式访问。例如，对于 AX 寄存器，可以分别访问高字节 AH 或低字节 AL 寄存器。因此，它们既可作为 4 个 16 位数据寄存器使用(AX、BX、CX、DX)，也可作为 8 个 8 位数据寄存器使用(AH、AL、BH、BL、CH、CL、DH、DL)。

AX、BX、CX、DX 还可以用于各自的专用目的。

AX(Accumulator)：主要作为累加器使用，它是算术运算的主要寄存器。另外，所有的 I/O 指令都使用这一寄存器与外部设备传送信息。

BX(Base)：在计算存储器地址时，它经常用作基址寄存器。

CX(Count)：在循环(Loop)和串处理指令中，用作隐含的寄存器。

DX(Data)：一般在做双字运算时把 DX 和 AX 组合在一起存放一个双字数，DX 用来存放高位字。此外，对于某些 I/O 操作，DX 可用来存放 I/O 的端口地址。

3. 地址指针与变址寄存器

地址指针与变址寄存器包括 4 个 16 位寄存器：堆栈指针寄存器 SP、基址指针寄存器 BP、源变址寄存器 SI 和目的变址寄存器 DI。它们一般用来存放主存地址的段内偏移地址，用于形成 20 位物理地址。另外，它们也可以和数据寄存器一样在运算过程中存放操作数，但只能以字(16 位)为单位使用。

(1) 堆栈指针寄存器 SP(Stack Pointer)：指出在堆栈段中栈顶的偏移地址。

(2) 基址指针寄存器 BP(Base Pointer)：指出要处理的数据在堆栈段中的起始地址，特别值得注意的是，凡包含 BP 的寻址方式中，如无特别说明，其段地址由堆栈段寄存器 SS 提供。也就是说，该寻址方式是对堆栈区的存储单元寻址的。

(3) 变址寄存器 SI(Source Index)和 DI(Destination Index)：在某些间接寻址方式中，用来存放段内偏移量的全部或一部分。在字符串操作指令中，SI 用作源变址寄存器，DI 用作目的变址寄存器。

4. 控制寄存器

控制寄存器包括指令指针寄存器 IP 和标志寄存器 FLAGS。

(1) 指令指针寄存器 IP(Instruction Pointer)：用来存放下一条将要执行的指令在代码段中的偏移地址，程序员不可以直接使用，每当执行一次取指令操作，它将自动加 1，“1”指 1 条指令，即指令的字节数不同，IP 加“1”的字节数也不同。它和 CS 相结合，形成指向指令存放单元的物理地址。

(2) 标志寄存器 FLAGS：存放该处理器的程序状态字。这是一个 16 位的寄存器，但实际上 8086 CPU 只用到 9 位，其中 6 位为状态标志位，3 位为控制标志位，如图 3.9 所示。

D_{15}	D_{14}	D_{13}	D_{12}	D_{11}	D_{10}	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
				OF	DF	IF	TF	SF	ZF		AF		PF		CF

图 3.9 8086 CPU 的标志寄存器

状态标志反映了当前运算和操作结果的状态条件,可作为程序控制转移与否的依据。它们分别是 CF、PF、AF、ZF、SF 和 OF。

CF(Carry Flag):进位标志位。算术运算指令执行后,若运算结果的最高位产生进位或借位,则 CF=1;否则 CF=0。

PF(Parity Flag):奇偶标志位。反映运算结果中 1 的个数是偶数还是奇数。运算指令执行后,若运算结果的低 8 位中含有偶数个 1,则 PF=1;否则 PF=0。

AF(Auxiliary carry Flag):辅助进位标志位。算术运算指令执行后,若运算结果的低 4 位向高 4 位(即 D_3 位向 D_4 位)产生进位或借位,则 AF=1;否则 AF=0。

ZF(Zero Flag):零标志位。若指令运算结果为 0,则 ZF=1;否则 ZF=0。

SF(Sign Flag):符号标志位。它与运算结果的最高位相同。若字节运算时 D_7 位为 1 或字运算时 D_{15} 位为 1,则 SF=1;否则 SF=0。用补码运算时,它能反映结果的符号特征。

OF(Overflow Flag):溢出标志位。当补码运算有溢出时(字节运算时为 $-128\sim+127$,字运算时为 $-32768\sim+32767$),则 OF=1;否则 OF=0。

控制标志位则可以由指令进行置位和复位,用来控制 CPU 的操作,它包括 DF、IF、TF。

DF(Direction Flag):方向标志位。用于串操作指令,指定字符串处理时的方向。设置 DF=0 时,每执行一次串操作指令,地址指针内容将自动递增;设置 DF=1 时,地址指针内容将自动递减。可用专用指令设置或清除 DF 位。

IF(Interrupt Enable Flag):中断允许标志位。用来控制 8086 CPU 是否允许接收外部中断请求。设置 IF=1 时,允许响应可屏蔽中断请求;设置 IF=0 时,禁止响应可屏蔽中断请求。可用专用指令设置或清除 IF 位。注意,IF 的状态不影响非屏蔽中断请求(NMI)和 CPU 内部中断请求。

TF(Trap Flag):单步标志位(或跟踪标志位)。它是为调试程序而设定的陷阱控制位。设置 TF=1 时,使 CPU 进入单步执行指令工作方式,此时 CPU 每执行完一条指令就自动产生一次内部中断;当该位复位后,CPU 恢复正常工作。没有专用指令设置或清除 TF 位。

【例 3.1】 设(AX)=0010 0011 0100 1101B,(DX)=0101 0010 0000 1001B,试指出两数相加后,6 位标志位的状态。

解 用补码公式对两数进行运算,并按定义对结果进行判别。

计算机中用补码存储数据,两数相加过程为:

$$\begin{array}{rl} & \mathbf{0}\ 010\ 0011\ 0100\ 1101 \\ + & \mathbf{0}\ 101\ 0010\ 0000\ 1001 \\ \hline & \mathbf{0}\ 111\ 0101\ 0101\ 0110 \end{array}$$

根据两数相加结果,可得如下结论。

① 结果非零,故 ZF=0。

② 低 8 位中共有 4 个 1(偶数个),故 PF=1。

③ 根据符号位,可知 SF=0。

④ 运算结束后，向更高位无进位，故 CF＝0。

⑤ 运算结果无溢出，故 OF＝0。

⑥ D_3 位向 D_4 位产生进位，故 AF＝1。

3.4　总线的工作周期

指令的执行是在统一的时钟脉冲 CLK 的控制下，按节拍逐步进行的，一个时钟脉冲时间称为一个**时钟周期**(Clock Cycle)，也称为一个 T 周期。时钟周期由计算机的主频决定，是 CPU 的定时基准。例如，8086 的主频为 5MHz，一个时钟周期为 200ns。

CPU 与外部交换信息总是通过总线进行的。CPU 通过总线对存储器或外设 I/O 端口进行一次访问所需要的时间称为**总线周期**(Bus Cycle)。8086 CPU 的一个基本的总线周期由 4 个时钟周期组成，分别称为 T_1、T_2、T_3 和 T_4。

一个总线周期完成一次数据传输，至少要有传送地址和传送数据两个过程。在第一个时钟周期 T_1 期间由 CPU 输出地址，在随后的 3 个时钟周期(T_2、T_3 和 T_4)期间用以传送数据。换言之，数据传送必须在 T_2～T_4 这 3 个周期内完成，否则在 T_4 周期后，总线将进行另一次操作，开始下一个总线周期。

在实际应用中，如果一些慢速设备在 3 个 T 周期内无法完成数据读/写时，在 T_4 后总线就不能为它们所用，这会造成系统读/写出错。为此，在总线周期中允许插入等待周期 T_W。当被选中进行数据读/写的存储器或外设无法在 3 个 T 周期内完成数据读/写时，就由其发出一个请求延长总线周期的信号到 8086 CPU 的 READY 引脚，8086 CPU 收到该请求后，就在 T_3 与 T_4 之间插入等待周期 T_W，加入 T_W 的个数与外部请求信号的持续时间长短有关，T_W 也以时钟周期 T 为单位，在 T_W 期间，总线上的状态一直保持不变。

如果在一个总线周期后不立即执行下一个总线周期，即总线上无数据传输操作，系统总线处于空闲状态，则这时执行**空闲周期** T_i，T_i 也以时钟周期 T 为单位，两个总线周期之间插入几个 T_i 与 8086 CPU 执行的指令有关。例如，在执行一条乘法指令时，需用 124 个时钟周期，而其中可能使用总线的时间极少，而且预取队列的充填也不用太多的时间，加入的 T_i 可能达到 100 多个。

总线周期时序如图 3.10 所示。

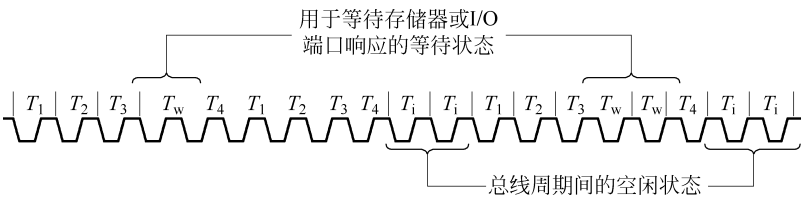


图 3.10　总线周期时序

一条指令从开始取指令到最后执行完毕所需的时间称为一个**指令周期**。不同的指令因其操作性质不同，执行时间的长短可能不同，所以指令周期也就不同。一个指令周期由一个或若干总线周期组成。

CPU 执行某一个程序之前，先要把编译后的目标程序放到主存储器的某个区域。在启

动执行后,CPU 就发出读指令的命令,根据代码段寄存器 CS 和指令指针 IP 生成 20 位物理地址并将其输出到地址总线上,在存储器中读取相应的存储单元,把它送至 CPU 的指令寄存器中;CPU 对读出指令经过译码器分析之后,发出一系列控制信号,执行指令规定的全部操作,控制各种信息在系统各部件之间传送。每条指令的执行由取指令、译码和执行等操作组成。

3.5 8086 中断系统

8086 中断系统可以直接识别和处理 256 个不同的中断源。这 256 个中断源都有唯一的一个中断识别号,又称中断类型码(0～255),与之对应。

3.5.1 8086 中断类型

根据中断的产生原因,8086 中断系统将 256 个中断源分为两大类:硬件中断与软件中断。

1. 硬件中断

中断起初是作为 CPU 与外围设备交换信息的一种同步控制方式而出现的,这类中断常称为硬件中断。**硬件中断**又称**外部中断**,它是由处理器外部的硬件、外围设备的请求而引起的中断。8086 有两条硬件中断请求信号线:NMI(非屏蔽中断)和 INTR(可屏蔽中断),外围设备是通过中断请求线向 CPU 提出中断请求的。

(1) 可屏蔽中断。由 INTR 线上的中断请求信号引起的中断称为**可屏蔽中断**。可屏蔽中断 INTR 受中断允许触发器状态的影响,这种请求可以用 CPU 指令 CLI 来屏蔽(使 IF=0),也可以用指令 STI 允许(使 IF=1)。出现在 INTR 线上的中断请求信号(即有效的高电平)必须保持到当前执行的指令结束为止。

CPU 在当前指令周期的最后一个 T 状态采样中断请求线 INTR,若发现有可屏蔽中断请求,且中断是开放的(IF=1),则 CPU 转入中断响应周期。8086 进入两个连续的中断响应周期,每个响应周期都是由 4 个 T 状态组成,而且都发出有效的中断响应信号。请求中断的中断源必须在第 2 个中断响应周期的 T_3 状态前,将其中断类型号送至 CPU 的数据总线。CPU 在 T_4 状态的前沿采样数据总线,获取中断类型号。CPU 根据中断类型号获取对应的中断服务程序入口地址,从而转去执行相应的服务程序。在一个系统中,产生可屏蔽中断的中断源可以有多个,为了协助 CPU 按中断优先权高低处理多个中断源,系统中常采用专门的中断控制器 8259 配合工作,实现多个中断源的管理。中断类型码在 08H～0FH 和 070H～077H 之间的中断属于这一级中断。

(2) 非屏蔽中断。由 NMI 线上的中断请求信号引起的中断称为**非屏蔽中断**。非屏蔽中断的中断类型码为 2。非屏蔽中断 NMI 具有比可屏蔽中断 INTR 更高的优先权。当 INTR、NMI 线上同时发生中断申请时,CPU 将首先响应 NMI 中断。非屏蔽中断的特点是不受中断允许触发器 IF 状态的影响,即不能被 CPU 用指令 CLI 来屏蔽。

当 NMI 线上出现一个由低到高的上跳边沿触发的中断请求信号后(持续时间需大于两个时钟周期),不管中断允许标志位 IF 的状态如何,都会在当前指令执行完以后,立即转入中断处理——转去执行中断类型号为 2 的非屏蔽中断的中断服务程序。

非屏蔽中断常用于紧急情况的故障处理。8086 使用 NMI 中断服务程序对 RAM 奇偶校验错、I/O 通道校验错或协处理器 8087 运算错进行处理。

2. 软件中断

随着计算机技术的发展和应用需求的提高，中断的概念也随之拓宽。除了传统的硬件中断外，又产生了软件中断的概念，在高档微处理器中则进一步丰富了软件中断的种类，延伸了其内涵，把许多在执行指令过程中产生错误的事件也纳入中断处理的范围。

软件中断是由处理器内部事件产生的中断，又称**内部中断**。它主要由指令驱动或由指令通过 CPU 状态间接驱动来引起中断。

软件中断有以下几种类型。

(1) 除法错中断。中断类型码为 0。当执行 DIV、IDIV 指令时，若用零作除数，或者商超过了寄存器所能表达的范围，则无条件产生该中断。

(2) 单步中断。中断类型码为 1。这是在调试程序过程中为单步运行程序提供的中断形式。当设定标志寄存器中陷阱标志 TF=1 时，CPU 每执行完一条指令后就产生该中断。若 TF=0，则处理器按正常方式连续执行指令。

在 8086 指令系统中没有能直接设置或改变 TF 的命令。若要设置或改变 TF 状态，则可以使用 PUSHF 指令把 16 位标志寄存器压入堆栈，并设法把栈顶 16 位字的第 8 位变为所要的状态(其他位不变)，然后用 POPF 把栈顶字弹出到标志寄存器中。

(3) 断点中断。中断类型码为 3。这是在调试程序过程中为设置程序断点而提供的中断形式。设置断点或执行 INT 3 指令可产生该中断。

(4) 溢出中断。中断类型码为 4。在算术运算程序中，若在算术运算之后加入一条 INTO 指令，则 INTO 指令将测试溢出标志 OF。当 OF=1(表示算术运算有溢出)时，该中断发生。

(5) 中断指令 INT n 引起的中断。用户可用中断指令 INT n 产生指定类型 n 的任何中断。当执行这条指令时，CPU 立即产生中断类型号为 n 的中断响应。

DOS 操作系统和基本输入/输出系统 BIOS 提供了大量的软件中断来实现系统功能的调用。用户在程序设计中可以利用 INT n 指令直接引用这些系统功能。

3. 中断优先权

在以下中断中，8086 规定中断优先权从高到低的顺序为①＞②＞③＞④。

① 除法错、溢出中断指令 INTO、中断指令 INT n。

② 非屏蔽中断 NMI。

③ 可屏蔽中断 INTR。

④ 单步中断。

3.5.2　中断向量与中断向量表

为了区分各中断源，不同的中断源都有唯一标识的**中断类型码**(也称为中断向量号或中断矢量号)。由中断类型码来查找中断入口地址进而转向中断服务程序的方法，称为**向量中断**。每个中断类型码对应一个**中断向量**，即中断服务程序的入口地址。8086 CPU 的中断系统就是采用这种向量中断来管理中断向量的。

每个中断服务程序都有其唯一确定的入口地址，包括中断服务程序的段地址 CS 和偏移地址 IP，共 4B。把系统中每一个中断源的中断服务程序入口地址集中起来，按中断类型

码的顺序存放在某一连续排列的存储区域内,这个存放中断入口地址的存储区就称为**中断向量表**。8086系统的中断向量表如图3.11所示。

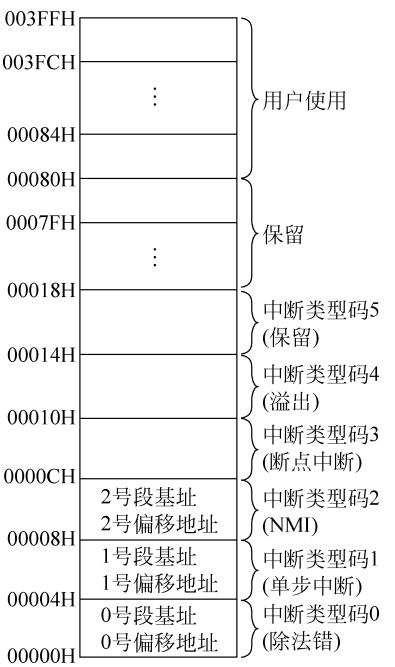


图3.11 8086系统的中断向量表

8086微机系统可提供256个不同类别的中断,由于每个中断入口地址需占用4B的地址空间,故其中断向量表需占用256×4=1KB的地址空间。8086微机系统在其内存的最低端开辟了1KB的存储区(即地址为00000H~003FFH)作为中断向量表。微机系统初始化时,系统顺序将各中断源(0~255)的中断服务程序入口地址填写在该表中。其中,中断类型码n的中断向量存储在中断向量表的地址为0000:4n,即对应的中断入口地址放在起始地址为0000:4n的连续4B的地址空间中,低地址的两字节存放中断服务程序入口地址的偏移地址(IP),高地址的两字节存放中断服务程序入口地址的段基址(CS)。这样,在响应中断时,CPU就可以根据所得到的中断类型码n,查找中断向量表,在从地址0000:4n开始的连续4B单元中获取中断源n的中断服务程序首地址。

中断向量表建立了不同的中断源与其相应的中断服务程序首地址之间的联系,它使CPU在响应中断时可以依据中断类型码自动转向中断服务程序,是8086中断系统中特有的、不可缺少的组成部分。

3.5.3 8086中断处理过程

不同类型的中断,其中断响应过程也不完全相同,如图3.12所示。

1. 可屏蔽中断的中断过程

8086 CPU是通过中断控制器8259A管理外部可屏蔽中断的。

(1) 中断源通过中断控制器8259向CPU发出中断请求信号。当有一个或多个外设向中断控制器8259发出中断请求时,8259经过识别,判优后通过INTR信号线向8086 CPU发出一个高电平的中断请求信号。

(2) CPU在每一个指令周期的最后一个时钟周期采样INTR信号线,若发现该信号线为高电平,即有中断源向CPU发中断请求,则CPU首先检测IF标志位的值,如果IF=1,表示CPU允许中断,CPU开始响应中断;否则,CPU忽略该级中断请求,继续执行原程序或响应更高优先级的中断请求服务。

(3) 当CPU开始响应可屏蔽中断请求时,通过$\overline{\text{INTA}}$信号线向8259连续发出两个负脉冲的中断响应信号。CPU的第1个中断响应信号用来通知8259,CPU准备响应中断,要求8259准备好中断类型码;当8259接收到CPU发来的第2个中断响应信号以后,立即将准备好的中断类型码通过数据线送至CPU的内部数据寄存器中,以便CPU据此找到相应的中断服务子程序的入口地址。

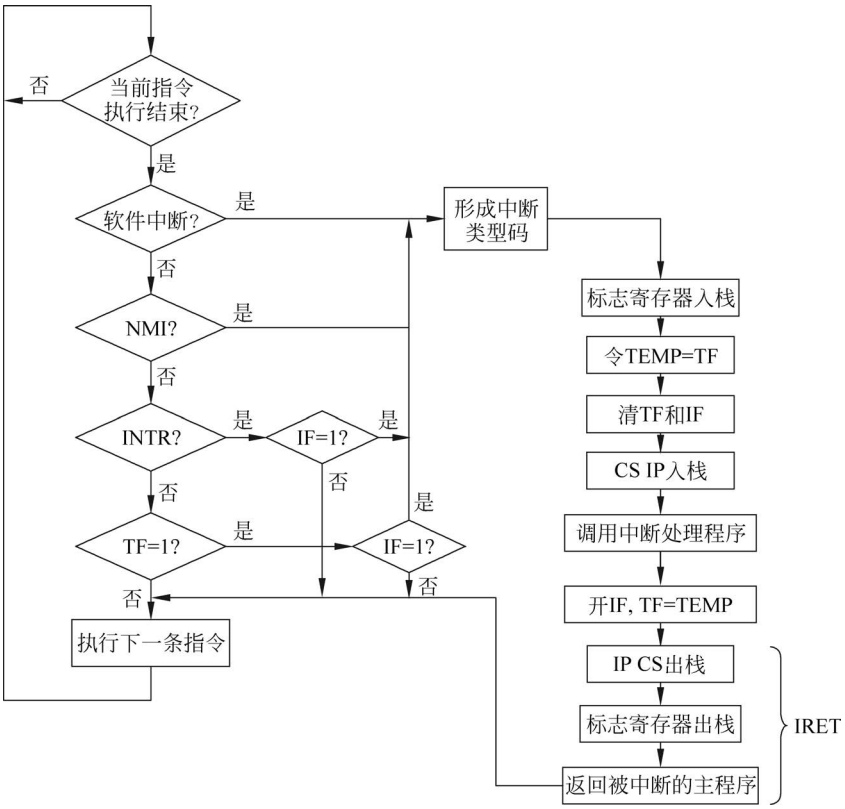


图 3.12　8086 中断系统的中断响应流程

(4) CPU 暂停执行当前程序,而转去执行相应的中断处理程序。在这个过程中,CPU 依次做如下处理。

① 将标志寄存器 FLAGS 的当前值压栈保存。

② 将标志寄存器中的中断允许标志位 IF 和单步标志位 TF 的值清零。将 IF 标志位清零是为了能够在中断响应过程中暂时屏蔽来自同级的其他中断源的请求;将 TF 标志位清零是为了避免以单步方式执行中断服务程序。

③ 保护断点,即将寄存器 CS 和 IP 的值压栈保存,以便在中断处理程序执行完以后,能够正确地返回到原程序中继续执行。

④ 根据得到的中断类型号,从中断向量表中找到相应的中断服务子程序的入口地址,将其段地址和段内偏移地址分别装入代码段寄存器 CS 和指令指针寄存器 IP 中,从而使 CPU 转去执行相应的中断服务程序。

⑤ CPU 执行中断服务程序。8086 中断系统中,在执行中断处理的具体内容前,首先将中断服务过程中需要用到的各个寄存器的当前值压入堆栈进行保护,以免在中断服务过程中改变了这些寄存器的值,而导致中断处理程序执行完后,不能正确返回源程序继续执行,这个过程称为**保护现场**。而在执行完中断处理的具体内容以后,再使用出栈指令,将先前保护的寄存器重新恢复为中断前的值,这就是**恢复现场**的工作。另外,在保护现场后,中断服务程序往往设置一条开中断指令,令 IF 标志位的值为 1,使在中断服务结束后,允许同级或

更高级别的其他中断源的中断请求进入即允许中断嵌套。

⑥ 返回断点,继续执行被中断的程序。中断服务程序执行的最后一条指令是IRET指令(中断返回指令),该指令将保存在堆栈中的断点值和标志寄存器FLAGS的值弹出,重新装入CS、IP和标志寄存器中,从而完成恢复断点的工作。

2. 非屏蔽中断和软件中断的执行过程

非屏蔽中断和软件中断的中断响应、中断处理及中断返回等过程,与可屏蔽中断基本相同,仅在中断请求和中断响应的条件上有所区别。

非屏蔽中断通常是由系统板上的一些硬件故障引起的中断。首先,硬件通过NMI信号线向CPU发出高电平的中断请求信号。CPU在每一个指令周期的最后一个时钟周期采样NMI信号线,若发现该信号线为高电平,即有非屏蔽中断(系统中的2号中断)请求发生,CPU立即予以响应,而不受中断允许标志位IF的影响。同时,非屏蔽级的中断类型号是系统事先约定好的,可以在CPU响应该级中断时,直接从2×4开始的连续4B单元中获取中断入口地址。随后的中断处理与中断返回过程,与可屏蔽中断的第(4)步~第(6)步相同。

软件中断的中断过程更为简单,当CPU执行INT n指令时,即可产生软件中断,CPU根据指令提供的中断类型码,从中断向量表中获取对应的中断入口地址,继而转至中断服务程序。随后的过程与可屏蔽中断的第(4)步~第(6)步基本相同,此处不再赘述。

图3.12所示为8086系统的中断响应流程。

3. 中断类型码的形成

由前述可知,各类中断的中断类型码是在中断响应阶段获得的,但CPU获取中断类型码的方法因中断源的不同而有所不同,具体有以下几种。

(1) 对各种内部中断(如被零除、溢出等),类型码是CPU根据异常类型(即系统内部事先定义的类型)在内部自动形成的。

(2) 对软件中断指令INT n,类型码由指令本身给出,也是在内部自动形成的。

(3) 对NMI,类型码被指定为2。因为它由系统内部事先定义,也不需要外部提供,所以本质上也是内部形成的。

(4) 对INTR,类型码在CPU的两个中断响应周期中由中断源通过中断控制器(如8259A)提供。

3.6 8086微处理器外部基本引脚与工作模式

3.6.1 8086系统总线结构

为提高系统性能、耐用性及适应性,8086 CPU设计为可工作在两种模式下,即最小模式和最大模式。**最小模式**用于由8086单一微处理器构成的小型系统;**最大模式**用于实现**多处理机系统**,其中,8086 CPU被称为**主处理器**,其他处理器被称为**协处理器**,如浮点运算协处理器8087、通道控制器8089。

为了减少芯片引脚个数,部分8086 CPU的外部引脚采用了复用技术。复用引脚分为**按时序复用**和**按模式复用**两种情况。对按时序复用的引脚,当CPU工作在不同的T周期时,这些引脚传送不同的信息;对按模式复用的引脚,则当CPU处于不同的工作模式时,这些引脚具有不同的功能含义。

8086 采用双列直插式(Double In line Package,DIP)封装,具有 40 条引脚,使用+5V 电源供电。时钟频率有 3 种:5MHz(8086)、8MHz(8086-1)和 10MHz(8086-2)。其引脚信号如图 3.13 所示,括号内为最大模式时的引脚名称。

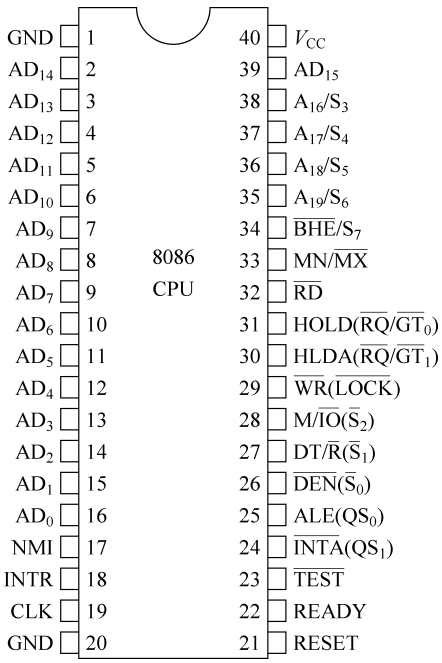


图 3.13　8086 CPU 引脚信号图

8086 CPU 的 40 条引脚信号按功能可分为 4 类:地址总线、数据总线、控制总线及其他(时钟与电源)。其中,控制总线又可分为公共控制信号,最小模式信号及最大模式信号。它的引脚信号定义如表 3.2 所示。

表 3.2　8086 引脚信号定义

信　号	名　称	功　能	引　脚　号	类　型
公用信号	AD_{15}～AD_0	地址/数据总线	39,2～16	双向、三态
	A_{19}/S_6～A_{16}/S_3	地址/状态总线	35～38	输出、三态
	$\overline{BHE}/S_7$	总线高允许/状态	34	输出、三态
	$MN/\overline{MX}$	最小/最大模式控制	33	输入
	$\overline{RD}$	读控制	32	输出、三态
	$\overline{TEST}$	等待测试控制	23	输入
	READY	等待状态控制	22	输入
	RESET	系统复位	21	输入
	NMI	不可屏蔽中断请求	17	输入
	INTR	可屏蔽中断请求	18	输入
	CLK	系统时钟	19	输入
	V_{CC}	+15V 电源	40	输入
	GND	接地	1,20	

续表

信　号	名　称	功　能	引　脚　号	类　型
最小模式信号 (MN/$\overline{MX}$=V_{CC})	HOLD	保持请求	31	输入
	HLDA	保持响应	30	输出
	$\overline{WR}$	写控制	29	输出、三态
	M/$\overline{IO}$	存储器输入输出控制	28	输出、三态
	DT/$\overline{R}$	数据发送/接收	27	输出、三态
	$\overline{DEN}$	数据允许	26	输出、三态
	ALE	地址锁存允许	25	输出
	$\overline{INTA}$	中断响应	24	输出
最大模式信号 (MN/$\overline{MX}$=GND)	$\overline{RQ}/\overline{GT}_{1,0}$	请求/允许总线访问控制	30,31	双向
	$\overline{LOCK}$	总线优先权锁定控制	29	输出、三态
	$\overline{S_0}$、$\overline{S_1}$、$\overline{S_2}$	总线周期状态	28～26	输出、三态
	QS_1、QS_0	指令队列状态	24,25	输出

3.6.2 8086 CPU的引脚信号

下面分别介绍4类不同的引脚信号功能。

1. 地址总线与数据总线

数据总线用来在CPU与内存储器或I/O设备之间交换信息,为双向、三态信号。地址总线由CPU发出,用来确定CPU要访问的内存单元或I/O端口的地址信号,为输出、三态信号。8086 CPU有20根地址总线和16根数据总线。在总线周期中,由于地址信息和数据信息在时间上不重叠,因此部分地址线与数据线共用一组引脚。状态信号用来指示CPU的状态信息,其中S_6～S_3和地址总线的高4位分时复用,S_7与$\overline{BHE}$分时复用。

(1) AD_{15}～AD_0(输入/输出,三态):分时复用地址/数据总线。AD_{15}～AD_0这16根信号线是分时复用的双重功能总线,数据总线D_{15}～D_0与地址总线的低16位A_{15}～A_0复用。在每个总线周期的第一个时钟周期T_1中,AD_{15}～AD_0用作地址总线的低16位A_{15}～A_0,给出内存单元或I/O端口的地址;在总线周期的其余时间(T_2、T_3、T_W和T_4状态),AD_{15}～AD_0作为数据总线D_{15}～D_0使用。

(2) A_{19}/S_6～A_{16}/S_3(输出,三态):分时复用的地址/状态复用信号。在每个总线周期的第一个时钟周期T_1中,A_{19}/S_6～A_{16}/S_3用作地址总线的高4位A_{19}～A_{16},在访问存储器时作为高4位地址,在访问I/O端口时,这4位置"0"(低电平)。在总线周期的其余时间(T_2、T_3、T_W和T_4状态),这4条信号线指示CPU的状态信息S_6～S_3。其中,S_6恒为低电平,表明8086 CPU当前正与总线相连;S_5反映标志寄存器中中断允许标志IF的当前值;而S_4和S_3组合起来指示当前正在使用的是哪个段寄存器,其编码如表3.3所示。

表3.3 S_4、S_3代码组合与当前段寄存器的关系

S_4	S_3	当前使用的段寄存器
0	0	附加段寄存器ES
0	1	堆栈段寄存器SS
1	0	存储器寻址时,使用代码段寄存器CS;对I/O端口或中断向量寻址时,不需要用段寄存器
1	1	数据段寄存器DS

（3）$\overline{BHE}/S_7$（输出，三态）：高 8 位数据总线允许/状态信号。它也是一个分时复用引脚。在总线周期的 T_1 状态，作为高 8 位数据总线允许信号，低电平有效。当 $\overline{BHE}=0$ 时，表示高 8 位数据总线 $AD_{15}\sim AD_8$ 上的数据有效；当 $\overline{BHE}=1$ 时，表示高 8 位数据总线 $AD_{15}\sim AD_8$ 上的数据无效，当前仅在数据总线 $AD_7\sim AD_0$ 上传送 8 位数据。而在 T_2、T_3、T_W、T_4 状态，此引脚输出状态信息 S_7，在 8086 微处理机系统中，S_7 没有定义。

$\overline{BHE}$ 和 AD_0 相配合访问存储器见表 3.1（3.2.1 节）。

2. 控制总线

控制总线中有 8 根引脚（第 24-31 号）在两种工作模式下定义的功能不同，其余的在两种工作模式下信号功能相同。

1）公用控制引脚信号

（1）$\overline{RD}$（输出、三态）：读信号，低电平有效。$\overline{RD}=0$ 时，表明 CPU 要进行一次内存或 I/O 端口的读操作，具体是对内存还是 I/O 端口进行读操作，决定于 $M/\overline{IO}$ 信号。

（2）READY（输入）：准备就绪信号。是来自存储器或 I/O 端口的应答，高电平有效。当 READY=1 时，表示内存或 I/O 端口准备就绪，马上可进行一次数据传输。CPU 在每个总线周期的 T_3 时钟周期开始处对 READY 信号采样，若检测到 READY 信号为低电平，则在 T_3 后插入一个 T_W 等待周期。在 T_W 时钟周期，CPU 继续对 READY 信号采样，若仍为低电平，就继续插入 T_W 等待周期，直到 READY 信号变为高电平，才进入 T_4 时钟周期，完成数据传送。

（3）$\overline{TEST}$（输入）：测试信号，低电平有效。用来支持构成多处理器系统，实现 8086 CPU 与协处理器之间同步协调的功能，只有当 CPU 执行 WAIT 指令时才使用。当 CPU 执行 WAIT 指令时，CPU 每隔 5 个时钟周期就对此引脚进行测试。若测试到该引脚为高电平，则 CPU 处于空转状态进行等待；若测试为低电平，则 CPU 结束等待状态，继续执行下一条指令。

（4）INTR（输入）：可屏蔽中断请求信号，高电平有效。当 INTR 为高电平时，表示外部有中断请求。CPU 在每条指令的最后时刻检测 INTR 引脚，若为高电平，且当前 CPU 允许中断（中断允许标志 IF=1），那么，CPU 就会在结束当前执行的指令后，响应中断请求，进入中断处理子程序。

（5）NMI（输入）：非屏蔽中断请求信号，上升沿有效。当 NMI 引脚输入一个由低到高的上升沿时，CPU 就会在结束当前执行的指令后，进入非屏蔽中断处理子程序。

（6）RESET（输入）：系统复位信号，高电平有效信号（至少保持 4 个时钟周期）。在 RESET 信号来到后，CPU 结束当前操作，并将处理器中的寄存器 FLAGS、IP、DS、SS、ES 及指令队列清零，将 CS 设置为 0FFFFH。当复位信号变为低电平时，CPU 从 FFFF0H 开始执行程序，实现系统的重启动过程。系统加电或操作员按下 RESET 键后会产生 RESET 信号。

（7）$MN/\overline{MX}$（输入）：工作模式控制信号。决定 CPU 工作在最小模式或最大模式。此引脚接＋5V 电源时，CPU 处于最小模式，接地时，CPU 处于最大模式。

2）最小模式下的控制引脚信号

（1）$\overline{INTA}$（输出）。

CPU 发向中断控制器的中断响应信号，低电平有效。在两个连续的总线周期输出两个

低电平信号,第一个低电平用来通知外设CPU,准备响应它的中断请求,在第二个低电平期间,外设通过数据总线送入它的中断类型码,并由CPU读取,以便取得相应中断服务程序的入口地址。

(2) ALE(输出)。

地址锁存允许信号,高电平有效。当ALE信号有效时,表示地址线上的地址信息有效,利用它的下降沿把地址信息和$\overline{BHE}$信号锁存在地址锁存器。ALE不能浮空。

(3) $\overline{DEN}$(输出、三态)。

数据允许信号,低电平有效。当$\overline{DEN}$信号有效时,表示CPU准备好接收和发送数据。在最小模式中,$\overline{DEN}$信号就是总线收发器的选通信号,总线收发器将$\overline{DEN}$作为输出允许信号。

(4) DT/$\overline{R}$(输出、三态)。

数据发送/接收信号,表示CPU是接收数据(低电平),还是发送数据(高电平),用于控制总线收发器数据传送的方向。

(5) M/$\overline{IO}$(输出,三态)。

存储器/输入、输出控制信号,用于区分是访问存储器(高电平),还是访问I/O端口(低电平)。

(6) $\overline{WR}$(输出)。

写信号,低电平有效。$\overline{WR}=0$时,表明CPU正在执行向存储器或I/O端口的输出操作。

(7) HOLD(输入)。

总线请求信号,高电平有效,是系统中其他总线主设备向CPU提出总线请求的输入信号。CPU让出总线控制权直到这个信号撤销后才恢复对总线的控制权。

(8) HLDA(输出)。

总线响应信号,高电平有效,是CPU对系统中其他总线主控设备请求总线使用权的应答信号。当CPU让出总线使用权时,就发出这个信号,并使CPU所有具有三态的引脚处于高阻状态,与外部隔离。

在最小模式下,8086 CPU直接产生全部总线控制信号(DT/$\overline{R}$、$\overline{DEN}$、$\overline{ALE}$、M/$\overline{IO}$)和命令输出信号($\overline{RD}$、$\overline{WR}$或$\overline{INTA}$),并提供请求访问总线的逻辑信号HLDA。当总线主控设备(例如DMA控制器Intel 8257/Intel 8237)请求总线控制权时,它向8086发送一个总线请求信号HOLD,如果8086 CPU响应HOLD请求,则8086 CPU输出响应信号HLDA,通知DMA控制器可以使用系统总线,同时使8086 CPU的地址总线、数据总线、$\overline{BHE}$信号以及有关的总线控制信号和命令输出信号处于高阻状态。此外,地址锁存器和数据收发器的输出也处于高阻状态。这样,8086 CPU不再控制总线,一直保持到HOLD信号变为无效,8086 CPU重新获得总线控制权为止。DMA控制器接收到来自CPU的响应信号后,掌握系统总线控制权,进行数据传送。当DMA控制器完成传送任务时,撤销发向CPU的总线请求信号,CPU重新获得对系统总线的控制权。

3) 最大模式下的控制引脚信号

(1) $\overline{S}_2$、$\overline{S}_1$、$\overline{S}_0$(输出,三态)。

总线周期状态信号。这三位的组合表示当前总线周期的操作类型,如表3.4所示。

表 3.4　$\overline{S}_2$、$\overline{S}_1$、$\overline{S}_0$ 译码表

总线状态信号			CPU 状态
$\overline{S}_2$	$\overline{S}_1$	$\overline{S}_0$	
0	0	0	中断状态
0	0	1	读 I/O 端口
0	1	0	写 I/O 端口，超前写 I/O 端口
0	1	1	暂停
1	0	0	取指令
1	0	1	读存储器(数据)
1	1	0	写存储器，超前写存储器
1	1	1	无效

当 $\overline{S}_2$、$\overline{S}_1$、$\overline{S}_0$ 中任一个为低电平时，都对应某一种总线操作，此时称为**有源状态**。而当一个总线周期即将结束(T_3 期间或 T_W 周期)，另一个总线周期尚未开始，并且 READY 信号也为高电平时，$\overline{S}_2$、$\overline{S}_1$、$\overline{S}_0$ 都变为高电平，此时称为**无源状态**。在前一个总线周期的 T4 时钟周期时，只要 $\overline{S}_2$、$\overline{S}_1$、$\overline{S}_0$ 中有一个变为低电平，就意味着即将开始一个新的总线周期。而在 T_3 或 T_W 期间返回无效状态，则表示一个总线周期的结束。在 DMA(直接存储器存取)方式下，$\overline{S}_2$、$\overline{S}_1$、$\overline{S}_0$ 处于高阻状态。

(2) QS_1、QS_0(输出)。

指令队列状态信号，用于指示 8086 内部 BIU 中指令队列的状态，以便外部协处理器跟踪 8086CPU 内部指令序列。QS_1 和 QS_0 表示的状态如表 3.5 所示。

表 3.5　QS_1、QS_0 组合与指令队列的状态

QS_1	QS_0	队列状态信号的含义
0	0	无操作，未从队列中取指令
0	1	从队列中取出当前指令的第一字节
1	0	队列空，由于执行转移指令，队列重新装填
1	1	从队列中取出指令的后继字节

外部逻辑通过监视总线状态和队列状态，可以模拟 CPU 的指令执行过程，并确定当前正在执行哪一条指令。有了这个功能，8086 才能告诉协处理器何时准备执行指令。

(3) $\overline{RQ}/\overline{GT_0}$、$\overline{RQ}/\overline{GT_1}$(双向)。

总线请求信号/总线请求响应信号，低电平有效。这两个信号是为多处理机应用而设计的，用于对总线控制权的请求和应答，其特点是请求和允许功能用一根信号线来实现，每一个引脚都可代替最小模式下 HOLD/HLDA 两个引脚的功能。这两个引脚可同时接两个协处理器，$\overline{RQ}/\overline{GT_0}$ 的优先级高于 $\overline{RQ}/\overline{GT_1}$。

总线访问的请求/允许时序分为三个阶段：请求、允许和释放。首先是协处理器向 8086 输出 $\overline{RQ}$ 请求使用总线，然后在 8086 CPU 的 T_4 或下一个总线周期的 T_1 期间，CPU 输出一个宽度为一个时钟周期的脉冲信号 $\overline{GT_0}$ 给请求总线的协处理器，作为总线响应信号；从下一个时钟周期开始，CPU 释放总线。当协处理器使用总线结束时，再输出一个宽度为一

个时钟周期的脉冲信号 $\overline{RQ}$ 给 CPU,表示总线使用结束,从下一个时钟周期开始,CPU 又控制总线。

(4) $\overline{LOCK}$(输出、三态)。

总线封锁信号,低电平有效。$\overline{LOCK}$ 信号用来封锁外部处理器的总线请求。当 $\overline{LOCK}$=0 时,表明 CPU 不允许其他总线主控设备占用总线。$\overline{LOCK}$ 信号通过指令在程序中设置。若一条指令加上前缀 LOCK,则 8086 在执行该指令期间,$\overline{LOCK}$ 引脚输出低电平并保持到该指令执行结束,以保证该指令在执行期间不会被外部处理器的总线请求打断。

3. 其他信号

(1) CLK(输入):时钟信号。为处理器提供基本的定时脉冲和内部的工作频率。8086 CPU 要求时钟信号的占空比(正脉冲与整个周期的比值)为 33%,即 1/3 周期高电平,2/3 周期低电平。

(2) V_{CC}(输入):电源。要求接正电压(+5±0.5V)。

(3) GND(输入):地线。8086 CPU 有两条接地线。

3.6.3 8086 CPU 的最小工作模式

8086 CPU 的 MN/$\overline{MX}$ 引脚接+5V 电源时,8086 CPU 工作在最小模式下,是单一微处理器构成的小型系统。

总线是计算机各种功能部件之间传送信息的公共通路,连接到总线上的功能模块有主动和被动两种方式。**主设备**可以启动一个总线周期,而**从设备**只能响应主设备的请求。某一时刻总线上只能有一个主设备占用总线。CPU 在不同的时间可以用作主设备,也可用作从设备;而存储器则只能用作从设备。

在最小模式中,作为单处理器的 8086 通常控制着系统总线,但也允许系统中的其他设备占用总线。例如直接内存访问(DMA)是一种完全由硬件执行 I/O 交换的工作方式。在这种方式中,DMA 控制器从 CPU 完全接管对总线的控制,数据交换不经过 CPU,而直接在内存和 I/O 设备之间进行,一般用于高速传送成组数据。DMA 控制器在传送数据时,需要使用总线,这时 DMA 控制器向 8086 发送一个总线请求信号,在 CPU 允许并响应的情况下,DMA 控制器获得总线控制权,使用完后,又将总线控制权交还给 CPU。

8086 CPU 在最小模式下的典型配置如图 3.14 所示。

8086 的最小模式具有以下几个特点。

(1) MN/$\overline{MX}$ 引脚接+5V 电源,决定了 CPU 工作在最小模式下。

(2) 使用一片 8284,作为时钟信号发生器。

(3) 使用 3 片 74LS373 或 Intel 8282,作为地址锁存器(总线锁存器)。

(4) 当系统中所连的存储器和外设端口较多时,需要增强数据总线的驱动能力,用两片 74LS245 或 8286/8287 作为总线收发器(数据收发器)。

1. 时钟发生器

8284 是用于 8086 系统的时钟发生器芯片,它为 8086 及其他外设芯片提供恒定的时钟信号,对准备信号(READY)及复位信号(RESET)进行同步。外界控制信号 RDY 及 $\overline{RES}$ 信号可以在任何时刻到来,8284 能把它们同步在时钟下降沿时输出 READY 及 RESET 信号到 8086 CPU。

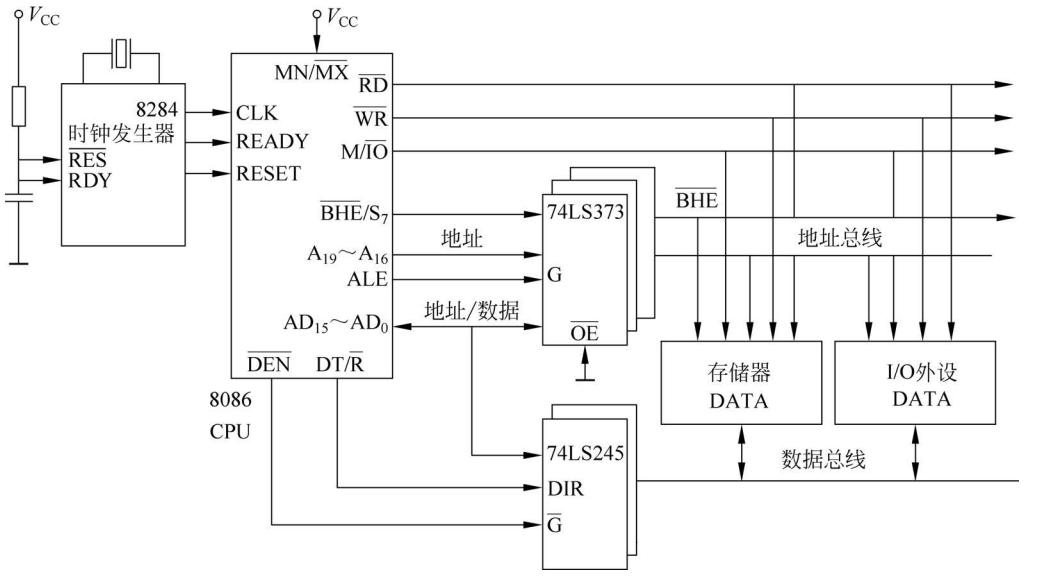


图 3.14　最小模式下的 8086 系统配置

8284 由时钟信号发生器、复位生成电路和就绪控制电路三部分组成，其引脚图如图 3.15 所示。

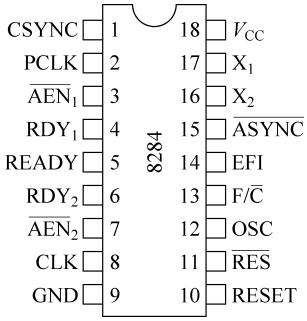


图 3.15　8284 的引脚图

时钟信号发生器产生恒定的时钟信号，复位信号发生电路产生系统复位信号 RESET，就绪信号控制电路用于对存储器或 I/O 接口产生的准备好信号 READY 进行同步。

8284 的典型连接如图 3.16 所示。振荡源一般采用晶体振荡器，在 X_1 与 X_2 引脚间接上晶体，由晶体振荡器产生时钟信号。另外也可以由 EFI 引脚加入的外接脉冲发生器作为振荡源，产生时钟信号。此时 8284 的 F/$\overline{C}$ 引脚应接高电平。8284 输出的时钟频率为振荡源频率的 1/3。

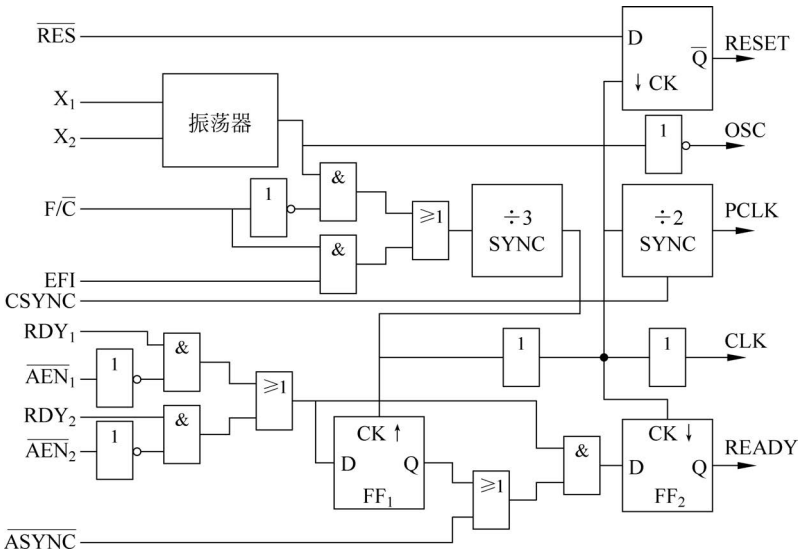


图 3.16　8284 的典型连接

2. 地址锁存器

8086 CPU 的地址/数据总线是复用的,$\overline{BHE}$ 和 S_7 也是复用的,在总线周期的 T_1 时钟周期传送地址信息和 $\overline{BHE}$ 信号,而在其他时钟周期传送数据、状态信息。为避免丢失地址信息和 $\overline{BHE}$ 信号,需在它们失效前将其锁存至地址锁存器,为地址总线提供存储器或 I/O 端口地址。当 ALE 信号有效时,表示地址线上的地址信息有效,ALE 信号的下降沿把地址信息和 $\overline{BHE}$ 信号锁存在地址锁存器。在总线周期的其余时间(T_2、T_3、T_W 和 T_4 状态),复用的地址/数据总线作为数据总线使用,实现 CPU 对存储器和 I/O 设备的读/写操作。

常用的地址锁存器芯片有 74LS373、74LS273、Intel 8282 和 Intel 8283 等。系统配置图中的 3 片 74LS373 芯片用来锁存地址/数据总线 AD_{15}～AD_0 中的地址信息、地址/状态总线 A_{19}～A_{16}/S_6～S_3 中的地址信息及 $\overline{BHE}/S_7$ 中的 $\overline{BHE}$ 信息。其中每片 74LS373 芯片锁存 8 位信息。

3. 总线收发器

当一个系统中所含的外设接口较多时,数据总线上需要有发送器和接收器来增加驱动能力。发送器和接收器简称为总线收发器或总线驱动器。

总线收发器芯片 74LS245 是 8 位的。所以在 8086 系统中,需要用两片 74LS245 对 AD_{15}～AD_0 中的数据信息进行缓冲和驱动。注意该芯片在 8086 总线周期的第二个时钟周期 T_2 开始工作,因为 T_1 周期时 AD_{15}～AD_0 上输出的是地址信息。

常用的总线收发器芯片有 74LS245、Intel 8286 和 Intel 8287 等。

3.6.4 8086 CPU 的最大工作模式

若微机系统中有两个或多个同时执行指令的处理器,就构成**多处理器系统**,可以有效地提高整个系统的性能。增加的处理器可以是 8086 处理器,也可以是**协处理器**。协处理器只是协助主处理器完成某些辅助工作。和 8086 配套使用的协处理器有两个:一个是用于数值计算的协处理器 8087,通过硬件实现高精度整数浮点运算;另一个是专用于输入/输出操作的协处理器 8089,8089 有一套专门用于输入/输出操作的指令系统,可以直接为输入/输出设备服务。增加协处理器后,使得浮点运算和输入/输出操作不再占用 8086 CPU 的时间,从而大大提高了系统的运行效率。

多处理器系统必须解决多处理器对系统总线的争用问题,以及处理器之间的通信问题。因为多个处理器通过公共系统总线共享存储器和 I/O 设备,所以必须增加相应的逻辑电路,以确保每次只有一个处理器取回执行结果,必须提供一种明确的方法来解决两个处理器之间的通信。8086 CPU 的最大工作模式就是专门为实现多处理器系统而设计的。在这种方式下,8086 CPU 不直接提供用于存储器或 I/O 端口访问的读/写命令等控制信号,而是由总线控制器 8288 产生总线命令和控制信号,对存储器和 I/O 端口进行读/写控制。

当 8086 CPU 的 MN/$\overline{MX}$ 引脚接地时,8086 CPU 工作于最大模式,用于构成多处理机系统,图 3.17 所示为最大模式下 8086 系统配置。可以看出,与最小模式下 8086 系统配置相比较,最大模式系统增加了一片专用的总线控制器 8288。

8086 的最大模式具有以下几个特点。

(1) MN/$\overline{MX}$ 端接地,决定了 CPU 工作在最大模式下。

(2) 使用一片 8284,作为系统时钟。

(3) 使用三片 8282 或 74LS373,作为地址锁存器。

(4) 使用两片 8286/8287,作为总线收发器。

(5) 使用一片 8288,作为总线控制器。

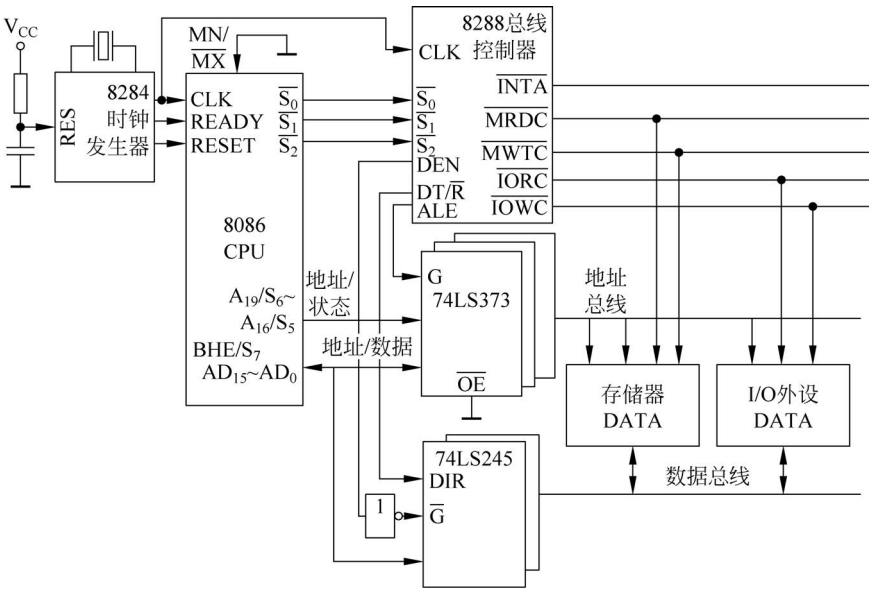
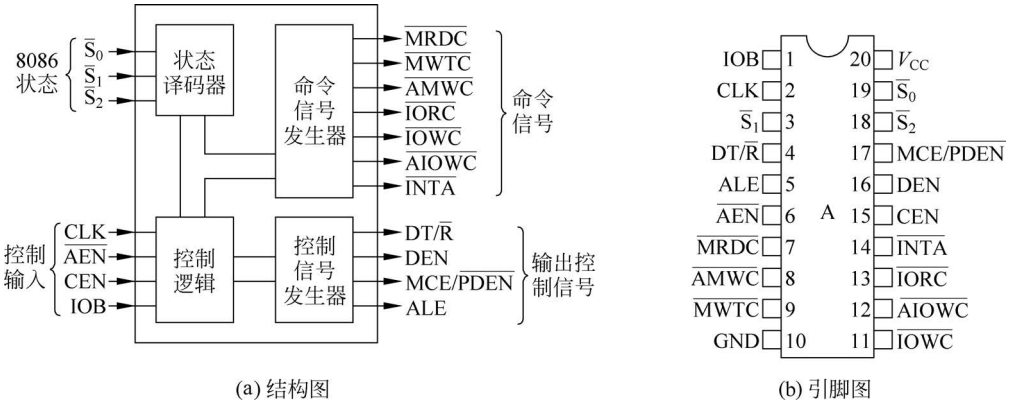


图 3.17　最大模式下的 8086 系统配置

1. 总线控制器

最大模式系统中,一般包含两个或多个处理器,这就需要解决主处理器和协处理器之间的协调工作,以及对系统总线的共享控制问题,8288 总线控制器就起了这个作用。它根据 8086 在执行指令时提供的总线周期状态信号 $\overline{S}_2$、$\overline{S}_1$、$\overline{S}_0$ 来建立控制时序,与输入控制信号 IOB、$\overline{AEN}$、CEN 和 CLK 相配合,产生读/写控制命令。通过 8288 可以提供灵活多变的系统配置,以实现最佳的系统性能。

8288 由状态译码器、命令信号发生器、控制信号发生器及控制逻辑组成,其内部结构和引脚信号如图 3.18 所示。



(a) 结构图　　(b) 引脚图

图 3.18　8288 结构框图与引脚信号

8288接收8086的总线周期状态信号$\overline{S}_2$、$\overline{S}_1$、$\overline{S}_0$,确定当前总线周期的操作类型,译码产生相应的存储器读/写命令、I/O端口读/写命令及中断响应信号。$\overline{S}_2$、$\overline{S}_1$、$\overline{S}_0$的代码组合对应的总线操作类型如表3.6所示。

表3.6 $\overline{S}_2$、$\overline{S}_1$、$\overline{S}_0$译码表

总线状态信号			CPU状态	8288命令输出
$\overline{S}_2$	$\overline{S}_1$	$\overline{S}_0$		
0	0	0	中断状态	$\overline{INTA}$
0	0	1	读I/O端口	$\overline{IORC}$
0	1	0	写I/O端口,超前写I/O端口	$\overline{IOWC}$,$\overline{AIOWC}$
0	1	1	暂停	无
1	0	0	取指令	$\overline{MRDC}$
1	0	1	读存储器(数据)	$\overline{MRDC}$
1	1	0	写存储器,超前写存储器	$\overline{MWTC}$,$\overline{AMWC}$
1	1	1	无效	无

8288总线控制器产生的总线控制命令如下。

(1) $\overline{IORC}$、$\overline{IOWC}$:I/O读、写命令。

(2) $\overline{MRDC}$、$\overline{MWTC}$:存储器读、写命令。

(3) $\overline{AIOWC}$和$\overline{AMWC}$:超前命令。这两个超前命令比$\overline{IOWC}$和$\overline{MWTC}$出现时间早一个时钟周期,在需要提前发出写命令的场合,可以选用这两个超前信号,从而能够在一定程度上避免微处理器进入等待状态。

(4) $\overline{INTA}$:中断响应命令。

(5) ALE:地址锁存允许信号。用于将地址选通到地址锁存器,高电平有效,在下降沿锁存。

(6) DEN:数据使能信号。DEN为高电平时,接通数据收发器。

(7) DT/$\overline{R}$:数据发送/接收信号。用来控制数据收发器的传送方向,DT/$\overline{R}$=1为发送状态;DT/$\overline{R}$=0为接收状态。

(8) MCE/$\overline{PDEN}$:主设备使能/外设数据允许命令。这是一条双重功能的控制线,当8288工作于系统总线方式时,用作主控级联允许信号MCE,在中断响应周期的T_1状态时有效,控制主8259A向从8259A输出级联地址;当8288工作于I/O总线方式时,用作外设数据允许信号$\overline{PDEN}$,控制外围设备通过I/O总线传送数据。

8288的工作受输入控制信号的控制,这些信号是IOB、$\overline{AEN}$、CEN和CLK。

(1) IOB:输入/输出总线方式。8288既可控制系统总线,又可以控制I/O总线。当IOB=0时,8288处于**系统总线方式**;当IOB=1时,8288处于I/O **总线工作方式**,8288仅用来控制I/O总线。

(2) $\overline{AEN}$:地址使能。由总线仲裁器8289输入,是支持多总线结构的同步控制信号。当$\overline{AEN}$=1时,8288各种命令无效,呈高阻态;当$\overline{AEN}$=0时,在系统总线方式下,至少在$\overline{AEN}$有效后115ns,8288才能输出命令,这段时间进行总线切换;在I/O总线方式下,$\overline{AEN}$不起作用,不影响I/O命令的发出。

(3) CEN：命令使能。当有多片 8288 协同工作时起片选作用。当 CEN＝1 时，允许该 8288 发出全部控制命令；当 CEN＝0 时，禁止该 8288 发出总线控制信号，同时使 DEN、$\overline{\text{PDEN}}$ 呈高阻状态。

(4) CLK：时钟信号。8288 产生命令和控制信号输出时，由 CLK 决定它们的定时关系。通常由微机的系统时钟提供。

2. 时钟发生器、地址锁存器和总线收发器

由图 3.17 可知，在最大配置中，这 3 种部件的工作与最小模式下配置相同。

3. 需要说明的问题

(1) 8086 CPU 在最小模式下的 HOLD 和 HLDA 引脚在最大模式下成为 $\overline{\text{RQ}}/\overline{\text{GT}_0}$ 和 $\overline{\text{RQ}}/\overline{\text{GT}_1}$ 信号线，这两条引脚通常同 8087(协处理器)或 8089(I/O 处理器)相连接，用于 8086 同协处理器之间传送总线请求与总线应答信号。

(2) 当系统为具有两个以上主 CPU 的多处理器系统时，必须配备总线仲裁器 8289，与 8288 相配合确定总线使用权的分配，从而保证系统中的各个处理器同步地进行工作，实现总线共享。

3.7　8086 微处理器的时序

所谓时序，顾名思义，就是时间的先后顺序。系统为完成某一项操作，必将涉及一系列部件的协调动作，并且每一部件动作的时间长短也有严格的限制，这就必须采用一定的方法对它们进行定时控制，这就是下面所要讨论的时序问题。前面已经介绍过，在微机系统中通常有三级时序，分别为时钟周期、总线周期和指令周期。

3.7.1　系统的复位与启动

8086 CPU 的 RESET 引脚用来启动或重启动系统。当 8086 在 RESET 引脚上检测到一个脉冲的上升沿时，它将停止正在进行的所有操作，处于初始化状态，直到 RESET 信号变低。因此，通过在 CPU 的 RESET 引脚上加正脉冲，可完成系统的启动和重启动。8086 CPU 要求加在 RESET 引脚上的复位正脉冲信号宽度至少为 4 个时钟周期，如果是初次加电启动，则要求宽度不少于 50μs。复位操作时序如图 3.19 所示。

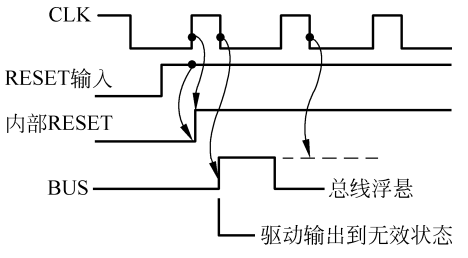


图 3.19　复位操作时序

图 3.19 中的 RESET 输入是引脚信号，CPU 内部是用时钟脉冲 CLK 来同步外部的复位信号的。当外部的复位信号到来时，经 8284 同步，在 RESET 输入信号到来后的 CLK 第一个上升沿形成内部 RESET 信号送给 CPU，CPU 就进入内部 RESET 过程。到本次时钟周期的下降沿，所有的三态输出线都被设置为无效状态，再到下一个时钟周期的上升沿，所有的三态输出线都被设置为高阻状态，直到 RESET 信号回复低电平。三态输出线包括 $AD_{15}\sim AD_0$、$A_{19}/S_6\sim A_{16}/S_3$、$\overline{\text{BHE}}/S_7$、$M/\overline{IO}(\overline{S}_2)$、$DT/\overline{R}(\overline{S}_1)$、$\overline{\text{DEN}}(\overline{S}_0)$、$\overline{\text{WR}}(\overline{\text{LOCK}})$、$\overline{\text{RD}}$ 和 $\overline{\text{INTA}}$。其他输出线只被设置为无效，而不设置

高阻,包括最小模式时的 ALE、HLDA 及最大模式时的 $\overline{RQ}/\overline{GT_1}$、$\overline{RQ}/\overline{GT_0}$、$QS_1$、$QS_0$。

8086 CPU 复位时,结束原有的操作和状态,维持在复位状态,各内部寄存器及指令队列被设置为初始值,如表 3.7 所示。

表 3.7 复位时 CPU 的初始化状态

寄存器	内容	寄存器	内容
标志寄存器	清零	堆栈段寄存器 SS	0000H
指令指针寄存器 IP	0000H	ES 附加	0000H
代码段寄存器 CS	FFFFH	指令队列	空
数据段寄存器 DS	0000H	其他寄存器	0000H

由表 3.6 可以看出,CPU 复位时,代码段寄存器 CS 被初始化为 FFFFH,而指令指针寄存器被初始化为 0000H。因此,当 CPU 复位完成,再重新启动时,就会从主存地址为 FFFF0H 的位置开始执行指令。通常在这个地址单元存放着一条无条件转移指令,将程序转移到系统程序的入口处。这样,一旦系统复位或重新启动,就会重新引导系统程序。

复位信号 RESET 从高到低的跳变会触发 CPU 内部的一个复位逻辑电路,经过 7 个时钟周期后,CPU 就被重新启动而恢复正常工作。

3.7.2 最小模式系统总线周期时序

1. 读/写总线周期

读/写总线周期指 CPU 通过外部总线完成从存储器或外设端口读/写一次数据所需要的时钟周期数。

8086 CPU 读/写总线周期时序如图 3.20(a)和图 3.20(b)所示。

各状态所完成的操作描述如下。

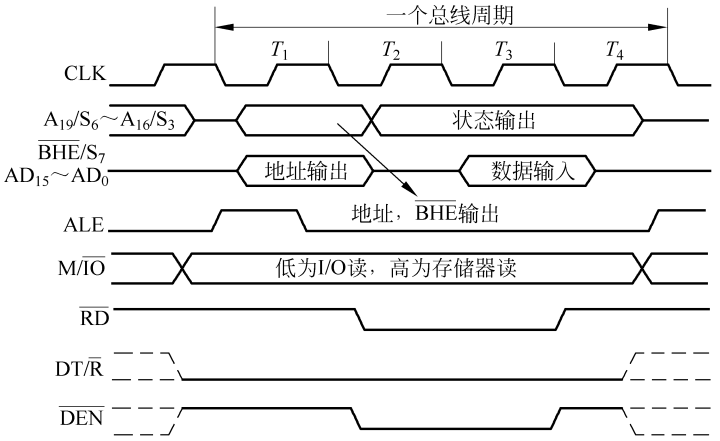
(1) T_1 状态。$M/\overline{IO}$ 信号在 T_1 状态变为有效。若为高电平,则表明是从存储器读取;若为低电平,则表明是从 I/O 端口读取。并且这个有效电平一直持续到本次总线周期结束,即 T_4 状态。

同时,CPU 在 T_1 状态通过 $A_{19}/S_6 \sim A_{16}/S_3$ 和 $AD_{15} \sim AD_0$ 发出访问外设或存储器的 20 位地址信息,并输出 $\overline{BHE}$ 有效信号,表示高 8 位数据线上的信息可以使用。

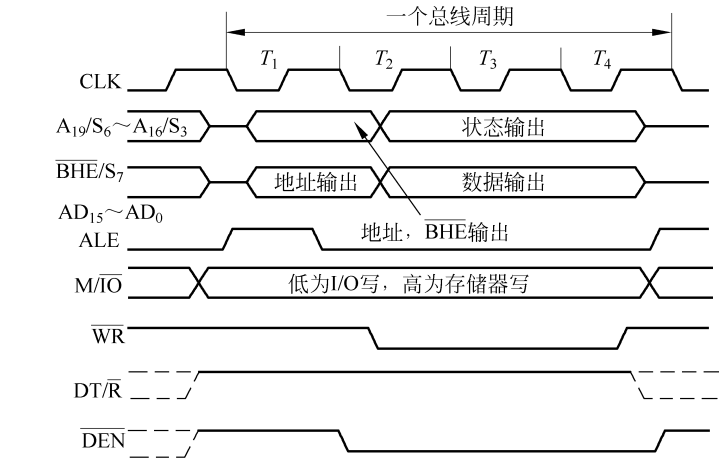
总线上的地址信息在 T_1 状态结束之前必须进行锁存,地址锁存器将 ALE 作为它的锁存允许信号,所以在 T_1 状态,CPU 发出一个 ALE 正脉冲信号,地址锁存器利用 ALE 的下降沿锁存地址信息。

如果系统中接有数据总线收发器,就要用到 $DT/\overline{R}$ 和 $\overline{DEN}$ 控制信号,$\overline{DEN}$ 用来选通收发器,$DT/\overline{R}$ 用来决定收发器的数据传送方向。如果是读操作在 T_1 状态,$DT/\overline{R}$ 变为低电平有效,表明本次总线周期让数据总线收发器接收数据;如果是写操作 $DT/\overline{R}$ 变为高电平有效,表示本次总线周期让数据总线收发器发送数据。

(2) T_2 状态。总线上撤销地址信息 $A_{19}/S_6 \sim A_{16}/S_3$,引脚输出状态信息 $S_6 \sim S_3$。$AD_{15} \sim AD_0$ 呈高阻状态,为传送数据做准备。



(a) 读总线周期



(b) 写总线周期

图 3.20　最小模式系统总线读/写操作时序

若进行读操作，则 CPU 在 T_2 状态输出 $\overline{RD}$ 低电平有效信号，否则，进行写操作，CPU 在 T_2 状态输出 $\overline{WR}$ 低电平有效信号，并立即往数据总线 $AD_{15} \sim AD_0$ 上发出向外设或存储器写入的数据。

$\overline{DEN}$ 信号也在 T_2 状态变为低电平有效状态，选通总线收发器工作。

(3) T_3 状态。CPU 继续提供状态信息，并维持 $\overline{RD}$ 或 $\overline{WR}$、M/$\overline{IO}$、DT/$\overline{R}$ 及 $\overline{DEN}$ 为有效电平。如果外设或存储器速度较快，则应在 T_3 状态往数据总线 $AD_{15} \sim AD_0$ 上送入 CPU 读取的数据信息。

(4) T_W 状态。如果所用外设或存储器速度较慢，不能配合 CPU 的工作，就需要在 T_3 和 T_4 之间插入一个或几个 T_W 等待状态。系统中的 READY 电路在 T_3 状态后生成 READY 信号，并经 8284 系统时钟电路同步后加到 CPU 的 READY 引脚上，CPU 在 T_3 状态开始时采样 READY 信号，若为低电平，则表明外设或存储器没有准备好，那么，就在 T_3 后插入 T_W 状态，而且在每个 T_W 状态的上升沿，CPU 都将检测 READY 信号，直至检测到

READY 高电平信号后,才结束 T_W 状态。在最后一个 T_W 状态中,CPU 读取的数据信息已经稳定在数据总线上。

(5) T_4 状态。若为读总线周期,则在 T_4 状态和前一个状态交界的下降沿处,CPU 读入已经稳定出现在数据总线上的数据,各控制信号和状态信号变为无效,$\overline{DEN}$ 信号进入高电平,关闭总线收发器 8288;若为写周期,则 CPU 认为外设或存储器已取走了数据,从而撤销数据信息。

2. 总线请求

在最小模式系统中,如果 CPU 以外的其他模块(如 DMA 控制器)需要占用总线,就会向 CPU 提出请求。CPU 接收到请求后,如同意让出总线使用权,就会向请求模块发出响应信号,由请求模块占用总线,请求模块使用完总线后再将总线控制权还给 CPU,这一过程称为**总线请求**。8086 CPU 为此专门设置了一组控制线 HOLD 和 HLDA。

CPU 在每个时钟的上升沿处都会检测 HOLD 信号。如果检测到高电平,就表明有模块提出总线保持请求,如果此时 CPU 允许响应,就会在本次总线周期的 T_4 周期或空闲周期 T_1 的下一个时钟周期发出 HLDA 响应信号,并使所有三态输出线都变为高阻状态(包括地址/数据线、地址/状态线及控制线 $\overline{RD}$、$\overline{WR}$、$\overline{INTA}$、M/$\overline{IO}$、$\overline{DEN}$、DT/$\overline{R}$),让出总线控制权,进入总线保持阶段。直到该模块使用完总线,使 HOLD 恢复低电平状态,CPU 随之将 HLDA 也变为低电平,才又收回总线控制权,其时序如图 3.21 所示。

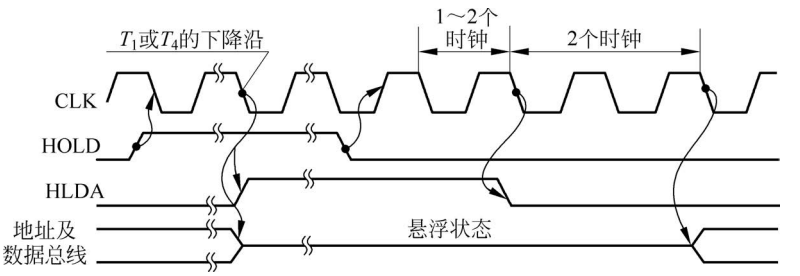


图 3.21 最小模式系统中总线保持请求与响应时序

在总线请求保持期间,CPU 继续执行已取到指令队列中的指令(与 DMA 并行操作),直到指令需要使用总线或指令队列为空为止。

3.7.3 最大模式系统总线周期时序

在最大模式系统中,8086 CPU 所有的对总线进行读/写操作的控制信号和命令信号都由总线控制器 8288 提供。

1. 读总线周期

最大模式系统读总线周期时序如图 3.22 所示。

在读总线周期中,8288 提供的总线操作命令信号有 ALE(地址锁存允许)、DT/$\overline{R}$(数据发送/接收)、$\overline{DEN}$(数据使能)、$\overline{MRDC}$(读存储器)、$\overline{IORC}$(读 I/O 端口)等。8288 对存储器和 I/O 端口的数据读取用两个不同的命令加以区别,不同于最小工作模式的用 M/$\overline{IO}$ 的不同状态区分。

各个时钟周期所完成的操作描述如下。

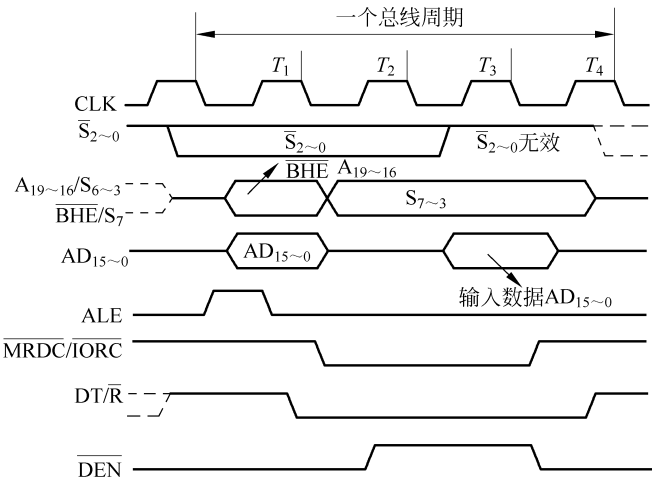


图 3.22　最大模式系统读总线周期时序

(1) T_1 状态。CPU 送出 20 位地址信息，从引脚送出 $\overline{BHE}$ 低电平有效信号。8288 送出 ALE 地址锁存允许的正脉冲信号；提供给数据总线收发器方向控制信号 DT/$\overline{R}$，使其为低电平有效。

(2) T_2 状态。CPU 撤销地址信息，使地址/数据线成为高阻状态，为数据传输做准备，而 $\overline{BHE}/S_7$ 和地址/状态线送出总线状态信息 $S_7\sim S_3$，并将该状态信息保持到 T_4 状态。8288 在 T_2 状态期间送出存储器或 I/O 端口读命令 $\overline{MRDC}/\overline{IORC}$，使其变为低电平有效，并且 8288 还在 T_2 上升沿给数据总线收发器发出高电平有效的选通信号 $\overline{DEN}$，允许数据通过总线收发器。

(3) T_3 状态。如果所访问的存储器或外设的存取速度较快，能在时序上满足基本总线周期的时序要求，就不必在 T_3 状态后插入 T_W 等待状态。这时总线状态信息 $\overline{S}_2$、$\overline{S}_1$、$\overline{S}_0$ 都转变为高电平，进入无源状态，并将这个无源状态从 T_3 状态一直持续到 T_4 状态。一旦进入无源状态，就意味着不久就可以启动下一个新的总线周期。若存储器或外设存取速度较慢，不能满足定时要求，则与最小模式系统一样，需要在 T_3 与 T_4 之间插入一个或几个 T_W 状态。

(4) T_4 状态。总线上的数据信息消失，状态信号 $S_7\sim S_3$ 变为高阻。$\overline{S}_2$、$\overline{S}_1$、$\overline{S}_0$ 则按下一个总线周期的操作类型，产生相应的电平变化。

2. 写总线周期

最大模式系统写总线周期时序如图 3.23 所示。

在写总线周期中，8288 提供的总线操作命令信号有 ALE(地址锁存允许)、DT/$\overline{R}$(数据发送/接收)、$\overline{DEN}$(数据使能)、$\overline{MWTC}$(写存储器)、$\overline{IOWC}$(写 I/O 端口)。8288 提供的存储器写命令和 I/O 端口写命令比 8086 CPU 的 $\overline{WR}$ 命令晚一个时钟周期，因为要保证 CPU 输出的数据稳定出现在数据总线上后，8288 才可以发出存储器或 I/O 端口写命令。当 $\overline{MWTC}$ 或 $\overline{IOWC}$ 不能满足定时要求时，可使用 8288 提供的另两个超前写命令 $\overline{AMWC}$(超前写存储器)和 $\overline{AIOWC}$(超前 I/O 写端口)，它们比 $\overline{MWTC}$ 和 $\overline{IOWC}$ 提前一个时钟周期。

但当 $\overline{\text{AMWC}}$ 或 $\overline{\text{AIOWC}}$ 出现时,不能保证总线上出现稳定数据信息。其操作过程与读总线周期相似,这里就不再赘述。

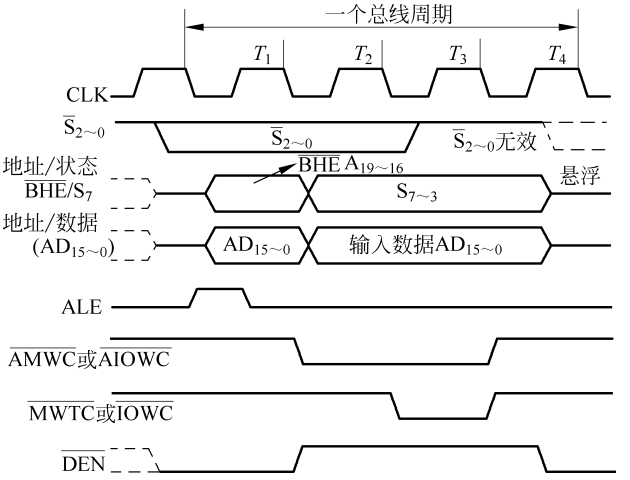


图 3.23 最大模式系统写总线周期时序

3. 总线请求

8086、8087 和 8089 都设有两个双重功能引脚 $\overline{RQ}/\overline{GT}_1$ 和 $\overline{RQ}/\overline{GT}_0$,其中的任一个都既可用来传送总线请求,也可发送总线保持响应信号和总线释放脉冲。但 $\overline{RQ}/\overline{GT}_0$ 的优先级高于 $\overline{RQ}/\overline{GT}_1$。

CPU 在每个时钟周期的上升沿检测 $\overline{RQ}/\overline{GT}_1$ 和 $\overline{RQ}/\overline{GT}_0$ 引脚,若采样到其中一个有 $\overline{RQ}$ 低电平有效信号,就表明有处理器提出总线保持请求。若 CPU 满足响应条件,就会在本次总线周期的 T_4 状态或空闲周期 T_1 的下降沿利用同一引脚发出响应信号,从而使 $\overline{GT}$ 低电平有效,并使系统具有三态的总线处于高阻状态,CPU 让出总线控制权,处于保持状态。同样,交出总线使用权的 CPU 仍将继续执行指令队列中已经预取的指令,直至遇到存取总线的指令或指令队列为空为止。请求使用总线的处理器使用完总线后,又利用同一 $\overline{RQ}/\overline{GT}$ 引脚向 CPU 发出负脉冲(释放脉冲),将总线控制权交还给 CPU。CPU 检测到释放脉冲后,又可控制对总线的操作。其中,从总线请求产生($\overline{RQ}$ 有效)到获得总线授予信号($\overline{GT}$ 有效)之间的时间延迟范围可以是 3～39 个时钟周期。最大模式系统中总线保持与响应时序如图 3.24 所示。

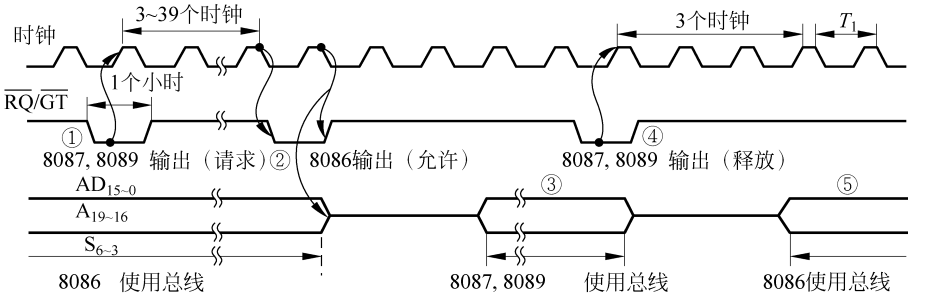


图 3.24 最大模式系统中总线保持与响应时序

3.8　例 题 解 析

1. 8086 CPU 在内部结构上由哪几部分组成？其功能是什么？

【解析】

8086 的内部结构分为两部分：总线接口部件 BIU 和执行部件 EU。总线接口部件 BIU 负责控制存储器与 I/O 端口的信息读写，包括指令获取与排队、操作数存取等；执行部件 EU 负责从指令队列中取出指令，完成指令译码与指令的执行。

2. 8086 的 BIU 由哪几部分组成？其功能是什么？

【解析】

8086 的总线接口部件主要由四部分组成：4 个段寄存器 CS/DS/ES/SS，用于保存各段地址；一个 16 位的指令指针寄存器 IP，用于保存当前指令的偏移地址；一个 20 位地址加法器，用于形成 20 位物理地址；指令流字节队列，用于保存指令；存储器接口，用于内总线与外总线的连接。

3. 8086 的 EU 由哪几部分组成？各有什么功能？

【解析】

8086 的 EU 主要由四部分组成：控制器、算术逻辑单元、标志寄存器、通用寄存器组。

(1) 控制器：从指令流顺序取指令、进行指令译码、完成指令的执行等。

(2) 算术逻辑单元 ALU：根据控制器完成 8/16 位二进制算数与逻辑运算。

(3) 标志寄存器：使用 9 位，标志分两类。其中状态标志 6 位，存放算数逻辑单元 ALU 运算结果特征；控制标志 3 位，控制 8086 的 3 种特定操作。

(4) 通用寄存器组：用于暂存数据或指针的寄存器阵列。

4. 说明标志位中溢出位与进位标志位的区别。

【解析】

进位标志位 CF 是指两个操作数在进行算术运算后，最高位(8 位操作为 D_7 位，16 位操作为 D_{15} 位)是否出现进位或借位的情况，有进位或借位，CF 置“1”，否则置“0”。

溢出位 OF 是反映带符号数(以二进制补码表示)运算结果是否超过机器所能表示的数值范围的情况。8086 中的数据用补码表示，对于 8 位的字节运算，数值范围为－128～＋127；对于 16 位的字运算，数值范围为－32 768～＋32 767。若超过上述范围，则称为“溢出”，OF 置“1”。

溢出和进位是两个不同的概念，某些运算结果，有“溢出”不一定有“进位”，反之，有“进位”也不一定有“溢出”。

5. 存储器物理地址为 400A5H～400AAH 的单元中，有 6 字节的数据分别为 11H、22H、33H、44H、55H、66H，若当前(DS)＝4002H，请说明它们的偏移地址值。如果要从存储器中读出这些数据，需要访问几次存储器，各读出哪些数据？

【解析】

这个题目考查重点是规则字和非规则字的应用。

由于：

物理地址=400A5H=段地址×16+偏移地址=40020H+偏移地址

偏移地址=400A5-40020=85H

若以最少访问次数而言,可以如下操作:从奇地址400A5H中读出一字节11H;从偶地址开始400A6H、400A7H两个单元读出一个字3322H;从偶地址400A8H、400A9H两个单元读出一个字5544H;从偶地址400AAH中读出一字节66H。最少读4次。

6. 8086被复位以后,有关寄存器的状态是什么?微处理器从何处开始执行程序?

【解析】

标志寄存器、IP、DS、SS、ES和指令队列置0,CS置全1(FFFFH)。处理器从FFFF0H存储单元取指令并开始执行。

7. 8086为什么采用地址/数据引线复用技术?

【解析】

考虑到芯片成本和体积,8086采用40根引线的封装结构。由于40根引线无法直接引出8086的全部20位地址、16位数据、诸多控制信号和状态信号,因此采用地址/数据线复用引线方法解决这一矛盾。从逻辑角度来看,地址与数据信号不会同时出现,二者可以分时复用同一组引线。

8. 8086 CPU形成总线数据时,为什么要对部分地址线进行锁存?用什么信号控制锁存?

【解析】

为了确保CPU对存储器和I/O端口的正常读/写操作,要求地址和数据同时出现在地址总线和数据总线上。但8086 CPU中$AD_0 \sim AD_{15}$总线是地址/数据分时复用的,即在总线周期的T_1状态传送出地址信息,T_3状态传送数据;因此借由8086 CPU送出的ALE高电平锁存信号,将在T_1状态传送出的地址信息存于锁存器中。

9. 8086构成系统时存储器分为哪两个存储体?它们如何与地址、数据总线连接?

【解析】

8086构成系统分为偶地址存储体和奇地址存储体。

偶地址存储体:连接$D_7 \sim D_0$,$A_0=0$时选通;

奇地址存储体:连接$D_{15} \sim D_8$,$\overline{BHE}=0$,$A_0=1$时选通。

10. 8086 CPU读/写总线周期各包含多少个时钟周期?什么情况下需要插入T_W等待周期?应插入多少个T_W,取决于什么因素?什么情况下会出现空闲状态T_i?

【解析】

8086 CPU读/写总线周期包含4个时钟周期。

当CPU与慢速的存储器或外设I/O端口交换信息时,系统中就要用一个电路来产生READY信号,并传递给CPU的READY引脚。CPU在T_3状态的下降沿对READY信号进行采样。如果READY信号是无效信号,那么,就会在T_3之后插入等待状态T_W。插入T_W的个数取决于CPU接收到高电平READY信号的时间。CPU在不执行总线周期时,总线接口部件就不和总线打交道,此时,进入总线空闲周期,出现空闲状态T_i。

11. 图3.25所示为8086最小模式下总线操作时序图,结合该图说明ALE、M/$\overline{IO}$、DT/$\overline{R}$、$\overline{RD}$、READY信号的功能。

【解析】

ALE为外部地址锁存器的选通脉冲,在T_1期间输出;M/$\overline{IO}$确定总线操作的对象是

存储器还是 I/O 接口电路，在 T_1 输出；DT/$\overline{R}$ 为数据发送/接收信号，用于控制总线收发器数据传送的方向，在 T_1 输出；$\overline{RD}$ 为读命令信号，在 T_2 输出；READY 信号为存储器或 I/O 接口“准备好”信号，在 T_3 期间给出，否则 8086 要在 T_3 与 T_4 间插入 T_W 等待状态。

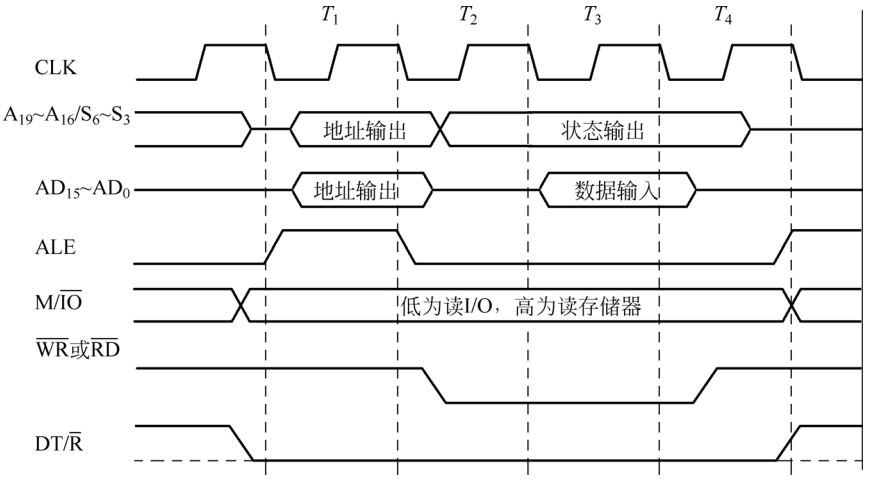


图 3.25　8086 最小模式下总线操作时序图

扫一扫
自测题

习　题　3

1. 8086 CPU 在内部结构上由哪几部分组成？各部分的功能是什么？

2. 简述 8086 CPU 的寄存器组织。

3. 8086 CPU 状态标志和控制标志有何不同？程序中是怎样利用这两类标志的？8086 的状态标志和控制标志分别有哪些？

4. 将 1001 1100 和 1110 0101 相加后，标志寄存器中 CF、PF、AF、ZF、SF、OF 各为何值？

5. 什么是存储器的物理地址和逻辑地址？在 8086 系统中，如何由逻辑地址计算物理地址？

6. 段寄存器 CS＝1200H，指令指针寄存器 IP＝4000H，此时，指令的物理地址是多少？指向这一物理地址的 CS 值和 IP 值是唯一的吗？

7. 在 8086 系统中，逻辑地址 FFFF：0001、00A2：37F 和 B800：173F 的物理地址分别是多少？

8. 在 8086 系统中，从物理地址 388H 开始顺序存放下列 3 个双字节的数据 651AH、D761H 和 007BH，请问物理地址 388H、389H、38AH、38BH、38CH 和 38DH 6 个单元中分别是什么数据？

9. 8086 微处理器有哪几种工作模式？各有什么特点？

10. 简述 8086 引脚信号中 M/$\overline{IO}$、DT/$\overline{R}$、$\overline{RD}$、$\overline{WR}$、ALE、$\overline{DEN}$ 和 $\overline{BHE}$ 的作用。

11. 简述最小模式下，8086 读总线周期和写总线周期各引脚上的信号动态变化过程。并说出 8086 的读周期时序与写周期时序的区别有哪些？