第 2 章　Intel 8086 系统结构

微处理器是组成微机系统的核心部件。本章首先介绍 8086 微处理器的结构，包括 8086 的功能结构、存储器及 I/O 组织和寻址方式以及 8086 的寄存器结构；然后讨论 Intel8086 微处理器的两种系统组态及各系统组态下的引脚功能；最后介绍 Intel 8086 的总线周期以及 8086 几个重要周期的典型时序。

2.1　8086 微处理器的结构

8086 微处理器是 Intel 系列微处理器中具有代表性的高性能 16 位微处理器，以致于后续推出的各种微处理器均保持与其兼容。

8086 微处理器采用 HMOS 工艺技术制造，外型封装为双列直插式，有 40 个引脚。主时钟频率有 4.77MHz(8088)，5MHz，8MHz 和 10MHz 几种。内部采用 16 位数据通路和流水线结构，从而允许其在总线空闲时预取指令，使取指令与执行指令实现了并行操作。8086 有 20 位地址线，可直接寻址的空间达 1MB。格式灵活、功能完善的指令系统不仅为程序设计带来方便，而且可对多种数据类型进行处理。8086/8088 支持多处理器系统，可方便地与数值协处理器 8087 和 I/O 处理器 8089 相连，组成多处理器系统，大大提高了系统的数据处理能力。本节将对 8086 微处理器的功能结构、存储器组织、寄存器结构等内容进行介绍。

2.1.1　8086 微处理器的功能结构

Intel 8086 微处理器属于第三代微处理器，具有 20 条地址线和 16 条数据总线，内部总线和 ALU 均为 16 位，可进行 8 位和 16 位操作。

8086 微处理器采用不同于第二代微处理器(8080，Z80)的一种全新结构形式，由两个独立的单元组成，一个称为总线端口单元 BIU(Bus Interface Unit)，另一个称为执行单元 EU(Execution Unit)，其功能框图如图 2.1 所示。图中虚线右半部分是 BIU，左半部分是 EU。两者并行操作，提高了 CPU 的运行效率。

1. 指令执行单元 EU

指令执行单元 EU 的功能是负责执行指令，即负责全部指令的译码和执行，同时管理 CPU 内部的有关寄存器。执行单元 EU 由一个 16 位的算术逻辑单元(ALU)、16 位的标志寄存器(实际仅用 9 位)、8 个 16 位的寄存器，以及数据暂存器和 EU 控制器等组成。

1) 算术逻辑运算单元(ALU)

它是一个 16 位的运算器，可用于 8 位或 16 位二进制算术运算或逻辑运算，运算结果可通过片内总线送到通用寄存器或标志寄存器，还可经 BIU 写入存储器。16 位的暂存器用来暂存参加运算的操作数。

2) 标志寄存器 (FR)

也叫程序状态字(PSW)寄存器，简称状态寄存器。其作用是用来存放 ALU 运算后的结

果特征或机器运行状态，标志寄存器长 16 位，实际使用了 9 位。

3) 通用寄存器组

它包含 8 个 16 位的寄存器，按功能分为两组，一组包括 AX，BX，CX，DX4 个寄存器，称为通用数据寄存器，用来存放操作数或地址。其中 AX 又称为累加器。另一组包括 DI，SI，SP 和 BP4 个寄存器，每个寄存器分别有各自的专门用途，故称为专用寄存器。其中，SI 叫源变址寄存器；DI 称为目的变址寄存器；SP 称为堆栈指示器，即堆栈指针；BP 为对堆栈操作的基址指示器，BP 中存放的是堆栈段中某一存储单元的偏移地址。

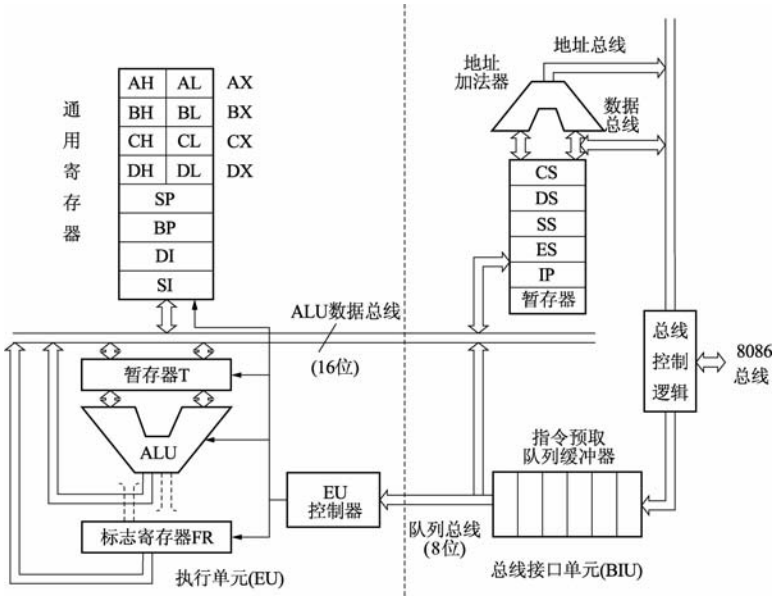


图 2.1　8086 CPU 的功能结构

4) EU 控制器

EU 控制器的作用是从 BIU 的指令队列中取指令，并对指令进行译码，根据指令要求向 EU 内部各部件发出相应的控制命令以完成每条指令所规定的功能。因此相当于传统计算机 CPU 中的控制器。

指令执行单元 EU 的工作就是执行指令，并不直接与外部发生联系，而是从总线端口单元 BIU 的指令队列中源源不断地获取指令并执行，省去了访问存储器取指令的时间，提高了 CPU 的利用率和整个系统的运行速度。如果在指令执行过程中需要访问存储器或需要从 I/O 端口取操作数时，则 EU 向 BIU 发出操作请求，并将访问地址(有效地址 EA)送给 BIU，由 BIU 从外部取回操作数送给 EU。当遇到转移指令、调用指令和返回指令时，EU 要等待 BIU 将指令队列中预取的指令清除，并按目标地址从存储器取出指令送入指令队列后，EU 才能继续执行指令。这时 EU 和 BIU 的并行操作显然要受到一定的影响，这是采用并行操作方式不可避免的。但只要转移指令、调用指令出现的概率不是很高，EU 和 BIU 间既相互配合又相互独立工作的工作方式仍将大大提高 CPU 的工作效率。

2. 总线端口单元 BIU

BIU 是 8086 微处理器在存储器和 I/O 设备之间的端口部件，负责对全部引脚的操作，即 8086 对存储器和 I/O 设备的所有操作都是由 BIU 完成的。所有对外部总线的操作都必须

有正确的地址和适当的控制信号，BIU 中的各部件主要是围绕这个目标设计的。BIU 提供了 16 位双向数据总线、20 位地址总线和若干条控制总线，其具体任务是：负责从内存单元中预取指令，并将其送到指令队列缓冲器暂存。CPU 执行指令时，BIU 根据指令的寻址方式通过地址加法器形成指令在存储器中的物理地址，然后访问该物理地址所对应的存储单元，从中取出指令代码送到指令队列缓冲器中等待执行。指令队列一共 6 个字节，一旦指令队列中空出 2 个字节，BIU 将自动进入读指令操作以填满指令队列；遇到转移类指令时，BIU 将指令队列中的已有指令作废，从新的目标地址中取指令送到指令队列中； EU 读写数据时，BIU 将根据 EU 送来的操作数地址形成操作数的物理地址，从内存单元或外设端口中读取操作数或者将指令的执行结果传送到该物理地址所指定的内存单元或外设端口中。

总线端口单元 BIU 主要由 4 个段寄存器、1 个指令指针寄存器、1 个与 EU 通信的内部寄存器、先入先出的指令队列、总线控制逻辑和计算 20 位物理地址的地址加法器组成。4 个段寄存器分别称为代码段寄存器(CS)、数据段寄存器(DS)、堆栈段寄存器(SS)和附加数据段寄存器(ES)。

1) 地址加法器和段寄存器

8086 微处理器的 20 位物理地址可直接寻址 1MB 存储空间，但 CPU 内部寄存器均为 16 位的寄存器。20 位的物理地址是由专门的地址加法器将有关段寄存器内容(段的起始地址)左移 4 位后，与 16 位的偏移地址相加。如在取指令时，由 16 位指令寄存器(IP)提供一个偏移地址(逻辑地址)，在地址加法器中与代码段寄存器(CS)内容相加，形成 20 位物理地址，送到总线上物理取指令的寻址。

2) 16 位指令寄存器(Instruction Pointer，IP)

指令寄存器 IP 用来存放下一条要执行指令的偏移地址 EA(也叫有效地址)，IP 只有和 CS 相结合，才能形成指向指令存放单元的物理地址。在程序执行过程中，IP 的内容由 BIU 自动修改，通常是进行＋1 修改，当 EU 执行转移指令、调用指令时，BIU 装入 IP 的则是目标地址。

3) 指令队列缓冲器

指令队列的作用是预存 BIU 从存储器中取出的指令代码。当 EU 正在执行指令，且不需要占用总线时，BIU 会自动地进行预取指令操作。8086 的指令队列为 6B，可按先后次序依次预存 6 个字节的指令代码。该队列寄存器按“先进先出”的方式工作，并按顺序取到 EU 执行。其操作遵循以下原则：

(1) 每当指令队列缓冲器中存满一条指令后，EU 就立即开始执行。

(2) 每当 BIU 发现队列中空了 2 个字节时，就会自动地寻找空闲的总线周期进行预取指令操作，直至填满为止。

(3) 每当 EU 执行一条转移、调用或返回指令后，BIU 就会清除指令队列缓冲器，并从新地址开始预取指令，实现程序段的转移。

BIU 和 EU 是各自独立工作的，在 EU 执行指令的同时，BIU 可预取下一条或几条指令。因此，在一般情况下，CPU 执行完一条指令后，就可立即执行存放在指令队列中的下一条指令，从而减少了 CPU 为取指令而等待的时间，提高了 CPU 的利用率，加快了整机的运行速度。另外也降低了对存储器存取速度的要求。

4) 总线控制逻辑电路

总线控制逻辑电路将 8086 微处理器的内部总线和外部总线相连，是 8086 微处理器与

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微型计算机系统。

2.1.2　8086 的存储器分段组织

1. 存储器地址空间和数据存储格式

8086 的存储器是以字节(8 位) 为单位组织的。它们具有 20 条地址总线，所以可寻址的地址空间容量为 2^{20} B(约 1MB)。每个字节对应一个唯一的地址，地址范围为 0～2^{20} −1(用十六进制表示为 00000～FFFFFH)，如图 2.2 所示。

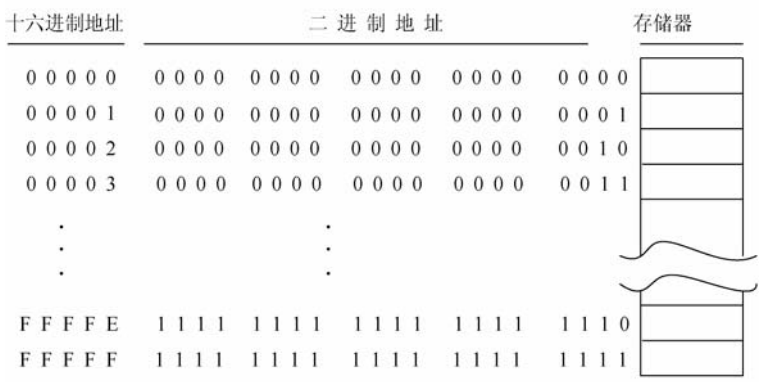


图 2.2　存储器的地址

存储器内两个连续的字节，定义为一个字。一个字中的每个字节，都有一个字节地址，每个字的低字节(低 8 位)存放在低地址中，高字节(高 8 位)存放在高地址中，字在存储器中的存放格式如图 2.3 所示。字的地址指低字节的地址。各位的编号方法是最低位(LSB)为位 0。一个字节中，最高位(MSB)的编号为位 7；一个字中最高位的编号为 15。这些约定如图 2.4 所示。

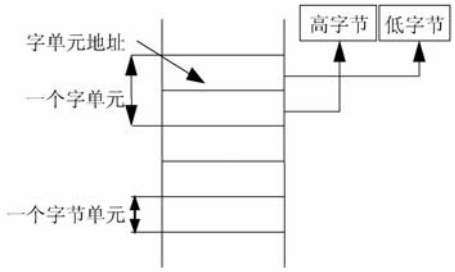


图 2.3　数据存储格式

8086 允许字从任何地址开始。字的地址为偶地址时，称字的存储是对准的，若字的地址为奇地址时，称字的存储是未对准的。

8086 微处理器数据总线 16 位，对于访问(读或写)字节的指令，需要一个总线周期，而对于访问一个奇地址的字的指令，则需要两个总线周期(CPU 自动完成)。

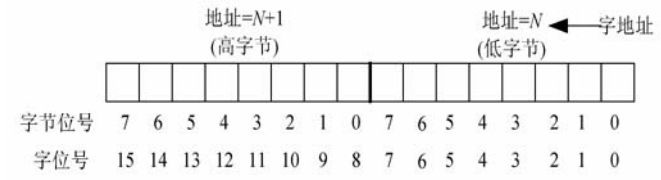


图 2.4　地址和位号的约定

2. 存储器的分段

前已述及，8086 微处理器地址总线 20 条，存储器地址空间为 1MB。但是，8086 微处理器内所有的寄存器(CS，DS，SS，ES，SP，BP，SI，DI 和 IP)都是 16 位的，最多只能寻址 64KB 空间。为了达到能对 1MB 的存储器寻址 8086/8088 系统中引入了存储空间分段概念，即将整个 1MB 的存储空间分成若干个存储段，每个段是存储器中可独立寻址的逻辑单位，称为逻辑段，每个段的长度为 64KB，段内地址是连续的，允许各个逻辑段在整个 1MB 存储空间内浮动，但每个逻辑段的起始地址(简称段基址/段首址)必须从能被 16 整除的地址开始，即段的起始地址的低 4 位二进制码必须是 0。一个段的起始地址的高 16 位被称为该段的段地址。显然，在 1MB 的存储器地址空间中，可以有 2^{16} 个段地址。任意相邻的两个段地址相距 16 个存储单元。段内一个存储单元的地址，可用相对于段起始地址的偏移量来表示，这个偏移量称为段内偏移地址，也称为有效地址 EA。偏移地址也是 16 位的，所以，一个段最大可以包括一个 64KB 的存储器空间。各个逻辑段之间可以首尾相连，也可以完全分离或者重叠(部分重叠或完全重叠)，如图 2.5 所示。

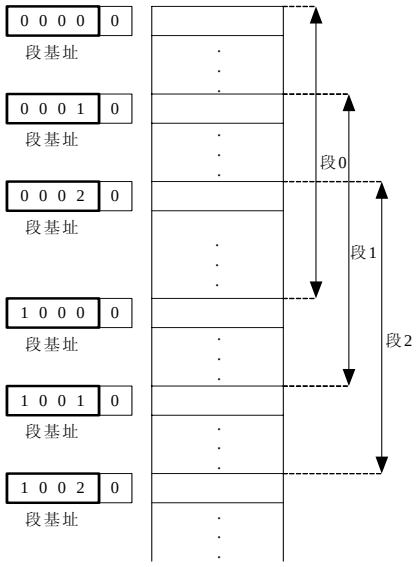


图 2.5　存储器分段和段的重叠

3. 物理地址的形成

由图 2.5 可知，存储器分段以后，任何一个存储单元，可以唯一地被包含在一个逻辑段中，也可以包含在两个或多个重叠的逻辑段中，只要能得到它所在段的段基址和段内偏移地址(存储单元本身到所在段段首址的字节数)就可以对它进行访问。而对 1MB 存储器内的任何一个单元进行访问，必须使用 20 位地址码，即物理地址。现在的问题是如何从 16 位的段基址和 16 位的段内偏移地址变换为 20 位的实际地址。

由上述分段概念可知，在 8086 系统中，每个存储单元在存储器中的位置可以用逻辑地址和物理地址来表示。所谓逻辑地址，是程序设计中使用的地址，由段基址和段内偏移地址两部分组成，段基址和段内偏移地址都是无符号的 16 位二进制数。物理地址也叫实际地址或绝对地址，是 CPU 访问存储器时实际使用的地址，地址总线上传送的就是这个地址。对 1MB 容量的存储器来说，物理地址为 20 位，其范围为 00000H～FFFFFH。存储器中任何一个存储单元的物理地址是 00000H～FFFFFH 内的某一值。显然，物理地址应由逻辑地址变换得到，两者的变换关系如图 2.6 所示。即将 16 位段基址左移 4 位(相当于在段基址的低 4 位补 4 个“0”)，然后与 16 位段内偏移地址相加而获得 20 位物理地址，这相当于完成以下地址计算：

物理地址＝段基址×16＋段内偏移地址

当 CPU 访问存储器时，必须完成上述的地址计算，此地址计算过程是由 CPU 内总线端口部件 BIU 中的地址加法器完成的。

例如，某条指令在代码段中的逻辑地址为：CS＝1000H，IP＝4052H，则其物理地址为 14052H，如图 2.7 所示。

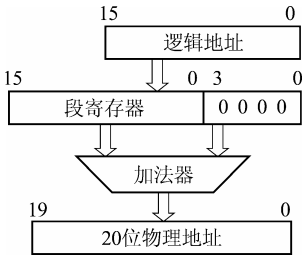


图 2.6　物理地址的形成过程

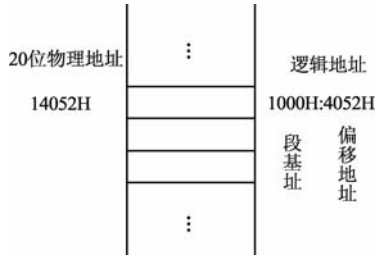


图 2.7　逻辑地址与物理地址

显然，当 CS＝1200H，IP＝2052H 时，物理地址也是 14052H。这就是说，在 8086/8088 存储器中，同一个物理地址可以对应多个逻辑地址，即可由不同的段基址和偏移地址组合得到。

在访问存储器时，段地址总是由段寄存器提供的。8086 微处理器的 BIU 单元设有 4 个段寄存器(CS、DS、SS、ES)，所以 CPU 可以通过这 4 个段寄存器来访问 4 个不同的段。用程序对段寄存器的内容进行修改，可实现对所有段的访问。

4. 信息的分段存储与段寄存器的关系

段寄存器的利用不仅使存储器地址空间扩大到 1MB，而且为信息按特征分段存储带来了方便。存储器中的信息可分为程序、数据和计算机的状态信息。为了操作方便，存储器可相应地划分为：程序区，该区存储程序的指令代码；数据区，它存储原始数据、中间结果和最终结果；堆栈区，用以存储需要压入堆栈的数据或状态信息。段寄存器的分工是：代码段寄存器 CS 划定并控制着程序区；数据段寄存器 DS 和附加段寄存器 ES 控制着数据区；而堆栈段寄存器 SS 对应堆栈存储区。表 2-1 列出了各种类型访问存储器时所要使用的段寄存器和段内偏移地址的来源，它规定了为各种目的访问存储器时所形成的 20 位物理地址的原则。

表 2-1　各种类型存储器访问所使用的段寄存器和段内偏移地址

存储器操作的类型	隐含的段寄存器	允许超越的段寄存器	段内偏移地址来源
取指令	CS	无	IP
堆栈操作	SS	无	SP
通用数据读取	DS	CS，ES，SS	由寻址方式求得有效地址
源数据串	DS	CS，ES，SS	SI
目的数据串	ES	无	DI
用 BP 作为基址寄存器	SS	CS，DS，ES	由寻址方式求得有效地址

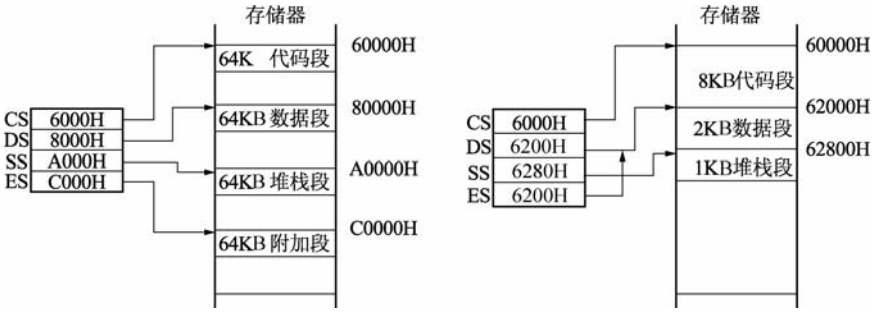
由表 2-1 可知：

(1) 对存储器任何类型的访问，其段地址要么由默认段寄存器提供，要么由“指定”段寄存器提供。所谓默认段寄存器是指在指令中不用专门的信息指定另外一个段寄存器的情况，此时的段地址就由默认的段寄存器提供。实际程序设计时，绝大多数属于这种情况，因此，要熟记各种类型访问内存时的段寄存器。但也有几种类型访问存储器时允许指定另外的段寄存器，这为访问不同的存储器段提供了灵活性。段寄存器指定是靠在指令码中增加一个字节的前缀码实现的。对于取指令时的代码段访问、堆栈操作的堆栈段访问以及字符串操作指令的目的地址不允许使用指定段寄存器方式，而只能使用默认段寄存器。

(2) 段寄存器 DS、ES 和 SS 的内容是在程序中通过指令设置的，任何传送类指令不能直接向 CS 中传送数据，但转移、调用、返回类指令可以设置和影响 CS 的内容。更改段寄存器的内容，意味着存储区的移动。这也说明无论程序区、数据区还是堆栈区都可以不限于 64KB 的容量，都可以通过重置段寄存器内容的方法予以扩大，而且各个存储区都可以在存储器中浮动。

(3) 表中“段内偏移地址来源”一栏指明，除了两种类型访问存储器是“依寻址方式求得有效地址”外，其他都指明了一个 16 位的指针寄存器或变址寄存器。如取指令访问内存时，段内偏移地址只能由指令指针 IP 提供；当堆栈进行压入和弹出操作时，段内偏移地址只能由 SP 提供等。除此之外的内存访问，段内偏移地址则由指令码规定的寻址方式确定。

每个段的最大容量为 64KB，但在实际程序设计时，一般情况下，不需要这么大的空间，因而段有部分重叠。图 2.8 给出了两种典型的分段方法。



(a) 段不重叠　　(b) 段部分重叠

图 2.8　典型分段方法

需要指出的是，基于 8086 微处理器的 IBM PC 是一个通用微机系统，在存储空间的安排上，有一部分空间被系统占用，用户不能使用。例如，在主存储器的地址低端和高端有一部分存储单元的用处是固定的，如用作中断向量表、显示缓冲区和系统启动地址等，用户是不能占用的。

2.1.3　8086 的寄存器结构

由图 2.1 可知，8086 微处理器内部具有 14 个 16 位的内部工作寄存器，用于提供指令执行、指令及操作数的寻址。寄存器结构如图 2.9 所示。14 个寄存器按功能不同可分为 3 组，分别为通用寄存器组、段寄存器组和控制寄存器组。

1. 通用寄存器组

图 2.9 中 8 个 16 位通用寄存器分为两组：数据寄存器及地址指针和变址寄存器。

1) 数据寄存器

数据寄存器包括 AX，BX，CX，DX 这 4 个寄存器，

图 2.9　8086 的寄存器结构

位于 CPU 的 EU 中。通用数据寄存器主要用来存放算术/逻辑运算操作数，中间结果和地址。由于这些寄存器的存在，避免了每次算术/逻辑运算都要访问存储器，因为访问存储器需要较长时间，因而 CPU 内有较多的通用数据寄存器，不仅为编程提供方便，更主要的是可以加快 CPU 的运行速度。

数据寄存器 AX，BX，CX，DX，既可作为一个 16 位的寄存器使用，存放 16 位的数据或地址，也可以分别作为两个 8 位寄存器使用，低 8 位分别称为 AL，BL，CL，DL，高 8 位分别称为 AH，BH，CH，DH。作为 8 位寄存器使用时只能存放数据，不能存放地址。这些寄存器的双重性使得 8086 微处理器可以处理字也可以处理字节数据。也就较好的实现了与 8 位微处理器的兼容。

2) 地址指针和变址寄存器

地址指针和变址寄存器包括 SP，BP，SI，DI 四个 16 位寄存器。它们一般是用来存放操作数的偏移地址。其中 SP 称为堆栈指示器，SP 中存放的是当前堆栈段中栈顶的偏移地址，堆栈操作中压栈操作和出栈操作指令就是从 SP 中得到段内偏移地址的。

BP 为对堆栈操作的基址寄存器，BP 中存放的是堆栈中某一存储单元(某一栈单元) 的偏移地址。当操作数在堆栈中时，用 BP 作变址寄存器，指出操作数在堆栈段中的偏移地址。SP 和 BP 通常与 SS 联用，为访问当前堆栈段提供方便。

SI 和 DI 称为变址寄存器，通常与 DS 联用，为访问当前数据段提供段内偏移地址。SI 和 DI 除作一般变址寄存器外，在串操作指令中还作为指示器使用，其中 SI 规定用作存放源操作数的偏移地址，称为源变址寄存器，DI 规定用作存放目的操作数的偏移地址，称为目的变址寄存器，且二者不能混用。由于串操作指令规定源操作数(源串)必须位于当前数据 DS 中，目的操作数(即目的串)必须位于附加数据段 ES 中，所以 SI 和 DI 中的内容是当前数据段或当前附加数据段中某一存储单元的偏移地址。因此，在串操作中，SI，DI 必须与 DS，ES 联用，这是一种约定。

当 SI，DI 和 BP 不作地址指针和变址寄存器使用时，也可将其当作一般数据寄存器使用，用来存放操作数或运算结果，当然这时只能作 16 位寄存器用，不能作 8 位寄存器。SP 只能作堆栈指示器，而不能作数据寄存器使用。

以上 8 个 16 位通用寄存器在一般情况下都具有通用性，从而提高了指令系统的灵活性。通用寄存器除具有通用特性外，还具有各自的特定用法，有些指令还隐含地使用这些寄存器。例如，串操作指令和移位指令中约定必须使用 CX 寄存器(而不能用其他寄存器)作为计数寄存器，其作用是存放串的长度和移位次数，这样，在指令中就不必给出 CX 寄存器号，缩短了指令长度，简化了指令的书写形式。通常称这种使用方式为“隐含寻址”，隐含寻址，实际上就是给某些通用数据寄存器规定一些特殊用法，程序设计者编程时必须遵循这些规定。由于隐含寻址的原因，把 AX 寄存器又称为累加器，BX 寄存器又称为基址寄存器，DX 寄存器又称为数据寄存器。表 2-2 给出了 8086 中通用寄存器的特殊用途和隐含性质。

表 2-2　通用寄存器的特定用法和隐含性质

寄存器名称	特定用法	隐含性质
AX，AL	• 在乘法和除法指令中作累加器 • 在 I/O 指令中用作数据寄存器	隐含寻址 显式寻址
AH	在 LAHF 中作目的寄存器	隐含寻址
AL	• 在 BCD 码及 ASCII 码运算指令中作累加器	隐含寻址

Binary-coded Decimal

(续)

寄存器名称	特定用法	隐含性质
BX	·在间接寻址中作地址寄存器 ·在间接寻址中作基址寄存器 ·在 XLAT 指令中作基址寄存器	显式寻址 显式寻址 隐含寻址
CX	在循环指令和字符串指令中作循环次数的计数寄存器，每做一次循环，CX 的内容减 1	隐含寻址
CL	在移位及循环移位指令中作移位次数及循环移位次数的计数寄存器	隐含寻址
DX	·在 I/O 指令间接寻址时作地址寄存器 ·在乘法和除法指令中作为辅助累加器 (当乘积或被除数为 32 位数时存放高 16 位)	显式寻址 隐含寻址
BP	在间接寻址中作为访问堆栈段的基址寄存器	显式寻址
SP	在堆栈操作中作堆栈指针	显式寻址
SI	·在字符串操作指令中作源变址寄存器 ·在间接寻址中作地址寄存器 ·在间接寻址中作变址寄存器	隐含寻址 显式寻址 显式寻址
DI	·在字符串操作指令中作目的变址寄存器 ·在间接寻址中作地址寄存器 ·在间接寻址中作变址寄存器	隐含寻址 显式寻址 显式寻址

2. 段寄存器组

前面已经指出，访问存储器的地址码由段基址和段内偏移地址两部分组成。段寄存器用来存放段基址。总线端口单元(BIU)设置 4 个 16 位的段寄存器，它们分别是代码段寄存器(CS)，数据段寄存器(DS)，堆栈段寄存器(SS)和附加数据段寄存器(ES)。CPU 可通过 4 个段寄存器访问存储器中 4 个不同的段(每段 64KB)。4 个段寄存器以及它们所指示的 4 个逻辑段分别是：

代码段寄存器 (Code Segment，CS)。它存放当前代码段的段基址值。CS 的内容左移 4 位再加上指令指针 IP 的内容就是下一条要执行的指令。例如，某指令在代码段内的偏移地址为 0100H，即 IP＝0100H，当前代码段寄存器 CS＝2000H，则该指令在主存储器中的物理地址为

PA＝(CS)左移 4 位＋(IP)＝20000H＋0100H＝20100H

数据段寄存器 (Data Segment，DS)。它存放当前数据段的段基址。通常数据段用来存放数据和变量。DS 的内容左移 4 位再加上按指令中存储器寻址方式计算出来的偏移地址，即为对数据段指定单元进行读写的地址。例如，当访问数据段中某一变量时，该变量的物理地址 PA 为

PA＝(DS)左移 4 位＋该变量的偏移地址

堆栈段寄存器 (Stack Segment，SS)。它存放当前堆栈段的段基址。堆栈是程序执行中所需要的临时数据存储区(堆栈区)，采用“后进先出”工作方式，堆栈操作所处理的就是该段中的数据，堆栈段的起始地址(段基址)由堆栈段寄存器 SS 指出。堆栈段一旦定义好之后，系统则自动以 SP 为指针指示栈顶位置(即栈顶的偏移地址)。对堆栈进行压入/弹出数据的操作时，只能使用 PUSH 和 POP 指令，这时栈顶的物理地址应为

PA＝(SS)左移 4 位＋(SP)

当其他指令要访问堆栈段中的某一存储单元时，必须通过基址寄存器 BP 进行，即将

该存储单元的偏移地址置入 BP 中，这时该存储单元的物理地址 PA 应为：

PA＝(SS)左移 4 位＋(BP)

附加段寄存器 (Extra Segment，ES)。附加段是一个附加数据段。附加段是在进行字符串操作时作为目的区使用的，ES 存放附加段的段基址，DI 存放目的区的偏移地址。

一般来说，当程序较少，数据量又不大时，代码段、数据段、堆栈段和附加段可设置在同一段内，即包含在 64KB 之内。当程序和数据量较大，超过 64KB 时，可定义多个代码段、数据段、附加段和堆栈段。这时在 CS，DS，SS 和 ES 中存放的是当前正在使用的逻辑段段基址，使用中可以通过修改这些段的寄存器的内容，以访问其他段扩大程序规模。必要时，可通过在指令中增加段超越前缀符来指向其他段。

3. 控制寄存器组

1) 指令指针 (Instruction Pointer，IP)

指令寄存器(IP)和传统 CPU 中的程序计数器 PC 的作用相似，用来存放下一条要执行的指令在当前代码段中的偏移地址。在程序运行中，IP 的内容由 BIU 自动修改，使之总是指向下一条要执行的指令地址，因此它是用来控制指令执行顺序的重要寄存器。其内容程序不能直接访问，但当执行转移指令、调用指令时，其内容可被修改，置入的是目标地址或子程序首地址，IP 的原内容被压入堆栈，返回时再被恢复。

2) 标志寄存器(FLAG)

标志寄存器也称程序状态字(PSW)寄存器。8086 CPU 中有一个 16 位的标志寄存器，用来存放运算结果的特征和机器工作状态，实际仅用了 9 位，具体格式如图 2.10 所示。

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

图 2.10　8086 标志寄存器格式

所用的 9 位标志，按功能可分为两类：一类叫状态标志，用来表示运算结果的特征，是指令执行后自动建立的，共 6 个，它们是 CF，PF，AF，ZF，SF 和 OF，这些特征会像某种先决条件一样影响后面的操作。另一类叫控制标志，用来控制 CPU 的操作或工作状态，共 3 个：DF，IF 和 TF。控制标志是人为设置的，指令系统中有专门用来设置或清除控制标志的指令，每一种控制标志，都对 CPU 的一个特定操作起控制作用。

(1) 状态标志位功能说明。

① (Carry Flag，CF)——进位标志。当本次算术运算结果使最高位产生进位(加法运算)或借位(减法运算)时，则此标志位置“1”，即 CF＝1；若加法运算结果最高位无进位，或减法运算结果最高位无借位，则 CF＝0。此外，循环移位指令执行过程会影响这一标志。

② (Parity Flag，PF)——奇偶标志。此标志是反映运算结果中含“1”r 个数是奇数还是偶数，当本次运算结果中含“1”的个数为偶数时，PF＝1，为奇数时，PF＝0。

③ (Auxiliary Carry Flag，AF)——辅助进位标志。当进行 8 位数(字节)或 16 位数(字)的低 8 位运算时，低 4 位向高 4 位(即第 3 位向第 4 位)有进位或借位，AF＝1，否则 AF＝0。AF 标志用于 BCD 码的十进制算术指令中，以判别是否要进行十进制调整。

④ (Zero Flag，ZF)——零标志。若本次运算结果为 0，则 ZF＝1，否则 ZF＝0。

⑤ (Sign F1ag SF)——符号标志。此标志用于反映带符号数运算结果的符号是正还是

负。对于带符号数，用最高位表示数的符号，当本次运算结果最高位为 1，表示结果为负数，则 SF＝1，否则 SF＝0。

⑥ (Overflow Flag，OF)——溢出标志。所谓溢出，就是当对带符号数进行字节运算时，其结果超出−128～+127 的范围，或字运算的结果超出−32768～+32767 的范围时，称为溢出。因为这时运算结果已超出目标单元所能表示的数值范围，从而会丢失有效数字，出现错误结果，因此当运算结果产生溢出时，应使 OF＝1，否则 OF＝0。

例如：将十六进制数 5349H 和 465AH 相加，并说明其标志位状态。

$$\begin{array}{rr} & 0101\ \ 0011\ \ 0100\ \ 1001 \\ + & 0100\ \ 0110\ \ 0101\ \ 1010 \\ \hline & 1001\ \ 1001\ \ 1010\ \ 0011 \end{array}$$

两正数相加(补码加)，结果为负，显然运算产生了溢出，即超出了机器所能表示的范围(15 位数值位最多只能表示+32767＝2^{15}−1)，故 OF＝1，由于运算结果的最高位为 1，所以 SF＝1，运算结果本身不为 0，故 ZF＝0，又由于运算结果的低 8 位中含 1 个数为偶数，故 PF＝1，运算结果的最高位没有向前产生进位，故 CF＝0，运算过程中第 3 位向第 4 位(即低 4 位向高 4 位)产生了进位，故 AF＝1。

(2) 控制标志位功能。

① (Interrupt Enable F1ag，IF)——中断允许标志。IF＝1 时，表示允许 CPU 响应外部可屏蔽中断请求；如果 IF＝0，则禁止 CPU 响应外部可屏蔽中断请求。用 STI 指令可使 IF 标志位置“1”，CLI 指令可使 IF 标志位置“0”。

② (Direction Flag，DF)——方向标志。控制字符串操作指令地址指针的变化方向。若 DF＝0，字符串操作指令使地址指针自动增量，即串操作由低地址向高地址进行；如果 DF＝1，表示地址指针自动减量，即由高地址向低地址进行串操作。用 STD 指令可使 DF 标志位置“1”，用 CLD 指令可使 DF 标志位置“0”。

③ (Trap Flag，TF)——单步标志。TF＝1，表示使 CPU 进入单步工作方式，即 CPU 每执行完一条指令就自动产生一次内部中断，使 CPU 转去执行一个单步中断服务程序。用户可利用此功能来检查每条指令的执行情况。这在程序调试过程中是很有用的。如果 TF＝0，表示 CPU 正常执行程序。

2.2　8086 的系统组态及引脚功能

2.2.1　8086 的两种系统组态

为了尽可能适应各种各样的使用场合，8086 CPU 芯片在设计时，就考虑了能够在两种方式下工作，即最小工作方式和最大工作方式。

1. 最小工作方式

所谓最小工作方式，就是系统中只有一个 8086 微处理器，在这种情况下，所有的总线控制信号，都是直接由 8086 CPU 产生的，系统中的总线控制逻辑电路被减到最少，最小工作方式适用于由单微处理器组成的小系统。在这种系统中， 8086 CPU 直接产生所有的总线控制信号，因而省去了总线控制逻辑。图 2.11 为最小系统的系统配置图。

由图 2.11 可知，当 8086 CPU 的 MN/MX 引脚接+5V 电源时，8086 CPU 工作于最小系

统状态，用于构成小型的单处理机系统。在图 2.11 所示的 8086 系统中，除 8086CPU、存储器和 I/O 端口电路外，还有 3 部分支持系统工作的器件：时钟发生器、地址锁存器和数据收发器。

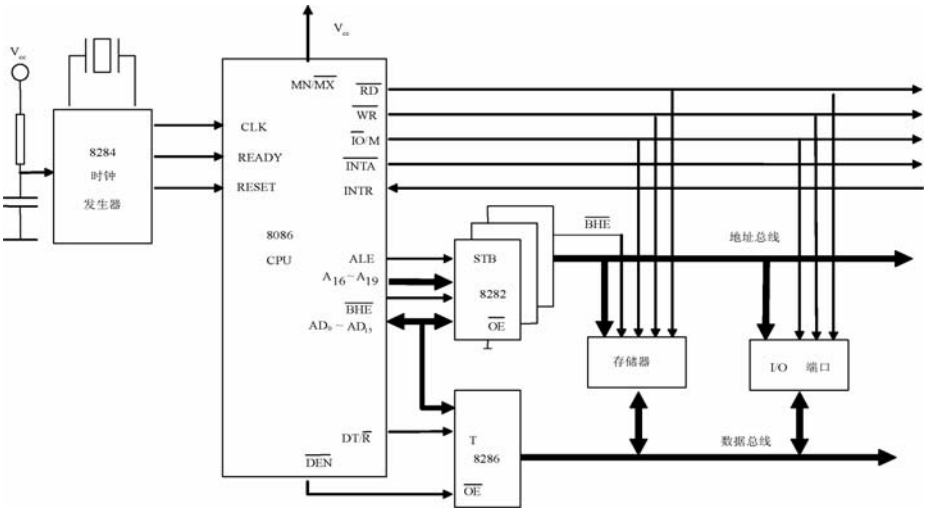
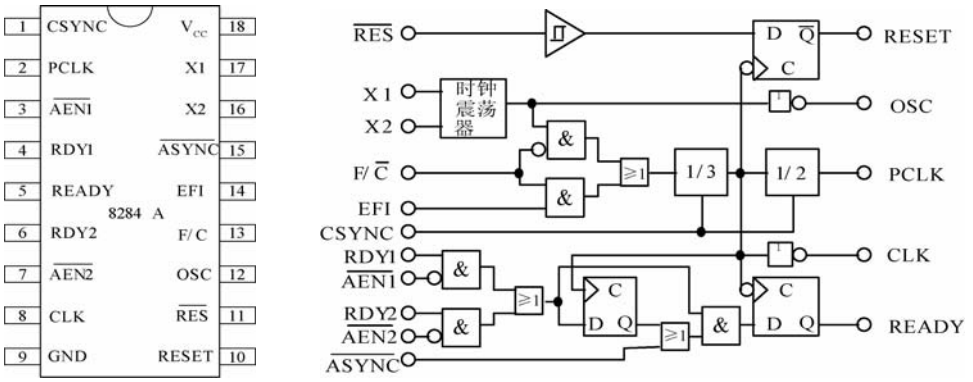


图 2.11　8086 最小系统配置

1) 时钟发生器 8284A

8284A 是用于 8086 系统的时钟发生器/驱动芯片，它为 8086 以及其他外设芯片提供所需要的时钟信号。图 2.12 为 8284A 的结构框图及引脚图。由图可见，8284A 由 3 部分电路组成。



(a) 引脚图　　(b) 内部结构图

图 2.12　8284A 的结构框图与引脚

(1) 时钟信号发生器电路提供系统所需要的时钟信号，有两个来源：一个是在 X1 与 X2 引脚之间接上晶体，由晶体振荡器产生信号；另一个是由 EFI 引脚加入的外接振荡信号产生时钟信号，两者由 F/$\overline{C}$ 端信号控制。F/$\overline{C}$ =0 时，表示有外接振荡器产生。

如果晶体振荡器的工作频率为 14.31818MHz，则该时钟脉冲(OSC)经3 分频后得到 4.77MHz 的时钟脉冲 CLK，即为处理器(如 8086)所需要的时钟信号，CLK 再经 2 分频后产生外设时钟 PCLK，其频率为 2.3805MHz。

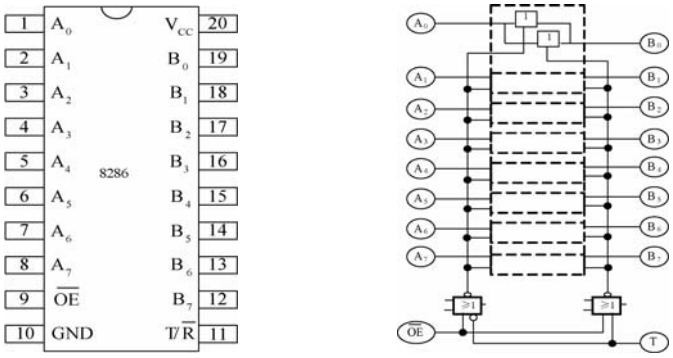
(2) 复位生成电路是由一个施密特触发器和一个同步触发器组成，输入信号 $\overline{RES}$ 在时钟脉冲下降沿加入同步触发器的 D 端，由 CLK(8086 的时钟信号)同步产生 RESET 信号，该信号为低电平有效。在 PC/XT 中，$\overline{RES}$ 由电源的 PWRGOOD 信号经 8284A 延时和同步后产生系统复位信号，使系统初始化。

(3) 就绪控制电路有两组输入信号，每一组都有允许信号 $\overline{AEN}$ 和设备就绪信号 RDY。$\overline{AEN}$ 是低电平有效信号，用以控制其对应的 RDY 信号是否有效，RDY 为高电平时，表示已经能正确地完成数据传输。$\overline{ASYNC}$ 输入端规定了就绪信号同步操作的两种方法，当 $\overline{ASYNC}$ 为低电平时，对有效的 RDY 信号提供两级同步，RDY 变为高电平后，首先在 CLK 的上升沿上同步到触发器 1，然后在 CLK 的下降沿上同步到触发器 2，使 READY 信号成为有效电平。RDY 变为低电平时，将直接在 CLK 下降沿上同步到触发器 2，使 READY 输出信号无效。如果 $\overline{ASYNC}$ 为高电平，则 RDY 输入信号直接与触发器 2 同步在 CLK 下降沿上，这种工作方式用于能保证满足 RDY 建立时间要求的同步设备中。

2) 数据总线收发器 8286/8287

当一个系统中数据总线上挂接的外设端口部件较多时，就必须在数据总线上接入总线收发器以增加总线的驱动能力。

在 8086CPU 和系统数据总线之间接入了一个双向总线驱动器 8286/8287。8286/8287 是一种具有三态输出的 8 位总线收发器，具有很强的总线驱动能力。图 2.13 是 8286 的引脚和内部单元结构图。由图可知，8286 具有 8 路双向缓冲电路，每一路双向缓冲电路都由两个三态缓冲器反向并联组成，以实现 8 位数据的双向传送。由于 8286 中使用的三态缓冲器是不反相的，所以 8286 的输入信号和输出信号是同相的。8287 的功能、内部结构和连接方式与 8286 基本相同，只是 8287 内使用的每个三态缓冲都有反相功能，所以 8287 的输入与输出信号是反相的。



(a) 8286 的引脚　　(b) 8286 的引脚与内部结构图

图 2.13　8286 的引脚及内部结构图

图中：

A_7～A_0 和 B_7～B_0 是数据输入/输出端。

$\overline{OE}$ ——输出允许信号，也叫缓冲器开启控制信号。该信号控制是否允许数据通过 8286/8287。当 EN＝0 时，允许数据通过 8286/8287，当 $\overline{OE}$ ＝1 时，禁止数据通过 8 位缓冲器，8286/8287 输出呈高阻抗状态。在 8086/8088 系统中，EN 端与 CPU 的数据允许信号 DEN 相连，当 CPU 与存储器或 I/O 端口进行数据交换时，用来控制是否允许数据通过 8286/8287，

DEN 有效(低电平)时，使 EN 有效，允许数据通过，反之，当 DEN 无效(高电平)时，使 EN 也无效，禁止数据通过。

T——数据传送方向控制信号，当 T＝1 时，8 位数据被正向传送，由 A_7～A_0 传送到 B_7～B_0，当 T＝0 时，8 位数据被反向传送，由 B_7～B_0 传送到 A_7～A_0。实际使用时，T 端与 CPU 的 DT/$\overline{R}$ (数据发送/接收)引脚相连，控制 8 位数据是从 CPU 向存储器或 I/O 端口写入(DT/$\overline{R}$ ＝1)，还是由存储器或 I/O 端口向 CPU 传送(DT/$\overline{R}$ ＝0)。$\overline{OE}$ 与 T 信号要配合使用，其组合功能如表 2-3 所示。

表 2-3　$\overline{OE}$ 与 T 的组合功能

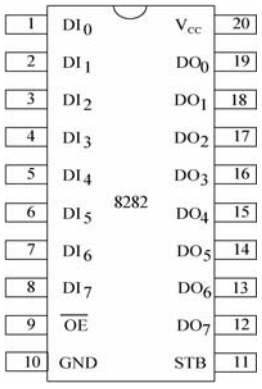
$\overline{OE}$	T	传送方向
0	1	A→B(正向)
0	0	B→A(反向)
1	×	高阻

在 8086 最小模式系统中，除 CPU 外，还允许接入其他总线主模块(如 DMA 控制器)共享总线，当其他总线主模块向 CPU 发出总线请求，要求使用总线时，如果 CPU 允许，则会使 DEN 和 DT/$\overline{R}$ 引脚呈高阻抗状态，从而也使 8286/8287 被禁止，输出端变为高阻抗状态。让出总线控制权。

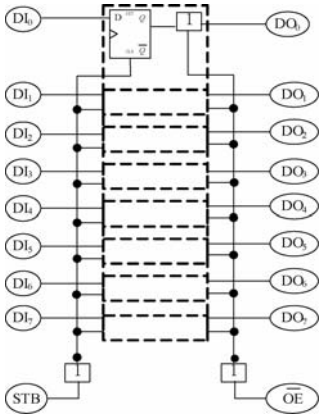
3) 地址锁存器 8282

由于 8086 CPU 的地址/数据和地址/状态总线是分时复用的，即 CPU 在读/写存储器或 I/O 端口时，总是在总线周期的 T_1 状态首先发出地址信号到 AD_{15}～AD_0 和 A_{19}/S_6～A_{16}/S_3 上，随后(T_2 以后)又用这些引脚来传送数据和状态信号，而存储器或 I/O 端口电路通常要求在与 CPU 进行数据传送的整个总线周期内必须保持稳定的地址信息，因而必须加入地址锁存器，在总线周期的 T_1 状态(即在数据送上总线之前)先将地址锁存起来。以使在整个读/写总线周期内保持地址稳定。

8282 是 8 位三态数据锁存器，其引脚及内部结构如图 2.14 所示。



(a) 8282/8283 的引脚



(b) 8282/8283 的引脚与内部结构图

图 2.14　8282/8283 的引脚及内部结构图

图中：

DI_7～DI_0：8 位数据输入端。

DO_7～DO_0：8 位数据输出端。

STB：选通信号，与 CPU 的地址锁存信号 ALE 相连，当选通信号 STB 产生(由高电平变为低电平)时，8 位输入数据(DI_7～DI_0)被锁入 8 个 D 触发器中。当 STB 为高电平时，锁

存器的输出端随出现在输入端的数据而变化。

$\overline{OE}$：输出允许信号，是由外部输入的控制信号，当$\overline{OE}$有效(为低电平)时，锁存器中的 8 位数据从 DO_7～DO_0 输出送到数据总线上。当$\overline{OE}$为高电平(无效)时，输出端 DO_7～DO_0 呈高阻抗状态，在不带 DMA 控制器的单处理器系统中，$\overline{OE}$信号接地，否则$\overline{OE}$将同 DMA 控制器 8237 的地址允许输出端 AEN 相连接。

在 8086 系列微机中 8282/8283 用作地址锁存器，20 位物理地址，加上$\overline{BHE}$信号也需要锁存(因为在整个总线周期的前半部分$\overline{BHE}$也必须保持有效)，所以共需使用 3 片 8282/8283 作地址锁存器。

CPU 在读/写总线周期的 T_1 状态把 20 位地址和$\overline{BHE}$信号送到系统总线上，在地址锁存允许信号 ALE 有效时，便将 20 位地址和 BHE 信号锁入 8282/8283 中，由于输出允许信号 OE 被固定接地，所以 CPU 输出的地址码和$\overline{BHE}$信号一旦被锁存后，便立即稳定输出在地址总线和控制总线上。8086 系统中也可用 74LS373 作为地址锁存器，其用法与 8282 基本相同，只是选通信号不用 STB，而用 LE 或 G 标示。

2. 最大工作方式

最大工作方式是相对于最小工作方式而言的，将 8086/8088 CPU 的引脚 MN/MX 接地，就使 CPU 工作于最大模式。最大系统用在中、大规模的微机应用系统中，在最大系统下，系统中至少包含两个微处理器，其中一个为主处理器，即 8086CPU，其他的微处理器称之为协处理器，是协助主处理器工作的。

图 2.15 是 8086 最大方式下的基本系统配置，与图 2.11 的最小方式系统配置相比，增加了一个总线控制器 8288。总线控制器 8288 用来产生具有适当定时的总线命令信号和总线控制信号。也就是说，在最大方式下，CPU 不直接产生系统所需的总线控制信号，所有的总线控制信号均由总线控制器 8288 产生。

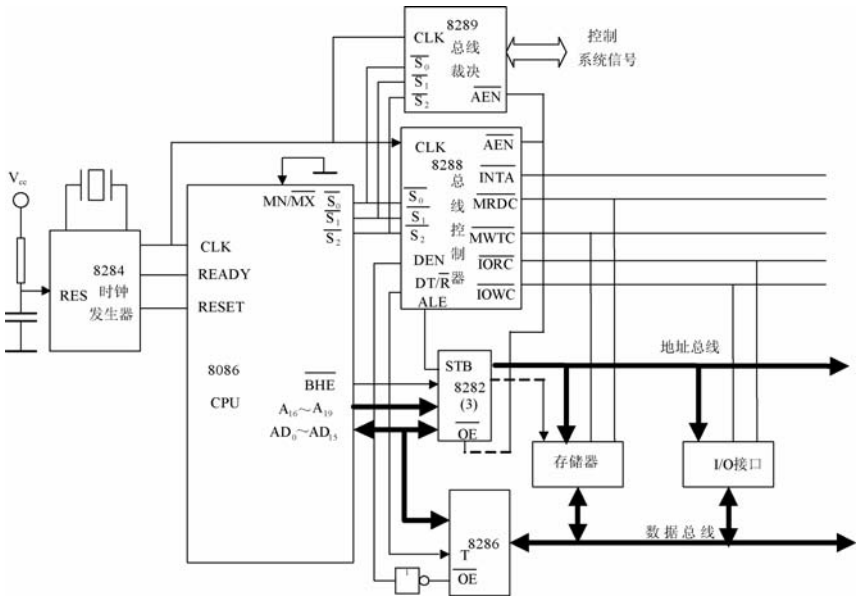
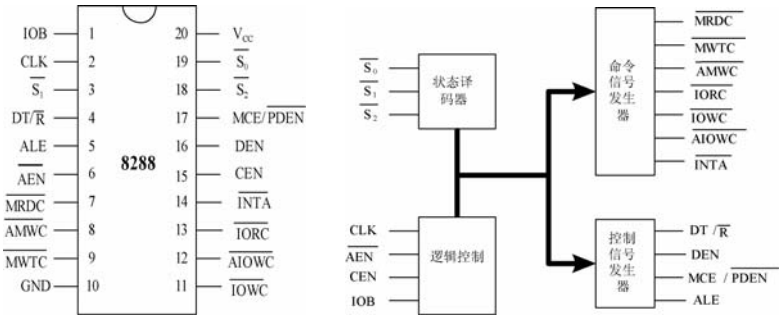


图 2.15　8086 最大系统配置

1) 多处理器系统的基本配置概念

8086 的最大方式是为实现多处理器系统而设计的。该方式支持 3 种基本配置，即协处理器配置，紧耦合配置和松耦合配置。所谓协处理器配置，就是在系统中除主 CPU 8086 或主 CPU8088 外，还接有一个协处理器 8087(数值协处理器)或协处理器 8089(I/O 协处理器)，所谓紧耦合配置，就是在系统中除主 CPU 外还有一个支持处理器，支持处理器可以独立操作，可以控制总线独立于主 CPU 工作；所谓松耦合配置，就是系统中可以配有多个总线主模块(主控处理器)，模块间通过系统总线相连，每个模块都可以成为系统总线的主控者。这些主模块可以是上述协处理器配置，紧耦合配置或是一个由 8086 与另一个能够成为系统主模块的处理器组成。各模块共享总线资源(存储器、I/O 子系统)，各模块也可以有自己的专用存储器或输入/输出设备，可通过各自的局部总线访问自己的子系统，独立地存取全部数据或进行取指令操作，因而可大大提高系统并行处理能力，一般用于大型机或大系统中。在松耦合配置中，各模块间为了解决总线争用和相互通信，每个模块都有一个总线控制器 8288 和一个总线裁决器 8289。8288 产生全部总线控制信号，8289 完成总线使用权的分配，通过总线仲裁电路把总线使用权转让给具有最高优先级的模块或处理器。8289 进行总线仲裁时可有多种仲裁方法，这里就不详述了。关于松耦合配置的多处理机系统，这里不作讨论。下面只对紧耦合配置及 8288 的控制作用作简单说明。

紧耦合配置的结构特点是：CPU 和支持处理器(8288)不仅共享整个存储器和 I/O 子系统，而且还共享一个总线控制器和时钟发生器。在这种配置中，8086 为主 CPU，支持处理器为从处理器。其控制过程是：系统总线的访问控制(即总线使用权的分配)由主 CPU 完成，支持处理器通过请求/允许总线访问控制信号 $\overline{RQ}$ / $\overline{GT0}$ 和 $\overline{RQ}$ / $\overline{GT1}$ 与主 CPU 相连。8086 是主 CPU，平时由它占用着系统总线，当支持处理器要使用总线时，通过 RQ/GT 线向主 CPU 发一负脉冲，请求使用总线，主 CPU 在时钟脉冲的上升沿检测 RQ/GT 线。当测得有请求时，如果允许让出总线，则在当前总线周期结束时的 T_4 状态或空闲状态 T_1 的下降沿，从同一引脚 RQ/GT 回送一个响应信号(负脉冲)，并使自己与总线脱离，(凡与总线有关的信号均无效或高阻态)，从而让出总线使用权。当支持处理器处理完毕，仍通过 RQ/GT 引脚向 CPU 发一个释放总线使用权的负脉冲，CPU 收到此负脉冲又恢复对总线的控制。总线请求/响应负脉冲必须保证宽度为一个时钟周期，否则容易出错。



(a) 引脚图　　(b) 内部结构图

图 2.16　8288 的引脚和内部结构

2) 总线控制器 8288

由上可知，在最大模式下，总线控制器 8288 为了支持上述几种系统配置，必须是以多

总线结构为前提来设计，图 2.16 是 8288 的引脚和内部结构图。其内部结构和输入/输出引脚信号体现了它应完成的总线控制功能。

总线控制器 8288 对 CPU 送来的总线周期状态信号 S_2，S_1，S_0 经其内部状态译码器、命令信号产生电路和控制信号产生电路的综合，并经输入控制信号 $\overline{AEN}$，CEN，IOB 的配合，输出系统所需的总线命令信号和总线控制信号，以实现对总线操作的控制。

(1) 总线命令信号由 CPU 输入的总线状态信号 S_2～S_0 经内部状态译码器译码后，经命令信号产生电路产生总线命令信号。状态信号 S_2～S_0 与 8288 产生的总线命令信号间的对应关系如表 2-4 所示。由表可知，S_2 实际上用来区分是进行存储器传送还是 I/O 传送，而 S_1 用来区分执行的操作是输入(即读)还是输出(即写)。

$\overline{MRDC}$ ——读存储器命令，输出，低电平有效。此信号用来通知内存将所寻址的单元中的内容送数据总线。它相当于最小模式中由 CPU 直接发出的总线控制信号 RD＝0，M/$\overline{IO}$ ＝1 的组合功能。

表 2-4　$\overline{S_2}$, $\overline{S_1}$, $\overline{S_0}$ 状态译码内容

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	操作状态	8288 产生的信号	$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	操作状态	8288 产生的信号
0	0	0	中断相应	$\overline{INTA}$	1	0	0	取指令	$\overline{MRDC}$
0	0	1	读 I/O 端口	$\overline{IORC}$	1	0	1	读存储器	$\overline{MRDC}$
0	1	0	写 I/O 端口	$\overline{IOWC}$ $\overline{AIOWC}$	1	1	0	写存储器	$\overline{MWTC}$ $\overline{AMWC}$
0	1	1	暂停	无	1	1	1	保留	无

$\overline{MWTC}$，$\overline{AMWC}$ ——写存储器命令，输出，低电平有效。此信号通知存储器接收数据总线上的数据，并将数据写入所寻址的单元中。该信号相当于最小模式下 CPU 直接发出的总线控制信号 WR＝0 和 M/IO＝1 的组合。其中 $\overline{AMWC}$ 是提前写存储器命令。它比 $\overline{MWTC}$ 提前一个时钟周期产生，以便一些慢速存储器芯片额外地多一个时钟周期去执行写入操作。

$\overline{IORC}$ ——读 I/O 端口命令，输出，低电平有效。此信号用来通知 I/O 端口，将所寻址的 I/O 端口中的数据送到数据总线。它相当于最小模式下由 CPU 直接发出的总线控制信号 RD＝0 和 M/$\overline{IO}$ ＝0 的组合。

$\overline{IOWC}$，$\overline{AIOWC}$ ——写 I/O 端口命令，输出，低电平有效。此信号用来通知 I/O 端口去接收数据总线上的数据，并将数据写入所寻址的 I/O 端口中。它相当于最小模式下由 CPU 发出的总线控制信号 WR＝0 和 M/IO＝0 的组合。其中 $\overline{AIOWC}$ 是提前写 I/O 端口信号，它比 $\overline{IOWC}$ 提前一个时钟周期出现，以便一些慢速外设可得到一个额外的时钟周期执行写操作。

$\overline{INTA}$ ——中断响应信号，输出、低电平有效。与最小模式下的 $\overline{INTA}$ 信号含义相同，即通知申请中断的外设，中断申请已被响应，将“中断类型码”放在数据总线上。

由上可知，在最大模式下，对存储器的读/写和对 I/O 端口的读/写分别使用了独立的读/写命令；而在最小模式下则是用 M/$\overline{IO}$ 与 RD 或 WR 信号的组合来控制读/写操作的。

(2) 总线控制信号。总线控制信号包括 ALE(地址锁存允许)、DT/$\overline{R}$ (数据发送/接收)、DEN(数据允许)以及 MCE/$\overline{PDEN}$ (主级联允许/外设数据允许)。前 3 种信号的功能和最小模

式下的相应信号相同，只是 DEN 信号的极性相反。所以这里对这 3 个信号就不再解释了。下面只对 MCE/$\overline{\text{PDEN}}$ 进行说明。

MCE / $\overline{\text{PDEN}}$ ——主控级联允许/外设数据允许信号，输出。这是一个具有双重功能的控制信号，其功能与 IOB 信号有关，当 IOB 接地，8288 工作于系统总线方式时，MCE 有效(高电平)，在含有多片中断控制器 8259A 的微机系统中，它在中断响应周期的 T1 状态，可将主 8259A 向从 8259A 输出的地址 CAS_2～CAS_0 进行锁存。当 IOB 接高电平时，8288 工作在 I/O 总线方式，PDEN 有效执行 PDEN 的功能，用来控制外设通过 I/O 总线传送数据。

(3) 控制输入信号。控制输入信号 $\overline{\text{AEN}}$，CEN 和 IOB 都是使 8288 支持多处理器系统时使用的信号。因为在多主控系统情况下，系统中有多个处理器，它们都是总线主模块，每个处理器各自带有 8288 和 8289。这时，系统是一个多总线结构，既有系统总线又有局部 I/O 总线，局部 I/O 总线为 8086/8088(以及 8087 或 8099)所有，系统总线为多个主控 CPU 共享。在这种情况下，8288 既可工作在 I/O 总线方式，也可工作于系统总线方式对总线进行控制。所以这些输入控制信号就是使 8288 能产生适应多处理器情况下所需总线控制信号。因此，对这几个信号的解释涉及到多机系统的一些概念，这里只作简单说明。

IOB ——I/O 总线方式控制信号，输入，高电平有效。8288 既可以控制系统总线，又可控制 I/O 总线，当 IOB 接高电平时，则 8288 工作于 I/O 总线方式，只用来控制 I/O 总线。在这种情况下，不论总线裁决器 8289 的 $\overline{\text{AEN}}$ 信号为何状态，所有的 I/O 命令都处于允许状态，只要 CPU 有 I/O 访问命令，8288 就会立即发出相应的 I/O 读写命令($\overline{\text{IORC}}$，$\overline{\text{MWTC}}$，$\overline{\text{AIOWC}}$ 或 $\overline{\text{IOWC}}$，$\overline{\text{AIOWC}}$)及 PDEN，DT/$\overline{\text{R}}$ 控制信号，I/O 读写信号用于对挂接在 I/O 局部总线上的设备(器件)进行读/写控制，$\overline{\text{PDEN}}$ 和 DT/$\overline{\text{R}}$ 信号用于控制局部 I/O 总线的总线收发器 8286/8287 工作。这时没有任何读/写命令被送入系统总线。

当 IOB 接地时，8288 处于系统总线工作方式。这时 8288 输出的命令信号用于对系统总线上的存储器和 I/O 端口进行读/写控制。在有多个主 CPU 共享系统总线上的存储器和外设资源的情况下，系统中必须使用总线裁决器 8259，8288 的 $\overline{\text{AEN}}$ 引脚(输入信号)受总线裁决器 8289 的控制，只有当 $\overline{\text{AEN}}$ 为低时，才输出总线命令信号和总线控制信号。

当 IOB 接地时，MCE / $\overline{\text{PDEN}}$ 输出 MCE(主级连允许)信号，用于控制多片级连的中断控制器 8259A。

CEN ——命令允许信号，由外部输入，高电平有效。在有多个总线控制器 8288 工作的系统中，必须利用 CEN 控制信号来选择执行当前总线周期应使用哪个 8288，所以这时 CEN 相当于 8288 的片选信号。CEN 有效时，允许 8288 输出全部的总线控制信号和命令信号，CEN 无效时，总线控制信号和命令信号端均呈高阻抗状态。由于在同一个时间内只允许有一个处理器为主模块，所以也只有一片 8288 的 CEN 信号有效。

$\overline{\text{AEN}}$ ——地址允许信号，由总线裁决器 8289 输入，低电平有效。当 $\overline{\text{AEN}}$ 为高电平时，所有总线命令信号引脚为高阻态；当 $\overline{\text{AEN}}$ 为低时，总线命令信号($\overline{\text{MRDC}}$，$\overline{\text{MWTC}}$，$\overline{\text{IORC}}$，$\overline{\text{IOWC}}$)先变为高电平，经一段时间(115μs～200μs)后，其中之一变为有效。$\overline{\text{AEN}}$ 是一个支持多总线结构的控制信号，用作多总线间的同步控制。当 8288 处于 I/O 总线工作方式时，$\overline{\text{AEN}}$ 不影响 I/O 命令线。

2.2.2　8086 的引脚功能

8086 微处理器采用 40 条引脚的双列直插式封装。为减少引脚，采用分时复用的地址/数据总线，因而部分引脚具有两种功能。8086 微处理器具有两种工作方式：最小工作方式和最大工作方式。在两种工作方式下，部分引脚的功能是不同的。

图 2.17 给出了 8086 引脚图。下面先说明 8086 在两种工作方式下公用引脚的定义，然后按工作方式介绍其他引脚的定义。

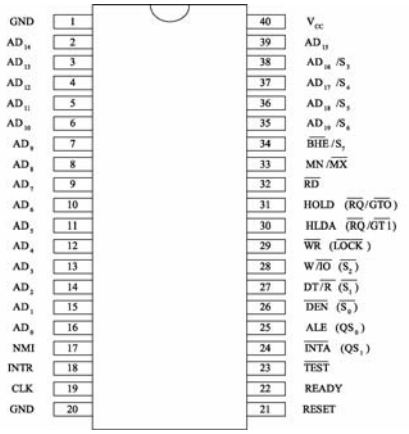


图 2.17　8086 引脚图

括号中的为最大模式时的引脚。

1. 两种工作方式公用引脚

8086 微处理器的引脚构成了微处理器级总线，引脚功能也就是微处理器级总线的功能。在 8086 微处理器的 40 条引脚中，引脚 1 和引脚 20 为接地端(GND)，引脚 40 为电源输入端(V_{CC})，采用的电源电压为+5V±10%；引脚 19 为时钟信号输入端(CLK)。时钟信号占空比为 33%时是最佳状态。其余 36 个引脚按其功能来分，属地址/数据总线的有 20 条引脚，属控制总线的有 16 条引脚。具体定义如下。

1) 地址/数据总线

8086 微处理器有 20 条地址总线，16 条数据总线。为减少引脚，采用分时复用方式，共占 20 条引脚。

(1) AD_{15}～AD_0(Address Data Bus，I/O，三态)为分时复用的地址数据总线。当执行对存储器读写或在 I/O 端口输入输出操作的总线周期的 T_1 状态时，作为地址总线输出 A_{15}～A_0 这 16 位地址，而在其他 T 状态时，作为双向数据总线输入或输出 D_{15}～D_0 这 16 位数据。

(2) A_{19}/S_6～A_{15}/S_3(Address Status Bus，输出，三态)为分时复用的地址/状态信号线。在存储器读写操作总线周期的 T_1 状态输出 4 位地址 A_{19}～A_{16}，对 I/O 端口输入输出操作时，这 4 条线不用，全为低电平。在总线周期的其他 T 状态，这 4 条线用来输出状态信息，但 S_6 始终为低电平；S_5 是标志寄存器(即 PSW)的中断允许标志位 IF 的当前状态；S_4 和 S_3 用来指示当前正在使用的段寄存器，如表 2-5 所示。其中 S_4S_3=10 表示对存储器访问时的段寄存器为 CS，或者表示对 I/O 端口进行访问以及在中断相应的总线周期中读取中断类型号(这两种情况不用段寄存器)。

表 2-5　S_4 和 S_3 的功能

S_4	S_3	段寄存器
0	0	ES
0	1	SS
1	0	CS(或 I/O，中断响应)
1	1	DS

从上面讨论可知，这 20 条引脚在总线周期的 T_1 状态输出地址。为了使地址信息在总线周期的其他 T 状态仍然保持有效，总线控制逻辑必须有一个锁存器，把 T_1 状态输出的 20 位地址进行锁存。

2) 控制总线

控制总线有 16 条引脚。其中 24～31 这 8 条引脚在两种工作方式下定义的功能有所不同，后面将结合工作方式予以讨论。两种工作方式下公用的 8 条控制引脚为：

(1) Non-Maskable Interrupt，NMI 输入。非可屏蔽中断请求信号输入引脚，上升沿有效。当该引脚输入一个由低到高的信号时，CPU 在执行完现行指令后，立即进行中断处理。CPU 对该中断请求信号的响应不受标志寄存器中断允许标志位 IF 状态的影响。

(2) Interrupt Request，INTR 输入。

中断请求信号输入引脚，高电平有效。当 INTR 为高电平时，表示外部有中断请求。CPU 在每条指令的最后一个时钟周期对 INTR 进行测试，以便决定现行指令执行完后是否响应中断。CPU 对可屏蔽中断的响应受中断允许标志位 IF 状态的影响。

(3) Read，$\overline{RD}$ 输出，三态。读控制输出信号引脚，低电平有效，用以指明要执行一个对内存单元或 I/O 端口的读操作，具体是读内存单元，还是读 I/O 端口，取决于控制信号。

(4) Reset，RESET 输入。系统复位信号输入引脚，高电平有效。8088/8086 微处理器要求复位信号至少维持 4 个时钟周期才能起到复位的效果，复位信号输入之后，CPU 结束当前操作，并对处理器的标志寄存器、IP、DS、SS、ES 寄存器及指令队列进行清零操作，而将 CS 设置为 0FFFFH。系统加电或操作员在键盘上进行“RESET”操作时产生 RESET 信号。

(5) Ready，READY 输入。“准备好”状态信号输入引脚，高电平有效，“Ready”输入引脚接收来自于内存单元或 I/O 端口向 CPU 发来的“准备好”状态信号(高电平)，表明内存单元或 I/O 端口已经准备就绪，将在下一个时钟周期将数据置入到数据总线上(输入时)或从数据总线上取走数据(输出时)，无论是读(输入)还是写(输出)，CPU 及其总线控制逻辑可以在下一个时钟周期完成总线周期。若 READY 信号为低电平，则表示存储器或 I/O 端口没有准备就绪，CPU 可自动插入一个或几个等待周期(在每个等待周期的开始，同样对 READY 信号进行检查)，直到 READY 信号有效为止。可见，该信号是协调 CPU 与内存单元或 I/O 端口之间进行信息传送的联络信号。

(6) Test，$\overline{TEST}$ 输入。测试信号输入引脚，低电平有效，TEST 信号与 WAIT 指令结合起来使用，CPU 执行 WAIT 指令后，处于等待状态，当 TEST 引脚输入低电平时，系统脱离等待状态，继续执行被暂停执行的指令。

(7) Minimum/Maximum Model Control(MN/$\overline{MX}$ 输入)，最小/最大工作方式设置信号输入引脚。该输入引脚电平的高、低决定了 CPU 工作在最小工作方式还是最大工作方式，当该引脚接+5V 时，CPU 工作于最小工作方式下；当该引脚接地时，CPU 工作于最大工作方

式下。

(8) Bus High Enable/Status，$\overline{BHE}$ /S_7 输出，三态。它也是一个分时复用引脚。在总线周期的 T_1 状态输出 $\overline{BHE}$，在总线周期的其他 T 状态输出 S_7。S_7 指示状态，目前还没有定义。$\overline{BHE}$ 信号低电平有效。$\overline{BHE}$ 有效表示用高 8 位数据线 AD_{15}～AD_8；否则只使用低 8 位数据线 AD_7～AD_0。$\overline{BHE}$ 和地址总线的 A_0 状态组合在一起表示的功能如表 2-6 所示。同地址信号一样，$\overline{BHE}$ 信号也需要进行锁存。

表 2-6　$\overline{BHE}$ 和 A_0 的代码组合和对应的操作

操　作	$\overline{BHE}$	A_0	使用的数据引脚
读或写偶地址的一个字	0	0	AD_{15}～AD_0
读或写偶地址的一个字节	1	0	AD_7～AD_0
读或写奇地址的一个字节	0	1	AD_{15}～AD_8
读或写奇地址的一个字	0 1	1 0	AD_{15}～AD_8 (第 1 个总线周期放低位数据字节) AD_7～AD_0 (第 2 个总线周期放高位数据字节)

2. 最小方式下引脚定义

当 MN/$\overline{MX}$ 引脚接+5V 时，CPU 处于最小工作方式，引脚 24～31 这 8 条控制引脚的功能如下：

1) $\overline{INTA}$ (Interrupt Acknowledge，输出)

中断响应信号输出引脚，低电平有效，该引脚是 CPU 响应中断请求后，向中断源发出的认可信号，用以通知中断源，以便提供中断类型码，该信号为两个连续的负脉冲。

2) ALE(Address Lock Enable，输出)

地址锁存允许输出信号引脚，高电平有效，CPU 通过该引脚向地址锁存器 8282/8283 发出地址锁存允许信号，把当前地址/数据复用总线上输出的地址信号和 $\overline{BHE}$，锁存到地址锁存器 8282/8283 中去。注意：ALE 信号不能被浮空。

3) $\overline{DEN}$ (Data Enable，输出，三态)

数据允许输出信号引脚，低电平有效，表示 CPU 当前准备发送或接收一项数据。如果系统中数据总线接有双向收发器 8286，该信号作为 8286 的选通信号。

4) DT/$\overline{R}$ (Data Transmit/Receive，输出，三态)

数据收发控制信号输出引脚，CPU 通过该引脚发出控制数据传送方向的控制信号，在使用 8286/8287 作为数据总线收发器时，信号用以控制数据传送的方向，当该信号为高电平时，表示数据由 CPU 经总线收发器 8286/8287 输出，否则，数据传送方向相反。

5) M/$\overline{IO}$ (Memory/Input &Output，输出，三态)

存储器/I/O 端口选择信号输出引脚,这是 CPU 区分进行存储器访问还是 I/O 访问的输出控制信号。当该引脚输出高电平 时，表明 CPU 要进行 I/O 端口的读写操作，低位地址总线上出现的是 I/O 端口的地址；当该引脚输出低电平时，表明 CPU 要进行存储器的读写操作，地址总线上出现的是访问存储器的地址。

6) $\overline{WR}$ (Write，输出，三态)

写控制信号输出引脚，低电平有效，配合实现对存储单元、I/O 端口所进行的写操作控制。

7) HOLD(Hold Request，输入)

总线保持请求信号输入引脚，高电平有效。这是系统中的其他总线部件向 CPU 发来的总线请求信号输入引脚。

8) HLDA(Hold Acknowledge，输出)

总线保持响应信号输出引脚，高电平有效，表示 CPU 认可其他总线部件提出的总线占用请求，准备让出总线控制权。

在最小方式下，$M/\overline{IO}$、$\overline{RD}$ 和 $\overline{WR}$ 的组合根据表 2-7 决定传送类型。

表 2-7　$M/\overline{IO}$、$\overline{RD}$ 和 $\overline{WR}$ 的组合决定的传送类型

$M/\overline{IO}$	$\overline{RD}$	$\overline{WR}$	传送类型
0	0	1	读 I/O 端口
0	1	0	写 I/O 端口
1	0	1	读存储器
1	1	0	写存储器

3. 最大方式下引脚定义

当 8088/8086CPU 的引脚固定接地时，CPU 处于最大模式下，这时的 24～31 共 8 个引脚的名称及功能如下：

1) QS_1、QS_0(Instruction Queue Status，输出)

指令队列状态信号输出引脚，这两个信号的组合给出了前一个 T 状态中指令队列的状态，以便于外部 8086 CPU 内部指令队列的动作跟踪，如表 2-8 所示。

表 2-8　指令队列状态位的编码

QS_1	QS_0	指令队列状态
0	0	无操作，队列中指令未被取出
0	1	从队列中取出当前指令的第一个字节
1	0	队列空
1	1	从队列中取出指令的后续字节

2) $\overline{S_2}$、$\overline{S_1}$、$\overline{S_0}$ (输出，三态)

总线周期状态信号输出引脚，低电平的信号输出端。这 3 组信号组合起来，可以指出当前总线周期中，所进行数据传输过程的类型，总线控制器 8288 利用这些信号来产生对存储单元、I/O 端口的控制信号。$\overline{S_2}$、$\overline{S_1}$、$\overline{S_0}$ 与具体物理过程之间的对应关系，如表 2-4 所示。

需要指出的是，从表 2-4 中可以看出，每一种的组合都对应一个具体的总线操作，除 $\overline{S_2}\overline{S_1}\overline{S_0}$ =111 外，其余都称为有源状态。也就是说，在有源状态(对应前一个总线周期的 T_4 和本总线周期的 T_1 和 T_2 状态)中，$\overline{S_2}$ $\overline{S_1}$ $\overline{S_0}$ 至少有一个信号为 0，当 $\overline{S_2}\overline{S_1}\overline{S_0}$ =111 时(对应总线周期的 T_3 和 T_W 且 READY＝1)，也就是一个总线操作即将结束，另一个总线周期还未开始时，称为无源状态，很显然，这时 $\overline{S_2}$ $\overline{S_1}$ $\overline{S_0}$ 中任一信号的改变，都意味着一个新的总线周期的开始。

3) $\overline{LOCK}$ (Lock，输出，三态)

总线封锁输出信号引脚，低电平有效，当该引脚输出低电平时，系统中其他总线部件就不能占用系统总线。此信号是由指令前缀 LOCK 产生的，在 LOCK 前缀后面的一条指令执行完毕之后，便撤消该信号。此外，在 8086 的 2 个中断响应脉冲之间，信号也自动变为有效的低电平，以防止其他总线部件在中断响应过程中占有总线而使一个完整的中断响应过程被中断。

4) $\overline{RQ}/\overline{GT_1}$ 、$\overline{RQ}/\overline{GT_0}$ (Request/Grant，输入/输出)

总线请求信号输入/总线允许信号输出引脚。这两个信号端可供 CPU 以外的两个处理器，用来发出使用总线的请求信号和接收 CPU 对总线请求信号的应答。这两个引脚都是双向的，请求与应答信号在同一引脚上分时传输，方向相反。其中 $\overline{RQ}/\overline{GT_1}$ 比 $\overline{RQ}/\overline{GT_0}$ 的优先级高。

在最大方式系统中，对存储器和 I/O 端口进行读写的命令信号和对 8282、8286 的控制信号均由 8288 产生。现对这些信号分别说明如下：

1) 用于对地址锁存器和数据收发器的控制信号

在 ALE、$DT/\overline{R}$ 和 DEN 中，ALE 和 $DT/\overline{R}$ 信号的功能和定时波形与最小方式下 CPU 直接产生的相应信号相同；而 DEN 信号的功能同最小方式下 CPU 直接产生的 $\overline{DEN}$ 信号相同，不同之处是极性相反，所以经过反向后作为数据收发器的 $\overline{OE}$ 控制信号。

2) 用于系统控制总线的命令信号

(1) $\overline{INTA}$：向中断控制器或中断设备输出的中断响应信号。

(2) $\overline{IORC}$：I/O 读命令，指示 I/O 端口把被访问的 I/O 端口中的数据放到系统数据总线上。

(3) $\overline{IOWC}$：I/O 写命令，指示 I/O 端口接受系统数据总线上的数据，并将其写入被访问的 I/O 端口内。

(4) $\overline{MRDC}$：存储器读命令，指示存储器把被访问的存储单元中的数据放到系统数据总线上。

(5) $\overline{MWTC}$：存储器写命令，指示存储器接受系统数据总线上的数据，并将其写入被访问的存储单元中。

另外，8288 还输出 $\overline{AIOWC}$ (先行 I/O 写命令)和 $\overline{AMWC}$ (先行存储器写命令)。这两个命令信号除了提前一个时钟周期输出外，分别与 $\overline{IOWC}$ 和 $\overline{MWTC}$ 一样。这样，就使慢速端口能增加一个时钟周期来准备写入数据。这两个信号不能用于多总线结构。

在 8086 最大方式系统中，系统总线中的地址总线和数据总线与最小方式系统相同。控制总线有 $\overline{BHE}$、$\overline{IORC}$、$\overline{IOWC}$、$\overline{MRDC}$、$\overline{MWTC}$、$\overline{LOCK}$、$\overline{RQ}/\overline{GT_1}$、$\overline{RQ}/\overline{GT_0}$、$\overline{INTA}$、INTR、NMI、$\overline{TEST}$、READY 和 RESET 等。

2.3　8086 的总线周期

2.3.1　总线周期的基本概念

在微机系统中，CPU 的操作都是在系统主时钟 CLK 的控制下按节拍有序进行的。按照一般的概念，CPU 执行一条指令的时间(包括取指令和执行完该指令所需的全部时间)称为一个指令周期。在指令周期内，通常需要对总线上的存储器或 I/O 端口进行一次或多次

读/写(访问)操作，这里把通过外部总线对存储器或 I/O 端口进行一次读/写操作的过程称为总线周期。因此，一个指令周期由若干个总线周期组成。而一个总线周期由若干时钟周期 T 组成。时钟周期也就是系统主时钟频率的倒数，是 CPU 的基本时间计量单位，例如，某 CPU 的主频为 5MHz，则其一个时钟周期就是 200ns，若主频为 10MHz，则一个时钟周期为 100ns。

在 8086 CPU 中，所有的外部操作(读/写存储器或 I/O 端口)都是由总线端口部件 BIU 通过系统总线完成的。因此，把 BIU 完成一次对存储器或 I/O 端口的读/写操作所需要的时间称为一个总线周期。8086 CPU 的一个基本总线周期由 4 个时钟周期(T_1，T_2，T_3，T_4)组成，时钟周期也称为时钟状态，即 T_1 状态、T_2 状态、T_3 状态和 T_4 状态。每一个时钟周期(时钟状态)内完成一些基本操作。例如：

在 T_1 状态，CPU 往数据/地址多路复用总线上发出访问存储器或 I/O 端口的地址信息。所谓数据/地址多路复用总线就是地址、数据在同一总线上分时传送。8086/8088 CPU 中正是利用了这种多路分时复用技术，才能用 40 个引脚实现了众多地址、数据及控制信号的传输。

在 T_2 状态，CPU 从总线上撤销地址，若为读周期，使数据/地址多路复用总线的低 16 位处于高阻抗状态，以便 CPU 有足够的时间从输出地址方式转变为输入数据方式，接着在 T_3～T_4 期间，CPU 从总线上接收数据。总线的高 4 位(A_{19}～A_{16})用来输出本总线周期状态信息，这些状态信息包括中断允许状态和当前正在使用的段寄存器名等。若为写周期，由于输出数据和输出地址都是写总线过程，因而不需要缓冲时间，CPU 在 T_2～T_4 期间把数据放到总线上。

在 T_3 状态，数据/地址多路复用总线的高 4 位继续传送周期状态信息，而多路复用线的低 16 位上出现由 CPU 输出的数据(为写周期)或为 CPU 从存储器或 I/O 端口读入的数据。在 T_3 时，数据在 CPU 和存储器或 I/O 端口间传送。

在 T_4 状态，8086 完成数据传送，使控制信号变为无效，结束总线周期。

需要指出的是：

(1) 上面所说的一个总线周期由 4 个时钟周期组成。这是指最基本的总线周期，实际上有时在一个基本总线周期的 4 个 *T* 内并不能完成一次读/写操作，还需要增加数量不定的附加状态。例如，当存储器或 I/O 端口在数据传输过程中不能及时配合 CPU 的操作，则要在总线周期的 T_3 和 T_4 之间插入一个或若干个等待状态 T_W。这时一个总线周期就不只 4 个时钟周期。另外，在完成一个总线周期后，如果不立即执行下一个总线操作(如字指令队列是满的，EU 又无完成操作请求)，这时 BIU 便进入空闲状态(用 T_i 表示)，一个空闲状态占一个时钟周期的时间。

(2) 根据总线周期的定义，只有当 BIU 要访问存储器或 I/O 端口时，才需要执行总线周期，也就是说总线周期是根据要求才会出现的。图 2.18 给出了 8086 CPU 典型总线周期序列。

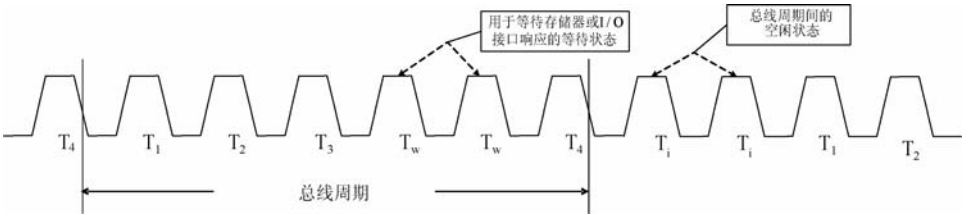


图 2.18　8086 CPU 典型总线周期

2.3.2　8086 的存储器读写周期

8086 微处理器的操作是由指令译码器输出的电位和外部输入的时钟信号联合作用并在由此而产生的各个命令控制下进行的，可分为内操作与外操作两种，内操作控制算术逻辑运算单元 ALU 进行算术逻辑运算，控制寄存器组进行寄存器选择以及判断是送往数据线还是地址线，进行读操作还是写操作等，所有这些操作都在 CPU 内部进行，用户可以不必关心。CPU 的外部操作是系统对 CPU 的控制或是 CPU 对系统的控制，用户必须了解这些控制信号以便正确使用。

8086 微处理器的外部操作主要有以下几种：① 存储器读/写；② I/O 端口的读/写；③ 中断响应；④ 总线保持(最小方式)；⑤总线请求/允许(最大方式)；⑥ 复位和启动。

本节主要介绍存储器的读/写周期。由于 8086 微处理器可以工作在两种不同的工作方式下，因此，对存储器的读/写也表现不同的时序，下面将针对不同工作方式来讨论它们的存储器读写周期。

1. 最小方式下的存储器读写周期

1) 存储器读周期

当 8086 微处理器进行存储器读操作时，便进入存储器读周期。8086 的存储器读周期时序如图 2.19 和 2.20 所示，其中图 2.20 为具有等待周期的存储器读周期时序。由图可知，基本的读周期由 4 个 T 周期组成：T_1、T_2、T_3 和 T_4。当选中的存储器的存取速度较慢时，则在 T_3 和 T_4 之间插入一个或多个等待周期 T_W。

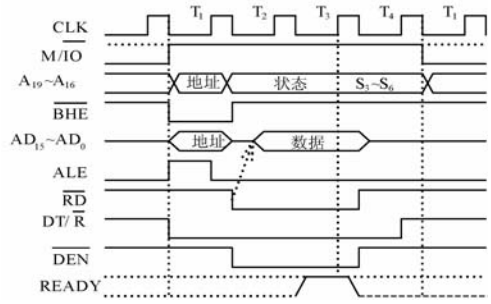


图 2.19　8086 的存储器读周期时序

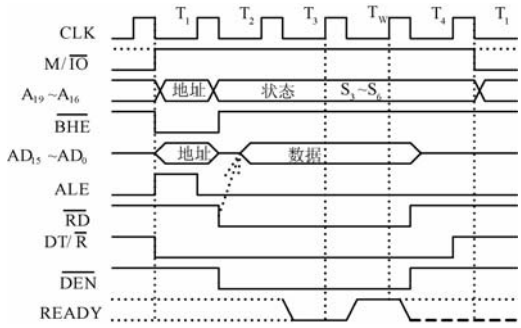


图 2.20　具有等待周期的存储器读周期时序(最小方式)

在 8086 读周期内，有关总线信号在各个 T 状态的变化如下：

(1) T_1 状态。

① $M/\overline{IO}$ 信号首先在 T_1 状态变为有效的高电平状态，用以指出 CPU 本次是进行存储器读操作。且 $M/\overline{IO}$ 信号在整个读总线周期内保持有效。

② 将访问存储器的 20 位物理地址通过多路复用总线输出，其中 20 位地址的高 4 位从 A_{19}/S_6～A_{16}/S_3 地址/状态复用线输出，低 16 位从 AD_{15}～AD_0 地址/数据复用线输出。

③ 地址 ALE 锁存信号有效，即 T_1 状态从 ALE 引脚输出一个正向脉冲，并用 ALE 的下降沿作为地址锁存器 8282/8283 的选通信号，对地址进行锁存。地址锁存以后，这些引脚才可在其他状态被分时复用为数据或状态信息的传送。

④ 高 8 位数据有效信号 $\overline{BHE}/S_7$ 有效，以实现对存储器高字节(即奇地址)的寻址，偶

地址的选体信号为 A_0。$\overline{BHE}$ 信号在 T_1 状态由 ALE 的下降沿锁入 8282/8283。

⑤ 若系统中接有数据总线收发器 8286/8287 时，为了控制数据传送方向，在 T_1 状态，$DT/\overline{R}$ 信号变为低电平，以控制 8286/8287 处于接收数据状态。

(2) T_2 状态。

① CPU 开始撤销地址，A_{19}/S_6～A_{16}/S_3；$\overline{BHE}/S_7$ 引脚开始输出状态信息 S_7～S_3，且一直持续到 T_4。对 8086，S_7 并未赋予实际意义。

② 低位地址线 AD_{15}～AD_0 开始进入高阻抗状态，为读入数据做准备。

③ 若系统中有 8286/8287，则 $\overline{DEN}$ 信号在 T_2 状态开始有效(为低电平)，使 8286/8287 在数据总线上出现输入数据之前(即在 T_3 之前)就处于输出允许状态，以便数据通过 8286/8287 进入 CPU，DEN 的低电平一直维持到 T_4 状态的中期结束。

④ $\overline{RD}$ 信号开始有效(变为低电平)，使被寻址的存储单元或 I/O 端口将数据送入数据总线。

(3) T_3 状态。

① CPU 检测 READY 信号。

经过 T_1，T_2 状态后，如果存储器能及时提供数据(READY 信号为高)，则在基本总线周期的 T_3 状态就将数据送到数据总线上，CPU 通过 AD_{15}～AD_0 接收数据。若存储器不能及时提供数据(READY 信号为低)，则 CPU 将在 T_3 状态的结束时刻(下降沿)插入 T_W 等待状态。因此，在 T_3 状态的一开始(下降沿)，CPU 便检测 READY 信号(READY 信号是通过时钟发生器 8284 送入 CPU 的 READY 引脚的)，若 READY 为低，表示存储器未准备好数据，则 CPU 在 T_3 和 T_4 之间插入等待状态 T_W。以延长总线周期。在每个等待状态内，总线上的活动与 T_3 周期相同。若 READY 为高，则说明数据已准备好，不用插入等待状态，在 $\overline{DEN}=0$，$DT/\overline{R}=0$ 的配合控制下，内存单元的数据通过数据收发器 8286/8287 送到数据总线 AD_{15}～AD_0 上。CPU 在 T_3 状态结束时读取数据。这时由状态信号 $\overline{S_4}$、$\overline{S_3}$ 可知当前读取的是指令还是数据，若 $\overline{S_4}\,\overline{S_3}=10$，表示访问 CS 段，读取的是指令，CPU 将它送入指令队列等待 EU 执行。否则读取的是数据，进入 ALU 去进行运算。

② CPU 在每个 T_W 状态的前沿对 READY 信号进行采样，当 READY 为低电平，则继续插入 T_W 状态。当采样到 READY 为高电平时，则在当前 T_W 状态执行完便进入 T_4 状态。在最后一个 T_W 状态数据已经稳定在数据总线上，CPU 在 T_W 状态结束时读取数据。在整个 T_W 状态期间，其他控制信号保持与 T_3 状态时相同。

(4) T_4 状态。

CPU 在 T_3 与 T_4 状态的交界处采样数据总线 AD_{15}～AD_0，完成读取数据操作，在 T_4 的后半周，数据从数据总线上撤消。各控制信号和状态信号线进入无效状态，$\overline{DEN}$ 无效，总线收发器不工作，一个总线读周期结束。

2) 存储器写周期

当 8086 CPU 进行存储器写操作时，便进入存储器写周期。8086 的存储器写周期时序如图

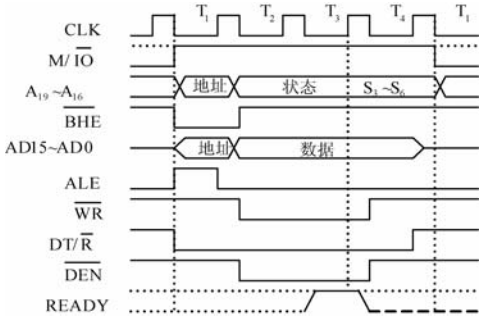


图 2.21　8086 存储器写周期时序(最小方式)

2.21 所示。由图可知，总线写操作的时序与前述的总线读操作有许多相同之处。

与读周期一样，存储器基本写周期也包含 4 个时钟周期 T_1，T_2，T_3 和 T_4。当存储器速度较慢时，在 T_3 和 T_4 之间插入等待状态 T_W。

在 T_1 状态，$M/\overline{IO}$ 信号为有效高电平，指示出 CPU 的数据是写入存储器内的；对于地址的传送过程与读周期完全相同；ALE 信号有效，地址将被锁存；选体信号 $\overline{BHE}$、A_0 有效，$DT/\overline{R}$ 变为高电平(因为是写操作，故应控制 8086/8087 为发送状态)。

在 T_2 状态，地址撤销，地址/状态线上输出状态信号 $\overline{S_6}$ ～ $\overline{S_3}$；CPU 将数据送入数据总线 AD_{15}～AD_0，写信号 $\overline{WR}$ 为有效低电平，$\overline{DEN}$ 信号有效，它作为数据总线收发器 8286/8287 的选通信号。

在 T_3 状态，CPU 采样 READY 引脚，若 READY 信号为低电平，则在 T_3 结束时插入等待状态 T_W，直到 READY 变为高电平为止，存储器从数据总线上取走数据。

在 T_4 状态，从数据线上撤销数据。各控制信号和状态信号变成无效，$\overline{DEN}$ 为高电平，使总线收发器 8286/8287 不工作，结束写周期。

总线写周期也有几点与读周期不同：

(1) 在 T_1 状态，$DT/\overline{R}$ 为高电平，表示本周期是写操作，用 DT 去控制总线收发器 8286/8287 发送 CPU 输出的数据到数据总线，以便写入存储器。

(2) 送到存储器的控制信号是写信号 $\overline{WR}$，而不是读信号 $\overline{RD}$，但它们出现时序一样，也是从 T_2 开始，低电平持续到 T_4 的前半周。

(3) 在写周期下，由 CPU 从地址/数据线上输出的地址和输出的数据是同方向的，因此，在 T_2 状态，地址一旦输出被锁存后 CPU 便立即向地址/数据线 AD_{15}～AD_0 上输出数据，而不再需要像读周期时那样，要维持一个时钟周期的浮空状态作缓冲。数据信号要保持到 T_4 状态的中间。

2. 最大方式下的存储器读写周期

8086 CPU 在最大方式下的存储器操作也是包括存储器读和存储器写两种操作，但在最大方式时，由于增设了总线控制器 8288，总线控制信号不再由 CPU 直接输出，而是由总线控制器根据 CPU 给出的状态信号 $\overline{S_2}$ ～ $\overline{S_0}$ 进行综合后产生，因此在分析操作时序时要考虑 CPU 和总线控制器 8288 两者产生的控制信号。

1) 最大方式存储器读周期

最大方式下的存储器读周期时序如图 2.22 所示。图中带*号的信号是由总线控制器 8288 根据 CPU 的 $\overline{S_2}\,\overline{S_1}\,\overline{S_0}$ 组合产生的，其交流特性要比 CPU 直接产生的相同信号好得多，因此在系统连接时，一般都采用 8288 输出的信号。

由图可知，最大方式下的存储器读周期时序与前述的最小方式下的读周期时序相类似，所不同的只有以下几点：

(1) 在每个总线周期开始之前的一段时间，$\overline{S_2}$，$\overline{S_1}$，$\overline{S_0}$ 必定被置为高电平，即 $\overline{S_2}\,\overline{S_1}\,\overline{S_0}$ ＝111。当总线控制器 8288 一旦检测到 $\overline{S_2}$，$\overline{S_1}$，$\overline{S_0}$ 中任何一个或几个从高电平变为低电平，便立即开始一个新的总线周期。例如，当 $\overline{S_2}\,\overline{S_1}\,\overline{S_0}$ ＝101，进入读存储器总线周期。

(2) 最小方式下由 CPU 直接产生的 ALE，$\overline{RD}$，$DT/\overline{R}$，$\overline{DEN}$ 等控制信号，在最大模式下由总线控制器 8288 产生，在图 2.22 中分别用 ALE*，$\overline{MRDC}/\overline{IORC}$ *，$DT/\overline{R}$ *，DEN*

表示，它们的时序关系见图 2.22 所示。

(3) 在最大方式下，读存储器用，$\overline{\text{MRDC}}$ / $\overline{\text{IORC}}$ *信号表示，而不是像最小方式中用 M/$\overline{\text{IO}}$ 和 $\overline{\text{RD}}$ 信号的组合来表示。

(4) 在读周期的 T_3 状态，当 CPU 读取总线上的数据后，$\overline{S_2}$，$\overline{S_1}$，$\overline{S_0}$ 便全部变为高电平($\overline{S_2}$ $\overline{S_1}$ $\overline{S_0}$ =111)，即进入无源状态，并一直保持到 T_4 状态。一旦进入无源状态就意味着很快可以启动一个新的总线周期。

(5) 等待状态 T_W 的插入过程与最小模式时相同。

(6) 在 T_4 状态，数据从总线上消失，状态信号引脚 S_7～S_3 进入高阻抗状态，而 $\overline{S_2}$，$\overline{S_1}$，$\overline{S_0}$ 则按照下一个总线周期的操作类型产生变化(即 $\overline{S_2}$ $\overline{S_1}$ $\overline{S_0}$ =000～110)。

2) 最大方式存储器写周期

最大方式下的存储器写周期要完成的功能也是要将 CPU 输出的数据写入指定的存储器单元。写周期的时序如图 2.23 所示。图中凡是带*号的信号都是由 8288 产生的。

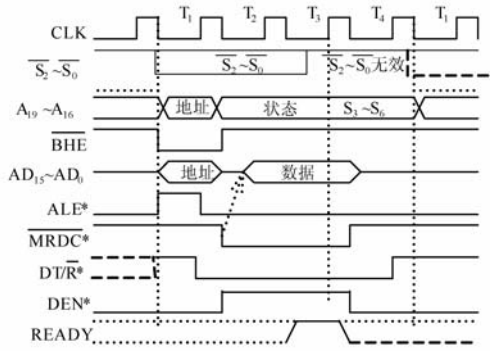


图 2.22　8086 存储器读周期时序(最大方式)

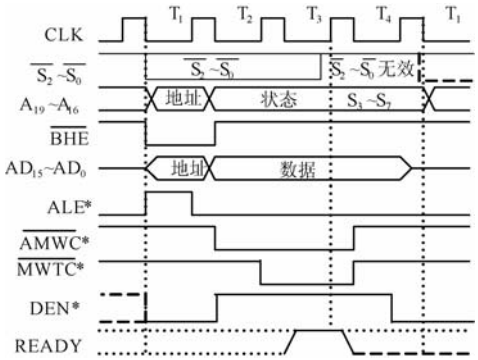


图 2.23　存储器写周期时序(最大方式)

由图可知，最大方式下的写周期时序与前述的读周期时序有很多相同之处，例如：

(1) 和读周期一样，在总线写周期开始之前，$\overline{S_2}$，$\overline{S_1}$，$\overline{S_0}$ 就已经按照操作类型设置好相应的电平。$\overline{S_2}$，$\overline{S_1}$，$\overline{S_0}$ 在各个 T 状态中的变化情况与最大方式下该周期中的变化是一样的。同样，也在 T_3 状态全部恢复为高电平，进入无源状态，从而为启动下一个新的总线周期作准备。

(2) ALE*和 DEN*的时序和作用与读周期相同；状态/地址信号 A_{19}/S_6～A_{16}/S_3 及 $\overline{\text{BHE}}/\overline{S_7}$ 在各个 T 状态中的变化与读周期也相同。

(3) 同样，在最大方式下的写周期中，当存储器速度较慢时，也可以用 READY 信号联络，当在 T_3 开始时 READY 信号仍无效(即为低电平)，也可在 T_3 和 T_4 之间插入 1 个或几个等待状态 T_W。

写周期时序与读周期时序所不同的是：

(1) 在最大方式下的存储器写周期中，CPU 通过总线控制器 8288 为存储器提供两组写信号：一组是普通的写信号 $\overline{\text{MWTC}}$ *，该信号从 T_3 状态开始有效，保持到 T_4 状态；另一组是提前一个时钟周期的写信号 $\overline{\text{AMWC}}$ *，该信号从 T_2 状态开始有效，保持到 T_4 状态。提前的写信号 $\overline{\text{AMWC}}$ *比普通的写信号提前一个时钟周期有效，这样可使慢速的存储器有足够的时间进行写操作。

(2) DT/$\overline{R}$ *信号为高电平，表示本总线周期是写操作，数据总线收发器 8286/8287 应处于发送状态。

2.3.3　8086 的 I/O 读写周期

I/O 读写周期的时序如图 2.24～2.28 所示，与存储器读/写周期的时序基本相同。不同之处在于：

(1) 一般 I/O 端口的工作速度较慢，因而需插入等待周期 T_w。

(2) T_1 期间只发出 16 位地址信号，即 A_{15}～A_0，A_{19}～A_{16} 为 0。

(3) 在最小方式下，M/$\overline{IO}$ 的信号由低电平取代原来的高电平，以指示 CPU 是对 I/O 端口操作。

(4) 在最大方式下，8288 发出的读/写命令为 $\overline{IORC}$、$\overline{AIOWC}$ 和 $\overline{IOWC}$，而非存储器读写时的 $\overline{MRDC}$、$\overline{AWMC}$ 和 $\overline{MWTC}$。

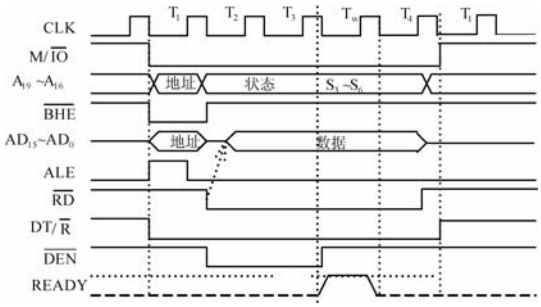


图 2.24　I /O 端口读周期时序(最下方式)

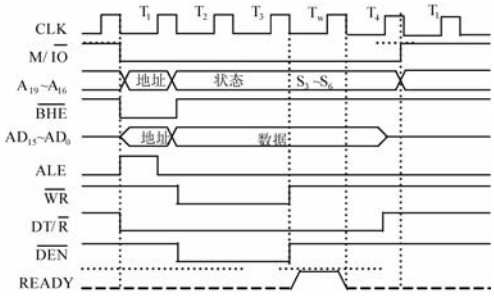


图 2.25　I/O 端口写周期时序(最小方式)

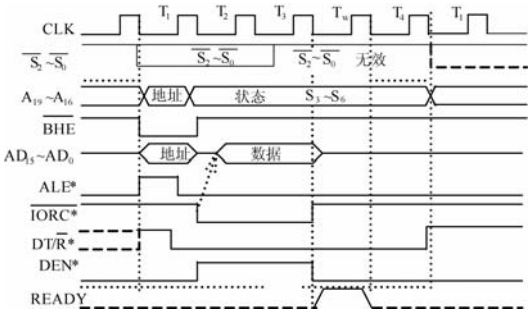


图 2.26　I/O 端口读周期时序(最大方式)

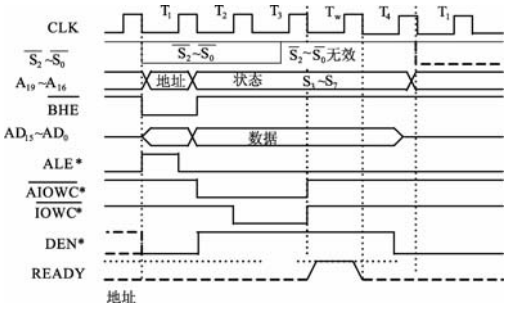


图 2.27　I/O 端口写周期时序(最大方式)

关于 I/O 端口的读写周期，本节只给出两种工作方式下的时序图，其时序分析读者可参照存储器的读写周期进行对比分析。

2.3.4　8086 其他典型时序分析

8086 CPU 的外部操作除存储器和 I/O 端口的读写操作外，还有中断响应、最小方式下的总线保持、最大方式总线请求/允许、复位和启动等操作。这些操作都是 CPU 在系统主时钟信号 CLK 的控制下按序一步步执行的，了解这些典型操作的时序也是理解和设计微机应用系统的基础。本节将对 8086 的一些典型操作时序进行讨论分析，以加强对 8086 系统的理解。

1. 中断响应操作

当8086 CPU的INTR引脚上有一有效电平(高电平)，且标志寄存器中IF=1，则8086 CPU在执行完当前指令后，响应中断。在响应中断时 CPU 执行两个中断响应周期，如图 2.28所示。

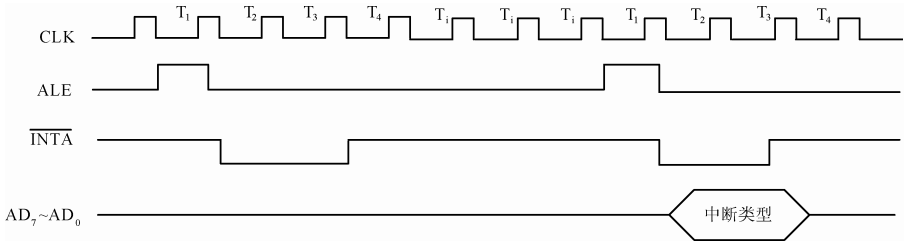


图 2.28　中断响应周期时序

每个中断响应周期由4个T周期组成。在第一个中断响应周期中，从T_2～T_4周期，$\overline{INTA}$为有效低电平，作为对中断请求设备的中断响应；在第二个中断响应周期中，同样从T_2～T_4周期，$\overline{INTA}$为有效低电平，该输出信号通知中断请求设备(通常是通过中断控制器)，把中断类型号(决定中断服务程序的入口地址)送到数据总线的低 8 位 AD_7～AD_0(在T_2～T_4期间)。在两个中断响应周期之间，有 3 个空闲周期(T_i)。

2. 最小方式下的总线保持

在一个具有多个总线主模块的系统中，总线控制权一般总是由 CPU 占用。当 CPU 以外的其他总线主模块(如 DMA 控制器)需要使用总线时，需向 CPU 发出总线请求信号，CPU收到此请求信号后，若同意让出总线控制权，就向发出总线请求的其他主模块发响应信号。

8086 微处理器提供了一对专用于最小方式下总线使用权转让的总线控制联络信号HOLD 和 HLDA。当 CPU 以外的其他总线主模块(主要是 DMAC)要求获得总线使用权时，就向 CPU 发出总线保持请求信号 HOLD，CPU 在每个时钟周期的上升沿检测 HOLD 引脚，如果检测到 HOLD 引脚为高电平(有效状态)，并且允许让出总线，则在总线周期的T_4状态或空闲状态T_i之后的下一个时钟周期，由 HLDA 引脚发出总线响应信号 HLDA(为高电平)，并且让出总线控制权，直到 HOLD 信号变为无效(变为低电平)，即其他主模块使用完总线交出总线控制权后，CPU 才收回总线控制权。图 2.29 所示为最小方式下的总线请求和总线响应的时序图。

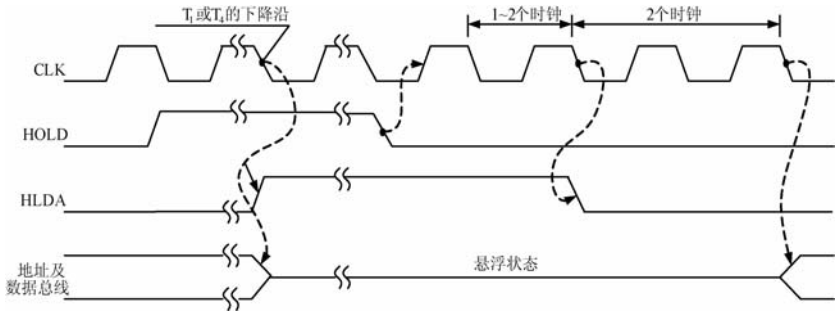


图 2.29　总线请求和响应时序(最小方式)

由图可知：

(1) 当 HOLD 信号变为高电平后，CPU 要在下一时钟周期的上升沿才检测到 HOLD 的高电平。若随后的时钟周期正好为 T_4 或 T_1，则在其下降沿使 HLDA 变为高电平，即发出响应信号；若 CPU 检测到 HOLD 后不正好是 T_4 或 T_1 状态，则可能会延迟几个时钟周期，再等到 T_4 或 T_1 状态时才发出 HLDA 信号(即使之为高电平)，表示让出总线。

(2) 当 8086 一旦让出总线控制权，便将具有三态输出的地址线、数据线和控制线(AD_{15}～AD_0，A_{19}/S_6～A_{16}/S_3，$M/\overline{IO}$，$DT/\overline{R}$，$\overline{DEN}$，$\overline{RD}$，$\overline{WR}$ 和 $\overline{INTA}$)都置于浮空状态，但地址锁存信号 ALE 不浮空。

(3) 在总线请求/响应周期中，因总线浮空，故将直接影响 8086 微处理器中总线端口部件 BIU 的工作，但执行部件 EU 将继续执行指令队列中的指令，直到遇到需要访问总线的指令时，EU 才停下来。当然，当把指令队列中的指令全执行完，EU 也会停下来。由此可见，CPU 和获得总线控制权的其他主模块之间在操作上有一段小小的重叠。

(4) 当 HOLD 变为无效(低电平)后，CPU 也接着在 CLK 的下降沿将 HLDA 信号变为低电平。但是，CPU 并不立即重新驱动已变为浮空的地址总线、数据总线和控制线，而是使这些引脚继续浮空，直到 CPU 需要执行一个新的总线操作周期时，才结束这些引脚的浮空状态。这样，就可能会出现一种情况，即在总线控制权切换的某一小段时间中，没有任何一个主模块驱动总线，而使控制线电平漂移到最小电平以下。为此，在控制线和电源之间应连接一个上拉电阻。

3. 最大方式下的总线请求/允许

8086 CPU 在最大方式下，也提供了总线主模式之间传递总线控制权的联络信号，但不是 HOLD 和 HLDA，而是两个具有双向传输信号功能(即总线请求和总线响应两信号都从同一引脚传送)的引脚 $\overline{RQ}/\overline{GT_0}$ 和 $\overline{RQ}/\overline{GT_1}$。称为总线请求/总线允许信号端，两个信号可以分别同时连接两个除 CPU 以外的其他总线主模块(即协处理器、DMA 控制器)。其中 $\overline{RQ}/\overline{GT_0}$ 的优先级比 $\overline{RQ}/\overline{GT_1}$ 高，也就是说，当与 $\overline{RQ}/\overline{GT_0}$ 和 $\overline{RQ}/\overline{GT_1}$ 相连接的两个其他主模块同时发出总线请求时，CPU 会先在 $\overline{RQ}/\overline{GT_0}$ 引脚上发出允许信号，等到 CPU 再次得到总线控制权后，才会响应 $\overline{RQ}/\overline{GT_1}$ 引脚上的请求。当然，如果 CPU 已经把总线控制权交给了与 $\overline{RQ}/\overline{GT_1}$ 相连接的主模块，此时又在 $\overline{RQ}/\overline{GT_0}$ 引脚上收到另一个主模块的总线请求，则要等前一个主模块释放总线之后，CPU 收回了总线控制权，才会去响应 $\overline{RQ}/\overline{GT_0}$ 引脚上的总线请求。由此可见，CPU 对总线请求的处理是不允许“嵌套”的，这与 CPU 对中断请求的处理不同。

8086 CPU 在最大方式下的总线请求/允许/释放操作的时序如图 2.30 所示。

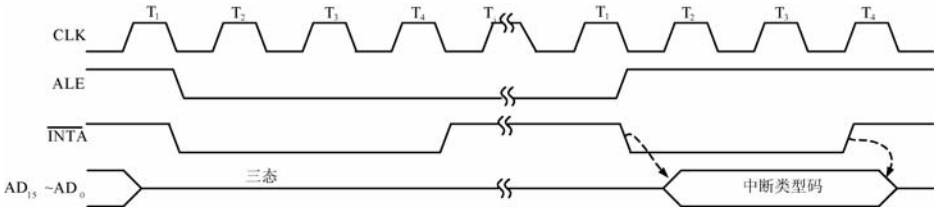


图 2.30　总线请求/允许时序(最大方式)

对于最大模式下的总线请求/允许/释放时序，有几点需要说明：

(1) 当CPU以外的其他总线主模块请求使用总线时，从RQ/GT(即 $\overline{RQ}/\overline{GT_0}$ 或 $\overline{RQ}/\overline{GT_1}$)引脚上向CPU发一个负脉冲RQ，脉冲宽度为一个时钟周期。

(2) CPU在每个时钟周期的上升沿检测RQ/GT引脚，看外部是否输入一个负脉冲RQ信号，若检测到外部输入的总线请求负脉冲，则在下一个 T_4 状态或 T_i 状态从同一引脚RQ/GT上向发出总线请求信号的主模块发一个允许脉冲GT，它也是一个负脉冲，宽度也是一个时钟周期。CPU一旦发出响应脉冲后，各地址/数据引脚、地址/状态引脚以及控制线RD，LOCK，S_2～S_0，BHE/S_7便处于高阻状态。于是CPU在逻辑上与总线断开。

(3) 其他总线主模块(协处理器、DMA控制器)收到CPU发出的允许脉冲GT后，便得到了总线控制权，于是它可以占用总线一个或几个总线周期。当使用总线完毕，其他总线主模块从RQ/GT引脚上向CPU发一个释放脉冲，释放负脉冲宽度为一个时钟周期。CPU检测到此释放负脉冲后，在下一个时钟周期便收回总线控制权。

(4) 从时序图中可以见到，每次总线控制权的切换都是通过3个环节实现的：其他总线主模块发总线请求，CPU发允许脉冲，其他总线主模块使用完总线后发释放脉冲。而且这3个脉冲均为负脉冲，宽度均为一个时钟周期。但是其传输方向不同。

(5) CPU响应总线请求(发GT脉冲)实际上是有条件的，当CPU正访问存储器或I/O端口时；CPU正在用低8位数据线传送数据；当CPU正在执行中断响应的第一个总线周期时；当CPU正在执行总线封锁指令时，若有总线请求，CPU均不予响应，即总线请求无效。由此可见，只有在总线空闲时收到总线请求，CPU才会在下一个时钟周期发出总线允许信号。

(6) 和最小模式下的总线保持请求/总线保持响应一样，在总线响应期间CPU虽然暂时与总线脱离，但CPU内部EU仍可执行指令队列中的指令，直到需要使用一个总线周期为止。同样，当CPU收到其他总线主模块发出的释放负脉冲后，也不立即驱动总线，所以在 $\overline{RQ}/\overline{GT_0}$ ， $\overline{RQ}/\overline{GT_1}$ 与电源间应接上拉电阻。如果这两个引脚不用，则可悬空。

4. 系统的复位和启动

8086 CPU的复位和启动是由时钟发生器8284向CPU的RESET引脚输入一个复位触发信号RESET来实现的。8086要求复位信号RESET至少维持4个时钟周期的高电平。如果是初次加电复位(又称“冷启动”)，则要求此高电平的持续时间不少于50μs。

当RESET信号一旦变为高电平时，8086 CPU就结束当前操作而进入复位状态，直到RESET信号变为低电平时为止。在复位状态，CPU内部的各寄存器被置为初态值，如表2-9所示。

表2-9　复位后内部寄存器的状态

寄存器	状态	寄存器	状态	寄存器	状态
FLAG(PSW)	0000H	IP	0000H	CS	0FFFFH
DS	0000H	SS	0000H	ES	0000H
指令队列	空	IF	0(禁止)		

由表可知，在复位状态，代码段寄存器(CS)为FFFFH，指令指针(IP)被清为0000H，所以，RESET恢复低电平后，8086 CPU便从FFFF0H单元处开始启动。FFFF0H称为系统的启动地址。

FFFF0H是ROM BIOS区中的一个单元，一般在FFFF0H处存放了一条无条件转移指令，用以使CPU转移到系统导引程序的入口处，这样，系统一旦被启动便自动进入系统

程序。

复位信号从高电平到低电平的跳变会触发 CPU 内部的一个复位逻辑电路，经过 7 个时钟周期后，CPU 就完成了启动操作。

复位时，由于标志寄存器(FR)被清 0，其中的中断允许标志(IF)也被清 0。这样，从 INTR 引脚输入的可屏蔽中断申请信号就不能被响应(即被屏蔽了)。因此在系统程序的适当位置要用开中断指令(STI)来设置中断允许标志。即，使 IF＝1，以开放中断。

复位时的操作时序如图 2.31 所示，由图可见，当 RESET 信号有效后，再经过一个状态，将执行：

(1) 把所有三态输出线(包括 AD_{15}～AD_0，A_{19}/S_6～A_{16}/S_3，$\overline{BHE}/\overline{S_7}$，$M/\overline{IO}$，$DT/\overline{R}$，$\overline{DEN}$，$\overline{RD}$，$\overline{WR}$ 和 $\overline{INTA}$)都置成高阻抗状态，直到 RESET 信号回低电平，结束复位操作为止。而且这些信号在进入高阻抗状态的前半个时钟周期先被置为不起作用状态。

(2) 把不具有三态功能的控制信号(ALE，HLDA，$\overline{RQ}/\overline{GT_0}$，$\overline{RQ}/\overline{GT_1}$，$QS_0$，$QS_1$)都置为无效状态。

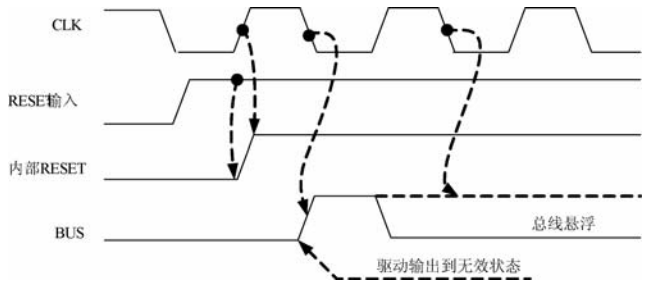


图 2.31　8086 复位时序

本 章 小 结

微处理器(CPU)是微型计算机的核心部件，Intel 8086 微处理器的基本结构和原理，体现了一般微处理器的原理，是微处理器的典型代表。学习微处理器结构的目的，一方面是为了理解微型计算机的工作原理，更重要的是为了应用。微处理器主要由运算器、控制器和寄存器阵列组成，各部分之间的信息交换是采用总线结构实现的，总线是各组件之间信息传输的公共通路，这种总线称为内部总线或片内总线。微型计算机是由微处理器、内存储器和 I/O 端口电路组成的，通过系统总线来实现相互之间的信息传送，总线是微处理器、内存储器和 I/O 端口之间相互交换信息的公共通路。系统总线由地址总线、数据总线和控制总线组成。

8086 微处理器设计了最小工作方式和最大工作方式以满足多种场合的应用，两种方式下，8086 微处理器部分引脚表现出不同的定义和功能，以实现相应方式下系统的管理与控制。最小工作方式适用于单处理器的小规模微机系统。在这种系统中，所有的总线控制信号，都是直接由 8086 微处理器产生的，系统中的总线控制逻辑电路被减到最少。最大工作方式应用在中、大规模的微机应用系统中，在最大方式下，系统中至少包含两个微处理器，其中一个为主处理器其他微处理器称之为协处理器，是协助主处理器工作的。

在微型计算机系统中，CPU 的操作都是在系统主时钟 CLK 的控制下按节拍有序进行

的。而所有的操作都是通过总线实现的，这里把通过外部总线对存储器或 I/O 端口进行一次读/写操作的过程称为总线周期。8086 系统中总线周期主要包括存储器读/写周期、I/O 端口读/写周期、中断响应周期、总线保持周期和总线请求/允许等针对总线的访问操作，而各种周期的时序分析是理解系统工作原理的基础和难点。

本章要重点掌握 8086 CPU 内部寄存器结构及功能、存储器地址空间和数据存储格式、存储器的分段和物理地址的形成、信息分段存储与段寄存器的关系、最小最大方式的总线结构与时序、CPU 的引脚定义和功能、存储器读写周期的时序、I/O 端口读写周期的时序等。

思考题与习题

2-1 8086CPU 由哪两部分组成?它们的主要功能各是什么?总线端口部件 BIU 由哪几部分组成？作用各是什么？

2-2 8086 CPU 为什么要采用地址/数据线分时复用？有何好处？

2-3 8086 CPU 中的标志寄存器分为哪两类标志？二者有何区别？

2-4 设段寄存器 CS＝2400H，指令指示器 IP＝6F30H，此时指令的物理地址 PA 是多少？指向这一物理地址的 CS 值和 IP 值是否是唯一的？

2-5 什么叫总线周期？8086/8088 系统中的总线周期由几个时钟周期组成？如果 CPU 的主时钟频率为 25MHz，一个时钟周期是多少？一个基本总线周期是多少时间？

2-6 在总线周期的 T_1，T_2，T_3，T_4 状态 CPU 分别执行什么动作？什么情况下需插入等待状态 T_W？何时插入？怎样插入？

2-7 RESET 信号到来后，CPU 的状态有何特征？系统从何处开始启动？

2-8 8086 在最大模式和最小模式下各有什么特点和不同？

2-9 8086 在最大工作方式、最小工作方式时各是如何配置的?各有何特点和不同?最大模式时，为什么一定要用总线控制器 8288？8288 的输入信号是什么？输出信号是什么？

2-10 当系统中有多个总线主模块时，在最大工作方式和最小工作方式下分别用什么方式来传送总线控制权的？

2-11 8086 的存储器空间各是多少？二者的存储器结构有何不同？寻址一个字节存储单元时有何不同？

2-12 什么是指令周期？什么是时钟周期？什么是总线周期？三者有何关系？简述 8086 最小工作方式下的总线读操作和写操作的过程及所涉及的主要控制信号。

2-13 设存储器内数据段中存放了两个字 2FE5H 和 3EA8H，已知 DS＝3500H，数据存放的偏移地址为 4B25H 和 3E5AH，画图说明这两个字在存储器中的存放情况。若要读取这两个字，需要对存储进行几次读操作?

2-14 时钟发生器 8284 向系统共输出几个时钟信号？

2-15 8086 系统中的总线收发器有什么作用？为什么系统中要加入总线收发器？

2-16 什么是时序？为什么要讨论时序？

2-17 8086/8088 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 周期？应插入多少个 T_w 取决于什么因素？

2-18 试简述 8086 系统最小工作方式时从储存器读数据时的时序过程。