微型计算机原理

教
案

数学与计算机科学学院

潘继强

目　录

第一章 微型计算机概述

一、计算机的定义

计算机是一种信息处理的工具，是一种在程序的控制下完成预定任务的电子仪器。包括硬件和软件两部分。

信息处理：包括信息的收集、加工、存储和传送。

二、微型计算机的组成

硬件：构成计算机系统的物理实体。计算机的硬件设备主要有：运算器、控制器、存储器、输入输出设备等部件。

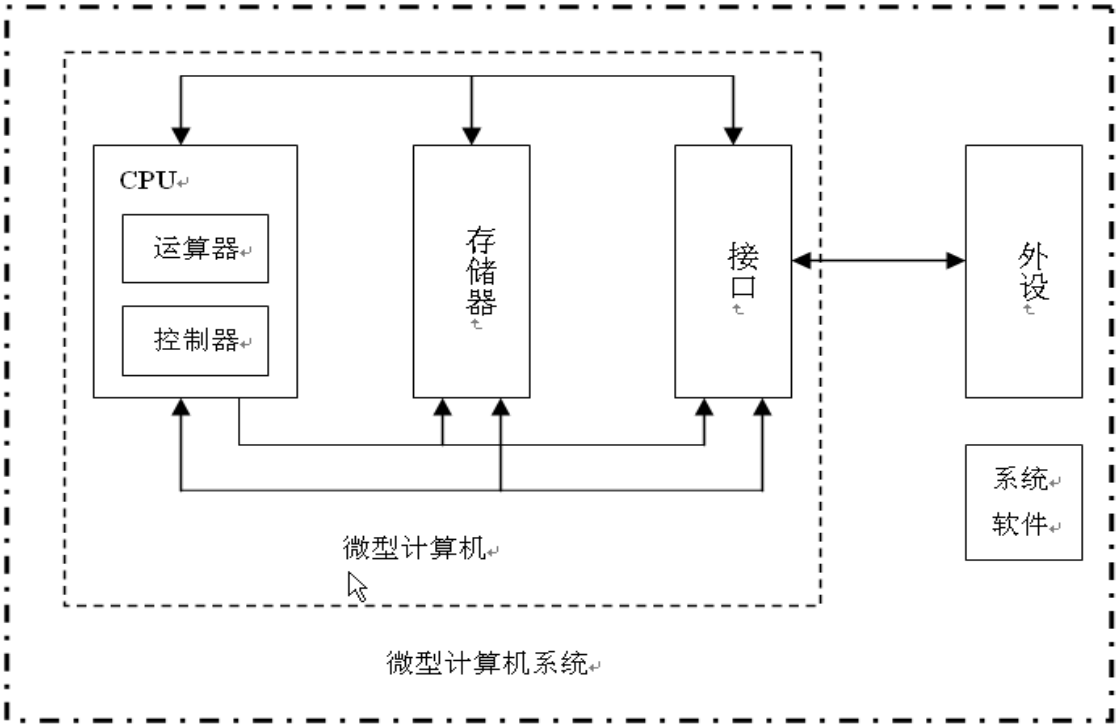


图 1.1 微型计算机组成图

1. 微处理器

运算器：完成算术及逻辑运算

算术运算：加、减、乘、除的定点、浮点运算。

逻辑运算：与（·或∧）、或（＋或∨）、非、异或

控制器：用于解释并协调整个系统完成指令的部件。控制器由指令寄存器、指令译码器、时序和控制电路，以及中断机构组成。指令寄存器存放当前正在执行的指令，而指令译码器对指令进行译码，此时，产生相应的控制信号送到时序

和控制电路，从而组合成 CPU 外部的其他部件所需要的时序和控制信号。这些信号送到微型计算机的其他部件，控制这些部件协调工作。

总之，微处理器是微型计算机的核心，它有两个指标：字长和主频。

2. 存储器

计算机中用于存储程序及数据的物理装置，分为内存和外存两大类。

内存包括 RAM 和 ROM，容量有限，用来存放经常使用的程序和数据，除必要的系统程序外，一般程序是存放在外存当中，只有在运行时才调入内存的某个区域。

外存比内存的容量大的多，但速度较慢，用来存放不常使用的程序和数据，通常作为某个外部设备。

3. 接口：用于计算机主机和外设进行匹配的电子部件。

4. 外设

输入设备：计算机从外部世界获取信息的入口。

输出设备：计算机向外部世界输出信息的出口，把运算结果或其它信息以数字、字符、图形等形式表示出来。

5. 总线

总线为 CPU 和其它部件之间提供数据、地址和控制信息的传输通道。有了总线结构以后，系统中各功能部件之间的相互关系就变为各个部件面向总线的单一关系。一个部件要符合总线标准，就可以连接到采用这种总线标准的系统中。

6. 系统软件

系统软件包括操作系统，一些语言处理程序和数据库。

其中操作系统是系统软件的核心，它管理计算机系统的全部硬件和软件资源，使计算机有条不紊的运行，为用户提供操作界面。

三、微型计算机的四个发展时期

1. 1971—1972 年 典型产品 Intel4004/8008，字长 4 位/8 位，主频 1MHz
2. 1973—1977 年 典型产品 Intel8080，字长 8 位，主频 2MHz。
3. 1978—1984 年 典型产品 Intel8086，字长 16 位，主频 5—10MHz。
4. 1985—至今 典型产品 Intel80386，字长 32 位，主频在 20MHz 左右。

第二章 微处理器 8086

一、16 位微处理器 8086

8086：字长 16 位，主频 5—10MHz，16 根数据总线和 20 根地址总线，可寻址 1MB 的内存储空间和 64KB 的 I/O 端口。

8088：准 16 位微处理器，内部寄存器、运算器以及内部数据总线都是按照 16 位来设计的，外部数据总线只有 8 条。

1.8086 的编程结构

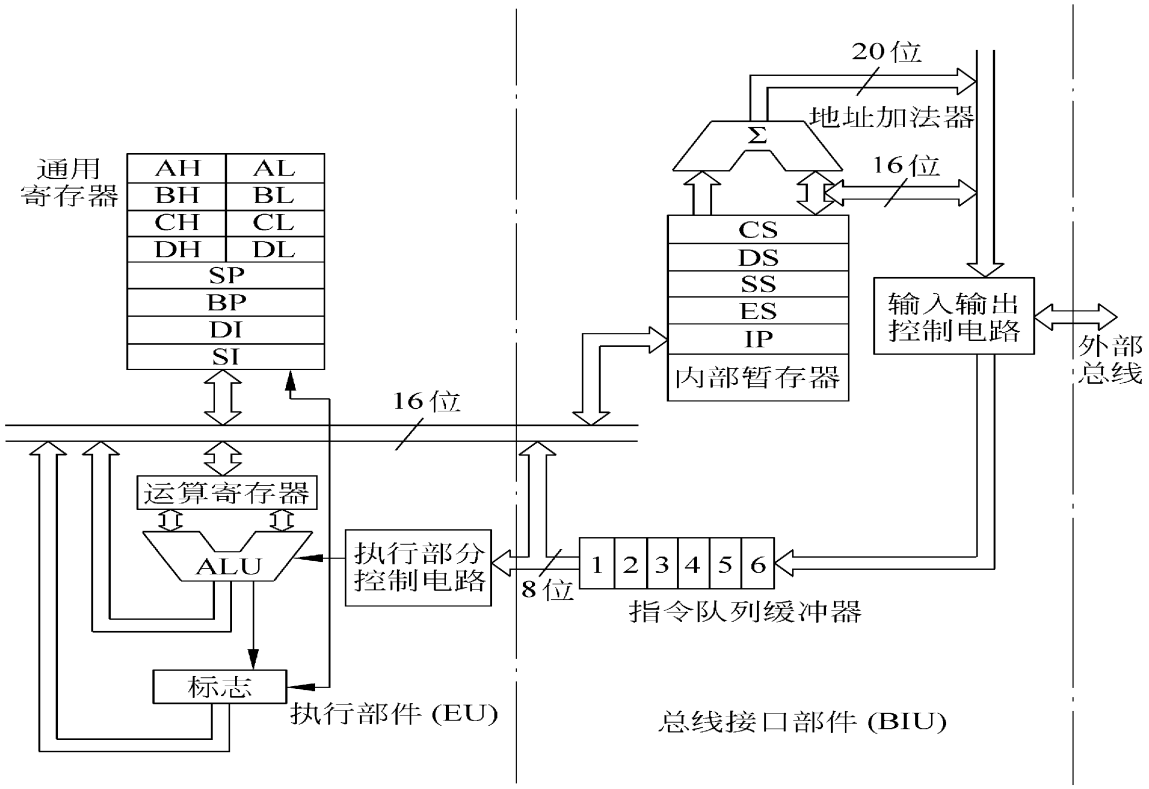


图 2.1 8086 的编程结构图

⑴总线接口部件 BIU

功能：负责与存储器、I/O 端口传送数据。

①BIU 要从内存取指令送到指令队列缓冲器。

②CPU 执行指令时，总线接口部件要配合执行部件从指定的内存单元或者外设端口取数据，将数据送给执行部件。

③把执行部件的操作结果传送到指定的内存单元或外设端口中。

8086 有 6 个字节的指令队列缓冲器，8088 有 4 个字节的指令队列缓冲器。

采用“先进先出”的原则，都会在执行指令的同时，从内存中取下面一条或

几条指令按顺序填入指令队列中。这样就保证了 8086/8088 执行完一条指令后可以立即执行下一条指令，也就是说执行指令和取指令的时间可以重叠，从而提高了 CPU 的利用率。

而早期的 8 位微处理器，取指令和执行指令是循环进行的。

⑵执行部件 EU

①执行单元 EU 的功能就是负责指令的执行。它包括下列部分。

②算术逻辑单元 ALU：ALU 完成 16 位或 8 位的二进制数的算术/逻辑运算，绝大部分指令的执行都由 ALU 完成。

③通用寄存器组和标志寄存器 FLAGS。

④控制电路 EU：接收从 BIU 中指令队列取来的指令，经过指令译码形成各种定时控制信号，向 EU 内各功能部件发送相应的控制命令，以完成每条指令所规定的操作。

2.BIU 和 EU 的动作管理

时钟周期——CPU 的基本时间计量单位，由计算机的主频决定。

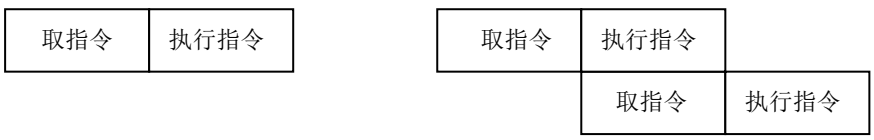
时钟周期=1/主频

例如：8086 的主频为 5MHz

时钟周期=1/(5×106Hz)=0.2×10-6s=200ns

总线周期——CPU 访问一次存储器或 I/O 端口所需要的时间。

由于 8086 的 BIU 中包含了一个 6 字节的指令队列缓冲器，CPU 在执行指令的同时，从存储器中读取下一条指令或下几条指令，取来的指令就放在指令队列中，这样，一般情况下，CPU 执行完一条指令就可以立即执行下一条指令，就够成了取指令和执行指令的二级流水线操作，从而提高了 CPU 的效率。如图而不像以往 8 位 CPU 重复进行先取指令和后执行指令的串行操作。负责取指令的总线接口单元和负责执行指令的执行单元需要按以下原则对流水线进行管理，才能提高 CPU 的执行效率。



（a）串行操作 CPU　　（b）8086 执行执行过程

图 2.2 CPU 执行指令过程

（1）当 8086 的指令队列中有两个空字节时，总线接口单元就会自动把指令取到指令队列中。

（2）当执行部件准备执行一条指令时，它会从总线接口部件的指令队列的队头取出指令代码，然后去执行指令。在执行指令过程中，如果需要访问存储器或 I/O 端口，那么执行部件就会请求总线接口单元，若总线接口单元空闲，则立即响应执行部件的总线请求；若总线接口单元正在取指令，则总线接口单元将先完成取指令的总线周期，再去响应执行部件发出的总线请求。

（3）当指令队列已满，而且执行部件又没有总线访问时，总线接口单元便进入空闲状态。

（4）在执行转移指令、调用指令和返回指令时，下面要执行的指令就不是在程序中的下一条指令了，而总线接口单元往指令队列装入指令时，总是按顺序进行的，这样，指令队列中已装入的指令就没有用。遇到这种情况，指令队列中的原有内容被自动清除，总线接口单元会接着往指令队列中装入另一个程序段中的指令。

3.8086 的总线周期

为了取得指令和数据，BIU 执行一个总线周期。在 8086/8088 中，一个基本的总线周期由 4 个时钟周期组成，将这 4 个时钟周期分别称为 4 个状态，即 T1 状态、T_2 状态、T_3 状态、T_4 状态。

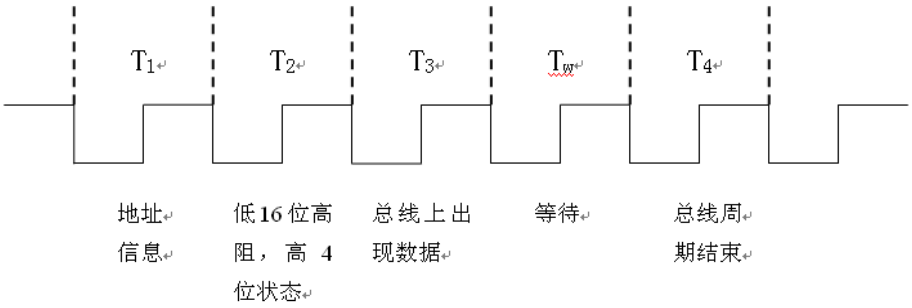


图 2.3 总线周期

二. 8086/8088 存储器和 I/O 编址

1. 8086/8088 存储器组织

8086/8088 的存储器是以字节为单位组织的。具有 20 条地址总线，所以可寻址的存储器地址空间容量为 1 MB（2^{20}B）。每个字节对应一个唯一的地址，地址

范围为 00000H～FFFFFH。

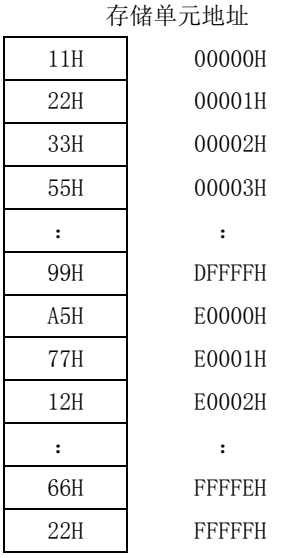


图 2.4 数据在存储器中的存放

一个存储单元中存放的信息称为该存储单元的内容。如 00000H 单元的内容为 11H，记为：(0000H)=11H。

存储器中两个连续的字节，定义为一个字。一个字中的每个字节，都有一个字节地址，字的低字节存放在低地址中，高字节存放在高地址中。如从地址 00002H 开始的两个连续单元中存放一个字，则该数据位 5533H，记为：(00002H)=5533H。

由于 8086/8088CPU 内部数据总线和寄存器均为 16 位，内部 ALU 只能进行 16 位运算，因此 8086/8088CPU 对地址只能进行 16 位运算，其寻址范围为 2^{16}=64KB，但处理器地址总线的位数是 20 位。所以为了获得 20 位地址，需要引入“分段”概念，将寻址范围扩大到 1MB。

将段寄存器内容左移 4 位（低 4 位补 4 个 0）与偏移量相加，即形成 20 位的物理地址。

物理地址=段地址×10H+段内偏移量

例如段寄存器（CS）=1120H, 指令指针（IP）=2008H，则逻辑地址表示为 CS:IP=1120H:2008H，物理地址则为 1120H×10H+2008H=13208H。

2. I/O 编址

8086/8088 系统和外部设备之间都是通过 I/O 接口来联系的。每个 I/O 接口都有一个或几个端口，微机系统要为每个端口分配一个地址，此地址称为端口号。一个端口通常对应一个或一组寄存器。

8086/8088 的 I/O 端口地址总线为 16 位，即提供 2^{16}=65536（64K）个 8 位的 I/O 端口，两个编号相邻的 8 位端口可以组合成一个 16 位端口。

三、8086 的引脚信号

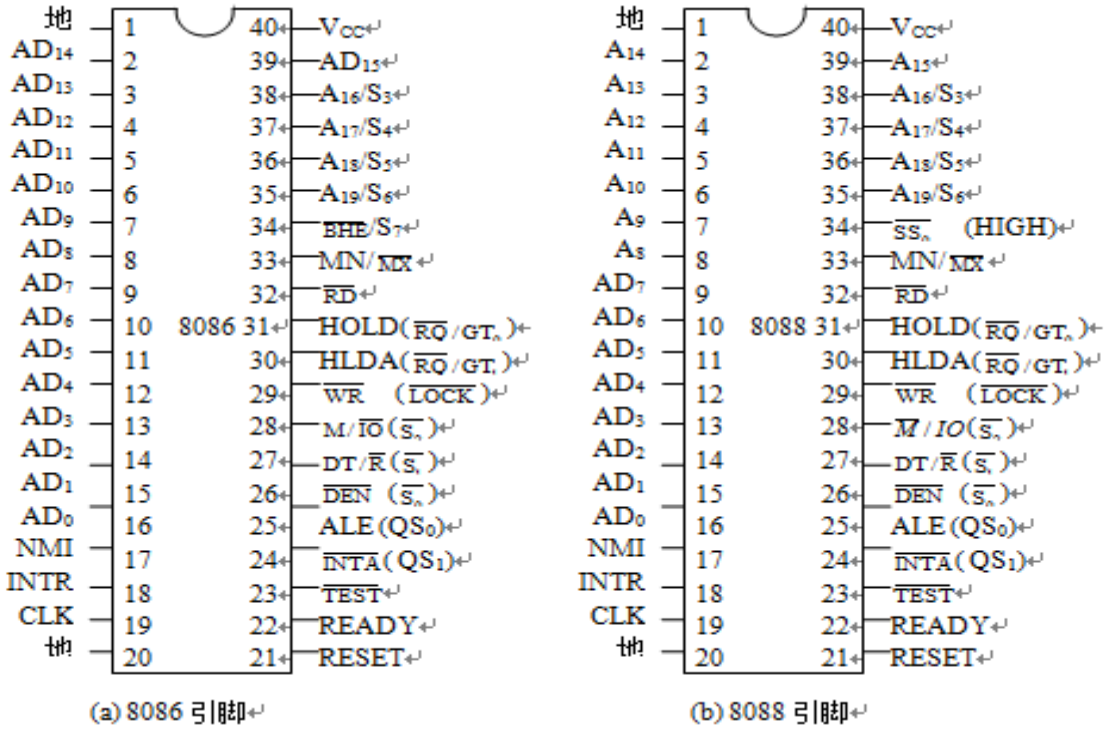


图 2.5 8086/8088 引脚图

1. 地址/数据复用引脚

AD_0～AD_{15}：地址/数据分时复用输入输出信号线。分时复用，T_1 时刻出现地址，T_3 时刻出现数据。

2. 地址/状态复用总线

A_{19}/ S_6～A_{16}/ S_3:每个总线周期开始 T_1 状态时做地址线用，其余状态输出状态信息。

当访问存储器时，T1 状态输出 A_{19}～A_{16}，与 AD_{15}～AD_0 一起构成访问存储器的 20 位物理地址。

当 CPU 访问 I/O 端口是，不使用这 4 个引脚，A_{19}～A_{16} 保持为 0。

在 T_2～T_4 状态，S_6 为 0 表示 8086CPU 当前与总线相连；S_5 表示中断允许标志位 IF 的当前设置；S_4S_3 用来指示当前正在使用哪个段寄存器。

3.控制总线

NMI：非屏蔽中断请求信号，输入，上升沿触发。

INTR：可屏蔽中断请求信号，输入，高电平有效。若此信号有效，表明有外设提出了中断请求，这时若 IF=1，则当前指令执行完成后立即响应中断。

RESET：CPU 的复位信号，输入，高电平有效。复位信号使处理器马上结束现行操作，对处理器内部寄存器进行初始化。8086 要求复位脉冲宽度不得小于 4 个时钟周期。

READY: 数据“准备好”信号，输入。高电平有效。当 CPU 对存储器或 I/O 短口进行操作时，在 T_3周期开始采样 READY 信号。若它为高电平，则表示存储器或 I/O 设备已准备好；若为低电平，则表示被访问的存储器或 I/O 设备还未准备好数据，此时在 T_3周期以后插入 Tw 周期，然后再在 Tw 周期中再采样 READY 信号，直到 READY 变为高电平时 Tw 周期才可以结束，进入 T4 周期，完成数据传送。

$\overline{\text{TEST}}$：用于测试的输入信号。当 WAIT 指令执行时，该信号低电平有效，CPU 就进入等待状态

$\text{M/}\overline{\text{IO}}$：用来区分当前操作是访问存储器还是访问 I/O 端口。

$\overline{\text{WR}}$：该引脚输出为低电平时，表示 CPU 正处于写存储器或写 I/O 端口。

$\overline{\text{RD}}$：低电平有效，表示 CPU 正在进行存储器或 I/O 端口读操作。

$\text{DT/}\overline{\text{R}}$：用于确定数据传送的方向，高电平为发送方向，即 CPU 写数据到存储器或 I/O 端口；低电平为接收方向，即 CPU 到存储器或 I/O 端口读数据。

$\overline{\text{DEN}}$：该信号有效时，表示数据总线上有有效数据。

ALE：高电平有效。当它有效时，表明 CPU 送出有效的地址信号，因此，它常作为锁存控制信号将 A_0～A_{19}锁存于地址锁存器中。

$\overline{\text{INTA}}$：CPU 输出的中断响应信号，是 CPU 对外部输入的 INTR 中断请求信号的响应。

HOLD：输入信号高电平有效，用于向 CPU 提出总线保持请求。当某一部件要占用系统总线时，可通过这条输入线向 CPU 提出请求。

HLDA：CPU 对 HOLD 请求的响应信号，高电平有效的输出信号。

$MN/\overline{MX}$：当 $MN/\overline{MX}$ =1 时，CPU 工作在最小模式下，构成的微型机中只包括一个微处理器；当 $MN/\overline{MX}$ =0 时，CPU 工作在最大模式下，构成的微型机系统中除了有 8086CPU 之外，还可以有另外的微处理器。

四、8086 的的操作和时序

1. 总线操作

总线读操作——指 CPU 从存储器或 I/O 端口读取数据。

总线写操作——指 CPU 将数据写入存储器或 I/O 端口。

⑴最小模式下的总线读操作（以读存储器为例）

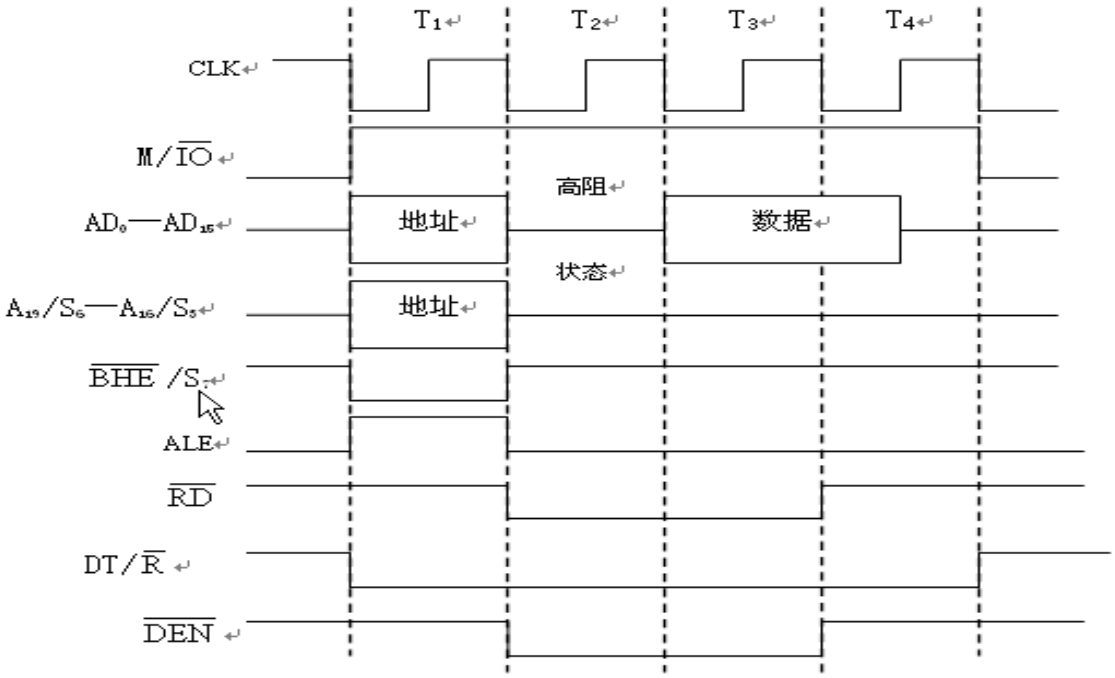


图 2.6 总线读操作时序图

T_1 状态：

首先要用 $M/\overline{IO}$ 信号指出 CPU 是要从存储器还是 I/O 端口读，所以 $M/\overline{IO}$ 信号在 T_1 状态成为有效。$M/\overline{IO}$ 信号的有效电平一直保持到整个总线周期的结束，即 T_4 状态。

8086 的 20 位地址信号通过分时复用总线 A_{19}/S_6 ～A_{16}/S_3 和 AD_{15}～AD_0 输出，

送到存储器或 I/O 端口。

CPU 便在 T1 状态从 ALE 引脚上输出一个正脉冲作为地址锁存信号。在 ALE 的下降沿到来之前，$M/\overline{IO}$ 信号、地址信号均以有效。

$\overline{BHE}$ 信号也通过 $\overline{BHE}/S_7$ 引脚送出，它用来表示高 8 位数据总线上的信息可以使用。

当系统中连接有数据总线收发器时，要用到 $DT/\overline{R}$ 和 $\overline{DEN}$ 作为控制信号。前者作为对数据传输方向的控制，后者允许数据传送。

T_2 状态：

地址信号消失，AD_{15}～AD_0 进入高阻状态，为读入数据做准备；而 A_{19}/S_6～A_{16}/S_3 和 $\overline{BHE}$ /S7 输出状态信息 S_7～S_3。

此时，$\overline{DEN}$ 信号变为低电平，从而在系统中接有总线收发器时，获得数据允许信号。

CPU 在 $\overline{RD}$ 引脚上输出读有效信号，送到系统中所有存储器和 I/O 接口芯片，但是，只有被地址信号选中的存储单元或 I/O 端口，才会被信号从中读出数据，从而将数据送到系统数据总线上。

T_3 状态：

在 T_3 状态前沿，CPU 对引脚 READY 进行采样，如 READY 信号为高电平，则 CPU 在 T_3 状态后沿通过 AD_{15}～AD_0 获取数据；如果为低电平，将插入等待状态 Tw，直到 READY 信号变为高电平。

Tw 状态：

当系统中所用的存储器或外设的工作速度较慢，从而不能用最基本的总线周期完成读操作时，系统中就要用一个电路来产生 READY 信号。低电平的 READY 信号必须在 T_3 状态启动之前向 CPU 发出，则 CPU 将会在 T_3 状态和 T_4 状态之间插入若干个等待状态 Tw，直到 READY 信号变高。在最后一个等待状态 Tw 的后沿处，CPU 通过 AD_{15}～AD_0 获取数据。

T_4 状态：

CPU 使 $\overline{RD}$ 信号变为高电平，于是，存储器模块上的总线驱动器又处于高阻

状态，从而让出总线。

⑵最小模式下的总线写操作（以写存储器为例）

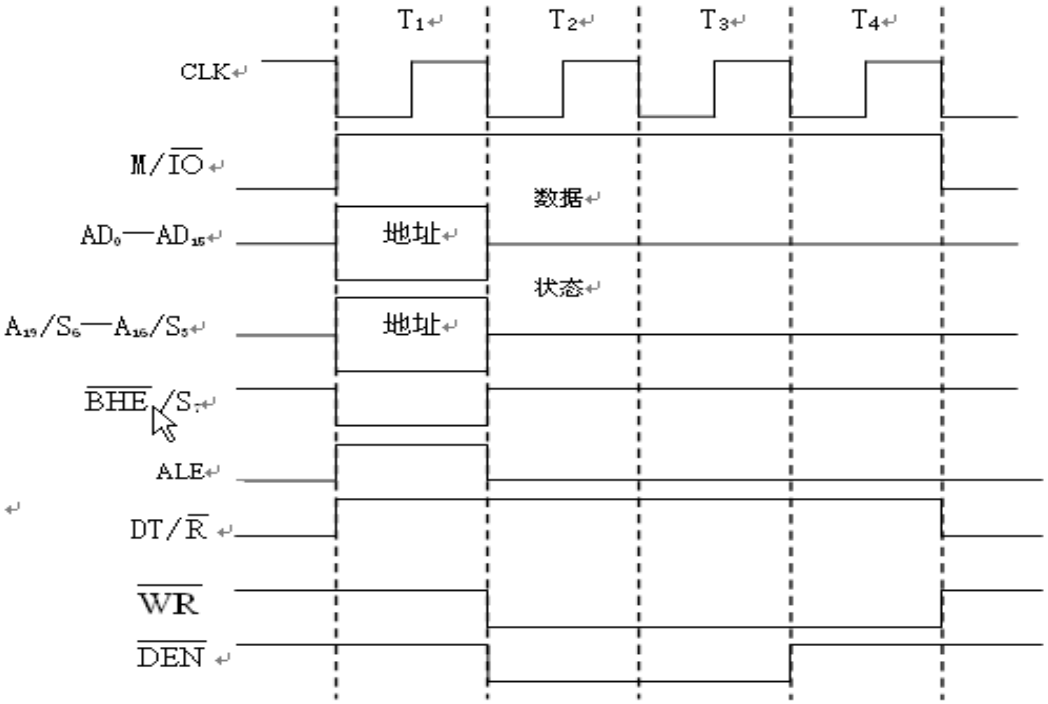


图 2.7 总线写操作时序图

对存储器或 I/O 端口操作时序基本相同。总线读操作中，选通信号是 $\overline{RD}$，而总线写操作中是 $\overline{WR}$。

在 T_2 状态中，AD_{15}～AD_0 上地址信号消失后，AD_{15}～AD_0 的状态不同。总线读操作中，此时 AD_{15}～AD_0 进入高阻状态，并在随后的状态中保持为输入方向；而在总线写操作中，此时 CPU 立即通过 AD_{15}～AD_0 输出数据，并一直保持到 T_4 状态中。

2. 中断系统和中断操作

8086/8088 可以处理 256 种不同的中断，每个中断对应一个中断类型码，中断类型码为 0——255。

⑴中断的分类

硬件中断是通过外部是硬件产生的

非屏蔽中断 NMI；可屏蔽中断 INTR

软件中断是 CPU 根据软件的某条指令或软件对标志寄存器中某个标志的设置而产生的。

比如：除数为 0 引起的中断；中断指令引起的中断

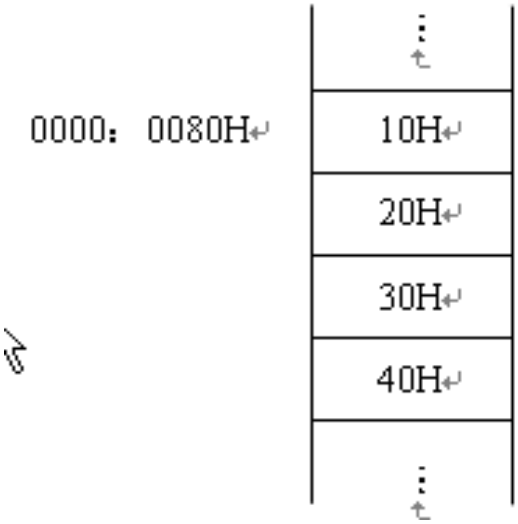
⑵中断向量和中断向量表

中断向量：中断处理子程序的入口地址。

每个中断类型对应一个中断向量。一个中断向量占用 4 个存储单元，其中前 2 个单元存放中断处理子程序入口地址的偏移地址；后 2 个单元存放中断处理子程序入口地址的段地址。

中断向量不是任意存放的。

中断向量表：内存 0 段的 0000—03FFH(1023D)区域。



20H 号中断对应的中断向量为 4030:2010H。

⑶硬件中断的响应和时序

可屏蔽中断的响应过程：

当 CPU 在 INTR 引脚上接收一个高电平的中断请求信号，并且当前中断允许标志为 1 时，CPU 就会在当前指令执行完以后，开始响应外部的中断请求，即 CPU 往发两个连续的负脉冲，外设接口接到第二个负脉冲以后，立即往数据总线上给 CPU 送来中断类型码。之后依次做以下工作：

①从数据总线上读取中断类型码，将其存入内部暂存器。

②将标志寄存器(PSW)的值压入堆栈；

③清楚中断允许标志 IF 和跟踪标志 TF；

④将断点保存到堆栈中；

⑤根据前面得到的中断类型码，到内存 0 段的中断向量表找到中断向量，再根据中断向量转入相应的中断处理子程序。

响应可屏蔽中断时的总线时序：

①执行 2 个中断响应总线周期。CPU 接收中断类型码，将它左移 2 位称为中断向量的起始地址，存入内部暂存器。

②执行一个总线写周期，将 PSW 的值压入堆栈。

③清除 IF 和 TF。

④执行一个总线写周期，将断点处 CS 的内容压入堆栈。

⑤执行一个总线写周期，将断点处 IP 的内容压入堆栈。

⑥执行一个总线读周期，CPU 将从中断向量所在的前 2 个字节中读取中断处理子程序入口地址的偏移地址送入 IP。

⑦执行一个总线读周期，将中断处理子程序入口地址的段地址送入 CS。

⑷中断处理子程序

①保护现场

②开中断

③中断服务——中断处理的具体内容

④关中断

⑤恢复现场

⑥中断返回

第三章 8086 的指令系统

一、计算机语言的发展

机器语言是由计算机能够识别并执行的 0、1 代码构成，是唯一能被机器识别并执行的语言。显然这种语言不便于人们阅读、理解、交流，编写程序复杂困难，并且很容易出错。既然这种语言在使用过程中有诸多不便之处，所以在计算机语言的发展过程中就出现了汇编语言和高级语言。

汇编语言是面向机器硬件的语言，是一种助记符语言。它以指令系统为核心，采用能够帮助理解和记忆的英文单词或其缩写符号来代替机器指令中的操作码，并对所需的数据、寄存器或有关数据的地址用相应的符号表示，并把每一条机器指令都用相应的符号化指令代替，与机器语言一一对应。

高级语言与计算机的硬件结构及指令系统无关，它有更强的表达能力，可方便地表示数据的运算和程序的控制结构，能更好的描述各种算法，而且容易学习掌握。但高级语言编译生成的程序代码一般比用汇编程序语言设计的程序代码要长，执行的速度也慢。

二、8086 寻址方式

所谓寻址方式是指寻找操作数或者操作数地址的方式。

一条汇编语言指令由操作码字段和操作数字段两部分组成。操作码指示计算机所要完成的操作，而操作数则是操作码的操作对象。换句话来讲，一条汇编语言指令需要解决两个问题，其一是要指出完成何种操作；其二是要指出大部分指令涉及的操作数和操作结果存放在何处，不但要指出操作数的值为多少或者存放在什么地方，而且还要指出操作结果送到哪里去。为了使指令形式比较简单，约定将操作结果送回到原来存放操作数的地方。这样，第二个问题，就归结为指出操作数的来源，这就是操作数的寻址方式。

通常，操作数有四个来源：立即数、寄存器、存储器、I/O 端口。

操作数字段可以有一个、两个或者没有，分别称为一地址指令、二地址指令和零地址指令。单操作数指令就是一地址指令，它只需要指定一个操作数。在 8086 指令系统中大部分属于二地址指令，此时这两个操作数分别称为目的操作数和源操作数，其格式如图 3.1 所示。

操作码	目的操作数,源操作数

图 3.1 二地址指令格式

1．立即寻址

操作数直接放在指令中，它是紧跟在操作码后面一个 8 位或 16 位的常数。

例 1：MOV AX，2345H

指令执行后，(AX)=2345H

这种寻址方式主要用来给寄存器或存储单元赋初值，并且只能出现在源操作数字段。

2．寄存器寻址

操作数在寄存器当中，在指令中指定寄存器号。

不同位数的操作数使用不同的寄存器。对于 16 位操作数，寄存器可以是 AX、BX、CX、DX、SI、DI、SP、BP，对于 8 位操作数，寄存器可以是 AH、AL、BH、BL、CH、CL、DH、DL。

例 2：MOV AX，BX

指令执行前(AX)=3088H，(BX)=5678H；则指令执行后(AX)=5678H，BX 保持不变。

3．直接寻址

指令操作码之后直接给出操作数的 16 位偏移地址，适用于处理单个变量。

例 3：MOV AX，[2000H]

设指令执行前(DS)=3000H，(32000H)=56H，(32001)=34H，则

物理地址=(DS)×16D+2000H =30000H+2000H=32000H

指令执行后：(AX)=3456H，寻址过程如图 3.2 所示：

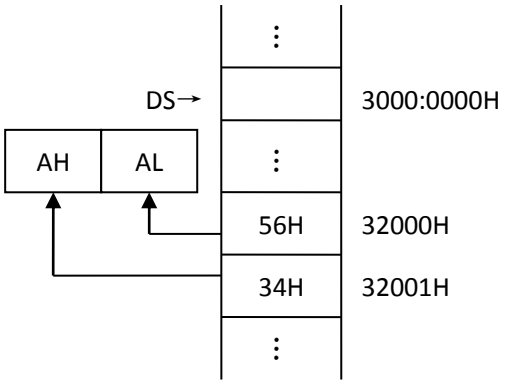


图 3.2 直接寻址示意图

在汇编语言指令中，可以用符号地址代替数值地址，如：

MOV　AX，VALUE

如果数据在附加段当中，应在指令中指明段跨越前缀，如：

MOV　AX，ES：VALUE

4．寄存器间接寻址

操作数的偏移地址在指令中指明的基址寄存器或变址寄存器中，而操作数则在存储器中。在 8086 系统中可以使用的基址或变址寄存器有 BX、BP、SI 和 DI，当使用 BP 时，默认的段寄存器为 SS，否则默认的段寄存器为 DS。

例 4：MOV　AX，[BX]

设指令执行前：(DS)=2000H，(BX)=1000H，(21000H)=A0H，(21001H)=50H，则

物理地址=(DS)×16D+(BX)=20000H+1000H=21000H

指令执行后：(AX)=50A0H，寻址过程如图 3.3 所示：

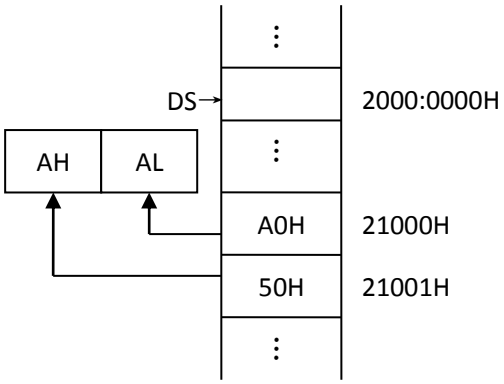


图 3.3 寄存器间接寻址示意图

5．寄存器相对寻址

操作数的偏移地址为基址寄存器或变址寄存器的内容和指令中指定的 8 位或 16 位的位移量之和。同样这里能够使用的基址或变址寄存器有 BX、BP、SI 和 DI，当使用 BP 时，默认的段寄存器为 SS，否则默认的段寄存器为 DS。

例 5：MOV　AX，COUNT[SI]，也可写成 MOV　AX，[COUNT＋SI]

设指令执行前：(DS)=3000H，(SI)=2000H，COUNT=3000H，(35000H)=45H，(35001H)=23H，则

物理地址=(DS)×16D+(SI)+ COUNT=30000H+2000H+3000H=35000H

指令执行后：(AX)=2345H，寻址过程如图 3.4 所示：

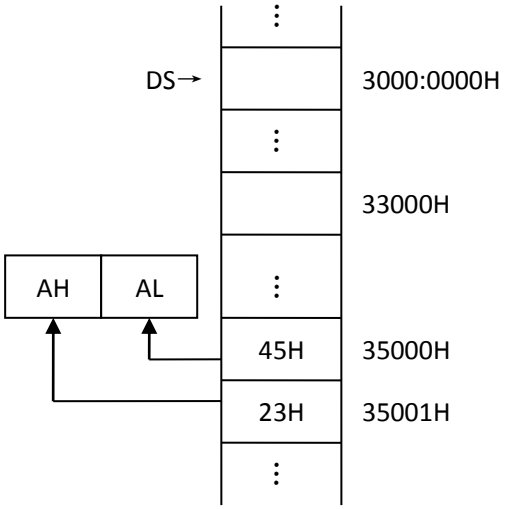


图 3.4 寄存器相对寻址示意图

6．基址变址寻址

操作数的偏移地址是指令中指定的一个基址寄存器和一个变址寄存器的内容之和。在 8086 系统中，能够作为基址寄存器使用的只能是 BX 和 BP，能够作为变址寄存器使用的只能是 SI 和 DI。

例 6：MOV　AX，[BX][DI]，也可写成 MOV　AX，[BX＋DI]

设指令执行前：(DS)=2000H，(BX)=1000H，(DI)=0010H，(21010H)=34H，(21011H)=12H，则

物理地址=(DS)×16D+(BX)+ (DI)=20000H+1000H+0010H=21010H

指令执行后：(AX)=1234H，寻址过程如图 3.5 所示：

7．相对基址变址寻址

指令中指定了一个基址寄存器和一个变址寄存器，同时还给出了一个 8 位或 16 位的位移量，将三者相加就得到操作数在内存中的偏移地址。

例 7：MOV　AX，VAL[BX][SI]，也可写成 MOV　AX，[VAL＋BX＋SI]

设指令执行前：(DS)=2000H，(BX)=1000H，(SI)=0200H，VAL=0050H，(21250H)=34H，(21251H)=12H，则

物理地址=(DS)×16D+(BX)+ (SI)＋VAL=20000H+1000H+0200H+0050H=21250H

指令执行后：(AX)=1234H，寻址过程如图 3.6 所示：

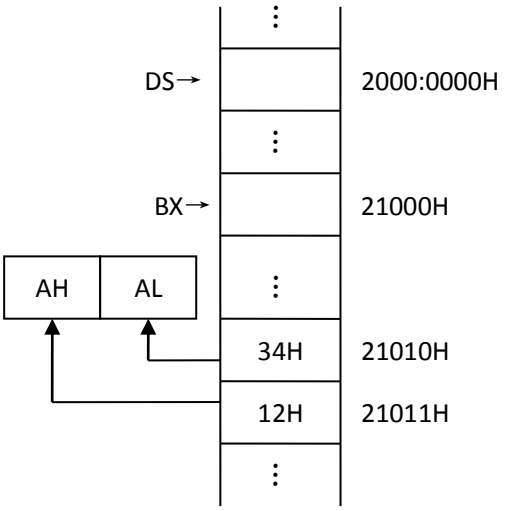


图 3.5 基址变址寻址示意图

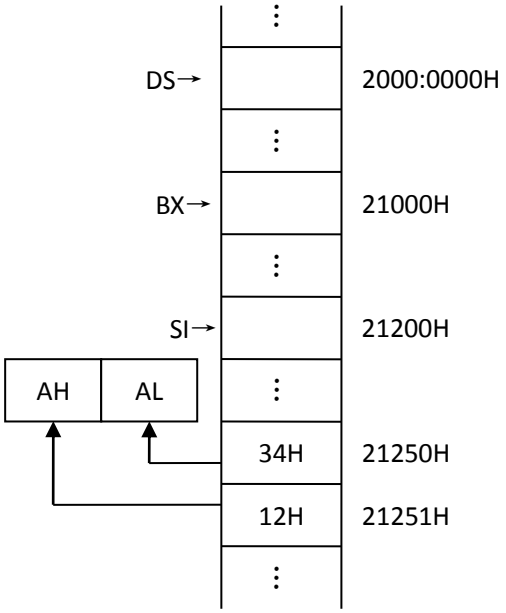


图 3.6 相对基址变址寻址示意图

三、程序转移寻址

程序正常顺序执行时，每取出一条指令执行(IP)＋n→(IP)，其中 n 为取出指令的字节数，然后形成下一条指令的物理地址。

物理地址=(CS)×16D+(IP)

但程序发生转移时，需要计算出偏移地址并修改 IP，有时候还需要修改 CS 的值，据此程序转移寻址可以分为段内转移和段间转移。段内转移是指在同一个代码段内，仅改变 IP 的值而不改变 CS 的值发生的转移。但是如果程序从一个代码段转移到另一个代码段，则不仅改变 IP 的值，同时要改变 CS 的值，这种情形

称为段间转移。

1．段内直接寻址

转向的偏移地址是当前 IP 寄存器的内容和指令中指定的 8 位或 16 位的位移量之和，然后再送入 IP。这种寻址方式适用于条件转移指令和无条件转移指令。当位移量为 8 位时，偏移地址形成一个转移范围在－128～+127 字节之间的短距离转移，则称为短跳转。当位移量为 16 位时，偏移地址形成一个转移范围在 64KB 内的近距离转移，则称为近跳转。

例 8：JZ NEXT

JMP SHORT LOP

JMP NLAB

2．段内间接寻址

转向的偏移地址在某个通用寄存器中或者在存储器的一个字单元中，则将寄存器的内容或存储单元的内容送入 IP，这个寄存器或存储单元的内容可以用操作数寻址方式中除立即寻址以外的任何一种寻址方式取得。这种寻址方式只能用于无条件转移指令。

例 9：指令 JMP BX

设指令执行前：(BX)=1256H，则

指令执行后：(IP)=1256H

例 10：指令 JMP [BX][SI]

设指令执行前：(DS)=2000H，(BX)=1200H，（SI）=0050H，（21250H）=50H，（21251H）=24H，则

物理地址=(DS)×16+(BX)+(SI)=20000H+1200H+0050H=21250H

指令执行后：(IP)=2450H

3．段间直接寻址

在指令中直接提供了转向段地址和偏移地址，所以只要用指令中指定的偏移地址取代 IP 寄存器的内容，用指令中指定的段地址取代 CS 寄存器的内容就可以从一个段转移到另一个段。这种寻址方式只能用于无条件转移指令。

例 11：JMP FAR PTR NEXT

4．段间间接寻址

用存储器中的两个相继字单元的内容来取代 IP 和 CS 寄存器中的原始内容，以达到段间转移的目的，其中第一个字单元的内容作为偏移地址送入 IP，第二个字单元的内容作为段地址送入 CS。这种寻址方式只能用于无条件转移指令。

例 12：JMP DWORD PTR［INTER+BX］

四、数据传送类指令

1.通用数据传送指令

①传送指令 MOV

格式：MOV　DST，SRC

功能：(DST)←(SRC)

使用 MOV 指令需要注意的是：

①DST 和 SRC 必须同时为 8 位或 16 位，否则会产生操作数类型不匹配的错误。

②SRC 可以是寄存器、存储单元、段寄存器和立即数，DST 只能是寄存器、存储单元、段寄存器(CS 除外)。

③DST 和 SRC 不能同为两个存储单元，也不能同为两个段寄存器。不能将立即数直接的送段寄存器。

④MOV 指令不影响标志位

⑵堆栈操作指令

格式：PUSH　SRC

　　POP　 DST

功能：8086 堆栈操作总是按字进行的。PUSH 每执行一次，堆栈栈顶指针 SP 减 2，推入堆栈的数据放在栈顶，低位字节放在较低地址单元，高位字节放在较高地址单元。POP 指令正好相反，每弹出一个字，堆栈栈顶指针 SP 加 2。

⑶交换指令 XCHG

格式：XCHG　OPR1，OPR2　 ；OPR 表示操作数

功能：操作数 OPR1 和操作数 OPR2 相交换

注意：XCHG 指令的两个操作数不能同为存储单元，两个操作数任意一个也不允许使用段寄存器，且不影响标志位。

2．累加器专用传送指令

⑴输入输出指令

格式：IN　AL，PORT 或 IN　AX，PORT

　　OUT　PORT，AL 或 OUT　PORT，AX　　；PORT 表示端口号

功能：执行输入指令 IN 时，CPU 可以一个 8 位端口读入一个字节到 AL 中，也可以从两个连续的 8 位端口读一个字到 AX 中；执行输出指令 OUT 时，CPU 可以将 AL 中的一个字节写到一个 8 位端口中，也可以将 AX 中的一个字写到两个连续的 8 位端口中。

输入输出指令可以分为两大类：一类是直接的输入输出指令，另一类是间接的输入输出指令。使用直接输入输出指令时，寻址范围为 0～255(十六进制 0～0FFH)。使用间接输入输出指令时，寻址范围为 0～65535(十六进制 0～0FFFFH)。

⑵换码指令 XLAT

格式：XLAT

功能：(AL)←((AL)+(BX))

使用 XLAT 之前，应先建立一个字节表格，表格的首地址提前存入 BX 寄存器，AL 中为字节表格中某一项与表格首单元之间的位移量，指令执行时，会将 BX 和 AL 中的值相加，把得到的值作为地址，然后将此地址所对应的存储单元的值取到 AL 中。

3．地址传送指令

⑴LEA 指令

格式：LEA　DST，SRC

功能：(DST)←SRC

LEA 装入偏移地址是根据 SRC 寻址方式计算偏移地址，而不是再用偏移地址来取操作数。DST 使用 16 位的寄存器，SRC 只能使用与存储器相关的寻址方式。

例 13：LEA　AX，[2400H]　；将内存单元的偏移地址 2400H 送 AX

　　LEA　BX，[BP+SI]　；指令执行后，BX 中的内容为 BP+SI 的值

4．类型转换指令

⑴CBW 字节转换为字指令

格式：CBW

功能：将 AL 的内容符号扩展到 AH，形成 AX 中的字。如果(AL)的最高位为 0，

则(AH)=0；如果(AL)的最高位为 1，则(AH)=0FFH。

⑵CWD 字转换为双字指令

格式：CWD

功能：将 AX 的内容符号扩展到 DX，形成 DX：AX 中的双字。如果(AX)的最高位为 0，则(DX)=0；如果(AX)的最高位为 1，则(DX=0FFFFH。

五、算术运算类指令

8086 的算术运算指令众多，可归结为双操作数指令和单操作数指令两类。双操作数指令的两个操作数中除源操作数为立即寻址以外，必须有一个操作数在寄存器中。单操作数指令不允许使用立即寻址。

1．加法指令

⑴加法指令 ADD

格式：ADD　DST，SRC

功能：(DST)←(DST)+(SRC)

⑵带进位加法指令 ADC

格式：ADC　DST，SRC

功能：(DST)←(DST)+(SRC)+CF

⑶加 1 指令 INC

格式：INC OPR

功能：(OPR)←(OPR)+1

2．减法指令

⑴减法指令 SUB

格式：SUB　DST，SRC

功能：(DST)←(DST)－(SRC)

⑵带借位减法指令 SBB

格式：SBB　DST，SRC

功能：(DST)←(DST)－(SRC)－CF

⑶减 1 指令 DEC

格式：DEC OPR

功能：(OPR)←(OPR)－1

和加 1 指令类似，一般用在循环程序中修改指针和循环次数。

⑷求补指令 NEG

格式：NEG OPR

功能：(OPR)←－(OPR)

⑸比较指令 CMP

格式：CMP OPR1，OPR2

功能：(OPR1)－(OPR2)

该指令和 SUB 指令一样执行减法运算，但并不保存结果，只是根据结果设置条件码标志。CMP 指令后往往跟着一条条件转移指令，根据比较结果产生不同的程序分支。

例 14：已知数组首地址为 X，段地址已放在 DS 中，末字为 0FFFFH，试统计其中 0 的个数并存放在末字单元，流程图如图 3.7 所示。

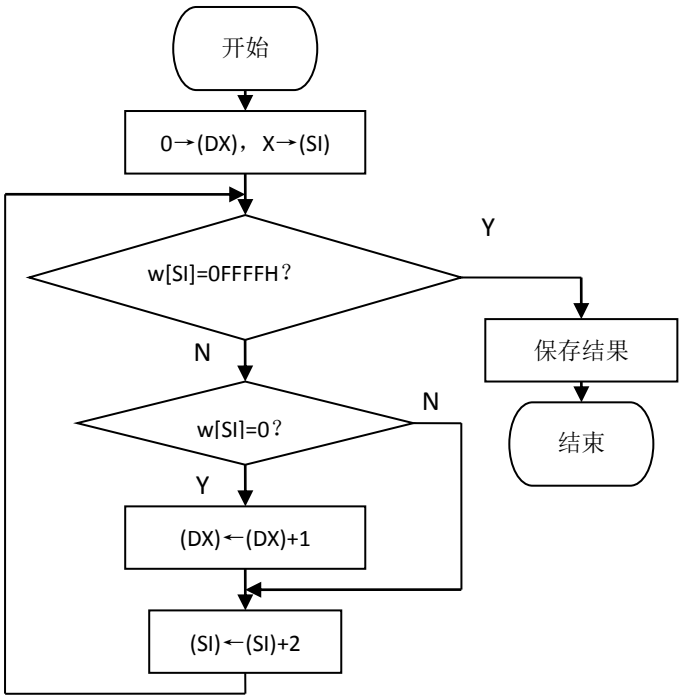


图 3.7 流程图

程序如下：

```
        MOV  DX，0
        LEA  SI，X
LOP：CMP  WORD  PTR [SI]，0FFFFH
```

```
        JZ ENDO
        CMP  WORD  PTR [SI], 0
        JNZ NEXT
        INC DX
NEXT: ADD  SI, 2
        JMP LOP
ENDO: MOV [SI], DX
```

3．乘法指令

⑴无符号数乘法指令 MUL

格式：MUL SRC

功能：(AX)←(AL)*(SRC)　　；字节操作数相乘

(DX，AX)←(AX)*(SRC)　；字操作数相乘

⑵带符号数乘法指令 IMUL

格式：IMUL SRC

功能：(AX)←(AL)*(SRC)　　；字节操作数相乘

(DX，AX)←(AX)*(SRC)　；字操作数相乘

在乘法指令里，目的操作数必须是累加器，字运算为 AX，字节运算为 AL。两个 8 位数相乘得到的是 16 位的积存放在 AX 中，两个 16 位数相乘得到的是 32 位的积存放在 DX、AX 中，其中 DX 存放高位字，AX 存放低位字。

4．除法指令

⑴无符号数除法指令 DIV

格式：DIV SRC

功能：当 SRC 为字节操作数，16 位的被除数在 AX 中时，表示为：

(AL)←(AX)/(SRC)的商

(AH)←(AX)/(SRC)的余数

当 SRC 为字操作数，32 位的被除数在 DX、AX 中时，表示为：

(AX)←(DX，AX)/(SRC)的商

(DX)←(DX，AX)/(SRC)的余数

⑵带带符号数除法指令 IDIV

格式：IDIV SRC

功能：与 DIV 指令相同，但操作数必须是带符号数，商和余数也是带符号数，且余数的符号和被除数的符号相同。

六、逻辑运算及移位指令

1．逻辑运算指令

8086 的逻辑运算指令包括 AND(与)、OR(或)、NOT(非)、XOR(异或)和 TEST(测试)指令。

以上五条指令中，NOT 属于单操作数指令，不允许使用立即寻址，其余 4 条指令都属于双操作数指令，除源操作数是立即数外，至少有一个操作数必须放在寄存器中。NOT 指令不影响条件码标志，其余 4 条指令是 CF 和 OF 为 0，AF 位无定义，而 SF、ZF 和 PF 位则根据运算结果设置。

在程序设计中，AND 一般用于对一个数据的指定位清 0；OR 指令常常用来对一些指定位置 1；NOT 指令常常用来将某个数据取反码；XOR 指令指令常用在程序的开头使某个寄存器清 0，以配合初始化工作的完成；TEST 指令执行与运算，但不保存结果，只根据结果设置相应的条件码标志，一般用于检测指定位是 1 还是 0。

2．移位指令

⑴非循环移位指令

格式：SHL/SHR/SAL/SAR OPR，CNT

CNT 表示移位的次数，只能是 1 或 CL，当移位次数大于 1 时，必须在移位指令前将移位次数置于 CL 寄存器中。各指令功能如图 3.8 所示。

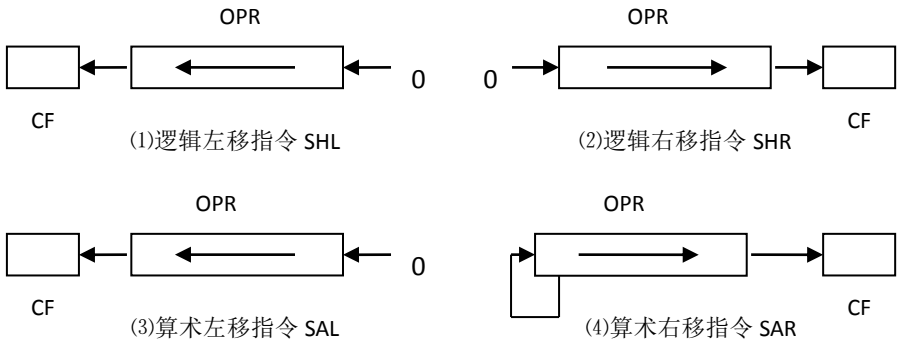


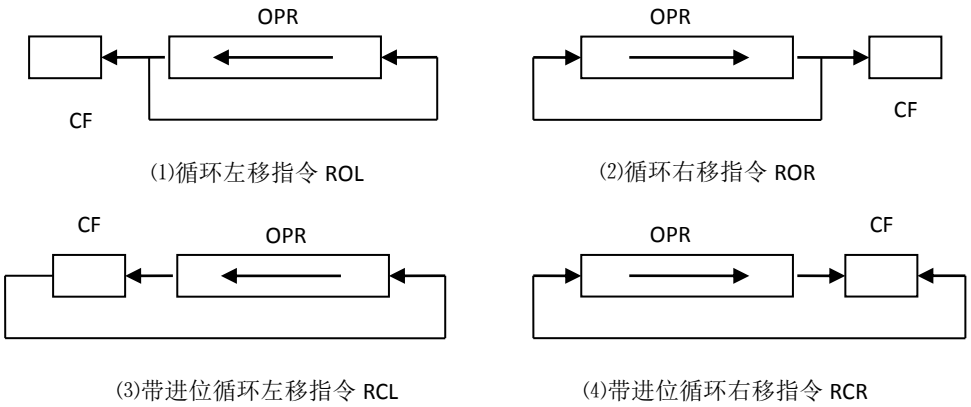
图 3.8 移位指令的功能

用移位指令时，逻辑移位指令适用于无符号数运算，算术移位指令则适用于带符号数运算，左移 1 位相对于将操作数乘 2，右移 1 位相对于将操作数除 2。由于移位指令执行速度快，可以编制一些常用的乘除法程序，比直接使用乘除法指令要快的多。

⑵循环移位指令

格式：ROL/ROR/RCL/RCR　OPR，CNT

CNT 表示移位的次数，和非循环移位指令一样，如果循环移位指令只移动一次，则在指令中直接指出，如果要移动若干位，则必须在 CL 寄存器中指定移位次数。各指令功能如图 3.9 所示。



⑴循环左移指令 ROL　⑵循环右移指令 ROR

⑶带进位循环左移指令 RCL　⑷带进位循环右移指令 RCR

图 3.9 循环移位指令的功能

七、控制转移指令

1．无条件转移指令 JMP

格式：JMP　OPR(目标地址)

功能：无条件地转移到指令指定的地址去执行从该地址开始的指令。

总的来说，转移可以分为段内转移和段间转移。段内转移是指在同一个代码段内进行转移，此时只需要改变 IP 寄存器中的值。段间转移是指程序从一个代码段转移到另一个代码段取执行程序，此时不仅改变 IP 的值，同时还要改变 CS 的值才能达到转移的目的。

2．条件转移指令

格式：JZ　OPR(目标地址)；条件转移指令比较多，以 JZ 为例说明格式

功能：条件转移指令以某一标志位的值或者某个比较结果作为判断是否进行转移的依据，如果满足指令中所要求的条件，则转移到目标地址 OPR 处执行程序，

否则按顺序执行排在条件转移指令后面的一条指令。所有的条件转移指令都是相对形式的，目标地址 OPR 只能在本条转移指令下一条指令地址的－128～＋127 个字节范围之内。

条件转移指令中，有一部分指令是根据对某一条件码标志的判断决定是否转移的，还有相当一部分指令在比较完两个数的大小以后，根据比较的结果决定是否转移的。对于具体的二进制数，将它们看成带符号数或无符号数，比较后得到的结果是不同的。

⑴条件转移指令根据某一条件码标志的值决定是否转移，这类指令如表 3.1 所示。

表 3.1 由条件码标志的值决定是否转移

操作码	含义
JZ/JE	ZF=1 条件成立，则转移
JNZ/JNE	ZF=0 条件成立，则转移
JS	SF=1 条件成立，则转移
JNS	SF=0 条件成立，则转移
JC	CF=1 条件成立，则转移
JNC	CF=0 条件成立，则转移
JO	OF=1 条件成立，则转移
JNO	OF=0 条件成立，则转移
JP	PF=1 条件成立，则转移
JNP	PF=0 条件成立，则转移

⑵条件转移指令根据两个无符号数的比较决定是否转移，这类指令如表 3.2 所示。

表 3.2 由两个无符号数的比较决定是否转移

操作码	含义
JB/JNAE	低于，即不高于且不等于，则转移
JBE/JNA	低于或者等于，即不高于，则转移
JA/JNBE	高于，即不低于且不等于，则转移
JAE/JNB	高于或者等于，即不低于，则转移

⑶条件转移指令根据两个带符号数的比较决定是否转移，这类指令如表 3.3

所示。

表 3.3 由两个带符号数的比较决定是否转移

操作码	含义
JG/JNLE	大于，即不小于且不等于，则转移
JGE/JNL	大于或者等于，即不小于，则转移
JL/JNGE	小于，即不大于且不等于，则转移
JLE/JNG	小于或者等于，即不大于，则转移

例 15：设 2000H 开始的区域中，存放着 100 个字节的无符号数，要求找出其中最大的一个数，并存到 2000H 单元。程序如下：

```
        MOV   BX，2000H
        MOV   AL，[BX]
        MOV   CX，99
LOP：   INC    BX
        CMP   AL，[BX]
        JAE    NEXT
        MOV   AL，[BX]
NEXT：  DEC   CX
        JNZ    LOP
        MOV   [2000H]，AL
```

3．循环控制指令

格式：LOOP OPR(目标地址)

功能：执行 LOOP 指令时，先将 CX 的内容减 1，然后判断 CX 是否为 0。如不为 0，则继续循环；如为 0，则退出循环，执行下一条指令。

因此，在使用 LOOP 指令前，必须用 MOV 指令对 CX 设置初值。

例 16：有一个首地址为 1000H 包含 20 个字的数组，要求求出该数组的内容之和(不考虑溢出)，并将结果存入 2000H 单元中。程序如下：

```
        MOV    SI，1000H
        MOV    AX，0
        MOV    CX，20
LOP：   ADD   AX，[SI]
```

```
ADD  SI，2
LOOP LOP
MOV  [2000H]，AX
```

4．子程序调用和返回指令

为便于模块化程序设计，往往把程序中某些具有独立功能的部分编写成独立的程序模块，称为子程序。程序可由调用程序(或称主程序)调用这些子程序，而在子程序执行完后又返回调用程序继续执行。8086 提供了子程序调用指令 CALL 和返回指令 RET。

⑴子程序调用指令 CALL

格式：CALL DST(子程序的入口地址)

功能：首先把子程序的返回地址存入堆栈，以便子程序返回调用程序时使用，然后转移到子程序的入口地址去继续执行。指令中的 DST 即为子程序的入口地址，也就是子程序的第一条指令的地址。

⑵返回指令 RET

格式：RET

功能：子程序执行完后返回调用程序继续执行。

和调用指令相对应的是返回指令。返回指令总是作为一个子程序的最后一条指令用来返回高一层的程序。返回指令在执行时，会从堆栈顶部弹出返回地址。为了正确的返回，返回指令的类型要和调用指令的类型相对应。

8086 的返回指令还有另外一种形式，叫带参数的返回指令，形式如下：

RET n

其中，n 可以为 0～FFFFH 范围内的任何一个偶数，表示这条指令从堆栈顶部弹出返回地址后，再使 SP 的值加上 n。

5．中断指令

⑴中断指令 INT

格式：INT n

功能：转到中断类型号为 n 的中断服务程序去执行。其中 n 为中断类型号，其值必须是在 0～255 范围内。

执行 INT n 指令时，将使 CPU 转到一个中断处理程序。此时，将标志寄存器

的值推入堆栈，堆栈指针 SP 减 2，然后清除中断允许标志 IF 和单步标志 TF，接着 CPU 将主程序的下一条指令地址即断点地址的段地址和偏移地址推入堆栈，同时堆栈指针 SP 减 4。

要进入中断处理程序，必须找到中断处理程序的入口地址。中断指令中直接给出了中断类型号，按照 8086 中断机制的原理，中断类型号的 4 倍就是中断向量的存放位置的偏移地址，而中断向量已有规则的排列在内存 0 段当中，中断向量的存放位置的段地址默认就是 0000H，中断向量就是中断处理程序的入口地址。因此，由中断类型号乘以 4 得到一个单元地址，由此地址开始的前 2 个单元存放的就是中断处理程序入口地址的偏移地址，后 2 个单元存放的就是中断处理程序入口地址的段地址。随后将中断处理程序入口地址的偏移地址和段地址分别送入 IP 和 CS 寄存器，这样 CPU 就会转入中断处理程序去执行。

⑵中断返回指令 IRET

格式：IRET

功能：返回中断处，执行后续指令。

各个中断处理程序的功能不同，但是，中断处理程序的最后一条指令总是 IRET。执行 IRET 指令时，先从堆栈中弹出 4 个单元的内容送入 IP 和 CS 寄存器，从而恢复断点地址，然后弹出标志寄存器的值。

第四章 存储系统

存储器是计算机的记忆部件，用来存放程序和数据。从用途和特点上可以分为两大类：

内存：计算机主机的重要组成部分，用来存放当前正在使用的或者经常使用的程序和数据，CPU 可以直接访问，但是容量受地址总线位数的限制。

外存：容量大，速度慢。用来存放暂时不用的程序和数据，使用时必须先调入内存才能使用。

一、存储器的分类

从功能上可以分为：RAM 和 ROM

1．RAM

静态 RAM（SRAM）：速度快，容量低，功耗大，不需要刷新。

动态 RAM（DRAM）：容量大，功耗低，需要刷新。

2．ROM

掩膜型 ROM：程序和数据由厂家写入，不能再修改。

PROM：用户可以根据自己的需要写入信息，以后不能再修改。

EPROM：可以擦除和重写，但是需要一种产生紫外光的专门设备，专业人士完成。

EEPROM：改变数据是在软件的控制下完成。

二、存储系统的层次结构

人们对存储器的要求是容量大、速度快、成本低，但这三个特性无法在同一存储器中兼得。为了解决这一矛盾，存储体系采用分级存储器结构，通常是将存储器分为高速缓冲存储器、主存储器和外存储器三级，如图 4.1 所示。通过硬件和管理软件组成一个既有足够大的存储空间又能保证满足 CPU 存取速度要求且价格适中的整体。

综上所述，一个较大的存储系统是由各种不同类型的存储设备构成，是一个具有多级层次结构的存储系统，各级存储器承担的职能各不相同。该系统中具有适当的存储容量和存取周期，使它能容纳系统的核心软件和较多的用户程序；高速缓冲存储器解决了存储系统与 CPU 运算速度相匹配的速度问题；辅助存储器则

解决了存储系统极大存储容量的容量问题。采用多级层次结构的存储器系统可以有效的解决存储器的速度、容量和价格之间的矛盾

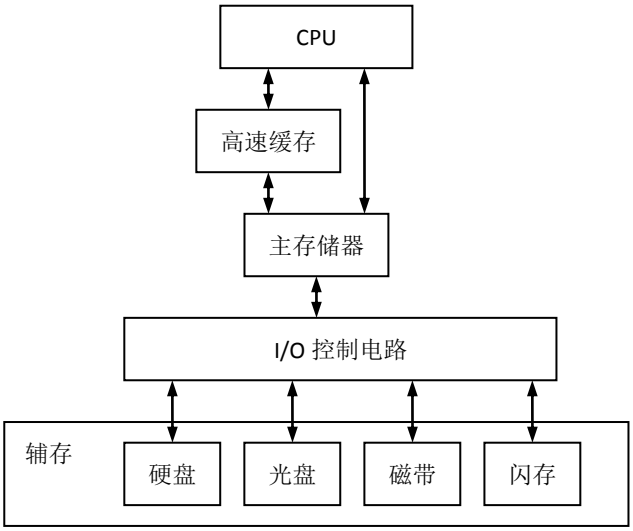


图 4.1 存储系统的多级存储结构

三、存储器扩展方法

由于生产的任何存储器芯片的容量是有限的，其在字数或字长方面和实际应用中对存储器的要求都有差距，所以为了满足实际存储器的容量要求，往往单个芯片不能满足字长或存储单元个数的要求，甚至字长、存储单元数两者都不满足要求。这时，就需要用多个存储芯片进行组合，对存储器进行字向和位向两方面的扩展。这种组合称为存储器的扩展，主要采用的方法有：位扩展法、字扩展法、字位同时扩展法。

1．位扩展

如果单个存储芯片字长（位数）不满足要求，这时就需要进行位扩展，以满足字长的要求。

例：使用 8K×1 的 RAM 存储器芯片，要求组成一个 8K×8 位的存储器。根据要求要用 8 片 8K×1 的存储芯片经位扩展构成 8K×8 位（即 8KB）的存储器，每个单元的 8 位二进制数被分别存在 8 个芯片上，可采用图 4.2 所示的位扩展法。

位扩展的特点：将每个存储芯片的地址线和控制线（包括片选信号线、读/写信号线等）全部并联在一起，而将各个芯片的数据线分别连接至数据总线的不同位上。使用位扩展法构成的存储器的每个单元中的内容被存储在不同的存储芯片上。

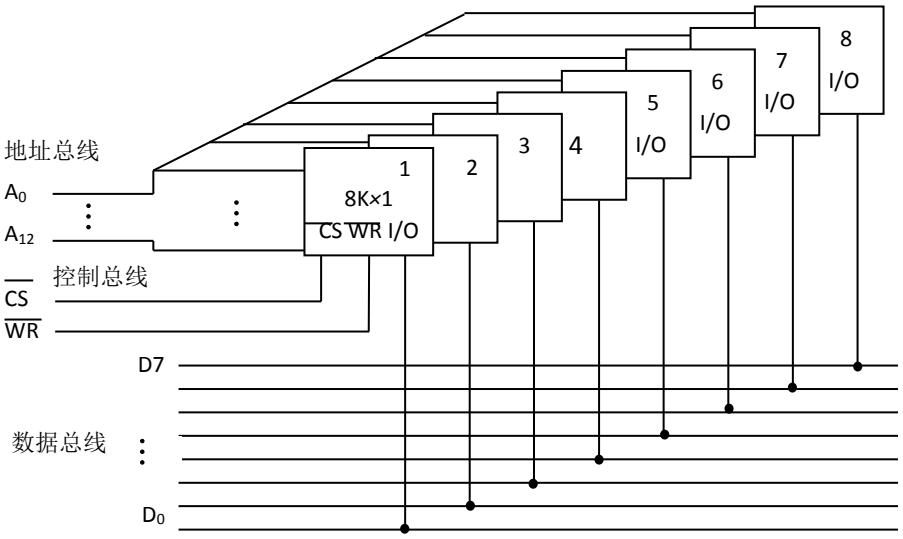


图 4.2 用 8K×1 位芯片组成 8K×8 位的存储器

2．字扩展

存储芯片上每个存储单元的字长已满足要求，只是存储单元的个数不够，需要增加的是存储单元的数量。

例：用 16K×8 位的芯片组成 64K×8 位的存储器，其字长已满足要求，只是容量不够，所以需要进行的是字扩展。显然，需要 4 片来实现，连接图如图 4.3 所示。

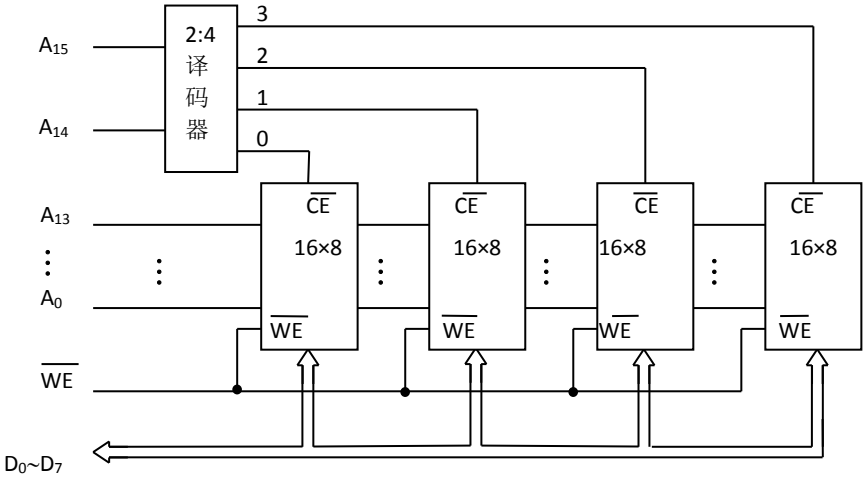


图 4.3　用 16K×8 位芯片组成 64K×8 位的存储器

字扩展的特点：将每个芯片的地址信号、数据信号和读/写信号等控制信号线按名称全部并联在一起，只将片选端分别和地址译码器的不同输出端相连。

第五章 输入输出系统

一、接口概述

接口技术：研究 CPU 和外设之间的数据传送方式、接口电路的工作原理和使用方法。

1．为什么使用接口电路？

①品种繁多； ②工作速度慢； ③信号类型与电平种类不同；④信息结构格式复杂。

存储器：功能单一，品种有限，速度较快，故可以直接连在总线上。

2．CPU 和输入输出设备之间的信号

⑴数据信息

这类信息可以是通过输入设备送到计算机的输入数据，也可以是经过计算机运算处理和加工后，送到输出设备的结果数据。在输入过程中，数据信息由外设经过外设接口之间的数据线进入接口，再到达系统的数据总线，从而送给 CPU。在输出过程中，数据信息从 CPU 经过数据总线进入接口，再通过接口和外设之间的数据线送到外设。

⑵状态信息

状态信息反映了当前外设的工作状态，是外设通过接口往 CPU 传送的。对于输入设备来说，通常用准备好信号(READY)信号来表明输入的数据是否准备就绪；对于输出设备来说，通常用忙(BUSY)信号表示输出设备是否处于空闲状态，如为空闲状态，则可接收 CPU 送来的信息，否则 CPU 要等待。

⑶控制信息

控制信息是 CPU 通过接口传送给外设的，CPU 通过发送控制信息控制外设的工作，外设的启动信号和停止信号就是常见的控制信息。

从含义上说，数据信息、状态信息和控制信息各不相同，应该分别传送。但在微型计算机系统中，CPU 通过接口和外设交换信息时，只有输入(IN)和输出指令(OUT)，所以，状态信息、控制信息也被广义地看成是一种数据信息。即状态信息作为一种输入数据，而控制信息作为一种输出数据。这样，状态信息和控制信息也通过数据总线来传送。

在接口中，这三种信息进入不同的寄存器。具体来说，CPU 送往外设的数据或者外设送往 CPU 的数据放在接口的数据缓冲器中，从外设送往 CPU 的状态信息放在接口的状态寄存器中，而 CPU 送往外设的控制信息要送到接口的控制寄存器中。称这些寄存器为 I/O 端口，每个端口有一个端口地址。一个外设需要几个端口地址。

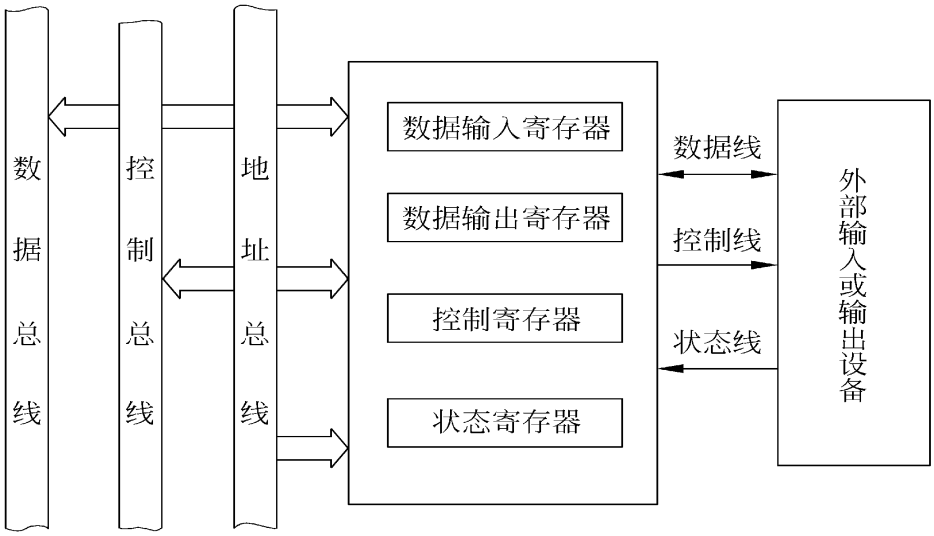


图 5.1 外设通过接口和系统的连接

用于对来自CPU和内存的数据或送向CPU和内存的数据起缓冲作用的端口叫数据端口；用于存放外部设备或接口本身的状态信息的端口叫状态端口；用于存放 CPU 发出的控制信息的端口叫控制端口。

二、CPU 和外设之间的数据传送方式

1．无条件传送方式：适用于随时读写的设备

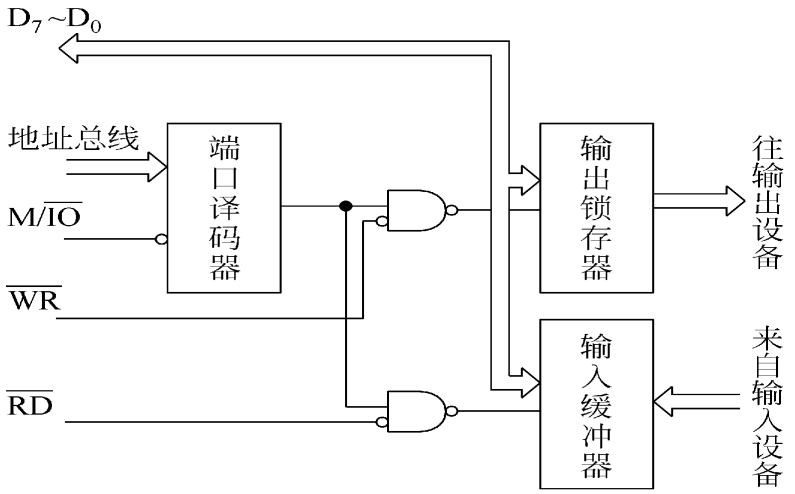


图 5.2 无条件传送方式工作原理图

2．程序查询方式（条件传送方式 ）

原理：CPU 通过执行程序不断地读取并测试外设的状态，如果外设处于准备好状态 (输入设备)或者空闲状态(输出设备)，则 CPU 执行输入指令或输出指令与外设交换信息。

对于程序查询方式来说，一个数据传送过程包括三个环节：

（1）CPU 从接口读取状态字。

（2）CPU 检测状态字的对应位是否满足“就绪”条件，如果不满足，则继续回到前一步读取状态字。

（3）如果状态字表明外设已处于“就绪”状态，则传送数据。

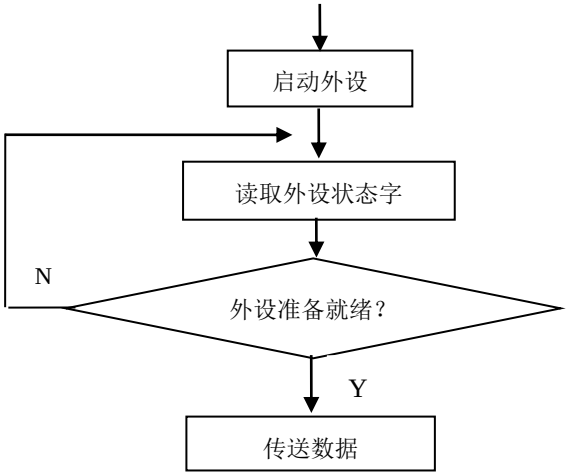


图 5.3 程序查询方式流程图

图 5.4 表明了用程序查询方式进行输入的接口电路的工作原理。输入设备将数据准备好之后便往接口发一个选通信号 STB，一方面将数据送入锁存器中，另一方面使接口中的 D 触发器输出 1，从而使接口中的三态缓冲器的 READY 置 1。数据信息和状态信息从不同的端口经过数据总线送到 CPU。按照数据传送过程的 3 个步骤，CPU 从外设输入数据之前先读取状态字，此时 READY 位已置 1，表明数据已进入到接口的锁存器中，这时 CPU 执行输入指令读取数据，同时状态位清零，这样便开始下一个数据传输过程。

图 5.5 表明了用程序查询方式进行输出的接口电路的工作原理。当 CPU 要往外设输出数据时，先读取接口中的状态字，如果状态字表明外设处于“空闲”状态，则说明可以向外输出数据，此时 CPU 执行输出指令，否则 CPU 必须等待。

CPU 执行输出指令时，由存储器/输入输出接口选择信号 和写信号 产生的

选通信号 STB 将数据总线上的数据存入接口中的数据输出锁存器，同时使 D 触发器输出 1。D 触发器的输出信号一方面告诉输出设备现在接口中已有数据可供提取；另一方面，D 触发器的输出信号使状态寄存器的对应标志位置 1，以便告诉 CPU 当前外设处于“忙”状态，从而阻止 CPU 再输出新的数据。

当输出设备从接口中取走数据后，通常会发送一个回应信号，回应信号使接口中的 D 触发器置 0，从而使状态寄存器中的对应标志位置 0，这样就可以开始下一个输出过程。

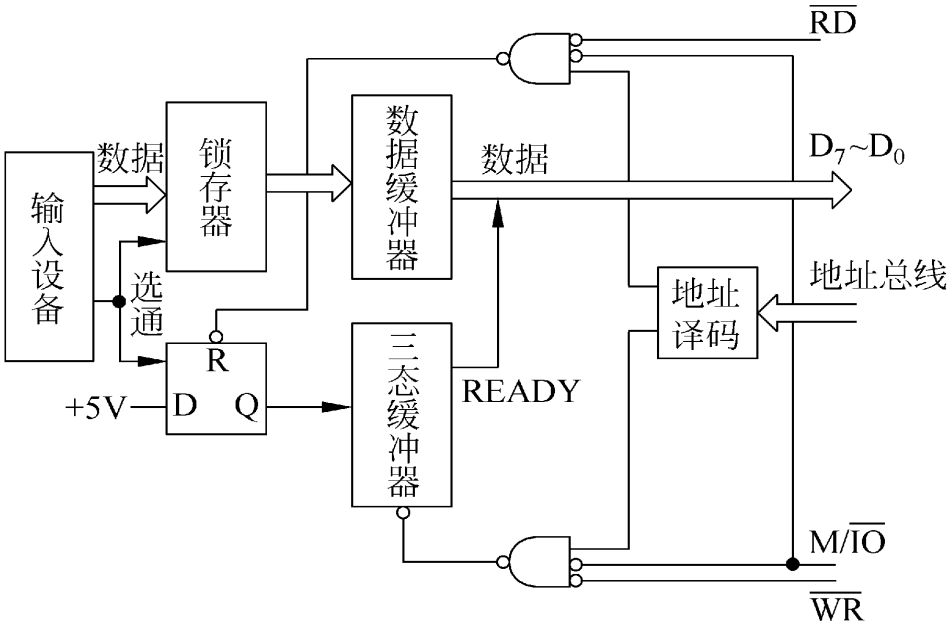


图 5.4 程序查询式输入接口

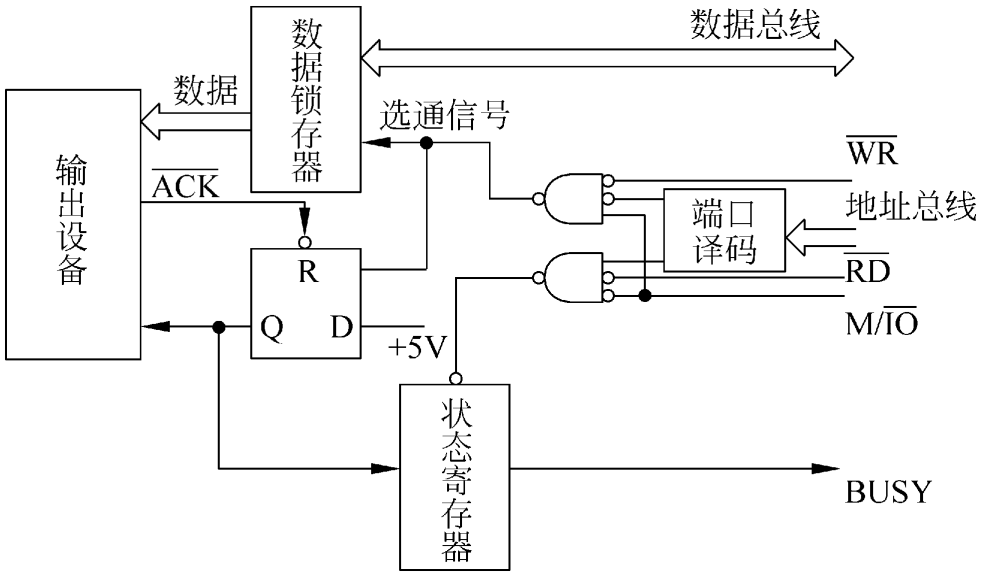


图 5.5 程序查询方式输出接口

在查询方式下，CPU 不断读取状态字和检测状态字，如果状态字表明外设未准备就绪，则 CPU 必须等待。其实质是让 CPU 降低有效的工作速度而适应慢速的外设。

使用程序查询方式进行输入输出操作时，如果系统中有多个利用查询方式进行输入输出的设备，通常是用轮流查询的方式来检测接口中的状态位。这时，显然 CPU 不能很好满足各个外设随机性地对 CPU 提出的输入/输出请求，因此，不具备实时性。

3．中断方式

在中断传送方式下，外部设备具有申请 CPU 服务的主动权，当输入设备将数据准备好或者输出设备准备好接收数据时，便可以向 CPU 发中断请求，使 CPU 暂时停下当前正在执行的程序而和外部设备完成一次数据传输。等输入/输出操作完成以后，CPU 继续执行原来的程序。

可见，中断方式就是外部设备中断 CPU 的工作，使 CPU 停止执行当前程序，而去执行一个数据输入输出程序，此程序称为中断处理子程序或中断服务子程序，中断处理子程序执行完后，CPU 又转回来执行原来的程序。

可以见得，在中断传送方式下，CPU 和外设处在并行的工作状态。CPU 不必在两个输入输出过程之间对接口进行状态检测和等待，而可以去作别的工作。

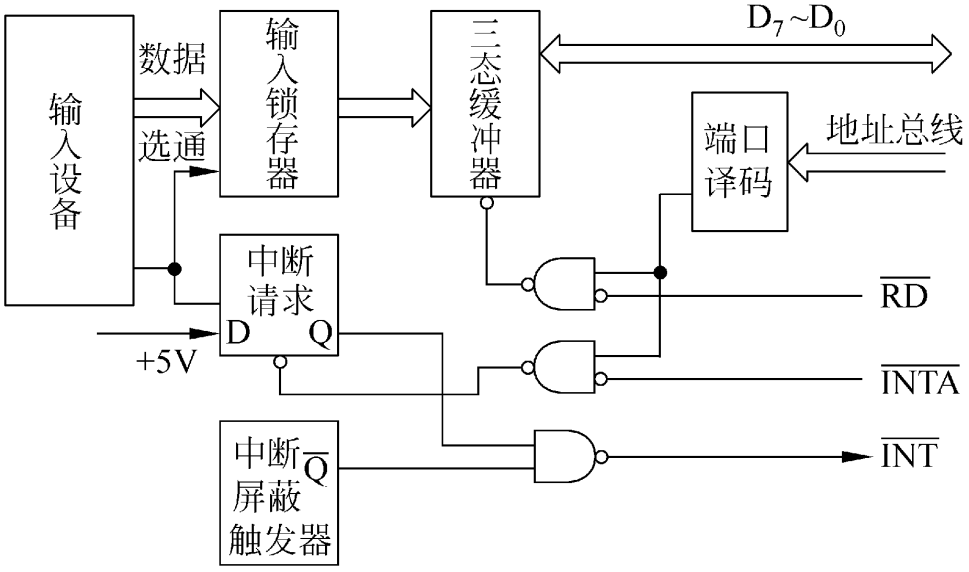


图 5.6 中断方式输入的接口电路

CPU 执行程序来说实现数据传送，每进行一次传送，CPU 执行一遍中断处理程序，此时都要保护断点和标志。此外，在中断处理程序中有一系列保护寄存器

和恢复寄存器的指令，这些指令与数据传送没有直接的关系。

在 8086 系统中，一旦进入中断，指令队列要清除，再装入中断处理程序的指令，而返回断点时，指令队列也要清除，重新装入断点处的指令继续执行。

4．直接存储器存取方式（DMA）

外部设备利用专门的接口电路直接和存储器进行高速数据传送，而不经过 CPU。当外围设备要求传送一批数据时，由 DMA 控制器向 CPU 发一个总线请求信号，要求 CPU 放弃对数据总线、地址总线和有关控制总线的使用权。DMA 控制器获得总线控制权之后，开始进行数据传送。在一批数据传送完毕后，DMA 控制器通知 CPU 可以使用总线，并把总线控制权交换给 CPU。

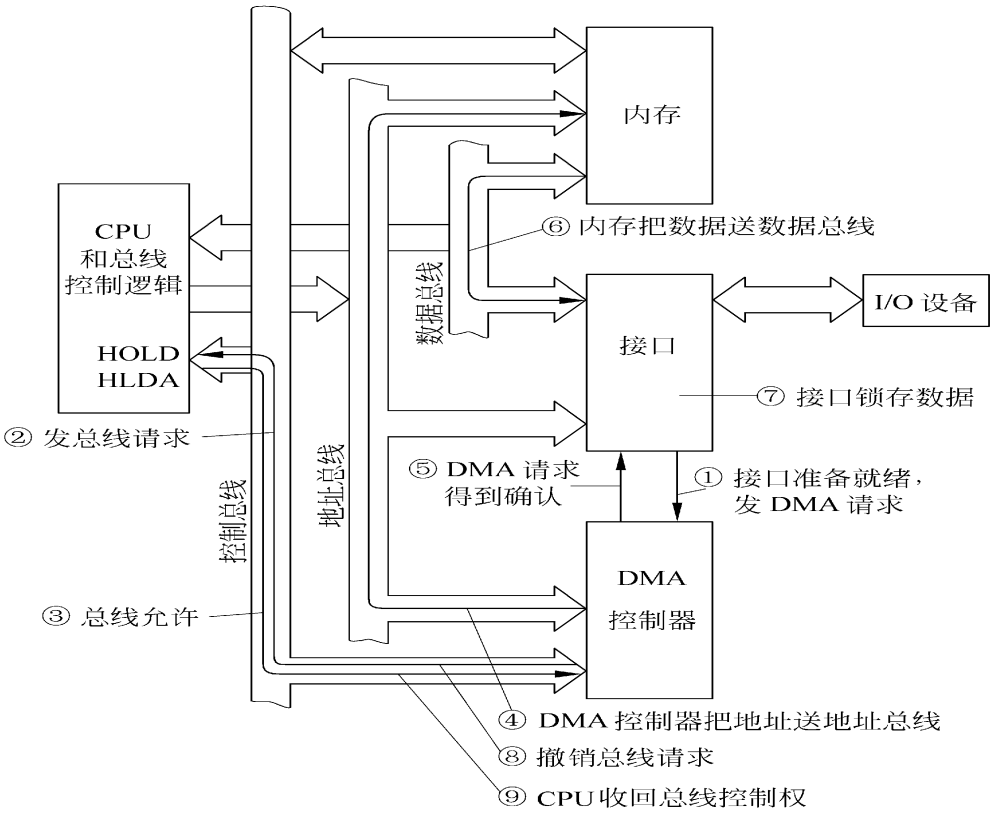


图 5.7 用 DMA 方式传送数据的过程图

DMA 控制器在外设与主存之间直接传送数据期间，完全代替 CPU 工作，它的主要功能有：

（1）当外设准备就绪，希望进行 DMA 操作时，会向 DMA 控制器发出 DMA 请求信号，DMA 控制器接到此信号后，会向 CPU 发出总线请求信号。

（2）当 CPU 响应此总线请求，发出总线响应信号后，DMA 控制器能接管对总线的控制，进入 DMA 操作周期。

（3）确定传送数据的主存单元地址及传送长度，并能自动修改内存地址计数值和字计数值。

（4）规定数据在内存与外设之间的传送方向，发出读写信号或其他控制信号，并执行数据传送操作。

（5）DMA 过程结束时，DMA 控制器能向 CPU 发出结束信号，并将总线控制权让给 CPU。

第六章 串并行通信和接口技术

一、并行接口概述

并行通信就是把一个字符的几个位同时进行传输。它具有传输速度快、效率高的优点。并行通讯所用的电缆较多，不适合长距离传输。所以，并行通信总是用在数据传输率要求较高，而传输的距离较短的场合。

能够实现并行通信的接口就是并行接口，典型的并行接口和外设连接如图 6.1 所示。

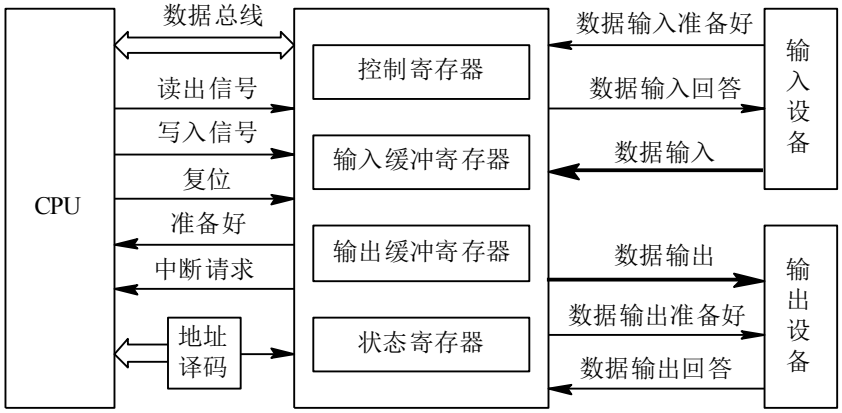


图 6.1 并行接口结构图

1. 并行输入接口工作过程

在输入接口的外设一侧，除了有外部数据线，一般还有两条联络线。一条叫作“数据输入准备好”，是外设送往接口的，它有效表示外设已经把数据送到接口的外部数据线上，它实际上起到打入脉冲的作用。 另一条叫作“数据输入回答”线，是接口送往外设的，它有效表示数据已经被送到接口的输入缓冲寄存器，而在 CPU 读取输入缓冲寄存器中的数据以后，“数据输入回答”线变无效，通知外设可以送下一个数据来。

典型的并行输入接口的工作过程：

（1）外设将数据送给接口，并且使联络线“数据输入准备好”成为高电平。

（2）接口在把数据接收到输入缓冲寄存器中的同时，使“数据输入回答”线变为高电平，作为对外设的响应。

（3）外设接到这个回答信号后，就撤除“数据输入准备好”信号。

（4）数据到达接口中后，接口会在状态寄存器中设置“输入准备好”状态位为“1“，以便 CPU 对其进行查询，接口也可以在此时向 CPU 发一个中断请求。

（5）CPU 既可以用软件查询的方式，也可以用中断方式来设法读取接口中的数据。

（6）CPU 从并行接口中读取数据后，接口会自动清除状态寄存器中的“输入准备好”状态位，并且使“数据输入回答”线变为低电平，以通知外设可以送下一个数据过来。此后，又可以开始下一个输入过程。

2. 并行输出接口工作过程

在输出接口的外设一侧，除了有外部数据线，一般还有两条联络线。一条叫作“数据输出准备好”，是接口送往外设的，它有效表示 CPU 已经把数据送到接口的输出缓冲寄存器，并出现在外部数据线上。它无效表示外设已经把数据取走，输出缓冲寄存器空。另一条叫作“数据输出回答”线，是外设送往接口的，它有效表示数据已经被外设接收。它无效表示外设还没有接收当前输出的数据。

下面是典型的并行输出接口的工作过程：

（1）在接口的输出缓冲寄存器中没有数据时，接口将状态寄存器中的“输出准备好”状态位置“1“，以表示接口输出缓冲寄存器为空，CPU 可以往接口中输出数据。

（2）如果 CPU 向接口输一个数据，数据被送到接口的输出缓冲寄存器后，接口会自动清除“输出准备好”状态位，并且将数据送往外设，与此同时，接口往外设发送一个“数据输出准备好信号”来通知外设接收数据。

（3）当外设取走数据之后，往接口发一个“数据输出回答”有效信号。

（4）接口收到此信号后，使“数据输出准备好” 信号变为无效，并将状态寄存器中的“输出准备好”状态位重新置“1”，以便 CPU 对其进行查询，接口也可以在此时向 CPU 发一个中断请求，以便 CPU 输出下一个数据。外设收到“数据输出准备好” 信号无效之后，也随之将“数据输出回答”信号置为无效。

二、可编程并行接口 8255A

1. 8255A 的内部结构和引脚

8255A 是可编程的并行输入输出接口芯片，它具有三个 8 位并行端口（A 口、B 口和 C 口），具有 40 个引脚，双列直插式封装，由+5V 供电，其内部结构和外部引脚分别如图 6.2 和图 6.3 所示。由图 6.2 可见，8255A 由数据端口 A、B、C(其中端口 C 被分成高 4 位和低 4 位两个部分)、A 组和 B 组控制部件、数据总线缓

冲器和读/写控制逻辑四个部分组成。

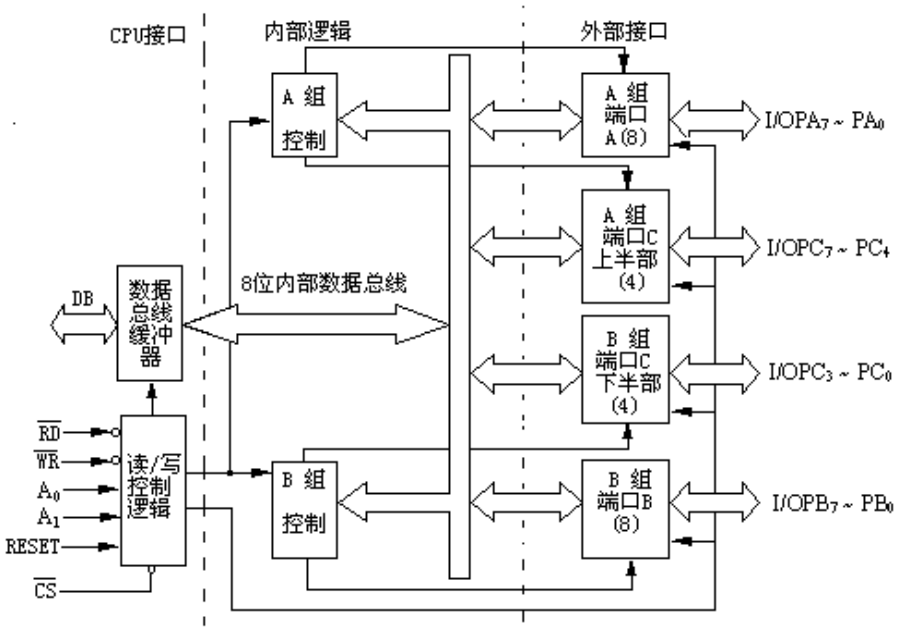


图 6.2 8255A 的内部结构

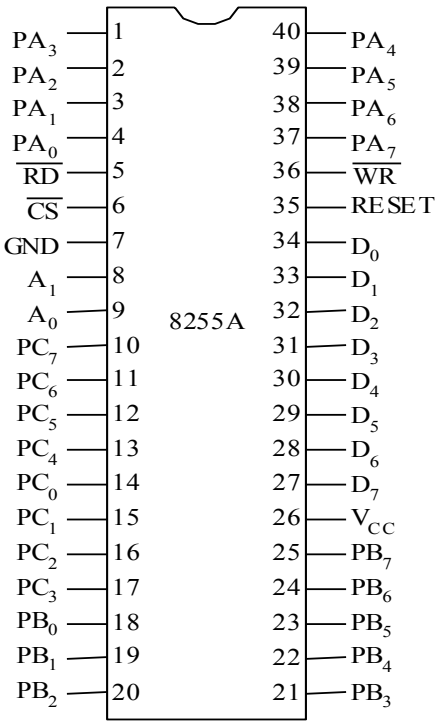


图 6.3 8255A 的引脚

⑴数据端口 A、B 和 C

端口 A、端口 B 和端口 C 都是 8 位端口，可以选择作为输入或输出。还可以将端口 C 的高 4 位和低 4 位分开使用，分别作为输入或输出。当端口 A 和端口 B 作为选通输入或输出的数据端口时，端口 C 的指定位与端口 A 和端口 B 配合使用，

用控制信号或状态信号。

⑵A 组和 B 组控制部件

端口 A 与端口 C 的高 4 位（PC7～PC4）构成 A 组，由 A 组控制部件实现控制功能，端口 B 与端口 C 的低 4 位（PC3～PC0）构成 B 组，由 B 组控制部件实现控制功能。它们各有一个控制单元，用来接收 CPU 送来的命令字，然后分别决定 A 组和 B 组的工作方式，或对端口 C 的某一位执行置位/复位操作。

⑶数据总线缓冲器

这是一个双向三态的 8 位缓冲器，用做 8255A 和系统数据总线之间的接口。这个缓冲器与 8255A 的外部引脚 D_7～D_0 连接。在接口电路中，通常将这 8 个引脚接至微机系统的数据总线 D_7～D_0。8255A 通过这个缓冲器和与之相连的 8 位数据总线 D_7～D_0 接收 CPU 送来的数据或控制字，外设通过 8255A 的某个端口传送给 CPU 的数据或状态信息，也要通过这个数据总线缓冲器送给 CPU。

⑷读/写控制逻辑

读/写控制逻辑电路的功能是负责管理 8255A 与 CPU 之间的数据传送过程。它能接收 CPU 的控制命令，并根据它们向片内各功能部件发出操作命令。可接收的控制命令如下：

①$\overline{CS}$：片选信号。由 CPU 输入，通常由端口的高位地址码（A15～A2）译码得到，有效，表示该 8255A 被选中。

②$\overline{RD}$、$\overline{WR}$：读、写控制信号。由 CPU 输入，$\overline{RD}$ 有效，表示 CPU 读 8255A，应由 8255A 向 CPU 传送数据或状态信息。$\overline{WR}$ 有效，表示 CPU 写 8255A，应由 CPU 将控制字或数据写入 8255A。

③RESET：复位信号。由 CPU 输入。有效时，清除 8255A 中所有内部寄存器，同时 3 个数据端口被自动设为输入口。

④A_1、A_0：端口地址选择信号。

当 A_1A_0=00，选择端口 A；

当 A_1A_0=01，选择端口 B；

当 A_1A_0=10，选择端口 C；

当 A_1A_0=11，选择控制字寄存器。

2．8255A 的控制字

8255A 有两个控制字，一个是方式字，用于选择工作模式；另一个是置位复位字，用于将端口 C 的任一指定位置 1 或置 0，这两个控制字共用一个地址号。

⑴方式选择控制字

8255A 的方式控制字由 8 位二进制数构成，各位的控制功能如图 6.4 所示。

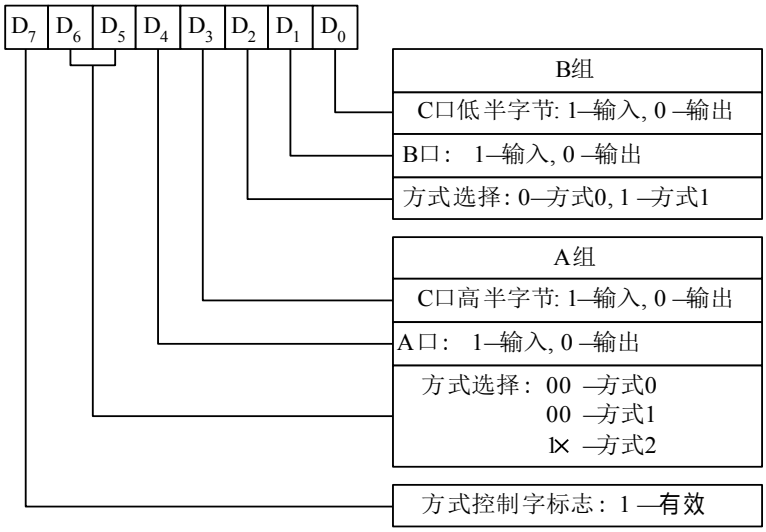


图 6.4 8255A 的方式选择控制字

⑵C 端口置 1/置 0 控制字

8255A 的端口 C 具有位控制功能，即端口 C 的 8 位中的任一位都可通过 CPU 向 8255A 的控制寄存器写入一个 C 端口置 1/置 0 控制字来置 1 或清 0，其格式如图 6.5 所示。

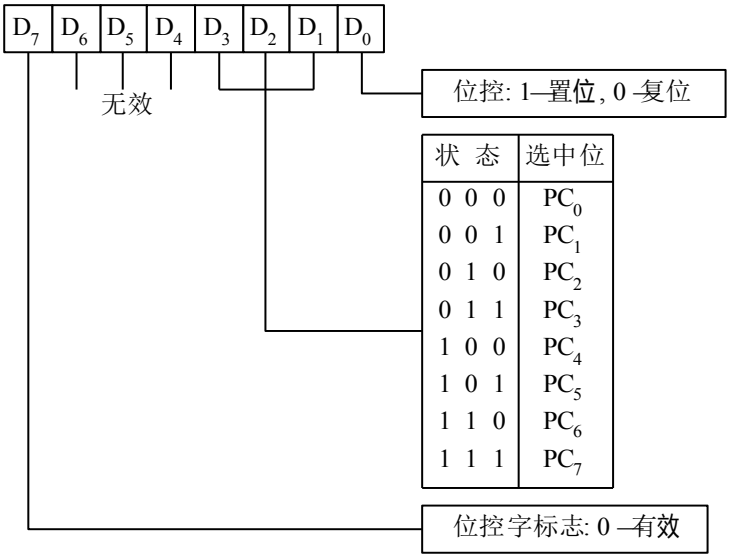


图 6.5 8255A 的 C 端口置 1/置 0 控制字

3. 8255A 的工作方式

⑴方式 0——基本输入/输出方式

方式 0 适用于不需要使用应答信号的简单输入/输出场合。在这种方式下，端口 A 和端口 B 可作为 8 位的输入端口或输出端口，端口 C 的高 4 位 和低 4 位可作为两个 4 位的端口，这 4 个端口中的任何一个既可作输入也可作输出。在实际应用时，端口 C 的两半部分也可以合在一起，构成一个 8 位的端口。

⑵方式 1——选通输入/输出方式

方式 1 只能进行单向传输，该工作方式下 8255A 与外设之间采用应答式传输数据，此时端口 C 自动提供选通信号和应答信号，只是输入操作与输出操作时端口 C 信号定义有些不同。

①方式 1 输入

端口 A 或 B 工作于方式 1 输入时，必须利用端口 C 的 6 条线作为控制和状态信号线，如图 6.6 所示。

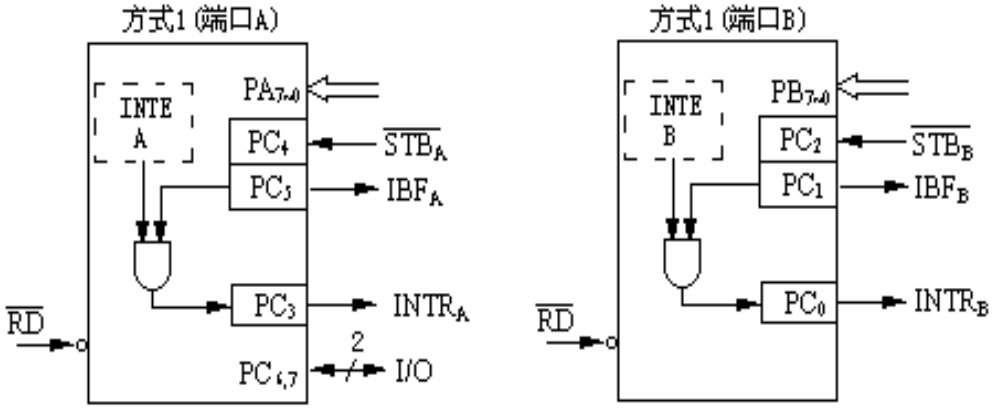


图 6.6 方式 1 输入时对应的端口状态

$\overline{STB}$：选通信号，输入，低电平有效。当$\overline{STB}$有效时，允许外设数据进入端口 A 或端口 B 的输入数据缓冲器。

IBF：输入缓冲器满信号，输出，高电平有效。当 IBF 有效时，表示当前已有一个新数据进入端口 A 或端口 B 缓冲器，尚未被 CPU 取走，外设不能送信的数据。CPU 完成数据读入操作后，IBF 便复位（变为低电平）。

INTR：中断请求信号，输出，高电平有效。若 IBF 和$\overline{STB}$均为高电平，则可使 INTR 有效，向 CPU 提出中断请求。

INTE：中断允许信号，高电平有效。在方式 1 下输入数据时，INTR 同样受中断允许状态 INTE 的控制。端口 A 的 INTEA 是由 PC4 控制的，端口 B 的 INTEB 是由 PC2 控制的。

当外设有数据需要输入时，将数据送到 8255A 口上，并利用 $\overline{STB}$ 脉冲将数据锁存，同时产生 INTR 信号并使 IBF 有效；有效的 IBF 通知外设数据已锁存，中断请求要求 CPU 从 8255A 的端口上读取数据；CPU 响应中断，读取数据后使 IBF 和 INTR 都变为无效。数据输入的时序图如图 6.7 所示。

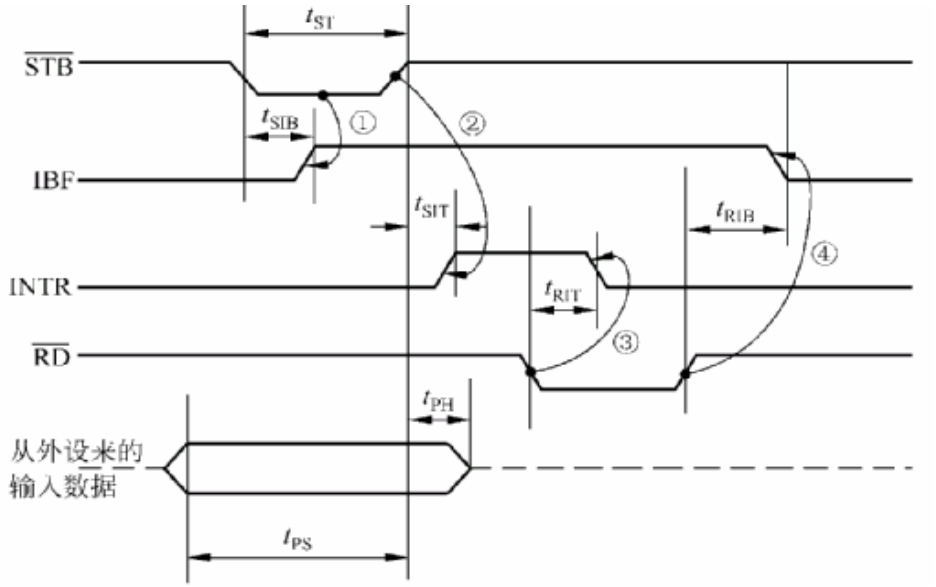


图 6.7 8255A 方式 1 输入时序图

②方式 1 输出

端口 A 和 B 均工作于方式 1 输出时，端口 A 使用 PC_3、PC_6 和 PC_7，而端口 B 使用 PC_0、PC_1 和 PC 各端口线的功能如图 6.8 所示。

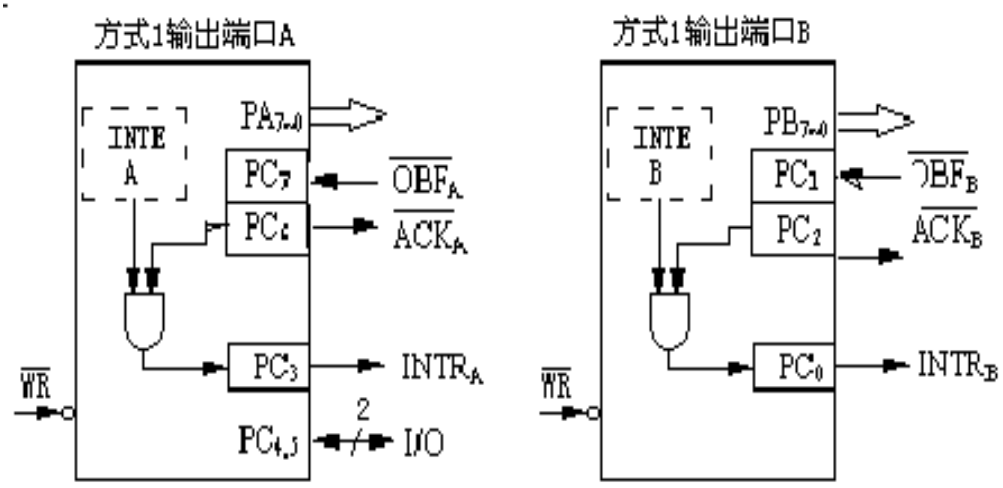


图 6.8 方式 1 输出时对应的端口状态

$\overline{OBF}$：输出缓冲器满信号，输出，低电平有效。当 CPU 已将要输出的数据送入 8255A 时有效，用来通知外设可以从 8255A 取数。

$\overline{ACK}$：外设应答信号，输入，低电平有效。当 $\overline{ACK}$ 有效时，表示 CPU 输出到 8255 的数据已被外设取走。

INTR：中断请求信号，输出，高电平有效。当外设接收到一个数据后，由此信号通知 CPU，刚才的输出数据已经被接收，可以再输出下一个数据。

INTE（Interrupt Enable）：中断允许信号，端口 A、B 输出的中断请求 INTR 可以通过对 PC 的位置 1/置 0 来加以允许或禁止。数据输出的时序图如图 6.9 所示。

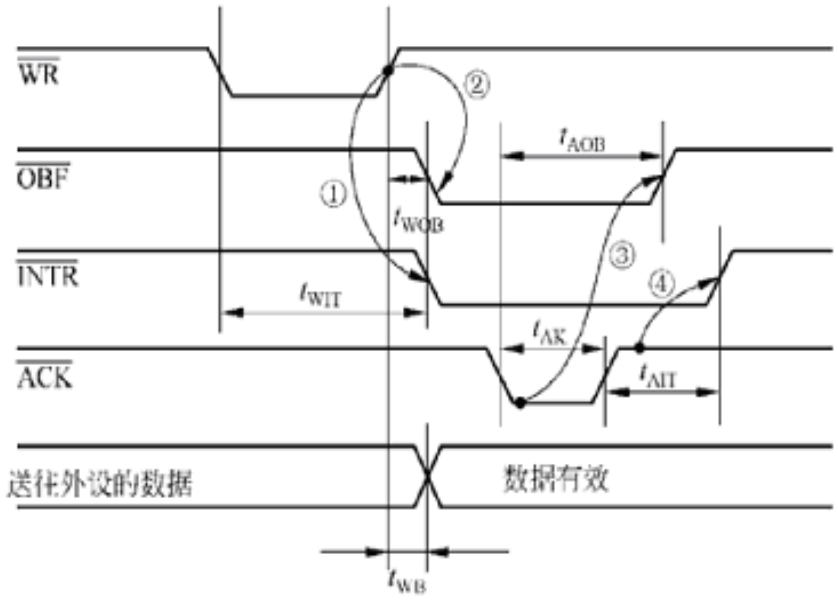


图 6.9 8255A 方式 1 输出时序图

⑶方式 2——双向传输方式

方式 2 只适用 A 端口。端口 A 工作于该方式 2 时，端口 C 有 5 位用于提供控制信号和状态信号，如图 6.10 所示。

4.8255A 的应用举例

例：8255A 的 A 口工作于方式 0 输入，接 8 个开关，B 口工作于方式 0 输出，接 8 个 LED 发光二极管。8255A 的连接电路如图 6.11 所示。试编写程序用开关控制对应的 LED 发光二极管亮。

由图可知 8255A 的端口地址为 120H～123H，A 口方式 0，输入，B 口方式 0，输出，则方式控制字为 10010000B。初始化程序为：

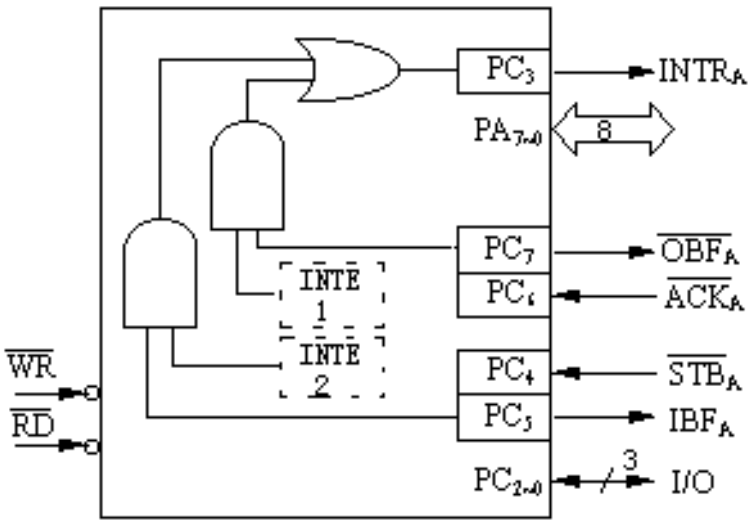


图 6.10 方式 2 对应的端口状态

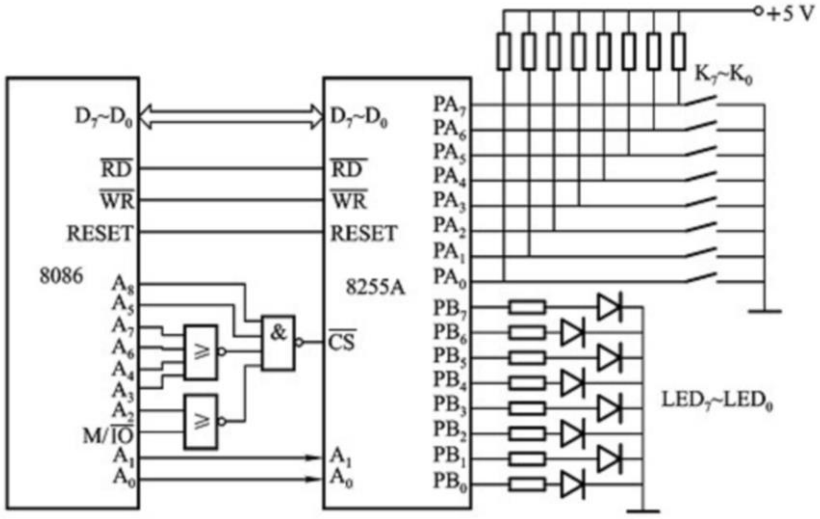


图 6.11 8255A 与开关状态连线图

```
MOV     DX，123H
MOV     AL，90H
OUT     DX，AL
MOV     DX，120H
IN   AL，DX
MOV     DX，121H
OUT     DX，AL
```

例： 8255A 作为并行打印机的接口。8255A 的 A 口连接打印机，工作于方式 1 输出，用查询方式将内存缓存区 BUF 中的 500 个字节数据送打印机输出，打印机接口电路如图 6.12 所示。写出相应的程序。

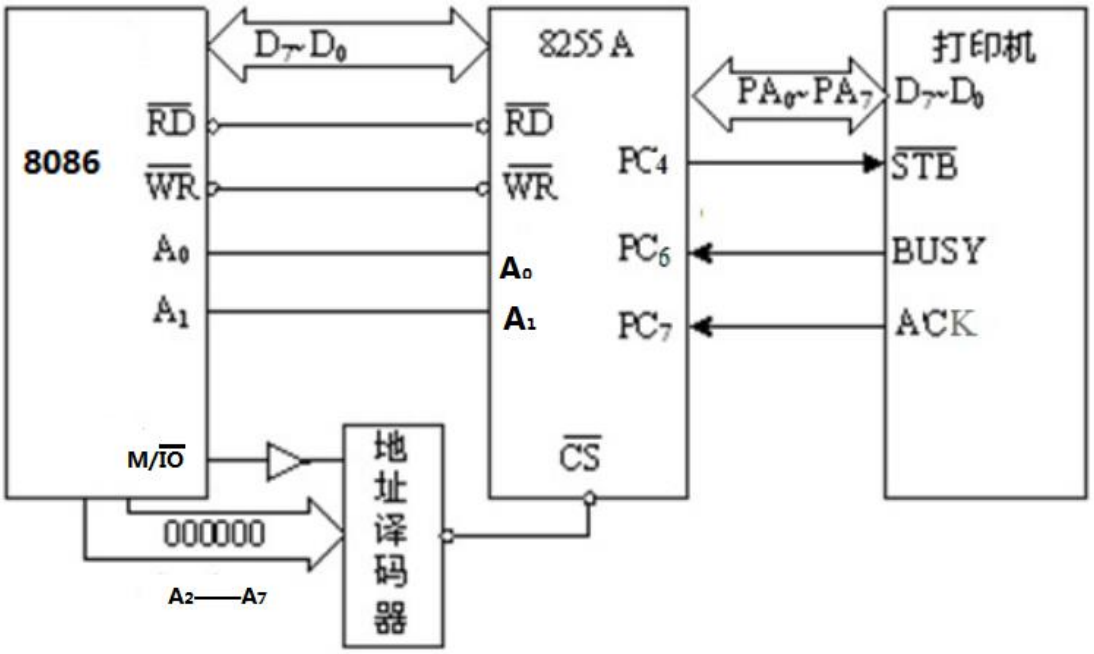


图 6.12 打印机连接电路图

```
        MOV   AL，0A8H
        OUT   03H，AL
        MOV   CX，500
        MOV   SI，OFFSET  BUF
NEXT：  IN    AL，02H
        TEST  AL，10H
        JNZ     NEXT
        MOV   AL，[SI]
        OUT   00H，AL
        INC     SI
        LOOP  NEXT
```

三、串行通信的基本概念

1.串行通信的定义

串行通信是指数据一位一位地依次传输，每一位数据占据一个固定的时间长度。只需要少数几条线就可以在系统之间交换信息，特别适用于计算机之间以及计算机与一些常用的外部设备之间的远距离数据交换，但串行通信的速度比较慢。

2．串行通信数据的传送模式

（1）单工通信模式：该模式仅能进行一个方向的数据传送。

（2）半双工通信模式：交替的进行双向数据传送。

（3）全双工通信模式：两个设备均能在发送的同时接收数据。

3．串行通信数据的收发方式

在数据通信中为使收、发信息准确，收发两端的动作必须相互协调配合。这种协调收发信息之间动作的措施称为“同步”。在串行通信中数据传送的同步方式有异步通信和同步通信两种。

（1）异步通信

异步通信是指发送设备和接收设备在约定的波特率下，不需要严格的同步，允许有相对的延迟。即两端的频率差别在 1/10 以内，就能正确的实现通信。在进行异步传送时必须确定字符格式即波特率。异步通信的数据传送格式如图 6.13 所示。

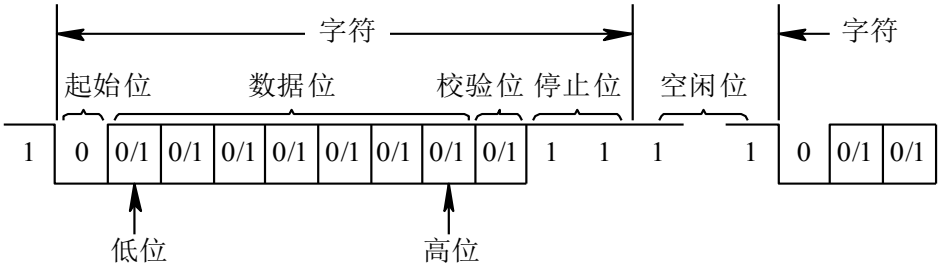


图 6.13 异步通信数据传输格式

（2）同步通信

所谓同步通信，是指在约定的波特率下，发送设备和接收设备的频率保持一致（同步）。因为发送和接收的每一位数据均保持同步，故传送信息的位数几乎不受限制。传送时在被数据的开头加上同步字符，同步字符的格式和个数可以根据需要来确定，它被接收设备作为确定数据块的起始界限。同步通信字符格式如图 6.14 所示。

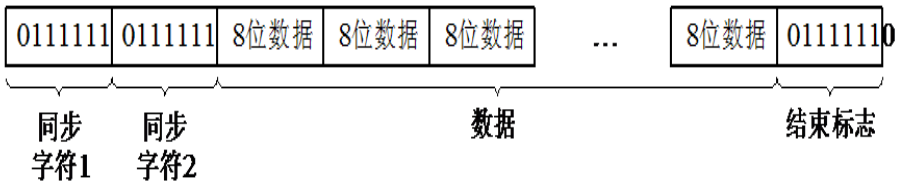


图 6.14 同步通信数据格式

四、可编程串行通信接口 Intel 8251A

1. 8251A 内部结构

8251A 由接收器、发送器、调制解调器控制电路、读写控制电路及数据总线缓冲器五部分组成。各部分之间由内部总线实现相互通信。它的结构如图 6.15 所示。

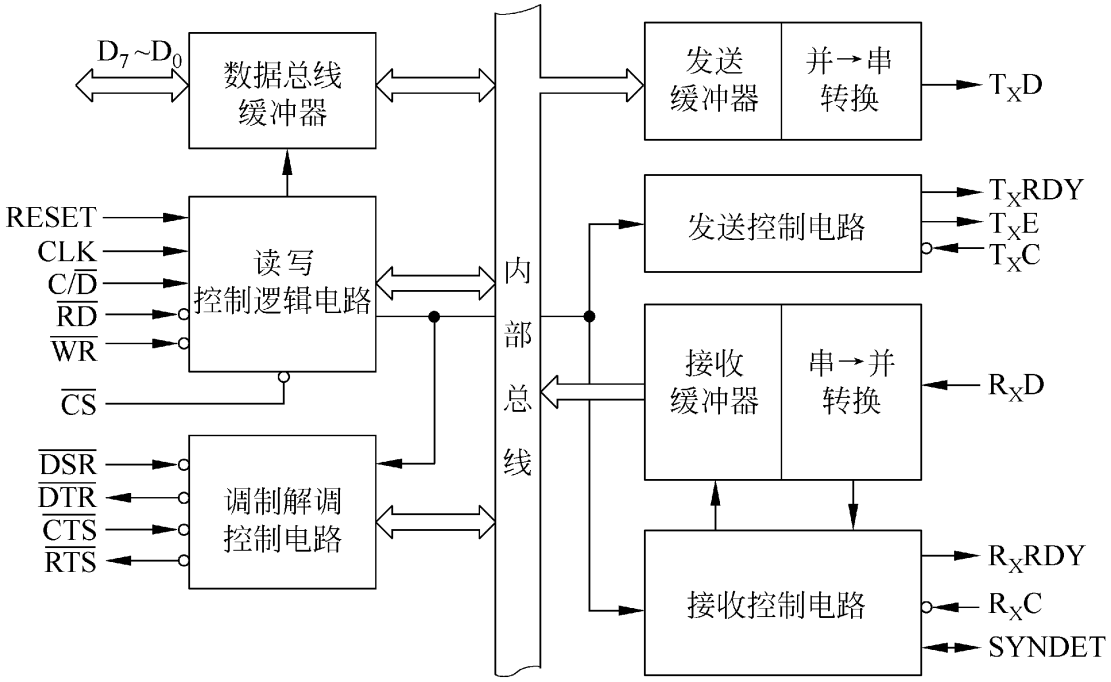


图 6.15 8251A 内部结构

（1）接收器

8251A 接收器由接收缓冲器和接收控制电路两部分组成。负责接收 R_xD 引线上输入的串行数据，并按规定方式将它们转换为并行数据，存放在接收数据缓冲寄存器中。

在异步方式，8251A 在允许接收和准备好接收数据时，监视 RXD 线。在无字符传送时，R_xD 线上为高电平，当发现 R_xD 数据线上出现低电平时，就启动一个内部计数器，当计数到一个数据位宽度的一半时（例如，波特率系数为 16，则为计数至第 8 个接收时钟脉冲时），重新采样 R_xD 线，若其仍为低电平，则确认它是起始位，而不是噪声信号。此后，每隔 16 个脉冲，采样一次 R_xD 线作为输入信号，送至移位寄存器，经过移位，又经过奇偶校验和去掉停止位后，就得到了变换为并行的数据，经 8251A 的内部数据总线传送至接收数据缓冲器，同时发出 R_xRDY 信号，告诉 CPU 字符已经可用。

在同步方式，8251A 监视 R_xD 线，以一次一位的方式移动数据。接收到每一位后，都将接收寄存器与存放同步字符的寄存器相比较。若不相等，则在移入下位后继续比较；若相等，则表示搜索到同步字符。于是使 SYNDET 为高电平，表示已达到同步（若规定为两个同步字符，则必须在检测到两个同步字符后才认为已达到同步）。然后利用 R_xC 时钟采样和移位 R_xD 线上的数据位，并按规定位数将其送至接收数据缓冲器，同时发出 R_xRDY 信号。

（2）发送器

8251A 发送器由发送缓冲器和发送控制电路两部分组成。负责接收 CPU 送来的并行数据，自动加上由控制字规定的成帧信号，再将其变换成串行数据从 TxD 引脚上送出去。

在异步方式时，发送器总是要加上起始位，并根据控制字规定加上奇偶校验位和停止位。在同步方式下，发送器最先发送的是同步字符，而在随后发送的数据中除了奇偶校验位外不再插入别的位，如果在发送过程中 CPU 未及时提供发送数据，发送器就自动发送同步字符，在同步通信中，两个字符之间不允许有间隔，不同于异步通信。

（3）读写控制电路

读写控制电路负责接收 CPU 输出的控制信号，在内部进行译码后实现相应的操作。

（4）调制解调控制器

当使用 8251A 实现远距离串行通信是，8251A 的数据输出端要经过调制器将数字信号转换成模拟信号，数据接收端收到的是经过解调器转换来的数字信号，因此 8251A 要与调制解调器直接相连。

（5）数据总线缓冲器

数据总线缓冲器是 CPU 与 8251A 之间信息交换的通道。它包含 3 个 8 位缓冲寄存器，其中两个用来存放 CPU 向 8251A 读取的数据或状态字，当 CPU 执行 IN 指令时，从这两个寄存器中读取数据字及状态字。另一个缓冲寄存器存放 CPU 向 8251A 写入的数据或控制字。当 CPU 执行 OUT 指令时，可向这个寄存器写入，由于两者共用一个缓冲寄存器，这要求 CPU 在向 8251A 写入控制字时，该寄存器中无将要发送的数据。

2．8251A 的控制字及工作方式

8251A 在使用前必须进行初始化，以确定它的工作方式，传送速率、字符格式以及停止位长度等。下面就 8251A 中使用的三种控制字：方式选择控制字、操作命令控制字和状态控制字及初始化编程进行介绍。8251A 编程结构如图 6.16 所示。

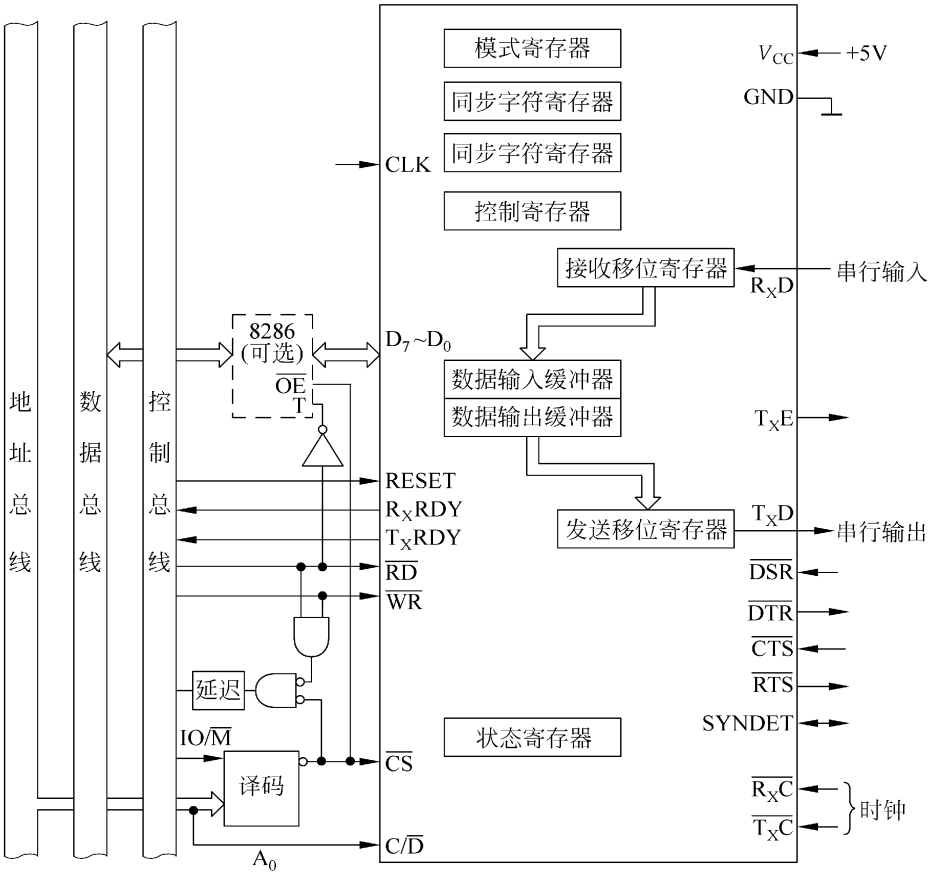


图 6.16 8251A 的编程结构图

⑴模式寄存器的格式

模式寄存器用来确定 8251A 的通信方式（同步或异步）、校验方式（奇校验、偶校验或不校验）、数据位数（5、6、7 或 8 位）及波特率参数等。模式寄存器的格式如图 6.17 所示。

⑵控制寄存器的格式

控制寄存器使 8251A 进入规定的工作状态以准备发送或接收数据，它应该在写入方式控制字后写入，用于控制 8251A 的工作，可以多次写入。命令控制字格式如图 6.18 所示。

⑶状态寄存器的格式

状态寄存器存放 8251A 的状态信息，供 CPU 查询。状态字各位的意义如图 6.19 所示。

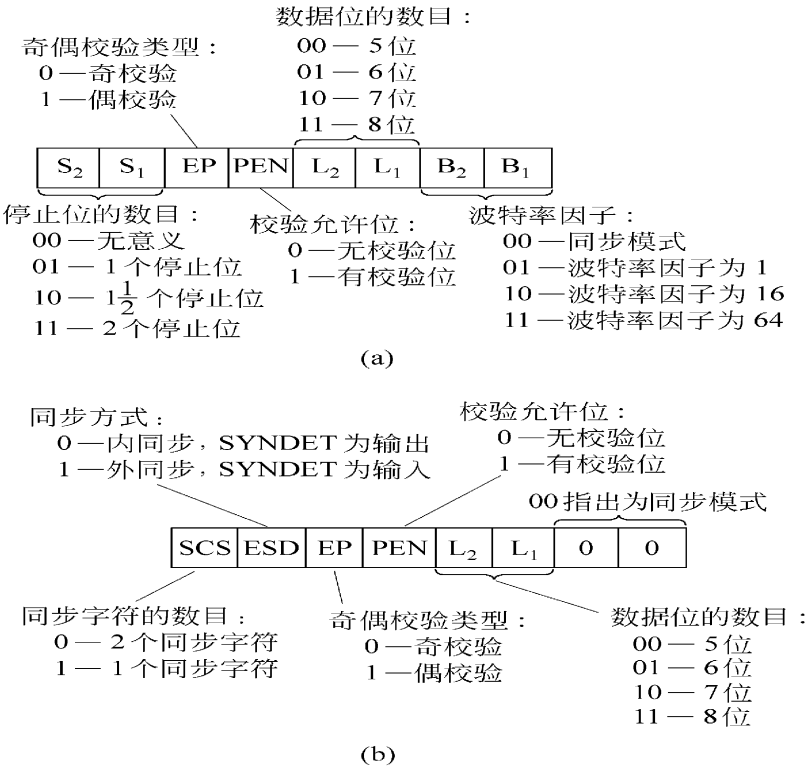


图 6.17 模式寄存器的格式

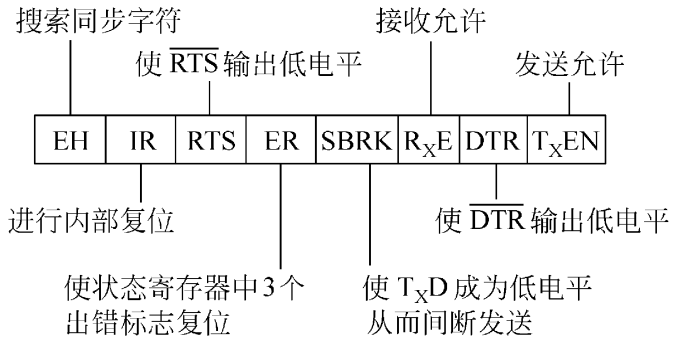


图 6.18 控制寄存器的格式

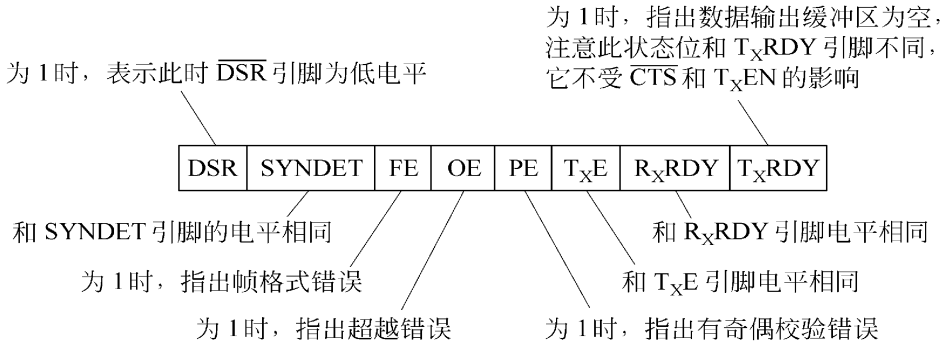


图 6.19 状态寄存器的格式

3. 8251A 初始化编程

对 8251A 进行初始化编程。必须在系统复位之后。其过程为：首先写入方式控制字，以决定通信方式、数据位数、校验方式等。若是同步方式则紧接着进入一个或两个同步字符，若是异步方式则这一步可省略，最后送入命令控制字，就可以发送或接收数据了。8251A 初始化过程的流程如图 6.20 所示。

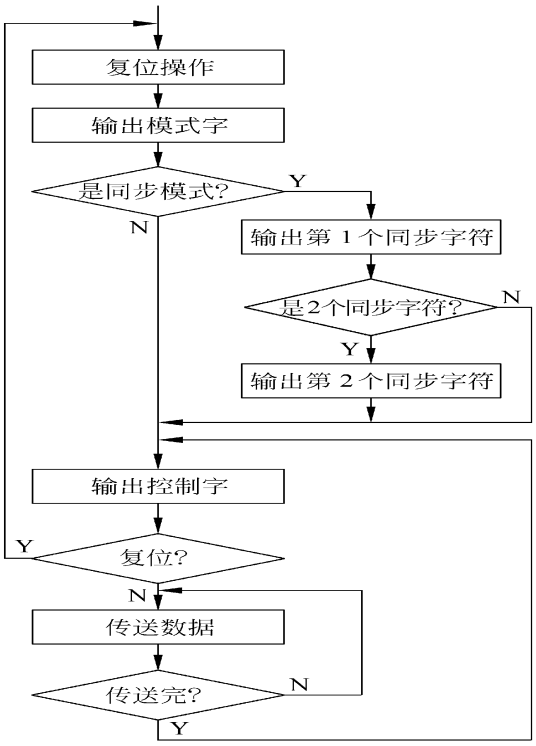


图 6.20 8251A 的初始化流程

4. 8251A 应用举例

⑴异步方式下初始化编程举例

设 8251A 工作在异步模式，波特率因子为 16，7 个数据位/字符，偶校验，2 个停止位，发送、接收允许，设端口地址为 00E2H 和 00E4H。完成初始化程序。

解：根据题目要求，可以确定模式字为：11111010B 即 FAH；控制字为：00110111B 即 37H；则初始化程序如下：

```
MOV  AL，0FAH
MOV  DX，00E2H
OUT  DX，AL
MOV  AL，37H
OUT  DX，AL
```

⑵同步模式下初始化程序举例

设端口地址为52H，采用内同步方式，2个同步字符（设同步字符为16H），偶校验，7位数据位/字符。

根据题目要求，可以确定模式字为：00111000B　即 38H，而控制字为：10010111B　即97H。它使8251A对同步字符进行检索；同时使状态寄存器中的3个出错标志复位；此外，使8251A的发送器启动，接收器也启动；控制字还通知8251A，CPU当前已经准备好进行数据传输。

```
MOV  AL，38H
OUT  52H，AL
MOV  AL，16H
OUT  52H，AL
OUT  52H，AL
MOV  AL，97H
OUT  52H，AL
```

第七章 中断管理与定时器

一、中断控制器 8259A

中断控制器的功能：

在有多个中断源的系统中，接受外部的中断请求，并进行判断，选中当前优先级最高的中断请求，再将此请求送到 CPU 的 INTR 端；当 CPU 响应中断并进入中断处理子程序后，中断控制器仍负责对外部请求的管理，当某个外部中断请求的优先级高于当前正在处理的中断优先级时，中断控制器会让此中断通过而到达 CPU 的 INTR 端，从而实现中断的嵌套。

1. 8259A 的内部结构和引脚

⑴ 8259A 的内部结构

8259A 的内部结构如图 7.1 所示，8259A 由以下八个模块组成。

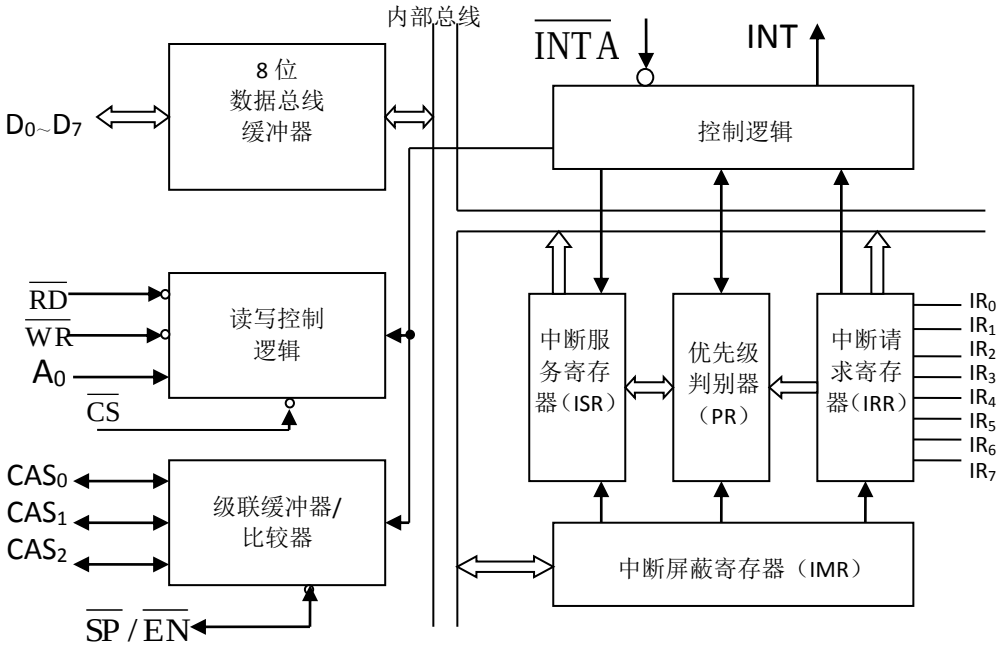


图 7.1 8259A 的内部结构

①数据总线缓冲器

数据总线缓冲器它是 8 位双向三态缓冲器，是 8259A 与系统数据总线的接口。CPU 通过它写入 8259A 控制字，8259A 状态信息通过它读入 CPU。在中断响应周期，8259A 送出的中断向量也是通过它传给 CPU。

②读/写控制逻辑

读/写控制逻辑用来接收来自 CPU 的读/写命令，配合片选信号 $\overline{CS}$ 和 A_0 端的地址输入信号完成对 8259A 内部寄存器的读/写操作。

③级联缓冲器

级联缓冲器提供多片 8259A 的管理和选择功能。在级联方式的主从结构中存放和比较每个 8259A 从设备的识别码。

④控制逻辑电路

中断控制逻辑按照编程设定的工作方式管理中断，负责向片内各部件发送控制信号，向 CPU 发送中断请求信号 INT 和接收 CPU 的中断响应信号 $\overline{INTA}$，控制 8259A 进入中断管理状态。

⑤中断请求寄存器 IRR

IRR 是一个 8 位的寄存器，用来存放 8 条外部中断请求信号，它的 8 个位 D_0～D_7 分别与外部中断请求信号线 IR_0～IR_7 相对应，当 IR_i 有请求（电平或边沿触发）时，IRR 中的相应位 Di 置“1”。

⑥中断屏蔽寄存器 IMR

IMR 是一个 8 位的寄存器，用来对外部中断请求线 IR_0～IR_7 上的中断请求信号禁止或允许的寄存器。若某位置“1”，与它对应的中断请求被禁止。

⑦中断服务寄存器 ISR

ISR 是一个 8 位的寄存器，用来存放所有正在进行服务的中断请求。若某位为“1”，表示正在为相应的中断源服务。

⑧优先权电路

优先权电路用来确定存放在 IRR 中各个中断请求信号对应的中断源的优先级，并可以对它们进行排队判优，以便选出当前中断请求优先级最高的中断，由中断控制逻辑向 CPU 发出中断请求信号 INT。

⑵82591A 的引脚

8259A 是具有 28 个引脚的双列直插式集成电路芯片，可以分为与 CPU 连接的引脚和与外设连接的引脚。

①与 CPU 连接的引脚

D_7～D_0：双向数据输入/输出引脚，用以与 CPU 进行信息交换。

$\overline{RD}$：读控制信号输入引脚，低电平有效，实现对 8259A 内部有关寄存器内

容的读操作。

$\overline{WR}$：写控制信号输入引脚，低电平有效，实现对 8259A 内部有关寄存器的写操作。

$\overline{CS}$：片选信号输入引脚，低电平有效，一般由系统地址总线的高位，经译码后形成，决定了 8259A 的端口地址范围。

A_0：8259A 两组内部寄存器的选择信号输入引脚，决定 8259A 的端口地址。

INT：中断请求信号输出引脚，高电平有效，用以向 CPU 发中断请求，应接在 CPU 的 INTR 输入端。

$\overline{INTA}$：中断响应信号输入引脚，低电平有效，在 CPU 发出第二个$\overline{INTA}$时，8259A 将其中最高级别的中断请求的中断类型码送出，应接在 CPU 的$\overline{INTA}$中断应答信号输出端。

$\overline{SP}/\overline{EN}$：$\overline{SP}$为级连管理信号输入引脚，在非缓冲方式下，若 8259A 在系统中作从片使用，则$\overline{SP}=1$；否则$\overline{SP}=0$；在缓冲方式下，$\overline{EN}$用作 8259A 外部数据总线缓冲器的启动信号。

②与外设连接的引脚

IR_7～IR_0：8 级中断请求信号输入引脚，规定的优先级为 IR_0＞IR_1＞…＞IR_7，当有多片 8259A 形成级联时，从片的 INT 与主片的 IR_i 相连。

CAS_2～CAS_0：级联信号引脚，当 8259A 为主片时，为输出；否则为输入，与$\overline{SP}/\overline{EN}$信号配合，实现芯片的级连，这三个引脚信号的不同组合 000～111，刚好对应于 8 个从片。

2.8259A 的工作方式

⑴优先权的管理方式

①全嵌套方式

在全嵌套方式下，8259A 所管理的 8 级中断优先权是固定不变的，其中 IR_0 的中断优先级最高，IR_7 的中断优先级最低。

当 CPU 响应中断请求后，在中断服务寄存器 ISR 中的相应位置 1，而且把它的中断类型号送至系统数据总线，然后进入中断服务程序。在中断服务完成之前，

和它同级或优先级低的中断源的中断请求被屏蔽，只有优先级比它高的中断源的中断请求才是允许的，从而出现中断嵌套。

②特殊全嵌套方式

特殊全嵌套方式与全嵌套方式基本相同，所不同的是，当 CPU 处理某一级中断时，如果有同级中断请求，那么 CPU 也要做出响应，从而形成了对同一级中断的特殊嵌套。

③优先级自动循环方式

中断源优先级可以动态改变，系统初始优先级由高到低排列为 IR_0～IR_7。某一个中断源的中断请求得到响应之后，其优先级自动降为最低，比它低一级的中断源优先级变为最高，其余按序循环。

④优先级特殊循环方式

优先级特殊循环方式与自动循环方式相比，只有一点不同，即初始化的优先级是由程序控制的。如指定 IR4 优先级最低，则 IR5 优先级最高，初始优先级顺序 IR_5、IR_6、IR_7、IR_0、IR_1、IR_2、IR_3、IR_4 由高到低排列。

⑵中断源的屏蔽方式

①普通屏蔽方式

8259A 的每个中断请求输入，都要受到屏蔽寄存器中相应位的控制。若相应位为“1”，则中断请求不能送 CPU。

②特殊屏蔽方式

特殊屏蔽方式能在一个中断服务程序的运行过程中，动态地改变系统中的中断优先级结构，即在中断处理的一部分，禁止低级中断，而在中断处理的另一部分，又能够允许低级中断，采用了这种方式之后，尽管系统正在处理高优先级中断，但对外界来讲，只有同级中断被屏蔽，而允许其他任何级别的中断请求。

⑶中断结束处理方式

当中断服务结束时，必须给 8259A 的 ISR 相应位清 0，表示该中断源的中断服务已结束，使 ISR 相应位清 0 的操作称为中断结束处理，包括如下三种方式。

①中断自动结束方式

该方式仅适用于只有单片 8259A 的场合，在这种方式下，系统一旦响应中断，那么 CPU 在发第二个 $\overline{INTA}$ 脉冲时，就会使中断响应寄存器 ISR 中相应位复位，

这样一来，虽然系统在进行中断处理，但对于 8259A 来讲，ISR 没有相应的指示，就像中断处理结束，返回主程序之后一样。

②一般的中断结束方式

该方式适用于全嵌套的情况下，当 CPU 用输出指令向 8259A 发一般中断结束命令 EOI 时，8259A 才会使中断响应寄存器 ISR 中优先级别最高的位复位。

③特殊中断结束方式

发中断结束命令的同时，需用软件方法指出中断响应寄存器 ISR 中需要复位的位。适用于 CPU 无法确定当前所处理的是哪级中断的情况。

⑷系统总线的连接方式

①缓冲方式

每片 8259A 都通过总线驱动器与系统数据总线相连，适用于多片 8259A 级联的大系统中。在缓冲方式下，8259A 的 $\overline{SP}/\overline{EN}$ 输出信号作为缓冲器的启动信号，用来启动总线驱动器，在 8259A 与 CPU 之间进行信息交换。

②非缓冲方式

每片 8259A 都直接和数据总线直接相连，适用于单片或片数不多的 8259A 组成的系统中，在这种方式下，8259A 的 $\overline{SP}/\overline{EN}$ 作为输入端设置，主片接高电平，从片接低电平。

⑸中断触发方式

①边沿触发方式

该方式以 IR 端上出现由低电平向高电平的跳变作为中断请求信号。

②电平触发方式

该方式以 IR 端上出现的高电平作为中断请求信号。

3. 8259A 的编程

8259A 的编程分为初始化命令编程和操作命令编程两种方式。初始化编程是根据工程实际需求，在 8259A 开始工作之前对 ICW_1～ICW_4 寄存器进行编程，用于建立 8259A 的基本工作方式。该编程一旦写入，一般在系统运行过程中不再改变。而操作命令编程是为了动态改变 8259A 对中断处理过程的控制，因此它可以在 8259A 工作之前写入，也可以在工作期间根据需要写入。8259A 的编程结构图如图 7.2 所示。

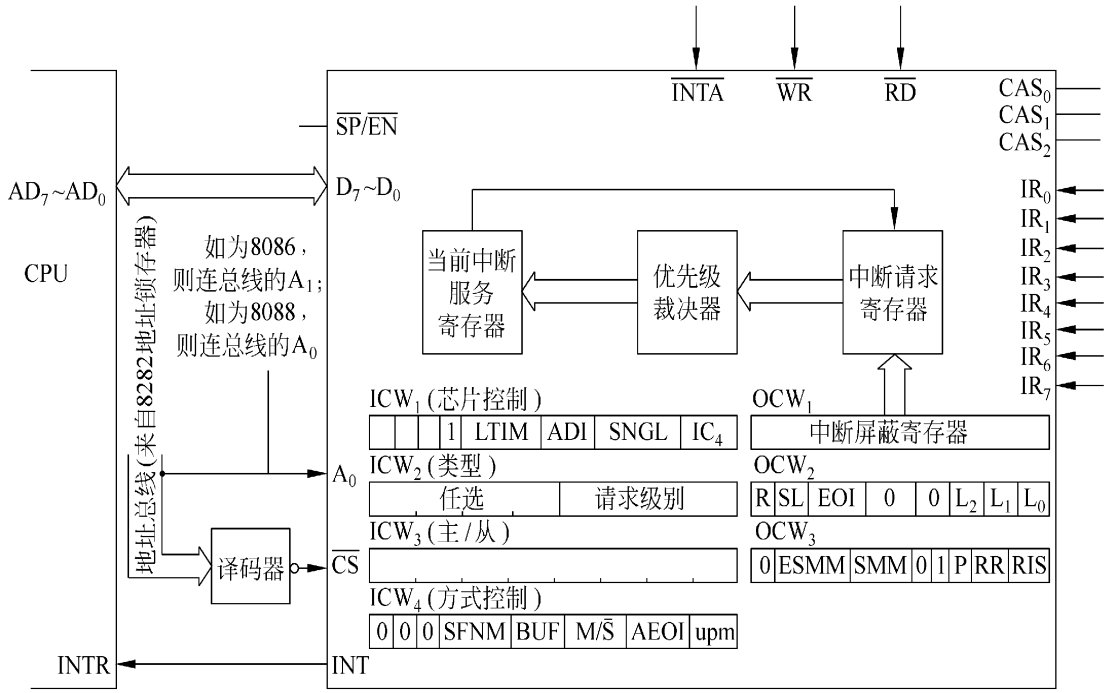


图 7.2 8259A 的编程结构图

⑴8259A 的初始化编程

8259A 的初始化编程是通过 CPU 向 8259A 写入四个初始化命令字来实现的，8259A 初始化编程流程如图 7.3 所示。

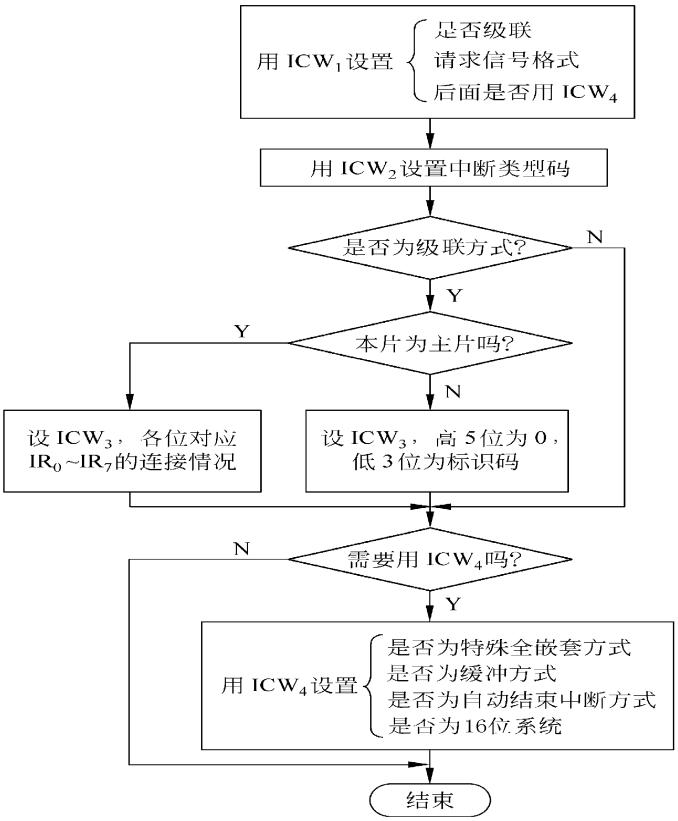
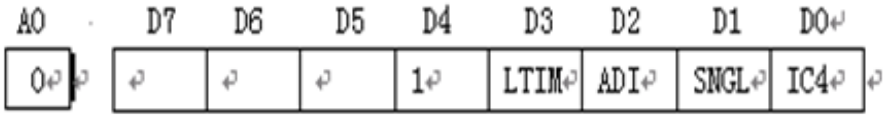


图 7.3 8259A 初始化流程

①ICW_1的格式和含义



D_4：ICW_1的标识位，区别与操作命令字 OCW_1和 OCW_2。

D_3：设定中断请求信号的形式。如 LTIM 为 0，则中断请求信号为边沿触发方式；如 LTIM 为 1，则中断请求信号为电平触发方式。

D_2：在 16 位和 32 位系统中不起作用，可为 0，也可为 1。

D_1：当系统中只有一片 8259A 时，此位为 1；当系统中有多片 8259A 时，则主片和从片的此位均为 0。

D_0：指出后面是否需要设置 ICW_4。

②ICW_2的格式和含义

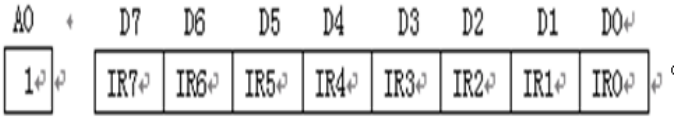
中断类型号的高 5 位就是 ICW_2的高 5 位，而低三位的值取决于引入中断的引脚序号。

ICW_2是任选的；

ICW_2高 5 位影响中断类型码，而中断类型码的低 3 位由 IR_0～IR_7决定

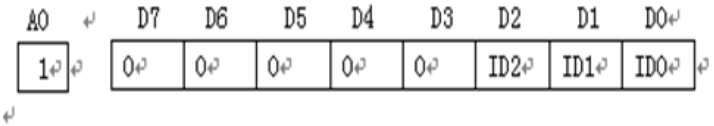
③ICW_3的格式和含义

如是主片,格式如下：



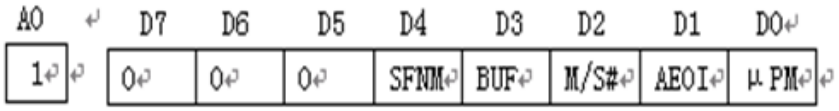
如某一引脚上连有从片，则对应位为 1，否则为 0。

如是从片，格式如下：



D2—D0 的值与从片的输出端 INT 到底连在主片的哪条中断请求输入引脚有关。

④ ICW4 的格式和含义



D_7—D_5：总是 0，是 ICW_4 的标识。

D_4：如为 1，则为特殊全嵌套方式。

D_3：如为 1，则为缓冲方式。

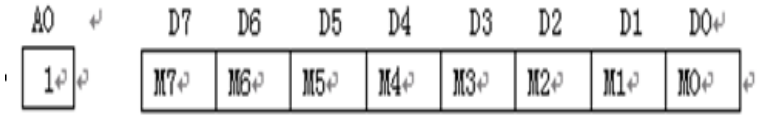
D_2：用来表示本片为主片还是从片。为 1 表示主片，为 0 表示从片。

D_1：如为 1，表示设置中断自动结束方式。当中断响应的第二个负脉冲到来时，当前中断服务寄存器 ISR 中的相应位会自动清除。

D_0：如为 1，表示当前 8259A 所在系统为非 8 位系统。

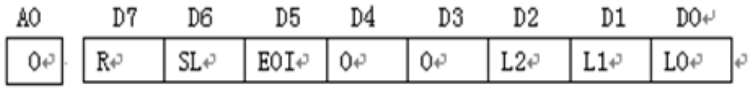
⑵8259A 的操作编程

①OCW_1 的格式和含义：



当 OCW_1 中某位为 1 时，对应于此位的中断请求受到屏蔽，否则表示允许。

②OCW2 的格式和含义：



D_7：决定系统的中断优先级是否按循环方式设置。如为 1，则采用优先级循环方式；如为 0，则为非循环方式。

D_6：决定 OCW_2 的 D_2、D_1、D_0 是否有效，如为 1，表示有效。

D_5：中断结束命令。当为 1 时，使当前中断服务寄存器中的对应位复位。

D_2、D_1、D_0：有两个用处。当 OCW_2 给出特殊的中断结束命令时，指出具体要清除当前中断服务寄存器的哪一位；当 OCW_2 给出特殊的优先级循环方式命令时，指出循环开始时哪个中断的优先级最低。

OCW_2 的功能：设置优先级循环方式；组成两类中断结束命令：一般的中断结束命令；特殊的中断结束命令。

③OCW_3 的格式

A0	D7	D6	D5	D4	D3	D2	D1	D0
0	0	ESMM	SMM	0	1	P	RR	RIS

ESMM：特殊的屏蔽模式允许位。

SMM：特殊屏蔽模式位。在这两位设置 1，可使 8259A 脱离当前的优先级方式，只要 CPU 内部的 IF 为 1，系统就可响应任何未被屏蔽的中断请求。

P：当 P=1 时，使 8259A 设置为中断查询方式。

当 RR=1，RIS=0，就构成了对 IRR 寄存器的读出命令。

当 RR=1，RIS=1，就构成了对 ISR 寄存器的读出命令。

功能：设置和撤销特殊屏蔽方式；设置中断查询方式；设置对内部寄存器的读出命令

二、定时器/计数器 8253

1.8253 的内部结构和引脚

⑴8253 的内部结构框图如图 7.4 所示

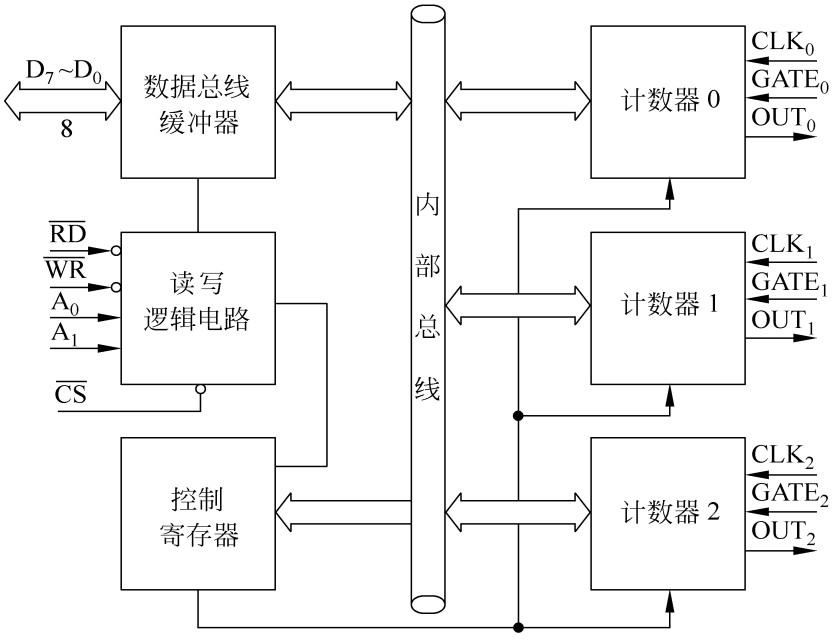


图 7.4 8253 的内部结构

①数据总线缓冲器

实现 8253 与 CPU 数据总线连接的 8 位双向三态缓冲器，用以传送 CPU 向 8253 的控制信息、数据信息以及 CPU 从 8253 读取的状态信息，包括某时刻的实时计

数值。

②读/写控制逻辑

控制 8253 的片选及对内部相关寄存器的读/写操作，它接收 CPU 发来的地址信号以实现片选、内部通道选择以及对读/写操作进行控制。

③控制字寄存器

在 8253 的初始化编程时，由 CPU 写入控制字，以决定通道的工作方式，此寄存器只能写入，不能读出。

④计数器 0、计数器 1、计数器 2

这是三个独立的、结构相同的计数器/定时器通道，每一个通道包含一个 16 位的计数寄存器，用以存放计数初始值，一个 16 位的减法计数器和一个 16 位的锁存器，锁存器在计数器工作的过程中，跟随计数值的变化，在接收到 CPU 发来的读计数值命令时，用以锁存计数值，供 CPU 读取，读取完毕之后，输出锁存器又跟随减 1 计数器变化。

⑵8253 的外部引脚

①D_7～D_0：双向、三态数据线引脚，与系统的数据线连接，传送控制、数据及状态信息。

②$\overline{RD}$：来自于 CPU 的读控制信号输入引脚，低电平有效。

③$\overline{WR}$：来自于 CPU 的写控制信号输入引脚，低电平有效。

④$\overline{CS}$：芯片选择信号输入引脚，低电平有效。

⑤A_1、A_0：地址信号输入引脚，用以选择 8253 芯片的计数器及控制字寄存器。8253 占用 4 个接口地址，地址由$\overline{CS}$、A_0、A_1确定。同时，再配合$\overline{RD}$、$\overline{WR}$控制信号，可以实现对 8253 的各种读/写操作。上述信号的组合功能由表 6.3 来说明。

⑥V_{CC}及 GND：+5V 电源及接地引脚

⑦$CLK_{0\sim2}$：每个计数器时钟输入引脚，计数器对此时钟信号进行计数。CLK 最高频率可达 5MH_Z。

⑧$GATE_{0\sim2}$：每个计数器的门控信号输入引脚，即计数器的控制输入信号，用来控制计数器的工作。

⑨$OUT_{0\sim2}$：计数器输出引脚，用来产生不同方式下的输出波。

2.8253 的编程

8253 的编程结构图如图 7.5 所示。8253 内部有三个计数器，分别为计数器 0、计数器 1、计数器 2，结构完全相同。每个计数器内部都有一个 16 位的初值寄存器 CR、计数执行部件 CE、输出锁存器 OL。三个计数器共用一个控制寄存器（8 位）。

执行部件 CE 实际上就是一个 16 位的减法计数器，它的起始值就是初值寄存器 CR 的值，而初值是通过程序设置的。输出锁存器 OL 用来锁存计数执行部件 CE 的内容，从而使 CPU 可以进行读操作。

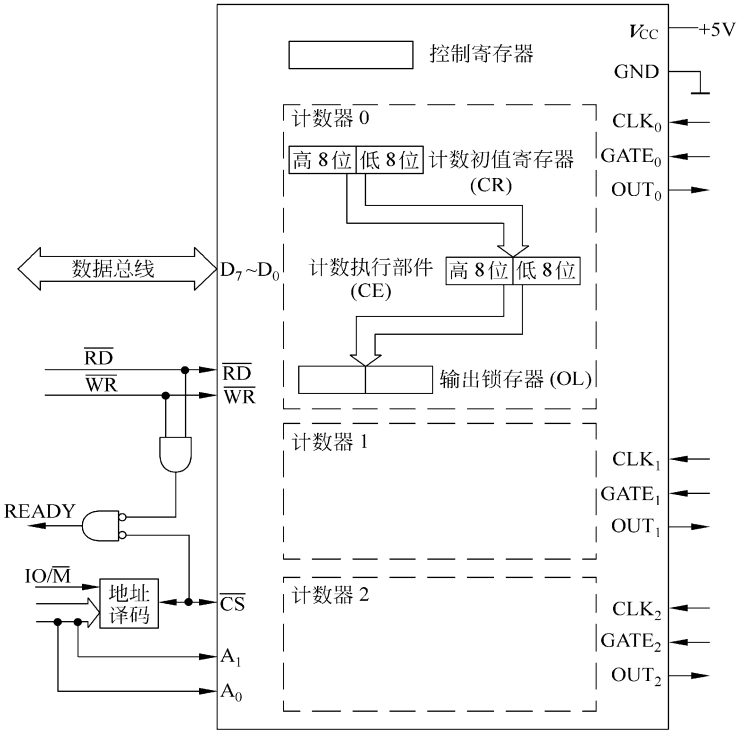


图 7.5 8253 的编程结构

工作原理：

计数执行部件 CE 从初值寄存器 CR 中获得计数初值，便进行减 1 计数。此时，输出锁存器 OL 随着执行部件 CE 的内容变化而变化。当有一个锁存命令道来时，锁存器便锁存当前计数，直到被读走之后，又开始跟随计数执行部件的动作。

⑴8253 的控制字

8253 有一个 8 位的控制字寄存器，其格式如下如图 7.6 所示。

D_0：数制选择控制。为 1 时，表明采用 BCD 码进行定时/计数；否则，采用二进制进行定时/计数。允许用户使用的二进制数为 0000H～FFFFH,十进制数为

0000～9999。

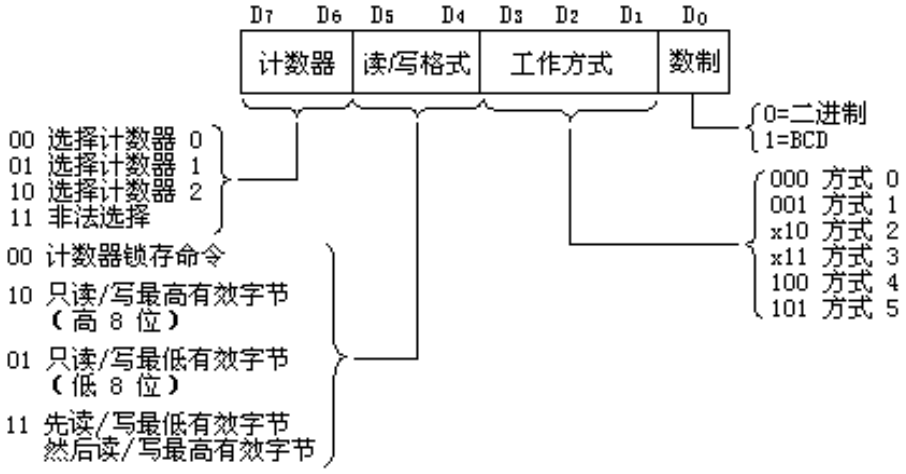


图 7.6 8253 控制寄存器格式

$D_3D_2D_1$：工作方式选择控制。000，方式 0；001，方式 1；X10，方式 2；X11，方式 3；100，方式 4；101，方式 5；

D_5、D_4：读写格式。00，计数锁存命令；01，读/写高 8 位命令；10，读/写低 8 位命令；11，先读/写低 8 位，再读写高 8 位命令。

D_7、D_6：通道选择控制。00　计数器 0；01，计数器 1；10，计数器 2

⑵8253 的初始化编程

要使用 8253，必须首先进行初始化编程，初始化编程包括设置控制字和送计数初值两个方面，控制字写入 8253 的控制字寄存器，而初始值则写入相应通道的计数寄存器中。

①先写入方式控制字，规定计数器的工作方式。

②再写入计数值，若方式控制字规定只写低 8 位，则只写低 8 位；若方式控制字规定只写高 8 位，则只写高 8 位；若为 16 位计数值则分两次写入，先写低 8 位，后写高 8 位。

例：设 8253 的端口地址为：04H～07H，要使计数器 1 工作在方式 0，仅用 8 位二进制计数，计数值为 128，进行初始化编程。

解：控制字为：01010000B=50H

初始化程序：

```
MOV   AL，50H
OUT   07H，AL
```

```
MOV  AL，80H
OUT   05H，AL
```

例：设 8253 的端口地址为：F8H～FBH，若用通道 0 工作在方式 1，按十进制计数，计数值为 5080H，进行初始化编程。

解：控制字为：00110011B=33H

初始化程序：

```
MOV  AL，33H
OUT  0FBH，AL
MOV  AL，80H
OUT  0F8H，AL
MOV  AL，50H
OUT  0F8H，AL
```

3．8253 工作方式

⑴模式 0——计数结束产生中断

当计数到达 0 时，输出端 OUT 为高电平，如图 7.7 所示。

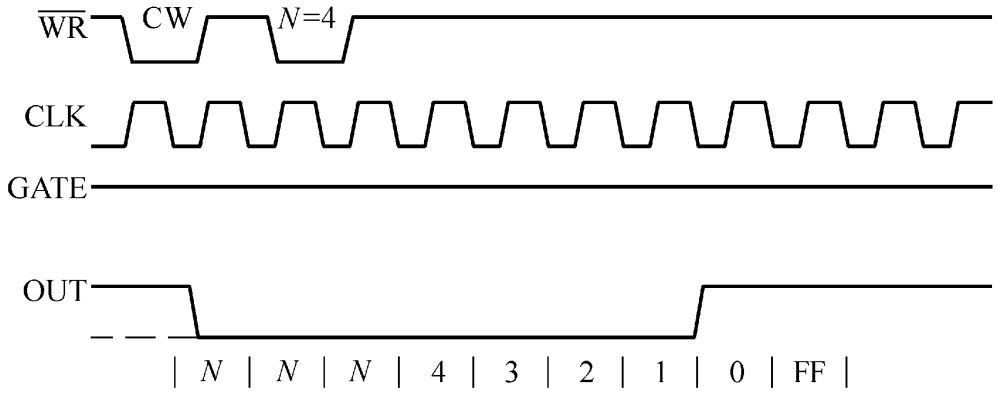


图 7.7 模式 0 时序图

⑵模式 2——分频器

正脉冲为 N-1 个时钟脉冲宽度，负脉冲为 1 个时钟脉冲宽度，如图 7.8 所示。

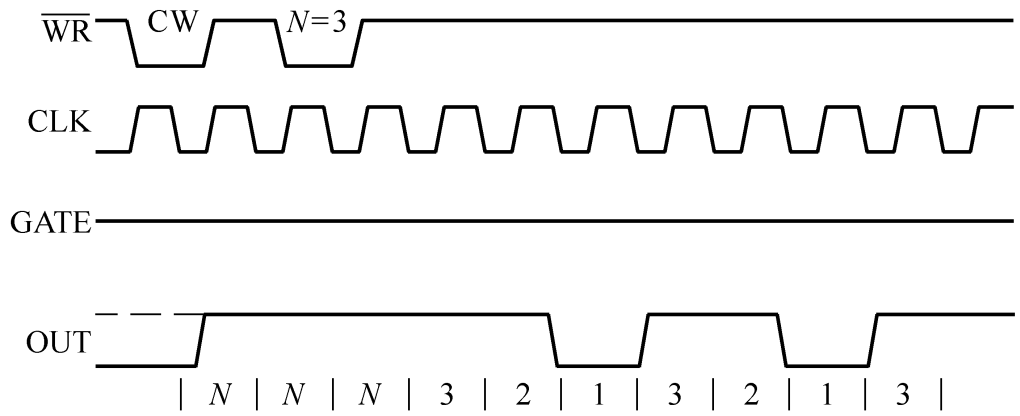


图 7.8 模式 2 时序图

⑷模式 3——方波发生器

和模式 2 类似，但输出为方波或基本对称的矩形波，如图 7.9 所示。

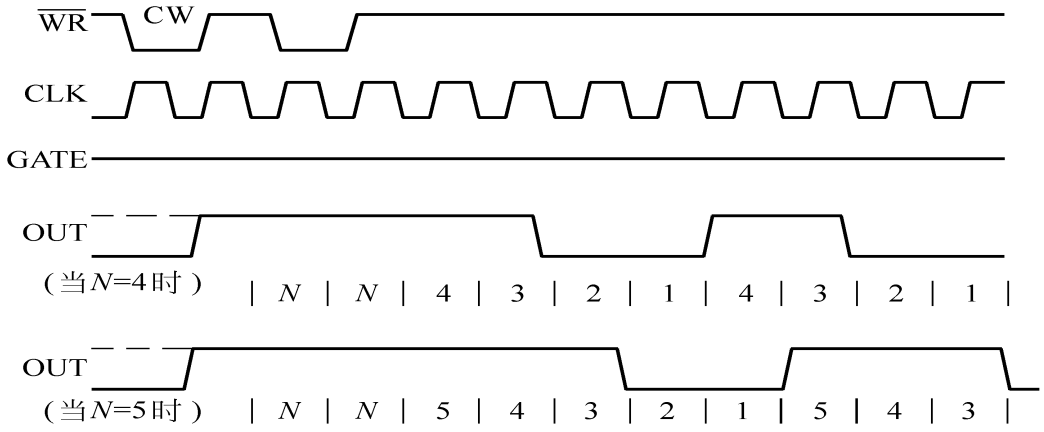


图 7.9 模式 3 时序图