实验指导书

周明德　主编

微型计算机硬件软件及其应用

清华大学出版社

封面设计：刘凤兰

京所通 64 — 10
书 号 15235·125
定 价 2.25 元

微型计算机

硬件软件及其应用实验指导书

周明德　主编

清华大学出版社

微型计算机

硬件软件及其应用实验指导书

周明德　主编

清华大学出版社

内 容 简 介

本实验指导书是与教材《微型计算机硬件软件及其应用》配套使用的。共分为三部分，第一部分是单板机编程实验，按照由浅入深的原则安排了九个实验。前几个结合实验介绍了单板机的使用和调试程序的方法，安排了数据块传送与搜索、字符变换、算术运算以及子程序调用等编程。打＊号的可作为选作。第二部分是微型计算机系统实验，介绍了如何在CP/M 操作系统支持下做实验。亦介绍了常用的编辑命令、汇编命令、连接命令，存盘和执行程序的方法。前几个是基本的编程实验，打＊号的编程的要求较高，可根据情况选作。第三部分是硬件和接口实验，前几个是基本逻辑电路以及Z80-CTC和Z80-PIO 的实验，打＊号的是一些应用性质的实验。

在附录中给出了主要实验的参考程序清单，这是一些典型的实用程序，可作为今后工作的参考。

本实验指导书内容丰富、系统，可作高校本科各专业学习微型计算机的实验指导书；也可作为非计算机专业的研究生学习微型计算机的实验指导书。亦可作为各种微型计算机技术培训班的实验指导书。

微型计算机硬件软件及其应用
实验指导书
周明德　主编
＊
清华大学出版社　出版
北京　　清华园
国防工业出版社印刷厂　印刷
新华书店北京发行所发行　　各地新华书店经售
＊
开本：787×1092 1/16　印张：14　字数：335千字
1984年10月第1版　　1985年10月第2次印刷
印数：300001～350000
统一书号：15235·125　　定价：2.25元

前　言

本实验指导书是与教材《微型计算机硬件软件及其应用》一书配套的。

计算机是一种实践性很强的学科，学习计算机必须理论与实践紧密结合。理论学习要与一定数量的习题练习相结合，更重要的是上机练习。凡是具有上机条件的必须尽量争取多上机多练习。如果能够创造条件，做到上机练习时间为理论学习时间的1/2，甚至能做到 1:1 或更多，那一定会大大提高学习的效果，真正能提高分析问题和解决问题的能力。

本实验指导书紧密结合教材的内容，安排了三部分实验。

第一部分是单板机编程实验。由于限于条件，不是所有的单位都能具备足够数量的微型计算机系统的；而且对于初学者说来，在单板机上做编程实验也简单、方便。这一部分我们按由浅入深循序渐进的原则安排了九个实验。打＊号的实验要求较高一些，可以作为学时较多的专业选用，或作为学习程度较高的同学选作。

第二部分是微型计算机系统实验。凡有条件的单位，应在单板机实验的基础上选作其中一部分。这样，有利于学生加深对微型计算机系统的了解；也可以初步掌握如何使用磁盘操作系统的命令和系统调用。在这一部分从机型上我们简要地介绍了一下目前国内较普遍的 Cromemco 系统和 Apple Ⅱ 系统，而主要是介绍如何使用CP/M(或其变型)操作系统，在 CP/M 操作系统的支持下做实验。此部分中打＊的几个实验编程要求较高，可以根据情况选作。

第三部分是硬件和接口实验，按照目前国内的实际情况，我们是以单板机加实验板来做这部分实验的。前面几个实验是硬件的基本逻辑电路实验和 CTC 及 PIO 的基本实验。对于加深教材中硬件和接口片子的了解是有好处的。打＊的实验是一些带应用性质的实验，也可以根据条件加以选择。

在附录中，我们给出了主要实验的参考程序清单。我们的目的，一方面是为学生自己编程序时作参考。我们坚决主张学生在做实验前，必须要充分预习，充分准备，要依靠自己在实验前编出程序，经过实验调试改正程序，得出正确的结果。这样做实验，才真正有收获，真正能加深对理论的理解，也才真正能提高分析问题和解决问题的能力。我们坚决反对学生事先没有准备，实验时照参考程序敲键的做法，这样时间化得再多，收效也是不大的。我们所以要提供参考程序，只是给学生自己编了程序以后作为对照用的，以提高学生编程序的能力。另一方面的目的也是给大家提供一些实用程序。这些程序是有一定的典型性和实用性的。可以作为将来工作中编制更复杂程序的基础。

附录中提供了两种调试程序的主要命令，可帮助我们在实验中调试程序。

附录中也收集了一些常用集成电路的引脚图，对于实验，对于将来的工作都有一定的参考价值。

本实验指导书内容较丰富、系统，可以作为大学本科各专业学习微型计算机的实验指导书，也可以作为非计算机专业的研究生学习微型计算机的实验指导书。也可作为各

种类型的微型计算机培训班的实验指导书。

本实验指导书的第一部分内容是张淑玲同志编写的；第三部分中的电子秒表实验和交通信号灯实验是白晓笛同志准备的；A/D 实验和 A/D 与D/A正逆变换实验是田开亮同志准备的；8251A串行通讯实验是付强同志准备的。全书由周明德同志主编。由于编者的水平有限，实践经验还不够丰富，缺点错误在所难免。敬请读者批评指正。

周明德

1984.2

目　录

第一部分　单板机实验

第一节　TP801-A单板计算机的结构和原理

TP801-A 是以Z80-CPU为核心的单板微型计算机，其结构如图 1.1 所示。

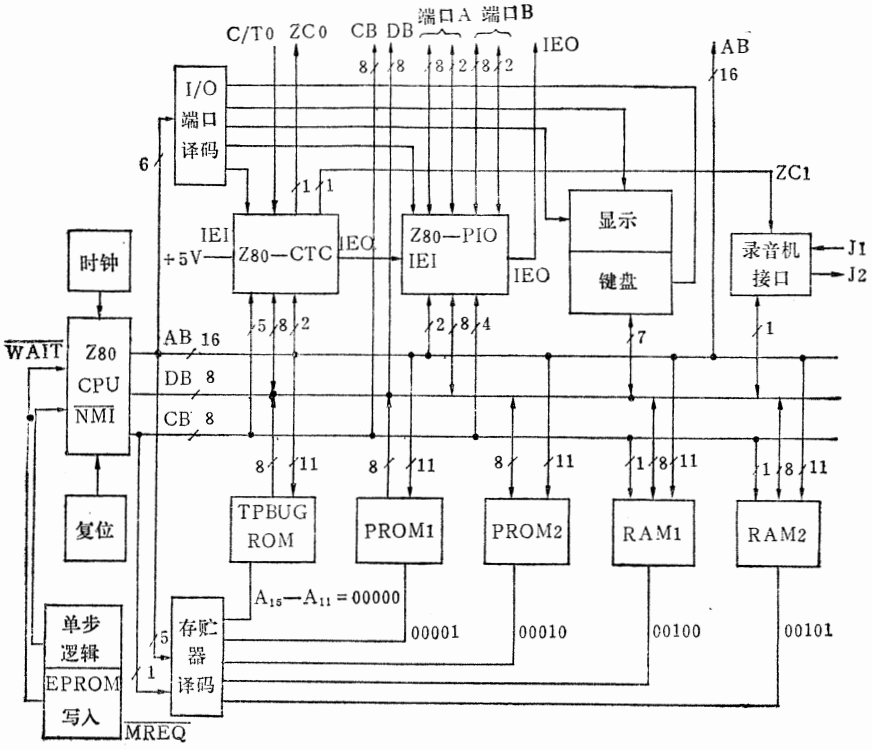


图1.1　TP801-A单板计算机原理框图

它以Z80-CPU 为微处理器，时钟频率为 2MHz。单板机上没有汇编程序，以机器语言进行操作，凡要在TP801 上运行的用户程序，必须经过手工汇编，把它们变为机器码才能在单板机上运行。

TP801 的存贮器分成两大部分：ROM 和 RAM。

ROM 又分为：ROM(其中驻有 TPBUG-A)，PROM1 和 PROM2。每一种都为 2K×8Bit 的片子，构成 2KROM。其中，ROM中存放系统的监控和调试程序TPBUG-A，它用于管理 TP801 的工作，也可用于调试用户的程序。

PROM1 中可以存放固化的用户程序。若TP801 用于一个固定的工作对象，则可把用户程序经过调试后就固化在 PROM1 中。若单板机的开关 S_2 合向下边 (详见后面介绍)，则当系统复位(上电即自动复位，或因按 RESET 按钮引起的复位)后，在经过适当初始化后，就可直接进入和执行固化在PROM1 中的用户程序，而不进入TPBUG-A。

PROM2 是一个 EPROM 片子，典型的为 Intel 2716。在TP801上，可以把在RAM中并经过调试的用户程序，写入到位于 PROM2 上的 EPROM 中。然后可以将此EPROM 片子插至 PROM1 位置，令单板机执行固化在其中的用户程序。

RAM又可分为RAM1和RAM2。由于上述的三种 ROM片子都是 2K × 8 Bit的，所以，RAM也以 2K字节作为一组，分别叫做 RAM1 和RAM2。TP801-A单板计算机的RAM 容量基本配置是 4K字节，但尚可以按需要扩展，再扩展 4K字节是比较方便的。

RAM可用作用户的程序区和数据区，以及系统和用户的堆栈。RAM中有一部分空间用作系统的数据区。

Z80-CPU 有 16 条地址线，可寻址 64K 存贮空间。而在 TP801-A 中的存贮器，ROM占 6K 存贮空间，RAM 为 4-8K。那么，这些ROM和RAM的地址是如何分配的呢？RAM 中的用户区是在什么地址范围内呢？这就必须分析一下存贮器的地址连线。TP801-A的存贮器结构如图 1.2 所示。

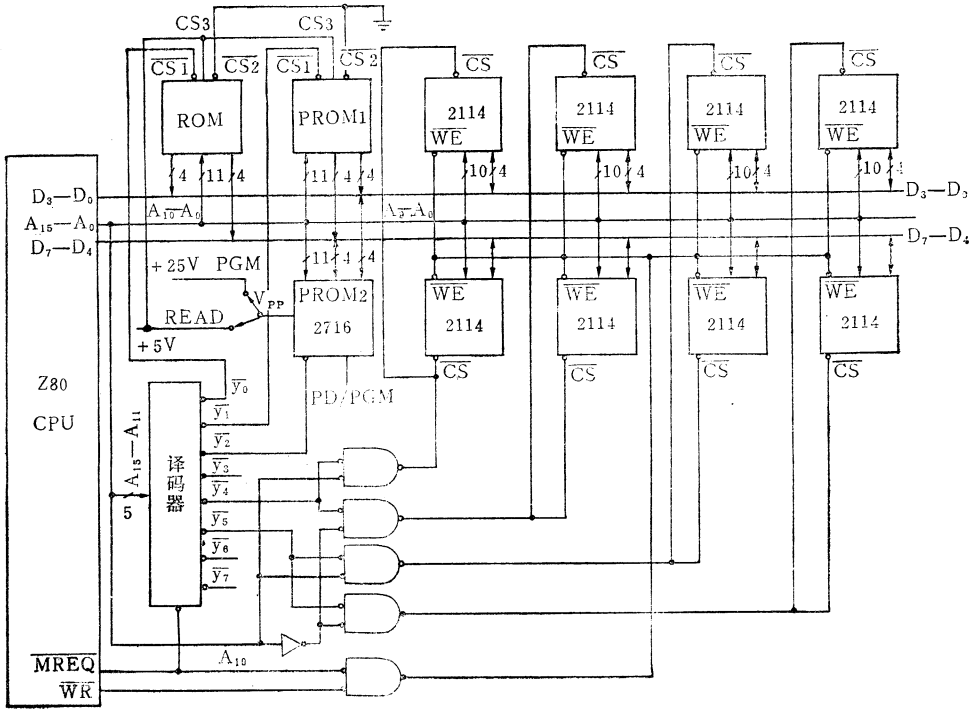
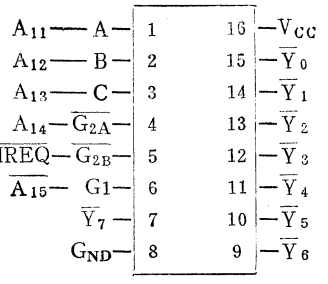


图1.2　TP801-A存贮器结构

ROM、PROM1和PROM2 都是2K × 8 Bit的片子，每个片子上有 11 条地址线，可直接区分片内的2^{11}=2K地址单元，它们可直接连至Z80-CPU地址总线的A_0-A_{10}。ROM和PROM1 的选片端都有 3 个，其中两个中一个为低电平有效的那端直接接地，而另一个高电平有效的那端直接接至＋ 5 V；另一个选片端连至高位地址线的译码输出。

CPU 的高位地址线 A_{15}-A_{11}，连至译码器，如图 1.3 所示。

$\overline{A_{15}}$ 连至选片端 G_1，而 A_{14} 连至选片端$\overline{G2A}$。故要选中此译码器片子，必须 A_{15}= A_{14}= 0。由于 A_{13}、A_{12}、A_{11} 的不同组合，译码器的输出有效的情况如图 1.3 中的真



G_1	$\overline{G_{2A}}$	$\overline{G_{2B}}$	C	B	A	输出
1	0	0	0	0	0	$\overline{Y}_0=0$ 其余为1
1	0	0	0	0	1	$\overline{Y}_1=0$ 其余为1
1	0	0	0	1	0	$\overline{Y}_2=0$ 其余为1
1	0	0	0	1	1	$\overline{Y}_3=0$ 其余为1
1	0	0	1	0	0	$\overline{Y}_4=0$ 其余为1
1	0	0	1	0	1	$\overline{Y}_5=0$ 其余为1
1	0	0	1	1	0	$\overline{Y}_6=0$ 其余为1
1	0	0	1	1	1	$\overline{Y}_7=0$ 其余为1
不是上述情况			×	×	×	输出全为1

图1.3 RAM的译码电路

值表所示。

由此可确定各个ROM的地址范围：

ROM：	$A_{15}A_{14}A_{13}A_{12}A_{11}$	$A_{10}——A_0$	
最低地址	0 0 0 0，0	000，0000，0000	0000H
最高地址	0 0 0 0，0	111，1111，1111	07FFH
PROM1：	$A_{15}A_{14}A_{13}A_{12}A_{11}$	$A_{10}——A_0$	
最低地址	0 0 0 0，1	000，0000，0000	0800H
最高地址	0 0 0 0，1	111，1111，1111	0FFFH
PROM2：	$A_{15}A_{14}A_{13}A_{12}A_{11}$	$A_{10}——A_0$	
最低地址	0 0 0 1，0	000，0000，0000	1000H
最高地址	0 0 0 1，0	111，1111，1111	17FFH

RAM1和RAM2都是由2114片子组成。2114是1K×4Bit的片子，两片构成1K字节。故，RAM1和RAM2各由4片2114组成。每一个片子上有10条地址线，它们直接连至地址总线的A_0-A_9，可以区分1K地址。RAM1和RAM2由译码器的输出线$\overline{Y}_4$和$\overline{Y}_5$来区分。而RAM1和RAM2的两个1K字节就要由地址线A_{10}来区分。故，$\overline{Y}_4$与A_{10}选择RAM1中的2K地址，$\overline{Y}_5$与A_{10}选择RAM2中的2K地址。由此可以确定RAM的地址范围。

RAM1：				
前1K地址	$A_{15}A_{14}A_{13}A_{12}A_{11}$	A_{10}	$A_9——A_0$	
最低地址	0 0 1 0，0	0	00，0000，0000	2000H
最高地址	0 0 1 0，0	0	11，1111，1111	23FFH
后1K地址	$A_{15}A_{14}A_{13}A_{12}A_{11}$	A_{10}	$A_9——A_0$	
最低地址	0 0 1 0，0	1	00，0000，0000	2400H
最高地址	0 0 1 0，0	1	11，1111，1111	27FFH
RAM2：	$A_{15}A_{14}A_{13}A_{12}A_{11}$	A_{10}	$A_9——A_0$	
前1K地址				
最低地址	0 0 1 0，1	0	00，0000，0000	2800H
最高地址	0 0 1 0，1	0	11，1111，1111	2BFFH
后1K地址				

最低地址　　0 0 1 0，1　　1　　00，0000，0000　　2C00H

最高地址　　0 0 1 0，1　　1　　11，1111，1111　　2FFFH

综合以上讨论，TP801-A中存贮器的地址分配如表 1-1 所示。

2114 片子与 2MHz 的 Z80-CPU 相连接，则即使在取指周期 2114 也能满足CPU的要求，不需要在 CPU 的基本时序中插入T_W状态。

TP801-A的监控程序需要利用 RAM 的一些空间作为它的数据区和堆栈。故RAM区域的分配如表 1-2 所示。

表1-1

地　址	器　件	A_{15}—A_{11}	A_{10}—A_0	译码器的有效输出
0000—07FFH	2K ROM	0 0 0 0 0	可变	$\overline{Y}_0$ = $\overline{CS0}$ = MON SEL
0800—0FFFH	2K PROM1	0 0 0 0 1	可变	$\overline{Y}_1$ = $\overline{CS1}$ = PROM1 SEL
1000—17FFH	2K PROM2	0 0 0 1 0	可变	$\overline{Y}_2$ = $\overline{CS2}$ = PROM2 SEL
1800—1FFFH	没用	0 0 0 1 1	可变	$\overline{Y}_3$ = $\overline{CS3}$　没用
2000—27FFH	2K RAM1	0 0 1 0 0	可变	$\overline{Y}_4$ = $\overline{CS4}$ = RAM1 SEL
2800—2FFFH	2K RAM2	0 0 1 0 1	可变	$\overline{Y}_5$ = $\overline{CS5}$ = RAM2 SEL
3000—37FFH	备用	0 0 1 1 0	可变	$\overline{Y}_6$ = $\overline{CS6}$　没用
3800—3FFFH	备用	0 0 1 1 1	可变	$\overline{Y}_7$ = $\overline{CS7}$　没用

表　1-2

地　址　空　间	用　途	字　节　数
2000—23FFH	RAM1的用户区	1K
2400—27FFH	RAM1的用户区	1K
2800—2BFFH	RAM2的用户区	1K
2C00—2F87H	RAM2的用户区	904
2F88—2F9FH	监控程序堆栈工作区	24
2FA0—2FB7H	用户程序寄存器存放区，用户程序堆栈区	24
2FB8—2FBFH	TP801-A 4个用户程序的入口地址	8
2FC0—2FFFH	TPBUG-A使用的RAM暂存区和断点表	64

TP801-A的 I/O设备及接口电路如图1.4所示。

接有一个Z80-CTC片子，其中通道 1 、 2 和 3 用于监控程序，通道 0 留给用户用。用户可根据需要把它编程为定时器或计数器用。

有一个 PIO 片子，TP801-A没有使用，故它的两个通道都可留给用户用。

TP801-A的典型输入是键盘，有 16 个 16 进制数字键和 12 个功能键，如图 1.5 所示。

这些键的功能和使用方法在后面介绍。

TP801-A 的典型输出设备是 6 个 7 段显示管，可以显示机器码。通常左面 4 个显示管显示内存单元的地址或 CPU 内部寄存器号，或端口地址；而右面两个，显示该单元或寄存器或外设端口的内容。

此外，TP801-A 还可以与盒式磁带机相连。可以把内存中的内容转贮到磁带上，或

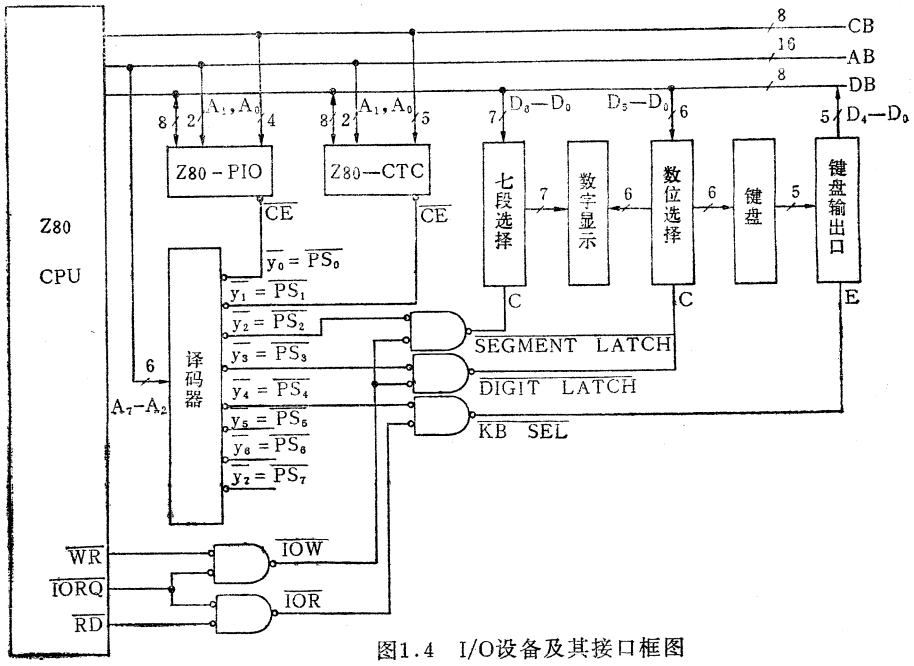


图1.4 I/O设备及其接口框图

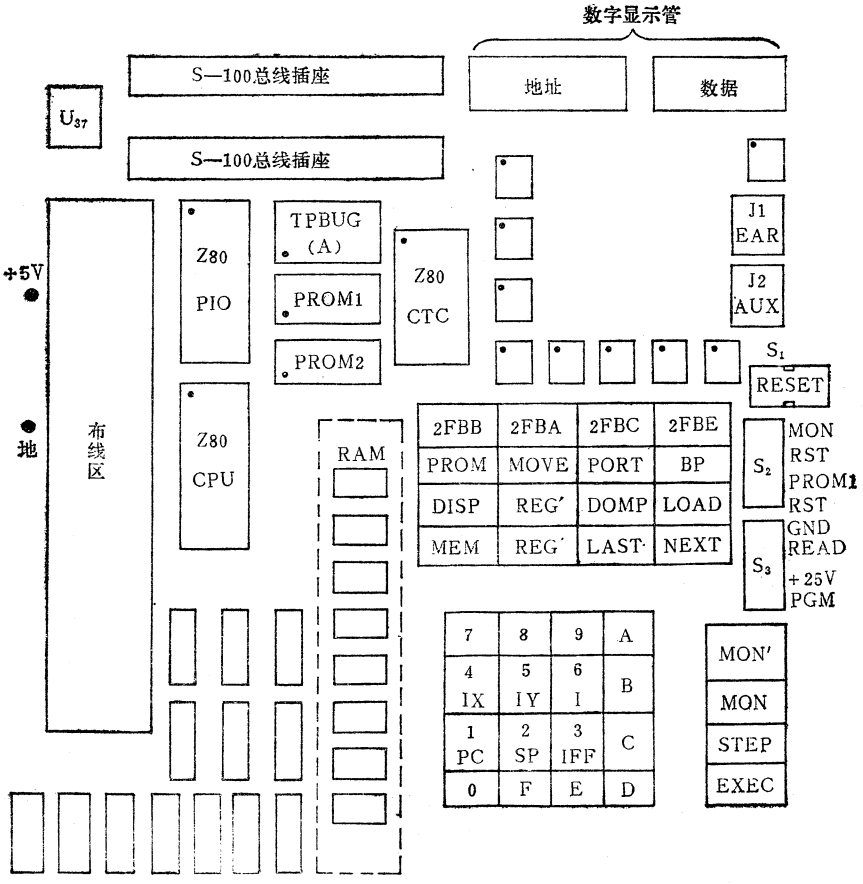


图1.5 TP801-A面板布置图

把磁带上记录的信息输入内存中。

所有上述这些I/O设备和接口电路，都要由地址总线的低8位来加以区别和选择。

Z80-PIO以及Z80-CTC，都要用地址总线的最低两位A_1和A_0来选择片内的不同端口。所以，要由地址总线的高6位——A_7-A_2经过译码器后作为选片信号。于是TP801-A的I/O设备和接口电路的端口地址如表1-3所示。

TP801-A是一个单板计算机，但任何一个计算机都必须在一个系统程序的管理下才能正常工作，在单板机中此程序固化在ROM中，称为监控程序，在TP801-A中即为固化在ROM中的TPBUG-A。

表 1-3

A_7—A_2	译码器输出	器件	A_1A_0	端口	端口地址
1 0 0 0 0 0	$\overline{Y}_0 = \overline{PSO} = \overline{PIO\ SEL}$	Z80—PIO	0 0	端口A数据	80H
			0 1	端口B数据	81H
			1 0	端口A控制寄存器	82H
			1 1	端口B控制寄存器	83H
1 0 0 0 0 1	$\overline{Y}_1 = \overline{PS1} = \overline{CTC\ SEL}$	Z80—CTC	0 0	通道(Ch)0	84H
			0 1	通道1	85H
			1 0	通道2	86H
			1 1	通道3	87H
1 0 0 0 1 0	$\overline{Y}_2 = \overline{PS2} = \overline{SEGLH}$	74LS273（八锁存器）	× ×	七段选择（只写）	88—8BH
1 0 0 0 1 1	$\overline{Y}_3 = \overline{PS3} = \overline{DIGLH}$	74LS273	× ×	数位选择（只写）	8C—8FH
1 0 0 1 0 0	$\overline{Y}_4 = \overline{PS4} = \overline{KB\ SEL}$	74LS244（八缓冲器）	× ×	读键值（只读）	90—93H
1 0 0 1 0 1	$\overline{Y}_5 = \overline{PS5}$	没使用			94—97H
1 0 0 1 1 0	$\overline{Y}_6 = \overline{PS6}$	没使用			98—9BH
1 0 0 1 1 1	$\overline{Y}_7 = \overline{PS7}$	没使用			9C—9FH

第二节 TPBUG-A介绍

TPBUG-A是一个2K字节的监控和调试程序，它主要是由以下几部分程序组成。

1. 初始化程序；
2. 键盘输入及处理程序；
3. 显示程序；
4. 各个功能键的处理程序；
5. 公用子程序。

这些程序在教材的第十章中做了较详细的分析。

为了实验中和以后工作中的需要，我们在这儿详细介绍一下显示程序与键盘输入程序。

一、显示程序

(一) 数码显示管

TP801-A 中所用的是 5V 电源的七段数字显示管，每一段为一发光二极管，其结构及原理如图 1.6 所示。

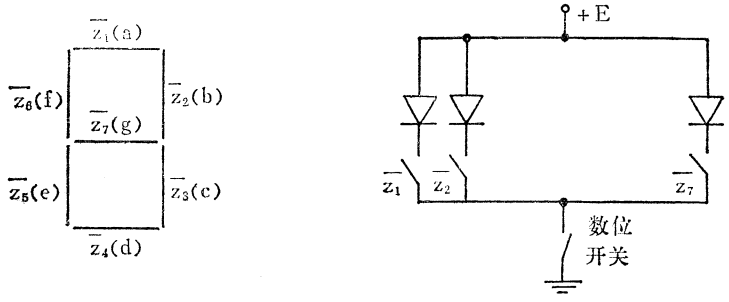


图1.6　显示管原理图

七段，共有七个段控制信号，以控制所显示的字形。当某一段的控制信号为低电平时，则相当于相应的触点闭合，这一段发光；整个管还有一个数位控制信号，也是低电平闭合，只有此触点闭合时，数字管才能发光。

七段用七位数即用 CPU 输出的D_6-D_0控制，D_7始终为 0，因而 16 进制数的显示编码如表 1-4 所示。

TP801-A 中用 6 个这样的数字管，可显示 6 个 16 进制数，其硬件电路如图 1.7 所示。

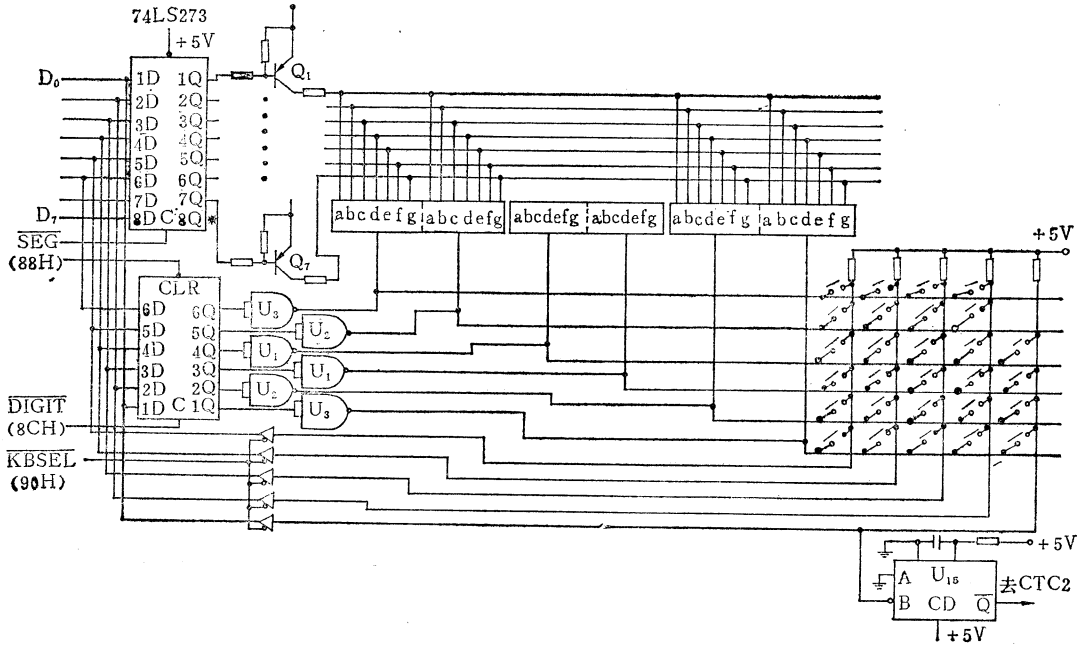


图1.7　键盘输入和显示硬件电路图

表 1-4

数　符	Z_8	Z_7	Z_6	Z_5	Z_4	Z_3	Z_2	Z_1	16进制字模编码
0	0	1	0	0	0	0	0	0	40H
1	0	1	1	1	1	0	0	1	79H
2	0	0	1	0	0	1	0	0	24H
3	0	0	1	1	0	0	0	0	30H
4	0	0	0	1	1	0	0	1	19H
5	0	0	0	1	0	0	1	0	12H
6	0	0	0	0	0	0	1	0	02H
7	0	1	1	1	1	0	0	0	78H
8	0	0	0	0	0	0	0	0	00H
9	0	0	0	1	1	0	0	0	18H
A	0	0	0	0	1	0	0	0	08H
b	0	0	0	0	0	0	1	1	03H
C	0	1	0	0	0	1	1	0	46H
d	0	0	1	0	0	0	0	1	21H
E	0	0	0	0	0	1	1	0	06H
F	0	0	0	0	1	1	1	0	0EH

要显示的字符的编码由数据总线送至锁存器 $\overline{SEG}$（端口地址88H）锁存，后经三极管 Q_1-Q_7，供给大的驱动电流，连到所有数字管相应的段。由于要显示的字模编码是同时连到所有显示管的，所以要使某一个显示管显示出数码，CPU还必须输出一个数位控制字（只有D_0-D_5有用），它由DIGIT锁存器锁存，以选择哪一个显示管工作。所以，要显示一个数码，首先要把这个数码转换成相应的字模编码，再输出给端口88H($\overline{SEG}$)；然后把控制哪一个数字管亮的数位控制字输出给端口 8CH($\overline{DIGIT}$)。

（二）显示程序

要显示的数放在以 DISMEM(在 TP801-A中为 2FF7H）单元开始的缓冲区中，总共为 6 个单元，由于每个数字显示管只能显示一位16进制数，故这 6 个存贮单元的高 4 位全为 0，低 4 位为要显示的数的二进制数码。故若要显示一个存贮单元或某一个寄存

器的内容，则必须通过一个子程序把这一字节的高 4 位和低 4 位拆开，放至显示缓冲区的两个单元中。

要显示某一个16进制数，首先要把这个数转换为显示管段形的字模编码，这个转换是由软件通过搜索一个表格（字模表）来实现的。这个表格以 16 进制数的次序，把它们相应的字模编码顺序存放，我们把表格的起始地址送 IX，以要显示的数作为偏移量，由 IX 加上偏移量找到字模编码的地址，则从表格中取出来的即为相应的数的字模编码。

显示程序的流程图如图 1.8 所示。

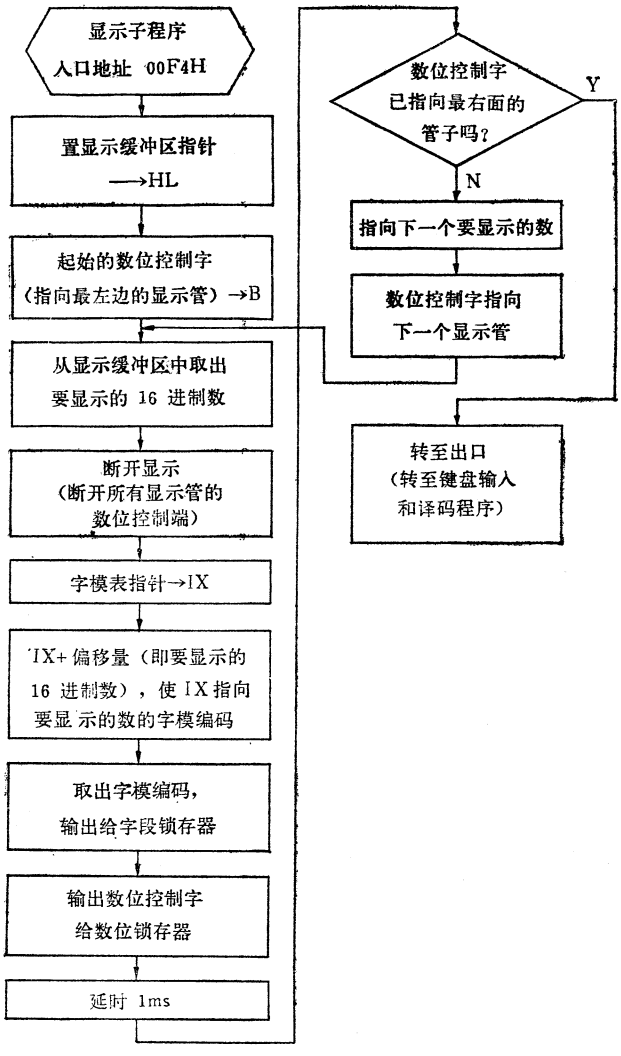


图1.8　显示程序流程图

显示程序为：

```
;          显示程序
; 功能：把数据从 DISMEM 开始的缓冲区送显示管。
```

```
; 输入：要显示的数放在以 DISMEM 开始的 6 个单元中。
; 输出：把预置的 6 个数输出以更新原有的显示。
; 所用的寄存器：HL 是指向 DISMEM 的指针
;               IX 是指向字模表的指针
;               B 是当前数位指示器
;               E保留要被显示的数
;               A和D是暂存寄存器
        ORG     00F4H
DISUP:  LD      HL, DISMEM      ; 指向显示缓冲区的起点
        LD      B, 20H          ; 指向最左边的显示管
DISUP1: LD      E, (HL)         ; 从显示缓冲区取数
        LD      D, 0
        LD      A, 0
        OUT     (DIGLH), A      ; 断开显示
        LD      IX, SEGPT       ; IX 指向字模表起点
        ADD     IX, DE          ; 指向要显示的数的字模
        LD      A, (IX+0)       ; 取出字模编码→A
        OUT     (SEGLH), A      ; 输出至段形锁存器
        LD      A, B
        OUT     (DIGLH), A      ; 输出至数位锁存器（控制是第几个
                                ; 数字显示管亮）
        LD      E, 45           ; 置 1ms 延时
DISUP2: DEC     E
        LD      A, 0
        CP      E
        JR      NZ, DISUP2
        LD      A, 01H
        CP      B               ; B＝1？（即是否已移至最右面的显
                                ; 示管）
        JR      Z, DISUP3       ; 是，转出口
        INC     HL              ; 指向下一个要显示数
        SRL     B               ; 移至下一个显示管
        JR      DISUP1          ; 循环
DISUP3: JP      DECKY           ; 转至键盘输入程序
DIGLH:  EQU     8CH
SEGLH:  EQU     88H
        ORG     07A6H
SEGPT:  DB      40H             ; 0
        DB      79H             ; 1
```

```
DB      24H             ; 2
DB      30H             ; 3
DB      19H             ; 4
DB      12H             ; 5
DB      02H             ; 6
DB      78H             ; 7
DB      00H             ; 8
DB      18H             ; 9
DB      08H             ; A
DB      03H             ; B
DB      46H             ; C
DB      21H             ; D
DB      06H             ; E
DB      0EH             ; F
DB      7FH             ; 空白
DB      0CH             ; 响应符P
DB      5FH             ; 撇号′,′
```

二、键盘输入程序

在TP801-A中是用16个数字键和12个命令键输入的，但键盘上方的8个命令键又可分为上、下两档，每一个键可以有两种命令。每一个键相当于一个按钮，把所按的键转换成相应的数或命令是通过键盘输入与译码程序实现的。

(一) 键盘输入硬件电路

键盘输入和显示的硬件电路见图1.7。

数据总线的信号D_0-D_5经反相后输出作为行线（键盘的输入信号线），它由端口$\overline{DIGIT}$（地址为8CH）控制；5条列线由＋5 V电源经10K电阻引出（键盘的输出信号线），通过三态缓冲器引至数据总线的D_4-D_0，由端口$\overline{KBSEL}$（地址为90H）控制。

在每一行与每一列之间接有一个键。

在工作时，程序按行扫描键盘，检查列的输出，由行信号与列信号的组合以确定是哪一个键闭合。即先让D_0=1(反相后为低电平)，D_1-D_5=0，此数据由端口8CH输出，即使键盘的最下面一行为低电平，若在此行中有键闭合，则相应的列输出为0(低电平)，而其他列为1。若这一行没有键闭合（即所有的列全为高电平），则再使上面一行为低电平，再检查列的输出，有键闭合的列输出为0，否则为1……，这样一行一行地扫描，由行的值与列的输出的配合，即可确定是哪一个键闭合。

例如，数码0闭合，则其对应的行值为01H，而列的输出即为0FH；数码1则为：行=02H，列=0FH；数码9则为：行=04H，列=1BH……如表1-5所示。

键盘输入程序首先检查是否有任一键闭合，若无任一键闭合则转至显示程序。当发现有某一键闭合时，则按行扫描键盘，每扫一行输入列值检查此行中是否有键闭合，若有键闭合，则转至键盘译码程序以确定此键值；若无键闭合，则扫描另一行……。程序

表 1-5

表格中的编码	键值	行值	列值
0FFH	0	B = 01	A = 0F
0EFH	1	B = 02	A = 0F
0F7H	2	B = 02	A = 17
0FBH	3	B = 02	A = 1B
0DFH	4	B = 04	A = 0F
0E7H	5	B = 04	A = 17
0EBH	6	B = 04	A = 1B
0CFH	7	B = 08	A = 0F
0D7H	8	B = 08	A = 17
0DBH	9	B = 08	A = 1B
0DDH	A	B = 08	A = 1D
0EDH	B	B = 04	A = 1D
0FDH	C	B = 02	A = 1D
0DH	D	B = 01	A = 1D
0BH	E	B = 01	A = 1B
07H	F	B = 01	A = 17
0EH	EXEC	B = 01	A = 1E
0FEH	SS	B = 02	A = 1E
0EEH	MON	B = 04	A = 1E
0DEH	MON′	B = 08	A = 1E
0CDH	NEXT LOAD	B = 10	A = 1D
0CBH	LAST DUMP	B = 10	A = 1B
0C7H	REG REG′	B = 10	A = 17
0BFH	MEM DISP	B = 10	A = 0F
0BDH	BP 2FBE	B = 20	A = 1D
0BBH	PORT 2FBC	B = 20	A = 1B
0B7H	MOVE 2FBA	B = 20	A = 17
0AFH	PROG 2FB8	B = 20	A = 0F

流程图如图 1.9 所示。

其程序为：

```
;              键盘译码和执行程序段
; 功能：熄灭数据显示，扫描键盘矩阵，搜索
; 有无键闭合，若有则插入 20ms 延时作为DEBOUNCE(消除键颤动的影响)
; 随后一次扫描一行，检查所有的列，找到闭
; 合的键值。若是数字键，则放到DISMEM中的下
; 一个空的单元（指针在KEYPTR中），若已输入两
; 个数置标志 DIG2，已输入 4 个数置标志 DIG4
; 当已输入了八个数，就调用修改内容子程序。
; 若输入的为命令键，就从转移表中（JPTAB）找
; 到相应命令的处理程序的入口。命令键的执
```

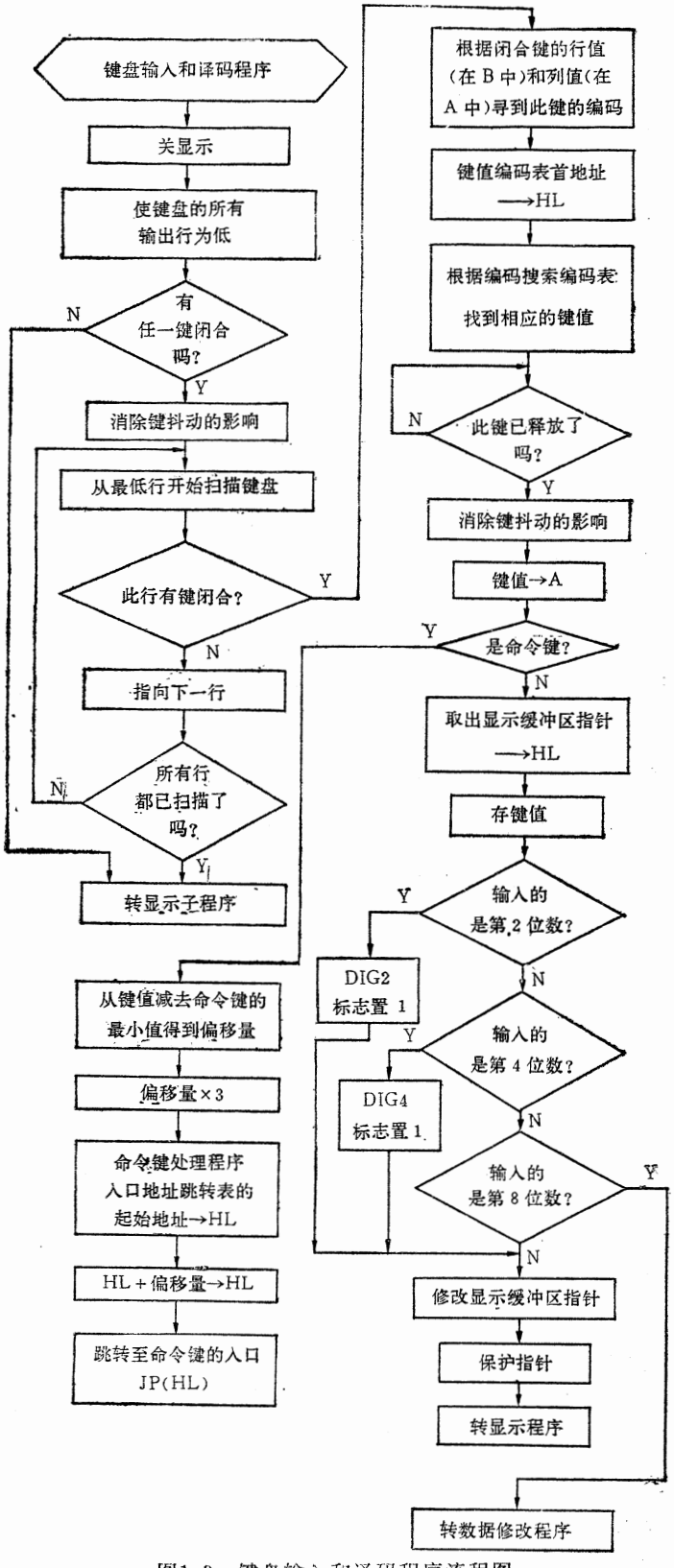


图1.9 键盘输入和译码程序流程图

```
;  行程序也包含在这个模块中。
;  所用的寄存器：
;          A——放键盘值；
;          B——扫描的行值；
;          C——暂存由行和列决定的编码；
;         HL——键值查阅表 KYTBL 指针；
;         HL——存贮器显示缓冲区指针。
                ORG     0123H
DECKY:          LD      A，7FH          ；断开显示
                OUT     (SEGLH)，A
                LD      A，3FH
                OUT     (DIGLH)，A      ；输出使所有行为低
                IN      A，(KBSEL)      ；输入列数据
                AND     1FH             ；屏蔽掉无用位
                CP      1FH             ；有任一键闭合吗？
                JP      Z，DISUP        ；无键闭合，转至显示程序
                CALL    D20MS           ；有键闭合，调 20ms延时，作
                                        ；为 DEBOUNCE
                LD      C，DIGLH        ；端口地址→C
                LD      B，01H          ；置使第一行为低的值
KEYDN1:         OUT     (C)，B          ；使所选择的行为低
                IN      A，(KBSEL)      ；输入列数据
                AND     1FH
                CP      1FH             ；有键闭合吗？
                JR      NZ，KEYDN2      ；有，转至键译码程序
                SLA     B               ；无，选择下一行
                LD      A，40H
                CP      B               ；所有行都扫描了吗？
                JR      NZ，KEYDN1      ；未完，循环
                JP      DISUP           ；已完，转至显示程序
```

（二）键盘译码程序

如上所述，由行值与列值的结合可找到所按的键，但要找到相应的键值就要从表格中去寻找，为了便于寻找，先要由行值和列值转换为另一个表格内的编码，这可以通过以下的程序段来实现：

```
;          键盘译码程序
;    B中为行值，A中为列值。
KEYDN2:  LD     C，00H
KEYDN3:  DEC    C
         SRL    B
```

```
        JR      NZ, KEYDN3    ; 若B未全移为0，则使C减1，直至B = 0
        SLA     C
        SLA     C
        SLA     C
        SLA     C
        ADD     A, C          ; 与A中的内容相结合，形成键值在
                              ; 表格中对应的编码
        LD      HL, KYTBL     ; 置键值转换表指针
KEYDN4: CP      (HL)          ; 查表
        JR      Z, KEYDN5     ; 找到，转至KEYDN5
        INC     HL
        INC     B             ; 修改键值
        JR      KEYDN4        ; 继续寻找，当在表中找到相应的编码
                              ; 时，在寄存器B中得到键值
KEYDN5: IN      A, (KBSEL)    ; 输入列值，检查键是否释放
        AND     1FH
        CP      1FH
        JR      NZ, KEYDN5    ; 未释放循环等待
        CALL    D20MS         ; 已释放，调用20ms作为DEBOUNCE
        LD      A, B          ; 键值→A
        CP      10H           ; 是命令键吗?
        JR      NC, KEYDN6    ; 是，转至命令键译码程序
        LD      HL, (KEYPTR)  ; 是数字键，把显示缓冲区地址→HL
        LD      (HL), B       ; 把键值存在缓冲区中
        OR      A             ; 清CY
        LD      BC, DSMEM1
        SBC     HL, BC        ; 已输入两个数?
        JR      Z, KEYDNA     ; 是，转至KEYDNA，置输入两个数标志
        OR      A
        LD      BC, DSMEM3
        LD      HL, (KEYPTR)
        SBC     HL, BC        ; 已输入4个数?
        JR      Z, KEYDN8     ; 是，转至KEYDN8，置输入4个数标志
        OR      A
        LD      BC, DSMEM7
        LD      HL, (KEYPTR)
        SBC     HL, BC        ; 已输入8个数?
        JR      Z, KEYDN9     ; 是，输至KEYDN9，置输入8个数标志
KEYDN7: LD      HL, (KEYPTR)
```

```
            INC   HL
            LD    (KEYPTR), HL ; 存指针
            JP    DISUP        ; 转至显示程序入口，等待输入新的键值
KEYDNA:     LD    HL, DIG2
            INC   (HL)         ; 置输入两个数标志
            JR    KEYDN7
KEYDN8:     LD    HL, DIG4
            INC   (HL)         ; 置输入四个数标志
            JR    KEYDN7
KEYDN9:     CALL  ALTER        ; 输入修改数据到MEM, PORT或REG
            LD    HL, (KEYPTR)
            DEC   HL
            LD    (KEYPTR), HL
            JP    DISUP
; 从命令键转移表中，找到命令键处理程序的入口。
KEYDN6:     SUB   10H          ; 得到偏移量
            LD    C, A
            ADD   A, C
            ADD   A, C         ; 偏移量×3
            LD    C, A
            LD    B, 0
            LD    A, (2FFF)    ; 取上、下档键标志
            BIT   0, A
            JR    NZ, DKB      ; 是下档键转至 DKB 处理
            LD    A, C
            CP    0AH
            JR    C, DKB
            LD    HL,JPTAB2-C  ; 设上档键指针
            ADD   HL, BC       ; 加偏移量找到所按命令键的入口
            JP    (HL)         ; 转至相应的命令键的入口
DKB:        LD    HL, JPTAB1   ; 置下档键指针
            ADD   HL, BC
            JP    (HL)         ; 转至入口
            ORG   01DCH
JPTAB1:     JP    CCS1         ; EXEC
            JP    CCS2         ; SS
            JP    MON          ; MON
            JP    MON'         ; MON'
            JP    NEXT         ; NEXT
```

```
            JP      LAST            ; LAST
            JP      CCS6            ; REG
            JP      CCS8            ; MEM
            JP      CCS9            ; BP
            JP      CCS7            ; PORT
            JP      MOVE            ; MOVE
            JP      CCS12           ; PROG
            ORG     0205H
JPTAB2:     JP      CCS11           ; LOAD
            JP      CCS10           ; DUMP
            JP      CCS5            ; REG'
            JP      DISP            ; DISP
            JP      USER4           ; 用户程序入口地址键 2FBE
            JP      USER3           ; 用户程序入口地址键 2FBC
            JP      USER2           ; 用户程序入口地址键 2FBA
            JP      USER1           ; 用户程序入口地址键 2FB8
            ORG     07DCH
KYTBL:      DB      0FFH            ; 0     B=01, A=0F
(键值转换表)DB      0EFH            ; 1     B=02, A=0F
            DB      0F7H            ; 2     B=02, A=17
            DB      0FBH            ; 3     B=02, A=1B
            DB      0DFH            ; 4     B=04, A=0F
            DB      0E7H            ; 5     B=04, A=17
            DB      0EBH            ; 6     B=04, A=1B
            DB      0CFH            ; 7     B=08, A=0F
            DB      0D7H            ; 8     B=08, A=17
            DB      0DBH            ; 9     B=08, A=1B
            DB      0DDH            ; A     B=08, A=1D
            DB      0EDH            ; B     B=04, A=1D
            DB      0FDH            ; C     B=02, A=1D
            DB      0DH             ; D     B=01, A=1D
            DB      0BH             ; E     B=01, A=1B
            DB      07H             ; F     B=01, A=17
            DB      0EH             ; EXEC  B=01, A=1E
            DB      0FEH            ; SS    B=02, A=1E
            DB      0EEH            ; MON   B=04, A=1E
            DB      0DEH            ; MON'  B=08, A=1E
            DB      0CDH            ; NEXT(LOAD)  B=10, A=1D
            DB      0CBH            ; LAST(DUMP)  B=10, A=1B
```

```
DB      0C7H        ; REG(REG')     B=10, A=17
DB      0BFH        ; MEM(DISP)     B=10, A=0F
DB      0BDH        ; BP(2FBE)      B=20, A=1D
DB      0BBH        ; PORT(2FBC)    B=20, A=1B
DB      0B7H        ; MOVE(2FBA)    B=20, A=17
DB      0AFH        ; PROG(2FB8)    B=20, A=0F
```

我们以几个具体的例子来说明上述程序。

1. 若按的为数字键 0，则B=01，A=0F。在上述程序一开始，C=0，DEC C，则C=FFH，接着 SRL B，则移位后B=0。C经过四次算术左移后即为C=F0H，与A相加得 FFH(这就是数字键 0 在表格中的编码)。把表格的起始地址 KYTBL 送入 HL，当执行 CP(HL) 时，就比较相等，此时B=0，即为键值。

KEYDN5 开始，监视与等待键释放，释放后插入 20ms的延时（在检查到有任何键闭合时，也插入 20ms 的延时)用以去除因键闭合和断开时的颤动引起的毛刺的影响——这称为 Debounce。然后判断为数字键，则存入显示缓冲区，转至显示程序（因为键盘输入的数据都要显示)。

2. 若按下的键为数字键 5。则找到此键闭合时，B=04，A=17。B应移三次才为 0，则此时C=FDH，经四次算术左移后为 D0H，与A相加的结果为 E7H，即为数据 5 对应的表格中的编码。

在 CP(HL) 时，必须经 5 次修改地址指针循环，才能比较相等，每循环一次B←B+1，故找到此数时B中的值为 5，即为该键值。

3. 若按下的为 MEM EXAM(存贮器检查) 键，则B=10，A=0F。必须经过 5 次移位才能使B=0，则此时C值为 FBH，经过四次算术左移后为 B0H，与A相加得 BFH，即为表格中 MEM EXAM 键的编码。

在 CP(HL) 时，必须经过 24 次（18H）循环才能找到此键，故找到时 B 中的值为 18H。这即代表此键的值。

在程序段 KEYDN5 中比较时它>10H，故确定为命令键，转至KEYDN6 进行命令键译码。

先减去 10H，则余下为 8，经过三倍，变为 24（18H）形成命令表中的偏移量。经检查为下档键，把下档键命令处理程序的入口地址跳转表的起始地址 JPTAB1 送至HL寄存器对。表中的每一条都是三字节的转移指令，故每一条指令之间差三个存贮单元。而命令键的值由 10H开始，故命令键的值减去 10H，三倍后即为该条命令在跳转表中的偏移量。如 MEM EXAM 命令与第一条命令之间就差 18H个存贮单元。

故把表格的起始地址 JPTAB1 送 HL，三倍后的偏移量送 BC，则 HL←HL+BC，JP(HL)，就能转至相应的命令处理程序。

第三节　TP801-A使用说明

（一）按钮 S1（红色）——位于机器右侧中部

功能：对整机提供复位（RESET）信号。

用法：机器接通电源后，按S1则数码管最左边的一位会出现机器响应符“P”。说明机器已在监控程序管理下准备接受用户的各种命令。

（二）开关 S2（黑色长方形）——位于S1下方

功能：开关S2有两个位置，即

1．向上拨动打到“MON”位置。此位置的作用是机器复位后处在监控程序管理下。此时，显示器出现“P”。

2．向下拨到PROM1位置。则在按S1（复位键）后，在机器完成了初始化后直接运行PROM1内的程序（此时就不在监控程序的管理之下工作）。

（三）开关 S3（黑色长方形）——位于S2下方

功能：开关S3有两个位置，即

1．向上拨到“READ”位置，即在监控程序下进行工作。

2．向下拨到“PGM”位置，即处于EPROM的写入状态。

（四）十六个数字键的使用说明

十六个数字键即0—9，A、B、C、D、E、F。

功能：

1．用来向计算机输入十六进制数字。这些数字可以组成四位十六进制的地址，它显示在数码管的左边四位。也可以组成两位十六进制数据，则在数码管的右边两位显示。

2．用于选择寄存器：当要求检查寄存器内容时，用一部分十六进制数码来选择寄存器。

（1）当要选择寄存器A、B、C、D、E、F（标志位寄存器）时，可按相应的16进制数字键A、B、C、D、E、F。则在数码管的最左边显示A、B、C、D、E、F等寄存器号，在最右边两位显示相应的寄存器的内容。

（2）当要选择H、L、IX、IY、I、PC、SP、IFF等寄存器时，则相应地按数字键7、8、4、5、6、1、2、3。则在数码管的最左边一位显示所按的数字键，但它代表相应的寄存器，即7代表H，8代表L…，3代表IFF。而在数码管的右边两位（对于8位寄存器）或右边四位（对于16位寄存器）显示该寄存器的内容。

（五）十二个功能键的使用说明

数字键右侧有四个键MON′、MON、STEP、EXEC。

数字键上方有八个功能键，这八个键可实现十六种命令。每个键上有两种命令。横

线上方的键符为上档键，受MON′键控制；横线下方的键符为下档键，受MON键控制。

1．MON键（MON itor 监控键）

用途：（1）使机器进入监控程序；

（2）中止现行程序的执行：若在程序的运行过程中按MON键，则在保护CPU内部寄存器后返回监控程序，在最左边的数码管显示出提示符“P”。此功能可使当程序进入死循环时，退出死循环。

（3）中止或退出当前的命令或输入的数据：无论在执行任何命令或输入任何数据时，按MON键，则中止当前的命令返回监控，显示提示符“P”。只有终止了前一个命令，才能进入下一个命令。所以，按MON键是结束一个命令操作的正常措施。

同时，按下MON键后，是下档功能键起作用。

注意：MON键与RESET键都使机器进入监控程序。但前者能保护CPU各寄存器的内容，而后者不保护。

2．MEM键（MEMory examine 存贮单元检查键）

用途：按此键可检查或更改RAM中的某存贮单元的内容。

利用更改存贮单元内容的方法来输入程序。利用检查存贮单元的内容可调试程序。

3．NEXT 键

用途：（1）检查或修改下一个RAM单元或I/O端口地址的内容；

（2）检查EPROM编程时的下一个错误。

4．LAST 键

用途：检查或修改上一个内存单元的内容。

5．REG键（REG ister examine 寄存器检查键）

用途：检查或修改CPU内部寄存器A、B、C、D、E、F、H、L、I、IFF、PC、IX、IY的内容。SP的内容只能读出，不能写入。

注意：寄存器内容的修改或检查，不能用NEXT键连续检查。每检查完一个寄存器的内容后，就用MON键中止命令返回到监控程序，再按下一个要检查的寄存器号，并按REG键完成新的寄存器检查。

6．EXEC键（EXECute 连续执行程序键）

此键用来连续执行存放在RAM、ROM或EPROM内的程序。

使用方法：

（1）当程序输入完后，按MON键，机器显示“P”。再输入程序的起始地址，并按EXEC键，则程序从输入的地址开始运行。若程序中有HALT指令，或程序中不停地循环，则当程序运行时显示器变暗，程序运行完，显示器也不出现“P”。只有按下MON键，显示器才出现“P”。

（2）当程序没有运行完，此时按MON键，则程序暂停运行，程序中的断点及CPU寄存器的内容会自动保护，并在显示器上出现“P”。或在程序中设置断点，当运行到断点时，若接着再按EXEC键，则程序从PC中包含的地址开始继续运行。

7．BP 键（Break Point 设置断点键）

该键用来在用户程序中设置一至五个断点。在调试程序时要求程序运行到某一条指

令之前，就停止运行。这样可检查前一段程序的运行结果是否正确，以便修改程序或者修改寄存器的内容及I/O端口地址中的内容。

设置断点的方法：

（1）按MON键显示“P”；

（2）键入断点地址显示××××（即键入的地址）；

（3）按BP键，仍显示××××（地址）。

说明你设置的断点地址已装入断点地址表（专门用于装入断点地址的一段存贮区）。所设置的断点地址必须为某条指令的第一个字节的地址。

设置第二个断点时，重复上述动作即可。

一个程序中最多可设置五个断点，要求最后一个断点设在停机指令所在的地址。

可用下述方法消除断点：

（1）按RESET键；

（2）按STEP（单步）键；

（3）在未键入4位16进制地址前按BP键。

三种方法任选一种即可消除所设的全部断点。

8. STEP键（single STEP 单步执行程序键）

用途：按此键只执行程序的一条指令。执行完这条指令后，在显示器的左边四位显示出下一条指令的地址；右两位显示出累加器A的内容。

用法：（1）按MON键，显示出“P”。

（2）用检查与更改寄存器内容的方法，给PC计数器置要执行的指令的地位。

（3）按STEP键

按一次STEP键，机器运行一条指令，然后按MON键回到监控程序，可以根据需要对存贮器、寄存器或外设端口等进行检查。检查结束后按MON键回到监控程序，若要继续执行下一条指令时，则直接按STEP键即可。

9. MOVE键（存贮块移动）

功能：当显示器上显示某一单元的地址时，按MOVE键，则从显示的地址单元开始，它以下各存贮单元内容均向下移动一个字节。而原显示的地址单元内容被清为“0”。

注意：TP801-A单板机在4K字节的RAM中，分配给用户可用的区域为2000H——2F87H，而MOVE键可以移动的范围为2000——2EFFH。也就是说从2F00——2F87H的136个字节的内容不受MOVE键的控制。这136个字节可放置数据表格和变量。

10. PORT键（PORT examine 端口检查键）

此键的用法与MEM键相似，只是它用来检查和修改输入输出端口地址的内容。

但注意，端口地址只需要两位十六进制数字。这两位数字显字在数码管的最左边两位。此端口地址的内容，显示在数码管的最右边两位。

11. MON′（交换键）

使机器回到监控程序的同时，使上档键有效。

从使机器回到监控程序的角度来说，MON′和MON键的功能是相同的，区别在于：按了MON键后是下档功能键有效；而按MON′键后是上档功能键有效。

在按MON′键后，在显示器的最左边一位出现提示符“′”。然后再按上面8个功

能键中的任何一个，则上档键功能有效。

12．REG′键（CPU 辅助寄存器访问键）

功能与REG相同，只是检查与修改的是 CPU 的辅助寄存器组。

用法：（1）按MON′键显示“′”；

（2）按A（即要显示的寄存器号）显示A；

（3）按REG′　　显示 A′××（内容）。

以此类推，用同样方法可检查或修改其它辅助寄存器的内容。

13．DISP 键

用途：计算相对转移指令中的偏移量。

方法：

（1）相对转移指令的源地址送入 IY；

（2）相对转移指令的目的地址送入 IX；

（3）按MON′键；

（4）按DISP键。

监控程序就进行偏移量的计算，计算完后，在显示器左边两位显示跳转方向，若为00，说明为正向转移；若为 FF，表示为负向转移。若为其它数据，说明偏移量是超出了使用范围。在显示器的右边两位显示计算结果，即用补码表示的偏移量，且把此偏移量填入到相对转移指令的第二字节。

14．DUMP键（转贮键）

把 RAM 中的程序或数据转录到盒式磁带上（在后面将结合实验介绍用法）。

15．LOAD键（装入键）

将已存贮在磁带上的信息，输入到RAM中去（在后面结合实验介绍用法）。

16．PROM 键（EPROM 编程键）

利用此键可把在 RAM 中的信息，写入到在 PROM2 位置上的 EPROM 中去（对 EPROM 编程）。使用方法在后面的实验中介绍。

17．2FB8、2FBA、2FBC、2FBE 四个程序启动键

当程序不太长时，可将四个程序同时输入到RAM中，且把它们的入口地址放到内存单元 2FB8、2FB9，2FBA、2FBB，……等 8 个单元中。则按上述四个键中的一个，就可随时运行相应的一个程序，相当于 EXEC 键的作用（具体用法在后面将结合实验予以介绍）。

第四节　实　　验

实验一　TP801-A单板机键盘操作练习

一、实验目的

通过实验了解和熟悉 TP801-A单板机键盘上各个键的功能及其使用方法。通过一些简单程序的练习进一步熟悉 Z80-CPU 的指令系统，能较熟练地使用单板机，以便学生

在单板机上进行简单的程序设计及对程序进行调试。

二、实验接线

1．将开关S2置于“MON”位置，开关S3置于“READ”位置。

2．接入电源前，注意用三用表测量一下，看电源电压的数值是否为+5V±10%。

3．将单板机左侧一对电源线的黑头接电源的公共端，红头接电源的正端。电源的输出电流要求大于1A。

4．本单板机没有任何过流保护措施。千万注意电源不能接反，否则单板机上的管子将因反向击穿而损坏。

5．接通电源后，按RESET按钮S1，数码管最左边一位显示出提示符“P”，说明单板机可正常工作。如不出现“P”，则马上断电并告诉指导教师检查原因。

三、实验内容

1．仔细阅读使用说明。

2．键盘操作练习。

（1）检查和修改存贮单元内容的练习。

给定如下程序：

地　址	目的程序	源程序		
2000	3E AA	LD A, 0AAH		
2002	01 BB 20	LD BC, 20BBH		
2005	11 CC 21	LD DE, 21CCH		
2008	21 DD 22	LD, HL, 22DDH		
200B	C5	PUSH BC	;	堆栈操作指令
200C	D5	PUSH DE	;	
200D	DD E1	POP IX	;	
200F	FD E1	POP IY	;	
2011	02	LD (BC), A		
2012	FD 7E 00	LD A, (IY)		
2015	2F	CPL		
2016	77	LD (HL), A		
2017	EB	EX DE, HL		
2018	D8	EX AF, AF′		
2019	D9	EXX		
201A	76	HALT		

①向机器输入给定程序。

掌握MON键、MEM键、NEXT键的使用

操作如下（字符下面划横线的是要用户用键盘打入的，没划线处为机器的显示）。

键入	显示
MON（进入监控程序）	P

2000（输入地址单元）	2000
MEM（存贮单元检查命令）	2000 ××
3E（输入数据）	2000 3 E
NEXT(检查下一单元命令)	2001 ××
AA（输入数据）	2001 A A
NEXT	2002 ××
01（输入数据）	2002 0 1
……	
……	

用上述方法把所给定的程序输入完毕。

想一想用××表示的这种随机数为什么会出现？这种随机数据的组合应有多少个？

②检查输入的机器码是否有误？

用 LAST 键由 201A 单元开始向上检查到 2000 单元，检查每一个存贮单元的内容与要求输入的内容是否相同，若相同就只作检查；若不相同，则重新输入两位正确的数。操作方法与 NEXT 键相同。

（2）对寄存器的内容进行检查。

REG 键、MON′键、REG′键的使用

键入	显示
MON（返回监控）	P
A（检查累加器A）	A
REG(打入寄存器检查命令)	A ××
MON(结束上一命令，返回监控)	P
C REG	C ××
MON	P
B REG	b ××
⋮	⋮
⋮	⋮
MON′(返回监控，上档键有效)	′
A REG′(检查辅助寄存器)	A′ ××

要求：用上述方法检查并记录寄存器A、B、C、D、E、F、H、L、IX、IY；A′、B′、C′、D′、E′、F′、H′、L′等的内容。

（3）运行已输入的程序

键入	显示
MON	P
2000(输入要执行的程序的起始地址，监控程序将它置入PC)	2000
EXEC	显示器变暗
等待一会时间	
MON	P

再次对各寄存器的内容和内存单元 20BBH、21CCH、22DDH的内容进行检查和记录。比较程序运行前后的变化。分析程序中各条指令的功能。

（4）用设置断点的方法运行上述程序

①设置断点

键入	显示
RESET	P
200 B	200 b
BP	200 b
MON	P
2011	2011
BP	2011
MON	P
⋮	⋮
⋮	⋮

用上述方法最多可设置五个断点。

②运行程序

键入	显示
MON	P
2000	2000
EXEC	200 b
EXEC	2011
⋮	
⋮	

每遇到一次断点，检查一次寄存器和有关的内存单元的内容。

注意：为了运行整个程序，第五个断点要设置在停机指令（HALT）处。这是因为在上述的设置断点的操作结束以后，实际上只是把设置的断点地址放入断点地址表中。而是在按 EXEC 键后，监控程序从断点地址表中找到各个断点地址，且用 RST8（操作码为 11001111）指令代替各个断点地址处的用户指令的第一字节，且把它们保存在断点地址表中。当程序运行到断点地址时，取出并执行 RST8 指令，它保护了现场，从断点地址表中取出各个断点处的用户指令送回原处后返回监控。从而使用户程序中断，且可用监控命令来检查前一段程序运行后的结果，便于调试用户程序。在检查完再按EXEC键时，就从断点处接着往下运行，且重复上述安置断点与执行 RST8 指令的过程。但是若不在HALT指令处设置断点，则从最后一个断点处用EXEC键接着执行时，因断点未消除，则监控程序仍用RST8 指令代替各断点处的用户指令，但在用户程序执行到HALT指令时未执行 RST8 指令，所以未能恢复断点处的用户指令。因此，当用 MON 键来停止用户程序返回监控时，在用户程序的各个断点处仍是RST8 指令，即使用上述的三种消除断点方法的任一种时，都不能改变这种情况，从而使用户程序无法执行。

③消除断点（方法前面已介绍）

但要注意，消除断点只是消除断点标志，使在按EXEC键执行程序时不再安放断点，

程序按无断点的方法运行。

（5）单步调试程序练习：STEP 键的用法

键入	显示
MON	P
1(即PC)	1
REG	1 ××××（显示PC的现行值）
2000	1 2000
STEP	2002 ××（××是累加器A的当前内容）
MON	P

此时可用命令检查各个寄存器与存贮单元的内容，以了解指令执行的结果是否正确。

⋮

⋮

MON	P
STEP	2005 ××（2005是下一条指令的地址，××总是累加器A的当前内容）

⋮

⋮

这样可以一条一条地执行指令，而且可以检查每一条指令执行的结果，这是调试程序的重要方法。把单步运行与设置断点结合起来，可以很快地找到程序中的错误和予以更正。同学们要学会这样的调试程序的方法。

（6）计算相对转移偏移量

给定如下程序：

地　址	目的程序	源程序
2200	3E 00	LD A, 00H
2202	06 05	LD B, 05H
2204	3C	LOOP: INC A
2205	10 □	DJNZ LOOP
2207	76	HALT

用上述的检查和更改存贮单元内容的方法输入上述程序，而当输入到 2006 单元时，不输入任何数据，把它空过去（即保留它原来的内容），由计算偏移量的程序把计算得到的偏移量填入。

①用 DISP 键计算相对转移偏移量

键入	显示
MON	P
5（即 IY）	5
REG	5 ××××
2205(键入源地址——即相对转移指令的存放地址	5 2205
MON	P

4（即 IX）	4
REG	4 ××××
2204（键入目的地址）	4 2204
MON′(结束上一命令，上档键有效)	′
DISP	FF FD

显示器左边两位表示跳转方向：当为 00 时，表示正向转移，FF 表示负向转移。若不是这两个数据，表示偏移量超出了使用范围。在本例中为 FF，即为负向转移。

右面两位为计算所得的用补码表示的偏移量。本例中的 FD 就为－3。

②用检查存贮单元的方法，检查 2206 单元的内容是否是计算所得的偏移量。

（7）修改程序的举例

①增填新的指令

若在两条指令之间要增填一条新的指令，举例如下：

若在前述的已输入至 2000H 单元开始的程序中，在 200C 单元的指令 PUSH DE 和 200D 单元的指令 POP IX 之间增加如下的两条指令：

机器码	源程序
E5	PUSH HL
80	ADD A,B

增加的方法：

键入	显示
MON	P
200D	200d
MOVE	200E dd（把原在 200d 中的内容 dd 移入 200E 中）
MOVE	200F dd（dd 移入 200F 中）

检查 RAM 中 200D 和 200E 两单元：

MON	P
200D	200d
MEM	200d 00
NEXT	200E 00
NEXT	200F dd
NEXT	2010 E1
⋮	⋮
⋮	⋮

说明原来 RAM 中从 200D 开始到 201A 单元，整个数据块向下移动了两个单元，移至 200F 到 201C，而 200D 和 200E 单元被清 0，因此，可向 200D 和 200E 这两个单元增加两条指令的机器码。

MON	P
200D	200d
MEM	200d 00
E5	200d E5

NEXT　　200E 00

80　　200E 80

②删掉某条指令：

若在程序修改过程中，要删除掉某条指令。但为了使程序中的转移地址不再重新计算和变换，则可将被删掉的指令所占的全部地址单元的内容置“0”，即把它们变为NOP指令。

例如有下列程序

地　址	目的程序		源　程　序
2000	21 40 20		LD HL, 2040H
2003	46		LD B, (HL)
2004	0E 00		LD C, 00H
2006	23	LOOP1：	INC HL
2007	7E		LD A,(HL)
2008	A7		AND A
2009	FA 0D 20		JP P, LOOP2
200C	0C		INC C
200D	05	LOOP2：	DEC B
200E	20 □		JR NZ, LOOP1
2010	79		LD A,C
2011	32 40 20		LD (2040H), A
2014	76		HALT

当把　200D　DEC　B

　　200E　JR　NZ, LOOP1

这两条指令合并成　DJNZ LOOP1

此时可变更如下：

200D　00　NOP

200E　10 □　DJNZ LOOP1

而其它的指令都不变。

四、实验前预习要求

1. 仔细阅读使用说明；

2. 仔细阅读本实验指导书；

3. 弄清楚实验用的程序的作用；

4. 要求记录的项目事先画好表格，并对所要测的数据和地址做好估计，作到心中有数。

五、实验报告要求

1. 整理好所测得的数据与理论的结果相比较。

2. 小结调试程序的方法。

实验二　简单的编程实验

一、实验目的

1．进一步熟悉监控命令的使用；

2．练习编制简单程序；

3．进一步掌握调试程序的方法。

二、实验内容

1．在内存单元 2022H 开始，建立 0——99（以二进制表示）一百个数。给定程序的流程图如图 1.10 所示。

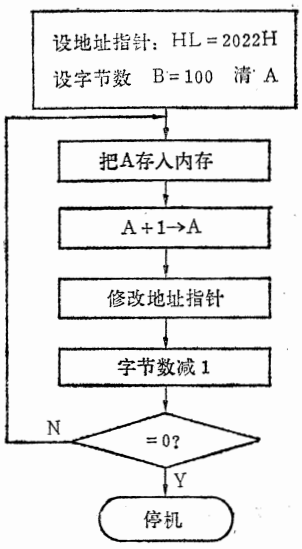


图1.10　在存贮区建立0—99的程序流程图

步骤：

（1）根据流程图（或参照流程图）编好用汇编语言写的源程序，并翻译成机器码，如上一个实验中的例子所示。指令中遇到 16 位二进制数时，要注意低位字节在前，而高位字节在后（地址高的存贮单元）。

（2）运行程序

① 先用 EXEC 键连续运行，检查运行结果是否正确（即检查自2022H开始的100个存贮单元的内容）。

② 若结果不对，可用设置断点或单步运行两种方法检查出错地点，并改正之。

③ 若要增填指令，则用 MOVE 键。

④ 若要删去某条指令，可把此指令所占的全部单元的内容，都换成00H。

⑤ 若一开始运行就正确，可以有意识地练习使用设置断点或单步运行的方法。

（3）检查和记录 5、10、15、20、25、30、40、50、60、70、80、90，这些数的机器码（二进制表示）和所在的地址。

2．用数据传送程序将上述数据块传送到以 2090H 单元开始的存贮区中去。

要求：

（1）给定数据传送程序的流程图如图 1.11 所示，编好源程序，并翻译成机器码。

（2）运行程序和检查结果，直至程序运行完全正确。

（3）检查和记录 5、10、15、20、25、30、40、50、60、70、80、90 等这些数所在的地址，包括原来的地址和传送以后的地址。

3．用单条数据块传送指令和成组数据块传送指令，将数据块从 2022H 为起始地址的存贮区，传送到以 2090H 单元为起始地址的存贮区中去。

要求：

（1）给定两种方法的程序流程图如图 1.12 和 1.13 所示，编出各个源程序，并翻译

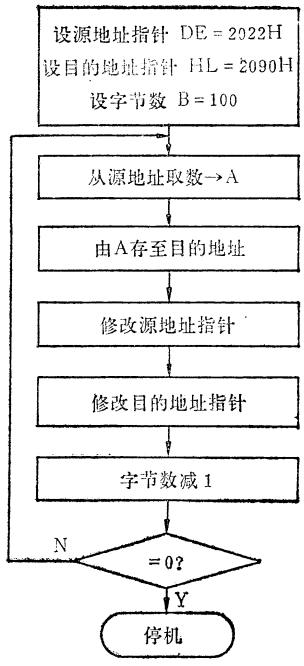


图1.11 数据块传送程序流程图

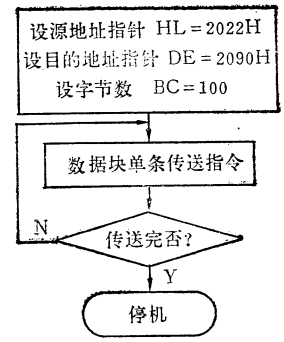


图1.12 用单条数据块传送指令的程序流程图

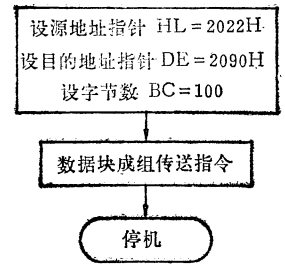


图1.13 用成组数据块传送指令程序流程图

成机器码。

（2）运行程序和检查结果，直至程序运行完全正确。

（3）检查和记录5、10、15、20、25、30、40、50、60、70、80、90，这些数所在的源地址和目的地址，并检查与以前记录的是否相同。

4. 选作：把数据块从2022H开始的存贮区传送到2050H单元开始的存贮区。编写此程序并运行；检查和记录5、10、15、20、25、30、40、50、60、70、80、90、95、99，这些数的源地址和目的地址。

三、预习要求

1. 复习使用说明；
2. 仔细阅读本实验指导书；
3. 根据要求，实验前编好各个源程序，并翻译成机器码；
4. 要求记录的项目事先画好表格，并对要测的数据和地址进行估算，作到心中有数。

四、报告要求

1. 整理好运行正确的源码程序；
2. 整理好所测得的数据；
3. 比较出三种数据块传送程序那个最好（即所占的内存单元少，执行的时间短）；
4. 对实验课程提出意见及要求。

实验三　编程练习及 DUMP 和 LOAD 键的使用

一、实验目的

1. 熟悉 ASCII 码表；
2. 练习编制线性搜索程序的方法；
3. 进一步熟悉调试程序的方法；
4. 学习使用 DUMP 键和 LOAD 键，转贮和装入信息。

二、实验内容

1. 查 ASCII 码表，查到 SP、！、”、#、$、…10 个连续的字符的 ASCII 码值。以这 10 个字符为一组，在以 2100H 单元为起始地址的存贮区中，连续存放 10组上述的 10 个字符。此程序的流程图如图 1.14 所示。

2. 要求编出程序搜索 $ 字符，并记下 $ 字符所在的全部地址。

（1）用单条的数据块搜索指令编程序，其流程图如图 1.15 所示。

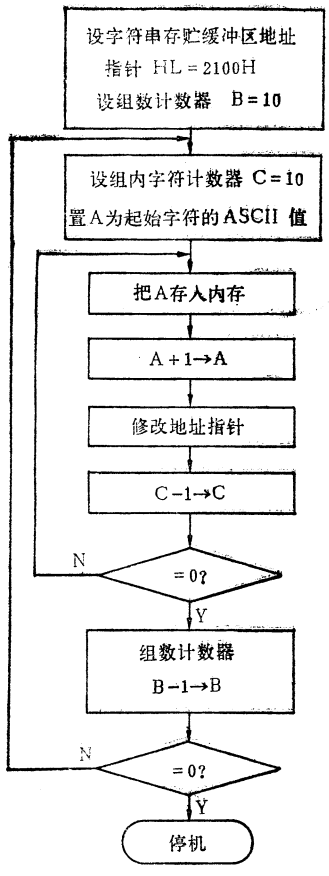


图1.14　在存贮区建立10组10个连续的字符的程序流程图

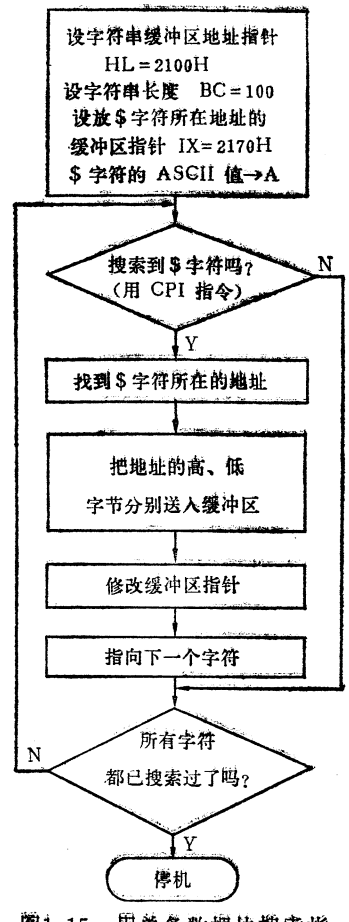


图1.15　用单条数据块搜索指令进行线性搜索

（2）用成组的数据块搜索指令编程序，其流程图如图 1.16 所示。

编程时注意：

① ASCII 码为七位二进制码，可把 D_7 视为“0”，则一个字符的 ASCII 码用一个字节表示。

② 设主字符串的程序用双重循环，一是组数循环（外循环），一是组内字符数循环（内循环）。

③ $字符所在的地址，一个地址要占两个字节，故存放它的地址也要建立一个缓冲区，也要设一个缓冲区指针。要注意它的地址的分配，不要与程序区和字符存放区相重叠（参考前面介绍的 RAM 存贮区的地址分配）。

3. 要求在存贮区建立 0 -99 共 100 个数，但要求奇数为负，偶数为正。找出这 100

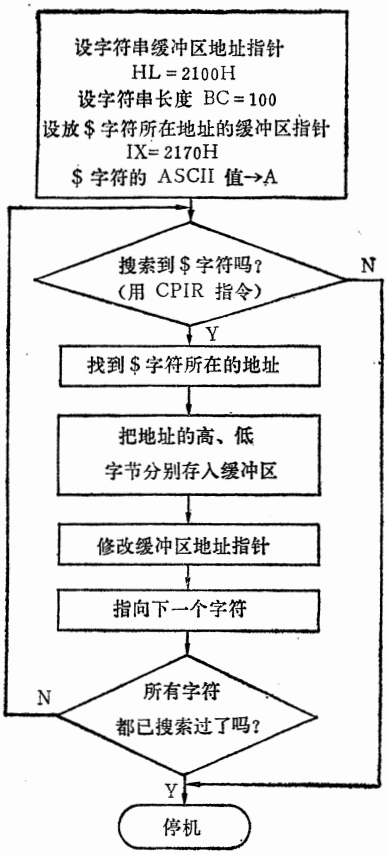


图1.16 用成组数据块搜索指令进行线性搜索

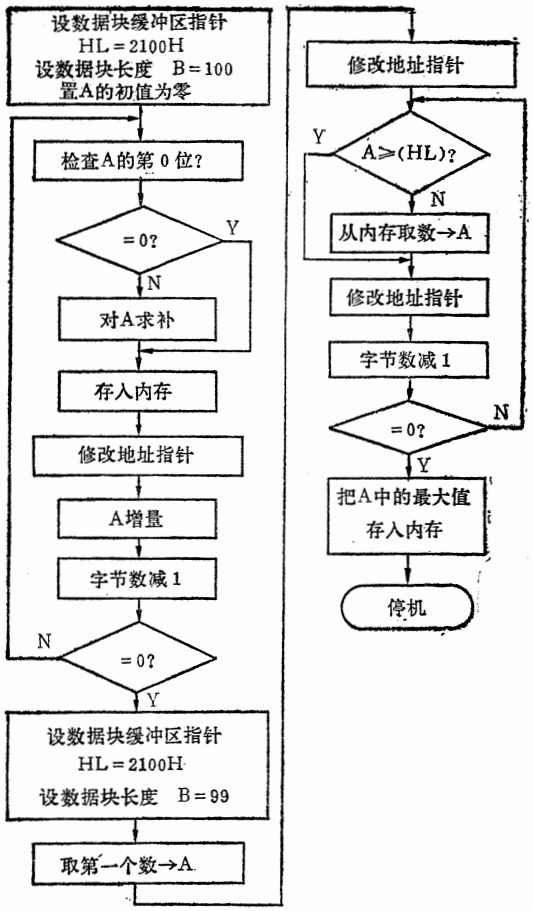


图1.17 在内存中建立100个正、负数并取出最大值的程序流程图

个数中的最大值，并把它存放到某个存贮单元。程序的流程图如图 1.17 所示。

4. 建立、运行和调试上述这些程序，直至运行正确。

5. 用下面所介绍的方法，把最后一个程序经过调试后的机器码程序转贮到磁带上；并从磁带上重新装入程序，检查转贮和装入过程是否正确。

三、预习要求

1．参照流程图，在实验前编好各个源程序，并翻译成机器码，分配好存贮区。

2．把要测的数据画好表格。

四、报告要求

1．整理出运行正确的各个源程序；

2．整理好要求测出的数据；

3．对实验内容提出要求及改进意见。

五、单板机信息（RAM中）转贮键DUMP和磁带信息装入键LOAD的使用介绍

1．DUMP 键（cassette DUMP 信息转贮键）

此键用来将 RAM 中的信息转贮到录音机的磁带中。

操作方法如下：

（1）将磁带装入录音机。

① 接通录音机电源；

② 将开关按钮拨向 TAPE（带）一边；

③ 计数器复零。

（2）用信号转录线将单板机的 J_2(AUX) 端与录音机的“MIC”输入端相连。

（3）用 MEM 键将内存中被转贮程序的起始地址的高位字节置入 2FC0H 单元，低位字节置入 2FC1H单元（注意：在程序中凡是遇到地址，都是存放地址的低位字节的单元其地址值小，而地址高位字节存放单元的地址值大，恰好与这里的要求相反）。

（4）用MEM键，将内存中被转贮程序的终了地址的高位字节置入2FC2H单元，低位字节置入 2FC3H单元。（同样与程序中的地址值置入存贮单元的次序恰好相反）

（5）按MON′键，显示提示符“/”。

（6）把磁带倒回零，使磁带机上的计数器为“0”。按下录音键，先放空一段，使磁带的导带走过去（计数器从 0 走到 5 即可）。也可以录入声音，说明这是什么程序。

（7）按DUMP键，则提示符“/”消失。

（8）当显示器上再次出现提示符“/”，说明已转录完毕，立即按录音机的Stop键。此时记录计数器的数码。

当要连续转贮第二个程序时，先让磁带走一段留下间隙，或录入声音说明下一个程序是什么，并记录磁带机的计数器值，这就是第二个程序的起点。并重复步骤（3）、（4）、（5）、（7）和（8）。

一盒磁带可记录多个程序。用磁带机上的计数器，记录被录制程序开始时和程序结束时的数据，这样对每个程序在磁带上的位置可以心中有数。同时记下每个程序的特征及用途，以备将来重复使用这些程序。

（9）录制完后将磁带倒回起始位置，以防止把磁带上的信息装入机器时，发生倒装。

2．LOAD 键（把磁带上的信息装入 RAM 键）

此键用来将已录制在盒式磁带上的信息，装入到 RAM 中去。

操作方法如下：

（1）将磁带装入录音机，并将录音机的开关拨至 TAPE 位置。

（2）将录音机的音调控制旋钮调到最高。音量控制旋钮调到适中。

（3）用转录线将录音机的“MONITOR OUT”或“EAR PHONE”孔与单板机的 J_1(EAR) 孔相接。

（4）按MON′键，显示提示符“′”。

（5）将录音机置成放音方式。

（6）当磁带已倒回时，使计数器回零。然后根据所记录下来的某一个程序在磁带上的位置（即磁带机计数器的值），将录音机磁带走到相应的计数器的数值。

（7）按 LOAD 键（要动作迅速），提示符“′”消失。

（8）此时，单板机右上角红色 LED 应发出明亮的光。若不发光，则马上加大音量，直到 LED 发光。在整个程序输入过程中，LED应保持发光。

（9）若输入过程是成功的话，则输入完毕后，显示器出现“′”。

(10) 若输入过程中某条指令有错误，则显示器将显示出下一条指令第一字节的起始地址，说明上条指令有差错。此时应重新进行装入工作。在输入前要检查音调和音量的位置是否设置恰当。

(11) 如果在同一盒磁带上有几个文件（一个程序就称为一个文件），若要连续装入几个文件，当磁带走到两个文件之间的空档时，二极管 LED 光较暗并有闪动，当进入到新的程序时，LED 则特别明亮。

实验四　不同进位制数相互转换的程序设计

一、实验目的

1. 进一步掌握单板机的使用和调试程序的方法。
2. 学习和掌握不同数制之间相互转换的程序设计方法。
3. 学习用户程序入口地址键的使用。

二、实验内容

1. 若在存贮器中，自 BUFFER 单元开始，放有一个数据块。其中，BUFFER 和 BUFFER＋1 两单元中存放此数据块的长度，自 BUFFER＋2 单元开始连续存放以 ASCII 码表示的十进制数字符。要求把它们转换为 BCD 码，且把两个相邻单元的数码并成一个单元（地址高的数放在高四位），仍放在自 BUFFER＋2 开始的存贮区中，把新的数据块的长度送入 BUFFER 和 BUFFER＋1 单元中。

在转换以前，先用程序在内存中建立 10 组 0—9 的 ASCII 码，在数据块的开始两个单元置入数据块的长度，程序的流程图如图 1.18 所示。编出源程序，翻译成机器码，装入内存，运行和检查结果。只有在此程序运行正确以后，才进行转换工作。

转换程序的流程图如图 1.19 所示。

参照流程图编出源程序，并把它翻译成机器码。此程序的存放地址不要与上述程序相重叠。运行此程序并检查结果，经过调试，直至程序运行正确。

2．若在存贮区中，自BLOCK单元开始放有一个数据块。其中，BLOCK和BLOCK＋1两单元中，放的是此数据块的长度，自BLOCK＋2单元开始存放数据，每一单元放的是两位BCD码。要求把每一个

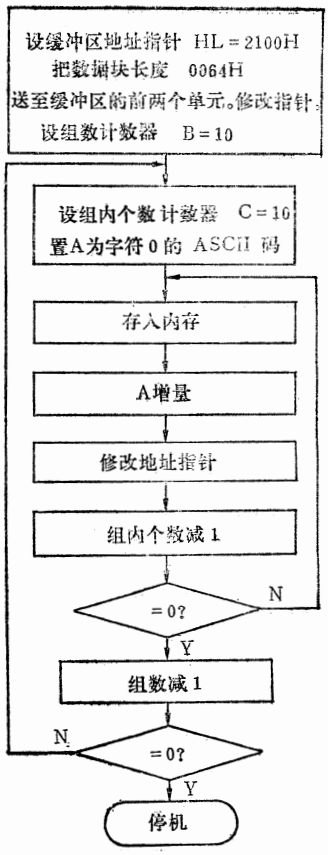


图1.18 在内存中建立10组0—9的ASCII码的程序流程图

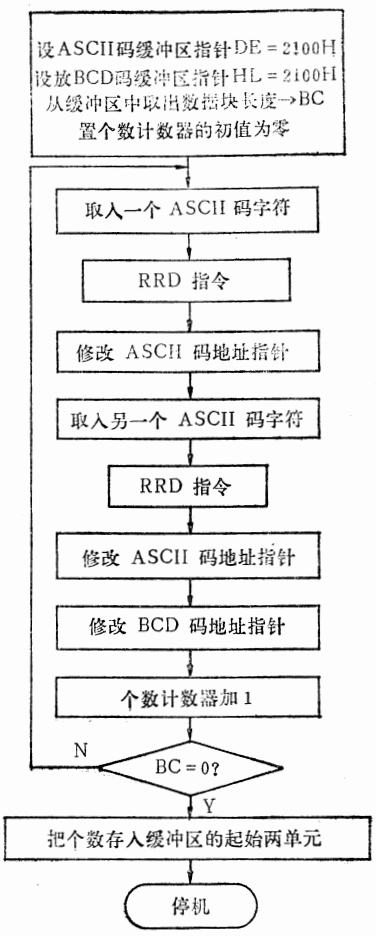


图1.19 把ASCII码转换为BCD码的程序流程图

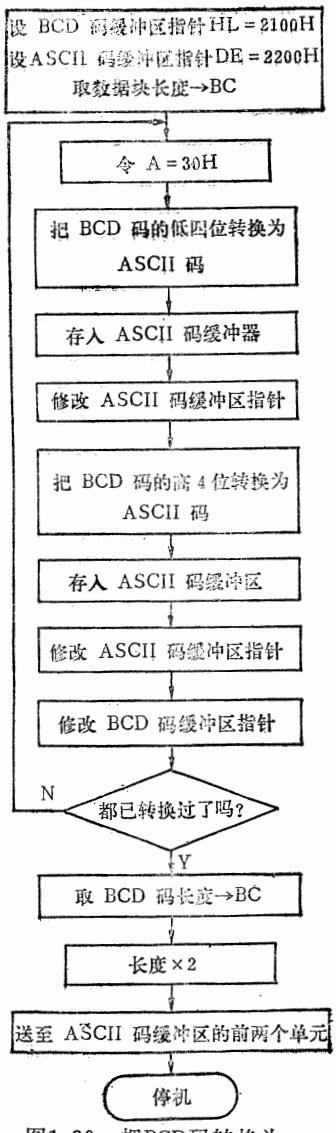


图1.20 把BCD码转换为ASCII码的程序流程图

单元的两位BCD码，分别转换为ASCII码，十位数的ASCII码放在后面，存放到一个新的缓冲区。在此缓冲区的开始两个单元，存放新的数据块的长度。

此程序只有在前一个程序运行正确以后才能运行。程序的流程图如图1.20所示。

3．在内存中，自BUFFER单元开始，放有一个数据块，其中，BUFFER和BUFFER＋1两单元中放的是数据块的长度，自BUFFER＋2单元开始，存放的是以ASCII码表示的十六进制数码（即′0′—′9′，′A′—′F′），要求把它们转换为十六进制数码（即0—9，A—F），并放入原单元中。

要实现转换，首先要用程序在内存中建立这样的数据块。现要求建立10组，每组都

为′0′—′9′，′A′—′F′，程序的流程图如图 1.21 所示。

程序与前面的建立 10组 10 进制数的ASCII码的程序相似，也要用双重循环，外层为组数循环，内层为组内个数循环。给A设置一个初值，每次内循环令其增量即可。但在16 进制数中有一个问题，即′0′—′9′的 ASCII 码值是连续的，′A′—′F′的 ASCII 码也是连续的，但′9′与′A′之间却不是连续的。所以，内层循环要分成两段，A分别要设置两个不同的初始值。

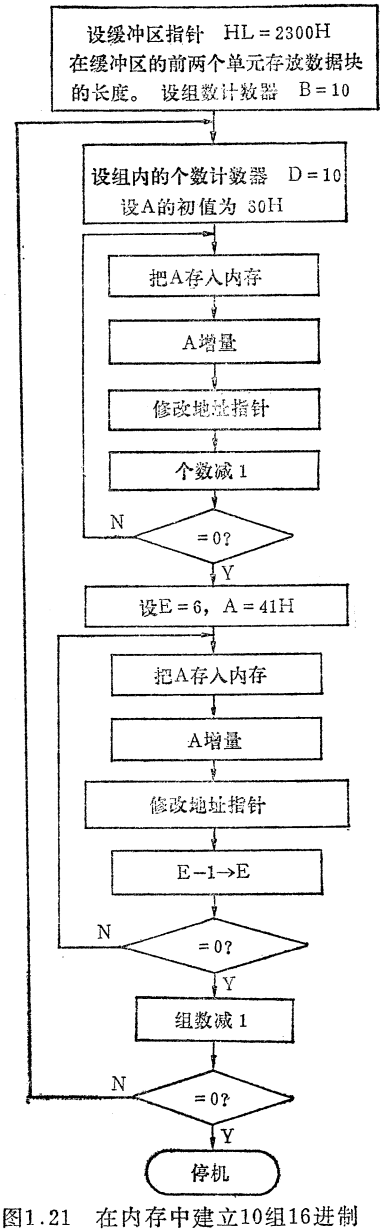


图1.21 在内存中建立10组16进制数码的ASCII码的程序流程图

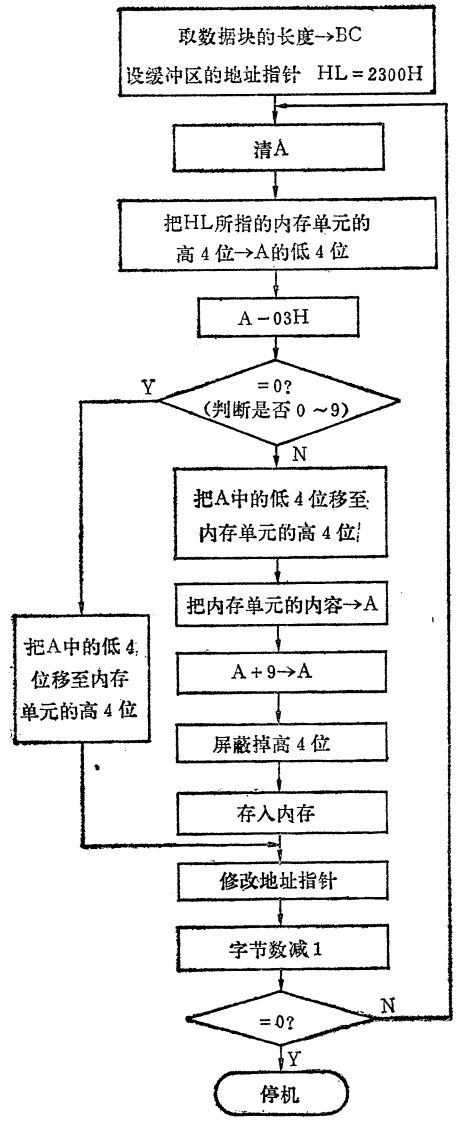


图1.22 把16进制数码的ASCII码转换为相应的16进制数码的程序流程图

参照流程图编出源程序，翻译成机器码，装入内存并运行。检查运行结果，直至运行完全正确。

转换程序的流程图如图 1.22 所示。

十进制数码的转换是十分方便的，只要屏蔽掉高四位即可；但若是'A'—'F'在屏蔽掉高四位后就变为01H～06H，必须分别加上9才能得到正确的值。所以，在程序中必须判断、区分，然后给以不同的处理。而'0'—'9'与'A'—'F'可以从高四位来加以区分，把它们的高四位单独取出来，减以3，结果为0的则为'0'—'9'；否则，即为'A'—'F'。

三、预习要求

1．仔细阅读本实验指导书。

2．每一个程序参照给出的流程图，在实验前编出源程序并翻译成机器码，且考虑程序的调试步骤和方法。

3．仔细参阅RAM地址区的分配，对实验中各个程序的程序区和数据区作好认真的地址分配，一定不能互相冲突。

四、报告要求

1．整理出每一个运行正确的源程序；

2．画出每题输入到RAM中的一组数据的存贮区域示意图（把一组数据块的字节长度及前五个数和最后五个数及其地址填写好）；

3．画出每题程序运行后的存贮区示意图，同样填入字节长度及前五个数和最后五个数及其地址。

五、上档键2FB8、2FBA、2FBC、2FBE的使用简介

上档键2FB8、2FBA、2FBC、2FBE可用于启动四个程序。

1．用法：把四个程序在内存中的起始地址送入专门设置的用户程序，启动地址表中的2FB8、2FB9，2FBA、2FBB，2FBC、2FBD、2FBE、2FBF。注意输送启动地址时，与程序中输送的方法一样。即低位字节在前，高位字节在后。四组启动地址的八个单元是有固定配合的，即不能把2FB9与2FBA组合在一起。

2．举例：有四个程序的起始地址，即

程序1：2000H；程序2：2100H；程序3：2200H；程序4：2300H

（1）当把程序输入到RAM中后，将四个程序的起始地址，分别送到启动地址表中。

键入	显示
MON	P
2FB8 MEM	2Fb8 ××
0 0	2Fb8 0 0
NEXT	2Fb9 ××
2 0	2Fb9 2 0
NEXT	2FbA ××
0 0	2FbA 0 0
NEXT	2Fbb ××

2 1	2Fbb 2 1
NEXT	2FbC ××
0 0	2FbC 0 0
NEXT	2Fbd ××
2 2	2Fbd 2 2
NEXT	2FbE ××
0 0	2FbE 0 0
NEXT	2FbF ××
2 3	2FbF 2 3
MON	P

四个启动键可同时用四个，也可以任选一个、二个、三个，但最多用四个。

（2）运行程序

键入	显示
MON′	′
2FB8	变暗
MON	P

可测得其运算结果。

用同样的方法可运行其它三个程序。但在运行前按一下 RESET 键。

实验五　八位数算术运算编程设计

一、实验内容

1．两个多字节十进制数编程设计

若在内存自2100H单元开始，连续存放两个10字节的十进制数，要求求和，且把"和"接着原来的数存放。求和的程序流程图如图1.23所示。

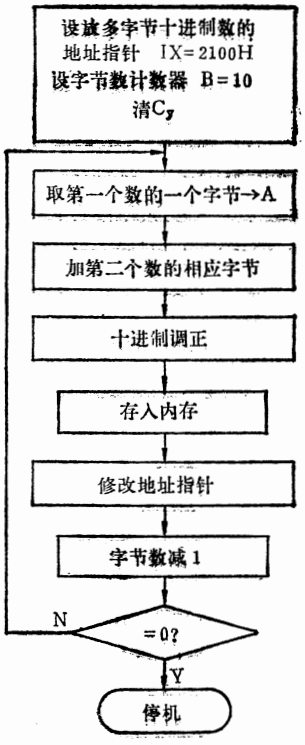


图1.23　多字节十进制数加法程序流程图

步骤：

（1）把实验前编好的且翻译成机器码的程序装入RAM中。

（2）用存贮器检查和更改的方法，自2100H单元开始，置入两个10字节的十进制数(用BCD码表示)，精心选择这两个数，使能看出DAA指令的调正功能。

（3）在程序运行时，有意识地在DAA指令前和DAA指令后设置断点。每次在DAA指令前，检查和记录累加器A中的内容和标志位（主要是Cy和H标志）的状态；在DAA指令后仍检查和记录累加器A中的内容和标志位的状态，以深入了解DAA指令是根据什么调整和如何调正的。

标志位的检查方法是这样的：

键入	显示
MON	P
F	F
REG	F 04

若如上例所示，F的内容为04H，又如何确定各个标志位的状态呢？

我们知道F寄存器实际上是各个标志位的集合，Z80-CPU中各个标志位的安排如下：

D_7							D_0	
S	Z	×	H	×	P/V	N	Cy	
0	0	0	0	0	1	0	0	04H

把检查到的F内容的两位十六进制数，展开成八位二进制数，则每一个标志位对应一位二进制数，这就是某个标志的状态。若如上例，F的内容为04H，在展开后就知道除P/V标志为“1”以外，其它标志全为“0”。

（4）检查最后运行的结果，调试程序，直至运行结果完全正确。

2．用累加的方法，做两个两位十进制数的乘法。

若有两个两位的十进制数（用BCD码表示）要相乘，当然可以把它们各自转换为二进制数，然后用二进制乘法的程序，把乘积再转换为BCD码。也可以用累加的方法，利用DAA指令，直接得到十进制的乘积。

两数相乘，可以用乘数（或被乘数，取其中较小的一个数）作为循环次数，控制部分积（初值为零）加被乘数（或乘数）的次数。在两个十进制数相乘时，加法要用DAA调正，而且部分积是两个字节的，低位字节与被乘数（或乘数）相加后，要考虑把所产生的进位加到高位字节上去，加完后也要用DAA调正；循环次数减1后也要用DAA调正。程序的流程图如图1.24所示。

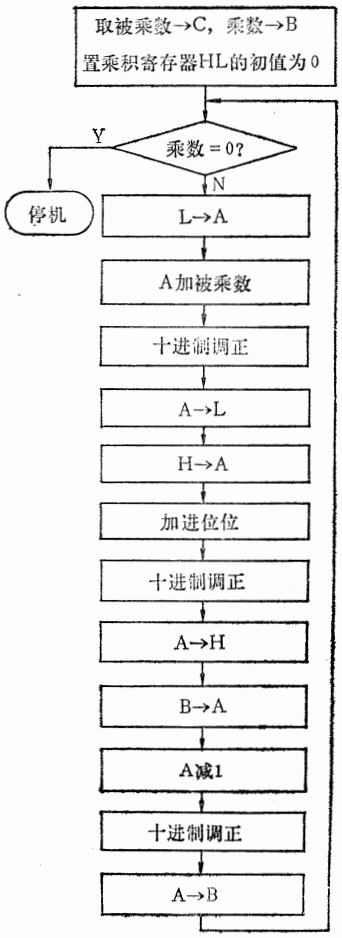


图1.24 利用重复相加的办法实现两个两位十进制数相乘的程序流程图

3．编两个八位带符号数的乘法程序。

两个数相乘，同号的乘积为正，而异号的乘积为负。故两个带符号数相乘，可把它们的符号位单独取出来，通过异或运算得到乘积的符号（相同两数异或的结果为“0”——即正，而不同的两数异或的结果为“1”——即负）。这样乘法就可以做绝对值（数值）相乘（用教科书中所介绍的方法）。但是在机器中，带符号数是用补码表示的，对于正数来说，把八位数看成是数值，这是正确的；而对于负数，只有经过取补（连符号位一起取补），才是它的数值。所以，在程序上在取出了被乘数或乘数后，先把它们的符号位取出来，且加以保存（备作以后进行异或运算）

以后，还要判断是正数还是负数，对于负数必须经过取补，找到它的数值。然后把这两个数值相乘(可采用部分积右移的方法；也可采用被乘数左移的方法)。得到乘积的绝对值后，对两个符号位进行异或运算，若乘积的符号为负，则必须对乘积的绝对值取补。程序的流程图如图1.25所示。

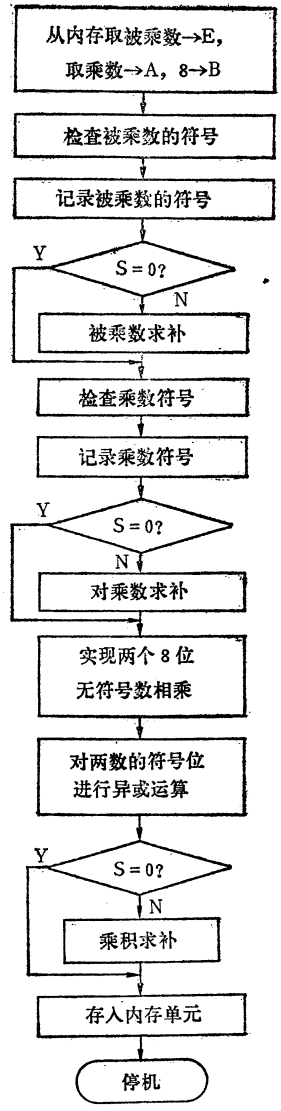


图1.25　两个八位带符号数的乘法程序流程图

4．编两个八位带符号数的除法程序

带符号数的除法的处理方法与带符号数的乘法的处理方法相似。编程的思路如下：

（1）负数用补码表示。

（2）确定被除数、除数和商所用的寄存器。

（3）检查被除数的符号并记录，若为负数则取补，变数值部分为绝对值。

（4）检查除数是否为零（除数为零，除法就无意义，故不做除法）。

（5）检查除数的符号并记录，若为负数，同样需取补。

（6）用两个八位无符号数相除的方法求商。

（7）被除数、除数的符号位相异或，求出商的符号位。

（8）商的符号为负时，则对求出的商取补。

二、预习要求

1．仔细阅读本实验指导书

2．参照流程图，在实验前编好各个源程序，并翻译成机器码。

3．分配好各个题所用的程序区和数据区，做到各得其所，互不冲突。

4．仔细预习DAA 指令的功能，精心设计运算的数据。

三、报告要求

1．整理好每一种运行正确的源程序，画出存贮器分配示意图。

2．每一种程序写出三组数据的运算结果。

3．总结DAA 指令根据什么进行调正？以及如何进行调正？

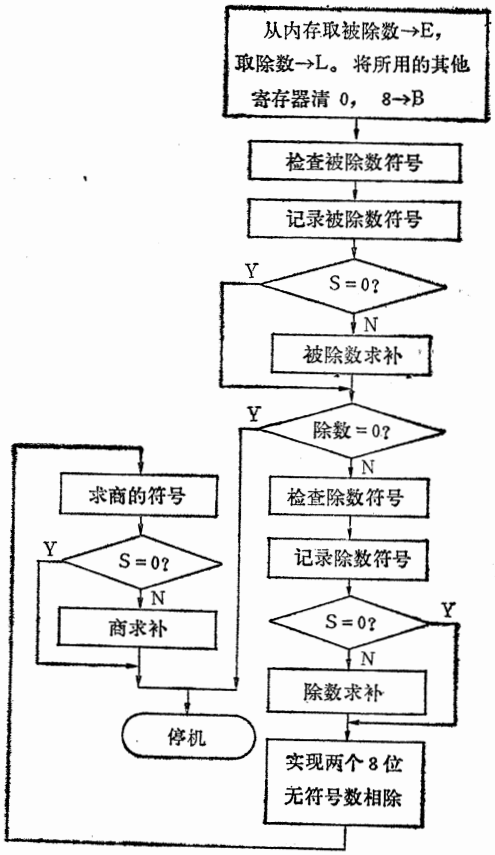


图1.26　两个 8 位带符号数除法程序流程图

实验六　子程序调用练习

一、实验内容

1. 给定下列程序，可使字符“8”从显示器的右端向左端不停地循环移动。

```
地址    机器码          源程序
2000    16  06          LD      D, 06H      ; 显示位置变迁次数
2002    3E  00          LD      A, 00H      ; 字符“8”的字模编码
2004    D3  88          OUT     (88H), A    ; 输出至段形锁存器
2006    3E  01          LD      A, 01H      ; 使最右面一个显示管
                                            ; 亮的数位控制字
2008    D3  8C    LOOP: OUT     (8C), A     ; 输出给数位锁存器
200A    CD  00  21      CALL    2100H       ; 调用延时子程序
200D    07              RLCA                ; 变迁数位
200E    15              DEC     D           ; 显示次数减1
200F    20□             JR      NZ, LOOP
2011    76              HALT
; 延时子程序
2100    F5              PUSH    AF
2101    C5              PUSH    BC
2102    01  FF  03      LD      BC, 3FFH
2105    3E  FF    TIME2:LD      A,  0FFH
2107    3D          BACK:DEC    A
2108    20              JR      NZ, BACK
210A    0B              DEC     BC
210B    78              LD      A,  B
210C    B1              OR      C
210D    20              JR      NZ, TIME2
210F    C1              POP     BC
2110    F1              POP     AF
2111    C9              RET
```

显示管的原理和TP801-A的显示程序，在本部分的第二节作了较详细的分析，在实验前必须认真阅读，弄懂原理。

我们知道，只要改变输送给段形锁存器的字模编码，就可以改变显示的字符。查阅本部分的表1-4，确定要显示的字符“2”、“4”、“A”、“F”的字模编码。

只要改变输送给数位锁存器的数位控制字，就可以控制让哪一个数码管显示。所以只要改变数位控制字的初值，就可以从不同的数码管开始显示。试着选用若干个数位控制字的初值来运行程序。

只要改变延时程序的循环次数的设置，就可以改变延时时间，从而改变数码管显示时间的长短。

改变D的初始值，就可以控制循环的次数，或者程序中再设置一个外循环，就更容易控制循环的次数。

步骤：

（1）运行上述程序，估算一下每个位置的停留时间。

（2）改变显示时间，重复运行程序，记录实验现象。

（3）改变显示的字符和显示的起始位置，重复运行程序。

（4）改变显示变迁次数，重复运行程序。

（5）设置外循环，重新运行程序。

记 录 表 格

显 示 字 符	显 示 时 间	字 符 移 动 次 数	显示字符的起始位置
“8”			
“2”			
“4”			
“0”			
“A”			
“F”			

2．编一个8位带符号数的加法程序，并将运算结果显示在数码管上。

（1）编程思路：

①给出一串八位带符号数，送入某段存贮区。

②检查每一个数的符号位。

③正负数分别求出累加和。每一种累加和都要两字节长。

④对负数求和的方法：

a．先取补把数值部分变为绝对值。

b．绝对值求累加和。

c．将累加和取补。

⑤求正负数的累加和。

⑥调用监控程序中的子程序，把累加和的两字节送入相应的显示缓冲区。

⑦利用监控程序中的显示程序，将运算结果显示在数码管上。

（2）流程图

根据上述思路可得如图1.27所示的流程图

（3）要利用监控程序中入口地址为00F4的显示程序，则必须把要显示的数预先送至显示缓冲区。6个数码管从左到右对应的显示缓冲区为：DISMEM（2FF7）、DSMEM 1（2FF8）、DSMEM 2（2FF9）、DSMEM 3（2FFA）、DSMEM 4（2FFB）、DSMEM 5（2FFC）、DSMEM 6（2FFD）、DSMEM 7（2FFE）。

而且，每一个显示管只能显示一位十六进制数，所以，一个存贮单元或一个寄存器

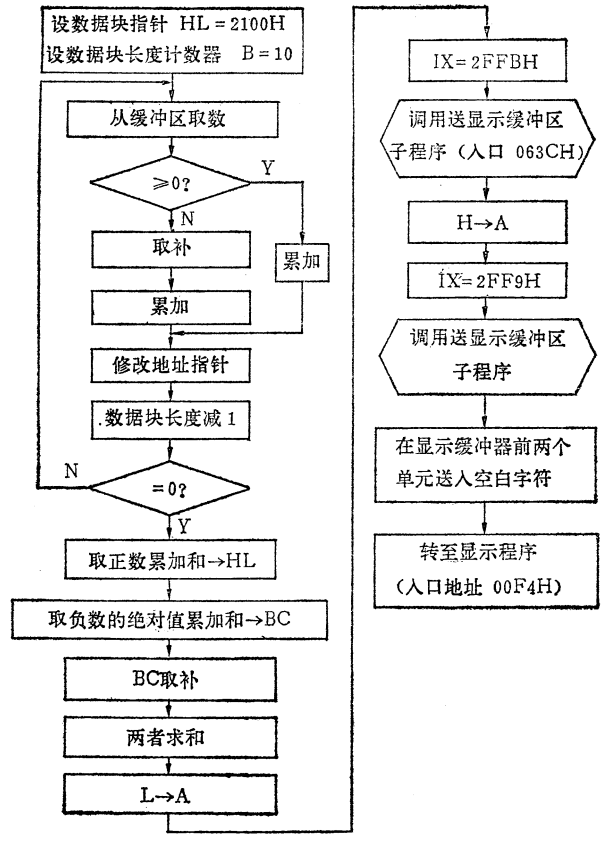


图1.27　显示若干个带符号二进制数的和的程序流程图

中的内容，在送入显示缓冲区前，必须把高 4 位和低 4 位分开，拆成两个字节。监控程序中入口地址为 063CH 的子程序就能完成这样的任务。但在调用此子程序之前，必须把要显示的字节送至累加器A；把缓冲区的地址送至变址寄存器 IX。子程序返回后，就能把A中的高 4 位和低 4 位拆开，且送至指定的显示缓冲区。子程序如下：

```
；入口地址为 063CH
        LD      B，A
        AND     0 FH
        LD      (IX＋1)，A
        LD      A，B
        SRL     A
        SRL     A
        SRL     A
        SRL     A
        LD      (IX＋0)，A
        RET
```

（4）上进的累加和为两个字节，则要用 4 个数码管显示，若我们用右面 4 个。而显示程序每执行一次，从最左面的显示管开始依次显示一遍。所以，在最左面两个显示管

的显示缓冲区中应存放显示空白的字符。而显示空白的字符应是什么呢？

从前面分析的显示程序中我们知道，若要显示一位 16 进制数，在显示缓冲区中存放的是此数本身，但真正送给字形锁存器的，却是此数的字模编码，这样的转换是由程序中搜索字模表来实现的。字模表是按 16 进制数的顺序存放的，在程序中把字模表的首地址送 HL，而从显示缓冲区取出的要显示的数送 E，D 清 0，通过 ADD HL，DE，就使 HL 指向要显示的数的字模编码的地址，再由 LD A,(HL)就把字模编码取至 A 中，然后由 A 输出给段形锁存器。显示管就可以显示相应的数。现在要在前两位显示空白，则必须把空白字符的字模编码 7 FH 也顺序放在字模表中，现在把它紧接在'F'的后面。则要从字模表中取出空白字符的字模编码，E 中的值必须是 10H，这就是空白字符应存放在显示缓冲区中的值。

二、预习要求

1．认真阅读本实验指导书。

2．认真阅读本部分第二节中的显示程序分析部分，确实弄懂显示原理和显示程序。

3．参照流程图编出源程序和翻译成机器码。

4．准备好数据和记录表格。

三、报告要求

1．整理好运行正确的源程序。

2．整理好记录表格。

3．求和程序至少做三组，每组有 10 个以上的数据，整理好这些数据以及运行后的显示结果。

*实验七　时钟程序

一、实验要求

1．编制一个用软件产生时、分、秒实时时钟的程序。

2．此程序从给定的起始值开始计时。

3．用 6 个数码管始终显示时、分、秒。

二、编程思路

1．编制一个定时为 1 秒的软件循环程序。

2．把预置的时、分、秒的起始值分别放至一个寄存器中。

3．调用 1 秒定时。

4．当定时到，秒值加 1，然后判断是否为 60，不到 60，则转至显示程序，显示完又转至调用 1 秒定时。

5．若秒值增至 60，则置秒值为 0，分值加 1，再判断是否为 60，不到 60，则转至显示程序。

6．若分值增至 60，则置分值为 0，时值加 1，再判断是否为 24，不到 24，则转至显示程序。

7．若时值增至 24，则时值置为 0，再转至显示程序。

8．程序不断地循环。

根据上述思路，可得如图 1.28 所示的程序流程图。

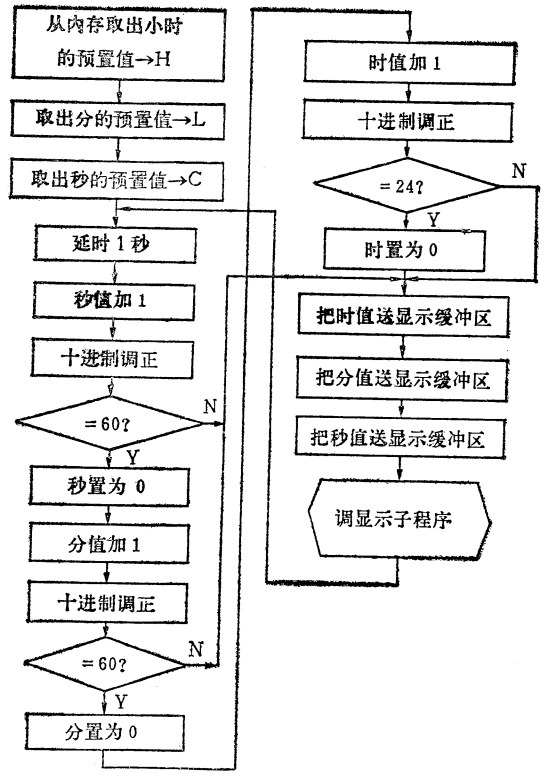


图1.28　用软件产生实时时钟且显示的程序流程图

要使程序能从显示程序循环回来，就不能直接利用 00F4 的显示子程序。因此程序段是在显示程序和键盘输入及译码程序之间不断地循环，无法返回用户程序。我们可以把 00F4 至 0122H 的显示程序，用数据块传送指令，搬至 RAM 中，然后把它的最后一条指令 JP DECKY 改为 RET，这样就可以作为子程序来使用。

三、预习要求

1．认真阅读本实验指导书。

2．进一步熟悉和掌握显示程序。

3．参照流程图，编出源程序并翻译成机器码。

四、报告要求

整理好运行正确的源程序。

*实验八　数据块的排序和搜索程序设计

一、实验内容

1．按实验三中的方法，在存贮区中建立0-99共一百个数，且要求奇数为负，偶数为正。

2．用气泡分类（排序）法，对这一百个带符号数按大小次序进行排序。程序的流程图和程序可参照教材中给定的。

3．用线性搜索的方法，搜索－29、＋70、－50、＋53这四个数，程序中要有搜索到此数，和搜索不到此数（即数据中无此数）的搜索次数计数器，并记录每个数的搜索次数。程序的流程图和程序可参照教材中的例子。

4．用对分搜索方法，对上述给定的四个数进行搜索。程序中也要有搜索次数计数器，并记录每个数的搜索次数。程序的流程图和程序也参考教材。

二、预习要求

1．认空阅读本实验指导书。

2．认真复习教材中的有关部分，确实弄懂和掌握几个程序。

3．编出各个源程序，并翻译成机器码。认真做好存贮器的分配。

4．估算两种搜索方法搜索给定的四个数的搜索次数，做到心中有数。

三、报告要求

1．整理出运行正确的各个源程序。

2．整理所记录的数据。

3．对线性搜索和对分搜索这两种搜索方法进行分析和比较。

*实验九　EPROM编程和校验

一、实验目的

1．学习和掌握对EPROM编程的方法。

2．初步掌握较验存贮器工作是否正常的方法。

3．学习和掌握运行已在PROM1中的程序的方法。

二、实验内容

1．用程序在RAM中建立0-FF共256个数据。

2．用下面要介绍的对EPROM编程的方法，把这256个数据，写入到EPROM中去。

3．编一个程序对已编程后的EPROM中的内容读出来加以检验，检查编程是否正

确，若有错，则把出错单元的地址记录下来。程序的流程图如图1.29所示。

4．用数据块传送程序，把TPBUG-A整个传送至RAM中，再把RAM中的 TPBUG-A程序写入到EPROM中（用另一块片子）。编程通过后，把EPROM插至 PROM1 位置，再把开关S2 打至PROM1 位置，复位后应直接运行PROM1 中的程序，但此时在PROM1 中的仍为TPBUG-A，故还应显示提示符 P，然后试着用各种监控命令，检查运行是否正常。

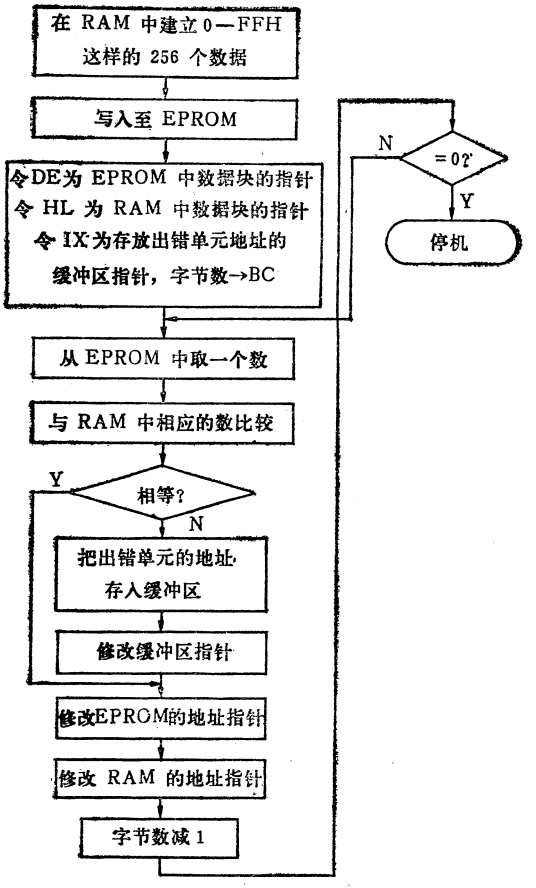


图1.29　写和校验EPROM的程序流程图

三、预习要求

1．认真阅读本实验指导书，特别是关于写入EPROM 的方法，要确实弄懂掌握。

2．编好各个源程序，并翻译成机器码。

四、报告要求

1．整理各个运行正确的源程序。

2．总结对EPROM 进行写入的方法。

3．总结校验内存贮器的方法。

4．PROG键（EPROM PROGrammer EPROM 编程键）的使用说明。

在TP801-A 中，可以将 RAM 中（起始地址为 2000H）的程序或数据，写入到插在 PROM2 位置上（起始地址为 1000H）的 EPROM 中去。所用的 EPROM，可以是 2758（1K字节），也可以是 2716(2K 字节)。

对EPROM进行写入，必须要有一个 25V ± 1V，30mA 的辅助电源。把此电源接至印刷电路板上标有＋25 V标记的焊点上。

对 EPROM的写入过程如下：

1．把需要写入的EPROM中的原有内容擦除干净，使各个单元的内容全为FFH。擦除方法是用紫外线灯照射EPROM 上方的石英玻璃窗，让紫外线透过窗口进行照射。所用的紫外线的波长为 2537A，照射强度为 12000μW/cm^2，照射 能量 为 15W·sec/cm^2，照射距离为 2.5-5.0cm，照射时间为20-40分钟。通常有专用的EPROM 擦除装置。

若是从未使用过的新片子，则可免去这一步。

2．在关闭电源的情况下，将EPROM片子插入至PROM2 位置。

3．合上＋5V和＋25V电源。此时应能按正常情况运行。即合上电源，按复位按钮，应出现提示符“P”。应能使用各个监控命令。

4．装入和运行程序，在RAM中建立要写入到 EPROM 中的数据块（必须从 2000H 开始)。

5．数据块准备好后，按 MON 键返回监控。

6．用键盘输入四个十六进制数字表示要写入EPROM的字节数（高位在前)。

7．将开关S3置于PGM位置，按下PROG 键，此时显示器变暗，开始进行写入。每个字节的写入约需 52ms。

8．写入完毕后，有两种可能：

（1）若重新出现提示符 P，则表示写入过程正确，且经过检验 与 RAM 中 的 内容一致。

（2）显示六个数字，则前四个为 EPROM 中第一个出错的地址，右面两 个为此单元的内容。按NEXT 键，则监控程序继续进行校验，若以后无错，则显示提示符 P；若有错则显示下一个出错的地址和内容。连续使用 NEXT 键，可以找到各个出 错的 地址和内容。

写入出错的原因主要是片子未擦干净，可以再次擦除，再次编程。

9．编程正确以后，将S3开关置于READ位置，接着进行上述的实验。

第二部分　微型计算机系统实验

利用单板计算机做汇编语言的程序设计，对于初学者来说，比较直观，操作方便，特别是程序的调试非常简单、方便，而且读者自己做一些手工汇编也是有益的。但是，单板机没有汇编程序，当程序较长时，手工汇编就很麻烦，而且容易出错。所以，有条件的还要做微型计算机系统的实验。此外，经过系统实验，对整个系统会有进一步的了解，特别是对磁盘操作系统的命令，操作系统所提供的支持会有进一步的了解，也便于用系统调用来编制用户程序。

目前，我国的微型计算机系统以Z80-CPU为核心的仍是主导。Cromemco系统原是我国引进较早，数量较多的一种微型计算机系统（以后就有别的机型）。另外，Apple Ⅱ系统因价格便宜，性能价格比也是较好的，近一、二年来也有相当数量的引进，而且在教育部门会更进一步得到普及。虽然Apple Ⅱ的CPU是R6502，但Apple Ⅱ机配上Z80 CPU卡，就可以引导和运行CP/M操作系统，也就可以运行Z80的汇编语言。所以，在这一部分我们以Cromemco和Apple Ⅱ这两种系统为主，介绍如何运行Z80的汇编语言的源程序。

第一节　CROMEMCO系统Ⅲ介绍

一、CROMEMCO系统Ⅲ的主要指标

- 中央处理器：4MHz的Z80A。
- 周期时间：250ns
- 最小指令执行时间：1μs。
- 指令系统：158指令，包括78条Intel8080的指令。
- 总线标准：工业标准S—100总线。
- 磁盘驱动器数：四个（每个驱动器配两个磁盘）。
- 盘存贮容量：8英寸盘每盘为256K字节（双面盘512K/盘）；5英寸盘每盘为80K字节（双面盘为181K/盘）。
- 串行接口：RS-232或电流环，波特率：110—76800。
- 并行接口：8位的TTL电平。
- 存贮容量：64K字节。多用户可达512K字节。
- 电源：＋8V30A，＋18V15A，－18V15A。

二、CROMEMCO系统的结构

主机主要是由四块印刷电路板构成，它们是：

（一）中央处理器板——ZPU板

它的核心是一片Z80A CPU，加上一些附加的中小规模集成电路构成。其功能是：

1．能与S—100总线相连接。S—100总线是根据8080制订的总线标准，与Z80的引线有一些差别。通过增加一些附加电路能使Z80的引线信号转换成S—100总线所要求的有关信号。

2．时钟频率的选择。根据需要可以通过开关及相应的电路来选择4 MHz或2 MHz的时钟频率。

3．自动跳转。在无控制面板操作的情况下，可实现在电源合上后，机器自动转移到存贮器一个固定地址，并执行程序。转向的地址由印刷板上的开关状态（16个位置）来决定。通常是自动跳转到地址为C000H的RDOS，然后由RDOS再引导CDOS（CDOS为一个磁盘文件，由RDOS把它调入内存，然后整个系统由CDOS管理）。

4．等待（WAIT）状态的选择。由于CPU的速度与RAM的速度不匹配，如果所选用的RAM速度低，就要求CPU在存贮器“读”（包括“取指”）/写的周期中插入T_w状态，板上有等待状态的选择功能，可在0—3个T_w周期之间选择。另外，还可选择在M_1周期（因取指周期对存贮器要求更高）是否需要附加一个T_w状态的选择功能。

（二）随机存贮器板——64KZ板

64KZ的核心是64K字节的RAM阵列，通常是用32片TMS 4116-15 16K×1 bit的动态RAM片子构成。

主要的外围电路有：

1．体选择。在板上有体选择（64K为一体）、块选择（以32K字节为一块）和工作方式选择三组开关，允许存贮器最多可扩展到8×64K=512K字节，以适应于多用户方式。

2．由40MHz时钟振荡器及8位时序移位寄存器，产生RAM所需要的控制时序信号。

3．“周期启动译码”逻辑。在满足规定条件时，产生启动、读、写等信号。

4．地址寄存器、数据输入寄存器、数据输出寄存器等。

5．刷新控制逻辑，控制在CPU运行时或暂停工作时，产生刷新信号。

此板与S—100总线兼容，可插到S-100总线上，与8080A及Z80A CPU均可配合工作。

（三）打印机接口板——PRI板

这是一个典型的并行传送的接口电路板，它允许CROMEMCO 3703或3779点阵式打印机和3355A全格式字符打印机同时工作。它用中断方式与CPU交换信息。

（四）磁盘控制器板——4FDC板

它不但是磁盘控制器（Floppy Disk Controller），同时包括I/O接口和固化的RDOS。它主要由三块大规模集成电路构成。

1．FD1771。这是磁盘控制器/格式化形成器电路，它能控制5英寸或8英寸磁盘，最多可带四个驱动器。

2．TMS5501。这是一个多功能的I/O接口电路，它的并行输入端口已用于磁盘与CPU交换信息；它的串行端口用于CPU与键盘显示终端接口。

3．1K字节的PROM 2708，用于存放RDOS。RDOS包含一个初始引导程序，它

引导存于磁盘上的CDOS。RDOS还包含有一个系统的监控程序（后面介绍）。

CROMEMCO系统Ⅲ的方框图如图2.1所示。

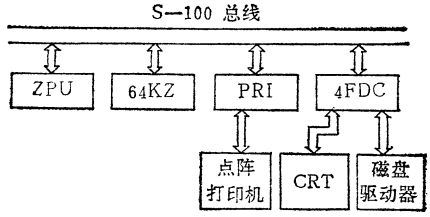


图2.1 CROMEMCO系统Ⅲ方框图

第二节 磁盘操作系统命令介绍

一、CROMEMCO系统的管理

任何一个计算机系统，小至一个单板机或单片计算机，大至大型甚至巨型计算机，都要在一个程序的管理之下，才能正常运行。

Cromemco系统是在RDOS和CDOS的管理下工作的。

RDOS是（Resident Disk Operating System）即常驻的磁盘操作系统，占1K字节，固化在4FDC板上的PROM 2708中，入口地址为C000H。

RDOS的功能分为两大方面：一个方面是在RDOS中包含一个初始引导程序，它可以把放在磁盘0道1扇区的引导程序1读入内存。再由引导程序1把在磁盘系统区的引导程序2读入内存，再由它把在磁盘文件区的CDOS引入内存，然后系统就在CDOS的管理下运行。另一个方面是包含一个系统监控程序，具有14条命令，它有几条与通常的监控和调试程序相似（与TPBUG-A相似）。这些命令具有显示和修改内存单元的内容，传送和比较内存块，转移程序控制，写数据到输出通道，改变串行通道的波特率等功能。此外，还有选择磁盘驱动器，寻磁道，读磁盘和写磁盘等有关磁盘操作的命令。

RDOS程序小、功能有限。所以，系统通常是在CDOS的管理和支持下运行。CDOS（Cromemco Disk Operating System）是Cromemco磁盘操作系统的缩写，它的主要用途是实现磁盘文件的输入和输出，它允许用户用文件名建立和处理顺序的和随机的磁盘文件。能实现对命名文件的管理，允许建立、修改、打开、关闭、读出和写入以及清除用户文件。通过CDOS的控制，可对文件内容进行检查，把磁盘文件名列成表，打印ASCII码文件、保存文件、对老文件重新命名及对可执行文件投入运行等。

CDOS还对系统的输入输出设备进行管理。它管理键盘输入、控制台、打印机和输出系统。

CDOS是在8位微型机的典型操作系统CP/M的基础上发展的，它是CP/M向上兼容的系统。故凡在CP/M上能用的命令和系统调用都可在CDOS上运行。

CDOS的结构与CP/M类似，整个操作系统也分成三层：

1. IOS（输入输出）系统

这是操作系统与硬件的输入输出设备以及磁盘驱动器的接口，用来管理控制台（键盘显示终端）、打印机、穿孔机、纸带阅读机和磁盘的输入输出。它主要包含这些设备的驱动程序。

2．DOS（磁盘操作）系统

这是CDOS的核心，负责管理、建立、打开、读和写磁盘文件。它由几十个子程序构成。这些程序也可以供用户程序调用，统称为系统调用。同时它也承担调用用户程序和编辑控制台的输入。

3．CONPROC（控制台处理）程序

这是操作系统与用户的接口。CDOS（以及CP/M）通常提供7条内部命令，用户可直接打入命令名来调用。操作系统还提供了一些实用程序，如编辑程序（EDITOR）、汇编程序（ASMB）等，它们以可执行的机器码形式（扩展名为·COM）存放在磁盘文件区，称为外部命令。用户可以在CDOS下直接打入这些实用程序（外部命令）名（不带扩展名·COM）来加以调用。此外，用户的程序只要经过编辑（形成源程序）。汇编或编译、连接和存盘以后，以机器码的形式（扩展名为·COM）存放在盘上，那么也可以与外部命令一样，在CDOS下也可以直接打入程序名（不带扩展名·COM），从盘上调入内存且立即执行。

存放在盘上的信息块称为文件，一个文件是具有名字（文件标记或文件标识符）的一维连续的字符序列或字序列。

CDOS对文件名的规定如下：

1．一个单义文件引用名，包括一个文件名称和一个任选的磁盘驱动器标识符和一个扩展文件名。其格式为：

〔×:〕文件名〔.ext〕

其中：

×：是任意一个磁盘标帜符，它说明文件在哪个驱动器上，它的合法值为A、B、C、D。

文件名：由多至8个可打印的ASCII字符组成（除掉字符$ * ? = / • : ,“空格”）。

ext：是由1到3个字符组成的扩展文件名，扩展名可有可无。常用一些扩展名来说明文件的类别。

2．多义文件引用名

星号（*）和问号（?）可以在文件名或扩展文件名中用来代替字符，以便建立一个多义文件引用名，多义文件引用名的格式和单义文件引用名相同。

星号可以代替从它所占的位置向右，直到下一个定界符（即句号“。”，问号“?”，或回车）号之间的任何字符。

问号可以代替它所占的位置上的任何单个的字符。

例：下面是一些单义文件引用名：

PROG · FOR

PROGRAM 1 · FOR

PROGRAM 2 · FOR

PROG · REL

PROGRAM1·REL
PROGRAM2·REL

下面是多义文件引用名的例子：

多义文件引用名	涉及的文件
PROGRAM?·REL	PROGRAM1·REL
	PROGRAM2·REL
PROG*·FOR	PROG·FOR
	PROGRAM1·FOR
	PROGRAM2·FOR
PROGRAM1·*	PROGRAM1·FOR
	PROGRAM1·REL
·	所有文件
*·FOR	PROG·FOR
	PROGRAM1·FOR
	PROGRAM2·FOR

二、CDOS的内部命令

（一）系统的启动

CDOS有两种引导方式，一种是由RDOS用命令引导CDOS；另一种是自动引导方式。

1．自动引导方式

当把机器合上电源，插上系统盘以后，按复位按钮，系统就自动引导CDOS（实际上仍是由RDOS引导CDOS，但不用操作员干预），在经过初始化以后，在屏幕上出现CDOS的提示符：

A．

说明现在系统在CDOS的管理之下，可以运行CDOS的各种命令。

2．由RDOS引导CDOS

系统上电，插上系统盘，复位以后，系统进入RDOS，出现RDOS的提示符：

；

说明现在系统在RDOS的管理之下，可以运行RDOS的各种命令。打入B命令就可以引导CDOS

；B

在CDOS引入后，就出现CDOS的提示符：

A．

此时系统就在CDOS的管理之下了。

3．在AppleⅡ机上若有Z80-CPU卡，则在上电，插入AppleⅡ的CP/M系统盘以后，就会自动引导CP/M操作系统，出现的提示符为：

A＞

此时系统就在CP/M的管理之下，可以运行CP/M的各种命令（大部分命令与

CDOS相同，下面我们介绍的命令，在两种系统中都能运行)。

(二) 命令的格式与分类

CDOS (或CP/M) 的每条命令占据一行，叫做命令行，以回车键作为命令行结束的标志。

命令行的一般格式为：

盘符：命令/开关　文件标记1＝文件标记2　参数〈CR〉

盘符（磁盘标识符）的合法值为A、B、C、D。若是内部命令，则不用盘符；若是外部命令，当文件在当前磁盘驱动器上时，也不用盘符。

命令由不大于8位的可打印字符（$ * ? = / . :空格除外）组成。有的命令可以带有开关。

文件标记如前所述。在用改名、复制等命令时就需要有两个文件标记。

从程序的驻留角度来看，命令可分为内部命令和外部命令两大类：

- 命令
  - 内部命令
    - ATRIB　(属性命令)
    - BYE　(返回RDOS命令)
    - DIR　(显示目录命令)
    - ERA　(删除命令)
    - REN　(改名命令)
    - SAVE　(存盘命令)
    - TYPE　(打印命令)
  - 外部命令
    - XFER　(传送命令)
    - DUMP　(打印记录命令)
    - WRTSYS　(传送系统区文件命令)
    - INIT　(磁盘初始化命令)
    - STAT　(检查系统状态命令)
    - 22PCOPY　(复制5英寸软盘命令)
    - 64KTEST　(测试内存命令)
    - DEBUG　(调试命令)
    - EDIT　(编辑命令)
    - ⋮

内部命令的程序较短，包含在CDOS之内，随着CDOS一起被引导到内存。内部命令调用方便，它的执行并不影响用户区。

而有的命令程序本身就很长，例如INIT命令占8K字节，DEBUG命令占10K字节等，这些命令若与CDOS一起引入内存，则将占去大量的RAM空间（甚至在RAM中放不下），故把它们单独地作成文件，放在磁盘的文件区，只有在使用时才调入内存，使用完后就退出内存。通常这些命令文件可以在磁盘目录上找到。这些就称为外部命令，或实用程序 (Utility)。

(三) 主要的内部命令的使用

1. 显示目录命令：DIR

在磁盘上有一个文件叫做目录文件，它一方面登记盘上所有文件的文件名、扩展名，以及文件的一些特性，文件的长度等；另一方面由它把文件与物理的盘上的存贮空间联系起来。通常，在我们使用一个系统时，总是首先希望了解一下，此系统提供给我们一些什么样的系统软件，这就可以通过显示系统盘的目录来了解。在使用一个盘时，也常常希望了解此盘上有哪些文件，这些文件的特征，以及盘上已使用了多少空间，还剩多少自由空间可供使用等等，这也可以通过显示目录命令来了解。

显示目录命令的格式：

$$\mathrm{DIR}\begin{cases}\text{空}\\ \text{〔×:〕}\\ \text{文件标记}\end{cases}$$

第一种只打入DIR就可以，表示显示现行盘的全部文件目录；第二种是显示由盘符×（盘的符号）指定的那个盘的全部文件目标；第三种是显示由文件标记所指定的盘上的某一或某些文件的目录。

例：

A. DIR

CDOS	COM	12K	EWS
CDOSGEN	COM	28K	EWS
LINK	COM	8 K	EWS
BASIC	COM	19K	EWS
ASMLIB	REL	2 K	EWS
ASMB	COM	12K	EWS
DEBOG	COM	10K	EWS
EDIT	COM	7 K	EWS
A11	Z80	1 K	U

＊＊＊ 9 FILES, 11 ENTRIES, 99K DISPLAYED, 72K LEFT＊＊＊

其中，A. 是CDOS的提示符；DIR是打入的命令行（以后我们以下面划横线，表示是用户打入的内容）；其余的是系统对DIR命令的响应。

最左边一列是文件名，第二列是文件的扩展名*，第三列是文件的长度，第四列是文件的属性。E表示删除保护（即此文件不能删除），W表示写保护，R表示读保护，U为用户文件，S为系统文件。

最后一行是小结，指出盘上共有多少个文件，占用多少个目录项（一个文件长度≤16K占用一个目录项，大于16K就要占用两个或多个目录项），已用去的磁盘空间和剩下的自由空间。

若现行盘为A盘，要了解B盘上的文件情况，则可打入

---

* 在CDOS中常用扩展名来表示一个文件的特征。如：.COM表示是以机器码形式存放，可执行的文件；·BAS表示是BASIC的源程序；·FOR表示是FORTRAN的源程序；·Z80表示汇编的源程序；·REL表示经过汇编后的浮动文件；·PRN表示可打印的文件等。

A. DIR B:

TIMER	Z80	1K	U
GETDN	Z80	1K	U
INDN	Z80	1K	U
INDN	BAK	1K	U
INDN	REL	1K	U
INDN	PRN	2K	U
INDN	COM	1K	U

*** 7 FILES, 7 ENTRIES, 8K DISPLAYED, 163K LEFT ***

若要了解某一类文件的情况，则可以在DIR命令后打入文件标记（通常为多义文件名）。

2. 删除文件命令：ERA

当盘上的文件已经不需要时，或盘上的空间已经不够用时，就可以删除不要的文件。

命令格式：

ERA 文件标记

命令执行以后，就把文件标记所指定的文件删除了。这可以由在命令执行前，和命令执行后，分别用DIR命令来显示目录以得到证实。

例如B盘上7个文件（见前面）经删除就如下：

A. ERAB: INDN. BAK

A. DIR B:

TIMER	Z80	1K	U
GETDN	Z80	1K	U
INDN	Z80	1K	U
INDN	REL	1K	U
INDN	PRN	2K	U
INDN	COM	1K	U

*** 6 FILES, 6 ENTRIES, 7K DISPLAYED, 164K LEFT ***

这就是要把B盘中的文件INDN·BAK删除掉。

要注意：删除命令不能删除具有E（删除）保护属性（即第四项）的文件。

在删除命令中也可以用多义文件名来删除多个没有E保护的文件。

例：

A. ERA B: INDN. *

A. DIR B:

TIMER	Z80	1K	U
GETDN	Z80	1K	U

*** 2 FILES, 2 ENTRIES, 2K DISPLAYED 169K LEFT ***

删除命令中用多义文件名时要慎重，不要把需要的文件也给删除了。

3. 改名命令：REN

当需要用新的文件名代替老的文件名时，就可以用这改名命令。

例如，通常在系统上要建立和运行一个汇编语言（FORTRAN等高级语言也如此）的程序，首先要调用编辑程序，建立一个源程序如INDN. Z80；这个源程序只要再次调用过编辑程序加以修改，则修改后的源程序仍用原来的名和扩展名，而把原来的程序变为后备程序，就建立了一个INDN. BAK；源程序必须由汇编程序进行汇编，汇编后又建立了两个文件，一个是浮动文件INDN. REL供连接和存盘用，另一个是供打印用的文件INDN. PRN；要运行此程序，还必须把浮动文件经过连接和存盘（这些过程我们在后面详细介绍），形成可执行的机器码文件INDN·COM。若程序经过运行证明是正确的，我们通常可能需要在盘上保存源程序（扩展名为·Z80）和机器码程序（扩展名为·COM），而其他的程序就可删除。若我们要用多义文件名来一次删除所有不需要的文件，这时，就首先要把源程序和机器码程序先改名为别的程序。

改名命令的格式为：

REN　新文件标记＝老文件标记

注意：REN命令只能对现行盘符起作用，因此，新旧文件标记中不应带有盘符。若要实现上面所提的要求，先可以用下面的命令，改变现行盘。

A. B:

B.

出现提示符B. 表示现行盘已改成B盘了。然后可以输入以下命令：

B. REN INPTDN. Z80=INDN. Z80

B. REN INPTDN. COM=INDN. COM

B. ERA INDN. *

B. DIR

TIMER	Z80	1K	U
GETDN	Z80	1K	U
INPTDN	Z80	1K	U
IMPTDN	COM	1K	U

*** 4 FILES, 4 ENTRIES, 4K DISPLAYED, 167K LEFT ***

在REN命令中，若新文件标记与盘目录中的某一文件相同，则系统拒绝执行这一命令。

具有删除保护的文件不能改名。

4. 打印命令：TYPE

这条命令可以把用ASCII码书写的文件送CRT显示或同时送打印机打印。

在CDOS和CP/M中，打印机与CRT的连接是由控制键CTRL-P（即同时按CTRL键和字母P键）控制的。当按打奇数次CTRL-P键后，打印机与CRT就连接上，则在CRT上的显示字符，同时在打印机上打印出来；若打入偶数次CTRL-P键，则打印机与CRT脱开，在CRT上显示的字符，不在打印机上打印。

打印命令的格式：

TYPE　文件标记

例：在CP/M操作系统下用TYPE命令打印汇编语言的源程序：

```
A>TYPE GETCH. Z80
GETCH:  PUSH    BC
        PUSH    AF
        LD      C, 1
        CALL    5
        CP      03H
        CALL    Z, 0
        LD      B, A
        POP     AF
        LD      A, B
        POP     BC
        RET
PUTCH:  PUSH    BC
        PUSH    DE
        PUSH    AF
        AND     7FH
        LD      E, A
        LD      C, 2
        CALL    5
        POP     AF
        POP     DE
        POP     BC
        RET
        END
```

也可以用TYPE命令打印汇编后的·PRN文件。

在CP/M下的例子：

```
A>TYPE A11·PRN
CROMEMCO Z80 Macro Assembler version 03.07
*** A11 ***
0000' 3E0A          0001  START:  LD    A, 10
0002' 0614          0002          LD    B, 20
0004' 80            0003          ADD   A, B
0005' 5F            0004          LD    E, A
0006' 212600'       0005          LD    HL, DATA
0009' 77            0006          LD    (HL), A
000A' 112B00'       0007          LD    DE, DATA1
000D' 3E30          0008          LD    A, 30H
000F' ED6F          0009          RLD
0011' 12            0010          LD    (DE), A
```

```
0012′ 13          0011          INC   DE
0013′ ED6F        0012          RLD
0015′ 12          0013          LD    (DE), A
0016′ 212D00′     0014          LD    HL, DATA1+2
0019′ 3624        0015          LD    (HL), '$'
001B  112B00′     0016          LD    DE, DATA1
001E′ 0E09        0017          LD    C, 9
0020′ CD0500      0018          CALL  5
0023′ C30000      0019          JP    0000H
0026′ (0005)      0020  DATA:   DS    5
002B′ (0005)      0021  DATA1:  DS    5
0030′ (0000')     0022          END   START
Errors                :
Range Count           :
Program Length  0030 (:)
```

使用TYPE命令要注意以下几点：

① 此命令只能用于打印ASCII码文件。若用于显示或打印非ASCII文件，将得到杂乱无章的结果。

② 在打印过程中，键入任何字符都将中止打印，并返回CDOS。

③ 命令中的文件标记不能用多义文件名。

④ 对于具有读保护的文件，此命令拒绝执行，且返回CDOS。

CDOS和CP/M中的外部命令，可以直接打入命令名（不要扩展名·COM），就可以从盘上调入和加以执行（后面我们结合使用来介绍）。

三、CDOS的系统调用

在CDOS和CP/M系统中，主要在DOS这一层，编制了几十个编了号的子程序。这些子程序是操作系统的核心部分，在系统中大量调用。这些子程序又可以供用汇编语言编程序的用户调用称为系统调用。这就是操作系统所提供的支持，可以大大简化用户程序的编制。

这些子程序包括：

1. 有关输入输出设备的管理子程序；
2. 有关磁盘管理的子程序；
3. 有关文件管理的子程序；
4. 其他子程序——如乘除法等。

（一）系统调用的格式

如上所述在系统中有几十个编了号的子程序，显然每一个子程序的入口地址不同，通常在调用子程序时，是把某一个子程序的入口地址直接放这调用指令中。这样，用户要调用各个子程序，就要记住各个子程序的入口地址，这是很麻烦的。而且，CDOS的版本号不同，或系统的配置不同，CDOS在内存中的分配是不同的，各个子程序的入口

地址也就有变化，这样，要记住入口地址就更麻烦了。为了方便用户，系统允许在调用子程序时，都以0005H为入口地址，即用CALL 5，而调用不同子程序，以各个子程序的编号（子程序的编号在不同版本和不同配置的CDOS中是不变的）来区分，所以，规定在调用前要把子程序的编号送入寄存器C，由系统根据编号，通过查表，转至各个子程序的入口。另外，有些子程序在调用前还需要一些条件，例如要设置一些输入或输出的缓冲区指针等等，这些条件就叫做入口参数。在子程序调用以后的结果会反映在一些寄存器中，这就叫做出口参数。所以，系统调用的一般格式为

```
BC←— P 1 ┐
DE←— P 2 ├入口参数
HL←— P 3 ┘
 C←—CW

CALL    5   调用

 A=   R 0 ┐
BC=   R 1 ├出口参数
DE=   R 2 ┘
HL=   R 3
```

1．入口参数

将调用的子程序的编号CW送至寄存器C，将参数P1、P2、P3送入BC、DE、HL寄存器对。但并不是每一调用都需要三个参数，通常只需要1—2个参数，或不需要参数。

2．调用

只一条指令CALL 5。

3．出口参数

调用后的结果放在一些寄存器中，有时还以寄存器A的状态来反映调用是否成功。

我们在这儿并不介绍所有的系统调用，只是从实验的需要，介绍几个有关输入输出设备的系统调用。

（二）控制台输入输出字符的系统调用

1．系统调用1是把键盘上输入的一个字符取到寄存器A中，取入的是该字符的ASCII码，最高位（D_7位）为0。

此调用的入口参数只有一个，即系统调用号→C。

此调用执行后，除了把键入的字符送至A中以外，还把此字符送CRT显示（若按过奇数次CTRL-P键，则打印机与CRT连通，结果也在打印机上输出）。

当使用这个系统调用时，若控制台没有键入字符，则程序就循环等待而不往下执行，所以此调用适用于一定会有键入的情况下。

若要测试是否有键入，就要用系统调用11。

2．系统调用11

此调用的功能是测试控制台是否键入一个字符。如果键入，寄存器A中的内容为—1（即FFH），否则为0。

有了这样的系统调用，就可以在程序执行中安排人工干预的出口。这样，在程序中

出现死循环的时候，可以由按CTRL-C来中止用户程序而返回CDOS。下面是一个能实现这样要求的程序例子。

```
IFBRK:  PUSH    BC
        PUSH    AF
        LD      C,11
        CALL    5
        JR      Z,GOON
        LD      C,1
        CALL    5
        CP      03H
        JP      Z,0000H
GOON:   POP     AF
        POP     BC
        RET
```

在保护了现场以后，先用系统调用11来了解是否有键入，若没有则返回；若有键入，再利用系统调用1来输入字符，然后比较是否为CTRL-C键（它的ASCII码为03H），若是则返回CDOS(入口地址为0000H)，若不是则返回主程序。

这个例子说明，系统调用11只起测试作用，在发现有键入时，仍要用系统调用1来输入字符。

3．系统调用2

这是向控制台输出字符的系统调用。入口参数是：系统调用号2→C，要输出的字符的ASCII码→E。此调用执行的结果是在CRT上显示要输出的字符，没有别的出口参数。

在E中的可以是一些非打印字符，当它们输出时，并不是显示而是起一些控制作用例如可以把回车与换行字符通过E输出，则可起到控制光标回车与换行的作用。

通常，在程序中要求从控制台键入时，可以先用系统调用2输出一个提示符，以要求输入。例如：

```
LD    E,'>'
LD    C,2
CALL  5
LD    C,1
CALL  5
```

(三) 控制台输入输出字符串的系统调用

1．系统调用10

此调用的功能是把从键盘上输入的字符串读进内存并在CRT上显示。

内存中准备输入字符串的地方称为字符串缓冲区。在调用前，必须把缓冲区的首地址送DE寄存器对，这就是一个入口参数。另一个入口参数是把系统调用号送寄存器C。

缓冲区的长度，由程序员决定，在调用前用程序或其他方法把它送至缓冲区的第一个单元。在此调用执行时，调用子程序统计键入的字符个数（以回车键作为输入字符的

结束，回车键不计入在内)，且把实际输入字符个数填至缓冲区的第二个单元，从第三个单元开始才依次存放输入的字符。若输入的字符个数小于预定的缓冲区的长度。则缓冲区的多余部分都填上00字节；若输入的字符个数多于缓冲区的长度，则超出的部分被略去。

2．系统调用9

此调用是把内存中的一个字符串，此字符串必须以字符$为结尾，送至CRT上显示($字符不显示)。调用前的入口参数为：调用号9→C，要输出的字符串的首地址→DE。

下面我们举一个例子来说明这两个系统调用的应用。

```
        ORG     100H
CR:     EQU     0DH         ；定义CR为回车符
LF:     EQU     0AH         ；定义LF为换行符
START:  LD      E,':'       ；以':'作为要求输入字符串的提示符
        LD      C，2
        CALL    5           ；输出提示符':'
        LD      DE,BUFFER
        LD      C，10
        CALL    5           ；从键盘输入字符串
        LD      HL,BLOCK    ；设置输出缓冲区指针
        LD      (HL)，CR    ；设置回车符
        INC     HL
        LD      (HL)，LF    ；设置换行符
        INC     HL
        LD      DE,BUFFER
        INC     DE
        LD      A,(DE)      ；取出实际输入的字符个数
        INC     DE
        LD      C，A
        LD      B，0        ；字符个数→BC
        EX      DE,HL       ；源地址为输入缓冲区→HL,目的
        LDIR                ；地址为输出缓冲区→DE，数据块
                            传送
        EX      DE,HL
        LD      (HL)，'$'   ；在输出缓冲区的末尾，添入'$'字
                            符
        LD      DE,BLOCK
        LD      C，9
        CALL    5           ；输出字符串
        HALT
```

```
                ORG     150H
BUFFER:
LENGPR:         DB      40H
LENGPC:         DS      1
LENGTH:         DS      40H
BLOCK:          DS      43H
                END     START
```

程序先输出一个提示符′：′(也可以用别的字符)，等待从键盘输入字符串。当从键盘上输入字符串以后，把输入的实际长度送入BUFFER＋1单元，从BUFFER＋2单元开始存放输入的字符串。然后设置输出缓冲区，先送入回车及换行符，使输出的字符串与输入的不重叠。接着用数据块传送指令，把输入的字符串传送至输出缓冲区，在输出缓冲区的最后添上字符′$′。最后用系统调用9输出字符串。

上述程序运行举例

```
A.EXAMPE
：THIS IS A EXAMPLE
THIS IS A EXAMPLE
```

其中，EXAMPE是上述程序的文件名，只要把源程序经过汇编和连接，以机器码的形式存盘，则可在CDOS(或CP/M)下，直接打入文件名（不要扩展名·COM)，调入程序且执行。执行时显示提示符，其后面是键入的字符串，下一行是程序的输出。

程序运行以后，内存缓冲区中的内容为：

```
0150  40  11  54  48  49  53  20  49  53  20  41  20  45  58  41  4D
0160  50  4C  45  00  00  00  00  00  00  00  00  00  00  00  00  00
0170  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0180  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0190  00  00  0D  0A  54  48  49  53  20  49  53  20  41  20  45  58
01A0  41  4D  50  4C  45  24  00  00  00  00  00  00  00  00  00  00
01B0  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
01C0  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
01D0  00  00  00  00  00
```

从0150H开始是输入缓冲区，第一个单元的内容40H是预定的缓冲区长度，第二个单元的内容11H是实际的输入字符的个数，从第三个单元开始就是输入字符串的ASCII码。由于实际输入长度，小于预定的缓冲区长度，所以，缓冲区的后面部分都为00。

从0192H开始是输出缓冲区，前两个单元是回车换行符，从第三个单元开始就是从输入缓冲区传送过来的输入字符串，接着是字符′$′。

第三节　建立和运行汇编语言源程序的过程

一、步骤

在微型机系统上建立和运行汇编语言源程序（除BASIC语言以外的其他高级语言的源程序也是这样的过程）的过程为：

1．调用文本编辑程序EDITOR，输入源程序和修改源程序，在磁盘中建立一个源程序的文件。

2．调用汇编程序ASMB，对源程序进行汇编，把助记符转换为机器码，把地址变为可浮动的相对地址，相对地址的起始值为0000H。

3．调用连接程序LINKER，对汇编后的浮动文件进行连接。连接程序一方面把源程序中所调用的程序库中的子程序与源程序连接起来，另一方面把浮动地址变为绝对地址。

4．用SAVE（存盘）命令，把连接后的文件以可执行的机器码形式存盘。

5．在CDOS下用文件名调入文件且执行。

下面我们较详细地介绍一下每一步中所用到的命令。

二、文本编辑程序EDITOR

文本编辑程序的主要功能是输入和修改源程序。在输入一个新文件的情况下，当退出编辑程序返回CDOS时，在盘上就建立起这个新的文本文件；在修改文件的情况下，当把文本修改完退出编辑程序返回CDOS时，盘上的文件就是修改后的新文件，而把修改前的文件变为一个后备文件亦保存在盘上。

编辑程序的命令较多，适当使用是非常方便和灵活的。对于初学者，我们只是简要地介绍一下以行编辑为基础的一些编辑命令。

（一）编辑程序的调用

不论是要输入和建立新文件，还是要修改文件，都要调用编辑程序。

调用的格式为：

A．EDIT　文件标记

（A＞ED　文件标记）

其中，A．（A＞）是DOS的提示符，EDIT(ED）是编辑程序的文件名。我们在下面要介绍的命令，对于CDOS和CP/M都适用，至于有一些区别，我们把在CP/M下的情况放在括号内。

若是汇编语言的源程序，文件标记中的扩展文件名必须是·Z80。

若给定的文件标记是磁盘文件目录中没有的新文件，则系统的响应如下所示：

```
A＞ED  B97.Z80
NEW  FILE
     :*
```

系统以NEW FILE表示是一个新文件，然后给出编辑程序的提示符'*'，表示现在系统处在编辑程序的管理之下，可以应用编辑命令。

若给定的文件标记，是磁盘目录中有的老文件，则系统不给出NEW FILE而直接输出编辑程序的提示符。

（二）编辑一个新文件

当我们调用编辑程序，系统给出NEW FILE和提示符'*'以后，就可以输入一个新文件。

编辑程序在内存中开辟出一个编辑缓冲区，现在这个编辑缓冲区是空的。

我们要用键盘向编辑缓冲区输入新的内容，必须使编辑程序工作在插入方式，可打入以下命令：

*I〈CR〉

□

打入I命令，使编辑程序进入插入方式，此时不出现任何提示符，只是光标□出现在下一行的开始（在CP/M下，出现行编号）。

此时就可以按照汇编语言源程序的格式，由键盘输入源程序。例如我们要输入以下的源程序：

```
LF:       EQU    0AH
CR:       EQU    0DH
START:    LD     B, 0
          PUSH   BC
LOOP:     LD     E, CR
          LD     C, 2
          CALL   5
          LD     E, LF
          LD     C, 2
          CALL   5
          POP    BC
          OR     A
          LD     A, B
          ADD    A, 1
          DAA
          AND    0FH
          LD     B, A
          PUSH   BC
          OR     30H
          LD     E, A
          LD     C, 2
          CALL   5
          LD     HL, 0FFFFH
DELAY:    DEC    HL
          LD     A, H
```

```
        OR      L
        JR      NZ,DELAY
        JP      LOOP
        END     START
```

在标号场与操作码场以及操作码场与操作数场之间的间隙，可以用空格键来实现，但比较好的方法是用控制键CTRL-I。它可以使场与场之间跳过固定的间隔，从而使程序排列整齐。

在插入方式下，以回车键（Return键）作为一行的结束。在某一行中，在尚未按回车键时，若发现这一行中有打入错误，则可以用RUB键（或DEL键，←键等擦除键），使字符返回到出错处，删去出错的字符，再接着打入正确的字符。若在按了回车键后发现有打入错误，千万不能在插入方式下企图进行修改，否则你为了要修改而打入的各种命令，都被认为是要输入的字符而送入编辑缓冲区，从而使程序面目全非，越改越乱。这时，只能在退出了插入方式后，才能用修改命令（在后面介绍）进行修改。

在用插入方式把源程序从键盘上输入完以后，要立即退出插入方式。

退出插入方式的命令：按ESC键或按CTRL-Z键。在按了上述键，屏幕上显示编辑程序提示符'*'以后，表示已退出了插入方式，这以后就可以用其它的编辑命令。

若我们在插入方式时，想输入前列的源程序，但由于键盘使用不熟练，错把源程序输入为如下情况：

```
LF:     EQU     0AH
CR:     EQU     0DH
START:  LD      B,0
LOOP:   LD      E,CR
        LD      C,2
        CALL    5
        LD      E,LF
        LD      C,2
        CALL    2
        POP     BC
        OR      A
        LD      A,B
        ADD     A,1
        ADA
        AND     0FH
        LD      B,A
        PUSH    BC
        OR      30H
        LD      E,A
        LD      C,2
        CALL    5
```

```
DELAY:    LD      HL,0FFFFH
          DEC     HL
          LD      A,H
          OR      L
          JR      NZ,DELAY
          JP      LOOP
          END     START
```

（三）编辑缓冲区指针

当我们用插入方式，把源程序输入完了，退出插入方式后，往往首先需要看看我们已输入至显示缓冲区的内容，对照一下输入过程是否正确。但是要显示哪一部分内容呢？或者发现有错要加以修改，则需要指出错在哪儿呢？这些都牵涉到缓冲区指针。

编辑缓冲区有三个指针：

1．TOP　它指出文件中第一个字符的位置

2．BOTTOM它指出文件中最后一个字符的位置。

3．字符指针（CHAR）它用于指示缓冲区中某一字符的位置。它在缓冲区中可以移动，可移至缓冲区中第一个字符之前，CHAR＝TOP；也可以移至最后一个字符之后，即CHAR＝BOTTOM。它不可见，但许多指令可以改变它的位置；许多指令与它的位置有关。

（四）基本的编辑命令

编辑程序有两种基本工作方式，一种是由I命令所引起的插入方式，这种方式只适用于从键盘输入文本，不能使用其它的编辑命令；另一种方式是命令方式。每一种命令都是在出现编辑程序的提示符'＊'后才能使用，命令以回车键作为结束标志，命令执行完后又出现提示符'＊'。

1．N命令

它将输入文件装入缓冲区。

当我们调用编辑程序来修改一个老文件时，虽然出现了编辑程序的提示符

```
A．EDIT  B97.Z80    〈CR〉
*
```

但此时编辑缓冲区中仍是空的，三个指针都指向缓冲区的顶部。必须用N命令，才能把指定的文件装入缓冲区。

```
＊N    〈CR〉
END OF INPUT FILE（在CP/M中没有这一行文字）
*
```

当再次出现提示符时，指定的文件已装入了编辑缓冲区，底部指针BOTTOM移至最后一个字符的后面，而TOP与CHAR两个指针仍指向顶部。

2．I命令

使用I命令，可进入插入方式，它可以在字符指针之前插入一个字符或一行或一段文本。插入完以后必须用ESC键或CTRL-Z键退出插入方式，在出现了提示符'＊'以后才能使用别的命令。

3．移动字符指针的命令

① 命令±B

命令＋B(或B)，使字符指针移至缓冲区的顶部，就是使 CHAR＝TOP。

命令－B，使字符指针移至缓冲区的底部，就是使 CHAR＝BOTTOM。

② ±nL

＋nL，使字符指针从当前位置往下（朝底部）移动n行，且放在该行的前面。

－nL，使字符指针从当前位置往上（朝顶部）移动n行，且放在该行的前面。

4．显示与打印命令

① ±nT

此命令打印字符指针之前（－）或之后（＋）n行，但不移动字符指针。

② ±nLT

这是一个组合命令，先把字符指针往前（－）或往后（＋)移动n行（命令±nL)，然后打印这一行（命令T)。

这个命令的缩写形式为±n。

③ 0LT

此命令把字符指针移至所在行的最前面，且打印该行。

④ 1LT〈CR〉的简化形式为只按回车键。它使指针往下移一行，且打印该行。

⑤ B＃T

这也是一个组合命令，B命令把字符指针移至缓冲区的顶部，＃号代表65535，T命令为打印，此命令从顶部往下打印65535行，实际上当CHAR＝BOTTOM时，打印就停止。所以，这是一个打印整个缓冲区的命令。

5．删除命令

① ±nK

此命令将删去字符指针之前（－）或之后（＋）的n行。且把下面的内容往上提使文本仍连在一起。

② ±nD

此命令删去字符指针之前（－）或之后（＋）的n个字符，也使文本连在一起。

6．变换命令S

此命令用来改变缓冲区中的一段文本。格式为：

＊S〈旧正文〉〈ESC〉〈新正文〉〈ESC〉〈CR〉

它将从字符指针位置开始寻找规定的旧正文，找到后就用新正文代替它，而其前后的内容都不改变。命令执行后可以用0LT命令来检查。

7．结束和取消编辑程序命令

① E

E即Exit。当经过编辑，或者是输入了一个新文件；或者是对老文件进行了修改。要把编辑缓冲区的内容存盘，而且要退出编辑程序返回DOS。就使用E命令。格式：

＊ E〈CR〉

A．

E命令把编辑缓冲区的内容存至扩展名为·Z80的文件中，而把盘上原来的文件变为扩展名为·BAK的文件。

② Q命令

当我们不慎把编辑缓冲区中的内容改乱了，我们不打算要这些内容，要返回DOS重新编辑，就可以使用Q命令。

Q命令不把编辑缓冲区的内容存盘，保留原输入文件，返回CDOS。但执行Q命令以后，编辑缓冲区被破坏了。所以，为了慎重，打入Q命令以后，编辑程序会提醒操作者是否真要执行此命令。

```
*  Q 〈CR〉
DISCART TEXT BUFFER, RESTORE INPUT FILE AND QUIT EDITOR(Y/N)?
```

(破坏编辑缓冲区，恢复输入文件和退出编辑程序是/否?)

只有当回答Y后才执行该命令。若回答为N，则重新回到编辑程序，以防因不慎按一下Q键而破坏编辑工作。

以上我们只介绍了以行编辑为基础的一些主要命令。但利用这些命令已经可以完成必要的编辑任务了。

(五) 编辑命令使用举例

我们假定已经调用过编辑命令要建立一个叫做B97.Z80的文件，而且已经用插入方式输入了源文件，插入完后退出了插入方式，而且用E命令把文件存盘，返回DOS。但是在插入时有若干处键入错误。正确的源程序和已建立的有错误的文件，在前面都已列出。下面我们就要用编辑命令进行修改。

1. 调用编辑程序，输入文件

```
A. EDIT B97.Z80〈CR〉
*  N
END OF INPUT FILE
*
```

当第二个提示符出现时，已经把盘上的指定文件，输入至编辑缓冲区了。为了证实一下缓冲区中是否是要输入的文件；或为了与源程序对照看输入过程有否错误，就要求显示（打印）缓冲区的内容。

2. 显示或打印缓冲区的内容

① 为了显示整个缓冲区，可以用命令

B # T

但是，在文本很长的情况下，用这个命令并不很方便。较合适的办法是一段一段查找。若所用的CRT的屏幕为24行的话，则可以20行为一段，一段一段地查找。

② 用命令20T

```
*  20T〈CR〉
LF:     EQU     0AH
CR:     EQU     0DH
START:  LD      B,0
LOOP:   LD      E,CR
          ⋮
        LD      E,A
```

```
                LD          C,2
```

我们就可以从第一行开始一行一行地与要输入的源程序相对照。在这个例子中，当我们查到第4行时，发现丢了一行。要把这一行补上去，这就要用插入命令I。但是，所插入的内容是插在字符指针之前，现在CHAR仍等于TOP，即在第一行之前，那么我们要在第四行之前插入一行，就必须先把字符指针移至第四行之前。

3．移动指针及插入一行

```
*  3LT〈CR〉
LOOP:       LD          E,CR
*
```

说明字符指针已移至第四行前面，就可以插入了。

```
* I〈CR〉
(打入CTRL-I) PUSH   BC〈CR〉
ESC *
```

当出现提示符说明退出了插入方式，可以用打入－1LT来检查新插入的一行是否正确。检查正确后就可以接着往下查对。发现第10行的CALL 5，错打成为CALL2，要加以改正。

4．删除及改正

第10行打错了，就可以先把它删去，然后再插入一行新的内容。删除与插入命令都与字符指针的位置有关。所以，先要把字符指针移至第10行前面。经过前面的修改，我们知道现在的字符指针是在第四行前面，要移至第10行前面，可用5LT命令。但是若不知道现在字符指针在何处，当然可以用0LT命令来显示，然后查找是第几行。但是，比较方便可靠的方法，是先把指针移至顶部，然后再移至第10行前，就可以用以下命令：

```
*  B9LT〈CR〉
                    CALL    2
*
```

指针已移至出错行，就可以用删除命令

```
* K〈CR〉
*
```

把出错的第10行删除了，但此命令又把下面的行提了上来，指针在下一行的前面，此时可以用插入命令，输入一行正确的内容

```
* I〈CR〉
(打入CTRL-I)   CALL  5 〈CR〉
ESC   *
```

这样就把第10行改正了，可以用打入－1LT来加以验证。

接着往下检查，发现第15行的DAA，错打成ADA。我们可以用改正上一个错误的同样方法，即先把字符指针移至出错行，接着用K命令删除这一行，再用I命令来插入新的一行。这是用行编辑改正错误的一般方法，我们建议读者使用这个方法，熟悉这个方法。这是一种易掌握而又可靠的方法。

下面我们介绍另一种方法。

5. 变换命令

我们先用 B14LT，把指针移至第 15 行的前面，然后用变换命令 S 把 ADA 换成 DAA

*SADA ESC DAA ESC 〈CR〉

变换完以后，用 0 LT 命令来加以验证。

再接着往下检查，发现程序的前 20 行已经正确了，就要检查程序的剩下部分。首先，要把剩下的部分显示出现，可打入

*B20L 〈CR〉

*20T 〈CR〉

```
                CALL        5
DELAY:          LD          HL, 0FFFFH
                DEC         HL
                LD          A, H
                OR          L
                JR          NZ, DELAY
                JP          LOOP
                END         START
```

*

发现剩下部分，本来在第三行的标号 DELAY: 错打至第二行。改正的方法：

```
* 1LT 〈CR〉
DELAY:          LD    HL, 0FFFFH
* 2K 〈CR〉
* I〈CR〉
(打入CTRL-I)    LD    HL, 0FFFFH 〈CR〉
DELAY:          DEC   HL 〈CR〉
ESC *
```

也可以用下面的办法：

```
* 1LT 〈CR〉
DELAY:          LD    HL, 0FFFFH
* 6D 〈CR〉
* 1LT 〈CR〉
                DEC   HL
*I 〈CR〉
 DELAY: ESC *
```

把整个文件修改完毕以后，可以仍在编辑程序下，用 B#T 命令（要连通打印机）把整个编辑缓冲区的内容打印出来，再仔细对照，确认无错以后，用 E 命令把文件存盘，返回 DOS

```
* E 〈CR〉
A.
```

在DOS下，再可以用TYPE命令把B97．Z80文件打印一下，再一次对照无误后，则新文件的建立和修改工作初步完成，下面就进行汇编。

三．汇编程序ASMB（Assembler）

经过编辑后建立的源文件，必须经过汇编。调用汇编程序的格式为：

A．ASMB　文件标记

Z80宏汇编程序，要求文件的扩展名必须为．Z80，所以，在文件标记中不要扩展名。例如：

A．ASMB　B97　〈CR〉

DOS就先从盘上把汇编程序调入内存，然后执行汇编程序。汇编程序再把源文件B97．Z80读入，边汇编边检查有否语法错误，若有则显示错误信息，如下所示。

```
CROMEMCO  Z80  Macro  Assembler  version  03.07
000A′  000000        0005          PUAH    BC
***  opcode error  ***
0010′  DD210000#     0008          LD      IX, BUFFER
***  multiple definition  ***
003A′  CD0000        0027          CALL    CHANGE
***  undefined symbol  ***
0040′  CD0000        0029          CALL    CHANGE
***  undefined symbol  ***
0043′  ED436400      0030          LD      64H
***  syntax error  ***
0047′  CD0000        0031          CALL    CHANGE
***  undefined symbol  ***
004A′  ED430A00      0032          LD      OAH
***  syntax error  ***
```

其中列出出错的语句编号和源语句，并指出是操作码错误(opcode error)，即误把PUSH BC打入为PUAH BC。多重定义（multiple definition）即对符号BUFFER有两处以上的定义。未定义符号（undefined symbol）即对符号CHANGE未作定义。语法错误（syntax error)LD 64H没有指明数据传送的目的等。

若经过汇编，语法检查也正确，则给出如下响应：

```
CROMEMCO Z80 Macro Assembler version 03.07
Errors              :
Range Count       0:
Program Length 00EC (:)
End of assembly
```

（有的汇编程序告诉你错误的总数为0）

在返回DOS后，就在盘上建立两个文件。一个是以·REL为扩展名的浮动文件，在此文件中，助记符都已汇编为机器码，而地址也已改变为以0000为基准的相对地址。这样的文件的地址不是真正在内存中存放的绝对地址，因此是不可执行的。另一个是以

·PRN 为扩展名的可打印的文件，如下所示：

```
CROMEMCO Z80 Macro Assembler version 03.07
*** W11 ***
        (000D)      0001  CR:      EQU   0DH
        (000A)      0002  LF:      EQU   0AH
0000'   1E3A        0003  START:   LD    E, ':'
0002'   0E02        0004           LD    C, 2
0004'   CD0500      0005           CALL  5
0007'   11D300'     0006           LD    DE, BUFFER
000A'   0E0A        0007           LD    C, 10
000C'   CD0500      0008           CALL  5
000F'   11D500'     0009           LD    DE,
                                   BUFFER+2
0012'   1A          0010           LD    A, (DE)
0013'   E60F        0011           AND   0FH
0015'   12          0012           LD    (DE), A
0016'   13          0013           INC   DE
0017'   1A          0014           LD    A, (DE)
0018'   E60F        0015           AND   0FH
001A'   12          0016           LD    (DE), A
001B'   13          0017           INC   DE
001C'   13          0018           INC   DE
001D'   1A          0019           LD    A, (DE)
001E'   E60F        0020           AND   0FH
0020'   12          0021           LD    (DE), A
0021'   13          0022           INC   DE
0022'   1A          0023           LD    A, (DE)
0023'   E60F        0024           AND   0FH
0025'   12          0025           LD    (DE), A
0026'   13          0026           INC   DE
0027'   13          0027           INC   DE
0028'   1A          0028           LD    A, (DE)
0029'   E60F        0029           AND   0FH
002B'   12          0030           LD    (DE), A
002C'   13          0031           INC   DE
002D'   1A          0032           LD    A, (DE)
002E'   E60F        0033           AND   0FH
0030'   12          0034           LD    (DE), A
0031'   11D500'     0035           LD    DE,
```

```
                                                    BUFFER+2
0034'  1A          0036          LD      A, (DE)
0035'  CDA400'     0037          CALL    MUL10
0038'  67          0038          LD      H, A
0039'  13          0039          INC     DE
003A'  13          0040          INC     DE
003B'  1A          0041          LD      A, (DE)
003C'  CDA400'     0042          CALL    MUL10
003F'  6F          0043          LD      L, A
0040'  13          0044          INC     DE
0041'  13          0045          INC     DE
0042'  1A          0046          LD      A, (DE)
0043'  CDA400'     0047          CALL    MUL10
0046'  4F          0048          LD      C, A
0047'  CDC600'     0049  LOOP:   CALL    DELAY
004A'  79          0050          LD      A, C
004B'  C601        0051          ADD     A, 1
004D'  27          0052          DAA
004E'  4F          0053          LD      C, A
004F'  FE60        0054          CP      60H
0051'  C26E00'  R  0055          JP      NZ, DISPY
0054'  0E00        0056          LD      C, 0
0056'  7D          0057          LD      A, L
0057'  C601        0058          ADD     A, 1
0059'  27          0059          DAA
005A'  6F          0060          LD      L, A
005B'  FE60        0061          CP      60H
005D'  C26E00'  R  0062          JP      NZ, DISPY
006O'  2E00        0063          LD      L, 0
0062'  7C          0064          LD      A, H
0063'  C601        0065          ADD     A, 1
0065'  27          0066          DAA
0066'  67          0067          LD      H, A
0067'  FE24        0068          CP      24H
0069'  C26E00'  R  0069          JP      NZ, DISPY
006C'  2600        0070          LD      H, 0
006E'  11DF00'     0071  DISPY:  LD      DE, BLOCK
0071'  3E0D        0072          LD      A, CR
0073'  12          0073          LD      (DE), A
```

```
0074'  13          0074              INC   DE
0075'  3E0A        0075              LD    A, LF
0077'  12          0076              LD    (DE), A
0078'  13          0077              INC   DE
0079'  7C          0078              LD    A, H
007A'  CDB200'     0079              CALL  TRAN
007D'  13          0080              INC   DE
007E'  3E3A        0081              LD    A, ':'
0080'  12          0082              LD    (DE), A
0081'  13          0083              INC   DE
0082'  7D          0084              LD    A, L
0083'  CDB200'     0085              CALL  TRAN
0086'  13          0086              INC   DE
0087'' 3E3A        0087              LD    A, ':'
0089'' 12          0088              LD    (DE), A
008A'  13          0089              INC   DE
008B'  79          0090              LD    A, C
008C'  CDB200'     0091              CALL  TRAN
008F'  13          0092              INC   DE
0090'  3E24        0093              LD    A, '$'
0092'  12          0094              LD    (DE), A
0093'  C5          0095              PUSH  BC
0094'  D5          0096              PUSH  DE
0095'  E5          0097              PUSH  HL
0096'  11DF00'     0098              LD    DE, BLOCK
0099'  0E09        0099              LD    C, 9
009B'  CD0500      0100              CALL  5
009E'  E1          0101              POP   HL
009F'  D1          0102              POP   DE
00A0'  C1          0103              POP   BC
00A1'  C34700'   R 0104              JP    LOOP
00A4'  87          0105  MUL10:      ADD   A, A
00A5'  27          0106              DAA
00A6'  47          0107              LD    B, A
00A7'  87          0108              ADD   A, A
00A8'  27          0109              DAA
00A9'  87          0110              ADD   A, A
00AA'  27          0111              DAA
00AB'  80          0112              ADD   A,B
00AC'  27          0113              DAA
```

```
00AD' 47          0114              LD      B, A
00AE' 13          0115              INC     DE
00AF' 1A          0116              LD      A, (DE)
00B0' 80          0117              ADD     A, B
00B1' C9          0118              RET
00B2' 47          0119  TRAN:       LD      B, A
00B3' CB3F        0120              SRL     A
00B5' CB3F        0121              SRL     A
00B7' CB3F        0122              SRL     A
00B9' CB3F        0123              SRL     A
00BB' F630        0124              OR      30H
00BD' 12          0125              LD      (DE), A
00BE' 13          0126              INC     DE
00BF' 78          0127              LD      A, B
00C0' E60F        0128              AND     0FH
00C2' F630        0129              OR      30H
00C4' 12          0130              LD      (DE), A
00C5' C9          0131              RET
00C6' E5          0132  DELAY:      PUSH    HL
00C7' F5          0133              PUSH    AF
00C8' 21FFFF      0134              LD      HL, 0FFFFH
00CB' 2B          0135  GOON:       DEC     HL
00CC' 7C          0136              LD      A, H
00CD' B5          0137              OR      L
00CE' 20FB        0138              JR      NZ, GOON
00D0' F1          0139              POP     AF
00D1' E1          0140              POP     HL
00D2' C9          0141              RET
                  0142  BUFFER:
00D3' 0A          0143  LENGTH:     DB      10
00D4' (0001)      0144  LENT1:      DS      1
00D5' (000A)      0145  LINE:       DS      10
00DF' (000C)      0146  BLOCK:      DS      12
00EB' (0000')     0147              END     START
Errors            :
Range Count       :
Program Length    00EB (:)
Symbol      Value     Defn      References
BLOCK       00DF'     0146      0071  0098
```

BUFFER	00D3′	0142	0006	0009	0035
CR	000D′	0001	0072		
DELAY	00C6′	0132	0049		
DISPY	006E′	0071	0055	0062	0069
GOON	00CB′	0135	0138		
LENGTH	00D3′	0143			
LENT1	00D4′	0144			
LF	000A′	0002	0075		
LINE	00D5′	0145			
LOOP	0047′	0049	0104		
MUL10	00A4′	0105	0037	0042	0047
START	0000′	0003	0147		
TRAN	00B2′	0119	0079	0085	0091

在文件的右边，列出了源程序，它完全按照打入时的样子列出（包括所打入的注释，如果有的话），而且给源程序的语句按顺序编了号。在左边就是汇编后的浮动文件。最左边一列是以 0000 为基准的相对地址，故在其上用 ′ 号表示。接着是汇编后的机器码，其中有的地址也是相对地址。

程序之后就是汇编的一些信息，若有语法错误，也将指出错误信息。最后，列出了本程序中的符号表。

汇编以后的浮动文件，必须经过连接变为绝对地址的机器码文件才可运行。

四、连接程序 LINKER

连接程序的功能：

1．在 Cromemco 系统中有两个汇编语言的子程序库：ASMLIB.REL 和 DEMO LIB.REL。其中包括输入字符子程序 GETCH 和输出字符子程序 PUTCH，输入十进制数子程序 GETDN 和输出十进制数的子程序 PUTDN等等。这些子程序库中的子程序，可以在汇编语言的源程序中加以调用。这样，就需要经过 LINKER 把它们连接起来。

2．在汇编语言中，常常用伪指令 ORG 来规定程序或数据块存放的绝对地址。这也需要由连接程序给它们规定绝对地址值。另外在 CP/M 中，允许汇编语言的源程序中，不用 ORG 指令来指定绝对地址。但 CP/M 操作系统要占用内存的 0000—00FFH 作为它的低端地址区。在没有用 ORG 指令指定的情况下，用户程序从 0103H 开始存放，而在 0100—0102 处安放一条无条件转移指令，转移至用户程序的入口。故在没有用 ORG 指令指定的情况下，0100—0102 存放的是 C30301 即 JP 0103。在用 ORG 指令指定程序的入口地址的绝对值的话（例如用 ORG 1000H），则连接后用户程序的起始地址为 1000H，而在 0100—0102 处安放的是一条 JP 1000H指令。这些都是由连接程序来完成的。

以后当把程序调入执行时，就先转到 0100 处执行，然后由此处的无条件转移指令，转至用户程序的真正入口。

调用连接程序的格式为：

A．LINK 文件标记/开关

例如：

A．LINK B97/E

因为连接程序一定是对.REL 文件进行连接，故文件标记中省去扩展名。开关 E，指出完成连接后立即返回 DOS。

在调用连接程序后，若连接成功，则机器的响应如下：

A．LINK B97/E

〔xxxx yyyy nn〕

其中 xxxx 是指文件经连接后入口地址的绝对值，例如 0103。yyyy 是指文件所占用的最高地址。nn 是指文件所占的页数，以 256 个字节为 1 页。

连接以后的机器码文件，还必须用 SAVE 命令把它存盘。

五、存盘（SAVE）命令

命令的格式为：

A．SAVE 文件标记　参数

其中，文件标记的扩展名必须为·COM。参数即为连接命令中所指出的程序所占的页数。例

A．SAVE B97.COM 1〈CR〉

（A＞ SAVE 1 B97．COM〈CR〉）

A．

这样就把机器码文件存盘，形成了一个可执行的文件。

六、运行

在经过了上述步骤，把用户的文件以机器码的形式存盘以后。用户文件就可以像系统的一个外部命令一样，直接用文件名（不带扩展名·COM）把文件调入内存并运行。在运行中 CRT 上有否显示，就完全取决于用户程序中有没有向 CRT 输出的程序段。程序运行完后是否返回 DOS，也取决于是否有指令 JP 0000H 或 CALL 0000H。

以上，我们是根据实验中的需要，最最简单地介绍了编辑命令、汇编命令和连接命令。详细的可参阅有关的手册。

第四节　微型计算机系统实验

实验一　微型计算机系统的操作及编辑程序的使用

一、实验目的

1．学习和掌握微型计算机系统的开机和关机的步骤和方法。

2．学习和熟悉操作系统命令的使用。

3．学习如何调用编辑程序和用插入方式输入和建立一个新文件，学习一些编辑命令。

4．本实验结束时，在盘上建立一个延时 1 秒的子程序的文件。

二、实验内容

1．微型计算机系统的开机和关机练习。Cromemco 微型机系统如图 2.2(a) 所示，是由主机、CRT(键盘显示终端) 和打印机组成。磁盘驱动器装在主机箱内。

Apple Ⅱ 也由以上几个主要部分组成，但键盘与 CRT 不是固定在一起的，磁盘驱动器单独放置。

开机的步骤为：

(1) 合上交流电源 (220V) 闸。

(2) 按由外及内的步骤，合上各部件的电源开关。

① 合上打印机开关；

② 合上 CRT 开关；

③ 合上主机开关。

(3) 把盘片插入驱动器。

软磁盘的外形如图 2.2(b) 所示。它以软塑料为基片，表面涂上磁性介质的圆形磁盘，始终装在方形的黑色封套里面，把磁盘插入驱动器后，盘片就可以在封套里旋转。在封套上开有一个槽，只有这一部分的盘片表面是裸露的，磁头就在这部位对磁盘进行读写，磁头可以沿着槽移动，就把磁盘表面分成一个一个的同心圆，称为磁道，信息就存贮在磁道上。磁盘上有一个小孔叫做索引孔，它表示磁道的开始。磁盘的边上有一个小缺口，用于整个盘片的写保护。对于 5 英寸盘来说，若把这个缺口粘贴上，则整个盘片是写保护的；没有粘贴，则表示盘片可写。对于 8 英寸盘则刚好相反。

封套表面有一边有标记，如图 2.2(b) 中所示。当把磁盘插入驱动器时，要注意磁盘插入的方向，插入时要轻，要对准了才插，不要用力过猛以防损坏磁盘，插入磁盘轻轻推到头后，关上驱动器的门。

对于 Cromemco 系统来说，一定要先合上电源，才插入磁盘。

(4) 按复位按钮 (或连着按几下键盘上的回车键) 系统或自动引导 CDOS，在显示 CDOS 的版本号后，出现提示符 "A."。或先进入 RDOS，出现 RDOS 的提示符 "；"，这时可以按字母键 B 就可以引导 CDOS。

(对于 Apple Ⅱ 机，插入 CP/M 系统盘后就可以自动引导 CP/M 系统，出现提示符 A〉)

当出现 CDOS 的提示符 A．以后，就可以使用系统的命令了。

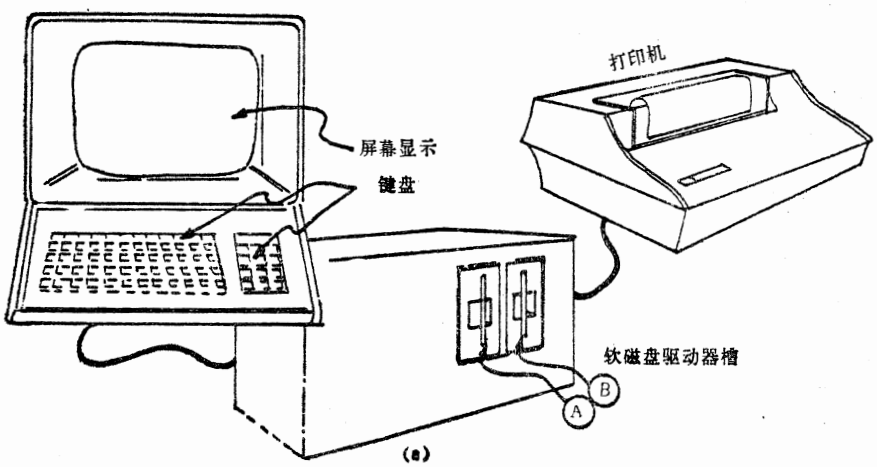
(5) 键盘练习

键盘上的键，按其作用，大致可分为：

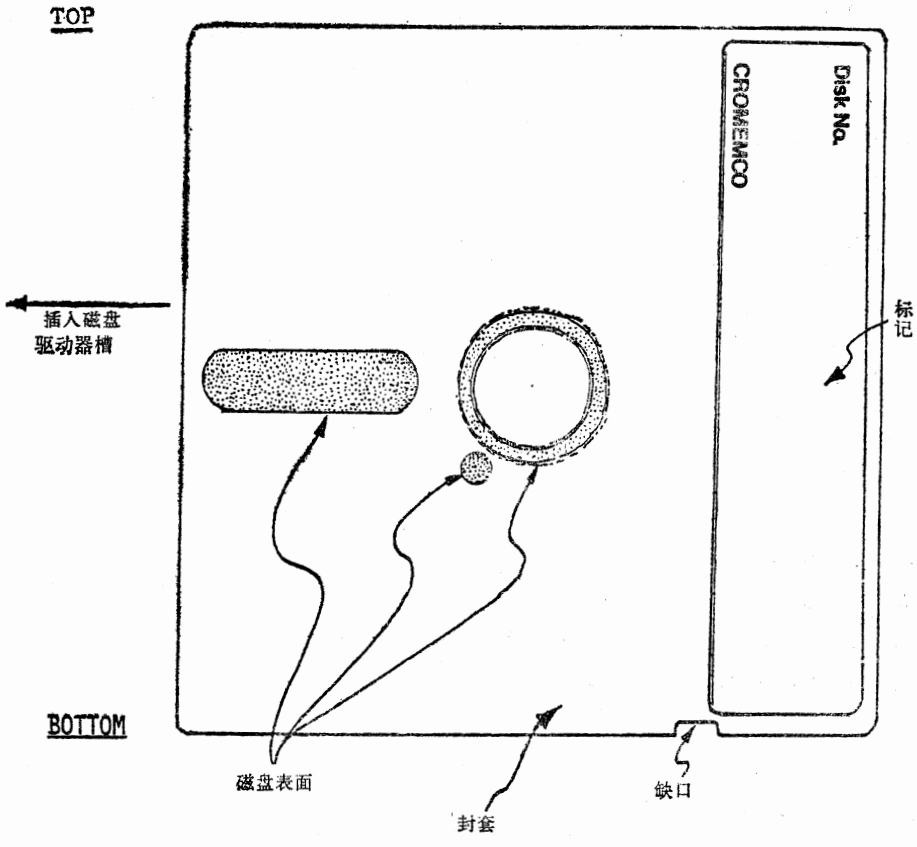
①字符键。如数字键 0 — 9，字母键 A — Z，各种标点符号等，键盘下方的长条键是空格键。按下这些键，可以在屏幕上显示相应的字符 (按空格键屏幕上出现一个间隔)。

②功能键。如回车键 (RETURN) 控制回车和换行；SHIFT 键控制上、下档字符，为了制造方便，总是字符数多于键数，所以有的键就代表两个字符分为上下档，在不按 SHIFT 键单独按某一键时下档字符有效，而同时按下 SHIFT 和某一键时则上档字符有效；删除键 RUB(或 DEL 或←)；控制光标移动的键← ↑ → ↓ 等。

Cromemco 微型计算机系统



(a)



(b)

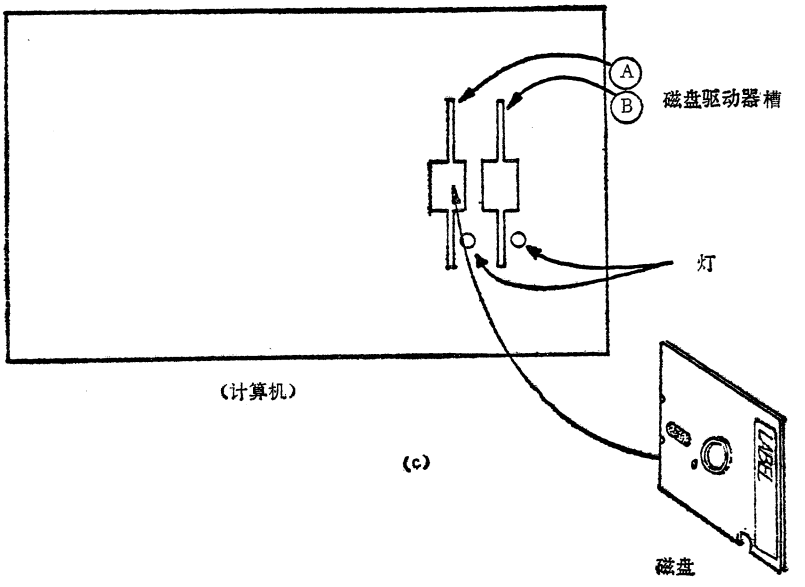


图2.2 Cromemco系统的配置和磁盘插入的方法

③控制键。这总是要由 CTRL 键和别的字母键组合起来才起一定的控制作用。如前面提到的 CTRL-P 是控制打印机与 CRT 连接的开关；CTRL-Z 是编辑程序退出插入方式的命令；CTRL-I 使光标跳至下一个固定位置；CTRL-U 将已输入的命令行作废等等。

④特殊功能键。在有的终端的键盘上设置了一些特殊功能键。这些键的功能可以是 CDOS 设置的标准功能，例如有的键代表 EDIT 命令，有的键代表 ASMB 命令，有的键代表 BASIC 命令等等，使使用更为方便。也可以由用户自己规定这些键的功能。

练习和熟悉这些键的功能和使用。特别是要尽可能记住各个键的位置，以提高键入的效率。

关机的步骤如下：

（1）打开驱动器的门，取出盘片，放入纸套内保存。

（2）先关主机电源，按由里及外的次序断开各个部件的电源。

（3）拉开交流电源闸。

反复练习几次上述的开机和关机的过程，逐渐养成正确的开机和关机的习惯。

2．操作系统命令练习

（1）在显示 DOS 的提示符以后，用 DIR 命令显示盘上的文件目录。

先列出系统盘（现行盘）上的目录，查看一下系统给我们提供了哪些软件。

再列出系统中各个盘的目录。

再试着列出一部分文件的目录（用多义文件标记）。

（2）根据文件目录，找到一个可用 TYPE 命令打印的文件，练习用 TYPE命令在 CRT 上显示此文件。

3．编辑程序命令练习

（1）调用编辑命令，建立一个新文件（自己命名）。

①用插入方式，输入以下内容：

```
START:    LD      A, 10
          LD      B, 20
          ADD     A, C
          ADA
          HALT
          END
```

②在退出插入方式后，把它存盘。

③用 DIR 命令检查盘上是否增加了一个新文件。

④把此文件再次用编辑命令调入，且加以修改变成如下内容：

```
START:    LD      A, 10
          LD      B, 20
          LD      HL, BUFFER
          ADD     A, B
          DAA
          LD      (HL), A
          INC     HL
          HALT
BUFFER:   DS      10
          END     START
```

修改后存盘。

（2）用 DIR 命令，查看编辑和修改后的文件名。

（3）用 TYPE 命令把修改后正确的源文件打印出来（连接打印机）。

（3）用 REN 命令把建立的文件改名。

（4）在上述操作都完成以后，用删除命令 ERA，把上述文件清除。

4. 在盘上建立一个延时一秒的子程序的源文件。

给定的程序清单如下：

```
DELAY:    PUSH    AF
          PUSH    BC
          PUSH    DE
          PUSH    HL
          LD      B, 50
LOOP:     CALL    DELAY1
          DEC     B
          JP      NZ, LOOP
          POP     HL
          POP     DE
          POP     BC
          POP     AF
```

```
                RET
DELAY 1：       LD      H，  0 AH
TIME 1：        LD      L，  0 DCH
TIME 2：        DEC     L
                NOP
                JP      HZ，TIME 2
                DEC     H
                LD      IX，  0
                JP      NZ，TIME 1
                RET
```

三、预习要求

1．仔细阅读本实验指导书。

2．仔细阅读第二节中的 CDOS 内部命令部分和第三节中的编辑命令部分。

3．尽可能看懂本实验中给出的程序。

四、报告要求

1．总结微型机系统开机和关机的步骤。

2．总结编辑程序使用的方法。

实验二　在CRT上连续显示0-9的编程练习

一、实验目的

1．进一步熟悉 DOS 的内部命令的使用方法。

2．进一步熟悉编辑命令。

3．学习和掌握建立与运行汇编语言源程序各步的命令。

4．初步学习利用系统调用编程的方法。

5．简单的程序设计练习。

二、实验内容

1．编一个能在 CRT 上重复显示 0-9 的源程序。

（1）要在 CRT 上显示字符，可以利用系统调用 2，一定要注意在用系统调用 2 前要保护现场。

（2）要显示的字符的 ASCII 码在调用前送寄存器 E。

（3）要输出 0-9，可以用循环程序来实现。例如让 B 的初值为 0，每循环 1 次加 1，但当加到 9 以后再加 1，应该又返回到 0，这就可以用 DAA 指令来实现。但这样从低 4 位到高 4 位会有进位产生，这样就可以设法把高 4 位屏蔽掉。然后把此数一方面送回 B 保存，另一方面把它变成 ASCII 码。

（4）调用1秒的延时程序后再循环。

程序的流程图如图2.3所示。

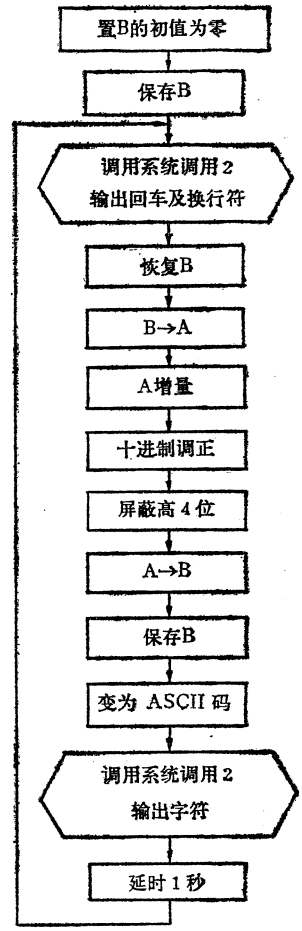


图2.3 在屏幕上循环显示0-9的程序流程图

2．把上一个实验中已建立起来的延时1秒的源程序调入，在此程序前插入已编好的能循环显示0-9的程序。

插入过程可能会产生打入错误，利用编辑命令加以修改，直至无错。

3．调用汇编程序，对建立的源程序进行汇编。注意观察汇编程序的响应，检查有否语法错误。若有的话，要根据指出的错误处和给出的错误信息，检查源程序，找出错误的原因。在找到所有的错误原因后，重新调用编辑程序进行修改，修改以后再用汇编命令进行汇编。这样，可能要反复多次，直至汇编后无语法错误为止。

4．用DIR命令，查看经汇编后新建立的文件是什么？

5．调用连接程序，使浮动文件变为绝对地址。

6．用SAVE命令把连接后的文件存盘。

7．再用DIR命令，查已存盘的文件名。

8．在验证了盘上确有以·COM为扩展名的机器码文件后，用文件名调用和运行程序，观察屏幕显示。

9．按复位按钮，使系统返回DOS。

若程序运行的结果不对，则仔细检查源程序，寻找错误。

CDOS提供DEBUG程序，CP/M提供DDT程序，都可以用来调试程序，帮助我们寻找错误。使用方法请见附录。

若确实经过努力仍寻不出错误所在，则可对照附录中的参考程序来寻找。

找出错误后，要重新经过编辑、汇编、连接和存盘这样的全过程，建立一个修改后的机器码文件再运行。直至运行正确。

10．用TYPE命令打印源程序和打印文件（扩展名为·PRN的文件）。打印源程序时连接打印机。

11．在盘上保留源程序和机器码文件，把其他文件（包括·BAK文件·REL文件·PRN文件）删除。

三、预习要求

1．仔细阅读本实验指导书。

2．仔细阅读第三节的内容。

3．仔细阅读第二节中系统调用部分。

4．参照流程图，在实验前编好要求的源程序。

5．写出上机步骤。

四、报告要求

1．整理出运行正确的源程序。
2．总结原编的程序中的错误。
3．总结从建立到运行源程序的步骤及所用的命令。
4．总结使用编辑程序进行修改的方法。

实验三　在CRT上输出字母A-Z，以及数字0-59的程序设计

一、实验目的

1．进一步熟悉 DOS 的命令。
2．进一步熟悉从建立到运行汇编程序的步骤和所用到的命令。
3．练习简单的程序设计。
4．学习用系统调用 9 来输出字符串。

二、实验内容

1．编制一个能在 CRT 上循环显示字母 A—Z 的程序。

（1）要在 CRT 上输出字符，仍要用系统调用 2。

（2）从 ASCII 码表上查到字母 A—Z 的 ASCII 码。

（3）显示 A—Z，用循环程序，先把累加器 A 设置为字母 A 的 ASCII 码值，每循环一次 A 加 1，比较是否大于 Z，若是则重新设置 A 的初值，不停地循环。

（4）在循环内调用 1 秒延时子程序。程序的流程图如图 2.4 所示。

2．调用编辑程序，输入上一个实验中的源文件，把 1 秒延时子程序前的程序都删除掉；然后插入新编的源程序。若插入时有打入错误，则用编辑命令加以改正，然后存盘。

3．把第 2 步后形成的 · BAK 文件，改名为别的文件（扩展名用 · Z80），备用。

4．对新的源文件进行汇编、连接，以机器码形式存盘。

若汇编时有语法错误，则查出错误，用编辑程序予以改正。

5．运行程序。

若运行的结果不对，则仔细寻找源程序中的错误，直至运行正确。

6．编制一个能在屏幕上循环显示 00-59 的源程序。

（1）要在屏幕上一次显示两个或两个以上的字符，用系统调用 9 比较方便。

（2）00-59，仍可用循环程序来实现，初值为 0，每循环一次加 1，最大值为 59 则用一个字节就可以了。每循环一次加 1 后检查是否为 60，若是则又从 0 开始。

（3）因是十进制数，加 1 后要用十进制调正。

（4）要输出显示的应是数码的 ASCII 码，所以，两位数要分别转换为 ASCII 码，放入输出缓冲区，输出缓冲区的最后要加入字符'$'。

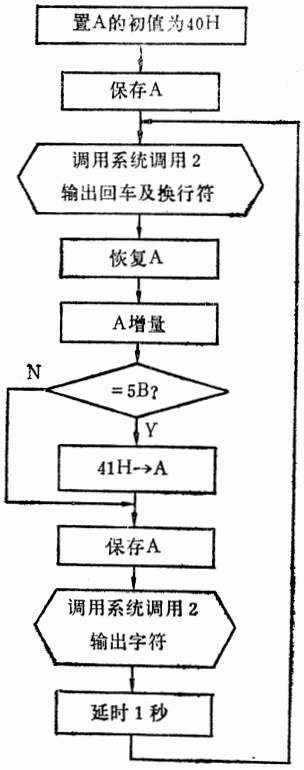


图2.4　在屏幕上循环显示字母A-Z的程序流程图

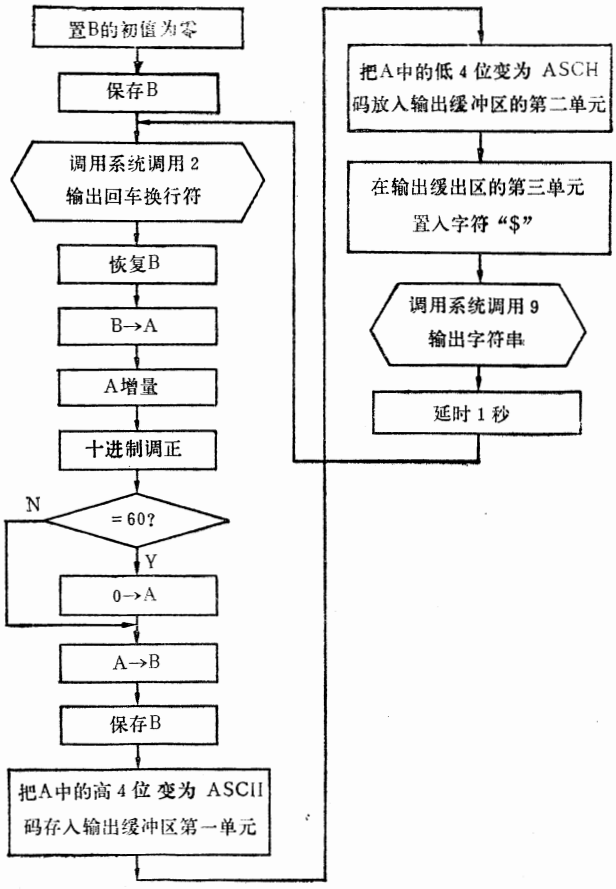


图2.5　在屏幕上循环显示00-59的程序流程图

（5）每一循环调用1秒延时子程序。

能实现上述要求的程序流程图如图2.5所示。

7．用编辑程序，输入由·BAK文件改名后的文件，对照新编出的源程序，加以增补和修改，直至无误。

8．对新的源程序进行汇编、连接，以机器码形式存盘。

9．运行程序。

10．用TYPE命令打印本实验中两个新程序的源程序和·PRN文件。

11．保留每个程序的源文件和机器码文件，把其它文件删除。

三、预习要求

1．仔细阅读本实验指导书。

2．仔细阅读系统调用9的功能和调用的方法。

3．参照流程图，在实验前编好源程序。

4．写出本实验的步骤。

四、报告要求

1．整理出运行正确的各个源程序，在程序清单上加上注释。
2．总结使用系统调用 9 的方法。
3．总结 DAA 指令的功能。
4．小结程序设计中的一些体会。

实验四　在CRT上显示实时时钟的程序设计

一、实验目的

1．学习如何使用系统调用 10，从键盘输入字符串。
2．进一步掌握如何使用系统调用 9，向 CRT 输出字符串。
3．进一步熟悉系统的操作命令。
4．进一步提高编制汇编语言源程序的能力。

二、实验内容

1．编制一个能从控制台输入时钟的初值，且能在 CRT 上显示实时时钟的源程序。

（1）为了能从控制台输入时钟的初值，先向控制台输出一个提示符':'。

（2）利用系统调用 10，用从控制台输入字符串的办法，输入时钟的起始值。

为了简化程序设计，规定输入起始值的格式为：

HH：MM：SS
（时）（分）（秒）

即用 24 小时时钟制，不管时、分、秒的值多大，一律用两位十进制数表示，在时、分、秒之间用：号加以分隔。

（3）时、分、秒的初值都是用两位十进制数表示的，而输入时是每一个十进制数的 ASCII 码。所以，在程序中先要把 ASCII 码转换为 BCD 码，然后把两位 BCD 码拼在一个字节中，送入某一寄存器。

（4）在从输入缓冲区取出时、分、秒的初值，经过处理后，就调用一秒的延时子程序。过了一秒后，使秒值加 1，检查是否为 60，若不是，则转至输出显示。若秒值已为 60，则使分值加 1，秒值置为 0，再判断分值是否已为 60，若不是，则转至输出显示。若分值已为 60，则时值加 1，分值置为 0，再判断时值是否已为 24，若不是，则转至输出显示。若时值已为 24，则令时值为 0，转至输出显示。

（5）为了简化程序，输出的时钟格式也为

HH：MM：SS
（时）（分）（秒）

要能这样显示，就要调用系统调用 9。就要设置一个输出缓冲区，在缓冲区的开始先设置回车与换行字符，使每次的输出能显示在不同的行上。然后要设法把在一个字节中表

示的时、分、秒的两位十进制数，分别转换为它们的 ASCII 码，按高位在前的次序送入输出缓冲区。而且在时、分、秒之间插入分隔符':'。最后，要添上字符'$'。

能实现上述要求的程序流程图如图 2.6 所示。

2．调用编辑程序，输入上一个实验中的后一个源文件。把延时子程序前的程序段全部删去，用插入方式输入新编的源程序。

3．通过汇编、连接的存盘，建立一个机器码的程序。

4．运行程序。

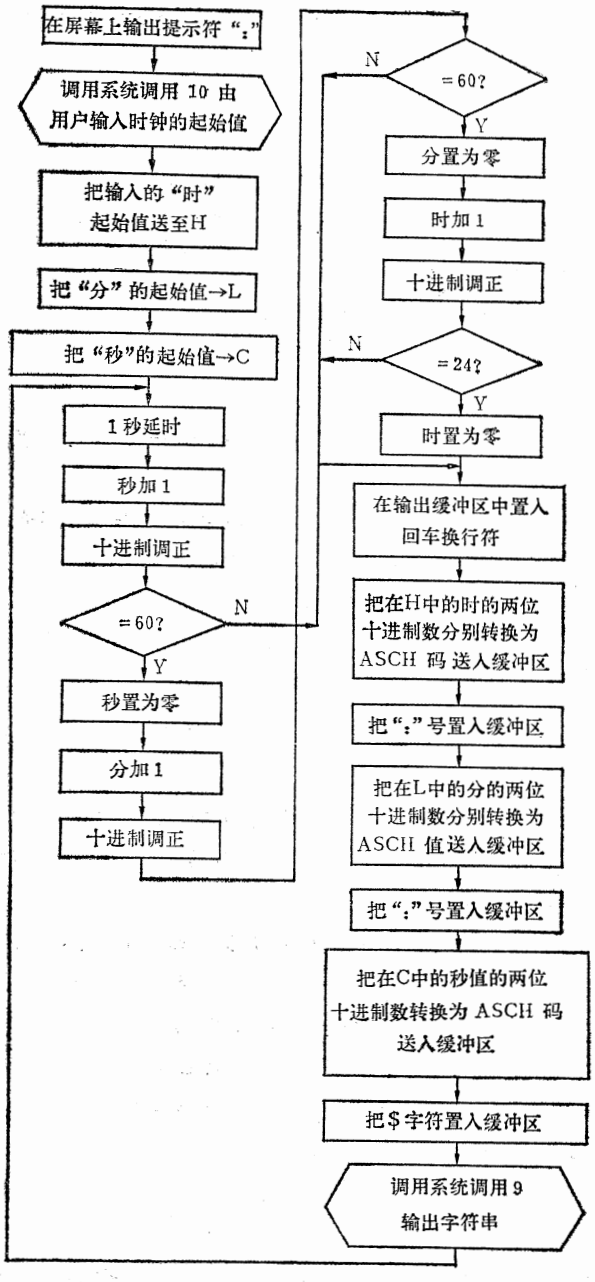


图2.6　在屏幕上用软件显示实时时钟的程序流程图

5. 用 TYPE 命令，打印源程序和 · PRN 文件。

三、预习要求

1. 仔细阅读本实验指导书。

2. 仔细阅读和掌握系统调用 9 和 10。

3. 参照流程图编制源程序。在源程序中要有必要的伪指令。要用伪指令指明输入缓冲区，规定缓冲区的长度（用 DB），留出放实际输入字符个数的存贮单元以及保留缓冲器所需要的存贮单元数（用 DS）。规定输出缓冲区。

4. 写出实验步骤。

四、报告要求

1. 整理运行正确的源程序，加上注释。

2. 总结使用系统调用 9 和 10 的方法。

3. 总结一些程序设计方法。

*实验五　从控制台输入两位十进制数子程序的程序设计

一、实验目的

1. 进一步熟悉利用系统调用 10 从控制台输入字符串，以及对它们进行转换的程序设计方法。

2. 学会利用系统调用，自己设计子程序的程序设计方法。

3. 进一步掌握伪指令的使用。

4. 通过本实验建立两个自编子程序的源文件，以备供后续的实验调用。

二、实验内容

1. 编制一个能从键盘输入两位十进制数，把它们变为 BCD 码放在累加器 A 中的子程序INBCD。

（1）在输入数据前，先由 CRT 上输出提示符':'。

（2）利用系统调用 10 输入两位十进制数。为了简化程序，规定输入格式为：

××

即，若输入的为一位十进制数，高位也必须输入为 0 。

（3）把输入的数码转换为 BCD 码，且把两位 BCD 码并成一个字节，送至累加器 A 中。

（4）作为一个子程序，在入口处应有保护现场的指令，而在返回前应有恢复现场的指令。

能实现上述要求的程序流程图如图 2.7 所示。

2. 编一个能把若干个从键盘输入的两位十进制数累加的程序。累加和在用两个字节表示的 4 位 BCD 码的范围内，且保存在寄存器对 HL 中。

（1）程序用循环的方法实现。

（2）调用在前面所建立的输入两位十进制数的子程序 INBCD。

（3）累加和的初值应置 0。

（4）因是十进制数运算，一定要恰当应用 DAA 指令。要注意 DAA 指令只能对在A中的运行结果进行较正。

（5）参与运算的数虽是一字节的，但累加和应是两字节的。

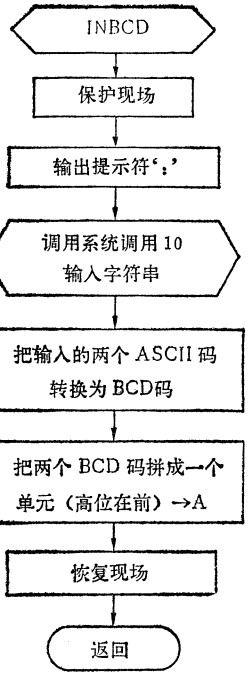


图2.7 从控制台输入两位十进制数子程序的流程图

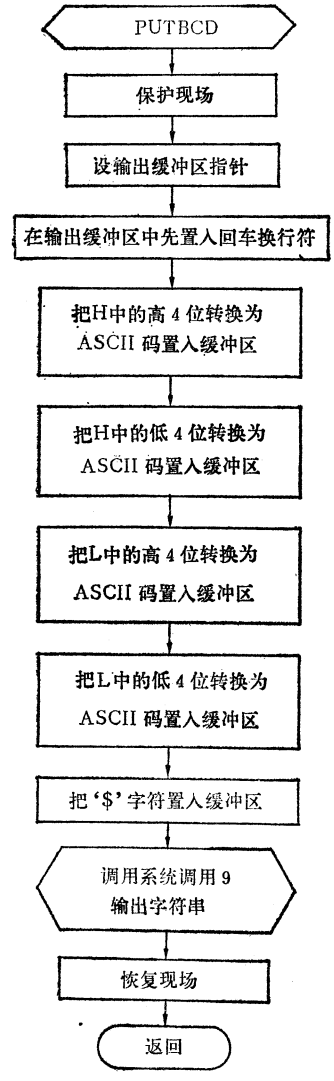


图2.8 把在HL中的 4 位BCD表示的数在CRT上显示的程序流程图

3. 编一个能把在 HL 寄存器对中用 BCD 码表示的 4 位十进制数，在 CRT 上输出显示的子程序。

（1）要输出就要利用系统调用 9。

（2）在输出缓冲区先置入回车换行字符，使每次输出不重叠。

（3）把在寄存器H中的两位BCD码分别转换为ASCII码，以高位在前的原则送入输出缓冲区。

（4）把在L中的两位BCD码，分别转换为ASCII码，也以高位在前的原则，送入输出缓冲区。

（5）在输出缓冲区的最后，添入字符'$'。

（6）作为子程序，在子程序的入口处要用指令保护现场，在返回前要用指令恢复现场。

能满足上述要求的程序流程图如图2.8所示。

4. 把以上三部分编成一个程序。通过编辑、汇编、连接和存盘形成机器码文件。

5. 运行程序。反复运行多次，每次改变输入数据，检验输出的结果是否正确。

6. 打印源文件，保留源文件和机器码文件，把其余文件删除。

7. 编一个程序，调用输入两位十进制数子程序以输入被乘数和乘数；利用累加的方法，实现这两个十进制数相乘，乘积放在HL寄存器对中；调用输出在HL中的十进制数的子程序，输出乘积。

（1）用累加的方法实现乘法，应该用两数中小的那个数，作为循环次数。

（2）乘积的起始值为0。

（3）每一循环做加法，这是十进制数相加，所以要用DAA调正，相加时必须考虑进位。

（4）循环次数是十进制数，故每次减1后也必须用DAA调正。

8. 用编辑程序，输入前面所建立的源文件，保留输入两位十进制数和输出在HL中十进制数这两个子程序，把其余的程序段删去，插入新编好的源程序。

9. 经过汇编、连接和存盘，建立机器码文件。

10. 运行程序。反复运行多次，输入不同的数，核验结果是否正确。

11. 打印源文件；保留源文件和机器码文件；把其余的文件删除。

三、预习要求

1. 仔细阅读本实验指导书。

2. 参照给定的流程图，在实验前编好输入两位十进制数子程序和输出在HL中的十进制数子程序。

3. 根据指导书中的思路，在实验前编好多个十进制数累加和两个十进制数乘法的程序。

4. 写出实验步骤。准备好多组运算数据，算出正确的结果以供校验用。

四、报告要求

1. 整理出各个运行正确的源程序，并加上注释。

2. 整理运算的数据和结果。

3. 总结编制子程序的方法。

*实验六　输入两位带符号十进制数子程序和输出带符号十进制数子程序的程序设计

一、实验目的

1. 进一步掌握系统调用 10。
2. 掌握输入带符号数的方法。
3. 学习带符号数乘除法的程序编制方法。
4. 掌握把带符号的二进制数以十进制的形式输出的程序设计方法。

二、实验内容

1. 编制一个利用系统调用 10，从键盘输入两位带符号的十进制数的子程序。

（1）先由控制台输出提示符':'。

（2）利用系统调用10输入字符串。为了简化程序设计，规定输入的格式为：

±××

即必须有符号位，数值部分必须是两位数。

（3）先检查输入的符号数，然后建立相应的标志。

（4）把输入的两位数的 ASCII 码先转换为 BCD 码。

（5）把十位数的 BCD 码乘以 10 再加上个位数的 BCD 码，把数值部分变为二进制数。

（6）取出符号标志，若是负数则取补。

（7）作为子程序，必须要有相应的保护现场和恢复现场的指令。

根据上述思路，可得如图 2.9 所示的程序流程图。

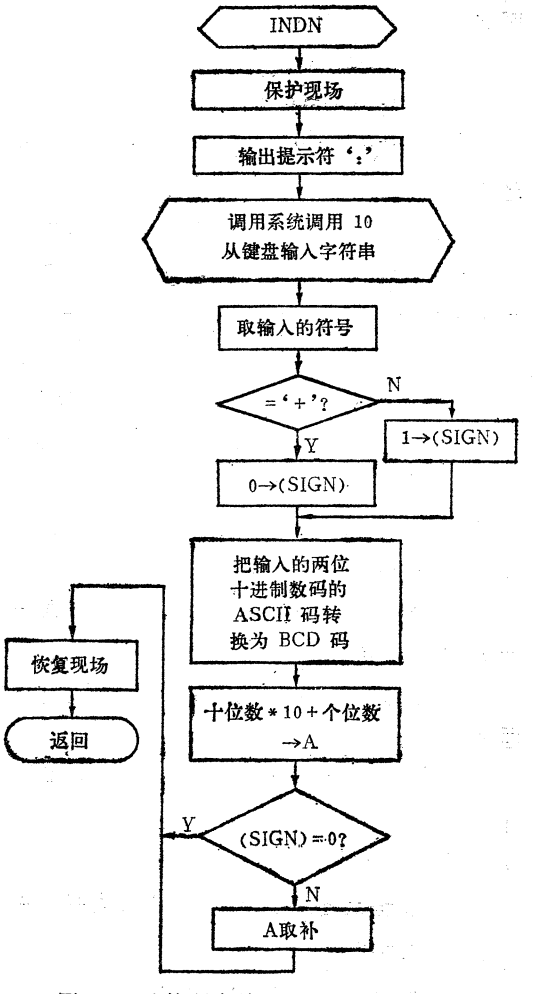


图2.9　从控制台输入两位带符号十进制数子程序的流程图

2. 编制一个能把在 HL 寄存器对中的带符号的二进制数，以十进制的形式(包括符号位）输出的子程序。

（1）要输出就要利用系统调用 9。

（2）设置输出缓冲区，先在缓冲

区中置入回车换行符。

（3）先检查 HL 中的符号，若是正数，则在输出缓冲区中置入正号的 ASCII 码；若是负数，则置入负号的 ASCII 码，且经过取补，得到数值。

（4）数值部分由二进制转换为十进制。因为是两字节数，则其数值范围为≤+32767，≥−32768。所以，转换后的十进制数有万、千、百、十、个五位。转换的方法是：先让数值部分减去一万，看它包含有多少个万；剩余部分减去一千，看它包含有多少个千……。

（5）转换后的五位十进制数，分别转换为相应的 ASCII 码，置入输出缓冲区。为了简化程序，输出始终是五位十进制数。

（6）在输出缓冲区的最后，置入字符'$'。

（7）作为子程序，必须要有相应的保护现场和恢复现场的指令。

根据上述思路，可得如图 2.10 所示的程序流程图。

3. 编制一个程序，从键盘输入两个两位的带符号十进制数，令其相乘，且把乘积在 CRT 上显示出来的程序。

（1）调用前面所编的输入两位带符号十进制数的子程序。

（2）两个带符号数相乘，同号的乘积为正，而异号的乘积为负。所以，可以把符号位取出来单独处理，由两数的符号位的异或得到乘积的符号。这样，乘法就是数值相乘了，直接可以用教材中的部分积右移或被乘数左移的方法。要注意的是：机器中是用补码来表示带符号数的，所以负数的数值要经过取补才能得到。同样，若乘积是负的，则数值也必须经过取补。

（3）调用上面所编制的输出带符号数的子程序来输出乘积。

根据上述思路，可得如图 2.11 所示的程序流程图。

4. 通过编辑、汇编、连接和存盘，建立起机器码文件。

5. 运行程序。

准备多组数据，反复运行程序，检验运行的结果。

6. 打印源文件；保留源文件和机器码文件；把其余文件删除。

7. 编一个程序，能从键盘输入被除数与除数（都是正数），做除法，且把商和余数由 CRT 输出显示。

（1）调用输入两位带符号十进制数子程序。

（2）两数相除。

（3）把商送 HL 寄存器对，调用输出十进制数子程序。

（4）把余数送 HL 寄存器对，调用输出十进制数子程序。

程序的流程图如图 2.12 所示。

8. 调用编辑程序，输入上一个程序的源文件。保留其中的输入和输出两个子程序，把其余的程序段全删除，插入新编好的源程序。

9. 通过汇编、连接和存盘，建立机器码文件。

10. 运行程序。

准备多组数据，反复运行程序，检验结果是否正确。

11. 打印源文件；保留源文件和机器码文件；把其余的文件全部删去。

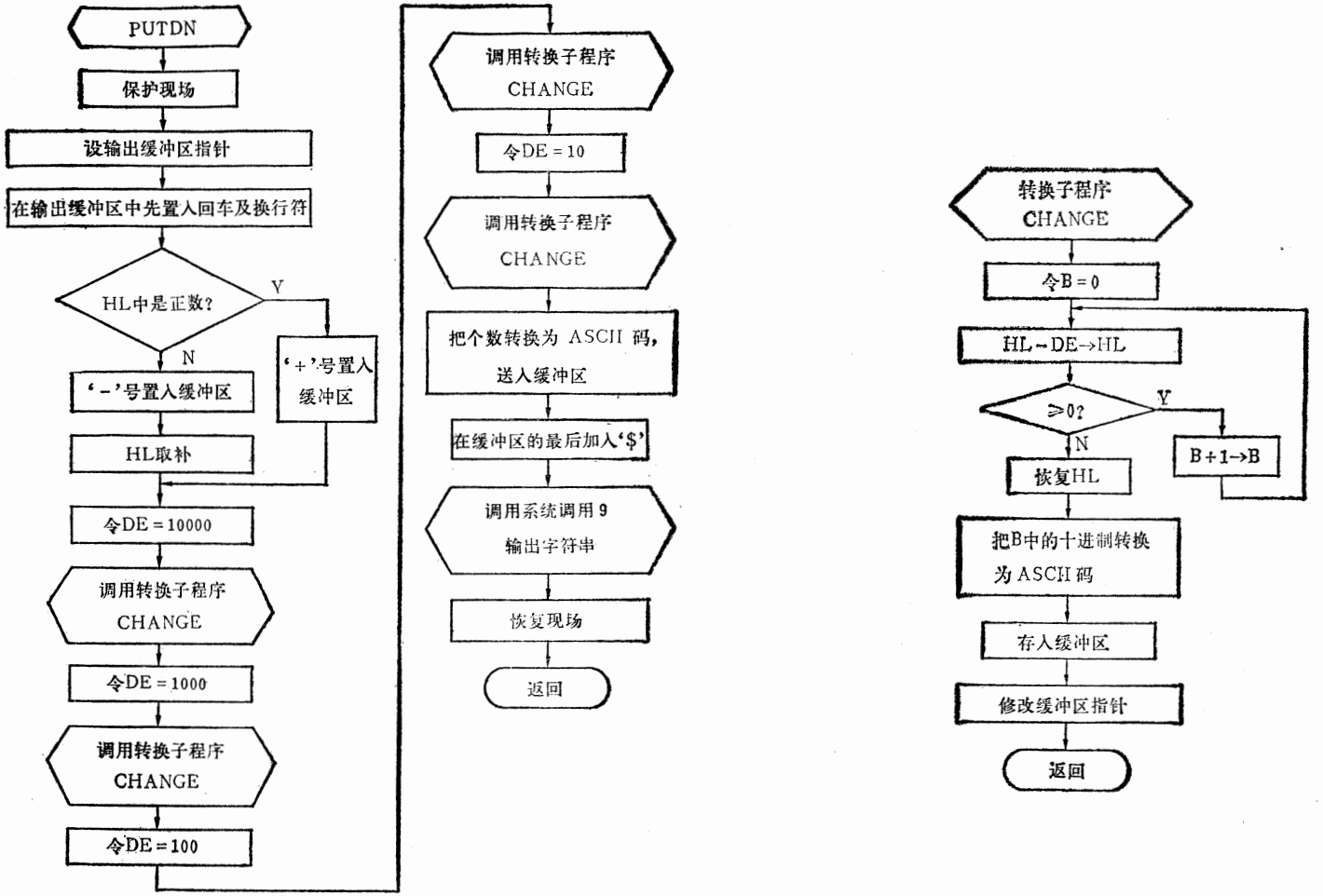


图2.10　把在HL中的带符号二进制数以十进制形式输出的程序流程图

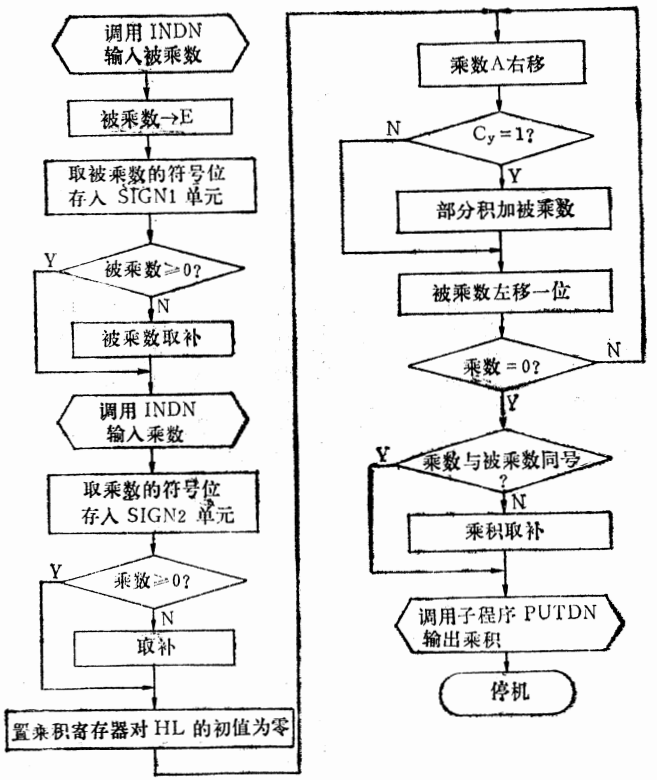


图2.11　由控制台输入两个两位带符号十进制数相乘且输出乘积的程序流程图

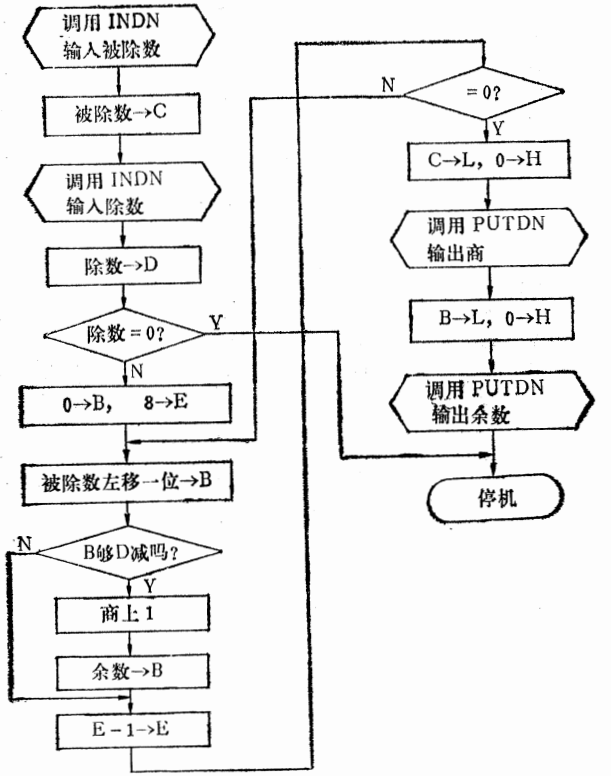


图2.12　由控制台输入两个两位十进制正数相除，输出商和余数的程序流程图

三、预习要求

1. 仔细阅读本实验指导书。

2. 参照流程图，在实验前编好两个子程序，乘法程序和除法程序的源程序。

3. 准备好多组运算的数据，计算运算的结果，准备进行校核。

四、报告要求

1. 整理好各个运行正确的源程序，并加上注释。

2. 整理运算的数据。

*实验七　从键盘输入十进制数的子程序的程序设计

一、实验目的

1. 进一步掌握系统调用 10 的使用方法。

2. 总结前面的输入两位十进制数的子程序的程序设计方法。

3. 建立一个能从键盘输入数值范围为≤+32767 而≥−32768 的 5 位带符号十进制数的子程序。

二、实验内容

1. 编制一个能输入数值范围为≤+32767 而≥−32768 的五位带符号十进制，转换为二进制数（补码表示）存放在一个寄存器对中的子程序。

（1）调用系统调用 10 以输入字符串。

（2）输入的最多为 5 位十进制数，也可少于 5 位(最少可为 1 位但必须有符号位)。这由系统调用 10 执行后的实际输入字符个数确定。

（3）先检查输入的符号位，然后设置相应的标志。

（4）用一个寄存器对放置累加和，令其初值为 0 。

（5）把输入的十进制数的 ASCII 码转换为 BCD 码，仍放在输入缓冲区内。

（6）采用累加和＊10＋×（其中×是指从输入缓冲区中取出的 BCD 码，最先取出的是最高位）的方法，把输入的十进制数转换为二进制数。累加和＊10＋×的循环次序由实际输入的十进制数的位数控制。

（7）若输入的是负数，则要经过取补。

（8）作为子程序就必须要有相应的保护现场和恢复现场的指令。

根据上述思路，可得如图 2.13 所示的程序流程图。

2. 编一个程序，调用上述子程序输入十进制数，接着调用上一个实验中建立的输出十进制数的子程序。

3. 调用编辑程序，输入上一个实验中最后一个程序的源文件，保留输出十进制数子程序，把其余的程序段删除；插入输入十进制数子程序。

4. 通过汇编、连接和存盘，建立机器码文件。

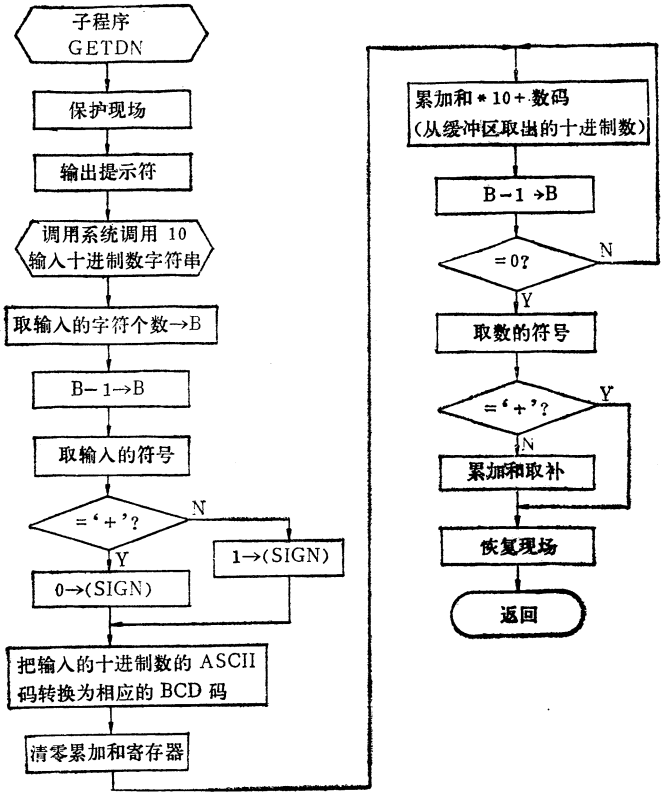


图2.13　从键盘输入范围为 $+32767 \geqslant X \geqslant -32768$ 十进制数的程序流程图

5．运行程序。

反复运行多次，检验程序输出的十进制数是否就是键入的十进制数。

6．打印源文件；保留源文件和机器码文件；把其余的文件删除。

三、预习要求

1．仔细阅读本实验指导书。

2．按照流程图在实验前编好程序。

四、报告要求

1．整理出经过运行，证明是正确的源程序，并加上注释。

2．总结几个子程序的程序设计方法。

*实验八　对输入的带符号数的数据块进行排序的程序设计

一、实验目的

1．进一步掌握系统调用 10 的应用方法。

2．进一步掌握输入十进制数子程序。

3．练习编制一些较复杂的汇编语言的程序。

二、实验内容

1．编制一个程序，能从键盘输入一个带符号数的数据块；然后用气泡排序法进行排序；再把排序的结果由 CRT 显示的程序。

（1）先输出提示符。

（2）输入数据块就要用系统调用 10。

（3）为了简化程序，规定输入数据块的格式为：

$$\pm\times\times\sqcup\pm\times\times\sqcup\pm\times\times\sqcup\pm\times\times\$$

即每一个数据最前面是符号位（正数也要有符号），然后是两位十进制数；在每一个数据之间用一个空格分隔；最后一个数据后，马上用'$'字符表示数据块的结束。

（4）数据块输入后，程序中一方面要统计输入的数据个数（等于空格数加 1）；另一方面要把两位带符号十进制数，转换为相应的二进制数，连续地存放在一个缓冲区中，以便对它们排序。

（5）每一个数据的转换方法与前面介绍的输入两位带符号十进制数子程序（或输入十进制数子程序）中的方法一样。

（6）在确定了数据块长度；又把每一个数据转换为二进制数连续存放在一个缓冲区后，就可以进行排序。我们采用在教材中介绍的气泡排序法。它的流程图和程序我们在这儿不再重复了。

（7）要把排序后的结果输出，就要采用系统调用 9。就要设置输出缓冲区。在输出缓冲区的开始设置回车与换行符。

（8）排序后的数据是带符号的二进制数，在送入输出缓冲区前，必须把它们转换成带符号（±号）的十进制数。用它们的 ASCII 码送入输出缓冲区。这个转换方法与输出十进制数子程序中的方法类似。且在每一个数据之间添上空格作为分隔。在输出缓冲区的最后还添上字符'$'。

根据上述思路，可得如图 2.14 所示的程序流程图。

2．通过编辑、汇编、连接和存盘，建立机器码文件。

3．运行程序。

反复运行多次，输入多组数据，检查输出是否正确。

4．打印源文件。

5．把整个实验中所建立的保留在盘上的所有文件全部删除。

三、预习要求

1．仔细阅读本实验指导书。

2．复习前面的输入两位带符号十进制数和输出十进制数子程序的程序设计方法。

3．根据流程图在实验前编好程序。

4．准备好几组供实验用的数据。

四、报告要求

1．整理出经过运行，证明是正确的源程序，并加上注释。

2．整理好输入与程序输出的几组数据。

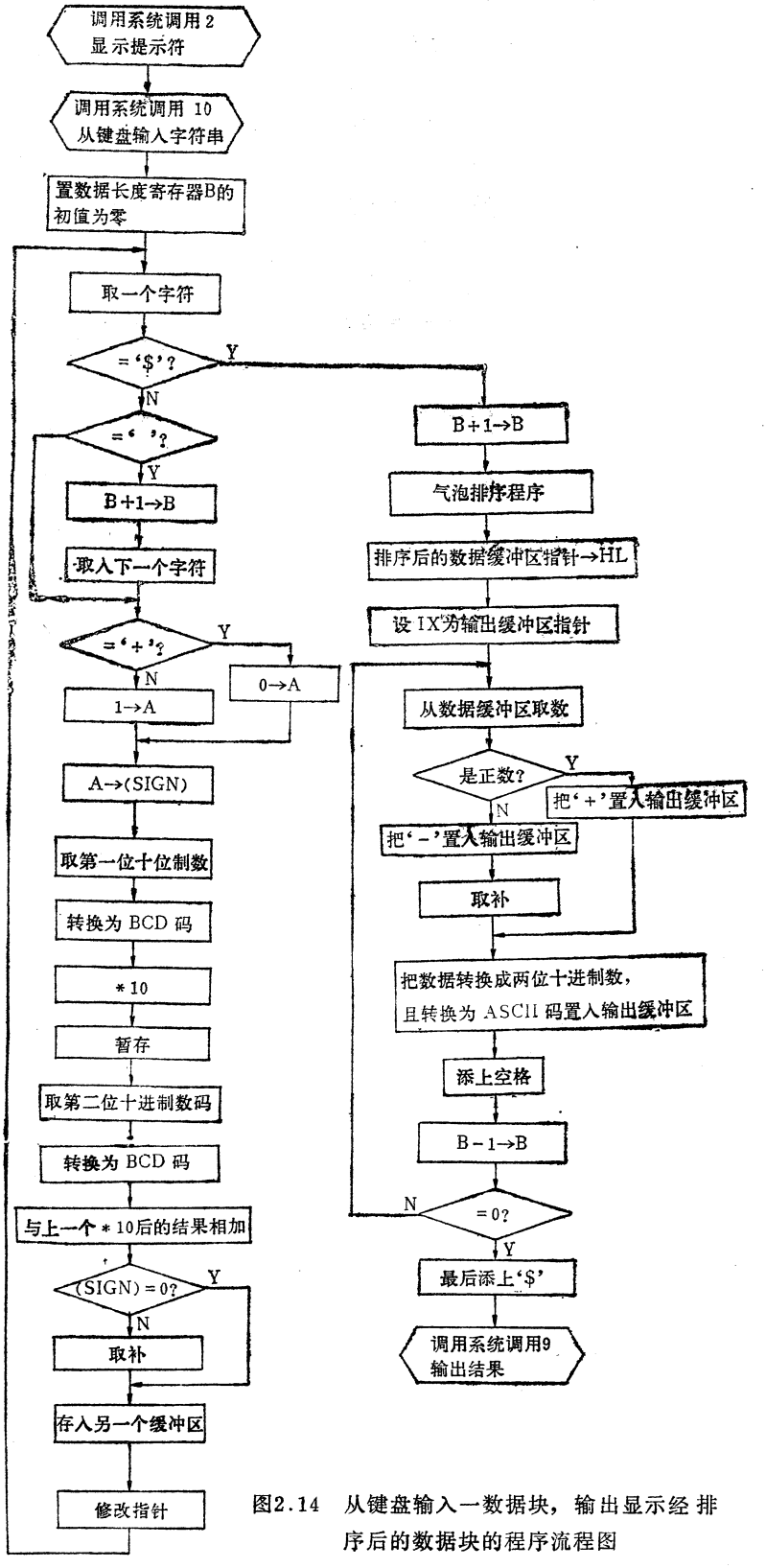


图2.14　从键盘输入一数据块，输出显示经排序后的数据块的程序流程图

3．总结汇编语言程序设计的方法。

4．对实验安排提出意见、要求和看法。

注：

如在本部分开始处提到的在 Cromemco 系统上有两个汇编程序库，一是 ASMLIB，另一是 DEMOLIB。在这两个程序库中已编好了从控制输入字符子程序 GETCH；向控制台输出字符子程序 PUTCH；向控制输出字符串子程序 LINO；从控制台输入十进制数子程序 GETDN 和向控制台输出十进制数子程序 PUTDN。而且程序设计比我们这儿要求的更为全面。但一方面在 Apple Ⅱ 的 CP/M系统中没有这两个程序库；另一方面为了练习汇编语言的程序，我们要求读者根据给出的流程图，自己编这些子程序。

第三部分　硬件和接口实验

硬件和接口实验的简便方法是用单板计算机和实验板组合起来进行。下面我们以TP801-A与TP80TS的组合为例，介绍硬件和接口实验。

第一节　TP80TS 实验板介绍

一、功能

1．TP80TS 实验板应能与 TP801-A 单板机相互间传送信息。在TP80TS 上应能获取地址总线、数据总线、CPU 的控制信号等信息。也要能把实验板上的信号送至单板机。连接。

2．应能提供实验板上的器件所需要的电源。

3．应能插放微型机的接口片子，各种中小规模集成电路片子，且能很方便地加以接连。

4．应能为集成电路片子提供输入信号。

5．应能相略地显示器件的工作状态。

二、结构

TP80TS 实验板的构结如图 3.0 所示。主要有以下几部分：

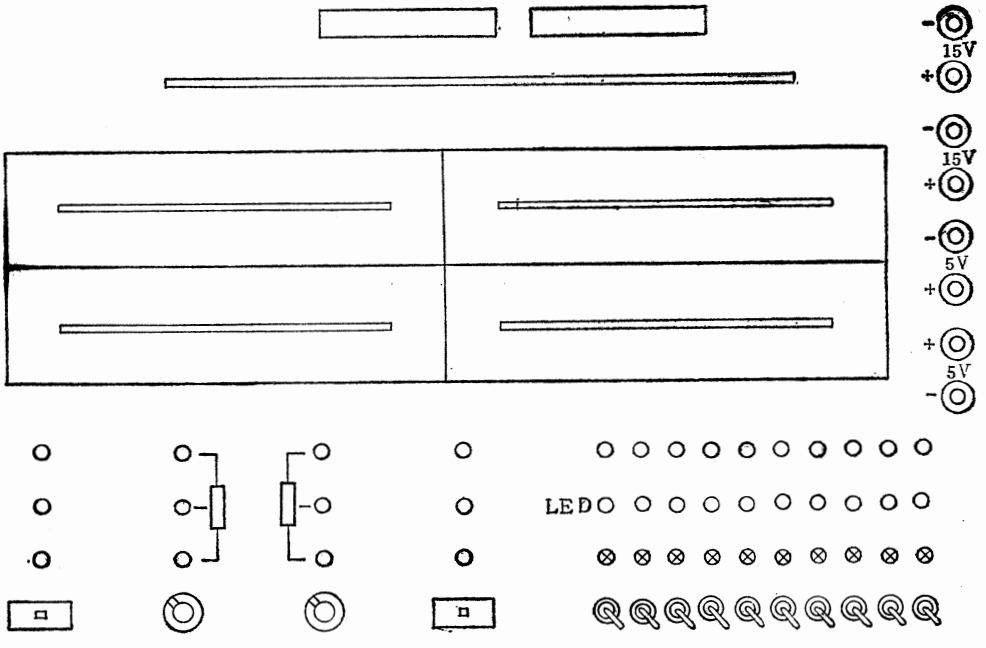


图3.0　TP80TS实验板的结构

1．在实验板母板（由印刷电路板制成）上印有与TP801-A相连接的地址总线、数据总线、Z80-CPU各控制信号线以及地址译码信号线等，共68条引线。

这些引线通过装在母板上方的两个插接件，通过扁平电缆与TP801-A相接(TP801A上也有相应引线的插接件)。

板的上方还有一排插孔，每一个插孔与上述的印制板上的信号线之一相通。所以，通过插接这些插孔，可以把这68条信号线中的任一条接至实验板的任何处，使实验者获得各种所需的信号。

2．在板中间的主体部分，安装有四块器件插座板（俗称面包板），板的结构如图3.0中所示。每块板上最上面一排孔是互相连接的，通常用来插接电源（如＋5 V）。板的最下面一排孔也是互相连接的，通常用来插接公共端（电源的地）。

板的中间横列一沟槽，以它为界上下各有47列，每一列上、下各有5个孔。某一列上方的5个孔是相通的，下方的5个孔也是相通的，而列与列之间是不通的。这些孔可用来插接双列直插式集成电路芯片，亦可插接晶体管和电阻电容等。

通常把双列直插式器件横跨在沟槽两边，由于同一列的5个孔是相同的，一个孔用于插器件，另外4个孔就可以用硬导线进行器件之间的电路连接（不用焊接），所以用于做实验是很方便的。

3．板的右下方有10个乒乓开关，由于开关的位置不同，可以供给5 V或0 V的信号电平，供实验者作为数字量输入用。

板的左下方还有两个按钮开关，也可提供5 V或0 V的信号电平，但开关有常闭和常开之分，它们的信号电平是不同的。

左下方中间有两个电位器，供电路调节使用，或提供模拟量输入。

4．板右下方，在乒乓开关的上面有10个发光二极管做成的显示灯，可用于显示数字信号。

它上面的两行插孔，分别用来插接开关和显示灯。

5．实验板的电源来自单独的电源供电，故实验板的右侧设有±5 V，±15 V共4对接线柱，可以很方便地把电源引至板上。实际上，接线柱并不与插座板插孔死接。因此，根据需要可选用任何电压的电源。

第二节　实　　验

实验一　基本逻辑电路实验

一、实验目的

1．熟悉TP80TS实验板的结构，学会使用方法。

2．学会如何使用发光二极管来显示一位二进制信息。

3．学会如何使用开关来输入一位二进制信号电平；学会消除开关颤动的影响的方法。

4．通过实验了解D锁存器与D触发器之间的区别。

5. 通过实验了解三态逻辑电路的功能。

6. 通过实验了解锁存器与缓冲器的区别。

二、实验内容

1. 发光二极管显示实验。

（1）板上有十个发光二极管显示灯，查一查它的连线情况，画出电路图，弄清它的工作原理。

（2）加上电源后把发光二极管的一端通过插孔接地，观察灯的工作情况。

（3）通过乒乓开关，得到＋5 V和0 V两种信号电平，分别把它们连至发光二极管，观察开关处在不同位置时二极管显示的情况。

（4）小结上述实验过程，得出发光二极管（以后简称LED）亮和暗的条件。若要用LED亮来代表一种二进制信息的“1”（高电平），以暗来代表“0”（低电平），应采取什么措施。

（5）把一个反相器（或用与非门74LS00，把两个输入端连在一起）的输出端，接至LED的阴极，要显示的数字信号接至反相器的输入端，如图3.1所示。

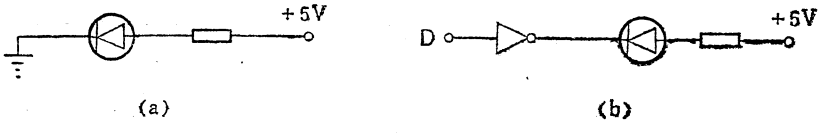


图3.1 LED指示灯的控制

用开关作为输入的数字信号。观察LED的显示情况。

保留这样的显示电路，以备下一实验用。

2. 以开关来获得数字信号的实验。

（1）一个乒乓开关，若按图3.2所示那样连线，则改变开关位置，在D端就可以得到一个数字信号，如图3.2(b)所示。

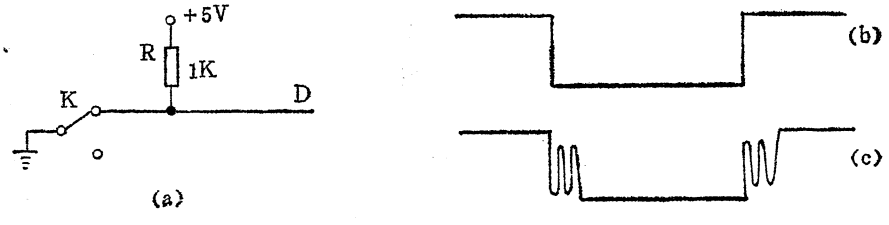


图3.2 用开关规定信号的电平

但是，实际上D端的信号不是图3.2(b)所示的理想情况，而如图3.2(c)中那样。这是因为当我们改变开关位置时，开关触点的簧片会产生颤动所造成的。这种情况是不希望的，要设法加以改进。

（2）利用基本的R-S触发器，可以消除开关颤动的影响。

基本的由两个与非门组成的R-S触发器如图3.3(a)所示。它具有两个不同的稳

定状态，这可以通过测量它的两个输出端 $\overline{Q}$ 和Q的状态而确定。在每次上电后，触发器的状态是不定的。

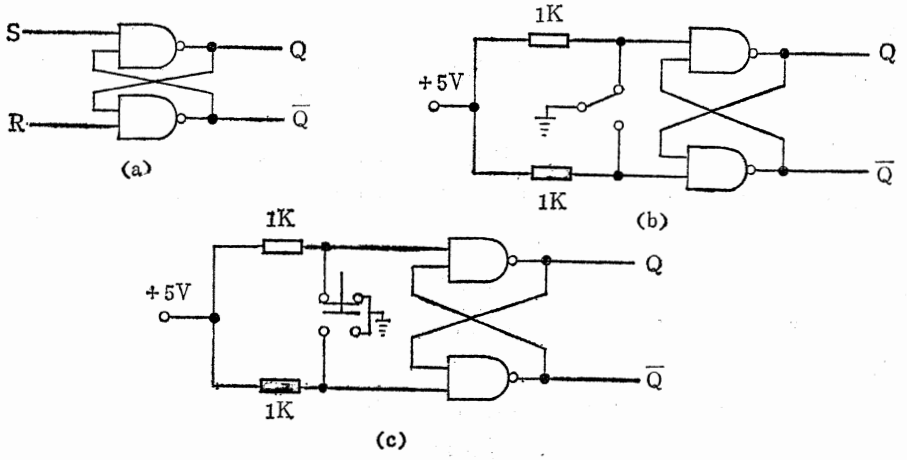


图3.3　利用R-S触发器产生数字信号，且消除开关的颤动

若把基本的R－S触发器按图3.3(b)连线，用开关作为它的输入。在图示的开关状态，则上一个与非门的输入是0电平，于是Q端的输出是高电平而 $\overline{Q}$ 是低电平；若改变开关的位置，则下面一个与非门输入为0电平，于是 $\overline{Q}$ 端输出为高电平，而Q端的输出为低电平。所以从Q端或 $\overline{Q}$ 端都可以得到数字信号（由开关的状态决定），而且由于双稳电路本身的互锁作用，即使开关有颤动，R－S触发器的状态也不会再改变。这样，在R－S触发器的输出端就可以得到如图3.2(b)所示的理想信号。

（3）若用一个按钮开关代替乒乓开关，由于它有常闭触点，当按钮未按下时，上面一个与非门的输入为0电平，则Q端为高电平而 $\overline{Q}$ 端为低电平；当按下按钮时，常闭触点打开，而常开触点闭合，则下面一个与非门的输入为0电平，于是 $\overline{Q}$ 端为高电平而Q端为低电平；当按钮释放时，常闭触点又闭合，电路恢复起始状态。所以，按一下按钮可以产生一个脉冲。

（4）用实验检查上述电路的工作情况。可用万用表测量Q和 $\overline{Q}$ 端的输出；也可用上一实验建立起来的LED显示电路来显示Q和 $\overline{Q}$ 端的状态。

3．三态缓冲器实验

在微型计算机中例如数据总线它是双向总线，要与许多个设备相联连，例如若有多个设备要把数据送至数据总线，但在任一瞬间，数据总线只能与一个设备相连通，其它的设备应处在断开状态，这就是要用到三态缓冲器。

（1）图3.4(a)所示的是74LS125中的一个三态缓冲器电路。用两个开关及相应的R－S触发器作为输入。一个连至它的数据输入端IN处，另一个连至它的控制端CNT处。用一个LED来显示输出。分别记录当CNT＝高电平时，改变输入端的信号状态时输出的显示；以及CNT＝低电平时的情况。分析和小结实验所得到的结果。

（2）按图3.4(b)连线，分别记录当控制端为高电平时以及控制端在低电平时，改变输入端的开关状态，在输出端的显示情况。

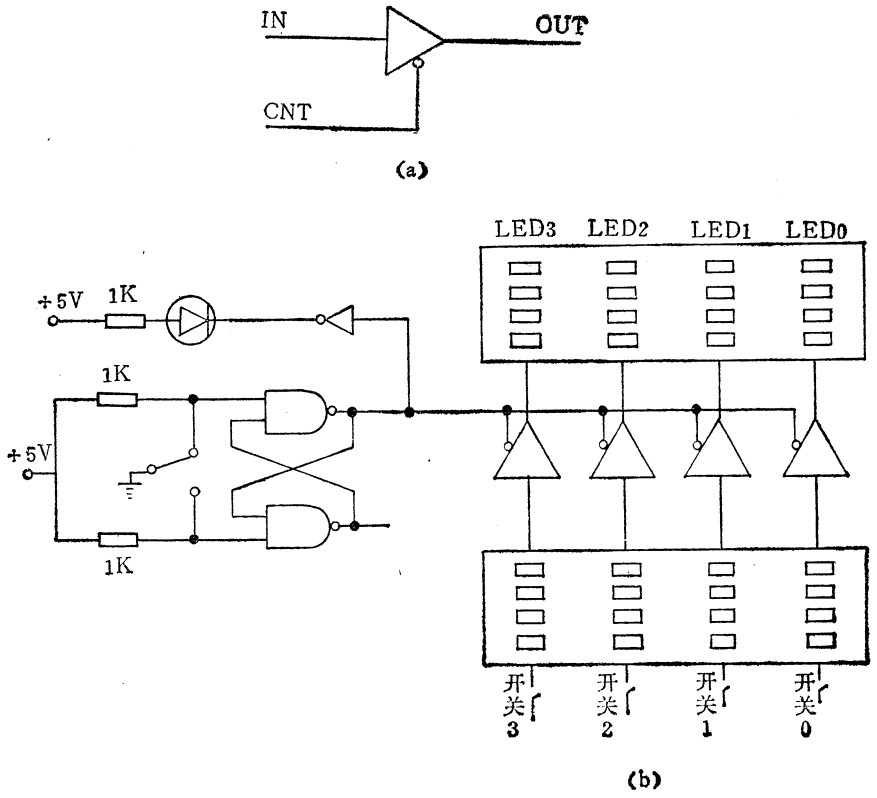


图3.4 三态缓冲器电路

4．D锁存器和D触发器实验

（1）为了暂存外部设备输至CPU的数据，或暂存CPU输给外部设备的数据，就要使用锁存器。D锁存器和D触发器都具有锁存数据的能力，都能用作为锁存器。但两者的工作原理有些区别：例如74LS75的D锁存器，有D输入端和允许（使能）端G。当G的信号有效期间（高电平有效）Q端的状态随着D端的状态变化；而当G端由高电平变为低电平时，把信号锁存，以后即使D端改变，Q端的状态不变了。

而74LS74的D触发器，是在时钟脉冲信号CK的上升沿锁存；而当CK处在低电平（稳态）或处在高电平（稳态)时，改变D端的状态，则Q端维持不变。

（2）D锁存器实验

按图3.5连线。用4个开关作为锁存器D端的输入信号，用4个LED来显示锁存器的Q端状态。用一个开关及R-S触发器的输出作为使能（ENABLE）信号连至使能端E(或G)。

先使E端处在低电平，改变各个开关的状态，观察和记录各个Q端的状态。再使E端处在高电平，改变各个开关的状态，观察和记录Q端的显示。

（3）D触发器实验

用两个74LS74片子代替图3.5中的D锁存器，其它仍按图3.5中所示的连线。

①使CK端处在低电平，改变各个开关的状态，观察和记录各个Q端的显示。

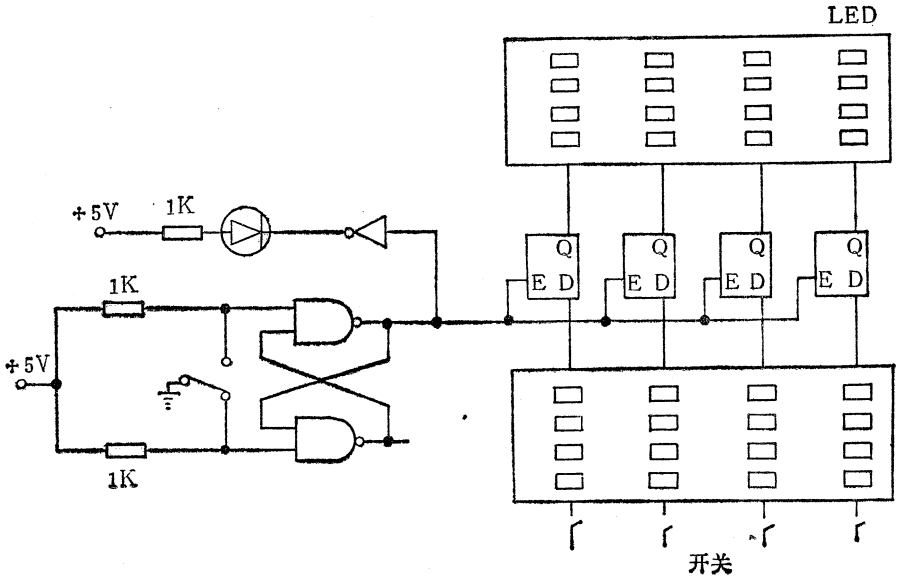


图3.5　D锁存器实验图

②使 CK 端处在高电平，改变各个开关的状态，观察和记录各个Q端的显示。

③先使 CK 端处在低电平，按照事先的要求准备好 4 个 D端的输入信号，使CK 端由低电平跳变到高电平，观察和记录各个Q端的显示。

三、本实验所需的器材

1. TP801-A单板机 1 台
2. TP80TS实验板 1 块
3. 74LS00　2 片
4. 74LS74　2 片
5. 74LS75　1 片
6. 74LS125　1 片
7. 1kΩ电阻　5 片
8. 硬导线若干

四、预习要求

1. 仔细阅读 TP80TS 实验板介绍，了解实验板的结构和使用方法。
2. 仔细阅读本实验指导书。
3. 复习数字电路课中的有关知识，对各个实验的结果预先要心中有数。
4. 准备好实验中要用的表格。

五、报告要求

1. 整理好实验中的记录表格。
2. 总结和分析D锁存器及D触发器的相同点和不同点，它们的使用方法。
3. 总结和分析锁存器与缓冲器的差别以及各自的应用场合和应用方法。

实验二　CTC的基本性能实验

一、实验目的

1. 进一步掌握实验板的使用方法。
2. 进一步了解CTC的工作原理。
3. 学会CTC的初始化编程方法。
4. 掌握 CTC 工作在计数方式下的主要性能和使用方法。
5. 掌握 CTC 工作在定时方式下的主要性能和使用方法。

二、实验内容

1. CTC 计数方式实验

（1）断开电源，按图3.6(a）接线。接好线后，经教师检查通过后合上电源。

（2）通过键盘，在单板机上输入令 CTC 工作于计数器方式，正脉冲有效，计数个数为 5 的 CTC 的初始化编程和主程序。

CTC 是一个可编程的计数器和定时器电路。在上电复位以后，必须经过初始化编程才能正常工作。通常对 CTC 的初始化编程，包括送通道控制字，送时间常数和送中断矢量这三个方面。CTC的通道控制字如下所示：

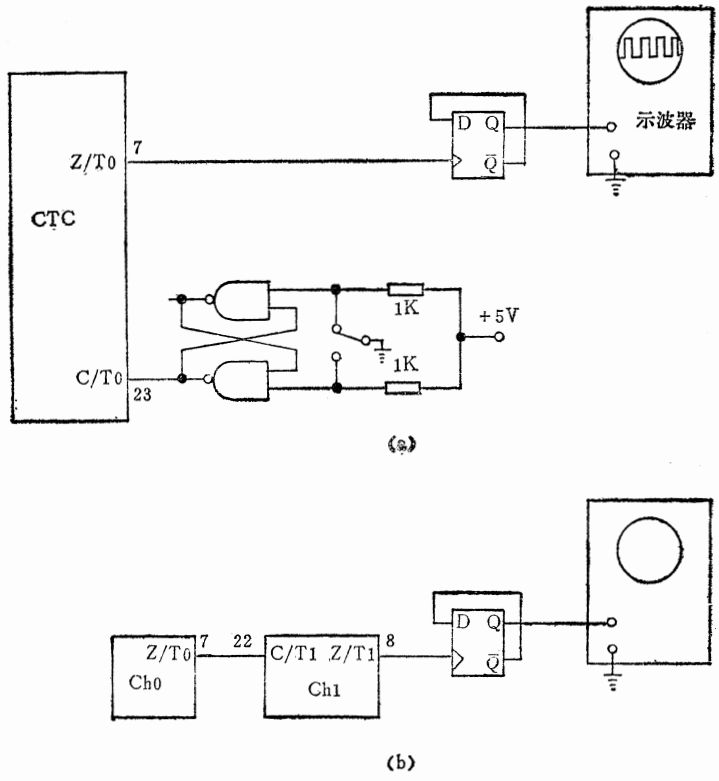
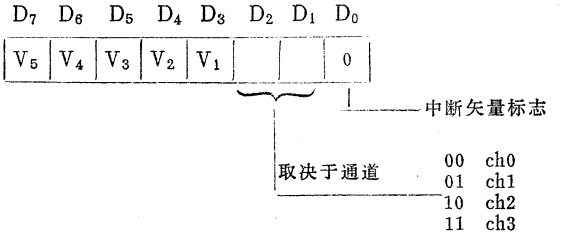


图3.6　CTC实验连线图

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
中断控制	方式控制	定标系数	边沿控制	启动方式	时 常 数	复位控制	通道控制
1 = 允许	1 = 计数	1 = 256	1 = 正	1 = 外	1 = 下跟	1 = 复位	字标志
0 = 禁止	0 = 定时	0 = 16	0 = 负	0 = 自动	0 = 无	0 = 工作	恒为 1

CTC的时间常数也是一个 8 位控制字，它实际上代表一个 8 位二进制的无符号数。00000001代表 1，11111111代表255；而00000000代表256。

CTC工作在Z80-CPU 的中断方式 2，在中断响应周期，要向 CPU 的数据总线送一个中断矢量。这个中断矢量由CTC初始化编程时规定。一个CTC片子的 4 个通道的中断矢量是互相有关连的，它们的前 5 位是相同的，D_2D_1两位由通道自动决定（若是通道 0 则送出去的必为00……），D_0必须为 0，如下所示。



其中，V_5-V_1必须在初始化编程时规定。4 个通道的初始化编程，中断矢量只要送一次，不管是哪个通道工作，都是送至通道 0（若中断禁止当然不用送中断矢量）。

（3）执行程序

改变开关，用R-S触发器的输出作为CTC 的计数脉冲，仔细观察是否输入五个脉冲后（5 个有效边沿），CTC的输出使D触发器改变状态（用示波器加以观察）。连续不断地输入脉冲，观察是否每隔五个脉冲，CTC 的输出使D触发器改变状态。

（4）改变 CTC 初始化程序中的时常数（如改为 2 或改为10），重新运行程序。重复上述的实验过程，观察CTC是否在记满了由时常数所规定的脉冲个数后输出信号。

（5）改变 CTC 初始化程序中的通道控制字，变为对输入脉冲的下降沿计数。运行程序。仔细观察是否在每输入一个负跳变时，CTC计数。

2．CTC 定时器方式实验

（1）实验的连线图仍如图3.6(a)所示。

（2）CTC编程为定时器方式，定标系数为16，自动启动；时常数为10。在单板机上输入上述程序。

（3）运行程序。观察示波器上是否出现连续波形。因为编程使CTC 为自动启动方式，则在输出时常数指令执行后，CTC 就开始定时，每当规定的定时时间到，在它的Z/T输出端，输出一个正脉冲，使D触发器翻转。由于CTC只要不命令它停止，就连续不断地重复工作，从而使D触发器不停地翻转，在示波器上显示连续波形。可以由已知的时钟频率和给定的时常数，算出 CTC 的定时时间，然后与由示波器上测得的时间相比较。

（4）改变时常数，重新运行程序重复上述实验。

（5）改变定标系数K，重新运行程序。重复上述实验。

（6）改变CTC的初始化程序，使CTC工作于于外启动（上升沿启动）方式。运行程序(启动信号处于低电平)。观察CTC是否工作；使启动信号变为高电平，观察CTC是否工作；示波器上是否出现连续波形。

（7）改变为下降沿启动，使启动信号处于高电平，运行程序。观察CTC是否工作；使启动信号变为低电平，观察CTC是否工作。

3．CTC通道级连的实验

（1）若要求定时时间为100ms（或1 s）则一个CTC通道就不能满足要求。这时可以采用软件扩展的方法；也可以采用通道级连在硬件上加以扩展。

断开电源，按图3.6(b）连线。

（2）通道级连就需要对两个通道进行初始化编程：令通道0工作在定时器方式，若定时时间为20ms，自动启动；通道1工作在计数器方式，计数次数为5。则当使通道1减为0，它的Z/T端输出一个正脉冲时，时间为100ms。

（3）输入程序；运行程序；观察和测量示波器上的时间（在定时时间更长时，可以在D触发器的输出端接一个LED显示灯，观察灯的显示)。

4．在运行中改变CTC时常数的实验

在第一部分的实验六中，曾做过一个实验，可以把某一个字符（例如一个16进制数）在单板机上的数码显示管上，连续地轮流显示。在这样的实验的程序中，是用软件来产生延时时间的。本实验中要用CTC来产生定时；而且要求在运行过程中，由程序自动来改变CTC的时常数。

（1）编一个主程序。

①CTC的初始化程序，使CTC工作于自动启动的定时器方式，中断允许，定时时间为20ms。

②用软件把CTC的定时时间扩展5倍。

③要把显示的字符的字模编码送段形锁存器。

④把数位控制字送数位锁存器。

⑤设定中断方式，开中断。

⑥动态停机，等待中断。

程序的流程图类似于图3.7(a）(更简化一些)。

（2）编一个CTC的中断服务程序。

①把用软件扩展的CTC定时的倍数减1；

②判断是否为0，非0则直接开中断，返回；

③若已为0，则表示要扩展的定时时间到，使数位控制字左移一位；

④重新置扩展倍数；

⑤开中断，返回。

程序的流程图也类似于图3.7(b)(更简化一些)。

（3）在单板机上装入程序。运行程序。观察运行结果，调试程序，直至字符能在数码管上连续地交替地显示。注意数码移动的速度。

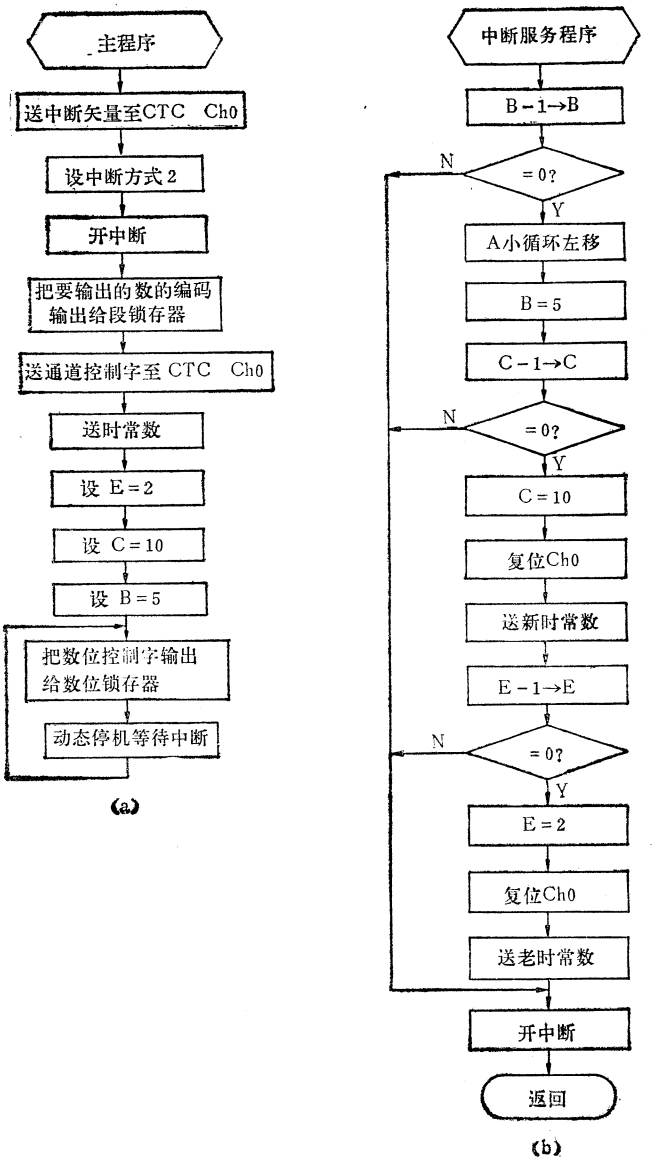


图3.7 交替改变CTC时常数的程序流程图

在以上实验中，数字移动的速度是固定的。下面我们要设法在程序运行中来改变CTC的时常数。例如，我们要求字符以一种速度移动16次；然后以另一种速度（例如更快）移动16次；又返回到前一种速度。这样，这两种速度交替地出现。

（4）编一个主程序。

这个主程序除了有上一个实验中的内容外，还要设一个计数器控制字符移动的次数；也要设置能控制两种时常数交替出现的计数器。程序流程图如图3.7(a)所示。

（5）编一个中断服务程序。

这个程序除了上一个程序中的内容以外，在扩展的定时到，使数位控制字移位以后，除了重新设置扩展倍数外，要把字符移动次数减1，然后判断是否为0（即是否已移动

16次)。若不为0，则开中断返回。若已为0，则要给CTC设置新的时间常数。当以新的时常数也移动16次以后，就返回老的时常数。

（6）改变CTC时常数的方法，可以用先使CTC复位，再送通道控制字；然后再送新的时常数，使CTC又开始工作。

程序的流程图如图3.7(b)所示。

（7）在单板机上装入程序。运行程序。调试程序，直至字符能以两种速度交替地移动。

三、实验器材

1. TP801-A单板机　1台。
2. TP80TS实验板　1块。
3. Z80-CTO　1片。
4. 74LS74　1片。
5. 74LS00　1片。
6. 1kΩ电阻　2个。
7. 双线脉冲示波器　1台。

四、预习要求

1. 仔细阅读本实验指导书。
2. 复习CTC的原理，编程方法和使用方法等。
3. 在实验前编出实验中要用的各个程序，且翻译成机器码。
4. 对各个实验，先分析预期的结果，使实验时心中有数。

五、报告要求

1. 整理出各个经过运行，证明是正确的源程序，并加以注释。
2. 总结CTC在计数方式时的性能、特点，编程方法，使用方法等。
3. 总结CTC在定时方式时的性能、特点，启动方式，编程方法和使用方法等。
4. 总结在程序运行中改变CTC时常数的方法。
5. 总结扩展CTC量程的方法。

实验三　PIO的基本实验

一、实验目的

1. 进一步熟悉实验板的使用方法。
2. 通过实验进一步理解PIO的输入方式，输出方式的工作原理，编程方法，使用方法等。
3. 进一步理解无条件传送方式，查询方式，中断方式的原理、特点和编程。

二、实验内容

1．PIO 工作在无条件传送方式的实验。

（1）断开电源，按图3.8（a）所示的连线。即以 PIO 的端口 B 作为数据输入，由开关提供输入信号；以 PIO 的端口 A 作为数据输出，用 8 个 LED 作为输出数据的显示。要注意的是 BSTB 信号要接地，这样当端口 B 工作于输入方式时，输入的数据可以锁存。

（2）编一个程序，它包括 PIO 的端口 A 和端口 B 的初始化程序。

我们已经知道 PIO 是一个可编程的通用的接口片子。通道 A 可以工作在四种不同的工作方式之下：

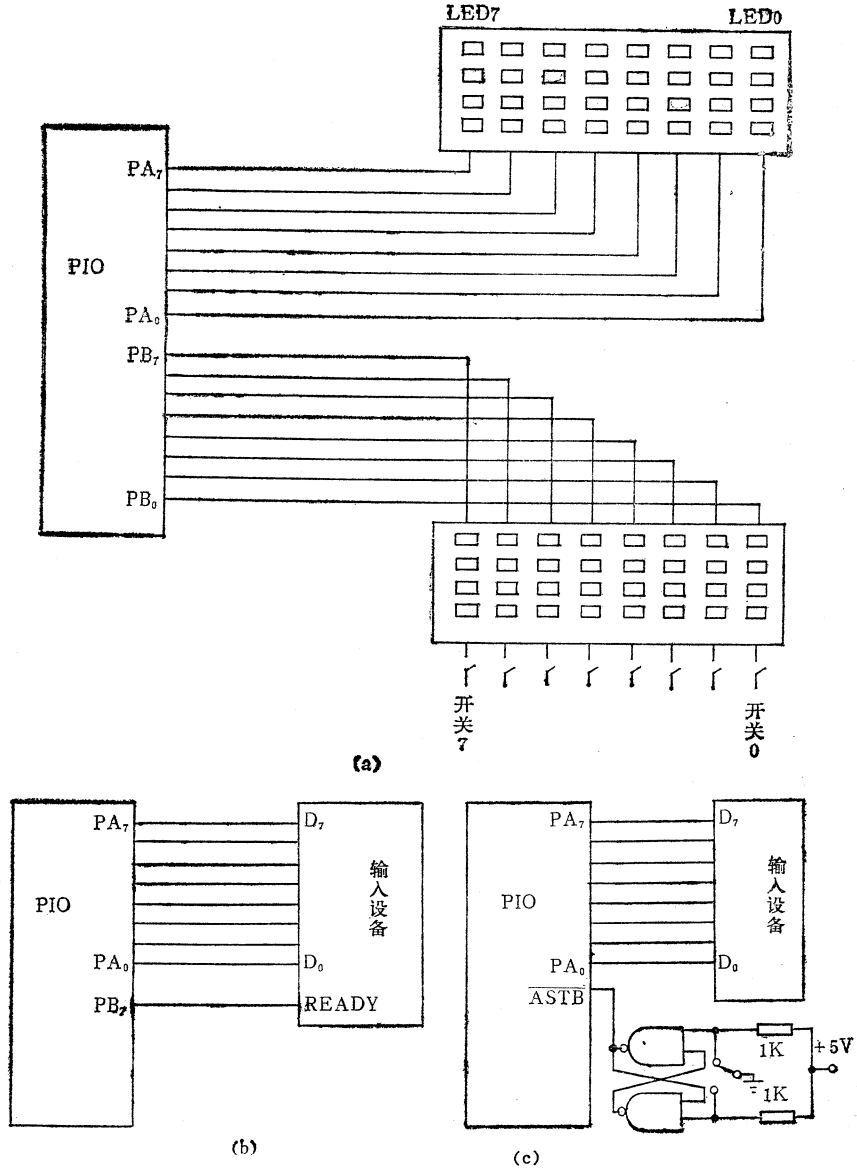


图3.8　PIO基本实验接线图

①方式 0 ——输出方式；
②方式 1 ——输入方式；
③方式 2 ——双向方式；
④方式 3 ——位控方法。

通道 B 可工作于除了双向方式以外的其它三种工作方式。

PIO 上电复位后，必须经过初始化编程才能正常工作。PIO 初始化编程的内容为：

①方式控制字：

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
M	M	×	×	1	1	1	1
00 = 方式 0 01 = 方式 1 10 = 方式 2 11 = 方式 3		任 意		方式控制字标志			

②I/O 选择字（只用于位控方式：）

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O
每一位控制相应的数据线						0 = 输出 1 = 输入	

③中断矢量：

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
V_7	V_6	V_5	V_4	V_3	V_2	V_1	0

0 —— 中断矢量的标志

④中断控制字：

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
中断控制 1 = 允许 0 = 禁止	与/或 1 = 与 0 = 或	高/低 1 = 高 0 = 低	下跟屏蔽字 1 = 有屏蔽字 0 = 无	0	1	1	1
				中断控制字标志			

⑤中断屏蔽字（只用于位控方式）：

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1/0	1/0	1/0	1/0	1/0	1/0	1/0	1/0
每一位分别控制每一条线的中断						1 = 屏蔽 0 = 允许	

方式控制字，中断矢量和中断控制字都有标志，所以在初始化编程时输送的次序可任意。而 I/O 选择字与中断屏蔽字没有标志，编程时的次序就不能任意。I/O 选择字必须跟在方式控制字的后面；中断屏蔽字必须跟在中断控制字的后面，而且在中断控制字中的 D_4 必须为 1 。

只有方式 3 才需要有 I/O 选择字和中断屏蔽字。若中断禁止，当然就没有必要再送中断矢量和中断屏蔽字了。

（3）编输入输出程序。

①由端口 B 输入数据，由端口 A 输出显示，并送内存保存。

②由端口 B 输入数据，取反后由端口 A 输出显示。

③由端口 B 输入数据，编一个软件延时子程序，延时时间约100ms，过100ms 后再从端口 B 输入数据；连续输入 5 个数据后，取中值(去掉最小值和最大值其他三值平均)，再由端口 A 输出。

2．PIO 工作于查询方式输入的实验

（1）断开电路，按图3.8(b）所示那样接线。用端口 A 的 PA_7—PA_0 作为数据输入，仍可用开关模拟输入设备的数据。但要工作于查询方式，输入设备必须还有一个状态信号 Ready，它通过 PB_7 输至 CPU。用开关通过 R-S 触发器来模拟 Ready 信息，通常情况下它处于低电平。 $\overline{\text{ASTB}}$ 信号接低电平。

（2）编一个程序，包括：

①PIO 端口 A 工作于输入方式的初始化程序（中断禁止)；端口 B 工作在输入方式的初始化程序。

②利用查询方式输入10个数送至内存缓冲区的程序。

③输出显示程序，把最后输入的一个数，送单板机数码显示管的右面两位显示。显示程序与第一部分单板机实验的实验六类似。

（3）在单板机上装入程序。运行程序。只要 Ready 信息处于低电平，则程序就循环等待。根据事先准备好的要输入的数据，拨动好开关（8 位数据)。然后使 Ready 信息变为高电平，又恢复至低电平。这样就输入一个数据；照同样方法重复10次。观察在单板机的数码管上是否显示最后输入的这个数据。调试程序，直至在内存中的10个数就是输入的数；数码管显示最后一个数。

3．PIO 工作于中断方式的实验

（1）断开电源，按图3.8(c)所示那样接线。端口 A 的 PA_7—PA_0，仍通过开关输入数据。用开关及 R-S 触发器的输出， 连至 $\overline{\text{ASTB}}$，通常处于高电平。

（2）编一个程序，包括：

①PIO 端口 A 工作于输入方式，中断允许，要输入中断矢量的初始化程序。

②主程序：设置中断方式，开中断；程序进入显示循环（显示最右面两个显示管的显示缓冲区中的内容，初始值置为00)。

③中断服务程序，主要是从 PIO 的端口 A 输入数据，送至右面两个显示管的显示缓冲区。

（3）在单板机上装入程序，运行程序。应该看到右面两个显示管显示00。

根据事先准备好的要送入的数据，安排开关的状态；产生一个有效的 $\overline{\text{ASTB}}$ 信号

(负脉冲)。观察显示器的显示。调试程序直至显示正确。

输入一组数据，观察是否在有$\overline{\text{ASTB}}$信号后CPU才读入；在没有$\overline{\text{ASTB}}$信号时CPU在运行什么?

三、实验器材

1. TP801-A单板机　　1台
2. TP80TS实验板　　1块
3. PIO　　1片
4. 74LS00　　1片
5. 电阻（1kΩ）　　2个

四、预习要求

1. 仔细阅读本实验指示书。
2. 复习PIO的工作原理、编程方法和使用方法。
3. 根据所提要求，在实验前编好各个实验程序，并翻译成机器码。
4. 准备好所要输入的数据和记录表格。
5. 准备好每个实验的预期结果，以做到心中有数。

五、报告要求

1. 整理好每个经过运行，证明是正确的源程序，并加上注释。
2. 整理好数据表格。
3. 总结PIO的编程方法和使用方法。
4. 总结和分析无条件传送方式、查询方式、中断方式的特点，及其应用场合，使用方法。

*实验四　PIO位控方式实验

一、实验目的

1. 进一步了解PIO的工作原理。
2. 熟悉PIO在位控方式下的编程方法和使用方法。
3. 学会当PIO在位控方式有多个中断源时，转入中断服务程序的处理方法。

二、实验内容

1. 位控方式下输出开关量的实验。

（1）断开电源，按下列要求接线。

利用PIO的端口A的两条数据线输出两个开关量。一个用来控制一个蜂鸣器；另一个用来控制一个显示灯LED。

（2）编一个程序，使蜂鸣器按一定的频率奏鸣。程序中包括：

①PIO 端口A工作于位控方式的初始化编程（但中断禁止）。

②要使蜂鸣器发出响声，就要求蜂鸣器有电流（若输出的信息为1）和无电流（输出的信息为0）交替变化。

③有电流和无电流变化的速度就是蜂鸣器的频率。

（3）在单板机上装入程序，运行程序。观察蜂鸣器是否发出响声。调试程序，直至运行正确。

（4）改变输出程序中的延时时间，观察蜂鸣器的频率是否跟着变化。

2．端口A输出控制指示灯闪动的实验。

（1）编程序，包括：

①PIO 工作于位控方式，中断禁止的初始化程序。

②使 LED 按每秒5次的速率闪动的程序。

（2）在单板机上装入程序，运行程序，直至结果完全正确。

3．位控方式下有多个中断源的实验

（1）断开电源，按下列要求接线：

①端口A的接线不变，一条线控制一个蜂鸣器；另一条线控制一个指示灯。

②端口B的两条数据线，接至两个故障源。每个故障源用一个开关及R-S触发器来模拟，通常，它们处于低电平。

（2）编一个主程序包括：

①端口A工作于位控方式，有两条线定为输出，中断禁止的 PIO 的初始化程序。

②端口B工作于位控方式，有两条线定为输入，且这两条线中的任一条处于高电平时能引起中断的 PIO 的初始化程序。

③循环显示程序，在单板机的右面两个显示管上显示00。

（3）中断服务程序。现在通道B有两个中断源，不论是哪一个中断源有请求，都由通道B向 CPU 发出中断请求信号；在中断响应周期，通道B把在初始化编程时规定的中断矢量送至数据总线。因而，不论是哪一个源有中断请求，都转至一个共同的入口。因而必须在中断服务程序中查询，以确定是哪一个中断源的请求，以及区分中断优先权。能满足上述要求的中断服务程序的流程图如图 3.9 所示。

（4）中断源1的服务程序。它使指示灯 LED 以每秒5次的速度闪动50次。程序的流程图如图3.9(c)所示。

（5）中断源2的服务程序。它使蜂鸣器以一定的频率（例如500Hz）响50次。程序流程图如图3.9(d)所示。

（6）在单板机上装入如上的主程序，中断服务程序，中断源1的服务程序和中断源2的服务程序。运行程序。

在运行正常的情况下，没有中断请求时，应在显示管上显示00。

让中断源1变为高电平，观察 CPU 是否接收中断，能否转至中断源1的中断服务程序。此时，数码管的显示停止，LED 应按要求的次数和速度闪动。观察中断服务结束后能否返回主程序——恢复数码管的显示。

让中断源2变为高电平，观察 CPU 能否接收中断，能否转至中断源2的服务程

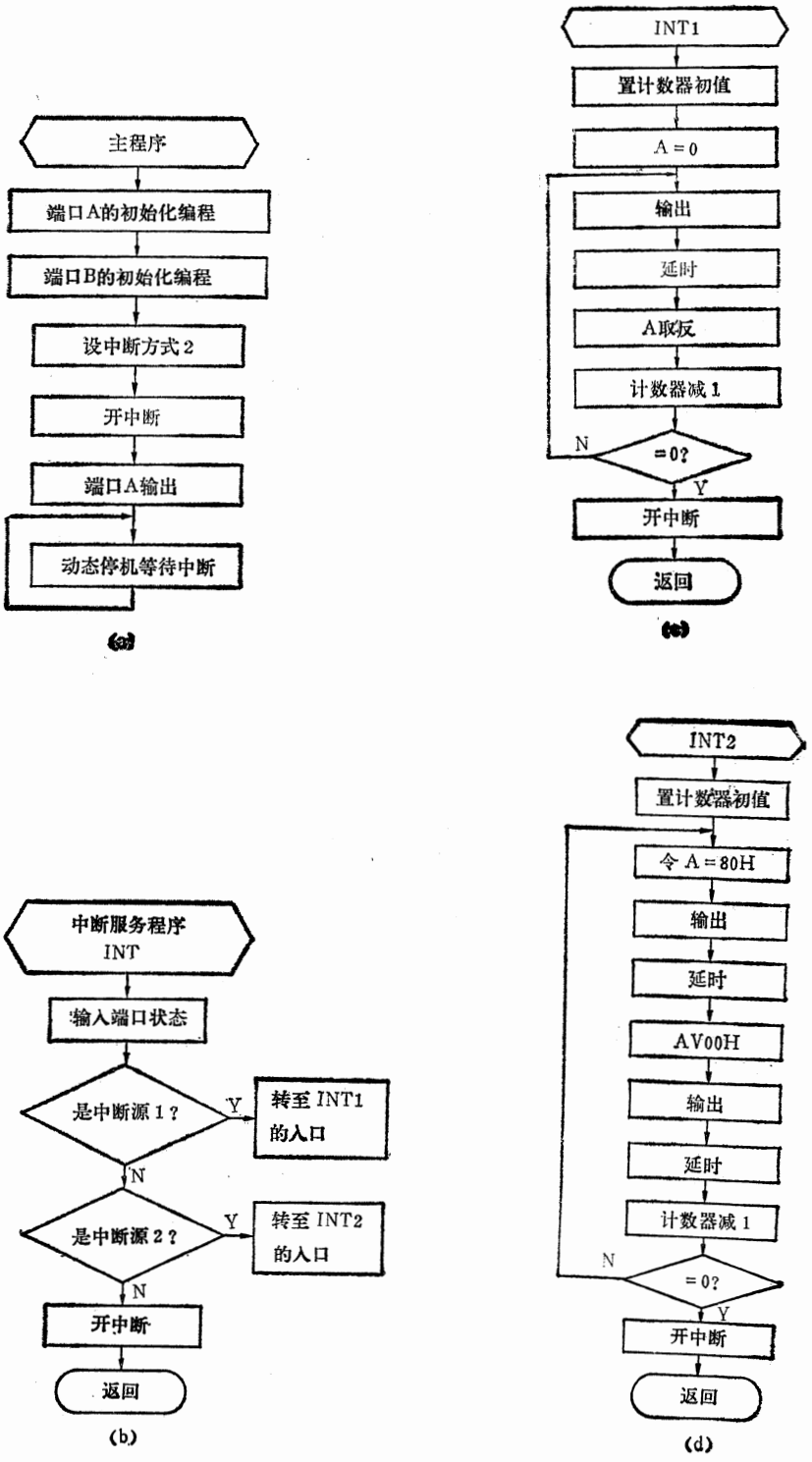


图3.9　PIO位控方式实验的主程序及中断服务程序

序——蜂鸣器以一定的频率响若干次。观察中断结束后能否返回主程序。

让两个中断源同时有效，观察能否转至中断优先权高的源的中断服务程序。

三、实验器材

1. TP801-A 单板机　1 台。
2. TP80TS 实验板　1 块。
3. PIO　1 片。
4. 压电蜂鸣器　1 个。
5. 74LS00　1 片。
6. 1kΩ电阻　6 个。

四、预习要求

1. 仔细阅读本实验指导书。
2. 复习中断的概念。
3. 按所提要求或根据流程图在实验前编好各个源程序，并翻译成机器码。
4. 估计每一个实验的预期结果，做到心中有数。

五、报告要求

1. 整理好每个经过运行而证明是正确的源程序，并加上注释。
2. 整理实验的结果。
3. 总结 PIO 位控方式的特点，编程方法和使用方法。

*实验五　Z80-CTC应用实验(电子计时秒表)

一、实验目的

1. 进一步了解 Z80-CTC 的原理和功能。
2. 掌握将 Z80-CTC 设置为定时器和计数器工作方式的程序设计方法。
3. 了解用 Z80-CTC 产生实时时钟的方法。
4. 掌握 Z80-CTC 定时和计数产生中断请求，转入中断服务程序的程序设计方法。

二、实验内容

本实验主要是应用 Z80-CTC，通过程序设计，实现一个电子计时秒表。作为一个秒表，一方面要能产生定时时钟，能显示1/10秒、秒和分，能进行计时。另一方面，要能像秒表一样进行控制：即按一次，能开始计时，以1/10秒为最小计时和显示单位；按第二次能停止计时，固定显示已计时的值；按第三次能返回起始状态（清除显示），完成一次秒表工作的全过程。再按按钮时能重复上述过程。

要达到上述要求，就要由 CTC 和程序设计相结合来实现。

1. CTC 通道的分配。

根据实验要求，要用两个CTC通道，一个利用通道1，工作于定时器方式，适当选择时间常数，使每隔1/100秒在它的Z/T1输出端输出一个脉冲，同时向CPU发中断请求。

另一个用通道0，工作于计数器方式，设置的时常数为1。计数脉冲由按钮开关及相应的R-S触发器模拟。每按一次按钮，通道0发一次中断请求，在中断服务程序中记录按钮次数，由主程序根据按钮次数，转入不同的处理程序加以处理。

能实现上述要求的电路图如图3.10所示。

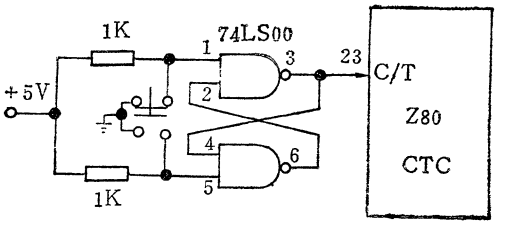


图3.10　CTC电子秒表实验的接线图

2．断开电源，按图3.10接线。

3．编一个主程序。包括：

（1）CTC通道0工作在计数方式的初始化编程。于是CTC通道0就等待着脉冲输入。

（2）因为按钮按的次数不同有不同的处理，所以要设置一个按钮次数计数器。它的初值应清0。

（3）在没有让秒表计数的情况下，显示的值应全为0。

（4）然后程序进入了检测按钮是否按了一次的循环，等待按钮输入。

（5）若按了一次按钮，应该计时，就通过向CTC通道1进行初始化编程来启动定时。程序应显示计时值（在CTC通道1的中断服务程序中进行）。

（6）然后程序就进入检测按钮是否按上二次的循环，等待第二次按钮输入。

（7）若检测到已按两次按钮，则应停止CTC通道1的定时，但计时的值不变，且不断地显示。

（8）然后程序就进入检测按钮是否按三次的循环，等待着第三次按钮输入。

（9）在输入了第三次按钮后，程序返回初始状态，开始新的循环。

程序的流程图如图3.11(a)所示。

4．编一个CTC通道0的中断服务程序。

这个服务程序很简单，它使按钮次数计数器的值加1，然后开中断返回。在主程序中，根据按钮次数计数器的值进行不同的处理。

5．编一个CTC通道1的中断服务程序。

要求显示的计时的值为1/10秒、秒和分。就需要有相应的计数器，它们的初值为0。CTC的定时时间规定为1/100秒，每过1/100秒通道1发中断请求，在中断服务程序中使1/10秒计数器加1，然后判断计数值是否已为10。若未到10，则转显示程序，显示现行的分、秒、1/10秒值。若已达到10，则使1/10秒计数器清0，秒计数器加1，再判断

秒计数器的值是否已达60。若未到60，则转显示程序；若已达60，使秒计数器清 0，分计数器加 1，然后也转显示程序（假定秒表的量程小于 1 小时）。

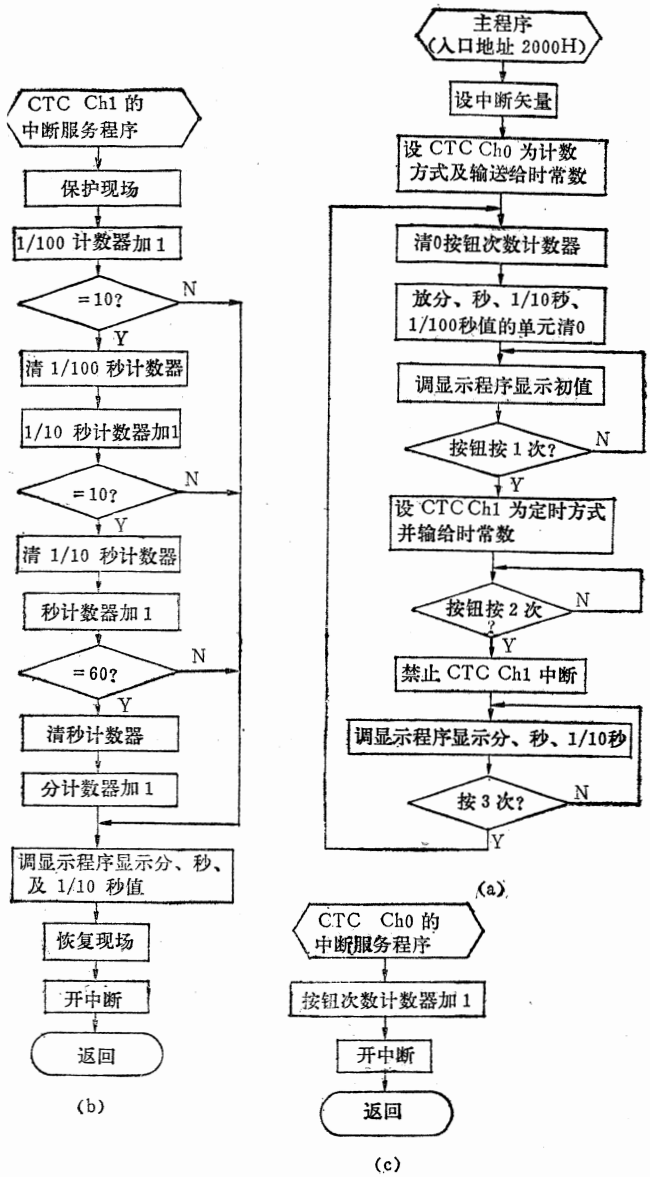


图3.11　电子计时秒表程序流程图

程序流程图如图3.11(c)所示。

6. 在单板机上装入主程序和两个中断服务程序。运行程序。

在未按按钮前应显示全 0 。按一次按钮后，应开始计时并显示。

按第二次按钮，应该停止计时。校核显示的计时值是否正确。

按第三次按钮，显示应清 0 。然后可以接着上述的循环。

三、实验器材

1. TP801-A单板机　1 台

2. TP80TS实验板　　1块
3. 74LS00　　1片
4. 1kΩ电阻　　2个

四、预习要求

1. 仔细阅读本实验指导书。
2. 复习 CTC 的原理、编程方法和使用方法。
3. 根据给定的流程图，在实验前编出主程序及中断服务程序。

五、报告要求

1. 整理好各个经过运行，证明是正确的源程序并加上注释。
2. 总结 CTC 的编程方法和使用方法。

*实验六　Z80-PIO应用实验(交通信号灯实时控制)

一、实验目的

1. 掌握将 Z80-PIO 设置为输出和位控工作方式的程序设计方法。
2. 掌握将 Z80-PIO 设置为允许中断或禁止中断的程序设计方法。
3. 掌握中断嵌套和非正常从中断返回的程序设计方法。
4. 掌握软件延时的程序设计方法。

二、实验内容

假设有一个十字路口，一条是大道，一条是小道。每个路口各有红、黄、绿三个交通信号灯。其中大道是主干线，经常地有大量的车辆通过；而小道只是偶然地有少量车辆通过。我们用红、黄、绿三个彩色发光二极管模拟控制交通的信号灯；用按钮和乒乓开关模拟车辆通过十字路口时的感受元件。

编一个程序，能按下述规则控制信号灯的转换。

(1) 通常情况下，大道总是绿灯，小道总是红灯。

(2) 若检测到小道来车后，经过 10 秒钟使小道变为绿灯，则大道应变为红灯。但为了使大道的车由动到停有个准备时间，在大道的灯变红之前，要有 4 秒钟的黄灯时间。即小道来车后，经过 6 秒大道由绿灯变为黄灯，再过 4 秒，大道由黄灯变为红灯，在这同时小道由红灯变为绿灯，如图 3.12(a)所示。

(3) 在小道变红灯后，如果大道没有来车，或来车但不到三辆。则经过 25 秒小道变为黄灯，再经 4 秒小道由黄灯变为红灯，在这同时，大道由红灯变为绿灯，如图 3.12(b)所示。

(4) 如果在小道绿灯期间，检测到大道已来三辆车，虽然小道亮绿灯的时间还不到 25 秒，当第三辆车到达后，小道立即换为黄灯，经过 4 秒，小道由黄灯变为红灯，同时大道由红灯变为绿灯，如图 3.12(c)所示

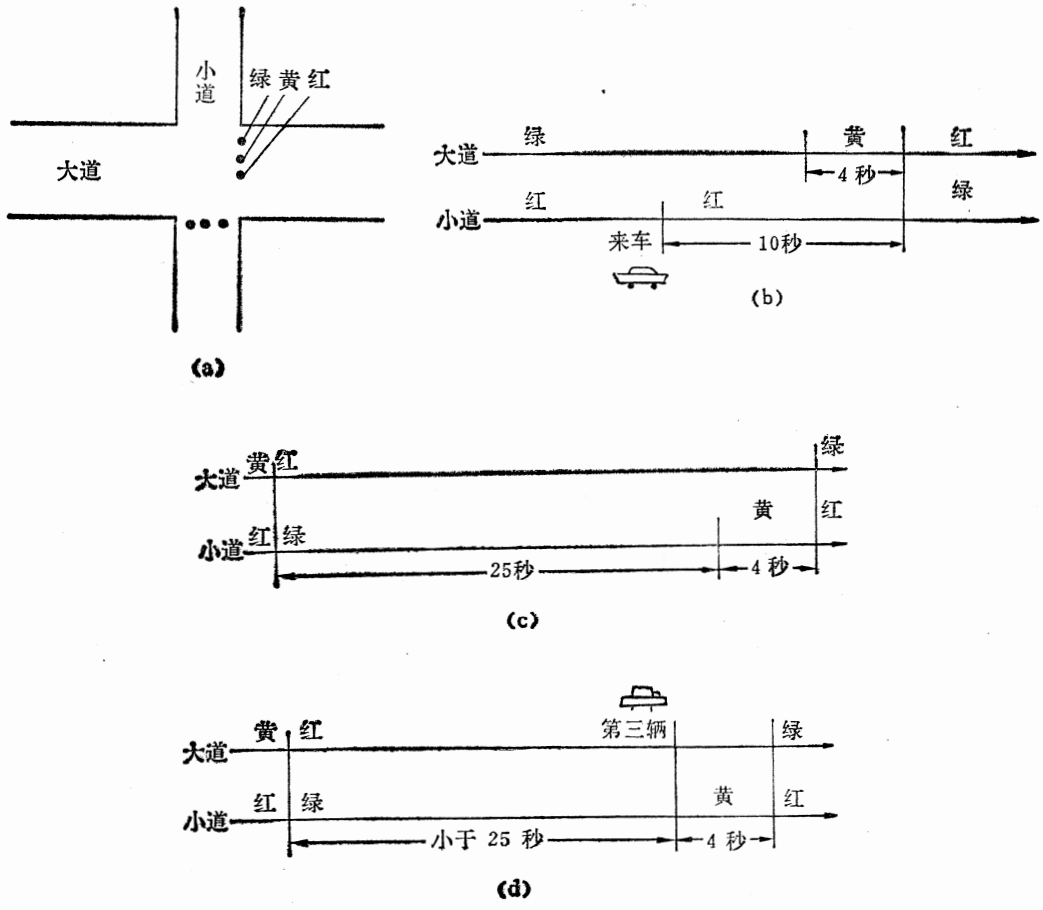


图3.12　交通信号灯转换示意图

要达到上述要求，实验的步骤如下：

1．断开电源，按图 3.13 所示连线。

（1）利用 PIO 工作在输出方式来控制大、小道的各三个交通信号灯。

（2）用一个开关及 R－S 触发器来模拟小道的车辆。开关每改变一下状态，代表小道来了一个车辆，要求改变大道绿灯小道红灯的状态。这由中断服务程序来实现。所以，开关状态的变化要能引起中断。这可以由把信号直接连至 BSTB端来实现。

（3）大道来了三辆车的检测，可以用计数器来检测。为了练习，让 PIO 端口 A 工作于位控方式，每个车作为一个中断源（用开关模拟）。只有当三个车都到来时(三个中断源相与）才发中断请求，才作相应的处理。

2．编主程序。

（1）编 PIO 端口 B 工作于输出方式的初始化程序。

（2）设置中断方式，开中断。

（3）程序使大道绿灯亮，小道红灯亮的程序循环，等待着小道来车。

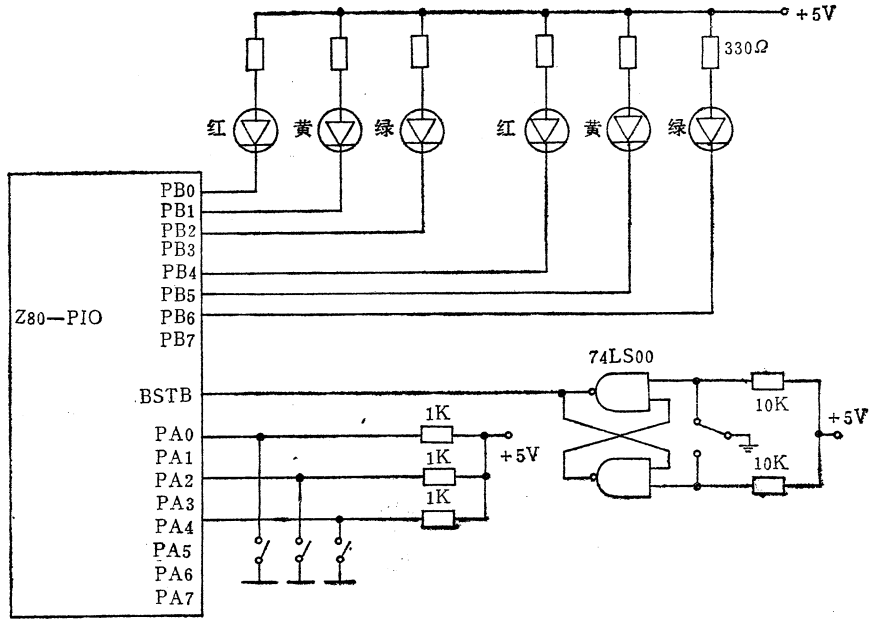


图3.13　PIO 用作交通信号灯的实验接线图

要控制交通信号灯中哪些亮，需要由端口 B 输出不同的 8 位数。若按图 3.13 中那样连接，则输出的数据组合如下：

大道	绿	黄	红	红
小道	红	红	绿	黄
编码	63H	65H	36H	56H

主程序的流程图如图 3.14(a)所示。

3．编 PIO 端口 B 的中断服务程序。

这是小道来车后的处理程序，它首先延时 6 秒（延时程序用软件实现，可调用单板机监控程序中的 20ms 延时子程序），使大道由绿灯变为黄灯（小道仍为红灯）；再延时 4 秒使大道由黄灯变为红灯，而小道由红灯变为绿灯。

其次就要监视大道是否有三辆车到达，这首先要对 PIO 端口 A 进行初始化编程。另一方面要延时 25 秒。在这 25 秒延时时间内大道没有来三辆车，则使小道由绿灯转为黄灯（大道仍为红灯），再延时 4 秒返回主程序。由于主程序处于显示大道绿灯小道红灯的循环，程序就可以循环进行。

程序的流程图如图 3.14(c)所示。

4．编 PIO 端口 A 的中断服务程序。

这是在小道亮绿灯的 25 秒时间内，检测到大道来了三辆车的处理程序，是一种中断嵌套的情况（在小道来车的中断服务程序中，又有了大道有三辆车的中断的情况）。而它的要求是提前结束小道绿灯的状态，立即使小道由绿灯转为黄灯，再经过 4 秒，恢复大道绿灯小道红灯。所以这一处理程序并不需要做任何实质性的操作，而是使 CPU 脱离 25 秒的软件循环，返回主程序。程序的流程图如图 3.14(b)所示。

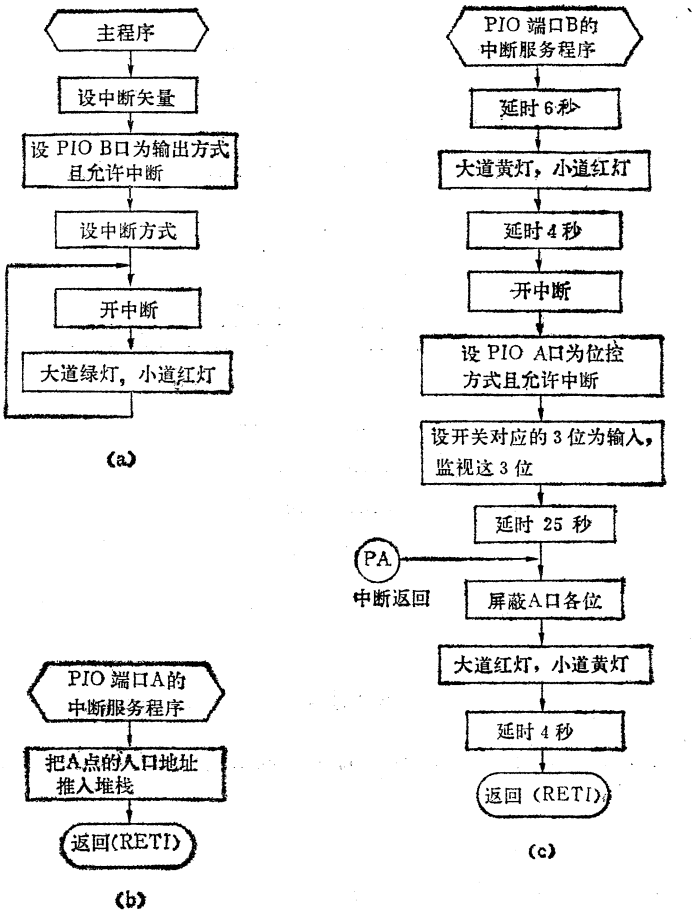


图3.14　PIO 用作交通信号灯控制的程序

5．把主程序和两个中断服务程序装入单板机，运行程序。

主程序运行正确的话，应显示出大道绿灯亮．小道红灯亮。

改变开关状态，模拟小道有车，观察各次灯显示的变化情况是否符合要求，经 25 秒后是否返回起始状态。

再一次模拟小道有车，待小道绿灯亮后，模拟大道有两辆车，观察灯显示的变化情况（应与前面的情况相同）。

再一次模拟小道有车，待小道绿灯亮后，立即模拟大道有三辆车，观察是否提前转回到大道绿灯小道红灯的情况。

三、实验器材

1．TP801-A 单板机　　1 台

2．TP80TS 实验板　　1 块。

3．74LS00　　1 片。

4．1K电阻　　2 个。

四、预习要求

1．仔细阅读本实验指导书。

2．复习 PIO 的原理、编程方法和程序设计方法。

3．根据流程图在实验前编好各个源程序，并翻译成机器码。

4．拟定实验步骤。

五、报告要求

1．整理好经过运行而证明是正确的各个源程序，并加上注释。

2．总结 PIO 各种工作方式的编程方法和使用方法。

3．总结中断嵌套的编程方法。

4．考虑如下思考题：

（1）在 PIO 端口 B 延时到 25 秒之后，为什么要将端口 A 的中断屏蔽？如果不这样做会有什么问题？

（2）如果用 LD A，83H 和 OUT(PIOAC)，A 这两条指令来代替，使端口 A 中断禁止是否可行？会出现什么现象？

（3）PIO 端口 B 中断服务程序处理完之后是否可以直接从中断返回？会出现什么问题？

（4）PIO 端口 A 中断跳过端口 B 延时 25 秒程序，能否用转移指令实现？这样做会有什么问题？

（5）RETI 指令与 RET 指令有什么相同处，有什么不同处？能否互换？

（6）是否可用其它方案（例如 CTC 计数方式）以满足监视大道上是否已有三辆车？试画出硬件连接图和软件流程图。

*实验七　A/D转换实验

一、实验目的

1．了解 A/D 转换片子 ADC0809 的工作原理。

2．掌握 A/D 转换片子与 CPU 接口的方法。

3．掌握实现 A/D 转换的程序设计方法。

二、实验原理

A/D 转换芯片 ADC0809 的结构方框图如图 3.15 所示。

片子的核心是一个逐次逼近式的 A/D 转换电路。输入端有一个 8 路模拟开关，可以轮流输入 8 个模拟量（由 IN0 至 IN7 输入的）。8 路的模拟开关由三位数字信号控制，这三位数字信号由 CPU 输给，内部有一个三位锁存器加以锁存。三位锁存器的输入线 ADDA—ADDC 通常接至数据总线，另外还要由锁存允许信号来选通锁存器。

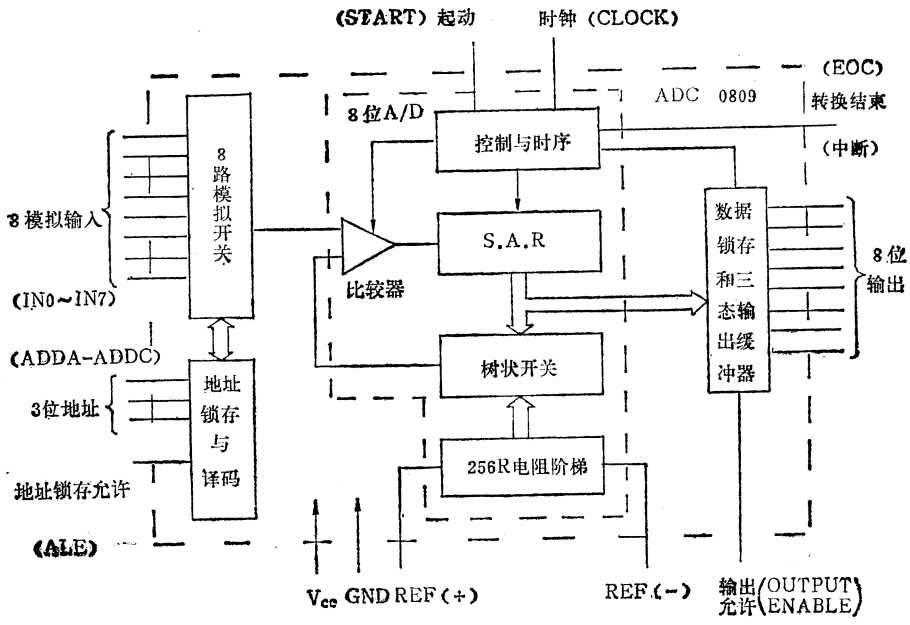


图3.15 ADC0809的结构框图

A/D转换要由一个命令信号接至它的START端使其启动；而A/D转换是否完成，由信号EOC（转换结束）来表示。

为了使ADC0809能与CPU直接接口，转换以后的数据由输出锁存器锁存，并通过三态缓冲器后输出，因而可直接接至数据总线上。输出的三态缓冲器由输出允许信号(OUTPUT ENABLE）控制。

当ADC0809与CPU相接口时，先要由CPU输出三位锁存信号，以选择输入哪路模拟量。其次，CPU用命令使ADC启动转换。然后就要检测A/D转换是否完成。在确认A/D转换已经完成的情况下，打开它的三态缓冲器电路，输入数据。ADC0809在典型应用情况下的接线如图3.16所示。

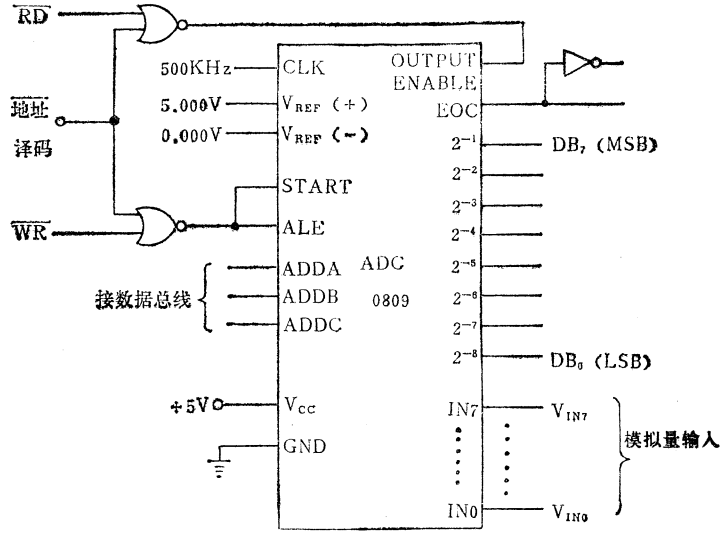


图3.16 ADC0809的典型使用方法

在本实验中，为了突出实验中应考虑的主要问题，就不考虑多路模拟量输入的问题。把模拟量直接接至IN0，而把ADDA、ADDB和ADDC接地。本实验中的接线图如图3.17所示。在命令A/D转换启动后，由EOC信号向CPU发中断请求，在中断服务程序中输入数据，一方面通过PIO端口B用LED显示，另一方面用数码管显示。

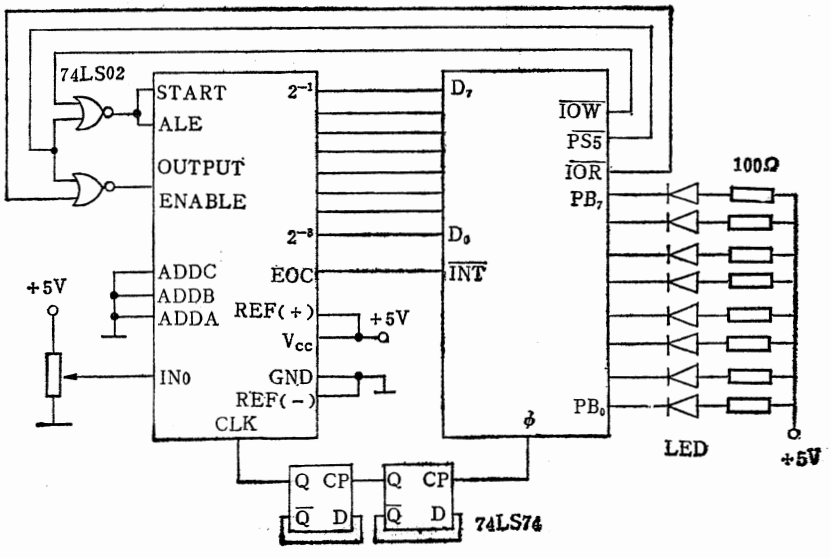


图3.17　A/D实验接线图

三、实验内容

1．断开电源，按图3.17所示接线。

2．编主程序。

（1）PIO端口B输出方式的初始化编程。

（2）设中断方式1。由于ADC0809不是Z80系列的片子，不能工作于中断方式2；ADC0809也不能产生RST指令，也不能工作于中断方式0；而且在实验中也只有一个中断源，所以，用中断方式1是最简单方便的。

（3）开中断。

（4）启动A/D转换，然后程序就进入循环，等待中断。

主程序的流程图如图3.18所示。

3．编中断服务程序

（1）输入A/D转换后的数据。

（2）由PIO端口B输出，由LED显示灯显示。

（3）把输入的数据送至显示缓冲器，调用显示程序，在单板机的右面两位数码管上显示输入的数据。

（4）开中断，返回。

中断服务程序的流程图如图3.18(b)所示。

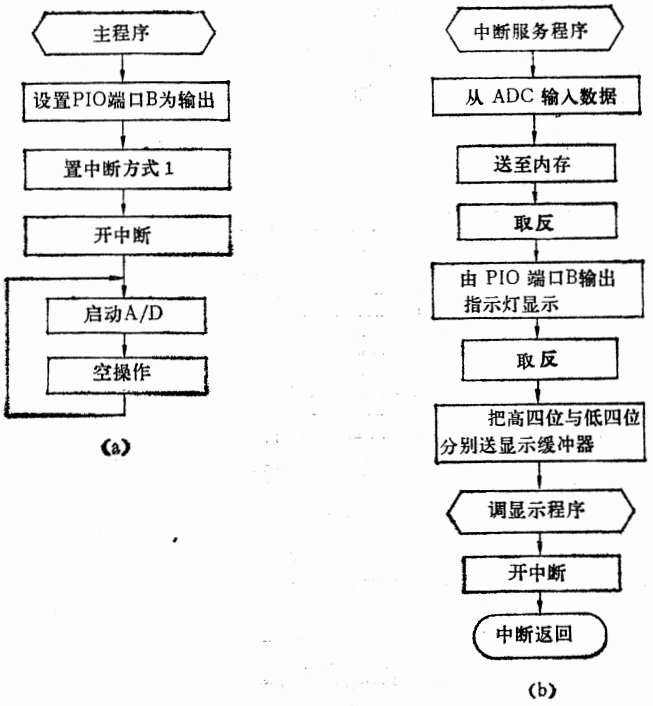


图3.18　A/D转换的程序流程图

4．在单板机上装入程序，运行程序。

先把输入的模拟量调至0V，观察数码显示是否为00。再把模拟量调至5V，观察数码显示是否为FF。经过调试，运行正确以后。连续地输入一些电压，记录相应的显示值。验证输入的结果和相应的精度。

三、实验器材

1．TP801-A 单板机　　1台。
2．TP80TS 实验板　　1块。
3．ADC0809　　1片。
4．74LS04(六反相器)　　2片。
5．万用表　　1块。
6．100k电位器　　1个。
7．1kΩ 电阻　　4个。
8．74LS00　　1片。
9．74LS02　　1片。

四、预习要求

1．仔细阅读本实验指示书。
2．复习A/D转换原理、接口方法和程序设计方法。

3．根据流程图，在实验前编好源程序并翻译成机器码。

4．准备好记录表格。

五、报告要求

1．整理好经过运行而证明是正确的源程序并加上注释。

2．整理好记录表格。

3．总结A/D转换的接口方法和程序设计方法。

4．考虑以下思考题。

（1）若采用TP801-A的读/写信号和其它信号，用程序启动ADC片转换，应怎样接线？试画出接线图。程序应如何改变？

（2）对于发光二极管LED9—LED2为什么有些数值使其有闪烁现象？原因在哪里？8位输出数字量对应的LED指示的数码为什么低位闪烁得比高位严重。如何改进？

*实验八　A/D和D/A正逆变换实验

一、实验目的

1．掌握D/A转换片子DAC0832的结构，接口方法和使用方法。

2．了解A/D和D/A正逆变换过程。

3．进一步掌握A/D及D/A片子与微机系统的接口方法，微型机系统利用A/D及D/A与模拟量相接口的程序设计方法。

二、实验原理

在实际的系统中，CPU通过A/D转换片子输入要采集或要监视的模拟量，经过一定的处理，通过D/A输出去控制模拟量。

在实验中为了模拟A/D与D/A的工作过程，我们安排了这个实验，通过A/D输入模拟量，又通过D/A输出。若工作正常的话，则输出的模拟量应该与输入的一致。

A/D片子仍用ADC0809，它的工作过程已在上一个实验中了解了。

D/A片子采用DAC0832，其结构如图3.19所示。

其主体部分是一个8位的D/A转换器，它实质上是一个电阻网络，由模拟开关（开关的状态取决于输入的数字量）控制标准电源在其上所产生的电流。数字量通过两级锁存（缓冲）后送至D/A转换器的输入端。两级锁存，可以做到在后一级正输出给D/A进行转换时，前一级就可以接收新的数据，从而提高了转换速度。这两级锁存就要分别加以控制。

当CPU把数据送至片子的数据输入端时，就要设法使第一级锁存选通。这是由输入锁存允许（ILE）和选片信号$\overline{CS}$及写命令信号（$\overline{WR1}$）的“与”来实现的。在使用时，这些信号当然要由CPU提供。对于Z80CPU来说，是由$\overline{IORQ}$、$\overline{WR}$以及相应的地址信号来控制的。

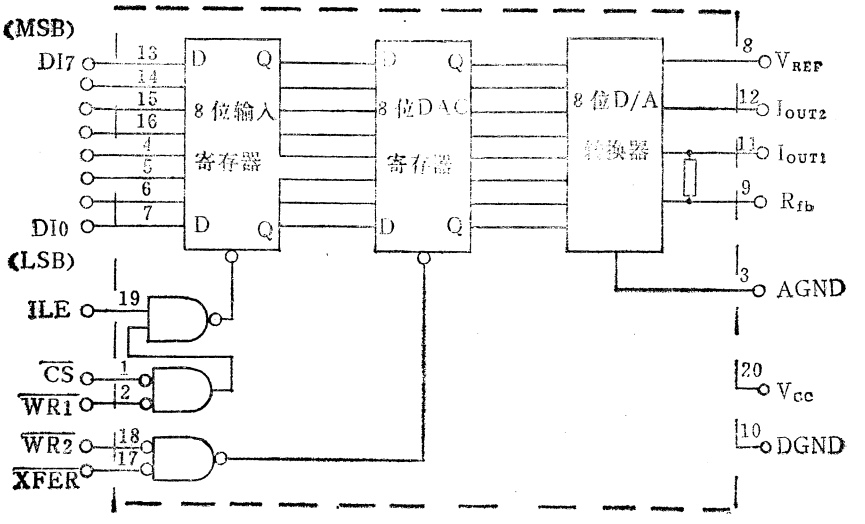


图3.19　DAC0832 结构图

在第一级锁存器已把 CPU 的输出数据锁存以后，在适当的时刻，把数据再传送给第二级锁存器，于是片子的输出就改变了。第二级锁存器的选通是由传送信号($\overline{XFER}$)和写命令信号 ($\overline{WR2}$)的“与”实现的。这些信号，当然也要由 CPU 来加以控制。实际上，CPU 是把这两个锁存器看作两个端口，由两条输出指令来分别加以控制。

DAC0832 输出的是电流，要转换为电压的话，还要经过一个运算放大器。

DAC0832 典型应用的接线图如图 3.20 所示。

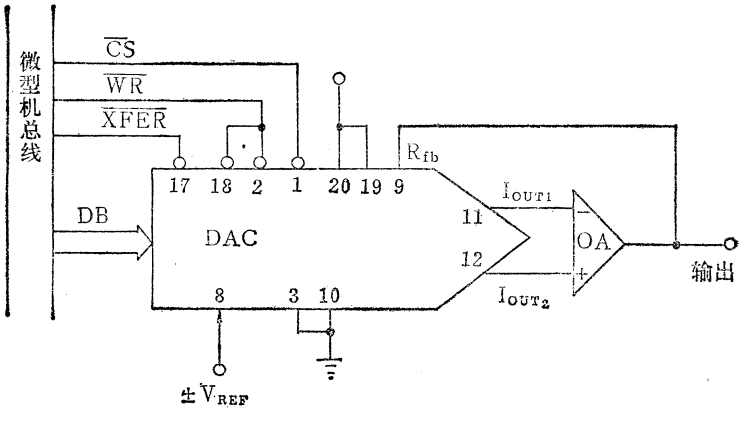


图3.20　DAC0832 的典型接线图

在本实验中，为了简化，第二级锁存器的选通信号是固定连接，使其始终是选通的。本实验的接线图如图 3.21 所示。

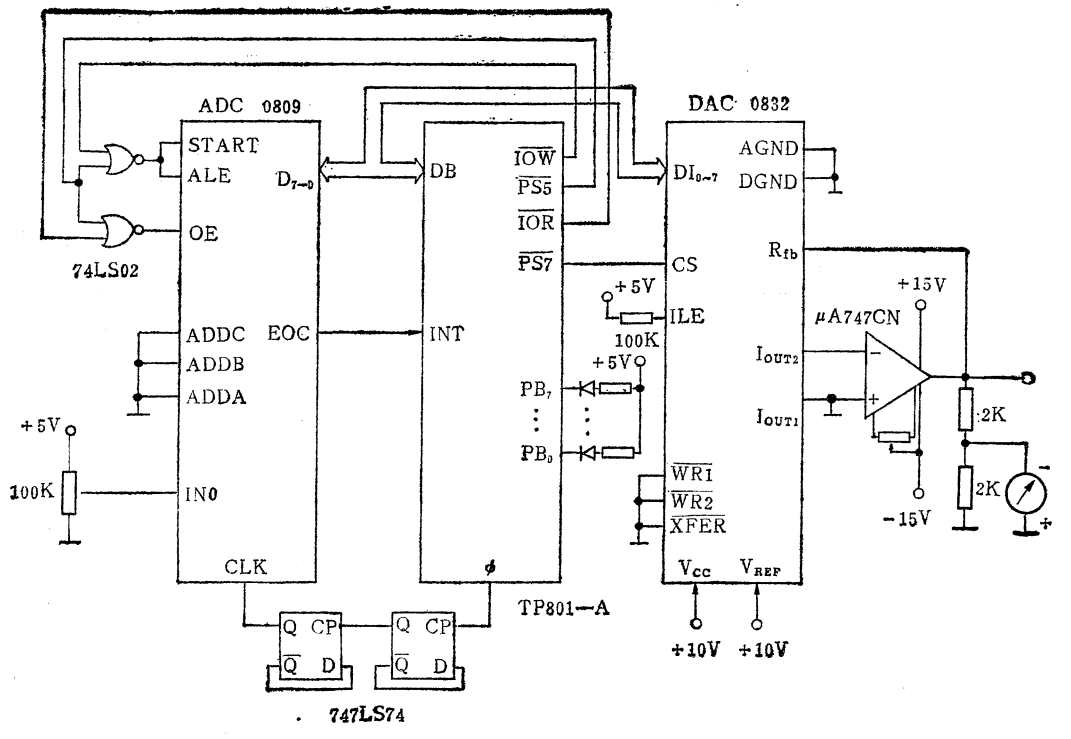


图3.21 A/D—D/A正逆 变换实验线路图

ADC0809 片子的连接与上一个实验中一样，由 CPU 令其启动，A/D 转换完成后，由 EOC 向 CPU 发中断请求，在中断服务程序中用输入指令打开 ADC0809 的输出三态门，把数据直接送至单板机的数据总线上。CPU 把数据输入后，一方面通过 PIO 的 端口 B 输出显示（LED 显示）；另一方面通过 D/A 输出。

DAC0832 的 控制信号端除了 $\overline{CS}$外，都固定连接，$\overline{CS}$连 至 TP801-A的 I/O 设备的译码输出线$\overline{PS7}$。DAC0832 的输出经过运算放大器 μA747CN，转换成电压 。 DAC 0832 的满量程输出为 10V，我们输入的最大量程为 5V，所 以分压后用电压表指示。

三、实验内容

1. 断开电路，按图 3.21 接线。

2. 编主程序。

主程序与上一个实验中的相似，主要是为了启动 A/D 转换，然后程序进入循环，等待 A/D 转换的完成。其流程图如图 3.22(a)所示。

3. 编中断服务程序。

（1）在中断服务程序中，输入 A/D 转换后的数据。

（2）把此数据通过 PIO 端口 B 输出，由 LED 显示。

（3）A/D 输入的数据，通过 D/A 输出。

（4）为了使单板机上的数码显示管能像数字电压表一样显示输出的模拟电压，就要把要输出的数据经过一定的转换。

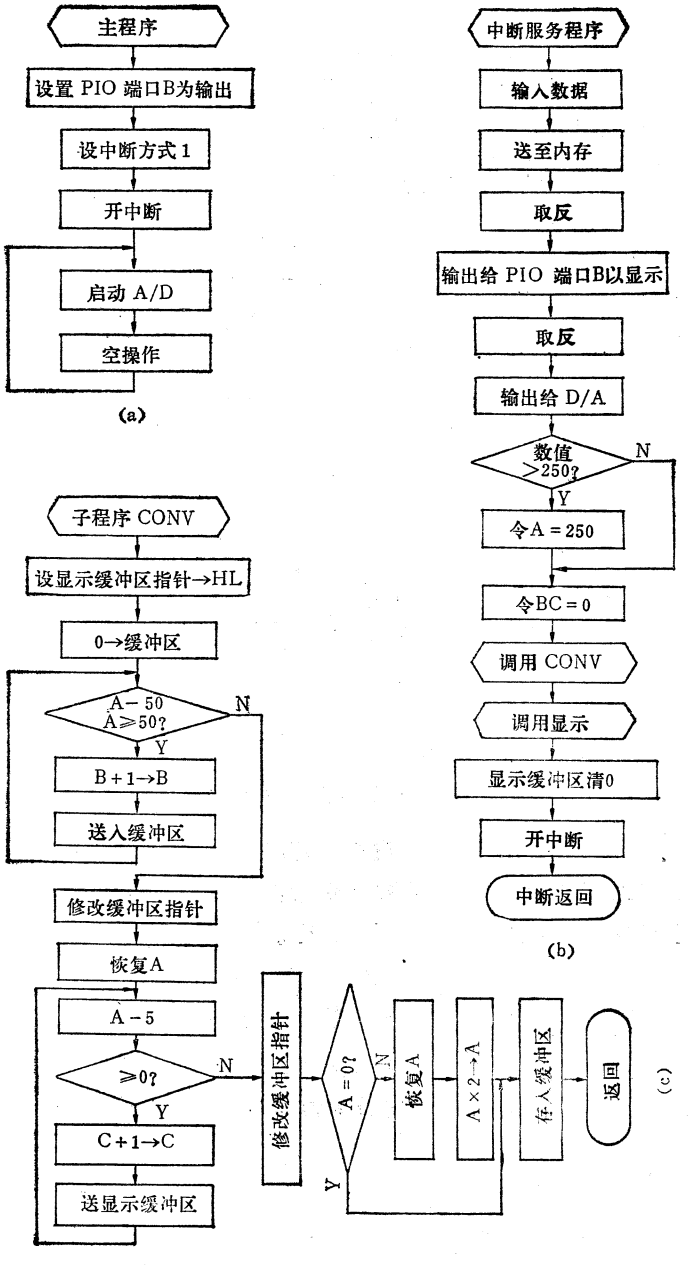


图3.22　A/D与D/A正逆变换程序流程图

显示的最大量程为 5V，用三位数字在单板机的右面三个数码管上显示。最左面一位代表个位，其值为 0—5；过来一位是 1/10 位，其值为 0—9；最右面一位为 1/100 位，其值也为 0—9。

我们要输出的数据是 8 位二进制数，其值的范围为 0—255。我们以 250 为满量程相应于 5.00 V。则要输出的数中包含有多少个 50，就是它的相应的伏数；余下的数中包含有多少个 5，就是它的相应的 1/10 伏数；最后剩下的数乘以 2 后则为 1/100 伏的数。

（5）调用显示程序。

（6）开中断，返回。

程序的流程图如图 3.22(b)、(c)所示。

4．在单板机上装入程序，运行程序。

当把输入的模拟量调至 0 时，观察 LED 的显示，电表测得的 D/A 转换后的电压是否为 0，数码管上的显示是否为 0。

当把输入的模拟量调至最大(5V)时，观察 LED 的显示，电表测得的 D/A 转换后的电压是否也为 5V，数码管上的显示是否为 5.00V。

在这之间做 10 个数据，分别测量和记录上述各处的数值。

四、实验器材

1．TP801-A 单板机　　1 台。
2．TP80TS 实验板　　1 块。
3．两路输出直流稳压电源（+5 V，+10V）　1 台。
4．ADC0809　　1 片。
5．DAC0832　　1 片。
6．万用表　　1 块。
7．74LS04（六反相器）　　2 片。
8．74LS74（双 D 触发器）　　1 片。
9．74LS02（四或非门）　　1 片。
10．μA747CN（运算放大器）　　1 片。
11．电位计（100k）　　1 个。
12．电阻2kΩ　　2 个。

五、预习要求

1．仔细阅读本实验指导书。
2．复习 D/A 转换器的原理、功能，接口方法和程序设计方法。
3．根据流程图在实验前编好主程序及中断服务程序，并翻译成机器码。
4．准备好数据表格，及拟定实验步骤。

六、报告要求

1．整理好经过运行且证明是正确的各个源程序，并加上注释。
2．整理好记录表格。分析实验结果，分析 A/D 及 D/A 片子的精度和线性度。
3．总结 A/D 和 D/A 与 CPU 的接口方法。程序设计的方法。
4．考虑若 DAC0832 连成双缓冲数据输入方式，要使用单板机上哪些信号？如何连线？程序应如何编？

*实验九　利用串行接口，实现单板机间的通讯

一、实验目的

1．了解串行通讯的特点、原理和方法。

2．了解串行通讯的一些技术问题。

3．掌握8251A串行接口片子的结构。工作原理、初始化编程方法。

4．掌握8251A与CPU之间的接口方法和CPU实现串行通讯的程序设计方法。

二、实验原理

串行通讯由于数据传送线少，适用于远距离通讯。CPU的工作以及输入输出是并行的，而通讯是串行的，所以在串行通讯中一个基本问题是实行串↔并转换。有了串行接口片子，串↔并的转换由接口片子中的硬件电路实现。于是，CPU与串行接口片子的接口与编程方法就同CPU与并行接口片子的接口和编程方法相似了。

关于8251A的结构、工作原理等已在教材中叙述，在这儿我们不再重复。

8251A与单板机之间的连线如图3.23所示。

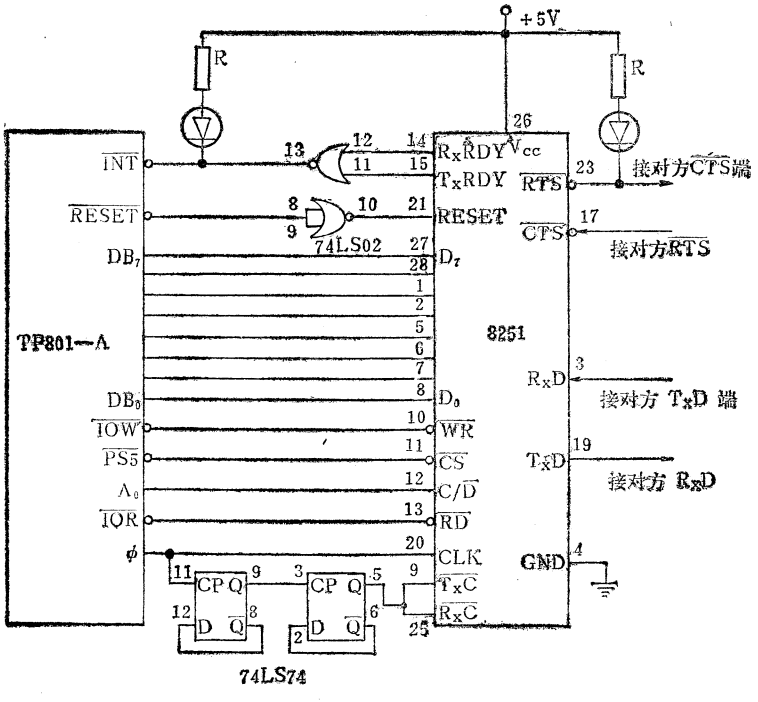


图3.23　8251A通讯接线图

8251A的数据线直接与单板机的数据线相连。按8251A的参数的规定，它的时钟频率在异步通讯时至少为通讯波特率的30倍，单板机的时钟为2MHz，所以，用时钟信号作为8251A的CLK信号是合适的。可以用单板机的时钟经过四分频后作为8251A的$\overline{RxC}$和$\overline{TxC}$信号。串行通讯的波特率与时钟频率之间的系数采用64，这样就满足了上

述要求，且实验中的异步通讯的波特率也在8251A的允许范围之内。我们用单板机的I/O设备的译码信号$\overline{PS5}$，作为8251A的选片信号，这样，当CPU对端口地址94～97H进行读写操作时，就选中8251A。把地址总线的AB0直接连到8251A的C/$\overline{D}$端，用来选择要进行读写操作的是8251A的控制/状态寄存器还是数据寄存器。

8251A对于CPU来说，是属于一个外部设备，所以要用$\overline{IORQ} \cdot \overline{RD} = \overline{IOR}$连到8251A的$\overline{RD}$信号端；而用$\overline{IORQ} \cdot \overline{WR} = \overline{IOW}$连到8251A的$\overline{WR}$信号端。

8251A的TxRDY和RxRDY两引线经过一个或非门以后，引到单板机的$\overline{INT}$引线上，作为向CPU的中断请求信号。

本方的TxD线和RxD线，引到与之通讯的对方的数据线。TxD接对方的RxD而RxD接对方的TxD。

为了做到双方的可靠通讯，需要用到一些联络信号线。8251A规定，只有在命令字中规定发送允许(TxEN有效)，本身的发送缓冲器空即TxRDY有效，而且$\overline{CTS}$信号有效(为低电平)才能开始发送。所以，为了可靠通讯，本方的$\overline{CTS}$接到对方的$\overline{RTS}$，而本方的$\overline{RTS}$接至对方的$\overline{CTS}$。

若本机工作于接收方式，则只有在命令字中令输出的$\overline{RTS}$有效，则就使对方的$\overline{CTS}$变为有效，对方才有可能发送。反之也是如此。用一个发光二极管来指示，本机是否正在接收数据。也可以通过一个发光二极管，检测8251A发生中断请求和TP801-A处理中断的情况。

三、实验内容

1．断开电源，按图3.23接线。

2．编主程序。

为了简化实验，我们规定通讯双方一方为接收，另一方为发送。实验者可根据自己的需要选择其中之一。

（1）凡要使用8251A，必须在其复位（硬件复位或软件复位）以后，先用方式控制字规定其工作方式。

8251A的方式控制字为：

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
同步情况	异步情况	奇偶校验		字符长度		波特率系数	
00=内	00=无效	×0=无奇偶校验		00=5位		00=同步方式	
01=外	01=1个停止位	01=奇校验		01=6位		01=异步×1	
10=内单	10=$1^1/_2$个			10=7位		10=异步×16	
11=内双	11=2个	11=偶校验		11=8位		11=异步×64	

本实验中令其工作在异步方式，波特率系数为64，字符长度为8位，偶校验，2个停止位。于是，方式控制字为0FFH。

然后，再用命令字使其工作于运行状态。8251A的命令字的格式为：

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
EH	IR	RTS	ER	SBRK	RxE	DTR	TxEN
进入搜索方式 1 = 启动搜索 同步字符	内部复位 1 = 迫使8251 返回到方式 指令格式	请求发送 1 = RTS引 线输出低 电平	错误标志 复位 1 = 使错误标 志全部复位	送中止字符 1 = 迫使T_XD 为低电平 0 = 正常	接收允许 1 = 允许 0 = 禁止	数据终端 准备好 1 = 使DTR 引线输出低	发送允许 1 = 允许 0 = 禁止

在本实验中，若是作为接收器，则命令字应为00110100B＝34H；若是作为发送器，则命令字应为00010001B＝11H。

（2）为了在程序中能区分是规定为发送器还是为接收器，在程序中建立一个标志（发送，A＝01H；接收，A＝0）。

（3）整个通讯分两段完成，第一段由发送方发送两个信息：

一是接收区的首地址（即下一段要传送的信息，传送至接收方的内存缓冲区的首地址）；另一是即将要传送的信息的长度。

第二段发送规定长度的信息。

（4）每一次发送或接收的程序中，在对8251A送了方式字，规定了工作方式；和送了命令字使其运行以后，就动态停机等待中断。发送方在中断服务程序中发送一个字节；接收方也在中断服务程序中接收一个字节。直至整个数据块传送完。

（5）信息传送完以后，若传送正常，则在单板机的显示管（最右两位）显示“P”。

若传送有错误，则显示“EP”；若有其它中断源，则显示“E′”。

主程序的流程图如图3.24所示。

3．编中断服务程序

（1）输入8251A的状态。

8251A不论是R_XRDY有效（接收器数据有效），还是T_XRDY有效（发送缓冲器空），都向CPU发中断请求。所以为了区分是谁的中断请求，或为了检验是否是指定的中断请求，都必须在中断服务程序中，先输入8251A的状态。

8251A状态寄存器的内容为：

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
DSR	SYNDET	FE	OE	PE	T_XE	R_XRDY	T_XRDY

其中DSR、SYNDET、T_XE、RxRDY，这几位反映了同名的几条引线的状态。D_0位即T_XRDY位与TxRDY引线的状态略有不同，TxRDY引线只有在发送允许，且CTS为低电平，和发送缓冲器空才有效；而TxRDY这一位，只要发送缓冲器空就为“1”。

其中，PE、OE、FE为三种出错标志位。PE（Parity error）为奇偶错标志，若接收到的字符，经过奇偶校验，不符合编程时的规定，则此标志位置位。OE(Overrun error）溢出（或丢失）错标志，若上一个接收到的数据，经过串→并，且送到了接收器数据寄存器；当接收器接收到了下一个数据，也经过了串→并，要送至接收器数据寄存器时，上一个数据尚未被CPU读走，则发生溢出错误，此标志置位，上一个数据丢失。FE(Frame error）帧错标志，当接收到的数据格式（包括启动位、停止位等）不符合

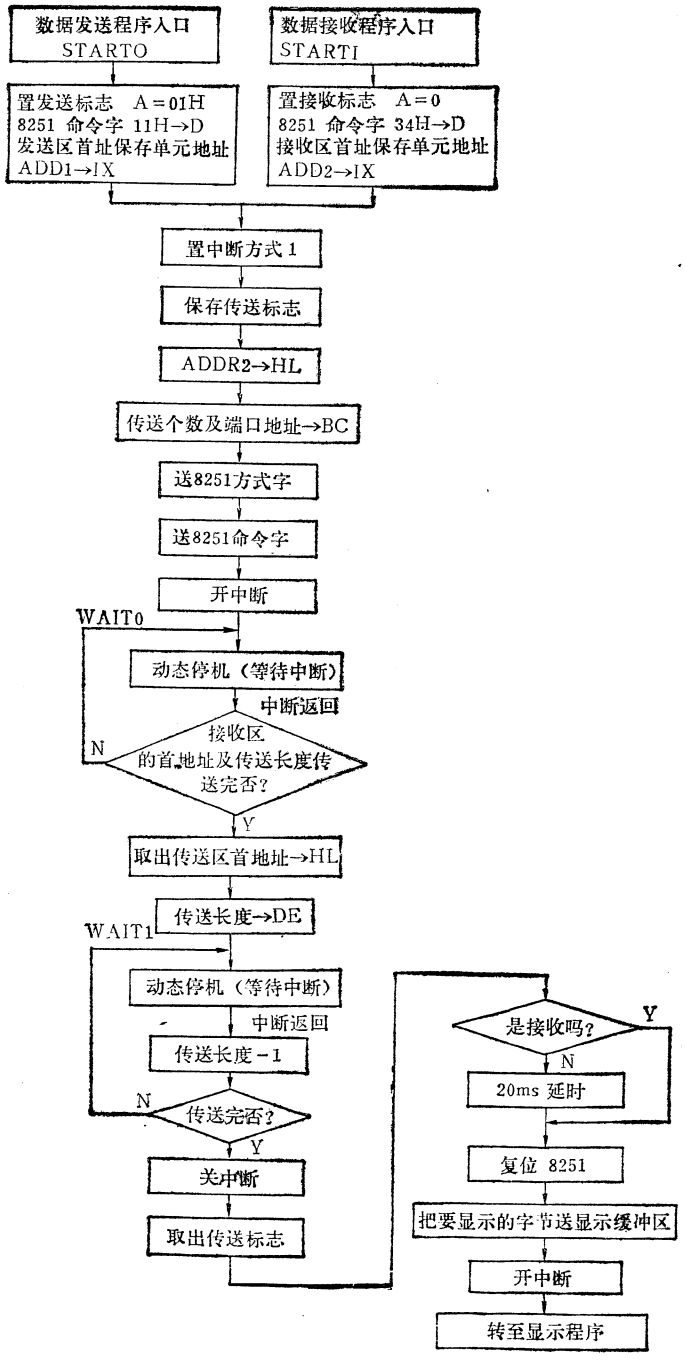


图3.24 串行通讯实验主程序流程图

初始化编程时方式字的规定，此标志置位。

通常在中断服务程序中，首先要读入上述的8251A状态寄存器的内容，通过检查TxRDY和RxRDY位以确定是发送中断还是接收中断。

（2）在本实验中，若是工作为发送器，则检查到是发送中断后就输出数据，然后开中断返回主程序。

（3）若是工作为接收器，在检查到是接收中断后，再检查几个出错标志，判断在接收过程中无错后输入至CPU。若有错则准备显示字符“Ep”。

（4）若输入的状态字中既不是TxRDY有效，也不是RxRDY有效，则系统中还有别的中断源，准备显示字符“E′”。

中断服务程序的流程图如图3.25所示。

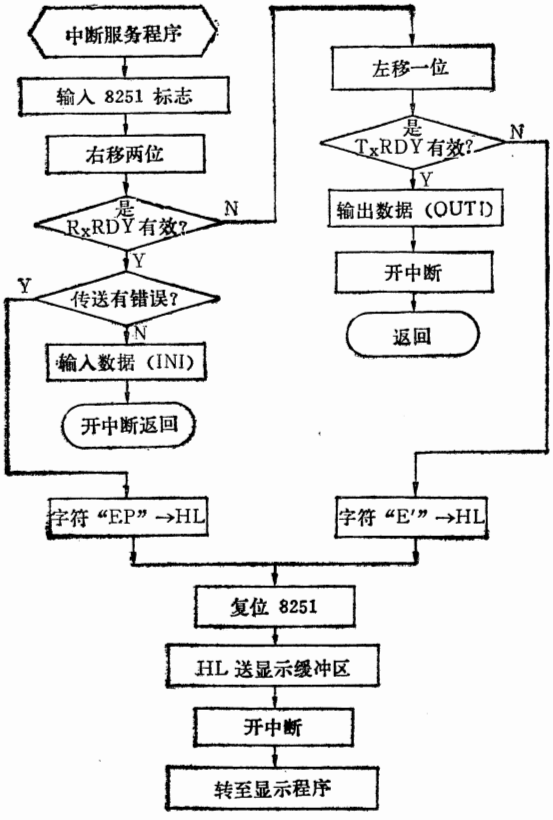


图3.25 中断服务程序的流程图

4．在单板机上输入主程序和中断服务程序。

5．因为数据的发送区和对方接收区都由发送方指定。因此，在发送前，发送方必须在ADDDR1（204AH、204BH）处填入发送区首地址（低位字节在前，下同），在ADDR2(204CH、204DH）处填入对方接收区首地址，及在LENGTH（204EH、204FH）处填入传送长度。

当Z80-CPU工作在中断方式1时，CPU接收了中断请求以后，就固定地转到0038H单元去处理。在TP801-A中，地址0038H单元在ROM区，而在ROM中固化了一条无

条件转移到地址2FD8H的指令。因此，当我们输入了主程序和中断服务程序之后，在启动主程序之前，必须在地址2FD8H处填入一条无条件转移到中断服务程 序入口的指令。

6．运行程序。在本实验中，必须发送方和接收方成对地工作。

试着发送几组不同的信息，检查接收是否正确。

7．有时间的话，也可以发送方改为接收方，而接收方改为发送方。检 查 发送与接收是否正确。

四、实验器材

1．TP801-A单板机	1台
2．TP80TS 实验板	1块
3．8251A	1片
4．74LS74(双D触发器)	1片
5．74LS02(四或非门)	1片

五、预习要求

1 仔细阅读本实验指导书。

2．复习串行通讯的原理。

3．复习8251A的原理、结构，编程方法和使用方法。

4．参照流程图，在实验前编好主程序及中断服务程序。

5．做好主程序、中断服务程序，发送区数据与接收区数据的内 存分 配。整备好几组要发送的数据。

六、报告要求

1．整理好经过运行且证明是正确的各个源程序，并加上注释。

2．整理好实验中的各种数据。

3．总结8251A的接口方法，编程方法和使用方法。

4．考虑下列思考题：

（1）在数据发送过程中，观察发光二极管 L_2 有什么现象？为什么？

（2）如果通讯双方都启动了数据接收程序，或都启动了数据发 送 程序。会有什么结果？为什么？

（3）按前面程序中给出的8251A的方式字，本实验中数据传送的波特率是 多少？

（4）为了使数据传送能在中断方式 2 下进行，在硬件连接上需作哪些 改 动？软件上呢？

附　录

附录1　第一部分　实验中的参考程序

实验二的参考程序

1．在内存中从2022H开始，建立0-99共一百个数。

地址	机器码	源	程	序
2000	21 22 20		LD	HL, 2022H
2003	AF		XOR	A
2004	06 64		LD	B, 64H
2006	77	LOOP:	LD	(HL), A
2007	3C		INC	A
2008	23		INC	HL
2009	10 Fb		DJNZ	LOOP
200B	76		HALT	

2．数据块传送程序

地址	机 器 码	源	程	序
200C	11 22 20		LD	DE, 2022H
200F	21 90 20		LD	HL, 2090H
2012	06 64		LD	B, ~~100~~ 64H
2014	1A	LOOP:	LD	A, (DE)
2015	77		LD	(HL), A
2016	13		INC	DE
2017	23		INC	HL
2018	10 FA		DJNZ	LOOP
201A	76		HALT	

3．用单条数据块传送指令

201B	21 22 20		LD	HL, 2022H
201E	11 90 20		LD	DE, 2090H
2021	01 64 00		LD	BC, 0064H
2024	ED A0	LOOP:	LDI	
2026	FA 24 20		JP	PE, LOOP
2029	76		HALT	

4．用数据块成组传送指令

202A	21 22 20		LD	HL, 2022H

```
202D    11 90 20            LD      DE,2090H
2030    01 64 00            LD      BC,0064H
2033    ED B0               LDIR
2035    76                  HALT
```

实验三的参考程序

1．在内存中自2100H单元开始，建立10组10个连续的字符的ASCII 码程序。

```
地址        机器码          源  程  序
2000    21 00 21            LD      HL,2100H
2003    06 0A               LD      B,0AH
2005    0E 0A       LOOP1:  LD      C,0AH
2007    3E 20               LD      A,20H
2009    77          LOOP2:  LD      (HL),A
200A    3C                  INC     A
200B    23                  INC     HL
200C    0D                  DEC     C
200D    20 FA               JR      NZ,LOOP2
200F    10 F4               DJNZ    LOOP1
2011    76                  HALT
```

2．用单条数据块搜索指令

```
2020    21 00 21            LD      HL,2100H
2023    01 64 00            LD      BC,0064H
2026    3E 24               LD      A,24H
2028    DD 21 70 21         LD      IX,2170H
202C    ED A1       LOOP:   CPI
202E    20 __               JR      NZ,GOON
2030    2B                  DEC     HL
2031    DD 75 00            LD      (IX+0),L
2034    DD 74 01            LD      (IX+1),H
2037    DD 23               INC     IX
2039    DD 23               INC     IX
203B    23                  INC     HL
203C    EA 2C 20    GOON:   JP      PE,LOOP
203F    76                  HALT
```

3．用数据块成组搜索指令

```
2040    21 00 21            LD      HL,2100H
2043    01 64 00            LD      BC,0064H
```

```
2046    3E 24              LD      A,24H
2048    DD 21 70 21        LD      IX,2170H
204C    ED B1       LOOP:  CPIR
204E    28[  ]             JR      Z,FOUND
2050    18[  ]             JR      DONE
2052    2B          FOUND: DEC     HL
2053    DD 75 00           LD      (IX+0),L
2056    DD 74 01           LD      (IX+1),H
2059    DD 23              INC     IX
205B    DD 23              INC     IX
205D    23                 INC     HL
205E    EA 4C 20           JP      PE,LOOP
2061    76          DONE:  HALT
```

4. 在内存建立0-99共100个数，要求奇数为负，偶数为正且找出最大值的程序

```
2000    21 00 21           LD      HL,2100H
2003    06 64              LD      B,100
2005    AF                 XOR     A
2006    CB 47       LOOP:  BIT     0,A
2008    28[  ]             JR      Z,MUN
200A    ED 44              NEG
200C    77          MUN:   LD      (HL),A
200D    3C                 INC     A
200E    23                 INC     HL
200F    10[  ]             DJNZ    LOOP
2011    21 00 21           LD      HL,2100H
2014    06 63              LD      B,99
2016    7E                 LD      A,(HL)
2017    23                 INC     HL
2018    BE          LOOP1: CP      (HL)
2019    FA 21 20           JP      M,SIGN
201C    EA 24 20           JP      PE,EXCH
201F    18[  ]             JR      UNEX
2021    EA 25 20    SIGN:  JP      PE,UNEX
2024    7E          EXCH:  LD      A,(HL)
2025    23          UNEX:  INC     HL
2026    10[  ]             DJNZ    LOOP1
2028    76                 HALT
```

实验四的参考程序

1．在内存缓冲区建立10组‘0’—‘9’的ASCII码的程序

```
2000    21 00 21          LD      HL,2100H
2003    36 64             LD      (HL),64H
2005    23                INC     HL
2006    36 00             LD      (HL),0
2008    23                INC     HL
2009    06 0A             LD      B,0AH
200B    0E 0A     LOOP1:  LD      C,0AH
200D    3E 30             LD      A,30H
200F    77        LOOP2:  LD      (HL),A
2010    3C                INC     A
2011    23                INC     HL
2012    0D                DEC     C
2013    20[  ]            JR      NZ,LOOP2
2015    10[  ]            DJNZ    LOOP1
2017    76                HALT
```

2．把ASCII码转换为BCD码的程序

```
2020    11 02 21          LD      DE,2102H
2023    21 02 21          LD      HL,2102H
2026    ED 4B 00 21       LD      BC,(2100H)
202A    DD 21 00 00       LD      IX,0000H
202E    1A          LOOP: LD      A,(DE)
202F    ED 67             RRD
2031    0B                DEC     BC
2032    13                INC     DE
2033    1A                LD      A,(DE)
2034    ED 67             RRD
2036    23                INC     HL
2037    13                INC     DE
2038    DD 23             INC     IX
203A    0B                DEC     BC
203B    78                LD      A,B
203C    B1                OR      C
203D    20[  ]            JR      NZ,LOOP
203F    DD 22 00 21       LD      (2100H),IX
```

```
2043    76              HALT
```

注：上述程序是在已知ASCII码的个数为偶数的条件下编制的。若已知个数为奇数，或不知是奇数或偶数个，则在把数据块长度取入BC后，可用 BIT 0,C 来检查，若是奇数，则先取一个ASCII码，屏蔽掉高 4 位，存入BCD码区域，再修改指针，字节数减1，然后就可以按上述程序进行转换。

3．把BCD码转换为ASCII码的程序

```
2050    21 02 21            LD      HL,2102H
2053    11 02 22            LD      DE,2202H
2056    ED 4B 00 21         LD      BC,(2100H)
205A    3E 30       LOOP:   LD      A,30H
205C    ED 67               RRD
205E    12                  LD      (DE),A
205F    13                  INC     DE
2060    ED 67               RRD
2062    12                  LD      (DE),A
2063    13                  INC     DE
2064    23                  INC     HL
2065    0B                  DEC     BC
2066    78                  LD      A,B
2067    B1                  OR      C
2068    20[  ]              JR      NZ,LOOP
206A    ED 4B 00 21         LD      BC,(2100H)
206E    CB 21               SLA     C
2070    CB 10               RL      B
2072    ED 43 00 22         LD      (2200H),BC
2076    76                  HALT
```

4．在内存建立10组16进制数的ASCII码的程序

```
2080    21 00 23            LD      HL,2300H
2083    36 A0               LD      (HL),0A0H
2085    23                  INC     HL
2086    36 00               LD      (HL),00H
2088    23                  INC     HL
2089    06 0A               LD      B,0AH
208B    16 0A       LOOP11: LD      D,0AH
208D    3E 30               LD      A,30H
208F    77          LOOP12: LD      (HL),A
2090    3C                  INC     A
2091    23                  INC     HL
2092    15                  DEC     D
```

```
2093    20[  ]              JR      NZ,LOOP12
2095    1E 06               LD      E,06H
2097    3E 41               LD      A,41H
2099    77          LOOP13: LD      (HL),A
209A    3C                  INC     A
209B    23                  INC     HL
209C    1D                  DEC     E
209D    20[  ]              JR      HZ,LOOP13
209F    10[  ]              DJNZ    LOOP11
20A0    76                  HALT
```

5. 把以ASCII码表示的16进制字符转换为16进制数的程序

```
20B0    21 02 23            LD      HL,2302H
20B3    ED 4B 00 23         LD      BC,(2300H)
20B7    AF          LOOP22: XOR     A
20B8    ED 6F               RLD
20BA    D6 03               SUB     03H
20BC    20[  ]              JR      NZ,OUT1
20BE    ED 67               RRD
20C0    18[  ]              JR      OUT2
20C2    7E          OUT1:   LD      A,(HL)
20C3    C6 09               ADD     A,9
20C5    E6 0F               AND     0FH
20C7    77                  LD      (HL),A
20C8    23          OUT2:   INC     HL
20C9    0B                  DEC     BC
20CA    78                  LD      A,B
20CB    B1                  OR      C
20CC    20[  ]              JR      NZ,LOOP22
20CE    76                  HALT
```

实验五的参考程序

1. 多字节十进制数相加程序

```
2000    DD 21 00 21         LD      IX,2100H
2004    06 0A               LD      B,0AH
2006    AF                  XOR     A
2007    DD 7E 00    LOOP:   LD      A,(IX+0)
200A    DD 8E 0A            ADC     A,(IX+0A)
```

```
200D   27                    DAA
200E   DD 77 14              LD      (IX+14), A
2011   DD 23                 INC     IX
2013   10[  ]                DJNZ    LOOP
2015   76                    HALT
```

2. 两个两位十进制数乘法程序

```
2020   21 30 21              LD      HL, 2130H
2023   4E                    LD      C, (HL)
2024   23                    INC     HL
2025   46                    LD      B, (HL)
2026   21 00 00              LD      HL, 0000H
2029   78            LOOP:   LD      A, B
202A   B7                    OR      A
202B   28[  ]                JR      Z, DONE
202D   7D                    LD      A, L
202E   81                    ADD     A, C
202F   27                    DAA
2030   6F                    LD      L, A
2031   7C                    LD      A, H
2032   CE 00                 ADC     A, 0
2034   27                    DAA
2035   67                    LD      H, A
2036   78                    LD      A, B
2037   3D                    DEC     A
2038   27                    DAA
2039   47                    LD      B, A
203A   18[  ]                JR      LOOP
203B   76            DONE:   HALT
```

3. 两个八位带符号数乘法程序

```
2040   21 40 21              LD      HL, 2140H
2043   5E                    LD      E, (HL)
2044   23                    INC     HL
2045   7B                    LD      A, E
2046   E6 80                 AND     80H
2048   32 50 21              LD      (2150H), A
204B   F2 50 20              JP      P, NEXT1
204C   7B                    LD      A, E
204D   ED 44                 NEG
204F   5F                    LD      E, A
```

```
2050    7E          NEXT1:  LD      A,(HL)
2051    E6 80               AND     80H
2053    32 51 21            LD      (2151H),A
2056    7E                  LD      A,(HL)
2057    F2 5A 20            JP      P,NEXT2
2058    ED 44               NEG
205A    21 00 00    NEXT2:  LD      HL,0000H
205D    54                  LD      D,H
206E    CB 3F       LOOP11: SRL     A
2060    D2 62 20            JP      NC,SHIFT
2061    19                  ADD     HL,DE
2062    CB 23       SHIFT:  SLA     E
2064    CB 12               RL      D
2066    B7                  OR      A
2067    20                  JR      NZ,LOOP11
2069    3A 50 21            LD      A,(2150H)
206C    47                  LD      B,A
206D    3A 51 21            LD      A,(2151H)
2070    A8                  XOR     B
2071    28                  JR      Z,NEXT3
2073    7D                  LD      A,L
2074    2F                  CPL
2075    6F                  LD      L,A
2076    7C                  LD      A,H
2077    2F                  CPL
2078    67                  LD      H,A
2079    23                  INC     HL
207A    22 52 21    NEXT3:  LD      (2152H),HL
207D    76                  HALT
```

4. 两个 8 位带符号数的除法程序

```
2080    DD 21 00 22         LD      IX,2200H
2084    AF                  XOR     A
2085    57                  LD      D,A
2086    67                  LD      H,A
2087    4F                  LD      C,A
2088    06 08               LD      B,08H
208A    DD 5E 00            LD      E,(IX+0)
208D    7B                  LD      A,E
208E    E6 80               AND     80H
```

```
2090    DD 77 02            LD      (IX+2),A
2093    7B                  LD      A,E
2094    F2 9A 20            JP      P,OUT1
2097    ED 44               NEG
2099    5F                  LD      E,A
209A    DD 6E 01    OUT1    LD      L,(IX+1)
209D    AF                  XOR     A
209E    85                  ADD     A,L
209F    28[  ]              JR      Z,OUT2
20A1    E6 80               AND     80H
20A3    DD 77 03            LD      (IX+3),A
20A6    7D                  LD      A,L
20A7    F2 AC 20            JP      P,LOOP22
20AA    ED 44               NEG
20AC    6F          LOOP22: LD      L,A
20AD    CB 23       LOOP21: SLA     E
20AE    CB 12               RL      D
20AF    7A                  LD      A,D
20B0    95                  SUB     L
20B1    38[  ]              JR      C,LESS
20B3    57                  LD      D,A
20B4    1C                  INC     E
20B5    10[  ]      LESS:   DJNZ    LOOP21
20B7    DD 7E 02            LD      A,(IX+2)
20BA    47                  LD      B,A
20BB    DD 7E 03            LD      A,(IX+3)
20BE    A8                  XOR     B
20BF    28[  ]              JR      Z,OUT2
20C1    7B                  LD      A,E
20C2    ED 44               NEG
20C4    5F                  LD      E,A
20C5    76          OUT2:   HALT
```

实验六的参考程序

1. 显示八位带符号数和的程序

```
2000    21 00 21            LD      HL,2100H
2003    DD 21 0A 21         LD      IX,210AH
2007    06 0A               LD      B,10
2009    AF                  XOR     A
```

```
200A  DD 77 00          LD    (IX+0), A
200D  DD 77 01          LD    (IX+1), A
2010  DD 77 02          LD    (IX+2), A
2013  DD 77 03          LD    (IX+3), A
2016  7E          LOOP: LD    A, (HL)
2017  E6 80             AND   80H
2019  7E                LD    A, (HL)
201A  28 __             JR    Z, NEXT1
201C  ED 44             NEG
201E  DD 86 02          ADD   A, (IX+2)
2021  DD 77 02          LD    (IX+2), A
2024  30 __             JR    NC, NEXT2
2026  DD 34 03          INC   (IX+3)
2029  18 __             JR    NEXT2
202B  DD 86 00    NEXT1: ADD  A, (IX+0)
202E  DD 77 00          LD    (IX+0), A
2031  30 __             JR    NC, NEXT2
2033  DD 34 01          INC   (IX+1)
2036  23          NEXT2: INC  HL
2037  10 __             DJNZ  LOOP
2039  2A 0A 21          LD    HL, (210A)
203C  ED 4B 0C 21       LD    BC, (210C)
2040  78                LD    A, B
2041  2F                CPL
2042  47                LD    B, A
2043  79                LD    A, C
2044  2F                CPL
2045  4F                LD    C, A
2046  03                INC   BC
2047  09                ADD   HL, BC
2048  E5                PUSH  HL
2049  7D                LD    A, L
204A  DD 21 FB 2F       LD    IX, 2FFBH
204E  CD 3C 06          CALL  063CH
2051  E1                POP   HL
2052  7C                LD    A, H
2053  DD 21 F9 2F       LD    IX, 2FF9H
2057  CD 3C 06          CALL  063CH
```

```
205A   3E 10          LD    A,10H
205C   32 F7 2F       LD    (2FF7),A
205F   32 F8 2F       LD    (2FF8),A
2062   C3 F4 00       JP    00F4H
```

*实验七的参考程序

软件时钟程序

```
2000   DD 21 00 21          LD    IX,2100H
2004   DD 66 00             LD    H,(IX+0)
2007   DD 6E 01             LD    L,(IX+1)
200A   DD 4E 02             LD    C,(IX+2)
200D   CD 50 20    BEGIN:    CALL  DELAY
2010   79                   LD    A,C
2011   C6 01                ADD   A,1
2013   27                   DAA
2014   4F                   LD    C,A
2015   FE 60                CP    60H
2017   20[  ]               JR    NZ,GOON
2019   0E 00                LD    C,0
201B   7D                   LD    A,L
201C   C6 01                ADD   A,1
201E   27                   DAA
201F   6F                   LD    L,A
2020   FE 60                CP    60H
2022   20[  ]               JR    NZ,GOON
2024   2E 00                LD    L,0
2026   7C                   LD    A,H
2027   C6 01                ADD   A,1
2029   27                   DAA
202A   67                   LD    H,A
202B   FE 24                CP    24H
202D   20[  ]               JR    NZ,GOON
202F   26 00                LD    H,0
2031   7C          GOON:    LD    A,H
2032   DD 21 F7 2F          LD    IX,2FF7
2036   CD 3C 06             CALL  063CH
2039   7D                   LD    A,L
```

```
203A  DD 21 F9 2F          LD    IX,2FF9
203E  CD 3C 06             CALL  063CH
2041  79                   LD    A,C
2042  DD 21 FB 2F          LD    IX,2FFB
2046  CD 3C 06             CALL  063CH
2049  CD 80 20             CALL  DISPLY
204C  C3                   JP    BEGIN
2050  F5           DELAY:  PUSH  AF
2051  C5                   PUSH  BC
2052  D5                   PUSH  DE
2053  E5                   PUSH  HL
2054  06 32                LD    B,50
2056  CD 60 20     LOOP:   CALL  DELAY1
2059  05                   DEC   B
205A  C2 56 20             JP    NZ,LOOP
205D  E1                   POP   HL
205E  D1                   POP   DE
205F  C1                   POP   BC
2060  F1                   POP   AF
2061  C9                   RET
2062  26 0A        DELAY1: LD    H,0AH
2064  2E DC        TIME1:  LD    L,0DCH
2066  2D           TIME2:  DEC   L
2067  00                   NOP
2068  C2 66 20             JP    NZ,TIME2
206B  25                   DEC   H
206C  DD 21 00 00          LD    IX,0
2070  C2 64 20             JP    NZ,TIME1
2073  C9                   RET
2080  21 F7 2F     DISPLY: LD    HL,2FF7H
2083  06 20                LD    B,20H
2085  5E           DISUP1: LD    E,(HL)
2086  16 00                LD    D,0
2088  3E 00                LD    A,0
208A  D3 8C                OUT   (8C),A
208C  DD 21 A6 07          LD    IX,07A6H
2090  DD 19                ADD   IX,DE
2092  DD 7E 00             LD    A,(IX+0)
2095  D3 88                OUT   (88H),A
```

```
2097    7B                      LD      A, B
2098    D3 8C                   OUT     (8CH), A
209A    1E 2D                   LD      E, 2DH
209C    1D              DISUP2: DEC     E
209D    3E 00                   LD      A, 00H
209F    20[  ]                  JR      NZ, DISUP2
20A1    3E 01                   LD      A, 01H
20A3    B8                      CP      B
20A4    28[  ]                  JR      Z, DISUP3
20A6    23                      INC     HL
20A7    CB 38                   SRL     B
20A9    18[  ]                  JR      DISUP1
20AB    C9              DISUP3: RET
```

*实验九的参考程序

1. 在RAM中建立0—FF共256个数程序

```
2120    21 00 20                LD      HL, 2000H
2123    AF                      XOR     A
2124    77                      LD      (HL), A
2125    23                      INC     HL
2126    06 FF                   LD      B, 0FFH
2128    3C              LOOP:   INC     A
2129    77                      LD      (HL), A
212A    23                      INC     HL
212B    10[  ]                  DJNZ    LOOP
212D    76                      HALT
```

2. EPROM的校验程序

```
2130    11 00 10                LD      DE, 1000H
2133    21 00 20                LD      HL, 2000H
2136    DD 21 00 22             LD      IX, 2200H
213A    01 00 01                LD      BC, 0100H
213D    1A              LOOP1:  LD      A, (DE)
213E    BE                      CP      (HL)
213F    28[  ]                  JR      Z, NEXT
```

```
2141    DD 75 00            LD      (IX+0), L
2144    DD 74 01            LD      (IX+1), H
2147    DD 23               INC     IX
2149    DD 23               INC     IX
214B    13          NEXT:   INC     DE
214C    23                  INC     HL
214D    0B                  DEC     BC
214E    78                  LD      A, B
214F    B1                  OR      C
2150    20[  ]              JR      NZ, LOOP1
2152    76                  HALT
```

附 录 2

第二部分 实验中的参考程序

实验二的参考程序

在显示器上每隔 1 秒显示字符0-9的程序。

```
LF:     EQU   0AH
CR:     EQU   0DH
START:  LD    B,0
        PUSH  BC
LOOP:   LD    E,CR
        LD    C,2
        CALL  5
        LD    E,LF
        LD    C,2
        CALL  5
        POP   BC
        OR    A
        LD    A,B
        ADD   A,1
        DAA
        AND   0FH
        LD    B,A
        PUSH  BC
        OR    30H
        LD    E,A
        LD    C,2
        CALL  5
        CALL  DELAY
        JR    LOOP
DELAY:  PUSH  AF
        PUSH  BC
        PUSH  DE
        PUSH  HL
        LD    B,50
LOOP11: CALL  DELAY1
```

```
            DEC     B
            JP      NZ, LOOP11
            POP     HL
            POP     DE
            POP     BC
            POP     AF
            RET
DELAY1;     LD      H, 0AH
TIME1:      LD      L, 0DCH
TIME2:      DEC     L
            NOP
            JP      NZ,  TIME2
            DEC     H
            LD      IX,  0
            JP      NZ,  TIME1
            RET
            END     START
```

实验三的参考程序

1．在CRT上循环显示字母A—Z的程序

```
CR:         EQU     0DH
LF:         EQU     0AH
START:      LD      A, 40H
            PUSH    AF
LOOP:       LD      E, CR
            LD      C, 2
            CALL    5
            LD      E, LF
            LD      C, 2
            CALL    5
            POP     AF
            INC     A
            CP      5BH
            JP      NZ, GOON
            LD      A, 41H
GOON:       PUSH    AF
            LD      E, A
            LD      C, 2
```

```
            CALL   5
            CALL   DELAY
            JR     LOOP
;延时程序 DELAY 与上一个实验中的程序一样，
;这儿不再重复。
            END    START
```

2．在CRT上循环显示00-59的程序

```
LF:         EQU    0 AH
CR:         EQU    0 DH
START:      LD     B, 0
            PUSH   BC
LOOP:       LD     E, CR
            LD     C, 2
            CALL   5
            LD     E, LF
            LD     C, 2
            CALL   5
            OR     A
            POP    BC
            LD     A, B
            ADD    A, 1
            DAA
            CP     60H
            JR     NZ, NEXT
            LD     A, 0
NEXT:       LD     B, A
            PUSH   BC
            LD     HL, DATA+2
            LD     (HL), '$'
            LD     HL, STORE
            LD     (HL), A
            LD     DE, DATA
            LD     A, 30H
            RLD
            LD     (DE), A
            INC    DE
            RLD
            LD     (DE), A
            LD     DE, DATA
```

```
        LD      C,9
        CALL    5
        CALL    DELAY
        JR      LOOP
; DELAY程序与前面的相同。
        END     START
```

实验四　实时时钟程序

```
CR:     EQU     0DH
LF:     EQU     0AH
START:  LD      E,':'
        LD      C,2
        CALL    5
        LD      DE,BUFFER
        LD      C,10
        CALL    5
        LD      DE,BUFFER+2
        LD      A,(DE)
        AND     0FH
        LD      (DE),A
        INC     DE
        LD      A,(DE)
        AND     0FH
        LD      (DE),A
        INC     DE
        INC     DE
        LD      A,(DE)
        AND     0FH
        LD      (DE),A
        INC     DE
        LD      A,(DE)
        AND     0FH
        LD      (DE),A
        INC     DE
        INC     DE
        LD      A,(DE)
        AND     0FH
        LD      (DE),A
```

```
        INC     DE
        LD      A,(DE)
        AND     0FH
        LD      (DE),A
        LD      DE,BUFFER+2
        LD      A,(DE)
        CALL    MUL10
        LD      H,A
        INC     DE
        INC     DE
        LD      A,(DE)
        CALL    MUL10
        LD      L,A
        INC     DE
        INC     DE
        LD      A,(DE)
        CALL    MUL10
        LD      C,A
LOOP:   CALL    DELAY
        LD      A,C
        ADD     A,1
        DAA
        LD      C,A
        CP      60H
        JP      NZ,DISPY
        LD      C,0
        LD      A,L
        ADD     A,1
        DAA
        LD      L,A
        CP      60H
        JP      NZ,DISPY
        LD      L,0
        LD      A,H
        ADD     A,1
        DAA
        LD      H,A
        CP      24H
        JP      NZ,DISPY
```

```
        LD      H,0
DISPY;  LD      DE,BLOCK
        LD      A,CR
        LD      (DE),A
        INC     DE
        LD      A,LF
        LD      (DE),A
        INC     DE
        LD      A,H
        CALL    TRAN
        INC     DE
        LD      A,':'
        LD      (DE),A
        INC     DE
        LD      A,L
        CALL    TRAN
        INC     DE
        LD      A,':'
        LD      (DE),A
        INC     DE
        LD      A,C
        CALL    TRAN
        INC     DE
        LD      A,'$'
        LD      (DE),A
        PUSH    BC
        PUSH    DE
        PUSH    HL
        LD      DE,BLOCK
        LD      C,9
        CALL    5
        POP     HL
        POP     DE
        POP     BC
        JP      LOOP
MUL10:  ADD     A,A
        DAA
        LD      B,A
        ADD     A,A
```

```
        DAA
        ADD     A,A
        DAA
        ADD     A,B
        DAA
        LD      B,A
        INC     DE
        LD      A,(DE)
        ADD     A,B
        RET
TRAN:   LD      B,A
        SRL     A
        SRL     A
        SRL     A
        SRL     A
        OR      30H
        LD      (DE),A
        INC     DE
        LD      A,B
        AND     0FH
        OR      30H
        LD      (DE),A
        RET
; 1秒延时子程序 DELAY 同前，这儿略去。
BUFFER:
LENGTH: DB      10
LENT1:  DS      1
LINE:   DS      10
BLOCK:  DS      12
        END     START
```

实验五的参考程序

1. 输入两位十进制数子程序

```
INBCD:  PUSH    DE
        PUSH    BC
        LD      E,':'
        LD      C,2
        CALL    5
```

```
        LD      DE, INBUF
        LD      C, 10
        CALL    5
        LD      DE, INBUF+2
        LD      A, (DE)
        SLA     A
        SLA     A
        SLA     A
        SLA     A
        LD      B, A
        INC     DE
        LD      A, (DE)
        AND     0FH
        OR      B
        POP     BC
        POP     DE
        RET
INBUF:
PINBUF: DB      2
BINBUF: DS      1
LENG:   DS      2
```

2. 输出在HL中的4位十进制数子程序

```
CR:     EQU     0DH
LF:     EQU     0AH
PUTBCD: PUSH    AF
        PUSH    BC
        PUSH    DE
        LD      DE, OUTBUF
        LD      A, CR
        LD      (DE), A
        INC     DE
        LD      A, LF
        LD      (DE), A
        INC     DE
        LD      A, H
        SRL     A
        SRL     A
        SRL     A
        SRL     A
```

```
        OR      30H
        LD      (DE),A
        INC     DE
        LD      A,H
        AND     0FH
        OR      30H
        LD      (DE),A
        INC     DE
        LD      A,L
        SRL     A
        SRL     A
        SRL     A
        SRL     A
        OR      30H
        LD      (DE),A
        INC     DE
        LD      A,L
        AND     0FH
        OR      30H
        LD      (DE),A
        INC     DE
        LD      A,'$'
        LD      (DE),A
        LD      DE,OUTBUF
        LD      C,9
        CALL    5
        POP     DE
        POP     BC
        POP     AF
        RET
OUTBUF: DS      7
```

3. 若干个输入的两位十进制数相加程序

```
COUNT:  EQU     10
START:  LD      HL,0
        LD      B,COUNT
ADDBCD: CALL    INBCD
        ADD     A,L
        DAA
        LD      L,A
```

```
        LD      A,H
        ADC     A,0
        DAA
        LD      H,A
        DJNZ    ADDBCD
        CALL    PUTBCD
        JP      0
        END     START
```

4. 输入两个两位十进制数相乘的程序

```
MULT:   LD      HL,0
        CALL    INBCD
        LD      D,A
        CALL    INBCD
        CP      D
        JR      C,LESS
        LD      B,D
        LD      D,A
        JR      MULT1
LESS:   LD      B,A
MULT1:  LD      A,B
        OR      A
        JR      Z,OUTP
        LD      A,L
        ADD     A,D
        DAA
        LD      L,A
        LD      A,H
        ADC     A,0
        DAA
        LD      H,A
        LD      A,B
        DEC     A
        DAA
        LD      B,A
        JR      MULT1
OUTP:   CALL    PUTBCD
        JP      0
        END     MULT
```

*实验六的参考程序

1．从键盘输入两位带符号十进制数子程序

```
INDN:   PUSH  BC
        PUSH  DE
        PUSH  HL
        LD    E,':'
        LD    C,2
        CALL  5
        LD    DE,INBUF
        LD    C,10
        CALL  5
        LD    HL,INBUF+2
        LD    A,(HL)
        CP    '+'
        JR    Z,POS
        LD    A,1
        JR    SIGN1
POS:    LD    A,0
SIGN1:  LD    (SIGN),A
        INC   HL
        LD    B,2
CHTOB:  LD    A,(HL)
        AND   0FH
        LD    (HL),A
        INC   HL
        DJNZ  CHTOB
        LD    HL,INBUF+3
        LD    A,(HL)
        ADD   A,A
        LD    B,A
        ADD   A,A
        ADD   A,A
        ADD   A,B
        INC   HL
        ADD   A,(HL)
        LD    B,A
        LD    A,(SIGN)
```

```
        CP      0
        JR      Z,RT
        LD      A,B
        NEG
        LD      B,A
RT:     LD      A,B
        POP     HL
        POP     DE
        POP     BC
        RET
INBUF:
PINBUF: DB      3
BINBUF: DS      1
LENG:   DS      3
```

2．把在HL中的带符号的二进制数以十进制形式输出的子程序

```
PUTDN:  PUSH    AF
        PUSH    BC
        PUSH    DE
        PUSH    IX
        LD      IX,BUFFER
        LD      (IX),0DH
        INC     IX
        LD      (IX),0AH
        INC     IX
        LD      A,H
        OR      A
        JP      P,PLUS
        CPL
        LD      H,A
        LD      A,L
        CPL
        LD      L,A
        INC     HL
        LD      (IX),'-'
        JR      GOON
PLUS:   LD      (IX),'+'
GOON:   INC     IX
        LD      DE,2710H
        CALL    CHANGE
```

```
        LD      DE,03E8H
        CALL    CHANGE
        LD      DE,64H
        CALL    CHANGE
        LD      DE,0AH
        CALL    CHANGE
        LD      A,30H
        ADD     A,L
        LD      (IX),A
        INC     IX
        LD      (IX),'$'
        LD      DE,BUFFER
        LD      C,9
        CALL    5
        POP     IX
        POP     DE
        POP     BC
        POP     AF
        RET
CHANGE: LD      B,0
AGAIN:  OR      A
        SBC     HL,DE
        JR      C,DOWN
        INC     B
        JR      AGAIN
DOWN:   ADD     HL,DE
        LD      A,30H
        ADD     A,B
        LD      (IX),A
        INC     IX
        RET
BUFFER: DS      10
```

3．输入两个8位带符号数相乘的程序

```
CR:     EQU     0DH
LF:     EQU     0AH
SMULT:  LD      E,CR
        LD      C,2
        CALL    5
        LD      E,LF
```

```
        LD      C,2
        CALL    5
        CALL    INDN
        LD      L,A
        AND     80H
        LD      (SSGN1),A
        JP      P,POSIT
        LD      A,L
        NEG
        LD      L,A
POSIT:  LD      E,L
        PUSH    DE
        LD      E,CR
        LD      C,2
        CALL    5
        LD      E,LF
        LD      C,2
        CALL    5
        CALL    INDN
        LD      L,A
        AND     80H
        LD      (SSGN2),A
        JP      P,POSIT1
        LD      A,L
        NEG
        LD      L,A
POSIT1: LD      A,L
        POP     DE
        LD      HL,0
        LD      D,H
LLOOP:  SRL     A
        JP      NC,SHIFT
        ADD     HL,DE
SHIFT:  SLA     E
        RL      D
        OR      A
        JR      NZ,LLOOP
        LD      A,(SSGN1)
        LD      B,A
```

```
        LD      A,(SSGN2)
        XOR     B
        JR      Z,NNEXT
        LD      A,L
        CPL
        LD      L,A
        LD      A,H
        CPL
        LD      H,A
        INC     HL
NNEXT:  CALL    PUTDN
        JP      0
; INDN  略
; PUTDN 略
SSGN1:  DS      1
SSGN2:  DS      1
        END     SMULT
```

4．输入两位十进制数（正数）相除的程序

```
CR:     EQU     0DH
LF:     EQU     0AH
DIVID:  LD      E,CR
        LD      C,2
        CALL    5
        LD      E,LF
        LD      C,2
        CALL    5
        CALL    INDN
        LD      C,A
        PUSH    BC
        LD      E,CR
        LD      C,2
        CALL    5
        LD      E,LF
        LD      C,2
        CALL    5
        CALL    INDN
        LD      D,A
        OR      A
        JP      Z,DONE
```

```
        POP     BC
        LD      B,0
        LD      E,8
LOOP:   SLA     C
        RL      B
        LD      A,B
        SUB     D
        JR      C,NEEXT
        INC     C
        LD      B,A
NEEXT:  DEC     E
        JR      NZ,LOOP
        LD      L,C
        LD      H,0
        CALL    PUTDN
        LD      L,B
        LD      H,0
        CALL    PUTDN
DONE:   JP      0
;INDN:  略
;PUTDN: 略
        END     DIVID
```

*实 验 七

1. 从键盘输入数值范围为+32767≥X≥−32768的十进制数程序

```
GETDN:  PUSH    AF
        PUSH    BC
        PUSH    DE
        LD      E, ':'
        LD      C, 2
        CALL    5
        LD      DE, BUFFER
        LD      C, 10
        CALL    5
        LD      DE, BUFFER+1
        EX      DE, HL
        LD      B, (HL)
        DEC     B
```

```
            INC     HL
            LD      A, 0
            LD      (SIGN), A
            LD      A, (HL)
            CP      '+'
            JR      Z, NEXT11
            LD      A, 1
            LD      (SIGN), A
NEXT11:     PUSH    BC
NEXT12:     INC     HL
            LD      A, (HL)
            AND     0FH
            LD      (HL),A
            DJNZ    NEXT12
            POP     BC
            LD      DE, BUFFER+3
            LD      HL, 0
LOOP11:     PUSH    BC
            ADD     HL, HL
            PUSH    HL
            POP     BC
            ADD     HL, HL
            ADD     HL, HL
            ADD     HL, BC
            LD      A, (DE)
            INC     DE
            LD      C,A
            LD      B,0
            ADD     HL, BC
            POP     BC
            DJNZ    LOOP11
            LD      A, (SIGN)
            OR      A
            JR      Z, DONE11
            LD      A, H
            CPL
            LD      H, A
            LD      A, L
            CPL
```

```
        LD      L, A
        INC     HL
DONE11: POP     DE
        POP     BC
        POP     AF
        RET
```

2. 由键盘输入在CRT上显示的程序

```
START:      CALL    GETDN
            CALL    PUTDN
            JP      0
; GETDN     略
; PUTDN     略
            GND     START
```

*实验八的参考程序

从键盘输入一个数据块，用气泡分类法排序，然后输出排序后的数据块的程序

```
START:  LD      E, ':'
        LD      C, 2
        CALL    5
        LD      DE, DATA11
        LD      C, 10
        CALL    5
        LD      B, 0
        LD      HL, DATA11+2
        LD      IX, DATA22
LOOP:   LD      A, (HL)
        CP      '$'
        JP      Z, CONT
        CP      ' '
        JP      NZ, GOFR
        INC     B
        INC     HL
        LD      A, (HL)
GOFR:   CP      '+'
        JR      Z, PLU1
        LD      A, 1
        JR      MINUS
PLU1:   LD      A, 0
```

```
MINUS:  LD      (SIG1), A
        INC     HL
        LD      A, (HL)
        AND     0FH
        ADD     A, A
        LD      D, A
        ADD     A, A
        ADD     A, A
        ADD     A, D
        PUSH    AF
        INC     HL
        LD      A, (HL)
        AND     0FH
        LD      E, A
        POP     AF
        ADD     A, E
        LD      E, A
        LD      A, (SIG1)
        OR      A
        JR      Z, PLU2
        LD      A, E
        NEG
        LD      E, A
PLU2:   LD      A, E
        LD      (IX), A
        INC     IX
        INC     HL
        JR      LOOP
CONT:   INC     B
        LD      E, 2
        LD      IY, DATA33
LOOP22: LD      IX, DATA22
        LD      (IY), B
        LD      C, B
        LD      D, B
        LD      B, 0
        ADD     IX, BC
        DEC     IX
LOOP33: LD      A, (IX)
```

```
        LD      B, (IX-1)
        CP      B
        JP      M, SFALG
        JP      PE, EXCH
        JR      UNEX
SFALG:  JP      PE, UNEX
EXCH:   LD      (IX-1), A
        LD      (IX), B
        LD      C, D
UNEX:   DEC     IX
        DEC     D
        LD      A, D
        CP      E
        JP      P, LOOP33
        LD      A, C
        LD      B, (IY)
        CP      B
        JR      Z, DON11
        INC     E
        JP      LOOP22
DON11:  LD      IX, DATA44
        LD      (IX), 0DH
        INC     IX
        LD      (IX), 0AH
        INC     IX
        LD      HL, DATA22
        LD      B, (IY)
LOOP4:  LD      A, (HL)
        OR      A
        JR      Z, PLU3
        NEG
        LD      (IX), '-'
        JR      GOON3
PLU3:   LD      (IX), '+'
GOON3:  INC     IX
        LD      D, 0
        LD      E, 0AH
AGAIN1: SUB     E
        JR      C, TEN
```

```
        INC     D
        JR      AGAIN1
TEN:    PUSH    AF
        LD      A, 30H
        ADD     A, D
        LD      (IX), A
        INC     IX
        POP     AF
        ADD     A, E
        ADD     A, 30H
        LD      (IX), A
        INC     IX
        LD      (IX), ' '
        INC     HL
        INC     IX
        DJNZ    LOOP4
        LD      (IX), '$'
        LD      DE, DATA44
        LD      C, 9
        CALL    5
        JP      0000H
DATA11:
DAT1:   DB      100
DAT2:   DS      1
DAT3:   DS      100
DATA22: DS      100
DATA33: DS      1
SIG1:   DS      1
DATA44: DS      100
        END     START
```

附录3 硬件与接口实验的参考程序

实验二的参考程序

1. CTC通道级连的参考程序

```
                        ORG     2000H
2000    3E 00   START:  LD      A, 00H
2002    D3 84           OUT     (84H), A    ; 送中断矢量
```

```
2004    3E 25           LD      A, 25H
2006    D3 84           OUT     (84H), A  ; 通道0方式字
2008    3E C6           LD      A,0C6H
200A    D3 84           OUT     (84H), A  ; 通道0时常数
200C    3E D5           LD      A, 0D5H
200E    D3 85           OUT     (85H), A  ; 通道1方式字
2010    3E 05           LD      A, 05H
2012    D3 85           OUT     (85H), A  ; 通道1时常数
2014    3E 21           LD      A, 21H
2016    ED 47           LD      I, A
2018    ED 5E           IM2
201A    FB              EI
201B    76        LOOP: HALT
201C    18 FD           JR      LOOP
                        ORG     2100H
2100    00 22           DW      2200H
                        ORG     2200H
2200    FB        INT:  EI
2201    ED 4D           RETI
```

2. 在运行过程中改变时常数的程序

（1）主程序

```
                        ORG     2000H
2000    3E 00     START: LD     A, 00H
2002    D3 84           OUT     (84H), A  ; CTC的中断矢量
2004    3E 21           LD      A, 21H
2006    ED 47           LD      I, A
2008    ED 5E           IM2
200A    FB              EI
200B    3E 00           LD      A, 00     ; 要显示的字符编码
200D    D3 88           OUT     (88H), A  ; 输出给段形锁存器
200F    3E A5           LD      A, 0A5H
2011    D3 84           OUT     (84H), A  ; 通道0方式控制字
2013    3E C4           LD      A, 0C4H
2015    D3 84           OUT     (84H), A  ; 通道0时常数
2017    1E 02           LD      E, 2
2019    0E 10           LD      C, 10H    ; 一种时常数的显示次数
201B    06 05           LD      B, 5      ; 延时时间的扩展倍数
201D    16 01           LD      D, 01H    ; 设数位初值
201F    7A        LOOP: LD      A, D
```

```
2020    D3 8C           OUT   (8CH), A  ; 数位控制字输出给数位
                                          锁存器
2022    76              HALT
2023    C3 1F 20        JP    LOOP
```

（2）中断服务程序

```
                        ORG   2100H
2100    00 22           DW    2200H
                        ORG   2200H
2200    05        INT:  DEC   B           ; 软件扩展次数减 1
2201    C2 2C 22        JP    NZ, RT    ; 未到 0，返回
2204    CB 02           RLC   D           ; 定时到使数位左移一位
2206    06 05           LD    B, 5      ; 重新设置扩展倍数
2208    0D              DEC   C           ; 移位次数减 1
2209    C2 2C 22        JP    NZ, RT    ; 未到 0，返回
220C    0E 10           LD    C, 10H    ; 已到 0，重新设置移位
                                          ; 次数
220E    3E 03           LD    A, 03H    ; 使CTC复位
2210    D3 84           OUT   (84H), A  ; 重新设置方式字
2212    3E A5           LD    A, 0A5H
2214    D3 84           OUT   (84H), A
2216    3E 64           LD    A, 64H    ; 设新的时常数
2218    D3 84           OUT   (84H), A
221A    1D              DEC   E
221B    C2 2C 22        JP    NZ, RT
221E    1E 02           LD    E, 2
2220    3E 03           LD    A, 03H
2222    D3 84           OUT   (84H), A
2224    3E A5           LD    A, 0A5H
2226    D3 84           OUT   (84H), A
2228    3E C4           LD    A, 0C4H   ; 恢复老的时常数
222A    D3 84           OUT   (84H), A
222C    FB        RT:   EI
222D    ED 4D           RETI
```

实验五　电子秒表实验

1. 主程序

```
                        ORG   2000H
2000    01 90 20  START:  LD    BC, CTC0V ; CTC的中断矢量值→BC
```

```
2003    78                  LD      A, B
2004    ED 47               LD      I, A
2006    79                  LD      A, C
2007    D3 84               OUT     (CTC0 ), A; 给通道 0 输中断矢量
2009    3E C5               LD      A, 0C5H  ; 通道 0 的方式字
                                             ; 使通道 0 工作于计数
                                             ; 方法
200B    D3 84               OUT     (CTC0), A
200D    3E 01               LD      A, 1     ; 通道 0 时常数
200F    D3 84               OUT     (CTC0), A
2011    ED 5E               IM2
2013    FB                  EI
2014    AF          ZERO:   XOR     A
2015    4F                  LD      C, A     ; 按钮次数计数器清 0.
2016    21 94 20            LD      HL, MS10 ; 置放分、秒、1/10秒
                                             ; 和1/100单元的指针
2019    06 04               LD      B, 4
201B    77          LOOP1:  LD      (HL), A
201C    23                  INC     HL
201D    10 FC               DJNZ    LOOP1    ; 置分、秒、1/10秒、
                                             ; 1/100秒
                                             ; 单元的初值为 0。
201F    CD 50 23    WAIT1:  CALL    DISUP0   ; 调显示程序
2022    79                  LD      A, C
2023    FE 01               CP      1        ; 是按了一次按钮吗?
2025    20 F8               JR      NZ, WAIT1; 否, 循环等待
2027    3E A5               LD      A, 0A5H  ; 是, 设CTC 通道1工作
2029    D3 85               OUT     (CTC1), A ; 于定时方式
202B    3E 4E               LD      A, 4EH   ; 设置时常数
202D    D3 85               OUT     (CTC1), A ; 开始计数
202F    79          WAIT2:  LD      A, C
2030    FE 02               CP      2        ; 是否按了两次按钮?
2032    20 FB               JR      NZ, WAIT2 ; 否, 循环等待
2034    3E 03               LD      A, 03H   ; 是, 送CTC通道复位字
2036    D3 85               OUT     (CTC1), A ; 停止计时。
2038    CD 50 23    WAIT3:  CALL    DISUP0   ; 调显示程序
203B    79                  LD      A, C
203C    FE 03               CP      3        ; 是否按了三次按钮?
203E    30 D4               JR      NC, ZERO ; ≥0, 转至ZERO
```

```
2040    18  F6              JR      WAIT3       ; 否则，循环等待
```

2．中断服务程序

（1）CTC0中断服务程序

```
                            ORG     2050H
2050    0C          CTC0S:  INC     C
2051    FB                  EI
2052    ED 4D               RETI
```

（2）CTC1中断服务程序

```
2054    D9          CTC1S:  EXX
2055    08                  EX      AF, AF'
2056    21 94 20            LD      HL, MS10
2059    34                  INC     (HL)        ; 1/100秒值加 1
205A    3E 0A               LD      A, 0AH
205C    BE                  CP      (HL)        ; 1/100秒值已为10吗?
205D    20  1F              JR      NZ, DONE    ; 否，转显示。
205F    36  00              LD      (HL), 0     ; 是，1/100秒值清0
2061    23                  INC     HL
2062    7E                  LD      A, (HL)
2063    3C                  INC     A           ; 1/10秒值加 1
2064    27                  DAA
2065    77                  LD      (HL),A      ; 送回
2066    3E 10               LD      A, 10H
2068    BE                  CP      (HL)        ; 1/10秒值已达到10
                                                ; 吗?
2069    20  13              JR      NZ, DONE    ; 否，转至显示
206B    36  00              LD      (HL), 0     ; 1/10秒值清 0
206D    23                  INC     HL
206E    7E                  LD      A, (HL)
206F    3C                  INC     A
2070    27                  DAA
2071    77                  LD      (HL), A     ; 秒值加 1
2072    3E 60               LD      A, 60H
2074    BE                  CP      (HL)        ; 秒值已达60?
2075    2007                JR      NZ, DONE    ; 否，转至显示
2077    36  00              LD      (HL), 0     ; 是，秒值清 0
2079    23                  INC     HL
207A    7E                  LD      A, (HL)
207B    3C                  INC     A
207C    27                  DAA
```

```
207D   77                    LD    (HL), A      ; 分值加 1
207E   CD 50 23    DONE:     CALL  DISUP0       ; 调显示程序
2081   D9                    EXX
2082   08                    EX    AF, AF'
2083   FB                    EI
2084   ED 4D                 RETI
```

3. 向量和工作单元表

```
                             ORG   2090H
2090   50  20      CTC0V:    DW    CTC0S
2092   54  20      CTC1V:    DW    CTC1S
2094               MS10:     DS    1            ; 1/100秒计数单元
2095               MS100:    DS    1            ; 1/10秒计数单元
2096               SECOND:   DS    1            ; 秒计数单元
2097               MINUTE:   DS    1            ; 分计数单元
0084               CTC0:     EQU   84H          ; CTC通道 0 端口地址
0085               CTC1:     EQU   85H          ; CTC 通道 1
                                                ; 端口地址
```

4. 显示子程序

```
                             ORG   2350H
2350   DD 21 F7 2F DISUP0:   LD    IX, DISMEM   ; 把分、秒和1/10秒
2354   3A 97 20             LD    A,(MINUTE)   ; 值分别送相应的
2357   CD 3C 06             CALL  UFOR1        ; 显示缓冲区
235A   DD 21 F9 2F          LD    IX, DSMEM2
235E   3A 96 20             LD    A,(SECOND)
2361   CD 3C 06             CALL  UFOR1
2364   DD 21 FB 2F          LD    IX, DSMEM 4
2368   3A 95 20             LD    A, (MS100)
236B   CD 3C 06             CALL  UFOR1
236E   21  F7 2F            LD    HL, DISMEM
2371   06  20               LD    B, 20H
2373   5E           DISUP1: LD    E, (HL)      ; 取要显示的字符
2374   16  00               LD    D, 0
2376   3E 00                LD    A, 0
2378   D3 8C                OUT   (DIGLH), A   ; 封锁显示
237A   DD 21 A6 07          LD    IX, SEGPT    ; 显示的字模表首址→
                                                IX
237E   DD 19                ADD   IX, DE       ; 查表
2380   DD 7E 00             LD    A, (IX)      ; 取出要显示的字符的
                                                ; 字模编码
```

```
2383    D3 88              OUT (SEGLH),A  ; 送至段形锁存器
2385    78                 LD  A, B
2386    D3 8C              OUT (DIGLH),A  ; 数位控制字送数位锁
                                          ; 存器
2388    1E 70              LD  E, 70H     ; 置 1 ms延时
238A    1D          DISUP2: DEC E
238B    3E 00              LD  A, 0
238D    BB                 CP  E
238E    20 FA              JR  NZ, DISUP2
2390    3E 01              LD  A, 01H
2392    F8                 CP  B
2393    28 05              JR  Z, DISUP3
2395    23                 INC HL
2396    CB 38              SRL B
2398    18 D9              JR  DISUP1
239A    C9          DISUP3: RET
                           ORG 2FF7H
2FF7                DISMEM: DS  1
2FF8                DSMEM1: DS  1
2FF9                DSMEM2: DS  1
2FFA                DSMEM3: DS  1
2FFB                DSMEM4: DS  1
2FFC                DSMEM5: DS  1
063C                UFOR1:  EQU 063CH
008C                DIGLH:  EQU 8CH
07A6                SEGPT:  EQU 07A6H
0088                SEGLH:  EQU 88H
                    END START
```

实验六　交通控制信号灯实验

1. 主程序

```
                    ORG 2000H
2000    01 90 20 START: LD  BC, PIOAV
2003    78              LD  A, B
2004    ED 47           LD  I, A
2006    79              LD  A, C
2007    D3 82           OUT (PIOAC), A  ; 设端口A中断矢量
2009    01 92 20        LD  BC, PIOBV
```

```
200C    79                  LD    A,C
200D    D3 83               OUT   (PIOBC),A   ; 设端口B中断矢量
200F    3E 0F               LD    A,0FH
2011    D3 83               OUT   (PIOBC),A   ; 设端口B为输出方式
2013    3E 87               LD    A,87H
2015    D3 83               OUT   (PIOBC),A   ; 中断允许
2017    ED 5E               IM2
2019    FB          LOOP:   EI
201A    3E 63               LD    A,63H
201C    D3 81               OUT   (PIOBD),A   ; 使大道绿灯亮，小道
                                              ; 红灯亮
201E    18 F9               JR    LOOP        ; 循环等待中断
```

2. 中断服务程序

（1）端口A中断服务程序

```
                            ORG   2020H
2020    D1          PIOAS:  POP   DE          ; 不要断点地址
2021    21 49 20            LD    HL,LOOP1    ; 要求的中断返回地址
                                              ; →HL
2024    E5                  PUSH  HL          ; 入栈
2025    ED 4D               RETI              ; 利用返回指令，无条
                                              ; 件
                                              ; 转移至要求的返回地
                                              ; 址
```

（2）端口B中断服务程序

```
                            ORG   2027H
2027    CD 70 20    PIOBS:  CALL  D6SEC       ; 调用延时6秒程序
202A    3E 65               LD    A,65H
202C    D3 81               OUT   (PIOBD),A   ; 使大道黄灯，小道红
                                              ; 灯
202E    CD 67 20            CALL  D4SEC       ; 调用延时4秒程序
2031    3E 36               LD    A,36H
2033    D3 81               OUT   (PIOBD),A   ; 使大道红灯，小道绿
                                              ; 灯
2035    FB                  EI
2036    3E CF               LD    A,0CFH
2038    D3 82               OUT   (PIOAC),A   ; 设端口A工作于位控
                                              ; 方式
203A    3E 15               LD    A,15H       ; 端口A的I/O选择
                                              ; 字
```

```
203C   D3 82              OUT   (PIOAC), A   ; 使位0、2与4为输入
203E   3E D7              LD    A, 0D7H     ; 端口A中断控制字
                                            ; 允许中断，“与”关
                                            ; 系，
2040   D3 82              OUT   (PIOAC), A   ; 低电平有效，下跟屏
                                            ; 蔽字
2042   3E EA              LD    A, 0EAH     ; 端口A的中断屏蔽字
2044   D3 82              OUT   (PIOAC), A   ; 允许位0、2与4中断
2046   CD 79 20           CALL  D25SEC      ; 调用25S延时程序
2049   3E D7      LOOP1:  LD    A, 0D7H
204B   D3 82              OUT   (PIOAC), A   ; 再设端口A中断控制
                                            ; 字
204D   3E FF              LD    A, 0FFH
204F   D3 82              OUT   (PIOAC), A   ; 使端口A的中断全屏
                                            ; 蔽
2051   3E 56              LD    A, 56H
2053   D3 81              OUT   (PIOBD), A   ; 使大道红，小道黄
2055   CD 67 20           CALL  D4SEC       ; 调用4秒延时程序
2058   DI                 POP   DE          ; 丢去断点
2059   21 19 20           LD    HL, LOOP    ; 返回地址→HL
205C   E5                 PUSH  HL
205D   ED 4D              RETI
```

3. 延时子程序

```
                          ORG   205FH
205F   06 32      D1SEC:  LD    B, 50
2061   CD 4F 06   LOOP2:  CALL  D20MS       ; 调20ms延时子程序
2064   10 FB              DJNZ  LOOP2
2066   C9                 RET
2067   0E 04      D4SEC:  LD    C, 4
2069   CD 5F 20   LOOP3:  CALL  D1SEC
206C   0D                 DEC   C
206D   20 FA              JR    NZ, LOOP3
206F   C9                 RET
2070   0E 06      D6SEC:  LD    C, 6
2072   CD 5F 20   LOOP4:  CALL  D1SEC
2075   0D                 DEC   C
2076   20 FA              JR    NZ, LOOP4
2078   C9                 RET
2079   0E 19      D25SEC: LD    C, 25
```

```
207B   CD 5F 20   LOOP5:  CALL D1SEC
207E   0D                DEC  C
207F   20 FA             JR   NZ, LOOP5
2081   C9                RET
```

4. 中断矢量和工作单元表

```
                         ORG  2090H
2090   20 20     PIOAV:  DW   PI0AS
2092   27 20     PIOBV:  DW   PI0BS
064F             D20MS:  EQU  064FH     ; 20ms 延时子程序入
                                        ; 口地址
0081             PIOBD:  EQU  81H       ; 通道B数据端口地址
0082             PIOAC:  EQU  82H       ; 通道A控制端口地址
0083             PIOBC:  EQU  83H       ; 通道B控制端口地址
                         END  START
```

实验七　A/D转换实验

1. 主程序

```
                         ORG  2000H
2000   3E 0F     START:  LD   A, 0FH
2002   D3 83             OUT  (PIOBC), A   ; 设PIO端口B为输出
                                           ; 方式
2004   ED 56             IM1
2006   FB                EI
2007   D3 94     LOOP:   OUT  (PS5), A     ; 命令开始A/D转换
2009   00                NOP
200A   18 FB             JR   LOOP
```

2. 中断服务程序

```
                         ORG  0038H        ; IM1的入口地址
0038   C3 D3 2F          JP   2FD3H        ; 在ROM中已固化了
                                           ; 此条指令
                         ORG  2FD3H        ; 在RAM的2FD3H
2FD3   C3 10 20          JP   INT          ; 处应设置一条无条件
                                           ; 转移指令
                         ORG  2010H
2010   DB 94     INT:    IN   A, (PS5)     ; 从A/D转换输入数
                                           ; 据
2012   32 32 20          LD   (VALUE), A   ; 存入内存
2015   2F                CPL
```

```
2016    D3 81                   OUT   (PIOBD), A ; 取反后经 PIO端口 B
                                                 ; 输出
                                                 ; 控制 LED指示灯
2018    2F                      CPL
2019    CB 3F                   SRL   A
201B    CB 3F                   SRL   A
201D    CB 3F                   SRL   A
201F    CB 3F                   SRL   A
2021    32  33 20              LD    (MSB4), A  ; 高 4 位存入显示缓冲
                                                 ; 区
2024    3A 32 20                LD    A, (VALUE)
2027    E6 0F                   AND   0FH
2029    32  34 20               LD    (LSB4), A  ; 低 4 位存入显示缓冲
                                                 区
202C    CD 40 20                CALL  DISPLY
202F    FB                      EI
2030    ED 4D                   RETI
```

3. 显示子程序

```
                                ORG   2040H
2040    21 33 20        DISPLY: LD    HL, MSB 4
2043    06 02                   LD    B, 02H
2045    DD 21 A607      FIND:   LD    IX, SEGPT
2049    16 00                   LD    D, 0
204B    5E                      LD    E, (HL)
204C    DD 19                   ADD   IX, DE
204E    DD 7E 00                LD    A, (IX)
2051    D3 88                   OUT   (SEGLH), A
2053    78                      LD    A, B
2054    D3 8C                   OUT   (DIGLH), A
2056    CD 66 20                CALL  DEALY
2059    3E 00                   LD    A, 0
205B    B8                      CP    B
205C    CA 65 20                JP    Z, RT
205F    CB 3B                   SRL   B
2061    23                      INC   HL
2062    C3 45 20                JP    FIND
2065    C9              RT:     RET
                                ORG   2066H
2066    0E FF           DEALY:  LD    C, 0FFH
```

```
2068    0D          D1:     DEC   C
2069    AF                  XOR   A
206A    B9                  CP    C
206B    C2 68 20            JP    NZ, D1
206E    C9                  RET
                    PIOBC:  EQU   83H
                    PIOBD:  EQU   81H
                    PS5:    EQU   94H
                    SEGPT:  EQU   07A6H
                    DIGLH:  EQU   8CH
                    SEGLH:  EQU   88H
                            ORG   2032H
2032                VALUE:  DS    1
2033                MSB4:   DS    1
2034                LSB4:   DS    1
                            END   START
```

实验八　A/D 与 D/A 正逆变换实验

1. 主程序

```
                            ORG   2000H
2000    3E 0F       BEGIN:  LD    A, 0FH
2002    D3 83               OUT   (PIOBC), A ; 设定 PIO 端口 B 为
                                               ; 输出方式
2004    ED 56               IM1
2006    FB                  EI
2007    D3 94       LOOP:   OUT   (PS5), A   ; 命令 A/D 转换启动
2009    00                  NOP
200A    18 FB               JR    LOOP
                            ORG   0038H
0038    C3 D3 2F            JP    2FD3H
                            ORG   2FD3H
2FD3    C3 10 20            JP    INT
                            ORG   2010H
2010    DB 94       INT:    IN    A, (PS5)   ; 输入A/D转换后的数据
2012    32 66 20            LD    (VALUE),A  ; 存入内存
2015    2F                  CPL
2016    D3 81               OUT   (PIOBD), A ;取反后由PIO端口B输出
2018    2F                  CPL
```

```
2019    D3 9C                OUT   (PS7), A     ; 输出给 D/A
201B    FE FA                CP    0FAH         ; 比较是否>250
201D    DA 22 20             JP    C, COMP      ; 否转至 COMP
2020    3E FA                LD    A, 0FAH      ; 是, 令A=250
2022    01 00 00     COMP:   LD    BC, 0000H
2025    CD 70 20             CALL  CONV         ; 调用转换程序
2028    CD 40 20             CALL  DISP         ; 调用显示程序
202B    DD 21 67 20          LD    IX, FIRST
202F    3E 00                LD    A, 00H
2031    DD 77 00             LD    (IX+0), A
2034    DD 77 01             LD    (IX+01), A
2037    DD 77 02             LD    (IX+02), A
203A    FB                   EI
203B    ED 4D                RETI
                             ORG   2040H
2040    21 67 20     DISP:   LD    HL, FIRST
2043    06 04                LD    B, 04H
2045    DD 21 A6 07  FIND:   LD    IX, SEGPT
2049    16 00                LD    D, 0
204B    SE                   LD    E, (HL)
204C    DD 19                ADD   IX, DE
204E    DD 7E 00             LD    A, (IX)
2051    D3 88                OUT   (SEGLH), A
2053    78                   LD    A, B
2054    D3 8C                OUT   (DIGLH), A
2056    CD A0 20             CALL  DEALY
2059    3E 01                LD    A, 01H
205B    B8                   CP    B
205C    CA 65 20             JP    Z, RT
205F    23                   INC   HL
2060    CB 38                SRL   B
2062    C3 45 20             JP    FIND
2065    C9           RT:     RET
2066                 VALUE:  DS    1
2067                 FIRST:  DS    3
                             ORG   2070H
2070    21 67 20     CONV:   LD    HL, FIRST
2073    36 00                LD    (HL), 0
2075    D6 32        CON1:   SUB   32H          ; A≥32H?
```

```
2077    DA 7F 20            JP    C, R1          ; 否，转R1
207A    04                  INC   B
207B    70                  LD    (HL), B        ; 是B 加1
207C    C3 75 20            JP    CON1
207F    23          R1:     INC   HL
2080    C6 32               ADD   A, 32H
2082    D6 05       R2:     SUB   05H
2084    DA 8C 20            JP    C, R3
2087    0C                  INC   C
2088    71                  LD    (HL), C
2089    C3 82 20            JP    R2
208C    23          R3:     INC   HL
208D    FE 00               CP    00H
208F    CA 95 20            JP    Z, TRANS
2092    C6 05               ADD   A, 05H
2094    87                  ADD   A, A           ; 最低位×2
2095    77          TRANS:  LD    (HL), A
2096    C9                  RET
                    PIOBC:  EQU   83H
                    PIOBD:  EQU   81H
                    PS5:    EQU   94H
                    PS7:    EQU   9CH
                    SEGPT:  EQU   07A6H
                    DIGLH:  EQU   8CH
                    SEGLH:  EQU   88H
                            ORG   20A0H
20A0    0E DF       DELAY:  LD    C, 0DFH
20A2    0D          D1:     DEC   C
20A3    AF                  XOR   A
20A4    B9                  CP    C
20A5    C2 A2 20            JP    NZ, D1
20A8    C9                  RET
                            END   BEGIN
```

实验九　串行通讯实验

1. 主程序

```
                            ORG   2000H
2000    AF          STARTI: XOR   A              ; 置接收标志
```

```
2001    16 34                   LD    D, 34H     ; 8251A 的命令字,
                                                 ; 允许接收，且请求发送
2003    DD 21 72 20             LD    IX, ADDR2
2007    18 08                   JR    RTPRM
2009    3E 01        START0:    LD    A, 01H     ; 置发送标志
200B    16 11                   LD    D, 11H     ; 8251A命令字，允许发送
200D    DD 21 70 20             LD    IX, ADDR1
2011    ED 56        RTPRM:     IM1
2013    08                      EX    AF, AF'    ; 保存传送标志
2014    21 72 20                LD    HL, ADDR2; 要发送给接收方的
                                                 ; 接收区首地址和传送长度
                                                 ; 存放的存贮器的首地址→
                                                 ; HL
2017    01 94 04                LD    BC, 0494H ; 端口地址→C，传送个
                                                   数→B
201A    3E FF                   LD    A, 0FFH
201C    D3 95                   OUT   (95H), A   ; 给 8251A输送方式控制字
201E    7A                      LD    A, D
201F    D3 95                   OUT   (95H), A   ; 输送命令字，8251A使
                                                 ; 工作于运行状态
2021    FB                      EI
2022    76           WAIT0:     HALT             ; 等待发送中断
2023    20 FD                   JR    NZ, WAIT0  ; 接收区的首址和传送长度
                                                 ; 传送完否，未完转 WAI-
                                                   T0 等待
2025    DD 6E 00                LD    L, (IX+0)
2028    DD 66 01                LD    H, (IX+1)  ; 取出传送区首址→HL
202B    ED 5B 74 20             LD    DE,        ; 取出传送长度→DE
                                      (LENGTH)
202F    76           WAIT1:     HALT             ; 等待第二次传送的中断
2030    1B                      DEC   DE         ; 传送了一个字符，长度
                                                   减 1
2031    7A                      LD    A, D
2032    B3                      OR    E
2033    20 FA                   JR    NZ, WAIT1  ; 检查传送完否，否转
                                                   WAIT1
2035    F3           EXIT:      DI
2036    08                      EX    AF, AF'    ; 取出传送标志
2037    B7                      OR    A
```

```
2038    28 03                   JR      Z, NOWAIT   ; 是接收,转NOWAIT
203A    CD 4F 06                CALL    064FH       ; 调 20ms延迟子程序
203D    21 10 11    NOWAIT:     LD      HL, 1110H   ; 准备显示 "P",表示
                                                    ; 是传送
                                                    ; 正常结束。
2040    3E 40       DISP:       LD      A, 40H      ; 8251A 的复位命令字
2042    D3 95                   OUT     (95H), A    ; 停止 8251A工作
2044    22 FB 2F                LD      (2FFBH),HL  ; 把要显示的字符送显
                                                    ; 示缓冲区
                                                    ; (最右面两位)
2047    FB                      EI
2048    C3 F4 00                JP      DISUP       ; 转至显示
```

2. 中断服务程序

```
                                ORG     2050H
2050    DB 95       INTSV:      IN      A, (95H)    ; 读入 8251A 状态
2052    1F                      RRA
2053    1F                      RRA                 ; 检查 RxRDY
2054    30 08                   JR      NC, OUTPUT  ; 不是RxRDY请求,转
                                                    ; OUTPUT
2056    E6 0E       INPUT:      AND     0EH         ; 是接收中断, 检查接
                                                    ; 收过程有否错误
2058    20 13                   JR      NZ, ERR2    ; 有错, 转至 ERR2
205A    ED A2                   INI                 ; 否则, 输入
205C    FB                      EI
205D    C9                      RET
205E    17          OUTPUT:     RLA                 ; 检查 TxRDY
205F    30 04                   JR      NC, ERR1    ; 若不是TxRDY请求,
                                                    ; 则系统
                                                    ; 中有别的中断源, 转
                                                    ; 至 ERR1
2061    ED A3                   OUTI                ; 输出数据
2063    FB                      EI
2064    C9                      RET
2065    21 0E 12    ERR1:       LD      HL, 120 EH  ; 准备显示 "E!",表示
                                                    ; 系统中
                                                    ; 有其它中断源
2068    18 03                   JR      ERR
206A    21 0E 11    ERR2:       LD      HL, 110EH   ; 准备显示 "EP"表示
                                                    ; 传送出错
```

```
206D    E1          ERR:    POP   AF          ; 恢复栈指针
206E    18 D0               JR    DISP
2070                ADDR1:  DS    2
2072                ADDR2:  DS    2
2074                LENGTH: DS    2
                            ORG   2FD3H
2FD3    C3 50 20            JP    INTSV
; DISUP  略
```

附　录　4

一、调试程序（DEBUGGER）使用介绍

初学者在编制程序时，往往会有不少错误；即使是一个有经验的程序员，在编制大的、复杂的程序时，也会出错误。特别是汇编语言程序长、语句多就更容易出错。若是格式上或是语法上的错误，汇编程序在汇编时会指出来。但其它的错误就不好发现，若程序经过运行，发现结果不对。则错误发生在什么地方呢？在一个较长程序中，怎么能很快地找到错误所在呢？这就像在单板机中一样，要借助于调试程序。在CDOS中有一个文件名为DEBUG的调试程序。下面，我们只是简要地介绍一下使用方法和主要的命令。

（一）调试程序的调用

1．必须在CDOS下，也即在出现CDOS的提示符时，才能调用调试程序。

2．有两种调用格式

A．DEBUG 〈CR〉

A．DEBUG 文件标记 〈CR〉

后一种格式中的文件标记，就是要调试的用户程序。CDOS先调入DEBUG程序，然后由DEBUG调入用户程序，放至从0100H开始的RAM中。

3．在调试程序调入后，出现指示符'－'。表示系统在DEBUG程序的管理之下，可以输入DEBUG命令。

（二）显示和修改内存与寄存器内容的命令

要了解程序进行是否正确，首先要能了解程序运行以后在内存和寄存器中的内容。

1．显示内存内容的命令DM（D）

内存的内容用16进制数表示，每行显示16个单元的内容，每个单元之间用空格分开。一行的最左面是所显示的这一行的第一个单元的地址；而最右面是这一行各个字节的ASCII字符（没有相应字符的用．表示）。

命令格式：

DM 〈CR〉

DM 开始地址-结束地址 〈CR〉

DM 开始地址 〈CR〉

DM 开始地址 S 范围 〈CR〉

其中，第一种格式，是从上次显示的最后一个地址之后显示8行（128个字节）内存内容。

第二种是从指定的开始地址，显示到指定的结束地址之间的全部存贮单元的内容。

第三种是从指定的开始地址起显示8行。

第四种是从指定的开始地址起，显示由S后的数值所指定的这么多个单元的内容。所以，第四种与第二种是相同的。S后面的范围值＝(结束地址－开始地址)＋1。

例：

```
-DM〈CR〉
0100   C3 03 01 3E 0D CD 99 01 3E 0A CD 99 01 3E 3E 3D  C..>.M..>.M..>>M
0110   99 01 CD D6 04 21 87 01 71 21 73 01 06 01 36 00  ..MV.!..Q!S...6.
0120   34 3A 87 01 BE D2 2F 01 2B 05 20 F4 C3 03 01 48  4:..>R/.+.TC..H
0130   54 5D 1B CD 76 05 0D 28 0F 1A 96 28 E3 80 B9 28  T].MV..(...(C.9C
0140   DF 90 81 B8 28 DA 18 EA 23 04 3A 87 01 B8 30 C8  -..8(Z.J#.:..80N
0150   F5 C5 E5 3A 87 01 21 73 01 06 00 4E CD BO 04 23  UEE:..!S...NMO.#
0160   3D 20 F8 3E 0D CD 99 01 3E 0A CD 99 01 E1 C1 F1  =X>.M..>.M..AAQ
0170   C3 28 01 C9 44 4D 23 23 23 23 E5 21 00 00 29 29  C(.IDM####E!..))
```

DM　命令可简写为D。

2．修改内存命令SM(S)

这个命令可用来修改内存单元的内容。其格式为：

SM　起始地址　〈CR〉

DEBUG先显示指定的起始地址，然后是该单元的内容，接着我们就可以按以下的方法之一进行操作。

（1）送入该地址的新内容（用两位16进制数表示），然后用〈CR〉结束。则新内容将取代原有内容，然后地址加1，显示下一个地址及内容。

（2）送入用单引号′　′括起来的字符串。则该字符串的ASCII码按地址顺序放入内存。

（3）上两种方式，可按任意次序在一行中输入。

（4）只输入〈CR〉，则不改变该单元的内容，接着显示下一个的地址和内容。

（5）-〈CR〉使地址减1。用来改正刚输入的内存内容。

（6）·〈CR〉　使修改命令结束，返回DEBUG，可输入别的命令。

例：

```
-SM 1000〈CR〉
1000    88  3E〈CR〉
1001    01  'THIS IS' 41 44 41 54 42〈CR〉
100D    46  -〈CR〉
100C    42  41〈CR〉
100D    46  ·〈CR〉
-DM S10〈CR〉
1000   3E 54 48 49 53 20 49 53 41 44 41 54 41 46 00 00 >THIS.IS ADATAF..
```

SM　命令可缩写为S。

3．显示寄存器命令DR

若为了调试程序，在程序中设置断点（方法在后面介绍）。则从断点返回DEBUG时，所有寄存器的内容被保存起来，然后用DR命令可显示所有寄存器的内容。其格式为：

```
-DR〈CR〉
SZHVNCE    A=00 BC=0000  DE=0000  HL=0000  SP=B300 PC=0100
           LD  A,00
SZHVNC     A′=00  BC′=0000 DE′=0000 HL′=0000 IX=0000  IY=0000
           I=00
```

其中第一行中的字母"SZHVNCE"表示标志寄存器F，而第二行中的表示F′。如某一位标志被置为1，则该标志被显示出来；若标志为0，则为空格。若某一条指令执行后，只有S和P/V标志被置1，则显示的为"SV"。在第一行中还有E标志，它表示中断触发器IFF的状态，E表示开中断。

第一行的后面是寄存器A、BC、DE、HL、SP和PC的内容。最后是该指令的反汇编形式，这是即将要执行的指令。

第二行以F′开始，然后是A′、BC′、DE′、HL′、IX、IY和I的内容。若反汇编指令有地址部分，则其相对值列在此行最后。

进入DEBUG时，各寄存器的初值为：

```
-DR〈CR〉
    A=00  BC=0000  DE=0000  HL=0000  SP=B300  PC=0100  JP 0103
    A′=00 BC′=0000 DE′=0000 HL′=0000 IX=0000  IY=0000  I =00
```

4．修改寄存器命令Sr

用Sr命令可以改变任一个寄存器的内容，这样就可以给它们置以初值，或为了调试目的而改变它们的值。其中r是要修改的寄存器名，它们是A、F、BC、DE、HL、IX、IY、SP、PC，A′、F′、BC′、DE′、HL′。其中F包括E标志。

在改变标志寄存器F或F′时，要置为1的标志位，必须输入；未输入的标志位，命令后被置0，而不管其初值是什么。

修改其他寄存器时要输足相应的一或二字节数（若输入的数，超过了寄存器的容量，则取最后输入的那部分）。如果没有输入新值，或输入有错，则寄存器保持原值。

例：

```
-SF〈CR〉
  SZHN  ZCE〈CR〉
-SF〈CR〉
  Z CE  SVCE〈CR〉
-SA〈CR〉
  A=00  3E〈CR〉
-SBC〈CR〉
  BC=0000  8010〈CR〉
-SDE〈CR〉
  DE=0000  342050〈CR〉
-SDE〈CR〉
```

```
  DE=2050  〈CR〉
-SHL〈CR〉
  HL=0000  3000〈CR〉
-SF'〈CR〉
            S  C 〈CR〉
-SIX〈CR〉
  IX=0000  5020〈CR〉
-DR〈CR〉
  S VCE A=3E BC=8010 DE=2050 HL=3000 SP=B300 PC=0100 JP 0103
  S  C  A'=00 BC'=0000 DE'=0000 HL'=0000 IX=5020 IY=0000 I=00
```

（三）断点、启动和跟踪命令

1. 断点命令B，BX

（1）命令B——设置或显示永久性断点

为了调试程序，或者是为了检查一下前面的程序运行是否正确；或者是为了缩小查错的范围，常常要在程序中设置断点。DEBUG 程序有两种设置断点的方法，一种是在启动（运行）命令G中设置，这样设置的断点，当命令执行完，断点也就消除了。另一种是设置永久性的断点，这样的断点设置完了以后，只要没有用 BX 命令加以清除，则始终有效。

命令格式：

-B〈CR〉

-B〈断点1〉〈断点2〉……〈CR〉

前一种格式是显示现有的永久性断点。第二种格式用来设置，最多为 12 个永久性的断点。

每一个断点可以有三个参数，但不一定每个断点都要有这三个参数。

① R——显示寄存器和标志（选用）

② ADDR——断点的内存地址

③ ：COUNT——断点的重复次数（选用）

例：

```
-B 106〈CR〉
-B R 109:34 〈CR〉
-B 〈CR〉
BP=0106:0001
BP=R 0109:0034
```

前两个命令是设置两个永久性断点(可以一次设)；第三个命令是显示断点，可看到已设置了前两个命令所要求设置的断点。其中R，是表示当程序运行到设定的断点而停下来时，显示各寄存器的内容；COUNT34 是用来表示当程序运行经过断点第 34 次时才停下来。

当然这样的要求是由 DEBUG 程序控制实现的。即当 DEBUG 程序执行每条指令之前，都要将指令地址与断点比较。若不符合则执行指令；若相符合，则看断点次数减

法计数器是否为1，若不为1，则减1后执行该指令，否则就停下来返回DEBUG。

由于Z80有多字节指令，所以所设的断点地址必须是多字节指令的第一个字节的地址。

（2）命令BX——清除永久性断点

格式为：

-BX〈CR〉

-BX 地址1 地址2 ……〈CR〉

前一种格式用来清除所有已设立的永久性断点。后一种格式，清除命令中所指定地址的断点。例：

-B〈CR〉

-B 0103 0106 0120;7 R012D;4 〈CR〉

-B〈CR〉

BP= 0103 ;0001

BP= 0106 ;0001

BP= 0120 ;0007

BP=R 012D ;0004

-BX 0103 0106 〈CR〉

-B 〈CR〉

BP= 0120 ;0007

BP=R 012D ;0004

2．启动运行命令G

用G命令运行调试用户程序。命令的格式为：

-G 起始地址/〈断点1〉〈断点2〉……〈CR〉

从指定的起始地址开始运行。若没有给定起始地址，则从现行的PC值开始执行(一开始时PC=0100)。

这个命令先从寄存器内容暂存区取回各个寄存器的内容，然后从起始地址（或现行PC值）开始运行。若有断点，则将RST 30H指令放在断点处，而原来的指令代码被放到断点信息区，而在内存的0030H单元处放一条转至断点处理子程序入口的无条件转移指令JP BAE1。这样，当执行到断点地址时，就能转入DEBUG中的断点处理程序。它把所有寄存器的内容存入内存暂存器，并显示出现；中断地址的指令将同时显示出来，并恢复所有G命令中用RST 30H代替的断点处的用户指令。

使用G命令的注意事项：

（1）只能对放在RAM中的程序设置断点。因为要在每一断点处用RST 30H指令代替用户程序中的指令，而在遇到断点时又要恢复用户指令。

（2）最多只允许设置12个断点。

（3）因为在0030—0032H单元，存放的是JP BAE1即转至断点处理程序的入口的转移指令，故它们不能被用户占用。

（4）断点一次有效，下次还要用G命令重新设置。因为每一个断点都有一个断点次数计数器，故在G命令中与在B命令中一样，可设置重复次数，它放在每一断点之后，用

括号括起来。如：

-G103/106(0E)125

表示程序只有执行经过地址 106 第14次时才停下来，或第一次执行到地址 125 时就停下来。

3. 跟踪命令T、C

跟踪命令的格式为：

T〈CR〉

T行数〈CR〉

前一个是步进命令，每执行一条指令都要显示寄存器的内容，并停下来等待新的调试命令。重复使用这样的命令，就可以步进调试，每次执行一条指令，并可检查执行的结果。

后一个是多行跟踪命令，每执行一条指令都要显示寄存器的内容，但只有执行完最后一条指令才停下，返回 DEBUG，可接收新的调试命令。

若内存中有以下程序

```
0103        LD      A，00
0105        LD      B，5
0107        INC     A
0108        EDC     B
0109        JR      NZ，0107
010B        JP      0103
-T〈CR〉
  A=00 BC=0000 DE=0000 HL=0000 SP=B300 PC=0105 LD B，5
  A'=00 BC'=0000 DE'=0000 HL'=0000 IX=0000 IY=0000 I=00
-T〈CD〉
  A=00 BC=0500 DE=0000 HL=0000 SP=B300 PC=0107 INC A
  A'=00 BC'=0000 DE'=0000 HL'=0000 IX=0000 IY=0000 I=00
-T4〈CR〉
  A=01 BC=0500 DE=0000 HL=0000 SP=B300 PC=0108 DEC B
  A'=00 BC'=0000 DE'=0000 HL'=0000 IX=0000 IY=0000 I=00
  A=01 BC=0400 DE=0000 HL=0000 SP=B300 PC=0109 JR NZ，0107
  A'=00 BC'=0000 DE'=0000 HL'=0000 IX=0000 IY=0000 I=00
  A=01 BC=0400 DE=0000 HL=0000 SP=B300 PC=0107 INC A
  A'=00 BC'=0000 DE'=0000 HL'=0000 IX=0000 IY=0000 I=00
  A=02 BC=0400 DE=0000 HL=0000 SP=B300 PC=0108 DEC B
  A'=00 BC'=0000 DE'=0000 HL'=0000 IX=0000 IY=0000 I=00
-
```

C命令与T命令类似，不同的是它把子程序当作一条指令来对待。这样对于调试主程序是很方便的。

（四）汇编和反汇编命令

有了以上这些命令就可以对用户程序进行调试。但是每一次发现错以后要修改都必须重新调用编辑程序，修改以后要重新汇编、连接、存盘才能运行。这个过程是比较麻烦的也是很浪费时间的。所以，若要反复调试一个程序段，就可以利用DEBUG中的汇编程序在内存中来建立一个程序，然后就在DEBUG下运行。调试和修改。

另外，若有一个机器码的文件，要了解这个文件的功能是相当困难的，常常要用反汇编把机器码文件变为源程序。

DEBUG具有这样的汇编和反汇编的功能：

1．逐行汇编命令A

A命令可以把一行的汇编语言汇编成机器码，且存到内存中，然后再汇编下一行。但A命令不能用标号和符号，也不能用宏指令和伪指令，其他与一般的汇编语言相同。

命令的格式为：

A　起始地址　〈CR〉

于是DEBUG就显示出指定的绝对地址，及该地址所存放的内容经过反汇编后的指令；于是就等待我们输入新的汇编指令，如输入无错误，则把它翻译成机器码，存入该地址。接着给出下一个地址，及其原内容的反汇编指令，等待输入新的汇编指令，直至退出A命令。

输入的新的汇编指令中，地址允许用表达式；地址和数均用16进制表示。

若原存放的指令不用修改，直接按〈CR〉键即可。

若要退出A命令，则要输入·〈CR〉。

若输入的新的汇编指令中有错，则显示？号，不改变地址，可再输入新的指令。

由于Z80中的指令有多字节的，所以A命令中的起始地址，应是一条指令的第一个字节的地址，否则给出的反汇编指令不正确，整个程序也可能乱套。

例：

```
-A  0100  〈CR〉
0100    NOP    LD    A, 0   〈CR〉
0102    NOP    LD    B, 5   〈CR〉
0104    NOP    INC   A      〈CR〉
0105    NOP    DEC   B      〈CR〉
0106    NOP    JR    NZ,0104〈CR〉
0108    NOP    ·〈CR〉
-
```

2．反汇编命令L

L命令能把在内存中的机器码，以汇编指令的形式在CRT和打印机上列出来。其格式为：

L〈CR〉

L起始地址〈CR〉

L起始地址，结束地址〈CR〉

L起始地址，S范围〈CR〉

第一种命令是从现在地址开始列16行反汇编指令；第二种命令是从开始地址起列16

行反汇编指令；第三种是从开始地址起反汇编，列出指令直至结束地址；第四种是从开始地址起反汇编，列出的行数由S后面的范围数字所指定。

例：

```
-L〈CR〉
0100    LD      A，0
0102    LD      B，5
0104    INC     A
0105    DEC     B
0106    JR      NZ，0104
0108    NOP
0109    NOP
010A    NOP
010B    NOP
010C    NOP
010D    NOP
010E    NOP
010F    NOP
0110    NOP
0111    NOP
0112    NOP
```

（五）举例

下面我们举一个简单的程序例子，看一下DEBUG命令是如何应用的。

要求建立一个能在CRT上循环显示0-9的程序。

1．用逐行编辑命令A建立程序

```
A.DEBUG〈CR〉
-A〈CR〉
0100    NOP     LD      B，0〈CR〉
0102    NOP     LD      A，B〈CR〉
0103    NOP     ADD     A，1〈CR〉
0105    NOP     LD      B，A〈CR〉
0104    NOP     LD      E，A〈CR〉
0107    NOP     CALL    0120〈CR〉
010A    NOP     CALL    0150〈CR〉
010D    NOP     JR      0102〈CR〉
0110    NOP     •〈CR〉
-
```

2．用L命令来检查输入的内容

```
-L0100，0110〈CR〉
0100    LD      B，0
```

```
0102     LD      A, B
0103     ADD     A, 1
0105     LD      B, A
0106     LD      E, A
0107     CALL    0120
010A     CALL    0150
010D     JR      0102
0110     NOP
-
```

3. 建立子程序

为了先检查主程序的工作，在子程序中可以先不安排任何内容，而用一个返回指令令其返回主程序。

```
-A  0120 〈CR〉
0120    NOP     RET〈CR〉
0121    NOP     •〈CR〉
-A  0150 〈CR〉
0150    NOP     RET〈CR〉
0151    NOP     •〈CR〉
-
```

4. 设置断点送行

要调试主程序，检查运行的结果与我们要求的是否符合，就要设置断点。

检查和设置断点的过程如下：

（1）先检查断点的现有情况；

（2）将永久性断点设置在010D处，其计数值设置为9；

（3）在0120处设置另一个断点；

（4）显示断点设置情况；

（5）清除设置的永久性断点；

（6）再显示断点情况，看是否清除干净；

（7）再在010D处设置永久性断点；

（8）执行程序，在执行过5次0102处时中断程序；

（9）再检查断点情况。

```
-B〈CR〉
-B    010D:9〈CR〉
-B    0120 〈CR〉
-B
 BP=    010D:0009
 BP=    0120 :0001
-BX〈CR〉
-B 〈CR〉
```

```
-B  R010D:9〈CR〉
-G0100/0102(5)〈CR〉
A=04  BC=0400  DE=0004  HL=0000  SP=B300  PC=0102  LD  A,B
A′=00  BC′=0000  DE′=0000  H′L=0000  IX=0000  IY=0000  I=00
-B〈CR〉
 BP=R    010D:5
-B  R 0150:14  〈CR〉
-G〈CR〉
A=09  BC=0900  DE=0009  HL=0000  SP=B300  PC=010D  JR  0102
-A′=00  BC′=0000  DE′=0000  HL′=0000  IX=0000  IY=0000  I=00
-
```

程序运行到这儿，看来都是正确的，B中的值由初始值0开始，每循环一次加1。关键是再循环一次又应该返回到0，为了检查运行是否正确，可以采用单步运行的方法。

```
-T〈CR〉
A=09  BC=0900  DE=0009  HL=0000  SP=B300  PC=0102  LD  A,B
A′=00  BC′=0000  DE′=0000  HL′=0000  IX=0000  IY=0000  I=00
-T2〈CR〉
A=09  BC=0900  DE=0009  HL=0000  SP=B300  PC=0103  ADD A,1
A′=00  BC′=0000  DE′=0000  HL′=0000  IX=0000  IY=0000  I=00
A=0A  BC=0900  DE=0009  HL=0000  SP=B300  PC=0105  LD  B,A
A′=00  BC′=0000  DE′=0000  HL′=0000  IX=0000  IY=0000  I=00
-
```

从以上的结果可看到，由于加法是二进制运算，所以09加1以后是0A，因此必须要用十进制调正指令。另外，送CRT显示时，应送0—9的ASCII码到寄存器E。这样，就要用汇编命令A来加以修改。

```
-A  0100  〈CR〉
0100    LD      B, 0        〈CR〉
0102    LD      A, B        〈CR〉
0104    ADD     A, 1        〈CR〉
0105    LD      B, A        DAA〈CR〉
0106    LD      E, A        LD    B, A〈CR〉
0107    CALL    0120        OR    30   〈CR〉
0109    LD      BC, 50CD    LD    E, A〈CR〉
010A    CALL    0150        CALL  0120 〈CR〉
010D    JR      0102        CALL  0150 〈CR〉
0110    NOP                 JR    0102 〈CR〉
0112    NOP                 .〈CR〉
-
```

用反汇编指令检验输入的内容。

```
-L  0100, 0112  〈CR〉
0100     LD      B, 0
0102     LD      A, B
0103     ADD     A, 1
0105     DAA
0106     LD      B, A
0107     OR      30
0109     LD      E, A
010A     CALL    0120
010D     CALL    0150
0110     JR      0102
0112     NOP
-BX  〈CR〉
-B R 0110:0F〈CR〉
-G0100/010A(7)〈CR〉
A=37  BC=0007  DE=0037  HL=0000  SP=B300 PC=010A CALL 0120
A'=00 BC'=0000 DE'=0000 HL'=0000 IX=0000 IY=0000 I=00
-
```

运行到这儿结果还是正确的，E中也是要显示字符的ASCII码。

```
-G〈CR〉
A=35  BC=0015  DE=0035  HL=0000  SP=B300  PC=0110  JR  0102
A'=00 BC'=0000 DE'=0000 HL'=0000 IX=0000  IY=0000  I=00
-
```

从以上的结果分析，由于有DAA指令，当09加1以后B的低4位恢复0，但由于有进位前四位就变为1，这样还是达不到只循环显示0—9的目的。所以，还要把高四位屏蔽掉。

```
-A  0100  〈CR〉
0100     LD      B, 0        〈CR〉
0102     LD      A, B        〈CR〉
0103     ADD     A, 1        〈CR〉
0105     DAA                 〈CR〉
0106     LD      B, A        AND   0F   〈CR〉
0108     JR      NC, 016A    LD    B, A 〈CR〉
0109     LD      E, A        OR    30   〈CR〉
010B     JR      NZ, 010D    LD    E, A 〈CR〉
010C     LD      BC, 50CD    CALL  0120 〈CR〉
010F     LD      BC, FE18    CALL  0150 〈CR〉
0112     NOP                 JR    0102 〈CR〉
0114     NOP                 .〈CR〉
```

```
-L  0100, 0114 〈CR〉
0100     LD      B, 0
0102     LD      A, B
0103     ADD     A, 1
0105     DAA
0106     AND     0F
0108     LD      B, A
0109     OR      30
010B     LD      E, A
010C     CALL    0120
010F     CALL    0150
0112     JR      0102
0114     NOP
-BX 〈CR〉
-B R 0112:63 〈CR〉
-G  0100/010C (0F) 〈CR〉
A=35  BC=0005  DE=0035  HL=0000  SP=B300  PC=010C  CALL 0120
A'=00 BC'=0000 DE'=0000 HL'=0000 IX=0000  IY=0000  I=00
-G〈CR〉
A=39  BC=0009  DE=0039  HL=0000  SP=B300  PC=0112  JR  0102
A'=00 BC'=0000 DE'=0000 HL'=0000 IX=0000  IY=0000  I=00
-
```

程序运行正确，主程序调试完毕。子程序可以用类似方法进行调试，不再赘述。

二、动态调试工具DDT(Dynamic Debugging Tool) 使用介绍

DDT 是在 CP/M 操作系统支持下的，类似于 CDOS 中的 DEBUG 的一种动态调试工具。它主要用来调试和修改用户以汇编语言编写的源程序。DDT 能对 • COM 或 • HEX(16进制文件) 类型的目标文件进行处理。限于篇幅，我们下面只讨论对于 • COM 文件如何进行处理。DDT 有许多功能子命令，在这儿我们只限于介绍适用于实验中需要的一些常用的子命令。

CP/M 是以8080汇编为基础的，所以 DDT 也是以8080汇编为基础。即 DDT 中的汇编命令是对以8080汇编语言写的源程序进行汇编；DDT 中的反汇编命令，能把机器码(8080能识别和执行的)反汇编为8080的汇编形式。这一些是不全适用于 Z80的汇编语言的。但是，一些其他的主要命令，如检查与修改存贮器内容；启动与设置断点命令；追踪命令等对于 Z80 的汇编语言的程序也是适用的。检查与修改寄存器的命令也是针对8080的，这对于 Z80 也是可以参考的。所以，已经经过汇编、连接和存盘的变为机器码的 Z80的汇编语言所写的源程序，也可以用 DDT 来进行调试。

(一) DDT 的调用

DDT是CP/M 的一个外部命令，它以 DDT.COM 的形式存放在系统盘的文件区。调

用的主要命令形式为：

A>DDT 〈CR〉

A>DDT 文件标记 〈CR〉

其中A>是CP/M的提示符。调用命令必须在CP/M下才能执行。

前一种形式是把DDT调入内存，然后系统在DDT的管理之下，用DDT中的命令再输入要调试的目标文件。

后一种形式是把DDT以及以文件标记（扩展名为 .COM）指明的要调试的目标文件一起调入内存。然后系统在DDT管理下，可以用命令对目标文件进行调试。

不论是哪一种命令，在DDT调入后，显示它的版本号，然后显示它的提示符“-”。表明系统已在DDT的管理之下。

（二）检查与修改内存内容的命令

1．检查内存内容命令

命令D用来检查指定的内存区域的内容，它以16进制的形式，把内存的内容在屏幕上显示出来。命令格式：

-D 〈CR〉

-DS 〈CR〉

-DS,f〈CR〉

其中，“-”是DDT的提示符，DDT的各个命令，只有在出现DDT提示符“-”后才能运行。

第一种命令是从当前地址（DDT启动时的地址为0100H）开始，连续显示12行(192个字节）内存单元的内容。在机器码的右边用ASCII码来表示每一个单元的内容（若不能用相应的ASCII码表示的话，则用·表示)。例：

-D〈CR〉

```
0100  01  BC  0F  C3  3D  01  43  4F  50  59  52  49  47  48  54  20  ....=.COPYRIGHT
0110  20  43  29  20  31  39  30  2C  20  44  44  49  47  49  54  41  (C)1980, DIGITA
0120  4C  20  52  45  53  45  41  52  43  48  20  20  20  20  20  20  L  RESEARCH
0130  44  44  54  20  56  45  52  53  20  32  2E  32  24  31  00  02  DDT  VERS  2.2$1..
0140  05  C5  11  30  01  0E  09  CD  05  00  C1  21  07  00  7E  3D  ...0.....=
0150  90  57  1E  00  D5  21  00  02  78  B1  CA  65  01  0B  7E  12  .W.....X..e..{
0160  13  23  C3  58  01  D1  C1  E5  62  78  B1  CA  87  01  0B  7B  .#X....bX.....
0170  E6  07  C2  7A  01  E3  7E  23  E3  6F  7D  17  6F  D2  83  01  ...Z..~#.o}.o
0180  1A  84  12  13  C3  69  01  D1  2E  00  E9  0E  10  CD  05  00  .....i.........
0190  32  5F  1F  C9  21  66  1F  70  2B  71  2A  65  1E  EB  0E  11  2_...f.p+q*e....
01A0  CD  05  00  32  5F  1E  C9  11  00  00  0E  12  CD  05  00  32  ...._..........2
01B0  5F  1E  C9  21  68  1E  70  2B  71  2A  67  1E  EB  0E  13  CD  ....h*p+q.9...
-
```

若要继续显示，只要打入D〈CR〉即可。

上述的第二种命令格式，S即为要显示的初始地址，则此命令从指定初始地址开始连续显示12行。

第三种命令格式是从指定的起始地址S开始显示到指定的结束地址f。例：

-D 0100，013F 〈CR〉

```
0100  21  00  00  39  22  15  02  31  57  02  CD  C1  01  FE  FF  C2  1...9"..1W.....
```

```
0110  1B 01 11 F3 01 CD 9C 01 C3 51 01 3E 80 32 13 02  .........Q7.2..
0120  21 00 00 E5 CD A2 01 E1 DA 51 01 47 70 E6 0F C2  1.........Q.G}..
0130  44 01 CD 72 01 CD 59 01 0F DA 51 01 7C CD 8F 01  D..r..Y...Q.{...
```

2. 修改内存内容命令 S

S 命令显示指定的内存单元的地址和它的内容，并等待着输入。命令格式为：

-S s 〈CR〉

其中第二个 s 即为起始地址。例如：

-S 0100 〈CR〉

0100 C3

0100 是指定的地址，C3 即为原存放内容。若不需要修改，直接按回车键，就显示下一个地址及内容。若要修改，就可以打入两位 16 进制数，按回车键，则一方面新的内容就写入指定地址；另一方面显示下一单元的地址和内容等待修改。这样就可以一直进行下去，直至打入“·”号和回车来结束修改命令；或检测出一个无效的输入值为止。

```
-S 0100 〈CR〉
0100  C3  〈CR〉
0101  03  00 〈CR〉
0102  01  10 〈CR〉
0103  01  21 〈CR〉
0104  00  00 〈CR〉
0105  10  20 〈CR〉
0106  3E  〈CR〉
0107  20  00 〈CR〉
0108  76  〈CR〉
0109  00  ·〈CR〉
-
```

可以用同样的方法来检验新输入的内容是否正确。若有错的话，还可以进行修正。

```
- S0100 〈CR〉
0100  C3  〈CR〉
0101  00  〈CR〉
0102  10  〈CR〉
0103  21  〈CR〉
0104  00  〈CR〉
0105  20  〈CR〉
0106  3E  〈CR〉
0107  00  〈CR〉
0108  76  〈CR〉
0109  00  ·〈CR〉
-
```

（三）检查与修改寄存器内容

1. 检查寄存器内容的命令 X

命令格式：

- X 〈CR〉

C0Z0M0E0I0 A=00 B=0000 D=0000 H=0000 S=0100 P=0100 JP1000

其中C、Z、M、E和I分别为进位、零、符号、奇偶和辅助进位标志。每一个标志右面的值（0或1）即为此标志的现行状态。A为累加器，B即为寄存器对BC，D为寄存器对DE，H为寄存器对HL，S为堆栈指针SP，P为程序计数器PC。每一个等号右面的值，即为这些寄存器的现行值。最右面即为PC所指单元的指令的汇编形式。

2．修改寄存器内容命令

命令格式：

- Xr 〈CR〉

其中r即为要修改的某一寄存器号。此命令先显示指定寄存器的内容。然后等待输入新的内容。若不要修改，则按回车键，命令结束返回DDT；若要修改，则输入两位(对于累加器A）或四位（对于其他寄存器）16进制数（对于标志位则输入0或1）再按回车键，新输入的内容送至指定的寄存器，且结束命令返回DDT。例如：

-XC 〈CR〉
C0 1〈CR〉
-XC 〈CR〉
C1 〈CR〉
-XA 〈CR〉
A=00 20 〈CR〉
-XA 〈CR〉
A=20 〈CR〉
-XP 〈CR〉
P=0100 1000 〈CR〉
-XP 〈CR〉
P=1000 〈CR〉
-X 〈CR〉
C1Z0M0E0I0 A=20 B=0000 D=0000 H=0000 S=0100 P=1000 LXI H, 2000
-

（四）程序转移命令

G命令可以转移到执行调试的目标程序。在G命令中可以设置断点。但DDT中最多只允许设置两个断点。

命令格式为：

-G 〈CR〉
-Gs 〈CR〉
-Gs, b 〈CR〉
-Gs, b, c 〈CR〉
-G, b 〈CR〉
-G, b, c 〈CR〉

第一个命令是从PC中包含的当前值开始执行，没有设置断点。

第二个命令是从由s指定的起始地址开始执行，也没有设置断点。

第三个命令从由s指定的起始地址开始执行，设置了一个断点，断点地址由b指定。

第四个命令从由s指定的起始地址开始执行，在由b和c指定的地址处设置了两个断点。

第五个命令，从PC的当前值开始执行，在b处设置了一个断点。

第六个命令，从PC的当前值开始执行，在b和c处设置了两个断点。

若要设置有效的断点，则断点地址必须不同于起始地址，且要大于起始地址。当程序运行到断点地址时就停下来，返回DDT。所以，就可以用各种DDT命令来检查程序运行的中间结果。此时两个断点都清除了，若还需要的话，则要重新设置。若要接着运行，直接打入G命令即可。

（五）追踪命令

T命令是追踪程序以观察程序执行情况的命令，它允许追踪1至65535个程序步。每执行一个程序步，就把当时的CPU全状态显示出来。命令格式为：

-T〈CR〉

-Tn〈CR〉

前一种格式是执行一条指令就停下来，显示CPU的全状态，及下一条指令的地址。

后一种格式是执行由n所指定的程序步后，停下来显示所有各步的CPU的状态。

举例：

若自内存0100单元开始放有一个要调试的目标程序，其8080和Z80的汇编形式为：

```
START:  LXI  H, 1000H        START:  LD   HL, 1000H
        LXI  D, 2000H                LD   DE, 2000H
        MVI  B, 64H                  LD   B, 64H
LOOP:   MOV  A, M            LOOP:   LD   A, (HL)
        STAX D                       LD   (DE), A
        INX  H                       INC  HL
        INX  D                       INC  DE
        DCR  B                       DEC  B
        JNZ  LOOP                    JP   NZ, LOOP
        HLT                          HALT
```

```
-T〈CR〉
C0Z0M0E0I0 A=00 B=0000 D=0000 H=1000 S=0100 P=0103 LXID, 2000H *0103
-
```

T命令执行了一条在0100处的指令，HL＝1000H，反映了指令执行的结果是正确的。PC的值是下一条要执行的指令的地址。

```
-T〈CR〉
C0Z0M0E0I0 A=00 B=0000 D=2000 H=1000 SP=0100 PC=0106 MVI B, 64H *0106
-T6〈CR〉
C0Z0M0E0I0 A=00 B=6400 D=2000 H=1000 SP=0100 PC=0108 MOV A, M
C0Z0M0E0I0 A=56 B=6400 D=2000 H=1000 SP=0100 PC=0109 STAX D
```

```
C0Z0M0E0I0 A=56 B=6400 D=2000 H=1000 SP=0100 PC=010A INX H
C0Z0M0E0I0 A=56 B=6400 D=2000 H=1001 SP=0100 PC=010B INXD
C0Z0M0E0I0 A=56 B=6400 D=2001 H=1001 SP=0100 PC=010C DCRB
C0Z0M1E0I0 A=56 B=6300 D=2001 H=1001 SP=0100 PC=010D JNZ 0108 *010D
-
```

从以上的运行情况来看，程序执行是正常的。若需要进一步验证从1000H单元取的数是否已传送至2000H单元，则可以在程序停下来时，用存贮器检查命令进行检查。

若要检验上述程序能否实现循环，则可

```
-T〈CR〉
  C0Z0M1E0I0 A=56 B=6300 D=2001 H=1001 SP=0100 PC=0108 MOV A, M *0108
-
```

以上我们简略地介绍了一些DDT的命令，供使用CP/M操作系统而没有DEBUG程序的情况下，来调试我们的实验程序。

附录5 常用集成电路引脚介绍

在我们的实验中，除了Z80-CTC，Z80-PIO，Intel 8251等以外，还用到一些中小规模集成电路。另外，将来在工作中，当把微型机与客观实际相接口时，除了通用接口片子以外，也还要用到一些中小规模集成电路。为了使用方便，又不太多地增加篇幅，我们把国外最通用的74系统的一些最常用的集成电路的引脚图及其逻辑功能，作一些简要的介绍。

TTL集成电路有五个典型的系列，附表-1列出了这些系列的典型特性。

附表-1

系列	门			触发器
	速度·功耗乘积	传输延迟时间	功耗	时钟输入频率范围
54LS/74LS	19pj	9.5ns	2 mW	dc至45MHz
54L/74L	33pj	33ns	1 mW	dc至3 MHz
54S/74S	57pj	3 ns	19mW	dc至125MHz
54/74	100pj	10ns	10mW	dc至35MHz
54H/74H	132pj	6 ns	22mW	dc至50MHz

特点：

- 由于完全相容，因此可在五种不同的系列中进行选择。
- 每一系列有各种功能可供选择。
- 所有高性能电路都有二极管钳位输入。
- 一般不需要用接头可控阻抗线。
- 输出阻抗低，因此抗交流噪声能力强，驱动电容性负载能力强。
- 所有系列都用5伏电源。
- 所有系列典型直流噪声容限都≥1伏。
- 功耗不受工作频率的影响。
- 在直流满载下可保证开关时间。

·与诸如 DTL、MOS、CMOS 等大多数逻辑系列相容。

下面我们介绍几个在使用中的共同问题。

1．绝对最大额定值

附表-2 列出 5 个系列的绝对最大额定值（除另有注明者外，均指在工作环境温度范围下）

附表-2

		54系列 74系列	54系列 54H系列 74系列 74H系列	54L系列 74L系列	54LS系列 74LS系列 二极管输入	54LS系列 74LS系列 发射极输入	54S系列 74S系列	单位
电源电压，V_{cc}			7	8	7	7	7	V
输入电压			5.5	5.5	7	5.5	5.5	V
发射极间电压（注 1）			5.5	5.5		5.5	5.5	V
小规模集成电路输出端的关态（高电平）电压（注 2）	06，07		30					V
	16,17,26		15					
	其他			8	7	7	7	
禁止三态输出端的高电平电压			V_{cc}		7	7	V_{cc}	V
工作环境温度范围	54系列				−55至	125		℃
	74系列				0 至70			℃
存贮温度范围					65至150			℃

注 1：这是多发射极管两个发射极间的电压，它加于功能方框图中与门或者与非门的直接输入端之间。

注 2：中规模电路的额定值在各相应的数据表中给出。

2．正与门/与非门不用的输入端的处理

为了获得最快的开关速度和最好的抗干扰能力，与门和与非门不用的输入端的电压应高于 V_{OH} 最小值，但又不超过绝对最大额定值。这样可以消除浮置输入端、内引线和外引线的分布电容，从而保证传输延迟时间不增长。处理不用的输入端的一些可行方法如下：

（1）将不用输入端接一独立电源，电压最好在 V_{OH} 最小值和 4.5V 之间。有二极管钳住输入的 54LS/74LS 系列可直接接 V_{cc}。

（2）如果不超过驱动输出端的最大驱动能力，也可将不用的输入端接使用的输入端。在高电平时，增加的输入将使驱动输出为满载，而在低电平时却不增加负载。

（3）将不用的输入端通过一个 1 kΩ 电阻，接 V_{cc}，这时，如果瞬间超过输入最大额定值，则阻抗高到足以保护输入。1 到 25 个不用的输入端均可各接一个 1 kΩ 电阻。有二极管钳住输入的 54LS/74LS 系列可直接接 V_{cc}。

（4）也可将不用的输入端接任何相容的固定高电平输出端，比如接输入端接地的倒相器或与非门的输出端，但不能超过输出端的最大高电平驱动能力。

3．输入电流的要求

输入电流的要求，反映了在推荐工作环境温度下和规定的 V_{cc} 范围内的最坏工作条件。附表-3 列出了各 TTL 系列在标准负载下的最大输入电流和常规基极电阻值。标准

负载的定义是将一个输入端接一个发射极或一个二极管再接上一个其值如附表－3 所示的升压电阻。不过，有些输入端上不止接一个输入三极管（或二极管），即是说，为了降低输入电流或改善性能，也可改变某些输入端的基极电阻值。因而，输入电流的要求可以改变。根据电路的具体型号查阅电性能表，便可确定每个输入端的输入电流要求。

附表-3　标准输入（一个负载）

系　　列	输入升压电阻的常规值	高电平最大输入电流	低电平最大输入电流
54/74	4 kΩ	40μA	－1.6mA
54H/74H	2.8kΩ	50μA	－ 2 mA
54L/74L*	40kΩ	10μA	－ 0.18mA
	8 kΩ	20μA	－ 0.8mA
54LS/74LS	18kΩ	20μA	－ 0.4mA
54S/74S	2.8kΩ	50μA	－ 2 mA

*　如表中所示，54L/74L有两种不同的标准输入。

因为低电平输入电流，基本上是输入基极电阻的函数，所以同一与非门（或与门）的两个或两个以上的输入端可以连在一起，在低电平时仍可看成是一个负载，而在高电平时，多一个不用的输入端就多一个负载。

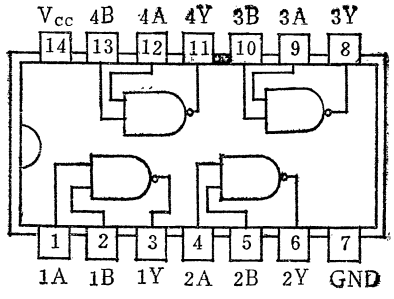
规定流入输入端的电流为正值。

4．驱动能力

推荐工作条件下的 I_{OL} 最大值表示输出端在低电平下从数个负载吸收电流的能力；而 I_{OH} 最大值表示输出端在高电平下提供电流的能力。每个标准输出在低电平下可以吸收本系列 10 个标准负载（74L 和 74LS 系列为 20 个标准负载）的电流。在高电平下可以向本系列 10 个或 20 个负载提供电流。当高电平扇出为 20 时，则可将与非门或者与门的多达 10 个不用的输入端接到该门的使用输入端（输入电流要求如前所述），而不会超出输出驱动 10 个有用输入端的扇出能力。有些输出端是为特殊应用而设计的，它们具有较大或较小的驱动能力，具体的见各种型号的推荐工作条件中的规定。

各负载可按希望的任何组合方式进行连接，驱动电路尤其如此，只要 I_{IH} 和 I_{IL} 的总负载分别小于 I_{OH} 和 I_{OL} 的最大推荐值即可。

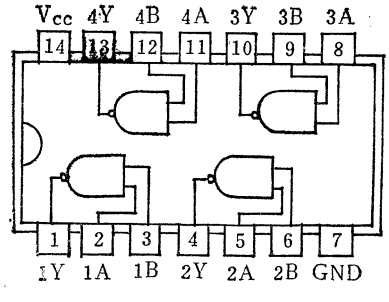
下面我们从与计算机接口的角度，把一些常用的集成电路的型号、引脚图、逻辑功能作一些介绍：



74LS00　2输入四正与非门

$Y = \overline{AB}$

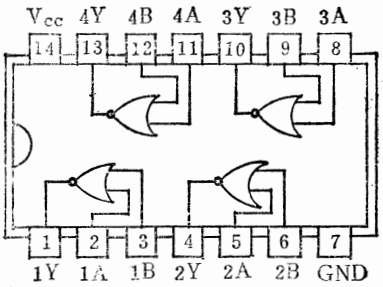
图附 1　74LS00引脚图



74LS01　2输入四与非门（OC）

$Y = \overline{AB}$

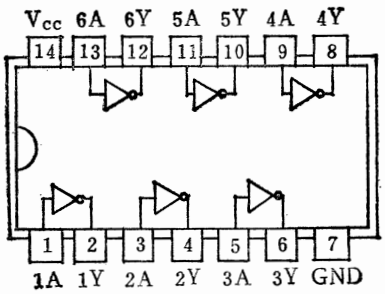
图附 2　74LS01引脚图



74LS02　2输入四正或非门

$Y = \overline{A+B}$

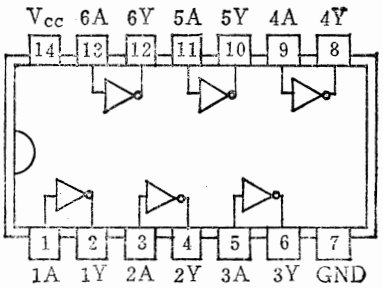
图附 3　74LS02引脚图



74LS04　六倒相器

$Y = \overline{A}$

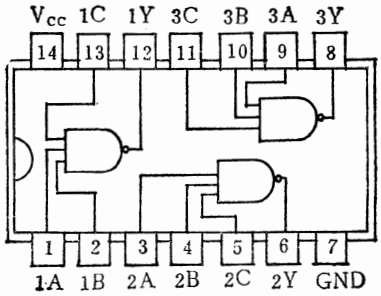
图附 4　74LS04引脚图



74LS05　六倒相器 (OC)

$Y = \overline{A}$

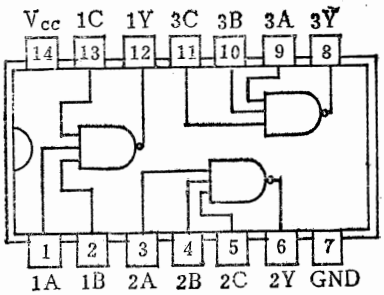
图附 5　74LS05引脚图



74LS10　3输入三正与非门

$Y = \overline{ABC}$

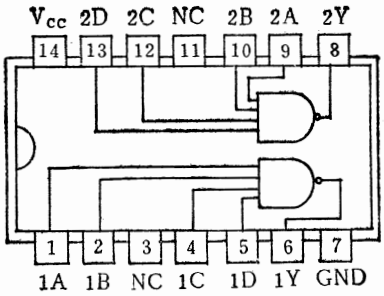
图附 6　74LS10引脚图



74LS12　3输入三正与非门 (OC)

$Y = \overline{ABC}$

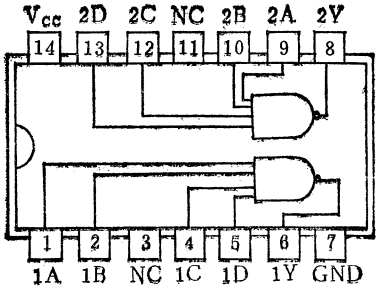
图附 7　74LS12引脚图



74LS20　4输入双正与非门

$Y = \overline{ABCD}$

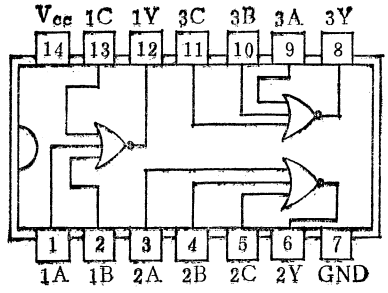
图附 8　74LS20引脚图



74LS22　4输入双正与非门 (OC)

$Y=\overline{ABCD}$

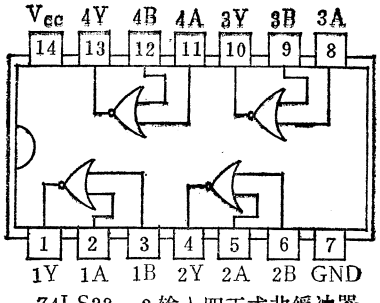
图附 9　74LS22引脚图



74LS27　3输入三正或非门

$Y=\overline{A+B+C}$

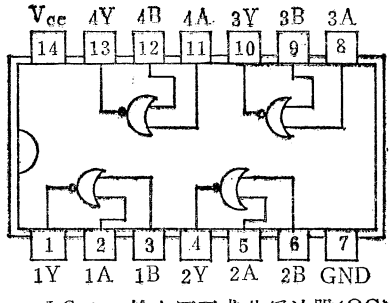
图附10　74LS引脚图



74LS28　2输入四正或非缓冲器

$Y=\overline{A+B}$

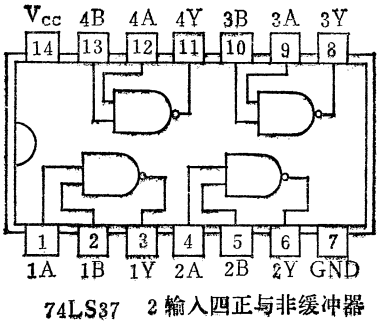
图附11　74LS引脚图



74LS33 2输入四正或非缓冲器(OC)

$Y=\overline{A+B}$

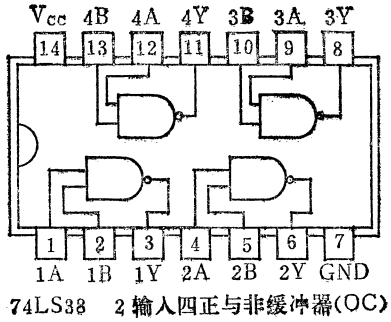
图附12　74LS33引脚图



74LS37　2输入四正与非缓冲器

$Y=\overline{AB}$

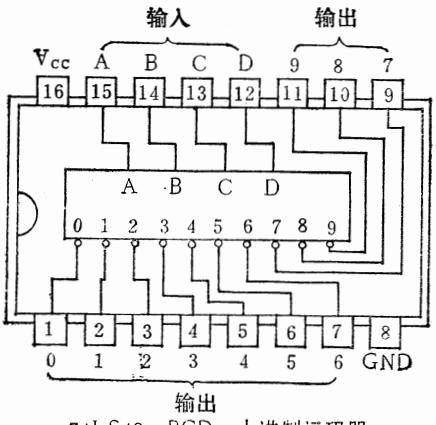
图附13　74LS37引脚图



74LS38　2输入四正与非缓冲器(OC)

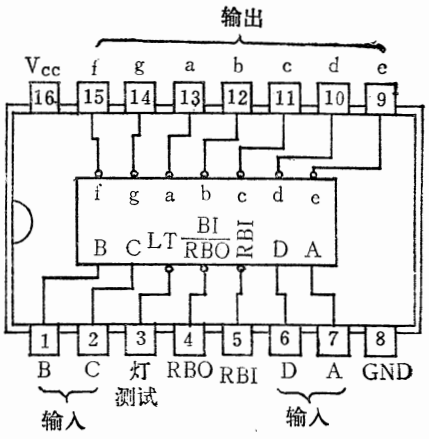
$Y=\overline{AB}$

图附14　74LS38引脚图



74LS42 BCD—十进制译码器

图附15 74LS42引脚图



74LS47 BCD—7段译码器/驱动器

有效低,OC 输出,15V输出

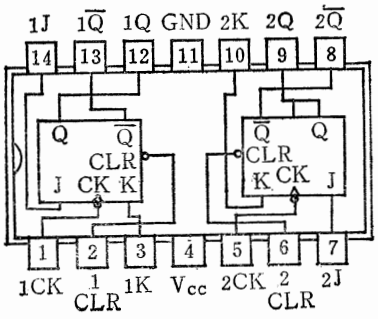
图附16 74LS47引脚图

注：1．要求显示0—9时，灭灯输入（BI）必须开路或保持高电平。

2．将一低电平直接加于灭灯输入（BI）时，则不管其他输入为何电平，所有各段输出都关闭。

3．当动态灭灯输入（RBI）和A、B、C、D输入为低电平，而试灯输入为高电平时，所有各段输出都关闭。

4．当灭灯输入/动态灭灯输出（BI/RBO）开路或保持高电平，而试灯输入为低电平时，则所有各段输出都接通。

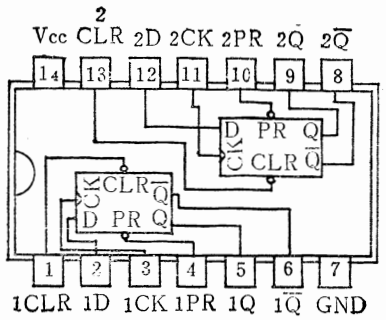


真值表

输入				输出	
CLR	CK	J	K	Q	$\overline{Q}$
L	X	X	X	L	H
H	↓	L	L	Q_0	$\overline{Q}_0$
H	↓	H	L	H	L
H	↓	L	H	L	H
H	↓	H	H	触发	
H	H	X	X	Q_0	$\overline{Q}_0$

74LS73 双J—K触发器

图附17 74LS73引脚图



真值表

输入				输出	
PR	CLR	CK	D	Q	$\overline{Q}$
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	不确定	
H	H	↑	H	H	L
H	H	↑	L	L	H
H	H	L	X	Q_0	$\overline{Q}_0$

74LS74 正沿触发双D型触发器

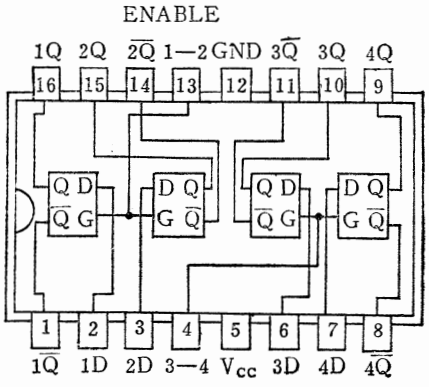
图附18 74LS74引脚图

74LS75

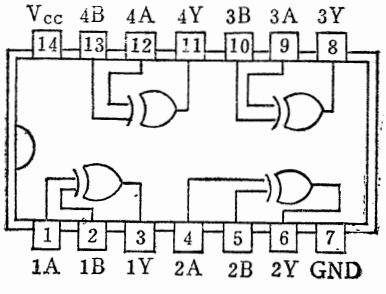
真值表

输入		输出	
D	G	Q	$\overline{Q}$
L	H	L	H
H	H	H	L
X	L	Q_0	$\overline{Q_0}$

Q_0 = G从高过渡到低前Q的电平



图附19　74LS75引脚图

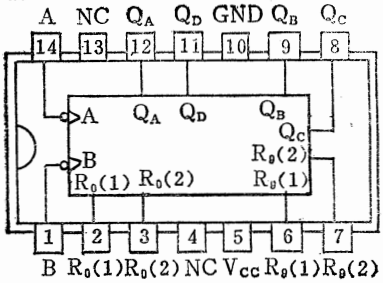


74LS86　2输入四异或门

$Y = A \oplus B$

图附20　74LS86引脚图

输入



输入

74LS90　十进制计数器

复位/计数功能表

复位输入				输出			
$R_0(1)$	$R_0(2)$	$R_9(1)$	$R_9(2)$	Q_D	Q_C	Q_B	Q_A
1	1	0	×	0	0	0	0
1	1	×	0	0	0	0	0
×	×	1	1	1	0	0	1
×	0	×	0	计数			
0	×	0	×	计数			
0	×	×	0	计数			
×	0	0	×	计数			

BCD计数时序（注A）

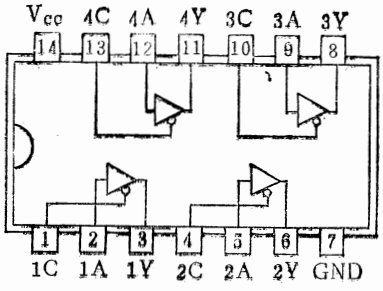
计数	输出			
	Q_D	Q_C	Q_B	Q_A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

二、五混合进制（注B）

计数	输出			
	Q_D	Q_C	Q_B	Q_A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	1	0	0	0
6	1	0	0	1
7	1	0	1	0
8	1	0	1	1
9	1	1	0	0

注：A．输出Q_A与输入B相接作BCD计数　B．输出Q_D与输入A相接作2-5混合进制计数

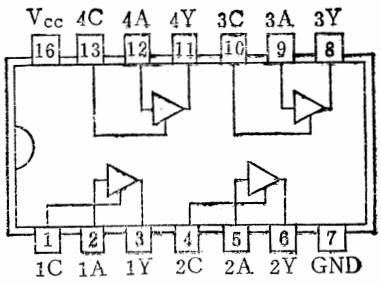
图附21　74LS90引脚图及逻辑



75LS125 四总线缓冲器（三态输出）

Y = A，C为高电平时输出断开

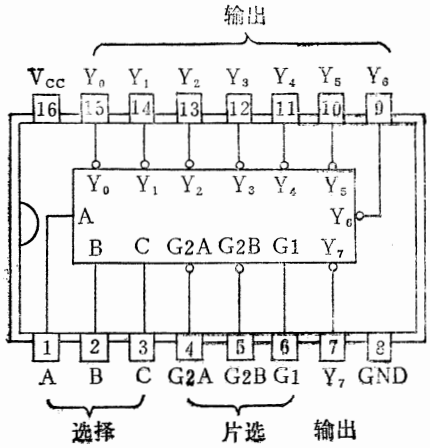
图附22 74LS125引脚图



74LS126 四总线缓冲器（三态输出）

Y = A，C为低电平时输出断开

图附23 74LS126引脚图



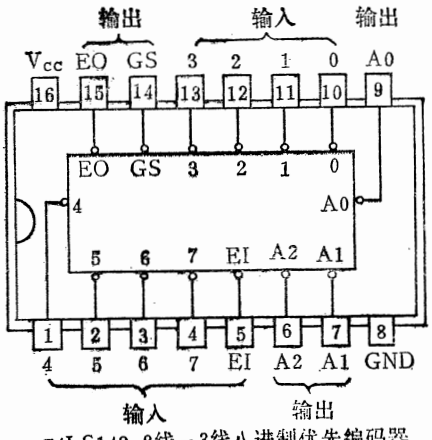
74LS138 3—8线译码器

真值表

输入					输出							
片选		选择										
G_1	G_2^*	C	B	A	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
×	1	×	×	×	1	1	1	1	1	1	1	1
0	×	×	×	×	1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1	1	1
1	0	1	0	0	1	1	1	1	0	1	1	1
1	0	1	0	1	1	1	1	1	1	0	1	1
1	0	1	1	0	1	1	1	1	1	1	0	1
1	0	1	1	1	1	1	1	1	1	1	1	0

* $G2 = G2A + G2B$

图附24 74LS138 引脚图及逻辑

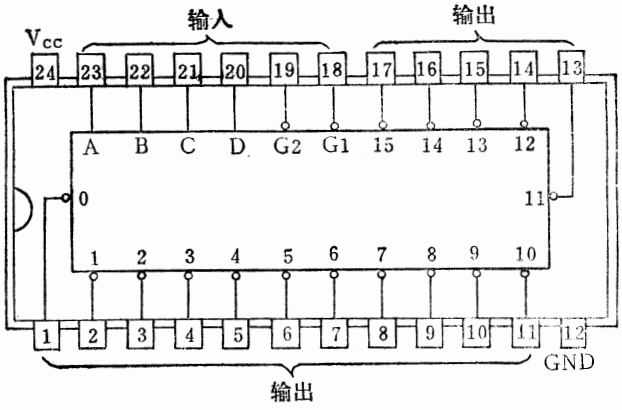


74LS148 8线－3线八进制优先编码器

真值表

输入									输出				
EI	0	1	2	3	4	5	6	7	A2	A1	A0	GS	EO
1	×	×	×	×	×	×	×	×	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	×	×	×	×	×	×	×	0	0	0	0	0	1
0	×	×	×	×	×	×	0	1	0	0	1	0	1
0	×	×	×	×	×	0	1	1	0	1	0	0	1
0	×	×	×	×	0	1	1	1	0	1	1	0	1
0	×	×	×	0	1	1	1	1	1	0	0	0	1
0	×	×	0	1	1	1	1	1	1	0	1	0	1
0	×	0	1	1	1	1	1	1	1	1	0	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1

图附25 74LS148引脚图及逻辑

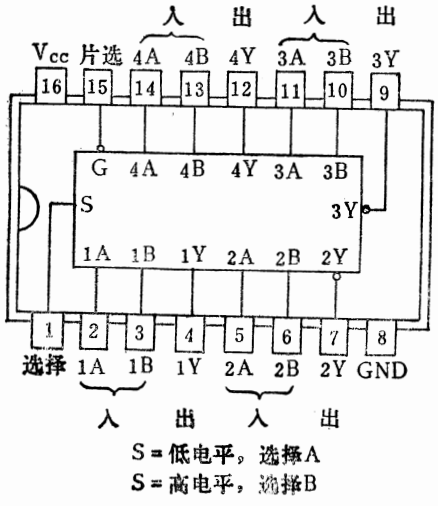


74LS154　4线－16线译码器

真值表

输入					输出														
G_1	G_2	D	C	B	A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	0	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
0	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
0	0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

图附26　74LS154引脚图及逻辑



四2选1数据选择器

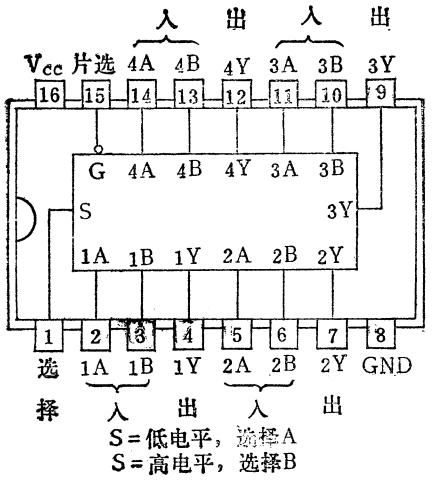
74LS157(原码输出)

74LS158(反码输出)

真值表

输入				输出	
片选	选择	A	B	157	158
1	×	×	×	0	1
0	0	0	×	0	1
0	0	1	×	1	0
0	1	×	0	0	1
0	1	×	1	1	0

图附27　74LS157与74LS158引脚图及逻辑



四2选1数据选择器

74LS257（原码三态输出）

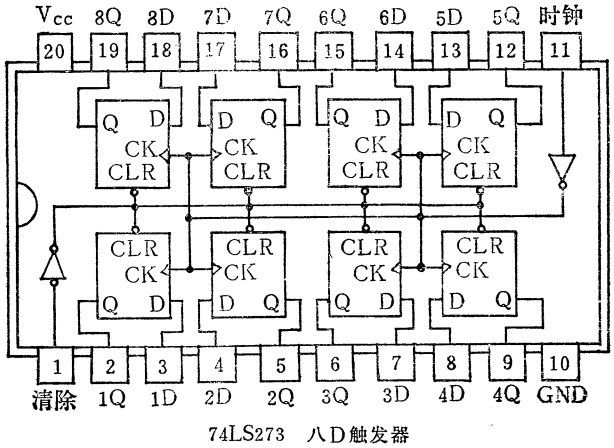
74LS258（反码三态输出）

真值表

输入				输出	
片选	选择	A	B	257	258
1	×	×	×	Z*	Z*
0	0	0	×	0	1
0	0	1	×	1	0
0	1	×	0	0	1
0	1	×	1	1	0

* Z为输出高阻

图附28　74LS257与74LS258引脚图及逻辑

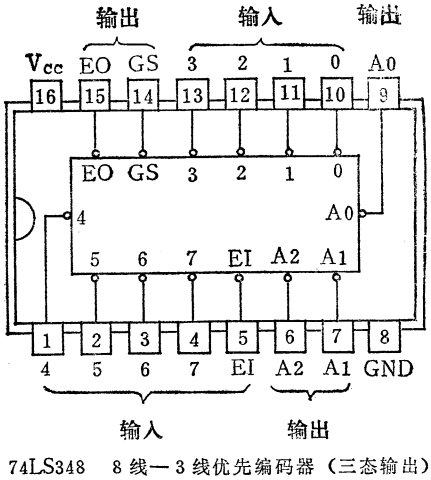


74LS273　八D触发器

公共时钟

平边输出

图附29　74LS273引脚图

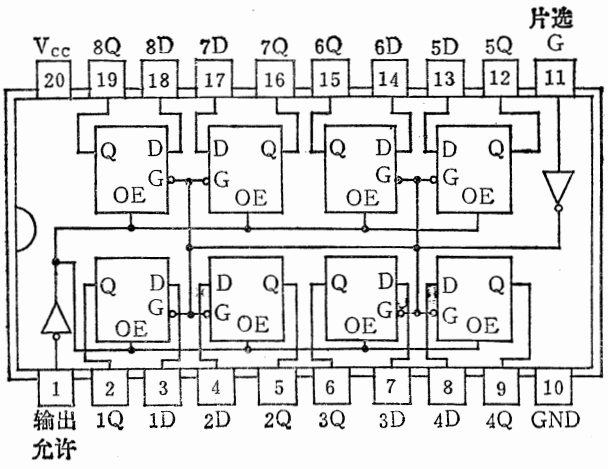


74LS348　8线—3线优先编码器（三态输出）

真值表

输入									输出				
EI	0	1	2	3	4	5	6	7	A2	A1	A0	GS	EO
1	×	×	×	×	×	×	×	×	Z	Z	Z	1	1
0	1	1	1	1	1	1	1	1	Z	Z	Z	1	0
0	×	×	×	×	×	×	×	0	0	0	0	0	1
0	×	×	×	×	×	×	0	1	0	0	1	0	1
0	×	×	×	×	×	0	1	1	0	1	0	0	1
0	×	×	×	×	0	1	1	1	0	1	1	0	1
0	×	×	×	0	1	1	1	1	1	0	0	0	1
0	×	×	0	1	1	1	1	1	1	0	1	0	1
0	×	0	1	1	1	1	1	1	1	1	0	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1

图附30　74LS348引脚图及逻辑

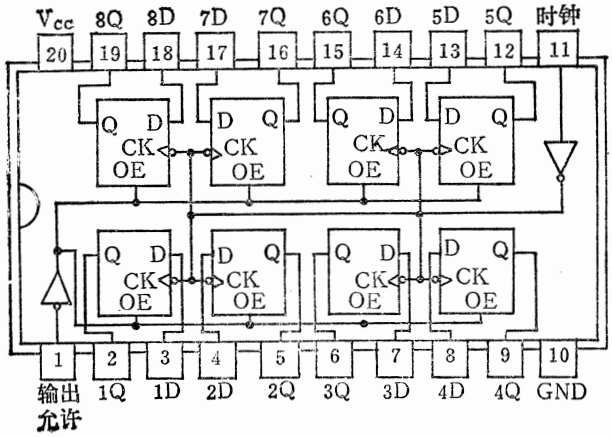


74LS363　八D锁存器三态输出

真值表

输出允许 OE	片选 G	D	输出
0	1	1	1
0	1	0	0
0	0	×	Q_0
1	×	×	Z

图附31　74LS363 引脚图及逻辑

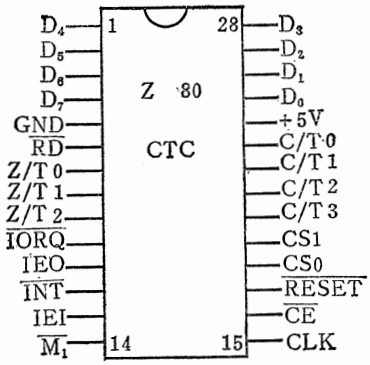


74LS364　八D触发器三态输出

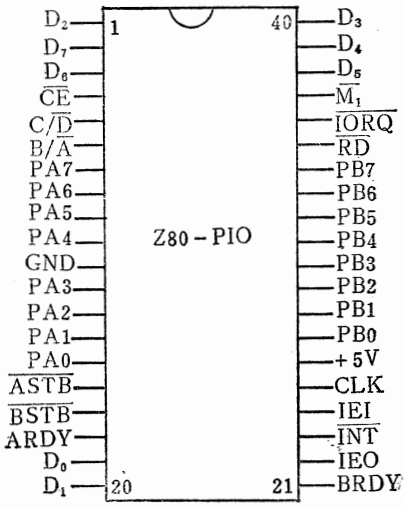
真值表

输出允许 OE	时钟 CK	D	输出
0	↑	1	1
0	↑	0	0
0	0	×	Q_0
1	×	×	Z

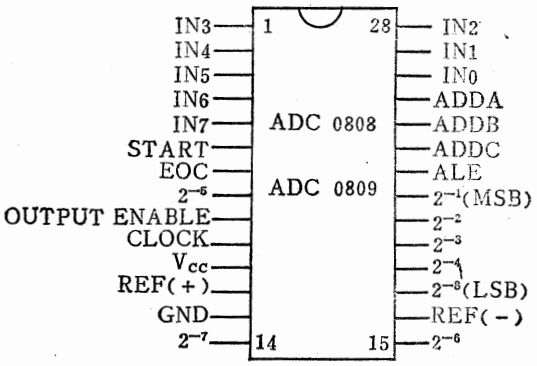
图附32　74LS364引脚图及逻辑



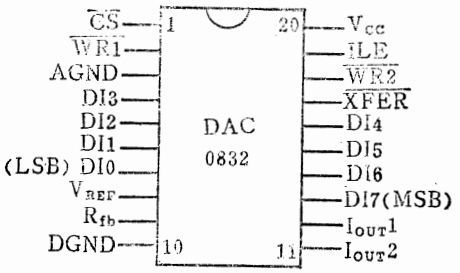
图附33　Z80-CTC引脚图



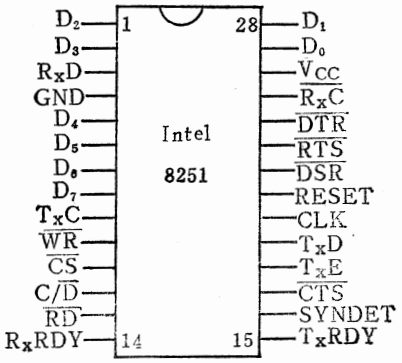
图附34　Z80-PIO引脚图



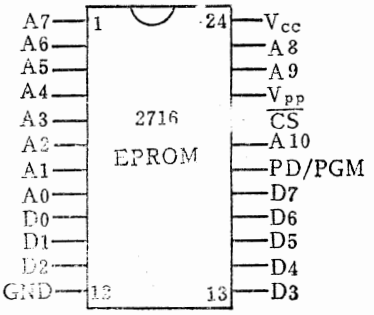
图附35　ADC0809引脚图



图附36　DAC0832引脚图



图附37　Intel 8251A引脚图



图附38　2716引脚图