

微机计算机原理及其应用

微机计算机原理及其应用

实验指导

wei ji suan
ji yuan li
ji qi ying
yong

3620

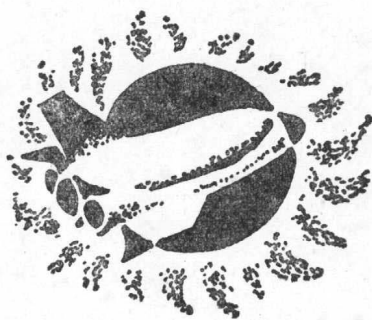
编著

湖北科学技术出版社

微计算机原理及其应用

实验指导

鄢定明 编著



湖北科学技术出版社

《微计算机原理及其应用》实验指导

鄢定明 编著

•

湖北科学技术出版社出版 湖北省新华书店发行

787×1092 毫米 16 开本 7 印张 173,000 字

1984 年 11 月第 1 版 1984 年 11 月第 1 次印刷

印数: 1—50,000

统一书号: 15304·44 定价: 1.40 元

编写说明

这本实验指导书是为《微计算机原理及其应用》配套而写的。全书分为三部分，第一部分介绍了Z80单板计算机的使用方法，主要讲操作。第二部分是编程实验，配合汇编语言程序设计的讲授。第三部分介绍Z80单板计算机的接口实验。编写顺序与《原理》书基本一致。

编写的指导思想是力求简明、实用。

学习微计算机常常感到入门难。初学者在诸多书籍、材料面前不知所措。我们编写这本指导书时，尽量做到删繁就简，突出基本概念、基本用法。每一个实验大体上只突出一个中心，说明一、两件事情，把可有可无的枝节尽量去掉，把难点分散，使内容由浅入深，脉络分明。

作为配套书，本书不再过多交待原理，只突出如何编程，如何使用。为了便于理解，大多数实验都配上了实际数据，尽可能使每个实验自成一体，有头有尾。相当一部分程序都是从教学、科研的实用程序中提取的，读者很容易从这些实例中获得实用的知识。有些程序段稍加修改即可作为应用程序的一部分。

书中备有Z80单板计算机基本知识的摘要，还补充了一些常用的数据和表格，以便查阅。为了便于将助记符形式的源程序译成机器码，特别将“按字母顺序排列的Z-80指令表”附在书后。

微计算机是一门实践性很强的学科，学好的关键在于动手。只要认真学好《原理》课，并且亲自动手编程，亲自动手做实验，就不难达到预期的目的。

本书由武汉邮电科学研究院计算机应用研究室鄢定明同志编写，并请李儒流、林兵两位同志审阅了全书，最后又请华中工学院王飞龙副教授审阅了全书。

本书在编写、出版的过程中得到了湖北省通信学会、湖北省科学技术出版社、空军雷达学院的大力支持，在此表示诚挚的谢意。

一九八四年八月

目 录

第一章 Z80单板计算机的使用方法.....	1
第一节 Z80单板计算机摘要.....	1
第二节 怎样使用单板计算机.....	8
第二章 编程实验.....	20
第一节 简单程序.....	20
第二节 分支与循环.....	24
第三节 堆栈和子程序.....	29
第四节 算术运算程序.....	34
第五节 代码转换.....	48
第六节 查表.....	57
第三章 单板计算机接口实验	59
第一节 Z80 CTC接口实验.....	59
第二节 Z80 PIO接口实验.....	64
第三节 为单板机增加接口芯片.....	71
第四节 显示器的实验.....	76
第五节 键盘的实验.....	81
附录: 按字母顺序排列的Z-80指令表.....	87

第一章 Z80单板计算机的使用方法

第一节 Z80单板计算机摘要

在《微计算机原理及其应用》一书中，已经详细介绍了Z80单板计算机的结构组成、工作原理以及实际使用的方法。为了方便起见，现将单板计算机应用中经常用到的内存地址分配、接口地址分配、可编程接口芯片的各种控制字的约定等，汇总摘要在这里，以备查阅。

一、存储器地址分配：

1. 地址空间译码器

采用一片74LS138作为地址空间译码器，八根输出线全部用完可组成16K字节存储容量。用MREQ参予译码，目的在于与访问接口相区别。

2. TP801的具体地址分配

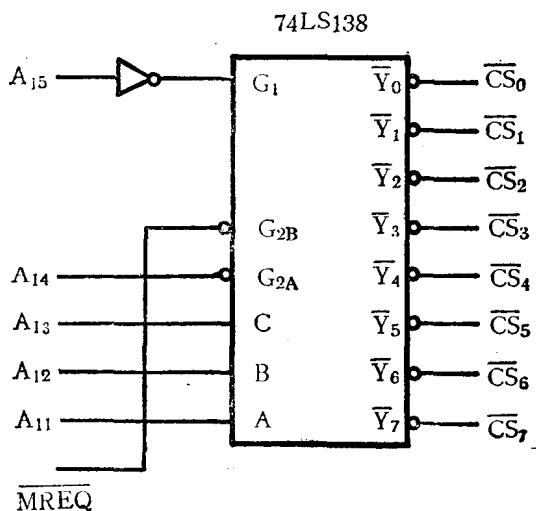


图 1.1

译码器的有效输出	对应的地址空间	配用的器件
$\overline{Y}_7 = \overline{CS}_7$	3800~3FFFH	没用
$\overline{Y}_6 = \overline{CS}_6$	3000~37FFH	没用
$\overline{Y}_5 = \overline{CS}_5 = \overline{RAM}_2SEL$	2800~2FFFH	$U_{20} \sim U_{23}$ (2KRAM)
$\overline{Y}_4 = \overline{CS}_4 = \overline{RAM}_1SEL$	2000~27FFH	$U_{16} \sim U_{19}$ (2KRAM)
$\overline{Y}_3 = \overline{CS}_3$	1800~1FFFH	没用
$\overline{Y}_2 = \overline{CS}_2 = \overline{PROM}_2SEL$	1000~17FFH	U_9 (2KEPROM)
$\overline{Y}_1 = \overline{CS}_1 = \overline{PROM}_1SEL$	0800~0FFFH	U_8 (2KEPROM)
$\overline{Y}_0 = \overline{CS}_0 = \overline{MONSEL}$	0000~07FFH	U_7 (2KEPROM)

3. TP801监控程序所占用的RAM常数区为：2F90~2FFFH，用户程序不得侵占。不同的单板机内存分配可能不一样，所占用的RAM常数区也可能不一样，使用时一定要搞清楚。

4. 可供扩充的译码线

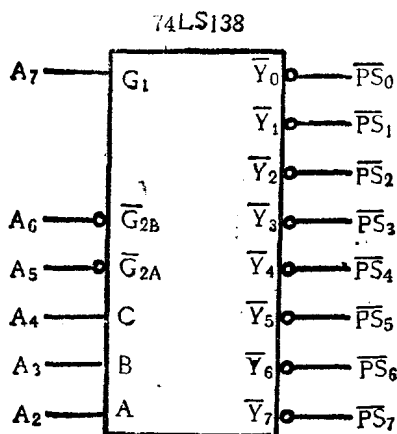


图 1.2

这里有三条地址译码线($\overline{CS_3}$ 、 $\overline{CS_6}$ 、 $\overline{CS_7}$)尚未用完, 用户可以用来扩充内存。

二、接口地址分配:

1. 接口地址译码器

采用一片74LS138作为接口地址译码, 但与内存译码不同的是: 没有将 $\overline{IORQ}$ 直接加到它的输入端或控制端。在利用它的输出信号时, 应注意到这一点区别, 以便正确识别接口。

2. 具体的接口地址分配如下:

译码器输入信号		所代表的口地址	译码器的输出信号	选中的接口芯片
$A_7 \sim A_2$	$A_1 A_0$			
100000	0 0	80H	$\overline{Y_0} = \overline{PS_0} = \overline{PIOSEL}$	PIO 口A数据寄存器
	0 1	81H		PIO 口B数据寄存器
	1 0	82H		PIO 口A命令寄存器
	1 1	83H		PIO 口B命令寄存器
100001	0 0	84H	$\overline{Y_1} = \overline{PS_1} = \overline{CTCSEL}$	CTC 0 通道
	0 1	85H		CTC 1 通道
	1 0	86H		CTC 2 通道
	1 1	87H		CTC 3 通道
100010	× ×	88~8BH	$\overline{Y_2} = \overline{PS_2} = \overline{SEGLH}$	74LS273, 七段选择
100011	× ×	8C~8FH	$\overline{Y_3} = \overline{PS_3} = \overline{DIGLH}$	74LS273, 数位选择
100100	× ×	90~93H	$\overline{Y_4} = \overline{PS_4} = \overline{KBSEL}$	74LS244, 读键值
100101	× ×	94~97H	$\overline{Y_5} = \overline{PS_5}$	未用
100110	× ×	98~9BH	$\overline{Y_6} = \overline{PS_6}$	未用
100111	× ×	9C~9FH	$\overline{Y_7} = \overline{PS_7}$	未用

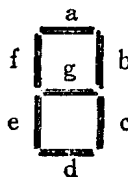
3. 可供扩充的接口译码线

其中 $\overline{PS_5}$ 、 $\overline{PS_6}$ 、 $\overline{PS_7}$ 都未用到, 利用这些信号, 再加上 A_1 与 A_0 可以区分12个接口地址, 可用于接口扩充。

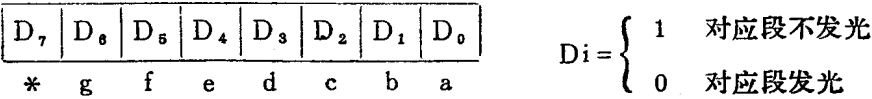
三、七段发光管显示器:

1. 段选码的概念

每个七段发光管显示器由a、b、c、d、e、f、g、七段组成, 它们依次排列成“日”字形。



可以用一种“段选码”来控制不同段的发光与否，以便获得不同的显示字符。“段选码”与各段的对应关系如下：



2. 单板机中常用的几种字符所对应的“段选码”：

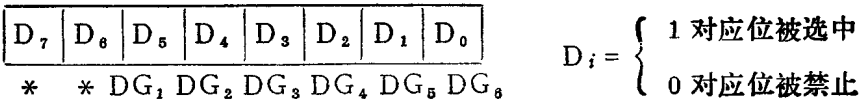
显示的字符	0	1	2	3	4	5	6	7	8	9	A	B	G	D	E	F	空格	-
相应的段选码(H)	40	79	24	30	19	12	02	78	00	18	08	03	46	21	06	0E	7E	3F

3. 位选码的概念：

单板机共有六个七段发光管，常用左面的四位数代表十六进制，地址右边的两位代表指令码或十六进制数据。



可以用一种“位选码”来选择不同的显示位。“位选码”与不同显示位的对应关系如下：



4. 改变七段显示字符的基本要点：

- { 段选码 → (88H) 口
- { 位选码 → (8CH) 口

四、键盘：

1. 键盘的行码与列码的概念

单板机用的简易键盘共有28个键：12个命令键和16个数字键。它们排列在一个6行×5列的方阵中。

第六行→	PROG	LOAD	DUMP	BP	
第五行→	MEM	PORT	REG	REG'	
第四行→	7	8	9	A	NEXT
第三行→	4	5	6	B	MON
第二行→	1	2	3	C	STEP
第一行→	0	F	E	D	EXEC
	↑	↑	↑	↑	↑
	第五列	第四列	第三列	第二列	第一列

可以用“行码”和“列码”来识别键盘的某一个键。“行码”与各行的对应关系是：

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	
*	*	第六行	第五行	第四行	第三行	第二行	第一行	

$$D_i = \begin{cases} 1 & \text{对应的行开放} \\ 0 & \text{对应的行禁止} \end{cases}$$

“列码”与各列的对应关系是：

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	
*	*	*	第五列	第四列	第三列	第二列	第一列	

$$D_i = \begin{cases} 1 & \text{无键合上} \\ 0 & \text{有键合上} \end{cases}$$

应用逐行扫描读出列码的办法来识别所按下的键。

2. 数字键的用途

(1) 在命令键MEM、PORT、REG、REG'之前输入4个、2个或1个数，表示“地址量”或寄存器代号，被送到显示缓冲区。

用来代表寄存器的数有：

1 (PC)、2 (SP)、3 (IFF)、4 (IX)、5 (IY)、6 (I)、7 (H)、8 (L)

(2) 紧跟在这些命令键之后输入的数字量将作为修改值放入存储器、接口或CPU寄存器映象区中。

3. 命令键的功能和用法将分别在以下几节中提到。

五、CTC的用法：

1. TP801机中CTC与CPU的连接

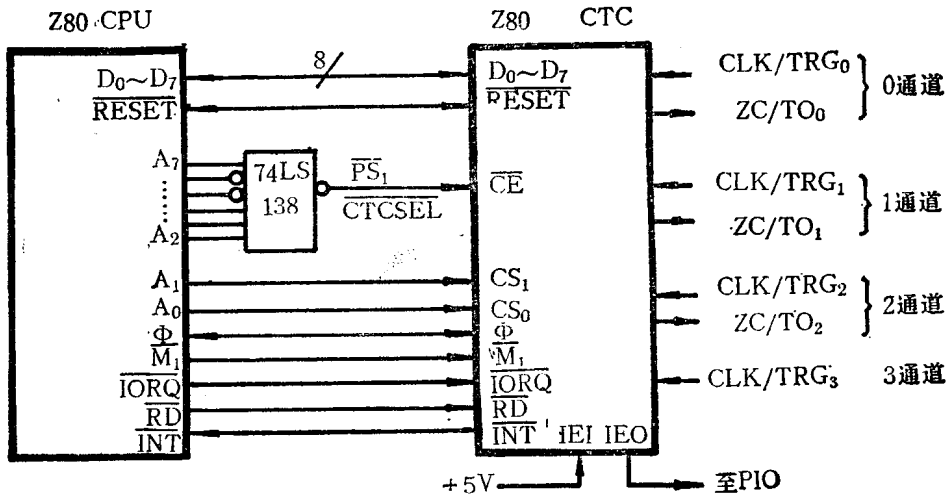


图 1.3

片选和通道选择:

控制信号的作用:

$A_7 A_6 A_5 A_4 A_3 A_2$	$A_1 A_0$	代表的	选中的通道
$\overline{CE}$	$CS_1 CS_0$	接口地址	
100001	0 0	84H	CTC 0
	0 1	85H	CTC 1
	1 0	86H	CTC 2
	1 1	87H	CTC 3
不是100001	× ×		未被选中

$\overline{M}_1$	$\overline{IORQ}$	$\overline{RD}$	功 能
0	0	1	中断响应周期
0	1	0	检查中断程序是否结束
1	0	0	CPU从CTC读出
1	0	1	由CPU向CTC写入

“读出”：减1计数器的内容

“写入”：控制寄存器

2. 命令字的用法

(1) 通道控制字各位的含义:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
-------	-------	-------	-------	-------	-------	-------	-------

“通道控制字”的识别标记，规定 $D_0 = 1$

$D_1 = 1$ ，CTC停止工作；

$D_1 = 0$ ，当减1计数器到零后，自动装入时间常数，重新工作

$D_2 = 1$ ，指明下一个字为时间常数

$D_2 = 0$ ，指明下一个字不是时间常数

$D_3 = 1$ ，定时器由外触发信号启动

$D_3 = 0$ ，定时器由内部触发启动

$D_4 = 1$ ，触发脉冲上升沿有效

$D_4 = 0$ ，触发脉冲下降沿有效

$D_5 = 1$ ，分频系数为256

$D_5 = 0$ ，分频系数为16

$D_6 = 1$ ，选择计数工作方式

$D_6 = 0$ ，选择定时工作方式

$D_7 = 1$ ，允许通道中断

$D_7 = 0$ ，禁止通道中断

(2) 设定时间常数值:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
-------	-------	-------	-------	-------	-------	-------	-------

最高有效位

最低有效位

“时间常数”一定要紧跟在“通道控制字”之后，而且通道控制字的 $D_2 = 1$ 。

在定时方式下“时间常数”值与分频系数一道决定了定时的长短（定时间隔 = 时间常数 × 分频系数 × 0.5 μs）；在计数方式下，它确定了CTC作为计数器的计数值。

“时间常数”值可以是1~255范围内的任何整数，若为全零，则表示时间常数 = 256
在内触发情况下，装入时间常数后，所指通道立即工作。

(3) 设定中断矢量的低8位：

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	
由用户设定								中断矢量的识别标记，规定D ₀ = 0
0	0							CTC 0 通道
0	1							CTC 1 通道
1	0							CTC 2 通道
0	0							CTC 3 通道

不管哪个通道的中断矢量都由84H口送入。

六、PIO的用法：

1. Z80CPU与PIO的连接：

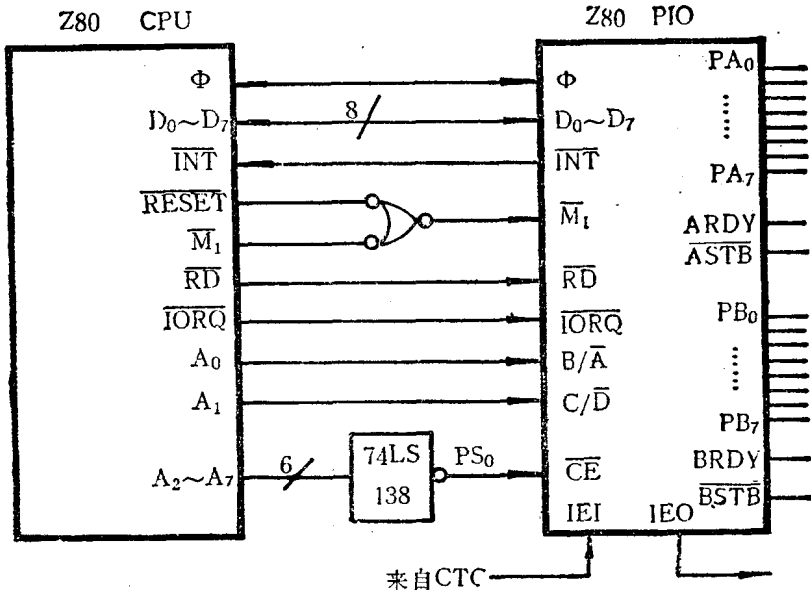


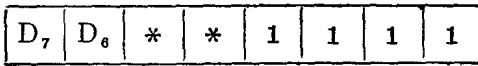
图 1.4

T P801机的片选和通道寄存器选择：

A ₇ A ₆ A ₅ A ₄ A ₃ A ₂	A ₁ A ₀	对应的口地址	选 中 的 单 元
CE	C/D A/B		
100000	0 0	80H	A 口的数据寄存器
	0 1	81H	B 口的数据寄存器
	1 0	82H	A 口的命令寄存器
	1 1	83H	B 口的命令寄存器
不是100000	× ×		芯片未被选中

2. PIO各命令字的用法

(1) 方式的选择字



D₀~D₃为“方式选择字”的识别标记,规定为全1

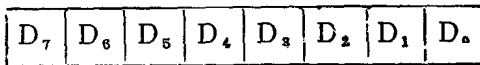
D₄、D₅可以任意设定

- | | | | |
|---|---|---|-----------------|
| 0 | 0 | — | 方式0, 带联络的输出 |
| 0 | 1 | — | 方式1, 带联络的输入 |
| 1 | 0 | — | 方式2, 带联络的双向工作方式 |
| 1 | 1 | — | 方式3, 位控方式 |

(2) 入/出选择字:

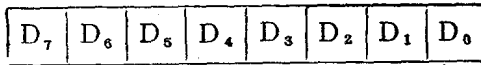
只有在位控方式下才有意义。

此命令字必须紧跟在方式选择字之后。



$$D_i = \begin{cases} 1 & \text{对应的位为输入} \\ 0 & \text{对应的位为输出} \end{cases}$$

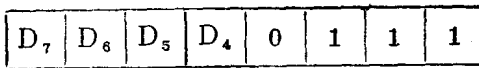
(3) 中断矢量低8位



由用户填入 (任选)

中断矢量的识别标志,规定D₀ = 0

(4) 中断控制字



“中断控制字”的识别标记

D₄ = 1 紧跟在后面的命令字为“中断屏蔽字”

D₅ = 1 受监视的I/O线、高电平请求中断有效

D₅ = 0 受监视的I/O线、低电平请求中断有效

D₆ = 1 受监视的I/O线“与”逻辑有效

D₆ = 0 受监视的I/O线“或”逻辑有效

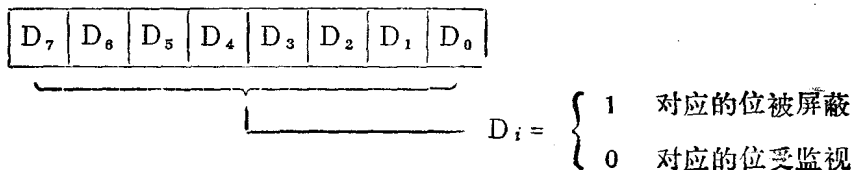
D₇ = 1 允许PIO相应的口, 可以请求中断

D₇ = 0 禁止该口请求中断

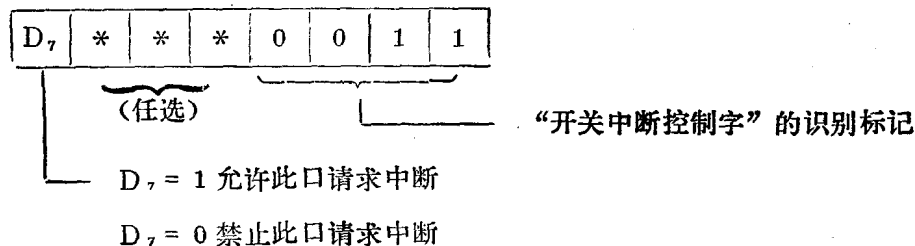
(只有在位控方式下, D₄、D₅、D₆才有意义)。

(5) 中断屏蔽字

一定要紧跟在中断控制字之后（当 $D_4 = 1$ 时），并且只有在“位控方式”下才有意义



(6) 另一种开关中断控制字



第二节 怎样使用单板计算机

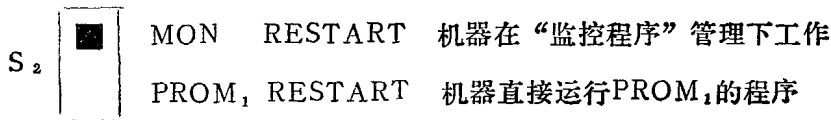
这一节的目的主要是熟悉单板计算机的键盘操作。学习怎样向单板机送程序，怎样检查程序，怎样执行程序以及怎样保存程序。

一、单板计算机的初始状态

1. 在单板机接通电源之前应作的检查

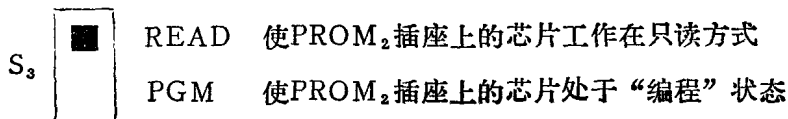
(1) 检查电源是否为 $+5V$ （可以有10%左右的偏差，但偏差不能太大）。电源不能接反！

(2) 检查开关 S_2



在通常情况下， S_2 应倒向“MON RESTART”。

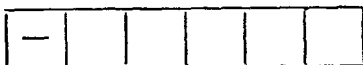
(3) 检查开关 S_3



在通常情况下， S_3 应倒向“READ”。

2. 提示符的作用

电源接通后，显示器上应显示提示符“—”（不同的单板机，提示符可能不一样）。



否则应按一下“S₁”——即“RESET”键。不论何种情况，重新按RESET键都应出现提示符。

反复按RESET键都不出现提示符，首先应复查各开关的初态，尤其是S₂！然后再找其它原因。

3. MON键的功能

按下“MON”键也能使单板计算机恢复初态，显示“提示符”。

用MON键，可以中止现行程序的执行。

用MON键，可以结束前一种命令状态，为新的命令状态作准备。例如：

键盘操作	显示				
<u>2</u> <u>0</u> <u>4</u> <u>1</u> <u>5</u>	2	0	4	1	5
MON	-				

(错误操作) (回到初态)

〔注〕：MON和RESET键虽然都能使单板机恢复初态，但程序返回的入口点并不一致（详见监控程序），因此功能上有些差别，用MON键可以保护用户CPU寄存器的状态和某些RAM专用单元的内容。

二、怎样输入程序

1. 准备工作

(1) 编制用助记符书写的程序，仔细审查，排除错误。

(2) 将源程序译成机器码并进行地址代真。“翻译”工作可以由人工做，也可请高一档的机器自动完成。例如，Cromemco CS—II，CS—III，TRS—80，带Z—80卡的APPLE—II等都有汇编程序，能将Z—80助记符程序译成机器码并进行地址代真，还能指出源程序中的错误。

假定经过“翻译”后，准备送到机内的程序为：

地址(H)	机器码(H)	源程序
		ORG 2000H
2000	8 A	LD A, (2010H)
2001	1 0	
2002	2 0	
2003	2 F	CPL
2004	32	LD (2011H), A
2005	11	
2006	20	
2007	76	HALT

2. 输入过程

将上述源程序输入单板机时的键盘操作如下：

键盘操作	显示		注释
<u>MON</u>	-		回到初态, 作好准备
<u>2 0 0 0</u>	2 0 0 0		给出起始地址
<u>MEM</u>	2 0 0 0	× ×	× × 为2000H 中原先的内容
<u>3 A</u>	2 0 0 0	3 A	将 3 A 写入到2000H 之中
<u>NEXT</u>	2 0 0 1	× ×	自动选中下一单元
<u>1 0</u>	2 0 0 1	1 0	将10写入到2001H之中
<u>NEXT</u> <u>2 0</u>	2 0 0 2	2 0	
<u>NEXT</u> <u>2 F</u>	2 0 0 3	2 F	
<u>NEXT</u> <u>3 2</u>	2 0 0 4	3 2	
<u>NEXT</u> <u>1 1</u>	2 0 0 5	1 1	
<u>NEXT</u> <u>2 0</u>	2 0 0 6	2 0	
<u>NEXT</u> <u>7 6</u>	2 0 0 7	7 6	
<u>MON</u>	-		结束输入状态, 回到初态

三、怎样检查计算机的内容

1. 用MEM键检查存储单元的内容

例如, 为了验证上面的输入是否正确, 可以用MEM键进行检查, 发现错误时可进行修改。

键盘操作	显示		注释
<u>2 0 0 0</u>	2 0 0 0		给出待检查的地址
<u>MEM</u>	2 0 0 0	3 A	正确
<u>NEXT</u>	2 0 0 1	1 0	正确
<u>NEXT</u>	2 0 0 2	0 0	错误! 2002中不应是00!

<u>2 0</u>	2 0 0 2	2 0	将正确的代码送入2002单元
<u>NEXT</u>	2 0 0 3	2 F	正确
.....			
<u>NEXT</u>	2 0 0 7	7 6	全部核对完毕
<u>MON</u>	-		退出。回到初态。

2. 检查寄存器的内容

除了A、B、C、D、E之外，其它寄存器应该用相应的数字代替。例如：

键盘操作	显示		注释
<u>A REG</u>	A	× ×	× × 为A寄存器之中的内容
<u>MON</u>	-		为下一步作准备
<u>B REG</u>	b	× ×	× × 为B寄存器的内容
<u>MON</u>	-		
<u>1 REG</u>	1 × ×	× ×	× × × × 为P C中的内容
<u>2 0 0 0</u>	1 2 0	0 0	强行将2000H写入P C

显示辅助寄存器内容，方法同上，只不过改用REG'键而已。

SP的内容只能被检查，不能修改。

读出的IFF的内容若为0 0 H表示禁止中断，若为0 4 H表示允许中断。

3. 检查或更改接口的内容

先输入两个十六进制数，表示口地址。

再压下PORT EXAM键，就可显示该口的内容，（实际上执行了一次输入指令）。连续按PORT键就会连续显示该口内容。对于CTC来说，按下PORT键将显示出CTC减1计数器的内容。连续操作可了解CTC减1计数器的变化。

在PORT键后，再输入两个数就可以修改该口的内容。（实际上执行了一次输出指令）。

注意，有些口只能写入，有些口只能读出，有些接口包含几个寄存器，用PORT来检查和修改的可能不是同一个寄存器。例如，象CTC这样的接口，“读”（即检查）——减1计数器的内容，“写”（即修改）——控制寄存器。

例如，显示9 0 口的内容可用如下键盘操作

<u>MON</u>	-		准备接受新的命令
<u>9 0</u>	9 0		先送入口地址
PORT	9 0	3 F	检查该口的内容

四、怎样执行程序

假定已将下述程序送入内存指定区域:

```

2000          3 A 1020      LD  A, (2010H)
2003          2 F          CPL
2004          8 2 1120      LD  (2011H), A
2007          76           HALT

```

执行该程序前的准备工作为, 先将任何一个数(如 6 A)送入2010H单元。执行后检查2011单元中的求反值是否正确。

<u>MON</u>	-		
<u>2 0 1 0 MEM 6 A</u>	2 0 1 0	6 A	设定待转换的数
<u>NEXT</u>	2 0 1 1	× ×	检查结果单元, × × 为随机数

可用下列三种方式执行程序:

1. 连续执行(用EXEC键)

只要送入程序首地址, 再按下EXEC键即可。例如:

<u>MON</u>	-		准备执行程序
<u>2 0 0 0</u>	2 0 0 0		先送入程序的起始地址
<u>EXEC</u>			程序停在HALT指令处
<u>MON</u>	-		用MON键退出HALT指令

检查进行结果:

<u>2 0 1 0 MEM</u>	2 0 1 0	6 A	检查原始数据
<u>NEXT</u>	2 0 1 1	9 5	检查结果单元的内容

$6 A = 0110\ 1010 = 10010101 = 95$ 结果正确

2. 单步执行方式(用STEP键)

每次只执行一条指令, 并显示出执行该指令后累加器的内容以及下一条指令的地址码。