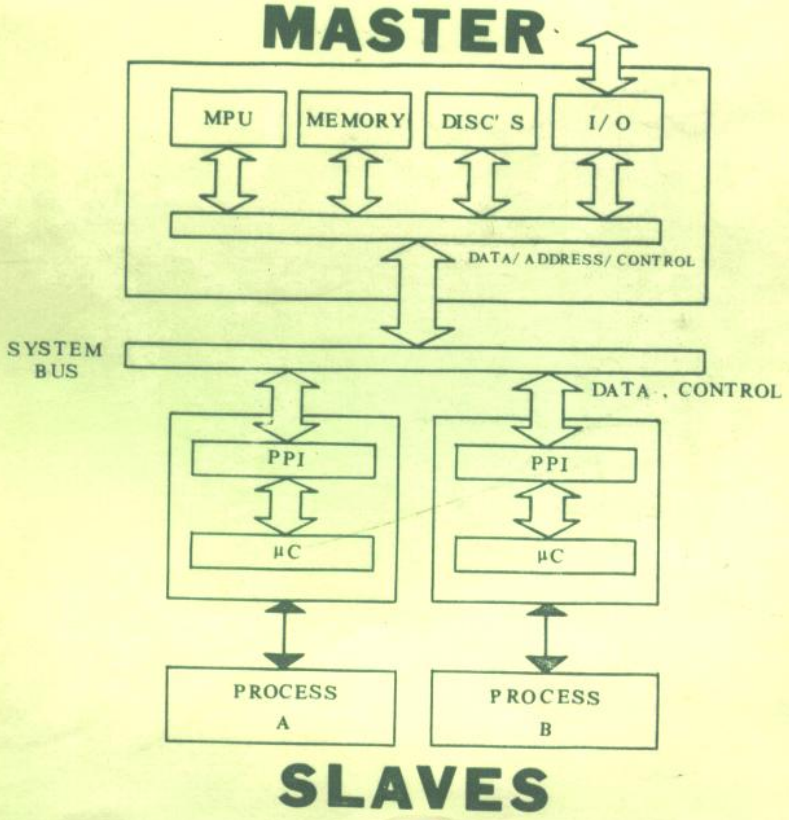
范肇基 编著

EI XING JI SUAN JI KAI FA XI TONG DE YING YONG

微型计算机开发系统的应用



浙江大学出版社

高等学校教学用书

微型计算机开发系统的应用

范肇基 编著

浙江大学出版社

内容简介

本书以美国TNTEL公司的八位微型计算机开发系统——MDS—236为主，讨论微型计算机开发系统在微型计算机及微处理机系统设计中的应用。

微机系统及微处理机系统设计可分为两大部分，即软件设计与硬件系统设计。本书除了叙述MDS—236微机开发系统的原理与结构外，重点叙述如何应用“实时在机仿效器”这一先进设备进行软件与硬件系统的设计。书中有许多实例可帮助读者较快地掌握软件及硬件系统的开发方法。

本书可作为高等学校微机系统设计课程的教科书，亦可作为科技工作者的参考书。

微型计算机开发系统的应用

范肇基编著

责任编辑　应伯根

• • •

浙江大学出版社出版

浙江大学印刷厂印刷

浙江省新华书店发行

• • •

开本787×1092　1/16　印张：7.5　字数：168千字

1986年8月第一版　1986年8月第一次印刷

印数：1—3 000

统一书号：15337·022　定价：1.20元

前　言

微型计算机开发系统是专门用于设计和调试微型计算机的一种系统。从它的设计功能上可以看出，它不仅为软件设计和调试配备了多种方法，而且还能应用于微机及微处理机系统的硬件设计与调试。特别是它具有设计的实时性，目前任何一种辅助设计工具都是不能与它相比拟的。正是由于它具有如此优越的性能，所以得到广大微机系统设计者越来越广泛的应用。

此书的编写目的是为了能有助于推广微型计算机开发系统这一先进工具。在编写过程中，力求使本书既有系统性又有较强的实用性。书中收集了一些实例，目的是帮助读者能较快地理解微型计算机开发系统的工作原理，同时还能更快地掌握它的使用方法。

本书由浙江大学副校长吕维雪教授审稿，浙江大学科学仪器系一些同志对本书也提出了不少宝贵意见。特在此表示深切的感谢。

由于水平有限，差错在所难免。对书中不妥之处，请广大读者批评指正。

编著者

一九八五年四月

目　录

第一章 概 述

§1-1 微型计算机及微处理机系统的特点

随着微电子技术的发展，密集度很高的大规模集成电路的器件不断涌现，给电子计算机的结构带来了很大的变化。

以往的中小型计算机中，代价最高的中央处理部件，在微型计算机及微处理机系统中由于采用了具有同样功能的大规模集成电路片子，不仅价格大大下降，而且体积也大大缩小。其他部件（如输入/输出的控制部件、各种外国设备的接口部件等）亦都日益采用功能与之相适应的大规模集成电路组成的“功能模块”。

功能齐全的各种“功能模块”的出现，使计算机硬件系统设计工作突破了以往中小型计算机设计方法的框框。现在微型计算机及微处理机硬件系统的设计，仅仅是选择一套与任务相适应的“功能模块”而已。通常是以某一系列中央处理器芯片(CPU)为中心，选择一套支援器件，如输入/输出接口、中断控制器、总线控制器、通迅控制接口以及各种外围设备的专用控制器等。

通常所谓的微处理机系统是指专门用于解决特定任务的微型计算机系统。它的基本原理与微型计算机相同，而它的规模一般都比通用的微型计算机小。例如，存贮器容量及附加的外围设备没有微型计算机大或齐全。然而系统的基本结构是一致的，因而两者设计的方法亦大同小异。

微型计算机及微处理机系统的硬件设计仅仅是选择合适的“功能模块”。中小型计算机硬件系统中组装及调试所需的大量时间就可省去，从而大大提高工作效率。

各种“功能模块”给硬件系统的设计带来了极大方便。但是这些“功能模块”的功能选择和操作都是由相应的软件控制的，即通过程序编制和执行而实现的。在微处理机系统的应用中，主要的工作是设计软件去驱动所选择的“功能模块”，开发软件的工作量大大超过开发硬件系统的工作量。这是微型计算机及微处理机系统与中小型计算机的重要区别之一。

由于半导体存贮器的集成度不断提高，微机系统的启动、监控、调试、诊断等系统软件的保存亦就采用与中、小型计算机不同的方法。在中、小型计算机中，通常是采用外存贮设备来引导输入监控程序及操作系统等系统软件；而在微型计算机及微处理机系统中，常常把监控、调试及诊断等系统软件保存在称为只读存贮器(ROM)的固件中，也就是说软件固化在固件中。这又是两者的不同之处。

一般说来，微型计算机或微处理机系统的功能正确与否，性能好与差，只能在专门为它编制的测试程序运行间表现出来。由此可知，微型计算机及微处理机系统在开发过程中，硬件和软件的结合到了非常紧密的程度。

当然，新的软件开发可利用已有的相同类型的计算机系统来进行，但硬件系统的调试却只能借用硬件的测试工具。

根据微型计算机开发过程中“软”与“硬”紧密结合的特点，系统设计者极需要一种能进行软件和硬件综合调试的工具，称之谓微型计算机开发系统。近几年来，微机开发系统发展极其迅速，其性能越来越好，功能越来越强。

本书主要论述微型计算机开发系统的原理和应用。为了能够较深入地讨论，必须选择某一种开发系统作为分析和研究的对象。在这里我们选择了美国 INTEL 公司的MDS—236型的微型计算机开发系统，其原理方框图如图 1-1 所示。尽管现在的开发系统种类很多，但从它们的工作原理与基本结构来看，都是在 MDS—236 开发系统的基础上发展而来的。所以，如果能详细地了解这种开发系统的原理，很好地掌握它的使用方法，再应用其他类型的开发系统就不感到困难了。

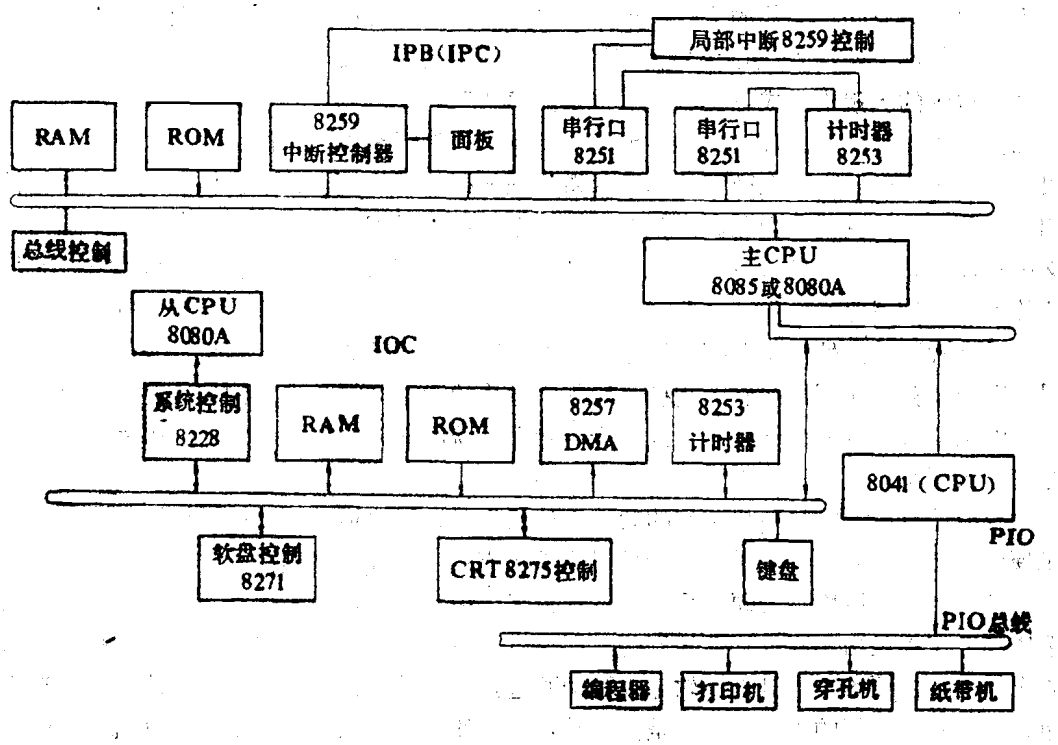


图1-1　MDS—236 型的微机开发系统的原理方框图

§1-2　微型计算机开发系统的特点和功能

微型计算机及微处理机系统有软件和硬件紧密结合互相依赖的特点，所以要求用于开发它们的工具亦要适应这些特点。就软件开发而言，可以在相同芯片组成的微型计算机系统上进行。一般的微型计算机系统都具备编辑、汇编以及高级语言的编译等系统软件，这些系统都是软件开发不可缺少的重要手段。用户应用软件在微型计算机上进行汇编或编译，除了能生成目标代码外，尚能对应用软件中语法上的错误作出提示，以便软件设计者及时排除这些错误。但是，用户软件中“算法”上的错误在汇编或编译阶段不易排除，这类错误只能在调试程序（DEBUG）的过程中发现并排除之。由于 DEBUG 是较低级别的调

试程序，用户依靠在 DEBUG 的控制下执行用户软件过程中才能发现软件中的错误。也就是说，在该程序运行的过程中，用户软件的设计者应密切地监视中央处理器(CPU)各寄存器、存贮单元及各标识的变化是否符合设计要求，来判断用户软件正确与否。这种调试方法人为因素太多，效率是不高的。

由前述可知，在微型计算机及微处理机系统的开发中，软件开发占很大的比重，也就是说提高软件开发的速度具有很重要意义。所以，仅仅依靠 DEBUG 来调试用户软件，显然是远远不能满足要求的。

此外，系统硬件存在的“故障”影响到软件正确运行，这类“故障”是无法直接在软件调试中得到明确提示的。由此可见，用上述方法来开发微型计算机及微处理机系统是不合适的，必须用一种新的开发工具来完成这项工作。微型计算机开发系统就是一种理想的工具。

根据微型计算机及微处理机系统的特点，微型计算机开发系统应具备下列功能：

·编辑程序方面

除了应具备一般微型计算机系统所有的“行编辑”功能外，还应有“屏幕编辑”的功能。这种编辑程序能充分利用显示终端设备本身的功能，并以光标作为编辑行指针，随时指示待输入字符的位置。由于光标是可见的，并且可用光标控制键移动光标，所以操作时既方便又直观，同时减少了按键错误，加快了文件编辑的速度，提高了效率。

·汇编程序方面

1. 宏汇编程序在生成目标代码的同时，还应输出符号表。这样，在调试程序时不仅可用绝对地址寻址，还能用符号寻址。这给使用者提供了很大的方便。

2. 为了便于模块的连接，要求汇编程序输出的代码模块中代码段和数据段分开。

3. 宏指令提供了调用同一子程序代码段等手段，从而提高了编制程序的效率。

4. 插入 (INCLUDE) 功能使使用者可应用原先已经编制好的、并且已存入磁盘文件内的代码段。汇编时，只要指出名字就可直接插入源程序中应用，这样可避免重复性的编制工作，加快了程序设计。

·程序调试方面

1. 应具备一般微型计算机系统程序调试的手段，如在监控程序控制下的各寄存器及存贮单元的显示和读、写命令，程序块传送命令，程序断点的设置，程序运行的命令等。还应有经汇编后生成的绝对目标程序，通过调试命令 (DEBUG) 装入内存，并在监控程序控制下用各种命令来调试用户程序的功能。

2. 应具备“在机仿效”的功能，通过一系列仿效命令实时地运行用户程序。在程序运行过程中可以设置各种条件作为程序断点的条件，以便检测该程序的正确性，同时能方便地发现程序中错误所在及原因。此外，还应对发生的“故障”进行“记录”，并使用相应的命令可随时显示“记录”的内容。通过各种显示出来的信息可以详细地分析程序出错的原因，以便及时纠正。

此外，还应具有把盘中的绝对目标文件装入内存，以及把调试好的程序存入盘中的命令。

资源出借功能。用户通过相应的命令向开发系统借用存贮器、输入/输出口及外部设备，以便在用户样机系统还未完全建立之前，就能进行用户系统的硬件调试。

3. 硬件故障的诊断。仿效器能发现硬件系统的故障，具备逻辑分析仪的部分功能。

·通用编程器

所谓通用编程器，实质上就是只读存贮器的写入器。在微型计算机及微处理机系统中，某些程序是固化在固件中的。这里所谓的固件就是只读存贮器(ROM或EPROM)。

微型计算机开发系统应有对只读存贮器（一般是指EPROM）的读、写功能，以便把调试好的程序固化在固件中。

从上面对微型计算机开发系统的要求可以看出，它与通用微型计算机相比，差别较大，功能亦丰富多了。总之，微型计算机开发系统是为了适应开发微型计算机系统及微处理机系统需要而产生的，它的功能越完备越好。

第二章　模块化程序设计

系统软件和大多数应用软件规模都比较大。设计这类软件很少采用“直线”的方式，因为这样做会给程序的设计和调试带来很多困难。

分析任何一个系统软件可知，它们常常是由许多功能组合而成的。针对每一种功能设计出相应程序段（或称程序模块），再把这些程序段“装配”成一个完整程序。这就是模块化程序设计方法。

§2-1　模块化程序设计的优点

一、程序的设计过程快

按不同的功能把一个大的程序分成若干个小模块，这些小模块完成某一特定的功能，它的本身有一定的独立性，因此可以单独编写程序和调试。这样，就可以由几个人分别进行程序设计和调试。每一个模块调试完成后，可通过库文件(LIB 程序)存入到文件库中。当全部模块调试完成后，再用连接程序(LINK程序)连接成一个完整的程序。

由于每个模块规模比较小，而且是针对某一个特定功能的，所以容易编写，在调试时容易发现错误和纠正错误。因此，就大大缩短了程序设计和调试的时间。

二、根据模块的特点来写程序

可根据不同模块的特点，用合适的程序语言编写模块程序。

在 MDS—236 开发系统中，可用三种语言来编写模块的源程序。它们是汇编语言(ASM80/85)；FORTRAN—80 语言和 PL/M—80 语言。用这三种语言编写的源程序经过汇编或编译之后生成浮动目标程序。然后便可用 LINK 程序连接成一个完整的程序。

三、模块可以共享

编写好的程序模块都被独立地存入到程序库中，不仅本程序可以用，而且也可以被其他程序引用。这样就为程序设计节省大量时间。

四、容易调试和纠正错误

模块的规模小且功能单一，便于调试。调试过程中错误较易发现，一旦出错只需修改本模块的源程序，然后再“翻译”成目标模块即可，不必重新“翻译”整个源程序。这又可节约时间。

§2-2　模块化程序设计的方法

下面以某一测量仪器为例来叙述程序模块设计的方法。这一仪器专用的微处理机系统

是以INTEL 8080A CPU为核心组成的小系统。按该系统的工作要求，可把用于它的系统软件分成：键盘的输入和处理；数据的采集与转换；数据的运算及处理；结果的显示及打印等几个部分。

一、程序模块的划分

程序模块的划分是根据该模块的功能决定的。按上面提出的实例而言，可以把它的系统软件分成与之对应的几个程序模块，并构成子程序的形式。因此，可以独立地编写和调试，在调试完成后再独立存入磁盘中。

该系统的软件可分为如下几个模块：

1. 键盘的输入和处理；
2. 数据的采集与转换（包括A/D转换）；
3. 数据的处理、运算(包括十进制浮点运算)及线性校正；
4. 数据的显示。用逐段按位扫描的方法，控制六个七段数码管的工作；
5. 输出数据除显示外，还可用微型打印机打印记录；
6. 整个系统的控制顺序、定时及同步均由主程序完成。它除了完成各子程序（程序模块）及时调用外，还对整个系统所用的符号进行定义，以便指供给各子程序应用。当然，各子程序专用的符号可在该子程序中定义。

二、源程序的编写

按照一般习惯，在充分了解模块的功能后，可以绘制本模块的流程图，便于设计者在编制源程序时作依据。

采用哪一种程序语言来编写模块的源程序，这要根据程序语言的特点和本模块的工作而决定的。但是，在规模较小（指系统软件）的系统中，为了节省程序存贮器的容量和加快处理速度，一般采用“汇编语言”来编写源程序。

在编写模块的源程序时，必须考虑到下面的情况，即各模块在调试完成后，要合成一个完整的系统程序（系统软件）。为了达到这一目的，必须在源程序清单的开头采用伪指令“CSEG”。这一伪指令使程序模块具有浮动（相对地址）的性质。

为了使程序模块能独立地进行调试和翻译成目标程序，所以在程序的末尾必须用伪指令“END”，而不能用返回指令“RET”结尾。否则不能作为一个完整的程序而无法进行翻译。

本程序模块用到的符号一定要有明确的定义。某些符号不能供好几个模块公用，所以通常是在主程序中给予定义。为了要调试子程序模块，这些符号必须在本程序段中进行定义。

可根据源程序清单调用开发系统配备的编辑程序，再通过键盘逐行地输入，生成模块源程序。MDS—236开发系统配备了行编辑程序和屏幕编辑程序。由于屏幕编辑程序具有很多优点，因而设计者都希望采用这种方法。具体使用方法，请参看有关用户手册，在这里不作详细叙述。

三、源程序的翻译——目标程序的生成

源程序是用不同语言编写的，它是不能运行的。为此，必须把它翻译成对应机器语言的目标程序。

这里以某一仪器的显示程序作为例子，并用 INTEL8080/8085 汇编语言编写的源程序。本模块定名为“PDFLE.SRC”。

下面将指出由于设计者疏忽而造成的几种常见的错误。

1. 汇编命令出错

如打入如下的汇编命令：

ASM80␣:F_2:PDFLED.SRC DEBGU PAGEWIDTH(80)

将显示 COMMON ERROR

这是由于控制词 DEBUG 错了，使得整个汇编命令失效。必须重新打入正确的命令：

ASM80 :F_2:PDFLED.SRC DEBUG PAGEWIDTH(80)

2. 源程序中的错误

这种错误是指不符合“汇编规则”的错误（或称为“语法”上的错误）。而不是程序“算法”上（或称为逻辑上）的错误。

当汇编结束后若显指出

ASSEMBLY COMPLETE. 5 ERRORS(××)

表示在汇编过程中产生 5 个错误。

从表 2-1 中的部分程序清单可以看出，第 5 行行首的字母“E”表示本行中有错误命令，其原因是把“*”号当作命令用了。第 7、19、20 行的错误都是由于字符串“KRPTRI”和“KEYPPR”没有定义而引起的。第13行的错误是数据FFH书写不符合规定造成的。汇编规则要求数据的第一个字符必须是字数（0～9），所以FFH应写成 0FFH，这样才符合要求。

表 2-1

LOC OBJ	LINE	SOURCE STATEMENT		
	1	* * * * * * * * * * * * * * *		
	2	* MDNKEY		*
	3	* KEY IN:	PTC/0-3	*
	4	* KEY OUT:	PTB/0-5	*
E 0000 0000	5	* * * * * * * * * * * * * * *		
( )				
	6	CSEG		
␣ 0008	7 KPTR	EQU	KRPTR1+8	
(5)				
1020	8 FACC4	EQU	1020H	
1026	9 MONTAB	EQU	FACC4+6	
18EA	10 KEYPPT	EOU	18EAH	
0008	11 PTC	EQU	08H	
0200	12 DISP	EQU	0200H	

续表

```
␣ 0000                13 DPL      EQU     FFH
(    7)
  000C                14 PTB      EQU     PTC+4
  0320                15 DISP2    EQU     DISP+0120H
  00DF                16 DPH      EQU     ODFH
  1080                17 AFLG     EQU     1080H
  1083                18 KEYFLG   EQU     AFLG+3
␣ 0004                19 KPTR1    EQU     KEYPPR+4
(   13)
␣ 0008                20 DACC     EQU     KEYPPR+8
(   19)
  18AO                21 ALOAD1   EQU     18AOH
  18A4                22 ALOAD2   EQU     ALOAD1+4
  18A8                23 ALOAD3   EQU     ALOAD1+8
  18AC                24 ALOAD4   EQU     ALOAD1+12
  18BO                25 ALOAD5   EQU     ALOAD1+16
  18B4                26 PLOAD1   EQU     ALOAD1+20
  18B8                27 PLOAD2   EQU     ALOAD1+24
  18BC                28 PLOAD3   EQU     ALOAD1+28
  0060                29 RES1     EQU     0060H
                      30 DECKY,
  0000 1E00           31          MVI E, 0
  0002 3ECO           32          MVI A, 0C0H
  0004 D30C           33          OUT,PTB
  0006 DBOB           34          IN,PTC
```

3. 符号重复定义的错误

程序中的符号必须经过定义，而且一般只能定义一次不能多次定义，否则将发生错误。

程序清单的行首若有“M”字母，则表示该行中的符号发生了重复定义，而每次定义使该符号具有不同意义。

如表2-2中各行的错误。其中行首注有字母“M”的，表示该符号重复定义，而且每次定义都有不同的数值。行首有字母“P”，表示该符号在汇编过程中数值发生变化。

如果确实需要对某一符号进行多次定义（即赋于不同的数值），可用宏定义的方法实现。如表2-3所示的程序段，它用到了宏定义。用伪指令LOCAL说明符号L2在程序执行过程中，每次遇到TIME时有不同的意义，即T_2和T_4有不同的数值。

表 2-2

1

2

3

续表

```
                          4
                          5
                          6                CSEG
 0018                     7 PTCL           EQU       18H
 8200                     8 SDMS           EQU       8200H
 820A                     9 KEYPPR         EQU       820AH
 0019                    10 PTA            EQU       PTCL+1
 001A                    11 PTB            EQU       PTCL+2
 00FE                    12 DPH            EQU       0FEH
 00BF                    13 DPL            EQU       0BFH
 8205                    14 DISPB          EQU       SDMS+5
 8210                    15 KPTR2          EQU       KEYPPR+6
M 8300                   16 DISP           EQU       8300H
  ( )
M 8340                   17 DISP1          EQU       DISP+40H
 (  16)
P 005E              C    18 DISP2          EQU       DISP2+20H
 (  17)
                         19 RDISP:
0000 2A1082              20                LHLD      KPTR2
0003 5E                  21                MOV       E,M
003C 00                  47                DB 0
003D 00                  48                DB 0
                         49 DISP2:
003E 3D                  50                DCR       A
003F C23E00         C    51                JNZ       DISP2
0042 C9                  52                RET
M                        53 DISP:
 (  18)
0043 2AOA82              54                LHLD      KEYPPR
0046 06FE                55                MVI       B, DPH
M                        56 DISP1:
 (  53)
  0048 5E                57                MOV       E,M
  0049 1600              58                MVI       D,0
```

表2-3

```
 TIME                    MACRO             T2, T4
                         LOCAL             L2
                         MVI               D, T2
 L2:                     MVI               A, T4
```

续表

```
                CALL        DELAY
                DCR         D
                JNZ         L2
                ENDM
START:          LXI         SP, STACK
L2:             TIME        50, 100
                MVI         C, "R"
                CALL        CO
                TIME        30, 60
                MVI         C, "S"
                CALL        CO
                JMP         L2
DELAY:          .
                .
                .
```

4. 程序结尾错误

程序模块通常是某一特定功能的子程序，它的结尾是返回指令RET，这是为了适应主程序调用时的需要。但是，把程序模块作为一个独立程序段进行汇编时，必须用“END”作为程序的最后一条指令。否则，将显示如下错误信息：

EOF ERROR

5. 汇编控制字的应用

各控制字的作用可参看用户手册。这里仅结合实际应用对某些常用的控制字作必要的论述。

在源程序清单编写好之后，立即可进行汇编操作。汇编的过程是磁盘驱动器工作最繁重的过程，因此磁盘驱动器较易损坏。为了减轻驱动器的工作，尽可能省去不必要的控制字。

在程序的第一次汇编中，设计者最关心的是程序中是否有语法上的错误，而不急于得到汇编后的目标文件。为此，可以使用“NOPRINT”及“NOOBJECT”控制字，从而在汇编过程结束后，将不生成任何目标文件而只显示：

ASSEMBLY COMPLETE, NO ERROR

或　　ASSEMBLY COMPLETE, X ERROR(××)

其中X是某一具体的数字。

如果显示没有错误，即可按要求进行汇编。若有错误，不妨再次调用编辑程序把源程序显示在屏幕上查找错误并进行改正。

如果源程序清单中错误比较多，为了在进一步的编辑中改正原程序中的错误，可以将这份清单打印下来，作为修正错误的参考。为此，要求在汇编的过程中，只要清单中的程序，不要符号表，也不要求生成目标文件。可用如下的命令：

ASM80 :F_2:PDFLED.SRC NOOBJECT NOSYBOLS (CR)

进行汇编操作。

命令中的“NOOBJECT”及“NOSYBOLS”两个控制字可以阻止目标文件及符号表的生成。这样既加快汇编操作速度又减少驱动器工作的时间，从而提高了设备的利用率及延长了设备的寿命。

总之，在汇编的操作中恰当地选用有关的控制字，防止不必要的文件的生成及无谓地加重设备负担。这是值得每个使用者注意的问题。

汇编控制字可写在汇编命令行的控制字的位置上，每个控制字之间用空格分开，如表2-3所示。此外，也可以将控制字写在源程序清单的第一行，但行首必须有符号“$”，每个控制字之间用空格分开。如果一行写不完可以另起一行续写，如表2-4所示。

为了使用方便，控制字有缺省规则。如果在汇编的命令中一个控制字也不用，则自动按缺省规则规定的控制字执行。

表2-4

```
                $MACROFILE NOSYMBOLS
                CSEG
CO              EQU         0F809H
                STKLN       4
TIME            MACRO       T2, T4
                LOCAL       L2
                MVI         D, T2
L2:             MVI         A, T4
                CALL        DELAY
                DCR         D
                JNZ         L2
                ENDM
START:          LXI         SP, STACK
L2:             TIME        50, 100
                MVI         C, "R"
                CALL        CO
                TIME        30, 60
                MVI         C, "S"
                CALL        CO
                JMP         L2
DELAY:
                END
```

当一个源文件按下面的汇编命令：

ASM80 :F_2:PDFLED.SRC

完成汇编后，自动地在第二号驱动器（即:F_2:）的磁盘上产生二个程序：

（1）浮动目标代码程序名为

:F_2:PDFLED. OBJ

文件的扩展部OBJ表示目标代码程序。文件名与源文件名相同。

（2）清单文件名为：

PDFLED.LST

扩展部为LST。

：F_2：是磁盘驱动器的设备名。它不是文件名的一部分，依据磁盘放入哪个驱动器而定。

清单文件又称列表文件，可显示或打印。使用如下命令：

COPY :F_2:PDFLED.LST TO :LP:

可把清单文件打印出来。其中，:LP:是行式打印机的设备名。

这一命令执行后立即打印出如表2-5所示的清单文件。

表2-5

```
ASM80 :F2: PDFLED. SRC NOPAGING PAGEWIDTH(55)
ISIS-II 8080/8085 MACRO ASSEMBLER, V4.0          MODULE    PAGE
LOC OBJ          LINE          SOURCE STATEMENT
                    1
                    2
                    3
                    4
                    5
                    6              CSEG
0018                7 PTCL         EQU       18H
0200                8 SDMS         EQU       0200H
010A                9 KEYPPR       EQQ       010AH
0019               10 PTA          EQU       PTCL+1
001A               11 PTB          EQU       PTCL+2
00FE               12 DPH          EQU       0FEH
00BF               13 DPL          EQU       0BFH
0205               14 DISPB        EQU       SDMS+5
0110               15 KPTR2        EQU       KEYPPR+6
                   16 RDISP:
0000 2A1001        17              LHLD      KPTR2
0003 5E            18              MOV       E, M
0004 1600          19              MVI       D, 0
0006 23            20              INX       H
0007 221001        21              SHLD      KPTR2
000A 212900    C   22              LXI       H, SEGPT
000D 19            23              DAD       D
000E 7E            24              MOV       A, M
000F D319          25              OUT       PTA
0011 3A0502        26              LDA       DISPB
0014 D31A          27              OUT       PTB
```