高等院校计算机基础课教材

微机原理与接口技术实验指导

黄海萍　陈用昌　编

国防工业出版社
http://www.ndip.cn

高等院校计算机基础课教材

微机原理与接口技术实验指导

黄海萍　陈用昌　编

国防工业出版社

·北京·

图书在版编目(CIP)数据

微机原理与接口技术实验指导 / 黄海萍,陈用昌编 . 北京:国防工业出版社, 2004.9
ISBN 7-118-03598-X

Ⅰ.微... Ⅱ.①黄...②陈... Ⅲ.①微型计算机 - 理论 - 高等学校 - 教学参考资料②微型计算机 - 接口 - 高等学校 - 教学参考资料 Ⅳ. TP36

中国版本图书馆 CIP 数据核字(2004)第 085231 号

国防工业出版社出版发行
(北京市海淀区紫竹院南路 23 号)
(邮政编码 100044)
国防工业出版社印刷厂 印刷
新华书店经售
*
开本 787×1092 1/16 印张 6½ 142 千字
2004 年 9 月第 1 版 2004 年 9 月北京第 1 次印刷
印数:1—4000 册 定价:12.00 元

前　言

“微机原理与接口技术”是一门具有很强实践性的课程，要通过不断地实践才能达到真正地学好该门课程的目的。为此我们专门编写了这本实验指导书，作为“微机原理与接口技术”、“微机系统与接口”等课程的配套教材，本着由浅入深的教学思路，通过提供系列化的软件、硬件综合实验手段来提高工科院校大学生计算机硬件与系统的应用水平，提高学生的独立思考和动手能力。

书中所涉及的实验平台是清华同方的TPC－H通用微机接口实验箱，在PC总线结构上把实验平台延伸到个人计算机总线的一个外部扩展。为便于学生学好这门课程，全书共分为两个部分：软件部分和硬件部分，其中软件部分旨在让学生熟悉汇编语言程序的建立和运行环境，掌握数据的输入输出操作和基本的功能调用；硬件部分实验覆盖了大中专院校微机接口实验教学大纲的主要内容，其中验证性实验主要是让学生对实验平台有一个比较感性的认识，了解各控制芯片的使用方法。设计性实验则要求学生具有一定的编程和硬件应用能力，实现控制功能的要求。综合性实验则要求学生要能熟练掌握各接口芯片的功用，能综合运用接口芯片达到实验要求。

本书在编写过程中得到了清华同方教学仪器设备公司以及冯一兵老师的大力支持和帮助。本实验指导由桂林电子工业学院景新幸教授和黄冰教授主审，同时得到了廖凯老师的协助，在此一并表示衷心感谢。由于编者水平有限，书中难免有错误与不当之处，恳请读者批评指正。

编　者

2004年7月

于桂林电子工业学院

内 容 简 介

微机原理与接口技术是工科院校的一门计算机基础课程，其任务是使学生从理论和实践中掌握计算机的基本组成、工作原理、接口电路及硬件连线；在此基础上提高软件和硬件的综合开发能力。长期以来尚没有一本系统性的实验配套教材。本书分软件和硬件两部分。软件部分先从汇编语言程序设计开始，介绍用汇编语言设计程序的基本方法，特别强调了 DEBUG 的调试方法；其次介绍循环、分支、子程序及宏的设计；由易到难，最终能熟练运用 8088 汇编语言。硬件部分先从验证开始，熟悉各种接口芯片的功能及使用方法；再自行用各种接口芯片设计接口电路，最终达到对基本的接口芯片能运用自如的目的。

目　录

第一章　软 件 部 分

第一节　汇编语言程序的建立和执行

一、程序的建立和执行

(一) 汇编语言上机过程(假定 U 盘为当前工作盘)

1. 编辑源程序

用 EDIT 编辑器编辑,文件后缀名为.ASM 的汇编语言源程序:

U > EDIT 文件名.ASM (当然也可以用其他编辑程序如 PCED 或行编辑程序 WS 或 TC 等编辑器来建立源文件。)

2. 汇编源文件

U > MASM 文件名.ASM

3. 链接目标文件

U > LINK 文件名.OBJ

4. 运行文件

U > 文件名.EXE

(二) DEBUG 调试命令

调试程序 DEBUG 是专门为汇编语言设计的一种调试工具,它通过步进、设置断点等方式为汇编语言程序员提供了非常有效的调试手段。

1. DEBUG 程序的调用

在 DOS 提示符下,可键入命令:

U > DEBUG [盘符:][路径][文件名.后缀][参数 1][参数 2]

其中,文件名是被调试的文件的名字,它必须是执行文件(.EXE),两个参数是运行被调试文件时所需要的命令参数,在 DEBUG 程序调入后出现提示符"—",此时,可键入所需的 DEBUG 命令。

2. DEBUG 的主要命令

(1) 显示内存单元内容的命令 D,格式为:

—D[地址]或

—D[范围]

(2) 修改内存单元内容的命令 E,它有如下两种格式。

① 用给定的内容代替指定范围的单元内容:

—E 地址　内容表

例如:—E　DS:100　F3 "XYZ" 8D

其中,F3、"X"、"Y"、"Z"、8D 各占 1 个字节,用这 5 个字节代替原内存单元 DS:100 到

104 的内容,“X”、“Y”、“Z”将分别按它们的 ASCII 值代入。

② 逐个修改相继的单元,格式为:

—E 地址

例如:

—E 100:

18E4:0100 89.78

此命令是将原 100 号单元的内容 89 改为 78,78 是程序员键入的。

(3) 检查和修改寄存器内容的命令 R,它有 3 种方式。

① 显示 CPU 内部所有寄存器内容和标志状态;格式为:

—R

R 命令显示中标志状态位的含义如表 1 - 1 所列。

表 1 - 1 R 命令显示中标志状态位的含义

标 志 名	置 位	复 位
溢出 Overflow (是/否)	OV	NV
方向 Deriction(减量/增量)	DN	UP
中断 Interrupt (允许/屏蔽)	EI	DI
符号 Sign(负/正)	NG	PL
零 Zero (是/否)	ZR	NZ
辅助进位 Auxiliary(是/否)	AC	NA
奇偶 Parity(偶/奇)	PE	PO
进位 Carry(是/否)	CY	NC

② 显示和修改某个指定寄存器的内容,格式为:

—R 寄存器名

例如打入:—R AX

系统将响应如下:

AX F1F4

:

表示 AX 当前内容为 F1F4,此时若不对其作修改,可按 ENTER 键,否则,打入修改后内容,如:

—R BX

BX 0369

:059F

则 BX 的内容由 0369 改为 059F。

③ 修改标志位状态,命令格式为:

—RF

系统将给出响应,如:

OV DN EI NG ZR AC PE CY -

此时若不做修改,可按 ENTER 键,否则在“ - ”号之后键入修改值,键入顺序任意,如:

OV　DN　EI　NG　ZR　AC　PE　CY－PONZDINV

(4) 运行命令 G,格式为:

—G[＝地址 1][地址 2[地址 3...]]

其中,地址 1 规定了运行起始地址,后面的若干地址均为断点地址。

(5) 追踪命令 T,有如下两种格式。

① 逐条指令追踪,格式为:

—T[＝地址]

该命令从指定地址起执行一条指令后停下来,显示寄存器内容和状态值。

② 多条指令追踪,格式为:

—T[＝地址][值]

该命令从指定地址起执行 n 条命令后停下来,n 由[值]确定。

(6) 汇编命令 A,格式为:

—A[地址]

该命令从指定地址开始允许输入汇编语句,把它们汇编成机器代码相继存放在从指定地址开始的存储器中。

(7) 反汇编命令 U,有如下两种格式。

① —U[地址]

该命令从指定地址开始,反汇编 32 个字节,若地址省略,则从上一个 U 命令的最后一条指令的下一单元开始显示 32 个字节。

② —U 范围

该命令对指定范围的内存单元进行反汇编,例如:

—U 04BA:0100 0108 或

—U 04BA:0100 L9

这两个命令是等效的。

(8) 命名命令 N,格式为:

—N 文件标识符 [文件标识符]

此命令将两个文件标志符格式化在 CS:6CH 的两个文件控制内,以便使用 L 或 W 命令把文件装入或者存盘。

(9) 装入命令 L,它有如下两种功能。

① 把磁盘上指定扇区的内容装入到内存指定地址起始的单元中,格式为:

—L　地址　驱动器　扇区号　扇区数

② 装入指定文件,格式为:

—L [地址]

此命令装入已在 CS:5CH 中格式化的文件控制块所指定的文件。在用 L 命令前,BX 和 CX 中应包含所读文件的字节数。

(10) 写命令 W,有如下两种格式。

① 把数据写入磁盘的指定扇区,格式为:

—W　地址　驱动器　扇区号　扇区数

② 把数据写入指定文件中,格式为:

—W [地址]

此命令把指定内存区域中的数据写入由 CS:5CH 处的 FCB 所规定的文件中。在用 W 命令前,BX 和 CX 中应包含要写入文件的字节数。

(11) 退出 DEBUG 命令 Q,该命令格式为:

—Q

它退出 DEBUG 程序,返回 DOS,但该命令本身并不把在内存中的文件存盘,如需存盘,应在执行 Q 命令前先执行写命令 W。

(三) 常用 DOS 调用

1. 字符输入

```
MOV   AH,01H
INT   21H
```

2. 字符串输入

```
.
.
.
BUF   DB 30
      DB ?
      DB 30 DUP (?) ; 定义缓冲区
.
.
.
LEA   DX,BUF
MOV   AH,0AH
INT   21H          ; 0A 号系统功能调用
```

3. 字符输出

```
MOV   AH,02H
MOV   DL,'A'
INT   21H          ;输出 A 字符
```

4. 字符串输出

```
MOV   AH,09H
LEA   DX,字符串地址
INT   21H
```

5. 返回DOS

```
MOV   AH,4CH
INT   21H
```

二、编程举例

[例] 比较字符串(文件名取为:CMPST.ASM)

试编写一程序:比较两个字符串 STRING1 和 STRING2 所含的字符是否相同。若相同则显示'Match',否则,显示'No match'。

这里可以用串比较指令来完成程序所要求的功能。上机过程如下。

1. 用 EDIT 建立汇编源程序

(当然也可以用其他编辑程序如 PCED 或行编辑程序 WS 来建立源文件。)

```
U > EDIT CMPST.ASM  < 回车 >
;PROGRAM TITLE GOES HERE----COMPARE STRING
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
DATAREA SEGEMENT                                        ;DEFINE DATA SEGMENT
  STRING1       DB       'Move the cursor backward.'
  STRING2       DB       'Move the cursor backwrad.'
  MESS1         DB       'Match.',13,10,'$'
  MESS2         DB       'No match!',13,10,'$'
DATAREA ENDS
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
PROGNAM SEGMENT                                         ;DEFINE CODE SEGMENT
;-----------------------------------------------------------------------------
MAIN PROC     FAR
    ASSUMECS: PROGNAM,DS: DATAREA, ES: DATAREA
START:
;SET UP  STACK FOR RETURN
       PUSH   DS                  ;SAVE OLD DATA SEGMENT
       SUB    AX,AX               ;PUT ZERO IN AX
       PUSH   AX                  ;SAVE IT ON STACK
;SET DS REGISTER TO CURRENT DATE SEGMENT
       MOV    AX,DATAREA          ;DATAREA SEGMENT ADDR
       MOV    DS,AX               ;INTO DS REGISTER
       MOV    ES,AX               ;INTO ES REGISTER
;MAIN PART OF PROGRAM GOES HERE
       LEA      SI,STRING1
       LEA      DI,STRING2
       CLD
       MOV      CX,25
       REPZ     CMPSB
       JZ       MATCH
```

```
        LEA     DX,MESS2
        JMP     SHORT DISP
MATCH:
        LEA     DX,MESS1
DISP:
        MOV     AH,09
        INT     21H
        RET                                     ;RETURN TO DOS
MAIN    ENDP                                    ;END OF MAIN PART OF PROGRAM
-------------------------------------------------------------------------------
PROGNAM ENDS                                    ;END OF CODE SEGMENT
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
        END     START                           ;END ASSEMBLY
```

将文件存入磁盘,并使系统返回 DOS。

2. 汇编程序 MASM(或 ASM)对源文件汇编产生目标文件.OBJ

```
U > MASM CMPST; < 回车 >
THE IBM PERSONAL COMPUTER MACRO ASSEMBLER
VERSION 1.00 (C) COPYRIGHT IBM CORP 1981
WARNING ERRORS 0
SEVERE ERRORS 0
```

如汇编指示出错则需重新调用编辑程序修改错误,直至汇编通过为止。如调试时需要用 LST 文件,则应在汇编过程中建立该文件。

3. 连接程序 LINK 产生执行文件 EXE

```
U > LINK CMPST; < 回车 >
IBM 5550 MULTISTATION LINKER 2.00
(C ) COPYRIGHT IBM CORP.1983
WARNING:NO STACK SEGMENT
THERE WAS 1 ERROR DETECTED.
```

4. 执行程序

可直接从 DOS 执行程序如下:

```
U > CMPST < 回车 >
Match.
```

终端上已显示出程序的运行结果。为了调试程序的另一部分,可重新进入编辑环境中(EDIT CMPST. ASM)程序修改两个字符串的内容,使它们互不相同。如修改后的数据区为:

```
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
DATAREA SEGMENT                    ;DEFINE DATA SEGMENT
    STRING1     DB      'Move the cursor backward.'
    STRING2     DB      'Move the cursor forward.'
    MESS1       DB      'Match.',13,10,'$'
    MESS2       DB      'No match!',13,10,'$'
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

然后重新汇编、链接、执行,结果为:

```
U>CMPST<回车>
No match!
```

至此,程序已调试完毕,运行结果正确。

另一种调试程序的方法是使用 DEBUG 程序。可调用如下:

```
U>DEBUG CMPST.EXE<回车>
-
```

此时,DEBUG 已将执行程序装入内存,可直接用 G 命令运行程序。

```
-G
Match.
PROGRAM TERMINATED NORMALLY
```

为了调试程序的另一部分,可在 DEBUG 中修改字符串内容。可先用 U 命令显示程序,以便了解指令地址。显示结果如下:

```
-U
19F3:0000    1E          PUSH    DS
19F3:0001    2BC0        SUB     AX,AX
19F3:0003    50          PUSH    AX
19F3:0004    B8EE19      MOV     AX,19EE
19F3:0007    8ED8        MOV     DS,AX
19F3:0009    8EC0        MOV     ES,AX
19F3:000B    8D360000    LEA     SI,[0000]
19F3:000F    8D3E1900    LEA     DI,[0019]
19F3:0013    FC          CLD
19F3:0014    B91900      MOV     CX,0019
19F3:0017    F3          REPZ
19F3:0018    A6          CMPSB
19F3:0019    7406        JZ      0021
19F3:001B    8D163B00    LEA     DX,[003B]
19F3:001F    EB04        JMP     0025
```

```
-U
19F3:0021  8D163200          LEA     DX,[0032]
19F3:0025  B409              MOV     AH,09
19F3:0027  CD21              INT     21
19F3:0029  CB                RETF
19F3:002A  FF7501            PUSH    [DI+01]
19F3:002D  40                INC     AX
19F3:002E  5A                POP     DX
19F3:002F  22C2              AND     AL,DL
19F3:0031  50                PUSH    AX
19F3:0032  807EDC20          CMP     BYTE PTR [BP-24],20
19F3:0036  B0FF              MOV     AL,FF
19F3:0038  7201              JB      003B
19F3:003A  40                INC     AX
19F3:003B  5A                POP     DX
19F3:003C  22C2              AND     AL,DL
19F3:003E  50                PUSH    AX
19F3:003F  80F920            CMP     CL,20
```

将断点设置在程序的主要部分运行之前。

```
-G0B
AX=19EE  BX=0000  CX=007A  DX=0000  SP=FFFC  BP=0000  SI=0000  DI=0000
DS=19EE  ES=19EE  SS=19EE  CS=19F3  IP=000B  NV UP DI PL ZR NA PE NC
19F3:000B  8D360000          LEA     SI,[0000]                  DS:0000=6F4D
```

根据其中指示的 DS 寄存器内容(19EE)查看数据段的情况如下：

```
-D0
19EE:0000  4D 6F 76 65 20 74 68 65-20 63 75 72 73 6F 72 20   Move the cursor
19EE:0010  62 61 63 6B 77 61 72 64-2E 4D 6F 76 65 20 74 68   backward. Move th
19EE:0020  65 20 63 75 72 73 6F 72-20 62 61 63 6B 77 61 72   e cursor backwar
19EE:0030  64 2E 4D 61 74 63 68 2E-0D 0A 24 4E 6F 20 6D 61   d. Match...$No ma
19EE:0040  74 63 68 21 0D 0A 24 00-00 00 00 00 00 00 00 00   tch!..$.........
19EE:0050  1E 2B C0 50 B8 EE 19 8E-D8 8E C0 8D 36 00 00 8D   .+@p8n..X.@.6...
19EE:0060  3E 19 00 FC B9 19 00 F3-A6 74 06 8D 16 3B 00 EB   >..|9..s&t..;.k
19EE:0070  04 8D 16 32 00 B4 09! CD-21 CB FF 75 01 40 5A 22   ...2.4.M! K.u.@Z"
```

可用 E 命令修改数据区的字符串,操作如下：

```
-E29
19EE:0029   62.66   61.6f   63.72   6B.77   77.61   61.72   72.64
```

```
19EE:0030    64.2e    2E.20
```

再次用 D 命令查看修改结果。

```
-D0
19EE:0000  4D 6F 76 65 20 74 68 65-20 63 75 72 73 6F 72 20   Move the cursor
19EE:0010  62 61 63 6B 77 61 72 64-2E 4D 6F 76 65 20 74 68   backward. Move th
19EE:0020  65 20 63 75 72 73 6F 72-20 66 6F 72 77 61 72 64   e cursor forward
19EE:0030  2E 20 4D 61 74 63 68 2E-0D 0A 24 4E 6F 20 6D 61   . Match...$No ma
19EE:0040  74 63 68 21 0D 0A 24 00-00 00 00 00 00 00 00 00   tch!..$........
19EE:0050  1E 2B C0 50 B8 EE 19 8E-D8 8E C0 8D 36 00 00 8D   .+@p&n..X.@.6...
19EE:0060  3E 19 00 FC B9 19 00 F3-A6 74 06 8D 16 3B 00 EB   〉..19..s&t...;.k
19EE:0070  04 8D 16 32 00 B4 09 CD-21 CB FF 75 01 40 5A 22   ...2.4.M! K.u.@Z"
```

用 G 命令运行程序，结果为：

```
-G
No match!
PROGRAM TERMINATED MORMALLY
```

用 Q 命令退出 DEBUG。

```
-Q
```

至此，程序已调试完毕。为了进一步说明 DEBUG 命令的使用方法，我们再次重复上述程序的调试过程，只是使用 E、A 和 F 命令来修改数据区的内容而已。必须注意，由于在用 DEBUG 调试程序时，只能修改当时有关的内存单元内容，因此重新用 DEBUG 装入执行程序时，仍是原来在磁盘中的文件内容。操作如下：

```
    U > DEBUG CMPST.EXE < 回车 >
-G0B
AX = 19EE  BX = 0000  CX = 007A  DX = 0000  SP = FFFC  BP = 0000  SI = 0000  DI = 0000
DS = 19EE  ES = 19EE  SS = 19EE  CS = 19F3  IP = 000B      NV UP DI PL ZR NA PE NC
19F3:000B 8D360000            LEA     SI,[0000]                 DS:0000 = 0F4D
-D0
19EE:0000  4D 6F 76 65 20 74 68 65-20 63 75 72 73 6F 72 20   Move the cursor
19EE:0010  62 61 63 6B 77 61 72 64-2E 4D 6F 76 65 20 74 68   backward. Move th
19EE:0020  65 20 63 75 72 73 6F 72-20 62 61 63 6B 77 61 72   e cursor backwar
19EE:0030  64 2E 4D 61 74 63 68 2E-0D 0A 24 4E 6F 20 6D 61   d.Match...$No ma
19EE:0040  74 63 68 21 0D 0A 24 00-00 00 00 00 00 00 00 00   tch!..$........
19EE:0050  1E 2B C0 50 B8 EE 19 8E-D8 8E C0 8D 36 00 00 8D   .+@p&n..X.@.6...
19EE:0060  3E 19 00 FC B9 19 00 F3-A6 74 06 8D 16 3B 00 EB   〉..19..s&t...;.k
19EE:0070  04 8D 16 32 00 B4 09 CD-21 CB FF 75 01 40 5A 22   ...2.4.M! K.u.@Z"
-E29 'forward.' 20
```

```
-D0
19EE:0000  4D 6F 76 65 20 74 68 65-20 63 75 72 73 6F 72 20   Move the cursor
19EE:0010  62 61 63 6B 77 61 72 64-2E 4D 6F 76 65 20 74 68   backward. Move th
19EE:0020  65 20 63 75 72 73 6F 72-20 66 6F 72 77 61 72 64   e cursor forward
19EE:0030  2E 20 4D 61 74 63 68 2E-0D 0A 24 4E 6F 20 6D 61   .Match...$No ma
19EE:0040  74 63 68 21 0D 0A 24 00-00 00 00 00 00 00 00 00   tch!.. $........
19EE:0050  1E 2B C0 50 B8 EE 19 8E-D8 8E C0 8D 36 00 00 8D   .+@p8n..X.@.6...
19EE:0060  3E 19 00 FC B9 19 00 F3-A6 74 06 8D 16 3B 00 EB   >..19..s&t...;.k
19EE:0070  04 8D 16 32 00 B4 09 CD-21 CB CC CC CC CC CC CC   ...2.4.M! KLLLLLL
-G
No match!
PROGRAM TERMINATED NORMALLY
```

可见这种 E 命令的方式避免使用 ASCII 码进入,对用户是比较方便的。其中最后一个 20 是空格键的 ASCII 码,以补足原来的字节数。

也可使用 A 命令字把数据区的内容恢复原状,具体如下:

```
-A19EE:29
19EE:0029          DB          'backward.'
19EE:0032
-D0
19EE:0000  4D 6F 76 65 20 74 68 65-20 63 75 72 73 6F 72 20   Move the cursor
19EE:0010  62 61 63 6B 77 61 72 64-2E 4D 6F 76 65 20 74 68   backward. Move th
19EE:0020  65 20 63 75 72 73 6F 72-20 62 61 63 6B 77 61 72   e cursor backwar
19EE:0030  64 2E 4D 61 74 63 68 2E-0D 0A 24 4E 6F 20 6D 61   d.Match...$No ma
19EE:0040  74 63 68 21 0D 0A 24 00-00 00 00 00 00 00 00 00   tch! ..$ ........
19EE:0050  1E 2B C0 50 B8 EE 19 8E-D8 8E C0 8D 36 00 00 8D   .+@p8n..X.@.6...
19EE:0060  3E 19 00 FC B9 19 00 F3-A6 74 06 8D 16 3B 00 EB   >..19..s&t...;.k
19EE:0070  04 8D 16 32 00 B4 09 CD-21 CB 6F 72 77 61 72 64   ...2.4.M! Korward
-G
Match.
AX=0924  BX=0000  CX=0000  DX=0032  SP=FFFC  BP=FFFF  SI=0019  DI=0032
DS=19EE  ES=19EE  SS=19EE  CS=19EE  IP=0096       NV UP DI NG NZ AC PE NC
19EE:0096 CC                INT     3
-Q
```

由于 A 命令是汇编命令,因此信息是用汇编格式进入的。如果修改的是程序中的语句,方法也是相同的。现在再看一下用 F 命令修改数据区的方法。

```
U>DEBUG CMPST.EXE<回车>
-G0B
```