TPC－1型

十六位微机实验培训系统

教师用实验指导书

目　录

汇编语言程序的建立和执行

《TPC－1型十六位微机实验培训系统》要求用汇编语言来编写程序。现在，我们来说明汇编语言程序从建立到执行的过程。

要建立和运行汇编语言程序，系统盘上应该首先有如下文件：

EDLIN.COM	行编辑程序
ASM.EXE	汇编程序
(或 MASM.EXE)	宏汇编程序
LINK.EXE	链接程序
DEBUG.COM	调试程序

其中 ASM.EXE 是普通汇编程序，它不支持宏汇编，如果要用宏汇编，则必须用 MASM.EXE。

下面，以建立和执行用户程序 ABC.EXE 为例来说明上机过程。

1．用 EDLIN 命令建立汇编语言源程序（ASM 文件）

源程序就是用汇编语言的语句编写的程序，它不能被机器识别。源程序必须以 ASM 为附加文件名。

例如打入命令：

A＞EDLIN ABC.ASM↙（下面带横线的部分为键盘输入）

此时用户可以通过编辑程序的插入命令编写用户程序 ABC.ASM。至于 EDLIN 的使用方法可以查阅手册。

2．用 ASM（或者 MASM）命令产生目标文件［OBJ 文件］

源程序建立以后，就要可以用汇编程序 ASM.EXE（或者 MASM.EXE）进行汇编。所谓汇编，实际上就是把以 ASM 为附加名的源文件转换成用二进制代码表示的目标文件，目标文件以 OBJ 为附加名。汇编过程中，汇编程序对源文件进行二次扫描，如果源程序中有语法错误，则汇编过程结束后，汇编程序会指出源程序中的错误，这时，用户可以再用编辑程序来修改源程序中的错误，最后，得到没有语法错误的 OBJ 文件。

例如：对 ABC.ASM 的汇编过程如下：

A＞ASM　ABC.ASM↙

此时，汇编程序给出如下回答：

The IBM Personal Computer Assembler Version 1.00 (C) Copyright IBM Corp 1989

Object filename [EX MOVS.OBJ]:

Source listing [NUL. LST]: ABC

Cross reference [NUL. CRF:] ABC

如果被汇编的程序没有语法错误，则屏幕上还给出如下信息：

Warning Severs

Errors Errors

0 0

从上面的操作过程可以见到，汇编程序的输入文件就是用户编写的汇编语言源程序，它必须以 ASM 为文件扩展名。汇编程序的输出文件有三个，第一个是目标文件，它以 OBJ 为扩展名，产生 OBJ 文件是我们进行汇编操作的主要目的，所以这个文件是一定要产生，也一定会产生的，操作时，这一步只要打入回车就行了；第二个是列表文件，它是 LST 为扩展名，列表文件同时给出源程序和机器语言程序，从而，可以使调试变得方便，列表文件是可有可无的，如果不需要，则在屏幕上出现提示信息 [NUL.LST]: 时，打入回车即可。如果需要，则打入文件名和回车；第三个是交叉符号表，此表给出了用户定义的所有符号，对每个符号都列出了将其定义的所在行号和引用的行号，并在定义行号上加上“#”号，同列表文件一样，交叉符号表也是为了便于调试而设置的，对于一些规模较大的程序，交叉符号表为调试工作带来很大的方便。当然，交叉符号表也是可有可无的，如不需要，那么在屏幕上出现提示信息 [NUL.CRF]: 时，打入回车即可。

汇编过程结束时，会给出源程序中的警告性错误 [Warning Errors] 和严重错误 [Severs Errors]，前者指一般性错误，后者指语法性错误，当存在这两类错误时，屏幕上除指出错误个数外，还给出错误信息代码，程序员可以通过查找手册弄清错误的性质。

如果汇编过程中，发现有错误，则程序员应该重新用编辑命令修改错误，再进行汇编，最终直到汇编正确通过。要指出的是汇编过程中只能指出源程序中的语法错误，并不能指出算法错误和其他错误。

3. 用 LINK 命令产生执行文件（EXE 文件）

汇编过程根据源程序产生出二进制的目标文件（OBJ 文件），但 OBJ 文件用的是浮动地址，它不能直接上机执行，所以还必须使用链接程序（LINK）将 OBJ 文件转换成可执行的 EXE 文件。LINK 命令还可以将某一个目标文件和其他多个模块（这些模块可以是由用户编写的，也可以是某个程序库中存在的）链接起来。

具体操作如下（以对 ABC.OBJ 进行链接为例）：

A>LINK ABC↙

此时，在屏幕上见到如下回答信息：

IBM 5552 Multistation Linker 2.00 (C) Copyright IBM Corp. 1985

Run File [ABC.EXE]:

List File [NUL.MAP]:

Libraries [.LIB]

Warning: NO STACK Segment

LINK 命令有一个输入文件，即 OBJ 文件，有时，用户程序用到库函数，此时，对于提示信息 Libraries [.LIB]，要输入库名。

LINK 过程产生两个输出文件，一个是扩展名为 EXE 的执行文件，产生此文件当然是 LINK 过程的主要目的，另一个是扩展名为 MAP 的列表分配文件，有人也称它为映象文件，它给出每个段在内存中的分配情况，比如某一个列表分配文件为如下内容：

Warning：NO STACK Segment

start	Stop	Length	Name Class
0000H	0015H	0016H	CODE
0020H	0045H	0026H	DATA
0050H	0061H	0012H	EXTRA

origin Group

Program entry Point at 0000：0000

MAP 文件也是可有可无的。

从 LINK 过程的提示信息中，可看到最后给出了一个"无堆栈段"的警告性错误，这并不影响程序的执行。当在，如果源程序中设置了堆栈段，则无此提示信息。

4．程序的执行

有了 .EXE 文件后，就可以执行程序了，此时，只要打入文件名即可。仍以 ABC 为例；

A＞ABC↙

A＞

实际上，大部分程序必须经过调试阶段才能纠正程序设计中的错误，从而得到正确的结果。所谓调试阶段，就是用调试程序（DEBUG 程序）发现错误，再经过编辑，汇编，链接纠正错误。关于 DEBUG 程序中的各种命令，可参阅 DOS 手册，下面给出最常用的几个命令。

先进入 DEBUG 程序并装入要调试的程序 ABC.EXE，操作命令如下：

A＞DEBUG　ABC.EXE↙；进入 DEGUG，并装配 ABC.EXE

—

此时，屏幕上出现一个短划线，为了察看程序运行情况，常常要分段运行程序，为此，要设立"断点"，即让程序运行到某处自动停下，并把所有寄存器的内容显示出来。为了确定我们所要设定的断点地址，常常用到反汇编命令，反汇编命令格式如下：

－U↙；从当前地址开始反汇编

也可以从某个地址处开始反汇编，如下所示。

－U200↙；从 CS：200 处开始反汇编

程序员心中确定了断点地址后，就可以用 G 命令来设置断点，比如，想把断点设置在 0120H 处，则如下打入命令：

－G0120↙

此时，程序在 0120H 处停下，并显示出所有寄存器以及各标志位的当前值，在最后一行还给出下一条将要执行的指令的地址、机器语言和汇编语言。程序员可以从显示的寄存器的内容来了解程序运行是否正确。

对于某些程序段，单从寄存器的内容看不到程序运行的结果，而需要观察数据段的内容，此时可用 d 命令，使用格式如下：

－d DS：0000↙；从数据段的 0 单元开始显示 128 个字节

在有些情况下，为了确定错误到底由哪条指令的执行所引起，要用到跟踪命令。跟踪命令也

叫单步执行命令，此命令使程序每执行一条指令，便给出所有寄存器的内容。

比如：

-T3↙；从当前地址往下执行三条指令

此命令使得从当前地址往下执行三条指令，每执行一条，便给出了各寄存器内容，最后，给出下一条要执行的指令的地址、机器语言和汇编语言。

从 DEBUG 退出时，使用如下命令：

-Q↙

每一个有经验的程序员必定熟练掌握调试程序的各主要命令。为此，初学者要花一些功夫查阅、掌握 DOS 手册中关于 DEBUG 程序说明。

一、软件部分

实验一　两个多位十进制数相加的实验

一、实验目的

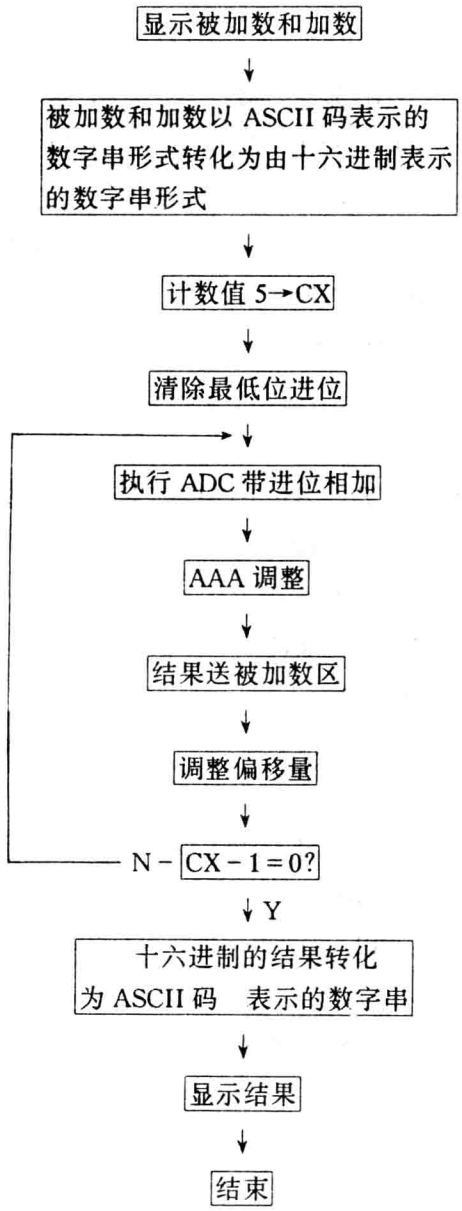
1. 学习数据传送和算术运算指令的用法。

2. 熟悉在 PC 机上建立、汇编、链接、调试和运行 8088 汇编语言程序的过程。

二、实验内容

将两个多位十进制数相加。要求被加数均以 ASCII 码形式各自顺序存放在以 DATA1 和 DATA2 为首的 5 个内存单元中（低位在前），结果送回 DATA1 处。

三、程序框图



四程序清单

```
CRLF    MACRO                                   ; 建立宏指令 CRLF
        MOV         DL, 0DH
        MOV         AH, 02H
        INT         21H
        MOV         DL, 0AH
        MOV         AH, 02H
        INT         21H
ENDM
DATA    SEGMENT
DATA1   DB  33H, 39H, 31H, 37H, 34H             ; 第一个数据（作为被加数）
DATA2   DB  36H, 35H, 30H, 38H,. 32H            ; 第二个数据（作加数）
DATA    ENDS
STACK   SEGMENT                                 ; 堆栈段
STA     DB          20 DUP (?)
TOP     EQU         LENGTH STA
STACK   ENDS
CODE    SEGMENT
ASSUME  CS: CODE, DS: DATA, SS: STACK, ES: DATA
START:  MOV         AX, DATA
        MOV         DS, AX
        MOV         AX, STACK
        MOV         SS, AX
        MOV         AX, TOP
        MOV         SP, AX
        MOV         SI, OFFSET DATA2
        MOV         BX, 05
        CALL        DISPL                       ; 显示被加数
        CRLF
        MOV         SI, OFFSET DATA1
        MOV         BX, 05                      ; 显示被加数
        CALL        DISPL
        CRLF
```

```
        MOV     DI, OFFSET DATA2
        CALL    ADDA                    ; 加法运算
        MOV     SI, OFFSET DATA1
        MOV     BX, 05                  ; 显示结果
        CALL    DISPL
        CRLF
        MOV     AX, 4C00H
        INT     21H
DISPL   PROC    NEAR                    ; 显示子功能
DS1:    MOV     AH, 02
        MOV     DL, [SI+BX-1]           ; 显示字符串中的一字符
        INT     21H
        DEC     BX                      ; 修改偏移量
        JNZ     DS1
        RET
DISPL   ENDP
ADDA    PROC    NEAR
        MOV     DX, SI
        MOV     BP, DI
        MOV     BX, 05
AD1:    SUB     BYTE PTR [SI+BX-1], 30H
        SUB     BYTE PTR [DI+BX-1], 30H
        DEC     BX                      ; 将 ASCII 码表示的数字串转化为
                                        ; 十六进制的数字串
        JNZ     AD1
        MOV     SI, DX
        MOV     DI, BP
        MOV     CX, 05                  ; 包括进位，共 5 位
        CLC                             ; 清进位位
AD2:    MOV     AL, [SI]
        MOV     BL, [DI]
        ADC     AL, BL                  ; 带进位相加
        AAA                             ; 非组合 BCD 码的加法调整
```

```
        MOV     [SI], AL        ; 结果送被加数区
        INC     SI
        INC     DI              ; 指向下一位
        LOOP    AD2             ; 循环
        MOV     SI, DX
        MOV     DI, BP
        MOV     BX, 05
AD3:    ADD     BYTE PTR [SI+BX-1], 30H
        ADD     BYTE PTR [DI+BX-1], 30H
        DEC     BX              ; 十六进制的数字串转化为 ASCII
                                  码表示的数字串
        JNZ     AD3
        RET
ADDA    REDP
CODE    ENDS
END     START
```

实验 2　两个数相加的实验

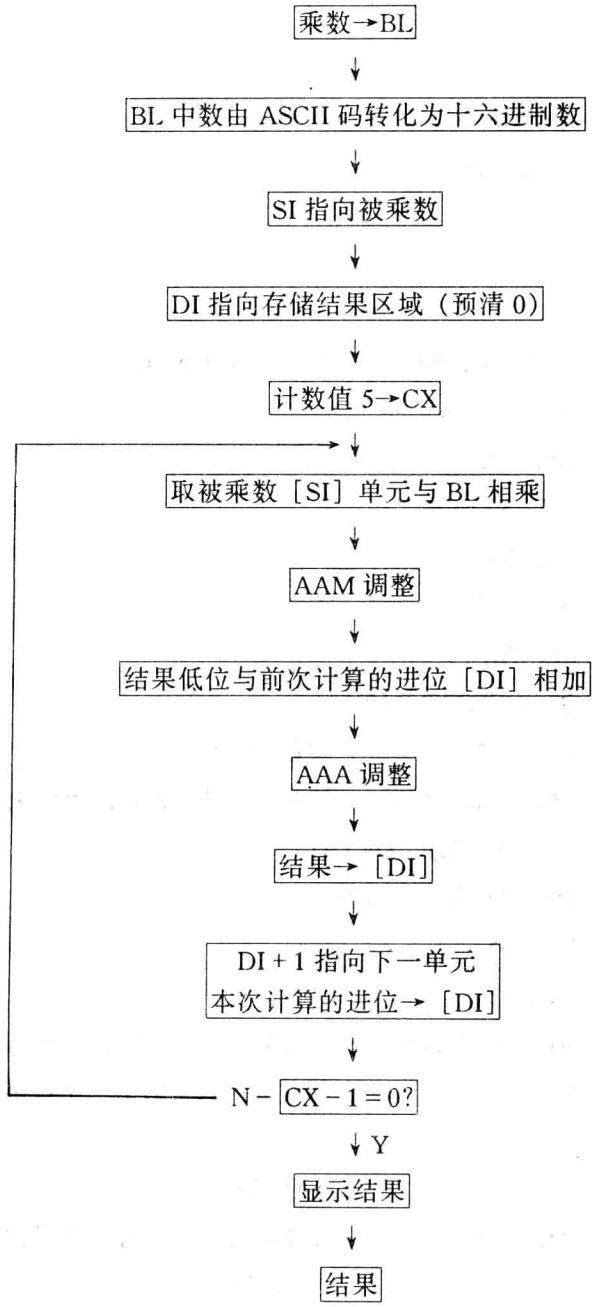
1、实验目的

掌握乘法指令和循环指令的用法

二、实验内容

1. 实现十进制数的乘法。被乘数和乘数均以 ASCII 码形式存放在内存中，乘积在屏幕上显示出来。

三、程序框图



四、程序清单

```
DATA    SEGMENT
DATA1   DB      32H, 39H, 30H, 35H, 34H
DATA2   DB      33H
RESULT  DB      6 DUP (00H)
DATA    ENDS
STACK   SEGMENT
STA     DB      20 DUP (?)
TOP     EQU     LENGTH  STA
STACK   ENDS

CODE    SEGMENT
ASSUME  CS, CODE, DS: DATA, SS: STACK, ES: DATA
START:  MOV     AX, DATA
        MOV     DS, AX
        MOV     AX, STACK
        MOV     SS, AX
        MOV     AX, TOP
        MOV     SP, AX
        MOV     SI, OFFSET DATA2
        MOV     BL, [SI]        ; 乘数 2→BL
        AND     BL, 00001111B   ; 屏蔽高 4 位，ASCII 码转化为十六进制数
        MOV     SI, OFFSET, DATA1
        MOV     DI, OFFSET, RESULT
        MOV     CX, 05
LOOP1:  MOV     AL, [SI]
        AND     AL, 00001111B   ; 取被乘数 1，ASCII 变十六进制数
        INC     SI              ; 指向被乘数 1 的下一字符单元
        MUL     BL              ; 相乘
        AAM                     ; AAM 调整
        ADD     AL, [DI]        ; 结果低位与前次计算的进位相加
        AAA                     ; AAA 调整
```

```
        MOV     [DI], AL
        INC     DI              ; 结果送存下一单元
        MOV     [DI], AH
        LOOP    LOOP1           ; 计算结果高位进位送存
        MOV     CX, 06
        MOV     SI, OFFSET RESULT
DISPL:  MOV     AH, 02
        MOV     DL, [SI]
        ADD     DL, 30H
        INT     21H
        DEC     SI
        LOOP    DISPL           ; 显示结果
        MOV     AX, 4C00H
        INT     21H             ; 结束
CODE    ENDS
END     START
```

实验 3　BCD 码相乘的程序

一、实验目的

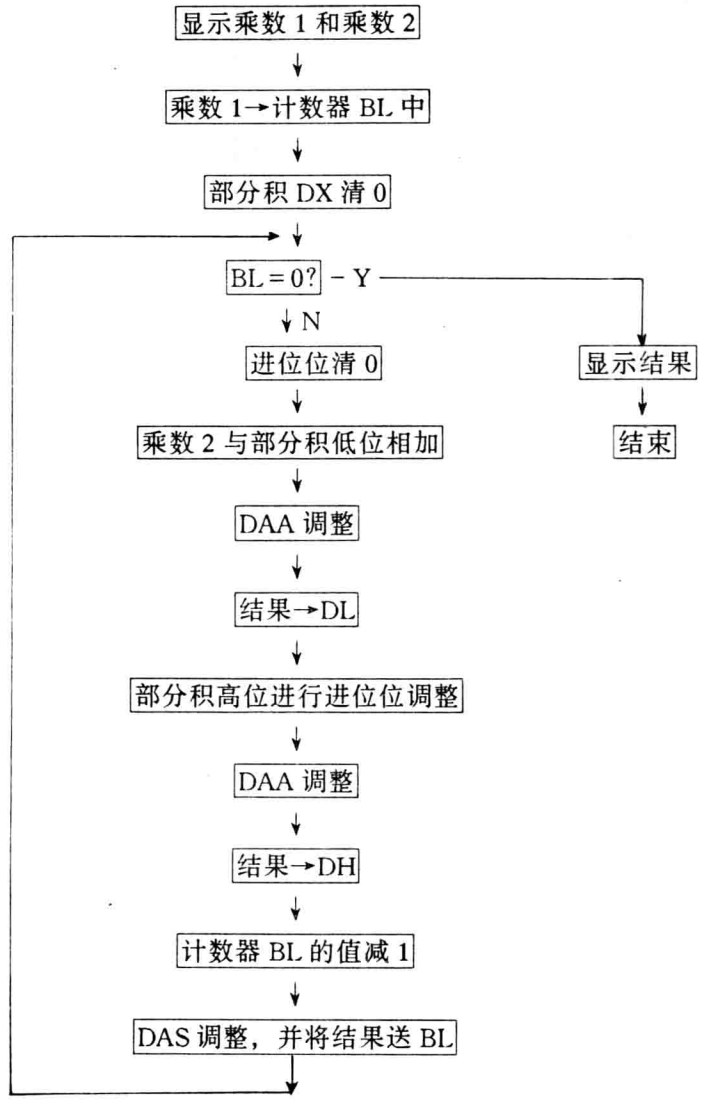
掌握用的组合的 BCD 码表示数据，并熟悉怎样实现组合 BCD 码乘法运算

二、实验内容

实现 BCD 码的乘法，要求被乘数和乘数以组合的 BCD 码形式存放，各占一个内存单元。乘积存放在另外两个内存单元中。

由于没有组合的 BCD 码乘法指令，程序中采用将乘数 1 作为计数器，累加另一个乘数的方法得到计算结果。

三、程序框图



四、程序清单

```
DISPL   MACRO
        ADD     DL, 3OH
        MOV     AH, 02
        INT     21H
ENDM

CRLF    MACRO
        MOV     DL, 0DH
        MOV     AH, 02
        INT     21H
        MOV     DL, 0AH
        MOV     AH, 02
        INT     21H
ENDM
DATA    SEGMENT
DATA1   DB      01001000B       ; 48H
DATA2   DB      01110101B       ; 75H
RESULT  DW      ?
DATA    ENDS
STACK   SEGMENT
STA     DB      20 DUP (?)
TOP     EQU     LENGTH STA
STACK   ENDS
CODE    SEGMENT
ASSUME  CS: CODE, DS: DATA, SS: STACK, ES: DATA
START:  MOV     AX, DATA
        MOV     DS, AX
        MOV     AX, STACK
        MOV     SS, AX
        MOV     AX, TOP
        MOV     SP, AX
        MOV     DI, OFFSET RESULT; 指向结果存储区
        MOV     SI, OFFSET DATA1 ; 指向乘数 1
        MOV     AH, [SI]
        MOV     CL, 04
        SHR     AH, CL          ; 显示乘数 1 高位字节
        MOV     DL, AH
        DISPL
        MOV     AL, [SI]
```

```
        MOV     BL, AL              ; 显示乘数 1 低位字节
        AND     AL, 00001111B
        MOV     DL, AL
        DISPL
        CRLF                        ; 回车换行
        MOV     SI, OFFSET DATA2 ; 指向乘数 2
        MOV     AH, [SI]
        MOV     CL, 04
        SHR     AH, CL              ; 显示乘数 2 高位字节
        MOV     DL, AH
        DISPL
        MOV     AL, [SI]
        AND     AL, 00001111B
        MOV     DL, AL              ; 显示乘数 2 低位字节
        DISPL
        CRLF                        ; 回车换行
        XOR     DX, DX              ; DX（部分积）清 0
SL:     CMP     BL, 0
        JZ      S2                  ; 乘数 1 的值已减为 0 否?
        CLC
        MOV     AL, [SI]
        ADC     AL, DL              ; 乘数 2 加部分积低位
        DAA                         ; 组合的 BCD 码调整
        MOV     DL, AL              ; 结果送部分积高位
        MOV     AL, 00H
        ADC     AL, DH              ; 往部分积高位送进位
        DAA                         ; 组合 BCD 码调整
        MOV     DH, AL              ; 结果送部分积高位
        MOV     AL, BL
        DEC     AL                  ; 乘数 1 的值减 1
        DAS                         ; 组合 BCD 码减法调整
        MOV     BL, AL              ; 结果送 BL
        JMP     S1
S2:     MOV     [RESULT], DX; 结果送存储器
        MOV     SI, DX
        MOV     CL, 4
        SHR     DH, CL
        MOV     DL, DH
        DISPL                       ; 显示结果高 2 位
        MOV     DX, SI
```