第八讲　程序设计的其它方法与技术(P150)

如何提高编程的效率和质量，简化程序设计的工作?

（1）　串操作指令；

（2）　宏指令的定义与调用；

（3）　模块程序设计方法及连接技术。

一、串操作指令

由于许多待解决的问题都属于对 ASCII 字符串或数组的操作。该类指令共有五条：

（1）传送串指令　　MOVS　OPD，OPS

（2）串比较指令　　CMPS　OPD，OPS

（3）搜索串指令　　SCAS　OPD

（4）取数指令　　LODS　OPS

（5）存数指令　　STOS　 OPD

1．串传送指令

格式：**MOVS　OPD，OPS**

实际 → { **MOVSD**　双字传送
　　MOVSW　字传送
　　MOVSB　字节传送

例：BUF_ES 是 ES 段内字节变量,BUF_DS 是 DS 段内字节变量，

MOVS BUF_ES,BUF_DS 将汇编成 MOVSB

功能：

（1）（*DS*：[SI] /[ESI]）→ *ES*：[DI]/ [EDI]

即将 SI/ESI 所指的源串中的一个字节或字、双字中的数传送到 DI/EDI 所指的目的单元中。隐含实现了存储器到存储器的寻址方式。

该指令的运行不影响标志位。

（DS 为源缺省段寄存器，可以用段前缀修改成 ES 等，但目的段 ES 不能修改）

（2）修改串指针，使之指向下一元素。修改方式为：

i.当 DF=0 （CLD），

（SI）/（ESI）+1/2/4

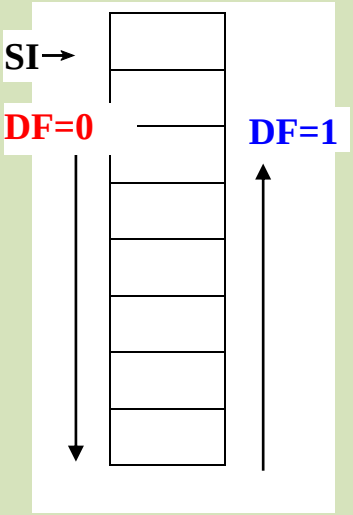
（DI）/（EDI）+1/2/4

ii.当 DF=1 （STD），

（SI）/（ESI）-1/2/4

（DI）/（EDI）-1/2/4

（源和目的指针同时增加或减少）



说明： 该指令可带重复前缀：REP 即当（CX）/(ECX)≠0 时，连续传送，直至（CX）/(ECX)=0。

增加操作：修改循环变量（CX）/(ECX)-1→CX/ECX。

例 1：将 BUF1 中的字符串传送到以 BUF2 为首址的字节缓冲区中。

```
DATA    SEGMENT   USE16
BUF1    DB    'SDEIUcnu13sa4'    ; 输出缓冲区 BUF1
COUNT   EQU   $-BUF1             ;输出缓冲区中的字符个数
BUF2    DB    COUNT  DUP(0)      ; 输入缓冲区 BUF2
DATA    ENDS
STACK   SEGMENT  STACK USE16
        DB  200  DUP(0)
STACK   ENDS
CODE    SEGMENT  USE16
        ASSUME  DS: DATA, ES: DATA, CS: CODE, SS: STACK
START: MOV  AX, DATA
       MOV  DS, AX          } 当前数据段和当前附加数据段重合
```

```
        MOV   ES，AX
        LEA   SI，BUF1          ；源串首址 → SI
        LEA   DI，  BUF2        ；目的串首址 → DI
        MOV   CX，COUNT         ；串长度 → CX
        REP     MOVSB           ；重复传送直至 CX=0
        MOV   AH，4CH
        INT    21H
CODE    ENDS
        END   START
```

其中，“REP MOVSB”语句代替了以下程序段：

```
      P：   JCXZ  EXIT
            MOV  AL，[SI]
            MOV  ES：[DI]，AL
            INC    SI
            INC    DI
            DEC CX
            JMP  P
      EXIT：
```

由于 DF=0 为常用的默认状态，因此在使用串指令前一般可以不必使用 CLD 指令。

2．串比较指令

格式： **CMPS OPD，OPS** 或

CMPSD	双字串比较
CMPSW	字串比较
CMPSB	字节串比较

功能：（1）（DS：[SI]/[ESI]）-（ES：[DI]/[EDI]），并根据相减结果设标志位，但结果并不保存，即源串、目的串内容并不改变。

（注意是源-目的）

（2）修改串指针，使之指向下一元素。

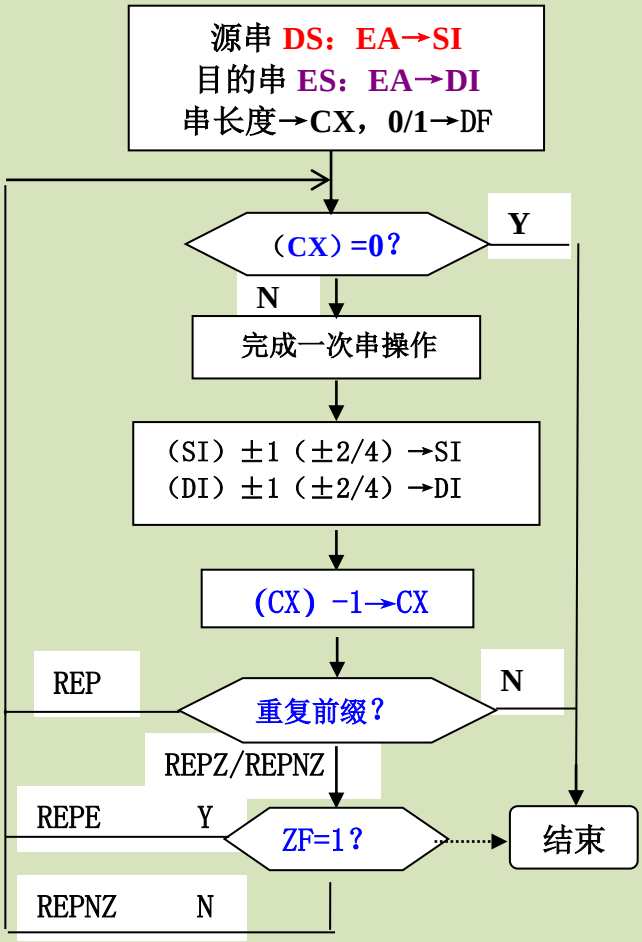
a. REPE/REPZ　当（CX）≠0 时，如两串相等（ZF=1）继续比较，不相等跳出循环；（找到两串的不同点）

b. REPNE/REPNZ　当（CX）≠0 时，如两串不相等（ZF=0）继续比较，相等跳出循环。（找到两串的相同点）

```
；A 串和 B 串比较
REPZ　CMPSB
；相等继续，不等退出
```

小结：

16 位串操作指令的工作流程可用如下框图表示：



注：兰色部分是有重复前缀时才执行的操作

二、 宏功能程序设计

对于程序中的重复部分，定义成子程序有时间和空间上的开销。为此，现在的机器均提供了宏汇编语言，包括：宏指令的定义与调用，重复汇编与条件汇编等。本节主要介绍宏指令的定义与调用。

宏指令的使用有以下三个步骤：

（1） 宏定义

（2） 宏调用

（3） 宏扩展 （汇编程序完成）

（一）宏定义

宏指令在使用之前要先定义

定义格式：宏指令名 MACRO [形式参数[，形式参数]]

宏体

ENDM

（注意与段、子程序定义格式的相似和不同）

例如：某一程序中要多次使用 DOS 9 号功能调用：

```
        ┇
    LEA   DX，BUF1
    MOV   AH，9
    INT    21H
        ┇
  LEA   DX，IN_BUF
```

```
        MOV   AH，9
        INT     21H
```

它们之间的差别仅在于输出缓冲区的首址不一样。如果将该首址定义成形参，就可将其写成宏定义：

```
WRITE     MACRO    A
          LEA   DX，A      ；A 为形参，代表输出缓冲区首址
          MOV   AH，9
          INT   21H
          ENDM
            ┇
WRITE   BUF1              ；宏调用
            ┇
WRITE   IN_BUF            ；宏调用
```

说明：（1）宏名字尽量不要与其它变量、标号、保留字同名。

（2）形参可有可无，个数不限，但总字符长度不超过 132 个，各参数之间用逗号隔开；（宏名字、形参名字均以字母开始，单个串长不超过 31 个。否则按重名处理，不一定报错，因此，错误难查）

（3）ENDM 与 MACRO 必须成对出现；

（4）宏指令必须先定义后调用（与子程序不同），因为它是在汇编期间处理的。

（5）宏定义体中可以调用先定义的宏指令，即宏的嵌套。（但不要嵌套自己）

（二）宏调用

宏指令一旦完成定义后，就可在程序中调用（即宏指令名在源程序中出现）。

调用格式：宏指令名 [实在参数[，实在参数]]

说明：（1）宏指令名要与原宏定义的名字一致；

（2）实参与形参应按位置关系一 一对应：

a. 实参个数多于形参，多余实参被忽略；

b. 实参个数小于形参（或逗号之间空，如：AX,，BX），缺少的实参被处理为空白（没有字符）或 0。要避免。

宏汇编程序处理宏指令方式是宏扩展。

将宏体复制到宏调用处，并用实参按位置顺序替换形参。这一过程称宏扩展。

宏扩展后的程序可以在汇编列表文件（.lst）中看到，宏扩展后的宏体语句均在前面冠以“+”（或 1）以示与其它语句的区别。

例：

```
+       LEA   DX，BUF1
+       MOV   AH，9
+       INT    21H
          ┇
+       LEA    DX，IN_BUF
+       MOV   AH，9
+       INT     21H
```

还可以将 DOS 9 号与 10 号调用写成一个宏定义：

```
IO   MACRO   A，B
     LEA     DX，A          ；形参 A 代表输出 / 输出缓冲区首址
     MOV     AH，B          ；形参 B 代表 DOS 调用号
     INT     21H
     ENDM
```

（三）宏库的使用

宏库的定义：

对于经常使用的宏定义，用户可将它们集中在一起，建成宏库供自己或别人随时调用。由于宏库为文本文件，可用一般编辑程序建立或修改，文件名也可由用户任意指定。

例如：利用编辑程序建立一个文件名为 **IO_M．LIB** 的宏库。

宏库的调用：

语句格式： **INCLUDE** 文本文件名

功能：将指定的文本文件从本行起加入汇编，直到该文本文件的最后一行汇编完后，再继续汇编 **INCLUDE** 后面的语句。

宏库 **IO_M．LIB** 的内容：

```
READ    MACRO   A
        LEA     DX,   A
        MOV     AH,   10
        INT     21H
        ENDM
WRITE   MACRO   A
        LEA     DX,   A
        MOV     AH,   9
        INT     21H
        ENDM
OUT1    MACRO   A
        MOV     DL,   A
        MOV     AH,   2
        INT     21H
        ENDM
```

```
CRLF      MACRO
          OUT1    0AH              ；宏的嵌套
          OUT1    0DH
          ENDM
STACK0    MACRO   A
          STACK   SEGMENT PARA STACK 'STACK'
          DB      A
          STACK   ENDS
          ENDM
```

宏库调用：（程序功能：将输入的字符串重新显示并将小写字母转成大写）

```
          INCLUDE   IO_M．LIB ；将指定的宏库加入本程序一起汇编
.386
 DATA     SEGMENT   USE16
BUF       DB   80，0，80   DUP(0)       ；10 号功能调用缓冲区。

STACK0〈200   DUP（0)〉          ；调用宏定义 STACK0 定义堆栈段
CODE      SEGMENT   USE16
          ASSUME   DS：DATA，SS：STACK，CS：CODE
START：   MOV   AX，DATA
          MOV   DS，AX
          READ   BUF              ；输入一字符串→BUF 缓冲区
          CRLF                    ；输出回车换行
          LEA    SI，  BUF＋2
          MOV   CL，BUF＋1
          MOV   CH，0
          CLD
Y1：      LODSB                   ；从输入串中取一字符→AL
          CMP    AL，′ a′
          JB      Y2
          CMP    AL，′ z′
          JA      Y2              ；该字符不为小写字母，则转 Y2
          SUB    AL，20H          ；将小写转换为大写字母的 ASCII 码
Y2：      OUT1    AL              ；在显示器上显示一个字符
          LOOP   Y1
          CRLF                     ；输出回车换行
          MOV    AH，4CH
```

```
        INT     21H
CODE    ENDS
        END   START
```

（四）宏指令与子程序的比较

(p174)

1、处理（使用）的时候不同；

2、处理（调用）的方式不同；

3、目标程序的长度不同；

4、执行速度不同；

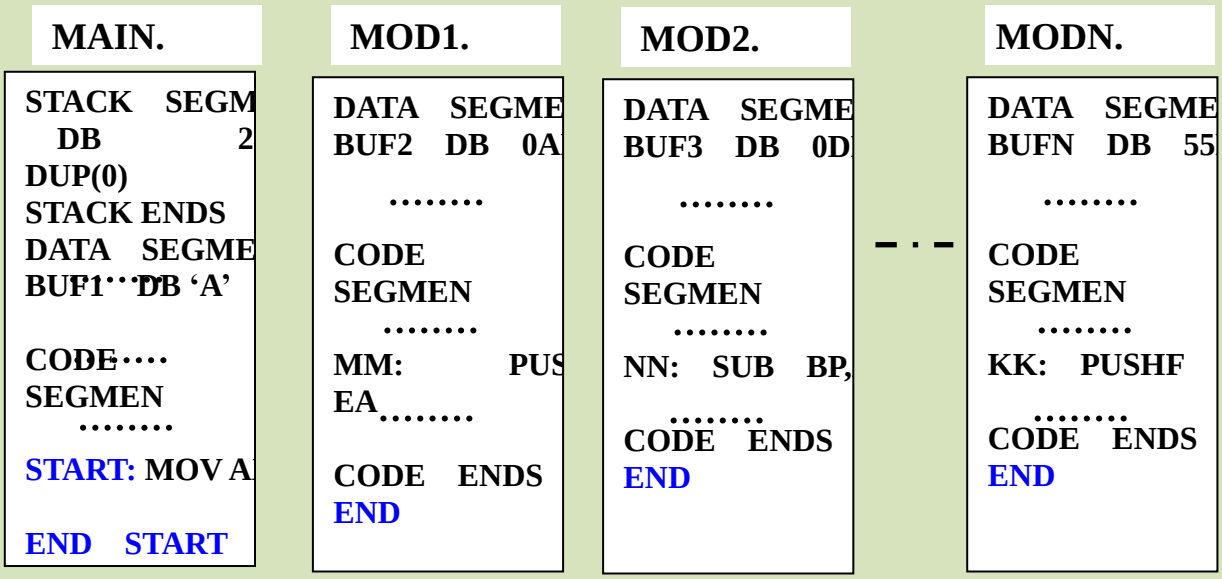
5、参数传递的方式不同。

三、 模块化程序设计

1、什么是模块化程序

一个以 END 语句为结束的源程序称为一个模块。一个模块是一个独立的汇编源文件，一个模块汇编后生成一个目标文件（*.obj），或称目标模块。

模块化程序是由一个以“END 标号”结束的主模块，和几个以“END”结束的子模块组成。



2、模块化程序设计的主要问题

（1）各模块内定义的段之间的关系；

（2）本模块内的变量和子程序与其它模块的关系；

（3）如何将各模块合成一个程序；

（4）由模块构成的库生成与使用。

（一）如何将各模块合成一个程序（7.2.1 节，P285）

1. 分别建立汇编源文件：MAIN.ASM、MOD1.ASM、MOD2.ASM 。。。 MODN.ASM

2. 分别汇编成 OBJ：

D：\MASM　MAIN;　　得到 MAIN.OBJ

D：\MASM　MOD1;　　得到 MOD1.OBJ

。。。

3. 连接成一个可执行文件：

D：\LINK　MAIN + MOD1 + MOD2 + 。。。　+MODN;　　得到 MAIN.EXE

4. 运行可执行文件：

D：\MAIN

（二）各模块内定义的段之间的关系

段定义语句：段名 SEGMENT [使用类型][定位方式][组合方式][‘类别’]

⋮

段名　ENDS

1. ‘类别’

‘类别’是用单引号括起来的字符串，该字符串可以是任何合法的名称。连接程序在进行连接处理时，将‘类别’相同的段（它们不一定同名），按出现的先后次序连续存放（但仍是不同名的段），且每段都有自己的首址。

如：
```
A1   SEGMENT   ‘DATA’
B1   SEGMENT   ‘CODE’
C1   SEGMENT   ‘TO’
D1   SEGMENT   ‘DATA’
E1   SEGMENT   ‘TO’
```

经连接程序连接后，存放的顺序为：

```
A1   SEGMENT   ‘DATA’
D1   SEGMENT   ‘DATA’
B1   SEGMENT   ‘CODE’
C1   SEGMENT   ‘TO’
E1   SEGMENT   ‘TO’
```

典型类别有：‘DATA’、‘CODE’、‘STACK’

2. 组合方式

组合方式向连接程序提供了本段同其它段的组合关系，主要有：

NONE（不选择）：本段与其它段不发生任何逻辑上的联系，自己有自己的段首址。（默认的方式）

PUBLIC： 表示应将本段与其它模块中的同名、同‘类别’段按各模块连接的顺序相邻地连接在一起，组成一个物理段， 16 位段

大小不超过 64K。（有一项不同就不合并）

合并的好处：以数据段为例，当组合成一个段后，在子模块中就不用再对 DS 送首址，方便而又有效。

STACK：　仅对堆栈段，功能同 PUBLIC。（有多个段定义但又不满足合并条件时，仅一个段作为实际堆栈段，只选择最后一个堆栈段的定义信息）

COMMON：　表示本段与同名、同‘类别’的其它段应具有相同段首址，即相覆盖。长度取决于最长的 COMMON 段。

“AT 表达式”：表示该段应放在表达式所指定的绝对地址上。

“MEMORY”；表示该段应放所有段之上（最高地址上）。

根据程序的需要合理选用组合方式：

（1）定义的段的数量多造成段（寄存器）的频繁转换，要避免。

（2）复杂的程序（代码长、处理的数据多）要通过增加段获得空间。

（三） 通信方式

（本模块内的变量和子程序与其它模块的关系）

局部符号：仅在定义自己的模块中被访问的符号为局部符号。

公共符号：不仅被定义自己的模块访问，还要供其它模块访问的符号为公共符号。要用 PUBLIC 伪指令说明。其格式为：

PUBLIC　符号[，符号，……]

外部符号：只在模块内访问而不在该模块内定义的符号为外部符号。要用 EXTRN/EXTERN 伪指令说明。其格式为：

EXTRN　符号：类型[，符号：类型，……]

例：已知主模块 WAN1 和子模块 WAN2 的定义如下，请指明各模块中的局部符号、公共符号和外部符号。（说明将子程序改写成子模块后的变化）

各模块的程序设计方法无特殊之处。

```
WAN1 . ASM
        EXTRN   WAN2：NEAR
        PUBLIC    DIV0，COUNT
DATA    SEGMENT  PARA  PUBLIC  'A1'
BEG     EQU     1
COUNT   DW      50
DIVO     DW      O
DATA    ENDS
STACK    SEGMENT     PARA     STACK
 'STACK'
        DB   200   DUP（0）
STACK   ENDS
CODE    SEGMENT  PARA  PUBLIC  'B1'
ASSUME   DS：DATA，SS：STACK，CS：CODE
START：   MOV   AX，DATA
        MOV   DS，AX
        MOV   BX，BEG
        MOV   CX，COUNT
        CALL   WAN2
        MOV   AH，4CH
        1NT    21H
CODE    ENDS
        END   START
```

```
WAN2 . ASM
        EXTRN   D1VO：WORD，COUNT：
   WORD
        PUBLIC   WAN2
DATA  SEGMENT  PARA  PUBLIC 'A1'
SUM     DW    0
DATA  ENDS
STACK    SEGMENT     PARA     STACK
 'STACK'
        DB   200   DUP（0）
STACK   ENDS
CODE     SEGMENT  PARA  PUBLIC
 'B1'
ASSUME   DS：DATA，CS：CODE，SS：
STACK
WAN2    PROC
        MOV   AX，0
NEXT：   ADD    AX，BX
        ADD    BX，2
        LOOP   NEXT
        MOV   SUM，AX
        MOV   DX，0
        DIV    COUNT
        MOV   DIVO，AX
        RET
WAN2    ENDP
COOE    ENDS
        END
```

（四） 由模块构成的库生成与使用。

1、特点

（1）由模块构成的库是目标代码不是文本源程序。(不同于宏库)

（2）由模块构成的库是通用的子程序目标代码库。

2、建库方法（7.2.2节，P286）

（1）建立以子程序为主体的子模块源程序（*.ASM），并汇编成目标文件（*.OBJ）。

（2）用库管理程序 LIB.EXE 建立和维护库文件 ？.LIB 。

可以将 *.OBJ 文件转成一个新的库文件 ？.LIB；也可以将 *.OBJ 文件追加到原有的一个库文件中。

3、库的使用

源程序使用库中的子程序就象使用子模块中的子程序一样。

连接程序(LINK)可以将任何语言生成的符合统一格式的OBJ、LIB中的代码连到一起。

第五章　　总　　结

1、串操作指令

（1）　串传送指令：MOVS　OPD，OPS

（2）　串比较指令：CMPS　OPD，OPS

（3）　串查找指令：SCAS　OPD

（4） 从源串中取数指令：LODS OPS

（5） 从目的串中送数指令 STOS OPD

2、宏功能程序设计

宏指令的定义与调用方式;

宏定义与子程序的区别。宏库。

3、模块化程序设计

组合方式;

局部符号、公共符号、外部符号;

一般程序改写成子程序、子程序改写成独立子模块的方法;

模块化程序设计的方法。

*