80X86汇编语言程序设计

80X86 Assembly Language Programming

第5章 程序设计的其他方法和技术

金良海 教授

计算机学院 医学图像信息研究中心

Email: lianghaijin@hust.edu.cn

本章的学习内容

本章学习汇编语言的高级程序设计技术：

(1)串操作指令的使用格式及功能；

(2)宏指令的定义与调用方式；

(3)模块程序设计方法及连接技术。

目标：提高编程效率和质量，简化程序设计工作。

本章的学习重点

(1)串操作指令MOVS、CMPS、SCAS的使用格式及功能；

(2)简单宏指令的定义与调用方式；

(3)模块程序设计的方法。

本章学习的难点

(1) MOVS与MOV、CMPS与CMP功能上的差别及串操作指令的正确使用方法；

(2)宏指令的定义与调用方式；

(3)模块之间的组合、定位及通信方式；

(4)模块化程序设计技术。

5.1 字符串操作

华中科技大学

1.串传送指令

MOVS、MOVSB、MOVSW、MOVSD

2.串比较指令

CMPS、CMPSB、CMPSW、CMPSD

3.串搜索指令

SCAS、SCASB、SCASW、SCASD

4.从源串中取数

LODS、LODSB、LODSW、LODSD

5.向目的串中存数

STOS、STOSB、STOSW、STOSD

5.1.1 串操作指令简介

源串指针： DS:SI / ESI 源串在当前数据段
目的串指针： ES:DI / EDI 目的串在附加数据段
重复计数器： CX / ECX
中间寄存器： AL / AX / EAX

传送/比较方向：
DF=0，SI / ESI，DI / EDI自动增量（加1,2,4）
DF=1，SI / ESI，DI / EDI自动减量（减1,2,4 ）

重复前缀：REP （CX/ECX）!= 0 时重复执行
REPE（CX/ECX）!= 0 && ZF=1 时重复执行
REPNE（CX/ECX）!= 0 && ZF=0 时重复执行
（对比 P115的 LOOP, LOOPE, LOOPNE)

5.1.2 串操作指令

1. MOVS, MOVSB, MOVSW, MOVSD
2. CMPS, CMPSB, CMPSW, CMPSD
3. SCAS, SCASB, SCASW, SCASD
4. LODS, LODSB, LODSW, LODSD
5. STOSB, STOSW, STOSD

(5.1.2) 1.串传送指令(1)

$$\{REP\} + \begin{cases} MOVS + OPD + OPS \\ \begin{cases} MOVSB \\ MOVSW \\ MOVSD \end{cases} \end{cases}$$

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

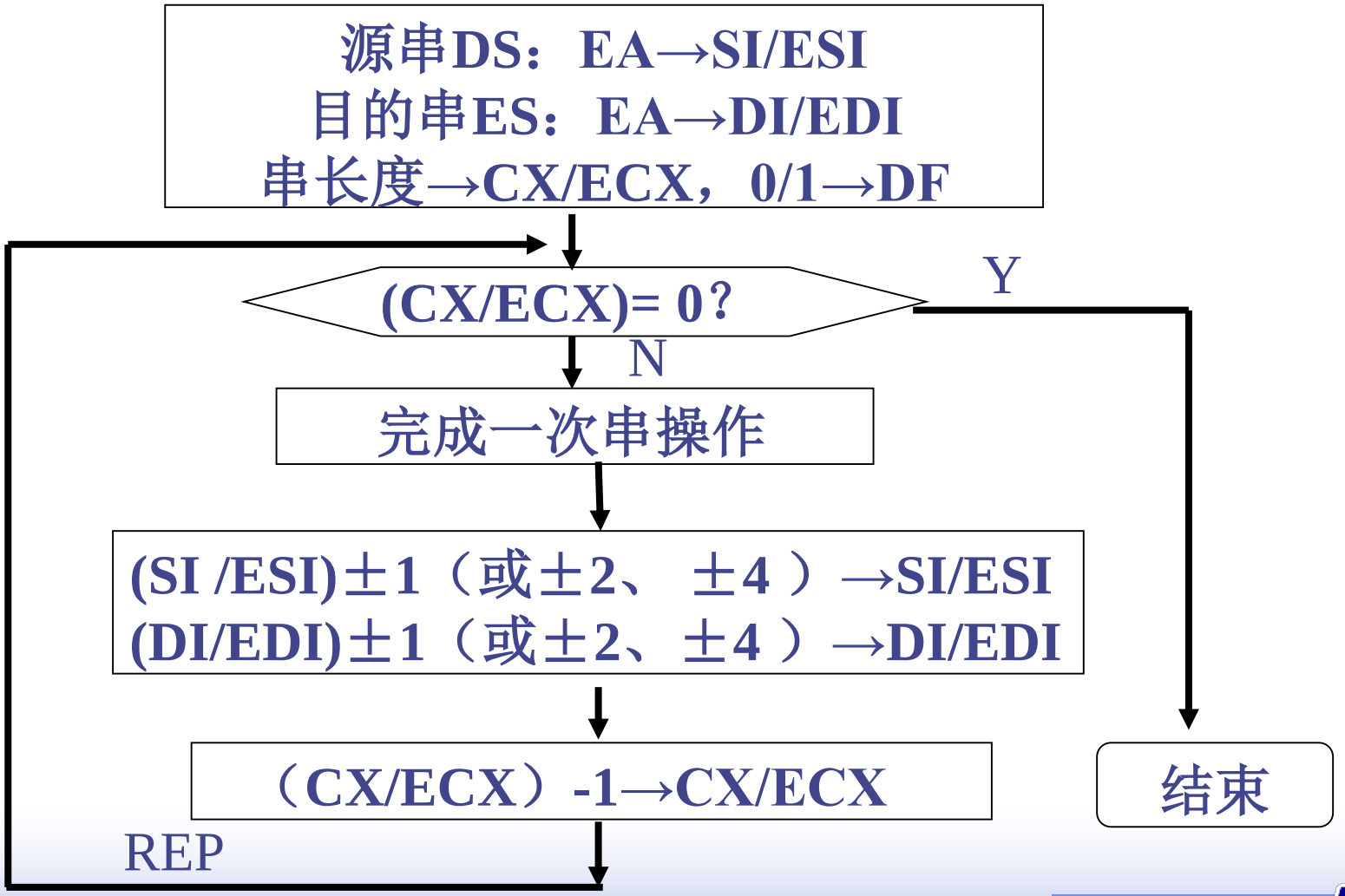
(2) 若 DF=0, 则 (SI/ESI)和(DI/EDI)增1 (字节)

或增2（字操作）、或增4（双字操作）

若DF=1,则(SI)和(DI)减1、或2、或4。

(3) MOVS 指令不影响标志位。

(5.1.2) 1.串传送指令(2)


源串DS：EA→SI/ESI
目的串ES：EA→DI/EDI
串长度→CX/ECX，0/1→DF
(CX/ECX)= 0？
Y
N
完成一次串操作
(SI /ESI)±1（或±2、 ±4 ）→SI/ESI
(DI/EDI)±1（或±2、 ±4 ）→DI/EDI
（CX/ECX）-1→CX/ECX
REP
结束

(5.1.2) 1.串传送指令(3)

```
stack   segment  stack
        db 100 dup(?)
stack   ends
data    segment
str1    db ‘abcd123’
count   db  $-str1
str2    db  count dup(?)
data    ends
code    segment
        assume cs:code,
               ds:data,
               es:data
start:  mov  ax, data
        mov  ds, ax
        mov  es, ax
        lea   si,  str1
        lea   di, str2
        mov  cx, count
        cld
        rep   movsb
        mov  ax, 4c00h
        int   21h
code   ends
        end   start
```

(5.1.2) 2.串比较指令(1)

$$\begin{Bmatrix}\text{REP}\\ \text{REPE}\\ \text{REPNE}\end{Bmatrix} + \begin{cases}\text{CMPS} + \text{OPD} + \text{OPS}\\ \begin{Bmatrix}\text{CMPSB}\\ \text{CMPSW}\\ \text{CMPSD}\end{Bmatrix}\end{cases}$$

注意它与比较指令CMP的区别

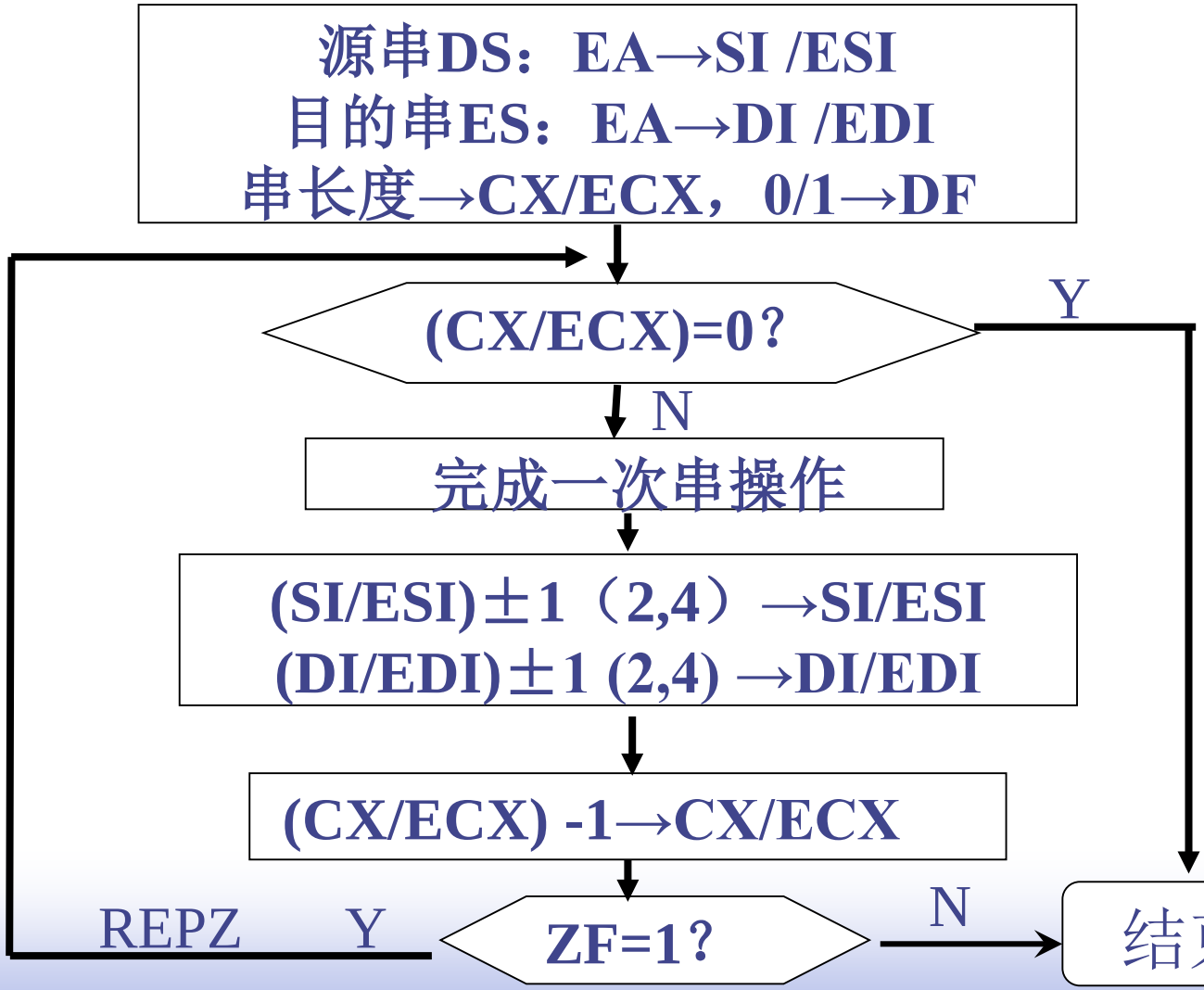
功能：(1) ([SI / ESI]) – ([DI / EDI]) , 设置标志位

(2) 若 DF=0, 则 (SI /ESI)和(DI /EDI)

增1、2、4（字节、字、双字操作）

若DF=1,则(SI/ESI)和(DI/EDI)减1或2、4

(5.1.2) 2.串比较指令(2)



问：循环比较指令结束时，应该用（CX)=0 还是ZF=0作为串相等的判断条件？

(5.1.2) 2.串比较指令(3)

注意：

- ZF 是根据串比较指令设置的，而不是最后的CX减1。
- 先执行比较，后修改 SI, DI
- 是否相等，要用ZF来判断
- 若串不等，SI 指向第一个不相等的字符的下一个字符。

(5.1.2) 3.串搜索指令(1)

$$\begin{Bmatrix} \text{REP} \\ \text{REPE} \\ \text{REPNE} \end{Bmatrix} + \begin{Bmatrix} \text{SCAS} + \text{OPD} + \text{OPS} \\ \begin{Bmatrix} \text{SCASB} \\ \text{SCASW} \\ \text{SCASD} \end{Bmatrix} \end{Bmatrix}$$

功能： (1) (AL/AX/EAX) – (ES: [DI/EDI]) ，设置标志位

(2) 若 DF=0, 则 (DI / EDI)

增1,2,4 (字节, 字, 双字)

若DF=1, 则(DI / EDI)减1,或2,或4

(5.1.2) 3.串搜索指令(2)

例：数据段DATA的缓冲区BUF中存放有一个长度为COUNT的字符串. (1)查找第一个非空格字符;(2)统计BUF中非空格字符的个数。

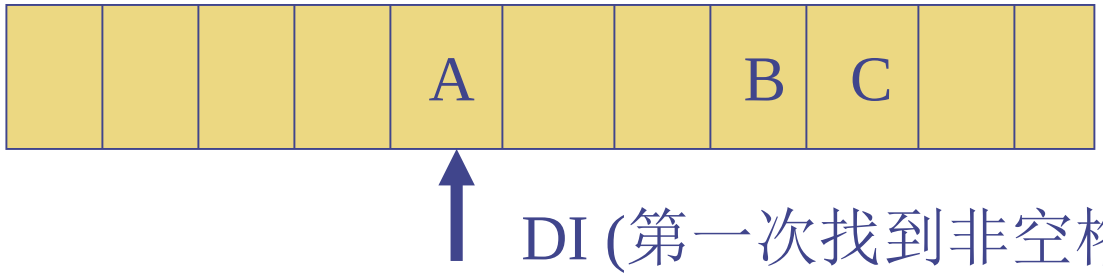
分析：一个串的开头和中间也许有很多空格，若首字符是空格，就要判断下一个字符是否为空格。依此类推，直到第一个非空格字符出现。

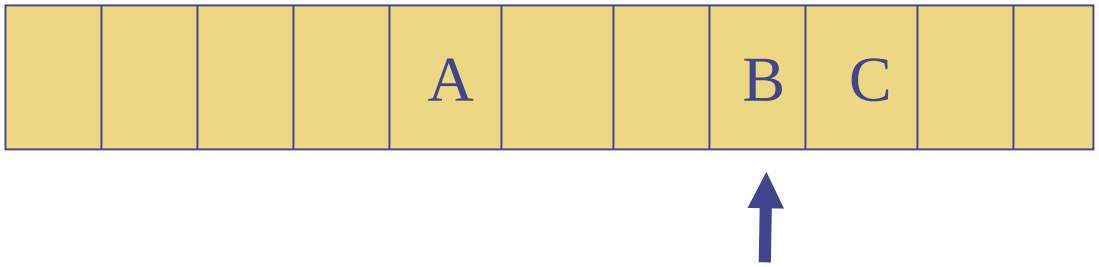
```
MOV    AX, DATA
MOV    ES,  AX
MOV    DI,   OFFSET BUF
MOV    CX,  COUNT
MOV    AL, ‘ ’
REPZ   SCASB
```

循环条件 (CX)!=0 且 ZF=1

(5.1.2) 3.串搜索指令(3)

例： 统计显示一个串中空格字符的个数。





（讨论程序结束的条件 (CX) ? ZF ）

(5.1.2) 3.串搜索指令(4)

```
        MOV   AX,  DATA
        MOV   ES,  AX
        MOV   DI,  OFFSET BUF
        MOV   CX,  COUNT
        MOV   BX,  0
        MOV   AL,  ' '
NEXT:   MOV   DX, CX
        REPZ  SCASB
        JZ    NEXT1
        INC   CX
        DEC   DI
NEXT1:  SUB   DX, CX
        ADD   BX, DX
        REPNZ  SCASB
        JNZ    EXIT
        INC    CX
        DEC    DI
        MOV    DX, COUNT
        SUB    DX, CX
        ADD    BX, DX
        JMP    NEXT
        ;
EXIT:   …
```

BX = 空格字符的个数

(5.1.2) 4.从源串中取数指令

语句格式：

LODS　OPS

LODSB

LODSW

LODSD

功能：　**(1) (DS:[SI/ESI]) → AL / AX / EAX**

(2) 若 DF=0, (SI /ESI) 如何变化?

若DF=1,则(SI /ESI)如何变化?

(3) 不影响标志位

(5.1.2) 5.往目的串中存数指令

语句格式：

STOS　OPD

STOSB

STOSW

STOSD

功能：　(1) (AL / AX /EAX) → ES:[DI/EDI]

(2) 若 DF=0，(DI/EDI)如何变化？

若DF=1，(DI/EDI) 如何变化？

(3) 不影响标志位

5.1.3 串操作指令综合实例

输入一个串，判断是否为一个命令列表中的命令，若是，则调用相应的处理程序。否则，显示出错提示。(要求能够连续处理命令，直到输入空串)

P159

5.2 宏功能程序设计

1. 宏定义
2. 宏调用
3. 宏定义和宏调用中的参数
4. 宏指令与子程序的比较

5.2 宏功能程序设计


```
inline int ABC(int buf[] , int N)
{
   int val = buf[0];
   for(int k = 1; k < N; k++)
   {
       if(val < buf[k]) val = buf[k];
   }
   return val;
}
```

```
void main()
{
  int  A[] = { 11, 2, 44, 5 };
  int  B[] = { -2, 0, 95 };
  int  v1 = ABC(A, 4);
  int  v2 = ABC(B, 3);
}
```

5.2.1 宏定义 (1)

宏指令名 MACRO [形式参数 [,形式参数]]
宏体
ENDM

例：输出以 A 为首址的字符串

```
WRITE   MACRO  A
        LEA    DX,  A
        MOV    AH,  9
        INT    21H
        ENDM
```

特别注意：
ENDM前有什么？

5.2.1 宏定义 (2)

将DOS 9号与10号调用写成一个宏定义：

```
IO   MACRO  A，B
     LEA    DX，A
     MOV    AH，B
     INT    21H
     ENDM
```

；形参A代表输出／输出缓冲区首址
；形参B代表DOS调用号

5.2.1 宏定义 (3)

宏定义中注意的问题：

(1)宏段的结束处，没有宏指令名

(2)形参可有可无，若有多个，之间以逗号分隔。

(3)ENDM与MACRO必须成对出现

(4)宏名字可以与其它变量、标号、保留字同名，汇编程序在处理时，**宏名字优先级最高**，利用这一特点，可设计新的指令系统。

(5)宏指令在使用之前要先定义，与子程序可写在调用指令后不同。

5.2.2 宏调用 (1)

调用格式：宏指令名 [实在参数[，实在参数]]

（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，缺少的实参被处理为空白（没有字符）。

5.2.2 宏调用 (2)

```
WRITE   MACRO  A
        LEA   DX,  A
        MOV   AH,  9
        INT   21H
        ENDM
DATA  SEGMENT
BUF1  DB  ‘good $’
BUF2  DB  ‘hello $’
DATA  ENDS

      ……
      WRITE  BUF1

       ……
      WRITE  BUF2
```

5.2.2 宏调用 (3)

宏指令的汇编过程

例：前例中的宏调用经汇编程序扩展后的形式

```
        + LEA   DX, BUF1
        + MOV   AH, 9
        + INT   21H
      ……
        + LEA   DX, BUF2
        + MOV   AH, 9
        + INT   21H
```

5.2.3 宏定义与宏调用中的参数(1)

一．带间隔符的参数

在宏调用中，一个实参可以是一串带间隔符的字符串，为不致混淆，应该用尖括号将它们括起来，说明为一个参数。

```
STACK0   MACRO    A
STACK    SEGEMNT  STACK
         DB  A
STACK    ENDS
         ENDM
STACK0   < 10, 20, 500 DUP (0) >
```

5.2.3 宏定义与宏调用中的参数(2)

二．宏体中的变量与标号

求从A开始连续B个奇数（或偶数）的和 AX.

```
SUM   MACRO  A , B
      MOV    BX , A
      MOV    CX , B
      MOV    AX , 0
NEXT: ADD    AX , BX
      ADD    BX , 2
      LOOP   NEXT
      ENDM
```

5.2.3 宏定义与宏调用中的参数(3)

若有两次调用：

```
SUM  1，50
  ┆
SUM  11，20
```

宏扩展后，就会引起标号重复定义的错误 。

解决办法：

（1）将标号定义成形参，每次调用均用实参代替。但该方式会造成参数过多，给编程带来麻烦；

（2）使用伪指令LOCAL，让机器自动生成不同标号。

5.2.3 宏定义与宏调用中的参数(4)

格式：LOCAL 形式参数[，形式参数]

功能：宏扩展时，汇编程序自动为形式参数生成特殊顺序号，范围为 ??0000 - ??FFFFH。

```
SUM     MACRO  A, B
        LOCAL  NEXT
        MOV    BX，A
        MOV    CX，B
        MOV    AX，0
NEXT:   ADD    AX，BX
        ADD    BX，2
        LOOP   NEXT
        ENDM
```

注意：该语句只能作宏体中的第一个语句。

5.2.3 宏定义与宏调用中的参数(5)

```
+       MOV  BX，1
+       MOV  CX，50
+       MOV  AX，0
+       MOV  AX，0
+ ??0000：ADD  AX，BX   ；标号NEXT转换为??0000
+       ADD  BX，2
+       LOOP ??0000
          ┇
+       MOV  BX，10
+       MOV  CX，20
+       MOV  AX，0
+ ??0001：ADD  AX，BX   ；标号NEXT转换为??0001
+       ADD  BX，2
+       LOOP  ??0001
```

5.2.3 宏定义与宏调用中的参数(6)

练习：

设计宏指令 WORD_ADD AA,BB,CC

功能：字单元AA与字单元BB中的内容相加后，
结果放入字单元CC中。

要求：调用宏指令前后，通用寄存器中的值不变。

5.2.3 宏定义与宏调用中的参数(7)

- 在宏定义中，&A 表示A是一个子字符串，需放置在相应的位置；

```
SHIFT  MACRO A,B,C
       MOV   CL, A
       S&B   C, CL
       ENDM
```

```
SHIFT  4, AL, AX
SHIFT  2, AR, BH
SHIFT  5, HR, DX
```

5.2.3 宏定义与宏调用中的参数(8)

- 在宏调用中，%X表示X是一个数，将这个数传到宏体中。

```
DATA1 MACRO A,B,C
      DW  A,B,C
ENDM
```

```
X = 10
Y = 20
DATA1 %X+2,5,%X+Y
DATA1 X+2,5,X+Y

DW 12,5,30
DW X+2,5,X+Y
```

5.2.4 重复汇编

相同和相似的一组语句,重复多次.　P167

例：打印乘法口诀表

参见：C5_167J.asm

(对比 C5_163J1.asm)

5.2.5 条件汇编

C5_170J.asm

```
.386
INCLUDEPART = 0
CODE SEGMENT USE16
        ASSUME CS:CODE
BEGIN:
        MOV AX, 0
IF INCLUDEPART EQ 1
        MOV BX, 0
        MOV CX, 0
ENDIF
        MOV AX, 4C00H
        INT 21H
CODE ENDS
        END BEGIN
```

5.2.6 宏库的建立和使用

宏库的建立

- 建立一个源文件，可任意命名。
- 在另一个源文件中，可以包含前一个文件的所有内容。

INCLUDE　源文件名

C5_172.asm　(宏库)

C5_173.asm

5.2.7 宏指令与子程序的比较

- 处理时间不同
- 处理方式不同
- 目标程序的长度不同
- 执行速度不同
- 参数传递方式不同

5.3 模块化程序设计

- 对于实际问题，可以将任务分解成多个模块，每个模块实现一个相对独立的子功能。各个模块分别编译（汇编）成.OBJ目标文件，最后将各个OBJ文件连接成一个可执行的文件。
- 在汇编语言中，每个模块(源文件)都是以END语句为结束标志的。
- 各个模块是怎样被连接在一起的(段的组合方式)?
- 模块之间是怎样通信的（相互调用及参数传递等）？

5.3.1 段的组合方式 (1)

- 每个模块可以有自己的堆栈段、数据段和代码段等。
- 各个模块的段名可以相同、也可以不同。连接程序怎样将各个模块的段组合到一起?

```
DATA1  SEGEMNT
A          DB  17 dup(0)          模块1
DATA1  ENDS

DATA2 SEGMENT  BYTE
B          DB 6 dup(1)           模块2
DATA2 ends
```

?

5.3.1 段的组合方式 (2)

段的组合方式

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

5.3.1 段的组合方式 (3)

段名 SEGMENT { USE16 | USE32 } { PARA | WORD | BYTE | PAGE } { NONE | PUBLIC | STACK | COMMON | AT | MEMORY } ['类别']

(1) 段名

每个段可以取任意的名字。

(2) 使用类别

USE16 表示使用16位段（可缺省不写）

USE32 表示使用32位段

5.3.1 段的组合方式(4)

(3) 定位方式

用来指定各段的段首址的确定方式。

- **PARA**: 段从能被16整除的地址处开始存放，即段物理首址的低四位必须为0。(缺省的方式)
- **WORD**: 段从能被2整除的地址处开始，即段物理首址的最低位必须为0。
- **BYTE**：段可从任何物理地址处开始存放。
- **PAGE**：段从能被256整除的地址处开始存放，即最低8位为0。

5.3.1 段的组合方式 (5)

(4)组合方式

用来指定本段同其它段的组合关系。

- NONE (不选择): 本段与其它段不发生任何逻辑上的连系，自己有自己的段首址。
- 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，大小不超过64K。
- STACK: 仅对堆栈段，功能同PUBLIC。
- 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: 将本段定位在所有连接在一起的段之上(在最高地址上)。

5.3.1 段的组合方式 (6)

(5)类别

‘类别’是用单引号括起来的字符串。

连接程序在进行连接处理时，将‘类别’相同的段，按出现的先后次序连续存放，但每段都有自己的首址。

5.3.1 段的组合方式 (7)

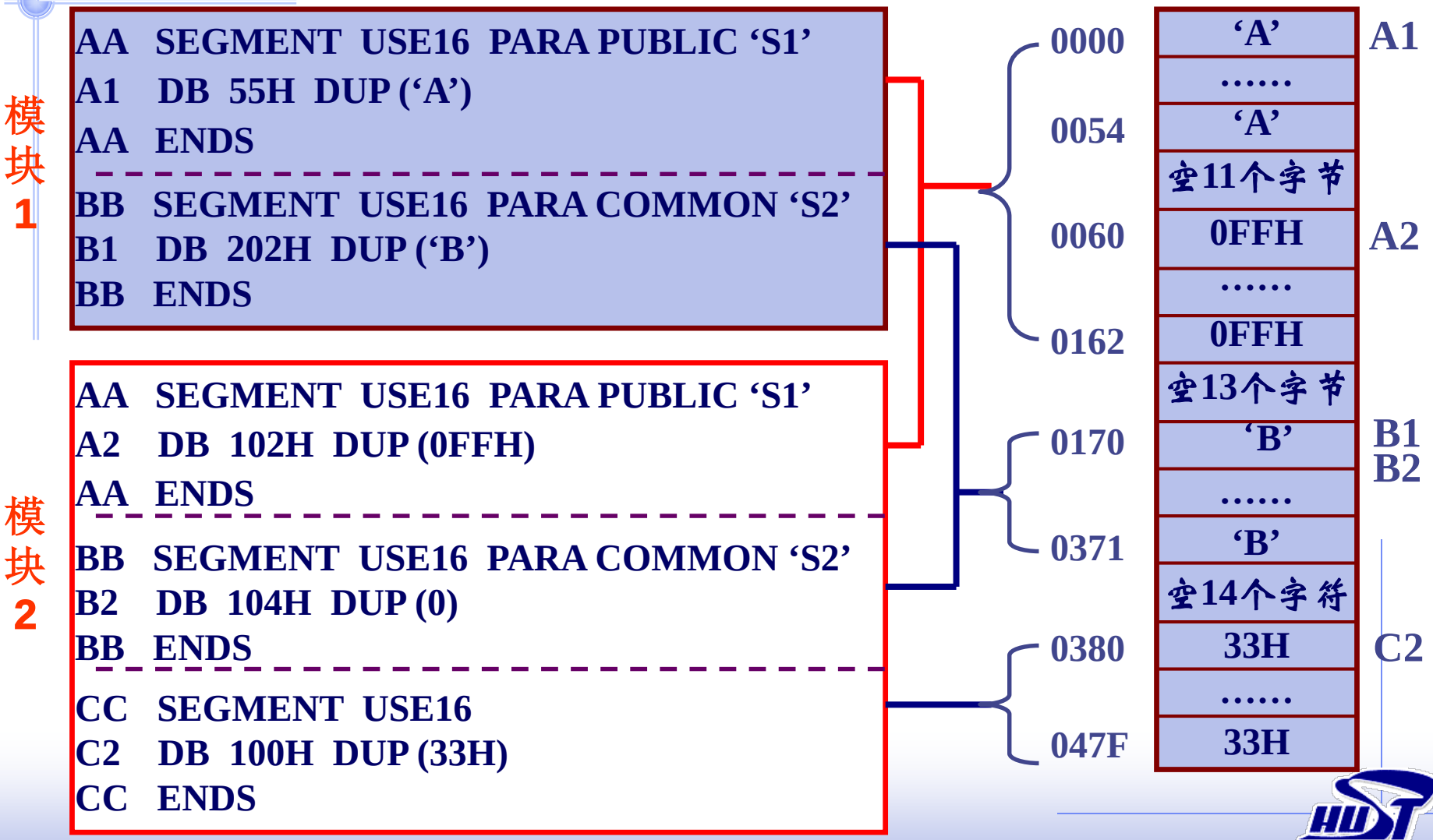
如：A SEGMENT ‘DATA’
B SEGMENT ‘CODE’
C SEGMENT ‘TO’
D SEGMENT ‘DATA’
E SEGMENT ‘TO’

注意：段的类别与段的名称是两个概念。

经连接程序连接后，存放的顺序为：

A SEGMENT ‘DATA’
D SEGMENT ‘DATA’
B SEGMENT ‘CODE’
C SEGMENT ‘TO’
E SEGMENT ‘TO’

5.3.1 段的组合方式 (8)


模块1
AA SEGMENT USE16 PARA PUBLIC 'S1'
A1 DB 55H DUP ('A')
AA ENDS
BB SEGMENT USE16 PARA COMMON 'S2'
B1 DB 202H DUP ('B')
BB ENDS
模块2
AA SEGMENT USE16 PARA PUBLIC 'S1'
A2 DB 102H DUP (0FFH)
AA ENDS
BB SEGMENT USE16 PARA COMMON 'S2'
B2 DB 104H DUP (0)
BB ENDS
CC SEGMENT USE16
C2 DB 100H DUP (33H)
CC ENDS
0000
0054
0060
0162
0170
0371
0380
047F
'A'
……
'A'
空11个字节
0FFH
……
0FFH
空13个字节
'B'
……
'B'
空14个字符
33H
……
33H
A1
A2
B1
B2
C2

5.3.2 通讯方式 (1)

模块之间是怎样通信:

- 相互调用
- 参数（符号）的相互引用

5.3.2 通讯方式 (2)

■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。

■语句格式： **PUBLIC 符号 [,符号]**

符号可以是变量名，可以是子程序名。

例如：

PUBLIC AVG, COUNT

PUBLIC SUB_P

5.3.2 通讯方式 (3)

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伪指令说明。

语句格式：

EXTRN 符号:类型 [,符号:类型]

注意:写法不是 extern.

例如：

```
EXTRN  AVG : WORD, COUNT : WORD
EXTRN  SUB_P : FAR
```

5.3.2 通讯方式 (4)

从 BEG (begin) 开始， 求 (COUNT) 个奇数（BEG为奇数） 或偶数（BEG为偶数）的和，存入 SUM 单元中；求它们的平均数，并存放到 AVG中。

程序：C5_main.asm
C5_sub.asm

可以实验段的定位和组合方式。

5.3.2 通讯方式(5)

若对其它模块用到的符号（在本模块中定义），不使用PUBLIC说明，在连接时，会报错：

unresolved external.

若引用了外部符号，而未用EXTRN说明，则在编译时，会报错：

Symbol not defined.

5.3.2 通讯方式(6)

程序设计中应注意的问题：

(1) 在一个文件模块中，不能定义相同名称的变量或者标号。不论它们是否在同一个段中。

(2) 不同的文件模块之间，可以有相同名称的变量和标号。

5.4 源程序综合举例

5.4.1 模块程序设计中的注意事项

1. 模块的划分的规则
2. 程序文件的命名
3. 标号的定义
4. 变量和缓冲区的定义
5. 模块注释

模块注释: 为方便子模块的调用，在子模块的前面必须有详细的说明。说明的内容包括：子模块功能、调用此子模块的入口参数和出口参数、子模块中所使用的主要变量等。

5.4.2 模块程序设计举例(1)

例1：编写一子程序模块DUMP，使之能将32位、16位寄存器的内容以有符号十进制数的形式显示出来，显示按“PUSHAD”进栈和“POPAD”出栈的顺序，格式为:

(EDI)= ××××××××　　(DI)= ××××
(ESI)= ××××××××　　(SI)= ××××
(EBP)= ××××××××　　(BP)= ××××
(ESP)= ××××××××　　(SP)= ××××
(EBX)= ××××××××　　(BX)= ××××
(EDX)= ××××××××　　(DX)= ××××
(ECX)= ××××××××　　(CX)= ××××
(EAX)= ××××××××　　(AX)= ××××

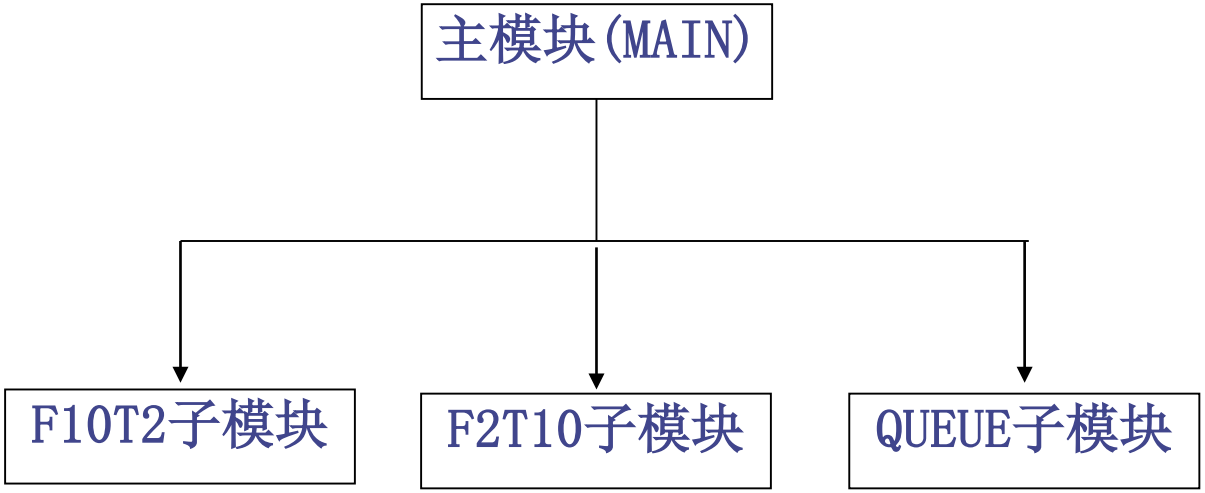
并要求主模块在调用该子模块之前与之后，全部寄存器的内容均不改变。

5.4.2 模块程序设计举例(2)

例2：从键盘输入一串以逗号为分隔符的十进制有符号数，然后按从小到大的顺序显示出来(仍以逗号为分隔符)，若输入的数中包括非法数，则停止输入并给出错误提示后返回DOS。

例：输入数组为： -180，90，123，-327，10
输出结果为： -327，-180，10，90，123

5.4.2 模块程序设计举例(3)



总结：

1. 串操作指令
2. 宏功能程序设计
3. 模块化程序设计

第5章 小结-串操作指令(1)

1. MOVS, MOVSB, MOVSW, MOVSD
2. CMPS, CMPSB, CMPSW, CMPSD
3. SCAS, SCASB, SCASW, SCASD
4. LODS, LODSB, LODSW, LODSD
5. STOSB, STOSW, STOSD

第5章 小结-串操作指令(2)

源串指针：　DS:SI / ESI　源串在当前数据段
目的串指针：　ES:DI / EDI　目的串在附加数据段
重复计数器：　CX / ECX
中间寄存器：　AL / AX / EAX
传送/比较方向：**(decrease)**

DF=0，SI / ESI，DI / EDI自动增量（加1,2,4）
DF=1，SI / ESI，DI / EDI自动减量（减1,2,4 ）

重复前缀：

REP　重复执行,直到（CX /ECX）=0；
REPE　（CX/ECX）<>0，且ZF=1时重复执行；
REPNE　（CX/ECX）<>0，且ZF=0时重复执行；
（对比 P115的 LOOP, LOOPE，LOOPNE)

第5章 小结-宏功能程序设计

- 宏指令的定义与调用方式；带间隔符实参的表示方法
- 宏体中变量和标号；为避免重复定义要用LOCAL
- 宏定义与子程序的区别；
- 宏库

段的组合方式

段名 SEGMENT { USE16 / USE32 } { PARA / WORD / BYTE / PAGE } { NONE / PUBLIC / STACK / COMMON / AT MEMORY } ['类别']

第5章 小结-模块化程序设计(2)

- 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