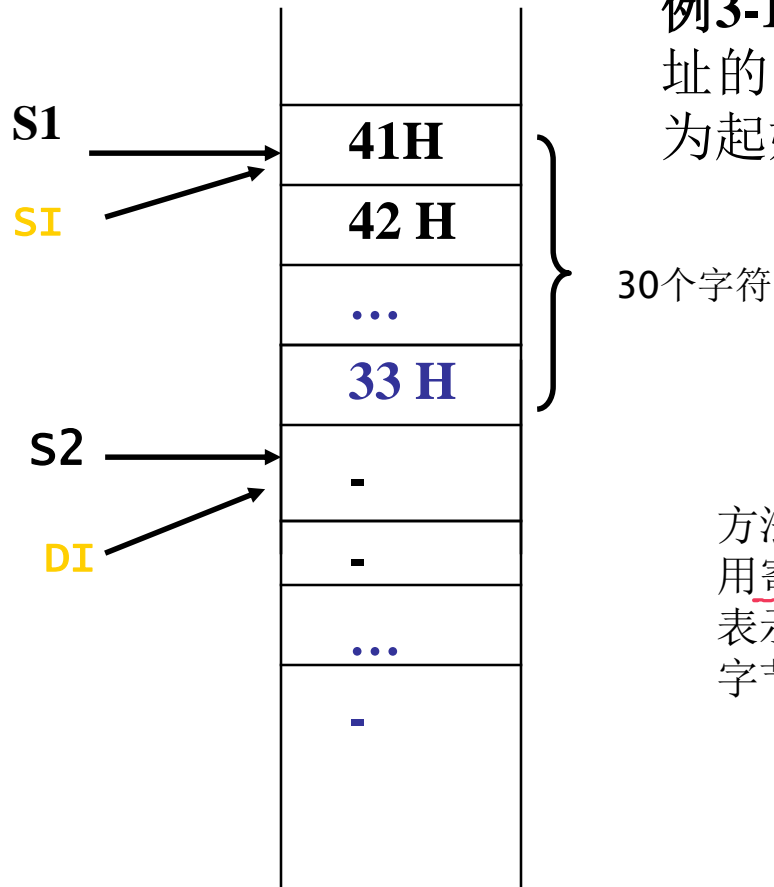




第三章 汇编语言程序设计举例



例3-1：数据块传送程序：将以S1为起始地址的30个字符依次传送到同数据段的以S2为起始地址的一片字节存储单元里。

方法一：数据块是用DB定义的一个字符串S1。用寄存器间接寻址方式访问S1和S2，即用[SI]表示S1中各字节的位移量，用[DI]表示S2中各字节的位移量。



DS EA ASCII

CS+IP 代码段地址 指令代码 指令

05F3 0003

程序如下:

```
DATA SEGMENT
S1 DB 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    DB 'XYZ0123'
S2 DB 30 DUP (?)
DATA ENDS
CODE SEGMENT
```

ASSUME DS: DATA, CS: CODE

START: MOV AX, DATA

MOV DS, AX

MOV SI, OFFSET S1

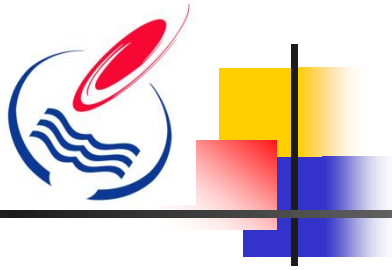
MOV DI, OFFSET S2

MOV CX, 30

```
NEXT: MOV AL, [SI]
      MOV [DI], AL
      INC SI
      INC DI
      LOOP NEXT
      MOV AH, 4CH
      INT 21H
```

CODE ENDS

START



我们也可以使用其它方法来实现，如用
变址寻址方式等。程序如下：

```
DATA    SEGMENT
S1  DB  'A' , 'B' , 'C' , 'D' , 'E'
      DB  'FGHIJKLMNOPQRSTUVWXYZ'
      DB  '0' , '1' , '2' , '3'
S2  DB  30  DUP ( ? )
DATA    ENDS
CODE    SEGMENT
      ASSUME  DS: DATA, CS: CODE
START:  MOV   AX, DATA
      MOV   DS, AX
```

```
MOV     SI, 0
MOV     CX, 30
NEXT:   MOV     AL, S1[SI]
        MOV     S2[SI], AL
        INC     SI
        LOOP    NEXT
MOV     AH, 4CH
INT     21H
CODE    ENDS
END     START
```

源地址
[offset SI + 0]

例3-2：从键盘上输入20个字符，然后以与键入字符的先后相同的顺序显示出来。



```
DSEG      SEGMENT
DATA      DB 20 DUP (?) (-个单元)
DSEG      ENDS
CSEG      SEGMENT
          ASSUME      CS: CSEG, DS: DSEG
GO:       MOV      AX, DSEG
          MOV      DS, AX
          MOV      CX, 20
          MOV      SI, OFFSET DATA
L01:      { MOV      AH, 01H
          { INT      21H
          MOV      [SI], AL
          INC      SI
          LOOP     L01
```

Handwritten notes:
- Red circle around **AX** in `MOV AX, DSEG`
- Blue circle around **AL** in `MOV [SI], AL`
- Blue arrow from **AL** to **I/O**
- Blue text: **ASCII码保存在AL中**

```
          MOV      CX, 20
          MOV      SI, OFFSET DATA
L02:      MOV      DL, [SI]
          { MOV      AH, 02H
          { INT      21H
          INC      SI
          LOOP     L02
          MOV      AH, 4CH
          INT      21H
CSEG      ENDS
          END      GO
```

Handwritten notes:
- Blue circle around **DL** in `MOV DL, [SI]`
- Red bracket grouping `MOV AH, 02H`, `INT 21H`, and `INC SI`



例3-3：在键盘上输入**20**个字符，然后用与输入字符的先后相反的顺序在屏幕上显示出来。

CODE SEGMENT

ASSUME CS: CODE

START: MOV CX, 20

L1: MOV AH, 01H

INT 21H

PUSH AX

LOOP L1

MOV DL, 0AH; 显示“回车”

MOV AH, 02H

INT 21H

MOV DL, 0DH; 显示“换行”

INT 21H

MOV CX, 20

L2: POP DX

MOV AH, 02H

INT 21H

LOOP L2

MOV AH, 4CH

INT 21H

CODE ENDS

END START

POP AX
MOV DL, AL
字寄存器

AL

光标回到所在行第一列

换行+回车



例3-4：数据的显示：

显示器. 键盘

对于外设，没有数据. 只有字符

输入. 输出, 是指字符.

一位十进制（BCD码）的显示：

数值转换成 ASCII 码

```
MOV    DL, AL
AND     DL, 0FH
ADD     DL, 30H
MOV     AH, 02H
INT     21H
```

AL 中有 03H

0000 0011
0000 1111



例3-4：数据的显示：

一位十六进制的显示：

```
MOV DL, AL
AND DL, 0FH
CMP DL, 09
JNA NEXT
ADD DL, 37H
JMP DISP
NEXT: ADD DL, 30H
DISP: MOV AH, 02H
INT 21H
```

CMP DL 0AH

JNAE NEXT

<

" "
<

CMP DL, 09

JNA NEXT

jmp if NA

有数据

```
MOV DL, AL
AND DL, 0FH
CMP DL, 09
JNA NEXT
ADD DL, 07
NEXT: ADD DL, 30H
MOV AH, 02H
INT 21H
```



例3-4：数据的显示：

1、把BL中一个字节的十进制数据（BCD码）显示出来。

CODE SEGMENT

ASSUME CS: CODE

START: MOV DL, BL

MOV CL, 04

SHR DL, CL ;高4位移至低4位

ADD DL, 30H

MOV AH, 02H

INT 21H ;高4位显示

MOV DL, BL

AND DL, 0FH

ADD/ **OR DL, 30H**

MOV AH, 02H

INT 21H ;低4位显示

MOV AH, 4CH

INT 21H

CODE ENDS

END START

30H~39H - 30H 输入一个字节的数据,保存若 dat, 01b
 40H~49H - 31H 在变量中
 50H~59H - 32H
 60H~69H - 33H
 70H~79H - 34H
 80H~89H - 35H
 90H~99H - 36H
 A0H~AH - 37H
 B0H~BH - 38H
 C0H~CH - 39H
 D0H~DH - 3AH
 E0H~EH - 3BH
 F0H~FH - 3CH

例3-4: 数据的显示:

2、把BL中一个字节的十六进制数据显示出来。

ASCII → 数字

数字 → ASCII

0~9 +30H

A~F +37H

a~f +57H

AND DL, 0FH

CODE SEGMENT

ASSUME CS: CODE

START: MOV DL, BL

MOV CL, 04

SHR DL, CL ;高4位移至低4位

CMP DL, 09

JNA NEXT

ADD DL, 07

NEXT: ADD DL, 30H

MOV AH, 02H

INT 21H ;高4位显示

MOV DL, BL

CMP DL, 0AH

JB NEXT2

ADD DL, 07

NEXT2: ADD DL, 30H

MOV AH, 02H

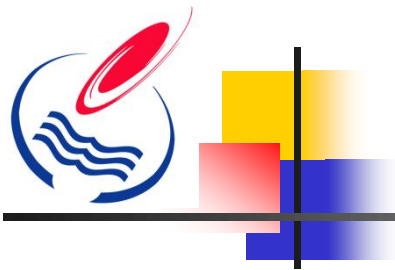
INT 21H ;低4位显示

MOV AH, 4CH

INT 21H

CODE ENDS

END START



换码指令: **XLAT** 或 **XLAT OPR**

执行操作: $(AL) \leftarrow ((\underline{BX}) + (\underline{AL}))$
基地址 *偏移*

例: **MOV BX, OFFSET TABLE ; (BX)=0040H**

MOV AL, 3

XLAT TABLE *可有可无*

指令执行后 (AL)=33H

注意:

- * 不影响标志位
- * 字节表格(长度不超过256) *字节*
- 首地址 → (BX)
- * 需转换代码 → (AL)

(DS)=F000H		
TABLE		
(BX) →	30 H	F0040
	31 H	F0041
(AL) = 3	32 H	F0042
	33 H	F0043

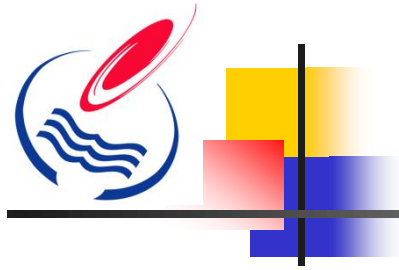
例3-5：编写一个加密0~9数字序列的程序，设0，1，2，3，4，5，6，7，8，9对应的密码表为：9，0，8，2，7，4，6，3，1，5，键盘输入0825，显示输出9184。

```

DATA    SEGMENT
STRDAT  DB    0, 8, 2, 5
TABLE   DB    '9082746315' ASCII
DATA    ENDS
CODE    SEGMENT
        ASSUME  CS:CODE , DS:DATA
GO:      MOV    AX, DATA
        MOV    DS, AX
        MOV    ES, AX
        LEA    SI, STRDAT
        LEA    BX, TABLE = MOV BX, OFFSET TABLE
        MOV    CX, 4 4个字节
L1:      MOV    AL, [SI]
        XLAT
        INC SI
        MOV    DL, AL
        MOV    AH, 02
        INT    21H
        LOOP   L1
        MOV    AH, 4CH
        INT    21H
        CODE   ENDS
        END    GO

```

例3-6：对一组字节型无符号数进行比较，把最大数显示在屏幕上。

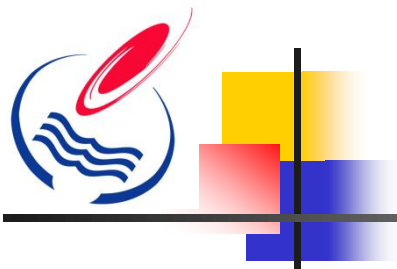


```

DATA          SEGMENT
BUFFER        DB    00H, 12H, 3BH, 43H, 60H, 0CH ...
COUNT        EQU $$-OFFSET BUFFER ($-BUFFER)  ; 其他地址 (当前地址) 0 1 2 3
MAX            DB    ?                               ; 4
DATA          ENDS

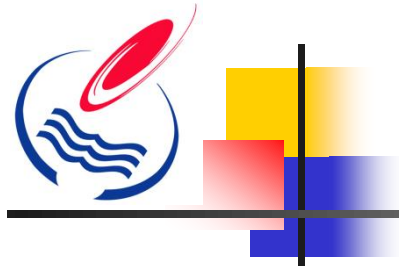
CODE          SEGMENT
ASSUME        CS:CODE, DS:DATA

START:        MOV     AX, DATA
              MOV     DS, AX
              MOV     SI, OFFSET BUFFER
              MOV     CX, COUNT  ; 立即寻址
              MOV     AL, [SI]
              INC     SI
              DEC     CX          ; 比较COUNT-1次
              COMPA:  CMP     AL, [SI]  ; 找大数
                    JA      NEXT
                    MOV     AL, [SI]
              NEXT:   INC     SI
              LOOP    COMPA        ; 比较完否?
              MOV     MAX, AL      ; 保存大数
    
```



```
MOV     BL, AL
MOV     DL, AL
MOV     CL, 4
SHR     DL, CL
L1:     CMP     DL, 0AH
        JB      L1
        ADD     DL, 7
        ADD     DL, 30H
        MOV     AH, 02H
        INT     21H           ; 显示高位
        MOV     DL, BL       ; 将大数送至DL
        AND     DL, 0FH      ; 截取其低4位
        CMP     DL, 0AH
        JB      L2
        ADD     DL, 7
L2:     ADD     DL, 30H
        MOV     AH, 02H
        INT     21H           ; 显示低位
        MOV     AH, 4CH
        INT     21H
CODE    ENDS
        END     START
```

例3-7：统计一批字型数据中负数的个数，结果放在RUSLT变量中。



```
DATA
BUFFER
COUNT
RUSLT
```

```
SEGMENT
DW 00H, 12H, 3BH, 0A3H, 94H, 0CH ...
DW $-OFFSET BUFFER ;或($-BUFFER)/2
DB 0
```

```
DATA
CODE
SEGMENT
```

```
ASSUME CS:CODE, DS:DATA
```

```
START: MOV AX, DATA
        MOV DS, AX
        LEA SI, BUFFER
        MOV BL, 0
        MOV CX, COUNT
```

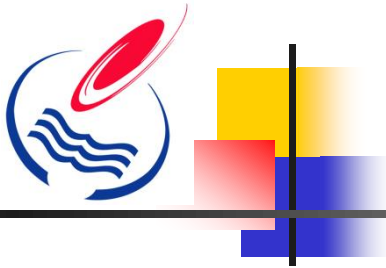
SHR CX, 1 用EDU则不用本指令.

```
COMPA: MOV AX, [SI] ;找大数
        OR AX, 0
        JNS NEXT
```

```
        INC BL
```

```
NEXT:  INC SI
        LOOP COMPA ;比较完否?
        MOV RUSLT, BL ;保存本数
```

```
.....
```



例3-8：间接转移

编写一个程序，根据输入的1-8的数字，转到8个不同的标号处进行各自的处理。即：

当输入1时，则转到标号L1处，输出字母A；

当输入2时，则转到标号L2处，输出字母B；

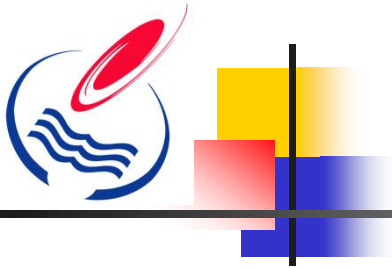
.....

当输入8时，则转到标号L8处，输出字母H。

假设： 数字1-8由键盘上输入。

分析： 首先将输入的ASCII码转换成对应的数字；然后根据数字，利用段内间接转移指令 JMP WORD PTR[BX]，转移到对应的标号处执行。

段内



```
DATA
TABLE1
DATA
CODE
```

```
SEGMENT
DW L1, L2, L3, L4, L5, L6, L7, L8 ; 地址表
ENDS
```

```
ASSUME CS:CODE, DS:DATA
```

```
START: MOV AX, DATA
```

```
MOV DS, AX
```

```
MOV AH, 01H
```

```
INT 21H
```

```
SUB AL, 30H
```

```
CMP AL, 8
```

```
JA L10
```

```
DEC
```

```
AI
```

高位与低位的对应关系 1-8

```
SHL AL, 1
```

```
MOV AH, 0
```

```
MOV SI, AX
```

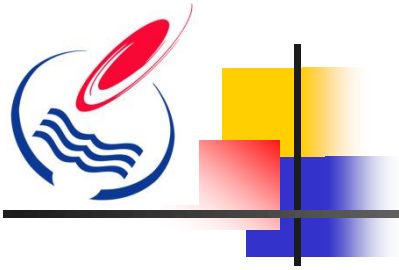
```
LEA BX, TABLE1
```

```
JMP WORD PTR [BX][SI]
```

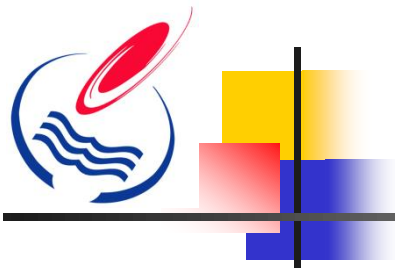
```
.....
```

```
JMP TABLE1[SI]
```

间接寻址 [BX+SI]



```
.....  
L1:      MOV    DL, 'A'  
          JMP    L9  
L2:      MOV    DL, 'B'  
          JMP    L9  
L3:      MOV    DL, 'C'  
          JMP    L9  
L4:      MOV    DL, 'D'  
          JMP    L9  
L5:      MOV    DL, 'E'  
          JMP    L9  
L6:      MOV    DL, 'F'  
          JMP    L9  
L7:      MOV    DL, 'G'  
          JMP    L9  
L8:      MOV    DL, 'H'  
          JMP    L9  
L9:      MOV    AH, 02H  
          INT    21H  
L10:     MOV    AH, 4CH  
          INT    21H  
CODE     ENDS  
END      START
```



例3-9：数据块传送程序：将以S1为起始地址的30个字符依次传送到同数据段的以S2为起始地址的一片字节存储单元里。（例3-1）

字符串操作指令：

MOVS str1, str2 ;将一个字节/字从**DS:SI** → **ES:DI**

MOVSB

MOVSW

CMPS、SCAS、LODS、STORS

指令前要先将源串首地址 → **DS:SI**

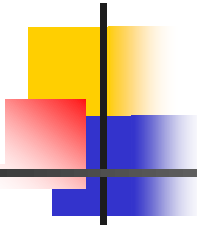
目标串首地址 → **ES:DI**

完成操作后**自动修改SI、DI**，使其指向串的下一个元素

串操作方向由**CLD**和**STD**指令设置

CLD 地址递增方向（**DF=0**）

STD 地址递减方向（**DF=1**）



重复前缀:

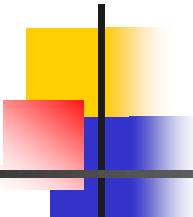
REP MOVS S1, S2

REP MOVSB / MOVSW

需要先将串的长度存入**CX寄存器**

每处理完一个元素自动使**CX-1**,直到**CX=0**才结束串传送——完成整个串的操作。

REPZ 、 REPNZ



```
DATA    SEGMENT
S1      DB  'ABCDEFGHJKLMNOPQRSTUVWXYZ'
COUNT  EQU  $-S1
S2      DB  COUNT DUP ( ? )
DATA    ENDS
CODE    SEGMENT
        ASSUME  CS:CODE, DS:DATA, ES:DATA
START:  MOV     AX, DATA
        MOV     DS, AX
        MOV     ES, AX
        MOV     SI, OFFSET S1
        MOV     DI, OFFSET S2
        MOV     CX, COUNT
        CLD
```

```
        MOVSB
        LOOP NEXT
        MOV     AH, 4CH
        INT     21H
CODE     ENDS
        END     START
```

Handwritten notes in red: "MOVSB" is crossed out and replaced with "MOVSB" (with a circled "B" and an arrow pointing to "S2, S1" in the original image). "LOOP NEXT" is crossed out. There are also arrows pointing to "S2, S1" and "START" with Chinese text "寄存器" (register) and "存贮器" (storage).



MOV AX DATAS

MOV DS AX

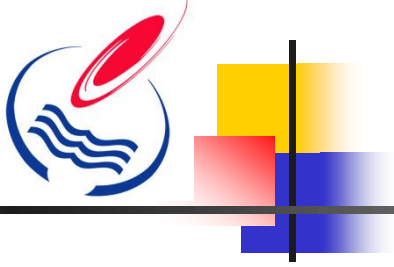
MOV AH, 0H

INT 21H

cmp al

```
DATA    SEGMENT
S1      DB  'ABCDEFGHJKLMNOPQRSTUVWXYZ'
COUNT  EQU  $-S1
S2      DB  COUNT    DUP ( ? )
DATA    ENDS
CODE    SEGMENT
        ASSUME  CS:CODE, DS:DATA, ES:DATA
START:  MOV     AX, DATA
        MOV     DS, AX
        MOV     ES, AX
        MOV     SI, OFFSET S1
        MOV     DI, OFFSET S2
        MOV     CX, COUNT
        CLD
```

```
CODE    REP MOVSB
        MOV     AH, 4CH
        INT     21H
        ENDS
        END     START
```



作业

- 1、把变量中定义（或输入）的**50**个字节型无符号数，按从小到大的顺序，重新排列在原变量中。
- 2、编写一个负数统计的程序：在内存 **BUFFER** 地址起有一组字节有符号数，要求统计其中负数的个数，并将统计结果以十进制的形式在屏幕上显示。