第六章 子程序设计

本章主要内容：

- 子程序调用与返回指令
- 编制子程序的基本要求
- 子程序设计举例
- DOS 功能子程序调用

子程序——在一个程序的不同的地方需要多次使用的某个程序段，将其进行独立编制。

调用与返回：在主程序中需要使用该功能时，就转移到子程序执行，执行完后又返回原程序继续执行。这样的程序结构称为子程序设计。

§6.1 调用与返回指令

在汇编语言中，子程序是以“过程”的形式表示。根据被调用过程与调用程序是否在同一个段内，可以分为两种情况。

➢段内调用与返回

主程序与子程序同在一个段内。这时，子程序的调用与返回只需修改指令指针IP。

右图中指令CALL PROCA就是段内调用。

```
CODEA SEGMENT
        …
        CALL PROCA
AAA:    …
        …
PROCA   PROC
        …
        RET
        …
PROCA   ENDP
        …
CODEA   ENDS
```

➢段间调用与返回

调用指令与子程序分别在不同的段，这时，需要同时修改CS和IP。

下面CODEB段中的CALL FAR PTR PROCB就是段间调用。

```
CODEA SEGMENT                    CODEB  SEGMENT
      ...                              ...
PROCB   PROC ...                       CALL  FAR PTR PROCB
      ...                              BBB:      ......
          RET
PROCB   ENDP                     CODEB  ENDS
        ...
CODEA  ENDS
```

1、调用指令

指令格式：　CALL　过程名

执行CALL指令时，先将断点压入堆栈中保存，然后转移到目标单元。

断点是调用子程序指令CALL的下一条指令的地址。前述程序结构图中，AAA和BBB就是两条调用子程序指令的断点。

CALL指令的执行对各标志位无影响。

（1）段内调用

(a) 段内直接调用

汇编指令书写格式为在 CALL 之后直接书写过程名

例如：CALL SUB1

（b）段内间接调用

子程序的起始地址（偏移量）由一个通用寄存器或一个字存储单元提供。

```
例如：CALL  BX
      CALL  CX
      CALL  WORD PTR  30H[BX][SI]
```

（2）段间调用

（a）段间直接调用

由于在定义过程时，对提供段间调用的过程，已经说明其属性为FAR。因此调用时，在CALL后直接书写过程名，也可以在过程名前面加FAR属性修饰。

例如： CALL PROC_NAME

CALL FAR PTR PROC_NAME

(b) 段间间接调用

调用指令提供一个双字存储单元的地址，它所指向的双字存储单元内容为被调用过程的起始地址。其中，两个低字节存放偏移量，两个高字节存放段基值。

例如：CALL DWORD PTR DISP[BX][DI]

（3）子程序调用指令与转移指令JMP的区别

两者都是无条件转移到目标单元，但CALL指令要保存“断点”，而JMP指令不保存断点。

3、返回指令

一个子程序最后执行的指令一定是返回指令，但不一定是最后一条指令。

根据子程序调用指令的使用情况，返回指令也分为段内返回和段间返回。其汇编指令书写形式都是**RET**，但它们的编码是不相同的。

（1）段内返回——指令编码为 **C3H**

执行该指令，将从堆栈顶部弹出一个字送入**IP**。

（2）段间返回——指令编码为 **CBH**

执行该指令，将从堆栈顶部弹出两个字分别送**IP**和**CS**中。

（3）带弹出值的返回指令

汇编指令格式为：**RET n**

其中n为一个立即数，长度为2字节。并且是一个偶数。

这条指令也分为段内返回和段间返回，它们的指令编码不同，分别为C2 n和CA n。

指令执行过程：

（1）从堆栈弹出1个字送IP（段内返回）或2个字送IP和CS；

（2）执行 SP<=(SP)+n 。将堆栈中已经用过的参数(n个字节)弹出舍去.

§6.2 编制子程序的基本要求

1、具有一定的通用性

选择和设计好子程序所需的各种入口参数和出口参数。

2、选择适当的参数传递方法

在主程序与子程序之间传递参数，可以选择的方法有：

A、使用通用寄存器

B、使用指定的存储单元

C、使用堆栈

3、注意信息保护

如果在子程序中需要使用某些寄存器或存储单元，为了不破坏它们原来在主程序中的值，为此需要进行信息保护。

信息的保护可以有两种方法：

A、在主程序中保存子程序中将要使用的一些寄存器的内容

```
...
PUSH  BX
PUSH  CX
CALL  SUB1
POP   CX
POP   BX
...
```

B、在子程序中保存将要使用的一些寄存器的内容

```
SUB2  PROC
      PUSH  BX
      PUSH  CX
       ......  ;  完成子程序功能指令序列
      POP   CX
      POP   BX
      RET
SUB2  ENDP
```

4、正确使用堆栈

由于堆栈中保存着主程序调用子程序时的断点地址。若在子程序中也使用了堆栈，注意各个数据压栈和出栈的顺序不能错，否则将导致数据错误和子程序返回地址错误。

5、编制子程序文件

子程序文件应包括文字说明与子程序本身两个部分。而文字说明一般包括：

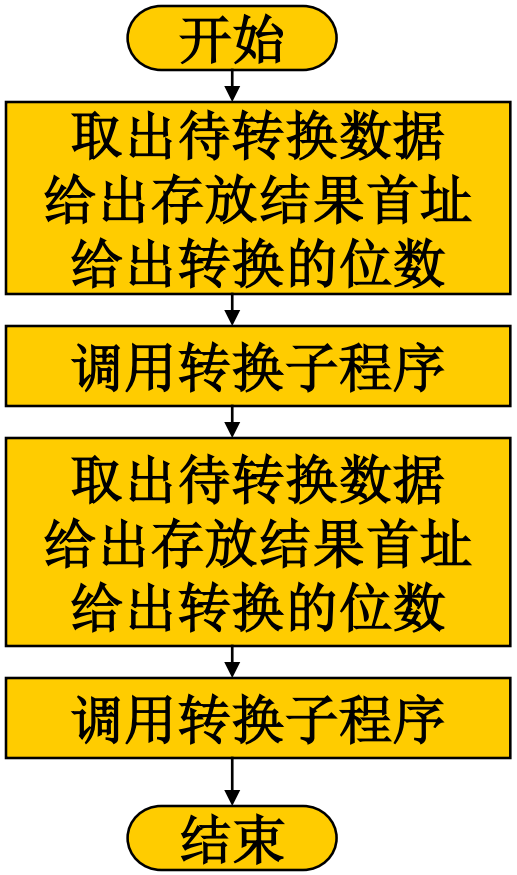
- 子程序名
- 子程序功能描述
- 子程序的入口参数与出口参数
- 使用哪些寄存器和存储单元
- 本子程序是否又调用其它子程序
- 子程序的调用形式、举例

§6.3 子程序设计举例

例 5.5.1 将两个给定的二进制数(8位和16位)转换为ASCII码字符串。

主程序提供被转换的数据和转换后的ASCII码字符串的存储区的首地址

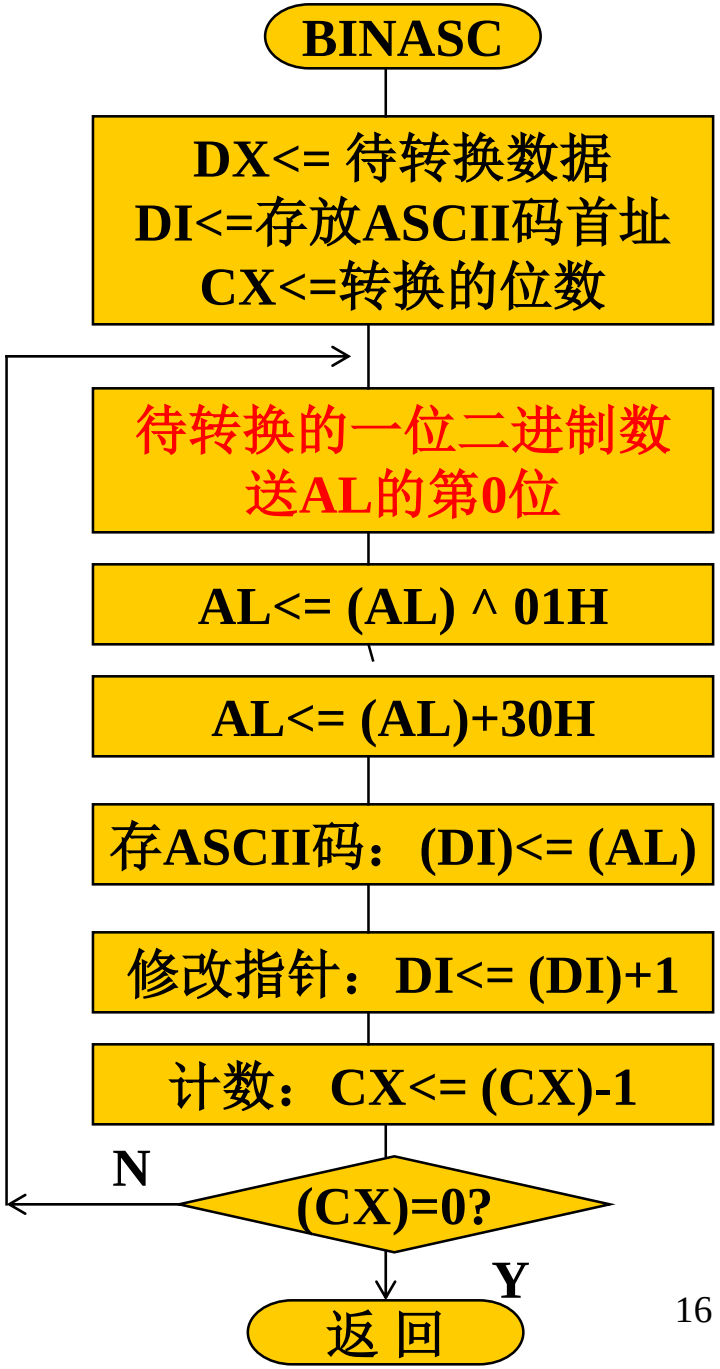
主程序框图



子程序框图：

子程序完成二进制数与ASCII码字符串的转换。子程序的入口参量有：被转换的数据、存储ASCII码字符串的首址和被转换数据的位数。无出口参量。

‘0’的ASCII码为30H，
‘1’的ASCII码为31H。



源程序的数据段和堆栈段安排如下：

```
DATA    SEGMENT
BIN1     DB  35H
BIN2     DW  0AB48H
ASCBUF   DB  20H DUP(?)
DATA    ENDS
STACK1  SEGMENT PARA STACK
         DW   20H DUP(0)
STACK1  ENDS
```

由于参量的传递方式有多种形式，其相应地在子程序中取入口参量的方法也有所不同。下面介绍三种参量的传递方法：

*用寄存器传递参量

*用堆栈传递参量

*用地址表传递参量

1、用寄存器传递参量

设调用子程序时，入口参量为：被转换的数在DX中，若数位<16，则从高到低地存放,转换后的ASCII码的存放首址在DI中。信息的保存由主程序完成。

主程序

```
COSEG   SEGMENT
        ASSUME CS:COSEG,DS:DATA,SS:STACK1
START:  MOV AX,DATA
        MOV DS,AX
        XOR DX,DX
        LEA  DI,ASCBUF;存放ASCII码的单元首址送DI
        MOV  DH,BIN1   ;待转换的第1个数据送DH
        MOV AX,8       ;待转换的二进制数的位数
        PUSH DI        ;保护信息
        CALL BINASC    ;调用转换子程序
        POP  DI        ;恢复信息
        MOV  DX,BIN2   ;待转换的第二个数据送DX
        MOV  AX, 16
        ADD  DI,8      ;设置下一个数的存放首址
        CALL BINASC
        MOV AH,4CH
        INT 21H
```

转换子程序

```
BINASC  PROC
        MOV CX，AX
LOP:    ROL DX，1   ；最高位移入最低位
        MOV AL，DL
        AND AL，1   ；保留最低位，屏蔽其它位
        ADD AL，30H ；AL中即为该数字符（0或1）的ASCII码
        MOV [DI]，AL；存结果
        INC DI          ；修改地址指针
        LOOP LOP
        RET
BINASC  ENDP
COSEG   ENDS
        END START
```

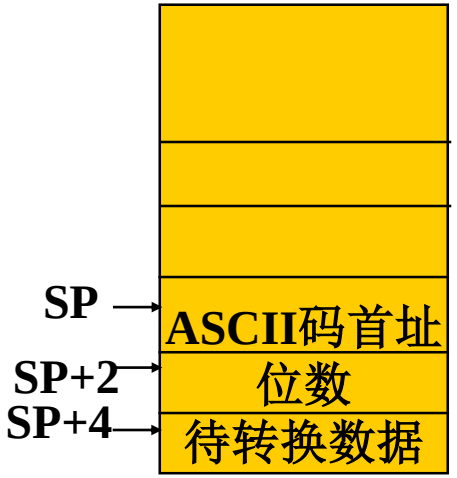
2、用堆栈传递参量

如果使用堆栈传递参量，一般应包括：

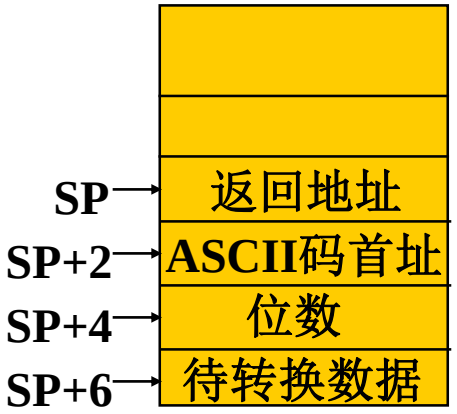
（1）在主程序中，将待转换的数据、存放ASCII码的首址和转换的位数压入堆栈；

（2）在子程序中保存信息。

```
；主程序
COSEG  SEGMENT
ASSUME CS：COSEG，DS：DATA，SS：STACK1
BEGIN：MOV AX，DATA
       MOV DS，AX
       MOV AH，BIN1
       PUSH AX          ；待转换数据压栈
       MOV AX，8
       PUSH AX          ；转换位数压栈
       LEA AX，ASCBUF
       PUSH AX     ；存放ASCII码的首址压栈
       CALL BINASC  ；调用转换子程序
       MOV AX，BIN2
      PUSH AX
      MOV AX，10H
      PUSH AX
     ADD DI，8
      PUSH DI
      CALL BINASC
      MOV  AH，4CH
       INT   21H
```

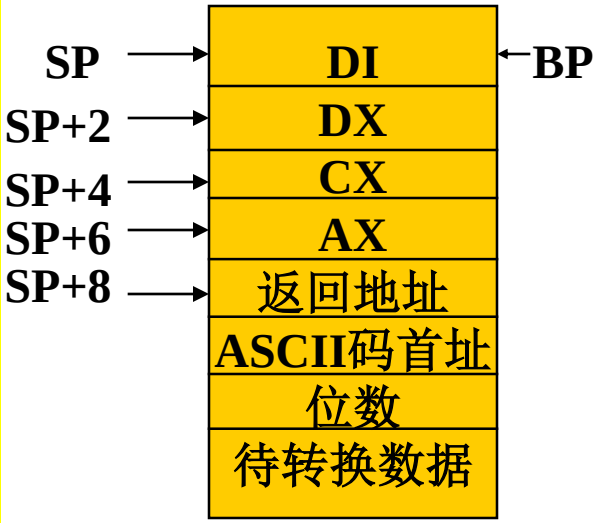


执行CALL指令前堆栈情况



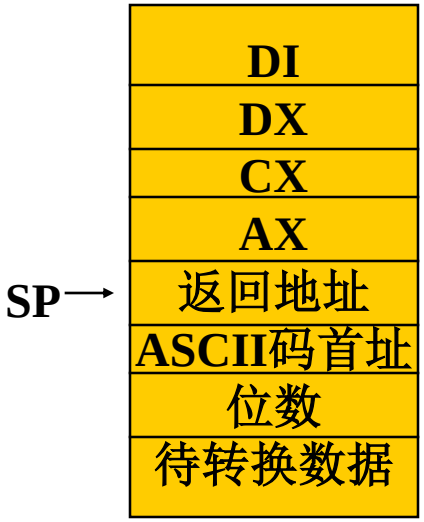
执行CALL指令后堆栈情况

```
；转换子程序
BINASC PROC
      PUSH  AX
      PUSH  CX
      PUSH  DX
      PUSH  DI
      MOV  BP，SP
      MOV DI，[BP+10] ;从堆栈取入口参数
      MOV CX，[BP+12]
     MOV DX，[BP+14];
LOP：  ROL DX，1
      MOV AL，DL
      AND AL，1
      ADD AL，30H
      MOV [DI]，AL
      INC DI
      LOOP LOP
```



子程序中保存信息并执行MOV BP，SP后

```
POP  DI
POP  DX
POP  CX
 POP AX
RET  6   ;返回并从堆栈中弹出6个字节
BINASC ENDP
COSEG  END BEGIN
```



执行RET 6前

执行RET 6后

3、用地址表传递参量

传递参数也可以采用传递参量的地址来实现。

在调用子程序前，将所有参量的地址依次存放在一个地址表中， 然后将该表的首地址传送给子程序。

数据段部分改为：

```
DATA    SEGMENT
BIN1    DB  35H
BIN2    DW  0AB48H
CUNT    DB  8，16
ASCBUP  DB 20H DUP(?)
ADR_TAB DW 3 DUP(0)   ;存放参量地址表
DATA    ENDS
```

主程序中有关指令序列修改为：

```
……
MOV  ADR_TAB,OFFSET BIN1   ;存参量地址
MOV  ADR_TAB+2,OFFSET CUNT
MOV  ADR_TAB+4,OFFSET ASCBUP
MOV  BX,OFFSET ADR_TAB ;传表首址
CALL BINASC8
MOV  ADR_TAB,OFFSET BIN2
MOV  ADR_TAB+2,OFFSET CUNT+1
MOV  ADR_TAB+4,OFFSET ASCBUP+8
MOV  BX,OFFSET ADR_TAB ;传表首址
CALL BINASC16
……
```

转换子程序设置两个入口，一个是转换8位数据的入口BINASC8，另一个是转换16位数据的入口BINASC16。

```
BINASC    PROC
BINASC8:  MOV DI，[BX]；取待转换8位数据
          MOV DH，[DI]
          JMP TRAN
BINASC16:MOV DI,[BX]   ;取待转换16位数据
          MOV DX,[DI]
TRAN:     MOV DI,[BX+2] ;取待转换数据位数
          MOV CL,[DI]
          XOR CH,CH
          MOV DI,[BX+4]:取存ASCII码首址
LOP:      ROL DX,1
          MOV AL,DL  ;待转换的1位送到AL中转换
          AND AL,1
          ADD AL,30H    ;构成相应的ASCII码
          MOV [DI],AL   ;存结果
          INC DI
          LOOP LOP
          RET
```

§6.4 DOS 功能子程序调用

DOS操作系统为程序设计人员提供了可以直接调用的功能子程序。调用这些子程序可以实现从键盘输入数据，将数据送显示器显示，以及磁盘操作等功能。

调用这些子程序时，需要使用软中断指令 INT 21H，并且在执行该指令之前，需要将要调用的功能号送入寄存器AH中，有关的参量送入指定的寄存器。

调用过程包括以下三个步骤：

* 送入口参量给指定寄存器
* AH<=功能号
* INT 21H

1、带显示的键盘输入（1号功能）

调用该功能子程序将等待键盘输入，直到按下一个键。将字符的ASCII码送入AL寄存器，并在屏幕上显示该字符。如果是Ctrl-C组合键，则停止程序运行。该功能调用无入口参量。

例如：MOV AH，01H

INT 21H

2、不带显示的键盘输入（8号功能）

该功能调用与1号功能的作用相似，区别是8号功能将不显示输入的字符。调用方法为：

MOV AH，8

INT 21H

3、不带显示的键盘字符输入（7号功能）

该功能与8号功能相似，但对Ctrl-C组合键和TAB制表键无反应。调用方法：

```
MOV AH，7
INT  21H
```

4、字符串输入（0AH号功能）

该功能调用可实现从键盘输入一个字符串，其长度可达255个字符。调用该功能前，应在内存中建立一个输入缓冲区。

缓冲区第一个字节是可输入的最大字符数+1；第二个字节是系统在调用该功能时，自动填入的本次调用时实际输入的字符个数；从第三个字节开始存放输入字符的ASCII码。

当用户输入回车键时，结束输入，并将回车键的ASCII码（0DH）作为最后一个字符送入缓冲区。但它不计入实际输入字符个数。

调用入口参量：

DS和DX寄存器分别装入输入缓冲区的段基值和偏移量

```
CHAR_BUF DB 31H      ;缓冲区的最大长度
         DB 0        ;存实际输入字符数
         DB 31H DUP(0);输入缓冲区
         ......
  MOV DX,SEG CHAR_BUF;如果DS已经指向CHAR_BUF所在
  MOV DS,DX          ;数据段，则可以省去这两条指令
  MOV DX,OFFSET CHAR_BUF
  MOV AH,0AH
  INT 21H
```

5、字符显示（2号功能）

该功能实现在屏幕上显示单个字符。

入口参数：DL<=要显示字符的ASCII码。

例如：MOV　DL ‘A’

　　MOV AH，2

　　INT 21H

6、字符打印（5号功能）

该功能将字符送入打印机接口，实现单个字符的打印操作。

入口参数：DL<= 打印字符的ASCII码

```
MOV  DL，  ‘A’
MOV  AH，5
INT    21H
```

7、字符串显示（9号功能）

该功能实现将一个字符串显示到屏幕上。

入口参数：

（1）将待显示的字符串存放在一个数据缓冲区，字符串以符号“$”作为结束标志。

（2）将字符串的首址的段基值和偏移量分别送入DS和DX中

```
例如：CHAR    DB  ‘This is a test.’,0AH,0DH,’$’
          ..........
          MOV  DX,OFFSET CHAR
          MOV AH,9
          INT  21H
```

8、直接输入输出（6号功能）

该功能可以实现键盘输入，也可以实现屏幕显示操作。两种操作通过DL的内容确定。

（1）（DL）=00—0FEH，显示输出。DL中是所显示字符的ASCII码。

例如：显示美元符号“$”的程序段为：

```
MOV  DL，24H ；$的ASCII码为24H
MOV AH，06
INT   21H
```

（2）（DL）=FFH，从键盘输入字符

该功能的字符输入不等待键盘输入，而是从键盘缓冲区中读取。读取的字符ASCII码送入AL中，如果没有键按下，则标志位ZF=1。

例如：

```
WAIT: MOV  DL，0FFH
      MOV  AH，6
      INT 21H
      JZ  WAIT
```

9、读出系统日期（2AH号功能）

读出的日期信息放入指定的寄存器中：

CX：年（1980—2099）

DH：月（1—12）

DL：日（1—31）

AL：星期（0—星期日，1—星期一……）

例如：

```
YEAR    DW  ?
MONTH DB  ?
DAY     DB  ?
        ......
        MOV  AH,2AH
        INT  21H
        MOV  YEAR,CX
        MOV  MONTH,DH
        MOV  DAY,DL
```

10、设置系统日期（2BH号功能）

该功能用来改变计算机CMOS中的系统日期。入口参数：

CX<=年号（1980—2099）

DH<= 月号（1—12）

DL<= 日（1—31）

返回参数在AL中，成功设置，则返回(AL)=0，否则（AL）=0FFH

例如：

```
MOV CX,2000
MOV DH,11
MOV DL,2
MOV AH,2BH
INT 21H
CMP AL,0
JNE ERROR ;转出错处理
……
```

11、读出系统时间（2CH号功能）

执行该功能将获得系统的当前时间。返回的时间参数存放在指定的寄存器中：

CH：小时（0—23）
CL：分（0—59）
DH：秒（0—59）
DL：百分秒（0—99）

12、设置系统时间（2DH号功能）

调用该功能，将设定系统时间。其入口参数为：

CH：小时 （0—23） CL：分 （0—59）
DH：秒 （0-59） DL：百分秒（0-99）

该功能执行后返回时，如果调用成功，则（AL）=0。否则（AL）=0FFH