

《微型计算机原理与接口技术》

第5版

第4章

汇编语言程序设计



§4.3 汇编语言程序设计方法与实例



汇编语言程序设计

- ◆ 汇编语言程序设计采用结构化程序设计方法。
- ✎ 每个程序只有一个入口，必须要有出口，中间内容不能含有死循环语句。
- ✎ 程序都按照顺序结构、条件分支结构和循环结构等3种基本结构进行构建。
- ✎ 设计时先考虑总体、全局目标，再考虑细节、局部问题，把复杂问题分解为一个个模块或子目标，一步步进行设计。
- ✎ 将这些基本结构、子模块合理组合起来，就可构成一个大的程序。



汇编语言程序设计

- ✎ 编程时要在程序行上适当加注释。这样设计出来的程序层次分明，结构清楚，可读性强，便于调试。
- ✎ 编写较复杂的程序时，一般应先画出程序流程图，将设计步骤细化，再按流程图设计编写程序。
- ✓ 下面先从3种基本结构入手，介绍编程方法和应用实例，再介绍实际应用较多的代码转换、过程调用等编程例子，后者也要用到 3 种基本结构。
- ✓ 通过学习这些实例，掌握汇编语言程序设计的基本方法，为编写复杂程序奠定基础。



4.3.1 顺序结构程序设计

4.3.2 分支程序设计

4.3.3 循环结构程序

4.3.4 代码转换程序

4.3.5 过程调用




4.3.1 顺序结构程序设计

- 顺序结构程序也称为简单程序，这种程序按指令排列的先后顺序逐条执行。

例4.33 编写显示一个笑脸字符在显示器上的程序，程序命名为HAPPY.ASM。

```

PROG1      SEGMENT
            ASSUME CS: PROG1                ; 只有1个代码段
START:      MOV     DL, 1                    ; DL ← 要显示字符
                                                    ; 的ASCII码
            MOV     AH, 2                    ; AH ← 功能号2
            INT     21H                      ; 显示笑脸 符
            MOV     AX, 4C00H
            INT     21H                      ; 返回DOS
PROG1      ENDS
            END     START
```



4.3.1 顺序结构程序设计

- ◇ 如果用循环程序将00~FFH先后送入DL，再利用DOS的2号功能调用，则可显示全部的标准和扩展ASCII码，包括全部控制符以及积分符、希腊字母等。

例4.34 由人机对话从键盘输入1个10进制数（0~9），查表求键入数字的平方值，存入AL寄存器中，并显示有关的提示信息。试编写汇编语言程序。

解：

- 数据段中，先给出数字0~9的平方值，逐个存入TABLE开始的内存中，形成表格，以便查找，再给出等待显示的提示信息。
- 代码段由3个部分组成：显示提示信息；等待键入数字；查表求键入数字的平方值，并将结果存入AL中。



例4.34

程序如下：

DATA SEGMENT

TABLE DB 0, 1, 4, 9, 16, 25, 36, 49, 64, 81

； 数字0~9的平方值

BUF DB 'Please input a number(0~9):', 0DH, 0AH, '\$'

； 提示信息

DATA ENDS

CODE SEGMENT

ASSUME CS: CODE, DS: DATA

START:

MOV AX, DATA

MOV DS, AX

； 设置DS

MOV DX, OFFSET BUF

； 设置DX，使字符串首地址=DS: DX

MOV AH, 9H

； 9号DOS功能调用

INT 21H

； 显示提示信息



例4.34

```
MOV AH, 01      ; 1号功能调用, 等待键入字符
INT  21H        ; AL ← 键入数字的ASCII码
AND  AL, 0FH    ; AL ← 截下数字值
                ; (表内元素序号)

MOV BX, OFFSET TABLE
                ; BX指向表头地址TABLE

MOV AH, 0; AX寄存器高字节清0
ADD  BX, AX
                ; 表头地址+键入数字(AL), 结果存入BX

MOV AL, [BX]    ; 查表求得平方值
                ;

MOV AX, 4C00H
INT  21H        ; 返回DOS
CODE ENDS
END  START
```



4.3.1 顺序结构程序设计

例4.35 在存储单元A1和A2中，各存有一个2字节的无符号数，低字节在前，高字节在后。编程将两数相加，结果存入SUM单元，也要求低字节在前，高字节在后，进位存入最后一个字节单元。

DATA SEGMENT

A1 DB 56H, 78H ; 数A1

A2 DB 4FH, 9AH ; 数A2

SUM DB 3 DUP (0)

; 存两数相加之和，考虑进位

DATA ENDS

;

CODE SEGMENT

ASSUME CS: CODE, DS: DATA



例4.35

BEGIN:

MOV AX, DATA	
MOV DS, AX	; 设置数据段基址
MOV BX, 0	; BX为地址指针, 初值清0
CLC	; 进位位清0
MOV AL, A1 [BX]	; 取低字节A1
ADC AL, A2 [BX]	; 与A2低字节相加
MOV SUM [BX], AL	; 存入SUM单元(低字节)
INC BX	; 调整指针
MOV AL, A1 [BX]	; 取高字节相加
ADC AL, A2 [BX]	
MOV SUM [BX], AL	; 存高字节
JNC STOP	; 无进位, 转STOP



例4.35

```
INC    BX                      ; 有进位
MOV    AL, 0
INC    AL
MOV    SUM[BX], AL            ; 进位存入SUM+2单元
STOP:  MOV    AX, 4C00H
      INT    21H
CODE   ENDS
END    BEGIN
```



4.3.1 顺序结构程序设计

4.3.2 分支程序设计

4.3.3 循环结构程序

4.3.4 代码转换程序

4.3.5 过程调用



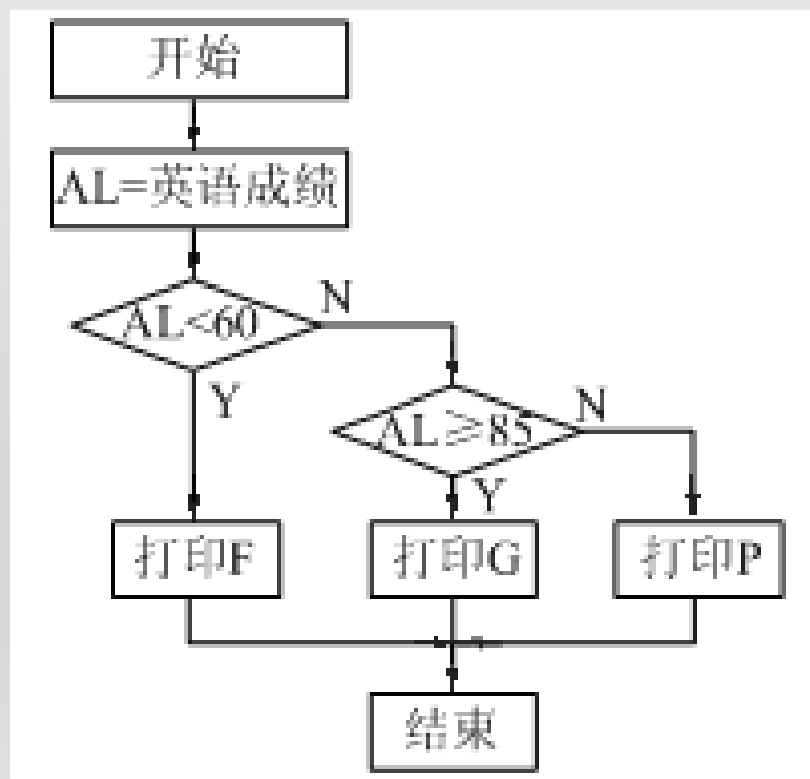
4.3.2 分支程序设计

- 要求程序根据不同条件选择不同的处理方法，即程序处理步骤中出现了分支，应根据某一特定条件，选择其中一个分支执行。

例4.36

设某学生的英语成绩已存放在AL寄存器中，如果分数低于60分，则打印F，如高于等于85分，则打印G，否则打印P。这就是一个分支程序。

程序框图➡



4.3.2 分支程序设计

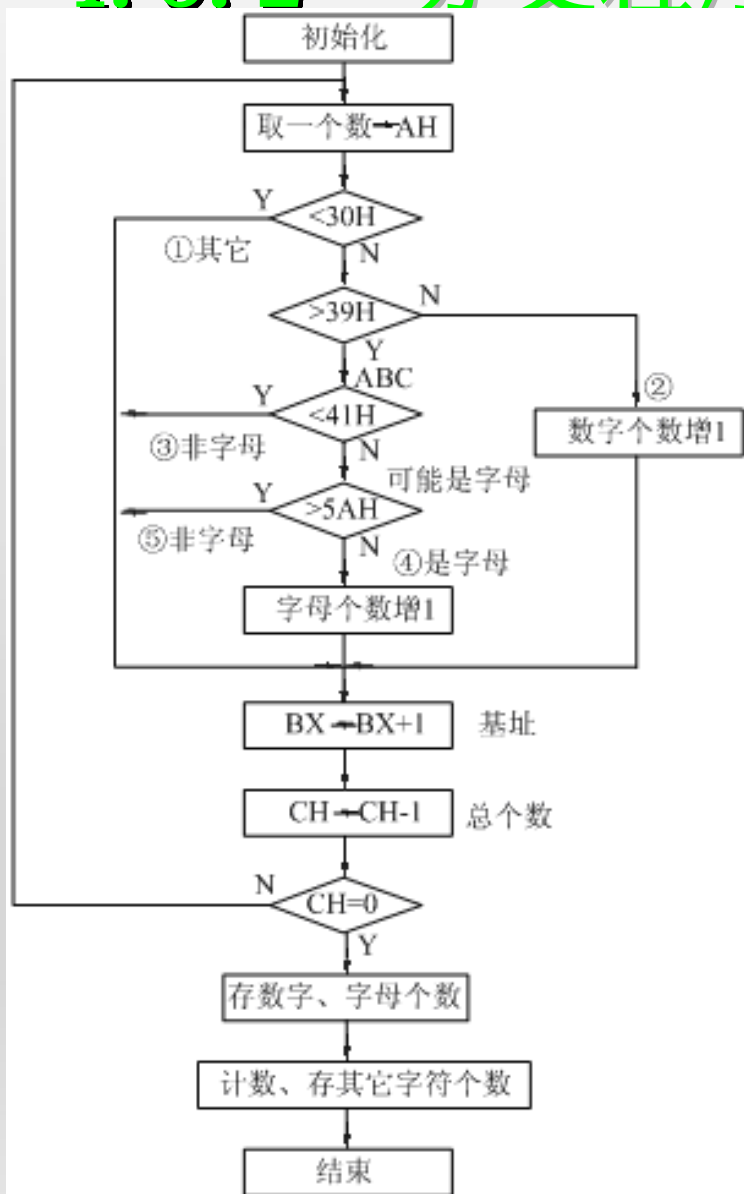
- 下面介绍一个比较复杂的分支程序，其中也包含了循环程序。

例4.37 在存储器中以首地址BUF开始存有一串字符，字符串个数用COUNT表示。要求统计数字0~9、字母A~Z和其它字符的个数，并分别将它们个数存储到NUM开始的3个内存单元中去。

- 在ASCII码表中，数字0~9的ASCII码为30H~39H，大写字母A~Z的ASCII码为41H~5AH，其余值为其它字符或控制符的ASCII码值。可以将ASCII码分成5个部分或5个分支来处理，其示意图如下



例4.37 分支程序设计



➤ 先从BUF单元取出1个字符的ASCII码，经分支程序判断它属于数字、字母还是其它字符，然后使相应计数器的值+1。

➤ 数字个数存放在DL中，字母个数存放在DH中。

➤ 接下来分析第2个数，直至所有字符处理完后，将统计出的个数送入相应存储单元。

← 程序框图



例4.37

```
DATA    SEGMENT
BUF      DB '+36', 'PRINT', 'abc', '2A0CH', '#'
; 一串字符

COUNT EQU $-BUF      ; COUNT=字符总个数
NUM      DB 3 DUP (?)  ; 先后存放存数字、字母
; 和其它字符个数

DATA    ENDS

;
CODE    SEGMENT
        ASSUME CS: CODE, DS: DATA
START:  MOV AX, DATA
        MOV DS, AX      ; 设置数据段
        MOV CH, COUNT   ; CH ← 数□□度
        MOV BX, 0       ; BX为基址指针, 初值清0
        MOV DX, 0
; DH 数字个数, DL字母个数, 初值清0
```



例4.37

```
LOOP1: MOV  AH, BUF[BX]    ; AH ←取一个数
        CMP  AH, 30H        ; <30H?
        JL   NEXT          ; ①是，转
        CMP  AH, 39H        ; >39H?
        JG   ABC            ; 是，转
        INC  DH              ; ②否，数字个数增1
        JMP  NEXT

ABC:    CMP  AH, 41H        ; <41H?
        JL   NEXT          ; ③是，非字母，转
        CMP  AH, 5AH; >5AH?
        JG   NEXT          ; ⑤是，非字母，转
        INC  DL              ; ④否，字母个数增1

NEXT:   INC  BX              ; 基址指针加1
        DEC  CH              ; 字符串长度减1
        JNZ  LOOP1          ; 未完，取下一个数
```



例4.37

```
MOV    NUM, DH      ; 已完, 存数字个数
MOV    NUM+1, DL     ; 存字母个数
MOV    AH, COUNT
SUB     AH, DH
SUB     AH, DL        ; 计算出其它字符个数
MOV    NUM+2, AH     ; 存其它字符个数
MOV    AX, 4C00H
INT     21H
```

```
CODE    ENDS
END      START
```



4.3.1 顺序结构程序设计

4.3.2 分支程序设计

4.3.3 循环结构程序

4.3.4 代码转换程序

4.3.5 过程调用



4.3.3 循环结构程序

- ◆ 要求某段程序反复执行多次，直到满足某些条件时为止，这种程序称为循环结构程序。
- ◆ 在循环程序中，常用计数器（如CX寄存器）来控制循环次数。先将计数器置1个初值，用来表示循环操作的次数，每执行一次循环操作后，计数器-1，减到0时，表示循环结束。



4.3.3 循环结构程序

例4.38 在一串给定个数的数据中寻找最大值，
存放到MAX存储单元中。

```
DATA    SEGMENT
BUF      DW  1234H, 3200H, 4832H, 5600H
          ; 一串字数据
COUNT  EQU  ($-BUF) / 2    ; 数据个数（循环次数）
MAX      DW  ?              ; 存最大值
DATA    ENDS

;
STACK   SEGMENT 'STACK'
STAPN   DB    100 DUP ( ? )
TOP     EQU   LENGTH STAPN
STACK   ENDS
```



例4.38 4.3.3 循环结构程序

CODE SEGMENT

MAIN PROC FAR

ASSUME CS: CODE, SS: STACK

START: MOV AX, STACK

MOV SS, AX

MOV SP, TOP

PUSH DS

SUB AX, AX

PUSH AX

MOV AX, DATA

MOV DS, AX

MOV CX, COUNT ; CX ← 字符个数

LEA BX, BUF ; BX ← BUF的偏移地址

MOV AX, [BX] ; AX ← 缓冲器中取一个数



例4.38. 3 循环结构程序

```
INC    BX           ; 修改地址指针
INC    BX
DEC    CX           ; 循环次数减1
AGAIN: CMP    AX, [BX] ; AX与后取的数比较
      JGE    NEXT      ; 如AX中数大于等于后者,则转
      MOV    AX, [BX] ; 如后取的数大, 则将其送AX
NEXT:  LOOP   AGAIN    ; 没处理完, 转（循环操作）
      RET              ; 返回DOS
MAIN   ENDP          ; 处理完, 结束
CODE   ENDS
      END    MAIN
```

- ◆ 本例通过LOOP指令执行循环操作，取字符串的地址指针BX要用指令修正，以指向下个字单元取数进行比较。



4.3.3 循环结构程序

例4.39 用循环程序设计方法，求A和B两个4字节BCD数之和，它们在内存中以压缩BCD码的形式存放，低字节在前，高字节在后。要求结果以同样形式存放在以SUM开始的单元中。

- 在例4.35中，进行2字节无符号运算时，采用顺序结构程序，用了两段加法程序。本例做4字节的加法运算，用循环结构编程，只要写一段加法程序，反复执行4次即可。



4.3.3 循环结构程序

例4.39

```
DATA    SEGMENT
A       DB  44H, 33H, 22H, 11H ; 数A, BCD数加后缀H
B       DB  88H, 77H, 66H, 55H ; 数B, 同上
SUM     DB   5 DUP ( ? )       ; 存和 (含进位)
DATA    ENDS

STACK   SEGMENT 'STACK'
STAPN   DB  100 DUP ( ? )
TOP     EQU   LENGTH STAPN
STACK   ENDS

CODE    SEGMENT
MAIN    PROC FAR
        ASSUME CS: CODE, DS: DATA,
               ES: DATA, SS: STACK
        ; 使用串操作指令要设附加段
```



4.3.3 循环结构程序

例4.39

```
START:  MOV  AX, STACK    ;设置堆栈段
        MOV  SS, AX
        MOV  SP, TOP
        PUSH DS
        SUB  AX, AX
        PUSH AX
        MOV  AX, DATA
        MOV  DS, AX      ;设置数据段
        MOV  ES, AX      ;设附加段, 与数据段相同
        MOV  SI, OFFSET A    ;SI ← 数A偏移地址
        MOV  BX, OFFSET B    ;BX ← 数B偏移地址
        MOV  DI, OFFSET SUM  ;DI ← 和□元地址
        MOV  CX, LENGTH SUM
        ;CX ← 和的□度(含进位位)为5
```



4.3.3 循环结构程序

例4.39

```
DEC    CX                ; 循环次数为4, 只要做4次加法
CLD                    ; 串操作清方向标志, 地址增量
CLC                    ; 进位位清0
MOV    AH, 0; AH存最后一次进位, 初值置0

GET_SUM:
    LODS A                ; AL ← 从A中取1数, SI自动+1
    ADC  AL, [BX]          ; 与B数相加, 结果 送AL
    DAA                    ; BCD数调整
    INC  BX                ; B数指针+1
    STOS SUM               ; SUM单元 结果, DI自动+1
    LOOP GET_SUM           ; CX ← CX-1, CX ≠ 0 转循环做加法
    ADC  AH, 0              ; 4次后CX=0, 将进位加到AH中
    MOV  AL, AH
    STOSB                  ; 进位存入SUM+4单元
```



4.3.3 循环结构程序

例4.39

```
        RET                ; 返回DOS  
MAIN    ENDP  
CODE    ENDS  
        END    MAIN
```

- 本例也是利用LOOP指令执行循环加法操作；
- 利用LODS A指令取A数时，源地址指针SI自动修改；
- 利用STOS指令存数时，目的地址指针DI自动修改；
- 但取B数时，地址指针BX必须用指令修改。



4.3.3 循环结构程序

例4.40 有一个无符号数组共含5个元素: 12, 7, 19, 8, 24, 它们存放在LIST开始的字单元中, 编程将数组中的数按从大到小的次序排列 (元素个数 $n=5$)。

编程思路:

- 编程时采用冒泡法排序。
- 比较从第1个元素开始, 与相邻数比较, 若大的在前小的在后, 次序就排好了, 不要交换, 否则交换。
- 然后将小的数与第3个元素比较, 经 $n-1(=4)$ 次比较后, 一行中最小的元素7排到了最后面。共循环比较了 $n-1(=4)$ 次。
- 再作第二轮比较, 这轮只要比较 $n-2(=3)$ 次, 即可将数组中的数按从大到小的次序排列好。
- 这是一个多重循环程序。
- 程序稍加修改后, 即可实现从小到大的排序。



4.3.3 循环结构程序

例4.40

◇ 比较过程中数组中数的排列:

原始数据	12	7	19	8	24
第一轮比较后	12	19	8	24	7
第二轮比较后	19	12	24	8	7
第三轮比较后	19	24	12	8	7
第四轮比较后	24	19	12	8	7

找出最小值7

找出第二小的值8

找出第三小的值12

已排好次序，大循环次数为 $n-1(=4)$



4.3.3 循环结构程序

例4.40

```
DATA    SEGMENT
LIST    DW    12, 7, 19, 8, 24    ; 数组字单元
COUNT  EQU    ($-LIST) / 2      ; 数组长度  $n=10/2=5$ 
DATA    ENDS

;
SORT    SEGMENT
        ASSUME CS: SORT, DS: DATA
BEGIN:  MOV    AX, DATA
        MOV    DS, AX
        MOV    CX, COUNT-1
        ; CX ← 比较轮数 (大循环次数)
LOOP1:  MOV    DX, CX; DX ← 大循环次数
        MOV    BX, 0    ; 地址指针
```



4.3.3 循环结构程序

例4.40

LOOP2:

```
MOV  AX, LIST [BX]      ; AX ← LIST (i)
CMP  AX, LIST [BX+2]    ; LIST(i) ≥ LIST(i+2)?
JAE  NO_CHANGE          ; 是, 转
XCHG AX, LIST [BX+2]    ; 否, 交换, 使大数在前
MOV  LIST [BX], AX
```

NO_CHANGE:

```
ADD  BX, 2              ; BX增2, 取下一个数
LOOP LOOP2              ; 一轮没比完, 转, 继续比
MOV  CX, DX              ; 一轮比完, CX ← 比□□数
LOOP LOOP1              ; CX ← CX-1, 非0则转下轮比较
MOV  AX, 4C00H           ; 比完, 返回DOS
INT  21H
```

SORT ENDS

END BEGIN



4.3.1 顺序结构程序设计

4.3.2 分支程序设计

4.3.3 循环结构程序

4.3.4 代码转换程序

4.3.5 过程调用



4.3.4 代码转换程序

- ◆ 在计算机中，经常需要将数据从一种形式转换成另一种形式。例如，把2进制数转换成10进制数，再转换成ASCII码显示出来；把键盘输入的10进制数转换成2进制数，再转换成16进制数等。这就要编写各种代码转换程序。
- ◆ 下面介绍几个代码转换程序，为方便起见，程序都以子程序的形式给出。



4.3.4 代码转换程序

例4.41 将AL寄存器中的二进制数转换成非压缩BCD数, 存入AX中, 再转换成ASCII码后在CRT上显示。

设AL中的初值为 $01100010B=62H$, 它等于10进制数的98。

将它除以10后, 可得商为9, 余数为8, 将其存放入AX中, 使 $AX=0908H$, 与 $3030H$ 相加 (也可相或), 即转换成ASCII码 $3938H$, 用2号DOS功能调用即可显示出来。



例4.41 4.3.4 代码转换程序


```
BIN_ASC PROC NEAR
    MOV     AH, 0
    MOV     BL, 10          ; 除数
    DIV     BL              ; AL ← 商(9), AH ← 余数(8)
    XCHG    AH, AL          ; AX=0908H(非压缩BCD数)
    ADD     AX, 3030H       ; AX=3938H (ASCII码)
    MOV     CX, AX          ; CX ← 3938H
    MOV     DL, CH
    MOV     AH, 2
    INT     21H             ; 显示9
    MOV     DL, CL
    MOV     AH, 2
    INT     21H             ; 显示8
    RET
BIN_ASC ENDP
```



4.3.4 代码转换程序

例4.42 将键盘输入的一个以回车符为结尾的10进制数（0~65535）转换成2进制数，并存入BX中，如输入一个非10进制数或回车符，则退出程序。

编程思想：

- 1) 利用DOS 1号功能调用，等待从键盘输入一个10进制数字，比如3，则在AL中得到3的ASCII码33H。
 - 2) 将ASCII码转换成BCD码。截下低4位，判断其是否为数字0~9，若是，则将该数存入BX中，若不是则退出程序。
 - 3) 再键入下一个数字，如数字5，也要判断其是否为数字0~9。
- 

4.3.4 代码转换程序

例4.42

- 4) 将10进制数转换成2进制数。将先键入的数字3乘以10后，与后键入的数5相加（累加），得 $(3 \times 10) + 5 = 35$ 。
- 5) 再键入第3个数字，如8，将前面累加的数乘以10后与后键入的数累加，可得到 $[(3 \times 10 + 5) \times 10] + 8 = 358$ ，还可继续进行下去，直至键入一个非10进制数或回车符为止。遇回车符表示键入的一个10进制数结束。



4.3.4 代码转换程序

例4.42

程序（子程序形式）如下：

```
DEC_BIN    PROC NEAR
```

```
    MOV     BX, 0
```

； BX存结果或中间结果，初值清0

GET_CHAR:

```
    MOV     AH, 1
```

； DOS 1号功能调用

```
    INT     21H
```

； AL ← 输入数字ASCII码

```
    CMP     AL, 0DH
```

； 是回车符

```
    JE      EXIT
```

； 是，转

```
    SUB     AL, 30H
```

； ASCII码转换成10进制数

```
    JL      EXIT
```

； <0（非数字），则退出

```
    CMP     AL, 9
```

； >0，则与9比较

```
    JG      EXIT
```

； >9，退出



4.3.4 代码转换程序

例4.42

CBW

； 是数字0~9，将AL中的字节转换成字，送到AX

XCHG AX, BX

； 将先键入的数（在BX中）→AX

MOV CX, 10

MUL CX ; 先键入数×10 → AX

XCHG AX, BX

； 交换后，AX → BX，新键入数 → AX

ADD BX, AX ; 累加，结果 → BX

JMP GET_CHAR ; 循环，键入新数

EXIT: RET

DEC_BIN ENDP

► 程序中也用到了分支结构和循环程序的设计方法。



4.3.4 代码转换程序

例4.43 编写将BX中的二进制数转换成16进制数，并在显示器上显示的程序。

- ▶ 由于每4位二进制数可用一个16进制数表示，所以BX中的二进制数可以转换成4个16进制数字。
- ▶ 每次将BX中的数左移4次，可得到一个16进制数，将其转换成ASCII码后，即可在显示器上显示出一个16进制数。
- ▶ 重复执行4次，就可将BX中的4个16进制数在显示器上显示出来。



4.3.4 代码转换程序

例4.43

◇ 操作过程如图4.11所示

图4.11 2进制数转换成16进制数的ASCII码



4.3.4 代码转换程序

例4.43

；程序入口已为BX存入了一个二进制数。

BIN_HEX **PROC** NEAR

MOV CH, 4

；转换后产生4个16进制数字（大循环次数）

ROTATE:	MOV CL, 4	；小循环次数（左移4次）
	ROL BX, CL	；对BX左移4次
	MOV AL, BL	；AL ← BL
	AND AL, 0FH	；截得一个16进制数字
	ADD AL, 30H	；加30H，转换成ASCII码
	CMP AL, 3AH	；与‘9+1’比，>9？
	JL DISPLAY	；≤9，转显示
	ADD AL, 7H	；>9，将数字0AH~0FH
		；转换成ASCII码



4.3.4 代码转换程序

例4.43

```
DISPLAY:  MOV DL, AL      ; DL←待□数字ASCII码
          MOV AH, 2
          INT 21H        ; 显示DL中数字
          DEC CH          ; 4个数字都显示完?
          JNZ ROTATE      ; 没有, 转大循环
          RET             ; 显示完, 退出
BIN_HEX  ENDP
```



4.3.4 代码转换程序

例4.44 将AX中的16位二进制数转换成4位压缩BCD数

- ◇ 方法很简单，只要将AX中的内容先后除以1000、100和10，每次得到的商即为BCD数的千位、百位和十位数，余数为个位数。
- ◇ 麻烦的是除法运算既要分字除和字节除，又要搞清楚每次运算时被除数、除数、商和余数分别放在什么寄存器中。
- ◇ 除法指令要求：
 - 源操作数为字时， $(DX, AX) / \text{源字}$ ，结果： $AX \leftarrow \text{商}$ ， $DX \leftarrow \text{余数}$
 - 字节除时， $AX / \text{源字节}$ ，结果： $AL \leftarrow \text{商}$ ， $AH \leftarrow \text{余数}$



4.3.4 代码转换程序

例4.44

- ◇ 下面给出转换程序, 为便于理解, 假设存放在AX中的16位二进制数的实际值为9346, 转换后应使AX=9346H (压缩BCD数), 注释中给出具体的转换步骤。

```
BIN_BCD    PROC NEAR
            CMP  AX, 9999      ; AX>9999?
            JBE  TRAN          ; 小于, 转
            JMP  EXIT          ; 大于, 转退出
TRAN:       SUB  DX, DX        ; DX初值清0
            MOV  CX, 1000      ; CX←1000
            DIV  CX             ; (DX,AX)/1000=9...346
                                   ; (AX=9, DX=346)
            XCHG AX, DX        ; 交换,使DX=9, AX=346
                                   ; (下次除法被除数)
```



4.3.4 代码转换程序

例4.44

MOV CL, 4	; 第一个商9左移4次
SHL DX, CL	; DX=0090H
MOV CL, 100	; CL←100
DIV CL	; 346/100=3...46, AL=3, AH=46
ADD DL, AL	; 将第2次的商加到DL中,
	; 使DX=0093H
MOV CL, 4	; DX左移4次
SHL DX, CL	; 左移后DX=0930H
XCHG AL, AH	; 交换, AX=0346H
SUB AH, AH	; AX=0046H,
	; 第2次余数做被除数
MOV CL, 10	; CL←10
DIV CL	; AX/10=4...6, 结果AL=4,AH=6



4.3.4 代码转换程序

例4.44

```
ADD    DL, AL    ; 4加到DL上,  
                ; 使DX=0934H  
  
MOV    CL, 4  
SHL    DX, CL    ; DX左移4次, DX=9340H  
ADD    DL, AH    ; 最后一次余数加到  
                ; DX上, DX=9346H  
  
MOV    AX, DX    ; 最后结果: AX=9346H
```

```
EXIT:  RET  
BIN_BCD ENDP
```



4.3.1 顺序结构程序设计

4.3.2 分支程序设计

4.3.3 循环结构程序

4.3.4 代码转换程序

4.3.5 过程调用



4.3.5 过程调用

- ◆ 汇编语言程序中把某些能完成特定功能而又经常要用到的程序段，编写成独立的模块，将它称为**过程**或**子程序**。
- ◆ 需要执行这段程序时就进行**过程调用**，执行完毕，再返回到原来调用它的主程序去。
- ◆ 采用过程调用结构编程，使程序结构清晰，语句简练不用重复编写某个程序段，也便于修改。子程序本身又可调用其它子程序，称为**子程序嵌套**。



4.3.5 过程调用

例4.45 用过程调用方法，编程实现将内存中4个BCD数相加，结果存入SUM开始的单元中去的运算。BCD数在内存中存放时，低字节在前，高字节在后。

- ▶ 由于每个BCD数各有4个字节，每两个字节相加的运算要重复4次。所以，这种运算可编写成子程序，供主程序调用。



4.3.5 过程调用

例4.45

```
DATA    SEGMENT
NUM_1   DB 44H, 33H, 22H, 11H    ; 第一个BCD数
NUM_2   DB 88H, 77H, 66H, 55H    ; 第二个BCD数
SUM      DB 5 DUP ( ? )          ; 存相加结果
DATA    ENDS

;
STACK   SEGMENT STACK            ; 堆栈段
        DW 50 DUP ( ? )

TOP      LABEL WORD
STACK    ENDS

;
CODE     SEGMENT                  ; 代码段
MAIN     PROC FAR                 ; 主过程
        ASSUME CS: CODE, DS: DATA, SS: STACK
```



4.3.5 过程调用

例4.45

START: MOV	AX, STACK	; 设置SS: SP
MOV	SS, AX	
MOV	SP, OFFSET TOP	
PUSH	DS	
SUB	AX, AX	
PUSH	AX	
MOV	AX, DATA	
MOV	DS, AX	
MOV	ES, AX	
LEA	SI, NUM_1	; SI ← 数1偏移地址
LEA	BX, NUM_2	; BX ← 数2偏移地址
LEA	DI, SUM	; DI ← 和数偏移地址
CLD		; 清方向标志
CLC		; 清进位标志



4.3.5 过程调用

例4.45

```
        MOV     AH, 0           ; AH存最后一次进位, 初值清0
        MOV     CX, 4           ; 做4次加法运算
LOOP1:
        CALL    ADD_B           ; 调用过程 (4次)
        LOOP    LOOP1           ; 没完, 继续
        ADC     AH, 0           ; 已完, 进位加到AH中
        MOV     AL, AH
        STOSB                    ; 进位存入SUM+4单元
        RET                      ; 返回DOS
MAIN    ENDP                    ; 主过程结束
```

; 子程序ADD_B (见下页)



4.3.5 过程调用

例4.45

； 子程序ADD_B

```
ADD_B PROC NEAR    ； 单字节加法子程序
    LODSB          ； AL ← 数1中取一字节，SI自动增1
    ADC AL, [BX]    ； 与数2带进位加
    DAA             ； BCD数调整
    STOSB           ； 存入SUM开始的单元中，
                    ； DI自动增1
    INC BX          ； 调整数2的地址指针
    RET             ； 返回主程序
ADD_B ENDP
CODE ENDS
END MAIN
```



例4.46 内存中有两个数组ARY1和ARY2，数组长度为20和10，要求编写一个程序，分别累加两个数组的值，存入SUM1和SUM2开始的单元中，低字节在前，高字节在后。

- ▶ 累加第1个数组值时，要做20次加法，加法可用子过程实现；累加第2个数组时，要做10次加法，加法也可调相同的子过程来完成，但两次调用前的入口参数和存放结果的单元不同。

； 数据段

DATA **SEGMENT** ； 数据段

ARY1 DB 20 DUP (?) ； 数组1， 20个随机数

SUM1 DB 2 DUP (?) ； 存数组1各数相加之和

ARY2 DB 10 DUP (?) ； 数组2， 10个随机数

SUM2 DB 2 DUP (?) ； 存数组2相加之和

DATA **ENDS**



例 4.46

； 堆栈段

```
STACK SEGMENT STACK
      DW 50 DUP ( ? )
      TOP LABEL WORD
STACK ENDS
```

；

```
CODE SEGMENT ; 代码段
```

```
MAIN PROC FAR ; 主程序
```

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

```
BEGIN:
```

```
      MOV AX, STACK
```

```
      MOV SS, AX
```

```
      MOV SP, OFFSET TOP
```

```
      PUSH DS
```

```
      SUB AX, AX
```

```
      PUSH AX
```



例4.46

MOV AX, DATA

MOV DS, AX

LEA SI, ARY1 ; 转子前入口参数,
; SI←ARY1首地址

MOV CX, LENGTH ARY1 ; CX←ARY1长度

MOV BX, OFFSET SUM1 ; BX←和□元首址

CALL SUM ; 转子过程, 求数组1之和

LEA SI, ARY2 ; 转子前设ARY2之入口参数

MOV CX, LENGTH ARY2

MOV BX, OFFSET SUM2

CALL SUM ; 转子过程, 求数组2之和

RET ; 返回DOS

MAIN ENDP ; 主程序结束

;

; 子程序SUM (见下页)

;



例4.46

```
SUM  PROC  NEAR          ; 求和子过程SUM
      XOR   AL, AL        ; AX清0, CF标志清0
      MOV   AH, 0         ; AH存进位, 初值清0
LOOP1:
      ADC   AL, [SI]       ; 数组中取一元素,
                          ; 带进位累加到AL
      ADC   AH, 0         ; 进位累加到AH中
      INC   SI            ; 修改地址指针
      LOOP  LOOP1         ; 未完, 继续
      MOV   [BX], AL      ; 已处理完, 存和数
      MOV   [BX+1], AH    ; 存进位累加值
      RET
SUM  ENDP                ; SUM子过程结束
;
CODE ENDS
      END  MAIN          ; 整个程序结束
```



4.3.5 过程调用

例4.47 编写显示回车换行子程序。

```
CRLF  PROC    NEAR
      MOV     DL, 0DH      ; 回车符
      MOV     AL, 2
      INT     21H
      MOV     DL, 0AH      ; 换行符
      MOV     AH, 2
      INT     21H
      RET
CRLF  ENDP
```



4.3.5 过程调用

例4.48 编写从键盘输入8个10进制数，将它转换成16进制数后在屏幕上显示的程序。首先从键盘输入一个10进制数（0~65536），该数以回车符结束，然后将它转换成16进制数的ASCII码，在显示器上显示出来。重复8次，即可在屏幕上显示8个16进制数。

► 编程时，只要编写一个主程序，再调前面介绍的相关程序，包括：

例4.42 DEC_BIN程序：将键入10进制数转换成ASCII码，再转换成二进制数，结果存入BX中。

例4.43 BIN_HEX程序：将BX中的二进制数转换成16进制数，再转换成ASCII码，显示出来。

例4.47 CRLF显示回车换行子程序。



4.3.5 过程调用

例4.48

```
DEC_HEX SEGMENT                ; 10进制转换成16进制数程序
    ASSUME CS: DEC_HEX
MAIN PROC FAR                  ; 主程序
    |
    MOV     CX, 8              ; 调用8次子程序
    PUSH    CX                 ; CX入栈保护
REPT: CALL   DEC_BIN           ; 10进制 二进制, 结果在BX中
      CALL   BIN_HEX          ; 二进制 16进制及其ASCII码并显示
      CALL   CRLF             ; 显示16进制数后回车、换行
    POP     CX                 ; 堆栈中弹出CX, 初值为, 逐次减1
    DEC     CX                 ; CX CX-1
    PUSH    CX                 ; 减1后的CX入栈
    CMP     CX, 0              ; CX=0?
    JNE     REPT               ; 非0则转
    RET                       ; 是0则退出
MAIN ENDP                      ; 主程序结束
```



例 4.48

； 10进制数转换成二进制数，结果存BX

DEC_BIN **PROC** NEAR ；

⋮

RET

DEC_BIN **ENDP**

； 将BX中的二进制数转换成16进制数并显示

BIN_HEX **PROC** NEAR

⋮

RET

BIN_HEX **ENDP**

； 回车换行子程序

CRLF **PROC** NEAR

⋮

RET

CRLF **ENDP**

； 代码段结束

DEC_HEX **ENDP**

END **MAIN**

