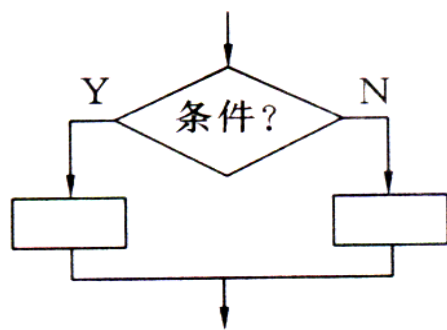


第7章 汇编语言程序设计

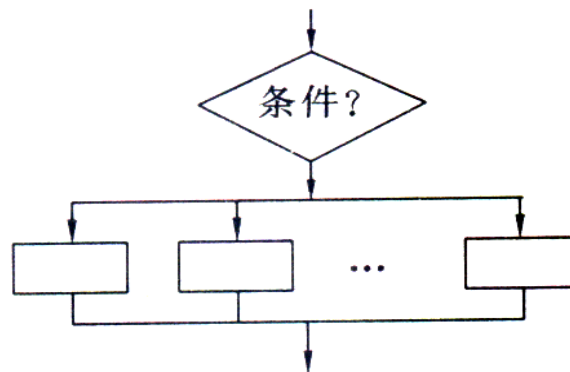
- 7.1 概述
- 与高级语言程序相比，汇编语言程序具有更高的效率，执行时间短，占用内存少，在实时领域得到广泛应用。
- 汇编语言程序设计的基本步骤：
 - (1) 描述问题
 - (2) 确定算法。
 - (3) 绘制流程图。
 - (4) 分配存储空间和工作单元。
 - (5) 编写程序。
 - (6) 上机调试。
- 程序基本结构形式：顺序、分支、循环、子程序。

7.2 分支程序设计

- **分支程序**：根据不同的情况或条件执行不同的功能，具有判断和转移功能，在程序中利用条件转移指令对运算结果的状态标志进行判断实现转移。
- **特点**：按条件判断、转移、分支。
- **7.2.1 分支程序的结构形式**
- **分支程序有2种结构形式**：二路分支结构、多路分支结构。
- 两种结构的**共同特点**：在某一特定条件下，只执行多个分支中的一个分支。



二路分支结构



多路分支结构

7.2.2 二路分支程序设计方法

- 例7.2.1: 有一函数, 任意给定自变量X值 ($-128 \leq X \leq 127$), 当 $X \geq 0$ 时函数值 $Y=1$ 、当 $X < 0$ 时 $Y= -1$; 设给定的X值存入变量XX1单元, 函数Y值存入变量YY1单元。

DATA SEGMENT

XX1 DB X

YY1 DB ?

DATA ENDS

{
Y=1 , 当 $X \geq 0$
Y= -1, 当 $X < 0$

CODE SEGMENT

ASSUME CS:CODE, DS:DATA

START: MOV AX, DATA }

MOV DS, AX }

MOV AL, XX1 ; 取X

CMP AL, 0 ; 判0

JNS AA2 ; $X \geq 0$ 转AA2

MOV AL, 0FFH ; 否则 ($X < 0$) , AL= - 1

JMP AA1

AA2: MOV AL, 1

AA1: MOV YY1, AL ; 存结果

MOV AH, 4CH }

INT 21H }

CODE ENDS

END START

7.2.3 多路分支程序设计方法

- 三种方法：逻辑分解法、地址表法、转移表法。
- 例7.2.2：某工厂有5种产品的加工程序1到5，分别存放在WORK1、WORK2...WORK5为首地址的内存区域中，分别键入1、键入2...键入5选择其对应的加工程序。
- 问题解析：
 - （1）WORK1—WORK5分别对应5个加工程序。
 - （2）按1—5的输入代码转入相应程序。
- 用3种方法实现：逻辑分解法、地址表法、转移表法

(1) 逻辑分解法

- 方法：逻辑判断、直接转移到目标程序。

```
CODE    SEGMENT
        ASSUME CS:CODE
START:  MOV    AH, 1    ; 键盘输入
        INT     21H
        CMP     AL, 31H
        JZ      WORK1 ; 为1, 转WORK1
-----
        CMP     AL, 32H
        JZ      WORK2 ; 为2, 转WORK2
-----
        CMP     AL, 33H
        JZ      WORK3 ; 为3, 转WORK3
-----
        CMP     AL, 34H
        JZ      WORK4 ; 为4, 转WORK4
-----
        CMP     AL, 35H
        JZ      WORK5 ; 为5, 转WORK5
-----
        JMP     WORK0 ; 都不是, 退出
```

```
WORK1:  ;
WORK2:  ;
WORK3:  ;
WORK4:  ;
WORK5:  ;
WORK0:  MOV    AH, 4CH
        INT     21H
CODE    ENDS
        END     START
```

(2) 地址表法

- **计算:** $\text{表地址} = \text{表首址} + (\text{键号} - 1) \times 2$
- **方法:** 目标地址列表, 然后查表, 间接转移到目标程序。

```
DATA SEGMENT
TABLE DW WORK1,WORK2,WORK3,WORK4,WORK5; 地址表
DATA ENDS
```

```
CODE SEGMENT
    ASSUME CS:CODE, DS:DATA
START: MOV AX, DATA }
      MOV DS, AX     }
      LEA BX, TABLE ; 取EA
      MOV AH, 1      ; 键盘输入
      INT 21H        }
      AND AL, 0FH    ; 屏蔽高4位
      DEC AL         ; 键号减1
      ADD AL, AL      ; 键号乘2
      SUB AH, AH      ; AH清0
      ADD BX, AX      ; 形成表地址
      JMP WORD PTR [BX]; 间接转移
```

```
WORK1:  |
WORK2:  |
WORK3:  |
WORK4:  |
WORK5:  |
      |
      MOV AH, 4CH }
      INT 21H     }
CODE ENDS
END START
```

(3) 段内转移表法

- 段内**短转移指令**占2个字节，计算：表地址=表首址+（键号-1）× 2
- 段内**近转移指令**占3个字节，计算：表地址=表首址+（键号-1）× 3
- **方法**：转移指令列表，通过执行转移指令最终执行目标程序。

CODE SEGMENT

ASSUME CS:CODE

START: **LEA** **BX, WORK** ; 转移表首址送BX

MOV **AH, 1** ; 键盘输入

INT **21H**

AND **AL, 0FH**

DEC **AL** ; 键号减1

(键号-1) { **MOV** **AH, AL**

× 3 { **ADD** **AL, AL** ; 键号乘2

ADD **AL, AH** ; 键号乘3

SUB **AH, AH**

ADD **BX, AX** ; 形成表地址

JMP **BX** ; 转移到列表

WORK: **JMP** **NEAR PTR WORK1**; 转移表

JMP **NEAR PTR WORK2**

JMP **NEAR PTR WORK3**

JMP **NEAR PTR WORK4**

JMP **NEAR PTR WORK5**

WORK1: ;

WORK2: ;

WORK3: ;

WORK4: ;

WORK5: ;

;

MOV **AH, 4CH**

INT **21H**

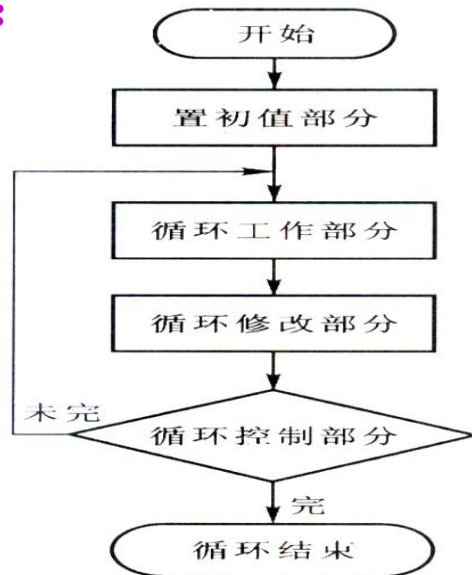
CODE **ENDS**

END **START**

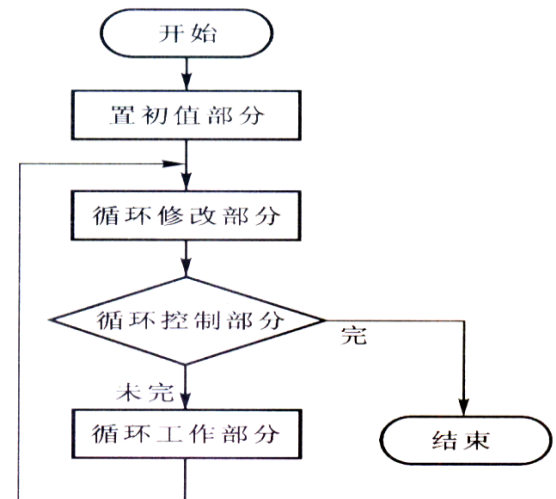
7.3 循环程序设计

- 7.3.1 循环程序的结构形式
- 循环程序由五部分组成：
 - 初始化部分：设置初值。
 - 循环工作部分：核心。
 - 循环修改部分：修改循环工作变量、指针。
 - 循环控制部分：修改控制变量，确定程序走向。
 - 循环结束部分：结果处理。
- 常用结构2种：

(1) 先执行，后判断。至少执行一次循环体。



(2) 先判断，后执行。允许零次循环。



7.3.2 循环程序的设计方法

- **循环条件控制方法3种：**计数控制、条件控制（判断数据）、变量控制（判断地址，变量地址）
- **例7.3.1：**某个数据段中，从偏移地址1000H单元开始，连续存放**255个8位无符号**整数x1、x2、...x254、x255。编写程序**求这些数据的和**，并将结果存入SUM1单元。
- 8位无符号数在0~255之间，255个数的和一定是一个不超过16位二进制的数。所以存放和的单元取字单元。
- **过程：**AX为累加和，先清0，DL为相应的“加”数，DH清0，实现
 $AX \leftarrow AX + DX$

1. 计数控制循环

- 方法：计数循环次数，按计数次数判断。
- (1) 先执行，后判断

DATA	SEGMENT		
	ORG	1000H	
NUMBER1	DB	x1, x2, ...x254, x255	
SUM1	DW	?	
DATA	ENDS		
CODE	SEGMENT		
	ASSUME	CS:CODE, DS:DATA	
START:	MOV	AX, DATA	}
	MOV	DS, AX	
	LEA	BX, NUMBER1	; 取数据EA
	MOV	AX, 0	; 累加和清0
	MOV	DH, 0	; 使用DX, 故DH先清0
	MOV	CL, 255	; 计数初值
AA1:	MOV	DL, [BX]	; 取数
	ADD	AX, DX	; 求和, $AX \leftarrow AX + DX$
	INC	BX	; 修改地址指针

	SUB	CL, 1	; 计数
	JNZ	AA1	; 判转
	MOV	SUM1, AX	; 存和
	MOV	AH, 4CH	}
	INT	21H	
CODE	ENDS		
	END	START	

计数控制循环

- (2) 先判断，后执行

```
DATA SEGMENT
    ORG 1000H
NUMBER1 DB x1, x2, ...x254, x255
SUM1 DW ?
DATA ENDS
CODE SEGMENT
    ASSUME CS:CODE, DS:DATA
```

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

```
      LEA BX, NUMBER1-1 ; 因为先减1判断，故首址调整
```

```
      MOV AX, 0
```

```
      MOV DH, 0
```

```
      MOV CL, 0 ; 因为先减1判断，故计数初值调整为256
```

```
AA1: INC BX ; 修改地址指针
```

```
      SUB CL, 1 ; 第一次CL - 1= 255，即0FFH
```

```
      JZ AA2 ; 判转
```

```
      MOV DL, [BX] ; 取数
```

```
      ADD AX, DX ; 求和
```

```
      JMP AA1
```

```
AA2: MOV SUM1, AX; 存和
      MOV AH, 4CH }
      INT 21H
CODE ENDS
      END START
```

2. 条件控制循环

- 方法：按终止数据或字符判断
- (1) 先执行，后判断

```
DATA    SEGMENT
        ORG      1000H
NUMBER1 DB      x1, x2, ...x254, x255
SUM1    DW       ?
DATA    ENDS
CODE    SEGMENT
        ASSUME CS:CODE, DS:DATA
START:  MOV      AX, DATA  }
        MOV      DS, AX    }
        LEA      BX, NUMBER1 ; 取EA
        MOV      AX, 0
        MOV      DH, 0
AA1:    MOV      DL, [BX]    ; 取数
        ADD      AX, DX     ; 求和
        INC      BX        ; 修改地址指针
        CMP      DL, x255   ; 终止数据
        JNZ      AA1        ; 判转
        MOV      SUM1, AX   ; 存和
        MOV      AH, 4CH    }
        INT      21H        }
CODE    ENDS
        END      START
```

条件控制循环

- (2) 先判断，后执行

```
DATA SEGMENT
    ORG 1000H
NUMBER1 DB x1, x2, ...x254, x255
SUM1 DW ?
DATA ENDS
CODE SEGMENT
```

```
AA2: MOV SUM1, AX; 存和
      MOV AH, 4CH }
      INT 21H
CODE ENDS
      END START
```

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

```
START: MOV AX, DATA }
      MOV DS, AX }
      LEA BX, NUMBER1-1
      MOV AX, x255
      MOV DH, 0
```

; 先将终止数据x255送累加和，相当于已累加

```
AA1: INC BX ; 修改地址指针
      MOV DL, [BX]
      CMP DL, x255 ; 终止数据
      JZ AA2 ; 判转
      ADD AX, DX ; 求和
      JMP AA1
```

3. 变量控制循环

- **方法：**按存储器偏移地址判断。最后数x255的偏移地址应为**10FEH**。
- (1) 先执行，后判断

DATA	SEGMENT		
	ORG	1000H	
NUMBER1	DB	x1, x2, ...x254, x255	
SUM1	DW	?	
DATA	ENDS		
CODE	SEGMENT		
	ASSUME CS:CODE, DS:DATA		
START:	MOV	AX, DATA	}
	MOV	DS, AX	
	LEA	BX, NUMBER1	
	MOV	AX, 0	
	MOV	DH, 0	
<hr/>			
AA1:	MOV	DL, [BX]	; 取数
	ADD	AX, DX	; 求和
	INC	BX	; 修改地址指针
	CMP	BX, 10FFH	; 存放x255之后的地址, 判断结束的地址
	JNZ	AA1	; 判转

	MOV	SUM1, AX;	存和
	MOV	AH, 4CH	}
	INT	21H	
CODE	ENDS		
	END	START	

变量控制循环

- (2) 先判断，后执行

```
DATA    SEGMENT
        ORG    1000H
NUMBER1 DB  x1, x2, ...x254, x255
SUM1    DW    ?
DATA    ENDS
CODE    SEGMENT
        ASSUME CS:CODE, DS:DATA
START:  MOV    AX, DATA }
        MOV    DS, AX   }
        LEA    BX, NUMBER1-1 ; 因为先加1判断，故首址调整
        MOV    AX, 0
        MOV    DH, 0
AA1:    INC    BX          ; 修改地址指针
        CMP    BX, 10FFH  ; 结束地址
        JZ     AA2
        MOV    DL, [BX]   ; 判转
        ADD    AX, DX     ; 求和
        JMP    AA1
AA2:    MOV    SUM1, AX; 存和
        MOV    AH, 4CH }
        INT    21H
CODE    ENDS
        END    START
```

7.3.3 单重循环

- 循环程序基本模式：单重循环、多重循环。
- 例7.3.2：求无符号整数平方根整数部分。依据公式：

$$n^2 = 1 + 3 + 5 + \dots + (2n-1)$$

- 方法：计数减去“递增奇数”的次数。

<pre>DATA SEGMENT NUMBER1 DW X ; 原数据 ROOT1 DW ? ; 平方根 DATA ENDS CODE SEGMENT ASSUME CS:CODE, DS:DATA START: MOV AX, DATA } MOV DS, AX MOV AX, NUMBER1 ; 取X MOV CX, -1 ; 次数初值送循环控制计数器CX MOV BX, -1 ; 奇数初值 ----- AA1: ADD BX, 2 ; 形成奇数1、3、5... SUB AX, BX ; 减奇数 INC CX ; 计次数, INC对CF无影响 JNC AA1 ; AX ≥ BX, 转AA1</pre>	<pre>MOV ROOT1, CX; 存根 MOV AH, 4CH } INT 21H CODE ENDS END START</pre>
---	---

判断依据

单重循环-1

- 例7.3.3: 编制在CRT上显示“中”字形的源程序。
- 方法: 利用DOS服务向显示器输出字符。

```
DATA                SEGMENT
NUMBER1 DB          0AH, 0DH                ; 0AH换行、0DH回车
          DB          '    A    ', 0AH, 0DH ; 单引号内有6个空格
          DB          'AAAAAAA', 0AH, 0DH ; 每行9个字节, 共7行
          DB          'A  A  A', 0AH, 0DH
          DB          'A  A  A', 0AH, 0DH
          DB          'AAAAAAA', 0AH, 0DH
          DB          '    A    ', 0AH, 0DH
          DB          '    A    ', 0AH, 0DH
DATA                ENDS
CODE                SEGMENT
          ASSUME CS:CODE, DS:DATA
START: MOV    AX, DATA }
      MOV    DS, AX    }
      LEA    BX, NUMBER1 ; 取EA
      MOV    CX, 65      ; 计数初值为
                          ; 7×9+2=65

AA1:  MOV    DL, [BX]
      INC    BX
      MOV    AH, 2; 显示
      INT    21H
      LOOP   AA1
      MOV    AH, 4CH }
      INT    21H    }
CODE                ENDS
END                START
```

单重循环-2

- **例7.3.4:** 在CRT上显示**16位**十进制数8988998899989899。
- **方法:** 逻辑尺控制分支。利用逻辑尺的位状态（0、1）判断分支，根据逻辑尺长度控制循环。

DATA SEGMENT

RULER1 DW 0100110011101011B ; 逻辑尺, 16位

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE, DS:DATA

START: MOV AX, DATA }

MOV DS, AX }

MOV BX, RULER1

MOV CX, 16

AA1: MOV DL, 38H; 8的ASCII码

SHL BX, 1 ; 逻辑左移

JNC AA2 ; 为0, 显示8

INC DL ; 形成9的ASCII码

AA2: MOV AH, 2 }

INT 21H }

LOOP AA1

MOV AH, 4CH }

INT 21H }

CODE ENDS

END START

7.3.4 多重循环

- **多重循环：**循环体内嵌套循环。
- **例7.3.5：**用逻辑尺方法，在CRT上显示“中”字。
- “中”字图形每行7个字符，“显示空格”取0、“显示A”取1。
- 从高位开始，按操作顺序设定8位标志，最低位取0，**逻辑尺为：**
00010000 → 10H
11111110 → FEH
10010010 → 92H
10010010 → 92H
11111110 → FEH
00010000 → 10H
00010000 → 10H
- 将逻辑尺数据依次存放在RULER1单元中。
- **本例是两重循环：**
 - (1) **外层：**依次取出7把逻辑尺给内层。
 - (2) **内层：**从高位开始，识别逻辑尺状态，0显示空格，1显示A。

例7.3.5 --1

DATA SEGMENT

RULER1 DB 10H, 0FEH, 92H, 92H, 0FEH, 10H, 10H; 7个逻辑尺

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE, DS:DATA

**START: MOV AX, DATA
MOV DS, AX }**

例7.3.5 --2

	LEA	SI, RULER1	; 逻辑尺EA
	MOV	CX, 7	; 外层循环计数初值, 7个逻辑尺
AA1:	MOV	DH, 8	; 内层循环计数初值, 每个逻辑尺8位
	MOV	DL, 0AH	; 换行
	MOV	AH, 2	}
	INT	21H	
	MOV	DL, 0DH	; 回车
	MOV	AH, 2	}
	INT	21H	
	MOV	BL, [SI]	; 取内层循环用的逻辑尺
AA2:	MOV	DL, 20H	; 空格的ASCII码
	SHL	BL, 1	; 逻辑左移
	JNC	AA3	; 判逻辑尺的标志位
	ADD	DL, 21H	; 形成A的ASCII码, 20H+21H=41H
AA3:	MOV	AH, 2	}; 显空格(0)或显A(1)
	INT	21H	
	DEC	DH	; 内层循环计数控制
	JNZ	AA2	
	INC	SI	; 下一个逻辑尺, 修改外层变量指针
	LOOP	AA1	; 外层循环计数控制

例7.3.5 --3

```
MOV    DL, 0AH        ; 换行
MOV    AH, 2          }
INT     21H            }
MOV    DL, 0DH        ; 回车
MOV    AH, 2          }
INT     21H            }
-----
MOV    AH, 4CH        }
INT     21H            }
CODE   ENDS
END     START
```

多重循环 --1

- **例7.3.6:** 设有5个学生参加4门课的考试，其中4门课的成绩以**压缩BCD**码方式存放在字组**COURSE1**中（设每门课的成绩满分为99分）；编制计算每个学生总分（总分存入字组**NUM1**中）的源程序。
- 多记录，多项目，累计

学生1 学生5

```
DATA SEGMENT
COURSE1 DB 70H, 88H, 92H, 90H, 99H ; 第一门课成绩
          DB 67H, 77H, 88H, 76H, 69H ; 第二门课成绩
          DB 74H, 87H, 77H, 74H, 70H
          DB 99H, 97H, 94H, 98H, 96H
NUM1 DW 5 DUP(0)
DATA ENDS

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA
START: MOV AX, DATA
        MOV DS, AX
        LEA SI, COURSE1 ; 取EA
        LEA DI, NUM1 ; 取目标EA
```

多重循环 --2

	SUB	SI, 5	； COURSE1首址减5以便循环，初值
	MOV	CL, 5	； 外层循环计数初值，5个学生
AA1:	MOV	BX, SI	； 形成某个学生第一门课成绩的首址减5
	SUB	AX, AX	； 清和寄存器
	MOV	CH, 4	； 内循环计数初值，4门课
AA2:	ADD	BX, 5	； 形成某个学生1、2...4门课成绩地址
	ADD	AL, [BX]	； 按BCD码求和
	DAA		； 压缩BCD码加调整
	ADC	AH, 0	； 累加进位
	DEC	CH	； 内循环计数控制
	JNZ	AA2	
	MOV	[DI], AX	； 存总分
	INC	SI	； 形成下个学生第一门课成绩的首址减5
	ADD	DI, 2	； 形成下个学生的总分地址
	DEC	CL	； 外循环计数控制
	JNZ	AA1	
	MOV	AH, 4CH	
	INT	21H	
CODE	ENDS		
	END	START	

前面使用**SUB SI, 5**的原因

多重循环 --3

- **例7.3.8:** 无符号数**从大到小**排序。从首地址AA1开始存放N个无符号字节数据，用冒泡法编制将数据按由大到小重新排序的源程序。

(1) **冒泡排序法:** N个数据，相邻两个单元大小判断，位置交换，**N-1**次后，**排出最大（或最小）数据**。经过**N-1**次的上述循环，便可实现数据排序。

(2) **关键步骤:** 比较，判断，交换。

(3) **判断指令:**

顺序	有符号	无符号
由大到小	大于 JGE	高于 JAE JNC
由小到大	小于 JLE	低于 JBE

小地址	最大 ///////
大地址	最小 ///////

由大到小排序

小地址	最小 ///////
大地址	最大 ///////

由小到大排序

例7.3.8 - ① N-1遍，每遍比较N-1次

```
DATA    SEGMENT
NUMBER1 DB    100, 3, 90, 80, 99, 77, 44, 66, 50 ; 9个数据
N        EQU    $- NUMBER1-1 ; 数据个数减1 (8个), $是汇编指针
DATA    ENDS

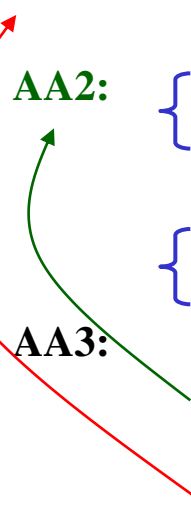
CODE    SEGMENT
        ASSUME CS:CODE, DS:DATA

START:  MOV     AX, DATA }
        MOV     DS, AX   }
        MOV     DX, N    ; 外循环计数初值, 即遍数

AA1:    LEA     BX, NUMBER1 ; 数据首址
        MOV     CX, N      ; 内循环计数初值, 即比较次数
AA2:    { MOV    AL, [BX]
        { CMP    AL, [BX+1] ; 相邻两个单元比较
        { JNC    AA3        ; 前大后小转 (小地址数大, 大地址数小)
        { XCHG   AL, [BX+1] ; 前小后大, 交换
        { MOV    [BX], AL
AA3:    INC     BX        ; 形成下一个比较地址
        LOOP    AA2       ; 内循环计数判转, 即比较次数
        DEC     DX        ; 外循环计数, 即计遍数
        JNZ     AA1       ; 外循环判转

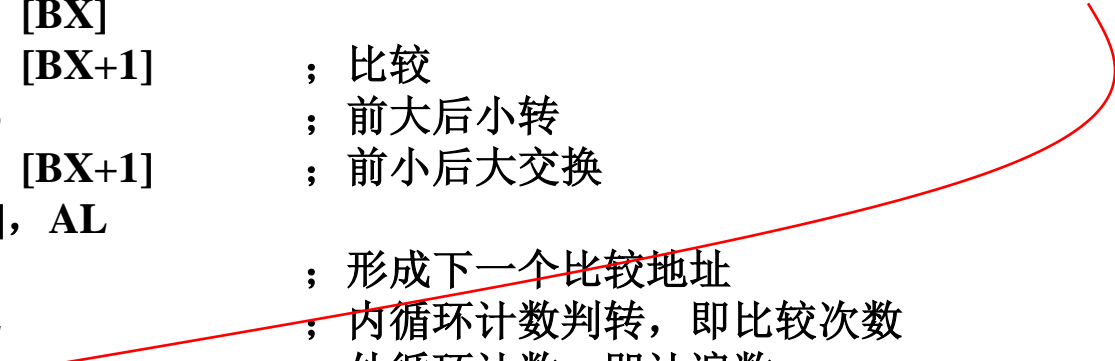
        MOV     AH, 4CH }
        INT     21H      }

CODE    ENDS
        END        START
```



例7.3.8 - ② N-1遍，每遍比较次数少1

```
DATA    SEGMENT
NUMBER1 DB    100, 3, 90, 80, 99, 77, 44, 66, 50 ; 9个数据
N        EQU    $- NUMBER1-1 ; 数据个数减1 (8个), $是汇编指针
DATA    ENDS
CODE    SEGMENT
        ASSUME CS:CODE, DS:DATA
START:  MOV     AX, DATA
        MOV     DS, AX
        MOV     DX, N ; 外循环计数初值, 即遍数
AA1:    LEA     BX, NUMBER1 ; 数据首址
        MOV     CX, DX ; 内循环计数初值, DX内容每循环一次减1
AA2:    MOV     AL, [BX]
        CMP     AL, [BX+1] ; 比较
        JNC     AA3 ; 前大后小转
        XCHG    AL, [BX+1] ; 前小后大交换
        MOV     [BX], AL
AA3:    INC     BX ; 形成下一个比较地址
        LOOP    AA2 ; 内循环计数判转, 即比较次数
        DEC     DX ; 外循环计数, 即计遍数
        JNZ     AA1 ; 外循环判转
        MOV     AH, 4CH
        INT     21H
CODE    ENDS
        END     START
```



例7.3.8 - ③ 有交换标志，N-1遍，每遍比较次数少1

； 同前，数据段定义

```
CODE    SEGMENT
        ASSUME CS:CODE, DS:DATA
START:  MOV     AX, DATA
        MOV     DS, AX
        MOV     DX, N                ; 外循环计数初值，即遍数
AA1:    LEA      BX, NUMBER1          ; 数据首址
        SUB     AH, AH                ; 设置交换标志，初值0
        MOV     CX, DX                ; 内循环计数初值，即比较次数
AA2:    MOV     AL, [BX]
        CMP     AL, [BX+1]            ; 比较
        JNC     AA3                  ; 前大后小转
        XCHG    AL, [BX+1]            ; 前小后大交换
        MOV     [BX], AL
        MOV     AH, 1                ; 设置交换标志=1
AA3:    INC     BX                    ; 形成下一个比较地址
        LOOP    AA2                  ; 内循环计数判转，即比较次数
        OR      AH, AH                ; 交换标志的反码送ZF标志位
        JZ      AA4                  ; 无交换转
        DEC     DX                    ; 外循环计数，即计遍数
        JNZ     AA1                  ; 外循环判转
AA4:    ;
        END     START
```

设置

改变

判断

7.4 子程序设计

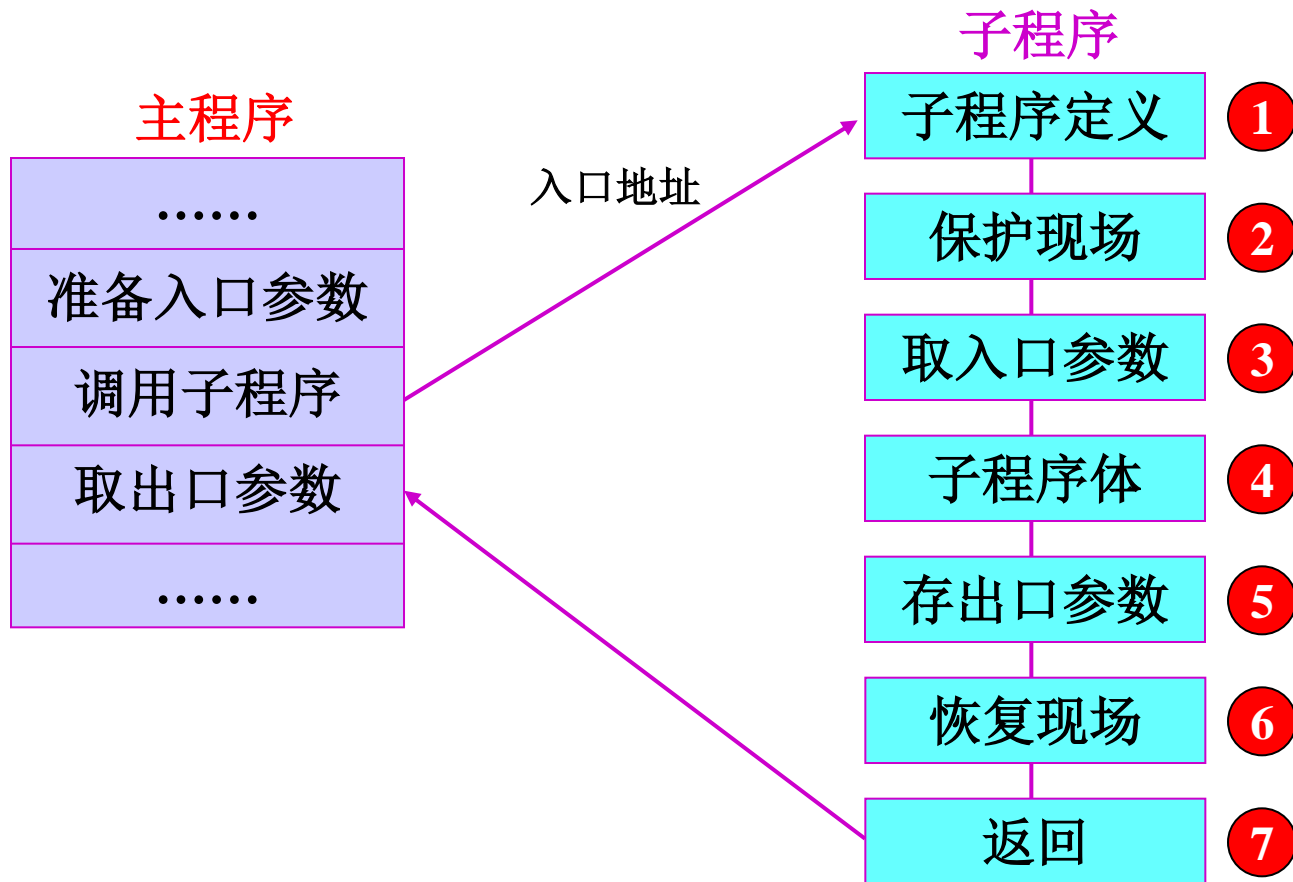
- **子程序（过程）**：可以被多次反复调用、能完成指定功能的程序段。
- **关键**：返回地址的保护与返回，涉及CALL、RET、堆栈。
- **子程序的说明文件**：
 - (1) **子程序名**：供调用子程序时使用。
 - (2) 子程序的**功能**：子程序完成的任务。
 - (3) 子程序**占用的寄存器和工作单元**：子程序执行时需要使用哪些寄存器；子程序执行后，各寄存器的变化情况。
 - (4) 子程序的**输入参数**：子程序运行所需的参数以及存放位置。
 - (5) 子程序的**输出参数**：子程序运行结束的结果参数及存放的位置。
 - (6) 子程序的运行时间：子程序运行所需的时钟周期数。
 - (7) 子程序的计算精度：子程序处理数据的位数。
 - (8) 子程序调用示例：子程序的调用格式。
 - (9) 可递归和可重入性：说明是否具有相关特性。

可递归调用：子程序能够调用其本身。

可重入性：子程序可被中断，且可被中断服务程序再次调用。

7.4.1 子程序的定义

- 子程序组成：7部分，子程序定义（即子程序入口）、保护现场、取入口参数、子程序体、存输出参数、恢复现场、返回。



子程序的定义

- 子程序定义格式:

过程名 **PROC** 属性

⋮

过程名 **ENDP**

- 属性: **NEAR** (默认)、**FAR**

- 过程属性确定原则:

- (1) 主程序和过程, **同段**, 用**NEAR**。
- (2) 主程序和过程, **不同段**, 用**FAR**。
- (3) 主程序可定义为**FAR**, 因为主程序可看作是**DOS**调用的一个子过程, 而**DOS**调用程序与主程序**不同段**。

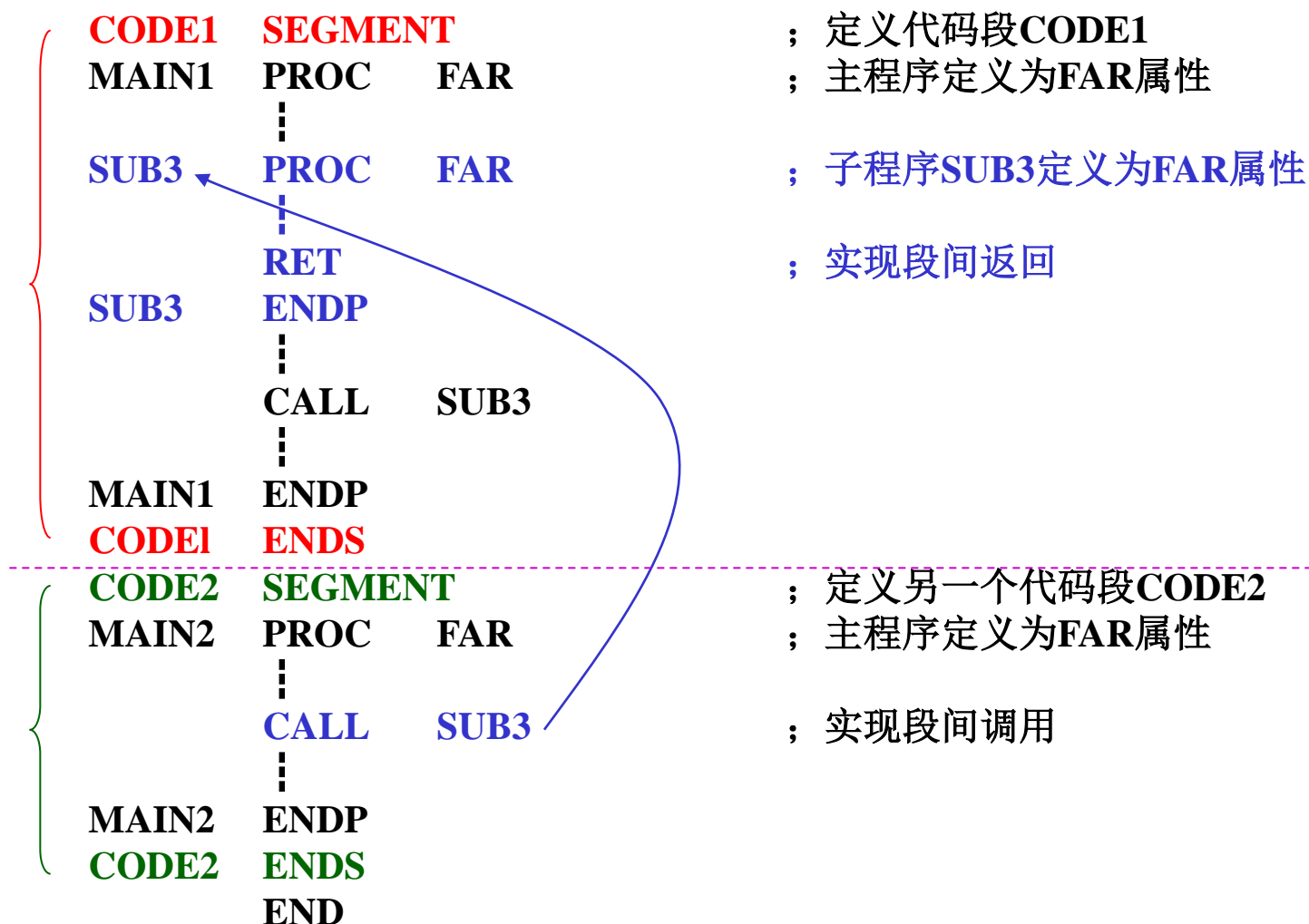
1. 主程序和子程序在同一代码段内

- 例7.4.1: 主程序MAIN和子程序SUB1, SUB2, 在同一代码段中。

CODE	SEGMENT	
MAIN	PROC FAR	; 主程序定义为FAR属性
	⋮	
	CALL SUB1	
	⋮	
SUB1	PROC NEAR	; 子程序SUB1定义为NEAR属性
	⋮	
	CALL SUB2	
	⋮	
	RET	
SUB2	PROC NEAR	; 子程序SUB2定义为NEAR属性
	⋮	
	RET	
SUB2	ENDP	
SUB1	ENDP	
	⋮	
	RET	
MAIN	ENDP	
CODE	ENDS	
	END	

2. 主程序和子程序不在同一个代码段

- 例7.4.2: 主程序MAIN2和子程序SUB3不在同一个代码段。



7.4.2 子程序的调用和返回

- 7.4.2 子程序的调用和返回
 - (1) 正确选择PROC属性。
 - (2) 注意堆栈状态。
- 7.4.3 现场的保存与恢复
 - 保护现场：在子程序中，保护那些主程序要用、而子程序也要用的寄存器内容。
 - 恢复现场：恢复现场保护的内容。
- 现场保护和现场恢复通常采用的方法：
 - (1) 使用堆栈指令
 - (2) 使用数据传送指令
 - (3) 使用PUSHA和POPA指令
 - (4) 使用PUSHAD和POPAD指令

1. 利用栈指令进行现场的保存与恢复

```
SUB4    PROC    NEAR
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        ⋮
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
SUB4    ENDP
```

2. 利用数据传送指令进行现场的保存与恢复

```
DATA    SEGMENT
BUFFER DW    4 DUP(?)
        ⋮
DATA    ENDS
CODE    SEGMENT
        ⋮
SUB5    PROC    NEAR
        LEA    DI, BUFFER
        MOV    [DI], AX
        MOV    [DI+2], BX
        MOV    [DI+4], CX
        MOV    [DI+6], DX
        ⋮
        LEA    DI, BUFFER
        MOV    AX, [DI]
        MOV    BX, [DI+2]
        MOV    CX, [DI+4]
        MOV    DX, [DI+6]
        RET
SUB5    ENDP
        ⋮
```

； 利用存储器保护

3. 用PUSHA和POPA指令进行现场的保存与恢复

```
SUB6  PROC  NEAR
      PUSHA
      ⋮
      POPA
      RET
SUB6  ENDP
```

4. 用PUSHAD和POPAD指令进行现场的保存与恢复

```
SUB7  PROC  NEAR
      PUSHAD
      ⋮
      POPAD
      RET
SUB7  ENDP
```

7.4.4 子程序的参数传送

- 参数传送：主程序和子程序之间的数据传送。
- 参数传送的三种方法：
 - (1) 通过寄存器传送参数
 - (2) 通过存储单元传送参数
 - (3) 通过堆栈传送参数或参数表地址

1. 通过寄存器传送参数

- **特点：**参数放在寄存器中，简便，实用，应用最多，适用于参数较少的情况。
- **例7.4.7：**用寄存器传送参数方式，编制键入**5位10进制数**加法程序。
- **设计说明：**
 - (1) **SUB8功能：**键盘输入，5位以内，10进制数→2进制数→CX，用非数字键结束本次数字输入。
10进制数→2进制数的转换算法：
$$((((D5 \times 10 + D4) \times 10 + D3) \times 10 + D2) \times 10 + D1) \times 10 + D0$$
 - (2) **SUB9功能：**CX（2进制数）→10进制数，在CRT上显示。
 - (3) 主程序**MAIN功能：**调用SUB8 →求和→调用SUB9。
约束条件：结果≤65535。
 - (4) MAIN、SUB8、SUB9同段，子程序可使用NEAR 定义。

例7.4.7 --1

```
CODE    SEGMENT
        ASSUME CS:CODE
MAIN    PROC    FAR                ; 主程序定义为FAR属性
        CALL    SUB8              ; 取被加数, 输出结果在CX中
        MOV     BX, CX            ; 保存被加数
        CALL    SUB8              ; 取加数
        ADD     CX, BX            ; 求和
        CALL    SUB9              ; 显示和
        MOV     AH, 4CH }
        INT     21H
```

例7.4.7 --2

SUB8 PROC NEAR ; 键入，十→二子程序SUB8，NEAR属性

PUSH AX
PUSH BX
PUSH DX

保护现场

XOR CX, CX ; 零送CX为每次键入值乘10作准备

AA2: MOV AH, 1
INT 21H ; 等待键入

CMP AL, 30H

JC AA1 ; 小于0键，返回

CMP AL, 3AH

JNC AA1 ; 大于9键，返回

键入非数字键，AL<'0'，
或AL≥'A'，返回

ADD CX, CX ; $CX \times 2 \rightarrow CX$

MOV BX, CX ; $CX \rightarrow BX$

ADD CX, CX ; $CX \times 2 \rightarrow CX$

ADD CX, CX ; $CX \times 2 \rightarrow CX$

ADD CX, BX ; CX原内容（上次键入值）乘10目的

$CX \times 10$

AND AX, 0FH ; 取本次键入值

ADD CX, AX ; 以前键入值逐乘10的值加本次值

$CX \times 10 + AX \rightarrow CX$ 类比
($D5 \times 10 + D4$)

JMP AA2 ; 转取下次键

AA1: POP DX
POP BX
POP AX

恢复现场

RET ; 返主，输出结果在CX中

SUB8 ENDP

例7.4.7 --3

SUB9 PROC NEAR ; 二→十, 显示, NEAR属性, 输入参数在CX中

PUSH AX }
PUSH BX }
PUSH DX }

CMP CX, 10000

JNC AA12 ; 判是否有万位, 避免最高位显0。

CMP CX, 1000

JNC AA4 ; 判是否有千位, 避免最高位显0。

CMP CX, 100

JNC AA6 ; 判是否有百位, 避免最高位显0。

CMP CX, 10

JNC AA8 ; 判是否有十位, 避免最高位显0。

JMP AA10 ; 只有个位, 转AA10

例7.4.7 --4

万

AA12: MOV DL, -1 ; 设置初值
AA3: SUB CX, 10000
INC DL ; 万位累计, 不影响CF
JNC AA3
ADD CX, 10000 ; 多减一次, 则补加, 恢复
OR DL, 30H ; 变为ASCII码
MOV AH, 2 ; 显万位
INT 21H }

千

AA4: MOV DL, -1
AA5: SUB CX, 1000
INC DL ; 千位累计, 不影响CF
JNC AA5
ADD CX, 1000 ; 多减一次, 则补加, 恢复
OR DL, 30H ; 变为ASCII码
MOV AH, 2 ; 显千位
INT 21H }

例7.4.7 --5

百

```
AA6:  MOV    DL, -1
AA7:  SUB     CX, 100
      INC     DL                ; 百位累计, 不影响CF
      JNC     AA7
      ADD     CX, 100           ; 多减一次, 则补加, 恢复
      OR      DL, 30H           ; 变为ASCII码
      MOV     AH, 2             ; 显百位
      INT     21H
```

十

```
AA8:  MOV    DL, -1
AA9:  SUB     CX, 10
      INC     DL                ; 十位累计, 不影响CF
      JNC     AA9
      ADD     CX, 10           ; 多减一次, 则补加, 恢复
      OR      DL, 30H           ; 变为ASCII码
      MOV     AH, 2             ; 显十位
      INT     21H
```

例7.4.7 --6

↑

```
AA10:  MOV    DL, CL
        OR     DL, 30H
        MOV    AH, 2          ; 显个位
        INT    21H
        -----
        POP    DX
        POP    BX
        POP    AX
        RET                    ; 返主
SUB9    ENDP
MAIN    ENDP
CODE    ENDS
        END    MAIN
```

2. 通过存储单元传送参数

- **特点：**开辟一个参数传递存储器，适合参数较多的情况。
- **例7.4.8：**用存储单元传送参数方式，编制**无符号多字节2进制数**的**求和程序**，并以**16进制数**形式显示在**CRT**上。
- **设计说明：**
 - (1) **SUB10功能：** $\text{NUMBER1} + \text{NUMBER2} \rightarrow \text{NUMBER3}$ 。参数传递采用存储单元传送方式。
 - (2) **SUB11功能：** 2进制数 \rightarrow 16进制数，在CRT上显示。参数传递采用利用寄存器传送参数表地址方式。
 - (3) 主程序**MAIN功能：** 调用SUB10 \rightarrow 传递结果地址 \rightarrow 调用SUB11。
 - (4) 为了简化程序，只进行8个字节相加，不考虑最高位的进位。

	99	22	33	44	99	66	77	88
NUMBER1	+0	+1	+2	+3	+4	+5	+6	+7
	99	88	77	66	77	44	44	22
NUMBER2	+0	+1	+2	+3	+4	+5	+6	+7
NUMBER3	+0	+1	+2	+3	+4	+5	+6	+7

例7.4.8 --1

DATA SEGMENT

NUMBER1 DB 99, 22, 33, 44, 99, 66, 77, 88 ; 设被加数, 8字节

NUMBER2 DB 99, 88, 77, 66, 77, 44, 44, 22 ; 设加数

NUMBER3 DB 8 DUP(0) ; 和参数表

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE, DS:DATA

MAIN PROC FAR ; 主程序定义为FAR属性

MOV AX, DATA
MOV DS, AX }

CALL SUB10 ; 调SUB10多字节加子程序

LEA BX, NUMBER3+7 ; 形成和末地址, 高位地址

CALL SUB11 ; 调SUB11二→十六子程序

MOV AH, 4CH ; 返回DOS

INT 21H }

例7.4.8 --2

SUB10	PROC	NEAR	; 多字节加子程序, 由变量直接传递参数
	PUSH	AX	}
	PUSH	CX	
	PUSH	BX	
	PUSH	SI	
	PUSH	DI	
	LEA	SI, NUMBER1	; 取被加数偏移地址
	LEA	DI, NUMBER2	; 取加数偏移地址
	LEA	BX, NUMBER3	; 取和数偏移地址
	MOV	CX, 8	; 计数初值, 8个字节
	AND	AL, AL	; 0→CF
目的 AA1:	MOV	AL, [SI]	
	ADC	AL, [DI]	; 求单字节和
	MOV	[BX], AL	; 存单字节和
	INC	SI	; 修改下次被加数偏移地址
	INC	DI	; 修改下次加数偏移地址
	INC	BX	; 修改下次和数偏移地址
	LOOP	AA1	; 判循环是否结束
	POP	DI	}
	POP	SI	
	POP	BX	
	POP	CX	
	POP	AX	
	RET		; 返主, 结果在NUMBER3中
SUB10	ENDP		

例7.4.8 --3

SUB11 PROC NEAR

; 二化十六子程序, 传递参数地址

PUSH DX

PUSH CX

PUSH AX

MOV DH, 8

; 计数初值, 8个字节

AA4: MOV DL, [BX]

; 从高位开始, 逐字节取出2进制数

显示高半
字节

MOV CL, 4

SHR DL, CL

; 逻辑右移, 高4位移到低4位后, 高位补0

; 提取一个字节的 高半字节

OR DL, 30H

; 形成0~9的ASCII码, 30H~39H

CMP DL, 3AH

; 判是否为0~9

JC AA2

; 是, 转移

ADD DL, 7

; 形成A~F的ASCII码, 41H~46H

显示1位
16进制数

AA2: MOV AH, 2

; 显示

INT 21H

例7.4.8 --4

显示低半
字节

显示1位
16进制数
AA3:

```
MOV DL, [BX]
AND DL, 0FH
OR DL, 30H
CMP DL, 3AH
JC AA3
ADD DL, 7
MOV AH, 2
INT 21H
```

; 从高位开始, 逐位取出2进制数, 再次取数
; 截取字节低4位, 析取一个字节的低半字节
; 形成0~9的ASCII码, 30H~39H
; 判是否为0~9
; 是, 转移
; 形成A~F的ASCII码, 41H~46H
; 显示

```
DEC BX
DEC DH
JNZ AA4
```

; 修改下次偏移地址
; 计循环次数, 字节数减1

```
MOV DL, ' H'
MOV AH, 2
INT 21H
```

} 显示 H

```
POP AX
POP CX
POP DX
```

```
RET
```

; 返主

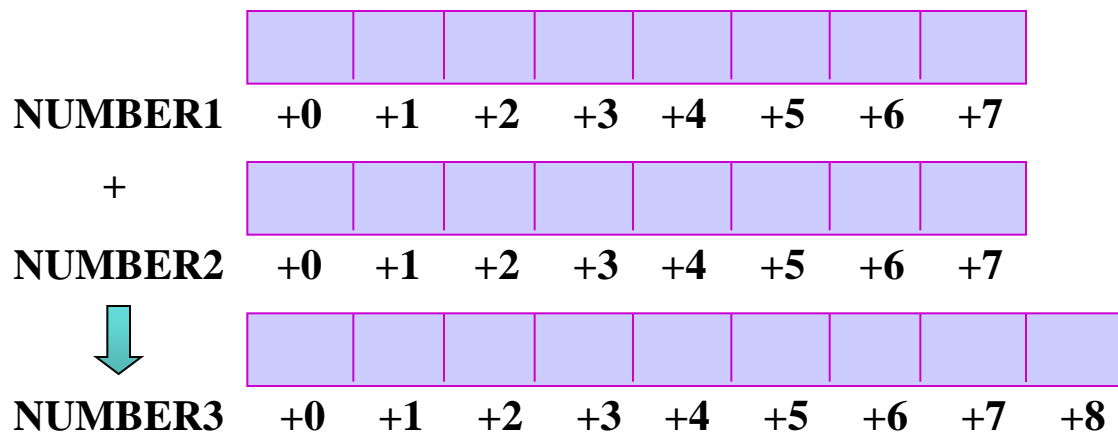
```
SUB11
MAIN
CODE
```

```
ENDP
ENDP
ENDS
END
```

```
MAIN
```

3. 通过堆栈传送参数或参数表地址

- **特点：**依靠堆栈操作交换参数。需要正确设置返回地址。
- **例7.4.9：**用堆栈传送参数和参数表地址方式，编制程序，键入**8位**的**非压缩BCD码**，**加法**，并**显示**。
- **设计说明：**
 - (1) **SUB12功能：**接收键盘输入的数字→**数字**及**数字位数**压入堆栈。键入非数字键返回。参数传递采用堆栈传送方式，约束条件：数字位数 ≤ 8
 - (2) **SUB13功能：**从堆栈取出加法运算结果的地址→在**CRT**上显示。参数传递采用堆栈传送方式。
 - (3) 主程序**MAIN功能：**调用SUB12 → 产生加数和被加数→加法运算（如下图），**非压缩BCD码调整**→调用SUB13 → 清0缓冲区。



例7.4.9 --1

DATA SEGMENT

NUMBER1 DB 8 DUP(0) ; 被加数参数表

NUMBER2 DB 8 DUP(0) ; 加数参数表

NUMBER3 DB 9 DUP(0) ; 和参数表

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE, DS:DATA

MAIN PROC FAR ; 主程序定义为FAR属性

MOV AX, DATA

MOV DS, AX

例7.4.9 --2

处理被加数

AA3:

CALL	SUB12	; 调SUB12非压缩BCD码接收子程序
POP	CX	; 弹出数据位数
LEA	BX, NUMBER1	; 取被加数参数表地址
POP	AX	; 弹出数据, 依次是个位、十位、...
MOV	[BX], AL	; 被加数存入参数表
INC	BX	; 形成下位地址
LOOP	AA3	; 计数控制

处理加数

AA4:

CALL	SUB12	; 调SUB12非压缩BCD码接收子程序
POP	CX	; 取加数位数
LEA	BX, NUMBER2	; 取加数参数表地址
POP	AX	; 取加数个位、十位、百位...
MOV	[BX], AL	; 加数存入参数表
INC	BX	; 形成下位地址
LOOP	AA4	; 计数控制

例7.4.9 --3

	LEA	SI, NUMBER1	; 取被加数参数表地址
	LEA	DI, NUMBER2	; 取加数参数表地址
	LEA	BX, NUMBER3	; 取和参数表地址
	MOV	CX, 8	; 8个字节加法, 8位非压缩BCD码, 计数
	OR	CX, CX	; 0→CF
AA5:	MOV	AL, [SI]	; 取被加数
8字节BCD 加法	ADC	AL, [DI]	; 加, NUMBER1+ NUMBER2
	AAA		; 非压缩BCD码加法调整, 核心指令
	MOV	[BX], AL	; 存和, 送到NUMBER3
	INC	SI	; 形成下位地址
	INC	DI	; 形成下位地址
	INC	BX	; 形成下位地址
	LOOP	AA5	; 计数控制
	ADC	CL, CL	; 最高位送CL
	MOV	[BX], CL	; 存最高位, 送到NUMBER3+8

例7.4.9 --4

LEA AX, NUMBER3+8 ; 取和参数表最高位地址
PUSH AX ; 向子程序提供结果最高位地址
CALL SUB13 ; 非压缩BCD码显示子程序

清0缓冲区

MOV CX, 16 ; 16个字节

LEA BX, NUMBER1 ; 取被加数参数表地址

XOR AL, AL

QQQ2: MOV [BX], AL ; 清除NUMBER1和NUMBER2中已有数据

INC BX ;

LOOP QQQ2 ;

MOV AH, 4CH ; 返回DOS

INT 21H }

例7.4.9 --5

SUB12	PROC	NEAR	;	非压缩BCD码接收子程序
	POP	BX	;	保存返回地址, 段内调用, 只保护IP
	SUB	CX, CX	;	键入数据位数计数器清0
AA1:	MOV	AH, 1	}	;
	INT	21H		
	CMP	AL, 30H		
判断 $0 \leq X \leq 9$	JC	AA2	;	判小于0键, 转AA2, 返回
	CMP	AL, 3AH		
	JNC	AA2	;	判大于9键, 转AA2, 返回
	INC	CX	;	键入数据位数计数器加1
	PUSH	AX	;	键入的数据压栈
	JMP	AA1		
AA2:	PUSH	CX	;	键入的数据位数压栈, 后入栈, 先弹出
	PUSH	BX	;	返回地址压栈, 保持在栈顶位置
	RET		;	返主
SUB12	ENDP			

实际上, 该子程序只是将键入的数据入栈, 并未反映出是非压缩BCD码。

例7.4.9 --6

SUB13	PROC	NEAR	;	非压缩BCD码显示子程序
	POP	AX	;	返回地址IP暂时出栈
	POP	BX	;	取加法结果最高位地址
	PUSH	AX	;	返回地址IP压栈, 恢复
	MOV	CX, 9	;	显示9位数据, 计数器初值
AA6:	MOV	DL, [BX]	;	取加法结果最高位、次高位...个位
	ADD	DL, 30H	;	形成ASCII码
	MOV	AH, 2	}	;
	INT	21H		
	DEC	BX	;	形成下一个显示位的地址
	LOOP	AA6	;	计数控制
	RET		;	返主
SUB13	ENDP			
MAIN	ENDP			
CODE	ENDS			
	END	MAIN		

第7章 结 束