

## MCS-51 单片机实用子程序库

;片内 RAM 初始化子程序

```
IBCLR:  MOV    A,R0
        MOV    R1,A
        CLR    A
IBC1:   MOV    @R1,A
        INC    R1
        DJNZ   R7,IBC1
        RET
```

;片外 RAM 初始化子程序

```
EBCLR1: MOV    A,ADDPL
        MOV    DPL,A
        MOV    A,ADDPH
        MOV    DPH,A
        CLR    C
EBC11:  MOVX    @DPTR,A
        INC    DPTR
        DJNZ   R7,EBC11
        RET
```

;片外 RAM 初始化子程序(双字节个单元)

```
EBCLR2: MOV    A,ADDPL
        MOV    DPL,A
        MOV    A,ADDPH
        MOV    DPH,A
        MOV    A,R7
        JZ     EBC21
        INC    R6
EBC21:  CLR    A
        MOVX    @DPTR,A
        INC    DPTR
        DJNZ   R7,EBC21
        DJNZ   R6,EBC21
        RET
```

;内部 RAM 数据复制程序

;入口: R0,R7

;占用资源: A

;堆栈需求: 2 字节

;出口: R1

```
IBMOV:  MOV    A,R0
        ADD    A,R7
        MOV    R0,A
        MOV    A,R1
        ADD    A,R7
        MOV    R1,A
IBM1:   DEC    R0
        DEC    R1
        MOV    A,@R0
        MOV    @R1,A
        DJNZ   R7,IBM1
        RET
```

;外部 RAM 数据复制程序

;入口: ADDPH,ADDPL,R7

;占用资源: ACC

;堆栈需求: 2 字节

;出口: R0,R1

```
EBMOV1  :  MOV    A,ADDPL
        ADD    A,R7
        MOV    DPL,A
        CLR    A
        ADDC   A,ADDPH
        MOV    DPH,A
        MOV    A,R7
        ADD    A,R1
        XCH    A,R0
        ADDC   A,#00H
        MOV    P2,A
EBM11:   DEC    R0
        CJNE   R0,#0FFH,EBM12
        DEC    P2
EBM12:   DEC    DPL
        MOV    A,DPL
        CJNE   A,#0FFH,EBM13
        DEC    DPH
EBM13:   MOVX   A,@R0
        MOVX   @DPTR,A
        DJNZ   R7,EBM11
        RET
```

;外部 RAM 数据复制程序

;入口: ADDPH,ADDPL,R6,R7

;占用资源: ACC

;堆栈需求: 2 字节

;出口: R0,R1

```
EBMOV2 : MOV A,ADDPL
        ADD A,R7
        MOV DPL,A
        MOV A,R6
        ADDC A,ADDPH
        MOV DPH,A
        MOV A,R7
        ADD A,R1
        XCH A,R0
        ADDC A,R6
        MOV P2,A
        MOV A,R7
        JZ EBM21
        INC R6
EBM21: DEC R0
        CJNE R0,#0FFH,EBM22
        DEC P2
EBM22: DEC DPL
        MOV A,DPL
        CJNE A,#0FFH,EBM23
        DEC DPH
EBM23: MOVX A,@R0
        MOVX @DPTR,A
        DJNZ R7,EBM21
        DJNZ R6,EBM21
        RET
```

;外部 RAM 数据复制到内部 RAM 程序

;入口: ADDPH,ADDPL,R7

;占用资源: ACC

;堆栈需求: 2 字节

;出口: R0

```
ITEMOV : MOV A,ADDPL
        ADD A,R7
        MOV DPL,A
        MOV A,ADDPH
        ADDC A,#00H
        MOV DPH,A
        MOV A,R0
        ADD A,R7
        MOV R0,A
ITEM1 : DEC R0
        DEC DPL
        MOV A,DPL
        CJNE A,#0FFH,ITEM2
```

```

        DEC    DPH
ITEM2   :   MOVX  A,@DPTR
        MOV    @R0,A
        DJNZ   R7,ITEM1
        RET

```

```

;限幅滤波程序
;入口:  A,SDAT,DELTY
;占用资源:  B
;堆栈需求:  2 字节
;出口:  A

```

```

JUGFILT :   MOV    B,A
        CLR     C
        SUBB    A,SDAT
        JNC     JUGFT1
        CPL     A
        INC     A
JUGFT1  :   SETB   A
        SUBB    A,#DELTY
        JNC     JUGFT3
        MOV     A,SDAT
        RET
JUGFT3  :   MOV    A,B
        MOV     SDAT,A
        RET

```

```

;中位值滤波程序
;入口:  ADDPH,ADDPL,N
;占用资源:  ESELSORT
;堆栈需求:  4 字节
;出口:  A

```

```

MEDFILT :   LCALL  ESELSORT
        MOV     A,N
        CLR     C
        RRC     A
        ADD     A,ADDPL
        MOV     DPL,A
        MOV     A,ADDPH
        MOV     DPH,A
        JNC     MEDFT1
        INC     DPH
MEDFT1  :   MOVX   A,@DPTR
        RET

```

;N 点算术平均滤波

;入口: ADDPH,ADDPL,N

;占用资源: B,R3,R4

;堆栈需求: 2 字节

;出口: A

```
AVFILT : MOV A,ADDPL
        MOV DPL,A
        MOV A,ADDPH
        MOV DPH,A
        CLR A
        MOV R3,A
        MOV R4,A
        MOV R7,N
AVFT1:  MOVX A,@DPTR
        INC DPTR
        ADD A,R4
        MOV R4,A
        JNC AVFT2
        INC R3
AVFT2:  DJNZ R7,AVFT1
        MOV R7,N
        MOV R2,#00H
        LCALL NDIV31
        MOV A,R4
        RET
```

;N 点加权平均滤波

;入口: ADDPH,ADDPL,N

;占用资源: B,R3,R4

;堆栈需求: 2 字节

;出口: A

```
QAVFILT : CLR A
        MOV R3,A
        MOV R4,A
        MOV R7,N
        MOV P2,ADDPH
        MOV R1,ADDPL
        MOV DPTR,#QAVTAB
QAVFT1 : MOVC A,@A+DPTR
        MOV B,A
        MOVX A,@R1
        INC DPTR
        INC R1
        MUL AB
        ADD A,R4
        MOV R4,A
```

```

        MOV    A,B
        ADDC   A,R3
        MOV    R3,A
        DJNZ   R7,QAVFT1
        MOV    A,R4
        JNB    ACC.7,QAVFT2
        INC    R3
QAVFT2  :  MOV    A,R3
        RET
QAVTAB  :  DB

```

;一阶加权滞后滤波程序

;入口: A,DELTY

;占用资源: B,R3,R4

;堆栈需求: 2 字节

;出口: A

```

BQFILT  :  MOV    B,A
        CLR    A
        MOV    DPTR,#ABTAB
        MOVC   A,@A+DPTR
        MUL    AB
        MOV    R4,A
        MOV    R3,B
        MOV    A,#01H
        MOVC   A,@A+DPTR
        MOV    B,DELTY
        MUL    AB
        ADD    A,R4
        MOV    R4,A
        MOV    A,B
        ADDC   A,R3
        MOV    R3,A
        MOV    A,R4
        JNB    ACC.7,FT1
        INC    R3
FT1  :  MOV    A,R3
        MOV    DELTY,A
        RET
BQTAB:  DB      80H,80H

```

;双字节取补程序 /(R3R4)=(R3R4)

;入口: R3,R4

;占用资源: ACC

;堆栈需求: 2 字节

;出口: R3,R4

```

CMPT:  MOV    A,R4
        CPL    A
        ADD    A,#01H
        MOV    R4,A
        MOV    A,R3
        CPL    A
        ADDC   A,#00H
        MOV    R3,A
        RET

```

;N 节取补程序  $\neg([R0])=([R0])$

;入口: R0,R7

;占用资源: ACC,B

;堆栈需求: 2 字节

;出口: R0

```

NCMPTN :  MOV    B,R0
          SETB   C
NCPT1:  MOV    A,@R0
          CPL    A
          ADDC   A,#00H
          MOV    @R0,A
          INC    R0
          DJNZ   R7,NCPT1
          MOV    R0,B
          RET

```

;双字节无符号数加法程序  $(R3R4+R6R7)=(R3R4)$

;入口: R3,R4,R6,R7

;占用资源: ACC

;堆栈需求: 2 字节

;出口: R3,R4,CF

```

NADD:  MOV    A,R4
        ADD    A,R7
        MOV    R4,A
        MOV    A,R3
        ADDC   A,R6
        MOV    R3,A
        RET

```

;N 字节无符号数加法程序  $([R0]+[R1])=([R0])$

;入口: R0,R1,R7

;占用资源: ACC,B

;堆栈需求: 2 字节

;出口: R0,CF

```

NADDN:  MOV    B,R0
        CLR    C
NADN1:  MOV    A,@R0
        ADDC   A,@R1
        MOV    @R0,A
        INC    R0
        INC    R1
        DJNZ   R7,NADN1
        MOV    R0,B
        RET

```

;双字节无符号数减法程序  $(R3R4-R6R7)=(R3R4)$

;入口: R3,R4,R6,R7

;占用资源: ACC

;堆栈需求: 2 字节

;出口: R3,R4

```

NSUB:   MOV    A,R4
        CLR    C
        SUBB   A,R7
        MOV    R4,A
        MOV    A,R3
        SUBB   A,R6
        MOV    R3,A
        RET

```

;N 字节无符号数减法程序  $([R0]-[R1])=([R0])$

;入口: R0,R1,R7

;占用资源: ACC,B

;堆栈需求: 2 字节

;出口: R0,CF

```

NSUBN:  MOV    B,R0
        MOV    R7,N
        CLR    C
NSUBN1  :  MOV    A,@R0
        SUBB   A,@R1
        MOV    @R0,A
        INC    R0
        INC    R1
        DJNZ   R7,NSUBN1
        MOV    R0,B
        RET

```

;单字节无符号数乘法程序  $(R3R4*R7)=(R2R3R4)$

;入口: R3,R4,R7

;占用资源: ACC,B



;堆栈需求: 2 字节

;出口: R2,R3,R4

```
NMUL21 : MOV    A,R4
        MOV    B,R7
        MUL    AB
        MOV    R4,A
        MOV    A,B
        XCH    A,R3
        MOV    B,R7
        MUL    AB
        ADD    A,R3
        MOV    R3,A
        CLR    A
        ADDC   A,B
        MOV    R2,A
        CLR    OV
        RET
```

;单字节无符号数乘法程序  $(R2R3R4 \times R7) = (R5R2R3R4)$

;入口: R2,R3,R4,R6,R7

;占用资源: ACC,B

;堆栈需求: 2 字节

;出口: R5,R2,R3,R4

```
NMUL31 : MOV    A,R4
        MOV    B,R7
        MUL    AB
        MOV    R4,A
        MOV    A,B
        XCH    A,R3
        MOV    B,R7
        MUL    AB
        ADD    A,R3
        MOV    R3,A
        CLR    A
        ADDC   A,B
        XCH    A,R2
        MOV    B,R7
        MUL    AB
        ADD    A,R2
        MOV    R2,A
        CLR    A
        ADDC   A,B
        MOV    R5,A
        CLR    OV
        RET
```

;单字节无符号数乘法程序  $(R5R2R3R4 \times R7) = (R7R5R2R3R4)$

;入口: R5,R2,R3,R4,R7

;占用资源: ACC,B

;堆栈需求: 2 字节

;出口: R7,R5,R2,R3,R4

```
NMUL41 : MOV    A,R4
        MOV    B,R7
        MUL    AB
        MOV    R4,A
        MOV    A,B
        XCH    A,R3
        MOV    B,R7
        MUL    AB
        ADD    A,R3
        MOV    R3,A
        CLR    A
        ADDC   A,B
        XCH    A,R2
        MOV    B,R7
        MUL    AB
        ADD    A,R2
        MOV    R2,A
        CLR    A
        ADDC   A,B
        XCH    A,R5
        MOV    B,R7
        MUL    AB
        ADD    A,R5
        MOV    R5,A
        CLR    A
        ADDC   A,B
        MOV    R7,A
        CLR    OV
        RET
```

;双字节无符号数乘法程序  $(R3R4 \times R6R7) = (R5R2R3R4)$

;入口: R3,R4,R6,R7

;占用资源: ACC,B

;堆栈需求: 2 字节

;出口: R5,R2,R3,R4

```
NMUL22 : MOV    A,R4
        MOV    B,R7
        MUL    AB
        XCH    A,R4
        MOV    R5,B
        MOV    B,R6
```

```

MUL    AB
ADD    A,R5
MOV    R5,A
CLR    A
ADDC   A,B
MOV    R2,A
MOV    A,R3
MOV    B,R7
MUL    AB
ADD    A,R5
MOV    R5,A
MOV    A,B
ADDC   A,R2
MOV    R2,A
CLR    A
ADDC   A,#00H
XCH    A,R3
MOV    B,R6
MUL    AB
ADD    A,R2
MOV    R2,A
MOV    A,B
ADDC   A,R3
XCH    A,R5
MOV    R3,A
CLR    OV
RET

```

;双字节无符号数乘法程序  $(R2R3R4 \times R6R7) = (R1R5R2R3R4)$

;入口: R2,R3,R4,R6,R7

;占用资源: ACC,B

;堆栈需求: 2 字节

;出口: R1,R5,R2,R3,R4

```

NMUL32 : MOV    A,R4
        MOV    B,R7
        MUL    AB
        XCH    A,R4
        MOV    R5,B
        MOV    B,R6
        MUL    AB
        ADD    A,R5
        MOV    R5,A
        CLR    A
        ADDC   A,B
        MOV    R1,A
        MOV    A,R3
        MOV    B,R7

```

```

MUL    AB
ADD    A,R5
MOV    R5,A
MOV    A,B
ADDC   A,R1
MOV    R1,A
CLR    A
ADDC   A,#00H
XCH    A,R3
MOV    B,R6
MUL    AB
ADD    A,R1
MOV    R1,A
MOV    A,B
ADDC   A,R3
XCH    A,R5
MOV    R3,A
MOV    A,R2
MOV    B,R7
MUL    AB
ADD    A,R1
MOV    R1,A
MOV    A,B
ADDC   A,R5
MOV    R5,A
CLR    A
ADDC   A,#00H
XCH    A,R2
MOV    B,R6
MUL    AB
ADD    A,R5
MOV    R5,A
MOV    A,B
ADDC   A,R2
XCH    A,R1
MOV    R2,A
CLR    OV
RET

```

;N 字节无符号数乘法程序  $([R0] \times [R1]) = ([R0])$

;入口: R0,R1,M,N

;占用资源: ACC,B,R2,R5,R6,R7,NCNT

;堆栈需求: 2 字节

;出口: R0

```

NMULMN : MOV    A,M
        ADD    A,R0
        MOV    R5,A

```

```

XCH    A,R1
XCH    A,R5
ADD    A,N
XCH    A,R0
MOV    R6,A
MOV    B,M
MOV    NCNT,B
NMLMN1 : DEC    R0
        DEC    R1
        CLR    A
        XCH    A,@R1
        MOV    @R0,A
        DJNZ   NCNT,NMLMN1
        MOV    NCNT,B
NMLMN2 : CLR    A
        XCH    A,@R0
        MOV    R2,A
        MOV    A,R6
        MOV    R0,A
        MOV    A,R5
        MOV    R1,A
        MOV    R7,N
        CLR    C
NMLMN3 : MOV    A,R2
        MOV    B,@R1
        INC    R1
        MUL    AB
        ADDC   A,@R0
        MOV    @R0,A
        INC    R0
        MOV    A,B
        ADDC   A,@R0
        MOV    @R0,A
        DJNZ   R7,NMLMN3
        INC    R0
        INC    R6
        DJNZ   NCNT,NMLMN2
        MOV    A,R0
        CLR    C
        SUBB   A,M
        SUBB   A,N
        MOV    R0,A
        RET

```

;单字节无符号除法程序 (R2R3R4/R7)=(R2)R3R4 余数 R7

;入口: R2,R3,R4,R7

;占用资源: ACC,B,F0

;堆栈需求: 3 字节

;出口: (R2),R3,R4,R7,OV

```
NDIV31 : MOV A,R2
        MOV B,R7
        DIV AB
        PUSH A
        MOV R2,B
        MOV B,#10H
NDV311 : CLR C
        MOV A,R4
        RLC A
        MOV R4,A
        MOV A,R3
        RLC A
        MOV R3,A
        MOV A,R2
        RLC A
        MOV R2,A
        MOV F0,C
        CLR C
        SUBB A,R7
        JB F0,NDV312
        JC NDV313
NDV312 : MOV R2,A
        INC R4
NDV313 : DJNZ B,NDV311
        POP A
        CLR OV
        JZ NDV314
        SETB OV
NDV314 : XCH A,R2
        MOV R7,A
        RET
```

;单字节无符号除法程序  $(R5R2R3R4/R7)=(R5)R2R3R4$  余数 R7

;入口: R2,R3,R4,R7

;占用资源: ACC,B,F0

;堆栈需求: 3 字节

;出口: (R5),R2,R3,R4,R7,OV

```
NDIV41 : MOV A,R5
        MOV B,R7
        DIV AB
        PUSH A
        MOV R5,B
        MOV B,#18H
NDV411 : CLR C
        MOV A,R4
```

```

RLC    A
MOV    R4,A
MOV    A,R3
RLC    A
MOV    R3,A
MOV    A,R2
RLC    A
MOV    R2,A
MOV    A,R5
RLC    A
MOV    R5,A
MOV    F0,C
CLR    C
SUBB   A,R7
JB     F0,NDV412
JC     NDV413
NDV412 : MOV    R5,A
        INC    R4
NDV413 : DJNZ   B,NDV411
        POP    A
        CLR    OV
        JZ     NDV414
        SETB   OV
NDV414 : XCH    A,R5
        MOV    R7,A
        RET

```

;双字节无符号除法程序  $(R5R2R3R4/R6R7)=(R2)R3R4$  余数  $R6R7$

;入口:  $R5,R2,R3,R4,R6,R7$

;占用资源:  $ACC,B,F0$

;堆栈需求: 4 字节

;出口:  $(R2),R3,R4,R6,R7,OV$

```

NDIV42 : MOV    A,R1
        PUSH   A
        MOV    B,#00H
NDV421 : MOV    A,R2
        CLR    C
        SUBB   A,R7
        MOV    R1,A
        MOV    A,R5
        SUBB   A,R6
        JC     NDV422
        MOV    R5,A
        MOV    A,R1
        MOV    R2,A
        INC    B
        SJMP   NDV421

```

```

NDV422  :  PUSH  B
          MOV   B,#10H
NDV423  :  CLR   C
          MOV   A,R4
          RLC   A
          MOV   R4,A
          MOV   A,R3
          RLC   A
          MOV   R3,A
          MOV   A,R2
          RLC   A
          MOV   R2,A
          XCH   A,R5
          RLC   A
          XCH   A,R5
          MOV   F0,C
          CLR   C
          SUBB  A,R7
          MOV   R1,A
          MOV   A,R5
          SUBB  A,R6
          JB    F0,NCV424
          JC    NDV425
NCV424  :  MOV   R5,A
          MOV   A,R1
          MOV   R2,A
          INC   R4
NDV425  :  DJNZ  B,NDV423
          POP   A
          CLR   OV
          JNZ   NDV426
          SETB  OV
NDV426  :  XCH   A,R2
          MOV   R7,A
          MOV   A,R5
          MOV   R6,A
          POP   A
          MOV   R1,A
          RET

```

;N 字节无符号除法程序(组合) ([R0]/[R1])=([R0])

;入口: R0,R1,M,N

;占用资源: ACC,R2,R3,R4,R5,R7,NCNT,F0,NADDN,NSUBBN,NRLCN

;堆栈需求: 4 字节

;出口: R0

```

;NDIVMN  :  MOV   A,M
          CLR   C

```



```

        SUBB    A,N
        MOV     NCNT,A
        ADD     A,R0
        MOV     R4,A
        XCH     A,R0
        MOV     R3,A
        MOV     A,R1
        MOV     R5,A
        MOV     R2,#00H
NDVMN1  :  MOV     R7,N
        LCALL   NSUBN
        MOV     A,R5
        MOV     R1,A
        JC      NDVMN2
        INC     R2
        SJMP    NDVMN1
NDVMN2  :  MOV     R7,N
        LCALL   NADDN
        MOV     A,NCNT
        SWAP    A
        RR      A
        MOV     NCNT,A
NDVMN3  :  MOV     A,R3
        MOV     R0,A
        MOV     R7,M
        LCALL   NRLCN
        MOV     F0,C
        MOV     A,R4
        MOV     R0,A
        MOV     A,R5
        MOV     R1,A
        MOV     R7,N
        LCALL   NSUBN
        JB      F0,NDVMN4
        JC      NDVMN5
NDVMN4  :  MOV     A,R3
        MOV     R0,A
        INC     @R0
        SJMP    NDVMN6
NDVMN5  :  MOV     A,R5
        MOV     R1,A
        MOV     R7,N
        LCALL   NADDN
NDVMN6  :  DJNZ    NCNT,NDVMN3
        MOV     A,R4
        MOV     R1,A
        MOV     A,R2
        MOV     @R1,A

```

```

MOV    A,R3
MOV    R0,A
RET

```

;N 字节无符号除法程序(集成)  $([R0]/R[1])=([R0])$

;入口: R0,R1,M,N

;占用资源: ACC,R2,R3,R4,R5,R7,F0

;堆栈需求: 2 字节

;出口: R0

```

NDIVMN  :  MOV    A,M
          CLR     C
          SUBB    A,N
          MOV     B,A
          ADD     A,R0
          MOV     R4,A
          XCH     A,R0
          MOV     R3,A
          MOV     A,R1
          MOV     R5,A
          MOV     R2,#00H
NDVMN1   :  MOV     R7,N
          CLR     C
NDVMN2   :  MOV     A,@R0
          SUBB    A,@R1
          MOV     @R0,A
          INC     R0
          INC     R1
          DJNZ    R7,NDVMN2
          MOV     A,R4
          MOV     R0,A
          MOV     A,R5
          MOV     R1,A
          JC      NDVMN3
          INC     R2
          SJMP    NDVMN1
NDVMN3   :  MOV     R7,N
          CLR     C
NDVMN4   :  MOV     A,@R0
          ADDC    A,@R1
          MOV     @R0,A
          INC     R0
          INC     R1
          DJNZ    R7,NDVMN4
          MOV     A,#08H
          MUL     AB
          MOV     B,A
NDVMN5   :  MOV     A,R3

```

```

        MOV    R0,A
        MOV    R7,M
        CLR    C
NDVMN6  :  MOV    A,@R0
        RLC    A
        MOV    @R0,A
        INC    R0
        DJNZ   R7,NDVMN6
        MOV    F0,C
        MOV    A,R4
        MOV    R0,A
        MOV    A,R5
        MOV    R1,A
        MOV    R7,N
        CLR    C
NDVMN7  :  MOV    A,@R0
        SUBB   A,@R1
        MOV    @R0,A
        INC    R0
        INC    R1
        DJNZ   R7,NDVMN7
        JB     F0,NDVMNB
        JC     NDVMN8
NDVMNB  :  MOV    A,R3
        MOV    R0,A
        INC    @R0
        SJMP   NDVMNA
NDVMN8  :  MOV    R7,N
        MOV    A,R4
        MOV    R0,A
        MOV    A,R5
        MOV    R1,A
        CLR    C
NDVMN9  :  MOV    A,@R0
        ADDC   A,@R1
        MOV    @R0,A
        INC    R0
        INC    R1
        DJNZ   R7,NDVMN9
NDVMNA  :  DJNZ   B,NDVMN5
        MOV    A,M
        CLR    C
        SUBB   A,N
        ADD    A,R3
        MOV    R1,A
        MOV    A,R2
        MOV    @R1,A
        MOV    A,R3

```

```
MOV    R0,A
RET
```

;N 字节数据左移程序 RLC([R0])=(CF[R0])

;入口: R0,R7

;占用资源: ACC,B

;堆栈需求: 2 字节

;出口: R0,CF

```
NRLCN:  MOV    B,R0
        CLR    C
NRLN1:  MOV    A,@R0
        RLC    A
        MOV    @R0,A
        INC    R0
        DJNZ   R7,NRLN1
        MOV    R0,B
        RET
```

;原码有符号双字节减法程序 (R3R4-R6R7)=R3R4

;入口: R3,R4,R6,R7

;占用资源: ACC,DADD

;堆栈需求: 6 字节

;出口: R3,R4,OV

```
DSUB:   MOV    A,R6
        CPL    ACC.7
        MOV    R6,A
        LCALL  DADD
        RET
```

;原码有符号双字节加法程序 (R3R4+R6R7)=R3R4

;入口: R3,R4,R6,R7

;占用资源: ACC,SR0,NADD,NSUB,CMPT

;堆栈需求: 4 字节

;出口: R3,R4,OV

```
DADD:   MOV    A,R3
        MOV    C,ACC.7
        MOV    SR0,C
        XRL    A,R6
        MOV    C,ACC.7
        MOV    A,R3
        CLR    ACC.7
        MOV    R3,A
        MOV    A,R6
        CLR    ACC.7
        MOV    R6,A
```

```

        JC      DAB2
        LCALL  NADD
        MOV    A,R3
        JB     ACC.7,DABE
DAB1:   MOV     C,SR0
        MOV    ACC.7,C
        MOV    R3,A
        CLR    OV
        RET
DABE:   SETB    OV
        RET
DAB2:   LCALL  NSUB
        MOV    A,R3
        JNB    ACC.7,DAB1
        LCALL  CMPT
        CPL    SR0
        SJMP   DAB1

```

;原码有符号双字节乘法程序  $(R3R4 * R6R7) = (R5R2R3R4)$

;入口: R3,R4,R6,R7

;占用资源: ACC,SR0,NMUL22

;堆栈需求: 4 字节

;出口: R5,R2,R3,R4

```

IMUL:   MOV     A,R3
        XRL     A,R6
        MOV     C,ACC.7
        MOV     SR0,C
        MOV     A,R3
        CLR     ACC.7
        MOV     R3,A
        MOV     A,R6
        CLR     ACC.7
        MOV     R6,A
        LCALL  NMUL22
        MOV     A,R5
        MOV     C,SR0
        MOV     ACC.7,C
        MOV     R5,A
        RET

```

;原码有符号双字节除法程序  $(R5R2R3R4 / R6R7) = (R3R4)$  余数(R6R7)

;入口: R5,R2,R3,R4

;占用资源: ACC,SR0,NDIV42

;堆栈需求: 6 字节

;出口: R3,R4,R6,R7,OV

```

IDIV:   MOV     A,R5

```

```

        XRL    A,R6
        MOV    C,ACC.7
        MOV    SR0,C
        MOV    A,R5
        CLR    ACC.7
        MOV    R5,A
        MOV    A,R6
        CLR    ACC.7
        MOV    R6,A
        LCALL  NDIV42
        MOV    A,R3
        JB     ACC.7,IDIVE
        JB     OV,IDIVE
        MOV    C,SR0
        MOV    ACC.7,C
        MOV    R3,A
        RET
IDIVE:  SETB   OV
        RET

```

;单字节顺序查找程序

;入口: R0,R1,A,R7

;占用资源: B

;堆栈需求: 2 字节

;出口: R0,R1,A

```

FINDB1  :  MOV    B,A
          MOV    DPL,R1
          MOV    DPH,R0
FINDB11 :  MOVX   A,@DPTR
          CJNE   A,B,FINDB12
          MOV    R1,DPL
          MOV    R0,DPH
          CLR    A
          RET
FINDB12 :  INC    DPTR
          DJNZ   R7,FINDB11
          MOV    A,#0FFH
          RET

```

;单字节顺序查找程序

;入口: R0,R1,A,R6,R7

;占用资源: B

;堆栈需求: 2 字节

;出口: R0,R1,A

```

FINDB2 : MOV B,A
        MOV DPL,R1
        MOV DPH,R0
        MOV A,R7
        JZ FINDB21
        INC R6
FINDB21 : MOVX A,@DPTR
        CJNE A,B,FINDB22
        MOV R1,DPL
        MOV R0,DPH
        CLR A
        RET
FINDB22 : INC DPTR
        DJNZ R7,FINDB21
        DJNZ R6,FINDB21
        MOV A,#0FFH
        RET

```

;双字节字符串顺序查找程序

;入口: R0,R1,R3,R4,R7

;占用资源: ACC,B

;堆栈需求: 2 字节

;出口: R0,R1,A

```

FINDS1 : MOV DPL,R1
        MOV DPH,R0
FINDS11 : MOVX A,@DPTR
        INC DPTR
        CLR C
        SUBB A,R4
        JNZ FINDS12
        MOVX A,@DPTR
        SUBB A,R3
        JNZ FINDS12
        MOV A,DPL
        CLR C
        SUBB A,#01H
        MOV R1,A
        MOV A,DPH
        SUBB A,#00H
        MOV R0,A
        CLR A
        RET
FINDS12 : DJNZ R7,FINDS11
        MOV A,#0FFH
        RET

```

;双字节字符串顺序查找程序

;入口: R0,R1,R3,R4,R6,R7

;占用资源: ACC,B

;堆栈需求: 2 字节

;出口: R0,R1,A

```
FINDS2 : MOV    DPL,R1
        MOV    DPH,R0
        MOV    A,R7
        JZ     FINDS21
        INC    R6
FINDS21 : MOVX   A,@DPTR
        INC    DPTR
        CLR    C
        SUBB   A,R4
        JNZ    FINDS22
        MOVX   A,@DPTR
        SUBB   A,R3
        JNZ    FINDS22
        MOV    A,DPL
        CLR    C
        SUBB   A,#01H
        MOV    R1,A
        MOV    A,DPH
        SUBB   A,#00H
        MOV    R0,A
        CLR    A
        RET
FINDS22 : DJNZ   R7,FINDS21
        DJNZ   R6,FINDS21
        MOV    A,#0FFH
        RET
```

;N 字节字符串顺序查找程序

;入口: ADDPH,ADDPL,R0,R6,R7,N

;占用资源: ACC,B,R2,NCNT

;堆栈需求: 2 字节

;出口: ADDPH,ADDPL,A

```
FINDN:  MOV    A,R0
        MOV    R2,A
        MOV    A,ADDPL
        MOV    DPL,A
        MOV    A,ADDPH
        MOV    DPH,A
        MOV    A,R7
        JZ     FINDN1
        INC    R6
FINDN1 : MOV    A,R2
```



```

        MOV    R0,A
        MOV    A,N
        MOV    NCNT,A
FINDN2  :  MOVX  A,@DPTR
        CLR    C
        SUBB   A,@R0
        JNZ    FINDN3
        INC    DPTR
        INC    R0
        DJNZ   NCNT,FINDN2
        MOV    A,DPL
        CLR    C
        SUBB   A,N
        MOV    ADDPL,A
        MOV    A,DPH
        SUBB   A,#00H
        MOV    ADDPH,A
        CLR    A
        RET
FINDN3  :  CLR    C
        MOV    A,R0
        SUBB   A,R2
        JNZ    FINDN4
        INC    DPTR
FINDN4  :  DJNZ   R7,FINDN1
        DJNZ   R6,FINDN1
        MOV    A,#0FFH
        RET

```

;单字节最值查找程序

;入口: R0,R1,R6,R7

;占用资源: ACC,B

;堆栈需求: 2 字节

;出口: R0(最大值),R1(最小值),R2,R3,R4,R5

```

FMAMIB  :  MOV    DPL,R1
        MOV    DPH,R0
        MOVX   A,@DPTR
        MOV    R0,A
        MOV    R1,A
        MOV    A,DPL
        MOV    R3,A
        MOV    R5,A
        MOV    A,DPH
        MOV    R2,A
        MOV    R4,A
        MOV    A,R7
        JZ     FMMB1

```

```

        INC     R6
FMMB1:  MOVX    A,@DPTR
        MOV     B,A
        SETB    C
        SUBB    A,R0
        JC      FMMB2
        MOV     R0,B
        MOV     R3,DPL
        MOV     R2,DPH
        SJMP    FMMB3
FMMB2:  MOV     A,B
        CLR     C
        SUBB    A,R1
        JNC     FMMB3
        MOV     R1,B
        MOV     R5,DPL
        MOV     R4,DPH
FMMB3:  INC     DPTR
        DJNZ    R7,FMMB1
        DJNZ    R6,FMMB1
        RET

```

;浮点数顺序查找程序

;入口: R0,R1,R2,R3,R4,R6,R7

;占用资源: B,NCNT,FCMP

;堆栈需求: 2 字节

;出口: R0,R1,A

```

FINDF:  MOV     DPL,R1
        MOV     DPH,R0
        MOV     A,R7
        MOV     B,A
        MOV     NCNT,R6
        JZ      FINDF1
        INC     NCNT
FINDF1 :  MOVX    A,@DPTR
        INC     DPTR
        MOV     R5,A
        MOVX    A,@DPTR
        INC     DPTR
        MOV     R6,A
        MOVX    A,@DPTR
        INC     DPTR
        MOV     R7,A
        LCALL   FCMP
        JNZ     FINDF2
        MOV     A,DPL
        CLR     C

```

```

        SUBB    A,#03H
        MOV     R1,A
        MOV     A,DPH
        SUBB    A,#00H
        MOV     R0,A
        CLR     A
        RET
FINDF2  :   DJNZ    B,FINDF1
        DJNZ    NCNT,FINDF1
        MOV     A,#0FFH
        RET

```

;浮点数最值查找程序

;入口: ADDPH,ADDPL,R6,R7

;占用资源: ACC,B,NCNT,ITEMOV,EBMOV,MOVB,MOVR1,FCMP

;堆栈需求: 5 字节

;出口: [R0](最大值),[R1](最小值),R2,R3,R4,R5

```

FMAMIF  :   MOV     A,ADDPL
        MOV     R3,A
        MOV     R5,A
        MOV     DPL,A
        MOV     A,ADDPH
        MOV     R2,A
        MOV     R4,A
        MOV     DPH,A
        MOV     B,R7
        MOV     R7,#03H
        LCALL   ITEMOV
        MOV     R7,#03H
        LCALL   IBMOV
        MOV     A,B
        JZ      FMMF1
        INC     NCNT
FMMF1:   PUSH    B
        MOVX    A,@DPTR
        INC     DPTR
        MOV     R2,A
        MOVX    A,@DPTR
        INC     DPTR
        MOV     R3,A
        MOVX    A,@DPTR
        INC     DPTR
        MOV     R4,A
        LCALL   MOVR1
        LCALL   FCMP
        JNC     FMMF2
        MOV     A,R0

```

```

        XCH    A,R1
        MOV    R0,A
        LCALL  MOVB
        MOV    R5,DPL
        MOV    R4,DPH
        MOV    A,R0
        XCH    A,R1
        MOV    R0,A
        SJMP   FMMF3
FMMF2:  MOV    A,R0
        XCH    A,R1
        MOV    R0,A
        LCALL  MOVR1
        LCALL  FCMP
        MOV    A,R0
        XCH    A,R1
        MOV    R0,A
        JZ     FMMF3
        JC     FMMF3
        LCALL  MOVB
        MOV    R3,DPL
        MOV    R2,DPH
FMMF3:  POP     B
        DJNZ   B,FMMF1
        DJNZ   NCNT,FMMF1
        RET

```

;单字节折半查找程序

;入口: A,R0,R1,R6,R7

;占用资源: B,R2

;堆栈需求: 2 字节

;出口: R0,R1

```

SEARCHB :  MOV    B,A
           MOV    A,R1
           ADD    A,R7
           MOV    R7,A
           MOV    A,R0
           ADDC   A,R6
           MOV    R6,A
           MOV    A,R7
           SUBB   A,#01H
           MOV    R7,A
           JNC    SECH1
           DEC    R6
SECH1:  MOV    A,R7
           CLR    C
           SUBB   A,R1

```

```

        MOV    A,R6
        SUBB   A,R0
        JNC    SECH2
        MOV    A,#0FFH
        RET
SECH2:  MOV    A,R7
        ADD    A,R1
        MOV    R2,A
        MOV    A,R6
        ADDC   A,R0
        RRC    A
        MOV    DPH,A
        MOV    A,R2
        RRC    A
        MOV    DPL,A
        MOVX   A,@DPTR
        CLR    C
        SUBB   A,B
        JNC    SECH3
        INC    DPTR
        MOV    R0,DPH
        MOV    R1,DPL
        SJMP   SECH1
SECH3:  JZ      SECH5
        MOV    A,DPL
        SUBB   A,#01H
        MOV    R7,A
        JNC    SECH4
        MOV    R6,DPH
        DEC    R6
SECH4:  SJMP   SECH1
SECH5:  MOV    R0,DPH
        MOV    R1,DPL
        CLR    A
        RET

```

;辛普生积分程序

;入口: DPTR,N,COUNT

;占用资源: ACC,R3,R4,R6,R7

;堆栈需求: 2 字节

;出口: R3,R4

```

SJF :   MOV    R7,N
        MOVX   A,@DPTR
        INC    DPTR
        MOV    R4,A
        MOV    R3,#00H
        DEC    R7
SJF1:   MOVX   A,@DPTR

```

```

    INC    DPTR
    CLR    C
    RLC    A
    MOV    R6,A
    CLR    A
    RLC    A
    XCH    A,R7
    JNB    ACC.0,SJF2
    XCH    A,R6
    RLC    A
    XCH    A,R6
    XCH    A,R7
    RLC    A
    XCH    A,R7
SJF2:  XCH    A,R7
        XCH    A,R6
        ADD    A,R4
        MOV    R4,A
        MOV    A,R6
        ADDC   A,R3
        MOV    R3,A
        DJNZ   R7,SJF1
SJF3:  MOVX   A,@DPTR
        ADD    A,R4
        MOV    R4,A
        CLR    A
        ADDC   A,R3
        MOV    R3,A
        MOV    R7,#COUNT
        LCALL  NMUL21
        MOV    A,N
        MOV    B,#03H
        MUL    AB
        MOV    R7,A
        LCALL  NDIV31
        RET

NMUL21 :  MOV    A,R4
        MOV    B,R7
        MUL    AB
        MOV    R4,A
        MOV    A,B
        XCH    A,R3
        MOV    B,R7
        MUL    AB
        ADD    A,R3
        MOV    R3,A
        CLR    A

```

```

    ADDC    A,B
    MOV     R2,A
    CLR     OV
    RET

```

```

NDIV31 : MOV     B,#10H
NDV311 : CLR     C
        MOV     A,R4
        RLC     A
        MOV     R4,A
        MOV     A,R3
        RLC     A
        MOV     R3,A
        MOV     A,R2
        RLC     A
        MOV     R2,A
        MOV     F0,C
        CLR     C
        SUBB    A,R7
        JB      F0,NDV312
        JC      NDV313
NDV312 : MOV     R2,A
        INC     R4
NDV313 : DJNZ    B,NDV311
        RET

```

```

;内部 RAM 数据排序程序(升序)
;入口:  R0(起始地址),N(数据个数)
;占用资源:  ACC,B,R5,R6,R7
;堆栈需求:  2 字节
;出口:  R0

```

```

ISELSORT: MOV     R7,N
          DEC     R7
ISST1:   MOV     A,R7
          MOV     R6,A
          MOV     A,R0
          MOV     R1,A
          MOV     R2,A
          MOV     B,@R1
ISST2:   INC     R1
          MOV     A,@R1
          CLR     C
          SUBB    A,B
          JC      ISST3
          MOV     A,R1

```

```

        MOV    R2,A
        MOV    B,@R1
ISST3:  DJNZ   R6,ISST2
        MOV    A,B
        XCH    A,@R1
        MOV    B,R2
        MOV    R1,B
        MOV    @R1,A
        DJNZ   R7,ISST1
        RET

```

;外部 RAM 数据排序程序(升序)

;入口: ADDPH,ADDPL(起始地址),N(数据个数)

;占用资源: ACC,B,R0,R1,R5,R7

;堆栈需求: 2 字节

;出口: ADDPH-ADDPL

```

EELSORT:  MOV    R7,N
          DEC    R7
ESST1:    MOV    A,R7
          MOV    R6,A
          MOV    DPL,ADDPL
          MOV    R1,DPL
          MOV    DPH,ADDPH
          MOV    R0,DPH
          MOVX   A,@DPTR
          MOV    B,A
ESST2:    INC    DPTR
          MOVX   A,@DPTR
          CLR    C
          SUBB   A,B
          JC     ESST3
          MOV    R0,DPL
          MOV    R1,DPH
          MOVX   A,@DPTR
          MOV    B,A
ESST3:    DJNZ   R6,ESST2
          MOVX   A,@DPTR
          XCH    A,B
          MOVX   @DPTR,A
          MOV    DPL,R0
          MOV    DPH,R1
          MOV    A,B
          MOVX   @DPTR,A
          DJNZ   R7,ESST1
          RET

```

;外部 RAM 浮点数排序程序(升序)



;入口: ADDPH,ADDPL(起始地址),N(数据个数)  
;占用资源: ACC,B,R0,R1,R2,R3,R4,R5,R6,R7,NCNT  
;堆栈需求: 5 字节  
;出口: ADDPH,ADDPL

```
FSORT:  MOV    A,N
        MOV    NCNT,A
        DEC    NCNT
FST1:   MOV    B,NCNT
        MOV    DPL,ADDPL
        MOV    R1,DPL
        MOV    DPH,ADDPH
        MOV    R0,DPH
        MOVX   A,@DPTR
        MOV    R2,A
        INC    DPTR
        MOVX   A,@DPTR
        MOV    R3,A
        INC    DPTR
        MOVX   A,@DPTR
        MOV    R4,A
FST2:   INC     DPTR
        MOVX   A,@DPTR
        MOV    R5,A
        INC    DPTR
        MOVX   A,@DPTR
        MOV    R6,A
        INC    DPTR
        MOVX   A,@DPTR
        MOV    R7,A
        PUSH   B
        LCALL  FCMP
        POP    B
        JNC    FST4
        MOV    A,DPL
        CLR    C
        SUBB   A,#02H
        MOV    R1,A
        MOV    R0,DPH
        JNC    FST3
        DEC    R0
FST3:   MOV    A,R5
        MOV    R2,A
        MOV    A,R6
        MOV    R3,A
        MOV    A,R7
        MOV    R4,A
FST4:   DJNZ   B,FST2
```

```

        MOV    A,DPL
        CLR    C
        SUBB   A,#02H
        MOV    DPL,A
        JNC    FST5
        DEC    DPH
FST5:   MOV    A,R2
        MOVX   @DPTR,A
        INC    DPTR
        MOV    A,R3
        MOVX   @DPTR,A
        INC    DPTR
        MOV    A,R4
        MOVX   @DPTR,A
        MOV    A,R0
        MOV    P2,A
        MOV    A,R5
        MOVX   @R1,A
        INC    R1
        MOV    A,R6
        MOVX   @R1,A
        INC    R1
        MOV    A,R7
        MOVX   @R1,A
        DJNZ   NCNT,FST1
        RET

```

;BCD 小数转换为二进制小数(2 位)

;入口: R0(低位首址),R7

;占用资源: ACC,B,R5

;堆栈需求: 2 字节

;出口: R3,R4

```

PDTB:   CLR    A
        MOV    R3,A
        MOV    R4,A
PDB1:   MOV    A,R3
        MOV    B,#9AH
        MUL    AB
        MOV    R5,B
        XCH    A,R4
        MOV    B,#19H
        MUL    AB
        ADD    A,R4
        MOV    A,R5
        ADDC   A,B

```

```

MOV    R5,A
MOV    A,@R0
MOV    B,#9AH
MUL    AB
ADD    A,R5
MOV    R4,A
CLR    A
ADDC   A,B
XCH    A,R3
MOV    B,#19H
MUL    AB
ADD    A,R4
MOV    R4,A
MOV    A,B
ADDC   A,R3
MOV    R3,A
MOV    A,@R0
MOV    B,#19H
MUL    AB
ADD    A,R3
MOV    R3,A
DEC    R0
DJNZ   R7,PDB1
RET

```

;BCD 小数转换为二进制小数(N 位)

;入口: R1(低位首址),M,N

;占用资源: ACC,B,R2,R3,R7

;堆栈需求: 2 字节

;出口: R0

```

PDTBMN : MOV    A,R0
        MOV    R2,A
        MOV    A,R1
        MOV    R3,A
        MOV    B,N
        CLR    A
PDBMN1 : MOV    @R0,A
        INC    R0
        DJNZ   B,PDBMN1
        MOV    A,N
        SWAP   A
        RR     A
        MOV    R7,A
PDBMN2 : MOV    A,R2
        MOV    R0,A
        MOV    A,R3
        MOV    R1,A

```

```

        MOV    B,M
        CLR    C
PDBMN3  :  MOV    A,@R1
        ADDC   A,@R1
        DA     A
        JNB    ACC.4,PDBMN4
        SETB   C
        CLR    ACC.4
PDBMN4  :  MOV    @R1,A
        INC    R1
        DJNZ   B,PDBMN3
        MOV    B,N
PDBMN5  :  MOV    A,@R0
        RLC    A
        MOV    @R0,A
        INC    R0
        DJNZ   B,PDBMN5
        DJNZ   R7,PDBMN2
        MOV    A,R2
        MOV    R0,A
        RET

```

;BCD 整数转换为二进制整数(1 位)

;入口: R0(高位地址),R7

;占用资源: ACC,B

;堆栈需求: 2 字节

;出口: R4

```

IDTB1:  CLR    A
        MOV    R4,A
IDB11:  MOV    A,R4
        MOV    B,#0AH
        MUL    AB
        ADD    A,@R0
        INC    R0
        MOV    R4,A
        DJNZ   R7,IDB11
        RET

```

;BCD 整数转换为二进制整数(2 位)

;入口: R0(高位地址),R7

;占用资源: ACC,B

;堆栈需求: 2 字节

;出口: R3,R4

```

IDTB2:  CLR    A
        MOV    R3,A
        MOV    R4,A

```

```

IDB21:  MOV    A,R4
        MOV    B,#0AH
        MUL    AB
        MOV    R4,A
        MOV    A,B
        XCH    A,R3
        MOV    B,#0AH
        MUL    AB
        ADD    A,R3
        MOV    R3,A
        MOV    A,R4
        ADD    A,@R0
        INC    R0
        MOV    R4,A
        CLR    A
        ADDC   A,R3
        MOV    R3,A
        DJNZ   R7,IDB21
        RET

```

;BCD 整数转换为二进制整数(3 位)

;入口: R0(高位地址),R7

;占用资源: ACC,B

;堆栈需求: 2 字节

;出口: R2,R3,R4

```

IDTB3:  CLR    A
        MOV    R2,A
        MOV    R3,A
        MOV    R4,A
IDB31:  MOV    A,R4
        MOV    B,#0AH
        MUL    AB
        MOV    R4,A
        MOV    A,B
        XCH    A,R3
        MOV    B,#0AH
        MUL    AB
        ADD    A,R3
        MOV    R3,A
        CLR    A
        ADDC   A,B
        XCH    A,R2
        MOV    B,#0AH
        MUL    AB
        ADD    A,R2
        MOV    R2,A
        MOV    A,R4

```

```

ADD    A,@R0
INC    R0
MOV    R4,A
CLR    A
ADDC   A,R3
MOV    R3,A
CLR    A
ADDC   A,R2
MOV    R2,A
DJNZ   R7,IDB31
RET

```

;BCD 整数转换为二进制整数(N 位)

;入口: R1(高位地址),M,N

;占用资源: ACC,B,R2,R7,NCNT,F0

;堆栈需求: 2 字节

;出口: R0

```

IDTBMN : MOV    A,R0
        MOV    R2,A
        MOV    B,N
        CLR    A
IDBMN1 : MOV    @R0,A
        INC    R0
        DJNZ   B,IDBMN1
        MOV    A,R2
        MOV    R0,A
        MOV    A,M
        MOV    NCNT,A
IDBMN2 : MOV    R7,N
        CLR    A
        CLR    F0
IDBMN3 : XCH    A,@R0
        MOV    B,#0AH
        MUL    AB
        MOV    C,F0
        ADDC   A,@R0
        MOV    F0,C
        MOV    @R0,A
        INC    R0
        MOV    A,B
        DJNZ   R7,IDBMN3
        MOV    A,R2
        MOV    R0,A
        MOV    A,@R1
        INC    R1
        ADD    A,@R0
        MOV    @R0,A

```

DJNZ NCNT,IDBMN2  
RET

;二进制小数(2 位)转换为十进制小数(分离 BCD 码)

;入口: R3,R4,R7

;占用资源: ACC,B

;堆栈需求: 3 字节

;出口: R0

```
PBTD:  MOV    A,R7
        PUSH   A
PBD1:  MOV    A,R4
        MOV    B,#0AH
        MUL    AB
        MOV    R4,A
        MOV    A,B
        XCH    A,R3
        MOV    B,#0AH
        MUL    AB
        ADD    A,R3
        MOV    R3,A
        CLR    A
        ADDC   A,B
        MOV    @R0,A
        INC    R0
        DJNZ   R7,PBD1
        POP    A
        MOV    R7,A
        MOV    A,R0
        CLR    C
        SUBB   A,R7
        MOV    R0,A
        RET
```

;二进制小数(M 位)转换为十进制小数(分离 BCD 码)

;入口: R1,M,N

;占用资源: ACC,B,R2,R3,R7,NCNT

;堆栈需求: 2 字节

;出口: R0

```
PBTDMN :  MOV    A,R0
          MOV    R2,A
          MOV    A,R1
          MOV    R3,A
          MOV    A,N
          MOV    NCNT,A
PBDMN1 :  MOV    R7,M
          CLR    A
```

```

        CLR    F0
PBDMN2 :   XCH    A,@R1
        MOV    B,#0AH
        MUL    AB
        MOV    C,F0
        ADDC   A,@R1
        MOV    F0,C
        MOV    @R1,A
        INC    R1
        MOV    A,B
        DJNZ   R7,PBDMN2
        ADDC   A,#00H
        MOV    @R0,A
        INC    R0
        MOV    A,R3
        MOV    R1,A
        DJNZ   NCNT,PBDMN1
        MOV    A,R2
        MOV    R0,A
        RET

```

;二进制整数(2 位)转换为十进制整数(分离 BCD 码)

;入口: R3,R4

;占用资源: ACC,R2,NDIV31

;堆栈需求: 5 字节

;出口: R0,NCNT

```

IBTD21 :   MOV    NCNT,#00H
        MOV    R2,#00H
IBD211 :   MOV    R7,#0AH
        LCALL  NDIV31
        MOV    A,R7
        MOV    @R0,A
        INC    R0
        INC    NCNT
        MOV    A,R3
        ORL    A,R4
        JNZ    IBD211
        MOV    A,R0
        CLR    C
        SUBB   A,NCNT
        MOV    R0,A
        RET

```

;二进制整数(2 位)转换为十进制整数(组合 BCD 码)

;入口: R3,R4

;占用资源: ACC,B,R7

;堆栈需求: 3 字节



;出口: R0

```
IBTD22 : MOV A,R0
        PUSH A
        MOV R7,#03H
        CLR A
IBD221 : MOV @R0,A
        INC R0
        DJNZ R7,IBD221
        POP A
        MOV R0,A
        MOV R7,#10H
IBD222 : PUSH A
        CLR C
        MOV A,R4
        RLC A
        MOV R4,A
        MOV A,R3
        RLC A
        MOV R3,A
        MOV B,#03H
IBD223 : MOV A,@R0
        ADDC A,@R0
        DA A
        MOV @R0,A
        INC R0
        DJNZ B,IBD223
        POP A
        MOV R0,A
        DJNZ R7,IBD222
        RET
```

;二进制整数(3 位)转换为十进制整数(分离 BCD 码)

;入口: R2,R3,R4

;占用资源: ACC,R2,NDIV31

;堆栈需求: 5 字节

;出口: R0,NCNT

```
IBTD31 : CLR A
        MOV NCNT,A
IBD311 : MOV R7,#0AH
        LCALL NDIV31
        MOV A,R7
        MOV @R0,A
        INC R0
        INC NCNT
        MOV A,R2
        ORL A,R3
```

```

    ORL    A,R4
    JNZ    IBD311
    MOV    A,R0
    CLR    C
    SUBB   A,NCNT
    MOV    R0,A
    RET

```

;二进制整数(3 位)转换为十进制整数(组合 BCD 码)

;入口: R2,R3,R4

;占用资源: ACC,B,R7

;堆栈需求: 3 字节

;出口: R0

```

IBTD32 :  MOV    A,R0
        PUSH   A
        MOV    R7,#04H
        CLR    A
IBD321 :  MOV    @R0,A
        INC    R0
        DJNZ   R7,IBD321
        POP    A
        MOV    R0,A
        MOV    R7,#18H
IBD322 :  PUSH   A
        CLR    C
        MOV    A,R4
        RLC    A
        MOV    R4,A
        MOV    A,R3
        RLC    A
        MOV    R3,A
        MOV    A,R2
        RLC    A
        MOV    R2,A
        MOV    B,#04H
IBD323 :  MOV    A,@R0
        ADDC   A,@R0
        DA     A
        MOV    @R0,A
        INC    R0
        DJNZ   B,IBD323
        POP    A
        MOV    R0,A
        DJNZ   R7,IBD322
        RET

```

;二进制整数(M 位)转换为十进制整数(组合 BCD 码)

;入口: R1,M,N  
;占用资源: ACC,B,R2,R3,R7  
;堆栈需求: 2 字节  
;出口: R0

```
IBTDMN : MOV  A,R0
        MOV  R2,A
        MOV  A,R1
        MOV  R3,A
        MOV  B,N
        CLR  A
IBDMN1  : MOV  @R0,A
        INC  R0
        DJNZ B,IBDMN1
        MOV  A,M
        SWAP A
        RR   A
        CLR  C
        MOV  R7,A
IBDMN2  : MOV  A,R2
        MOV  R0,A
        MOV  A,R3
        MOV  R1,A
        MOV  B,M
IBDMN3  : MOV  A,@R1
        RLC  A
        MOV  @R1,A
        INC  R1
        DJNZ B,IBDMN3
        MOV  B,N
IBDMN4  : MOV  A,@R0
        ADDC A,@R0
        DA   A
        JNB  ACC.4,IBDMN5
        SETB C
        CLR  ACC.4
IBDMN5  : MOV  @R0,A
        INC  R0
        DJNZ B,IBDMN4
        DJNZ R7,IBDMN2
        MOV  A,R2
        MOV  R0,A
        RET
```

;附:

;三字节无符号除法程序 $(R2R3R4/R7)=(R2)R3R4$  余数 R7

;入口: R2,R3,R4,R7  
 ;占用资源: ACC,B,F0  
 ;堆栈需求: 3 字节  
 ;出口: (R2),R3,R4,R7,OV

```
NDIV31 : MOV A,R2
        MOV B,R7
        DIV AB
        PUSH A
        MOV R2,B
        MOV B,#10H
NDV311 : CLR C
        MOV A,R4
        RLC A
        MOV R4,A
        MOV A,R3
        RLC A
        MOV R3,A
        MOV A,R2
        RLC A
        MOV R2,A
        MOV F0,C
        CLR C
        SUBB A,R7
        JB F0,NDV312
        JC NDV313
NDV312 : MOV R2,A
        INC R4
NDV313 : DJNZ B,NDV311
        POP A
        CLR OV
        JZ NDV314
        SETB OV
NDV314 : XCH A,R2
        MOV R7,A
        RET
```

#### (一) M C S - 5 1 定点运算子程序库及其使用说明

定点运算子程序库文件名为 DQ51.ASM，为便于使用，先将有关约定说明如下：

- 1．多字节定点操作数：用[R0]或[R1]来表示存放在由 R0 或 R1 指示的连续单元中的数据。地址小的单元存放数据的高字节。例如：[R0]=123456H，若(R0)=30H，则(30H)=12H，(31H)=34H，(32H)=56H。
- 2．运算精度：单次定点运算精度为结果最低位的当量值。
- 3．工作区：数据工作区固定在 PSW、A、B、R2～R7，用户只要不在工作区中存放无关的或非消耗性的信息，程序就具有较好的透明性。

(1) 标号：BCDA 功能：多字节BCD码加法

入口条件：字节数在 R7 中，被加数在[R0]中，加数在[R1]中。

出口信息：和在[R0]中，最高位进位在 CY 中。

影响资源：PSW、A、R2 堆栈需求：2 字节

BCDA: MOV A,R7 ;取字节数至 R2 中

MOV R2,A

ADD A,R0 ;初始化数据指针

MOV R0,A

MOV A,R2

ADD A,R1

MOV R1,A

CLR C

BCD1: DEC R0 ;调整数据指针

DEC R1

MOV A,@R0

ADDC A,@R1 ;按字节相加

DAA ;十进制调整

MOV @R0,A ;和存回[R0]中

DJNZ R2,BCD1 ;处理完所有字节

RET

( 2 ) 标号：BCDB 功能：多字节BCD码减法

入口条件：字节数在 R7 中，被减数在[R0]中，减数在[R1]中。

出口信息：差在[R0]中，最高位借位在 CY 中。

影响资源：PSW、A、R2、R3 堆栈需求：6 字节

BCDB: LCALL NEG1 ;减数[R1]十进制取补

LCALL BCDA ;按多字节BCD码加法处理

CPL C ;将补码加法的进位标志转换成借位标志

MOV F0,C ;保护借位标志

LCALL NEG1 ;恢复减数[R1]的原始值

MOV C,F0 ;恢复借位标志

RET

NEG1: MOV A,R0 ;[R1]十进制取补子程序入口

XCH A,R1 ;交换指针

XCH A,R0

LCALL NEG ;通过[R0]实现[R1]取补

MOV A,R0

XCH A,R1 ;换回指针

XCH A,R0

RET

( 3 ) 标号：NEG 功能：多字节BCD码取补

入口条件：字节数在 R7 中，操作数在[R0]中。

出口信息：结果仍在[R0]中。

影响资源：PSW、A、R2、R3 堆栈需求：2 字节

NEG: MOV A,R7 ;取(字节数减一)至 R2 中

DEC A

MOV R2,A

MOV A,R0 ;保护指针

MOV R3,A

```

NEG0:  CLR C
        MOV A,#99H
        SUBB A,@R0 ;按字节十进制取补
        MOV @R0,A ;存回[R0]中
        INC R0 ;调整数据指针
        DJNZ R2,NEG0 ;处理完(R2)字节
        MOV A,#9AH ;最低字节单独取补
        SUBB A,@R0
        MOV @R0,A
        MOV A,R3 ;恢复指针
        MOV R0,A
        RET

```

(4) 标号: BRLN 功能: 多字节BCD码左移十进制一位(乘十)

入口条件: 字节数在R7中, 操作数在[R0]中。  
 出口信息: 结果仍在[R0]中, 移出的十进制最高位在R3中。  
 影响资源: PSW、A、R2、R3 堆栈需求: 2字节

```

BRLN:  MOV A,R7 ;取字节数至R2中
        MOV R2,A
        ADD A,R0 ;初始化数据指针
        MOV R0,A
        MOV R3,#0 ;工作单元初始化
BRL1:  DEC R0 ;调整数据指针
        MOV A,@R0 ;取一字节
        SWAP A ;交换十进制高低位
        MOV @R0,A ;存回
        MOV A,R3 ;取低字节移出的十进制高位
        XCHD A,@R0 ;换出本字节的十进制高位
        MOV R3,A ;保存本字节的十进制高位
        DJNZ R2,BRL1 ;处理完所有字节
        RET

```

(5) 标号: MUL D 功能: 双字节二进制无符号数乘法

入口条件: 被乘数在R2、R3中, 乘数在R6、R7中。  
 出口信息: 乘积在R2、R3、R4、R5中。  
 影响资源: PSW、A、B、R2~R7 堆栈需求: 2字节

```

MULD:  MOV A,R3 ;计算R3乘R7
        MOV B,R7
        MULAB
        MOV R4,B ;暂存部分积
        MOV R5,A
        MOV A,R3 ;计算R3乘R6
        MOV B,R6
        MULAB
        ADD A,R4 ;累加部分积
        MOV R4,A

```

```

CLR A
ADDC A,B
MOV R3,A
MOV A,R2 ; 计算 R2 乘 R7
MOV B,R7
MULAB
ADD A,R4 ; 累加部分积
MOV R4,A
MOV A,R3
ADDC A,B
MOV R3,A
CLR A
RLC A
XCH A,R2 ; 计算 R2 乘 R6
MOV B,R6
MULAB
ADD A,R3 ; 累加部分积
MOV R3,A
MOV A,R2
ADDC A,B
MOV R2,A
RET

```

( 6 ) 标号 : MUL 2 功能 : 双字节二进制无符号数平方

入口条件 : 待平方数在 R2、R3 中。

出口信息 : 结果在 R2、R3、R4、R5 中。

影响资源 : PSW、A、B、R2 ~ R5 堆栈需求 : 2 字节

```

MUL2:  MOV A,R3 ; 计算 R3 平方
        MOV B,A
        MULAB
        MOV R4,B ; 暂存部分积
        MOV R5,A
        MOV A,R2 ; 计算 R2 平方
        MOV B,A
        MULAB
        XCH A,R3 ; 暂存部分积,并换出 R2 和 R3
        XCH A,B
        XCH A,R2
        MULAB ; 计算  $2 \times R2 \times R3$ 
        CLR C
        RLC A
        XCH A,B
        RLC A
        JNC MU20
        INC R2 ; 累加溢出量
MU20:  XCH A,B ; 累加部分积
        ADD A,R4

```

```

MOV R4,A
MOV A,R3
ADDC A,B
MOV R3,A
CLR A
ADDC A,R2
MOV R2,A
RET

```

( 7 ) 标号 : D I V D 功能 : 双字节二进制无符号数除法

入口条件 : 被除数在 R2、R3、R4、R5 中 , 除数在 R6、R7 中。

出口信息 : OV=0 时 , 双字节商在 R2、R3 中 , OV=1 时溢出。

影响资源 : PSW、A、B、R1~R7 堆栈需求 : 2 字节

DIVD: CLR C ; 比较被除数和除数

```

MOV A,R3
SUBB A,R7
MOV A,R2
SUBB A,R6
JC DVD1
SETB OV ; 溢出
RET

```

DVD1: MOV B,#10H ; 计算双字节商

DVD2: CLR C ; 部分商和余数同时左移一位

```

MOV A,R5
RLC A
MOV R5,A
MOV A,R4
RLC A
MOV R4,A
MOV A,R3
RLC A
MOV R3,A
XCH A,R2
RLC A
XCH A,R2
MOV F0,C ; 保存溢出位
CLR C
SUBB A,R7 ; 计算 ( R2R3 - R6R7 )
MOV R1,A
MOV A,R2
SUBB A,R6
ANL C,F0 ; 结果判断
JC DVD3
MOV R2,A ; 够减 , 存放新的余数
MOV A,R1
MOV R3,A
INC R5 ; 商的低位置一

```



```

DVD3: DJNZ B,DVD2 ; 计算完十六位商 (R4R5)
      MOV A,R4 ; 将商移到 R2R3 中
      MOV R2,A
      MOV A,R5
      MOV R3,A
      CLR OV ; 设立成功标志
      RET

```

( 8 ) 标号 : D 4 5 7 功能 : 双字节二进制无符号数除以单字节二进制数

入口条件 : 被除数在 R4、R5 中 , 除数在 R7 中。

出口信息 : OV=0 时 , 单字节商在 R3 中 , OV=1 时溢出。

影响资源 : PSW、A、R3 ~ R7 堆栈需求 : 2 字节

```

D457:  CLR C
      MOV A,R4
      SUBB A,R7
      JC DV50
      SETB OV ; 商溢出
      RET
DV50:  MOV R6,#8 ; 求平均值 (R4R5 / R7 - R3)
DV51:  MOV A,R5
      RLC A
      MOV R5,A
      MOV A,R4
      RLC A
      MOV R4,A
      MOV F0,C
      CLR C
      SUBB A,R7
      ANL C,/F0
      JC DV52
      MOV R4,A
DV52:  CPL C
      MOV A,R3
      RLC A
      MOV R3,A
      DJNZ R6,DV51
      MOV A,R4 ; 四舍五入
      ADD A,R4
      JC DV53
      SUBB A,R7
      JC DV54
DV53:  INC R3
DV54:  CLR OV
      RET

```

( 9 ) 标号 : D V 3 1 功能 : 三字节二进制无符号数除以单字节二进制数

入口条件：被除数在 R3、R4、R5 中，除数在 R7 中。

出口信息：OV=0 时，双字节商在 R4、R5 中，OV=1 时溢出。

影响资源：PSW、A、B、R2~R7 堆栈需求：2 字节

DV31: CLR C

MOV A,R3

SUBB A,R7

JC DV30

SETB OV ; 商溢出

RET

DV30: MOV R2,#10H ; 求 R3R4R5 / R7 - R4R5

DM23: CLR C

MOV A,R5

RLC A

MOV R5,A

MOV A,R4

RLC A

MOV R4,A

MOV A,R3

RLC A

MOV R3,A

MOV F0,C

CLR C

SUBB A,R7

ANL C,F0

JC DM24

MOV R3,A

INC R5

DM24: DJNZ R2,DM23

MOV A,R3 ; 四舍五入

ADD A,R3

JC DM25

SUBB A,R7

JC DM26

DM25: INC R5

MOV A,R5

JNZ DM26

INC R4

DM26: CLR OV

RET ; 商在 R4R5 中

( 1 0 ) 标号：MULS 功能：双字节二进制有符号数乘法（补码）

入口条件：被乘数在 R2、R3 中，乘数在 R6、R7 中。

出口信息：乘积在 R2、R3、R4、R5 中。

影响资源：PSW、A、B、R2~R7 堆栈需求：4 字节

MULS: MOV R4,#0 ; 清零 R4R5

MOV R5,#0

LCALL MDS ; 计算结果的符号和两个操作数的绝对值

LCALL MULD ; 计算两个绝对值的乘积  
SJMP MDSE ; 用补码表示结果

( 1 1 ) 标号: DIVS 功能: 双字节二进制有符号数除法 (补码)

入口条件: 被除数在 R2、R3、R4、R5 中, 除数在 R6、R7 中。

出口信息: OV=0 时商在 R2、R3 中, OV=1 时溢出。

影响资源: PSW、A、B、R1~R7 堆栈需求: 5 字节

DIVS: LCALL MDS ; 计算结果的符号和两个操作数的绝对值

PUSH PSW ; 保存结果的符号

LCALL DIVD ; 计算两个绝对值的商

JNB OV,DVS1 ; 溢出否?

POP ACC ; 溢出, 放去结果的符号, 保留溢出标志

RET

DVS1: POP PSW ; 未溢出, 取出结果的符号

MOV R4,#0

MOV R5,#0

MDSE: JB F0,MDS2 ; 用补码表示结果

CLR OV ; 结果为正, 原码即补码, 计算成功

RET

MDS: CLR F0 ; 结果符号初始化

MOV A,R6 ; 判断第二操作数的符号

JNB ACC.7,MDS1 ; 为正, 不必处理

CPL F0 ; 为负, 结果符号取反

XCH A,R7 ; 第二操作数取补, 得到其绝对值

CPL A

ADD A,#1

XCH A,R7

CPL A

ADDC A,#0

MOV R6,A

MDS1: MOV A,R2 ; 判断第一操作数或运算结果的符号

JNB ACC.7,MDS3 ; 为正, 不必处理

CPL F0 ; 为负, 结果符号取反

MDS2: MOV A,R5 ; 求第一操作数的绝对值或运算结果的补码

CPL A

ADD A,#1

MOV R5,A

MOV A,R4

CPL A

ADDC A,#0

MOV R4,A

MOV A,R3

CPL A

ADDC A,#0

MOV R3,A

MOV A,R2

CPL A

```

        ADDC A,#0
        MOV R2,A
MDS3: CLR OV ;运算成功
        RET

```

( 1 2 ) 标号： S H 2 功能：双字节二进制无符号数开平方（快速）

入口条件：被开方数在 R2、R3 中。

出口信息：平方根仍在 R2、R3 中，整数部分的位数为原数的一半，其余为小数。

影响资源：PSW、A、B、R2~R7 堆栈需求：2 字节

```

SH2: MOV A,R2
      ORL A,R3
      JNZ SH20
      RET ;被开方数为零，不必运算
SH20: MOV R7,#0 ;左规次数初始化
      MOV A,R2
SH22: ANL A,#0C0H ;被开方数高字节小于 4 0 H 否？
      JNZ SQRH ;不小于 4 0 H，左规格化完成，转开方过程
      CLR C ;每左规一次，被开方数左移两位
      MOV A,R3
      RLC A
      MOV F0,C
      CLR C
      RLC A
      MOV R3,A
      MOV A,R2
      MOV ACC.7,C
      MOV C,F0
      RLC A
      RLC A
      MOV R2,A
      INC R7 ;左规次数加一
      SJMP SH22 ;继续左规

```

( 1 3 ) 标号： S H 4 功能：四字节二进制无符号数开平方（快速）

入口条件：被开方数在 R2、R3、R4、R5 中。

出口信息：平方根在 R2、R3 中，整数部分的位数为原数的一半，其余为小数。

影响资源：PSW、A、B、R2~R7 堆栈需求：2 字节

```

SH4: MOV A,R2
      ORL A,R3
      ORL A,R4
      ORL A,R5
      JNZ SH40
      RET ;被开方数为零，不必运算
SH40: MOV R7,#0 ;左规次数初始化
      MOV A,R2
SH41: ANL A,#0C0H ;被开方数高字节小于 4 0 H 否？

```

```

    JNZ SQRH ; 不小于 4 0 H , 左规格化完成
    MOV R6,#2 ; 每左规一次,被开方数左移两位
SH42: CLR C ; 被开方数左移一位
    MOV A,R5
    RLC A
    MOV R5,A
    MOV A,R4
    RLC A
    MOV R4,A
    MOV A,R3
    RLC A
    MOV R3,A
    MOV A,R2
    RLC A
    MOV R2,A
    DJNZ R6,SH42 ; 被开方数左移完两位
    INC R7 ; 左规次数加一
    SJMP SH41 ; 继续左规
SQRH: MOV A,R2 ; 规格化后高字节按折线法分为三个区间
    ADD A,#57H
    JC SQR2
    ADD A,#45H
    JC SQR1
    ADD A,#24H
    MOV B,#0E3H ; 第一区间的斜率
    MOV R4,#80H ; 第一区间的平方根基数
    SJMP SQR3
SQR1: MOV B,#0B2H ; 第二区间的斜率
    MOV R4,#0A0H ; 第二区间的平方根基数
    SJMP SQR3
SQR2: MOV B,#8DH ; 第三区间的斜率
    MOV R4,#0D0H ; 第三区间的平方根基数
SQR3: MULAB ; 与区间基点的偏移量乘区间斜率
    MOV A,B
    ADD A,R4 ; 累加到平方根的基数上
    MOV R4,A
    MOV B,A
    MULAB ; 求当前平方根的幂
    XCH A,R3 ; 求偏移量 ( 存放在 R2R3 中 )
    CLR C
    SUBB A,R3
    MOV R3,A
    MOV A,R2
    SUBB A,B
    MOV R2,A
SQR4: SETB C ; 用减奇数法校正一个字节的平方根
    MOV A,R4 ; 当前平方根的两倍加一存入 R5R6 中
    RLC A

```

```

MOV R6,A
CLR A
RLC A
MOV R5,A
MOV A,R3 ; 偏移量小于该奇数否?
SUBB A,R6
MOV B,A
MOV A,R2
SUBB A,R5
JC SQR5 ; 小于,校正结束,已达到一个字节的精度
INC R4 ; 不小于,平方根加一
MOV R2,A ; 保存新的偏移量
MOV R3,B
SJMP SQR4 ; 继续校正
SQR5: MOV A,R4 ; 将一个字节精度的根存入 R2
      XCH A,R2
      RRC A
      MOV F0,C ; 保存最终偏移量的最高位
      MOV A,R3
      MOV R5,A ; 将最终偏移量的低八位存入 R5 中
      MOV R4,#8 ; 通过 (R5R6 / R2) 求根的低字节
SQR6: CLR C
      MOV A,R3
      RLC A
      MOV R3,A
      CLR C
      MOV A,R5
      SUBB A,R2
      JB F0,SQR7
      JC SQR8
SQR7: MOV R5,A
      INC R3
SQR8: CLR C
      MOV A,R5
      RLC A
      MOV R5,A
      MOV F0,C
      DJNZ R4,SQR6 ; 根的第二字节计算完,在 R3 中
      MOV A,R7 ; 取原被开方数的左规次数
      JZ SQRE ; 未左规,开方结束
SQR9: CLR C ; 按左规次数右移平方根,得到实际根
      MOV A,R2
      RRC A
      MOV R2,A
      MOV A,R3
      RRC A
      MOV R3,A
      DJNZ R7,SQR9

```

SQRE: RET

( 1 4 ) 标号： H A S C 功能：单字节十六进制数转换成双字节 ASCII 码

入口条件：待转换的单字节十六进制数在累加器 A 中。

出口信息：高四位的 ASCII 码在 A 中，低四位的 ASCII 码在 B 中。

影响资源：PSW、A、B 堆栈需求： 4 字节

HASC: MOV B,A ; 暂存待转换的单字节十六进制数

LCALL HAS1 ; 转换低四位

XCH A,B ; 存放低四位的 ASCII 码

SWAP A ; 准备转换高四位

HAS1: ANLA,#0FH ; 将累加器的低四位转换成 ASCII 码

ADD A,#90H

DAA

ADDC A,#40H

DAA

RET

( 1 5 ) 标号： A S C H 功能：ASCII 码转换成十六进制数

入口条件：待转换的 ASCII 码 ( 30H ~ 39H 或 41H ~ 46H ) 在 A 中。

出口信息：转换后的十六进制数 ( 00H ~ 0FH ) 仍在累加器 A 中。

影响资源：PSW、A 堆栈需求： 2 字节

ASCH: CLR C

SUBB A,#30H

JNB ACC.4,ASH1

SUBB A,#7

ASH1: RET

( 1 6 ) 标号： H B C D 功能：单字节十六进制整数转换成单字节 B C D 码整数

入口条件：待转换的单字节十六进制整数在累加器 A 中。

出口信息：转换后的 B C D 码整数 ( 十位和个位 ) 仍在累加器 A 中，百位在 R3 中。

影响资源：PSW、A、B、R3 堆栈需求： 2 字节

HBCD: MOV B,#100 ; 分离出百位，存放在 R3 中

DIV AB

MOV R3,A

MOV A,#10 ; 余数继续分离十位和个位

XCH A,B

DIV AB

SWAP A

ORL A,B ; 将十位和个位拼装成 B C D 码

RET

( 1 7 ) 标号： H B 2 功能：双字节十六进制整数转换成双字节 B C D 码整数

入口条件：待转换的双字节十六进制整数在 R6、R7 中。

出口信息：转换后的三字节 B C D 码整数在 R3、R4、R5 中。

影响资源：PSW、A、R2~R7 堆栈需求：2 字节

HB2: CLR A ; B C D 码初始化

MOV R3,A

MOV R4,A

MOV R5,A

MOV R2,#10H ; 转换双字节十六进制整数

HB3: MOV A,R7 ; 从高端移出待转换数的一位到 CY 中

RLC A

MOV R7,A

MOV A,R6

RLC A

MOV R6,A

MOV A,R5 ; B C D 码带进位自身相加，相当于乘 2

ADDC A,R5

DAA ; 十进制调整

MOV R5,A

MOV A,R4

ADDC A,R4

DAA

MOV R4,A

MOV A,R3

ADDC A,R3

MOV R3,A ; 双字节十六进制数的万位数不超过 6，不用调整

DJNZ R2,HB3 ; 处理完 1 6 b i t

RET

( 1 8 ) 标号：H B D 功能：单字节十六进制小数转换成单字节 B C D 码小数

入口条件：待转换的单字节十六进制小数在累加器 A 中。

出口信息：CY=0 时转换后的 B C D 码小数仍在 A 中。CY=1 时原小数接近整数 1。

影响资源：PSW、A、B 堆栈需求：2 字节

HBD: MOV B,#100 ; 原小数扩大一百倍

MUL AB

RLC A ; 余数部分四舍五入

CLR A

ADDC A,B

MOV B,#10 ; 分离出十分位和百分位

DIV AB

SWAP A

ADD A,B ; 拼装成单字节 B C D 码小数

DAA ; 调整后若有进位，原小数接近整数 1

RET

( 1 9 ) 标号：H B D 2 功能：双字节十六进制小数转换成双字节 B C D 码小数

入口条件：待转换的双字节十六进制小数在 R2、R3 中。

出口信息：转换后的双字节 B C D 码小数仍在 R2、R3 中。

影响资源：PSW、A、B、R2、R3、R4、R5 堆栈需求：6 字节



```

HBD2: MOV R4,#4 ;四位十进制码
HBD3: MOV A,R3 ;原小数扩大十倍
      MOV B,#10
      MULAB
      MOV R3,A
      MOV R5,B
      MOV A,R2
      MOV B,#10
      MULAB
      ADD A,R5
      MOV R2,A
      CLRA
      ADDC A,B
      PUSH ACC ;保存溢出的一位十进制码
      DJNZ R4,HBD3 ;计算完四位十进制码
      POP ACC ;取出万分位
      MOV R3,A
      POP ACC ;取出千分位
      SWAPA
      ORL A,R3 ;拼装成低字节 B C D 码小数
      MOV R3,A
      POP ACC ;取出百分位
      MOV R2,A
      POP ACC ;取出十分位
      SWAPA
      ORL A,R2 ;拼装成高字节 B C D 码小数
      MOV R2,A
      RET

```

( 2 0 ) 标号：BCDH 功能：单字节 B C D 码整数转换成单字节十六进制整数

入口条件：待转换的单字节 B C D 码整数在累加器 A 中。  
 出口信息：转换后的单字节十六进制整数仍在累加器 A 中。  
 影响资源：PSW、A、B、R4 堆栈需求：2 字节

```

BCDH: MOV B,#10H ;分离十位和个位
      DIV AB
      MOV R4,B ;暂存个位
      MOV B,#10 ;将十位转换成十六进制
      MULAB
      ADD A,R4 ;按十六进制加上个位
      RET

```

( 2 1 ) 标号：BH2 功能：双字节 B C D 码整数转换成双字节十六进制整数

入口条件：待转换的双字节 B C D 码整数在 R2、R3 中。  
 出口信息：转换后的双字节十六进制整数仍在 R2、R3 中。  
 影响资源：PSW、A、B、R2、R3、R4 堆栈需求：4 字节  
 BH2: MOV A,R3 ;将低字节转换成十六进制

```

LCALL BCDH
MOV R3,A
MOV A,R2 ; 将高字节转换成十六进制
LCALL BCDH
MOV B,#100 ; 扩大一百倍
MUL AB
ADD A,R3 ; 和低字节按十六进制相加
MOV R3,A
CLR A
ADDC A,B
MOV R2,A
RET

```

( 2 2 ) 标号： B H D 功能：单字节 B C D 码小数转换成单字节十六进制小数

入口条件：待转换的单字节 B C D 码数在累加器 A 中。

出口信息：转换后的单字节十六进制小数仍在累加器 A 中。

影响资源：PSW、A、R2、R3 堆栈需求：2 字节

BHD: MOV R2,#8 ; 准备计算一个字节小数

BHD0: ADD A,ACC ; 按十进制倍增

DAA

XCH A,R3

RLC A ; 将进位标志移入结果中

XCH A,R3

DJNZ R2,BHD0 ; 共计算 8 b i t 小数

ADD A,#0B0H ; 剩余部分达到 0 . 5 0 否？

JNC BHD1 ; 四舍

INCR3 ; 五入

BHD1: MOV A,R3 ; 取结果

RET

( 2 3 ) 标号： B H D 2 功能：双字节 B C D 码小数转换成双字节十六进制小数

入口条件：待转换的双字节 B C D 码小数在 R4、R5 中。

出口信息：转换后的双字节十六进制小数在 R2、R3 中。\*

影响资源：PSW、A、R2 ~ R6 堆栈需求：2 字节

BHD2: MOV R6,#10H ; 准备计算两个字节小数

BHD3: MOV A,R5 ; 按十进制倍增

ADD A,R5

DAA

MOV R5,A

MOV A,R4

ADDC A,R4

DAA

MOV R4,A

MOV A,R3 ; 将进位标志移入结果中

RLC A

MOV R3,A

```

MOV A,R2
RLC A
MOV R2,A
DJNZ R6,BHD3 ; 共计算 1 6 b i t 小数
MOV A,R4
ADD A,#0B0H ; 剩余部分达到 0 . 5 0 否?
JNC BHD4 ; 四舍
INCR3 ; 五入
MOV A,R3
JNZ BHD4
INCR2
BHD4: RET

```

( 2 4 ) 标号： MM 功能：求单字节十六进制无符号数据块的极值

入口条件：数据块的首址在 DPTR 中，数据个数在 R7 中。

出口信息：最大值在 R6 中，地址在 R2R3 中；最小值在 R7 中，地址在 R4R5 中。

影响资源：PSW、A、B、R1~R7 堆栈需求： 4 字节

```

MM: MOV B,R7 ; 保存数据个数
    MOVX A,@DPTR ; 读取第一个数据
    MOV R6,A ; 作为最大值的初始值
    MOV R7,A ; 也作为最小值的初始值
    MOV A,DPL ; 取第一个数据的地址
    MOV R3,A ; 作为最大值存放地址的初始值
    MOV R5,A ; 也作为最小值存放地址的初始值
    MOV A,DPH
    MOV R2,A
    MOV R4,A
    MOV A,B ; 取数据个数
    DECA ; 减一，得到需要比较的次数
    JZ MME ; 只有一个数据，不需要比较
    MOV R1,A ; 保存比较次数
    PUSH DPL ; 保护数据块的首址
    PUSH DPH
MM1: INC DPTR ; 指向一个新的数据
    MOVX A,@DPTR ; 读取这个数据
    MOV B,A ; 保存
    SETB C ; 与最大值比较
    SUBB A,R6
    JC MM2 ; 不超过当前最大值，保持当前最大值
    MOV R6,B ; 超过当前最大值，更新最大值存放地址
    MOV R2,DPH ; 同时更新最大值存放地址
    MOV R3,DPL
    SJMP MM3
MM2: MOV A,B ; 与最小值比较
    CLR C
    SUBB A,R7
    JNC MM3 ; 大于或等于当前最小值，保持当前最小值

```

```

MOV R7,B ;更新最小值
MOV R4,DPH ;更新最小值存放地址
MOV R5,DPL
MM3: DJNZ R1,MM1 ;处理完全部数据
POP DPH ;恢复数据首址
POP DPL
MME: RET

```

( 2 5 ) 标号： MMS 功能：求单字节十六进制有符号数据块的极值

入口条件：数据块的首址在 DPTR 中，数据个数在 R7 中。  
 出口信息：最大值在 R6 中，地址在 R2R3 中；最小值在 R7 中，地址在 R4R5 中。  
 影响资源：PSW、A、B、R1~R7 堆栈需求：4 字节

```

MMS: MOV B,R7 ;保存数据个数
MOVX A,@DPTR ;读取第一个数据
MOV R6,A ;作为最大值的初始值
MOV R7,A ;也作为最小值的初始值
MOV A,DPL ;取第一个数据的地址
MOV R3,A ;作为最大值存放地址的初始值
MOV R5,A ;也作为最小值存放地址的初始值
MOV A,DPH
MOV R2,A
MOV R4,A
MOV A,B ;取数据个数
DECA ;减一，得到需要比较的次数
JZ MMSE ;只有一个数据，不需要比较
MOV R1,A ;保存比较次数
PUSH DPL ;保护数据块的首址
PUSH DPH
MMS1: INC DPTR ;调整数据指针
MOVX A,@DPTR ;读取一个数据
MOV B,A ;保存
SETB C ;与最大值比较
SUBB A,R6
JZ MMS4 ;相同，不更新最大值
JNB OV,MMS2 ;差未溢出，符号位有效
CPL ACC.7 ;差溢出，符号位取反
MMS2: JB ACC.7,MMS4 ;差为负，不更新最大值
MOV R6,B ;更新最大值
MOV R2,DPH ;更新最大值存放地址
MOV R3,DPL
SJMP MMS7
MMS4: MOV A,B ;与最小值比较
CLR C
SUBB A,R7
JNB OV,MMS6 ;差未溢出，符号位有效
CPL ACC.7 ;差溢出，符号位取反
MMS6: JNB ACC.7,MMS7 ;差为正，不更新最小值

```

```

MOV R7,B ;更新最小值
MOV R4,DPH ;更新最小值存放地址
MOV R5,DPL
MMS7: DJNZ R1,MMS1 ;处理完全部数据
      POPDPH ;恢复数据首址
      POPDPL
MMSE: RET

```

( 2 6 ) 标号： F D S 1 功能：顺序查找 ( R O M ) 单字节表格

入口条件：待查找的内容在 A 中，表格首址在 DPTR 中，表格的字节数在 R7 中。

出口信息：OV=0 时，顺序号在累加器 A 中；OV=1 时，未找到。

影响资源：PSW、A、B、R2、R6 堆栈需求：2 字节

```

FDS1: MOV B,A ;保存待查找的内容
      MOV R2,#0 ;顺序号初始化 (指向表首)
      MOV A,R7 ;保存表格的长度
      MOV R6,A
FD11: MOV A,R2 ;按顺序号读取表格内容
      MOVC A,@A+DPTR
      CJNE A,B,FD12 ;与待查找的内容比较
      CLR OV ;相同，查找成功
      MOV A,R2 ;取对应的顺序号
      RET
FD12: INC R2 ;指向表格中的下一个内容
      DJNZ R6,FD11 ;查完全部表格内容
      SETB OV ;未查找到，失败
      RET

```

( 2 7 ) 标号： F D S 2 功能：顺序查找 ( R O M ) 双字节表格

入口条件：查找内容在 R4、R5 中，表格首址在 DPTR 中，数据总个数在 R7 中。

出口信息：OV=0 时顺序号在累加器 A 中，地址在 DPTR 中；OV=1 时未找到。

影响资源：PSW、A、R2、R6、DPTR 堆栈需求：2 字节

```

FDS2: MOV A,R7 ;保存表格中数据的个数
      MOV R6,A
      MOV R2,#0 ;顺序号初始化 (指向表首)
FD21: CLRA ;读取表格内容的高字节
      MOVC A,@A+DPTR
      XRL A,R4 ;与待查找内容的高字节比较
      JNZ FD22
      MOV A,#1 ;读取表格内容的低字节
      MOVC A,@A+DPTR
      XRL A,R5 ;与待查找内容的低字节比较
      JNZ FD22
      CLR OV ;相同，查找成功
      MOV A,R2 ;取对应的顺序号
      RET
FD22: INC DPTR ;指向下一个数据

```

```

INC DPTR
INC R2 ; 顺序号加一
DJNZ R6,FD21 ; 查完全部数据
SETB OV ; 未查找到,失败
RET

```

( 2 8 ) 标号 : F D D 1 功能 : 对分查找 ( R O M ) 单字节无符号增序数据表格

入口条件 : 待查找的内容在累加器 A 中, 表格首址在 DPTR 中, 字节数在 R7 中。

出口信息 : OV=0 时, 顺序号在累加器 A 中; OV=1 时, 未找到。

影响资源 : PSW、A、B、R2、R3、R4 堆栈需求 : 2 字节

```

FDD1: MOV B,A ; 保存待查找的内容
      MOV R2,#0 ; 区间低端指针初始化 (指向第一个数据)
      MOV A,R7
      DEC A
      MOV R3,A ; 区间高端指针初始化 (指向最后一个数据)
FD61: CLR C ; 判断区间大小
      MOV A,R3
      SUBB A,R2
      JC FD69 ; 区间消失, 查找失败
      RRC A ; 取区间大小的一半
      ADD A,R2 ; 加上区间的低端
      MOV R4,A ; 得到区间的中心
      MOVC A,@A+DPTR ; 读取该点的内容
      CJNE A,B,FD65 ; 与待查找的内容比较
      CLR OV ; 相同, 查找成功
      MOV A,R4 ; 取顺序号
      RET

```

FD65: JC FD68 ; 该点的内容比待查找的内容大否?

```

      MOV A,R4 ; 偏大, 取该点位置
      DEC A ; 减一
      MOV R3,A ; 作为新的区间高端
      SJMP FD61 ; 继续查找

```

FD68: MOV A,R4 ; 偏小, 取该点位置

```

      INC A ; 加一
      MOV R2,A ; 作为新的区间低端
      SJMP FD61 ; 继续查找

```

```

FD69: SETB OV ; 查找失败
      RET

```

( 2 9 ) 标号 : F D D 2 功能 : 对分查找 ( R O M ) 双字节无符号增序数据表格

入口条件 : 查找内容在 R4、R5 中, 表格首址在 DPTR 中, 数据个数在 R7 中。

出口信息 : OV=0 时顺序号在累加器 A 中, 址在 DPTR 中; OV=1 时未找到。

影响资源 : PSW、A、B、R1~R7、DPTR 堆栈需求 : 2 字节

```

FDD2: MOV R2,#0 ; 区间低端指针初始化 (指向第一个数据)
      MOV A,R7

```

```

    DEC A
    MOV R3,A ; 区间高端指针初始化, 指向最后一个数据
    MOV R6,DPH ; 保存表格首址
    MOV R7,DPL
FD81: CLR C ; 判断区间大小
    MOV A,R3
    SUBB A,R2
    JC FD89 ; 区间消失, 查找失败
    RRC A ; 取区间大小的一半
    ADD A,R2 ; 加上区间的低端
    MOV R1,A ; 得到区间的中心
    MOV DPH,R6
    CLR C ; 计算区间中心的地址
    RLC A
    JNC FD82
    INC DPH
FD82: ADD A,R7
    MOV DPL,A
    JNC FD83
    INC DPH
FD83: CLR A ; 读取该点的内容的高字节
    MOVC A,@A+DPTR
    MOV B,R4 ; 与待查找内容的高字节比较
    CJNE A,B,FD84 ; 不相同
    MOV A,#1 ; 读取该点的内容的低字节
    MOVC A,@A+DPTR
    MOV B,R5
    CJNE A,B,FD84 ; 与待查找内容的低字节比较
    MOV A,R1 ; 取顺序号
    CLR OV ; 查找成功
    RET
FD84: JC FD86 ; 该点的内容比待查找的内容大否?
    MOV A,R1 ; 偏大, 取该点位置
    DEC A ; 减一
    MOV R3,A ; 作为新的区间高端
    SJMP FD81 ; 继续查找
FD86: MOV A,R1 ; 偏小, 取该点位置
    INC A ; 加一
    MOV R2,A ; 作为新的区间低端
    SJMP FD81 ; 继续查找
FD89: MOV DPH,R6 ; 相同, 恢复首址
    MOV DPL,R7
    SETB OV ; 查找失败
    RET

```

( 3 0 ) 标号: DDM1 功能: 求单字节十六进制无符号数据块的平均值

入口条件：数据块的首址在 DPTR 中，数据个数在 R7 中。

出口信息：平均值在累加器 A 中。

影响资源：PSW、A、R2~R6 堆栈需求：4 字节

DDM1: MOV A,R7 ; 保存数据个数

MOV R2,A

PUSH DPH

PUSH DPL

CLR A ; 初始化累加和

MOV R4,A

MOV R5,A

DM11: MOVX A,@DPTR ; 读取一个数据

ADD A,R5 ; 累加到累加和中

MOV R5,A

JNC DM12

INC R4

DM12: INC DPTR ; 调整指针

DJNZ R2,DM11 ; 累加完全部数据

LCALL D457 ; 求平均值 (R4R5 / R7 - R3)

MOV A,R3 ; 取平均值

POP DPL

POP DPH

RET

( 3 1 ) 标号：DDM2 功能：求双字节十六进制无符号数据块的平均值

入口条件：数据块的首址在 DPTR 中，双字节数据总个数在 R7 中。

出口信息：平均值在 R4、R5 中。

影响资源：PSW、A、R2~R6 堆栈需求：4 字节

DDM2: MOV A,R7 ; 保存数据个数

MOV R2,A ; 初始化数据指针

PUSH DPL ; 保持首址

PUSH DPH

CLR A ; 初始化累加和

MOV R3,A

MOV R4,A

MOV R5,A

DM20: MOVX A,@DPTR ; 读取一个数据的高字节

MOV B,A

INC DPTR

MOVX A,@DPTR ; 读取一个数据的低字节

INC DPTR

ADD A,R5 ; 累加到累加和中

MOV R5,A

MOV A,B

ADDC A,R4

MOV R4,A

JNC DM21

INC R3



DM21: DJNZ R2,DM20 ;累加全部数据  
POP DPH ;恢复首址  
POP DPL  
LJMP DV31 ;求 R3R4R5 / R7 - R4R5 ,得到平均值

( 3 2 ) 标号: X R 1 功能: 求单字节数据块的 ( 异或 ) 校验和

入口条件: 数据块的首址在 DPTR 中, 数据的个数在 R6、R7 中。

出口信息: 校验和在累加器 A 中。

影响资源: PSW、A、B、R4 ~ R7 堆栈需求: 2 字节

XR1: MOV R4,DPH ;保存数据块的首址  
MOV R5,DPL  
MOV A,R7 ;双字节计数器调整  
JZ XR10  
INC R6  
XR10: MOV B,#0 ;校验和初始化  
XR11: MOVX A,@DPTR ;读取一个数据  
XRL B,A ;异或运算  
INC DPTR ;指向下一个数据  
DJNZ R7,XR11 ;双字节计数器减一  
DJNZ R6,XR11  
MOV DPH,R4 ;恢复数据首址  
MOV DPL,R5  
MOV A,B ;取校验和  
RET

( 3 3 ) 标号: X R 2 功能: 求双字节数据块的 ( 异或 ) 校验和

入口条件: 数据块的首址在 DPTR 中, 双字节数据总个数在 R6、R7 中。

出口信息: 校验和在 R2、R3 中。

影响资源: PSW、A、R2 ~ R7 堆栈需求: 2 字节

XR2: MOV R4,DPH ;保存数据块的首址  
MOV R5,DPL  
MOV A,R7 ;双字节计数器调整  
JZ XR20  
INC R6  
XR20: CLR A ;校验和初始化  
MOV R2,A  
MOV R3,A  
XR21: MOVX A,@DPTR ;读取一个数据的高字节  
XRL A,R2 ;异或运算  
MOV R2,A  
INC DPTR  
MOVB A,@DPTR ;读取一个数据的低字节  
XRL A,R3 ;异或运算  
MOV R3,A  
INC DPTR ;指向下一个数据  
DJNZ R7,XR21 ;双字节计数器减一

```

DJNZ R6,XR21
MOV DPH,R4 ; 恢复数据首址
MOV DPL,R5
RET

```

( 3 4 ) 标号： S O R T 功能：单字节无符号数据块排序（增序）

入口条件：数据块的首址在 R0 中，字节数在 R7 中。

出口信息：完成排序（增序）

影响资源：PSW、A、R2 ~ R6 堆栈需求：2 字节

```

SORT: MOV A,R7
      MOV R5,A ; 比较次数初始化
SRT1: CLR F0 ; 交换标志初始化
      MOV A,R5 ; 取上遍比较次数
      DEC A ; 本遍比上遍减少一次
      MOV R5,A ; 保存本遍次数
      MOV R2,A ; 复制到计数器中
      JZ SRT5 ; 若为零，排序结束
      MOV A,R0 ; 保存数据指针
      MOV R6,A
SRT2: MOV A,@R0 ; 读取一个数据
      MOV R3,A
      INC R0 ; 指向下一个数据
      MOV A,@R0 ; 再读取一个数据
      MOV R4,A
      CLR C
      SUBB A,R3 ; 比较两个数据的大小
      JNC SRT4 ; 顺序正确（增序或相同），不必交换
      SETB F0 ; 设立交换标志
      MOV A,R3 ; 将两个数据交换位置
      MOV @R0,A
      DEC R0
      MOV A,R4
      MOV @R0,A
      INC R0 ; 指向下一个数据
SRT4: DJNZ R2,SRT2 ; 完成本遍的比较次数
      MOV A,R6 ; 恢复数据首址
      MOV R0,A
      JB F0,SRT1 ; 本遍若进行过交换，则需继续排序
SRT5: RET ; 排序结束
      END

```

## (二) M C S - 5 1 浮点运算符程序库及其使用说明

本浮点程序库有三个不同层次的版本，以便适应不同的应用场合：

- 1 . 小型库 ( FQ51A.ASM ) : 只包含浮点加、减、乘、除子程序。
- 2 . 中型库 ( FQ51B.ASM ) : 在小型库的基础上再增加绝对值、倒数、比较、平方、开平方、数制转换等子程序。

3. 大型库 (FQ51.ASM): 包含本说明书中的全部子程序。

为便于读者使用本程序库, 先将有关约定说明如下:

1. 双字节定点操作数: 用[R0]或[R1]来表示存放在由 R0 或 R1 指示的连续单元中的数据, 地址小的单元存放高字节。如果[R0]=1234H, 若(R0)=30H, 则(30H)=12H, (31H)=34H。

2. 二进制浮点操作数: 用三个字节表示, 第一个字节的最高位为数符, 其余七位为阶码 (补码形式), 第二字节为尾数的高字节, 第三字节为尾数的低字节, 尾数用双字节纯小数 (原码) 来表示。当尾数的最高位为 1 时, 便称为规格化浮点数, 简称操作数。在程序说明中, 也用[R0]或[R1]来表示 R0 或 R1 指示的浮点操作数, 例如: 当[R0]=-6.000 时, 则二进制浮点数表示为 83C000H。若(R0)=30H, 则(30H)=83H, (31H)=0C0H, (32H)=00H。

3. 十进制浮点操作数: 用三个字节表示, 第一个字节的最高位为数符, 其余七位为阶码 (二进制补码形式), 第二字节为尾数的高字节, 第三字节为尾数的低字节, 尾数用双字节 B C D 码纯小数 (原码) 来表示。当十进制数的绝对值大于 1 时, 阶码就等于整数部分的位数, 如 876.5 的阶码是 03H, -876.5 的阶码是 83H; 当十进制数的绝对值小于 1 时, 阶码就等于 80H 减去小数点后面零的个数, 例如 0.00382 的阶码是 7EH, -0.00382 的阶码是 0FEH。在程序说明中, 用[R0]或[R1]来表示 R0 或 R1 指示的十进制浮点操作数。例如有一个十进制浮点操作数存放在 30H、31H、32H 中, 数值是 -0.07315, 即-0.7315 乘以 10 的-1 次方, 则(30H)=0FFH, 31H=73H, (32H)=15H。若用[R0]来指向它, 则应使(R0)=30H。

4. 运算精度: 单次定点运算精度为结果最低位的当量值; 单次二进制浮点算术运算的精度优于十万分之三; 单次二进制浮点超越函数运算的精度优于万分之一; B C D 码浮点数本身的精度比较低 (万分之一到千分之一), 不宜作为运算的操作数, 仅用于输入或输出时的数制转换。不管那种数据格式, 随着连续运算的次数增加, 精度都会下降。

5. 工作区: 数据工作区固定在 A、B、R2~R7, 数符或标志工作区固定在 PSW 和 23H 单元(位 1CH~1FH)。在浮点系统中, R2、R3、R4 和位 1FH 为第一工作区, R5、R6、R7 和位 1EH 为第二工作区。用户只要不在工作区中存放无关的或非消耗性的信息, 程序就具有较好的透明性。

6. 子程序调用范例: 由于本程序库特别注意了各子程序接口的相容性, 很容易采用积木方式 (或流水线方式) 完成一个公式的计算。以浮点运算为例:

计算  $y = \ln | \sin (a/b + d) |$

已知:  $a=-123.4$ ;  $b=0.7577$ ;  $c=56.34$ ;  $d=1.276$ ; 它们分别存放在 30H、33H、36H、39H 开始的连续三个单元中。用 B C D 码浮点数表示时, 分别为  $a=831234H$ ;  $b=007577H$ ;  $c=025634H$ ;  $d=011276H$ 。

求解过程: 通过调用 B T O F 子程序, 将各变量转换成二进制浮点操作数, 再进行各种运算, 最后调用 F T O B 子程序, 还原成十进制形式, 供输出使用。程序如下:

```
TEST: MOV R0,#39H ;指向 B C D 码浮点操作数 d
      LCALL BTOF ;将其转换成二进制浮点操作数
      MOV R0,#36H ;指向 B C D 码浮点操作数 c
      LCALL BTOF ;将其转换成二进制浮点操作数
      MOV R0,#33H ;指向 B C D 码浮点操作数 b
      LCALL BTOF ;将其转换成二进制浮点操作数
      MOV R0,#30H ;指向 B C D 码浮点操作数 a
      LCALL BTOF ;将其转换成二进制浮点操作数
      MOV R1,#33H ;指向二进制浮点操作数 b
      LCALL FMUL ;进行浮点乘法运算
      MOV R1,#36H ;指向二进制浮点操作数 c
      LCALL FDIV ;进行浮点除法运算
      MOV R1,#39H ;指向二进制浮点操作数 d
      LCALL FADD ;进行浮点加法运算
      LCALL FSIN ;进行浮点正弦运算
```

LCALL FABS ;进行浮点绝对值运算  
LCALL FSQR ;进行浮点开平方运算  
LCALL FLN ;进行浮点对数运算  
LCALL FTOB ;将结果转换成 B C D 码浮点数

STOP: LJMP STOP

END

运行结果, [R0]=804915H, 即  $y = -0.4915$ , 比较精确的结果应该是  $-0.491437$ 。

( 1 ) 标号: F S D T 功能: 浮点数格式化

入口条件: 待格式化浮点操作数在[R0]中。

出口信息: 已格式化浮点操作数仍在[R0]中。

影响资源: PSW、A、R2、R3、R4、位 1FH 堆栈需求: 6 字节

FSDT: LCALL MVR0 ;将待格式化操作数传送到第一工作区中

LCALL RLN ;通过左规完成格式化

LJMP MOV0 ;将已格式化浮点操作数传回到[R0]中

( 2 ) 标号: F A D D 功能: 浮点数加法

入口条件: 被加数在[R0]中, 加数在[R1]中。

出口信息: OV=0 时, 和仍在[R0]中, OV=1 时, 溢出。

影响资源: PSW、A、B、R2~R7、位 1EH、1FH 堆栈需求: 6 字节

FADD: CLR F0 ;设立加法标志

SJMPAS ;计算代数和

( 3 ) 标号: F S U B 功能: 浮点数减法

入口条件: 被减数在[R0]中, 减数在[R1]中。

出口信息: OV=0 时, 差仍在[R0]中, OV=1 时, 溢出。

影响资源: PSW、A、B、R2~R7、位 1EH、1FH 堆栈需求: 6 字节

FSUB: SETB F0 ;设立减法标志

AS: LCALL MVR1 ;计算代数和。先将[R1]传送到第二工作区

MOV C,F0 ;用加减标志来校正第二操作数的有效符号

RRC A

XRLA,@R1

MOV C,ACC.7

ASN: MOV 1EH,C ;将第二操作数的有效符号存入位 1EH 中

XRLA,@R0 ;与第一操作数的符号比较

RLC A

MOV F0,C ;保存比较结果

LCALL MVR0 ;将[R0]传送到第一工作区中

LCALL AS1 ;在工作寄存器中完成代数运算

MOV0: INC R0 ;将结果传回到[R0]中的子程序入口

INC R0

MOV A,R4 ;传回尾数的低字节

MOV @R0,A

DEC R0

MOV A,R3 ;传回尾数的高字节

```

MOV @R0,A
DEC R0
MOV A,R2 ;取结果的阶码
MOV C,1FH ;取结果的数符
MOV ACC.7,C ;拼入阶码中
MOV @R0,A
CLR ACC.7 ;不考虑数符
CLR OV ;清除溢出标志
CJNE A,#3FH,MV01 ;阶码是否上溢?
SETB OV ;设立溢出标志
MV01: MOV A,@R0 ;取出带数符的阶码
      RET
MVR0: MOV A,@R0 ;将[R0]传送到第一工作区中的子程序
      MOV C,ACC.7 ;将数符保存在位 1FH 中
      MOV 1FH,C
      MOV C,ACC.6 ;将阶码扩充为 8 b i t 补码
      MOV ACC.7,C
      MOV R2,A ;存放在 R2 中
      INC R0
      MOV A,@R0 ;将尾数高字节存放在 R3 中
      MOV R3,A
      INC R0
      MOV A,@R0 ;将尾数低字节存放在 R4 中
      MOV R4,A
      DEC R0 ;恢复数据指针
      DEC R0
      RET
MVR1: MOV A,@R1 ;将[R1]传送到第二工作区中的子程序
      MOV C,ACC.7 ;将数符保存在位 1EH 中
      MOV 1EH,C
      MOV C,ACC.6 ;将阶码扩充为 8 b i t 补码
      MOV ACC.7,C
      MOV R5,A ;存放在 R5 中
      INC R1
      MOV A,@R1 ;将尾数高字节存放在 R6 中
      MOV R6,A
      INC R1
      MOV A,@R1 ;将尾数低字节存放在 R7 中
      MOV R7,A
      DEC R1 ;恢复数据指针
      DEC R1
      RET
AS1: MOV A,R6 ;读取第二操作数尾数高字节
      ORL A,R7
      JZ AS2 ;第二操作数为零,不必运算
      MOV A,R3 ;读取第一操作数尾数高字节
      ORL A,R4
      JNZ EQ1

```

```
MOV A,R6 ;第一操作数为零,结果以第二操作数为准
MOV R3,A
MOV A,R7
MOV R4,A
MOV A,R5
MOV R2,A
MOV C,1EH
MOV 1FH,C
```

AS2: RET

EQ1: MOV A,R2 ;对阶,比较两个操作数的阶码

```
XRL A,R5
JZ AS4 ;阶码相同,对阶结束
JB ACC.7,EQ3 ;阶符互异
MOV A,R2 ;阶符相同,比较大小
CLR C
SUBB A,R5
JC EQ4
```

EQ2: CLR C ;第二操作数右规一次

```
MOV A,R6 ;尾数缩小一半
RRC A
MOV R6,A
MOV A,R7
RRC A
MOV R7,A
INC R5 ;阶码加一
ORL A,R6 ;尾数为零否?
JNZ EQ1 ;尾数不为零,继续对阶
MOV A,R2 ;尾数为零,提前结束对阶
MOV R5,A
SJMP AS4
```

EQ3: MOV A,R2 ;判断第一操作数阶符

JNB ACC.7,EQ2 ;如为正,右规第二操作数

EQ4: CLR C

```
LCALL RR1 ;第一操作数右规一次
ORL A,R3 ;尾数为零否?
JNZ EQ1 ;不为零,继续对阶
MOV A,R5 ;尾数为零,提前结束对阶
MOV R2,A
```

AS4: JB F0,AS5 ;尾数加减判断

```
MOV A,R4 ;尾数相加
ADD A,R7
MOV R4,A
MOV A,R3
ADDC A,R6
MOV R3,A
JNC AS2
```

LJMP RR1 ;有进位,右规一次

AS5: CLR C ;比较绝对值大小

```

MOV A,R4
SUBB A,R7
MOV B,A
MOV A,R3
SUBB A,R6
JC AS6
MOV R4,B ; 第一尾数减第二尾数
MOV R3,A
LJMP RLN ; 结果规格化
AS6: CPL 1FH ; 结果的符号与第一操作数相反
CLR C ; 结果的绝对值为第二尾数减第一尾数
MOV A,R7
SUBB A,R4
MOV R4,A
MOV A,R6
SUBB A,R3
MOV R3,A
RLN: MOV A,R3 ; 浮点数规格化
ORL A,R4 ; 尾数为零否?
JNZ RLN1
MOV R2,#0C1H ; 阶码取最小值
RET
RLN1: MOV A,R3
JB ACC.7,RLN2 ; 尾数最高位为一否?
CLR C ; 不为一,左规一次
LCALL RL1
SJMP RLN ; 继续判断
RLN2: CLR OV ; 规格化结束
RET
RL1: MOV A,R4 ; 第一操作数左规一次
RLC A ; 尾数扩大一倍
MOV R4,A
MOV A,R3
RLC A
MOV R3,A
DEC R2 ; 阶码减一
CJNE R2,#0C0H,RL1E ; 阶码下溢否?
CLR A
MOV R3,A ; 阶码下溢,操作数以零计
MOV R4,A
MOV R2,#0C1H
RL1E: CLR OV
RET
RR1: MOV A,R3 ; 第一操作数右规一次
RRC A ; 尾数缩小一半
MOV R3,A
MOV A,R4
RRC A

```

```

MOV R4,A
INC R2 ;阶码加一
CLR OV ;清溢出标志
CJNE R2,#40H,RR1E ;阶码上溢否?
MOV R2,#3FH ;阶码溢出
SETB OV
RR1E: RET

```

#### (4) 标号: FMUL 功能: 浮点数乘法

入口条件: 被乘数在[R0]中, 乘数在[R1]中。

出口信息: OV=0 时, 积仍在[R0]中, OV=1 时, 溢出。

影响资源: PSW、A、B、R2~R7、位 1EH、1FH 堆栈需求: 6 字节

FMUL: LCALL MVR0 ;将[R0]传送到第一工作区中

```

MOV A,@R0
XRL A,@R1 ;比较两个操作数的符号
RLC A
MOV 1FH,C ;保存积的符号
LCALL MUL0 ;计算积的绝对值
LJMP MOV0 ;将结果传回到[R0]中

```

MUL0: LCALL MVR1 ;将[R1]传送到第二工作区中

MUL1: MOV A,R3 ;第一尾数为零否?

```

ORL A,R4
JZ MUL6
MOV A,R6 ;第二尾数为零否?
ORL A,R7
JZ MUL5
MOV A,R7 ;计算  $R3R4 \times R6R7 - R3R4$ 
MOV B,R4
MULAB
MOV A,B
XCH A,R7
MOV B,R3
MULAB
ADD A,R7
MOV R7,A
CLR A
ADDC A,B
XCH A,R4
MOV B,R6
MULAB
ADD A,R7
MOV R7,A
MOV A,B
ADDC A,R4
MOV R4,A
CLR A
RLC A

```



```

XCH A,R3
MOV B,R6
MUL AB
ADD A,R4
MOV R4,A
MOV A,B
ADDC A,R3
MOV R3,A
JB ACC.7,MUL2 ; 积为规格化数否?
MOV A,R7 ; 左规一次
RLC A
MOV R7,A
LCALL RL1
MUL2: MOV A,R7
JNB ACC.7,MUL3
INC R4
MOV A,R4
JNZ MUL3
INC R3
MOV A,R3
JNZ MUL3
MOV R3,#80H
INC R2
MUL3: MOV A,R2 ; 求积的阶码
      ADD A,R5
MD: MOV R2,A ; 阶码溢出判断
     JB ACC.7,MUL4
     JNB ACC.6,MUL6
     MOV R2,#3FH ; 阶码上溢, 设立标志
     SETB OV
     RET
MUL4: JB ACC.6,MUL6
MUL5: CLRA ; 结果清零 (因子为零或阶码下溢)
      MOV R3,A
      MOV R4,A
      MOV R2,#41H
MUL6: CLR OV
      RET

```

( 5 ) 标号: F D I V 功能: 浮点数除法

入口条件: 被除数在[R0]中, 除数在[R1]中。

出口信息: OV=0 时, 商仍在[R0]中, OV=1 时, 溢出。

影响资源: PSW、A、B、R2~R7、位 1EH、1FH 堆栈需求: 5 字节

```

FDIV: INC R0
      MOV A,@R0
      INC R0
      ORLA,@R0

```

```

    DEC R0
    DEC R0
    JNZ DIV1
    MOV @R0,#41H ; 被除数为零, 不必运算
    CLR OV
    RET
DIV1: INC R1
    MOV A,@R1
    INC R1
    ORL A,@R1
    DEC R1
    DEC R1
    JNZ DIV2
    SETB OV ; 除数为零, 溢出
    RET
DIV2: LCALL MVR0 ; 将[R0]传送到第一工作区中
    MOV A,@R0
    XRL A,@R1 ; 比较两个操作数的符号
    RLC A
    MOV 1FH,C ; 保存结果的符号
    LCALL MVR1 ; 将[R1]传送到第二工作区中
    LCALL DIV3 ; 调用工作区浮点除法
    LJMP MOV0 ; 回传结果
DIV3: CLR C ; 比较尾数的大小
    MOV A,R4
    SUBB A,R7
    MOV A,R3
    SUBB A,R6
    JC DIV4
    LCALL RR1 ; 被除数右规一次
    SJMP DIV3
DIV4: CLR A ; 借用 R0R1R2 作工作寄存器
    XCH A,R0 ; 清零并保护之
    PUSH ACC
    CLR A
    XCH A,R1
    PUSH ACC
    MOV A,R2
    PUSH ACC
    MOV B,#10H ; 除法运算, R3R4 / R6R7 - R0R1
DIV5: CLR C
    MOV A,R1
    RLC A
    MOV R1,A
    MOV A,R0
    RLC A
    MOV R0,A
    MOV A,R4

```

```

RLC A
MOV R4,A
XCH A,R3
RLC A
XCH A,R3
MOV F0,C
CLR C
SUBB A,R7
MOV R2,A
MOV A,R3
SUBB A,R6
ANL C,/F0
JC DIV6
MOV R3,A
MOV A,R2
MOV R4,A
INC R1
DIV6: DJNZ B,DIV5
MOV A,R6 ; 四舍五入
CLR C
RRC A
SUBB A,R3
CLR A
ADDC A,R1 ; 将结果存回 R3R4
MOV R4,A
CLR A
ADDC A,R0
MOV R3,A
POP ACC ; 恢复 R0R1R2
MOV R2,A
POP ACC
MOV R1,A
POP ACC
MOV R0,A
MOV A,R2 ; 计算商的阶码
CLR C
SUBB A,R5
LCALL MD ; 阶码检验
LJMP RLN ; 规格化

```

( 6 ) 标号： F C L R 功能：浮点数清零

入口条件：操作数在[R0]中。

出口信息：操作数被清零。

影响资源：A 堆栈需求： 2 字节

```

FCLR: INC R0
      INC R0
      CLR A

```

```

MOV @R0,A
DEC R0
MOV @R0,A
DEC R0
MOV @R0,#41H
RET

```

( 7 ) 标号： F Z E R 功能：浮点数判零

入口条件：操作数在[R0]中。

出口信息：若累加器 A 为零，则操作数[R0]为零，否则不为零。

影响资源：A 堆栈需求： 2 字节

```

FZER: INC R0
      INC R0
      MOVA,@R0
      DEC R0
      ORLA,@R0
      DEC R0
      JNZ ZERO
      MOV @R0,#41H
ZERO: RET

```

( 8 ) 标号： F M O V 功能：浮点数传送

入口条件：源操作数在[R1]中，目标地址为[R0]。

出口信息：[R0]=[R1]，[R1]不变。

影响资源：A 堆栈需求： 2 字节

```

FMOV: INC R0
      INC R0
      INC R1
      INC R1
      MOVA,@R1
      MOV @R0,A
      DEC R0
      DEC R1
      MOVA,@R1
      MOV @R0,A
      DEC R0
      DEC R1
      MOVA,@R1
      MOV @R0,A
      RET

```

( 9 ) 标号： F P U S 功能：浮点数压栈

入口条件：操作数在[R0]中。

出口信息：操作数压入栈顶。

影响资源：A、R2、R3 堆栈需求： 5 字节

FPUS: POP ACC ; 将返回地址保存在 R2R3 中

```
MOV R2,A
POP ACC
MOV R3,A
MOV A,@R0 ; 将操作数压入堆栈
PUSH ACC
INC R0
MOV A,@R0
PUSH ACC
INC R0
MOV A,@R0
PUSH ACC
DEC R0
DEC R0
MOV A,R3 ; 将返回地址压入堆栈
PUSH ACC
MOV A,R2
PUSH ACC
RET ; 返回主程序
```

( 1 0 ) 标号: F P O P 功能: 浮点数出栈

入口条件: 操作数处于栈顶。

出口信息: 操作数弹至[R0]中。

影响资源: A、R2、R3 堆栈需求: 2 字节

FPOP: POP ACC ; 将返回地址保存在 R2R3 中

```
MOV R2,A
POP ACC
MOV R3,A
INC R0
INC R0
POP ACC ; 将操作数弹出堆栈, 传送到[R0]中
MOV @R0,A
DEC R0
POP ACC
MOV @R0,A
DEC R0
POP ACC
MOV @R0,A
MOV A,R3 ; 将返回地址压入堆栈
PUSH ACC
MOV A,R2
PUSH ACC
RET ; 返回主程序
```

( 1 1 ) 标号: F C M P 功能: 浮点数代数值比较 ( 不影响待比较操作数 )

入口条件: 待比较操作数分别在[R0]和[R1]中。

出口信息：若 CY=1，则[R0]<[R1]，若 CY=0 且 A=0 则 [R0]=[R1]，否则[R0]>[R1]。

影响资源：A、B、PSW 堆栈需求：2 字节

FCMP: MOV A,@R0 ; 数符比较

XRL A,@R1

JNB ACC.7,CMP2

MOV A,@R0 ; 两数异号，以[R0]数符为准

RLC A

MOV A,#0FFH

RET

CMP2: MOV A,@R1 ; 两数同号，准备比较阶码

MOV C,ACC.6

MOV ACC.7,C

MOV B,A

MOV A,@R0

MOV C,ACC.7

MOV F0,C ; 保存[R0]的数符

MOV C,ACC.6

MOV ACC.7,C

CLR C ; 比较阶码

SUBB A,B

JZ CMP6

RLC A ; 取阶码之差的符号

JNB F0,CMP5

CPL C ; [R0]为负时，结果取反

CMP5: MOV A,#0FFH ; 两数不相等

RET

CMP6: INC R0 ; 阶码相同时，准备比较尾数

INC R0

INC R1

INC R1

CLR C

MOV A,@R0

SUBB A,@R1

MOV B,A ; 保存部分差

DEC R0

DEC R1

MOV A,@R0

SUBB A,@R1

DEC R0

DEC R1

ORL A,B ; 生成是否相等信息

JZ CMP7

JNB F0,CMP7

CPL C ; [R0]为负时，结果取反

CMP7: RET

( 1 2 ) 标号：F A B S 功能：浮点绝对值函数

入口条件：操作数在[R0]中。

出口信息：结果仍在[R0]中。

影响资源：A 堆栈需求：2 字节

FABS: MOV A,@R0 ;读取操作数的阶码

CLR ACC.7 ;清除数符

MOV @R0,A ;回传阶码

RET

( 1 3 ) 标号： F S G N 功能：浮点符号函数

入口条件：操作数在[R0]中。

出口信息：累加器 A=1 时为正数，A=0FFH 时为负数，A=0 时为零。

影响资源：PSW、A 堆栈需求：2 字节

FSGN: INC R0 ;读尾数

MOV A,@R0

INC R0

ORL A,@R0

DEC R0

DEC R0

JNZ SGN

RET ;尾数为零，结束

SGN: MOV A,@R0 ;读取操作数的阶码

RLCA ;取数符

MOV A,#1 ;按正数初始化

JNC SGN1 ;是正数，结束

MOV A,#0FFH ;是负数，改变标志

SGN1: RET

( 1 4 ) 标号： F I N T 功能：浮点取整函数

入口条件：操作数在[R0]中。

出口信息：结果仍在[R0]中。

影响资源：PSW、A、R2、R3、R4、位 1FH 堆栈需求：6 字节

FINT: LCALL MVR0 ;将[R0]传送到第一工作区中

LCALL INT ;在工作寄存器中完成取整运算

LJMP MOV0 ;将结果传回到[R0]中

INT: MOV A,R3

ORL A,R4

JNZ INTA

CLR 1FH ;尾数为零，阶码也清零，结束取整

MOV R2,#41H

RET

INTA: MOV A,R2

JZ INTB ;阶码为零否？

JB ACC.7,INTB ;阶符为负否？

CLR C

SUBB A,#10H ;阶码小于 16 否？

JC INTD

```

    RET ;阶码大于16,已经是整数
INTB: CLR A ;绝对值小于一,取整后正数为零,负数为负一
    MOV R4,A
    MOV C,1FH
    RRC A
    MOV R3,A
    RLA
    MOV R2,A
    JNZ INTC
    MOV R2,#41H
INTC: RET
INTD: CLR F0 ;舍尾标志初始化
INTE: CLR C
    LCALL RR1 ;右规一次
    ORL C,F0 ;记忆舍尾情况
    MOV F0,C
    CJNE R2,#10H,INTE ;阶码达到16(尾数完全为整数)否?
    JNB F0,INTF ;舍去部分为零否?
    JNB 1FH,INTF ;操作数为正数否?
    INC R4 ;对于带小数的负数,向下取整
    MOV A,R4
    JNZ INTF
    INC R3
INTF: LJMP RLN ;将结果规格化

```

(15) 标号: FRCP 功能: 浮点倒数函数

入口条件: 操作数在[R0]中。

出口信息: OV=0时,结果仍在[R0]中,OV=1时,溢出。

影响资源: PSW、A、B、R2~R7、位1EH、1FH 堆栈需求: 5字节

```

FRCP: MOV A,@R0
    MOV C,ACC.7
    MOV 1FH,C ;保存数符
    MOV C,ACC.6 ;绝对值传送到第二工作区
    MOV ACC.7,C
    MOV R5,A
    INC R0
    MOV A,@R0
    MOV R6,A
    INC R0
    MOV A,@R0
    MOV R7,A
    DEC R0
    DEC R0
    ORL A,R6
    JNZ RCP
    SETB OV ;零不能求倒数,设立溢出标志
    RET

```



```

RCP: MOV A,R6
      JB ACC.7,RCP2 ;操作数格式化否?
      CLR C ;格式化之
      MOV A,R7
      RLC A
      MOV R7,A
      MOV A,R6
      RLC A
      MOV R6,A
      DEC R5
      SJMP RCP
RCP2: MOV R2,#1 ;将数值1.00传送到第一工作区
      MOV R3,#80H
      MOV R4,#0
      LCALL DIV3 ;调用工作区浮点除法,求得倒数
      LJMP MOV0 ;回传结果

```

(16) 标号: FSQU 功能: 浮点数平方

入口条件: 操作数在[R0]中。  
 出口信息: OV=0 时, 平方值仍然在[R0]中, OV=1 时溢出。  
 影响资源: PSW、A、B、R2~R7、位1EH、1FH 堆栈需求: 9 字节

```

FSQU: MOV A,R0 ;将操作数
      XCH A,R1 ;同时作为乘数
      PUSH ACC ;保存 R 1 指针
      LCALL FMUL ;进行乘法运算
      POP ACC
      MOV R1,A ;恢复 R 1 指针
      RET

```

(17) 标号: FSQR 功能: 浮点数开平方(快速逼近算法)

入口条件: 操作数在[R0]中。  
 出口信息: OV=0 时, 平方根仍在[R0]中, OV=1 时, 负数开平方出错。  
 影响资源: PSW、A、B、R2~R7 堆栈需求: 2 字节

```

FSQR: MOV A,@R0
      JNB ACC.7,SQR
      SETB OV ;负数开平方,出错
      RET
SQR: INC R0
      INC R0
      MOV A,@R0
      DEC R0
      ORLA,@R0
      DEC R0
      JNZ SQ
      MOV @R0,#41H ;尾数为零,不必运算
      CLR OV

```

```

    RET
SQ: MOV A,@R0
    MOV C,ACC.6 ;将阶码扩展成 8 b i t 补码
    MOV ACC.7,C
    INC A ;加一
    CLR C
    RRC A ;除二
    MOV @R0,A ;得到平方根的阶码,回存之
    INC R0 ;指向被开方数尾数的高字节
    JC SQR0 ;原被开方数的阶码是奇数吗?
    MOV A,@R0 ;是奇数,尾数右规一次
    RRC A
    MOV @R0,A
    INC R0
    MOV A,@R0
    RRC A
    MOV @R0,A
    DEC R0
SQR0: MOV A,@R0
    JZ SQR9 ;尾数为零,不必运算
    MOV R2,A ;将尾数传送到 R2R3 中
    INC R0
    MOV A,@R0
    MOV R3,A
    MOV A,R2 ;快速开方,参阅定点子程序说明
    ADD A,#57H
    JC SQR2
    ADD A,#45H
    JC SQR1
    ADD A,#24H
    MOV B,#0E3H
    MOV R4,#80H
    SJMP SQR3
SQR1: MOV B,#0B2H
    MOV R4,#0A0H
    SJMP SQR3
SQR2: MOV B,#8DH
    MOV R4,#0D0H
SQR3: MUL AB
    MOV A,B
    ADD A,R4
    MOV R4,A
    MOV B,A
    MUL AB
    XCH A,R3
    CLR C
    SUBB A,R3
    MOV R3,A

```

```

MOV A,B
XCH A,R2
SUBB A,R2
MOV R2,A
SQR4: SETB C
MOV A,R4
RLC A
MOV R6,A
CLR A
RLC A
MOV R5,A
MOV A,R3
SUBB A,R6
MOV B,A
MOV A,R2
SUBB A,R5
JC SQR5
INC R4
MOV R2,A
MOV R3,B
SJMP SQR4
SQR5: MOV A,R4
XCH A,R2
RRC A
MOV F0,C
MOV A,R3
MOV R5,A
MOV R4,#8
SQR6: CLR C
MOV A,R3
RLC A
MOV R3,A
CLR C
MOV A,R5
SUBB A,R2
JB F0,SQR7
JC SQR8
SQR7: MOV R5,A
INC R3
SQR8: CLR C
MOV A,R5
RLC A
MOV R5,A
MOV F0,C
DJNZ R4,SQR6
MOV A,R3 ; 将平方根的尾数回传到[R0]中
MOV @R0,A
DEC R0

```

```

MOV A,R2
MOV @R0,A
SQR9: DEC R0 ; 数据指针回归原位
      CLR OV ; 开方结果有效
      RET

```

( 1 8 ) 标号： F P L N 功能：浮点数多项式计算

入口条件：自变量在[R0]中，多项式系数在调用指令之后，以 4 0 H 结束。

出口信息：OV=0 时，结果仍在[R0]中，OV=1 时，溢出。

影响资源：DPTR、PSW、A、B、R2~R7、位 1EH、1FH 堆栈需求： 4 字节

FPLN: POP DPH ; 取出多项式系数存放地址

```

      POP DPL
      XCH A,R0 ; R0、R1 交换角色，自变量在[R1]中
      XCH A,R1
      XCH A,R0
      CLR A ; 清第一工作区
      MOV R2,A
      MOV R3,A
      MOV R4,A
      CLR 1FH

```

PLN1: CLR A ; 读取一个系数，并装入第二工作区

```

      MOVC A,@A+DPTR
      MOV C,ACC.7
      MOV 1EH,C
      MOV C,ACC.6
      MOV ACC.7,C
      MOV R5,A
      INC DPTR
      CLR A
      MOVC A,@A+DPTR
      MOV R6,A
      INC DPTR
      CLR A
      MOVC A,@A+DPTR
      MOV R7,A
      INC DPTR ; 指向下一个系数
      MOV C,1EH ; 比较两个数符
      RRC A
      XRL A,23H
      RLC A
      MOV F0,C ; 保存比较结果
      LCALL AS1 ; 进行代数加法运算
      CLR A ; 读取下一个系数的第一个字节
      MOVC A,@A+DPTR
      CJNE A,#40H,PLN2 ; 是结束标志吗？
      XCH A,R0 ; 运算结束，恢复 R0、R1 原来的角色
      XCH A,R1

```

```

XCH A,R0
LCALL MOV0 ;将结果回传到[R0]中
CLR A
INC DPTR
JMP @A+DPTR ;返回主程序
PLN2: MOV A,@R1 ;比较自变量和中间结果的符号
XRL A,23H
RLC A
MOV 1FH,C ;保存比较结果
LCALL MUL0 ;进行乘法运算
SJMP PLN1 ;继续下一项运算

```

( 1 9 ) 标号： F L O G 功能：以 1 0 为底的浮点对数函数

入口条件：操作数在[R0]中。  
 出口信息：OV=0 时，结果仍在[R0]中，OV=1 时，负数或零求对数出错。  
 影响资源：DPTR、PSW、A、B、R2~R7、位 1EH、1FH 堆栈需求：9 字节  
 FLOG: LCALL FLN ;先以 e 为底求对数  
       JNB OV,LOG  
       RET ;如溢出则停止计算  
 LOG: MOV R5,#0FFH;系数 0 . 4 3 4 3 0 ( 1 / L n 1 0 )  
       MOV R6,#0DEH  
       MOV R7,#5CH  
       LCALL MUL1 ;通过相乘来换底  
       LJMP MOV0 ;传回结果

( 2 0 ) 标号： F L N 功能：以 e 为底的浮点对数函数

入口条件：操作数在[R0]中。  
 出口信息：OV=0 时，结果仍在[R0]中，OV=1 时，负数或零求对数出错。  
 影响资源：DPTR、PSW、A、B、R2~R7、位 1EH、1FH 堆栈需求：7 字节  
 FLN: LCALL MVR0 ;将[R0]传送到第一工作区  
       JB 1FH,LNOV;负数或零求对数，出错  
       MOV A,R3  
       ORL A,R4  
       JNZ LN0  
       LNOV: SETB OV  
       RET  
 LN0: CLR C  
       LCALL RL1 ;左规一次  
       CLR A  
       XCH A,R2 ;保存原阶码，清零工作区的阶码  
       PUSH ACC  
       LCALL RLN ;规格化  
       LCALL MOV0 ;回传  
       LCALL FPLN ;用多项式计算尾数的对数  
 DB 7BH,0F4H,30H; 0 . 0 2 9 8 0 8

```

DB 0FEH,85H,13H ; - 0 . 1 2 9 9 6
DB 7FH,91H,51H ; 0 . 2 8 3 8 2
DB 0FFH,0FAH,0BAH ; - 0 . 4 8 9 7
DB 0,0FFH,0CAH ; 0 . 9 9 9 1 8
DB 70H,0C0H,0 ; 1 . 1 4 4 2 × 1 0-5
DB 40H ; 结束
    POP ACC ; 取出原阶码
    JNZ LN1
    RET ; 如为零, 则结束
LN1: CLR 1EH ; 清第二区数符
    MOV C,ACC.7
    MOV F0,C ; 保存阶符
    JNC LN2
    CPL A ; 当阶码为负时, 求其绝对值
    INC A
LN2: MOV R2,A ; 阶码的绝对值乘以 0 . 6 9 3 1 5
    MOV B,#72H
    MUL AB
    XCH A,R2
    MOV R7,B
    MOV B,#0B1H
    MUL AB
    ADD A,R7
    MOV R7,A ; 乘积的尾数在 R6R7R2 中
    CLR A
    ADDC A,B
    MOV R6,A
    MOV R5,#8 ; 乘积的阶码初始化 ( 整数部分为一字节 )
LN3: JB ACC.7,LN4 ; 乘积格式化
    MOV A,R2
    RLC A
    MOV R2,A
    MOV A,R7
    RLC A
    MOV R7,A
    MOV A,R6
    RLC A
    MOV R6,A
    DEC R5
    SJMP LN3
LN4: MOV C,F0 ; 取出阶符, 作为乘积的数符
    MOV ACC.7,C
    LJMP ASN ; 与尾数的对数合并, 得原操作数的对数

```

( 2 1 ) 标号: F E 1 0 功能: 以 1 0 为底的浮点指数函数

入口条件: 操作数在[R0]中。

出口信息: OV=0 时, 结果仍在[R0]中, OV=1 时, 溢出。

影响资源：DPTR、PSW、A、B、R2~R7、位1EH、1FH 堆栈需求：6字节

FE10: MOV R5,#2 ;加权系数为3 . 3 2 1 9 ( L o g 2 1 0 )

MOV R6,#0D4H

MOV R7,#9AH

SJMPEXP ;先进行加权运算,后以2为底统一求幂

( 2 2 ) 标号： F E X P 功能：以e为底的浮点指数函数

入口条件：操作数在[R0]中。

出口信息：OV=0时,结果仍在[R0]中,OV=1时,溢出。

影响资源：DPTR、PSW、A、B、R2~R7、位1EH、1FH 堆栈需求：6字节

FEXP: MOV R5,#1 ;加权系数为1 . 4 4 2 7 2 ( L n g 2 e )

MOV R6,#0B8H

MOV R7,#0ABH

EXP: CLR 1EH ;加权系数为正数

LCALL MVR0 ;将[R0]传送到第一工作区

LCALL MUL1 ;进行加权运算

SJMPE20 ;以2为底统一求幂

( 2 3 ) 标号： F E 2 功能：以2为底的浮点指数函数

入口条件：操作数在[R0]中。

出口信息：OV=0时,结果仍在[R0]中,OV=1时,溢出。

影响资源：DPTR、PSW、A、B、R2~R7、位1EH、1FH 堆栈需求：6字节

FE2: LCALL MVR0 ;将[R0]传送到第一工作区

E20: MOV A,R3

ORLA,R4

JZ EXP1 ;尾数为零

MOV A,R2

JB ACC.7,EXP2 ;阶符为负?

SETB C

SUBB A,#6 ;阶码大于6否?

JC EXP2

JB 1FH,EXP0 ;数符为负否?

MOV @R0,#3FH ;正指数过大,幂溢出

INC R0

MOV @R0,#0FFH

INC R0

MOV @R0,#0FFH

DEC R0

DEC R0

SETB OV

RET

EXP0: MOV @R0,#41H ;负指数过大,幂下溢,清零处理

CLR A

INC R0

MOV @R0,A

INC R0

```

MOV @R0,A
DEC R0
DEC R0
CLR OV
RET
EXP1: MOV @R0,#1 ;指数为零, 幂为 1 . 0 0
      INC R0
      MOV @R0,#80H
      INC R0
      MOV @R0,#0
      DEC R0
      DEC R0
      CLR OV
      RET
EXP2: MOV A,R2 ;将指数复制到第二工作区
      MOV R5,A
      MOV A,R3
      MOV R6,A
      MOV A,R4
      MOV R7,A
      MOV C,1FH
      MOV 1EH,C
      LCALL INT ;对第一区取整
      MOV A,R3
      JZ EXP4
EXP3: CLR C ;使尾数高字节 R3 对应一个字节整数
      RRC A
      INC R2
      CJNE R2,#8,EXP3
EXP4: MOV R3,A
      JNB 1FH,EXP5
      CPL A ;并用补码表示
      INC A
EXP5: PUSH ACC ;暂时保存之
      LCALL RLN ;重新规格化
      CPL 1FH
      SETB F0
      LCALL AS1 ;求指数的小数部分
      LCALL MOV0 ;回传指数的小数部分
      LCALL FPLN ;通过多项式计算指数的小数部分的幂
      DB 77H,0B1H,0C9H ; 1 . 3 5 6 4 × 1 0-3
      DB 7AH,0A1H,68H ; 9 . 8 5 1 4 × 1 0-3
      DB 7CH,0E3H,4FH ; 0 . 0 5 5 4 9 5
      DB 7EH,0F5H,0E7H ; 0 . 2 4 0 1 4
      DB 0,0B1H,72H ; 0 . 6 9 3 1 5
      DB 1,80H,0 ; 1 . 0 0 0 0 0
      DB 40H ;结束
      POP ACC ;取出指数的整数部分

```



```
ADD A,R2 ;按补码加到幂的阶码上
MOV R2,A
CLR 1FH ;幂的符号为正
LJMP MOV0 ;将幂传回[R0]中
```

( 2 4 ) 标号： D T O F 功能：双字节十六进制定点数转换成格式化浮点数

入口条件：双字节定点数的绝对值在[R0]中，数符在位 1FH 中，整数部分的位数在 A 中。

出口信息：转换成格式化浮点数在[R0]中（三字节）

影响资源：PSW、A、R2、R3、R4、位 1FH 堆栈需求：6 字节

DTOF: MOV R2,A ;按整数的位数初始化阶码

```
MOV A,@R0 ;将定点数作尾数
MOV R3,A
INC R0
MOV A,@R0
MOV R4,A
DEC R0
LCALL RLN ;进行规格化
LJMP MOV0 ;传送结果到[R0]中
```

( 2 5 ) 标号： F T O D 功能：格式化浮点数转换成双字节定点数

入口条件：格式化浮点操作数在[R0]中。

出口信息：OV=1 时溢出，OV=0 时转换成功：定点数的绝对值在[R0]中（双字节），数符在位 1FH 中，F0=1 时为整数，CY=1 时为一字节整数一字节小数，否则为纯小数。

影响资源：PSW、A、B、R2、R3、R4、位 1FH 堆栈需求：6 字节

FTOD: LCALL MVR0 ;将[R0]传送到第一工作区

```
MOV A,R2
JZ FTD4 ;阶码为零，纯小数
JB ACC.7,FTD4 ;阶码为负，纯小数
SETB C
SUBB A,#10H
JC FTD1
SETB OV ;阶码大于 16，溢出
RET
```

FTD1: SETB C

```
MOV A,R2
SUBB A,#8 ;阶码大于 8 否？
JC FTD3
```

FTD2: MOV B,#10H ;阶码大于 8，按双字节整数转换

```
LCALL FTD8
SETB F0 ;设立双字节整数标志
CLR C
CLR OV
RET
```

FTD3: MOV B,#8 ;按一字节整数一字节小数转换

```
LCALL FTD8
SETB C ;设立一字节整数一字节小数标志
```

```

    CLR F0
    CLR OV
    RET
FTD4: MOV B,#0 ;按纯小数转换
    LCALL FTD8
    CLR OV ;设立纯小数标志
    CLR F0
    CLR C
    RET
FTD8: MOV A,R2 ;按规定的整数位数进行右规
    CJNE A,B,FTD9
    MOV A,R3 ;将双字节结果传送到[R0]中
    MOV @R0,A
    INC R0
    MOV A,R4
    MOV @R0,A
    DEC R0
    RET
FTD9: CLR C
    LCALL RR1 ;右规一次
    SJMP FTD8

```

( 2 6 ) 标号： B T O F 功能：浮点 B C D 码转换成格式化浮点数

入口条件：浮点 B C D 码操作数在[R0]中。

出口信息：转换成的格式化浮点数仍在[R0]中。

影响资源：PSW、A、B、R2~R7、位 1DH~1FH 堆栈需求：6 字节

BTOF: INC R0 ;判断是否为零。

```

    INC R0
    MOV A,@R0
    MOV R7,A
    DEC R0
    MOV A,@R0
    MOV R6,A
    DEC R0
    ORL A,R7
    JNZ BTF0
    MOV @R0,#41H ;为零,转换结束。
    RET

```

```

BTF0: MOV A,@R0
    MOV C,ACC.7
    MOV 1DH,C ;保存数符。
    CLR 1FH ;以绝对值进行转换。
    MOV C,ACC.6 ;扩充阶码为八位。
    MOV ACC.7,C
    MOV @R0,A
    JNC BTF1
    ADD A,#19 ;是否小于 1 E - 1 9 ?

```

JC BTF2

MOV @R0,#41H ; 小于 1 E - 1 9 时以 0 计。

INC R0

MOV @R0,#0

INC R0

MOV @R0,#0

DEC R0

DEC R0

RET

BTF1: SUBB A,#19

JC BTF2

MOV A,#3FH ; 大于 1 E 1 9 时封顶。

MOV C,1DH

MOV ACC.7,C

MOV @R0,A

INC R0

MOV @R0,#0FFH

INC R0

MOV @R0,#0FFH

DEC R0

DEC R0

RET

BTF2: CLR A ; 准备将 B C D 码尾数转换成十六进制浮点数。

MOV R4,A

MOV R3,A

MOV R2,#10H ; 至少两个字节。

BTF3: MOV A,R7

ADD A,R7

DAA

MOV R7,A

MOV A,R6

ADDC A,R6

DAA

MOV R6,A

MOV A,R4

RLC A

MOV R4,A

MOV A,R3

RLC A

MOV R3,A

DEC R2

JNB ACC.7,BTF3 ; 直到尾数规格化。

MOV A,R6 ; 四舍五入。

ADD A,#0B0H

CLR A

ADDC A,R4

MOV R4,A

CLR A

```

    ADDC A,R3
    MOV R3,A
    JNC BTF4
    MOV R3,#80H
    INC R2

```

BTF4: MOV DPTR,#BTFL ; 准备查表得到十进制阶码对应的浮点数。

```

    MOV A,@R0
    ADD A,#19 ; 计算表格偏移量。
    MOV B,#3
    MUL AB
    ADD A,DPL
    MOV DPL,A
    JNC BTF5
    INC DPH

```

BTF5: CLR A ; 查表。

```

    MOVC A,@A+DPTR
    MOV C,ACC.6
    MOV ACC.7,C
    MOV R5,A
    MOV A,#1
    MOVC A,@A+DPTR
    MOV R6,A
    MOV A,#2
    MOVC A,@A+DPTR
    MOV R7,A
    LCALL MUL1 ; 将阶码对应的浮点数和尾数对应的浮点数相乘。
    MOV C,1DH ; 取出数符。
    MOV 1FH,C
    LJMP MOV0 ; 传送转换结果。

```

( 2 7 ) 标号： F T O B 功能：格式化浮点数转换成浮点 B C D 码

入口条件：格式化浮点操作数在[R0]中。

出口信息：转换成的浮点 B C D 码仍在[R0]中。

影响资源：PSW、A、B、R2~R7、位 1DH~1FH 堆栈需求：6 字节

```

FTOB: INC R0
      MOV A,@R0
      INC R0
      ORL A,@R0
      DEC R0
      DEC R0
      JNZ FTB0
      MOV @R0,#41H
      RET

```

```

FTB0: MOV A,@R0
      MOV C,ACC.7
      MOV 1DH,C
      CLR ACC.7

```

```

MOV @R0,A
LCALL MVR0
MOV DPTR,#BFL0 ; 绝对值大于或等于 1 时的查表起点。
MOV B,#0 ; 十的 0 次幂。
MOV A,R2
JNB ACC.7,FTB1
MOV DPTR,#BTFL ; 绝对值小于 1 E - 6 时的查表起点。
MOV B,#0EDH ; 十的 - 1 9 次幂。
ADD A,#16
JNC FTB1
MOV DPTR,#BFLN ; 绝对值大于或等于 1 E - 6 时的查表起点。
MOV B,#0FAH ; 十的 - 6 次幂。
FTB1: CLR A ; 查表, 找到一个比待转换浮点数大的整数幂。
      MOV C,@A+DPTR
      MOV C,ACC.6
      MOV ACC.7,C
      MOV R5,A
      MOV A,#1
      MOV C,@A+DPTR
      MOV R6,A
      MOV A,#2
      MOV C,@A+DPTR
      MOV R7,A
      MOV A,R5 ; 和待转换浮点数比较。
      CLR C
      SUBB A,R2
      JB ACC.7,FTB2 ; 差为负数。
      JNZ FTB3
      MOV A,R6
      CLR C
      SUBB A,R3
      JC FTB2
      JNZ FTB3
      MOV A,R7
      CLR C
      SUBB A,R4
      JC FTB2
      JNZ FTB3
      MOV R5,B ; 正好是表格中的数。
      INC R5 ; 幂加一。
      MOV R6,#10H ; 尾数为 0 . 1 0 0 0。
      MOV R7,#0
      SJMP FTB6 ; 传送转换结果。
FTB2: INC DPTR ; 准备表格下一项。
      INC DPTR
      INC DPTR
      INC B ; 幂加一。
      SJMP FTB1 ;

```

FTB3: PUSH B ; 保存幂值。  
LCALL DIV3 ; 相除, 得到一个二进制浮点数的纯小数。

FTB4: MOV A,R2 ; 取阶码。

JZ FTB5 ; 为零吗?

CLR C

LCALL RR1 ; 右规。

SJMP FTB4

FTB5: POP ACC ; 取出幂值。

MOV R5,A ; 作为十进制浮点数的阶码。

LCALL HB2 ; 转换尾数的十分位和百分位。

MOV R6,A

LCALL HB2 ; 转换尾数的千分位和万分位。

MOV R7,A

MOV A,R3 ; 四舍五入。

RLC A

CLR A

ADDC A,R7

DAA

MOV R7,A

CLR A

ADDC A,R6

DAA

MOV R6,A

JNC FTB6

MOV R6,#10H

INC R5

FTB6: INC R0 ; 存放转换结果。

INC R0

MOV A,R7

MOV @R0,A

DEC R0

MOV A,R6

MOV @R0,A

DEC R0

MOV A,R5

MOV C,1DH ; 取出数符。

MOV ACC.7,C

MOV @R0,A

RET

HB2: MOV A,R4 ; 尾数扩大 1 0 0 倍。

MOV B,#100

MUL AB

MOV R4,A

MOV A,B

XCH A,R3

MOV B,#100

MUL AB

ADD A,R3

```

MOV R3,A
JNC HB21
INC B
HB21: MOV A,B ;将整数部分转换成B C D码。
      MOV B,#10
      DIV AB
      SWAP A
      ORL A,B
      RET
BTFL: DB 41H,0ECH,1EH ; 1.0000E-19
      DB 45H,93H,93H ; 1.0000E-18
      DB 48H,0B8H,78H ; 1.0000E-17
      DB 4BH,0E6H,96H ; 1.0000E-16
      DB 4FH,90H,1DH ; 1.0000E-15
      DB 52H,0B4H,25H ; 1.0000E-14
      DB 55H,0E1H,2EH ; 1.0000E-13
      DB 59H,8CH,0BDH ; 1.0000E-12
      DB 5CH,0AFH,0ECH ; 1.0000E-11
      DB 5FH,0DBH,0E7H ; 1.0000E-10
      DB 63H,89H,70H ; 1.0000E-9
      DB 66H,0ABH,0CCH ; 1.0000E-8
      DB 69H,0D6H,0C0H ; 1.0000E-7
BFLN: DB 6DH,86H,38H ; 1.0000E-6
      DB 70H,0A7H,0C6H ; 1.0000E-5
      DB 73H,0D1H,0B7H ; 1.0000E-4
      DB 77H,83H,12H ; 1.0000E-3
      DB 7AH,0A3H,0D7H ; 1.0000E-2
      DB 7DH,0CCH,0CDH ; 1.0000E-1
BFLO: DB 1,80H,00H ; 1.0000
      DB 4,0A0H,00H ; 1.0000E1
      DB 7,0C8H,00H ; 1.0000E2
      DB 0AH,0FAH,00H ; 1.0000E3
      DB 0EH,9CH,40H ; 1.0000E4
      DB 11H,0C3H,50H ; 1.0000E5
      DB 14H,0F4H,24H ; 1.0000E6
      DB 18H,98H,97H ; 1.0000E7
      DB 1BH,0BEH,0BCH ; 1.0000E8
      DB 1EH,0EEH,6BH ; 1.0000E9
      DB 22H,95H,03H ; 1.0000E10
      DB 25H,0BAH,44H ; 1.0000E11
      DB 28H,0E8H,0D5H ; 1.0000E12
      DB 2CH,91H,85H ; 1.0000E13
      DB 2FH,0B5H,0E6H ; 1.0000E14
      DB 32H,0E3H,60H ; 1.0000E15
      DB 36H,8EH,1CH ; 1.0000E16
      DB 39H,31H,0A3H ; 1.0000E17
      DB 3CH,0DEH,0BH ; 1.0000E18
      DB 40H,8AH,0C7H ; 1.0000E19

```

( 2 8 ) 标号： F C O S 功能：浮点余弦函数

入口条件：操作数在[R0]中。

出口信息：结果仍在[R0]中。

影响资源：DPTR、PSW、A、B、R2~R7、位1DH~1FH 堆栈需求：6字节

FCOS: LCALL FABS ;  $\cos(-X) = \cos X$

MOV R5,#1 ; 常数 1.5708 (  $\pi / 2$  )

MOV R6,#0C9H

MOV R7,#10H

CLR 1EH

LCALL MVR0

CLR F0

LCALL AS1 ;  $x + (\pi / 2)$

LCALL MOV0 ; 保存结果，接着运行下面的FSIN 程序

( 2 9 ) 标号： F S I N 功能：浮点正弦函数

入口条件：操作数在[R0]中。

出口信息：结果仍在[R0]中。

影响资源：DPTR、PSW、A、B、R2~R7、位1DH~1FH 堆栈需求：6字节

FSIN: MOV A,@R0

MOV C,ACC.7

MOV 1DH,C ; 保存自变量的符号

CLR ACC.7 ; 统一按正数计算

MOV @R0,A

LCALL MVR0 ; 将[R0]传送到第一工作区

MOV R5,#0 ; 系数 0.636627 (  $2 / \pi$  )

MOV R6,#0A2H

MOV R7,#0FAH

CLR 1EH

LCALL MUL1 ; 相乘，自变量按 (  $\pi / 2$  ) 规一化

MOV A,R2 ; 将结果复制到第二区

MOV R5,A

MOV A,R3

MOV R6,A

MOV A,R4

MOV R7,A

LCALL INT ; 第一区取整，获得象限信息

MOV A,R2

JZ SIN2

SIN1: CLR C ; 将浮点象限数转换成定点象限数

LCALL RR1

CJNE R2,#10H,SIN1

MOV A,R4

JNB ACC.1,SIN2

CPL 1DH ; 对于第三、四象限，结果取反

SIN2: JB ACC.0,SIN3



```

        CPL 1FH ; 对于第一、三象限，直接求规一化的小数
        SJMP SIN4
SIN3: MOV A,R4 ; 对于第二、四象限，准备求其补数
        INC A
        MOV R4,A
        JNZ SIN4
        INC R3
SIN4: LCALL RLN ; 规格化
        SETB F0
        LCALL AS1 ; 求自变量归一化等效值
        LCALL MOV0 ; 回传
        LCALL FPLN ; 用多项式计算正弦值
        DB 7DH,93H,28H ; 0 . 0 7 1 8 5
        DB 41H,0,0 ; 0
        DB 80H,0A4H,64H ; - 0 . 6 4 2 1 5
        DB 41H,0,0 ; 0
        DB 1,0C9H,2 ; 1 . 5 7 0 4
        DB 41H,0,0 ; 0
        DB 40H ; 结束
        MOV A,@R0 ; 结果的绝对值超过 1 . 0 0 吗？
        JZ SIN5
        JB ACC.6,SIN5
        INC R0 ; 绝对值按 1 . 0 0 封顶
        MOV @R0,#80H
        INC R0
        MOV @R0,#0
        DEC R0
        DEC R0
        MOV A,#1
SIN5: MOV C,1DH ; 将数符拼入结果中
        MOV ACC.7,C
        MOV @R0,A
        RET

```

( 3 0 ) 标号： F A T N 功能：浮点反正切函数

入口条件：操作数在[R0]中。

出口信息：结果仍在[R0]中。

影响资源：DPTR、PSW、A、B、R2~R7、位1CH~1FH 堆栈需求：7字节

```

FATN: MOV A,@R0
        MOV C,ACC.7
        MOV 1DH,C ; 保存自变量数符
        CLR ACC.7 ; 自变量取绝对值
        MOV @R0,A
        CLR 1CH ; 清求余运算标志
        JB ACC.6,ATN1 ; 自变量为纯小数否？
        JZ ATN1
        SETB 1CH ; 置位求余运算标志

```

```

    LCALL FRCP ;通过倒数运算,转换成纯小数
ATN1: LCALL FPLN ;通过多项式运算,计算反正切函数值
    DB 0FCH,0E4H,91H ; - 0 . 0 5 5 8 0 2
    DB 7FH,8FH,37H ; 0 . 2 7 9 2 2
    DB 0FFH,0EDH,0E0H ; - 0 . 4 6 4 6 0
    DB 7BH,0E8H,77H ; 0 . 0 2 8 3 7 7
    DB 0,0FFH,68H ; 0 . 9 9 7 7
    DB 72H,85H,0ECH ; 3 . 1 9 3 0 × 1 0-5
    DB 40H ;结束
    JNB 1CH,ATN2 ;需要求余运算否?
    CPL 1FH ;准备运算标志
    MOV C,1FH
    MOV F0,C ;常数 1 . 5 7 0 8 ( / 2 )
    MOV R5,#1
    MOV R6,#0C9H
    MOV R7,#10H
    LCALL AS1 ;求余运算
    LCALL MOV0 ;回传
ATN2: MOV A,@R0 ;拼入结果的数符
    MOV C,1DH
    MOV ACC,7,C
    MOV @R0,A
    RET

```

( 3 1 ) 标号: RTOD 功能: 浮点弧度数转换成浮点度数

入口条件: 浮点弧度数在[R0]中。  
 出口信息: 转换成的浮点度数仍在[R0]中。  
 影响资源: PSW、A、B、R2~R7、位 1EH、1FH 堆栈需求: 6 字节  
 RTOD: MOV R5,#6 ;系数 ( 1 8 0 / ) 传送到第二工作区  
 MOV R6,#0E5H  
 MOV R7,#2FH  
 SJMP DR ;通过乘法进行转换

( 3 2 ) 标号: DTOR 功能: 浮点度数转换成浮点弧度数

入口条件: 浮点度数在[R0]中。  
 出口信息: 转换成的浮点弧度数仍在[R0]中。  
 影响资源: PSW、A、B、R2~R7、位 1EH、1FH 堆栈需求: 6 字节  
 DTOR: MOV R5,#0FBH ;系数 ( / 1 8 0 ) 传送到第二工作区  
 MOV R6,#8EH  
 MOV R7,#0FAH  
 DR: LCALL MVR0 ;将[R0]传送到第一工作区  
 CLR 1EH ;系数为正  
 LCALL MUL1 ;通过乘法进行转换  
 LJMP MOV0 ;结果传送到[R0]中  
 END