

《???— 5 1 单片机实用子程序库 (9 6 年版) 》

周 航 慈

目前已有若干版本的子程序库公开发表, 它们各有特色。笔者在1988年也编制了两个子程序库(定点子程序库和浮点子程序库), 并在相容性、透明性、容错性和算法优化方面作了一些工作。本程序库中的开平方算法为笔者研究的快速逼近算法, 它能达到牛顿迭代法同样的精度, 而速度加快二十倍左右, 超过双字节定点除法的速度。经过八年来全国广大用户的实际使用, 反馈了不少信息, 陆续扩充了一些新的子程序, 纠正了一些隐含错误, 成为现在这个最新版本。

本子程序库对《单片机应用程序设计技术》一书附录中的子程序库作了重大修订:

(1) 按当前流行的以 IBM PC 为主机的开发系统对汇编语言的规定, 将原子程序库的标号和位地址进行了调整, 读者不必再进行修改, 便可直接使用。

(2) 对浮点运算符程序库进行了进一步的测试和优化, 对十进制浮点数和二进制浮点数的相互转换子程序进行了彻底改写, 提高了运算精度和可靠性。

(3) 新增添了若干个浮点子程序(传送、比较、清零、判零等), 使编写数据处理程序的工作变得更简单直观。

在使用说明中开列了最主要的几项: 标号、入口条件、出口信息、影响资源、堆栈需求, 各项的意义请参阅《单片机应用程序设计技术》第六章 6.3.7 节的内容。程序清单中开列了四个栏目: 标号、指令、操作数、注释。为方便读者理解, 注释尽力详细。

子程序库的使用方法如下:

1. 将子程序库全部内容链接在应用程序之后, 统一编译即可。优点是简单方便, 缺点是程序太长, 大量无关子程序也包含在其中。

2. 仅将子程序库中的有关部分内容链接在应用程序之后, 统一编译即可。有些子程序需要调用一些低级子程序, 这些低级子程序也应该包含在内。优点是程序紧凑, 缺点是需要对子程序库进行仔细删节。

(一) ???— 5 1 定点运算符程序库及其使用说明

定点运算符程序库文件名为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: ??? 功能: 多字节???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

DA A ; 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： ??? 功能：多字节???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???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： ??? 功能：多字节???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: ???? 功能: 多字节???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?br>

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: ???? 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
      MUL  AB
      MOV  R4,B  ; 暂存部分积
      MOV  R5,A
      MOV  A,R3  ; 计算R3乘R6
      MOV  B,R6
      MUL  AB

```

```

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
MUL  AB
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
MUL  AB
ADD  A,R3    ; 累加部分积
MOV  R3,A
MOV  A,R2
ADDC A,B
MOV  R2,A
RET

```

(6) 标号: ??? 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
      MUL  AB
      MOV  R4,B    ; 暂存部分积
      MOV  R5,A
      MOV  A,R2    ; 计算R2平方
      MOV  B,A
      MUL  AB
      XCH  A,R3    ; 暂存部分积, 并换出R2和R3
      XCH  A,B
      XCH  A,R2
      MUL  AB    ; 计算2×R2×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: ???? 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: ?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: ??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：???? 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: ???? 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: ??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 ?否?
     JNZ SQRH ; 不小于 4 0 ?, 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: ??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 ?否?
     JNZ SQRH ; 不小于 4 0 ?, 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: MUL  AB      ; 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
    MUL  AB        ; 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: ???? 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: ANL  A,#0FH ; 将累加器的低四位转换成ASCII码
      ADD  A,#90H
      DA   A
      ADDC A,#40H
      DA   A
      RET
```

(1 5) 标号： ??? 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： ??? 功能：单字节十六进制整数转换成单字节???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???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???码
      RET
```

(1 7) 标号： ?? 2 功能：双字节十六进制整数转换成双字节???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???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 ; ???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 ; ???码带进位自身相加, 相当于乘 2
ADDC A,R5
DA A ; 十进制调整
MOV R5,A
MOV A,R4
ADDC A,R4
DA A
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 ???
RET

```

(1 8) 标号: ??? 功能: 单字节十六进制小数转换成单字节???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???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???码小数
DA A ; 调整后若有进位, 原小数接近整数 1
RET

```

(1 9) 标号: ??? 2 功能: 双字节十六进制小数转换成双字节???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???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

```

```

MUL  AB
MOV  R3,A
MOV  R5,B
MOV  A,R2
MOV  B,#10
MUL  AB
ADD  A,R5
MOV  R2,A
CLR  A
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
SWAP  A
ORL  A,R3  ; 拼装成低字节???码小数
MOV  R3,A
POP  ACC  ; 取出百分位
MOV  R2,A
POP  ACC  ; 取出十分位
SWAP  A
ORL  A,R2  ; 拼装成高字节???码小数
MOV  R2,A
RET

```

(2 0) 标号: ??? 功能: 单字节???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???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
      MUL  AB
      ADD  A,R4  ; 按十六进制加上个位
      RET

```

(2 1) 标号: ?? 2 功能: 双字节???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???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: ??? 功能: 单字节???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???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
      DA A
      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 ???小数
      ADD A,#0B0H ; 剩余部分达到 0 . 5 0 否?
      JNC BHD1 ; 四舍
      INC R3 ; 五入
BHD1: MOV A,R3 ; 取结果
      RET

```

(2 3) 标号: ??? 2 功能: 双字节???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???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
      DA A
      MOV R5,A
      MOV A,R4
      ADDC A,R4
      DA A
      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 ???小数
MOV A,R4
ADD A,#0B0H ; 剩余部分达到 0 . 5 0 否?
JNC BHD4 ; 四舍
INC R3 ; 五入
MOV A,R3
JNZ BHD4
INC R2
BHD4: RET

```

(2 4) 标号: ?? 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
DEC A ; 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: ??? 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
 DEC A ; 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
 POP DPH ; 恢复数据首址

POP DPL
MMSE: RET

(2 6) 标号: ??? 1 功能: 顺序查找 (???) 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: ??? 2 功能: 顺序查找 (???) 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: CLR A ; 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: ??? 1 功能: 对分查找 (???) 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: ??? 2 功能: 对分查找 (???) 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: ???1 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: ??? 2 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  ; 累加完全部数据
      POP  DPH  ; 恢复首址
      POP  DPL
      LJMP DV31  ; 求R3R4R5/R7→R4R5, 得到平均值

```

(3 2) 标号: ?? 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： ??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
      MOVX  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： ???? 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) ???—5 1 浮点运算子程序库及其使用说明

本浮点子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???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；???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 $? = ?? \sqrt{?} ??? (a? / ? + ?) ?$

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???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???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???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???码浮点操作数?
      LCALL  BTOF      ; 将其转换成二进制浮点操作数
      MOV    R0,#36H    ; 指向???码浮点操作数?
      LCALL  BTOF      ; 将其转换成二进制浮点操作数
      MOV    R0,#33H    ; 指向???码浮点操作数?
      LCALL  BTOF      ; 将其转换成二进制浮点操作数
      MOV    R0,#30H    ; 指向???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?
      LCALL  FMUL      ; 进行浮点乘法运算
      MOV    R1,#36H    ; 指向二进制浮点操作数?
      LCALL  FDIV      ; 进行浮点除法运算
      MOV    R1,#39H    ; 指向二进制浮点操作数?
      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???码浮点数
```

```
STOP: LJMP  STOP
```

```
END
```

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

(1) 标号: ??? 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: ??? 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
SJMP AS ; 计算代数和

(3) 标号: ??? 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
XRL A,@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
XRL A,@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 ???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 ???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1, 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: ???? 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 $R3R4 \times R6R7 \rightarrow R3R4$ 
        MOV   B,R4
        MUL   AB
        MOV   A,B
        XCH   A,R7
        MOV   B,R3
        MUL   AB
        ADD   A,R7
        MOV   R7,A
        CLR   A
        ADDC  A,B
        XCH   A,R4
        MOV   B,R6
        MUL   AB
        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: CLR A ; 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: ??? 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
      ORL A,@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, $R3R4/R6R7 \rightarrow R0R1$

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: ???? 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: ???? 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
        MOV  A,@R0
        DEC  R0
        ORL  A,@R0

```

```

DEC  R0
JNZ  ZERO
MOV  @R0,#41H
ZERO: RET

```

(8) 标号: ???? 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
      MOV  A,@R1
      MOV  @R0,A
      DEC  R0
      DEC  R1
      MOV  A,@R1
      MOV  @R0,A
      DEC  R0
      DEC  R1
      MOV  A,@R1
      MOV  @R0,A
      RET

```

(9) 标号: ???? 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: ???? 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: ???? 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: ???? 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: ???? 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
RLC A ; 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: ???? 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 1 6 否?
      JC INTD
      RET ; 阶码大于 1 6 , 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
      RL A
      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 1 6 (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

```

(1 5) 标号: ??? 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 . 0 0 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

```

(1 6) 标号: ??? 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?1 指针
      LCALL FMUL ; 进行乘法运算
      POP  ACC
      MOV  R1,A  ; 恢复?1 指针
      RET
```

(1 7) 标号： ???? 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
      ORL  A,@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 ???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: ??? 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 ?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: ??? 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?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 / ?? 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: ??? 功能: 以?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: ??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 (???2 1 0 )
    MOV   R6,#0D4H
    MOV   R7,#9AH
    SJMP  EXP ; 先进行加权运算, 后以 2 为底统一求幂

```

(2 2) 标号: ???? 功能: 以?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 (???2?)
    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
    SJMP  E20 ; 以 2 为底统一求幂

```

(2 3) 标号: ??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
      ORL   A,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;  $1.3564 \times 10^{-3}$ 
      DB  7AH,0A1H,68H;  $9.8514 \times 10^{-3}$ 
      DB  7CH,0E3H,4FH;  $0.055495$ 
      DB  7EH,0F5H,0E7H;  $0.24014$ 
      DB  0,0B1H,72H;  $0.69315$ 
      DB  1,80H,0 ;  $1.00000$ 
      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: ??? 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

```

DFOB: 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: ??? 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 ; 右嬉准?br> SJMP FTD8

```

(2 6) 标号: ??? 功能: 浮点???码转换成格式化浮点数

入口条件: 浮点???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? - 1 9?
JC BTF2
MOV @R0,#41H; 小于 1? - 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? 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???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
DA A
MOV R7,A

```

```

MOV  A,R6
ADDC  A,R6
DA  A
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: ???? 功能: 格式化浮点数转换成浮点???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???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 ?- 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 ?- 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。
      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
      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
      DA  A
      MOV  R7,A
      CLR  A
      ADDC A,R6
      DA  A
      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???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

BFL0: 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: ??? 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  ; ?+ (  $\pi / 2$  )
      LCALL MOV0  ; 保存结果, 接着运行下面的FSIN程序

```

(2 9) 标号: ??? 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 . 6 3 6 6 2 7 (  $2 / \pi$  )
      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: ??? 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; -0.055802
      DB  7FH,8FH,37H; 0.27922
      DB  0FFH,0EDH,0E0H; -0.46460
      DB  7BH,0E8H,77H; 0.028377
      DB  0,0FFH,68H; 0.9977
      DB  72H,85H,0ECH; 3.1930 × 10-5
      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 1.5708 (π/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

```

(31) 标号: ??? 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 (180/π) 传送到第二工作区
      MOV  R6,#0E5H
      MOV  R7,#2FH
      SJMP  DR  ; 通过乘法进行转换

```

(32) 标号: ??? 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 ( $\pi/180$ ) 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
      LJMPL MOV0 ; 结果传送到[R0]中
      END
```