第四章 80C51单片机汇编语言程序设计

帅千钧

Email:sqj@cuc.edu.cn

办公室：主楼813

本章内容

- 汇编语言程序设计概述
- 汇编语言程序设计基础
  - 顺序结构程序
  - 分支结构程序
  - 循环结构程序
  - 子程序设计
- 应用举例

§1 汇编语言程序设计概述

- 程序设计语言
  - 机器语言
  - 汇编语言
  - 高级语言
- 语言处理过程
  - 汇编：把汇编语言程序翻译成机器语言程序的过程。

  - 编译：把高级语言程序翻译成汇编语言程序的过程。

§1 汇编语言程序设计概述

- 汇编语言：

  汇编语言是以助记符表示的指令。

  每一条指令称为汇编语言的一条语句。

- 汇编语言程序设计：

  就是使用汇编语言指令来编写计算机应用程序的过程。

§1 汇编语言程序设计规范

- 按指令格式和语法规则编写程序。
  - 常数的表示：

    十进制数：　　20

    十六进制数：　　87H，0F0H

    二进制数：　　01011001B

    字符：　　‘H’

    字符串：　　“Hello”
- 使用伪指令提供汇编信息。

§1 汇编语言的语句格式

- 语句格式表示如下：

  [标号:]操作码　[操作数]　　[;注释]

  - 标号，是用户定义的指令的符号地址，标号便于程序中的其它语句查询访问该语句。
    - 标号可以有、也可以没有；
    - 标号是以字母开始的1～8个ASCII字符组成；
    - 标号不可以和助记符、寄存器名、伪指令等重名。
  - 操作码，指令助记符或伪指令。
  - 操作数，参与运算的数据或数据地址。
  - 注释，用户定义的语句说明，不汇编成机器码。

例如，汇编语言格式：

地址	机器码	源程序	注释
		ORG 0000H	; 整个程序起始地址
0000	20 00 30	LJMP MAIN	; 跳向主程序
		ORG 0030H	; 主程序起始地址
0030	C3	MAIN: CLR C	; MAIN为程序标号
0031	E6	LOOP: MOV A , @R0	
0032	37	ADDC A, @R1	
0033	08	INC R0	
0034	D9 FB	DJNZ R1, LOOP	; 相对转移
0036	80 02	SJMP NEXT	
0038	78 03	MOV R0, #03H	
003A	18	NEXT: DEC R0	
003B	80 FE	SJMP $	; HERE: SJMP HERE
		END	; 结束标记

§1 汇编语言的伪指令

- 伪指令：汇编控制指令，仅提供汇编信息，不属于指令系统，不产生机器代码。
- 常用伪指令及其功能：
  - **ORG (Origin)**

    功能：定位目的程序的起始地址。

    格式： ORG 表达式

例如：

指令地址	机器码	源程序
		ORG 2000H
2000H	78 30	MAIN：MOV R0，#30H
2002H	E6	MOV A，@R0
		…

  - 注意：表达式必须为16位地址值。

- **END**

  功能：汇编结束指令。

例如：START：　…

　　…

　　END

- 注：一定放在程序末尾！

- **EQU (EQUate)**

  功能：赋值伪指令。

  格式：字符名称　EQU　数值或汇编符号

例如：

```
TH      EQU    30H
FO      EQU    40H
CNT     EQU    R0
MOV A，TH              ；(30H)→A
MOV A，FO              ；(40H)→A
MOV CNT，#00H          ；00H →R0
```

- **DB　　(Define Byte)**

从指定单元开始定义(存储)若干个字节的数据或ASCII码字符，常用于定义数据常数表。

格式：DB　　字节常数(或ASCII字符)

例如：　　ORG　1000H

　　DB　34H，0DEH，’A’，’B’

　　DB　0AH，0BH，20

- **DW　　(Define Word)**

从指定单元开始定义（存储）若干个字的数据或ASCII码字符。

格式：DW　　字常数(或ASCII字符)

例如：　　ORG　2000H

　　DW　1234H，’B’

　　DW　0AH，20

常用的伪指令

DB,DW,DS都是针对程序存储器的使用

- **DS**

  在程序存储器中保留指定数目的单元

  格式： [标号:] DS 常数

- **BIT**

  位地址符号指令，把位地址赋于规定的字符名称

  格式：字符名称　　BIT　位地址

  例如：　ABC　BIT　P1.1

  QQ　BIT　P3.2

  CS　BIT　P2.7

- **DATA**

格式：符号名 DATA 表达式

伪操作指令举例：

```
TH    EQU   03H
CS     BIT    P2.7
          ORG  0030H
MAIN:  CLR   CS
LOOP:  MOV   A，@R0
          MOV  R1，#TH          ；相当于 MOV  R1，#03H
NEXT:  SJMP   $
          ORG 1100H
          DB    01H , 04H，  09H , 05H      ；
          END
```

ROM中

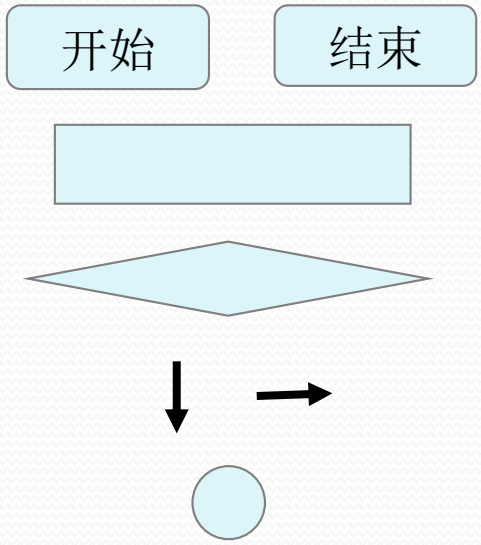
地址	数据
1100	01
1101	04
1102	09
1103	05

编制程序的步骤

- 任务分析(硬件、软件系统分析)
- 确定算法和工作步骤
- 程序总体设计和流程图绘制

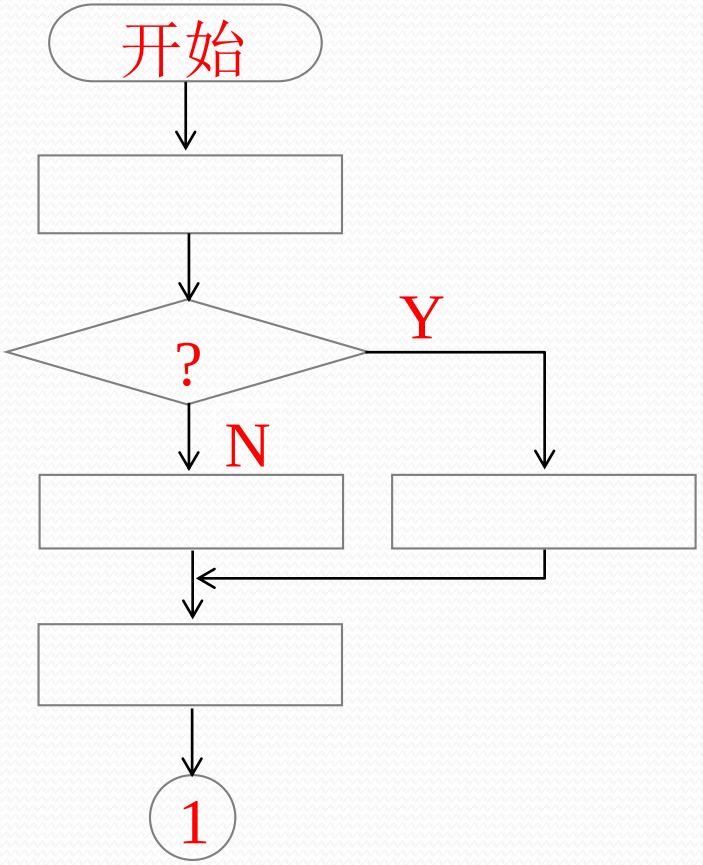
关于流程图符号：

- 开始、结束----圆角矩形
- 工作任务----矩形
- 判断分支----菱形
- 程序流向----
- 程序连接----



编制源程序

- 分配内存，确定程序与数据区存放地址以及I/O接口地址
- 按功能设计程序，明确各程序之间的相互关系
  - 模块化设计
  - 尽量采用循环及子程序结构
- 调试、修改，最终确定程序。



§2 汇编语言程序设计基础

- 编辑，(源程序，以.asm作为扩展名存盘)
- 汇编，(手工或机器汇编)
  - 确定程序中每条汇编语言指令的指令机器码
  - 确定每条指令在存储器中的存放地址。
  - 提供错误信息。
  - 提供目标执行文件（*.OBJ/*.HEX）和列表文件（*.LST）。
- 例如：

地址	目标码	源程序
		ORG 1000H
1000H	74 7F	MOV A,#7FH
1002H	79 44	MOV R1,#44H
		END

汇编语言程序设计基础

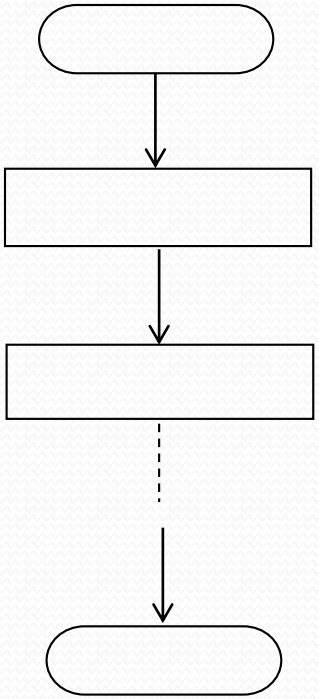
- 常用程序结构：
  - 直线程序
  - 分支程序
  - 循环程序
  - 子程序

直线程序(顺序结构程序)

- PC值自动增1、顺序一条一条地执行程序。

例1：把一个16位二进制数转换成补码(R4R5)

```
MOV  A，R5        ；取低字节
CPL  A
ADD  A，#1        ；低字节变补
MOV  R5，A
MOV  A，R4        ；取高字节
CPL  A
ADDC A，#0        ；高字节变补
MOV  R4，A
END
```



例2：压缩式BCD码分解成为单字节BCD码

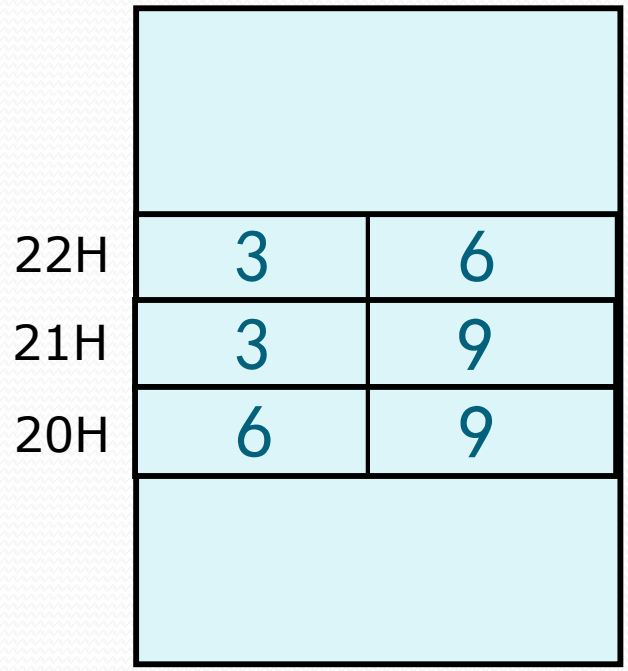
```
MOV  R0，#40H  ；设指针
MOV  A，@R0    ；取一个字节
MOV  R2，A     ；暂存
ANL  A，#0FH   ；清0高半字节保留个位
INC  R0
MOV  @R0，A    ；保存数据个位
MOV  A，R2
SWAP A         ；十位换到低半字节
ANL  A，#0FH
INC  R0
MOV  @R0，A    ；保存数据十位
```

片内 RAM	
42H	0　十
41H	0　个
40H	十　个

例2：将20H单元的压缩(Packed)BCD码拆成两个ACSII码存入21H、22H单元。

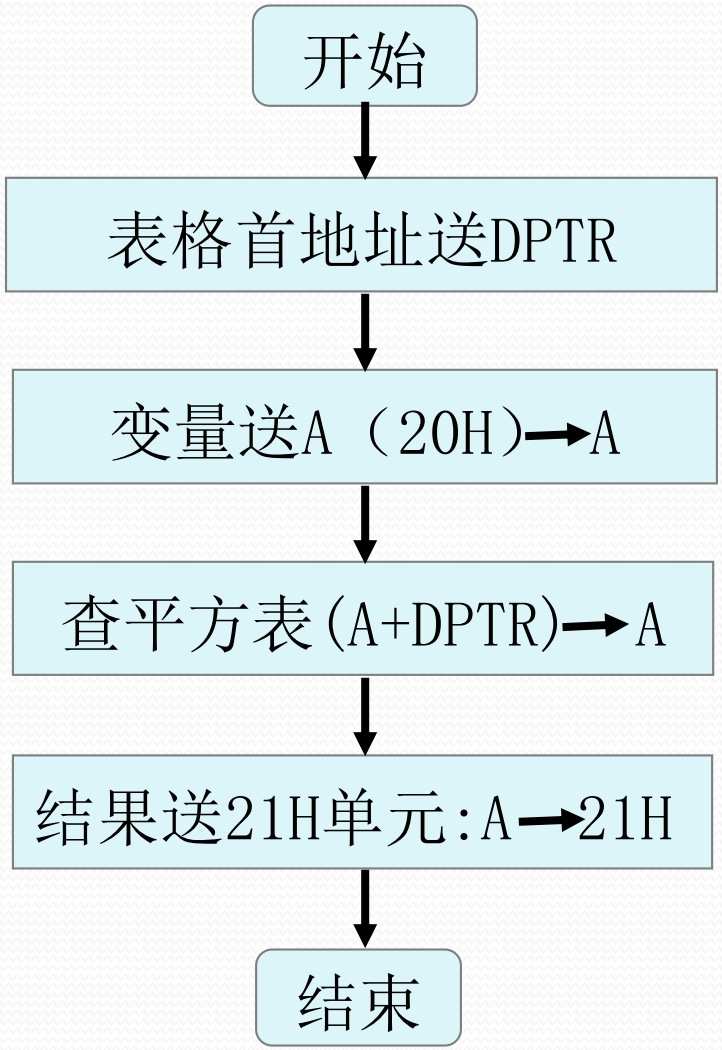
BCD	ASCII
0	30H
1	31H
2	32H
…	…
9	39H

内部RAM



例3：试编程求内部RAM的20H单元中变量的平方值，变量取值范围为0~5，结果存在21H单元。

```
ORG  1000H
     MOV  DPTR,#Table
     MOV  A,20H
     MOVC  A,@A+DPTR
     MOV  21H,A
     SJMP  $
ORG  2000H
Table:DB  0,1,4,9,16,25
END
```

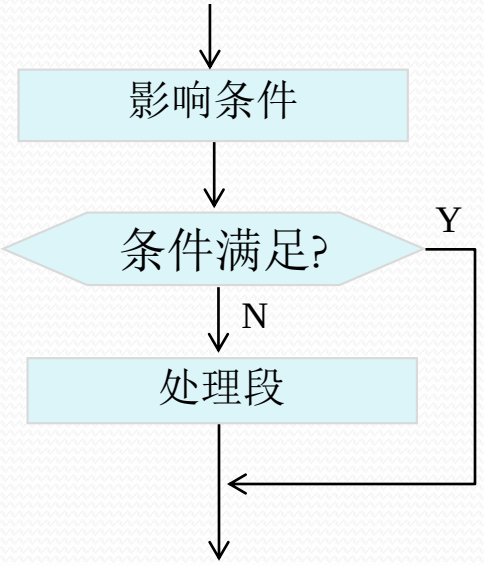


分支程序

- 分支程序可根据要求无条件或有条件地改变程序执行流向。
- 分支程序由转移控制指令构成程序判断框部分，形成程序分支结构。编写分支程序主要在于正确使用转移指令。
- 分支程序有：单分支结构、多分支结构(散转结构)。

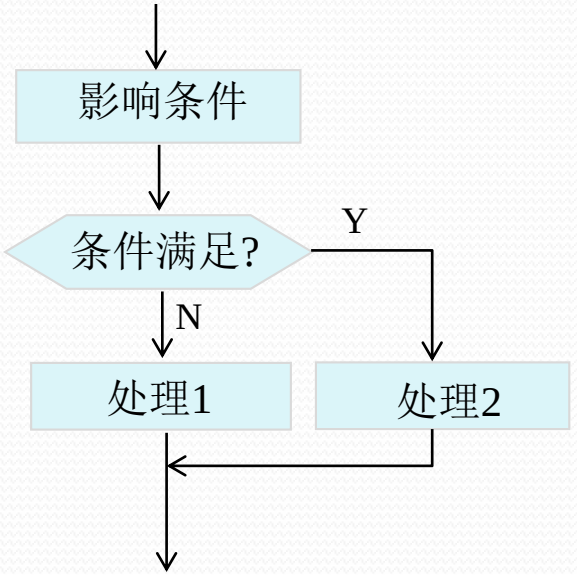
单分支结构

- 一个条件判决，程序有两条流向。
- 这种结构的流程图：



例4：求R2中补码绝对值，正数不变，负数变补。

```
START：MOV  A，R2
       JNB  ACC.7，NEXT；为正数？
       CPL  A              ；负数变补
       INC  A
       MOV  R2，A
NEXT：  SJMP NEXT          ；结束
```



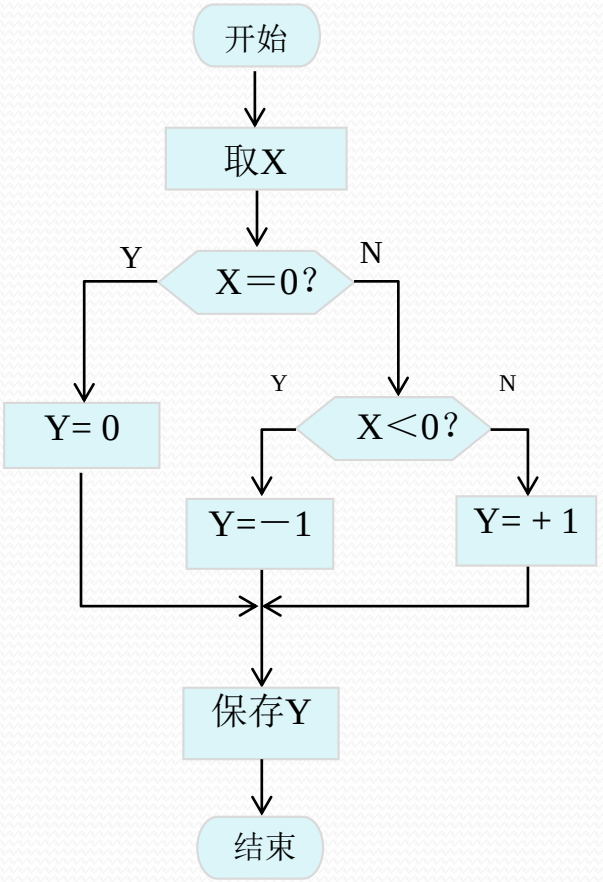
多重单分支结构

- 多次使用条件转移指令，有多个条件判断，形成多个逐级分支。

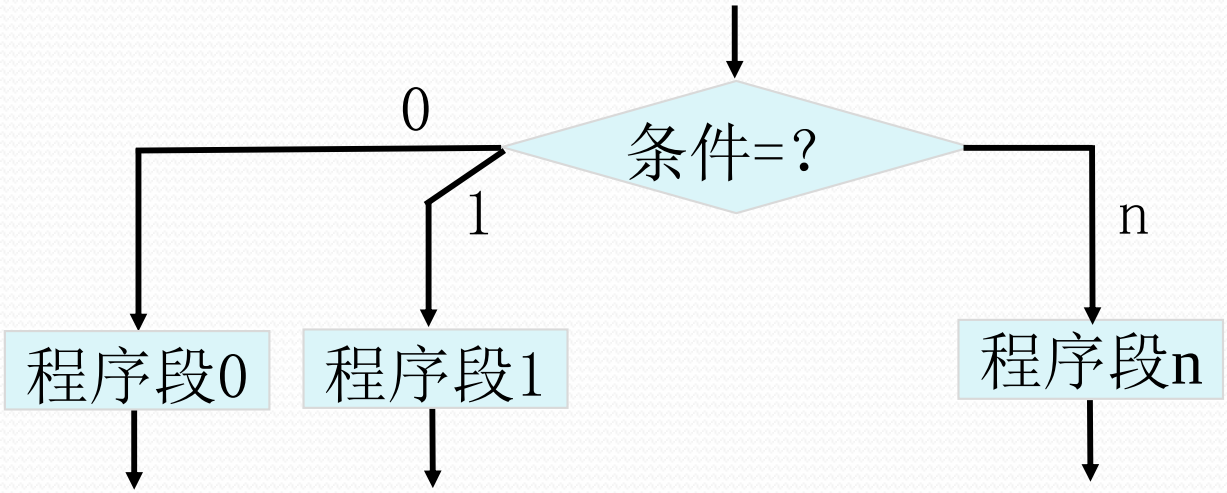
例5：X为补码表示的数，在40H单元，求符号函数Y=SGN(X)，存于41H单元。

$$SGN(X)=\begin{cases}+1 & 当\ X>0\\ 0 & 当\ X=0\\ -1 & 当\ X<0\end{cases}$$

```
START:  MOV  A，40H          ；取X
        JZ     STOR          ；X=0，Y=X
        JB     ACC7，NEGA    ；X＜0
        MOV  A，#1           ；X＞0，Y=+1
        SJMP  STOR
NEGA:   MOV  A，#0FFH        ；X＜0，Y= -1
STOR:   MOV  41H，A          ；保存Y
        SJMP $
        END
```



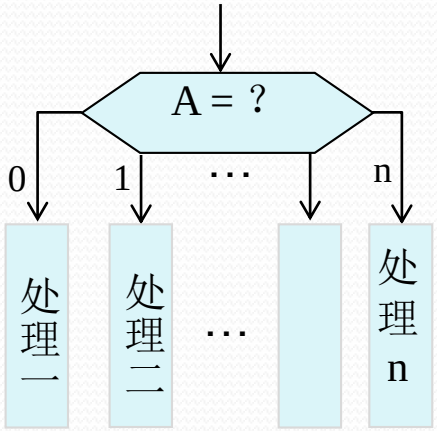
多分支结构(散转结构)



多分支结构

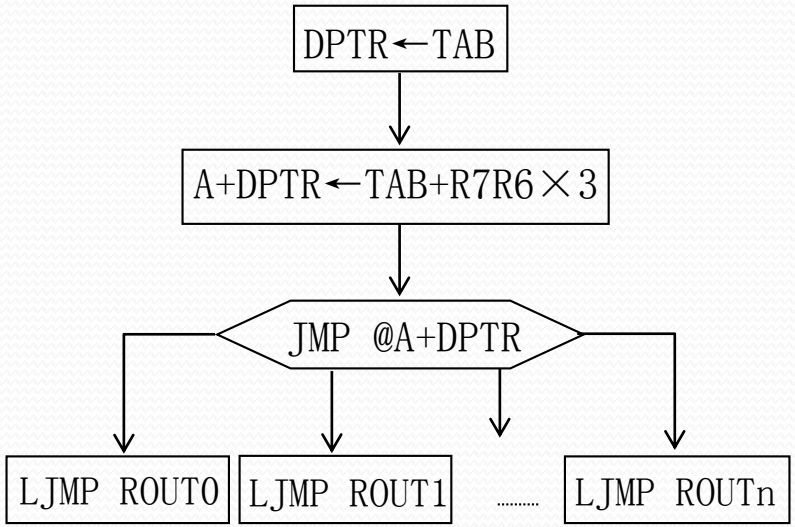
- **用指令：JMP　@A+DPTR**
- **0、1…n为分支号：**分支号=0，程序转移到ADDR0处；当分支号=1，程序转移到ADDR1处；…。

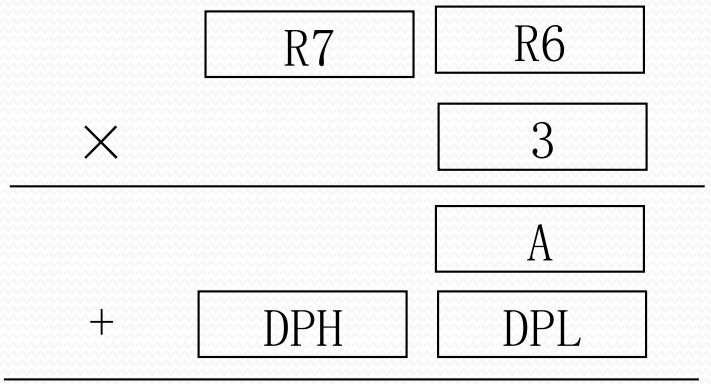
- 方法一：转移表法。
  - 用分支转移指令JMP @A+DPTR
  - 表里存放的是转移指令
  - 目的是要计算出表地址



例题：分支个数小于255的情况(书P81)

例题：分支个数大于255的情况，设R7R6=分支号





例题：

```
START：MOV DPTR，#TAB          ；指向表首地址
       MOV A，R7       ；分支号高字节×3
       MOV B，#03H
       MUL AB                  ；乘积不超过1字节
       ADD A，DPH
       MOV DPH，A
       MOV A，R6       ；分支号低字节×3
       MOV B，#03H
       MUL AB
       XCH A，B
       ADD A，DPH      ；DPH←DPH+((R7、R6)×3)高字节
       MOV DPH，A
       XCH A，B                ；A←((R7、R6)×3)低字节
       JMP @A+DPTR             ；实现多分支转移
TAB：  LJMP ADDR0      ；转移表
       LJMP ADDR1
            …
       LJMP ADDRN
ADDR0：      …                 ；程序段0 …
            …
ADDR1：      …                 ；程序段1 …
```

- 方法二：地址表法。
  - 表里存放的是定义分支程序入口地址
  - 目的是要计算出表里入口地址的具体值.

例题：用指令JMP @A+DPTR

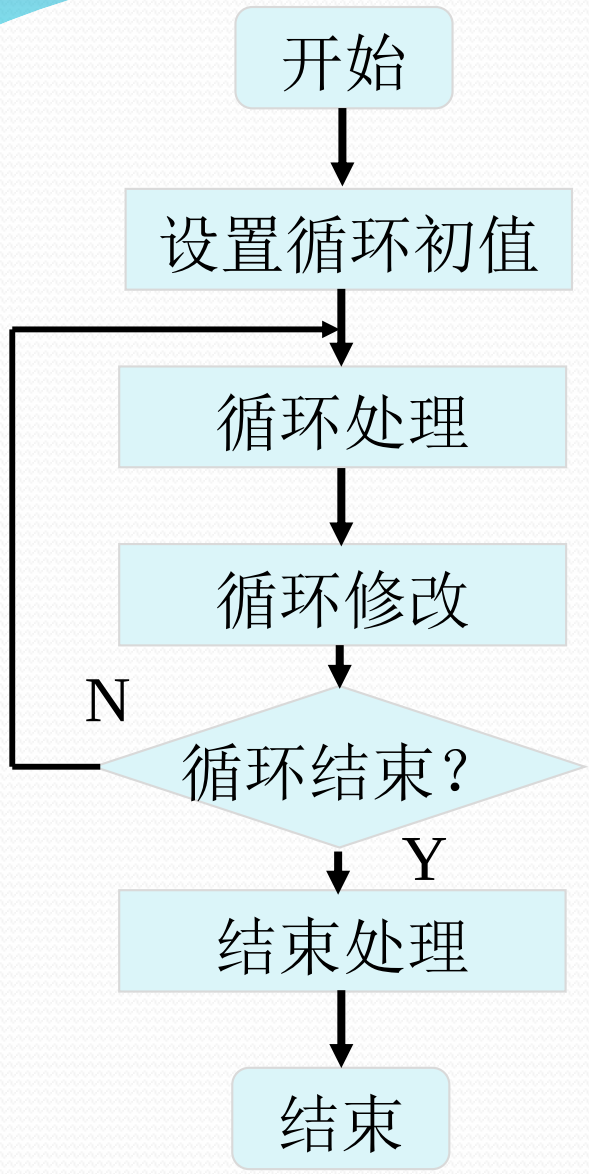
```
START：MOV  DPTR，#TAB          ；取表首地址
       MOV A，R3
       RLC   A                  ；分支号×2
       JNC LOOP
       INC DPH
LOOP：  MOV R4，A
       MOVC A，@A+DPTR          ；取分支程序入口地址高8位
       XCH A，R4                ；R4暂存地址高8位
       INC A                    ；分支号地址加1
       MOVC A，@A+DPTR          ；取分支程序入口地址低8位
       MOV DPL，A               ；地址低8位给DPL
       MOV DPH，R4              ；地址高8位给DPH
       CLR A
       JMP @A+DPTR
TAB：  DW    ADDR0              ；分支地址表
       DW    ADDR1
             …
ADDR0：      …                  ；程序段0 …
             …
ADDR1：      …                  ；程序段1 …
```

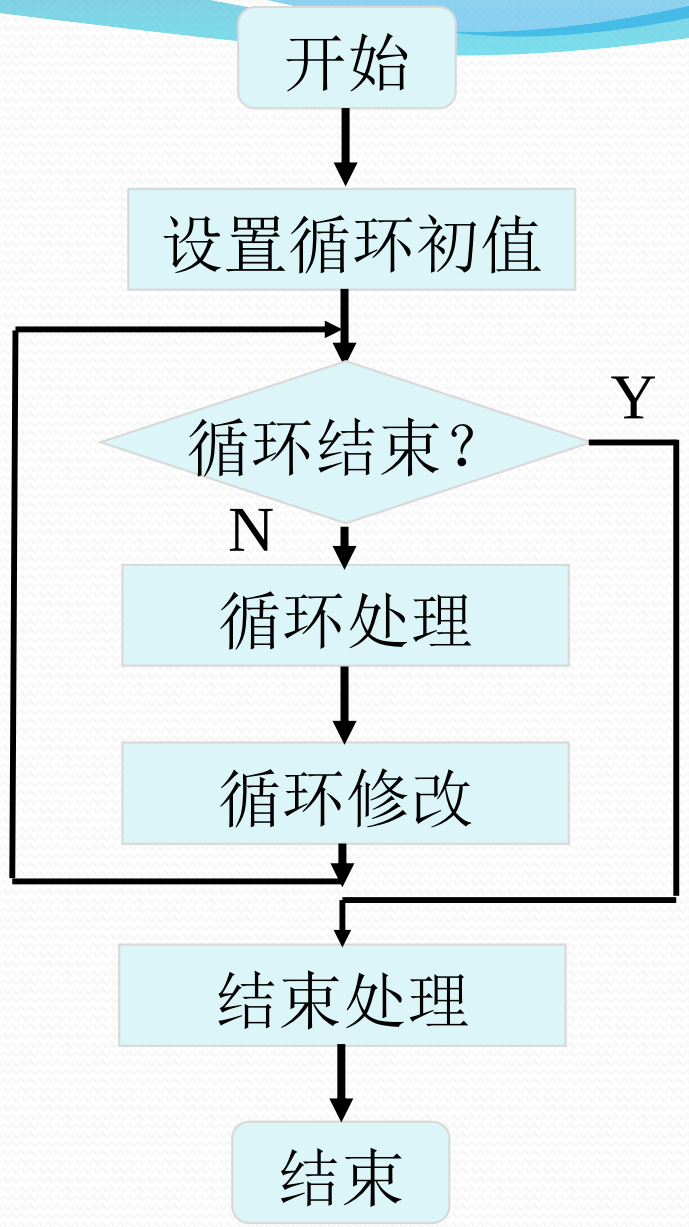
例题：用指令RET，通过堆栈弹出PC实现转移。

```
START：  MOV   DPTR，#TAB          ；取表首地址
         MOV  A，R3
         CLR    C
         RLC    A                  ；分支号×2
         JNC   LOOP
         INC   DPH
LOOP：   MOV   R4，A
         INC    A
         MOVC A，@A+DPTR           ；取分支程序入口地址低8位
         PUSH  ACC                 ；地址低8位入栈保存
         MOV   A，R4
         MOVC  A，@A+DPTR          ；取分支程序入口地址高8位
         PUSH  ACC                 ；地址高8位入栈保存
         RET                       ；分支地址→PC，转移
TAB：    DW     ADDR0              ；分支地址表
         DW     ADDR1
                …
ADDR0：          …                 ；程序段0 …
                …
ADDR1：          …                 ；程序段1 …
```

循环程序

- 循环程序一般结构：
  - 初始化部分
  - 循环体部分：处理、修改、控制部分
  - 结束部分：处理和保存循环结果
- 其结构一般有两种：
  - 先进入处理部分，再控制循环
    - 至少执行一次循环体
  - 先控制循环，再进入处理部分
    - 循环体是否执行，取决于判断结果


开始
设置循环初值
循环处理
循环修改
循环结束？
N
Y
结束处理
结束


开始
设置循环初值
循环结束？
Y
N
循环处理
循环修改
结束处理
结束

- 单重循环

  是一种简单的循环结构：循环体中不再包含循环的程序结构。

- 多重循环

  循环嵌套，循环体中还有循环的程序结构。

•单重循环

例6：将内部RAM中起始地址为data的数据串送到外部RAM中起始地址为buffer的存储区域中，直到发现‘$’字符，传送停止。

事先不知道循环次数，先判断，后执行。

```
        MOV  R0, #data
        MOV  DPTR, #buffer
        MOV R1,#20H
LOOP1:  MOV  A, @R0
        CJNE  A,#24H,LOOP2      ;判断是否为$字符
        SJMP  LOOP3             ;是，转结束
LOOP2:  MOVX  @DPTR, A          ;不是，传送数据
        INC R0
        INC DPTR
        DJNZ R1, LOOP1          ;传送下一数据
LOOP3:  SJMP LOOP3              ;停止
```

例7：求n个单字节数据的累加和，设数据串起始地址单元为43H，数据串长度在42H单元，累加和不超过2个字节，高字节存在41H，低字节存在40H。

```
SUM:   MOV    R0，#42H；设指针
       MOV    A，@R0
       MOV    R2，A    ；循环计数器←n
       CLR    A        ；结果单元清0
       MOV    R3，A    ；R3存高字节
ADD1:  INC    R0       ；修改指针
       ADD    A，@R0   ；累加
       JNC    NEXT     ；处理进位
       INC    R3       ；有进位，高字节加1
NEXT:  DJNZ   R2,ADD1  ；循环控制：数据是否加完？
       MOV    40H，A   ；循环结束，保存结果
       MOV    41H，R3
       SJMP $
```

片内	RAM
…	…
	Xn
…	…
43H	X1
42H	n
41H	SUM_H
40H	SUM_L

循环控制方法：计数控制、条件控制

- 计数控制(DJNZ)

循环次数已知，通过循环计数器来控制循环次数。

例8：为一串7位ASCII码数据的D_7位加上奇校验，设数据存放在片外RAM的2101H起始单元，数据长度在2100H单元。

```
        MOV   DPTR，#2100H
        MOVX  A，@DPTR
        MOV   R2，A
NEXT：  INC   DPTR
        MOVX  A，@DPTR
        ORL   A，#80H
        JNB    P，PASS
        MOVX  @DPTR，A
PASS：  DJNZ  R2，NEXT
HERE：  SJMP  HERE
```

片外RAM	
…	…
2102H	01101000
2101H	10101101
2100H	n

与	0	清0
	1	保持
或	0	保持
	1	置1

条件控制(CJNE)

循环次数未知，通过设定循环结束标志实现循环控制。

例9：找正数表最小值。正数表存在片外RAM中以POST为起始地址的单元中，用-1作为表的结束标志。

```
START：MOV   DPTR，#POST          ；数表首地址
       MOV   B，#127              ；预置最小值
NEXT：  MOVX  A，@DPTR            ；取数
       INC   DPTR                 ；修改指针
       CJNE  A，#-1，NEXT1        ；是否为数表结尾？
       SJMP  HERE                 ；循环结束
NEXT1：CJNE  A，B，NEXT2          ；比较
NEXT2：JNC   NEXT
       MOV   B，A                 ；保存较小值
       SJMP  NEXT
HERE：  SJMP  HERE
```

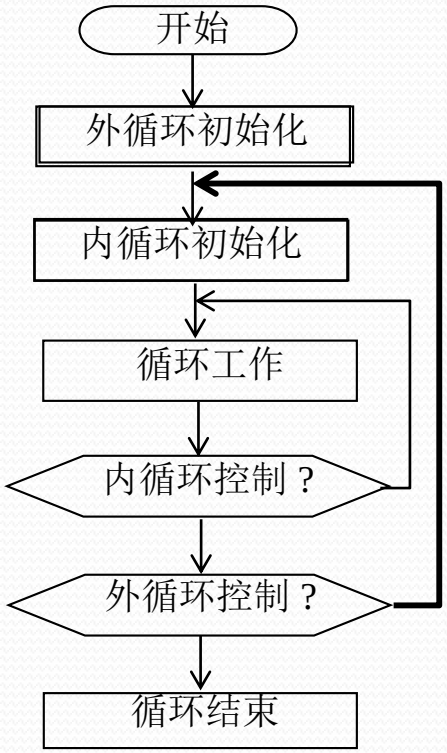
•多重循环

循环体中套循环结构，以双重循环使用较多。

例10：将内存一串单字节无符号数升序排序。

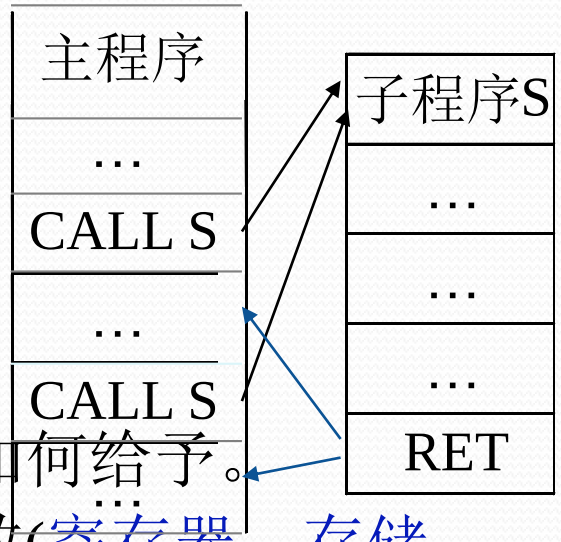
步骤：

- 每次取相邻单元的两个数比较，决定是否需要交换数据位置。
- 第一次循环，比较N-1次，取到数据表中最大值。
- 第二次循环，比较N-2次，取到次大值。
- …
- 第N-1次循环：比较一次，排序结束。



子程序

- 子程序：能完成某项特定功能的独立程序段，可被反复调用。
  - 子程序入口用标号作为子程序名，即为子程序入口地址。
  - 调用子程序之前设置好堆栈。
- 用返回指令RET结束子程序。
  - 此时堆栈释放调用程序的返回地址。
- 要能正确传递参数：
  - 入口参数：子程序中要处理的数据如何给予。
  - 出口参数：子程序处理结果如何存放(寄存器、存储器、堆栈方式)。
- 子程序可以嵌套。
  - 子程序中可调用其它子程序。
  - 子程序嵌套须考虑堆栈容量。



例11：利用查表法求两个个位数的平方和 $c=a^2+b^2$，a、b、c分别存于内部RAM41H、42H、40H三个单元中。

```
        MOV   A,41H          ;取A
        ACALL  SQR           ;调用查表子程序
        MOV   R1,A           ;A的平方暂存R1中
        MOV   A,42H          ;取B
        ACALL  SQR           ;调用查表子程序
        ADD   A,R1           ;求出平方和暂存A中
        MOV   40H,A          ;结果存于40H中
        SJMP   $
SQR:    MOV   DPTR,#TAB      ;子程序
        MOVC  A,@A+DPTR
        RET
TAB:    DB 0,1,4,9,16,25,36,49,64,81
        END
```

§3 应用举例

- 单片机应用程序设计
  - 主程序
    - 是顺序执行的无限循环的程序
    - 运行过程处于全封闭状态
  - 子程序
    - 是完成某种特定功能的一个程序段
    - 要注意的是其入口参数和出口参数

```
ORG 0000H
LCALL MAIN
ORG 0030H
MAIN: …
```

§3.1 运算类应用程序

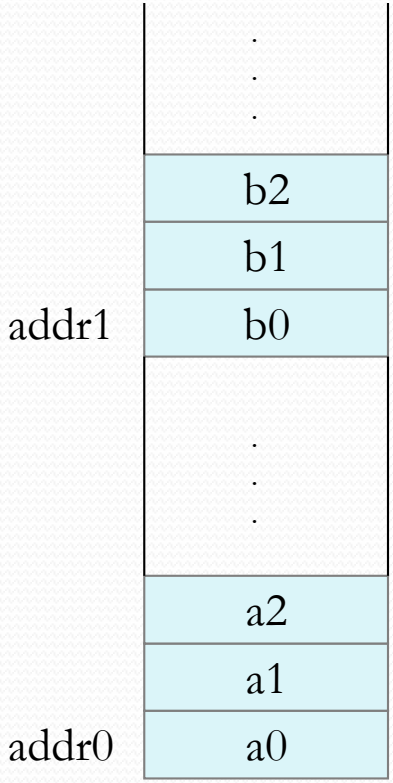
- 各种形式的加、减、乘、除运算程序。
  - 加法：
    - 掌握多字节无符号数加法程序和多字节求累加和的加法程序编制方法及其区别(见下面例12和例7)
    - 十进制加法运算(例13，多字节十进制加法)

例12：多字节无符号数加法

入口参数：字节数在R2中，被加数在[R0]中，加数在[R1]中。

出口参数：和在[R0]中，最高进位位存F0中。

```
MBADD：MOV  R2，#data          ;赋初值
       MOV  R0，#addr0
       MOV  R1，#addr1
       CLR  C
ADD1：  MOV A，@R0             ;取被加数
       ADDC A,@R1              ;相加
       MOV  @R0,A              ;和保存
       INC  R0                 ;修改指针
       INC  R1
       DJNZ  R2,ADD1
       MOV  F0,C
       RET
```



$$\begin{array}{rccccc} & & \cdots & a2 & a1 & a0 \\ + & C & \cdots & b2 & b1 & b0 \\ \hline & F0 & \cdots & c2 & c1 & c0 \end{array}$$

- 例7：求n个单字节数据的累加和，设数据串起始地址单元为43H，数据串长度在42H单元，累加和不超过2个字节，高字节存在41H，低字节存在40H。

片内 RAM	
…	…
	Xn
…	…
43H	X1
42H	n
41H	SUM H
40H	SUM L

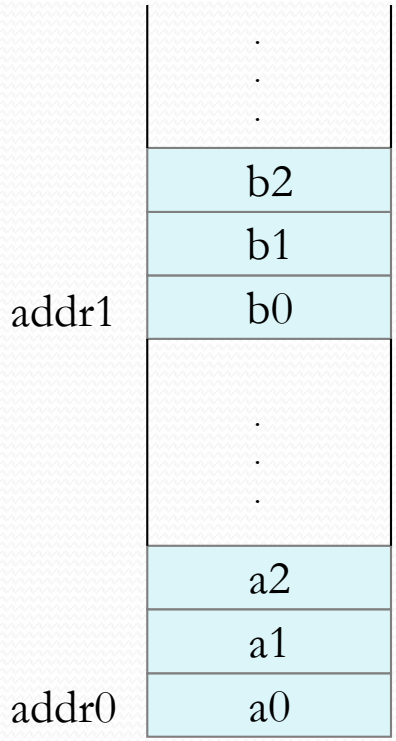
```
SUM:   MOV   R0，#42H;  设指针
       MOV   A，@R0
       MOV   R2，A     ;  循环计数器←n
       CLR   A         ;  结果单元清0
       MOV   R3，A     ;  R3存高字节
ADD1:  INC   R0        ;  修改指针
       ADD   A，@R0    ;  累加
       JNC   NEXT      ;  处理进位
       INC   R3        ;  有进位，高字节加1
NEXT:  DJNZ  R2,ADD1   ;  循环控制：数据是否加完？
       MOV   40H，A    ;  循环结束，保存结果
       MOV   41H，R3
       RET
```

例13：多字节BCD码加法

入口参数：字节数在R2中，被加数在[R0]中，加数在[R1]中。

出口参数：和在[R0]中，最高进位位存F0中。

```
        MOV  R2,#data
        MOV  R0,#addr0
        MOV  R1,#addr1
BCDAD:  MOV  A，R2
        MOV  R3，A
        MOV  A，R0
        MOV  R4，A
        CLR  C
BCDAD1: MOV A,@R0
        ADDC A,@R1
        DA  A
        MOV  @R0,A
        INC  R0
        INC  R1
        DJNZ  R3,BCDAD1
        MOV  F0,C
        MOV A，R4
        MOV  R0，A
        RET
```



$$\begin{array}{rcl} & & \cdots\ a2\ \ a1\ \ a0 \\ + & C & \cdots\ b2\ \ b1\ \ b0 \\ \hline & F0 & \cdots\ c2\ \ c1\ \ c0 \end{array}$$

减法程序

1. 多字节无符号数减法(书例题P84)
2. 多字节BCD码减法：

入口参数：字节数在R2中，被减数在[R0]中，减数在[R1]中。

出口参数：差在[R0]中，最高位借位在F0中。

◆ 多字节BCD码求补码

入口参数：字节数在R2中，操作数在[R0]中。

出口参数：结果在[R0]中。

```
NEG:   MOV   A，R2
       DEC  A
       MOV  R7，A          ；设置循环次数R7
       CLR C
       MOV A,#9AH          ；最低字节单独取补
       SUBB A,@R0
       MOV @R0,A
NEG0: INC  R0
      MOV A,#99H
      SUBB A,@R0           ；按字节十进制取补
      MOV @R0,A            ；存回[R0]中
      DJNZ R7,NEG0         ；处理完（R7）字节
      RET
```

多字节BCD码减法

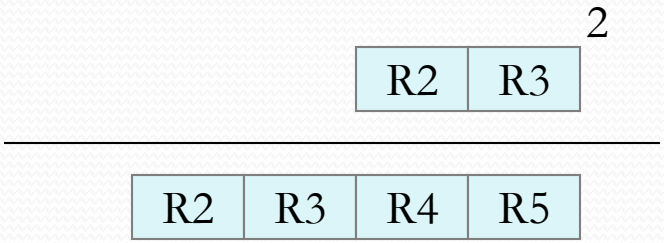
- 入口参数：字节数在R2中，被减数在[R0]中，减数在[R1]中。
- 出口参数：差在[R0]中，最高位借位在F0中。

```
MBSUB：  MOV  R2,#data
         MOV  R0,#addr1
         LCALL  NEG          ；减数取十进制补码
          MOV  R0,#addr0
          MOV  R1,#addr1
         LCALL  BCDAD
         CPL  C
         MOV  F0，C          ；保存进位
         RET
```

- 乘法程序
  - 双字节无符号数乘法(书例题P85)
  - 双字节无符号数平方

入口参数：操作数在R2R3中

出口参数：结果在R2R3R4R5中




```
MUL: 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
     XCH A,B
     XCH A,R2
     MUL AB        ;  计算2R2*R3
     CLR C
     RLC A
     XCH A,B
     RLC A
     JNC MUL0
     INC R2        ;  累加溢出量
MUL0:  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
```

§3.2 代码转换类应用程序

- BCD码转换成16进制

```
        MOV A, #69D
```

- BCDH: MOV B,#10H

```
        DIV  A,B
        MOV R2,B
        MOV  B,#10H
        MUL  A,B
        ADD A,R2
       RET
```

- 16进制转换成BCD码

```
        MOV A, #69H
HBCD:MOV B,#100
        DIV  A,B
        MOV R3,A
        MOV  A,#10
        XCH  A,B
        DIV  A,B
        SWAP  A
        ADD  A, B
        MOV  R2,A
        RET
```

十六进制数与ASCCII码之间的转换

- 十六进制数与ASCII码之间的关系

十六进制数		十六进制数		十六进制数		十六进制数	
0	30H	4	34H	8	38H	C	43H
1	31H	5	35H	9	39H	D	44H
2	32H	6	36H	A	41H	E	45H
3	33H	7	37H	B	42H	F	46H

●单字节十六进制转换成双字节ACSII 码

入口条件：单字节十六进制数在A中。

出口信息：高四位的ASCII码在A中，低四位的ASCII码在B中。

0~9的ASCII码：30~39H；A~F的ASCII码：41~46H。

- 方法一：

```
        MOV A, #6FH
HXASC:MOV B,A
        ANL A,#0FH
        MOV DPTR,#TAB
        MOVC A,@A+DPTR
        XCH  A,B
        SWAP  A
        ANL  A,#0FH
        MOVC  A,@A+DPTR
TAB:    DB  30H,31H…
        DB  41H,42H…
```

- 方法二：

```
        MOV A, #6FH
HXASC:MOV B,A
        LCALL HXASC1
        XCH A,B
        SWAP A ；  高四位
HXASC1: ANL A,#0FH
        ADD A,#90H
        DA A
        ADDC A,#40H
        DA A
        RET
```

- ASCII码转换为十六进制数

入口条件：单字节ASCII数在R2中。

出口信息：转换后的十六进制数仍存在R2中。

0~9的ASCII码：30~39H；A~F的ASCII码：41~46H。

```
ASCHX：MOV A,R2
       CLR C
       SUBB  A,#30H
       MOV  R2, A
       SUBB A,#0AH
       JC  NEXT
       XCH  A,R2
       SUBB  A,#07H
       MOV R2,A
NEXT:  RET
```

```
ASCHX: MOV A,R2
       CLR C
       SUBB A,#30H
       JNB ACC.4,NEXT
       SUBB A,#07H
NEXT: MOV R2,A
       RET
```

§3.3 查表类应用程序

- 表格数据在程序存储器中，用指令MOVC
  - MOVC A，@A+DPTR
  - MOVC A，@A+PC
- 表格数据在数据存储器中，用指令MOVX
  - MOVX A，@DPTR
  - MOVX @DPTR，A

顺序查找（ROM）单字节表格

入口参数：待查内容在A中，表格首址在DPTR中，表格的字节数在R7中。

出口参数：OV=0时，顺序号在累加器A中；OV=1时，未找到。

```
FDB: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
FD1: MOV A,R2        ；按顺序号读取表格内容
    MOVC A,@A+DPTR
    CJNE A,B,FD2     ；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
FD2: INC R2          ；指向表格中的下一个内容
    DJNZ R6,FD1      ；查完全部表格内容
    SETB OV          ；未查找到，失败
    RET
```

§3.4 定时程序

- 单循环定时程序

```
        MOV R2, #N
DLY: NOP
        NOP
        DJNZ R2, DLY
        RET
```

- 循环嵌套定时程序

```
        MOV R2, #N
DLY1:MOV R3, #M
DLY2: NOP
        NOP
        DJNZ R3, DLY2
        DJNZ R2, DLY1
        RET
```

实验编程

- 设晶振频率为12MHz，累加器里是十进制数，初值为0：试编程每隔1ms让累加器增1，当累加器增至100时，P1.0口输出反相，同时累加器返回初值，重复循环。

课后作业