第3章

指令系统与汇编语言程序设计

3.1 指令系统简介

[标号:]　操作码　[目的操作数]　[，　源操作数]　　[；注释]

WT：　ADDC　A，＃5BH　；　（A）←（A）+5BH，

direct：
@Ri：
Rn：
#data：
#data16：
addr11：
addr16：
rel：
bit：
A或ACC：

3.2 寻 址 方 式

1．直接寻址

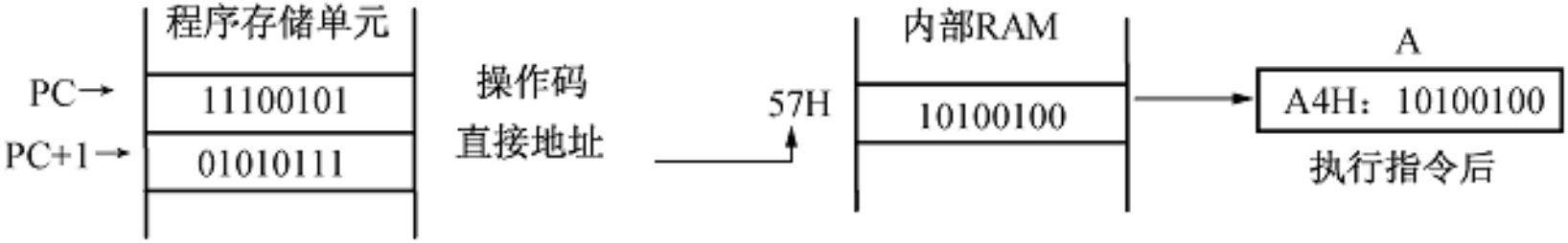


图 3-1　直接寻址指令操作示意

3.2 寻 址 方 式

2．立即数寻址

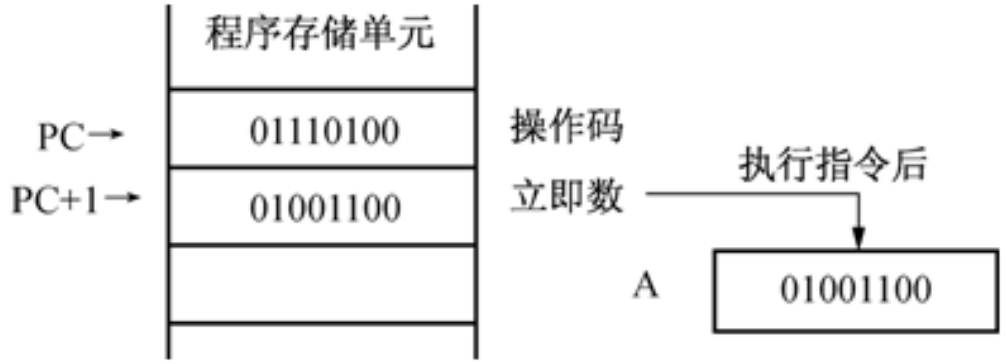


图 3-2　立即数寻址指令操作示意

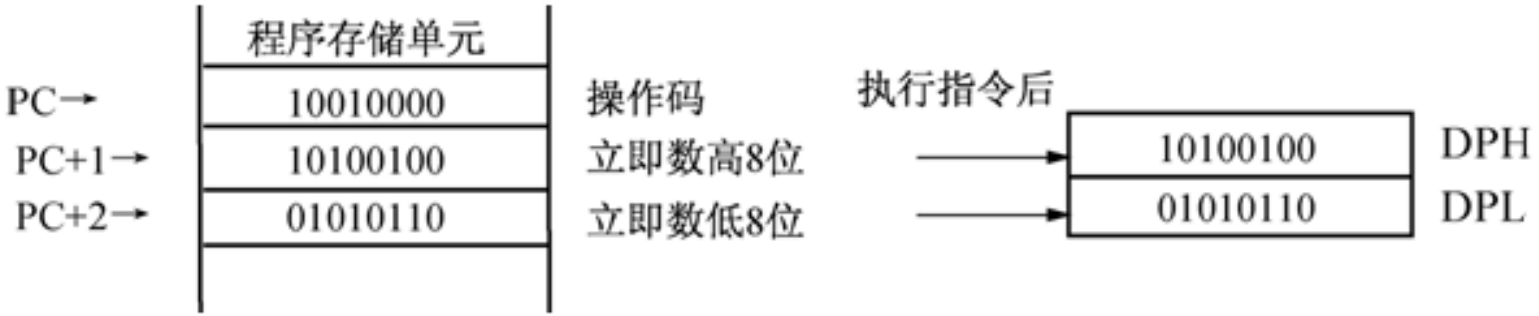


图 3-3　立即数寻址指令操作示意（16 位）

3.2 寻 址 方 式

3．相对寻址

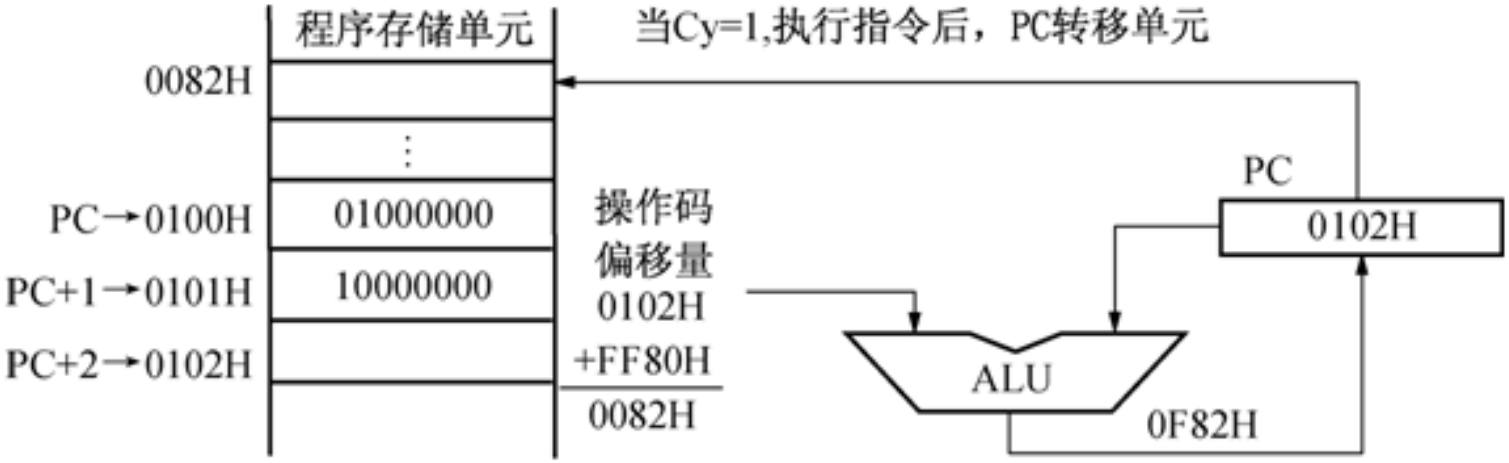


图 3-4　相对寻址指令操作示意

3.2 寻 址 方 式

4．变址寻址

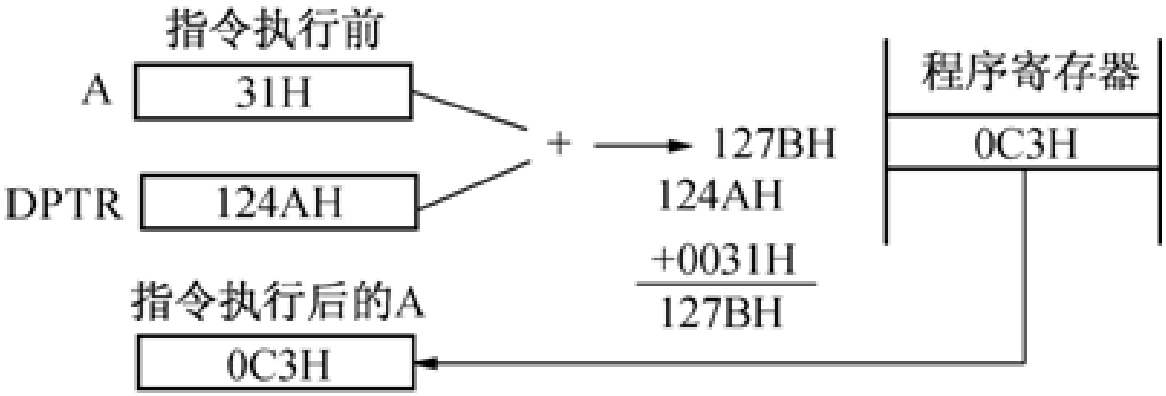


图 3-5　变址寻址指令操作示意

3.2 寻 址 方 式

5．寄存器寻址

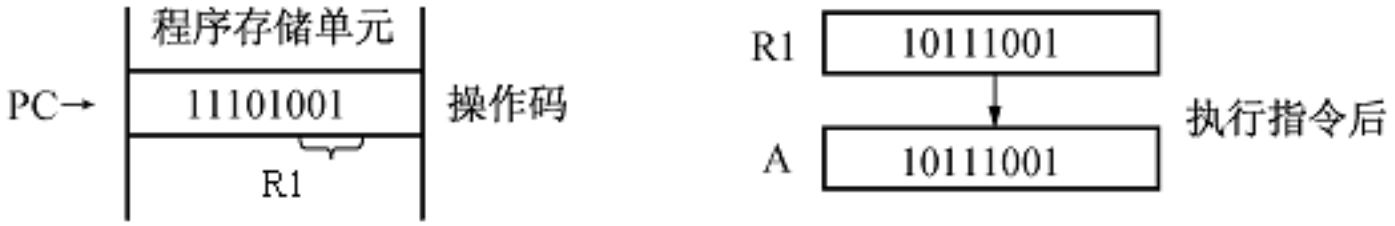


图 3-6　寄存器寻址指令操作示意

3.2 寻 址 方 式

6．寄存器间接寻址

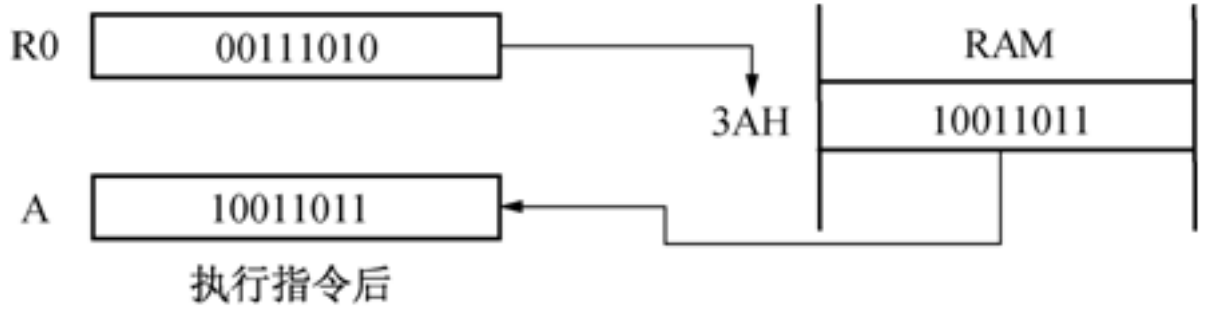


图 3-7　寄存器间接寻址指令操作示意

3.2 寻 址 方 式

7．位寻址

表 3-1　51 单片机寻址方式

序　　号	寻 址 方 式	使用的变量	寻 址 空 间
1	直接寻址	直接给出地址，无变量	内部 RAM 和特殊功能寄存器
2	立即数寻址	直接给出数值，无变量	程序存储器
3	相对寻址	PC+偏移量	程序存储器
4	变址寻址	A+DPTR、A+PC	程序存储器
5	寄存器寻址	R0～R7、A、B、C、DPTR	内部 RAM 或外部 RAM
6	寄存器间接寻址	@R0、@R1、SP	内部 RAM
		@R0、@R1、@DPTR	外部 RAM
7	位寻址	Bit	内部 RAM 和特殊功能寄存器的位空间

3.3 单片机汇编指令

3.3.1 数据传送指令

1. 单片机内部数据传送指令

（1）以累加器 A 作为目的操作数存放单元的数据传送类指令（4 条）：

```
MOV  A, Rn          ; A←（Rn）        : 将寄存器中的数据传送至累加器A
MOV  A, direct      ; A←（direct）    : 将直接地址单元中的数据传送至累加器A
MOV  A, @Ri         ; A←（Ri）        : 对寄存器以间接寻址方式将数据传送至累加器A
MOV  A, #data       ; A←data          : 将立即数传送至累加器A
```

（2）以寄存器作为目的操作数存放单元的数据传送类指令（3 条）：

```
MOV  Rn, A          ; Rn←（A）        : 将累加器中的数据传送至寄存器
MOV  Rn, direct     ; Rn←（direct）   : 将内部RAM直接地址单元中的数据传送至Rn
MOV  Rn, #data      ; Rn←data         : 将立即数传送至寄存器
```

3.3 单片机汇编指令

3.3.1 数据传送指令

（3）以内部 RAM 直接地址单元为目的操作数存放单元的数据传送类指令（5 条）：

```
MOV  direct，A        ；direct←（A）    ：将累加器中的数据传送至直接地址单元
MOV  direct，Rn       ；direct←（Rn）   ：将寄存器中的数据传送至直接地址单元
MOV  direct，@Ri      ；direct←（Ri）   ：以寄存器间址方式将数据传送至直接地址单元
MOV  direct1，direct2    ；direct1←（direct2）：将直接地址单元的数据传送至另一直
                                                接地址单元
MOV  direct，#data    ；direct←data     ：将立即数传送至直接地址单元
```

（4）以寄存器间接寻址方式确定地址单元为目的操作数存放单元的传送类指令（3 条）：

```
MOV  @Ri，A       ；（Ri）←（A）          ：将累加器的数据传送至寄存器间址指定的单元
MOV  @Ri，direct     ；（Ri）←（direct）：将直接地址单元的数据传送至寄存器间址指定
                                            的单元
MOV  @Ri，#data   ；（Ri）←data       ：将立即数传送至寄存器间址指定的单元
```

3.3 单片机汇编指令

3.3.1 数据传送指令

2．16位二进制数据传送指令

```
MOV  DPTR，#dat16              ；DPTR←data16 ：将16位二进制数据送入DPTR
```

3．针对外部扩展存储器的数据传送指令

```
MOVX  A，@Ri                   ；A←（（Ri）），    （i=0，1）
MOVX  @Ri，A                   ；  （Ri）←（A）
```

3.3 单片机汇编指令

3.3.1 数据传送指令

4. 数据交换指令

```
XCH   A, Rn          ;  (A) ↔ (Rn)          : 累加器A与工作寄存器Rn进行8位数据交换
XCH   A, direct      ;  (A) ↔ (direct)      : 累加器A与直接地址单元进行8位数据交换
XCH   A, @Ri         ;  (A) ↔ ( (Ri) )      : 累加器A与Ri间址单元进行8位数据交换

XCHD  A, @Ri             ;  (A) 3~0↔ ( (Ri) ) 3~0

SWAP  A                                  ;  (A) 3~0↔ (A) 7~4
```

3.3 单片机汇编指令

3.3.1 数据传送指令

5. 查表指令

```
MOVC  A, @A+DPTR          ; A ← ( (A) + (DPTR) )
MOVC  A, @A+PC            ; A ← ( (A) + (PC) )
```

6. 堆栈操作指令

```
PUSH  direct          ; SP← (SP) +1,  (SP) ← (direct)
POP   direct          ; direct← ( (SP) ) , SP← (SP) -1
```

3.3 单片机汇编指令

3.3.2 算术运算指令

表 3-2 算术运算指令

操作码	目标操作数	参与运算的操作数	指令功能说明	Cy	AC	OV	P
ADD	A	Rn; direct; @Ri; # data	不带进位加	√	√	√	√
ADDC	A	Rn; direct; @Ri; # data	带进位 C 的加				
SUBB	A	Rn; direct; @Ri; # data	带借位 C 的减				
DA	A		对加法作十进制修正	√	—	—	√
MUL	A B	（A）*（B）	乘积进（B）（A）	0	—	√	√
DIV	AB	（A）/（B）	商进（A），余数进（B）				
INC	A; Rn; direct; @Ri; DPTR		加 1 操作	—	—	—	√
DEC	A; Rn; direct; @Ri （无 DPTR）		减 1 操作				

3.3 单片机汇编指令

3.3.2 算术运算指令

1. 加法指令

```
ADD   A, Rn              ; A ← (A) - (Rn) :
ADD   A, direct          ; A ← (A) - (direct)
ADD   A, @Ri             ; A ← (A) - ( (Ri) )
ADD   A, #data           ; A ← (A) -data
```

```
ADDC  A, Rn              ; A ← (A) + (Rn) + (C)
ADDC  A, direct          ; A ← (A) + (direct) + (C)
ADDC  A, @Ri             ; A ← (A) + ( (Ri) ) + (C)
ADDC  A, #data           ; A ← (A) +data+ (C)
```

3.3 单片机汇编指令

3.3.2 算术运算指令

2．带借位的减法指令

```
SUBB  A，Rn                ；A←（A）-（Rn）-（C）
SUBB  A，direct            ；A←（A）-（direct）-（C）
SUBB  A，@Ri               ；A←A）-（（Ri））-（C）
SUBB A，#data              ；A←（A）- data -（C）
```

3．加1指令

```
INC  A                ；A←（A）+1              ：累加器加1指令
INC  Rn               ；Rn←（Rn）+1            ：寄存器加1指令
INC  direct           ；direct←（direct）+1    ：8位加1指令
INC  @Ri              ；（Ri）←（（Ri））+1     ：8位加1指令
INC  DPTR             ；DPTR←（DPTR）+1        ：16位加1指令
```

3.3 单片机汇编指令

3.3.2 算术运算指令

4. 减1指令

```
DEC  A              ; A←（A）- 1              : 累加器减1
DEC  Rn             ; Rn←（Rn）- 1            : 寄存器减1指令
DEC  direct         ; direct←（direct）- 1       : 8位减1指令
DEC  @Ri            ; （Ri）←（（Ri））- 1     : 8位减1指令
```

5. 十进制调整指令

```
DA  A
```

6. 乘法指令

```
MUL  AB          ; BA←（A）×（B）
```

7. 除法指令

```
DIV  AB
```

3.3 单片机汇编指令

3.3.3 逻辑运算指令

表 3-3 逻辑运算类指令

操 作 码	目标操作数	参与运算的操作数	指令功能说明
ANL	A	Rn; direct; @Ri; #data	逻辑与
	direct	A; #data	
ORL	A	Rn; direct; @Ri; #data	逻辑或
	direct	A; #data	
XRL	A	Rn; direct; @Ri; #data	逻辑异或
	direct	A; #data	
CPL	A	无	累加器取反
CLR	A	无	累加器清 0

3.3 单片机汇编指令

3.3.3 逻辑运算指令

1. 逻辑与运算指令

```
ANL   A, Rn                ; A←（A）&（Rn），这里“&”表示“与”逻辑操作
ANL   A, direct            ; A←（A）&（direct）
ANL   A, @Ri               ; A←（A）&（（Ri））
ANL   A, #data             ; A←（A）&  data
ANL   direct, A            ; direct←（direct）&（A）
ANL   direct, #data        ; direct←（direct） &data
```

3.3 单片机汇编指令

3.3.3 逻辑运算指令

2．逻辑或运算指令

```
ORL  A，Rn                ；A←（A）or（Rn），这里“or”表示“或”逻辑操作
ORL  A，direct            ；A←（A）or（direct）
ORL  A，@Ri               ；A←（A）or（（Ri））
ORL  A，#data             ；A←（A）or  data
ORL  direct，A            ；direct←（direct）or（A）
ORL  direct，#data        ；direct←（direct）or  data
```

3．逻辑异或运算指令

```
XRL  A，Rn                ；A←（A）xor（Rn），这里“xor”表示“异或”逻辑操作
XRL  A，direct            ；A←（A）xor（direct）
XRL  A，@Ri               ；A←（A）xor（（Ri））
XRL  A，#data             ；A←（A）xor  data
XRL  direct，A            ；direct←（direct）xor（A）
XRL  direct，#data        ；direct←（direct）xor  data
```

3.3 单片机汇编指令

3.3.3 逻辑运算指令

4. 累加器清0指令

```
CLR   A                    ; A←0   : 将A中8位数据的每一位都清0
```

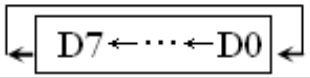
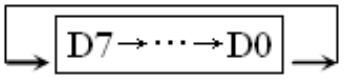
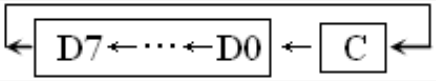
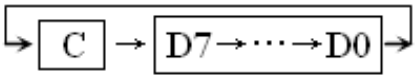
5. 累加器取反指令

```
CPL   A                    ; A←（$\overline{A}$ ）  : 将A中8位数据的每一位都取反
```

3.3 单片机汇编指令

3.3.4 移位指令

表 3-4　移位指令

操　作　码	目标操作数	参与运算的操作数	指令功能说明
RL	A		左循环，不影响进位位 C
RR	A		右循环，不影响进位位 C
RLC	A		左大循环，影响进位位 C
RRC	A		右大循环，影响进位位 C

```
RL   A    ; ACC.（i+1）←ACC.i，ACC.0←ACC.7          ：不带进位的循环左移指令
RR   A    ; ACC.i ACC.（i+1），ACC.7←ACC.0          ：不带进位的循环右移指令
RLC  A    ; ACC.（i+1）←ACC.i，ACC.0←Cy，Cy←ACC.7   ：带进位的循环左移指令
RRC  A    ; ACC.i←ACC.（i+1），ACC.7←Cy，Cy←ACC.0   ：带进位的循环右移指令
```

3.3 单片机汇编指令

3.3.5 控制转移类指令

1. 无条件转移指令

```
LJMP  daar16          ; PC←addr16
AJMP  addr11          ; PC10~0 ←  addr11
SJMP  rel             ; PC←（PC）+rel
JMP  @A+DPTR          ; PC←（A）+（DPTR）
```

2. 条件转移指令

```
JZ    rel    ; 若（A）= 0，则PC←（PC）+2+rel    : 若累加器为0，转移，否则顺序执行
JNZ   rel    ; 若（A）≠0，则PC←（PC）+2+rel    : 若累加器不为0，转移，否则顺序执行
JC    rel    ; 若Cy=1，则PC←（PC）+2+rel       : 若进位C为1，转移，否则顺序执行
JNC   rel    ;  若Cy=0，则PC←（PC）+2+rel      : 若进位C不为1，转移，否则顺序执行
```

3.3 单片机汇编指令

3.3.5 控制转移类指令

3. 位测试转移指令

```
JB   bit,  rel    ; 若（bit）=1，  则PC←（PC）+3+rel : 测位地址中数据为1，转移，否
                                                      则顺序执行
JNB  bit,  rel    ; 若（bit）=0，则PC←（PC）+3+rel  : 测位地址中数据为0，转移，否
                                                      则顺序执行
JBC  bit,  rel    ; 若（bit）=1，则PC←（PC）+3+rel，且bit←0 : 测位地址中数据为1，
                                                    ; 转移，并对该位数据
                                                      清0，否则顺序执行
```

3.3 单片机汇编指令

3.3.5 控制转移类指令

4. 比较转移指令

```
CJNE  A, #data, rel      ; 若（A）≠data，则PC←（PC）+3+rel ：两数不等则转移
CJNE  Rn, #data, rel     ; 若（Rn）≠data，则PC←（PC）+3+rel
CJNE  @Ri, #data, rel    ; 若（（Ri））≠data，则PC←（PC）+3+rel
CJNE  A, direct, rel     ; 若（A）≠（direct），则PC←（PC）+3+rel
```

5. 计数转移指令

```
DJNZ  Rn,  rel       ; Rn←（Rn）-1，若（Rn）≠0，则转移，PC←（PC）+2+rel
DJNZ  direct, rel    ; direct←（direct）-1，若（direct）≠0，则转移，PC←（PC）
                       +3+rel
```

6. 空操作指令

NOP 是单字节空操作指令，PC←（PC）+1，不进行任何操作，可用作短暂延时。

3.3 单片机汇编指令

3.3.6 子程序调用/返回指令

1. 子程序调用指令

```
LCALL  addr16          ;（PC）压栈，PC←addr16 ： addr16是16位子程序入口地址
ACALL  addr11          ;（PC）压栈，PC10～0←addr11
```

SP←（SP）+1，（SP）←（PC）$_{7\sim0}$，SP←（SP）+1，（SP）←（PC）$_{15\sim8}$，PC←addr16

SP←（SP）+1，（SP）←（PC）$_{7\sim0}$，SP←（SP）+1，（SP）←（PC）$_{15\sim8}$，PC←addr11

2. 返回指令

RET　　RETI

$PC_{15\sim8}$←（SP），SP←（SP）−1，$PC_{7\sim0}$←（SP），SP←（SP）−1

3.3 单片机汇编指令

3.3.7 位操作指令

1．位传送指令

```
MOV  C，bit          ；Cy←（bit）：将指定的位地址中的1位数据传送至1位累加器C
MOV  bit，C          ；bit←（Cy）：将1位累加器C中的数据传送至指定的位地址
```

2．位清0指令

```
CLR  C          ；Cy←0      ：对进位位或位累加器C清0
CLR  bit        ；bit←0     ：对指定的位地址中的内容清0，如 CLR  0A4H
```

3．位置1指令

```
SETB   C        ；Cy←1      ：对进位位或位累加器C置1
SETB   bit      ；bit←1     ：对指定的位地址中的内容置1，  如 SETB  P1.0
```

3.3 单片机汇编指令

3.3.7 位操作指令

4. 位取反指令

```
CPL   C              ; 对进位位或位累加器C取反，如 CPL  P3.5
CPL   bit            ; 对指定的位地址中的内容取反，如 CPL  ACC.0
```

5. 按位与指令

```
ANL  C, bit          ; Cy←（Cy）∧（bit）  ：将Cy与位地址中的内容相与后存入Cy
ANL  C, /bit         ; Cy←（Cy）∧（$\overline{bit}$）：将Cy与位地址中的反码相与后存入Cy
```

6. 按位或指令

```
ORL  C, bit          ; Cy←（Cy）∨（bit）  ：将Cy与位地址中的内容相或后存入Cy
ORL  C, /bit         ; Cy←（Cy）∨（$\overline{bit}$）：将Cy与位地址中的反码相或后存入Cy
```

3.4 单片机汇编程序设计

3.4.1 单片机编程语言

1．机器语言

2．汇编语言

3．高级语言

3.4 单片机汇编程序设计

3.4.2 汇编语言伪指令

1．汇编起始地址命令ORG

格式：ORG　16 位地址

2．汇编结束命令END

格式：　[标号：] END [标号]

3．定义字节数据命令DB

格式：[标号：]　DB　8 位数据项或数据项表

4．定义双倍字节数据命令DW

格式：[标号：]　DW　16 位数据项或数据项表

3.4 单片机汇编程序设计

3.4.2 汇编语言伪指令

5．定义存储区命令DS

格式：[标号：] DS 16 位数或表达式

6．赋值命令EQU

格式：符号名 EQU 数据或表达式

7．数据地址赋值命令DATA

格式：符号名 DATA 数据或表达式

8．位定义命令BIT

格式：符号名 BIT 位地址或符号地址

3.4 单片机汇编程序设计

3.4.3 汇编语言程序设计的流程

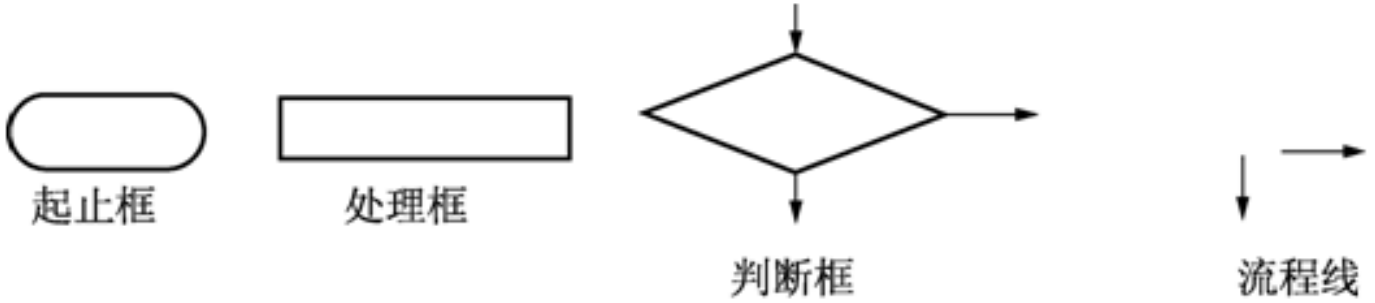


图 3-8　常用的流程图符号

3.5 汇编语言程序的基本结构

3.5.1 顺序程序设计

【程序 3-1】

```
ORG   0000H          ; 程序起始地址设为0000H
MOV   A，30H         ; 将30H中的数据装入A
ADD   A，31H         ; 将A与31H中的数据相加后再装入A
CLR   C              ; 对进位位清0，为减法作准备
SUBB   A，32H        ; 将A中的数据减去32H中的数据并将差装入A
MOV   33H，A         ; 将结果存入33H单元中
END
```

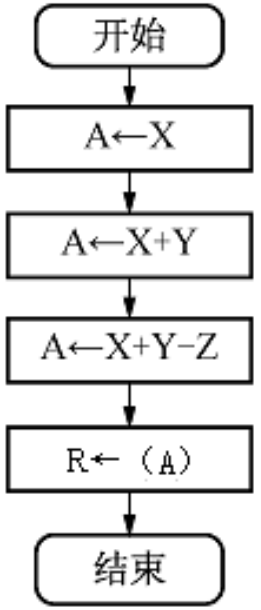


图 3-9　程序流程图

3.5 汇编语言程序的基本结构

3.5.2 分支程序设计

【程序 3-2】

```
        ORG     0200H
START:  MOV     A，40H          ；A←a
        MOV     R0，41H         ；R0←b
        CJNE    A，41H，GP2     ；A不等b转移
GP1:    MOV     50H，A          ；a=b顺序执行
        MOV     51H，R0
        SJMP    GP3             ；比较结束
GP2:    JNC     GP1             ；C=0，则a>b
        MOV     51H，A          ；小数放入51H
        MOV     50H，R0         ；大数放入50H
GP3:    SJMP    GP3
        END
```

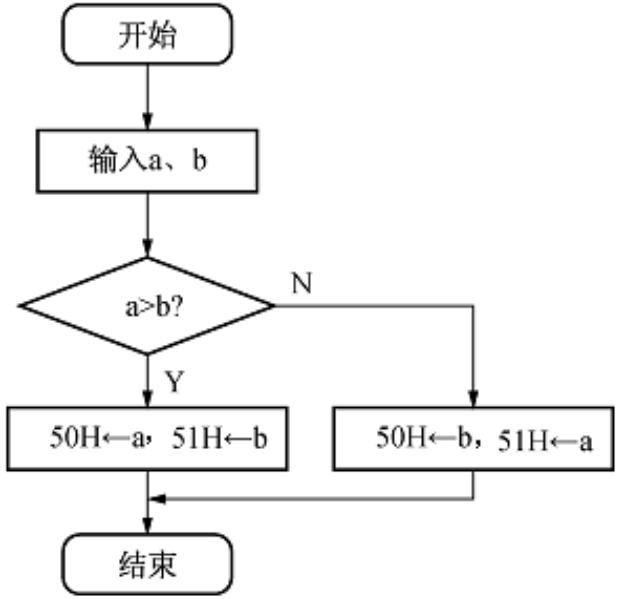


图 3-10　比较两数大小的程序流程图

3.5 汇编语言程序的基本结构

3.5.2 分支程序设计

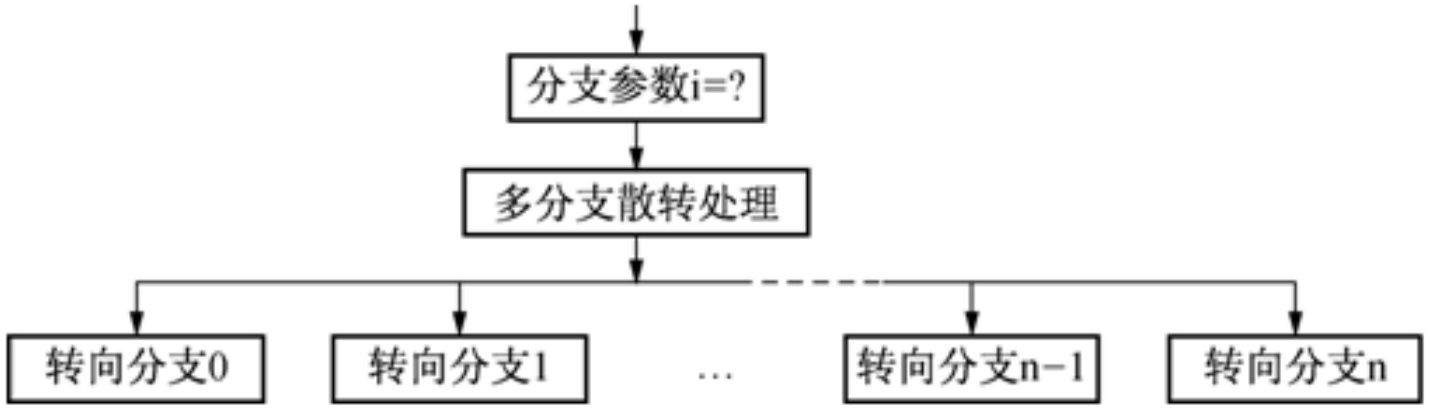


图 3-11　多分支流程图

3.5 汇编语言程序的基本结构

3.5.2 分支程序设计

【程序 3-3】

```
        ORG     0100H
START:  MOV     A,  40H       ; A←X
        JZ      GP2           ; X=0转移
        JNB     ACC.7,  GP1   ; X>0转移
        MOV     A,  #0FFH     ; X<0则Y=-1
        SJMP    GP2
GP1:    MOV     A,  #01H      ; X>0则Y=1
GP2:    MOV     41H,  A       ; 36H←Y
        SJMP    $
        END
```

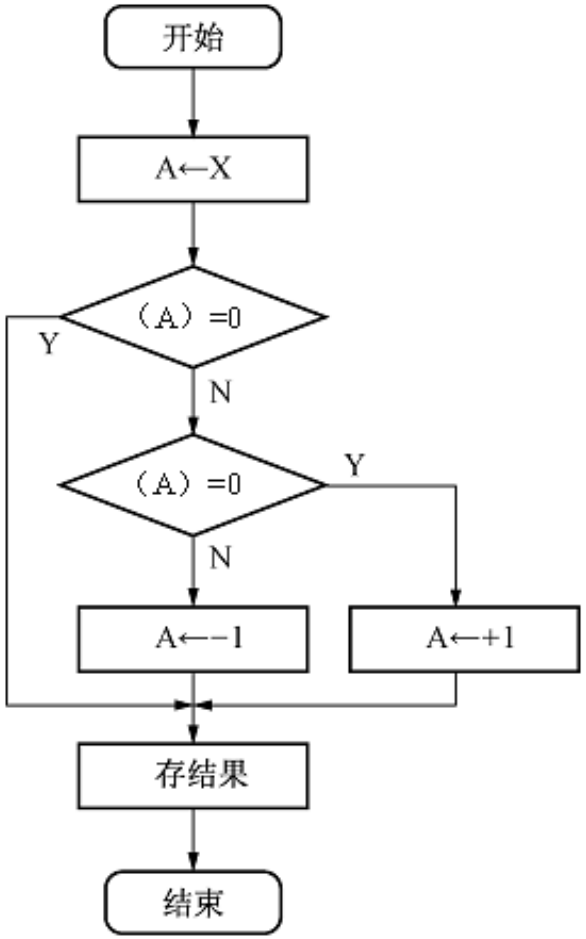


图 3-12　符号函数流程图

3.5 汇编语言程序的基本结构

3.5.2 分支程序设计

【程序 3-4】

```
            ……
    RL      A              ；左移，即把序号乘以2
    MOV     DPTR，  #TAB   ；表首地址装入DPTR
        JMP     @A+DPTR
TAB:    LJMP    PROG0      ；转向分支程序0
        LJMP    PROG1      ；转向分支程序1
                ……
        LJMP    PROGn      ；转向分支程序n
                ……
```

3.5 汇编语言程序的基本结构

3.5.2 分支程序设计

【程序 3-5】

```
        ……
        MOV     A，  # n
        MOV     DPTR，  #TAB  ；DPTR←表首地址
        MOVC    A，  @A+DPTR  ；A←偏移量
        JMP     @A+DPTR       ；转向某一分支程序
TAB:    DB      PROG0         ；偏移量表
        DB      PROG1
        ……
        DB      PROGn
        ……
PROG0:  ……                    ；分支程序0
PROG1:  ……                    ；分支程序1
        ……
PROGn:  ……                    ；分支程序n
```

3.5 汇编语言程序的基本结构

3.5.2 分支程序设计

【程序 3-6】

```
        ……
        MOV     DPTR，  #TAB  ；DPTR←表首地址
        RL      A             ；序号×2
        JNC     HADD          ；序号×2在0～255之间转移
        INC     DPH           ；序号×2大于255，  DPTR+256
HADD:   MOV     R1，A
        MOVC    A，  @A+DPTR  ；查表取高8位地址
        XCH     A，  R1
        INC     A
        MOVC    A，  @A+DPTR  ；查表取低8位地址
        MOV     DPL，  A
        MOV     DPH，  R1
        CLR     A
        JMP     @A+DPTR       ；转向某一分支程序
TAB:    DW      PROG0
        DW      PROG1
        ……
        DW      PROGn
```

3.5 汇编语言程序的基本结构

3.5.3 循环程序设计

【程序 3-7】

```
        ORG     0200H          ; 循环程序起始于0200H
START:  MOV     R0，  #M       ; 将内部RAM中数据块的起始地址装入R0
        MOV     DPTR，  #N     ; 将外部RAM中数据块首地址装入DPTR
        MOV     R1，  #64H     ; 计数值100进入R1
RND:    MOV     A，  @R0       ; 通过R0间址，将待搬运数据装入A
        MOVX    @DPTR，  A     ; 将A中数据传入DPTR指定的地址单元
        INC     R0             ; 内部地址加1
        INC     DPTR           ; 外部地址加1
        DJNZ    R1，  RND      ; （R1）-1≠0则继续循环
HE:     SJMP    HE             ; 搬运结束
        END
```

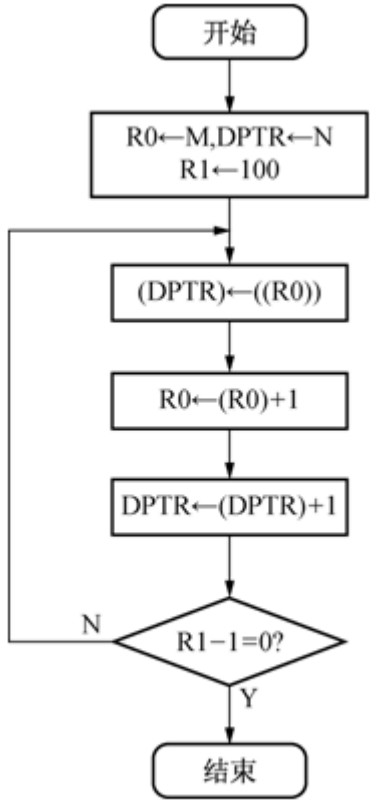


图 3-13　搬运数据块流程图

3.5 汇编语言程序的基本结构

3.5.3 循环程序设计

【程序 3-8】

```
        ORG     0100H
START:  MOV     R0，  #10H
        MOV     R1，  #32H          ；计数值50进入R1
GP1:    CJNE    @R0，  #a，  GP2    ；比较，不相等转移
        MOV     70H，  R0           ；找到，地址送进70H单元
        SJMP    GP3
GP2:    INC     R0                  ；地址加1，继续
        DJNZ    R1，  GP1           ；计数减1=/0则转移
        MOV     70H，  #0FFH        ；未找到，70H←FFH
GP3:    SJMP    GP3
        END                         ；结束
```

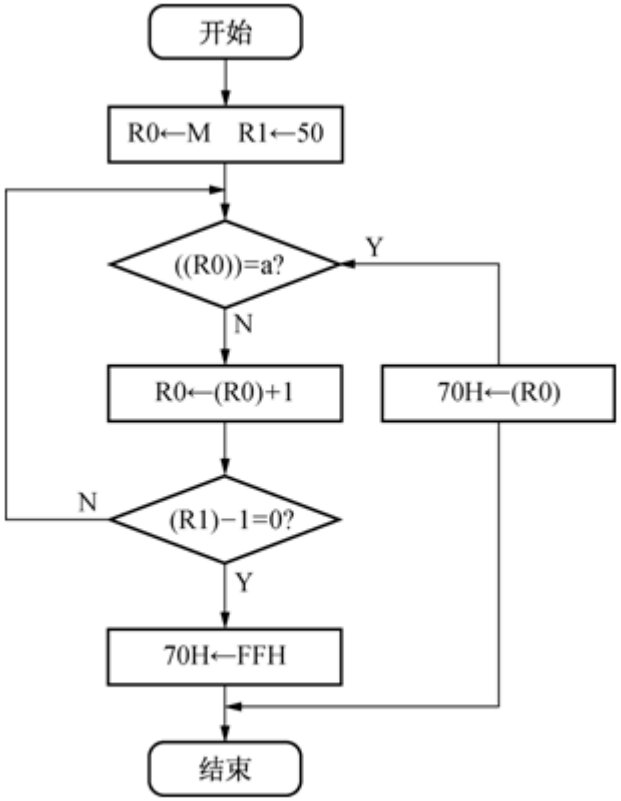


图 3-14　查找程序流程图

3.5 汇编语言程序的基本结构

3.5.4 子程序设计

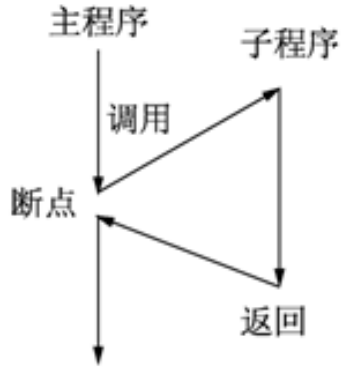


图 3-15　子程序调用

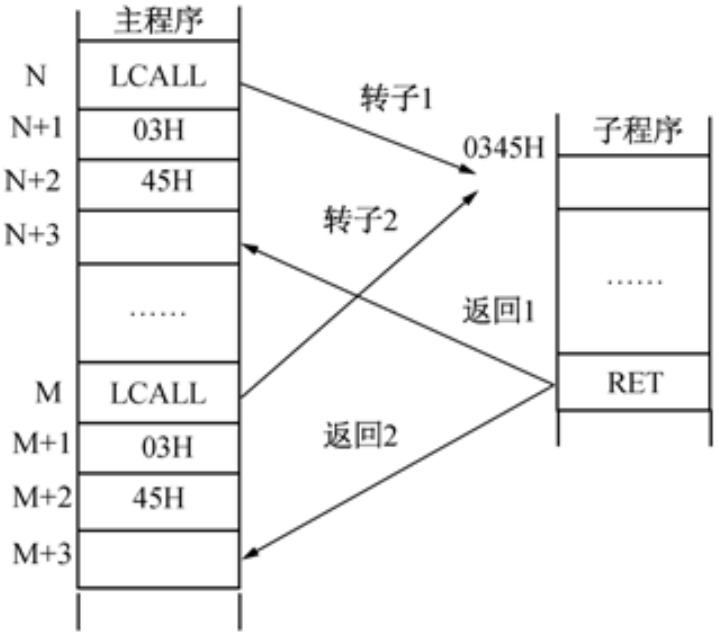


图 3-16　转子与返回示意图

3.5 汇编语言程序的基本结构

3.5.4 子程序设计

【程序 3-9】

```
SUBP:   PUSH    ACC       ; 将A中可能的内容保存好，以便不会影响主程序
        MOV     A，R2     ; 取出待转换数据
        MOV     B，#64H   ; 将100装入寄存器B
        DIV     AB        ; 将待转换数据除以100，结果的百位数进入A
        MOV     @R0，A    ; 存好百位数
        MOV     A，#0AH   ; 将10装入A
        XCH     A，B      ; 余数（B）送A
        DIV     AB        ; 除以10，得十位数和个位数
        SWAP    A         ; 十位数放于高半字节
        ADD     A，B      ; 个位数放于低半字节
        INC     R0        ; 存储指针+1
        MOV     @R0，A    ; 十位和个位存入（R0）+1单元
        POP     ACC       ; 把进入子程序前的A的数据返回A
        RET               ; 返回主程序
```

【程序 3-10】

```
        ORG     0300H
MAIN:   MOV     SP, #70H        ; 设定堆栈指针
        MOV     R1, #31H        ; R1为存放结果的地址指针
        MOV     A, 30H          ; 取出待转换的数据于A
        SWAP    A               ; 高低4位交换，先转换高位字节
        PUSH    ACC             ; 将A的数据压入堆栈
        LCALL   SUBP            ; 调用ASCII码转换子程序，转换低半字节
        POP     ACC             ; 待转换的数据出栈
        MOV     @ R1,  A        ; 存高半字节转换结果
        INC     R1
        PUSH    30H
        LCALL   SUBP            ; 调转换成ASCII码子程序
        POP     ACC
        MOV     @ R1,  A        ; 存低半字节转换结果
        SJMP    $
SUBP:   MOV     R0,  SP
        DEC     R0
        DEC     R0
        XCH     A,  @R0         ; 取被转换数据
        ANL     A,  # 0FH       ; 保留低半字节
        ADD     A,  #02H        ; 修改A
        MOVC    A,  @A+PC       ; 查表
        XCH     A,  @R0         ; 结果送回堆栈
        RET
TAB:    DB      30H, 31H, 32H, 33H…
        END
```

【程序 3-11】

主程序：

```
        ORG     0300H
MAIN:   MOV     SP, #70H        ; 设堆栈指针
        MOV     R1, #30H        ; 取第一数据块首地址送R1中
        LCALL   SUBP            ; 第一次调用求最大值子程序
        MOV     51H, A          ; 第一个数据块的最大值暂存于51H
        MOV     R1, #40H        ; 取第二数据块首地址送R1中
        LCALL   SUBP            ; 第二次调用求最大值子程序
        CJNE    A, 40H, NOE     ; 两个最大值进行比较
NOE:    JNC     ALG             ; C为0，则A大，则转ALG
        MOV     A, 40H          ; C为1，则A小，把40H中内容送入A
ALG:    MOV     50H, A          ; 最后数据存入50H
        SJMP    $
```

子程序：

```
        ORG     0400H
SUBP:   MOV     A, @R1          ; 取数据块长度
        MOV     R2, A           ; R2作计数器
        CLR     A               ; A清0，准备作比较
LP1:    INC     R1              ; 指向下一个数据地址
        CLR     C               ; Cy清0，为减法作准备
        SUBB    A, @R1          ; 用减法作比较
        JNC     LP3         ; 若A大，则转LP3
        MOV     A, @R1      ; A小，则将大数送A中
        SJMP    LP4         ; 无条件转LP4，作循环
LP3:    ADD     A, @R1      ; 恢复A中值
LP4:    DJNZ    R2, LP1     ; 计数器减1，不为0，转继续比较
        RET                 ; 数据比较结束，子程序返回
```