单片机学习资料

一、单片机初学者必看（转载）

很多想学单片机的人问我的第一句话就是怎样才能学好单片机？对于这个问题我今天就我自己是如何开始学单片机，如何开始上手，如何开始熟练这个过程给大家讲讲。

先说说单片机，一般我们现在用的比较多的的 MCS-51 的单片机，它的资料比较多，用的人也很多，市场也很大。就我个人的体会怎么样才能更快的学会单片机这门课。单片机这门课是一项非常重视动手实践的科目，不能总是看书，但是学习它首先必须得看书，因为从书中你需要大概了解一下，单片机的各个功能寄存器，而说明白点，我们使用单片机就是用软件去控制单片机的各个功能寄存器，再说明白点，就是控制单片机那些管脚的电平什么时候输出高，什么时候输出低。由这些高低电平的变化来控制你的系统板，实现我们需要的各个功能。至于看书，只需大概了解单片机各管脚都是干什么的？能实现什么样的功能？第一次，第二次你可能看不明白，但这不要紧，因为还缺少实际的感观认识。所以我总是说，学单片机看书看两三天的就够了，看小说你一天能看五六本，看单片机你两三天看两三遍就够了，可以不用仔细的看。推荐一本书，就这一本就足够，书名是《新编 MCS-51 单片机应用设计》，是哈尔滨工业大学出版社出的，作者是张毅刚。大概了解一下书上的内容，然后实践，这是非常关键的，如果说学单片机你不实践那是不可能学会的，关于实践有两种方法你可以选择，一种方法：你自己花钱买一块单片机的学习板，不要求功

能太全的，对于初学者来说你买功能非常多的那种板子，上面有很多东西你这辈子都用不着，我建议有流水灯、数码管、独立键盘、矩阵键盘、AD 或 DA（原理一样）、液晶、蜂鸣器，这就差不多了。如果上面我提到的这些，你能熟练应用，那可以说对于单片机方面的硬件你已经入门了，剩下的就是自己练习设计电路，不断的积累经验。只要过了第一关，后面的路就好走多了，万事开头难，大家可能都听过。方法二：你身边如果有单片机方面的高手，向他求助，让他帮你搭个简单的最小系统板。对于高手来说，做个单片机的最小系统板只需要一分钟的时间，而对于初学者可就难多了，因为只有对硬件了解了，才能熟练运用。而如果你身边没有这样的高手，又找不到可以帮助你的人，那我劝你最好是自己买上一块，毕竟自己有一块要方便的多，以后做单片机类的小实验时都能用得上，还省事。

有了单片机学习板之后你就要多练习，最好是自己有台电脑，一天少看电影，少打游戏，把学习板和电脑连好，打开调试软件坐在电脑前，先学会怎么用调试软件，然后从最简单的流水灯实验做起，等你能让那八个流水灯按照你的意愿随意流动时你已经入门了，你会发现单片机是多么迷人的东西啊，太好玩了，这不是在学习知识，而是在玩，当你编写的程序按你的意愿实现时你比做什么事都开心，你会上瘾的，真的。做电子类的人真的会上瘾。然后让数码管亮起来，这两项会了后，你已经不能自拔了，你已经开始考虑你这辈子要走哪一行了。就是要这样练习，在写程序的时候你肯定会遇到很多问题，而这时你再去翻书找，或是问别人，当得到解答后你会记住一辈子的，知识必须用于现实生活中，解决实际问题，这样才能发挥它的作用，你自己好好想想，上了这么多年大学，天天上课，你在课堂上学到了什么？是不是为了期末考试而忙碌呢？考完得了 90 分，哈哈哈好高兴

啊，下学期开学回来忘的一干二净，是不是？你学到什么了？但是我告诉你单片机一旦学会，永远不会忘了。另外我再说说用汇编和 C 语言编程的问题。很多同学大一二就开设了 C 语言的课，我也上过，我知道那时天天就是几乘几，几加几啊，求个阶乘啊。学完了有什么用？让你用 C 语言编单片机的程序你是不是就傻了？书上的东西我们必须要会运用。单片机编程用 C 语言或汇编语言都可以，但是我建议用 C 语言比较好，如果原来有 C 语言的基础那学起来会更好，如果没有，也可以边学单片机边学 C 语言，C 语言也挺简单，只是一门工具而已，我劝你最好学会，将来肯定用得着，要不你以后也得学，你一点汇编都不会根本无所谓，但你一点 C 语言都不会那你将来会吃苦头。汇编写程序代码效率高，但相对难度较大，而且很啰嗦，尤其是遇到算法方面的问题时，根本是麻烦的不得了，现在单片机的主频在不断的提高，我们完全不需要那么高效率的代码，因为有高频率的时钟，单片机的 ROM 也在不断的提高，足够装得下你用 C 语言写的任何代码，C 语言的资料又多又好找，将来可移植性非常好，只需要变一个 IO 口写个温度传感器的程序在哪里都能用，所以我劝大家用 C 语言。

总结上面，只要你有信心，做事能坚持到底，有不成功不放弃的强烈意志，那学个单片机来说就是件非常容易的事。

步骤：

1.找本书大概了解一下单片机结构，大概了解就行。不用都看懂，又不让你出书的。（三天）

2.找学习板练习编写程序，学单片机就是练编程序，遇到不会的再问人或查书。　（二十天）

3.自己网上找些小电路类的资料练习设计外围电路。焊好后自己调试，熟悉过程。　　（十天）

4.自己完全设计具有个人风格的电路，产品，。。。你已经是高手了。

看到了吗？下功夫一个多月你就能成为高手，我就讲这么多了，学不学得会，下不下得了功夫就看你的了。

最后我呢再给大家推荐两款我们专门给单片机的初学者设计的单片机学习板，大家可以看看它的资料，需要说明一下，如果使用我的单片机学习板，我可以负责终生的技术支持，直到教会你为止。支持的方式你可以通过 QQ，EMIL 等随时问我关于单片机开发及电路设计方面的问题。

二、单片机指令

3．2 分类指令

在介绍各条分类指令之前，将指令中的操作数及注释中的符号说明如下。

RN：当前指定的工作寄存器组中的 RO-R7(其中 N＝0，1，2，…，7)。

RI：当前指定的工作寄存器组中的 RO，R1(其中 I＝0，1)。

(RI)：RI 间址寻址指定的地址单元。

((RI))：RI 间址寻址指定地址单元中的内容。

DIR：8 位直接字节地址(在片内 RAM 和 SFR 存储空间中)。

#DATA8：8 位立即数。

#DATAL6：16 位立即数。

ADDRL6：16 位地址值。

ADDRLL：11 位地址值。

BIT：位地址(在位地址空间中)。

REL：相对偏移量(一字节补码数)。

下面介绍各条分类指令的主要功能和操作，详细的指令操作说明及机器码形式可见附录。

3．2．1 数据传送与交换类指令

共有 28 条指令，包括以 A，RN，DPTR，直接地址单元，间接地址单元为目的的操作数的指令；访问外部 RAM 的指令；读程序存储器的指

令；数据交换指令以及准栈操作指令。

1. 以 A 为目的操作数

MOV A,
- Rn ;Rn→A
- dir ;(dir)→A
- @Ri ;((Ri))→A
- #data ;#data→A

例 R1＝20H,(20H)＝55H,指令 MOV A,@R1 执行后,A＝55H。

2. 以 Rn 为目的操作数

MOV Rn,
- A ;A→Rn
- dir ;(dir)→Rn
- #data ;#data→Rn

例 (40H)＝30H,指令 MOV R7,40H 执行后,R7＝30H。

3. 以 DPTR 为目的操作数

MOV DPTR,#data16 ; #data16→DPTR

4. 以直接地址为目的操作数

MOV dir,
- A ;A→dir
- Rn ;Rn→dir
- dir′ ;(dir′)→dir
- @Ri ;((Ri))→dir
- #data ;#data→dir

例 R0=50H,(50H)=10H,指令 MOV 35H,@R0 执行后,(35H)=10H。这一操作也可用指令 MOV 35H,50H 来完成。

5. 以间接地址为目的操作数

```
        ┌A       ;A→(Ri)
MOV @Ri,┤dir     ;dir→(Ri)
        └#data   ;#data→(Ri)
```

例 R1=30H,指令 MOV @R1,A 执行后,A→30H。

6. 访问外部数据 RAM

```
MOVX  A,    @DPTR   ;((DPTR))→A
MOVX  A     ,@Ri    ;((P2 Ri))→A
MOVX  @DPTR ,A      ;A→(DPTR)
MOVX  @Ri   ,A      ;A→(P2 Ri)
```

例 1 DPTR=2000H,外部 RAM(2000H)=18H,指令 MOVX A,@DPTR 执行后,A=18H。

例 2 P2=10H,R1=50H,A=64H,指令 MOVX @R1,A 执行后,外部 RAM(1050H)=64H。

7. 读程序存储器

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

例 A=20H, DPTR=2000H,指令 MOVC A,@A+DPTR 执行后,程序存储器 2020H 单元中的内容送入 A。这条指令特别适宜用来查阅在程序 ROM 中已建立的数据表格,是一种常用的查表指令。

9．堆栈操作

PUSH DIR ； SP 十 1-6P，(DIR)一(SP)

POP DIR ； ((SP))一 DIR，SP－1--P ，

例 1 SP＝07H，(35H)＝55H，指令 PUSH 35H 执行后，55H 送入 08H 地址单元，SP=

08H。

例 2 SP＝13H，(13H)= 1FH，指令 POP 25H 执行后，1FH 压入 25H 地址单元，SP 此时为 12H。

综合例 把片内 RAM 中 50H 地址单元中的内容与 40H 地址单元中的内容互换。

方法一(直接地址传送法)：

MOV A ， 50H

```
        MOV  B      ,A
        MOV  A      ,40H
        MOV  50H    ,A
        MOV  A      ,B
        MOV  40H    ,A
        SJMP  $
```

方法二（间接地址传送法）：

```
        MOV  R0     ,40H
        MOV  R1     ,50H
        MOV  A      ,@R0
        MOV  B      ,@R1
        MOV  @R1    ,A
        MOV  @R0    ,B
        SJMP  $
```

方法三（直接地址间相互传送法）：

```
        MOV  R7     ,50H
        MOV  50H    ,40H
        MOV  40H    ,R7
        SJMP  $
```

方法四（字节交换传送法）：

```
        MOV  A      ,50H
        XCH  A      ,40H
        MOV  50H    ,A
        SJMP  $
```

方法五（堆栈传送法）：

```
        PUSH   50H
        PUSH   40H
        POP    50H
        POP    40H
        SJMP   $
```

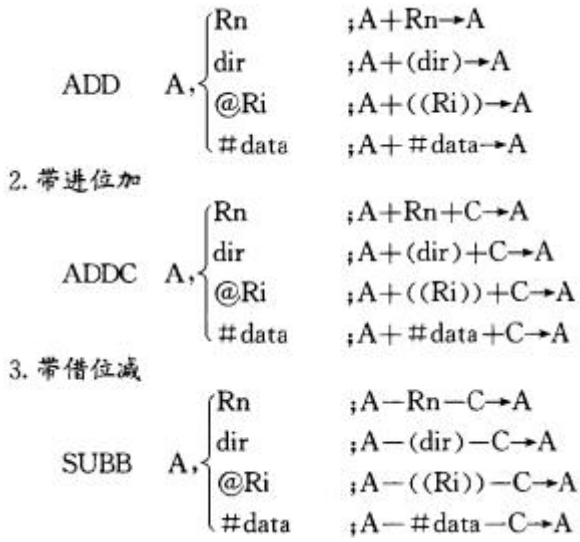
数据传送与交换类指令是各类指令中数量最多、使用最频繁的一类指令，编程时应能十分熟练地灵活运用

3．2．2 算术运算类指令

共有 24 条指令，主要包括加、减、乘、除、增量、减量和十进制调整等指令。其中，大多数指令都同时以 A 为源操作数之一和目的操

作数。

1． 加



说明：借位位来自程序状态字 PSW 中的进位位 C，只是在作减法运算时，被用作借位位。

例 A＝38H，R1＝20H，(20H)＝23H，C＝1 指令 SUBB A，@R1 执行后，A＝14H。

4．乘法

MUL AB ； A×B-BA

说明：本指令实现 8 位无符号乘法。A，B 中各放一个 8 位乘数，指令执行后，16 位积的高位在 B 中，低位在 A 中。

例 A＝50H，B＝40H，指令 MUL AB 执行后，A＝00H，B=32H

5．除法

DIV AB ； A÷ B-商在 A 中，余数在 B 中

说明：本指令实现 8 位无符号除法。A 放被除数，B 放除数，指令执

行后，A 中为商，B 中为 余数。若除数 B＝00H，则指令执行后，溢出标志 OV＝1，且 A，B 内容不变。

例 1 A＝28H，B＝12H，指令 DIV AB 执行后，A＝02H，B＝04H。

例 2 A＝08H，B＝09H，指令 DIV AB 执行后，A＝00H，B＝08H。

6. 增量

```
INC  A      ;A+1→A
     Rn     ;Rn+1→Rn
     dir    ;(dir)+1→dir
     @Ri    ;((Ri))+1→(Ri)
     DPTR   ;DPTR+1→DPTR
```

例 (20H)＝55H，指令 INC 20H 执行后，(20H)＝56H。

7. 减量

```
DEC  A      ;A−1→A
     Rn     ;Rn−1→Rn
     dir    ;(dir)−1→dir
     @Ri    ;((Ri))−1→(Ri)
```

例 R1＝35H，(35H)＝27H，指令 DEC @R1 执行后，(35H)＝26H。

8. 十进制调整

```
DA  A   ;把 A 中按二进制相加后的结果调整成按 BCD 数相加的结果
```

例 $A=56_{BCD}$，$B=67_{BCD}$，C＝0，指令 ADD A，B 和 DA A 执行后，

$A=23_{BCD}$，C＝1。

说明：①以上两条指令完成了 $56_{BCD}+67_{BCD}=123_{BCD}$ 的运算。

②DA A 指令不能对减法结果作正确的 BCD 数调整。

③DA A 指令的操作过程为，若 2 个 BCD 数相加后的和中，低 4 位大于 9，或有半进位，则在低位加 06H；高 4 位大于 9，或有进位，则在高 4 位加 06H；高 4 位与低 4 位都大于 9，或高 4 位有进位，低 4 位大于 9 则分别加 06H，如上例。

```
  01010110  →56BCD
+ 01100111  →67BCD   } ADD  A,B
  10111101
+ 01100110           } DA  A
1 00100011  →123BCD
```

综合例 1 把在 R4 和 R5 中的两字节数取补(高位在 R4 中)

```
CLR C
MOV A , R5
CPL A
INC A
MOV R5, A
```

```
MOV A, R4
CPL A
ADDC A, #00H
MOV R4,A
SJMP ＄
```

综合例 2 把 R7 中的无符号数扩大 10D 倍(设原数小于 25D)

```
MOV A ， R7
MOV B ， #0AH
MUL AB
MOV R7, A
SJMP ＄
```

综合例 3 把 R1R0 和 R3R2 中的 2 个 4 位跃 D 数相加，结果送入 R5R4 中，如有进位则存于进位位 C 中。

```
CLR C
MOV A,R0
ADD A,R2
DA A
MOV R4,A
MOV A,R1
ADDC A,R3
DA A
MOV R5,A
```

SJMP ＄

在 MCS-51 系列单片机的算术运算类指令中，乘除法指令是许多 8 位微处理器和一些 8 位单片机所没有的，执行时间为 4 个机器周期。这种指令对编制比较复杂的运算程序，例如，比例-积分-微分(PID)运算、浮点运算、多字节数乘除运算等是经常要用到的。

3．2．3 逻辑运算与循环类指令

共有 24 条指令。逻辑运算指令主要包括逻辑"与"、"或"、"异或"、求反和清零；循环指令则都是对 A 的大循环操作，包括有左、右方向以及带与不带进位位的不同循环方式。

1.“与”操作

$$\text{ANL}\ \ \text{A},\begin{cases}\text{Rn} & ;\text{A}\wedge\text{Rn}\rightarrow\text{A}\\ \text{dir} & ;\text{A}\wedge(\text{dir})\rightarrow\text{A}\\ @\text{Ri} & ;\text{A}\wedge((\text{Ri}))\rightarrow\text{A}\\ \#\text{data} & ;\text{A}\wedge\#\text{data}\rightarrow\text{A}\end{cases}$$

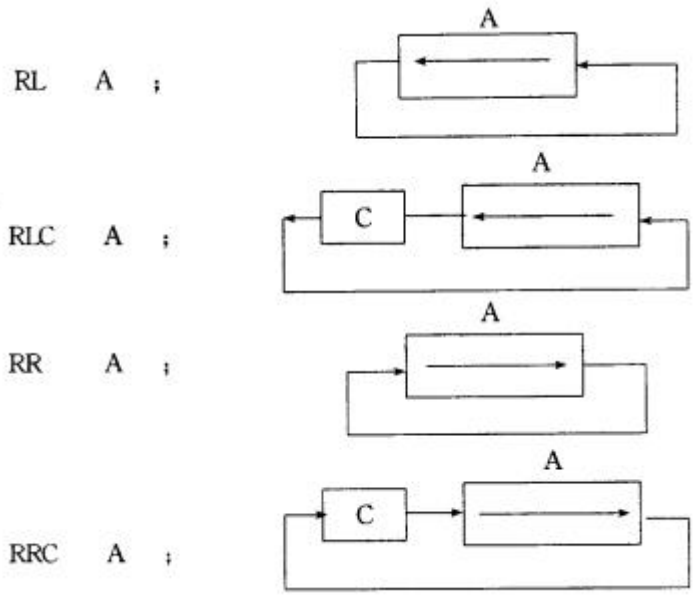
$$\text{ANL}\ \ \text{dir},\begin{cases}\text{A} & ;(\text{dir})\wedge\text{A}\rightarrow\text{dir}\\ \#\text{data} & ;\#\text{data}\wedge(\text{dir})\rightarrow\text{dir}\end{cases}$$

例 A＝05H,(30H)＝16H,指令 ANL A,30H 执行后，A＝04H

2.“或”操作

$$\text{ORL}\ \ \text{A},\begin{cases}\text{Rn} & ;\text{A}\vee\text{Rn}\rightarrow\text{A}\\ \text{dir} & ;\text{A}\vee(\text{dir})\rightarrow\text{A}\\ @\text{Ri} & ;\text{A}\vee((\text{Ri}))\rightarrow\text{A}\\ \#\text{data} & ;\text{A}\vee\#\text{data}\rightarrow\text{A}\end{cases}$$

$$\text{ORL}\ \ \text{dir},\begin{cases}\text{A} & ;(\text{dir})\vee\text{A}\rightarrow\text{dir}\\ \#\text{data} & ;(\text{dir})\vee\#\text{data}\rightarrow\text{dir}\end{cases}$$



例 A=16H，指令 RR A 执行后，A＝0BH。

综合例 把 R2R3 中的 16 位补码数(高位在 R2 中)右移一位，并不改变符号。

```
MOV A ，R2
MOV C ，ACC．7 ；把符号位存入进位位 C
RRC A
MOV R2, A
MOV A, R3
RRC A
MOV R3 , A
SJMP ＄
```

3．2．4 子程序调用与转移类指令

共有 14 条指令。子程序调用类有绝对调用和长调用两种；转移类分

为无条件转移和条件转移两组。无条件转移包括绝对转移、长转移、短转移和间接转移；条件转移包括结果为零、结果为 非零、减一后结果为非零以及两数不相等的转移条件，它们全部采用相对转移的方式。

绝对于程序调用和绝对转移指令的机器码形式比较特殊，操作码不是在前面而是在中间，并且调用和转移的范围都只在 2K 地址范围内，这在使用时应予以注意。

1．绝对调用

ACALL ADDRLL ；ADDRLL 一 PC0-10，PC11-16 不变

说明：①调用范围 本指令在 2K 地址范围内的子程序调用。本指令实现的操作将不改变原 PC 的高 5 位(PC1L-15)，仅把 11 位地址 ADDRLL 送入 PC 的低 11 位(PC0-10)，以此确定子程序的入口地址。由于整个 64K 程序存储器空间被分成 32 个基本 2K 地址范围(见表 2．1)，编程时，必须保证紧接 AC 从 L 指令后面的那一条指令的第一字节与被调用于程序的入口地址在同一 2K 范围内，否则将不能使用 ACALL 指令实现这种调用。

②机器码形式 本指令为二字节指令。设子程序入口地址 ADDRLL 的各位是 A10A9A8A7A6A5A4A3A2A1A。，则 ACALL 指令的二进制机器码为 A10A9A810001A7A6A5A4A3A2A1A0，其中 10001 为 ACALL 指令的操作码。

例 子程序调用指令 ACALL 在程序存储器中的首地址为 0100H，子程序入口地址为 0250H。试确定能否使用 ACALL 指令实现调用?如

果能使用，确定该指令的机器码。

解 因为 ACALL 指令的首地址在 0100H，而 ACALL 是 2 字节指令，所以下一条指令的首地址在 0102H。由表 2．1 可见，0102H 和 0250H 在同一 2K 地址范围内，故可用 ACALL 调用。调用入口地址为 0250H 的 ACALL 指令的机器码形式为：0101000101010000B＝5150H

2，长调用

LCALL ADDRL6 ； ADDRL6 一 PCO-L5

说明：本指令为 64K 程序存储器空间中的全范围子程序调用指令，子程序入口地址可在 64K 地址空间中的任一处。本指令为 3 字节指令。

3．无条件转移指令

(1)绝对转移

AJMP ADDRLL ； ADDRLL 一 PC0-10

说明：①转移范围 本指令为 2K 地址范围内的转移指令。对转移目的地址的要求与 ACALL 指令中对于程序入口地址的要求相同。

②机器码形式 本指令为 2 字节指令。设 ADDRLL 的各位是 A10A9A8A7A6A5A4A3A2A1A0，则指令 AJMP ADDRLL 的二进制机器码为 AL0A9A800001A7A6A5A4A3A2ALA0。

表 2.1　程序存储器空间中的 32 个基本 2K 地址范围

0000H～07FFH	5800H～5FFFH	B000H～B7FFH
0800H～0FFFH	6000H～67FFH	B800H～BFFFH
1000H～17FFH	6800H～6FFFH	C000H～C7FFH
1800H～1FFFH	7000H～77FFH	C800H～CFFFH
2000H～27FFH	7800H～7FFFH	D000H～D7FFH
2800H～2FFFH	8000H～87FFH	D800H～DFFFH
3000H～37FFH	8800H～8FFFH	E000H～E7FFH
3800H～3FFFH	9000H～97FFH	E800H～EFFFH
4000H～47FFH	9800H～9FFFH	F000H～F7FFH
4800H～4FFFH	A000H～A7FFH	F800H～FFFFH
5000H～57FFH	A800H～AFFFH	

例　绝对转移指令 AJMP 在程序存储器中的首地址为 2500H，要求转移到 2250H 地址处执行程序，试确定能否使用 AJMP 指令实现转移?如能实现，其指令的机器码形式是什么?

解　因为 AJMP 指令的首址为 2500H，其下一条指令的首址为 2502H，由表 2．1 可见，2502H 与转移目的地址 2250H 在同一 2K 地址范围内，故可用 AJMP 指令实现程序的转移。指令的机器码：

0100000L01010000B＝4150H

(2)长转移

LJMP ADDRL6 ； ADDRL6 一 PC0-15

说明：本指令为 64K 程序存储器空间的全范围转移指令。转移地址可为 16 位地址值中的 任一值。本指令为 3 字节指令。

(3)短转移

SJMP REL ； PC 十 2 十 REL-PC

说明：本指令为一页地址范围内的相对转移指令。因为肥 L 为 L 字节补码(偏移量)，且 SJMP REL 指令为 2 字节指令，所以转移范围为一 126D 一十 129D。

(4)间接转移

JMP ＠A 十 DPTR ； A 十 DPTR-PC

例 1 A＝02H，DPTR＝2000H，指令 JMP ＠A 十 DPTR 执行后，PC＝2002H。也就是说，程序转移到 2002H 地址单元去执行。

例 2 现有一段程序：

MOV DPTR ，#TABLE

JMP ＠A 十 DPTR

TABLE:

AJMP ROUT0

AJMP ROUTL

AJMP ROUT2

： ：

AJMP ROUTN

根据 JMP ＠A 十 DPTR 指令的操作可知，当 A＝00H 时，程序转入到地址 ROUT0 处执行；当 A＝02H 时，转到 ROUTL 处执行……。可见这是一段多路转移程序，进入的路数由 A 确定。因为 AJMP 指令是 2 字节指令，所以要求 A 必定为偶数。

4．条件转移指令

(1)累加器为零(非零)转移

JZ REL ； A＝0 则转移(PC 十 2 十 REL 一 PC)；

A≠0 程序顺序执行

JNZ REL ； A≠0 则转移（PC+2+REL-PC）；

A=0 程序顺序执行

(2)减一非零转移

DJNZ RN，REL; ；RN 一 1-RN，RN≠ O，则转移(PC 十 2 十 RE- PC);

RN＝0，程序顺序执行

DJNZ DIR，REL；(DIR)一 L-DIR，(DIR)≠0 则转移(PC 十 3 十 REL-PC)；

(DIR)＝O，程序顺序执行

说明：UNZ RN，REL 是 2 字节指令，而 DJNZ DIR，REL 是 3 字节指令，所以在满足转移的条件后，前者是 PC 十 2 十 REL 一 PC，而后者是 PC 十 3 十 REL 一 PC。

例 试说明下列一段程序运行后 A 中的结果。

MOV 23H，＃ 0AH

```
        CLR  A
LOOP:   ADD  A,    23H
        DJNZ 23H,  LOOP
        SJMP $
```

根据程序可知,运算结果 A ＝10+9+8+7+6+5+4+3+2+1

＝55D＝37H

(3)两数不等转移

```
CJNE  A,dir, rel
    ;A≠(dir),则转移(PC+3+rel→PC);A＝(dir),程序顺序执行
CJNE  A,#data, rel     ;A≠#data,则转移(PC+3+rel→PC);
                        A＝data,程序顺序执行
CJNE  Rn, #data,rel    ;Rn≠#data,则转移(PC+3+rel→PC);
                        Rn＝#data,程序顺序执行
CJNE  @Ri,#data, rel   ;((Ri))≠#data,则转移(PC+3+rel→PC);
                        ((Ri))＝#data,程序顺序执行
```

说明：①CJNE 指令都是 3 字节指令。

②若第一操作数大于或等于第二操作数，则影响标志 C＝0(如指令 CJNE A，DIR，REL 中 A>=(DIR)等)；若第一操作数小于第二操作数，则 C＝L(如指令 CJNE A，DIR，REL 中 A＜(D5R)等)。利用对 C 的判断，可使这几条指令实现两操作数相等与否的判断，还可完成两数大小的比较。

例 1 R7＝56H，指令 CJNE R7，#34H，＄十 08H 执行后，程序转移到放本条 CJNE 指令的首地址(＄)加 08H 后的地址单元去执行。

例 2 安排程序，要求读 PL 端口上的信息，若不为 55H 则程序停着等待，只有到 P1 端口为 55H 时，程序往下顺序执行。

程序为：MOV A， #55H

CJNE A， P1，＄ '

5．相对偏移量 REL 的求法

在短转移和条件转移中，用偏移量 REL 和转移指令所处的地址值来计算转移的目的地址。REL 是 1 字节补码值，如 REL 是正数的补码，程序往前转移；如 REL 是负数的补码，程序往回转移。下面介绍计算 REL 大小的方法。

设本条转移指令的首地址为 AD--源地址，字节数为 BN-2 字节或 3 字节，要转移到的地址为 AD--目的地址，这三者之间的关系为：

AD＝AS 十 BN 十 REL 补

于是 REI＝(AD-AS 一 8N)补

这就是在已知源地址，目的地址和指令的长度时，计算 REL 大小的

公式。

例 1 一条短转移指令 SJMP REL 的首地址为 2010H 单元。求：①转移到目的地址为 2020H 单元的 REL；②求转移到目的地址为 2000H 单元的 REL。

解 SJMP REL 是 2 字节指令，BN＝2；由题意知，AS＝2010H，所以

① $rel=(2020H-2010H-2)_{补}=(0EH)_{补}$
$=0EH$

② $rel=(2000H-2010H-2)_{补}=(-12H)_{补}$
$=EEH$

在①中，rel 是正数的补码，所以实现了向前(地址增大)方向的转移。在②中，rel 是负数的补码，所以实现了向回(地址减小)方向的转移。

例 2 MCS-51 单片机指令系统中，没有停机指令，通常用短转移指令 SJMP $ ($ 为本条指令的首地址)来实现动态停机的操作。试确定这条指令中相对偏移量的大小。

解 据题意，$a_s=\$$，$B_n=2$，$a_d=\$$

所以，$rel=(\$-\$-2)_{补}=(-2)_{补}=FEH$

顺便指出，因为 SJMP 指令的操作码为 80H，所以 SJMP $ 指令的机器码是 80FEH。

例 3 计算下面程序中 CJNE 指令的偏移量。

```
LOOP:  MOV   A,P1
       CJNE  A,#55H,LOOP
```

解 由于 MOV A,P1 是 2 字节指令，故 CJNE A,#55H,LOOP 指令的首地址是 LOOP+2。又因为 CJNE A,#55H,LOOP 是 3 字节指令，于是有

$a_d=LOOP$，$a_s=LOOP+2$，$B_n=3$

$rel=[LOOP-(LOOP+2)-3]_{补}=(-5)_{补}$

$=FBH$

整条指令的机器码为 B455FBH。

综合例 1 设单片机的晶振频率为 6MHz，编写一段延时约 100ms 的子程序。

```
Delay:  MOV  R7   ,#64H
LOOP:   MOV  R6   ,#FAH   ;循环 250D 次
        DJNZ R6   ,$      ;本条指令执行时间为 4μs
        DJNZ R7   ,LOOP
        RET
```

综合例 2 试编写在单片机中使用的电子钟中断记时程序。设每隔一秒种进入一次本程序，其中，R7 存放小时值，R6 存放分钟值，R5 存放秒钟值。

流程图：如图 2.10 所示。

```
START:MOV   A   ,R5
      ADD   A   ,#01H
      DA    A
      MOV   R5  ,A
      CJNE  A   ,#60H,NEXT
      MOV   R5  ,#00H
      MOV   A   ,R6
      ADD   A   ,#01H
      DA    A
      MOV   R6  ,A


      CJNE  A   ,#60H,NEXT
      MOV   R6  ,#00H
      MOV   A   ,R7
      ADD   A   ,#01H
      DA    A
      CJNE  A   ,#24H,NEXT
      MOV   R7  ,#00H
NEXT: RETI
```

由于程序中的时、分、秒数是已经作过十进制调正后的 BCD 数，因此＃60H，＃60H，＃24H 虽以十六进制数出现，但却表示 BCD 数。在于程序调用与转移指令中，由于绝对转移和绝对调用指令 AJMP 和 ACALL 指令字节少，转移范围大，因而是常用的指令。但使用时应注意其机器码形式及允许使用的条件。相对转移类指令因本身长度有 2 字节和 3 字节之分，这会影响到偏移量大小的计算，因而也要十分注意。此外，条件转移指令中，由于没有结果为正和结果为负等转移条件的指令，因而这些转移的要求，只能由 CJNE 指令加上对进位位的判断来实现。

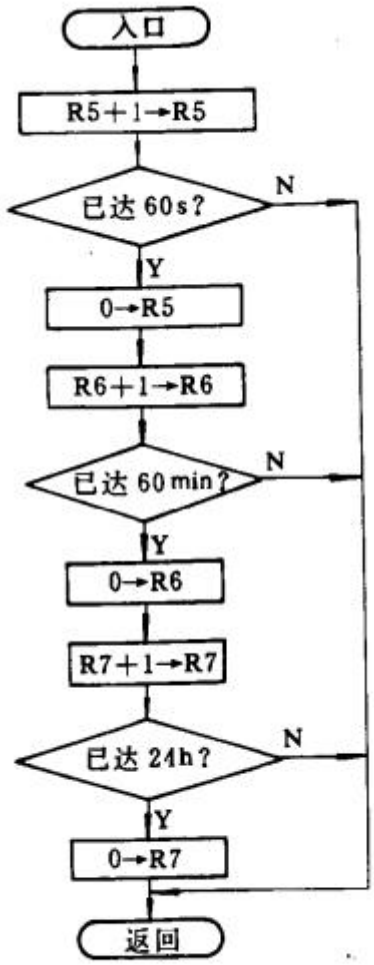


图 2.10　电子钟记时程序流程图

3．2．5 位操作类指令

共有 17 条指令。其共同特点是对进位位 C 和直接位地址 NT 的操作。其中包括清零、置 1、求反、逻辑与、逻辑或、传送以及判断转移。MCS-51 单片机中这些丰富的位操作指令表现出其具有优异的布尔处理能力

1. 清位

```
CLR    C        ;0→C
CLR    bit      ;0→bit
```

2. 置位

```
SETB   C        ;1→C
SETB   bit      ;1→bit
```

3. 位求反

```
CPL    C        ;$\overline{C}$→C
CPL    bit      ;($\overline{bit}$)→bit
```

4. 位与

```
ANL    C,bit    ;C∧(bit)→C
ANL    C,bit    ;C∧($\overline{bit}$)→C
```

5. 位或

```
ORL    C,bit    ;C∨(bit)→C
ORL    C,bit    ;C∨($\overline{bit}$)→C
```

6. 位传送

```
MOV    C,bit    ;(bit)→C
MOV    bit,C    ;C→bit
```

7. 位转移

```
JC     rel          ;C=1,则转移(PC+2+rel→PC);否则程序顺序执行
JNC    rel          ;C=0,则转移(PC+2+rel→PC);否则程序顺序执行
JB     bit, rel     ;(bit)=1,则转移(PC+3+rel→PC);否则程序顺序执行
JNB    bit, rel     ;(bit)=0,则转移(PC+3+rel→PC);否则程序顺序执行
JBC    bit, rel     ;(bit)=1,则转移(PC+3+rel→PC),且该位清零;否则程序顺
                     序执行
```

例 在内部 RAM 40H 和 41H 地址单元中，有 2 个无符号数，试编写一段程序比较其大小，将大数存于内部 RAM 中 GREAT 地址单元，小数存于 LESS 地址单元，如两数相等，则分别送入这 2 个单元。

程序如下：

```
        MOV    A       ,40H
        CJNE   A       ,41H,NOTEQ ;两数不等则转 NOTEQ
        MOV    GREAT   ,A
        MOV    LESS    ,A
        SJMP   $
NOTEQ:  JC     ALIT                ;A 小则转 ALIT
        MOV    GREAT   ,A
        MOV    LESS    ,41H
        SJMP   $
ALIT:   MOV    LESS    ,A
        MOV    GREAT   ,41H
        SJMP   $
```

2.2.6 CPU 控制类指令

共有 3 条指令，包括返回指令和空操作指令。

①RET　　；从调用子程序返回，与 LCALL 或 ACALL 指令配合使用

②RETI　；从中断服务程序中返回

③NOP　　；空操作

综合例　R3 和 R2 中分别放有 2 个 8 位补码数，试编制一段程序将 R3 与 R2 相乘，并把结果送入 R5、R4 中。

流程框图：见图 2.11。

程序如下：

```
MULS:   MOV     A       ,R3
        MOV     C       ,ACC.7
        JNC     NEXT1
        CPL     A
        INC     A
NEXT1:  MOV     B       ,A

        MOV     A       ,R2
        MOV     C       ,ACC.7
        JNC     NEXT2
        CPL     A
        INC     A
NEXT2:  MUL     AB
        MOV     R5      ,B
        MOV     R4      ,A
        MOV     A       ,R3
        XRL     A       ,R2
        JNB     ACC.7   ,NEXT3
        MOV     A       ,R4
        CLR     C
        CPL     A
        INC     A
        MOV     R4      ,A
        MOV     A       ,R5
        CPL     A
        ADDC    A       ,#00H
        MOV     R5      ,A
NEXT3:  SJMP    $
```

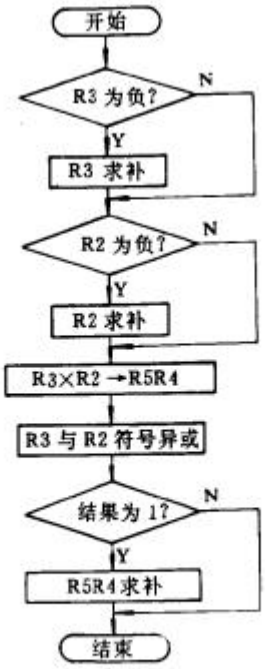


图 2.11　2 个 8 位补码数的乘法程序框图