



第4章 STC单片机CPU指令系统

何宾
2015.02

本章主要内容

- STC单片机CPU寻址模式
- STC单片机 CPU指令集

STC单片机CPU寻址模式

一条机器指令包含两部分，即：操作码和操作数。

- **操作码的目的是要对被操作对象进行处理。**

- **典型地，对被操作对象实现逻辑与或非运算、加减乘除运算等。**

- **在机器/汇编语言指令中，将操作对象称为操作数。**

STC单片机CPU寻址模式

在STC 8051单片机中，这些被操作的对象（操作数）可以保存在下面：

- CPU的内部寄存器
- 片内Flash程序存储器
- 片内RAM
- 片内扩展RAM
- 片外存储器
- 也可能仅是一个常数（它作为操作码的一部分）。

STC单片机CPU寻址模式

因此，就需要预先确定一些规则：

- 一方面使得操作数可以保存在这些区域内；
- 另一方面，CPU可以找到它们。

在STC 8051单片机中，将CPU寻找操作对象（操作数）所保存位置的方式，称为寻址模式。

STC单片机CPU寻址模式

在8051单片机常用的寄存器符号有：

- A：表示8051的累加器ACC。
- DPTR：表示16位的数据指针，指向外部数据空间或代码存储空间。
- PC：表示16位的程序计数器，指向下一条将要执行指令的地址。
- C：表示进位标志CY。
- AB：表示A和B寄存器对，用于乘和除操作。
- R0-R7：表示当前所使用寄存器组内的8位通用寄存器。
- SP：表示堆栈指针。
- DPS：数据指针选择寄存器。

注：特殊汇编器符号用来表示8051 CPU的内部功能寄存器，不可以修改这些符号。

STC单片机CPU寻址模式

STC15系列单片机采用的是8051 CPU内核，所以其寻址模式和传统的8051单片机是一样的。寻址模式包括：

- 立即数寻址
- 直接寻址
- 间接寻址
- 寄存器寻址
- 相对寻址
- 变址寻址
- 位寻址

STC单片机CPU寻址模式

--立即数寻址模式

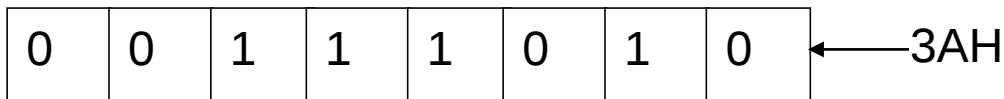
一些指令直接加载常数的值，而不是地址。比如指令：

MOV A, #3AH

■ 功能

□ 将8位的十六进制立即数3A送给累加器。

ACC累加器



STC单片机CPU寻址模式

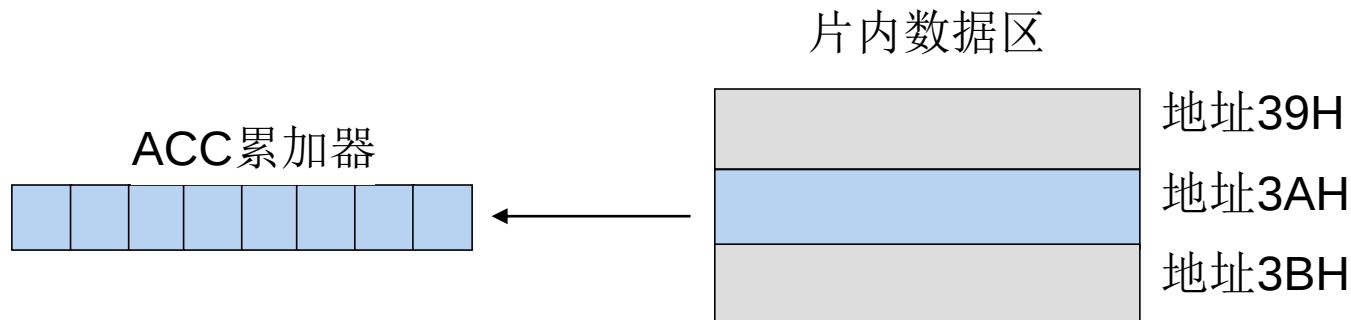
--直接寻址模式

操作数由一个直接8位地址域指定。

- 当使用这种模式时，只能访问片内RAM和特殊功能寄存器SFR。
- 比如指令：

MOV A, 3AH

将片内RAM中地址为3AH单元的数据送给累加器A。



STC单片机CPU寻址模式

对比记忆：

■ 立即数寻址模式

MOV A, #3AH

■ 直接寻址模式

MOV A, 3AH

注：（1）如果操作数前带“#”符号，则操作数表示的是一个立即数，是立即数寻址方式。

（2）如果操作数前面不带“#”符号，则操作数表示的是存储器的地址，3A是存储器的地址，表示把存储器地址为3A单元的内容送到累加器A中。

STC单片机CPU寻址模式

--间接寻址模式

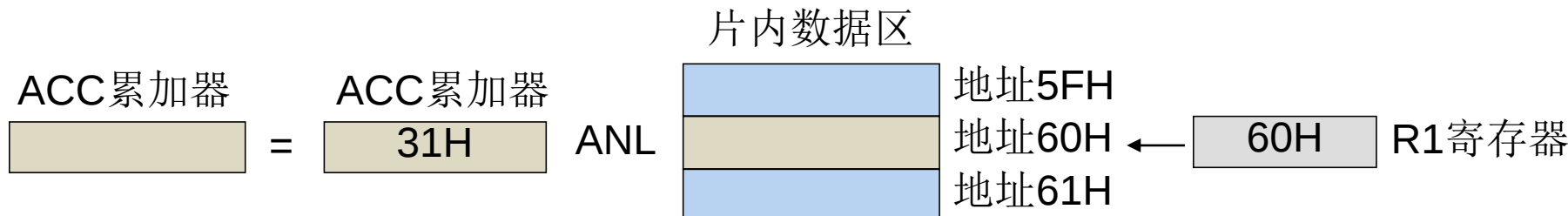
由指令指定一个寄存器，该寄存器包含操作数的地址。

■ 比如指令：

ANL A,@R1

假设：

累加器A中的内容为31H R1寄存器的内容为60H，即（R1）=60H，则以60H作为存储器的地址，将60H地址单元的内容与累加器A中的数31H进行逻辑“与”运算，运算结果存放在累加器A中。



STC单片机CPU寻址模式

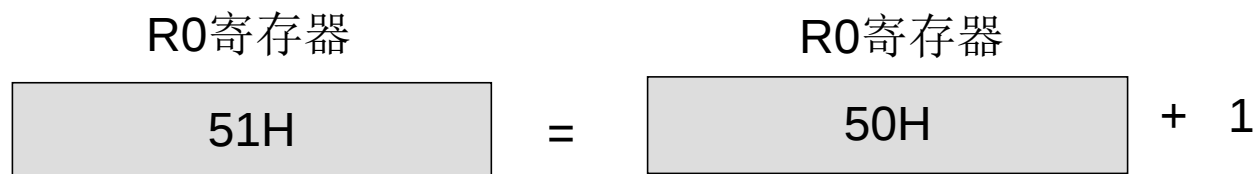
--寄存器寻址模式

某些特定指令用来访问寄存器组中的R0-R7寄存器、累加器A、通用寄存器B、地址寄存器和进位CY。由于这些指令不需要地址域，因此这些指令访问效率更高。

■ 比如指令：

INC R0

功能：将寄存器R0的内容加1，再送回R0。假设当前寄存器R0中的内容为50H。

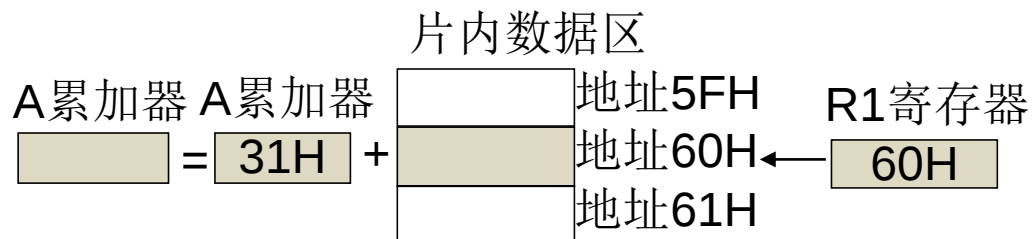


STC单片机CPU寻址模式

对比记忆

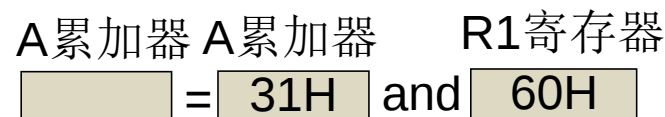
■ 间接寻址模式

ANL A,@R1



■ 寄存器寻址模式

ANL A,R1



注：间接寻址是把R1寄存器中的内容60H作为地址，对60H地址中的内容进行操作。而寄存器寻址是直接对寄存器中的内容进行操作。

STC单片机CPU寻址模式

--相对寻址模式

相对寻址时将程序计数器PC中的当前值与指令第二个字节给出的数相加，其结果作为转移指令的目的转移地址。

- PC中的当前值为基地址，指令第二个字节给出的数作为偏移量。

偏移量为带符号的数，范围为-128~+127。

- 由于目的地址是相对于PC中的基地址而言，所以这种寻址方式称为相对寻址。
- 这种寻址方式主要用于跳转指令，比如指令：

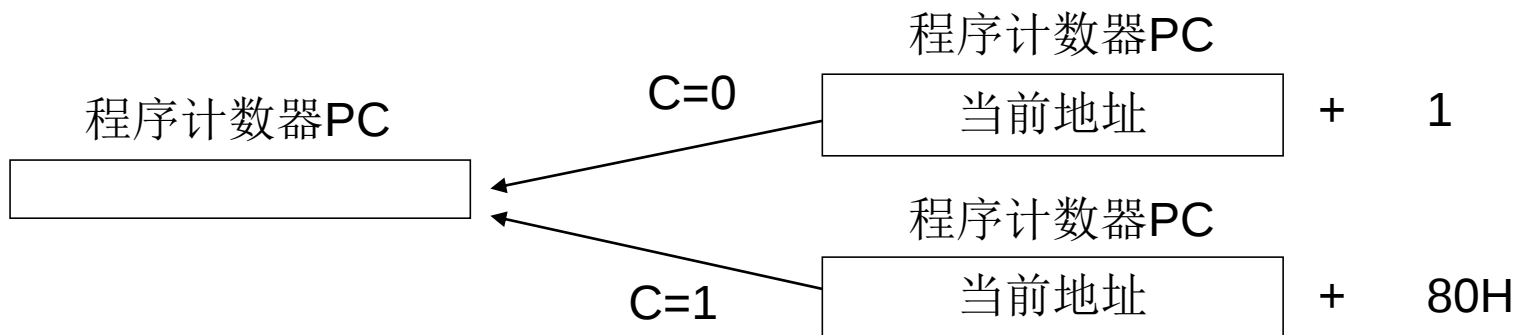
JC 80H

功能：当进位标志为1时，则进行跳转。

如需原始PPT文件请点击此处
<http://www.gpnewtech.com/ppt>

STC单片机CPU寻址模式

--相对寻址模式



STC单片机CPU寻址模式

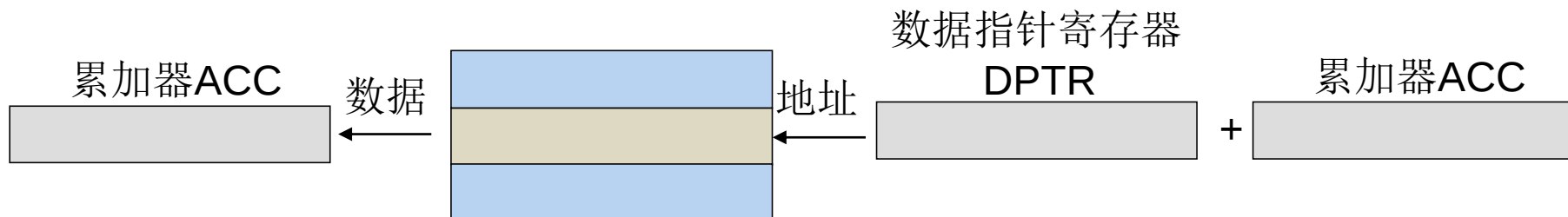
--变址寻址模式

这种模式使用数据指针作为基地址，累加器值作为偏移地址来读取程序Flash存储器。

■ 比如指令：

MOVC A,@A+DPTR

功能：将DPTR和A的内容相加所得到的值作为程序存储器的地址，并将该地址单元的内容送累计累加器A。



STC单片机CPU寻址模式

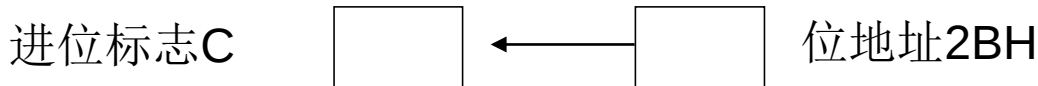
--位寻址模式

位寻址是对一些内部数据存储器RAM和特殊功能寄存器SFR进行位操作时的寻址模式。

- 在进行位操作时，指令操作数直接给出该位的地址，然后根据操作码的类型对该位进行操作。
- 在这种模式下，操作数是256比特中的某一位。
- 比如指令：

MOV C , 2BH

功能：把位寻址区的2BH位状态送进位位C。



STC单片机 CPU指令集



STC15系列单片机内的8051 CPU指令集包含111条指令，
这些指令与传统的8051指令完全兼容，但是大幅度提高了执行指令的时间效率。

■ STC15单片机内8051 CPU指令集分为：

- 算术运算指令
- 逻辑指令
- 数据传输指令
- 布尔指令
- 程序分支指令

算术指令

--加法指令

ADD A,Rn

- 该指令将寄存器Rn的内容和累加器A的内容相加，结果保存在累加器A中。并设置CY标志、AC标志，以及溢出标志。
 - 当和的第3位和第7位有进位时，分别将AC和CY标志置位，否则置0。
 - 对于带符号运算数，当和的第7位与第6位中有一位进位，而另一位不产生进位时，溢出标志OV置位，否则清0。
 - 也可以这样说，当两个正数相加时，相加的结果为负数；或当两个负数相加时，相加的结果为正数时，在这两种情况下设置OV为1。

算术指令

--加法指令

ADD A,Rn指令的内容

助记符	操作	标志	机器码	字节数	周期数
ADD A,Rn	$(PC) \leftarrow (PC) + 1$ $(A) \leftarrow (A) + (Rn)$	CY,AC,O V	00101rrrr	1	1

注：rrr为寄存器的编号，因此机器码范围是28H~2FH。

算术指令

--加法指令

【例】假设累加器A中的数据为C3H，R0寄存器中的数据为AAH。当执行指令：

ADD A, R0

结果：

(A) = 6DH, (AC) = 0, (CY) = 1, (OV) = 1

计算过程为：

$$\begin{array}{r} 1100,0011 \\ + 1010,1010 \\ \hline 1,0110,1101 \end{array}$$

算术指令

--加法指令

ADD A,direct

- 该指令将直接寻址单元的内容和累加器A的内容相加，结果保存在累加器A中。CY,AC,OV标志的设置同上。

ADD A,direct指令的内容

助记符	操作	标志	操作码	字节数	周期数
ADD A,direct	$(PC) \leftarrow (PC) + 2$ $(A) \leftarrow (A) + (\text{direct})$	CY,AC,OV	00000101	2	2

注：在操作码后面跟着一个字节的直接地址。

算术指令

--加法指令

ADD A,@Ri

- 该指令将间接寻址单元的内容和累加器A的内容相加，结果保存在累加器A中。CY,AC,OV标志的设置同上。

ADD A,@Ri指令的内容

助记符	操作	标志	操作码	字节数	周期数
ADD A,@Ri	$(PC) \leftarrow (PC) + 1$ $(A) \leftarrow (A) + ((Ri))$	CY,AC,OV	0010011i	1	2

注：i表示R0或者R1。当i=0时，表示R0寄存器；当i=1时，表示R1寄存器。

算术指令

--加法指令

ADD A,#data

- 该指令将一个立即数和累加器A的内容相加，结果保存在累加器A中。CY,AC,OV标志的设置同上。

ADD A,#data指令的内容

助记符	操作	标志	操作码	字节数	周期数
ADD A,#data	$(PC) \leftarrow (PC) + 2$ $(A) \leftarrow (A) + \text{data}$	CY,AC,OV	00100100	2	2

注：在操作码后面跟着一个字节的立即数。

算术指令

--加法指令

ADDC A,Rn

- 该指令将寄存器Rn的内容与累加器A的内容及进位标志CY的内容相加，结果保存在累加器A中。CY,AC,OV标志的设置同上。

ADDC A,Rn指令的内容

助记符	操作	标志	操作码	字节数	周期数
ADDC A,Rn	$(PC) \leftarrow (PC) + 1$ $(A) \leftarrow (A) + (C) + (Rn)$	CY,AC,OV	00111rrr	1	1

注：rrr为寄存器的编号，因此机器码范围是38H~3FH。

算术指令

--加法指令

【例】假设累加器A中的数据为C3H，R0寄存器中的数据为AAH，进位标志为1时，当执行指令：

ADDC A,R0

结果：

$(A) = 6EH$ ， $(AC) = 0$ ， $(CY) = 1$ ， $(OV) = 1$

计算过程为：

$$\begin{array}{r} 1100,0011 \\ 1010,1010 \\ + \quad \quad 1 \\ \hline 10110,1110 \end{array}$$

算术指令

--加法指令

ADDC A,direct

- 该指令将直接寻址单元的内容与累加器A的内容及进位标志CY中的内容相加，结果保存在累加器A中。CY,AC,OV标志的设置同上。

ADDC A,direct 指令的内容

助记符	操作	标志	操作码	字节数	周期数
ADDC A,direct	$(PC) \leftarrow (PC) + 2$ $(A) \leftarrow (A) + (C) + (\text{direct})$	CY,AC,OV	00110101	2	2

注：在操作码后面跟着一个字节的直接地址。

算术指令

--加法指令

ADDC A,@Ri

- 该指令将间接寻址单元的内容与累加器A的内容及进位标志CY中的内容相加，结果保存在累加器A中。CY,AC,OV标志的设置同上。

ADDC A,@Ri 指令的内容

助记符	操作	标志	操作码	字节数	周期数
ADDC A,@Ri	$(PC) \leftarrow (PC) + 1$ $(A) \leftarrow (A) + (C) + ((Ri))$	CY,AC,OV	0011011i	1	2

注：i表示R0或者R1。当i=0时，表示R0寄存器；当i=1时，表示R1寄存器。

算术指令

--加法指令

ADDC A,#data

- 该指令将一个立即数与累加器A的内容及进位标志CY中的内容相加, 结果保存在ACC中。CY,AC,OV标志的设置同上。

ADDC A,#data 指令的内容

助记符	操作	标志	操作码	字节数	周期数
ADDC A,#data	$(PC) \leftarrow (PC) + 2$ $(A) \leftarrow (A) + (C) + \text{data}$	CY,AC,OV	00110100	2	2

注：在操作码后面跟着一个字节的立即数。

算术指令

--减法指令

SUBB A,Rn

- 该指令从累加器A中减去寄存器Rn和进位标志CY内的内容，将结果保存在累加器A中。

SUBB A,Rn 指令的内容

助记符	操作	标志	操作码	字节数	周期数
SUBB A,Rn	$(PC) \leftarrow (PC) + 1$ $(A) \leftarrow (A) - (C) - (Rn)$	CY,AC,OV	10011rrr	1	1

注：rrr为寄存器的编号，因此机器码范围是98H~9FH。

算术指令

--减法指令

- 如果第7位需要一个借位，则设置进位（借位）标志；否则，清除CY标志。
- 如果第3位需要一个借位，则设置AC标志；否则，清除AC标志。
- 如果第6位需要借位，而7位没有借位时；或者第7位有借位，而第6位没有借位时，在这两种情况下都会设置OV标志。
- 或者可以这样说，当减去有符号的整数时，当一个正数减去一个负数，产生一个负数结果时；或者一个负数减去一个正数时，产生一个正数结果时，设置OV标志。

算术指令

--减法指令

【例】假设累加器A中的数据为C9H，R2寄存器中的数据为54H
进位标志为1时，当执行指令：

SUBB A,R2

结果：

$(A) = 74H$ ， $(AC) = 0$ ， $(CY) = 0$ ， $(OV) = 1$

计算过程为：

$$\begin{array}{r} 1100,1001 \\ 0101,0100 \\ - \quad \quad 1 \\ \hline 0111,0100 \end{array}$$

算术指令

--减法指令

SUBB A,direct

- 该指令从累加器A中减去直接寻址单元的内容和进位标志CY的内容，然后结果保存在累加器A中。CY、AC、OV设置如上。

SUBB A,direct 指令的内容

助记符	操作	标志	操作码	字节数	周期数
SUBB A,direct	$(PC) \leftarrow (PC) + 2$ $(A) \leftarrow (A) - (C) - (\text{direct})$	CY,AC,OV	10010101	2	2

注：在操作码后面跟着一个字节的直接地址。

算术指令

--减法指令

SUBB A,@Ri

- 该指令从累加器A中减去间接寻址单元的内容和进位标志CY的内容，然后结果保存在累加器A中。CY、AC、OV设置如上。

SUBB A,@Ri 指令的内容

助记符	操作	标志	操作码	字节数	周期数
SUBB A,@Ri	$(PC) \leftarrow (PC) + 1$ $(A) \leftarrow (A) - (C) - ((Ri))$	CY,AC,OV	1001011i	1	2

注：i表示R0或者R1。当i=0时，表示R0寄存器；当i=1时，表示R1寄存器。

算术指令

--减法指令

SUBB A,#data

- 该指令从累加器A中减去一个立即数和进位标志CY的内容，然后结果保存在累加器A中。CY、AC、OV设置如上。

SUBB A,#data 指令的内容

助记符	操作	标志	操作码	字节数	周期数
SUBB A,#data	$(PC) \leftarrow (PC) + 2$ $(A) \leftarrow (A) - (C) - data$	CY,AC,OV	10010100	2	2

注：在操作码后面跟着一个字节的立即数。

算术指令

--递增指令



INC A

- 该指令将累加器A的内容加1，结果保存在累加器A中。若累加器的结果为0xFF时，将其内容设置为0。

INC A 指令的内容

助记符	操作	标志	操作码	字节数	周期数
INC A	$(PC) \leftarrow (PC) + 1$ $(A) \leftarrow (A) + 1$	N	00000100	1	1

算术指令

--递增指令

INC Rn

- 该指令将寄存器Rn的内容加1，结果保存在Rn中。若Rn的结果为0xFF时，将其内容设置为0。

INC Rn 指令的内容

助记符	操作	标志	操作码	字节数	周期数
INC Rn	$(PC) \leftarrow (PC) + 1$ $(Rn) \leftarrow (Rn) + 1$	N	00001rrrr	1	2

注：rrr为寄存器的编号，因此机器码范围是08H~0FH。

算术指令

--递增指令



INC direct

- 该指令将直接寻址单元的内容加1，结果保存在直接地址单元中。若直接地址单元的结果为0xFF时，将其内容设置为0。

INC direct 指令的内容

助记符	操作	标志	操作码	字节数	周期数
INC direct	$(PC) \leftarrow (PC) + 2$ $(\text{direct}) \leftarrow (\text{direct}) + 1$	N	00000101	2	3

注：在操作码后面跟着一个字节的直接地址。

算术指令

--递增指令

INC @Ri

- 该指令将间接寻址单元的内容加1，结果保存在间接地址单元中。若间接地址单元的结果为0xFF时，将其内容设置为0。

INC @Ri 指令的内容

助记符	操作	标志	操作码	字节数	周期数
INC @Ri	$(PC) \leftarrow (PC) + 1$ $((Ri)) \leftarrow ((Ri)) + 1$	N	0000011i	1	3

注：i表示R0或者R1。当i=0时，表示R0寄存器；当i=1时，表示R1寄存器。

算术指令

--递增指令

INC DPTR

- 该指令将DPTR的内容加1，结果保存在DPTR中。若DPTR的结果为0xFFFF时，将其内容设置为0x0000。

INC DPTR 指令的内容

助记符	操作	标志	操作码	字节数	周期数
INC DPTR	$(PC) \leftarrow (PC) + 1$ $(DPTR) \leftarrow (DPTR) + 1$	N	10100011	1	1

算术指令

--递增指令

【例】假设寄存器R0中的数据为7EH，内部RAM地址为7EH和7FH单元的数据分别为FFH和40H，即：(7E)=FFH，(7F)=40H，则当执行指令：

INC @R0 ; 内部RAM地址为7EH单元的内容加1，变成0

INC R0 ; 寄存器R0中的数据变为7FH

INC @R0 ; 内部RAM地址为7FH单元的内容加1，变成41H

结果：

(R0)=7FH，内部RAM地址为7EH和7FH单元的数据变为00H和41H。

算术指令

--递减指令

DEC A

- 该指令将累加器A的内容减1，结果保存在累加器A中。如果累加器A中的内容为0，则变为0xFF。

DEC A 指令的内容

助记符	操作	标志	操作码	字节数	周期数
DEC A	$(PC) \leftarrow (PC) + 1$ $(A) \leftarrow (A) - 1$	N	00010100	1	1

算术指令

--递减指令

DEC Rn

- 该指令将寄存器Rn的内容减1，结果保存在寄存器Rn中。如果Rn的内容为0，则变为0xFF。

DEC Rn 指令的内容

助记符	操作	标志	操作码	字节数	周期数
DEC Rn	$(PC) \leftarrow (PC) + 1$ $(Rn) \leftarrow (Rn) - 1$	N	00011rrr	1	2

注：rrr为寄存器的编号，因此机器码范围是18H~1FH。

算术指令

--递减指令

DEC direct

- 该指令将直接寻址单元的内容减1，结果保存在直接地址单元中。如果直接寻址单元的内容为0，则变为0xFF。

DEC direct 指令的内容

助记符	操作	标志	操作码	字节数	周期数
DEC direct	$(PC) \leftarrow (PC) + 2$ $(direct) \leftarrow (direct) - 1$	N	00010101	2	3

注：在操作码后面跟着一个字节的直接地址。

算术指令

--递减指令

DEC @Ri

- 该指令将间接寻址单元的内容减1，结果保存在间接地址单元中。如果间接寻址单元的内容为0，则变为0xFF。

DEC @Ri 指令的内容

助记符	操作	标志	操作码	字节数	周期数
DEC @Ri	$(PC) \leftarrow (PC) + 1$ $((Ri)) \leftarrow ((Ri)) - 1$	N	0001011i	1	3

注：i表示R0或者R1。当i=0时，表示R0寄存器；当i=1时，表示R1寄存器。

算术指令

--递减指令

【例】假设寄存器R0中的数据为7FH，内部RAM地址为7EH和7FH单元的数据分别为00H和40H，即：(7F)=00H，(7E)=40H，则当执行指令：

DEC @R0 ; 内部RAM地址为7FH单元的内容减1，变成FFH

DEC R0 ; 寄存器R0中的数据变为7EH

DEC @R0 ; 内部RAM地址为7EH单元的内容减1，变成3FH

结果：

(R0)=7EH，内部RAM地址为7EH和7FH单元的数据变为FFH和3FH。

算术指令

--乘法指令

MUL AB

- 该指令将累加器A和寄存器B中的两个无符号8位二进制数相乘，所得的16位乘积的低8位结果保存在累加器A中，高8位结果保存在寄存器B中。
- 如果乘积大于255，则溢出标志OV置1；否则OV清零。
- 在执行该命令时，总是清除进位标志CY。

MUL AB 指令的内容

助记符	操作	标志	操作码	字节数	周期数
MUL AB	$(PC) \leftarrow (PC) + 1$ $(A) \leftarrow (A) \times (B)$ 结果第7位到第0位 $(B) \leftarrow (A) \times (B)$ 结果第15位到第8位	CY,OV	10100100	1	2

算术指令

--乘法指令

【例】假设累加器A中的数据为
(80)₁₀=50H，寄存器B中的数据为
(160)₁₀=A0H，则执行指令：

MUL AB

结果：

乘积为(12800)₁₀=3200H，则

(A) = 00H，(B) = 32H，

(CY) = 0，(OV) = 1。

```
      01010000
    × 10100000
    ──────────
      00000000
      00000000
      00000000
      00000000
      00000000
      00000000
      01010000
      00000000
      +01010000
      ──────────
    0011001000000000
```

算术指令

--除法指令

DIV AB

- 该指令用累加器A中的无符号整数除以寄存器B中无符号整数。所得的商保存在累加器A中，余数保存在寄存器B中。
- 当除数（B寄存器的内容）为0时，结果不定，溢出标志OV置1。
- 在执行该指令时，清除进位标志CY。

DIV AB 指令的内容

助记符	操作	标志	操作码	字节数	周期数
DIV AB	$(PC) \leftarrow (PC) + 1$ $(A)_{15-8} \leftarrow (A)/(B)$ $(B)_{7-0} \leftarrow (A)/(B)$	CY,OV	10000100	1	6

算术指令

--除法指令

【例】假设累加器A中的数据为 $(251)_{10} = FBH$ ，寄存器B中的数据为 $(18)_{10} = 12H$ ，则执行指令：

DIV AB

结果： $(A) = 0DH$ ， $(B) = 11H$ ， $(CY) = 0$ ， $(OV) = 0$

$$\begin{array}{r} 00001101 \\ 00010010 \overline{) 11111011} \\ \underline{-10010} \\ 11010 \\ \underline{-10010} \\ 100011 \\ \underline{-010010} \\ 10001 \end{array}$$

算术指令

--BCD调整指令

DA A

- 该指令的功能是对BCD码的加法结果进行调整。
- 两个压缩型BCD码按十进制数相加后，须经此指令的调整才能得到压缩型BCD码的和。
- 本指令是根据A的最初数值和程序状态字PSW的状态，决定对A进行加06H、60H或66H操作的。

DA A 指令的内容

助记符	操作	标志	操作码	字节数	周期数
DA A	$(PC) \leftarrow (PC) + 1$ 如果 $[(A_{3-0}) > 9] \vee [(AC) = 1]$ 则： $(A_{3-0}) \leftarrow (A_{3-0}) + 6$ 如果 $[(A_{7-4}) > 9] \vee [(C) = 1]$ 则： $(A_{7-4}) \leftarrow (A_{7-4}) + 6$	CY	11010100	1	3

算术指令

--BCD调整指令

【例】假设累加器A中的数据为56H，表示10进制数56的BCD码。寄存器R3的内容为67H，表示10进制数67的BCD码。进位标志为1，则执行指令：

ADDC A, R3 ; 累加器A的结果为BEH, (AC)=0, (CY)=0
DA A ; 表示十进制数的124

注：

因为在执行完ADDC指令后，(A)=BEH，

(A)₃₋₀>9，所以(A)₃₋₀+6→(A)₃₋₀=4H，向第4位有进位。

(A)₇₋₄>9，所以(A)₇₋₄+6+进位→(A)₇₋₄=2H，最高位有进位。

逻辑指令

--逻辑与指令

ANL A, Rn

- 该指令将累加器A的内容和寄存器R_n的内容做逻辑与操作，结果保存在累加器A中。

ANL A,Rn指令的内容

助记符	操作	标志	操作码	字节数	周期数
ANL A,Rn	$(PC) \leftarrow (PC) + 1$ $(A) \leftarrow (A) \wedge (Rn)$	N	01011rrr	1	1

注：rrr为寄存器的编号，因此机器码范围是58H~5FH。

逻辑指令

--逻辑与指令

【例】假设累加器A中的数据为C3H，寄存器R0的内容为55H，则执行指令：

ANL A, R0

结果：累加器A中的数据为41H。

计算过程：

$$\begin{array}{r} 11000011 \\ \wedge 01010101 \\ \hline 01000001 \end{array}$$

【例】执行指令

ANL P1, #01110011B

结果：将端口1的第7位、第3位和第2位清零。

逻辑指令

--逻辑与指令

ANL A , direct

- 该指令将累加器A的内容和直接寻址单元的内容做逻辑与操作，结果保存在累加器A中。

ANL A,direct指令的内容

助记符	操作	标志	操作码	字节数	周期数
ANL A,direct	$(PC) \leftarrow (PC) + 2$ $(A) \leftarrow (A) \wedge (\text{direct})$	N	01010101	2	2

注：在操作码后面跟着一个字节的直接地址。

逻辑指令

--逻辑与指令

ANL A, @Ri

- 该指令将累加器A的内容和间接寻址单元中的内容做逻辑与操作，结果保存在累加器A中。

ANL A,@Ri指令的内容

助记符	操作	标志	操作码	字节数	周期数
ANL A,@Ri	$(PC) \leftarrow (PC) + 1$ $(A) \leftarrow (A) \wedge ((Ri))$	N	0101011i	1	2

注：i表示R0或者R1。当i=0时，表示R0寄存器；当i=1时，表示R1寄存器。

逻辑指令

--逻辑与指令

ANL A, #data

- 该指令将累加器A的内容和立即数做逻辑与操作，结果保存在累加器A中。

ANL A,#data指令的内容

助记符	操作	标志	操作码	字节数	周期数
ANL A,#data	$(PC) \leftarrow (PC) + 2$ $(A) \leftarrow (A) \wedge \text{data}$	N	01010100	2	2

注：在操作码后面跟着一个字节的立即数。

逻辑指令

--逻辑与指令

ANL direct, A

- 该指令将累加器A的内容和直接寻址单元的内容做逻辑与操作，结果保存在直接寻址单元中。

ANL direct,A指令的内容

助记符	操作	标志	操作码	字节数	周期数
ANL direct, A	$(PC) \leftarrow (PC) + 2$ $(direct) \leftarrow (direct) \wedge (A)$	N	01010010	2	3

注：在操作码后面跟着一个字节的直接地址。

逻辑指令

--逻辑与指令

ANL direct, #data

- 该指令对立即数和直接寻址单元的内容做逻辑与操作，结果保存在直接寻址单元中。

ANL direct,#data指令的内容

助记符	操作	标志	操作码	字节数	周期数
ANL direct , #data	$(PC) \leftarrow (PC) + 3$ $(direct) \leftarrow (direct) \wedge data$	N	01010011	3	3

注：在操作码后面跟着一个字节的直接地址和一个字节的立即数。

逻辑指令

--逻辑或指令

ORL A,Rn

- 该指令将累加器A的内容和寄存器Rn中内容做逻辑或操作，结果保存在累加器A中。

ORL A,Rn指令的内容

助记符	操作	标志	操作码	字节数	周期数
ORL A,Rn	$(PC) \leftarrow (PC) + 1$ $(A) \leftarrow (A) \vee (Rn)$	N	01001rrr	1	1

注：rrr为寄存器的编号，因此机器码范围是48H~4FH。

逻辑指令

--逻辑或指令

【例】假设累加器A中的数据为C3H，寄存器R0的内容为55H，则执行指令：

ORL A, R0

结果：累加器A中的数据为D7H。

计算过程：

$$\begin{array}{r} 11000011 \\ \vee 01010101 \\ \hline 11010111 \end{array}$$

【例】执行指令：

ORL P1, #00110010B

结果：将端口1的第5位、第4位和第1位置1。

逻辑指令

--逻辑或指令

ORL A,direct

- 该指令将累加器A的内容和直接寻址单元的内容做逻辑或操作，结果保存在累加器A中。

ORL A,direct指令的内容

助记符	操作	标志	操作码	字节数	周期数
ORL A,direct	$(PC) \leftarrow (PC) + 2$ $(A) \leftarrow (A) \vee (\text{direct})$	N	01000101	2	2

注：在操作码后面跟着一个字节的直接地址

逻辑指令

--逻辑或指令

ORL A, @Ri

- 该指令将累加器A的内容和间接寻址单元中内容做逻辑或操作，结果保存在累加器A中。

ORL A,@Ri指令的内容

助记符	操作	标志	操作码	字节数	周期数
ORL A,@Ri	$(PC) \leftarrow (PC) + 1$ $(A) \leftarrow (A) \vee ((Ri))$	N	0100011i	1	2

注：i表示R0或者R1。当i=0时，表示R0寄存器；当i=1时，表示R1寄存器。

逻辑指令

--逻辑或指令

ORL A, #data

- 该指令将累加器A的内容和立即数做逻辑或操作，结果保存在累加器A中。

ORL A,#data指令的内容

助记符	操作	标志	操作码	字节数	周期数
ORL A,#data	$(PC) \leftarrow (PC) + 2$ $(A) \leftarrow (A) \vee \text{data}$	N	01000100	2	2

注：在操作码后面跟着一个字节的立即数。

逻辑指令

--逻辑或指令

ORL direct,A

- 该指令将直接寻址单元的内容和累加器A中内容做逻辑或操作，结果保存在直接寻址单元中。

ORL direct,A指令的内容

助记符	操作	标志	操作码	字节数	周期数
ORL direct, A	$(PC) \leftarrow (PC) + 2$ $(direct) \leftarrow (direct) \vee (A)$	N	01000010	2	3

注：在操作码后面跟着一个字节的直接地址。

逻辑指令

--逻辑或指令

ORL direct , #data

- 该指令将直接寻址单元中内容和立即数做逻辑或操作，结果保存在直接寻址单元中。

ORL direct,#data指令的内容

助记符	操作	标志	操作码	字节数	周期数
ORL direct , #data	$(PC) \leftarrow (PC) + 3$ $(direct) \leftarrow (direct) \vee data$	N	01000011	3	3

注：在操作码后面跟着一个字节的直接地址和一个字节的立即数。

逻辑指令

--逻辑异或指令

XRL A,Rn

- 该指令将累加器A的内容和寄存器Rn的内容做逻辑异或操作，结果保存在累加器A中。

XRL A,Rn指令的内容

助记符	操作	标志	操作码	字节数	周期数
XRL A,Rn	$(PC) \leftarrow (PC) + 1$ $(A) \leftarrow (A) \oplus (Rn)$	N	01101rrr	1	1

注：rrr为寄存器的编号，因此机器码范围是68H~6FH。

逻辑指令

--逻辑异或指令

【例】假设累加器A中的数据为C3H，寄存器R0的内容为AAH，则执行指令：

XRL A, R0

结果：累加器A中的数据为69H。

计算过程：

$$\begin{array}{r} 11000011 \\ \oplus 10101010 \\ \hline 01101001 \end{array}$$

【例】执行指令：

XRL P1, #00110001B

结果：将端口1的第5位、第4位和第0位取反。

如需原始PPT文件请点击此处
<http://www.gpnewtech.com/ppt>

逻辑指令

--逻辑异或指令

XRL A, direct

- 该指令将累加器A的内容和直接寻址单元的内容做逻辑异或操作，结果保存在累加器A中。

XRL A,direct指令的内容

助记符	操作	标志	操作码	字节数	周期数
XRL A,direct	$(PC) \leftarrow (PC) + 2$ $(A) \leftarrow (A) \oplus (\text{direct})$	N	01100101	2	2

注：在操作码后面跟着一个字节的直接地址。

逻辑指令

--逻辑异或指令

XRL A,@Ri

- 该指令将累加器A的内容和间接寻址单元的内容做逻辑异或操作，结果保存在累加器A中。

XRL A,@Ri指令的内容

助记符	操作	标志	操作码	字节数	周期数
XRL A,@Ri	$(PC) \leftarrow (PC) + 1$ $(A) \leftarrow (A) \oplus ((Ri))$	N	0110011i	1	2

注：i表示R0或者R1。当i=0时，表示R0寄存器；当i=1时，表示R1寄存器。

逻辑指令

--逻辑异或指令

XRL A, #data

- 该指令将累加器A的内容和一个立即数做逻辑异或操作，结果保存在累加器A中。

XRL A,#data指令的内容

助记符	操作	标志	操作码	字节数	周期数
XRL A,#data	$(PC) \leftarrow (PC) + 2$ $(A) \leftarrow (A) \oplus \text{data}$	N	01100100	2	2

注：在操作码后面跟着一个字节的立即数。

逻辑指令

--逻辑异或指令

XRL direct , A

- 该指令将直接寻址单元的内容和累加器ACC的内容做逻辑异或操作，结果保存在直接寻址的单元中。

XRL direct,A指令的内容

助记符	操作	标志	操作码	字节数	周期数
XRL direct, A	$(PC) \leftarrow (PC) + 2$ $(direct) \leftarrow (direct) \oplus (A)$	N	01100010	2	3

注：在操作码后面跟着一个字节的直接地址。

逻辑指令

--逻辑异或指令

XRL direct, #data

- 该指令将直接寻址的内容和一个立即数做逻辑异或操作，结果保存在直接寻址的单元中。

XRL direct,#data指令的内容

助记符	操作	标志	操作码	字节数	周期数
XRL direct , #data	$(PC) \leftarrow (PC) + 3$ $(direct) \leftarrow (direct) \oplus data$	N	01100011	3	3

注：在操作码后面跟着一个字节的直接地址和一个字节的立即数。

逻辑指令

--清除指令

CLR A

- 该指令将累加器A中的各位清0，如下表所示。

CLR A 指令的内容

助记符	操作	标志	操作码	字节数	周期数
CLR A	$(PC) \leftarrow (PC) + 1$ $(A) \leftarrow 0$	N	11100100	1	1

【例】假设累加器A中的数据为5CH，则执行指令：

CLR A

结果：(A)=00H

逻辑指令

--取反指令

CPL A

- 该指令将累加器A按位取反，即：将累加器A各位中，逻辑1变成逻辑0，逻辑0变成逻辑1。

CPL A 指令的内容

助记符	操作	标志	操作码	字节数	周期数
CPL A	$(PC) \leftarrow (PC) + 1$ $(A) \leftarrow \overline{(A)}$	N	11110100	1	1

逻辑指令

--取反指令

【例】假设P1端口的数据为5BH，则执行指令：

CPL P1.1

CPL P1.2

结果：

将P1端口设置为5DH=01011101B

逻辑指令

--移位指令

RL A

- 该指令将累加器A中的内容循环左移。

RL A 指令的内容

助记符	操作	标志	操作码	字节数	周期数
RL A	$(PC) \leftarrow (PC) + 1$ $(A_{n+1}) \leftarrow (A_n), n = 0 \sim 6$ $(A_0) \leftarrow (A_7)$	N	00100011	1	1

【例】假设累加器A的数据为C5H（11000101B），则执行指令：

RL A

结果：累加器A的内容变成8BH=10001011B

逻辑指令

--移位指令

RLC A

- 该指令将累加器A的内容和进位标志CY一起循环左移。

RLC A 指令的内容

助记符	操作	标志	操作码	字节数	周期数
RLC A	$(PC) \leftarrow (PC) + 1$ $(A_{n+1}) \leftarrow (A_n), n = 0 \sim 6$ $(A_0) \leftarrow (CY)$ $(CY) \leftarrow (A_7)$	CY	00110011	1	1

【例】假设累加器A的数据为C5H（11000101B），进位标志（CY）=1，则执行指令：

RLC A

结果：累加器A的内容变成8BH=10001011B，进位标志（CY）=1。

如需原始PPT文件请点击此处
<http://www.gpnewtech.com/ppt>

逻辑指令

--移位指令

RR A

- 该指令将累加器A的内容循环右移。

RR A 指令的内容

助记符	操作	标志	操作码	字节数	周期数
RR A	$(PC) \leftarrow (PC) + 1$ $(A_n) \leftarrow (A_{n+1}), n = 0 \sim 6$ $(A_7) \leftarrow (A_0)$	N	00000011	1	1

【例】假设累加器A的数据为C5H（11000101B），则执行指令：

RR A

结果：累加器A的内容变成E2H（11100010B）。如需原始PPT文件请点击此处
<http://www.gpnewtech.com/ppt>

逻辑指令

--移位指令



RRC A

- 该指令将累加器ACC的内容和进位标志CY一起循环右移。

RRC A 指令的内容

助记符	操作	标志	操作码	字节数	周期数
RRC A	$(PC) \leftarrow (PC) + 1$ $(A_n) \leftarrow (A_{n+1}), n = 0 \sim 6$ $(A_7) \leftarrow (CY)$ $(CY) \leftarrow (A_0)$	CY	00010011	1	1

【例】假设累加器A的数据为C5H（11000101B），进位标志（CY=0），则执行指令：

RRC A

结果：累加器A的数据变成62H=01100010B，进位标志（CY）=1。

如需原始PPT文件请点击此处
<http://www.gpnewtech.com/ppt>

逻辑指令

--半字节交换指令

SWAP A

- 该指令将累加器A中的半字节互换，即：将累加器A的高、低半字节互换。

SWAP A 指令的内容

助记符	操作	标志	操作码	字节数	周期数
SWAP A	$(PC) \leftarrow (PC) + 1$ $(A_{3-0}) \leftarrow (A_{7-4})$ $(A_{7-4}) \leftarrow (A_{3-0})$	N	11000100	1	1

【例】假设累加器A的数据为C5H（11000101B），则执行指令：

SWAP A

结果：累加器A的数据变成5CH=01011100B。

数据传输指令

--内部数据传输指令

该类型数据传输指令是在任何两个内部RAM或者SFR间实现数据传输。

- 这些指令使用直接、间接、寄存器和立即数寻址。

数据传输指令

--内部数据传输指令

MOV A,Rn

- 该指令将寄存器R_n中的内容复制到累加器A中，且R_n的内容不发生变化。

MOV A,Rn指令的内容

助记符	操作	标志	操作码	字节数	周期数
MOV A, Rn	$(PC) \leftarrow (PC) + 1$ $(A) \leftarrow (Rn)$	N	11101rrrr	1	1

注：rrrr为寄存器的编号，因此机器码范围是E8H~EFH。

数据传输指令

--内部数据传输指令

MOV A,direct

- 该指令将直接寻址单元的内容复制到累加器A中，且直接寻址单元的内容不发生变化。

MOV A,direct指令的内容

助记符	操作	标志	操作码	字节数	周期数
MOV A, direct	$(PC) \leftarrow (PC) + 2$ $(A) \leftarrow (\text{direct})$	N	11100101	2	2

注：在操作码后面跟着一个字节的直接地址。

数据传输指令

--内部数据传输指令

MOV A,@Ri

- 该指令将间接寻址单元中的内容复制到累加器A中，且间接寻址单元的内容不发生变化。

MOV A,@Ri指令的内容

助记符	操作	标志	操作码	字节数	周期数
MOV A, @Ri	$(PC) \leftarrow (PC) + 1$ $(A) \leftarrow ((Ri))$	N	1110011i	1	2

注：i表示R0或者R1。当i=0时，表示R0寄存器；当i=1时，表示R1寄存器。

数据传输指令

--内部数据传输指令

MOV A,#data

- 该指令将立即数复制到累加器A中，且立即数的内容不发生变化。

MOV A,#data指令的内容

助记符	操作	标志	操作码	字节数	周期数
MOV A, #data	$(PC) \leftarrow (PC) + 2$ $(A) \leftarrow \text{data}$	N	01110100	2	2

注：在操作码后面跟着一个字节的立即数。

数据传输指令

--内部数据传输指令

MOV Rn, A

- 该指令将累加器A的内容复制到寄存器R_n中，且累加器A的内容不发生变化。

MOV Rn,A指令的内容

助记符	操作	标志	操作码	字节数	周期数
MOV Rn, A	$(PC) \leftarrow (PC) + 1$ $(Rn) \leftarrow (A)$	N	11111rrr	1	1

数据传输指令

--内部数据传输指令

MOV Rn , direct

- 该指令将直接寻址单元的内容复制到寄存器Rn中, 且直接寻址单元的内容不发生变化。

MOV Rn, direct指令的内容

助记符	操作	标志	操作码	字节数	周期数
MOV Rn, direct	$(PC) \leftarrow (PC) + 2$ $(Rn) \leftarrow (direct)$	N	10101rrr	2	3

注: rrr为寄存器的编号, 因此机器码范围是A8H~AFH。

数据传输指令

--内部数据传输指令

MOV Rn , #data

- 该指令将立即数复制到寄存器R_n中，且立即数的内容不发生变化。

MOV Rn,#data指令的内容

助记符	操作	标志	操作码	字节数	周期数
MOV Rn, #data	$(PC) \leftarrow (PC) + 2$ $(Rn) \leftarrow data$	N	01111rrr	2	2

- 注：（1）rrr为寄存器的编号，因此机器码范围是78H~7FH。
- （2）在操作码后面跟着一个字节的立即数。

数据传输指令

--内部数据传输指令

MOV direct , A

- 该指令将累加器A的内容复制到直接寻址单元中，且累加器A的内容不发生变化。

MOV direct,A指令的内容

助记符	操作	标志	操作码	字节数	周期数
MOV direct,A	$(PC) \leftarrow (PC) + 2$ $(direct) \leftarrow (A)$	N	11110101	2	2

注：

(1) 在操作码后面跟着一个字节的直接地址。

数据传输指令

--内部数据传输指令

MOV direct , Rn

- 该指令将寄存器Rn的内容复制到直接寻址单元中，且Rn的内容不发生变化。

MOV direct,Rn指令的内容

助记符	操作	标志	操作码	字节数	周期数
MOV direct , Rn	$(PC) \leftarrow (PC) + 2$ $(direct) \leftarrow (Rn)$	N	10001rrr	2	2

注：

- (1) rrr为寄存器的编号，因此机器码范围是88H~8FH。
- (2) 在操作码后面跟着一个字节的直接地址。

数据传输指令

--内部数据传输指令

MOV direct , direct

- 该指令将直接寻址单元的内容复制到另一个直接寻址单元中，且源直接寻址单元的内容不发生变化。

MOV direct,direct指令的内容

助记符	操作	标志	操作码	字节数	周期数
MOV direct , direct	$(PC) \leftarrow (PC) + 3$ $(direct) \leftarrow (direct)$	N	10000101	3	3

注：在操作码后面跟着两个字节的直接地址，一个是源操作数地址，另一个是目的操作数地址。

数据传输指令

--内部数据传输指令

MOV direct , @Ri

- 该指令将间接寻址单元的内容复制到直接寻址单元中，且间接寻址单元的内容不发生变化。

MOV direct,@Ri指令的内容

助记符	操作	标志	操作码	字节数	周期数
MOV direct , @Ri	$(PC) \leftarrow (PC) + 2$ $(direct) \leftarrow ((Ri))$	N	1000011i	2	3

注：

- (1) i表示R0或者R1。当i=0时，表示R0寄存器；当i=1时，表示R1寄存器。
- (2) 在操作码后面跟着一个字节的直接地址。

数据传输指令

--内部数据传输指令

MOV direct , #data

- 该指令将立即数复制到直接寻址单元中，且立即数的内容不发生变化。

MOV direct,#data指令的内容

助记符	操作	标志	操作码	字节数	周期数
MOV direct ,#data	$(PC) \leftarrow (PC) + 3$ $(direct) \leftarrow data$	N	0111010 1	3	3

注：在操作码后面跟着一个字节的直接地址和一个字节的立即数。

数据传输指令

--内部数据传输指令

MOV @Ri, A

- 该指令将累加器A的内容复制到间接寻址的单元中，且累加器A的内容不发生变化。

MOV @Ri,A指令的内容

助记符	操作	标志	操作码	字节数	周期数
MOV @Ri ,A	$(PC) \leftarrow (PC) + 1$ $((Ri)) \leftarrow (A)$	N	1111011i	1	2

注：i表示R0或者R1。当i=0时，表示R0寄存器；当i=1时，表示R1寄存器。

数据传输指令

--内部数据传输指令

MOV @Ri, direct

- 该指令将直接寻址单元的内容复制到间接寻址的寄存器中，且直接寻址寄存器内容不发生变化。

MOV @Ri,direct指令的内容

助记符	操作	标志	操作码	字节数	周期数
MOV @Ri, direct	$(PC) \leftarrow (PC) + 2$ $((Ri)) \leftarrow (direct)$	N	1010011i	2	3

注：

- i表示R0或者R1。当i=0时，表示R0寄存器；当i=1时，表示R1寄存器。
- 在操作码后面跟着一个字节的直接地址。

数据传输指令

--内部数据传输指令

MOV @Ri, #data

- 该指令将立即数内容复制到间接寻址单元中，且立即数的内容不发生变化。

MOV @Ri,#data指令的内容

助记符	操作	标志	操作码	字节数	周期数
MOV @Ri , #data	$(PC) \leftarrow (PC) + 2$ $((Ri)) \leftarrow data$	N	0111011i	2	2

注：

- (1) i表示R0或者R1。当i=0时，表示R0寄存器；当i=1时，表示R1寄存器。
- (2) 在操作码后面跟着一个字节的立即数。

数据传输指令

--内部数据传输指令

MOV DPTR,#data 16

- 该指令将一个**16**位的立即数复制到数据指针**DPTR**寄存器中，且**16**位立即数的内容不发生变化。

MOV DPTR,#data16指令的内容

助记符	操作	标志	操作码	字节数	周期数
MOV DPTR ,#data 16	$(PC) \leftarrow (PC) + 3$ $DPH \leftarrow data_{15-8}$ $DPL \leftarrow data_{7-0}$	N	10010000	3	3

注：在操作码后面跟着两个字节（**16**位）的立即数。

数据传输指令

--内部数据传输指令

【例】假设内部RAM地址为30H的单元的内容为40H，而40H单元的内容为10H。端口1的数据为CAH（11001010B）则执行指令：

MOV R0 , #30H ; 将立即数30H送到寄存器R0 , (R0) =30H

MOV A , @R0 ; 将30H作为指向内部RAM的地址，内部RAM地址为30H
; 单元的内容40H送到累加器A中

MOV R1 , A ; 将累加器A的内容40H，送到寄存器R1中，(R1) =40H

MOV B , @R1 ; 将40H作为指向内部RAM的地址，内部RAM地址为40H
; 单元的内容10H送到寄存器B中

MOV @R1 , P1 ; 将P1端口的内容，送到R1寄存器所指向的内部RAM的
; 地址单元中，即内部RAM地址为40H的单元的内容变为
; CAH。

MOV P2 , P1 ; 将P1端口的内容送到P2端口中，P2端口的内容变为CAH。

数据传输指令

--外部数据传输指令

该类型传输指令是在累加器和外部地址空间实现数据传输数据，这种传输只能使用MOVX指令。

数据传输指令

--外部数据传输指令

MOVX A,@Ri

- 该指令将外部数据存储区的一个字节的内容复制到累加器A中。
8位外部数据存储区地址由R0或R1确定，且外部数据存储器单元的内容不发生变化。

MOVX A,@Ri指令的内容

助记符	操作	标志	操作码	字节数	周期数
MOVX A,@Ri	$(PC) \leftarrow (PC) + 1$ $(A) \leftarrow ((Ri))$	N	1110001i	1	3

注：i表示R0或者R1。当i=0时，表示R0寄存器；当i=1时，表示R1寄存器。

数据传输指令

--外部数据传输指令

【例】假设有一个时分复用地址/数据线的`外部RAM`存储器，容量为`256B`，该存储器连接到`STC`单片机的`P0`端口上，端口`P3`用于提供外部`RAM`所需要的控制信号。端口`P1`和`P2`用作通用输入/输出端口。`R0`和`R1`中的数据分别为`12H`和`34H`，外部`RAM`地址为`34H`的单元内容为`56H`，执行指令：

`MOVX A, @R1`；将外部`RAM`地址为`34H`单元的内容`56H`送到累加器`A`

`MOVX @R0,A`；将累加器`A`的内容`56`，送到外部`RAM`地址为`12H`的单元中。

数据传输指令

--外部数据传输指令

MOVX A,@DPTR

- 该指令将外部数据存储区的一个字节的内容复制到累加器A中。
16位外部数据存储区单元的地址由DPTR寄存器确定，且外部数据存储单元的内容不发生变化。

MOVX A,@DPTR指令的内容

助记符	操作	标志	操作码	字节数	周期数
MOVX A, @DPTR	$(PC) \leftarrow (PC) + 1$ $(A) \leftarrow (DPTR)$	N	11100000	1	2

数据传输指令

--外部数据传输指令

MOVX @Ri, A

- 该指令将累加器A的内容复制到外部数据存储单元中。8位外部数据存储区地址由R0或R1确定，且累加器A中的内容不发生变化。

MOVX @Ri,A指令的内容

助记符	操作	标志	操作码	字节数	周期数
MOVX @Ri, A	$(PC) \leftarrow (PC) + 1$ $((Ri)) \leftarrow (A)$	N	1111001i	1	4

注：i表示R0或者R1。当i=0时，表示R0寄存器；当i=1时，表示R1寄存器。

数据传输指令

--外部数据传输指令

MOVX @DPTR,A

- 该指令将累加器A的内容复制到外部数据存储单元中。16位外部数据存储区单元的地址由DPTR寄存器确定，且累加器A中的内容不发生变化。

MOVX @DPTR,A指令的内容

助记符	操作	标志	操作码	字节数	周期数
MOVX @DPTR, A	$(PC) \leftarrow (PC) + 1$ $(DPTR) \leftarrow (A)$	N	11110000	1	3

数据传输指令

--查找表传输指令

只在累加器和程序存储器之间实现数据传输，这种传输只能使用MOVC指令。

数据传输指令

--查找表传输指令

MOVC A,@A+DPTR

- 该指令将数据指针寄存器DPTR和累加器A的内容相加所得到的存储器地址单元的内容复制到累加器A中。

MOVC A,@A+DPTR指令的内容

助记符	操作	标志	操作码	字节数	周期数
MOV A,@A+DPTR	$(PC) \leftarrow (PC) + 1$ $(A) \leftarrow ((A) + (DPTR))$	N	10010011	1	5

数据传输指令

--查找表传输指令

【例】假设累加器A的值在0~4之间，下面的子程序将累加器A中的值转换为用DB伪指令定义的4个值之一

```
REL_PC: INC A
```

```
        MOVC A,@A+PC
```

```
        RET
```

```
        DB 66H
```

```
        DB 77H
```

```
        DB 88H
```

```
        DB 99H
```

数据传输指令

--查找表传输指令

MOVC A,@A+PC

- 该指令将程序计数器PC和累加器A的内容相加所得到的存储器地址单元的内容复制到累加器A中。

MOVC A, @A+PC指令的内容

助记符	操作	标志	操作码	字节数	周期数
MOVC A,@A+PC	$(PC) \leftarrow (PC) + 1$ $(A) \leftarrow ((A) + (PC))$	N	10000011	1	4

数据传输指令

--堆栈操作指令

POP direct

- 该指令将堆栈指针SP所指向栈顶的内容保存到直接寻址单元中，然后执行 $(SP)-1 \rightarrow (SP)$ 的操作，此操作不影响标志位。

POP direct 指令的内容

助记符	操作	标志	操作码	字节数	周期数
POP direct	$(PC) \leftarrow (PC) + 2$ $(direct) \leftarrow ((SP))$ $(SP) \leftarrow (SP) - 1$	N	11010000	2	2

数据传输指令

--堆栈操作指令

【例】假设堆栈指针的初值为32H，内部RAM地址30H~32H单元的数据分别为20H、23H和01H，则执行指令：

POP DPH

POP DPL

结果：

堆栈指针的值变成30H，(DPH)=01H，(DPL)=23H。

如果继续执行指令：

POP SP

则在这种特殊情况下，在写入出栈数据20H之前，栈指针减小到2FH，然后再随着20H的写入，(SP)=20H。

数据传输指令

--堆栈操作指令



PUSH direct

- 该指令将指针执行后堆栈指针(SP)+1指向栈顶单元，将直接寻址单元的内容送入SP所指向的堆栈空间，此操作不影响标志位。

PUSH direc指令的内容

助记符	操作	标志	操作码	字节数	周期数
PUSH direc	$(PC) \leftarrow (PC) + 2$ $(SP) \leftarrow (SP) + 1$ $((SP)) \leftarrow (\text{direct})$	N	11000000	2	3

数据传输指令

--堆栈操作指令

【例】假设在进入中断服务程序之前堆栈指针的值为09H，数据指针DPTR的值为0123H，则执行下面的指令：

PUSH DPL

PUSH DPH

结果：

堆栈指针变成0BH，并把数据23H和01H分别保存到内部RAM的0AH和0BH的存储单元中。

数据传输指令

--数据交换指令

XCH A ,Rn

■ 该指令将累加器A的内容和寄存器R_n中的内容互相交换。

XCH A,Rn指令的内容

助记符	操作	标志	操作码	字节数	周期数
XCH A,Rn	$(PC) \leftarrow (PC) + 1$ $(A) \leftrightarrow (Rn)$	N	11001rrr	1	2

注：rrr为寄存器的编号，因此机器码范围是C8H~CFH。

数据传输指令

--数据交换指令

XCH A, direct

■ 该指令将累加器A的内容和直接寻址单元的内容互相交换。

XCH A,direct指令的内容

助记符	操作	标志	操作码	字节数	周期数
XCH A,direct	$(PC) \leftarrow (PC) + 2$ $(A) \leftrightarrow (\text{direct})$	N	11000101	2	3

注：在操作码后面跟着一个字节的直接地址。

数据传输指令

--数据交换指令

XCH A,@Ri

■ 该指令将累加器A的内容和间接寻址的内容互相交换。

XCH A,@Ri指令的内容

助记符	操作	标志	操作码	字节数	周期数
XCH A,@Ri	$(PC) \leftarrow (PC) + 1$ $(A) \leftrightarrow ((Ri))$	N	1100011i	1	3

注：i表示R0或者R1。当i=0时，表示R0寄存器；当i=1时，表示R1寄存器。

数据传输指令

--数据交换指令

【例】假设R0的内容为地址20H，累加器A的内容为3FH。内部RAM地址为20H单元的内容为75H，执行指令：

XCH A , @R0

将20H所指向的内部RAM的单元的数据75H和累加器A的内容3FH进行交换。

结果：

累加器A的内容变成75H，而内部RAM地址为20H单元的内容变成3FH。

数据传输指令

--数据交换指令

XCHD A,@Ri

- 该指令将累加器A的内容和间接寻址单元内容的低半字节互相交换。

XCHD A,@Ri指令的内容

助记符	操作	标志	操作码	字节数	周期数
XCHD A,@Ri	$(PC) \leftarrow (PC) + 1$ $(A3-0) \leftrightarrow ((Ri)3-0)$	N	1101011i	1	3

注：i表示R0或者R1。当i=0时，表示R0寄存器；当i=1时，表示R1寄存器。

数据传输指令

--数据交换指令

【例】假设寄存器R0的内容为20H，累加器A的内容为36H。内部RAM地址为20H的单元内容为75H，执行指令：

XCHD A , @R0

将20H所指向的内部RAM的单元的数据75H和累加器A的内容36H的低四位数据进行交换。

结果：

累加器A的内容变成35H，而内部RAM地址为20H单元的内容变成76H。

布尔指令

--清除指令

CLR bit

- 该指令将目的比特位清0。

CLR bit 指令的内容

助记符	操作	标志	操作码	字节数	周期数
CLR bit	$(PC) \leftarrow (PC) + 2$ $(bit) \leftarrow 0$	N	11000010	2	3

注：在操作码后面跟着一个字节的位地址。

布尔指令 --清除指令

【例】假设端口P1的数据为5DH (01011101B) , 执行指令 :

CLR P1.2

结果 :

端口P1的内容为59H (01011001B)

布尔指令

--清除指令

CLR C

■ 该指令将进位标志位CY清0

CLR C指令的内容

助记符	操作	标志	操作码	字节数	周期数
CLR C	$(PC) \leftarrow (PC) + 1$ $(C) \leftarrow 0$	N	11000011	1	1

布尔指令

--设置指令

SETB bit

■ 该指令将目标比特位置1

SETB bit指令的内容

助记符	操作	标志	操作码	字节数	周期数
SETB bit	$(PC) \leftarrow (PC) + 2$ $(bit) \leftarrow 1$	N	11010010	2	3

注：在操作码后面跟着一个字节的位地址。

布尔指令

--设置指令

SETB C

- 该指令将进位标志CY置1。

SETB C指令的内容

助记符	操作	标志	操作码	字节数	周期数
SETB C	$(PC) \leftarrow (PC) + 1$ $(C) \leftarrow 1$	N	11010011	1	1

布尔指令

--设置指令

【例】假设端口P1的数据为34H（00110100B），执行指令：

SETB C

SETB P1.0

结果：

进位标志（CY）=1，端口P1的数据变成为35H（00110101B）。

布尔指令

--取反指令

CPL bit

- 该指令将目标比特位取反。

CPL bit 指令的内容

助记符	操作	标志	操作码	字节数	周期数
CPL bit	$(PC) \leftarrow (PC) + 2$ $(bit) \leftarrow \overline{(bit)}$	N	10110010	2	3

注：在操作码后面跟着一个字节的位地址。

布尔指令

--取反指令

【例】假设端口P1的数据为5BH（01011011B），执行指令：

CPL P1.1

CPL P1.2

结果：

端口P1的内容变成为5DH（01011101B）。

布尔指令

--取反指令

CPL C

该指令将进位标志CY取反。如果CY为1，执行该指令后CY为0；反之亦然。

CPL C指令的内容

助记符	操作	标志	操作码	字节数	周期数
CPL C	$(PC) \leftarrow (PC) + 1$ $(C) \leftarrow \overline{(C)}$	CY	10110011	1	1

布尔指令 --逻辑与指令

ANL C , bit

- 该指令对进位标志CY和一个比特位做逻辑与操作，结果保存在CY中。

ANL C,bit 指令的内容

助记符	操作	标志	操作码	字节数	周期数
ANL C , bit	$(PC) \leftarrow (PC) + 2$ $(CY) \leftarrow (CY) \wedge (\text{bit})$	CY	10000010	2	2

注：在操作码后面跟着一个字节的位地址。

布尔指令

--逻辑与指令

ANL C , /bit

- 该指令对进位标志CY和一个比特位取反后做逻辑与操作，结果保存在CY中。

ANL C, /bit指令的内容

助记符	操作	标志	操作码	字节数	周期数
ANL C , /bit	$(PC) \leftarrow (PC) + 2$ $(CY) \leftarrow (CY) \wedge \overline{(\text{bit})}$	CY	10110000	2	2

注：在操作码后面跟着一个字节的位地址。

布尔指令 --逻辑与指令

【例】假设P1端口的第0位为1，且累加器A的第7位为1，同时溢出标志OV的内容为0，执行指令：

MOV C , P1.0 ; 进位标志CY设置为1

ANL C , ACC.7 ; 进位标志CY设置为1

ANL C , /OV ; 进位标志CY设置为1

布尔指令

--逻辑或指令

ORL C, bit

- 该命令把进位标志CY的内容和比特位内容做逻辑或,结果保存在CY中。

ORL C, bit 指令的内容

助记符	操作	标志	操作码	字节数	周期数
ORL C,bit	$(PC) \leftarrow (PC) + 2$ $(CY) \leftarrow (CY) \vee \text{bit}$	CY	01110010	2	2

注：在操作码后面跟着一个字节的位地址。

布尔指令

--逻辑或指令

ORL C, /bit

- 该命令把进位标志CY的内容和比特位内容取反后做逻辑或操作，结果保存在CY中。

ORL C,/bit指令的内容

助记符	操作	标志	操作码	字节数	周期数
ORL C,/bit	$(PC) \leftarrow (PC) + 2$ $(CY) \leftarrow (CY) \vee /(\text{bit})$	CY	10100000	2	2

注：在操作码后面跟着一个字节的位地址。

布尔指令

--逻辑或指令

【例】假设P1端口的第0位为1，或者累加器A的第7位为1，或者溢出标志OV的内容为0，执行指令：

MOV C , P1.0	; 进位标志CY设置为1
ORL C , ACC.7	; 进位标志CY设置为1
ORL C , /OV	; 进位标志CY设置为1

布尔指令

--传输指令

MOV C, bit

- 该命令把一个比特位的值复制到进位标志CY中，且比特位的值不发生变化。

MOV C, bit 指令的内容

助记符	操作	标志	操作码	字节数	周期数
MOV C,bit	$(PC) \leftarrow (PC) + 2$ $(CY) \leftarrow (bit)$	CY	10100010	2	2

注：在操作码后面跟着一个字节的位地址。

布尔指令

--传输指令

MOV bit ,C

- 该命令把进位标志CY的内容和比特位内容取反后做逻辑或操作，结果保存在CY中。

MOV bit,C指令的内容

助记符	操作	标志	操作码	字节数	周期数
MOV bit ,C	$(PC) \leftarrow (PC) + 2$ $(bit) \leftarrow (CY)$	N	10010010	2	3

注：在操作码后面跟着一个字节的位地址。

布尔指令 --传输指令

【例】假设进位标志CY的初值为1，端口P2中的数据为C5H（11000101B），端口P1中的数据为35H（00110101B），执行指令：

```
MOV P1.3 , C      ; P1端口的值变为3DH ( 00111101B )  
MOV C , P2.3      ; 进位标志CY设置为0  
MOV P1.2 , C      ; P1端口的值变为39H ( 00111001B )
```

布尔指令

--跳转指令

JB bit,rel

- 该命令判断bit位中的数据是否为1，如果为1则跳转到(PC) + rel 指定的目标地址；否则，程序转向下一条指令，该操作不影响标志位。

JB bit,rel 指令的内容

助记符	操作	标志	操作码	字节数	周期数
JB bit,rel	$(PC) \leftarrow (PC) + 3$ 如果 (bit) = 1 , 则 $(PC) \leftarrow (PC) + rel$	N	00100000	3	5

注：在操作码后面跟着一个字节的位地址和一个字节的目的地地址。

布尔指令

--跳转指令

【例】假设端口1的数据为CAH (11001010B) , 累加器A的内容为56H (01010110B) 。则执行指令：

JB P1.2 , LABEL1 ; 跳转条件不成立

JB ACC.2 , LABEL2 ; 跳转条件成立

结果：

程序跳转到标号LABEL2的地方执行。

布尔指令

--跳转指令

JNB bit , rel

- 该命令判断bit中的数据是否为0，如果为0则程序跳转到(PC) + rel指定的目标地址去，否则，程序转向下一条指令，该操作不影响标志位。

JNB bit,rel 指令的内容

助记符	操作	标志	操作码	字节	周期
JNB bit ,rel	$(PC) \leftarrow (PC) + 3$ 如果 (bit) = 0 , 则 $(PC) \leftarrow (PC) + rel$	N	00110000	3	5

注：在操作码后面跟着一个字节的位地址和一个字节的目的地地址。

布尔指令

--跳转指令

【例】假设端口1的数据为CAH（11001010B），累加器A的内容为56H（01010110B）。则执行指令：

JNB P1.3 , LABEL1 ; 跳转条件不成立

JNB ACC.3 , LABEL2 ; 跳转条件成立

结果：

程序跳转到标号LABEL2的地方执行。

布尔指令

--跳转指令

JC rel

- 该命令判断进位标志位CY是否为1，如果为1则跳转到 (PC)+rel 指定的目标地址；否则，程序转向下一条指令，该操作不影响标志位。

JC rel 指令的内容

助记符	操作	标志	操作码	字节数	周期数
JC rel	$(PC) \leftarrow (PC) + 2$ 如果 $(CY) = 1$, 则 $(PC) \leftarrow (PC) + rel$	N	01000000	2	3

注：在操作码后面跟着一个字节的目的地地址。

布尔指令

--跳转指令

【例】假设进位标志为CY为0，则执行指令：

```
JC LABEL1      ; 跳转条件不成立  
CPL C           ; 取反，进位标志CY变为1  
JC LABEL2      ; 跳转条件成立
```

结果：

程序跳转到标号LABEL2的地方执行。

布尔指令

--跳转指令

JNC rel

- 该命令判断进位标志位CY是否为0，如果为0则跳转到(PC)+rel指定的目标地址；否则，程序转向下一条指令，该操作不影响标志位。

JNC rel 指令的内容

助记符	操作	标志	操作码	字节数	周期数
JNC rel	$(PC) \leftarrow (PC) + 2$ 如果(CY) = 0 , 则 $(PC) \leftarrow (PC) + rel$	N	01010000	2	3

注：在操作码后面跟着一个字节的目地址。

布尔指令

--跳转指令

【例】假设进位标志为CY为1，则执行指令：

JNC LABEL1 ; 跳转条件不成立

CPL C ; 取反，进位标志CY变为0

JNC LABEL2 ; 跳转条件成立

结果：

程序跳转到标号LABEL2的地方执行。

布尔指令

--跳转指令

JBC bit,rel

- 该命令判断指定bit位是否为1，如果为1则将该位清零，并且跳转到(PC) + rel指定的目标地址；否则，程序转向下一条指令，该操作操作不影响标志位。

JBC bit , rel 指令的内容

助记符	操作	标志	操作码	字节数	周期数
JBC bit , rel	$(PC) \leftarrow (PC) + 3$ 如果(bit) = 1 , 则 : $bit \leftarrow 0 , (PC) \leftarrow (PC) + rel$	N	00010000	3	5

注：在操作码后面跟着一个字节的目的地址和一个字节的位地址。

如需原始PPT文件请点击此处
<http://www.gdnewtech.com/ppt>

布尔指令

--跳转指令

【例】 假设累加器A的内容为56H (01010110B) , 则执行指令 :

JBC ACC.3 , LABEL1 ; 跳转条件不成立

JBC ACC.2 LABEL2 ; 跳转条件成立 , 并且将ACC.2清零

结果 :

程序跳转到标号 LABEL2 的地方执行 , 累加器 A 的内容变为 52H (01010010B) 。

程序分支指令

8051支持有条件和无条件的程序分支指令，这些程序分支指令用于修改程序的执行顺序。

程序分支指令

--调用指令

ACALL addr11

- 该命令无条件的调用在指定地址处的子程序。目标地址由递增PC的高5位、操作码的第7到第5位和指令第2字节并置组成。所以，所调用的子程序的首地址必须与ACALL后面指令的第一个字节在同一个2KB区域内。

ACALL addr11 指令的内容

助记符	操作	标志	操作码	字节数	周期数
ACALL addr11	$(PC) \leftarrow (PC) + 2$ $(SP) \leftarrow (SP) + 1$ $((SP)) \leftarrow (PC_{7-0})$ $(SP) \leftarrow (SP) + 1$ $((SP)) \leftarrow (PC_{15-8})$ $(PC_{10-0}) \leftarrow$	无	a ₁₀ a ₉ a ₈ 10010	2	4

程序分支指令

--调用指令

【例】假设堆栈指针的初值为07H，标号SUBRTN位于程序存储器地址为0345H的位置，如果执行位于地址0123H处的指令：

ACALL SUBRTN

结果：

堆栈指针的内容变成09H，内部RAM地址为08H和09H的位置保存的内容为25H和01H，PC值变为0345H。

程序分支指令

--调用指令

LCALL addr16

该命令无条件的调用首地址为addr16处的子程序。执行该指令时，将PC加3，以获得下一条指令的地址。然后将指令第2，第3字节所提供的16位目标地址送入PC₁₅₋₀，程序转向子程序的首地址执行。所调用的子程序的首地址可以在64KB的范围内。

LCALL addr16 指令的内容

助记符	操作	标志	操作码	字节数	周期数
LCALL addr16	(PC) ← (PC) + 3 (SP) ← (SP) + 1 ((SP)) ← (PC ₇₋₀) (SP) ← (SP) + 1 ((SP)) ← (PC ₁₅₋₈) (PC) ← 页面地址	N	00010010	3	4 151

程序分支指令 --调用指令

【例】假设堆栈指针的初值为07H，标号SUBRTN位于程序存储器地址为1234H的位置，如果执行位于地址0123H处的指令：

LCALL SUBRTN

结果：

堆栈指针的内容变成09H，内部RAM地址为08H和09H的位置保存的内容为26H和01H，PC值变为1234H。

程序分支指令

--返回指令

RET

- 该命令将栈顶高地址和低地址字节连续的送给PC的高字节和低字节，并把堆栈指针减2，返回ACALL或LCALL的下一条指令，继续往下执行，该指令的操作不影响标志位。

RET 指令的内容

助记符	操作	标志	操作码	字节数	周期数
RET	$(PC_{15-8}) \leftarrow ((SP))$ $(SP) \leftarrow (SP) - 1$ $(PC_{7-0}) \leftarrow ((SP))$ $(SP) \leftarrow (SP) - 1$	N	00100010	1	4

程序分支指令 --调用指令

【例】堆栈指针的内容为0BH，内部RAM地址为0AH和0BH的位置保存的内容为23H和01H，如果执行指令：

RET

结果：

堆栈指针的内容变成09H，程序将从地址为0123H的地方继续执行。

程序分支指令

--返回指令

RETI

- 该命令将从中断程序返回，并清除相应的内部中断状态寄存器。CPU在执行RETI后，至少要再执行一条指令，才能响应新的中断请求。

RETI 指令的内容

助记符	操作	标志	操作码	字节数	周期数
RETI	$(PC_{15-8}) \leftarrow ((SP))$ $(SP) \leftarrow (SP) - 1$ $(PC_{7-0}) \leftarrow ((SP))$ $(SP) \leftarrow (SP) - 1$	N	00110010	1	4

程序分支指令

--返回指令

【例】堆栈指针的内容为0BH，结束在地址0123H处的指令执行结束期间产生中断，内部RAM地址为0AH和0BH的位置保存的内容为23H和01H，如果执行指令：

RETI

结果：

堆栈指针的内容变成09H，中断返回后继续从程序代码地址为0123H的位置继续执行。

程序分支指令

--无条件转移指令

AJMP addr11

- 该命令实现无条件跳转。绝对跳转操作它的目标地址是由PC当前值的高5位，操作码的7-5位，和第二个字节并置而成。目标地址必须包含AJMP指令后第一条指令的第一个字节在内的2KB范围内。

AJMP addr11指令的内容

助记符	操作	标志	操作码	字节数	周期数
AJMP addr11	$(PC) \leftarrow (PC) + 2$ $(PC_{10-0}) \leftarrow \text{页面地址}$	N	a ₁₀ a ₉ a ₈ 00001	2	3

程序分支指令

--无条件转移指令

【例】假设标号JMPADR位于程序存储器的0123H的位置，如果指令：

AJMP JMPADR

位于程序存储器地址为0345H的位置

结果：

执行完该指令后，PC的值变为0123H。

程序分支指令

--无条件转移指令

LJMP addr16

- 该命令实现无条件长跳转操作，跳转的16位目的地址由指令的第2和第3字节组成。因此，程序指向的目标地址可以包含程序存储器的整个64KB的空间。

LJMP 指令的内容

助记符	操作	标志	操作码	字节数	周期数
LJMP addr16	$(PC) \leftarrow \text{addr}_{15} \dots \text{addr}_0$	N	00000010	3	4

注：在操作码后面带着一个字节的目標地址的A₁₅~A₈位和一个字节的目標地址的A₇~A₀位。

程序分支指令

--无条件转移指令

【例】假设标号JMPADR位于程序存储器的1234H的位置，如果指令：

LJMP JMPADR

位于程序存储器地址为1234H的位置

结果：

执行完该指令后，PC的值变为1234H。

程序分支指令

--无条件转移指令

SJMP rel

- 该命令实现无条件短跳转操作，跳转的目的地址是由PC的当前值和指令的第二字节带符号的相对地址相加而成的。

SJMP rel 指令的内容

助记符	操作	标志	操作码	字节数	周期数
SJMP rel	$(PC) \leftarrow (PC) + 2$ $(PC) \leftarrow (PC) + rel$	N	10000000	2	3

注：在操作码后面带着一个字节的目標地址rel。

程序分支指令

--无条件转移指令

【例】假设标号RELADR位于程序存储器的0123H的位置，如果指令：

SJMP RELADR

位于程序存储器地址为0100H的位置。

结果：

执行完该指令后，PC的值变为0223H。

程序分支指令

--无条件转移指令

JMP @A+DPTR

- 该命令实现无条件的跳转操作，跳转的目标地址是将累加器A中的8位无符号数与数据指针DPTR的内容相加而得。相加运算不影响累加器A和数据指针DPTR的原内容。若相加结果大于64KB，则从程序存储器的零地址往下延续。

JMP @A+DPTR指令的内容

助记符	操作	标志	操作码	字节数	周期数
JMP @A+DPTR	$(PC) \leftarrow (A) + (DPTR)$	N	01110011	1	5

程序分支指令

--无条件转移指令

【例】假设累加器A中的值是偶数（0~6）。下面的指令序列将使程序跳转到位于跳转表JMP_TBL的4条AJMP指令中的某一条去执行：

```
MOV DPTR,#JMP_TBL
```

```
JMP @A+DPTR
```

```
JMP_TBL: AJMP LABEL0
```

```
AJMP LABEL1
```

```
AJMP LABEL2
```

```
AJMP LABEL3
```

如果开始执行上面指令时，累加器A中的值为04H，那么程序最终会跳到标号为LABEL2的地方执行。

程序分支指令

--有条件转移指令

JNZ rel

- 该命令实现有条件跳转。判断累加器A的内容是否不为0，如果不为0，则跳转到(PC) + rel指定的目标地址；否则，程序转向下一条指令。

JNZ rel指令的内容

助记符	操作	标志	操作码	字节数	周期数
JNZ rel	$(PC) \leftarrow (PC) + 2$ 如果 $(A) \neq 0$, 则 $(PC) \leftarrow (PC) + rel$	N	01110000	2	4

注：在操作码后面带着一个字节的目標地址rel。

程序分支指令

--有条件转移指令

【例】假设累加器A的内容00H，则执行指令：

```
JNZ LABEL1    ; 跳转条件不成立  
INC A          ; 累加器的内容加1  
JNZ LABEL2    ; 跳转条件成立
```

结果：

程序跳转到标号LABEL2的地方执行。

程序分支指令

--有条件转移指令

JZ rel

- 该命令实现有条件跳转。判断累加器A的内容是否为0，如果为0，则跳转到(PC) + rel指定的目标地址；否则，程序转向下一条指令。

JZ rel 指令的内容

助记符	操作	标志	操作码	字节数	周期数
JZ rel	$(PC) \leftarrow (PC) + 2$ 如果 $(A) = 0$, 则 $(PC) \leftarrow (PC) + rel$	N	01100000	2	4

注：在操作码后面带着一个字节的目標地址rel。

程序分支指令

--有条件转移指令

【例】假设累加器A的内容01H，则执行指令：

```
JZ LABEL1      ; 跳转条件不成立  
DEC A          ; 累加器的内容减1  
JZ LABEL2      ; 跳转条件成立
```

结果：程序跳转到标号LABEL2的地方执行。

程序分支指令

--有条件转移指令

CJNE A, direct, rel

- 该命令对累加器A和直接寻址单元内容相比较，若它们的值不相等则程序转移到(PC) + rel指向的目标地址。若直接寻址单元的内容小于累加器内容，则清除进位标志CY；否则，置位进位标志CY。

CJNE A, direct, rel指令的内容

助记符	操作	标志	操作码	字节数	周期数
CJNE A, direct , rel	$(PC) \leftarrow (PC) + 3$ 如果 $(A) \neq (\text{direct})$ 则 $(PC) \leftarrow (PC) + \text{rel}$ 如果 $(A) < (\text{direct})$, 则 $(CY) \leftarrow 1$ 否则 $(CY) \leftarrow 0$	N	10110101	3	3

程序分支指令

--有条件转移指令

CJNE A, #data, rel

- 该命令将比较累加器A的内容和立即数，若它们的值不相等，则程序转移(PC) + rel指向的目标地址。进位标志CY设置同上，该指令不影响累加器A的内容。

CJNE A, #data, rel指令的内容

助记符	操作	标志	操作码	字节数	周期数
CJNE A, direct , rel	$(PC) \leftarrow (PC) + 3$ 如果 $(A) \neq (data)$ 则 $(PC) \leftarrow (PC) + rel$ 如果 $(A) < (data)$, 则 $(CY) \leftarrow 1$ 否则 $(CY) \leftarrow 0$	N	10110101	3	3

如需原始PPT文件请点击此处

<http://www.gpnewtech.com/ppt>

程序分支指令

--有条件转移指令

CJNE Rn, #data, rel

- 该命令将寄存器Rn的内容和立即数进行比较，若它们的值不相等，则程序转移到(PC) + rel指向的目标地址。进位标志CY设置同上。

CJNE Rn, #data,rel 指令的内容

助记符	操作	标志	操作码	字节数	周期数
CJNE Rn, #data,rel	$(PC) \leftarrow (PC) + 3$ 如果 $(Rn) \neq data$, 则 $(PC) \leftarrow (PC) + rel$ 如果 $(Rn) < data$, 则 $(CY) \leftarrow 1$ 否则 $(CY) \leftarrow 0$	CY	10111rrr	3	4

注：rrr为寄存器的编号，因此机器码范围是B8H~BFH。
如需原始PPT文件请点击此处
<http://www.gpnewtech.com/ppt>

程序分支指令

--有条件转移指令

CJNE @Ri , #data , rel

- 该命令将间接寻址的内容和立即数相比较，若它们的值不相等，则程序转移到(PC) + rel指向的目标地址。进位标志CY设置同上。

CJNE @Ri , #data,rel指令的内容

助记符	操作	标志	操作码	字节数	周期数
CJNE @Ri , #data,rel	$(PC) \leftarrow (PC) + 3$ 如果 $((Ri)) \neq (data)$, 则 $(PC) \leftarrow (PC) + rel$ 如果 $((Ri)) < data$, 则 $(CY) \leftarrow 1$ 否则 $(CY) \leftarrow 0$	CY	1011011i	3	5

注：i表示R0或者R1。当i=0时，表示R0寄存器；当i=1时，表示R1寄存器。

如需原始PPT文件请点击此处
<http://www.gpnewtech.com/ppt>

程序分支指令

--有条件转移指令



【例】 假设累加器A的内容34H，寄存器R7的内容为56H。则执行指令：

CJNE R7 , #60H , NOT_EQ

..... ; R7的内容为60H

NOT_EQ: JC REQ_LOW ; 如果R7<60H

..... ; R7>60H

结果：

第一条指令将进位标志CY设置为1，程序跳转到标号NOT_EQ的地方。接下来测试进位标志CY，可以确定寄存器R7的内容大于还是小于60H。

程序分支指令

--有条件转移指令

DJNZ Rn,rel

- 该命令实现有条件跳转。每执行一次指令，寄存器Rn的内容减1，并判断其内容是否为0。若不为0则转向(PC) + rel指向的目标地址，继续执行循环程序，否则，结束循环程序，执行下一条指令。

DJNZ Rn , rel指令的内容

助记符	操作	标志	操作码	字节数	周期数
DJNZ Rn , rel	$(PC) \leftarrow (PC) + 2$ $(Rn) \leftarrow (Rn) - 1$ 如果 $(Rn) \neq 0$, 则 $(PC) \leftarrow (PC) + rel$	N	11011rrr	2	4

程序分支指令

--有条件转移指令

DJNZ direct,rel

- 该命令实现有条件跳转。每执行一次指令，直接寻址单元的内容减1，并判断其内容是否为0。若不为0则转向(PC) + rel指向的目标地址，继续执行循环程序，否则，结束循环程序，执行下一条指令。

DJNZ direct , rel 指令的内容

助记符	操作	标志	操作码	字节数	周期数
DJNZ direct , rel	$(PC) \leftarrow (PC) + 3$ $(direct) \leftarrow (Rn) - 1$ 如果 $(direct) \neq 0$ 则 $(PC) \leftarrow (PC) + rel$	N	11010101	3	5

程序分支指令

--有条件转移指令

【例】假设内部RAM地址为40H、50H和60H的单元分别保存着数据01H、70H和15H，则执行指令：

DJNZ 40H , LABEL_1

DJNZ 50H , LABEL_2

DJNZ 60H , LABEL_3

结果：

程序将跳转到标号LABEL_2处执行，且相应的3个RAM单元的内容变成00H、6FH和15H。

程序分支指令

--空操作指令

NOP

- 该命令表示无操作，PC+1。

NOP 指令的内容

助记符	操作	标志	操作码	字节数	周期数
NOP	$(PC) \leftarrow (PC) + 1$	N	0x00	1	1

程序分支指令

--空操作指令

【例】假设期望在端口P2的第7位引脚上输出一个长时间的低电平脉冲，该脉冲持续5个及其周期（精确）。若仅仅使用SETB和CLR指令序列，生成的脉冲只能持续一个机器周期。因此，需要设法增加4个额外的机器周期，可以按照下面的方式实现所要求的功能（假设在此期间没有使能中断）：

CLR P2.7

NOP

NOP

NOP

NOP

SETB P2.7