# XA 用户手册

# 6 指令集和寻址

这一节包含 XA 的数据类型和寻址方式的有关信息.目标是帮助用户熟悉处理器的编程性能.

## 6.1 寻址方式

寻址方式是形成操作数有效地址的方法.XA提供了7种基本的有效寻址方式,可以对字,字节,位数据进行操作,或指定分支地址的目标地址.这些基本的寻址方式均支持众多的指令.表1包含了这些XA的基本寻址方式.一个指令可以使用这些寻址方式的组合,例如ADD R0, #020是寄存器寻址和立即寻址的组合.

所有模式(非寄存器寻址)产生一个16-bit的地址,这个地址同DS/ES[23:16],对数据/ PC/CS[23:16],对代码,形成一个24-bit的完整地址.

一个XA指令可以有0,1,或2个操作数,由寻址方式决定操作数.目标操作数将被结果取代,或在某些方面被指令改变.目标操作树在一个寻址方式表达式中首先被列出.源操作数的数值被指令移动或获得,但不会改变.源操作数在寻址模式表达式中第二个被列出.

**Table 6.1** 基本的寻址方式

| 寻址方式 | 助记符 | 操作数 |
|---|---|---|
| 寄存器寻址 | R | 操作数在寄存器中. |
| 间接寻址 | [R] | R 中是字/字节的 16-bit 地址. |
| 间接偏移寻址 | [R+off 8/16] | 字/字节的地址由 R 中的 16-bit 地址,加上有符号的 8/16-bit 偏移量. |
| 直接寻址 | 存储器地址 | 给定地址的字/字节. |
| 特殊功能寄存器寻址 1 | SFR 地址 | 给定 SFR 地址处的字/字节. |
| 立即寻址 | 4/5 位,8/16 位立即数 | 包含 4/5 位,8/16 位立即数的 8/16 位的整数. |
| 位寻址 | 位 | 寄存器,数据区域,SFR 的 10-bit 的位寻址空间 |

1.这是可以看成直接寻址的特殊情况,因为SFR空间同数据空间是分离的.

## 6.2 寻址模式详解

### 6.2.1 寄存器寻址

使用这种寻址方式的指令包含一个操作数区域,对寄存器寻址.寄存器可以是字节,字,双字或可寻址的位.例如:

ADD R6, R4

操作以前: R4包含005Ah,R6包含A5A5h.

操作以后: R4包含005Ah,R6包含A5FFh.

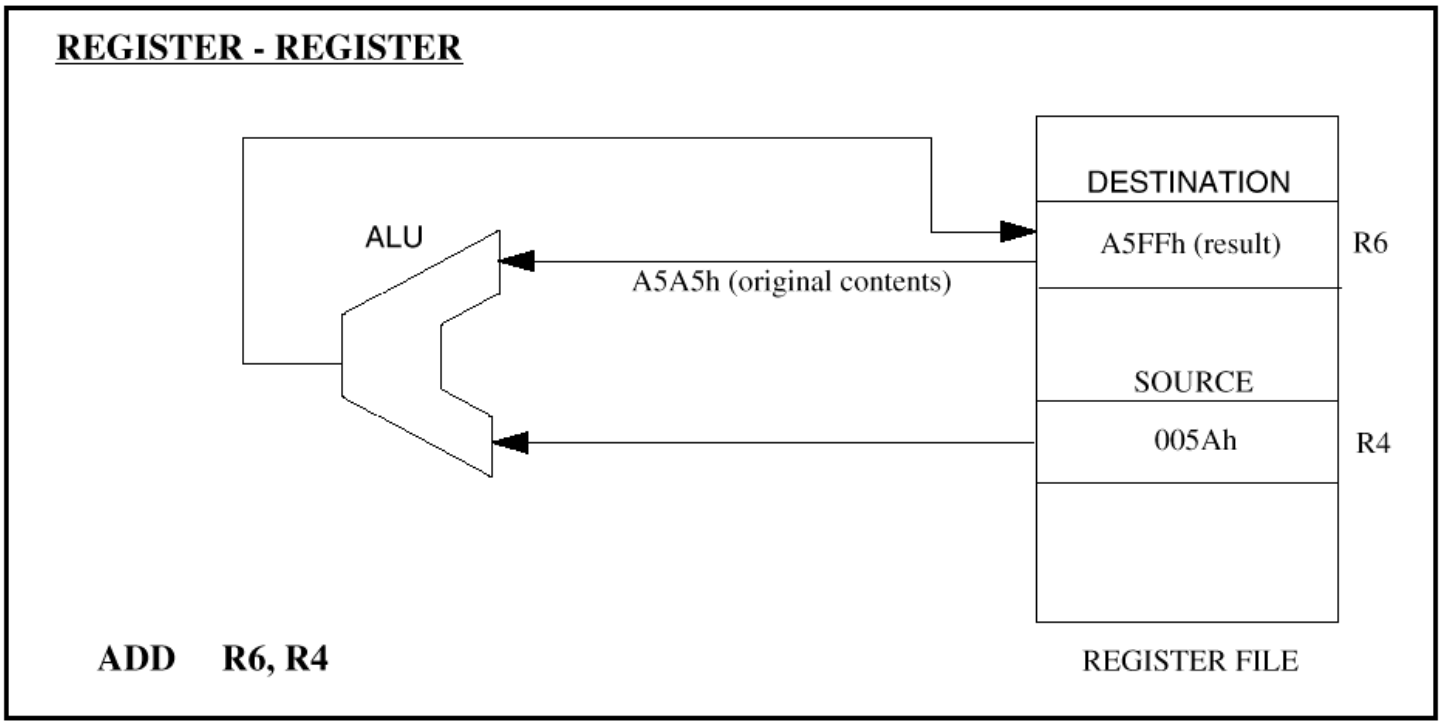



图**6.1**

### **6.2.2** 间接寻址

使用这种寻址方式的指令包含一个16-bit的地址区域.这个区域包含一个指针寄存器,它是寄存器队列中的8个指针寄存器之一,包含操作数的16-bit地址,该地址可以在任何的64K数据段中.对于数据,段的选择由DS或ES决定;对于代码,由PC23-16或CS决定.根据相应的间接寄存器序号,SSEL.n=0,选择DS和PC; SSEL.n=1,选择ES和CS.对于字操作,操作数的地址必须是偶数.例如:

ADD R6, [R4]

操作以前**:** R6包含1005h, R4包含A000h.SSEL.4=1,ES=8,第8段A0000h处内容为A5A5h.

操作以后**:** R6包含B5AAh, R4包含A000h.第8段A0000h处内容为A5A5.

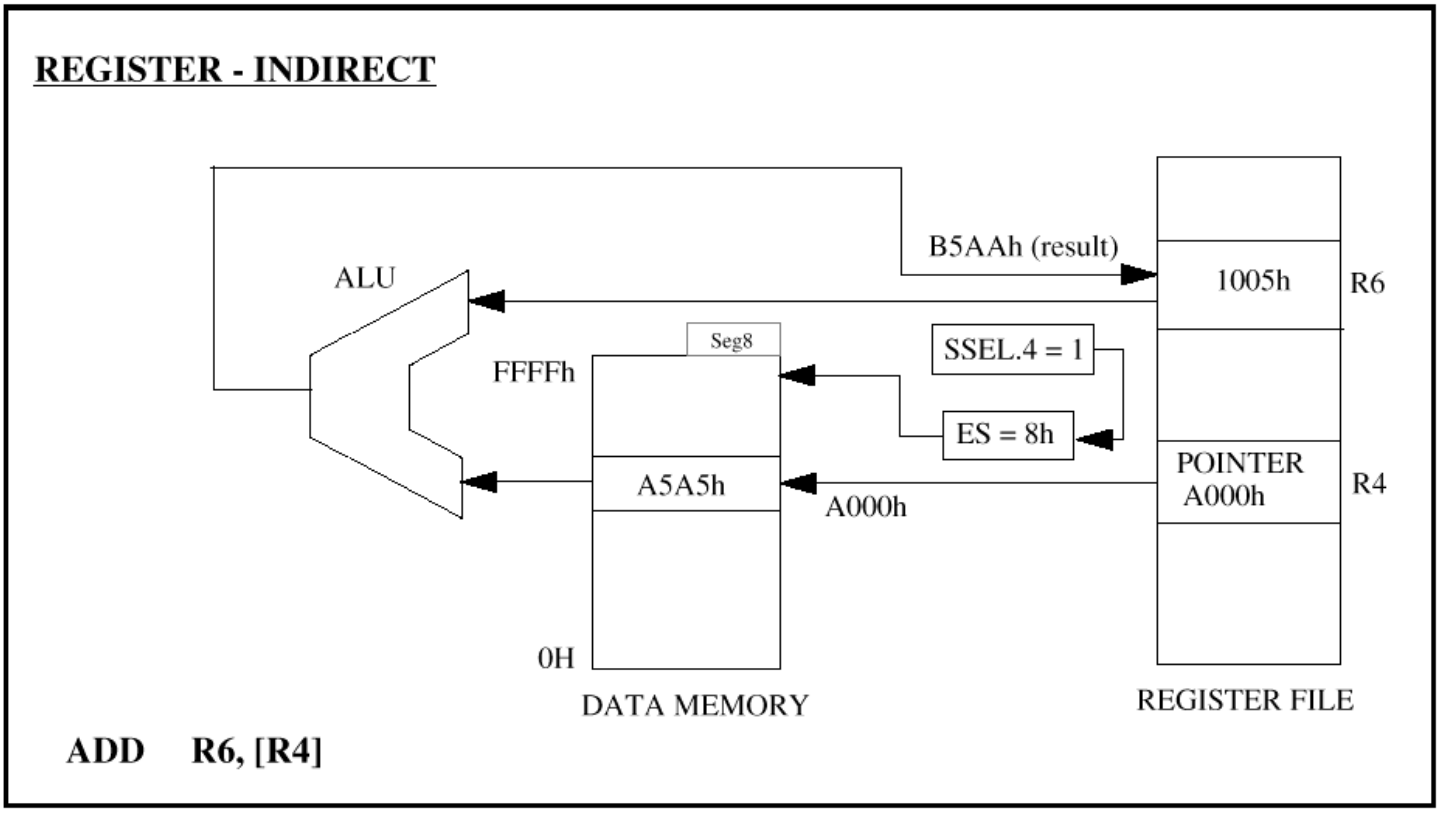



图 **6.2**

### 6.2.3 间接偏移寻址

这个寻址模式同以上的寄存器间接寻址类似,只是加上了一个附加偏移数值以得到最后的有效地址.使用这种寻址方式的指令包含一个16-bit的地址区域和一个8/16-bit的有符号的偏置区域.这个区域包含一个指针寄存器,它是寄存器队列中的8个指针寄存器之一,包含操作数的16-bit地址,该地址可以在任何的64K数据段中.指针寄存器的内容加上有符号的偏移量得到操作数的有效地址(必须是偶数).对于数据,段的选择由DS或ES决定;对于代码,由PC23-16或CS决定.根据相应的间接寄存器序号,SSEL.n=0,选择DS和PC; SSEL.n=1,选择ES和CS.对于字操作,操作数的地址必须是偶数.例如:

ADD R5, [R3 +30h]

操作以前: R3内容为C000h,R5内容为0065h, SSEL.3 = 1, ES=04,第4段C030h处的字为A540h

操作以后: R3内容为C000h, R5内容为A5A5h, 第4段C030h处的字为A540h

### 6.2.4 直接寻址

使用这种寻址方式的指令包含一个10-bit的地址区域,它包含操作数的实际空间地址(该地址可以处于数据空间的任何段或SFR空间).直接寻址数据空间总是处于一个段的底部的1K字节(0:3FFh).相关的段总是由DS决定.例如:

SUB R0, 200h

操作以前: R0内容为A5FFh,DS = 02, 第2段地址200H处的内容为5555h

操作以后: R0内容为50Aah, 第2段地址200H处的内容为5555h

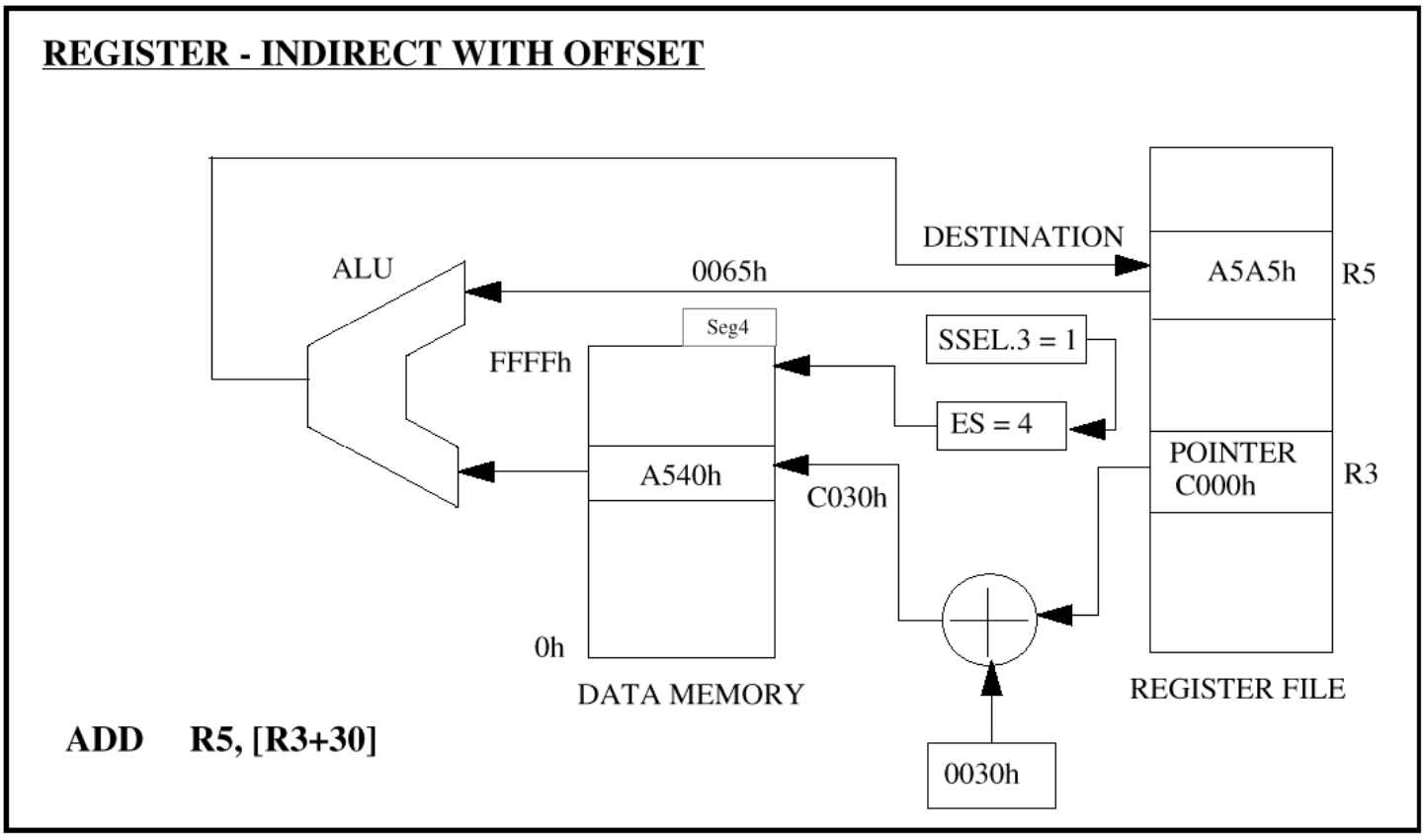



图 **6.3**

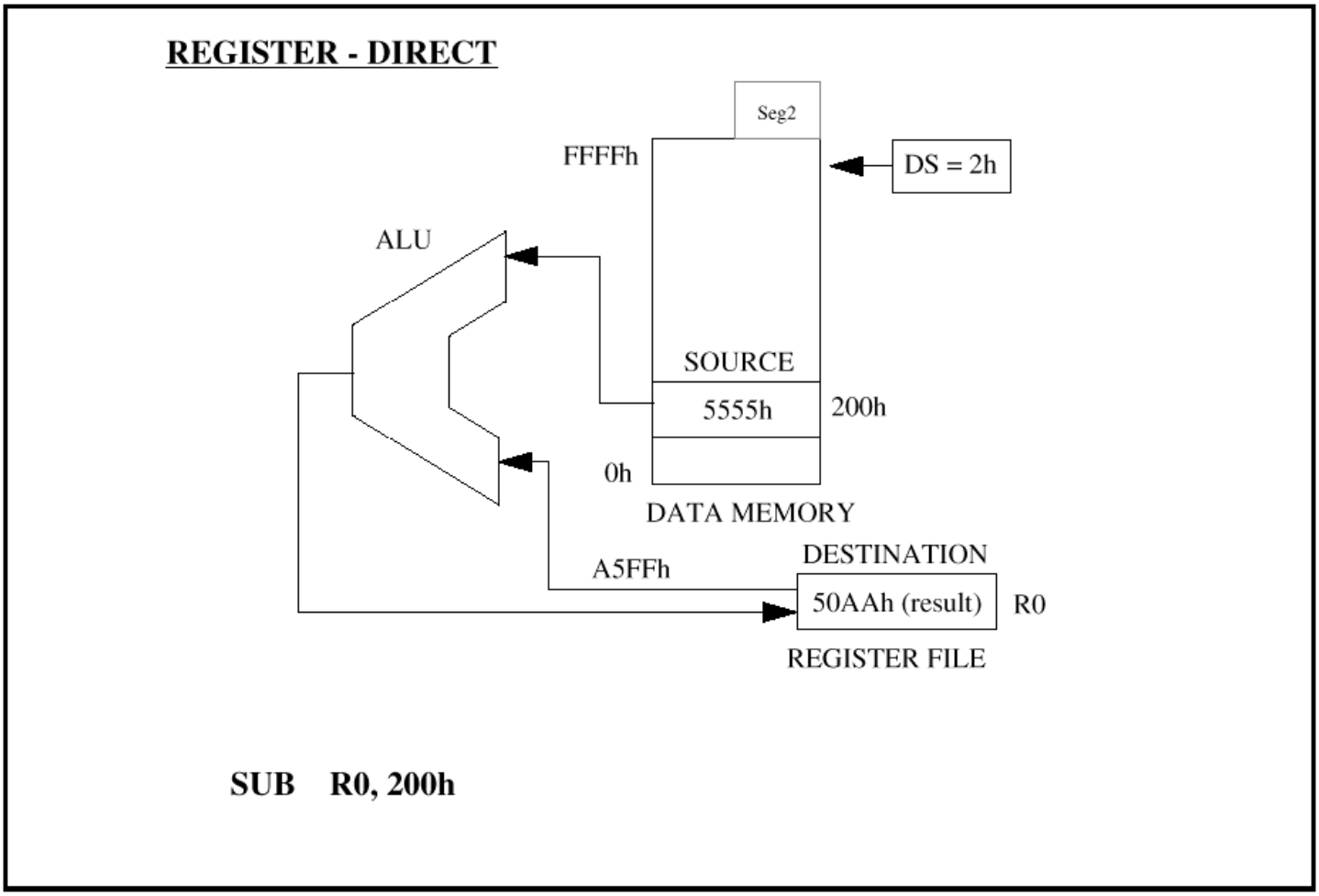



图 **6.4**

### 6.2.5 SFR寻址

这种寻址方式同上面的直接寻址方式类似,只是它寻址的是1K特殊功能寄存器空间.同以上的直接寻址模式相比,它具有译码后相同的指令区域,但实际上是分离的空间.1K的SFR空间总是直接寻址(400:7FFh),并定位于直接寻址RAM空间之上.例如:

MOV R0H, 406h 4

操作以前: R0H的内容为05h,地址406h的内容为A5h.

操作以后: R0H的内容为A5h,地址406h的内容为A5h.

### 6.2.6 立即寻址

在立即寻址中,实际的操作数在指令中明确的给出.源操作数是由4/5,8/16位的整数构成的.同指令ADDS和MOVS一起使用的4-bit短立即操作数是符号延伸.例如:

ADD R0L,#0B9h

操作以前: R0内容为13h.

操作以后: R0L内容为CCh.

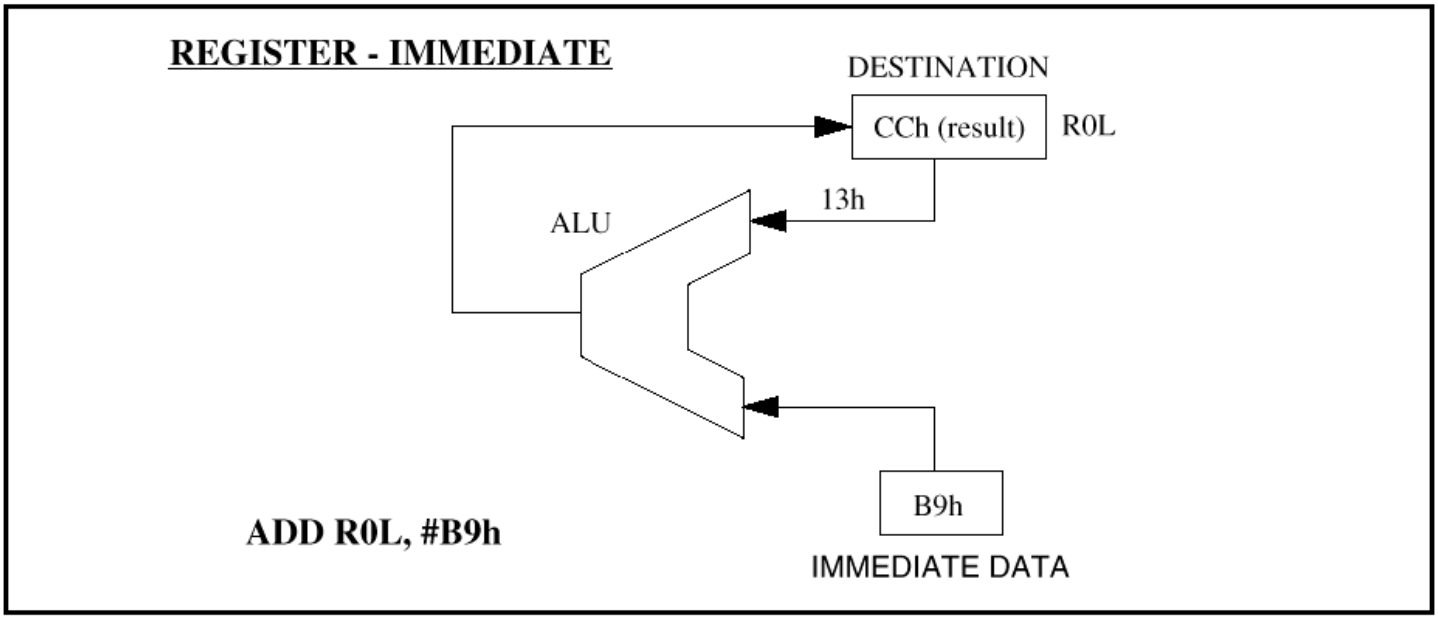



图 **6.5**

### **6.2.7** 位寻址

使用位寻址的指令包含一个10-bit的区域,它包含了位操作数的地址.XA的位寻址支持3种空间,都译码成相同的形式.这些空间是:寄存器队列的256个位;数据空间的256个位(位地址为20h-3Fh,当前数据段);SFR空间的512个位(位地址为400h-4FFh).

位地址00-FF位于寄存器队列,100-1FF位于数据区域,200-3FF位于SFR空间.

在直接寻址数据空间中,每一个段存在一个分离的位寻址空间.直接寻址空间处于的当前工作段由DS寄存器决定.

10-bit区域译码成位地址如下:

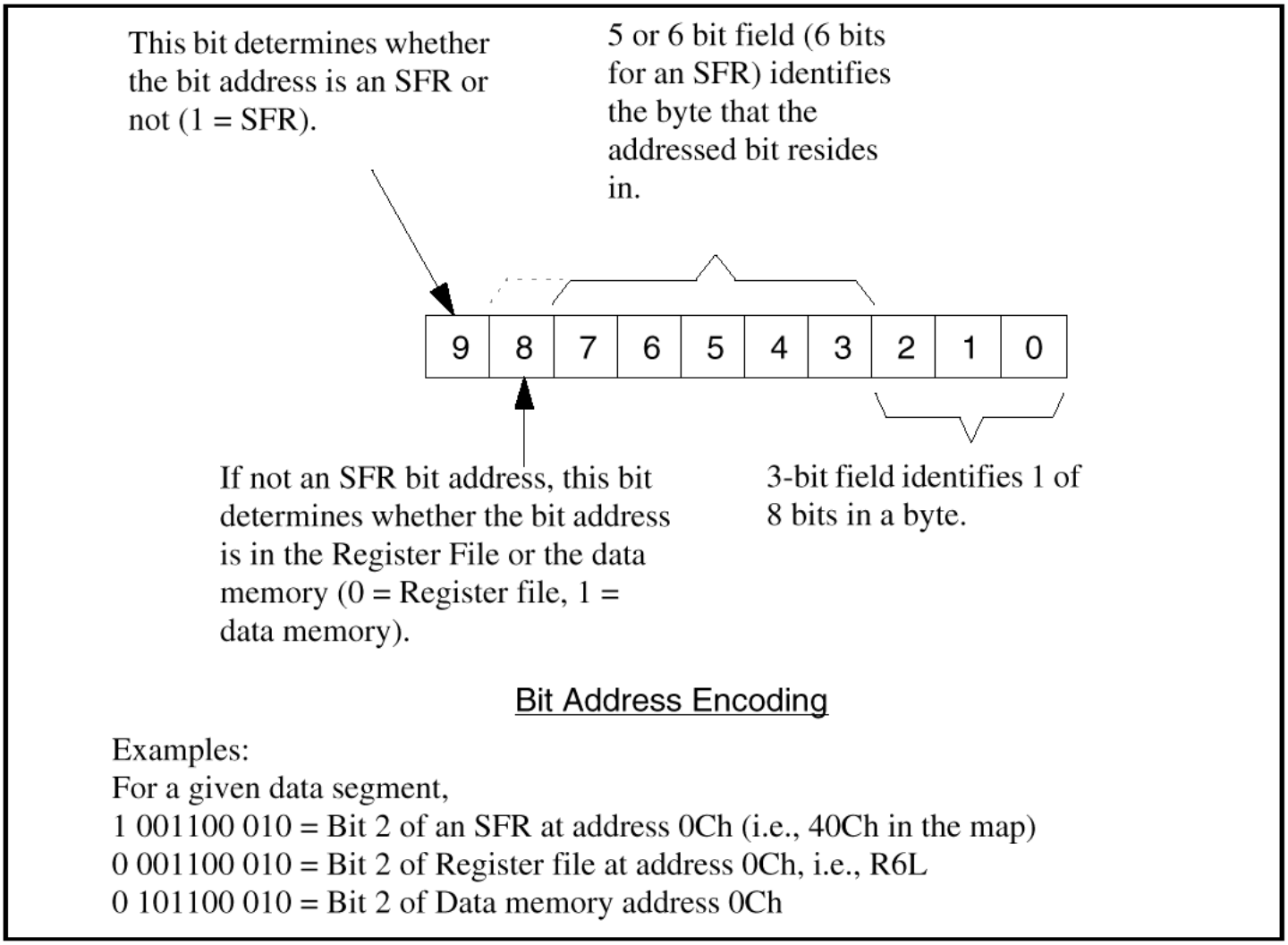



图**6.6**

## 6.3 相关分支和跳转

在XA中,可以用指令Jump,Call,Branch调用的程序地址,但必须是字相邻.例如,一个分支指令可以发生在地址的任何位置,但只能分支到偶数地址位置.强制相邻为偶数地址有3个优点:

■不用提供多余的位就能使分支范围扩大一倍。

■快速地执行，因为XA总是同时获取一个指令的头两个字节。

■允许已传输的80C51代码的分支仍处在程序内部，虽然在传输中代码趋于增大；使目标分支留在范围内的机会增大。

rel8偏移是一个9-bit的双互补整数，提供当前PC到目标PC的位移距离。相类似，rel16偏移是一个17-bit的双互补整数，提供当前PC到目标PC的位移距离。用于目标地址计算的PC值是分支指令(Jump,Call)的下一条指令的地址。

8-bit有符号的偏移量介于-128到127之间，rel8的分支范围(见下面的简单计算)为：指令处于偶数地址，实际上处于+254到-256；指令处于奇数地址，实际上处于+253到-257,满足了代码区域内目标地址是字相邻。

16-bit有符号的偏移量介于-32768到+32767之间。rel16的分支范围为：指令处于偶数地址，实际上处于+65534到-65536；指令处于奇数地址，实际上处于+65534到-65537，满足了代码区域内目标地址是字相邻。

Rel8范围的简单计算:

假定目标地址是字节相邻,从当前PC向前的范围是:

分支目标地址-当前PC

现在,最大的带符号正数=+127;所以rel8 << 1 为+254

如果当前PC = 奇数，则

范围 = 254 - 1 = +253 前进范围是奇数

如果当前PC = 偶数，则

范围 = +254 前进范围是偶数

同样,保持同样的范围:

分支目标地址-当前PC
现在,最大的带符号正数=-128;所以rel8 << 1 为-256
如果当前PC = 奇数, 则
范围 = -257 前进范围是奇数
如果当前PC = 偶数, 则
范围 = +256 前进范围是偶数

## 6.4 XA的数据类型

XA使用下列的数据类型:
■位。
■4/5-bit有符号整数。
■8-bit(字节)有符号和无符号整数。
■8-bit双BCD数。
■16-bit(字) 有符号和无符号整数。
■在数据空间和SFR空间的位寻址的10-bit地址。
■由16-bit地址和8-bit段选择组成的24-bit有效地址。详情见寻址方式。
一个字节由8个bit位组成。字有16个bit位，包含两个连续字节。双字由两个连续的16bit的字组成。
负整数为补码形式。4-bit有符号整数(符号延伸到字节/字)用做MOVS和ADDS指令的立即操作数。
每个字节可以有两个进制代码十进制数。BCD操作使用字节运算符。

## 6.5 指令集概述

XA使用强大和有效的指令集,提供7种不同类型的寻址方式.基于寄存器和存储器内容的通用"Branch"和"Jump"可以控制程序的流向.强化后的指令支持结构化的高级语言和实时多任务系统.
这一节讨论XA提供的指令集,并演示如何使用.它包括指令形式,指令使用操作数的解释.在根据指令种类进行概述以后,本节还对每一个指令的操作进行详细的解释,根据字母顺序.
提供5个总表解释现有的指令.
第一个表是根据XA指令通常用途来排列的.
第二个表和第三个表是根据寻址方式,操作数,和指令类型来排列的.
第四个表列出全部状态标志被不同指令更新的情况.
第五个表列出拥有不同寻址方式的指令,概述了每一条指令需要的字节数目,运行的时钟周期.
这些指令的长度和运行速度均经过优化,可以使用在较严格的代码中.指令中参加操作译码的仅仅是第一个字节和有时候是第二字节.指令的长度和第一个执行周期可以由第一个字节确定.指令字节的头两个以后,一般是立即操作数,相对偏移,偏移,位地址,立即数.

助记符,符号使用的术语
通常:

| | |
|---|---|
| offset8 | 一个8-bit有符号偏移(指令中的立即数),加到一个寄存器上产生一个绝对地址. |
| offset16 | 一个16-bit有符号偏移(指令中的立即数),加到一个寄存器上产生一个绝对地址. |
| direct | 一个包含在指令中的11-bit立即地址. |
| #data4 | 一个包含在指令中的4-bit立即数.(有符号立即数,范围+7到-8;移位,0-15) |
| #data5 | 一个包含在指令中的5-bit立即数.(移位,0-31) |
| #data8 | 一个包含在指令中的8-bit立即数.(+127到-128) |
| #data16 | 一个包含在指令中的16-bit立即数.(+32767到-32768) |
| bit | 可位寻址的10-bit地址. |
| rel8 | 分支中的8-bit相对偏移(+254 到 -256). |
| rel16 | 分支中的16-bit相对偏移(+65534到-65536). |
| addr16 | 在64K代码段内的16-bit绝对分支地址. |
| addr24 | 24-bit绝对分支地址,能操作整个XA地址空间. |
| SP | 当前操作指针(用户或系统),根据当前的操作模式. |
| USP | 用户堆栈指针 |

SSP 系统堆栈指针.
C PSW中的进位标志.
AC PSW中的辅助进位标志.
V PSW中的溢出标志.
N PSW中的负数标志.
Z PSW中的为0标志.
DS 数据段寄存器.保持XA中24-bit数据空间的高字节地址.主要用于本地数据段.
ES 附加段寄存器. 保持XA中24-bit数据空间的高字节地址.主要用于寻址远距离数据结构.
direct 使用当前DS形成操作数据空间的24-bit地址;或只使用16-bit地址操作SFR空间.解释如下:
如果(数据空间)则direct=(DS:direct)
如果为(SFR空间)则(direct)=(sfr)

操作译码区域:

SZ 数据尺寸.这个区域译码后不管操作数为字节,字,双字.
IND 这个区域在一些指令中表明间接操作.
H/L 这个区域选择是否PUSH和POP R 使用寄存器的高或低半部分.
dddd 目标寄存器栏,在寄存器队列中指定16个寄存器当中的一个.
ddd 间接调用的目标寄存器栏,在寄存器队列中指定8个指针寄存器的一个.
ssss 源寄存器栏, 在寄存器队列中指定16个寄存器当中的一个.
sss 间接调用的寄存器栏, 在寄存器队列中指定8个指针寄存器的一个.

助记符:

RS 源寄存器
Rd 目标寄存器
[ ] 在指令助记符中,表明一个间接调用(例如,[R4]调用由寄存器R4内容指向地址的内容)
[R+] 在间接调用中,表明寄存器指针自动增加.
[WS:R] 表明指针寄存器由选择的段寄存器扩展到24-bit指针(DS或ES,除了MOVC的所有指令;MOVC使用$PC_{23-16}$或CS).
Rlist 位图.表明在PUSH或POP操作中,寄存器队列中每个寄存器提供的情况.这些寄存器是R0到R7(字),或R0L到R7H(字节).

伪代码

( ) 在指令操作伪代码中用来表明"…的内容"(例如,(R4)调用寄存器R4的内容).
<--- 伪代码分配.有时候使用<-->表明双向分配(数据交换)
((SP)) 由当前数据指针指向的数据区域的内容.在系统模式下,当前的SP是SSP,使用0段.在用户模式下,当前的SP是USP,当前段由DS决定.这个段用于SP的使用,不仅仅是PUSH和POP.有时候,((SSP))或((USP))指明使用指定的SP,不管当前的操作模式.
Rn.x 表示寄存器n的第x个位.
Rn.x-y 表示寄存器n从第x个位到第y个位的范围.

注:如果不做特别说明,所有间接寻址都使用数据段寄存器作为高8位地址.例如,[ES:RS]表明用附加段寄存器作为高8位地址.

执行时间

PZ 在0页模式
nt 未发生的
t 发生的

操作数宽度语法

**.w** = 字操作数
**.b** = 字节操作数
**.d** = 双字操作数

缺省操作数尺寸要根据操作数的使用.例如,MOV R0,R1总是字宽度,而MOV R0L,R0H总是字节宽度.对于间接-立即,直接-立即,直接-直接等情况,用户必须指定操作数宽度.

其它

0x = 16进制数值前缀

[] = 间接地址.

[[]] = 双间接地址

dest = 目标

src = 源

**XA的指令集**

| 助记符 | 用途 |
|---|---|
| MOV, MOVC, MOVS, MOVX, LEA, XCH, PUSH, POP, PUSHU, POPU | 数据传输 |
| ADD, ADDS, ADDC, SUB, SUBB | 数学加减 |
| MULU.b, MULU.w, MUL.w DIVU.b, DIVU.w, DIVU.d, DIV.w, DIV.d | 数学乘除 |
| RR, RRC, RL, RLC, LSR, ASR, ASL, NORM | 移位和旋转 |
| CLR, SETB, MOV, ANL, ORL | 位操作 |
| JB, JBC, JNB, JNZ, JZ, DJNZ, CJNE | 条件跳转/调用 |
| BOV, BNV, BPL, BCC, BCS, BEQ, BNE, BG, BGE, BGT, BL, BLE, BLT, BMI | 条件分支 |
| AND, OR, XOR | 布尔功能 |
| JMP, FJMP, CALL, FCALL, BR | 无条件跳转/调用/分支 |
| RET, RETI | 从子程序,中断返回 |
| SEXT, NEG, CPL, DA | 符号延伸,取反,取补,十进制调整 |
| BKPT, TRAP#, RESET | 异常中断 |
| NOP | 空操作 |

表**6.3** 显示了数据传输和计算具备的基本寻址模式和相关指令

| 模式/操作数 | **MOVX** | **MOV** | **CMP** | **ADD ADDC** | **SUB SUBB** | **AND OR XOR** | **ADDS MOVS** | **MUL DIV** | 移位 | **XCH** | 字节 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| R, R | | ● | ● | ● | ● | ● | | ● | ● | ● | 2 |
| R, [R] | ● | ● | ● | ● | ● | ● | | | | ● | 2 |
| [R], R | ● | ● | ● | ● | ● | ● | | | | | 2 |
| R, [R+off8] | | ● | ● | ● | ● | ● | | | | | 3 |
| [R+off8], R | | ● | ● | ● | ● | ● | | | | | 3 |
| R, [R+off16] | | ● | ● | ● | ● | ● | | | | | 4 |
| [R+off16], R | | ● | ● | ● | ● | ● | | | | | 4 |
| R, [R+] | | ● | ● | ● | ● | ● | | | | | 2 |
| [R+], R | | ● | ● | ● | ● | ● | | | | | 2 |
| [R+], [R+] | | ● | | ● | ● | ● | | | | | 2 |
| dir, R | | ● | ● | ● | ● | ● | | | | | 3 |
| R, dir | | ● | ● | ● | ● | ● | | | | ● | 3 |
| dir, [R] | | ● | | | | | | | | | 3 |
| [R], dir | | ● | | | | | | | | | 3 |
| R, #data | | ● | ● | ● | ● | ● | ● | ● | ● | | 2*/3/4 |
| [R], #data | | ● | ● | ● | ● | ● | ● | | | | 2*/3/4 |
| [R+], #data | | ● | ● | ● | ● | ● | ● | | | | 2*/3/4 |
| **[**R+off8], #data | | ● | ● | ● | ● | ● | ● | | | | 3*/4/5 |
| **[**R+off16], #data | | ● | ● | ● | ● | ● | ● | | | | 4*/5/6 |
| dir, #data | | ● | ● | ● | ● | ● | ● | | | | 3*/4/5 |
| dir, dir | | ● | | | | | | | | | 4 |
| R, USP | | ● | | | | | | | | | 2 |

注**:**

-移位种类包括旋转,移位,正常化.

-USP=用户堆栈指针

*:ADDS和MOVS使用一个短立即数(4 bit)

**BKPT, NOP, RESET, RET, RETI无立即数.

| 模式/操作数 | MOVC | PUSH POP | SEXT,CPL ,NEG,DA | JUMPCALL | DJNZ | CJNE | 位操作 | 其它 | 字节 |
|---|---|---|---|---|---|---|---|---|---|
| R, [R+] | ● | | | | | | | | 2 |
| [R+], R | ● | | | | | | | | 2 |
| A, [A+DPTR] | ● | | | | | | | | 2 |
| A, [A+PC] | ● | | | | | | | | 2 |
| direct | | ● | | | | | | | 3 |
| Rlist | | ● | | | | | | | 2 |
| R | | | ● | | | | | | 2 |
| addr24 | | | | ● | | | | | 4 |
| [R] | | | | ● | | | | | 2 |
| [A+DPTR] | | | | JMP | | | | | 2 |
| R, rel | | | | | ● | | | | 3 |
| direct, rel | | | | | ● | | | | 4 |
| R, direct, rel | | | | | | ● | | | 4 |
| R, #data, rel | | | | | | ● | | | 4/5 |
| [R], #data, rel | | | | | | ● | | | 4/5 |
| bit | | | | | | | ● | | **3** |
| bit, C; C, bit | | | | | | | ● | | **3** |
| C, /bit | | | | | | | ● | 条件分支 | **3** |
| rel | | | | | | | | 条件分支 | 2 |
| bit, rel | | | | | | | | TRAP | 4 |
| #data4 | | | | | | | | TRAP | 2 |
| R, R+off8 | | | | | | | | LEA | 3 |
| r, R+off16 | | | | | | | | LEA | 4 |
| <none> ** | | | | | | | | ● | 1/2 |

注**:**

-移位种类包括旋转,移位,正常化.

-USP=用户堆栈指针

*:ADDS和MOVS使用一个短立即数(4 bit)

**BKPT, NOP, RESET, RET, RETI无立即数.

表 **6.4** 状态标志的更新

| 指令类型 | 标志更新 | | | | |
|---|---|---|---|---|---|
| | **C** | **AC** | **V** | **N** | **Z** |
| ADD, ADDC, CMP, SUB, SUBB | X | X | X | X | X |
| ADDS, MOVS | - | - | - | X | X |
| AND, OR, XOR | - | - | - | X | X |
| ASR, LSR | * | - | - | X | X |
| 分支所有位操作,NOP | - | - | - | - | - |
| Calls, Jumps, and Returns | - | - | - | - | - |
| CJNE | X | - | - | X | X |
| CPL | - | - | - | X | X |
| DA | * | - | - | X | X |
| DIV, MUL | * | - | * | X | X |
| DJNZ | - | - | - | X | X |
| LEA | - | - | - | - | - |
| MOV, MOVC, MOVX | - | - | - | X | X |
| NEG | - | - | X | X | X |
| NORM | - | - | - | X | X |
| PUSH, POP | - | - | - | - | - |
| RESET | * | * | * | * | * |
| RL, RR | - | - | - | * | * |
| RLC, RRC | * | - | - | X | X |
| SEXT | - | - | - | - | - |
| TRAP, BKPT | - | - | - | - | - |

| | | | | | |
|---|---|---|---|---|---|
| XCH | - | - | - | - | - |
| ASL | * | - | X | X | X |

注:

-:标志位不更新

X:根据标准定义进行更新

*:标志位更新不标准,参考各自的指令解释.

注意:对PSW的改写将优先于标志更新.

## 指令集概要

表6.5根据指令的类型列出了全部的XA指令.此表可以作为查找某一指令的快速参考,然后再参照详细的按字母排列的指令解释.

表中给定的指令时间和后面的详细指令描述都是基于内部代码执行和操作内部数据RAM和寄存器. 由于操作外部代码和数据的可编程性能,以及pipeline功能的作用,所有条件下的时序不可能写成简单的形式.这里给定的时序数据还假定指令已经读到予获取队列中而不需要等待.在分支情况下,一个片内获取已经包含在时间数中.如果分支后到达的指令不超过2个字节,则给定的时间仍然是有效的.如果是长指令,CPU将进行等待,直到予获取队列包含了全部的指令后才开始执行.这将需要1到2个附加代码获取,由于XA的最长指令是6个字节.

下面汇集了一些可能引起同给定时间有差异的事件和条件.它们通常是延迟,直到XA等待的一些信息变为有效.

■指令获取。如果予获取队列不包含需要的完整指令。执行就必须等待。除了下面是分支以外。队列的状态依赖与前面执行指令的情况。

■指令顺序相关。典型的例子是，当一个指令从SFR总线或外部数据总线读取了数据，下面的指令又对源目标施加写操作。在读完成以前，CPU必须等待，然后才能进行写操作。在读完成以前，不能恢复执行。

■内部数据对SFR的操作。SFR的读需要两个附加的时钟周期，因为XA外设工作于CPU时钟的2分频。需要一个时钟周期同步CPU和SFR总线。

■程序流向变化。当程序的流向发生变化，XA必须清除予获取队列并从新的执行地址处重新调入。所有的分支，跳转，调用的发布时间包含一个内部代码获取。如果新地址处的指令大于两个字节，要增加一个附加的获取周期使队列获得一个完整的指令。从中断和字程序返回时，由于返回可能到达原来的代码地址，所以第一次代码获取可能只取得1个字节。

■内部对外部代码执行。可编程外部总线时序和其它总线事项将引起内部和外部代码操作时间的差异。使用8-bit外部总线执行外部代码将会对整体性能有影响。使用WAIT信号也会导致附加的变化。外部总线在16-bit地址边界处需要一个ALE信号，当程序流向发生变化,或执行了一个外部总线操作。都会对时序产生影响.8-bit数据总线的设置会引起差异，使用WAIT信号则引起的时间差异更大。

■内部对外部数据操作。可编程的外部总线时序同样会引起内部数据和外部数据操作时间上的差异。

表**6.5**

| 助记符 | 解释 | 字节 | 周期 |
|---|---|---|---|
| 数学运算 | | | |
| ADD Rd, Rs | 寄存器直接加 | 2 | 3 |
| ADD Rd, [Rs] | 寄存器间接-寄存器加 | 2 | 4 |
| ADD [Rd], Rs | 寄存器-寄存器间接加 | 2 | 4 |
| ADD Rd, [Rs+offset8] | 8-bit偏移间接寄存器-寄存器加 | 3 | 6 |
| ADD [Rd+offset8], Rs | 寄存器-8-bit偏移间接寄存器加 | 3 | 6 |
| ADD Rd, [Rs+offset16] | 16-bit偏移间接寄存器-寄存器加 | 4 | 6 |
| ADD [Rd+offset16], Rs | 寄存器-16-bit偏移间接寄存器加 | 4 | 6 |
| ADD Rd, [Rs+] | 自动增量间接寄存器-寄存器加 | 2 | 5 |
| ADD [Rd+], Rs | 寄存器-自动增量间接寄存器加 | 2 | 6 |
| ADD direct, Rs | 寄存器-直接地址加 | 3 | 4 |
| ADD Rd, direct | 直接地址-寄存器加 | 3 | 4 |
| ADD Rd, #data8 | 8-bit立即数-寄存器加 | 3 | 3 |
| ADD Rd, #data16 | 16-bit立即数-寄存器加 | 4 | 3 |
| ADD [Rd], #data8 | 8-bit立即数-寄存器间接加 | 3 | 4 |
| ADD [Rd], #data16 | 16-bit立即数-寄存器间接加 | 4 | 4 |
| ADD [Rd+], #data8 | 8-bit立即数-自动增量寄存器间接加 | 3 | 5 |

| | | | |
|---|---|---|---|
| ADD [Rd+], #data16 | 16-bit立即数-自动增量寄存器间接加 | 4 | 5 |
| ADD [Rd+offset8], #data8 | 8-bit立即数-8-bit偏移间接寄存器加 | 4 | 6 |
| ADD [Rd+offset8], #data16 | 16-bit立即数-8-bit偏移间接寄存器加 | 5 | 6 |
| ADD [Rd+offset16], #data8 | 8-bit立即数-16-bit偏移间接寄存器加 | 5 | 6 |
| ADD [Rd+offset16], #data16 | 16-bit立即数-16-bit偏移间接寄存器加 | 6 | 6 |
| ADD direct, #data8 | 8-bit立即数-直接地址加 | 4 | 4 |
| ADD direct, #data16 | 16-bit立即数-直接地址加 | 5 | 4 |
| ADDC Rd, Rs | 带进位寄存器加 | 2 | 3 |
| ADDC Rd, [Rs] | 带进位寄存器间接-寄存器加 | 2 | 4 |
| ADDC [Rd], Rs | 带进位寄存器-寄存器间接加 | 2 | 4 |
| ADDC Rd, [Rs+offset8] | 带进位8-bit偏移寄存器间接-寄存器加 | 3 | 6 |
| ADDC [Rd+offset8], Rs | 带进位寄存器-8-bit偏移寄存器间接 | 3 | 6 |
| ADDC Rd, [Rs+offset16] | 带进位16-bit偏移寄存器间接-寄存器加 | 4 | 6 |
| ADDC [Rd+offset16], Rs | 带进位寄存器-16-bit偏移寄存器间接 | 4 | 6 |
| ADDC Rd, [Rs+] | 带进位自动增量寄存器间接-寄存器加 | 2 | 5 |
| ADDC [Rd+], Rs | 带进位寄存器-自动增量寄存器间接 | 2 | 5 |
| ADDC direct, Rs | 带进位寄存器-直接地址加 | 3 | 4 |
| ADDC Rd, direct | 带进位直接地址-寄存器加 | 3 | 4 |
| ADDC Rd, #data8 | 带进位8-bit立即数-寄存器加 | 3 | 3 |
| ADDC Rd, #data16 | 带进位16-bit立即数-寄存器加 | 4 | 3 |
| ADDC [Rd], #data8 | 带进位8-bit立即数-寄存器间接加 | 3 | 4 |
| ADDC [Rd], #data16 | 带进位16-bit立即数-寄存器间接加 | 4 | 4 |
| ADDC [Rd+], #data8 | 带进位8-bit立即数-自动增量寄存器间接加 | 3 | 5 |
| ADDC [Rd+], #data16 | 带进位16-bit立即数-自动增量寄存器间接加 | 4 | 5 |
| ADDC [Rd+offset8], #data8 | 带进位16-bit立即数-8-bit偏移寄存器间接加 | 4 | 6 |
| ADDC [Rd+offset8], #data16 | 带进位8-bit立即数-8-bit偏移寄存器间接加 | 5 | 6 |
| ADDC [Rd+offset16], #data8 | 带进位8-bit立即数-16-bit偏移寄存器间接加 | 5 | 6 |
| ADDC [Rd+offset16], #data16 | 带进位16-bit立即数-16-bit偏移寄存器间接加 | 6 | 6 |
| ADDC direct, #data8 | 带进位8-bit立即数-直接地址加 | 4 | 4 |
| ADDC direct, #data16 | 带进位16-bit立即数-直接地址加 | 5 | 4 |
| ADDS Rd, #data4 | 4-bit立即数-寄存器加 | 2 | 3 |
| ADDS [Rd], #data4 | 4-bit立即数-寄存器间接加 | 2 | 4 |
| ADDS [Rd+], #data4 | 4-bit立即数-自动增量寄存器间接加 | 2 | 5 |
| ADDS [Rd+offset8], #data4 | 4-bit立即数-8-bit偏移寄存器间接加 | 3 | 6 |
| ADDS [Rd+offset16], #data4 | 4-bit立即数-16-bit偏移寄存器间接加 | 4 | 6 |
| ADDS direct, #data4 | 4-bit立即数-直接地址 | 3 | 4 |
| ASL Rd, Rs | 目标寄存器逻辑左移,移位次数由源寄存器决定. | 2 | 见注1 |
| ASL Rd, #data4 | 寄存器逻辑左移,移位次数由4-bit立即数决定. | 2 | 见注1 |
| ASR Rd, Rs | 目标寄存器算术右移,移位次数由源寄存器决定. | 2 | 见注1 |
| ASR Rd, #data4 | 寄存器算术右移,移位次数由4-bit立即数决定. | 2 | 见注1 |
| CMP Rd, Rs | 比较源寄存器和目标寄存器 | 2 | 3 |
| CMP [Rd], Rs | 比较寄存器和寄存器间接 | 2 | 4 |
| CMP Rd, [Rs] | 比较寄存器间接和寄存器 | 2 | 4 |
| CMP [Rd+offset8], Rs | 比较寄存器和8-bit偏移寄存器间接 | 3 | 6 |
| CMP [Rd+offset16], Rs | 比较寄存器和16-bit偏移寄存器间接 | 4 | 6 |
| CMP Rd, [Rs+offset8] | 比较8-bit偏移寄存器间接和寄存器 | 3 | 6 |
| CMP Rd,[Rs+offset16] | 比较16-bit偏移寄存器间接和寄存器 | 4 | 6 |
| CMP Rd, [Rs+] | 比较自动增量寄存器间接-寄存器 | 2 | 5 |
| CMP [Rd+], Rs | 比较寄存器和自动增量寄存器间接 | 2 | 5 |
| CMP direct, Rs | 比较寄存器和直接地址 | 3 | 4 |
| CMP Rd, direct | 比较直接地址和寄存器 | 3 | 4 |
| CMP Rd, #data8 | 比较8-bit立即数和寄存器 | 3 | 3 |
| CMP Rd, #data16 | 比较16-bit立即数和寄存器 | 4 | 3 |
| CMP [Rd], #data8 | 比较8-bit立即数和寄存器间接 | 3 | 4 |
| CMP [Rd], #data16 | 比较16-bit立即数和寄存器间接 | 4 | 4 |

| | | | |
|---|---|---|---|
| CMP [Rd+], #data8 | 比较8-bit立即数和自动增量寄存器间接 | 3 | 5 |
| CMP [Rd+], #data16 | 比较16-bit立即数和自动增量寄存器间接 | 4 | 5 |
| CMP [Rd+offset8], #data8 | 比较8-bit立即数和8-bit偏移寄存器间接 | 4 | 6 |
| CMP [Rd+offset8], #data16 | 比较16-bit立即数和8-bit偏移寄存器间接 | 5 | 6 |
| CMP [Rd+offset16], #data8 | 比较8-bit立即数和16-bit偏移寄存器间接 | 5 | 6 |
| CMP [Rd+offset16], #data16 | 比较16-bit立即数和16-bit偏移寄存器间接 | 6 | 6 |
| CMP direct, #data8 | 比较8-bit立即数和直接地址 | 4 | 4 |
| CMP direct, #data16 | 比较16-bit立即数和直接地址 | 5 | 4 |
| DA Rd | 十进制调整字节寄存器 | 2 | 4 |
| DIV.w Rd, Rs | 16X8有符号寄存器除 | 2 | 14 |
| DIV.w Rd, #data8 | 16X8有符号寄存器除以8-bit立即数字节 | 3 | 14 |
| DIV.d Rd, Rs | 32X16有符号双寄存器除 | 2 | 24 |
| DIV.d Rd, #data16 | 32X16有符号双寄存器除以/16-bit立即数 | 4 | 24 |
| DIVU.b Rd, Rs | 8X8无符号寄存器除 | 2 | 12 |
| DIVU.b Rd, #data8 | 8X8无符号寄存器除以立即数字节 | 3 | 12 |
| DIVU.w Rd, Rs | 16X8无符号寄存器除 | 2 | 12 |
| DIVU.w Rd, #data8 | 6X8无符号寄存器除以立即数字节 | 3 | 12 |
| DIVU.d Rd, Rs | 32X16无符号双寄存器除 | 2 | 22 |
| DIVU.d Rd, #data16 | 32X16无符号双寄存器除以立即数字 | 4 | 22 |
| LEA Rd, Rs+offset8 | 调入有8-bit偏移的16-bit有效地址到寄存器 | 3 | 3 |
| LEA Rd, Rs+offset16 | 调入有16-bit偏移的16-bit有效地址到寄存器 | 4 | 3 |
| MUL.w Rd, Rs | 16X16有符号寄存器乘 | 2 | 12 |
| MUL.w Rd, #data16 | 16X16有符号寄存器乘以16-bit数据 | 4 | 12 |
| MULU.b Rd, Rs | 8X8无符号寄存器乘 | 2 | 12 |
| MULU.b Rd, #data8 | 8X8无符号寄存器乘以8-bit立即数 | 3 | 12 |
| MULU.w Rd, Rs | 16X16无符号寄存器乘 | 2 | 12 |
| MULU.w Rd, #data16 | 16X16无符号寄存器乘以16-bit立即数 | 4 | 12 |
| NEG Rd | 寄存器取补 | 2 | 3 |
| SEXT Rd | 上次操作符号位作用到寄存器 | 2 | 3 |
| SUB Rd, Rs | 寄存器减 | 2 | 3 |
| SUB Rd, [Rs] | 寄存器减去寄存器间接 | 2 | 4 |
| SUB [Rd], Rs | 寄存器间接减去寄存器 | 2 | 4 |
| SUB Rd, [Rs+offset8] | 寄存器减去8-bit偏移寄存器间接 | 3 | 6 |
| SUB [Rd+offset8], Rs | 8-bit偏移寄存器间接减去寄存器 | 3 | 6 |
| SUB Rd, [Rs+offset16] | 寄存器减去16-bit偏移寄存器间接 | 4 | 6 |
| SUB [Rd+offset16], Rs | 16-bit偏移寄存器间接减去寄存器 | 4 | 6 |
| SUB Rd, [Rs+] | 寄存器减去寄存器间接 | 2 | 5 |
| SUB [Rd+], Rs | 寄存器间接减去寄存器 | 2 | 5 |
| SUB direct, Rs | 直接地址减去寄存器 | 3 | 4 |
| SUB Rd, direct | 寄存器减去直接地址 | 3 | 4 |
| SUB Rd, #data8 | 寄存器减去8-bit立即数 | 3 | 3 |
| SUB Rd, #data16 | 寄存器减去16-bit立即数 | 4 | 3 |
| SUB [Rd], #data8 | 寄存器间接减去8-bit立即数 | 3 | 4 |
| SUB [Rd], #data16 | 寄存器间接减去16-bit立即数 | 4 | 4 |
| SUB [Rd+], #data8 | 自动增量寄存器间接减去8-bit立即数 | 3 | 5 |
| SUB [Rd+], #data16 | 自动增量寄存器间接减去16-bit立即数 | 4 | 5 |
| SUB [Rd+offset8], #data8 | 8-bit偏移寄存器间接减去8-bit立即数 | 4 | 6 |
| SUB [Rd+offset8], #data16 | 8-bit偏移寄存器间接减去16-bit立即数 | 5 | 6 |
| SUB [Rd+offset16], #data8 | 16-bit偏移寄存器间接减去8-bit立即数 | 5 | 6 |
| SUB [Rd+offset16], #data16 | 16-bit偏移寄存器间接减去16-bit立即数 | 6 | 6 |
| SUB direct, #data8 | 直接地址减去8-bit立即数 | 4 | 4 |
| SUB direct, #data16 | 直接地址减去16-bit立即数 | 5 | 4 |
| SUBB Rd, Rs | 带借位寄存器减 | 2 | 3 |
| SUBB Rd, [Rs] | 带借位寄存器减去寄存器间接 | 2 | 4 |
| SUBB [Rd], Rs | 带借位寄存器间接减去寄存器 | 2 | 4 |

| | | | |
|---|---|---|---|
| SUBB Rd, [Rs+offset8] | 带借位寄存器减去8-bit偏移寄存器间接 | 3 | 6 |
| SUBB [Rd+offset8], Rs | 带借位8-bit偏移寄存器间接减去寄存器 | 3 | 6 |
| SUBB Rd, [Rs+offset16] | 带借位寄存器减去16-bit偏移寄存器间接 | 4 | 6 |
| SUBB [Rd+offset16], Rs | 带借位16-bit偏移寄存器间接减去寄存器 | 4 | 6 |
| SUBB Rd, [Rs+] | 带借位寄存器减去自动增量寄存器间接 | 2 | 5 |
| SUBB [Rd+], Rs | 带借位自动增量位寄存器间接减去寄存器 | 2 | 5 |
| SUBB direct, Rs | 带借位直接地址减去寄存器 | 3 | 4 |
| SUBB Rd, direct | 带借位寄存器减去直接地址 | 3 | 4 |
| SUBB Rd, #data8 | 带借位寄存器减去8-bit立即数 | 3 | 3 |
| SUBB Rd, #data16 | 带借位寄存器减去16-bit立即数 | 4 | 3 |
| SUBB [Rd], #data8 | 带借位寄存器间接减去8-bit立即数 | 3 | 4 |
| SUBB [Rd], #data16 | 带借位寄存器间接减去16-bit立即数 | 4 | 4 |
| SUBB [Rd+], #data8 | 带借位自动增量寄存器间接减去8-bit立即数 | 3 | 5 |
| SUBB [Rd+], #data16 | 带借位自动增量寄存器间接减去16-bit立即数 | 4 | 5 |
| SUBB [Rd+offset8], #data8 | 带借位8-bit偏移寄存器间接减去8-bit立即数 | 4 | 6 |
| SUBB [Rd+offset8], #data16 | 带借位8-bit偏移寄存器间接减去16-bit立即数 | 5 | 6 |
| SUBB [Rd+offset16], #data8 | 带借位16-bit偏移寄存器间接减去8-bit立即数 | 5 | 6 |
| SUBB [Rd+offset16], #data16 | 带借位16-bit偏移寄存器间接减去16-bit立即数 | 6 | 6 |
| SUBB direct, #data8 | 带借位直接地址减去8-bit立即数 | 4 | 4 |
| SUBB direct, #data16 | 带借位直接地址减去16-bit立即数 | 5 | 4 |
| 逻辑操作 | | | |
| AND Rd, Rs | 寄存器之间直接逻辑与 | 2 | 3 |
| AND Rd, [Rs] | 寄存器-寄存器间接地址逻辑与 | 2 | 4 |
| AND [Rd], Rs | 寄存器间接地址-寄存器逻辑与 | 2 | 4 |
| AND Rd, [Rs+offset8] | 寄存器-8-bit偏移寄存器间接逻辑与 | 3 | 6 |
| AND [Rd+offset8], Rs | 8-bit偏移寄存器间接-寄存器逻辑与 | 3 | 6 |
| AND Rd, [Rs+offset16] | 寄存器-8-bit偏移寄存器间接逻辑与 | 4 | 6 |
| AND [Rd+offset16], Rs | 16-bit偏移寄存器间接-寄存器逻辑与 | 4 | 6 |
| AND Rd, [Rs+] | 寄存器-自动增量寄存器间接逻辑与 | 2 | 5 |
| AND [Rd+], Rs | 自动增量寄存器间接-寄存器逻辑与 | 2 | 5 |
| AND direct, Rs | 直接地址-寄存器逻辑与 | 3 | 4 |
| AND Rd, direct | 寄存器-直接地址逻辑与 | 3 | 4 |
| AND Rd, #data8 | 寄存器-8-bit立即数逻辑与 | 3 | 3 |
| AND Rd, #data16 | 寄存器-16-bit立即数逻辑与 | 3 | 5 |
| AND [Rd], #data8 | 寄存器间接-8-bit立即数逻辑与 | 3 | 4 |
| AND [Rd], #data16 | 寄存器间接-16-bit立即数逻辑与 | 4 | 4 |
| AND [Rd+], #data8 | 自动增量寄存器间接-8-bit立即数逻辑与 | 3 | 5 |
| AND [Rd+], #data16 | 自动增量寄存器间接-16-bit立即数逻辑与 | 4 | 5 |
| AND [Rd+offset8], #data8 | 8-bit偏移寄存器间接-8-bit立即数逻辑与 | 4 | 6 |
| AND [Rd+offset8], #data16 | 8-bit偏移寄存器间接-16-bit立即数逻辑与 | 5 | 6 |
| AND [Rd+offset16], #data8 | 16-bit偏移寄存器间接-8-bit立即数逻辑与 | 5 | 6 |
| AND [Rd+offset16], #data16 | 16-bit偏移寄存器间接-16-bit立即数逻辑与 | 6 | 6 |
| AND direct, #data8 | 直接地址-8-bit立即数 | 4 | 4 |
| AND direct, #data16 | 直接地址-16-bit立即数 | 5 | 4 |
| CPL Rd | 寄存器求反 | 2 | 3 |
| LSR Rd, Rs | 目标寄存器逻辑右移,移动次数由源寄存器决定 | 2 | 见注1 |
| LSR Rd, #data4 | 目标寄存器逻辑右移,移动次数由8-bit立即数决定 | 2 | 见注1 |
| NORM Rd, Rs | 目标寄存器逻辑右移到最高位为1,移动次数由源寄存器决定 | 2 | 见注1 |
| OR Rd, Rs | 寄存器之间逻辑或 | 2 | 3 |
| OR Rd, [Rs] | 寄存器-寄存器间接逻辑或 | 2 | 4 |
| OR [Rd], Rs | 寄存器间接-寄存器逻辑或 | 2 | 4 |
| OR Rd, [Rs+offset8] | 寄存器-8-bit偏移寄存器间接逻辑或 | 3 | 6 |
| OR [Rd+offset8], Rs | 8-bit偏移寄存器间接-寄存器逻辑或 | 3 | 6 |
| OR Rd, [Rs+offset16] | 寄存器-16-bit偏移寄存器间接逻辑或 | 4 | 6 |

| | | | |
|---|---|---|---|
| OR [Rd+offset16], Rs | 16-bit偏移寄存器间接-寄存器逻辑或 | 4 | 6 |
| OR Rd, [Rs+] | 寄存器-自动增量寄存器间接逻辑或 | 2 | 5 |
| OR [Rd+], Rs | 自动增量寄存器间接-寄存器逻辑或 | 2 | 5 |
| OR direct, Rs | 直接地址-寄存器逻辑或 | 3 | 4 |
| OR Rd, direct | 寄存器-直接地址逻辑或 | 3 | 4 |
| OR Rd, #data8 | 寄存器-8-bit立即数逻辑或 | 3 | 3 |
| OR Rd, #data16 | 寄存器-16-bit立即数逻辑或 | 4 | 3 |
| OR [Rd], #data8 | 寄存器间接-8-bit立即数逻辑或 | 3 | 4 |
| OR [Rd], #data16 | 寄存器间接-16-bit立即数逻辑或 | 4 | 4 |
| OR [Rd+], #data8 | 自动增量寄存器间接-8-bit立即数逻辑或 | 3 | 5 |
| OR [Rd+], #data16 | 自动增量寄存器间接-16-bit立即数逻辑或 | 4 | 5 |
| OR [Rd+offset8], #data8 | 8-bit偏移寄存器间接-8-bit立即数逻辑或 | 4 | 6 |
| OR [Rd+offset8], #data16 | 8-bit偏移寄存器间接-16-bit立即数逻辑或 | 5 | 6 |
| OR [Rd+offset16], #data8 | 16-bit偏移寄存器间接-8-bit立即数逻辑或 | 5 | 6 |
| OR [Rd+offset16], #data16 | 16-bit偏移寄存器间接-16-bit立即数逻辑或 | 6 | 6 |
| OR direct, #data8 | 直接地址-8-bit立即数逻辑或 | 4 | 4 |
| OR direct, #data16 | 直接地址-16-bit立即数逻辑或 | 5 | 4 |
| RL Rd, #data4 | 寄存器左旋转,旋转次数由4-bit立即数决定. | 2 | 见注1 |
| RLC Rd, #data4 | 寄存器通过进位左旋转,旋转次数由4-bit立即数决定 | 2 | 见注1 |
| RR Rd, #data4 | 寄存器右旋转,旋转次数由4-bit立即数决定. | 2 | 见注1 |
| RRC Rd, #data4 | 寄存器通过进位右旋转,旋转次数由4-bit立即数决定 | 2 | 见注1 |
| XOR Rd, Rs | 寄存器之间逻辑异或 | 2 | 3 |
| XOR Rd, [Rs] | -寄存器间接逻辑异或 | 2 | 4 |
| XOR [Rd], Rs | 寄存器间接-寄存器逻辑异或 | 2 | 4 |
| XOR Rd, [Rs+offset8] | 寄存器-8-bit偏移寄存器间接逻辑异或 | 3 | 6 |
| XOR [Rd+offset8], Rs | 8-bit偏移寄存器间接-寄存器逻辑异或 | 3 | 6 |
| XOR Rd, [Rs+offset16] | 寄存器-16-bit偏移寄存器间接逻辑异或 | 4 | 6 |
| XOR [Rd+offset16], Rs | 16-bit偏移寄存器间接-寄存器逻辑异或 | 4 | 6 |
| XOR Rd, [Rs+] | 寄存器-自动增量寄存器间接逻辑异或 | 2 | 5 |
| XOR [Rd+], Rs | 自动增量寄存器间接-寄存器逻辑异或 | 2 | 5 |
| XOR direct, Rs | 直接地址-寄存器逻辑异或 | 3 | 4 |
| XOR Rd, direct | 寄存器-直接地址逻辑异或 | 3 | 4 |
| XOR Rd, #data8 | 寄存器-8-bit立即数逻辑异或 | 3 | 3 |
| XOR Rd, #data16 | 寄存器-16-bit立即数逻辑异或 | 4 | 3 |
| XOR [Rd], #data8 | 寄存器间接-8-bit立即数逻辑异或 | 3 | 4 |
| XOR [Rd], #data16 | 寄存器间接-16-bit立即数逻辑异或 | 4 | 4 |
| XOR [Rd+], #data8 | 自动增加寄存器间接-8-bit立即数逻辑异或 | 3 | 5 |
| XOR [Rd+], #data16 | 自动增加寄存器间接-16-bit立即数逻辑异或 | 4 | 5 |
| XOR [Rd+offset8], #data8 | 8-bit偏移寄存器间接-8-bit立即数逻辑异或 | 4 | 6 |
| XOR [Rd+offset8], #data16 | 8-bit偏移寄存器间接-16-bit立即数逻辑异或 | 5 | 6 |
| XOR [Rd+offset16], #data8 | 16-bit偏移寄存器间接-8-bit立即数逻辑异或 | 5 | 6 |
| XOR [Rd+offset16], #data16 | 16-bit偏移寄存器间接-16-bit立即数逻辑异或 | 6 | 6 |
| XOR direct, #data8 | 直接地址-8-bit立即数逻辑异或 | 4 | 4 |
| XOR direct, #data16 | 直接地址-16-bit立即数逻辑异或 | 5 | 4 |
| | | | |
| 数据传输 | | | |
| MOV Rd, Rs | 数据从寄存器-寄存器 | 2 | 3 |
| MOV Rd, [Rs] | 数据从寄存器间接-寄存器 | 2 | 3 |
| MOV [Rd], Rs | 数据从寄存器-寄存器间接 | 2 | 3 |
| MOV Rd, [Rs+offset8] | 数据从8-bit偏移寄存器间接-寄存器 | 3 | 5 |
| MOV [Rd+offset8], Rs | 数据从寄存器-8-bit偏移寄存器间接 | 3 | 5 |
| MOV Rd, [Rs+offset16] | 数据从16-bit偏移寄存器间接-寄存器 | 4 | 5 |
| MOV [Rd+offset16], Rs | 数据从寄存器-16-bit偏移寄存器间接 | 4 | 5 |

| | | | |
|---|---|---|---|
| MOV Rd, [Rs+] | 数据从自动增量寄存器间接-寄存器 | 2 | 4 |
| MOV [Rd+], Rs | 数据从寄存器-自动增量寄存器间接 | 2 | 4 |
| MOV direct, Rs | 数据从寄存器-直接地址 | 3 | 4 |
| MOV Rd, direct | 数据从直接地址-寄存器 | 3 | 4 |
| MOV [Rd+], [Rs+] | 数据从自动增量寄存器间接-自动增量寄存器间接 | 2 | 6 |
| MOV direct, [Rs] | 数据从寄存器间接-直接地址 | 3 | 4 |
| MOV [Rd], direct | 数据从直接地址-寄存器间接 | 3 | 4 |
| MOV Rd, #data8 | 数据从8-bit立即数-寄存器 | 3 | 3 |
| MOV Rd, #data16 | 数据从16-bit立即数-寄存器 | 4 | 3 |
| MOV [Rd], #data8 | 数据从8-bit立即数-寄存器间接 | 3 | 3 |
| MOV [Rd], #data16 | 数据从16-bit立即数-寄存器间接 | 4 | 3 |
| MOV [Rd+], #data8 | 数据从8-bit立即数-自动增量寄存器间接 | 3 | 4 |
| MOV [Rd+], #data16 | 数据从16-bit立即数-自动增量寄存器间接 | 4 | 4 |
| MOV [Rd+offset8], #data8 | 数据从8-bit立即数-8-bit偏移寄存器间接 | 4 | 5 |
| MOV [Rd+offset8], #data16 | 数据从16-bit立即数-8-bit偏移寄存器间接 | 5 | 5 |
| MOV [Rd+offset16], #data8 | 数据从8-bit立即数-16-bit偏移寄存器间接 | 5 | 5 |
| MOV [Rd+offset16], #data16 | 数据从16-bit立即数-16-bit偏移寄存器间接 | 6 | 5 |
| MOV direct, #data8 | 数据从8-bit立即数-直接地址 | 4 | 3 |
| MOV direct, #data16 | 数据从16-bit立即数-直接地址 | 5 | 3 |
| MOV direct, direct | 数据从直接地址-直接地址 | 4 | 4 |
| MOV Rd, USP | 数据从堆栈指针-寄存器(仅在系统模式中) | 2 | 3 |
| MOV USP, Rs | 数据从寄存器-堆栈指针(仅在系统模式中) | 2 | 3 |
| MOVC Rd, [Rs+] | 数据从自动增量代码区WS:RS地址处-寄存器 | 2 | 4 |
| MOVC A, [A+DPTR] | 数据从代码区[A+DPTR]地址处-A | 2 | 6 |
| MOVC A, [A+PC] | 数据从代码区[A+PC]地址处-A | 2 | 6 |
| MOVS Rd, #data4 | 数据从4-bit符号延伸立即数-寄存器 | 2 | 3 |
| MOVS [Rd], #data4 | 数据从4-bit符号延伸立即数-寄存器间接 | 2 | 3 |
| MOVS [Rd+], #data4 | 数据从4-bit符号延伸立即数-自动增量寄存器间接 | 2 | 4 |
| MOVS [Rd+offset8], #data4 | 数据从4-bit符号延伸立即数-8-bit偏移寄存器间接 | 3 | 5 |
| MOVS [Rd+offset16], #data4 | 数据从4-bit符号延伸立即数-16-bit偏移寄存器间接 | 4 | 5 |
| MOVS direct, #data4 | 数据从4-bit符号延伸立即数-立即地址 | 3 | 3 |
| MOVX Rd, [Rs] | 数据从外部寄存器间接-寄存器 | 2 | 6 |
| MOVX [Rd], Rs | 数据从寄存器-外部寄存器间接 | 2 | 6 |
| PUSH direct | 直接地址进如当前堆栈 | 3 | 5 |
| PUSHU direct | 直接地址进如用户堆栈 | 3 | 5 |
| PUSH Rlist | 多个寄存器进入当前堆栈 | 2 | 见注2 |
| PUSHU Rlist | 多个寄存器进入用户堆栈 | 2 | 见注2 |
| POP direct | 数据(字节或字)从当前堆栈弹出到直接地址 | 3 | 5 |
| POPU direct | 数据(字节或字)从用户堆栈弹出到直接地址 | 3 | 5 |
| POP Rlist | 从当前堆栈中弹出多个寄存器(字节或字) | 2 | 见注3 |
| POPU Rlist | 从用户堆栈中弹出多个寄存器(字节或字) | 2 | 见注3 |
| XCH Rd, Rs | 两个寄存器交换 | 2 | 5 |
| XCH Rd, [Rs] | 寄存器-寄存器间接交换 | 2 | 6 |
| XCH Rd, direct | 寄存器-直接地址交换 | 3 | 6 |
| | | | |
| 程序分支 | | | |
| BCC rel8 | 如果进位清除则分支 | 2 | 6t/3nt |
| BCS rel8 | 如果进位设置则分支 | 2 | 6t/3nt |
| BEQ rel8 | 如果零标志位设置则分支 | 2 | 6t/3nt |
| BNE rel8 | 如果零标志位没设置则分支 | 2 | 6t/3nt |
| BG rel8 | 如果大于(无符号)则分支 | 2 | 6t/3nt |
| BGE rel8 | 如果大于或等于(有符号)则分支 | 2 | 6t/3nt |
| BGT rel8 | 如果大于(有符号)则分支 | 2 | 6t/3nt |
| BL rel8 | 如果小于或等于(无符号)则分支 | 2 | 6t/3nt |
| BLE rel8 | 如果小于或等于(有符号)则分支 | 2 | 6t/3nt |

| | | | |
|---|---|---|---|
| BLT rel8 | 如果小于(有符号)则分支 | 2 | 6t/3n |
| BMI rel8 | 如果负标志设置则分支 | 2 | 6t/3n |
| BPL rel8 | 如果负标志清除则分支 | 2 | 6t/3n |
| BNV rel8 | 如果溢出标志清除则分支 | 2 | 6t/3n |
| BOV rel8 | 如果溢出标志设置则分支 | 2 | 6t/3n |
| BR rel8 | 无条件短分支 | 2 | 6 |
| CALL [Rs] | 使用寄存器间接调用子程序 | 2 | 8/5(PZ) |
| CALL rel16 | 相对调用(+/- 64K) | 3 | 7/4(PZ) |
| CJNE Rd,direct,rel8 | 比较寄存器和直接地址,不相等则跳转 | 4 | 10t/7nt |
| CJNE Rd,#data8,rel8 | 比较寄存器和8-bit立即数,不相等则跳转 | 4 | 9t/6nt |
| CJNE Rd,#data16,rel8 | 比较寄存器和16-bit立即数,不相等则跳转 | 5 | 9t/6nt |
| CJNE [Rd],#data8,rel8 | 比较寄存器和8-bit立即数,不相等则跳转 | 4 | 10t/7nt |
| CJNE [Rd],#data16,rel8 | 比较寄存器间接和16-bit立即数,不相等则跳转 | 5 | 10t/7nt |
| DJNZ Rd,rel8 | 寄存器减1,不为0则跳转 | 3 | 8t/5nt |
| DJNZ direct,rel8 | 直接地址减1,不为0则跳转 | 4 | 9t/5nt |
| FCALL addr24 | 长调用(可以在24-bit地址空间的任意位置) | 4 | 12/8(PZ) |
| FJMP addr24 | 长跳转(可以在24-bit地址空间的任意位置) | 4 | 6 |
| JB bit,rel8 | 如果位设置则跳转 | 4 | 10t/6nt |
| JBC bit,rel8 | 如果位设置则跳转,并清除该位 | 4 | 11t/7nt |
| JMP rel16 | 无条件长跳转 | 3 | 6 |
| JMP [Rs] | 跳转到由寄存器间接决定的地址(64K). | 2 | 7 |
| JMP [A+DPTR] | 跳转到由A+DPTR间接决定的地址(64K). | 2 | 5 |
| JMP [[Rs+]] | 跳转到由寄存器决定的双间接地址. | 2 | 8 |
| JNB bit,rel8 | 如果位没设置则跳转 | 4 | 10t/6nt |
| JNZ rel8 | 如果累加器不为0则跳转 | 2 | 6t/3nt |
| JZ rel8 | 如果累加器为0则跳转 | 2 | 6t/3nt |
| NOP | 空操作 | 1 | 3 |
| RET | 从子程序返回 | 2 | 8/6(PZ) |
| RETI | 从中断返回 | 2 | 10/8(PZ) |
| | | | |
| 位操作 | | | |
| ANL C, bit | 进位-位逻辑与 | 3 | 4 |
| ANL C, /bit | 进位-位的反逻辑与 | 3 | 4 |
| CLR bit | 清除位 | 3 | 4 |
| MOV C, bit | 位->进位移动 | 3 | 4 |
| MOV bit, C | 进位->位移动 | 3 | 4 |
| ORL C, bit | 进位-位逻辑或 | 3 | 4 |
| ORL C, /bit | 进位-位的反逻辑或 | 3 | 4 |
| SETB bit | 设置指定位 | 3 | 4 |
| | | | |
| 异常和陷阱 | | | |
| BKPT | 引起断点异常中断 | 1 | 23/19(PZ) |
| RESET | 引起硬件复位,同外部复位相同 | 2 | 18 |
| TRAP #data4 | 引起16个硬件陷阱之一 | 2 | 23/19(PZ) |

注1:对于8和16 bit移位,每附加两bit是4+1;对于32 bit移位,每附加两bit是6+1;

注2:每个寄存器的压入占用3个时钟周期.

注3:第一个寄存器为4个时钟周期,其余的附加寄存器为2个时钟周期.

**ADD** 整数加

语法**:** ADD dest, source

操作**:** dest <- src + dest

描述**:**

对源操作数和目标操作数进行二进制加法.结果保存在目标操作数中.操作后源操作数不变.

注:如果用于对PSWL写,将优先于标志位更新.

尺寸**:** 字节-字节**,**字-字
标志位更新**:** C, AC, V, N, Z

**ADD Rd, Rs**
字节数: 2
时钟数: 3
操作: (Rd) <-- (Rd) + (Rs)
译码:

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | SZ | 0 | 0 | 1 | d | d | d | d | s | s | s | s |

**ADD Rd, [Rs]**
字节数: 2
时钟数: 4
操作: (Rd) <-- (Rd) + ((WS:Rs))
译码:

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | SZ | 0 | 1 | 0 | d | d | d | d | 0 | s | s | s |

**ADD [Rd], Rs**
字节数: 2
时钟数: 4
操作: (WS:Rd) <-- (WS:Rd) + (Rs)
译码:

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | SZ | 0 | 1 | 0 | s | s | s | s | 1 | d | d | d |

**ADD Rd, [Rs+offset8]**
字节数: 3
时钟数: 6
操作: (Rd) <-- (Rd) + ((WS:Rs)+offset8)
译码:

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | SZ | 1 | 0 | 0 | d | d | d | d | 0 | s | s | s |

**ADD [Rd+offset8], Rs**
字节数: 3
时钟数: 6
操作: ((WS:Rd)+offset8) <-- ((WS:Rd)+offset8) + (Rs)
译码:

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | SZ | 1 | 0 | 0 | s | s | s | s | 1 | d | d | d |

字节3:8-bit偏移地址

**ADD Rd, [Rs+offset16]**
字节数: 4
时钟数: 6
操作: (Rd) <-- (Rd) + ((WS:Rs)+offset16)
译码:

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | SZ | 1 | 0 | 1 | d | d | d | d | 0 | s | s | s |

**ADD [Rd+offset16], Rs**
字节数: 4
时钟数: 6
操作: ((WS:Rd)+offset16) <-- ((WS:Rd)+offset16) + (Rs)

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | SZ | 1 | 0 | 1 | s | s | s | s | 1 | d | d | d |

字节3:16-bit偏移地址的高8-bit
字节4:16-bit偏移地址的低8-bit

**ADD Rd, [Rs+]**
字节数: 2
时钟数: 5
操作: (Rd) <-- (Rd) + ((WS:Rs))
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | SZ | 0 | 1 | 1 | d | d | d | d | 0 | s | s | s |

**ADD [Rd+], Rs**
字节数: 2
时钟数: 5
操作: ((WS:Rd)) <-- ((WS:Rd)) + (Rs)
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | SZ | 0 | 1 | 1 | s | s | s | s | 1 | d | d | d |

**ADD direct, Rs**
字节数: 3
时钟数: 4
操作: (direct) <-- (direct) + (Rs)
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | SZ | 1 | 1 | 0 | s | s | s | s | 1 | 3-bit直接地址 | | |

字节3:直接地址低8-bit

**ADD Rd, direct**
字节数: 3
时钟数: 4
操作: (Rd) <-- (Rd) + (direct)
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | SZ | 1 | 1 | 0 | d | d | d | d | 0 | 3-bit直接地址 | | |

字节3:直接地址低8-bit

**ADD Rd, #data8**
字节数: 3
时钟数: 3
操作: (Rd) <-- (Rd) + #data8
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | d | d | d | d | 0 | 0 | 0 | 0 |

字节4:8-bit立即数

**ADD Rd, #data16**
字节数: 4
时钟数: 3
操作: (Rd) <-- (Rd) + #data16
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | d | d | d | d | 0 | 0 | 0 | 0 |

字节3:16-bit立即数的高8-bit

字节4:16-bit立即数的低8-bit

**ADD [Rd], #data8**

字节数: 3

时钟数: 4

操作:  ((WS:Rd)) <-- ((WS:Rd)) + #data8

译码:

| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | d | d | d | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节4:8-bit立即数

**ADD [Rd], #data16**

字节数: 4

时钟数: 4

操作:  ((WS:Rd)) <-- ((WS:Rd)) + #data16

译码:

| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | d | d | d | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit立即数的高8-bit

字节4:16-bit立即数的低8-bit

**ADD [Rd+], #data8**

字节数: 3

时钟数: 5

操作:  ((WS:Rd)) <-- ((WS:Rd)) + #data8

译码:

| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | d | d | d | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节4:8-bit立即数

**ADD [Rd+], #data16**

字节数: 4

时钟数: 5

操作:  ((WS:Rd)) <-- ((WS:Rd)) + #data16

译码:

| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | d | d | d | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit立即数的高8-bit

字节4:16-bit立即数的低8-bit

**ADD [Rd+offset8], #data8**

字节数: 4

时钟数: 6

操作:  ((WS:Rd)+offset8) <-- ((WS:Rd)+offset8) + #data8

译码:

| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | d | d | d | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit偏移地址

字节4:8-bit立即数

**ADD [Rd+offset8], #data16**

字节数: 5

时钟数: 6

操作:  ((WS:Rd)+offset8) <-- ((WS:Rd)+offset8) + #data16

译码:

| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | d | d | d | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit偏移地址
字节4:16-bit立即数的高8-bit
字节5:16-bit立即数的低8-bit

**ADD [Rd+offset16], #data8**
字节数: 5
时钟数: 6
操作:  ((WS:Rd)+offset16) <-- ((WS:Rd)+offset16) + #data8
译码:

| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | d | d | d | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit偏移地址的高8-bit
字节4:16-bit偏移地址的低8-bit
字节4:8-bit立即数

**ADD [Rd+offset16], #data16**
字节数: 6
时钟数: 6
操作:  ((WS:Rd)+offset16) <-- ((WS:Rd)+offset16) + #data16
译码:

| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | d | d | d | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit偏移地址的高8-bit
字节4:16-bit偏移地址的低8-bit
字节5:16-bit立即数的高8-bit
字节6:16-bit立即数的低8-bit

**ADD direct, #data8**
字节数: 4
时钟数: 4
操作:  (direct) <-- (direct) + #data8
译码:

| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 3-bit直接地址 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:直接地址低8-bit
字节4:8-bit立即数

**ADD direct, #data16**
字节数: 5
时钟数: 4
操作:  (direct) <-- (direct) + #data16
译码:

| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 3-bit直接地址 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:直接地址低8-bit
字节3:16-bit立即数的高8-bit
字节4:16-bit立即数的低8-bit

**ADDC** 带进位整数加
**语法:** ADDC dest**,** source
**操作:** dest <- dest + src + C
**描述:**
对源操作数和目标操作数带前面产生的进位进行二进制加法.结果保存在目标操作数中.操作后源操作数

不变.如果C=1,结果大于源和目标的和;如果C=0,则相等.这种形式的加法用来支持多精度运算.使用时,先复位C,然后对多精度数值从低部分到高部分进行相加.
尺寸**:** 字节-字节**,**字-字
标志位更新**:** C, AC, V, N, Z

**ADDC Rd, Rs**
字节数: 2
时钟数: 3
操作: (Rd) <-- (Rd) + (Rs) + (C)
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | SZ | 0 | 0 | 1 | d | d | d | d | s | s | s | s |

**ADDC Rd, [Rs]**
字节数: 2
时钟数: 4
操作: (Rd) <-- (Rd) + ((WS:Rs)) + (C)
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | SZ | 0 | 1 | 0 | d | d | d | d | 0 | s | s | s |

**ADDC [Rd], Rs**
字节数: 2
时钟数: 4
操作: ((WS:Rd)) <-- ((WS:Rd)) + (Rs) + (C)
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | SZ | 0 | 1 | 0 | s | s | s | s | 1 | d | d | d |

**ADDC Rd, [Rs+offset8]**
字节数: 3
时钟数: 6
操作: (Rd) <-- (Rd) + ((WS:Rs)+offset8) + (C)
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | SZ | 1 | 0 | 0 | d | d | d | d | 0 | s | s | s |

字节3:8-bit偏移地址

**ADDC [Rd+offset8], Rs**
字节数: 3
时钟数: 6
操作: ((WS:Rd)+offset8) <-- ((WS:Rd)+offset8) + (Rs) + (C)
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | SZ | 1 | 0 | 0 | s | s | s | s | 1 | d | d | d |

字节3:8-bit偏移地址

**ADDC Rd, [Rs+offset16]**
字节数: 4
时钟数: 6
操作: (Rd) <-- (Rd) + ((WS:Rs)+offset16) + (C)
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | SZ | 1 | 0 | 1 | d | d | d | d | 0 | s | s | s |

字节3:16-bit偏移地址的高8-bit
字节4:16-bit偏移地址的低8-bit

**ADDC [Rd+offset16], Rs**

字节数: 4

时钟数: 6

操作:  ((WS:Rd)+offset16) <-- ((WS:Rd)+offset16) + (Rs) + (C)

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | SZ | 1 | 1 | 1 | s | s | s | s | 1 | d | d | d |

**ADDC Rd, [Rs+]**

字节数: 2

时钟数: 5

操作:  (Rd) <-- (Rd) + ((WS:Rs)) + (C)

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | SZ | 0 | 1 | 1 | d | d | d | d | 0 | s | s | s |

**ADDC [Rd+], Rs**

字节数: 2

时钟数: 5

操作:  ((WS:Rd)) <-- ((WS:Rd)) + (Rs) + (C)

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | SZ | 0 | 1 | 1 | s | s | s | s | 1 | d | d | d |

**ADDC direct, Rs**

字节数: 3

时钟数: 4

操作:  (direct) <-- (direct) + (Rs) + (C)

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | SZ | 1 | 1 | 0 | s | s | s | s | 1 | 3-bit直接地址 | | |

字节3:直接地址低8-bit

**ADDC Rd, direct**

字节数: 3

时钟数: 3

操作:  (Rd) <-- (Rd) + (direct) + (C)

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | SZ | 1 | 1 | 0 | d | d | d | d | 0 | 3-bit直接地址 | | |

字节3:直接地址低8-bit

**ADDC Rd, #data8**

字节数: 3

时钟数: 3

操作:  (Rd) <-- (Rd) + #data8 + (C)

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | d | d | d | 0 | 0 | 0 | 1 |

字节4:8-bit立即数

**ADDC Rd, #data16**

字节数: 4

时钟数: 3

操作:  (Rd) <-- (Rd) + #data16 + (C)

译码:

| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | d | d | d | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit立即数的高8-bit
字节4:16-bit立即数的低8-bit

**ADDC [Rd], #data8**
字节数: 3
时钟数: 4
操作: ((WS:Rd)) <-- ((WS:Rd)) + #data8 + (C)
译码:

| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | d | d | d | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节4:8-bit立即数

**ADDC [Rd], #data16**
字节数: 4
时钟数: 4
操作: ((WS:Rd)) <-- ((WS:Rd)) + #data16 + (C)
译码:

| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | d | d | d | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit立即数的高8-bit
字节4:16-bit立即数的低8-bit

**ADDC [Rd+], #data8**
字节数: 3
时钟数: 5
操作: ((WS:Rd)) <-- ((WS:Rd)) + #data8 + (C)
(Rd) <-- (Rd) + 1
译码:

| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | d | d | d | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节4:8-bit立即数

**ADDC [Rd+], #data16**
字节数: 4
时钟数: 6
操作: ((WS:Rd)) <-- ((WS:Rd)) + #data16 + (C)
(Rd) <-- (Rd) + 2
译码:

| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | d | d | d | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit立即数的高8-bit
字节4:16-bit立即数的低8-bit

**ADDC [Rd+offset8], #data8**
字节数: 4
时钟数: 6
操作: ((WS:Rd)+offset8) <-- ((WS:Rd)+offset8) + #data8 + (C)
译码:

| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | d | d | d | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit偏移地址
字节4:8-bit立即数

**ADDC [Rd+offset8], #data16**

字节数: 5

时钟数: 6

操作: ((WS:Rd)+offset8) <-- ((WS:Rd)+offset8) + #data16 + (C)

译码:

| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

| 0 | d | d | d | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|

字节3:8-bit偏移地址

字节4:16-bit立即数的高8-bit

字节5:16-bit立即数的低8-bit

**ADDC [Rd+offset16], #data8**

字节数: 5

时钟数: 6

操作: ((WS:Rd)+offset16) <-- ((WS:Rd)+offset16) + #data8 + (C)

译码:

| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|

| 0 | d | d | d | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|

字节3:16-bit偏移地址的高8-bit

字节4:16-bit偏移地址的低8-bit

字节5:8-bit立即数

**ADDC [Rd+offset16], #data16**

字节数: 6

时钟数: 6

操作: ((WS:Rd)+offset16) <-- ((WS:Rd)+offset16) + #data16 + (C)

译码:

| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|

| 0 | d | d | d | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|

字节3:16-bit偏移地址的高8-bit

字节4:16-bit偏移地址的低8-bit

字节5:16-bit立即数的高8-bit

字节6:16-bit立即数的低8-bit

**ADDC direct, #data8**

字节数: 4

时钟数: 4

操作: (direct) <-- (direct) + #data8 + (C)

译码:

| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| 0 | 3-bit直接地址 | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|

字节3:直接地址低8-bit

字节4:8-bit立即数

**ADDC direct, #data16**

字节数: 5

时钟数: 4

操作: (direct) <-- (direct) + #data16 + (C)

译码:

| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| 0 | 3-bit直接地址 | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|

字节3:直接地址低8-bit

字节4:16-bit立即数的高8-bit

字节5:16-bit立即数的低8-bit

**ADDS** 短加

语法**:** ADDS dest, #value

操作**:** dest <- dest + #data4

描述**:**

4-bit有符号立即数加到目标中.立即数的符号延伸到合适的尺寸,然后加到目标操作数指定的变量中(可以是字节,或字).立即数的范围是+7到-8.此指令的原意是用来加减指针和计数器.

尺寸**:** 字节-字节**,**字-字

标志位更新**:** N, Z

注:ADDS不会更新C和AC标志,因为这个指令是用来代替80C51中的INC和DEC指令的(这两个指令不更新标志)

**ADDS Rd, #data4**

字节数: 2

时钟数: 4

操作: (Rd) <-- (Rd) + #data4

译码:

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | SZ | 0 | 0 | 1 | d | d | d | d | 4-bit立即数 | |

**ADDS [Rd], #data4**

字节数: 2

时钟数: 4

操作: ((WS:Rd)) <-- ((WS:Rd)) + #data4

译码:

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | SZ | 0 | 1 | 0 | 0 | d | d | d | 4-bit立即数 | |

**ADDS [Rd+], #data4**

字节数: 2

时钟数: 5

操作: ((WS:Rd)) <-- ((WS:Rd)) + #data4

译码:

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | SZ | 0 | 1 | 1 | 0 | d | d | d | 4-bit立即数 | |

**ADDS [Rd+offset8], #data4**

字节数: 3

时钟数: 6

操作: ((WS:Rd)+offset8) <-- ((WS:Rd)+offset8) + #data4

译码:

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | SZ | 1 | 0 | 0 | 0 | d | d | d | 4-bit立即数 | |

字节3:8-bit偏移地址

**ADDS [Rd+offset16], #data4**

字节数: 4

时钟数: 6

操作: ((WS:Rd)+offset16) <-- ((WS:Rd)+offset16) + #data4

译码:

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | SZ | 1 | 0 | 1 | 0 | d | d | d | 4-bit立即数 | |

字节3:16-bit偏移地址的高8-bit

字节4:16-bit偏移地址的低8-bit

**ADDS direct, #data4**

字节数: 3

时钟数: 4

操作: (direct) <-- (direct) + #data4

译码:

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | SZ | 1 | 1 | 0 | 0 | 3-bit直接地址 | | 4-bit立即数 |

**AND 逻辑与**

**语法:** AND dest, src

**操作:** dest <- dest AND src

**描述:**

源和目标按位与.源操作数指定的字节或字和目标操作数指定的变量进行逻辑与,放入目标变量中.操作后源数据不变.

**尺寸:** 字节-字节,字-字

**标志位更新:** N, Z

**AND Rd, Rs**

字节数: 2

时钟数: 3

操作: (Rd) <-- (Rd) • (Rs)

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | SZ | 0 | 0 | 1 | d | d | d | d | s | s | s | s |

**AND Rd, [Rs]**

字节数: 2

时钟数: 4

操作: (Rd) <-- (Rd) AND ((WS:Rs))

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | SZ | 0 | 1 | 0 | d | d | d | d | 0 | s | s | s |

**AND [Rd], Rs**

字节数: 2

时钟数: 4

操作: ((WS:Rd)) <-- ((WS:Rd)) AND (Rs)

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | SZ | 0 | 1 | 0 | s | s | s | s | 1 | d | d | d |

**AND Rd, [Rs+offset8]**

字节数: 3

时钟数: 6

操作: (Rd) <-- (Rd) • ((WS:Rs)+offset8)

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | SZ | 1 | 0 | 0 | d | d | d | d | 0 | s | s | s |

字节3:8-bit偏移地址

**AND [Rd+offset8], Rs**

字节数: 3

时钟数: 6

操作: ((WS:Rd)+offset8) <-- ((WS:Rd)+offset8) • (Rs)

译码:

| 0 | 1 | 0 | 1 | SZ | 1 | 0 | 0 | s | s | s | s | 1 | d | d | d |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit偏移地址

**AND Rd, [Rs+offset16]**

字节数: 4

时钟数: 6

操作: (Rd) <-- (Rd) AND ((WS:Rs)+offset16)

译码:

| 0 | 1 | 0 | 1 | SZ | 1 | 0 | 1 | d | d | d | d | 0 | s | s | s |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit偏移地址的高8-bit

字节4:16-bit偏移地址的低8-bit

**AND [Rd+offset16], Rs**

字节数: 4

时钟数: 6

操作: ((WS:Rd)+offset16) <-- ((WS:Rd)+offset16) • (Rs)

译码:

| 0 | 1 | 0 | 1 | SZ | 1 | 0 | 1 | s | s | s | s | 1 | d | d | d |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit偏移地址的高8-bit

字节4:16-bit偏移地址的低8-bit

**AND Rd, [Rs+]**

字节数: 2

时钟数: 5

操作: (Rd) <-- (Rd) AND ((WS:Rs))

译码:

| 0 | 1 | 0 | 1 | SZ | 0 | 1 | 1 | d | d | d | d | 0 | s | s | s |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

**AND [Rd+], Rs**

字节数: 2

时钟数: 5

操作: ((WS:Rd)) <-- ((WS:Rd)) AND (Rs)

译码:

| 0 | 1 | 0 | 1 | SZ | 0 | 1 | 1 | s | s | s | s | 1 | d | d | d |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

**AND direct, Rs**

字节数: 3

时钟数: 4

操作: (direct) <-- (direct) • (Rs)

译码:

| 0 | 1 | 0 | 1 | SZ | 1 | 1 | 0 | s | s | s | s | 1 | 3-bit直接地址 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:直接地址低8-bit

**AND Rd, direct**

字节数: 3

时钟数: 3

操作: (Rd) <-- (Rd) AND (direct)

译码:

| 0 | 1 | 0 | 1 | SZ | 1 | 1 | 0 | d | d | d | d | 0 | 3-bit直接地址 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:直接地址低8-bit

**AND Rd, #data8**

字节数: 3

时钟数: 3

操作: (Rd) <-- (Rd) • #data8

译码:

| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | d | d | d | d | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit立即数

**AND Rd, #data16**

字节数: 4

时钟数: 3

操作: (Rd) <-- (Rd) AND #data16

译码:

| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | d | d | d | d | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit立即数的高8-bit

字节4:16-bit立即数的低8-bit

**AND [Rd], #data8**

字节数: 3

时钟数: 4

操作: ((WS:Rd)) <-- ((WS:Rd)) AND #data8

译码:

| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | d | d | d | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit立即数

**AND [Rd], #data16**

字节数: 4

时钟数: 4

操作: ((WS:Rd)) <-- ((WS:Rd)) AND #data16

译码:

| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | d | d | d | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit立即数的高8-bit

字节4:16-bit立即数的低8-bit

**AND [Rd+], #data8**

字节数: 4

时钟数: 5

操作: ((WS:Rd)) <-- ((WS:Rd)) AND #data8

(Rd) <-- (Rd) + 1

译码:

| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | d | d | d | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit立即数

**AND [Rd+], #data16**

字节数: 4

时钟数: 5

操作: ((WS:Rd)) <-- ((WS:Rd)) AND #data16

(Rd) <-- (Rd) + 2

译码:

| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | d | d | d | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit立即数的高8-bit
字节4:16-bit立即数的低8-bit

**AND [Rd+offset8], #data8**
字节数: 4
时钟数: 6
操作: ((WS:Rd)+offset8) <-- ((WS:Rd)+offset8) AND #data8
译码:

| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | d | d | d | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit偏移地址
字节4:8-bit立即数

**AND [Rd+offset8], #data16**
字节数: 5
时钟数: 6
操作: ((WS:Rd)+offset8) <-- ((WS:Rd)+offset8) AND #data16
译码:

| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | d | d | d | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit偏移地址
字节4:16-bit立即数的高8-bit
字节5:16-bit立即数的低8-bit

**AND [Rd+offset16], #data8**
字节数: 5
时钟数: 6
操作: ((WS:Rd)+offset16) <-- ((WS:Rd)+offset16) AND #data8
译码:

| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | d | d | d | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit偏移地址的高8-bit
字节4:16-bit偏移地址的低8-bit
字节5:8-bit立即数

**AND [Rd+offset16], #data16**
字节数: 6
时钟数: 6
操作: ((WS:Rd)+offset16) <-- ((WS:Rd)+offset16) AND #data16
译码:

| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | d | d | d | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit偏移地址的高8-bit
字节4:16-bit偏移地址的低8-bit
字节5:16-bit立即数的高8-bit
字节6:16-bit立即数的低8-bit

**AND direct, #data8**
字节数: 4
时钟数: 4
操作: (direct) <-- (direct) AND #data8
译码:

| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 3-bit直接地址 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3: 直接地址的低8-bit
字节4: 8-bit立即数

**AND direct, #data16**
字节数: 5
时钟数: 4
操作: (direct) <-- (direct) AND #data16
译码:

| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| 0 | 3-bit直接地址 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|

字节3: 直接地址的低8-bit
字节4:16-bit立即数的高8-bit
字节5:16-bit立即数的低8-bit

**ANL** 指定位同进位逻辑与
语法**:** ANL C, bit
操作**:** C <- C (AND) Bit
描述**:** 读入指定位,同进位逻辑与,放入C中
尺寸**:** 位
标志位更新**:**位
注: 这里的进位标志是被写入,而不是由于ALU操作被刷新.
字节数**:** 3
时钟数**:** 4
译码**:**

| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

| 0 | 1 | 0 | 0 | 0 | 0 | 2bit位地址 |
|---|---|---|---|---|---|---|

字节3: 位地址的低8-bit

**ANL** 指定位取反**,**再同进位逻辑与
语法**:** ANL C, /bit
操作**:** Carry <- C (AND) bit取反
描述**:** 读入指定位,取反,同进位逻辑与,放入C中
尺寸**:** 位
标志位更新**:** 无
注: 这里的进位标志是被写入,而不是由于ALU操作被刷新.
字节数**:** 3
时钟数**:** 4
译码:

| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

| 0 | 1 | 0 | 1 | 0 | 0 | 2bit位地址 |
|---|---|---|---|---|---|---|

字节3: 位地址的低8-bit

**ASL** 数学左移
语法**:** ASL dest, coun
操作**:** (C) <- (dest.msb)
(dest.bit n+1) <- (dest.bit n)
count = count-1
Do While (count not equal to 0)
End While
if sign change during shift,
(V) <- 1
描述**:** 如果count操作数>0,则由目标操作数指定的变量数学做移,移位的次数由count操作数决定.移位中

最低位添入0,最高位移到C中.如果count操作数=0,则不执行移位.为了保留原操作数的符号,结果的最高位延续为符号位.
记数操作数可以是:
-立即数(#data4/5)
-寄存器(只使用5 bit,最多31个移位)
记数操作数可以是立即数或寄存器.count是一个0-31的正数,而目标操作数是有符号的整数.操作后count操作数不变.数据尺寸可以是8,16或32 bit.在32-bit移位时,目标操作数是双字寄存器的低部分.
注意:
- 如果移位记数超过了数据尺寸, count将被限制在5bits, 另外对于立即移位的count,如果count=0,移位停止.
- 双字寄存器是两个相邻的寄存器(R1:R0, R3:R2, R5:R4, 或R7:R6).
尺寸: 字节,字,双字
**标志位更新:** C, N, Z

**ASL Rd, Rs**
字节数**:**2
时钟数**:** 对于8/16 bit 每2个bit移位-> 4 + 1
对于32 bit 每2个bit移位-> 6 + 1
操作:


译码**:**
字节和字尺寸

| 1 | 1 | 0 | 1 | SZ1 | SZ0 | 0 | 1 |
|---|---|---|---|---|---|---|---|

| d | d | d | d | 4-bit立即数 |
|---|---|---|---|---|

双字尺寸

| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|

| d | d | d | 5-bit立即数 |
|---|---|---|---|

注: SZ1/SZ0 = 00:字节操作;
SZ1/SZ0 = 10: 字操作;
SZ1/SZ0 = 11: 双字操作.

**ASR** 数学右移
语法**:** ASR dest, count
操作**:** (C) <- (dest.0)
(dest.bit n) <- (dest.bit n+1)
count = count-1
Do While (count not equal to 0)
End While
dest.msb <- Sign bit
描述**:**如果count操作数>0,则由目标操作数指定的变量数学右移,移位的次数由count操作数决定.移位中最高位添入0,最低位移到C中.如果count操作数=0,则不执行移位.为了保留原操作数的符号,结果的最高位延续为符号位.
记数操作数可以是:
-立即数(#data4/5)
-寄存器(只使用5 bit,最多31个移位)
记数操作数可以是立即数或寄存器.count是一个0-31的正数,而目标操作数是有符号的整数.操作后count操作数不变.数据尺寸可以是8,16或32 bit.在32-bit移位时,目标操作数是双字寄存器的低部分.
注意:

- 如果移位记数超过了数据尺寸，count将被限制在5bits，另外对于立即移位的count,如果count=0,移位停止.

- 双字寄存器是两个相邻的寄存器(R1:R0, R3:R2, R5:R4，或R7:R6).

尺寸: 字节,字,双字

标志位更新: C, N, Z

**ASR Rd, Rs**

操作:


字节数: 2

时钟数: 对于8/16 bit 每2个bit移位-> 4 + 1

对于32 bit 每2个bit移位-> 6 + 1

译码:

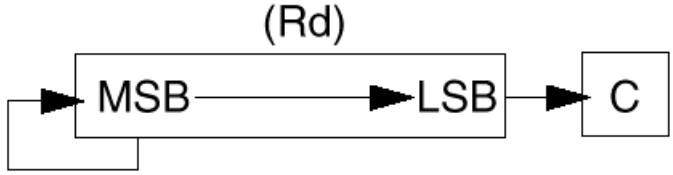
字节和字尺寸

| 1 | 1 | 0 | 0 | SZ1 | SZ0 | 1 | 0 | d | d | d | d | s | s | s | s |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

**ASR Rd, #data4**

Rd,#data5

操作:




字节数: 2

时钟数: 对于8/16 bit 每2个bit移位-> 4 + 1

对于32 bit 每2个bit移位-> 6 + 1

译码:

字节和字尺寸

| 1 | 1 | 0 | 1 | SZ1 | SZ0 | 1 | 0 | d | d | d | d | 4-bit立即数 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|

双字尺寸

| 1 | 1 | 0 | 1 | SZ1 | SZ0 | 1 | 0 | d | d | d | 5-bit立即数 |
|---|---|---|---|---|---|---|---|---|---|---|---|

注: SZ1/SZ0 = 00:字节操作;

SZ1/SZ0 = 10: 字操作;

SZ1/SZ0 = 11: 双字操作.

**BCC 如果进位C=0则跳转**

语法: BCC rel8

操作: (PC) <-- (PC) + 2

if (C) = 0 then

(PC) <-- (PC + rel8*2)

(PC.0) <-- 0

如果上一次数学/逻辑指令(或其它指令更新了C标志)没产生进位(C=1),则分支发生.分支范围在+254字节到-256字节，目标地址必须是字相邻.

注:详细的分支范围见节6.3
尺寸:位
标志位更新:无
字节数:2
时钟数: 6t/3nt
译码:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | rel8 |

**BCS** 如果进位**C=1**则跳转

语法: BCS rel8
操作: (PC) <-- (PC) + 2
if (C) = 1 then
(PC) <-- (PC + rel8*2)
(PC.0) <-- 0
描述:
如果上一次数学/逻辑指令(或其它指令更新了C标志)产生了进位(C=1),则分支发生.分支范围在+254字节到-256字节, 目标地址必须是字相邻.
注:详细的分支范围见节6.3
尺寸:位
标志位更新:无
字节数:2
时钟数: 6t/3nt
译码:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | rel8 |

**BEQ** 如果为**0**则跳转

语法: BEQ rel8
操作: (PC) <-- (PC) + 2
if (Z) = 1 then
(PC) <-- (PC + rel8*2)
(PC.0) <-- 0
描述:
如果上一次数学/逻辑指令(或其它指令更新了Z标志)产生了等于于0的结果(Z=1),则分支发生.分支范围在+254字节到-256字节, 目标地址必须是字相邻.
注:详细的分支范围见节6.3
尺寸:位
标志位更新:无
字节数:2
时钟数: 6t/3nt
译码:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | rel8 |

**BG** 如果大于则跳转**(**无符号**)**

语法: BG rel8
操作: (PC) <-- (PC) + 2
if (Z) OR (C) = 0 then
(PC) <-- (PC + rel8*2)
(PC.0) <-- 0
描述:
如果上一次比较指令的结果是:目标数值大于源数值(有符号操作),则分支发生.分支范围在+254字节到-

256字节，目标地址必须是字相邻.
注:详细的分支范围见节6.3
尺寸:位
标志位更新:无
字节数:2
时钟数: 6t/3nt
译码:

| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | rel8 |
|---|---|---|---|---|---|---|---|---|

**BGE** 如果大于或等于则跳转(有符号)

语法: BGE rel8
操作: (PC) <-- (PC) + 2
if (N) XOR (V) = 0 then
(PC) <-- (PC + rel8*2)
(PC.0) <-- 0
描述:
如果上一次比较指令的结果是:目标数值大于或等于源数值(有符号操作),则分支发生.分支范围在+254字节到-256字节，目标地址必须是字相邻.
注:详细的分支范围见节6.3
尺寸:位
标志位更新:无
字节数:2
时钟数: 6t/3nt
译码:

| 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | rel8 |
|---|---|---|---|---|---|---|---|---|

**BGT** 如果大于则跳转(有符号)

语法: BGT rel8
操作: (PC) <-- (PC) + 2
if ((Z) OR (N)) XOR (V) = 0 then
(PC) <-- (PC + rel8*2)
(PC.0) <-- 0
描述:
如果上一次比较指令的结果是:目标数值大于源数值(有符号操作),则分支发生.分支范围在+254字节到-256字节，目标地址必须是字相邻.
注:详细的分支范围见节6.3
尺寸:位
标志位更新:无
字节数:2
时钟数: 6t/3nt
译码:

| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | rel8 |
|---|---|---|---|---|---|---|---|---|

**BKPT** 断点

语法: BKPT
操作: (PC) <-- (PC) + 1
(SSP) <-- (SSP) - 6
((SSP)) <-- (PC)
((SSP)) <-- (PSW)
(PSW) <-- code memory (bkpt vector)

(PC.15-0) <-- code memory (bkpt vector)
(PC.23-16) <-- 0; (PC.0) <-- 0

描述:
引起一个断点中断.断点中断类似与立即中断,使用矢量调用运行在系统模式下的一段指定代码.这个指令用于仿真系统,提供一个简单的方法来实现硬件断点.
为了在所有条件下使断点正常工作,必须有一个指令,它的长度不大于所有的指令,这就是单字节指令NOP.这是因为断点可以NOP的位置,NOP后面可能是另外一些指令,如分支(到)或其它不需要通过断点的指令.如果断点的长度大于NOP,它将会在执行时破坏指令的顺序.
断点的操作码特别分配为FFh.NOP的操作码特别分配为00h这样没编程的EPROM的位置将是断点,或空操作,不会是有其它副作用的指令.
尺寸**:** 无
标志位更新**:** 无(5)
字节数:1
时钟数: 23/19 (PZ)
译码:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

5. 所有的PSW由矢量表中调入.它可能是Debugger的保存,但不一定要这样做.

**BL** 如果小于或等于则跳转**(**无符号**)**

语法**:** BL rel8
操作**:** (PC) <-- (PC) + 2
if (Z) OR (C) = 1 then
(PC) <-- (PC + rel8*2)
(PC.0) <-- 0

描述**:**
如果上一次比较指令的结果是:目标数值小于或等于源数值(无符号操作),则分支发生.分支范围在+254字节到-256字节, 目标地址必须是字相邻.
注:详细的分支范围见节6.3
尺寸**:**位
标志位更新**:**无
字节数**:**2
时钟数**:** 6t/3nt
译码**:**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | rel8 |

**BLE** 如果小于或等于则跳转**(**有符号**)**

语法**:**BLE rel8
操作**:** (PC) <-- (PC) + 2
if ((Z) OR (N)) XOR (V) = 1 then
(PC) <-- (PC + rel8*2)
(PC.0) <-- 0

描述**:**
如果上一次比较指令的结果是:目标数值小于或等于源数值(有符号操作),则分支发生.分支范围在+254字节到-256字节, 目标地址必须是字相邻.
注:详细的分支范围见节6.3
尺寸**:**位
标志位更新**:**无
字节数**:**2
时钟数**:** 6t/3nt

译码:

| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | rel8 |
|---|---|---|---|---|---|---|---|---|

**BLT** 如果小于则跳转(有符号)

语法: BLT rel8

操作: (PC) <-- (PC) + 2
if (N) XOR (V) = 1 then
(PC) <-- (PC + rel8*2)
(PC.0) <-- 0

描述:

如果上一次比较指令的结果是:目标数值小于源数值(有符号操作),则分支发生.分支范围在+254字节到-256字节, 目标地址必须是字相邻.

注:详细的分支范围见节6.3

尺寸:位

标志位更新:无

字节数:2

时钟数: 6t/3nt

译码:

| 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | rel8 |
|---|---|---|---|---|---|---|---|---|

**BMI** 如果为负则跳转

语法: BMI rel8

操作: (PC) <-- (PC) + 2
if (N) = 1 then
(PC) <-- (PC + rel8*2)
(PC.0) <-- 0

描述:

如果上一次数学/逻辑指令(或其它指令更新了N标志)产生了小于0的结果(N=1),则分支发生.分支范围在+254字节到-256字节, 目标地址必须是字相邻.

注:详细的分支范围见节6.3

尺寸:位

标志位更新:无

字节数:2

时钟数: 6t/3nt

译码:

| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | rel8 |
|---|---|---|---|---|---|---|---|---|

**BNE** 如果不相等则跳转

语法: BNE rel8

操作: (PC) <-- (PC) + 2
if (Z) = 0 then
(PC) <-- (PC + rel8*2)
(PC.0) <-- 0

描述:

如果上一次数学/逻辑指令(或其它指令更新了Z标志)产生了非0结果(Z=0),则分支发生.分支范围在+254字节到-256字节, 目标地址必须是字相邻.

注:详细的分支范围见节6.3

尺寸:位

标志位更新:无

字节数:2

时钟数: 6t/3nt
译码:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | rel8 |

**BNV** 如果没溢出则跳转

语法: BNV rel8
操作: (PC) <-- (PC) + 2
if (V) = 0 then
(PC) <-- (PC + rel8*2)
(PC.0) <-- 0
描述:
如果上一次数学/逻辑指令(或其它指令更新了V标志)没产生溢出(V=0),则分支发生.分支范围在+254字节到-256字节, 目标地址必须是字相邻.
注:详细的分支范围见节6.3
尺寸:位
标志位更新:无
字节数:2
时钟数: 6t/3nt
译码:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | rel8 |

**BOV** 如果溢出则跳转

语法: BOV rel8
操作: (PC) <-- (PC) + 2
if (V) = 1 then
(PC) <-- (PC + rel8*2)
(PC.0) <-- 0
描述:
如果上一次数学/逻辑指令(或其它指令更新了V标志)产生了溢出(V=1),则分支发生.分支范围在+254字节到-256字节, 目标地址必须是字相邻.
注:详细的分支范围见节6.3
尺寸:位
标志位更新:无
字节数:2
时钟数: 6t/3nt
译码:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | rel8 |

**BPL** 如果为正则跳转

语法: BPL rel8
操作: (PC) <-- (PC) + 2
if (N) = 0 then
(PC) <-- (PC + rel8*2)
(PC.0) <-- 0
描述:
如果上一次数学/逻辑指令(或其它指令更新了N标志)产生了大于0的结果(N=0)则分支发生.分支范围在+254字节到-256字节, 目标地址必须是字相邻.
注:详细的分支范围见节6.3
尺寸:位
标志位更新:无

字节数:2
时钟数: 6t/3nt
译码:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | rel8 |

**BR** 无条件分支

语法: BR rel8
操作: (PC) <-- (PC) + 2
(PC) <-- (PC + rel8*2)
(PC.0) <-- 0
描述:
无条件分支,范围在+254字节到-256字节, 目标地址必须是字相邻.
注:详细的分支范围见节6.3
尺寸:无
标志位更新:无
字节数:2
时钟数:6
译码:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | rel8 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |

**CALL** 相对子程序调用

语法: CALL rel16
操作: (PC) <-- (PC) + 3
(SP) <-- (SP) - 4
((SP)) <-- (PC.23-0)
(PC) <-- (PC + rel16*2)
(PC.0) <-- 0
描述:
无条件分支,范围在+65,534字节到-65,536字节, 目标地址必须是字相邻.24-bit返回地址保存在堆栈中.
注:如果XA在0页模式中,只有16-bit的地址压入堆栈
注:详细的分支范围见节6.3
尺寸:无
标志位更新:无
字节数: 3
时钟数:7/4(PZ)
译码:
字节2: rel16的高8 bit
byte 3: rel16的低8 bit

**CALL** 间接子程序调用

语法: CALL [Rs]
操作: (PC) <-- (PC) + 2
(SP) <-- (SP) - 4
((SP)) <-- (PC.23-0)
(PC.15-1) <-- (Rs.15-1)
(PC.0) <-- 0
描述:
无条件分支到由操作数寄存器包含的地址,可以在CALL指令后面64K页内的任意范围.返回地址(CALL指

令后面的地址)保存在堆栈中.目标地址必须是字相邻,因为CALL使PC.0=0.
注: (1)由于PC总是指向CALL指令后面的地址,所以如果发生在一个不同的页,被调用的程序必须在那个页中.
(2)如果XA在0页模式中,压入堆栈的地址只有16-bit.
尺寸**:** 无
标志位更新**:** 无
字节数**:**2
时钟数**:**8/5(PZ)
译码**:**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | s | s | s |

**CJNE** 比较,不相等则跳转

语法**:** CJNE dest, src, rel8
操作**:** (PC) <-- (PC) + # of instruction bytes
(dest) - (direct) (result not stored)
if (Z) = 0 then
(PC) <-- (PC + rel8*2); (PC.0) <-- 0
描述**:**
由源操作数指定的字节或字同目标操作数指定的变量相比较,并更新状态标志.如果相等则跳转到指定地址.操作后源和目标操作数不变.分支的范围是+254字节到-256字节,而且目标地址在代码区域中必须是字相邻.
注:详细跳转范围参考6.3节
尺寸**:** 字节-字节,字-字
标志更新**:** C, N, Z
(注意: 这种特殊类型的比较为了同80C51一致,不更新V和AV标志)

**CJNE Rd, direct, rel8**
字节数:4
时钟数:10t/7nt
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | SZ | 0 | 1 | 0 | d | d | d | d | 0 | 3-bit直接地址 | | |

字节3:直接地址的低8-bit
字节4:rel8

**CJNE Rd, #data8, rel8**
字节数:4
时钟数:9t/6nt
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | d | d | d | d | 0 | 0 | 0 | 0 |

字节3:rel
字节4:8-bit立即数

**CJNE Rd, #data16, rel8**
字节数:5
时钟数:9t/6nt
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | d | d | d | d | 0 | 0 | 0 | 0 |

字节3:rel
字节4:16-bit立即数的高8-bit
字节5:16-bit立即数的低8-bit

**CJNE [Rd], #data8, rel8**

字节数:4

时钟数:10t/7nt

译码:

| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | d | d | d | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:rel

字节4:8-bit立即数

**CJNE [Rd], #data16, rel8**

字节数:5

时钟数:10t/7nt

译码:

| 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | d | d | d | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3: rel8

字节3:16-bit立即数的高8-bit

字节4:16-bit立即数的低8-bit

**CLR** 清除指定位

**语法:** CLR bit

**操作:** (bit) <-- 0

**描述:** 清除指定位

**储存:** 位

**标志位更新:** 无

**字节数:** 3

**时钟数:** 4

译码:

| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2-bit |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:位地址的低8-bit

**CMP** 整数比较

**语法:** CMP dest, src

**操作**: dest - src

**描述:**

源操作数同目标操作数比较, dest – src.标志位根据减法规则.操作后缘操作数和目标操作数不变.

**尺寸:** 字节-字节,字-字

**标志位更新:** C, AC, V, N, Z

CMP Rd, Rs

字节数: 2

时钟数: 3

操作: (Rd) - (Rs)

译码:

| 0 | 1 | 0 | 0 | SZ | 0 | 0 | 1 | d | d | d | d | s | s | s | s |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

**CMP Rd, [Rs]**

字节数: 2

时钟数: 4

操作: (Rd) - ((WS:Rs))

译码:

| 0 | 1 | 0 | 0 | SZ | 0 | 1 | 0 | d | d | d | d | 0 | s | s | s |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

**CMP [Rd], Rs**

字节数: 2

时钟数: 4

操作: ((WS:Rd)) - (Rs)

译码:

| 0 | 1 | 0 | 0 | SZ | 0 | 1 | 0 | s | s | s | s | 1 | d | d | d |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

**CMP Rd, [Rs+offset8]**

字节数: 3

时钟数: 6

操作: (Rd) - ((WS:Rs)+offset8)

译码:

| 0 | 1 | 0 | 0 | SZ | 1 | 0 | 0 | d | d | d | d | 0 | s | s | s |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit偏移地址

**CMP [Rd+offset8], Rs**

字节数: 3

时钟数: 6

操作: ((WS:Rd)+offset8) - (Rs)

译码:

| 0 | 1 | 0 | 0 | SZ | 1 | 0 | 1 | s | s | s | s | 0 | d | d | d |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit偏移地址

**CMP Rd, [Rs+offset16]**

字节数: 4

时钟数: 6

操作: (Rd) - ((WS:Rs)+offset16)

译码:

| 0 | 1 | 0 | 0 | SZ | 1 | 0 | 1 | s | s | s | s | 0 | d | d | d |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit偏移地址的高8-bit

字节4:16-bit偏移地址的低8-bit

**CMP [Rd+offset16], Rs**

字节数: 4

时钟数: 6

操作: ((WS:Rd)+offset16) - (Rs)

译码:

| 0 | 1 | 0 | 0 | SZ | 1 | 0 | 1 | s | s | s | s | 1 | d | d | d |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

**CMP Rd, [Rs+]**

字节数: 2

时钟数: 5

操作: (Rd) - ((WS:Rs))

(Rs) <-- (Rs) + 1 (字节操作) 或 2 (字操作)

译码:

| 0 | 1 | 0 | 0 | SZ | 0 | 1 | 1 | s | s | s | s | 1 | d | d | d |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

**CMP [Rd+], Rs**

字节数: 2

时钟数: 5
操作:　((WS:Rd)) - (Rs)
　　(Rd) <-- (Rd) + 1 (字节操作)  或  2 (字操作)
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | SZ | 0 | 1 | 1 | s | s | s | s | 1 | d | d | d |

**CMP direct, Rs**
字节数: 3
时钟数: 4
操作: (direct) - (Rs)
译码:

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | SZ | 1 | 1 | 0 | s | s | s | s | 1 | 3-bit直接地址 |

字节3:直接地址低8-bit

**CMP Rd, direct**
字节数: 3
时钟数: 4
操作: (Rd) - (direct)
译码:

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | SZ | 1 | 1 | 0 | d | d | d | d | 0 | 3-bit直接地址 |

字节3:直接地址低8-bit

**CMP Rd, #data8**
字节数: 3
时钟数: 3
操作: (Rd) - #data8
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | d | d | d | d | 0 | 1 | 0 | 0 |

字节3:8-bit立即数

**CMP Rd, #data16**
字节数: 4
时钟数: 4
操作: (Rd) - #data16
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | d | d | d | d | 0 | 1 | 0 | 0 |

字节3:16-bit立即数的高8-bit
字节4:16-bit立即数的低8-bit

CMP [Rd], #data8
字节数: 3
时钟数: 4
操作:　((WS:Rd)) - #data8
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | d | d | d | 0 | 1 | 0 | 0 |

字节4:8-bit立即数

**CMP [Rd], #data16**
字节数: 4
时钟数: 4

操作:　((WS:Rd)) - #data16
译码:

| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | d | d | d | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit立即数的高8-bit
字节4:16-bit立即数的低8-bit

**CMP [Rd+], #data8**
字节数: 3
时钟数: 5
操作:　((WS:Rd)) - #data8
　　(Rd) <-- (Rd) + 1
译码:

| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | d | d | d | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit立即数

**CMP [Rd+], #data16**
字节数: 4
时钟数: 5
操作:　((WS:Rd)) - #data16
　　(Rd) <-- (Rd) + 2
译码:

| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | d | d | d | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit立即数的高8-bit
字节4:16-bit立即数的低8-bit

CMP [Rd+offset8], #data8
字节数: 4
时钟数: 6
操作: ((WS:Rd)+offset8) - #data8
译码:

| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | d | d | d | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit偏移地址
字节4:8-bit立即数

**CMP [Rd+offset8], #data16**
字节数: 5
时钟数: 6
操作: ((WS:Rd)+offset8) - #data16
译码:

| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | d | d | d | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit偏移地址
字节4:16-bit立即数的高8-bit
字节5:16-bit立即数的低8-bit

**CMP [Rd+offset16], #data8**
字节数: 6
时钟数: 6
操作: ((WS:Rd)+offset16) - #data8
译码:

| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | d | d | d | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit偏移地址的高8-bit
字节4:16-bit偏移地址的低8-bit
字节5:8-bit立即数

**CMP [Rd+offset16], #data16**
字节数: 6
时钟数: 6
操作: ((WS:Rd)+offset16) - #data16
译码:

| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | d | d | d | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit偏移地址的高8-bit
字节4:16-bit偏移地址的低8-bit
字节5:16-bit立即数的高8-bit
字节6:16-bit立即数的低8-bit

**CMP direct, #data8**
字节数: 4
时钟数: 4
操作: (direct) - #data8
译码:

| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 3-bit直接地址 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3: 直接地址的低8-bit
字节4:8-bit立即数

**CMP direct, #data16**
字节数: 5
时钟数: 4
操作: (direct) - #data16
译码:

| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 3-bit直接地址 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3: 直接地址的低8-bit
字节4:16-bit立即数的高8-bit
字节5:16-bit立即数的低8-bit

**CPL** 取反

语法**:** CPL Rd
操作**:** Rd <-- (Rd取反)
描述**:**
对寄存器Rd取反,结果回存到Rd中.目标可以是字节,或字.
尺寸**:** 字节**,**字
标志更新: N, Z
字节数**:** 2
时钟数**:** 3
译码:

| 1 | 0 | 0 | 1 | SZ | 0 | 0 | 0 | d | d | d | d | 1 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

**DA** 十进制调整

语法**:** DA Rd
操作**:** if (Rd.3-0) > 9 or (AC) = 1
then (Rd.3-0) <-- (Rd.3-0) + 6

if (Rd.7-4) > 9 or (C) = 1
then (Rd.7-4) <-- (Rd.7-4) + 6

**描述:**

在ADD或ADDC对BCD数进行操作后,使用此指令对目标寄存器进行调整(二进制代码的十进制).这个操作只能作用于字节寄存器.如果目标数值的低4 bit大于9,或AC=1,则要加上6.如果加上6引起高4-bit进位,则C=1.如果目标数值的高4 bit大于9,或由于低位加引起C=1,则要加上60h..如果加上60h引起高4 bit进位,则C=1.如果C已经为1,DA指令不会对C清除.

**尺寸:** 字节

**标志位更新:** C, N, Z

进位可能被置为1,但不会被清除.见上面的进位标志的更新.

字节数:2

时钟数:4

译码:

| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

| d | d | d | d | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

注: 请参考下面的表格.

表格6.6 展示了DA指令可能引起的动作,及相关的输入条件.

| 低半字节 (bit 3-0) | AC | 进位到 高半字节 | 高半字节 (bit 7-4) | 起始C标志 | 要加的数值 | 结果C标志 |
|---|---|---|---|---|---|---|
| 0 - 9 | 0 | 0 | 0 - 9 | 0 | 00 | 0 |
| A - F | 0 | 1 | 0 - 8 | 0 | 06 | 0 |
| 0 - 3 * | 1 | 0 | 0 - 9 | 0 | 06 | 0 |
| 0 - 9 | 0 | 0 | A - F | 0 | 60 | 1 |
| A - F | 0 | 1 | 9 - F | 0 | 66 | 1 |
| 0 - 3 * | 1 | 0 | A - F | 0 | 66 | 1 |
| 0 – 9 | 0 | 0 | 0 - 2 ** | 1 | 60 | 1 |
| A - F | 0 | 1 | 0 - 2 ** | 1 | 66 | 1 |
| 0 - 3 * | 1 | 0 | 0 - 3 *** | 1 | 66 | 1 |

*由于两个BCD数字相加引起AC=1的最大数字是3,ADDC指令,9+9+1(进位标志)=13h

**由于两个BCD数字相加引起高半字节没有从低半字节接收进位(AC=0)的最大数字为2.例如,98h+97h=12Fh

***由于两个BCD数字相加引起高半字节从低半字节接收进位(AC=1)的最大数字为3.例如,99h+99h=132h

**DIV.w 16x8** 有符号除

**DIV.d 32x16** 有符号除

**DIVU.b 8x8** 无符号除

**DIVU.w 16x8** 无符号除

**DIVU.d 32x16** 无符号除

**描述:**

由目标操作数指定的变量除以由源操作数指定的字或字节.

对于DIVU.b,目标操作数可以是任意字寄存器的低有效字节.对于DIV.w和DIVU.w,目标操作数必须是字寄存器,对于DIV.d和DIVU.d,目标操作数必须是字寄存器而且是一个双字寄存器的低字(见下面注释).结果的商保留在目标寄存器中(对于DIVU.b, DIVU.w, DIV.w,和DIVU.w是8-bit,对于DIV.d and DIVU.d是16-bit)的低半部分,余数(同商宽度一致)在高半部分(除了DIVU.b,商保存在字寄存器的低半部分,余数保存在相同字寄存器的高半部分.)

注: 双寄存器是两个相邻的寄存器(R1:R0, R3:R2, R5:R4,和R7:R6).

**尺寸:** 字节-字节,字-字,双字-双字

**标志位更新:** C, V, N, Z

进位C总是被清除.标志V在以下情况=1,其它情况下=0:

- DIVU.b: 如果除以0,则V=1.除以0也会引起一个硬件异常中断.
- DIV.w, DIVU.w: 如果除的结果大于8-bit(结果放不进目标里)V=1.
- DIV.d, DIVU.d: 如果除的结果大于16-bit(结果放不进目标里)V=1.

Z,N的置位取决于结果的商,而不取决于余数.

范例:

a) DIVU.b R4L, R4H - R4L/R4H,商放入R4L,余数放入R4H.

b) DIV.w R0, R2L – R0/R2L,结果商放入R0L,余数放入R0H.

c) DIV.d R4,R2 – R5:R4/R2,结果商在R4,余数在R5.

注:除了DIVU.b,所有的除指令目标寄存器的宽度同指令说明的( ".b" ".w",".d")一致,源寄存器是它的一半.

**DIV.w Rd, Rs**

(有符号16 bits / 8 bits --> 8 bit商, 8 bit余数)

字节数: 2

时钟数: 14

操作: (RdL) <-- (Rd) / (Rs) 8-bit整数部分(有符号除)

(RdH) <-- (Rd) / (Rs) 8-bit余数部分

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | d | d | d | d | s | s | s | s |

**DIV.w Rd, #data8**

(有符号16 bits / 8 bits --> 8 bit商, 8 bit余数)

字节数: 3

时钟数: 14

操作: (RdL) <-- (Rd) / #data8的8-bit整数部分 (有符号除)

(RdH) <-- (Rd) / #data8的8-bit余数部分

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | d | d | d | d | 1 | 0 | 1 | 1 |

字节4:8-bit立即数

**DIV.d Rd, Rs**

(有符号32 bits / 16 bits --> 16 bit商, 16 bit余数)

字节数: 2

时钟数: 24

操作: (Rd) <-- (Rd) / (Rs) 的16-bit整数部分 (有符号除)

(Rd+1)<--(Rd) / (Rs) 16-bit余数部分

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | d | d | d | 0 | s | s | s | s |

**DIV.d Rd, #data16**

(有符号32 bits / 16 bits --> 16 bit商, 16 bit余数)

字节数: 4

时钟数: 24

操作: (Rd) <-- (Rd) / #data16的16-bit整数部分 (有符号除)

(Rd+1)<-- (Rd) / #data16的16-bit余数部分

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | d | d | d | 0 | 1 | 0 | 0 | 1 |

字节3:16-bit立即数的高8-bit

字节4:16-bit立即数的低8-bit

**DIVU.b Rd, Rs**

(无符号8 bits / 8 bits --> 8 bit商, 8 bit余数)

字节数: 2

时钟数: 12

操作:　(RdL) <-- (RdL) / (Rs)  的8-bit整数部分  (无符号除)
　　(RdH) <-- (RdL) / (Rs)  的8-bit余数部分
译码:

| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | d | d | d | d | s | s | s | s |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

**DIVU.b Rd, #data8**
(无符号8 bits / 8 bits --> 8 bit商, 8 bit余数)
字节数: 3
时钟数: 12
操作:　(RdL) <-- (RdL) / #data8的8-bit整数部分  (无符号除)
　　(RdH) <-- (RdL) / #data8的8-bit余数部分
译码:

| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | d | d | d | d | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节4:8-bit立即数

**DIVU.w Rd, Rs**
(无符号16 bits / 8 bits --> 8 bit商, 8 bit余数)
字节数: 2
时钟数: 12
操作:　(RdL) <-- (Rd) / (Rs)  的8-bit整数部分(无符号除)
　　(RdH) <-- (Rd) / (Rs)  的8-bit余数部分
译码:

| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | d | d | d | d | s | s | s | s |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

**DIVU.w Rd, #data8**
(无符号16 bits / 8 bits --> 8 bit商, 8 bit余数)
字节数: 3
时钟数: 12
操作:　(RdL) <-- (Rd) / #data8的8-bit整数部分  (无符号除)
　　(RdH) <-- (Rd) / #data8的8-bit余数部分
译码:

| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | d | d | d | d | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:  字节4:8-bit立即数

**DIVU.d Rd, Rs**
(无符号32 bits / 16 bits --> 16 bit商, 16 bit余数)
字节数: 4
时钟数: 22
操作:　(Rd) <--(Rd) / (Rs)  的16-bit整数部分(无符号除)
　　(Rd+1)<-- (Rd) / (Rs)  的16-bit余数部分
译码:

| 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | d | d | d | 0 | s | s | s | s |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

**DIVU.d Rd, #data16**
(无符号32 bits / 16 bits --> 16 bit商, 16 bit余数)
字节数: 4
时钟数: 22
操作:　(Rd) <-- (Rd) / #data16的16-bit整数部分(无符号除)
　　(Rd+1)<-- (Rd) / #data16的16-bit余数部分
译码:

| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | | d | d | d | 0 | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit立即数的高8-bit
字节4:16-bit立即数的低8-bit

**DJNZ** 递减,如果不为**0**则跳转

语法**:** DJNZ dest, rel8
操作**:**

(PC) <-- (PC) + 3
(dest) <-- (dest) - 1
if (Z) = 0 then
(PC) <-- (PC + rel8*2); (PC.0) <-- 0

描述**:**
循环控制指令.参数是:一个条件码(Z),计数器(寄存器或内存),偏移数值.首先指令对计数器减1,测试条件判断结果是否为0(中断循环);如果为0,则执行下一条指令.如果不为0,则执行分支,地址为PC+rel8.PC的数值是DJNZ后一条指令的地址.
分支范围是+254字节到-256字节,目标地址必须是字相邻.目标操作数可以是字节或字.
注:参考6.3节的跳转范围
尺寸**:** 字节**,**字
标志位更新**:** N, Z

**DJNZ Rd, rel8**
字节数:3
时钟数: 8t/5nt
译码:

| 1 | 0 | 0 | 0 | SZ | 1 | 1 | 1 | | d | d | d | d | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3: rel8

**DJNZ direct, rel8**
字节数:4
时钟数: 9t/5nt
译码:

| 1 | 1 | 1 | 0 | SZ | 0 | 1 | 0 | | 0 | 0 | 0 | 0 | 1 | 3-bit直接地址 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:直接地址的低8-bit
字节4: rel8

**FCALL** 绝对长子程序调用

语法**:** FCALL addr24
操作**:** (PC) <-- (PC) + 4
(SP) <-- (SP) - 4
((SP)) <-- (PC)
(PC.23-0) <-- addr24
(PC.0) <-- 0

描述**:**
无条件分支到一个由第二个操作数决定的绝对地址,地址可以是XA的16M空间的任意位置.被调用程序的24-bit返回地址(CALL下一条指令的地址)保存在堆栈中.目标地址必须是字相邻,因为CALL或分支总是使PC.0=0.
尺寸**:** 无
标志位更新**:** 无
字节数:4
时钟数: 12/8(PZ)

译码:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 地址的中8-bit(bits 15-8) |

字节3: 地址的低8字节(bits 7-0)
字节4: 地址的高8字节(bits 23-16)

**FJMP** 绝对长跳转

语法: FJMP addr24
操作: (PC.23-0) <-- addr24
(PC.0) <-- 0
描述:无条件跳转到一个绝对地址,地址由第二个操作数决定,可以是XA的16M空间的任意位置.
注:目标地址必须是字相邻,因为JMP将使PC.0=0
注:如果XA使用0页模式,则只使用16-bit地址.
尺寸: 无
标志位更新: 无
字节数:4
时钟数:6
译码:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 地址的中8-bit(bits 15-8) |

字节3: 地址的低8字节(bits 7-0)
字节4: 地址的高8字节(bits 23-16)

**JB** 如果指定位=**1**则相对跳转

语法: JB bit, rel8
操作: (PC) <-- (PC) + 4
if (bit) = 1 then
(PC) <-- (PC + rel8*2);
(PC.0) <-- 0
描述:
如果指定位=1,则程序跳转到PC+rel8的位置.如果指定位=0,则执行下一条指令.跳转的范围是+254字节到-256字节,但目标地址必须是字相邻.
注:参考6.3节的跳转范围
尺寸: 位
标志位更新: 无
字节数: 4
时钟数: 10t/6nt
译码:

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | bit: 2 |

字节3: 位地址的低8位
字节4: rel8

**JBC** 如果指定位=**1**则跳转,并清除该位

语法: JBC bit, rel8
操作: (PC) <-- (PC) + 4
if (bit) = 1 then
(PC) <-- (PC + rel8*2);
(PC.0) <-- 0; (bit) <-- 0
描述:如果指定位=1,则跳转,地址由PC + rel8*2决定.指定位被清除,表明标志撤消.如果指定位=0,则执行下一条指令. 跳转范围是+254字节到-256字节,目标地址在代码中必须是字相邻的
注:参考6.3节的跳转范围

尺寸: 位
标志位更新: 无
字节数:4
时钟数: 11t/7nt
译码:

| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|

| 1 | 1 | 0 | 0 | 0 | 0 | bit: 2 |
|---|---|---|---|---|---|---|

字节3:位地址的低8 bit
字节4: rel8

**JMP** 相对跳转

语法: JMP rel16
操作: (PC) <-- (PC) + 3
(PC) <-- (PC + rel16*2)
(PC.0) <-- 0
描述:无条件跳转.跳转范围是+65,535字节到-65,536字节,目标地址在代码中必须是字相邻的.
注:参考6.3节的跳转范围
尺寸: 无
标志位更新 无
字节数:3
时钟数:6
译码:

| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|

| rel16的高8 bit |
|---|

字节3: rel16的低8 bit

**JMP** 通过寄存器的间接跳转

语法: JMP [Rs]
操作: (PC) <-- (PC) + 2
(PC.15-1) <-- (Rs.15-1) (注意PC.23-16不变化)
(PC.0) <-- 0
描述:无条件跳转,地址由操作数寄存器的内容决定,可以在**JMP**指令后的**64K**代码页中.参加目标地址计算的**PC**数值是**JMP**指令下一条指令的地址.
注:子程序地址必须是字相邻,因为JMP将是PC.0=0
尺寸: 无
标志位更新: 无
字节谁: 2
时钟数: 7
译码:

| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| 0 | 1 | 1 | 1 | 0 | s | s | s |
|---|---|---|---|---|---|---|---|

**JMP** 通过寄存器的间接跳转

语法: JMP [A+DPTR]
操作: (PC) <-- (PC) + 2
(PC15-1) <-- (A) + (DPTR)
(PC.0) <-- 0
描述:无条件跳转,地址由两个**80C51**兼容寄存器A+DPTR的和确定,可以达到64K范围的任意位置. 这个指令是为了兼容80C51.详情见第9章80C51兼容特性.
注:子程序地址必须是字相邻,因为JMP将是PC.0=0
标志位更新: 无
字节数:2
时钟数:5

注: A和DPTR是为80C51代码传输而予定义的寄存器.

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |

**JMP** 双间接跳转

语法: JMP [[Rs+]]

操作: (PC) <-- (PC) + 2
(PC.15-0) <-- 代码区域((WS:Rs))
(PC.0) <-- 0
(Rs) <-- (Rs) + 2

描述:无条件跳转,地址是由指定寄存器指向的代码区域的数值.指定寄存器自动增加.使用这个2字节指令可以索引顺序操作的程序地址表,减少代码尺寸.使用另一个JMP [[Rs+]]将进入到表中另一个操作.通过执行"Jump Double-indirect"指令而运行的操作必须在同一个64K地址页中(但表可以在不同的页).这个指令的使用可以压缩代码,减少空间需求的费用.寄存器(表索引)每次运行后自动增加.24-bit的地址由PC的低16-bit和PC的高8-bit或CS寄存器确定,高端地址的选择有段选择SFR决定.

注:子程序地址必须是字相邻,因为JMP将是PC.0=0

标志更新: 无

字节数:2

时钟数:8

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | s | s | s |

**JNB** 如果指定位不等于**1**则跳转

语法: JNB bit, rel8

操作: (PC) <-- (PC) + 4
if (bit) = 0 then
(PC.15-0) <-- (PC + rel8*2); (PC.0) <-- 0

描述:如果指定位=0,程序将跳转到PC+rel8的位置.如果指定位=1,则程序继续执行. 分支范围是+254字节到-256字节,目标地址在代码空间中必须是字相邻.

尺寸: 位

标志位更新: 无

字节数: 4

时钟数: 10t/6nt

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | bit: 2 | |

字节**3:**位地址的低**8-bit**

byte 4: rel8

**JNZ** 如果寄存器**A!=0**则跳转

语法: JNZ rel8

操作: (PC) <-- (PC) + 2
if (A) not equal to 0, then
(PC.15-0) <-- (PC + rel8*2);(PC.0) <-- 0

描述:如果80C51的累加器A!=0,则发生相对分支.分支范围是+254字节到-256字节,目标地址在代码空间中必须是字相邻.累加器的内容保持不变.这个指令是为了兼容80C51.详情见第9章80C51兼容特性.

注:参考6.3节的跳转范围

尺寸: 位

标志位更新: 无

字节数: 2

时钟数: 6t/3nt

译码:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | rel8 |

**JZ** 如果寄存器**A=0**则跳转

语法: JZ rel8

操作: (PC) <-- (PC) + 2
If (A) = 0 then
(PC.15-0) <-- (PC + rel8*2);(PC.0) <-- 0

描述:如果**80C51**的累加器**A=0,**则发生相对分支.分支范围是+254字节到-256字节,目标地址在代码空间中必须是字相邻.累加器的内容保持不变.这个指令是为了兼容80C51.详情见第9章80C51兼容特性.

注:参考6.3节的跳转范围

尺寸: 位

标志位更新: 无

字节数: 2

时钟数: 6t/3nt

译码:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | rel8 |

**LEA** 调入有效地址

语法: LEA Rd, Rs+offset8/16

操作: (Rd) <-- (Rs)+offset8/16

描述:由源操作数指定的字加上offset数值,存储到目标操作数指定的寄存器中.源和目标操作数都是寄存器.Offset是8/16bit的立即数.操作后源操作数不受影响.

这个指令模拟其它指令的地址计算,例如使用带偏移的间接寻址指令,结果地址储存以备它用.

注:操作后的结果总是字,因为它模拟的是带偏移的间接寻址指令的地址计算.

尺寸: 字-字

标志位更新: 无

**LEA Rd, Rs+offset8**

字节数:3

时钟数:3

操作: (Rd) <-- (Rs)+offset8

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | d | d | d | 0 | s | s | s |

字节3:直接地址低8-bit

**LEA Rd, Rs+offset16**

字节数:4

时钟数:3

操作: (Rd) <-- (Rs)+offset16

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | d | d | d | 0 | s | s | s |

字节3:16-bit偏移地址的高8-bit

字节4:16-bit偏移地址的低8-bit

**LSR** 逻辑右移

语法: LSR dest, count

操作:

(C) <- (dest.0)
(dest.bit n) <- (dest.bit n+1)
count = count-1
Do While (count not equal to 0)
End While

描述:如果**count**操作数**>0,**则由目标操作数指定的变量逻辑右移,移位的次数由**count**操作数决定.移位中最高位添入**0,**最低位移到**C**中.如果**count**操作数**=0,**则不执行移位. **count**操作数为正数,从**0-31.**数据尺寸可以为**8,16,32bit.**在**32-bit**移位中,目标操作数是实际使用尺寸的**1/2.**操作后**count**的数值不变.

注意:

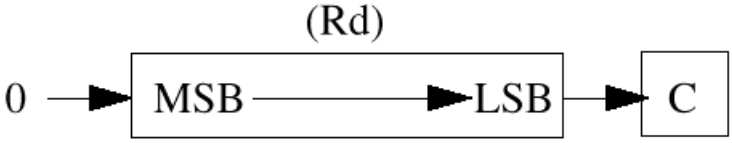
- 对于逻辑左移, 使用ASL忽略N标志.
- 如果超过了数据尺寸, count 将被限制在5 bits, 另外对于立即移位的count,如果count=0,移位停止.
- 双字寄存器是两个相邻的寄存器(R1:R0, R3:R2, R5:R4, 或R7:R6).

尺寸: 字节,字,双字

标志位更新: C, N, Z (LSR后N = 0;如果count = 0,则N不变化)

**LSR Rd, Rs (Rs = 字节寄存器)**

操作:




字节数: 2

时钟数: 8/16 bit 移位 --> 每移位2个bit=4+1

32 bit 移位 --> 每移位2个bit=6+1

译码:

| d | d | d | d | s | s | s | s |
|---|---|---|---|---|---|---|---|

| 1 | 1 | 0 | 0 | SZ1 | SZ0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

**LSR Rd, #data4**

**Rd, #data5**

操作:


字节数: 2

时钟数: 8/16 bit 移位 --> 每移位2个bit=4+1

32 bit 移位 --> 每移位2个bit=6+1

译码: (字节和字尺寸)

| d | d | D | d | #data4 |
|---|---|---|---|---|

| 1 | 1 | 0 | 1 | SZ1 | SZ0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

(双字尺寸)

| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

| d | d | d | #data5 |
|---|---|---|---|

注: SZ1/SZ0 = 00: 操作; SZ1/SZ0 = 01: 保留; SZ1/SZ0 = 10: 字操作; SZ1/SZ0 = 11: 双字操作.

**MOV** 移动数据

语法: MOV dest, src

操作: dest <- src

描述:由源操作数确定的字或字节拷贝到由目标操作数确定的变量.操作后源操作数不受影响.

源和目标操作谁可以是寄存器,由指针寄存器指向的间接地址,带8-bit或16-bit立即数偏移量的间接地址,

直接地址.源操作数可以是包含在指令中的立即数.自动增量寄存器指针也可以用做简单的(无偏移)间接寻址.
尺寸**:** 字节-字节**,**字-字
标志位更新**:** N, Z

**MOV Rd, Rs**
字节数:2
时钟数:3
操作: (Rd) <-- (Rs)
译码:

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | SZ | 0 | 0 | 1 | d | d | d | d | s | s | s | s |

**MOV Rd, [Rs]**
字节数:2
时钟数:3
操作: (Rd) <-- ((WS:Rs))
译码:

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | SZ | 0 | 1 | 0 | d | d | d | d | 0 | s | s | s |

**MOV [Rd], Rs**
字节数:2
时钟数:3
操作: ((WS:Rd)) <-- (Rs)
译码:

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | SZ | 0 | 1 | 0 | s | s | s | s | 1 | d | d | d |

**MOV Rd, [Rs+offset8]**
字节数:3
时钟数:5
操作: (Rd) <-- ((WS:Rs)+offset8)
译码:

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | SZ | 1 | 0 | 0 | d | d | d | d | 0 | s | s | s |

字节3:8-bit偏移地址

**MOV [Rd+offset8], Rs**
字节数:3
时钟数:5
操作: ((WS:Rd)+offset8) <-- (Rs)
译码:

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | SZ | 1 | 0 | 0 | s | s | s | s | 1 | d | d | d |

字节3:8-bit偏移地址

**MOV Rd, [Rs+offset16]**
字节数:4
时钟数:5
操作: (Rd) <-- ((WS:Rs)+offset16)
译码:

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | SZ | 1 | 0 | 1 | d | d | d | d | 0 | s | s | s |

字节3:16-bit偏移地址的高8-bit
字节4:16-bit偏移地址的低8-bit

**MOV [Rd+offset16], Rs**

字节数:4

时钟数:5

操作: ((WS:Rd)+offset16) <-- (Rs)

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | SZ | 1 | 0 | 1 | s | s | s | s | 1 | d | d | d |

字节3:16-bit偏移地址的高8-bit

字节4:16-bit偏移地址的低8-bit

**MOV Rd, [Rs+]**

字节数:2

时钟数:4

操作: (Rd) <-- ((WS:Rs))

(Rs) <-- (Rs) + 1 (字节操作) 或 2 (字操作)

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | SZ | 0 | 1 | 1 | s | s | s | s | 1 | d | d | d |

**MOV [Rd+], Rs**

字节数:2

时钟数:4

操作: ((WS:Rd)) <-- (Rs)

(Rd) <-- (Rd) + 1 (字节操作) 或 2 (字操作)

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | SZ | 0 | 1 | 1 | s | s | s | s | 1 | d | d | d |

**MOV [Rd+], [Rs+]**

字节数:2

时钟数:6

操作: ((WS:Rd)) <-- ((WS:Rs))

(Rs) <-- (Rs) + 1 (字节操作) 或 2 (字操作)

(Rd) <-- (Rd) + 1 (字节操作) 或 2 (字操作)

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | SZ | 0 | 0 | 0 | 0 | d | d | d | 0 | s | s | s |

**MOV direct, Rs**

字节数:3

时钟数:4

操作: (direct) <-- (Rs)

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | SZ | 1 | 1 | 0 | s | s | s | s | 1 | 3-bit直接地址 | | |

字节3:直接地址低8-bit

**MOV Rd, direct**

字节数:3

时钟数:4

操作: (Rd) <-- (direct)

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | SZ | 1 | 1 | 0 | d | d | d | d | 0 | 3-bit直接地址 | | |

字节3:直接地址低8-bit

**MOV direct, [Rs]**

字节数:3

时钟数:4

操作: (direct) <-- ((WS:Rs))

译码:

| 1 | 0 | 1 | 0 | SZ | 0 | 0 | 0 | 1 | s | s | s | 0 | 3-bit直接地址 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

**MOV [Rd], direct**

字节数:3

时钟数:4

操作: ((WS:Rd)) <-- (direct)

译码:

| 1 | 0 | 1 | 0 | SZ | 0 | 0 | 0 | 0 | d | d | d | 0 | 3-bit直接地址 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:直接地址低8-bit

**MOV Rd, #data8**

字节数:3

时钟数:3

操作: (Rd) <-- #data8

译码:

| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | d | d | d | d | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节4:8-bit立即数

**MOV Rd, #data16**

字节数:4

时钟数:3

操作: (Rd) <-- #data16

译码:

| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | d | d | d | d | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit立即数的高8-bit

字节4:16-bit立即数的低8-bit

**MOV [Rd], #data8**

字节数:3

时钟数:3

操作: ((WS:Rd)) <-- #data8

译码:

| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | d | d | d | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit偏移地址

**MOV [Rd], #data16**

字节数:4

时钟数:3

操作: ((WS:Rd)) <-- #data16

译码:

| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | d | d | d | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit立即数的高8-bit

字节4:16-bit立即数的低8-bit

**MOV [Rd+], #data8**

字节数:3

时钟数:4

操作: ((WS:Rd)) <-- #data8

译码:

| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | d | d | d | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit立即数

**MOV [Rd+], #data16**

字节数:4

时钟数:4

操作: ((WS:Rd)) <-- #data16

译码:

| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | d | d | d | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit立即数的高8-bit

字节4:16-bit立即数的低8-bit

**MOV [Rd+offset8], #data8**

字节数:5

时钟数:5

操作: ((WS:Rd)+offset8) <-- #data8

译码:

| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | d | d | d | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit偏移地址

字节4:8-bit立即数

**MOV [Rd+offset8], #data16**

字节数:5

时钟数:5

操作: ((WS:Rd)+offset8) <-- #data16

译码:

| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | d | d | d | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit偏移地址

字节4:16-bit立即数的高8-bit

字节5:16-bit立即数的低8-bit

**MOV [Rd+offset16], #data8**

字节数:5

时钟数:5

操作: ((WS:Rd)+offset16) <-- #data8

译码:

| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | d | d | d | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit偏移地址的高8-bit

字节4:16-bit偏移地址的低8-bit

字节5:8-bit立即数

**MOV [Rd+offset16], #data16**

字节数:6

时钟数:5

操作: ((WS:Rd)+offset16) <-- #data16

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | d | d | d | 1 | 0 | 0 | 0 |

字节3:16-bit偏移地址的高8-bit
字节4:16-bit偏移地址的低8-bit
字节5:16-bit立即数的高8-bit
字节6:16-bit立即数的低8-bit

**MOV direct, #data8**
字节数:4
时钟数:3
操作: (direct) <-- #data8
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 3-bit直接地址 | | | 1 | 0 | 0 | 0 |

字节3:直接地址低8-bit
字节4:8-bit立即数

**MOV direct, #data16**
字节数:5
时钟数:3
操作: (direct) <-- #data16
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 3-bit直接地址 | | | 1 | 0 | 0 | 0 |

字节3:直接地址低8-bit
字节4:16-bit立即数的高8-bit
字节5:16-bit立即数的低8-bit

**MOV direct, direct**
字节数:4
时钟数:4
操作: (direct) <-- (direct)
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | SZ | 1 | 1 | 1 | 0 | d | 3bit目标直接地址 | | | | 3bit源直接地址 | |

字节3:目标直接地址低8-bit
字节4:源直接地址低8-bit

**MOV Rd, USP (**从用户堆栈复制**)**
字节数:2
时钟数:3
操作: (Rd) <-- (USP)
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | d | d | d | d | 1 | 1 | 1 | 1 |

**MOV USP, Rs (**复制到用户堆栈**)**
字节数:2
时钟数:3
操作: (USP) <-- (Rs)
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | s | s | s | s | 1 | 1 | 1 | 1 |

**MOV** 复制指定位到进位

语法**:** MOV C, bit
操作**:** (C) <-- (bit)
描述**:** 复制指定位到进位标志.
尺寸: 位
标志位更新**:**无
注意: C是作为一个目标来写,而不是作为一个状态位.
字节数:3
时钟数:4
译码:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 | 2bit位地址 |

字节3: 位地址的低8-bit

**MOV** 复制进位到指定位

语法**:** MOV bit, C
操作: (bit) <-- (C)
描述**:** 复制进位标志到指定位.
尺寸: 位
标志位更新**:** 无
字节数:3
时钟数:4
译码:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 0 | 2bit位地址 |

字节3: 位地址的低8-bit

**MOVC** 移动代码

语法**:** MOVC Rd, [Rs+]
操作: (Rd) <--代码区域((WS:Rs))
(Rs) <-- (Rs) + 1 (字节操作) 或 2 (字操作)
描述**:**代码区域的内容复制到内部寄存器中.由源操作数定义的字节或字复制到目标操作数指定的变量中.在MOVC中,指针的段可以选择为PC 23-16或CS(当前工作段这里写为WS),DS或ES用于其它指令.
尺寸**:** 字节-字节,字-字
标志位更新**:** N, Z
字节数:2
时钟数:4
译码:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | SZ | 0 | 0 | 0 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| d | d | d | d | 0 | s | s | s |

**MOVC** 移动代码到**A (DPTR)**

语法**:** MOVC A, [A+DPTR]
操作**:** PC <- PC+2
(A) <-- 代码区域(PC.23-16:(A) + (DPTR))
描述: 定位于A+DPTR处的代码字节复制到A.A和DPTR寄存器是予定义寄存器,用于80C51兼容.见第9章的80C51兼容性.
尺寸: 字节-字节
标志位更新**:** N, Z
字节数**:** 2
时钟数**:** 6
译码:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |

**MOVC 移动代码到A(PC)**

语法**:** MOVC A, [A+PC]

操作**:** PC <- PC+2

(A) <-- 代码区域 [PC.23-16: (A +PC.15-0)]

注意: 只使用A+PC的16-bits

描述**:**定位于A+PC处的代码字节复制到A.A寄存器是予定义寄存器,用于80C51兼容.见第9章的80C51兼容性.

尺寸: 字节-字节

标志位更新**:** N, Z

字节数: 2

时钟数: 3

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

**MOVS 短移动**

语法**:** MOVS dest, #data

描述:4-bit有符号立即数移动到目标地址.立即数的符号将延伸到合适的尺寸,然后移动到目标操作数指定的变量,可能是字,字节.立即数的范围是+7 to –8.在较小的数据转移到目标地址时,这个指令能节省时间和空间.

尺寸**:** 字节-字节**,**字-字

标志位更新**:** N, Z

**MOVS Rd, #data4**

字节数: 2

时钟数: 3

操作: (Rd) <-- 符号调整#data4

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | SZ | 0 | 0 | 1 | | d | d | d | d | 4-bit立即数 | | |

**MOVS [Rd], #data4**

字节数: 2

时钟数: 3

操作: ((WS:Rd)) <-- 符号调整#data4

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | SZ | 0 | 1 | 0 | | 0 | d | d | d | 4-bit立即数 | | |

**MOVS [Rd+], #data4**

字节数: 2

时钟数: 4

操作: ((WS:Rd)) <-- 符号调整#data4

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | SZ | 0 | 1 | 1 | | 0 | d | d | d | 4-bit立即数 | | |

**MOVS [Rd+offset8], #data4**

字节数: 3

时钟数: 5

操作: ((WS:Rd)+offset8) <--符号调整#data4

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | SZ | 1 | 0 | 0 | | 0 | d | d | d | 4-bit立即数 | | |

字节3:8-bit偏移地址

**MOVS [Rd+offset16], #data4**

字节数: 4

时钟数: 5

操作: ((WS:Rd)+offset16) <-- 符号调整#data4

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | SZ | 1 | 0 | 1 | 0 | d | d | d | 4-bit立即数 | | | |

字节3:16-bit偏移地址的高8-bit

字节4:16-bit偏移地址的低8-bit

**MOVS direct, #data4**

字节数: 3

时钟数: 3

操作: (direct) <-- 符号调整#data4

译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | SZ | 1 | 1 | 0 | 0 | 3-bit直接地址 | | | 4-bit立即数 | | | |

字节3: 直接地址的低8-bit

**MOVX** 移动外部数据

**语法:** MOVX dest, src

**描述:**移动外部数据到/自内部寄存器.由源操作数决定的字或字节将被复制到目标操作数变量中.此指令可以操作外部数据的地址范围为0-64K.标准的间接寻址指令MOV也可以操作外部数据,但必须在内部RAM边界以上,但MOVX总是实施外部操作.MOVX只能操作外部第一个64K空间.这个指令同80C51代码兼容.

注意:在80C51 MOVX指令中还使用@Ri作为指针(I=0,或1),指针是8-bit,外部总线的高端地址不驱动.XA总是驱动所有的已启动的外部总线地址线.指针的使用依赖于是否使用兼容模式.如果(CM=0,兼容模式关闭,缺省),在数据0段内,R0和R1的16-bit作为地址.如果CM=1(兼容模式开启),R0L或R0H的8-bit作为地址的低8-bit,这时剩下的地址位,包括数据段使用的为0.

尺寸: 字节-字节,字-字

**标志位更新:** N, Z

**MOVX Rd, [Rs]**

字节数: 2

时钟数: 6

操作: (Rd) <-- 外部数据区域 ((Rs))

**译码:**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | SZ | 1 | 1 | 1 | d | d | d | d | 0 | s | s | S |

**MOVX [Rd], Rs**

**字节数:** 2

**时钟数:** 6

**操作:** 外部数据区域 ((Rd)) <-- (Rs)

**译码:**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | SZ | 1 | 1 | 1 | s | s | s | s | 1 | d | d | d |

**MUL.w 16x16** 有符号乘

**MULU.b 8x8** 无符号乘

**MULU.w 16x16** 无符号乘

描述:目标操作数和源操作数相乘.目标寄存器是目标操作数的2倍尺寸(字节相乘是字,字相乘是双字).结果存储到双倍寄存器中.
注意:双寄存器在寄存器队列中字相邻(R1:R0, R3:R2, R5:R4,和R7:R6).
尺寸: 字节-字节,字-字
标志位更新:C, V, N, Z
乘指令总是清除进位标志.V标志在下面的情况下被置位,其余情况被清除:
- MULU.b: 如果相乘的结果大于FFh(高位字节不为0),则V=1.
- MULU.w: 如果相乘的结果大于FFFFh(高位字不为0),则V=1.
- MUL.w: 如果相乘的结果的绝对值大于7FFFh() V=1

范例:
a) MUL.w R0, R5的乘机放在双字寄存器R0中(低有效字在R0中,高有效字在R1中).
b) MULU.b R4L, R4H 的乘机高有效字节放在R4H中,低有效字节放在R4L中.

**MUL.w Rd, Rs**
(有符号 16 bits * 16 bits --> 32 bits)
字节数: 2
时钟数: 12
操作: (Rd+1)<-- (Rd) * (Rs) (有符号乘)的高有效字
(Rd) <-- (Rd) * (Rs) (有符号乘)的低有效字
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | d | d | d | d | s | s | s | s |

**MUL.w Rd, #data16**
(有符号16 bits * 16 bits --> 32 bits)
字节数: 4
时钟数: 12
操作: (Rd+1)<-- (Rd) * #data16 (有符号乘) 高有效字
(Rd) <-- (Rd) * #data16低有效字
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | d | d | d | d | 1 | 0 | 0 | 0 |

字节3:16-bit立即数的高8-bit
字节4:16-bit立即数的低8-bit

**MULU.b Rd, Rs**
(无符号8 bits * 8 bits --> 16 bits)
字节数: 2
时钟数: 12
操作: (RdH) <-- (RdL) * (Rs) (无符号乘)的高有效字节
(RdL) <-- (RdL) * (Rs) 低有效字节
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | d | d | d | d | s | s | s | s |

**MULU.b Rd, #data8**
(无符号8 bits * 8 bits --> 16 bits)
字节数: 3
时钟数: 12
操作: (RdH) <-- (RdL) * #data8 (无符号乘)的高有效字节
(RdL) <-- (RdL) * #data8的低有效字节
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | d | d | d | d | 0 | 0 | 0 | 0 |

字节3:8-bit立即数

**MULU.w Rd, Rs**
(无符号 16 bits * 16 bits --> 32 bits)
字节数**:** 2
时钟数**:** 12
操作**:** (Rd+1)<-- (Rd) * (Rs) (无符号乘)的高有效字
(Rd) <-- (Rd) * (Rs)的低有效字
译码**:**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | d | d | d | d | s | s | s | s |

**MULU.w Rd, #data16**
(无符号16 bits * 16 bits --> 32 bits)
字节数**:** 4
时钟数**:** 12
操作**:** (Rd+1)<-- (Rd) * #data16 (无符号乘)的高有效字
(Rd) <-- (Rd) * #data16的低有效字
译码**:**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | d | d | d | d | 0 | 0 | 0 | 0 |

字节3:16-bit立即数的高8-bit
字节4:16-bit立即数的低8-bit

**NEG** 取负数
语法**:** NEG Rd
操作**:** Rd <-- (Rd) + 1
描述**:** 目标寄存器取负(互补).目标可以是字节,字.
尺寸: 字,字节
标志位更新**:** V, N, Z
当两个互补数溢出时,V=1: 原始值=结果值= 8000 hex(字操作);或 80 hex(字节操作)
字节数**:** 2
时钟数**:** 3
译码**:**

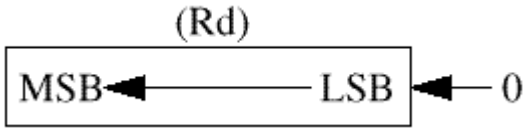
| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | SZ | 0 | 0 | 0 | d | d | d | d | 1 | 0 | 1 | 1 |

**NOP** 空操作
语法**:** NOP
操作**:** PC <- PC + 1
描述**:**继续执行以后的操作.这个指令的长度为1字节,可以用来是分支目标处的指令为字相邻,或其它用途.也可以用来产生精确的时间延迟.
尺寸: 无
标志位更新**:** 无
字节数**:** 1
时钟数**:** 3
译码**:**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**NORM** 标准化
语法: NORM Rd, Rs

操作:




描述:对目标进行逻辑左移,直到最高位为1,移位的次数放置在源寄存器中.数据可以是8,16,32 bit.如果目标的最高位已经为1,返回的记数为0.如果目标为0,返回的记数为0,N=0,Z=1.对于其它的情况,N=1,Z=0.
注意:双字是两个相邻的寄存器(R1:R0, R3:R2, R5:R4, 或R7:R6).最后一对最好别用,因为R7是堆栈指针.
尺寸: 字节,字,双字
标志位更新: N, Z
字节数: 2
时钟数: 对于8或16 bit 移位 -> 每2个移位4 + 1
对于32 bit 移位 -> 每2个移位6 + 1
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| d | d | d | d | s | s | s | s | 1 | 1 | 0 | 0 | SZ1 | SZ0 | 1 | 1 |

注:
SZ1/SZ0 = 00:字节操作
SZ1/SZ0 = 01:保留
SZ1/SZ0 = 10:字操作
SZ1/SZ0 = 11:双字操作.

**OR 逻辑或**

语法: OR dest, src
描述:源和目标的逻辑或.源数据在操作后不受影响.
尺寸: 字节-字节,字-字
标志位更新: N, Z

**OR Rd, Rs**
字节数: 2
时钟数: 3
操作: (Rd) <-- (Rd) + (Rs)
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | SZ | 0 | 0 | 1 | d | d | d | d | s | s | s | s |

**OR Rd, [Rs]**
字节数: 2
时钟数: 4
操作: (Rd) <-- (Rd) + ((WS:Rs))
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | SZ | 0 | 1 | 0 | d | d | d | d | 0 | s | s | s |

**OR [Rd], Rs**
字节数: 2
时钟数: 4
操作: ((WS:Rd)) <-- ((WS:Rd)) + (Rs)
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | SZ | 0 | 1 | 0 | s | s | s | s | 1 | d | d | d |

**OR Rd, [Rs+offset8]**
字节数: 3

时钟数: 6
操作: (Rd) <-- (Rd) (OR) ((WS:Rs)+offset8)
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | SZ | 1 | 0 | 0 | d | d | d | d | 0 | s | s | s |

字节3:8-bit偏移地址

**XOR [Rd+offset8], Rs**
字节数: 3
时钟数: 6
操作: ((WS:Rd)+offset8) <-- ((WS:Rd)+offset8) (OR) (Rs)
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | SZ | 1 | 0 | 0 | s | s | s | s | 1 | d | d | d |

字节3:8-bit偏移地址

**OR Rd, [Rs+offset16]**
字节数: 4
时钟数: 6
操作: (Rd) <-- (Rd) (OR) ((WS:Rs)+offset16)
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | SZ | 1 | 0 | 1 | d | d | d | d | 0 | s | s | s |

字节3:16-bit偏移地址的高8-bit
字节4:16-bit偏移地址的低8-bit

**OR [Rd+offset16], Rs**
字节数: 4
时钟数: 6
操作: ((WS:Rd)+offset16) <-- ((WS:Rd)+offset16) (OR) (Rs)
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | SZ | 1 | 0 | 1 | s | s | s | s | 1 | d | d | d |

字节3:16-bit偏移地址的高8-bit
字节4:16-bit偏移地址的低8-bit

**OR Rd, [Rs+]**
字节数: 2
时钟数: 5
操作: (Rd) <-- (Rd) (OR) ((WS:Rs))
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | SZ | 0 | 1 | 1 | d | d | d | d | 0 | s | s | s |

**OR [Rd+], Rs**
字节数: 2
时钟数: 5
操作: ((WS:Rd)) <-- ((WS:Rd)) (OR) (Rs)
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | SZ | 0 | 1 | 1 | s | s | s | s | 1 | d | d | d |

**OR direct, Rs**
字节数: 3
时钟数: 4
操作: (direct) <-- (direct) (OR) (Rs)

译码:

| 0 | 1 | 1 | 1 | SZ | 1 | 1 | 0 | s | s | s | s | 1 | 3bit直接地址 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:直接地址低8-bit

**OR Rd, direct**

字节数: 3

时钟数: 4

操作: (Rd) <-- (Rd) (OR) (direct)

译码:

| 0 | 1 | 1 | 0 | SZ | 1 | 1 | 0 | d | d | d | d | 0 | 3bit直接地址 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:直接地址低8-bit

**OR Rd, #data8**

字节数: 3

时钟数: 3

操作: (Rd) <-- (Rd) (OR) #data8

译码:

| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | d | d | d | d | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit立即数

**OR Rd, #data16**

字节数: 4

时钟数: 3

操作: (Rd) <-- (Rd) (OR) #data16

译码:

| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | d | d | d | d | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit立即数的高8-bit

字节4:16-bit立即数的低8-bit

**OR [Rd], #data16**

字节数: 4

时钟数: 4

操作: ((WS:Rd)) <-- ((WS:Rd)) + #data16

译码:

| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | d | d | d | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit立即数的高8-bit

字节4:16-bit立即数的低8-bit

**OR [Rd+], #data8**

字节数: 3

时钟数: 5

操作: ((WS:Rd)) <-- ((WS:Rd)) + #data8

译码:

| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | d | d | d | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit立即数

**OR [Rd+], #data16**

字节数: 4

时钟数: 5

操作: ((WS:Rd)) <-- ((WS:Rd)) + #data16

译码:

| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | d | d | d | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit立即数的高8-bit
字节4:16-bit立即数的低8-bit

**OR [Rd+offset8], #data8**
字节数: 4
时钟数**:** 6
操作**:** ((WS:Rd)+offset8) <-- ((WS:Rd)+offset8) + #data8
译码**:**

| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | d | d | d | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit偏移地址
字节4:8-bit立即数

**OR [Rd+offset8], #data16**
字节数**:** 5
时钟数**:** 6
操作**:** ((WS:Rd)+offset8) <-- ((WS:Rd)+offset8) + #data16
译码**:**

| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | d | d | d | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit偏移地址
字节4:16-bit立即数的高8-bit
字节5:16-bit立即数的低8-bit

**OR [Rd+offset16], #data8**
字节数**:** 5
时钟数**:** 6
操作**:** ((WS:Rd)+offset16) <-- ((WS:Rd)+offset16) + #data8
译码**:**

| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | d | d | d | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit偏移地址的高8-bit
字节4:16-bit偏移地址的低8-bit
字节5:8-bit立即数

**OR [Rd+offset16], #data16**
字节数**:** 6
时钟数**:** 6
操作**:** ((WS:Rd)+offset16) <-- ((WS:Rd)+offset16) + #data16
译码**:**

| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 3-bit直接地址 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit偏移地址的高8-bit
字节4:16-bit偏移地址的低8-bit
字节5:16-bit立即数的高8-bit
字节6:16-bit立即数的低8-bit

**OR direct, #data8**
字节数**:** 4
时钟数**:** 4
操作**:** (direct) <-- (direct) + #data8
译码**:**

| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| 0 | 3-bit直接地址 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|

字节3: 直接地址的低8-bit
字节4:8-bit立即数

**OR direct, #data16**
字节数: 5
时钟数: 4
操作: (direct) <-- (direct) + #data16
译码:

| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| 0 | 3-bit直接地址 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|

字节3: 直接地址的低8-bit
字节4:16-bit立即数的高8-bit
字节5:16-bit立即数的低8-bit

**ORL** 进位同位逻辑或

语法: ORL C, bit
操作**:** (C) <-- (C) + (bit)
描述: 指定位同进位逻辑或.读取指定位,再同进位逻辑或.(C作为目标位,而不是作为状态位)
尺寸**:** 位
标志位更新**:** 无
字节数**:** 3
时钟数**:** 4
译码**:**

| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

| 0 | 1 | 1 | 0 | 0 | 0 | 2-bit |
|---|---|---|---|---|---|---|

字节 3: 位地址的低8-bit

**ORL** 进位同位反逻辑或

语法: ORL C, /bit
操作**:** (C) <-- (C) + (bit)
描述: 指定位取反同进位逻辑或.读取指定位,然后取反,再同进位逻辑或.(C作为目标位,而不是作为状态位)
标志位更新**:** 无
字节数**:** 3
时钟数**:** 4
译码**:**

| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

| 0 | 1 | 1 | 1 | 0 | 0 | 2-bit |
|---|---|---|---|---|---|---|

字节 3: 位地址的低8-bit

**POP** 弹出
**POPU** 从用户堆栈中弹出

语法**:** POP dest
描述**:**从堆栈中弹出数据**,**并写到指定的直接寻址地址**.**数据可以是字节**,**字**.POP**使用当前的堆栈**,**而**POPU**使用用户堆栈**.**
尺寸**:** 字节**,**字
标志位更新**:** 无

**POP direct**
字节数: 3
时钟数: 5
操作: (direct) <-- ((SP))
(SP) <-- (SP) + 2
译码:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | SZ | 1 | 1 | 1 |

| | | | | | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 3-bit直接地址 |

字节 3: 8 bits直接地址

**POPU direct**
字节数: 3
时钟数: 5
操作: (direct) <-- ((USP))
(USP) <-- (USP) + 2
译码:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | SZ | 1 | 1 | 1 |

| | | | | | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 3-bit直接地址 |

字节 3: 8 bits直接地址

**POP** 组弹出
**POPU** 用户模式下组弹出

---

语法**:** POP Rlist
POPU Rlist
描述**:** 把指定的寄存器**(**一个或多个**)**弹出堆栈**.**使用一个单一的指令**,**可以把**R0-R7**有选择的以字的形式弹出堆栈**,**或把 **R0L-R3H/R4L-R7H**有选择的以字节的形式弹出堆栈**.POP**使用当前的堆栈**,**而**POPU**使用用户堆栈**.**
注:Rlist是表明每个寄存器是否弹出的位图.寄存器的顺序是R7,R6, R5,…, R0(字寄存器),或R3H.... R0L, 或 R7H... R4L(字节寄存器).弹出的顺序是从右到左,在Rlist中最右边指定的寄存器首先被弹出.弹出的顺序和相应压入的顺序相反.注意,如果使用相同的寄存器列表,分别用于PUSH和POP,则原始的寄存器内容将被恢复.每一个寄存器被使用的次序并不重要,因为Rlist操作数将被译码成固定的顺序(见下面)
尺寸: 字节,字
标志位更新**:** 无

**POP Rlist**
字节数: 2
时钟数: 每个附加寄存器4 + 2
操作: 对所选的寄存器重复执行:
(Ri) <-- ((SP))
(SP) <-- (SP) + 2
译码:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | HL | 1 | 0 | SZ | 1 | 1 | 1 |

| |
|---|
| 寄存器列表Rlist |

**POPU Rlist**
字节数: 2
时钟数: 每个附加寄存器4 + 2
操作: 对所选的寄存器重复执行:
(Ri) <-- ((USP))
(USP) <-- (USP) + 2

译码:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | HL | 1 | 1 | SZ | 1 | 1 | 1 | 寄存器列表Rlist |

Rlist位定义在寄存器组弹出的寄存器选择(R4L 到 R7H)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| R7H | R7L | R6H | R6L | R5H | R5L | R4H | R4L |

Rlist位定义在寄存器组弹出的寄存器选择(R0L 到 R3H)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| R3H | R3L | R2H | R2L | R1H | R1L | R0H | R0L |

Rlist位定义在寄存器组弹出的寄存器选择(R0 到 R7)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| R7 | R7 | R6 | R6 | R5 | R5 | R4 | R4 |

**PUSH** 压入堆栈
**PUSHU** 压入用户堆栈

语法**:** PUSH src
PUSHU src
描述**:**指定的直接 寻址数据被压入堆栈.数据尺寸可以是字节或字.PUSH使用当前的堆栈指针,PUSHU使用用户堆栈.
尺寸**:** 字节,字
标志位更新**:** 无

**PUSH direct**
字节数: 3
时钟树: 5
操作: (SP) <-- (SP) - 2
((SP)) <-- (direct)
译码**:**

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | SZ | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 3-bit直接地址 |

字节3: 8 bits直接地址

**PUSHU direct**
字节数**:** 3
时钟数: 5
操作: (USP) <-- (USP) - 2
((USP)) <-- (direct)
译码:

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | SZ | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 3-bit直接地址 |

字节 3: 8 bits直接地址

**PUSH** 组压入
**PUSHU** 用户模式下的组压入

语法**:** PUSH Rlist
PUSHU Rlist
描述**:**把指定的寄存器(一个或多个)压入堆栈.使用一个单一的指令,可以把R0-R7有选择的以字的形式压入堆栈,或把 R0L-R3H/R4L-R7H有选择的以字节的形式压入堆栈.PUSH使用当前的堆栈,而PUSHU使用用户堆栈.
注:Rlist是表明每个寄存器是否压入的位图.寄存器的顺序是R7,R6, R5,…, R0(字寄存器),或R3H.... R0L, 或 R7H... R4L(字节寄存器).压入的顺序是从左到右,在Rlist中最左边指定的寄存器首先被压入.压入的顺

序和相应弹出的顺序相反.注意,如果使用相同的寄存器列表,分别用于PUSH和POP,则原始的寄存器内容将被恢复.每一个寄存器被使用的次序并不重要,因为Rlist操作数将被译码成固定的顺序(见下面)
尺寸: 字节,字
标志位更新: 无

**PUSHU Rlist**
字节数: 2
时钟数: 每个附加寄存器3 + 3
操作: 对所选的寄存器重复执行:
(USP) <-- (USP) - 2
((USP)) <-- (Ri)
译码:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | HL | 0 | 1 | SZ | 1 | 1 | 1 |

| |
|---|
| 寄存器列表Rlist |

Rlist位定义在寄存器组弹出的寄存器选择(R4L 到 R7H)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| R7H | R7L | R6H | R6L | R5H | R5L | R4H | R4L |

Rlist位定义在寄存器组弹出的寄存器选择(R0L 到 R3H)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| R3H | R3L | R2H | R2L | R1H | R1L | R0H | R0L |

Rlist位定义在寄存器组弹出的寄存器选择(R0 到 R7)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| R7 | R7 | R6 | R6 | R5 | R5 | R4 | R4 |

**RESET** 软件复位

语法: RESET
操作: (PC) <-- vector(0)
(PSW) <-- vector(0)
(SFRs) <-- reset values (refer to the description of reset for details)
描述:整片复位,如同外部施加复位信号一样,但不对输入配置如EA,BUSW进行采样.当执行复位指令时,片子进行内部复位,但无外部复位脉冲产生.以上的输入是在外部复位脉冲上升沿被锁存,因此不会影响带片子的配置.
标致位更新: 全部的PSW将由复位矢量中的PSW值取代.
字节数: 2
时钟数: 18
译码:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

**RET** 从子程序中返回

语法: RET
操作: (PC) <-- ((SP))
(SP) <-- (SP) + 4
描述:一个24-bit地址从堆栈中弹出以取代现在程序计数器的值.这个指令用于从由CALL或FCALL调用的子程序中返回.
注:如果在0页模式,只有16-bit的地址从堆栈中弹出.
尺寸: 无
标志位更新: 无
字节数: 2
时钟数: 8/6 (PZ)
译码:

| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

**RETI** 从中断返回

语法**:** RETI

操作**:** (PSW) <-- ((SSP))
(PC.23-0) <-- ((SSP))
(SSP) <-- (SSP) + 6

描述:从堆栈中弹出24-bit返回地址取代现在的程序计数器数值.程序状态字也从堆栈中弹出和恢复. 这个指令是系统级的指令(限于系统模式),用于从中断或异常中返回.在用户模式下使用RETI指令将产生一个中断.

注:如果在0页模式,只有16-bit的地址从堆栈中弹出.

尺寸: 无

标志位更新**:**从系统堆栈中弹出的PSW将取代现有的值.

字节数**:** 2

时钟数**:** 10/8 (PZ)

译码:

| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

**RL** 左旋转

语法**:** RL Rd, #data4

操作**:**


count <- #data4
Do While (count not equal to 0)
(dest $_0$ ) <- (dest $_{msb}$ )
(dest $_n$ ) <- (dest $_{n-1}$ )
(count) <- count -1
End While

描述: 只要记数操作数不为0,目标操作数通过进位向右旋转,旋转的次数由指定的立即数决定.数据可以是8或16bit.位移动的次数可以为0-15.如果记数操作数为0,则不进行旋转.

尺寸: 字节,字

标志位更新**:** N, Z

字节数**:** 2

时钟数**:** 每个移位4 + 1

译码:

| 1 | 1 | 0 | 1 | SZ | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|

| d | d | d | D | 4-bit立即数 |
|---|---|---|---|---|

**RLC** 通过进位左旋转

语法**:** RLC Rd, #data4

操作**:**

count <- #data4
Do While (count not equal to 0)
(temp) <- (C)
(C) <- (dest msb )
(dest n ) <- (dest n-1 )
(dest 0 ) <- (temp)
(count) <- count -1
End While

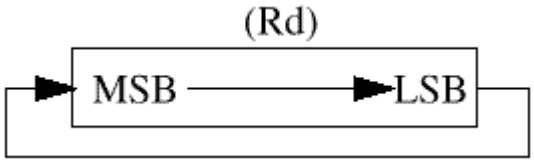
描述: 只要记数操作数不为0,目标操作数通过进位向右旋转,旋转的次数由指定的立即数决定.数据可以是8或16bit.位移动的次数可以为0-15.如果记数操作数为0,则不进行旋转.
尺寸: 字,字节
标志位更新: C, N, Z
字节数: 2
时钟数: 每个移位4 + 1
译码:

| 1 | 1 | 0 | 1 | SZ | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|

| d | d | d | D | 4-bit立即数 |
|---|---|---|---|---|

**RR** 右旋转

语法: RR Rd, #data4
操作:




count <- #data4
Do While (count not equal to 0)
(dest msb ) <- (dest 0 )
(dest n-1 ) <- (dest n )
(count) <- count -1

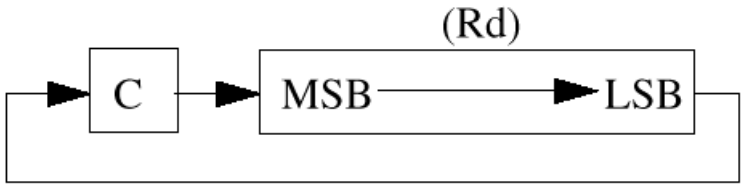
描述:如果记数操作数大于0,目标操作数向右旋转,旋转的次数由指定的立即数决定.数据可以是8或16-bit.移动的位置可以为0-15.如果记数操作数为0,则不会旋转.
尺寸: 字节,字
标志位更新: N, Z
字节数: 2
时钟数: 每个移位4 + 1
译码:

| 1 | 0 | 1 | 1 | SZ | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

| d | d | d | d | 4-bit立即数 |
|---|---|---|---|---|

**RRC** 通过进位向右旋转

语法: RRC Rd, #data4
操作:




count <- #data4
Do While (count not equal to 0)
(temp) <- (C)

(C) <- (dest 0 )
(dest n ) <- (dest n+1 )
(dest msb ) <- (temp)
(count) <- count -1
End While

描述:只要记数操作数不为0,目标操作数通过进位向右旋转,旋转的次数由指定的立即数决定.数据可以是8或16bit.位移动的次数可以为0-15.如果记数操作数为0,则不进行旋转.
尺寸: 字,字节
标志位更新: C, N, Z
字节数:2
时钟数: 每个移位4 + 1
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | | SZ | 1 | 1 | 1 | | d | d | d | d | 4-bit立即数 | | |

**SETB** 设置位

语法: SETB bit
操作: (bit) <-- 1
描述: 把指定的位写1.
尺寸: 位
标志位更新: 无
字节数:3
时钟数:4
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | | 0 | 0 | 0 | 1 | 0 | 0 | 2-bit地址 |

字节 3: 位地址的低8-bit

**SEXT** 符号延伸

语法: SEXT Rd
操作: 如果 N = 1 则(Rd) <-- FF (字节模式)或 FFFF (字模式)
如果 N = 0 则(Rd) <-- 00 (字节模式)或 0000 (字模式)
描述: 把上次ALU操作的N标志拷贝到目标寄存器中.目标寄存器可以是字节或字寄存器.
范例: SEXT.b R1
如果上次运算的结果使N=1,则R1 <-- FF
尺寸: 字节,字
标志位更新: 无
字节数:2
时钟数:3
译码:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| d | d | d | d | 1 | 0 | 0 | 1 | | 1 | 0 | 0 | 1 | SZ | 0 | 0 | 0 |

**SUB** 整数减

语法: SUB dest, src
操作: dest <- dest - src
描述: 目的操作数减去源操作数,结果保存在目的操作数中.操作后,源操作数不受影响.
尺寸: 字节-字节,字-字
标志位更新: C, AC, V, N, Z

**SUB Rd, Rs**
字节数: 2
时钟数: 3

操作:  (Rd) <-- (Rd) - (Rs)
译码**:**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | SZ | 0 | 0 | 1 | d | d | d | d | s | s | s | s |

**SUB Rd, [Rs]**
字节数: 2
时钟数: 4
操作:  (Rd) <-- (Rd) - ((WS:Rs))
译码**:**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | SZ | 0 | 1 | 0 | d | d | d | d | 0 | s | s | s |

**SUB [Rd], Rs**
字节数: 2
时钟数: 4
操作:  ((WS:Rd)) <-- ((WS:Rd)) - (Rs)
译码**:**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | SZ | 0 | 1 | 0 | s | s | s | s | 1 | d | d | d |

**SUB Rd, [Rs+offset8]**
字节数: 3
时钟数: 6
操作:  (Rd) <-- (Rd) - ((WS:Rs)+offset8)
译码**:**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | SZ | 1 | 0 | 0 | d | d | d | d | 0 | s | s | s |

字节3:8-bit偏移地址

**SUB [Rd+offset8], Rs**
字节数: 3
时钟数: 6
操作:  ((WS:Rd)+offset8) <-- ((WS:Rd)+offset8) - (Rs)
译码

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | SZ | 1 | 0 | 0 | s | s | s | s | 1 | d | d | d |

字节3:8-bit偏移地址

**SUB Rd, [Rs+offset16]**
字节数: 4
时钟数: 6
操作:  (Rd) <-- (Rd) - ((WS:Rs)+offset16)
译码

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | SZ | 1 | 0 | 1 | s | s | s | s | 1 | d | d | d |

字节3:16-bit偏移地址的高8-bit
字节4:16-bit偏移地址的低8-bit

**SUB [Rd+offset16], Rs**
字节数: 4
时钟数: 6
操作:  ((WS:Rd)+offset16) <-- ((WS:Rd)+offset16) - (Rs)
译码

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | SZ | 1 | 0 | 1 | s | s | s | s | 1 | d | d | d |

字节3:16-bit偏移地址的高8-bit
字节4:16-bit偏移地址的低8-bit

**SUB Rd, [Rs+]**
字节数: 2
时钟数: 5
操作:  (Rd) <-- (Rd) - ((WS:Rs))
译码

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | SZ | 0 | 1 | 1 | d | d | d | d | 0 | s | s | s |

**SUB [Rd+], Rs**
字节数: 2
时钟数: 5
操作:  ((WS:Rd)) <-- ((WS:Rd)) - (Rs)
译码

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | SZ | 0 | 1 | 1 | s | s | s | s | 1 | d | d | d |

**SUB direct, Rs**
字节数: 3
时钟数: 4
操作:  (direct) <-- (direct) - (Rs)
译码

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | SZ | 1 | 1 | 0 | s | s | s | s | 1 | 3-bit直接地址 |

字节3:直接地址低3-bit

**SUB Rd, direct**
字节数: 3
时钟数: 4
操作:  (Rd) <-- (Rd) - (direct)
译码

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | SZ | 1 | 1 | 0 | d | d | d | d | 0 | 3-bit直接地址 |

字节3:直接地址低3-bit

**SUB Rd, #data8**
字节数: 3
时钟数: 4
操作:  (Rd) <-- (Rd) - #data8
译码

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | d | d | d | d | 0 | 0 | 1 | 0 |

字节3:8-bit立即数

**SUB Rd, #data16**
字节数: 3
时钟数: 4
操作:  (Rd) <-- (Rd) - #data16
译码

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | d | d | d | d | 0 | 0 | 1 | 0 |

字节3:16-bit立即数的高8-bit
字节4:16-bit立即数的低8-bit

**SUB [Rd], #data8**
字节数: 3

时钟数: 4
操作:  ((WS:Rd)) <-- ((WS:Rd)) - #data8
译码

| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | d | d | d | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节4:8-bit立即数

**SUB [Rd], #data16**
字节数: 3
时钟数: 5
操作:  ((WS:Rd)) <-- ((WS:Rd)) - #data16
译码

| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | d | d | d | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit立即数的高8-bit
字节4:16-bit立即数的低8-bit

**SUB [Rd+], #data8**
字节数: 3
时钟数: 5
操作:  ((WS:Rd)) <-- ((WS:Rd)) - #data8
译码

| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | d | d | d | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit立即数

**SUB [Rd+], #data16**
字节数: 4
时钟数: 5
操作:  ((WS:Rd)) <-- ((WS:Rd)) - #data16
译码

| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | d | d | d | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit立即数的高8-bit
字节4:16-bit立即数的低8-bit

**SUB [Rd+offset8], #data8**
字节数: 4
时钟数: 6
操作:  ((WS:Rd)+offset8) <-- ((WS:Rd)+offset8) - #data8
译码

| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | d | d | d | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit偏移地址
字节4:8-bit立即数

**SUB [Rd+offset8], #data16**
字节数: 5
时钟数: 6
操作:  ((WS:Rd)+offset8) <-- ((WS:Rd)+offset8) - #data16
译码

| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | d | d | d | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit偏移地址
字节4:16-bit立即数的高8-bit
字节5:16-bit立即数的低8-bit

**SUB [Rd+offset16], #data8**

字节数: 5

时钟数: 6

操作:  ((WS:Rd)+offset16) <-- ((WS:Rd)+offset16) - #data8

译码

| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|

| 0 | d | d | d | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|

字节3:16-bit偏移地址的高8-bit

字节4:16-bit偏移地址的低8-bit

字节5:8-bit立即数

**SUB [Rd+offset16], #data16**

字节数: 6

时钟数: 6

操作:  ((WS:Rd)+offset16) <-- ((WS:Rd)+offset16) - #data16

译码

| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|

| 0 | d | d | d | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|

字节3:16-bit偏移地址的高8-bit

字节4:16-bit偏移地址的低8-bit

字节5:16-bit立即数的高8-bit

字节6:16-bit立即数的低8-bit

**SUB direct, #data8**

字节数: 4

时钟数: 4

操作:  (direct) <-- (direct) - #data8

译码

| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| 0 | 3-bit直接地址 | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|

字节3:直接地址的低8-bit

字节4:8-bit立即数

**SUB direct, #data16**

字节数: 5

时钟数: 4

操作:  (direct) <-- (direct) - #data16

译码

| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| 0 | 3-bit直接地址 | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|

字节3:直接地址的低8-bit

字节4:16-bit立即数的高8-bit

字节5:16-bit立即数的低8-bit

**SUBB** 带借位减

语法: SUBB dest, src

操作**:** dest <- dest - src - C

描述**:** 目的操作数带借位减去源操作数,结果保存在目的操作数内.源操作数不受操作的影响.

如果由上次操作得到的进位=0,结果精确的等于操作数的差;如果C=1,结果小于操作数的差.

这种减法的形式用来支持多精度的运算.这样使用时,要先清除C,然后从低有效部分运算到高有效部分.

尺寸**:** 字节-字节,字-字

标志位更新**:** C, AC, V, N, Z

**SUBB Rd, Rs**
字节数: 2
时钟数: 3
操作:  (Rd) <-- (Rd) - (Rs) - (C)
译码

| 0 | 0 | 1 | 1 | SZ | 0 | 0 | 1 | d | d | d | d | 0 | s | s | s |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

SUBB Rd, [Rs]
字节数: 2
时钟数: 4
操作:  (Rd) <-- (Rd) - ((WS:Rs)) - (C)
译码

| 0 | 0 | 1 | 1 | SZ | 0 | 1 | 0 | d | d | d | d | 0 | s | s | s |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

**SUBB [Rd], Rs**
字节数: 2
时钟数: 4
操作:  ((WS:Rd)) <-- ((WS:Rd)) - (Rs) - (C)
译码

| 0 | 0 | 1 | 1 | SZ | 0 | 1 | 0 | s | s | s | s | 1 | d | d | d |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

**SUBB Rd, [Rs+offset8]**
字节数: 3
时钟数: 6
操作:  (Rd) <-- (Rd) - ((WS:Rs)+offset8) - (C)
译码

| 0 | 0 | 1 | 1 | SZ | 1 | 0 | 0 | d | d | d | d | 0 | s | s | s |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit偏移地址

**SUBB [Rd+offset8], Rs**
字节数: 3
时钟数: 6
操作:  ((WS:Rd)+offset8) <-- ((WS:Rd)+offset8) - (Rs) - (C)
译码

| 0 | 0 | 1 | 1 | SZ | 1 | 0 | 0 | s | s | s | s | 1 | d | d | d |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit偏移地址

**SUBB Rd, [Rs+offset16]**
字节数: 4
时钟数: 6
操作:  (Rd) <-- (Rd) - ((WS:Rs)+offset16) - (C)
译码

| 0 | 0 | 1 | 1 | SZ | 1 | 0 | 1 | d | d | d | d | 0 | s | s | s |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit偏移地址高8-bit
字节4:16-bit偏移地址低8-bit

**SUBB [Rd+offset16], Rs**
字节数: 4
时钟数: 6
操作:  ((WS:Rd)+offset16) <-- ((WS:Rd)+offset16) - (Rs) - (C)

译码

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | SZ | 1 | 0 | 1 | s | s | s | s | 1 | d | d | d |

字节3:16-bit偏移地址高8-bit
字节4:16-bit偏移地址低8-bit

**SUBB Rd, [Rs+]**
字节数: 2
时钟数: 5
操作: (Rd) <-- (Rd) - ((WS:Rs)) - (C)
译码

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | SZ | 0 | 1 | 1 | d | d | d | d | 0 | s | s | s |

**SUBB [Rd+], Rs**
字节数: 2
时钟数: 5
操作: ((WS:Rd)) <-- ((WS:Rd)) - (Rs) - (C)
译码

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | SZ | 0 | 0 | 1 | s | s | s | s | 1 | d | d | d |

**SUBB direct, Rs**
字节数: 3
时钟数: 4
操作: (direct) <-- (direct) - (Rs) - (C)
译码

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | SZ | 1 | 1 | 0 | s | s | s | s | 1 | 3-bit直接地址 | | |

字节3:直接地址的低8-bit.

**SUBB Rd, direct**
字节数: 3
时钟数: 4
操作: (Rd) <-- (Rd) - (direct) - (C)
译码

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | SZ | 1 | 1 | 0 | d | d | d | d | 0 | 3-bit直接地址 | | |

字节3:直接地址的低8-bit.

**SUBB Rd, #data8**
字节数: 3
时钟数: 3
操作: (Rd) <-- (Rd) - #data8 - (C)
译码

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | d | d | d | d | 0 | 0 | 1 | 1 |

字节3:8-bit立即数

**SUBB Rd, #data16**
字节数: 4
时钟数: 3
操作: (Rd) <-- (Rd) - #data16 - (C)
译码

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | d | d | d | d | 0 | 0 | 1 | 1 |

字节3:16-bit立即数的高8 bit

字节4:16-bit立即数的低8 bit

**SUBB [Rd], #data8**

字节数: 3

时钟数: 3

操作: ((WS:Rd)) <-- ((WS:Rd)) - #data8 - (C)

译码

| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | d | d | d | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit立即数

**SUBB [Rd], #data16**

字节数: 4

时钟数: 4

操作: ((WS:Rd)) <-- ((WS:Rd)) - #data16 - (C)

译码

| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | d | d | d | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit立即数的高8 bit

字节4:16-bit立即数的低8 bit

**SUBB [Rd+], #data8**

字节数: 3

时钟数: 5

操作: ((WS:Rd)) <-- ((WS:Rd)) - #data8 - (C)

译码

| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | d | d | d | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit立即数

**SUBB [Rd+], #data16**

字节数: 4

时钟数: 6

操作: ((WS:Rd)) <-- ((WS:Rd)) - #data16 - (C)

译码

| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | d | d | d | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit立即数的高8 bit

字节4:16-bit立即数的低8 bit

**SUBB [Rd+offset8], #data8**

字节数: 4

时钟数: 6

操作: ((WS:Rd)+offset8) <-- ((WS:Rd)+offset8) - #data8 - (C)

译码

| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | d | d | d | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit偏移地址

字节4:16-bit立即数的高8 bit

字节5:8-bit立即数

**SUBB [Rd+offset8], #data16**

字节数: 5

时钟数: 6

操作: ((WS:Rd)+offset8) <-- ((WS:Rd)+offset8) - #data16 - (C)

译码

| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | d | d | d | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit偏移地址
字节4:16-bit立即数的高8 bit
字节5:16-bit立即数的低8 bit

**SUBB [Rd+offset16], #data8**

字节数: 5
时钟数: 6
操作: ((WS:Rd)+offset16) <-- ((WS:Rd)+offset16) - #data8 - (C)
译码

| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | d | d | d | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit偏移地址的高8 bit
字节4:16-bit偏移地址的低8 bit
字节5:8-bit立即数

**SUBB [Rd+offset16], #data16**

字节数: 6
时钟数: 6
操作: ((WS:Rd)+offset16) <-- ((WS:Rd)+offset16) - #data16 - (C)
译码

| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | d | d | d | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit偏移地址的高8 bit
字节4:16-bit偏移地址的低8 bit
字节5:16-bit立即数的高8 bit
字节6:16-bit立即数的低8 bit

**SUBB direct, #data8**

字节数: 4
时钟数: 4
操作: (direct) <-- (direct) - #data8 - (C)
译码

| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 3 bit直接地址 | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:直接地址的低8 bit
字节4:8-bit立即数

**SUBB direct, #data16**

字节数: 5
时钟数: 4
操作: (direct) <-- (direct) - #data16 - (C)
译码

| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 3 bit直接地址 | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:直接地址的低8 bit
字节4:16-bit立即数的高8-bit
字节5:16-bit立即数的低8-bit

**TRAP** 软件中断

语法: TRAP #data4
操作: (PC) <-- (PC) + 2
(SSP) <-- (SSP) - 6

((SSP)) <-- (PC)
((SSP)) <-- (PSW)
(PSW) <-- 代码区 (trap vector (#data4))
(PC.15-0) <-- 代码区 (trap vector (#data4))
(PC.23-16) <-- 0; (PC.0) <-- 0

**描述:** 引起指定的软件中断.执行服务程序的入口点由矢量表提供.Trap程序执行完毕后,使用RETI指令返回,恢复运行.Trap像一个立即中断,可以调用一个运行在系统模式下的程序片段.在应用程序中可以使用它获得系统级别的动作,例如转换数据段寄存器.更详细的描述见中断和异常章节.

注:处理程序的地址必须是字相邻,因为PC在指向服务程序以前总是被强制成偶数.

**尺寸:** 无

**标志更新:** 无

**字节:** 2

**时钟周期:** 23/19 (PZ)

**译码:**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 4bit立即数 | | | |

## XCH 交换

**语法:** XCH dest, src

**操作**: dest <--> src

**描述:** 源操作数和目的操作数交换.

**尺寸:** 字节-字节,字-字.

**标志更新:** 无

### XCH Rd, Rs

字节数: 2

时钟数: 5

操作: (Rd) <--> (Rs)

译码

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | SZ | 0 | 0 | 0 | d | d | d | d | 0 | s | s | s |

### XCH Rd, [Rs]

字节数: 2

时钟数: 6

操作: (Rd) <--> ((WS:Rs))

译码

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | SZ | 0 | 0 | 0 | d | d | d | d | 0 | s | s | s |

### XCH Rd, direct

字节数: 3

时钟数: 6

操作: (Rd) <--> (direct)

译码

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | SZ | 0 | 0 | 0 | d | d | d | d | 1 | 直接地址3 bit | | |

## XOR 异或

**语法:** XOR dest, src

**操作:** dest <- dest (XOR) src

**描述:** 源操作数和目的操作数按位异或.操作后,源操作数不变

**尺寸:** 字节-字节, 字-字

标志更新**:** N, Z

**XOR Rd, Rs**

字节数: 2

时钟数: 3

操作: (Rd) <-- (Rd) (XOR) (Rs)

译码

| 0 | 1 | 1 | 1 | SZ | 0 | 0 | 1 | d | d | d | d | 0 | s | s | s |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

**XOR Rd, [Rs]**

字节数: 2

时钟数: 4

操作: (Rd) <-- (Rd) (XOR) ((WS:Rs))

译码

| 0 | 1 | 1 | 1 | SZ | 0 | 1 | 0 | d | d | d | d | 0 | s | s | s |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

XOR [Rd], Rs

字节数: 2

时钟数: 4

操作: ((WS:Rd)) <-- ((WS:Rd)) (XOR) (Rs)

译码

| 0 | 1 | 1 | 1 | SZ | 0 | 1 | 0 | s | s | s | s | 1 | d | d | d |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

**XOR Rd, [Rs+offset8]**

字节数: 3

时钟数: 6

操作: (Rd) <-- (Rd) (XOR) ((WS:Rs)+offset8)

译码

| 0 | 1 | 1 | 1 | SZ | 1 | 0 | 0 | d | d | d | d | 0 | s | s | s |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit偏移

**XOR [Rd+offset8], Rs**

字节数: 3

时钟数: 6

操作: ((WS:Rd)+offset8) <-- ((WS:Rd)+offset8) (XOR) (Rs)

译码

| 0 | 1 | 1 | 1 | SZ | 1 | 0 | 0 | s | s | s | s | 1 | d | d | d |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit偏移

**XOR Rd, [Rs+offset16]**

字节数: 4

时钟数: 6

操作: (Rd) <-- (Rd) (XOR) ((WS:Rs)+offset16)

译码

| 0 | 1 | 1 | 1 | SZ | 1 | 0 | 1 | d | d | d | d | 0 | s | s | s |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit偏移高8-bit

字节4:16-bit偏移低8-bit

**XOR [Rd+offset16], Rs**

字节数: 4

时钟数: 6

操作: ((WS:Rd)+offset16) <-- ((WS:Rd)+offset16) (XOR) (Rs)

译码

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | SZ | 1 | 0 | 1 | s | s | s | s | 1 | d | d | d |

字节3:16-bit偏移量高8-bit

字节4:16-bit偏移量低8-bit

**XOR Rd, [Rs+]**

字节数: 2

时钟数: 5

操作: (Rd) <-- (Rd) (XOR) ((WS:Rs))

译码

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | SZ | 0 | 1 | 1 | d | d | d | d | 0 | s | s | s |

**XOR [Rd+], Rs**

字节数: 2

时钟数: 5

操作: ((WS:Rd)) <-- ((WS:Rd)) (XOR) (Rs)

译码

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | SZ | 0 | 1 | 1 | s | s | s | s | 1 | d | d | d |

XOR direct, Rs

字节数: 3

时钟数: 4

操作: (direct) <-- (direct) (XOR) (Rs)

译码

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | SZ | 1 | 1 | 0 | s | s | s | s | 1 | 直接地址3 bit | | |

字节3:直接地址低8 bit.

**XOR Rd, direct**

字节数: 3

时钟数: 4

操作: (Rd) <-- (Rd) (XOR) (direct)

译码

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | SZ | 1 | 1 | 0 | d | d | d | d | 0 | 直接地址3 bit | | |

字节3:直接地址低8 bit.

**XOR Rd, #data8**

字节数: 3

时钟数:3

操作: (Rd) <-- (Rd) (XOR) #data8

译码

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | d | d | d | d | 0 | 1 | 1 | 1 |

字节3:8-bit立即数

**XOR Rd, #data16**

字节数: 4

时钟数: 3

操作: (Rd) <-- (Rd) (XOR) #data16

译码

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | d | d | d | d | 0 | 1 | 1 | 1 |

字节3:16-bit立即数的高8-bit
字节4:16-bit立即数的低8-bit

### XOR [Rd], #data8

字节数: 3
时钟数: 4
操作:  ((WS:Rd)) <-- ((WS:Rd)) (XOR) #data8
译码

| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | d | d | d | 0 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit立即数

### XOR [Rd], #data16

字节数: 4
时钟数: 4
操作:  ((WS:Rd)) <-- ((WS:Rd)) (XOR) #data16
译码

| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | d | d | d | 0 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit立即数的高8bit
字节4:16-bit立即数的低8bit

### XOR [Rd+], #data8

字节数: 3
时钟数: 5
操作:  ((WS:Rd)) <-- ((WS:Rd)) (XOR) #data8
(Rd) <-- (Rd) + 1
译码

| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | d | d | d | 0 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit立即数

### XOR [Rd+], #data16

字节数: 4
时钟数: 5
操作:  ((WS:Rd)) <-- ((WS:Rd)) (XOR) #data16
(Rd) <-- (Rd) + 2
译码

| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | d | d | d | 0 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit立即数的高8 bit
字节4:16-bit立即数的低8 bit

### XOR [Rd+offset8], #data8

字节数: 4
时钟数: 6
操作:  ((WS:Rd)+offset8) <-- ((WS:Rd)+offset8) (XOR) #data8
译码

| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | d | d | d | 0 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit偏移
字节4:8-bit立即数

### XOR [Rd+offset8], #data16

字节数: 5

时钟数: 6

操作:  ((WS:Rd)+offset8) <-- ((WS:Rd)+offset8) (XOR) #data16

译码

| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | d | d | d | 0 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:8-bit偏移

字节4:16-bit立即数的高8 bit

字节5:16-bit立即数的低8 bit

### XOR [Rd+offset16], #data8

字节数: 5

时钟数: 6

操作:  ((WS:Rd)+offset16) <-- ((WS:Rd)+offset16) (XOR) #data8

译码

| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | d | d | d | 0 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit偏移的高8 bit

字节4:16-bit偏移的低8 bit

字节5:8-bit立即数

### XOR [Rd+offset16], #data16

字节数: 6

时钟数: 6

操作:  ((WS:Rd)+offset16) <-- ((WS:Rd)+offset16) (XOR) #data16

译码

| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | d | d | d | 0 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:16-bit偏移的高8 bit

字节4:16-bit偏移的低8 bit

字节5:16-bit立即数的高8 bit

字节4:16-bit立即数的低8 bit

### XOR direct, #data8

字节数: 4

时钟数: 4

操作:  (direct) <-- (direct) (XOR) #data8

译码

| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 3bit的直接地址 | 0 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:直接地址的低8 bit

字节4:8-bit立即数

### XOR direct, #data16

字节数: 5

时钟数: 4

操作:  (direct) <-- (direct) (XOR) #data16

译码

| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 3bit的直接地址 | 0 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

字节3:直接地址的低8 bit

字节4:16-bit立即数的高8bit

字节5:16-bit立即数的低8bit

## 6.6 XA非法操作数组合

如果出现以下情况,指令为非法组合.

(1)在一个指令中指定或暗示2个单寄存器的写操作.

(2)源寄存器在自动增量读以前数据可能遭到破坏

这些情况XA的硬件检测不到.在写XA代码时,不要使用这些指令/操作数组合.

| 非法指令 | 非法组合原因 |
|---|---|
| (any op) Rx, [Rx+] | 自动增加和外部写(1) |
| mov [Rx+], [Rx+] | 一个寄存器的双增加.(2) |
| (any op) [Rx+], Rx | 自动增加写可能在源寄存器读以前而被破坏(3). |
| NORM Rx, Rx | 结果和移位记数存储在同一个寄存器中.(4) |
| XCH Rx, Rx | 对一个寄存器两次写.(4) |
| (any op) [Rx+], Ry | 对同一寄存器施加自动增加和间接地址写.(5) |
| (any op) [Rx+], [Ry+] | 对同一寄存器施加自动增加和间接地址写.(5) |
| (any op) [Rx+], #data | 对同一寄存器施加自动增加和间接地址写.(5) |
| XCH Rx, [Rx] | 对同一寄存器施加间接地址写和外部写.(6) |
| XCH Rx, direct | 对同一寄存器施加直接地址写和外部写.(7) |
| POP R7 | 对R7/SP施加堆栈指针自动增加和外部写.(8) |

注:

1 源寄存器和目的寄存器相同的寻址方式是非法的.这将对同一寄存器施加数据读和自动增加操作.

2 这个指令是非法的,因为源和目的指针寄存器是相同的寄存器.这将导致对一个寄存器施加两个自动增加操作.

3 这个指令是非法的,因为源和目的寄存器是相同的寄存器.源寄存器在同时会自动增加和读,将出现不确定的结果.

4 这个指令是非法的,因为源和目的寄存器是相同的寄存器.这将导致对同一寄存器施加两次写.

5 这个寻址方式是非法的,因为目的间接地址指向了指针寄存器本身.仅在80C51兼容模式启动时才是可能的.这将在同一寄存器上施加自动增加和数据写.

6 这个寻址方式是非法的,因为源间接地址指向了目的寄存器本身.仅在80C51兼容模式启动时才是可能的.这将在同一寄存器上施加两次数据写.

7 这个寻址方式是非法的,因为源间接地址指向了目的寄存器本身.仅在80C51兼容模式启动时才是可能的.这将在同一寄存器上施加两次数据写.

8 一个POP到R7(堆栈指针)将导致在相同寄存器上施加数据读和自动增加操作.