ARM指令集--数据操作指令

SOFTEEM 软帝

- ARM指令的种类:

（1） 数据处理指令；

（2） 程序状态寄存器与通用寄存器之间的传送指令；

（3） **Load／Store**指令；

（4） 转移指令；

（5） 异常中断指令；

（6） 协处理器指令。

3.3.1 ARM数据处理指令

数据处理指令大致可分为6类：

（1） 数据传送指令；

（2） 算术运算指令；

（3） 逻辑运算指令；

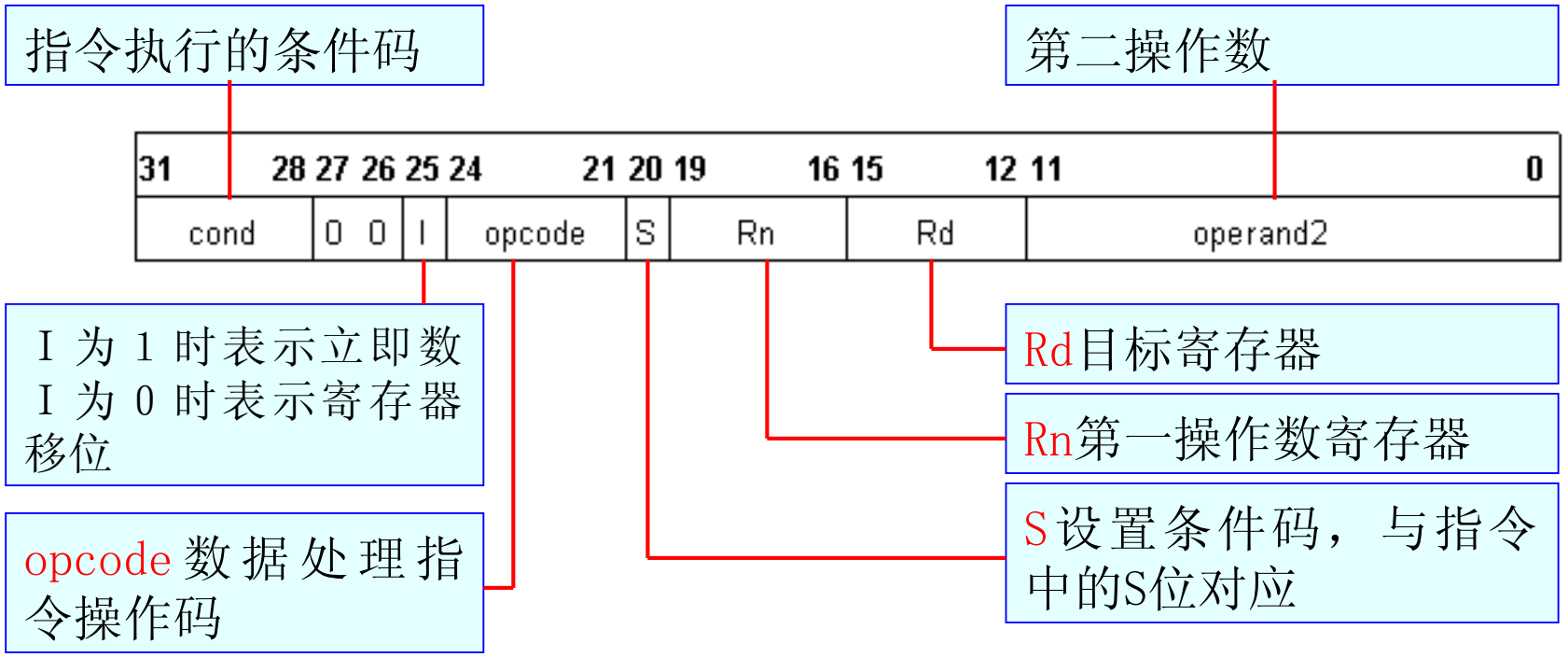
（4） 比较指令；

（5） 测试指令；

（6） 乘法指令。

数据处理指令只能对寄存器的内容进行操作，而不能对内存中的数据进行操作。所有ARM数据处理指令均可选择使用S后缀，并影响状态标志。

- ARM数据处理指令——指令编码



opcode操作码功能表

操作码	指令助记符	说明
0000	AND	逻辑与操作指令
0001	EOR	逻辑异或操作指令
0010	SUB	减法运算指令
0011	RSB	逆向减法指令
0100	ADD	加法运算指令
0101	ADC	带进位加法
0110	SBC	带进位减法指令
0111	RSC	带进位逆向减法指令
1000	TST	位测试指令
1001	TEQ	相等测试指令
1010	CMP	比较指令
1011	CMN	负数比较指令
1100	ORR	逻辑或操作指令
1101	MOV	数据传送
1110	BIC	位清除指令
1111	MVN	数据非传送

3.3 ARM指令的使用

助记符	说明	操作	条件码位置
MOV Rd,operand2	数据传送	Rd←operand2	MOV{cond}{S}
MVN Rd,operand2	数据非传送	Rd←(~operand2)	MVN{cond}{S}

SOFTEEM 软帝

- ARM数据处理指令——数据传送

助记符	说明	操作	条件码位置
MOV Rd,operand2	数据传送	Rd←operand2	MOV{cond}{S}
MVN Rd,operand2	数据非传送	Rd←(~operand2)	MVN{cond}{S}

MOV指令将8位图立即数或寄存器传送到目标寄存器（Rd），可用于移位运算等操作。指令格式如下：

```
MOV{cond}{S}    Rd,operand2
```

MOV指令举例如下：

```
MOV     R1,#0x10            ;R1=0x10
MOV     R0,R1               ;R0=R1
MOVS    R3,R1,LSL #2        ;R3=R1<<2，并影响标志位
MOV     PC,LR               ;PC=LR，子程序返回
```

- ARM数据处理指令——数据传送

助记符	说明	操作	条件码位置
MOV Rd,operand2	数据传送	Rd←operand2	MOV{cond}{S}
MVN Rd,operand2	数据非传送	Rd←(~operand2)	MVN{cond}{S}

MVN指令将8位图立即数或寄存器（operand2）按位取反后传送到目标寄存器（Rd），因为其具有取反功能，所以可以装载范围更广的立即数。指令格式如下：

```
MVN{cond}{S}    Rd,operand2
```

MVN指令举例如下：

```
MVN     R1,#0xFF            ;R1=0xFFFFFF00
MVN     R1,R2               ;将R2取反，结果存到R1
错例:MVN    R15,R3,ASR R0   ;R15不允许与被控制移位的寄
                            ;存器一起出现
```

SOFTEEM 软帝

- ARM数据处理指令——算术运算

助记符	说明	操作	条件码位置
ADD Rd, Rn, operand2	加法运算指令	Rd←Rn+operand2	ADD{cond}{S}
SUB Rd, Rn, operand2	减法运算指令	Rd←Rn-operand2	SUB{cond}{S}
RSB Rd, Rn, operand2	逆向减法指令	Rd←operand2-Rn	RSB{cond}{S}
ADC Rd, Rn, operand2	带进位加法	Rd←Rn+operand2+Carry	ADC{cond}{S}
SBC Rd, Rn, operand2	带进位减法指令	Rd←Rn-operand2-(NOT)Carry	SBC{cond}{S}
RSC Rd, Rn, operand2	带进位逆向减法指令	Rd←operand2-Rn-(NOT)Carry	RSC{cond}{S}

3.3 ARM指令的使用

• ARM数据处理指令——算术运算

助记符	说明	操作	条件码位置
ADD Rd, Rn, operand2	加法运算指令	Rd←Rn+operand2	ADD{cond}{S}
SUB Rd, Rn, operand2	减法运算指令	Rd←Rn-operand2	SUB{cond}{S}
RSB Rd, Rn, operand2	逆向减法指令	Rd←operand2-Rn	RSB{cond}{S}
ADC Rd, Rn, operand2	带进位加法	Rd←Rn+operand2+Carry	ADC{cond}{S}
SBC Rd, Rn, operand2	带进位减法指令	Rd←Rn-operand2-(NOT)Carry	SBC{cond}{S}
RSC Rd, Rn, operand2	带进位逆向减法指令	Rd←operand2-Rn-(NOT)Carry	RSC{cond}{S}

加法运算指令—ADD指令将operand2的值与Rn的值相加，结果保存到Rd寄存器。指令格式如下：ADD{cond}{S}　　Rd,Rn,operand2

应用示例：

```
ADDS    R1,R1,#1        ;R1=R1+1，并影响标志位
ADD     R1,R1,R2        ;R1=R1+R2
```

SOFTEEM 软帝

- ARM数据处理指令——算术运算

助记符	说明	操作	条件码位置
ADD Rd, Rn, operand2	加法运算指令	Rd←Rn+operand2	ADD{cond}{S}
SUB Rd, Rn, operand2	减法运算指令	Rd←Rn-operand2	SUB{cond}{S}
RSB Rd, Rn, operand2	逆向减法指令	Rd←operand2-Rn	RSB{cond}{S}
ADC Rd, Rn, operand2	带进位加法	Rd←Rn+operand2+Carry	ADC{cond}{S}
SBC Rd, Rn, operand2	带进位减法指令	Rd←Rn-operand2-(NOT)Carry	SBC{cond}{S}
RSC Rd, Rn, operand2	带进位逆向减法指令	Rd←operand2-Rn-(NOT)Carry	RSC{cond}{S}

减法运算指令—SUB指令用寄存器Rn减去operand2，结果保存到Rd中。指令格式如下：SUB{cond}{S} Rd,Rn,operand2

应用示例：
```
SUBS    R0,R0,#1      ;R0=R0-1 ，并影响标志位
SUBS    R2,R1,R2      ;R2=R1-R2 ，并影响标志位

SUB     R6,R7,#0X10; R6=R7-0X10
```

SOFTEEM 软帝

- ARM数据处理指令——算术运算

助记符	说明	操作	条件码位置
ADD Rd, Rn, operand2	加法运算指令	Rd←Rn+operand2	ADD{cond}{S}
SUB Rd, Rn, operand2	减法运算指令	Rd←Rn-operand2	SUB{cond}{S}
RSB Rd, Rn, operand2	逆向减法指令	Rd←operand2-Rn	RSB{cond}{S}
ADC Rd, Rn, operand2	带进位加法	Rd←Rn+operand2+Carry	ADC{cond}{S}
SBC Rd, Rn, operand2	带进位减法指令	Rd←Rn-operand2-(NOT)Carry	SBC{cond}{S}
RSC Rd, Rn, operand2	带进位逆向减法指令	Rd←operand2-Rn-(NOT)Carry	RSC{cond}{S}

逆向减法运算指令—RSB指令将operand2的值减去Rn，结果保存到Rd中。指令格式如下： RSB{cond}{S} Rd,Rn,operand2

应用示例：
```
RSB   R3,R1,#0xFF00 ;R3=0xFF00-R1
RSBS  R1,R2,R2,LSL #2      ;R1=(R2<<2)-R2=R2×3
RSB   R0,R1,#0 ;R0=-R1
```

SOFTEEM 软帝

- ARM数据处理指令——算术运算

助记符	说明	操作	条件码位置
ADD Rd, Rn, operand2	加法运算指令	Rd←Rn+operand2	ADD{cond}{S}
SUB Rd, Rn, operand2	减法运算指令	Rd←Rn-operand2	SUB{cond}{S}
RSB Rd, Rn, operand2	逆向减法指令	Rd←operand2-Rn	RSB{cond}{S}
ADC Rd, Rn, operand2	带进位加法	Rd←Rn+operand2+Carry	ADC{cond}{S}
SBC Rd, Rn, operand2	带进位减法指令	Rd←Rn-operand2-(NOT)Carry	SBC{cond}{S}
RSC Rd, Rn, operand2	带进位逆向减法指令	Rd←operand2-Rn-(NOT)Carry	RSC{cond}{S}

带进位加法指令—ADC将operand2的值与Rn的值相加，再加上CPSR中的C条件标志位，结果保存到Rd寄存器。指令格式如下：

```
ADC{cond}{S}    Rd,Rn,operand2
```

应用示例（使用ADC实现64位加法，结果存于R1、R0中）：

```
ADDS    R0,R0,R2        ;R0等于低32位相加，并影响标志位
ADC     R1,R1,R3        ;R1等于高32位相加，并加上低位进位
```

SOFTEEM 软帝

- ARM数据处理指令——算术运算

助记符	说明	操作	条件码位置
ADD Rd, Rn, operand2	加法运算指令	Rd←Rn+operand2	ADD{cond}{S}
SUB Rd, Rn, operand2	减法运算指令	Rd←Rn-operand2	SUB{cond}{S}
RSB Rd, Rn, operand2	逆向减法指令	Rd←operand2-Rn	RSB{cond}{S}
ADC Rd, Rn, operand2	带进位加法	Rd←Rn+operand2+Carry	ADC{cond}{S}
SBC Rd, Rn, operand2	带进位减法指令	Rd←Rn-operand2-(NOT)Carry	SBC{cond}{S}
RSC Rd, Rn, operand2	带进位逆向减法指令	Rd←operand2-Rn-(NOT)Carry	RSC{cond}{S}

带进位减法指令—SBC用寄存器Rn减去operand2，再减去CPSR中的C条件标志位的非(即若C标志清零，则结果减去1)，结果保存到Rd中。指令格式如下： SBC{cond}{S} Rd,Rn,operand2

应用示例（使用SBC实现64位减法，结果存于R1、R0中）：

```
SUBS    R0,R0,R2    ; 低32位相减，并影响标志位
SBC     R1,R1,R3    ;高32位相减，并减去低位借位
```

SOFTEEM 软帝

- ARM数据处理指令——算术运算

助记符	说明	操作	条件码位置
ADD Rd, Rn, operand2	加法运算指令	Rd←Rn+operand2	ADD{cond}{S}
SUB Rd, Rn, operand2	减法运算指令	Rd←Rn-operand2	SUB{cond}{S}
RSB Rd, Rn, operand2	逆向减法指令	Rd←operand2-Rn	RSB{cond}{S}
ADC Rd, Rn, operand2	带进位加法	Rd←Rn+operand2+Carry	ADC{cond}{S}
SBC Rd, Rn, operand2	带进位减法指令	Rd←Rn-operand2-(NOT)Carry	SBC{cond}{S}
RSC Rd, Rn, operand2	带进位逆向减法指令	Rd←operand2-Rn-(NOT)Carry	RSC{cond}{S}

带进位逆向减法指令—RSC指令用寄存器operand2减去Rn，再减去CPSR中的C条件标志位，结果保存到Rd中。指令格式如下：

```
RSC{cond}{S}    Rd,Rn,operand2
```

应用示例（使用RSC指令实现求64位数值的负数 ）：

```
RSBS    R2,R0,#0
RSC     R3,R1,#0
```

SOFTEEM 软帝

- ARM数据处理指令——算术运算

助记符	说明	操作	条件码位置
ADD Rd, Rn, operand2	加法运算指令	Rd←Rn+operand2	ADD{cond}{S}
SUB Rd, Rn, operand2	减法运算指令	Rd←Rn-operand2	SUB{cond}{S}
RSB Rd, Rn, operand2	逆向减法指令	Rd←operand2-Rn	RSB{cond}{S}
ADC Rd, Rn, operand2	带进位加法	Rd←Rn+operand2+Carry	ADC{cond}{S}
SBC Rd, Rn, operand2	带进位减法指令	Rd←Rn-operand2-(NOT)Carry	SBC{cond}{S}
RSC Rd, Rn, operand2	带进位逆向减法指令	Rd←operand2-Rn-(NOT)Carry	RSC{cond}{S}

例子:

```
RSCLES R0,R5,R0,LSL R4   ;有条件执行,设置标志
```

错例:

```
RSCLES R0,R15,R0,LSL R4 ;R15不允许与被控制移位的寄存器一起出现
```

- ARM数据处理指令——逻辑运算指令

助记符	说明	操作	条件码位置
AND Rd, Rn, operand2	逻辑与操作指令	Rd←Rn & operand2	AND{cond}{S}
ORR Rd, Rn, operand2	逻辑或操作指令	Rd←Rn | operand2	ORR{cond}{S}
EOR Rd, Rn, operand2	逻辑异或操作指令	Rd←Rn ^ operand2	EOR{cond}{S}
BIC Rd, Rn, operand2	位清除指令	Rd←Rn & (~operand2)	BIC{cond}{S}

SOFTEEM 软帝

- ARM数据处理指令——逻辑运算指令

助记符	说明	操作	条件码位置
AND Rd, Rn, operand2	逻辑与操作指令	Rd←Rn & operand2	AND{cond}{S}
ORR Rd, Rn, operand2	逻辑或操作指令	Rd←Rn | operand2	ORR{cond}{S}
EOR Rd, Rn, operand2	逻辑异或操作指令	Rd←Rn ^ operand2	EOR{cond}{S}
BIC Rd, Rn, operand2	位清除指令	Rd←Rn & (~operand2)	BIC{cond}{S}

逻辑与操作指令——AND指令将operand2的值与寄存器Rn的值按位作逻辑“与”操作，结果保存到Rd中。指令格式如下：

```
AND{cond}{S}    Rd,Rn,operand2
```

应用示例：

```
ANDS    R0,R0,#0x01  ;R0=R0 & 0x01，取出最低位数据
AND     R2,R1,R3     ;R2=R1 & R3
```

SOFTEEM 软帝

- ARM数据处理指令——逻辑运算指令

助记符	说明	操作	条件码位置
AND Rd, Rn, operand2	逻辑与操作指令	Rd←Rn & operand2	AND{cond}{S}
ORR Rd, Rn, operand2	逻辑或操作指令	Rd←Rn | operand2	ORR{cond}{S}
EOR Rd, Rn, operand2	逻辑异或操作指令	Rd←Rn ^ operand2	EOR{cond}{S}
BIC Rd, Rn, operand2	位清除指令	Rd←Rn & (~operand2)	BIC{cond}{S}

逻辑或操作指令—ORR指令将operand2的值与寄存器Rn的值按位作逻辑“或”操作，结果保存到Rd中。指令格式如下：

```
ORR{cond}{S}    Rd,Rn, operand2
```

应用示例：

```
ORR      R0,R0,#0x0F                ;将R0的低4位置1
MOV    R1,R2,LSR #24
ORR R3,R1,R3,LSL #8;使用ORR指令将R2的高8位;数据移入到R3低8位中
```

- ARM数据处理指令——逻辑运算指令

助记符	说明	操作	条件码位置
EOR Rd, Rn, operand2	逻辑异或操作指令	Rd←Rn ^ operand2	EOR{cond}{S}

逻辑异或操作指令——EOR指令将operand2的值与寄存器Rn的值按位作逻辑“异或”操作，结果保存到Rd中。指令格式如下：

```
EOR{cond}{S}    Rd,Rn, operand2
```

应用示例：

```
EOR     R1,R1,#0x0F          ;将R1的低4位异或
EOR     R2,R1,R0             ;R2=R1^R0
EORS    R0,R5,#0x01          ; 将R5和0x01进行逻辑异或，
                             ;结果保存到R0，并影响标志位
```

错例：

```
EORS R0,R15,R3,ROR R6 ; R15不允许与被控制移位的寄存器一起出现
```

• ARM数据处理指令——逻辑运算指令

助记符	说明	操作	条件码位置
AND Rd, Rn, operand2	逻辑与操作指令	Rd←Rn & operand2	AND{cond}{S}
ORR Rd, Rn, operand2	逻辑或操作指令	Rd←Rn | operand2	ORR{cond}{S}
EOR Rd, Rn, operand2	逻辑异或操作指令	Rd←Rn ^ operand2	EOR{cond}{S}
BIC Rd, Rn, operand2	位清除指令	Rd←Rn & (~operand2)	BIC{cond}{S}

位清除指令——BIC指令将寄存器Rn的值与operand2的值的反码按位作逻辑“与”操作，结果保存到Rd中。指令格式如下：

```
BIC{cond}{S}    Rd,Rn, operand2
```

应用示例：

```
BIC     R1,R1,#0x0F          ;将R1的低4位清零，其它位不变
BIC     R1,R2,R3             ;将R3的反码和R2相逻辑“与”，
                             ;结果保存到R1中
```

3.3 ARM指令的使用

- ARM数据处理指令——比较指令

助记符	说明	操作	条件码位置
CMP Rn, operand2	比较指令	标志N、Z、C、V←Rn-operand2	CMP{cond}
CMN Rn, operand2	负数比较指令	标志N、Z、C、V←Rn+operand2	CMN{cond}
TST Rn, operand2	位测试指令	标志N、Z、C、V←Rn & operand2	TST{cond}
TEQ Rn, operand2	相等测试指令	标志N、Z、C、V←Rn ^ operand2	TEQ{cond}

- ARM数据处理指令——比较指令

助记符	说明	操作	条件码位置
CMP Rn, operand2	比较指令	标志N、Z、C、V←Rn-operand2	CMP{cond}

比较指令——CMP指令将寄存器Rn的值减去operand2的值，根据操作的结果更新CPSR中的相应条件标志位，以便后面的指令根据相应的条件标志来判断是否执行。指令格式如下： CMP{cond} Rn, operand2

应用示例：

```
CMP     R1,#10          ; R1与10比较，设置相关标志位
CMP     R1,R2           ; R1与R2比较，设置相关标志位
```

注意：CMP指令与SUBS指令的区别在于CMP指令不保存运算结果。在进行两个数据的大小判断时，常用CMP指令及相应的条件码来操作。

错例： CMP R2,R15,ASR R0 ;R15不允许与被控制移位的寄存器一起出现.在有寄存器控制移位的任何数据处理指令中,不能将R15用于任何操作数

- ARM数据处理指令——比较指令

助记符	说明	操作	条件码位置
CMN Rn, operand2	负数比较指令	标志N、Z、C、V←Rn+operand2	CMN{cond}

负数比较指令——CMN指令使用寄存器Rn的值加上operand2的值，根据操作的结果更新CPSR中的相应条件标志位，以便后面的指令根据相应的条件标志来判断是否执行。指令格式如下：

```
CMN{cond}   Rn, operand2
```

应用示例：

```
CMN     R0,#1          ; R0+1，判断R0是否为1的补码
                       ;如果是，则设置Z标志位
```

注意：CMN指令与ADDS指令的区别在于CMN指令不保存运算结果。CMN指令可用于负数比较，比如CMN R0,#1指令则表示R0与-1比较，若R0为-1(即1的补码)，则Z置位；否则Z复位。

• ARM数据处理指令——比较指令

助记符	说明	操作	条件码位置
TST Rn, operand2	位测试指令	标志N、Z、C、V←Rn & operand2	TST{cond}

位测试指令—TST指令将寄存器Rn的值与operand2的值按位作逻辑“与”操作，根据操作的结果更新CPSR中的相应条件标志位，以便后面的指令根据相应的条件标志来判断是否执行。指令格式如下：

```
TST{cond}    Rn, operand2
```

应用示例：

```
TST      R0,#0x01          ; 判断R0的最低位是否为1
TST      R1,#0x0F          ; 判断R1的低4位是否为1
```

注意：TST指令与ANDS指令的区别在于TST指令不保存运算结果。TST指令通常与EQ、NE条件码配合使用，当所有测试位均为0时，EQ有效，而只要有一个测试位不为0，则NE有效。

- ARM数据处理指令——比较指令

助记符	说明	操作	条件码位置
TEQ Rn, operand2	相等测试指令	标志N、Z、C、V←Rn ^ operand2	TEQ{cond}

相等测试指令—TEQ指令将寄存器Rn的值与operand2的值按位作逻辑“异或”操作，根据操作的结果更新CPSR中的相应条件标志位，以便后面的指令根据相应的条件标志来判断是否执行。指令格式如下：

```
TEQ{cond}    Rn, operand2
```

应用示例：

```
TEQ      R0,R1          ; 比较R0与R1是否相等（不影响V位和C位）
```

注意：TEQ指令与EORS指令的区别在于TEQ指令不保存运算结果。使用TEQ进行相等测试时，常与EQ、NE条件码配合使用。当两个数据相等时，EQ有效；否则NE有效。

- ARM数据处理指令——比较指令

助记符	说明	操作	条件码位置
TST Rn, operand2	位测试指令	标志N、Z、C、V←Rn & operand2	TST{cond}
TEQ Rn, operand2	相等测试指令	标志N、Z、C、V←Rn ^ operand2	TEQ{cond}

例子:

```
TST     R0 ,#0x3F8
TEQEQ   R10,R9
TSTNE   R1 ,R5,ASR R1
```

错例:

```
TEQ     R15,R1,ROR R0   ;R15不允许与被控制移位的寄存器一
                        ;起出现
```

3.3 ARM指令的使用

- ARM指令集——乘法指令(Multiplication)

ARM7TDMI具有三种乘法指令，分别为：

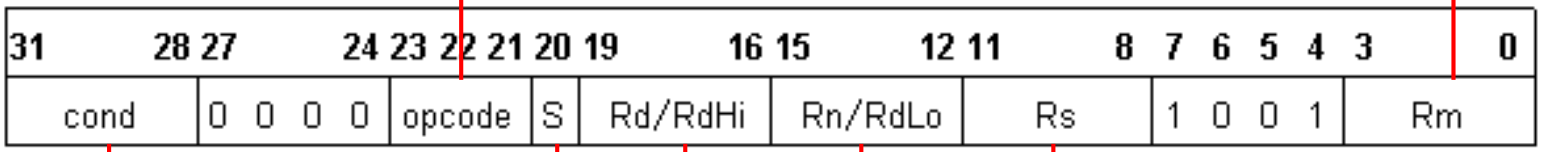
- 32×32位乘法指令；
- 32× 32位乘加指令；
- 32× 32位结果为64位的乘/乘加指令。

- ARM指令——乘法指令

•乘法指令编码

Opcode乘法指令操作码

Rm为被乘数寄存器



指令执行的条件码

S设置条件码，与指令中的S位对应

Rn/RdHi为目标寄存器或64位乘法指令的目标寄存器（高32位）

Rs为乘数寄存器

Rd/RdLo为MLA指令相加的寄存器或64位乘法指令的目标寄存器（低32位）

opcode操作码功能表

操作码	指令助记符	说明
000	MUL	32位乘法指令
001	MLA	32位乘加指令
100	UMULL	64位无符号乘法指令
101	UMLAL	64位无符号乘加指令
110	SMULL	64位有符号乘法指令
111	SMLAL	64位有符号乘加指令

- ARM指令——乘法指令

助记符	说明	操作	条件码位置
MUL Rd,Rm,Rs	32位乘法指令	Rd←Rm*Rs (Rd≠Rm)	MUL{cond}{S}
MLA Rd,Rm,Rs,Rn	32位乘加指令	Rd←Rm*Rs+Rn (Rd≠Rm)	MLA{cond}{S}
UMULL RdLo,RdHi,Rm,Rs	64位无符号乘法指令	(RdLo,RdHi) ←Rm*Rs	UMULL{cond}{S}
UMLAL RdLo,RdHi,Rm,Rs	64位无符号乘加指令	(RdLo,RdHi) ←Rm*Rs+(RdLo,RdHi)	UMLAL{cond}{S}
SMULL RdLo,RdHi,Rm,Rs	64位有符号乘法指令	(RdLo,RdHi) ←Rm*Rs	SMULL{cond}{S}
SMLAL RdLo,RdHi,Rm,Rs	64位有符号乘加指令	(RdLo,RdHi) ←Rm*Rs+(RdLo,RdHi)	SMLAL{cond}{S}

3.3 ARM指令的使用

- ARM指令——乘法指令

助记符	说明	操作	条件码位置
MUL Rd,Rm,Rs	32位乘法指令	Rd←Rm*Rs (Rd≠Rm)	MUL{cond}{S}
MLA Rd,Rm,Rs,Rn	32位乘加指令	Rd←Rm*Rs+Rn (Rd≠Rm)	MLA{cond}{S}

32位乘法指令—MUL指令将Rm和Rs中的值相乘，结果的低32位保存到Rd中。

32位乘加指令—MLA指令将Rm和Rs中的值相乘，再将乘积加上第3个操作数Rn ，结果的低32位保存到Rd中。

指令格式如下： MUL{cond}{S} Rd,Rm,Rs

MLA{cond}{S} Rd,Rm,Rs,Rn

注意:R15不能用做Rd,Rm,Rs,Rn. Rd不能与Rm相同.

应用示例：

```
MUL   R1,R2,R3      ;R1=R2×R3
MULS  R0,R3,R7      ;R0=R3×R7，同时影响CPSR中的N位和Z位
MLA   R1,R2,R3,R0 ;R1=R2×R3+R0
```

错例:

```
MUL   R15,R0,R3     ;不允许使用R15
MLA   R1 ,R1,R6     ;Rd不能与Rm相同
```

- ARM指令——乘法指令

助记符	说明	操作	条件码位置
MUL Rd, Rm, Rs	32位乘法指令	Rd←Rm*Rs (Rd≠Rm)	MUL{cond}{S}
MLA Rd, Rm, Rs, Rn	32位乘加指令	Rd←Rm*Rs+Rn (Rd≠Rm)	MLA{cond}{S}
UMULL RdLo, RdHi, Rm, Rs	64位无符号乘法指令	(RdLo, RdHi) ←Rm*Rs	UMULL{cond}{S}
UMLAL RdLo, RdHi, Rm, Rs	64位无符号乘加指令	(RdLo, RdHi) ←Rm*Rs+(RdLo, RdHi)	UMLAL{cond}{S}
SMULL RdLo, RdHi, Rm, Rs	64位有符号乘法指令	(RdLo, RdHi) ←Rm*Rs	SMULL{cond}{S}
SMLAL RdLo, RdHi, Rm, Rs	64位有符号乘加指令	(RdLo, RdHi) ←Rm*Rs+(RdLo, RdHi)	SMLAL{cond}{S}

64位无符号乘法指令—UMULL指令将Rm和Rs中的值作无符号数相乘，结果的低32位保存到RdLo中，而高32位保存到RdHi中。指令格式如下：

```
UMULL{cond}{S}    RdLo,RdHi,Rm,Rs
```

应用示例：

```
UMULL    R0,R1,R5,R8        ; (R1、R0)=R5×R8
```

- ARM指令——乘法指令

助记符	说明	操作	条件码位置
MUL Rd, Rm, Rs	32位乘法指令	Rd←Rm*Rs (Rd≠Rm)	MUL{cond}{S}
MLA Rd, Rm, Rs, Rn	32位乘加指令	Rd←Rm*Rs+Rn (Rd≠Rm)	MLA{cond}{S}
UMULL RdLo, RdHi, Rm, Rs	64位无符号乘法指令	(RdLo, RdHi) ←Rm*Rs	UMULL{cond}{S}
UMLAL RdLo, RdHi, Rm, Rs	64位无符号乘加指令	(RdLo, RdHi) ←Rm*Rs+(RdLo, RdHi)	UMLAL{cond}{S}
SMULL RdLo, RdHi, Rm, Rs	64位有符号乘法指令	(RdLo, RdHi) ←Rm*Rs	SMULL{cond}{S}
SMLAL RdLo, RdHi, Rm, Rs	64位有符号乘加指令	(RdLo, RdHi) ←Rm*Rs+(RdLo, RdHi)	SMLAL{cond}{S}

64位无符号乘加指令—UMLAL指令将Rm和Rs中的值作无符号数相乘，64位乘积与RdHi、RdLo相加，结果的低32位保存到RdLo中，而高32位保存到RdHi中。指令格式如下：

```
UMLAL{cond}{S}    RdLo,RdHi,Rm,Rs
```

应用示例：

```
UMLAL    R0,R1,R5,R8        ;(R1、R0)=R5×R8+(R1、R0)
```

- ARM指令——乘法指令

助记符	说明	操作	条件码位置
MUL Rd,Rm,Rs	32位乘法指令	Rd←Rm*Rs (Rd≠Rm)	MUL{cond}{S}
MLA Rd,Rm,Rs,Rn	32位乘加指令	Rd←Rm*Rs+Rn (Rd≠Rm)	MLA{cond}{S}
UMULL RdLo,RdHi,Rm,Rs	64位无符号乘法指令	(RdLo,RdHi) ←Rm*Rs	UMULL{cond}{S}
UMLAL RdLo,RdHi,Rm,Rs	64位无符号乘加指令	(RdLo,RdHi) ←Rm*Rs+(RdLo,RdHi)	UMLAL{cond}{S}
SMULL RdLo,RdHi,Rm,Rs	64位有符号乘法指令	(RdLo,RdHi) ←Rm*Rs	SMULL{cond}{S}
SMLAL RdLo,RdHi,Rm,Rs	64位有符号乘加指令	(RdLo,RdHi) ←Rm*Rs+(RdLo,RdHi)	SMLAL{cond}{S}

64位有符号乘法指令—SMULL指令将Rm和Rs中的值作有符号数相乘，结果的低32位保存到RdLo中，而高32位保存到RdHi中。指令格式如下：

```
SMULL{cond}{S}    RdLo,RdHi,Rm,Rs
```

应用示例：

```
SMULL    R2,R3,R7,R6          ; (R3、R2)=R7×R6
```

- ARM指令——乘法指令

助记符	说明	操作	条件码位置
MUL Rd,Rm,Rs	32位乘法指令	Rd←Rm*Rs (Rd≠Rm)	MUL{cond}{S}
MLA Rd,Rm,Rs,Rn	32位乘加指令	Rd←Rm*Rs+Rn (Rd≠Rm)	MLA{cond}{S}
UMULL RdLo,RdHi,Rm,Rs	64位无符号乘法指令	(RdLo,RdHi) ←Rm*Rs	UMULL{cond}{S}
UMLAL RdLo,RdHi,Rm,Rs	64位无符号乘加指令	(RdLo,RdHi) ←Rm*Rs+(RdLo,RdHi)	UMLAL{cond}{S}
SMULL RdLo,RdHi,Rm,Rs	64位有符号乘法指令	(RdLo,RdHi) ←Rm*Rs	SMULL{cond}{S}
SMLAL RdLo,RdHi,Rm,Rs	64位有符号乘加指令	(RdLo,RdHi) ←Rm*Rs+(RdLo,RdHi)	SMLAL{cond}{S}

64位有符号乘加指令——SMLAL指令将Rm和Rs中的值作有符号数相乘，64位乘积与RdHi、RdLo相加，结果的低32位保存到RdLo中，而高32位保存到RdHi中。指令格式如下：

```
SMLAL{cond}{S}    RdLo,RdHi,Rm,Rs
```

应用示例：

```
SMLAL    R2,R3,R7,R6        ; (R3、R2)=R7×R6+(R3、R2)
```

- ARM指令——乘法指令

助记符	说明	操作	条件码位置
UMULL RdLo,RdHi,Rm,Rs	64位无符号乘法指令	(RdLo,RdHi) ←Rm*Rs	UMULL{cond}{S}
UMLAL RdLo,RdHi,Rm,Rs	64位无符号乘加指令	(RdLo,RdHi) ←Rm*Rs+(RdLo,RdHi)	UMLAL{cond}{S}
SMULL RdLo,RdHi,Rm,Rs	64位有符号乘法指令	(RdLo,RdHi) ←Rm*Rs	SMULL{cond}{S}
SMLAL RdLo,RdHi,Rm,Rs	64位有符号乘加指令	(RdLo,RdHi) ←Rm*Rs+(RdLo,RdHi)	SMLAL{cond}{S}

例子:

```
UMULL     R0,R4,R5,R6
UMLALS    R4,R5,R3,R8
SMLALLES  R8,R9,R7,R6
SMULLNE   R0,R1,R9,R0      ;Rs可以与其它寄存器相同
```

错例:

```
UMULL     R1,R15,R10,R2    ;不允许使用R15
SMULLLE   R0,R1,R0,R5      ;RdLo,RdHi,Rm必须是不同寄存器
```

Q & A