《ARM 系列处理器应用技术完全手册》

作者：华清远见

第 7 章 乘法指令

本章目标

ARM 乘法指令完成两个数据的乘法。两个 32 位二进制数相乘的结果是 64 位的积。在有些 ARM 的处理器版本中，将乘积的结果保存到两个独立的寄存器中。另外一些版本只将最低有效 32 位存放到一个寄存器中。

无论是哪种版本的处理器，都有乘-累加的变型指令，将乘积连续累加得到总和。而且有符号数和无符号数都能使用。对于有符号数和无符号数，结果的最低有效位是一样的。因此，对于只保留 32 位结果的乘法指令，不需要区分有符号数和无符号数两种情况。

乘法指令的二进制编码格式如图 7.1 所示。

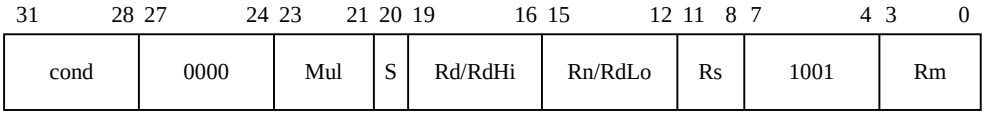

图 7.1　乘法指令的二进制编码

表 7.1 显示了各种形式乘法指令的功能。

表 7.1　　各种形式乘法指令

操作码[23:21]	助记符	意　义	操　作
000	MUL	乘（保留 32 位结果）	Rd：=（Rm×Rs）[31：0]
001	MLA	乘-累加（32 位结果）	Rd：=（Rm×Rs＋Rn）[31：0]
100	UMULL	无符号数长乘	RdHi：RdLo：=Rm×Rs
101	UMLAL	无符号长乘-累加	RdHi：RdLo：＋=Rm×Rs
110	SMULL	有符号数长乘	RdHi：RdLo：=Rm×Rs
111	SMLAL	有符号数长乘-累加	RdHi：RdLo：＋=Rm×Rs

其中：

① “RdHi：RdLo”是由 RdHi（最高有效 32 位）和 RdLo（最低有效 32 位）链接形成的 64 位数，“[31：0]”只选取结果的最低有效 32 位。

② 简单的赋值由“：=”表示。

③ 累加（将右边加到左边）是由“＋=”表示。

同其他数据处理指令一样，位 S 控制条件码的设置。当在指令中设置了位 S 时，则有以下结果。

① 对于产生 32 位结果的指令形式，将标志位 N 设置为 Rd 的第 31 位的值；对于产生长结果的指令形式，将其设置为 RdHi 的第 31 位的值。

② 对于产生 32 位结果的指令形式，如果 Rd 等于零，则标志位 Z 置位；对于产生长结果的指令形式，RdHi 和 RdLo 同时为零时，标志位 Z 置位。

③ 将标志位 C 设置成无意义的值。

④ 标志位 V 不变。

注意　乘法指令不能对第二操作数使用立即数或被移位的寄存器。

7.1　MUL 乘法指令

1．指令编码格式

MUL（Multiply）32 位乘法指令将 Rm 和 Rs 中的值相乘，结果的最低 32 位保存到 Rd 中。指令的编码格式如图 7.2 所示。

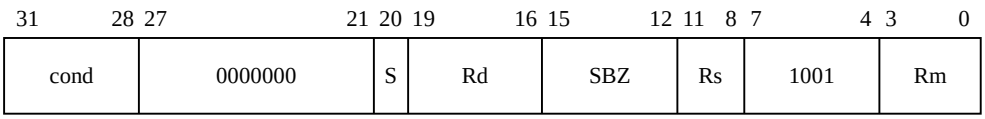

图 7.2　MUL 指令的编码格式

2．指令的语法格式

```
MUL{<cond>}{S}   <Rd>，<Rm>，<Rs>
```

① <cond>

为指令编码中的条件域。它指示指令在什么条件下执行。当<cond>忽略时，指令为无条件执行（cond=AL（Alway））。

② S

S 位（bit[20]）决定指令的操作是否影响 CPSR 中的条件标志位 N 位和 Z 位的值。当 S=1 时，更新 CPSR 中的条件标志位的值；当 S=0 时，指令不更新 CPSR 中的条件标志位。

③ <Rd>

寄存器位目标寄存器。

④ <Rm>

第一个乘数所在寄存器。

⑤ <Rs>

第二乘数所在寄存器。

3．指令操作的伪代码

指令操作的伪代码如下面程序段所示。

```
If  ConditionPassed{cond}  then
    Rd={Rm*Rs}[31:0]
    If  S==1  then
        N  flag = Rd[31]
        Z  flag = if  Rd==0  then  1  else  0
        C  flag = unaffected
        V  flag = unaffected
```

注意　当程序计数器 r15 被用作<Rd>、<Rm>、<Rs>时，指令的执行结果不可预知；当目的寄存器<Rd>和<Rm>一样时，指令的执行结果不可预知；在 ARM 版本 v5 以后的体系中，在 MULS 指令执行结束后，标志位 C 保持不变，在 v5 以前的版本中，MULS 指令执行后，标志位 C 结果不可预知。

4．指令举例

（1）R1=R2×R3

```
MUL    R1, R2, R3
```

（2）R0=R3×R7，同时设置 CPSR 中 N 位和 Z 位。

```
MULS    R0, R3, R7
```

7.2　MLA 乘-累加指令

1．指令编码格式

MLA（Multiply Accumulate）32 位乘累加指令将 Rm 和 Rs 中的值相乘，再将乘积加上第 3 个操作数，结果的最低 32 位保存到 Rd 中。
指令的编码格式如图 7.3 所示。

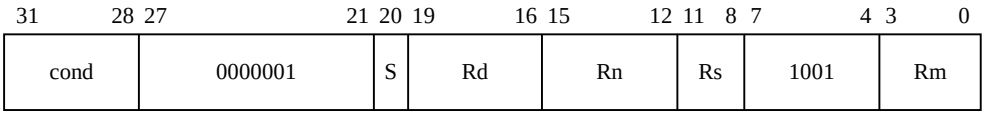

图 7.3　MLA 指令的编码格式

2．指令的语法格式

```
MLA{<cond>}{S}   <Rd>，<Rm>，<Rs>，<Rn>
```

① <cond>
为指令编码中的条件域。它指示指令在什么条件下执行。当<cond>忽略时，指令为无条件执行（cond=AL（Alway））。
② S
S 位（bit[20]）决定指令的操作是否影响 CPSR 中的条件标志位 N 位和 Z 位的值。当 S=1 时，更新 CPSR 中的条件标志位的值；当 S=0 时，指令不更新 CPSR 中的条件标志位。
③ <Rd>
寄存器位目标寄存器。
④ <Rm>
第一个乘数所在寄存器。
⑤ <Rs>
第二乘数所在寄存器。
⑥ <Rn>
将要累加到<Rm>×<Rs>结果中的第 3 操作数。

3．指令操作的伪代码

指令操作的伪代码如下面程序段所示。

```
If  ConditionPassed{cond}  then
    Rd={Rm*Rs+Rn}[31:0]
    If  S==1  then
        N  flag = Rd[31]
        Z  flag = if  Rd==0  then  1  else  0
        C  flag = unaffected
        V  flag = unaffected
```

4．指令举例

下面指令完成 R1=R2×R3＋10 的操作。

```
MOV     R0, #0x0A;
MLA     R1, R2, R3, R0;
```

7.3 UMULL 无符号数长乘指令

1．指令编码格式

UMULL（Unsigned Multiply Long）为 64 位无符号乘法指令。指令将 Rm 和 Rs 中的值做无符号数相乘，结果的低 32 位保存到 RsLo 中，而高 32 位保存到 RdHi 中。
指令的编码格式如图 7.4 所示。

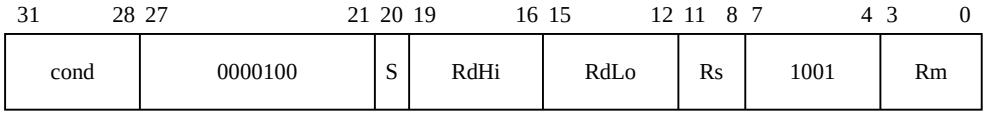

图 7.4 UMULL 指令的编码格式

2．指令的语法格式

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

① <cond>
为指令编码中的条件域。它指示指令在什么条件下执行。当<cond>忽略时，指令为无条件执行（cond=AL（Alway））。
② S
S 位（bit[20]）决定指令的操作是否影响 CPSR 中的条件标志位 N 位和 Z 位的值。当 S=1 时，更新 CPSR 中的条件标志位的值；当 S=0 时，指令不更新 CPSR 中的条件标志位。
③ <RdLo>
寄存器位目标寄存器。存储结果的低 32 位值。
④ <RdHi>
寄存器位目标寄存器。存储结果的高 32 位值。
⑤ <Rm>
第一乘数寄存器。
⑥ <Rn>
第二乘数寄存器。

3．指令操作的伪代码

指令操作的伪代码如下面程序段所示。

```
If  ConditionPassed{cond}  then
    RdHi={Rm*Rs }[63:32]
    RdLo={Rm*Rs }[31:0]
    If  S==1  then
        N  flag = RdHi[31]
```

```
Z  flag = if  ( (RdHi==0) and (RdLo==0) ) then  1  else  0
C  flag = unaffected
V  flag = unaffected
```

4．指令举例

下面指令完成（R1，R0）=R5×R8 操作。

```
UMULL   R0, R1, R5, R8;
```

7.4 UMLAL 无符号长乘-累加操作指令

1．指令编码格式

UMLAL（Unsigned Multiply Accumulate Long）为 64 位无符号长乘-累加指令。指令将 Rm 和 Rs 中的值做无符号数相乘，64 位乘积与 RdHi，RdLo 相加，结果的低 32 位保存到 RsLo 中，而高 32 位保存到 RdHi 中。

指令的编码格式如图 7.5 所示。

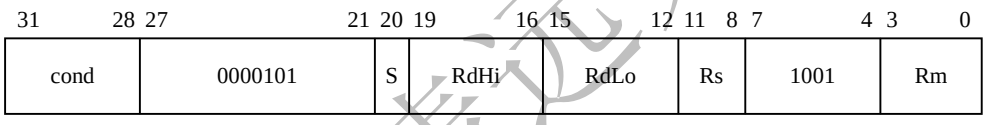

图 7.5　UMALL 指令的编码格式

2．指令的语法格式

```
UMALL{<cond>}{S}   <RdLo>，<RdHi>，<Rm>，<Rs>
```

① <cond>

为指令编码中的条件域。它指示指令在什么条件下执行。当<cond>忽略时，指令为无条件执行（cond=AL（Alway））。

② S

S 位（bit[20]）决定指令的操作是否影响 CPSR 中的条件标志位 N 位和 Z 位的值。当 S=1 时，更新 CPSR 中的条件标志位 N 位和 Z 位的值；当 S=0 时，指令不更新 CPSR 中的条件标志位。

③ <RdLo>

存储将要累加到<Rm>×<Rn>乘积结果中的加数的低 32 位数值的寄存器；同时也为寄存器位目标寄存器，存储最终结果的低 32 位值。

④ <RdHi>

存储将要累加到<Rm>×<Rn>乘积结果中的加数的高 32 位数值的寄存器；同时也为寄存器位目标寄存器，存储最终结果的高 32 位值。

⑤ <Rm>

第一乘数寄存器。

⑥ <Rn>

第二乘数寄存器。

3．指令操作的伪代码

指令操作的伪代码如下面程序段所示。

```
If  ConditionPassed{cond}  then
    RdHi={Rm*Rs }[63:32]+RdHi+CarryFrom{{Rm*Rs}[31:0]+RdLo}
    RdLo={Rm*Rs }[31:0]+RdLo
    If  S==1  then
        N  flag = RdHi[31]
        Z  flag = if  ((RdHi==0) and (RdLo==0)) then  1  else  0
        C  flag = unaffected
        V  flag = unaffected
```

4．指令举例

下面的指令完成（R1，R0）= R5×R8＋（R1，R0）操作。

```
UMLAL    R0, R1, R5,R8;
```

7.5　SMULL 无符号长乘-累加操作指令

1．指令编码格式

SMULL（Signed Multiply Long）64 位有符号长乘法指令。指令将 Rm 和 Rs 中的值做有符号数相乘，结果的低 32 位保存到 RsLo 中，而高 32 位保存到 RdHi 中。
指令的编码格式如图 7.6 所示。

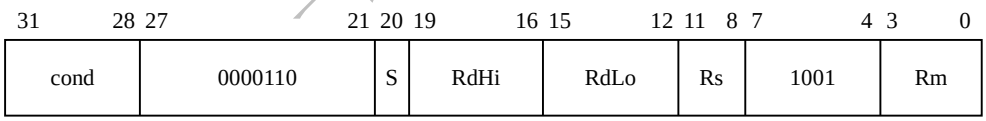

图 7.6　SMULL 指令的编码格式

2．指令的语法格式

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

① <cond>
为指令编码中的条件域。它指示指令在什么条件下执行。当<cond>忽略时，指令为无条件执行（cond=AL（Alway））。
② S
S 位（bit[20]）决定指令的操作是否影响 CPSR 中的条件标志位 N 位和 Z 位的值。当 S=1 时，更新 CPSR 中的条件标志位 N 位和 Z 位的值；当 S=0 时，指令不更新 CPSR 中的条件标志位。
③ <RdLo>
寄存器位目标寄存器，存储最终结果的低 32 位值。

④ <RdHi>

寄存器位目标寄存器，存储最终结果的高 32 位值。

⑤ <Rm>

第一乘数寄存器。

⑥ <Rn>

第二乘数寄存器。

3．指令操作的伪代码

指令操作的伪代码如下面程序段所示。

```
If  ConditionPassed{cond}  then
    RdHi={Rm*Rs }[63:32]
    RdLo={Rm*Rs }[31:0]
    If  S==1  then
        N  flag = RdHi[31]
        Z  flag = if  （（RdHi==0）and（RdLo==0））then  1  else  0
        C  flag = unaffected
        V  flag = unaffected
```

4．指令举例

下面的指令完成（R3，R2）= R7×R6 操作。

```
SMULL    R2, R3, R7,R6;
```

7.6　SMLAL 有符号长乘-累加操作指令

1．指令编码格式

SMLAL（Signed Multiply Accumulate Long）为 64 位有符号长乘法指令。指令将 Rm 和 Rs 中的值做有符号数相乘，64 位乘积与 RdHi，RdLo 相加，结果的低 32 位保存到 RsLo 中，而高 32 位保存到 RdHi 中。指令的编码格式如图 7.7 所示。

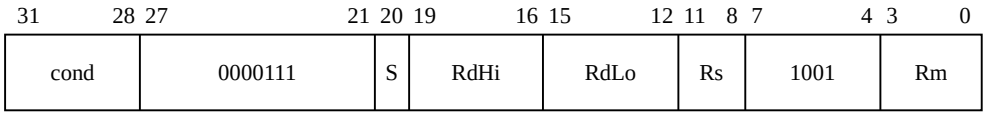

图 7.7　SMLAL 指令的编码格式

2．指令的语法格式

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

① <cond>

为指令编码中的条件域。它指示指令在什么条件下执行。当<cond>忽略时，指令为无条件执行（cond=AL（Alway））。

② S

S 位（bit[20]）决定指令的操作是否影响 CPSR 中的条件标志位 N 位和 Z 位的值。当 S=1 时，更新 CPSR 中的条件标志位 N 位 Z 位的值；当 S=0 时，指令不更新 CPSR 中的条件标志位。

其他参数详见 SMULL 指令。

3．指令操作的伪代码

指令操作的伪代码如下面程序段所示。

```
If  ConditionPassed{cond}  then
    RdHi={Rm*Rs }[63:32]+RdHi+CarryFrom{{Rm*Rs}[31:0]+RdLo}
    RdLo={Rm*Rs }[31:0]+RdLo
    If  S==1  then
        N  flag = RdHi[31]
        Z  flag = if  ((RdHi==0) and (RdLo==0)) then  1  else  0
        C  flag = unaffected
        V  flag = unaffected
```

4．指令举例

下面的指令完成（R3，R2）= R7×R6+（R3，R2）操作。

```
SMLAL   R2, R3, R7,R6;
```

联系方式

集团官网：www.hqyj.com　　嵌入式学院：www.embedu.org　　移动互联网学院：www.3g-edu.org

企业学院：www.farsight.com.cn　　物联网学院：www.topsight.cn　　研发中心：dev.hqyj.com

集团总部地址：北京市海淀区西三旗悦秀路北京明园大学校内 华清远见教育集团

北京地址：北京市海淀区西三旗悦秀路北京明园大学校区，电话：010-82600386/5

上海地址：上海市徐汇区漕溪路银海大厦 A 座 8 层，电话：021-54485127

深圳地址：深圳市龙华新区人民北路美丽 AAA 大厦 15 层，电话：0755-22193762

成都地址：成都市武侯区科华北路 99 号科华大厦 6 层，电话：028-85405115

南京地址：南京市白下区汉中路 185 号鸿运大厦 10 层，电话：025-86551900

武汉地址：武汉市工程大学卓刀泉校区科技孵化器大楼 8 层，电话：027-87804688

西安地址：西安市高新区高新一路 12 号创业大厦 D3 楼 5 层，电话：029-68785218