ARM存储器访问指令

3.2.4 存储器访问指令

- ARM指令集——存储器访问指令

ARM处理器是Load/Store型的，即它对数据的操作是通过将数据从存储器加载到片内寄存器中进行处理，处理完成后的结果经过寄存器存回到存储器中，以加快对片外存储器进行数据处理的执行速度.

存储器访问指令分为单寄存器操作指令，多寄存器操作指令和寄存器和存储器交换指令。

- ARM存储器访问指令——单寄存器加载（LDR）

助记符	说明	操作	条件码位置
LDR Rd,addressing	加载字数据	Rd←[addressing]，addressing索引	LDR{cond}
LDRB Rd,addressing	加载无符号字节数据	Rd←[addressing]，addressing索引	LDR{cond}B
LDRT Rd,addressing	以用户模式加载字数据	Rd←[addressing]，addressing索引	LDR{cond}T
LDRBT Rd, addressing	以用户模式加载无符号字节数据	Rd←[addressing]，addressing索引	LDR{cond}BT
LDRH Rd, addressing	加载无符号半字数据	Rd←[addressing]，addressing索引	LDR{cond}H
LDRSB Rd, addressing	加载有符号字节数据	Rd←[addressing]，addressing索引	LDR{cond}SB
LDRSH Rd, addressing	加载有符号半字数据	Rd←[addressing]，addressing索引	LDR{cond}SH

- ARM存储器访问指令——单寄存器存储（STR）

助记符	说明	操作	条件码位置
STR Rd, addressing	存储字数据	[addressing]←Rd，addressing索引	STR{cond}
STRB Rd,addressing	存储字节数据	[addressing]←Rd，addressing索引	STR{cond}B
STRT Rd,addressing	以用户模式存储字数据	[addressing]←Rd，addressing索引	STR{cond}T
STRBT Rd,addressing	以用户模式存储字节数据	[addressing]←Rd，addressing索引	STR{cond}BT
STRH Rd,addressing	存储半字数据	[addressing] ←Rd，addressing索引	STR{cond}H

所有单寄存器加载/存储指令可分为“字和无符号字节加载存储指令”和“半字和有符号字节加载存储指令。

• ARM存储器访问指令——单寄存器加载/存储

•LDR和STR——①字和无符号字节加载/存储指令

LDR指令用于从内存中读取单一字或字节数据存入寄存器中，STR指令用于将寄存器中的单一字或字节数据保存到内存。指令格式如下：

```
LDR{cond}{T}      Rd,<地址>    ;将指定地址上的字数据读入Rd
STR{cond}{T}      Rd,<地址>    ;将Rd中的字数据存入指定地址
LDR{cond}B{T}     Rd,<地址>    ;将指定地址上的字节数据读入Rd
STR{cond}B{T}     Rd,<地址>    ;将Rd中的字节数据存入指定地址
```

其中，T为可选后缀。若指令有T，那么即使处理器是在特权模式下，存储系统也将访问看成是在用户模式下进行的。T在用户模式下无效，不能与前索引偏移一起使用T。

• ARM存储器访问指令——单寄存器加载/存储

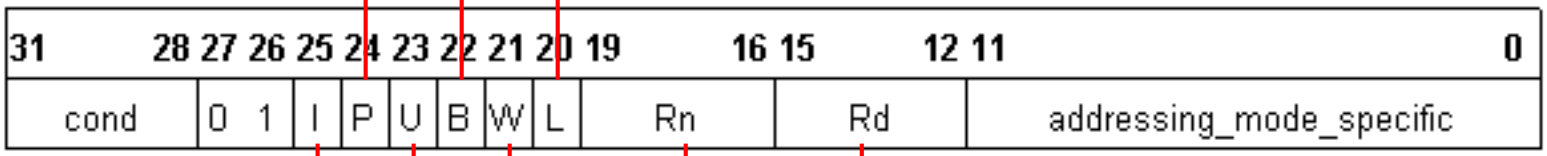
•LDR和STR——①字和无符号字节加载/存储指令编码

B为1表示字节访问，为0表示字访问

P表示前/后变址

指令执行的条件码

L用于区别加载（L为1）或存储（L为0）



I为0时，偏移量为12位立即数，为1时，偏移量为寄存器移位

为指令的寻址方式

Rd为源/目标寄存器

Rn为基址寄存器

U表示加/减

W表示回写

• ARM存储器访问指令——单寄存器加载/存储

•LDR和STR——①字和无符号字节加载/存储指令

LDR/STR指令寻址非常灵活，它由两部分组成，其中一部分为一个基址寄存器，可以为任一个通用寄存器；另一部分为一个地址偏移量。地址偏移量有以下3种格式：

▪立即数。立即数可以是一个无符号的数值。这个数据可以加到基址寄存器，也可以从基址寄存器中减去这个数值。

如：`LDR  R1,[R0,#0x12]`

▪寄存器。寄存器中的数值可以加到基址寄存器，也可以从基址寄存器中减去这个数值。

如：`LDR  R1,[R0,R2]`

▪寄存器及移位常数。寄存器移位后的值可以加到基址寄存器，也可以从基址寄存器中减去这个数值。

如：`LDR  R1,[R0,R2,LSL #2]`

- ARM存储器访问指令——单寄存器加载/存储

•LDR和STR——①字和无符号字节加载/存储指令

从寻址方式的地址计算方法分，加载/存储指令有以下4种格式：

- 零偏移。 如：LDR Rd,[Rn]
- 前索引偏移。 如：LDR Rd,[Rn,#0x04]!
- 程序相对偏移。 如：LDR Rd,labe1
- 后索引偏移。 如：LDR Rd,[Rn],#0x04

注意：大多数情况下，必须保证字数据操作的地址是32位对齐的。

Rd是用于加载或存储的ARM寄存器. Rn是存储器的基址寄存器.

• ARM存储器访问指令——单寄存器加载/存储

•LDR和STR——①字和无符号字节加载/存储指令

▪前索引偏移（pre－indexed） 如：LDR Rd,[Rn,#0x04]!

在数据传送之前,将偏移量加到Rn中,其结果作为传送数据的存储器地址.若使用后缀"!",则结果写回到Rn中,且Rn不允许是R15

Rd←[Rn+0x04];Rn←Rn+4

▪后索引偏移（post－indexed） 如：LDR Rd,[Rn],#0x04

Rn的值用做传送数据的存储器地址.在数据传送后,将偏移量加到Rn中.结果写回到Rn. Rn不允许是R15

Rd←[Rn]; Rn←Rn+0x04

- ARM存储器访问指令——单寄存器加载/存储

•LDR和STR——①字和无符号字节加载/存储指令

例子

```
LDR     R8, [R10]          ; R8←[R10]
LDRNE   R2, [R5,#960]      ; (有条件地) R2←[R5+960] , R5←R5+960
LDR     R0, localdata      ; 加载一个字,该字位于标号localdata所在地址
STR     R2, [R9,#consta-struc]  ;consta-struc是常量表达式,该常量;
值的范围为0—4095

STRB    R0, [R3,-R8,ASR #2]     ;R0→[R3-R8/4],存储R0的最低有效
                                ;字节R3和R8不变.

STR     R5, [R7],#-8            ;R5→[R7] , R7←R7-8
```

• ARM存储器访问指令——单寄存器加载/存储

•LDR和STR——②半字和有符号字节加载/存储指令

这类LDR/STR指令可加载有符号半字或字节，可加载/存储无符号半字。偏移量格式、寻址方式与加载/存储字和无符号字节指令相同。

```
LDR{cond}SB   Rd,<地址>   ;将指定地址上的有符号字节读入Rd
LDR{cond}SH   Rd,<地址>   ;将指定地址上的有符号半字读入Rd
LDR{cond}H    Rd,<地址>   ;将指定地址上的半字数据读入Rd
STR{cond}H    Rd,<地址>   ;将Rd中的半字数据存入指定地址
```

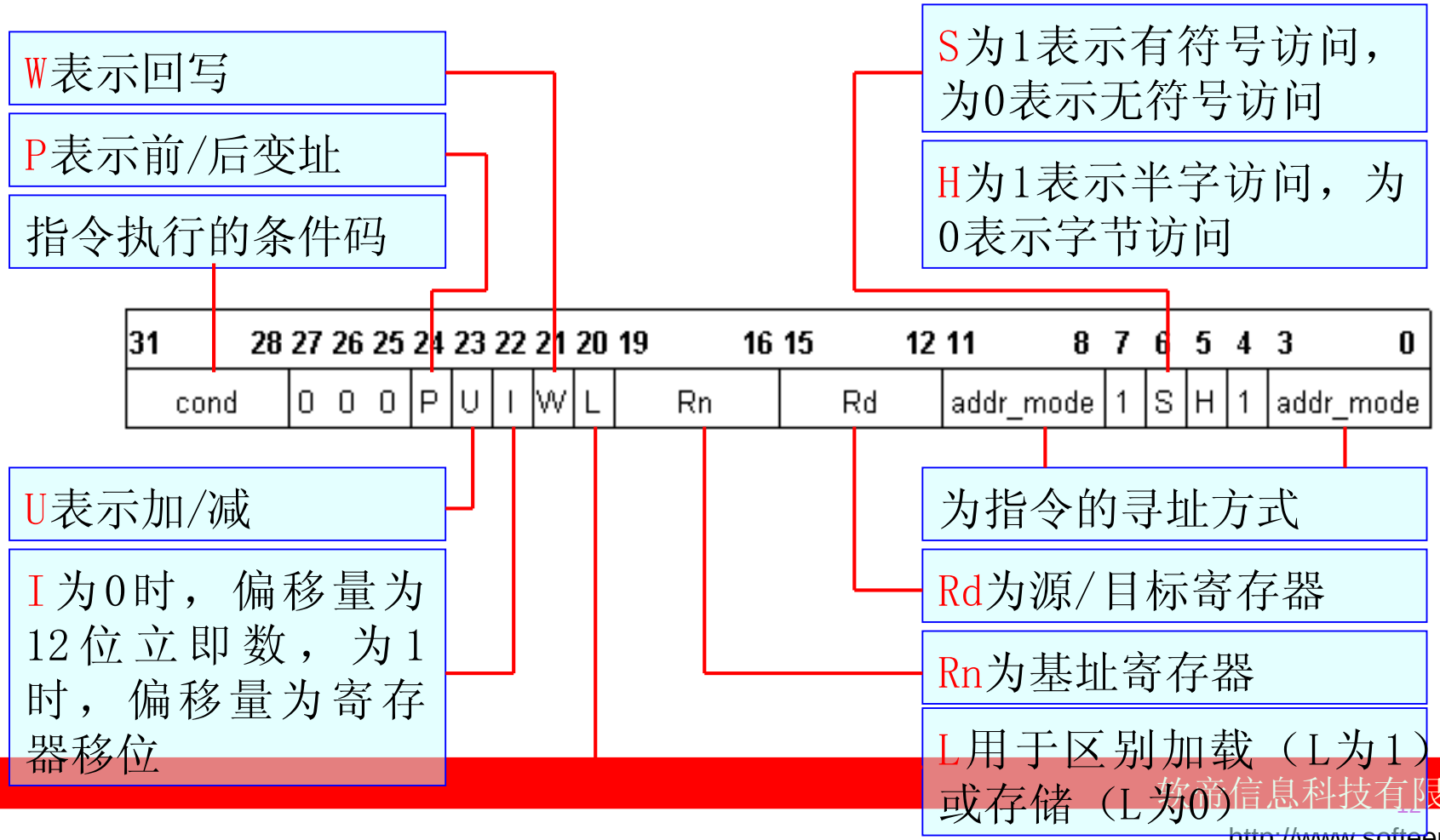
注意：

1. 有符号位半字/字节加载是指用符号位加载扩展到32位，无符号半字加载是指用零扩展到32位；

2. 半字读写的指定地址必须为偶数，否则将产生不可靠的结果；

- ARM存储器访问指令——单寄存器加载/存储

•LDR和STR——②半字和有符号字节加载/存储指令编码



- ARM存储器访问指令——单寄存器加载/存储

•LDR和STR——②半字和有符号字节加载/存储指令

例子

```
LDREQSH   R11, [R6]        ;(有条件地),R11←[R6]加载16位半字,
                           ;带符号扩展到32位
LDRH      R1 , [R0,#22]    ;R1←[R0+22],加载16位半字,零扩展到
                           ;32位
STRH      R4 , [R0,R1]!    ;R4→[R0+R1],存储最低的有效半字到
                           ; R0+R1地址开始的两个字节,地址写回
                           ;到R0
LDRSB     R6 , constf      ;加载位于标号constf地址中的字节,带
                           ;符号扩展
```

错例

```
LDRSB     R1 , [R6], R3,LSL #4 ;这种格式只对字和无符号字节
                               ;传送有效
```

- ARM存储器访问指令——单寄存器加载/存储

•LDR和STR指令应用示例：

1. 加载/存储字和无符号字节指令

```
LDR     R2,[R5]          ;将R5指向地址的字数据存入R2
STR     R1,[R0,#0x04]    ;将R1的数据存储到R0+0x04地址
LDRB    R3,[R2],#1       ;将R2指向地址的字节数据存入R3，R2＝R2+1
STRB    R6,[R7]          ;将R7指向地址的字节数据存入R6
```

2. 加载/存储半字和有符号字节指令

```
LDRSB   R1,[R0,R3]       ;将R0+R3地址上的字节数据存入R1，
                         ;高24位用符号扩展
LDRH    R6,[R2],#2       ;将R2指向地址的半字数据存入R6，高16位用0扩展
                         ;读出后，R2=R2+2
STRH    R1,[R0,#2]!      ;将R1的半字数据保存到R0+2地址，
                         ;只存储R1的低2字节内容，R0=R0+2
```

- ARM存储器访问指令——多寄存器加载/存储

助记符	说明	操作	条件码位置
LDM{mode} Rn{!},reglist	多寄存器加载	reglist←[Rn...], Rn回写等	LDM{cond} {mode}
STM{mode} Rn{!},reglist	多寄存器存储	[Rn...]←reglist,Rn回写等	STM{cond} {mode}

多寄存器加载/存储指令可以实现在一组寄存器和一块连续的内存单元之间传输数据。LDM为加载多个寄存器；STM为存储多个寄存器。允许一条指令传送16个寄存器的任何子集或所有寄存器。它们主要用于现场保护、数据复制、常数传递等。

- ARM存储器访问指令——多寄存器加载/存储

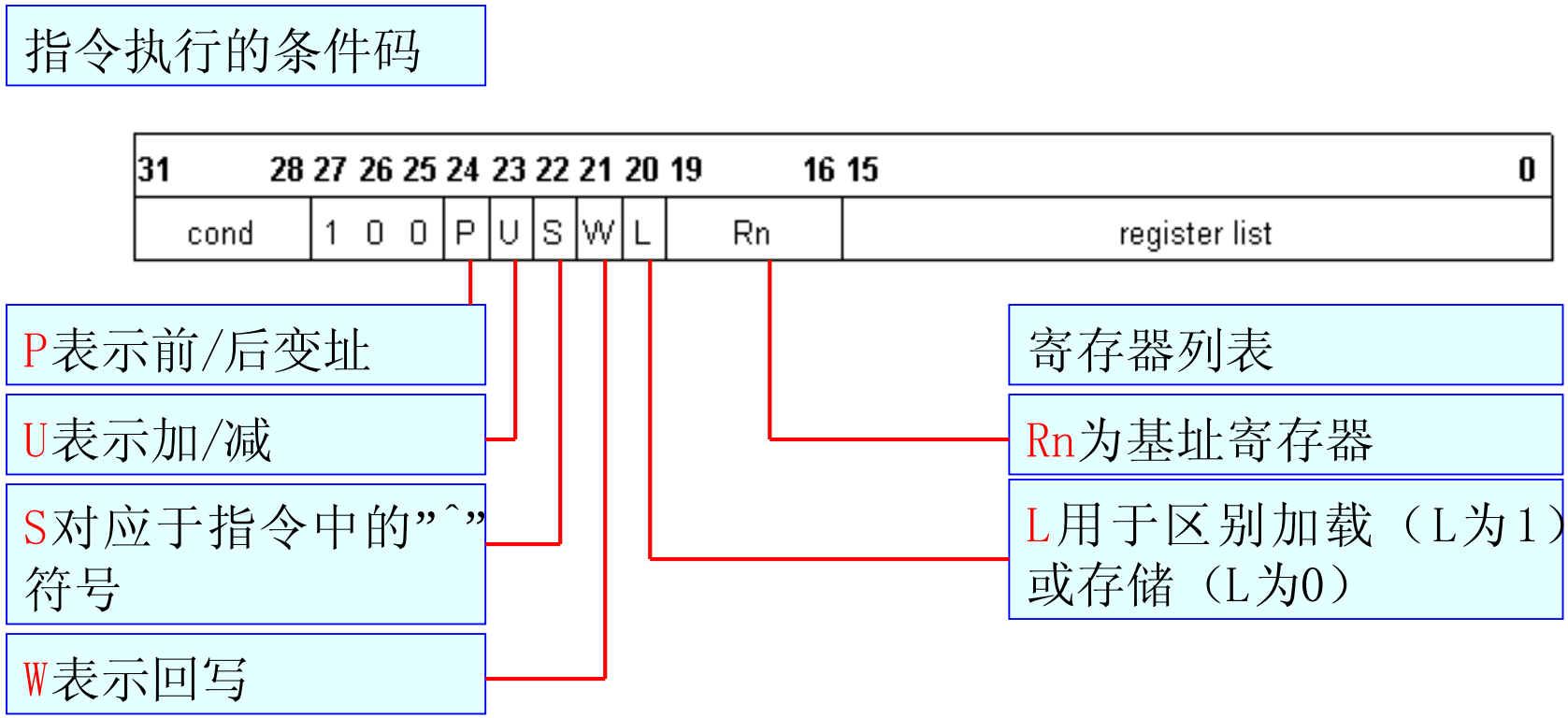
多寄存器加载/存储指令格式如下：

```
LDM{cond}<模式>  Rn{!},reglist{^}
STM{cond}<模式>  Rn{!},reglist{^}
```

- cond：指令执行的条件；
- 模式：控制地址的增长方式，一共有8种模式；
- !：表示在操作结束后，将最后的地址写回Rn中；
- reglist ：表示寄存器列表，可以包含多个寄存器，它们使用“,”隔开，如{R1,R2,R6-R9}，寄存器由小到大排列；
- ^：加入该后缀后，进行数据传送且寄存器列表不包含PC时，加载/存储的寄存器是用户模式下的，而不是当前模式的寄存器。若在LDM指令且寄存器列表中包含有PC时使用，那么除了正常的多寄存器传送外，还将SPSR也拷贝到CPSR中，这可用于异常处理返回。注意：该后缀不允许在用户模式或系统模式下使用。

- ARM存储器访问指令——多寄存器加载/存储

  - LDM和STM——多寄存器加载/存储指令编码



- ARM存储器访问指令——多寄存器加载/存储

多寄存器加载/存储指令的8种模式如下表所示，右边四种为堆栈操作、左边四种为数据传送操作。

模式	说明	模式	说明
IA	每次传送后地址加4	FD	满递减堆栈
IB	每次传送前地址加4	ED	空递减堆栈
DA	每次传送后地址减4	FA	满递增堆栈
DB	每次传送前地址减4	EA	空递增堆栈
数据块传送操作		堆栈操作	

进行数据复制时，先设置好源数据指针和目标指针，然后使用块拷贝寻址指令LDMIA/STMIA、LDMIB/STMIB、LDMDA/STMDA、LDMDB/STMDB进行读取和存储 。

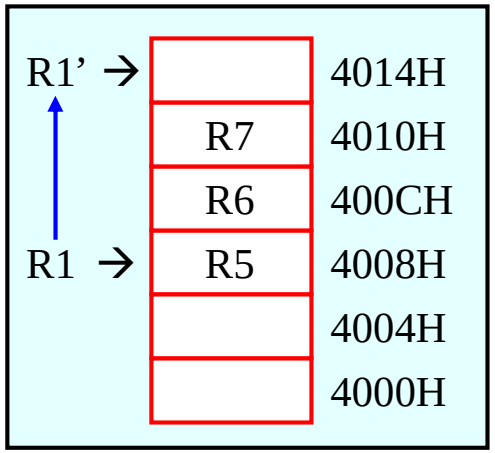
进行堆栈操作操作时，要先设置堆栈指针（SP），然后使用堆栈寻址指令STMFD/LDMFD 、STMED/LDMED、STMFA/LDMFA和STMEA/LDMEA实现堆栈操作。

- ARM存储器访问指令——多寄存器加载/存储

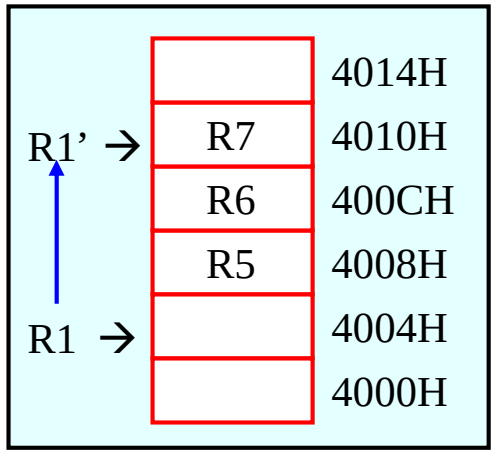
数据块传送指令操作过程如右图所示，其中R1为指令执行前的基址寄存器，R1'则为指令执行后的基址寄存器。

注意：

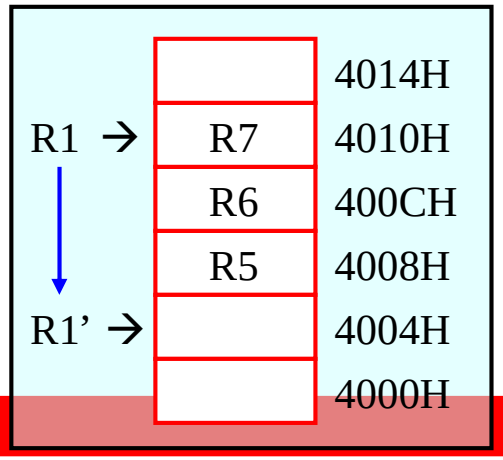
每条指令是如何将3个寄存器的数据存入寄存器的，以及在使用自动变址的情况下基址寄存器是如何改变的



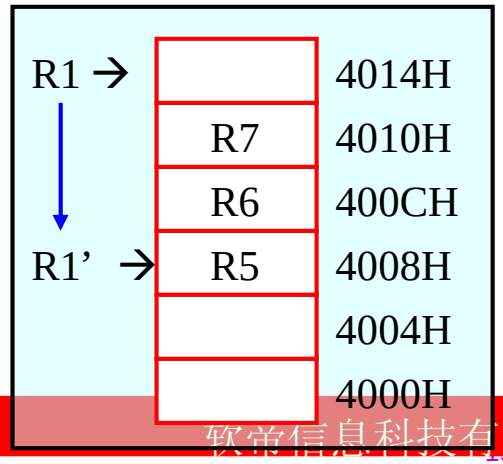
指令STMIA R1!,{R5-R7}



指令STMIB R1!,{R5-R7}



指令STMDA R1!,{R5-R7}



指令STMDB R1!,{R5-R7}

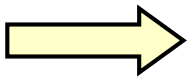
- ARM存储器访问指令——多寄存器加载/存储

堆栈操作和数据块传送指令类似，也有4种模式，它们之间的关系如下表所示：

数据块传送 存储	堆栈操作 压栈	说明
STMDA	STMED	空递减
STMIA	STMEA	空递增
STMDB	STMFD	满递减
STMIB	STMFA	满递增

数据块传送 加载	堆栈操作 出栈	说明
LDMDA	LDMFA	满递减
LDMIA	LDMFD	满递增
LDMDB	LDMEA	空递减
LDMIB	LDMED	空递增

```
;使用数据块传送指令进行堆栈操作
STMDA   R0!,{R5-R6}
. . .
LDMIB   R0!,{R5-R6}
```



```
;使用堆栈指令进行堆栈操作
STMED   R0!,{R5-R6}
. . .
LDMED   R0!,{R5-R6}
```

两段代码的执行结果是一样的，但是使用堆栈指令的压栈和出栈操作编程很简单（只要前后一致即可），而使用数据块指令进行压栈和出栈操作则需要考虑空与满、加与减对应的问题。

- ARM存储器访问指令——多寄存器加载/存储

例子

```
LDMIA   R8,   {R0,R2,R9}; 加载R8指向的地址上多
                          ; 字数据，保存到R0，R2
                          ; R9中，R8值不变
STMDB   R1!, {R3-R6,R11,R12}
STMFD   R13!,{R0,R4-R7,LR};寄存器进栈
LDMFD   R13!,{R0,R4-R7,PC};寄存器出栈,从子程序
                           返回
```

错例

```
STMIA   R5!,{R5,R4,R9};R5存储的值不可预知
LDMDA   R2 ,{}          ;列表中至少要有1个寄存器
```

- ARM存储器访问指令——寄存器和存储器交换指令

助记符	说明	操作	条件码位置
SWP Rd,Rm,Rn	寄存器和存储器字数据交换	Rd←[Rn]，[Rn]←Rm (Rn≠Rd或Rm)	SWP{cond}
SWPB Rd,Rm,Rn	寄存器和存储器字节数据交换	Rd←[Rn]，[Rn]←Rm (Rn≠Rd或Rm)	SWP{cond}B

SWP指令用于将一个内存单元(该单元地址放在寄存器Rn中)的内容读取到一个寄存器Rd中，同时将另一个寄存器Rm的内容写入到该内存单元中。使用SWP可实现信号量操作。

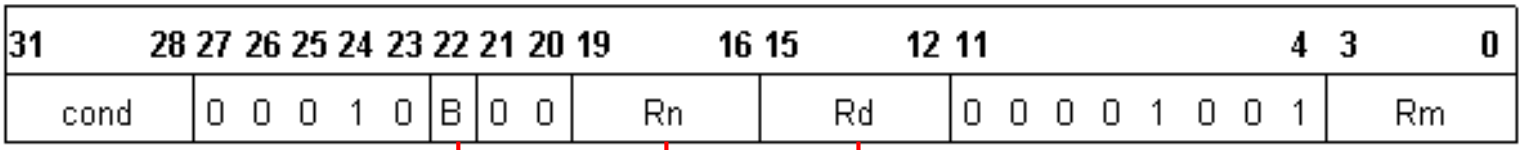
指令格式如下：

```
SWP{cond}{B}    Rd,Rm,[Rn]
```

其中，B为可选后缀，若有B，则交换字节，否则交换32位字；Rd用于保存从存储器中读入的数据；Rm的数据用于存储到存储器中，若Rm与Rn相同，则为寄存器与存储器内容进行交换；Rn为要进行数据交换的存储器地址，Rn不能与Rd和Rm相同。

- ARM存储器访问指令——寄存器和存储器交换指令

•SWP和SWPB——寄存器和存储器交换指令编码



指令执行的条件码

B用于区别无符号字节（B为1）或字（B为0）

Rm源寄存器

Rd目标寄存器

Rn为基址寄存器

•SWP指令应用示例：

```
SWP     R1,R1,[R0]      ;将R1的内容与R0指向的存储单元的内容进行交换
SWPB    R1,R2,[R0]      ;将R0指向的存储单元内的容读取一字节数据到R1中
                        ;(高24位清零)，并将R2的内容写入到该内存单元中
                        ;(最低字节有效)
```

- ARM杂项指令——状态寄存器读指令

在ARM处理器中，只有MRS指令可以对状态寄存器CPSR和SPSR进行读操作。通过读CPSR可以了解当前处理器的工作状态。读SPSR寄存器可以了解到进入异常前的处理器状态。

MRS指令格式

```
MRS{cond}    Rd,psr
```

- ARM杂项指令——状态寄存器读指令

在ARM处理器中，只有MRS指令可以对状态寄存器CPSR和SPSR进行读操作。通过读CPSR可以了解当前处理器的工作状态。读SPSR寄存器可以了解到进入异常前的处理器状态。

MRS指令格式

```
MRS{cond}    Rd,psr
```

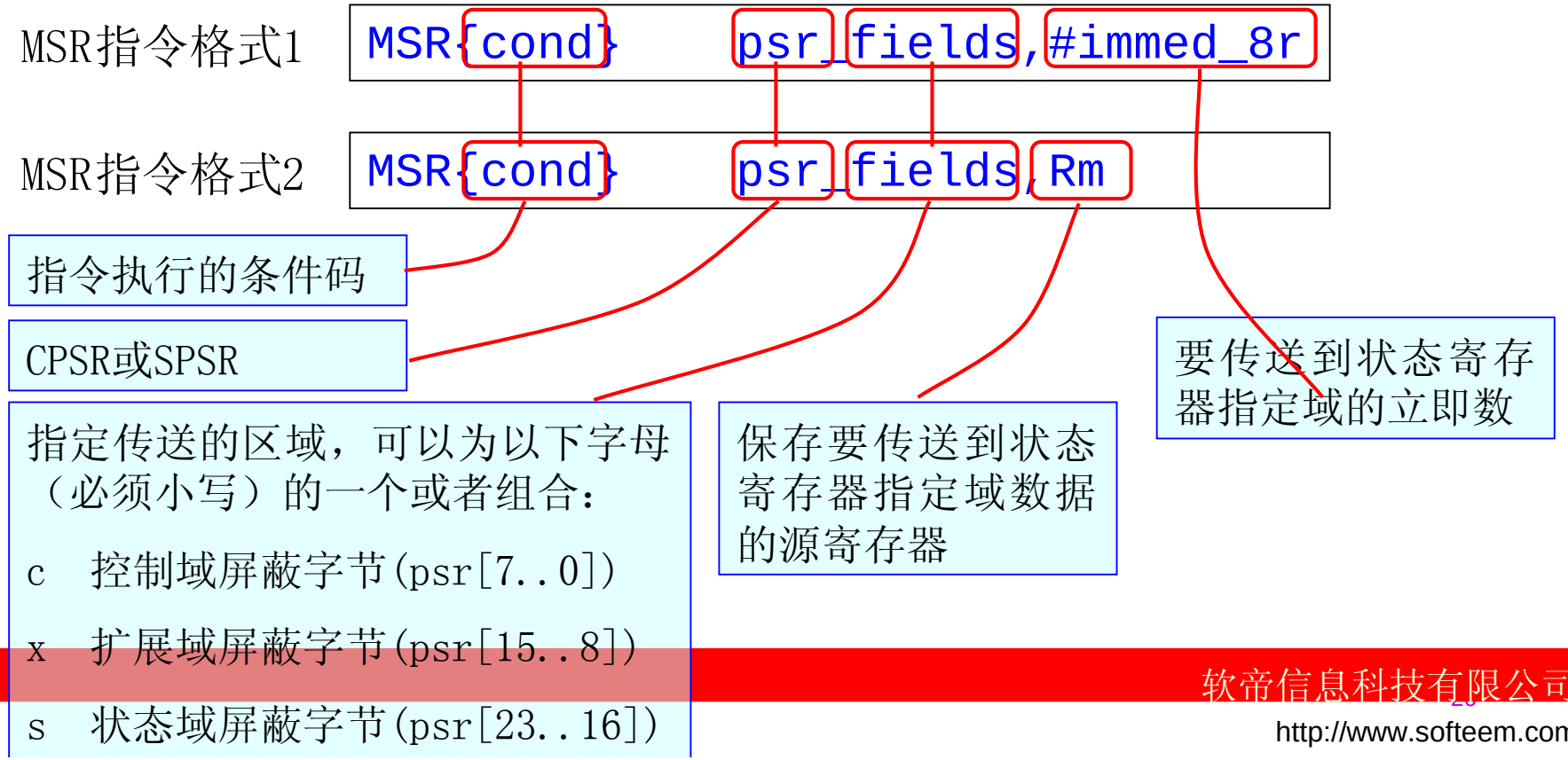
应用示例：

```
MRS     R1,CPSR        ; 将CPSR状态寄存器读取，保存到R1中
MRS     R2,SPSR        ; 将SPSR状态寄存器读取，保存到R2中
```

3.3 ARM指令的使用

- ARM杂项指令——状态寄存器写指令

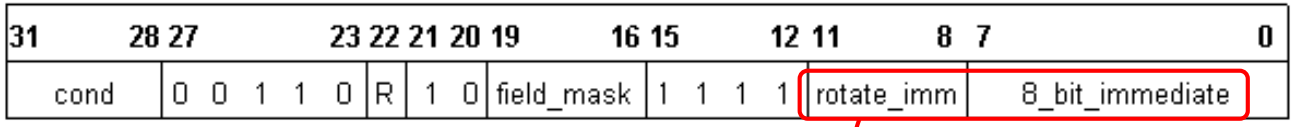
在ARM处理器中，只有MSR指令可以对状态寄存器CPSR和SPSR进行写操作。与MRS配合使用，可以实现对CPSR或SPSR寄存器的读-修改-写操作，可以切换处理器模式、或者允许/禁止IRQ/FIQ中断等。



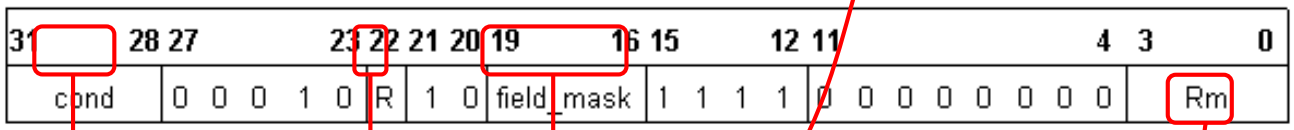
- ARM杂项指令——状态寄存器写指令

在ARM处理器中，只有MSR指令可以对状态寄存器CPSR和SPSR进行写操作。与MRS配合使用，可以实现对CPSR或SPSR寄存器的读-修改-写操作，可以切换处理器模式、或者允许/禁止IRQ/FIQ中断等。

MSR指令1编码



MSR指令2编码



指令执行的条件码

CPSR或SPSR

指定传送的区域，可以为以下字母（必须小写）的一个或者组合：

c 控制域屏蔽字节(psr[7..0])

x 扩展域屏蔽字节(psr[15..8])

s 状态域屏蔽字节(psr[23..16])

Rotate : 立即数对齐

8_bit_immediate : 8位立即数

要传送到状态寄存器指定域的立即数

- ARM杂项指令——状态寄存器写指令

在ARM处理器中，只有MSR指令可以对状态寄存器CPSR和SPSR进行写操作。与MRS配合使用，可以实现对CPSR或SPSR寄存器的读-修改-写操作，可以切换处理器模式、或者允许/禁止IRQ/FIQ中断等。

应用示例1：

```
;子程序：使能IRQ中断
ENABLE_IRQ
    MRS     R0, CPSR            (1)
    BIC     R0, R0,#0x80        (2)
    MSR     CPSR_c,R0           (3)
    MOV     PC,LR               (4)
```

1. 将CPSR寄存器内容读出到R0；

2. 修改对应于CPSR中的I控制位；

应用示例2：

```
;子程序：禁能IRQ中断
DISABLE_IRQ
    MRS     R0 CPSR             (1)
    ORR     R0, R0,#0x80        (2)
    MSR     CPSR_c,R0           (3)
    MOV     PC,LR               (4)
```

3. 将修改后的值写回 CPSR寄存器的对应控制域；

4. 返回上一层函数；

Q & A