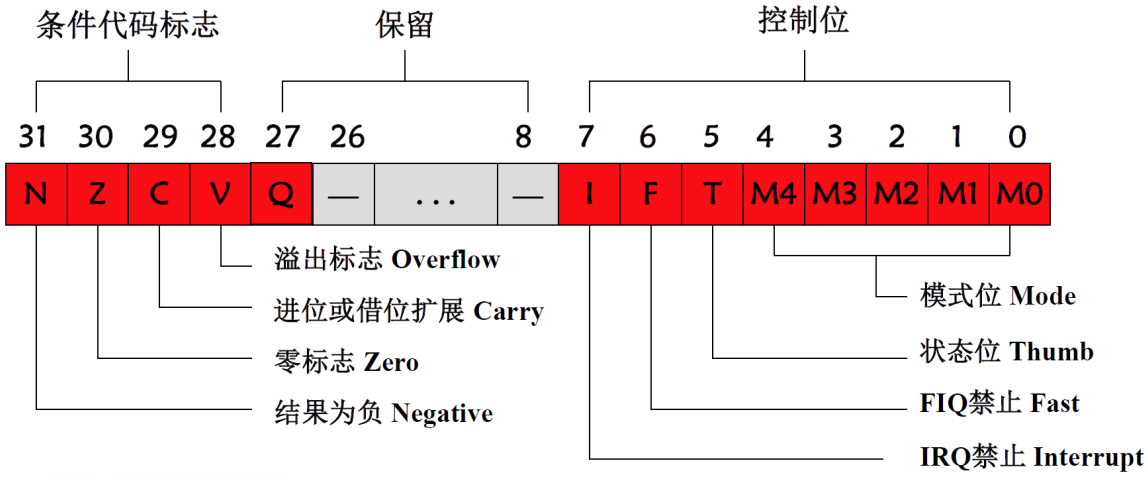
ARM 指令

- MCR/MRC 协处理器操作
  - MCR/MRC {条件} 协处理器编码，协处理器操作码1，目的寄存器，源寄存器1，源寄存器2，协处理器操作码
  - MCR：协处理器<-ARM处理器
  - MRC：ARM处理器<-协处理器
  - 例：MCR P2，3，R2，C4，C5，6 ;指定协处理器P2执行第3种操作，类型6，操作数是 R2，结果放在C4中

```
MRC P15,0,R0,C1,0,0
ORR R0,#0x1
MCR P15,0,R0,C1,0,0
```

- MSR/MRS 通用寄存器和状态寄存器数据传送指令
  - MRS{cond} <Rd>, CPSR/SPSR
    - 如 MRS R0,CPSR；把状态放到R0中
  - MSR{cond} CPSR/SPSR_<field>,<op1>
    - 如 MSR CPRS_f, R0；把条件域修改为R0的值



状态寄存器位	Field域	标识
位[31:24]	条件标志域	f
位[23:16]	状态位域	s
位[15:8]	扩展位域	x
位[7:0]	控制位域	c

- BL 子程序调用
  - 将返回地址放到LR中，同时PC指向子程序入口点
  - 返回时 MOV PC, LR
- 移位操作
  - LSL 逻辑左移
  - LSR 逻辑右移
  - ROR 循环右移
  - ASR 算术右移（保持最左端的位数不变）
  - RRX 扩展的循环右移
    - 左侧空位用状态寄存器C位填充，右侧移出的位数放进C中
    - **没有操作数** 每次执行只能移动一位
- BIC 位清除
  - BIC R0, R0, #5 ； 清除R0中的第0和第2位
- LDM 批量数据加载

- 格式：

  LDM{<cond>}{<type>} <Rn> {!}, <regs> {^}

- 功能：连续存储单元→寄存器（多个）
- 该指令一般用于多个寄存器数据的出栈。
- 格式说明：
- **type**字段种类：**8**种。
- **Rn**：基址寄存器，其值是内存单元的起始地址。不允许为**R15**。
- **Regs**：寄存器列表，从最低序号寄存器开始，与书写顺序无关。
- **!**后缀：指令执行完毕后，将最后的地址写入基址寄存器。
- **^**后缀：当**regs**中不包含**PC**时，该后缀用于指示指令所用的寄存器为用户模式下的寄存器。

  否则，指示指令执行时，**SPSR →CPSR**。

- type字段种类：

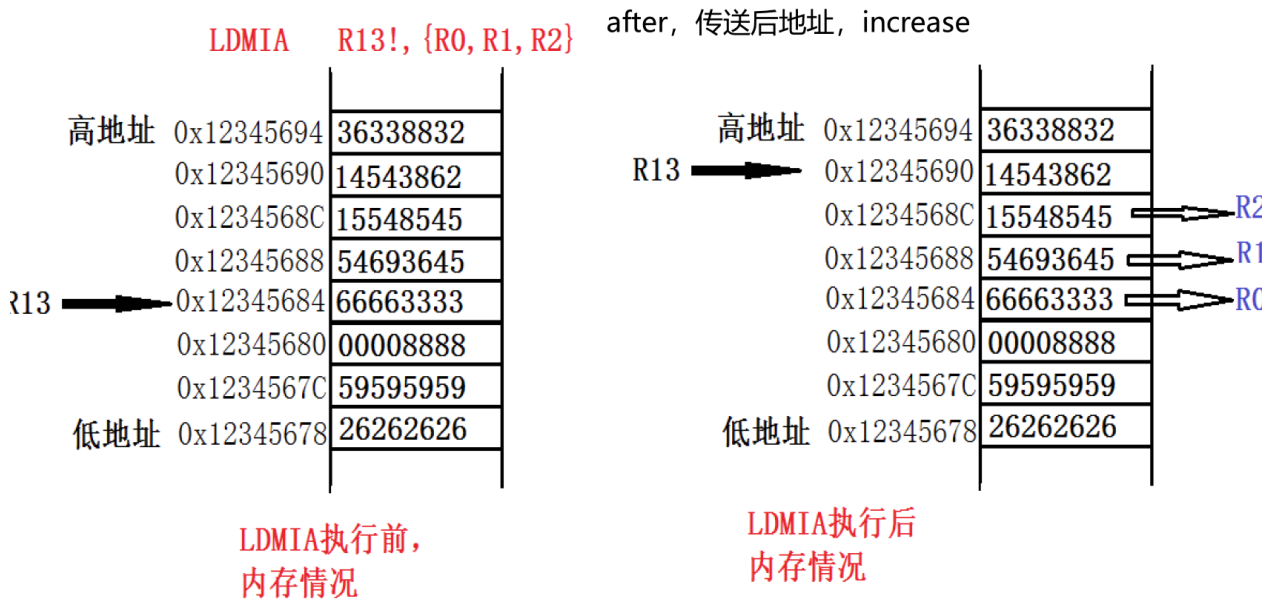
（1）拷贝

IA：每次传送后地址加	LDMIA（内存块读）	STMIA（内存块写）
IB：每次传送前地址加	LDMIB（内存块读）	STMIB（内存块写）
DA：每次传送后地址减	LDMDA（内存块读）	STMDA（内存块写）
DB：每次传送前地址减	LDMDB（内存块读）	STMDB（内存块写）

（2）堆栈

FD：满递减堆栈	LDMFD（出栈）	STMFD（进栈）
ED：空递减堆栈	LDMED（出栈）	STMED（进栈）
FA：满递增堆栈	LDMFA（出栈）	STMFA（进栈）
EA：空递增堆栈	LDMEA（出栈）	STMEA（进栈）

- 例如 **LDMIA R13!, {R0-R1, R2}**



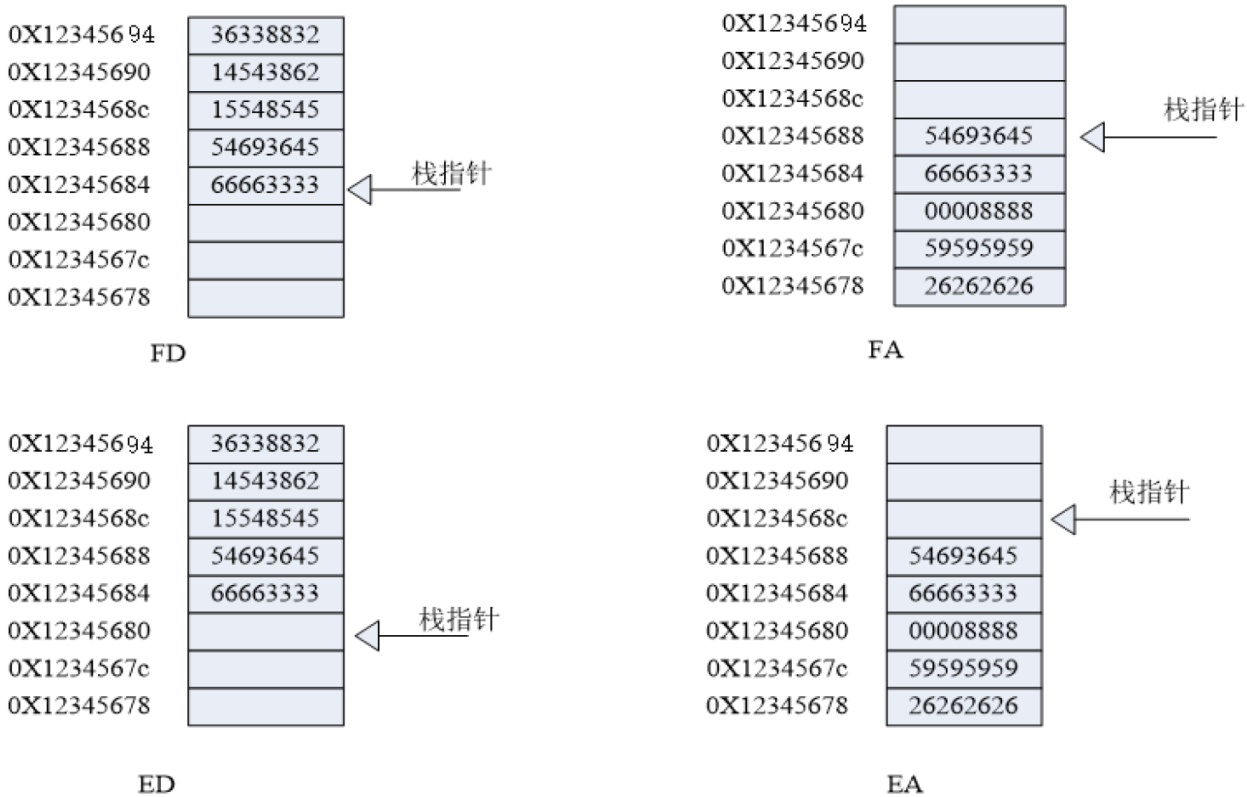
- **ARM微处理器支持四种类型的堆栈工作方式，即：**

  （1）满递增方式FA（Full Ascending）：堆栈指针指向最后入栈的数据位置，且由低地址向高地址生成。

  （2）满递减方式FD（Full Decending）：堆栈指针指向最后入栈的数据位置，且由高地址向低地址生成。

  （3）空递增方式EA（Empty Ascending）：堆栈指针指向下一个入栈数据的空位置，且由低地址向高地址生成。

  （4）空递减方式ED（Empty Decending）：堆栈指针指向下一个入栈数据的空位置，且由高地址向低地址生成。



- MVN 数据取反传送
  - MVN R0, #0; R0=0xFFFFFFFF

- 运算指令
  - RSB 反向减法
    - RSB R0,R1,#5; R0=5-R1
  - SMULL 64位有符号乘法
    - SMULL R0,R1,R2,R3; R0=R2*R3的低32位，R1为高
  - SMLAL 64位有符号数乘加
    - SMLAL R0,R1,R2,R3; R0=R2*R3低32位+R0，R1为高
- BIC 位清除
  - BIC R0,R1,#5; 将R1中的#5中为1的位（0和2位）清零，赋给R0
- 逻辑指令
  - AND, ORR, EOR
- 比较测试指令
  - CMP 比较指令
    - 不需要显示写出S来指定更改状态标志
  - TEQ 相等测试
    - 按位逻辑异或，根据结果更新CPSR
    - TEQ R0,#5; 判断是否相等
  - TST 位测试
    - 不需要显示写出S来指定更改状态标志，检查是否有相应的位
    - TST R0,#5; 检查R0中第0和2位是否为1
- SWP 字数据交换指令
  - SWP R0,R1,[R2]; R0=[R2],[R2]=R1
- SWI 软件中断
  - SWI{cond} immi; 24位立即数若不指定，则由R0指定，参数用其他寄存器传递
  - SWI 0x05
- RLIST 寄存器列表定义
  - <name> RLIST <regs>；给寄存器列表命名为name
- B/BL/BX/BLX 跳转
  - B/BL{cond} <addr>
  - PC=PC+addr<2 addr的值是相对于当前PC的一个偏移量，是24位有符号数，符号扩展后左移两位，故范围为PC-32MB~PC+32MB
  - BL，带返回的跳转指令，用于子程序调用，将子程序的返回地址保存到LR(R14中
  - BX <Rn>; 目标地址为寄存器和0xFFFFFFFE与的结果
    - ADR R0, exit
    - BX R0
  - BLX <addr>/<Rn>

- BX BLX 的目标地址可以是ARM指令，也可以是Thumb指令

- 伪指令
  - 在ARM汇编语言程序里，有一些特殊指令助记符，这些助记符与指令系统的助记符不同，没有相对应的操作码，通常称这些特殊指令助记符为伪指令。告诉编译器相关信息，仅在汇编过程中起作用。
  - 通用伪指令
    - 定义局部变量：LCLA、LCLL、LCLS
    - 定义全局变量：GBLA、GBLL、GBLS
    - 变量赋值：SETA、SETL、SETS
    - 寄存器列表定义名称：RLIST
    - AREA 段名，属性等
      - AREA test, CODE, READONLY
      - AREA |1data|, DATA, READWRITE
    - CODE16/CODE32
      - 指示编译器后面的代码时16位的Thumb或32位的ARM
        - 例如：

```
CODE32                          ； 32位的ARM指令
AREA        ||.text||,CODE，READONLY
…
LDR         R0，= 0x8500+1
BX          R0      ；程序跳转，并将处理器切换到Thumb状态
…
CODE16                          ；16位的Thumb指令
ADD         R3，R3，1
END                             ；源文件结束
```

  - ARM 伪指令
    - LTORG 声明一个数据缓冲池（文字池）的开始，通常放在无条件跳转指令之后，或者子程序返回指令之后，以免处理器错误地将数据缓冲池中地数据作为指令来执行。

例如：

```
Func1
        ……
        MOV     PC，  LR
        LTORG
DATA
        SPACE   256       ；从DATA标号开始预留256字节的内存单元。
        END
```

    - ADR 把地址加载到寄存器中

        - ADR是基于PC的相对偏移的地址值读到目标寄存器中，编译时先计算当前指令位置到expr的距离，然后用ADD或SUB替换
        - ADR R0, LOOP; ADR R1, LOOP+0x40*2
    - ADRL 支持的范围更大，用两条合适的指令替换
        - ADD register,PC,offset1;　ADD register,register,offset2
    - LDR (伪指令) 把常量或地址加载到寄存器中
        - LDR R1, =0xFFE;汇编器汇编为 MOV R1, #0xFFE
        - LDR R1, =START;加载START处地址到R1中
    - LDR (非伪指令)
        - 后面带=的是伪指令
        - LDR R0, 0x12345678;把地址中的值存放到R0中，而MOV无法做到
        - LDR R0,=0x12345678;把该值写入R0中，类似MOV
    - NOP 会编译为无效指令
- 汇编控制伪指令
    - MACRO

```
MACRO SUB_M &dest,&src1,$src2
        SUB &dest,&src1,&src2
MEND
```