# Arm 常用指令一览表 V1.1

作者：李志勇

| 指令 | 功能 | 实例 | 注释 |
|---|---|---|---|
| Mov | 给一个寄存器赋值 | Mov r0, #10<br>Mov r0, r1 | R0 = 10<br>R0 = r1 |
| Movt | 给一个寄存器的高 16 位赋值 | Movt r0, #10<br>Movt r0, r1 | 把一个 16 位的 op2 放到 r0 的[31-16]，r0 的[15-0]不变<br>注意：op2 不能大于 16 位，但可以不符合立即数规则，如 op2 可以是 0x1234，op1 必须是可读可写的 |
| Mvn | 把一个数值按位取反后赋值给一个寄存器 | Mvn r0，#0xff<br>Mvn r0， r1 | R0 = ~0xff<br>R0 = ~r1 |
| Add | 计算两个数值的加法 | Add r0，r0，#10<br>Add r0，r0，r1 | R0 = r0 + 10<br>R0 = r0 + r1 |
| Adc | 带进位的加法 | Adc r0，r0，#10<br>Adc r0，r0，r1 | R0 = r0 + 10 + C<br>R0 = r0 + r1 + C |
| Sub | 计算两个数值的减法 | Sub r0，r0，#10<br>Sub r0，r0，r1 | R0 = r0 - 10<br>R0 = r0 - r1 |
| Sbc | 带借位的减法 | Sbc r0，r0，#10<br>Sbc r0，r0，r1 | R0 = r0 - 10 - !C<br>R0 = r0 - r1 - !C |
| Rsb | 反转减法 | Rsb r0，r0，r1<br>Rsb r0，r0，#10 | R0 = r1 - r0<br>R0 = 10 - r0 |
| Rsc | 带借位的反转减法 | Rsc r0，r0，r1<br>Rsc r0，r0，#10 | R0 = r1 - r0 - !C<br>R0 = 10 - r0 - !C |
| Mul | 乘法 | Mul r0，r1，r2 | R0 = r1 * r2 |
| Mla | 乘加 | Mul r0，r1，r2，r3 | R0 = r1 * r2 + r3 |
| Mls | 乘减 | Mls r0，r1，r2，r3 | R0 = r3 - r1 * r2 |
| Orr | 按位或 | Orr r0，r0，r1<br>Orr r0，r0，#10 | R0 = r0 \| r1<br>R0 = r0 \| 10 |
| Eor | 按位异或 | Eor r0，r0，#10<br>Eor r0，r0，r1 | R0 = r0 ^ 10<br>R0 = r0 ^ r1 |
| And | 按位与 | And r0，r0，r1<br>And r0，r0，#10 | R0 = r0 & r1<br>R0 = r0 & 10 |

| | | | |
|---|---|---|---|
| Bic | 位取反 | Bic r0, r0, r1<br>Bic r0, r0, #10 | R0 = r0 & (~r1)<br>R0 = r0 & (~10) |
| Lsr | 逻辑右移 | R0, lsr r1<br>R0, lsr #10 | R0 >>> r1<br>R0 >>> 1 |
| Lsl | 逻辑左移 | R0, lsl r1<br>R0, lsl #1 | R0 <<< r1<br>R0 <<< 1 |
| Asr | 算术右移 | R0, asr r1<br>R0, asr #1 | R0 >> r1<br>R0 >> 1 |
| Ror | 循环右移 | R0, ror r1<br>R0, ror #1 | (R0 >>> r1) \| (R0 <<< (32 - r1))<br>(R0 >>> 1) \| (R0 <<< (32 - 1)) |
| Cmp | 比较 | Cmp r0, r1<br>Cmp r0, #10 | R0 - r1 影响 cpsr 标志位<br>R0 - 10 影响 cpsr 标志位 |
| Teq | 比较（按位异或） | Teq r0, r1<br>Teq r0, #10 | R0 ^ r1 影响 cpsr 标志位<br>R0 ^ 10 影响 cpsr 标志位 |
| Tst | 比较（按位与） | Tst r0, r1<br>Tst r0, #10 | R0 & r1 影响 cpsr 标志位<br>R0 & 10 影响 cpsr 标志位 |
| Mrs | 读 cpsr | Mrs r0, cpsr | R0 = cpsr |
| Msr | 写 cpsr | Msr cpsr, r0 | Cpsr = r0 |
| Swi | 软中断 | Swi 10 | 产生软中断异常 |
| Svc | 等同于 swi | Svc 10 | 产生软中断异常 |
| Ldr | 把数据从内存加载的寄存器 | Ldr r0, addr<br>ldr r0, =addr<br>ldr r1, [r0]<br>ldr r1, [r0, #4]<br>ldr r1, [r0, #4]!<br>ldr r1, [r0], #4 | R0 = *addr<br>R0 = addr<br>R1 = *r0<br>R1 = *(r0 + 4)<br>R1 = *(r0 + 4); r0 += 4<br>R1 = *r0; r0 += 4 |
| Str | 把数据从寄存器保存的内存 | Str r0, addr<br>Str r1, [r0]<br>Str r1, [r0, #4]<br>Str r1, [r0, #4]!<br>Str r1, [r0], #4 | *addr = r0<br>*r0 = r1<br>*(r0 + 4) = r1<br>*(r0 + 4) = r1; r0 += 4<br>*r0 = r1; r0 += 4 |
| Ldm | 把数据从内存加载的寄存器 | Ldmfd sp!, {r0-r12, lr} | 把寄存器的值放到慢递减栈中 |
| Stm | 把数据从寄存器保存的内存 | Stmfd sp!, {r0-r12, lr} | 从慢递减栈中把值取到寄存器 |

| | | | |
|---|---|---|---|
| Push | 压栈 | Push {r0-r12, lr} | 把寄存器的值放到慢递减栈中 |
| Pop | 出栈 | Pop {r0-r12, lr} | 从慢递减栈中把值取到寄存器 |
| b | 跳转 | B lable | 跳到 lable 处执行 |
| Bl | 跳转并保存返回地址 | Bl lable | 保存下一条指令的地址到 lr，并跳转到 lable 处执行 |
| Bx | 跳转（可切换状态） | Bx r0 | 跳转到 r0 所指的位置执行 |
| Clz | 计算一个数值高位零的个数 | Clz r0, r1 | 计算 r1 中开头的零的个数，把计算结果放到 r0 |
| Qadd | 饱和加法 | Qadd r0, r0, r1 | 运算结果的饱和到[0x80000000,0x7fffffff] |
| Qadd8 | 饱和 8 位加法 | Qadd8 r0, r0, r1 | r1 和 r2 的每一个字节分别相加，饱和到[-2^7, 2^7-1]<br>注意：每一个操作数都是寄存器，不影响 Q 位 |
| Qadd16 | 饱和 16 位加法 | Qadd16 r0, r0, r1 | r1 和 r2 的每一个 16 位相加，饱和到[-2^15, 2^15-1]<br>注意：每一个操作数都是寄存器，不影响 Q 位 |
| Qsub | 饱和减法 | Qsub r0, r0, r1 | 运算的结果饱和到[0x80000000,0x7fffffff] |
| Qsub16 | 饱和 16 位减法 | Qsub16 r0, r0, r1 | r1 和 r2 的每一个 16 位分别相减，饱和到[-2^15, 2^15-1]<br>注意：每一个操作数都是寄存器 ，不影响 Q 位 |
| Qsub8 | 饱和 8 位减法 | Qsub8 r0, r0, r1 | r1 和 r2 的每一个字节分别相减，饱和到[-2^7, 2^7-1]<br>注意：每一个操作数都是寄存器 ，不影响 Q 位 |
| Ssat | 有符号饱和 | Ssat r0, #sat, r1 | 1<=sat<=32<br>把 r1 饱和到[-2^(sat-1), 2^(sat-1)-1]，结果放到 r0，如果饱和会置位 Q<br>注意：把 r1 饱和到 sat 个位，饱和后符号不变 |
| Ssat16 | 有符号 16 位饱和 | Ssat16 r0, #sat, r1 | 1<=sat<=16<br>把 r1 中的每一个 16 位饱和到[-2^(sat-1), 2^(sat-1)-1]，结果放到 r0，如果饱和会置位 Q<br>注意：把 r1 中的每一个 16 位饱和到 sat 个位，饱和后符号不变 |
| Usat | 无符号饱和 | Usat r0, #sat, r1 | 0<=sat<=31<br>把 r1 饱和到[0, 2^sat-1]，结果放到 r0，如果饱和会置位 Q<br>注意：负数饱和到 0 |
| Usat16 | 无符号 16 位饱和 | Usat16 r0, #sat, r1 | usat16 r0,#sat,r1<br>0<=sat<=31<br>把 r1 中的两个 16 位分别饱和到[0, 2^sat-1]，结果放到 r0，如果饱和会置位 Q<br>注意：负数饱和到 0 |
| Rev | 大小端转换 | Rev r0, r1 | 把 r1 进行大小端转换，结果放到 r0 |

| | | | |
|---|---|---|---|
| Rev16 | 16 位大小端转换 | Rev16 r0, r1 | 把 r1 的每个 16 位进行大小端转换，结果放到 r0 |
| Revsh | 16 位大小端转换并有符号扩展 | Revsh r0, r1 | 把 r1 的低 16 位进行大小端转换，并扩展为一个 32 位的有符号数 |
| Rbit | 位反转 | Rbit r0, r1 | 把 r1 进行位顺序翻转，结果放到 r0 |
| Uxtb16 | 无符号 8 位扩展 16 位 | Uxtb16 r0，r1 | 把 r1 中的两个 8 位数无符号扩展为两个 16 位数，结果放到 r0 |
| Uxtb | 无符号 8 位扩展 32 位 | Uxtb8 r0，r1 | 把 r1 中的 8 位数无符号扩展为一个 32 位数，结果放到 r0 |
| Uxth | 无符号 16 位扩展 32 位 | Uxth r0，r1 | 把 r1 中的 16 位数无符号扩展为一个 32 位数，结果放到 r0 |
| Sxtb16 | 有符号 8 位扩展 16 位 | Sxtb16 r0，r1 | 把 r1 中的两个 8 位数有符号扩展为两个 16 位数，结果放到 r0 |
| Sxtb | 有符号 8 位扩展 32 位 | Sxtb8 r0，r1 | 把 r1 中的 8 位数有符号扩展为一个 32 位数，结果放到 r0 |
| Sxth | 有符号 16 位扩展 32 位 | Sxth r0，r1 | 把 r1 中的 16 位数有符号扩展为一个 32 位数，结果放到 r0 |

注意：这里并不是 arm 指令集的全部，只是 arm 指令集中比较常用的指令，如想查看全部指令集，请到 arm 官方网站自行下载：www.arm.com