[指令系统 2](#_Toc505266436)

[16位数据操作 2](#_Toc505266437)

[转移指令 3](#_Toc505266438)

[存储器数据传送指令 4](#_Toc505266439)

[其他 5](#_Toc505266440)

[32位数据操作指令 6](#_Toc505266441)

[32位存储器数据传送指令 9](#_Toc505266442)

[32位转移指令 10](#_Toc505266443)

[其它32位指令 12](#_Toc505266444)

# 指令系统

## 16位数据操作

|  |  |
| --- | --- |
| ADC | 带进位加法 |
| ADD | 加法 |
| AND | 按位与（原文为逻辑与，有误——译注）。这里的按位与和C的”&”功能相同 |
| ASR | 算术右移 |
| BIC | 按位清0（把一个数跟另一个无符号数的反码按位与） |
| CMN | 负向比较（把一个数跟另一个数据的二进制补码相比较） |
| CMP | 比较（比较两个数并且更新标志） |
| CPY | 把一个寄存器的值拷贝到另一个寄存器中 |
| EOR | 近位异或 |
| LSL | 逻辑左移（如无其它说明，所有移位操作都可以一次移动最多31格译注） |
| LSR | 逻辑右移 |
| MOV | 寄存器加载数据，既能用于寄存器间的传输，也能用于加载立即数 |
| MUL | 乘法 |
| MVN | 加载一个数的NOT值（取到逻辑反的值） |
| NEG | 取二进制补码 |
| ORR | 按位或（原文为逻辑或，有误译注） |
| ROR | 圆圈右移 |
| SBC | 带借位的减法 |
| SUB | 减法 |
| TST | 测试（执行按位与操作，并且根据结果更新Z） |
| REV | 在一个32位寄存器中反转字节序 |
| REVH | 把一个32位寄存器分成两个16位数，在每个16位数中反转字节序 |
| REVSH | 把一个32位寄存器的低16位半字进行字节反转，然后带符号扩展到32位 |
| SXTB | 带符号扩展一个字节到32位 |
| SXTH | 带符号扩展一个半字到32位 |
| UXTB | 无符号扩展一个字节到32位 |
| UXTH | 无符号扩展一个半字到32位 |

## 转移指令

|  |  |
| --- | --- |
| B | 无条件转移 |
| B<cond> | 条件转移 |
| BL | 转移并连接。用于呼叫一个子程序，返回地址被存储在LR中 |
| BLX #im | 使用立即数的BLX不要在CM3中使用 |
| CBZ | 比较，如果结果为0就转移（只能跳到后面的指令译注） |
| CBNZ | 比较，如果结果非0就转移（只能跳到后面的指令译注） |
| IT | if then |

## 存储器数据传送指令

|  |  |
| --- | --- |
| LDR | 从存储器中加载字到一个寄存器中 |
| LDRH | 从存储器中加载半字到一个寄存器中 |
| LDRB | 从存储器中加载字节到一个寄存器中 |
| LDRSH | 从存储器中加载半字，再经过带符号扩展后存储一个寄存器中 |
| LDRSB | 从存储器中加载字节，再经过带符号扩展后存储一个寄存器中 |
| STR | 把一个寄存器按字存储到存储器中 |
| STRH | 把一个寄存器存器的低半字存储到存储器中 |
| STRB | 把一个寄存器的低字节存储到存储器中 |
| LDMIA | 加载多个字，并且在加载后自增基址寄存器 |
| STMIA | 存储多个字，并且在存储后自增基址寄存器 |
| PUSH | 压入多个寄存器到栈中 |
| POP | 从栈中弹出多个值到寄存器中 |

|  |  |
| --- | --- |
| LDRB Rd,[Rn, #offset] | 从地址Rn+offset处读取一个字节送到Rd |
| LDRH Rd,[Rn, #offset] | 从地址Rn+offset处读取一个半字送到Rd |
| LDR Rd,[Rn, #offset] | 从地址Rn+offset处读取一个字送到Rd |
| LDRD Rd1,Rd2, [Rn, #offset] | 从地址Rn+offset处读取一个双字(64位整数)送到Rd1(低32位)和Rd2(高32位)中 |

|  |  |
| --- | --- |
| STRB Rd,[Rn, #offset] | 把Rd中的低字节存储到地址Rn+offset处 |
| STRH Rd,[Rn, #offset] | 把Rd中的低半字存储到地址Rn+offset处 |
| STR Rd,[Rn, #offset] | 把Rd中的低字存储到地址Rn+offset处 |
| STRD Rd1, Rd2,[Rn, #offset] | 把Rd1（低32位）和Rd2（高32位）表达的双字存储到地址Rn+offset处 |

|  |  |
| --- | --- |
| LDMIA Rd!, {寄存器列表} | 从Rd处读取多个字，并依次送到寄存器列表中的寄存器。每读一个字后Rd自增一次，16位宽度 |
| STMIA Rd!, {寄存器列表} | 依次存储寄存器列表中各寄存器的值到Rd给出的地址。每存一个字后Rd自增一次，16位宽度 |
| LDMIA.W Rd!, {寄存器列表} | 从Rd处读取多个字，并依次送到寄存器列表中的寄存器。每读一个字后Rd自增一次，32位宽度 |
| LDMDB.W Rd!, {寄存器列表} | 从Rd处读取多个字，并依次送到寄存器列表中的寄存器。每读一个字前Rd自减一次，32位宽度 |
| STMIA.W Rd!, {寄存器列表} | 依次存储寄存器列表中各寄存器的值到Rd给出的地址。每存一个字后Rd自增一次，32位宽度 |
| STMDB.W Rd!, {寄存器列表} | 存储多个字到Rd处。每存一个字前Rd自减一次，32位宽度 |

LDM/STM来鲸吞。它们相当于把若干个LDR/STR给合并起来了，有利于减少代码量

## 其他

|  |  |
| --- | --- |
| SVC | 系统服务调用 |
| BKPT | 断点指令。如果使能了调试，则进入调试状态（停机）。否则的话产生调试监视器异常。在调试监视器异常被使能时，调用其服务例程；如果连调试监视器异常也被除能，则无奈下只好诉诸于一个fault异常 |
| NOP | 无操作 |
| CPSIE | 使能PRIMASK(CPSIE i)/ FAULTMASK(CPSIE f)清0相应的位 |
| CPSID | 除能PRIMASK(CPSID i)/ FAULTMASK(CPSID f)置位相应的位 |

## 32位数据操作指令

|  |  |
| --- | --- |
| ADC | 带进位加法 |
| ADD | 加法 |
| ADDW | 宽加法（可以加12位立即数） |
| AND | 按位与（原文是逻辑与，有误。对应C言的“|”运算符译注） |
| ASR | 算术右移 |
| BIC | 位清零（把一个数按位取反后，与另一个数逻辑与） |
| BFC | 位段清零 |
| BFI | 位段插入 |
| CMN | 负向比较（把一个数和另一个数的二进制补码比较，并更新标志位） |
| CMP | 比较两个数并更新标志位 |
| CLZ | 计算前导零的数目 |
| EOR | 按位异或 |
| LSL | 逻辑左移 |
| LSR | 逻辑右移 |
| MLA | 乘加 |
| MLS | 乘减 |
| MOVW | 把16位立即数放到寄存器的低16位，高16位清0 |
| MOV | 加载16位立即数到寄存器（其实汇编器会产生MOVW译注） |
| MOVT | 把16位立即数放到寄存器的高16位，低16位不影响 |
| MVN | 移动一个数的补码 |
| MUL | 乘法 |
| ORR | 按位或（原文为逻辑或，有误译注） |
| ORN | 把源操作数按位取反后，再执行按位或（原文为逻辑或，有误译注） |
| RBIT | 位反转（把一个32位整数用2进制表达后，再旋转180度译注） |
| REV | 对一个32位整数按字节反转 |
| REVH/  REV16 | 对一个32位整数的高低半字都执行字节反转 |
| REVSH | 对一个32位整数的低半字执行字节反转，再带符号扩展成32位数 |
| ROR | 圆圈右移 |
| RRX | 带进位位的逻辑右移一格（最高位用C填充，执行后不影响C的值译注） |
| SFBX | 从一个32位整数中提取任意长度和位置的位段，并且带符号扩展成32位整数 |
| SDIV | 带符号除法 |
| SMLAL | 带符号长乘加（两个带符号的32位整数相乘得到64位的带符号积，再把积加到另一个带符号64位整数中） |
| SMULL | 带符号长乘法（两个带符号的32位整数相乘得到64位的带符号积） |
| SSAT | 带符号的饱和运算 |
| SBC | 带借位的减法 |
| SUB | 减法 |
| SUBW | 宽减法，可以减12位立即数 |
| SXTB | 字节带符号扩展到32位数 |
| TEQ | 测试是否相等（对两个数执行异或，更新标志但不存储结果） |
| TST | 测试（对两个数执行按位与，更新Z标志但不存储结果） |
| UBFX | 无符号位段提取 |
| UDIV | 无符号除法 |
| UMLAL | 无符号长乘加（两个无符号的32位整数相乘得到64位的无符号积，再把积加到另一个无符号64位整数中） |
| UMULL | 无符号长乘法（两个无符号的32位整数相乘得到64位的无符号积） |
| USAT | 无符号饱和操作（但是源操作数是带符号的译注） |
| UXTB | 字节被无符号扩展到32位（高24位清0译注） |
| UXTH | 半字被无符号扩展到32位（高16位清0译注） |

## 32位存储器数据传送指令

|  |  |
| --- | --- |
| LDR | 加载字到寄存器 |
| LDRB | 加载字节到寄存器 |
| LDRH | 加载半字到寄存器 |
| LDRSH | 加载半字到寄存器，再带符号扩展到32位 |
| LDM | 从一片连续的地址空间中加载若干个字，并选中相同数目的寄存器放进去 |
| LDRD | 从连续的地址空间加载双字（64位整数）到2个寄存器 |
| STR | 存储寄存器中的字 |
| STRB | 存储寄存器中的低字节 |
| STRH | 存储寄存器中的低半字 |
| STM | 存储若干寄存器中的字到一片连续的地址空间中，占用相同数目的字 |
| STRD | 存储2个寄存器组成的双字到连续的地址空间中 |
| PUSH | 把若干寄存器的值压入堆栈中 |
| POP | 从堆栈中弹出若干的寄存器的值 |

## 32位转移指令

|  |  |
| --- | --- |
| B | 无条件转移 |
| BL | 转移并连接（呼叫子程序） |
| TBB | 以字节为单位的查表转移。从一个字节数组中选一个8位前向跳转地址并转移 |
| TBH | 以半字为单位的查表转移。从一个半字数组中选一个16位前向跳转的地址并转移 |

## 其它32位指令

|  |  |
| --- | --- |
| LDREX | 加载字到寄存器，并且在内核中标明一段地址进入了互斥访问状态 |
| LDREXH | 加载半字到寄存器，并且在内核中标明一段地址进入了互斥访问状态 |
| LDREXB | 加载字节到寄存器，并且在内核中标明一段地址进入了互斥访问状态 |
| STREX | 检查将要写入的地址是否已进入了互斥访问状态，如果是则存储寄存器的字 |
| STREXH | 检查将要写入的地址是否已进入了互斥访问状态，如果是则存储寄存器的半字 |
| STREXB | 检查将要写入的地址是否已进入了互斥访问状态，如果是则存储寄存器的字节 |
| CLREX | 在本地处理器上清除互斥访问状态的标记（先前由LDREX/LDREXH/LDREXB做的标记） |
| MRS | 加载特殊功能寄存器的值到通用寄存器 |
| MSR | 存储通用寄存器的值到特殊功能寄存器 |
| NOP | 无操作 |
| SEV | 发送事件 |
| WFE | 休眠并且在发生事件时被唤醒 |
| WFI | 休眠并且在发生中断时被唤醒 |
| **ISB** | 指令同步隔离（与流水线和MPU等有关译注） |
| **DSB** | 数据同步隔离（与流水线、MPU和cache等有关译注） |
| **DMB** | 数据存储隔离（与流水线、MPU和cache等有关译注） |