

# ARM Instructions










# B

- B.1 **Data-processing Instructions**
- B.2 **Memory Instructions**
- B.3 **Branch Instructions**
- B.4 **Miscellaneous Instructions**
- B.5 **Condition Flags**

This appendix summarizes ARMv4 instructions used in this book. Condition encodings are given in Table 6.3.

## B.1 DATA-PROCESSING INSTRUCTIONS

Standard data-processing instructions use the encoding in Figure B.1. The 4-bit *cmd* field specifies the type of instruction as given in Table B.1. When the *S*-bit is 1, the status register is updated with the condition flags produced by the instruction. The *I*-bit and bits 4 and 7 specify one of three encodings for the second source operand, *Src2*, as described in Section 6.4.2. The *cond* field specifies which condition codes to check, as given in Section 6.3.2.

Application Software	
Operating Systems	
Architecture	
Micro-architecture	
Logic	
Digital Circuits	
Analog Circuits	
Devices	
Physics	

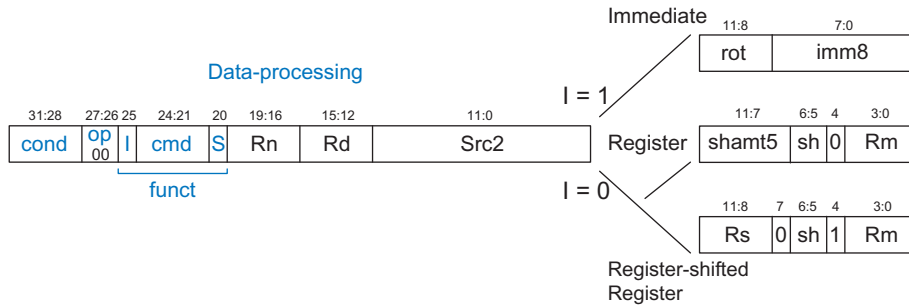


Figure B.1 Data-processing instruction encodings

Table B.1 Data-processing instructions

cmd	Name	Description	Operation
0000	AND Rd, Rn, Src2	Bitwise AND	$Rd \leftarrow Rn \& Src2$
0001	EOR Rd, Rn, Src2	Bitwise XOR	$Rd \leftarrow Rn \wedge Src2$
0010	SUB Rd, Rn, Src2	Subtract	$Rd \leftarrow Rn - Src2$
0011	RSB Rd, Rn, Src2	Reverse Subtract	$Rd \leftarrow Src2 - Rn$
0100	ADD Rd, Rn, Src2	Add	$Rd \leftarrow Rn + Src2$
0101	ADC Rd, Rn, Src2	Add with Carry	$Rd \leftarrow Rn + Src2 + C$
0110	SBC Rd, Rn, Src2	Subtract with Carry	$Rd \leftarrow Rn - Src2 - \bar{C}$
0111	RSC Rd, Rn, Src2	Reverse Sub w/ Carry	$Rd \leftarrow Src2 - Rn - \bar{C}$
1000 ( $S = 1$ )	TST Rd, Rn, Src2	Test	Set flags based on $Rn \& Src2$
1001 ( $S = 1$ )	TEQ Rd, Rn, Src2	Test Equivalence	Set flags based on $Rn \wedge Src2$
1010 ( $S = 1$ )	CMP Rn, Src2	Compare	Set flags based on $Rn - Src2$
1011 ( $S = 1$ )	CMN Rn, Src2	Compare Negative	Set flags based on $Rn + Src2$
1100	ORR Rd, Rn, Src2	Bitwise OR	$Rd \leftarrow Rn   Src2$
1101	<b>Shifts:</b>		
$I = 1$ OR ( $instr_{11:4} = 0$ )	MOV Rd, Src2	Move	$Rd \leftarrow Src2$
$I = 0$ AND ( $sh = 00$ ; $instr_{11:4} \neq 0$ )	LSL Rd, Rm, Rs/shamt5	Logical Shift Left	$Rd \leftarrow Rm \ll Src2$
$I = 0$ AND ( $sh = 01$ )	LSR Rd, Rm, Rs/shamt5	Logical Shift Right	$Rd \leftarrow Rm \gg Src2$

(continued)

Table B.1 Data-processing instructions—Cont'd

cmd	Name	Description	Operation
$I = 0$ AND ( $sh = 10$ )	ASR Rd, Rm, Rs/shamt5	Arithmetic Shift Right	$Rd \leftarrow Rm \gg \gg Src2$
$I = 0$ AND ( $sh = 11$ ; $instr_{11:7, 4} = 0$ )	RRX Rd, Rm, Rs/shamt5	Rotate Right Extend	$\{Rd, C\} \leftarrow \{C, Rd\}$
$I = 0$ AND ( $sh = 11$ ; $instr_{11:7} \neq 0$ )	ROR Rd, Rm, Rs/shamt5	Rotate Right	$Rd \leftarrow Rn \text{ ror } Src2$
1110	BIC Rd, Rn, Src2	Bitwise Clear	$Rd \leftarrow Rn \& \sim Src2$
1111	MVN Rd, Rn, Src2	Bitwise NOT	$Rd \leftarrow \sim Rn$

NOP (no operation) is typically encoded as 0xE1A000, which is equivalent to MOV R0, R0.

### B.1.1 Multiply Instructions

Multiply instructions use the encoding in Figure B.2. The 3-bit *cmd* field specifies the type of multiply, as given in Table B.2.

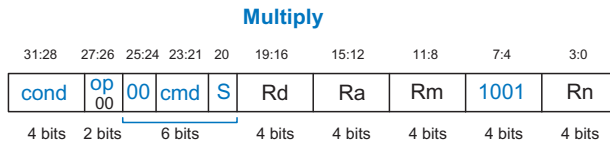


Figure B.2 Multiply instruction encoding

Table B.2 Multiply instructions

cmd	Name	Description	Operation
000	MUL Rd, Rn, Rm	Multiply	$Rd \leftarrow Rn \times Rm$ (low 32 bits)
001	MLA Rd, Rn, Rm, Ra	Multiply Accumulate	$Rd \leftarrow (Rn \times Rm) + Ra$ (low 32 bits)
100	UMULL Rd, Rn, Rm, Ra	Unsigned Multiply Long	$\{Rd, Ra\} \leftarrow Rn \times Rm$ (all 64 bits, Rm/Rn unsigned)
101	UMLAL Rd, Rn, Rm, Ra	Unsigned Multiply Accumulate Long	$\{Rd, Ra\} \leftarrow (Rn \times Rm) + \{Rd, Ra\}$ (all 64 bits, Rm/Rn unsigned)
110	SMULL Rd, Rn, Rm, Ra	Signed Multiply Long	$\{Rd, Ra\} \leftarrow Rn \times Rm$ (all 64 bits, Rm/Rn signed)
111	SMLAL Rd, Rn, Rm, Ra	Signed Multiply Accumulate Long	$\{Rd, Ra\} \leftarrow (Rn \times Rm) + \{Rd, Ra\}$ (all 64 bits, Rm/Rn signed)

## B.2 MEMORY INSTRUCTIONS

The most common memory instructions (LDR, STR, LDRB, and STRB) operate on words or bytes and are encoded with  $op = 01$ . Extra memory instructions operating on halfwords or signed bytes are encoded with  $op = 00$  and have less flexibility generating  $Src2$ . The immediate offset is only 8 bits and the register offset cannot be shifted. LDRB and LDRH zero-extend the bits to fill a word, while LDRSB and LDRSH sign-extend the bits. Also see memory indexing modes in Section 6.3.6.

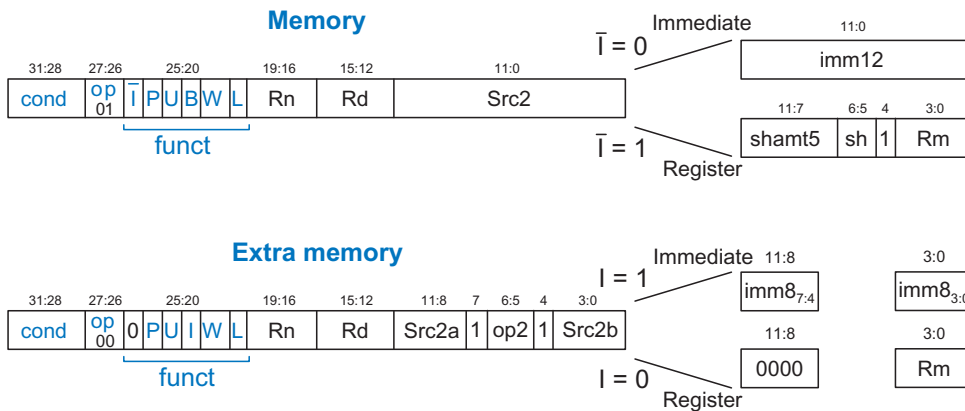


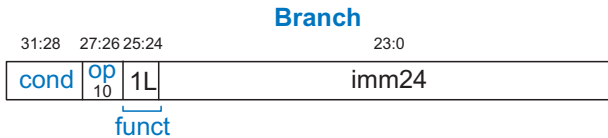
Figure B.3 Memory instruction encodings

Table B.3 Memory instructions

op	B	op2	L	Name	Description	Operation
01	0	N/A	0	STR	Rd, [Rn, ±Src2]	Mem[Adr] ← Rd
01	0	N/A	1	LDR	Rd, [Rn, ±Src2]	Rd ← Mem[Adr]
01	1	N/A	0	STRB	Rd, [Rn, ±Src2]	Mem[Adr] ← Rd <sub>7:0</sub>
01	1	N/A	1	LDRB	Rd, [Rn, ±Src2]	Rd ← Mem[Adr] <sub>7:0</sub>
00	N/A	01	0	STRH	Rd, [Rn, ±Src2]	Mem[Adr] ← Rd <sub>15:0</sub>
00	N/A	01	1	LDRH	Rd, [Rn, ±Src2]	Rd ← Mem[Adr] <sub>15:0</sub>
00	N/A	10	1	LDRSB	Rd, [Rn, ±Src2]	Rd ← Mem[Adr] <sub>7:0</sub>
00	N/A	11	1	LDRSH	Rd, [Rn, ±Src2]	Rd ← Mem[Adr] <sub>15:0</sub>

## B.3 BRANCH INSTRUCTIONS

Figure B.4 shows the encoding for branch instructions (B and BL) and Table B.4 describes their operation.



**Figure B.4** Branch instruction encoding

**Table B.4** Branch instructions

L	Name	Description	Operation
0	B label	Branch	$PC \leftarrow (PC+8)+imm24 \ll 2$
1	BL label	Branch with Link	$LR \leftarrow (PC+8) - 4; PC \leftarrow (PC+8)+imm24 \ll 2$

## B.4 MISCELLANEOUS INSTRUCTIONS

The ARMv4 instruction set includes the following miscellaneous instructions. Consult the ARM Architecture Reference Manual for details.

Instructions	Description	Purpose
LDM, STM	Load/store multiple	Save and recall registers in subroutine calls
SWP / SWPB	Swap (byte)	Atomic load and store for process synchronization
LDRT, LDRBT, STRT, STRBT	Load/store word/byte with translation	Allow operating system to access memory in user virtual memory space
SWI <sup>1</sup>	Software Interrupt	Create an exception, often used to call the operating system
CDP, LDC, MCR, MRC, STC	Coprocessor access	Communicate with optional coprocessor
MRS, MSR	Move from/to status register	Save status register during exceptions

<sup>1</sup> SWI was renamed SVC (supervisor call) in ARMv7.

## B.5 CONDITION FLAGS

Condition flags are changed by data-processing instructions with  $S = 1$  in the machine code. All instructions except `CMP`, `CMN`, `TEQ`, and `TST` must have an “S” appended to the instruction mnemonic to make  $S = 1$ . [Table B.5](#) shows which condition flags are affected by each instruction.

**Table B.5** Instructions that affect condition flags

Type	Instructions	Condition Flags
Add	ADDS, ADCS	N, Z, C, V
Subtract	SUBS, SBCS, RSBS, RSCS	N, Z, C, V
Compare	CMP, CMN	N, Z, C, V
Shifts	ASRS, LSLS, LSRS, RORS, RRXS	N, Z, C
Logical	ANDS, ORRS, EORS, BICS	N, Z, C
Test	TEQ, TST	N, Z, C
Move	MOVS, MVNS	N, Z, C
Multiply	MULS, MLAS, SMLALS, SMULLS, UMLALS, UMULLS	N, Z