

APPENDIX

ARM and Thumb Instruction Encodings

Andrew Sloss, ARM; Dominic Symes, ARM;

Chris Wright, Ultimodule Inc.

- **B2.1** ARM Instruction Set Encodings B-3
- **B2.2** Thumb Instruction Set Encodings B-9
- **B2.3** Program Status Registers B-11

This appendix gives tables for the instruction set encodings of the 32-bit ARM and 16-bit Thumb instruction sets. We also describe the fields of the processor status registers *cpsr* and *spsr*.

B2.1

ARM Instruction Set Encodings

Table B2.1 summarizes the bit encodings for the 32-bit ARM instruction set architecture ARMv6. This table is useful if you need to decode an ARM instruction by hand. We've expanded the table to aid quick manual decode. Any bitmaps not listed are either unpredictable or undefined for ARMv6.

To use Table B2.1 efficiently, follow this decoding procedure:

■ Look at the leading hex digit of the instruction, bits 28 to 31. If this has a value 0xF, then jump to the end of Table B2.1. Otherwise, the top hex digit represents a condition *cond*. Decode *cond* using Table B2.2.

TABLE B2.1 ARM instruction decode table.

(d_i)
by σ
indexed l
classes (
Instruction

Instruction classes (indexed by op)	31 30 29 28	28 27 3	26 2	25 24	24 23	22 2	21 20	19	18 17 16	16 15 14 13 12 11 10	11 10 9 8 7	6 5	4	3 2 1 0
AND EOR SUB RSB ADD ADC SBC RSC	cond	0) 0	0 0		ф	S		Rn	Rd	shift_size	shift	0	Rm
EOR SUB ADC SBC	риоэ	0	0	0 0		do	S	·	Rn	Rd	Rs	0 shift	1 1	Rm
-	риоэ	0	0	0 0	0	0	S 0		Rd	0 0 0 0	Rs	0 0 1	1	Rm
MLA	cond	0	0	0 0	0	0	S		Rd	Rn	Rs	0 0 1		Rm
_	cond	0	0	0 0	0	1	0 0		RdHi	RdLo	Rs	0 0 1	1	Rm
	риоэ	0	0	0 0	1	do	S		RdHi	RdLo	Rs	0 0 1	П	Rm
STRH LDRH post	cond	0	0	0 0	D	0	do 0		Rn	Rd	0 0 0 0	1 0 1		Rm
	риоэ	0) 0	0 0	U	1	do 0		Rn	Rd	immed [7:4]	1 0 1	1	immed [3:0]
LDRD STRD LDRSB LDRSH post	сопд	0	0 (0 0	U	0	0 op		Rn	Rd	0 0 0 0	1 1 op) 1	Rm
LDRD STRD LDRSB LDRSH post	сопа	0	0	0 0	U	1 0	do 0		Rn	Rd	immed [7:4]	1 op	, 1	immed [3:0]
MRS Rd, cpsr MRS Rd, spsr	сопд	0	0	0 1	0) <i>do</i>	0 0	1	1 1	Rd	0 0 0 0 0	0	0 0 0	0000
MSR cpsr, Rm MSR spsr, Rm	cond	0	0	0 1	0	op 1	0 1	f	s x c	1 1 1 1	0 0 0 0 0	0 0 (0	Rm
ВХЈ	cond	0	0	0 1	0	0 1	0 1	1 1	1 1	1 1 1 1	1 1 1 1 0	0 1	0	Rm
SMLAxy	сопд	0	0 (0 1	0	0 (0 (Rd	Rn	Rs	1 y x	0	Rm
SMLAWy	cond	0	0	0 1	0	0 1	0 1		Rd	Rn	Rs	1 y 0	0	Rm
SMULWy	риоэ	0	0	0 1	0	0 1	0 1		Rd	0 0 0 0	Rs	l y 1	0	Rm
SMLALxy	сопд	0	0 (0 1	0	1 (0 0		RdHi	RdLo	Rs	y x	0	Rm
SMULxy	cond	0	0	0 1	0	1	0 1		Rd	0 0 0 0	Rs	1 y x	0	Rm
TST TEQ CMP CMN	риоэ	0	0	0 1	0	до	1		Rn	0 0 0 0	shift_size	shift	0	Rm
ORR BIC	cond	0	0	0 1	П) <i>do</i>	S		Rn	Rd	shift_size	shift	0	Rm
MOV MVN	cond	0	0	0 1	1	<i>op</i> 1	S 1	0	0 0 0	Rd	shift_size	shift	0	Rm
_	cond	0		0 1	0	0	0 1	1	1 1	1 1 1 1	1 1 1 1	do 0 0	, 1	Rm
CLZ	cond	0	0	0	0	_	0	_	1	Rd	1 1 1 1 0	1 0 0 0	П	Rm
QADD QSUB QDADD QDSUB	cond	0	0	0 1	0	до	0		Rn	Rd	0 0 0 0	0 1 0	П	Rm
	1 1 1 0	0	0 (0 1	0	0 1	0 1			immed [15:4]		0 1 1	1	<i>immed</i> [3:0]
TST TEQ CMP CMN	cond	0	0	0 1	0	Об	1		Rn	0 0 0 0	Rs	0 shift	<i>t</i> 1	Rm
	cond	0	0	0 1	-	0 do	S		Rn	Rd		0 shift	<i>t</i> 1	Rm
	cond	0	0	0		op 1	S	0	0 0 0	Rd	Rs	0 shift	1	Rm
	cond	0	0	0 1	0) <i>do</i>	0 0		Rn	Rd	0 0 0 0	0 0 1	1	Rm
STREX	cond	0		0 1	-	0	0 0		Rn	Rd	1 1 1 1	0 0 1	-	Rm
LDREX	cond	0	0	0 1		0	0 1		Rn	Rd	1 1 1 1	0 0 1		1 1 1 1

 Rm

 Rm

Rm Rm Rm

TABLE B2.1 ARM instruction decode table. (Continued.)

(nstruction classes (indexed by op)	

1 [3:0] 1 | Rm 1 | immed 1 [3:0]

W op Rn Rd 0 0 0 0 1 0 1 1 R	W op Rn Rd immed 1 0 1 1 imm [3.4]	W op Rn Rd 0 0 0 0 1 1 op 1 R	$W op \qquad Rn \qquad Rd \qquad immed \qquad 1 1 op 1 imn [3.3]$	S Rn Rd rotate innned	$\begin{vmatrix} 1 & 0 & f & s & x & c \end{vmatrix}$	1 Rn 0 0 0 0 rotate immed) S Rn Rd rotate immed	1 S 0 0 0 0 Rd rotate immed	T op Rn Rd immed12	W op Rn Rd immed12	T op Rn Rd shift_size shift 0 R	op Rn Rd 1 1 1 1 0 0 0 1 R	op Rn Rd 1 1 1 1 0 0 1 1 R	op Rn Rd 1 1 1 1 0 1 0 1 1 R	op Rn Rd 1 1 1 1 1 0 1 1 1 1 R	op Rn Rd 1 1 1 1 1 1 0 0 1 R	Rn Rd 1 1 1 1 1 1 1 1 1	$0 0 Rn Rd shift_size op 0 1 R$	1	$oxed{1 \ 0 \ immed 4} \ oxed{Rd} \ oxed{Rd} \ oxed{1 \ 1 \ 1 \ 1 \ 1} \ oxed{0 \ 0 \ 1} \ oxed{1} \ oxed{R}$	0 0 Rn Rd 1 1 1 1 1 1 1 1 R	1 1 1 1 0p 0 1 1	$0 \ 0 \ Rn! = 1111 \ Rd \ rot \ 0 \ 0 \ 1 \ 1 \ 1 \ R$	0 0 1 1 1 1 1 Rd rot 0 0 0 1 1 1 R	1 0 Rn!=1111 Rd rot 0 0 0 1 1 1 R	1 0 1 1 1 1 Rd rot 0 0 0 1 1 1 R	1 1 Rn!=1111 Rd rot 0 0 0 1 1 1 R	1 1 1 1 1 1 Rd rot 0 0 0 1 1 1 R	W op Rn Rd shift_size shift 0 R	Rn'=1111 Rs 0 op X 1	$egin{array}{ c c c c c c c c c c c c c c c c c c c$	H THE THE COLUMN
U 0 V	U 1	U = V	U 1 V	do	0 op 1	<i>do</i> 0	1 op (1 op 1	I do D	A do A	O op	0	0	0	0	0	0	1 0	1 op 1	1 op 1	1 0	1 op 1	1 op	1 op (-	-	_	-			0 0 0	,
0 0 0 1	0 0 0 1	0 0 0 1	0 0 0 1	0 0 1 0	0 0 1 1	0 0 1 1	0 0 1 1	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 0	0 1 1 0	0 1 1 0	0 1 1 0	0 1 1 0	0 1 1 0	0 1 1 0	0 1 1 0	0 1 1 0	0 1 1 0	0 1 1 0	0 1 1 0	0 1 1 0	0 1 1 0	0 1 1 0	0 1 1 0	0 1 1 0	0 1 1 1	0 1 1 1	0 1 1 1	-
cond	cond	cond	cond	cond	риоэ	cond	cond	сопд	сопд	cond	cond	сопд	cond	cond	cond	cond	сопд	cond	cond	сопд	сопд	cond	cond	сопд	сопд	cond	cond	сопд	cond	cond	cond	1
STRH LORH pre	- —	LDRD STRD LDRSB LDRSH <i>pre</i>	LDRD STRD LDRSB LDRSH <i>pre</i>	AND EOR SUB RSB ADD ADC SBC RSC	MSR cpsr, #imm MSR spsr, #imm	TST TEQ CMP CMN	ORR BIC	MOV MVN	STR LDR STRB LDRB post	STR LDR STRB LDRB pre	STR LDR STRB LDRB post	{ S Q SH U UQ UH}ADD16	{ S Q SH U UQ UH}ADDSUBX	{ S Q SH U UQ UH}SUBADDX	{ S Q SH U UQ UH}SUB16	{ S Q SH U UQ UH}ADD8	{ S Q SH U UQ UH}SUB8	PKHBT PKHTB	{S U}SAT	{S U}SAT16	SEL	REV REV16 REVSH	{S U}XTAB16	{S U}XTB16	{S U}XTAB	{S U}XTB	{S U}XTAH	{S U}XTH	STR LDR STRB LDRB pre	SMLAD SMLSD	SMUAD SMUSD	

 Rm

 Rm

 Rm

 Rm

 Rm

 Rm

 Rm

 Rm

 Rm

TABLE B2.1 ARM instruction decode table. (Continued.)

The fraction classes (indexed by op)	77 07 67 NC 16	77	77 67 47 67 07	Č Ą	C7 +		77	17 10 17 1	6 01 11 71 61 41 61 01 /1 01 61 07 17	11 10 7 0	0	J.	0	7
SMMLA SMMLS	cond	0	-		0		0 1	Rd	Rn!=1111	Rs	do	R 1	R	Rm
SMMUL	cond	0	-		0	1 0	-	Rd	1 1 1 1	Rs	0 0	R	R	Rm
USADA8	cond	0				0 0	0	Rd	Rn! = 1111	Rs	0 0	0	R	Rm
USAD8	cond	0	-			0 0	0	Rd	1 1 1 1	Rs	0 0	0	R	Rm
Undefined and expected to stay so	сопд	0	-		-	_	-		×		1 1	_	.,	×
STMDA LDMDA STMIA LDMIA	cond	-	0	0	do 0	<	M op	Rn		register_list	ist			
	cond	-	0	0 1	do	<	M op	Rn		register_list	ist			
B to instruction_address+8+4*offset	cond		0	1 0				igis	signed 24-bit branch offset	ch offset				
•—	cond	_	0					sign	signed 24-bit branch offset	ch offset				
	сопд	1	1 (0 0	0	1 0	do 1	Rn	Rd	copro)	op1		Cm
	cond		1	0	0 1	T = 0	do 1	Rn	Cd	copro		оb	option	
$STC\{L\} \mid LDC\{L\} post$	cond	-	-	0	O	L 1	do	Rn	Cd	copro		inni	immed8	
	cond		1	0	U	T M	do 1	Rn	Cd	copro		inni	immed8	
CDP	cond			1 0		ldo		Сп	Cd	copro	op2	0		Ст
MCR MRC	cond	1	1	0 1		ldo	do	Сп	Rd	copro	op2	7		Cm
SWI	cond	1	1	1 1					immed24	4				
CPS CPSIE CPSID	1 1 1 1	0	0	0	0	0 0	0	M do	0 0 0 0 0	0 0 0 0	a i f	0	mode	в
SETEND LE SETEND BE	1 1 1 1	0	0	0	0	0 0	0	0 0 0	0 0 0 0 1	do 0 0	0 0 0 0	1	0 0 0	0 0
PLD pre	1 1 1 1	0	1 (0 1	U	1 0	-	Rn	1 1 1 1		immed12	112		
	1 1 1 1	0	1	1	\bigcup	1 0	- 1	Rn	1 1 1 1	shift_size shift	ze sh	ift $ 0 $		Rm
RFEDA RFEIA RFEDB RFEIB	1 1 1 1	1	0 (0 0	do do	0	M 1	Rn	0 0 0 0	1 0 1	0 0 0 0		0 0 0	0 0 0
SRSDA SRSIA SRSDB SRSIB	1 1 1 1	1	0	0 0	do do	1	M 0	1 1 0	0 0 0 0 1	0 1 0 1	0 0 0 1	0	mode	е
BLX instruction+8+4*offset+2*a	1 1 1 1	1	0	1 a	1			sigis	signed 24-bit branch offset	ch offset				
	1 1 1 1	1	1 (0 0	0	1 (do o	Rn	Rd	copro	0	ldo		Ст
_	1 1 1 1	1	1 (0	0 1	T = 0	do 1	Rn	Cq	copro		option	ио	
$STC2\{L\} \mid LDC2\{L\} post$	1 1 1 1	1	1 (0	$0 \mid \Omega$	T]	do	Rn	Cd	copro		imn	immed8	
_	1 1 1 1	1	1 (0 1	Ω	T = M	do 1	Rn	Cd	copro		imn	immed8	
	1 1 1 1	1	1	0 1		ob1		Cn	Cq	copro	op2	2 0		Cm
MCR2 MRC2	1 1 1 1	1	1	0 1	_	ob1	do	Cn	Cq	copro	op2	2 1)	Ст

TABLE B2.2	Decoding	table	for	cond.
------------	----------	-------	-----	-------

Binary	Hex	cond
0000	0	EQ
0001	1	NE
0010	2	CS/HS
0011	3	CC/LO
0100	4	MI
0101	5	PL
0110	6	VS
0111	7	VC

Binary	Hex	cond
1000	8	HI
1001	9	LS
1010	А	GE
1011	В	LT
1100	С	GT
1101	D	LE
1110	E	{AL}

- Index through Table B2.1 using the second hex digit, bits 24 to 27 (shaded).
- Index using bit 4, then bit 7 or bit 23 of the instruction where these bits are shaded.
- Once you have located the correct table entry, look at the bits named *op*. Concatenate these to form a binary number that indexes the | separated instruction list on the left. For example if there are two *op* bits value 1 and 0, then the binary value 10 indicates instruction number 2 in the list (the third instruction).
- The instruction operands have the same name as in the instruction description of Appendix B1.

The table uses the following abbreviations:

- L is 1 if the L suffix applies for LDC and STC operations.
- *M* is 1 if CPS changes processor mode. *mode* is defined in Table B2.3.

TABLE B2.3 Decoding table for mode.

Binary	Hex	mode
10000	0x10	user mode (_usr)
10001	0x11	FIQ mode (_fiq)
10010	0x12	IRQ mode (_irq)
10011	0x13	supervisor mode (_svc)
10111	0x17	abort mode (_abt)
11011	Ox1B	undefined mode (_und)
11111	0x1F	system mode

	_	,	_ ,				
shift	shift_size	Rs	Shift action				
00	0 to 31	N/A	LSL #shift_size				
00	N/A	Rs	LSL Rs				
01	0	N/A	LSR #32				
01	1 to 31	N/A	LSR #shift_size				
01	N/A	Rs	LSR Rs				
10	0	N/A	ASR #32				
10	1 to 31	N/A	ASR #shift_size				
10	N/A	Rs	ASR Rs				
11	0	N/A	RRX				
11	1 to 31	N/A	ROR #shift_size				
11	N/A	Rs	ROR Rs				
N/A	0 to 31	N/A	The <i>shift</i> value is implicit: For PKHBT it is 00. For PKHTB it is 10. For SAT it is 2* <i>sh</i> .				

TABLE B2.4 Decoding table for shift, shift_size, and Rs.

- op1 and op2 are the opcode extension fields in coprocessor instructions.
- post indicates a postindexed addressing mode such as [Rn], Rm or [Rn], #immed.
- pre indicates a preindexed addressing mode such as [Rn, Rm] or [Rn, #immed].
- register_list is a bit field with bit k set if register Rk appears in the register list.
- rot is a byte rotate. The second operand is Rm ROR (8*rot).
- rotate is a bit rotate. The second operand is #immed ROR (2*rotate).
- *shift* and *sh* encode a shift type and direction. See Table B2.4.
- *U* is the up/down select for addressing modes. If U = 1, then we add the offset to the base address, as in [Rn], #4 or [Rn, Rm]. If U = 0 then we subtract the offset from the base address, as in [Rn, #-4] or [Rn], -Rm.
- *unindexed* indicates an addressing mode of the form [Rn],{option}.
- \blacksquare *R* is 1 if the R (round) instruction suffix is present.
- T is 1 if the \top suffix is present on load and store instructions.
- *W* is 1 if ! (writeback) is specified in the instruction mnemonic.
- \blacksquare X is 1 if the X (exchange) instruction suffix is present.
- \blacksquare x and y are 0 for the B suffix, 1 for the T suffix.
- ' is 1 if the ' suffix is applied in LDM or STM instructions.

B2.2

Thumb Instruction Set Encodings

Table B2.5 summarizes the bit encodings for the 16-bit Thumb instruction set. This table is useful if you need to decode a Thumb instruction by hand. We've expanded the table to aid quick manual decode. The table contains instruction definitions up to architecture THUMBv3. Any bitmaps not listed are either unpredictable or undefined for THUMBv3.

To use the table efficiently, follow this decoding procedure:

- Index through the table using the first hex digit of the instruction, bits 12 to 15 (shaded).
- Index on any shaded bits from bits 0 to 11.
- Once you have located the correct table entry, look at the bits named *op*. Concatenate these to form a binary number that indexes the | separated instruction list on the left. For example, if there are two *op* bits value 1 and 0, then the binary value 10 indicates instruction number 2 in the list (the third instruction).

TABLE B2.5 Thumb instruction decode table.

Instruction	ciasses	(maexea	υy	OP)

ASR ADD MOV ADD AND ASR TST ORR CPY ADD	SUB CMP SUB EOR ADC NEG MUL Ld, Lm MOV		ROR CMN	
BX LDR STR LDR STR	Ld, [po STRH LDRH LDR	LDRE Ld, [Lr 3 Ld, [ned*4] B LDRSB B LDRSH n, #immed* Ln, #imme	<i>pre</i> *4] ed]

0	0	0	0	ор	immed5					Lm	Ld			
0	0	0	1	0	immed5					Lm	Ld			
0	0	0	1	1	0 op Lr			Lm		Ln	Ld			
0	0	0	1	1	1	ор	iı	ттес	13	Ln	Ld			
0	0	1	0	ор		Ld/Lr	ı			immed8				
0	0	1	1	ор		Ld				immed8				
0	1	0	0	0	0	0	0	0	p	Lm/Ls	Ld			
0	1	0	0	0	0	0	1	0	p	Lm/Ls	Ld			
0	1	0	0	0	0	1	0	0	p	Lm	Ld/Ln			
0	1	0	0	0	0	1	1	ор		Lm	Ld			
0	1	0	0	0	1	1	0	0 0		Lm	Ld			
0	1	0	0	0	1	op	0	0 1		Hm & 7	Ld			
0	1	0	0	0	1	op	0	1 0		Lm	Hd & 7			
0	1	0	0	0	1	op	0	1	1	Hm & 7	Hd & 7			
0	1	0	0	0	1	0	1	0	1	Hm & 7	Ln			
0	1	0	0	0	1	0	1	1	0	Lm	Hn & 7			
0	1	0	0	0	1	0	1	1 1		Hm & 7	Hn & 7			
0	1	0	0	0	1	1	1	ор	op Rm		0 0 0			
0	1	0	0	1		Ld		immed8						
0	1	0	1	0	C	p	Lm			Ln	Ld			
0	1	0	1	1	op		Lm		Ln	Ld				
0	1	1	0	op	immed			15		Ln	Ld			
0	1	1	1	op	immed			15		Ln	Ld			
1	0	0	0	ор	immed5			15		Ln	Ld			

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

TABLE B2.5 Thumb instruction decode table. (Continued.)

Instruction classes (indexed by op)

SWI immed8

B instruction_address+4+offset*2
BLX ((instruction+4+
 (poff<<12)+offset*4) &~ 3)</pre>

This must be preceded by a branch prefix instruction.

This is the branch prefix instruction. It must be followed by a relative BL or BLX instruction. BL instruction+4+ (poff<<12)+ offset*2 This must be preceded by a branch prefix instruction.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	ор		Ld					imn	ıed8			
1	0	1	0	ор		Ld		immed8							
1	0	1	1	0	0	0	0	ор	op immed7						
1	0	1	1	0	0	1	0	0	p		Lm			Ld	
1	0	1	1	1	0	1	0		p		Lm			Ld	\neg
1	0	1	1	ор	1	0	R				regist	er_lis	st		
1	0	1	1	0	1	1	0	0	1	0	1	ор	0	0	0
1	0	1	1	0	1	1	0	0	1	1	ор	0	а	i	f
1	0	1	1	1	1	1	0				imm	ed8			
1	1	0	0	ор		Ln					registe	er_lis	st		
1	1	0	1	с	ond «	< 111	0	signed 8-bit offset							
1	1	0	1	1	1	1	0				2	С			\neg
1	1	0	1	1	1	1	1				imn	ied8			\neg
1	1	1	0	0				signe	ed 11	-bit	offset				
1	1	1	0	1	unsigned 10-bit <i>offset</i> 0										
1	1	1	1	0	signed 11-bit prefix offset poff										
1	1	1	1	1			τ	ınsigned 11-bit <i>offset</i>							

■ The instruction operands have the same name as in the instruction description of Appendix B1.

The table uses the following abbreviations:

- register_list is a bit field with bit k set if register Rk appears in the register list.
- \blacksquare R is 1 if lr is in the register list of PUSH or pc is in the register list of POP.



Program Status Registers

Table B2.6 shows how to decode the 32-bit program status registers for ARMv6.

TABLE B2.6 cpsr and spsr decode table.

:	31	30	29	28	3 27	26 25	24	23 22 21 20	0 19 18 17 1 6	15 14 13 12 11	10 9	876	543210
	Ν	Ζ	С	V	Q	Res	J	Res	GE[3:0]	Res	ΕA	IFT	mode

Field	Use
N	Negative flag, records bit 31 of the result of flag-setting operations.
Z	Zero flag, records if the result of a flag-setting operation is zero.
С	Carry flag, records unsigned overflow for addition, not-borrow for subtraction, and is also used by the shifting circuit. See Table B1.3.
V	Overflow flag, records signed overflows for flag-setting operations.
Q	Saturation flag. Certain operations set this flag on saturation. See for example QADD in Appendix B1 (ARMv5E and above).
J	J=1 indicates Java execution (must have $T=0$). Use the BXJ instruction to change this bit (ARMv5J and above).
Res	These bits are reserved for future expansion. Software should preserve the values in these bits.
GE[3:0]	The SIMD greater-or-equal flags. See SADD in Appendix B1 (ARMv6).
E	Controls the data endianness. See SETEND in Appendix B1 (ARMv6).
Α	A = 1 disables imprecise data aborts (ARMv6).
1	I = 1 disables IRQ interrupts.
F	F = 1 disables FIQ interrupts.
Т	T=1 indicates Thumb state. $T=0$ indicates ARM state. Use the BX or BLX instructions to change this bit (ARMv4T and above).
mode	The current processor mode. See Table B2.4.