1.4 Instruction set summary

This section provides a description of the instruction sets used on the ARM7TDMI processor.

This section describes:

- *Format summary*
- *ARM instruction summary* on page 1-13
- *Thumb instruction summary* on page 1-20.

1.4.1 Format summary

This section provides a summary of the ARM, and Thumb instruction sets:

- *ARM instruction summary* on page 1-13
- *Thumb instruction summary* on page 1-20.

A key to the instruction set tables is provided in Table 1-1.

The ARM7TDMI processor uses an implementation of the ARMv4T architecture. For a complete description of both instruction sets, see the *ARM Architecture Reference Manual*.

Table 1-1 Key to tables

Type	Description
`{cond}`	Condition field, see Table 1-6 on page 1-19.
`<Oprnd2>`	Operand2, see Table 1-4 on page 1-18.
`{field}`	Control field, see Table 1-5 on page 1-19.
S	Sets condition codes, optional.
B	Byte operation, optional.
H	Halfword operation, optional.
T	Forces address translation. Cannot be used with pre-indexed addresses.
Addressing modes	See *Addressing modes* on page 1-15.
`#32bit_Imm`	A 32-bit constant, formed by right-rotating an 8-bit value by an even number of bits.
`<reglist>`	A comma-separated list of registers, enclosed in braces ({ and }).

The ARM instruction set formats are shown in Figure 1-5 on page 1-12.

See the *ARM Architectural Reference Manual* for more information about the ARM instruction set formats.

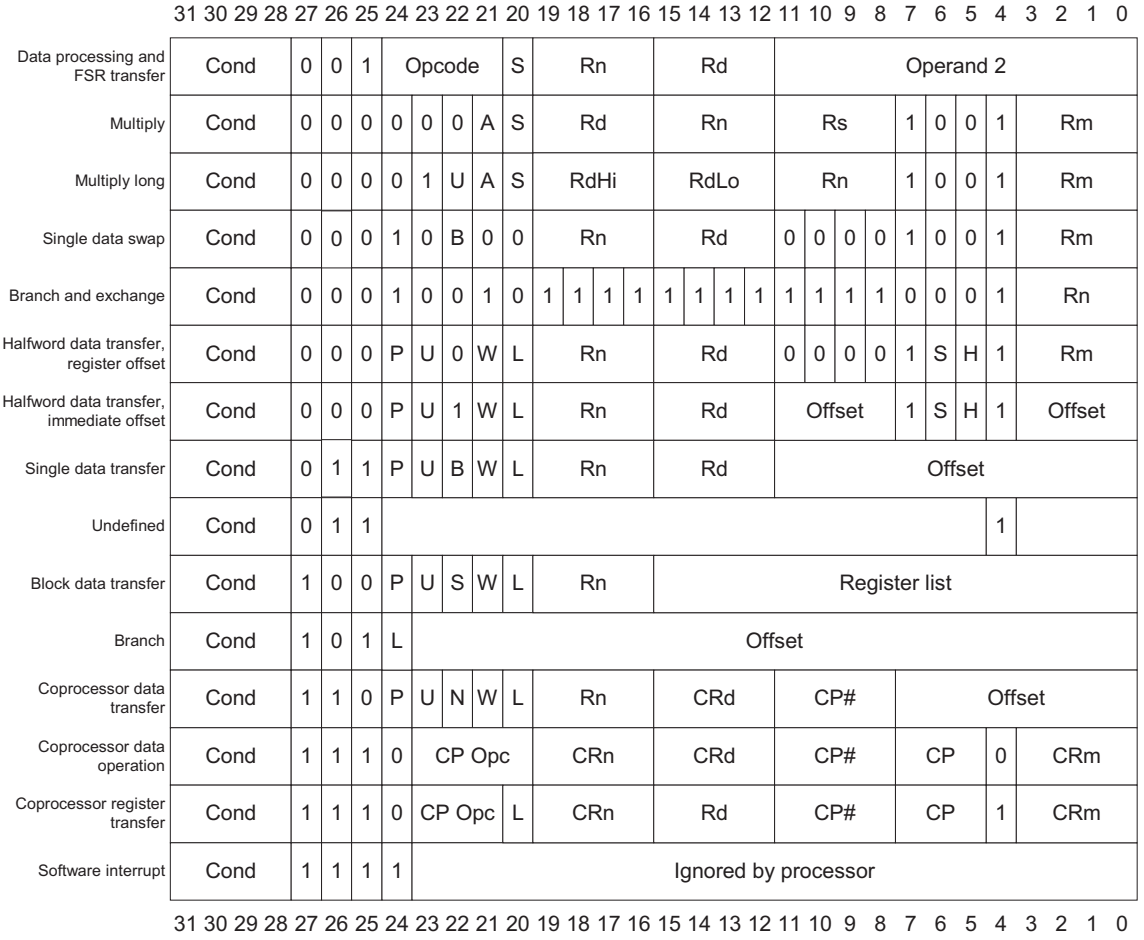

Figure 1-5 ARM instruction set formats

Note

Some instruction codes are not defined but do not cause the Undefined instruction trap to be taken, for instance a multiply instruction with bit [6] changed to a 1. These instructions must not be used because their action might change in future ARM implementations. The behavior of these instruction codes on the ARM7TDMI processor is unpredictable.

1.4.2 ARM instruction summary

The ARM instruction set summary is listed in Table 1-2.

Table 1-2 ARM instruction summary

Operation		Assembly syntax
Move	Move	`MOV{cond}{S} Rd, <Oprnd2>`
	Move NOT	`MVN{cond}{S} Rd, <Oprnd2>`
	Move SPSR to register	`MRS{cond} Rd, SPSR`
	Move CPSR to register	`MRS{cond} Rd, CPSR`
	Move register to SPSR	`MSR{cond} SPSR{field}, Rm`
	Move register to CPSR	`MSR{cond} CPSR{field}, Rm`
	Move immediate to SPSR flags	`MSR{cond} SPSR_f, #32bit_Imm`
	Move immediate to CPSR flags	`MSR{cond} CPSR_f, #32bit_Imm`
Arithmetic	Add	`ADD{cond}{S} Rd, Rn, <Oprnd2>`
	Add with carry	`ADC{cond}{S} Rd, Rn, <Oprnd2>`
	Subtract	`SUB{cond}{S} Rd, Rn, <Oprnd2>`
	Subtract with carry	`SBC{cond}{S} Rd, Rn, <Oprnd2>`
	Subtract reverse subtract	`RSB{cond}{S} Rd, Rn, <Oprnd2>`
	Subtract reverse subtract with carry	`RSC{cond}{S} Rd, Rn, <Oprnd2>`
	Multiply	`MUL{cond}{S} Rd, Rm, Rs`
	Multiply accumulate	`MLA{cond}{S} Rd, Rm, Rs, Rn`
	Multiply unsigned long	`UMULL{cond}{S} RdLo, RdHi, Rm, Rs`
	Multiply unsigned accumulate long	`UMLAL{cond}{S} RdLo, RdHi, Rm, Rs`
	Multiply signed long	`SMULL{cond}{S} RdLo, RdHi, Rm, Rs`
	Multiply signed accumulate long	`SMLAL{cond}{S} RdLo, RdHi, Rm, Rs`
	Compare	`CMP{cond} Rd, <Oprnd2>`
	Compare negative	`CMN{cond} Rd, <Oprnd2>`
Logical	Test	`TST{cond} Rn, <Oprnd2>`

Table 1-2 ARM instruction summary (continued)

Operation		Assembly syntax
	Test equivalence	`TEQ{cond} Rn, <Oprnd2>`
	AND	`AND{cond}{S} Rd, Rn, <Oprnd2>`
	EOR	`EOR{cond}{S} Rd, Rn, <Oprnd2>`
	ORR	`ORR{cond}{S} Rd, Rn, <Oprnd2>`
	Bit clear	`BIC{cond}{S} Rd, Rn, <Oprnd2>`
Branch	Branch	`B{cond} label`
	Branch with link	`BL{cond} label`
	Branch and exchange instruction set	`BX{cond} Rn`
Load	Word	`LDR{cond} Rd, <a_mode2>`
	Word with user-mode privilege	`LDR{cond}T Rd, <a_mode2P>`
	Byte	`LDR{cond}B Rd, <a_mode2>`
	Byte with user-mode privilege	`LDR{cond}BT Rd, <a_mode2P>`
	Byte signed	`LDR{cond}SB Rd, <a_mode3>`
	Halfword	`LDR{cond}H Rd, <a_mode3>`
	Halfword signed	`LDR{cond}SH Rd, <a_mode3>`
	Multiple block data operations	-
	• Increment before	`LDM{cond}IB Rd{!}, <reglist>{^}`
	• Increment after	`LDM{cond}IA Rd{!}, <reglist>{^}`
	• Decrement before	`LDM{cond}DB Rd{!}, <reglist>{^}`
	• Decrement after	`LDM{cond}DA Rd{!}, <reglist>{^}`
	• Stack operation	`LDM{cond}<a_mode4L> Rd{!}, <reglist>`
	• Stack operation, and restore CPSR	`LDM{cond}<a_mode4L> Rd{!}, <reglist+pc>^`
	• Stack operation with user registers	`LDM{cond}<a_mode4L> Rd{!}, <reglist>^`
Store	Word	`STR{cond} Rd, <a_mode2>`
	Word with user-mode privilege	`STR{cond}T Rd, <a_mode2P>`

Table 1-2 ARM instruction summary (continued)

Operation		Assembly syntax
	Byte	`STR{cond}B Rd, <a_mode2>`
	Byte with user-mode privilege	`STR{cond}BT Rd, <a_mode2P>`
	Halfword	`STR{cond}H Rd, <a_mode3>`
	Multiple block data operations	-
	• Increment before	`STM{cond}IB Rd{!}, <reglist>{^}`
	• Increment after	`STM{cond}IA Rd{!}, <reglist>{^}`
	• Decrement before	`STM{cond}DB Rd{!}, <reglist>{^}`
	• Decrement after	`STM{cond}DA Rd{!}, <reglist>{^}`
	• Stack operation	`STM{cond}<a_mode4S> Rd{!}, <reglist>`
	• Stack operation with user registers	`STM{cond}<a_mode4S> Rd{!}, <reglist>^`
Swap	Word	`SWP{cond} Rd, Rm, [Rn]`
	Byte	`SWP{cond}B Rd, Rm, [Rn]`
Coprocessors	Data operation	`CDP{cond} p<cpnum>, <op1>, CRd, CRn, CRm, <op2>`
	Move to ARM register from coprocessor	`MRC{cond} p<cpnum>, <op1>, Rd, CRn, CRm, <op2>`
	Move to coprocessor from ARM register	`MCR{cond} p<cpnum>, <op1>, Rd, CRn, CRm, <op2>`
	Load	`LDC{cond} p<cpnum>, CRd, <a_mode5>`
	Store	`STC{cond} p<cpnum>, CRd, <a_mode5>`
Software interrupt		`SWI 24bit_Imm`

Addressing modes

The addressing modes are procedures shared by different instructions for generating values used by the instructions. The five addressing modes used by the ARM7TDMI processor are:

Mode 1 Shifter operands for data processing instructions.

Mode 2 Load and store word or unsigned byte.

Mode 3 Load and store halfword or load signed byte.

Mode 4 Load and store multiple.

Mode 5 Load and store coprocessor.

The addressing modes are listed with their types and mnemonics Table 1-3.

Table 1-3 Addressing modes

Addressing mode	Type or addressing mode	Mnemonic or stack type
Mode 2 <a_mode2>	Immediate offset	`[Rn, #+/-12bit_Offset]`
	Register offset	`[Rn, +/-Rm]`
	Scaled register offset	`[Rn, +/-Rm, LSL #5bit_shift_imm]`
		`[Rn, +/-Rm, LSR #5bit_shift_imm]`
		`[Rn, +/-Rm, ASR #5bit_shift_imm]`
		`[Rn, +/-Rm, ROR #5bit_shift_imm]`
		`[Rn, +/-Rm, RRX]`
	Pre-indexed offset	-
	Immediate	`[Rn, #+/-12bit_Offset]!`
	Register	`[Rn, +/-Rm]!`
	Scaled register	`[Rn, +/-Rm, LSL #5bit_shift_imm]!`
		`[Rn, +/-Rm, LSR #5bit_shift_imm]!`
		`[Rn, +/-Rm, ASR #5bit_shift_imm]!`
		`[Rn, +/-Rm, ROR #5bit_shift_imm]!`
		`[Rn, +/-Rm, RRX]!`
	Post-indexed offset	-
	Immediate	`[Rn], #+/-12bit_Offset`
	Register	`[Rn], +/-Rm`
	Scaled register	`[Rn], +/-Rm, LSL #5bit_shift_imm`
		`[Rn], +/-Rm, LSR #5bit_shift_imm`
		`[Rn], +/-Rm, ASR #5bit_shift_imm`
		`[Rn], +/-Rm, ROR #5bit_shift_imm`
		`[Rn, +/-Rm, RRX]`

Table 1-3 Addressing modes (continued)

Addressing mode	Type or addressing mode	Mnemonic or stack type
Mode 2, privileged <a_mode2P>	Immediate offset	`[Rn, #+/-12bit_Offset]`
	Register offset	`[Rn, +/-Rm]`
	Scaled register offset	`[Rn, +/-Rm, LSL #5bit_shift_imm]`
		`[Rn, +/-Rm, LSR #5bit_shift_imm]`
		`[Rn, +/-Rm, ASR #5bit_shift_imm]`
		`[Rn, +/-Rm, ROR #5bit_shift_imm]`
		`[Rn, +/-Rm, RRX]`
	Post-indexed offset	-
	Immediate	`[Rn], #+/-12bit_Offset`
	Register	`[Rn], +/-Rm`
	Scaled register	`[Rn], +/-Rm, LSL #5bit_shift_imm`
		`[Rn], +/-Rm, LSR #5bit_shift_imm`
		`[Rn], +/-Rm, ASR #5bit_shift_imm`
		`[Rn], +/-Rm, ROR #5bit_shift_imm`
		`[Rn, +/-Rm, RRX]`
Mode 3, <a_mode3>	Immediate offset	`[Rn, #+/-8bit_Offset]`
	Pre-indexed	`[Rn, #+/-8bit_Offset]!`
	Post-indexed	`[Rn], #+/-8bit_Offset`
	Register	`[Rn, +/-Rm]`
	Pre-indexed	`[Rn, +/-Rm]!`
	Post-indexed	`[Rn], +/-Rm`
Mode 4, load <a_mode4L>	IA, increment after	FD, full descending
	IB, increment before	ED, empty descending
	DA, decrement after	FA, full ascending

Table 1-3 Addressing modes (continued)

Addressing mode	Type or addressing mode	Mnemonic or stack type
	DB decrement before	EA, empty ascending
Mode 4, store <a_mode4S>	IA, increment after	FD, full descending
	IB, increment before	ED, empty descending
	DA, decrement after	FA, full ascending
	DB decrement before	EA, empty ascending
Mode 5, coprocessor data transfer <a_mode5>	Immediate offset	`[Rn, #+/-(8bit_Offset*4)]`
	Pre-indexed	`[Rn, #+/-(8bit_Offset*4)]!`
	Post-indexed	`[Rn], #+/-(8bit_Offset*4)`

Operand 2

An operand is the part of the instruction that references data or a peripheral device. Operand 2 is listed in Table 1-4.

Table 1-4 Operand 2

Operand	Type	Mnemonic
Operand 2 <Oprnd2>	Immediate value	`#32bit_Imm`
	Logical shift left	`Rm LSL #5bit_Imm`
	Logical shift right	`Rm LSR #5bit_Imm`
	Arithmetic shift right	`Rm ASR #5bit_Imm`
	Rotate right	`Rm ROR #5bit_Imm`
	Register	`Rm`
	Logical shift left	`Rm LSL Rs`
	Logical shift right	`Rm LSR Rs`
	Arithmetic shift right	`Rm ASR Rs`
	Rotate right	`Rm ROR Rs`
	Rotate right extended	`Rm RRX`

Fields

Fields are listed in Table 1-5.

Table 1-5 Fields

Type	Suffix	Sets	Bit
Field {field}	_c	Control field mask bit	3
	_f	Flags field mask bit	0
	_s	Status field mask bit	1
	_x	Extension field mask bit	2

Condition fields

Condition fields are listed in Table 1-6.

Table 1-6 Condition fields

Field type	Suffix	Description	Condition
Condition {cond}	EQ	Equal	Z set
	NE	Not equal	Z clear
	CS	Unsigned higher, or same	C set
	CC	Unsigned lower	C clear
	MI	Negative	N set
	PL	Positive, or zero	N clear
	VS	Overflow	V set
	VC	No overflow	V clear
	HI	Unsigned higher	C set, Z clear
	LS	Unsigned lower, or same	C clear, Z set
	GE	Greater, or equal	N=V (N and V set or N and V clear)
	LT	Less than	N<>V (N set and V clear) or (N clear and V set)

Table 1-6 Condition fields (continued)

Field type	Suffix	Description	Condition
	GT	Greater than	Z clear, N=V (N and V set or N and V clear)
	LE	Less than, or equal	Z set or N<>V (N set and V clear) or (N clear and V set)
	AL	Always	Flag ignored

1.4.3 Thumb instruction summary

The Thumb instruction set formats are shown in Figure 1-6 on page 1-21.

See the *ARM Architectural Reference Manual* for more information about the ARM instruction set formats.