1.4.1. ARM instruction summary

The ARM instruction set summary is shown in Table 1.2.

Table 1.2. ARM instruction summary

Operation	Description	Assembler
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>`

Operation	Description	Assembler
	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 operations	`LDM{cond}<a_mode4L> Rd{!}, <reglist>`
	Stack operations and restore CPSR	`LDM{cond}<a_mode4L> Rd{!}, <reglist+pc>^`
	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>`
	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>`

Operation	Description	Assembler
	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 operations	STM{cond}<a_mode4S> Rd{!}, <reglist>
	User registers	STM{cond}<a_mode4S> Rd{!}, <reglist>^
Swap	Word	SWP{cond} Rd, Rm, [Rn]
	Byte	SWP{cond}B Rd, Rm, [Rn]
Coprocessors	Data operations	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 mode 2, <a_mode2>, is shown in Table 1.3.

Table 1.3. Addressing mode 2

Operation	Assembler
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 immediate offset	[Rn, #+/-12bit_Offset]!
Pre-indexed register offset	[Rn, +/-Rm]!
Pre-indexed scaled register offset	[Rn, +/-Rm, LSL #5bit_shift_imm]!
	[Rn, +/-Rm, LSR #5bit_shift_imm]!

Operation	Assembler
	[Rn, +/-Rm, ASR #5bit_shift_imm]!
	[Rn, +/-Rm, ROR #5bit_shift_imm]!
	[Rn, +/-Rm, RRX]!
Post-indexed immediate offset	[Rn], #+/-12bit_Offset
Post-indexed register offset	[Rn], +/-Rm
Post-indexed 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]

Addressing mode 2 (privileged), <a_mode2P>, is shown in Table 1.4.

Table 1.4. Addressing mode 2 (privileged)

Operation	Assembler
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 immediate offset	[Rn], #+/-12bit_Offset
Post-indexed register offset	[Rn], +/-Rm
Post-indexed 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]

Addressing mode 3 (signed byte, and halfword data transfer), <a_mode3>, is shown in Table 1.5.

Table 1.5. Addressing mode 3

Operation	Assembler
Immediate offset	[Rn, #+/-8bit_Offset]
Pre-indexed	[Rn, #+/-8bit_Offset]!
Post-indexed	[Rn], #+/-8bit_Offset

Operation	Assembler
Register	`[Rn, +/-Rm]`
Pre-indexed	`[Rn, +/-Rm]!`
Post-indexed	`[Rn], +/-Rm`

Addressing mode 4 (load), `<a_mode4L>`, is shown in Table 1.6.

Table 1.6. Addressing mode 4 (load)

Addressing mode	Stack type
IA Increment after	FD Full descending
IB Increment before	ED Empty descending
DA Decrement after	FA Full ascending
DB Decrement before	EA Empty ascending

Addressing mode 4 (store), `<a_mode4S>`, is shown in Table 1.7.

Table 1.7. Addressing mode 4 (store)

Addressing mode	Stack type
IA Increment after	EA Empty ascending
IB Increment before	FA Full ascending
DA Decrement after	ED Empty descending
DB Decrement before	FD Full descending

Addressing mode 5 (coprocessor data transfer), `<a_mode5>`, is shown in Table 1.8.

Table 1.8. Addressing mode 5

Operation	Assembler
Immediate offset	`[Rn, #+/-(8bit_Offset*4)]`
Pre-indexed	`[Rn, #+/-(8bit_Offset*4)]!`
Post-indexed	`[Rn], #+/-(8bit_Offset*4)`

Operand 2, `<Oprnd2>`, is shown in Table 1.9.

Table 1.9. Operand 2

Operation	Assembler
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`

Operation	Assembler
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, `{field}`, are shown in Table 1.10.

Table 1.10. Fields

Suffix	Sets
_c	Control field mask bit (bit 3)
_f	Flags field mask bit (bit 0)
_s	Status field mask bit (bit 1)
_x	Extension field mask bit (bit 2)

Condition fields, `{cond}`, are shown in Table 1.11.

Table 1.11. Condition fields

Suffix	Description
EQ	Equal
NE	Not equal
CS	Unsigned higher, or same
CC	Unsigned lower
MI	Negative
PL	Positive, or zero
VS	Overflow
VC	No overflow
HI	Unsigned higher
LS	Unsigned lower, or same
GE	Greater, or equal
LT	Less than
GT	Greater than
LE	Less than, or equal

Suffix	Description
AL	Always