

## ARMv4T Partial Instruction Set Summary

Operation		Syntax	Page Num.	S updates			Action and Comments	
Move	Move	<b>mov</b> {cond}{s} <i>Rd</i> , <i>shift_op</i>	A4-56 (156)	N	Z	C	<i>Rd</i> := <i>shift_op</i>	
	with NOT	<b>mvn</b> {cond}{s} <i>Rd</i> , <i>shift_op</i>	A4-68 (168)	N	Z	C	<i>Rd</i> := NOT( <i>shift_op</i> )	
	CPSR to register	<b>mrs</b> {cond} <i>Rd</i> , <b>cpsr</b>	A4-60 (160)				<i>Rd</i> := CPSR	
	SPSR to register	<b>mrs</b> {cond} <i>Rd</i> , <b>spsr</b>	A4-60 (160)				<i>Rd</i> := SPSR; Not valid in System or User modes	
	register to CPSR	<b>msr</b> {cond} <b>cpsr_fields</b> , <i>Rm</i>	A4-62 (162)				CPSR := <i>Rd</i> (selected bytes only)	
	register to SPSR	<b>msr</b> {cond} <b>spsr_fields</b> , <i>Rm</i>	A4-62 (162)				SPSR := <i>Rd</i> (selected bytes only); Not valid in System or User modes	
	immediate to CPSR	<b>msr</b> {cond} <b>cpsr_fields</b> , # <i>imm8r</i>	A4-62 (162)				CPSR := <i>imm8r</i> (selected bytes only)	
	immediate to SPSR	<b>msr</b> {cond} <b>spsr_fields</b> , # <i>imm8r</i>	A4-62 (162)				SPSR := <i>imm8r</i> (selected bytes only); Not valid in System or User modes	
Arithmetic	Add	<b>add</b> {cond}{s} <i>Rd</i> , <i>Rn</i> , <i>shift_op</i>	A4-6 (106)	N	Z	C	<i>Rd</i> := <i>Rn</i> + <i>shift_op</i>	
	with carry	<b>adc</b> {cond}{s} <i>Rd</i> , <i>Rn</i> , <i>shift_op</i>	A4-4 (104)	N	Z	C	<i>Rd</i> := <i>Rn</i> + <i>shift_op</i> + Carry	
	Subtract	<b>sub</b> {cond}{s} <i>Rd</i> , <i>Rn</i> , <i>shift_op</i>	A4-98 (198)	N	Z	C	<i>Rd</i> := <i>Rn</i> - <i>shift_op</i>	
	with carry	<b>sbc</b> {cond}{s} <i>Rd</i> , <i>Rn</i> , <i>shift_op</i>	A4-76 (176)	N	Z	C	<i>Rd</i> := <i>Rn</i> - <i>shift_op</i> - NOT(Carry)	
	reverse subtract	<b>rsb</b> {cond}{s} <i>Rd</i> , <i>Rn</i> , <i>shift_op</i>	A4-72 (172)	N	Z	C	<i>Rd</i> := <i>shift_op</i> - <i>Rn</i>	
	reverse subtract with carry	<b>rsc</b> {cond}{s} <i>Rd</i> , <i>Rn</i> , <i>shift_op</i>	A4-74 (174)	N	Z	C	<i>Rd</i> := <i>shift_op</i> - <i>Rn</i> - NOT(Carry)	
	Multiply	<b>mul</b> {cond}{s} <i>Rd</i> , <i>Rm</i> , <i>Rs</i>	A4-66 (166)	N	Z	C?	<i>Rd</i> := <i>Rm</i> × <i>Rs</i> (lower 32 bits only)	
	with accumulate	<b>mla</b> {cond}{s} <i>Rd</i> , <i>Rm</i> , <i>Rs</i> , <i>Rn</i>	A4-54 (154)	N	Z	C?	<i>Rd</i> := ( <i>Rm</i> × <i>Rs</i> ) + <i>Rn</i> (lower 32 bits only)	
Logical	AND	<b>and</b> {cond}{s} <i>Rd</i> , <i>Rn</i> , <i>shift_op</i>	A4-8 (108)	N	Z	C	<i>Rd</i> := <i>Rn</i> AND <i>shift_op</i>	
	OR	<b>orr</b> {cond}{s} <i>Rd</i> , <i>Rn</i> , <i>shift_op</i>	A4-70 (170)	N	Z	C	<i>Rd</i> := <i>Rn</i> OR <i>shift_op</i>	
	XOR	<b>eor</b> {cond}{s} <i>Rd</i> , <i>Rn</i> , <i>shift_op</i>	A4-26 (126)	N	Z	C	<i>Rd</i> := <i>Rn</i> XOR <i>shift_op</i>	
	Bit clear	<b>bic</b> {cond}{s} <i>Rd</i> , <i>Rn</i> , <i>shift_op</i>	A4-12 (112)	N	Z	C	<i>Rd</i> := <i>Rn</i> AND NOT( <i>shift_op</i> )	
	Rotates and shifts	Usually <b>mov</b> {cond}{s} <i>Rd</i> , <i>shift_op</i>	A4-56 (156)	N	Z	C	Usually <i>Rd</i> := <i>shift_op</i> ; May use other instructions (eg, <b>add</b> , <b>sub</b> )	
	Compare	Compare	<b>cmp</b> {cond} <i>Rn</i> , <i>shift_op</i>	A4-25 (125)	N	Z	C	Update CPSR flags for <i>Rn</i> - <i>shift_op</i>
		negated	<b>cmn</b> {cond} <i>Rn</i> , <i>shift_op</i>	A4-23 (123)	N	Z	C	Update CPSR flags for <i>Rn</i> + <i>shift_op</i>
		Bitwise test	<b>tst</b> {cond} <i>Rn</i> , <i>shift_op</i>	A4-107 (207)	N	Z	C	Update CPSR flags for <i>Rn</i> AND <i>shift_op</i>
Equivalence test		<b>teq</b> {cond} <i>Rn</i> , <i>shift_op</i>	A4-106 (206)	N	Z	C	Update CPSR flags for <i>Rn</i> XOR <i>shift_op</i>	
Branch	Branch	<b>b</b> {cond} <i>label</i>	A4-10 (110)				<i>R15</i> := address of <i>label</i> ; Jump to <i>label</i>	
	with link	<b>bl</b> {cond} <i>label</i>	A4-10 (110)				<i>R14</i> := <i>R15</i> - 4; <i>R15</i> := address of <i>label</i> ; Call subroutine/function at <i>label</i>	
	and exchange	<b>bx</b> {cond} <i>Rm</i>	A4-19 (119)				<i>R15</i> := <i>Rm</i> ; Changes to Thumb mode if bit 0 of <i>Rm</i> is 1	
	Return from subroutine	Usually <b>mov pc</b> , <b>lr</b> or <b>bx lr</b>	A4-11 (111)				<i>R15</i> := <i>R14</i>	
Load	Load word	<b>ldr</b> {cond} <i>Rd</i> , <i>am2</i>	A4-37 (137)				<i>Rd</i> := word stored at address <i>am2</i>	
	using User mode privileges	<b>ldr</b> {cond} <b>t</b> <i>Rd</i> , <i>am2P</i>	A4-50 (150)				<i>Rd</i> := word stored at address <i>am2P</i> ; Usually used in non-User modes	
	Load byte	<b>ldr</b> {cond} <b>b</b> <i>Rd</i> , <i>am2</i>	A4-40 (140)				<i>Rd</i> := ZeroExtend(byte at address <i>am2</i> )	
	signed	<b>ldr</b> {cond} <b>sb</b> <i>Rd</i> , <i>am3</i>	A4-46 (146)				<i>Rd</i> := SignExtend(byte at address <i>am3</i> )	
	using User mode privileges	<b>ldr</b> {cond} <b>bt</b> <i>Rd</i> , <i>am2P</i>	A4-42 (142)				<i>Rd</i> := ZeroExtend(byte at address <i>am2P</i> ); Usually used in non-User modes	
	Load half-word	<b>ldr</b> {cond} <b>h</b> <i>Rd</i> , <i>am3</i>	A4-44 (144)				<i>Rd</i> := ZeroExtend(half-word at address <i>am3</i> )	
Store	Store word	<b>str</b> {cond} <i>Rd</i> , <i>am2</i>	A4-88 (188)				Store word in <i>Rd</i> at address <i>am2</i>	
	using User mode privileges	<b>str</b> {cond} <b>t</b> <i>Rd</i> , <i>am2P</i>	A4-96 (196)				Store word in <i>Rd</i> at address <i>am2P</i> ; Usually used in non-User modes	
	Store byte	<b>str</b> {cond} <b>b</b> <i>Rd</i> , <i>am2</i>	A4-90 (190)				Store byte in <i>Rd</i> [7..0] at address <i>am2</i>	
	using User mode privileges	<b>str</b> {cond} <b>bt</b> <i>Rd</i> , <i>am2P</i>	A4-92 (192)				Store byte in <i>Rd</i> [7..0] at address <i>am2P</i> ; Usually used in non-User modes	
	Store half-word	<b>str</b> {cond} <b>h</b> <i>Rd</i> , <i>am3</i>	A4-94 (194)				Store half-word in <i>Rd</i> [15..0] at address <i>am3</i>	
	Load Multiple	Pop, or Block data load	<b>ldm</b> {cond}{ <i>am4L</i> } <i>Rd</i> !, { <i>reglist</i> }	A4-30 (130)				Load all listed registers from address in <i>Rd</i>
using User mode privileges		<b>ldm</b> {cond}{ <i>am4L</i> } <i>Rd</i> !, { <i>reglist-pc</i> }^	A4-32 (132)				Load all listed registers (PC register not listed) from address in <i>Rd</i> ; Not valid in User mode	
return and restore CPSR		<b>ldm</b> {cond}{ <i>am4L</i> } <i>Rd</i> !, { <i>reglist+pc</i> }^	A4-34 (134)				Load registers, CPSR := SPSR, branch to PC; Not valid in User or System modes	
Store Multiple	Push, or Block data store	<b>stm</b> {cond}{ <i>am4S</i> } <i>Rd</i> !, { <i>reglist</i> }	A4-84 (184)				Store all listed registers to address in <i>Rd</i>	
	using User mode privileges	<b>stm</b> {cond}{ <i>am4S</i> } <i>Rd</i> !, { <i>reglist</i> }^	A4-86 (186)				Store all listed registers to address in <i>Rd</i> ; Usually used in non-User modes	
Miscellaneous	Swap word	<b>swp</b> {cond} <i>Rd</i> , <i>Rm</i> , [ <i>Rn</i> ]	A4-102 (202)				temp := word at address in <i>Rn</i> , store <i>Rm</i> to address in <i>Rn</i> , <i>Rd</i> := temp	
	Swap byte	<b>swp</b> {cond} <b>b</b> <i>Rd</i> , <i>Rm</i> , [ <i>Rn</i> ]	A4-104 (204)				temp := ZeroExtend(byte at address in <i>Rn</i> ), store <i>Rm</i> [7..0] to address in <i>Rn</i> , <i>Rd</i> := temp	
	Software interrupt	<b>swi</b> {cond} <i>imm24</i>	A4-100 (200)				Branch-and-link to address 0x00000008 in Supervisor mode	
Pseudo-instructions	No operation	<b>nop</b> {cond}					Does nothing; translates to <b>mov</b> {cond} <b>r0</b> , <b>r0</b>	
	Load variable	<b>ldr</b> {cond} <i>Rd</i> , <i>label</i>					<i>Rd</i> := word stored at address <i>label</i> ; translates to <b>ldr</b> {cond} <i>Rd</i> , [ <b>r15</b> , # <i>offset</i> ]	
	Move constant	<b>ldr</b> {cond} <i>Rd</i> , = <i>imm32</i>					<b>mov</b> <i>Rd</i> , # <i>imm32</i> or <b>ldr</b> <i>Rd</i> , [ <b>r15</b> , # <i>offset</i> ]	
	Load address	<b>adr</b> {cond} <i>Rd</i> , <i>label</i>					<b>add</b> <i>Rd</i> , <b>r15</b> , # <i>offset</i>	
	long version	<b>adr</b> {cond} <b>l</b> <i>Rd</i> , <i>label</i>					Two-instruction form of <b>adr</b>	

# ARMv4T Partial Instruction Set Tables

## Register Names and Aliases

Reg	Aliases	Purpose in ARM Thumb Procedure Call Standard
r0	a1	Argument/result/scratch register 1
r1	a2	Argument/result/scratch register 2
r2	a3	Argument/result/scratch register 3
r3	a4	Argument/result/scratch register 4
r4	v1	Variable register 1
r5	v2	Variable register 2
r6	v3	Variable register 3
r7	v4	Variable register 4
r8	v5	Variable register 5
r9	v6 or sb	Variable register 6; sometimes Stack Base register
r10	v7 or sl	Variable register 7; sometimes Stack Limit register
r11	v8 or fp	Variable register 8; usually Frame Pointer register
r12	ip	Intra-procedure-call scratch register
r13	sp	Stack Pointer
r14	lr	Link Register
r15	pc	Program Counter

## Program Status Register Format

31	28	24	16	8	7	6	5	4	0	
N	Z	C	V	Undef.	Undef.	Undef.	I	F	T	Mode

## Program Status Register Modes

Value	Mode	Accessible registers
0b1000	User	PC, R14-R0, CPSR
0b10001	Fast Interrupt	PC, R14_fiq-R8_fiq, R7-R0, CPSR, SPSR_fiq
0b10010	Interrupt	PC, R14_irq-R13_irq, R12-R0, CPSR, SPSR_irq
0b10011	Supervisor	PC, R14_svc-R13_svc, R12-R0, CPSR, SPSR_svc
0b10111	Abort	PC, R14_abt-R13_abt, R12-R0, CPSR, SPSR_abt
0b11011	Undefined	PC, R14_und-R13_und, R12-R0, CPSR, SPSR_und
0b11111	System	PC, R14-R0, CPSR

## Program Status Register Fields: *fields*

Suffix	Bits in PSRs	Description
c	0-7	Control field mask
x	8-15	Extension field mask (no bits currently defined)
s	16-23	Status field mask (no bits currently defined)
f	24-31	Flags field mask (bits 24-27 undefined)

## Optional Condition Field: *cond*

Mnemonic	Description	Condition flags state
EQ	Equal	Z set
NE	Not equal	Z clear
CS or HS	Carry set/unsigned higher or same	C set
CC or LO	Carry clear/unsigned lower	C clear
MI	Minus/negative	N set
PL	Plus/positive or zero	N clear
VS	Overflow	V set
VC	No overflow	V clear
HI	Unsigned higher	C set and Z clear
LS	Unsigned lower or same	C clear or Z set
GE	Signed greater than or equal	N equal to V
LT	Signed less than	N not equal to V
GT	Signed greater than	Z clear and N equal to V
LE	Signed less than or equal	Z set and N not equal to V
AL	Always (unconditional, default)	Irrelevant

## Data Processing Mode: *shifter\_op*

Operation	Syntax	Comments
Immediate value	#imm8r	
Register	Rm	
Logical shift left immediate	Rm, lsl #imm5	Allowed 0-31 only
Logical shift left by register	Rm, lsl Rs	
Logical shift right immediate	Rm, lsr #imm5	Allowed 1-32 only
Logical shift right by register	Rm, lsr Rs	
Arithmetic shift right immediate	Rm, asr #imm5	Allowed 1-32 only
Arithmetic shift right by register	Rm, asr Rs	
Rotate right immediate	Rm, ror #imm5	Allowed 1-31 only
Rotate right by register	Rm, ror Rs	
Rotate right with extend	Rm, rrx	

## Load or Store Word/Unsigned Byte Mode: *am2*

Operation	Syntax	Cmts.
Pre-indexed	Immediate offset	[Rn], #±imm12{!}
	Zero offset	[Rn]
	Register offset	[Rn], ±Rm{!}
	Scaled register offset	[Rn], ±Rm, lsl #imm5{!}
		[Rn], ±Rm, lsr #imm5{!}
		[Rn], ±Rm, asr #imm5{!}
		[Rn], ±Rm, ror #imm5{!}
		[Rn], ±Rm, rrx{!}
Post-indexed	Immediate offset	[Rn], #±imm12
	Register offset	[Rn], ±Rm
	Scaled register offset	[Rn], ±Rm, lsl #imm5
		[Rn], ±Rm, lsr #imm5
		[Rn], ±Rm, asr #imm5
		[Rn], ±Rm, ror #imm5
		[Rn], ±Rm, rrx

## Load or Store with Translation Mode: *am2P*

Operation	Syntax	Cmts.
Post-indexed	Immediate offset	[Rn], #±imm12
	Register offset	[Rn], ±Rm
	Scaled register offset	[Rn], ±Rm, lsl #imm5
		[Rn], ±Rm, lsr #imm5
		[Rn], ±Rm, asr #imm5
		[Rn], ±Rm, ror #imm5
		[Rn], ±Rm, rrx

## Load or Store Half-Word/Signed Byte Mode: *am3*

Operation	Syntax	Comments
Pre-indexed	Immediate offset	[Rn], #±imm8{!}
	Zero offset	[Rn]
	Register offset	[Rn], ±Rm{!}
Post-indexed	Immediate offset	[Rn], #±imm8
	Register offset	[Rn], ±Rm

## Load Multiple Data Mode: *am4L*

Suffix	Non-stack Addressing Mode	Suffix	Stack Addressing Mode
ia	Increment after	fd	Full descending
ib	Increment before	ed	Empty descending
da	Decrement after	fa	Full ascending
db	Decrement before	ea	Empty ascending

## Store Multiple Data Mode: *am4S*

Suffix	Non-stack Addressing Mode	Suffix	Stack Addressing Mode
ia	Increment after	ea	Empty ascending
ib	Increment before	fa	Full ascending
da	Decrement after	ed	Empty descending
db	Decrement before	fd	Full descending

## Exception Vector Table

Address	Mode	Exception Type
0x00000000	Supervisor	Reset
0x00000004	Undefined	Undefined instruction
0x00000008	Supervisor	Software interrupt
0x0000000C	Abort	Prefetch abort (instruction fetch abort)
0x00000010	Abort	Data abort (data access memory abort)
0x00000014	(None)	(Not used)
0x00000018	Interrupt	Normal-priority interrupt
0x0000001C	Fast Interrupt	High-priority (fast) interrupt

## Miscellaneous

Symbol	Meaning
imm5	Immediate 5-bit number, either 0-31, 1-32 or 1-31
imm8	Immediate 8-bit number, between 0-255
imm8r	A 32-bit number that can be formed by rotating an 8-bit number (0-255) by an even number between 0 and 30
imm12	Immediate 12-bit number, between 0-4095
imm24	Immediate 24-bit number, between 0-16,777,215
imm32	Immediate 32-bit number, between 0-4,294,967,295
{s}	If present, the instruction will update the condition flags
N	Negative flag: 1 if result is negative
Z	Zero flag: 1 if result is zero
C	Carry flag
V	Signed Overflow flag
C?	Carry flag ends in an unpredictable state, if flags are set
V?	Overflow flag ends in an unpredictable state, if flags are set
I	Interrupt Disable bit in the PSRs: 1 to disable interrupts
F	Fast Interrupt Disable bit: 1 to disable fast interrupts
T	ARM or Thumb state: 0 for ARM execution, 1 for Thumb
{reglist}	List of registers separated by commas or dashes, surrounded by braces, eg. {r0, r1, r2} or {r0-r3, r5}
{reglist-pc}	List of registers that does <i>not</i> include PC (R15)
{reglist+pc}	List of registers that <i>does</i> include PC (R15)
{!}	If present, the instruction updates the base register after the memory transfer.
	Post-indexed accesses <i>always</i> update the base register
±	Either + or - may be supplied; + is assumed if not present

This document contains a summary of the ARMv4T instruction set architecture in tabular format. It does not list every instruction available in the ARM architecture: the coprocessor instructions, in particular, have not been listed. Page numbers refer to both the printed and on-line versions of the *ARM Architecture Reference Manual*, Second Edition, published by Addison-Wesley in December 2000 (ISBN 0-201-73719-1).

This document was created by John Zaitseff for the Digital Systems Laboratory at the University of New South Wales.

Copyright © 2003, School of Electrical Engineering and Telecommunications, University of New South Wales, Sydney, Australia. All rights reserved.