ARM64 (AArch64) Reference Sheet

Idr D, [R] D = Mem[R] Idp D1, D2, [R] D1 = Mem[R] D2 = Mem[R + 8] Str S, [R] Mem[R] = S Stp S1, S2, [R] Mem[R] = S Mem[R + 8] = S2 add D, O1, O2 D = O1 + O2 sub D, O1, O2 D = O1 - O2 neg D, O1 D = -(O1) mul D, O1, O2 D = O1 / O2 (unsigned) sdiv D, O1, O2 D = O1 / O2 (signed) Isl D, R, #v D = R << v Isr D, R, #v D = R >> v (logical) asr D, R, #v D = R >> v (arithmetic) and D, O1, O2 D = O1 O2 orr D, O1, O2 D = O1	<u>Instructions</u> mov D, S	D = S
D1		
D2 = Mem[R + 8] str S, [R]	·	
str S, [R] Mem[R] = S stp S1, S2, [R] Mem[R] = S1 Mem[R + 8] = S2 add D, O1, O2 D = O1 + O2 sub D, O1, O2 D = O1 - O2 neg D, O1 D = -(O1) mul D, O1, O2 D = O1 * O2 udiv D, O1, O2 D = O1 / O2 (unsigned) sdiv D, O1, O2 D = O1 / O2 (signed) lsl D, R, #v D = R << v lsr D, R, #v D = R >> v (logical) asr D, R, #v D = R >> v (arithmetic) and D, O1, O2 D = O1 & O2 orr D, O1, O2 D = O1 & O2 mvn D, O D = ~O cmp O1, O2 Sets CCs: O1 - O2 tst O1, O2 Sets CCs: O1 & O2 br address PC = address cbz R, label If R == 0, PC = addr of label branch (PC = address of label) branch if equal branch if negative b.eq label branch if non-negative b.gt label branch if greater than b.ge label branch if less than b.le label branch if less or equal b.lt label branch if less or equal	14p b1, b2, [N]	
stp S1, S2, [R] Mem[R] = S1 Mem[R + 8] = S2 add D, O1, O2 D = O1 + O2 sub D, O1, O2 D = O1 - O2 neg D, O1 D = -(O1) mul D, O1, O2 D = O1 / O2 (unsigned) sdiv D, O1, O2 D = O1 / O2 (signed) sdiv D, O1, O2 D = O1 / O2 (signed) lsl D, R, #v D = R << v	str S. [R]	
### Mem[R + 8] = S2 add D, 01, 02		
sub D, 01, 02 D = 01 - 02 neg D, 01 D = -(01) mul D, 01, 02 D = 01 * 02 udiv D, 01, 02 D = 01 / 02 (unsigned) sdiv D, 01, 02 D = 01 / 02 (signed) lsl D, R, #v D = R << v		
neg D, O1	add D, 01, 02	D = 01 + 02
mul D, O1, O2 D = O1 * O2 udiv D, O1, O2 D = O1 / O2 (unsigned) sdiv D, O1, O2 D = O1 / O2 (signed) lsl D, R, #v D = R << v	sub D, 01, 02	D = O1 - O2
udiv D, 01, 02 D = 01 / 02 (unsigned) sdiv D, 01, 02 D = 01 / 02 (signed) lsl D, R, #v D = R << v	neg D, 01	D = -(01)
Sdiv D, O1, O2	mul D, O1, O2	D = 01 * 02
lsl D, R, #v D = R << v	udiv D, O1, O2	D = 01 / 02 (unsigned)
D = R >> v (logical)	sdiv D, 01, 02	D = 01 / 02 (signed)
asr D, R, #v and D, O1, O2 orr D, O1, O2 eor D, O1, O2 mvn D, O cmp O1, O2 br address cbz R, label cbnz R, label b label b label b ne label b.mi label b.mi label b.mi label b.mi label b.mi label b.gt label b.lt label b.lt label bladdress <fname> bladdress cfname> cfname> cfname> cfar S, w (arithmetic) D = O1 & O2 carthed O2 D = O1 O2 D = O1</fname>	lsl D, R, #v	$D = R \ll v$
and D, O1, O2 orr D, O1, O2 eor D, O1, O2 mvn D, O cmp O1, O2 br address cbz R, label cbnz R, label b label b label b label b ne label b .mi label b .pl label b .ge label b .ge label b .ge label b .le label b	lsr D, R, #v	$D = R \gg v \text{ (logical)}$
orr D, 01, 02 eor D, 01, 02 D = 01	asr D, R, #v	D = R >> v (arithmetic)
eor D, O1, O2 mvn D, O cmp O1, O2 tst O1, O2 br address cbz R, label cbnz R, label blabel bladdress <fname> blabel blabel bladdress blabel bladdress blabel bladdress <fname> x30 = PC + 4 PC = address blabel bladdress blabel bladdress</fname></fname>	and D, 01, 02	D = 01 & 02
mvn D, O cmp O1, O2 Sets CCs: O1 - O2 tst O1, O2 Sets CCs: O1 & O2 br address cbz R, label Cbnz R, label Cpc = address Cpc = Add	orr D, 01, 02	$D = 01 \mid 02$
cmp 01, 02 tst 01, 02 Sets CCs: 01 - 02 br address cbz R, label cbnz R, label b label b label b.eq label b.mi label b.pl label b.pl label b.ge label b.tt label b.lt label b.lt label blabel branch if less or equal blabel branch if less or equal blabel branch if less or equal	eor D, 01, 02	$D = 01 ^002$
tst 01, 02 br address cbz R, label cbnz R, label b label b label b.eq label b.mi label b.pl label b.pl label b.gt label b.tt label b.lt label b.lt label b.lt label branch if less than b.le label branch if less or equal bladdress <fname> x30 = PC + 4 PC = R ret Sets CCs: 01 & 02 PC = address PC = address PC = addr of label branch if R != 0, PC = addr of label branch (PC = address of label) branch if not equal branch if negative branch if greater than branch if greater or equal branch if less than branch if less or equal x30 = PC + 4 PC = R PC = x30</fname>	mvn D, O	D = ~O
br address cbz R, label If R == 0, PC = addr of label cbnz R, label If R != 0, PC = addr of label branch (PC = address of label) branch if equal b.ne label branch if not equal b.mi label branch if non-negative b.pl label branch if greater than b.ge label branch if greater or equal b.lt label branch if less than b.le label branch if less or equal bl address <fname> x30 = PC + 4 PC = address branch if less or equal contact representations and representations are provided to the provid</fname>	cmp 01, 02	Sets CCs: 01 - 02
cbz R, label cbnz R, label If R != 0, PC = addr of label If R != 0, PC = addr of label b label branch (PC = address of label) b.eq label branch if equal b.ne label branch if not equal b.mi label branch if non-negative b.pl label branch if greater than b.ge label branch if greater or equal b.lt label branch if less than b.le label branch if less or equal bl address <fname> x30 = PC + 4 PC = address blr R <fname> x30 = PC + 4 PC = R PC = x30</fname></fname>	tst 01, 02	Sets CCs: 01 & 02
cbnz R, label If R != 0, PC = addr of label b label branch (PC = address of label) b.eq label branch if equal b.ne label branch if not equal b.mi label branch if non-negative b.pl label branch if greater than b.ge label branch if greater or equal b.lt label branch if less than b.le label branch if less or equal bl address <fname> x30 = PC + 4 PC = address blr R <fname> x30 = PC + 4 PC = R PC = x30</fname></fname>	br address	PC = address
cbnz R, label If R != 0, PC = addr of label b label branch (PC = address of label) b.eq label branch if equal b.ne label branch if not equal b.mi label branch if negative b.pl label branch if greater than b.ge label branch if greater or equal b.lt label branch if less than b.le label branch if less or equal bl address <fname> x30 = PC + 4 PC = address blr R <fname> x30 = PC + 4 PC = R PC = x30</fname></fname>	cbz R, label	·
b label branch (PC = address of label) b.eq label branch if equal b.ne label branch if not equal b.mi label branch if negative b.pl label branch if non-negative b.gt label branch if greater than b.ge label branch if greater or equal b.lt label branch if less than b.le label branch if less or equal bl address <fname> x30 = PC + 4 PC = address blr R <fname> x30 = PC + 4 PC = R PC = x30</fname></fname>	cbnz R, label	
(PC = address of label) b.eq label branch if equal b.ne label branch if not equal b.mi label branch if negative b.pl label branch if greater than b.gt label branch if greater or equal b.lt label branch if less than b.le label branch if less or equal bl address <fname> x30 = PC + 4 PC = address blr R <fname> x30 = PC + 4 PC = R ret PC = x30</fname></fname>		label
b.eq label branch if equal b.ne label branch if not equal b.mi label branch if negative b.pl label branch if non-negative b.gt label branch if greater than b.ge label branch if greater or equal b.lt label branch if less than b.le label branch if less or equal bl address <fname> x30 = PC + 4 PC = address blr R <fname> x30 = PC + 4 PC = R ret PC = x30</fname></fname>	b label	branch
b.ne label branch if not equal b.mi label branch if negative b.pl label branch if non-negative b.gt label branch if greater than b.ge label branch if greater or equal b.lt label branch if less than b.le label branch if less or equal bl address <fname> x30 = PC + 4 PC = address blr R <fname> x30 = PC + 4 PC = R ret PC = x30</fname></fname>		(PC = address of label)
b.mi label branch if negative b.pl label branch if non-negative b.gt label branch if greater than b.ge label branch if greater or equal b.lt label branch if less than b.le label branch if less or equal bl address <fname> x30 = PC + 4 PC = address blr R <fname> x30 = PC + 4 PC = R ret PC = x30</fname></fname>	·	·
b.pl label branch if non-negative b.gt label branch if greater than b.ge label branch if greater or equal b.lt label branch if less than b.le label branch if less or equal bl address <fname> x30 = PC + 4 PC = address blr R <fname> x30 = PC + 4 PC = R ret PC = x30</fname></fname>		·
b.gt label branch if greater than b.ge label branch if greater or equal b.lt label branch if less than b.le label branch if less or equal bl address <fname> x30 = PC + 4 PC = address blr R <fname> x30 = PC + 4 PC = R ret PC = x30</fname></fname>		~
b.ge label branch if greater or equal b.lt label branch if less than b.le label branch if less or equal bl address <fname> x30 = PC + 4 PC = address blr R <fname> x30 = PC + 4 PC = R ret PC = x30</fname></fname>	•	
equal b.lt label branch if less than b.le label branch if less or equal bl address <fname> x30 = PC + 4 PC = address blr R <fname> x30 = PC + 4 PC = R PC = R</fname></fname>		
b.le label branch if less or equal bl address <fname> $x30 = PC + 4$ $PC = address$ blr R <fname> $x30 = PC + 4$ $PC = R$ ret $PC = x30$</fname></fname>	b.ge label	<u> </u>
bl address <fname> $x30 = PC + 4$ PC = address blr R <fname> $x30 = PC + 4$ PC = R ret $PC = x30$</fname></fname>	b.lt label	branch if less than
bl address <fname> $x30 = PC + 4$ PC = address blr R <fname> $x30 = PC + 4$ PC = R ret $PC = x30$</fname></fname>	b.le label	branch if less or equal
blr R <fname> $x30 = PC + 4$ PC = R ret $PC = x30$</fname>	bl address <fname></fname>	
PC = R $PC = x30$	blr R <fname></fname>	
ret PC = x30		
	ret	

Addressing Modes

Register (general-purpose)

The name of the register, for example:

x0 or w0 ... x28 or w28

Note: Registers x29 - x31 are reserved by convention.

x29: frame pointer x30: link register x31: stack pointer

Immediate (constant)

A number prefixed with #. Can be decimal or hex: #8 #0x1F #-32

Memory

Access memory at the address stored in a register: [x0]

Access memory at the address stored in a register plus a constant:

[x0, #8]

Access memory at the address stored in a register plus another register:

[x0, x1]

Condition Codes

Z: Zero

N: Negative

C: Carry (unsigned overflow)

V: (Signed) overflow

Component Registers

64-bit x-prefixed registers can be accessed as 32-bit registers with a w-prefix:

