ARM64 (AArch64) Reference Sheet

Instructions

`mov D, S`	`D = S`
`ldr D, [R]`	`D = Mem[R]`
`ldp D1, D2, [R]`	`D1 = Mem[R]` `D2 = Mem[R + 8]`
`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`
`eor 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`
`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 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` `(value of x0 returned)`

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: