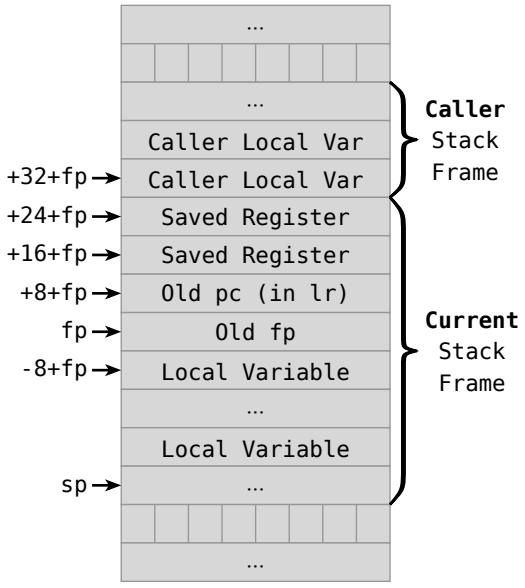
ARM64 Reference

Registers and ARM64

Register	What is it?
x0	Argument 1 and Return Register (caller-saved)
x1 ... x7	Arguments 2-8 for the current function (caller-saved)
x9 ... x15	General Purpose **caller**-saved
x19 ... x28	General Purpose **callee**-saved
(fp) x29	Frame Pointer: Address of the current stack frame's frame record
(lr) x30	Link Register: The previous function's pc
sp	Stack Pointer: Address of the top of the current stack frame
pc	Program Counter: Address of the current instruction
xzr	Zero Register: 64-bit register with the value zero

Stack and ARM64



Arithmetic Instructions

ARM64		C Pseudocode	Notes
`add xD, xM, xN`	↔	`xD = xM + xN`	`xD`, `xM`, `xN` do not need to be unique registers
`sub xD, xM, xN`	↔	`xD = xM - xN`	
`mul xD, xM, xN`	↔	`xD = xM * xN`	
`sdiv xD, xM, xN`	↔	`xD = xM / xN`	
`udiv xD, xM, xN`	↔	`xD = xM / xN`	
`and xD, xM, xN`	↔	`xD = xM & xN`	
`orr xD, xM, xN`	↔	`xD = xM | xN`	
`lsl xD, xM, xN`	↔	`xD = xM << xN`	

The third argument in add, sub, and bitops can be replaced with `#const`, an unsigned 12 bit integer.

Moving/Storing/Loading Instructions for Registers and Pairs

ARM64		C Pseudocode	Notes
`mov xD, xS`	↔	`xD = xS`	
`mov xD, #s`	↔	`xD = #s`	`#s` is typically at most size `int16_t`[a]
`movk xD, #s, lsl #off`	↔	`xD = xD | (#s << #off)`	`#off` $\in \{0, 16, 32, 48\}$[b]
`ldr xD, =const`	↔	`xD = #const`	`#const` is at most size `int64_t`
`ldr xD, [xP]`	↔	`xD = *xP`	
`str xS, [xP]`	↔	`*xP = xS`	
`ldr xD, [xP, #off]`	↔	`xD = *(xP + #off)`	`#off` may be a register instead
`str xS, [xP, #off]`	↔	`*(xP + #off) = xS`	
`ldr xD, [xP, #off]!`	↔	`xP += #off; xD = *xP`	
`str xS, [xP, #off]!`	↔	`xP += #off; *xP = xS`	
`ldr xD, [xP], #off`	↔	`xD = *xP; xP += #off`	
`str xS, [xP], #off`	↔	`*xP = xS; xP += #off`	
`stp fp, lr, [sp, #-16]!`	↔	`sp -= 16; *sp = fp; *(sp + 8) = lr`	
`ldp fp, lr, [sp], #16`	↔	`fp = *sp; lr = *(sp + 8); sp += 16`	

Note that `xP` can be replaced with `sp`.

[a] Larger immediates up to size `int64_t` can work in special cases, but `int16_t`'s are safe to use in all cases.

[b] `#s` is size `int16_t`. The `|` means that other bits in `xD` are not zeroed.

Control Flow

ARM64		C Pseudocode
`cmp xM, xN`	↔	`A = xM; B = xN`
`b.SUFFIX LABEL`	↔	`pc = &LABEL if COND`
`b LABEL`	↔	`pc = &LABEL`
`br xN`	↔	`pc = xN`
`bl LABEL`	↔	`lr = pc + 4; pc = &LABEL`
`blr xN`	↔	`lr = pc + 4; pc = xN`
`ret`	↔	`pc = lr`

Suffix		Condition	Notes
`___eq`	↔	`A == B`	
`___ne`	↔	`A != B`	
`___cs`	↔	`A >= B`	Unsigned
`___cc`	↔	`A < B`	
`___hi`	↔	`A > B`	
`___ls`	↔	`A <= B`	
`___ge`	↔	`A >= B`	Signed
`___lt`	↔	`A < B`	
`___gt`	↔	`A > B`	
`___le`	↔	`A <= B`	