AArch64 most common instructions

Containers: x (64-bit register), w (32-bit register)   if {S} is present flags will be affected
rd, rn, rm: w or x registers; op2: register, modified register or #immn (n-bit immediate)
n (of n-bit immediate values) depends heavily on context. Out of bond values will generate a syntax erro

	Instruction	Mnemonic	Syntax	Explanation	Flags
Arithmetic operations	Addition	ADD{S}	ADD{S} rd, rn, op2	rd = rn + op2	{Yes}
	Subtraction	SUB{S}	SUB{S} rd, rn, op2	rd = rn − op2	{Yes}
	Negation	NEG{S}	NEG{S} rd, op2	rd = −op2	{Yes}
	with carry	NGC{S}	NGC{S} rd, rm	rd = −rm − ~C	{Yes}
	Multiply	MUL	MUL rd, rn, rm	rd = rn x rm	
	Unsigned multiply long	UMULL	UMULL xd, wn, wm	xd = wn x wm	
	Unsigned multiply high	UMULH	UMULH xd, xn, xm	xd = <127:64> of xn x xm	
	Signed multiply long	SMULL	SMULL xd, wn, wm	xd = wm x wn (signed operands)	
	Signed multiply high	SMULH	SMULH xd, xn, xm	xd = <127:64> of xn x xm (signed operands)	
	Multiply and add	MADD	MADD rd, rn, rm, ra	rd = ra + (rn x rm)	
	Multiply and sub	MSUB	MSUB rd, rn, rm, ra	rd = ra − (rn x rm)	
	Multiply and neg	MNEG	MNEG rd, rn, rm	rd = −(rn x rm)	
	Unsigned multiply and add long	UMADDL	UMADDL xd, wn, wm, xa	xd = xa + (wm x wn)	
	Unsigned multiply and sub long	UMSUBL	UMSUBL xd, wn, wm, xa	xd = xa − (wm x wn)	
	Unsigned multiply and neg long	UMNEGL	UMNEGL xd, wn, wn	xd = −(wm x wn)	
	Signed multiply and add long	SMADDL	SMADDL xd, wn, wm, xa	xd = xa + (wm x wn)	
	Signed multiply and sub long	SMSUBL	SMSUBL xd, wn, wm, xa	xd = xa − (wm x wn)	
	Signed multiply and neg long	SMNEGL	SMNEGL xd, wn, wm	xd = − (wm x wn)	
	Unsigned divide	UDIV	UDIV rd, rn, rm	rd = rn / rm	
	Signed divide	SDIV	SDIV rd, rn, rm	rd = rn / rm	
	Note: the remainder may be computed using the MSUB instruction as numerator – (quotient x denominator)				
Bitwise logical operations	Bitwise AND	AND	AND{S} rd, rn, op2	rd = rn & op2	{Yes}
	Bitwise AND with neg	BIC	BIC{S} rd, rn, op2	rd = rn & ~op2	{Yes}
	Bitwise OR	ORR	ORR rd, rn, op2	rd = rn | op2	
	Bitwise OR with neg	ORN	ORN rd, rn, op2	rd = rn | ~op2	
	Bitwise XOR	EOR	EOR rd, rn, op2	rd = rn ⊕ op2	
	Bitwise XOR with neg	EON	EON rd, rn, op2	rd = rn ⊕ ~op2	
	Logical shift left	LSL	LSL rd, rn, op2	Logical shift left (stuffing zeros enter from right)	
	Logical shift right	LSR	LSR rd, rn, rm	Logical shift right (stuffing zeros enter from left)	
	Arithmetic shift right	ASR	ASR rd, rn, op2	Arithmetic shift right (preserves sign)	
	Rotate right	ROR	ROR rd, rn, op2	Rotate right (considering the register as a ring)	
	Move to register	MOV	MOV rd, op2	rd = op2	
	Move to register, neg	MVN	MVN rd, op2	rd = ~op2	
	Test bits	TST	TST rn, op2	rn & op2	Yes
Bitfield ops	Bitfield insert	BFI	BFI rd, rn, #lsb, #width	Moves a bitfield of #width bits starting at source bit 0 to destination starting at bit #lsb	
	Bitfield extract	UBFX	UBFZ rd, rn, #lsb, #width	Moves a bitfield of #width bits starting at source bit #lsb to destination starting at bit 0; clears all other	
	Signed bitfield extract	SBFX	SBFZ rd, rn, #lsb, #width	Moves a bitfield of #width bits starting at source bit #lsb to destination starting at bit 0; sign extends the	
Bit/Byte ops	Count leading sign	CLS	CLS rd, rm	Count leading sign bits	
	Count leading sign	CLZ	CLZ rd, rm	Count leading zero bits	
	Reverse bit	RBIT	RBIT rd, rm	Reverse bit order	
	Reverse byte	REV	REV rd, rm	Reverse byte order	
	Reverse byte in half word	REV16	REV16 rd, rm	Reverse byte order on each half word	
	Reverse byte in word	REV32	REV32 xd, xm	Reverse byte order on each word	
Load and Store operations	Store single register	STR	rt, [addr]	Mem[addr] = rt	
	Subtype byte	STRB	wt, [addr]	Byte[addr] = wt<7:0>	
	Subtype half word	STRH	wt, [addr]	HalfWord[addr] = wt<15:0>	
	unscaled address offset	STUR	STUR rt, [addr]	Mem[addr] = rt (unscaled address)	
	Store register pair	STP	STP rt, rm, [addr]	Stores rt and rm in consecutive addresses starting at addr	
	Load single register	LDR	LDR rt, [addr]	rt = Mem[addr]	
	Sub-type byte	LDRB	LDRB wt, [addr]	wt = Byte[addr] (only 32-bit containers)	
	Sub-type signed byte	LDRSB	LDRSB rt, [addr]	rt = Sbyte[addr] (signed byte)	
	Sub-type half word	LDRH	LDRH wt, [addr]	wt = HalfWord[addr] (only 32-bit containers)	
	Sub-type signed half word	LDRSH	LDRSH rt, [addr]	rt = Mem[addr] (load one half word, signed)	
	Sub-type signed word	LDRSW	LDRSW xt, [addr]	xt = Sword[addr] (signed word, only for 64-bit containers)	
	unscaled address offset	LDUR	LDUR rt, [addr]	rt = Mem[addr] (unscaled address)	
	Load register pair	LDP	LDP rt, rm, [addr]	Loads rt and rm from consecutive addresses starting at addr	

Instruction	Mnemonic	Syntax	Explanation	Flags
Branch	B	B target	Jump to target	
Branch and link	BL	BL target	Writes the addr of the next instr to X30 and jumps to target	
Return	RET	RET {Xm}	Returns from sub-routine jumping to register Xm (default: X30)	
Conditional branch	B.CC	B.cc target	If (cc) jump to target	
Compare and branch if zero	CBZ	CBZ rd, target	If (rd=0) jump to target	
Compare and branch if not zero	CBNZ	CBNZ rd, target	If (rd≠0) jump to target	
Conditional select	CSEL	CSEL rd, rn, rm, cc	If (cc) rd = rn else rd = rm	
with increment,	CSINC	CSINC rd, rn, rm, cc	If (cc) rd = rn else rd = rm+1	
with negate,	CSNEG	CSNEG rd, rn, rm, cc	If (cc) rd = rn else rd = −rm	
with invert	CSINV	CSINV rd, rn, rm, cc	If (cc) rd = rn else rd = ~rm	
Conditional set	CSET	CSET rd, cc	If (cc) rd = 1 else rd = 0	
with mask,	CSETM	CSETM rd, cc	If (cc) rd = −1 else rd = 0	
with increment,	CINC	CINC rd, rn, cc	If (cc) rd = rn+1 else rd = rn	
with negate,	CNEG	CNEG rd, rn, cc	If (cc) rd = −rn else rd = rn	
with invert	CINV	CINV rd, rn, cc	If (cc) rd = ~rn else rd = rn	
Compare	CMP	CMP rd, op2	Rd − op2	Yes
with negative	CMN	CMN rd, op2	rd − (−op2)	Yes
Conditional compare	CCMP	CCMP rd, rn, #imm4, cc	If (cc) NZCV = CMP(rd,rn) else NZCV = #imm4	Yes
with negative	CCMN	CCMP rd, rn, #imm4, cc	If (cc) NZCV = CMP(rd,−rn) else NZCV = #imm4	Yes
Note: for these instructions rn can also be an #imm5 (5-bit unsigned immediate value 0..31)				

AArch64 accessory information

Condition codes (magnitude of operands)		
LO	Lower, unsigned	C = 0
HI	Higher, unsigned	C = 1 and Z = 0
LS	Lower or same, unsigned	C = 0 or Z = 1
HS	Higher or same, unsigned	C = 1
LT	Less than, signed	N != V
GT	Greater than, signed	Z = 0 and N = V
LE	Less than or equal, signed	Z = 1 and N != V
GE	Greater than or equal, signed	N = V

Condition codes (direct flags)		
EQ	Equal	Z = 1
NE	Not equal	Z = 0
MI	Negative	N = 1
PL	Positive or zero	N = 0
VS	Overflow	V = 1
VC	No overflow	V = 0
CS	Carry	C = 0
CC	No carry	C = 1

Sub types (suffix of some instructions)		
B/SB	byte/signed byte	8 bits
H/SH	half word/signed half wor	16 bits
W/SW	word/signed word	32 bits

Flags set to 1 when:	
N	the result of the last operation was negative, cleared to 0 otherwise
Z	the result of the last operation was zero, cleared to 0 otherwise
C	the last operation resulted in a carry, cleared to 0 otherwise
V	the last operation caused overflow, cleared to 0 otherwise

Sizes, in Assembly and C		
8	byte	char
16	Half word	short int
32	word	int
64	double word	long int
128	quad word	-

Addressing modes (base: register; offset: register or immediate)	
[base]	MEM[base]
[base, offset]	MEM[base+offset]
[base, offset]!	MEM[base+offset] then base = base + offset (pre indexed)
[base], offset	MEM[base] then base = base + offset (post indexed)

Calling convention (register use)
Params: X0..X7; Result: X0
Reserved: X8, X16..X18 (do not use these)
Unprotected: X9..X15 (callee may corrupt)
Protected: X19..X28 (callee must preserve)

Op2 processing (applied to Op2 before anything else)	
LSL|LSR|ASR #imm6	
SXTW / SXTB {#imm2}	Sign extension/Sign extension after LSL #imm2