Aarch64 most common instructions

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

	Instruction	Mnemonic	Syntax	Explanation	Flags
Arithmetic	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}
	Unsigned multiply	MUL	MUL rd, rn, rm	rd = rn x rm	
	Unsigned divide	UDIV	UDIV rd, rn, rm	rd = rn / rm	

~

Bitwise logical	Bitwise AND	AND	AND{S} rd, rn, op2	rd = rn & op2	{Yes}
	Bitwise OR	ORR	ORR rd, rn, op2	rd = rn | op2	
	Bitwise XOR	EOR	EOR 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 (carry not involved)	
	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

Load and Store	Store single register	STUR	STUR rt, [rn {, imm9}]	addr=rn+imm9, Mem[addr+7:addr] = rt	
	Load single register	LDUR	LDUR rt, [rn {, imm9}]	addr=rn+imm9, rt = Mem[addr+7:addr],	
	Sub-type signed word	LDRSW	LDRSW rt, [rn {, imm9}]	addr=rn+imm9, rt = Mem[addr+3:addr]	

	Instruction	Mnemonic	Syntax	Explanation	Flags
Branch operations	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 operations	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	

Comparisons	Compare	CMP	CMP rd, op2	rd − op2	Yes
	with negative	CMN	CMN rd, op2	rd − (−op2)	Yes
	Test	TST	TST rd, op2	rd AND (op2)	Yes

Aarch64 general 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 word	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

Addressing modes (base: register; offset: register or immediate)	
[base]	MEM[base]
[base, offset]	MEM[base+offset]

Calling convention (register use)
Params: X0..X7; Result: X0
Reserved: X8, X16..X18
Unprotected: X9..X15 (may alter)
Protected: X19..X28 (must preserve)