ARMv8 AArch64 Quick Reference (adapted from: https://courses.cs.washington.edu/courses/cse469/18wi/Materials/arm64.pdf)

Arithmetic Instructions			
ADC{S}	rd, rn, rm	rd = rn + rm + C	
ADD{S}	rd, rn, op2	rd = rn + op2	S
ADR	Xd, ±rel$_{21}$	Xd = PC + rel$^{\pm}$	
ADRP	Xd, ±rel$_{33}$	Xd = PC$_{63:12}$:0$_{12}$ + rel$^{\pm}_{33:12}$:0$_{12}$	
CMN	rd, op2	rd + op2	S
CMP	rd, op2	rd − op2	S
MADD	rd, rn, rm, ra	rd = ra + rn × rm	
MNEG	rd, rn, rm	rd = − rn × rm	
MSUB	rd, rn, rm, ra	rd = ra − rn × rm	
MUL	rd, rn, rm	rd = rn × rm	
NEG{S}	rd, op2	rd = −op2	
NGC{S}	rd, rm	rd = −rm − ∼C	
SBC{S}	rd, rn, rm	rd = rn − rm − ∼C	
SDIV	rd, rn, rm	rd = rn ÷ rm	
SMADDL	Xd, Wn, Wm, Xa	Xd = Xa + Wn × Wm	
SMNEGL	Xd, Wn, Wm	Xd = − Wn × Wm	
SMSUBL	Xd, Wn, Wm, Xa	Xd = Xa − Wn × Wm	
SMULH	Xd, Xn, Xm	Xd = (Xn × Xm)$_{127:64}$	
SMULL	Xd, Wn, Wm	Xd = Wn × Wm	
SUB{S}	rd, rn, op2	rd = rn - op2	S
UDIV	rd, rn, rm	rd = rn ÷ rm	
UMADDL	Xd, Wn, Wm, Xa	Xd = Xa + Wn × Wm	
UMNEGL	Xd, Wn, Wm	Xd = − Wn × Wm	
UMSUBL	Xd, Wn, Wm, Xa	Xd = Xa − Wn × Wm	
UMULH	Xd, Xn, Xm	Xd = (Xn × Xm)$_{127:64}$	
UMULL	Xd, Wn, Wm	Xd = Wn × Wm	

Registers	
X0-X7	Arguments and return values
X8	Indirect result
X9-X15	Temporary
X16-X17	Intra-procedure-call temporary
X18	Platform defined use
X19-X28	Temporary (must be preserved)
X29	Frame pointer (must be preserved)
X30	Return address
SP	Stack pointer
XZR	Zero
PC	Program counter

Logical and Mov Instructions			
AND{S}	rd, rn, op2	rd = rn & op2	
ASR	rd, rn, rm	rd = rn >> rm	
ASR	rd, rn, #i$_6$	rd = rn >> i	
BIC{S}	rd, rn, op2	rd = rn & ∼op2	
EON	rd, rn, op2	rd = rn ⊕ ∼op2	
EOR	rd, rn, op2	rd = rn ⊕ op2	
LSL	rd, rn, rm	rd = rn << rm	
LSL	rd, rn, #i$_6$	rd = rn << i	
LSR	rd, rn, rm	rd = rn >> rm	
LSR	rd, rn, #i$_6$	rd = rn >> i	
MOV	rd, rn	rd = rn	S
MOV	rd, #i	rd = i	
MOVK	rd,#i$_{16}${, sh}	rd$_{sh+15:sh}$ = i	
MOVN	rd,#i$_{16}${, sh}	rd = ~(i$^{\emptyset}$<< sh)	
MOVZ	rd,#i$_{16}${, sh}	rd = i$^{\emptyset}$<< sh	
MVN	rd, op2	rd = ∼op2	
ORN	rd, rn, op2	rd = rn | ∼op2	
ORR	rd, rn, op2	rd = rn | op2	
ROR	rd, rn, #i$_6$	rd = rn ≫ i	
ROR	rd, rn, rm	rd = rn ≫ rm	
TST	rn, op2	rn & op2	

Branch Instructions		
B	rel28	PC = PC + rel$^{\pm}_{27:2}$:0$_2$
Bcc	rel21	if(cc) PC = PC + rel$^{\pm}_{20:2}$:0$_2$
BL	rel28	X30 = PC + 4; PC += rel$^{\pm}_{27:2}$:0$_2$
BLR	Xn	X30 = PC + 4; PC = Xn
BR	Xn	PC = Xn
CBNZ	rn, rel21	if(rn 6= 0) PC += rel$^{\emptyset}_{21:2}$:0$_2$
CBZ	rn, rel21	if(rn = 0) PC += rel$^{\emptyset}_{21:2}$:0$_2$
RET	{Xn}	PC = Xn
TBNZ	rn, #i, rel16	if(rni 6= 0) PC += rel$^{\pm}_{15:2}$:0$_2$
TBZ	rn, #i, rel16	if(rni = 0) PC += rel$^{\pm}_{15:2}$:0$_2$

Conditional Instructions		
CCMN	rn, #i5, #f4, cc	if(cc) rn + i; else N:Z:C:V = f
CCMN	rn, rm, #f4, cc	if(cc) rn + rm; else N:Z:C:V = f
CCMP	rn, #i5, #f4, cc	if(cc) rn − i; else N:Z:C:V = f
CCMP	rn, rm, #f4, cc	if(cc) rn − rm; else N:Z:C:V = f
CINC	rd, rn, cc	if(cc) rd = rn + 1; else rd = rn
CINV	rd, rn, cc	if(cc) rd = ∼rn; else rd = rn
CNEG	rd, rn, cc	if(cc) rd = −rn; else rd = rn
CSEL	rd, rn, rm, cc	if(cc) rd = rn; else rd = rm
CSET	rd, cc	if(cc) rd = 1; else rd = 0
CSETM	rd, cc	if(cc) rd = ∼0; else rd = 0
CSINC	rd, rn, rm, cc	if(cc) rd = rn; else rd = rm + 1
CSINV	rd, rn, rm, cc	if(cc) rd = rn; else rd = ∼rm
CSNEG	rd, rn, rm, cc	if(cc) rd = rn; else rd = −rm

Load and Store Instructions		
LDP	rt, rt2, [addr]	rt2:rt = [addr]$_{2N}$
LDPSW	Xt, Xt2, [addr]	Xt = [addr]$^{\pm}_{32}$; Xt2 = [addr+4]$^{\pm}_{32}$
LD{U}R	rt, [addr]	rt = [addr]$_N$
LD{U}R{B,H}	Wt, [addr]	Wt = [addr]$^{\emptyset}_N$
LD{U}RS{B,H}	rt, [addr]	rt = [addr]$^{\pm}_N$
LD{U}RSW	Xt, [addr]	Xt = [addr]$^{\pm}_{32}$
STP	rt, rt2, [addr]	[addr]$_{2N}$ = rt2:rt
ST{U}R	rt, [addr]	[addr]N = rt
ST{U}R{B,H}	Wt, [addr]	[addr]$_N$ = Wt$_{N0}$

Addressing Modes (addr)		
xxP,LDPSW	[Xn{, #i$_{7+s}$}]	addr = Xn + i$_{6+s:s}$$^{\pm}0_s$
xxP,LDPSW	[Xn], #i$_{7+s}$	addr=Xn; Xn+=i$_{6+s:s}$$^{\pm}0_s$
xxP,LDPSW	[Xn, #i$_{7+s}$]!	Xn+=i$_{6+s:s}$$^{\pm}0_s$; addr=Xn
xxR*	[Xn], #i$_9$	addr = Xn; Xn += i$^{\pm}$
xxR*	[Xn, #i$_9$]!	Xn += i$^{\pm}$; addr = Xn

Notes for Instruction Set
S SP/WSP may be used as operand(s) instead of XZR/WZR

Bit Manipulation Instructions		
BFI	rd, rn, #p, #n	$^{rd}p+n-1:p = {}^{rn}n-1:0$
BFXIL	rd, rn, #p, #n	$^{rd}n-1:0 = {}^{rn}p+n-1:p$
CLS	rd, rn	rd = CountLeadingOnes(rn)
CLZ	rd, rn	rd = CountLeadingZeros(rn)
EXTR	rd, rn, rm, #p	$rd = rn_{p-1:0}:{}^{rm}{}_{N0}$
RBIT	rd, rn	rd = ReverseBits(rn)
REV	rd, rn	rd = BSwap(rn)
REV16	rd, rn	for(n=0..1|3) rd_{Hn}=BSwap(rn_{Hn})
REV32	Xd, Xn	Xd=BSwap($Xn_{63:32}$):BSwap($Xn_{31:0}$)
{S,U}BFIZ	rd, rn, #p, #n	$rd = rn^{?}{}_{n-1:0} \ll p$
{S,U}BFX	rd, rn, #p, #n	$rd = rn^{?}{}_{p+n-1:p}$
{S,U}XT{B,H}	rd, Wn	$rd = Wn^{?}{}_{N0}$
SXTW	Xd, Wn	$Xd = Wn^{\pm}$

Condition Codes (cc)		
EQ	Equal	Z
NE	Not equal	!Z
CS/HS	Carry set, Unsigned higher or same	C
CC/LO	Carry clear, Unsigned lower	!C
MI	Minus, Negative	N
PL	Plus, Positive or zero	!N
VS	Overflow	V
VC	No overflow	!V
HI	Unsigned higher	C & !Z
LS	Unsigned lower or same	!C | Z
GE	Signed greater than or equal	N = V
LT	Signed less than	N ≠V
GT	Signed greater than	!Z & N = V
LE	Signed less than or equal	Z | N ≠V
AL	Always (default)	1

Operand 2 (op2)		
All	*rm*	rm
all	rm, LSL #i_6	rm << i
all	rm, LSR #i_6	rm >> ii
all	rm, ASR #i_6	rm >> i
logical	rm, ROR #i_6	rm >>> i
arithmetic	Wm, {S,U}XTB{ #i_3}	$Wm^{?}{}_{B0}$ << i
arithmetic	Wm, {S,U}XTH{ #i_3}	$Wm^{?}{}_{H0}$ << i
arithmetic	Wm, {S,U}XTW{ #i_3}	$Wm^{?}$ << i
arithmetic	Xm, {S,U}XTX{ #i_3}	$Xm^{?}$ << i
arithmetic	#i_{12}	$i^{\emptyset}$
arithmetic	#i_{24}	$i^{\emptyset}_{23:12}:0_{12}$
AND,EOR,ORR,TST	#mask	mask

Keys	
N	Operand bit size (8, 16, 32 or 64)
s	Operand log byte size (0=byte,1=hword,2=word,3=dword)
rd, rn, rm, rt	General register of either size (Wn or Xn)
{,sh}	Optional halfword left shift (LSL #{16,32,48})
$val^{\pm}$, $val^{\emptyset}$, $val^{?}$	Value is sign/zero extended (? depends on instruction)
x ÷ >> < >	Operation is signed