ARMv8 A64 Quick Reference

Arithmetic Instructions

ADC{S}	rd, rn, rm	rd = rn + rm + C	
ADD{S}	rd, rn, op2	rd = rn + op2	S
ADR	Xd, $\pm rel_{21}$	Xd = PC + $rel^{\pm}$	
ADRP	Xd, $\pm rel_{33}$	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 $\bar{\div}$ rm	
SMADDL	Xd, Wn, Wm, Xa	Xd = Xa + Wn $\bar{\times}$ Wm	
SMNEGL	Xd, Wn, Wm	Xd = − Wn $\bar{\times}$ Wm	
SMSUBL	Xd, Wn, Wm, Xa	Xd = Xa − Wn $\bar{\times}$ Wm	
SMULH	Xd, Xn, Xm	Xd = $(Xn \bar{\times} Xm)_{127:64}$	
SMULL	Xd, Wn, Wm	Xd = Wn $\bar{\times}$ 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 = $(Xn \times Xm)_{127:64}$	
UMULL	Xd, Wn, Wm	Xd = Wn × Wm	

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 = $Wn^{\pm}$	

Logical and Move Instructions

AND{S}	rd, rn, op2	rd = rn & op2	
ASR	rd, rn, rm	rd = rn $\bar{\gg}$ rm	
ASR	rd, rn, $\#i_6$	rd = rn $\bar{\gg}$ 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 = $\sim(i^{\emptyset} \ll sh)$	
MOVZ	rd,$\#i_{16}${, sh}	rd = $i^{\emptyset} \ll sh$	
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	rel_{28}	PC = PC + $rel^{\pm}_{27:2}{:}0_2$	
Bcc	rel_{21}	if(cc) PC = PC + $rel^{\pm}_{20:2}{:}0_2$	
BL	rel_{28}	X30 = PC + 4; PC += $rel^{\pm}_{27:2}{:}0_2$	
BLR	Xn	X30 = PC + 4; PC = Xn	
BR	Xn	PC = Xn	
CBNZ	rn, rel_{21}	if(rn ≠ 0) PC += $rel^{\emptyset}_{21:2}{:}0_2$	
CBZ	rn, rel_{21}	if(rn = 0) PC += $rel^{\emptyset}_{21:2}{:}0_2$	
RET	{Xn}	PC = Xn	
TBNZ	rn, #i, rel_{16}	if(rn_i ≠ 0) PC += $rel^{\pm}_{15:2}{:}0_2$	
TBZ	rn, #i, rel_{16}	if(rn_i = 0) PC += $rel^{\pm}_{15:2}{:}0_2$	

Atomic Instructions

CAS{A}{L}	rs, rt, [Xn]	if (rs = $[Xn]_N$) $[Xn]_N$ = rt	1
CAS{A}{L}{B,H}	Ws, Wt, [Xn]	if (Ws_{N0} = $[Xn]_N$) $[Xn]_N$ = Wt_{N0}	1
CAS{A}{L}P	rs,rs2,rt,rt2,[Xn]	if (rs2:rs = $[Xn]_{2N}$) $[Xn]_{2N}$ = rt2:rt	1
LDao{A}{L}{B,H}	Ws, Wt, [Xn]	Wt=$[Xn]^{\emptyset}_N$; $[Xn]_N$=ao($[Xn]_N$,Ws_{N0})	1
LDao{A}{L}	rs, rt, [Xn]	rt = $[Xn]_N$; $[Xn]_N$ = ao($[Xn]_N$, rs)	1
STao{A}{L}{B,H}	Ws, [Xn]	$[Xn]_N$ = ao($[Xn]_N$, Ws_{N0})	1
STao{A}{L}	rs, [Xn]	$[Xn]_N$ = ao($[Xn]_N$, rs)	1
SWP{A}{L}{B,H}	Ws, Wt, [Xn]	Wt = $[Xn]^{\emptyset}_N$; $[Xn]_N$ = Ws_{N0}	1
SWP{A}{L}	rs, rt, [Xn]	rt = $[Xn]_N$; $[Xn]_N$ = rs	1

Conditional Instructions

CCMN	rn, $\#i_5$, $\#f_4$, cc	if(cc) rn + i; else N:Z:C:V = f	
CCMN	rn, rm, $\#f_4$, cc	if(cc) rn + rm; else N:Z:C:V = f	
CCMP	rn, $\#i_5$, $\#f_4$, cc	if(cc) rn − i; else N:Z:C:V = f	
CCMP	rn, rm, $\#f_4$, 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 = $[addr]^{\pm}_{32}$	
PRFM	prfop, addr	Prefetch(addr, prfop)	
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 + $i^{\pm}_{6+s:s}{:}0_s$
xxP,LDPSW	[Xn], $\#i_{7+s}$	addr=Xn; Xn+=$i^{\pm}_{6+s:s}{:}0_s$
xxP,LDPSW	[Xn, $\#i_{7+s}$]!	Xn+=$i^{\pm}_{6+s:s}{:}0_s$; addr=Xn
xxR*,PRFM	[Xn{, $\#i_{12+s}$}]	addr = Xn + $i^{\emptyset}_{11+s:s}{:}0_s$
xxR*	[Xn], $\#i_9$	addr = Xn; Xn += $i^{\pm}$
xxR*	[Xn, $\#i_9$]!	Xn += $i^{\pm}$; addr = Xn
xxR*,PRFM	[Xn,Xm{, LSL #0|s}]	addr = Xn + Xm ≪ s
xxR*,PRFM	[Xn,Wm,{S,U}XTW{ #0|s}]	addr = Xn + $Wm^{?} \ll s$
xxR*,PRFM	[Xn,Xm,SXTX{ #0|s}]	addr = Xn + $Xm^{\pm} \ll s$
xxUR*,PRFM	[Xn{, $\#i_9$}]	addr = Xn += $i^{\pm}$
LDR{SW},PRFM	$\pm rel_{21}$	addr = PC + $rel^{\pm}_{20:2}{:}0_2$

Atomic Operations (ao)

ADD	[Xn] + rs	SMAX	[Xn] $\bar{>}$ rs ? [Xn] : rs
CLR	[Xn] & ~rs	SMIN	[Xn] $\bar{<}$ rs ? [Xn] : rs
EOR	[Xn] ⊕ rs	UMAX	[Xn] > rs ? [Xn] : rs
SET	[Xn] | rs	UMIN	[Xn] < rs ? [Xn] : rs

Operand 2 (op2)

all	*rm*	rm
all	rm, LSL #i_6	rm $\ll$ i
all	rm, LSR #i_6	rm $\gg$ i
all	rm, ASR #i_6	rm $\bar{\gg}$ i
logical	rm, ROR #i_6	rm $\ggg$ i
arithmetic	Wm, {S,U}XTB{ #i_3}	$Wm^{?}_{B0} \ll i$
arithmetic	Wm, {S,U}XTH{ #i_3}	$Wm^{?}_{H0} \ll i$
arithmetic	Wm, {S,U}XTW{ #i_3}	$Wm^{?} \ll i$
arithmetic	Xm, {S,U}XTX{ #i_3}	$Xm^{?} \ll i$
arithmetic	#i_{12}	$i^{\emptyset}$
arithmetic	#i_{24}	$i^{\emptyset}_{23:12}:0_{12}$
AND,EOR,ORR,TST	#mask	mask

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

Special Purpose Registers

SPSR_EL{1..3}	Process state on exception entry to EL{1..3}	64
ELR_EL{1..3}	Exception return address from EL{1..3}	
SP_EL{0..2}	Stack pointer for EL{0..2}	64
SPSel	SP selection (0: SP=SP_EL0, 1: SP=SP_ELn)	
CurrentEL	Current Exception level (at bits 3..2)	RO
DAIF	Current interrupt mask bits (at bits 9..6)	
NZCV	Condition flags (at bits 31..28)	
FPCR	Floating-point operation control	
FPSR	Floating-point status	

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)
prfop	P{LD,LI,ST}L{1..3}{KEEP,STRM}
{,sh}	Optional halfword left shift (LSL #{16,32,48})
$val^{\pm}$, $val^{\emptyset}$, $val^{?}$	Value is sign/zero extended (? depends on instruction)
$\bar{\times}$ $\bar{\div}$ $\bar{\gg}$ $\bar{>}$ $\bar{<}$	Operation is signed

Checksum Instructions

CRC32{B,H}	Wd, Wn, Wm	Wd=CRC32(Wn,0x04c11db7,Wm_{N0})	
CRC32W	Wd, Wn, Wm	Wd = CRC32(Wn,0x04c11db7,Wm)	
CRC32X	Wd, Wn, Xm	Wd = CRC32(Wn,0x04c11db7,Xm)	
CRC32C{B,H}	Wd, Wn, Wm	Wd=CRC32(Wn,0x1edc6f41,Wm_{N0})	
CRC32CW	Wd, Wn, Wm	Wd = CRC32(Wn,0x1edc6f41,Wm)	
CRC32CX	Wd, Wn, Xm	Wd = CRC32(Wn,0x1edc6f41,Xm)	

Load and Store Instructions with Attribute

LD{A}XP	rt, rt2, [Xn]	rt:rt2 = [Xn, <SetExclMonitor>]$_{2N}$	
LD{A}{X}R	rt, [Xn]	rt = [Xn, <SetExclMonitor>]$_{N}$	
LD{A}{X}R{B,H}	Wt, [Xn]	Wt = [Xn, <SetExclMonitor>]$^{\emptyset}_{N}$	
LDNP	rt,rt2,[Xn{,#i_{7+s}}	rt2:rt = [Xn + $i^{\pm}_{6+s:s}$:0_s, <Temp>]$_{2N}$	
LDTR	rt, [Xn{, #i_9}]	rt = [Xn += $i^{\pm}$, <Unpriv>]$_{N}$	
LDTR{B,H}	Wt, [Xn{, #i_9}]	Wt = [Xn += $i^{\pm}$, <Unpriv>]$^{\emptyset}_{N}$	
LDTRS{B,H}	rt, [Xn{, #i_9}]	rt = [Xn += $i^{\pm}$, <Unpriv>]$^{\pm}_{N}$	
LDTRSW	Xt, [Xn{, #i_9}]	Xt = [Xn += $i^{\pm}$, <Unpriv>]$^{\pm}_{32}$	
STLR	rt, [Xn]	[Xn, <Release>]$_{N}$ = rt	
STLR{B,H}	Wt, [Xn]	[Xn, <Release>]$_{N}$ = Wt_{N0}	
ST{L}XP	Wd, rt, rt2, [Xn]	[Xn, <Excl>]$_{2N}$=rt:rt2; Wd=fail?1:0	
ST{L}XR	Wd, rt, [Xn]	[Xn, <Excl>]$_{N}$=rt; Wd=fail?1:0	
ST{L}XR{B,H}	Wd, Wt, [Xn]	[Xn, <Excl>]$_{N}$=Wt_{N0}; Wd=fail?1:0	
STNP	rt,rt2,[Xn{,#i_{7+s}}	[Xn + $i^{\pm}_{6+s:s}$:0_s, <Temp>]$_{2N}$ = rt2:rt	
STTR	rt, [Xn{, #i_9}]	[Xn += $i^{\pm}$, <Unpriv>]$_{N}$ = rt	
STTR{B,H}	Wt, [Xn{, #i_9}]	[Xn += $i^{\pm}$, <Unpriv>]$_{N}$ = Wt_{N0}	

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

Notes for Instruction Set

S	SP/WSP may be used as operand(s) instead of XZR/WZR
1	Introduced in ARMv8.1

System Instructions

AT	S1{2}E{0..3}{R,W}, Xn	PAR_EL1 = AddrTrans(Xn)	
BRK	#i_{16}	SoftwareBreakpoint(i)	
CLREX	{#i_4}	ClearExclusiveLocal()	
DMB	barrierop	DataMemoryBarrier(barrierop)	
DSB	barrierop	DataSyncBarrier(barrierop)	
ERET		PC=ELR_ELn;PSTATE=SPSR_ELn	
HVC	#$_{16}$	CallHypervisor(i)	
ISB	{SY}	InstructionSyncBarrier(SY)	
MRS	Xd, sysreg	Xd = sysreg	
MSR	sysreg, Xn	sysreg = Xn	
MSR	SPSel, #i_1	PSTATE.SP = i	
MSR	DAIFSet, #i_4	PSTATE.DAIF |= i	
MSR	DAIFClr, #i_4	PSTATE.DAIF &= ~i	
NOP			
SEV		SendEvent()	
SEVL		EventRegisterSet()	
SMC	#i_{16}	CallSecureMonitor(i)	
SVC	#i_{16}	CallSupervisor(i)	
WFE		WaitForEvent()	
WFI		WaitForInterrupt()	
YIELD			

Cache and TLB Maintenance Instructions

DC	{C,CI,I}SW, Xx	DC clean and/or inv by Set/Way	
DC	{C,CI,I}VAC, Xx	DC clean and/or inv by VA to PoC	
DC	CVAU, Xx	DC clean by VA to PoU	
DC	ZVA, Xx	DC zero by VA (len in DCZID_EL0)	
IC	IALLU{IS}	IC inv all to PoU	
IC	IVAU, Xx	IC inv VA to PoU	
TLBI	ALLE{1..3}{IS}	TLB inv all	
TLBI	ASIDE1{IS}, Xx	TLB inv by ASID	
TLBI	IPAS2{L}E1{IS}, Xx	TLB inv by IPA {last level}	
TLBI	VAA{L}E1{IS}, Xx	TLB inv by VA, all ASID {last level}	
TLBI	VA{L}E{1..3}{IS}, Xx	TLB inv by VA {last level}	
TLBI	VMALL{S12}E1{IS}	TLB inv by VMID, all, at stage 1{&2}	

DMB and DSB Options

OSH{,LD,ST}	Outer shareable, {all,load,store}
NSH{,LD,ST}	Non-shareable, {all,load,store}
ISH{,LD,ST}	Inner shareable, {all,load,store}
LD	Full system, load
ST	Full system, store
SY	Full system, all

ARMv8-A System

Control and Translation Registers

SCTLR_EL{1..3}	System Control	
ACTLR_EL{1..3}	Auxiliary Control	64
CPACR_EL1	Architectural Feature Access Control	
HCR_EL2	Hypervisor Configuration	64
CPTR_EL{2,3}	Architectural Feature Trap	
HSTR_EL2	Hypervisor System Trap	
HACR_EL2	Hypervisor Auxiliary Control	
SCR_EL3	Secure Configuration	
TTBR0_EL{1..3}	Translation Table Base 0 (4/16/64kb aligned)	64
TTBR1_EL1	Translation Table Base 1 (4/16/64kb aligned)	64
TCR_EL{1..3}	Translation Control	64
VTTBR_EL2	Virt Translation Table Base (4/16/64kb aligned)	64
VTCR_EL2	Virt Translation Control	
{A}MAIR_EL{1..3}	{Auxiliary} Memory Attribute Indirection	64
LOR{S,E}A_EL1	LORegion {Start,End} Address	64,1
LOR{C,N,ID}_EL1	LORegion {Control,Number,ID}	64,1

Exception Registers

AFSR{0,1}_EL{1..3}	Auxiliary Fault Status {0,1}	
ESR_EL{1..3}	Exception Syndrome	
FAR_EL{1..3}	Fault Address	64
HPFAR_EL2	Hypervisor IPA Fault Address	64
PAR_EL1	Physical Address	64
VBAR_EL{1..3}	Vector Base Address (2kb aligned)	64
RVBAR_EL{1..3}	Reset Vector Base Address	RO,64
RMR_EL{1..3}	Reset Management	
ISR_EL1	Interrupt Status	RO

Performance Monitors Registers

PMCR_EL0	PM Control	
PMCNTEN{SET,CLR}_EL0	PM Count Enable {Set,Clear}	
PMOVSCLR_EL0	PM Overflow Flag Status Clear	
PMSWINC_EL0	PM Software Increment	WO
PMSELR_EL0	PM Event Counter Selection	
PMCEID{0,1}_EL0	PM Common Event ID {0,1}	RO
PMCCNTR_EL0	PM Cycle Count Register	64
PMXEVTYPER_EL0	PM Selected Event Type	
PMXEVCNTR_EL0	PM Selected Event Count	
PMUSERENR_EL0	PM User Enable	
PMOVSSET_EL0	PM Overflow Flag Status Set	
PMINTEN{SET,CLR}_EL1	PM Interrupt Enable {Set,Clear}	
PMEVCNTR{0..30}_EL0	PM Event Count {0..30}	
PMEVTYPER{0..30}_EL0	PM Event Type {0..30}	
PMCCFILTR_EL0	PM Cycle Count Filter	

ID Registers

MIDR_EL1	Main ID	RO
MPIDR_EL1	Multiprocessor Affinity	RO,64
REVIDR_EL1	Revision ID	RO
CCSIDR_EL1	Current Cache Size ID	RO
CLIDR_EL1	Cache Level ID	RO
AIDR_EL1	Auxiliary ID	RO
CSSELR_EL1	Cache Size Selection	
CTR_EL0	Cache Type	RO
DCZID_EL0	Data Cache Zero ID	RO
VPIDR_EL2	Virtualization Processor ID	
VMPIDR_EL2	Virtualization Multiprocessor ID	64
ID_AA64PFR{0,1}_EL1	AArch64 Processor Feature {0,1}	RO,64
ID_AA64DFR{0,1}_EL1	AArch64 Debug Feature {0,1}	RO,64
ID_AA64AFR{0,1}_EL1	AArch64 Auxiliary Feature {0,1}	RO,64
ID_AA64ISAR{0,1}_EL1	AArch64 Instruction Set Attribute {0,1}	RO,64
ID_AA64MMFR{0,1}_EL1	AArch64 Memory Model Feature {0,1}	RO,64
CONTEXTIDR_EL1	Context ID	
TPIDR_EL{0..3}	Software Thread ID	64
TPIDRRO_EL0	EL0 Read-only Software Thread ID	64

Exception Vectors

0x000,0x080,0x100,0x180	{Sync,IRQ,FIQ,SError} from cur lvl with SP_EL0
0x200,0x280,0x300,0x380	{Sync,IRQ,FIQ,SError} from cur lvl with SP_ELn
0x400,0x480,0x500,0x580	{Sync,IRQ,FIQ,SError} from lower lvl using A64
0x600,0x680,0x700,0x780	{Sync,IRQ,FIQ,SError} from lower lvl using A32

System Control Register (SCTLR)

M	0x00000001	MMU enabled	
A	0x00000002	Alignment check enabled	
C	0x00000004	Data and unified caches enabled	
SA	0x00000008	Enable SP alignment check	
SA0	0x00000010	Enable SP alignment check for EL0	E1
UMA	0x00000200	Trap EL0 access of DAIF to EL1	E1
I	0x00001000	Instruction cache enabled	
DZE	0x00004000	Trap EL0 DC instruction to EL1	E1
UCT	0x00008000	Trap EL0 access of CTR_EL0 to EL1	E1
nTWI	0x00010000	Trap EL0 WFI instruction to EL1	E1
nTWE	0x00040000	Trap EL0 WFE instruction to EL1	E1
WXN	0x00080000	Write permission implies XN	
SPAN	0x00800000	Set privileged access never	E1,1
E0E	0x01000000	Data at EL0 is big-endian	E1
EE	0x02000000	Data at EL1 is big-endian	
UCI	0x04000000	Trap EL0 cache instructions to EL1	E1

Generic Timer Registers

CNTFRQ_EL0	Ct Frequency (in Hz)	
CNT{P,V}CT_EL0	Ct {Physical,Virtual} Count	RO,64
CNTVOFF_EL2	Ct Virtual Offset	64
CNTHCTL_EL2	Ct Hypervisor Control	
CNTKCTL_EL1	Ct Kernel Control	
CNT{P,V}_{TVAL,CTL,CVAL}_EL0	Ct {Physical,Virtual} Timer	
CNTHP_{TVAL,CTL,CVAL}_EL2	Ct Hypervisor Physical Timer	
CNTPS_{TVAL,CTL,CVAL}_EL1	Ct Physical Secure Timer	
CNTHV_{TVAL,CTL,CVAL}_EL2	Ct Virtual Timer	1

Exception Classes

0x00	Unknown reason
0x01	Trapped WFI or WFE instruction execution
0x07	Trapped access to SIMD/FP
0x08	Trapped VMRS access
0x0e	Illegal Execution state
0x11,0x15	SVC instruction execution in AArch{32,64} state
0x12,0x16	HVC instruction execution in AArch{32,64} state
0x13,0x17	SMC instruction execution in AArch{32,64} state
0x18	Trapped MSR, MRS, or System instruction execution
0x1f	Implementation defined exception to EL3
0x20,0x21	Instruction Abort from {lower,current} level
0x22,0x26	{PC,SP} alignment fault
0x24,0x25	Data Abort from {lower,current} level
0x28,0x2c	Trapped float-point exception from AArch{32,64} state
0x2f	SError interrupt
0x30,0x31	Breakpoint exception from {lower,current} level
0x32,0x33	Software Step exception from {lower,current} level
0x34,0x35	Watchpoint exception from {lower,current} level
0x38,0x3c	{BKPT,BRK} instruction excecution from AArch{32,64} state

Secure Configuration Register (SCR)

NS	0x0001	System state is non-secure unless in EL3	
IRQ	0x0002	IRQs taken to EL3	
FIQ	0x0004	FIQs taken to EL3	
EA	0x0008	External aborts and SError taken to EL3	
SMD	0x0080	Secure monitor call disable	
HCE	0x0100	Hyp Call enable	
SIF	0x0200	Secure instruction fetch	
RW	0x0400	Lower level is AArch64	
ST	0x0800	Trap secure EL1 to CNTPS registers to EL3	
TWI	0x1000	Trap EL{0..2} WFI instruction to EL3	
TWE	0x2000	Trap EL{0..2} WFE instruction to EL3	
TLOR	0x4000	Trap LOR registers	1