

①



# LEGv8 Reference Data

## CORE INSTRUCTION SET in Alphabetical Order by Mnemonic

NAME, MNEMONIC	FOR- MAT	OPCODE (9) (Hex)	OPERATION (in Verilog)	Notes
ADD	ADD	R 458	$R[Rd] = R[Rn] + R[Rm]$	
ADD Immediate	ADDI	I 488-489	$R[Rd] = R[Rn] + ALUImm$	(2,9)
ADD Immediate & Set flags	ADDIS	I 588-589	$R[Rd], FLAGS = R[Rn] + ALUImm$	(1,2,9)
ADD & Set flags	ADDS	R 558	$R[Rd], FLAGS = R[Rn] + R[Rm]$	(1)
AND	AND	R 450	$R[Rd] = R[Rn] \& R[Rm]$	
AND Immediate	ANDI	I 490-491	$R[Rd] = R[Rn] \& ALUImm$	(2,9)
AND Immediate & Set flags	ANDIS	I 790-791	$R[Rd], FLAGS = R[Rn] \& ALUImm$	(1,2,9)
AND & Set flags	ANDS	R 750	$R[Rd], FLAGS = R[Rn] \& R[Rm]$	(1)
Branch	B	0A0-0BF	$PC = PC + BranchAddr$	(3,9)
Branch conditionally	B, cond	CB 2A0-2A7	if(FLAGS==cond) $PC = PC + CondBranchAddr$	(4,9)
Branch with Link	BL	B 4A0-4BF	$R[30] = PC + 4;$ $PC = PC + BranchAddr$	(3,9)
Branch to Register	BR	R 6B0	$PC = R[Rt]$	
Compare & Branch if Not Zero	CBNZ	CB 5A8-5AF	if( $R[Rt] \neq 0$ ) $PC = PC + CondBranchAddr$	(4,9)
Compare & Branch if Zero	CBZ	CB 5A0-5A7	if( $R[Rt] = 0$ ) $PC = PC + CondBranchAddr$	(4,9)
Exclusive OR	EOR	R 650	$R[Rd] = R[Rn] \wedge R[Rm]$	
Exclusive OR Immediate	EORI	I 690-691	$R[Rd] = R[Rn] \wedge ALUImm$	(2,9)
Load Register Unscaled offset	LDUR	D 7C2	$R[Rt] = M[R[Rn] + DTAddr]$	(5)
Load Byte Unscaled offset	LDURB	D 1C2	$R[Rt] = \{56'b0, M[R[Rn] + DTAddr](7:0)\}$	(5)
Load Half Unscaled offset	LDURH	D 3C2	$R[Rt] = \{48'b0, M[R[Rn] + DTAddr](15:0)\}$	(5)
Load Signed Word Unscaled offset	LDURSW	D 5C4	$R[Rt] = \{32\{ M[R[Rn] + DTAddr][31], M[R[Rn] + DTAddr](31:0)\}\}$	(5)
Load eXclusive Register	LDXR	D 642	$R[Rd] = M[R[Rn] + DTAddr]$	(5,7)
Logical Shift Left	LSL	R 69B	$R[Rd] = R[Rn] \ll shamt$	
Logical Shift Right	LSR	R 69A	$R[Rd] = R[Rn] \gg shamt$	
Move wide with Keep	MOVK	IM 794-797	$R[Rd] (Instruction[22:21]*16-15) = MOVImm$	(6,9)
Move wide with Zero	MOVZ	IM 694-697	$R[Rd] = \{ MOVImm \ll (Instruction[22:21]*16) \}$	(6,9)
Inclusive OR	ORR	R 550	$R[Rd] = R[Rn]   R[Rm]$	
Inclusive OR Immediate	ORRI	I 590-591	$R[Rd] = R[Rn]   ALUImm$	(2,9)
Store Register Unscaled offset	STUR	D 7C0	$M[R[Rn] + DTAddr] = R[Rt]$	(5)
Store Byte Unscaled offset	STURB	D 1C0	$M[R[Rn] + DTAddr](7:0) = R[Rt](7:0)$	(5)
Store Half Unscaled offset	STURH	D 3C0	$M[R[Rn] + DTAddr](15:0) = R[Rt](15:0)$	(5)
Store Word Unscaled offset	STURW	D 5C0	$M[R[Rn] + DTAddr](31:0) = R[Rt](31:0)$	(5)
Store eXclusive Register	STXR	D 640	$M[R[Rn] + DTAddr] = R[Rt];$ $R[Rm] = \{ atomic \} ? 0 : 1$	(5,7)
Subtract Immediate	SUBI	I 688-689	$R[Rd] = R[Rn] - ALUImm$	(2,9)
Subtract Immediate & Set flags	SUBIS	I 788-789	$R[Rd], FLAGS = R[Rn] - ALUImm$	(1,2,9)
Subtract & Set flags	SUBS	R 758	$R[Rd], FLAGS = R[Rn] - R[Rm]$	(1)

- (1) FLAGS are 4 condition codes set by the ALU operation: Negative, Zero, oVerflow, Carry  
(2)  $ALUImm = \{ 52'b0, ALU\_immediate \}$   
(3)  $BranchAddr = \{ 36\{BR\_address[25]\}, BR\_address, 2'b0 \}$   
(4)  $CondBranchAddr = \{ 43\{COND\_BR\_address[25]\}, COND\_BR\_address, 2'b0 \}$   
(5)  $DTAddr = \{ 55\{DT\_address[8]\}, DT\_address \}$   
(6)  $MOVImm = \{ 48'b0, MOV\_immediate \}$   
(7) Atomic test&set pair;  $R[Rm] = 0$  if pair atomic, 1 if not atomic  
(8) Operands considered unsigned numbers (vs. 2's complement)  
(9) Since I, B, and CB instruction formats have opcodes narrower than 11 bits, they occupy a range of 11-bit opcodes

- (10) If neither is operand a NaN and  $Value1 == Value2$ ,  $FLAGS = 4'b0110$ ;  
If neither is operand a NaN and  $Value1 < Value2$ ,  $FLAGS = 4'b1000$ ;  
If neither is operand a NaN and  $Value1 > Value2$ ,  $FLAGS = 4'b0010$ ;  
If an operand is a Nan, operands are unordered

## ARITHMETIC CORE INSTRUCTION SET

②

NAME, MNEMONIC	FOR- MAT	OPCODE/ SHAMT (Hex)	OPERATION (in Verilog)	Notes
Floating-point ADD Single	FADDS	R 0F1 / 0A	$S[Rd] = S[Rn] + S[Rm]$	
Floating-point ADD Double	FADDD	R 0F3 / 0A	$D[Rd] = D[Rn] + D[Rm]$	
Floating-point CoMPare Single	FCMPS	R 0F1 / 08	$FLAGS = (S[Rn] vs S[Rm])$	(1,10)
Floating-point CoMPare Double	FCMPD	R 0F3 / 08	$FLAGS = (D[Rn] vs D[Rm])$	(1,10)
Floating-point DIVide Single	FDIVS	R 0F1 / 06	$S[Rd] = S[Rn] / S[Rm]$	
Floating-point DIVide Double	FDIVD	R 0F3 / 06	$D[Rd] = D[Rn] / D[Rm]$	
Floating-point MULtiply Single	FMULS	R 0F1 / 02	$S[Rd] = S[Rn] * S[Rm]$	
Floating-point MULtiply Double	FMULD	R 0F3 / 02	$D[Rd] = D[Rn] * D[Rm]$	
Floating-point SUBtract Single	FSUBS	R 0F1 / 0E	$S[Rd] = S[Rn] - S[Rm]$	
Floating-point SUBtract Double	FSUBD	R 0F3 / 0E	$D[Rd] = D[Rn] - D[Rm]$	
Load Single floating-point	LDURS	R 7C2	$S[Rt] = M[R[Rn] + DTAddr]$	(5)
Load Double floating-point	LDURD	R 7C0	$D[Rt] = M[R[Rn] + DTAddr]$	(5)
MULtiply	MUL	R 4D8 / 1F	$R[Rd] = (R[Rn] * R[Rm]) (63:0)$	
Signed DIVide	SDIV	R 4D6 / 02	$R[Rd] = R[Rn] / R[Rm]$	
Signed MULtiply High	SMULH	R 4DA	$R[Rd] = (R[Rn] * R[Rm]) (127:64)$	(5)
STore Single floating-point	STURS	R 7E2	$M[R[Rn] + DTAddr] = S[Rt]$	(5)
STore Double floating-point	STURD	R 7E0	$M[R[Rn] + DTAddr] = D[Rt]$	(5)
Unsigned DIVide	UDIV	R 4D6 / 03	$R[Rd] = R[Rn] / R[Rm]$	(8)
Unsigned MULtiply High	UMULH	R 4DE	$R[Rd] = (R[Rn] * R[Rm]) (127:64)$	(8)

## CORE INSTRUCTION FORMATS

<b>R</b>	opcode	Rm	shamt	Rn	Rd
31	21 20	16 15	10 9	5 4	0
<b>I</b>	opcode	ALU immediate		Rn	Rd
31	22 21	10 9		5 4	0
<b>D</b>	opcode	DT address		Rn	Rt
31	21 20	12 11 10 9		5 4	0
<b>B</b>	opcode	BR address			
31	26 25				
<b>CB</b>	Opcode	COND_BR address			Rt
31	24 23				5 4 0
<b>IW</b>	opcode	MOV immediate			Rd
31	21 20				5 4 0

## PSEUDOINSTRUCTION SET

NAME	MNEMONIC	OPERATION
CoMPare	CMP	$FLAGS = R[Rn] - R[Rm]$
CoMPare Immediate	CMPI	$FLAGS = R[Rn] - ALUImm$
Load Address	LDA	$R[Rd] = R[Rn] + DTAddr$
MOVE	MOV	$R[Rd] = R[Rn]$

## REGISTER NAME, NUMBER, USE, CALL CONVENTION

NAME	NUMBER	USE	PRESERVED ACROSS A CALL?
X0 – X7	0-7	Arguments / Results	No
X8	8	Indirect result location register	No
X9 – X15	9-15	Temporaries	No
X16 (IP0)	16	May be used by linker as a scratch register; other times used as temporary register	No
X17 (IP1)	17	May be used by linker as a scratch register; other times used as temporary register	No
X18	18	Platform register for platform independent code; otherwise a temporary register	No
X19-X27	19-27	Saved	Yes
X28 (SP)	28	Stack Pointer	Yes
X29 (FP)	29	Frame Pointer	Yes
X30 (LR)	30	Return Address	Yes
XZR	31	The Constant Value 0	N.A.

### OPCODES IN NUMERICAL ORDER BY OPCODE

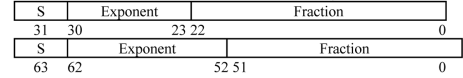
Instruction Mnemonic	Format	Width (bits)	Opcode Binary	Shamt Binary	11-bit Opcode Range (1)	
					Start (Hex)	End (Hex)
B	B	6	000101		0A0	0BF
FMULS	R	11	00011110001	000010		0F1
FDIVS	R	11	00011110001	000110		0F1
FCMPS	R	11	00011110001	001000		0F1
FADDS	R	11	00011110001	001010		0F1
FSUBS	R	11	00011110001	001110		0F1
FMULD	R	11	00011110011	000010		0F3
FDIVD	R	11	00011110011	000110		0F3
FCMPD	R	11	00011110011	001000		0F3
FADD	R	11	00011110011	001010		0F3
FSUBD	R	11	00011110011	001110		0F3
STURB	D	11	00111000000		1C0	
LDURB	D	11	00111000010		1C2	
B.cond	CB	8	01010100		2A0	2A7
STURH	D	11	01111000000		3C0	
LDURH	D	11	01111000010		3C2	
AND	R	11	10001010000		450	
ADD	R	11	10001011000		458	
ADDI	I	10	10010001000		488	489
ANDI	I	10	10010010000		490	491
BL	B	6	100101		4A0	4BF
SDIV	R	11	10011010110	000010		4D6
UDIV	R	11	10011010110	000011		4D6
MUL	R	11	10011011000	011111		4D8
SMULH	R	11	10011011010		4DA	
UMULH	R	11	10011011110		4DE	
ORR	R	11	10101010000		550	
ADDS	R	11	10101011000		558	
ADDIS	I	10	10110001000		588	589
ORRI	I	10	10110010000		590	591
CBZ	CB	8	10110100		5A0	5A7
CBNZ	CB	8	10110101		5A8	5AF
STURW	D	11	10111000000		5C0	
LDURSW	D	11	10111000100		5C4	
STURS	R	11	10111100000		5E0	
LDURS	R	11	10111100010		5E2	
STXR	D	11	11001000000		640	
LDXR	D	11	11001000010		642	
EOR	R	11	11001010000		650	
SUB	R	11	11001011000		658	
SUBI	I	10	11010001000		688	689
EORI	I	10	11010010000		690	691
MOVZ	IM	9	110100101		694	697
LSR	R	11	11010011010		69A	
LSL	R	11	11010011011		69B	
BR	R	11	11010110000		6B0	
ANDS	R	11	11101010000		750	
SUBS	R	11	11101011000		758	
SUBIS	I	10	11110001000		788	789
ANDIS	I	10	11110010000		790	791
MOVK	IM	9	111100101		794	797
STUR	D	11	11111000000		7C0	
LDUR	D	11	11111000010		7C2	
STURD	R	11	11111100000		7E0	
LDURD	R	11	11111100010		7E2	

- (1) Since I, B, and CB instruction formats have opcodes narrower than 11 bits, they occupy a range of 11-bit opcodes, e.g., the 6-bit B format occupies 32 ( $2^5$ ) 11-bit opcodes.

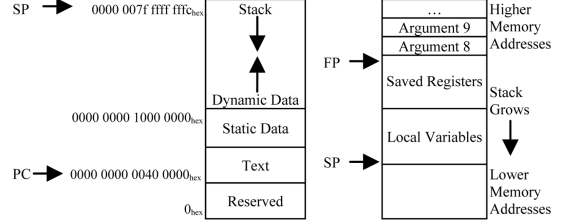
### IEEE 754 FLOATING-POINT STANDARD

$(-1)^s \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - \text{Bias})}$   
 where Single Precision Bias = 127,  
 Double Precision Bias = 1023

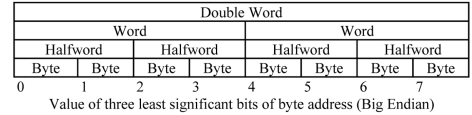
#### IEEE Single Precision and Double Precision Formats:



#### MEMORY ALLOCATION



#### DATA ALIGNMENT



#### EXCEPTION SYNDROME REGISTER (ESR)

Exception Class (EC)	Instruction Length (IL)	Instruction Specific Syndrome field (ISS)
31	26	25
		24
		0

#### EXCEPTION CLASS

EC	Class	Cause of Exception	Number	Name	Cause of Exception
0	Unknown	Unknown	34	PC	Misaligned PC exception
7	SIMD	SIMD/FP registers disabled	36	Data	Data Abort
14	FPE	Illegal Execution State	40	FPE	Floating-point exception
17	Sys	Supervisor Call Exception	52	WPT	Data Breakpoint exception
32	Instr	Instruction Abort	56	BKPT	SW Breakpoint Exception

#### SIZE PREFIXES AND SYMBOLS

SIZE	PREFIX	SYMBOL	SIZE	PREFIX	SYMBOL
$10^3$	Kilo-	K	$2^{10}$	Kibi-	Ki
$10^6$	Mega-	M	$2^{20}$	Mebi-	Mi
$10^9$	Giga-	G	$2^{30}$	Gibi-	Gi
$10^{12}$	Tera-	T	$2^{40}$	Tebi-	Ti
$10^{15}$	Peta-	P	$2^{50}$	Pebi-	Pi
$10^{18}$	Exa-	E	$2^{60}$	Exbi-	Ei
$10^{21}$	Zetta-	Z	$2^{70}$	Zebi-	Zi
$10^{24}$	Yotta-	Y	$2^{80}$	Yobi-	Yi
$10^{-3}$	milli-	m	$10^{-15}$	femto-	f
$10^{-6}$	micro-	$\mu$	$10^{-18}$	atto-	a
$10^{-9}$	nano-	n	$10^{-21}$	zepto-	z
$10^{-12}$	pico-	p	$10^{-24}$	yocto-	y