

# LEGv8

## Reference Data

### CORE INSTRUCTION SET in Alphabetical Order by Mnemonic

NAME, MNEMONIC	FOR-MAT	OPCODE (9)	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	$R[Rd], \text{FLAGS} = R[Rn] + ALUImm$	(1,2,9)
ADD & Set flags	ADDS	R 558	$R[Rd], \text{FLAGS} = R[Rn] + R[Rm]$	(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	$R[Rd], \text{FLAGS} = R[Rn] \& ALUImm$	(1,2,9)
AND & Set flags	ANDS	R 750	$R[Rd], \text{FLAGS} = R[Rn] \& R[Rm]$	(1)
Branch	B	B 0A0-0BF	$PC = PC + \text{BranchAddr}$	(3,9)
Branch conditionally	B.cond	CB 2A0-2A7	$if(\text{FLAGS} == \text{cond}) PC = PC + \text{CondBranchAddr}$	(4,9)
Branch with Link	BL	B 4A0-4BF	$R[30] = PC + 4;$ $PC = PC + \text{BranchAddr}$	(3,9)
Branch to Register	BR	R 6B0	$PC = R[Rt]$	
Compare & Branch if Not Zero	CBNZ	CB 5A8-5AF	$if(R[Rt] != 0) PC = PC + \text{CondBranchAddr}$	(4,9)
Compare & Branch if Zero	CBZ	CB 5A0-5A7	$if(R[Rt] == 0) PC = PC + \text{CondBranchAddr}$	(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	$R[Rt] = \{32; M[R[Rn] + DTAddr](31:0)\}$	(5)
Load Exclusive Register	LDXR	D 642	$R[Rd] = M[R[Rn] + DTAddr]$	(5,7)
Logical Shift Left	LSL	R 69B	$R[Rd] = R[Rn] << \text{shamt}$	
Logical Shift Right	LSR	R 69A	$R[Rd] = R[Rn] >>> \text{shamt}$	
MOVE wide with Keep	MOVK	IM 794-797	$R[Rd] (\text{Instruction}[22:21]*16) = MOVImm$	(6,9)
MOVE wide with Zero	MOVZ	IM 694-697	$R[Rd] = \{ MOVImm << (\text{Instruction}[22:21]*16) \}$	(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 xclusive Register	STXR	D 640	$M[R[Rn] + DTAddr] = R[Rt];$ $R[Rm] = (\text{atomic}) ? 0 : 1$	(5,7)
SUBtract Immediate	SUBI	I 688-689	$R[Rd] = R[Rn] - ALUImm$	(2,9)
SUBtract Immediate & Set flags	SUBIS	I 788-789	$R[Rd], \text{FLAGS} = R[Rn] - ALUImm$	(1,2,9)
SUBtract & Set flags	SUBS	R 758	$R[Rd], \text{FLAGS} = R[Rn] - R[Rm]$	(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

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	$\text{FLAGS} = (S[Rn] \text{ vs } S[Rm])$	(1,10)
Floating-point CoMPare Double	FCMPD	R 0F3 / 08	$\text{FLAGS} = (D[Rn] \text{ vs } D[Rm])$	(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	$R[Rd] = (R[Rn] * R[Rm]) (127:64)$	
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

R	opcode	Rm	shamt	Rn	Rd
31	21	16 15	10 9	5 4	0
I	opcode	ALU_immediate	Rn	Rd	
31	22 21	10 9	5 4		0
D	opcode	DT_address	op	Rn	Rt
31	21 20	12 11 10 9		5 4	0
B	opcode	BR_address			0
31	26 25				0
CB	Opcode	COND BR_address		Rt	
31	24 23			5 4	0
IW	opcode	MOV immediate		Rd	
31	21 20			5 4	0

### PSEUDOINSTRUCTION SET

NAME	MNEMONIC	OPERATION
CoMPare	CMP	$\text{FLAGS} = R[Rn] - R[Rm]$
CoMPare Immediate	CMPI	$\text{FLAGS} = R[Rn] - ALUImm$
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

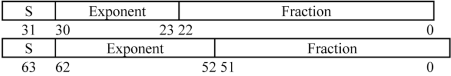
Instruction		Opcode		Shamt Binary	11-bit Opcode Range (1)	
Mnemonic	Format	Width (bits)	Binary		Start (Hex)	End (Hex)
B	B	6	000101		0A0	0BF
EMULS	R	11	00011110001	000010		0F1
EDIVS	R	11	00011110001	000110		0F1
FCMPS	R	11	00011110001	001000		0F1
FADDS	R	11	00011110001	001010		0F1
FSUBS	R	11	00011110001	001110		0F1
FMULD	R	11	00011110011	000010		0F3
FDIVD	R	11	00011110011	000110		0F3
FCMPD	R	11	00011110011	001000		0F3
FADDD	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	1001000100		488	489
ANDI	I	10	1001001000		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	1011000100		588	589
ORRI	I	10	1011001000		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	1101000100		688	689
EORI	I	10	1101001000		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	1111000100		788	789
ANDIS	I	10	1111001000		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<sup>6</sup>) 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:

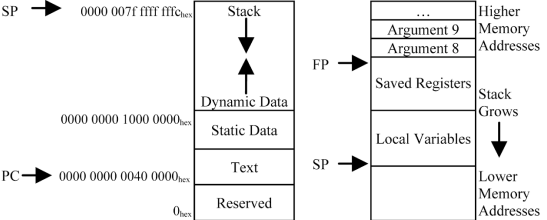


IEEE 754 Symbols

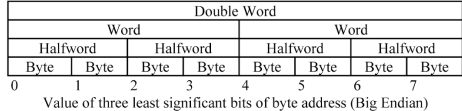
Exponent	Fraction	Object
0	0	± 0
0	≠ 0	± Denorm
1 to MAX - 1	anything	± F1. Pt. Num.
MAX	0	± ∞
MAX	≠ 0	NaN

S.P. MAX = 255, D.P. MAX = 2047

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

STURB	D	11	00111000000	1C0
LDURB	D	11	00111000010	1C2
STURH	D	11	01111000000	3C0
LDURH	D	11	01111000010	3C2
STURW	D	11	10111000000	5C0
LDURSW	D	11	10111000010	5C4
STXR	D	11	11001000000	640
LDXR	D	11	11001000010	642
STUR	D	11	11111000000	7C0
LDUR	D	11	11111000010	7C2

\* MSB:  $op[31]$

D-type:  $op[27:24] \& op[22]$

ADDI	I	10	1001000100	488	489
ANDI	I	10	1001001000	490	491
ADDIS	I	10	1011000100	588	589
ORRI	I	10	1011001000	590	591
SUBI	I	10	1101000100	688	689
EORI	I	10	1101001000	690	691
SUBIS	I	10	1111000100	788	789
ANDIS	I	10	1111001000	790	791

I-type:  $op[31] \& op[28:26] \& op[23:22]$

B.cond	CB	8	0101010	2A0	2A7
CBZ	CB	8	1011010	5A0	5A7
CBNZ	CB	8	1011010	5A8	5AF

CB-type:  $op[28:25]$

MOVZ	IM	9	110100101	694	697
MOVK	IM	9	111100101	794	797

IM-type:  $op[31:30] \& op[28:23]$

B	B	6	000101	0A0	0BF
---	---	---	--------	-----	-----

B-type:  $op[31:26]$

