LEGv8 Reference Data Card ("Green Card") 1. Pull along perforation to separate card 2. Fold bottom side (columns 3 and 4) together

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	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]!=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] ^ R[Rm]	
Exclusive OR Immediate	EORI	I	690-691	R[Rd] = R[Rn] ^ 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] << shamt	
Logical Shift Right	LSR	R	69A	R[Rd] = R[Rn] >>> shamt	
MOVe wide with Keep	MOVK	IM	794-797	R[Rd] (Instruction[22:21]*16: Instruction[22:21]*16-15) = MOVImm	(6,9)
MOVe wide with Zero	MOVZ	IM	694-697	R[Rd] = { MOVImm << (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](l5:0)	(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	SUB	R	658	R[Rd] = R[Rn] – R[Rm]	
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

②

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	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)	
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	20 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			
	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	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.