GREEN CARD FOR LEGv8

Arithmetic Operations	Assembly code				Semantics	Comments
add	ADD	Xd,	Xn,	Xm	X5 = X2 + X7	register-to-register
add & set flags	ADDS	Xd,	Xn,	Xm	X5 = X2 + X7	flags NZVC
add immediate	ADDI	Xd,	Xn,	#uimm12	X5 = X2 + #19	0 ≤ 12 bit unsigned ≤ 4095
add immediate & set flags	ADDIS	Xd,	Xn,	#uimm12	X5 = X2 + #19	flags NZVC
subtract	SUB	Xd,	Xn,	Xm	X5 = X2 – X7	register-to-register
subtract & set flags	SUBS	Xd,	Xn,	Xm	X5 = X2 – X7	flags NZVC
subtract immediate	SUBI	Xd,	Xn,	#uimm12	X5 = X2 – #20	0 ≤ 12 bit unsigned ≤ 4095
subtract immediate & set flags	SUBIS	Xd,	Xn,	Xm	X5 = X2 – #20	flags NZVC

Data Transfer Operations	Assembly code			Semantics	Comments
load register	LDUR	Xt,	[Xn, #simm9]	X2 = M[X6, #18]	double word load into Xt from Xn + #simm9
load signed word	LDURSW	Xt,	[Xn, #simm9]	X2 = M[X6, #18]	word load to lower 32b Xt from Xn + #simm9; sign extend upper 32b
load half	LDURH	Xt,	[Xn, #simm9]	X2 = M[X6, #18]	½ word load to lower 16b Xt from Xn + #simm9; zero extend upper 48b
load byte	LDURB	Xt,	[Xn, #simm9]	X2 = M[X6, #18]	byte load to least 8b Xt from Xn + #simm9 zero extend upper 56b
store register	STUR	Xt,	[Xn, #simm9]	M[X5, #12] = X4	double word store from Xt to Xn + #simm9
store word	STURW	Xt,	[Xn, #simm9]	M[X5, #12] = X4	word store from lower 32b of Xt to Xn + #simm9
store half word	STURH	Xt,	[Xn, #simm9]	M[X5, #12] = X4	½ word load from lower 16b of Xt to Xn + #simm9
store byte	STURB	Xt,	[Xn, #simm9]	M[X5, #12] = X4	byte load from least 8b of Xt to Xn + #simm9
offset		***#simm9 = -256 to +255***			***-256 ≤ 9 bits signed immediate ≤ +255***

move wide with zero	MOVZ	Xd,	#uimm16,	LSL N	X9 = 0..0N0..0	zeros out Xd then place a 16b (#uimm) into the first (N = 0)/second (N = 16)/third (N = 32)/fourth (N = 48) 16b slot of Xd
move wide with keep	MOVK	Xd,	#uimm16,	LSL N	X9 = x..xNx..x	place a 16b (#uimm) into the first (N = 0)/second (N = 16)/ third (N = 32)/fourth (N = 48) 16b slot of Xd, without changing the other values (x's)

register aliases	***X28 = SP; X29 = FP; X30 = LR; X31 = XZR***

Logical Operations	Assembly code				Semantics	Using C operations of & | ^ << >>
and	AND	Xd,	Xn,	Xm	X5 = X2 & X7	bit-wise AND
and immediate	ANDI	Xd,	Xn,	#uimm12	X5 = X2 & #19	bit-wise AND with 0 ≤ 12 bit unsigned ≤ 4095
inclusive or	ORR	Xd,	Xn,	Xm	X5 = X2 | X7	bit-wise OR
inclusive or immediate	ORRI	Xd,	Xn,	#uimm12	X5 = X2 | #11	bit-wise OR with 0 ≤ 12 bit unsigned ≤ 4095
exclusive or	EOR	Xd,	Xn,	Xm	X5 = X2 ^ X7	bit-wise EOR
exclusive or immediate	EOR	Xd,	Xn,	#uimm12	X5 = X2 ^ #57	bit-wise EOR with 0 ≤ 12 bit unsigned ≤ 4095
logical shift left	LSL	Xd,	Xn,	#uimm6	X1 = X2 << #10	shift left by a constant ≤ 63
logical shift right	LSR	Xd,	Xn,	#uimm6	X5 = X3 >> #20	shift right by a constant ≤ 63

Unconditional branches	Assembly code		Semantics	Also known as Jumps
branch	B	#simm26	goto PC + #1200	PC relative branch PC + 26b offset; -2^25 ≤ #simm26 ≤ 2^25-1; 4b instruction
branch to register	BR	Xt	target in Xt	Xt contains a full 64b address
branch with link	BL	#simm26	X30 = PC + 4; PC + #11000	PC relative branch to PC + 26b offset; 16 million instructions; X30 = LR contains return from subroutine address

Conditional branches	Assembly code			Semantics	Comments
conditional branch = 0	CBZ	Xt	#simm19	If (X2 == 0) goto PC + #99	if Xt = 0 branch to PC + 19b offset: -2^18 4b instructions ≤ #simm26 ≤ 2^18-1 4b instructions
conditional branch != 0	CBNZ	Xt	#simm19	If (X2 != 0) goto PC + #89	if Xt = 0 branch to PC + 19b offset: -2^18 4b instructions ≤ #simm26 ≤ 2^18-1 4b instructions
branch conditionally	B.cond	#simm19			if cond = true branch to PC +1 19b offset: -2^18 4b instructions ≤ #simm19 ≤ 2^18-1 4b instructions

Conditional cases (cond)	Signed Numbers			Unsigned Numbers			Comments
=	B.EQ	Z=1		B.EQ	Z=1		equal
≠	B.NE	Z=0		B.NE	Z=0		not equal
<	B.LT	N!=V		B.LO	C=0		less than: or lower
≤	B.LE	~(Z=0 & N=V)		B.LS	~(Z==0 & N=V)		less than or equal: or lower or same
>	B.GT	(Z=0 & N=V)		B.HI	(Z=0 & C=1)		greater than: or higher
≥	B.GE	N=V		B.HS	C=1		great than or equal: or higher or same
	B.MI	N=1	B.PL	branch on minus: branch on plus			
	B.VS	N=1	B.PL	branch on overflow set; branch on overflow clear			

Notes on FLAGS	NVZC	Set explicitly by arithmetic operations with "S" in the mnemonic
negative	N	msb of result = 1 — indicates a negative result if operands are two's complement
oVerflow	V	(carry out of msb) ⊗ (carry out of msb-1) = 1 — indicates the result is an overflow if operands are two's complement
zero	Z	result = 0 — indicates the result is all zeros
carry	C	carry out of msb = 1 — indicates a carry out of the msb of the result

Pseudoinstructions	Assembly code			Semantics	Comments
move reg-to-reg	MOV	Xd,	Xn	Xd = Xn	text replacement for ORR Xd, XZR, Xn
move immediate-to-reg	MOV	Xd,	#uimm12	Xd = #12	text replacement for ORRI Xd, XZR, #uimm12
compare	CMP	Xn,	Xm	set flags NVZC	text replacement for SUBS XZR, Xm, Xn

Instructions & Pseudoinstructions for Homeworks and Exams —they are not part of LEG8					
compare immediate	CMPI	Xn,	#simm11	set flags NVZC	text replacement for SUBIS XZR, Xm, #uimm12 but we interpret it as #simm11
multiply	MUL	Xd,	Xn,	Xm	Xd = Xn × Xm