LDX — Load Index Register X — LDX

Operation: IXH ⇐ (M), IXL ⇐ (M + 1)

Description: Loads the most significant byte of index register X from the byte of memory at the address specified by the program, and loads the least significant byte of index register X from the next byte of memory at one plus the address specified by the program.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇕	⇕	0	—

N R15
Set if MSB of result is set; cleared otherwise.

Z $\overline{R15} \cdot \overline{R14} \cdot \overline{R13} \cdot \overline{R12} \cdot \overline{R11} \cdot \overline{R10} \cdot \overline{R9} \cdot \overline{R8} \cdot \overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$
Set if result is $0000; cleared otherwise.

V 0
Cleared

Source Form: LDX (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	LDX (IMM)			LDX (DIR)			LDX (EXT)			LDX (IND,X)			LDX (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	CE	1	OP	DE	1	OP	FE	1	OP	EE	1	OP	CD	1
2	OP + 1	jj	1	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	EE	1
3	OP + 2	kk	1	00dd	(00dd)	1	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4				00dd + 1	(00dd + 1)	1	hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5							hhll + 1	(hhll + 1)	1	X + ff + 1	(X + ff + 1)	1	Y + ff	(Y + ff)	1
6													Y + ff + 1	(Y + ff + 1)	1

STX — Store Index Register X — STX

Operation: M ⇐ (IXH), M + 1 ⇐ (IXL)

Description: Stores the most significant byte of index register X in memory at the address specified by the program, and stores the least significant byte of index register X at the next location in memory, at one plus the address specified by the program.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇕	⇕	0	—

N IX15
Set if MSB of result is set; cleared otherwise.

Z $\overline{IX15} \cdot \overline{IX14} \cdot \overline{IX13} \cdot \overline{IX12} \cdot \overline{IX11} \cdot \overline{IX10} \cdot \overline{IX9} \cdot \overline{IX8} \cdot \overline{IX7} \cdot \overline{IX6} \cdot \overline{IX5} \cdot \overline{IX4} \cdot \overline{IX3} \cdot \overline{IX2} \cdot \overline{IX1} \cdot \overline{IX0}$
Set if result is $0000; cleared otherwise.

V 0
Cleared

Source Form: STX (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	STX (DIR)			STX (EXT)			STX (IND,X)			STX (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	DF	1	OP	FF	1	OP	EF	1	OP	CD	1
2	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	EF	1
3	00dd	(IXH)	0	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4	00dd + 1	(IXL)	0	hhll	(IXH)	0	X + ff	(IXH)	0	FFFF	—	1
5				hhll + 1	(IXL)	0	X + ff + 1	(IXL)	0	Y + ff	(IXH)	0
6										Y + ff + 1	(IXL)	0

PSHX — Push Index Register X onto Stack — PSHX

Operation: ⇓ (IXL), SP ⇐ (SP) – $0001
⇓ (IXH), SP ⇐ (SP) – $0001

Description: The contents of index register X are pushed onto the stack (low-order byte first) at the address contained in the stack pointer. The stack pointer is then decremented by two.

Push instructions are commonly used to save the contents of one or more CPU registers at the start of a subroutine. Just before returning from the subroutine, corresponding pull instructions are used to restore the saved CPU registers so the subroutine will appear not to have affected these registers.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form: PSHX

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	PSHX (INH)		
	Addr	Data	R/W
1	OP	3C	1
2	OP + 1	—	1
3	SP	(IXL)	0
4	SP – 1	(IXH)	0

PULX — Pull Index Register X from Stack — PULX

Operation: SP ⇐ (SP) + $0001; ⇑ (IXH)
SP ⇐ (SP) + $0001; ⇑ (IXL)

Description: Index register X is pulled from the stack (high-order byte first) beginning at the address contained in the stack pointer plus one. The stack pointer is incremented by two in total.

Push instructions are commonly used to save the contents of one or more CPU registers at the start of a subroutine. Just before returning from the subroutine, corresponding pull instructions are used to restore the saved CPU registers so the subroutine will appear not to have affected these registers.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form: PULX

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	PULX (INH)		
	Addr	Data	R/W
1	OP	38	1
2	OP + 1	—	1
3	SP	—	1
4	SP + 1	get IXH	1
5	SP + 2	get IXL	1