Game Boy CPU (SM83) instruction set (JSON)

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD
0x	NOP 1 4 - - - -	LD BC, n16 3 12 - - - -	LD [BC], A 1 8 - - - -	INC BC 1 8 - - - -	INC B 1 4 Z 0 H -	DEC B 1 4 Z 1 H -	LD B, n8 2 8 - - - -	RLCA 1 4 0 0 0 C	LD [a16], SP 3 20 - - - -	ADD HL, BC 1 8 - 0 H C	LD A, [BC] 1 8 - - - -	DEC BC 1 8 - - - -	INC C 1 4 Z 0 H -	DEC C 1 4 Z 1 H -
1x	STOP n8 2 4 - - - -	LD DE, n16 3 12 - - - -	LD [DE], A 1 8 - - - -	INC DE 1 8 - - - -	INC D 1 4 Z 0 H -	DEC D 1 4 Z 1 H -	LD D, n8 2 8 - - - -	RLA 1 4 0 0 0 C	JR e8 2 12 - - - -	ADD HL, DE 1 8 - 0 H C	LD A, [DE] 1 8 - - - -	DEC DE 1 8 - - - -	INC E 1 4 Z 0 H -	DEC E 1 4 Z 1 H -
2x	JR NZ, e8 2 12/8 - - - -	LD HL, n16 3 12 - - - -	LD [HL+], A 1 8 - - - -	INC HL 1 8 - - - -	INC H 1 4 Z 0 H -	DEC H 1 4 Z 1 H -	LD H, n8 2 8 - - - -	DAA 1 4 Z - 0 C	JR Z, e8 2 12/8 - - - -	ADD HL, HL 1 8 - 0 H C	LD A, [HL+] 1 8 - - - -	DEC HL 1 8 - - - -	INC L 1 4 Z 0 H -	DEC L 1 4 Z 1 H -
3x	JR NC, e8 2 12/8 - - - -	LD SP, n16 3 12 - - - -	LD [HL-], A 1 8 - - - -	INC SP 1 8 - - - -	INC [HL] 1 12 Z 0 H -	DEC [HL] 1 12 Z 1 H -	LD [HL], n8 2 12 - - - -	SCF 1 4 - 0 0 1	JR C, e8 2 12/8 - - - -	ADD HL, SP 1 8 - 0 H C	LD A, [HL-] 1 8 - - - -	DEC SP 1 8 - - - -	INC A 1 4 Z 0 H -	DEC A 1 4 Z 1 H -
4x	LD B, B 1 4 - - - -	LD B, C 1 4 - - - -	LD B, D 1 4 - - - -	LD B, E 1 4 - - - -	LD B, H 1 4 - - - -	LD B, L 1 4 - - - -	LD B, [HL] 1 8 - - - -	LD B, A 1 4 - - - -	LD C, B 1 4 - - - -	LD C, C 1 4 - - - -	LD C, D 1 4 - - - -	LD C, E 1 4 - - - -	LD C, H 1 4 - - - -	LD C, L 1 4 - - - -
5x	LD D, B 1 4 - - - -	LD D, C 1 4 - - - -	LD D, D 1 4 - - - -	LD D, E 1 4 - - - -	LD D, H 1 4 - - - -	LD D, L 1 4 - - - -	LD D, [HL] 1 8 - - - -	LD D, A 1 4 - - - -	LD E, B 1 4 - - - -	LD E, C 1 4 - - - -	LD E, D 1 4 - - - -	LD E, E 1 4 - - - -	LD E, H 1 4 - - - -	LD E, L 1 4 - - - -
6x	LD H, B 1 4 - - - -	LD H, C 1 4 - - - -	LD H, D 1 4 - - - -	LD H, E 1 4 - - - -	LD H, H 1 4 - - - -	LD H, L 1 4 - - - -	LD H, [HL] 1 8 - - - -	LD H, A 1 4 - - - -	LD L, B 1 4 - - - -	LD L, C 1 4 - - - -	LD L, D 1 4 - - - -	LD L, E 1 4 - - - -	LD L, H 1 4 - - - -	LD L, L 1 4 - - - -
7x	LD [HL], B 1 8 - - - -	LD [HL], C 1 8 - - - -	LD [HL], D 1 8 - - - -	LD [HL], E 1 8 - - - -	LD [HL], H 1 8 - - - -	LD [HL], L 1 8 - - - -	HALT 1 4 - - - -	LD [HL], A 1 8 - - - -	LD A, B 1 4 - - - -	LD A, C 1 4 - - - -	LD A, D 1 4 - - - -	LD A, E 1 4 - - - -	LD A, H 1 4 - - - -	LD A, L 1 4 - - - -
8x	ADD A, B 1 4 Z 0 H C	ADD A, C 1 4 Z 0 H C	ADD A, D 1 4 Z 0 H C	ADD A, E 1 4 Z 0 H C	ADD A, H 1 4 Z 0 H C	ADD A, L 1 4 Z 0 H C	ADD A, [HL] 1 8 Z 0 H C	ADD A, A 1 4 Z 0 H C	ADC A, B 1 4 Z 0 H C	ADC A, C 1 4 Z 0 H C	ADC A, D 1 4 Z 0 H C	ADC A, E 1 4 Z 0 H C	ADC A, H 1 4 Z 0 H C	ADC A, L 1 4 Z 0 H C
9x	SUB A, B 1 4 Z 1 H C	SUB A, C 1 4 Z 1 H C	SUB A, D 1 4 Z 1 H C	SUB A, E 1 4 Z 1 H C	SUB A, H 1 4 Z 1 H C	SUB A, L 1 4 Z 1 H C	SUB A, [HL] 1 8 Z 1 H C	SUB A, A 1 4 1 1 0 0	SBC A, B 1 4 Z 1 H C	SBC A, C 1 4 Z 1 H C	SBC A, D 1 4 Z 1 H C	SBC A, E 1 4 Z 1 H C	SBC A, H 1 4 Z 1 H C	SBC A, L 1 4 Z 1 H C
Ax	AND A, B 1 4 Z 0 1 0	AND A, C 1 4 Z 0 1 0	AND A, D 1 4 Z 0 1 0	AND A, E 1 4 Z 0 1 0	AND A, H 1 4 Z 0 1 0	AND A, L 1 4 Z 0 1 0	AND A, [HL] 1 8 Z 0 1 0	AND A, A 1 4 Z 0 1 0	XOR A, B 1 4 Z 0 0 0	XOR A, C 1 4 Z 0 0 0	XOR A, D 1 4 Z 0 0 0	XOR A, E 1 4 Z 0 0 0	XOR A, H 1 4 Z 0 0 0	XOR A, L 1 4 Z 0 0 0
Bx	OR A, B 1 4 Z 0 0 0	OR A, C 1 4 Z 0 0 0	OR A, D 1 4 Z 0 0 0	OR A, E 1 4 Z 0 0 0	OR A, H 1 4 Z 0 0 0	OR A, L 1 4 Z 0 0 0	OR A, [HL] 1 8 Z 0 0 0	OR A, A 1 4 Z 0 0 0	CP A, B 1 4 Z 1 H C	CP A, C 1 4 Z 1 H C	CP A, D 1 4 Z 1 H C	CP A, E 1 4 Z 1 H C	CP A, H 1 4 Z 1 H C	CP A, L 1 4 Z 1 H C
Cx	RET NZ 1 20/8 - - - -	POP BC 1 12 - - - -	JP NZ, a16 3 16/12 - - - -	JP a16 3 16 - - - -	CALL NZ, a16 3 24/12 - - - -	PUSH BC 1 16 - - - -	ADD A, n8 2 8 Z 0 H C	RST $00 1 16 - - - -	RET Z 1 20/8 - - - -	RET 1 16 - - - -	JP Z, a16 3 16/12 - - - -	PREFIX 1 4 - - - -	CALL Z, a16 3 24/12 - - - -	CALL a16 3 24 - - - -
Dx	RET NC 1 20/8 - - - -	POP DE 1 12 - - - -	JP NC, a16 3 16/12 - - - -	—	CALL NC, a16 3 24/12 - - - -	PUSH DE 1 16 - - - -	SUB A, n8 2 8 Z 1 H C	RST $10 1 16 - - - -	RET C 1 20/8 - - - -	RETI 1 16 - - - -	JP C, a16 3 16/12 - - - -	—	CALL C, a16 3 24/12 - - - -	—
Ex	LDH [a8], A 2 12 - - - -	POP HL 1 12 - - - -	LD [C], A 1 8 - - - -	—	—	PUSH HL 1 16 - - - -	AND A, n8 2 8 Z 0 1 0	RST $20 1 16 - - - -	ADD SP, e8 2 16 0 0 H C	JP HL 1 4 - - - -	LD [a16], A 3 16 - - - -	—	—	—
Fx	LDH A, [a8] 2 12 - - - -	POP AF 1 12 Z N H C	LD A, [C] 1 8 - - - -	DI 1 4 - - - -	—	PUSH AF 1 16 - - - -	OR A, n8 2 8 Z 0 0 0	RST $30 1 16 - - - -	LD HL, SP + e8 2 12 0 0 H C	LD SP, HL 1 8 - - - -	LD A, [a16] 3 16 - - - -	EI 1 4 - - - -	—	—

Prefixed ($CB $xx)

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	RLC B 2 8 Z 0 0 C	RLC C 2 8 Z 0 0 C	RLC D 2 8 Z 0 0 C	RLC E 2 8 Z 0 0 C	RLC H 2 8 Z 0 0 C	RLC L 2 8 Z 0 0 C	RLC [HL] 2 16 Z 0 0 C	RLC A 2 8 Z 0 0 C	RRC B 2 8 Z 0 0 C	RRC C 2 8 Z 0 0 C	RRC D 2 8 Z 0 0 C	RRC E 2 8 Z 0 0 C	RRC H 2 8 Z 0 0 C	RRC L 2 8 Z 0 0 C	RRC [HL] 2 16 Z 0 0 C	RRC A 2 8 Z 0 0 C
1x	RL B 2 8 Z 0 0 C	RL C 2 8 Z 0 0 C	RL D 2 8 Z 0 0 C	RL E 2 8 Z 0 0 C	RL H 2 8 Z 0 0 C	RL L 2 8 Z 0 0 C	RL [HL] 2 16 Z 0 0 C	RL A 2 8 Z 0 0 C	RR B 2 8 Z 0 0 C	RR C 2 8 Z 0 0 C	RR D 2 8 Z 0 0 C	RR E 2 8 Z 0 0 C	RR H 2 8 Z 0 0 C	RR L 2 8 Z 0 0 C	RR [HL] 2 16 Z 0 0 C	RR A 2 8 Z 0 0 C
2x	SLA B 2 8 Z 0 0 C	SLA C 2 8 Z 0 0 C	SLA D 2 8 Z 0 0 C	SLA E 2 8 Z 0 0 C	SLA H 2 8 Z 0 0 C	SLA L 2 8 Z 0 0 C	SLA [HL] 2 16 Z 0 0 C	SLA A 2 8 Z 0 0 C	SRA B 2 8 Z 0 0 C	SRA C 2 8 Z 0 0 C	SRA D 2 8 Z 0 0 C	SRA E 2 8 Z 0 0 C	SRA H 2 8 Z 0 0 C	SRA L 2 8 Z 0 0 C	SRA [HL] 2 16 Z 0 0 C	SRA A 2 8 Z 0 0 C
3x	SWAP B 2 8 Z 0 0 0	SWAP C 2 8 Z 0 0 0	SWAP D 2 8 Z 0 0 0	SWAP E 2 8 Z 0 0 0	SWAP H 2 8 Z 0 0 0	SWAP L 2 8 Z 0 0 0	SWAP [HL] 2 16 Z 0 0 0	SWAP A 2 8 Z 0 0 0	SRL B 2 8 Z 0 0 C	SRL C 2 8 Z 0 0 C	SRL D 2 8 Z 0 0 C	SRL E 2 8 Z 0 0 C	SRL H 2 8 Z 0 0 C	SRL L 2 8 Z 0 0 C	SRL [HL] 2 16 Z 0 0 C	SRL A 2 8 Z 0 0 C
4x	BIT 0, B 2 8 Z 0 1 -	BIT 0, C 2 8 Z 0 1 -	BIT 0, D 2 8 Z 0 1 -	BIT 0, E 2 8 Z 0 1 -	BIT 0, H 2 8 Z 0 1 -	BIT 0, L 2 8 Z 0 1 -	BIT 0, [HL] 2 12 Z 0 1 -	BIT 0, A 2 8 Z 0 1 -	BIT 1, B 2 8 Z 0 1 -	BIT 1, C 2 8 Z 0 1 -	BIT 1, D 2 8 Z 0 1 -	BIT 1, E 2 8 Z 0 1 -	BIT 1, H 2 8 Z 0 1 -	BIT 1, L 2 8 Z 0 1 -	BIT 1, [HL] 2 12 Z 0 1 -	BIT 1, A 2 8 Z 0 1 -
5x	BIT 2, B 2 8 Z 0 1 -	BIT 2, C 2 8 Z 0 1 -	BIT 2, D 2 8 Z 0 1 -	BIT 2, E 2 8 Z 0 1 -	BIT 2, H 2 8 Z 0 1 -	BIT 2, L 2 8 Z 0 1 -	BIT 2, [HL] 2 12 Z 0 1 -	BIT 2, A 2 8 Z 0 1 -	BIT 3, B 2 8 Z 0 1 -	BIT 3, C 2 8 Z 0 1 -	BIT 3, D 2 8 Z 0 1 -	BIT 3, E 2 8 Z 0 1 -	BIT 3, H 2 8 Z 0 1 -	BIT 3, L 2 8 Z 0 1 -	BIT 3, [HL] 2 12 Z 0 1 -	BIT 3, A 2 8 Z 0 1 -
6x	BIT 4, B 2 8 Z 0 1 -	BIT 4, C 2 8 Z 0 1 -	BIT 4, D 2 8 Z 0 1 -	BIT 4, E 2 8 Z 0 1 -	BIT 4, H 2 8 Z 0 1 -	BIT 4, L 2 8 Z 0 1 -	BIT 4, [HL] 2 12 Z 0 1 -	BIT 4, A 2 8 Z 0 1 -	BIT 5, B 2 8 Z 0 1 -	BIT 5, C 2 8 Z 0 1 -	BIT 5, D 2 8 Z 0 1 -	BIT 5, E 2 8 Z 0 1 -	BIT 5, H 2 8 Z 0 1 -	BIT 5, L 2 8 Z 0 1 -	BIT 5, [HL] 2 12 Z 0 1 -	BIT 5, A 2 8 Z 0 1 -
7x	BIT 6, B 2 8 Z 0 1 -	BIT 6, C 2 8 Z 0 1 -	BIT 6, D 2 8 Z 0 1 -	BIT 6, E 2 8 Z 0 1 -	BIT 6, H 2 8 Z 0 1 -	BIT 6, L 2 8 Z 0 1 -	BIT 6, [HL] 2 12 Z 0 1 -	BIT 6, A 2 8 Z 0 1 -	BIT 7, B 2 8 Z 0 1 -	BIT 7, C 2 8 Z 0 1 -	BIT 7, D 2 8 Z 0 1 -	BIT 7, E 2 8 Z 0 1 -	BIT 7, H 2 8 Z 0 1 -	BIT 7, L 2 8 Z 0 1 -	BIT 7, [HL] 2 12 Z 0 1 -	BIT 7, A 2 8 Z 0 1 -
8x	RES 0, B 2 8 - - - -	RES 0, C 2 8 - - - -	RES 0, D 2 8 - - - -	RES 0, E 2 8 - - - -	RES 0, H 2 8 - - - -	RES 0, L 2 8 - - - -	RES 0, [HL] 2 16 - - - -	RES 0, A 2 8 - - - -	RES 1, B 2 8 - - - -	RES 1, C 2 8 - - - -	RES 1, D 2 8 - - - -	RES 1, E 2 8 - - - -	RES 1, H 2 8 - - - -	RES 1, L 2 8 - - - -	RES 1, [HL] 2 16 - - - -	RES 1, A 2 8 - - - -
9x	RES 2, B 2 8 - - - -	RES 2, C 2 8 - - - -	RES 2, D 2 8 - - - -	RES 2, E 2 8 - - - -	RES 2, H 2 8 - - - -	RES 2, L 2 8 - - - -	RES 2, [HL] 2 16 - - - -	RES 2, A 2 8 - - - -	RES 3, B 2 8 - - - -	RES 3, C 2 8 - - - -	RES 3, D 2 8 - - - -	RES 3, E 2 8 - - - -	RES 3, H 2 8 - - - -	RES 3, L 2 8 - - - -	RES 3, [HL] 2 16 - - - -	RES 3, A 2 8 - - - -
Ax	RES 4, B 2 8 - - - -	RES 4, C 2 8 - - - -	RES 4, D 2 8 - - - -	RES 4, E 2 8 - - - -	RES 4, H 2 8 - - - -	RES 4, L 2 8 - - - -	RES 4, [HL] 2 16 - - - -	RES 4, A 2 8 - - - -	RES 5, B 2 8 - - - -	RES 5, C 2 8 - - - -	RES 5, D 2 8 - - - -	RES 5, E 2 8 - - - -	RES 5, H 2 8 - - - -	RES 5, L 2 8 - - - -	RES 5, [HL] 2 16 - - - -	RES 5, A 2 8 - - - -
Bx	RES 6, B 2 8 - - - -	RES 6, C 2 8 - - - -	RES 6, D 2 8 - - - -	RES 6, E 2 8 - - - -	RES 6, H 2 8 - - - -	RES 6, L 2 8 - - - -	RES 6, [HL] 2 16 - - - -	RES 6, A 2 8 - - - -	RES 7, B 2 8 - - - -	RES 7, C 2 8 - - - -	RES 7, D 2 8 - - - -	RES 7, E 2 8 - - - -	RES 7, H 2 8 - - - -	RES 7, L 2 8 - - - -	RES 7, [HL] 2 16 - - - -	RES 7, A 2 8 - - - -
Cx	SET 0, B 2 8 - - - -	SET 0, C 2 8 - - - -	SET 0, D 2 8 - - - -	SET 0, E 2 8 - - - -	SET 0, H 2 8 - - - -	SET 0, L 2 8 - - - -	SET 0, [HL] 2 16 - - - -	SET 0, A 2 8 - - - -	SET 1, B 2 8 - - - -	SET 1, C 2 8 - - - -	SET 1, D 2 8 - - - -	SET 1, E 2 8 - - - -	SET 1, H 2 8 - - - -	SET 1, L 2 8 - - - -	SET 1, [HL] 2 16 - - - -	SET 1, A 2 8 - - - -
Dx	SET 2, B 2 8 - - - -	SET 2, C 2 8 - - - -	SET 2, D 2 8 - - - -	SET 2, E 2 8 - - - -	SET 2, H 2 8 - - - -	SET 2, L 2 8 - - - -	SET 2, [HL] 2 16 - - - -	SET 2, A 2 8 - - - -	SET 3, B 2 8 - - - -	SET 3, C 2 8 - - - -	SET 3, D 2 8 - - - -	SET 3, E 2 8 - - - -	SET 3, H 2 8 - - - -	SET 3, L 2 8 - - - -	SET 3, [HL] 2 16 - - - -	SET 3, A 2 8 - - - -
Ex	SET 4, B 2 8 - - - -	SET 4, C 2 8 - - - -	SET 4, D 2 8 - - - -	SET 4, E 2 8 - - - -	SET 4, H 2 8 - - - -	SET 4, L 2 8 - - - -	SET 4, [HL] 2 16 - - - -	SET 4, A 2 8 - - - -	SET 5, B 2 8 - - - -	SET 5, C 2 8 - - - -	SET 5, D 2 8 - - - -	SET 5, E 2 8 - - - -	SET 5, H 2 8 - - - -	SET 5, L 2 8 - - - -	SET 5, [HL] 2 16 - - - -	SET 5, A 2 8 - - - -

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
Fx	SET 6, B 2 8 - - - -	SET 6, C 2 8 - - - -	SET 6, D 2 8 - - - -	SET 6, E 2 8 - - - -	SET 6, H 2 8 - - - -	SET 6, L 2 8 - - - -	SET 6, [HL] 2 16 - - - -	SET 6, A 2 8 - - - -	SET 7, B 2 8 - - - -	SET 7, C 2 8 - - - -	SET 7, D 2 8 - - - -	SET 7, E 2 8 - - - -	SET 7, H 2 8 - - - -	SET 7, L 2 8 - - - -	SET 7, [HL] 2 16 - - - -	SET 7, A 2 8 - - - -

🟥 Misc / control instructions
🟧 Jumps / calls
🟪 8-bit load instructions
🟩 16-bit load instructions
🟨 8-bit arithmetic / logical instructions
🟫 16-bit arithmetic / logical instructions
🟦 8-bit shift, rotate and bit instructions

	INS reg	← Instruction mnemonic
Length in bytes →	2 8	← Duration in T-states*
	Z N H C	← Flags affected

*) Often instruction durations are given in "M-cycles" (machine cycles) instead of "T-states" (system clock ticks) because each instruction takes a multiple of four T-states to complete, thus a `NOP` takes one M-cycle or four T-states to complete.

`Z` - Zero Flag
`N` - Subtract Flag
`H` - Half Carry Flag
`C` - Carry Flag
`0` - The flag is reset
`1` - The flag is set
`-` - The flag is left untouched

If an operation has the flags defined as `Z`, `N`, `H`, or `C`, the corresponding flags are set as the operation performed dictates.

The duration of conditional calls and returns is different when action is taken or not. This is indicated by two numbers separated by "/". The higher number (on the left side of "/") is the duration of the instruction when action is taken, the lower number (on the right side of "/") is the duration of the instruction when action is not taken.

STOP: The opcode of this instruction is $10, but it has to be followed by an additional byte that is ignored by the CPU (any value works, but normally $00 is used).

`n8` means immediate 8-bit data

`n16` means immediate little-endian 16-bit data

`a8` means 8-bit unsigned data, which is added to $FF00 in certain instructions to create a 16-bit address in HRAM (High RAM)

`a16` means little-endian 16-bit address

`e8` means 8-bit signed data

`LD A, [C]` has the alternative mnemonic `LD A, [$FF00+C]`

`LD [C], A` has the alternative mnemonic `LD [$FF00+C], A`

`LDH A, [a8]` has the alternative mnemonic `LD A, [$FF00+a8]`

`LDH [a8], A` has the alternative mnemonic `LD [$FF00+a8], A`

`LD A, [HL+]` has the alternative mnemonic `LD A, [HLI]` or `LDI A, [HL]`

`LD [HL+], A` has the alternative mnemonic `LD [HLI], A` or `LDI [HL], A`

`LD A, [HL-]` has the alternative mnemonic `LD A, [HLD]` or `LDD A, [HL]`

`LD [HL-], A` has the alternative mnemonic `LD [HLD], A` or `LDD [HL], A`

`LD HL, SP+e8` has the alternative mnemonic `LDHL SP, e8`

ALU instructions (`ADD`, `ADC`, `SUB`, `SBC`, `AND`, `XOR`, `OR`, and `CP`) can be written with the left-hand side `A` omitted. Thus for example `ADD A, B` has the alternative mnemonic `ADD B`, and `CP A, $F` has the alternative mnemonic `CP $F`.

This page is based on the opcodes list of PASTRAISER. Errata