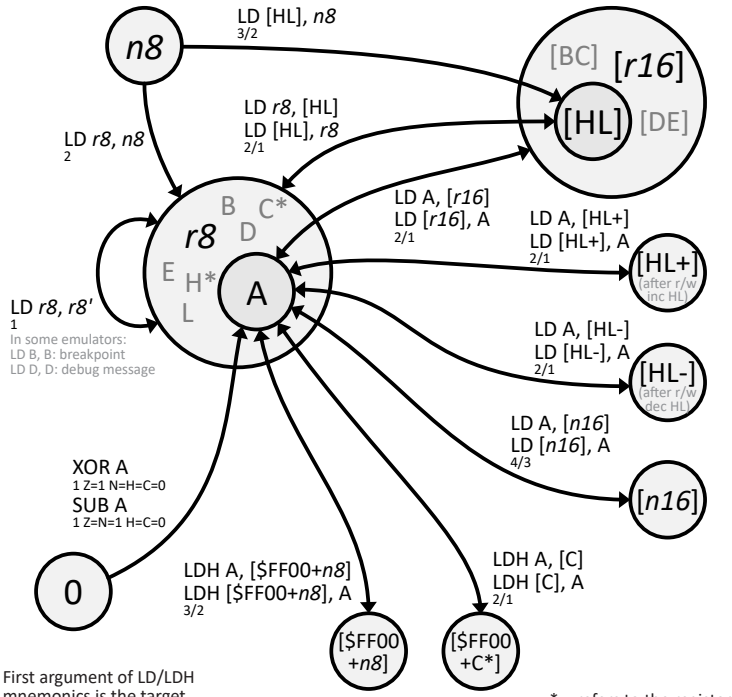
Game Boy Instructions Cheat Sheet

8-Bit Loads



First argument of LD/LDH mnemonics is the target.

* = refers to the register (rather than the flag of the same name)

8-Bit Shifts

operation variant	on A Z=N=H=0	on *r8* Z=● N=H=0	on [HL] Z=● N=H=0	new value for C and subject (b_7 most, …, b_0 least significant old bit)
rotate left through carry (add self and C, meaning RLA ≈ ADC A)	RLA 1 s.a.	RL *r8* 2 s.a.	RL [HL] 4/2 s.a.	b_7 | b_6 b_5 b_4 b_3 b_2 b_1 b_0 C
rotate left cyclic	RLCA 1 s.a.	RLC *r8* 2 s.a.	RLC [HL] 4/2 s.a.	b_7 | b_6 b_5 b_4 b_3 b_2 b_1 b_0 b_7
rotate right through carry	RRA 1 s.a.	RR *r8* 2 s.a.	RR [HL] 4/2 s.a.	b_0 | C b_7 b_6 b_5 b_4 b_3 b_2 b_1
rotate right cyclic	RRCA 1 s.a.	RRC *r8* 2 s.a.	RRC [HL] 4/2 s.a.	b_0 | b_0 b_7 b_6 b_5 b_4 b_3 b_2 b_1
rotate by 4 cyclic (swap nibbles)		SWAP *r8* 2 s.a.	SWAP [HL] 4/2 s.a.	0 | b_3 b_2 b_1 b_0 b_7 b_6 b_5 b_4
shift left (multiplication by 2)	ADD A 1 s. ADD	SLA *r8* 2 s.a.	SLA [HL] 4/2 s.a.	b_7 | b_6 b_5 b_4 b_3 b_2 b_1 b_0 0
shift right arithmetic (signed division by 2, round down)		SRA *r8* 2 s.a.	SRA [HL] 4/2 s.a.	b_0 | b_7 b_7 b_6 b_5 b_4 b_3 b_2 b_1
shift right logical (unsigned division by 2, round down)		SRL *r8* 2 s.a.	SRL [HL] 4/2 s.a.	b_0 | 0 b_7 b_6 b_5 b_4 b_3 b_2 b_1

Stack Operations by Example

before	after PUSH AF	after POP BC
address (byte) pointed to by SP	SP moves *before* each load	each load done before moving SP
(section of the 16-bit memory space; same 5-byte section shown in all three figures)	F A; bits of F: Z N H C 0 0 0 0; most to least significant; F's low nibble is always 0	F A; C B

Stack Operations

operation	*r16*	AF	PC
push the value that's in register	PUSH *r16* 4/1	PUSH AF 4/1	
pop top stack element to register	POP *r16* 3/1	POP AF 3/1 see figure above	RET 4/1

Increments and Decrements

operation	*r8*	[HL]	*r16*	SP
increment (add 1)	INC *r8* 1 Z=● N=0 H=●	INC [HL] 3/1 Z=● N=0 H=●	INC *r16* 2/1	INC SP 2/1
decrement (subtract 1)	DEC *r8* 1 Z=● N=1 H=●	DEC [HL] 3/1 Z=● N=1 H=●	DEC *r16* 2/1	DEC SP 2/1

8-Bit Arithmetics on A

operation	*n8*	*r8*	[HL]
add	ADD *n8*	ADD *r8*	ADD [HL]
add carry flag and Z=● N=0 H=● C=●	ADC *n8* 2 s.l.	ADC *r8* 1 s.l.	ADC [HL] 2/1 s.l.
subtract	SUB *n8*	SUB *r8*	SUB [HL]
subtract carry flag and Z=● N=1 H=● C=●	SBC *n8* 2 s.l.	SBC *r8* 1 s.l.	SBC [HL] 2/1 s.l.
compare (subtract but leave A) Z := argument==A; C := argument>A	CP *n8* 2 s. SUB	CP *r8* 1 s. SUB	CP [HL] 2/1 s. SUB

Bitwise/Logic Operations on A

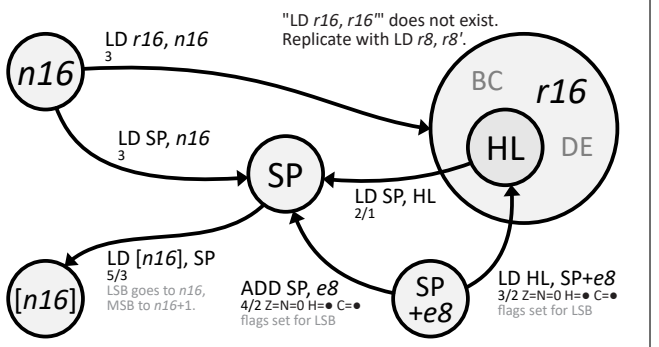
operation	with *n8*	with *r8*	with [HL]
AND Z=● N=C=0 H=1	AND *n8* 2 s.l.	AND *r8* 1 s.l.	AND [HL] 2/1 s.l.
OR Z=● N=H=C=0	OR *n8* 2 s.l.	OR *r8* 1 s.l.	OR [HL] 2/1 s.l.
XOR (exclusive OR) Z=● N=H=C=0	XOR *n8* 2 s.l.	XOR *r8* 1 s.l.	XOR [HL] 2/1 s.l.
NOT, a unary operation (255−A; XOR 255; complement)	CPL 1 N=H=1		

Bit Operations

operation	on *u3*, *r8*	on *u3*, [HL]	on C=F_4
load to !Z Z=● N=0 H=1	BIT *u3*, *r8* 2 s.l.	BIT *u3*, [HL] 3/2 s.l.	
reset (set to 0; clear)	RES *u3*, *r8* 2	RES *u3*, [HL] 4/2	
set (set to 1)	SET *u3*, *r8* 2	SET *u3*, [HL] 4/2	SCF 1 N=H=0 C=1
complement (apply NOT)			CCF 1 N=H=0 C=!C

16-Bit Loads and SP Arithmetics

28 Sep 2025



16-Bit Arithmetics

operation	*r16* to HL	SP to HL	*e8* to SP
add	ADD HL, *r16* 2/1 N=0 H=● C=● flags set for MSB	ADD HL, SP 2/1 N=0 H=● C=● flags set for MSB	ADD SP, *e8* 4/2 Z=N=0 H=● C=● flags set for LSB

Jumps, Calls and Returns

operation	always	if *cc*
jump anywhere ("LD PC, *n16*")	JP *n16* 4/3	JP *cc*, *n16* (4 true, 3 false)/3
jump relatively to a *n16* that's -130…125 bytes from here	JR *n16* 3/2	JR *cc*, *n16* (3 true, 2 false)/2
jump to HL ("LD PC, HL")	JP HL 1	
call anything ("PUSH PC" of next instruction and JP *n16*)	CALL *n16* 6/3	CALL *cc*, *n16* (6 true, 3 false)/3
call 8**u3* (faster CALL {$00, $08, $10, …, $30, $38})	RST *n16* 4/1	
return ("POP PC")	RET 4/1	RET *cc* (5 true, 2 false)/1
return and enable interrupts (shorthand for EI followed by RET)	RETI 4/1	

Miscellaneous

operation	instruction
make A a BCD again after ADD, ADC, SUB, SBC another BCD	DAA Z=● C=(● OR'd with C) H=0
disable interrupts also done automatically on interrupt	DI 1
enable interrupts after the instruction following EI	EI 1
wait for interrupt in low energy mode	HALT -/1 (might read next byte twice)
apply CGB speed switch if KEY1/SPD ($FF4D) bit 0 set, otherwise:	STOP -/2 (might only read the first)
wait for button press in super low energy mode – requires P1 on!	Beware: STOP has some weird behavior. Might damage a DMG.
do nothing (no operation; "INC PC")	NOP 1

Legend/Definitions/Notation

LEGEND ***r8***: One of the 8-bit registers (A, B, C, D, E, H, L). F contains flags and doesn't count as *r8*. ***n8***: An 8-bit integer (usually unsigned). ***e8***: An 8-bit integer that's explicitly signed. ***r16***: One of the 16-bit registers (BC, DE, HL), composed of the 8-bit registers with the same name, e.g. BC's most significant byte (MSB) is shared with register B. This also applies to AF, but it doesn't count as *r16*, neither do the dedicated SP (stack pointer) and PC (program counter) registers. ***n16***: A 16-bit integer (usually unsigned). ***cc***: A condition code (Z: flag Z set; NZ: flag Z clear; C: flag C set; NC: flag C clear). ***u3***: An unsigned 3-bit integer. **[*x*]**: Value pointed to by *x*/at address *x*.

NOTATION **Ambiguities:** C and H refer to the flags unless stated otherwise. **Information below the instruction:** M-cycles/size (size omitted if same as cycles), followed by flags (if any are affected). Values for flags are given as 0 (clear), 1 (set) or ● (as per the result). **Flag behavior denoted by ●:** Z indicates if the result is 0. H indicates if low nibble over-/underflowed. C does the same for the whole byte. (H and N are used by the DAA instruction only.) **(*x*)**: Alternative wording(s). **"*x*"**: Hypothetical instruction.