Gameboy CPU (LR35902) instruction set

	x0	x1	x2	x3	x4	x5	x6	x7	
0x	NOP 1 4 - - - -	LD BC,d16 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,d8 2 8 - - - -	RLCA 1 4 0 0 0 C	LD (-
1x	STOP 0 2 4 - - - -	LD DE,d16 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,d8 2 8 - - - -	RLA 1 4 0 0 0 C	 -
2x	JR NZ,r8 2 12/8 - - - -	LD HL,d16 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,d8 2 8 - - - -	DAA 1 4 Z - 0 C	J 2 -
3x	JR NC,r8 2 12/8 - - - -	LD SP,d16 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),d8 2 12 - - - -	SCF 1 4 - 0 0 1	J 2 -
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 - - - -	L -
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 - - - -	L -
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 - - - -	L -
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 - - - -	L -
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	AD Z
9x	SUB B 1 4 Z 1 H C	SUB C 1 4 Z 1 H C	SUB D 1 4 Z 1 H C	SUB E 1 4 Z 1 H C	SUB H 1 4 Z 1 H C	SUB L 1 4 Z 1 H C	SUB (HL) 1 8 Z 1 H C	SUB A 1 4 Z 1 H C	S Z
Ax	AND B 1 4 Z 0 1 0	AND C 1 4 Z 0 1 0	AND D 1 4 Z 0 1 0	AND E 1 4 Z 0 1 0	AND H 1 4 Z 0 1 0	AND L 1 4 Z 0 1 0	AND (HL) 1 8 Z 0 1 0	AND A 1 4 Z 0 1 0	 Z
Bx	OR B 1 4 Z 0 0 0	OR C 1 4 Z 0 0 0	OR D 1 4 Z 0 0 0	OR E 1 4 Z 0 0 0	OR H 1 4 Z 0 0 0	OR L 1 4 Z 0 0 0	OR (HL) 1 8 Z 0 0 0	OR A 1 4 Z 0 0 0	 Z
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,d8 2 8 Z 0 H C	RST 00H 1 16 - - - -	 1 -
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 d8 2 8 Z 1 H C	RST 10H 1 16 - - - -	 1 -
Ex	LDH (a8),A 2 12 - - - -	POP HL 1 12 - - - -	LD (C),A 2 8 - - - -			PUSH HL 1 16 - - - -	AND d8 2 8 Z 0 1 0	RST 20H 1 16 - - - -	AD 0
Fx	LDH A,(a8) 2 12 - - - -	POP AF 1 12 Z N H C	LD A,(C) 2 8 - - - -	DI 1 4 - - - -		PUSH AF 1 16 - - - -	OR d8 2 8 Z 0 0 0	RST 30H 1 16 - - - -	LD 0

Prefix CB

	x0	x1	x2	x3	x4	x5	x6	x7	
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	 Z
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	 Z
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	 Z
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	 Z
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 16 Z 0 1 -	BIT 0,A 2 8 Z 0 1 -	BI Z
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 16 Z 0 1 -	BIT 2,A 2 8 Z 0 1 -	BI Z
6x	BIT 4,B 2 8	BIT 4,C 2 8	BIT 4,D 2 8	BIT 4,E 2 8	BIT 4,H 2 8	BIT 4,L 2 8	BIT 4,(HL) 2 16	BIT 4,A 2 8	BI

	Z 0 1 -	Z 0 1 -	Z 0 1 -	Z 0 1 -	Z 0 1 -	Z 0 1 -	Z 0 1 -	Z 0 1 -	Z
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 16 Z 0 1 -	BIT 6,A 2 8 Z 0 1 -	BI Z
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 - - - -	RE -
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 - - - -	RE -
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 - - - -	RE -
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 - - - -	RE -
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 - - - -	SE -
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 - - - -	SE -
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 - - - -	SE -
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 - - - -	SE -

```
Misc/control instructions
Jumps/calls
8bit load/store/move instructions                                 +---------+
16bit load/store/move instructions                                | INS reg | ← Instruction mne
8bit arithmetic/logical instructions          Length in bytes →   |  2  8   | ← Duration in cyc
16bit arithmetic/logical instructions                             | Z N H C | ← Flags affected
8bit rotations/shifts and bit instructions                        +---------+
```

Instruction **STOP** has according to manuals opcode **10 00** and thus is 2 bytes long. Anyhow it seems there is no reason for it
Flags affected are always shown in **Z H N C** order. If flag is marked by "**0**" it means it is reset after the instruction. If i
"**C**" corresponding flag is affected as expected by its function.

d8 means immediate 8 bit data
d16 means immediate 16 bit data
a8 means 8 bit unsigned data, which are added to $FF00 in certain instructions (replacement for missing **IN** and **OUT** instruc
a16 means 16 bit address
r8 means 8 bit signed data, which are added to program counter

LD A,(C) has alternative mnemonic **LD A,($FF00+C)**
LD C,(A) has alternative mnemonic **LD ($FF00+C),A**
LDH A,(a8) has alternative mnemonic **LD A,($FF00+a8)**
LDH (a8),A has alternative mnemonic **LD ($FF00+a8),A**
LD A,(HL+) has alternative mnemonic **LD A,(HLI)** or **LDI A,(HL)**
LD (HL+),A has alternative mnemonic **LD (HLI),A** or **LDI (HL),A**
LD A,(HL-) has alternative mnemonic **LD A,(HLD)** or **LDD A,(HL)**
LD (HL-),A has alternative mnemonic **LD (HLD),A** or **LDD (HL),A**
LD HL,SP+r8 has alternative mnemonic **LDHL SP,r8**

Registers

15 . . . 8	7 . . . 0
A (accumulator)	F (flags)
B	C
D	E
H	L

15 . . . 0
SP (stack pointer)
PC (program counter)

Flag register (F) bits:

7	6	5	4	3	2	1	0
Z	N	H	C	0	0	0	0

Z - Zero Flag
N - Subtract Flag
H - Half Carry Flag
C - Carry Flag
0 - Not used, always zero