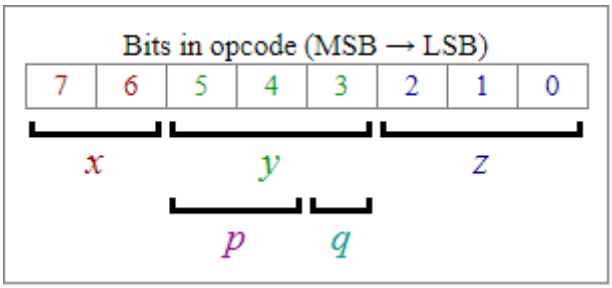
Info

Opcodes can be divided in the following sections



Using these sections you can easily decode the opcodes

15 … 8	7 … 0
A	F (flags)
B	C
D	E
H	L
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

Other relevant info:

Instruction **STOP** has according to manuals opcode **10 00** and thus is 2 bytes long. Anyhow it seems there is no reason for it so some assemblers code it simply as one byte instruction **10**. 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 it is marked by **"1"** it is set. If it is marked by **"-"** it is not changed. If it is marked by **"Z"**, **"N"**, **"H"** or **"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** instructions)
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**

Table "r"								
8-bit registers								
Index	0	1	2	3	4	5	6	7
Value	B	C	D	E	H	L	(HL)	A

Table "rp"				
Register pairs featuring SP				
Index	0	1	2	3
Value	BC	DE	HL	SP

Table "rp2"				
Register pairs featuring AF				
Index	0	1	2	3
Value	BC	DE	HL	AF

Table "cc"								
Conditions								
Index	0	1	2	3	4	5	6	7
Value	NZ	Z	NC	C	PO	PE	P	M

Table "alu"								
Arithmetic/logic operations								
Index	0	1	2	3	4	5	6	7
Value	ADD A,	ADC A,	SUB	SBC A,	AND	XOR	OR	CP

Table "rot"								
Rotation/shift operations								
Index	0	1	2	3	4	5	6	7
Value	RLC	RRC	RL	RR	SLA	SRA	SLL	SRL

Table "im"								
Interrupt modes								
Index	0	1	2	3	4	5	6	7
Value	0	0/1	1	2	0	0/1	1	2

Table "bli"				
Block instructions				
Index[a,b]	**b=0**	**b=1**	**b=2**	**b=3**
a=4	LDI	CPI	INI	OUTI
a=5	LDD	CPD	IND	OUTD
a=6	LDIR	CPIR	INIR	OTIR
a=7	LDDR	CPDR	INDR	OTDR

Unprefixed Opcodes

For X= 0				
Z = 0		Y=0 **NOP** Y=1 **LD** *(a16),SP* Y=2 **STOP**	Y=3 **JR** *r8* Y=4..7 **JP** *cc[Y-4], r8*	Nop, stop, write stackpointer And relatitive jumps
Z = 1	Q=0 Q=1	**LD** *rp[p], d16* **ADD *HL,*** *rp[p]*		16-bit load immediate/add
Z = 2	Q=0 Q=1 	P=0 **LD** *(BC), A* P=1 **LD** *(DE), A* P=0 **LD** *A, (BC)* P=1 **LD** *A, (DE)*	P=2 **LDI** *HL, A* P=3 **LDD** *HL, A* P=2 **LDI** *A, HL* P=3 **LDD** *A, HL*	Indirect loading
Z = 3	Q=0 Q=1	**INC** *rp[p]* **DEC** *rp[p]*		16-bit INC/DEC
Z = 4		**INC** *r[Y]*		8-bit INC
Z = 5		**DEC** *r[Y]*		8-bit DEC
Z = 6		**LD** *r[Y], n*		8-bit load immediate into *r[y]*
Z = 7		Y=0 **RLCA** Y=1 **RRCA** Y=2 **RLA** Y=3 **RRA**	Y=4 **DAA** Y=5 **CPL** Y=6 **SCF** Y=7 **CCF**	Assorted operations on accumulator/flags
For x= 1				
Z = 6		Y=6 **HALT**		Power down cpu until interrupt occurs
others		**LD** *r[Y], r[Z]*		8-bit load *r[z]* into *r[y]*
For x= 2				
		Alu[y] *r[Z]*		Operate on accumulator and register/memory location
For x= 3				
Z = 0	Q=0 Q=1 	P=0 **RET** *NZ* P=1 **RET** *NC* P=0 **RET** *Z* P=1 **RET** *C*	P=2 **LDH** *(a8), A* P=3 **LDH** *A,(a8)* P=2 **ADD** *SP, r8* P=3 **LD** *HL, SP+r8*	Conditional returns & and various others
Z = 1	Q=0 Q=1	**POP** *rp2[P]* P=0 **RET** P=1 **RETI**	 P=2 **JP** *(HL)* P=3 **LD** *SP, HL*	POP & various ops
Z = 2	Q=0 Q=1 	P=0 **JP** *NZ, a16* P=1 **JP** *NC, a16* P=0 **JP** *Z, a16* P=1 **JP** *C, a16*	P=2 **LD** *(C), A* P=3 **LD** *A, (C)* P=2 **LD** *a16, A* P=3 **LD** *A, a16*	Jumps and 16-bit loads
Z = 3		Y=0 **JP** *a16* Y=1 (CB prefix)	Y=6 **DI** Y=7 **EI**	Assorted operations
Z = 4		**CALL** *cc[Y]*		Conditional call
Z = 5	Q=0 Q=1	**PUSH** *rp2[p]* **CALL** *a16*		(the call is only for Q=1, P=0. P>0 are all unused)
Z = 6		***Alu[Y]****, d8*		Operate on accumulator and immediate operand
Z = 7		**RST** *Y*8*		restart

CB-Prefixed Opcodes

X=0	**ROT[Y]** *r[Z]*	Roll/shift register or memory location
X=1	**BIT** *Y, r[Z]*	Test bit
X=2	**RES** *Y, r[Z]*	Reset bit
X=3	**SET** *Y, r[Z]*	Set bit

Sources

Decoding method from: http://www.z80.info/decoding.htm#intro
Table of Gameboy opcodes: http://www.pastraiser.com/cpu/gameboy/gameboy_opcodes.html