GAME BOY CPU INSTRUCTIONS

Mnemonic	Symbolic Operation	Comments	CPU Clocks ×4	Z	N	H	C

8-Bit Transfer

LD r, s	r ← s	s = r, n, [HL]	r: 1, n: 2, [HL]: 2	-	-	-	-
LD d, r	d ← r	d = r, [HL]	r: 1, [HL]: 2				
LD d, n	d ← n		r: 2, [HL]: 3				
LD A, [ss]	A ← [ss]	ss = BC, DE, HL, nn	BC,DE,HL: 2, nn: 4				
LD [dd], A	[dd] ← A	dd = BC, DE, HL, nn					
LD A, [HL−]	A ← [HL], HL ← HL−1		2				
LD [HL−], A	[HL] ← A, HL ← HL−1		2				
LD A, [HL+]	A ← [HL], HL ← HL+1		2				
LD [HL+], A	[HL] ← A, HL ← HL+1		2				
LDH [C], A,	[$FF00+C] ← A		2				
LDH A, [C]	A ← [$FF00+C]		2				
LDH [n], A	[$FF00+n] ← A		3				
LDH A, [n]	A ← [$FF00+n]		3				

16-Bit Transfer

LD dd, nn	dd ← nn	dd = BC, DE, HL, SP	3	-	-	-	-
LD [nn], SP	[nn] ← SP_L, [nn+1] ← SP_H		5				
LD SP, HL	SP ← HL		2				
LD HL, SP+e	HL ← SP+e		3	0	0	*	*
PUSH ss	[SP−1] ← ss_H, [SP−2] ← ss_L, SP ← SP−2	ss = BC, DE, HL, AF	4	-	-	-	-
POP dd	dd_L ← [SP], dd_H ← [SP+1], SP ← SP+2	dd = BC, DE, HL, AF	3				

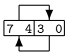
Arithmetic & Logical

ADD s	A ← A + s	CY is the carry flag. s = r, n, [HL]	r: 1, n: 2, [HL]: 2	*	0	*	*
ADC s	A ← A + s + CY						
SUB s	A ← A − s			*	1	*	*
SBC s	A ← A − s − CY						
CP s	A − s			*	1	*	*
AND s	A ← A ∧ s	} Bitwise operations		*	0	1	0
OR s	A ← A ∨ s			*	0	0	0
XOR s	A ← A ⊕ s						
INC s	s ← s + 1	s = r, [HL]	r: 1, [HL]: 3	*	0	*	-
DEC s	s ← s − 1			*	1	*	-

16-Bit Arithmetic

ADD HL, ss	HL ← HL + ss	ss = BC, DE, HL, SP	2	-	0	*	*
ADD SP, e	SP ← SP + e		4	0	0	*	*
INC ss	ss ← ss + 1		2	-	-	-	-
DEC ss	ss ← ss − 1		2				

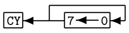
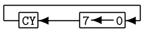
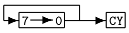
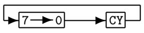
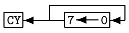
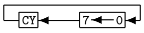
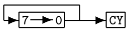
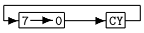
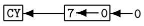
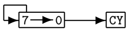
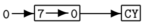
Miscellaneous

SWAP s		Swap nibbles. s = r, [HL]	r: 2, [HL]: 4	*	0	0	0
DAA	Adjusts A to packed BCD		1	*	-	0	*
CPL	A ← $\overline{A}$		1	-	1	1	-
CCF	CY ← $\overline{CY}$	CY is the carry flag.	1	-	0	0	*
SCF	CY ← 1		1	-	0	0	1
NOP	No operation		1	-	-	-	-
HALT	Enter HALT mode		N/A				
STOP	Enter STOP mode		N/A				
DI	Disable Interrupts		1				
EI	Enable Interrupts		1				

GAME BOY CPU INSTRUCTIONS

Mnemonic	Symbolic Operation	Comments	CPU Clocks ×4	Z	N	H	C

Rotates & Shifts

RLCA			1	0	0	0	*
RLA							
RRCA							
RRA							
RLC s		s = A, r, [HL]	r: 2, [HL]: 4	*	0	0	*
RL s							
RRC s							
RR s							
SLA s		s = r, [HL]	r: 2, [HL]: 4				
SRA s							
SRL s							

Bit Operations

BIT b, s	Z ← $\overline{s_b}$	Z is zero flag. s = r, [HL]	r: 2, [HL]: 3	*	0	1	-
SET b, s	s_b ← 1		r: 2, [HL]: 4	-	-	-	-
RES b, s	s_b ← 0						

Jumps

JP nn	PC ← nn		4	-	-	-	-
JP cc, nn	If cc is true, PC ← nn		If cc is true, 4 else 3				
JP HL	PC ← HL		1				
JR e	PC ← PC + e	Base address is that of *next* instruction	3				
JR cc, e	If cc is true, PC ← PC + e		If cc is true, 3 else 2				

Calls & Returns

CALL nn	[SP−1] ← PC_H, [SP−2] ← PC_L, SP ← SP−2, PC ← nn		6	-	-	-	-
CALL cc, nn	If cc is true, same as CALL nn		If cc is true, 6 else 3				
RST f	Fast shortcut for CALL f		4				
RET	PC_L ← [SP], PC_H ← [SP+1], SP ← SP+2		4				
RET cc	If cc is true, same as RET		If cc is true, 5 else 2				
RETI	RET then enable interrupts		4				

Terminology

-	Flag is not affected by this operation.
*	Flag is affected according to result of operation.
b	A bit number in any 8-bit register or memory location.
cc	Flag condition code: C, NC, Z, NZ
d	Any 8-bit destination register or memory location.
dd	Any 16-bit destination register or memory location.
e	8-bit signed offset (-128 to 127).
f	Special call location in page zero (00h, 08h... 38h).
n	Any 8-bit integer constant.
nn	Any 16-bit integer constant.
r	Any 8-bit register (A, B, C, D, E, H or L).
s	Any 8-bit source register or memory location.
s_b	A bit in a specific 8-bit register or memory location.
ss	Any 16-bit source register or memory location.

Flags

Z	Zero
N	Subtract
H	Half Carry
C	Carry