CHAPTER 4: CPU INSTRUCTION SET

1. GENERAL PURPOSE REGISTERS

7 0	7 0
A	F
B	C
D	E
H	L

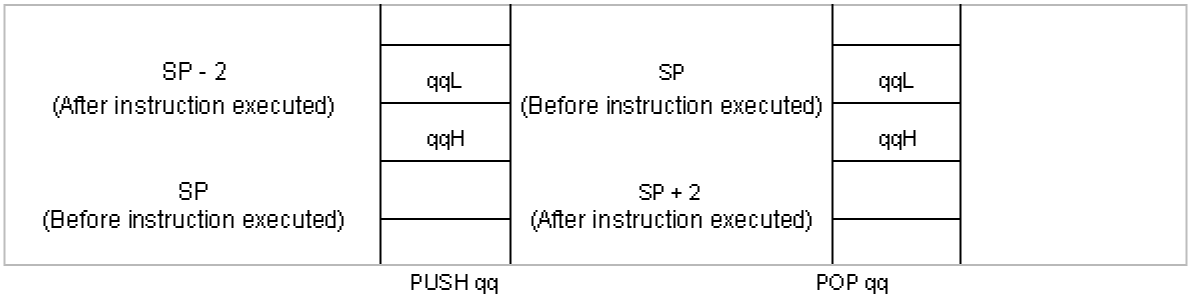
15 0
PC
SP

- Accumulator: A
  An 8-bit register for storing data and the results of arithmetic and logical operations.

- Auxiliary registers: B, C, D, E, F, H, and L
  These serve as auxiliary registers to the accumulator. As register pairs (BC, DE, HL), they are 8-bit registers that function as data pointers.

- Program counter: PC
  A 16-bit register that holds the address data of the program to be executed next.
  Usually incremented automatically according to the byte count of the fetched instructions. When an instruction with branching is executed, however, immediate data and register contents are loaded.

- Stack pointer: SP
  A 16-bit register that holds the starting address of the stack area of memory.
  The contents of the stack pointer are decremented when a subroutine CALL instruction or PUSH instruction is executed or when an interrupt occurs and incremented when a return instruction or pop instruction is executed.


SP - 2
(After instruction executed)
SP
(Before instruction executed)
qqL
qqH
PUSH qq
SP
(Before instruction executed)
SP + 2
(After instruction executed)
qqL
qqH
POP qq

- Flag Register: F
  Consists of 4 flags that are set and reset according to the results of instruction execution.
  Flags CY and Z are tested by various conditional branch instructions.

7	6	5	4	3	2	1	0
Z	N	H	CY	╲	╲	╲	╲

Z: Set to 1 when the result of an operation is 0; otherwise reset.
N: Set to 1 following execution of the substruction instruction, regardless of the result.
H: Set to 1 when an operation results in carrying from or borrowing to bit 3.
CY: Set to 1 when an operation results in carrying from or borrowing to bit 7.

2.1 8-Bit Transfer and Input/Output Instructions

		CY	N	H	Z	CYCL	7 6	5 4 3	2 1 0
LD r, r'	r ← r'	--	--	--	--	1	01	r	r'

Loads the contents of register r' into register r.

Codes for registers r and r'

Register	r, r'
A	111
B	000
C	001
D	101
E	011
H	100
L	101

Examples: LD A, B ; A ← B
LD B, D ; B ← D

		CY	N	H	Z	CYCL	7 6	5 4 3	2 1 0
LD r, n	r ← n	--	--	--	--	2	00	r	110
							←	n	→

Loads 8-bit immediate data n into register r.

Example: L D B, 0x24 ; B ← 0x24

		CY	N	H	Z	CYCL	7 6	5 4 3	2 1 0
LD r, (HL)	r ← (HL)	--	--	--	--	2	01	r	110

Loads the contents of memory (8 bits) specified by register pair HL into register r.

Example: When (HL) = 0x5C,
LD H, (HL) ; H ← 0x5C

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
LD (HL), r	(HL) ← r	--	--	--	--	2	01	110	r

Stores the contents of register r in memory specified by register pair HL.

Example: When A = 0x3CH, HL = 0x8AC5
LD (HL), A ; (0x8AC5) ← 0x3C

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
LD (HL), n	(HL) ← n	--	--	--	--	3	00	110	110
							←	n	→

Loads 8-bit immediate data n into memory specified by register pair HL.

Example: When HL = 0x8AC5,
LD (HL), 0 ; 0x8AC5 ← 0

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
LD A, (BC)	A ← (BC)	--	--	--	--	2	00	001	010

Loads the contents specified by the contents of register pair BC into register A.

Example: When (BC) = 0x2F,
LD A, (BC) ; A ← 0x2F

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
LD A, (DE)	A ← (DE)	--	--	--	--	2	00	011	010

Loads the contents specified by the contents of register pair DE into register A.

Example: When (DE) = 0x5F,
LD A, (DE) ; A ← 0x5F

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
LD A, (C)	A ← (FF00H+C)	--	--	--	--	2	11	110	010

Loads into register A the contents of the internal RAM, port register, or mode register at the address in the range 0xFF00-0xF FFF specified by register C.

Example: When C = 0x95,
LD A, (C) ; A ← contents of (0xFF95)

			CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0	
	LD (C), A	(FF00H+C) ← A	--	--	--	--	2	11 100 010	

Loads the contents of register A in the internal RAM, port register, or mode register at the address in the range 0xFF00-0xFFFF specified by register C.

Example: When C = 0x9F,
LD (C), A ; (0xFF9F) ← A

			CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0	
	LD A, (n)	A ← (n)	--	--	--	--	3	11 110 000	
								← n →	

Loads into register A the contents of the internal RAM, port register, or mode register at the address in the range 0xFF00-0xFFFF specified by the 8-bit immediate operand n.

Note, however, that a 16-bit address should be specified for the mnemonic portion of n, because only the lower-order 8 bits are automatically reflected in the machine language.

Example: To load data at 0xFF34 into register A, type the following.
LD A, (FF34)

Typing only LD A, (34) would cause the address to be incorrectly interpreted as 0034, resulting in the instruction LD A, (0034) .

			CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0	
	LD (n), A	(n) ← A	--	--	--	--	3	11 100 000	
								← n →	

Loads the contents of register A to the internal RAM, port register, or mode register at the address in the range 0xFF00-0xFFFF specified by the 8-bit immediate operand n.

Note, however, that a 16-bit address should be specified for the mnemonic portion of n, because only the lower-order 8 bits are automatically reflected in the machine language.

Example: To load the contents of register A in 0xFF34, type the following.
LD (FF34), A

Typing only LD (34), A would cause the address to be incorrectly interpreted as 0034, resulting in the instruction LD (0034), A .

			CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0	
	LD A, (nn)	A ← (nn)	--	--	--	--	4	11 111 010	
								← n →	
								← n →	

Loads into register A the contents of the internal RAM or register specified by 16-bit immediate operand nn.

Example: LD A, (0xFF44) ; A ← (LY)
LD A, (0x8000) ; A ← (0x8000)

		CY	N	H	Z	CYCL	7 6	5 4 3	2 1 0
LD (nn), A	(nn) ← A	--	--	--	--	4	11	101	010
							← n →		
							← n →		

Loads the contents of register A to the internal RAM or register specified by 16-bit immediate operand nn.

Example: LD (0xFF44), A ; (LY) ← A
LD (0x8000), A ; (0x8000) ← A

		CY	N	H	Z	CYCL	7 6	5 4 3	2 1 0
LD A, (HLI)	A ← (HL) HL ← HL+1	--	--	--	--	2	00	101	010

Loads in register A the contents of memory specified by the contents of register pair HL and simultaneously increments the contents of HL.

Example: When HL = 0x1FF and (0x1FF) = 0x56,
LD A, (HLI) ; A ← 0x56, HL ← 0x200

		CY	N	H	Z	CYCL	7 6	5 4 3	2 1 0
LD A, (HLD)	A ← (HL) HL ← HL-1	--	--	--	--	2	00	111	010

Loads in register A the contents of memory specified by the contents of register pair HL and simultaneously decrements the contents of HL.

Example: When HL = 0x8A5C and (0x8A5C) = 0x3C,
LD A, (HLD) ; A ← 0x3C, HL ← 0x8A5B

		CY	N	H	Z	CYCL	7 6	5 4 3	2 1 0
LD (BC), A	(BC) ← A	--	--	--	--	2	00	000	010

Stores the contents of register A in the memory specified by register pair BC.

Example: When BC = 0x205F and A = 0x3F,
LD (BC) , A ; (0x205F) ← 0x3F

		CY	N	H	Z	CYCL	7 6	5 4 3	2 1 0
LD (DE), A	(DE) ← A	--	--	--	--	2	00	010	010

Stores the contents of register A in the memory specified by register pair DE.

Example: When DE = 0x205C and A = 0x00,
LD (DE) , A ; (0x205C) ← 0x00

Instruction	Operation	CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
LD (HLI), A	(HL) ← A HL ← HL+1	--	--	--	--	2	00	100	010

Stores the contents of register A in the memory specified by register pair HL and simultaneously increments the contents of HL.

Example: When HL = 0xFFFF and A = 0x56,
LD (HLI), A ; (0xFFFF) ← 0x56, HL = 0x0000

Instruction	Operation	CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
LD (HLD), A	(HL) ← A HL ← HL-1	--	--	--	--	2	00	110	010

Stores the contents of register A in the memory specified by register pair HL and simultaneously decrements the contents of HL.

Example: HL = 0x4000 and A = 0x5,
LD (HLD), A ; (0x4000) ← 0x5, HL = 0x3FFF

2.2 16-Bit Transfer Instructions

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
LD dd, nn	dd ← nn	--	--	--	--	3	00	dd0	001
						L-ADRS	← n →		
						H-ADRS	← n →		

Loads 2 bytes of immediate data to register pair dd.

dd codes are as follows:

Register Pair	dd
BC	00
DD	01
HL	10
SP	11

Example: LD HL, 0x3A5B ; H ← 0x3A, L ← 0x5B

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
LD SP, HL	SP ← HL	--	--	--	--	2	11	111	001

Loads the contents of register pair HL in stack pointer SP.

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
PUSH qq	(SP - 1) ← qqH (SP - 2) ← qqL SP ← SP - 2	--	--	--	--	4	11	qq0	101

Pushes the contents of register pair qq onto the memory stack. First 1 is subtracted from SP and the contents of the higher portion of qq are placed on the stack. The contents of the lower portion of qq are then placed on the stack. The contents of SP are automatically decremented by 2.

qq codes are as follows:

Register Pair	qq
BC	00
DE	01
HL	10
AF	11

Example: When SP = 0xFFFE,
PUSH BC ; (0xFFFC), (0xFFFC) ← B, SP ← 0xFFFC

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
POP qq	qqL ← (SP) qqH ← (SP+1) SP ← SP+2	--	--	--	--	3	11	qq0	001

Pops contents from the memory stack and into register pair qq.
First the contents of memory specified by the contents of SP are loaded in the lower portion of qq. Next, the contents of SP are incremented by 1 and the contents of the memory they specify are loaded in the upper portion of qq. The contents of SP are automatically incremented by 2.

Example: When SP = 0xFFFC, (0xFFFC) = 0x5F, and (0xFFFD) = 0x3C,
POP BC ; B ← 0x3C, C ← 0x5F, SP ← 0xFFFE

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
LDHL SP, e	HL ← SP+e	*	*	0	0	3	11	111	000
						* Varies with instruction results	←	e	→

e = -128 to +127

The 8-bit operand e is added to SP and the result is stored in HL.

Flag
Z: Reset
H: Set if there is a carry from bit 11; otherwise reset.
N: Reset
CY: Set if there is a carry from bit 15; otherwise reset.

Example: When SP = 0xFFF8,
LDHL SP, 2 ; HL ← 0xFFFA, CY ← 0, H ← 0, N ← 0, Z ← 0

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
LD (nn), SP	(nn) ← SP_L	--	--	--	--	5	00	001	000
	(nn+1) ← SP_H					L-ADRS	←	n	→
						H-ADRS	←	n	→

Stores the lower byte of SP at address nn specified by the 16-bit immediate operand nn and the upper byte of SP at address nn + 1.

Example: When SP = 0xFFF8,
LD (0xC100) , SP ; 0xC100 ← 0xF8
0xC101← 0xFF

2.3 8-Bit Arithmetic and Logical Operation Instructions

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
ADD A, r	A ← A + r	*	*	0	*	1	10	000	r

Adds the contents of register r to those of register A and stores the results in register A.

Flag Z: Set if the result is 0; otherwise reset.
H: Set if there is a carry from bit 3; otherwise reset.
N: Reset
CY: Set if there is a carry from bit 7; otherwise reset.

Example: When A = 0x3A and B = 0xC6,
ADD A, B ; A ← 0, Z ← 1, H ← 1, N ← 0, CY ← 1

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
ADD A, n	A ← A + n	*	*	0	*	2	11	000	110
							←	n	→

Adds 8-bit immediate operand n to the contents of register A and stores the results in register A..

Example: When A = 0x3C,
ADD A. 0xFF ; A ← 0x3B, Z ← 0, H ← 1, N ← 0, CY ← 1

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
ADD A, (HL)	A ← A + (HL)	*	*	0	*	2	10	000	110

Adds the contents of memory specified by the contents of register pair HL to the contents of register A and stores the results in register A..

Example: When A = 0x3C and (HL) = 0x12,
ADD A, (HL) ; A ← 0x4E, Z ← 0, H ← 0, N ← 0, CY ← 0

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
ADC A, s	A ← A+s+CY	*	*	0	*	--	--	–	--

Adds the contents of operand s and CY to the contents of register A and stores the results in register A..
r, n, and (HL) are used for operand s.

	CYCL	7 6	5 4 3	2 1 0
ADC A, r	1	10	001	r
ADC A, n	2	11	001	110
		←	n□	→□
ADC A, (HL)	2	10	001	110

Examples: When A = 0xE1, E = 0x0F, (HL) = 0x1E, and CY = 1,
ADC A, E ; A ← 0xF1, Z ← 0, H ← 1, CY ← 0
ADC A, 0x3B ; A ← 0x1D, Z ← 0, H ← 0, CY ← -1
ADC A, (HL) ; A ← 0x00, Z ← 1, H ← 1, CY ← 1

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
SUB s	A ← A-s	*	*	1	*	--	--	–	--

Subtracts the contents of operand s from the contents of register A and stores the results in register A.
r, n, and (HL) are used for operand s.

	CYCL	7 6	5 4 3	2 1 0
SUB r	1	10	010	r
SUB n	2	11	010	110
		←	n	→
SUB (HL)	2	10	010	110

Flag Z: Set if result is 0; otherwise reset.
H: Set if there is a borrow from bit 4; otherwise reset.
N: Set
CY: Set if there is a borrow; otherwise reset.

Examples: When A = 0x3E, E = 0x3E, and (HL) = 0x40,
SUB E ; A ← 0x00, Z ← 1, H ← 0, N ← 1 CY ← 0
SUB 0x0F ; A ← 0x2F, Z ← 0, H ← 1, N ← 1 CY ← 0
SUB (HL) ; A ← 0xFE, Z ← 0, H ← 0, N ← 1 CY ← 1

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
SBC A, s	A ← A-s-CY	*	*	1	*	--	--	–	--

Subtracts the contents of operand s and CY from the contents of register A and stores the results in register A.
r, n, and (HL) are used for operand s.

	CYCL	7 6	5 4 3	2 1 0
SBC A, r	1	10	011	r
SBC A, n	2	11	011	110
		←	n	→
SBC A. (HL)	2	10	011	110

Examples: When A = 0x3B, (HL) = 0x4F, H = 0x2A, and CY = 1,
SBC A, H ; A ← 0x10, Z ← 0, H ← 0, N ← 1 CY ← 0
SBC A, 0x3A ; A ← 0x00, Z ← 1, H ← 0, N ← 1 CY ← 0
SBC A, (HL) ; A ← 0xEB, Z ← 0, H ← 1, N ← 1 CY ← 1

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
AND s	A ← A ∧ s	0	1	0	*	--	--	–	--

Takes the logical-AND for each bit of the contents of operand s and register A. and stores the results in register A.
r, n, and (HL) are used for operand s.

	CYCL	7 6	5 4 3	2 1 0
AND r	1	10	100	r
AND n	2	11	100	110
			←— n —→	
AND (HL)	2	10	100	110

Examples: When A= 0x5A, L = 0x3F and (HL) = 0x0,
AND L ; A ← 0x1A, Z ← 0, H ← 1, N ← 0 CY ← 0
AND 0x38 ; A ← 0x18, Z ← 0, H ← 1, N ← 0 CY ← 0
AND (HL) ; A ← 0x00, Z ← 1, H ← 1, N ← 0 CY ← 0

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
OR s	AVs	0	0	0	*	--	--	–	–

Takes the logical-OR for each bit of the contents of operand s and register A and stores the results in register A. r, n, and (HL) are used for operand s.

	CYCL	7 6	5 4 3	2 1 0
OR r	1	10	110	r
OR n	2	11	110	110
			←— n —→	
OR (HL)	2	10	110	110

Examples: When A= 0x5A, (HL) = 0x0F,
OR A ; A ← 0x5A, Z ← 0
OR 3 ; A ← 0x5B, Z ← 0
OR (HL) ; A ← 0x5F, Z ← 0

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
XOR s	A ⊕ s	0	0	0	*	--	--	–	--

Takes the logical exclusive-OR for each bit of the contents of operand s and register A. and stores the results in register A. r, n, and (HL) are used for operand s.

	CYCL	7 6	5 4 3	2 1 0
XOR r	1	10	101	r
XOR n	2	11	101	110
			←— n —→	
XOR (HL)	2	10	101	110

Examples: When A = 0xFF and (HL) = 0x8A,
XOR A ; A ← 0x00, Z ← 1
XOR 0x0F ; A ← 0xF0, Z ← 0
XOR (HL) ; A ← 0x75, Z ← 0

		CY	N	H	Z	CYCL	7 6	5 4 3	2 1 0
CP s	A — s	*	*	1	*	--	--	–	--

Compares the contents of operand s and register A and sets the flag if they are equal.
r, n, and (HL) are used for operand s.

	CYCL	7 6	5 4 3	2 1 0
CP r	1	10	111	r
CP n	2	11	111	110
		← n →		
CP (HL)	2	10	111	110

Examples: When A = 0x3C, B = 0x2F, and (HL) = 0x40,
CP B ; Z ← 0, H ← 1, N ← 1, CY ← 0
CP 0x3C ; Z ← 1, H ← 0, N ← 1, CY ← 0
CP (HL) ; Z ← 0, H ← 0, N ← 1, CY ← 1

		CY	N	H	Z	CYCL	7 6	5 4 3	2 1 0
INC r	r ← r + 1	--	*	0	*	1	00	r	100

Increments the contents of register r by 1.

Example: When A = 0xFF,
INC A ; A ← 0, Z ← 1, H ← 1, N ← 0

		CY	N	H	Z	CYCL	7 6	5 4 3	2 1 0
INC (HL)	(HL) ← (HL) + 1	--	*	0	*	3	00	110	100

Increments by 1 the contents of memory specified by register pair HL.

Example: When (HL) = 0x50,
INC (HL) ; (HL) ← 0x51, Z ← 0, H ← 0, N ← 0

		CY	N	H	Z	CYCL	7 6	5 4 3	2 1 0
DEC r	r ← r - 1	--	*	1	*	1	00	r	101

Subtract 1 from the contents of register r by 1.

Example: When L = 0x01,
DEC L ; L ← 0, Z ← 1, H ← 0, N ← 1

			CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0	
	DEC (HL)	(HL) ← (HL) - 1	--	*	1	*	3	00 110 101	

Increments by 1 the contents of memory specified by register pair HL.

Example: When (HL) = 0x00,
DEC (HL) ; (HL) ← 0xFF, Z ← 0, H ← 1, N ← 1

2.4 16-Bit Arithmetic Operation Instructions

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
ADD HL, ss	HL ← HL + ss	*	*	0	--	2	00	ss1	001

Adds the contents of register pair ss to the contents of register pair HL and stores the results in HL.

ss codes are as follows.

Register Pair	ss
BC	00
DE	01
HL	10
SP	11

Flag Z: No change
H: Set if there is a carry from bit 11; otherwise reset.
N: Rest
CY: Set if there is a carry from bit 15; otherwise reset.

Example: When HL = 0x8A23, BC = 0x0605,
ADD HL, BC ; HL ← 0x9028, H ← 1, N ← 0, CY ← 0
ADD HL, HL ; HL ← 0x1446, H ← 1, N ← 0, CY ← 1

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
ADD SP, e	SP ← SP + e	*	*	0	0	4	11	101	000
							←	e	→

Adds the contents of the 8-bit immediate operand e and SP and stores the results in SP.

Example: SP = 0xFFF8
ADD SP, 2 ; SP ← 0xFFFA, CY ← 0, H ← 0, N ← 0, Z ← 0

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
INC ss	ss ← ss + 1	--	--	--	--	2	00	ss0	011

Increments the contents of register pair ss by 1.

Example: When DE = 0x235F,
INC DE ; DE ← 0x2360

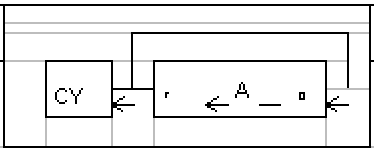
		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
DEC ss	ss ← ss - 1	--	--	--	--	2	00	ss1	011

Decrements the contents of register pair ss by 1.

Example: When DE = 0x235F,
DEC DE ; DE ← 0x235E

2.5 Rotate Shift Instructions

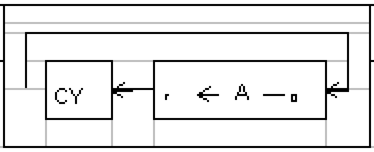
	CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
RLCA	A_r	0	0	0	1	00	000	111



Rotates the contents of register A to the left.
That is, the contents of bit 0 are copied to bit 1 and the previous contents of bit 1 (the contents before the copy operation) are copied to bit 2. The same operation is repeated in sequence for the rest of the register. The contents of bit 7 are placed in both CY and bit 0 of register A..

Example: When A = 0x85 and CY = 0,
RLCA ; A ← 0x0A, CY ← 1, Z ← 0, H ← 0, N ← 0

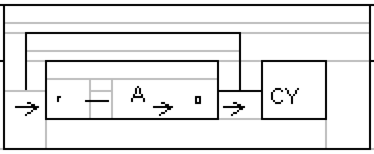
	CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
RLA	A_r	0	0	0	1	00	010	111



Rotates the contents of register A to the left.

Example: When A = 0x95 and CY = 1,
RLA ; A ← 0x2B, C ← 1, Z ← 0, H ← 0, N ← 0

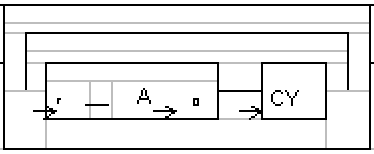
	CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
RRCA	A_0	0	0	0	1	00	001	111



Rotates the contents of register A to the right.

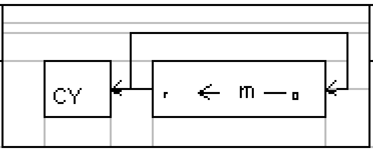
Example: When A = 0x3B and CY = 0,
RRCA ; A ← 0x9D, CY ← 1, Z ← 0, H ← 0, N ← 0

	CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
RRA	A_0	0	0	0	1	00	011	111



Rotates the contents of register A to the right.

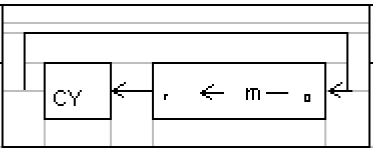
Example: When A = 0x81 and CY = 0,
RRA ; A ← 0x40, CY ← 1, Z ← 0, H ← 0, N ← 0

			CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
	RLC m		mr	0	0	*	--	-- – --

Rotates the contents of operand m to the left.
r and (HL) are used for operand m.

			CYCL	7 6 5 4 3 2 1 0
	RLC r	2	11 001 011	
			00 000 r	
	RLC (HL)	4	11 001 011	
			00 000 110	

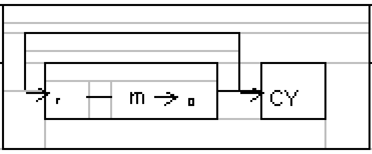
Examples: When B = 0x85, (HL) = 0, and CY = 0,
RLC B ; B ← 0x0B, CY ← 1, Z ← 0, H ← 0, N ← 0
RLC (HL) ; (HL) ← 0x00, CY ← 0, Z ← 1, H ← 0, N ← 0

			CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
	RL m		mr	0	0	*	--	-- – --

Rotates the contents of operand m to the left.
r and (HL) are used for operand m.

			CYCL	7 6 5 4 3 2 1 0
	RL r	2	11 001 011	
			00 010 r	
	RL (HL)	4	11 001 011	
			00 010 110	

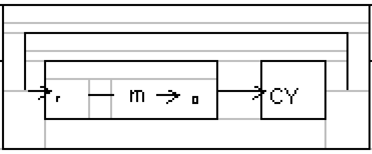
Examples: When L = 0x80, (HL) = 0x11, and CY = 0,
RL L ; L ← 0x00, CY ← 1, Z ← 1, H ← 0, N ← 0
RL (HL) ; (HL) ← 0x22, CY ← 0, Z ← 0, H ← 0, N ← 0

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
RRC m		m₀	0	0	*	--	--	–	--

Rotates the contents of operand m to the right.
r and (HL) are used for operand m.

	CYCL	7 6	5 4 3	2 1 0
RRC r	2	11	001	011
		00	001	r
RRC (HL)	4	11	001	011
		00	001	110

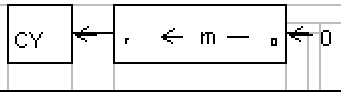
Examples: When C = 0x1, (HL) = 0x0, CY = 0,
RRC C ; C ← 0x80, CY ← 1, Z ← 0, H ← 0, N ← 0
RRC (HL) ; (HL) ← 0x00, CY ← 0, Z ← 1, H ← 0, N ← 0

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
RR m		m₀	0	0	*	--	--	–	–

Rotates the contents of operand m to the right.
r and (HL) are used for operand m.

	CYCL	7 6	5 4 3	2 1 0
RR r	2	11	001	011
		00	011	r
RR (HL)	4	11	011	011
		00	011	110

Examples: When A= 0x1, (HL) = 0x8A, CY = 0,
RR A ; A ← 0x00, CY ← 1, Z ← 1, H ← 0, N ← 0
RR (HL) ; (HL) ← 0x45, CY ← 0, Z ← 0, H ← 0, N ← 0

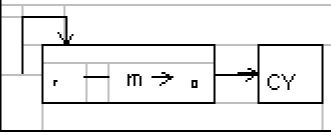
		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
SLA m		m_7	0	0	*	--	--	–	--
									

Shifts the contents of operand m to the left. That is, the contents of bit 0 are copied to bit 1 and the previous contents of bit 1 (the contents before the copy operation) are copied to bit 2. The same operation is repeated in sequence for the rest of the operand. The content of bit 7 is copied to CY, and bit 0 is reset.

r and (HL) are used for operand m.

	CYCL	7 6	5 4 3	2 1 0
SLA r	2	11	001	011
		00	100	r
SLA (HL)	4	11	011	011
		00	100	110

Examples: When D = 0x80, (HL) = 0xFF, and CY = 0,
SLA D ; D ← 0x00, CY ← 1, Z ← 1, H ← 0, N ← 0
SLA (HL) ; (HL) ← 0xFE, CY ← 1, Z ← 0, H ← 0, N ← 0

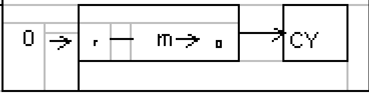
		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
SRA m		m_0	0	0	*	--	--	–	--
									

Shifts the contents of operand m to the right. That is, the contents of bit 7 are copied to bit 6 and the previous contents of bit 6 (the contents before the copy operation) are copied to bit 5. The same operation is repeated in sequence for the rest of the operand . The contents of bit 0 are copied to CY, and the content of bit 7 is unchanged.

r and (HL) are used for operand m.

	CYCL	7 6	5 4 3	2 1 0
SRA r	2	11	001	011
		00	101	r
SRA (HL)	4	11	001	011
		00	101	110

Example: When A = 0x8A, (HL) = 0x01, and CY = 0,
SRA D ; A ← 0xC5, CY ← 0, Z ← 0, H ← 0, N ← 0
SRA (HL) ; (HL) ← 0x00, CY ← 1, Z ← 1, H ← 0, N ← 0

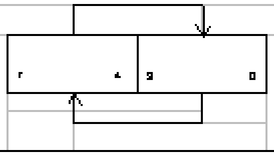
		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
SRL m		m_0	0	0	*	--	-- – --

Shifts the contents of operand m to the right. That is, the contents of bit 7 are copied to bit 6 and the previous contents of bit 6 (the contents before the copy operation) are copied to bit 5. The same operation is repeated in sequence for the rest of the operand . The contents of bit 0 are copied to CY, and bit 7 is reset.

r and (HL) are used for operand m.

	CYCL	7 6 5 4 3 2 1 0
SRL r	2	11 001 011
		00 111 r
SRL (HL)	4	11 001 011
		00 111 110

Examples: When A= 0x01, (HL) = 0xFF, CY + 0,
SRL A ; A← 0x00, CY← 1, Z← 1, H← 0, N← 0
SRL (HL) ; (HL)← 0x7F, CY← 1, Z← 0, H← 0, N← 0

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
SWAP m		0	0	0	*	--	-- – --

Shifts the contents of the lower-order 4 bits (0-3) of operand m unmodified to the higher-order 4 bits (4-7) of that operand and shifts the contents of the higher-order 4 bits to the lower-order 4 bits.
r and (HL) are used for operand m.

	CYCL	7 6 5 4 3 2 1 0
SWAP r	2	11 001 011
		00 110 r
SWAP (HL)	4	11 001 011
		00 110 110

Examples: When A= 0x00 and (HL) = 0xF0,
SWAP A ; A← 0x00, Z ← 1, H ← 0, N ← 0, CY ← 0
SWAP (HL) ; (HL) ← 0x0F, Z← 0, H ← 0, N ← 0, CY ← 0

2.6 Bit Operations

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
BIT b, r	Z ← $\overline{r_b}$	--	1	0	$\overline{r_b}$	2	11	001	011
							01	b	r

Copies the complement of the contents of the specified bit in register r to the Z flag of the program status word (PSW).

The codes for b and r are as follows.

Bit	b	Register	r
0	000	A	111
1	001	B	000
2	010	C	001
3	011	D	010
4	100	E	011
5	101	H	100
6	110	L	101
7	111		

Examples: When A= 0x80 and L = 0xEF
BIT 7, A ; Z ← 0, H ← 1, N ← 0
BIT 4, L ; Z ← 1, H ← 1, N ← 0

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
BIT b, (HL)	Z ← $\overline{(HL)}_b$	--	1	0	$\overline{(HL)}_b$	3	11	001	011
							01	b	110

Copies the complement of the contents of the specified bit in memory specified by the contents of register pair HL to the Z flag of the program status word (PSW).

Examples: When (HL) = 0xFE,
BIT 0, (HL) ; Z ← 1, H ← 1, N ← 0
BIT 1, (HL) ; Z ← 0, H ← 1, N ← 0

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
SET b, r	r_b ← 1	--	--	--	--	2	11	001	011
							11	b	r

Sets to 1 the specified bit in specified register r.

Example: When A= 0x80 and L = 0x3B,
SET 3, A ; A ← 0x84
SET 7, L ; L ← 0xBB

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
SET b, (HL)	$(HL)_b \leftarrow 1$	--	--	--	--	4	11	001	011
							11	b	110

Sets to 1 the specified bit in the memory contents specified by registers H and L.

Example: When 0x00 is the memory contents specified by H and L,
SET 3, (HL) ; (HL) ← 04H

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
RES b, r	$r_b \leftarrow 0$	--	--	--	--	2	11	001	011
							10	b	r

Resets to 0 the specified bit in the specified register r.

Example: When A= 0x80 and L = 0x3B,
RES 7, A ; A ← 0x00
RES 1, L ; L ← 0x39

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
RES b, (HL)	$(HL)_b \leftarrow 0$	--	--	--	--	4	11	001	011
							10	b	110

Resets to 0 the specified bit in the memory contents specified by registers H and L.

Example: When 0xFF is the memory contents specified by H and L,
RES 3, (HL) ; (HL) ← 0xF7

2.7 Jump Instructions

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
JP nn	PC ← nn	--	--	--	--	4	11 000 011
						L - ADRS	← n →
						H - ADRS	← n →

Loads the operand nn to the program counter (PC).
nn specifies the address of the subsequently executed instruction.
The lower-order byte is placed in byte 2 of the object code and the higher-order byte is placed in byte 3.

Example: JP 8000H ; Jump to 0x8000.

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
JP cc, nn	If cc true, PC ← nn	--	--	--	--	4/3	11 0cc 010
						L - ADRS	← n →
						H - ADRS	← n →

*Cycle no. is 3 when cc nonmatching

Loads operand nn in the PC if condition cc and the flag status match.
The subsequent instruction starts at address nn.
If condition cc and the flag status do not match, the contents of the PC are incremented, and the instruction following the current JP instruction is executed.

The relation between conditions and cc codes are as follows.

cc	Condition	Flag
00	NZ	Z = 0
01	Z	Z = 1
10	NC	CY = 0
11	C	CY = 1

Example: When Z = 1 and C = 0,

```
          JP  NZ,  8000H ;      Moves to next instruction after 3 cycles.
          JP    Z,  8000H ;     Jumps to address 0x8000.
          JP    C, 8000H ;      Moves to next instruction after 3 cycles.
          JP  NC,  8000H ;      Jumps to address 0x8000.
```

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
JR e	PC ← PC + e	--	--	--	--	3	00 011 000
							← e - 2 →

e = -127 to +129

Jumps -127 to +129 steps from the current address.

			CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
	JR cc, e	If cc true, PC ← PC + e	--	--	--	--	3/2	00	1cc	000
								← e - 2 →		

e = -127 to +129

If condition cc and the flag status match, jumps -127 to +129 steps from the current address. If cc and the flag status do not match, the instruction following the current JP instruction is executed.

			CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
	JP (HL)	PC ← HL	--	--	--	--	1	11	101	001

Loads the contents of register pair HL in program counter PC.
The next instruction is fetched from the location specified by the new value of PC.

Example: When HL = 0x8000,
JP (HL) ; Jumps to 0x8000.

2.8 Call and Return Instructions

			CY	N	H	Z	CYCL	7 6 5 4 3 2 1 0
	CALL nn	$(SP - 1) \leftarrow PC_H$ $(SP - 2) \leftarrow PC_L$ $PC \leftarrow nn$ $SP \leftarrow SP-2$	--	--	--	--	6	11 001 101
							L - ADRS	⟵ n ⟶
							H - ADRS	⟵ n ⟶

In memory, pushes the PC value corresponding to the instruction at the address following that of the CALL instruction to the 2 bytes following the byte specified by the current SP. Operand nn is then loaded in the PC.

The subroutine is placed after the location specified by the new PC value.
When the subroutine finishes, control is returned to the source program using a return instruction and by popping the starting address of next instruction, which was just pushed, and moving it to the PC.

With the push, the current value of the SP is decremented by 1, and the higher-order byte of the PC is loaded in the memory address specified by the new SP value. The value of the SP is then again decremented by 1, and the lower-order byte of the PC is loaded in the memory address specified by that value of the SP.

The lower-order byte of the address is placed in byte 2 of the object code, and the higher-order byte is placed in byte 3.

Examples: When PC = 0x8000 and SP = 0xFFFE,

```
Address
0x8000    CALL  1234H    ;     Jumps to address 0x1234, and
0x8003                         (FFFDH) ← 80H
                               (FFFCH) ← 03H
                               SP ← FFFCH
```

			CY	N	H	Z	CYCL	7 6 5 4 3 2 1 0
	CALL cc, nn	If cc true, $(SP - 1) \leftarrow PC_H$ $(SP - 2) \leftarrow PC_L$ $PC \leftarrow nn$ $SP \leftarrow SP - 2$	--	--	--	--	6/3	11 0cc 100
							L-ADRS	⟵ n ⟶
							H-ADRS	⟵ n ⟶

If condition cc matches the flag, the PC value corresponding to the instruction following the CALL instruction in memory is pushed to the 2 bytes following the memory byte specified by the SP. Operand nn is then loaded in the PC.

Examples: When Z = 1,

```
Address
0x7FFC     CALL  NZ,  0x1234    ;     Moves to next instruction after 3 cycles.
0x8000     CALL   Z,  0x1234    ;     Pushes 0x8003 to the stack,
0x8003                                and jumps to 0x1234.
```

			CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
	RET	$PC_L \leftarrow (SP)$ $PC_H \leftarrow (SP + 1)$ $SP \leftarrow SP + 2$	--	--	--	--	4	11	001	001

Pops from the memory stack the PC value pushed when the subroutine was called, returning control to the source program.

In this case, the contents of the address specified by the SP are loaded in the lower-order byte of the PC, and the content of the SP is incremented by 1. The contents of the address specified by the new SP value are then loaded in the higher-order byte of the PC, and the SP is again incremented by 1. (The value of SP is 2 larger than before instruction execution.)

The next instruction is fetched from the address specified by the content of PC.

```
Examples:   Address
            8000H   CALL  9000H
            8003H
            9000H
                    RET               ; Returns to address 0x8003
```

			CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
	RETI	$PC_L \leftarrow (SP)$ $PC_H \leftarrow (SP + 1)$ $SP \leftarrow SP + 2$	--	--	--	--	4	11	011	001

Used when an interrupt-service routine finishes.
The execution of this return is as follows.

The address for the return from the interrupt is loaded in program counter PC.
The master interrupt enable flag is returned to its pre-interrupt status.

```
Examples:   0x0040
                    RETI              ; Pops the stack and returns to address 0x8001.
            8000H    INC  L           :An external interrupt occurs here.
            8001H
```

			CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
	RET	If cc true, $PC_L \leftarrow (SP)$ $PC_H \leftarrow (SP+1)$ $SP \leftarrow SP + 2$	--	--	--	--	5/2	11	0cc	000

If condition cc and the flag match, control is returned to the source program by popping from the memory stack the PC value pushed to the stack when the subroutine was called.

```
Example:    Address
            0x8000   CALL    0x9000
            0x8003

0x9000      CP       0
            RET      Z                ; Returns to address 0x8003 if Z = 1.
                                        Moves to next instruction after 2 cycles if Z = 0.
```

			CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
	RST t	(SP - 1) ← PC_H (SP - 2) ← PC_L SP ← SP - 2 PC_H ← 0 PC_L ← P	--	--	--	--	4	11 t 111

Pushes the current value of the PC to the memory stack and loads to the PC the page 0 memory addresses provided by operand t.
Then next instruction is fetched from the address specified by the new content of PC.

With the push, the content of the SP is decremented by 1, and the higher-order byte of the PC is loaded in the memory address specified by the new SP value. The value of the SP is then again decremented by 1, and the lower-order byte of the PC is loaded in the memory address specified by that value of the SP.

The RST instruction can be used to jump to 1 of 8 addresses.

Because all of the addresses are held in page 0 memory, 0x00 is loaded in the higher-order byte of the PC, and the value of P is loaded in the lower-order byte.

The relation between the t codes and P are as follows.

Operand	t	(PC_H)	P (PC_L)
0	000	0x00	0x00
1	001	0x00	0x08
2	010	0x00	0x10
3	011	0x00	0x18
4	100	0x00	0x20
5	101	0x00	0x28
6	110	0x00	0x30
7	111	0x00	0x38

```
Example:    Address
            0x8000    RST    1    ;  Pushes 0x8001 to the stack ,
            0x8001                   and jumps to 0x0008.
```

2.9 General-Purpose Arithmetic Operations and CPU Control Instructions

			CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
	DAA	Decimal adjust acc	*	0	--	*	1	00	100	111

When performing addition and subtraction, binary coded decimal representation is used to set the contents of register A to a binary coded decimal number (BCD).
The following table shows the processing that accompanies execution of the DAA instruction immediately following execution of addition (ADD and ADC)and substraction (SUB and SBC) instructions.

Instruction before Execution	CY Contents before Execution	Bits 4-7 Register A	H Contents before Execution	Bits 0-3 Register A	Number Added to Register A	CY Contents after Execution
	0	0x0- 0x9	0	0x0- 0x9	0x00	0
	0	0x0- 0x8	0	0xA-0xF	0x06	0
ADD	0	0x0- 0x9	1	0x0- 0x3	0x06	0
ADC	0	0xA-0xF	0	0x0- 0x9	0x60	1
	0	0x9-0xF	0	0xA-0xF	0x66	1
	0	0xA-0xF	1	0x0- 0x3	0x66	1
	1	0x0- 0x2	0	0x0- 0x9	0x60	1
	1	0x0- 0x2	0	0xA-0xF	0x66	1
(N = 0)	1	0x0- 0x3	1	0x0- 0x3	0x66	1
SUB	0	0x0- 0x9	0	0x0- 0x9	0x00	0
SBC	0	0x0- 0x8	1	0x6 - 0xF	0xFA	0
	1	0x7-0xF	0	0x0- 0x9	0xA0	1
(N = 1)	1	0x6-0xF	1	0x6 - 0xF	0x9A	1

Examples: When A = 0x45 and B = 0x38,

```
          ADD   A, B  ; A ← 0x7D, N ← 0
          DAA         ; A ← 0x7D + 0x06 (0x83), CY ← 0
          SUB   A, B  ; A ← 0x83 - 0x38 (0x4B), N ← 1
          DAA         ; A ← 0x4B + 0xFA (0x45)
```

			CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
	CPL	$A \leftarrow \overline{A}$	--	1	1	--	1	00	101	111

Takes the one's complement of the contents of register A.

Example: When A = 0x35,

```
         CPL     ;    A ← 0xCA
```

			CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
	NOP	No operation	--	--	--	--	1	00	000	000

Only advances the program counter by 1; performs no other operations that have an effect.

2.9 General-Purpose Arithmetic Operations and CPU Control Instructions

		CY	N	H	Z	CYCL	7 6	5 4 3	2 1 0
DAA	Decimal adjust acc	*	0	--	*	1	00	100	111

When performing addition and subtraction, binary coded decimal representation is used to set the contents of register A to a binary coded decimal number (BCD).
The following table shows the processing that accompanies execution of the DAA instruction immediately following execution of addition (ADD and ADC)and substraction (SUB and SBC) instructions.

Instruction before Execution	CY Contents before Execution	Bits 4-7 Register A	H Contents before Execution	Bits 0-3 Register A	Number Added to Register A	CY Contents after Execution
	0	0x0- 0x9	0	0x0- 0x9	0x00	0
	0	0x0- 0x8	0	0xA-0xF	0x06	0
ADD	0	0x0- 0x9	1	0x0- 0x3	0x06	0
ADC	0	0xA-0xF	0	0x0- 0x9	0x60	1
	0	0x9-0xF	0	0xA-0xF	0x66	1
	0	0xA-0xF	1	0x0- 0x3	0x66	1
	1	0x0- 0x2	0	0x0- 0x9	0x60	1
	1	0x0- 0x2	0	0xA-0xF	0x66	1
(N = 0)	1	0x0- 0x3	1	0x0- 0x3	0x66	1
SUB	0	0x0- 0x9	0	0x0- 0x9	0x00	0
SBC	0	0x0- 0x8	1	0x6 - 0xF	0xFA	0
	1	0x7-0xF	0	0x0- 0x9	0xA0	1
(N = 1)	1	0x6-0xF	1	0x6 - 0xF	0x9A	1

```
Examples:  When A= 0x45 and B = 0x38,
           ADD   A, B  ;  A← 0x7D, N ← 0
           DAA         ;  A← 0x7D + 0x06 (0x83), CY ← 0
           SUB   A, B  ;  A← 0x83 - 0x38 (0x4B), N ← 1
           DAA         ;  A← 0x4B + 0xFA (0x45)
```

		CY	N	H	Z	CYCL	7 6	5 4 3	2 1 0
CPL	A ← $\overline{A}$	--	1	1	--	1	00	101	111

Takes the one's complement of the contents of register A.

```
Example:  When A= 0x35,
          CPL     ;    A← 0xCA
```

		CY	N	H	Z	CYCL	7 6	5 4 3	2 1 0
NOP	No operation	--	--	--	--	1	00	000	000

Only advances the program counter by 1; performs no other operations that have an effect.

			CY	N	H	Z	CYCL	7 6	5 4 3	2 1 0
	HALT	Halt	--	--	--	--	1	01	110	110

After a HALT instruction is executed, the system clock is stopped and HALT mode is entered. Although the system clock is stopped in this status, the oscillator circuit and LCD controller continue to operate.

In addition, the status of the internal RAM register ports remains unchanged.

HALT mode is canceled by an interrupt or reset signal.

The program counter is halted at the step after the HALT instruction. If both the interrupt request flag and the corresponding interrupt enable flag are set, HALT mode is exited, even if the interrupt master enable flag is not set.

Once HALT mode is canceled, the program starts from the address indicated by the program counter.

If the master enable flag is set, the contents of the program counter are pushed to the stack and control jumps to the starting address of the interrupt.

If the RESET terminal goes LOW in HALT mode, the mode becomes that of a normal reset.

			CY	N	H	Z	CYCL	7 6	5 4 3	2 1 0
	STOP	Stop	--	--	--	--	1	00	010	000
								00	000	000

Execution of a STOP instruction stops both the system clock and oscillator circuit. STOP mode is entered, and the LCD controller also stops.

However, the status of the internal RAM registers ports remains unchanged.

STOP mode can be canceled by a reset signal.

If the RESET terminal goes LOW in STOP mode, it becomes that of a normal reset status.

The following conditions should be met before a STOP instruction is executed and STOP mode is entered.

- All interrupt-enable (IE) flags are reset.
- Input to P10 — P13 is LOW for all.