Part Six:

The Z-80 Instruction Set

Notation and Other Conventions

This section includes a detailed description of all the Z-80 assembly language instructions. The first line of each of these pages shows the assembly language opcode mnemonic followed by its operand(s). Some instructions have no operands at all. Other instructions have one or two operands. Anything which is capitalized should be copied exactly when you use the editor to write the assembly language source code. Anything shown in lowercase letters will be replaced by an appropriate register, number, or label. For example, the first instruction described in the eight-bit load group is:

LD r,r'

LD is the mnemonic for the Load instruction. If you wish to move the contents of register H into register A, the actual source code is

LD A,H

This should be read as "load register A with the contents of register H."

A detailed explanation of the operand notation is given below, but in general you should note that single lowercase letters are used for eight-bit numbers or registers and double lowercase letters are used for 16-bit numbers or registers. Also note that parentheses around a register pair indicates that the register pair is to be used as a pointer to a memory location. For example, the instruction INC HL means that 1 is to be added to the HL register pair. The instruction INC (HL) means that 1 will be added to a number in memory whose address is found in register pair HL.

Symbol	Specifies one of the registers
r	A, B, C, D, E, H, or L.

Symbol	Specifies a register pair
qq	BC, DE, HL, or AF
ss	BC, DE, HL, or SP
dd	BC, DE, HL, or SP
pp	BC, DE, IX, or SP
rr	BC, DE, IX, or SP

Symbol	Specifies a number or symbol in the range
n	0 to 255 (one byte)
nn	0 to 65535 (two bytes)
d	−128 to 127 (one byte)
e	−126 to 129 (one byte)

Symbol	Specifies any of the following
s	r, n, (HL), (IX + d), or (IY + d)
m	r, (HL) (IX + d), or (IY + d)
(nn)	Specifies the contents of memory location **nn**
b	Specifies an expression in the range (0,7)
cc	Specifies the state of the Flags for conditional JR, JP, CALL and RET instructions

Instruction Format Examples With Explanation

Format Example 1

LD r,(HL)

Operation: r ⇦ (HL)

This is the shorthand description of the instruction. The arrow indicates that data is moved into register r.

When you write the assembly language code, the lowercase r will be replaced by A, B, C, D, E, H or L.

Format:

Mnemonic: LD **Operands:** r,(HL)

Object Code:

0	1	r	r	r	1	1	0

The object code for this instruction is one byte long. To figure out the object code, replace bits 3, 4 and 5 with the appropriate numbers from the table. For example:

Source Code	Object Code
LD A,(HL)	01111110
LD B,(HL)	01000110
LD C,(HL)	01001110

This instruction uses two machine (M) cycles. The first machine cycle consists of four timing (T) states and the second machine cycle consists of three T states for a total of seven T states. In the TRS-80 one T state takes .5636714 microseconds because the clock speed is 1.774038 MHz, for Model I, 4 MHz for Model II and 2.02752 MHz for Model III. The execution time (E.T.), in microseconds, is calculated for the TRS-80. (One microsecond is 10^{-6} seconds or 1/1,000,000 of a second.)

Description:

The eight-bit contents of memory location (HL) are loaded into register r, where r identifies register A, B, C, D, E, H or L, assembled as follows in the object code:

Register		r
A	=	111
B	=	000
C	=	001
D	=	010
E	=	011
H	=	100
L	=	101

M cycles: 2 T states: 7(4,3) 4 MHz E.T.: 1.75

Condition Bits Affected: None

Example:

If register pair HL contains the number 75A1H, and memory address 75 A1H contains the byte 58H, the execution of

LD C, (HL)

will result in 58H in register C.

Format Example 2

JP cc,nn

Operation: IF cc TRUE, PC ◊ nn

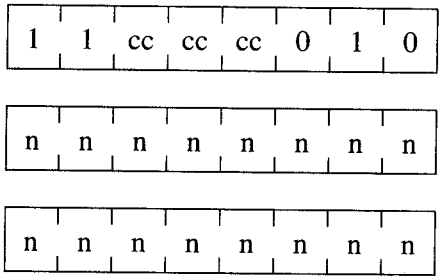
The jump is made only if the condition cc is true. The arrow indicates that the number nn is moved into the program counter PC. This will cause the program to jump to address nn.

When you write the assembly language code, cc will be replaced by one of the following: NZ, Z, NC, C, PO, PE, P or M. nn will be replaced by a number from 0 to 65535 or a label.

Format:

Mnemonic: JP **Operands:** cc, nn

Object Code:



Note: The first n operand in this assembled object code is the low order byte of a two-byte memory address.

The object code for this instruction is three bytes long. To figure out the object code, replace bits 3, 4 and 5 of the first byte with the appropriate number from the table. The second two bytes of the object code are the address being jumped to. For example:

Source Code	Object Code	
JP NZ, 0FF00H	11000010	C2H
	00000000	00H
	11111111	FFH
JP M, 1002H	11111010	FAH
	00000010	02H
	00010000	10H

Note that the low order, or right hand byte, of the address comes first in the object code.

Description:

If condition cc is true, the instruction loads operand nn into register pair PC (Program Counter), and the program continues with the instruction beginning at address nn. If condition cc is false, the Program Counter is incremented as usual, and the program continues with the next sequential instruction. Condition cc is programmed as one of eight status bits which correspond to condition bits in the Flag Register (register F). These eight status bits are defined in the table below which also specifies the corresponding cc bit fields in the assembled object code.

The Relevant Flag column shows the value the flag must have if the jump is to occur.

cc	Condition	Relevant Flag
000	NZ non zero	Z = 0
001	Z zero	Z = 1
010	NC no carry	C = 0
011	C carry	C = 1
100	PO parity odd or no overflow	P/V = 0
101	PE parity even or overflow	P/V = 1
110	P sign positive	S = 0
111	M sign negative	S = 1

M cycles: 3 T states: 10(4,3,3) 4 MHz E.T.: 2.50

Condition Bits Affected: None

Example:

If the Carry Flag (C flag in the F register) is set and the contents of address 1520 are 03H, after the execution of

JP C,1520H

the Program Counter will contain 1520H, and on the next machine cycle the CPU will fetch from address 1520H the byte 03H. In other words, program execution jumps to the instruction at 1520H.

Format Example 3

CPIR

Operation: $A - (HL), HL \Leftarrow HL + 1, BC \Leftarrow BC - 1$

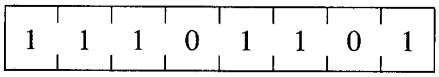
The shorthand description indicates that three different things are happening:

1. BC is decremented
2. HL is incremented
3. A byte in memory is subtracted from the A register (but the results are not saved).

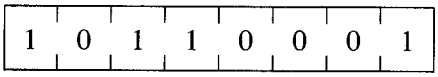
Format:

Mnemonic: CPIR **Operands:**

Object Code:



ED



B1

The assembly language instruction has no operands.

The object code is two bytes long.

Description:

The contents of the memory location addressed by the HL register pair is compared with the contents of the Accumulator. In case of a true compare, a condition bit is set. The HL is incremented and the Byte Counter (register pair BC) is decremented. If decrementing causes the BC to go to zero or if A = (HL), the instruction is terminated. If BC is not zero and A ≠ (HL), the program counter is decremented by 2 and the instruction is repeated. Note that if BC is set to zero before the execution, the instruction will loop through 64K bytes, if no match is found. Also, interrupts will be recognized after each data comparison.

For BC ≠ Ø and A ≠ (HL):

M cycles: 5 T states: 21(4,4,3,5,5) 4 MHz E.T.: 5.25

For BC = Ø or A = (HL):

M cycles: 4 T states: 16(4,4,3,5) 4 MHz E.T.: 4.ØØ

The total execution time of this instruction depends on how long it takes to find the byte being searched for and the length of the block being searched. If the instruction loops three times before BC = Ø or A = (HL), then there will be 58 (2x21 + 16) timing (T) states executed.

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if A = (HL); reset otherwise
H: Set if borrow from Bit 4; reset otherwise
P/V: Set if BC becomes zero; reset otherwise
N: Set
C: Not affected

Example:

If the HL register pair contains 1111H, the Accumulator contains F3H, the Byte Counter contains ØØØ7H, and memory locations have these contents:

(1111H) : 52H
(1112H) : ØØH
(1113H) : F3H

then after the execution of

CPIR

the contents of register pair HL will be 1114H, the contents of the Byte Counter will be 0004H. Since BC ≠ 0, the P/V flag is still set. This means that it did not search through the whole block before the instruction stopped. Since a match was found, the Z flag is set.

The CPIR instruction will affect five of the six condition codes.

Z-80 Instruction Set
Table of Contents

8 Bit Load Group

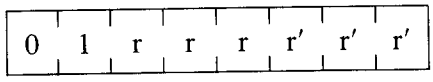
LD r,r′ LoaD

Operation: r ⇦ r′

Format:

Mnemonic: LD **Operands:** r, r′

Object Code:



Description:

The contents of any register r′ are loaded into any other register r. Note: r, r′ identifies any of the registers A, B, C, D, E, H, or L, assembled as follows in the object code:

Register		r, r′
A	=	111
B	=	000
C	=	001
D	=	010
E	=	011
H	=	100
L	=	101

M cycles: 1 T states: 4 4 MHz E.T.: 1.0

Condition Bits Affected: None

Example:

If the H register contains the number 8AH, and the E register contains 10H, the instruction

LD H,E

would result in both registers containing 10H.

LD r,n

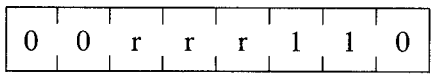
LoaD

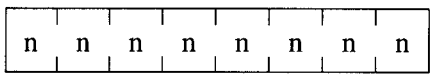
Operation: r ⟨ n

Format:

Mnemonic: LD **Operands:** r, n

Object Code:





Description:

The eight-bit integer n is loaded into any register r, where r identifies register A, B, C, D, E, H or L, assembled as follows in the object code:

Register		r
A	=	111
B	=	000
C	=	001
D	=	010
E	=	011
H	=	100
L	=	101

M cycles: 2 T states: 7(4,3) 4 MHz E.T.: 1.75

Condition Bits Affected: None

Example 1:

After the execution of

LD E,A5H

the contents of register E will be A5H.

Example 2:

After the execution of

LD A,0

register A will contain zero.

LD r,(HL)

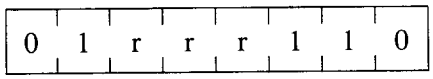
LoaD

Operation: r ◊ (HL)

Format:

Mnemonic: LD **Operands:** r, (HL)

Object Code:



Description:

The eight-bit contents of memory location (HL) are loaded into register r, where r identifies register A, B, C, D, E, H or L, assembled as follows in the object code:

Register		r
A	=	111
B	=	000
C	=	001
D	=	010
E	=	011
H	=	100
L	=	101

M cycles: 2 T states: 7(4,3) 4 MHz E.T.: 1.75

Condition Bits Affected: None

Example:

If register pair HL contains the number 75A1H, and memory address 75A1H contains the byte 58H, the execution of

LD C,(HL)

will result in 58H in register C.

LD r,(IX + d)

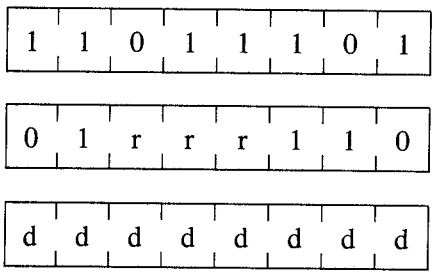
LoaD

Operation: r ◊ (IX + d)

Format:

Mnemonic: LD **Operands:** r, (IX + d)

Object Code:



Description:

The operand (IX + d) (the contents of the Index Register IX summed with a displacement integer d) is loaded into register r, where r identifies register A, B, C, D, E, H or L, assembled as follows in the object code:

Register		r
A	=	111
B	=	000
C	=	001
D	=	010
E	=	011
H	=	100
L	=	101

M cycles: 5 T states: 19(4,4,3,5,3) 4 MHz E.T.: 4.75

Condition Bits Affected: None

Example:

If the Index Register IX contains the number 25AFH, the instruction

LD B,(IX + 19H)

will cause the calculation of the sum 25AFH + 19H, which points to memory location 25C8H. If this address contains byte 39H, the instruction will result in register B also containing 39H.

A typical use of this instruction is shown below. If TABL is a location in memory this program will load the first four bytes of the table into registers A, B, C and D.

```
LD      IX, TABL        ; IX points to the table
LD      A, (IX+0)       ; Load first byte
LD      B, (IX+1)       ; Load second byte
LD      C, (IX+2)       ; Load third byte
LD      D, (IX+3)       ; Load fourth byte
```

LD r,(IY + d)

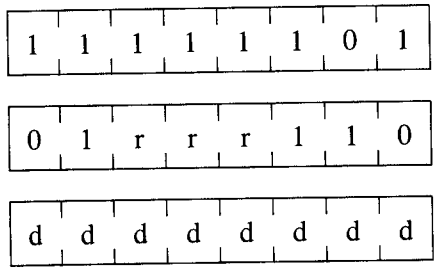
LoaD

Operation: r ◊ (IY + d)

Format:

Mnemonic: LD **Operands:** r, (IY + d)

Object Code:



Description:

The operand (IY + d) (the contents of the Index Register IY summed with a two's complement displacement integer d) is loaded into register r, where r identifies register A, B, C, D, E, H, or L, assembled as follows in the object code:

Register		r
A	=	111
B	=	000
C	=	001
D	=	010
E	=	011
H	=	100
L	=	101

M cycles: 5 T states: 19(4,4,3,5,3) 4 MHz E.T.: 4.75

Condition Bits Affected: None

Example:

If the Index Register IY contains the number 25AFH, the instruction

LD B,(IY + 19H)

will cause the calculation of the sum 25AFH + 19H, which points to memory location 25C8H. If this address contains byte 39H, the instruction will result in register B also containing 39H.

LD (HL),r

LoaD

Operation: (HL) ◊ r

Format:

Mnemonic: LD **Operands:** (HL), r

Object Code:

0	1	1	1	0	r	r	r

Description:

The contents of register r are loaded into the memory location specified by the contents of the HL register pair. The symbol r identifies register A, B, C, D, E, H or L, assembled as follows in the object code:

Register		r
A	=	111
B	=	000
C	=	001
D	=	010
E	=	011
H	=	100
L	=	101

M cycles: 2 T states: 7(4,3) 4 MHz E.T.: 1.75

Condition Bits Affected: None

Example:

If the contents of register pair HL specify memory location 2146H, and the B register contains the byte 29H, after the execution of

LD (HL),B

memory address 2146H will also contain 29H.

LD (IX + d),r

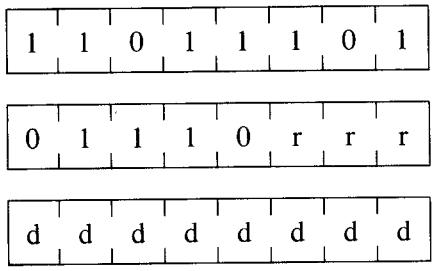
LoaD

Operation: (IX + d) ◊ r

Format:

Mnemonic: LD **Operands:** (IX + d), r

Object Code:



Description:

The contents of register r are loaded into the memory address specified by the contents of Index Register IX summed with d, a two's complement displacement integer. The symbol r identifies register A, B, C, D, E, H or L, assembled as follows in the object code:

Register		r
A	=	111
B	=	000
C	=	001
D	=	010
E	=	011
H	=	100
L	=	101

M cycles: 5 T states: 19(4,4,3,5,3) 4 MHz E.T.: 4.75

Condition Bits Affected: None

Example:

If the C register contains the byte 1CH, and the Index Register IX contains 3100H, then the instruction

LD (IX+6H), C

will perform the sum 3100H+6H and will load 1CH into memory location 3106H.

LD (IY+d),r

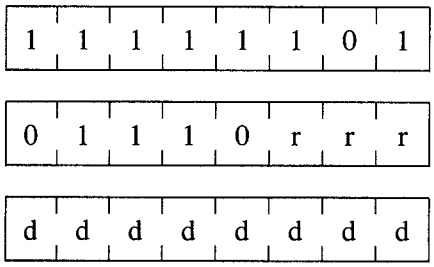
LoaD

Operation: (IY+d) ⇦ r

Format:

Mnemonic: LD **Operands:** (IY+d), r

Object Code:



Description:

The contents of register r are loaded into the memory address specified by the sum of the contents of the Index Register IY and d, a two's complement displacement integer. The symbol r is specified according to the following table.

Register		r
A	=	111
B	=	000
C	=	001
D	=	010
E	=	011
H	=	100
L	=	101

M cycles: 5 T states: 19(4,4,3,5,3) 4 MHz E.T.: 4.75

Condition Bits Affected: None

Example:

If the C register contains the byte 48H, and the Index Register IY contains 2A11H, then the instruction

LD (IY+4H),C

will perform the sum 2A11H + 4H, and will load 48H into memory location 2A15.

LD (HL),n

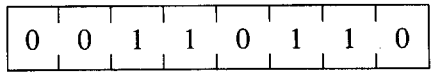
LoaD

Operation: (HL) ◊ n

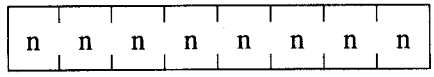
Format:

Mnemonic: LD **Operands:** (HL), n

Object Code:



36



Description:

Integer n is loaded into the memory address specified by the contents of the HL register pair.

M cycles: 3 T states: 10(4,3,3) 4 MHz E.T.: 2.50

Condition Bits Affected: None

Example:

If the HL register pair contains 4444H, the instruction
LD (HL),28H
will result in the memory location 4444H containing the byte 28H.

LD (IX + d),n

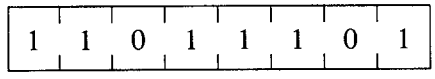
Load

Operation: (IX + d) ◊ n

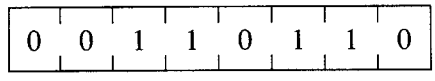
Format:

Mnemonic: LD **Operands:** (IX + d), n

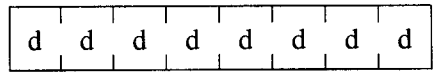
Object Code:

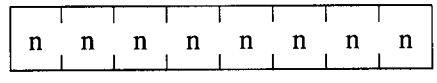


DD



36





Description:

The n operand is loaded into the memory address specified by the sum of the contents of the Index Register IX and the two's complement displacement operand d.

M cycles: 5 T states: 19(4,4,3,5,3) 4 MHz E.T.: 4.75

Condition Bits Affected: None

Example:

If the Index Register IX contains the number 219AH the instruction

LD (IX + 5H),5AH

would result in the byte 5AH in the memory address 219FH. (219FH = 219AH + 5H.)

LD (IY+d),n

LoaD

Operation: (IY+d) ⇦ n

Format:

Mnemonic: LD **Operands:** (IY+d), n

Object Code:

1	1	1	1	1	1	0	1	FD
0	0	1	1	0	1	1	0	36
d	d	d	d	d	d	d	d	
n	n	n	n	n	n	n	n	

Description:

Integer n is loaded into the memory location specified by the contents of the Index Register summed with a two's complement displacement integer d.

M cycles: 5 T states: 19(4,4,3,5,3) 4 MHz E.T.: 4.75

Condition Bits Affected: None

Example:

If the Index Register IY contains the number A940H, the instruction
LD (IY+10H),97H
would result in byte 97H in memory location A950H.

LD A,(BC)

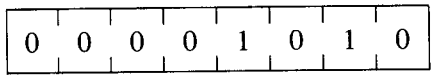
LoaD

Operation: A◊(BC)

Format:

Mnemonic: LD **Operands:** A, (BC)

Object Code:



0A

Description:

The contents of the memory location specified by the contents of the BC register pair are loaded into the Accumulator.

M cycles: 2 T states: 7(4,3) 4 MHz E.T.: 1.75

Condition Bits Affected: None

Example:

If the BC register pair contains the number 4747H, and memory address 4747H contains the byte 12H, then the instruction
LD A,(BC)
will result in byte 12H in register A.

LD A,(DE)

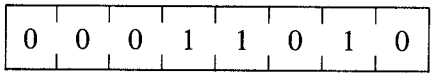
LoaD

Operation: A◊(DE)

Format:

Mnemonic: LD **Operands:** A, (DE)

Object Code:



1A

Description:

The contents of the memory location specified by the register pair DE are loaded into the Accumulator.

M cycles: 2 T states: 7(4,3) 4 MHz E.T.: 1.75

Condition Bits Affected: None

Example:

If the DE register pair contains the number 30A2H and memory address 30A2H contains the byte 22H, then the instruction

LD A,(DE)

will result in byte 22H in register A.

LD A,(nn)

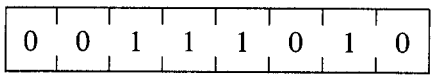
LoaD

Operation: A ◊ (nn)

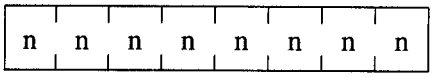
Format:

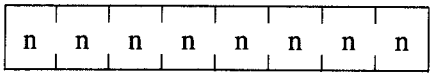
Mnemonic: LD **Operands:** A, (nn)

Object Code:



3A





Description:

The contents of the memory location specified by the operands nn are loaded into the Accumulator. The first n operand is the low order byte of a two-byte memory address.

M cycles: 4 T states: 13(4,3,3,3) 4 MHz E.T.: 3.25

Condition Bits Affected: None

Example:

If the contents of memory address 8832H is byte Ø4H, after the instruction
LD A,(8832H)
byte Ø4H will be in the Accumulator.

LD (BC),A

LoaD

Operation: (BC) ◊ A

Format:

Mnemonic: LD **Operands:** (BC), A

Object Code:

0	0	0	0	0	0	1	0

Ø2

Description:

The contents of the Accumulator are loaded into the memory location specified by the contents of the register pair BC.

M cycles: 2 T states: 7(4,3) 4 MHz E.T.: 1.75

Condition Bits Affected: None

Example:

If the Accumulator contains 7AH and the BC register pair contains 1212H the instruction
LD (BC),A
will result in 7AH being in memory location 1212H.

LD (DE),A

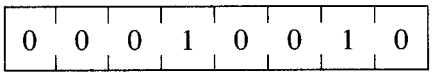
LoaD

Operation: (DE) ◊ A

Format:

Mnemonic: LD **Operands:** (DE), A

Object Code:



12

Description:

The contents of the Accumulator are loaded into the memory location specified by the DE register pair.

M cycles: 2 T states: 7(4,3) 4 MHz E.T.: 1.75

Condition Bits Affected: None

Example:

If the contents of register pair DE are 1128H, and the Accumulator contains byte A0H, the instruction

LD (DE),A

will result in A0H being in memory location 1128H.

LD (nn),A

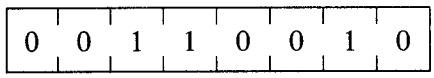
LoaD

Operation: (nn) ◊ A

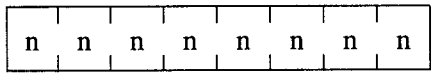
Format:

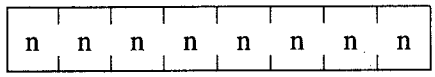
Mnemonic: LD **Operands:** (nn), A

Object Code:



32





Description:

The contents of the Accumulator are loaded into the memory address specified by the operands nn. The first n operand in the assembled object code above is the low order byte of nn.

M cycles: 4 T states: 13(4,3,3,3) 4 MHz E.T.: 3.25

Condition Bits Affected: None

Example:

If the contents of the Accumulator are byte D7H, after the execution of
LD (3141H),A
D7H will be in memory location 3141H.

LD A,I

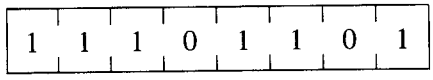
LoaD

Operation: A ⇦ I

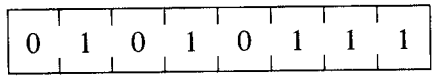
Format:

Mnemonic: LD **Operands:** A, I

Object Code:



ED



57

Description:

The contents of the Interrupt Vector Register I are loaded into the Accumulator.

M cycles: 2 T states: 9(4,5) 4 MHz E.T.: 2.25

Condition Bits Affected:

S: Set if I-Reg. is negative; reset otherwise
Z: Set if I-Reg. is zero; reset otherwise
H: Reset
P/V: Contains contents of IFF2
N: Reset
C: Not affected

Note: If an interrupt occurs during execution of this instruction, the Parity flag will contain a 0.

Example:

If the Interrupt Vector Register contains the byte 4AH, after the execution of
LD A,I
the accumulator will also contain 4AH.

LD A,R

LoaD

Operation: A ⇳ R

Format:

Mnemonic: LD **Operands:** A, R

Object Code:

1	1	1	0	1	1	0	1	ED
0	1	0	1	1	1	1	1	5F

Description:

The contents of Memory Refresh Register R are loaded into the Accumulator.

M cycles: 2 T states: 9(4,5) 4 MHz E.T.: 2.25

Condition Bits Affected:

S: Set if R-Reg. is negative; reset otherwise
Z: Set if R-Reg. is zero; reset otherwise
H: Reset
P/V: Contains contents of IFF2
N: Reset
C: Not affected

Example:

If the Memory Refresh Register contains the byte 4AH, after the execution of

LD A,R

the Accumulator will also contain 4AH.

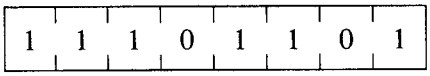
LD I,A

LoaD

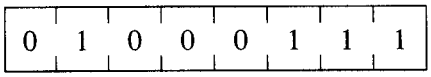
Operation: I ⇳ A

Format:

Mnemonic: LD **Operands:** I, A



ED



47

Description:

The contents of the Accumulator are loaded into the Interrupt Control Vector Register, I.

M cycles: 2 T states: 9(4,5) 4 MHz E.T.: 2.25

Condition Bits Affected: None

Example:

If the Accumulator contains the number 81H, after the instruction

LD I,A

the Interrupt Vector Register will also contain 81H.

LD R,A

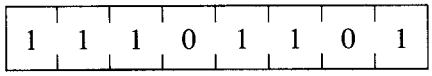
LoaD

Operation: R⇦A

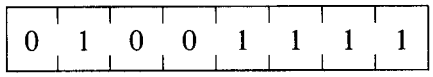
Format:

Mnemonic: LD **Operands:** R, A

Object Code:



ED



4F

Description:

The contents of the Accumulator are loaded into the Memory Refresh register R.

M cycles: 2 T states: 9(4,5) 4 MHz E.T.: 2.25

Condition Bits Affected: None

Example:

If the Accumulator contains the number B4H, after the instruction

```
LD   R,A
```

the Memory Refresh Register will also contain B4H.

16 Bit Load Group

LD dd,nn

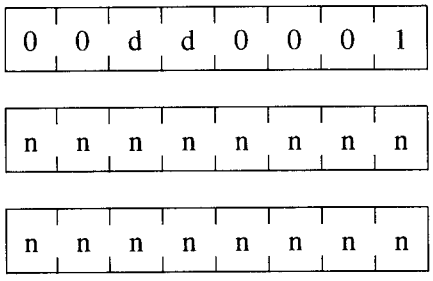
LoaD

Operation: dd ⇦ nn

Format:

Mnemonic: LD **Operands:** dd, nn

Object Code:



Description:

The two-byte integer nn is loaded into the dd register pair, where dd defines the BC, DE, HL, or SP register pairs, assembled as follows in the object code:

Pair	dd
BC	00
DE	01
HL	10
SP	11

The first n operand in the assembled object code is the low order byte.

M cycles: 3 T states: 10(4,3,3) 4 MHz E.T.: 2.50

Condition Bits Affected: None

Example:

After the execution of

LD HL,5000H

the contents of the HL register pair will be 5000H.

After the execution of
LD BC,2501H
the BC register will contain 2501H.

LD IX,nn

LoaD

Operation: IX ◊ nn

Format:

Mnemonic: LD **Operands:** IX, nn

Object Code:

1	1	0	1	1	1	0	1	DD
0	0	1	0	0	0	0	1	21
n	n	n	n	n	n	n	n	
n	n	n	n	n	n	n	n	

Description:

Integer nn is loaded into the Index Register IX. The first n operand in the assembled object code above is the low order byte.

M cycles: 4 T states: 14(4,4,3,3) 4 MHz E.T.: 3.50

Condition Bits Affected: None

Example:

After the instruction
LD IX,45A2H
the Index Register will contain integer 45A2H.

LD IY,nn

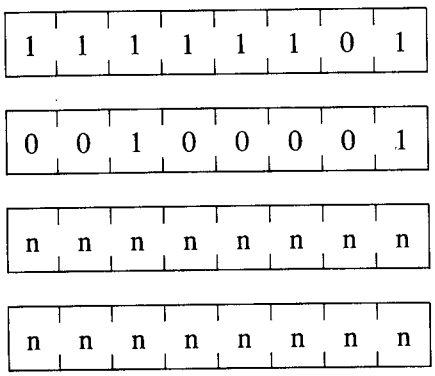
LoaD

Operation: IY ◊ nn

Format:

Mnemonic: LD **Operands:** IY, nn

Object Code:



Description:

Integer nn is loaded into the Index Register IY. The first n operand in the assembled object code above is the low order byte.

M cycles: 4 T states: 14(4,4,3,3) 4 MHz E.T.: 3.5Ø

Condition Bits Affected: None

Example:

After the instruction:
LD IY,7733H
the Index Register IY will contain the integer 7733H.

LD HL,(nn)

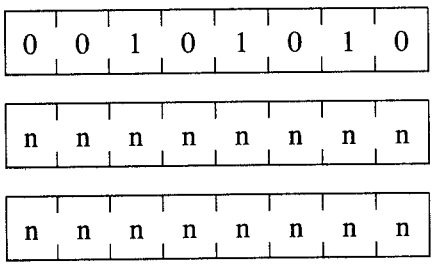
LoaD

Operation: H ◊ (nn + 1), L ◊ (nn)

Format:

Mnemonic: LD **Operands:** HL, (nn)

Object Code:



Description:

The contents of memory address nn are loaded into the low order portion of register pair HL (register L), and the contents of the next highest memory address (nn + 1) are loaded into the high order portion of HL (register H). The first n operand in the assembled object code above is the low order byte of nn.

M cycles: 5 T states: 16(4,3,3,3,3) 4 MHz E.T.: 4.00

Condition Bits Affected: None

Example:

If address 4545H contains 37H and address 4546H contains A1H, after the instruction

LD HL,(4545H)

the HL register pair will contain A137H.

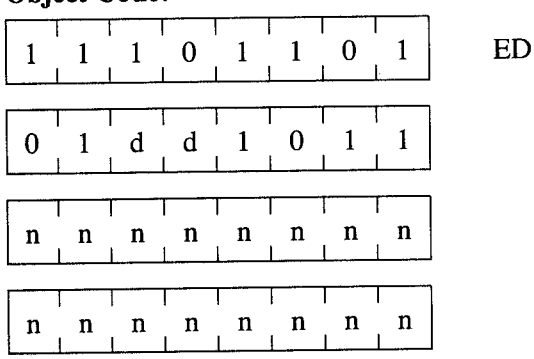
LD dd,(nn) LoaD

Operation: $dd_H \Leftarrow (nn + 1)$, $dd_L \Leftarrow (nn)$

Format:

Mnemonic: LD **Operands:** dd, (nn)

Object Code:



Description:

The contents of address nn are loaded into the low order portion of register pair dd, and the contents of the next highest memory address (nn + 1) are loaded into the high order portion of dd. Register pair dd defines BC, DE, HL, or SP register pairs, assembled as follows in the object code:

Pair	dd
BC	00
DE	01
HL	10
SP	11

The first n operand in the assembled object code above is the low order byte of (nn).

M cycles: 6 T states: 20(4,4,3,3,3,3) 4 MHz E.T.: 5.00

Condition Bits Affected: None

Example 1:

If Address 2130H contains 65H and address 2131M contains 78H after the instruction

LD BC,(2130H)

the BC register pair will contain 7865H.

Example 2:

If address FFFE contains 01H and address FFFF contains 02H, then after the instruction

LD SP,(0FFFEH)

the SP will contain 0201H.

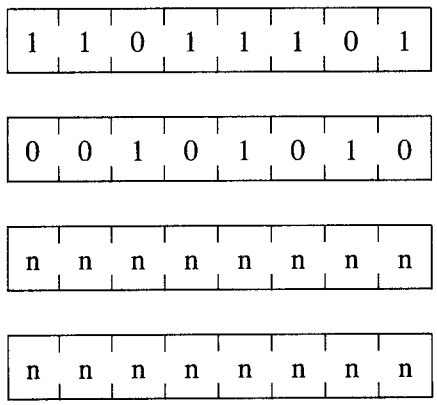
LD IX,(nn) LoaD

Operation: $IX_H \Leftarrow (nn+1)$, $IX_L \Leftarrow (nn)$

Format:

Mnemonic: LD **Operands:** IX, (nn)

Object Code:



Description:

The contents of the address nn are loaded into the low order portion of Index Register IX, and the contents of the next highest memory address (nn + 1) are loaded into the high order portion of IX. The first n operand in the assembled object code above is the low order byte of nn.

M cycles: 6 T states: 20(4,4,3,3,3,3) 4 MHz E.T.: 5.00

Condition Bits Affected: None

Example:

If address 6066H contains 92H and address 6067H contains DAH, after the instruction

LD IX,(6066H)

the Index Register IX will contain DA92H.

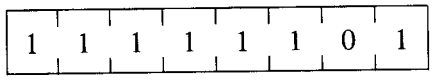
LD IY,(nn) LoaD

Operation: $IY_H \leftrightarrows (nn+1)$, $IY_L \leftrightarrows (nn)$

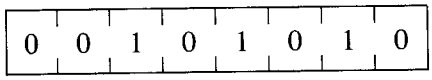
Format:

Mnemonic: LD **Operands:** IY, (nn)

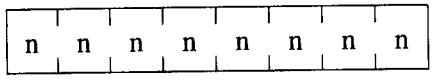
Object Code:

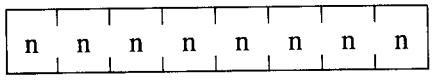


FD



2A





Description:

The contents of address nn are loaded into the low order portion of Index Register IY, and the contents of the next highest memory address (nn + 1) are loaded into the high order portion of IY. The first n operand in the assembled object code above is the low order byte of nn.

M cycles: 6 T states: 20(4,4,3,3,3,3) 4 MHz E.T.: 5.00

Condition Bits Affected: None

Example:

If address 6666H contains 92H and address 6667H contains DAH, after the instruction

LD IY,(6666H)

the Index Register IY will contain DA92H.

LD (nn),HL

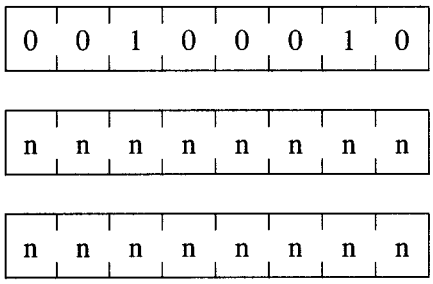
LoaD

Operation: (nn + 1) ◊H, (nn) ◊L

Format:

Mnemonic: LD **Operands:** (nn), HL

Object Code:



Description:

The contents of the low order portion of register pair HL (register L) are loaded into memory address nn, and the contents of the high order portion of HL (register H) are loaded into the next highest memory address (nn + 1). The first n operand in the assembled object code above is the low order byte of nn.

M cycles: 5 T states: 16(4,3,3,3,3) 4 MHz E.T.: 4.00

Condition Bits Affected: None

Example 1:

If the content of register pair HL is 483AH, after the instruction

LD (B229H),HL

address B229H will contain 3AH, and address B22AH will contain 48H.

Example 2:

If the register pair HL contains 504AH, then after the instruction

LD (PLACE),HL

the address PLACE will contain 4AH and address PLACE + 1 will contain 50H.

Note: PLACE is a label which must be defined elsewhere in the program.

LD (nn),dd

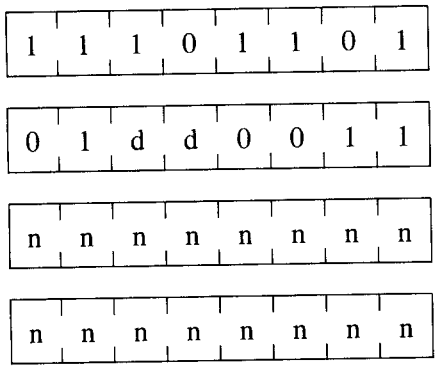
LoaD

Operation: $(nn+1) \leftarrow dd_H$, $(nn) \leftarrow dd_L$

Format:

Mnemonic: LD **Operands:** (nn), dd

Object Code:



Description:

The low order byte of register pair dd is loaded into memory address (nn); the upper order byte is loaded into memory address (nn + 1). Register pair dd defines either BC, DE, HL, or SP, assembled as follows in the object code:

Pair	dd
BC	00
DE	01
HL	10
SP	11

The first n operand in the assembled object code is the low order byte of a two byte memory address.

M cycles: 6 T states: 20(4,4,3,3,3,3) 4 MHz E.T.: 5.00

Condition Bits Affected: None

Example:

If register pair BC contains the number 4644H, the instruction

LD (1000H),BC

will result in 44H in memory location 1000H, and 46H in memory location 1001H.

LD (nn),IX

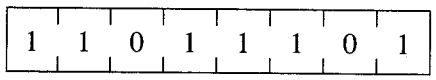
LoaD

Operation: $(nn + 1) \Leftarrow IX_H$, $(nn) \Leftarrow IX_L$

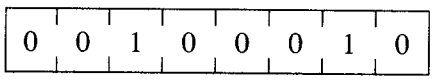
Format:

Mnemonic: LD **Operands:** (nn), IX

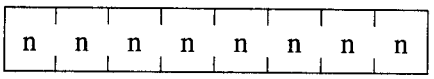
Object Code:

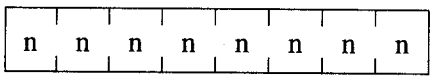


DD



22





Description:

The low order byte in Index Register IX is loaded into memory address nn; the upper order byte is loaded into the next highest address (nn + 1). The first n operand in the assembled object code above is the low order byte of nn.

M cycles: 6 T states: 20(4,4,3,3,3,3) 4 MHz E.T.: 5.00

Condition Bits Affected: None

Example:

If the Index Register IX contains 5A30H, after the instruction

LD (4392H),IX

memory location 4392H will contain number 30H and location 4393H will contain 5AH.

LD (nn),IY

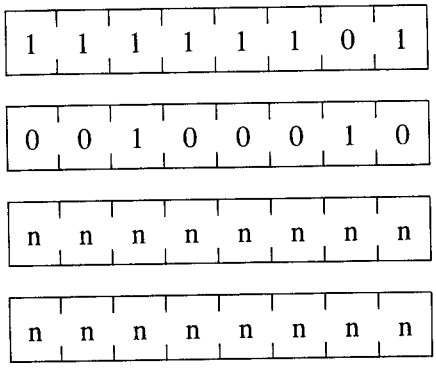
LoaD

Operation: $(nn+1) \Leftarrow IY_H$, $(nn) \Leftarrow IY_L$

Format:

Mnemonic: LD **Operands:** (nn), IY

Object Code:



Description:

The low order byte in Index Register IY is loaded into memory address nn; the upper order byte is loaded into memory location (nn + 1). The first n operand in the assembled object code above is the low order byte of nn.

M cycles: 6 T states: 20(4,4,3,3,3,3) 4 MHz E.T.: 5.00

Condition Bits Affected: None

Example:

If the Index Register IY contains 4174H after the instruction

LD 8838H,IY

memory location 8838H will contain number 74H and memory location 8839H will contain 41H.

LD SP,HL

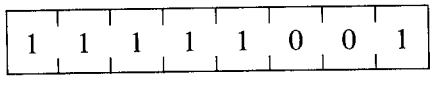
LoaD

Operation: SP ⇦ HL

Format:

Mnemonic: LD **Operands:** SP, HL

Object Code:



Description:

The contents of the register pair HL are loaded into the Stack Pointer SP.

M cycles: 1 T states: 6 4 MHz E.T.: 1.50

Condition Bits Affected: None

Example:

If the register pair HL contains 442EH, after the instruction
LD SP,HL
the Stack Pointer will also contain 442EH.

LD SP,IX

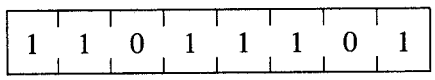
LoaD

Operation: SP ⇳ IX

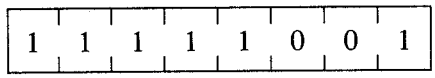
Format:

Mnemonic: LD **Operands:** SP, IX

Object Code:



DD



F9

Description:

The two-byte contents of Index Register IX are loaded into the Stack Pointer SP.

M cycles: 2 T states: 10(4,6) 4 MHz E.T.: 2.50

Condition Bits Affected: None

Example:

If the contents of the Index Register IX are 98DAH, after the instruction
LD SP,IX
the contents of the Stack Pointer will also be 98DAH.

LD SP,IY

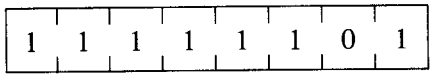
LoaD

Operation: SP ◊ IY

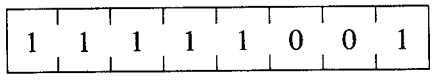
Format:

Mnemonic: LD **Operands:** SP, IY

Object Code:



FD



F9

Description:

The two byte contents of Index Register IY are loaded into the Stack Pointer SP.

M cycles: 2 T states: 10(4,6) 4 MHz E.T.: 2.50

Condition Bits Affected: None

Example:

If Index Register IY contains the integer A227H, after the instruction

LD SP,IY

the Stack Pointer will also contain A227H.

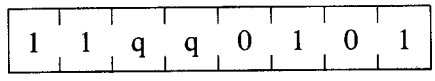
PUSH qq

Operation: $(SP-2) \lozenge qq_L$, $(SP-1) \lozenge qq_H$

Format:

Mnemonic: PUSH **Operands:** qq

Object Code:



Description:

The contents of the register pair qq are pushed into the external memory LIFO (last-in, first-out) Stack. The Stack Pointer (SP) register pair holds the 16-bit address of the current "top" of the Stack. This instruction first decrements the SP and loads the high order byte of register pair qq into the memory address now specified by the SP, then decrements the SP again and loads the low order byte of qq into the memory location corresponding to this new address in the SP. The operand qq means register pair BC, DE, HL, or AF, assembled as follows in the object code:

Pair	qq
BC	00
DE	01
HL	10
AF	11

M cycles: 3 T states: 11(5,3,3) 4 MHz E.T.: 2.75

Condition Bits Affected: None

Example:

If the AF register pair contains 2233H and the Stack Pointer contains 1007H, after the instruction

PUSH AF

memory address 1006H will contain 22H, memory address 1005H will contain 33H, and the Stack Pointer will contain 1005H. In other words the number from register pair AF is now on the top of the stack, and the stack pointer is pointing to it.

Before:

Register AF	Address	Stack
2233	1007	FF
	1008	35

Stack Pointer
1007

After: PUSH AF

Register AF	Address	Stack
2233	1005	33
	1006	22
	1007	FF
	1008	35

Stack Pointer
1005

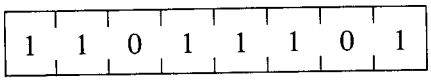
PUSH IX

Operation: $(SP-2) \Leftarrow IX_L$, $(SP-1) \Leftarrow IX_H$

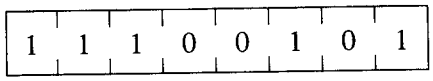
Format:

Mnemonic: PUSH **Operands:** IX

Object Code:



DD



E5

Description:

The contents of the Index Register IX are pushed into the external memory LIFO (last-in, first-out) Stack. The Stack Pointer (SP) register pair holds the 16-bit address of the current "top" of the Stack. This instruction first decrements the SP and loads the high order byte of IX into the memory address now specified by the SP, then decrements the SP again and loads the low order byte into the memory location corresponding to this new address in the SP.

M cycles: 3 T states: 15(4,5,3,3) 4 MHz E.T.: 3.75

Condition Bits Affected: None

Example:

If the Index Register IX contains 2233H and the Stack Pointer contains 1007H, after the instruction

PUSH IX

memory address 1006H will contain 22H, memory address 1005H will contain 33H, and the Stack Pointer will contain 1005H. The number from the IX register pair is now on the top of the stack.

Before:

Register IX	Address	Stack
2233	1007	FF
	1008	35

Stack Pointer

1007

After: PUSH IX

Register IX	Address	Stack
2233	1005	33
	1006	22
	1007	FF
	1008	35

Stack Pointer
1005

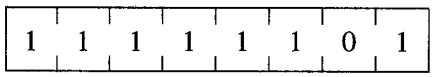
PUSH IY

Operation: $(SP-2) \Leftarrow IY_L$, $(SP-1) \Leftarrow IY_H$

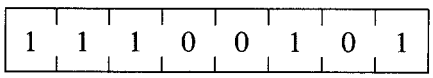
Format:

Mnemonic: PUSH **Operands:** IY

Object Code:



FD



E5

Description:

The contents of the Index Register IY are pushed into the external memory LIFO (last-in, first-out) Stack. The Stack Pointer (SP) register pair holds the 16-bit address of the current "top" of the Stack. This instruction first decrements the SP and loads the high order byte of IY into the memory address now specified by the SP; then decrements the SP again and loads the low order byte into the memory location corresponding to this new address in the SP.

M cycles: 4 T states: 15(4,5,3,3) 4 MHz E.T.: 3.75

Condition Bits Affected: None

Example:

If the Index Register IY contains 2233H and the Stack Pointer contains 1007H, after the instruction

PUSH IY

memory address 1006H will contain 22H, memory address 1005H will contain 33H, and the Stack Pointer will contain 1005H. The number from register pair IY is now on the top of the stack.

Before:

Register IY	Address	Stack
2233	1007	FF
	1008	35

Stack Pointer
1007

After: PUSH IY

Register IY	Address	Stack
2233	1005	33
	1006	22
	1007	FF
	1008	35

Stack Pointer
1005

POP qq

Operation: $qq_H \Diamond (SP + 1), qq_L \Diamond (SP)$

Format:

Mnemonic: POP **Operands:** qq

Object Code:

1	1	q	q	0	0	0	1

Description:

The top two bytes of the external memory LIFO (last-in, first-out) Stack are popped into register pair qq. The Stack Pointer (SP) register pair holds the 16-bit address of the current "top" of the Stack. This instruction first loads into the low order portion of qq, the byte at the memory location corresponding to the contents of SP; then SP is incremented and the contents of the corresponding adjacent memory location are loaded into the high order portion of qq and the SP is now incremented again. The operand qq defines register pair BC, DE, HL, or AF, assembled as follows in the object code:

Pair	r
BC	00
DE	01
HL	10
AF	11

M cycles: 3 T states: 10(4,3,3) 4 MHz E.T.: 2.50

Condition Bits Affected: None

Example:

If the Stack Pointer contains 1000H, memory location 1000H contains 55H, and location 1001H contains 33H, the instruction

POP HL

will result in register pair HL containing 3355H, and the Stack Pointer containing 1002H. In other words register pair HL contains the number which was on the top of the stack, and the stack pointer is pointing to the current top of the stack.

Before:

Register HL	Address	Stack
2233	1000	55
	1001	33
	1002	A4
	1003	62

Stack Pointer

1000

After: POP HL

Register HL	Address	Stack
3355	1002	A4
	1003	62

Stack Pointer

1002

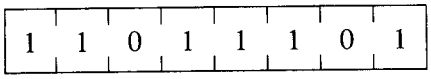
POP IX

Operation: $IX_H \Diamond (SP + 1)$, $IX_L \Diamond (SP)$

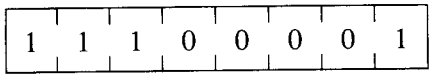
Format:

Mnemonic: POP **Operands:** IX

Object Code:



DD



E1

Description:

The top two bytes of the external memory LIFO (last-in, first-out) Stack are popped into Index Register IX. The Stack Pointer (SP) register pair holds the 16-bit address of the current "top" of the Stack. This instruction first loads into the low order portion of IX the byte at the memory location corresponding to the contents of SP; then SP is incremented and the contents of the corresponding adjacent memory location are loaded into the high order portion of IX. The SP is now incremented again.

M cycles: 4 T states: 14(4,4,3,3) 4 MHz E.T.: 3.50

Condition Bits Affected: None

Example:

If the Stack Pointer contains 1000H, memory location 1000H contains 55H, and location 1001H contains 33H, the instruction

POP IX

will result in the Index Register IX containing 3355H, and the Stack Pointer containing 1002H. Register pair IX contains the number which used to be on the top of the stack.

Before:

Register IX	Address	Stack
24F9	1000	55
	1001	33
	1002	A4
	1003	62

Stack Pointer

1000

After: POP IX

Register IX	Address	Stack
3355	1002	A4
	1003	62

Stack Pointer
1002

POP IY

Operation: $IY_H \Leftrightarrow (SP+1), IY_L \Leftrightarrow (SP)$

Format:

Mnemonic: POP **Operands:** IY

Object Code:

1	1	1	1	1	1	0	1	FD

1	1	1	0	0	0	0	1	E1

Description:

The top two bytes of the external memory LIFO (last-in, first-out) Stack are popped into Index Register IY. The Stack Pointer (SP) register pair holds the 16-bit address of the current "top" of the Stack. This instruction first loads into the low order portion of IY the byte at the memory location corresponding to the contents of SP; then SP is incremented and the contents of the corresponding adjacent memory location are loaded into the high order portion of IY. The SP is now incremented again.

M cycles: 4 T states: 14(4,4,3,3) 4 MHz E.T.: 3.50

Condition Bits Affected: None

Example:

If the Stack Pointer contains 1000H, memory location 1000H contains 55H, and location 1001H contains 33H, the instruction

POP IY

will result in Index Register IY containing 3355H, and the Stack Pointer containing 1002H. Register pair IY contains the number which used to be on the top of the stack.

Before:

Register IY	Address	Stack
24F9	1000	55
	1001	33
	1002	A4
	1003	62

Stack Pointer

1000

After: POP IY

Register IY	Address	Stack
3355	1002	A4
	1003	62

Stack Pointer

1002

Exchange, Block Transfer and Search Group

EX DE,HL

EXchange

Operation: DE ⇔ HL

Format:

Mnemonic: EX **Operands:** DE, HL

Object Code:

1	1	1	0	1	0	1	1

EB

Description:

The two-byte contents of register pairs DE and HL are exchanged.

M cycles: 1 T states: 4 4 MHz E.T.: 1.00

Condition Bits Affected: None

Example:

If the content of register pair DE is the number 2822H, and the content of the register pair HL is number 499AH, after the instruction

EX DE,HL

the content of register pair DE will be 499AH and the content of register pair HL will be 2822H.

EX AF,AF′

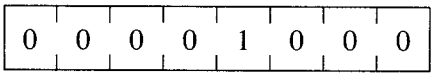
EXchange

Operation: AF ⇔ AF′

Format:

Mnemonic: EX **Operands:** AF, AF′

Object Code:



08

Description:

The two-byte contents of the register pairs AF and AF′ are exchanged. (Note: register pair AF′ consists of registers A′ and F′.)

M cycles: 1 T states: 4 4 MHz E.T.: 1.00

Condition Bits Affected: None

Example:

If the content of register pair AF is number 9900H, and the content of register pair AF′ is number 5944H, after the instruction

EX AF,AF′

the contents of AF will be 5944H, and the contents of AF will be 9900H.

EXX

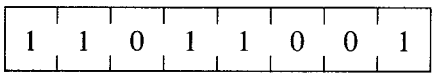
EXchange

Operation: (BC) ⇔ (BC′), (DE) ⇔ (DE′), (HL) ⇔ (HL′)

Format:

Mnemonic: EXX **Operands:**

Object Code:



D9

Description:

Each two-byte value in register pairs BC, DE, and HL is exchanged with the two-byte value in BC′, DE′, and HL′, respectively.

M cycles: 1 T states: 4 4 MHz E.T.: 1.00

Condition Bits Affected: None

Example 1:

If the contents of register pairs BC, DE, and HL are the numbers 445AH, 3DA2H, and 8859H, respectively, and the contents of register pairs BC′, DE′, and HL′ are 0988H, 9300H, and 00E7H, respectively, after the instruction

EXX

the contents of the register pairs will be as follows: BC: 0988H; DE: 9300H; HL: 00E7H; BC′: 445AH; DE′: 3DA2H; and HL′: 8859H.

Example 2:

If the contents of the registers are as shown:

BC : 1111H
DE : 2222H
HL : 3333H
BC′ : 4444H
DE′ : 5555H
HL′ : 6666H

Then after an EXX instruction the registers will contain:

BC : 4444H
DE : 5555H
HL : 6666H
BC′ : 1111H
DE′ : 2222H
HL′ : 3333H

EX (SP), HL

EXchange

Operation: H ⇳ (SP + 1), L ⇳ (SP)

Format:

Mnemonic: EX **Operands:** (SP),HL

Object Code:

1	1	1	0	0	0	1	1

E3

Description:

The low order byte contained in register pair HL is exchanged with the contents of the memory address specified by the contents of register pair SP (Stack Pointer), and the high order byte of HL is exchanged with the next highest memory address (SP + 1).

M cycles: 5 T states: 19(4,3,4,3,5) 4 MHz E.T.: 4.75

Condition Bits Affected: None

Example:

If the HL register pair contains 7012H, the SP register pair contains 8856H, the memory location 8856H contains the byte 11H, and the memory location 8857H contains the byte 22H, then the instruction

EX (SP),HL

will result in the HL register pair containing number 2211H, memory location 8856H containing the byte 12H, the memory location 8857H containing the byte 70H and the Stack Pointer containing 8856H.

Before:

Register HL	Address	Stack
7012	8856	11
	8857	22
	8858	

Stack Pointer

8856

After:

Register HL	Address	Stack
2211	8856	12
	8857	70
	8858	

Stack Pointer

8856

EX (SP),IX

EXchange

Operation: $IX_H \Leftrightarrow (SP + 1)$, $IX_L \Leftrightarrow (SP)$

Format:

Mnemonic: EX **Operands:** (SP), IX

Object Code:

1	1	0	1	1	1	0	1

DD

1	1	1	0	0	0	1	1

E3

Description:

The low order byte in Index Register IX is exchanged with the contents of the memory address specified by the contents of register pair SP (Stack Pointer), and the high order byte of IX is exchanged with the next highest memory address (SP + 1).

Condition Bits Affected: None

Example:

If the Index Register IX contains 3988H, the SP register pair contains 0100H, the memory location 0100H contains the byte 90H, and memory location 0101H contains byte 48H, then the instruction

EX (SP),IX

will result in the IX register pair containing number 4890H, memory location 0100H containing 88H, memory location 0101H containing 39H and the Stack Pointer containing 0100H.

Before:

Register IX	Address	Stack
3988	0100	90
	0101	48

Stack Pointer

0100

After:

Register IX	Address	Stack
4890	0100	88
	0101	39

Stack Pointer

0100

EX (SP),IY

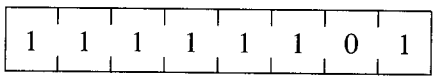
EXchange

Operation: $IY_H \Leftrightarrow (SP + 1)$, $IY_L \Leftrightarrow (SP)$

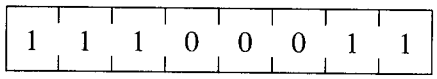
Format:

Mnemonic: EX **Operands:** (SP), IY

Object Code:



FD



E3

Description:

The low order byte in Index Register IY is exchanged with the contents of the memory address specified by the contents of register pair SP (Stack Pointer), and the high order byte of IY is exchanged with the next highest memory address (SP + 1).

M cycles: 6 T states: 23(4,4,3,4,3,5) 4 MHz E.T.: 5.75

Condition Bits Affected: None

Example:

If the Index Register IY contains 3988H, the SP register pair contains 0100H, the memory location 0100H contains the byte 90H, and memory location 0101H contains byte 48H, then the instruction

EX (SP),IY

will result in the IY register pair containing number 4890H, memory location 0100H containing 88H, memory location 0101H containing 39H, and the Stack Pointer containing 0100H.

Before:

Register IY	Address	Stack
3988	0100	90
	0101	48

Stack Pointer

0100

After:

Register IY	Address	Stack
4890	0100	88
	0101	39

Stack Pointer

0100

LDI

LoaD & Increment

Operation: (DE) ◊ (HL), DE ◊ DE + 1, HL ◊ HL + 1, BC ◊ BC − 1

Format:

Mnemonic: LDI **Operands:**

Object Code:

1	1	1	0	1	1	0	1	ED

1	0	1	0	0	0	0	0	AO

Description:

A byte of data is transferred from the memory location addressed by the contents of the HL register pair to the memory location addressed by the contents of the DE register pair. Then both these register pairs are incremented and the BC (Byte Counter) register pair is decremented.

M cycles: 4 T states: 16(4,4,3,5) 4 MHz E.T.: 4.00

Condition Bits Affected:

S: Not affected
Z: Not affected
H: Reset
P/V: Set if BC − 1 ≠ 0; reset otherwise
N: Reset
C: Not affected

Example 1:

If the HL register pair contains 1111H, memory location 1111H contains the byte 88H, the DE register pair contains 2222H, the memory location 2222H contains byte 66H, and the BC register pair contains 7H, then the instruction

LDI

will result in the following contents in register pairs and memory addresses:

HL : 1112H
(1111H) : 88H
DE : 2223H
(2222H) : 88H
BC : 6H

and the condition Bits will be:

		0	1	0	
S	Z	H	P/V	N	C

Example 2:

If the contents of registers and memory are as shown:

HL : 7C00H
(7C00) : FFH
DE : 3C00H
(3C00) : 00H
BC : 1H

Then after an LDI instruction the registers and memory will contain the following:

HL : 7C01H
(7C00) : FFH
DE : 3C01H
(3C00) : FFH
BC 0H

and the condition bits will be:

		0	0	0	
S	Z	H	P/V	N	C

Example 3:

The following program will move 80 consecutive bytes from BUF1 to BUF2:

```
LD      HL, BUF1
LD      DE, BUF2
LD      BC, 80
LOOP    LDI
JP      NZ, LOOP
```

LDIR

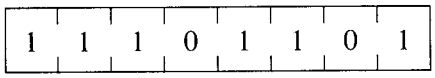
LoaD Increment & Repeat

Operation: (DE) ◊ (HL), DE ◊ DE + 1, HL ◊ HL + 1, BC ◊ BC − 1

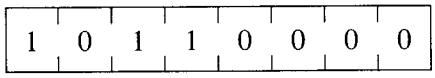
Format:

Mnemonic: LDIR **Operands:**

Object Code:



ED



B0

Description:

This two-byte instruction transfers a byte of data from the memory location addressed by the contents of the HL register pair to the memory location addressed by the DE register pair. Then both these register pairs are incremented and the BC (Byte Counter) register pair is decremented. If decrementing causes the BC to go to zero, the instruction is terminated. If BC is not zero the program counter (PC) is decremented by 2 and the instruction is repeated. Note that if BC is set to zero prior to instruction execution, the instruction will loop through 64K bytes. Also, interrupts will be recognized after each data transfer.

For BC ≠ 0:

M cycles: 5 T states: 21(4,4,3,5,5) 4 MHz E.T.: 5.25

For BC = 0:

M cycles: 4 T states: 16(4,4,3,5) 4 MHz E.T.: 4.00

Condition Bits Affected:

S: Not affected
Z: Not affected
H: Reset
P/V: Reset
N: Reset
C: Not affected

Example:

If the HL register pair contains 1111H, the DE register pair contains 2222H, the BC register pair contains 0003H, and memory locations have these contents:

(1111H) : 88H (2222H) : 66H
(1112H) : 36H (2223H) : 59H
(1113H) : A5H (2224H) : C5H

then after the execution of

LDIR

the contents of register pairs and memory locations will be:

HL	:	1114H			
DE	:	2225H			
BC	:	0000H			
(1111H)	:	88H	(2222H)	:	88H
(1112H)	:	36H	(2223H)	:	36H
(1113H)	:	A5H	(2224H)	:	A5H

and the H, P/V, and N flags are all zero.

LDD

LoaD Decrement

Operation: (DE) ◊ (HL), DE ◊ DE − 1, HL ◊ HL − 1, BC ◊ BC − 1

Format:

Mnemonic: LDD **Operands:**

Object Code:

1	1	1	0	1	1	0	1	ED

1	0	1	0	1	0	0	0	A8

Description:

This two-byte instruction transfers a byte of data from the memory location addressed by the contents of the HL register pair to the memory location addressed by the contents of the DE register pair. Then both of these register pairs, including the BC (Byte Counter) register pair, are decremented.

M cycles: 4 T states: 16(4,4,3,5) 4 MHz E.T.: 4.00

Condition Bits Affected:

S:	Not affected
Z:	Not affected
H:	Reset
P/V:	Set if BC − 1 ≠ 0; reset otherwise
N:	Reset
C:	Not affected

Example 1:

If the HL register pair contains 1111H, memory location 1111H contains the byte 88H, the DE register pair contains 2222H, memory location 2222H contains byte 66H, and the BC register pair contains 7H, then the instruction

LDD

will result in the following contents in register pairs and memory addresses:

HL : 1110H
(1111H) : 88H
DE : 2221H
(2222H) : 88H
BC : 6H

and the condition bits will be:

		0	1	0	
S	Z	H	P/V	N	C

Example 2:

If the contents of registers and memory are as shown:

HL : 7CFFH
(7CFF) : 3CH
DE : 3CFFH
(3CFF) : 00H
BC : 1H

Then after a LDD instruction the registers and memory will contain the following:

HL : 7CFEH
(7CFF) : 3CH
DE : 3CFEH
(3CFF) : 3CH
BC : 0H

and the condition bits will be:

		0	0	0	
S	Z	H	P/V	N	C

LDDR

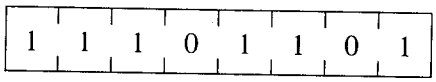
LoaD Decrement & Repeat

Operation: (DE) ◊ (HL), DE ◊ DE – 1, HL ◊ HL – 1, BC ◊ BC – 1

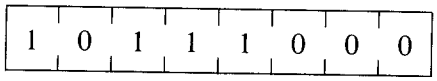
Format:

Mnemonic: LDDR **Operands:**

Object Code:



ED



B8

Description:

This two-byte instruction transfers a byte of data from the memory location addressed by the contents of the HL register pair to the memory location addressed by the contents of the DE register pair. Then both of these registers as well as the BC (Byte Counter) are decremented. If decrementing causes the BC to go to zero, the instruction is terminated. If BC is not zero, the program counter (PC) is decremented by 2 and the instruction is repeated. Note that if BC is set to zero prior to instruction execution, the instruction will loop through 64K bytes. Also, interrupts will be recognized after each data transfer.

For BC≠0:

M cycles: 5 T states: 21(4,4,3,5,5) 4 MHz E.T.: 5.25

For BC=0:

M cycles: 4 T states: 16(4,4,3,5) 4 MHz E.T.: 4.00

Condition Bits Affected:

S: Not affected
Z: Not affected
H: Reset
P/V: Reset
N: Reset
C: Not affected

Example:

If the HL register pair contains 1114H, the DE register pair contains 2225H, the BC register pair contains 0003H, and memory locations have these contents:

(1114H)	:	A5H	(2225H)	:	C5H
(1113H)	:	36H	(2224H)	:	59H
(1112H)	:	88H	(2223H)	:	66H

then after the execution of

LDDR

the contents of register pairs and memory locations will be:

HL	:	1111H			
DE	:	2222H			
BC	:	0000H			
(1114H)	:	A5H	(2225H)	:	A5H
(1113H)	:	36H	(2224H)	:	36H
(1112H)	:	88H	(2223H)	:	88H

and the H, P/V, and N flags are all zero.

CPI

ComPare & Increment

Operation: A – (HL), HL◊HL + 1, BC◊BC – 1

Format:

Mnemonic: CPI **Operands:**

Object Code:

1	1	1	0	1	1	0	1	ED

1	0	1	0	0	0	0	1	A1

Description:

The contents of the memory location addressed by the HL register pair is compared with the contents of the Accumulator. In case of a true compare, the Z condition bit is set. Then HL is incremented and the Byte Counter (register pair BC) is decremented.

M cycles: 4 T states: 16(4,4,3,5) 4 MHz E.T.: 4.00

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if A = (HL); reset otherwise
H:	Set if borrow from Bit 4; reset otherwise
P/V:	Reset if BC becomes 0; set otherwise
N:	Set
C:	Not affected

Example:

If the HL register pair contains 1111H, memory location 1111H contains 3BH, the Accumulator contains 3BH, and the Byte Counter contains 0001H, then after the execution of

CPI

the Byte Counter will contain 0000H, the HL register pair will contain 1112H, the Z flag in the F register will be set, and the P/V flag in the F register will be reset. There will be no effect on the contents of the Accumulator or address 1111H.

If the contents of memory and registers are as shown

HL : 8A00H
(8A00H) : 6DH
A : 75H
BC : 5H

Then during the execution of a CPI instruction the Arithmetic and Logic Unit will do the following subtraction:

Borrow needed here
⇩

```
  75H = 0111  0101
- 6DH = 0110  1101
-------------------
   8H = 0000  1000
```

After CPI is executed registers and memory will contain the following:

HL : 8A01H
(8A00H) : 6DH
A : 75H
BC : 4H

and the condition bits would be:

0	0	1	1	1	1
S	Z	H	P/V	N	C
⇕	⇕	⇕	⇕	⇕	⇕

result positive
match not found
borrow from bit 4

not affected
always set
BC not zero

Example 3:

The following program is used to verify that the contents of two 80-byte buffers are identical. Each time a mismatch is found the program calls a subroutine called ERROR.

```
STRT    LD      HL, BUF1
        LD      DE, BUF2
        LD      BC, 80
LOOP    LD      A, (DE)
        CPI
        CALL    NZ, ERROR
        INC     DE
        JR      PO, LOOP
END
```

CPIR

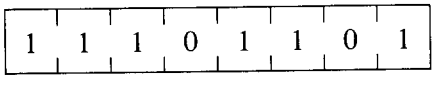
ComPare Increment & Repeat

Operation: $A - (HL), HL \Leftarrow HL + 1, BC \Leftarrow BC - 1$

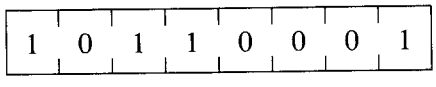
Format:

Mnemonic: CPIR **Operands:**

Object Code:



ED



B1

Description:

The contents of the memory location addressed by the HL register pair is compared with the contents of the Accumulator. In case of a true compare, the Z condition bit is set. The HL is incremented and the Byte Counter (register pair BC) is decremented. If decrementing causes the BC to go to zero or if A = (HL), the instruction is terminated. If BC is not zero and A ≠ (HL), the program counter is decremented by 2 and the instruction is repeated. Note that if BC is set to zero before the execution, the instruction will loop through 64K bytes, if no match is found. Also, interrupts will be recognized after each data comparison.

For BC ≠ 0 and A ≠ (HL):

M cycles: 5 T states: 21(4,4,3,5,5) 4 MHz E.T.: 5.25

For BC ≠ 0 or A = (HL):

M cycles: 4 T states: 16(4,4,3,5) 4 MHz E.T.: 4.00

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if A = (HL); reset otherwise
H:	Set if borrow from Bit 4; reset otherwise
P/V:	Reset if BC becomes 0; set otherwise
N:	Set
C:	Not affected

Example:

If the HL register pair contains 1111H, the Accumulator (Register A) contains F3H, the Byte Counter contains 0007H, and memory locations have these contents:

(1111H) : 52H
(1112H) : 00H
(1113H) : F3H

then after the execution of

CPIR

the contents of register pair HL will be 1114H, and the contents of the Byte Counter will be 0004H. Since BC ≠ 0, the P/V flag is still set. This means that it did not search through the whole block before the instruction stopped. Since a match was found, the Z flag is set.

The following program uses the CPIR instruction to count the number of nulls (00H) found in an 80-byte buffer. The count is kept in register E.

```
STRT    LD      BC, 80
        LD      HL, BUFF
        LD      A, 0
        LD      E, 0
LOOP    CPIR
        JR      NZ, FOO
        INC     E
FOO     JP      PE, LOOP
END
```

CPD

ComPare & Decrement

Operation: A – (HL), HL ◊ HL – 1, BC ◊ BC – 1

Format:

Mnemonic: CPD **Operands:**

Object Code:

1	1	1	0	1	1	0	1	ED

1	0	1	0	1	0	0	1	A9

Description:

The contents of the memory location addressed by the HL register pair is compared with the contents of the Accumulator. In case of a true compare, the Z condition bit is set. The HL and the Byte Counter (register pair BC) are decremented.

M cycles: 4 T states: 16(4,4,3,5) 4 MHz E.T.: 4.00

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if A = (HL); reset otherwise
H: Set if borrow from Bit 4; reset otherwise
P/V: Reset if BC becomes zero; set otherwise
N: Set
C: Not affected

Example:

If the HL register pair contains 1111H, memory location 1111H contains 3BH, the Accumulator contains 3BH, and the Byte Counter contains 0001H, then after the execution of

CPD

the Byte Counter will contain 0000H, the HL register pair will contain 1110H, the Z flag in the F register will be set and the P/V flag in the F register will be reset. There will be no effect on the contents of the Accumulator or address 1111H.

Since the CPD instruction decrements HL, it is used to search through memory from high to low addresses. Otherwise it is similar to the CPI instruction.

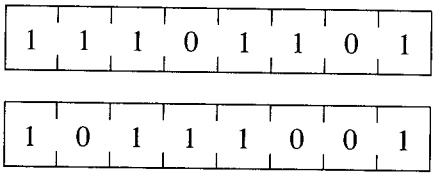
CPDR ComPare Decrement & Repeat

Operation: A – (HL), HL ◊ HL – 1, BC ◊ BC – 1

Format:

Mnemonic: CPDR **Operands:**

Object Code:



Description:

The contents of the memory location addressed by the HL register pair is compared with the contents of the Accumulator. In case of a true compare, the Z condition bit is set. The HL and BC (Byte Counter) register pairs are decremented. If decrementing causes the BC to go to zero or if A = (HL), the instruction is terminated. If BC is not zero and A ≠ (HL), the program counter is decremented by 2 and the instruction is repeated. Note that if BC is set to zero prior to instruction execution, the instruction will loop through 64K bytes, if no match is found. Also, interrupts will be recognized after each data comparison.

For BC ≠ 0 and A ≠ (HL):

M cycles: 5 T states: 21(4,4,3,5,5) 4 MHz E.T.: 5.25

For BC = 0 or A = (HL):

M cycles: 4 T states: 16(4,4,3,5) 4 MHz E.T.: 4.00

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if A = (HL), reset otherwise
H: Set if borrow from Bit 4; reset otherwise
P/V: Reset if BC becomes zero; set otherwise
N: Set
C: Not affected

Example:

If the HL register pair contains 1118H, the Accumulator contains F3H, the Byte Counter contains 0003H, and memory locations have these contents:

(1118H) : 52H
(1117H) : 00H
(1116H) : F3H

then after the execution of

CPDR

the contents of register pair HL will be 1115H, the contents of the Byte Counter will be 0000H, the P/V flag in the F register will be reset, and the Z flag in the F register will be set.

8 Bit Arithmetic and Logical Group

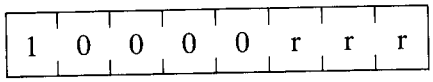
ADD A,r

Operation: A ◊ A + r

Format:

Mnemonic: ADD **Operands:** A, r

Object Code:



Description:

The contents of register r are added to the contents of the Accumulator, and the result is stored in the Accumulator. The symbol r identifies the registers A, B, C, D, E, H or L assembled as follows in the object code:

Register		r
A	=	111
B	=	000
C	=	001
D	=	010
E	=	011
H	=	100
L	=	101

M cycles: 1 T states: 4 4 MHz E.T.: 1.00

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Set if carry from Bit 3; reset otherwise
P/V: Set if overflow; reset otherwise
N: Reset
C: Set if carry from Bit 7; reset otherwise

Example:

If the contents of the Accumulator are 44H, and the contents of register C are 11H, after the execution of

ADD A,C

the contents of the Accumulator will be 55H. See Appendix K for more details of condition bits affected.

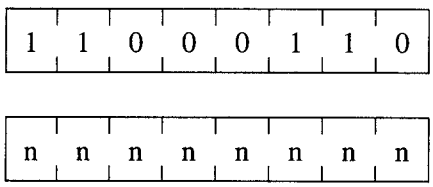
ADD A,n

Operation: A ◊ A + n

Format:

Mnemonic: ADD **Operands:** A, n

Object Code:



Description:

The integer n is added to the contents of the Accumulator and the results are stored in the Accumulator.

M cycles: 2 T states: 7(4,3) 4 MHz E.T.: 1.75

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Set if carry from Bit 3; reset otherwise
P/V: Set if overflow; reset otherwise
N: Reset
C: Set if carry from Bit 7; reset otherwise

Example:

If the contents of the Accumulator are 23H, after the execution of

ADD A,33H

the contents of the Accumulator will be 56H.

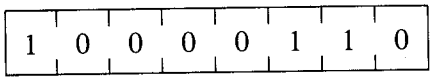
ADD A,(HL)

Operation: A◊A+(HL)

Format:

Mnemonic: ADD **Operands:** A, (HL)

Object Code:



86

Description:

The byte at the memory address specified by the contents of the HL register pair is added to the contents of the Accumulator and the result is stored in the Accumulator.

M cycles: 2 T states: 7(4,3) 4 MHz E.T.: 1.75

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Set if carry from Bit 3; reset otherwise
P/V: Set if overflow; reset otherwise
N: Reset
C: Set if carry from Bit 7; reset otherwise

Example:

If the contents of the Accumulator are A0H, and the content of the register pair HL is 2323H, and memory location 2323H contains byte 08H, after the execution of

ADD A,(HL)

the Accumulator will contain A8H.

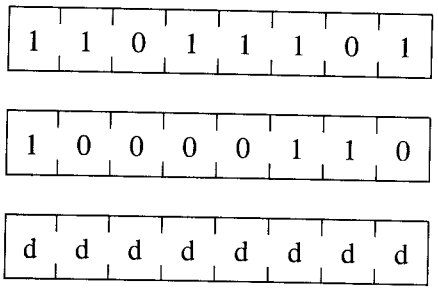
ADD A,(IX+d)

Operation: A◊A+(IX+d)

Format:

Mnemonic: ADD **Operands:** A, (IX+d)

Object Code:



Description:

The contents of the Index Register (register pair IX) is added to a two's complement displacement d to point to an address in memory. The contents of this address is then added to the contents of the Accumulator and the result is stored in the Accumulator.

M cycles: 5 T states: 19(4,4,3,5,3) 4 MHz E.T.: 4.75

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Set if carry from Bit 3; reset otherwise
P/V: Set if overflow; reset otherwise
N: Reset
C: Set if carry from Bit 7; reset otherwise

Example:

If the Accumulator contents are 11H, the Index Register IX contains 1000H, and if the content of memory location 1005H is 22H, after the execution of

ADD A,(IX+5H)

the contents of the Accumulator will be 33H.

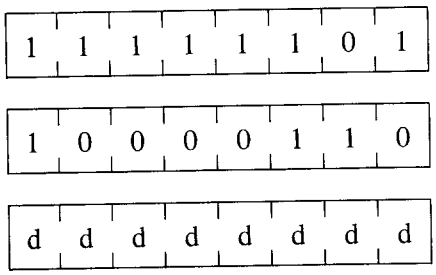
ADD A,(IY+d)

Operation: A◊A+(IY+d)

Format:

Mnemonic: ADD **Operands:** A, (IY+d)

Object Code:



Description:

The contents of the Index Register (register pair IY) is added to the displacement d to point to an address in memory. The contents of this address is then added to the contents of the Accumulator and the result is stored in the Accumulator.

M cycles: 5 T states: 19(4,4,3,5,3) 4 MHz E.T.: 4.75

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Set if carry from Bit 3; reset otherwise
P/V: Set if overflow; reset otherwise
N: Reset
C: Set if carry from Bit 7; reset otherwise

Example:

If the Accumulator contents are 11H, the Index Register pair IY contains 1000H, and if the content of memory location 1005H is 22H, after the execution of

ADD A,(IY+5H)

the contents of the Accumulator will be 33H.

ADC A,S

ADd with Carry

Operation: A ◊ A+s+CY

Format:

Mnemonic: ADC **Operands:** A, s

The s operand is any of r, n, (HL), (IX+d) or (IY+d) as defined for the analogous ADD instruction. These various possible opcode-operand combinations are assembled as follows in the object code:

Object Code:

ADC A, r	1 0 0 0 1 r r r	
ADC A, n	1 1 0 0 1 1 1 0	CE
	n n n n n n n n	
ADC A, (HL)	1 0 0 0 1 1 1 0	8E
ADC A, (IX+d)	1 1 0 1 1 1 0 1	DD
	1 0 0 0 1 1 1 0	8E
	d d d d d d d d	
ADC A, (IY+d)	1 1 1 1 1 1 0 1	FD
	1 0 0 0 1 1 1 0	8E
	d d d d d d d d	

r identifies registers A, B, C, D, E, H, or L assembled as follows in the object code field above:

Register		r
A	=	111
B	=	000
C	=	001
D	=	010
E	=	011
H	=	100
L	=	101

Description:

The s operand, along with the Carry Flag (''C'' in the F register) is added to the contents of the Accumulator, and the result is stored in the Accumulator.

Instruction	M Cycles	T States	4 MHz E.T. in μs
ADC A, r	1	4	1.00
ADC A, n	2	7(4,3)	1.75
ADC A, (HL)	2	7(4,3)	1.75
ADC A, (IX+d)	5	19(4,4,3,5,3)	4.75
ADC A, (IY+d)	5	19(4,4,3,5,3)	4.75

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Set if carry from Bit 3; reset otherwise
P/V: Set if overflow; reset otherwise
N: Reset
C: Set if carry from Bit 7; reset otherwise

Example 1:

If the Carry Flag is set, the Accumulator contains 16H, the HL register pair contains 6666H, and address 6666H contains 10H, after the execution of

ADC A, (HL)

the Accumulator will contain 27H.

Example 2:

If the Carry Flag is set, the Accumulator contains 30H, and register C contains 05H, then after the execution of

ADC A, C

the Accumulator will contain 36H.

SUB s

SUBtract

Operation: A ◊ A − s

Format:

Mnemonic: SUB **Operands:** s

The s operand is any of r, n, (HL), (IX+d) or (IY+d) as defined for the analogous ADD instruction. These various possible opcode-operand combinations are assembled as follows in the object code:

Object Code:

SUB r	1 0 0 1 0 r r r	
SUB n	1 1 0 1 0 1 1 0	D6
	n n n n n n n n	
SUB (HL)	1 0 0 1 0 1 1 0	96
SUB (IX+d)	1 1 0 1 1 1 0 1	DD
	1 0 0 1 0 1 1 0	96
	d d d d d d d d	
SUB (IY+d)	1 1 1 1 1 1 0 1	FD
	1 0 0 1 0 1 1 0	96
	d d d d d d d d	

r identifies registers A, B, C, D, E, H or L assembled as follows in the object code field above:

Register		r
A	=	111
B	=	000
C	=	001
D	=	010
E	=	011
H	=	100
L	=	101

Description:

The s operand is subtracted from the contents of the Accumulator, and the result is stored in the Accumulator.

Instruction	M Cycles	T States	4 MHz E.T. in μs
SUB r	1	4	1.00
SUB n	2	7(4,3)	1.75
SUB (HL)	2	7(4,3)	1.75
SUB (IX+d)	5	19(4,4,3,5,3)	4.75
SUB (IY+d)	5	19(4,4,3,5,3)	4.75

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Set if borrow from Bit 4; reset otherwise
P/V: Set if overflow; reset otherwise
N: Set
C: Set if borrow; reset otherwise

Example:

If the Accumulator contains 29H and register D contains 11H, after the execution of

SUB D

the Accumulator will contain 18H.

SBC A,s

SuBtract with borrow (Carry)

Operation: A ◊ A − s − CY

Format:

Mnemonic: SBC **Operands:** A, s

The s operand is any of r, n, (HL), (IX+d) or (IY+d) as defined for the analogous ADD instructions. These various possible opcode-operand combinations are assembled as follows in the object code:

Object Code:

SBC A, r

1	0	0	1	1	r	r	r

SBC A, n

1	1	0	1	1	1	1	0

DE

n	n	n	n	n	n	n	n

SBC A, (HL)	1 0 0 1 1 1 1 0	9E
SBC A, (IX+d)	1 1 0 1 1 1 0 1	DD
	1 0 0 1 1 1 1 0	9E
	d d d d d d d d	
SBC A,(IY + d)	1 1 1 1 1 1 0 1	FD
	1 0 0 1 1 1 1 0	9E
	d d d d d d d d	

r identifies registers A, B, C, D, E, H, or L assembled as follows in the object code field above:

Register		r
A	=	111
B	=	000
C	=	001
D	=	010
E	=	011
H	=	100
L	=	101

Description:

The s operand, along with the Carry Flag (''C'' in the F register) is subtracted from the contents of the Accumulator, and the result is stored in the Accumulator.

Instruction	M Cycles	T States	4 MHz E.T. in μs
SBC A, r	1	4	1.00
SBC A, n	2	7(4,3)	1.75
SBC A, (HL)	2	7(4,3)	1.75
SBC A, (IX+d)	5	19(4,4,3,5,3)	4.75
SBC A, (IY+d)	5	19(4,4,3,5,3)	4.75

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Set if borrow from Bit 4; reset otherwise
P/V: Set if overflow; reset otherwise
N: Set
C: Set if borrow; reset otherwise

Example 1:

If the Carry Flag is set, the Accumulator contains 16H, the HL register pair contains 3433H, and address 3433H contains 05H, after the execution of

SBC A,(HL)

the Accumulator will contain 10H.

Example 2:

If the Carry Flag is set, the Accumulator contains 21H and register B contains 0, then after the execution of

SBC A,B

the Accumulator contains 20H.

AND s

Operation: A ◊ A ⌓ s

Format:

Mnemonic: AND **Operands:** s

The s operand is any of r, n, (HL), (IX+d) or (IY+d), as defined for the analogous ADD instructions. These various possible opcode-operand combinations are assembled as follows in the object code:

Object Code:

AND r	1 0 1 0 0 r r r	
AND n	1 1 1 0 0 1 1 0	E6
	n n n n n n n n	
AND (HL)	1 0 1 0 0 1 1 0	A6
AND (IX+d)	1 1 0 1 1 1 0 1	DD
	1 0 1 0 0 1 1 0	A6
	d d d d d d d d	
AND (IY+d)	1 1 1 1 1 1 0 1	FD
	1 0 1 0 0 1 1 0	A6
	d d d d d d d d	

r identifies register A, B, C, D, E, H or L assembled as follows in the object code field above:

Register		r
A	=	111
B	=	000
C	=	001
D	=	010
E	=	011
H	=	100
L	=	101

Description:

A logical AND operation, Bit by Bit, is performed between the byte specified by the s operand and the byte contained in the Accumulator; the result is stored in the Accumulator.

Instruction	M Cycles	T States	4 MHz E.T. in μs
AND r	1	4	1.00
AND n	2	7(4,3)	1.75
AND (HL)	2	7(4,3)	1.75
AND (IX+d)	5	19(4,4,3,5,3)	4.75
AND (IX+d)	5	19(4,4,3,5,3)	4.75

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Set
P/V: Set if parity even; reset otherwise
N: Reset
C: Reset

Table of AND Values:

IF A	IF B	Then A (After)
0	0	0
0	1	0
1	0	0
1	1	1

Example:

If the B register contains 7BH (01111011) and the Accumulator contains C3H (11000011), after the execution of

AND B

the Accumulator will contain 43H (01000011).

OR s

Operation: A ◊ A ⇦ s

Format:

Mnemonic: OR **Operands:** s

The s operand is any of r, n, (HL), (IX+d), or (IY+d), as defined for the analogous ADD instructions. These various possible opcode-operand combinations are assembled as follows in the object code:

Object Code:

OR r	1 0 1 1 0 r r r	
OR n	1 1 1 1 0 1 1 0	F6
	n n n n n n n n	
OR (HL)	1 0 1 1 0 1 1 0	B6
OR (IX+d)	1 1 0 1 1 1 0 1	DD
	1 0 1 1 0 1 1 0	B6
	d d d d d d d d	
OR (IY+d)	1 1 1 1 1 1 0 1	FD
	1 0 1 1 0 1 1 0	B6
	d d d d d d d d	

r identifies register A, B, C, D, E, H or L assembled as follows in the object code field above:

Register		r
A	=	111
B	=	000
C	=	001
D	=	010
E	=	011
H	=	100
L	=	101

Description:

A logical OR operation, Bit by Bit, is performed between the byte specified by the s operand and the byte contained in the Accumulator; the result is stored in the Accumulator.

Instruction	M Cycles	T States	4 MHz E.T. in μs
OR r	1	4	1.00
OR n	2	7(4,3)	1.75
OR (HL)	2	7(4,3)	1.75
OR (IX+d)	5	19(4,4,3,5,3)	4.75
OR (IY+d)	5	19(4,4,3,5,3)	4.75

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Reset
P/V: Set if parity even; reset otherwise
N: Reset
C: Reset

Table of OR Values:

IF A	IF B	Then A (After)
0	0	0
0	1	1
1	0	1
1	1	1

Example:

If the H register contains 48H (01001000) and the Accumulator contains 12H (00010010), after the execution of

OR H

the Accumulator will contain 5AH (01011010).

XOR s

eXclusive OR

Operation: A ◊ A⊕s

Format:

Mnemonic: XOR **Operands:** s

The s operand is any of r, n, (HL), (IX+d) or (IY+d), as defined for the analogous ADD instructions. These various possible opcode-operand combinations are assembled as follows in the object code:

Object Code:

XOR r	1 0 1 0 1 r r r	
XOR n	1 1 1 0 1 1 1 0	EE
	n n n n n n n n	
XOR (HL)	1 0 1 0 1 1 1 0	AE
XOR (IX+d)	1 1 0 1 1 1 0 1	DD
	1 0 1 0 1 1 1 0	AE
	d d d d d d d d	
XOR (IY+d)	1 1 1 1 1 1 0 1	FD
	1 0 1 0 1 1 1 0	AE
	d d d d d d d d	

r identifies registers A, B, C, D, E, H or L assembled as follows in the object code field above:

Register		r
A	=	111
B	=	000
C	=	001
D	=	010
E	=	011
H	=	100
L	=	101

Description:

A logical exclusive-OR operation, bit by bit, is performed between the byte specified by the s operand and the byte contained in the Accumulator; the result is stored in the Accumulator.

Instruction	M Cycles	T States	4 MHz E.T. in μs
XOR r	1	4	1.00
XOR n	2	7(4,3)	1.75
XOR (HL)	2	7(4,3)	1.75
XOR (IX+d)	5	19(4,4,3,5,3)	4.75
XOR (IY+d)	5	19(4,4,3,5,3)	4.75

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Reset
P/V: Set if parity even; reset otherwise
N: Reset
C: Reset

Table of XOR Values:

IF A	IF B	Then A (After)
0	0	0
0	1	1
1	0	1
1	1	0

Note: in Table above that any two like numbers will result in zero.

Example 1:

If the Accumulator contains 96H (10010110), after the execution of
XOR 5DH (Note: 5DH = 01011101)
the Accumulator will contain CBH (11001011).

Example 2:

The instruction
XOR A
will zero the Accumulator.

CP s

ComPare

Operation: A − s

Format:

Mnemonic: CP **Operands:** s

The s operand is any of r, n, (HL), (IX+d) or (IY+d), as defined for the analogous ADD instructions. These various possible opcode-operand combinations are assembled as follows in the object code:

Object Code:

CP r	1 0 1 1 1 r r r	
CP n	1 1 1 1 1 1 1 0	FE
	n n n n n n n n	
CP (HL)	1 0 1 1 1 1 1 0	BE
CP (IX+d)	1 1 0 1 1 1 0 1	DD
	1 0 1 1 1 1 1 0	BE
	d d d d d d d d	
CP (IY+d)	1 1 1 1 1 1 0 1	FD
	1 0 1 1 1 1 1 0	BE
	d d d d d d d d	

r identifies register A, B, C, D, E, H or L assembled as follows in the object code field above:

Register		r
A	=	111
B	=	000
C	=	001
D	=	010
E	=	011
H	=	100
L	=	101

Description:

The contents of the s operand are compared with the contents of the Accumulator. If there is a true compare, a flag is set.

Instruction	M Cycles	T States	4 MHz E.T. in μs
CP r	1	4	1.00
CP n	2	7(4,3)	1.75
CP (HL)	2	7(4,3)	1.75
CP (IX+d)	5	19(4,4,3,5,3)	4.75
CP (IY+d)	5	19(4,4,3,5,3)	4.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Set if borrow from Bit 4; reset otherwise
P/V:	Set if overflow; reset otherwise
N:	Set
C:	Set if borrow in Bit 7; reset otherwise

Example 1:

If the Accumulator contains 63H, the HL register pair contains 6000H and memory location 6000H contains 60H, the instruction

CP (HL)

will result in all the flags being reset except N.

Example: 2

If the Accumulator contains 65H and register C also contains 65H, then after the execution of

CP C

the Z flag will be set.

See Appendix E for more details of condition codes affected.

INC r

INCrement

Operation: r ◊ r + 1

Format:

Mnemonic: INC **Operands:** r

Object Code:

0	0	r	r	r	1	0	0

Description:

Register r is incremented. r identifies any of the registers A, B, C, D, E, H or L, assembled as follows in the object code.

Register		r
A	=	111
B	=	000
C	=	001
D	=	010
E	=	011
H	=	100
L	=	101

M cycles: 1 T states: 4 4 MHz E.T.: 1.00

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Set if carry from Bit 3; reset otherwise
P/V: Set if r was 7FH before operation; reset otherwise
N: Reset
C: Not affected

Example:

If the contents of register D are 28H, after the execution of

INC D

the contents of register D will be 29H.

INC (HL)

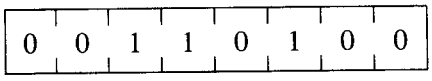
INCrement

Operation: (HL) ◊ (HL) + 1

Format:

Mnemonic: INC **Operands:** (HL)

Object Code:



34

Description:

The byte contained in the address specified by the contents of the HL register pair is incremented.

M cycles: 3 T states: 11(4,4,3) 4 MHz E.T.: 2.75

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Set if carry from Bit 3; reset otherwise
P/V: Set if (HL) was 7FH before operation; reset otherwise
N: Reset
C: Not Affected

Example:

If the contents of the HL register pair are 3434H, and the contents of address 3434H are 82H, after the execution of

INC (HL)

memory location 3434H will contain 83H.

INC (IX+d)

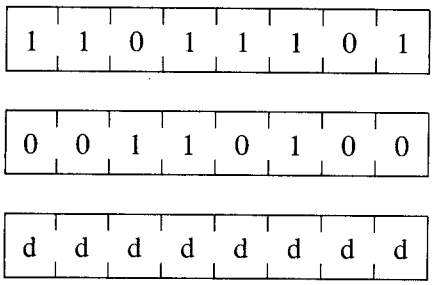
INCrement

Operation: (IX+d) ◊ (IX+d) + 1

Format:

Mnemonic: INC **Operands:** (IX+d)

Object Code:



Description:

The contents of the Index Register IX (register pair IX) are added to a two's complement displacement integer d to point to an address in memory. The contents of this address are then incremented.

M cycles: 6 T states: 23(4,4,3,5,4,3) 4 MHz E.T.: 5.75

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Set if carry from Bit 3; reset otherwise
P/V: Set if (IX+d) was 7FH before operation; reset otherwise
N: Reset
C: Not affected

Example:

If the contents of the Index Register pair IX are 2020H, and the memory location 2030H contains byte 34H, after the execution of

INC (IX+10H)

the contents of memory location 2030H will be 35H.

INC (IY + d)

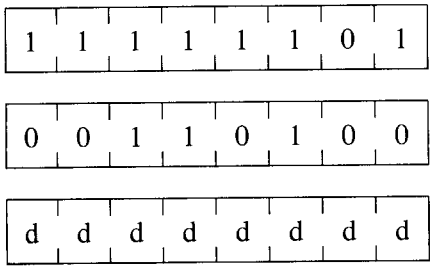
INCrement

Operation: (IY+d) ⇦ (IY+d) + 1

Format:

Mnemonic: INC **Operands:** (IY+d)

Object Code:



Description:

The contents of the Index Register IY (register pair IY) are added to a two's complement displacement integer d to point to an address in memory. The contents of this address are then incremented.

M cycles: 6 T states: 23(4,4,3,5,4,3) 4 MHz E.T.: 5.75

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Set if carry from Bit 3; reset otherwise
P/V: Set if (IY+d) was 7FH before operation; reset otherwise
N: Reset
C: Not Affected

Example:

If the contents of the Index Register pair IY are 2020H, and the memory location 2030H contain byte 34H, after the execution of

INC (IY+10H)

the contents of memory location 2030H will be 35H.

DEC m

DECrement

Operation: m ⇳ m − 1

Format:

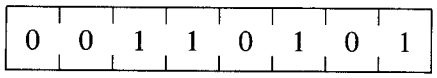
Mnemonic: DEC **Operands:** m

The m operand is any of r, (HL), (IX+d) or (IY+d), as defined for the analogous INC instructions. These various possible opcode-operand combinations are assembled as follows in the object code:

Object Code:

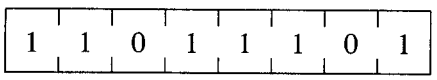
DEC r

DEC (HL)

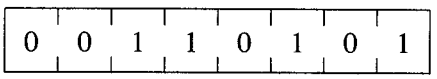


35

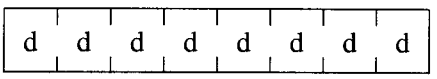
DEC (IX+d)



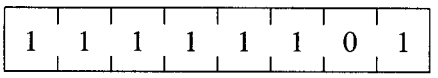
DD



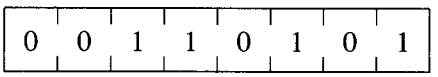
35



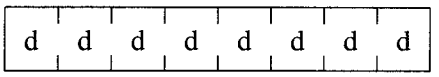
DEC (IY+d)



FD



35



r identifies register A, B, C, D, E, H or L assembled as follows in the object code field above:

Register		**r**
A	=	111
B	=	000
C	=	001
D	=	010
E	=	011
H	=	100
L	=	101

Description:

The byte specified by the m operand is decremented.

Instruction	M Cycles	T States	4 MHz E.T. in μs
DEC r	1	4	1.00
DEC (HL)	3	11(4,4,3)	2.75
DEC (IX+d)	6	23(4,4,3,5,4,3)	5.75
DEC (IY+d)	6	23(4,4,3,5,4,3)	5.75

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Set if borrow from Bit 4; reset otherwise
P/V: Set if m was 80H before operation; reset otherwise
N: Set
C: Not affected

Example:

If the D register contains byte 2AH, after the execution of

DEC D

register D will contain 29H.

General Purpose Arithmetic and CPU Control Groups

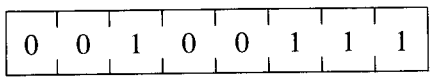
DAA

Operation: Decimal-Adjust Accumulator

Format:

Mnemonic: DAA **Operands:**

Object Code:



Description:

This instruction modifies the results of addition or subtraction so that the results of binary arithmetic are correct for decimal numbers. The Binary Coded Decimal (BCD) code uses the 8-bit accumulator as follows: the eight bits are broken up into two groups of four bits, which represent a two-digit decimal number from 00 to 99. If numbers like this are added with the binary adder in the Z-80, answers larger than 10 may result in each decimal place. The DAA instruction will "adjust" the answer so that each decimal place has a value of 9 or less, and so that the digits have the correct decimal value, though they were added by a binary circuit. The carry and half-carry flags are used in this conversion, as is a circuit that detects digits that are 10 or bigger.

Operation	C Before DAA	HEX Value in Upper Digit (bits 7-4)	H Before DAA	HEX Value in Lower Digit (bits 3-0)	Number Added to Byte	C After DAA
	0	0-9	0	0-9	00	0
	0	0-8	0	A-F	06	0
	0	0-9	1	0-3	06	0
ADD	0	A-F	0	0-9	60	1
ADC	0	9-F	0	A-F	66	1
INC	0	A-F	1	0-3	66	1
	1	0-2	0	0-9	60	1
	1	0-2	0	A-F	66	1
	1	0-3	1	0-3	66	1
SUB	0	0-9	0	0-9	00	0
SBC	0	0-8	1	6-F	FA	0
DEC	1	7-F	0	0-9	A0	1
NEG	1	6-F	1	6-F	9A	1

M cycles: 1 T states: 4 4 MHz E.T.: 1.00

Condition Bits Affected:

S:	Set if most significant bit of Acc. is 1 after operation; reset otherwise
Z:	Set if Acc. is zero after operation; reset otherwise
H:	See instruction
P/V:	Set if Acc. is even parity after operation; reset otherwise
N:	Not affected
C:	See instruction

Example:

If an addition operation is performed between 15 (BCD) and 27 (BCD), simple decimal arithmetic gives this result:

```
  15
+27
----
  42
```

But when the binary representations are added in the Accumulator according to standard binary arithmetic,

```
  0001 0101
+0010 0111
----------
  0011 1100 = 3C
```

the sum is not decimal. The DAA instruction adjusts this result so that the correct BCD representation is obtained:

```
  0011 1100
+0000 0110(adding 06 from table)
---------------
  0100 0010 = 42
```

CPL

ComPLement

Operation: $A \Leftarrow \overline{A}$

Format:

Mnemonic: CPL **Operands:**

Object Code:

0	0	1	0	1	1	1	1

2F

Description:

Contents of the Accumulator (register A) are inverted (one's complement).

M cycles: 1 T states: 4 4 MHz E.T.: 1.00

Condition Bits Affected:

S: Not affected
Z: Not affected
H: Set
P/V: Not affected
N: Set
C: Not affected

Example:

If the contents of the Accumulator are 1011 0100, after the execution of

CPL

the Accumulator contents will be 0100 1011.

NEG

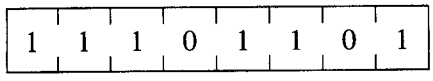
NEGate

Operation: A ⇦ 0 − A

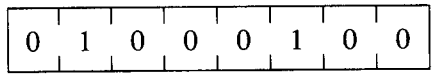
Format:

Mnemonic: NEG **Operands:**

Object Code:



ED



44

Description:

Contents of the Accumulator are negated (two's complement). This is the same as subtracting the contents of the Accumulator from zero. Note that 80H is left unchanged.

M cycles: 2 T states: 8(4,4) 4 MHz E.T.: 2.00

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Set if borrow from Bit 4; reset otherwise
P/V: Set if Acc. was 80H before operation; reset otherwise
N: Set
C: Set if Acc. was not 00H before operation; reset otherwise

Example:

If the contents of the Accumulator are

1	0	0	1	1	0	0	0

after the execution of

NEG

the Accumulator contents will be

0	1	1	0	1	0	0	0

CCF

Complement Carry Flag

Operation: $CY \Leftrightarrow \overline{CY}$

Format:

Mnemonic: CCF **Operands:**

Object Code:

0	0	1	1	1	1	1	1

3F

Description:

The C flag in the F register is inverted.

M cycles: 1 T states: 4 4 MHz E.T.: 1.00

Condition Bits Affected:

S: Not affected
Z: Not affected
H: Previous carry will be copied
P/V: Not affected
N: Reset
C: Set if CY was 0 before operation; reset otherwise

SCF

Set Carry Flag

Operation: CY◊1

Format:

Mnemonic: SCF **Operands:**

Object Code:

0	0	1	1	0	1	1	1

37

Description:

The C flag in the F register is set.

M cycles: 1 T states: 4 4 MHz E.T.: 1.00

Condition Bits Affected:

S: Not affected
Z: Not affected
H: Reset
P/V: Not affected
N: Reset
C: Set

NOP

No OPeration

Operation:

Format:

Mnemonic: NOP **Operands:**

Object Code:

0	0	0	0	0	0	0	0

00

Description:

CPU performs no operation during this machine cycle.

M cycles: 1 T states: 4 4 MHz E.T.: 1.00

Condition Bits Affected: None

HALT

Operation:

Format:

Mnemonic: HALT **Operands:**

Object Code:

0	1	1	1	0	1	1	0

76

Description:

The HALT instruction suspends CPU operation until a subsequent interrupt or reset is received. While in the halt state, the processor will execute NOP's to maintain memory refresh logic.

M cycles: 1 T states: 4 4 MHz E.T.: 1.00

Condition Bits Affected: None

DI

Disable Interrupts

Operation: IFF◊0

Format:

Mnemonic: DI **Operands:**

Object Code:

1	1	1	1	0	0	1	1

F3

Description:

DI disables the maskable interrupt by resetting the interrupt enable flip-flops (IFF1 and IFF2). Note that this instruction disables the maskable interrupt during its execution.

M cycles: 1 T states: 4 4 MHz E.T.: 1.00

Condition Bits Affected: None

Example:

When the CPU executes the instruction

DI

the maskable interrupt is disabled until it is subsequently re-enabled by an EI instruction. The CPU will not respond to an Interrupt Request (INT) signal.

EI

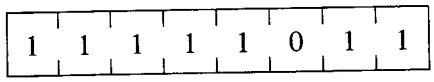
Enable Interrupts

Operation: IFF ◊ 1

Format:

Mnemonic: EI **Operands:**

Object Code:



FB

Description:

EI enables the maskable interrupt by setting the interrupt enable flip-flops (IFF1 and IFF2). Note that this instruction disables the maskable interrupt during its execution.

M cycles: 1 T states: 4 4 MHz E.T.: 1.00

Condition Bits Affected: None

Example:

When the CPU executes instruction

RETI

the maskable interrupt is enabled. The CPU will now respond to an Interrupt Request (INT) signal.

IM Ø

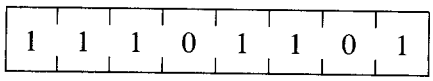
Interrupt Mode Ø

Operation:

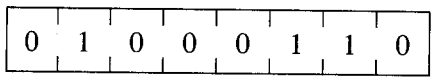
Format:

Mnemonic: IM **Operands:** Ø

Object Code:



ED



46

Description:

The IM Ø instruction sets interrupt mode Ø. In this mode the interrupting device can insert any instruction on the data bus and allow the CPU to execute it. The first byte of a multi-byte instruction is read during interrupt acknowledge cycle. Subsequent bytes are read in by a normal memory read sequence.

M cycles: 2 T states: 8(4,4) 4 MHz E.T.: 2.ØØ

Condition Bits Affected: None

IM 1

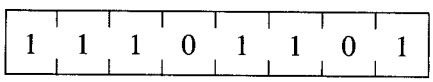
Interrupt Mode 1

Operation:

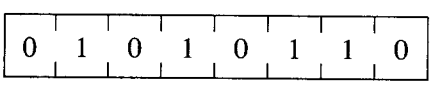
Format:

Mnemonic: IM **Operands:** 1

Object Code:



ED



56

Description:

The IM instruction sets interrupt mode 1. In this mode the processor will respond to an interrupt by executing a restart to location 0038H.

M cycles: 2 T states: 8(4,4) 4 MHz E.T.: 2.00

Condition Bits Affected: None

IM 2

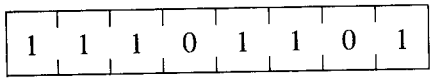
Interrupt Mode 2

Operation:

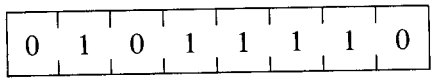
Format:

Mnemonic: IM **Operands:** 2

Object Code:



ED



5E

Description:

The IM 2 instruction sets interrupt mode 2. This mode allows an indirect call to any location in memory. With this mode the CPU forms a 16-bit memory address. The upper eight bits are the contents of the Interrupt Vector Register I and the lower eight bits are supplied by the interrupting device.

M cycles: 2 T states: 8(4,4) 4 MHz E.T.: 2.00

Condition Bits Affected: None

16 Bit Arithmetic Group

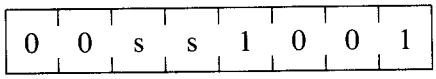
ADD HL,ss

Operation: HL◊HL+ss

Format:

Mnemonic: ADD **Operands:** HL, ss

Object Code:



Description:

The contents of register pair ss (any of register pairs BC, DE, HL or SP) are added to the contents of register pair HL, and the result is stored in HL. Operand ss is specified as follows in the assembled object code.

Register Pair	ss
BC	00
DE	01
HL	10
SP	11

M cycles: 3 T states: 11(4,4,3) 4 MHz E.T.: 2.75

Condition Bits Affected:

S: Not affected
Z: Not affected
H: Set if carry out of Bit 11; reset otherwise
P/V: Not affected
N: Reset
C: Set if carry from Bit 15; reset otherwise

Example:

If register pair HL contains the integer 4242H and register pair DE contains 1111H, after the execution of

ADD HL, DE

the HL register pair will contain 5353H.

ADC HL,ss

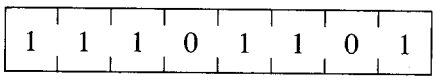
ADd with Carry

Operation: HL◊HL+ss+CY

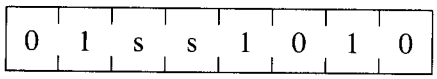
Format:

Mnemonic: ADC **Operands:** HL, ss

Object Code:



ED



Description:

The contents of register pair ss (any of register pairs BC, DE, HL or SP) are added with the Carry Flag (C flag in the F register) to the contents of register pair HL, and the result is stored in HL. Operand ss is specified as follows in the assembled object code.

Register Pair	ss
BC	00
DE	01
HL	10
SP	11

M cycles: 4 T states: 15(4,4,4,3) 4 MHz E.T.: 3.75

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Set if carry out of Bit 11; reset otherwise
P/V: Set if overflow; reset otherwise
N: Reset
C: Set if carry from Bit 15; reset otherwise

Example:

If the register pair BC contains 2222H, register pair HL contains 5437H and the Carry Flag is set, after the execution of

ADC HL, BC

the contents of HL will be 765AH.

SBC HL,ss

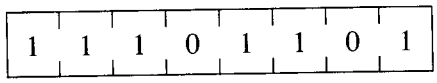
SuBtract with Carry

Operation: HL ◊ HL – ss – CY

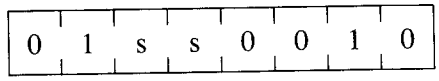
Format:

Mnemonic: SBC **Operands:** HL, ss

Object Code:



ED



Description:

The contents of the register pair ss (any of register pairs BC, DE, HL or SP) and the Carry Flag (C flag in the F register) are subtracted from the contents of register pair HL and the result is stored in HL. Operand ss is specified as follows in the assembled object code.

Register Pair	ss
BC	00
DE	01
HL	10
SP	11

M cycles: 4 T states: 15(4,4,4,3) 4 MHz E.T.: 3.75

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Set if borrow from Bit 12; reset otherwise
P/V: Set if overflow; reset otherwise
N: Set
C: Set if borrow; reset otherwise

Example:

If the contents of the HL register pair are 9999H, the contents of register pair DE are 1111H, and the Carry Flag is set, after the execution of

SBC HL, DE

the contents of HL will be 8887H.

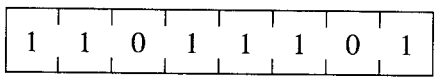
ADD IX,pp

Operation: IX◊IX+pp

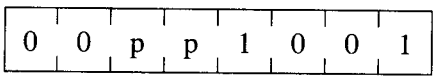
Format:

Mnemonic: ADD **Operands:** IX,pp

Object Code:



DD



Description:

The contents of register pair pp (any of register pairs BC, DE, IX or SP) are added to the contents of the Index Register IX, and the results are stored in IX. Operand pp is specified as follows in the assembled object code.

Register Pair	pp
BC	ØØ
DE	Ø1
IX	1Ø
SP	11

M cycles: 4 T states: 15(4,4,4,3) 4 MHz E.T.: 3.75

Condition Bits Affected:

S: Not affected
Z: Not affected
H: Set if carry out of Bit 11; reset otherwise
P/V: Not affected
N: Reset
C: Set if carry from Bit 15; reset otherwise

Example:

If the contents of Index Register IX are 3333H and the contents of register pair BC are 5555H, after the execution of

ADD IX, BC

the contents of IX will be 8888H.

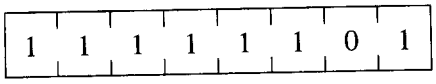
ADD IY,rr

Operation: IY ◊ IY + rr

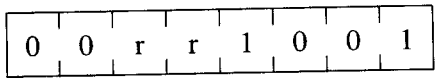
Format:

Mnemonic: ADD **Operands:** IY, rr

Object Code:



FD



Description:

The contents of register pair rr (any of register pairs BC, DE, IY or SP) are added to the contents of Index Register IY, and the result is stored in IY. Operand rr is specified as follows in the assembled object code.

Register Pair	rr
BC	00
DE	01
IY	10
SP	11

M cycles: 4 T states: 15(4,4,4,3) 4 MHz E.T.: 3.75

Condition Bits Affected:

S: Not affected
Z: Not affected
H: Set if carry out of Bit 11; reset otherwise
P/V: Not affected
N: Reset
C: Set if carry from Bit 15; reset otherwise

Example:

If the contents of Index Register IY are 333H and the contents of register pair BC are 555H, after the execution of

ADD IY, BC

the contents of IY will be 888H.

INC ss

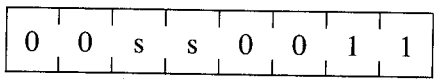
INCrement

Operation: ss ◊ ss + 1

Format:

Mnemonic: INC **Operands:** ss

Object Code:



Description:

The contents of register pair ss (any of register pairs BC, DE, HL or SP) are incremented. Operand ss is specified as follows in the assembled object code.

Register Pair	ss
BC	00
DE	01
HL	10
SP	11

M cycles: 1 T states: 6 4 MHz E.T.: 1.50

Condition Bits Affected: None

Example:

If the register pair contains 1000H, after the execution of
INC HL
HL will contain 1001H.

INC IX

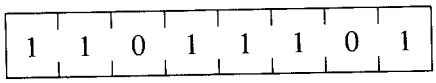
INCrement

Operation: IX ◊ IX + 1

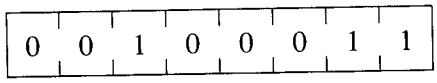
Format:

Mnemonic: INC **Operands:** IX

Object Code:



DD



23

Description:

The contents of the Index Register IX are incremented.

M cycles: 2 T states: 10(4,6) 4 MHz E.T.: 2.50

Condition Bits Affected: None

Example:

If the Index Register IX contains the integer 3300H after the execution of
INC IX
the contents of Index Register IX will be 3301H.

INC IY

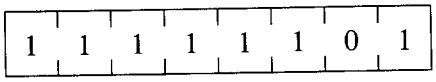
INCrement

Operation: IY⇩IY+1

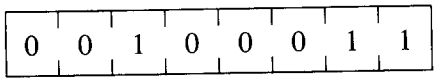
Format:

Mnemonic: INC **Operands:** IY

Object Code:



FD



23

Description:

The contents of the Index Register IY are incremented.

M cycles: 2 T states: 10(4,6) 4 MHz E.T.: 2.50

Condition Bits Affected: None

Example:

If the contents of the Index Register are 2977H, after the execution of

INC IY

the contents of Index Register IY will be 2978H.

DEC ss

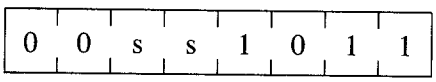
DECrement

Operation: ss ◊ ss − 1

Format:

Mnemonic: DEC **Operands:** ss

Object Code:



Description:

The contents of register pair ss (any of the register pairs BC, DE, HL or SP) are decremented. Operand ss is specified as follows in the assembled object code.

Register Pair	ss
BC	00
DE	01
HL	10
SP	11

M cycles: 1 T states: 6 4 MHz E.T.: 1.50

Condition Bits Affected: None

Example:

If register pair HL contains 1001H, after the execution of

DEC HL

the contents of HL will be 1000H.

DEC IX

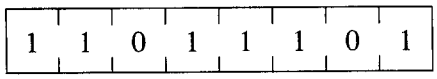
DECrement

Operation: IX◊IX−1

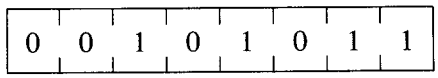
Format:

Mnemonic: DEC **Operands:** IX

Object Code:



DD



2B

Description:

The contents of Index Register IX are decremented.

M cycles: 2 T states: 10(4,6) 4 MHz E.T.: 2.50

Condition Bits Affected: None

Example:

If the contents of Index Register IX are 2006H, after the execution of

DEC IX

the contents of Index Register IX will be 2005H.

DEC IY

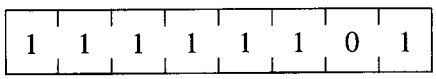
DECrement

Operation: IY◊IY−1

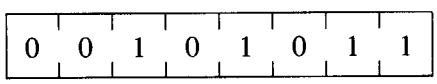
Format:

Mnemonic: DEC **Operands:** IY

Object Code:



FD



2B

Description:

The contents of the Index Register IY are decremented.

M cycles: 2 T states: 10(4,6) 4 MHz E.T.: 2.50

Condition Bits Affected: None

Example:

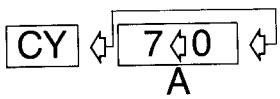
If the contents of the Index Register IY are 7649H, after the execution of

DEC IY

the contents of Index Register IY will be 7648H.

Rotate and Shift Group

RLCA — Rotate Left Circular Accumulator

Operation: 

Format:

Mnemonic: RLCA **Operands:**

Object Code:

0	0	0	0	0	1	1	1

07

Description:
The contents of the Accumulator (register A) are rotated left: the content of bit 0 is moved to bit 1; the previous content of bit 1 is moved to bit 2; this pattern is continued throughout the register. The content of bit 7 is copied into the Carry Flag (C flag in register F) and also into bit 0. (Bit 0 is the least significant bit.)

M cycles: 1 T states: 4 4 MHz E.T.: 1.00

Condition Bits Affected:

S: Not affected
Z: Not affected
H: Reset
P/V: Not affected
N: Reset
C: Data from Bit 7 of Acc.

Example:
If the contents of the Accumulator are

7	6	5	4	3	2	1	0
1	0	0	0	1	0	0	0

after the execution of

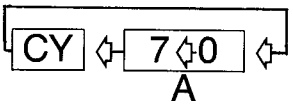
RLCA

the contents of the Carry Flag and the Accumulator will be

C	7	6	5	4	3	2	1	0
1	0	0	0	1	0	0	0	1

RLA

Rotate Left Accumulator

Operation: 

Format:

Mnemonic: RLA **Operands:**

Object Code:

0	0	0	1	0	1	1	1

17

Description:

The contents of the Accumulator (register A) are rotated left: the content of bit 0 is copied into bit 1; the previous content of bit 1 is copied into bit 2; this pattern is continued throughout the register. The content of bit 7 is copied into the Carry Flag (C flag in register F) and the previous content of the Carry Flag is copied into bit 0. Bit 0 is the least significant bit.

M cycles: 1 T states: 4 4 MHz E.T.: 1.00

Condition Bits Affected:

S: Not affected
Z: Not affected
H: Reset
P/V: Not affected
N: Reset
C: Data from Bit 7 of Acc.

Example:

If the contents of the Carry Flag and the Accumulator are

C	7	6	5	4	3	2	1	0
1	0	1	1	1	0	1	1	0

after the execution of

RLA

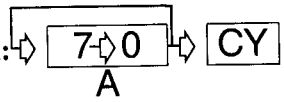
the contents of the Carry Flag and the Accumulator will be

C	7	6	5	4	3	2	1	0
0	1	1	1	0	1	1	0	1

RRCA

Rotate Right Circular Accumulator

Operation:



Format:

Mnemonic: RRCA **Operands:**

Object Code:

0	0	0	0	1	1	1	1

0F

Description:

The contents of the Accumulator (register A) are rotated right: the content of bit 7 is copied into bit 6; the previous content of bit 6 is copied into bit 5; this pattern is continued throughout the register. The content of bit 0 is copied into bit 7 and also into the Carry Flag (C flag in register F.) Bit 0 is the least significant bit.

M cycles: 1 T states: 4 4 MHz E.T.: 1.00

Condition Bits Affected:

S: Not affected
Z: Not affected
H: Reset
P/V: Not affected
N: Reset
C: Data from Bit 0 of Acc.

Example:

If the contents of the Accumulator are

7	6	5	4	3	2	1	0
0	0	0	1	0	0	0	1

After the execution of

RRCA

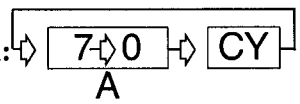
the contents of the Accumulator and the Carry Flag will be

7	6	5	4	3	2	1	0	C
1	0	0	0	1	0	0	0	1

RRA

Rotate Right Accumulator

Operation:



Format:

Mnemonic: RRA **Operands:**

Object Code:

0	0	0	1	1	1	1	1

1F

Description:

The contents of the Accumulator (register A) are rotated right: the content of bit 7 is copied into bit 6; the previous content of bit 6 is copied into bit 5; this pattern is continued throughout the register. The content of bit Ø is copied into the Carry Flag (C flag in register F) and the previous content of the Carry Flag is copied into bit 7. Bit Ø is the least significant bit.

M cycles: 1 T states: 4 4 MHz E.T.: 1.ØØ

Condition Bits Affected:

S: Not affected
Z: Not affected
H: Reset
P/V: Not affected
N: Reset
C: Data from Bit Ø of Acc.

Example:

If the contents of the Accumulator and the Carry Flag are

7	6	5	4	3	2	1	0	C
1	1	1	0	0	0	0	1	0

after the execution of

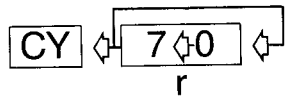
RRA

the contents of the Accumulator and the Carry Flag will be

7	6	5	4	3	2	1	0	C
0	1	1	1	0	0	0	0	1

RLC r

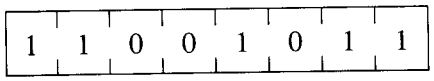
Rotate Left Circular

Operation: 

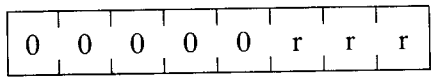
Format:

Mnemonic: RLC **Operands:** r

Object Code:



CB



Description:

The eight-bit contents of register r are rotated left: the content of bit 0 is copied into bit 1; the previous content of bit 1 is copied into bit 2; this pattern is continued throughout the register. The content of bit 7 is copied into the Carry Flag (C flag in register F) and also into bit 0. Operand r is specified as follows in the assembled object code:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Note: Bit 0 is the least significant bit.

M cycles: 2 T states: 8(4,4) 4 MHz E.T.: 2.00

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Reset
P/V: Set if parity even; reset otherwise
N: Reset
C: Data from Bit 7 of source register

Example:

If the contents of register r are

7	6	5	4	3	2	1	0
1	0	0	0	1	0	0	0

after the execution of

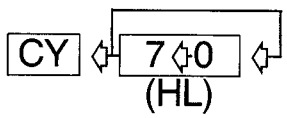
RLC r

the contents of the Carry Flag and register r will be

C	7	6	5	4	3	2	1	0
1	0	0	0	1	0	0	0	1

RLC (HL)

Rotate Left Circular

Operation: 

Format:

Mnemonic: RLC **Operands:** (HL)

Object Code:

1	1	0	0	1	0	1	1	CB
0	0	0	0	0	1	1	0	Ø6

Description:

The contents of the memory address specified by the contents of register pair HL are rotated left: the content of bit Ø is copied into bit 1; the previous content of bit 1 is copied into bit 2; this pattern is continued throughout the byte. The content of bit 7 is copied into the Carry Flag (C flag in register F) and also into bit Ø. Bit Ø is the least significant bit.

M cycles: 4 T states: 15(4,4,4,3) 4 MHz E.T.: 3.75

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Reset
P/V: Set if parity even; reset otherwise
N: Reset
C: Data from Bit 7 of source register

Example:

If the contents of the HL register pair are 2828H, and the contents of memory location 2828H are

7	6	5	4	3	2	1	0
1	0	0	0	1	0	0	0

after the execution of
RLC (HL)
the contents of the Carry Flag and memory locations 2828H will be

C	7	6	5	4	3	2	1	0
1	0	0	0	1	0	0	0	1

RLC (IX+d)

Rotate Left Circular

Operation: CY ⇦ 7 ⇦ 0 ⇦ (IX+d)

Format:

Mnemonic: RLC **Operands:** (IX+d)

Object Code:

1	1	0	1	1	1	0	1	DD
1	1	0	0	1	0	1	1	CB
d	d	d	d	d	d	d	d	
0	0	0	0	0	1	1	0	Ø6

Description:

The contents of the memory address specified by the sum of the contents of the Index Register IX and a two's complement displacement integer d, are rotated left: the contents of bit 0 is copied into bit 1; the previous content of bit 1 is copied into bit 2; this pattern is continued throughout the byte. The content of bit 7 is copied into the Carry Flag (C flag in register F) and also into bit 0. Bit 0 is the least significant bit.

M cycles: 6 T states: 23(4,4,3,5,4,3) 4 MHz E.T.: 5.75

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Reset
P/V: Set if parity even; reset otherwise
N: Reset
C: Data from Bit 7 of source register

Example:

If the contents of the Index Register IX are 1000H, and the contents of memory location 1002H are

7	6	5	4	3	2	1	0
1	0	0	0	1	0	0	0

after the execution of

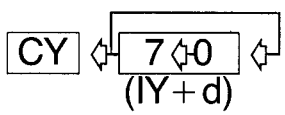
RLC (IX+2H)

the contents of the Carry Flag and memory location 1002H will be

C	7	6	5	4	3	2	1	0
1	0	0	0	1	0	0	0	1

RLC (IY+d)

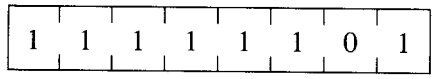
Rotate Left Circular

Operation: 

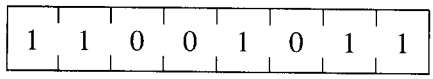
Format:

Mnemonic: RLC **Operands:** (IY+d)

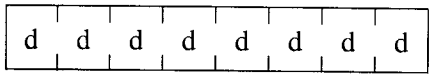
Object Code:

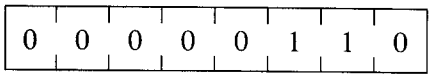


FD



CB





06

Description:

The contents of the memory address specified by the sum of the contents of the Index Register IY and a two's complement displacement integer d are rotated left: the content of bit 0 is copied into bit 1; the previous content of bit 1 is copied into bit 2; this process is continued throughout the byte. The content of bit 7 is copied into the Carry Flag (C flag in register F) and also into bit 0. Bit 0 is the least significant bit.

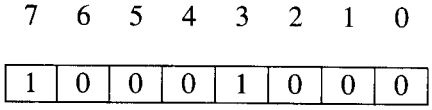
M cycles: 6 T states: 23(4,4,3,5,4,3) 4 MHz E.T.: 5.75

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Reset
P/V: Set if parity even; reset otherwise
N: Reset
C: Data from Bit 7 of source register

Example:

If the contents of the Index Register IY are 1000H, and the contents of memory location 1002H are



after the execution of
RLC (IY+2H)
the contents of the Carry Flag and memory location 1002H will be

C	7	6	5	4	3	2	1	0
1	0	0	0	1	0	0	0	1

RL m

Rotate Left

Operation: CY ⇦ 7 ⇦ 0 ⇦ m

Format:

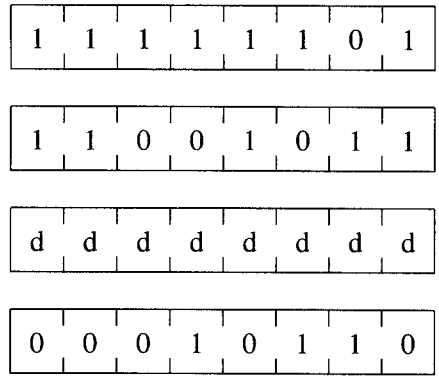
Mnemonic: RL **Operands:** m

The m operand is any of r, (HL), (IX+d) or (IY+d), as defined for the analogous RLC instructions. These various possible opcode-operand combinations are specified as follows in the assembled object code:

Object Code:

RL r	1 1 0 0 1 0 1 1	CB
	0 0 0 1 0 r r r	
RL (HL)	1 1 0 0 1 0 1 1	CB
	0 0 0 1 0 1 1 0	16
RL (IX+d)	1 1 0 1 1 1 0 1	DD
	1 1 0 0 1 0 1 1	CB
	d d d d d d d d	
	0 0 0 1 0 1 1 0	16

RL (IY+d)



r identifies register B, C, D, E, H, L or A specified as follows in the assembled object code above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description:

The contents of the m operand are rotated left: the content of bit 0 is copied into bit 1; the previous content of bit 1 is copied into bit 2; this pattern is continued throughout the byte. The content of bit 7 is copied into the Carry Flag (C flag in register F) and the previous content of the Carry Flag is copied into bit 0. Bit 0 is the least significant bit.

Instruction	M Cycles	T States	4 MHz E.T. in μs
RL r	2	8(4,4)	2.00
RL (HL)	4	15(4,4,4,3)	3.75
RL (IX+d)	6	23(4,4,3,5,4,3)	5.75
RL (IY+d)	6	23(4,4,3,5,4,3)	5.75

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Reset
P/V: Set if parity even; reset otherwise
N: Reset
C: Data from Bit 7 of source register

Example:

If the contents of the Carry Flag and register D are

C	7	6	5	4	3	2	1	0
0	1	0	0	0	1	1	1	1

after the execution of

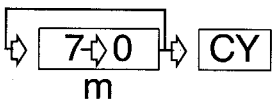
RL D

the contents of the Carry Flag and register D will be

C	7	6	5	4	3	2	1	0
1	0	0	0	1	1	1	1	0

RRC m

Rotate Right Circular

Operation: 

Format:

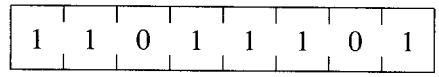
Mnemonic: RRC **Operands:** m

The m operand is any of r, (HL), (IX+d) or (IY+d), as defined for the analogous RLC instructions. These various possible opcode-operand combinations are specified as follows in the assembled object code:

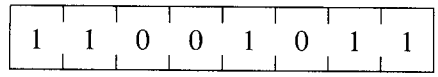
Object Code:

RRC r	1 1 0 0 1 0 1 1	CB
	0 0 0 0 1 r r r	
RRC (HL)	1 1 0 0 1 0 1 1	CB
	0 0 0 0 1 1 1 0	ØE

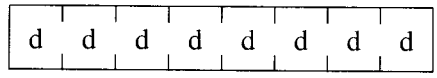
RRC (IX+d)

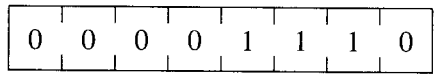


DD



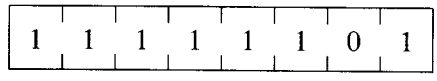
CB



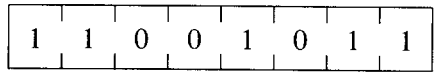


ØE

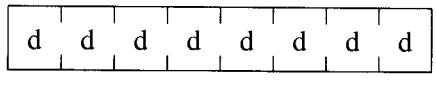
RRC (IY+d)

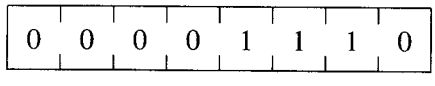


FD



CB





ØE

r identifies register B, C, D, E, H, L or A specified as follows in the assembled object code above:

Register	r
B	ØØØ
C	ØØ1
D	Ø1Ø
E	Ø11
H	1ØØ
L	1Ø1
A	111

Description:

The contents of operand m are rotated right: the content of bit 7 is copied into bit 6; the previous content of bit 6 is copied into bit 5; this pattern is continued throughout the byte. The content of bit Ø is copied into the Carry Flag (C flag in the F register) and also into bit 7. Bit Ø is the least significant bit.

Instruction	M Cycles	T States	4 MHz E.T. in μs
RRC r	2	8(4,4)	2.00
RRC (HL)	4	15(4,4,4,3)	3.75
RRC (IX+d)	6	23(4,4,3,5,4,3)	5.75
RRC (IY+d)	6	23(4,4,3,5,4,3)	5.75

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Reset
P/V: Set if parity even; reset otherwise
N: Reset
C: Data from Bit 0 of source register

Example:

If the contents of register A are

7	6	5	4	3	2	1	0
0	0	1	1	0	0	0	1

after the execution of

RRC A

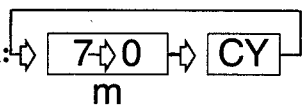
the contents of register A and the Carry Flag will be

7	6	5	4	3	2	1	0	C
1	0	0	1	1	0	0	0	1

RR m

Rotate Right

Operation:



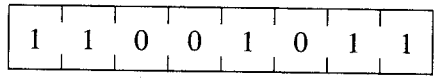
Format:

Mnemonic: RR **Operands:** m

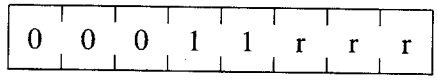
The m operand is any of r, (HL), (IX+d) or (IY+d), as defined for the analogous RLC instructions. These various possible opcode-operand combinations are specified as follows in the assembled object code:

Object Code:

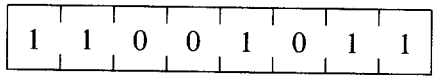
RR r



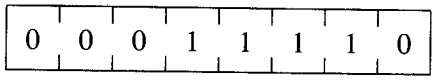
CB



RR (HL)

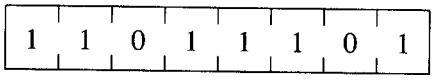


CB

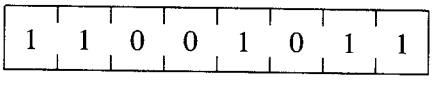


1E

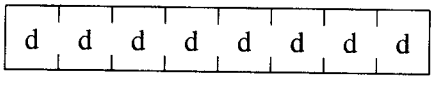
RR (IX+d)

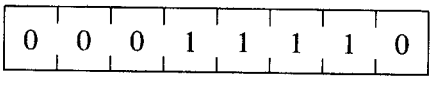


DD



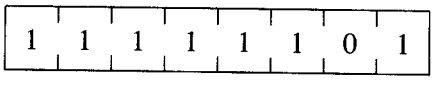
CB



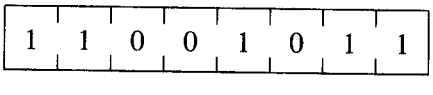


1E

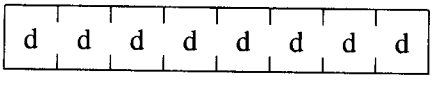
RR (IY+d)

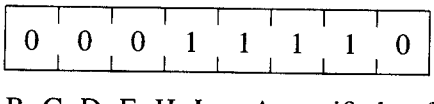


FD



CB





1E

r identifies registers B, C, D, E, H, L or A specified as follows in the assembled object code above:

Register	**r**
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description:

The contents of operand m are rotated right: the contents of bit 7 is copied into bit 6; the previous content of bit 6 is copied into bit 5; this pattern is continued throughout the byte. The content of bit Ø is copied into the Carry Flag (C flag in register F) and the previous content of the Carry Flag is copied into bit 7. Bit Ø is the least significant bit.

Instruction	M Cycles	T States	4 MHz E.T. in μs
RR r	2	8(4,4)	2.ØØ
RR (HL)	4	15(4,4,4,3)	3.75
RR (IX+d)	6	23(4,4,3,5,4,3)	5.75
RR (IY+d)	6	23(4,4,3,5,4,3)	5.75

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Reset
P/V: Set if parity is even; reset otherwise
N: Reset
C: Data from Bit Ø of source register

Example:

If the contents of the HL register pair are 4343H, and the contents of memory location 4343H and the Carry Flag are

7	6	5	4	3	2	1	0	C
1	1	0	1	1	1	0	1	0

after the execution of

RR (HL)

the contents of location 4343H and the Carry Flag will be

7	6	5	4	3	2	1	0	C
0	1	1	0	1	1	1	0	1

SLA m

Shift Left Arithmetic

Operation: CY ⇐ 7⇐0 ⇐0
m

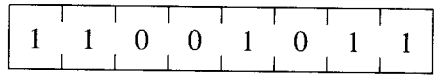
Format:

Mnemonic: SLA **Operands:** m

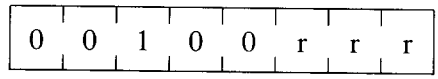
The m operand is any of r, (HL), (IX+d) or (IY+d), as defined for the analogous RLC instructions. These various possible opcode-operand combinations are specified as follows in the assembled object code:

Object Code:

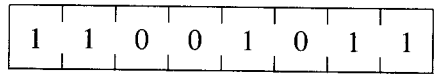
SLA r



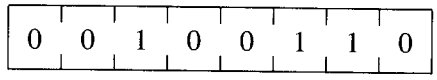
CB



SLA (HL)

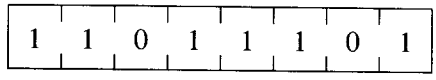


CB

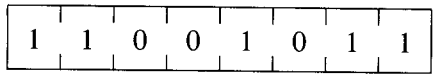


26

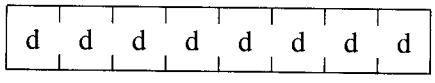
SLA (IX+d)

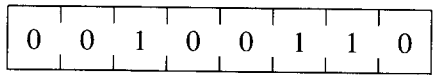


DD



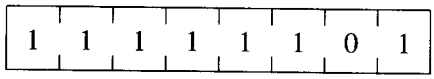
CB



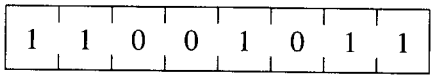


26

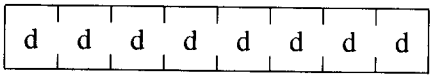
SLA (IY+d)

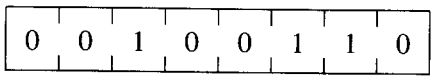


FD



CB





26

r identifies registers B, C, D, E, H, L or A specified as follows in the assembled object code field above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description:

An arithmetic shift left is performed on the contents of operand m: bit 0 is reset, the previous content of bit 0 is copied into bit 1, the previous content of bit 1 is copied into bit 2; this pattern is continued throughout; the content of bit 7 is copied into the Carry Flag (C flag in register F). Bit 0 is the least significant bit.

Instruction	M Cycles	T States	4 MHz E.T. in μs
SLA r	2	8(4,4)	2.00
SLA (HL)	4	15(4,4,4,3)	3.75
SLA (IX+d)	6	23(4,4,3,5,4,3)	5.75
SLA (IY+d)	6	23(4,4,3,5,4,3)	5.75

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Reset
P/V: Set if parity is even; reset otherwise
N: Reset
C: Data from Bit 7

Example:

If the contents of register L are

7	6	5	4	3	2	1	0
1	0	1	1	0	0	0	1

after the execution of

SLA L

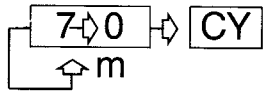
the contents of the Carry Flag and register L will be

C	7	6	5	4	3	2	1	0
1	0	1	1	0	0	0	1	0

SRA m

Shift Right Arithmetic

Operation:



Format:

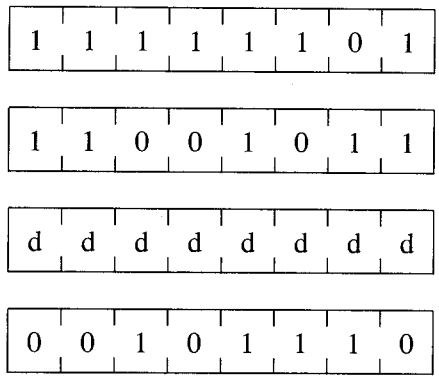
Mnemonic: SRA **Operands:** m

The m operand is any of r, (HL), (IX+d) or (IY+d), as defined for the analogous RLC instructions. These various possible opcode-operand combinations are specified as follows in the assembled object code:

Object Code:

SRA r	1 1 0 0 1 0 1 1	CB
	0 0 1 0 1 r r r	
SRA (HL)	1 1 0 0 1 0 1 1	CB
	0 0 1 0 1 1 1 0	2E
SRA (IX+d)	1 1 0 1 1 1 0 1	DD
	1 1 0 0 1 0 1 1	CB
	d d d d d d d d	
	0 0 1 0 1 1 1 0	2E

SRA (IY+d)



r means register B, C, D, E, H, L or A specified as follows in the assembled object code field above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

An arithmetic shift right is performed on the contents of operand m: the content of bit 7 is copied into bit 6; the previous content of bit 6 is copied into bit 5; this pattern is continued throughout the byte. The content of bit 0 is copied into the Carry Flag (C flag in register F), and the previous content of bit 7 is unchanged. Bit 0 is the least significant bit.

Instruction	M Cycles	T States	4 MHz E.T. in μs
SRA r	2	8(4,4)	2.00
SRA (HL)	4	15(4,4,4,3)	3.75
SRA (IX+d)	6	23(4,4,3,5,4,3)	5.75
SRA (IY+d)	6	23(4,4,3,5,4,3)	5.75

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Reset
P/V: Set if parity is even; reset otherwise
N: Reset
C: Data from Bit 0 of source register

Example:

If the contents of the Index Register IX are 1000H, and the contents of memory location 1003H are

7	6	5	4	3	2	1	0
1	0	1	1	1	0	0	0

after the execution of

SRA (IX+3H)

the contents of memory location 1003H and the Carry Flag will be

7	6	5	4	3	2	1	0	C
1	1	0	1	1	1	0	0	0

SRL m

Shift Right Logical

Operation: 0 ⇨ [7⇨0] (m) ⇨ [CY]

Format:

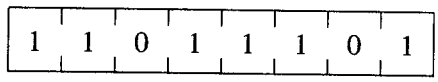
Mnemonic: SRL **Operands:** m

The operand m is any of r, (HL), (IX+d) or (IY+d), as defined for the analogous RLC instructions. These various possible opcode-operand combinations are specified as follows in the assembled object code:

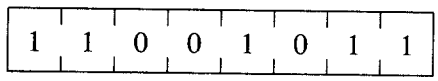
Object Code:

SRL r	1 1 0 0 1 0 1 1	CB
	0 0 1 1 1 r r r	
SRL (HL)	1 1 0 0 1 0 1 1	CB
	0 0 1 1 1 1 1 0	3E

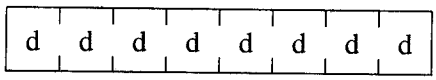
SRL (IX+d)

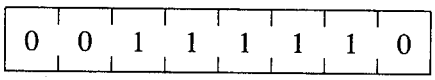


DD



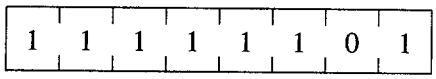
CB



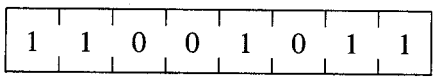


3E

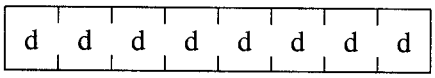
SRL (IY+d)

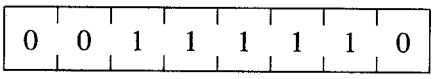


FD



CB





3E

r identifies registers B, C, D, E, H, L or A specified as follows in the assembled object code fields above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description:

The contents of operand m are shifted right: the content of bit 7 is copied into bit 6; the content of bit 6 is copied into bit 5; this pattern is continued throughout the byte. The content of bit 0 is copied into the Carry Flag, and bit 7 is reset. Bit 0 is the least significant bit.

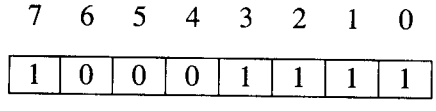
Instruction	M Cycles	T States	4 MHz E.T. in μs
SRL r	2	8(4,4,)	2.00
SRL (HL)	4	15(4,4,4,3)	3.75
SRL (IX+d)	6	23(4,4,3,5,4,3)	5.75
SRL (IY+d)	6	23(4,4,3,5,4,3)	5.75

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Reset
P/V: Set if parity is even; reset otherwise
N: Reset
C: Data from Bit 0 of source register

Example:

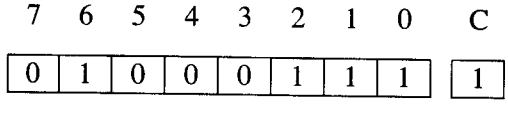
If the contents of register B are



after the execution of

SRL B

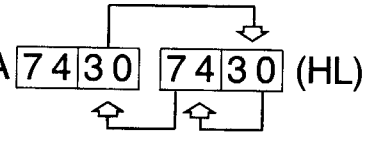
the contents of register B and the Carry Flag will be



RLD

Rotate Left Decimal

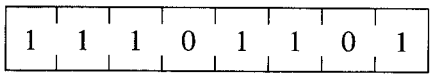
Operation:



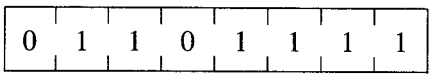
Format:

Mnemonic: RLD **Operands:**

Object Code:



ED



6F

Description:

The contents of the low order four bits (bits 3, 2, 1 and Ø) of the memory location (HL) are copied into the high order four bits (7, 6, 5 and 4) of that same memory location; the previous contents of those high order four bits are copied into the low order four bits of the Accumulator (register A), and the previous contents of the low order four bits of the Accumulator are copied into the low order four bits of memory location (HL). The contents of the high order bits of the Accumulator are unaffected. Note: (HL) means the memory location specified by the contents of the HL register pair.

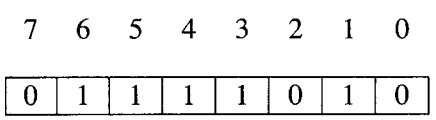
M cycles: 5 T states: 18(4,4,3,4,3) 4 MHz E.T.: 4.5Ø

Condition Bits Affected:

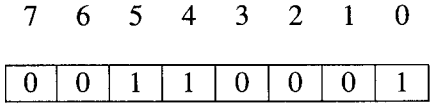
S: Set if Acc. is negative after operation; reset otherwise
Z: Set if Acc. is zero after operation; reset otherwise
H: Reset
P/V: Set if parity of Acc. is even after operation; reset otherwise
N: Reset
C: Not affected

Example:

If the contents of the HL register pair are 5ØØØH, and the contents of the Accumulator and memory location 5ØØØH are



Accumulator



(5ØØØH)

after the execution of
RLD
the contents of the Accumulator and memory location 5000H will be

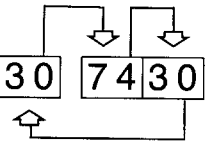
7	6	5	4	3	2	1	0	
0	1	1	1	0	0	1	1	Accumulator

7	6	5	4	3	2	1	0	
0	0	0	1	1	0	1	0	(5000H)

RRD

Rotate Right Decimal

Operation: A 7 4 3 0 7 4 3 0 (HL)



Format:

Mnemonic: RRD **Operands:**

Object Code:

1	1	1	0	1	1	0	1	ED
0	1	1	0	0	1	1	1	67

Description:

The contents of the low order four bits (bits 3, 2, 1 and 0) of memory location (HL) are copied into the low order four bits of the Accumulator (register A); the previous contents of the low order four bits of the Accumulator are copied into the high order four bits (7, 6, 5 and 4) of location (HL); and the previous contents of the high order four bits of (HL) are copied into the low order four bits of (HL). The contents of the high order bits of the Accumulator are unaffected. Note: (HL) means the memory location specified by the contents of the HL register pair.

M cycles: 5 T states: 18(4,4,3,4,3) 4 MHz E.T.: 4.50

Condition Bits Affected:

S: Set if Acc. is negative after operation; reset otherwise
Z: Set if Acc. is zero after operation; reset otherwise
H: Reset
P/V: Set if parity of Acc. is even after operation; reset otherwise
N: Reset
C: Not affected

Example:

If the contents of the HL register pair are 5000H, and the contents of the Accumulator and memory location 5000H are

7	6	5	4	3	2	1	0	
1	0	0	0	0	1	0	0	Accumulator

7	6	5	4	3	2	1	0	
0	0	1	0	0	0	0	0	(5000H)

after the execution of

RRD

the contents of the Accumulator and memory location 5000H will be

7	6	5	4	3	2	1	0	
1	0	0	0	0	0	0	0	Accumulator

7	6	5	4	3	2	1	0	
0	1	0	0	0	0	1	0	(5000H)

Bit Set, Reset and Test Group

BIT b, r

BIT test

Operation: $Z \Leftrightarrow \bar{r}_b$

Format:

Mnemonic: BIT **Operands:** b, r

Object Code:

1	1	0	0	1	0	1	1

CB

0	1	b	b	b	r	r	r

Description:

After the execution of this instruction, the Z flag in the F register will contain the complement of the indicated bit within the indicated register. Operands b and r are specified as follows in the assembled object code:

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

M cycles: 2 T states: 8(4,4) 4 MHz E.T.: 2.00

Condition Bits Affected:

S: Unknown
Z: Set if specified Bit is 0; reset otherwise
H: Set
P/V: Unknown
N: Reset
C: Not affected

Example:

If bit 2 in register B contains Ø, after the execution of

BIT 2, B

the Z flag in the F register will contain 1, and bit 2 in register B will remain Ø. (Bit Ø in register B is the least significant bit.)

BIT b,(HL)

Bit Test

Operation: $Z \Leftarrow \overline{(HL)_b}$

Format:

Mnemonic: BIT **Operands:** b, (HL)

Object Code:

1	1	0	0	1	0	1	1

CB

0	1	b	b	b	1	1	0

Description:

This instruction tests bit b in the memory location specified by the contents of the HL register pair and sets the Z flag accordingly. Operand b is specified as follows in the assembled object code:

Bit Tested	b
Ø	ØØØ
1	ØØ1
2	Ø1Ø
3	Ø11
4	1ØØ
5	1Ø1
6	11Ø
7	111

M cycles: 3 T states: 12(4,4,4) 4 MHz E.T.: 3.ØØ

Condition Bits Affected:

S: Unknown
Z: Set if specified Bit is Ø; reset otherwise
H: Set
P/V: Unknown
H: Reset
C: Not affected

Example:

If the HL register pair contains 444H, and bit 4 in the memory location 444H contains 1, after the execution of

BIT 4,(HL)

the Z flag in the F register will contain Ø, and bit 4 in memory location 444H will still contain 1. (Bit Ø in memory location 444H is the least significant bit.)

BIT b,(IX+d)

Bit Test

Operation: $Z \Leftarrow \overline{(IX+d)_b}$

Format:

Mnemonic: BIT **Operands:** b, (IX+d)

Object Code:

1	1	0	1	1	1	0	1	DD
1	1	0	0	1	0	1	1	CB
d	d	d	d	d	d	d	d	
0	1	b	b	b	1	1	0	

Description:

After the execution of this instruction, the Z flag in the F register will contain the complement of the indicated bit within the contents of the memory location pointed to by the sum of the contents register pair IX (Index Register IX) and the two's complement displacement integer d. Operand b is specified as follows in the assembled object code.

Bit Tested	b
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

M cycles: 5 T states: 20(4,4,3,5,4) 4 MHz E.T.: 5.00

Condition Bits Affected:

S: Unknown
Z: Set if specified Bit is 0; reset otherwise
H: Set
P/V: Unknown
N: Reset
C: Not affected

Example:

If the contents of Index Register IX are 2000H, and bit 6 in memory location 2004H contains 1, after the execution of

BIT 6,(IX+4H)

the Z flag in the F register will contain 0, and bit 6 in memory location 2004H will still contain 1. (Bit 0 in memory location 2004H is the least significant bit.)

BIT b,(IY+d)

BIT Test

Operation: $Z \Leftarrow \overline{(IY+d)_b}$

Format:

Mnemonic: BIT **Operands:** b, (IY+d)

Object Code:

1	1	1	1	1	1	0	1	FD
1	1	0	0	1	0	1	1	CB
d	d	d	d	d	d	d	d	
0	1	b	b	b	1	1	0	

Description:

After the execution of this instruction, the Z flag in the F register will contain the complement of the indicated bit within the contents of the memory location pointed to by the sum of the contents of register pair IY (Index Register IY) and the two's complement displacement integer d. Operand b is specified as follows in the assembled object code:

Bit Tested	b
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

M cycles: 5 T states: 20(4,4,3,5,4) 4 MHz E.T.: 5.00

Condition Bits Affected:

S: Unknown
Z: Set if specified Bit is 0; reset otherwise
H: Set
P/V: Unknown
N: Reset
C: Not affected

Example:

If the contents of Index Register are 2000H, and bit 6 in memory location 2004H contains 1, after the execution of

BIT 6,(IY+4H)

the Z flag in the F register still contain 0, and bit 6 in memory location 2004H will still contain 1. (Bit 0 in memory location 2004H is the least significant bit.)

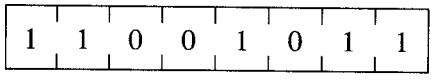
SET b,r

Operation: $r_b \Leftarrow 1$

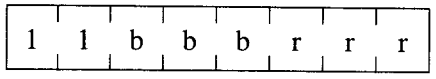
Format:

Mnemonic: SET **Operands:** b, r

Object Code:



CB



Description:

Bit b (any bit, 7 through Ø) in register r (any of register B, C, D, E, H, L or A) is set. Operands b and r are specified as follows in the assembled object code:

Bit Tested	b	Register	r
Ø	ØØØ	B	ØØØ
1	ØØ1	C	ØØ1
2	Ø1Ø	D	Ø1Ø
3	Ø11	E	Ø11
4	1ØØ	H	1ØØ
5	1Ø1	L	1Ø1
6	11Ø	A	111
7	111		

M cycles: 2 T states: 8(4,4) 4 MHz E.T.: 2.ØØ

Condition Bits Affected: None

Example:

After the execution of

SET 4,A

bit 4 in register A will be set. (Bit Ø is the least significant bit.)

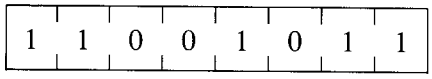
SET b,(HL)

Operation: $(HL)_b \leftarrow 1$

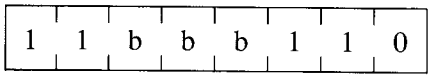
Format:

Mnemonic: SET **Operands:** b, (HL)

Object Code:



CB



Description:

Bit b (any bit, 7 through 0) in the memory location addressed by the contents of register pair HL is set. Operand b is specified as follows in the assembled object code:

Bit Tested	b
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

M cycles: 4 T states: 15(4,4,4,3) 4 MHz E.T.: 3.75

Condition Bits Affected: None

Example:

If the contents of the HL register pair are 3000H, after the execution of

SET 4,(HL)

bit 4 in memory location 3000H will be 1. (Bit 0 in memory location 3000H is the least significant bit.)

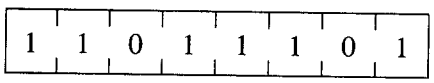
SET b,(IX+d)

Operation: $(IX+d)_b \Leftarrow 1$

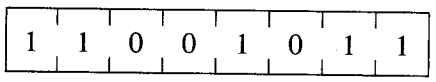
Format:

Mnemonic: SET **Operands:** b, (IX+d)

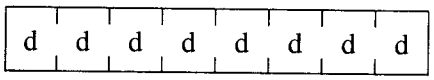
Object Code:

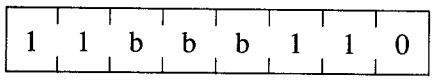


DD



CB





Description:

Bit b (any bit, 7 through Ø) in the memory location addressed by the sum of the contents of the IX register pair (Index Register IX) and the two's complement integer d is set. Operand b is specified as follows in the assembled object code:

Bit Tested	b
Ø	ØØØ
1	ØØ1
2	Ø1Ø
3	Ø11
4	1ØØ
5	1Ø1
6	11Ø
7	111

M cycles: 6 T states: 23(4,4,3,5,4,3) 4 MHz E.T.: 5.75

Condition Bits Affected: None

Example:

If the contents of Index Register are 2ØØØH, after the execution of

SET Ø,(IX+3H)

bit Ø in memory location 2ØØ3H will be 1. (Bit Ø in memory location 2ØØ3H is the least significant bit.)

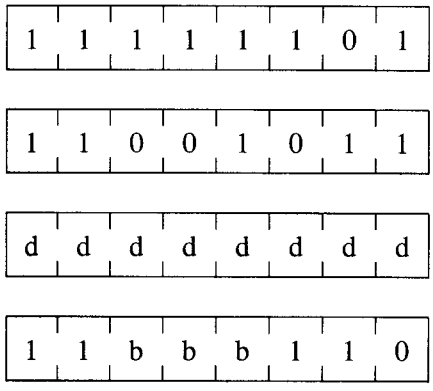
SET b,(IY+d)

Operation: $(IY+d)_b \Leftarrow 1$

Format:

Mnemonic: SET **Operands:** b, (IY+d)

Object Code:



Description:

Bit b (any bit, 7 through 0) in the memory location addressed by the sum of the contents of the IY register pair (Index Register IY) and the two's complement displacement d is set. Operand b is specified as follows in the assembled object code:

Bit Tested	b
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

M cycles: 6 T states: 23(4,4,3,5,4,3) 4 MHz E.T.: 5.75

Condition Bits Affected: None

Example:

If the contents of Index Register IY are 2000H, after the execution of
SET 0,(IY+3H)
bit 0 in memory location 2003H will be 1. (Bit 0 in memory location 2003H is the least significant bit.)

RES b,m

RESet

Operation: $S_b \Leftarrow 0$

Format:

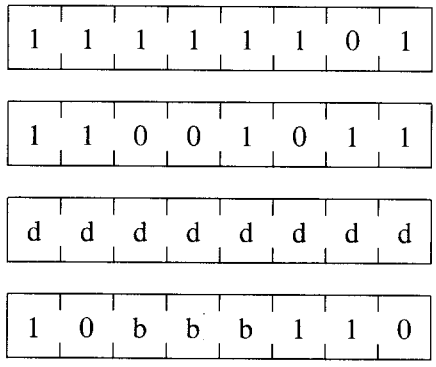
Mnemonic: RES **Operands:** b, m

Operand b is any bit (7 through 0) of the contents of the m operand, (any of r, (HL), (IX+d) or (IY+d) as defined for the analogous SET instructions. These various possible opcode-operand combinations are assembled as follows in the object code:

Object Code:

RES b, r	1 1 0 0 1 0 1 1	CB
	1 0 b b b r r r	
RES b, (HL)	1 1 0 0 1 0 1 1	CB
	1 0 b b b 1 1 0	
RES b, (IX+d)	1 1 0 1 1 1 0 1	DD
	1 1 0 0 1 0 1 1	CB
	d d d d d d d d	
	1 0 b b b 1 1 0	

RES b, (IY+d)



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

Description:

Bit b in operand m is reset.

Instruction	M Cycles	T States	4 MHz E.T. in μs
RES r	4	8(4,4)	2.00
RES (HL)	4	15(4,4,4,3)	3.75
RES (IX+d)	6	23(4,4,3,5,4,3)	5.75
RES (IY+d)	6	23(4,4,3,5,4,3)	5.75

Condition Bits Affected: None

Example 1:

After the execution of

RES 6,D (object code CB, B2H)

bit 6 in register D will be reset. (Bit 0 in register D is the least significant bit.)

Example 2:

If HL contains 7000H and address 7000H contains FFH, after

RES 0,(HL)

address 7000H will contain FEH.

Jump Group

JP nn

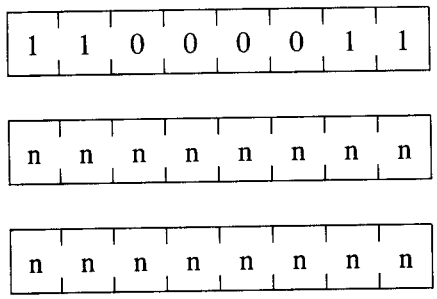
JumP

Operation: PC ⇦ nn

Format:

Mnemonic: JP **Operands:** nn

Object Code:



Note: The first operand in this assembled object code is the low order byte of a 2-byte address.

Description:

Operand nn is loaded into register pair PC (Program Counter) and points to the address of the next program instruction to be executed.

M cycles: 3 T states: 10(4,3,3) 4 MHz E.T.: 2.50

Condition Bits Affected: None

Example:

JP 50A1

This instruction will cause the program to jump to the instruction at 50A1H by loading the number 50A1H into the PC register.

JP cc,nn

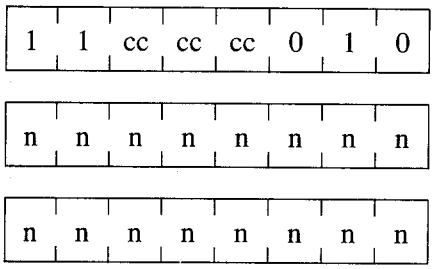
JumP

Operation: IF cc TRUE, PC ◊ nn

Format:

Mnemonic: JP **Operands:** cc, nn

Object Code:



Note: The first n operand in this assembled object code is the low order byte of a 2-byte memory address.

Description:

If condition cc is true, the instruction loads operand nn into register pair PC (Program Counter), and the program continues with the instruction beginning at address nn. If condition cc is false, the Program Counter is incremented as usual, and the program continues with the next sequential instruction. Condition cc is programmed as one of eight status bits which correspond to condition bits in the Flag Register (register F). These eight status bits are defined in the table below, which also specifies the corresponding cc bit fields in the assembled object code.

cc	Condition	Relevant Flag
000	NZ non-zero	Z (=0)
001	Z zero	Z (=1)
010	NC no-carry	C (=0)
011	C carry	C (=1)
100	PO parity odd	P/V (=0)
101	PE parity even	P/V (=1)
110	P sign positive	S (=0)
111	M sign negative	S (=1)

M cycles: 3 T states: 10(4,3,3) 4 MHz E.T.: 2.50

Condition Bits Affected: None

Example:

If the Carry Flag (C flag in the F register) is set and the contents of address 1520 are 03H, after the execution of

JP C,1520H

the Program Counter will contain 1520H, and on the next machine cycle the CPU will fetch from address 1520H the byte 03H.

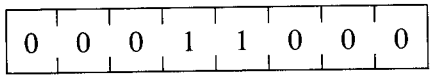
JR e — Jump Relative

Operation: PC ◊ PC + e

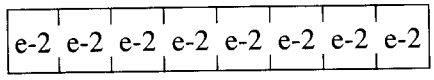
Format:

Mnemonic: JR **Operands:** e

Object Code:



18



Description:

This instruction provides for unconditional branching to other segments of a program. The value of the displacement e is added to the Program Counter (PC) and the next instruction is fetched from the location designated by the new contents of the PC. This jump as measured from the address of the instruction opcode has a range of −126 to +129 bytes. The assembler automatically adjusts for the twice incremented PC.

M cycles: 3 T states: 12(4,3,5) 4 MHz E.T.: 3.00

Condition Bits Affected: None

Example 1:

To jump forward five locations from address 480, the following assembly language statement is used:

JR $+5

The resulting object code and final PC value is shown below:

Location	Instruction
480	18
481	03
482	—◊PC before jump
483	—
484	—
485	◊PC after jump

Note: when using an assembler, $ + 5 used above would normally be replaced by a label.

Example 2:

This program will skip around the NOP instruction.

```
START   JR, END
        NOP
END     —
```

JR C,e

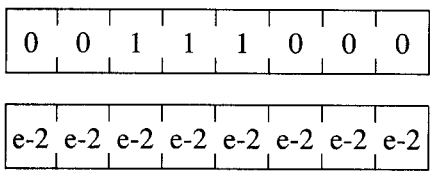
Jump Relative

Operation: If C = 0, continue
If C = 1, PC◊PC + e

Format:

Mnemonic: JR **Operands:** C, e

Object Code:



Description:

This instruction provides for conditional branching to other segments of a program depending on the results of a test on the Carry Flag. If the flag is equal to a '1,' the value of the displacement e is added to the Program Counter (PC) and the next instruction is fetched from the location designated by the new contents of the PC. The jump as measured from the address of the instruction opcode has a range of −126 to +129 bytes. The assembler automatically adjusts for the twice incremented PC.

If the flag is equal to a '0,' the next instruction to be executed is taken from the location following this instruction.

If condition is met:
M cycles: 3 T states: 12(4,3,5) 4 MHz E.T.: 3.00
If condition is not met:
M cycles: 2 T states: 7(4,3) 4 MHz E.T.: 1.75

Condition Bits Affected: None

Example:

The Carry Flag is set and it is required to jump back four locations from 480. The assembly language statement is:

JR C, $-4

The resulting object code and final PC value is shown below:

Location	Instruction
47C	◊PC after jump
47D	—
47E	—
47F	—
480	38
481	FA (two's complement − 6)
482	◊PC before jump

JR NC,e

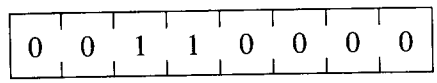
Jump Relative

Operation: If C = 1, continue
If C = 0, PC◊PC + e

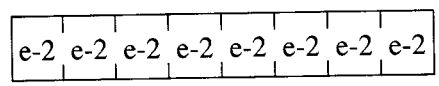
Format:

Mnemonic: JR **Operands:** NC, e

Object Code:



30



Description:

This instruction provides for conditional branching to other segments of a program depending on the results of a test on the Carry Flag. If the flag is equal to '0,' the value of the displacement e is added to the Program Counter (PC) and

the next instruction is fetched from the location designated by the new contents of the PC. The jump as measured from the address of the instruction opcode has a range of −126 to +129 bytes. The assembler automatically adjusts for the twice incremented PC.

If the flag is equal to a '1,' the next instruction to be executed is taken from the location following this instruction.

If the condition is met:

M cycles: 3 T states: 12(4,3,5) 4 MHz E.T.: 3.00

If the condition is not met:

M cycles: 7 T states: 7(4,3) 4 MHz E.T.: 1.75

Condition Bits Affected: None

Example:

The Carry Flag is reset and it is required to repeat the jump instruction. The assembly language statement is:

JR NC,$

The resulting object code and PC after the jump are shown below:

Location	Instruction
480	30 ◊ PC after jump
481	FD (two's complement − 2)
482	— ◊ PC before jump

Note: this instruction would cause an infinite loop in the program.

JR Z,e

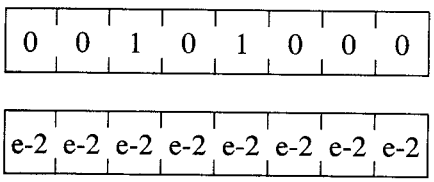
Jump Relative

Operation: Z = 0, continue
If Z = 1, PC ◊ PC + e

Format:

Mnemonic: JR **Operands:** Z, e

Object Code:



28

Description:

This instruction provides for conditional branching to other segments of a program depending on the results of a test on the Zero Flag. If the flag is equal to a '1,' the value of the displacement e is added to the Program Counter (PC) and the next instruction is fetched from the location designated by the new contents of the PC. The jump as measured from the address of the instruction opcode has a range of −126 to +129 bytes. The assembler automatically adjusts for the twice incremented PC.

If the Zero Flag is equal to a '0,' the next instruction to be executed is taken from the location following this instruction.

If the condition is met:

M cycles: 3 T states: 12(4,3,5) 4 MHz E.T.: 3.00

If the condition is not met:

M cycles: 2 T states: 7(4,3) 4 MHz E.T.: 1.75

Condition Bits Affected: None

Example:

The Zero Flag is set and it is required to jump forward five locations from address 300. The following assembly language statement is used:

JR Z, $+5

The resulting object code and final PC value is shown below:

Location	Instruction
300	28
301	03
302	—◊PC before jump
303	—
304	—
305	—◊PC after jump

JR NZ,e

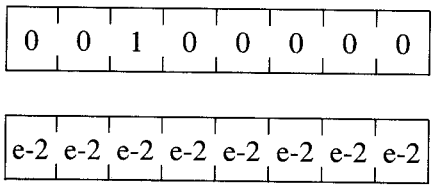
Jump Relative

Operation: If Z = 1, continue
If Z = 0, PC◊PC + e

Format:

Mnemonic: JR **Operands:** NZ, e

Object Code:



Description:

This instruction provides for conditional branching to other segments of a program depending on the results of a test on the Zero Flag. If the flag is equal to a '0,' the value of the displacement e is added to the Program Counter (PC) and the next instruction is fetched from the location designated by the new contents of the PC. The jump as measured from the address of the instruction opcode has a range of −126 to +129 bytes. The assembler automatically adjusts for the twice incremented PC.

If the Zero Flag is equal to a '1,' the next instruction to be executed is taken from the location following this instruction.

If the condition is met:

M cycles: 3 T states: 12(4,3,5) 4 MHz E.T.: 3.00

If the condition is not met:

M cycles: 2 T states: 7(4,3) 4 MHz E.T.: 1.75

Condition Bits Affected: None

Example:

The Zero Flag is reset and it is required to jump back four locations from 480. The assembly language statement is:

JR NZ, $-4

The resulting object code and final PC value is shown below:

Location	Instruction
47C	◊PC after jump
47D	—
47E	—
47F	—
480	20
481	FA (two's complement − 6)
482	—◊PC before jump

JP (HL)

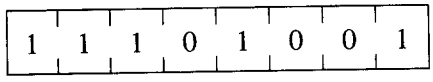
Jump

Operation: PC ◊ HL

Format:

Mnemonic: JP **Operands:** (HL)

Object Code:



E9

Description:

The Program Counter (register pair PC) is loaded with the contents of the HL register pair. The next instruction is fetched from the location designated by the new contents of the PC.

M cycles: 1 T states: 4 4 MHz E.T.: 1.00

Condition Bits Affected: None

Example 1:

If the contents of the Program Counter are 1000H and the contents of the HL register pair are 4800H, after the execution of

JP (HL)

the contents of the Program Counter will be 4800H.

The program will jump to the instruction at address 4800H.

Example 2:

A typical software routine which uses JP (HL) is a jump table lookup program. Assume that n 16-bit addresses are listed in consecutive bytes of memory starting at address TBL. Also assume that the Accumulator contains a number from 0 to n-1 representing the routine to be jumped to.

```
LD      HL, TBL  ; HL points to the first byte in the table.
ADD     A, A     ; double A
LD      DE, 0
LD      E, A
ADD     HL, DE   ; if A originally contained 5, then HL now points to the
                   5th address in the table
LD      E, (HL)
INC     HL
LD      D, (HL)  ; DE now contains the 5th address of the table
LD      HL, DE   ; HL now contains the 5th address of the table
JP      (HL)
```

JP (IX)

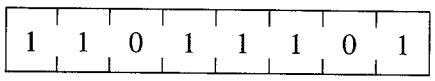
JumP

Operation: PC ◊ IX

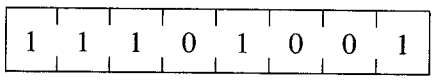
Format:

Mnemonic: JP **Operands:** (IX)

Object Code:



DD



E9

Description:

The Program Counter (register pair PC) is loaded with the contents of the IX Register Pair (Index Register IX). The next instruction is fetched from the location designated by the new contents of the PC.

M cycles: 2 T states: 8(4,4) 4 MHz E.T.: 2.00

Condition Bits Affected: None

Example:

If the contents of the Program Counter are 1000H, and the contents of the IX Register Pair are 4800H, after the execution of

JP (IX)

the contents of the Program Counter will be 4800H.

JP (IY)

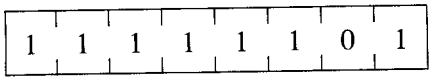
JumP

Operation: PC ◊ IY

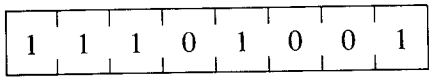
Format:

Mnemonic: JP **Operands:** (IY)

Object Code:



FD



E9

Description:

The Program Counter (register pair PC) is loaded with the contents of the IY Register Pair (Index Register IY). The next instruction is fetched from the location designated by the new contents of the PC.

M cycles: 2 T states: 8(4,4) 4 MHz E.T.: 2.00

Condition Bits Affected: None

Example:

If the contents of the Program Counter are 1000H and the contents of the IY Register Pair are 4800H, after the execution of

JP (IY)

the contents of the Program Counter will be 4800H.

DJNZ e

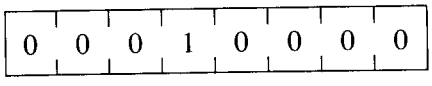
Decrement Jump Not Zero

Operation:

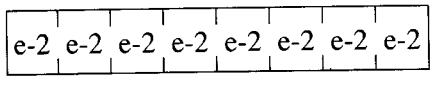
Format:

Mnemonic: DJNZ **Operands:** e

Object Code:



10



Description:

The instruction is similar to the conditional jump instructions except that a register value is used to determine branching. The B register is decremented and if a non zero value remains, the value of the displacement e is added to the Program Counter (PC). The next instruction is fetched from the location

designated by the new contents of the PC. The jump is measured from the address of the instruction opcode has a range of −126 to +129 bytes. The assembler automatically adjusts for the twice incremented PC.

If the result of decrementing leaves B with a zero value, the next instruction to be executed is taken from the location following this instruction.

If B ≠ Ø:

M cycles: 3 T states: 13(5,3,5) 4 MHz E.T.: 3.25

If B = Ø:

M cycles: 2 T states: 8(5,3) 4 MHz E.T.: 2.ØØ

Condition Bits Affected: None

Example:

A typical software routine is used to demonstrate the use of the DJNZ instruction. This routine moves a line from an input buffer (INBUF) to an output buffer (OUTBUF). It moves the bytes until it finds a carriage return, or until it has moved 80 bytes, whichever occurs first.

```
        LD      B, 8Ø       ; Set up counter
        LD      HL, Inbuf   ; Set up pointers
        LD      DE, Outbuf
LOOP:   LD      A, (HL)     ; Get next byte from
                            ; input buffer
        LD      (DE), A     ; Store in output buffer
        CP      ØDH         ; Is it a CR?
        JR      Z, DONE     ; Yes finished
        INC     HL          ; Increment pointers
        INC     DE
        DJNZ    LOOP        ; Loop back if 8Ø
                            ; bytes have not
                            ; been moved
DONE:
```

Call and Return Group

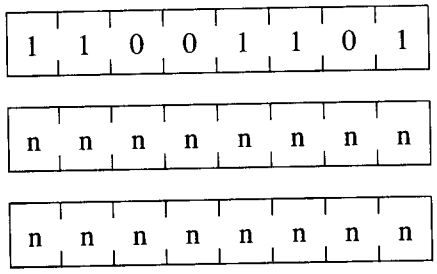
CALL nn

Operation: $(SP-1) \Leftarrow PC_H$, $(SP-2) \Leftarrow PC_L$, $PC \Leftarrow nn$

Format:

Mnemonic: CALL **Operands:** nn

Object Code:



Note: The first of the two n operands in the assembled object code above is the least significant byte of a two-byte memory address.

Description:

After pushing the current contents of the Program Counter (PC) onto the top of the external memory stack, the operands nn are loaded into PC to point to the address in memory where the first opcode of a subroutine is to be fetched. (At the end of the subroutine, a RETurn instruction can be used to return to the original program flow by popping the top of the stack back into PC.) The push is accomplished by first decrementing the current contents of the Stack Pointer (register pair SP), loading the high-order byte of the PC contents into the memory address now pointed to by the SP; then decrementing SP again, and loading the low-order byte of the PC contents into the top of stack. Note: Because this is a three-byte instruction, the Program Counter will have been incremented by three before the push is executed.

M cycles: 5 T states: 17(4,3,4,3,3) 4 MHz E.T.: 4.25

Condition Bits Affected: None

Example:

If the contents of the Program Counter are 1A47H, the contents of the Stack Pointer are 3002H, and memory locations have the contents:

Location	Contents
1A47H	CDH
1A48H	35H
1A49H	21H

then if an instruction fetch sequence begins, the three-byte instruction CD3521H will be fetched to the CPU for execution. The mnemonic equivalent of this is

CALL 2135H

After the execution of this instruction, the contents of memory address 3001H will be 1AH, the contents of address 3000H will be 4AH, the contents of the Stack Pointer will be 3000H, and the contents of the Program Counter will be 2135H, pointing to the address of the first opcode of the subroutine now to be executed.

Before:

Stack Pointer	Address	Stack
3002	3002	50
	3003	1B
	3004	3C

Program Counter

1A47

After CALL 2135H:

Stack Pointer	Address	Stack
3000	3000	4A
	3001	1A
	3002	50
	3003	1B

Program Counter

2135

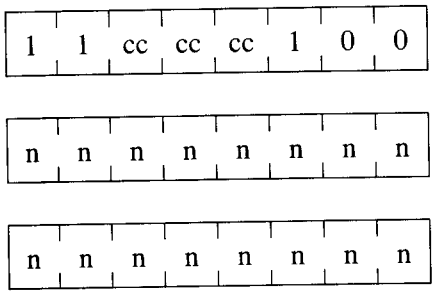
CALL cc,nn

Operation: IF cc TRUE: $(SP-1) \leftarrow PC_H$
$(SP-2) \leftarrow PC_L$, $PC \leftarrow nn$

Format:

Mnemonic: CALL **Operands:** cc, nn

Object Code:



Note: The first of the two n operands in the assembled object code above is the least significant byte of the two-byte memory address.

Description:

If condition cc is true, this instruction pushes the current contents of the Program Counter (PC) onto the top of the external memory stack, then loads the operands nn into PC to point to the address in memory where the first opcode of a subroutine is to be fetched. (At the end of the subroutine, a RETurn instruction can be used to return to the original program flow by popping the top of the stack back into PC.) If condition cc is false, the Program Counter is incremented as usual, and the program continues with the next sequential instruction. The stack push is accomplished by first decrementing the current contents of the Stack Pointer (SP), loading the high-order byte of the PC contents into the memory address now pointed to by SP, then decrementing SP again, and loading the low-order byte of the PC contents into the top of the stack. **Note:** Because this is a three-byte instruction, the Program Counter will have been incremented by three before the push is executed. Condition cc is programmed as one of eight status bits which corresponds to condition bits in the Flag Register (register F). Those eight status bits are defined in the table below, which also specifies the corresponding cc bit fields in the assembled object code:

cc	Condition	Relevant Flag
000	NZ non-zero	Z (=0)
001	Z zero	Z (=1)
010	NC non-carry	C (=0)
011	C carry	C (=1)
100	PO parity odd	P/V (=0)
101	PE parity even	P/V (=1)
110	P sign positive	S (=0)
111	M sign negative	S (=1)

If cc is true:
M cycles: 5 T states: 17(4,3,4,3,3) 4 MHz E.T.: 4.25
If cc is false:
M cycles: 3 T states: 10(4,3,3) 4 MHz E.T.: 2.50

Condition Bits Affected: None

Example:

If the C Flag in the F register is reset, the contents of the Program Counter are 1A47H, the contents of the Stack Pointer are 3002H, and memory locations have the contents:

Location	Contents
1A47H	D4H
1A48H	35H
1A49H	21H

then if an instruction fetch sequence begins, the three-byte instruction D43521H will be fetched to the CPU for execution. The mnemonic equivalent of this is

CALL NC, 2135H

After the execution of this instruction, the contents of memory address 3001H will be 1AH, the contents of address 3000H will be 4AH, the contents of the Stack Pointer will be 3000H, and the contents of the Program Counter will be 2135H, pointing to the address of the first opcode of the subroutine now to be executed.

RET

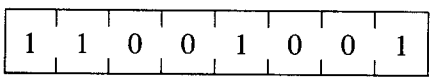
RETurn

Operation: $PC_L \Leftrightarrow (SP)$, $PC_H \Leftrightarrow (SP+1)$

Format:

Mnemonic: RET **Operands:**

Object Code:



C9

Description:

Control is returned to the original program flow by popping the previous contents of the Program Counter (PC) off the top of the external memory stack, where they were pushed by the CALL instruction. This is accomplished by first loading the low-order byte of the PC with the contents of the memory address

pointed to by the Stack Pointer (SP), then incrementing the SP and loading the high-order byte of the PC with the contents of the memory address now pointed to by the SP. (The SP is now incremented a second time.) On the following machine cycle the CPU will fetch the next program opcode from the location in memory now pointed to by the PC.

M cycles: 3 T states: 10(4,3,3) 4 MHz E.T.: 2.50

Condition Bits Affected: None

Example:

If the contents of the Program Counter are 3535H, the contents of the Stack Pointer are 2000H, the contents of memory location 2000H are B5H, and the contents of memory location 2001H are 18H, then after the execution of

RET

the contents of the Stack Pointer will be 2002H and the contents of the Program Counter will be 18B5H, pointing to the address of the next program opcode to be fetched.

Before:

Program Counter	Address	Stack
3535	2000	B5
	2001	18
	2002	2E
	2003	30

Stack Pointer
2000

After RET:

Program Counter	Address	Stack
18B5	2002	2E
	2003	30

Stack Pointer
2002

RET cc

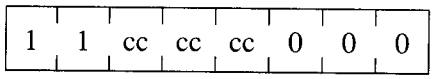
RETurn

Operation: IF cc TRUE: $PC_L \Diamond (SP)$, $PC_H \Diamond (SP+1)$

Format:

Mnemonic: RET **Operands:** cc

Object Code:



Description:

If condition cc is true, control is returned to the original program flow by popping the previous contents of the Program Counter (PC) off the top of the external memory stack, where they were pushed by the CALL instruction. This is accomplished by first loading the low-order byte of the PC with the contents of the memory address pointed to by the Stack Pointer (SP), then incrementing the SP, and loading the high-order byte of the PC with the contents of the memory address now pointed to by the SP. (The SP is now incremented a second time.) On the following machine cycle the CPU will fetch the next program opcode from the location in memory now pointed to by the PC. If condition cc is false, the PC is simply incremented as usual, and the program continues with the next sequential instruction. Condition cc is programmed as one of eight status bits which correspond to condition bits in the Flag Register F). These eight status bits are defined in the table below, which also specifies the corresponding cc bit fields in the assembled object code.

cc	Condition	Relevant Flag
000	NZ non-zero	Z (=0)
001	Z zero	Z (=1)
010	NC non-carry	C (=0)
011	C carry	C (=1)
100	PO parity odd	P/V (=0)
101	PE parity even	P/V (=1)
110	P sign positive	S (=0)
111	M sign negative	S (=1)

If cc is true:

M cycles: 3 T states: 11(5,3,3) 4 MHz E.T.: 2.75

If cc is false:

M cycles: 1 T states: 5 4 MHz E.T.: 1.25

Condition Bits Affected: None

Example:

If the S flag in the F register is set, the contents of the Program Counter are 3535H, the contents of the Stack Pointer are 2000H, the contents of memory location 2000H are B5H, and the contents of memory location 2001H are 18H, then after the execution of

```
RET  M
```

the contents of the Stack Pointer will be 2002H and the contents of the Program Counter will be 18B5H, pointing to the address of the next program opcode to be fetched.

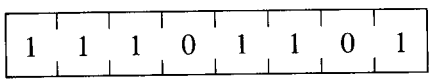
RETI

Operation: Return from interrupt

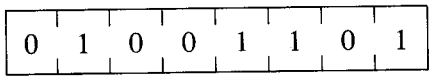
Format:

Mnemonic: RETI **Operands:**

Object Code:



ED



4D

Description:

This instruction is used at the end of an interrupt service routine to:

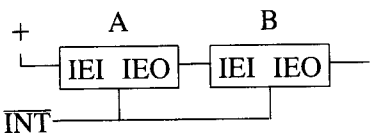
1. Restore the contents of the Program Counter (PC) (analogous to the RET instruction).
2. To signal an I/O device that the interrupt routine has been completed. The RETI instruction facilitates the nesting of interrupts, allowing higher priority devices to suspend service of lower priority service routines. This instruction also resets the IFF1 and IFF2 flip flops.

M cycles: 4 T states: 14(4,4,3,3) 4 MHz E.T.: 3.5Ø

Condition Bits Affected: None

Example:

Given: Two interrupting devices, A and B, connected in a daisy chain configuration with A having a higher priority than B.



B generates an interrupt and is acknowledged. (The interrupt enable out, IEO, of B goes low, blocking any lower priority devices from interrupting while B is being serviced). Then A generates an interrupt, suspending service of B. (The

IEO of A goes 'low' indicating that a higher priority device is being serviced.) The A routine is completed and a RETI is issued resetting the IEO of A, allowing the B routine to continue. A second RETI is issued on completion of the B routine and the IEO of B is reset (high), allowing lower priority devices interrupt access.

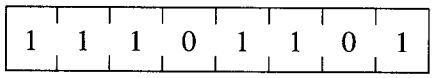
RETN

Operation: Return from non maskable interrupt

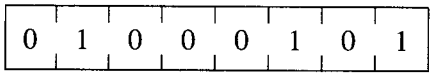
Format:

Mnemonic: RETN **Operands:**

Object Code:



ED



45

Description:

Used at the end of a service routine for a non maskable interrupt, this instruction executes an unconditional return which functions identically to the RET instruction. That is, the previously stored contents of the Program Counter (PC) are popped off the top of the external memory stack; the low-order byte of PC is loaded with the contents of the memory location pointed to by the Stack Pointer (SP), SP is incremented, the high-order byte of PC is loaded with the contents of the memory location now pointed to by SP, and SP is incremented again. Control is now returned to the original program flow: on the following machine cycle the CPU will fetch the next opcode from the location in memory now pointed to by the PC. Also the state of IFF2 is copied back into IFF1 to the state it had prior to the acceptance of the NMI.

M cycles: 4 T states: 14(4,4,3,3) 4 MHz E.T.: 3.5Ø

Condition Bits Affected: None

Example:

If the contents of the Stack Pointer are 1ØØØH and the contents of the Program Counter are 1A45H when a non maskable interrupt (NMI) signal is received, the CPU will ignore the next instruction and will instead restart to memory address ØØ66H. That is, the current Program Counter contents of 1A45H will be pushed onto the external stack address of ØFFFH and ØFFEH, high order byte first, and

0066H will be loaded onto the Program Counter. That address begins an interrupt service routine which ends with RETN instruction. Upon the execution of RETN, the former Program Counter contents are popped off the external memory stack, low-order first, resulting in a Stack Pointer contents again of 1000H. The program flow continues where it left off with an opcode fetch to address 1A45H.

RST p

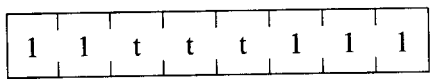
ReSTart

Operation: $(SP-1) \Leftarrow PC_H$, $(SP-2) \Leftarrow PC_L$, $PC_H \Leftarrow O$, $PC_L \Leftarrow P$

Format:

Mnemonic: RST **Operands:** P

Object Code:



Description:

The current Program Counter (PC) contents are pushed onto the external memory stack, and the page zero memory location given by operand p is loaded into the PC. Program execution then begins with the opcode in the address now pointed to by PC. The push is performed by first decrementing the contents of the Stack Pointer (SP), loading the high-order byte of PC into the memory address now pointed to by SP, decrementing SP again, and loading the low-order byte of PC into the address now pointed to by SP. The ReSTart instruction allows for a Call to a subroutine at one of eight addresses as shown in the table below. The operand p is assembled into the object code using the t column of the table. **Note:** Since all addresses are in page zero of memory, the high order byte of PC is loaded with 00H. The number selected from the "p" column of the table is loaded into the low-order byte of PC.

At the end of the subroutine a RETurn instruction can be used to return to the original program by popping the top of the stack back into PC.

P	t
00H	000
08H	001
10H	010
18H	011
20H	100
28H	101
30H	110
38H	111

M cycles: 3 T states: 11(5,3,3) 4 MHz E.T.: 2.75

Example:

If the contents of the Program Counter are 15B3H, after the execution of

RST 18H (Object code 11Ø11111)

the PC will contain ØØ18H, as the address of the next opcode to be fetched, and the top number on the stack will be 15B3H.

Input and Output Group

IN A,(n)

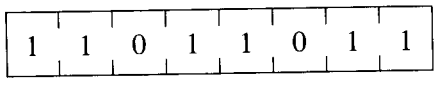
INput

Operation: A ⇦ (n)

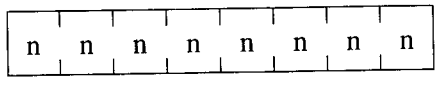
Format:

Mnemonic: IN **Operands:** A, (n)

Object Code:



DB



Description:

The number of the input port is n. Data is input to register A. The operand n is placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. The contents of the Accumulator also appear on the top half (A8 through A15) of the address bus at this time. Then one byte from the selected port is placed on the data bus and written into the Accumulator (register A) in the CPU.

M cycles: 3 T states: 11(4,3,4) 4 MHz E.T.: 2.75

Condition Bits Affected: None

Example:

If the contents of the Accumulator are 23H and the byte 7BH is available at the peripheral device mapped to I/O port address 01H, then after the execution of

IN A,(01H)

the Accumulator will contain 7BH.

IN r,(C)

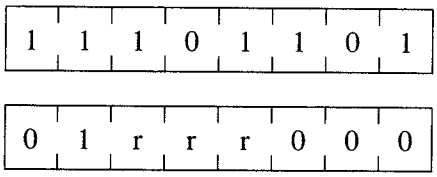
INput

Operation: r ◊ (C)

Format:

Mnemonic: IN **Operands:** r, (C)

Object Code:



Description:

Register C contains the number of the input port. Data is input to register r. The contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. The contents of Register B are placed on the top half (A8 through A15) of the address bus at this time. Then one byte from the selected port is placed on the data bus and written into register r in the CPU. Register r identifies any of the CPU registers shown in the following table, which also shows the corresponding three-bit "r" field for each. The flags will be affected, checking the input data.

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

M cycles: 3 T states: 12(4,4,4) 4 MHz E.T.: 3.00

Condition Bits Affected:

S: Set if input data is negative; reset otherwise
Z: Set if input data is zero; reset otherwise
H: Reset
P/V: Set if parity is even; reset otherwise
N: Reset
C: Not affected

Example:

If the contents of register C are 07H, the contents of register B are 10H, and the byte 7BH is available at the peripheral device mapped to I/O port address 07H, then after the execution of

IN D,(C)

register D will contain 7BH

A typical use of the IN r, (C) instruction is for polled I/O. The following program continually polls or inputs data from port FF until a non-zero number appears. The program then reads in data from port FE. In this application, port FF is used as a data ready signal for port FE.

```
        LD      C, 0FFH       ; C points at port FF
LOOP    IN      B, (C)        ; input port FF to register B
        JR      Z, LOOP       ; continue polling until not zero
        IN      A, (0FEH)     ; input port FE to register A
```

INI

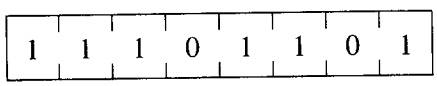
INput & Increment

Operation: (HL)◊(C), B◊B−1, HL◊HL+1

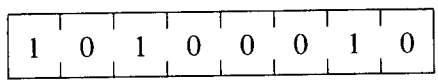
Format:

Mnemonic: INI **Operands:**

Object Code:



ED



A2

Description:

Register C contains the number of the input port. Data input is placed in memory at the address pointed at by HL. The contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B may be used as a byte counter, and its contents are placed on the top half (A8 through A15) of the address bus at this time. Then one byte from the selected port is placed on the data bus and written to the CPU. The contents of the HL register pair are then placed on the address bus and the input byte is written into the corresponding location of memory. Finally the byte counter is decremented and register pair HL is incremented.

M cycles: 4 T states: 16(4,5,3,4) 4 MHz E.T.: 4.00

Condition Bits Affected:

S: Unknown
Z: Set if B − 1 = 0; reset otherwise
H: Unknown
P/V: Unknown
N: Set
C: Not affected

Example:

If the contents of register C are 07H, the contents of register B are 10H, the contents of the HL register pair are 1000H, and the byte 7BH is available at the peripheral device mapped to I/O port address 07H, then after the execution of

INI

memory location 1000H will contain 7BH, the HL register pair will contain 1001H, and register B will contain 0FH.

The following program will input data from input ports 1 through 80 and place the data into a buffer in memory.

```
        LD      B, 80
        LD      C,0
        LD      HL, BUFF
LOOP    INC     C
        INI
        JP      NZ, LOOP
```

INIR

INput Increment & Repeat

Operation: (HL) ◊ (C), B ◊ B − 1, HL ◊ HL + 1

Format:

Mnemonic: INIR **Operands:**

Object Code:

1	1	1	0	1	1	0	1	ED

1	0	1	1	0	0	1	0	B2

Description:

Register C contains the number of the input port. The data input is placed in memory at the address pointed at by the HL register pair. The contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B is used as a byte counter, and its contents are placed on the top half (A8 through A15) of the address bus at this time. Then one byte from the selected port is placed on the data bus and written to the CPU. The contents of the HL register pair are placed on the address bus and the input byte is written into the corresponding location of memory. Then register pair HL is incremented, the byte counter is decremented. If decrementing causes B to go to zero, the instruction is terminated. If B is not zero, the PC is decremented by two and the instruction repeated. Note that if B is set to zero prior to instruction execution, 256 bytes of data will be input. Also interrupts will be recognized after each data transfer.

If B $\neq$ 0:

M cycles: 5 T states: 21(4,5,3,4,5) 4 MHz E.T.: 5.25

If B = 0:

M cycles: 4 T states: 16(4,5,3,4) 4 MHz E.T.: 4.00

Condition Bits Affected:

S: Unknown
Z: Set
H: Unknown
P/V: Unknown
N: Set
C: Not affected

Example:

If the contents of register C are 07H, the contents of register B are 03H, the contents of the HL register pair are 1000H, and the following sequence of bytes are available at the peripheral device mapped to I/O port of address 07H:

51H
A9H
03H

then after the execution of

INIR

the HL register pair will contain 1003H, register B will contain zero, and memory locations will have contents as follows:

Location	Contents
1000H	51H
1001H	A9H
1002H	03H

Here is a program to input 80 bytes from I/O port number FF and put them into an 80-byte buffer starting at address BUFF.

```
LD      HL, BUFF    ; HL points at first byte of buffer
LD      B, 80       ; load byte counter
LD      C, OFFH     ; port FF
IN IR               ; input 80 bytes
```

Note: this assumes that the input port can be synchronized with the input instructions.

IND

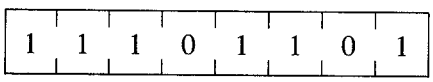
INput & Decrement

Operation: (HL) ◊ (C), B ◊ B − 1, HL ◊ HL − 1

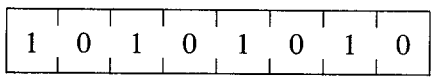
Format:

Mnemonic: IND **Operands:**

Object Code:



ED



AA

Description:

The contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B may be used as a byte counter, and its contents are placed on the top half (A8 through A15) of the address bus at this time. Then one byte from the selected port is placed on the data bus and written to the CPU. The contents of the HL register pair are placed on the address bus and the input byte is written into the corresponding location of memory. Finally the byte counter and register pair HL are decremented.

M cycles: 4 T states: 16(4,5,3,4) 4 MHz E.T.: 4.00

Condition Bits Affected:

S: Unknown
Z: Set if B − 1 = 0; reset otherwise
H: Unknown
P/V: Unknown
N: Set
C: Not affected

Example:

If the contents of register C are 07H, the contents of register B are 10H, the contents of the HL register pair are 1000H, and the byte 7BH is available at the peripheral device mapped to I/O port address 07H, then after the execution of

IND

memory location 1000H will contain 7BH, the HL register pair will contain 0FFFH, and register B will contain 0FH.

INDR

INput Decrement & Repeat

Operation: (HL) ◊ (C), B ◊ B − 1, HL ◊ HL − 1

Format:

Mnemonic: INDR **Operands:**

Object Code:

1	1	1	0	1	1	0	1	ED

1	0	1	1	1	0	1	0	BA

Description:

The contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B is used as a byte counter, and its contents are placed on the top half (A8 through A15) of the address bus at this time. Then one byte from the selected port is placed on the data bus and written to the CPU. The contents of the HL register pair are placed on the address bus and the input byte is written into the corresponding location of memory. Then HL and the byte counter are decremented. If decrementing causes B to go to zero, the instruction is terminated. If B is not zero, the PC is decremented by two and the instruction repeated. Note that if B is set to zero prior to instruction execution, 256 bytes of data will be input. Also interrupts will be recognized after each data transfer.

If B ≠ 0:

M cycles: 5 T states: 21(4,5,3,4,5) 4 MHz E.T.: 5.25

If B = 0:

M cycles: 4 T states: 16(4,5,3,4) 4 MHz E.T.: 4.00

Condition Bits Affected:

S: Unknown
Z: Set
H: Unknown
P/V: Unknown
N: Set
C: Not affected

Example:

If the contents of register C are 07H, the contents of register B are 03H, the contents of the HL register pair are 1000H, and the following sequence of bytes are available at the peripheral device mapped to I/O port address 07H:

51H
A9H
03H

then after the execution of

INDR

the HL register pair will contain 0FFDH, register B will contain zero, and memory locations will have contents as follows:

Location	Contents
0FFEH	03H
0FFFH	A9H
1000H	51H

OUT (n),A

OUTput

Operation: (n) ⇦ A

Format:

Mnemonic: OUT **Operands:** (n), A

Object Code:

1	1	0	1	0	0	1	1

D3

n	n	n	n	n	n	n	n

Description:

The operand n is placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. The contents of the Accumulator (register A) also appear on the top half (A8 through A15) of the address bus at this time. Then the byte contained in the Accumulator is placed on the data bus and written into the selected peripheral device.

M cycles: 3 T states: 11(4,3,4) 4 MHz E.T.: 2.75

Condition Bits Affected: None

Example:

If the contents of the Accumulator are 23H, then after the execution of

OUT 01H,A

the byte 23H will have been written to the peripheral device mapped to I/O port address 01H.

OUT (C),r

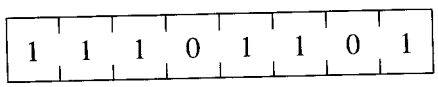
OUTput

Operation: (C) ⇦ r

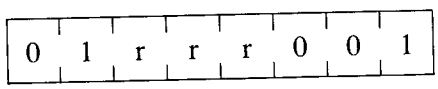
Format:

Mnemonic: OUT **Operands:** (C), r

Object Code:



ED



Description:

The contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. The contents of Register B are placed on the top half (A8 through A15) of the address bus at this time. Then the byte contained in register r is placed on the data bus and written into the selected peripheral device. Register r identifies any of the CPU registers shown in the following table, which also shows the corresponding three-bit "r" field for each which appears in the assembled object code:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

M cycles: 3 T states: 12(4,4,4) 4 MHz E.T.: 3.00

Condition Bits Affected: None

Example:

If the contents of register C are 01H and the contents of register D are 5AH, after the execution of

OUT (C),D

the byte 5AH will have been written to the peripheral device mapped to I/O port address 01H.

OUTI

OUTput & Increment

Operation: (C) ◊ (HL), B ◊ B − 1, HL ◊ HL + 1

Format:

Mnemonic: OUTI **Operands:**

Object Code:

1	1	1	0	1	1	0	1	ED

1	0	1	0	0	0	1	1	A3

Description:

The contents of the HL register pair are placed on the address bus to select a location in memory. The byte contained in this memory location is temporarily stored in the CPU. Then, after the byte counter (B) is decremented, the contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B may be used as a byte counter, and its decremented value is placed on the top half (A8 through

A15) of the address bus. The byte to be output is placed on the data bus and written into selected peripheral device. Finally the register pair HL is incremented.

M cycles: 4 T states: 16(4,5,3,4) 4 MHz E.T.: 4.00

Condition Bits Affected:

S: Unknown
Z: Set if B − 1 = 0; reset otherwise
H: Unknown
P/V: Unknown
N: Set
C: Not affected

Example:

If the contents of register C are 07H, the contents of register B are 10H, the contents of the HL register pair are 1000H, and the contents of memory address 1000H are 59H, then after the execution of

OUTI

register B will contain 0FH, the HL register pair will contain 1001H, and the byte 59H will have been written to the peripheral device mapped to I/O port address 07H.

OTIR

OuTput Increment & Repeat

Operation: (C) ◊ (HL), B ◊ B − 1, HL ◊ HL + 1

Format:

Mnemonic: OTIR **Operands:**

Object Code:

1	1	1	0	1	1	0	1	ED
1	0	1	1	0	0	1	1	B3

Description:

The contents of the HL register pair are placed on the address bus to select a location in memory. The byte contained in this memory location is temporarily stored in the CPU. Then, after the byte counter (B) is decremented, the contents of register C are placed on the bottom half (A0 through A7) of the address bus

to select the I/O device at one of 256 possible ports. Register B may be used as a byte counter, and its decremented value is placed on the top half (A8 through A15) of the address bus at this time. Next the byte to be output is placed on the data bus and written into the selected peripheral device. Then register pair HL is incremented. If the decremented B register is not zero, the Program Counter (PC) is decremented by two and the instruction is repeated. If B has gone to zero, the instruction is terminated. Note that if B is set to zero prior to instruction execution, the instruction will output 256 bytes of data. Also, interrupts will be recognized after each data transfer.

If B ≠ 0:

M cycles: 5 T states: 21(4,5,3,4,5) 4 MHz E.T.: 5.25

If B = 0:

M cycles: 4 T states: 16(4,5,3,4) 4 MHz E.T.: 4.00

Condition Bits Affected:

S: Unknown
Z: Set
H: Unknown
P/V: Unknown
N: Set
C: Not affected

Example:

If the contents of register C are 07H, the contents of register B are 03H, the contents of the HL register pair are 1000H, and memory locations have the following contents:

Location	Contents
1000H	51H
1001H	A9H
1002H	03H

then after the execution of

OTIR

the HL register pair will contain 1003H, register B will contain zero, and a group of bytes will have been written to the peripheral device mapped to I/O port address 07H in the following sequence:

51H
A9H
03H

OUTD

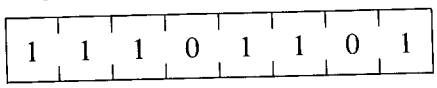
OUTput & Decrement

Operation: (C) ◊ (HL), B ◊ B − 1, HL ◊ HL − 1

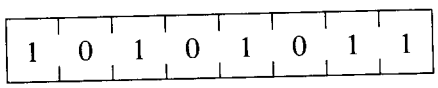
Format:

Mnemonic: OUTD **Operands:**

Object Code:



ED



AB

Description:

The contents of the HL register pair are placed on the address bus to select a location in memory. The byte contained in this memory location is temporarily stored in the CPU. Then, after the byte counter (B) is decremented, the contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B may be used as a byte counter, and its decremented value is placed on the top half (A8 through A15) of the address bus at this time. Next the byte to be output is placed on the data bus and written into the selected peripheral device. Finally the register pair HL is incremented.

M cycles: 4 T states: 16(4,5,3,4) 4 MHz E.T.: 4.00

Condition Bits Affected:

S: Unknown
Z: Set if B − 1 = 0; reset otherwise
H: Unknown
P/V: Unknown
N: Set
C: Not affected

Example:

If the contents of register C are 07H, the contents of register B are 10H, the contents of the HL register pair are 1000H, and the contents of memory location 1000H are 59H, after the execution of

OUTD

register B will contain 0FH, the HL register pair will contain 0FFFH, and the byte 59H will have been written to the peripheral device mapped to I/O port address 07H.

OTDR

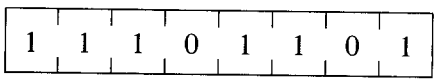
OUTput Decrement & Repeat

Operation: (C) ◊ (HL), B ◊ B − 1, HL ◊ HL − 1

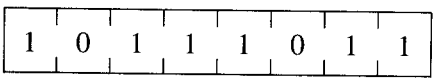
Format:

Mnemonic: OTDR **Operands:**

Object Code:



ED



BB

Description:

The contents of the HL register pair are placed on the address bus to select a location in memory. The byte contained in this memory location is temporarily stored in the CPU. Then, after the byte counter (B) is decremented, the contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B may be used as a byte counter, and its decremented value is placed on the top half (A8 through A15) of the address bus at this time. Next the byte to be output is placed on the data bus and written into the selected peripheral device. Then register pair HL is decremented and if the decremented B register is not zero, the Program Counter (PC) is decremented by 2 and the instruction is repeated. If B has gone to zero, the instruction is terminated. Note that if B is set to zero prior to instruction execution, the instruction will output 256 byte of data. Also, interrupts will be recognized after each data transfer.

If B ≠ 0:

M cycles: 5 T states: 21(4,5,3,4,5) 4 MHz E.T.: 5.25

If B = 0:

M cycles: 4 T states: 16(4,5,3,4) 4 MHz E.T.: 4.00

Condition Bits Affected:

S: Unknown
Z: Set
H: Unknown
P/V: Unknown
N: Set
C: Not affected

Example:

If the contents of register C are 07H, the contents of register B are 03H, the contents of the HL register pair are 1000H, and memory locations have the following contents:

Location	Contents
0FFEH	51H
0FFFH	A9H
1000H	03H

then after the execution of

OTDR

the HL register pair will contain 0FFDH, register B will contain zero, and a group of bytes will have been written to the peripheral device mapped to I/O port address 07H in the following sequence:

03H
A9H
51H