TOSHIBA

TLCS-870/C Series

Instruction Set

TOSHIBA CORPORATION

TLCS-870/C SERIES INSTRUCTION SET

The basic machine instructions in the TLCS-870/C series of microcomputers consist of 132 types, providing a total of 731 instructions. The table below shows these instructions classified by type. The instructions in the TLCS-870/C series vary in length from one byte to a maximum of five bytes. The frequently used instructions have their object code length reduced, making it possible to create memory-efficient programs.

The TLCS-870/C series offers a simple instruction architecture based on a memory-mapped I/O system. Although only 42 kinds of mnemonics are available, it supports as many as 17 addressing modes, allowing for powerful memory manipulation.

In addition to the basic machine instructions, the TLCS-870/C series offers assembler-based extended machine instructions in order to increase coding efficiency.

Table 1. Instruction Set

Transfer, Exchange	8-bit Data Transfer, Exchange		7 types	49 instructions
	16-bit Data Transfer, Exchange		7	43
	Flag Manipulation		4	4
	SP Manipulation		1	2
	Push, Pop		4	6
Operation	8-bit Operation	Compare	4	29
		Increment, Decrement	4	28
		Arithmetic	16	116
		Logical	12	87
		Decimal Adjust	2	2
	16-bit Operation	Compare	3	15
		Increment, Decrement	2	2
		Arithmetic	12	60
		Logical	9	45
		2's Complement	1	1
	Multiply, Divide		2	2
Shift, Rotate	Shift	8 bit (Logical)	2	2
		16-bit (Arithmetic)	2	2
	Rotate	8-bit	2	2
Nibble Manipulation	Swap, Nibble Rotate		3	27
Bit Manipulation	Set, Clear, Invert, Transfer, Exchange, Operate		18	162
Branch	Jump		6	24
	Call		4	16
	Return		3	3
Others	Software Interrupt, Other		2	2
Total			132 types	731 instructions

Table 2. Addressing Modes

Register Indirect	7 types
Direct	2
Register	1
Immediate	1
Relative	2
Absolute	1
Vector	1
Direct Bit	1
Register Indirect Bit	1
Total	17

1.1 Explanation of Symbols

The symbols shown below are used in the explanation of instructions and addressing modes in the pages to follow.

Symbol	Description	Symbol	Description
A	A register (8-bit accumulator)	mn	16-bit immediate data
W	W register	d	Signed 5-bit or 8-bit displacement (−16 to +15, −128 to +127)
B	B register		
C	C register	x, y	8-bit direct address (0000 to $00FF_H$)
D	D register	vw, uz	16-bit direct address (0000 to $FFFF_H$)
E	E register		
H	H register	(XX)	Memory contents at the address specified by XX
L	L register		
WA	WA register (16-bit accumulator)	(XX+1, XX)	Memory contents of two consecutive bytes beginning with the address specified by XX
BC	BC register pair		
DE	DE register pair	b	Bit number (0 to 7)
HL	HL register pair	.b	Content of bit specified by b
IX	IX register	←	Transfer
IY	IY register	↔	Exchange
PC	Program counter	+	Add
SP	Stack pointer	−	Subtract
PSW	Program status word	×	Multiply
JF	Jump status flag	÷	Divide
ZF	Zero flag	∧	Bitwise AND
CF	Carry flag (1-bit accumulator)	∨	Bitwise OR
HF	Half carry flag	∀	Bitwise exclusive OR
SF	Sign flag	null	No operation (Goes to instruction at the next address without performing anything.)
VF	Overflow flag		
$\overline{CF}$	Inverse of carry flag content	$	Start address of instruction (Program counter content under instruction execution becomes $+2 or $+3.)
IMF	Interrupt master enable flag		
r, g	8-bit register (See Table Ⓐ)		
rr, gg	16-bit register (See Table Ⓑ)	(src)	Source memory
n	4-bit or 8-bit immediate data	(dst)	Destination memory

r, g	8-bit Register
0	A
1	W
2	C
3	B
4	E
5	D
6	L
7	H

Table Ⓐ

rr, gg	16-bit Register
0	WA
1	BC
2	DE
3	HL
4	IX
5	IY
6	SP
7	HL

Table Ⓑ

Flag Setting Conditions	
*	The value specified by the operation is set.
Z	Zero detection information is set. • Transfer: When the 8-bit source data is 00_H, the flag is set to 1. Otherwise, the flag is set to 0. • Exchange: When the content of g or (src) before the exchange is 00_H, the flag is set to 1. Otherwise, the flag is set to 0. • Operation: When the result of operation is 00_H (8-bit operation) or 0000_H (16-bit operation), the flag is set to 1. Otherwise, the flag is set to 0. However, for multiply operation this applies when the 8 high-order bits of the product are 00_H; for divide operation this applies when the remainder is 00_H. When the result is other than 00_H, the flag is set to 0. • Shift, Rotate: When the register content after being shifted or rotated is 00_H, the flag is set to 1. Otherwise, the flag is set to 0. • Other: When Z appears in the JF column, it means that the value set in ZF is also set in JF.
C	Carry information is set. • Addition: When the operation results in a carry out of the most significant bit, the flag is set. • Subtraction: When the operation results in a borrow into the most significant bit, the flag is set. • Dividision: When the divisor is 00_H or the quotient is equal to or greater than 100_H, the flag is set to 1. Otherwise, the flag is set to 0. • Other: When C appears in the JF column, it means that the value set in CF is also set in JF.
$\bar{C}$	An inverse of the value set in CF is set.
H	Half carry information is set. • Addition: When the operation results in a carry out of bit 3, the flag is set. • Subtraction: When the operation results in a borrow into bit 3, the flag is set.
S	Sign information (most significant bit of data) is set.
V	Overflow information is set.
$\bar{J}$	An inverse of the value set in JF is set.
1	A logic 1 is set.
0	A logic 0 is set.
U	An indeterminate value is set.
–	The value before instruction execution is retained, with the flag unchanged.

1.2 Mnemonics

The following describes rules on mnemonics of the TLCS-870/C series instructions.
Each mnemonic consists of an opcode and an operand. (Some instructions do not have the operand.) Leave one or more spaces after the opcode before writing the operand. If the instruction has two or more operands, separate each entry in a row of operands with a comma.
If the instruction has two operands-source and destination, always place the destination operand before the source operand. If there are multiple source operands, write the data to be operated on before the operands.
If the operand includes bit specification, separate the address and the bit specifications with a period.

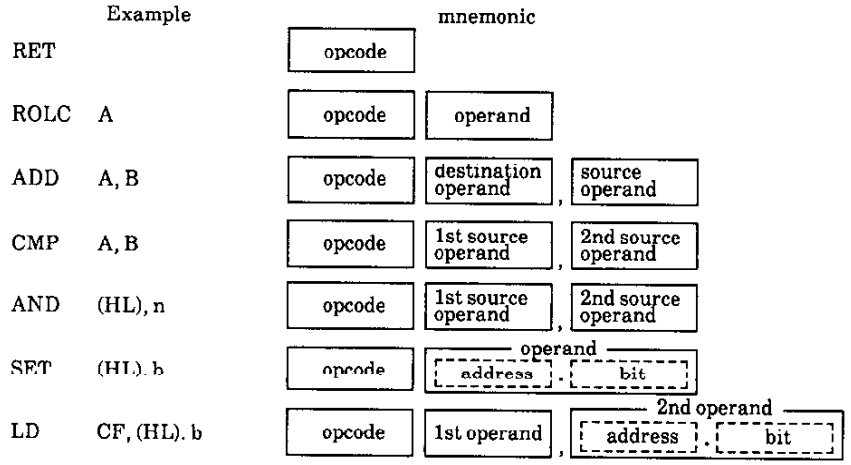


Table 3. Mnemonic List

MNEMONIC	FUNCTION	MNEMONIC	FUNCTION
ADD	Add	NOP	No Operation
ADDC	Add with Carry	OR	Logical OR
AND	Logical AND	POP	Pop up
CALL	Call	PUSH	Push down
CALLV	Vector Call	RET	Return from subroutine
CLR	Clear Bit / Byte	RETI	Return from maskable Interrupt service routine
CMP	Compare	RETN	Return from Non-maskable Interrupt service routine
CPL	1's Complement Bit	ROLC	Rotate Left through Carry
DAA	Decimal Adjust for 8-bit Addition	ROLD	Rotate Left Digit
DAS	Decimal Adjust for 8-bit Subtraction	RORC	Rotate Right through Carry
DEC	Decrement Byte / Word (register)	RORD	Rotate Right Digit
DI*	Disable Maskable Interrupt	SET	Bit Test & Set
DIV	Divide Byte Quotient	SHLC	Logical Shift Left
EI*	Enable Interrupt	SHLCA	Arithmetic Shift Left
INC	Increment Byte / Word (register)	SHRC	Logical Shift Right
J*	Optimized Jump	SHRCA	Arithmetic Shift Right
JP	Absolute Jump	SUB	Subtract
JR	Relative Jump	SUBB	Subtract with Borrow
JRS	Short Relative Jump	SWAP	Swap Nibble
LD	Load Bit / Byte / Word (register) / Effective address	SWI	Software Interrupt
LDW	Load Word (memory)	TEST*	Bit Test
MUL	Multiply	XCH	Exchange
NEG	Negate	XOR	Logical Exclusive OR

Note: Marked by an asterisk (*) are extended assembler machine instructions.

1.3 Object Code Format

The TLCS-870/C series has 1-byte opcode instructions and 2-byte opcode instructions.

(1) Code format of 1-byte opcode instructions

Place the opcode in the first byte and the operand in the second and subsequent bytes. If the operand is 2-byte data, place the lower byte first and then the upper byte. If the operand consists of source and destination and the source is immediate data (e.g., LD(x),n), place the source after the destination.

Example		First byte	Second byte	Third byte	Fourth byte
LD	A, B	opcode			
LD	A, n	opcode	n		
LD	WA, mn	opcode	n	m	
LD	(x), n	opcode	x	n	
LDW	(x), mn	opcode	x	n	m

(2) Code Format of 2-byte Opcode Instructions

Place the first opcode in the first byte and then the operand specified by the first opcode, then place the second opcode and the operand specified by the second opcode.
The first opcode, which is used to specify an addressing mode, is called the "prefix code."
There are two types of prefix: the register prefix to specify a register, and the source/destination memory prefix to specify the source or destination memory address. Note that the first opcode in the five instructions shown below ignore the content of a specified register :

① RETN
② LD PSW, n
③ PUSH PSW
④ POP PSW
⑤ JR M/P/SLT/SGE/SLE/SGT/VS/VC, a

	Example		First byte	Second byte	Third byte	Fourth byte
Register prefix	LD	B, C	1st opcode	2nd opcode		
	ADD	B, n	1st opcode	2nd opcode	n	
Source memory prefix	LD	B, (HL)	1st opcode	2nd opcode		
	LD	B, (x)	1st opcode	x	2nd opcode	
	LD	B, (HL+d)	1st opcode	d	2nd opcode	
Destination memory prefix	LD	(HL+d), n	1st opcode	d	2nd opcode	n

1.4 Addressing Modes

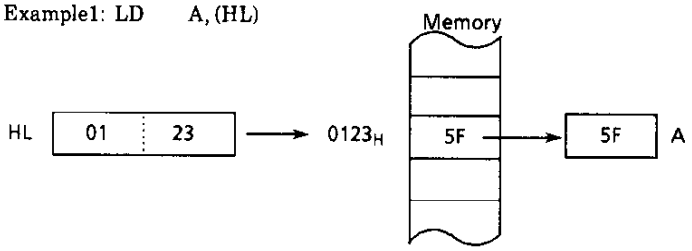
The TLCS-870/C series supports a total of 17 addressing modes (or the methods for specifying addresses). Some instructions use a combination of multiple addressing modes.

(1) Register indirect

① Register indirect (HL), (DE), (IX), (IY)

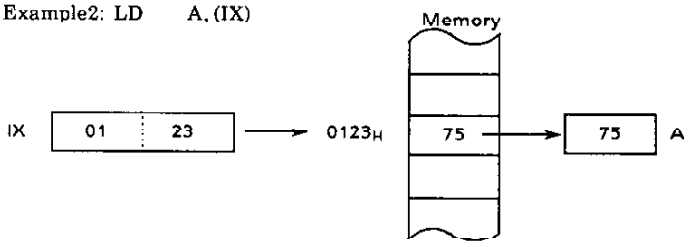
The address is specified by the content of a 16-bit register HL, DE, IX, or IY.

Example1: LD A, (HL)



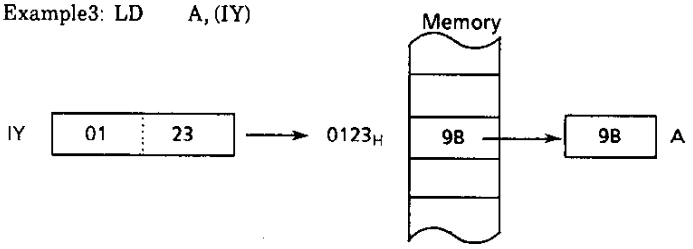
The memory content at the address specified by the content of the HL register, i.e., $5F_H$ at address 0123_H, is loaded into the A register.

Example2: LD A, (IX)



The memory content at the address specified by the content of the IX register, i.e., 75_H at address 0123_H, is loaded into the A register.

Example3: LD A, (IY)

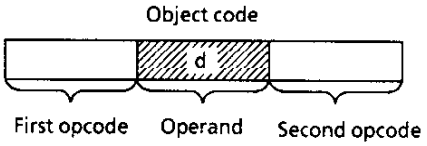


The memory content at the address specified by the content of the IY register, i.e., $9B_H$ at address 0123_H, is loaded into the A register.

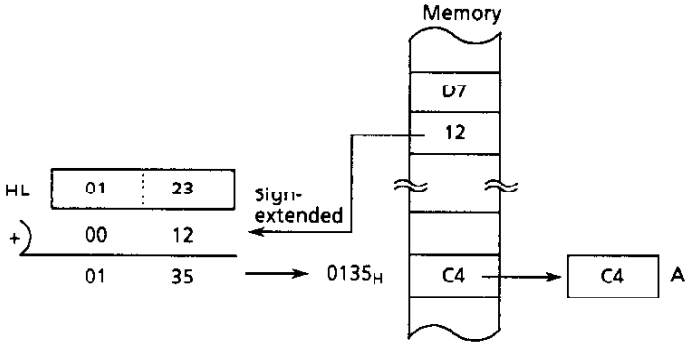
② Register indirect with 8-bit displacement offset (HL+d), (IX+d), (IY+d)

The address is specified by the content of a 16-bit register HL, IX, or IY plus 8-bit displacement 'd' in the object code that is added after being sign-extended (see the table below). Note that the content of the 16-bit register HL, IX, or IY does not change.

Displacement 'd'	Sign-extended Value
$00\sim7F_H$	$0000\sim007F_H$ (0 ~ + 127)
$80\sim FF_H$	$FF80\sim FFFF_H$ (- 128 ~ - 1)

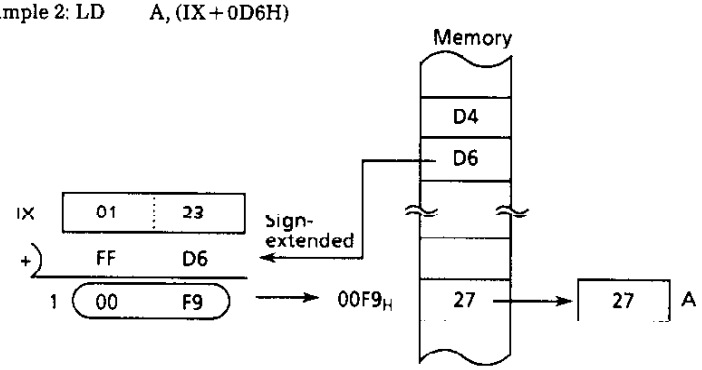


Example 1: LD A, (HL+12H)



The memory content at the address specified by the content (0123_H) of the HL register plus the displacement (12_H) that is added after being sign-extended (0012_H), i.e., $C4_H$ at address 0135_H, is loaded into the A register.

Example 2: LD A, (IX+0D6H)



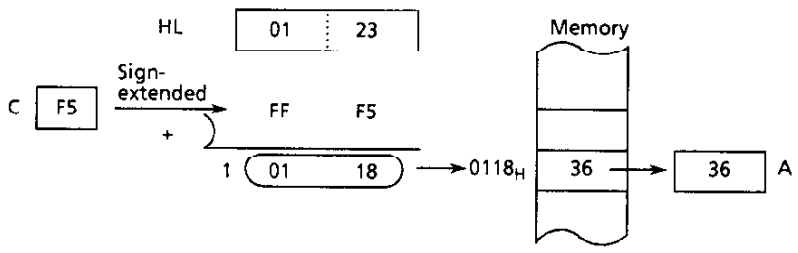
The memory content at the address specified by the content (0123_H) of the IX register plus the displacement ($D6_H$) that is added after being sign-extended ($FFD6_H$), i.e., 27_H at address $00F9_H$, is loaded into the A register.

③ Register index (HL+C)

The address is specified by the content of the HL register plus the content of the C register that is added after being sign-extended (see the table below). Note that the contents of the HL and C registers do not change.

C Register	Sign-extended Value
$00\sim7F_H$	$0000\sim007F_H$ (0 ~ + 127)
$80\sim FF_H$	$FF80\sim FFFF_H$ (- 128 ~ - 1)

Example: LD A, (HL+C)

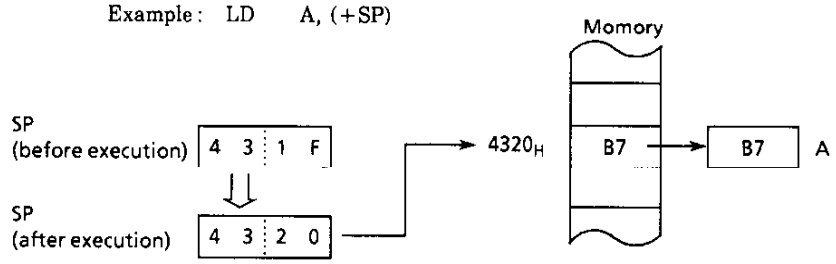


The memory content at the address specified by the content (0123_H) of the HL register plus the content ($F5_H$) of the C register that is added after being sign-extended ($FFF5_H$), i.e., 36_H at address 0118_H, is loaded into the A register.

④ Stack pointer indirect auto increment (+SP)

The content of the SP is incremented. The incremented content of the SP indicates the effective address. Incrementing of the SP does not cause the flag to change.
This addressing mode can only be used for specifying the source memory address (the source of transfer).

Example: LD A, (+SP)

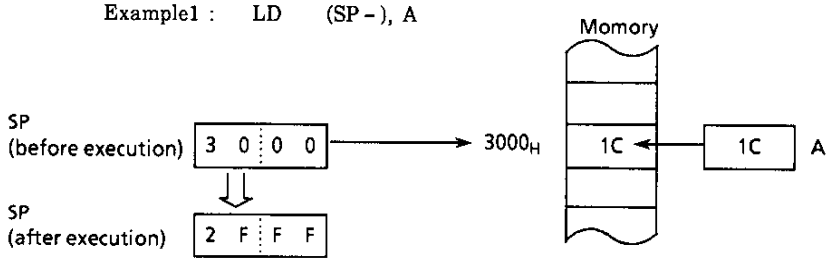


The content of the SP is incremented first. Then the memory content at the address indicated by this incremented value, i.e., $B7_H$ at address 4320_H, is loaded into the A register.

⑤ Stack pointer indirect auto decrement (SP−)

The effective address is indicated by the content of the SP. After data processing, the SP content is automatically decremented. Decrementing of the SP does not cause the flag to change.
This addressing mode can only be used for specifying the destination memory address (the destination of transfer).

Example1 : LD (SP−), A

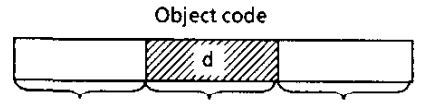


The content IC_H of the A register is stored in memory location at the address indicated by the content of the SP, i.e., address 3000_H. Then the SP is decremented to $2FFF_H$.

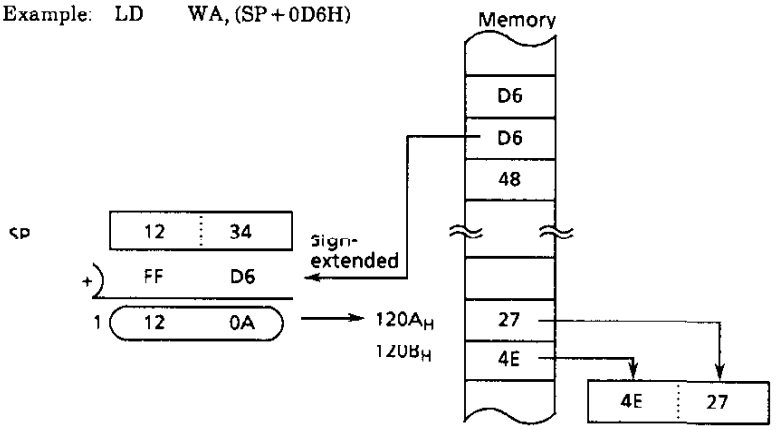
⑥ Stack pointer indirect with 8-bit displacement offset (SP + d)

The effective address is indicated by the content of the stack pointer SP plus 8-bit displacement 'd' in the object code that is added after being sign-extended (see the table below). Note that the SP content does not change.

Displacement 'd'	Sign-extended Value
$00 \sim 7F_H$	$0000 \sim 007F_H$ (0 ~ + 127)
$80 \sim FF_H$	$FF80 \sim FFFF_H$ (− 128 ~ − 1)



Example: LD WA, (SP + 0D6H)



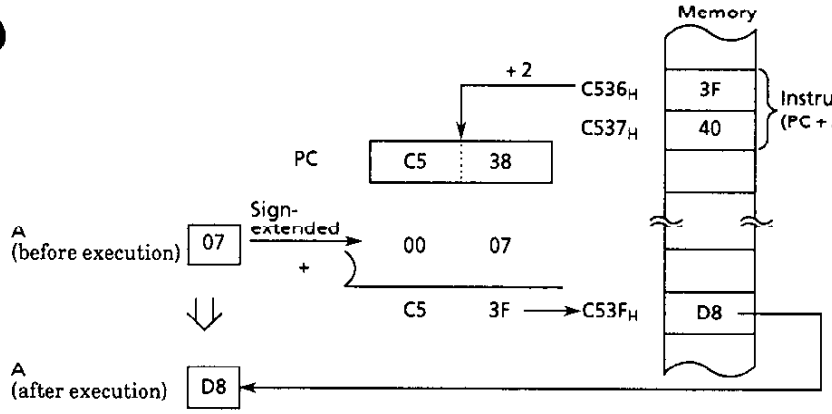
Consecutive 2-byte memory contents beginning with the address specified by the SP content (1234_H) plus the displacement ($D6_H$) that is added after being sign-extended ($FFD6_H$), i.e., $4E27_H$ starting from address $120A_H$, are loaded into the WA register.

⑦ Register offset relative index (PC + A)

The address is specified by the content of the program counter (start address of execution instruction + 2) plus the content of the A register that is added after being sign-extended (see the table below). This addressing mode can only be used to specify the source address (the source of transfer). Use of this addressing mode helps to easily program the code conversion from BCD code to 7-segment code, as well as table lookup, table search, and 'n' ways of multi-direction branch processing.

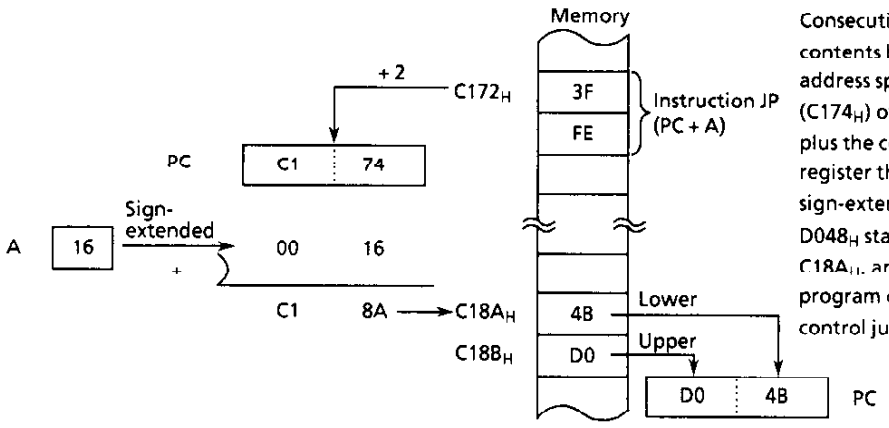
Accumulator	Sign-extended Value
$00 \sim 7F_H$	$0000 \sim 007F_H$ (0 ~ + 127)
$80 \sim FF_H$	$FF80 \sim FFFF_H$ (− 128 ~ − 1)

Example 1: LD A, (PC + A)



The memory content at the address specified by the content ($C538_H$) of the program counter plus the content (07_H) of the A register that is added after being sign-extended (0007_H), i.e., D8H at address $C53F_H$, is loaded into the A register.

Example 2: JP (PC + A)



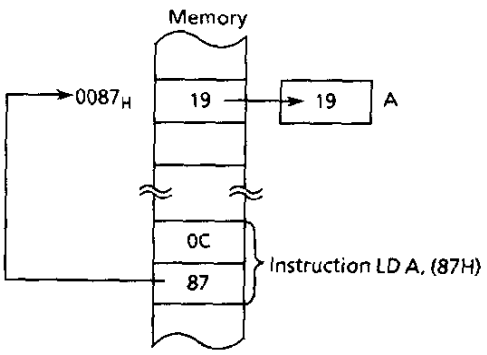
Consecutive 2-byte memory contents beginning with the address specified by the content ($C174_H$) of the program counter plus the content (16_H) of the A register that is added after being sign-extended (0016_H), i.e., $D048_H$ starting from address $C18A_H$, are loaded into the program counter. Consequently, control jumps to address $D048_H$.

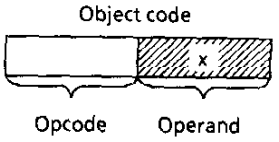
(2) Direct

① 8-bit direct (x)

The address, one in the range of addresses 0000 to 00FFH, is specified directly by the 8-bit value x included in the object code.

Example: LD A, (87H)



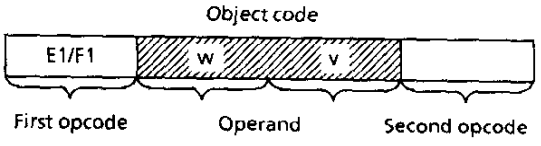


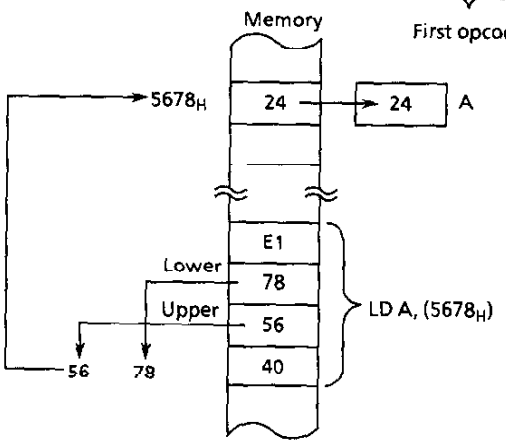
The memory content at the address specified directly by the value x (87_H) of the object code, i.e., 19_H at address 0087_H, is loaded into the A register.

② 16-bit direct (vw)

The address, one in the range of addresses 0000 to FFFFH, is specified directly by the 16-bit value vw included in the object code.

Example: LD A, (5678H)





The memory content at the address specified directly by the value vw (5678_H) of the object code, i.e., 24_H at address 5678_H, is loaded into the A register.

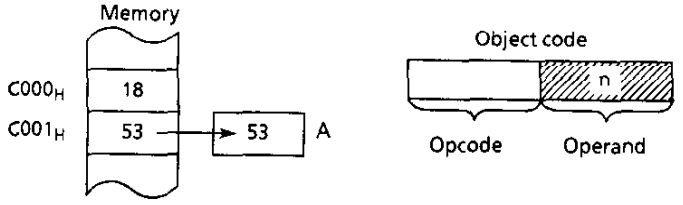
(3) Register, r or rr

The register specified by the register specification field in the object code (opcode) is the subject to be operated on.

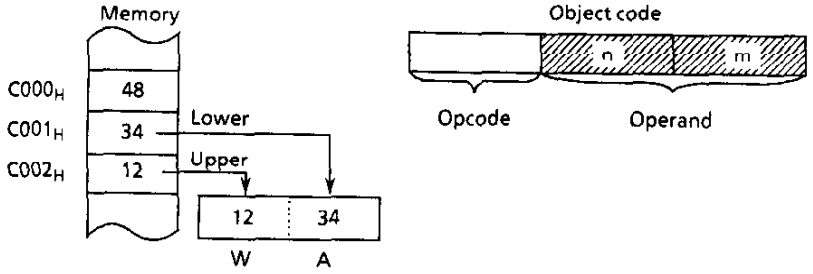
(4) Immediate, n or mn

The immediate data in the object code is the subject to be operated on. Note that when 16-bit immediate data is operated on, it is stored in memory in order of 8 low-order bits and 8 high-order bits, beginning with the smaller address.

Example 1: LD A, 53H



Example 2: LD WA, 1234H



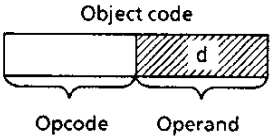
Note : When writing the assembler source program, you cannot enclose the immediate data with parentheses. If enclosed in parentheses, direct addressing mode is assumed.

(5) Relative

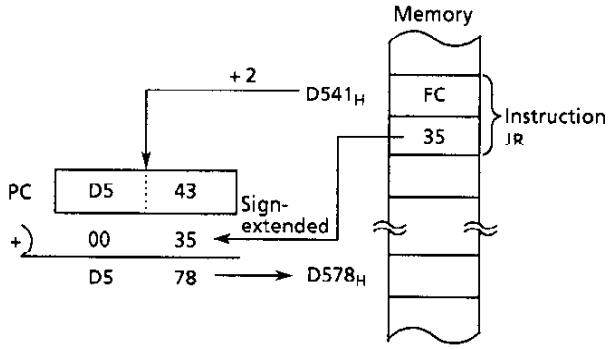
① 8-bit displacement relative

The address is specified by the content of the program counter (start address of execution instruction +2 or +3) plus the 8-bit displacement value 'd' in the object code that is added after being sign-extended (see the table below). The JR instruction is the only instruction that has this addressing mode.

Displacement 'd'	Sign-extended Value
$00 \sim 7F_H$	$0000 \sim 007F_H$ (0 ~ + 127)
$80 \sim FF_H$	$FF80 \sim FFFF_H$ (− 128 ~ − 1)



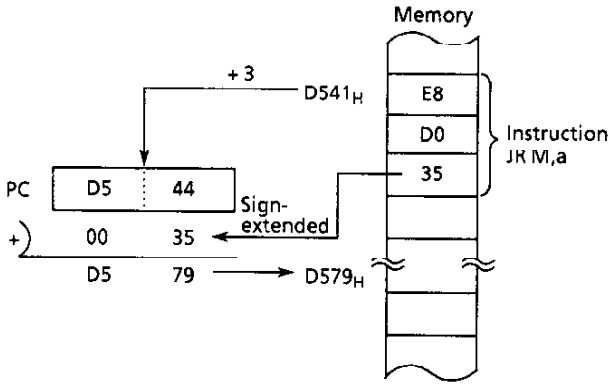
Example 1: JR $+2+35H or JR 0D578H



Control jumps to the address specified by the content ($D543_H$) of the program counter plus the displacement value 35_H that is added after being sign-extended, i.e., address $D578_H$.

Note: $ denotes the start address of the instruction to be executed.

Example 2: JR M,$+3+35H or JR M, 0D579H



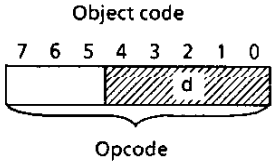
If the sign flag = 1, control jumps to the address specified by the content ($D544_H$) of the program counter plus the displacement value 35_H that is added after being sign-extended, i.e., address $D579_H$.

Note: $ denotes the start address of the instruction to be executed.

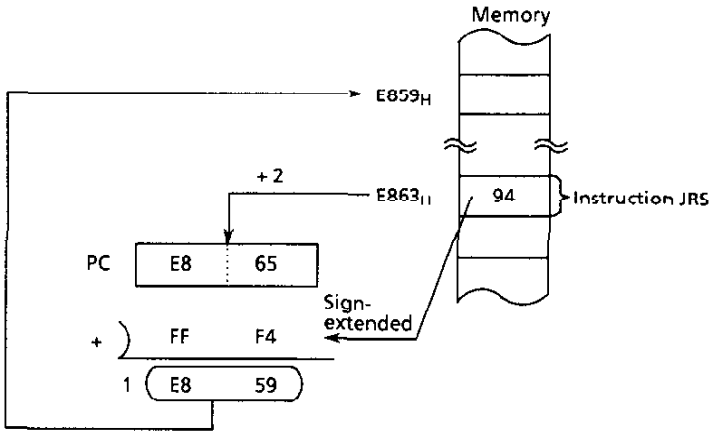
② 5-bit displacement relative

The address is specified by the content of the program counter (start address of execution instruction +2) plus the 5-bit displacement value 'd' in the opcode that is added after being sign-extended (see the table below). The JRS instruction is the only instruction that has this addressing mode.

Displacement 'd'	Sign-extended Value
00 to $0F_H$	0000 to $000F_H$ (0 to + 15)
10 to $1F_H$	FFF0 to $FFFF_H$ (– 16 to – 1)



Example: JRS T,$+2+14H or JRS T,0E859H



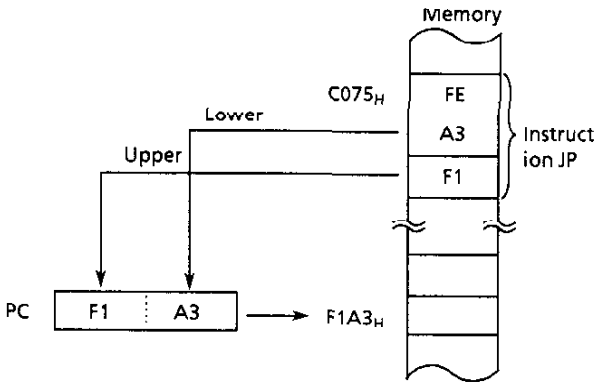
If the jump status flag = 1, control jumps to the address specified by the content ($E865_H$) of the program counter plus the displacement value 14_H that is added after being sign-extended ($FFF4_H$), i.e., address $E859_H$.

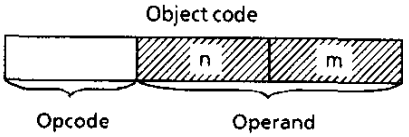
Note: $ denotes the start address of the instruction to be executed.

(6) Absolute

The address is specified by the 16-bit value (stored in order of 8 low-order bits and 8 high-order bits) in the object code.

Example: JP 0F1A3H



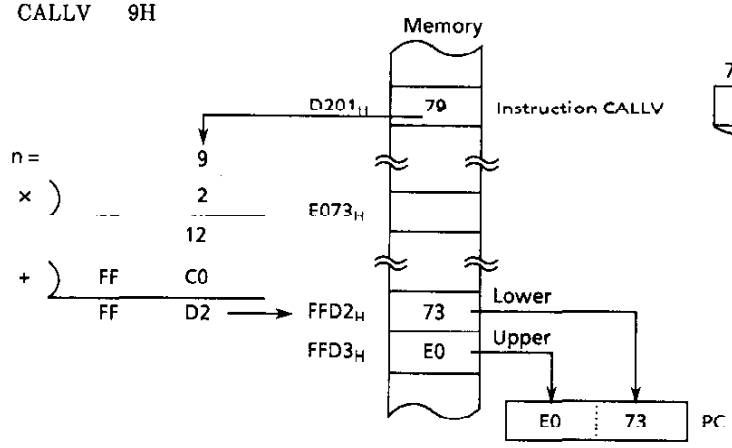


Control jumps to the address specified by the operand, i.e., address $F1A3_H$.

(7) Vector

The address is specified by the 16-bit data (vector address) that is read from a memory location whose address is indicated by twice the 4-bit data n in the opcode plus $FFC0_H$. The CALLV instruction is the only instruction that has this addressing mode.

Example: CALLV 9H



The address indicated by consecutive 2-byte contents $E073_H$ starting from address $FFD2_H$ that is obtained by doubling the data n (9_H) of the opcode and adding $FFC0_H$ is called.

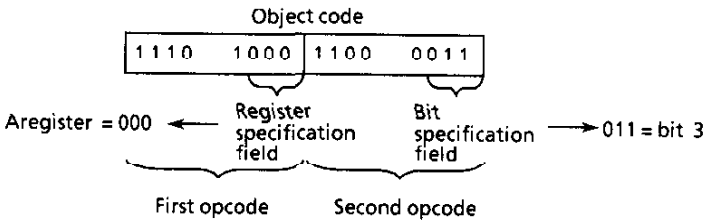
(8) Direct bit

① Register bit

The register bit to be operated on is specified by the register specification and bit specification fields in the opcodes.

Example: SET A.3

The content of the A register bit 3 is set to 1.



② Memory bit

The bit to be operated on is specified by the bit specification field in the opcode at the address specified by memory bit addressing mode (HL), (DE), (IX), (IY), (HL+d), (IX+d), (IY+d), (HL+C), (+SP), (SP+d), (PC+A), (x), or (vw).

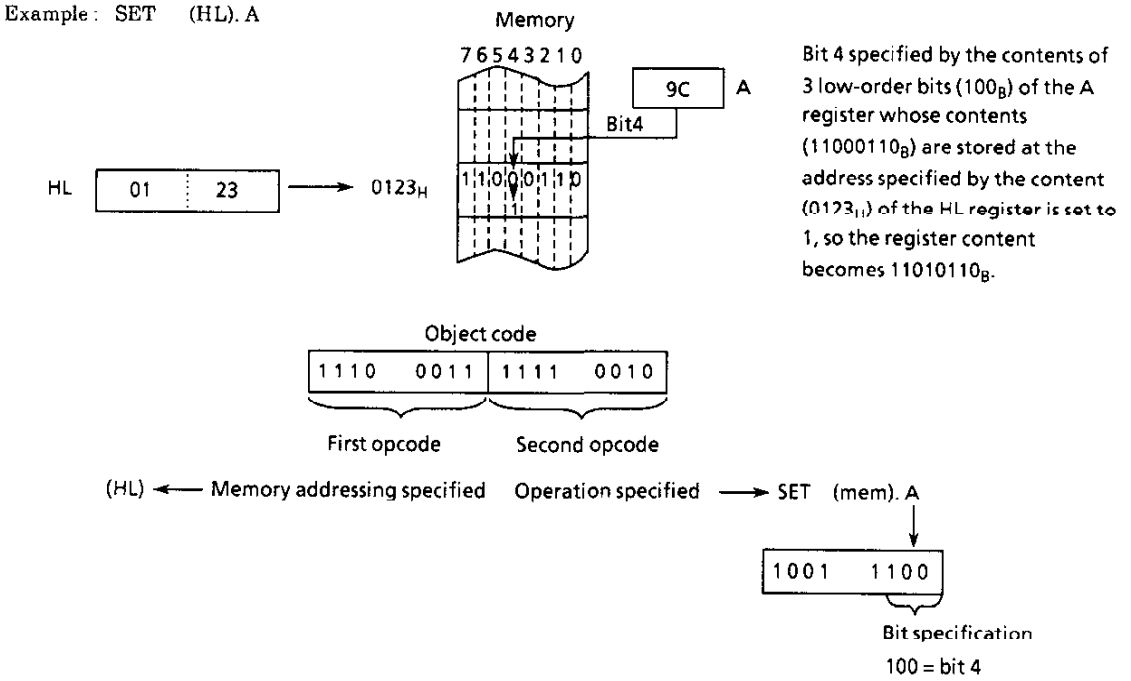
Example 1 : SET (HL). 1

Example 2 : SET (HL+57H). 6

Example 3 : SET (0058H). 3

(9) Register indirect bit

The bit to be operated on is specified by the contents of 3 low-order bits of the A register at the address specified by memory bit addressing mode (HL), (DE), (IX), (IY), (HL+d), (IX+d), (IY+d), (HL+C), (+SP), (SP+d), (PC+A), (x), or (vw).



1.5 Instruction Details

(1) Transfer and exchange

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
LD A,r	0001 0rrr	1 Z - - - -	1	A ← r
	Loads the content of 8-bit register r into the accumulator. The zero flag is set to 1 when r = 00_H and cleared to 0 when r ≠ 00_H.			
LD r,A	0100 0rrr	1 Z - - - -	1	r ← A
	Loads the accumulator content into 8-bit register r. The zero flag is set to 1 when A = 00_H and cleared to 0 when A ≠ 00_H.			
LD r,g	1110 1ggg 0100 0rrr	1 Z - - - -	1	r ← g
	Loads the content of 8-bit register g into 8-bit register r. The zero flag is set to 1 when g = 00_H and cleared to 0 when g ≠ 00_H.			
LD rr,gg	1110 1ggg 0100 1rrr	1 - - - - -	2	rr ← gg
	Loads the content of 16-bit register gg (WA, BC, DE, or HL) into 16-bit register rr. Example: If the instruction LD HL,DE is executed when DE = 1234_H, then HL = 1234_H			
LD A,(x)	0000 1100 xxxx xxxx	1 Z - - - -	3	A ← (x)
	Loads the memory content at the address (0000 to $00FF_H$) specified directly by x in the object code into the accumulator. The zero flag is set to 1 when the data transferred to the accumulator is 00_H.			
LD A,(HL)	0000 1101	1 Z - - - -	2	A ← (HL)
	Loads the memory content at the address (0000 to $00FF_H$) specified by the HL register pair into the accumulator. The zero flag is set to 1 when the data transferred to the accumulator is 00_H.			
LD r,(x)	1110 0000 xxxx xxxx 0100 0rrr	1 Z - - - -	4	r ← (x)
	Loads the memory content at the address (0000 to $00FF_H$) specified directly by x in the object code into 8-bit register r. The zero flag is set to 1 when the data transferred to the register r is 00_H.			
LD r,(vw)	1110 0001 wwww wwww vvvv vvvv 0100 0rrr	1 Z - - - -	5	r ← (vw)
	Loads the memory content at the address (0000 to $FFFF_H$) specified directly by vw in the object code into 8-bit register r. The zero flag is set to 1 when the data transferred to the register r is 00_H.			
LD r,(DE)	1110 0010 0100 0rrr	1 Z - - - -	3	r ← (DE)
	Loads the memory content at the address (0000 to $FFFF_H$) specified by the DE register pair into 8-bit register r. The zero flag is set to 1 when the data transferred to the register r is 00_H.			
LD r,(HL)	1110 0011 0100 0rrr	1 Z - - - -	3	r ← (HL)
	Loads the memory content at the address (0000 to $FFFF_H$) specified by the HL register pair into 8-bit register r. The zero flag is set to 1 when the data transferred to the register r is 00_H.			
LD r,(IX)	1110 0100 0100 0rrr	1 Z - - - -	3	r ← (IX)
	Loads the memory content at the address (0000 to $FFFF_H$) specified by the index register IX into 8-bit register r. The zero flag is set to 1 when the data transferred to the register r is 00_H.			
LD r,(IY)	1110 0101 0100 0rrr	1 Z - - - -	3	r ← (IY)
	Loads the memory content at the address (0000 to $FFFF_H$) specified by the index register IY into 8-bit register r. The zero flag is set to 1 when the data transferred to the register r is 00_H.			
LD r,(IX+d)	1101 0100 dddd dddd 0100 0rrr	1 Z - - - -	5	r ← (IX+d)
	Loads the memory content at the address specified by the content of the index register IX plus the 8-bit data d in the object code that is added after being sign-extended into 8-bit register r. The zero flag is set to 1 when the data transferred to the register r is 00_H.			
LD r,(IY+d)	1101 0101 dddd dddd 0100 0rrr	1 Z - - - -	5	r ← (IY+d)
	Loads the memory content at the address specified by the content of the index register IY plus the 8-bit data d in the object code that is added after being sign-extended into 8-bit register r. The zero flag is set to 1 when the data transferred to the register r is 00_H.			
LD r,(SP+d)	1101 0110 dddd dddd 0100 0rrr	1 Z - - - -	5	r ← (SP+d)
	Loads the memory content at the address specified by the content of the stack pointer plus the 8-bit data d in the object code that is added after being sign-extended into 8-bit register r. The zero flag is set to 1 when the data transferred to the register r is 00_H.			
LD r,(HL+d)	1101 0111 dddd dddd 0100 0rrr	1 Z - - - -	5	r ← (HL+d)
	Loads the memory content at the address specified by the content of the HL register pair plus the 8-bit data d in the object code that is added after being sign-extended into 8-bit register r. The zero flag is set to 1 when the data transferred to the register r is 00_H.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
LD r,(HL+C)	1110 0111 : 0100 0rrr	1 Z - - - -	5	r ← (HL+C)
	Loads the memory content at the address specified by the content of the HL register pair plus the content of the C register that is added after being sign-extended into 8-bit register r. The zero flag is set to 1 when the data transferred to the register r is 00_H.			
LD r,(+SP)	1110 0110 : 0100 0rrr	1 Z - - - -	4	SP ← SP+1:r ← (SP)
	Loads the memory content at the address specified by the content of the stack pointer after being incremented into 8-bit register r. The zero flag is set to 1 when the data transferred to the register r is 00_H. This instruction is used for popping 8-bit data from the stack into a register.			
LD r,(PC+A)	0100 1111 : 0100 0rrr	1 Z - - - -	5	r ← (PC+A)
	Loads the memory content at the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended into 8-bit register r. The zero flag is set to 1 when the data transferred to the register r is 00_H. This instruction is most suitable for the code conversion processing.			
LD rr,(x)	1110 0000 : xxxx xxxx : 0100 1rrr	1 - - - - -	5	rr ← (x+1,x)
	Loads consecutive 2-byte memory contents beginning with the address (0000 to $00FF_H$) specified directly by x in the object code into 16-bit register rr. Example: If the instruction LD WA,(72_H) is executed when the contents at addresses 0072_H and 0073_H respectively are $8E_H$ and 59_H, then W = 59_H and A = 83_H.			
LD rr,(vw)	1110 0001 : wwww wwww : vvvv vvvv : 0100 1rrr	1 - - - - -	6	rr ← (vw+1,vw)
	Loads consecutive 2-byte memory contents beginning with the address (0000 to $FFFF_H$) specified directly by vw in the object code into 16-bit register rr. If the address specified by vw is $0FFF_H$, the memory content at address 1000_H is loaded into the upper byte of the register.			
LD rr,(DE)	1110 0010 : 0100 1rrr	1 - - - - -	4	rr ← (DE+1,DE)
	Loads consecutive 2-byte memory contents beginning with the address specified by the DE register pair into 16-bit register rr.			
LD rr,(HL)	1110 0011 : 0100 1rrr	1 - - - - -	4	rr ← (HL+1,HL)
	Loads consecutive 2-byte memory contents beginning with the address specified by the HL register pair into 16-bit register rr.			
LD rr,(IX)	1110 0100 : 0100 1rrr	1 - - - - -	4	rr ← (IX+1,IX)
	Loads consecutive 2-byte memory contents beginning with the address specified by the index register IX into 16-bit register rr.			
LD rr,(IY)	1110 0101 : 0100 1rrr	1 - - - - -	4	rr ← (IY+1,IY)
	Loads consecutive 2-byte memory contents beginning with the address specified by the index register IY into 16-bit register rr.			
LD rr,(IX+d)	1101 0100 : dddd dddd : 0100 1rrr	1 - - - - -	6	rr ← (IX+d+1,IX+d)
	Loads consecutive 2-byte memory contents beginning with the address specified by the content of the index register IX plus the 8-bit data d in the object code that is added after being sign extended into 16-bit register rr.			
LD rr,(IY+d)	1101 0101 : dddd dddd : 0100 1rrr	1 - - - - -	6	rr ← (IY+d+1,IY+d)
	Loads consecutive 2-byte memory contents beginning with the address specified by the content of the index register IY plus the 8-bit data d in the object code that is added after being sign-extended into 16-bit register rr.			
LD rr,(SP+d)	1101 0110 : dddd dddd : 0100 1rrr	1 - - - - -	6	rr ← (SP+d+1,SP+d)
	Loads consecutive 2-byte memory contents beginning with the address specified by the content of the stack pointer plus the 8-bit data d in the object code that is added after being sign-extended into 16-bit register rr. Example: If the instruction LD WA,(SP + 32_H) is executed when the memory contents at addresses 5216_H and 5217_H respectively are $9F_H$ and $C3_H$, then A = $9F_H$ and W = $C3_H$.			
LD rr,(HL+d)	1101 0111 : dddd dddd : 0100 1rrr	1 - - - - -	6	rr ← (HL+d+1,HL+d)
	Loads consecutive 2-byte memory contents beginning with the address specified by the content of the HL register pair plus the 8-bit data d in the object code that is added after being sign-extended into 16-bit register rr.			
LD rr,(HL+C)	1110 0111 : 0100 1rrr	1 - - - - -	6	rr ← (HL+C+1,HL+C)
	Loads consecutive 2-byte memory contents beginning with the address specified by the content of the HL register pair plus the content of the C register that is added after being sign-extended into 16-bit register rr.			
LD rr,(+SP)	1110 0110 : 0100 1rrr	1 - - - - -	5	SP ← SP+1:rr ← (SP+1,SP)
	Loads consecutive 2-byte memory contents beginning with the address specified by the content of the stack pointer after being incremented into 16-bit register rr.			
LD rr,(PC+A)	0100 1111 : 0100 1rrr	1 - - - - -	6	rr ← (PC+A+1,PC+A)
	Loads consecutive 2-byte memory contents beginning with the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended into 16-bit register rr.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
LD (x),A	0000 1110 : xxxx xxxx	1 - - - - -	3	(x) ← A
	Stores the accumulator content in a memory location whose address (0000 to 00FF$_H$) is specified directly by x in the object code.			
LD (HL),A	0000 1111	1	2	(HL) ← A
	Stores the accumulator content in a memory location whose address (0000 to FFFF$_H$) is specified by the HL register pair.			
LD (x),r	1111 0000 : xxxx xxxx : 0111 1rrr	1 - - - - -	4	(x) ← r
	Stores the content of 8-bit register r in memory location whose address (0000 to 00FF$_H$) is specified directly by x in the object code.			
LD (vw),r	1111 0001 : wwww wwww : vvvv vvvv : 0111 1rrr	1 - - - - -	5	(vw) ← r
	Stores the content of 8-bit register r in a memory location whose address (0000 to FFFF$_H$) is specified directly by vw in the object code.			
LD (DE),r	1111 0010 : 0111 1rrr	1 - - - - -	3	(DE) ← r
	Stores the content of 8-bit register r in a memory location whose address (0000 to FFFF$_H$) is specified by the DE register pair.			
LD (HL),r	1111 0011 : 0111 1rrr	1 - - - - -	3	(HL) ← r
	Stores the content of 8-bit register r in a memory location whose address (0000 to FFFF$_H$) is specified by the HL register pair.			
LD (IX),r	1111 0100 : 0111 1rrr	1 - - - - -	3	(IX) ← r
	Stores the content of 8-bit register r in a memory location whose address (0000 to FFFF$_H$) is specified by the index register IX.			
LD (IY),r	1111 0101 : 0111 1rrr	1 - - - - -	3	(IY) ← r
	Stores the content of 8-bit register r in a memory location whose address (0000 to FFFF$_H$) is specified by the index register IY.			
LD (IX+d),r	0101 0100 : dddd dddd : 0111 1rrr	1 - - - - -	5	(IX+d) ← r
	Stores the content of 8-bit register r in a memory location whose address is specified by the content of the index register IX plus the 8-bit data d in the object code that is added after being sign-extended.			
LD (IY+d),r	0101 0101 : dddd dddd : 0111 1rrr	1 - - - - -	5	(IY+d) ← r
	Stores the content of 8-bit register r in a memory location whose address is specified by the content of the index register IY plus the 8-bit data d in the object code that is added after being sign-extended.			
LD (SP+d),r	0101 0110 : dddd dddd : 0111 1rrr	1 - - - - -	5	(SP+d) ← r
	Stores the content of 8-bit register r in a memory location whose address is specified by the content of the stack pointer plus the 8-bit data d in the object code that is added after being sign-extended.			
LD (HL+d),r	0101 0111 : dddd dddd : 0111 1rrr	1 - - - - -	5	(HL+d) ← r
	Stores the content of 8-bit register r in a memory location whose address is specified by the content of the HL register pair plus the 8-bit data d in the object code that is added after being sign-extended.			
LD (HL+C),r	1111 0111 : 0111 1rrr	1 - - - - -	5	(HL+C) ← r
	Stores the content of 8-bit register r in a memory location whose address is specified by the content of the HL register pair plus the content of the C register that is added after being sign extended.			
LD (SP-),r	1111 0110 : 0111 1rrr	1 - - - - -	4	(SP) ← r:SP ← SP-1
	Stores the content of 8-bit register r in a memory location whose address is specified by the stack pointer. Then the content of the stack pointer is decremented. This instruction is used for pushing an 8-bit register into the stack.			
LD (x),rr	1111 0000 : xxxx xxxx : 0110 1rrr	1 - - - - -	5	(x+1,x) ← rr
	Stores the content of 16-bit register rr in consecutive 2-byte memory locations beginning with the address (0000 to 00FF$_H$) specified directly by x in the object code. The register content is stored in order of the lower byte and the upper byte.			
LD (vw),rr	1111 0001 : wwww wwww : vvvv vvvv : 0110 1rrr	1 - - - - -	6	(vw+1,vw) ← rr
	Stores the content of 16-bit register rr in consecutive 2-byte memory locations beginning with the address (0000 to FFFF$_H$) specified directly by vw in the object code. The register content is stored in order of the lower byte and the upper byte.			
LD (DE),rr	1111 0010 : 0110 1rrr	1 - - - - -	4	(DE+1,DE) ← rr
	Stores the content of 16-bit register rr in consecutive 2-byte memory locations beginning with the address specified by the DE register pair. The register content is stored in order of the lower byte and the upper byte.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
LD (HL),rr	1111 0011 : 0110 1rrr	1 - - - - -	4	(HL+1,HL) ← rr
	Stores the content of 16-bit register rr in consecutive 2-byte memory locations beginning with the address specified by the HL register pair. The register content is stored in order of the lower byte and the upper byte.			
LD (IX),rr	1111 0100 : 0110 1rrr	1 - - - - -	4	(IX+1,IX) ← rr
	Stores the content of 16-bit register rr in consecutive 2-byte memory locations beginning with the address specified by the index register IX. The register content is stored in order of the lower byte and the upper byte.			
LD (IY),rr	1111 0101 : 0110 1rrr	1 - - - - -	4	(IY+1,IY) ← rr
	Stores the content of 16-bit register rr in consecutive 2-byte memory locations beginning with the address specified by the index register IY. The register content is stored in order of the lower byte and the upper byte.			
LD (IX+d),rr	0101 0100 : dddd dddd : 0110 1rrr	1 - - - - -	6	(IX+d+1,IX+d) ← rr
	Stores the content of 16-bit register rr in consecutive 2-byte memory locations beginning with the address specified by the content of the index register IY plus the 8-bit data d in the object code that is added after being sign-extended. The register content is stored in order of the lower byte and the upper byte.			
LD (IY+d),rr	0101 0101 : dddd dddd : 0110 1rrr	1 - - - - -	6	(IY+d+1,IY+d) ← rr
	Stores the content of 16-bit register rr in consecutive 2-byte memory locations beginning with the address specified by the content of the index register IX plus the 8-bit data d in the object code that is added after being sign-extended. The register content is stored in order of the lower byte and the upper byte.			
LD (SP+d),rr	0101 0110 : dddd dddd : 0110 1rrr	1 - - - - -	6	(SP+d+1,SP+d) ← rr
	Stores the content of 16-bit register rr in consecutive 2-byte memory locations beginning with the address specified by the content of the stack pointer plus the 8-bit data d in the object code that is added after being sign-extended. The register content is stored in order of the lower byte and the upper byte.			
LD (HL+d),rr	0101 0111 : dddd dddd : 0110 1rrr	1 - - - - -	6	(HL+d+1,HL+d) ← rr
	Stores the content of 16-bit register rr in consecutive 2-byte memory locations beginning with the address specified by the content of the HL register pair plus the 8-bit data d in the object code that is added after being sign-extended. The register content is stored in order of the lower byte and the upper byte.			
LD (HL+C),rr	1111 0111 : 0110 1rrr	1 - - - - -	6	(HL+C+1,HL+C) ← rr
	Stores the content of 16-bit register rr in consecutive 2-byte memory locations beginning with the address specified by the content of the HL register pair plus the content of the C register that is added after being sign-extended. The register content is stored in order of the lower byte and the upper byte.			
LD (SP-),rr	1111 0110 : 0110 1rrr	1 - - - - -	5	(SP+1,SP) ← rr;SP ← SP-1
	Stores the content of 16-bit register rr in consecutive 2-byte memory locations beginning with the address specified by the stack pointer. Then the content of the stack pointer is decremented. This instruction is used for pushing a 16-bit register into the stack.			
LD r,n	0001 1rrr : nnnn nnnn	1 - - - - -	2	r ← n
	Loads the immediate data n in the object code into 8-bit register r. Example: If the instruction LD A,53_H is executed, then A = 53_H.			
LD rr, mn	0100 1rrr : nnnn nnnn : mmmm mmmm	1 - - - - -	3	rr ← mn
	Loads the immediate data n in the object code into 16-bit register rr. Example: If the instruction LD WA,1234_H is executed, then A = 34_H and W = 12_H.			
LD (x),n	0000 1010 : xxxx xxxx : nnnn nnnn	1 - - - - -	4	(x) ← n
	Stores the object code's immediate data n in a memory location whose address (0000 to $00FF_H$) is specified directly by x in the object code.			
LD (vw),n	1111 0001 : wwww wwww : vvvv vvvv 1111 1001 : nnnn nnnn	1 - - - - -	6	(vw) ← n
	Stores the object code's immediate data n in a memory location whose address (0000 to $FFFF_H$) is specified directly by vw in the object code.			
LD (DE),n	1111 0010 : 1111 1001 : nnnn nnnn	1 - - - - -	4	(DE) ← n
	Stores the object code's immediate data n in a memory location whose address is specified by the DE register pair.			
LD (HL),n	0000 1011 : nnnn nnnn	1 - - - - -	4	(HL) ← n
	Stores the object code's immediate data n in a memory location whose address is specified by the HL register pair.			
LD (IX),n	1111 0100 : 1111 1001 : nnnn nnnn	1 - - - - -	4	(IX) ← n
	Stores the object code's immediate data n in a memory location whose address is specified by the index register IX.			
LD (IY),n	1111 0101 : 1111 1001 : nnnn nnnn	1 - - - - -	4	(IY) ← n
	Stores the object code's immediate data n in a memory location whose address is specified by the index register IY.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
LD (IX+d),n	0101 0100 : dddd dddd : 1111 1001 : nnnn nnnn	1 - - - - -	6	(IX+d) ← n
	Stores the object code's immediate data n in a memory location whose address is specified by the content of the index register IX plus the 8-bit data d in the object code that is added after being sign-extended.			
LD (IY+d),n	0101 0101 : dddd dddd : 1111 1001 : nnnn nnnn	1 - - - - -	6	(IY+d) ← n
	Stores the object code's immediate data n in a memory location whose address is specified by the content of the index register IY plus the 8-bit data d in the object code that is added after being sign-extended.			
LD (SP+d),n	0101 0110 : dddd dddd : 1111 1001 : nnnn nnnn	1 - - - - -	6	(SP+d) ← n
	Stores the object code's immediate data n in a memory location whose address is specified by the content of the stack pointer plus the 8-bit data d in the object code that is added after being sign-extended.			
LD (HL+d),n	0101 0111 : dddd dddd : 1111 1001 : nnnn nnnn	1 - - - - -	6	(HL+d) ← n
	Stores the object code's immediate data n in a memory location whose address is specified by the content of the HL register pair plus the 8-bit data d in the object code that is added after being sign-extended.			
LD (HL+C),n	1111 0111 : 1111 1001 : nnnn nnnn	1 - - - - -	6	(HL+C) ← n
	Stores the object code's immediate data n in a memory location whose address is specified by the content of the HL register pair plus the content of the C register that is added after being sign-extended.			
LD (SP-),n	1111 0110 : 1111 1001 : nnnn nnnn	1 - - - - -	5	(SP) ← n:SP ← SP-1
	Stores the object code's immediate data n in a memory location whose address is specified by the stack pointer. Then the content of the stack pointer is decremented.			
LDW (x),mn	0000 1000 : xxxx xxxx : nnnn nnnn : mmmm mmmm	1 - - - - -	6	(x+1,x) ← mn
	Stores the object code's 16-bit immediate data mn in consecutive 2-byte memory locations beginning with the address (0000 to $00FF_H$) specified directly by x in the object code. The immediate data is stored in order of the lower byte n and the upper byte m. Example: If the instruction LDW (73_H), 1234_H is executed, data 34_H and 12_H are written to addresses 0073_H and 0074_H, respectively.			
LDW (HL),mn	0000 1001 : nnnn nnnn : mmmm mmmm	1 - - - - -	5	(HL+1,HL) ← mn
	Stores the object code's 16-bit immediate data mn in consecutive 2-byte memory locations beginning with the address specified by the HL register pair. The immediate data is stored in order of the lower byte and the upper byte.			
•1 PUSH rr	0101 00rr	- - - - - -	3	(SP,SP-1) ← rr:SP ←SP-2
	Stores the contents of 8 high-order bits of register rr in a memory location whose address is specified by the stack pointer and the contents of 8 low-order bits in a memory location whose address is specified by the said address minus 1. Then the content of the stack pointer is decremented by 2. Only WA, BC, DE, or HL can be specified for rr. Example: If the instruction PUSH WA is executed when $SP = 013F_H$ and $WA = 1234_H$, the contents of addresses $013F_H$ and $013E_H$ respectively become 12_H and 34_H. Also, $SP = 013D_H$.			
PUSH gg	1110 1ggg : 1101 1000	- - - - - -	4	(SP,SP-1) ← gg:SP ←SP-2
	Stores the contents of 8 high-order bits of register gg in a memory location whose address is specified by the stack pointer and the contents of 8 low-order bits in a memory location whose address is specified by the said address minus 1. Then the content of the stack pointer is decremented by 2. Example: If the instruction PUSH IX is executed when $SP = 013F_H$ and $IX = 1234_H$, the contents of addresses $013F_H$ and $013E_H$ respectively become 12_H and 34_H. Also, $SP = 013D_H$.			
•1 POP rr	1101 00rr	- - - - - -	4	SP ← SP+2:rr ← (SP,SP-1)
	Loads the memory content at the address specified by the content of the stack pointer plus 2 into the 8 high-order bits of register rr and the memory content at the said address minus 1 into the 8 low-order bits of register rr. Only WA, BC, DE, or HL can be specified for rr.			
POP gg	1110 1ggg : 1101 1001	- - - - - -	5	SP ← SP+2:gg ← (SP,SP-1)
	Loads the memory content at the address specified by the content of the stack pointer plus 2 into the 8 high-order bits of register gg and the memory content at the said address minus 1 into the 8 low-order bits of register gg.			
PUSH PSW	1110 1000 : 1101 1100	- - - - - -	3	(SP) ← PSW:SP ← SP-1
	Stores the content of the program status word PSW in a memory location whose address is specified by the stack pointer and then decrements the content of the stack pointer by 1. Example: If the instruction PUSH PSW is executed when $SP = 2345_H$, $PSW = 63_H$ (JF = HF = SF = VF = 0, ZF = CF = 1), the content of address 2345_H becomes 63_H. Also, $SP = 2344_H$.			
POP PSW	1110 1000 : 1101 1101	* * * * * *	4	SP ← SP+1:PSW ← (SP)
	Loads the memory content at the address specified by the content of the stack pointer after being incremented into the program status word PSW. Example: If the instruction POP PSW is executed when the memory content at address $0137_H = 63_H$ and $SP = 0136_H$, then $SP = 0137_H$ and $PSW = 63_H$ (JF = HF = SF = VF = 0, ZF = CF = 1).			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
LD PSW,n	1110 1000 : 1101 1110 : nnnn nnnn	* * * * * *	3	PSW ← n
	Stores the object code's immediate data n in the program status word PSW. Bits in the immediate data n, beginning with the most significant bit, correspond to JF, ZF, CF, HF, SF, and VF, with the two least significant bits ignored. Example: If the instruction LD PSW,00110100B is executed, then JF = 0, ZF = 0, CF = 1, HF = 1, SF = 0, and VF = 1.			
LD SP,SP+d	0011 0111 : dddd dddd	1 - - - - -	3	SP ← SP+d
	Loads the content of the stack pointer plus the object code's 8-bit data d into the stack pointer. Therefore, this instruction functions as an immediate add instruction for the stack pointer. Example: If the instruction LD SP,SP+4 is executed when SP = 2345_H, then SP = 2349_H and JF = 1.			
LD SP,SP-d	0011 1111 : dddd dddd	1 - - - - -	3	SP ← SP-d
	Loads the content of the stack pointer minus the object code's 8-bit data d into the stack pointer. Therefore, this instruction functions as an immediate subtract instruction for the stack pointer.			
XCH r,g	1110 1ggg : 0111 0rrr	1 Z - - - -	3	r ⇔ g
	Exchanges the content of 8-bit register r with that of 8-bit register g. When the content of g before the exchange = 00_H, the zero flag is set to 1. Otherwise, it is cleared to 0. Example: If the instruction XCH A,B is executed when A = $3C_H$ and B = $5F_H$, then A = $5F_H$, B = $3C_H$, and ZF = 0.			
XCH rr,gg	1110 1ggg : 0111 1rrr	1 - - - - -	3	rr ⇔ gg
	Exchanges the content of 16-bit register rr with that of 16-bit register gg. The zero flag does not change state. Example: If the instruction XCH HL,DE is executed when HL = 0123_H and DE = 9587_H, then HL = 9587_H and DE = 0123_H.			
XCH r,(x)	1110 0000 : xxxx xxxx : 0111 0rrr	1 Z - - - -	5	
	Exchanges the memory content at the address (0000 to $00FF_H$) specified directly by x in the object code with that of 8-bit register r. When the data transferred to the register r = 00_H, the zero flag is set to 1.			
XCH r,(vw)	1110 0001 : wwww wwww : vvvv vvvv : 0111 0rrr	1 Z - - - -	6	
	Exchanges the memory content at the address (0000 to $FFFF_H$) specified directly by vw in the object code with that of 8-bit register r. When the data transferred to the register r = 00_H, the zero flag is set to 1.			
XCH r,(DE)	1110 0010 : 0111 0rrr	1 Z - - - -	4	r ⇔ (DE)
	Exchanges the memory content at the address specified by the DE register pair with that of 8-bit register r. When the data transferred to the register r = 00_H, the zero flag is set to 1.			
XCH r,(HL)	1110 0011 : 0111 0rrr	1 Z - - - -	4	r ⇔ (HL)
	Exchanges the memory content at the address specified by the HL register pair with that of 8-bit register r. When the data transferred to the register r = 00_H, the zero flag is set to 1.			
XCH r,(IX)	1110 0100 : 0111 0rrr	1 Z - - - -	4	r ⇔ (IX)
	Exchanges the memory content at the address specified by the index register IX with that of 8-bit register r. When the data transferred to the register r = 00_H, the zero flag is set to 1.			
XCH r,(IY)	1110 0101 : 0111 0rrr	1 Z - - - -	4	r ⇔ (IY)
	Exchanges the memory content at the address specified by the index register IY with that of 8-bit register r. When the data transferred to the register r = 00_H, the zero flag is set to 1.			
XCH r,(IX+d)	1101 0100 : dddd dddd : 0111 0rrr	1 Z - - - -	6	r ⇔ (IX+d)
	Exchanges the memory content at the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended with that of 8-bit register r. When the data transferred to the register r = 00_H, the zero flag is set to 1.			
XCH r,(IY+d)	1101 0101 : dddd dddd : 0111 0rrr	1 Z - - - -	6	r ⇔ (IY+d)
	Exchanges the memory content at the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended with that of 8-bit register r. When the data transferred to the register r = 00_H, the zero flag is set to 1.			
XCH r,(SP+d)	1101 0110 : dddd dddd : 0111 0rrr	1 Z - - - -	6	r ⇔ (SP+d)
	Exchanges the memory content at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended with that of 8-bit register r. When the data transferred to the register r = 00_H, the zero flag is set to 1.			
XCH r,(HL+d)	1101 0111 : dddd dddd : 0111 0rrr	1 Z - - - -	6	r ⇔ (HL+d)
	Exchanges the memory content at the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended with that of 8-bit register r. When the data transferred to the register r = 00_H, the zero flag is set to 1.			
XCH r,(HL+C)	1110 0111 : 0111 0rrr	1 Z - - - -	6	r ⇔ (HL+C)
	Exchanges the memory content at the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended with that of 8 bit register r. When the data transferred to the register r = 00_H, the zero flag is set to 1.			
XCH r,(+SP)	1110 0110 : 0111 0rrr	1 Z - - - -	5	SP ← SP+1:r ⇔ (SP)
	Exchanges the memory content at the address specified by the content of the stack pointer after being incremented with that of 8-bit register r. When the data transferred to the register r = 00H, the zero flag is set to 1.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
XCH r,(PC+A)	0100 1111 : 0111 0rrr	1 Z - - - -	6	r ⇔ (PC+A)
	Exchanges the memory content at the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended with that of 8-bit register r. When the data transferred to the register r = 00_H, the zero flag is set to 1.			
XCH rr,(x)	1110 0000 : xxxx xxxx : 1101 1rrr	1 - - - - -	7	rr ⇔ (x+1,x)
	Exchanges consecutive 2-byte memory contents beginning with the address (0000 to $00FF_H$) specified directly by x in the object code with those of 16-bit register rr.			
XCH rr,(vw)	1110 0001 : wwww wwww : vvvv vvvv 1101 1rrr	1 - - - - -	8	rr ⇔ (vw+1,vw)
	Exchanges consecutive 2-byte memory contents starting from the address (0000 to $FFFF_H$) specified directly by vw in the object code with those of 16-bit register rr.			
XCH rr,(DE)	1110 0010 : 1101 1rrr	1 - - - - -	6	rr ⇔ (DE+1,DE)
	Exchanges consecutive 2-byte memory contents starting from the address specified by the DE register pair with those of 16-bit register rr			
XCH rr,(HL)	1110 0011 : 1101 1rrr	1 - - - - -	6	rr ⇔ (HL+1,HL)
	Exchanges consecutive 2-byte memory contents starting from the address specified by the HL register pair with those of 16-bit register rr.			
XCH rr,(IX)	1110 0100 : 1101 1rrr	1 - - - - -	6	rr ⇔ (IX+1,IX)
	Exchanges consecutive 2-byte memory contents starting from the address specified by the index register IX with those of 16-bit register rr.			
XCH rr,(IY)	1110 0101 : 1101 1rrr	1 - - - - -	6	rr ⇔ (IY+1,IY)
	Exchanges consecutive 2-byte memory contents starting from the address specified by the index register IY with those of 16-bit register rr.			
XCH rr,(IX+d)	1101 0100 : dddd dddd : 1101 1rrr	1 - - - - -	8	rr ⇔ (IX+d+1,IX+d)
	Exchanges consecutive 2-byte memory contents starting from the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended with those of 16-bit register rr.			
XCH rr,(IY+d)	1101 0101 : dddd dddd : 1101 1rrr	1 - - - - -	8	rr ⇔ (IY+d+1,IY+d)
	Exchanges consecutive 2-byte memory contents starting from the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended with those of 16-bit register rr.			
XCH rr,(SP+d)	1101 0110 : dddd dddd : 1101 1rrr	1 - - - - -	8	rr ⇔ (SP+d+1,SP+d)
	Exchanges consecutive 2-byte memory contents starting from the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended with those of 16-bit register rr.			
XCH rr,(HL+d)	1101 0111 : dddd dddd : 1101 1rrr	1 - - - - -	8	rr ⇔ (HL+d+1,HL+d)
	Exchanges consecutive 2-byte memory contents starting from the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended with those of 16-bit register rr.			
XCH rr,(HL+C)	1110 0111 : 1101 1rrr	1 - - - - -	8	rr ⇔ (HL+C+1,HL+C)
	Exchanges consecutive 2-byte memory contents starting from the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended with those of 16-bit register rr.			
XCH rr,(+SP)	1110 0110 : 1101 1rrr	1 - - - - -	7	SP ← SP+1;rr ⇔ (SP+1,SP)
	Exchanges consecutive 2-byte memory contents starting from the address specified by the content of the stack pointer after being incremented with those of 16 bit register rr.			
XCH rr,(PC+A)	0100 1111 : 1101 1rrr	1 - - - - -	8	rr ⇔ (PC+A+1,PC+A)
	Exchanges consecutive 2-byte memory contents starting from the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended with those of 16-bit register rr.			

(2) Arithmetic/logic operations

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
CMP A,n	0110 0111 : nnnn nnnn	Z Z C H S V	2	A-n
	Compares the accumulator content with the immediate data n in the object code. The carry flag is set when A < n and cleared to 0 when A > n.			
CMP g,n	1110 1ggg : 0110 0111 : nnnn nnnn	Z Z C H S V	3	g-n
	Compares the content of 8-bit register g with the immediate data n in the object code. The carry flag is set when g < n and cleared to 0 when g > n.			
CMP gg,mn	1110 1ggg : 0110 1111 : nnnn nnnn : mmmm mmmm	Z Z C U S V	4	gg-mn
	Compares the content of 16-bit register gg with the immediate data mn in the object code. The carry flag is set when gg < mn and cleared to 0 when gg > mn.			
CMP r,g	1110 1ggg : 00rr r111	Z Z C H S V	2	r-g
	Compares the content of 8-bit register r with that of 8-bit register g. The carry flag is set when r < g and cleared to 0 when r > g.			
CMP rr,gg	1110 1ggg : 10rr r111	Z Z C U S V	4	rr-gg
	Compares the content of 16-bit register rr with that of 16-bit register gg.			
CMP r,(x)	1110 0000 : xxxx xxxx : 00rr r111	Z Z C H S V	4	r-(x)
	Compares the memory content at the address (0000 to 00FF$_H$) specified directly by x in the object code with that of 8-bit register r.			
CMP r,(vw)	1110 0001 : wwww wwww : vvvv vvvv : 00rr r111	Z Z C H S V	5	r-(vw)
	Compares the memory content at the address (0000 to FFFF$_H$) specified directly by vw in the object code with that of 8-bit register r.			
CMP r,(DE)	1110 0010 : 00rr r111	Z Z C H S V	3	r-(DE)
	Compares the memory content at the address specified by the DE register pair with that of 8-bit register r.			
CMP r,(HL)	1110 0011 : 00rr r111	Z Z C H S V	3	r-(HL)
	Compares the memory content at the address specified by the HL register pair with that of 8-bit register r.			
CMP r,(IX)	1110 0100 : 00rr r111	Z Z C H S V	3	r-(IX)
	Compares the memory content at the address specified by the index register IX with that of 8-bit register r.			
CMP r,(IY)	1110 0101 : 00rr r111	Z Z C H S V	3	r-(IY)
	Compares the memory content at the address specified by the index register IY with that of 8-bit register r.			
CMP r,(IX+d)	1101 0100 : dddd dddd : 00rr r111	Z Z C H S V	5	r-(IX+d)
	Compares the memory content at the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended with that of 8-bit register r.			
CMP r,(IY+d)	1101 0101 : dddd dddd : 00rr r111	Z Z C H S V	5	r-(IY+d)
	Compares the memory content at the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended with that of 8-bit register r.			
CMP r,(SP+d)	1101 0110 : dddd dddd : 00rr r111	Z Z C H S V	5	r-(SP+d)
	Compares the memory content at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended with that of 8-bit register r.			
CMP r,(HL+d)	1101 0111 : dddd dddd : 00rr r111	Z Z C H S V	5	r-(HL+d)
	Compares the memory content at the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended with that of 8-bit register r.			
CMP r,(HL+C)	1110 0111 : 00rr r111	Z Z C H S V	5	r-(HL+C)
	Compares the memory content at the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended with that of 8-bit register r.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
CMP r,(+SP)	1110 0110 : 00rr r111	Z Z C H S V	4	SP ← SP+1;r-(SP)
	Compares the memory content at the address specified by the content of the stack pointer after being incremented with that of 8-bit register r.			
CMP r,(PC+A)	0100 1111 : 00rr r111	Z Z C H S V	5	r (PC+A)
	Compares the memory content at the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended with that of 8-bit register r.			
CMP (x),n	0000 0111 : xxxx xxxx : nnnn nnnn	Z Z C H S V	5	(x)-n
	Compares the memory content at the address (0000 to 00FF$_H$) specified directly by x in the object code with the immediate data n in the object code.			
CMP (vw),n	1110 0001 : wwww wwww : vvvv vvvv 0110 0111 : nnnn nnnn	Z Z C H S V	6	(vw)-n
	Compares the memory content at the address (0000 to FFFF$_H$) specified directly by vw in the object code with the immediate data n in the object code.			
CMP (DE),n	1110 0010 : 0110 0111 : nnnn nnnn	Z Z C H S V	4	(DE)-n
	Compares the memory content at the address specified by the DE register pair with the immediate data n in the object code.			
CMP (HL),n	1110 0011 : 0110 0111 : nnnn nnnn	Z Z C H S V	4	(HL)-n
	Compares the memory content at the address specified by the HL register pair with the immediate data n in the object code.			
CMP (IX),n	1110 0100 : 0110 0111 : nnnn nnnn	Z Z C H S V	4	(IX)-n
	Compares the memory content at the address specified by the index register IX with the immediate data n in the object code.			
CMP (IY),n	1110 0101 : 0110 0111 : nnnn nnnn	Z Z C H S V	4	(IY)-n
	Compares the memory content at the address specified by the index register IY with the immediate data n in the object code.			
CMP (IX+d),n	1101 0100 : dddd dddd : 0110 0111 nnnn nnnn	Z Z C H S V	6	(IX+d)-n
	Compares the memory content at the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended with the immediate data n in the object code.			
CMP (IY+d),n	1101 0101 : dddd dddd : 0110 0111 nnnn nnnn	Z Z C H S V	6	(IY+d)-n
	Compares the memory content at the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended with the immediate data n in the object code.			
CMP (SP+d),n	1101 0110 : dddd dddd : 0110 0111 nnnn nnnn	Z Z C H S V	6	(SP+d)-n
	Compares the memory content at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended with the immediate data n in the object code.			
CMP (HL+d),n	1101 0111 : dddd dddd : 0110 0111 nnnn nnnn	Z Z C H S V	6	(HL+d)-n
	Compares the memory content at the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign extended with the immediate data n in the object code.			
CMP (HL+C),n	1110 0111 : 0110 0111 : nnnn nnnn	Z Z C H S V	6	(HL+C)-n
	Compares the memory content at the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended with the immediate data n in the object code.			
CMP (+SP),n	1110 0110 : 0110 0111 : nnnn nnnn	Z Z C H S V	5	SP ← SP+1;(SP)-n
	Compares the memory content at the address specified by the content of the stack pointer after being incremented with the immediate data n in the object code.			
CMP (PC+A),n	0100 1111 : 0110 0111 : nnnn nnnn	Z Z C H S V	6	(PC+A)-n
	Compares the memory content at the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended with the immediate data n in the object code.			
CMP rr,(x)	1110 0000 : xxxx xxxx : 10rr r111	Z Z C U S V	6	rr (x)
	Compares consecutive 2-byte memory contents starting from the address (0000 to 00FF$_H$) specified directly by x in the object code with those of 16-bit register rr.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
CMP rr,(vw)	1110 0001 : wwww wwww : vvvv vvvv 10rr r111	Z Z C U S V	7	rr-(vw)
	Compares consecutive 2-byte memory contents starting from the address (0000 to $FFFF_H$) specified directly by vw in the object code with those of 16-bit register rr.			
CMP rr,(DE)	1110 0010 : 10rr r111	Z Z C U S V	5	rr-(DE)
	Compares consecutive 2-byte memory contents starting from the address specified by the DE register pair with those of 16-bit register rr.			
CMP rr,(HL)	1110 0011 : 10rr r111	Z Z C U S V	5	rr-(HL)
	Compares consecutive 2-byte memory contents starting from the address specified by the HL register pair with those of 16-bit register rr.			
CMP rr,(IX)	1110 0100 : 10rr r111	Z Z C U S V	5	rr-(IX)
	Compares consecutive 2-byte memory contents starting from the address specified by the index register IX with those of 16-bit register rr.			
CMP rr,(IY)	1110 0101 : 10rr r111	Z Z C U S V	5	rr-(IY)
	Compares consecutive 2 byte memory contents starting from the address specified by the index register IY with those of 16-bit register rr.			
CMP rr,(IX+d)	1101 0100 : dddd dddd : 10rr r111	Z Z C U S V	7	rr-(IX+d)
	Compares consecutive 2-byte memory contents starting from the address specified by the index register IX plus the object code's 8-bit data d that is added after being sign-extended with those of 16-bit register rr.			
CMP rr,(IY+d)	1101 0101 : dddd dddd : 10rr r111	Z Z C U S V	7	rr-(IY+d)
	Compares consecutive 2-byte memory contents starting from the address specified by the index register IY plus the object code's 8-bit data d that is added after being sign-extended with those of 16-bit register rr.			
CMP rr,(SP+d)	1101 0110 : dddd dddd : 10rr r111	Z Z C U S V	7	rr-(SP+d)
	Compares consecutive 2-byte memory contents starting from the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended with those of 16-bit register rr.			
CMP rr,(HL+d)	1101 0111 : dddd dddd : 10rr r111	Z Z C U S V	7	rr-(HL+d)
	Compares consecutive 2-byte memory contents starting from the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended with those of 16-bit register rr.			
CMP rr,(HL+C)	1110 0111 : 10rr r111	Z Z C U S V	7	rr-(HL+C)
	Compares consecutive 2-byte memory contents starting from the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended with those of 16-bit register rr.			
CMP rr,(+SP)	1110 0110 : 10rr r111	Z Z C U S V	6	SP ← SP+1:rr-(SP)
	Compares consecutive 2-byte memory contents starting from the address specified by the content of the stack pointer after being incremented with those of 16-bit register rr.			
CMP rr,(PC+A)	0100 1111 : 10rr r111	Z Z C U S V	7	rr-(PC+A)
	Compares consecutive 2-byte memory contents starting from the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended with those of 16-bit register rr.			
ADD A,n	0110 0001 : nnnn nnnn	C Z C H S V	2	A ← A+n
	Adds the object code's immediate data n to the accumulator content and places the result in the accumulator.			
ADD g,n	1110 1ggg : 0110 0001 : nnnn nnnn	C Z C H S V	3	g ← g+n
	Adds the object code's immediate data n to the content of 8-bit register g and places the result in the register g.			
ADD gg,mn	1110 1ggg : 0110 1001 : nnnn nnnn mmmm mmmm	C Z C U S V	4	gg ← gg+mn
	Adds the object code's immediate data mn to the content of 16-bit register gg and places the result in the register gg.			
ADD r,g	1110 1ggg : 00rr r001	C Z C H S V	2	r ← r+g
	Adds the content of 8-bit register g to that of 8-bit register r and places the result in the register r.			
ADD rr,gg	1110 1ggg : 10rr r001	C Z C U S V	4	rr ← rr+gg
	Adds the content of 16-bit register gg to that of 16-bit register rr and places the result in the register rr.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
ADD r,(x)	1110 0000 : xxxx xxxx : 00rr r001	C Z C H S V	4	r ← r+(x)
	Adds the memory content at the address (0000 to 00FF$_H$) specified directly by x in the object code to the content of 8-bit register r and places the result in the register r.			
ADD r,(vw)	1110 0001 : wwww wwww : vvvv vvvv 00rr r001	C Z C H S V	5	r ← r+(vw)
	Adds the memory content at the address (0000 to FFFF$_H$) specified directly by vw in the object code to the content of 8-bit register r and places the result in the register r.			
ADD r,(DE)	1110 0010 : 00rr r001	C Z C H S V	3	r ← r+(DE)
	Adds the memory content at the address specified by the DE register pair to the content of 8-bit register r and places the result in the register r.			
ADD r,(HL)	1110 0011 : 00rr r001	C Z C H S V	3	r ← r+(HL)
	Adds the memory content at the address specified by the HL register pair to the content of 8-bit register r and places the result in the register r.			
ADD r,(IX)	1110 0100 : 00rr r001	C Z C H S V	3	r ← r+(IX)
	Adds the memory content at the address specified by the index register IX to the content of 8-bit register r and places the result in the register r.			
ADD r,(IY)	1110 0101 : 00rr r001	C Z C H S V	3	r ← r+(IY)
	Adds the memory content at the address specified by the index register IY to the content of 8-bit register r and places the result in the register r.			
ADD r,(IX+d)	1101 0100 : dddd dddd : 00rr r001	C Z C H S V	6	r ← r+(IX+d)
	Adds the memory content at the address specified by the index register IX plus the object code's 8-bit data d that is added after being sign-extended to the content of 8-bit register r and places the result in the register r.			
ADD r,(IY+d)	1101 0101 : dddd dddd : 00rr r001	C Z C H S V	5	r ← r+(IY+d)
	Adds the memory content at the address specified by the index register IY plus the object code's 8-bit data d that is added after being sign-extended to the content of 8-bit register r and places the result in the register r.			
ADD r,(SP+d)	1101 0110 : dddd dddd : 00rr r001	C Z C H S V	5	r ← r+(SP+d)
	Adds the memory content at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended to the content of 8-bit register r and places the result in the register r.			
ADD r,(HL+d)	1101 0111 : dddd dddd : 00rr r001	C Z C H S V	5	r ← r+(HL+d)
	Adds the memory content at the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended to the content of 8-bit register r and places the result in the register r.			
ADD r,(HL+C)	1110 0111 : 00rr r001	C Z C H S V	5	r ← r+(HL+C)
	Adds the memory content at the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended to the content of 8-bit register r and places the result in the register r.			
ADD r,(+SP)	1110 0110 : 00rr r001	C Z C H S V	4	SP ← SP+1;r ← r+(SP)
	Adds the memory content at the address specified by the content of the stack pointer after being incremented to the content of 8-bit register r and places the result in the register r.			
ADD r,(PC+A)	0100 1111 : 00rr r001	C Z C H S V	5	r ← r+(PC+A)
	Adds the memory content at the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended to the content of 8-bit register r and places the result in the register r.			
ADD (x),n	1110 0000 : xxxx xxxx : 0110 0001 nnnn nnnn	C Z C H S V	6	(x) ← (x)+n
	Adds the object code's immediate data n to the memory content at the address (0000 to 00FF$_H$) specified directly by x in the object code and places the result in the said address.			
ADD (vw),n	1110 0001 : wwww wwww : vvvv vvvv 0110 0001 : nnnn nnnn	C Z C H S V	7	(vw) ← (vw)+n
	Adds the object code's immediate data n to the memory content at the address (0000 to FFFF$_H$) specified directly by vw in the object code and places the result in the said address.			
ADD (DE),n	1110 0010 : 0110 0001 : nnnn nnnn	C Z C H S V	5	(DE) ← (DE)+n
	Adds the object code's immediate data n to the memory content at the address specified by the DE register pair and places the result in the said address.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
ADD (HL),n	1110 0011 : 0110 0001 : nnnn nnnn	C Z C H S V	5	(HL) ← (HL)+n
	Adds the object code's immediate data n to the memory content at the address specified by the HL register pair and places the result in the said address.			
ADD (IX),n	1110 0100 : 0110 0001 : nnnn nnnn	C Z C H S V	6	(IX) ← (IX)+n
	Adds the object code's immediate data n to the memory content at the address specified by the index register IX and places the result in the said address.			
ADD (IY),n	1110 0101 : 0110 0001 : nnnn nnnn	C Z C H S V	5	(IY) ← (IY)+n
	Adds the object code's immediate data n to the memory content at the address specified by the index register IY and places the result in the said address.			
ADD (IX+d),n	1101 0100 : dddd dddd : 0110 0001 nnnn nnnn	C Z C H S V	7	(IX+d) ← (IX+d)+n
	Adds the object code's immediate data n to the memory content at the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended and places the result in the said address.			
ADD (IY+d),n	1101 0101 : dddd dddd : 0110 0001 nnnn nnnn	C Z C H S V	7	(IY+d) ← (IY+d)+n
	Adds the object code's immediate data n to the memory content at the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended and places the result in the said address.			
ADD (SP+d),n	1101 0110 : dddd dddd : 0110 0001 nnnn nnnn	C Z C H S V	7	(SP+d) ← (SP+d)+n
	Adds the object code's immediate data n to the memory content at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended and places the result in the said address.			
ADD (HL+d),n	1101 0111 : dddd dddd : 0110 0001 nnnn nnnn	C Z C H S V	7	(HL+d) ← (HL+d)+n
	Adds the object code's immediate data n to the memory content at the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended and places the result in the said address.			
ADD (HL+C),n	1110 0111 : 0110 0001 : nnnn nnnn	C Z C H S V	7	(HL+C) ← (HL+C)+n
	Adds the object code's immediate data n to the memory content at the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended and places the result in the said address.			
ADD (+SP),n	1110 0110 : 0110 0001 : nnnn nnnn	C Z C H S V	6	SP ← SP+1:(SP) ← (SP)+n
	Adds the object code's immediate data n to the memory content at the address specified by the content of the stack pointer after being incremented and places the result in the said address.			
ADD (PC+A),n	0100 1111 : 0110 0001 : nnnn nnnn	C Z C H S V	7	(PC+A) ← (PC+A)+n
	Adds the object code's immediate data n to the memory content at the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended and places the result in the said address.			
ADD rr,(x)	1110 0000 : xxxx xxxx : 10rr r001	C Z C U S V	6	rr ← rr+(x)
	Adds consecutive 2-byte memory contents starting from the address (0000 to 00FF$_H$) specified directly by x in the object code to the content of 16-bit register rr and places the result in the register rr.			
ADD rr,(vw)	1110 0001 : wwww wwww : vvvv vvvv 10rr r001	C Z C U S V	7	rr ← rr+(vw)
	Adds consecutive 2-byte memory contents starting from the address (0000 to FFFF$_H$) specified directly by vw in the object code to the content of 16-bit register rr and places the result in the register rr.			
ADD rr,(DE)	1110 0010 : 10rr r001	C Z C U S V	5	rr ← rr+(DE)
	Adds consecutive 2-byte memory contents starting from the address specified by the DE register pair to the content of 16-bit register rr and places the result in the register rr.			
ADD rr,(HL)	1110 0011 : 10rr r001	C Z C U S V	5	rr ← rr+(HL)
	Adds consecutive 2-byte memory contents starting from the address specified by the HL register pair to the content of 16-bit register rr and places the result in the register rr.			
ADD rr,(IX)	1110 0100 : 10rr r001	C Z C U S V	6	rr ← rr+(IX)
	Adds consecutive 2-byte memory contents starting from the address specified by the index register IX to the content of 16-bit register rr and places the result in the register rr.			
ADD rr,(IY)	1110 0101 : 10rr r001	C Z C U S V	6	rr ← rr+(IY)
	Adds consecutive 2-byte memory contents starting from the address specified by the index register IY to the content of 16-bit register rr and places the result in the register rr.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
ADD rr,(IX+d)	1101 0100 : dddd dddd : 10rr r001	C Z C U S V	7	rr ← rr+(IX+d)
	Adds consecutive 2-byte memory contents starting from the address specified by the index register IX plus the object code's 8-bit data d that is added after being sign-extended to the content of 16-bit register rr and places the result in the register rr.			
ADD rr,(IY+d)	1101 0101 : dddd dddd : 10rr r001	C Z C U S V	7	rr ← rr+(IY+d)
	Adds consecutive 2-byte memory contents starting from the address specified by the index register IY plus the object code's 8-bit data d that is added after being sign-extended to the content of 16-bit register rr and places the result in the register rr.			
ADD rr,(SP+d)	1101 0110 : dddd dddd : 10rr r001	C Z C U S V	7	rr ← rr+(SP+d)
	Adds consecutive 2-byte memory contents starting from the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended to the content of 16-bit register rr and places the result in the register rr.			
ADD rr,(HL+d)	1101 0111 : dddd dddd : 10rr r001	C Z C U S V	7	rr ← rr+(HL+d)
	Adds consecutive 2-byte memory contents starting from the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended to the content of 16-bit register rr and places the result in the register rr.			
ADD rr,(HL+C)	1110 0111 : 10rr r001	C Z C U S V	7	rr ← rr+(HL+C)
	Adds consecutive 2-byte memory contents starting from the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended to the content of 16-bit register rr and places the result in the register rr.			
ADD rr,(+SP)	1110 0110 : 10rr r001	C Z C U S V	6	SP ← SP+1:rr ← rr+(SP)
	Adds the memory content at the address specified by the content of the stack pointer after being incremented to the content of 16-bit register rr and places the result in the register rr.			
ADD rr,(PC+A)	0100 1111 : 10rr r001	C Z C U S V	7	rr ← rr+(PC+A)
	Adds consecutive 2-byte memory contents starting from the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended to the content of 16-bit register rr and places the result in the register rr.			
ADDC A,n	0110 0000 : nnnn nnnn	C Z C H S V	2	A ← A+n+CF
	Adds the object code's immediate data n and the content of the carry flag to the accumulator content and places the result in the accumulator.			
ADDC g,n	1110 1ggg : 0110 0000 : nnnn nnnn	C Z C H S V	5	g ← g+n+CF
	Adds the object code's immediate data n and the content of the carry flag to the content of 8-bit register g and places the result in the register g.			
ADDC gg,mn	1110 1ggg : 0110 1000 : nnnn nnnn mmmm mmmm	C Z C U S V	4	gg ← gg+mn+CF
	Adds the object code's immediate data mn and the content of the carry flag to the content of 16-bit register gg and places the result in the register gg.			
ADDC r,g	1110 1ggg : 00rr r000	C Z C H S V	2	r ← r+g+CF
	Adds the content of 8-bit register g and that of the carry flag to the content of 8-bit register r and places the result in the register r.			
ADDC rr,gg	1110 1ggg : 10rr r000	C Z C U S V	4	rr ← rr+gg+CF
	Adds the content of 16 bit register gg and that of the carry flag to the content of 16-bit register rr and places the result in the register rr.			
ADDC r,(x)	1110 0000 : xxxx xxxx : 00rr r000	C Z C H S V	4	r ← r+(x)+CF
	Adds the memory content at the address (0000 to 00FF$_H$) specified directly by x in the object code and the content of the carry flag to that of 8-bit register r and places the result in the register r.			
ADDC r,(vw)	1110 0001 : wwww wwww : vvvv vvvv 00rr r000	C Z C H S V	5	r ← r+(vw)+CF
	Adds the memory content at the address (0000 to FFFF$_H$) specified directly by vw in the object code and the content of the carry flag to that of 8-bit register r and places the result in the register r.			
ADDC r,(DE)	1110 0010 : 00rr r000	C Z C H S V	3	r ← r+(DE)+CF
	Adds the memory content at the address specified by the DE register pair and the content of the carry flag to that of 8-bit register r and places the result in the register r.			
ADDC r,(HL)	1110 0011 : 00rr r000	C Z C H S V	3	r ← r+(HL)+CF
	Adds the memory content at the address specified by the HL register pair and the content of the carry flag to that of 8-bit register r and places the result in the register r.			
ADDC r,(IX)	1110 0100 : 00rr r000	C Z C H S V	3	r ← r+(IX)+CF
	Adds the memory content at the address specified by the index register IX and the content of the carry flag to that of 8-bit register r and places the result in the register r.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
ADDC r,(IY)	1110 0101 : 00rr r000	C Z C H S V	3	r ← r+(IY)+CF
	Adds the memory content at the address specified by the index register IY and the content of the carry flag to that of 8-bit register r and places the result in the register r.			
ADDC r,(IX+d)	1101 0100 : dddd dddd : 00rr r000	C Z C H S V	6	r ← r+(IX+d)+CF
	Adds the memory content at the address specified by the index register IX plus the object code's 8-bit data d that is added after being sign-extended and the content of the carry flag to that of 8-bit register r and places the result in the register r.			
ADDC r,(IY+d)	1101 0101 : dddd dddd : 00rr r000	C Z C H S V	5	r ← r+(IY+d)+CF
	Adds the memory content at the address specified by the index register IY plus the object code's 8-bit data d that is added after being sign-extended and the content of the carry flag to that of 8-bit register r and places the result in the register r.			
ADDC r,(SP+d)	1101 0110 : dddd dddd : 00rr r000	C Z C H S V	5	r ← r+(SP+d)+CF
	Adds the memory content at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended and the content of the carry flag to that of 8-bit register r and places the result in the register r.			
ADDC r,(HL+d)	1101 0111 : dddd dddd : 00rr r000	C Z C H S V	5	r ← r+(HL+d)+CF
	Adds the memory content at the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign extended and the content of the carry flag to that of 8-bit register r and places the result in the register r.			
ADDC r,(HL+C)	1110 0111 : 00rr r000	C Z C H S V	5	r ← r+(HL+C)+CF
	Adds the memory content at the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended and the content of the carry flag to that of 8-bit register r and places the result in the register r.			
ADDC r,(+SP)	1110 0110 : 00rr r000	C Z C H S V	4	SP ← SP+1:r ← r+(SP)+CF
	Adds the memory content at the address specified by the content of the stack pointer after being incremented and the content of the carry flag to that of 8-bit register r and places the result in the register r.			
ADDC r,(PC+A)	0100 1111 : 00rr r000	C Z C H S V	5	r ← r+(PC+A)+CF
	Adds the memory content at the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended and the content of the carry flag to that of 8-bit register r and places the result in the register r.			
ADDC (x),n	1110 0000 : xxxx xxxx : 0110 0000 nnnn nnnn	C Z C H S V	8	(x) ← (x)+n+CF
	Adds the object code's immediate data n and the content of the carry flag to the memory content at the address (0000 to 00FF$_H$) specified directly by x in the object code and places the result in the said address.			
ADDC (vw),n	1110 0001 : wwww wwww : vvvv vvvv 0110 0000 : nnnn nnnn	C Z C H S V	7	(vw) ← (vw)+n+CF
	Adds the object code's immediate data n and the content of the carry flag to the memory content at the address (0000 to FFFF$_H$) specified directly by vw in the object code and places the result in the said address.			
ADDC (DE),n	1110 0010 : 0110 0000 : nnnn nnnn	C Z C H S V	5	(DE) ← (DE)+n+CF
	Adds the object code's immediate data n and the content of the carry flag to the memory content at the address specified by the DE register pair and places the result in the said address.			
ADDC (HL),n	1110 0011 : 0110 0000 : nnnn nnnn	C Z C H S V	5	(HL) ← (HL)+n+CF
	Adds the object code's immediate data n and the content of the carry flag to the memory content at the address specified by the HL register pair and places the result in the said address.			
ADDC (IX),n	1110 0100 : 0110 0000 : nnnn nnnn	C Z C H S V	5	(IX) ← (IX)+n+CF
	Adds the object code's immediate data n and the content of the carry flag to the memory content at the address specified by the index register IX and places the result in the said address.			
ADDC (IY),n	1110 0101 : 0110 0000 : nnnn nnnn	C Z C H S V	5	(IY) ← (IY)+n+CF
	Adds the object code's immediate data n and the content of the carry flag to the memory content at the address specified by the index register IY and places the result in the said address.			
ADDC (IX+d),n	1101 0100 : dddd dddd : 0110 0000 nnnn nnnn	C Z C H S V	7	(IX+d) ← (IX+d)+n+CF
	Adds the object code's immediate data n and the content of the carry flag to the memory content at the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended and places the result in the said address.			
ADDC (IY+d),n	1101 0101 : dddd dddd : 0110 0000 nnnn nnnn	C Z C H S V	7	(IY+d) ← (IY+d)+n+CF
	Adds the object code's immediate data n and the content of the carry flag to the memory content at the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended and places the result in the said address.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
ADDC (SP+d),n	1101 0110 dddd dddd 0110 0000 nnnn nnnn	C Z C H S V	7	(SP+d) ← (SP+d)+n+CF
	Adds the object code's immediate data n and the content of the carry flag to the memory content at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended and places the result in the said address.			
ADDC (HL+d),n	1101 0111 dddd dddd 0110 0000 nnnn nnnn	C Z C H S V	7	(HL+d) ← (HL+d)+n+CF
	Adds the object code's immediate data n and the content of the carry flag to the memory content at the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended and places the result in the said address.			
ADDC (HL+C),n	1110 0111 0110 0000 nnnn nnnn	C Z C H S V	7	(HL+C) ← (HL+C)+n+CF
	Adds the object code's immediate data n and the content of the carry flag to the memory content at the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended and places the result in the said address.			
ADDC (+SP),n	1110 0110 0110 0000 nnnn nnnn	C Z C H S V	6	SP ← SP+1:(SP) ← SP+n+CF
	Adds the object code's immediate data n and the content of the carry flag to the memory content at the address specified by the content of the stack pointer after being incremented and places the result in the said address.			
ADDC (PC+A),n	0100 1111 0110 0000 nnnn nnnn	C Z C H S V	7	(PC+A) ← (PC+A)+n+CF
	Adds the object code's immediate data n and the content of the carry flag to the memory content at the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended and places the result in the said address.			
ADDC rr,(x)	1110 0000 xxxx xxxx 10rr r000	C Z C U S V	6	rr ← rr+(x)+CF
	Adds consecutive 2-byte memory contents starting from the address (0000 to 00FF$_H$) specified directly by x in the object code and the content of the carry flag to that of 16-bit register rr and places the result in the register rr.			
ADDC rr,(vw)	1110 0001 wwww wwww vvvv vvvv 10rr r000	C Z C U S V	7	rr ← rr+(vw)+CF
	Adds consecutive 2-byte memory contents starting from the address (0000 to FFFF$_H$) specified directly by vw in the object code and the content of the carry flag to that of 16-bit register rr and places the result in the register rr.			
ADDC rr,(DE)	1110 0010 10rr r000	C Z C U S V	5	rr ← rr+(DE)+CF
	Adds consecutive 2-byte memory contents starting from the address specified by the DE register pair and the content of the carry flag to that of 16-bit register rr and places the result in the register rr.			
ADDC rr,(HL)	1110 0011 10rr r000	C Z C U S V	5	rr ← rr+(HL)+CF
	Adds consecutive 2-byte memory contents starting from the address specified by the HL register pair and the content of the carry flag to that of 16-bit register rr and places the result in the register rr.			
ADDC rr,(IX)	1110 0100 10rr r000	C Z C U S V	5	rr ← rr+(IX)+CF
	Adds consecutive 2-byte memory contents starting from the address specified by the index register IX and the content of the carry flag to that of 16-bit register rr and places the result in the register rr.			
ADDC rr,(IY)	1110 0101 10rr r000	C Z C U S V	5	rr ← rr+(IY)+CF
	Adds consecutive 2-byte memory contents starting from the address specified by the index register IY and the content of the carry flag to that of 16-bit register rr and places the result in the register rr.			
ADDC rr,(IX+d)	1101 0100 dddd dddd 10rr r000	C Z C U S V	7	rr ← rr+(IX+d)+CF
	Adds consecutive 2-byte memory contents starting from the address specified by the index register IX plus the object code's 8-bit data d that is added after being sign-extended and the content of the carry flag to that of 16-bit register rr and places the result in the register rr.			
ADDC rr,(IY+d)	1101 0101 dddd dddd 10rr r000	C Z C U S V	7	rr ← rr+(IY+d)+CF
	Adds consecutive 2-byte memory contents starting from the address specified by the index register IY plus the object code's 8-bit data d that is added after being sign-extended and the content of the carry flag to that of 16-bit register rr and places the result in the register rr.			
ADDC rr,(SP+d)	1101 0110 dddd dddd 10rr r000	C Z C U S V	7	rr ← rr+(SP+d)+CF
	Adds consecutive 2-byte memory contents starting from the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended and the content of the carry flag to that of 16-bit register rr and places the result in the register rr.			
ADDC rr,(HL+d)	1101 0111 dddd dddd 10rr r000	C Z C U S V	7	rr ← rr+(HL+d)+CF
	Adds consecutive 2-byte memory contents starting from the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended and the content of the carry flag to that of 16-bit register rr and places the result in the register rr.			
ADDC rr,(HL+C)	1110 0111 10rr r000	C Z C U S V	7	rr ← rr+(HL+C)+CF
	Adds consecutive 2-byte memory contents starting from the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended and the content of the carry flag to that of 16-bit register rr and places the result in the register rr.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
ADDC rr,(+SP)	1110 0110 : 10rr r000	C Z C U S V	6	SP ← SP+1;rr ← rr+(SP)
	Adds the memory content at the address specified by the content of the stack pointer after being incremented and the content of the carry flag to that of 16-bit register rr and places the result in the register rr.			
ADDC rr,(PC+A)	0100 1111 : 10rr r000	C Z C U S V	7	rr ← rr+(PC+A)+CF
	Adds consecutive 2-byte memory contents starting from the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended and the content of the carry flag to that of 16-bit register rr and places the result in the register rr.			
SUB A,n	0110 0011 : nnnn nnnn	C Z C H S V	2	A ← A-n
	Subtracts the object code's immediate data n from the accumulator content and places the result in the accumulator.			
SUB g,n	1110 1ggg : 0110 0011 : nnnn nnnn	C Z C H S V	3	g ← g-n
	Subtracts the object code's immediate data n from the content of 8 bit register g and places the result in the register g.			
SUB gg,mn	1110 1ggg : 0110 1011 : nnnn nnnn : mmmm mmmm	C Z C U S V	4	gg ← gg-mn
	Subtracts the object code's immediate data mn from the content of 16-bit register gg and places the result in the register gg.			
SUB r,g	1110 1ggg : 00rr r011	C Z C H S V	2	r ← r-g
	Subtracts the content of 8-bit register g from the content of 8-bit register r and places the result in the register r.			
SUB rr,gg	1110 1ggg : 10rr r011	C Z C U S V	4	rr ← rr-gg
	Subtracts the content of 16-bit register gg from the content of 16-bit register rr and places the result in the register rr.			
SUB r,(x)	1110 0000 : xxxx xxxx : 00rr r011	C Z C H S V	4	r ← r-(x)
	Subtracts the memory content at the address (0000 to 00FF$_H$) specified directly by x in the object code from the content of 8-bit register r and places the result in the register r.			
SUB r,(vw)	1110 0001 : wwww wwww : vvvv vvvv : 00rr r011	C Z C H S V	5	r ← r-(vw)
	Subtracts the memory content at the address (0000 to FFFF$_H$) specified directly by vw in the object code from the content of 8-bit register r and places the result in the register r.			
SUB r,(DE)	1110 0010 : 00rr r011	C Z C H S V	3	r ← r-(DE)
	Subtracts the memory content at the address specified by the DE register pair from the content of 8-bit register r and places the result in the register r.			
SUB r,(HL)	1110 0011 : 00rr r011	C Z C H S V	3	r ← r-(HL)
	Subtracts the memory content at the address specified by the HL register pair from the content of 8-bit register r and places the result in the register r.			
SUB r,(IX)	1110 0100 : 00rr r011	C Z C H S V	3	r ← r-(IX)
	Subtracts the memory content at the address specified by the index register IX from the content of 8-bit register r and places the result in the register r.			
SUB r,(IY)	1110 0101 : 00rr r011	C Z C H S V	3	r ← r-(IY)
	Subtracts the memory content at the address specified by the index register IY from the content of 8-bit register r and places the result in the register r.			
SUB r,(IX+d)	1101 0100 : dddd dddd : 00rr r011	C Z C H S V	6	r ← r-(IX+d)
	Subtracts the memory content at the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended from the content of 8-bit register r and places the result in the register r.			
SUB r,(IY+d)	1101 0101 : dddd dddd : 00rr r011	C Z C H S V	5	r ← r-(IY+d)
	Subtracts the memory content at the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended from the content of 8-bit register r and places the result in the register r.			
SUB r,(SP+d)	1101 0110 : dddd dddd : 00rr r011	C Z C H S V	5	r ← r-(SP+d)
	Subtracts the memory content at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended from the content of 8-bit register r and places the result in the register r.			
SUB r,(HL+d)	1101 0111 : dddd dddd : 00rr r011	C Z C H S V	5	r ← r-(HL+d)
	Subtracts the memory content at the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended from the content of 8-bit register r and places the result in the register r.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
SUB r,(HL+C)	1110 0111 : 00rr r011	C Z C H S V	5	r ← r-(HL+C)
	Subtracts the memory content at the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended from the content of 8-bit register r and places the result in the register r.			
SUB r,(+SP)	1110 0110 : 00rr r011	C Z C H S V	4	SP ← SP+1:r ← r-(SP)
	Subtracts the memory content at the address specified by the content of the stack pointer after being incremented from the content of 8-bit register r and places the result in the register r.			
SUB r,(PC+A)	0100 1111 : 00rr r011	C Z C H S V	5	r ← r-(PC+A)
	Subtracts the memory content at the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended from the content of 8-bit register r and places the result in the register r.			
SUB (x),n	1110 0000 : xxxx xxxx : 0110 0011 : nnnn nnnn	C Z C H S V	6	(x) ← (x)-n
	Subtracts the object code's immediate data n from the memory content at the address (0000 to 00FF$_H$) specified directly by x in the object code and places the result in the said address.			
SUB (vw),n	1110 0001 : wwww wwww : vvvv vvvv : 0110 0011 : nnnn nnnn	C Z C H S V	7	(vw) ← (vw)-n
	Subtracts the object code's immediate data n from the memory content at the address (0000 to FFFF$_H$) specified directly by vw in the object code and places the result in the said address.			
SUB (DE),n	1110 0010 : 0110 0011 : nnnn nnnn	C Z C H S V	5	(DE) ← (DE)-n
	Subtracts the object code's immediate data n from the memory content at the address specified by the DE register pair and places the result in the said address.			
SUB (HL),n	1110 0011 : 0110 0011 : nnnn nnnn	C Z C H S V	5	(HL) ← (HL)-n
	Subtracts the object code's immediate data n from the memory content at the address specified by the HL register pair and places the result in the said address.			
SUB (IX),n	1110 0100 : 0110 0011 : nnnn nnnn	C Z C H S V	5	(IX) ← (IX)-n
	Subtracts the object code's immediate data n from the memory content at the address specified by the index register IX and places the result in the said address.			
SUB (IY),n	1110 0101 : 0110 0011 : nnnn nnnn	C Z C H S V	5	(IY) ← (IY)-n
	Subtracts the object code's immediate data n from the memory content at the address specified by the index register IY and places the result in the said address.			
SUB (IX+d),n	1101 0100 : dddd dddd : 0110 0011 : nnnn nnnn	C Z C H S V	7	(IX+d) ← (IX+d)-n
	Subtracts the object code's immediate data n from the memory content at the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended and places the result in the said address.			
SUB (IY+d),n	1101 0101 : dddd dddd : 0110 0011 : nnnn nnnn	C Z C H S V	7	(IY+d) ← (IY+d)-n
	Subtracts the object code's immediate data n from the memory content at the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended and places the result in the said address.			
SUB (SP+d),n	1101 0110 : dddd dddd : 0110 0011 : nnnn nnnn	C Z C H S V	7	(SP+d) ← (SP+d)-n
	Subtracts the object code's immediate data n from the memory content at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended and places the result in the said address.			
SUB (HL+d),n	1101 0111 : dddd dddd : 0110 0011 : nnnn nnnn	C Z C H S V	7	(HL+d) ← (HL+d)-n
	Subtracts the object code's immediate data n from the memory content at the address specified by the content of the HL register pair plus the object code's 8 bit data d that is added after being sign-extended and places the result in the said address.			
SUB (HL+C),n	1110 0111 : 0110 0011 : nnnn nnnn	C Z C H S V	7	(HL+C) ← (HL+C)-n
	Subtracts the object code's immediate data n from the memory content at the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended and places the result in the said address.			
SUB (+SP),n	1110 0110 : 0110 0011 : nnnn nnnn	C Z C H S V	6	SP ← SP+1:(SP) ← (SP)-n
	Subtracts the object code's immediate data n from the memory content at the address specified by the content of the stack pointer after being incremented and places the result in the said address.			
SUB (PC+A),n	0100 1111 : 0110 0011 : nnnn nnnn	C Z C H S V	7	(PC+A) ← (PC+A)-n
	Subtracts the object code's immediate data n from the memory content at the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended and places the result in the said address.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
SUB rr,(x)	1110 0000 : xxxx xxxx : 10rr r011	C Z C U S V	6	rr ← rr-(x)
	Subtracts consecutive 2-byte memory contents beginning with the address (0000 to 00FF$_H$) specified directly by x in the object code from the content of 16-bit register rr and places the result in the register rr.			
SUB rr,(vw)	1110 0001 : wwww wwww : vvvv vvvv : 10rr r011	C Z C U S V	7	rr ← rr-(vw)
	Subtracts consecutive 2-byte memory contents beginning with the address (0000 to FFFF$_H$) specified directly by vw in the object code from the content of 16-bit register rr and places the result in the register rr.			
SUB rr,(DE)	1110 0010 : 10rr r011	C Z C U S V	5	rr ← rr-(DE)
	Subtracts consecutive 2-byte memory contents beginning with the address specified by the DE register pair from the content of 16-bit register rr and places the result in the register rr.			
SUB rr,(HL)	1110 0011 : 10rr r011	C Z C U S V	5	rr ← rr-(HL)
	Subtracts consecutive 2-byte memory contents beginning with the address specified by the HL register pair from the content of 16-bit register rr and places the result in the register rr.			
SUB rr,(IX)	1110 0100 : 10rr r011	C Z C U S V	5	rr ← rr-(IX)
	Subtracts consecutive 2 byte memory contents beginning with the address specified by the index register IX from the content of 16-bit register rr and places the result in the register rr.			
SUB rr,(IY)	1110 0101 : 10rr r011	C Z C U S V	5	rr ← rr-(IY)
	Subtracts consecutive 2-byte memory contents beginning with the address specified by the index register IY from the content of 16-bit register rr and places the result in the register rr.			
SUB rr,(IX+d)	1101 0100 : dddd dddd : 10rr r011	C Z C U S V	7	rr ← rr-(IX+d)
	Subtracts consecutive 2-byte memory contents beginning with the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended from the content of 16-bit register rr and places the result in the register rr.			
SUB rr,(IY+d)	1101 0101 : dddd dddd : 10rr r011	C Z C U S V	7	rr ← rr-(IY+d)
	Subtracts consecutive 2-byte memory contents beginning with the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended from the content of 16-bit register rr and places the result in the register rr.			
SUB rr,(SP+d)	1101 0110 : dddd dddd : 10rr r011	C Z C U S V	7	rr ← rr-(SP+d)
	Subtracts consecutive 2-byte memory contents beginning with the address specified by the content of the stack pointer plus the object code's 8 bit data d that is added after being sign-extended from the content of 16-bit register rr and places the result in the register rr.			
SUB rr,(HL+d)	1101 0111 : dddd dddd : 10rr r011	C Z C U S V	7	rr ← rr-(HL+d)
	Subtracts consecutive 2-byte memory contents beginning with the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended from the content of 16-bit register rr and places the result in the register rr.			
SUB rr,(HL+C)	1110 0111 : 10rr r011	C Z C U S V	7	rr ← rr-(HL+C)
	Subtracts consecutive 2-byte memory contents beginning with the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended from the content of 16-bit register rr and places the result in the register rr.			
SUB rr,(+SP)	1110 0110 : 10rr r011	C Z C U S V	6	SP ← SP+1:rr ← rr-(SP)
	Subtracts consecutive 2-byte memory contents beginning with the address specified by the content of the stack pointer after being incremented from the content of 16-bit register rr and places the result in the register rr.			
SUB rr,(PC+A)	0100 1111 : 10rr r011	C Z C U S V	7	rr ← rr-(PC+A)
	Subtracts consecutive 2-byte memory contents beginning with the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended from the content of 16-bit register rr and places the result in the register rr.			
SUBB A,n	0110 0010 : nnnn nnnn	C Z C H S V	2	A ← A-n-CF
	Subtracts the object code's immediate data n and the carry flag content from the accumulator content and places the result in the accumulator.			
SUBB g,n	1110 1ggg : 0110 0010 : nnnn nnnn	C Z C H S V	3	g ← g-n-CF
	Subtracts the object code's immediate data n and the carry flag content from the content of 8-bit register g and places the result in the register g.			
SUBB gg,mn	1110 1ggg : 0110 1010 : nnnn nnnn : mmmm mmmm	C Z C U S V	4	gg ← gg-mn-CF
	Subtracts the object code's immediate data mn and the carry flag content from the content of 16-bit register gg and places the result in the register gg.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
SUBB r,g	1110 1ggg : 00rr r010	C Z C H S V	2	r ← r-g-CF
	Subtracts the content of 8-bit register g and the carry flag content from the content of 8-bit register r and places the result in the register r.			
SUBB rr,gg	1110 1ggg : 10rr r010	C Z C U S V	4	rr ← rr-gg-CF
	Subtracts the content of 16-bit register gg and the carry flag content from the content of 16-bit register rr and places the result in the register rr.			
SUBB r,(x)	1110 0000 : xxxx xxxx : 00rr r010	C Z C H S V	4	r ← r-(x)-CF
	Subtracts the memory content at the address (0000 to 00FFH) specified directly by x in the object code and the carry flag content from the content of 8-bit register r and places the result in the register r.			
SUBB r,(vw)	1110 0001 : wwww wwww : vvvv vvvv 00rr r010	C Z C H S V	5	r ← r-(vw)-CF
	Subtracts the memory content at the address (0000 to FFFFH) specified directly by vw in the object code and the carry flag content from the content of 8-bit register r and places the result in the register r.			
SUBB r,(DE)	1110 0010 : 00rr r010	C Z C H S V	3	r ← r-(DE)-CF
	Subtracts the memory content at the address specified by the DE register pair and the carry flag content from the content of 8-bit register r and places the result in the register r.			
SUBB r,(HL)	1110 0011 : 00rr r010	C Z C H S V	3	r ← r-(HL)-CF
	Subtracts the memory content at the address specified by the HL register pair and the carry flag content from the content of 8-bit register r and places the result in the register r.			
SUBB r,(IX)	1110 0100 : 00rr r010	C Z C H S V	3	r ← r-(IX)-CF
	Subtracts the memory content at the address specified by the index register IX and the carry flag content from the content of 8-bit register r and places the result in the register r.			
SUBB r,(IY)	1110 0101 : 00rr r010	C Z C H S V	3	r ← r-(IY)-CF
	Subtracts the memory content at the address specified by the index register IY and the carry flag content from the content of 8-bit register r and places the result in the register r.			
SUBB r,(IX+d)	1101 0100 : dddd dddd : 00rr r010	C Z C H S V	5	r ← r-(IX+d)-CF
	Subtracts the memory content at the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended and the carry flag content from the content of 8-bit register r and places the result in the register r.			
SUBB r,(IY+d)	1101 0101 : dddd dddd : 00rr r010	C Z C H S V	5	r ← r-(IY+d)-CF
	Subtracts the memory content at the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended and the carry flag content from the content of 8-bit register r and places the result in the register r.			
SUBB r,(SP+d)	1101 0110 : dddd dddd : 00rr r010	C Z C H S V	5	r ← r-(SP+d)-CF
	Subtracts the memory content at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended and the carry flag content from the content of 8-bit register r and places the result in the register r.			
SUBB r,(HL+d)	1101 0111 : dddd dddd : 00rr r010	C Z C H S V	5	r ← r-(HL+d)-CF
	Subtracts the memory content at the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended and the carry flag content from the content of 8-bit register r and places the result in the register r.			
SUBB r,(HL+C)	1110 0111 : 00rr r010	C Z C H S V	5	r ← r-(HL+C)-CF
	Subtracts the memory content at the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended and the carry flag content from the content of 8-bit register r and places the result in the register r.			
SUBB r,(+SP)	1110 0110 : 00rr r010	C Z C H S V	4	SP ← SP+1:r ← r-(SP)-CF
	Subtracts the memory content at the address specified by the content of the stack pointer after being incremented and the carry flag content from the content of 8-bit register r and places the result in the register r.			
SUBB r,(PC+A)	0100 1111 : 00rr r010	C Z C H S V	5	r ← r-(PC+A)-CF
	Subtracts the memory content at the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended and the carry flag content from the content of 8-bit register r and places the result in the register r.			
SUBB (x),n	1110 0000 : xxxx xxxx : 0110 0010 nnnn nnnn	C Z C H S V	6	(x) ← (x)-n-CF
	Subtracts the object code's immediate data n and the carry flag content from the memory content at the address (0000 to 00FF$_H$) specified directly by x in the object code and places the result in the said address.			
SUBB (vw),n	1110 0001 : wwww wwww : vvvv vvvv 0110 0010 : nnnn nnnn	C Z C H S V	7	(vw) ← (vw)-n-CF
	Subtracts the object code's immediate data n and the carry flag content from the memory content at the address (0000 to FFFF$_H$) specified directly by vw in the object code and places the result in the said address.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
SUBB (DE),n	1110 0010 : 0110 0010 : nnnn nnnn	C Z C H S V	5	(DE) ← (DE)-n-CF
	Subtracts the object code's immediate data n and the carry flag content from the memory content at the address specified by the DE register pair and places the result in the said address.			
SUBB (HL),n	1110 0011 : 0110 0010 : nnnn nnnn	C Z C H S V	5	(HL) ← (HL) n CF
	Subtracts the object code's immediate data n and the carry flag content from the memory content at the address specified by the HL register pair and places the result in the said address.			
SUBB (IX),n	1110 0100 : 0110 0010 : nnnn nnnn	C Z C H S V	5	(IX) ← (IX)-n-CF
	Subtracts the object code's immediate data n and the carry flag content from the memory content at the address specified by the index register IX and places the result in the said address.			
SUBB (IY),n	1110 0101 : 0110 0010 : nnnn nnnn	C Z C H S V	5	(IY) ← (IY)-n-CF
	Subtracts the object code's immediate data n and the carry flag content from the memory content at the address specified by the index register IY and places the result in the said address.			
SUBB (IX+d),n	1101 0100 : dddd dddd : 0110 0010 : nnnn nnnn	C Z C H S V	7	(IX+d) ← (IX+d)-n-CF
	Subtracts the object code's immediate data n and the carry flag content from the memory content at the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended and places the result in the said address.			
SUBB (IY+d),n	1101 0101 : dddd dddd : 0110 0010 : nnnn nnnn	C Z C H S V	7	(IY+d) ← (IY+d)-n-CF
	Subtracts the object code's immediate data n and the carry flag content from the memory content at the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended and places the result in the said address.			
SUBB (SP+d),n	1101 0110 : dddd dddd : 0110 0010 : nnnn nnnn	C Z C H S V	7	(SP+d) ← (SP+d)-n-CF
	Subtracts the object code's immediate data n and the carry flag content from the memory content at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended and places the result in the said address.			
SUBB (HL+d),n	1101 0111 : dddd dddd : 0110 0010 : nnnn nnnn	C Z C H S V	7	(HL+d) ← (HL+d)-n-CF
	Subtracts the object code's immediate data n and the carry flag content from the memory content at the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended and places the result in the said address.			
SUBB (HL+C),n	1110 0111 : 0110 0010 : nnnn nnnn	C Z C H S V	7	(HL+C) ← (HL+C)-n-CF
	Subtracts the object code's immediate data n and the carry flag content from the memory content at the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended and places the result in the said address.			
SUBB (+SP),n	1110 0110 : 0110 0010 : nnnn nnnn	C Z C H S V	6	SP ← SP+1:(SP) ← SP-n-CF
	Subtracts the object code's immediate data n and the carry flag content from the memory content at the address specified by the content of the stack pointer after being incremented and places the result in the said address.			
SUBB (PC+A),n	0100 1111 : 0110 0010 : nnnn nnnn	C Z C H S V	7	(PC+A) ← (PC+A)-n-CF
	Subtracts the object code's immediate data n and the carry flag content from the memory content at the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended and places the result in the said address.			
SUBB rr,(x)	1110 0000 : xxxx xxxx : 10rr r010	C Z C U S V	6	rr ← rr-(x)-CF
	Subtracts consecutive 2-byte memory contents beginning with the address (0000 to 00FF$_H$) specified directly by x in the object code and the carry flag content from the content of 16-bit register rr and places the result in the register rr.			
SUBB rr,(vw)	1110 0001 : wwww wwww : vvvv vvvv : 10rr r010	C Z C U S V	7	rr ← rr-(vw)-CF
	Subtracts consecutive 2-byte memory contents beginning with the address (0000 to FFFF$_H$) specified directly by vw in the object code and the carry flag content from the content of 16-bit register rr and places the result in the register rr.			
SUBB rr,(DE)	1110 0010 : 10rr r010	C Z C U S V	5	rr ← rr-(DE)-CF
	Subtracts consecutive 2-byte memory contents beginning with the address specified by the DE register pair and the carry flag content from the content of 16-bit register rr and places the result in the register rr.			
SUBB rr,(HL)	1110 0011 : 10rr r010	C Z C U S V	5	rr ← rr-(HL)-CF
	Subtracts consecutive 2-byte memory contents beginning with the address specified by the HL register pair and the carry flag content from the content of 16-bit register rr and places the result in the register rr.			
SUBB rr,(IX)	1110 0100 : 10rr r010	C Z C U S V	5	rr ← rr-(IX)-CF
	Subtracts consecutive 2-byte memory contents beginning with the address specified by the index register IX and the carry flag content from the content of 16-bit register rr and places the result in the register rr.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
SUBB rr,(IY)	1110 0101 : 10rr r010	C Z C U S V	5	rr ← rr-(IY)-CF
	Subtracts consecutive 2-byte memory contents beginning with the address specified by the index register IY and the carry flag content from the content of 16-bit register rr and places the result in the register rr.			
SUBB rr,(IX+d)	1101 0100 : dddd dddd : 10rr r010	C Z C U S V	7	rr ← rr-(IX+d)-CF
	Subtracts consecutive 2-byte memory contents beginning with the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended and the carry flag content from the content of 16-bit register rr and places the result in the register rr.			
SUBB rr,(IY+d)	1101 0101 : dddd dddd : 10rr r010	C Z C U S V	7	rr ← rr-(IY+d)-CF
	Subtracts consecutive 2-byte memory contents beginning with the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended and the carry flag content from the content of 16-bit register rr and places the result in the register rr.			
SUBB rr,(SP+d)	1101 0110 : dddd dddd : 10rr r010	C Z C U S V	7	rr ← rr-(SP+d)-CF
	Subtracts consecutive 2-byte memory contents beginning with the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended and the carry flag content from the content of 16-bit register rr and places the result in the register rr.			
SUBB rr,(HL+d)	1101 0111 : dddd dddd : 10rr r010	C Z C U S V	7	rr ← rr-(HL+d)-CF
	Subtracts consecutive 2-byte memory contents beginning with the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended and the carry flag content from the content of 16-bit register rr and places the result in the register rr.			
SUBB rr,(HL+C)	1110 0111 : 10rr r010	C Z C U S V	7	rr ← rr-(HL+C)-CF
	Subtracts consecutive 2-byte memory contents beginning with the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended and the carry flag content from the content of 16-bit register rr and places the result in the register			
SUBB rr,(+SP)	1110 0110 : 10rr r010	C Z C U S V	6	SP ← SP+1:rr ← rr-(SP)-CF
	Subtracts consecutive 2-byte memory contents beginning with the address specified by the content of the stack pointer after being incremented and the carry flag content from the content of 16-bit register rr and places the result in the register rr.			
SUBB rr,(PC+A)	0100 1111 : 10rr r010 :	C Z C U S V	7	rr ← rr-(PC+A)-CF
	Subtracts consecutive 2-byte memory contents beginning with the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended and the carry flag content from the content of 16-bit register rr and places the result in the register rr.			
AND A,n	0110 0100 : nnnn nnnn :	Z Z - - - -	2	A ← A∧n
	Logically ANDs the accumulator content bitwise (or bit-by-bit) with the object code's immediate data n and places the result in the accumulator.			
AND g,n	1110 1ggg : 0110 0100 : nnnn nnnn	Z Z - - - -	3	g ← g∧n
	Logically ANDs the content of 8-bit register g bitwise with the object code's immediate data n and places the result in the register g.			
AND gg,mn	1110 1ggg : 0110 1100 : nnnn nnnn : mmmm mmmm	Z Z - - - -	4	gg ← gg∧mn
	Logically ANDs the content of 16-bit register gg bitwise with the object code's immediate data mn and places the result in the register gg.			
AND r,g	1110 1ggg : 00rr r100 :	Z Z - - - -	2	r ← r∧g
	Logically ANDs the content of 8-bit register r bitwise with that of 8-bit register g and places the result in the register r.			
AND rr,gg	1110 1ggg : 10rr r100 :	Z Z - - - -	4	rr ← rr∧gg
	Logically ANDs the content of 16-bit register rr bitwise with that of 16-bit register gg and places the result in the register rr.			
AND r,(x)	1110 0000 : xxxx xxxx : 00rr r100	Z Z - - - -	4	r ← r∧(x)
	Logically ANDs the content of 8-bit register r bitwise with the memory content at the address (0000 to $00FF_H$) specified directly by x in the object code and places the result in the register r.			
AND r,(vw)	1110 0001 : wwww wwww : vvvv vvvv	Z Z - - - -	5	r ← r∧(vw)
	Logically ANDs the content of 8-bit register r bitwise with the memory content at the address (0000 to $FFFF_H$) specified directly by vw in the object code and places the result in the register r.			
AND r,(DE)	1110 0010 : 00rr r100 :	Z Z - - - -	3	r ← r∧(DE)
	Logically ANDs the content of 8-bit register r bitwise with the memory content at the address specified by the DE register pair and places the result in the register r.			
AND r,(HL)	1110 0011 : 00rr r100 :	Z Z - - - -	3	r ← r∧(HL)
	Logically ANDs the content of 8-bit register r bitwise with the memory content at the address specified by the HL register pair and places the result in the register r.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
AND r,(IX)	1110 0100 : 00rr r100	Z Z - - - -	3	r ← r∧(IX)
	Logically ANDs the content of 8-bit register r bitwise with the memory content at the address specified by the index register IX and places the result in the register r.			
AND r,(IY)	1110 0101 : 00rr r100	Z Z - - - -	3	r ← r∧(IY)
	Logically ANDs the content of 8-bit register r bitwise with the memory content at the address specified by the index register IY and places the result in the register r.			
AND r,(IX+d)	1101 0100 : dddd dddd : 00rr r100	Z Z - - - -	5	r ← r∧(IX+d)
	Logically ANDs the content of 8-bit register r bitwise with the memory content at the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended and places the result in the register r.			
AND r,(IY+d)	1101 0101 : dddd dddd : 00rr r100	Z Z - - - -	5	r ← r∧(IY+d)
	Logically ANDs the content of 8-bit register r bitwise with the memory content at the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended and places the result in the register r.			
AND r,(SP+d)	1101 0110 : dddd dddd : 00rr r100	Z Z - - - -	5	r ← r∧(SP+d)
	Logically ANDs the content of 8-bit register r bitwise with the memory content at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended and places the result in the register r.			
AND r,(HL+d)	1101 0111 : dddd dddd : 00rr r100	Z Z - - - -	5	r ← r∧(HL+d)
	Logically ANDs the content of 8-bit register r bitwise with the memory content at the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended and places the result in the register r.			
AND r,(HL+C)	1110 0111 : 00rr r100	Z Z - - - -	5	r ← r∧(HL+C)
	Logically ANDs the content of 8-bit register r bitwise with the memory content at the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended and places the result in the register r.			
AND r,(+SP)	1110 0110 : 00rr r100	Z Z - - - -	4	SP ← SP+1 : r ← r∧(SP)
	Logically ANDs the content of 8-bit register r bitwise with the memory content at the address specified by the content of the stack pointer after being incremented and places the result in the register r.			
AND r,(PC+A)	0100 1111 : 00rr r100	Z Z - - - -	5	r ← r∧(PC+A)
	Logically ANDs the content of 8-bit register r bitwise with the memory content at the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended and places the result in the register r.			
AND (x),n	1110 0000 : xxxx xxxx : 0110 0100 : nnnn nnnn	Z Z - - - -	6	(x) ← (x)∧n
	Logically ANDs the memory content at the address (0000 to 00FF$_H$) specified directly by x in the object code bitwise with the object code's immediate data n and places the result in the said address.			
AND (vw),n	1110 0001 : wwww wwww : vvvv vvvv : 0110 0100 : nnnn nnnn	Z Z - - - -	7	(vw) ← (vw)∧n
	Logically ANDs the memory content at the address (0000 to FFFF$_H$) specified directly by vw in the object code bitwise with the object code's immediate data n and places the result in the said address.			
AND (DE),n	1110 0010 : 0110 0100 : nnnn nnnn	Z Z - - - -	5	(DE) ← (DE)∧n
	Logically ANDs the memory content at the address specified by the DE register pair bitwise with the object code's immediate data n and places the result in the said address.			
AND (HL),n	1110 0011 : 0110 0100 : nnnn nnnn	Z Z - - - -	5	(HL) ← (HL)∧n
	Logically ANDs the memory content at the address specified by the HL register pair bitwise with the object code's immediate data n and places the result in the said address.			
AND (IX),n	1110 0100 : 0110 0100 : nnnn nnnn	Z Z - - - -	5	(IX) ← (IX)∧n
	Logically ANDs the memory content at the address specified by the index register IX bitwise with the object code's immediate data n and places the result in the said address.			
AND (IY),n	1110 0101 : 0110 0100 : nnnn nnnn	Z Z - - - -	5	(IY) ← (IY)∧n
	Logically ANDs the memory content at the address specified by the index register IY bitwise with the object code's immediate data n and places the result in the said address.			
AND (IX+d),n	1101 0100 : dddd dddd : 0110 0100 : nnnn nnnn	Z Z - - - -	7	(IX+d) ← (IX+d)∧n
	Logically ANDs the memory content at the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended bitwise with the object code's immediate data n and places the result in the said address.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
AND (IY+d),n	1101 0101 : dddd dddd : 0110 0100 : nnnn nnnn	Z Z - - - -	7	(IY+d) ← (IY+d)∧n
	Logically ANDs the memory content at the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended bitwise with the object code's immediate data n and places the result in the said address.			
AND (SP+d),n	1101 0110 : dddd dddd : 0110 0100 : nnnn nnnn	Z Z - - - -	7	(SP+d) ← (SP+d)∧n
	Logically ANDs the memory content at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended bitwise with the object code's immediate data n and places the result in the said address.			
AND (HL+d),n	1101 0111 : dddd dddd : 0110 0100 : nnnn nnnn	Z Z - - - -	7	(HL+d) ← (HL+d)∧n
	Logically ANDs the memory content at the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended bitwise with the object code's immediate data n and places the result in the said address.			
AND (HL+C),n	1110 0111 : 0110 0100 : nnnn nnnn	Z Z - - - -	7	(HL+C) ← (HL+C)∧n
	Logically ANDs the memory content at the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended bitwise with the object code's immediate data n and places the result in the said address.			
AND (+SP),n	1110 0110 : 0110 0100 : nnnn nnnn	Z Z - - - -	6	SP ← SP+1:(SP) ← (SP)∧n
	Logically ANDs the memory content at the address specified by the content of the stack pointer after being incremented bitwise with the object code's immediate data n and places the result in the said address.			
AND (PC+A),n	0100 1111 : 0110 0100 : nnnn nnnn	Z Z - - - -	7	(PC+A) ← (PC+A)∧n
	Logically ANDs the memory content at the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended bitwise with the object code's immediate data n and places the result in the said address.			
AND rr,(x)	1110 0000 : xxxx xxxx : 10rr r100	Z Z - - - -	6	rr ← rr∧(x)
	Logically ANDs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address (0000 to 00FF$_H$) specified directly by x in the object code and places the result in the register rr.			
AND rr,(vw)	1110 0001 : wwww wwww : vvvv vvvv : 10rr r100	Z Z - - - -	7	rr ← rr∧(vw)
	Logically ANDs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address (0000 to FFFF$_H$) specified directly by vw in the object code and places the result in the register rr.			
AND rr,(DE)	1110 0010 : 10rr r100	Z Z - - - -	5	rr ← rr∧(DE)
	Logically ANDs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address specified by the DE register pair and places the result in the register rr.			
AND rr,(HL)	1110 0011 : 10rr r100	Z Z - - - -	5	rr ← rr∧(HL)
	Logically ANDs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address specified by the HL register pair and places the result in the register rr.			
AND rr,(IX)	1110 0100 : 10rr r100	Z Z - - - -	5	rr ← rr∧(IX)
	Logically ANDs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address specified by the index register IX and places the result in the register rr.			
AND rr,(IY)	1110 0101 : 10rr r100	Z Z - - - -	5	rr ← rr∧(IY)
	Logically ANDs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address specified by the index register IY and places the result in the register rr.			
AND rr,(IX+d)	1101 0100 : dddd dddd : 10rr r100	Z Z - - - -	7	rr ← rr∧(IX+d)
	Logically ANDs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended and places the result in the register rr.			
AND rr,(IY+d)	1101 0101 : dddd dddd : 10rr r100	Z Z - - - -	7	rr ← rr∧(IY+d)
	Logically ANDs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended and places the result in the register rr.			
AND rr,(SP+d)	1101 0110 : dddd dddd : 10rr r100	Z Z - - - -	7	rr ← rr∧(SP+d)
	Logically ANDs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended and places the result in the register rr.			
AND rr,(HL+d)	1101 0111 : dddd dddd : 10rr r100	Z Z - - - -	7	rr ← rr∧(HL+d)
	Logically ANDs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended and places the result in the register rr.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
AND rr,(HL+C)	1110 0111 : 10rr r100 :	Z Z - - - -	7	rr ← rr∧(HL+C)
	Logically ANDs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended and places the result in the register rr.			
AND rr,(+SP)	1110 0110 : 10rr r100 :	Z Z - - - -	6	SP ← SP+1 : rr ← rr∧(SP)
	Logically ANDs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address specified by the content of the stack pointer after being incremented and places the result in the register rr.			
AND rr,(PC+A)	0100 1111 : 10rr r100 :	Z Z - - - -	7	rr ← rr∧(PC+A)
	Logically ANDs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended and places the result in the register rr.			
OR A,n	0110 0110 : nnnn nnnn :	Z Z - - - -	2	A ← A∨n
	Logically ORs the accumulator content bitwise (or bit-by-bit) with the object code's immediate data n and places the result in the accumulator.			
OR g,n	1110 1ggg : 0110 0110 : nnnn nnnn	Z Z - - - -	3	g ← g∨n
	Logically ORs the content of 8-bit register g bitwise with the object code's immediate data n and places the result in the register g.			
OR gg,mn	1110 1ggg : 0110 1110 : nnnn nnnn : mmmm mmmm	Z Z - - - -	4	gg ← gg∨mn
	Logically ORs the content of 16-bit register gg bitwise with the object code's immediate data mn and places the result in the register gg.			
OR r,g	1110 1ggg : 00rr r110 :	Z Z - - - -	2	r ← r∨g
	Logically ORs the content of 8-bit register r bitwise with that of 8-bit register g and places the result in the register r.			
OR rr,gg	1110 1ggg : 10rr r110 :	Z Z - - - -	4	rr ← rr∨gg
	Logically ORs the content of 16-bit register rr bitwise with that of 16-bit register gg and places the result in the register rr.			
OR r,(x)	1110 0000 : xxxx xxxx : 00rr r110	Z Z - - - -	4	r ← r∨(x)
	Logically ORs the content of 8-bit register r bitwise with the memory content at the address (0000 to $00FF_H$) specified directly by x in the object code and places the result in the register r.			
OR r,(vw)	1110 0001 : wwww wwww : vvvv vvvv : 00rr r110	Z Z - - - -	5	r ← r∨(vw)
	Logically ORs the content of 8-bit register r bitwise with the memory content at the address (0000 to $FFFF_H$) specified directly by vw in the object code and places the result in the register r.			
OR r,(DE)	1110 0010 : 00rr r110 :	Z Z - - - -	3	r ← r∨(DE)
	Logically ORs the content of 8-bit register r bitwise with the memory content at the address specified by the DE register pair and places the result in the register r.			
OR r,(HL)	1110 0011 : 00rr r110 :	Z Z - - - -	3	r ← r∨(HL)
	Logically ORs the content of 8-bit register r bitwise with the memory content at the address specified by the HL register pair and places the result in the register r.			
OR r,(IX)	1110 0100 : 00rr r110 :	Z Z - - - -	3	r ← r∨(IX)
	Logically ORs the content of 8-bit register r bitwise with the memory content at the address specified by the index register IX and places the result in the register r.			
OR r,(IY)	1110 0101 : 00rr r110 :	Z Z - - - -	3	r ← r∨(IY)
	Logically ORs the content of 8-bit register r bitwise with the memory content at the address specified by the index register IY and places the result in the register r.			
OR r,(IX+d)	1101 0100 : dddd dddd : 00rr r110	Z Z - - - -	5	r ← r∨(IX+d)
	Logically ORs the content of 8-bit register r bitwise with the memory content at the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended and places the result in the register r.			
OR r,(IY+d)	1101 0101 : dddd dddd : 00rr r110	Z Z - - - -	5	r ← r∨(IY+d)
	Logically ORs the content of 8-bit register r bitwise with the memory content at the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended and places the result in the register r.			
OR r,(SP+d)	1101 0110 : dddd dddd : 00rr r110	Z Z	6	r ← r∨(SP+d)
	Logically ORs the content of 8-bit register r bitwise with the memory content at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended and places the result in the register r.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
OR r,(HL+d)	1101 0111 : dddd dddd : 00rr r110	Z Z - - - -	5	r ← rV(HL+d)
	Logically ORs the content of 8-bit register r bitwise with the memory content at the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended and places the result in the register r.			
OR r,(HL+C)	1110 0111 : 00rr r110	Z Z - - - -	6	r ← rV(HL+C)
	Logically ORs the content of 8-bit register r bitwise with the memory content at the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended and places the result in the register r.			
OR r,(+SP)	1110 0110 : 00rr r110 :	Z Z - - - -	4	SP ← SP+1:r ← rV(SP)
	Logically ORs the content of 8-bit register r bitwise with the memory content at the address specified by the content of the stack pointer after being incremented and places the result in the register r.			
OR r,(PC+A)	0100 1111 : 00rr r110 :	Z Z - - - -	5	r ← rV(PC+A)
	Logically ORs the content of 8-bit register r bitwise with the memory content at the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended and places the result in the register r.			
OR (x),n	1110 0000 : xxxx xxxx : 0110 0110 nnnn nnnn	Z Z - - - -	6	(x) ← (x)∨n
	Logically ORs the memory content at the address (0000 to 00FF$_H$) specified directly by x in the object code bitwise with the object code's immediate data n and places the result in the said address.			
OR (vw),n	1110 0001 : wwww wwww : vvvv vvvv 0110 0110 : nnnn nnnn	Z Z - - - -	7	(vw) ← (vw)∨n
	Logically ORs the memory content at the address (0000 to FFFF$_H$) specified directly by vw in the object code bitwise with the object code's immediate data n and places the result in the said address.			
OR (DE),n	1110 0010 : 0110 0110 : nnnn nnnn	Z Z - - - -	5	(DE) ← (DE)∨n
	Logically ORs the memory content at the address specified by the DE register pair bitwise with the object code's immediate data n and places the result in the said address.			
OR (HL),n	1110 0011 : 0110 0110 : nnnn nnnn	Z Z - - - -	5	(HL) ← (HL)∨n
	Logically ORs the memory content at the address specified by the HL register pair bitwise with the object code's immediate data n and places the result in the said address.			
OR (IX),n	1110 0100 : 0110 0110 : nnnn nnnn	Z Z - - - -	5	(IX) ← (IX)∨n
	Logically ORs the memory content at the address specified by the index register IX bitwise with the object code's immediate data n and places the result in the said address.			
OR (IY),n	1110 0101 : 0110 0110 : nnnn nnnn	Z Z - - - -	5	(IY) ← (IY)∨n
	Logically ORs the memory content at the address specified by the index register IY bitwise with the object code's immediate data n and places the result in the said address.			
OR (IX+d),n	1101 0100 : dddd dddd : 0110 0110 nnnn nnnn	Z Z - - - -	7	(IX+d) ← (IX+d)∨n
	Logically ORs the memory content at the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended bitwise with the object code's immediate data n and places the result in the said address.			
OR (IY+d),n	1101 0101 : dddd dddd : 0110 0110 nnnn nnnn	Z Z - - - -	7	(IY+d) ← (IY+d)∨n
	Logically ORs the memory content at the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended bitwise with the object code's immediate data n and places the result in the said address.			
OR (SP+d),n	1101 0110 : dddd dddd : 0110 0110 nnnn nnnn	Z Z - - - -	7	(SP+d) ← (SP+d)∨n
	Logically ORs the memory content at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended bitwise with the object code's immediate data n and places the result in the said address.			
OR (HL+d),n	1101 0111 : dddd dddd : 0110 0110 nnnn nnnn	Z Z - - - -	7	(HL+d) ← (HL+d)∨n
	Logically ORs the memory content at the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended bitwise with the object code's immediate data n and places the result in the said address.			
OR (HL+C),n	1110 0111 : 0110 0110 : nnnn nnnn	Z Z - - - -	7	(HL+C) ← (HL+C)∨n
	Logically ORs the memory content at the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended bitwise with the object code's immediate data n and places the result in the said address.			
OR (SP),n	1110 0110 : 0110 0110 : nnnn nnnn	Z Z - - - -	6	SP ← SP+1:(SP) ← (SP)∨n
	Logically ORs the memory content at the address specified by the content of the stack pointer after being incremented bitwise with the object code's immediate data n and places the result in the said address.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
OR (PC+A),n	0100 1111 : 0110 0110 : nnnn nnnn	Z Z - - - -	7	(PC+A) ← (PC+A) ∨ n
	Logically ORs the memory content at the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended bitwise with the object code's immediate data n and places the result in the said address.			
OR rr,(x)	1110 0000 : xxxx xxxx : 10rr r110	Z Z - - - -	9	rr ← rr ∨ (x)
	Logically ORs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address (0000 to $00FF_H$) specified directly by x in the object code and places the result in the register rr.			
OR rr,(vw)	1110 0001 : wwww wwww : vvvv vvvv : 10rr r110	Z Z - - - -	7	rr ← rr ∨ (vw)
	Logically ORs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address (0000 to $FFFF_H$) specified directly by vw in the object code and places the result in the register rr.			
OR rr,(DE)	1110 0010 : 10rr r110	Z Z - - - -	5	rr ← rr ∨ (DE)
	Logically ORs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address specified by the DE register pair and places the result in the register rr.			
OR rr,(HL)	1110 0011 : 10rr r110	Z Z - - - -	5	rr ← rr ∨ (HL)
	Logically ORs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address specified by the HL register pair and places the result in the register rr.			
OR rr,(IX)	1110 0100 : 10rr r110	Z Z - - - -	5	rr ← rr ∨ (IX)
	Logically ORs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address specified by the index register IX and places the result in the register rr.			
OR rr,(IY)	1110 0101 : 10rr r110	Z Z - - - -	5	rr ← rr ∨ (IY)
	Logically ORs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address specified by the index register IY and places the result in the register rr.			
OR rr,(IX+d)	1101 0100 : dddd dddd : 10rr r110	Z Z - - - -	7	rr ← rr ∨ (IX+d)
	Logically ORs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended and places the result in the register rr.			
OR rr,(IY+d)	1101 0101 : dddd dddd : 10rr r110	Z Z - - - -	7	rr ← rr ∨ (IY+d)
	Logically ORs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended and places the result in the register rr.			
OR rr,(SP+d)	1101 0110 : dddd dddd : 10rr r110	Z Z - - - -	7	rr ← rr ∨ (SP+d)
	Logically ORs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended and places the result in the register rr.			
OR rr,(HL+d)	1101 0111 : dddd dddd : 10rr r110	Z Z - - - -	7	rr ← rr ∨ (HL+d)
	Logically ORs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended and places the result in the register rr.			
OR rr,(HL+C)	1110 0111 : 10rr r110	Z Z - - - -	7	rr ← rr ∨ (HL+C)
	Logically ORs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended and places the result in the register rr.			
OR rr,(+SP)	1110 0110 : 10rr r110	Z Z - - - -	6	SP ← SP+1 : rr ← rr ∨ (SP)
	Logically ORs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address specified by the content of the stack pointer after being incremented and places the result in the register rr.			
OR rr,(PC+A)	0100 1111 : 10rr r110	Z Z - - - -	7	rr ← rr ∨ (PC+A)
	Logically ORs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended and places the result in the register rr.			
XOR A,n	0110 0101 : nnnn nnnn	Z Z - - - -	2	A ← A ⊻ n
	Logically XORs the accumulator content bitwise (or bit-by-bit) with the object code's immediate data n and places the result in the accumulator. If n = FFH, this instruction takes the 1's complement of the data (inverts the data). Example: If the instruction XOR A,OFF_H is executed when A = 69_H, then A = 96_H and ZF = 0.			
XOR g,n	1110 1ggg : 0110 0101 : nnnn nnnn	Z Z - - - -	3	g ← g ⊻ nV
	Logically XORs the content of 8-bit register g bitwise with the object code's immediate data n and places the result in the register g. If n = FF_H, this instruction takes the 1's complement of the data (inverts the data).			
XOR gg,mn	1110 1ggg : 0110 1101 : nnnn nnnn : mmmm mmmm	Z Z - - - -	4	gg ← gg ⊻ mn
	Logically XORs the content of 16-bit register gg bitwise with the object code's immediate data mn and places the result in the register gg. If mn = $FFFF_H$, this instruction takes the 1's complement of the data (inverts the data).			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
XOR r,g	1110 1ggg : 00rr r101	Z Z - - - -	2	r ← r⊻g
	Logically XORs the content of 8-bit register r bitwise with that of 8-bit register g and places the result in the register r.			
XOR rr,gg	1110 1ggg : 10rr r101	Z Z - - - -	4	rr ← rr⊻gg
	Logically XORs the content of 16-bit register rr bitwise with that of 16-bit register gg and places the result in the register rr.			
XOR r,(x)	1110 0000 : xxxx xxxx : 00rr r101	Z Z - - - -	4	r ← r⊻(x)
	Logically XORs the content of 8-bit register r bitwise with the memory content at the address (0000 to $00FF_H$) specified directly by x in the object code and places the result in the register r.			
XOR r,(vw)	1110 0001 : wwww wwww : vvvv vvvv : 00rr r101	Z Z - - - -	5	r ← r⊻(vw)
	Logically XORs the content of 8-bit register r bitwise with the memory content at the address (0000 to $FFFF_H$) specified directly by vw in the object code and places the result in the register r.			
XOR r,(DE)	1110 0010 : 00rr r101	Z Z - - - -	3	r ← r⊻(DE)
	Logically XORs the content of 8-bit register r bitwise with the memory content at the address specified by the DE register pair and places the result in the register r.			
XOR r,(HL)	1110 0011 : 00rr r101	Z Z - - - -	3	r ← r⊻(HL)
	Logically XORs the content of 8-bit register r bitwise with the memory content at the address specified by the HL register pair and places the result in the register r.			
XOR r,(IX)	1110 0100 : 00rr r101	Z Z - - - -	3	r ← r⊻(IX)
	Logically XORs the content of 8-bit register r bitwise with the memory content at the address specified by the index register IX and places the result in the register r.			
XOR r,(IY)	1110 0101 : 00rr r101	Z Z - - - -	3	r ← r⊻(IY)
	Logically XORs the content of 8-bit register r bitwise with the memory content at the address specified by *the index register IY and places the result in the register r.*			
XOR r,(IX+d)	1101 0100 : dddd dddd : 00rr r101	Z Z - - - -	5	r ← r⊻(IX+d)
	Logically XORs the content of 8-bit register r bitwise with the memory content at the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended and places the result in the register r.			
XOR r,(IY+d)	1101 0101 : dddd dddd : 00rr r101	Z Z - - - -	5	r ← r⊻(IY+d)
	Logically XORs the content of 8-bit register r bitwise with the memory content at the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended and places the result in the register r.			
XOR r,(SP+d)	1101 0110 : dddd dddd : 00rr r101	Z Z - - - -	5	r ← r⊻(SP+d)
	Logically XORs the content of 8-bit register r bitwise with the memory content at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended and *places the result in the register r.*			
XOR r,(HL+d)	1101 0111 : dddd dddd : 00rr r101	Z Z - - - -	5	r ← r⊻(HL+d)
	Logically XORs the content of 8-bit register r bitwise with the memory content at the address specified by *the content of the HL register pair plus the object code's 8-bit data d that* is added after being sign-extended and places the result in the register r.			
XOR r,(HL+C)	1110 0111 : 00rr r101	Z Z - - - -	5	r ← r⊻(HL+C)
	Logically XORs the content of 8-bit register r bitwise with the memory content at the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended and places the result in the register r.			
XOR r,(+SP)	1110 0110 : 00rr r101	Z Z - - - -	4	SP ← SP+1:r ← r⊻(SP)
	Logically XORs the content of 8-bit register r bitwise with the memory content at the address specified by the content of the stack pointer after being incremented and places the result in the register r.			
XOR r,(PC+A)	0100 1111 : 00rr r101	Z Z - - - -	5	r ← r⊻(PC+A)
	Logically XORs the content of 8-bit register r bitwise with the memory content at the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended and *places the result in the register r.*			
XOR (x),n	1110 0000 : xxxx xxxx : 0110 0101 : nnnn nnnn	Z Z - - - -	6	(x) ← (x)⊻n
	Logically XORs the memory content at the address (0000 to $00FF_H$) specified directly by x in the object code bitwise with the object code's immediate data n and places the result in the said address.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
XOR (vw),n	1110 0001 : wwww wwww : vvvv vvvv 0110 0101 : nnnn nnnn	Z Z - - - -	7	(vw) ← (vw) ⊻ n
	Logically XORs the memory content at the address (0000 to FFFF$_H$) specified directly by vw in the object code bitwise with the object code's immediate data n and places the result in the said address.			
XOR (DE),n	1110 0010 : 0110 0101 : nnnn nnnn	Z Z - - - -	5	(DE) ← (DE) ⊻ n
	Logically XORs the memory content at the address specified by the DE register pair bitwise with the object code's immediate data n and places the result in the said address.			
XOR (HL),n	1110 0011 : 0110 0101 : nnnn nnnn	Z Z - - - -	5	(HL) ← (HL) ⊻ n
	Logically XORs the memory content at the address specified by the HL register pair bitwise with the object code's immediate data n and places the result in the said address.			
XOR (IX),n	1110 0100 : 0110 0101 : nnnn nnnn	Z Z - - - -	5	(IX) ← (IX) ⊻ n
	Logically XORs the memory content at the address specified by the index register IX bitwise with the object code's immediate data n and places the result in the said address.			
XOR (IY),n	1110 0101 : 0110 0101 : nnnn nnnn	Z Z - - - -	5	(IY) ← (IY) ⊻ n
	Logically XORs the memory content at the address specified by the index register IY bitwise with the object code's immediate data n and places the result in the said address.			
XOR (IX+d),n	1101 0100 : dddd dddd : 0110 0101 nnnn nnnn	Z Z - - - -	7	(IX+d) ← (IX+d) ⊻ n
	Logically XORs the memory content at the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended bitwise with the object code's immediate data n and places the result in the said address.			
XOR (IY+d),n	1101 0101 : dddd dddd : 0110 0101 nnnn nnnn	Z Z - - - -	7	(IY+d) ← (IY+d) ⊻ n
	Logically XORs the memory content at the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended bitwise with the object code's immediate data n and places the result in the said address.			
XOR (SP+d),n	1101 0110 : dddd dddd : 0110 0101 nnnn nnnn	Z Z - - - -	7	(SP+d) ← (SP+d) ⊻ n
	Logically XORs the memory content at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended bitwise with the object code's immediate data n and places the result in the said address.			
XOR (HL+d),n	1101 0111 : dddd dddd : 0110 0101 nnnn nnnn	Z Z - - - -	7	(HL+d) ← (HL+d) ⊻ n
	Logically XORs the memory content at the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended bitwise with the object code's immediate data n and places the result in the said address.			
XOR (HL+C),n	1110 0111 : 0110 0101 : nnnn nnnn	Z Z - - - -	7	(HL+C) ← (HL+C) ⊻ n
	Logically XORs the memory content at the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended bitwise with the object code's immediate data n and places the result in the said address.			
XOR (+SP),n	1110 0110 : 0110 0101 : nnnn nnnn	Z Z - - - -	6	SP ← SP+1:(SP) ← (SP) ⊻ n
	Logically XORs the memory content at the address specified by the content of the stack pointer after being incremented bitwise with the object code's immediate data n and places the result in the said address.			
XOR (PC+A),n	0100 1111 : 0110 0101 : nnnn nnnn	Z Z - - - -	7	(PC+A) ← (PC+A) ⊻ n
	Logically XORs the memory content at the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended bitwise with the object code's immediate data n and places the result in the said address.			
XOR rr,(x)	1110 0000 : xxxx xxxx : 10rr r101	Z Z - - - -	6	rr ← rr ⊻ (x)
	Logically XORs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address (0000 to 00FF$_H$) specified directly by x in the object code and places the result in the register rr.			
XOR rr,(vw)	1110 0001 : wwww wwww : vvvv vvvv 10rr r101	Z Z - - - -	7	rr ← rr ⊻ (vw)
	Logically XORs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address (0000 to FFFF$_H$) specified directly by vw in the object code and places the result in the register rr.			
XOR rr,(DE)	1110 0010 : 10rr r101	Z Z - - - -	5	rr ← rr ⊻ (DE)
	Logically XORs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address specified by the DE register pair and places the result in the register rr.			
XOR rr,(HL)	1110 0011 : 10rr r101	Z Z - - - -	5	rr ← rr ⊻ (HL)
	Logically XORs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address specified by the HL register pair and places the result in the register rr.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
XOR rr,(IX)	1110 0100 : 10rr r101 :	Z Z - - - -	5	rr ← rr⊻(IX)
	Logically XORs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address specified by the index register IX and places the result in the register rr.			
XOR rr,(IY)	1110 0101 : 10rr r101 :	Z Z - - - -	6	rr ← rr⊻(IY)
	Logically XORs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address specified by the index register IY and places the result in the register rr.			
XOR rr,(IX+d)	1101 0100 : dddd dddd : 10rr r101	Z Z - - - -	7	rr ← rr⊻(IX+d)
	Logically XORs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended and places the result in the register rr.			
XOR rr,(IY+d)	1101 0101 : dddd dddd : 10rr r101	Z Z - - - -	7	rr ← rr⊻(IY+d)
	Logically XORs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended and places the result in the register rr.			
XOR rr,(SP+d)	1101 0110 : dddd dddd : 10rr r101	Z Z - - - -	7	rr ← rr⊻(SP+d)
	Logically XORs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address specified by the content of the stack pointer plus the object code's 8 bit data d that is added after being sign-extended and places the result in the register rr.			
XOR rr,(HL+d)	1101 0111 : dddd dddd : 10rr r101	Z Z - - - -	7	rr ← rr⊻(HL+d)
	Logically XORs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended and places the result in the register rr.			
XOR rr,(HL+C)	1110 0111 : 10rr r101 :	Z Z - - - -	7	rr ← rr⊻(HL+C)
	Logically XORs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended and places the result in the register rr.			
XOR rr,(+SP)	1110 0110 : 10rr r101 :	Z Z - - - -	6	SP ← SP+1:rr ← rr⊻(SP)
	Logically XORs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address specified by the content of the stack pointer after being incremented and places the result in the register rr.			
XOR rr,(PC+A)	0100 1111 : 10rr r101 :	Z Z - - - -	7	rr ← rr⊻(PC+A)
	Logically XORs the content of 16-bit register rr bitwise with consecutive 2-byte memory contents beginning with the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended and places the result in the register rr.			
INC r	0010 0rrr :	C Z - - - -	1	r ← r+1
	Increments the content of 8-bit register r. If overflow occurs, the jump status and zero flags both are set to 1. The carry flag does not change state. Example: If the instruction INC L is executed when $L=FF_H$, then $L=00_H$, ZF = 1, and JF = 1.			
INC rr	0011 0rrr :	C Z - - - -	2	rr ← rr+1
	Increments the content of 16-bit register rr. If overflow occurs, the jump status and zero flags both are set to 1. Example: If the instruction INC HL is executed when $HL=1234_H$, then $HL=1235_H$, ZF = 0, and JF = 0.			
INC (x)	1110 0000 : xxxx xxxx : 1111 0000	C Z - - - -	5	(x) ← (x)+1
	Increments the memory content at the address (0000 to $00FF_H$) specified directly by x in the object code. If overflow occurs, the jump status and zero flags both are set to 1.			
INC (vw)	1110 0001 : wwww wwww : vvvv vvvv 1111 0000 :	C Z - - - -	6	(vw) ← (vw)+1
	Increments the memory content at the address (0000 to $FFFF_H$) specified directly by vw in the object code. If overflow occurs, the jump status and zero flags both are set to 1.			
INC (DE)	1110 0010 : 1111 0000 :	C Z - - - -	4	(DE) ← (DE)+1
	Increments the memory content at the address specified by the DE register pair. If overflow occurs, the jump status and zero flags both are set to 1.			
INC (HL)	1110 0011 : 1111 0000 :	C Z - - - -	4	(HL) ← (HL)+1
	Increments the memory content at the address specified by the HL register pair. If overflow occurs, the jump status and zero flags both are set to 1.			
INC (IX)	1110 0100 : 1111 0000 :	C Z - - - -	4	(IX) ← (IX)+1
	Increments the memory content at the address specified by the index register IX. If overflow occurs, the jump status and zero flags both are set to 1.			
INC (IY)	1110 0101 : 1111 0000 :	C Z	4	(IY) ← (IY)+1
	Increments the memory content at the address specified by the index register IY. If overflow occurs, the jump status and zero flags both are set to 1.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
INC (IX+d)	1101 0100 : dddd dddd : 1111 0000	C Z - - - -	6	(IX+d) ← (IX+d)+1
	Increments the memory content at the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended. If overflow occurs, the jump status and zero flags both are set to 1.			
INC (IY+d)	1101 0101 : dddd dddd : 1111 0000	C Z - - - -	6	(IY+d) ← (IY+d)+1
	Increments the memory content at the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended. If overflow occurs, the jump status and zero flags both are set to 1.			
INC (SP+d)	1101 0110 : dddd dddd : 1111 0000	C Z - - - -	6	(SP+d) ← (SP+d)+1
	Increments the memory content at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended. If overflow occurs, the jump status and zero flags both are set to 1.			
INC (HL+d)	1101 0111 : dddd dddd : 1111 0000	C Z - - - -	6	(HL+d) ← (HL+d)+1
	Increments the memory content at the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended. If overflow occurs, the jump status and zero flags both are set to 1.			
INC (HL+C)	1110 0111 : 1111 0000	C Z - - - -	6	(HL+C) ← (HL+C)+1
	Increments the memory content at the address specified by the content of the HL register pair plus the C register content that is added after being sign extended. If overflow occurs, the jump status and zero flags both are set to 1.			
INC (+SP)	1110 0110 : 1111 0000	C Z - - - -	5	SP ← SP+1:(SP) ← (SP)+1
	Increments the memory content at the address specified by the content of the stack pointer after being incremented. If the memory content overflows, the jump status and zero flags both are set to 1.			
INC (PC+A)	0100 1111 : 1111 0000	C Z - - - -	6	(PC+A) ← (PC+A)+1
	Increments the memory content at the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended. If overflow occurs, the jump status and zero flags both are set to 1.			
DEC r	0010 1rrr	C Z - - - -	1	r ← r-1
	Decrements the content of 8-bit register r. If underflow occurs (result = FF_H), the jump status flag is set to 1. The carry flag does not change state. Example: If the instruction DEC L is executed when L = 00_H, then L = FF_H, ZF = 0, and JF = 1.			
DEC rr	0011 1rrr	C Z	2	rr ← rr-1
	Decrements the content of 16-bit register rr. If underflow occurs (result = FFH), the jump status flag is set to 1. Example: If the instruction DEC HL is executed when HL = 8765_H, then HL = 8764_H, ZF = 0, and JF = 0.			
DEC (x)	1110 0000 : xxxx xxxx : 1111 1000	C Z - - - -	5	(x) ← (x)-1
	Decrements the memory content at the address (0000 to $00FF_H$) specified directly by x in the object code. If underflow occurs, the jump status flag is set to 1.			
DEC (vw)	1110 0001 : wwww wwww : vvvv vvvv : 1111 1000	C Z - - - -	6	(vw) ← (vw)-1
	Decrements the memory content at the address (0000 to $FFFF_H$) specified directly by vw in the object code. If underflow occurs, the jump status flag is set to 1.			
DEC (DE)	1110 0010 : 1111 1000	C Z - - - -	4	(DE) ← (DE)-1
	Decrements the memory content at the address specified by the DE register pair. If underflow occurs, the jump status flag is set to 1.			
DEC (HL)	1110 0011 : 1111 1000	C Z - - - -	4	(HL) ← (HL)-1
	Decrements the memory content at the address specified by the HL register pair. If underflow occurs, the jump status flag is set to 1.			
DEC (IX)	1110 0100 : 1111 1000	C Z - - - -	4	(IX) ← (IX)-1
	Decrements the memory content at the address specified by the index register IX. If underflow occurs, the jump status flag is set to 1.			
DEC (IY)	1110 0101 : 1111 1000	C Z - - - -	4	(IY) ← (IY)-1
	Decrements the memory content at the address specified by the index register IY. If underflow occurs, the jump status flag is set to 1.			
DEC (IX+d)	1101 0100 : dddd dddd : 1111 1000	C Z - - - -	6	(IX+d) ← (IX+d)-1
	Decrements the memory content at the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended. If underflow occurs, the jump status flag is set to 1.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
DEC (IY+d)	1101 0101 : dddd dddd : 1111 1000	C Z - - - -	6	(IY+d) ← (IY+d)-1
	Decrements the memory content at the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended. If underflow occurs, the jump status flag is set to 1.			
DEC (SP+d)	1101 0110 : dddd dddd : 1111 1000	C Z - - - -	6	(SP+d) ← (SP+d)-1
	Decrements the memory content at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended. If underflow occurs, the jump status flag is set to 1.			
DEC (HL+d)	1101 0111 : dddd dddd : 1111 1000	C Z - - - -	6	(HL+d) ← (HL+d)-1
	Decrements the memory content at the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended. If underflow occurs, the jump status flag is set to 1.			
DEC (HL+C)	1110 0111 : 1111 1000	C Z - - - -	6	(HL+C) ← (HL+C)-1
	Decrements the memory content at the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended. If underflow occurs, the jump status flag is set to 1.			
DEC (+SP)	1110 0110 : 1111 1000	C Z - - - -	5	SP ← SP+1:(SP) ← (SP)-1
	Decrements the memory content at the address specified by the content of the stack pointer after being decremented. If underflow occurs, the jump status flag is set to 1.			
DEC (PC+A)	0100 1111 : 1111 1000	C Z - - - -	6	(PC+A) ← (PC+A)-1
	Decrements the memory content at the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended. If underflow occurs, the jump status flag is set to 1.			
DAA g	1110 1ggg : 1101 1010	C Z C H - -	2	Decimal adjustment against g (after addition)

When adding 8-bit packed BCD data, this instruction decimal adjusts the content of register g after executing the add instruction (ADD or ADDC). For multidigit BCD data, the operation is repeated by adding and adjusting 8 bits at a time beginning with the low-order digit.

Carry flag before adjustment	High-order 4 bits of register g before adjustment	Half carry flag before adjustment	Low-order 4 bits of register g before adjustment	Value added for adjustment	Carry flag after adjustment
0	0~9	0	0~9	00	0
0	0~8	0	A~F	06	0
0	0~9	1	0~3	06	0
0	A~F	0	0~9	60	1
0	9~F	0	A~F	66	1
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

Example: If the instruction ADD A,B is executed when A = 26_H and B = 57_H, then A = $7D_H$, CF = 0, and HF = 0. If the instruction DAA A is executed in this state, the result is A = 83_H and CF = 0

DAS g	1110 1ggg : 1101 1011	C Z C H - -	2	Decimal adjustment against g (after subtraction)

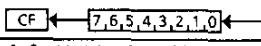
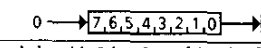
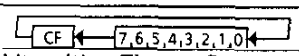
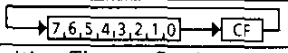
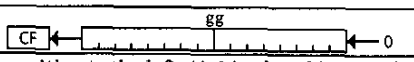
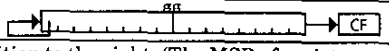
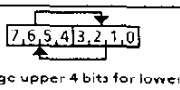
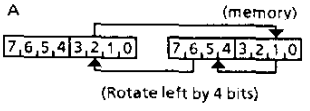
When subtracting 8-bit packed BCD data, this instruction decimal adjusts the content of register g after executing the subtract instruction (SUB or SUBB). For multidigit BCD data, the operation is repeated by subtracting and adjusting 8 bits at a time beginning with the low-order digit.

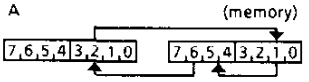
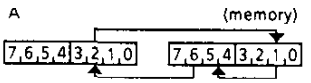
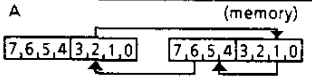
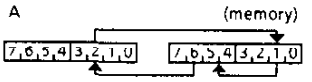
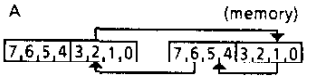
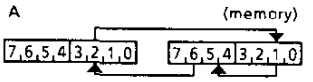
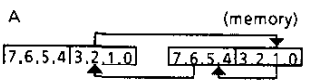
Carry flag before adjustment	High-order 4 bits of register g before adjustment	Half carry flag before adjustment	Low-order 4 bits of register g before adjustment	Value added for adjustment	Carry flag after adjustment
0	0~9	0	0~9	00	0
0	0~8	1	6~F	FA	0
1	7~F	0	0~9	A0	1
1	6~F	1	6~F	9A	1

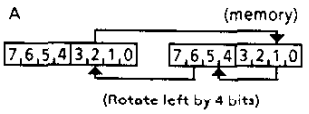
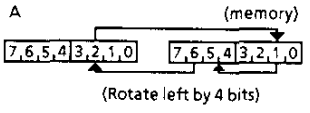
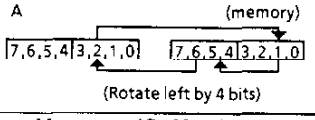
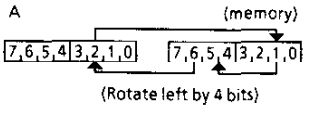
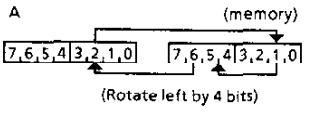
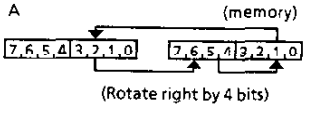
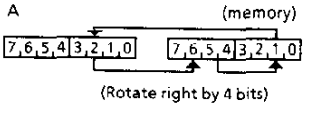
Example: If the instruction SUB A,B is executed when A = 87_H and B = 39_H, then A = $4E_H$, CF = 0, and HF = 1. If the instruction DAS A is executed in this state, the result is A = 48_H and CF = 0.

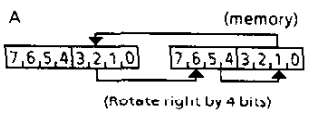
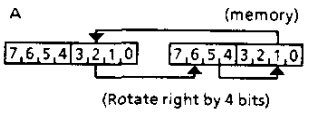
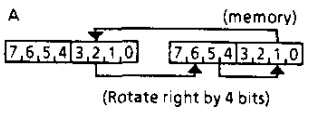
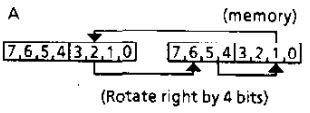
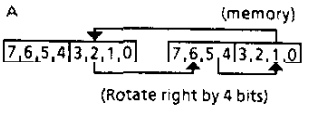
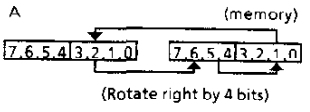
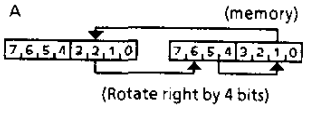
MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
MUL W,A	1110 1000 : 1111 0010	Z Z - - - -	8	WA ← W×A
	Multiplies the W register content (unsigned integer) by the accumulator content (unsigned integer) and places the result in the 16-bit accumulator. The zero flag is set to 1 when the 8 high-order bits of the result (the value stored in the W register) = 00_H; otherwise, the flag is cleared to 0. Example 1: If this instruction is executed when W = 87_H and A = $F2_H$, then WA = $7F9E_H$ and ZF = 0. Example 2: If this instruction is executed when W = 16_H and A = 05_H, then WA = $006E_H$ and ZF = 1.			
MUL B,C	1110 1001 : 1111 0010	Z Z - - - -	8	BC ← B×C
	Multiplies the B register content (unsigned integer) by the C register content (unsigned integer) and places the result in the BC register pair. The zero flag is set to 1 when the 8 high-order bits of the result (the value stored in the B register) = 00_H; otherwise, the zero flag is set to 1.			
MUL D,E	1110 1010 : 1111 0010	Z Z - - - -	8	DE ← D×E
	Multiplies the D register content (unsigned integer) by the E register content (unsigned integer) and places the result in the DE register pair. The zero flag is set to 1 when the 8 high-order bits of the result (the value stored in the D register) = 00_H; otherwise, the zero flag is set to 1.			
MUL H,L	1110 1011 : 1111 0010	Z Z - - - -	8	HL ← H×L
	Multiplies the H register content (unsigned integer) by the L register content (unsigned integer) and places the result in the HL register pair. The zero flag is set to 1 when the 8 high-order bits of the result (the value stored in the D register) = 00_H; otherwise, the zero flag is set to 1.			
DIV WA,C	1110 1000 : 1111 0011	? ? C - - -	8	A ← WA÷C, W ← residue
	Divides the 16-bit accumulator content (unsigned integer) by the C register content (unsigned integer) and places the quotient in the accumulator and the remainder in the W register. The carry flag is set to 1 when the divisor (C register content) = 00_H or the quotient = 100_H or greater. (In this case, the accumulator and W register contents are indeterminate.) Otherwise, the flag is cleared to 0. Also, the zero flag is set to 1 when the remainder = 00_H, otherwise, the flag is cleared to 0. The C register content does not change. Example1: If this instruction is executed when WA = 1234_H and C = 56_H, then A = 36_H, W = 10_H, CF = 0, and ZF = 0. Example 2: If this instruction is executed when WA = 4830_H and C = $9A_H$, then A = 78_H, W = 00_H, CF = 0, and ZF = 1. Example 3: If this instruction is executed when WA = 3210_H and C = 27_H, then A = 48_H, W = 18_H, CF = 1, and ZF = 0.			
DIV DE,C	1110 1010 : 1111 0011	Z Z C - - -	8	E ← DE÷C, D ← residue
	Divides the content of the DE register pair (unsigned integer) by the C register content (unsigned integer) and places the 8 low-order bits of the quotient in the E register and the remainder in the D register. The carry flag is set to 1 when the divisor (C register content) = 00_H or the quotient = 100_H or greater. (In this case, the content of the DE register pair is indeterminate.) Also, the zero flag is set to 1 when the remainder = 00_H. The C register content does not change.			
DIV HL,C	1110 1011 : 1111 0011	Z Z C - - -	8	L ← HL÷C, H ← residue
	Divides the content of the HL register pair (unsigned integer) by the C register content (unsigned integer) and places the 8 low-order bits of the quotient in the L register and the remainder in the H register. The carry flag is set to 1 when the divisor (C register content) = 00_H or the quotient = 100_H or greater. (In this case, the content of the HL register pair is indeterminate.) Also, the zero flag is set to 1 when the remainder = 00_H. The C register content does not change.			
NEG CS,gg	1110 1ggg : 1111 1010	1 - - - - -	3	if CF=1 then gg ← 0-gg else null
	When the carry flag = 1, this instruction takes the 2's complement of the content of 16-bit register gg, stores the result in the register gg, and sets the jump status flag to 1. When the carry flag = 0, this instruction sets the jump status flag to 1 and goes on to execute the next instruction. (In this case, the content of register gg does not change.) Example 1: If the instruction NEG CS,HL is executed when HL = 5678_H and CF = 1_H, then HL = $A988_H$ and CF = 1. Example 2 If the instruction NEG CS,DE is executed when DE = $89AB_H$ and CF = 0_H, then DE = $89AB_H$ and CF = 0.			

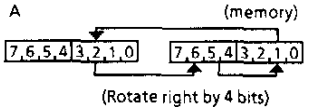
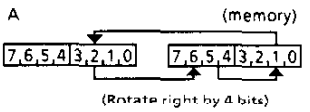
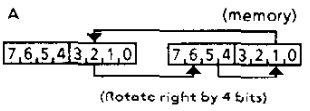
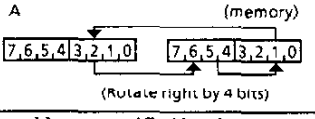
(3) Shift, rotate, and nibble manipulation

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
SHLC g	1110 1ggg : 1111 0100	C Z * - - -	2	
Logically shifts the content of 8-bit register g one bit position to the left. (A 0 is placed in the LSB of register g. The MSB content of register g is placed in the carry flag.) The zero flag is set to 1 when the content of register g after being shifted = 00_H. Example: If this instruction is executed when A = 00111011_B and CF = 1, then A = 01110110_B, CF = 0, and ZF = 0.				
SHRC g	1110 1ggg : 1111 0101	C Z * - - -	2	
Logically shifts the content of 8-bit register g one bit position to the right. (A 0 is placed in the MSB of register g. The LSB content of register g is placed in the carry flag.) The zero flag is set to 1 when the content of register g after being shifted = 00_H. Example: If this instruction is executed when A = 01011101_B and CF = 0, then A = 00101110_B, CF = 1, and ZF = 0.				
ROLC g	1110 1ggg : 1111 0110	C Z * - - -	2	
Rotates the contents of 8-bit register g and carry flag left by one bit position. The zero flag is set to 1 when the content of register g after being rotated = 00_H. Example 1: If this instruction is executed when A = 10010110_B and CF = 0, then A = 00101100_B, CF = 1, JF = 1, and ZF = 0. Example 2: If this instruction is executed when A = 10000000_B and CF = 0, then A = 00000000_B, CF = 1, JF = 1, and ZF = 1.				
RORC g	1110 1ggg : 1111 0111	C Z * - - -	2	
Rotates the contents of 8-bit register g and carry flag right by one bit position. The zero flag is set to 1 when the content of register g after being rotated = 00_H. Example 1: If this instruction is executed when A = 01101101_B and CF = 1, then A = 10110110_B, CF = 1, and ZF = 0.				
SHLCA gg	1110 1ggg : 1111 0000	C Z * - S V	3	
Arithmetically shifts the content of 16 bit register gg one bit position to the left. (A 0 is placed in the LSB of register gg. The MSB content of register gg is placed in the carry flag.) The zero flag is set to 1 when the content of register gg after being shifted = 0000_H. The overflow flag is set to 1 when the MSB content of register gg changes state as a result of shifting. Example: If the instruction SHLCA HL is executed when HL = 3456_H and CF = 1, then HL = $68AC_H$.				
SHRCA gg	1110 1ggg : 1111 0001	C Z * - S 0	3	
Arithmetically shifts the content of 16-bit register gg one bit position to the right. (The MSB of register gg does not change state. The LSB content of register gg is placed in the carry flag.) The zero flag is set to 1 when the content of register gg after being shifted = 0000_H. Example: If the instruction SHRCA DE is executed when DE = $89AB_H$ and CF = 0, then DE = $C4D5_H$, CF = 1, JF = 0, ZF = 0, SF = 1, and VF = 0.				
SWAP g	1110 1ggg : 1111 1111	1 - - - - -	4	
Exchanges the contents of 4 high-order bits of 8-bit register g with those of 4 low-order bits of the register. Example: If this instruction is executed when A = 25_H, then A = 52_H.				
ROLD A,(x)	1110 0000 : xxxx xxxx : 1111 0110	1 - - - - -	8	
Rotates the 12-bit data consisting of the memory content at the address (0000 to $00FF_H$) specified directly by x in the object code and the lower 4-bit content of the accumulator that is concatenated to the memory content left by four bits. Note however that the upper 4-bit content of the accumulator does not change. Example: If the instruction ROLD A,(87_H) is executed when accumulator = 12_H and the content of address 0087_H = 56_H, then accumulator = 15_H and the content of address 0087_H = 62_H.				

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
ROLD A,(vw)	1110 0001 : wwww wwww : vvvv vvvv 1111 0110	1 - - - - -	9	
	Rotates the 12-bit data consisting of the memory content at the address (0000 to FFFF$_H$) specified directly by vw in the object code and the lower 4-bit content of the accumulator that is concatenated to the memory content left by four bits. Note however that the upper 4-bit content of the accumulator does not change.			
ROLD A,(DE)	1110 0010 : 1111 0110	1 - - - - -	7	
	Rotates the 12-bit data consisting of the memory content at the address specified by the DE register pair and the lower 4-bit content of the accumulator that is concatenated to the memory content left by four bits. Note however that the upper 4-bit content of the accumulator does not change.			
ROLD A,(HL)	1110 0011 : 1111 0110	1 - - - - -	7	
	Rotates the 12-bit data consisting of the memory content at the address specified by the HL register pair and the lower 4-bit content of the accumulator that is concatenated to the memory content left by four bits. Note however that the upper 4-bit content of the accumulator does not change.			
ROLD A,(IX)	1110 0100 : 1111 0110	1 - - - - -	7	
	Rotates the 12-bit data consisting of the memory content at the address specified by the index register IX and the lower 4-bit content of the accumulator that is concatenated to the memory content left by four bits. Note however that the upper 4-bit content of the accumulator does not change.			
ROLD A,(IY)	1110 0101 : 1111 0110	1 - - - - -	7	
	Rotates the 12-bit data consisting of the memory content at the address specified by the index register IY and the lower 4-bit content of the accumulator that is concatenated to the memory content left by four bits. Note however that the upper 4-bit content of the accumulator does not change.			
ROLD A,(IX+d)	1101 0100 : dddd dddd : 1111 0110	1 - - - - -	9	
	Rotates the 12-bit data consisting of the memory content at the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended and the lower 4-bit content of the accumulator that is concatenated to the memory content left by four bits. Note however that the upper 4-bit content of the accumulator does not change.			
ROLD A,(IY+d)	1101 0101 : dddd dddd : 1111 0110	1 - - - - -	9	
	Rotates the 12-bit data consisting of the memory content at the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended and the lower 4-bit content of the accumulator that is concatenated to the memory content left by four bits. Note however that the upper 4-bit content of the accumulator does not change.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
ROLD A,(SP+d)	1101 0110 : dddd dddd : 1111 0110	1 - - - - -	9	
	Rotates the 12-bit data consisting of the memory content at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended and the lower 4-bit content of the accumulator that is concatenated to the memory content left by four bits. Note however that the upper 4-bit content of the accumulator does not change.			
ROLD A,(HL+d)	1101 0111 : dddd dddd : 1111 0110	1 - - - - -	9	
	Rotates the 12-bit data consisting of the memory content at the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended and the lower 4-bit content of the accumulator that is concatenated to the memory content left by four bits. Note however that the upper 4-bit content of the accumulator does not change.			
ROLD A,(HL+C)	1110 0111 : 1111 0110 :	1 - - - - -	9	
	Rotates the 12-bit data consisting of the memory content at the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended and the lower 4-bit content of the accumulator that is concatenated to the memory content left by four bits. Note however that the upper 4-bit content of the accumulator does not change.			
ROLD A,(+SP)	1110 0110 : 1111 0110 :	1 - - - - -	8	
	Rotates the 12-bit data consisting of the memory content at the address specified by the content of the stack pointer after being incremented and the lower 4-bit content of the accumulator that is concatenated to the memory content left by four bits. Note however that the upper 4-bit content of the accumulator does not change.			
ROLD A,(PC+A)	0100 1111 : 1111 0110 :	1 - - - - -	9	
	Rotates the 12-bit data consisting of the memory content at the address specified by the content of the program counter plus the C register content that is added after being sign-extended and the lower 4-bit content of the accumulator that is concatenated to the memory content left by four bits. Note however that the upper 4-bit content of the accumulator does not change.			
RORD A,(x)	1110 0000 : xxxx xxxx : 1111 0111	1 - - - - -	8	
	Rotates the 12-bit data consisting of the memory content at the address (0000 to $00FF_H$) specified directly by x in the object code and the lower 4-bit content of the accumulator that is concatenated to the memory content right by four bits. Note however that the upper 4-bit content of the accumulator does not change. Example: If the instruction RORD A,(87H) is executed when accumulator = 12_H and the content of address $0087_H = 56_H$, then accumulator = 16_H and the content of address $0087_H = 25_H$.			
RORD A,(vw)	1110 0001 : wwww wwww : vvvv vvvv : 1111 0111	1 - - - - -	9	
	Rotates the 12-bit data consisting of the memory content at the address (0000 to $FFFF_H$) specified directly by vw in the object code and the lower 4-bit content of the accumulator that is concatenated to the memory content right by four bits. Note however that the upper 4-bit content of the accumulator does not change.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
RORD A,(DE)	1110 0010 : 1111 0111	1 - - - - -	7	
	Rotates the 12-bit data consisting of the memory content at the address specified by the DE register pair and the lower 4-bit content of the accumulator that is concatenated to the memory content right by four bits. Note however that the upper 4-bit content of the accumulator does not change.			
RORD A,(HL)	1110 0011 : 1111 0111	1 - - - - -	7	
	Rotates the 12-bit data consisting of the memory content at the address specified by the HL register pair and the lower 4-bit content of the accumulator that is concatenated to the memory content right by four bits. Note however that the upper 4-bit content of the accumulator does not change.			
RORD A,(IX)	1110 0100 : 1111 0111	1 - - - - -	7	
	Rotates the 12-bit data consisting of the memory content at the address specified by the index register IX and the lower 4-bit content of the accumulator that is concatenated to the memory content right by four bits. Note however that the upper 4-bit content of the accumulator does not change.			
RORD A,(IY)	1110 0101 : 1111 0111	1 - - - - -	7	
	Rotates the 12-bit data consisting of the memory content at the address specified by the index register IY and the lower 4-bit content of the accumulator that is concatenated to the memory content right by four bits. Note however that the upper 4-bit content of the accumulator does not change.			
RORD A,(IX+d)	1101 0100 : dddd dddd : 1111 0111	1 - - - - -	9	
	Rotates the 12-bit data consisting of the memory content at the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended and the lower 4-bit content of the accumulator that is concatenated to the memory content right by four bits. Note however that the upper 4-bit content of the accumulator does not change.			
RORD A,(IY+d)	1101 0101 : dddd dddd : 1111 0111	1 - - - - -	9	
	Rotates the 12-bit data consisting of the memory content at the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended and the lower 4-bit content of the accumulator that is concatenated to the memory content right by four bits. Note however that the upper 4-bit content of the accumulator does not change.			
RORD A,(SP+d)	1101 0110 : dddd dddd : 1111 0111	1 - - - - -	9	
	Rotates the 12-bit data consisting of the memory content at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended and the lower 4-bit content of the accumulator that is concatenated to the memory content right by four bits. Note however that the upper 4-bit content of the accumulator does not change.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
RORD A,(HL+d)	1101 0111 : dddd dddd : 1111 0111	1 - - - - -	9	
	Rotates the 12-bit data consisting of the memory content at the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended and the lower 4-bit content of the accumulator that is concatenated to the memory content right by four bits. Note however that the upper 4-bit content of the accumulator does not change.			
RORD A,(HL+C)	1110 0111 : 1111 0111	1 - - - - -	9	
	Rotates the 12-bit data consisting of the memory content at the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended and the lower 4-bit content of the accumulator that is concatenated to the memory content right by four bits. Note however that the upper 4-bit content of the accumulator does not change.			
RORD A,(+SP)	1110 0110 : 1111 0111	1 - - - - -	8	
	Rotates the 12-bit data consisting of the memory content at the address specified by the content of the stack pointer after being incremented and the lower 4-bit content of the accumulator that is concatenated to the memory content right by four bits. Note however that the upper 4-bit content of the accumulator does not change.			
RORD A,(PC+A)	0100 1111 : 1111 0111	1 - - - - -	9	
	Rotates the 12-bit data consisting of the memory content at the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended and the lower 4-bit content of the accumulator that is concatenated to the memory content right by four bits. Note however that the upper 4-bit content of the accumulator does not change.			

(4) Bit and Flag Manipulations

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
SET g.b	1110 1ggg : 1100 0bbb	Z * - - - -	3	ZF ← $\overline{g.b}$: g.b ← 1
	Places the content of a bit of 8-bit register g that is specified by b in the object code into the zero flag after inverting it and sets the bit to 1. Example: If the instruction SET A.7 is executed when A = $3C_H$, then ZF = 1 and A = BC_H.			
SET (x).b	1100 0bbb : xxxx xxxx	Z * - - - -	5	ZF ← $\overline{(x).b}$: (x).b ← 1
	Places the content of one of bits at the address (0000 to $00FF_H$) specified directly by x in the object code, with the bit itself specified by b in the object code, into the zero flag after inverting it and sets the bit to 1.			
SET (vw).b	1110 0001 : wwww wwww : vvvv vvvv 1100 0bbb	Z * - - - -	6	ZF ← $\overline{(vw).b}$: (vw).b ← 1
	Places the content of one of bits at the address (0000 to $FFFF_H$) specified directly by vw in the object code, with the bit itself specified by b in the object code, into the zero flag after inverting it and sets the bit to 1.			
SET (DE).b	1110 0010 : 1100 0bbb	Z * - - - -	4	ZF ← $\overline{(DE).b}$: (DE).b ← 1
	Places the content of one of bits at the address specified by the DE register pair, with the bit itself specified by b in the object code, into the zero flag after inverting it and sets the bit to 1.			
SET (HL).b	1110 0011 : 1100 0bbb	Z * - - - -	4	ZF ← $\overline{(HL).b}$: (HL).b ← 1
	Places the content of one of bits at the address specified by the HL register pair, with the bit itself specified by b in the object code, into the zero flag after inverting it and sets the bit to 1.			
SET (IX).b	1110 0100 : 1100 0bbb	Z * - - - -	4	ZF ← $\overline{(IX).b}$: (IX).b ← 1
	Places the content of one of bits at the address specified by the index register IX, with the bit itself specified by b in the object code, into the zero flag after inverting it and sets the bit to 1.			
SET (IY).b	1110 0101 : 1100 0bbb	Z * - - - -	4	ZF ← $\overline{(IY).b}$: (IY).b ← 1
	Places the content of one of bits at the address specified by the index register IY, with the bit itself specified by b in the object code, into the zero flag after inverting it and sets the bit to 1.			
SET (IX+d).b	1101 0100 : dddd dddd : 1100 0bbb	Z * - - - -	6	ZF ← $\overline{(IX+d).b}$: (IX+d).b ← 1
	Places the content of one of bits at the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by b in the object code, into the zero flag after inverting it and sets the bit to 1.			
SET (IY+d).b	1101 0101 : dddd dddd : 1100 0bbb	Z * - - - -	6	ZF ← $\overline{(IY+d).b}$: (IY+d).b ← 1
	Places the content of one of bits at the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by b in the object code, into the zero flag after inverting it and sets the bit to 1.			
SET (SP+d).b	1101 0110 : dddd dddd : 1100 0bbb	Z * - - - -	6	ZF ← $\overline{(SP+d).b}$: (SP+d).b ← 1
	Places the content of one of bits at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by b in the object code, into the zero flag after inverting it and sets the bit to 1.			
SET (HL+d).b	1101 0111 : dddd dddd : 1100 0bbb	Z * - - - -	6	ZF ← $\overline{(HL+d).b}$: (HL+d).b ← 1
	Places the content of one of bits at the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by b in the object code, into the zero flag after inverting it and sets the bit to 1.			
SET (HL+C).b	1110 0111 : 1100 0bbb	Z * - - - -	6	ZF ← $\overline{(HL+C).b}$: (HL+C).b ← 1
	Places the content of one of bits at the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended, with the bit itself specified by b in the object code, into the zero flag after inverting it and sets the bit to 1.			
SET (+SP).b	1110 0110 : 1100 0bbb	Z * - - - -	5	SP ← SP+1 : ZF ← $\overline{(SP).b}$: (SP).b ← 1
	Places the content of one of bits at the address specified by the content of the stack pointer after being incremented, with the bit itself specified by b in the object code, into the zero flag after inverting it and sets the bit to 1.			
SET (PC+A).b	0100 1111 : 1100 0bbb	Z * - - - -	6	ZF ← $\overline{(PC+A).b}$: (PC+A).b ← 1
	Places the content of one of bits at the address specified by the content of the program counter plus the accumulator content that is added after being sign extended, with the bit itself specified by b in the object code, into the zero flag after inverting it and sets the bit to 1.			
SET (x).A	1110 0000 : xxxx xxxx : 1111 0010	Z * - - - -	5	ZF ← $\overline{(x).A}$: (x).A ← 1
	Places the content of one of bits at the address (0000 to $00FF_H$) specified directly by x in the object code, with the bit itself specified by 3 low-order bits of the A register, into the zero flag after inverting it and sets the bit to 1.			
SET (vw).A	1110 0001 : wwww wwww : vvvv vvvv 1111 0010	Z * - - - -	6	ZF ← $\overline{(vw).A}$: (vw).A ← 1
	Places the content of one of bits at the address (0000 to $FFFF_H$) specified directly by vw in the object code, with the bit itself specified by 3 low-order bits of the A register, into the zero flag after inverting it and sets the bit to 1.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
SET (DE).A	1110 0010 : 1111 0010	Z * - - - -	4	ZF ← $\overline{(DE).A}$:(DE).A ← 1
	Places the content of one of bits at the address specified by the DE register pair, with the bit itself specified by 3 low-order bits of the A register, into the zero flag after inverting it and sets the bit to 1.			
SET (HL).A	1110 0011 : 1111 0010	Z *	4	ZF ← $\overline{(HL).A}$:(HL).A ← 1
	Places the content of one of bits at the address specified by the HL register pair, with the bit itself specified by 3 low-order bits of the A register, into the zero flag after inverting it and sets the bit to 1.			
SET (IX).A	1110 0100 : 1111 0010	Z * - - - -	4	ZF ← $\overline{(IX).A}$:(IX).A ← 1
	Places the content of one of bits at the address specified by the index register IX, with the bit itself specified by 3 low-order bits of the A register, into the zero flag after inverting it and sets the bit to 1.			
SET (IY).A	1110 0101 : 1111 0010	Z * - - - -	4	ZF ← $\overline{(IY).A}$:(IY).A ← 1
	Places the content of one of bits at the address specified by the index register IY, with the bit itself specified by 3 low-order bits of the A register, into the zero flag after inverting it and sets the bit to 1.			
SET (IX+d).A	1101 0100 : dddd dddd : 1111 0010	Z * - - - -	6	ZF ← $\overline{(IX+d).A}$:(IX+d).A ← 1
	Places the content of one of bits at the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register, into the zero flag after inverting it and sets the bit to 1.			
SET (IY+d).A	1101 0101 : dddd dddd : 1111 0010	Z * - - - -	6	ZF ← $\overline{(IY+d).A}$:(IY+d).A ← 1
	Places the content of one of bits at the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register, into the zero flag after inverting it and sets the bit to 1.			
SET (SP+d).A	1101 0110 : dddd dddd : 1111 0010	Z * - - - -	6	ZF ← $\overline{(SP+d).A}$:(SP+d).A ← 1
	Places the content of one of bits at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register, into the zero flag after inverting it and sets the bit to 1.			
SET (HL+d).A	1101 0111 : dddd dddd : 1111 0010	Z * - - - -	6	ZF ← $\overline{(HL+d).A}$:(HL+d).A ← 1
	Places the content of one of bits at the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register, into the zero flag after inverting it and sets the bit to 1.			
SET (HL+C).A	1110 0111 : 1111 0010	Z * - - - -	6	ZF ← $\overline{(HL+C).A}$:(HL+C).A ← 1
	Places the content of one of bits at the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register, into the zero flag after inverting it and sets the bit to 1.			
SET (+SP).A	1110 0110 : 1111 0010	Z * - - - -	5	SP ← SP+1:ZF ← $\overline{(SP).A}$:(SP).A ← 1
	Places the content of one of bits at the address specified by the content of the stack pointer after being incremented, with the bit itself specified by 3 low-order bits of the A register, into the zero flag after inverting it and sets the bit to 1.			
SET (PC+A).A	0100 1111 : 1111 0010	Z * - - - -	6	ZF ← $\overline{(PC+A).A}$:(PC+A).A ← 1
	Places the content of one of bits at the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register, into the zero flag after inverting it and sets the bit to 1.			
CLR g.b	1110 1ggg : 1100 1bbb	Z * - - - -	3	ZF ← $\overline{g.b}$:g.b ← 0
	Places the content of a bit of 8-bit register g that is specified by b in the object code into the zero flag after inverting it and clears the bit to 0. Example: If the instruction CLR A.2 is executed when A = $3C_H$, then ZF = 0 and A = 38_H.			
CLR (x).b	1100 1bbb : xxxx xxxx	Z * - - - -	5	ZF ← $\overline{(x).b}$:(x).b ← 0
	Places the content of one of bits at the address (0000 to $00FF_H$) specified directly by x in the object code, with the bit itself specified by b in the object code, into the zero flag after inverting it and clears the bit to 0.			
CLR (vw).b	1110 0001 : wwww wwww : vvvv vvvv : 1100 1bbb	Z * - - - -	6	ZF ← $\overline{(vw).b}$:(vw).b ← 0
	Places the content of one of bits at the address (0000 to $FFFF_H$) specified directly by vw in the object code, with the bit itself specified by b in the object code, into the zero flag after inverting it and clears the bit to 0.			
CLR (DE).b	1110 0010 : 1100 1bbb	Z * - - - -	4	ZF ← $\overline{(DE).b}$:(DE).b ← 0
	Places the content of one of bits at the address specified by the DE register pair, with the bit itself specified by b in the object code, into the zero flag after inverting it and clears the bit to 0.			
CLR (HL).b	1110 0011 : 1100 1bbb	Z * - - - -	4	ZF ← $\overline{(HL).b}$:(HL).b ← 0
	Places the content of one of bits at the address specified by the HL register pair, with the bit itself specified by b in the object code, into the zero flag after inverting it and clears the bit to 0.			
CLR (IX).b	1110 0100 : 1100 1bbb	Z * - - - -	4	ZF ← $\overline{(IX).b}$:(IX).b ← 0
	Places the content of one of bits at the address specified by the index register IX, with the bit itself specified by b in the object code, into the zero flag after inverting it and clears the bit to 0.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
CLR (IY).b	1110 0101 : 1100 1bbb	Z * - - - -	4	ZF ← $\overline{(IY).b}$: (IY).b ← 0
	Places the content of one of bits at the address specified by the index register IY, with the bit itself specified by b in the object code, into the zero flag after inverting it and clears the bit to 0.			
CLR (IX+d).b	1101 0100 : dddd dddd : 1100 1bbb	Z -	6	ZF ← $\overline{(IX+d).b}$: (IX+d).b ← 0
	Places the content of one of bits at the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by b in the object code, into the zero flag after inverting it and clears the bit to 0.			
CLR (IY+d).b	1101 0101 : dddd dddd : 1100 1bbb	Z * - - - -	6	ZF ← $\overline{(IY+d).b}$: (IY+d).b ← 0
	Places the content of one of bits at the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by b in the object code, into the zero flag after inverting it and clears the bit to 0.			
CLR (SP+d).b	1101 0110 : dddd dddd : 1100 1bbb	Z * - - - -	6	ZF ← $\overline{(SP+d).b}$: (SP+d).b ← 0
	Places the content of one of bits at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by b in the object code, into the zero flag after inverting it and clears the bit to 0.			
CLR (HL+d).b	1101 0111 : dddd dddd : 1100 1bbb	Z * - - - -	6	ZF ← $\overline{(HL+d).b}$: (HL+d).b ← 0
	Places the content of one of bits at the address specified by the content of the HL register pair plus the object code's 8 bit data d that is added after being sign-extended, with the bit itself specified by b in the object code, into the zero flag after inverting it and clears the bit to 0.			
CLR (HL+C).b	1110 0111 : 1100 1bbb	Z * - - - -	6	ZF ← $\overline{(HL+C).b}$: (HL+C).b ← 0
	Places the content of one of bits at the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended, with the bit itself specified by b in the object code, into the zero flag after inverting it and clears the bit to 0.			
CLR (+SP).b	1110 0110 : 1100 1bbb	Z * - - - -	5	SP ← SP+1 : ZF ← $\overline{(SP).b}$: (SP).b ← 0
	Places the content of one of bits at the address specified by the content of the stack pointer after being incremented, with the bit itself specified by b in the object code, into the zero flag after inverting it and clears the bit to 0.			
CLR (PC+A).b	0100 1111 : 1100 1bbb	Z * - - - -	6	ZF ← $\overline{(PC+A).b}$: (PC+A).b ← 0
	Places the content of one of bits at the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended, with the bit itself specified by b in the object code, into the zero flag after inverting it and clears the bit to 0.			
CLR (x).A	1110 0000 : xxxx xxxx : 1111 1010	Z * - - - -	5	ZF ← $\overline{(x).A}$: (x).A ← 0
	Places the content of one of bits at the address (0000 to 00FF$_H$) specified directly by x in the object code, with the bit itself specified by 3 low-order bits of the A register, into the zero flag after inverting it and clears the bit to 0.			
CLR (vw).A	1110 0001 : wwww wwww : vvvv vvvv : 1111 1010	Z * - - - -	6	ZF ← $\overline{(vw).A}$: (vw).A ← 0
	Places the content of one of bits at the address (0000 to 00FF$_H$) specified directly by vw in the object code, with the bit itself specified by 3 low-order bits of the A register, into the zero flag after inverting it and clears the bit to 0.			
CLR (DE).A	1110 0010 : 1111 1010	Z * - - - -	4	ZF ← $\overline{(DE).A}$: (DE).A ← 0
	Places the content of one of bits at the address specified by the DE register pair, with the bit itself specified by 3 low-order bits of the A register, into the zero flag after inverting it and clears the bit to 0.			
CLR (HL).A	1110 0011 : 1111 1010	Z * - - - -	4	ZF ← $\overline{(HL).A}$: (HL).A ← 0
	Places the content of one of bits at the address specified by the HL register pair, with the bit itself specified by 3 low-order bits of the A register, into the zero flag after inverting it and clears the bit to 0.			
CLR (IX).A	1110 0100 : 1111 1010	Z * - - - -	4	ZF ← $\overline{(IX).A}$: (IX).A ← 0
	Places the content of one of bits at the address specified by the index register IX, with the bit itself specified by 3 low-order bits of the A register, into the zero flag after inverting it and clears the bit to 0.			
CLR (IY).A	1110 0101 : 1111 1010	Z * - - - -	4	ZF ← $\overline{(IY).A}$: (IY).A ← 0
	Places the content of one of bits at the address specified by the index register IY, with the bit itself specified by 3 low-order bits of the A register, into the zero flag after inverting it and clears the bit to 0.			
CLR (IX+d).A	1101 0100 : dddd dddd : 1111 1010	Z * - - - -	6	ZF ← $\overline{(IX+d).A}$: (IX+d).A ← 0
	Places the content of one of bits at the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register, into the zero flag after inverting it and clears the bit to 0.			
CLR (IY+d).A	1101 0101 : dddd dddd : 1111 1010	Z * - - - -	6	ZF ← $\overline{(IY+d).A}$: (IY+d).A ← 0
	Places the content of one of bits at the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register, into the zero flag after inverting it and clears the bit to 0.			
CLR (SP+d).A	1101 0110 : dddd dddd : 1111 1010	Z * - - - -	6	ZF ← $\overline{(SP+d).A}$: (SP+d).A ← 0
	Places the content of one of bits at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register, into the zero flag after inverting it and clears the bit to 0.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
CLR (HL+d).A	1101 0111 : dddd dddd : 1111 1010	Z * - - - -	6	ZF ← $\overline{(HL+d).A}$: (HL+d).A ← 0
	Places the content of one of bits at the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register, into the zero flag after inverting it and sets the bit to 1.			
CLR (HL+C).A	1110 0111 : 1111 1010	Z * - - - -	6	ZF ← $\overline{(HL+C).A}$: (HL+C).A ← 0
	Places the content of one of bits at the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register, into the zero flag after inverting it and clears the bit to 0.			
CLR (+SP).A	1110 0110 : 1111 1010	Z * - - - -	5	SP ← SP+1 : ZF ← $\overline{(SP).A}$: (SP).A ← 0
	Places the content of one of bits at the address specified by the content of the stack pointer after being incremented, with the bit itself specified by 3 low-order bits of the A register, into the zero flag after inverting it and clears the bit to 0.			
CLR (PC+A).A	0100 1111 : 1111 1010	Z * - - - -	6	ZF ← $\overline{(PC+A).A}$: (PC+A).A ← 0
	Places the content of one of bits at the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register, into the zero flag after inverting it and clears the bit to 0.			
LD CF,g.b	1110 1ggg : 0101 1bbb	C - * - - -	2	CF ← g.b
	Loads the content of a bit of 8-bit register g that is specified by b in the object code into the carry flag. Example: If the instruction LD CF,A.4 is executed when A = 01101101_B, then CF = 0 and JF = 1.			
LD CF,(x).b	0101 1bbb : xxxx xxxx	C - * - - -	4	CF ← (x).b
	Loads the content of one of bits at the address (0000 to $00FF_H$) specified directly by x in the object code, with the bit itself specified by b in the object code, into the carry flag.			
LD CF,(vw).b	1110 0001 : wwww wwww : vvvv vvvv : 0101 1bbb	C - * - - -	5	CF ← (vw).b
	Loads the content of one of bits at the address (0000 to $FFFF_H$) specified directly by vw in the object code, with the bit itself specified by b in the object code, into the carry flag.			
LD CF,(DE).b	1110 0010 : 0101 1bbb	C - * - - -	3	CF ← (DE).b
	Loads the content of one of bits at the address specified by the DE register pair, with the bit itself specified by b in the object code, into the carry flag.			
LD CF,(HL).b	1110 0011 : 0101 1bbb	C - * - - -	3	CF ← (HL).b
	Loads the content of one of bits at the address specified by the HL register pair, with the bit itself specified by b in the object code, into the carry flag.			
LD CF,(IX).b	1110 0100 : 0101 1bbb	C - * - - -	3	CF ← (IX).b
	Loads the content of one of bits at the address specified by the index register IX, with the bit itself specified by b in the object code, into the carry flag.			
LD CF,(IY).b	1110 0101 : 0101 1bbb	C - * - - -	3	CF ← (IY).b
	Loads the content of one of bits at the address specified by the index register IY, with the bit itself specified by b in the object code, into the carry flag.			
LD CF,(IX+d).b	1101 0100 : dddd dddd : 0101 1bbb	C - * - - -	5	CF ← (:IX+d).b
	Loads the content of one of bits at the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by b in the object code, into the carry flag.			
LD CF,(IY+d).b	1101 0101 : dddd dddd : 0101 1bbb	C - * - - -	5	CF ← (IY+d).b
	Loads the content of one of bits at the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by b in the object code, into the carry flag.			
LD CF,(SP+d).b	1101 0110 : dddd dddd : 0101 1bbb	C - * - - -	5	CF ← (SP+d).b
	Loads the content of one of bits at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by b in the object code, into the carry flag.			
LD CF,(HL+d).b	1101 0111 : dddd dddd : 0101 1bbb	C - * - - -	5	CF ← (HL+d).b
	Loads the content of one of bits at the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by b in the object code, into the carry flag.			
LD CF,(HL+C).b	1110 0111 : 0101 1bbb	C - * - - -	5	CF ← (HL+C).b
	Loads the content of one of bits at the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended, with the bit itself specified by b in the object code, into the carry flag.			
LD CF,(+SP).b	1110 0110 : 0101 1bbb	C - * - - -	4	SP ← SP+1 : CF ← (SP).b
	Loads the content of one of bits at the address specified by the content of the stack pointer after being incremented, with the bit itself specified by b in the object code, into the carry flag.			

	MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
	LD CF,(PC+A).b	0100 1111 : 0101 1bbb	C - * - - -	5	CF ← (PC+A).b
		Loads the content of one of bits at the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended, with the bit itself specified by b in the object code, into the carry flag.			
	LD CF,(x).A	1110 0000 : xxxx xxxx : 1111 1100	C - * - - -	4	CF ← (x).A
		Loads the content of one of bits at the address (0000 to 00FF$_H$) specified directly by x in the object code, with the bit itself specified by 3 low-order bits of the A register, into the carry fla			
	LD CF,(vw).A	1110 0001 : wwww wwww : vvvv vvvv : 1111 1100	C - * - - -	5	CF ← (vw).A
		Loads the content of one of bits at the address (0000 to FFFF$_H$) specified directly by vw in the object code, with the bit itself specified by 3 low-order bits of the A register, into the carry flag.			
	LD CF,(DE).A	1110 0010 : 1111 1100	C - * - - -	3	CF ← (DE).A
		Loads the content of one of bits at the address specified by the DE register pair, with the bit itself specified by 3 low-order bits of the A register, into the carry flag.			
	LD CF,(HL).A	1110 0011 : 1111 1100	C - * - - -	3	CF ← (HL).A
		Loads the content of one of bits at the address specified by the HL register pair, with the bit itself specified by 3 low-order bits of the A register, into the carry flag.			
	LD CF,(IX).A	1110 0100 : 1111 1100	C - * - - -	3	CF ← (IX).A
		Loads the content of one of bits at the address specified by the index register IX, with the bit itself specified by 3 low-order bits of the A register, into the carry flag.			
	LD CF,(IY).A	1110 0101 : 1111 1100	C - * - - -	3	CF ← (IY).A
		Loads the content of one of bits at the address specified by the index register IY, with the bit itself specified by 3 low-order bits of the A register, into the carry flag.			
	LD CF,(IX+d).A	1101 0100 : dddd dddd : 1111 1100	C - * - - -	5	CF ← (IX+d).A
		Loads the content of one of bits at the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register, into the carry flag.			
	LD CF,(IY+d).A	1101 0101 : dddd dddd : 1111 1100	C - * - - -	5	CF ← (IY+d).A
		Loads the content of one of bits at the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register, into the carry flag.			
	LD CF,(SP+d).A	1101 0110 : dddd dddd : 1111 1100	C - * - - -	5	CF ← (SP+d).A
		Loads the content of one of bits at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register, into the carry flag.			
	LD CF,(HL+d).A	1101 0111 : dddd dddd : 1111 1100	C - * - - -	5	CF ← (HL+d).A
		Loads the content of one of bits at the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register, into the carry flag.			
	LD CF,(HL+C).A	1110 0111 : 1111 1100	C - * - - -	5	CF ← (HL+C).A
		Loads the content of one of bits at the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register, into the carry flag.			
	LD CF,(+SP).A	1110 0110 : 1111 1100	C - * - - -	4	SP ← SP+1:CF ← (SP).A
		Loads the content of one of bits at the address specified by the content of the stack pointer after being incremented, with the bit itself specified by 3 low-order bits of the A register, into the carry flag.			
	LD CF,(PC+A).A	0100 1111 : 1111 1100	C - * - - -	5	CF ← (PC+A).A
		Loads the content of one of bits at the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register, into the carry flag.			
*2	TEST g.b	1110 1ggg : 0101 1bbb	* - J - - -	2	JF ← $\overline{g.b}$
		Places the content of a bit of 8-bit register g that is specified by b in the object code into the jump status flag after inverting it. Example: If the instruction TEST A.5 is executed when A = 01011100$_B$, then JF = 1 and CF = 0.			
*2	TEST (x).b	0101 1bbb : xxxx xxxx	* - J - - -	4	JF ← $\overline{(x).b}$
		Places the content of one of bits at the address (0000 to 00FF$_H$) specified directly by x in the object code, with the bit itself specified by b in the object code, into the jump status flag after inverting it.			
*2	TEST (vw).b	1110 0001 : wwww wwww : vvvv vvvv : 0101 1bbb	* - J - - -	5	JF ← $\overline{(vw).b}$
		Places the content of one of bits at the address (0000 to FFFF$_H$) specified directly by vw in the object code, with the bit itself specified by b in the object code, into the jump status flag after inverting it.			

	MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
*2	TEST (DE).b	1110 0010 : 0101 1bbb	* - J - - -	3	JF ← $\overline{(DE).b}$
		Places the content of one of bits at the address specified by the DE register pair, with the bit itself specified by b in the object code, into the jump status flag after inverting it.			
*2	TEST (HL).b	1110 0011 : 0101 1bbb	* - J - - -	3	JF ← $\overline{(HL).b}$
		Places the content of one of bits at the address specified by the HL register pair, with the bit itself specified by b in the object code, into the jump status flag after inverting it.			
*2	TEST (IX).b	1110 0100 : 0101 1bbb	* - J - - -	3	JF ← $\overline{(IX).b}$
		Places the content of one of bits at the address specified by the index register IX, with the bit itself specified by b in the object code, into the jump status flag after inverting it.			
*2	TEST (IY).b	1110 0101 : 0101 1bbb	* - J - - -	3	JF ← $\overline{(IY).b}$
		Places the content of one of bits at the address specified by the index register IY, with the bit itself specified by b in the object code, into the jump status flag after inverting it.			
*2	TEST (IX+d).b	1101 0100 : dddd dddd : 0101 1bbb	* - J - - -	5	JF ← $\overline{(IX+d).b}$
		Places the content of one of bits at the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by b in the object code, into the jump status flag after inverting it.			
*2	TEST (IY+d).b	1101 0101 : dddd dddd : 0101 1bbb	* - J - - -	5	JF ← $\overline{(IY+d).b}$
		Places the content of one of bits at the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by b in the object code, into the jump status flag after inverting it.			
*2	TEST (SP+d).b	1101 0110 : dddd dddd : 0101 1bbb	* - J - - -	5	JF ← $\overline{(SP+d).b}$
		Places the content of one of bits at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by b in the object code, into the jump status flag after inverting it.			
*2	TEST (HL+d).b	1101 0111 : dddd dddd : 0101 1bbb	* - J - - -	5	JF ← $\overline{(HL+d).b}$
		Places the content of one of bits at the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by b in the object code, into the jump status flag after inverting it.			
*2	TEST (HL+C).b	1110 0111 : 0101 1bbb	* - J - - -	6	JF ← $\overline{(HL+C).b}$
		Places the content of one of bits at the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended, with the bit itself specified by b in the object code, into the jump status flag after inverting it.			
*2	TEST (+SP).b	1110 0110 : 0101 1bbb	* - J - - -	4	SP ← SP+1 : JF ← $\overline{(SP).b}$
		Places the content of one of bits at the address specified by the content of the stack pointer after being incremented, with the bit itself specified by b in the object code, into the jump status flag after inverting it.			
*2	TEST (PC+A).b	0100 1111 : 0101 1bbb	* - J - - -	5	JF ← $\overline{(PC+A).b}$
		Places the content of one of bits at the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended, with the bit itself specified by b in the object code, into the jump status flag after inverting it.			
*2	TEST (x).A	1110 0000 : xxxx xxxx : 1111 1100	* - J - - -	4	JF ← $\overline{(x).A}$
		Places the content of one of bits at the address (0000 to 00FF$_H$) specified directly by x in the object code, with the bit itself specified by 3 low-order bits of the A register, into the jump status flag after inverting it.			
*2	TEST (vw).A	1110 0001 : wwww wwww : vvvv vvvv : 1111 1100	* - J - - -	5	JF ← $\overline{(vw).A}$
		Places the content of one of bits at the address (0000 to FFFF$_H$) specified directly by vw in the object code, with the bit itself specified by 3 low-order bits of the A register, into the jump status flag after inverting it.			
*2	TEST (DE).A	1110 0010 : 1111 1100	* - J - - -	3	JF ← $\overline{(DE).A}$
		Places the content of one of bits at the address specified by the DE register pair, with the bit itself specified by 3 low-order bits of the A register, into the jump status flag after inverting it.			
*2	TEST (HL).A	1110 0011 : 1111 1100	* - J - - -	3	JF ← $\overline{(HL).A}$
		Places the content of one of bits at the address specified by the HL register pair, with the bit itself specified by 3 low-order bits of the A register, into the jump status flag after inverting it.			
*2	TEST (IX).A	1110 0100 : 1111 1100	* - J - - -	3	JF ← $\overline{(IX).A}$
		Places the content of one of bits at the address specified by the index register IX, with the bit itself specified by 3 low-order bits of the A register, into the jump status flag after inverting it.			
*2	TEST (IY).A	1110 0101 : 1111 1100	* - J - - -	3	JF ← $\overline{(IY).A}$
		Places the content of one of bits at the address specified by the index register IY, with the bit itself specified by 3 low-order bits of the A register, into the jump status flag after inverting it.			

	MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
*2	TEST (IX+d).A	1101 0100 : dddd dddd : 1111 1100	* - J - - -	5	JF ← $\overline{(IX+d).A}$
		Places the content of one of bits at the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register, into the jump status flag after inverting it.			
*2	TEST (IY+d).A	1101 0101 : dddd dddd : 1111 1100	* - J - - -	5	JF ← $\overline{(IY+d).A}$
		Places the content of one of bits at the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register, into the jump status flag after inverting it.			
*2	TEST (SP+d).A	1101 0110 : dddd dddd : 1111 1100	* - J - - -	5	JF ← $\overline{(SP+d).A}$
		Places the content of one of bits at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register, into the jump status flag after inverting it.			
*2	TEST (HL+d).A	1101 0111 : dddd dddd : 1111 1100	* - J - - -	5	JF ← $\overline{(HL+d).A}$
		Places the content of one of bits at the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register, into the jump status flag after inverting it.			
*2	TEST (HL+C).A	1110 0111 : 1111 1100	* - J - - -	5	JF ← $\overline{(HL+C).A}$
		Places the content of one of bits at the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register, into the jump status flag after inverting it.			
*2	TEST (+SP).A	1110 0110 : 1111 1100	* - J - - -	4	SP ← SP+1 : JF ← $\overline{(SP).A}$
		Places the content of one of bits at the address specified by the content of the stack pointer after being incremented, with the bit itself specified by 3 low-order bits of the A register, into the jump status flag after inverting it.			
*2	TEST (PC+A).A	0100 1111 : 1111 1100	* - J - - -	5	JF ← $\overline{(PC+A).A}$
		Places the content of one of bits at the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register, into the jump status flag after inverting it.			
	LD g.b,CF	1110 1ggg : 1110 1bbb	1 - - - - -	2	g.b ← CF
		Stores the content of the carry flag into a bit of 8-bit register g that is specified by b in the object code. Example 1: If the instruction LD A.5,CF is executed when A = 15_H and CF = 1, then A = 35_H and CF = 1. Example 2: If the instruction LD B.2,CF is executed when B = $7E_H$ and CF = 0, then B = $7A_H$ and CF = 0.			
	LD (x).b,CF	1110 0000 : xxxx xxxx : 1110 1bbb	1 - - - - -	5	(x).b ← CF
		Stores the content of the carry flag into one of bits at the address (0000 to $00FF_H$) specified directly by x in the object code, with the bit itself specified by b in the object code.			
	LD (vw).b,CF	1110 0001 : wwww wwww : vvvv vvvv : 1110 1bbb	1 - - - - -	6	(vw).b ← CF
		Stores the content of the carry flag into one of bits at the address (0000 to $FFFF_H$) specified directly by vw in the object code, with the bit itself specified by b in the object code.			
	LD (DE).b,CF	1110 0010 : 1110 1bbb	1 - - - - -	4	(DE).b ← CF
		Stores the content of the carry flag into one of bits at the address specified by the DE register pair, with the bit itself specified by b in the object code.			
	LD (HL).b,CF	1110 0011 : 1110 1bbb	1 - - - - -	4	(HL).b ← CF
		Stores the content of the carry flag into one of bits at the address specified by the HL register pair, with the bit itself specified by b in the object code.			
	LD (IX).b,CF	1110 0100 : 1110 1bbb	1 - - - - -	4	(IX).b ← CF
		Stores the content of the carry flag into one of bits at the address specified by the index register IX, with the bit itself specified by b in the object code.			
	LD (IY).b,CF	1110 0101 : 1110 1bbb	1 - - - - -	4	(IY).b ← CF
		Stores the content of the carry flag into one of bits at the address specified by the index register IY, with the bit itself specified by b in the object code.			
	LD (IX+d).b,CF	1101 0100 : dddd dddd : 1110 1bbb	1 - - - - -	6	(IX+d).b ← CF
		Stores the content of the carry flag into one of bits at the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by b in the object code.			
	LD (IY+d).b,CF	1101 0101 : dddd dddd : 1110 1bbb	1 - - - - -	6	(IY+d).b ← CF
		Stores the content of the carry flag into one of bits at the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by b in the object code.			
	LD (SP+d).b,CF	1101 0110 : dddd dddd : 1110 1bbb	1 - - - - -	6	(SP+d).b ← CF
		Stores the content of the carry flag into one of bits at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by b in the object code.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
LD (HL+d).b,CF	1101 0111 : dddd dddd : 1110 1bbb	1 - - - - -	6	(HL+d).b ← CF
	Stores the content of the carry flag into one of bits at the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by b in the object code.			
LD (HL+C).b,CF	1110 0111 : 1110 1bbb	1 - - - - -	6	(HL+C).b ← CF
	Stores the content of the carry flag into one of bits at the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended, with the bit itself specified by b in the object code.			
LD (+SP).b,CF	1110 0110 : 1110 1bbb	1 - - - - -	5	SP ← SP+1:(SP).b ← CF
	Stores the content of the carry flag into one of bits at the address specified by the content of the stack pointer after being incremented, with the bit itself specified by b in the object code.			
LD (PC+A).b,CF	0100 1111 : 1110 1bbb	1 - - - - -	6	(PC+A).b ← CF
	Stores the content of the carry flag into one of bits at the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended, with the bit itself specified by b in the object code.			
LD (x).A,CF	1110 0000 : xxxx xxxx : 1111 0011	1 - - - - -	5	(x).A ← CF
	Stores the content of the carry flag into one of bits at the address (0000 to 00FF$_H$) specified by x in the object code, with the bit itself specified by 3 low-order bits of the A register.			
LD (vw).A,CF	1110 0001 : wwww wwww : vvvv vvvv : 1111 0011	1 - - - - -	6	(vw).A ← CF
	Stores the content of the carry flag into one of bits at the address directly specified by vw in the object code, with the bit itself specified by 3 low-order bits of the A register.			
LD (DE).A,CF	1110 0010 : 1111 0011	1 - - - - -	4	(DE).A ← CF
	Stores the content of the carry flag into one of bits at the address specified by the DE register pair, with the bit itself specified by 3 low-order bits of the A register.			
LD (HL).A,CF	1110 0011 : 1111 0011	1 - - - - -	4	(HL).A ← CF
	Stores the content of the carry flag into one of bits at the address specified by the HL register pair, with the bit itself specified by 3 low-order bits of the A register.			
LD (IX).A,CF	1110 0100 : 1111 0011	1 - - - - -	4	(IX).A ← CF
	Stores the content of the carry flag into one of bits at the address specified by the index register IX, with the bit itself specified by 3 low-order bits of the A register.			
LD (IY).A,CF	1110 0101 : 1111 0011	1	4	(IY).A ← CF
	Stores the content of the carry flag into one of bits at the address specified by the index register IY, with the bit itself specified by 3 low-order bits of the A register.			
LD (IX+d).A,CF	1101 0100 : dddd dddd : 1111 0011	1 - - - - -	6	(IX+d).A ← CF
	Stores the content of the carry flag into one of bits at the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register.			
LD (IY+d).A,CF	1101 0101 : dddd dddd : 1111 0011	1 - - - - -	6	(IY+d).A ← CF
	Stores the content of the carry flag into one of bits at the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register.			
LD (SP+d).A,CF	1101 0110 : dddd dddd : 1111 0011	1 - - - - -	6	(SP+d).A ← CF
	Stores the content of the carry flag into one of bits at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register.			
LD (HL+d).A,CF	1101 0111 : dddd dddd : 1111 0011	1 - - - - -	6	(HL+d).A ← CF
	Stores the content of the carry flag into one of bits at the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register.			
LD (HL+C).A,CF	1110 0111 : 1111 0011	1 - - - - -	6	(HL+C).A ← CF
	Stores the content of the carry flag into one of bits at the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register.			
LD (+SP).A,CF	1110 0110 : 1111 0011	1 - - - - -	5	SP ← SP+1:(SP).A ← CF
	Stores the content of the carry flag into one of bits at the address specified by the content of the stack pointer after being incremented, with the bit itself specified by 3 low-order bits of the A register.			
LD (PC+A).A,CF	0100 1111 : 1111 0011	1 - - - - -	6	(PC+A).A ← CF
	Stores the content of the carry flag into one of bits at the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
CPL g.b	1110 1ggg : 1110 0bbb	Z * - - - -	3	ZF ← $\overline{g.b}$: g.b ← $\overline{g.b}$
	Places the content of a bit of 8-bit register g that is specified by b in the object code into the zero flag after inverting it and inverts the content of the bit. Example: If the instruction CPL A.3 is executed when A = $3C_H$, then ZF = 0 and A = 34_H.			
CPL (x).b	1110 0000 : xxxx xxxx : 1110 0bbb	Z * - - - -	5	ZF ← $\overline{(x).b}$: (x).b ← $\overline{(x).b}$
	Places the content of one of bits at the address (0000 to $00FF_H$) specified directly by x in the object code, with the bit itself specified by b in the object code, into the zero flag after inverting it and then inverts the content of the bit.			
CPL (vw).b	1110 0001 : wwww wwww : vvvv vvvv 1110 0bbb	Z * - - - -	8	ZF ← $\overline{(vw).b}$: (vw).b ← $\overline{(vw).b}$
	Places the content of one of bits at the address (0000 to $FFFF_H$) specified directly by vw in the object code, with the bit itself specified by b in the object code, into the zero flag after inverting it and then inverts the content of the bit.			
CPL (DE).b	1110 0010 : 1110 0bbb	Z * - - - -	4	ZF ← $\overline{(DE).b}$: (DE).b ← $\overline{(DE).b}$
	Places the content of one of bits at the address specified by the DE register pair, with the bit itself specified by b in the object code, into the zero flag after inverting it and then inverts the content of the bit.			
CPL (HL).b	1110 0011 : 1110 0bbb	Z * - - - -	4	ZF ← $\overline{(HL).b}$: (HL).b ← $\overline{(HL).b}$
	Places the content of one of bits at the address specified by the HL register pair, with the bit itself specified by b in the object code, into the zero flag after inverting it and then inverts the content of the bit.			
CPL (IX).b	1110 0100 : 1110 0bbb	Z * - - - -	4	ZF ← $\overline{(IX).b}$: (IX).b ← $\overline{(IX).b}$
	Places the content of one of bits at the address specified by the index register IX, with the bit itself specified by b in the object code, into the zero flag after inverting it and then inverts the content of the bit.			
CPL (IY).b	1110 0101 : 1110 0bbb	Z * - - - -	4	ZF ← $\overline{(IY).b}$: (IY).b ← $\overline{(IY).b}$
	Places the content of one of bits at the address specified by the index register IY, with the bit itself specified by b in the object code, into the zero flag after inverting it and then inverts the content of the bit.			
CPL (IX+d).b	1101 0100 : dddd dddd : 1110 0bbb	Z * - - - -	6	ZF ← $\overline{(IX+d).b}$: (IX+d).b ← $\overline{(IX+d).b}$
	Inverts the content of one of bits at the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by b in the object code.			
CPL (IY+d).b	1101 0101 : dddd dddd : 1110 0bbb	Z * - - - -	6	ZF ← $\overline{(IY+d).b}$: (IY+d).b ← $\overline{(IY+d).b}$
	Inverts the content of one of bits at the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by b in the object code.			
CPL (SP+d).b	1101 0110 : dddd dddd : 1110 0bbb	Z * - - - -	6	ZF ← $\overline{(SP+d).b}$: (SP+d).b ← $\overline{(SP+d).b}$
	Inverts the content of one of bits at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by b in the object code.			
CPL (HL+d).b	1101 0111 : dddd dddd : 1110 0bbb	Z * - - - -	6	ZF ← $\overline{(HL+d).b}$: (HL+d).b ← $\overline{(HL+d).b}$
	Inverts the content of one of bits at the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by b in the object code.			
CPL (HL+C).b	1110 0111 : 1110 0bbb	Z * - - - -	6	ZF ← $\overline{(HL+C).b}$: (HL+C).b ← $\overline{(HL+C).b}$
	Inverts the content of one of bits at the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended, with the bit itself specified by b in the object code.			
CPL (+SP).b	1110 0110 : 1110 0bbb	Z * - - - -	5	SP ← SP+1 : ZF ← $\overline{(SP).b}$: (SP).b ← $\overline{(SP).b}$
	Inverts the content of one of bits at the address specified by the content of the stack pointer after being incremented, with the bit itself specified by b in the object code.			
CPL (PC+A).b	0100 1111 : 1110 0bbb	Z * - - - -	6	ZF ← $\overline{(PC+A).b}$: (PC+A).b ← $\overline{(PC+A).b}$
	Inverts the content of one of bits at the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended, with the bit itself specified by b in the object code.			
CPL (x).A	1110 0000 : xxxx xxxx : 1111 1011	Z * - - - -	5	ZF ← $\overline{(x).A}$: (x).A ← $\overline{(x).A}$
	Places the content of one of bits at the address (0000 to $00FF_H$) specified directly by x in the object code, with the bit itself specified by 3 low-order bits of the A register, into the zero flag after inverting it and then inverts the content of the bit.			
CPL (vw).A	1110 0001 : wwww wwww : vvvv vvvv 1111 1011	Z * - - - -	6	ZF ← $\overline{(vw).A}$: (vw).A ← $\overline{(vw).A}$
	Places the content of one of bits at the address (0000 to $FFFF_H$) specified directly by vw in the object code, with the bit itself specified by 3 low-order bits of the A register, into the zero flag after inverting it and then inverts the content of the bit.			
CPL (DE).A	1110 0010 : 1111 1011	Z * - - - -	4	ZF ← $\overline{(DE).A}$: (DE).A ← $\overline{(DE).A}$
	Places the content of one of bits at the address specified by the DE register pair, with the bit itself specified by 3 low-order bits of the A register, into the zero flag after inverting it and then inverts the content of the bit.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
CPL (HL).A	1110 0011 : 1111 1011	Z • - - - -	4	ZF ← $\overline{(HL).A}$:(HL).A ← $\overline{(HL).A}$
	Places the content of one of bits at the address specified by the HL register pair, with the bit itself specified by 3 low-order bits of the A register, into the zero flag after inverting it and then inverts the content of the bit.			
CPL (IX).A	1110 0100 : 1111 1011	Z • - - - -	4	ZF ← $\overline{(IX).A}$:(IX).A ← $\overline{(IX).A}$
	Places the content of one of bits at the address specified by the index register IX, with the bit itself specified by 3 low-order bits of the A register, into the zero flag after inverting it and then inverts the content of the bit.			
CPL (IY).A	1110 0101 : 1111 1011	Z • - - - -	4	ZF ← $\overline{(IY).A}$:(IY).A ← $\overline{(IY).A}$
	Places the content of one of bits at the address specified by the index register IY, with the bit itself specified by 3 low-order bits of the A register, into the zero flag after inverting it and then inverts the content of the bit.			
CPL (IX+d).A	1101 0100 : dddd dddd : 1111 1011	Z • - - - -	6	ZF ← $\overline{(IX+d).A}$:(IX+d).A ← $\overline{(IX+d).A}$
	Inverts the content of one of bits at the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register.			
CPL (IY+d).A	1101 0101 : dddd dddd : 1111 1011	Z • - - - -	6	ZF ← $\overline{(IY+d).A}$:(IY+d).A ← $\overline{(IY+d).A}$
	Inverts the content of one of bits at the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register.			
CPL (SP+d).A	1101 0110 : dddd dddd : 1111 1011	Z • - - - -	6	ZF ← $\overline{(SP+d).A}$:(SP+d).A ← $\overline{(SP+d).A}$
	Inverts the content of one of bits at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register.			
CPL (HL+d).A	1101 0111 : dddd dddd : 1111 1011	Z • - - - -	6	ZF ← $\overline{(HL+d).A}$:(HL+d).A ← $\overline{(HL+d).A}$
	Inverts the content of one of bits at the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register.			
CPL (HL+C).A	1110 0111 : 1111 1011	Z • - - - -	6	ZF ← $\overline{(HL+C).A}$:(HL+C).A ← $\overline{(HL+C).A}$
	Inverts the content of one of bits at the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register.			
CPL (+SP).A	1110 0110 : 1111 1011	Z • - - - -	5	SP ← SP+1:ZF ← $\overline{(SP).A}$:(SP).A ← $\overline{(SP).A}$
	Inverts the content of one of bits at the address specified by the content of the stack pointer after being incremented, with the bit itself specified by 3 low-order bits of the A register.			
CPL (PC+A).A	0100 1111 : 1111 1011	Z • - - - -	6	ZF ← $\overline{(PC+A).A}$:(PC+A).A ← $\overline{(PC+A).A}$
	Inverts the content of one of bits at the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended, with the bit itself specified by 3 low-order bits of the A register.			
XOR CF,g.b	1110 1ggg : 0101 0bbb	C - • - - -	2	CF ← CF⊻g.b
	Logically XORs the content of a bit of 8-bit register g that is specified by b in the object code with that of the carry flag and places the result into the carry flag. Example: If the instruction XOR CF, A.3 is executed when A = $5B_H$ and CF = 1, then A = $5B_H$, CF = 0 and JF = 1.			
XOR CF,(x).b	1110 0000 : xxxx xxxx : 0101 0bbb	C - • - - -	4	CF ← CF⊻(x).b
	Logically XORs the content of one of bits at the address (0000 to 00FF$_H$) specified directly by x in the object code, the bit itself being specified by b in the object code, with that of the carry flag and places the result into the carry flag.			
XOR CF,(vw).b	1110 0001 : wwww wwww : vvvv vvvv : 0101 0bbb	C - • - - -	5	CF ← CF⊻(vw).b
	Logically XORs the content of one of bits at the address (0000 to FFFF$_H$) specified directly by vw in the object code, the bit itself being specified by b in the object code, with that of the carry flag and places the result into the carry flag.			
XOR CF,(DE).b	1110 0010 : 0101 0bbb	C - • - - -	3	CF ← CF⊻(DE).b
	Logically XORs the content of one of bits at the address specified by the DE register pair, the bit itself being specified by b in the object code, with that of the carry flag and places the result into the carry flag.			
XOR CF,(HL).b	1110 0011 : 0101 0bbb	C - • - - -	3	CF ← CF⊻(HL).b
	Logically XORs the content of one of bits at the address specified by the HL register pair, the bit itself being specified by b in the object code, with that of the carry flag and places the result into the carry flag.			
XOR CF,(IX).b	1110 0100 : 0101 0bbb	C - • - - -	3	CF ← CF⊻(IX).b
	Logically XORs the content of one of bits at the address specified by the index register IX, the bit itself being specified by b in the object code, with that of the carry flag and places the result into the carry flag.			
XOR CF,(IY).b	1110 0101 : 0101 0bbb	C - • - - -	3	CF ← CF⊻(IY).b
	Logically XORs the content of one of bits at the address specified by the index register IY, the bit itself being specified by b in the object code, with that of the carry flag and places the result into the carry flag.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
XOR CF,(IX+d).b	1101 0100 : dddd dddd : 0101 0bbb	$\overline{C}$ - * - - -	5	CF ← CF ⊻ (IX+d).b
	Logically XORs the content of one of bits at the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended, the bit itself being specified by b in the object code, with that of the carry flag and places the result into the carry flag.			
XOR CF,(IY+d).b	1101 0101 : dddd dddd : 0101 0bbb	$\overline{C}$ - * - - -	5	CF ← CF ⊻ (IY+d).b
	Logically XORs the content of one of bits at the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended, the bit itself being specified by b in the object code, with that of the carry flag and places the result into the carry flag.			
XOR CF,(SP+d).b	1101 0110 : dddd dddd : 0101 0bbb	$\overline{C}$ - * - - -	5	CF ← CF ⊻ (SP+d).b
	Logically XORs the content of one of bits at the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended, the bit itself being specified by b in the object code, with that of the carry flag and places the result into the carry flag.			
XOR CF,(HL+d).b	1101 0111 : dddd dddd : 0101 0bbb	$\overline{C}$ - * - - -	5	CF ← CF ⊻ (HL+d).b
	Logically XORs the content of one of bits at the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended, the bit itself being specified by b in the object code, with that of the carry flag and places the result into the carry flag.			
XOR CF,(HL+C).b	1110 0111 : 0101 0bbb	$\overline{C}$ - * - - -	5	CF ← CF ⊻ (HL+C).b
	Logically XORs the content of one of bits at the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended, the bit itself being specified by b in the object code, with that of the carry flag and places the result into the carry flag.			
XOR CF,(+SP).b	1110 0110 : 0101 0bbb	$\overline{C}$ - * - - -	4	SP ← SP+1:CF ← CF ⊻ (SP).b
	Logically XORs the content of one of bits at the address specified by the content of the stack pointer after being incremented, the bit itself being specified by b in the object code, with that of the carry flag and places the result into the carry flag.			
XOR CF,(PC+A).b	0100 1111 : 0101 0bbb	$\overline{C}$ - * - - -	5	CF ← CF ⊻ (PC+A).b
	Logically XORs the content of one of bits at the address specified by the content of the program counter plus the accumulator content that is added after being sign-extended, the bit itself being specified by b in the object code, with that of the carry flag and places the result into the carry flag.			
SET CF	0000 0101	0 - 1 - - -	1	CF ← 1
	Sets the carry flag to 1.			
CLR CF	0000 0100	1 - 0 - - -	1	CF ← 0
	Clears the carry flag to 0.			
CPL CF	0000 0110	* - * - - -	1	JF ← CF:CF ← $\overline{CF}$
	Places the content of the carry flag into the jump status flag and then inverts the carry flag. Example: If this instruction is executed when CF = 0, then JF = 0 and CF = 1.			
*2 DI	1100 1000 : 0011 1010	Z * - - - -	5	ZF ← $\overline{IMF}$:IMF ← 0
	Places the content of the interrupt master enable flag into the zero flag after inverting it and then clears the interrupt master enable flag to 0. (Maskable interrupts thereby are disabled against acceptance.)			
*2 EI	1100 0000 : 0011 1010	Z * - - - -	5	ZF ← $\overline{IMF}$:IMF ← 1
	Places the content of the interrupt master enable flag into the zero flag after inverting it and then sets the interrupt master enable flag to 1. (Maskable interrupts thereby are enabled for acceptance.)			

(5) Jump

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
JRS T,$+2+d	100d dddd	1 - - - - -	4/2(t/f)	if JF=1 then PC ← PC+d else null
	When the jump status flag = 1, this instruction causes control to jump to the address specified by the content of the program counter (indicating two addresses ahead the location where the JRS instruction is stored) plus 5-bit displacement d in the object code that is added after being sign-extended (−16 to +15). (In this case, the instruction is executed in 4 cycles.) When the jump status flag = 0, sets the jump status flag to 1 and goes on to execute the next instruction. (In this case, the instruction is executed in 2 cycles.) Example: If the instruction JRS T,$+9 stored at address $C134_H$ is executed when the jump status flag = 1, control jumps to address $C13D_H$.			
JRS F,$+2+d	101d dddd	1 - - - - -	4/2(t/f)	if JF=0 then PC ← PC+d else null
	When the jump status flag = 0, this instruction causes control to jump to the address specified by the content of the program counter plus 5-bit displacement d in the object code that is added after being sign-extended (−16 to +15) (4 cycles). When the jump status flag = 1, performs nothing and goes on to execute the next instruction (2 cycles).			
JR T,$+2+d	1101 1110 dddd dddd	1 - - - - -	4/2(t/f)	if JF=1 then PC ← PC+d else null
	When the jump status flag = 1, this instruction causes control to jump to the address specified by the content of the program counter (indicating two addresses ahead the location where the JR instruction is stored) plus 8-bit displacement d in the object code that is added after being sign extended (−128 to +127) (4 cycles). When the jump status flag = 0, sets the jump status flag to 1 and goes on to execute the next instruction (2 cycles). Example: If the instruction JR T,$+0F611 stored at address $C134_H$ is executed when the jump status flag = 1, control jumps to address $C12A_H$.			
JR F,$+2+d	1101 1111 dddd dddd	1	4/2(t/f)	if JF=0 then PC ← PC+d else null
	When the jump status flag = 0, this instruction causes control to jump to the address specified by the content of the program counter plus 8-bit displacement d in the object code that is added after being sign-extended (−128 to +127) (4 cycles). When the jump status flag = 1, performs nothing and goes on to execute the next instruction (2 cycles).			
JR EQ,$+2+d JR Z,$+2+d	1101 1000 dddd dddd	1 - - - - -	4/2(t/f)	if ZF=1 then PC ← PC+d else null
	When the zero flag = 1, this instruction causes control to jump to the address specified by the content of the program counter plus 8-bit displacement d in the object code that is added after being sign-extended (−128 to +127) (4 cycles). When the zero flag = 0, sets the jump status flag to 1 and goes on to execute the next instruction (2 cycles).			
JR NE,$+2+d JR NZ,$+2+d	1101 1001 dddd dddd	1 - - - - -	4/2(t/f)	if ZF=0 then PC ← PC+d else null
	When the zero flag = 0, this instruction causes control to jump to the address specified by the content of the program counter plus 8-bit displacement d in the object code that is added after being sign-extended (−128 to +127) (4 cycles). When the zero flag = 1, sets the jump status flag to 1 and goes on to execute the next instruction (2 cycles).			
JR CS,$+2+d JR LT,$+2+d	1101 1010 dddd dddd	1 - - - - -	4/2(t/f)	if CF=1 then PC ← PC+d else null
	When the carry flag = 1, this instruction causes control to jump to the address specified by the content of the program counter plus 8-bit displacement d in the object code that is added after being sign-extended (−128 to +127) (4 cycles). When the carry flag = 0, sets the jump status flag to 1 and goes on to execute the next instruction (2 cycles).			
JR CC,$+2+d JR GE,$+2+d	When the carry flag = 0, this instruction causes control to jump to the address specified by the content of the program counter plus 8-bit displacement d in the object code that is added after being sign-extended (−128 to +127) (4 cycles). When the carry flag = 1, sets the jump status flag to 1 and goes on to execute the next instruction (2 cycles)			
JR LE,$+2+d	1101 1100 dddd dddd	1 - - - - -	4/2(t/f)	if (CF∨ZF)=1 then PC ← PC+d else null
	When the carry or zero flag = 1, this instruction causes control to jump to the address specified by the content of the program counter plus 8-bit displacement d in the object code that is added after being sign-extended (−128 to +127) (4 cycles). When both carry and zero flags = 0, sets the jump status flag to 1 and goes on to execute the next instruction (2 cycles).			
JR GT,$+2+d	1101 1101 dddd dddd	1 - - - - -	4/2(t/f)	if (CF∨ZF)=0 then PC ← PC+d else null
	When both carry and zero flags = 0, this instruction causes control to jump to the address specified by the content of the program counter plus 8-bit displacement d in the object code that is added after being sign-extended (−128 to +127) (4 cycles). When the carry or zero flag = 1, sets the jump status flag to 1 and goes on to execute the next instruction (2 cycles).			
JR M,$+3+d	1110 1000 1101 0000 dddd dddd	1 - - - - -	5/3(t/f)	if SF=1 then PC ← PC+d else null
	When the sign flag = 1, this instruction causes control to jump to the address specified by the content of the program counter (indicating three addresses ahead the location where the JR instruction is stored) plus 8-bit displacement d in the object code that is added after being sign-extended (−128 to +127) (5 cycles). When the sign flag = 0, sets the jump status flag to 1 and goes on to execute the next instruction (3 cycles).			
JR P,$+3+d	1110 1000 1101 0001 dddd dddd	1 - - - - -	5/3(t/f)	if SF=0 then PC ← PC+d else null
	When the sign flag = 0, this instruction causes control to jump to the address specified by the content of the program counter plus 8-bit displacement d in the object code that is added after being sign-extended (−128 to +127) (5 cycles). When the sign flag = 1, sets the jump status flag to 1 and goes on to execute the next instruction (3 cycles).			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
JR SLT,$+3+d	1110 1000 : 1101 0010 : dddd dddd	1 - - - - -	5/3(t/f)	if (SF⊻VF)=1 then PC ← PC+d else null
	When the XOR'ed result of the sign and overflow flags = 1, this instruction causes control to jump to the address specified by the content of the program counter plus 8-bit displacement d in the object code that is added after being sign-extended (−128 to +127) (5 cycles). When the XOR'ed result of the sign and overflow flags = 0, sets the jump status flag to 1 and goes on to execute the next instruction (3 cycles).			
JR SGE,$+3+d	1110 1000 : 1101 0011 : dddd dddd	1 - - - - -	5/3(t/f)	if (SF⊻VF)=0 then PC ← PC+d else null
	When the XOR'ed result of the sign and overflow flags = 0, this instruction causes control to jump to the address specified by the content of the program counter plus 8-bit displacement d in the object code that is added after being sign-extended (−128 to +127) (5 cycles). When the XOR'ed result of the sign and overflow flags = 1, sets the jump status flag to 1 and goes on to execute the next instruction (3 cycles).			
JR SLE,$+3+d	1110 1000 : 1101 0100 : dddd dddd	1 - - - - -	5/3(t/f)	if ZF∨(SF⊻VF)=1 then PC ← PC+d else null
	When the XOR'ed result of the sign and overflow flags or the content of the zero flag = 1, this instruction causes control to jump to the address specified by the content of the program counter plus 8-bit displacement d in the object code that is added after being sign-extended (−128 to +127) (5 cycles). When both of the XOR'ed result of the sign and overflow flags and the content of the zero flag = 0, sets the jump status flag to 1 and goes on to execute the next instruction (3 cycles).			
JR SGT,$+3+d	1110 1000 : 1101 0101 : dddd dddd	1 - - - - -	5/3(t/f)	if ZF∨(SF⊻VF)=0 then PC ← PC+d else null
	When both of the XOR'ed result of the sign and overflow flags and the content of the zero flag = 0, this instruction causes control to jump to the address specified by the content of the program counter plus 8-bit displacement d in the object code that is added after being sign-extended (−128 to +127) (5 cycles). When the XOR'ed result of the sign and overflow flags or the content of the zero flag = 1, sets the jump status flag to 1 and goes on to execute the next instruction (3 cycles).			
JR VS,$+3+d	1110 1000 : 1101 0110 : dddd dddd	1 - - - - -	5/3(t/f)	if VF=1 then PC ← PC+d else null
	When the overflow flag = 1, this instruction causes control to jump to the address specified by the content of the program counter plus 8-bit displacement d in the object code that is added after being sign-extended (−128 to +127) (5 cycles). When the overflow flag = 0, sets the jump status flag to 1 and goes on to execute the next instruction (3 cycles).			
JR VC,$+3+d	1110 1000 : 1101 0111 : dddd dddd	1 - - - - -	5/3(t/f)	if VF=0 then PC ← PC+d else null
	When the overflow flag = 0, this instruction causes control to jump to the address specified by the content of the program counter plus 8-bit displacement d in the object code that is added after being sign-extended (−128 to +127) (5 cycles). When the overflow flag = 1, sets the jump status flag to 1 and goes on to execute the next instruction (3 cycles).			
JR $+2+d	1111 1100 : dddd dddd	1 - - - - -	4	PC ← PC+d
	Causes control to unconditionally jump to the address specified by the content of the program counter (indicating two addresses ahead the location where the JR instruction is stored) plus 8-bit displacement d in the object code that is added after being sign-extended (−128 to +127). Example: If the instruction JR $+73$_H$ stored at address D5A7$_H$ is executed, control jumps to address D61CH.			
JP mn	1111 1110 : nnnn nnnn : mmmm mmmm	- - - - - -	4	PC ← mn
	Causes control to unconditionally jump to the address specified directly by 16-bit data mn in the object code.			
JP gg	1110 1ggg : 1111 1110	- - - - - -	3	PC ← gg
	Causes control to unconditionally jump to the address specified by the 16-bit register gg. Example: If the instruction JP HL is executed when HL = E325$_H$, control jumps to address E325$_H$.			
JP (x)	1110 0000 : xxxx xxxx : 1111 1110	- - - - - -	6	PC ← (x+1,x)
	Causes control to unconditionally jump to the address stored in consecutive 2-byte memory locations beginning with the address (0000 to 00FF$_H$) that is specified directly by x in the object code. Example: If the instruction JP (85$_H$) is executed when the contents of addresses 0085$_H$ and 0086$_H$ respectively are 27$_H$ and C3H, control jumps to address C327$_H$.			
JP (vw)	1110 0001 : wwww wwww : vvvv vvvv : 1111 1110	- - - - - -	7	PC ← (vw+1,vw)
	Causes control to unconditionally jump to the address stored in consecutive 2-byte memory locations beginning with the address (0000 to FFFF$_H$) that is specified directly by vw in the object code.			
JP (DE)	1110 0010 : 1111 1110	- - - - - -	5	PC ← (DE+1,DE)
	Causes control to unconditionally jump to the address stored in consecutive 2 byte memory locations beginning with the address that is specified by the DE register pair. Example: If the instruction JP (DE) is executed when DE = 0125$_H$ and the contents of addresses 0125$_H$ and 0126$_H$ respectively are 87$_H$ and E5$_H$, control jumps to address E587$_H$.			
JP (HL)	1110 0011 : 1111 1110	- - - - - -	5	PC ← (HL+1,HL)
	Causes control to unconditionally jump to the address stored in consecutive 2-byte memory locations beginning with the address that is specified by the HL register pair. Example: If the instruction JP (HL) is executed when HL = 0125$_H$ and the contents of addresses 0125$_H$ and 0126$_H$ respectively are 87$_H$ and E5$_H$, control jumps to address E587$_H$.			
JP (IX)	1110 0100 : 1111 1110	- - - - - -	5	PC ← (IX+1,IX)
	Causes control to unconditionally jump to the address stored in consecutive 2-byte memory locations beginning with the address that is specified by the index register IX. Example: If the instruction JP (IX) is executed when IX = 0125$_H$ and the contents of addresses 0125$_H$ and 0126$_H$ respectively are 87$_H$ and E5$_H$, control jumps to address E587$_H$.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
JP (IY)	1110 0101 : 1111 1110	- - - - - -	5	PC ← (IY+1,IY)
	Causes control to unconditionally jump to the address stored in consecutive 2-byte memory locations beginning with the address that is specified by the index register IY. Example: If the instruction JP (IY) is executed when IY = 0125_H and the contents of addresses 0125_H and 0126_H respectively are 87_H and $E5_H$, control jumps to address $E587_H$.			
JP (IX+d)	1101 0100 : dddd dddd : 1111 1110	- - - - - -	7	PC ← (IX+d+1,IX+d)
	Causes control to unconditionally jump to the address stored in consecutive 2-byte memory locations beginning with the address that is specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended.			
JP (IY+d)	1101 0101 : dddd dddd : 1111 1110	- - - - - -	7	PC ← (IY+d+1,IY+d)
	Causes control to unconditionally jump to the address stored in consecutive 2-byte memory locations beginning with the address that is specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended.			
JP (SP+d)	1101 0110 : dddd dddd : 1111 1110	- - - - - -	7	PC ← (SP+d+1,SP+d)
	Causes control to unconditionally jump to the address stored in consecutive 2-byte memory locations beginning with the address that is specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended.			
JP (HL+d)	1101 0111 : dddd dddd : 1111 1110	- - - - - -	7	PC ← (HL+d+1,HL+d)
	Causes control to unconditionally jump to the address stored in consecutive 2-byte memory locations beginning with the address that is specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended.			
JP (HL+C)	1110 0111 : 1111 1110	- - - - - -	7	PC ← (HL+C+1,HL+C)
	Causes control to unconditionally jump to the address stored in consecutive 2-byte memory locations beginning with the address that is specified by the content of the HL register pair plus the C register content that is added after being sign-extended.			
JP (+SP)	1110 0110 : 1111 1110	- - - - - -	6	SP ← SP+1:PC ← (SP+1,SP)
	Causes control to unconditionally jump to the address stored in consecutive 2-byte memory locations beginning with the address that is specified by the content of the stack pointer after being incremented.			
JP (PC+A)	0100 1111 : 1111 1110	- - - - - -	7	PC ← (PC+A+1,PC+A)
	Causes control to unconditionally jump to the address stored in consecutive 2-byte program memory locations beginning with the address that is specified by the content of the program counter PC (indicating two addresses ahead the location where the JR instruction is stored) plus the accumulator content that is added after being sign-extended (MSB is the sign bit). This instruction is suitable for multidirection branch processing.			

(6) Subroutine call, return, software interrupt, and no operation

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
CALLV n	0111 nnnn	- - - - - -	7	(SP,SP-1) ← PC-1:SP ← SP-2: PC ← (n×2+FFC1H,n×2+FFC0H)
	Performs a vectored subroutine call. Namely, this instruction saves the content of the program counter PC after subtracting by 1 (address where the CALLV instruction is stored + 1, i.e., the return address) in order of the upper and the lower bytes to data memory at the address specified by the stack pointer SP and decrements the content of the stack pointer twice. Then it places into the program counter the consecutive 2 bytes of program memory contents (vector) beginning with the address specified by 4-bit immediate data n (0 to 15) in the object code multiplied by 2 plus $FFC0_H$. This instruction helps to compact the program size. Example: If the instruction CALLV 4_H stored at address $D1A4_H$ is executed when SP = 0123_H and the contents of addresses $FFC8_H$ and $FFC9_H$ respectively are 45_H and $E6_H$, data $D1_H$ and $A5_H$ respectively are written to addresses 0123_H and 0122_H, the SP is decremented to 0121_H, and address $E645_H$ is called.			
CALL mn	1111 1101 nnnn nnnn mmmm mmmm	- - - - - -	6	(SP,SP-1) ← PC+1:SP ← SP-2: PC ← mn
	Performs a long-absolute subroutine call. Namely, this instruction saves the content of the program counter PC plus 1 (the return address) in order of the upper and the lower bytes to data memory at the address specified by the stack pointer SP and decrements the content of the stack pointer twice. Then it places the 16-bit absolute address mn in the object code into the program counter. Example: If the instruction CALL $0CE05_H$ stored at address $E273_H$ is executed when SP = 0138_H, data $E2_H$ and 76_H respectively are written to addresses 0138_H and 0137_H, the SP is decremented to 0136_H, and address $CE05_H$ is called.			
CALL gg	1110 1ggg 1111 1101	- - - - - -	6	(SP,SP-1) ← PC+1:SP ← SP-2: PC ← gg
	Saves the content of the program counter PC (the return address) in order of the upper and the lower bytes to data memory at the address specified by the stack pointer SP and decrements the content of the stack pointer twice. Then it places the content of 16-bit register gg into the program counter. Example: If the instruction CALL HL stored at address $D491_H$ is executed when SP = 0126_H and HL = $C8A7_H$, data $D4_H$ and 93_H respectively are written to addresses 0126_H and 0125_H, the SP is decremented to 0124_H, and address $C8A7_H$ is called.			
CALL (x)	1110 0000 xxxx xxxx 1111 1101	- - - - - -	9	(SP,SP-1) ← PC+1:SP ← SP-2: PC ← (x+1,x)
	Saves the content of the program counter PC plus 1 (the return address) in order of the upper and the lower bytes to data memory at the address specified by the stack pointer SP and decrements the content of the stack pointer twice. Then it places into the program counter the consecutive 2 bytes of memory contents (subroutine's entry address) beginning with the address (0000 to $00FF_H$) specified directly by x in the object code.			
CALL (vw)	1110 0001 wwww wwww vvvv vvvv 1111 1101	- - - - - -	10	(SP,SP-1) ← PC+1:SP ← SP-2: PC ← (vw+1,vw)
	Saves the content of the program counter PC plus 2 (the return address) in order of the upper and the lower bytes to data memory at the address specified by the stack pointer SP and decrements the content of the stack pointer twice. Then it places into the program counter the consecutive 2 bytes of memory contents beginning with the address (0000 to $FFFF_H$) specified directly by vw in the object code.			
CALL (DE)	1110 0010 1111 1101	- - - - - -	8	(SP,SP-1) ← PC+1:SP ← SP-2: PC ← (DE+1,DE)
	Saves the content of the program counter PC (the return address) in order of the upper and the lower bytes to data memory at the address specified by the stack pointer SP and decrements the content of the stack pointer twice. Then it places into the program counter the consecutive 2 bytes of memory contents beginning with the address specified by the content of the DE register pair.			
CALL (HL)	1110 0011 1111 1101	- - - - - -	8	(SP,SP-1) ← PC+1:SP ← SP-2: PC ← (HL+1,HL)
	Saves the content of the program counter PC (the return address) in order of the upper and the lower bytes to data memory at the address specified by the stack pointer SP and decrements the content of the stack pointer twice. Then it places into the program counter the consecutive 2 bytes of memory contents beginning with the address specified by the content of the HL register pair.			
CALL (IX)	1110 0100 1111 1101	- - - - - -	8	(SP,SP-1) ← PC+1:SP ← SP-2: PC ← (IX+1,IX)
	Saves the content of the program counter PC (the return address) in order of the upper and the lower bytes to data memory at the address specified by the stack pointer SP and decrements the content of the stack pointer twice. Then it places into the program counter the consecutive 2 bytes of memory contents beginning with the address specified by the content of the index register IX.			
CALL (IY)	1110 0101 1111 1101	- - - - - -	8	(SP,SP-1) ← PC+1:SP ← SP-2: PC ← (IY+1,IY)
	Saves the content of the program counter PC (the return address) in order of the upper and the lower bytes to data memory at the address specified by the stack pointer SP and decrements the content of the stack pointer twice. Then it places into the program counter the consecutive 2 bytes of memory contents beginning with the address specified by the content of the index register IY.			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
CALL (IX+d)	1101 0100 : dddd dddd : 1111 1101	- - - - - -	10	(SP,SP-1) ← PC+1:SP ← SP-2: PC ← (IX+d+1,IX+d)
	Saves the content of the program counter PC plus 1 (the return address) in order of the upper and the lower bytes to data memory at the address specified by the stack pointer SP and decrements the content of the stack pointer twice. Then it places into the program counter the consecutive 2 bytes of memory contents beginning with the address specified by the content of the index register IX plus the object code's 8-bit data d that is added after being sign-extended.			
CALL (IY+d)	1101 0101 : dddd dddd : 1111 1101	- - - - - -	10	(SP,SP-1) ← PC+1:SP ← SP-2: PC ← (IY+d+1,IY+d)
	Saves the content of the program counter PC plus 1 (the return address) in order of the upper and the lower bytes to data memory at the address specified by the stack pointer SP and decrements the content of the stack pointer twice. Then it places into the program counter the consecutive 2 bytes of memory contents beginning with the address specified by the content of the index register IY plus the object code's 8-bit data d that is added after being sign-extended.			
CALL (SP+d)	1101 0110 : dddd dddd : 1111 1101	- - - - - -	10	(SP,SP-1) ← PC+1:SP ← SP-2: PC ← (SP+d+1,SP+d)
	Saves the content of the program counter PC plus 1 (the return address) in order of the upper and the lower bytes to data memory at the address specified by the stack pointer SP and decrements the content of the stack pointer twice. Then it places into the program counter the consecutive 2 bytes of memory contents beginning with the address specified by the content of the stack pointer plus the object code's 8-bit data d that is added after being sign-extended.			
CALL (HL+d)	1101 0111 : dddd dddd : 1111 1101	- - - - - -	10	(SP,SP-1) ← PC+1:SP ← SP-2:
	Saves the content of the program counter PC plus 1 (the return address) in order of the upper and the lower bytes to data memory at the address specified by the stack pointer SP and decrements the content of the stack pointer twice. Then it places into the program counter the consecutive 2 bytes of memory contents beginning with the address specified by the content of the HL register pair plus the object code's 8-bit data d that is added after being sign-extended.			
CALL (+SP)	1110 0110 : 1111 1101 :	- - - - - -	9	(SP+1,SP) ← PC:SP ← SP-1: PC ← (SP+3,SP+2)
	Decrements the content of the stack pointer. Then it saves the lower byte of the program counter to memory at the address specified by the content of the stack pointer minus 1 and places the memory content whose address is specified by the stack pointer into the lower type of the program counter. Furthermore, it saves the upper byte of the program counter to memory at the address specified by the content of the stack pointer and places the memory content whose address is specified by the content of the stack pointer plus 1 into the upper type of the program counter. Finally, it decrements the content of the stack pointer twice.			
CALL (HL+C)	1110 0111 : 1111 1101 :	- - - - - -	10	(SP,SP-1) ← PC+1:SP ← SP-2: PC ← (HL+C+1,HL+C)
	Saves the content of the program counter PC (the return address) in order of the upper and the lower bytes to data memory at the address specified by the stack pointer SP and decrements the content of the stack pointer twice. Then it places into the program counter the consecutive 2 bytes of memory contents beginning with the address specified by the content of the HL register pair plus the C register content that is added after being sign-extended (MSB is the sign bit).			
CALL (PC+A)	0100 1111 : 1111 1101 :	- - - - - -	10	(SP,SP-1) ← PC+1:SP ← SP-2:
	Saves the content of the program counter PC (the return address) in order of the upper and the lower bytes to data memory at the address specified by the stack pointer SP and decrements the content of the stack pointer twice. Then it places into the program counter the consecutive 2 bytes of memory contents beginning with the address specified by the content of the program counter plus the content of accumulator A that is added after being sign-extended (MSB is the sign bit).			
RET	1111 1010 :	- - - - - -	6	SP ← SP+2:PC ← (SP,SP-1)
	Returns from a subroutine. Namely, this instruction increments the content of the stack pointer SP twice and loads into the program counter PC the content (return address) of the data memory whose address is specified by the incremented SP value, in order of the lower and the upper types. Example: If this instruction is executed when SP = 0123_H and the contents of addresses 0124_H and 0125_H respectively are $3E_H$ and $C8_H$, the SP is incremented to 0125_H and control jumps to address $C83E_H$.			
RETI	1111 1011 :	* * * * * *	6	SP ← SP+3:PC ← (SP-1,SP-2):
	Returns from a maskable interrupt service routine. Namely, this instruction increments the content of the stack pointer SP twice and loads into the program counter PC the content (return address) of the data memory whose address is specified by the incremented SP value. Then it increments the content of the stack pointer again and loads the content (return address) of the data memory whose address is specified by the incremented SP value into the program status word PSW and interrupt master enable flag IMF (bits 7 to 2 placed in the PSW, and bit 0 placed in the IMF).			
RETN	1110 1000 : 1111 1011 :	* * * * * *	7	SP ← SP+3:PC ← (SP-1,SP-2): PSW ← (SP):IMF ← (SP).0
	Returns from a nonmaskable interrupt service routine. Namely, this instruction increments the content of the stack pointer SP twice and loads into the program counter PC the content (return address) of the data memory whose address is specified by the incremented SP value. Then it increments the content of the stack pointer again and loads the content (return address) of the data memory whose address is specified by the incremented SP value into the program status word PSW and interrupt master enable flag IMF (bits 7 to 2 placed in the PSW, and bit 0 placed in the IMF).			

MNEMONIC	OBJECT CODE (Binary)	FLAG J Z C H S V	CYCLE	FUNCTION
SWI	1111 1111	- - - - - -	9	(SP) ← PSW:(SP).0 ← IMF: (SP-1,SP-2) ← PC-1: PC ← (FFFDH,FFFCH)
	Interrupts the CPU in software. Namely, this instruction saves the contents of the program status word PSW and interrupt master enable flag IMF to data memory at the address specified by the content of the stack pointer SP (PSW content saved to memory bits 7 to 2, and IMF content saved to memory bit 0) and decrements the content of the stack pointer. Next, it saves the content of the program counter PC after subtracting by 1 (address where the SWI instruction is stored + 1, i.e., the return address) in order of the upper and the lower bytes to data memory at the address specified by the content of the stack pointer and decrements the content of the stack pointer twice again. The interrupt master enable flag IMF is cleared to 0. Then the program memory content (vector) at addresses $FFFC_H$ and $FFFD_H$ are placed into the program counter. Example: If the SWI instruction stored at address $CA74_H$ is executed when $SP = 0128_H$, $PSW = 46_H$, and the contents of addresses $FFFC_H$ and $FFFD_H$ respectively are 03_H and $E7_H$, data 46_H, 75_H, and CA_H respectively are written to addresses 0128_H, 0127_H, and 0126_H, the SP is decremented to 0125_H, the IMF is cleared to 0, and control jumps to address $E703_H$.			
NOP	0000 0000	- - - - - -	1	no Operation
	Performs no operation. (Goes on to execute the next instruction without performing anything.)			

TLCS-870/C MACHINE INSTRUCTION

(1) Data transfer, exchange

MNEMONIC	OBJECT CODE (Binary)			FLAG J Z C H S V	CYCLE	FUNCTION
LD A,r	0001 0rrr			1 Z - - - -	1	A ← r
LD r,A	0100 0rrr			1 Z - - - -	1	r ← A
LD r,g	1110 1ggg	0100 0rrr		1 Z - - - -	1	r ← g
LD rr,gg	1110 1ggg	0100 1rrr		1 - - - - -	2	rr ← gg
LD A,(x)	0000 1100	xxxx xxxx		1 Z - - - -	3	A ← (x)
LD A,(HL)	0000 1101			1 Z - - - -	2	A ← (HL)
LD r,(x)	1110 0000	xxxx xxxx	0100 0rrr	1 Z - - - -	4	r ← (x)
LD r,(vw)	1110 0001	wwww wwww	vvvv vvvv	1 Z - - - -	5	r ← (vw)
	0100 0rrr					
LD r,(DE)	1110 0010	0100 0rrr		1 Z - - - -	3	r ← (DE)
LD r,(HL)	1110 0011	0100 0rrr		1 Z - - - -	3	r ← (HL)
LD r,(IX)	1110 0100	0100 0rrr		1 Z - - - -	3	r ← (IX)
LD r,(IY)	1110 0101	0100 0rrr		1 Z - - - -	3	r ← (IY)
LD r,(IX+d)	1101 0100	dddd dddd	0100 0rrr	1 Z - - - -	5	r ← (IX+d)
LD r,(IY+d)	1101 0101	dddd dddd	0100 0rrr	1 Z - - - -	5	r ← (IY+d)
LD r,(SP+d)	1101 0110	dddd dddd	0100 0rrr	1 Z - - - -	5	r ← (SP+d)
LD r,(HL+d)	1101 0111	dddd dddd	0100 0rrr	1 Z - - - -	5	r ← (HL+d)
LD r,(HL+C)	1110 0111	0100 0rrr		1 Z - - - -	5	r ← (HL+C)
LD r,(+SP)	1110 0110	0100 0rrr		1 Z - - - -	4	SP ← SP+1:r ← (SP)
LD r,(PC+A)	0100 1111	0100 0rrr		1 Z - - - -	5	r ← (PC+A)
LD rr,(x)	1110 0000	xxxx xxxx	0100 1rrr	1 - - - - -	5	rr ← (x+1,x)
LD rr,(vw)	1110 0001	wwww wwww	vvvv vvvv	1 - - - - -	6	rr ← (vw+1,vw)
	0100 1rrr					
LD rr,(DE)	1110 0010	0100 1rrr		1 - - - - -	4	rr ← (DE+1,DE)
LD rr,(HL)	1110 0011	0100 1rrr		1 - - - - -	4	rr ← (HL+1.HL)
LD rr,(IX)	1110 0100	0100 1rrr		1 - - - - -	4	rr ← (IX+1,IX)
LD rr,(IY)	1110 0101	0100 1rrr		1 - - - - -	4	rr ← (IY+1,IY)
LD rr,(IX+d)	1101 0100	dddd dddd	0100 1rrr	1 - - - - -	6	rr ← (IX+d+1,IX+d)
LD rr,(IY+d)	1101 0101	dddd dddd	0100 1rrr	1 - - - - -	6	rr ← (IY+d+1,IY+d)
LD rr,(SP+d)	1101 0110	dddd dddd	0100 1rrr	1 - - - - -	6	rr ← (SP+d+1,SP+d)
LD rr,(HL+d)	1101 0111	dddd dddd	0100 1rrr	1 - - - - -	6	rr ← (HL+d+1,HL+d)
LD rr,(HL+C)	1110 0111	0100 1rrr		1 - - - - -	6	rr ← (HL+C+1,HL+C)
LD rr,(+SP)	1110 0110	0100 1rrr		1 - - - - -	5	SP ← SP+1:rr ← (SP+1,SP)
LD rr,(PC+A)	0100 1111	0100 1rrr		1 - - - - -	6	rr ← (PC+A+1,PC+A)
LD (x),A	0000 1110	xxxx xxxx		1 - - - - -	3	(x) ← A
LD (HL),A	0000 1111			1 - - - - -	2	(HL) ← A
LD (x),r	1111 0000	xxxx xxxx	0111 1rrr	1 - - - - -	4	(x) ← r
LD (vw),r	1111 0001	wwww wwww	vvvv vvvv	1 - - - - -	5	(vw) ← r
	0111 1rrr					
LD (DE),r	1111 0010	0111 1rrr		1 - - - - -	3	(DE) ← r
LD (HL),r	1111 0011	0111 1rrr		1 - - - - -	3	(HL) ← r
LD (IX),r	1111 0100	0111 1rrr		1 - - - - -	3	(IX) ← r
LD (IY),r	1111 0101	0111 1rrr		1 - - - - -	3	(IY) ← r
LD (IX+d),r	0101 0100	dddd dddd	0111 1rrr	1 - - - - -	5	(IX+d) ← r
LD (IY+d),r	0101 0101	dddd dddd	0111 1rrr	1 - - - - -	5	(IY+d) ← r
LD (SP+d),r	0101 0110	dddd dddd	0111 1rrr	1 - - - - -	5	(SP+d) ← r
LD (HL+d),r	0101 0111	dddd dddd	0111 1rrr	1 - - - - -	5	(HL+d) ← r
LD (HL+C),r	1111 0111	0111 1rrr		1 - - - - -	5	(HL+C) ← r
LD (SP-),r	1111 0110	0111 1rrr		1 - - - - -	4	(SP) ← r:SP ← SP-1
LD (x),rr	1111 0000	xxxx xxxx	0110 1rrr	1 - - - - -	6	(x+1,x) ← rr
LD (vw),rr	1111 0001	wwww wwww	vvvv vvvv	1 - - - - -	6	(vw+1,vw) ← rr
	0110 1rrr					
LD (DE),rr	1111 0010	0110 1rrr		1 - - - - -	4	(DE+1,DE) ← rr
LD (HL),rr	1111 0011	0110 1rrr		1 - - - - -	4	(HL+1,HL) ← rr
LD (IX),rr	1111 0100	0110 1rrr		1 - - - - -	4	(IX+1,IX) ← rr
LD (IY),rr	1111 0101	0110 1rrr		1 - - - - -	4	(IY+1,IY) ← rr
LD (IX+d),rr	0101 0100	dddd dddd	0110 1rrr	1 - - - - -	6	(IX+d+1,IX+d) ← rr
LD (IY+d),rr	0101 0101	dddd dddd	0110 1rrr	1 - - - - -	6	(IY+d+1,IY+d) ← rr
LD (SP+d),rr	0101 0110	dddd dddd	0110 1rrr	1 - - - - -	6	(SP+d+1,SP+d) ← rr
LD (HL+d),rr	0101 0111	dddd dddd	0110 1rrr	1 - - - - -	6	(HL+d+1.HL+d) ← rr
LD (HL+C),rr	1111 0111	0110 1rrr		1 - - - - -	6	(HL+C+1,HL+C) ← rr
LD (SP-),rr	1111 0110	0110 1rrr		1 - - - - -	5	(SP+1,SP) ← rr:SP ← SP-1

	MNEMONIC	OBJECT CODE (Binary)				FLAG J Z C H S V	CYCLE	FUNCTION
	LD r,n	0001 1rrr	nnnn nnnn			1 - - - - -	2	r ← n
	LD rr, mn	0100 1rrr	nnnn nnnn	mmmm mmmm		1 - - - - -	3	rr ← mn
	LD (x),n	0000 1010	xxxx xxxx	nnnn nnnn		1 - - - - -	4	(x) ← n
	LD (vw),n	1111 0001	wwww wwww	vvvv vvvv		1 - - - - -	6	(vw) ← n
		1111 1001	nnnn nnnn					
	LD (DE),n	1111 0010	1111 1001	nnnn nnnn		1 - - - - -	4	(DE) ← n
	LD (HL),n	0000 1011	nnnn nnnn			1 - - - - -	4	(HL) ← n
	LD (IX),n	1111 0100	1111 1001	nnnn nnnn		1 - - - - -	4	(IX) ← n
	LD (IY),n	1111 0101	1111 1001	nnnn nnnn		1 - - - - -	4	(IY) ← n
	LD (IX+d),n	0101 0100	dddd dddd	1111 1001		1 - - - - -	6	(IX+d) ← n
		nnnn nnnn						
	LD (IY+d),n	0101 0101	dddd dddd	1111 1001		1 - - - - -	6	(IY+d) ← n
		nnnn nnnn						
	LD (SP+d),n	0101 0110	dddd dddd	1111 1001		1 - - - - -	6	(SP+d) ← n
		nnnn nnnn						
	LD (HL+d),n	0101 0111	dddd dddd	1111 1001		1 - - - - -	6	(HL+d) ← n
		nnnn nnnn						
	LD (HL+C),n	1111 0111	1111 1001	nnnn nnnn		1 - - - - -	6	(HL+C) ← n
	LD (SP-),n	1111 0110	1111 1001	nnnn nnnn		1 - - - - -	5	(SP) ← n:SP ← SP-1
	LDW (x),mn	0000 1000	xxxx xxxx	nnnn nnnn		1 - - - - -	6	(x+1,x) ← mn
		mmmm mmmm						
	LDW (HL),mn	0000 1001	nnnn nnnn	mmmm mmmm		1 - - - - -	5	(HL+1,HL) ← mn
*1	PUSH rr	0101 00rr				- - - - - -	3	(SP,SP-1) ← rr:SP ←SP-2
	PUSH gg	1110 1ggg	1101 1000			- - - - - -	4	(SP,SP-1) ← gg:SP ←SP-2
*1	POP rr	1101 00rr				- - - - - -	4	SP ← SP+2:rr ← (SP,SP-1)
	POP gg	1110 1ggg	1101 1001			- - - - - -	5	SP ← SP+2:gg ← (SP,SP-1)
	PUSH PSW	1110 1000	1101 1100			- - - - - -	3	(SP) ← PSW:SP ← SP-1
	POP PSW	1110 1000	1101 1101			* * * * * *	4	SP ← SP+1:PSW ← (SP)
	LD PSW,n	1110 1000	1101 1110	nnnn nnnn		* * * * * *	3	PSW ← n
	LD SP,SP+d	0011 0111	dddd dddd			1 - - - - -	3	SP ← SP+d
	LD SP,SP-d	0011 1111	dddd dddd			1 - - - - -	3	SP ← SP-d
	XCH r,g	1110 1ggg	0111 0rrr			1 Z - - - -	3	r ⇔ g
	XCH rr,gg	1110 1ggg	0111 1rrr			1 - - - - -	3	rr ⇔ gg
	XCH r,(x)	1110 0000	xxxx xxxx	0111 0rrr		1 Z - - - -	5	r ⇔ (x)
	XCH r,(vw)	1110 0001	wwww wwww	vvvv vvvv		1 Z - - - -	6	r ⇔ (vw)
		0111 0rrr						
	XCH r,(DE)	1110 0010	0111 0rrr			1 Z - - - -	4	r ⇔ (DE)
	XCH r,(HL)	1110 0011	0111 0rrr			1 Z - - - -	4	r ⇔ (HL)
	XCH r,(IX)	1110 0100	0111 0rrr			1 Z - - - -	4	r ⇔ (IX)
	XCH r,(IY)	1110 0101	0111 0rrr			1 Z - - - -	4	r ⇔ (IY)
	XCH r,(IX+d)	1101 0100	dddd dddd	0111 0rrr		1 Z - - - -	6	r ⇔ (IX+d)
	XCH r,(IY+d)	1101 0101	dddd dddd	0111 0rrr		1 Z - - - -	6	r ⇔ (IY+d)
	XCH r,(SP+d)	1101 0110	dddd dddd	0111 0rrr		1 Z - - - -	6	r ⇔ (SP+d)
	XCH r,(HL+d)	1101 0111	dddd dddd	0111 0rrr		1 Z - - - -	6	r ⇔ (HL+d)
	XCH r,(HL+C)	1110 0111	0111 0rrr			1 Z - - - -	6	r ⇔ (HL+C)
	XCH r,(+SP)	1110 0110	0111 0rrr			1 Z - - - -	5	SP ← SP+1:r ⇔ (SP)
	XCH r,(PC+A)	0100 1111	0111 0rrr			1 Z - - - -	6	r ⇔ (PC+A)
	XCH rr,(x)	1110 0000	xxxx xxxx	1101 1rrr		1 - - - - -	7	rr ⇔ (x+1,x)
	XCH rr,(vw)	1110 0001	wwww wwww	vvvv vvvv		1 - - - - -	8	rr ⇔ (vw+1,vw)
		1101 1rrr						
	XCH rr,(DE)	1110 0010	1101 1rrr			1 - - - - -	6	rr ⇔ (DE+1,DE)
	XCH rr,(HL)	1110 0011	1101 1rrr			1 - - - - -	6	rr ⇔ (HL+1,HL)
	XCH rr,(IX)	1110 0100	1101 1rrr			1 - - - - -	6	rr ⇔ (IX+1,IX)
	XCH rr,(IY)	1110 0101	1101 1rrr			1 - - - - -	6	rr ⇔ (IY+1,IY)
	XCH rr,(IX+d)	1101 0100	dddd dddd	1101 1rrr		1 - - - - -	8	rr ⇔ (IX+d+1,IX+d)
	XCH rr,(IY+d)	1101 0101	dddd dddd	1101 1rrr		1 - - - - -	8	rr ⇔ (IY+d+1,IY+d)
	XCH rr,(SP+d)	1101 0110	dddd dddd	1101 1rrr		1 - - - - -	8	rr ⇔ (SP+d+1,SP+d)
	XCH rr,(HL+d)	1101 0111	dddd dddd	1101 1rrr		1 - - - - -	8	rr ⇔ (HL+d+1,HL+d)
	XCH rr,(HL+C)	1110 0111	1101 1rrr			1 - - - - -	8	rr ⇔ (HL+C+1,HL+C)
	XCH rr,(+SP)	1110 0110	1101 1rrr			1 - - - - -	7	SP ← SP+1:rr ⇔ (SP+1,SP)
	XCH rr,(PC+A)	0100 1111	1101 1rrr			1 - - - - -	8	rr ⇔ (PC+A+1,PC+A)

(2) Arithmetic and logical operation

MNEMONIC	OBJECT CODE (Binary)			FLAG J Z C H S V	CYCLE	FUNCTION
CMP A,n	0110 0111	nnnn nnnn		Z Z C H S V	2	A-n
CMP g,n	1110 1ggg	0110 0111	nnnn nnnn	Z Z C H S V	3	g-n
CMP gg,mn	1110 1ggg mmmm mmmm	0110 1111	nnnn nnnn	Z Z C U S V	4	gg-mn
CMP r,g	1110 1ggg	00rr r111		Z Z C H S V	2	r-g
CMP rr,gg	1110 1ggg	10rr r111		Z Z C U S V	4	rr-gg
CMP r,(x)	1110 0000	xxxx xxxx	00rr r111	Z Z C H S V	4	r-(x)
CMP r,(vw)	1110 0001 00rr r111	wwww wwww	vvvv vvvv	Z Z C H S V	5	r-(vw)
CMP r,(DE)	1110 0010	00rr r111		Z Z C H S V	3	r-(DE)
CMP r,(HL)	1110 0011	00rr r111		Z Z C H S V	3	r-(HL)
CMP r,(IX)	1110 0100	00rr r111		Z Z C H S V	3	r-(IX)
CMP r,(IY)	1110 0101	00rr r111		Z Z C H S V	3	r-(IY)
CMP r,(IX+d)	1101 0100	dddd dddd	00rr r111	Z Z C H S V	5	r-(IX+d)
CMP r,(IY+d)	1101 0101	dddd dddd	00rr r111	Z Z C H S V	5	r-(IY+d)
CMP r,(SP+d)	1101 0110	dddd dddd	00rr r111	Z Z C H S V	5	r-(SP+d)
CMP r,(HL+d)	1101 0111	dddd dddd	00rr r111	Z Z C H S V	5	r-(HL+d)
CMP r,(HL+C)	1110 0111	00rr r111		Z Z C H S V	5	r-(HL+C)
CMP r,(+SP)	1110 0110	00rr r111		Z Z C H S V	4	SP ← SP+1:r-(SP)
CMP r,(PC+A)	0100 1111	00rr r111		Z Z C H S V	5	r-(PC+A)
CMP (x),n	0000 0111	xxxx xxxx	nnnn nnnn	Z Z C H S V	5	(x)-n
CMP (vw),n	1110 0001 0110 0111	wwww wwww	vvvv vvvv nnnn nnnn	Z Z C H S V	6	(vw)-n
CMP (DE),n	1110 0010	0110 0111	nnnn nnnn	Z Z C H S V	4	(DE)-n
CMP (HL),n	1110 0011	0110 0111	nnnn nnnn	Z Z C H S V	4	(HL)-n
CMP (IX),n	1110 0100	0110 0111	nnnn nnnn	Z Z C H S V	4	(IX)-n
CMP (IY),n	1110 0101	0110 0111	nnnn nnnn	Z Z C H S V	4	(IY)-n
CMP (IX+d),n	1101 0100 nnnn nnnn	dddd dddd	0110 0111	Z Z C H S V	6	(IX+d)-n
CMP (IY+d),n	1101 0101 nnnn nnnn	dddd dddd	0110 0111	Z Z C H S V	6	(IY+d)-n
CMP (SP+d),n	1101 0110 nnnn nnnn	dddd dddd	0110 0111	Z Z C H S V	6	(SP+d)-n
CMP (HL+d),n	1101 0111 nnnn nnnn	dddd dddd	0110 0111	Z Z C H S V	6	(HL+d)-n
CMP (HL+C),n	1110 0111	0110 0111	nnnn nnnn	Z Z C H S V	6	(HL+C)-n
CMP (+SP),n	1110 0110	0110 0111	nnnn nnnn	Z Z C H S V	5	SP ← SP+1:(SP)-n
CMP (PC+A),n	0100 1111	0110 0111	nnnn nnnn	Z Z C H S V	6	(PC+A)-n
CMP rr,(x)	1110 0000	xxxx xxxx	10rr r111	Z Z C U S V	6	rr-(x)
CMP rr,(vw)	1110 0001 10rr r111	wwww wwww	vvvv vvvv	Z Z C U S V	7	rr-(vw)
CMP rr,(DE)	1110 0010	10rr r111		Z Z C U S V	5	rr-(DE)
CMP rr,(HL)	1110 0011	10rr r111		Z Z C U S V	5	rr-(HL)
CMP rr,(IX)	1110 0100	10rr r111		Z Z C U S V	5	rr-(IX)
CMP rr,(IY)	1110 0101	10rr r111		Z Z C U S V	5	rr-(IY)
CMP rr,(IX+d)	1101 0100	dddd dddd	10rr r111	Z Z C U S V	7	rr-(IX+d)
CMP rr,(IY+d)	1101 0101	dddd dddd	10rr r111	Z Z C U S V	7	rr-(IY+d)
CMP rr,(SP+d)	1101 0110	dddd dddd	10rr r111	Z Z C U S V	7	rr-(SP+d)
CMP rr,(HL+d)	1101 0111	dddd dddd	10rr r111	Z Z C U S V	7	rr-(HL+d)
CMP rr,(HL+C)	1110 0111	10rr r111		Z Z C U S V	7	rr-(HL+C)
CMP rr,(+SP)	1110 0110	10rr r111		Z Z C U S V	6	SP ← SP+1:rr-(SP)
CMP rr,(PC+A)	0100 1111	10rr r111		Z Z C U S V	7	rr-(PC+A)
ADD A,n	0110 0001	nnnn nnnn		C Z C H S V	2	A ← A+n
ADD g,n	1110 1ggg	0110 0001	nnnn nnnn	C Z C H S V	3	g ← g+n
ADD gg,mn	1110 1ggg mmmm mmmm	0110 1001	nnnn nnnn	C Z C U S V	4	gg ← gg+mn
ADD r,g	1110 1ggg	00rr r001		C Z C H S V	2	r ← r+g
ADD rr,gg	1110 1ggg	10rr r001		C Z C U S V	4	rr ← rr+gg
ADD r,(x)	1110 0000	xxxx xxxx	00rr r001	C Z C H S V	4	r ← r+(x)
ADD r,(vw)	1110 0001 00rr r001	wwww wwww	vvvv vvvv	C Z C H S V	5	r ← r+(vw)
ADD r,(DE)	1110 0010	00rr r001		C Z C H S V	3	r ← r+(DE)
ADD r,(HL)	1110 0011	00rr r001		C Z C H S V	3	r ← r+(HL)
ADD r,(IX)	1110 0100	00rr r001		C Z C H S V	3	r ← r+(IX)
ADD r,(IY)	1110 0101	00rr r001		C Z C H S V	3	r ← r+(IY)
ADD r,(IX+d)	1101 0100	dddd dddd	00rr r001	C Z C H S V	5	r ← r+(IX+d)
ADD r,(IY+d)	1101 0101	dddd dddd	00rr r001	C Z C H S V	5	r ← r+(IY+d)
ADD r,(SP+d)	1101 0110	dddd dddd	00rr r001	C Z C H S V	5	r ← r+(SP+d)
ADD r,(HL+d)	1101 0111	dddd dddd	00rr r001	C Z C H S V	5	r ← r+(HL+d)
ADD r,(HL+C)	1110 0111	00rr r001		C Z C H S V	5	r ← r+(HL+C)
ADD r,(+SP)	1110 0110	00rr r001		C Z C H S V	4	SP ← SP+1:r ← r+(SP)
ADD r,(PC+A)	0100 1111	00rr r001		C Z C H S V	5	r ← r+(PC+A)

MNEMONIC	OBJECT CODE (Binary)			FLAG J Z C H S V	CYCLE	FUNCTION
ADD (x),n	1110 0000 nnnn nnnn	xxxx xxxx	0110 0001	C Z C H S V	6	(x) ← (x)+n
ADD (vw),n	1110 0001 0110 0001	wwww wwww nnnn nnnn	vvvv vvvv	C Z C H S V	7	(vw) ← (vw)+n
ADD (DE),n	1110 0010	0110 0001	nnnn nnnn	C Z C H S V	5	(DE) ← (DE)+n
ADD (HL),n	1110 0011	0110 0001	nnnn nnnn	C Z C H S V	5	(HL) ← (HL)+n
ADD (IX),n	1110 0100	0110 0001	nnnn nnnn	C Z C H S V	5	(IX) ← (IX)+n
ADD (IY),n	1110 0101	0110 0001	nnnn nnnn	C Z C H S V	5	(IY) ← (IY)+n
ADD (IX+d),n	1101 0100 nnnn nnnn	dddd dddd	0110 0001	C Z C H S V	7	(IX+d) ← (IX+d)+n
ADD (IY+d),n	1101 0101 nnnn nnnn	dddd dddd	0110 0001	C Z C H S V	7	(IY+d) ← (IY+d)+n
ADD (SP+d),n	1101 0110 nnnn nnnn	dddd dddd	0110 0001	C Z C H S V	7	(SP+d) ← (SP+d)+n
ADD (HL+d),n	1101 0111 nnnn nnnn	dddd dddd	0110 0001	C Z C H S V	7	(HL+d) ← (HL+d)+n
ADD (HL+C),n	1110 0111	0110 0001	nnnn nnnn	C Z C H S V	7	(HL+C) ← (HL+C)+n
ADD (+SP),n	1110 0110	0110 0001	nnnn nnnn	C Z C H S V	6	SP ← SP+1:(SP) ← (SP)+n
ADD (PC+A),n	0100 1111	0110 0001	nnnn nnnn	C Z C H S V	7	(PC+A) ← (PC+A)+n
ADD rr,(x)	1110 0000	xxxx xxxx	10rr r001	C Z C U S V	6	rr ← rr+(x)
ADD rr,(vw)	1110 0001 10rr r001	wwww wwww	vvvv vvvv	C Z C U S V	7	rr ← rr+(vw)
ADD rr,(DE)	1110 0010	10rr r001		C Z C U S V	5	rr ← rr+(DE)
ADD rr,(HL)	1110 0011	10rr r001		C Z C U S V	5	rr ← rr+(HL)
ADD rr,(IX)	1110 0100	10rr r001		C Z C U S V	5	rr ← rr+(IX)
ADD rr,(IY)	1110 0101	10rr r001		C Z C U S V	5	rr ← rr+(IY)
ADD rr,(IX+d)	1101 0100	dddd dddd	10rr r001	C Z C U S V	7	rr ← rr+(IX+d)
ADD rr,(IY+d)	1101 0101	dddd dddd	10rr r001	C Z C U S V	7	rr ← rr+(IY+d)
ADD rr,(SP+d)	1101 0110	dddd dddd	10rr r001	C Z C U S V	7	rr ← rr+(SP+d)
ADD rr,(HL+d)	1101 0111	dddd dddd	10rr r001	C Z C U S V	7	rr ← rr+(HL+d)
ADD rr,(HL+C)	1110 0111	10rr r001		C Z C U S V	7	rr ← rr+(HL+C)
ADD rr,(+SP)	1110 0110	10rr r001		C Z C U S V	6	SP ← SP+1:rr ← rr+(SP)
ADD rr,(PC+A)	0100 1111	10rr r001		C Z C U S V	7	rr ← rr+(PC+A)
ADDC A,n	0110 0000	nnnn nnnn		C Z C H S V	2	A ← A+n+CF
ADDC g,n	1110 1ggg	0110 0000	nnnn nnnn	C Z C H S V	3	g ← g+n+CF
ADDC gg,mn	1110 1ggg mmmm mmmm	0110 1000	nnnn nnnn	C Z C U S V	4	gg ← gg+mn+CF
ADDC r,g	1110 1ggg	00rr r000		C Z C H S V	2	r ← r+g+CF
ADDC rr,gg	1110 1ggg	10rr r000		C Z C U S V	4	rr ← rr+gg+CF
ADDC r,(x)	1110 0000	xxxx xxxx	00rr r000	C Z C H S V	4	r ← r+(x)+CF
ADDC r,(vw)	1110 0001 00rr r000	wwww wwww	vvvv vvvv	C Z C H S V	5	r ← r+(vw)+CF
ADDC r,(DE)	1110 0010	00rr r000		C Z C H S V	3	r ← r+(DE)+CF
ADDC r,(HL)	1110 0011	00rr r000		C Z C H S V	3	r ← r+(HL)+CF
ADDC r,(IX)	1110 0100	00rr r000		C Z C H S V	3	r ← r+(IX)+CF
ADDC r,(IY)	1110 0101	00rr r000		C Z C H S V	3	r ← r+(IY)+CF
ADDC r,(IX+d)	1101 0100	dddd dddd	00rr r000	C Z C H S V	5	r ← r+(IX+d)+CF
ADDC r,(IY+d)	1101 0101	dddd dddd	00rr r000	C Z C H S V	5	r ← r+(IY+d)+CF
ADDC r,(SP+d)	1101 0110	dddd dddd	00rr r000	C Z C H S V	5	r ← r+(SP+d)+CF
ADDC r,(HL+d)	1101 0111	dddd dddd	00rr r000	C Z C H S V	5	r ← r+(HL+d)+CF
ADDC r,(HL+C)	1110 0111	00rr r000		C Z C H S V	5	r ← r+(HL+C)+CF
ADDC r,(+SP)	1110 0110	00rr r000		C Z C H S V	4	SP ← SP+1:r ← r+(SP)+CF
ADDC r,(PC+A)	0100 1111	00rr r000		C Z C H S V	5	r ← r+(PC+A)+CF
ADDC (x),n	1110 0000 nnnn nnnn	xxxx xxxx	0110 0000	C Z C H S V	6	(x) ← (x)+n+CF
ADDC (vw),n	1110 0001 0110 0000	wwww wwww nnnn nnnn	vvvv vvvv	C Z C H S V	7	(vw) ← (vw)+n+CF
ADDC (DE),n	1110 0010	0110 0000	nnnn nnnn	C Z C H S V	5	(DE) ← (DE)+n+CF
ADDC (HL),n	1110 0011	0110 0000	nnnn nnnn	C Z C H S V	5	(HL) ← (HL)+n+CF
ADDC (IX),n	1110 0100	0110 0000	nnnn nnnn	C Z C H S V	5	(IX) ← (IX)+n+CF
ADDC (IY),n	1110 0101	0110 0000	nnnn nnnn	C Z C H S V	5	(IY) ← (IY)+n+CF
ADDC (IX+d),n	1101 0100 nnnn nnnn	dddd dddd	0110 0000	C Z C H S V	7	(IX+d) ← (IX+d)+n+CF
ADDC (IY+d),n	1101 0101 nnnn nnnn	dddd dddd	0110 0000	C Z C H S V	7	(IY+d) ← (IY+d)+n+CF
ADDC (SP+d),n	1101 0110 nnnn nnnn	dddd dddd	0110 0000	C Z C H S V	7	(SP+d) ← (SP+d)+n+CF
ADDC (HL+d),n	1101 0111 nnnn nnnn	dddd dddd	0110 0000	C Z C H S V	7	(HL+d) ← (HL+d)+n+CF
ADDC (HL+C),n	1110 0111	0110 0000	nnnn nnnn	C Z C H S V	7	(HL+C) ← (HL+C)+n+CF
ADDC (+SP),n	1110 0110	0110 0000	nnnn nnnn	C Z C H S V	6	SP ← SP+1:(SP) ← SP+n+CF
ADDC (PC+A),n	0100 1111	0110 0000	nnnn nnnn	C Z C H S V	7	(PC+A) ← (PC+A)+n+CF

MNEMONIC	OBJECT CODE (Binary)			FLAG J Z C H S V	CYCLE	FUNCTION
ADDC rr,(x)	1110 0000	xxxx xxxx	10rr r000	C Z C U S V	6	rr ← rr+(x)+CF
ADDC rr,(vw)	1110 0001	wwww wwww	vvvv vvvv	C Z C U S V	7	rr ← rr+(vw)+CF
	10rr r000					
ADDC rr,(DE)	1110 0010	10rr r000		C Z C U S V	5	rr ← rr+(DE)+CF
ADDC rr,(HL)	1110 0011	10rr r000		C Z C U S V	5	rr ← rr+(HL)+CF
ADDC rr,(IX)	1110 0100	10rr r000		C Z C U S V	5	rr ← rr+(IX)+CF
ADDC rr,(IY)	1110 0101	10rr r000		C Z C U S V	5	rr ← rr+(IY)+CF
ADDC rr,(IX+d)	1101 0100	dddd dddd	10rr r000	C Z C U S V	7	rr ← rr+(IX+d)+CF
ADDC rr,(IY+d)	1101 0101	dddd dddd	10rr r000	C Z C U S V	7	rr ← rr+(IY+d)+CF
ADDC rr,(SP+d)	1101 0110	dddd dddd	10rr r000	C Z C U S V	7	rr ← rr+(SP+d)+CF
ADDC rr,(HL+d)	1101 0111	dddd dddd	10rr r000	C Z C U S V	7	rr ← rr+(HL+d)+CF
ADDC rr,(HL+C)	1110 0111	10rr r000		C Z C U S V	7	rr ← rr+(HL+C)+CF
ADDC rr,(+SP)	1110 0110	10rr r000		C Z C U S V	6	SP ← SP+1;rr ← rr+(SP)
ADDC rr,(PC+A)	0100 1111	10rr r000		C Z C U S V	7	rr ← rr+(PC+A)+CF
SUB A,n	0110 0011	nnnn nnnn		C Z C H S V	2	A ← A-n
SUB g,n	1110 1ggg	0110 0011	nnnn nnnn	C Z C H S V	3	g ← g-n
SUB gg,mn	1110 1ggg	0110 1011	nnnn nnnn	C Z C U S V	4	gg ← gg-mn
	mmmm mmmm					
SUB r,g	1110 1ggg	00rr r011		C Z C H S V	2	r ← r-g
SUB rr,qq	1110 1qqq	10rr r011		C Z C U S V	4	rr ← rr-qq
SUB r,(x)	1110 0000	xxxx xxxx	00rr r011	C Z C H S V	4	r ← r-(x)
SUB r,(vw)	1110 0001	wwww wwww	vvvv vvvv	C Z C H S V	5	r ← r-(vw)
	00rr r011					
SUB r,(DE)	1110 0010	00rr r011		C Z C H S V	3	r ← r-(DE)
SUB r,(HL)	1110 0011	00rr r011		C Z C H S V	3	r ← r-(HL)
SUB r,(IX)	1110 0100	00rr r011		C Z C H S V	3	r ← r-(IX)
SUB r,(IY)	1110 0101	00rr r011		C Z C H S V	3	r ← r-(IY)
SUB r,(IX+d)	1101 0100	dddd dddd	00rr r011	C Z C H S V	5	r ← r-(IX+d)
SUB r,(IY+d)	1101 0101	dddd dddd	00rr r011	C Z C H S V	5	r ← r-(IY+d)
SUB r,(SP+d)	1101 0110	dddd dddd	00rr r011	C Z C H S V	5	r ← r-(SP+d)
SUB r,(HL+d)	1101 0111	dddd dddd	00rr r011	C Z C H S V	5	r ← r-(HL+d)
SUB r,(HL+C)	1110 0111	00rr r011		C Z C H S V	5	r ← r-(HL+C)
SUB r,(+SP)	1110 0110	00rr r011		C Z C H S V	4	SP ← SP+1;r ← r-(SP)
SUB r,(PC+A)	0100 1111	00rr r011		C Z C H S V	5	r ← r-(PC+A)
SUB (x),n	1110 0000	xxxx xxxx	0110 0011	C Z C H S V	6	(x) ← (x)-n
	nnnn nnnn					
SUB (vw),n	1110 0001	wwww wwww	vvvv vvvv	C Z C H S V	7	(vw) ← (vw)-n
	0110 0011	nnnn nnnn				
SUB (DE),n	1110 0010	0110 0011	nnnn nnnn	C Z C H S V	5	(DE) ← (DE)-n
SUB (HL),n	1110 0011	0110 0011	nnnn nnnn	C Z C H S V	5	(HL) ← (HL)-n
SUB (IX),n	1110 0100	0110 0011	nnnn nnnn	C Z C H S V	5	(IX) ← (IX)-n
SUB (IY),n	1110 0101	0110 0011	nnnn nnnn	C Z C H S V	5	(IY) ← (IY)-n
SUB (IX+d),n	1101 0100	dddd dddd	0110 0011	C Z C H S V	7	(IX+d) ← (IX+d)-n
	nnnn nnnn					
SUB (IY+d),n	1101 0101	dddd dddd	0110 0011	C Z C H S V	7	(IY+d) ← (IY+d)-n
	nnnn nnnn					
SUB (SP+d),n	1101 0110	dddd dddd	0110 0011	C Z C H S V	7	(SP+d) ← (SP+d)-n
	nnnn nnnn					
SUB (HL+d),n	1101 0111	dddd dddd	0110 0011	C Z C H S V	7	(HL+d) ← (HL+d)-n
	nnnn nnnn					
SUB (HL+C),n	1110 0111	0110 0011	nnnn nnnn	C Z C H S V	7	(HL+C) ← (HL+C)-n
SUB (+SP),n	1110 0110	0110 0011	nnnn nnnn	C Z C H S V	6	SP ← SP+1;(SP) ← (SP)-n
SUB (PC+A),n	0100 1111	0110 0011	nnnn nnnn	C Z C H S V	7	(PC+A) ← (PC+A)-n
SUB rr,(x)	1110 0000	xxxx xxxx	10rr r011	C Z C U S V	6	rr ← rr-(x)
SUB rr,(vw)	1110 0001	wwww wwww	vvvv vvvv	C Z C U S V	7	rr ← rr-(vw)
	10rr r011					
SUB rr,(DE)	1110 0010	10rr r011		C Z C U S V	5	rr ← rr-(DE)
SUB rr,(HL)	1110 0011	10rr r011		C Z C U S V	5	rr ← rr-(HL)
SUB rr,(IX)	1110 0100	10rr r011		C Z C U S V	5	rr ← rr-(IX)
SUB rr,(IY)	1110 0101	10rr r011		C Z C U S V	5	rr ← rr-(IY)
SUB rr,(IX+d)	1101 0100	dddd dddd	10rr r011	C Z C U S V	7	rr ← rr-(IX+d)
SUB rr,(IY+d)	1101 0101	dddd dddd	10rr r011	C Z C U S V	7	rr ← rr-(IY+d)
SUB rr,(SP+d)	1101 0110	dddd dddd	10rr r011	C Z C U S V	7	rr ← rr-(SP+d)
SUB rr,(HL+d)	1101 0111	dddd dddd	10rr r011	C Z C U S V	7	rr ← rr-(HL+d)
SUB rr,(HL+C)	1110 0111	10rr r011		C Z C U S V	7	rr ← rr-(HL+C)
SUB rr,(+SP)	1110 0110	10rr r011		C Z C U S V	6	SP ← SP+1;rr ← rr-(SP)
SUB rr,(PC+A)	0100 1111	10rr r011		C Z C U S V	7	rr ← rr-(PC+A)

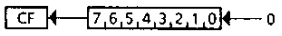
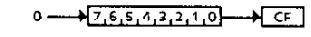
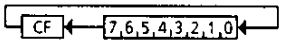
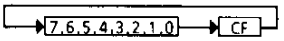
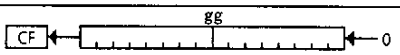
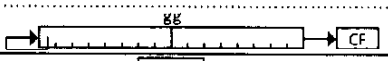
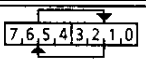
MNEMONIC	OBJECT CODE (Binary)			FLAG J Z C H S V	CYCLE	FUNCTION
SUBB A,n	0110 0010	nnnn nnnn		C Z C H S V	2	A ← A-n-CF
SUBB g,n	1110 1ggg	0110 0010	nnnn nnnn	C Z C H S V	3	g ← g-n-CF
SUBB gg,mn	1110 1ggg mmmm mmmm	0110 1010	nnnn nnnn	C Z C U S V	4	gg ← gg-mn-CF
SUBB r,g	1110 1ggg	00rr r010		C Z C H S V	2	r ← r-g-CF
SUBB rr,gg	1110 1ggg	10rr r010		C Z C U S V	4	rr ← rr-gg-CF
SUBB r,(x)	1110 0000	xxxx xxxx	00rr r010	C Z C H S V	4	r ← r-(x)-CF
SUBB r,(vw)	1110 0001 00rr r010	wwww wwww	vvvv vvvv	C Z C H S V	5	r ← r-(vw)-CF
SUBB r,(DE)	1110 0010	00rr r010		C Z C H S V	3	r ← r-(DE)-CF
SUBB r,(HL)	1110 0011	00rr r010		C Z C H S V	3	r ← r-(HL)-CF
SUBB r,(IX)	1110 0100	00rr r010		C Z C H S V	3	r ← r-(IX)-CF
SUBB r,(IY)	1110 0101	00rr r010		C Z C H S V	3	r ← r-(IY)-CF
SUBB r,(IX+d)	1101 0100	dddd dddd	00rr r010	C Z C H S V	5	r ← r-(IX+d)-CF
SUBB r,(IY+d)	1101 0101	dddd dddd	00rr r010	C Z C H S V	5	r ← r-(IY+d)-CF
SUBB r,(SP+d)	1101 0110	dddd dddd	00rr r010	C Z C H S V	5	r ← r-(SP+d)-CF
SUBB r,(HL+d)	1101 0111	dddd dddd	00rr r010	C Z C H S V	5	r ← r-(HL+d)-CF
SUBB r,(HL+C)	1110 0111	00rr r010		C Z C H S V	5	r ← r-(HL+C)-CF
SUBB r,(+SP)	1110 0110	00rr r010		C Z C H S V	4	SP ← SP+1;r ← r-(SP)-CF
SUBB r,(PC+A)	0100 1111	00rr r010		C Z C H S V	5	r ← r-(PC+A)-CF
SUBB (x),n	1110 0000 nnnn nnnn	xxxx xxxx	0110 0010	C Z C H S V	6	(x) ← (x)-n-CF
SUBB (vw),n	1110 0001 0110 0010	wwww wwww nnnn nnnn	vvvv vvvv	C Z C H S V	7	(vw) ← (vw)-n-CF
SUBB (DE),n	1110 0010	0110 0010	nnnn nnnn	C Z C H S V	5	(DE) ← (DE)-n-CF
SUBB (HL),n	1110 0011	0110 0010	nnnn nnnn	C Z C H S V	5	(HL) ← (HL)-n-CF
SUBB (IX),n	1110 0100	0110 0010	nnnn nnnn	C Z C H S V	5	(IX) ← (IX)-n-CF
SUBB (IY),n	1110 0101	0110 0010	nnnn nnnn	C Z C H S V	5	(IY) ← (IY)-n-CF
SUBB (IX+d),n	1101 0100 nnnn nnnn	dddd dddd	0110 0010	C Z C H S V	7	(IX+d) ← (IX+d)-n-CF
SUBB (IY+d),n	1101 0101 nnnn nnnn	dddd dddd	0110 0010	C Z C H S V	7	(IY+d) ← (IY+d)-n-CF
SUBB (SP+d),n	1101 0110 nnnn nnnn	dddd dddd	0110 0010	C Z C H S V	7	(SP+d) ← (SP+d)-n-CF
SUBB (HL+d),n	1101 0111 nnnn nnnn	dddd dddd	0110 0010	C Z C H S V	7	(HL+d) ← (HL+d)-n-CF
SUBB (HL+C),n	1110 0111	0110 0010	nnnn nnnn	C Z C H S V	7	(HL+C) ← (HL+C)-n-CF
SUBB (+SP),n	1110 0110	0110 0010	nnnn nnnn	C Z C H S V	6	SP ← SP+1:(SP) ← SP-n-CF
SUBB (PC+A),n	0100 1111	0110 0010	nnnn nnnn	C Z C H S V	7	(PC+A) ← (PC+A)-n-CF
SUBB rr,(x)	1110 0000	xxxx xxxx	10rr r010	C Z C U S V	6	rr ← rr-(x)-CF
SUBB rr,(vw)	1110 0001 10rr r010	wwww wwww	vvvv vvvv	C Z C U S V	7	rr ← rr-(vw)-CF
SUBB rr,(DE)	1110 0010	10rr r010		C Z C U S V	5	rr ← rr-(DE)-CF
SUBB rr,(HL)	1110 0011	10rr r010		C Z C U S V	5	rr ← rr-(HL)-CF
SUBB rr,(IX)	1110 0100	10rr r010		C Z C U S V	5	rr ← rr-(IX)-CF
SUBB rr,(IY)	1110 0101	10rr r010		C Z C U S V	5	rr ← rr-(IY)-CF
SUBB rr,(IX+d)	1101 0100	dddd dddd	10rr r010	C Z C U S V	7	rr ← rr-(IX+d)-CF
SUBB rr,(IY+d)	1101 0101	dddd dddd	10rr r010	C Z C U S V	7	rr ← rr-(IY+d)-CF
SUBB rr,(SP+d)	1101 0110	dddd dddd	10rr r010	C Z C U S V	7	rr ← rr-(SP+d)-CF
SUBB rr,(HL+d)	1101 0111	dddd dddd	10rr r010	C Z C U S V	7	rr ← rr-(HL+d)-CF
SUBB rr,(HL+C)	1110 0111	10rr r010		C Z C U S V	7	rr ← rr-(HL+C)-CF
SUBB rr,(+SP)	1110 0110	10rr r010		C Z C U S V	6	SP ← SP+1;rr ← rr-(SP)-CF
SUBB rr,(PC+A)	0100 1111	10rr r010		C Z C U S V	7	rr ← rr-(PC+A)-CF
AND A,n	0110 0100	nnnn nnnn		Z Z - - - -	2	A ← A∧n
AND g,n	1110 1ggg	0110 0100	nnnn nnnn	Z Z	3	g ← g∧n
AND gg,mn	1110 1ggg mmmm mmmm	0110 1100	nnnn nnnn	Z Z - - - -	4	gg ← gg∧mn
AND r,g	1110 1ggg	00rr r100		Z Z - - - -	2	r ← r∧g
AND rr,gg	1110 1ggg	10rr r100		Z Z - - - -	4	rr ← rr∧gg
AND r,(x)	1110 0000	xxxx xxxx	00rr r100	Z Z - - - -	4	r ← r∧(x)
AND r,(vw)	1110 0001 00rr r100	wwww wwww	vvvv vvvv	Z Z - - - -	5	r ← r∧(vw)
AND r,(DE)	1110 0010	00rr r100		Z Z - - - -	3	r ← r∧(DE)
AND r,(HL)	1110 0011	00rr r100		Z Z - - - -	3	r ← r∧(HL)
AND r,(IX)	1110 0100	00rr r100		Z Z - - - -	3	r ← r∧(IX)
AND r,(IY)	1110 0101	00rr r100		Z Z - - - -	3	r ← r∧(IY)
AND r,(IX+d)	1101 0100	dddd dddd	00rr r100	Z Z - - - -	5	r ← r∧(IX+d)
AND r,(IY+d)	1101 0101	dddd dddd	00rr r100	Z Z - - - -	5	r ← r∧(IY+d)
AND r,(SP+d)	1101 0110	dddd dddd	00rr r100	Z Z - - - -	5	r ← r∧(SP+d)
AND r,(HL+d)	1101 0111	dddd dddd	00rr r100	Z Z - - - -	5	r ← r∧(HL+d)
AND r,(HL+C)	1110 0111	00rr r100		Z Z - - - -	5	r ← r∧(HL+C)
AND r,(+SP)	1110 0110	00rr r100		Z Z - - - -	4	SP ← SP+1;r ← r∧(SP)
AND r,(PC+A)	0100 1111	00rr r100		Z Z - - - -	5	r ← r∧(PC+A)

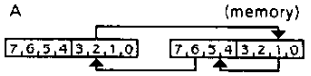
MNEMONIC	OBJECT CODE (Binary)			FLAG J Z C H S V	CYCLE	FUNCTION
AND (x),n	1110 0000 nnnn nnnn	xxxx xxxx	0110 0100	Z Z - - - -	6	(x) ← (x)∧n
AND (vw),n	1110 0001 wwww wwww 0110 0100	vvvv vvvv nnnn nnnn		Z Z - - - -	7	(vw) ← (vw)∧n
AND (DE),n	1110 0010	0110 0100	nnnn nnnn	Z Z - - - -	5	(DE) ← (DE)∧n
AND (HL),n	1110 0011	0110 0100	nnnn nnnn	Z Z - - - -	5	(HL) ← (HL)∧n
AND (IX),n	1110 0100	0110 0100	nnnn nnnn	Z Z - - - -	5	(IX) ← (IX)∧n
AND (IY),n	1110 0101	0110 0100	nnnn nnnn	Z Z - - - -	5	(IY) ← (IY)∧n
AND (IX+d),n	1101 0100 dddd dddd nnnn nnnn	0110 0100		Z Z - - - -	7	(IX+d) ← (IX+d)∧n
AND (IY+d),n	1101 0101 dddd dddd nnnn nnnn	0110 0100		Z Z - - - -	7	(IY+d) ← (IY+d)∧n
AND (SP+d),n	1101 0110 dddd dddd nnnn nnnn	0110 0100		Z Z - - - -	7	(SP+d) ← (SP+d)∧n
AND (HL+d),n	1101 0111 dddd dddd nnnn nnnn	0110 0100		Z Z - - - -	7	(HL+d) ← (HL+d)∧n
AND (HL+C),n	1110 0111	0110 0100	nnnn nnnn	Z Z - - - -	7	(HL+C) ← (HL+C)∧n
AND (+SP),n	1110 0110	0110 0100	nnnn nnnn	Z Z - - - -	6	SP ← SP+1:(SP) ← (SP)∧n
AND (PC+A),n	0100 1111	0110 0100	nnnn nnnn	Z Z - - - -	7	(PC+A) ← (PC+A)∧n
AND rr,(x)	1110 0000	xxxx xxxx	10rr r100	Z Z - - - -	6	rr ← rr∧(x)
AND rr,(vw)	1110 0001 wwww wwww 10rr r100	vvvv vvvv		Z Z - - - -	7	rr ← rr∧(vw)
AND rr,(DE)	1110 0010	10rr r100		Z Z - - - -	5	rr ← rr∧(DE)
AND rr,(HL)	1110 0011	10rr r100		Z Z - - - -	5	rr ← rr∧(HL)
AND rr,(IX)	1110 0100	10rr r100		Z Z - - - -	5	rr ← rr∧(IX)
AND rr,(IY)	1110 0101	10rr r100		Z Z - - - -	5	rr ← rr∧(IY)
AND rr,(IX+d)	1101 0100 dddd dddd	10rr r100		Z Z - - - -	7	rr ← rr∧(IX+d)
AND rr,(IY+d)	1101 0101 dddd dddd	10rr r100		Z Z - - - -	7	rr ← rr∧(IY+d)
AND rr,(SP+d)	1101 0110 dddd dddd	10rr r100		Z Z - - - -	7	rr ← rr∧(SP+d)
AND rr,(HL+d)	1101 0111 dddd dddd	10rr r100		Z Z - - - -	7	rr ← rr∧(HL+d)
AND rr,(HL+C)	1110 0111	10rr r100		Z Z - - - -	7	rr ← rr∧(HL+C)
AND rr,(+SP)	1110 0110	10rr r100		Z Z - - - -	6	SP ← SP+1:rr ← rr∧(SP)
AND rr,(PC+A)	0100 1111	10rr r100		Z Z - - - -	7	rr ← rr∧(PC+A)
OR A,n	0110 0110	nnnn nnnn		Z Z - - - -	2	A ← A∨n
OR g,n	1110 1ggg	0110 0110	nnnn nnnn	Z Z - - - -	3	g ← g∨n
OR gg,mn	1110 1ggg mmmm mmmm	0110 1110	nnnn nnnn	Z Z - - - -	4	gg ← gg∨mn
OR r,g	1110 1ggg	00rr r110		Z Z - - - -	2	r ← r∨g
OR rr,gg	1110 1ggg	10rr r110		Z Z - - - -	4	rr ← rr∨gg
OR r,(x)	1110 0000	xxxx xxxx	00rr r110	Z Z - - - -	4	r ← r∨(x)
OR r,(vw)	1110 0001 wwww wwww 00rr r110	vvvv vvvv		Z Z - - - -	5	r ← r∨(vw)
OR r,(DE)	1110 0010	00rr r110		Z Z - - - -	3	r ← r∨(DE)
OR r,(HL)	1110 0011	00rr r110		Z Z - - - -	3	r ← r∨(HL)
OR r,(IX)	1110 0100	00rr r110		Z Z - - - -	3	r ← r∨(IX)
OR r,(IY)	1110 0101	00rr r110		Z Z - - - -	3	r ← r∨(IY)
OR r,(IX+d)	1101 0100 dddd dddd	00rr r110		Z Z - - - -	5	r ← r∨(IX+d)
OR r,(IY+d)	1101 0101 dddd dddd	00rr r110		Z Z - - - -	5	r ← r∨(IY+d)
OR r,(SP+d)	1101 0110 dddd dddd	00rr r110		Z Z - - - -	5	r ← r∨(SP+d)
OR r,(HL+d)	1101 0111 dddd dddd	00rr r110		Z Z - - - -	5	r ← r∨(HL+d)
OR r,(HL+C)	1110 0111	00rr r110		Z Z - - - -	5	r ← r∨(HL+C)
OR r,(+SP)	1110 0110	00rr r110		Z Z - - - -	4	SP ← SP+1:r ← r∨(SP)
OR r,(PC+A)	0100 1111	00rr r110		Z Z - - - -	5	r ← r∨(PC+A)
OR (x),n	1110 0000 nnnn nnnn	xxxx xxxx	0110 0110	Z Z - - - -	6	(x) ← (x)∨n
OR (vw),n	1110 0001 wwww wwww 0110 0110	vvvv vvvv nnnn nnnn		Z Z - - - -	7	(vw) ← (vw)∨n
OR (DE),n	1110 0010	0110 0110	nnnn nnnn	Z Z - - - -	5	(DE) ← (DE)∨n
OR (HL),n	1110 0011	0110 0110	nnnn nnnn	Z Z - - - -	5	(HL) ← (HL)∨n
OR (IX),n	1110 0100	0110 0110	nnnn nnnn	Z Z - - - -	5	(IX) ← (IX)∨n
OR (IY),n	1110 0101	0110 0110	nnnn nnnn	Z Z - - - -	5	(IY) ← (IY)∨n
OR (IX+d),n	1101 0100 dddd dddd nnnn nnnn	0110 0110		Z Z - - - -	7	(IX+d) ← (IX+d)∨n
OR (IY+d),n	1101 0101 dddd dddd nnnn nnnn	0110 0110		Z Z - - - -	7	(IY+d) ← (IY+d)∨n
OR (SP+d),n	1101 0110 dddd dddd nnnn nnnn	0110 0110		Z Z - - - -	7	(SP+d) ← (SP+d)∨n
OR (HL+d),n	1101 0111 dddd dddd nnnn nnnn	0110 0110		Z Z - - - -	7	(HL+d) ← (HL+d)∨n
OR (HL+C),n	1110 0111	0110 0110	nnnn nnnn	Z Z - - - -	7	(HL+C) ← (HL+C)∨n
OR (+SP),n	1110 0110	0110 0110	nnnn nnnn	Z Z - - - -	6	SP ← SP+1:(SP) ← (SP)∨n
OR (PC+A),n	0100 1111	0110 0110	nnnn nnnn	Z Z - - - -	7	(PC+A) ← (PC+A)∨n

MNEMONIC	OBJECT CODE (Binary)			FLAG J Z C H S V	CYCLE	FUNCTION
OR rr,(x)	1110 0000	xxxx xxxx	10rr r110	Z Z - - - -	6	rr ← rrV(x)
OR rr,(vw)	1110 0001	wwww wwww	vvvv vvvv	Z Z - - - -	7	rr ← rrV(vw)
	10rr r110					
OR rr,(DE)	1110 0010	10rr r110		Z Z - - - -	5	rr ← rrV(DE)
OR rr,(HL)	1110 0011	10rr r110		Z Z - - - -	5	rr ← rrV(HL)
OR rr,(IX)	1110 0100	10rr r110		Z Z - - - -	5	rr ← rrV(IX)
OR rr,(IY)	1110 0101	10rr r110		Z Z - - - -	5	rr ← rrV(IY)
OR rr,(IX+d)	1101 0100	dddd dddd	10rr r110	Z Z - - - -	7	rr ← rrV(IX+d)
OR rr,(IY+d)	1101 0101	dddd dddd	10rr r110	Z Z - - - -	7	rr ← rrV(IY+d)
OR rr,(SP+d)	1101 0110	dddd dddd	10rr r110	Z Z - - - -	7	rr ← rrV(SP+d)
OR rr,(HL+d)	1101 0111	dddd dddd	10rr r110	Z Z - - - -	7	rr ← rrV(HL+d)
OR rr,(HL+C)	1110 0111	10rr r110		Z Z - - - -	7	rr ← rrV(HL+C)
OR rr,(+SP)	1110 0110	10rr r110		Z Z - - - -	6	SP ← SP+1:rr ← rrV(SP)
OR rr,(PC+A)	0100 1111	10rr r110		Z Z - - - -	7	rr ← rrV(PC+A)
XOR A,n	0110 0101	nnnn nnnn		Z Z - - - -	2	A ← A⊻n
XOR g,n	1110 1ggg	0110 0101	nnnn nnnn	Z Z - - - -	3	g ← g⊻nV
XOR gg,mn	1110 1ggg	0110 1101	nnnn nnnn	Z Z - - - -	4	gg ← gg⊻mn
	mmmm mmmm					
XOR r,g	1110 1ggg	00rr r101		Z Z - - - -	2	r ← r⊻g
XOR rr,gg	1110 1ggg	10rr r101		Z Z - - - -	4	rr ← rr⊻gg
XOR r,(x)	1110 0000	xxxx xxxx	00rr r101	Z Z - - - -	4	r ← r⊻(x)
XOR r,(vw)	1110 0001	wwww wwww	vvvv vvvv	Z Z - - - -	5	r ← r⊻(vw)
	00rr r101					
XOR r,(DE)	1110 0010	00rr r101		Z Z - - - -	3	r ← r⊻(DE)
XOR r,(HL)	1110 0011	00rr r101		Z Z - - - -	3	r ← r⊻(HL)
XOR r,(IX)	1110 0100	00rr r101		Z Z - - - -	3	r ← r⊻(IX)
XOR r,(IY)	1110 0101	00rr r101		Z Z - - - -	3	r ← r⊻(IY)
XOR r,(IX+d)	1101 0100	dddd dddd	00rr r101	Z Z - - - -	5	r ← r⊻(IX+d)
XOR r,(IY+d)	1101 0101	dddd dddd	00rr r101	Z Z - - - -	5	r ← r⊻(IY+d)
XOR r,(SP+d)	1101 0110	dddd dddd	00rr r101	Z Z - - - -	5	r ← r⊻(SP+d)
XOR r,(HL+d)	1101 0111	dddd dddd	00rr r101	Z Z - - - -	5	r ← r⊻(HL+d)
XOR r,(HL+C)	1110 0111	00rr r101		Z Z - - - -	5	r ← r⊻(HL+C)
XOR r,(+SP)	1110 0110	00rr r101		Z Z - - - -	4	SP ← SP+1:r ← r⊻(SP)
XOR r,(PC+A)	0100 1111	00rr r101		Z Z - - - -	5	r ← r⊻(PC+A)
XOR (x),n	1110 0000	xxxx xxxx	0110 0101	Z Z - - - -	6	(x) ← (x)⊻n
	nnnn nnnn					
XOR (vw),n	1110 0001	wwww wwww	vvvv vvvv	Z Z - - - -	7	(vw) ← (vw)⊻n
	0110 0101	nnnn nnnn				
XOR (DE),n	1110 0010	0110 0101	nnnn nnnn	Z Z - - - -	5	(DE) ← (DE)⊻n
XOR (HL),n	1110 0011	0110 0101	nnnn nnnn	Z Z - - - -	5	(HL) ← (HL)⊻n
XOR (IX),n	1110 0100	0110 0101	nnnn nnnn	Z Z - - - -	5	(IX) ← (IX)⊻n
XOR (IY),n	1110 0101	0110 0101	nnnn nnnn	Z Z - - - -	5	(IY) ← (IY)⊻n
XOR (IX+d),n	1101 0100	dddd dddd	0110 0101	Z Z - - - -	7	(IX+d) ← (IX+d)⊻n
	nnnn nnnn					
XOR (IY+d),n	1101 0101	dddd dddd	0110 0101	Z Z - - - -	7	(IY+d) ← (IY+d)⊻n
	nnnn nnnn					
XOR (SP+d),n	1101 0110	dddd dddd	0110 0101	Z Z - - - -	7	(SP+d) ← (SP+d)⊻n
	nnnn nnnn					
XOR (HL+d),n	1101 0111	dddd dddd	0110 0101	Z Z - - - -	7	(HL+d) ← (HL+d)⊻n
	nnnn nnnn					
XOR (HL+C),n	1110 0111	0110 0101	nnnn nnnn	Z Z - - - -	7	(HL+C) ← (HL+C)⊻n
XOR (+SP),n	1110 0110	0110 0101	nnnn nnnn	Z Z - - - -	6	SP ← SP+1:(SP) ← (SP)⊻n
XOR (PC+A),n	0100 1111	0110 0101	nnnn nnnn	Z Z - - - -	7	(PC+A) ← (PC+A)⊻n
XOR rr,(x)	1110 0000	xxxx xxxx	10rr r101	Z Z - - - -	6	rr ← rr⊻(x)
XOR rr,(vw)	1110 0001	wwww wwww	vvvv vvvv	Z Z - - - -	7	rr ← rr⊻(vw)
	10rr r101					
XOR rr,(DE)	1110 0010	10rr r101		Z Z - - - -	5	rr ← rr⊻(DE)
XOR rr,(HL)	1110 0011	10rr r101		Z Z - - - -	5	rr ← rr⊻(HL)
XOR rr,(IX)	1110 0100	10rr r101		Z Z - - - -	5	rr ← rr⊻(IX)
XOR rr,(IY)	1110 0101	10rr r101		Z Z - - - -	5	rr ← rr⊻(IY)
XOR rr,(IX+d)	1101 0100	dddd dddd	10rr r101	Z Z - - - -	7	rr ← rr⊻(IX+d)
XOR rr,(IY+d)	1101 0101	dddd dddd	10rr r101	Z Z - - - -	7	rr ← rr⊻(IY+d)
XOR rr,(SP+d)	1101 0110	dddd dddd	10rr r101	Z Z - - - -	7	rr ← rr⊻(SP+d)
XOR rr,(HL+d)	1101 0111	dddd dddd	10rr r101	Z Z - - - -	7	rr ← rr⊻(HL+d)
XOR rr,(HL+C)	1110 0111	10rr r101		Z Z - - - -	7	rr ← rr⊻(HL+C)
XOR rr,(+SP)	1110 0110	10rr r101		Z Z - - - -	6	SP ← SP+1:rr ← rr⊻(SP)
XOR rr,(PC+A)	0100 1111	10rr r101		Z Z - - - -	7	rr ← rr⊻(PC+A)

MNEMONIC	OBJECT CODE (Binary)			FLAG J Z C H S V	CYCLE	FUNCTION
INC r	0010 0rrr			C Z - - - -	1	r ← r+1
INC rr	0011 0rrr			C Z - - - -	2	rr ← rr+1
INC (x)	1110 0000	xxxx xxxx	1111 0000	C Z - - - -	5	(x) ← (x)+1
INC (vw)	1110 0001 wwww wwww vvvv vvvv 1111 0000			C Z - - - -	6	(vw) ← (vw)+1
INC (DE)	1110 0010	1111 0000		C Z - - - -	4	(DE) ← (DE)+1
INC (HL)	1110 0011	1111 0000		C Z - - - -	4	(HL) ← (HL)+1
INC (IX)	1110 0100	1111 0000		C Z - - - -	4	(IX) ← (IX)+1
INC (IY)	1110 0101	1111 0000		C Z - - - -	4	(IY) ← (IY)+1
INC (IX+d)	1101 0100	dddd dddd	1111 0000	C Z - - - -	6	(IX+d) ← (IX+d)+1
INC (IY+d)	1101 0101	dddd dddd	1111 0000	C Z - - - -	6	(IY+d) ← (IY+d)+1
INC (SP+d)	1101 0110	dddd dddd	1111 0000	C Z - - - -	6	(SP+d) ← (SP+d)+1
INC (HL+d)	1101 0111	dddd dddd	1111 0000	C Z - - - -	6	(HL+d) ← (HL+d)+1
INC (HL+C)	1110 0111	1111 0000		C Z - - - -	6	(HL+C) ← (HL+C)+1
INC (+SP)	1110 0110	1111 0000		C Z - - - -	5	SP ← SP+1:(SP) ← (SP)+1
INC (PC+A)	0100 1111	1111 0000		C Z - - - -	6	(PC+A) ← (PC+A)+1
DEC r	0010 1rrr			C Z - - - -	1	r ← r-1
DEC rr	0011 1rrr			C Z - - - -	2	rr ← rr-1
DEC (x)	1110 0000	xxxx xxxx	1111 1000	C Z - - - -	5	(x) ← (x)-1
DEC (vw)	1110 0001 wwww wwww vvvv vvvv 1111 1000			C Z - - - -	6	(vw) ← (vw)-1
DEC (DE)	1110 0010	1111 1000		C Z - - - -	4	(DE) ← (DE)-1
DEC (HL)	1110 0011	1111 1000		C Z - - - -	4	(HL) ← (HL)-1
DEC (IX)	1110 0100	1111 1000		C Z - - - -	4	(IX) ← (IX)-1
DEC (IY)	1110 0101	1111 1000		C Z - - - -	4	(IY) ← (IY)-1
DEC (IX+d)	1101 0100	dddd dddd	1111 1000	C Z - - - -	6	(IX+d) ← (IX+d)-1
DEC (IY+d)	1101 0101	dddd dddd	1111 1000	C Z - - - -	6	(IY+d) ← (IY+d)-1
DEC (SP+d)	1101 0110	dddd dddd	1111 1000	C Z - - - -	6	(SP+d) ← (SP+d)-1
DEC (HL+d)	1101 0111	dddd dddd	1111 1000	C Z - - - -	6	(HL+d) ← (HL+d)-1
DEC (HL+C)	1110 0111	1111 1000		C Z - - - -	6	(HL+C) ← (HL+C)-1
DEC (+SP)	1110 0110	1111 1000		C Z - - - -	5	SP ← SP+1:(SP) ← (SP)-1
DEC (PC+A)	0100 1111	1111 1000		C Z - - - -	6	(PC+A) ← (PC+A)-1
DAA g	1110 1ggg	1101 1010		C Z C H - -	2	Decimal adjustment against g (after addition)
DAS g	1110 1ggg	1101 1011		C Z C H - -	2	Decimal adjustment against g (after subtraction)
MUL W,A	1110 1000	1111 0010		Z Z - - - -	8	WA ← W×A
MUL B,C	1110 1001	1111 0010		Z Z - - - -	8	BC ← B×C
MUL D,E	1110 1010	1111 0010		Z Z - - - -	8	DE ← D×E
MUL H,L	1110 1011	1111 0010		Z Z - - - -	8	HL ← H×L
DIV WA,C	1110 1000	1111 0011		Z Z C - - -	8	A ← WA÷C,W ← residue
DIV DE,C	1110 1010	1111 0011		Z Z C - - -	8	E ← DE÷C,D ← residue
DIV HL,C	1110 1011	1111 0011		Z Z C - - -	8	L ← HL÷C,H ← residue
NEG CS,gg	1110 1ggg	1111 1010		1 - - - - -	3	if CF=1 then gg ← 0-gg else null

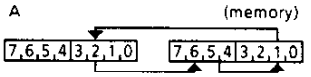
(3) Shift, rotate and nibble manipulation

MNEMONIC	OBJECT CODE (Binary)			FLAG J Z C H S V	CYCLE	FUNCTION
SHLC g	1110 1ggg	1111 0100		C Z * - - -	2	
SHRC g	1110 1ggg	1111 0101		C Z * - - -	2	
ROLC g	1110 1ggg	1111 0110		C Z * - - -	2	
RORC g	1110 1ggg	1111 0111		C Z * - - -	2	
SHLCA gg	1110 1ggg	1111 0000		C Z * - S V	3	
SHRCA gg	1110 1ggg	1111 0001		C Z * - S 0	3	
SWAP g	1110 1ggg	1111 1111		1 - - - - -	4	 (Exchange upper 4 bits for lower 4 bits)
ROLD A,(x)	1110 0000	xxxx xxxx	1111 0110	1 - - - - -	8	
ROLD A,(vw)	1110 0001	wwww wwww	vvvv vvvv	1 - - - - -	9	
	1111 0110					
ROLD A,(DE)	1110 0010	1111 0110		1 - - - - -	7	
ROLD A,(HL)	1110 0011	1111 0110		1 - - - - -	7	
ROLD A,(IX)	1110 0100	1111 0110		1	7	
ROLD A,(IY)	1110 0101	1111 0110		1 - - - - -	7	
ROLD A,(IX+d)	1101 0100	dddd dddd	1111 0110	1 - - - - -	9	
ROLD A,(IY+d)	1101 0101	dddd dddd	1111 0110	1 - - - - -	9	
ROLD A,(SP+d)	1101 0110	dddd dddd	1111 0110	1 - - - - -	9	
ROLD A,(HL+d)	1101 0111	dddd dddd	1111 0110	1 - - - - -	9	
ROLD A,(HL+C)	1110 0111	1111 0110		1 - - - - -	9	
ROLD A,(+SP)	1110 0110	1111 0110		1 - - - - -	8	
ROLD A,(PC+A)	0100 1111	1111 0110		1 - - - - -	9	
RORD A,(x)	1110 0000	xxxx xxxx	1111 0111	1 - - - - -	8	
RORD A,(vw)	1110 0001	wwww wwww	vvvv vvvv	1 - - - - -	9	
	1111 0111					
RORD A,(DE)	1110 0010	1111 0111		1 - - - - -	7	
RORD A,(HL)	1110 0011	1111 0111		1 - - - - -	7	
RORD A,(IX)	1110 0100	1111 0111		1 - - - - -	7	
RORD A,(IY)	1110 0101	1111 0111		1 - - - - -	7	
RORD A,(IX+d)	1101 0100	dddd dddd	1111 0111	1 - - - - -	9	
RORD A,(IY+d)	1101 0101	dddd dddd	1111 0111	1 - - - - -	9	
RORD A,(SP+d)	1101 0110	dddd dddd	1111 0111	1 - - - - -	9	
RORD A,(HL+d)	1101 0111	dddd dddd	1111 0111	1 - - - - -	9	
RORD A,(HL+C)	1110 0111	1111 0111		1 - - - - -	9	
RORD A,(+SP)	1110 0110	1111 0111		1 - - - - -	8	
RORD A,(PC+A)	0100 1111	1111 0111		1 - - - - -	9	



(Rotate left by 4 bits)

Note : (+SP) indicates that SP←SP + 1 is executed before rotation



(Rotate right by 4 bits)

Note : (+SP) indicates that SP←SP + 1 is executed before rotation

(4) Bit manipulation and flag manipulation

MNEMONIC	OBJECT CODE (Binary)			FLAG J Z C H S V	CYCLE	FUNCTION
SET g.b	1110 1ggg	1100 0bbb		Z * - - - -	3	ZF ← $\overline{g.b}$:g.b ← 1
SET (x).b	1100 0bbb	xxxx xxxx		Z * - - - -	5	ZF ← $\overline{(x).b}$:(x).b ← 1
SET (vw).b	1110 0001 1100 0bbb	wwww wwww	vvvv vvvv	Z * - - - -	6	ZF ← $\overline{(vw).b}$:(vw).b ← 1
SET (DE).b	1110 0010	1100 0bbb		Z * - - - -	4	ZF ← $\overline{(DE).b}$:(DE).b ← 1
SET (HL).b	1110 0011	1100 0bbb		Z * - - - -	4	ZF ← $\overline{(HL).b}$:(HL).b ← 1
SET (IX).b	1110 0100	1100 0bbb		Z * - - - -	4	ZF ← $\overline{(IX).b}$:(IX).b ← 1
SET (IY).b	1110 0101	1100 0bbb		Z * - - - -	4	ZF ← $\overline{(IY).b}$:(IY).b ← 1
SET (IX+d).b	1101 0100	dddd dddd	1100 0bbb	Z * - - - -	6	ZF ← $\overline{(IX+d).b}$:(IX+d).b ← 1
SET (IY+d).b	1101 0101	dddd dddd	1100 0bbb	Z * - - - -	6	ZF ← $\overline{(IY+d).b}$:(IY+d).b ← 1
SET (SP+d).b	1101 0110	dddd dddd	1100 0bbb	Z * - - - -	6	ZF ← $\overline{(SP+d).b}$:(SP+d).b ← 1
SET (HL+d).b	1101 0111	dddd dddd	1100 0bbb	Z * - - - -	6	ZF ← $\overline{(HL+d).b}$:(HL+d).b ← 1
SET (HL+C).b	1110 0111	1100 0bbb		Z * - - - -	6	ZF ← $\overline{(HL+C).b}$:(HL+C).b ← 1
SET (+SP).b	1110 0110	1100 0bbb		Z * - - - -	5	SP ← SP+1:ZF ← $\overline{(SP).b}$:(SP).b ← 1
SET (PC+A).b	0100 1111	1100 0bbb		Z * - - - -	6	ZF ← $\overline{(PC+A).b}$:(PC+A).b ← 1
SET (x).A	1110 0000	xxxx xxxx	1111 0010	Z * - - - -	5	ZF ← $\overline{(x).A}$:(x).A ← 1
SET (vw).A	1110 0001 1111 0010	wwww wwww	vvvv vvvv	Z * - - - -	6	ZF ← $\overline{(vw).A}$:(vw).A ← 1
SET (DE).A	1110 0010	1111 0010		Z * - - - -	4	ZF ← $\overline{(DE).A}$:(DE).A ← 1
SET (HL).A	1110 0011	1111 0010		Z * - - - -	4	ZF ← $\overline{(HL).A}$:(HL).A ← 1
SET (IX).A	1110 0100	1111 0010		Z * - - - -	4	ZF ← $\overline{(IX).A}$:(IX).A ← 1
SET (IY).A	1110 0101	1111 0010		Z * - - - -	4	ZF ← $\overline{(IY).A}$:(IY).A ← 1
SET (IX+d).A	1101 0100	dddd dddd	1111 0010	Z * - - - -	6	ZF ← $\overline{(IX+d).A}$:(IX+d).A ← 1
SET (IY+d).A	1101 0101	dddd dddd	1111 0010	Z * - - - -	6	ZF ← $\overline{(IY+d).A}$:(IY+d).A ← 1
SET (SP+d).A	1101 0110	dddd dddd	1111 0010	Z * - - - -	6	ZF ← $\overline{(SP+d).A}$:(SP+d).A ← 1
SET (HL+d).A	1101 0111	dddd dddd	1111 0010	Z * - - - -	6	ZF ← $\overline{(HL+d).A}$:(HL+d).A ← 1
SET (HL+C).A	1110 0111	1111 0010		Z * - - - -	6	ZF ← $\overline{(HL+C).A}$:(HL+C).A ← 1
SET (+SP).A	1110 0110	1111 0010		Z * - - - -	5	SP ← SP+1:ZF ← $\overline{(SP).A}$:(SP).A ← 1
SET (PC+A).A	0100 1111	1111 0010		Z * - - - -	6	ZF ← $\overline{(PC+A).A}$:(PC+A).A ← 1
CLR g.b	1110 1ggg	1100 1bbb		Z * - - - -	3	ZF ← $\overline{g.b}$:g.b ← 0
CLR (x).b	1100 1bbb	xxxx xxxx		Z * - - - -	5	ZF ← $\overline{(x).b}$:(x).b ← 0
CLR (vw).b	1110 0001 1100 1bbb	wwww wwww	vvvv vvvv	Z * - - - -	6	ZF ← $\overline{(vw).b}$:(vw).b ← 0
CLR (DE).b	1110 0010	1100 1bbb		Z * - - - -	4	ZF ← $\overline{(DE).b}$:(DE).b ← 0
CLR (HL).b	1110 0011	1100 1bbb		Z * - - - -	4	ZF ← $\overline{(HL).b}$:(HL).b ← 0
CLR (IX).b	1110 0100	1100 1bbb		Z * - - - -	4	ZF ← $\overline{(IX).b}$:(IX).b ← 0
CLR (IY).b	1110 0101	1100 1bbb		Z * - - - -	4	ZF ← $\overline{(IY).b}$:(IY).b ← 0
CLR (IX+d).b	1101 0100	dddd dddd	1100 1bbb	Z * - - - -	6	ZF ← $\overline{(IX+d).b}$:(IX+d).b ← 0
CLR (IY+d).b	1101 0101	dddd dddd	1100 1bbb	Z * - - - -	6	ZF ← $\overline{(IY+d).b}$:(IY+d).b ← 0
CLR (SP+d).b	1101 0110	dddd dddd	1100 1bbb	Z * - - - -	6	ZF ← $\overline{(SP+d).b}$:(SP+d).b ← 0
CLR (HL+d).b	1101 0111	dddd dddd	1100 1bbb	Z * - - - -	6	ZF ← $\overline{(HL+d).b}$:(HL+d).b ← 0
CLR (HL+C).b	1110 0111	1100 1bbb		Z * - - - -	6	ZF ← $\overline{(HL+C).b}$:(HL+C).b ← 0
CLR (+SP).b	1110 0110	1100 1bbb		Z * - - - -	5	SP ← SP+1:ZF ← $\overline{(SP).b}$:(SP).b ← 0
CLR (PC+A).b	0100 1111	1100 1bbb		Z * - - - -	6	ZF ← $\overline{(PC+A).b}$:(PC+A).b ← 0
CLR (x).A	1110 0000	xxxx xxxx	1111 1010	Z * - - - -	5	ZF ← $\overline{(x).A}$:(x).A ← 0
CLR (vw).A	1110 0001 1111 1010	wwww wwww	vvvv vvvv	Z * - - - -	6	ZF ← $\overline{(vw).A}$:(vw).A ← 0
CLR (DE).A	1110 0010	1111 1010		Z * - - - -	4	ZF ← $\overline{(DE).A}$:(DE).A ← 0
CLR (HL).A	1110 0011	1111 1010		Z * - - - -	4	ZF ← $\overline{(HL).A}$:(HL).A ← 0
CLR (IX).A	1110 0100	1111 1010		Z * - - - -	4	ZF ← $\overline{(IX).A}$:(IX).A ← 0
CLR (IY).A	1110 0101	1111 1010		Z * - - - -	4	ZF ← $\overline{(IY).A}$:(IY).A ← 0
CLR (IX+d).A	1101 0100	dddd dddd	1111 1010	Z * - - - -	6	ZF ← $\overline{(IX+d).A}$:(IX+d).A ← 0
CLR (IY+d).A	1101 0101	dddd dddd	1111 1010	Z * - - - -	6	ZF ← $\overline{(IY+d).A}$:(IY+d).A ← 0
CLR (SP+d).A	1101 0110	dddd dddd	1111 1010	Z * - - - -	6	ZF ← $\overline{(SP+d).A}$:(SP+d).A ← 0
CLR (HL+d).A	1101 0111	dddd dddd	1111 1010	Z * - - - -	6	ZF ← $\overline{(HL+d).A}$:(HL+d).A ← 0
CLR (HL+C).A	1110 0111	1111 1010		Z * - - - -	6	ZF ← $\overline{(HL+C).A}$:(HL+C).A ← 0
CLR (+SP).A	1110 0110	1111 1010		Z * - - - -	5	SP ← SP+1:ZF ← $\overline{(SP).A}$:(SP).A ← 0
CLR (PC+A).A	0100 1111	1111 1010		Z * - - - -	6	ZF ← $\overline{(PC+A).A}$:(PC+A).A ← 0
LD CF,g.b	1110 1ggg	0101 1bbb		C - *	2	CF ← g.b
LD CF,(x).b	0101 1bbb	xxxx xxxx		C - * - - -	4	CF ← (x).b
LD CF,(vw).b	1110 0001 0101 1bbb	wwww wwww	vvvv vvvv	C - * - - -	5	CF ← (vw).b
LD CF,(DE).b	1110 0010	0101 1bbb		C - * - - -	3	CF ← (DE).b
LD CF,(HL).b	1110 0011	0101 1bbb		C - * - - -	3	CF ← (HL).b
LD CF,(IX).b	1110 0100	0101 1bbb		C - * - - -	3	CF ← (IX).b
LD CF,(IY).b	1110 0101	0101 1bbb		C - * - - -	3	CF ← (IY).b
LD CF,(IX+d).b	1101 0100	dddd dddd	0101 1bbb	C - * - - -	5	CF ← (:IX+d).b
LD CF,(IY+d).b	1101 0101	dddd dddd	0101 1bbb	C - * - - -	5	CF ← (IY+d).b
LD CF,(SP+d).b	1101 0110	dddd dddd	0101 1bbb	C - *	5	CF ← (SP+d).b
LD CF,(HL+d).b	1101 0111	dddd dddd	0101 1bbb	C - * - - -	5	CF ← (HL+d).b
LD CF,(HL+C).b	1110 0111	0101 1bbb		C - * - - -	5	CF ← (HL+C).b
LD CF,(+SP).b	1110 0110	0101 1bbb		C - * - - -	4	SP ← SP+1:CF ← (SP).b
LD CF,(PC+A).b	0100 1111	0101 1bbb		C - * - - -	5	CF ← (PC+A).b

	MNEMONIC	OBJECT CODE (Binary)			FLAG J Z C H S V	CYCLE	FUNCTION
	LD CF,(x).A	1110 0000	xxxx xxxx	1111 1100	$\overline{C}$ - * - - -	4	CF ← (x).A
	LD CF,(vw).A	1110 0001	wwww wwww	vvvv vvvv	$\overline{C}$ - * - - -	5	CF ← (vw).A
		1111 1100					
	LD CF,(DE).A	1110 0010	1111 1100		$\overline{C}$ - * - - -	3	CF ← (DE).A
	LD CF,(HL).A	1110 0011	1111 1100		$\overline{C}$ - * - - -	3	CF ← (HL).A
	LD CF,(IX).A	1110 0100	1111 1100		$\overline{C}$ - * - - -	3	CF ← (IX).A
	LD CF,(IY).A	1110 0101	1111 1100		$\overline{C}$ - * - - -	3	CF ← (IY).A
	LD CF,(IX+d).A	1101 0100	dddd dddd	1111 1100	$\overline{C}$ - * - - -	5	CF ← (IX+d).A
	LD CF,(IY+d).A	1101 0101	dddd dddd	1111 1100	$\overline{C}$ - * - - -	5	CF ← (IY+d).A
	LD CF,(SP+d).A	1101 0110	dddd dddd	1111 1100	$\overline{C}$ - * - - -	5	CF ← (SP+d).A
	LD CF,(HL+d).A	1101 0111	dddd dddd	1111 1100	$\overline{C}$ - * - - -	5	CF ← (HL+d).A
	LD CF,(HL+C).A	1110 0111	1111 1100		$\overline{C}$ - * - - -	5	CF ← (HL+C).A
	LD CF,(+SP).A	1110 0110	1111 1100		$\overline{C}$ - * - - -	4	SP ← SP+1:CF ← (SP).A
	LD CF,(PC+A).A	0100 1111	1111 1100		$\overline{C}$ - * - - -	5	CF ← (PC+A).A
*2	TEST g.b	1110 1ggg	0101 1bbb		* - $\overline{J}$ - - -	2	JF ← $\overline{g.b}$
*2	TEST (x).b	0101 1bbb	xxxx xxxx		* - $\overline{J}$ - - -	4	JF ← $\overline{(x).b}$
*2	TEST (vw).b	1110 0001	wwww wwww	vvvv vvvv	* - $\overline{J}$ - - -	5	JF ← $\overline{(vw).b}$
		0101 1bbb					
*2	TEST (DE).b	1110 0010	0101 1bbb		* - $\overline{J}$ - - -	3	JF ← $\overline{(DE).b}$
*2	TEST (HL).b	1110 0011	0101 1bbb		* - $\overline{J}$ - - -	3	JF ← $\overline{(HL).b}$
*2	TEST (IX).b	1110 0100	0101 1bbb		* - $\overline{J}$ - - -	3	JF ← $\overline{(IX).b}$
*2	TEST (IY).b	1110 0101	0101 1bbb		* - $\overline{J}$ - - -	3	JF ← $\overline{(IY).b}$
*2	TEST (IX+d).b	1101 0100	dddd dddd	0101 1bbb	* - $\overline{J}$ - - -	5	JF ← $\overline{(IX+d).b}$
*2	TEST (IY+d).b	1101 0101	dddd dddd	0101 1bbb	* - $\overline{J}$ - - -	5	JF ← $\overline{(IY+d).b}$
*2	TEST (SP+d).b	1101 0110	dddd dddd	0101 1bbb	* - $\overline{J}$ - - -	5	JF ← $\overline{(SP+d).b}$
*2	TEST (HL+d).b	1101 0111	dddd dddd	0101 1bbb	* - $\overline{J}$ - - -	5	JF ← $\overline{(HL+d).b}$
*2	TEST (HL+C).b	1110 0111	0101 1bbb		* - $\overline{J}$ - - -	5	JF ← $\overline{(HL+C).b}$
*2	TEST (+SP).b	1110 0110	0101 1bbb		* - $\overline{J}$ - - -	4	SP ← SP+1:JF ← $\overline{(SP).b}$
*2	TEST (PC+A).b	0100 1111	0101 1bbb		* - $\overline{J}$ - - -	5	JF ← $\overline{(PC+A).b}$
*2	TEST (x).A	1110 0000	xxxx xxxx	1111 1100	* - $\overline{J}$ - - -	4	JF ← $\overline{(x).A}$
*2	TEST (vw).A	1110 0001	wwww wwww	vvvv vvvv	* - $\overline{J}$ - - -	5	JF ← $\overline{(vw).A}$
		1111 1100					
*2	TEST (DE).A	1110 0010	1111 1100		* - $\overline{J}$ - - -	3	JF ← $\overline{(DE).A}$
*2	TEST (HL).A	1110 0011	1111 1100		* - $\overline{J}$ - - -	3	JF ← $\overline{(HL).A}$
*2	TEST (IX).A	1110 0100	1111 1100		* - $\overline{J}$ - - -	3	JF ← $\overline{(IX).A}$
*2	TEST (IY).A	1110 0101	1111 1100		* - $\overline{J}$ - - -	3	JF ← $\overline{(IY).A}$
*2	TEST (IX+d).A	1101 0100	dddd dddd	1111 1100	* - $\overline{J}$ - - -	5	JF ← $\overline{(IX+d).A}$
*2	TEST (IY+d).A	1101 0101	dddd dddd	1111 1100	* - $\overline{J}$ - - -	5	JF ← $\overline{(IY+d).A}$
*2	TEST (SP+d).A	1101 0110	dddd dddd	1111 1100	* - $\overline{J}$ - - -	5	JF ← $\overline{(SP+d).A}$
*2	TEST (HL+d).A	1101 0111	dddd dddd	1111 1100	* - $\overline{J}$ - - -	5	JF ← $\overline{(HL+d).A}$
*2	TEST (HL+C).A	1110 0111	1111 1100		* - $\overline{J}$ - - -	5	JF ← $\overline{(HL+C).A}$
*2	TEST (+SP).A	1110 0110	1111 1100		* - $\overline{J}$ - - -	4	SP ← SP+1:JF ← $\overline{(SP).A}$
*2	TEST (PC+A).A	0100 1111	1111 1100		* - $\overline{J}$ - - -	5	JF ← $\overline{(PC+A).A}$
	LD g.b,CF	1110 1ggg	1110 1bbb		1 - - - - -	2	g.b ← CF
	LD (x).b,CF	1110 0000	xxxx xxxx	1110 1bbb	1 - - - - -	5	(x).b ← CF
	LD (vw).b,CF	1110 0001	wwww wwww	vvvv vvvv	1 - - - - -	6	(vw).b ← CF
		1110 1bbb					
	LD (DE).b,CF	1110 0010	1110 1bbb		1 - - - - -	4	(DE).b ← CF
	LD (HL).b,CF	1110 0011	1110 1bbb		1 - - - - -	4	(HL).b ← CF
	LD (IX).b,CF	1110 0100	1110 1bbb		1 - - - - -	4	(IX).b ← CF
	LD (IY).b,CF	1110 0101	1110 1bbb		1 - - - - -	4	(IY).b ← CF
	LD (IX+d).b,CF	1101 0100	dddd dddd	1110 1bbb	1 - - - - -	6	(IX+d).b ← CF
	LD (IY+d).b,CF	1101 0101	dddd dddd	1110 1bbb	1 - - - - -	6	(IY+d).b ← CF
	LD (SP+d).b,CF	1101 0110	dddd dddd	1110 1bbb	1 - - - - -	6	(SP+d).b ← CF
	LD (HL+d).b,CF	1101 0111	dddd dddd	1110 1bbb	1 - - - - -	6	(HL+d).b ← CF
	LD (HL+C).b,CF	1110 0111	1110 1bbb		1 - - - - -	6	(HL+C).b ← CF
	LD (+SP).b,CF	1110 0110	1110 1bbb		1 - - - - -	5	SP ← SP+1:(SP).b ← CF
	LD (PC+A).b,CF	0100 1111	1110 1bbb		1 - - - - -	6	(PC+A).b ← CF
	LD (x).A,CF	1110 0000	xxxx xxxx	1111 0011	1 - - - - -	5	(x).A ← CF
	LD (vw).A,CF	1110 0001	wwww wwww	vvvv vvvv	1 - - - - -	6	(vw).A ← CF
		1111 0011					
	LD (DE).A,CF	1110 0010	1111 0011		1 - - - - -	4	(DE).A ← CF
	LD (HL).A,CF	1110 0011	1111 0011		1 - - - - -	4	(HL).A ← CF
	LD (IX).A,CF	1110 0100	1111 0011		1 - - - - -	4	(IX).A ← CF
	LD (IY).A,CF	1110 0101	1111 0011		1 - - - - -	4	(IY).A ← CF
	LD (IX+d).A,CF	1101 0100	dddd dddd	1111 0011	1 - - - - -	6	(IX+d).A ← CF
	LD (IY+d).A,CF	1101 0101	dddd dddd	1111 0011	1 - - - - -	6	(IY+d).A ← CF
	LD (SP+d).A,CF	1101 0110	dddd dddd	1111 0011	1 - - - - -	6	(SP+d).A ← CF
	LD (HL+d).A,CF	1101 0111	dddd dddd	1111 0011	1 - - - - -	6	(HL+d).A ← CF
	LD (HL+C).A,CF	1110 0111	1111 0011		1 - - - - -	6	(HL+C).A ← CF
	LD (+SP).A,CF	1110 0110	1111 0011		1 - - - - -	5	SP ← SP+1:(SP).A ← CF
	LD (PC+A).A,CF	0100 1111	1111 0011		1 - - - - -	6	(PC+A).A ← CF

MNEMONIC	OBJECT CODE (Binary)			FLAG J Z C H S V	CYCLE	FUNCTION
CPL g.b	1110 1ggg	1110 0bbb		Z * - - - -	3	ZF ← $\overline{g.b}$:g.b ← $\overline{g.b}$
CPL (x).b	1110 0000	xxxx xxxx	1110 0bbb	Z * - - - -	5	ZF ← $\overline{(x).b}$:(x).b ← $\overline{(x).b}$
CPL (vw).b	1110 0001	wwww wwww	vvvv vvvv	Z * - - - -	6	ZF ← $\overline{(vw).b}$:(vw).b ← $\overline{(vw).b}$
	1110 0bbb					
CPL (DE).b	1110 0010	1110 0bbb		Z * - - - -	4	ZF ← $\overline{(DE).b}$:(DE).b ← $\overline{(DE).b}$
CPL (HL).b	1110 0011	1110 0bbb		Z * - - - -	4	ZF ← $\overline{(HL).b}$:(HL).b ← $\overline{(HL).b}$
CPL (IX).b	1110 0100	1110 0bbb		Z * - - - -	4	ZF ← $\overline{(IX).b}$:(IX).b ← $\overline{(IX).b}$
CPL (IY).b	1110 0101	1110 0bbb		Z * - - - -	4	ZF ← $\overline{(IY).b}$:(IY).b ← $\overline{(IY).b}$
CPL (IX+d).b	1101 0100	dddd dddd	1110 0bbb	Z * - - - -	6	ZF ← $\overline{(IX+d).b}$:(IX+d).b ← $\overline{(IX+d).b}$
CPL (IY+d).b	1101 0101	dddd dddd	1110 0bbb	Z * - - - -	6	ZF ← $\overline{(IY+d).b}$:(IY+d).b ← $\overline{(IY+d).b}$
CPL (SP+d).b	1101 0110	dddd dddd	1110 0bbb	Z * - - - -	6	ZF ← $\overline{(SP+d).b}$:(SP+d).b ← $\overline{(SP+d).b}$
CPL (HL+d).b	1101 0111	dddd dddd	1110 0bbb	Z * - - - -	6	ZF ← $\overline{(HL+d).b}$:(HL+d).b ← $\overline{(HL+d).b}$
CPL (HL+C).b	1110 0111	1110 0bbb		Z * - - - -	6	ZF ← $\overline{(HL+C).b}$:(HL+C).b ← $\overline{(HL+C).b}$
CPL (+SP).b	1110 0110	1110 0bbb		Z * - - - -	5	SP ← SP+1:ZF ← $\overline{(SP).b}$:(SP).b ← $\overline{(SP).b}$
CPL (PC+A).b	0100 1111	1110 0bbb		Z * - - - -	6	ZF ← $\overline{(PC+A).b}$:(PC+A).b ← $\overline{(PC+A).b}$
CPL (x).A	1110 0000	xxxx xxxx	1111 1011	Z * - - - -	5	ZF ← $\overline{(x).A}$:(x).A ← $\overline{(x).A}$
CPL (vw).A	1110 0001	wwww wwww	vvvv vvvv	Z * - - - -	6	ZF ← $\overline{(vw).A}$:(vw).A ← $\overline{(vw).A}$
	1111 1011					
CPL (DE).A	1110 0010	1111 1011		Z * - - - -	4	ZF ← $\overline{(DE).A}$:(DE).A ← $\overline{(DE).A}$
CPL (HL).A	1110 0011	1111 1011		Z * - - - -	4	ZF ← $\overline{(HL).A}$:(HL).A ← $\overline{(HL).A}$
CPL (IX).A	1110 0100	1111 1011		Z * - - - -	4	ZF ← $\overline{(IX).A}$:(IX).A ← $\overline{(IX).A}$
CPL (IY).A	1110 0101	1111 1011		Z * - - - -	4	ZF ← $\overline{(IY).A}$:(IY).A ← $\overline{(IY).A}$
CPL (IX+d).A	1101 0100	dddd dddd	1111 1011	Z * - - - -	6	ZF ← $\overline{(IX+d).A}$:(IX+d).A ← $\overline{(IX+d).A}$
CPL (IY+d).A	1101 0101	dddd dddd	1111 1011	Z * - - - -	6	ZF ← $\overline{(IY+d).A}$:(IY+d).A ← $\overline{(IY+d).A}$
CPL (SP+d).A	1101 0110	dddd dddd	1111 1011	Z * - - - -	6	ZF ← $\overline{(SP+d).A}$:(SP+d).A ← $\overline{(SP+d).A}$
CPL (HL+d).A	1101 0111	dddd dddd	1111 1011	Z * - - - -	6	ZF ← $\overline{(HL+d).A}$:(HL+d).A ← $\overline{(HL+d).A}$
CPL (HL+C).A	1110 0111	1111 1011		Z * - - - -	6	ZF ← $\overline{(HL+C).A}$:(HL+C).A ← $\overline{(HL+C).A}$
CPL (+SP).A	1110 0110	1111 1011		Z * - - - -	5	SP ← SP+1:ZF ← $\overline{(SP).A}$:(SP).A ← $\overline{(SP).A}$
CPL (PC+A).A	0100 1111	1111 1011		Z * - - - -	6	ZF ← $\overline{(PC+A).A}$:(PC+A).A ← $\overline{(PC+A).A}$
XOR CF,g.b	1110 1ggg	0101 0bbb		$\overline{C}$ - * - - -	2	CF ← CF ⊻ g.b
XOR CF,(x).b	1110 0000	xxxx xxxx	0101 0bbb	$\overline{C}$ - * - - -	4	CF ← CF ⊻ (x).b
XOR CF,(vw).b	1110 0001	wwww wwww	vvvv vvvv	$\overline{C}$ - * - - -	5	CF ← CF ⊻ (vw).b
	0101 0bbb					
XOR CF,(DE).b	1110 0010	0101 0bbb		$\overline{C}$ - * - - -	3	CF ← CF ⊻ (DE).b
XOR CF,(HL).b	1110 0011	0101 0bbb		$\overline{C}$ - * - - -	3	CF ← CF ⊻ (HL).b
XOR CF,(IX).b	1110 0100	0101 0bbb		$\overline{C}$ - * - - -	3	CF ← CF ⊻ (IX).b
XOR CF,(IY).b	1110 0101	0101 0bbb		$\overline{C}$ - * - - -	3	CF ← CF ⊻ (IY).b
XOR CF,(IX+d).b	1101 0100	dddd dddd	0101 0bbb	$\overline{C}$ - * - - -	5	CF ← CF ⊻ (IX+d).b
XOR CF,(IY+d).b	1101 0101	dddd dddd	0101 0bbb	$\overline{C}$ - * - - -	5	CF ← CF ⊻ (IY+d).b
XOR CF,(SP+d).b	1101 0110	dddd dddd	0101 0bbb	$\overline{C}$ - * - - -	5	CF ← CF ⊻ (SP+d).b
XOR CF,(HL+d).b	1101 0111	dddd dddd	0101 0bbb	$\overline{C}$ - * - - -	5	CF ← CF ⊻ (HL+d).b
XOR CF,(HL+C).b	1110 0111	0101 0bbb		$\overline{C}$ - * - - -	5	CF ← CF ⊻ (HL+C).b
XOR CF,(+SP).b	1110 0110	0101 0bbb		$\overline{C}$ - * - - -	4	SP ← SP+1:CF ← CF ⊻ (SP).b
XOR CF,(PC+A).b	0100 1111	0101 0bbb		$\overline{C}$ - * - - -	5	CF ← CF ⊻ (PC+A).b
SET CF	0000 0101			0 - 1 - - -	1	CF ← 1
CLR CF	0000 0100			1 - 0 - - -	1	CF ← 0
CPL CF	0000 0110			* - * - - -	1	JF ← CF:CF ← $\overline{CF}$
*2 DI	1100 1000	0011 1010		Z * - - - -	5	ZF ← $\overline{IMF}$:IMF ← 0
*2 EI	1100 0000	0011 1010		Z * - - - -	5	ZF ← $\overline{IMF}$:IMF ← 1

(5) Jump

MNEMONIC	OBJECT CODE (Binary)			FLAG J Z C H S V	CYCLE	FUNCTION
JRS T,$+2+d	100d dddd			1 - - - - -	4/2(t/f)	if JF=1 then PC ← PC+d else null
JRS F,$+2+d	101d dddd			1 - - - - -	4/2(t/f)	if JF=0 then PC ← PC+d else null
JR T,$+2+d	1101 1110	dddd dddd		1 - - - - -	4/2(t/f)	if JF=1 then PC ← PC+d else null
JR F,$+2+d	1101 1111	dddd dddd		1 - - - - -	4/2(t/f)	if JF=0 then PC ← PC+d else null
JR EQ,$+2+d JR Z,$+2+d	1101 1000	dddd dddd		1 - - - - -	4/2(t/f)	if ZF=1 then PC ← PC+d else null
JR NE,$+2+d JR NZ,$+2+d	1101 1001	dddd dddd		1 - - - - -	4/2(t/f)	if ZF=0 then PC ← PC+d else null
JR CS,$+2+d JR LT,$+2+d	1101 1010	dddd dddd		1 - - - - -	4/2(t/f)	if CF=1 then PC ← PC+d else null
JR CC,$+2+d JR GE,$+2+d	1101 1011	dddd dddd		1 - - - - -	4/2(t/f)	if CF=0 then PC ← PC+d else null
JR LE,$+2+d	1101 1100	dddd dddd		1 - - - - -	4/2(t/f)	if (CF∨ZF)=1 then PC ← PC+d else null
JR GT,$+2+d	1101 1101	dddd dddd		1 - - - - -	4/2(t/f)	if (CF∨ZF)=0 then PC ← PC+d else null
JR M,$+3+d	1110 1000	1101 0000	dddd dddd	1 - - - - -	5/3(t/f)	if SF=1 then PC ← PC+d else null
JR P,$+3+d	1110 1000	1101 0001	dddd dddd	1 - - - - -	5/3(t/f)	if SF=0 then PC ← PC+d else null
JR SLT,$+3+d	1110 1000	1101 0010	dddd dddd	1 - - - - -	5/3(t/f)	if (SF⊻VF)=1 then PC ← PC+d else null
JR SGE,$+3+d	1110 1000	1101 0011	dddd dddd	1 - - - - -	5/3(t/f)	if (SF⊻VF)=0 then PC ← PC+d else null
JR SLE,$+3+d	1110 1000	1101 0100	dddd dddd	1 - - - - -	5/3(t/f)	if ZF∨(SF⊻VF)=1 then PC ← PC+d else null
JR SGT,$+3+d	1110 1000	1101 0101	dddd dddd	1 - - - - -	5/3(t/f)	if ZF∨(SF⊻VF)=0 then PC ← PC+d else null
JR VS,$+3+d	1110 1000	1101 0110	dddd dddd	1 - - - - -	5/3(t/f)	if VF=1 then PC ← PC+d else null
JR VC,$+3+d	1110 1000	1101 0111	dddd dddd	1 - - - - -	5/3(t/f)	if VF=0 then PC ← PC+d else null
JR $+2+d	1111 1100	dddd dddd		1 - - - - -	4	PC ← PC+d
JP mn	1111 1110	nnnn nnnn	mmmm mmmm	- - - - - -	4	PC ← mn
JP gg	1110 1ggg	1111 1110		- - - - - -	3	PC ← gg
JP (x)	1110 0000	xxxx xxxx	1111 1110	- - - - - -	6	PC ← (x+1,x)
JP (vw)	1110 0001 1111 1110	wwww wwww	vvvv vvvv	- - - - - -	7	PC ← (vw+1,vw)
JP (DE)	1110 0010	1111 1110		- - - - - -	5	PC ← (DE+1,DE)
JP (HL)	1110 0011	1111 1110		- - - - - -	5	PC ← (HL+1,HL)
JP (IX)	1110 0100	1111 1110		- - - - - -	5	PC ← (IX+1,IX)
JP (IY)	1110 0101	1111 1110		- - - - - -	5	PC ← (IY+1,IY)
JP (IX+d)	1101 0100	dddd dddd	1111 1110	- -	7	PC ← (IX+d+1,IX+d)
JP (IY+d)	1101 0101	dddd dddd	1111 1110	- - - - - -	7	PC ← (IY+d+1,IY+d)
JP (SP+d)	1101 0110	dddd dddd	1111 1110	- - - - - -	7	PC ← (SP+d+1,SP+d)
JP (HL+d)	1101 0111	dddd dddd	1111 1110	- - - - - -	7	PC ← (HL+d+1,HL+d)
JP (HL+C)	1110 0111	1111 1110		- - - - - -	7	PC ← (HL+C+1,HL+C)
JP (+SP)	1110 0110	1111 1110		- - - - - -	6	SP ← SP+1:PC ← (SP+1,SP)
JP (PC+A)	0100 1111	1111 1110		- - - - - -	7	PC ← (PC+A+1,PC+A)

(6) Call, return, software interrupt and no operation

MNEMONIC	OBJECT CODE (Binary)			FLAG J Z C H S V	CYCLE	FUNCTION
CALLV n	0111 nnnn			- - - - - -	7	(SP,SP-1) ← PC-1:SP ← SP-2; PC ← (n×2+FFC1H,n×2+FFC0H)
CALL mn	1111 1101	nnnn nnnn	mmmm mmmm	- - - - - -	6	(SP,SP-1) ← PC+1:SP ← SP-2; PC ← mn
CALL gg	1110 1ggg	1111 1101		- - - - - -	6	(SP,SP-1) ← PC+1:SP ← SP-2; PC ← gg
CALL (x)	1110 0000	xxxx xxxx	1111 1101	- - - - - -	9	(SP,SP-1) ← PC+1:SP ← SP-2; PC ← (x+1,x)
CALL (vw)	1110 0001 1111 1101	wwww wwww	vvvv vvvv	- - - - - -	10	(SP,SP-1) ← PC+1:SP ← SP-2; PC ← (vw+1,vw)
CALL (DE)	1110 0010	1111 1101		- - - - - -	8	(SP,SP-1) ← PC+1:SP ← SP-2; PC ← (DE+1,DE)
CALL (HL)	1110 0011	1111 1101		- - - - - -	8	(SP,SP-1) ← PC+1:SP ← SP-2; PC ← (HL+1,HL)
CALL (IX)	1110 0100	1111 1101		- - - - - -	8	(SP,SP-1) ← PC+1:SP ← SP-2; PC ← (IX+1,IX)
CALL (IY)	1110 0101	1111 1101		- - - - - -	8	(SP,SP-1) ← PC+1:SP ← SP-2; PC ← (IY+1,IY)
CALL (IX+d)	1101 0100	dddd dddd	1111 1101	- - - - - -	10	(SP,SP-1) ← PC+1:SP ← SP-2; PC ← (IX+d+1,IX+d)
CALL (IY+d)	1101 0101	dddd dddd	1111 1101	- - - - - -	10	(SP,SP-1) ← PC+1:SP ← SP-2; PC ← (IY+d+1,IY+d)
CALL (SP+d)	1101 0110	dddd dddd	1111 1101	- - - - - -	10	(SP,SP-1) ← PC+1:SP ← SP-2; PC ← (SP+d+1,SP+d)
CALL (HL+d)	1101 0111	dddd dddd	1111 1101	- - - - - -	10	(SP,SP-1) ← PC+1:SP ← SP-2; PC ← (HL+d+1,HL+d)
CALL (+SP)	1110 0110	1111 1101		- - - - - -	9	(SP+1,SP) ← PC:SP ← SP-1; PC ← (SP+3,SP+2)
CALL (HL+C)	1110 0111	1111 1101		- - - - - -	10	(SP,SP-1) ← PC+1:SP ← SP-2; PC ← (HL+C+1,HL+C)
CALL (PC+A)	0100 1111	1111 1101		- - - - - -	10	(SP,SP-1) ← PC+1:SP ← SP-2; PC ← (PC+A+1,PC+A)
RET	1111 1010			- - - - - -	6	SP ← SP+2:PC ← (SP,SP-1)
RETI	1111 1011			* * * * * *	6	SP ← SP+3:PC ← (SP-1,SP-2); PSW ← (SP):IMF ← (SP).0
RETN	1110 1000	1111 1011		* * * * * *	7	SP ← SP+3:PC ← (SP-1,SP-2); PSW ← (SP):IMF ← (SP).0
SWI	1111 1111			- - - - - -	9	(SP) ← PSW:(SP).0 ← IMF; (SP-1,SP-2) ← PC-1: PC ← (FFFDH,FFFCH)
NOP	0000 0000			- - - - - -	1	no Operation

*1 : "rr" is for only WA, BC, DE or HL

*2 : Assembler expansion machine instruction

1.7 TLCS-870/C INSTRUCTION CODE MAP

(1) First opcodes

L / U	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP				CLR CF	SETCF	CPL CF	CMP (x), n	LDW (mem), mn (x)	(HL)	LD (mem), mn (x)	(HL)	LD A, (mem) (x)	(HL)	LD (mem), A (x)	(HL)
1	LD A, r A	W	C	B	E	D	L	H	LD r, n A	W	C	B	E	D	L	H
2	INC r A	W	C	B	E	D	L	H	DEC r A	W	C	B	E	D	L	H
3	INC rr WA	BC	DE	HL	IX	IY	SP	LD SP,SP+d	DEC rr WA	BC	DE	HL	IX	IY	SP	LD SP,SP-d
4	LD r, A A	W	C	B	E	D	L	H	LD rr, mn WA	BC	DE	HL	IX	IY	SP	(src) (PC+A)
5	PUSH rr (Note 4) WA	BC	DE	HL	(dst) (IX+d)	(IY+d)	(SP+d)	(HL+d)	LD CF, (x).b 0	1	2	3	4	5	6	7
6	ADDC A, n	ADD A, n	SUBB A, n	SUB A, n	AND A, n	XOR A, n	OR A, n	CMP A, n								
7	CALLV n															
8	JRS T, a															
9																
A	JRS F, a															
B																
C	SET (x).b 0	1	2	3	4	5	6	7	CLR (x).b 0	1	2	3	4	5	6	7
D	POP rr (Note 4) WA	BC	DE	HL	(src) (IX+d)	(IY+d)	(SP+d)	(HL+d)	JR cc, a EQ/Z	NE/NZ	LT/CS	GE/CC	LE	GT	T	F
E	source memory prefix : (src) (x)	(vw)	(DE)	(HL)	(IX)	(IY)	(+SP)	(HL+C)	register prefix : g/gg A/WA	W/BC	C/DE	B/HL	E/IY	D/IY	L/SP	H/HL
F	destination memory prefix : (dst) (x)	(vw)	(DE)	(HL)	(IX)	(IY)	(SP−)	(HL+C)			RET	RETI	JR a	CALL mn	JP mn	SWI

(Note 1) Blank codes are undefined.

(Note 2) The register prefix instructions (E8 to EFH) have the second opcodes that are listed in Section 1.7, paragraph (2) in the next page.

(Note 3) The source/destination prefix instructions (E0 to $E7_H$, F0 to $F7_H$, 54 to 57_H, D4 to $D7_H$, and $4F_H$) have the second opcodes that are listed in Section 1.7, paragraph (3) in the next page.

(Note 4) Note 4: The WA, BC, DE, and HL are the only registers (rr) that can be used in the instructions PUSH rr and POP rr.

(2) Second opcodes (Register prefixes)

U \ L	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	ADDC A,g	ADD A,g	SUBB A,g	SUB A,g	AND A,g	XOR A,g	OR A,g	CMP A,g	ADDC W,g	ADD W,g	SUBB W,g	SUB W,g	AND W,g	XOR W,g	OR W,g	CMP W,g
1	C,g	C,g	C,g	C,g	C,g	C,g	C,g	C,g	C,g	B,g	B,g	B,g	B,g	B,g	B,g	B,g
2	E,g	E,g	E,g	E,g	E,g	E,g	E,g	E,g	D,g	D,g	D,g	D,g	D,g	D,g	D,g	D,g
3	L,g	L,g	L,g	L,g	L,g	L,g	L,g	L,g	H,g	H,g	H,g	H,g	H,g	H,g	H,g	H,g
4	LD r,g								LD rr,gg							
	A	W	C	B	E	D	L	H	WA	BC	DE	HL	IX	IY	SP	
5	XOR CF,g.b								LD CF,g.b							
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
6	ADDC g,n	ADD g,n	SUBB g,n	SUB g,n	AND g,n	XOR g,n	OR g,n	CMP g,n	ADDC gg,mn	ADD gg,mn	SUBB gg,mn	SUB gg,mn	AND gg,mn	XOR gg,mn	OR gg,mn	CMP gg,mn
7	XCH r,g								XCH rr,gg							
	A	W	C	B	E	D	L	H	WA	BC	DE	HL	IX	IY	SP	
8	ADDC WA,gg	ADD WA,gg	SUBB WA,gg	SUB WA,gg	AND WA,gg	XOR WA,gg	OR WA,gg	CMP WA,gg	ADDC BC,gg	ADD BC,gg	SUBB BC,gg	SUB BC,gg	AND BC,gg	XOR BC,gg	OR BC,gg	CMP BC,gg
9	DE,gg	DE,gg	DE,gg	DE,gg	DE,gg	DE,gg	DE,gg	DE,gg	HL,gg	HL,gg	HL,gg	HL,gg	HL,gg	HL,gg	HL,gg	HL,gg
A	IX,gg	IX,gg	IX,gg	IX,gg	IX,gg	IX,gg	IX,gg	IX,gg	IY,gg	IY,gg	IY,gg	IY,gg	IY,gg	IY,gg	IY,gg	IY,gg
B	SP,gg	SP,gg	SP,gg	SP,gg	SP,gg	SP,gg	SP,gg	SP,gg	HL,gg	HL,gg	HL,gg	HL,gg	HL,gg	HL,gg	HL,gg	HL,gg
C	SET g.b								CLR g.b							
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
D	JR cc,a															
	M	P	SLT	SEG	SLE	SGT	VS	VC	PUSH gg	POP gg	DAA g	DAS g	PUSH PSW	POP PSW	LD PSW,n	
E	CPL g.b								LD g.b,CF							
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
F	SHLCA gg	SHRCA gg	MUL gg_H,gg_L	DIV gg,C	SHLC gg	SHRC gg	ROLC gg	RORC gg			NEG CS,gg	RETN		CALL gg	JP gg	SWAP g

(Note 1) Blank codes are undefined.

(Note 2) The first opcodes are listed in the table below.

g	First opcode
A	E8
W	E9
C	EA
B	EB
E	EC
D	ED
L	EE
H	EF

gg	First opcode
WA	E8
BC	E9
DE	EA
HL	EB
IX	EC
IY	ED
SP	EE
HL	EF

(3) 2nd opCode (Memory phefix)

U \ L	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	ADDC A, (src)	ADD A, (src)	SUBB A, (src)	SUB A, (src)	AND A, (src)	XOR A, (src)	OR A, (src)	CMP A, (src)	ADDC W, (src)	ADD W, (src)	SUBB W, (src)	SUB W, (src)	AND W, (src)	XOR W, (src)	OR W, (src)	CMP W, (src)
1	C, (src)	C, (src)	C, (src)	C, (src)	C, (src)	C, (src)	C, (src)	C, (src)	B, (src)	B, (src)	B, (src)	B, (src)	B, (src)	B, (src)	B, (src)	B, (src)
2	E, (src)	E, (src)	E, (src)	E, (src)	E, (src)	E, (src)	E, (src)	E, (src)	D, (src)	D, (src)	D, (src)	D, (src)	D, (src)	D, (src)	D, (src)	D, (src)
3	L, (src)	L, (src)	L, (src)	L, (src)	L, (src)	L, (src)	L, (src)	L, (src)	H, (src)	H, (src)	H, (src)	H, (src)	H, (src)	H, (src)	H, (src)	H, (src)
4	LD r, (src)								LD rr, (src)							
	A	W	C	B	E	D	L	H	WA	BC	DE	HL	IX	IY	SP	
5	XOR CF, (src).b								LD CF, (src).b							
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
6	ADDC (src), n	ADD (src), n	SUBB (src), n	SUB (src), n	AND (src), n	XOR (src), n	OR (src), n	CMP (src), n	LD (dst), rr							
									WA	BC	DE	HL	IX	IY	SP	
7	XCH r, (src)								LD (dst), r							
	A	W	C	B	E	D	L	H	A	W	C	B	E	D	L	H
8	ADDC WA, (src)	ADD WA, (src)	SUBB WA, (src)	SUB WA, (src)	AND WA, (src)	XOR WA, (src)	OR WA, (src)	CMP WA, (src)	ADDC BC, (src)	ADD BC, (src)	SUBB BC, (src)	SUB BC, (src)	AND BC, (src)	XOR BC, (src)	OR BC, (src)	CMP BC, (src)
9	DE, (src)	DE, (src)	DE, (src)	DE, (src)	DE, (src)	DE, (src)	DE, (src)	DE, (src)	HL, (src)	HL, (src)	HL, (src)	HL, (src)	HL, (src)	HL, (src)	HL, (src)	HL, (src)
A	IX, (src)	IX, (src)	IX, (src)	IX, (src)	IX, (src)	IX, (src)	IX, (src)	IX, (src)	IY, (src)	IY, (src)	IY, (src)	IY, (src)	IY, (src)	IY, (src)	IY, (src)	IY, (src)
B	SP, (src)	SP, (src)	SP, (src)	SP, (src)	SP, (src)	SP, (src)	SP, (src)	SP, (src)	HL, (src)	HL, (src)	HL, (src)	HL, (src)	HL, (src)	HL, (src)	HL, (src)	HL, (src)
C	SET (src).b								CLR (src).b							
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
D									XCH rr, (src)							
									WA	BC	DE	HL	IX	IY	SP	HL
E	CPL (src).b								LD (src).b, CF							
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
F	INC (src)		SET (src). A	LD (src).A, CF			ROLD A,(src)	RORD A,(src)	DEC (src)	LD (dst), n	CLR (src).A	CPL (src).A	LD CF, CF, (src).A	CALL (src)	JP (src)	

(Note 1) Blank codes are undefined.

(Note 2) The first opcodes are listed in the table below.

(src)	First opcode
(x)	E0
(vw)	E1
(DE)	E2
(HL)	E3
(IX)	E4
(IY)	E5
(+SP)	E6
(HL+C)	E7
(IX+d)	D4
(IY+d)	D5
(SP+d)	D6
(HL+d)	D7
(PC+A)	4F

(dst)	First opcode
(x)	F0
(vw)	F1
(DE)	F2
(HL)	F3
(IX)	F4
(IY)	F5
(SP−)	F6
(HL+C)	F7
(IX+d)	54
(IY+d)	55
(SP+d)	56
(HL+d)	57