CMOS 8-BIT MICROCONTROLLER

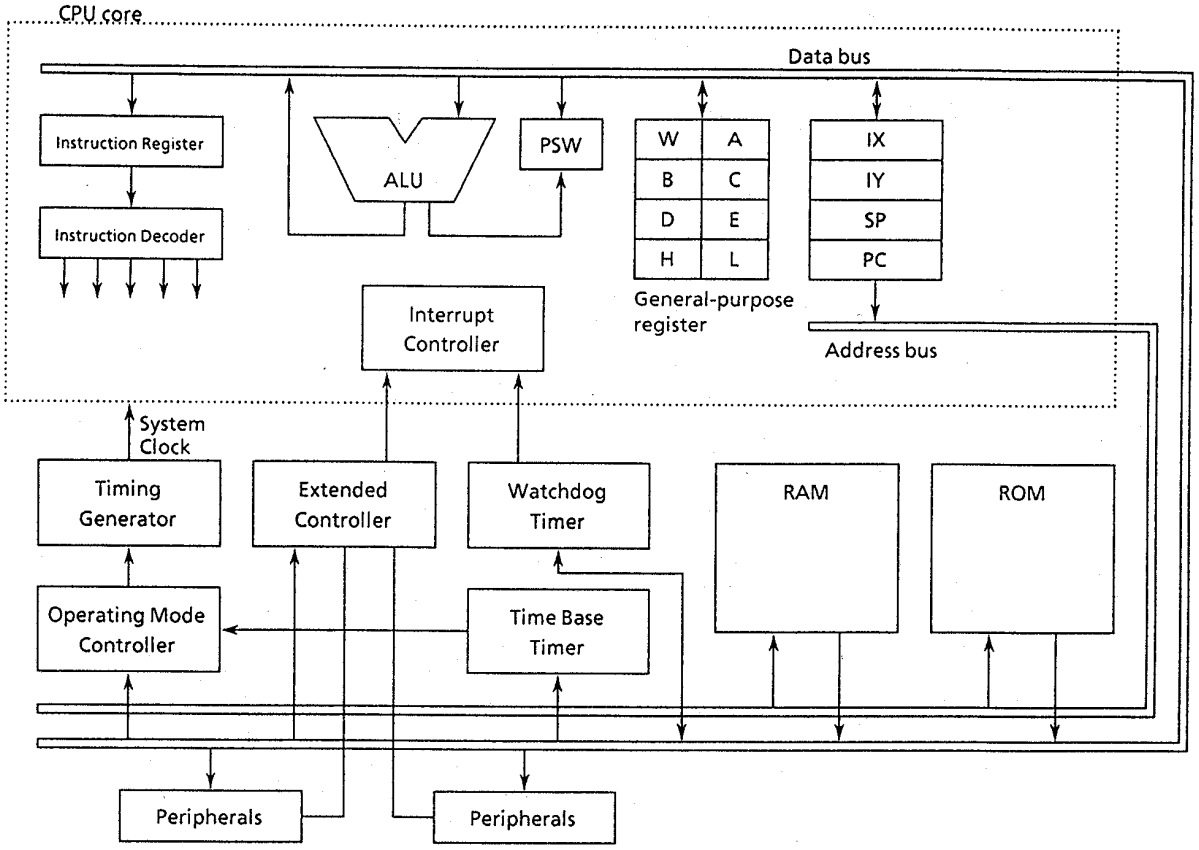
TLCS-870/C Series

The TLCS-870/C series is the compact and high-speed / high-performance with low power-consumption 8-bit single-chip microcomputer, of Toshiba proprietary.

FEATURES

- ◆ Orthogonal and plenty instruction set : 731 basic instructions of 132 types
  More suitable for C compiler, with optimized instruction set consists of (Sub set on TLCS-870/X's) + alph
  - Multiplication and Division (8bits × 8bits, 16bits ÷ 8bits)
  - Bit manipulation (Set/Clear/Complement/Load/Store/Test/Exclusive OR)
  - 16-bit data operations
  - 1-byte object code jump/subroutine-call (Short relative jump / Vector call)
- ◆ Register Construction
  - General purpose registers : 8bits × 8 (also available for 16bits × 4)
  - 16-bit general-purpose registers : 16bits × 2
  - Program Counter (PC) : 16bits × 1
  - Stack Pointer (SP) : 16bits × 1
  - Program Status Word (PSW) : 6bits × 1
- ◆ Memory Space
  for both Program and Data memory : 64K byte
- ◆ Memory Mapped I/O
- ◆ Interrupts
  - 15-factor basis (expandable)
  - Individual latch for each factor
  - External interrupt, with alternative edge and noise-cancellor
- ◆ Lower power Consumption Operation (6 modes)
  - STOP mode Stops Oscillation (battery or capacitor backup)
  - SLOW mode Operates slowly under Low frequency Clock
  - IDLE1 mode Stops CPU, leaves Peripheral hardware operated (under high frequency clock).
  - IDLE2 mode Stops CPU, leaves Peripheral hardware operated (under high frequency and low frequency clock).
  - IDLE0 mode Stops CPU, leaves only Time Base Timer operated (under high frequency clock).
  - SLEEP mode Stops CPU, leaves Peripheral hardware operated (under low frequency clock).
  - SLEEP2 mode Stops CPU, leaves only the Time-Based-Timer on Peripheral hardware operated (under low frequency clock).
  - SLEEP0 mode Stops CPU, leaves only Time Base Timer operated (under low frequency clock).
- ◆ High Speed Operation
  - Minimum Instruction Execution time : 200 ns @20 MHz (under non-division)
- ◆ High Speed for Low Voltage (for L version)
  - Minimum Operating Voltage : 1.8 V @4.2 MHz
- ◆ External Memory accessible
- ◆ Fail-safe functions
  - Watchdog Timer (Reset or Interrupt)
  - Address Trap (Reset or Interrupt)
  - System Clock Reset
  - Undefined Instruction Interrupt
  - Software Interrupt
  - Reset Terminal
  - Detecting Voltage for power supply (applied partly)
  - Detecting halt of oscillation (applied partly)

BLOCK DIAGRAM



Note : PSW ; Program Status Word

OPERATIONAL DESCRIPTION

1. CPU Core Functions

The CPU core consists of CPU, a system clock controller and an interrupt controller.
This section explains CPU core, program memory and data memory.

1.1 Memory Address Map

The TLCS-870/C series places its memory space in 64K bytes. The internal memory consists of ROM, RAM, SFR (special function register), DBR (data buffer register), MEM1 and MEM2 and they are placed in its 64K bytes. The figure 1-1 indicates the memory map of TLCE-870/C. The general-purpose registers are not placed in memory space.

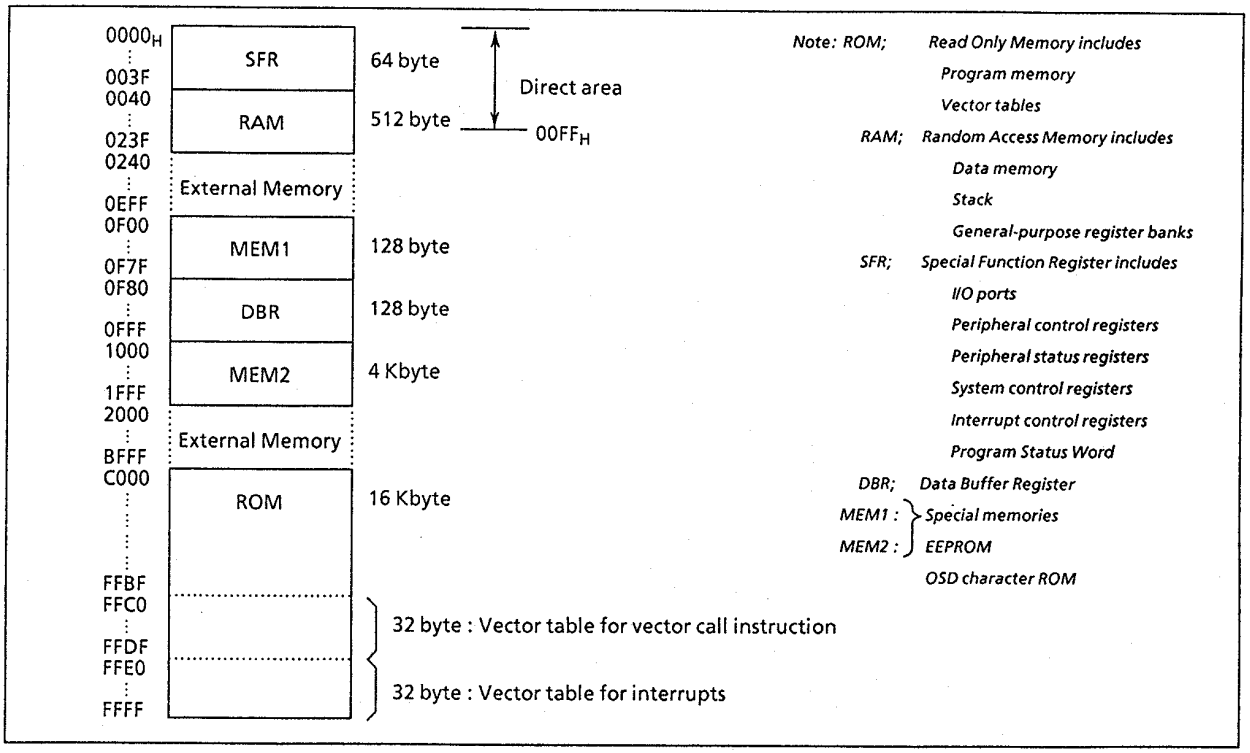


Figure 1-1. Memory Address Map (Example)

The addresses 0000_H to $00FF_H$ are direct area; instructions with shorten operations are available against such an area.

1.2 Program Memory(ROM)

The addresses $FFC0_H$ to $FFFF_H$ are utilized also for specified purposes.

(1) Interrupt Vector Table ($FFE0_H$ to $FFFF_H$)

This table consists of a reset vector and 15 interrupt vectors (2bytes / vector). The reset vector stores a reset store address. The interrupt vectors store interrupt service routine entry addresses respectively.

(2) Vector table for **vector call** instructions (addresses $FFC0_H$ to $FFDF_H$)

This table stores call vectors (subroutine entry address, 2 bytes/vector) for the vector call instructions [CALLV n] (n ; table number). There are 16 vectors. The CALLV instruction increases memory efficiency when utilized in frequently used subroutine calls (called from 3 or more locations).

Programs and fixed data are stored in the program memory. The instruction to be executed next is read from the address indicated by the current contents of the program counter (PC). There are relative jump and absolute jump instructions. The concepts of page or bank boundaries are not used in the program memory concerning any jump instruction.

Example: The relationship between the jump instructions and the PC.

① 5-bit PC-relative jump [JRS cc, $ + 2 + d]

E8C4: JRS T, $ + 2 + 08H

When JF = 1, the jump is made to $E8CE_H$, which is 08_H added to the contents of the PC. (The PC contains the address of the instruction being executed + 2; therefore, in this case, the PC contents are $E8C4_H + 2 = E8C6_H$.)

② 8-bit PC-relative jump [JR cc, $ + 2 + d] / [JR cc, $ + 3 + d]

E8C4H : JR Z, $ + 2 + 80H

When ZF = 1, the jump is mode to $E846_H$, which is $FF80_H$ (– 128) added to the current contents of the PC ($E8C6_H$).

③ 16-bit absolute jump [JP lmn]

E8C4H : JP 0C235H

An unconditional jump is made to address $C235_H$. The absolute jump instruction can jump anywhere within the entire 64K-bytes space.

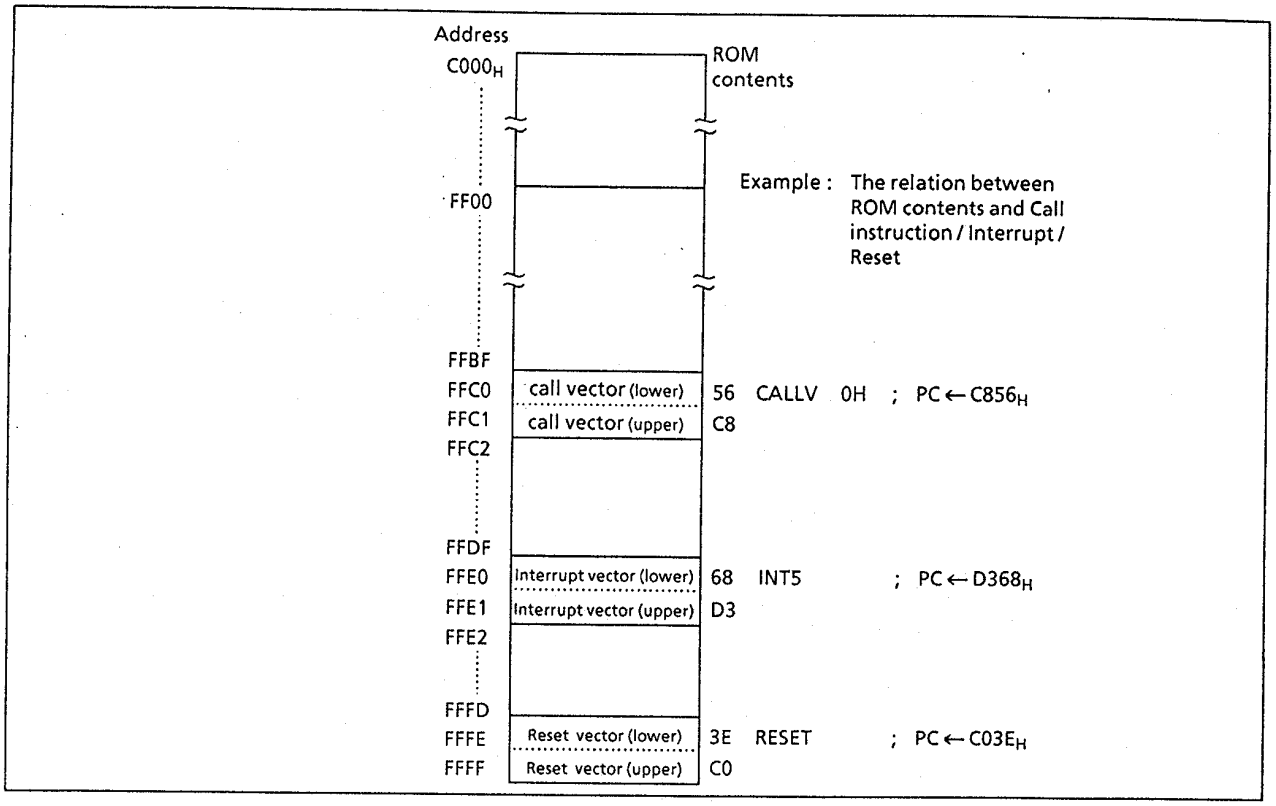


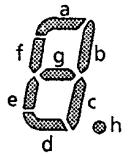
Figure 1-2. Program Memory Map (Example)

In the TLCS-870/C Series, the same instruction used to access the data memory (e. g. [LD A, (IX)]) is also used to read out ROM data stored in the program memory. The register-offset-PC-relative addressing (PC + A) instructions can also be used, and the code conversion, table look-up and n-way multiple-direction jump processing can easily be programmed.

Example 1 : Loads the ROM contents at the address specified by the HL register contents into the accumulator:

```
LD      A, (HL)          ; A ←ROM (HL)
```

Example 2 : Converts BCD to 7-segment code (common anode LED). When A = 05_H, 92_H is output to port P3 after executing the following program:




```
        ADD     A, TABLE – $ – 4     ; P3 ←ROM (TABLE + A)
        LD      (P3), (PC + A)
        JRS     T, SNEXT
TABLE : DB      0C0H, 0F9H, 0A4H, 0B0H, 99H, 92H, 82H, 0D8H, 80H, 98H
SNEXT :
```

SHLC A
JP (PC + A)
34
C2
78
C3
37
DA
B0
E1

Notes: "$" is a header address of ADD instruction.
DB is a byte data definition instruction.

Example 3 : Multiple jump in accordance with the contents of accumulator ($0 \leqq A \leqq 3$):

```
        SHLC    A            ; if A = 00H then PC←C234H
        JP      (PC + A)       if A = 01H then PC←C378H
                               if A = 02H then PC←DA37H
                               if A = 03H then PC←E1B0H
        DW      0C234H, 0C378H, 0DA37H, 0E1B0H
```

Note: DW is a word-data definition instruction. (word = 2bytes)

1.3 Program Counter (PC)

The program counter (PC) is a 16-bit register which indicates the program memory address where the instruction to be executed next is stored. After reset, the user defined reset vector stored in the vector table (addresses $FFFF_H$ and $FFFE_H$) is loaded into the PC ; therefore, program execution is possible from any desired address. For example, when $C0_H$ and $3E_H$ are stored at addresses $FFFF_H$ and $FFFE_H$, respectively, the execution starts from address $C03E_H$ after reset.
The TLCS-870/X utilizes pipelined processing (instruction pre-fetch); therefore, the PC always indicates 2 addresses in advance. For example, while a 1-byte instruction stored at address $C123_H$ is being executed, the PC contains $C125_H$.

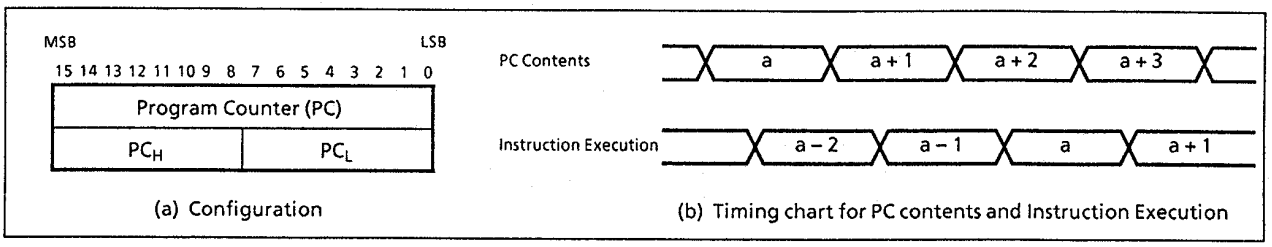


Figure 1-3. Program Counter

1.4 Data Memory (RAM)

The TLCS-870/C can place internal data memory (internal ROM or internal RAM) and external data memory (RAM or ROM), up to 64K bytes.
Since the contents of data memory is undefined after reset, initialize it on software.

Example : Clearing RAM

```
            LD      HL, 0040H       ; Sets starting address
            LD      A, H            ; Sets initializing data
            LD      BC, 0FFFH       ; Sets number of bytes
SRAMCLR :   LD      (HL), A
            INC     HL
            DEC     BC
            JRS     F, SRAMCLR
```

1.5 General-purpose Registers

The TLCS-870/C has eight 8-bit registers internally as general-purpose registers: W, A, B, C, D, E, H and L. They are also available for 16-bit registers, by pairing: WA, BC, DE and HL.

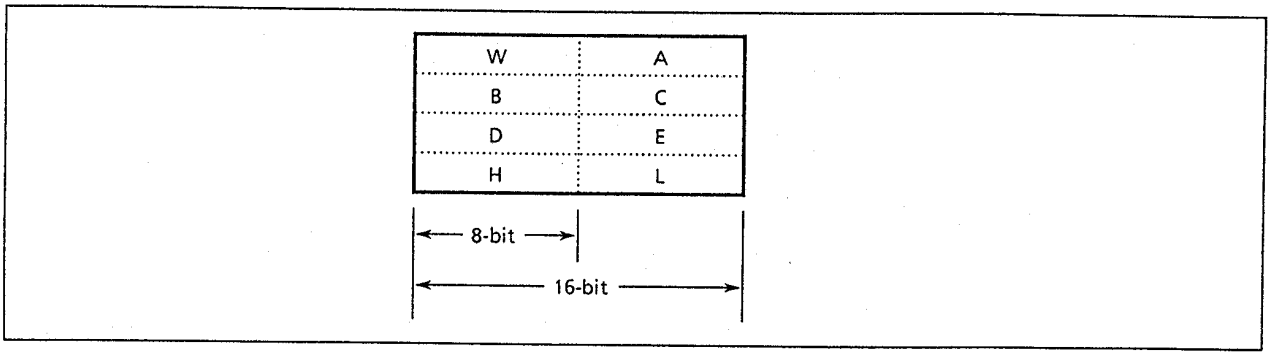


Figure 1-4. General-purpose Registers

The registers W, A, B, C, D, E, H and L are used for 8-bit handling. The paired registers WA, BC, DE and HL are used for 16-bit handling. These registers function as follows besides general-purpose registers.

(1) A

The A register is utilized for specifying bit to be operated, under bit manipulations.

Example : a) SET (56H).A ; Sets the bit, specified by A register, on address 0056_H to '1'.
b) CPL (IX + 03H).A ; Inverts the bit, specified by A register, on address which sum (sign extended) of IX and 03_H is

In addition, the A register is also utilized for offset register under register offset relative addressing (PC + A) mode.

Example : LD A,(PC + A) ; Loads the data, located on address which sum (sign extended) of PC and A indicates, on A register

(2) C

The C register is utilized for divisor on division. The CPU leaves its residue on the register which has been the upper byte of dividend, while its quotient is loaded on the register which has been lower byte of it.

Example : DIV WA,C ; A←WA ÷ C, W←residue

In addition, the C register is also utilized for offset register under register index addressing (HL + C) mode.

Example : LD A,(HL + C) ; Loads the data, located on address which sum (sign extended) of HL and C indicates, on A register

(3) DE

The DE register is utilized for address-specifying register under indirect addressing mode.

Example : LD A,(DE) ; Loads the data, located on address DE indicates, on A register

(4) HL

The HL register is utilized for address-specifying register under indirect addressing mode and for index register under index addressing mode.

Example : a) LD A,(HL) ; Loads the data, located on address HL indicates, on A register
b) LD A,(HL + 52H) ; Loads the data, located on address which sum (sign extended) of HL and the immediate data 52_H indicates, on A register
c) LD A,(HL + C) ; Loads the data, located on address which sum (sign extended) of HL and C indicates, on A register

The general-purpose registers are not initialized by resetting. The values after power-on are undefined.

1.6 16-bit general-purpose Registers (IX, IY)

The TLCS-870/C has two 16-bit general-purpose registers (IX, IY) internally. Each of them consists of 16 bits and they are utilized for address-specifying register under register indirect addressing mode and for index register under index addressing mode.

Example: a) LD A,(IX) ; Loads the data, located on address IX indicates, on A register
b) LD A,(IY + 52H) ; Loads the data, located on address which sum (sign extended) of IY and the immediate data 52_H indicates, on A register

Besides they are utilized for data transfer and arithmetic and logic operation.

Example: a) LD IX,(3AH) ; Loads the data, located on address $003A_H$, on IX register
b) ADD IX,5678H ; Adds IX to 5678_H, and loads its result on IX

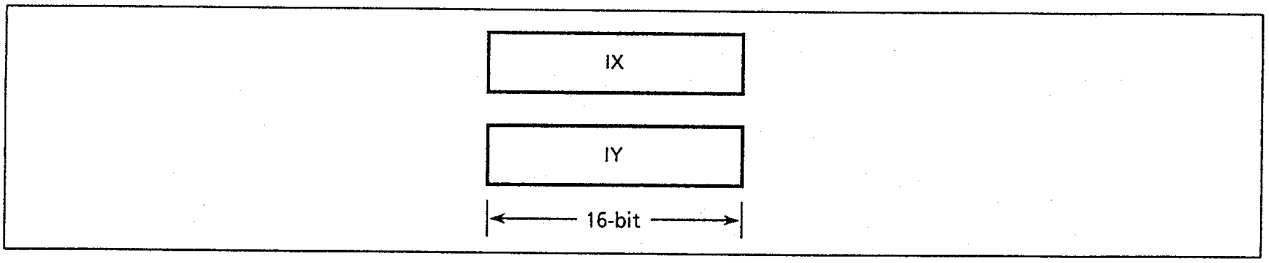


Figure 1-5. 16-bit general-purpose registers

The 16-bit general-purpose registers are not initialized by resetting. These values after power-on are undefined.

1.7 Program Status Word (PSW)

The Program Status Word (PSW) consists of the following 6-bit flags, and it is located on address $003F_H$ in SFR area.

- Jump Status (JF)
- Zero Flag (ZF)
- Carry Flag (CF)
- Half Carry Flag (HF)
- Sign Flag (SF)
- Overflow Flag (VF)

In order to access PSW, specified instructions are utilized. However, read instructions for data memory are also available.

```
Example: 1) LD PSW,00H   ;  Writes 00H on PSW
         2) PUSH PSW     ;  Saves data on PSW to stack
         3) POP PSW      ;  Restores stacked data to PSW
         4) LD A,(3FH)   ;  Loads data on PSW on A register
```

As write instruction for data memory is executed, such data is not written on PSW. In that case, a certain value specified by the instruction is loaded on PSW.

```
Example:  LD (3FH),00H  ;  PSW does not become 00H. On account of instruction [LD (x),n], JF
                           becomes '1' and other flags remain unchanged.
```

As interrupt is accepted, PSW is saved to stack meanwhile it accompanies with the program counter (PC). On executing instruction [RETI] / [RETN] the stacked data returns to PSW.

The flags on PSW is not initialized by resetting. The value after power-on is undefined.

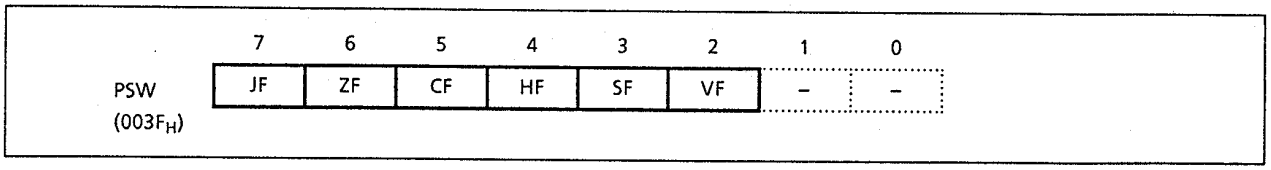


Figure 1-6. PSW Configuration

The flags are set/cleared on account of the conditions specified by the instructions. The flags, except for the Half Carry Flag (HF), are eligible to dominate conditions to execute jump instructions [JR cc,a], [JRS cc,a].

cc	Means	Condition
T	True	JF = 1
F	False	JF = 0
Z	Zero	ZF = 1
NZ	Not zero	ZF = 0
CS	Carry Set	CF = 1
CC	Carry Clear	CF = 0
VS	Overflow Set	VF = 1
VC	Overflow Clear	VF = 0
M	Minus	SF = 1
P	Plus	SF = 0
EQ	Equal	ZF = 1
NE	Not Equal	ZF = 0
LT	Unsigned Less Than	CF = 1
GE	Unsigned Greater Than or Equal	CF = 0
LE	Unsigned Less Than or Equal	(CF∨ZF) = 1
GT	Unsigned Greater Than	(CF∀ZF) = 0
SLT	Signed Less Than	(SF∀VF) = 1
SGE	Signed Greater Than or Equal	(SF∀VF) = 0
SLE	Signed Less Than or Equal	ZF∨ (SF∀VF) = 1
SGT	Signed Greater Than	ZF∨ (SF∀VF) = 0

(1) **Zero flag (ZF)**

The ZF is set to "1" if the operation result or the transfer data is 00_H (for 8-bit operations and data Transfers)/0000_H (for 16-bit operations); otherwise the ZF is cleared to "0".

During the bit manipulation instruction [SET, CLR, and CPL], the ZF is set to "1" if the contents of the specified bit is "0" ; otherwise the ZF is cleared to "0".

This flag is set to "1" when the upper 8 bits of the product are 00_H during the multiplication instruction [MUL r1, r2], and when 00_H for the remainder during the division instruction [DIV rr, C] ; otherwise it is cleared to "0".

(2) **Carry flag (CF)**

The CF is set to "1" when a carry out of the MSB (most significant bit) of the result occurred during addition or when a borrow into the MSB of the result occurred during subtraction ; otherwise the CF is cleared to "0". During [DIV rr, C] instruction, this flag is set to "1" when the divisor is 00_H (divided by zero error), or when the quotient is 100_H or higher (overflow error) ; otherwise it is cleared. The CF is also affected during the shift/rotate instructions.

The data shifted out from a register is set to the CF.

This flag is also a 1-bit register (a boolean accumulator) for the bit manipulation instructions. Set/clear/complement are possible with the CF manipulation instructions.

Example : Bit manipulation (Calculate the exclusive OR from the 2-bit data: bit 5 on address 0007_H and bit 0 on address $009A_H$. Then load its result on bit 2 on address 0001_H.)

```
LD      CF,(07H).5      ; (0001H)2 ← (0007H)5 ∀ (009AH)0
XOR     CF,(9AH).0
LD      (01H).2,CF
```

(3) **Half carry flag (HF)**

The HF is set to "1" when a carry occurred between bits 3 and 4 of the operation result during an 8-bit addition, or when a borrow occurred from bit 4 into bit 3 of the result during an 8-bit subtraction ; otherwise the HF is cleared to "0" . This flag is useful in the decimal adjustment for BCD operations (adjustments using the [DAA r], or [DAS r] instructions).

Example : BCD operation
(A becomes 47_H after executing the following program when A = 19_H, B = 28_H)

```
ADD   A, B      ; A ← 41H, HF ← 1, CF = 0
DAA   A         ; A ← 41H + 06H = 47H (decimal-adjust)
```

(4) **Sign flag (SF)**

The SF is set to "1" if the MSB of the result is "1" during arithmetic operations [ADD, ADDC, SUB, SUBB, CMP, MCMP, SHLCA and SHRCA] ; otherwise the SF is cleared to "0".

(5) **Overflow flag (VF)**

The VF is set to "1" when an arithmetic overflow occurred during arithmetic operations [ADD, ADDC, SUB, SUBB,CMP, MCMP, SHLCA and SHRCA] ; otherwise the VF is cleared to "0".
Therefore the VF indicates a minus result produced as sum of two plus numbers or a plus result produced as the sum of two minus numbers when adding [ADD, ADDC].

(6) **Jump status flag (JF)**

Zero or carry information is set to the JF after operation (e. g. INC, ADD, CMP, TEST).
The JF provides the jump condition for conditional jump instructions [JRS T/F, a], [JR T/F, a] (T or F is a condition code). Jump is performed if the JF is "1" for a true condition (T), or the JF is "0" for a false condition (F).
The JF is set to "1" after executing the load/exchange/swap/nibble rotate/jump instruction, so that [JRS T, a] and [JR T, a] can be regarded as an unconditional jump instruction.

Example : Jump status flag and conditional jump instruction

```
INC   A
JRS   T, SLABLE1    ; Jump when a carry is caused by the immediately
 ⋮                    preceding operation instruction.
LD    A, (HL)
JRS   T, SLABLE2    ; The JF is set to "1" by the immediately preceding
 ⋮                    instruction, making it an unconditional jump
                      instruction.
```

Example : The accumulator and flags will become as shown below after executing the following instructions when the WA register pair, the HL register pair, the data memory at address $00C5_H$, the carry flag, the half carry flag, the sign flag and the overflow flag contents being "$219A_H$", "$00C5_H$", "$D7_H$", "1", "0", "1" and "0", respectively.

Instruction	Acc. after execution	Flag after execution					
		JF	ZF	CF	HF	SF	VF
ADDC A, (HL)	72	1	0	1	1	0	1
SUBB A, (HL)	C2	1	0	1	0	1	0
CMP A, (HL)	9A	0	0	1	0	1	0
AND A, (HL)	92	0	0	1	0	1	0
LD A, (HL)	D7	1	0	1	0	1	0
ADD A, 66H	00	1	1	1	1	0	0

Instruction	Acc. after execution	Flag after execution					
		JF	ZF	CF	HF	SF	VF
INC A	9B	0	0	1	0	1	0
ROLC A	35	1	0	1	0	1	0
RORC A	CD	0	0	0	0	1	0
ADD WA,0F508H	16A2	1	0	1	0	0	0
MUL W, A	13DA	0	0	1	0	1	0
SET A.5	BA	1	1	1	0	1	0

1.8 Stack and Stack Pointer

1.8.1 Stack

The stack provides the area in which the return address or status, etc. are saved before a jump is performed to the processing routine during the execution of a subroutine call instruction or the acceptance or an interrupt. On a subroutine call instruction, the contents of the PC (the return address) is saved; on an interrupt acceptance, the contents of the PC and the PSW are saved (the PSW is pushed first, followed by PC_H and PC_L). Therefore, a subroutine call occupies 2 bytes on the stack; an interrupt occupies 3 bytes.

When returning from the processing routine, executing a subroutine return instruction [RET] restores the contents to the PC from the stack; executing an interrupt return instruction [RETI] / [RETN] restores the contents to the PC and the PSW (the PC_L is popped first, followed by PC_H and PSW_H).

The stack can be normally located anywhere within 64K bytes memory space, that is assigned to data memory.

1.8.2 Stack Pointer (SP)

The stack pointer (SP) is a 16-bit register containing the address of the next free locations on the stack. The SP is post decremented when a subroutine call or a push instruction is executed, or when an interrupt is accepted; and the SP is preincremeted when a return or a pop instruction is executed. Figure 1-9 shows the stacking order.

The SP is not initialized hardware-wise but requires initialization by an initialize routine (sets the highest stack address). The value after power-on is undefined.

The SP is available for index register and also can be managed with data transfer or arithmetic and logical operation, similar to other index registers.

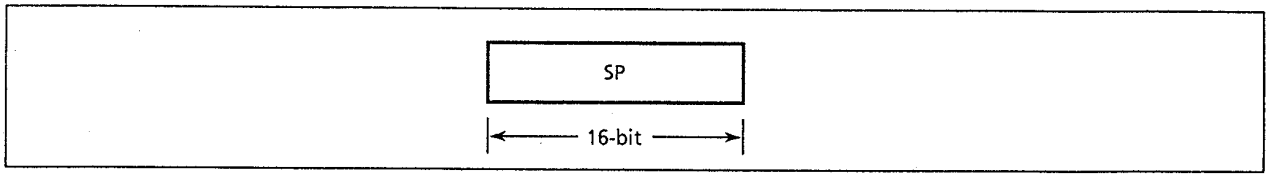


Figure 1-7. Stack Pointer

```
Example :  ① LD    SP,043FH      ;  SP ← 043F_H
           ② LD    HL,SP         ;  HL ← SP
           ③ LD    SP,SP + 04H   ;  SP ← SP + 04_H
           ④ ADD   SP,5678H      ;  SP ← SP + 5678_H
           ⑤ LD    A,(SP + 12H)  ;  A ← contents located on address (SP + 12_H)
```

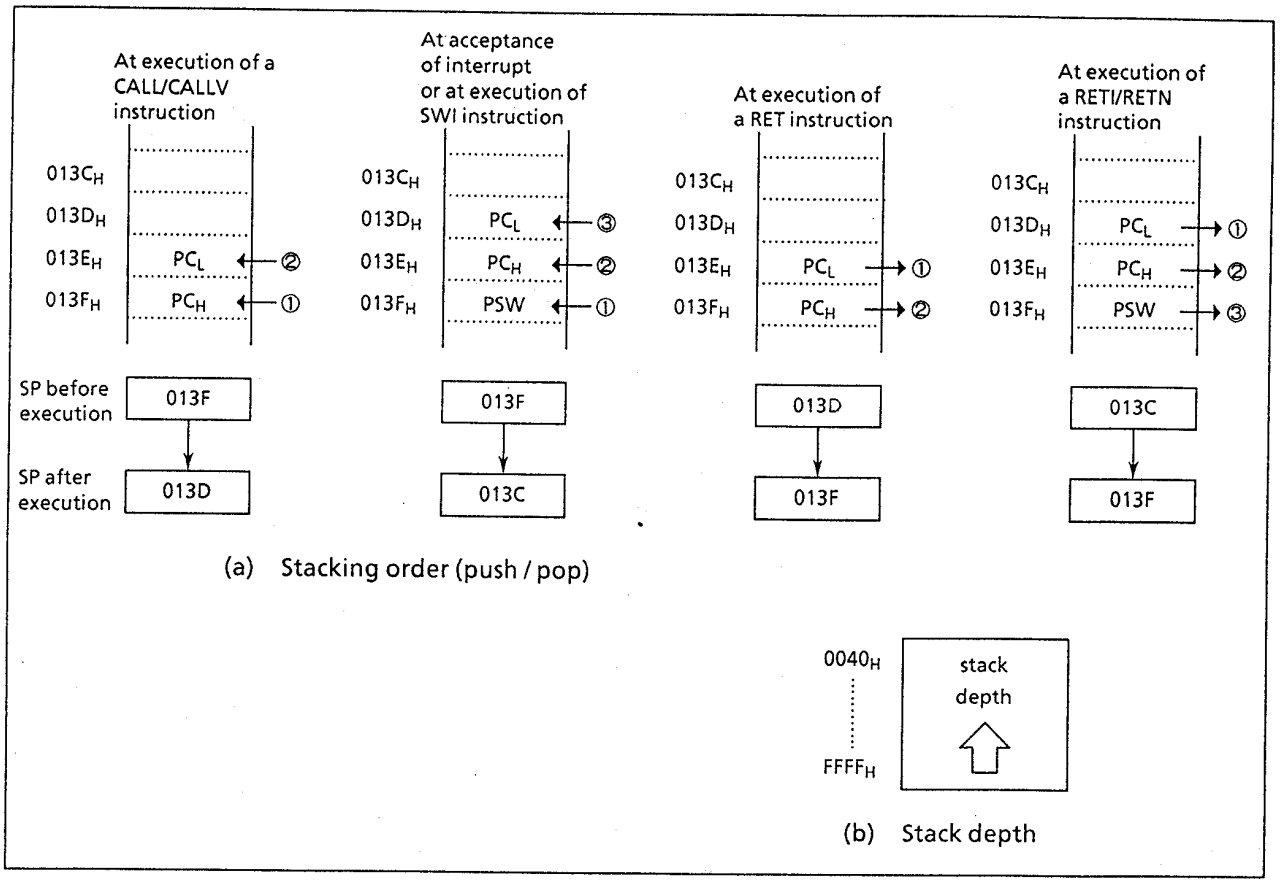


Figure 1-8. Stack

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 - - - - -	5	(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 1111 1001	wwww wwww nnnn nnnn	vvvv vvvv	1 - - - - -	6	(vw) ← n
	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 nnnn nnnn	dddd dddd	1111 1001	1 - - - - -	6	(IX+d) ← n
	LD (IY+d),n	0101 0101 nnnn nnnn	dddd dddd	1111 1001	1 - - - - -	6	(IY+d) ← n
	LD (SP+d),n	0101 0110 nnnn nnnn	dddd dddd	1111 1001	1 - - - - -	6	(SP+d) ← n
	LD (HL+d),n	0101 0111 nnnn nnnn	dddd dddd	1111 1001	1 - - - - -	6	(HL+d) ← n
	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 mmmm mmmm	xxxx xxxx	nnnn nnnn	1 - - - - -	6	(x+1,x) ← mn
	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 0111 0rrr	wwww wwww	vvvv vvvv	1 Z - - - -	6	r ⇔ (vw)
	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 1101 1rrr	wwww wwww	vvvv vvvv	1 - - - - -	8	rr ⇔ (vw+1,vw)
	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 nnnn nnnn	vvvv vvvv	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,gg	1110 1ggg	10rr r011		C Z C U S V	4	rr ← rr-gg
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)

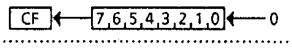
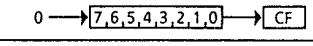
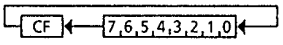
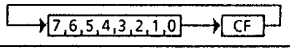
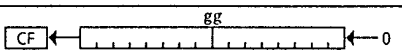
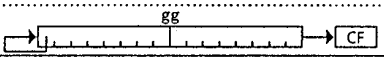
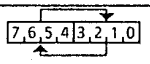
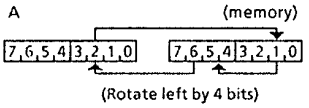
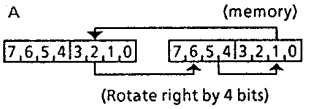
MNEMONIC	OBJECT CODE (Binary)			FLAG J Z C H S V	CYCLE	FUNCTION
SUBB A,n	0110 0010	nnnn nnnn		C Z C H S V	2	A ← A-n-CF
SUBB g,n	1110 1ggg	0110 0010	nnnn nnnn	C Z C H S V	3	g ← g-n-CF
SUBB gg,mn	1110 1ggg mmmm mmmm	0110 1010	nnnn nnnn	C Z C U S V	4	gg ← gg-mn-CF
SUBB r,g	1110 1ggg	00rr r010		C Z C H S 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 0110 0100	wwww wwww nnnn nnnn	vvvv vvvv	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 nnnn nnnn	dddd dddd	0110 0100	Z Z - - - -	7	(IX+d) ← (IX+d)∧n
AND (IY+d),n	1101 0101 nnnn nnnn	dddd dddd	0110 0100	Z Z - - - -	7	(IY+d) ← (IY+d)∧n
AND (SP+d),n	1101 0110 nnnn nnnn	dddd dddd	0110 0100	Z Z - - - -	7	(SP+d) ← (SP+d)∧n
AND (HL+d),n	1101 0111 nnnn nnnn	dddd dddd	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 10rr r100	wwww wwww	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 00rr r110	wwww wwww	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 0110 0110	wwww wwww nnnn nnnn	vvvv vvvv	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 nnnn nnnn	dddd dddd	0110 0110	Z Z - - - -	7	(IX+d) ← (IX+d)∨n
OR (IY+d),n	1101 0101 nnnn nnnn	dddd dddd	0110 0110	Z Z - - - -	7	(IY+d) ← (IY+d)∨n
OR (SP+d),n	1101 0110 nnnn nnnn	dddd dddd	0110 0110	Z Z - - - -	7	(SP+d) ← (SP+d)∨n
OR (HL+d),n	1101 0111 nnnn nnnn	dddd dddd	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 1111 0000	wwww wwww	vvvv vvvv	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 1111 1000	wwww wwww	vvvv vvvv	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	 (Rotate left by 4 bits) Note : (+SP) indicates that SP←SP + 1 is executed before rotation
ROLD A,(vw)	1110 0001 1111 0110	wwww wwww	vvvv vvvv	1 - - - - -	9	
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	 (Rotate right by 4 bits) Note : (+SP) indicates that SP←SP + 1 is executed before rotation
RORD A,(vw)	1110 0001 1111 0111	wwww wwww	vvvv vvvv	1 - - - - -	9	
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	

(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		$\overline{C}$ - * - - -	2	CF ← g.b
LD CF,(x).b	0101 1bbb	xxxx xxxx		$\overline{C}$ - * - - -	4	CF ← (x).b
LD CF,(vw).b	1110 0001 0101 1bbb	wwww wwww	vvvv vvvv	$\overline{C}$ - * - - -	5	CF ← (vw).b
LD CF,(DE).b	1110 0010	0101 1bbb		$\overline{C}$ - * - - -	3	CF ← (DE).b
LD CF,(HL).b	1110 0011	0101 1bbb		$\overline{C}$ - * - - -	3	CF ← (HL).b
LD CF,(IX).b	1110 0100	0101 1bbb		$\overline{C}$ - * - - -	3	CF ← (IX).b
LD CF,(IY).b	1110 0101	0101 1bbb		$\overline{C}$ - * - - -	3	CF ← (IY).b
LD CF,(IX+d).b	1101 0100	dddd dddd	0101 1bbb	$\overline{C}$ - * - - -	5	CF ← (:IX+d).b
LD CF,(IY+d).b	1101 0101	dddd dddd	0101 1bbb	$\overline{C}$ - * - - -	5	CF ← (IY+d).b
LD CF,(SP+d).b	1101 0110	dddd dddd	0101 1bbb	$\overline{C}$ - * - - -	5	CF ← (SP+d).b
LD CF,(HL+d).b	1101 0111	dddd dddd	0101 1bbb	$\overline{C}$ - * - - -	5	CF ← (HL+d).b
LD CF,(HL+C).b	1110 0111	0101 1bbb		$\overline{C}$ - * - - -	5	CF ← (HL+C).b
LD CF,(+SP).b	1110 0110	0101 1bbb		$\overline{C}$ - * - - -	4	SP ← SP+1:CF ← (SP).b
LD CF,(PC+A).b	0100 1111	0101 1bbb		$\overline{C}$ - * - - -	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	C - * - - -	4	CF ← (x).A
	LD CF,(vw).A	1110 0001	wwww wwww	vvvv vvvv	C - * - - -	5	CF ← (vw).A
		1111 1100					
	LD CF,(DE).A	1110 0010	1111 1100		C - * - - -	3	CF ← (DE).A
	LD CF,(HL).A	1110 0011	1111 1100		C - * - - -	3	CF ← (HL).A
	LD CF,(IX).A	1110 0100	1111 1100		C - * - - -	3	CF ← (IX).A
	LD CF,(IY).A	1110 0101	1111 1100		C - * - - -	3	CF ← (IY).A
	LD CF,(IX+d).A	1101 0100	dddd dddd	1111 1100	C - * - - -	5	CF ← (IX+d).A
	LD CF,(IY+d).A	1101 0101	dddd dddd	1111 1100	C - * - - -	5	CF ← (IY+d).A
	LD CF,(SP+d).A	1101 0110	dddd dddd	1111 1100	C - * - - -	5	CF ← (SP+d).A
	LD CF,(HL+d).A	1101 0111	dddd dddd	1111 1100	C - * - - -	5	CF ← (HL+d).A
	LD CF,(HL+C).A	1110 0111	1111 1100		C - * - - -	5	CF ← (HL+C).A
	LD CF,(+SP).A	1110 0110	1111 1100		C - * - - -	4	SP ← SP+1:CF ← (SP).A
	LD CF,(PC+A).A	0100 1111	1111 1100		C - * - - -	5	CF ← (PC+A).A
*2	TEST g.b	1110 1ggg	0101 1bbb		* - J - - -	2	JF ← $\overline{g.b}$
*2	TEST (x).b	0101 1bbb	xxxx xxxx		* - J - - -	4	JF ← $\overline{(x).b}$
*2	TEST (vw).b	1110 0001	wwww wwww	vvvv vvvv	* - J - - -	5	JF ← $\overline{(vw).b}$
		0101 1bbb					
*2	TEST (DE).b	1110 0010	0101 1bbb		* - J - - -	3	JF ← $\overline{(DE).b}$
*2	TEST (HL).b	1110 0011	0101 1bbb		* - J - - -	3	JF ← $\overline{(HL).b}$
*2	TEST (IX).b	1110 0100	0101 1bbb		* - J - - -	3	JF ← $\overline{(IX).b}$
*2	TEST (IY).b	1110 0101	0101 1bbb		* - J - - -	3	JF ← $\overline{(IY).b}$
*2	TEST (IX+d).b	1101 0100	dddd dddd	0101 1bbb	* - J - - -	5	JF ← $\overline{(IX+d).b}$
*2	TEST (IY+d).b	1101 0101	dddd dddd	0101 1bbb	* - J - - -	5	JF ← $\overline{(IY+d).b}$
*2	TEST (SP+d).b	1101 0110	dddd dddd	0101 1bbb	* - J - - -	5	JF ← $\overline{(SP+d).b}$
*2	TEST (HL+d).b	1101 0111	dddd dddd	0101 1bbb	* - J - - -	5	JF ← $\overline{(HL+d).b}$
*2	TEST (HL+C).b	1110 0111	0101 1bbb		* - J - - -	5	JF ← $\overline{(HL+C).b}$
*2	TEST (+SP).b	1110 0110	0101 1bbb		* - J - - -	4	SP ← SP+1:JF ← $\overline{(SP).b}$
*2	TEST (PC+A).b	0100 1111	0101 1bbb		* - J - - -	5	JF ← $\overline{(PC+A).b}$
*2	TEST (x).A	1110 0000	xxxx xxxx	1111 1100	* - J - - -	4	JF ← $\overline{(x).A}$
*2	TEST (vw).A	1110 0001	wwww wwww	vvvv vvvv	* - J - - -	5	JF ← $\overline{(vw).A}$
		1111 1100					
*2	TEST (DE).A	1110 0010	1111 1100		* - J - - -	3	JF ← $\overline{(DE).A}$
*2	TEST (HL).A	1110 0011	1111 1100		* - J - - -	3	JF ← $\overline{(HL).A}$
*2	TEST (IX).A	1110 0100	1111 1100		* - J - - -	3	JF ← $\overline{(IX).A}$
*2	TEST (IY).A	1110 0101	1111 1100		* - J - - -	3	JF ← $\overline{(IY).A}$
*2	TEST (IX+d).A	1101 0100	dddd dddd	1111 1100	* - J - - -	5	JF ← $\overline{(IX+d).A}$
*2	TEST (IY+d).A	1101 0101	dddd dddd	1111 1100	* - J - - -	5	JF ← $\overline{(IY+d).A}$
*2	TEST (SP+d).A	1101 0110	dddd dddd	1111 1100	* - J - - -	5	JF ← $\overline{(SP+d).A}$
*2	TEST (HL+d).A	1101 0111	dddd dddd	1111 1100	* - J - - -	5	JF ← $\overline{(HL+d).A}$
*2	TEST (HL+C).A	1110 0111	1111 1100		* - J - - -	5	JF ← $\overline{(HL+C).A}$
*2	TEST (+SP).A	1110 0110	1111 1100		* - J - - -	4	SP ← SP+1:JF ← $\overline{(SP).A}$
*2	TEST (PC+A).A	0100 1111	1111 1100		* - 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 1011	1 - - - - -	5	(x).A ← CF
	LD (vw).A,CF	1110 0001	wwww wwww	vvvv vvvv	1 - - - - -	6	(vw).A ← CF
		1111 1011					
	LD (DE).A,CF	1110 0010	1111 1011		1 - - - - -	4	(DE).A ← CF
	LD (HL).A,CF	1110 0011	1111 1011		1 - - - - -	4	(HL).A ← CF
	LD (IX).A,CF	1110 0100	1111 1011		1 - - - - -	4	(IX).A ← CF
	LD (IY).A,CF	1110 0101	1111 1011		1 - - - - -	4	(IY).A ← CF
	LD (IX+d).A,CF	1101 0100	dddd dddd	1111 1011	1 - - - - -	6	(IX+d).A ← CF
	LD (IY+d).A,CF	1101 0101	dddd dddd	1111 1011	1 - - - - -	6	(IY+d).A ← CF
	LD (SP+d).A,CF	1101 0110	dddd dddd	1111 1011	1 - - - - -	6	(SP+d).A ← CF
	LD (HL+d).A,CF	1101 0111	dddd dddd	1111 1011	1 - - - - -	6	(HL+d).A ← CF
	LD (HL+C).A,CF	1110 0111	1111 1011		1 - - - - -	6	(HL+C).A ← CF
	LD (+SP).A,CF	1110 0110	1111 1011		1 - - - - -	5	SP ← SP+1:(SP).A ← CF
	LD (PC+A).A,CF	0100 1111	1111 1011		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,$+2+d	1101 1000	dddd dddd		1 - - - - -	4/2(t/f)	if ZF=1 then PC ← PC+d else null
JR Z,$+2+d						
JR NE,$+2+d	1101 1001	dddd dddd		1 - - - - -	4/2(t/f)	if ZF=0 then PC ← PC+d else null
JR NZ,$+2+d						
JR CS,$+2+d	1101 1010	dddd dddd		1 - - - - -	4/2(t/f)	if CF=1 then PC ← PC+d else null
JR LT,$+2+d						
JR CC,$+2+d	1101 1011	dddd dddd		1 - - - - -	4/2(t/f)	if CF=0 then PC ← PC+d else null
JR GE,$+2+d						
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

r, g : 8-bit register (see table 1)
rr, gg : 16-bit register (see table 2)
b : bit number (see table 3)
n : 4-bit or 8-bit immediate data
mn : 16-bit immediate data
d : 5-bit or 8-bit displacement
x : 8-bit direct address (0000H to 00FFH)
vw : 16-bit direct address (0000H to FFFFH)
$: Instruction start address

Table 1

register	r/g
A	000
W	001
C	010
B	011
E	100
D	101
L	110
H	111

Table 2

register pair	rr/gg
WA	000
BC	001
DE	010
HL	011
IX	100
IY	101
SP	110

Table 3

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

TLCS-870/C Instruction Set

1 st OpCode

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

2 nd OpCode (mem)

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