MF658-04

EPSON

CMOS 8-BIT SINGLE CHIP MICROCOMPUTER **E0C88 Family**

E0C88 CORE CPU MANUAL

ENERGY SAVING EPSON

SEIKO EPSON CORPORATION

E0C88 Core CPU Manual

PREFACE

This manual explains the architecture, operation and instruction of the core CPU E0C88 of the CMOS 8-bit single chip microcomputer E0C88 Family.
Also, since the memory configuration and the peripheral circuit configuration is different for each device of the E0C88 Family, you should refer to the respective manuals for specific details other than the basic functions.

CONTENTS

1 OUTLINE

The E0C88 is the core CPU of the 8-bit single chip microcomputer E0C88 Family that utilizes original EPSON architecture. It has a maximum 16M bytes address space and high speed, abundant instruction sets. It handles a wide range of operating voltages and features low power consumption. In addition, it has adopted a unified architecture and a peripheral circuit interface for its memory mapped I/O mode to flexibly meet future expansion of the E0C88 Family.

1.1 Features

The E0C88 boasts the below features.

Address space	Maximum 16M bytes
Instruction cycle	1–15 cycles (1 cycle = 2 clocks)
Instruction set	608 types
Register configuration	Data registers 2
	Index registers 3
	(One is used as a data register)
	Program counter
	Stack pointer
	System condition flag
	Customize condition flag
Exception processing factors	Reset, zero division and interrupt
Exception processing vectors	Maximum 128 vectors
Standby function	HALT/SLEEP
Peripheral circuit interface	Memory mapped I/O system

1.2 Instruction Set Features

(1) It adopts high efficiency machine cycle plus high speed and abundant instruction sets.
(2) Memory management can be done easily by 12 types of addressing modes.
(3) It has effective 16 bit operation functions including address calculation.
(4) It includes powerful decimal operation functions such as a decimal operation mode and pack/unpack instruction.
(5) It supports the realization of various types of special service microcomputers through customized flag instructions.
(6) It is composed of an instruction system that enables relocatable programming, thus permitting easy development of software libraries.

1.3 Block Diagram

Figure 1.3.1 shows the E0C88 block diagram.

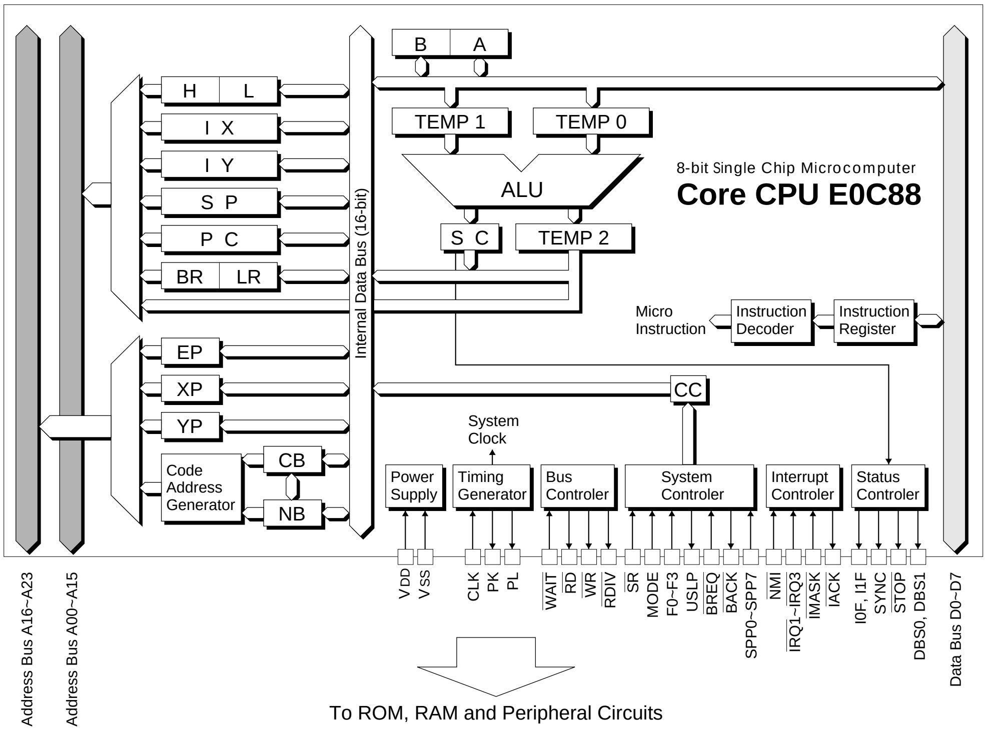
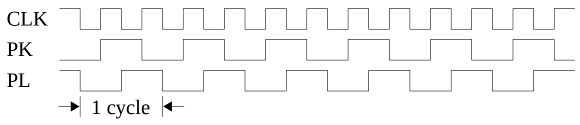


Fig. 1.3.1
E0C88 block diagram

1.4 Input-Output Signal

Tables 1.4.1 (a) and 1.4.1 (b) show the input/output signals between the E0C88 and the peripheral circuits.

Table 1.4.1(a) Input/output signal list (1)

Type	Name	Signal name	I/O	Function
Power	Power	V_{DD}	I	Inputs the + side power.
	Ground	V_{SS}	I	Inputs the - side power (GND).
Clock	Clock input	CLK	I	Inputs the system clock from the peripheral circuit.
	Clock output	PK PL	O	Outputs the two phase divided signals to be generated from the system clock input to the CLK terminal as following phase. 
Address bus	Address bus	A00–A23	O	A 24-bit address bus.
Data bus	Data bus	D0–D7	I/O	An 8-bit bidirectional data bus.
Bus control signal	Wait	$\overline{WAIT}$	I	Controls the wait state insertion for the access time extension during memory access. This control will become valid with LOW level input.
	Read	$\overline{RD}$	O	A memory (and peripheral circuit) read signal. It shifts to LOW level during readout.
	Write	$\overline{WR}$	O	A memory (and peripheral circuit) write signal. It shifts to LOW level during writing.
	Read interrupt vector address	$\overline{RDIV}$	O	An interrupt vector address read signal. It shifts to LOW level during readout of the vector address.
System control signal	Reset	$\overline{SR}$	I	A LOW level input shifts the CPU into the reset status.
	Mode setting	MODE	I	Sets the CPU operation mode by means of the peripheral circuit. LOW: Minimum mode HIGH: Maximum mode
	Customize condition flag	F0–F3	I	A status signal input by a peripheral circuit. The meaning of the signal differs depending on the peripheral circuit.
	Micro sleep	USLP	O	The USLP is set to HIGH level 1 cycle prior to the CPU's entry into the SLEEP status as a result of the SLP (SLEEP) instruction. The peripheral circuit controls the oscillation stop based on this signal.
	Bus authority request	$\overline{BREQ}$	I	This is the bus authority request signal when the peripheral circuit makes a DMA transmission. LOW level input to this terminal causes the CPU to release bus. The address bus, data bus and read/write signal shift to the high impedance status.
	Bus authority acknowledge	$\overline{BACK}$	O	This is response signal that indicates a bus authorization has been released to the peripheral circuit. It shifts to LOW level when bus authorization has been released.
	Stack pointer page	SPP0–SPP7	I	This is a page address of the stack pointer that is specified by the peripheral circuit. When the stack pointer accesses the memory, this address is output to the page section (AD16–AD23) of the address bus.

Refer to Chapter 3, "CPU OPERATION AND PROCESSING STATUSES" for the timing of each signal and related information.

Table 1.4.1(b) Input/output signal list (2)

Type	Name	Signal name	I/O	Function
Interrupt signal	Non-maskable interrupt	$\overline{\text{NMI}}$	I	This is an interrupt signal not permitting masking by the software. The input is sensed at the falling edge.
	Interrupt request 3	$\overline{\text{IRQ3}}$	I	This is an interrupt signal permitting masking by the software. The interrupt priority is level 3 and the input is sensed at a LOW level.
	Interrupt request 2	$\overline{\text{IRQ2}}$	I	This is an interrupt signal permitting masking by the software. The interrupt priority is level 2 and the input is sensed at a LOW level.
	Interrupt request 1	$\overline{\text{IRQ1}}$	I	This is an interrupt signal permitting masking by the software. The interrupt priority is level 1 and the input is sensed at a LOW level.
	Interrupt mask	$\overline{\text{IMASK}}$	I	This is an interrupt mask signal input by the peripheral circuit. When the page section, etc. of the stack pointer configured on the peripheral circuit section is accessed, LOW level is input to this terminal and the below interrupt is masked. $\overline{\text{NMI}}$, $\overline{\text{IRQ3}}$, $\overline{\text{IRQ2}}$, $\overline{\text{IRQ1}}$
	Interrupt acknowledge	$\overline{\text{IACK}}$	O	This is a response signal that indicates that an interrupt request has been received. It shifts to LOW level when an interrupt has been received. The peripheral circuit receives this signal and holds the vector address. This signal also shifts to LOW level when exceptional processing is executed by reset and zero division.
	Interrupt flag	I0F I1F	O	A status of the interrupt flag (I0, I1) in the system condition flag (SC) is output.
Status signal	First operation code fetch signal	SYNC	O	This is a signal that becomes active when the CPU fetches the first operation code. It shifts to HIGH level during the bus cycle of the first operation code fetch. The interrupt is sampled at the rising edge of this signal.
	Stop signal	$\overline{\text{STOP}}$	O	This is a signal that becomes low level when the CPU shifts into the following status: • CPU stops by HALT instruction • CPU stops by SLP instruction • The bus authorization has been released by LOW level input to the $\overline{\text{BREQ}}$ terminal.
	Data bus status	DBS0 DBS1	O	This is a 2 bit status signal that indicates the data bus status as follows. DSB1 DSB0 State 0 0 High impedance 0 1 Interrupt vector address read 1 0 Memory write 1 1 Memory read

Note: Input/output signals may differ from the above table, for example, a peripheral circuit signal may be added by each device of the E0C88 Family.

2 ARCHITECTURE

The E0C88 has a maximum 16M bytes address space and can thus respond to large scale applications. Here we will explain such points as this address space and memory control as well as the configuration of the registers.

2.1 Address Space and CPU Model

CPU models of the four types MODEL0 to MODEL3 are set in the E0C88 according to the size of the address space and whether or not there is a multiplication/division instruction. The differences in each model are as shown in Table 2.1.1 and have been designed to permit selection according to the microcomputer service and the scope of the application as per Table 2.1.1.

Either the minimum mode that makes the programming field a maximum 64K bytes or the maximum mode that makes it a maximum 8M bytes for MODEL2 and MODEL3 can be selected, depending on the MODE terminal setting of CPU. Figure 2.1.1 shows the memory map concept for each CPU model.

The program memory is managed by dividing the bank for each 32K bytes and the data memory into one page for each 64K bytes.

See "2.3 Program Memory" and "2.4 Data Memory".

Note: The memory configuration varies for the respective devices of the E0C88 Family. Refer to the manual for each device.

Table 2.1.1 CPU model

CPU model	Address space	Multiplication division instruction
MODEL0	64K bytes	Not available
MODEL1	64K bytes	Available
MODEL2	16M bytes	Not available
MODEL3	16M bytes	Available

Table 2.1.2 Setting of the operation mode (MODEL2/3)

MODE	Operation mode	Programming area
0	Minimum mode	Maximum 64K bytes
1	Maximum mode	Maximum 8M bytes

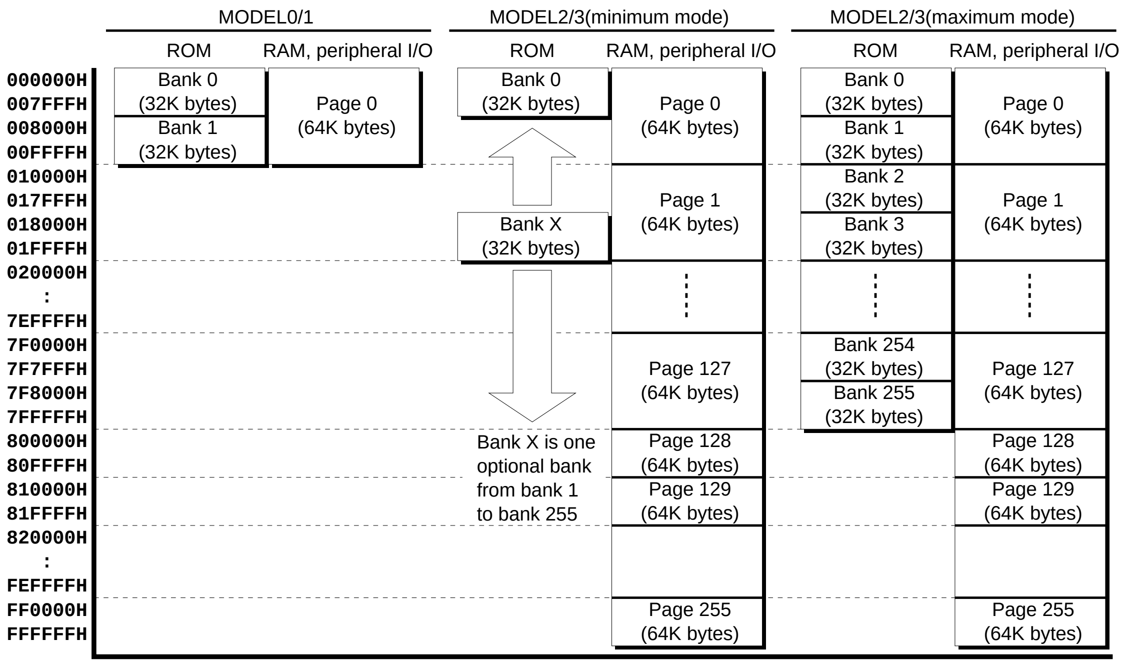


Fig. 2.1.1 Memory map

2.2 ALU and Registers

2.2.1 ALU

The ALU (arithmetic and logic unit) performs the operation between the 8-bit and the 16-bit data stored in the two types of temporary registers TEMP 0 and TEMP 1. The ALU functions are as indicated in Table 2.2.1.1.

After having been stored in the 16-bit temporary register TEMP 2, the operation result is either stored in the register/memory or used as address data according to the operation instruction.
In addition, the Z (zero) flag, C (carry) flag V (overflow) flag and N (negative) flag are set/reset according to the operation result.

See "2.2.3 Flags".

Table 2.2.1.1 ALU operation functions

Arithmetic function	Arithmetic instruction	16-bit operation
Addition	ADD, ADC	○
Subtraction	SUB, SBC	○
Logical product	AND	
Logical sum	OR	
Exclusive OR	XOR	
Comparison	CP	○
Bit test	BIT	
Increment/decrement	INC, DEC	○
Multiplication	MLT	
Division	DIV	
Compliment	CPL, NEG	
Rotate	RL, RLC, RR, RRC	
Shift	SLA, SLL, SRA, SRL	
Pack/unpack	PACK, UPCK	
Code extension	SEP	

2.2.2 Register configuration

Figure 2.2.2.1 shows the register configuration of the E0C88.

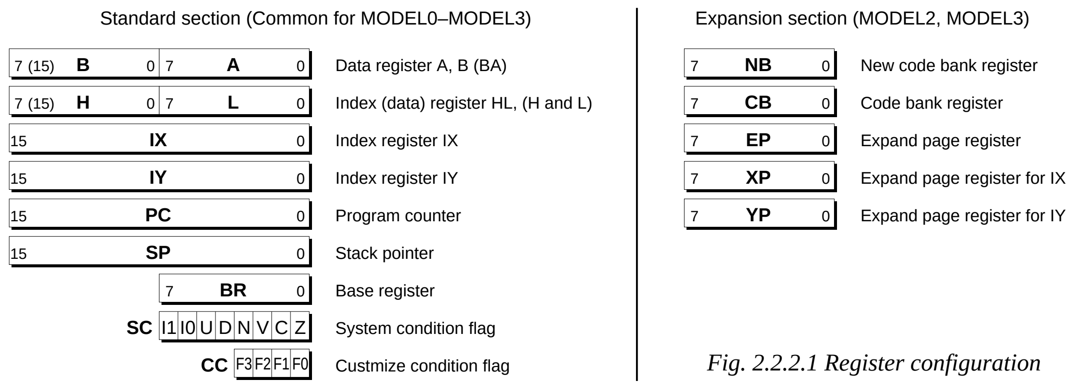


Fig. 2.2.2.1 Register configuration

A and B registers

The A and B registers are respective 8-bit data registers and they perform data transfer and operation with other registers and/or data memories, transfer of immediate data and operations. They are respectively used independently for 8-bit transfer/operations and used in a BA pair that makes the B register the upper 8-bit for 16-bit transfer/operations.

HL register

The HL register is a 16-bit index register that is used for indirect addressing of the data memory (specification of the address within the page). It performs 16-bit data transfer and operations with other registers and/or memories.

It can also be used as a data register by splitting it into respective 8-bit H and L registers. In this case, the L register can also be used as a displacement at the time of indirect addressing by the IX and IY registers.

See "2.4 Data Memory".
See "4.1 Addressing Mode".

IX and IY registers

The IX and IY registers are respective 16-bit index registers that are used for indirect addressing of the data memory (specification of the address within the page). They perform 16-bit data transfer and/or operations with other registers and/or data memories.

See "2.4 Data Memory".
See "4.1 Addressing Mode".

PC (Program Counter)

The PC is a 16-bit counter register that does the addressing of the program memory and it indicates the following address to be executed.

See "2.3 Program Memory".

SP (Stack Pointer)

The SP is a 16-bit counter register that indicates the stack address (address within the stack page). It performs 16-bit data transfer and/or operations with the other registers and/or data memories.

See "2.4.3 Stack".

BR (Base Register)

The BR is an 8-bit index register and is used for upper 8-bit address specification within the page at the time of 8-bit absolute addressing (specifies the lower 8 bits with immediate data.).

See "4.1 Addressing Mode".

SC (System Condition Flag)

The SC is an 8-bit flag and is configured with Z, C, V and Z flags that indicate the operation result, D and U flags that set the operation mode, and I0 and I1 flags that set the interrupt priority level.

See "2.2.3 Flags".

CC (Customize Condition Flag)

The CC is a 4-bit flag that indicates the various types of statuses that are selected by the peripheral circuit. It is set/reset by the peripheral circuit and is used as a branch instruction condition.

See "2.2.3 Flags".

NB (New Code Bank Register)

The NB register is an 8-bit register that specifies the program memory bank. The NB register is set for the CPU models MODEL2 and MODEL3.

See "2.3 Program Memory".

CB (Code Bank Register)

The CB register is an 8-bit register that indicates the currently selected bank of the program memory. When the data has been set to the NB register, the data is loaded into the CB register and a new bank is selected.
The CB register is set for the CPU models MODEL2 and MODEL3.

See "2.3 Program Memory".

EP, XP and YP (Expand Page Registers)

These registers are 8-bit registers that specify the data memory page.
The EP register is used at the time of indirect addressing by the HL register or absolute addressing by the immediate data.
The XP register and YP register are used at the time of indirect addressing by the IX register or indirect addressing by the IY register, respectively.
These registers are set by the CPU models MODEL2 and MODEL3.

See "2.4.2 Page registers EP, XP, YP".
See "4.1 Addressing Mode".

2.2.3 *Flags*

The system condition flag (SC) that indicates such things as the operation result status within the CPU and the customize condition flag (CC) that indicates the peripheral circuit status are set for the E0C88.

System condition flag (SC)

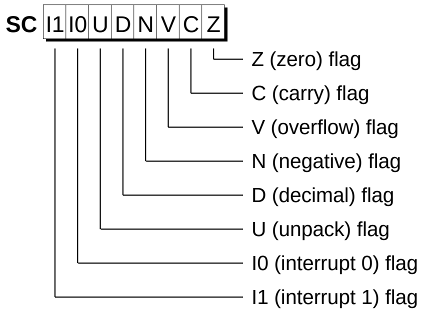


Fig. 2.2.3.1 System condition flag

Figure 2.2.3.1 indicates the system condition flags and is composed of the register SC that is configured by an 8-bit flag.
The system condition flags Z (zero), C (carrier), V (overflow) and N (negative) flags are set/reset according to the operation results and the I0 and I1 (interrupt) flags are set/reset by the interrupt.
These flags can also be operated by the below instructions.

AND	SC,#nn	(Resets the optional flag)
OR	SC,#nn	(Sets the optional flag)
XOR	SC,#nn	(Inverts the optional flag)
LD	SC,#nn	(Flag write)
LD	SC,A	(Flag write)
POP	SC	(Flag return)
RETE		(Flag evacuation)

The Z, C, N and V flags are used for condition judgments at the time of conditional jump/call instruction execution for JRS instructions and/or CARS instructions.

See "4.4 Detailed Explanation of Instructions".

Here following the respective flags are explained.

(1) Z (zero) flag

The Z flag is set to '1' when the arithmetic instruction execution result has become '0' and is set at '0' when the result is other than '0'.

(2) C (carry) flag

When a carry (carry from the most significant bit) has been generated by the execution of an addition instruction, or when a borrow (borrow to the most significant bit) has been generated by the execution of an addition instruction/comparison instruction, the C flag is set to '1' and otherwise is set to '0'. However, the C flag will not vary depending on the execution of an 1 addition-subtraction instruction (INC and DEC instructions).
The C flag also varies according to the execution of the rotate/shift instruction.
It is reset to '0' when multiplication-division instructions (MLT and DIV instructions) have been executed.

(3) V (overflow) flag

The V flag is set to '1' when the result of the operation exceeds the range of the complementary representation by 8 bits or 16 bits and is reset to '0', when it is within the range. 8 bits become
-128–127 for the range of the complimentary representation and 16 bits become -32768–32767.
However, the V flag will not change according to the execution of an logic operation instruction (AND, OR and XOR instructions, excluding cases where the destination is SC) and a 1 increment-decrement instruction (INC and DEC instructions) even within an operation instruction. When a multiplication instruction (MLT instruction) has been executed, it is reset to '0'. When a division instruction (DIV instruction) has been executed, it is set to '1' when the quotient is exceeded the 8-bit data range.
The V flag indicates the overflow of a complementary operation, in contrast to the fact that the C flag indicates an over (under) flow of an absolute value operation.
When performing a complimentary operation that is likely to overflow, the V flag must be checked and the operation result corrected when it is '1'.

See "2.2.4 Complimentary operation and overflow".

(4) N (negative) flag

When the result of a performed operation is minus (The most significant bit is '1'), N flag is set to '1' and when it is plus (The most significant bit is '0'), N flag is reset to '0'. However, the N flag does not change according to the execution of an 1 increment-decrement instruction (INC and DEC instructions).

(5) D (decimal) flag

The D flag is the bit that sets the CPU such that it performs a decimal operation (The operation result is decimal corrected) at the time of execution of an 8-bit addition subtraction instruction. Setting it to '1' causes it to perform a decimal operation and it performs a hexadecimal operation at '0'.

See "2.2.5 Decimal operation and unpack operation".

(6) U (unpack) flag

The U flag is the bit that sets the CPU such that it performs an unpack operation (executes the operation for the upper 4 bits as '0') upon execution of an 8-bit addition-subtraction operation. Setting it to '1' causes it to perform a unpack operation and it performs a 8-bit operation at '0'.

See "2.2.5 Decimal operation and unpack operation".

(7) I0 and I1 (interrupt) flags

The I0 and I1 flags are the bits that set the interrupt priority level. The CPU accepts interrupts that are set higher level than interrupt priority level set with these two bits. Also when an interrupt is generated, it is automatically set to new value that it will mask the interrupts for that level and below.

See "3.5.3 Interrupts".

We have indicated the flags that change due to execution of an instruction by "↕" in the instruction set lists and other documents. The D and U flags have a "★" attached to them, indicating that those instructions permit decimal operations and unpack operations.

Customize condition flag (CC)

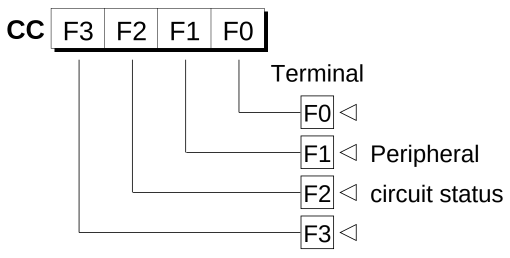


Fig. 2.2.3.2 Customize condition flag

The customize condition flags are as shown in Figure 2.2.3.2 and consist of the CC registers that are made up of 4-bit flags.
Each of the CC flags consist of the names F0–F3 and vary according to the signals that are input to the F0–F3 terminals of the E0C88 from the peripheral circuit. Since the signal indicating the status of the peripheral circuit is input here, the program can be branched according to the status of the peripheral circuit reflected for each flag.
The E0C88 has been conceived to permit special purpose microcomputers to be created easily.
The CC flag is used for condition judgment at the time of a conditional jump/call instruction execution of a JRS instruction and/or a CARS instruction.

See "4.4 Detailed Explanation of Instructions".

2.2.4 Complimentary operation and overflow

Complementary representations are used within the E0C88 for the handling of minus data. Here following we will explain about operations using complimentary expressions and compliments.

Compliments

When a minus number is handled by the microcomputer a complimentary representation is generally used. Compliments contain two types of expressions, 1 compliment and 2 compliment type. Normally when referring simply to a compliment the 2 compliment type is indicated. In the E0C88 as well, a minus number is expressed by <u>2 compliments</u>.

Compliments of the optional number N are expressed by the following expression and the range where a 2 compliment representation is permissible is -128–127 in the case of 8 bits and -32768–32767 in the case of 16 bits. The range where a 1 compliment representation is possible is -127–127 in the case of 8 bits and -32767–32767 in the case of 16 bits.
When an complement representation is used, the most superior bit of the minus number must absolutely become '1', the content of the most superior bit is reflected in the N (negative) flag.

In addition, the "CPL" instruction (conversion to 1 compliment) and a "NEG" instruction (conversion to 2 compliments) are prepared for conversion of 8 bits data to compliment. The "SEP" instruction is prepared for expanding the 8 bit compliment to 16 bits.

Example: NEG instruction and SEP instruction

Instruction		B reg.	A reg.	N flag
LD	A,#127	0000 0000	0111 1111	0
NEG	A	0000 0000	1000 0001	1
SEP		1111 1111	1000 0001	1

2 compliment

8-bit $-N = 2^8 - N = 256 - N$

127	= 0111 1111b
126	= 0111 1110b
	:
2	= 0000 0010b
1	= 0000 0001b
0	= 0000 0000b
-1	= 1111 1111b (= 1 0000 0000b - 0000 0001b)
-2	= 1111 1110b (= 1 0000 0000b - 0000 0010b)
	:
-127	= 1000 0001b (= 1 0000 0000b - 0111 1111b)
-128	= 1000 0000b (= 1 0000 0000b - 1000 0000b)

16-bit $-N = 2^{16} - N = 65536 - N$

32767	= 0111 1111 1111 1111b
32766	= 0111 1111 1111 1110b
	:
2	= 0000 0000 0000 0010b
1	= 0000 0000 0000 0001b
0	= 0000 0000 0000 0000b
-1	= 1111 1111 1111 1111b
	(= 1 0000 0000 0000 0000b - 0000 0000 0000 0001b)
-2	= 1111 1111 1111 1110b
	(= 1 0000 0000 0000 0000b - 0000 0000 0000 0010b)
	:
-32767	= 1000 0000 0000 0001b
	(= 1 0000 0000 0000 0000b - 0111 1111 1111 1111b)
-32768	= 1000 0000 0000 0000b
	(= 1 0000 0000 0000 0000b - 1000 0000 0000 0000b)

1 compliment

8-bit $-N = 2^8 - 1 - N = 255 - N\ (= \overline{N})$

127	= 0111 1111b
126	= 0111 1110b
	:
2	= 0000 0010b
1	= 0000 0001b
0	= 0000 0000b
-1	= 1111 1110b (= 1111 1111b - 0000 0001b)
-2	= 1111 1101b (= 1111 1111b - 0000 0010b)
	:
-126	= 1000 0001b (= 1111 1111b - 0111 1110b)
-127	= 1000 0000b (= 1111 1111b - 0111 1111b)

16-bit $-N = 2^{16} - 1 - N = 65535 - N\ (= \overline{N})$

327677	= 0111 1111 1111 1111b
327666	= 0111 1111 1111 1110b
	:
2	= 0000 0000 0000 0010b
1	= 0000 0000 0000 0001b
0	= 0000 0000 0000 0000b
-1	= 1111 1111 1111 1110b
	(= 1111 1111 1111 1111b - 0000 0000 0000 0001b)
-2	= 1111 1111 1111 1101b
	(= 1111 11111111 1111b - 0000 0000 0000 0010b)
	:
-32766	= 1000 0000 0000 0001b
	(= 1111 11111111 1111b - 0111 1111 1111 1110b)
-32767	= 1000 0000 0000 0000b
	(= 1111 11111111 1111b - 0111 1111 1111 1111b)

Compliment expression and V (overflow) flag

In the case of an operation by an absolute value such as an address operation, a correct operation result is obtained in the range of 0–255 with 8 bits and in the range of 0–65535 with 16 bits. When an overflow or an underflow has occurred due to an operation and it misses the range, the C (carrier) flag is set to '1'.
The correct operation result range when the operands have become compliments is -128–127 for 8 bits and -32768–32767 for 16 bits and whether operation result is correct or not cannot only be judged by the C flag. To perform this judgment, the V (overflow) flag is set and the V flag is set to '1', when it has exceeded the compliment representation range.

Since the ALU does not differentiate absolute operations and complementary operations, the setting/resetting of the C flag and V flag is done by whether or not the operation result is within the above mentioned range. Consequently, when the V flag may also be set to '1' for absolute value operations.
Since in this case the V flag has no meaning, the V flag must not be verified by the program. Since only a complimentary operation can judge an overflow by the V flag, you should judge it by whether or not the data handled by the application has an attached code.
Here following are indicated examples of 8-bit operations and the changes of the V and C flags resulting from their operation results.

Example:

Addition example (ADD A,B)

A reg.	B reg.	Result (A reg.)	V flag	C flag
0101 1010	1010 0101	1111 1111	0	0
0101 1011	1010 0101	0000 0000	0	1
0101 1011	0010 0101	1000 0000	1	0

Subtraction example (SUB A,B)

A reg.	B reg.	Result (A reg.)	V flag	C flag
0101 1010	0101 1010	0000 0000	0	0
0101 1010	0101 1011	1111 1111	0	1
0101 1010	1101 1010	1000 0000	1	1

2.2.5 Decimal operation and unpack operation

When executing the below 8-bit arithmetic instructions on the E0C88, you can set it to perform decimal operations in addition to the normal hexadecimal operations, unpack operations and operations by combinations of these. These settings are done by the D (decimal) flag and the U (unpack) flag.

Arithmetic instructions permitting 10 decimal and unpack operations

ADD, ADC, SUB, SBC, NEG

They are all 8-bit arithmetic instructions and attaching a "★" to the D flag and U flag sections in the instruction set list indicates that a decimal operation and unpack operation is possible.

Decimal operation

When the arithmetic instruction (ADD, ADC, SUB, SBC or NEG) has been executed in the status where the D flag is set to '1', a decimal operation can be done. The operation result is obtained by the BCD (binary-coded decimal) code.
When a decimal operation is done, an "OR SC,#00010000B" or similar instruction sets the D flag to '1' and the operands to BCD code prior to execution the arithmetic instruction.
When the operands are not in BCD code, the correct result may sometimes not be obtained.

• SC flag at the time of a decimal operation
Following execution of the decimal operation, the N/V/C/Z flags of the SC are set according to the operation result, as shown below.

N:	Always	Reset (0)
V:	Always	Reset (0)
C:	When there has been a carry from the 2-digit decimal value or a borrow to the 2-digit decimal value	Set (1)
	When there has not been	Reset (0)
Z:	When the operation result = 0	Set (1)
	When the operation result ≠ 0	Reset (0)

Examples:

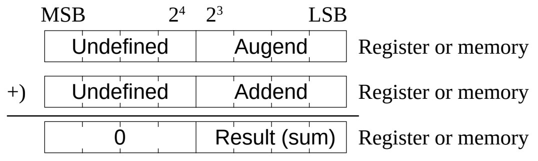
Instruction	Setting value		Result	SC			
	A reg.	B reg.	A reg.	N	V	C	Z
ADD A,B	55	28	83	0	0	0	0
ADD A,B	74	98	72	0	0	1	0
SUB A,B	55	55	00	0	0	0	1
SUB A,B	55	28	27	0	0	0	0
SUB A,B	74	98	76	0	0	1	0

bits of operands are disregarded (considered as '0) and the operation for the lower 4 bits alone is done.

Unpack operation

When executing an 8-bit arithmetic (ADD, ADC, SUB, SBC, NEG) instruction by setting the U flag to '1', you can perform the operation in the below indicated unpack format.
The unpack operation disregards the upper 4-bit data and performs the operation for the lower 4 bits alone. After execution, only the operation results for the lower 4 bits are output and '0' is output for the upper 4 bits.
Since the unpack operation stores 1 digit of data for the memory address, the digit matching of the operand can be done easily. (The digit matching in this case, becomes memory address pointing alone.)

<Example of ADD instruction>



• SC flag at the time of an unpack operation
Since an unpack operation is only affects the lower 4-bit data, the SC flag also changes according to the operation result for the lower 4 bits. Following execution of the unpack operation, the N/V/C/Z flags of the SC are set according to the operation result, as shown below.

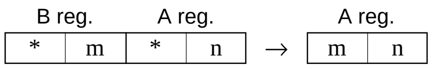
N:	When the 2^3 bit is '1'	Set (1)
	When the 2^3 bit is '0'	Reset (0)
V:	When it exceeds the 4-bit complementary range (-8 to 7)	Set (1)
	When it is within the range	Reset (0)
C:	When there has been a carry from the 2^3 bit and a borrow to the 2^3 bit	Set (1)
	When there has not been	Reset (0)
Z:	When the lower 4 bits = 0	Set (1)
	When the lower 4 bits ≠ 0	Reset (0)

Example: ADD A,B

Setting value		Result	SC			
A reg.	B reg.	A reg.	N	V	C	Z
20H	D0H	00H	0	0	0	1
2EH	53H	01H	0	0	1	0
C7H	52H	09H	1	1	0	0

• Auxiliary unpack operation instruction
"PACK" and "UPCK" instructions have been prepared that mutually convert the unpack format and the pack format (normal 8-bit data format), permitting easy format conversion.

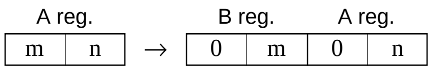
***PACK** instruction:*
Converts the unpack format data of the BA register into pack format and stores it in the A register.



Example: PACK instruction

Setting value	Result	SC
BA reg.	A reg.	N V C Z
38C4H	84H	Unchanged

***UPCK** instruction:*
Converts the 8-bit data of the A register into unpack format and stores it in the BA register.



Example: UPCK instruction

Setting value	Result	SC
A reg.	BA reg.	N V C Z
84H	0804H	Unchanged

2.2.6 Multiplication and division

The E0C88 MODEL1 and MODEL3 possess multiplication and division functions. In MODEL0 and MODEL2, these functions and the multiplication/division instructions explained below cannot be used.

Multiplication

Multiplication is done using the MLT instruction. When executing an MLT instruction, a L register × A register operation is performed and the product is stored in the HL register. The N/V/C/Z flags of the SC are set as following according to this operation result.

N:	When the MSB of the HL register (product) is '1'	Set (1)
	When it is '0'	Reset (0)
V:	Always	Reset (0)
C:	Always	Reset (0)
Z:	When the HL register (product) is 0000H	Set (1)
	When other than 0000H	Reset (0)

Here below are shown execution examples of the MLT instruction.

Example: (Result: HL reg. = product)

Setting value		Result	SC			
L reg.	A reg.	HL reg.	N	V	C	Z
00H	64H	0000H	0	0	0	1
64H	58H	2260H	0	0	0	0
C8H	58H	44C0H	0	0	0	0
A5H	93H	5EBFH	0	0	0	0
C8H	A5H	80E8H	1	0	0	0

Since multiplication handles the above set value as 8-bit data without a sign and an operation without a sign is executed, the N flag that is set according to the operation result does not indicate a sign. Consequently, even when negative number are multiplied with each other such as C8H × A5H in the above mentioned example, the N flag may at times not be set to '0'.

Division

Division is done using the DIV instruction.
When executing the DIV instruction, an HL register ÷ A register operation is executed, the quotient being stored in the L register and the remainder in the H register.
When the quotient exceeds 8 bits, the V flag (overflow) is set and the content of the HL register is held by the preceding dividend.
When a DIV instruction is executed by setting the A register to '0', a zero division exception processing is generated.
The N/V/C/Z flags of the SC are set as follows, according to the result of this operation.

N:	When the MSB of the L register (quotient) is '1'	Set (1)
	When it is '0'	Reset (0)
V:	When the quotient is not restricted to 8 bits or less	Set (1)
	When it is restricted	Reset (0)
C:	Always	Reset (0)
Z:	When the L register (quotient) is 00H	Set (1)
	When it is other than 00H	Reset (0)

Example: SC operating examples

Setting value		SC				Comment
HL reg.	A reg.	N	V	C	Z	
nz	nz	↕	↕	0	↕	
0000H	nz	0	0	0	1	
nz	00H	1	1	0	0	Zero division exception processing has occured
0000H	00H	1	1	0	0	

nz indicates other than '0' of 8-bit or 16-bit data.

• Division and multiplication execution examples

Below are indicated execution examples of DIV instructions.

Setting value		Result			SC			
HL reg.	A reg.	L reg.	H reg.	A reg.	N	V	C	Z
1A16H	64H	42H	4EH	64H	0	0	0	0
332CH	64H	83H	00H	64H	1	0	0	0
0000H	58H	00H	00H	58H	0	0	0	1
0301H	02H	01H	03H	02H	1	1	0	0

(Result: L reg. = quotient, H reg. = remainder)

Since the quotient exceeds 8 bits in the 0301H ÷ 02H in the above example, the value of the HL register is held and the result is not output. In cases such as this, it performs the division by separating the dividend into the upper 8 bits and the lower 8 bits as shown below.

<An execution example of 0301H ÷ 02H>

```
LD    HL,#0003H ; Dividend = upper 8 bits
LD    A,#02H    ; Divisor
DIV             ; L = quotient, H = remainder
LD    [hhll],L  ; Stores the quotient
                ;   (upper 8 bits) into memory
LD    L,#01H    ; Dividend
                ;    = H register + upper 8 bits
DIV             ;
```

Setting value		Result			SC			
HL reg.	A reg.	L reg.	H reg.	A reg.	N	V	C	Z
0003H	02H	01H	01H	02H	0	0	0	0
0101H	02H	80H	01H	02H	1	0	0	0

Remainder: 01H

Quotient: 0180H

2.3 Program Memory

2.3.1 Configuration of the program memory

The first 8M bytes (address 000000H–7FFFFFH) within the 16M byte of address space of the E0C88 are designed to be used as a programming field. However, since the address space is limited to a maximum 64K bytes for MODEL0 and MODEL1, the programming field is also limited to that or less.

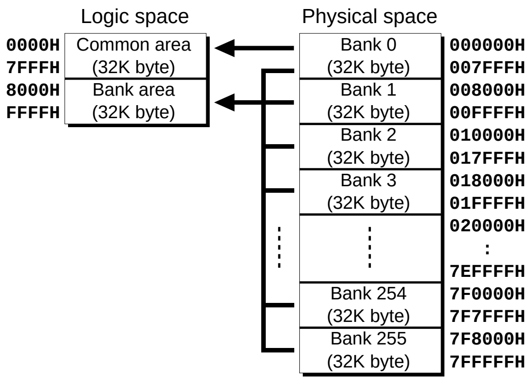


Fig. 2.3.1.1 Configuration of program memory

The E0C88 has adopted a bank mapping system to manage memory that exceeds the 64K bytes logic space of the 8-bit CPU. The maximum 8M bytes program memory is respectively divided into 32K bytes banks from bank 0 up to bank 255.

They are laid out on the 64K bytes logic space, such that two banks logically continue as the address 0000H–FFFFH. It executes the program within that address space. The addressing within the logic space is done by the PC (program counter).

The bank 0 (address 000000H–007FFFH) as a common bank in the logic space address 000000H–007FFFH. It normally becomes fixed for this physical address. The address 000000H–0000FFH is allocated by the exception processing (such as interrupt) vector.

See "3.5.2 Exception processing factor and vectors".

Since the common area is fixed, there is no need to allocate an exception processing vector for each bank. General purpose subroutines can also be described into the common area.

The selected bank is laid out by the CB (code bank) register in the latter half address 8000H–FFFFH bank area.
The banks that it lays out in this section can be optionally selected by the program. However, for MODEL0 and MODEL1, they are fixed in bank 1 and for the minimum modes of MODEL2 and MODEL3, they are fixed in one optionally selected bank.

2.3.2 PC (Program counter) and CB (Code bank register)

The PC (program counter) holds the program address to be executed. The PC content is the address within the 64K bytes logic space and it addresses as program memory logically continuing each 32K common area and bank area each that is not continued by a physical address.
The common area is fixed to bank 0 of the physical address, but one optional bank from among 256 banks can be selected for the bank area (MODEL2 and MODEL3).
CB (code bank) is the register that indicates the bank address (0–255) allocated to this bank area.
The physical address that is output to the address bus to actually access the memory is created within the CPU as shown in Figure 2.3.2.1.

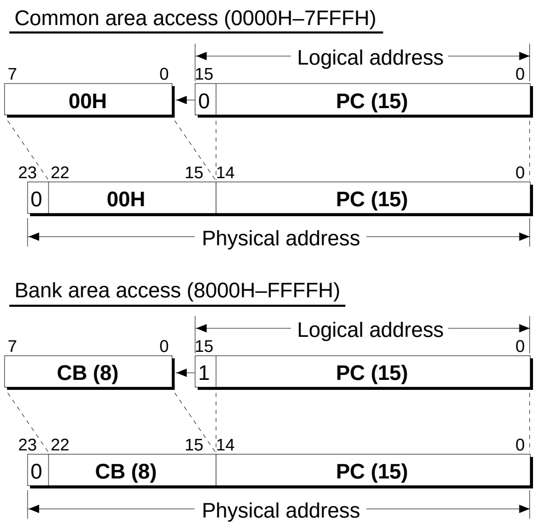


Fig. 2.3.2.1 Logic address and physical address (MODEL2/3)

As shown in the figure, 15 bits excluding the most significant bit are output to the address bus within the 16-bit PC. Its content is output to the address bus A00–A14. The most significant bit of the PC indicates the common area at '0' and the bank area at '1', and this content determines whether or not it will output CB to the address bus. In the case of the common area, 00H is output to A15–A22 of the address bus and in the case of the bank area, the content 8 bits of the CB are output. A23 of the address bus is for the exclusive use of data memory area and it always outputs '0' at the time of maximum 8M byte program memory access. As indicated above, since the most significant bit of the PC is not output to the address bus, you should be aware of this at the time of system development.

The PC content is output as is for MODEL0 and MODEL1, because the address bus is 16 bits.

Value of program counter when "LD BA, PC" or "LD HL, PC" instruction is executed
The instruction "LD BA, PC" and "LD HL, PC" load the current value of the program counter into the BA and HL registers, respectively. Remember that when the processor fetches one of these load instruction, it increments the program counter by two to point to next instruction. So when "LD BA, PC" or "LD HL, PC" is executed, the value of the program counter that is loaded is not the address of the load instruction, but the address of the instruction following it. In other words, PC = <Address of load instruction> + 2. For example, if the instruction "LD BA, PC" is at address 100H, 102H is loaded into the BA register.

2.3.3 Bank management

The execution of the program is basically limited to within the bank allocated to the logic space. The bank is only modified at the time of a branch instruction is executed when another bank is specified by a program.

Note: The CB will not be updated even if the PC count has been overflow by the program execution. It will be reexecuted from the beginning of the common area.

Here following we will explain the bank specification method and the operation during branch instruction execution.
In addition, the items indicated related to bank modification are summarized for only MODEL2 and MODEL3.

Bank setting at the time of resetting

At the time of the initial resetting, the CB is initialized to '1' and bank 1 is allocated to the bank area.
Since the common area is fixed to bank 0, the logic address becomes the same as the physical address.
This setting is specified by another bank in the program and is not modified until the branching is actually executed by the branch instruction.

Bank specification

The CB that indicates the bank that has been selected cannot be directly modified by the program.
The NB (new code bank) register has been prepared for bank specification and it writes the bank address (0–255) of the branch address before executing the branch instruction.

```
LD    NB,A       (specified by the A register)
LD    NB,#bb     (specified by the 8-bit immediate data)
```

The content of the NB is loaded into the CB at the point where the branching is actually done by execution of the branch instruction there following and a new bank is selected for the bank area. When the conditions to not fit for a condition jump or the like, branching is not done and the content of the CB is conversely loaded into the NB. Consequently, it is set up, such that when it executes a branch instruction instead of setting the value for the NB, at that point it will branch into the logic space.

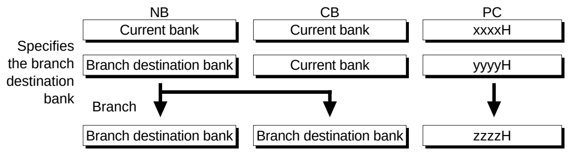


Fig. 2.3.3.1 Bank modification

2.3.4 Branch instruction

Branch instruction modifies the PC and CB to branch the program to an optional address.
The types of branch instructions are classified as follows, according to their operation differences.

Table 2.3.4.1 Types of branch instructions

Type	Condition	Instruction
PC relative jump	Conditional Unconditional	JRS, JRL, DJR
Indirect jump	Unconditional	JP
PC relative call	Conditional Unconditional	CARS, CARL
Indirect call	Unconditional	CALL
Return	Unconditional	RET, RETS, RETE
Software interrupt	Unconditional	INT

There are unconditional branch instructions that also unconditionally branch into the respective above mentioned instructions and several types of conditional branch instructions that branch according to the flag status.
When the condition for a conditional branch instruction has not been met, it does not branch and instead executes the instruction following that branch instruction.

See "4.4 Detailed Explanation of Instructions".

PC relative jump instruction (JRS, JRL, DJR)

The PC relative jump is an instruction that adds the relative address that is specified by the operand for the PC and is branched to that address. It permits relocatable programming.
The relative address is a displacement from the address at branching to the branch destination address, and is specified by one or two bytes.
The relative address that can be specified is the range of -128–127 where the "JRS" instruction is an 8-bit complementary and -32768–32767 where the "JRL" instruction is a 16-bit complementary.
In addition, the branch destination address that is added to the PC becomes the logic address for this relative address.

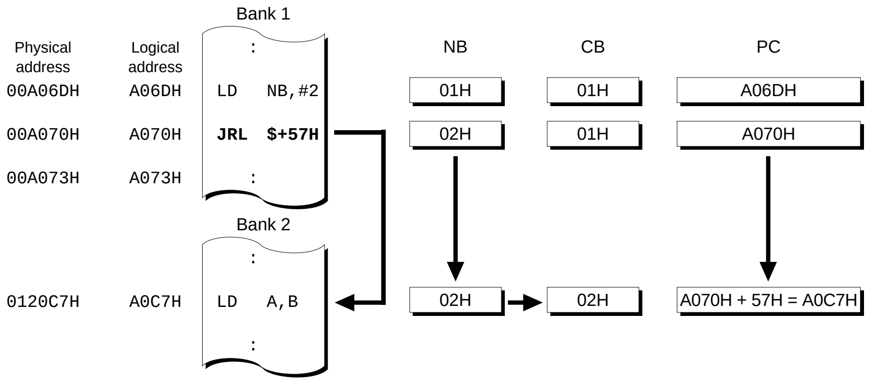


Fig. 2.3.4.1 PC relative jump operation

It can be branched to another bank by prior setting of the NB, but the branch destination strictly cannot specify a physical address within the logic space.
Figure 2.3.4.1 shows the operation of the PC relative jump.

The "JRS" instruction is set by an unconditional jump and 20 types of conditional jump instructions.
The "JRL" instruction is set by an unconditional jump and 4 types of conditional jump instructions.

The "DJR NZ,rr" instruction does '1' subtraction of B register and when the corresponding result is other than '0', it executes the "JRS" unconditional jump instruction.
This instruction permits the simple entry of the repeat routine for that initial value portion making B register the counter.

Example: Wait routine for a 50 cycle time

```
 LD   B,#12   ;Sets the initial value for the B register (2 cycle)
 DJR  NZ,$    ;Repeats until the B register becomes '0' (48 cycle)
```

Indirect jump instruction (JP)

The indirect jump is the instruction that indirectly specifies branch destination address.
The "JP [kk]" instruction loads the content of the address 00kk (kk = 00H–FFH, page is fixed at 0) of the memory into the lower 8 bits of the PC and loads the content of the address 00kk+1 of the memory into the upper 8 bits of the PC, then unconditionally branches into those addresses. The address 00kk it specifies here is set up as the vector field for exception processing and software interrupts.

The "JP HL" instruction unconditionally branches the content of the HL register as an address. Since this instruction can convert operation results as they are into branch destination addresses, it is effective for such things as the creation of jump tables.

PC relative call instruction (CARS, CARL)

The PC relative call is the instruction that adds the relative address specified by the operand to the PC and calls subroutines from that address.
The relative address is a displacement from the address at branching to the branch destination address, and is specified by one or two bytes.
The relative address that can be indicated are the ranges -128–127 where the "CARS" instruction is an 8-bit complimentary indication and -32768–32767 where the "CARS" instruction is an 16-bit complimentary indication.
In addition, since this relative address is added to the PC, the branch destination banks becomes the logic address.
Branching to other addresses as well can be done by prior setting of the NB, but the branch destination strictly cannot specify a physical relative address within a logic space.
At the time of execution of a subroutine call, the PC value (top address of the instruction following the call instruction) is pushed into the stack as return information.
In the maximum mode of MODEL2/3, in addition to the PC value, the CB value is also pushed onto the stack. When returning from a subroutine, the program sequence returns to the bank where the subroutine was called.

In the minimum mode of MODEL2/3, only the PC value is pushed onto the stack, as with MODEL0/1. Consequently, program memory of 64K bytes or more cannot be used. Figure 2.3.4.2 shows the PC relative call operation.

The "CARS" instruction is set by an unconditional call and 20 types of conditional call instructions. The "CARL" instruction is set by an unconditional call and 4 types of conditional call instructions.

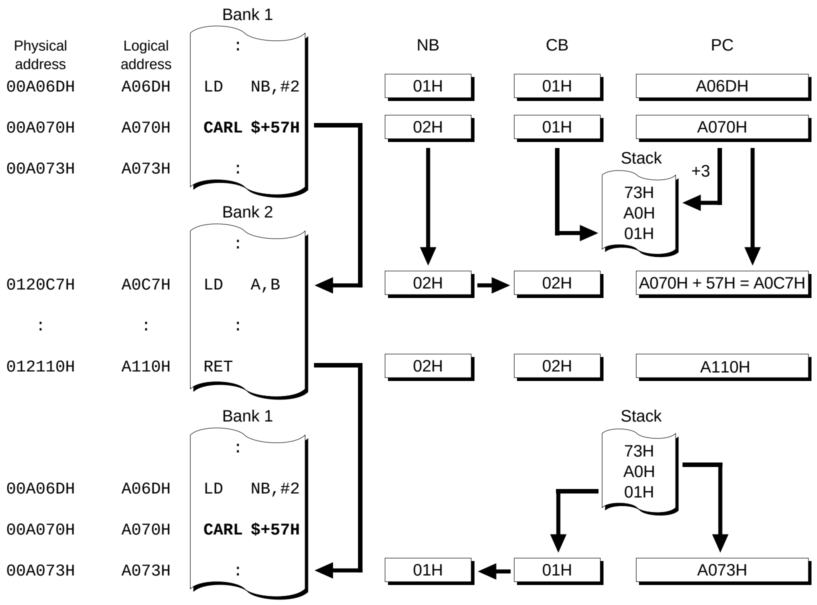


Fig. 2.3.4.2 PC relative call operation

Indirect call instruction (CALL)

The indirect call is a call instruction that indirectly specifies the subroutine address.
The "CALL [hhll]" instruction loads the content of the memory address hhll (hhll = 0000H–FFFFH, page is specified by EP register) into the lower 8 bits of the PC and loads the content of the memory address hhll+1 into the upper 8 bits of the PC to unconditionally call the subroutines for those addresses. At the time of execution of a subroutine call, the PC value (top address of the instruction following the call instruction) and the CB value (in case of the MODEL2/3 maximum mode) are pushed into the stack as return information.

Return instructions (RET, RETS and RETE)

A return instruction is an instruction for returning to the routine called from the subroutine accessed by the call instruction. The return instruction pops the PC value (top address of the instruction following the call instruction) that was pushed onto the stack on executing the subroutine call to the program counter PC.
In the maximum mode of MODEL2/3, the CB value is also popped from the stack and the program returns to the bank where the subroutine was called.

In the minimum mode of MODEL2/3, only the PC value is popped, as with MODEL0/1. When the bank is changed at the time of the execution or after execution of the call instruction, return to the correct address is impossible even if the return instruction is executed.
The "RET" instruction returns the processing to the top address of the instruction following the call instruction with the return information as is.
Since the "RETS" instruction returns by adding a '2' to the PC value of the return information, it can skip the 1 byte instruction following the call instruction.

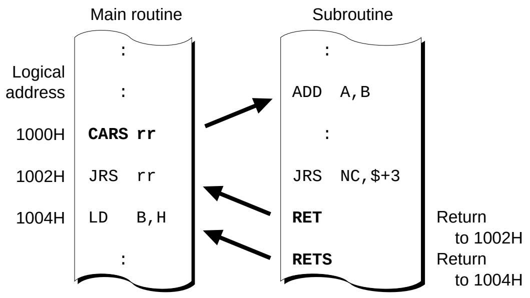


Fig. 2.3.4.3 Return from subroutine

The "RETE" instruction is the return instruction exclusively for the software interrupt routine and exception processing routine and differs from the "RET" instruction in that the content of the SC (system condition flag) is contained in the return information.

See "3.5 Exceptional Processing Status".

Software interrupt instruction (INT)

The software interrupt instruction "INT [kk]" is an instruction that specifies the vector address of the address 00kk (kk = 00H–FFH, page is fixed at 0) to execute its interrupt routine. It is a type of indirect call instruction, but the SC (system condition flag) is also pushed into the stack before branching. Consequently, the interrupt routines executed by this instruction must invariably return by the "RETE" instruction.

See "3.5 Exceptional Processing Status".

Value of program counter when relative branch instruction is executed

JRS, CARS and DJR instructions

The JRS, CARS and DJR instructions are signed 8-bit relative branch instructions in which relative address rr (-128 to 127) added to the current value of the program counter with sign to determine which address control is branched to. This branch address is given by following equation:

<Branch address> = <Address of branch instruction> + rr + (n - 1)

n ... length of the branch instruction

For example, if the instruction "JRS LE,rr" is at address 100H, branch address is set to 102H + rr.

JRL and CARL instructions

The JRL and CARL instructions are signed 16-bit relative branch instructions in which relative address qqrr (-32768 to 32767) added to the current value of the program counter with sign to determine which address control is branched to. This branch address is given by following equation:

<Branch address> = <Address of branch instruction> + qqrr + 2

For example, if the instruction "JRL C,qqrr" is at address 100H, branch address is set to 102H + qqrr.

2.4 Data Memory

2.4.1 Data memory configuration

Everything within the address space (maximum 16M bytes) of the E0C88, with the exception of the field it uses as program memory can be used as data memory.

RAM, display memory, I/O memory controlling the peripheral circuits and like memory is laid out in the data memory field.

The data memory is managed by making 64K bytes one page. Figure 2.4.1.1 shows the data memory configuration.

Since the address space is 64K bytes, it is not necessary to consider management by page for MODEL0/1. MODEL2/3 is configured with 255 pages (maximum).

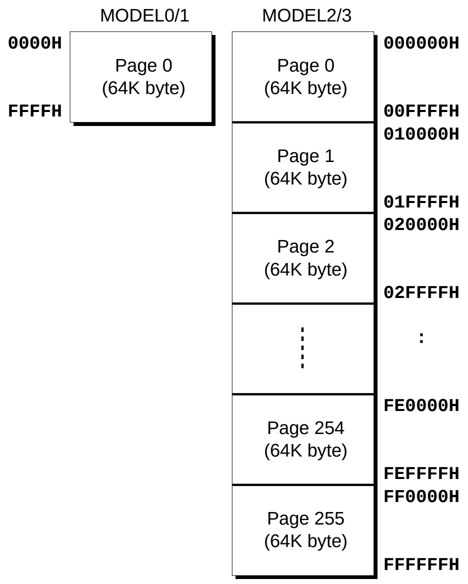


Fig. 2.4.1.1 Data memory configuration

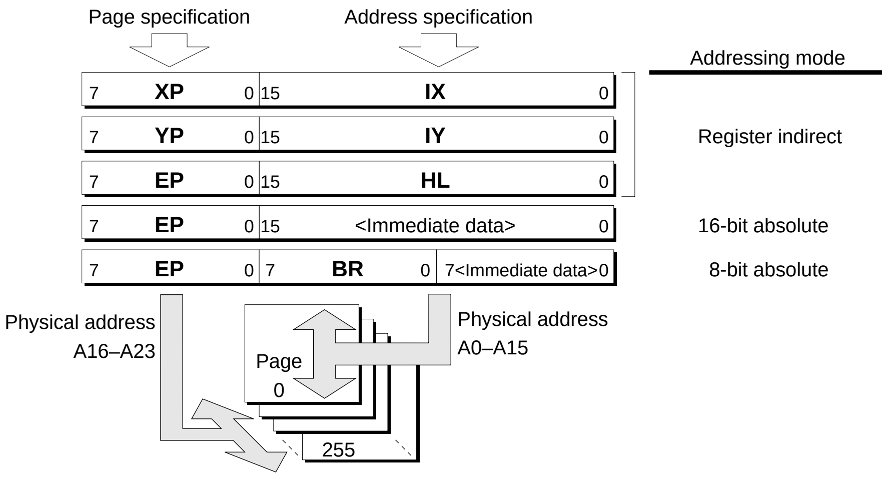


Fig. 2.4.2.1 Data memory addressing

2.4.2 Page registers EP, XP, YP

The physical space of the data memory is logically delimited into 64K bytes of page. Consequently, the upper 8 bits of the physical address are managed as page sections and the lower 16 bits as logical addresses. The address specification within a page is done primarily by index register and immediate data according to the addressing mode. The 3 page registers EP, XP and YP are set for specification of the page sections in MODEL2 and MODEL3. They are appropriately used according to the addressing mode specification. Figure 2.4.2.1 shows the correspondence of the page registers with the addressing modes.

See "4.1 Addressing Mode".

2.4.3 Stack

The stack is memory that is accessed in the LIFO (Last In, First Out) format and in the E0C88 it is allocated to the RAM field of the data memory. When a subroutine call, exception processing (interrupt), or the like has been generated, the stack is used for register information evacuation by the CPU. In addition, it can effect such operations as register evacuation at an optional program location.

Here following we will describe the storing of data in a stack as "push" and the removal of stored data as "pop".

Stack pointer SP

Data is sequentially pushed from the uppermost address of the stack and, conversely, when data will be removed it is popped in order, from the last pushed data. The register that indicates the stack address that does this push and pop is the SP (stack pointer).

The SP is subtracts '1' (pre-decrement) by one byte data push and adds '1' (post-increment) by one byte data pop.

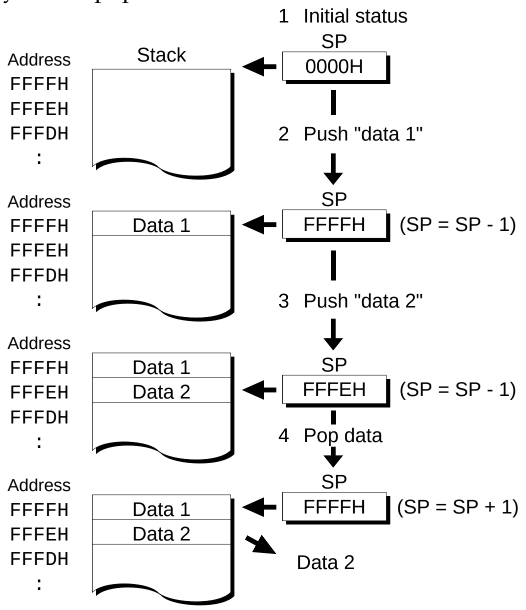


Fig. 2.4.3.1 Operation of stack

The stack position within the physical memory decided by the SPP0–SPP7 (stack pointer page) signal that is input to the core CPU from the peripheral circuit as a page address and when the stack is accessed, the content of the SPP0–SPP7 is output as is to the page section (A16–A23) of the address.
The address within that page is specified by SP. Generally, setting the address 0000H as the initial value of the SP causes the data to be sequentially pushed toward the lower address from the final address FFFFH of that page.

*Note: Since the SP (stack pointer) is undefined at the time of the initial resetting, you should be sure to initialize using the program ("LD SP,**" instruction) before the stack is used.*

Subroutine call and stack

When executing a call instruction, the top address of the instruction following the call instruction and the CB (in case of MODEL 2/3 maximum mode) are pushed into the stack as return addresses prior to the branching to the subroutine.
Return information that has been pushed into a stack is popped by execution of a return instruction and reset to PC and CB.
The type of nesting that calls another separate subroutine from within a subroutine is possible up to any level within the usable page memory allocated by the stack.

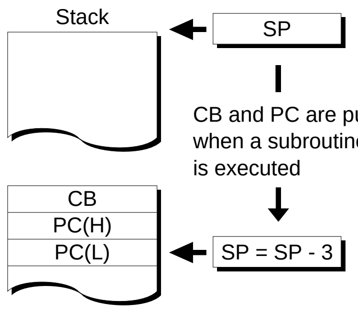


Fig. 2.4.3.2 Stack consumption at the time of a subroutine call execution

In the maximum mode of MODEL2/3, a subroutine call causes a 3 bytes (CB and PC) consumption of the stack. In the minimum mode of MODEL2/3 and MODEL0/1, it consumes a 2 bytes portion of PC, except for CB.

Exception processing and stack

The return information is pushed to the stack the same as for a subroutine call, at the time of an exception processing (such as interrupt) generation as well. An SC is included in the return information at this time, in addition to the return addresses PC and CB (in case of the MODEL2/3 maximum mode).

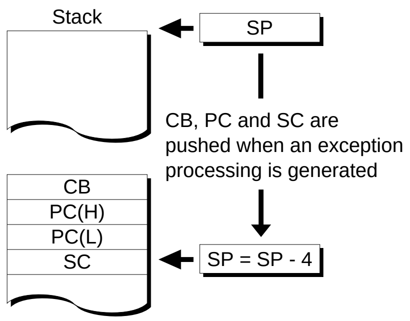

Stack
SP
CB, PC and SC are pushed when an exception processing is generated
CB
PC(H)
PC(L)
SC
SP = SP - 4

Fig. 2.4.3.3 Stack consumption when an exception processing is generated

In the maximum mode of MODEL2/3, the generation of an exception processing (such as interrupt) causes a 4 bytes (PC, CB and SC) consumption of the stack. In the minimum mode of MODEL2/3 and MODEL0/1, it consumes a 3 bytes portion of PC and SC, except for CB.

Other stack operations

The return information by the subroutine call or exception processing (such as interrupt) is automatically pushed, but the general purpose register is not pushed. When you want to return from the subroutine or exception processing routine, maintaining the contents of general purpose register as it was prior to branching, instructions to push and pop the contents of the register must be arranged at the beginning and end of the routine, respectively.
The push/pop of the register is done by the "PUSH" instruction and the "POP" instruction. The registers that can push/pop according to this instruction are as follows.
A, B, L, H, BR, SC, EP*, IP (XP and YP)*, BA, HL, IX, IY
Those with an asterisk "*" do not exist in MODEL0/1.

The "PUSH ALL"/"PUSH ALE" (for MODEL2/3) instruction that pushes all the above mentioned registers except for SC with 1 instruction and the "POP ALL"/"POP ALE" (for MODEL2/3) instruction that pops with 1 instruction have been prepared.

```
PUSH  ALL = PUSH  BA        POP  ALL = POP  BR
            PUSH  HL                   POP  IY
            PUSH  IX                   POP  IX
            PUSH  IY                   POP  IY
            PUSH  BR                   POP  BA

PUSH  ALE = PUSH  BA        POP  ALE = POP IP
            PUSH  HL                   POP  EP
            PUSH  IX                   POP  BR
            PUSH  IY                   POP  IY
            PUSH  BR                   POP  IX
            PUSH  EP                   POP  HL
            PUSH  IP                   POP  BA
```

"ALL" in the operand is for MODEL0/1. "ALE" is for MODEL2/3, and expanded registers EP and IP (XP and YP) are also pushed/popped.
The storing of arguments transferred to subroutines and the like in stack field is often done for structured programming, however, instructions that control the SP without the use of the above mentioned "PUSH" and "POP" instructions and that permit easy direct access to stack field have also been prepared.

```
ADD, SUB, CP, INC, DEC, LD
```

Note: Since the stack is allocated to general purpose RAM, be careful not to overlap the data field and stack field.

2.4.4 Memory mapped I/O

The E0C88 Family makes the E0C88 the core CPU and builds in various types of circuits, such as input/output ports into its periphery. The E0C88 has adopted a memory mapped I/O system for controlling those various peripheral circuits and registers for handling the interchanges of control bits and data of the peripheral circuits has been laid out in the data memory field.
The term I/O memory is used to differentiate this memory field from general purpose RAM, but since they have the page control and access methods in common as data memories, it can control peripheral circuit using normal memory access instructions.

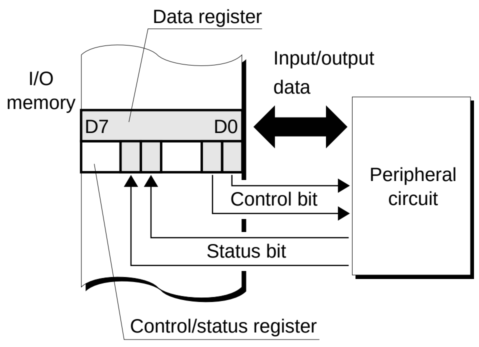


Fig. 2.4.4.1 Peripheral circuit and memory mapped I/O

Models with built-in LCD driver use part of the data memory as the display memory for segment data. Each bit of the display memory field corresponds 1 to 1 with the segment and the turning on/off of the bit causes the corresponding segment to light/go out.
The control of this segment can also be done by the normal memory access instruction.

Note: Depending on the model, there may be instances where part of the I/O memory and/or the display memory may be set up for writing only. In such cases, it is not possible effect direct bit control (read/modify/write) of those sections by such means as arithmetic and logic operation instructions. When bit control is performed, a buffer storing the same contents in the R/W memory is secured, and the data must be modified in the buffer, then written it into primary memory.

Please refer to the various manuals of the E0C88 Family for details on the peripheral circuits, I/O memory and display memory.

3 CPU OPERATION AND PROCESSING STATUS

CPU operates in synchronising with the system clock. The CPU process also includes the various types of statuses such as the status that sequentially executes programs and the standby status. Here we will explain the various types of processing statuses including interrupts and the timing of the operations.

3.1 Timing Generator and Bus Control

First we will explain the clock and bus control on which the CPU operation is based.

3.1.1 Bus cycle

The timing generator of the E0C88 generates a two phase divided signal from the clock CLK that has been input and factors the CLK into states. One state becomes 1/2 cycle of the CLK. The one bus cycle that becomes the instruction execution unit is composed of four states.

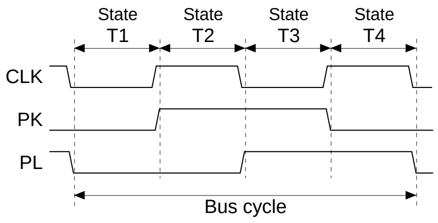


Fig. 3.1.1.1 State and bus cycle

The numeric values indicated as cycles in the instruction set list indicate the number of bus cycles. In the E0C88, the data bus status in each bus cycle is output externally on the DBS0 and DBS1 signals as a 2 bit status. The peripheral circuit can easily effect such things as the directional control of the bus driver by means of this signal. The state of data bus indicated by DBS0 and DBS1 are as shown in Table 3.1.1.1.

Table 3.1.1.1 Sate of data bus

DBS1	DBS0	State
0	0	High impedance
0	1	Interrupt vector address read
1	0	Memory write
1	1	Memory read

Here following is indicated the timing chart for each bus status.

High impedance

During an internal register access, the data bus goes into high-impedance state. Both the read signal $\overline{RD}$ and the write signal $\overline{WR}$ are fixed to a high level, and the address bus outputs a dummy address during the bus cycle period.

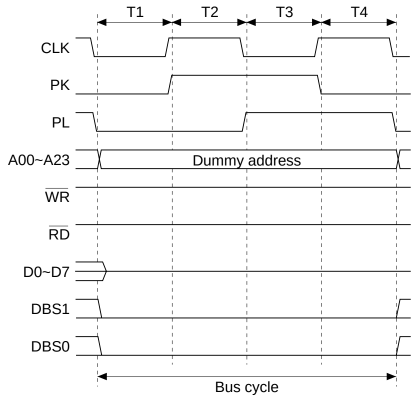


Fig. 3.1.1.2 Bus cycle at the time of internal register access

Interrupt vector address read

The interrupt vector address is read from the data bus between the T2–T3 states.
At the time of this read, an interrupt vector address read dedicated signal $\overline{RDIV}$ is output, instead of a read signal $\overline{RD}$ not being output. The address bus outputs a dummy address during the bus cycle period.

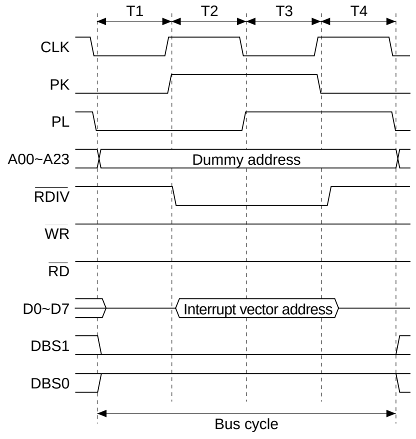


Fig. 3.1.1.3 Bus cycle at time of the interrupt vector address read

Memory write

At the time of a memory write, written data is output to the data bus between T2–T4 states and the write signal $\overline{WR}$ is output to the T3 state. The address bus outputs the target address during the bus cycle period.

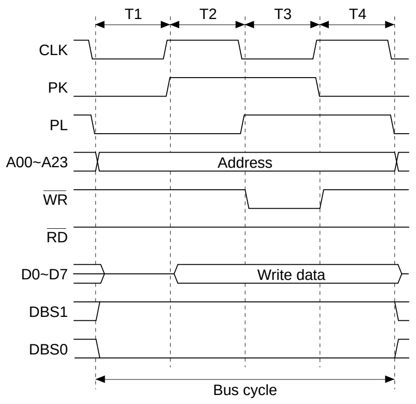


Fig. 3.1.1.4 Bus cycle at the time of memory write

Memory read

At the time of memory reading, the read signal $\overline{RD}$ between the T2–T3 states is output it reads the data on the data bus. The address bus outputs the target address during the bus cycle period.

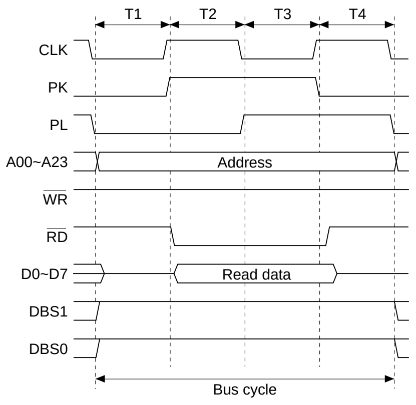


Fig. 3.1.1.5 Bus cycle at the time of memory read

3.1.2 Wait state

The E0C88 can extend the bus cycle by inserting a wait state in order to precisely access the low speed device connected to the bus line.
The E0C88 has a function for inserting a WAIT for access time extensions as wait states and controls it by the input signal of the $\overline{WAIT}$ terminal.

The $\overline{WAIT}$ signal is sampled at the CLK rising edge of the T3 state. When the $\overline{WAIT}$ signal at this time is low level, it inserts wait states Tw1 and Tw2 between T3 state and T4 state and extends the access time.
When the $\overline{WAIT}$ signal is high level, the wait state is not inserted.
The wait states Tw1 and Tw2 are continuously inserted while the $\overline{WAIT}$ signal is low level. The sampling for releasing the insertion of the wait state is done at the CLK rising edge of the Tw2 state and when the $\overline{WAIT}$ signal returns to high level, the following wait states are not inserted, but rather it begins the T4 state.

The wait state is inserted only when it accesses the devices connected on the memory space and is not inserted when it accesses the internal register.
Below is shown the timing chart for wait insertion for each cycle of the interrupt vector address read, the memory write and the memory read.

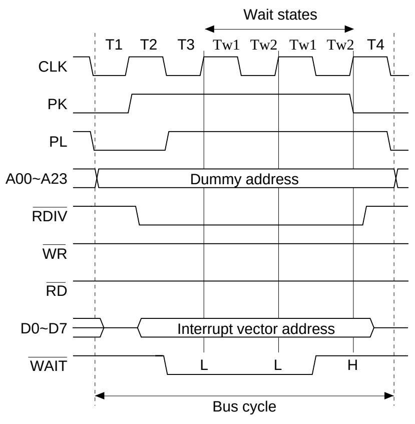


Fig. 3.1.2.1 Wait insert of the interrupt vector address read cycle

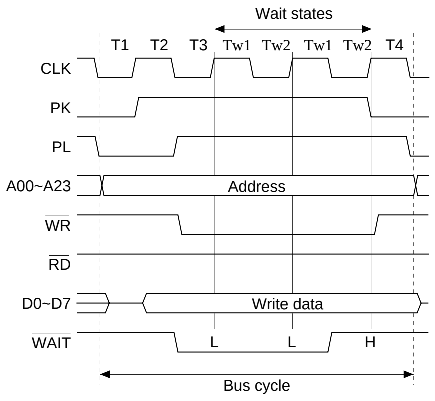


Fig. 3.1.2.2 Wait insert of the memory write cycle

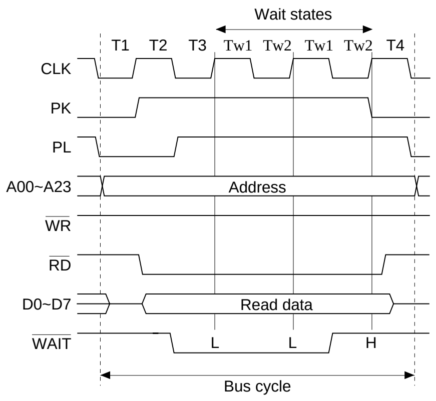


Fig. 3.1.2.3 Wait insert of the memory read cycle

3.2 Outline of Processing Statuses

The operations of the E0C88 can be classified by the content of their processing into five types, reset status, program execution status, exception processing status, bus authority release status and standby status. Table 3.2.1 shows the classification of the processing statuses and Figure 3.2.1 the status transition diagram.

Table 3.2.1 Classification of the processing statuses

Processing status		Outline
Reset status		Status where the CPU is reset and stopped.
Program execution status		Status where the CPU successively executes programs.
Exception processing status		Transitive status where exception processing (fetching of a vector address, PC and SC evacuation, setting of a branch address for the PC) is activated by an exception processing factor such as a reset or interrupt.
Bus authority release status		Status where an external bus is released by a bus authority request signal from outside.
Standby status	HALT	Status where it stops the CPU and reduces power consumption.
	SLEEP	Status where it stops the CPU and peripheral circuit and reduces power consumption.

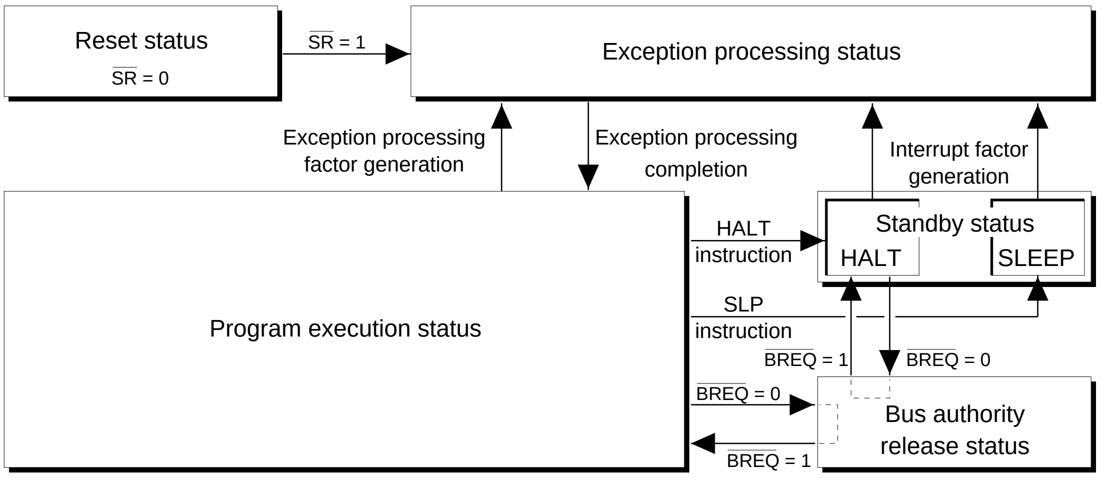


Fig. 3.2.1 Status transition diagram

3.3 Reset Status

The reset status indicates the status where the E0C88 is reset and stops. The E0C88 is reset by inputting a low level into the $\overline{SR}$ terminal. Since the resetting is done out of synchronization with the CLK, it shifts from all the processing status to immediate reset status. Part of the internal registers are initialized by the reset. Table 3.3.1 indicates the initial set value of the register.

Figure 3.3.1 shows the reset status and the sequence following reset release.
The address bus, data bus and read/write signals become high impedance during the reset period when the $\overline{SR}$ terminal is low level. However, since the address bus and read/write signals are pulled up within the CPU, a high level is output.
Reset is released when the $\overline{SR}$ terminal becomes high level and it starts the first bus cycle at the point where the falling edge of the CLK has been input twice. In this bus cycle, a dummy address is output to the address bus and the interrupt acknowledge $\overline{IACK}$ becomes enabled by the following bus cycle. As a result, this starts the exception processing for reset that loads the start address stored in the vector table into the PC (program counter) which is in undefined status. At this time it simultaneously also does the processing for loading the initial value 01H of the NB (new code bank register) into the CB (code bank register). As a result bank1 (008000H–00FFFFH) is selected for the bank area after resetting.

<u>After an initial reset, the program is executed from start address stored in 000000H–000001H of the memory.</u>

Table 3.3.1 Initial set value of the internal registers

Register name	Symbol	Bit length	Initial value
Data register A	A	8	Undefined
Data register B	B	8	Undefined
Index (data) register L	L	8	Undefined
Index (data) register H	H	8	Undefined
Index register IX	IX	16	Undefined
Index register IY	IY	16	Undefined
Program counter	PC	16	Undefined*
Stack pointer	SP	16	Undefined
Base register	BR	8	Undefined
Zero flag	Z	1	0
Carry flag	C	1	0
Overflow flag	V	1	0
Negative flag	N	1	0
Decimal flag	D	1	0
Unpack flag	U	1	0
Interrupt flag 0	I0	1	1
Interrupt flag 1	I1	1	1
New code bank register	NB	8	01H
Code bank register	CB	8	Undefined*
Expand page register	EP	8	00H
Expand page register for IX	XP	8	00H
Expand page register for IY	YP	8	00H

* The value stored in the top of bank 0 (000000H–000001H) is loaded into the PC by the reset exception processing. At the same time, the initial value 01H of the NB is loaded into the CB.

Registers NB, CB, EP, XP and YP are set for the MODEL2/3 and do not exist in the MODEL0/1.

Note: *Use the program to initialize, if necessary, for registers that have not been initialized by resetting.*

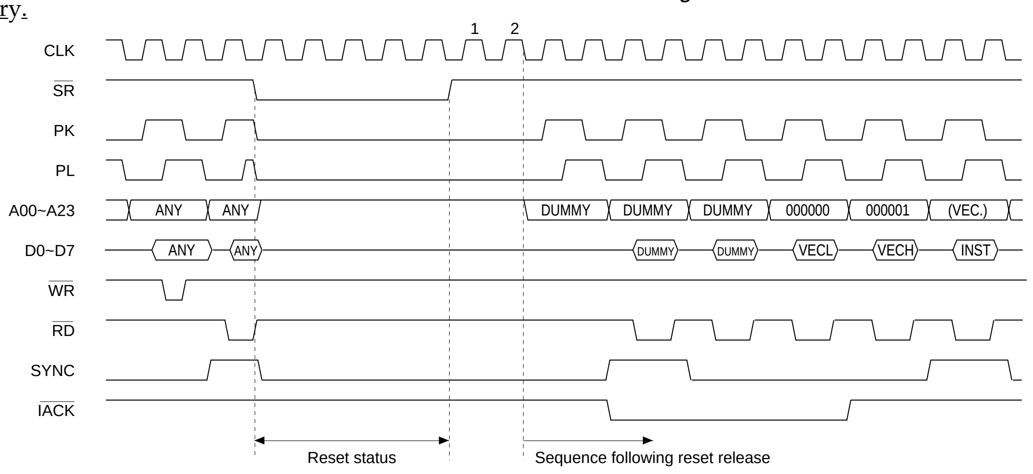


Fig. 3.3.1 Reset status and sequence following reset release

3.4 *Program Execution Status*

The program execution status indicates the status where the E0C88 successively executes programs.
In the E0C88, the fetching of the first operation code of the instruction is done overlapping the last cycle of the immediately prior instruction.
Consequently, the execution cycle for 1 instruction of the E0C88 begins either from the fetch cycle for the second op-code, the read cycle for the first operand or the first execution cycle (varies depending on the instruction) and terminates with the fetch cycle for the first op-code of the following instruction. 1 cycle instruction only becomes the fetch cycle of the first op-code of the following instruction. In addition, there are also instances where it shifts to the fetch cycle of the first op-code rather than interposing an execute cycle after an operand read cycle.
In the fetch cycle of the first op-code, the SYNC signal during that period becomes high level.

Figure 3.4.1 shows an example of the following program and instruction execution cycle in accordance with the conditions.

Program list

```
001000   44 6E      LD   A,[BR:6EH]
001002   CE 10 34   SUB  A,[IX+34H]
001005   50         LD   L,A
001006   69         LD   [HL],B
```

Register and memory conditions

B	=	7FH
H	=	81H
BR	=	83H
IX	=	8000H
EP	=	00H
XP	=	00H
M(008034H)	=	27H
M(00836EH)	=	9BH

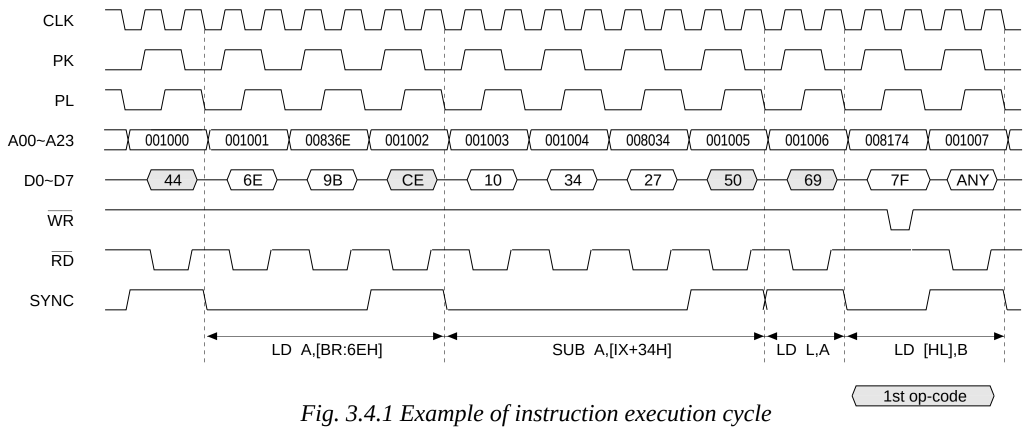


Fig. 3.4.1 Example of instruction execution cycle

3.5 *Exception Processing Status*

Exception processing status indicates a transition status where the E0C88 suspends normal program execution and changes the processing flow due to an exception processing factor such as an interrupt.
Figure 3.5.1 shows the exception processing sequence.

Exception processing begins with the termination of the instruction cycle being executed at the time when an exception processing factor has occurred. As indicated in the exception processing flow, after evacuation of the return information for reopening the suspended routine into the stack, it loads the start address of the exception processing routine (processing routine set by the user) from the vector address corresponding to the exception processing factor into the PC, then branches to that processing routine. However, for reset exception processing, the return information is not evacuated.
The transitive status up to the branching to the exception processing routine is the exception processing status and it returns to the normal program execution status after branching.

Exception processing routines created by the user take the subroutine format, however, since the SC is pushed into the stack, the return instruction invariably uses a "RETE" instruction. The "RETE" instruction causes the resumption of the execution of the routine suspended by the exception processing.

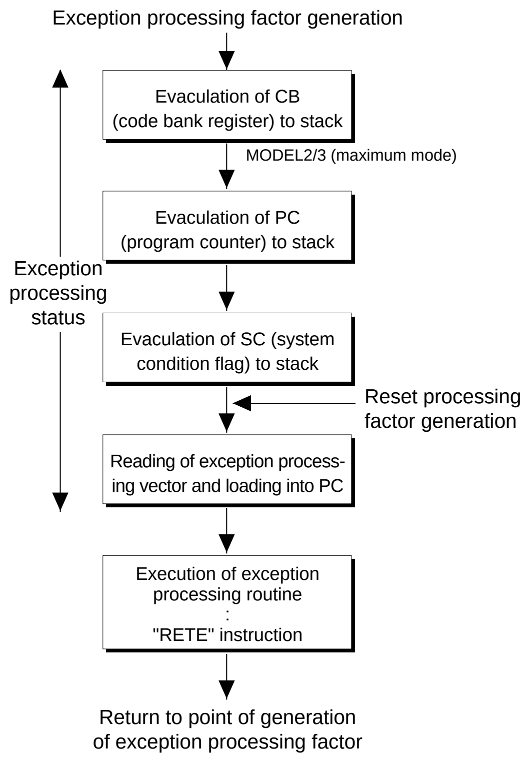

Exception processing factor generation
Evaculation of CB (code bank register) to stack
MODEL2/3 (maximum mode)
Evaculation of PC (program counter) to stack
Exception processing status
Evaculation of SC (system condition flag) to stack
Reset processing factor generation
Reading of exception processing vector and loading into PC
Execution of exception processing routine
:
"RETE" instruction
Return to point of generation of exception processing factor

Fig. 3.5.1 Exception processing flow

3.5.1 Exception processing types and priority

Table 3.5.1.1 indicates the types of exception processing and priorities.
A priority order is set for the exception processing factors and when multiple factors have been generated at the same time, the exception processing having the highest priority is executed first.
When a new exception processing factor has been generated for an exception processing status, a new exception processing is executed following the termination of the exception processing at that time (prior to the execution of an exception processing routine).
For example, when an $\overline{\mathrm{NMI}}$ has been generated during $\overline{\mathrm{IRQ3}}$ exception processing execution, sampling of the $\overline{\mathrm{NMI}}$ is done at the final stage of the $\overline{\mathrm{IRQ3}}$ exception processing, and the $\overline{\mathrm{NMI}}$ processing routine created by the user is executed ahead of the $\overline{\mathrm{IRQ3}}$ processing routine created by the user. The $\overline{\mathrm{IRQ3}}$ processing routine is executed after the $\overline{\mathrm{NMI}}$ processing routine has been terminated.
For this reason, the exception processing due to an interrupt has been set up such that an interrupt having a lower priority than that interrupt will be masked.
Since the exception processing by an INT instruction can be started by the program, a priority is not set.

Table 3.5.1.1 Types of exception processing and priorities

Priority	Type	Exception processing start timing
High ↑	Reset	Initial fetch cycle following change of $\overline{\mathrm{SR}}$ terminal from low level to high level
	Zero division	Immediately following DIV instruction when a DIV instruction (division) has been executed by divisor zero.
	$\overline{\mathrm{NMI}}$	<Non-maskable interrupt> When an instruction or exception processing is terminated during execution at the point where a falling edge has been input into the $\overline{\mathrm{NMI}}$ terminal.
	$\overline{\mathrm{IRQ3}}$	<Interrupt request 3> When an instruction or exception processing is terminated during execution at the point where a low level has been input into the $\overline{\mathrm{IRQ3}}$ terminal.
	$\overline{\mathrm{IRQ2}}$	<Interrupt request 2> When an instruction or exception processing is terminated during execution at the point where a low level has been input into the $\overline{\mathrm{IRQ2}}$ terminal.
↓ Low	$\overline{\mathrm{IRQ1}}$	<Interrupt request 1> When an instruction or exception processing is terminated during execution at the point where a low level has been input into the $\overline{\mathrm{IRQ1}}$ terminal.
None	INT instruction	<Software interrupt> Execution of the INT instruction

3.5.2 *Exception processing factor and vectors*

The start address of an exception processing routine is set as the vector for the vector address corresponding to each exception processing factor. This vector is loaded into the PC following exception processing and branched to the exception processing routine.
Table 3.5.2.1 shows the correspondence of the vector addresses with the exception processing factors.

The vectors are fixed at the 2 bytes address information that indicate the logic address, regardless of the CPU model. The bank for the exception processing routine cannot be specified even in the maximum mode of MODEL2/3. Consequently, it is necessary to set the start address of the exception processing routine to within the common area (000000H–007FFFH) in order to branch from multiple banks to a common exception processing routine.
The $\overline{\text{IRQ1}}$ to $\overline{\text{IRQ3}}$ vector addresses are set by a peripheral circuit. In case of an INT instruction, the instruction operand becomes the vector address as is and when other exception processing factors are also included, it reserves up to a maximum of 128 vectors.

Table 3.5.2.1 Correspondence of vector addresses with exception processing factors

Exception processing factor	Vector address	Vector address generation source
Reset	000000H–000001H	Within CPU
Zero division	000002H–000003H	Within CPU
$\overline{\text{NMI}}$	000004H–000005H	Within CPU
$\overline{\text{IRQ1}}$–$\overline{\text{IRQ3}}$	000006H–0000FFH	Peripheral circuit
INT instruction	000000H–0000FFH	Instruction operand

3.5.3 *Interrupts*

There are four types of interrupts $\overline{\text{NMI}}$, $\overline{\text{IRQ3}}$, $\overline{\text{IRQ2}}$ and $\overline{\text{IRQ1}}$ and they are respectively set by the interrupt priority levels indicated in Table 3.5.3.1.

Table 3.5.3.1 Interrupt levels

Priority	Interrupt priority level	Interrupt factor
High	4	$\overline{\text{NMI}}$
↑	3	$\overline{\text{IRQ3}}$
↓	2	$\overline{\text{IRQ2}}$
Low	1	$\overline{\text{IRQ1}}$

An interrupt can be masked (set such that an interrupt is not accepted) by the interrupt flags I0 and I1. When the interrupt priority level has been set to the 2 bits I0 and I1 by the program, only interrupts above that priority level will be accepted. Among them, the $\overline{\text{NMI}}$ of level 4 is always accepted regardless of the I0 and I1 settings.
In addition, when exception processing is executed by the generation of an interrupt factor, I0 and I1 are set to same level of the accepted interrupt and interrupts of the same level or lower are masked. Since the setting of this mask is done after stack evacuation of the SC (system condition flag), the SC returns to its original status at the point where the interrupt processing routine has been terminated by an "RETE" instruction and the interrupt mask also returns to the original priority level.
When you wish to enable multiple interrupts at the same level or lower by the interrupt processing routine, you should re-set the priority level within that routine.

Table 3.5.3.2 Interrupt mask settings

I1	I0	Acceptable interrupts
1	1	$\overline{\text{NMI}}$
1	0	$\overline{\text{NMI}}$, $\overline{\text{IRQ3}}$
0	1	$\overline{\text{NMI}}$, $\overline{\text{IRQ3}}$, $\overline{\text{IRQ2}}$
0	0	$\overline{\text{NMI}}$, $\overline{\text{IRQ3}}$, $\overline{\text{IRQ2}}$, $\overline{\text{IRQ1}}$

Table 3.5.3.3 I0 and I1 following interrupt acceptance

Accepted einterrupt factor	I1	I0
$\overline{\text{NMI}}$	1	1
$\overline{\text{IRQ3}}$	1	1
$\overline{\text{IRQ2}}$	1	0
$\overline{\text{IRQ1}}$	0	1

Interrupts are disabled while the instruction to modify the contents of NB or SC is being executed. The exception processing of the interrupt generated during that period is started after the following instruction has been executed.

3.5.4 Exception processing sequence

Exception processing sampling is done at the rising edge of the SYNC signal (at the start of the first op-code fetch cycle of the instruction). When an exception processing factor has been generated here, the CPU outputs an interrupt acknowledge signal $\overline{\text{IACK}}$ and begins the exception processing. In case of the $\overline{\text{IRQ1}}$–$\overline{\text{IRQ3}}$ interrupts, the peripheral circuit that generated the interrupt receives the $\overline{\text{IACK}}$ signal then holds the vector address.
Below are indicated the sequences of each exception processing.

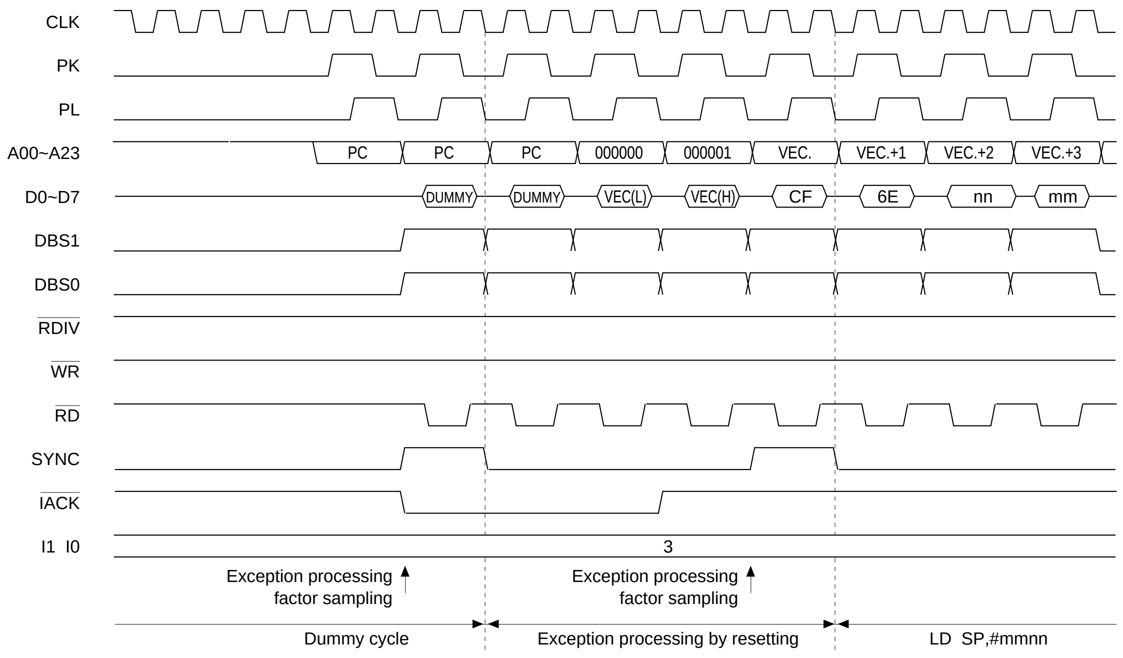
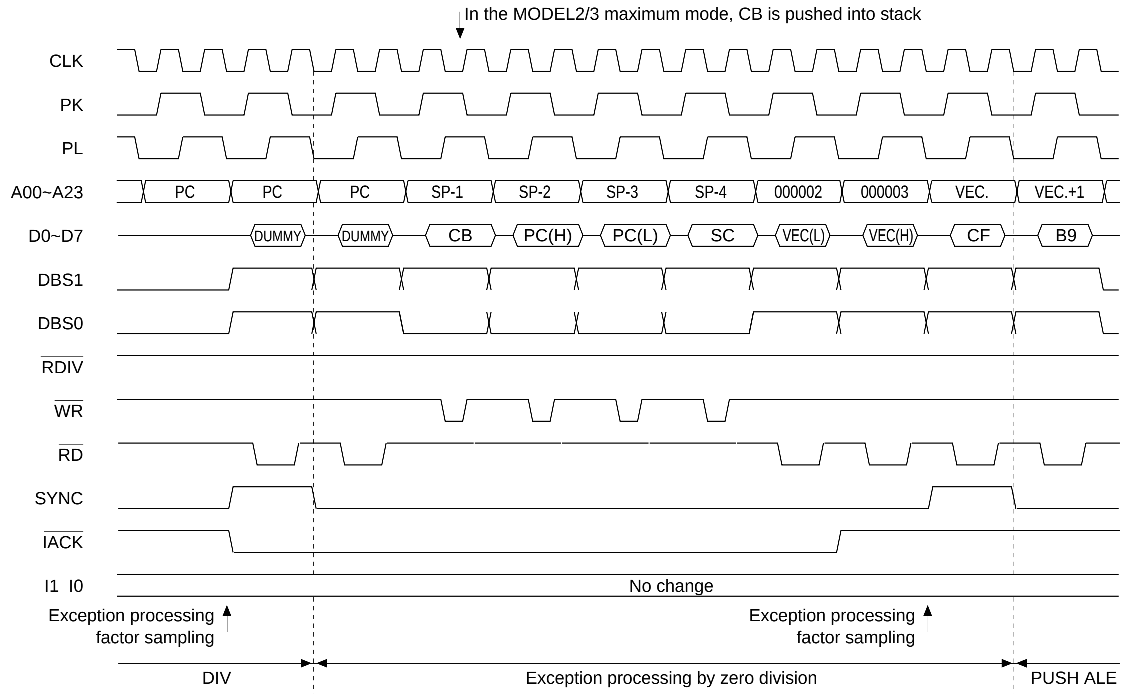


Fig. 3.5.4.1 Exception processing sequence - reset -



In the MODEL2/3 maximum mode, CB is pushed into stack
CLK
PK
PL
A00~A23
PC
PC
PC
SP-1
SP-2
SP-3
SP-4
000002
000003
VEC.
VEC.+1
D0~D7
DUMMY
DUMMY
CB
PC(H)
PC(L)
SC
VEC(L)
VEC(H)
CF
B9
DBS1
DBS0
RDIV
WR
RD
SYNC
IACK
I1 I0
No change
Exception processing factor sampling
Exception processing factor sampling
DIV
Exception processing by zero division
PUSH ALE

Fig. 3.5.4.2 Exception processing sequence - zero division -

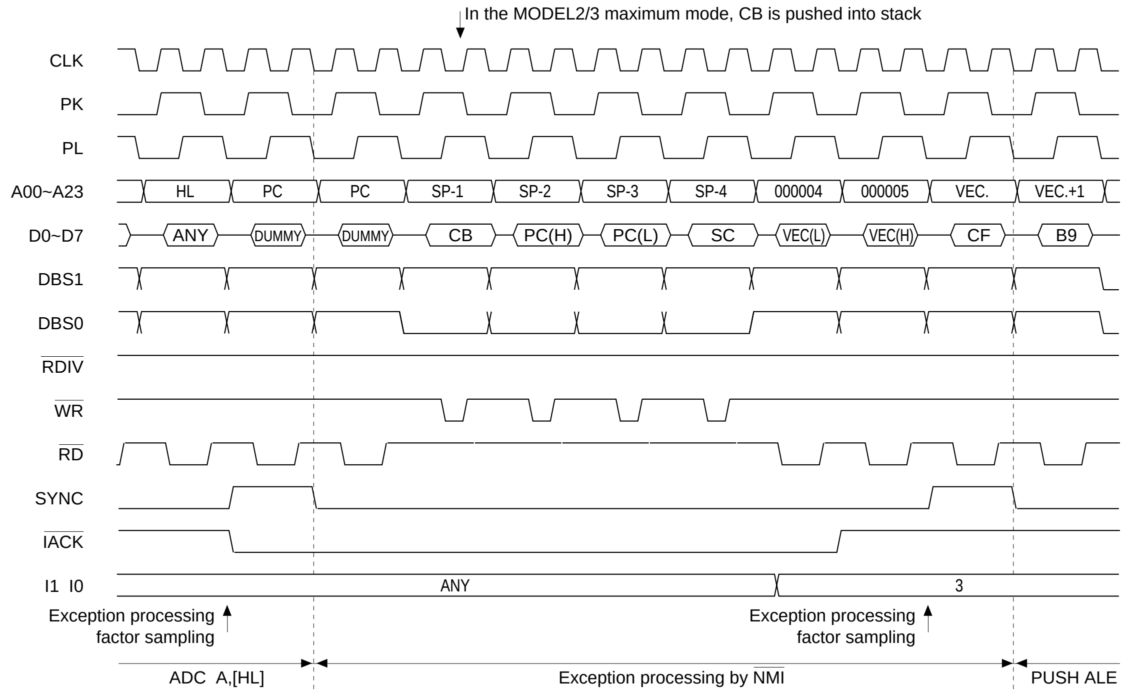


Fig. 3.5.4.3 Exception processing sequence - $\overline{NMI}$ -

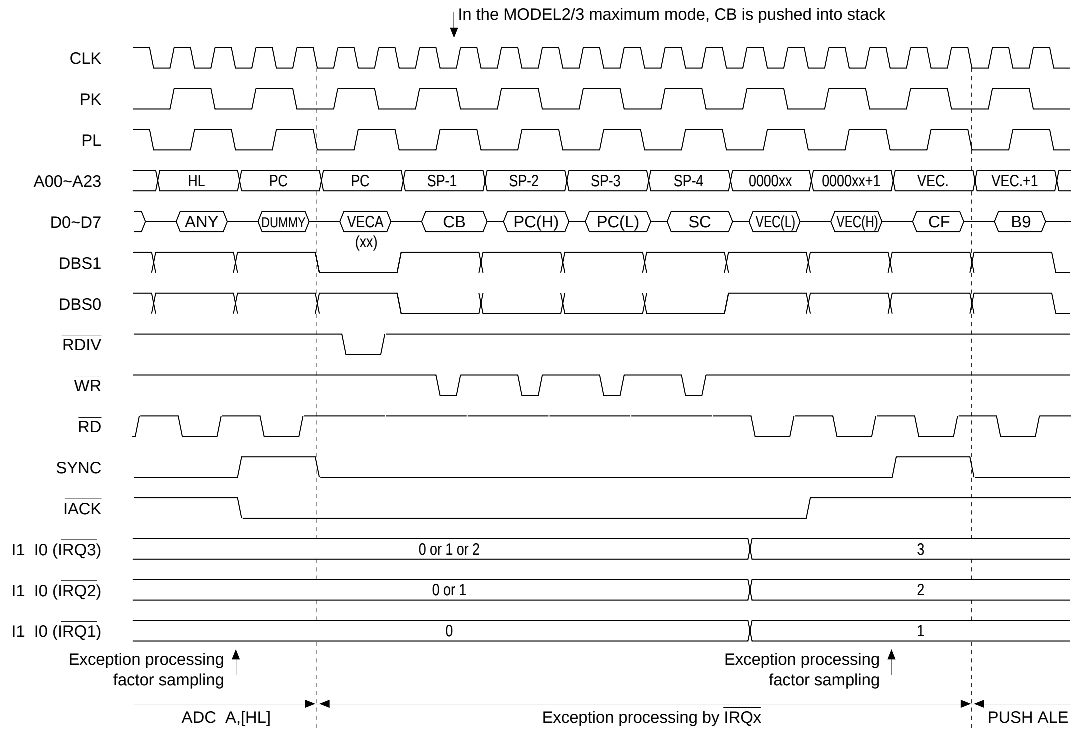


Fig. 3.5.4.4 Exception processing sequence - $\overline{IRQ1}$–$\overline{IRQ3}$ -

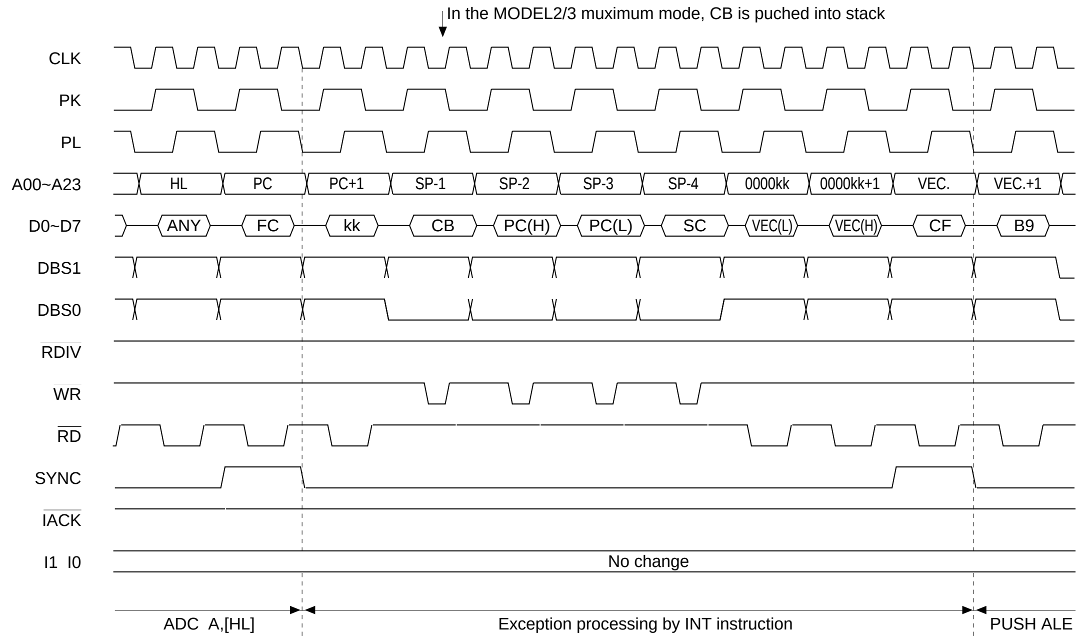


Fig. 3.5.4.5 Exception processing sequence - INT instruction -

3.6 Bus Authority Release Status

The E0C88 has a function that releases the bus for the bus authority request from outside the CPU for such operations as DMA (direct memory access) transmission.
This status, which releases the bus by responding to an external request, is called the bus authority release status.
In the bus authority release status the address bus (A00–A23), the data bus (D0–D7) and the read/write signal ($\overline{RD}$/$\overline{WR}$) become high impedance and the bus master (device external to the CPU that issued the bus authority release request) can directly access another device, such as memory, connected to the bus.

Figure 3.6.1 shows the bus authority release sequence from the program execution status.
The device that becomes the bus master inputs the low level to the $\overline{BREQ}$ terminal of the CPU, then requests a bus authority release.
The CPU for this signal samples twice, at the falling edge of the CLK of the T2 state (when WAIT has been inserted, the Tw1 state) and at the falling edge of the CLK of the T4 state, for each bus cycle. When the $\overline{BREQ}$ signal was a low level following the T2 state at the time of the sampling of the T4 state, the CPU suspends the instruction during execution in that bus cycle and sets the $\overline{BACK}$ signal to low level, then shifting to the bus authority release status. External bus master receives this $\overline{BACK}$ signal, then starts the bus control. In addition, the external bus master must maintain the $\overline{BREQ}$ signal to low level until the use of the bus from the bus authority release request terminates.

After shifting to the bus authority release status, the CPU inserts the Tz1 and Tz2 states and samples the $\overline{BREQ}$ signal at the falling edge of the CLK of the Tz2 state. The Tz1/Tz2 states are continuously inserted until high level is detected by this sampling. When high level has been detected, the CPU returns the $\overline{BACK}$ signal to high level at the rising edge of the CLK of the Tz2 state and immediately after that Tz2 state has terminated, it returns to the normal bus cycle, thus resuming the processing that had been suspended.

The bus authority release status can be inserted at the break-point of the bus cycle, in contrast to the aforementioned exception processing status can be inserted at the break-point of each instruction execution cycle.
However, during execution of the exception processing that outputs the $\overline{IACK}$ signal, a bus release request will not be accepted as long as the $\overline{IACK}$ signal is low level.

In the foregoing, we have explained about the shift from the program execution status to the bus authority release status, however, in the bus standby status as well it can shift from the HALT status to the bus authority release status.
The fact that the sampling of the bus release request signal in the HALT status is done at the falling edge of the CLK of the Th2 state (described hereafter) differs from the program execution status.
Figure 3.6.2 shows the sequence of the bus authority release from the HALT status.

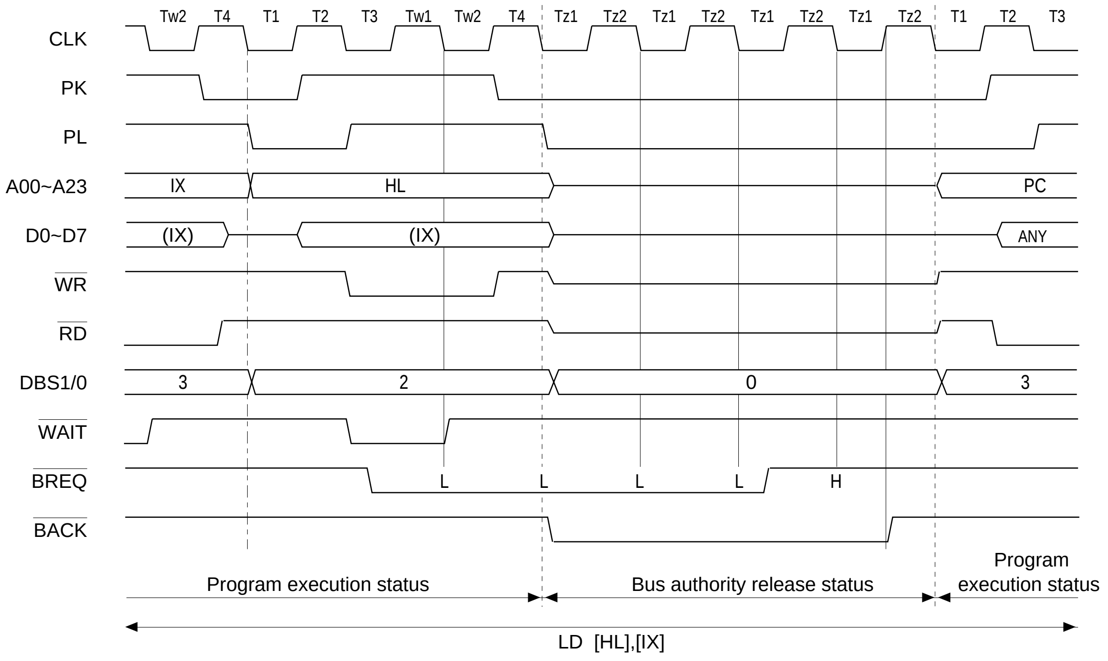


Fig. 3.6.1 Bus authority release sequence from program execution status

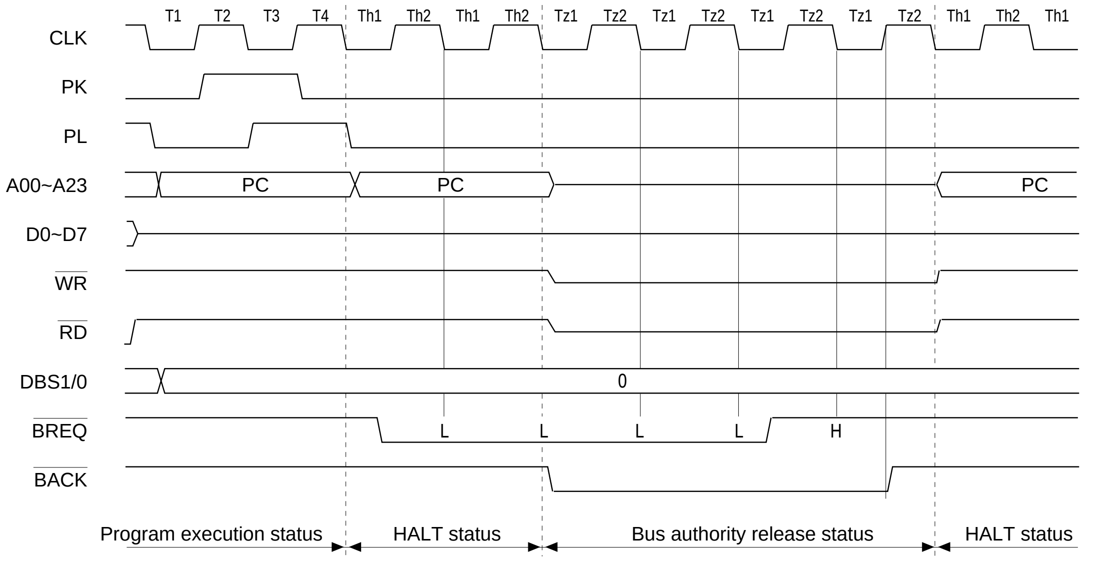


Fig. 3.6.2 Bus authority release sequence from the HALT status

3.7 Standby Status

The E0C88 has a function that stops the CPU operation and using it can greatly reduce power consumption. You can use this function to stop the CPU when there is no processing to be executed in the CPU, while there is an application program present. This is a standby status where the CPU has been stopped to shift it to low power consumption. This status, as explained below, is available in two types, a HALT status and a SLEEP status.

3.7.1 HALT status

Since only the CPU stops, you can switch to the HALT status using the "HALT" instruction.
You can shift from the HALT status to the exception processing by the optional interrupts ($\overline{\text{NMI}}$, $\overline{\text{IRQ1}}$–$\overline{\text{IRQ3}}$) and when restarted by an interrupt, a "RETE" instruction following execution of an exception processing routine causes it to resume program execution from the instruction following the "HALT" instruction. Since peripheral circuits such as oscillation circuit operate in the HALT status, it is not necessary to establish an interrupt circuit or the like for externally restarting a CPU of an MCU (E0C88 Family) and restarting can be done in an instant.
The content of registers and the like within the CPU at the point where the "HALT" instruction was executed are also held in the HALT status.

Figure 3.7.1.1 shows the sequence of shifting to the HALT status and restarting.
In the HALT status the Th1 and Th2 states are continuously inserted. During this period, interrupt sampling is done at the falling edge of the CLK of the Th2 state and the generation of an interrupt factor causes it to shift to immediate exception processing.

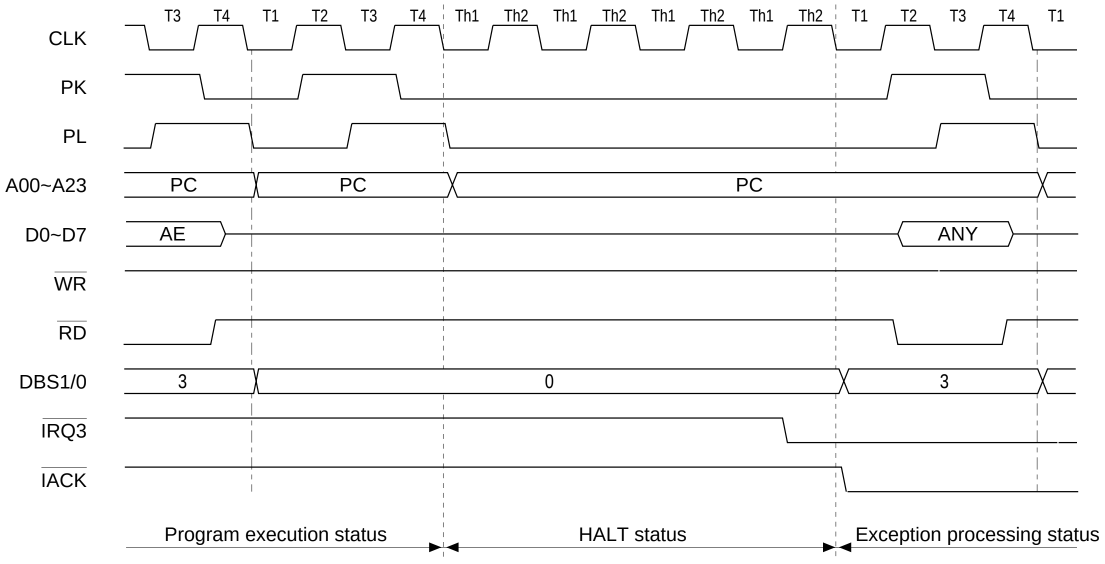


Fig. 3.7.1.1 Sequence of shifting to the HALT status and restarting

3.7.2 SLEEP status

The SLEEP status is the status where the operations of the peripheral circuits within the MCU and CPU stop and can be shifted to this status by the "SLP" instruction.
From the SLEEP status it can be shifted to the exception processing by a reset or an interrupt ($\overline{\text{NMI}}$, $\overline{\text{IRQ1}}$–$\overline{\text{IRQ3}}$) from outside the MCU. When restarted by an interrupt, the "RETE" instruction following the execution of an exception processing routine permits resumption of program execution from the instruction following the "SLP" instruction.
Power consumption in the SLEEP status can be greatly reduced in comparison with the the HALT status, because such peripheral circuits as the oscillation circuit are also stopped. However, since a safety period is needed for the oscillation circuit when restarting, it is effective when used for extended standby where instantaneous restarting is not necessary.
In the SLEEP status, as in the HALT status, the content at the time of execution of the "SLP" instruction is held for registers and the like within the CPU by impression of the rated voltage.

Figure 3.7.2.1 shows the sequence of shifting to the SLEEP status and restarting.
When an external interrupt is generated in the SLEEP status, the peripheral circuit starts to operate and the oscillation circuit also begins to oscillate.
When the oscillation starts, the CLK input to the CPU is masked by the peripheral circuit and the input to the CPU is begun after a certain stable waiting time (several 10 msec–several sec) has elapsed. The CPU samples the interrupt at the falling edge of initially input CLK and starts exception processing.

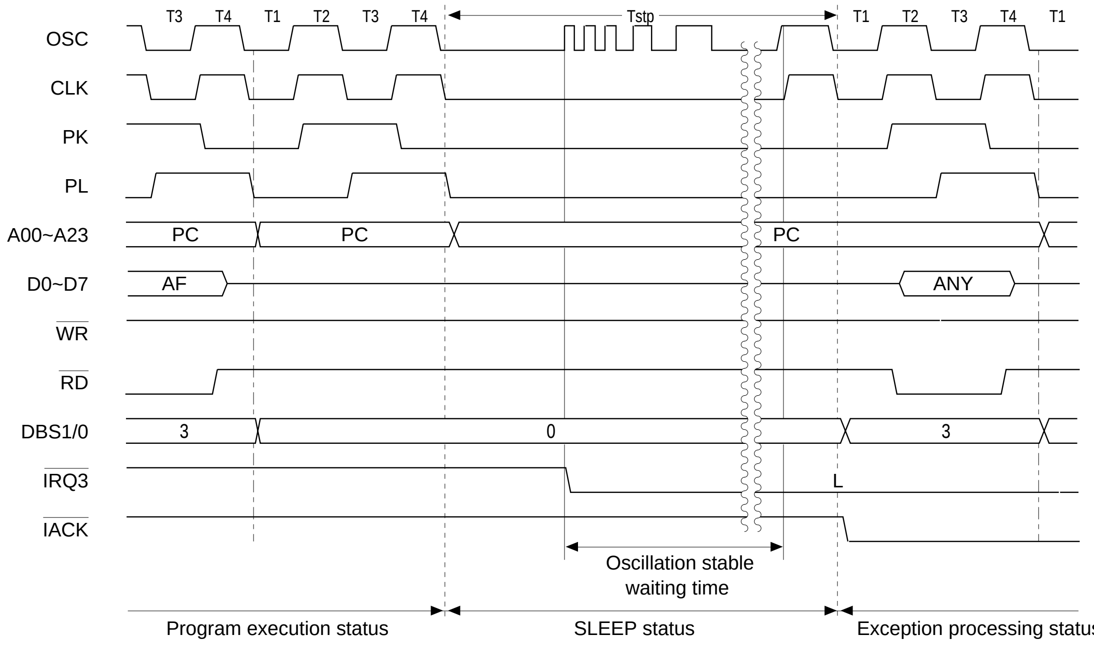


Fig. 3.7.2.1 Sequence of the shift to the SLEEP status and restarting

4 INSTRUCTION SETS

The E0C88 offers high machine cycle efficiency as well as ample, high speed instruction sets. It has 608 instructions (MODEL3) that are designed as an instruction system permitting relocatable programming.
Here we will explain about the addressing modes for memory management and about the details of each instruction.

4.1 Addressing Mode

The E0C88 has the 12 types of addressing modes that are explained here following and the address specifications corresponding to the various statuses are done concisely and accurately.
The below explanation and examples are basically focused on the source side.

Table 4.1.1 Types of addressing modes

No.	Addressing mode
1	Immediate data addressing
2	Register direct addressing
3	Register indirect addressing
4	Register indirect addressing with displacement
5	Register indirect addressing with index register
6	8-bit absolute addressing
7	16-bit absolute addressing
8	8-bit indirect addressing
9	16-bit indirect addressing
10	Signed 8-bit PC relative addressing
11	Signed 16-bit PC relative addressing
12	Implied register addressing

Immediate data addressing

Immediate data addressing is the addressing mode when immediate data is used as the operation or transmission source data. It specifies the source operand of the instruction as direct source data with 8-bit immediate data and 16-bit immediate data following the "#".
The following symbols indicate the immediate data for notation of the instruction sets.

Table 4.1.2 Immediate data symbols

Symbol	Use	Size	Range
#nn	General purpose data	8 bits	0–255
#hh	For BR setting	8 bits	0–255
#bb	For NB setting	8 bits	0–255
#pp	For page setting	8 bits	0–255
#mmnn	General purpose data	16 bits	0–65535

Example: LD A,#03H

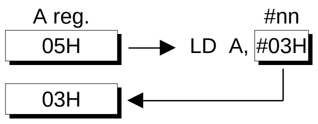


Fig. 4.1.1 Immediate data addressing

Register direct addressing

Register direct addressing is the addressing mode a register is specified as the source or destination.
It uses a register name lower then the operand for the notation of the instruction set.

Type of register notations
8-bit: A, B, L, H, BR, SC, NB, EP, XP, YP
16-bit: BA, HL, IX, IY, PC, SP, IP (YP and XP)

It can only use MODEL2/3 for NB, EP, XP, YP and IP.

When it uses this mode for the source operand, the content of the specified register becomes the source data for the operation or transmission. When used for the destination operand, such operations as the storage of data and calculations can be done for that register.

Example: LD A,B

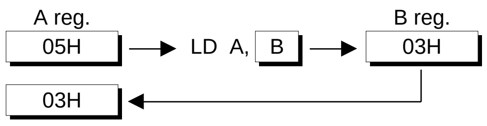


Fig. 4.1.2 Register direct addressing

Register indirect addressing

The register indirect addressing is the addressing mode for accessing the data memory and it indirectly specifies the address of the data memory by means of the index register.
There are three types of index registers used for address specification, HL, IX and IY, and their content becomes the data memory address that is accessed.
For instruction sets, the index register names are surrounded by parentheses [ ] and are thus noted as [HL], [IX] and [IY].

When it uses this mode for the source operand, the content of the specified index register becomes the address of the data memory and the content stored in that address becomes the source data. When used for the destination operand, such operations as the storage of data and calculations can be done for the specified data memory.

In MODEL2/3, specification of the page section is also necessary and the expand page registers EP (for HL), XP (for IX) and YP (for IY) are used for this purpose.

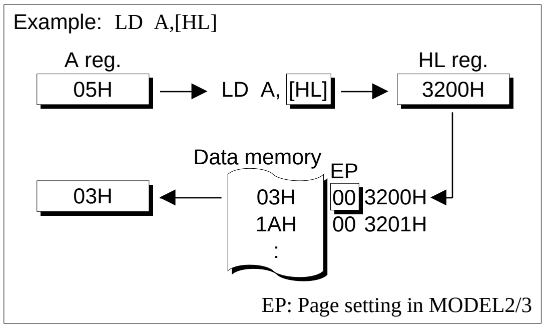


Fig. 4.1.3 Register indirect addressing

Register indirect addressing with displacement

Register indirect addressing with displacement is the addressing mode for accessing the data memory and it specifies the data memory address by displacement with the register. The data memory address becomes the value resulting from the adding of the displacement (signed 8-bit data, -128–127) to the content of the specified register. The registers used for address specification are IX, IY and SP. They use the symbol dd for displacement by the signed 8-bit data and are noted as [IX+dd], [IY+dd] and [SP+dd].

When this mode has been used as the source operand, the value resulting from adding displacement to the content of the specified register becomes the data memory address and the content stored in that address becomes the source data.
When used for the destination operand, such operations as the storage of data and calculations can be done for the specified data memory.

In MODEL2/3, it is also necessary to specify the page section and the expand page registers XP (for IX) and YP (for IY) are used for this purpose. When using a SP (stack pointer), the content of the page register for the SP that is set for peripheral circuit of each model is used for the page specification.

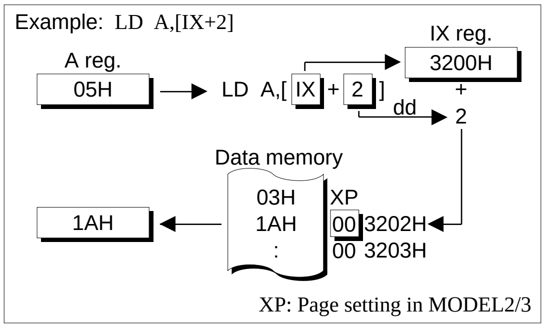


Fig. 4.1.4 Register indirect addressing with displacement

Register indirect addressing with index register

Register indirect addressing with index register is the same mode as the register indirect addressing with displacement and uses content of the L register rather than 8-bit data for displacement.
In this case, the content of the L register is handled as signed 8-bit data (-128–127).
Index registers IX and IY are used for address specification and the register used as displacement is fixed as the L register. [IX+L] and [IY+L] are noted for the instruction sets.

In MODEL2/3, specification of the page section as well is necessary and the expand page register XP (for IX) and YP (for IY) are used for this purpose.

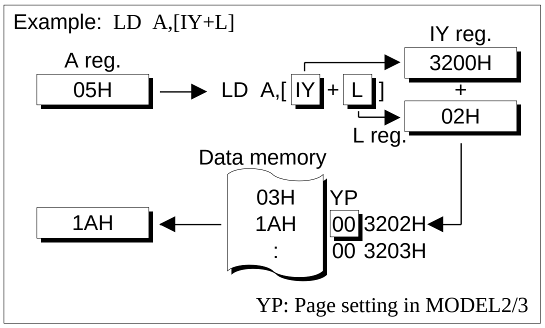


Fig. 4.1.5 Register indirect addressing with index register

8-bit absolute addressing

8-bit absolute addressing is the addressing mode for accessing the data memory and it directly specifies the lower 8 bits of the address according to the 8-bit absolute address. The upper 8 bits of the address are indirectly specified by the BR register content.
It uses the symbol *ll* for the 8-bit absolute address that specifies the address and notes it as [BR:*ll*].
When this mode has been used as the source operand, the content stored in the data memory whose address has been specified becomes the source data, making the content of the BR register the upper 8 bits of the address and specified the 8-bit absolute address as the lower 8 bits. When a destination operand has been used such operations as storage of data and calculations can be done for the specified data memory.

In MODEL2/3, it is also necessary to specify a page section and the expand page register EP is used for this purpose.

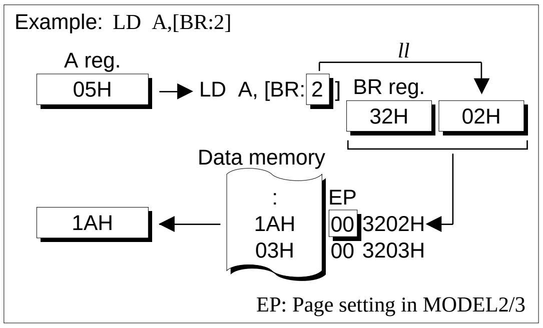


Fig. 4.1.6 8-bit absolute addressing

16-bit absolute addressing

The 16-bit absolute addressing is an addressing mode for accessing the data memory and it directly specifies the address by 16-bit absolute addresses. The symbol hh*ll* is used for the 16-bit absolute address (0–65535) that performs the address specification for the instruction set and it is noted as [hh*ll*].
When this mode has been used as the source operand, the specified 16-bit absolute address becomes the direct data memory address and the content stored in that address becomes the source data. When a destination operand has been used such operations as storage of data and calculations can be done for the specified data memory.

In MODEL2/3, it is also necessary to specify a page section and the expand page register EP is used for this purpose.

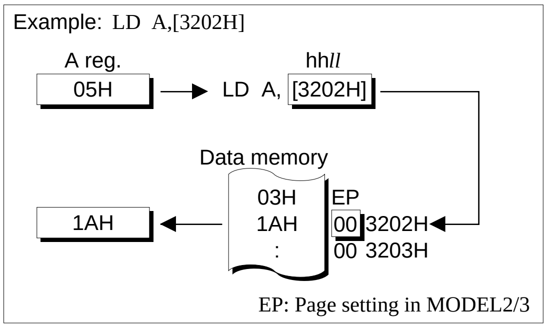


Fig. 4.1.7 16-bit absolute addressing

8-bit indirect addressing

8-bit indirect addressing is the addressing mode that uses the content of the vector field (000000H–0000FFH) as the branch destination address for the branch instruction and it specifies the vector address with an 8-bit absolute address. It branches by loading the content of the specified memory address into the lower 8 bits of the PC (program counter) and the content of the following address into the upper 8 bits of the PC.

In MODEL2/3, the branch destination bank can also be selected by setting the NB register.
The symbol kk is used for the 8-bit absolute address (0–255) that does the address specification and it is noted as [kk].
There are two types of instructions for this addressing mode, "JP [kk]" and "INT [kk]".

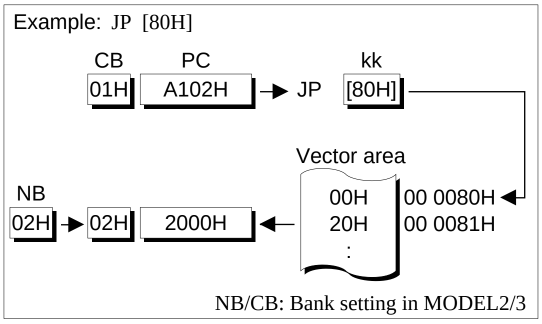


Fig. 4.1.8 8-bit indirect addressing

16-bit indirect addressing

16-bit indirect addressing is the addressing mode of the "CALL [hh*ll*]" instruction and it indirectly specifies the branch destination address by the 16-bit absolute address (0–65535). It branches the content of the specified data memory address to the lower 8 bits of the PC (program counter) and the content of the following address to the upper 8 bits of the PC.

In MODEL2/3, it is also necessary to specify a page section and the expand page register EP is used for this purpose. The branch destination bank can also be selected by setting the NB register.

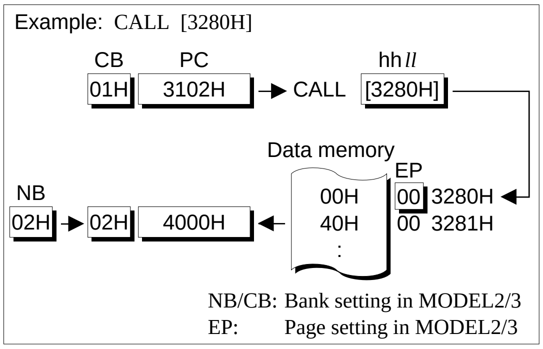


Fig. 4.1.9 16-bit indirect addressing

Signed 8-bit PC relative addressing

Signed 8-bit PC relative addressing is the addressing mode used by the branch instruction. A signed 8-bit PC relative value (-128–127) specified by an operand is added to the PC at that time and it branches to that address.

The PC value at that time becomes as follows.
2 bytes instruction: PC = instruction top address + 1
3 bytes instruction: PC = instruction top address + 2

For notation of the instruction set, it uses the symbol rr for signed 8-bit PC relative address (-128–127).

In MODEL2/3, the branch destination bank can also be selected by setting the NB register.

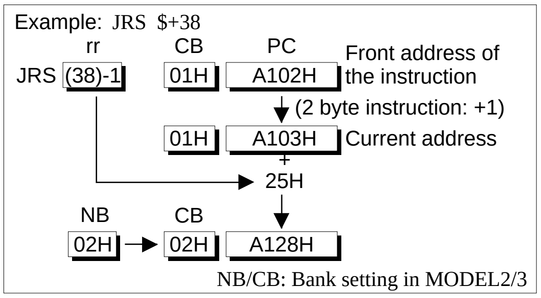


Fig. 4.1.10 Signed 8-bit PC relative addressing

Signed 16-bit PC relative addressing

Signed 16-bit PC relative addressing is the addressing mode used by the branch instruction. A signed 16-bit PC relative value (-32768–32767) specified by an operand is added to the PC at that time and it branches to that address.
The PC value at that time becomes the instruction top address + 2.

For notation of the instruction set, it uses the symbol qqrr for signed 16-bit PC relative address (-32768–32767).

In MODEL2/3, the branch destination bank can also be selected by setting the NB register.

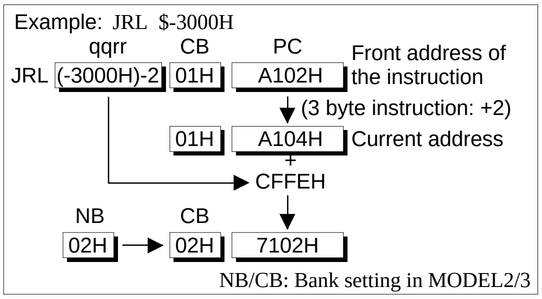


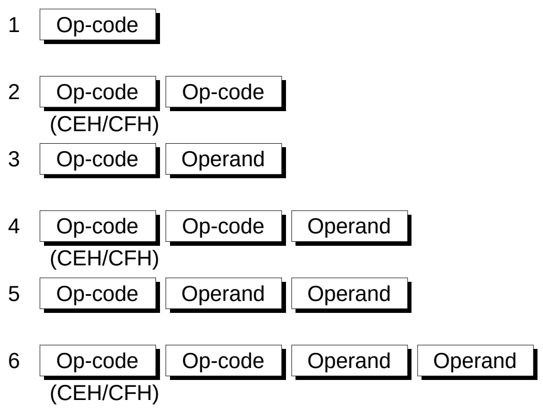
Fig. 4.1.11 Signed 16-bit PC relative addressing

Implied register addressing

The implied register addressing does not have an operand, but rather becomes the register direct addressing implicitly specified by the register. There are five types of instructions for this addressing mode, MLT, DIV, SEP, PACK and UPCK.

4.2 Instruction Format

One instruction of the E0C88 is configured as follows by a 1 byte to 4 bytes code.



1 Op-code
2 Op-code Op-code
(CEH/CFH)
3 Op-code Operand
4 Op-code Op-code Operand
(CEH/CFH)
5 Op-code Operand Operand
6 Op-code Op-code Operand Operand
(CEH/CFH)

Fig. 4.2.1 Instruction format

Op-code

The instruction set of the E0C88 has 608 types (MODEL3) of instructions and it cannot express all the instructions in 1 byte op-code (operation code). Hence, it makes CEH and CFH of the code into an expanded and uses it for the first op-code and expands the instruction by making the following 1 byte the second op-code. The 16-bit arithmetic/ transfer instructions and stack control instructions are expanded by using code CFH, and the other instructions are expanded by using code CEH.

The addressing mode for each instruction is specified by the lower 3 bits of the first op-code or second op-code. The instructions for register direct addressing, register indirect addressing, register indirect addressing with index register are composed of op-codes alone.

Operands

The instructions for 8-bit immediate data addressing, register indirect addressing with displacement, 8-bit absolute addressing (when the source has been specified by register), 8-bit indirect addressing and signed 8-bit PC relative addressing have 1 byte operand and the values specified by the 8-bit data as they are, become operands.

The instructions for 16-bit immediate data addressing, 8-bit absolute addressing (when the source has been specified by immediate data), 16-bit absolute addressing, 16-bit indirect addressing and signed 16-bit PC relative addressing have 2-byte operands and the lower 8 bits of the value specified by the 16-bit data becomes the first operand and the upper 8 bits becomes the second operand. (In case of the 8-bit absolute addressing, the address specification becomes the first operand and the immediate data becomes the second operand.)

4.3 Instruction Set List

Here has been provided a list classifying the instruction sets of the E0C88 by function.
Since a list by addressing modes is also provided in the "APPENDIX," you should refer to them as necessary.

4.3.1 Function classification

Table 4.3.1.1 indicates the function classifications of the instructions.

Table 4.3.1.1 Instruction function classifications

Function classification	Mnemonic	Operation
8-bit arithmetic	ADD	Addition
and logic	ADC	Addition with carry
operation	SUB	Subtraction
	SBC	Subtraction with carry
	AND	Logical product
	OR	Logical sum
	XOR	Exclusive OR
	CP	Comparison
	BIT	Bit test
	INC	1 increment
	DEC	1 decrement
	MLT	Multiplication
	DIV	Division
	CPL	Complement of 1
	NEG	Complement of 2
8-bit transfer	LD	Load
	EX	Byte exchange
	SWAP	Nibble exchange
Rotate/shift	RL	Rotate to left with carry
	RLC	Rotate to left
	RR	Rotate to right with carry
	RRC	Rotate to right
	SLA	Arithmetic shift to left
	SLL	Logical shift to left
	SRA	Arithmetic shift to right
	SRL	Logical shift to right

Function classification	Mnemonic	Operation
Auxiliary	PACK	Pack
operation	UPCK	Unpack
	SEP	Code extension
16-bit arithmetic	ADD	Addition
operation	ADC	Addition with carry
	SUB	Subtraction
	SBC	Subtraction with carry
	CP	Comparison
	INC	1 increment
16-bit transfer	DEC	1 decrement
	LD	Load
Stack control	EX	Word exchange
	PUSH	Push
Branch	POP	Pop
	JRS	Relative short jump
	JRL	Relative long jump
	JP	Indirect jump
	DJR	Loop
	CARS	Relative short call
	CARL	Relative long call
	CALL	Indirect call
	RET	Return
	RETE	Exception processing return
	RETS	Return and skip
System control	INT	Software interrupt
	NOP	No operation
	HALT	Shifts to HALT status
	SLP	Shifts to SLEEP status

4.3.2 *Symbol meanings*

Table 4.3.2.1 indicates the meanings of the symbols used in the instruction list by function for the following items.

Table 4.3.2.1 Symbol meanings

Register relationship	
A	Data register A
A(H)	Upper 4 bits of A register
A(L)	Lower 4 bits of A register
B	Data register B
BA	BA pair register
H	Data register H
L	Data register L
HL	Index register HL
IX	Index register IX
IX(H)	Upper 8 bits of IX register
IX(L)	Lower 8 bits of IX register
IY	Index register IY
IY(H)	Upper 8 bits of IY register
IY(L)	Lower 8 bits of IY register
SP	Stack pointer SP
BR	Base register BR
SC	System condition flag SC
CC	Customize condition flag CC
PC	Program counter PC
PC(H)	Upper 8 bits of PC
PC(L)	Lower 8 bits of PC
NB	New code bank register NB
CB	Code bank register CB
EP	Expand page register EP
XP	Expand page register XP for IX
YP	Expand page register YP for IY
IP	XP and YP register

Immediate data	
nn	8-bit immediate data (unsigned)
hh	Absolute address (upper 8 bits) setting data (unsigned)
ll	Absolute address (lower 8 bits) setting data (unsigned)
pp	Page setting data (unsigned)
bb	Bank setting data (unsigned)
dd	Signed 8-bit displacement
rr	8-bit relative address setting data (signed)
kk	Vector address setting data (unsigned)
mmnn	16-bit immediate data (unsigned)
hh*ll*	16-bit absolute address setting data (unsigned)
qqrr	16-bit relative address setting data (signed)

Memory relationship	
[HL]	Memory specified by HL register
[HL](H)	Upper 4 bits of [HL]
[HL](L)	Lower 4 bits of [HL]
[HL]	Memory specified by HL register
[IX]	Memory specified by IX register
[IX+dd]	Memory specified by IX register + dd
[IX+L]	Memory specified by IX register + L register
[IY]	Memory specified by IY register
[IY+dd]	Memory specified by IY register + dd
[IY+L]	Memory specified by IY register + L register
[BR:*ll*]	Memory specified by BR register and "*ll*"
[hh*ll*]	Memory specified by "hh*ll*"
[kk]	Vector specified by "kk"
[SP]	Stack specified by SP
[SP+dd]	Stack specified by SP+ dd

Flag relationship	
Z	Zero flag
C	Carry flag
V	Overflow flag
N	Negative flag
D	Decimal flag
U	Unpack flag
I0	Interrupt flag 0
I1	Interrupt flag 1
↕	Setting/resetting of flag
–	No change
0	Resetting of flag
F0	Customize condition flag F0
F1	Customize condition flag F1
F2	Customize condition flag F2
F3	Customize condition flag F3

Calculation operations and other	
+	Addition
-	Subtraction
*	Multiplication
/	Division
∧	Logical product
∨	Logical sum
∀	Exclusive OR
★	Instruction permitting decimal and unpack operation

4.3.3 *Instruction list by functions*

8-bit Trnsfer Instructions (1/3)

Mnemonic		Machine Code	Operation	Cycle	Byte	SC								Comment	Page
						I1	I0	U	D	N	V	C	Z		
LD	A,A	40	A←A	1	1	–	–	–	–	–	–	–	–		*115*
	A,B	41	A←B	1	1	–	–	–	–	–	–	–	–		*115*
	A,L	42	A←L	1	1	–	–	–	–	–	–	–	–		*115*
	A,H	43	A←H	1	1	–	–	–	–	–	–	–	–		*115*
	A,BR	CE,C0	A←BR	2	2	–	–	–	–	–	–	–	–		*115*
	A,SC	CE,C1	A←SC	2	2	–	–	–	–	–	–	–	–		*115*
	A,#nn	B0,nn	A←nn	2	2	–	–	–	–	–	–	–	–		*122*
	A,[BR:*ll*]	44,*ll*	A←[BR:*ll*]	3	2	–	–	–	–	–	–	–	–		*125*
	A,[hh*ll*]	CE,D0,*ll*,hh	A←[hh*ll*]	5	4	–	–	–	–	–	–	–	–		*127*
	A,[HL]	45	A←[HL]	2	1	–	–	–	–	–	–	–	–		*127*
	A,[IX]	46	A←[IX]	2	1	–	–	–	–	–	–	–	–		*129*
	A,[IY]	47	A←[IY]	2	1	–	–	–	–	–	–	–	–		*130*
	A,[IX+dd]	CE,40,dd	A←[IX+dd]	4	3	–	–	–	–	–	–	–	–		*132*
	A,[IY+dd]	CE,41,dd	A←[IY+dd]	4	3	–	–	–	–	–	–	–	–		*133*
	A,[IX+L]	CE,42	A←[IX+L]	4	2	–	–	–	–	–	–	–	–		*135*
	A,[IY+L]	CE,43	A←[IY+L]	4	2	–	–	–	–	–	–	–	–		*136*
	A,NB	CE,C8	A←NB	2	2	–	–	–	–	–	–	–	–		*116*
	A,EP	CE,C9	A←EP	2	2	–	–	–	–	–	–	–	–	*MODEL2/3*	*116*
	A,XP	CE,CA	A←XP	2	2	–	–	–	–	–	–	–	–	*only*	*116*
	A,YP	CE,CB	A←YP	2	2	–	–	–	–	–	–	–	–		*116*
LD	B,A	48	B←A	1	1	–	–	–	–	–	–	–	–		*115*
	B,B	49	B←B	1	1	–	–	–	–	–	–	–	–		*115*
	B,L	4A	B←L	1	1	–	–	–	–	–	–	–	–		*115*
	B,H	4B	A←H	1	1	–	–	–	–	–	–	–	–		*115*
	B,#nn	B1,nn	B←nn	2	2	–	–	–	–	–	–	–	–		*122*
	B,[BR:*ll*]	4C,*ll*	B←[BR:*ll*]	3	2	–	–	–	–	–	–	–	–		*125*
	B,[hh*ll*]	CE,D1,*ll*,hh	B←[hh*ll*]	5	4	–	–	–	–	–	–	–	–		*127*
	B,[HL]	4D	B←[HL]	2	1	–	–	–	–	–	–	–	–		*127*
	B,[IX]	4E	B←[IX]	2	1	–	–	–	–	–	–	–	–		*129*
	B,[IY]	4F	B←[IY]	2	1	–	–	–	–	–	–	–	–		*130*
	B,[IX+dd]	CE,48,dd	B←[IX+dd]	4	3	–	–	–	–	–	–	–	–		*132*
	B,[IY+dd]	CE,49,dd	B←[IY+dd]	4	3	–	–	–	–	–	–	–	–		*133*
	B,[IX+L]	CE,4A	B←[IX+L]	4	2	–	–	–	–	–	–	–	–		*135*
	B,[IY+L]	CE,4B	B←[IY+L]	4	2	–	–	–	–	–	–	–	–		*136*
LD	L,A	50	L←A	1	1	–	–	–	–	–	–	–	–		*115*
	L,B	51	L←B	1	1	–	–	–	–	–	–	–	–		*115*
	L,L	52	L←L	1	1	–	–	–	–	–	–	–	–		*115*
	L,H	53	L←H	1	1	–	–	–	–	–	–	–	–		*115*
	L,#nn	B2,nn	L←nn	2	2	–	–	–	–	–	–	–	–		*122*
	L,[BR:*ll*]	54,*ll*	L←[BR:*ll*]	3	2	–	–	–	–	–	–	–	–		*125*
	L,[hh*ll*]	CE,D2,*ll*,hh	L←[hh*ll*]	5	4	–	–	–	–	–	–	–	–		*127*
	L,[HL]	55	L←[HL]	2	1	–	–	–	–	–	–	–	–		*127*
	L,[IX]	56	L←[IX]	2	1	–	–	–	–	–	–	–	–		*129*
	L,[IY]	57	L←[IY]	2	1	–	–	–	–	–	–	–	–		*130*
	L,[IX+dd]	CE,50,dd	L←[IX+dd]	4	3	–	–	–	–	–	–	–	–		*132*
	L,[IY+dd]	CE,51,dd	L←[IY+dd]	4	3	–	–	–	–	–	–	–	–		*133*
	L,[IX+L]	CE,52	L←[IX+L]	4	2	–	–	–	–	–	–	–	–		*135*
	L,[IY+L]	CE,53	L←[IY+L]	4	2	–	–	–	–	–	–	–	–		*136*

* New code bank register NB and expand page registers EP/XP/YP are set only for MODEL2/3.
In MODEL0/1, instructions that access these registers cannot be used.

8-bit Trnsfer Instructions (2/3)

Mnemonic		Machine Code	Operation	Cycle	Byte	SC								Comment	Page
						I1	I0	U	D	N	V	C	Z		
LD	H,A	58	H←A	1	1	–	–	–	–	–	–	–	–		*115*
	H,B	59	H←B	1	1	–	–	–	–	–	–	–	–		*115*
	H,L	5A	H←L	1	1	–	–	–	–	–	–	–	–		*115*
	H,H	5B	H←H	1	1	–	–	–	–	–	–	–	–		*115*
	H,#nn	B3,nn	H←nn	2	2	–	–	–	–	–	–	–	–		*122*
	H,[BR:*ll*]	5C,*ll*	H←[BR:*ll*]	3	2	–	–	–	–	–	–	–	–		*125*
	H,[hh*ll*]	CE,D3,*ll*,hh	H←[hh*ll*]	5	4	–	–	–	–	–	–	–	–		*127*
	H,[HL]	5D	H←[HL]	2	1	–	–	–	–	–	–	–	–		*127*
	H,[IX]	5E	H←[IX]	2	1	–	–	–	–	–	–	–	–		*129*
	H,[IY]	5F	H←[IY]	2	1	–	–	–	–	–	–	–	–		*130*
	H,[IX+dd]	CE,58,dd	H←[IX+dd]	4	3	–	–	–	–	–	–	–	–		*132*
	H,[IY+dd]	CE,59,dd	H←[IY+dd]	4	3	–	–	–	–	–	–	–	–		*133*
	H,[IX+L]	CE,5A	H←[IX+L]	4	2	–	–	–	–	–	–	–	–		*135*
	H,[IY+L]	CE,5B	H←[IY+L]	4	2	–	–	–	–	–	–	–	–		*136*
LD	BR,A	CE,C2	BR←A	2	2	–	–	–	–	–	–	–	–		*116*
	BR,#hh	B4,hh	BR←hh	2	2	–	–	–	–	–	–	–	–		*122*
LD	SC,A	CE,C3	SC←A	3	2	↕	↕	↕	↕	↕	↕	↕	↕		*116*
	SC,#nn	9F,nn	SC←nn	3	2	↕	↕	↕	↕	↕	↕	↕	↕		*122*
LD	[BR:*ll*],A	78,*ll*	[BR:*ll*]←A	3	2	–	–	–	–	–	–	–	–		*117*
	[BR:*ll*],B	79,*ll*	[BR:*ll*]←B	3	2	–	–	–	–	–	–	–	–		*117*
	[BR:*ll*],L	7A,*ll*	[BR:*ll*]←L	3	2	–	–	–	–	–	–	–	–		*117*
	[BR:*ll*],H	7B,*ll*	[BR:*ll*]←H	3	2	–	–	–	–	–	–	–	–		*117*
	[BR:*ll*],#nn	DD,*ll*,nn	[BR:*ll*]←nn	4	3	–	–	–	–	–	–	–	–		*124*
	[BR:*ll*],[HL]	7D,*ll*	[BR:*ll*]←[HL]	4	2	–	–	–	–	–	–	–	–		*128*
	[BR:*ll*],[IX]	7E,*ll*	[BR:*ll*]←[IX]	4	2	–	–	–	–	–	–	–	–		*129*
	[BR:*ll*],[IY]	7F,*ll*	[BR:*ll*]←[IY]	4	2	–	–	–	–	–	–	–	–		*131*
LD	[hh*ll*],A	CE,D4,*ll*,hh	[hh*ll*]←A	5	4	–	–	–	–	–	–	–	–		*118*
	[hh*ll*],B	CE,D5,*ll*,hh	[hh*ll*]←B	5	4	–	–	–	–	–	–	–	–		*118*
	[hh*ll*],L	CE,D6,*ll*,hh	[hh*ll*]←L	5	4	–	–	–	–	–	–	–	–		*118*
	[hh*ll*],H	CE,D7,*ll*,hh	[hh*ll*]←H	5	4	–	–	–	–	–	–	–	–		*118*
LD	[HL],A	68	[HL]←A	2	1	–	–	–	–	–	–	–	–		*118*
	[HL],B	69	[HL]←B	2	1	–	–	–	–	–	–	–	–		*118*
	[HL],L	6A	[HL]←L	2	1	–	–	–	–	–	–	–	–		*118*
	[HL],H	6B	[HL]←H	2	1	–	–	–	–	–	–	–	–		*118*
	[HL],#nn	B5,nn	[HL]←nn	3	2	–	–	–	–	–	–	–	–		*124*
	[HL],[BR:*ll*]	6C,*ll*	[HL]←[BR:*ll*]	4	2	–	–	–	–	–	–	–	–		*126*
	[HL],[HL]	6D	[HL]←[HL]	3	1	–	–	–	–	–	–	–	–		*128*
	[HL],[IX]	6E	[HL]←[IX]	3	1	–	–	–	–	–	–	–	–		*129*
	[HL],[IY]	6F	[HL]←[IY]	3	1	–	–	–	–	–	–	–	–		*131*
	[HL],[IX+dd]	CE,60,dd	[HL]←[IX+dd]	5	3	–	–	–	–	–	–	–	–		*132*
	[HL],[IY+dd]	CE,61,dd	[HL]←[IY+dd]	5	3	–	–	–	–	–	–	–	–		*134*
	[HL],[IX+L]	CE,62	[HL]←[IX+L]	5	2	–	–	–	–	–	–	–	–		*135*
	[HL],[IY+L]	CE,63	[HL]←[IY+L]	5	2	–	–	–	–	–	–	–	–		*137*
LD	[IX],A	60	[IX]←A	2	1	–	–	–	–	–	–	–	–		*119*
	[IX],B	61	[IX]←B	2	1	–	–	–	–	–	–	–	–		*119*
	[IX],L	62	[IX]←L	2	1	–	–	–	–	–	–	–	–		*119*
	[IX],H	63	[IX]←H	2	1	–	–	–	–	–	–	–	–		*119*
	[IX],#nn	B6,nn	[IX]←nn	3	2	–	–	–	–	–	–	–	–		*125*

8-bit Trnsfer Instructions (3/3)

Mnemonic		Machine Code	Operation	Cycle	Byte	SC								Comment	Page
						I1	I0	U	D	N	V	C	Z		
LD	[IX],[BR:*ll*]	64,*ll*	[IX]←[BR:*ll*]	4	2	–	–	–	–	–	–	–	–		*126*
	[IX],[HL]	65	[IX]←[HL]	3	1	–	–	–	–	–	–	–	–		*128*
	[IX],[IX]	66	[IX]←[IX]	3	1	–	–	–	–	–	–	–	–		*130*
	[IX],[IY]	67	[IX]←[IY]	3	1	–	–	–	–	–	–	–	–		*131*
	[IX],[IX+dd]	CE,68,dd	[IX]←[IX+dd]	5	3	–	–	–	–	–	–	–	–		*133*
	[IX],[IY+dd]	CE,69,dd	[IX]←[IY+dd]	5	3	–	–	–	–	–	–	–	–		*134*
	[IX],[IX+L]	CE,6A	[IX]←[IX+L]	5	2	–	–	–	–	–	–	–	–		*136*
	[IX],[IY+L]	CE,6B	[IX]←[IY+L]	5	2	–	–	–	–	–	–	–	–		*137*
LD	[IY],A	70	[IY]←A	2	1	–	–	–	–	–	–	–	–		*119*
	[IY],B	71	[IY]←B	2	1	–	–	–	–	–	–	–	–		*119*
	[IY],L	72	[IY]←L	2	1	–	–	–	–	–	–	–	–		*119*
	[IY],H	73	[IY]←H	2	1	–	–	–	–	–	–	–	–		*119*
	[IY],#nn	B7,nn	[IY]←nn	3	2	–	–	–	–	–	–	–	–		*125*
	[IY],[BR:*ll*]	74,ll	[IY]←[BR:*ll*]	4	2	–	–	–	–	–	–	–	–		*126*
	[IY],[HL]	75	[IY]←[HL]	3	1	–	–	–	–	–	–	–	–		*128*
	[IY],[IX]	76	[IY]←[IX]	3	1	–	–	–	–	–	–	–	–		*130*
	[IY],[IY]	77	[IY]←[IY]	3	1	–	–	–	–	–	–	–	–		*131*
	[IY],[IX+dd]	CE,78,dd	[IY]←[IX+dd]	5	3	–	–	–	–	–	–	–	–		*133*
	[IY],[IY+dd]	CE,79,dd	[IY]←[IY+dd]	5	3	–	–	–	–	–	–	–	–		*134*
	[IY],[IX+L]	CE,7A	[IY]←[IX+L]	5	2	–	–	–	–	–	–	–	–		*136*
	[IY],[IY+L]	CE,7B	[IY]←[IY+L]	5	2	–	–	–	–	–	–	–	–		*137*
LD	[IX+dd],A	CE,44,dd	[IX+dd]←A	4	3	–	–	–	–	–	–	–	–		*120*
	[IX+dd],B	CE,4C,dd	[IX+dd]←B	4	3	–	–	–	–	–	–	–	–		*120*
	[IX+dd],L	CE,54,dd	[IX+dd]←L	4	3	–	–	–	–	–	–	–	–		*120*
	[IX+dd],H	CE,5C,dd	[IX+dd]←H	4	3	–	–	–	–	–	–	–	–		*120*
LD	[IY+dd],A	CE,45,dd	[IY+dd]←A	4	3	–	–	–	–	–	–	–	–		*120*
	[IY+dd],B	CE,4D,dd	[IY+dd]←B	4	3	–	–	–	–	–	–	–	–		*120*
	[IY+dd],L	CE,55,dd	[IY+dd]←L	4	3	–	–	–	–	–	–	–	–		*120*
	[IY+dd],H	CE,5D,dd	[IY+dd]←H	4	3	–	–	–	–	–	–	–	–		*120*
LD	[IX+L],A	CE,46	[IX+L]←A	4	2	–	–	–	–	–	–	–	–		*121*
	[IX+L],B	CE,4E	[IX+L]←B	4	2	–	–	–	–	–	–	–	–		*121*
	[IX+L],L	CE,56	[IX+L]←L	4	2	–	–	–	–	–	–	–	–		*121*
	[IX+L],H	CE,5E	[IX+L]←H	4	2	–	–	–	–	–	–	–	–		*121*
LD	[IY+L],A	CE,47	[IY+L]←A	4	2	–	–	–	–	–	–	–	–		*121*
	[IY+L],B	CE,4F	[IY+L]←B	4	2	–	–	–	–	–	–	–	–		*121*
	[IY+L],L	CE,57	[IY+L]←L	4	2	–	–	–	–	–	–	–	–		*121*
	[IY+L],H	CE,5F	[IY+L]←H	4	2	–	–	–	–	–	–	–	–		*121*
LD	NB,A	CE,CC	NB←A	3	2	–	–	–	–	–	–	–	–		*117*
	NB,#bb	CE,C4,bb	NB←bb	4	3	–	–	–	–	–	–	–	–		*123*
LD	EP,A	CE,CD	EP←A	2	2	–	–	–	–	–	–	–	–		*117*
	EP,#pp	CE,C5,pp	EP←pp	3	3	–	–	–	–	–	–	–	–	*MODEL2/3*	*123*
LD	XP,A	CE,CE	XP←A	2	2	–	–	–	–	–	–	–	–	*only*	*117*
	XP,#pp	CE,C6,pp	XP←pp	3	3	–	–	–	–	–	–	–	–		*123*
LD	YP,A	CE,CF	YP←A	2	2	–	–	–	–	–	–	–	–		*117*
	YP,#pp	CE,C7,pp	YP←pp	3	3	–	–	–	–	–	–	–	–		*124*
EX	A,B	CC	A↔B	2	1	–	–	–	–	–	–	–	–		*105*
	A,[HL]	CD	A↔[HL]	3	1	–	–	–	–	–	–	–	–		*105*
SWAP	A	F6	A(H)↔A(L)	2	1	–	–	–	–	–	–	–	–		*188*
	[HL]	F7	[HL](H)↔[HL](L)	3	1	–	–	–	–	–	–	–	–		*188*

* New code bank register NB and expand page registers EP/XP/YP are set only for MODEL2/3.
In MODEL0/1, instructions that access these registers cannot be used.

16-bit Trnsfer Instructions (1/2)

Mnemonic		Machine Code	Operation	Cycle	Byte	SC								Comment	Page
						I1	I0	U	D	N	V	C	Z		
LD	BA,BA	CF,E0	BA←BA	2	2	–	–	–	–	–	–	–	–		*138*
	BA,HL	CF,E1	BA←HL	2	2	–	–	–	–	–	–	–	–		*138*
	BA,IX	CF,E2	BA←IX	2	2	–	–	–	–	–	–	–	–		*138*
	BA,IY	CF,E3	BA←IY	2	2	–	–	–	–	–	–	–	–		*138*
	BA,SP	CF,F8	BA←SP	2	2	–	–	–	–	–	–	–	–		*138*
	BA,PC	CF,F9	BA←PC+2	2	2	–	–	–	–	–	–	–	–		*138*
	BA,#mmnn	C4,nn,mm	BA←mmnn	3	3	–	–	–	–	–	–	–	–		*143*
	BA,[hh*ll*]	B8,*ll*,hh	A←[hh*ll*], B←[hh*ll*+1]	5	3	–	–	–	–	–	–	–	–		*144*
	BA,[HL]	CF,C0	A←[HL], B←[HL+1]	5	2	–	–	–	–	–	–	–	–		*145*
	BA,[IX]	CF,D0	A←[IX], B←[IX+1]	5	2	–	–	–	–	–	–	–	–		*146*
	BA,[IY]	CF,D8	A←[IY], B←[IY+1]	5	2	–	–	–	–	–	–	–	–		*146*
	BA,[SP+dd]	CF,70,dd	A←[SP+dd], B←[SP+dd+1]	6	3	–	–	–	–	–	–	–	–		*147*
LD	HL,BA	CF,E4	HL←BA	2	2	–	–	–	–	–	–	–	–		*138*
	HL,HL	CF,E5	HL←HL	2	2	–	–	–	–	–	–	–	–		*138*
	HL,IX	CF,E6	HL←IX	2	2	–	–	–	–	–	–	–	–		*138*
	HL,IY	CF,E7	HL←IY	2	2	–	–	–	–	–	–	–	–		*138*
	HL,SP	CF,F4	HL←SP	2	2	–	–	–	–	–	–	–	–		*139*
	HL,PC	CF,F5	HL←PC+2	2	2	–	–	–	–	–	–	–	–		*139*
	HL,#mmnn	C5,nn,mm	HL←mmnn	3	3	–	–	–	–	–	–	–	–		*143*
	HL,[hh*ll*]	B9,*ll*,hh	L←[hh*ll*], H←[hh*ll*+1]	5	3	–	–	–	–	–	–	–	–		*144*
	HL,[HL]	CF,C1	L←[HL], H←[HL+1]	5	2	–	–	–	–	–	–	–	–		*145*
	HL,[IX]	CF,D1	L←[IX], H←[IX+1]	5	2	–	–	–	–	–	–	–	–		*146*
	HL,[IY]	CF,D9	L←[IY], H←[IY+1]	5	2	–	–	–	–	–	–	–	–		*146*
	HL,[SP+dd]	CF,71,dd	L←[SP+dd], H←[SP+dd+1]	6	3	–	–	–	–	–	–	–	–		*147*
LD	IX,BA	CF,E8	IX←BA	2	2	–	–	–	–	–	–	–	–		*138*
	IX,HL	CF,E9	IX←HL	2	2	–	–	–	–	–	–	–	–		*138*
	IX,IX	CF,EA	IX←IX	2	2	–	–	–	–	–	–	–	–		*138*
	IX,IY	CF,EB	IX←IY	2	2	–	–	–	–	–	–	–	–		*138*
	IX,SP	CF,FA	IX←SP	2	2	–	–	–	–	–	–	–	–		*139*
	IX,#mmnn	C6,nn,mm	IX←mmnn	3	3	–	–	–	–	–	–	–	–		*143*
	IX,[hh*ll*]	BA,*ll*,hh	IX$_{(L)}$←[hh*ll*], IX$_{(H)}$←[hh*ll*+1]	5	3	–	–	–	–	–	–	–	–		*144*
	IX,[HL]	CF,C2	IX$_{(L)}$←[HL], IX$_{(H)}$←[HL+1]	5	2	–	–	–	–	–	–	–	–		*145*
	IX,[IX]	CF,D2	IX$_{(L)}$←[IX], IX$_{(H)}$←[IX+1]	5	2	–	–	–	–	–	–	–	–		*146*
	IX,[IY]	CF,DA	IX$_{(L)}$←[IY], IX$_{(H)}$←[IY+1]	5	2	–	–	–	–	–	–	–	–		*146*
	IX,[SP+dd]	CF,72,dd	IX$_{(L)}$←[SP+dd], IX$_{(H)}$←[SP+dd+1]	6	3	–	–	–	–	–	–	–	–		*147*
LD	IY,BA	CF,EC	IY←BA	2	2	–	–	–	–	–	–	–	–		*138*
	IY,HL	CF,ED	IY←HL	2	2	–	–	–	–	–	–	–	–		*138*
	IY,IX	CF,EE	IY←IX	2	2	–	–	–	–	–	–	–	–		*138*
	IY,IY	CF,EF	IY←IY	2	2	–	–	–	–	–	–	–	–		*138*
	IY,SP	CF,FE	IY←SP	2	2	–	–	–	–	–	–	–	–		*139*
	IY,#mmnn	C7,nn,mm	IY←mmnn	3	3	–	–	–	–	–	–	–	–		*143*
	IY,[hh*ll*]	BB,*ll*,hh	IY$_{(L)}$←[hh*ll*], IY$_{(H)}$←[hh*ll*+1]	5	3	–	–	–	–	–	–	–	–		*144*
	IY,[HL]	CF,C3	IY$_{(L)}$←[HL], IY$_{(H)}$←[HL+1]	5	2	–	–	–	–	–	–	–	–		*145*
	IY,[IX]	CF,D3	IY$_{(L)}$←[IX], IY$_{(H)}$←[IX+1]	5	2	–	–	–	–	–	–	–	–		*146*
	IY,[IY]	CF,DB	IY$_{(L)}$←[IY], IY$_{(H)}$←[IY+1]	5	2	–	–	–	–	–	–	–	–		*146*
	IY,[SP+dd]	CF,73,dd	IY$_{(L)}$←[SP+dd], IY$_{(H)}$←[SP+dd+1]	6	3	–	–	–	–	–	–	–	–		*147*

16-bit Trnsfer Instructions (2/2)

Mnemonic		Machine Code	Operation	Cycle	Byte	SC								Comment	Page
						I1	I0	U	D	N	V	C	Z		
LD	SP,BA	CF,F0	SP←BA	2	2	–	–	–	–	–	–	–	–		*140*
	SP,[hh*ll*]	CF,78,*ll*,hh	SP(L)←[hh*ll*], SP(H)←[hh*ll*+1]	6	4	–	–	–	–	–	–	–	–		*145*
	SP,HL	CF,F1	SP←HL	2	2	–	–	–	–	–	–	–	–		*140*
	SP,IX	CF,F2	SP←IX	2	2	–	–	–	–	–	–	–	–		*140*
	SP,IY	CF,F3	SP←IY	2	2	–	–	–	–	–	–	–	–		*140*
	SP,#mmnn	CF,6E,nn,mm	SP←mmnn	4	4	–	–	–	–	–	–	–	–		*144*
LD	[hh*ll*],BA	BC,*ll*,hh	[hh*ll*]←A, [hh*ll*+1]←B	5	3	–	–	–	–	–	–	–	–		*140*
	[hh*ll*],HL	BD,*ll*,hh	[hh*ll*]←L, [hh*ll*+1]←H	5	3	–	–	–	–	–	–	–	–		*140*
	[hh*ll*],IX	BE,*ll*,hh	[hh*ll*]←IX(L), [hh*ll*+1]←IX(H)	5	3	–	–	–	–	–	–	–	–		*140*
	[hh*ll*],IY	BF,*ll*,hh	[hh*ll*]←IY(L), [hh*ll*+1]←IY(H)	5	3	–	–	–	–	–	–	–	–		*140*
	[hh*ll*],SP	CF,7C,*ll*,hh	[hh*ll*]←SP(L), [hh*ll*+1]←SP(H)	6	4	–	–	–	–	–	–	–	–		*141*
LD	[HL],BA	CF,C4	[HL]←A, [HL+1]←B	5	2	–	–	–	–	–	–	–	–		*141*
	[HL],HL	CF,C5	[HL]←L, [HL+1]←H	5	2	–	–	–	–	–	–	–	–		*141*
	[HL],IX	CF,C6	[HL]←IX(L), [HL+1]←IX(H)	5	2	–	–	–	–	–	–	–	–		*141*
	[HL],IY	CF,C7	[HL]←IY(L), [HL+1]←IY(H)	5	2	–	–	–	–	–	–	–	–		*141*
LD	[IX],BA	CF,D4	[IX]←A, [IX+1]←B	5	2	–	–	–	–	–	–	–	–		*142*
	[IX],HL	CF,D5	[IX]←L, [IX+1]←H	5	2	–	–	–	–	–	–	–	–		*142*
	[IX],IX	CF,D6	[IX]←IX(L), [IX+1]←IX(H)	5	2	–	–	–	–	–	–	–	–		*142*
	[IX],IY	CF,D7	[IX]←IY(L), [IX+1]←IY(H)	5	2	–	–	–	–	–	–	–	–		*142*
LD	[IY],BA	CF,DC	[IY]←A, [IY+1]←B	5	2	–	–	–	–	–	–	–	–		*142*
	[IY],HL	CF,DD	[IY]←L, [IY+1]←H	5	2	–	–	–	–	–	–	–	–		*142*
	[IY],IX	CF,DE	[IY]←IX(L), [IY+1]←IX(H)	5	2	–	–	–	–	–	–	–	–		*142*
	[IY],IY	CF,DF	[IY]←IY(L), [IY+1]←IY(H)	5	2	–	–	–	–	–	–	–	–		*142*
LD	[SP+dd],BA	CF,74,dd	[SP+dd]←A, [SP+dd+1]←B	6	3	–	–	–	–	–	–	–	–		*143*
	[SP+dd],HL	CF,75,dd	[SP+dd]←L, [SP+dd+1]←H	6	3	–	–	–	–	–	–	–	–		*143*
	[SP+dd],IX	CF,76,dd	[SP+dd]←IX(L), [SP+dd+1]←IX(H)	6	3	–	–	–	–	–	–	–	–		*143*
	[SP+dd],IY	CF,77,dd	[SP+dd]←IY(L), [SP+dd+1]←IY(H)	6	3	–	–	–	–	–	–	–	–		*143*
EX	BA,HL	C8	BA↔HL	3	1	–	–	–	–	–	–	–	–		*105*
	BA,IX	C9	BA↔IX	3	1	–	–	–	–	–	–	–	–		*105*
	BA,IY	CA	BA↔IY	3	1	–	–	–	–	–	–	–	–		*105*
	BA,SP	CB	BA↔SP	3	1	–	–	–	–	–	–	–	–		*105*

8-bit Arithmetic and Logic Operation Instructions (1/4)

Mnemonic		Machine Code	Operation	Cycle	Byte	SC								Comment	Page
						I1	I0	U	D	N	V	C	Z		
ADD	A,A	00	A←A+A	2	1	–	–	★	★	↕	↕	↕	↕		*67*
	A,B	01	A←A+B	2	1	–	–	★	★	↕	↕	↕	↕		*67*
	A,#nn	02,nn	A←A+nn	2	2	–	–	★	★	↕	↕	↕	↕		*67*
	A,[BR:*ll*]	04,*ll*	A←A+[BR:*ll*]	3	2	–	–	★	★	↕	↕	↕	↕		*67*
	A,[hh*ll*]	05,*ll*,hh	A←A+[hh*ll*]	4	3	–	–	★	★	↕	↕	↕	↕		*68*
	A,[HL]	03	A←A+[HL]	2	1	–	–	★	★	↕	↕	↕	↕		*68*
	A,[IX]	06	A←A+[IX]	2	1	–	–	★	★	↕	↕	↕	↕		*68*
	A,[IY]	07	A←A+[IY]	2	1	–	–	★	★	↕	↕	↕	↕		*68*
	A,[IX+dd]	CE,00,dd	A←A+[IX+dd]	4	3	–	–	★	★	↕	↕	↕	↕		*69*
	A,[IY+dd]	CE,01,dd	A←A+[IY+dd]	4	3	–	–	★	★	↕	↕	↕	↕		*69*
	A,[IX+L]	CE,02	A←A+[IX+L]	4	2	–	–	★	★	↕	↕	↕	↕		*69*
	A,[IY+L]	CE,03	A←A+[IY+L]	4	2	–	–	★	★	↕	↕	↕	↕		*69*
	[HL],A	CE,04	[HL]←[HL]+A	4	2	–	–	★	★	↕	↕	↕	↕		*70*
	[HL],#nn	CE,05,nn	[HL]←[HL]+nn	5	3	–	–	★	★	↕	↕	↕	↕		*70*
	[HL],[IX]	CE,06	[HL]←[HL]+[IX]	5	2	–	–	★	★	↕	↕	↕	↕		*71*
	[HL],[IY]	CE,07	[HL]←[HL]+[IY]	5	2	–	–	★	★	↕	↕	↕	↕		*71*
ADC	A,A	08	A←A+A+C	2	1	–	–	★	★	↕	↕	↕	↕		*60*
	A,B	09	A←A+B+C	2	1	–	–	★	★	↕	↕	↕	↕		*60*
	A,#nn	0A,nn	A←A+nn+C	2	2	–	–	★	★	↕	↕	↕	↕		*60*
	A,[BR:*ll*]	0C,*ll*	A←A+[BR:*ll*]+C	3	2	–	–	★	★	↕	↕	↕	↕		*60*
	A,[hh*ll*]	0D,*ll*,hh	A←A+[hh*ll*]+C	4	3	–	–	★	★	↕	↕	↕	↕		*61*
	A,[HL]	0B	A←A+[HL]+C	2	1	–	–	★	★	↕	↕	↕	↕		*61*
	A,[IX]	0E	A←A+[IX]+C	2	1	–	–	★	★	↕	↕	↕	↕		*62*
	A,[IY]	0F	A←A+[IY]+C	2	1	–	–	★	★	↕	↕	↕	↕		*62*
	A,[IX+dd]	CE,08,dd	A←A+[IX+dd]+C	4	3	–	–	★	★	↕	↕	↕	↕		*62*
	A,[IY+dd]	CE,09,dd	A←A+[IY+dd]+C	4	3	–	–	★	★	↕	↕	↕	↕		*62*
	A,[IX+L]	CE,0A	A←A+[IX+L]+C	4	2	–	–	★	★	↕	↕	↕	↕		*63*
	A,[IY+L]	CE,0B	A←A+[IY+L]+C	4	2	–	–	★	★	↕	↕	↕	↕		*63*
	[HL],A	CE,0C	[HL]←[HL]+A+C	4	2	–	–	★	★	↕	↕	↕	↕		*63*
	[HL],#nn	CE,0D,nn	[HL]←[HL]+nn+C	5	3	–	–	★	★	↕	↕	↕	↕		*64*
	[HL],[IX]	CE,0E	[HL]←[HL]+[IX]+C	5	2	–	–	★	★	↕	↕	↕	↕		*64*
	[HL],[IY]	CE,0F	[HL]←[HL]+[IY]+C	5	2	–	–	★	★	↕	↕	↕	↕		*64*
SUB	A,A	10	A←A-A	2	1	–	–	★	★	↕	↕	↕	↕		*180*
	A,B	11	A←A-B	2	1	–	–	★	★	↕	↕	↕	↕		*180*
	A,#nn	12,nn	A←A-nn	2	2	–	–	★	★	↕	↕	↕	↕		*180*
	A,[BR:*ll*]	14,*ll*	A←A-[BR:*ll*]	3	2	–	–	★	★	↕	↕	↕	↕		*180*
	A,[hh*ll*]	15,*ll*,hh	A←A-[hh*ll*]	4	3	–	–	★	★	↕	↕	↕	↕		*181*
	A,[HL]	13	A←A-[HL]	2	1	–	–	★	★	↕	↕	↕	↕		*181*
	A,[IX]	16	A←A-[IX]	2	1	–	–	★	★	↕	↕	↕	↕		*181*
	A,[IY]	17	A←A-[IY]	2	1	–	–	★	★	↕	↕	↕	↕		*181*
	A,[IX+dd]	CE,10,dd	A←A-[IX+dd]	4	3	–	–	★	★	↕	↕	↕	↕		*182*
	A,[IY+dd]	CE,11,dd	A←A-[IY+dd]	4	3	–	–	★	★	↕	↕	↕	↕		*182*
	A,[IX+L]	CE,12	A←A-[IX+L]	4	2	–	–	★	★	↕	↕	↕	↕		*182*
	A,[IY+L]	CE,13	A←A-[IY+L]	4	2	–	–	★	★	↕	↕	↕	↕		*182*
	[HL],A	CE,14	[HL]←[HL]-A	4	2	–	–	★	★	↕	↕	↕	↕		*183*
	[HL],#nn	CE,15,nn	[HL]←[HL]-nn	5	3	–	–	★	★	↕	↕	↕	↕		*183*
	[HL],[IX]	CE,16	[HL]←[HL]-[IX]	5	2	–	–	★	★	↕	↕	↕	↕		*184*
	[HL],[IY]	CE,17	[HL]←[HL]-[IY]	5	2	–	–	★	★	↕	↕	↕	↕		*184*

8-bit Arithmetic and Logic Operation Instructions (2/4)

Mnemonic		Machine Code	Operation	Cycle	Byte	SC								Comment	Page
						I1	I0	U	D	N	V	C	Z		
SBC	A,A	18	A←A-A-C	2	1	–	–	★	★	↕	↕	↕	↕		167
	A,B	19	A←A-B-C	2	1	–	–	★	★	↕	↕	↕	↕		167
	A,#nn	1A,nn	A←A-nn-C	2	2	–	–	★	★	↕	↕	↕	↕		167
	A,[BR:*ll*]	1C,*ll*	A←A-[BR:*ll*]-C	3	2	–	–	★	★	↕	↕	↕	↕		167
	A,[hh*ll*]	1D,*ll*,hh	A←A-[hh*ll*]-C	4	3	–	–	★	★	↕	↕	↕	↕		168
	A,[HL]	1B	A←A-[HL]-C	2	1	–	–	★	★	↕	↕	↕	↕		168
	A,[IX]	1E	A←A-[IX]-C	2	1	–	–	★	★	↕	↕	↕	↕		168
	A,[IY]	1F	A←A-[IY]-C	2	1	–	–	★	★	↕	↕	↕	↕		168
	A,[IX+dd]	CE,18,dd	A←A-[IX+dd]-C	4	3	–	–	★	★	↕	↕	↕	↕		169
	A,[IY+dd]	CE,19,dd	A←A-[IY+dd]-C	4	3	–	–	★	★	↕	↕	↕	↕		169
	A,[IX+L]	CE,1A	A←A-[IX+L]-C	4	2	–	–	★	★	↕	↕	↕	↕		169
	A,[IY+L]	CE,1B	A←A-[IY+L]-C	4	2	–	–	★	★	↕	↕	↕	↕		169
	[HL],A	CE,1C	[HL]←[HL]-A-C	4	2	–	–	★	★	↕	↕	↕	↕		170
	[HL],#nn	CE,1D,nn	[HL]←[HL]-nn-C	5	3	–	–	★	★	↕	↕	↕	↕		170
	[HL],[IX]	CE,1E	[HL]←[HL]-[IX]-C	5	2	–	–	★	★	↕	↕	↕	↕		171
	[HL],[IY]	CE,1F	[HL]←[HL]-[IY]-C	5	2	–	–	★	★	↕	↕	↕	↕		171
AND	A,A	20	A←A∧A	2	1	–	–	–	–	↕	–	–	↕		75
	A,B	21	A←A∧B	2	1	–	–	–	–	↕	–	–	↕		75
	A,#nn	22,nn	A←A∧nn	2	2	–	–	–	–	↕	–	–	↕		75
	A,[BR:*ll*]	24,*ll*	A←A∧[BR:*ll*]	3	2	–	–	–	–	↕	–	–	↕		75
	A,[hh*ll*]	25,*ll*,hh	A←A∧[hh*ll*]	4	3	–	–	–	–	↕	–	–	↕		76
	A,[HL]	23	A←A∧[HL]	2	1	–	–	–	–	↕	–	–	↕		76
	A,[IX]	26	A←A∧[IX]	2	1	–	–	–	–	↕	–	–	↕		76
	A,[IY]	27	A←A∧[IY]	2	1	–	–	–	–	↕	–	–	↕		76
	A,[IX+dd]	CE,20,dd	A←A∧[IX+dd]	4	3	–	–	–	–	↕	–	–	↕		77
	A,[IY+dd]	CE,21,dd	A←A∧[IY+dd]	4	3	–	–	–	–	↕	–	–	↕		77
	A,[IX+L]	CE,22	A←A∧[IX+L]	4	2	–	–	–	–	↕	–	–	↕		77
	A,[IY+L]	CE,23	A←A∧[IY+L]	4	2	–	–	–	–	↕	–	–	↕		77
	B,#nn	CE,B0,nn	B←B∧nn	3	3	–	–	–	–	↕	–	–	↕		78
	L,#nn	CE,B1,nn	L←L∧nn	3	3	–	–	–	–	↕	–	–	↕		78
	H,#nn	CE,B2,nn	H←H∧nn	3	3	–	–	–	–	↕	–	–	↕		78
	SC,#nn	9C,nn	SC←SC∧nn	3	2	↓	↓	↓	↓	↓	↓	↓	↓		79
	[BR:*ll*],#nn	D8,*ll*,nn	[BR:*ll*]←[BR:*ll*]∧nn	5	3	–	–	–	–	↕	–	–	↕		79
	[HL],A	CE,24	[HL]←[HL]∧A	4	2	–	–	–	–	↕	–	–	↕		79
	[HL],#nn	CE,25,nn	[HL]←[HL]∧nn	5	3	–	–	–	–	↕	–	–	↕		80
	[HL],[IX]	CE,26	[HL]←[HL]∧[IX]	5	2	–	–	–	–	↕	–	–	↕		80
	[HL],[IY]	CE,27	[HL]←[HL]∧[IY]	5	2	–	–	–	–	↕	–	–	↕		80
OR	A,A	28	A←A∨A	2	1	–	–	–	–	↕	–	–	↕		149
	A,B	29	A←A∨B	2	1	–	–	–	–	↕	–	–	↕		149
	A,#nn	2A,nn	A←A∨nn	2	2	–	–	–	–	↕	–	–	↕		149
	A,[BR:*ll*]	2C,*ll*	A←A∨[BR:*ll*]	3	2	–	–	–	–	↕	–	–	↕		150
	A,[hh*ll*]	2D,*ll*,hh	A←A∨[hh*ll*]	4	3	–	–	–	–	↕	–	–	↕		150
	A,[HL]	2B	A←A∨[HL]	2	1	–	–	–	–	↕	–	–	↕		150
	A,[IX]	2E	A←A∨[IX]	2	1	–	–	–	–	↕	–	–	↕		151
	A,[IY]	2F	A←A∨[IY]	2	1	–	–	–	–	↕	–	–	↕		151
	A,[IX+dd]	CE,28,dd	A←A∨[IX+dd]	4	3	–	–	–	–	↕	–	–	↕		151
	A,[IY+dd]	CE,29,dd	A←A∨[IY+dd]	4	3	–	–	–	–	↕	–	–	↕		151
	A,[IX+L]	CE,2A	A←A∨[IX+L]	4	2	–	–	–	–	↕	–	–	↕		152

8-bit Arithmetic and Logic Operation Instructions (3/4)

Mnemonic		Machine Code	Operation	Cycle	Byte	SC								Comment	Page
						I1	I0	U	D	N	V	C	Z		
OR	A,[IY+L]	CE,2B	A←A∨[IY+L]	4	2	–	–	–	–	↕	–	–	↕		*152*
	B,#nn	CE,B4,nn	B←B∨nn	3	3	–	–	–	–	↕	–	–	↕		*152*
	L,#nn	CE,B5,nn	L←L∨nn	3	3	–	–	–	–	↕	–	–	↕		*152*
	H,#nn	CE,B6,nn	H←H∨nn	3	3	–	–	–	–	↕	–	–	↕		*153*
	SC,#nn	9D,nn	SC←SC∨nn	3	2	↑	↑	↑	↑	↑	↑	↑	↑		*153*
	[BR:*ll*],#nn	D9,*ll*,nn	[BR:*ll*]←[BR:*ll*]∨nn	5	3	–	–	–	–	↕	–	–	↕		*153*
	[HL],A	CE,2C	[HL]←[HL]∨A	4	2	–	–	–	–	↕	–	–	↕		*154*
	[HL],#nn	CE,2D,nn	[HL]←[HL]∨nn	5	3	–	–	–	–	↕	–	–	↕		*154*
	[HL],[IX]	CE,2E	[HL]←[HL]∨[IX]	5	2	–	–	–	–	↕	–	–	↕		*154*
	[HL],[IY]	CE,2F	[HL]←[HL]∨[IY]	5	2	–	–	–	–	↕	–	–	↕		*154*
XOR	A,A	38	A←A∀A	2	1	–	–	–	–	↕	–	–	↕		*189*
	A,B	39	A←A∀B	2	1	–	–	–	–	↕	–	–	↕		*189*
	A,#nn	3A,nn	A←A∀nn	2	2	–	–	–	–	↕	–	–	↕		*189*
	A,[BR:*ll*]	3C,*ll*	A←A∀[BR:*ll*]	3	2	–	–	–	–	↕	–	–	↕		*189*
	A,[hh*ll*]	3D,*ll*,hh	A←A∀[hh*ll*]	4	3	–	–	–	–	↕	–	–	↕		*190*
	A,[HL]	3B	A←A∀[HL]	2	1	–	–	–	–	↕	–	–	↕		*190*
	A,[IX]	3E	A←A∀[IX]	2	1	–	–	–	–	↕	–	–	↕		*190*
	A,[IY]	3F	A←A∀[IY]	2	1	–	–	–	–	↕	–	–	↕		*190*
	A,[IX+dd]	CE,38,dd	A←A∀[IX+dd]	4	3	–	–	–	–	↕	–	–	↕		*191*
	A,[IY+dd]	CE,39,dd	A←A∀[IY+dd]	4	3	–	–	–	–	↕	–	–	↕		*191*
	A,[IX+L]	CE,3A	A←A∀[IX+L]	4	2	–	–	–	–	↕	–	–	↕		*191*
	A,[IY+L]	CE,3B	A←A∀[IY+L]	4	2	–	–	–	–	↕	–	–	↕		*191*
	B,#nn	CE,B8,nn	B←B∀nn	3	3	–	–	–	–	↕	–	–	↕		*192*
	L,#nn	CE,B9,nn	L←L∀nn	3	3	–	–	–	–	↕	–	–	↕		*192*
	H,#nn	CE,BA,nn	H←H∀nn	3	3	–	–	–	–	↕	–	–	↕		*192*
	SC,#nn	9E,nn	SC←SC∀nn	3	2	↕	↕	↕	↕	↕	↕	↕	↕		*193*
	[BR:*ll*],#nn	DA,*ll*,nn	[BR:*ll*]←[BR:*ll*]∀nn	5	3	–	–	–	–	↕	–	–	↕		*193*
	[HL],A	CE,3C	[HL]←[HL]∀A	4	2	–	–	–	–	↕	–	–	↕		*193*
	[HL],#nn	CE,3D,nn	[HL]←[HL]∀nn	5	3	–	–	–	–	↕	–	–	↕		*194*
	[HL],[IX]	CE,3E	[HL]←[HL]∀[IX]	5	2	–	–	–	–	↕	–	–	↕		*194*
	[HL],[IY]	CE,3F	[HL]←[HL]∀[IY]	5	2	–	–	–	–	↕	–	–	↕		*194*
CP	A,A	30	A-A	2	1	–	–	–	–	↕	↕	↕	↕		*90*
	A,B	31	A-B	2	1	–	–	–	–	↕	↕	↕	↕		*90*
	A,#nn	32,nn	A-nn	2	2	–	–	–	–	↕	↕	↕	↕		*90*
	A,[BR:*ll*]	34,*ll*	A-[BR:*ll*]	3	2	–	–	–	–	↕	↕	↕	↕		*90*
	A,[hh*ll*]	35,*ll*,hh	A-[hh*ll*]	4	3	–	–	–	–	↕	↕	↕	↕		*91*
	A,[HL]	33	A-[HL]	2	1	–	–	–	–	↕	↕	↕	↕		*91*
	A,[IX]	36	A-[IX]	2	1	–	–	–	–	↕	↕	↕	↕		*92*
	A,[IY]	37	A-[IY]	2	1	–	–	–	–	↕	↕	↕	↕		*92*
	A,[IX+dd]	CE,30,dd	A-[IX+dd]	4	3	–	–	–	–	↕	↕	↕	↕		*92*
	A,[IY+dd]	CE,31,dd	A-[IY+dd]	4	3	–	–	–	–	↕	↕	↕	↕		*92*
	A,[IX+L]	CE,32	A-[IX+L]	4	2	–	–	–	–	↕	↕	↕	↕		*93*
	A,[IY+L]	CE,33	A-[IY+L]	4	2	–	–	–	–	↕	↕	↕	↕		*93*
	B,#nn	CE,BC,nn	B-nn	3	3	–	–	–	–	↕	↕	↕	↕		*93*
	L,#nn	CE,BD,nn	L-nn	3	3	–	–	–	–	↕	↕	↕	↕		*94*
	H,#nn	CE,BE,nn	H-nn	3	3	–	–	–	–	↕	↕	↕	↕		*94*
	BR,#hh	CE,BF,hh	BR-hh	3	3	–	–	–	–	↕	↕	↕	↕		*94*
	[BR:*ll*],#nn	DB,*ll*,nn	[BR:*ll*]-nn	4	3	–	–	–	–	↕	↕	↕	↕		*95*

8-bit Arithmetic and Logic Operation Instructions (4/4)

Mnemonic		Machine Code	Operation	Cycle	Byte	SC								Comment	Page
						I1	I0	U	D	N	V	C	Z		
CP	[HL],A	CE,34	[HL]-A	3	2	–	–	–	–	↕	↕	↕	↕		*95*
	[HL],#nn	CE,35,nn	[HL]-nn	4	3	–	–	–	–	↕	↕	↕	↕		*96*
	[HL],[IX]	CE,36	[HL]-[IX]	4	2	–	–	–	–	↕	↕	↕	↕		*96*
	[HL],[IY]	CE,37	[HL]-[IY]	4	2	–	–	–	–	↕	↕	↕	↕		*96*
BIT	A,B	94	A∧B	2	1	–	–	–	–	↕	–	–	↕		*81*
	A,#nn	96,nn	A∧nn	2	2	–	–	–	–	↕	–	–	↕		*81*
	B,#nn	97,nn	B∧nn	2	2	–	–	–	–	↕	–	–	↕		*81*
	[BR:*ll*],#nn	DC,*ll*,nn	[BR:*ll*]∧nn	4	3	–	–	–	–	↕	–	–	↕		*82*
	[HL],#nn	95,nn	[HL]∧nn	3	2	–	–	–	–	↕	–	–	↕		*82*
INC	A	80	A←A+1	2	1	–	–	–	–	–	–	–	↕		*106*
	B	81	B←B+1	2	1	–	–	–	–	–	–	–	↕		*106*
	L	82	L←L+1	2	1	–	–	–	–	–	–	–	↕		*106*
	H	83	H←H+1	2	1	–	–	–	–	–	–	–	↕		*106*
	BR	84	BR←BR+1	2	1	–	–	–	–	–	–	–	↕		*106*
	[BR:*ll*]	85,*ll*	[BR:*ll*]←[BR:*ll*]+1	4	2	–	–	–	–	–	–	–	↕		*107*
	[HL]	86	[HL]←[HL]+1	3	1	–	–	–	–	–	–	–	↕		*107*
DEC	A	88	A←A-1	2	1	–	–	–	–	–	–	–	↕		*102*
	B	89	B←B-1	2	1	–	–	–	–	–	–	–	↕		*102*
	L	8A	L←L-1	2	1	–	–	–	–	–	–	–	↕		*102*
	H	8B	H←H-1	2	1	–	–	–	–	–	–	–	↕		*102*
	BR	8C	BR←BR-1	2	1	–	–	–	–	–	–	–	↕		*102*
	[BR:*ll*]	8D,*ll*	[BR:*ll*]←[BR:*ll*]-1	4	2	–	–	–	–	–	–	–	↕		*102*
	[HL]	8E	[HL]←[HL]-1	3	1	–	–	–	–	–	–	–	↕		*103*
CPL	A	CE,A0	A←$\overline{A}$	3	2	–	–	–	–	↕	–	–	↕		*101*
	B	CE,A1	B←$\overline{B}$	3	2	–	–	–	–	↕	–	–	↕		*101*
	[BR:*ll*]	CE,A2,*ll*	[BR:*ll*]←$\overline{[BR:ll]}$	5	3	–	–	–	–	↕	–	–	↕		*101*
	[HL]	CE,A3	[HL]←$\overline{[HL]}$	4	2	–	–	–	–	↕	–	–	↕		*101*
NEG	A	CE,A4	A←0-A	3	2	–	–	★	★	↕	↕	↕	↕		*148*
	B	CE,A5	B←0-B	3	2	–	–	★	★	↕	↕	↕	↕		*148*
	[BR:*ll*]	CE,A6,*ll*	[BR:*ll*]←0-[BR:*ll*]	5	3	–	–	★	★	↕	↕	↕	↕		*148*
	[HL]	CE,A7	[HL]←0-[HL]	4	2	–	–	★	★	↕	↕	↕	↕		*148*
MLT		CE,D8	HL←L*A	12	2	–	–	–	–	↕	0	0	↕	*MODEL1/3*	*147*
DIV		CE,D9	L←HL/A, H←Remainder	13	2	–	–	–	–	↕	↕	0	↕	*only*	*104*

16-bit Arithmetic Operation Instructions (1/2)

Mnemonic		Machine Code	Operation	Cycle	Byte	SC								Comment	Page
						I1	I0	U	D	N	V	C	Z		
ADD	BA,BA	CF,00	BA←BA+BA	4	2	–	–	–	–	↕	↕	↕	↕		*71*
	BA,HL	CF,01	BA←BA+HL	4	2	–	–	–	–	↕	↕	↕	↕		*71*
	BA,IX	CF,02	BA←BA+IX	4	2	–	–	–	–	↕	↕	↕	↕		*71*
	BA,IY	CF,03	BA←BA+IY	4	2	–	–	–	–	↕	↕	↕	↕		*71*
	BA,#mmnn	C0,nn,mm	BA←BA+mmnn	3	3	–	–	–	–	↕	↕	↕	↕		*72*
	HL,BA	CF,20	HL←HL+BA	4	2	–	–	–	–	↕	↕	↕	↕		*72*
	HL,HL	CF,21	HL←HL+HL	4	2	–	–	–	–	↕	↕	↕	↕		*72*
	HL,IX	CF,22	HL←HL+IX	4	2	–	–	–	–	↕	↕	↕	↕		*72*
	HL,IY	CF,23	HL←HL+IY	4	2	–	–	–	–	↕	↕	↕	↕		*72*
	HL,#mmnn	C1,nn,mm	HL←HL+mmnn	3	3	–	–	–	–	↕	↕	↕	↕		*72*
	IX,BA	CF,40	IX←IX+BA	4	2	–	–	–	–	↕	↕	↕	↕		*73*
	IX,HL	CF,41	IX←IX+HL	4	2	–	–	–	–	↕	↕	↕	↕		*73*
	IX,#mmnn	C2,nn,mm	IX←IX+mmnn	3	3	–	–	–	–	↕	↕	↕	↕		*73*
	IY,BA	CF,42	IY←IY+BA	4	2	–	–	–	–	↕	↕	↕	↕		*73*
	IY,HL	CF,43	IY←IY+HL	4	2	–	–	–	–	↕	↕	↕	↕		*73*
	IY,#mmnn	C3,nn,mm	IY←IY+mmnn	3	3	–	–	–	–	↕	↕	↕	↕		*74*
	SP,BA	CF,44	SP←SP+BA	4	2	–	–	–	–	↕	↕	↕	↕		*74*
	SP,HL	CF,45	SP←SP+HL	4	2	–	–	–	–	↕	↕	↕	↕		*74*
	SP,#mmnn	CF,68,nn,mm	SP←SP+mmnn	4	4	–	–	–	–	↕	↕	↕	↕		*74*
ADC	BA,BA	CF,04	BA←BA+BA+C	4	2	–	–	–	–	↕	↕	↕	↕		*65*
	BA,HL	CF,05	BA←BA+HL+C	4	2	–	–	–	–	↕	↕	↕	↕		*65*
	BA,IX	CF,06	BA←BA+IX+C	4	2	–	–	–	–	↕	↕	↕	↕		*65*
	BA,IY	CF,07	BA←BA+IY+C	4	2	–	–	–	–	↕	↕	↕	↕		*65*
	BA,#mmnn	CF,60,nn,mm	BA←BA+mmnn+C	4	4	–	–	–	–	↕	↕	↕	↕		*65*
	HL,BA	CF,24	HL←HL+BA+C	4	2	–	–	–	–	↕	↕	↕	↕		*66*
	HL,HL	CF,25	HL←HL+HL+C	4	2	–	–	–	–	↕	↕	↕	↕		*66*
	HL,IX	CF,26	HL←HL+IX+C	4	2	–	–	–	–	↕	↕	↕	↕		*66*
	HL,IY	CF,27	HL←HL+IY+C	4	2	–	–	–	–	↕	↕	↕	↕		*66*
	HL,#mmnn	CF,61,nn,mm	HL←HL+mmnn+C	4	4	–	–	–	–	↕	↕	↕	↕		*66*
SUB	BA,BA	CF,08	BA←BA-BA	4	2	–	–	–	–	↕	↕	↕	↕		*184*
	BA,HL	CF,09	BA←BA-HL	4	2	–	–	–	–	↕	↕	↕	↕		*184*
	BA,IX	CF,0A	BA←BA-IX	4	2	–	–	–	–	↕	↕	↕	↕		*184*
	BA,IY	CF,0B	BA←BA-IY	4	2	–	–	–	–	↕	↕	↕	↕		*184*
	BA,#mmnn	D0,nn,mm	BA←BA-mmnn	3	3	–	–	–	–	↕	↕	↕	↕		*185*
	HL,BA	CF,28	HL←HL-BA	4	2	–	–	–	–	↕	↕	↕	↕		*185*
	HL,HL	CF,29	HL←HL-HL	4	2	–	–	–	–	↕	↕	↕	↕		*185*
	HL,IX	CF,2A	HL←HL-IX	4	2	–	–	–	–	↕	↕	↕	↕		*185*
	HL,IY	CF,2B	HL←HL-IY	4	2	–	–	–	–	↕	↕	↕	↕		*185*
	HL,#mmnn	D1,nn,mm	HL←HL-mmnn	3	3	–	–	–	–	↕	↕	↕	↕		*185*
	IX,BA	CF,48	IX←IX-BA	4	2	–	–	–	–	↕	↕	↕	↕		*186*
	IX,HL	CF,49	IX←IX-HL	4	2	–	–	–	–	↕	↕	↕	↕		*186*
	IX,#mmnn	D2,nn,mm	IX←IX-mmnn	3	3	–	–	–	–	↕	↕	↕	↕		*186*
	IY,BA	CF,4A	IY←IY-BA	4	2	–	–	–	–	↕	↕	↕	↕		*186*
	IY,HL	CF,4B	IY←IY-HL	4	2	–	–	–	–	↕	↕	↕	↕		*186*
	IY,#mmnn	D3,nn,mm	IY←IY-mmnn	3	3	–	–	–	–	↕	↕	↕	↕		*187*
	SP,BA	CF,4C	SP←SP-BA	4	2	–	–	–	–	↕	↕	↕	↕		*187*
	SP,HL	CF,4D	SP←SP-HL	4	2	–	–	–	–	↕	↕	↕	↕		*187*
	SP,#mmnn	CF,6A,nn,mm	SP←SP-mmnn	4	4	–	–	–	–	↕	↕	↕	↕		*187*

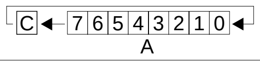
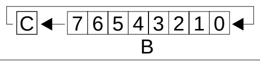
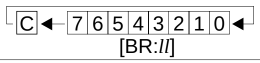
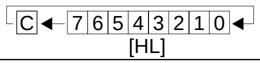
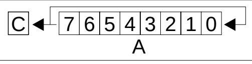
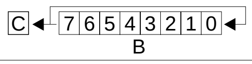
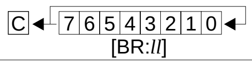
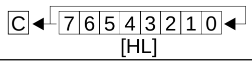
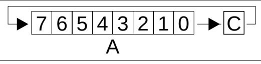
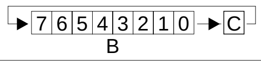
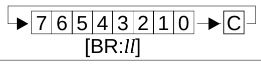
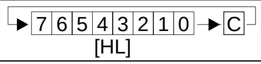
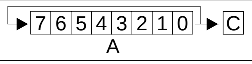
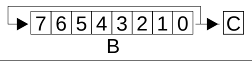
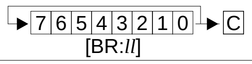
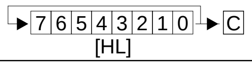
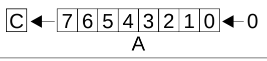
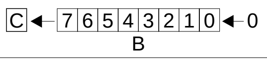
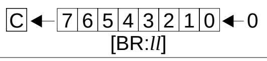
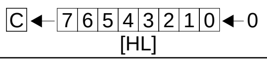
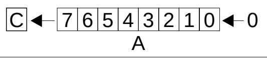
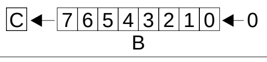
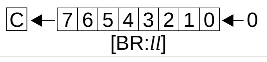
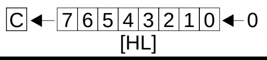
16-bit Arithmetic Operation Instructions (2/2)

Mnemonic		Machine Code	Operation	Cycle	Byte	SC								Comment	Page
						I1	I0	U	D	N	V	C	Z		
SBC	BA,BA	CF,0C	BA←BA-BA-C	4	2	–	–	–	–	↕	↕	↕	↕		*171*
	BA,HL	CF,0D	BA←BA-HL-C	4	2	–	–	–	–	↕	↕	↕	↕		*171*
	BA,IX	CF,0E	BA←BA-IX-C	4	2	–	–	–	–	↕	↕	↕	↕		*171*
	BA,IY	CF,0F	BA←BA-IY-C	4	2	–	–	–	–	↕	↕	↕	↕		*171*
	BA,#mmnn	CF,62,nn,mm	BA←BA-mmnn-C	4	4	–	–	–	–	↕	↕	↕	↕		*172*
	HL,BA	CF,2C	HL←HL-BA-C	4	2	–	–	–	–	↕	↕	↕	↕		*172*
	HL,HL	CF,2D	HL←HL-HL-C	4	2	–	–	–	–	↕	↕	↕	↕		*172*
	HL,IX	CF,2E	HL←HL-IX-C	4	2	–	–	–	–	↕	↕	↕	↕		*172*
	HL,IY	CF,2F	HL←HL-IY-C	4	2	–	–	–	–	↕	↕	↕	↕		*172*
	HL,#mmnn	CF,63,nn,mm	HL←HL-mmnn-C	4	4	–	–	–	–	↕	↕	↕	↕		*172*
CP	BA,BA	CF,18	BA-BA	4	2	–	–	–	–	↕	↕	↕	↕		*97*
	BA,HL	CF,19	BA-HL	4	2	–	–	–	–	↕	↕	↕	↕		*97*
	BA,IX	CF,1A	BA-IX	4	2	–	–	–	–	↕	↕	↕	↕		*97*
	BA,IY	CF,1B	BA-IY	4	2	–	–	–	–	↕	↕	↕	↕		*97*
	BA,#mmnn	D4,nn,mm	BA-mmnn	3	3	–	–	–	–	↕	↕	↕	↕		*97*
	HL,BA	CF,38	HL-BA	4	2	–	–	–	–	↕	↕	↕	↕		*98*
	HL,HL	CF,39	HL-HL	4	2	–	–	–	–	↕	↕	↕	↕		*98*
	HL,IX	CF,3A	HL-IX	4	2	–	–	–	–	↕	↕	↕	↕		*98*
	HL,IY	CF,3B	HL-IY	4	2	–	–	–	–	↕	↕	↕	↕		*98*
	HL,#mmnn	D5,nn,mm	HL-mmnn	3	3	–	–	–	–	↕	↕	↕	↕		*98*
	IX,#mmnn	D6,nn,mm	IX-mmnn	3	3	–	–	–	–	↕	↕	↕	↕		*99*
	IY,#mmnn	D7,nn,mm	IY-mmnn	3	3	–	–	–	–	↕	↕	↕	↕		*99*
	SP,BA	CF,5C	SP-BA	4	2	–	–	–	–	↕	↕	↕	↕		*100*
	SP,HL	CF,5D	SP-HL	4	2	–	–	–	–	↕	↕	↕	↕		*100*
	SP,#mmnn	CF,6C,nn,mm	SP-mmnn	4	4	–	–	–	–	↕	↕	↕	↕		*100*
INC	BA	90	BA←BA+1	2	1	–	–	–	–	–	–	–	↕		*107*
	HL	91	HL←HL+1	2	1	–	–	–	–	–	–	–	↕		*107*
	IX	92	IX←IX+1	2	1	–	–	–	–	–	–	–	↕		*107*
	IY	93	IY←IY+1	2	1	–	–	–	–	–	–	–	↕		*107*
	SP	87	SP←SP+1	2	1	–	–	–	–	–	–	–	↕		*108*
DEC	BA	98	BA←BA-1	2	1	–	–	–	–	–	–	–	↕		*103*
	HL	99	HL←HL-1	2	1	–	–	–	–	–	–	–	↕		*103*
	IX	9A	IX←IX-1	2	1	–	–	–	–	–	–	–	↕		*103*
	IY	9B	IY←IY-1	2	1	–	–	–	–	–	–	–	↕		*103*
	SP	8F	SP←SP-1	2	1	–	–	–	–	–	–	–	↕		*103*

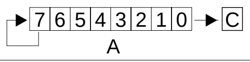
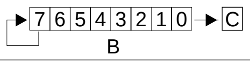
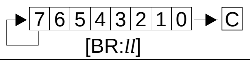
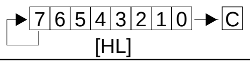
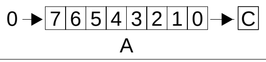
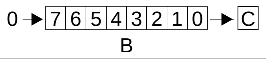
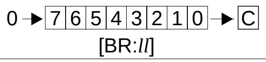
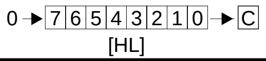
Auxiliary Operation Instructions

Mnemonic		Machine Code	Operation	Cycle	Byte	SC								Comment	Page
						I1	I0	U	D	N	V	C	Z		
PACK		DE	B A → A $\boxed{*}\boxed{m}\boxed{*}\boxed{n} \longrightarrow \boxed{m}\boxed{n}$	2	1	–	–	–	–	–	–	–	–		*155*
UPCK		DF	A → B A $\boxed{m}\boxed{n} \longrightarrow \boxed{0}\boxed{m}\boxed{0}\boxed{n}$	2	1	–	–	–	–	–	–	–	–		*188*
SEP		CE,A8	B A → B A $\boxed{}\boxed{0*******} \rightarrow \boxed{00000000}\boxed{0*******}$ $\boxed{}\boxed{1*******} \rightarrow \boxed{11111111}\boxed{1*******}$	3	2	–	–	–	–	–	–	–	–		*173*

Rotate/Shift Instructions (1/2)

Mnemonic		Machine Code	Operation	Cycle	Byte	SC								Comment	Page
						I1	I0	U	D	N	V	C	Z		
RL	A	CE,90		3	2	–	–	–	–	↕	–	↕	↕		*162*
	B	CE,91		3	2	–	–	–	–	↕	–	↕	↕		*162*
	[BR:*ll*]	CE,92,*ll*		5	3	–	–	–	–	↕	–	↕	↕		*163*
	[HL]	CE,93		4	2	–	–	–	–	↕	–	↕	↕		*163*
RLC	A	CE,94		3	2	–	–	–	–	↕	–	↕	↕		*163*
	B	CE,95		3	2	–	–	–	–	↕	–	↕	↕		*163*
	[BR:*ll*]	CE,96,*ll*		5	3	–	–	–	–	↕	–	↕	↕		*164*
	[HL]	CE,97		4	2	–	–	–	–	↕	–	↕	↕		*164*
RR	A	CE,98		3	2	–	–	–	–	↕	–	↕	↕		*164*
	B	CE,99		3	2	–	–	–	–	↕	–	↕	↕		*164*
	[BR:*ll*]	CE,9A,*ll*		5	3	–	–	–	–	↕	–	↕	↕		*165*
	[HL]	CE,9B		4	2	–	–	–	–	↕	–	↕	↕		*165*
RRC	A	CE,9C		3	2	–	–	–	–	↕	–	↕	↕		*166*
	B	CE,9D		3	2	–	–	–	–	↕	–	↕	↕		*166*
	[BR:*ll*]	CE,9E,*ll*		5	3	–	–	–	–	↕	–	↕	↕		*166*
	[HL]	CE,9F		4	2	–	–	–	–	↕	–	↕	↕		*166*
SLA	A	CE,80		3	2	–	–	–	–	↕	↕	↕	↕		*173*
	B	CE,81		3	2	–	–	–	–	↕	↕	↕	↕		*173*
	[BR:*ll*]	CE,82,*ll*		5	3	–	–	–	–	↕	↕	↕	↕		*174*
	[HL]	CE,83		4	2	–	–	–	–	↕	↕	↕	↕		*174*
SLL	A	CE,84		3	2	–	–	–	–	↕	–	↕	↕		*175*
	B	CE,85		3	2	–	–	–	–	↕	–	↕	↕		*175*
	[BR:*ll*]	CE,86,*ll*		5	3	–	–	–	–	↕	–	↕	↕		*175*
	[HL]	CE,87		4	2	–	–	–	–	↕	–	↕	↕		*176*

Rotate/Shift Instructions (2/2)

Mnemonic		Machine Code	Operation	Cycle	Byte	SC								Comment	Page
						I1	I0	U	D	N	V	C	Z		
SRA	A	CE,88		3	2	–	–	–	–	↕	0	↕	↕		*177*
	B	CE,89		3	2	–	–	–	–	↕	0	↕	↕		*177*
	[BR:*ll*]	CE,8A,*ll*		5	3	–	–	–	–	↕	0	↕	↕		*177*
	[HL]	CE,8B		4	2	–	–	–	–	↕	0	↕	↕		*178*
SRL	A	CE,8C		3	2	–	–	–	–	0	–	↕	↕		*178*
	B	CE,8D		3	2	–	–	–	–	0	–	↕	↕		*178*
	[BR:*ll*]	CE,8E,*ll*		5	3	–	–	–	–	0	–	↕	↕		*179*
	[HL]	CE,8F		4	2	–	–	–	–	0	–	↕	↕		*179*

Stack Control Instructions

Mnemonic		Machine Code	Operation	Cycle	Byte	SC								Comment	Page
						I1	I0	U	D	N	V	C	Z		
PUSH	A	CF,B0	[SP-1]←A, SP←SP-1	3	2	–	–	–	–	–	–	–	–		*158*
	B	CF,B1	[SP-1]←B, SP←SP-1	3	2	–	–	–	–	–	–	–	–		*158*
	L	CF,B2	[SP-1]←L, SP←SP-1	3	2	–	–	–	–	–	–	–	–		*158*
	H	CF,B3	[SP-1]←H, SP←SP-1	3	2	–	–	–	–	–	–	–	–		*158*
	BR	A4	[SP-1]←BR, SP←SP-1	3	1	–	–	–	–	–	–	–	–		*159*
	SC	A7	[SP-1]←SC, SP←SP-1	3	1	–	–	–	–	–	–	–	–		*160*
	BA	A0	[SP-1]←B, [SP-2]←A, SP←SP-2	4	1	–	–	–	–	–	–	–	–		*158*
	HL	A1	[SP-1]←H, [SP-2]←L, SP←SP-2	4	1	–	–	–	–	–	–	–	–		*158*
	IX	A2	[SP-1]←IX(H), [SP-2]←IX(L), SP←SP-2	4	1	–	–	–	–	–	–	–	–		*158*
	IY	A3	[SP-1]←IY(H), [SP-2]←IY(L), SP←SP-2	4	1	–	–	–	–	–	–	–	–		*158*
	EP	A5	[SP-1]←EP, SP←SP-1	3	1	–	–	–	–	–	–	–	–		*159*
	IP	A6	[SP-1]←XP, [SP-2]←YP, SP←SP-2	4	1	–	–	–	–	–	–	–	–		*159*
PUSH	ALL	CF,B8	PUSH BA, HL, IX, IY, BR	12	2	–	–	–	–	–	–	–	–		*160*
	ALE	CF,B9	PUSH BA, HL, IX, IY, BR, EP, IP	15	2	–	–	–	–	–	–	–	–	*MODEL2/3 only*	*160*
POP	A	CF,B4	A←[SP], SP←SP+1	3	2	–	–	–	–	–	–	–	–		*155*
	B	CF,B5	B←[SP], SP←SP+1	3	2	–	–	–	–	–	–	–	–		*155*
	L	CF,B6	L←[SP], SP←SP+1	3	2	–	–	–	–	–	–	–	–		*155*
	H	CF,B7	H←[SP], SP←SP+1	3	2	–	–	–	–	–	–	–	–		*155*
	BR	AC	BR←[SP], SP←SP+1	2	1	–	–	–	–	–	–	–	–		*156*
	SC	AF	SC←[SP], SP←SP+1	2	1	↕	↕	↕	↕	↕	↕	↕	↕		*156*
	BA	A8	A←[SP], B←[SP+1], SP←SP+2	3	1	–	–	–	–	–	–	–	–		*155*
	HL	A9	L←[SP], H←[SP+1], SP←SP+2	3	1	–	–	–	–	–	–	–	–		*155*
	IX	AA	IX(L)←[SP], IX(H)←[SP+1], SP←SP+2	3	1	–	–	–	–	–	–	–	–		*155*
	IY	AB	IY(L)←[SP], IY(H)←[SP+1], SP←SP+2	3	1	–	–	–	–	–	–	–	–		*155*
	EP	AD	EP←[SP], SP←SP+1	2	1	–	–	–	–	–	–	–	–		*156*
	IP	AE	YP←[SP], XP←[SP+1], SP←SP+2	3	1	–	–	–	–	–	–	–	–		*156*
POP	ALL	CF,BC	POP BR, IY, IX, HL, BA	11	2	–	–	–	–	–	–	–	–		*157*
	ALE	CF,BD	POP IP, EP, BR, IY, IX, HL, BA	14	2	–	–	–	–	–	–	–	–	*MODEL2/3 only*	*157*

* New code bank register NB and expand page registers EP/XP/YP are set only for MODEL2/3.
In MODEL0/1, instructions that access these registers cannot be used.

Branch Instructions (1/4)

Mnemonic		Machine Code	Condition	Operation	Cycle	Byte	SC								Page
							I1	I0	U	D	N	V	C	Z	
JRS	rr	F1,rr	Unconditionable	*MODEL0/1* PC←PC+rr+1 *MODEL2/3* PC←PC+rr+1, CB←NB	2	2	–	–	–	–	–	–	–	–	*112*
JRS	C,rr	E4,rr	C=1	*MODEL0/1* If Condition is true, then PC←PC+rr+1 else PC←PC+2 *MODEL2/3* If Condition is true, then PC←PC+rr+1, CB←NB else PC←PC+2, NB←CB	2	2	–	–	–	–	–	–	–	–	*113*
	NC,rr	E5,rr	C=0												
	Z,rr	E6,rr	Z=1												
	NZ,rr	E7,rr	Z=0												
JRS	LT,rr	CE,E0,rr	[N∀V]=1	*MODEL0/1* If Condition is true, then PC←PC+rr+2 else PC←PC+3 *MODEL2/3* If Condition is true, then PC←PC+rr+2, CB←NB else PC←PC+3, NB←CB	3	3	–	–	–	–	–	–	–	–	*113*
	LE,rr	CE,E1,rr	Z∨[N∀V]=1												
	GT,rr	CE,E2,rr	Z∨[N∀V]=0												
	GE,rr	CE,E3,rr	[N∀V]=0												
	V,rr	CE,E4,rr	V=1												
	NV,rr	CE,E5,rr	V=0												
	P,rr	CE,E6,rr	N=0												
	M,rr	CE,E7,rr	N=1												
	F0,rr	CE,E8,rr	F0=1												
	F1,rr	CE,E9,rr	F1=1												
	F2,rr	CE,EA,rr	F2=1												
	F3,rr	CE,EB,rr	F3=1												
	NF0,rr	CE,EC,rr	F0=0												
	NF1,rr	CE,ED,rr	F1=0												
	NF2,rr	CE,EE,rr	F2=0												
	NF3,rr	CE,EF,rr	F3=0												
JRL	qqrr	F3,rr,qq	Unconditionable	*MODEL0/1* PC←PC+qqrr+2 *MODEL2/3* PC←PC+qqrr+2, CB←NB	3	3	–	–	–	–	–	–	–	–	*110*
JRL	C,qqrr	EC,rr,qq	C=1	*MODEL0/1* If Condition is true, then PC←PC+qqrr+2 else PC←PC+3 *MODEL2/3* If Condition is true, then PC←PC+qqrr+2, CB←NB else PC←PC+3, NB←CB	3	3	–	–	–	–	–	–	–	–	*111*
	NC,qqrr	ED,rr,qq	C=0												
	Z,qqrr	EE,rr,qq	Z=1												
	NZ,qqrr	EF,rr,qq	Z=0												
DJR	NZ,rr	F5,rr	B=0	*MODEL0/1* B←B-1, If B=0, then PC←PC+rr+1 else PC←PC+2 *MODEL2/3* B←B-1, If B=0, then PC←PC+rr+1, CB←NB else PC←PC+2, NB←CB	4	2	–	–	–	–	–	–	–	↕	*104*

Branch Instructions (2/4)

Mnemonic		Machine Code	Condition	Operation	Cycle	Byte	SC I1 I0 U D N V C Z	Page
JP	HL	F4	Unconditionable	*MODEL0/1* PC←HL	2	1	– – – – – – – –	*109*
				MODEL2/3 PC←HL, CB←NB				
	[kk]	FD,kk	Unconditionable	*MODEL0/1* PC(L)←[00kk], PC(H)←[00kk+1]	4	2	– – – – – – – –	*109*
				MODEL2/3 PC(L)←[00kk] PC(H)←[00kk+1], CB←NB				
CARS	rr	F0,rr	Unconditionable	*MODEL0/1* [SP-1]←PC(H), [SP-2]←PC(L), SP←SP-2, PC←PC+rr+1	4	2	– – – – – – – –	*86*
				MODEL2/3 (Minimum mode) [SP-1]←PC(H), [SP-2]←PC(L), SP←SP-2, PC←PC+rr+1, CB←NB				
				MODEL2/3 (Maximum mode) [SP-1]←CB, [SP-2]←PC(H), [SP-3]←PC(L), SP←SP-3, PC←PC+rr+1, CB←NB	5			
CARS	C,rr	E0,rr	C=1	*MODEL0/1* If Condition is true then [SP-1]←PC(H), [SP-2]←PC(L), SP←SP-2, PC←PC+rr+1	4	2	– – – – – – – –	*87*
				else PC←PC+2	2			
	NC,rr	E1,rr	C=0	*MODEL2/3 (Minimum mode)* If Condition is true then [SP-1]←PC(H), [SP-2]←PC(L), SP←SP-2, PC←PC+rr+1, CB←NB	4			
				else PC←PC+2, NB←CB	2			
	Z,rr	E2,rr	Z=1	*MODEL2/3 (Maximum mode)* If Condition is true then [SP-1]←CB, [SP-2]←PC(H), [SP-3]←PC(L), SP←SP-3, PC←PC+rr+1, CB←NB	5			
				else PC←PC+2, NB←CB	2			
	NZ,rr	E3,rr	Z=0					
CARS	LT,rr	CE,F0,rr	[N∀V]=1	*MODEL0/1* If Condition is true then [SP-1]←PC(H), [SP-2]←PC(L), SP←SP-2, PC←PC+rr+2	5	3	– – – – – – – –	*88*
	LE,rr	CE,F1,rr	Z∨[N∀V]=1	else PC←PC+3	3			
	GT,rr	CE,F2,rr	Z∨[N∀V]=0	*MODEL2/3 (Minimum mode)* If Condition is true then [SP-1]←PC(H), [SP-2]←PC(L), SP←SP-2, PC←PC+rr+2, CB←NB	5			
	GE,rr	CE,F3,rr	[N∀V]=0	else PC←PC+3, NB←CB	3			
	V,rr	CE,F4,rr	V=1	*MODEL2/3 (Maximum mode)* If Condition is true then [SP-1]←CB, [SP-2]←PC(H), [SP-3]←PC(L), SP←SP-3, PC←PC+rr+2, CB←NB	6			
	NV,rr	CE,F5,rr	V=0	else PC←PC+3, NB←CB	3			
	P,rr	CE,F6,rr	N=0					
	M,rr	CE,F7,rr	N=1					
	F0,rr	CE,F8,rr	F0=1					
	F1,rr	CE,F9,rr	F1=1					
	F2,rr	CE,FA,rr	F2=1					
	F3,rr	CE,FB,rr	F3=1					
	NF0,rr	CE,FC,rr	F0=0					
	NF1,rr	CE,FD,rr	F1=0					
	NF2,rr	CE,FE,rr	F2=0					
	NF3,rr	CE,FF,rr	F3=0					

Branch Instructions (3/4)

Mnemonic		Machine Code	Condition	Operation	Cycle	Byte	SC								Page
							I1	I0	U	D	N	V	C	Z	
CARL	qqrr	F2,rr,qq	Unconditionable	*MODEL0/1* [SP-1]←PC(H), [SP-2]←PC(L), SP←SP-2, PC←PC+qqrr+2	5	3	–	–	–	–	–	–	–	–	*84*
				MODEL2/3 (Minimum mode) [SP-1]←PC(H), [SP-2]←PC(L), SP←SP-2, PC←PC+qqrr+2, CB←NB											
				MODEL2/3 (Maximum mode) [SP-1]←CB, [SP-2]←PC(H), [SP-3]←PC(L), SP←SP-3, PC←PC+qqrr+2, CB←NB	6										
CARL	C,qqrr	E8,rr,qq	C=1	*MODEL0/1* If Condition is true then [SP-1]←PC(H), [SP-2]←PC(L), SP←SP-2, PC←PC+qqrr+2	5	3	–	–	–	–	–	–	–	–	*85*
	NC,qqrr	E9,rr,qq	C=0	else PC←PC+3	3										
				MODEL2/3 (Minimum mode) If Condition is true then [SP-1]←PC(H), [SP-2]←PC(L), SP←SP-2, PC←PC+qqrr+2, CB←NB	5										
	Z,qqrr	EA,rr,qq	Z=1	else PC←PC+3, NB←CB	3										
	NZ,qqrr	EB,rr,qq	Z=0	*MODEL2/3 (Maximum mode)* If Condition is true then [SP-1]←CB, [SP-2]←PC(H), [SP-3]←PC(L), SP←SP-3, PC←PC+qqrr+2, CB←NB	6										
				else PC←PC+3, NB←CB	3										
CALL	[hh*ll*]	FB,*ll*,hh	Unconditionable	*MODEL0/1* [SP-1]←PC(H), [SP-2]←PC(L), SP←SP-2, PC(L)←[hh*ll*], PC(H)←[hh*ll*+1]	7	3	–	–	–	–	–	–	–	–	*83*
				MODEL2/3 (Minimum mode) [SP-1]←PC(H), [SP-2]←PC(L), SP←SP-2, PC(L)←[hh*ll*], PC(H)←[hh*ll*+1], CB←NB											
				MODEL2/3 (Maximum mode) [SP-1]←CB, [SP-2]←PC(H), [SP-3]←PC(L), SP←SP-3, PC(L)←[hh*ll*], PC(H)←[hh*ll*+1], CB←NB	8										

Branch Instructions (4/4)

Mnemonic		Machine Code	Operation	Cycle	Byte	SC								Comment	Page
						I1	I0	U	D	N	V	C	Z		
INT	[kk]	FC,kk	*MODEL0/1* [SP-1]←PC(H), [SP-2]←PC(L), [SP-3]←SC, SP←SP-3, PC(L)←[00kk], PC(H)←[00kk+1]	7	2	–	–	–	–	–	–	–	–		*108*
			MODEL2/3 (Minimum mode) [SP-1]←PC(H), [SP-2]←PC(L), [SP-3]←SC, SP←SP-3, PC(L)←[00kk], PC(H)←[00kk+1], CB←NB												
			MODEL2/3 (Maximum mode) [SP-1]←CB, [SP-2]←PC(H), [SP-3]←PC(L), [SP-4]←SC, SP←SP-4, PC(L)←[00kk], PC(H)←[00kk+1], CB←NB	8											
RET		F8	*MODEL0/1, MODEL2/3 (Minimum mode)* PC(L)←[SP], PC(H)←[SP+1], SP←SP+2	3	1	–	–	–	–	–	–	–	–		*161*
			MODEL2/3 (Maximum mode) PC(L)←[SP], PC(H)←[SP+1], CB←[SP+2], NB←CB, SP←SP+3	4											
RETE		F9	*MODEL0/1, MODEL2/3 (Minimum mode)* SC←[SP], PC(L)←[SP+1], PC(H)←[SP+2], SP←SP+3	4	1	↕	↕	↕	↕	↕	↕	↕	↕		*161*
			MODEL2/3 (Maximum mode) SC←[SP], PC(L)←[SP+1], PC(H)←[SP+2], CB←[SP+3], NB←CB, SP←SP+4	5											
RETS		FA	*MODEL0/1, MODEL2/3 (Minimum mode)* PC(L)←[SP], PC(H)←[SP+1], SP←SP+2, PC←PC+2	5	1	–	–	–	–	–	–	–	–		*162*
			MODEL2/3 (Maximum mode) PC(L)←[SP], PC(H)←[SP+1], CB←[SP+2], NB←CB, SP←SP+3, PC←PC+2	6											

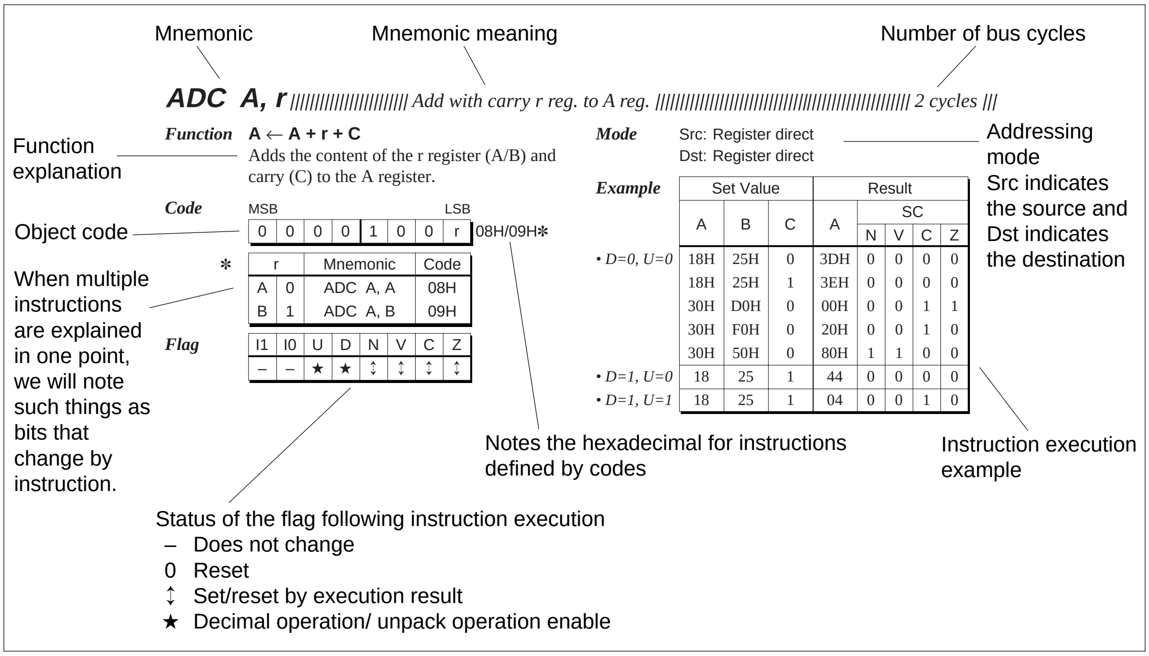
System Control Instructions

Mnemonic		Machine Code	Operation	Cycle	Byte	SC								Comment	Page
						I1	I0	U	D	N	V	C	Z		
NOP		FF	No Operation	2	1	–	–	–	–	–	–	–	–		*149*
HALT		CE,AE	HALT	3	2	–	–	–	–	–	–	–	–		*106*
SLP		CE,AF	SLEEP	3	2	–	–	–	–	–	–	–	–		*176*

4.4 Detailed Explanation of Instructions

Here we will explain each instruction individually.
This explanation will be given according to the following format.

View of the explanation



ADC A, r *Add with carry r reg. to A reg.* *2 cycles*

Function **A ← A + r + C**
Adds the content of the r register (A/B) and carry (C) to the A register.

Code MSB 0 0 0 0 1 0 0 r LSB 08H/09H✻

✻

r		Mnemonic	Code
A	0	ADC A, A	08H
B	1	ADC A, B	09H

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Register direct
Dst: Register direct

Example

	Set Value			Result				
	A	B	C	A	SC			
					N	V	C	Z
• *D=0, U=0*	18H	25H	0	3DH	0	0	0	0
	18H	25H	1	3EH	0	0	0	0
	30H	D0H	0	00H	0	0	1	1
	30H	F0H	0	20H	0	0	1	0
	30H	50H	0	80H	1	1	0	0
• *D=1, U=0*	18	25	1	44	0	0	0	0
• *D=1, U=1*	18	25	1	04	0	0	1	0

Mnemonic
Mnemonic meaning
Number of bus cycles
Function explanation
Object code
When multiple instructions are explained in one point, we will note such things as bits that change by instruction.
Addressing mode Src indicates the source and Dst indicates the destination
Notes the hexadecimal for instructions defined by codes
Instruction execution example
Status of the flag following instruction execution
– Does not change
0 Reset
↕ Set/reset by execution result
★ Decimal operation/ unpack operation enable

The meaning of the symbols are the same as for the instruction list. See section "4.3.2 Symbol meanings".

We will use the below symbols when explaining multiple registers as aggregations.

r .... Data registers A/B, or A/B/L/H
ir .... Index registers IX/IY
rp .... 16-bit (pair) registers BA/HL or 16-bit registers (BA)/HL/IX/IY/(SP)
er .... New code bank register NB and expand page registers EP/XP/YP

cc1 ..... Branch conditions C/NC/Z/NZ
cc2 ..... Branch conditions LT/LE/GT/GE/V/NV/P/M/F0/F1/F2/F3/NF0/NF1/NF2/NF3

Instructions for which the number of bus cycles differ in the maximum mode and minimum mode are indicated with (MAX) and (MIN) for the number of cycles. MIN includes the MODEL0/1.

ADC A, r ||||||||||||||||||||||| *Add with carry r reg. to A reg.* |||||||||||||||||||||||||||||||||||||||||||||||||| *2 cycles* |||

Function **A ← A + r + C**

Adds the content of the r register (A/B) and carry (C) to the A register.

Code

MSB LSB

0	0	0	0	1	0	0	r	08H/09H✻

✻

r		Mnemonic	Code
A	0	ADC A, A	08H
B	1	ADC A, B	09H

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Register direct
Dst: Register direct

Example

	Set Value			Result				
	A	B	C	A	SC			
					N	V	C	Z
• *D=0, U=0*	18H	25H	0	3DH	0	0	0	0
	18H	25H	1	3EH	0	0	0	0
	30H	D0H	0	00H	0	0	1	1
	30H	F0H	0	20H	0	0	1	0
	30H	50H	0	80H	1	1	0	0
• *D=1, U=0*	18	25	1	44	0	0	0	0
• *D=1, U=1*	18	25	1	04	0	0	1	0

ADC A, #nn |||||||||||||||| *Add with carry immediate data nn to A reg.* ||||||||||||||||||||||||||||| *2 cycles* |||

Function **A ← A + nn + C**

Adds 8-bit immediate data nn and carry (C) to the A register.

Code

MSB LSB

0	0	0	0	1	0	1	0	0AH
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Immediate data
Dst: Register direct

Example

	Set Value			Result				
	A	nn	C	A	SC			
					N	V	C	Z
• *D=0, U=0*	18H	25H	0	3DH	0	0	0	0
	18H	25H	1	3EH	0	0	0	0
	30H	D0H	0	00H	0	0	1	1
	30H	F0H	0	20H	0	0	1	0
	30H	50H	0	80H	1	1	0	0
• *D=1, U=0*	18	25	1	44	0	0	0	0
• *D=1, U=1*	18	25	1	04	0	0	1	0

ADC A, [BR:ll] |||||||||| *Add with carry location [BR:ll] to A reg.* ||||||||||||||||||||||||||||||||| *3 cycles* |||

Function **A ← A + [BR:*ll*] + C**

Adds the content of the data memory and the carry (C) to the A register. The data memory address has been specified by the content of the BR register (upper byte specification) and the 8-bit absolute address *ll* (lower byte specification).
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB LSB

0	0	0	0	1	1	0	0	0CH
			l	*l*				*ll*

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: 8-bit absolute
Dst: Register direct

Example

	Set Value			Result				
	A	[BR:*ll*]	C	A	SC			
					N	V	C	Z
• *D=0, U=0*	18H	25H	0	3DH	0	0	0	0
	18H	25H	1	3EH	0	0	0	0
	30H	D0H	0	00H	0	0	1	1
	30H	F0H	0	20H	0	0	1	0
	30H	50H	0	80H	1	1	0	0
• *D=1, U=0*	18	25	1	44	0	0	0	0
• *D=1, U=1*	18	25	1	04	0	0	1	0

ADC A, [hhll] *Add with carry location [hhll] to A reg.* *4 cycles*

Function **A ← A + [hh*ll*] + C**

Adds the content of the data memory and the carry (C) to the A register. The data memory address has been specified by the 16-bit absolute address hh*ll*.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
0	0	0	0	1	1	0	1	0DH
			l	*l*				*ll*
			h	h				hh

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: 16-bit absolute
Dst: Register direct

Example

	Set Value			Result				
	A	[hh*ll*]	C	A	SC			
					N	V	C	Z
• *D=0, U=0*	18H	25H	0	3DH	0	0	0	0
	18H	25H	1	3EH	0	0	0	0
	30H	D0H	0	00H	0	0	1	1
	30H	F0H	0	20H	0	0	1	0
	30H	50H	0	80H	1	1	0	0
• *D=1, U=0*	18	25	1	44	0	0	0	0
• *D=1, U=1*	18	25	1	04	0	0	1	0

ADC A, [HL] *Add with carry location [HL] to A reg.* *2 cycles*

Function **A ← A + [HL] + C**

Adds the content of the data memory and the carry (C) to the A register. The data memory address has been specified by the HL register.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
0	0	0	0	1	0	1	1	0BH

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Register indirect
Dst: Register direct

Example

	Set Value			Result				
	A	[HL]	C	A	SC			
					N	V	C	Z
• *D=0, U=0*	18H	25H	0	3DH	0	0	0	0
	18H	25H	1	3EH	0	0	0	0
	30H	D0H	0	00H	0	0	1	1
	30H	F0H	0	20H	0	0	1	0
	30H	50H	0	80H	1	1	0	0
• *D=1, U=0*	18	25	1	44	0	0	0	0
• *D=1, U=1*	18	25	1	04	0	0	1	0

ADC A, [ir] |||||||||||||||||| *Add with carry location [ir reg.] to A reg.* |||||||||||||||||||||||||||||||||| *2 cycles* |||

Function **A ← A + [ir] + C**

Adds the content of the data memory and the carry (C) to the A register. The data memory address has been specified by the ir register (IX/IY).
The content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory (MODEL2/3).

Code

MSB LSB

0	0	0	0	1	1	1	ir	0EH/0FH✻

✻

ir		Mnemonic	Code
IX	0	ADC A, [IX]	0EH
IY	1	ADC A, [IY]	0FH

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Register indirect
Dst: Register direct

Example

	Set Value			Result				
	A	[ir]	C	A	SC			
					N	V	C	Z
• *D=0, U=0*	18H	25H	0	3DH	0	0	0	0
	18H	25H	1	3EH	0	0	0	0
	30H	D0H	0	00H	0	0	1	1
	30H	F0H	0	20H	0	0	1	0
	30H	50H	0	80H	1	1	0	0
• *D=1, U=0*	18	25	1	44	0	0	0	0
• *D=1, U=1*	18	25	1	04	0	0	1	0

ADC A, [ir+dd] ||||||||| *Add with carry location [ir reg. + dd] to A reg.* |||||||||||||||||||||||| *4 cycles* |||

Function **A ← A + [ir+dd] + C**

Adds the content of the data memory and the carry (C) to the A register. The data memory address has been specified by the sum of the content of the ir register (IX/IY) and the displacement dd.
The displacement dd is handled as signed data and the range is -128 to 127.
The content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory (MODEL2/3).

Code

MSB LSB

1	1	0	0	1	1	1	0	CEH
0	0	0	0	1	0	0	ir	08H/09H✻
			d	d				dd

✻

ir		Mnemonic	Code
IX	0	ADC A, [IX+dd]	08H
IY	1	ADC A, [IY+dd]	09H

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Register indirect with displacement
Dst: Register direct

Example

	Set Value			Result				
	A	[ir+dd]	C	A	SC			
					N	V	C	Z
• *D=0, U=0*	18H	25H	0	3DH	0	0	0	0
	18H	25H	1	3EH	0	0	0	0
	30H	D0H	0	00H	0	0	1	1
	30H	F0H	0	20H	0	0	1	0
	30H	50H	0	80H	1	1	0	0
• *D=1, U=0*	18	25	1	44	0	0	0	0
• *D=1, U=1*	18	25	1	04	0	0	1	0

ADC A, [ir+L] |||||||||||| *Add with carry location [ir reg. + L] to A reg.* |||||||||||||||||||||||||| *4 cycles* |||

Function **A ← A + [ir+L] + C**

Adds the content of the data memory and the carry (C) to the A register. The data memory address has been specified by the sum of the content of the ir register (IX/IY) and the content of the L register.
The content of the L register is handled as signed data and the range is -128 to 127.
The content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	0	1	0	1	ir	0AH/0BH✻

✻

ir		Mnemonic	Code
IX	0	ADC A, [IX+L]	0AH
IY	1	ADC A, [IY+L]	0BH

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Register indirect with index register
Dst: Register direct

Example

	Set Value			Result				
					SC			
	A	[ir+L]	C	A	N	V	C	Z
• *D=0, U=0*	18H	25H	0	3DH	0	0	0	0
	18H	25H	1	3EH	0	0	0	0
	30H	D0H	0	00H	0	0	1	1
	30H	F0H	0	20H	0	0	1	0
	30H	50H	0	80H	1	1	0	0
• *D=1, U=0*	18	25	1	44	0	0	0	0
• *D=1, U=1*	18	25	1	04	0	0	1	0

ADC [HL], A |||||||||||||| *Add with carry A reg. to location [HL]* |||||||||||||||||||||||||||||||||||| *4 cycles* |||

Function **[HL] ← [HL] + A + C**

Adds the content of the A register and the carry (C) to the data memory that has been address specified by the HL register.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	0	1	1	0	0	0CH

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Register direct
Dst: Register indirect

Example

	Set Value			Result				
					SC			
	[HL]	A	C	[HL]	N	V	C	Z
• *D=0, U=0*	18H	25H	0	3DH	0	0	0	0
	18H	25H	1	3EH	0	0	0	0
	30H	D0H	0	00H	0	0	1	1
	30H	F0H	0	20H	0	0	1	0
	30H	50H	0	80H	1	1	0	0
• *D=1, U=0*	18	25	1	44	0	0	0	0
• *D=1, U=1*	18	25	1	04	0	0	1	0

ADC [HL], #nn ||||||||| *Add with carry immediate data nn to location [HL] |||||||||||||||||| 5 cycles |||*

Function **[HL] ← [HL] + nn + C**
Adds the 8-bit immediate data nn and the carry (C) to the data memory that has been address specified by the HL register.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	0	1	1	0	1	0DH
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Immediate data
Dst: Register indirect

Example

	Set Value			Result				
	[HL]	nn	C	[HL]	SC			
					N	V	C	Z
• *D=0, U=0*	18H	25H	0	3DH	0	0	0	0
	18H	25H	1	3EH	0	0	0	0
	30H	D0H	0	00H	0	0	1	1
	30H	F0H	0	20H	0	0	1	0
	30H	50H	0	80H	1	1	0	0
• *D=1, U=0*	18	25	1	44	0	0	0	0
• *D=1, U=1*	18	25	1	04	0	0	1	0

ADC [HL], [ir] |||||||||||| *Add with carry location [ir reg.] to location [HL] |||||||||||||||||||| 5 cycles |||*

Function **[HL] ← [HL] + [ir] + C**
Adds the content of the data memory that has been address specified by the ir register (IX/IY) and the carry (C) to the data memory that has been address specified by the HL register.
The content of the EP register becomes the page address of the data memory [HL] and the content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory [ir] (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	0	1	1	1	ir	0EH/0FH✻

✻

ir		Mnemonic	Code
IX	0	ADC [HL], [IX]	0EH
IY	1	ADC [HL], [IY]	0FH

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Register indirect
Dst: Register indirect

Example

	Set Value			Result				
	[HL]	[ir]	C	[HL]	SC			
					N	V	C	Z
• *D=0, U=0*	18H	25H	0	3DH	0	0	0	0
	18H	25H	1	3EH	0	0	0	0
	30H	D0H	0	00H	0	0	1	1
	30H	F0H	0	20H	0	0	1	0
	30H	50H	0	80H	1	1	0	0
• *D=1, U=0*	18	25	1	44	0	0	0	0
• *D=1, U=1*	18	25	1	04	0	0	1	0

ADC BA, rp *Add with carry rp reg. to BA reg.* *4 cycles*

Function **BA ← BA + rp + C**

Adds the content of the rp register (BA/HL/IX/IY) and the carry (C) to the BA register.

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	0	0	0	0	1	rp		04H–07H✻

✻

rp		Mnemonic	Code
BA	00	ADC BA, BA	04H
HL	01	ADC BA, HL	05H
IX	10	ADC BA, IX	06H
IY	11	ADC BA, IY	07H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Register direct
Dst: Register direct

Example

Set Value			Result				
BA	rp	C	BA	SC			
				N	V	C	Z
1380H	3546H	0	48C6H	0	0	0	0
1380H	3546H	1	48C7H	0	0	0	0
1380H	EC80H	0	0000H	0	0	1	1
5218H	4174H	0	938CH	1	1	0	0
5342H	C32AH	1	166DH	0	0	1	0

(rp≠BA)

ADC BA, #mmnn *Add with carry immediate data mmnn to BA reg.* *4 cycles*

Function **BA ← BA + mmnn + C**

Adds the 16-bit immediate data mmnn and the carry (C) to the BA register.

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	1	1	0	0	0	0	0	60H
			n	n				nn
			m	m				mm

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Immediate data
Dst: Register direct

Example

Set Value			Result				
BA	mmnn	C	BA	SC			
				N	V	C	Z
1380H	3546H	0	48C6H	0	0	0	0
1380H	3546H	1	48C7H	0	0	0	0
1380H	EC80H	0	0000H	0	0	1	1
5218H	4174H	0	938CH	1	1	0	0
5342H	C32AH	1	166DH	0	0	1	0

ADC HL, rp ||||||||||||||| Add with carry rp reg. to HL reg. ||||||||||||||||||||||||||||||||||||||||||||| 4 cycles |||

Function **HL ← HL + rp + C**

Adds the content of the rp register (BA/HL/IX/IY) and the carry (C) to the HL register.

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	0	1	0	0	1	rp		24H–27H✻

✻

rp		Mnemonic	Code
BA	00	ADC HL, BA	24H
HL	01	ADC HL, HL	25H
IX	10	ADC HL, IX	26H
IY	11	ADC HL, IY	27H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Register direct
Dst: Register direct

Example

Set Value			Result				
HL	rp	C	HL	SC			
				N	V	C	Z
1380H	3546H	0	48C6H	0	0	0	0
1380H	3546H	1	48C7H	0	0	0	0
1380H	EC80H	0	0000H	0	0	1	1
5218H	4174H	0	938CH	1	1	0	0
5342H	C32AH	1	166DH	0	0	1	0

(rp≠HL)

ADC HL, #mmnn ||| Add with carry immediate data mmnn to HL reg. ||||||||||||||||||||| 4 cycles |||

Function **HL ← HL + mmnn + C**

Adds the 16-bit immediate data mmnn and the carry (C) to the HL register.

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	1	1	0	0	0	0	1	61H
			n	n				nn
			m	m				mm

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Immediate data
Dst: Register direct

Example

Set Value			Result				
HL	mmnn	C	HL	SC			
				N	V	C	Z
1380H	3546H	0	48C6H	0	0	0	0
1380H	3546H	1	48C7H	0	0	0	0
1380H	EC80H	0	0000H	0	0	1	1
5218H	4174H	0	938CH	1	1	0	0
5342H	C32AH	1	166DH	0	0	1	0

ADD A, r *Add r reg. to A reg.* *2 cycles*

Function **A ← A + r**
Adds the content of the r register (A/B) to the A register.

Mode Src: Register direct
Dst: Register direct

Code

MSB							LSB	
0	0	0	0	0	0	0	r	00H/01H✻

✻

r		Mnemonic	Code
A	0	ADD A, A	00H
B	1	ADD A, B	01H

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Example

	Set Value		Result				
	A	B	A	SC			
				N	V	C	Z
• *D=0, U=0*	18H	25H	3DH	0	0	0	0
	30H	D0H	00H	0	0	1	1
	30H	F0H	20H	0	0	1	0
	30H	50H	80H	1	1	0	0
• *D=1, U=0*	18	24	42	0	0	0	0
• *D=1, U=1*	18	24	02	0	0	1	0

ADD A, #nn *Add immediate data nn to A reg.* *2 cycles*

Function **A ← A + nn**
Adds 8-bit immediate data nn to the A register.

Mode Src: Immediate data
Dst: Register direct

Code

MSB							LSB	
0	0	0	0	0	0	1	0	02H
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Example

	Set Value		Result				
	A	nn	A	SC			
				N	V	C	Z
• *D=0, U=0*	18H	25H	3DH	0	0	0	0
	30H	D0H	00H	0	0	1	1
	30H	F0H	20H	0	0	1	0
	30H	50H	80H	1	1	0	0
• *D=1, U=0*	18	24	42	0	0	0	0
• *D=1, U=1*	18	24	02	0	0	1	0

ADD A, [BR:ll] *Add location [BR:ll] to A reg.* *3 cycles*

Function **A ← A + [BR:*ll*]**
Adds the content of the data memory to the A register. The data memory address has been specified by the content of the BR register (upper byte specification) and the 8-bit absolute address *ll* (lower byte specification). The content of the EP register becomes the page address of the data memory (MODEL2/3).

Mode Src: 8-bit absolute
Dst: Register direct

Code

MSB							LSB	
0	0	0	0	0	1	0	0	04H
			l	*l*				*ll*

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Example

	Set Value		Result				
	A	[BR:*ll*]	A	SC			
				N	V	C	Z
• *D=0, U=0*	18H	25H	3DH	0	0	0	0
	30H	D0H	00H	0	0	1	1
	30H	F0H	20H	0	0	1	0
	30H	50H	80H	1	1	0	0
• *D=1, U=0*	18	24	42	0	0	0	0
• *D=1, U=1*	18	24	02	0	0	1	0

ADD A, [hhll] *Add location [hhll] to A reg.* *4 cycles*

Function **A ← A + [hh*ll*]**

Adds the content of the data memory that has been address specified by the 16-bit absolute address hh*ll* to the A register.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
0	0	0	0	0	1	0	1	05H
			l	*l*				*ll*
			h	h				hh

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: 16-bit absolute
Dst: Register direct

Example

	Set Value		Result				
				SC			
	A	[hh*ll*]	A	N	V	C	Z
• *D=0, U=0*	18H	25H	3DH	0	0	0	0
	30H	D0H	00H	0	0	1	1
	30H	F0H	20H	0	0	1	0
	30H	50H	80H	1	1	0	0
• *D=1, U=0*	18	24	42	0	0	0	0
• *D=1, U=1*	18	24	02	0	0	1	0

ADD A, [HL] *Add location [HL] to A reg.* *2 cycles*

Function **A ← A + [HL]**

Adds the content of the data memory that has been address specified by the HL register to the A register.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
0	0	0	0	0	0	1	1	03H

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Register indirect
Dst: Register direct

Example

	Set Value		Result				
				SC			
	A	[HL]	A	N	V	C	Z
• *D=0, U=0*	18H	25H	3DH	0	0	0	0
	30H	D0H	00H	0	0	1	1
	30H	F0H	20H	0	0	1	0
	30H	50H	80H	1	1	0	0
• *D=1, U=0*	18	24	42	0	0	0	0
• *D=1, U=1*	18	24	02	0	0	1	0

ADD A, [ir] *Add location [ir reg.] to A reg.* *2 cycles*

Function **A ← A + [ir]**

Adds the content of the data memory that has been address specified by the ir register (IX/IY) to the A register.
The content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
0	0	0	0	0	1	1	ir	06H/07H✻

✻

ir		Mnemonic	Code
IX	0	ADD A, [IX]	06H
IY	1	ADD A, [IY]	07H

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Register indirect
Dst: Register direct

Example

	Set Value		Result				
				SC			
	A	[ir]	A	N	V	C	Z
• *D=0, U=0*	18H	25H	3DH	0	0	0	0
	30H	D0H	00H	0	0	1	1
	30H	F0H	20H	0	0	1	0
	30H	50H	80H	1	1	0	0
• *D=1, U=0*	18	24	42	0	0	0	0
• *D=1, U=1*	18	24	02	0	0	1	0

ADD A, [ir+dd] *Add location [ir reg. + dd] to A reg.* *4 cycles*

Function **A ← A + [ir+dd]**

Adds the content of the data memory to the A register. The data memory address has been specified by the sum of the content of the ir register (IX/IY) and the displacement dd.
The displacement dd is handled as signed data and the range is -128 to 127.
The content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	0	0	0	0	ir	00H/01H✻
			d	d				dd

✻

ir		Mnemonic	Code
IX	0	ADD A, [IX+dd]	00H
IY	1	ADD A, [IY+dd]	01H

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Register indirect with displacement
Dst: Register direct

Example

	Set Value		Result				
	A	[ir+dd]	A	SC			
				N	V	C	Z
• D=0, U=0	18H	25H	3DH	0	0	0	0
	30H	D0H	00H	0	0	1	1
	30H	F0H	20H	0	0	1	0
	30H	50H	80H	1	1	0	0
• D=1, U=0	18	24	42	0	0	0	0
• D=1, U=1	18	24	02	0	0	1	0

ADD A, [ir+L] *Add location [ir reg. + L] to A reg.* *4 cycles*

Function **A ← A + [ir+L]**

Adds the content of the data memory to the A register. The data memory address has been specified by the sum of the content of the ir register (IX/IY) and the content of the L register.
The content of the L register is handled as signed data and the range is -128 to 127.
The content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	0	0	0	1	ir	02H/03H✻

✻

ir		Mnemonic	Code
IX	0	ADD A, [IX+L]	02H
IY	1	ADD A, [IY+L]	03H

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Register indirect with index register
Dst: Register direct

Example

	Set Value		Result				
	A	[ir+L]	A	SC			
				N	V	C	Z
• D=0, U=0	18H	25H	3DH	0	0	0	0
	30H	D0H	00H	0	0	1	1
	30H	F0H	20H	0	0	1	0
	30H	50H	80H	1	1	0	0
• D=1, U=0	18	24	42	0	0	0	0
• D=1, U=1	18	24	02	0	0	1	0

ADD [HL], A |||||||||||||| *Add A reg. to location [HL]* |||||||||||||||||||||||||||||||||||||||||||||||||||||| *4 cycles* |||

Function **[HL] ← [HL] + A**

Adds the content of the A register to the data memory that has been address specified by the HL register.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	0	0	1	0	0	04H

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Register direct
Dst: Register indirect

Example

	Set Value		Result				
	[HL]	A	[HL]	SC			
				N	V	C	Z
• *D=0, U=0*	18H	25H	3DH	0	0	0	0
	30H	D0H	00H	0	0	1	1
	30H	F0H	20H	0	0	1	0
	30H	50H	80H	1	1	0	0
• *D=1, U=0*	18	24	42	0	0	0	0
• *D=1, U=1*	18	24	02	0	0	1	0

ADD [HL], #nn ||||||||| *Add immediate data nn to location [HL]* |||||||||||||||||||||||||||||||||| *5 cycles* |||

Function **[HL] ← [HL] + nn**

Adds the 8-bit immediate data nn to the data memory that has been address specified by the HL register.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	0	0	1	0	1	05H
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Immediate data
Dst: Register indirect

Example

	Set Value		Result				
	[HL]	nn	[HL]	SC			
				N	V	C	Z
• *D=0, U=0*	18H	25H	3DH	0	0	0	0
	30H	D0H	00H	0	0	1	1
	30H	F0H	20H	0	0	1	0
	30H	50H	80H	1	1	0	0
• *D=1, U=0*	18	24	42	0	0	0	0
• *D=1, U=1*	18	24	02	0	0	1	0

ADD [HL], [ir] |||||||||| *Add location [ir reg.] to location [HL]* |||||||||||||||||||||||||||||||||||| *5 cycles* |||

Function **[HL] ← [HL] + [ir]**

Adds the content of the data memory that has been address specified by the ir register (IX/IY) to the data memory that has been address specified by the HL register.
The content of the EP register becomes the page address of the data memory [HL] and the content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory [ir] (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	0	0	1	1	ir	06H/07H✻

✻

ir		Mnemonic	Code
IX	0	ADD [HL], [IX]	06H
IY	1	ADD [HL], [IY]	07H

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Register indirect
Dst: Register indirect

Example

	Set Value		Result				
	[HL]	[ir]	[HL]	SC			
				N	V	C	Z
• *D=0, U=0*	18H	25H	3DH	0	0	0	0
	30H	D0H	00H	0	0	1	1
	30H	F0H	20H	0	0	1	0
	30H	50H	80H	1	1	0	0
• *D=1, U=0*	18	24	42	0	0	0	0
• *D=1, U=1*	18	24	02	0	0	1	0

ADD BA, rp |||||||||||||||| *Add rp reg. to BA reg.* ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *4 cycles* |||

Function **BA ← BA + rp**

Adds the content of the rp register (BA/HL/IX/IY) to the BA register.

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	0	0	0	0	0	rp		00H–03H✻

✻

rp		Mnemonic	Code
BA	00	ADD BA, BA	00H
HL	01	ADD BA, HL	01H
IX	10	ADD BA, IX	02H
IY	11	ADD BA, IY	03H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Register direct
Dst: Register direct

Example

Set Value		Result				
BA	rp	BA	SC			
			N	V	C	Z
1380H	3546H	48C6H	0	0	0	0
1380H	EC80H	0000H	0	0	1	1
5218H	4174H	938CH	1	1	0	0
5342H	C32AH	166CH	0	0	1	0

(rp≠BA)

ADD BA, #mmnn *Add immediate data mmnn to BA reg.* *3 cycles*

Function **BA ← BA + mmnn**
Adds the 16-bit immediate data mmnn to the BA register.

Code

MSB							LSB	
1	1	0	0	0	0	0	0	C0H
			n	n				nn
			m	m				mm

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result				
BA	mmnn	BA	SC			
			N	V	C	Z
1380H	3546H	48C6H	0	0	0	0
1380H	EC80H	0000H	0	0	1	1
5218H	4174H	938CH	1	1	0	0
5342H	C32AH	166CH	0	0	1	0

ADD HL, rp *Add rp reg. to HL reg.* *4 cycles*

Function **HL ← HL + rp**
Adds the content of the rp register (BA/HL/IX/IY) to the HL register.

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	0	1	0	0	0	rp		20H–23H✻

✻

rp		Mnemonic	Code
BA	00	ADD HL, BA	20H
HL	01	ADD HL, HL	21H
IX	10	ADD HL, IX	22H
IY	11	ADD HL, IY	23H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Register direct
Dst: Register direct

Example

Set Value		Result				
HL	rp	HL	SC			
			N	V	C	Z
1380H	3546H	48C6H	0	0	0	0
1380H	EC80H	0000H	0	0	1	1
5218H	4174H	938CH	1	1	0	0
5342H	C32AH	166CH	0	0	1	0

(rp≠HL)

ADD HL, #mmnn *Add immediate data mmnn to HL reg.* *3 cycles*

Function **HL ← HL + mmnn**
Adds the 16-bit immediate data mmnn to the HL register.

Code

MSB							LSB	
1	1	0	0	0	0	0	1	C1H
			n	n				nn
			m	m				mm

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result				
HL	mmnn	HL	SC			
			N	V	C	Z
1380H	3546H	48C6H	0	0	0	0
1380H	EC80H	0000H	0	0	1	1
5218H	4174H	938CH	1	1	0	0
5342H	C32AH	166CH	0	0	1	0

ADD IX, rp *Add rp reg. to IX reg. 4 cycles*

Function **IX ← IX + rp**
Adds the content of the rp register (BA/HL) to the IX register.

Mode Src: Register direct
Dst: Register direct

Code

MSB … LSB

1	1	0	0	1	1	1	1	CFH
0	1	0	0	0	0	0	rp	40H/41H✻

✻

rp		Mnemonic	Code
BA	0	ADD IX, BA	40H
HL	1	ADD IX, HL	41H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Example

Set Value		Result				
IX	rp	IX	SC			
			N	V	C	Z
1380H	3546H	48C6H	0	0	0	0
1380H	EC80H	0000H	0	0	1	1
5218H	4174H	938CH	1	1	0	0
5342H	C32AH	166CH	0	0	1	0

ADD IX, #mmnn *Add immediate data mmnn to IX reg. 3 cycles*

Function **IX ← IX + mmnn**
Adds the 16-bit immediate data mmnn to the IX register.

Mode Src: Immediate data
Dst: Register direct

Code

MSB … LSB

1	1	0	0	0	0	1	0	C2H
			n	n				nn
			m	m				mm

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Example

Set Value		Result				
IX	mmnn	IX	SC			
			N	V	C	Z
1380H	3546H	48C6H	0	0	0	0
1380H	EC80H	0000H	0	0	1	1
5218H	4174H	938CH	1	1	0	0
5342H	C32AH	166CH	0	0	1	0

ADD IY, rp *Add rp reg. to IY reg. 4 cycles*

Function **IY ← IY + rp**
Adds the content of the rp register (BA/HL) to the IY register.

Mode Src: Register direct
Dst: Register direct

Code

MSB … LSB

1	1	0	0	1	1	1	1	CFH
0	1	0	0	0	0	1	rp	42H/43H✻

✻

rp		Mnemonic	Code
BA	0	ADD IY, BA	42H
HL	1	ADD IY, HL	43H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Example

Set Value		Result				
IY	rp	IY	SC			
			N	V	C	Z
1380H	3546H	48C6H	0	0	0	0
1380H	EC80H	0000H	0	0	1	1
5218H	4174H	938CH	1	1	0	0
5342H	C32AH	166CH	0	0	1	0

ADD IY, #mmnn *Add immediate data mmnn to IY reg. 3 cycles*

Function **IY ← IY + mmnn**
Adds the 16-bit immediate data mmnn to the IY register.

Mode Src: Immediate data
Dst: Register direct

Code

MSB … LSB

1	1	0	0	0	0	1	1	C3H
			n	n				nn
			m	m				mm

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Example

Set Value		Result				
IY	mmnn	IY	SC			
			N	V	C	Z
1380H	3546H	48C6H	0	0	0	0
1380H	EC80H	0000H	0	0	1	1
5218H	4174H	938CH	1	1	0	0
5342H	C32AH	166CH	0	0	1	0

ADD SP, rp *Add rp reg. to SP 4 cycles*

Function **SP ← SP + rp**
Adds the content of the rp register (BA/HL) to the stack pointer (SP).

Mode Src: Register direct
Dst: Register direct

Code

MSB … LSB

1	1	0	0	1	1	1	1	CFH
0	1	0	0	0	1	0	rp	44H/45H✻

✻

rp		Mnemonic	Code
BA	0	ADD SP, BA	44H
HL	1	ADD SP, HL	45H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Example

Set Value		Result				
SP	rp	SP	SC			
			N	V	C	Z
1380H	3546H	48C6H	0	0	0	0
1380H	EC80H	0000H	0	0	1	1
5218H	4174H	938CH	1	1	0	0
5342H	C32AH	166CH	0	0	1	0

ADD SP, #mmnn *Add immediate data mmnn to SP 4 cycles*

Function **SP ← SP + mmnn**
Adds the 16-bit immediate data mmnn to the stack pointer (SP).

Mode Src: Immediate data
Dst: Register direct

Code

MSB … LSB

1	1	0	0	1	1	1	1	CFH
0	1	1	0	1	0	0	0	68H
			n	n				nn
			m	m				mm

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Example

Set Value		Result				
SP	mmnn	SP	SC			
			N	V	C	Z
1380H	3546H	48C6H	0	0	0	0
1380H	EC80H	0000H	0	0	1	1
5218H	4174H	938CH	1	1	0	0
5342H	C32AH	166CH	0	0	1	0

AND A, r *Logical AND of r reg. and A reg.* *2 cycles*

Function **A ← A ∧ r**

Takes a logical product of the content of the r register (A/B) and the content of the A register and stores the result in the A register.

Mode Src: Register direct
Dst: Register direct

Example

Set Value		Result				
			SC			
A	B	A	N	V	C	Z
3BH	61H	21H	0	–	–	0
5AH	A5H	00H	0	–	–	1
D6H	93H	92H	1	–	–	0

Code

MSB							LSB	
0	0	1	0	0	0	0	r	20H/21H✻

✻

r		Mnemonic	Code
A	0	AND A, A	20H
B	1	AND A, B	21H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

AND A, #nn *Logical AND of immediate data nn and A reg.* *2 cycles*

Function **A ← A ∧ nn**

Takes a logical product of the 8-bit immediate data nn and the content of the A register and stores the result in the A register.

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result				
			SC			
A	nn	A	N	V	C	Z
3BH	61H	21H	0	–	–	0
5AH	A5H	00H	0	–	–	1
D6H	93H	92H	1	–	–	0

Code

MSB							LSB	
0	0	1	0	0	0	1	0	22H
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

AND A, [BR:ll] *Logical AND of location [BR:ll] and A reg.* *3 cycles*

Function **A ← A ∧ [BR:*ll*]**

Takes a logical product of the content of the data memory and the content of the A register and stores the result in the A register. The data memory address has been specified by the content of the BR register (upper byte specification) and the 8-bit absolute address *ll* (lower byte specification).
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Mode Src: 8-bit absolute
Dst: Register direct

Example

Set Value		Result				
			SC			
A	[BR:*ll*]	A	N	V	C	Z
3BH	61H	21H	0	–	–	0
5AH	A5H	00H	0	–	–	1
D6H	93H	92H	1	–	–	0

Code

MSB							LSB	
0	0	1	0	0	1	0	0	24H
			l	*l*				*ll*

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

AND A, [hhll] |||||||||||| *Logical AND of location [hhll] and A reg.* |||||||||||||||||||||||||||||||||| *4 cycles* |||

Function **A ← A ∧ [hh*ll*]**

Takes a logical product of the content of the data memory that has been address specified by the 16-bit absolute address hh*ll* and the content of the A register and stores the result in the A register.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
0	0	1	0	0	1	0	1	25H
			l	*l*				*ll*
			h	h				hh

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: 16-bit absolute
Dst: Register direct

Example

Set Value		Result				
A	[hh*ll*]	A	SC			
			N	V	C	Z
3BH	61H	21H	0	–	–	0
5AH	A5H	00H	0	–	–	1
D6H	93H	92H	1	–	–	0

AND A, [HL] |||||||||||||||| *Logical AND of location [HL] and A reg.* |||||||||||||||||||||||||||||||||||| *2 cycles* |||

Function **A ← A ∧ [HL]**

Takes a logical product of the content of the data memory that has been address specified by the HL register and the content of the A register and stores the result in the A register.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
0	0	1	0	0	0	1	1	23H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Register indirect
Dst: Register direct

Example

Set Value		Result				
A	[HL]	A	SC			
			N	V	C	Z
3BH	61H	21H	0	–	–	0
5AH	A5H	00H	0	–	–	1
D6H	93H	92H	1	–	–	0

AND A, [ir] |||||||||||||||||||| *Logical AND of location [ir reg.] and A reg.* |||||||||||||||||||||||||||||||| *2 cycles* |||

Function **A ← A ∧ [ir]**

Takes a logical product of the content of the data memory that has been address specified by the ir register (IX/IY) and the content of the A register and stores the result in the A register.
The content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
0	0	1	0	0	1	1	ir	26H/27H✻

✻

ir		Mnemonic	Code
IX	0	AND A, [IX]	26H
IY	1	AND A, [IY]	27H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Register indirect
Dst: Register direct

Example

Set Value		Result				
A	[ir]	A	SC			
			N	V	C	Z
3BH	61H	21H	0	–	–	0
5AH	A5H	00H	0	–	–	1
D6H	93H	92H	1	–	–	0

AND A, [ir+dd] ||||||||| Logical AND of location [ir reg. + dd] and A reg. |||||||||||||||||||| 4 cycles |||

Function **A ← A ∧ [ir+dd]**

Takes a logical product of the content of the data memory and the content of the A register and stores the result in the A register. The data memory address has been specified by the sum of the content of the ir register (IX/IY) and the displacement dd.
The displacement dd is handled as signed data and the range is -128 to 127.
The content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	1	0	0	0	0	ir	20H/21H✻
			d	d				dd

✻

ir		Mnemonic	Code
IX	0	AND A, [IX+dd]	20H
IY	1	AND A, [IY+dd]	21H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Register indirect with displacement
Dst: Register direct

Example

Set Value		Result				
A	[ir+dd]	A	SC			
			N	V	C	Z
3BH	61H	21H	0	–	–	0
5AH	A5H	00H	0	–	–	1
D6H	93H	92H	1	–	–	0

AND A, [ir+L] ||||||||||| Logical AND location [ir reg. + L] and A reg. ||||||||||||||||||||||||| 4 cycles |||

Function **A ← A ∧ [ir+L]**

Takes a logical product of the content of the data memory and the content of the A register and stores the result in the A register. The data memory address has been specified by the sum of the content of the ir register (IX/IY) and the content of the L register.
The content of the L register is handled as signed data and the range is -128 to 127.
The content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	1	0	0	0	1	ir	22H/23H✻

✻

ir		Mnemonic	Code
IX	0	AND A, [IX+L]	22H
IY	1	AND A, [IY+L]	23H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Register indirect with index register
Dst: Register direct

Example

Set Value		Result				
A	[ir+L]	A	SC			
			N	V	C	Z
3BH	61H	21H	0	–	–	0
5AH	A5H	00H	0	–	–	1
D6H	93H	92H	1	–	–	0

AND B, #nn *Logical AND of immediate data nn and B reg.* *3 cycles*

Function **B ← B ∧ nn**

Takes a logical product of the 8-bit immediate data nn and the content of the B register and stores the result in the B register.

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	1	0	0	0	0	B0H
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result				
B	nn	B	SC			
			N	V	C	Z
3BH	61H	21H	0	–	–	0
5AH	A5H	00H	0	–	–	1
D6H	93H	92H	1	–	–	0

AND L, #nn *Logical AND of immediate data nn and L reg.* *3 cycles*

Function **L ← L ∧ nn**

Takes a logical product of the 8-bit immediate data nn and the content of the L register and stores the result in the L register.

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	1	0	0	0	1	B1H
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result				
L	nn	L	SC			
			N	V	C	Z
3BH	61H	21H	0	–	–	0
5AH	A5H	00H	0	–	–	1
D6H	93H	92H	1	–	–	0

AND H, #nn *Logical AND of immediate data nn and H reg.* *3 cycles*

Function **H ← H ∧ nn**

Takes a logical product of the 8-bit immediate data nn and the content of the H register and stores the result in the H register.

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	1	0	0	1	0	B2H
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result				
H	nn	H	SC			
			N	V	C	Z
3BH	61H	21H	0	–	–	0
5AH	A5H	00H	0	–	–	1
D6H	93H	92H	1	–	–	0

AND SC, #nn |||||||||||| *Logical AND of immediate data nn and SC* |||||||||||||||||||||||||||||||| *3 cycless* |||

Function **SC ← SC ∧ nn**

Takes a logical product of the 8-bit immediate data nn and the content of the system condition flag (SC) and sets the result in the system condition flag (SC).

Code

MSB							LSB	
1	0	0	1	1	1	0	0	9CH
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
↓	↓	↓	↓	↓	↓	↓	↓

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result							
SC	nn	SC							
		I1	I0	U	D	N	V	C	Z
3BH	61H	0	0	1	0	0	0	0	1
5AH	A5H	0	0	0	0	0	0	0	0
D6H	93H	1	0	0	1	0	0	1	0

AND [BR:ll], #nn ||| *Logical AND of immediate data nn and location [BR:ll]* |||||||||||| *5 cycles* |||

Function **[BR:*ll*] ← [BR:*ll*] ∧ nn**

Takes a logical product of the 8-bit immediate data and the content of the data memory and stores the result in that address. The data memory address has been specified by the content of the BR register (upper byte specification) and the 8-bit absolute address *ll* (lower byte specification).
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	1	1	0	0	0	D8H
			l	*l*				*ll*
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Immediate data
Dst: 8-bit absolute

Example

Set Value		Result				
[BR:*ll*]	nn	[BR:*ll*]	SC			
			N	V	C	Z
3BH	61H	21H	0	–	–	0
5AH	A5H	00H	0	–	–	1
D6H	93H	92H	1	–	–	0

AND [HL], A |||||||||||||| *Logical AND of A reg. and location [HL]* |||||||||||||||||||||||||||||||||| *4 cycles* |||

Function **[HL] ← [HL] ∧ A**

Takes a logical product of the content of the A register and the data memory that has been address specified by the HL register and stores the result in that address.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	1	0	0	1	0	0	24H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Register direct
Dst: Register indirect

Example

Set Value		Result				
[HL]	A	[HL]	SC			
			N	V	C	Z
3BH	61H	21H	0	–	–	0
5AH	A5H	00H	0	–	–	1
D6H	93H	92H	1	–	–	0

AND [HL], #nn ||||||||| *Logical AND of immediate data nn and location [HL]* |||||||||||||| *5 cycles* |||

Function **[HL] ← [HL] ∧ nn**

Takes a logical product of the 8-bit immediate data nn and the data memory that has been address specified by the HL register and stores the result in that address.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Mode Src: Immediate data
Dst: Register indirect

Example

Set Value		Result				
			SC			
[HL]	nn	[HL]	N	V	C	Z
3BH	61H	21H	0	–	–	0
5AH	A5H	00H	0	–	–	1
D6H	93H	92H	1	–	–	0

Code

MSB LSB

1	1	0	0	1	1	1	0	CEH
0	0	1	0	0	1	0	1	25H
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

AND [HL], [ir] |||||||||| *Logical AND of location [ir reg.] to location [HL]* |||||||||||||||||||| *5 cycles* |||

Function **[HL] ← [HL] ∧ [ir]**

Takes a logical product of the content of the data memory that has been address specified by the ir register (IX/IY) and the data memory that has been address specified by the HL register and stores the result in data memory [HL].
The content of the EP register becomes the page address of the data memory [HL] and the content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory [ir] (MODEL2/3).

Mode Src: Register indirect
Dst: Register indirect

Example

Set Value		Result				
			SC			
[HL]	[ir]	[HL]	N	V	C	Z
3BH	61H	21H	0	–	–	0
5AH	A5H	00H	0	–	–	1
D6H	93H	92H	1	–	–	0

Code

MSB LSB

1	1	0	0	1	1	1	0	CEH
0	0	1	0	0	1	1	ir	26H/27H✻

✻

ir		Mnemonic	Code
IX	0	AND [HL], [IX]	26H
IY	1	AND [HL], [IY]	27H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

BIT A, B *Test bit of A reg. with B reg.* *2 cycles*

Function **A ∧ B**

Takes a logical product of the content of the B register and the content of the A register and checks the bits of the A register.
The flags (N/Z) change depending on the said result, but the content of the register is not changed.

Code

MSB							LSB	
1	0	0	1	0	1	0	0	94H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Register direct
Dst: Register direct

Example

Set Value		Result				
A	B	A	SC			
			N	V	C	Z
3BH	61H	3BH	0	–	–	0
5AH	A5H	5AH	0	–	–	1
D6H	93H	D6H	1	–	–	0

BIT A, #nn *Test bit of A reg. with immediate data nn* *2 cycles*

Function **A ∧ nn**

Takes a logical product of the 8-bit immediate data nn and the content of the A register and checks the bits of the A register.
The flags (N/Z) change depending on the said result, but the content of the register is not changed.

Code

MSB							LSB	
1	0	0	1	0	1	1	0	96H
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result				
A	nn	A	SC			
			N	V	C	Z
3BH	61H	3BH	0	–	–	0
5AH	A5H	5AH	0	–	–	1
D6H	93H	D6H	1	–	–	0

BIT B, #nn *Test bit of B reg. with immediate data nn* *2 cycles*

Function **B ∧ nn**

Takes a logical product of the 8-bit immediate data nn and the content of the B register and checks the bits of the B register.
The flags (N/Z) change depending on the said result, but the content of the register is not changed.

Code

MSB							LSB	
1	0	0	1	0	1	1	1	97H
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result				
B	nn	B	SC			
			N	V	C	Z
3BH	61H	3BH	0	–	–	0
5AH	A5H	5AH	0	–	–	1
D6H	93H	D6H	1	–	–	0

BIT [BR:ll], #nn |||||| *Test bit of location [BR:ll] with immediate data nn* |||||||||||||||||| *4 cycles* |||

Function **[BR:*ll*] ∧ nn**

Takes a logical product of the 8-bit immediate data nn and the content of the data memory and checks the bits of the data memory. The data memory address has been specified by the content of the BR register (upper byte specification) and the 8-bit absolute address *ll* (lower byte specification). The flags (N/Z) change depending on the said result, but the content of the data memory is not changed.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Mode Src: Immediate data
Dst: 8-bit absolute

Example

Set Value		Result				
[BR:*ll*]	nn	[BR:*ll*]	SC			
			N	V	C	Z
3BH	61H	3BH	0	–	–	0
5AH	A5H	5AH	0	–	–	1
D6H	93H	D6H	1	–	–	0

Code

MSB							LSB	
1	1	0	1	1	1	0	0	DCH
			l	*l*				*ll*
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

BIT [HL], #nn ||||||||||| *Test bit of location [HL] with immediate data nn* |||||||||||||||||||||| *3 cycles* |||

Function **[HL] ∧ nn**

Takes a logical product of the 8-bit immediate data nn and the data memory that has been address specified by the HL register and checks the bits of the data memory. The flags (N/Z) change depending on the said result, but the content of the data memory is not changed.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Mode Src: Register direct
Dst: Register indirect

Example

Set Value		Result				
[HL]	nn	[HL]	SC			
			N	V	C	Z
3BH	61H	3BH	0	–	–	0
5AH	A5H	5AH	0	–	–	1
D6H	93H	D6H	1	–	–	0

Code

MSB							LSB	
1	0	0	1	0	1	0	1	95H
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

CALL [hhll] |||||||||||||| *Call subroutine at location [hhll]* ||||||||||||||||||||||||||| *7(MIN)/8(MAX) cycles* |||

Function

<MODEL0/1>

[SP-1]←PC(H), [SP-2]←PC(L),
SP←SP-2,
PC(L)←[hh*ll*], PC(H)←[hh*ll*+1]

<MODEL2/3, Minimum mode>

[SP-1]←PC(H), [SP-2]←PC(L),
SP←SP-2, PC(L)←[hh*ll*],
PC(H)←[hh*ll*+1], CB←NB

<MODEL2/3, Maximum mode>

[SP-1]←CB, [SP-2]←PC(H),
[SP-3]←PC(L), SP←SP-3,
PC(L)←[hh*ll*], PC(H)←[hh*ll*+1],
CB←NB

After evacuation of the top address +3 value of this instruction to the stack as a return address, it unconditionally calls the subroutine. As the branch destination address (top address of the subroutine), the content of the data memory specified by the 16-bit absolute address hh*ll* becomes the lower byte and the content of the following address becomes the upper byte.

In the maximum mode of the MODEL2/3, the currently selected bank address (content of the CB) is also evacuated upon evacuation of the return address.
Upon branching in the MODEL2/3, the bank address set in the NB is loaded into the CB and the bank is also changed.
The content of the EP register becomes the page address of the data memory.

Code

MSB							LSB	
1	1	1	1	1	0	1	1	FBH
			l	*l*				*ll*
			h	h				hh

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode 16-bit indirect

Example When NB = 02H in the MODEL2/3, it executes the "CALL [2000H]" instruction in the physical address 9000H.

	NB	CB	PC(logical addr.)	EP	M(032000H)	SP
Before execution	02H	01H	9000H	03H	ABCDH	0000H
After execution	02H	02H	ABCDH	03H	ABCDH	FFFDH

In the above example it branches to the physical address 012BCDH.
Since there is no EP in the MODEL0/1, the content of M(2000H) is transferred to the PC and if the content of M(2000H) is ABCDH, since there are also no NB and CB, it branches to the physical address ABCDH.

Stack content after execution

(1) MODEL2/3 (maximum mode)

00FFFDH	03H (PC(L))
00FFFEH	90H (PC(H))
00FFFFH	01H (CB)

(2) MODEL2/3 (minimum mode), MODEL0/1

00FFFEH	03H (PC(L))
00FFFFH	90H (PC(H))

CARL qqrr ||||||||||||||||| *Call subroutine at relative location qqrr* ||||||||||||||||| *5(MIN)/6(MAX) cycles* |||

Function

<MODEL0/1>

[SP-1]←PC(H), [SP-2]←PC(L), SP←SP-2, PC←PC+qqrr+2

<MODEL2/3, Minimum mode>

[SP-1]←PC(H), [SP-2]←PC(L), SP←SP-2, PC←PC+qqrr+2, CB←NB

<MODEL2/3, Maximum mode>

[SP-1]←CB, [SP-2]←PC(H), [SP-3]←PC(L), SP←SP-3, PC←PC+qqrr+2, CB←NB

After evacuation of the top address +3 value of this instruction to the stack as a return address, it unconditionally calls the subroutine. The branch destination address (top address of the subroutine) becomes the address resulting from the addition of a signed 16-bit relative address qqrr (-32768 to 32767) to the top address +2 of this instruction.

In the maximum mode of the MODEL2/3, the currently selected bank address (content of the CB) is also evacuated upon evacuation of the return address.
Upon branching in the MODEL2/3, the bank address set in the NB is loaded into the CB and the bank is also changed.

Code

MSB							LSB	
1	1	1	1	0	0	1	0	F2H
			r	r				rr
			q	q				qq

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Signed 16-bit PC relative

Example When NB = 02H in the MODEL2/3, it executes the "CARL $+2000H" instruction in the physical address 9000H.

	NB	CB	PC(logical addr.)	SP
Before execution	02H	01H	9000H	0000H
			9002H+(2000H-2)	
After execution	02H	02H	B000H	FFFDH

In the above example it branches to the physical address 013000H.
In the MODEL0/1, since there are no NB and CB, it branches to the physical address B000H.

Stack content after execution

(1) MODEL2/3 (maximum mode)

00FFFDH	03H (PC(L))
00FFFEH	90H (PC(H))
00FFFFH	01H (CB)

(2) MODEL2/3 (minimum mode), MODEL0/1

00FFFEH	03H (PC(L))
00FFFFH	90H (PC(H))

CARL cc1, qqrr ////// *Call subroutine at relative location qqrr if condition cc1 is true* /////////////////

Function

```
<MODEL0/1>
  If cc1 is true
    then CARL qqrr
    else PC←PC+3

<MODEL2/3>
  If cc1 is true
    then CARL qqrr
    else PC←PC+3, NB←CB
```

When the condition cc1 has been established, the CPU executes the "CARL qqrr" instruction and when a condition has not been established, it executes the following instruction.

See "CARL qqrr" instruction.

In the MODEL2/3, when a condition has not been established, the content of the NB specifying the branch destination bank returns to the current bank address (content of the CB).

Condition cc1 is of the below 4 types.

cc1	Condition	
C	Carry	(Carry flag C = 1)
NC	Non Carry	(Carry flag C = 0)
Z	Zero	(Zero flag Z = 1)
NZ	Non Zero	(Zero flag Z = 0)

The bus cycle becomes as follows, depending on whether it is minimum mode or maximum mode and whether a condition is established or not.

Mode	Condition	Bus cycle
Minimum	True	5 cycles
Minimum	False	3 cycles
Maximum	True	6 cycles
Maximum	False	3 cycles

Code

MSB						LSB	
1	1	1	0	0	0	cc1	E8H–EBH✻
			r	r			rr
			q	q			qq

✻

cc1		Mnemonic	Code
C	00	CARL C,qqrr	E8H
NC	01	CARL NC,qqrr	E9H
Z	10	CARL Z,qqrr	EAH
NZ	11	CARL NZ,qqrr	EBH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Signed 16-bit PC relative

Example At the time of condition establishment, operates the same as the "CARL qqrr" instruction. When a condition has not been established, the operation of the "CARL cc1,qqrr" in the physical address 9000H is as indicated below. The stack operation is not done.

	NB	CB	PC(logical addr.)	SP
Before execution	02H	01H	9000H	0000H
	↓			
After execution	01H	01H	9003H	0000H

There are no NB and CB in the MODEL0/1.

CARS rr *Call subroutine at relative location rr* *4(MIN)/5(MAX) cycles*

Function

<MODEL0/1>
[SP-1]←PC(H), [SP-2]←PC(L),
SP←SP-2, PC←PC+rr+1

<MODEL2/3, Minimum mode>
[SP-1]←PC(H), [SP-2]←PC(L),
SP←SP-2, PC←PC+rr+1, CB←NB

<MODEL2/3, Maximum mode>
[SP-1]←CB, [SP-2]←PC(H),
[SP-3]←PC(L), SP←SP-3,
PC←PC+rr+1, CB←NB

After evacuation of the top address +2 value of this instruction to the stack as a return address, it unconditionally calls the subroutine. The branch destination address (top address of the subroutine) becomes the address resulting from the addition of a signed 8-bit relative address rr (-128 to 127) to the top address +1 of this instruction.

In the maximum mode of the MODEL2/3, the currently selected bank address (content of the CB) is also evacuated upon evacuation of the return address.
Upon branching in the MODEL2/3, the bank address set in the NB is loaded into the CB and the bank is also changed.

Code

MSB							LSB	
1	1	1	1	0	0	0	0	F0H
			r	r				rr

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Signed 8-bit PC relative

Example When NB = 02H in the MODEL2/3, it executes the "CARS $+20H" instruction in the physical address 9000H.

	NB	CB	PC(logical addr.)	SP
Before execution	02H	01H	9000H	0000H
			9001H + (20H - 1)	
After execution	02H	02H	9020H	FFFDH

In the above example it branches to the physical address 011020H.
In the MODEL0/1, since there are no NB and CB, it branches to the physical address 9020H.

Stack content after execution

(1) MODEL2/3 (maximum mode)

00FFFDH	02H (PC(L))
00FFFEH	90H (PC(H))
00FFFFH	01H (CB)

(2) MODEL2/3 (minimum mode), MODEL0/1

00FFFEH	02H (PC(L))
00FFFFH	90H (PC(H))

CARS cc1, rr |||||||||||| *Call subroutine at relative location rr if condition cc1 is true* ||||||||||||||||||||||

Function <MODEL0/1>

```
If cc1 is true
  then CARS rr
  else PC←PC+2
```

<MODEL2/3>

```
If cc1 is true
  then CARS rr
  else PC←PC+2, NB←CB
```

When the condition cc1 has been established, the CPU executes the "CARS rr" instruction and when a condition has not been established, it executes the following instruction.

See "CARS rr" instruction.

In the MODEL2/3, when a condition has not been established, the content of the NB specifying the branch destination bank returns to the current bank address (content of the CB).

Condition cc1 is of the below 4 types.

cc1	Condition	
C	Carry	(Carry flag C = 1)
NC	Non Carry	(Carry flag C = 0)
Z	Zero	(Zero flag Z = 1)
NZ	Non Zero	(Zero flag Z = 0)

The bus cycle becomes as follows, depending on whether it is minimum mode or maximum mode and whether a condition is established or not.

Mode	Condition	Bus cycle
Minimum	True	4 cycles
Minimum	False	2 cycles
Maximum	True	5 cycles
Maximum	False	2 cycles

Code

MSB							LSB	
1	1	1	0	0	0	cc1		E0H–E3H✻
			r	r				rr

✻

cc1		Mnemonic	Code
C	00	CARS C, rr	E0H
NC	01	CARS NC, rr	E1H
Z	10	CARS Z, rr	E2H
NZ	11	CARS NZ, rr	E3H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Signed 8-bit PC relative

Example At the time of condition establishment, operates the same as the "CARS rr" instruction. When a condition has not been established, the operation of the "CARS cc1,rr" in the physical address 9000H is as indicated below. The stack operation is not done.

	NB	CB	PC(logical addr.)	SP
Before execution	02H	01H	9000H	0000H
After execution	01H	01H	9002H	0000H

There are no NB and CB in the MODEL0/1.

CARS cc2, rr |||||||||||| *Call subroutine at relative location rr if condition cc2 is true* |||||||||||||||||||||

Function

```
<MODEL0/1>
  If cc2 is true
    then  [SP-1]←PC(H),
          [SP-2]←PC(L), SP←SP-2,
          PC←PC+rr+2
    else  PC←PC+3

<MODEL2/3, Minimum mode>
  If cc2 is true
    then  [SP-1]←PC(H),
          [SP-2]←PC(L),
          SP←SP-2, PC←PC+rr+2
          CB←NB
    else  PC←PC+3, NB←CB

<MODEL2/3, Maximum mode>
  If cc2 is true
    then  [SP-1]←CB, [SP-2]←PC(H),
          [SP-3]←PC(L), SP←SP-3,
          PC←PC+rr+2, CB←NB
    else  PC←PC+3, NB←CB
```

When the condition cc2 has been established, after evacuation of the top address +3 value of this instruction to the stack as a return address, it calls the subroutine. The branch destination address (top address of the subroutine) becomes the address resulting from the addition of a signed 8-bit relative address rr (-128 to 127) to the top address +2 of this instruction.

When a condition has not been established, it executes the following instruction.

In the maximum mode of the MODEL2/3, the currently selected bank address (content of the CB) is also evacuated upon evacuation of the return address.
Upon branching in the MODEL2/3, the bank address set in the NB is loaded into the CB and the bank is also changed.
When a condition has not been established, the content of the NB specifying the branch destination bank returns to the current bank address (content of the CB).

Condition cc2 is of the below 16 types.

cc2	Condition	
LT	Less Than	([N ∀ V] = 1)
LE	Less or Equal	(Z ∨ [N ∀ N] = 1)
GT	Greater Than	(Z ∨ [N ∀ N] = 0)
GE	Greater or Equal	([N ∀ N] = 0)
V	Overflow	(V = 1)
NV	Non Overflow	(V = 0)
P	Plus	(N = 0)
M	Minus	(N = 1)
F0	F0 is set	(F0 = 1)
F1	F1 is set	(F1 = 1)
F2	F2 is set	(F2 = 1)
F3	F3 is set	(F3 = 1)
NF0	F0 is reset	(F0 = 0)
NF1	F1 is reset	(F1 = 0)
NF2	F2 is reset	(F2 = 0)
NF3	F3 is reset	(F3 = 0)

The bus cycle becomes as follows, depending on whether it is minimum mode or maximum mode and whether a condition is established or not.

Mode	Condition	Bus cycle
Minimum	True	5 cycles
Minimum	False	3 cycles
Maximum	True	6 cycles
Maximum	False	3 cycles

CARS cc2, rr

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	1	1	1	cc2				F0H–FFH✻
			r	r				rr

✻

cc2		Mnemonic	Code
LT	0000	CARS LT, rr	F0H
LE	0001	CARS LE, rr	F1H
GT	0010	CARS GT, rr	F2H
GE	0011	CARS GE, rr	F3H
V	0100	CARS V, rr	F4H
NV	0101	CARS NV, rr	F5H
P	0110	CARS P, rr	F6H
M	0111	CARS M, rr	F7H
F0	1000	CARS F0, rr	F8H
F1	1001	CARS F1, rr	F9H
F2	1010	CARS F2, rr	FAH
F3	1011	CARS F3, rr	FBH
NF0	1100	CARS NF0, rr	FCH
NF1	1101	CARS NF1, rr	FDH
NF2	1110	CARS NF2, rr	FEH
NF3	1111	CARS NF3, rr	FFH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Signed 8-bit PC relative

Example At the time of condition establishment when NB = 02H in the MODEL2/3, operation of the "CARS cc2,$+20" in the physical address 9000H is as indicated below.

	NB	CB	PC(logical addr.)	SP
Before execution	02H	01H	9000H	0000H
			9002H + (20H - 2)	
After execution	02H	02H	9020H	FFFDH

In the above example it branches to the physical address 011020H.
In the MODEL0/1, since there are no NB and CB, it branches to the physical address 9020H.

Stack content after execution

(1) MODEL2/3 (maximum mode)

00FFFDH	03H (PC(L))
00FFFEH	90H (PC(H))
00FFFFH	01H (CB)

(2) MODEL2/3 (minimum mode), MODEL0/1

00FFFEH	03H (PC(L))
00FFFFH	90H (PC(H))

When a condition has not been established, the operation of the "CARS cc2,$+20H" in the physical address 9000H is as indicated below. The stack operation is not done.

	NB	CB	PC(logical addr.)	SP
Before execution	02H	01H	9000H	0000H
After execution	01H	01H	9003H	0000H

There are no NB and CB in the MODEL0/1.

CP A, r *Compare r reg. with A reg.* *2 cycles*

Function **A - r**

This function subtracts the content of r register (A/B) from the content of the A register and changes the content of the flag (N/V/C/Z) according the result thereof. The content of the A register is not changed.

Code

MSB							LSB	
0	0	1	1	0	0	0	r	30H/31H✻

✻

r		Mnemonic	Code
A	0	CP A, A	30H
B	1	CP A, B	31H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Register direct
Dst: Register direct

Example

Set Value		Result				
A	B	A	SC			
			N	V	C	Z
74H	2AH	74H	0	0	0	0
1DH	1DH	1DH	0	0	0	1
3CH	59H	3CH	1	0	1	0
C3H	62H	C3H	0	1	0	0

CP A, #nn *Compare immediate data nn with A reg.* *2 cycles*

Function **A - nn**

This function subtracts the 8-bit immediate data nn from the content of the A register and changes the content of the flag (N/V/C/Z) according the result thereof. The content of the A register is not changed.

Code

MSB							LSB	
0	0	1	1	0	0	1	0	32H
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result				
A	nn	A	SC			
			N	V	C	Z
74H	2AH	74H	0	0	0	0
1DH	1DH	1DH	0	0	0	1
3CH	59H	3CH	1	0	1	0
C3H	62H	C3H	0	1	0	0

CP A, [BR:ll] *Compare location [BR:ll] with A reg.* *3 cycles*

Function **A - [BR:*ll*]**

This function subtracts the content of the data memory from the content of the A register and changes the content of the flag (N/V/C/Z) according the result thereof. The data memory address has been specified by the content of the BR register (upper byte specification) and the 8-bit absolute address *ll* (lower byte specification). The content of the A register is not changed.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
0	0	1	1	0	1	0	0	34H
			l	*l*				*ll*

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: 8-bit absolute
Dst: Register direct

Example

Set Value		Result				
A	[BR:*ll*]	A	SC			
			N	V	C	Z
74H	2AH	74H	0	0	0	0
1DH	1DH	1DH	0	0	0	1
3CH	59H	3CH	1	0	1	0
C3H	62H	C3H	0	1	0	0

CP A, [hh*ll*] *Compare location [hhll] with A reg.* *4 cycles*

Function **A - [hh*ll*]**

This function subtracts the content of the data memory that has been address specified by the 16-bit absolute address hh*ll* from the content of the A register and changes the content of the flag (N/V/C/Z) according the result thereof. The content of the A register is not changed.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
0	0	1	1	0	1	0	1	35H
			l	*l*				*ll*
			h	h				hh

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: 16-bit absolute
Dst: Register direct

Example

Set Value		Result				
A	[hh*ll*]	A	SC			
			N	V	C	Z
74H	2AH	74H	0	0	0	0
1DH	1DH	1DH	0	0	0	1
3CH	59H	3CH	1	0	1	0
C3H	62H	C3H	0	1	0	0

CP A, [HL] *Compare location [HL] with A reg.* *2 cycles*

Function **A - [HL]**

This function subtracts the content of the data memory that has been address specified by the HL register from the content of the A register and changes the content of the flag (N/V/C/Z) according the result thereof. The content of the A register is not changed.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
0	0	1	1	0	0	1	1	33H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Register indirect
Dst: Register direct

Example

Set Value		Result				
A	[HL]	A	SC			
			N	V	C	Z
74H	2AH	74H	0	0	0	0
1DH	1DH	1DH	0	0	0	1
3CH	59H	3CH	1	0	1	0
C3H	62H	C3H	0	1	0	0

CP A, [ir] |||||||||||||||||||||| *Compare location [ir reg.] with A reg.* |||||||||||||||||||||||||||||||||||||| *2 cycles* |||

Function **A - [ir]**

This function subtracts the content of the data memory that has been address specified by the ir register (IX/IY) from the content of the A register and changes the content of the flag (N/V/C/Z) according the result thereof. The content of the A register is not changed.
The content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory (MODEL2/3).

Mode Src: Register indirect
Dst: Register direct

Example

Set Value		Result				
A	[ir]	A	SC			
			N	V	C	Z
74H	2AH	74H	0	0	0	0
1DH	1DH	1DH	0	0	0	1
3CH	59H	3CH	1	0	1	0
C3H	62H	C3H	0	1	0	0

Code

MSB / LSB

0	0	1	1	0	1	1	ir	36H/37H✻

✻

ir		Mnemonic	Code
IX	0	CP A, [IX]	36H
IY	1	CP A, [IY]	37H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

CP A, [ir+dd] ||||||||||||| *Compare location [ir reg. + dd] with A reg.* ||||||||||||||||||||||||||||| *4 cycles* |||

Function **A - [ir+dd]**

This function subtracts the content of the data memory from the content of the A register and changes the content of the flag (N/V/C/Z) according the result thereof. The data memory address has been specified by the sum of the content of the ir register (IX/IY) and the displacement dd.
The content of the A register is not changed.
The displacement dd is handled as signed data and the range is -128 to 127.
The content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory (MODEL2/3).

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Register indirect with displacement
Dst: Register direct

Example

Set Value		Result				
A	[ir+dd]	A	SC			
			N	V	C	Z
74H	2AH	74H	0	0	0	0
1DH	1DH	1DH	0	0	0	1
3CH	59H	3CH	1	0	1	0
C3H	62H	C3H	0	1	0	0

Code

MSB / LSB

1	1	0	0	1	1	1	0	CEH
0	0	1	1	0	0	0	ir	30H/31H✻
			d	d				dd

✻

ir		Mnemonic	Code
IX	0	CP A, [IX+dd]	30H
IY	1	CP A, [IY+dd]	31H

CP A, [ir+L] |||||||||||||||| *Compare location [ir reg. + L] with A reg.* |||||||||||||||||||||||||||||||| *4 cycles* |||

Function **A - [ir+L]**

This function subtracts the content of the data memory from the content of the A register and changes the content of the flag (N/V/C/Z) according the result thereof. The data memory address has been specified by the sum of the content of the ir register (IX/IY) and the content of the L register.
The content of the A register is not changed.
The content of the L register is handled as signed data and the range is -128 to 127.
The content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory (MODEL2/3).

Code

MSB / LSB

1	1	0	0	1	1	1	0	CEH
0	0	1	1	0	0	1	ir	32H/33H✻

✻

ir		Mnemonic	Code
IX	0	CP A, [IX+L]	32H
IY	1	CP A, [IY+L]	33H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Register indirect with index register
Dst: Register direct

Example

Set Value		Result				
A	[ir+L]	A	SC			
			N	V	C	Z
74H	2AH	74H	0	0	0	0
1DH	1DH	1DH	0	0	0	1
3CH	59H	3CH	1	0	1	0
C3H	62H	C3H	0	1	0	0

CP B, #nn |||||||||||||||||||| *Compare immediate data nn with B reg.* |||||||||||||||||||||||||||||||||||| *3 cycles* |||

Function **B - nn**

This function subtracts the 8-bit immediate data nn from the content of the B register and changes the content of the flag (N/V/C/Z) according the result thereof. The content of the B register is not changed.

Code

MSB / LSB

1	1	0	0	1	1	1	0	CEH
1	0	1	1	1	1	0	0	BCH
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result				
B	nn	B	SC			
			N	V	C	Z
74H	2AH	74H	0	0	0	0
1DH	1DH	1DH	0	0	0	1
3CH	59H	3CH	1	0	1	0
C3H	62H	C3H	0	1	0	0

CP L, #nn *Compare immediate data nn with L reg.* *3 cycles*

Function **L - nn**

This function subtracts the 8-bit immediate data nn from the content of the L register and changes the content of the flag (N/V/C/Z) according the result thereof. The content of the L register is not changed.

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	1	1	1	0	1	BDH
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result				
			SC			
L	nn	L	N	V	C	Z
74H	2AH	74H	0	0	0	0
1DH	1DH	1DH	0	0	0	1
3CH	59H	3CH	1	0	1	0
C3H	62H	C3H	0	1	0	0

CP H, #nn *Compare immediate data nn with H reg.* *3 cycles*

Function **H - nn**

This function subtracts the 8-bit immediate data nn from the content of the H register and changes the content of the flag (N/V/C/Z) according the result thereof. The content of the H register is not changed.

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	1	1	1	1	0	BEH
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result				
			SC			
H	nn	H	N	V	C	Z
74H	2AH	74H	0	0	0	0
1DH	1DH	1DH	0	0	0	1
3CH	59H	3CH	1	0	1	0
C3H	62H	C3H	0	1	0	0

CP BR, #hh *Compare immediate data hh with BR reg.* *3 cycles*

Function **BR - hh**

This function subtracts the 8-bit immediate data nn from the content of the BR register and changes the content of the flag (N/V/C/Z) according the result thereof. The content of the BR register is not changed.

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	1	1	1	1	1	BFH
			h	h				hh

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result				
			SC			
BR	hh	BR	N	V	C	Z
74H	2AH	74H	0	0	0	0
1DH	1DH	1DH	0	0	0	1
3CH	59H	3CH	1	0	1	0
C3H	62H	C3H	0	1	0	0

CP [BR:*ll*], #nn ||||||| *Compare immediate data nn with location [BR:ll]* |||||||||||||||||||| *4 cycles* |||

Function **[BR:*ll*] - nn**

This function subtracts the 8-bit immediate data nn from the content of the data memory and changes the content of the flag (N/V/C/Z) according the result thereof. The data memory address has been specified by the content of the BR register (upper byte specification) and the 8-bit absolute address *ll* (lower byte specification). The content of the data memory is not changed.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	1	1	0	1	1	DBH
			l	*l*				*ll*
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Immediate data
Dst: 8-bit absolute

Example

Set Value		Result				
[BR:*ll*]	nn	[BR:*ll*]	SC			
			N	V	C	Z
74H	2AH	74H	0	0	0	0
1DH	1DH	1DH	0	0	0	1
3CH	59H	3CH	1	0	1	0
C3H	62H	C3H	0	1	0	0

CP [HL], A |||||||||||||||||| *Compare A reg. with location [HL]* |||||||||||||||||||||||||||||||||||||||||| *3 cycles* |||

Function **[HL] - A**

This function subtracts the content of the A register from the content of the data memory that has been address specified by the HL register and changes the content of the flag (N/V/C/Z) according the result thereof. The content of the data memory is not changed.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	1	1	0	1	0	0	34H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Register direct
Dst: Register indirect

Example

Set Value		Result				
[HL]	A	[HL]	SC			
			N	V	C	Z
74H	2AH	74H	0	0	0	0
1DH	1DH	1DH	0	0	0	1
3CH	59H	3CH	1	0	1	0
C3H	62H	C3H	0	1	0	0

CP [HL], #nn |||||||||||| *Compare immediate data nn with location [HL]* |||||||||||||||||||||||| *4 cycles* |||

Function **[HL] - nn**

This function subtracts the 8-bit immediate data nn from the content of the data memory that has been address specified by the HL register and changes the content of the flag (N/V/C/Z) according the result thereof. The content of the data memory is not changed. The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	1	1	0	1	0	1	35H
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Immediate data
Dst: Register indirect

Example

Set Value		Result				
[HL]	nn	[HL]	SC			
			N	V	C	Z
74H	2AH	74H	0	0	0	0
1DH	1DH	1DH	0	0	0	1
3CH	59H	3CH	1	0	1	0
C3H	62H	C3H	0	1	0	0

CP [HL], [ir] |||||||||||||| *Compare location [ir reg.] with location [HL]* |||||||||||||||||||||||||| *4 cycles* |||

Function **[HL] - [ir]**

This function subtracts the content of the data memory that has been address specified by the ir register (IX/IY) from the content of the data memory that has been address specified by the HL register and changes the content of the flag (N/V/C/Z) according the result thereof. The content of the data memory is not changed.
The content of the EP register becomes the page address of the data memory [HL] and the content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory [ir] (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	1	1	0	1	1	ir	36H/37H✻

✻

ir		Mnemonic	Code
IX	0	CP [HL], [IX]	36H
IY	1	CP [HL], [IY]	37H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Register indirect
Dst: Register indirect

Example

Set Value		Result				
[HL]	[ir]	[HL]	SC			
			N	V	C	Z
74H	2AH	74H	0	0	0	0
1DH	1DH	1DH	0	0	0	1
3CH	59H	3CH	1	0	1	0
C3H	62H	C3H	0	1	0	0

CP BA, rp *Comparerp reg. with BA reg.* *4 cycles*

Function **BA - rp**

This function subtracts the content of rp register (BA/HL/IX/IY) from the content of the BA register and changes the content of the flag (N/V/C/Z) according the result thereof. The content of the BA register is not changed.

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	0	0	1	1	0	rp		18H–1BH✻

✻

rp		Mnemonic	Code
BA	00	CP BA, BA	18H
HL	01	CP BA, HL	19H
IX	10	CP BA, IX	1AH
IY	11	CP BA, IY	1BH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Register direct
Dst: Register direct

Example

Set Value		Result				
BA	rp	BA	SC			
			N	V	C	Z
3F71H	145AH	3F71H	0	0	0	0
53D1H	53D1H	53D1H	0	0	0	1
A291H	632EH	A291H	0	1	0	0
2862H	4C25H	2862H	1	0	1	0

(rp≠BA)

CP BA, #mmnn *Compare immediate data mmnn with BA reg.* *3 cycles*

Function **BA - mmnn**

This function subtracts the 16-bit immediate data mmnn from the content of the BA register and changes the content of the flag (N/V/C/Z) according the result thereof. The content of the BA register is not changed.

Code

MSB							LSB	
1	1	0	1	0	1	0	0	D4H
			n	n				nn
			m	m				mm

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result				
BA	mmnn	BA	SC			
			N	V	C	Z
3F71H	145AH	3F71H	0	0	0	0
53D1H	53D1H	53D1H	0	0	0	1
A291H	632EH	A291H	0	1	0	0
2862H	4C25H	2862H	1	0	1	0

CP HL, rp *Comparerp reg. with HL reg. 4 cycles*

Function **HL - rp**

This function subtracts the content of rp register (BA/HL/IX/IY) from the content of the HL register and changes the content of the flag (N/V/C/Z) according the result thereof. The content of the HL register is not changed.

Code

MSB LSB

1	1	0	0	1	1	1	1	CFH
0	0	1	1	1	0	rp		38H–3BH✻

✻

rp		Mnemonic	Code
BA	00	CP HL, BA	38H
HL	01	CP HL, HL	39H
IX	10	CP HL, IX	3AH
IY	11	CP HL, IY	3BH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Register direct
Dst: Register direct

Example

Set Value		Result				
HL	rp	HL	SC			
			N	V	C	Z
3F71H	145AH	3F71H	0	0	0	0
53D1H	53D1H	53D1H	0	0	0	1
A291H	632EH	A291H	0	1	0	0
2862H	4C25H	2862H	1	0	1	0

(rp≠HL)

CP HL, #mmnn *Compare immediate data mmnn with HL reg. 3 cycles*

Function **HL - mmnn**

This function subtracts the 16-bit immediate data mmnn from the content of the HL register and changes the content of the flag (N/V/C/Z) according the result thereof. The content of the HL register is not changed.

Code

MSB LSB

1	1	0	1	0	1	0	1	D5H
			n	n				nn
			m	m				mm

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result				
HL	mmnn	HL	SC			
			N	V	C	Z
3F71H	145AH	3F71H	0	0	0	0
53D1H	53D1H	53D1H	0	0	0	1
A291H	632EH	A291H	0	1	0	0
2862H	4C25H	2862H	1	0	1	0

CP IX, #mmnn *Compare immediate data mmnn with IX reg.* *3 cycles*

Function **IX - mmnn**

This function subtracts the 16-bit immediate data mmnn from the content of the IX register and changes the content of the flag (N/V/C/Z) according the result thereof. The content of the IX register is not changed.

Code

MSB							LSB	
1	1	0	1	0	1	1	0	D6H
			n	n				nn
			m	m				mm

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result				
IX	mmnn	IX	SC			
			N	V	C	Z
3F71H	145AH	3F71H	0	0	0	0
53D1H	53D1H	53D1H	0	0	0	1
A291H	632EH	A291H	0	1	0	0
2862H	4C25H	2862H	1	0	1	0

CP IY, #mmnn *Compare immediate data mmnn with IY reg.* *3 cycles*

Function **IY - mmnn**

This function subtracts the 16-bit immediate data mmnn from the content of the IY register and changes the content of the flag (N/V/C/Z) according the result thereof. The content of the IY register is not changed.

Code

MSB							LSB	
1	1	0	1	0	1	1	1	D7H
			n	n				nn
			m	m				mm

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result				
IY	mmnn	IY	SC			
			N	V	C	Z
3F71H	145AH	3F71H	0	0	0	0
53D1H	53D1H	53D1H	0	0	0	1
A291H	632EH	A291H	0	1	0	0
2862H	4C25H	2862H	1	0	1	0

CP SP, rp |||||||||||||||||||| Comparerp reg. with SP |||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 4 cycles |||

Function **SP - rp**

This function subtracts the content of rp register (BA/HL) from the content of the stack pointer (SP) and changes the content of the flag (N/V/C/Z) according the result thereof. The content of the SP is not changed.

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	1	0	1	1	1	0	rp	5CH/5DH✻

✻

rp		Mnemonic	Code
BA	0	CP SP, BA	5CH
HL	1	CP SP, HL	5DH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Register direct
Dst: Register direct

Example

Set Value		Result				
SP	rp	SP	SC			
			N	V	C	Z
3F71H	145AH	3F71H	0	0	0	0
53D1H	53D1H	53D1H	0	0	0	1
A291H	632EH	A291H	0	1	0	0
2862H	4C25H	2862H	1	0	1	0

CP SP, #mmnn |||||||| Compare immediate data mmnn with SP |||||||||||||||||||||||||||||||||||| 4 cycles |||

Function **SP - mmnn**

This function subtracts the 16-bit immediate data mmnn from the content of the stack pointer (SP) and changes the content of the flag (N/V/C/Z) according the result thereof. The content of the SP is not changed.

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	1	1	0	1	1	0	0	6CH
			n	n				nn
			m	m				mm

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result				
SP	mmnn	SP	SC			
			N	V	C	Z
3F71H	145AH	3F71H	0	0	0	0
53D1H	53D1H	53D1H	0	0	0	1
A291H	632EH	A291H	0	1	0	0
2862H	4C25H	2862H	1	0	1	0

CPL r *Complement r reg.* *3 cycles*

Function **$r \leftarrow \overline{r}$**
Inverts the each bit of the r register (A/B) and creates one compliment.

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	0	0	0	0	r	A0H/A1H✻

✻

r		Mnemonic	Code
A	0	CPL A	A0H
B	1	CPL A	A1H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Register direct

Example

Set Value	Result					
r	r	SC				
		N	V	C	Z
11111111	00000000	0	–	–	1
10100101	01011010	1	–	–	0

CPL [BR:ll] *Complement location [BR:ll]* *5 cycles*

Function **[BR:*ll*] $\leftarrow$ $\overline{\text{[BR:}ll\text{]}}$**
Inverts the each bit of the data memory and creates one compliment. The data memory address has been specified by the content of the BR register (upper byte specification) and the 8-bit absolute address *ll* (lower byte specification).
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	0	0	0	1	0	A2H
			l	*l*				*ll*

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode 8-bit absolute

Example

Set Value	Result				
[BR:*ll*]	[BR:*ll*]	SC			
		N	V	C	Z
11111111	00000000	0	–	–	1
10100101	01011010	1	–	–	0

CPL [HL] *Complement location [HL]* *4 cycles*

Function **[HL] $\leftarrow$ $\overline{\text{[HL]}}$**
Inverts the each bit of the data memory that has been address specified by the HL register and creates one compliment.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	0	0	0	1	1	A3H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Register indirect

Example

Set Value	Result				
[HL]	[HL]	SC			
		N	V	C	Z
11111111	00000000	0	–	–	1
10100101	01011010	1	–	–	0

DEC r ||||||||||||||||||||||||||||||| *Decrement r reg.* |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *2 cycles* |||

Function **r ← r - 1**
Decrements (-1) the content of the r register (A/B/L/H).

Mode Register direct

Code

MSB							LSB	
1	0	0	0	1	0	r		88H–8BH✻

✻

r		Mnemonic	Code
A	00	DEC A	88H
B	01	DEC B	89H
L	10	DEC L	8AH
H	11	DEC H	8BH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	↕

Example

Set Value	Result				
r	r	SC			
		N	V	C	Z
63H	62H	–	–	–	0
01H	00H	–	–	–	1

DEC BR ||||||||||||||||||||||||| *Decrement BR reg.* |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *2 cycles* |||

Function **BR ← BR - 1**
Decrements (-1) the content of the BR register.

Mode Register direct

Code

MSB							LSB	
1	0	0	0	1	1	0	0	8CH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	↕

Example

Set Value	Result				
BR	BR	SC			
		N	V	C	Z
63H	62H	–	–	–	0
01H	00H	–	–	–	1

DEC [BR:*ll*] |||||||||||||||| *Decrement location [BR:ll]* |||||||||||||||||||||||||||||||||||||||||||||||||||||| *4 cycles* |||

Function **[BR:*ll*] ← [BR:*ll*] - 1**
Decrements (-1) the content of the data memory. The data memory address has been specified by the content of the BR register (upper byte specification) and the 8-bit absolute address *ll* (lower byte specification). The content of the EP register becomes the page address of the data memory (MODEL2/3).

Mode 8-bit absolute

Code

MSB							LSB	
1	0	0	0	1	1	0	1	8DH
			l	*l*				*ll*

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	↕

Example

Set Value	Result				
[BR:*ll*]	[BR:*ll*]	SC			
		N	V	C	Z
63H	62H	–	–	–	0
01H	00H	–	–	–	1

DEC [HL] |||||||||||||||||||||| *Decrement location [HL]* |||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *3 cycles* |||

Function **[HL] ← [HL] - 1**

Decrements (-1) the content of the data memory that has been address specified by the HL register.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	0	0	0	1	1	1	0	8EH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	↕

Mode Register indirect

Example

Set Value	Result				
[HL]	[HL]	SC			
		N	V	C	Z
63H	62H	–	–	–	0
01H	00H	–	–	–	1

DEC rp |||||||||||||||||||||||||| *Decrement rp reg.* |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *2 cycles* |||

Function **rp ← rp - 1**

Decrements (-1) the content of the rp register (BA/HL/IX/IY).

Code

MSB						LSB	
1	0	0	1	1	0	rp	98H–9BH✻

✻

r		Mnemonic	Code
BA	00	DEC BA	98H
HL	01	DEC HL	99H
IX	10	DEC IX	9AH
IY	11	DEC IY	9BH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	↕

Mode Register direct

Example

Set Value	Result				
rp	rp	SC			
		N	V	C	Z
4285H	4284H	–	–	–	0
0001H	0000H	–	–	–	1

DEC SP ||||||||||||||||||||||||| *Decrement SP reg.* ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *2 cycles* |||

Function **SP ← SP - 1**

Decrements (-1) the content of the stack pointer (SP).

Code

MSB							LSB	
1	0	0	0	1	1	1	1	8FH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	↕

Mode Register direct

Example

Set Value	Result				
SP	SP	SC			
		N	V	C	Z
4285H	4284H	–	–	–	0
0001H	0000H	–	–	–	1

DIV *Divide HL reg. by A reg.* *13 cycles*

Function **L ← HL/A, H ← remainder**

Divides the content of the HL register by the content of the A register it stores the quotient in the L register and the remainder in the H register.
When it divides by the divisor '0', a zero division exception processing is generated.
When it divides by a divisor other than '0' and the quotient exceeds 8 bits, the V flag is set to '1' and the dividend (content of the HL register) is saved.

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	1	0	1	1	0	0	1	D9H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	0	↕

Mode Implied (Register direct)

Example

Set Value		Result					
HL	A	L	H	SC			
				N	V	C	Z
1A16H	64H	42H	4EH	0	0	0	0
332CH	64H	83H	00H	1	0	0	0
0000H	58H	00H	00H	0	0	0	1
0301H	02H	01H	03H	1	1	0	0

Note This instruction cannot be used in the MODEL0/2.

DJR NZ, rr *Decrement B reg. & Jump relative if B reg. is not zero* *4 cycles*

Function

```
<MODEL0/1>
B ← B - 1,
If B ≠ 0  then  JRS rr
          else  PC ← PC + 2

<MODEL2/3>
B ← B - 1,
If B ≠ 0  then  JRS rr
          else  PC ← PC + 2, NB ← CB
```

Decrements (-1) the B register and as a result thereof, the B register is other than '0', it executes the branch instruction "JRS rr". When the B register has become '0', it executes the following instruction.
See the "JRS rr" instruction.

In the MODEL2/3, the bank address set in the NB upon branching is loaded into the CB, and the bank is also changed. When it shifts to the following instruction instead of branching, this function returns the content of the NB to the current bank address (content of the CB).

Code

MSB							LSB	
1	1	1	1	0	1	0	1	F5H
			r	r				rr

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	↕

Mode Signed 8-bit PC relative

Example At the time of condition establishment when NB = 02H and B = 05H in the MODEL2/3, operation of the "DJR NZ,$-05H" in the physical address 9000H is as indicated below.

	NB	CB	PC(logical addr.)	B
Before execution	02H	01H	9000H	05H
			9001H+(FFFBH-1)	
After execution	02H	02H	8FFBH	04H

In the above example it branches to the physical address 010FFBH.
In the MODEL0/1, since there are no NB and CB, it branches to the physical address 8FFBH.

When a condition has not been established, the operation of the "DJR NZ,$-05H" in the physical address 9000H is as indicated below.

	NB	CB	PC(logical addr.)	B
Before execution	01H	02H	9000H	01H
After execution	02H	02H	9002H	00H

There are no NB and CB in the MODEL0/1.

EX A, B ||||||||||||||||||||||||| *Exchange A reg. and B reg.* ||||||||||||||||||||||||||||||||||||||||||||||||||||| *2 cycles* |||

Function **A ↔ B**
Exchanges the content of the B register with the A register.

Code

MSB							LSB	
1	1	0	0	1	1	0	0	CCH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register direct
Dst: Register direct

Example

Set Value		Result					
A	B	A	B	SC			
				N	V	C	Z
82H	49H	49H	82H	–	–	–	–

EX A, [HL] ||||||||||||||||||| *Exchange A reg. and location [HL]* ||||||||||||||||||||||||||||||||||||||||||| *3 cycles* |||

Function **A ↔ [HL]**
Exchanges the content of the data memory that has been address specified by HL register with the A register.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	0	1	CDH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register indirect
Dst: Register direct

Example

Set Value		Result					
A	[HL]	A	[HL]	SC			
				N	V	C	Z
82H	49H	49H	82H	–	–	–	–

EX BA, rp |||||||||||||||||||| *Exchange BA reg. and rp reg.* ||||||||||||||||||||||||||||||||||||||||||||||||| *3 cycles* |||

Function **BA ↔ rp**
Exchanges the content of the rp register (HL/IX/IY/SP) with the BA register.

Code

MSB							LSB	
1	1	0	0	1	0	rp		C8H–CBH✻

✻

rp		Mnemonic	Code
HL	00	EX BA, HL	C8H
IX	01	EX BA, IX	C9H
IY	10	EX BA, IY	CAH
SP	11	EX BA,SP	CBH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register direct
Dst: Register direct

Example

Set Value		Result					
BA	rp	BA	rp	SC			
				N	V	C	Z
35D6H	C284H	C284H	35D6H	–	–	–	–

HALT ||||||||||||||||||||||||||||||||| *Set CPU to HALT mode* ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *3 cycles* |||

Function **HALT**

Sets the CPU in the HALT status.
In the HALT status, the CPU stops operation, thus reducing power consumption. Peripheral circuits such as the oscillation circuit still operate.
An interrupt causes it to return from the HALT status to the normal program execution status.

See Section 3.7.1, "Halt status".

Code

MSB … LSB

1	1	0	0	1	1	1	0	CEH
1	0	1	0	1	1	1	0	AEH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

INC r |||||||||||||||||||||||||||||||| *Increment r reg.* ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *2 cycles* |||

Function **r ← r + 1**

Increments (+1) the content of the r register (A/B/L/H).

Code

MSB … LSB

1	0	0	0	0	0	r	80H–83H✻

✻

r		Mnemonic	Code
A	00	INC A	80H
B	01	INC B	81H
L	10	INC L	82H
H	11	INC H	83H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	↕

Mode Register direct

Example

Set Value	Result	SC			
r	r	N	V	C	Z
52H	53H	–	–	–	0
FFH	00H	–	–	–	1

INC BR ||||||||||||||||||||||||||| *Increment BR reg.* |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *2 cycles* |||

Function **BR ← BR + 1**

Increments (+1) the content of the BR register.

Code

MSB … LSB

1	0	0	0	0	1	0	0	84H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	↕

Mode Register direct

Example

Set Value	Result	SC			
BR	BR	N	V	C	Z
52H	53H	–	–	–	0
FFH	00H	–	–	–	1

INC [BR:ll] ||||||||||||||||| *Increment location [BR:ll]* ||||||||||||||||||||||||||||||||||||||||||||||||||||| *4 cycles* |||

Function **[BR:*ll*] ← [BR:*ll*] +1**

Increments (+1) the content of the data memory. The data memory address has been specified by the content of the BR register (upper byte specification) and the 8-bit absolute address *ll* (lower byte specification).
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Mode 8-bit absolute

Example

Set Value	Result				
[BR:*ll*]	[BR:*ll*]	SC			
		N	V	C	Z
52H	53H	–	–	–	0
FFH	00H	–	–	–	1

Code MSB LSB

1	0	0	0	0	1	0	1	85H
			l	*l*				*ll*

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	↕

INC [HL] ||||||||||||||||||||||| *Increment location [HL]* |||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *3 cycles* |||

Function **[HL] ← [HL] + 1**

Increments (+1) the content of the data memory that has been address specified by the HL register.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Mode Register indirect

Example

Set Value	Result				
[HL]	[HL]	SC			
		N	V	C	Z
52H	53H	–	–	–	0
FFH	00H	–	–	–	1

Code MSB LSB

1	0	0	0	0	1	1	0	86H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	↕

INC rp |||||||||||||||||||||||||||||| *Increment rp reg.* ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *2 cycles* |||

Function **rp ← rp + 1**

Increments (+1) the content of the rp register (BA/HL/IX/IY).

Mode Register direct

Example

Set Value	Result				
rp	rp	SC			
		N	V	C	Z
3259H	325AH	–	–	–	0
FFFFH	0000H	–	–	–	1

Code MSB LSB

1	0	0	1	0	0	rp	90H–93H✻

✻

r		Mnemonic	Code
BA	00	INC BA	90H
HL	01	INC HL	91H
IX	10	INC IX	92H
IY	11	INC IY	93H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	↕

INC SP |||||||||||||||||||||||||||||| *Increment SP reg.* |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *2 cycles* |||

Function **SP ← SP +1**

Increments (+1) the content of the stack pointer (SP).

Code

MSB							LSB	
1	0	0	0	0	1	1	1	87H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	↕

Mode Register direct

Example

Set Value	Result				
SP	SP	SC			
		N	V	C	Z
3259H	325AH	–	–	–	0
FFFFH	0000H	–	–	–	1

INT [kk] |||||||||||||||||||||||||||| *Software Interrupt* |||||||||||||||||||||||||||||||||||||||||||||||||||| *7(MIN)/8(MAX) cycles* |||

Function

<MODEL0/1>

[SP-1]←PC(H), [SP-2]←PC(L),
[SP-3]←SC, SP←SP-3,
PC(L)←[00kk], PC(H)←[00kk+1]

<MODEL2/3, Minimum mode>

[SP-1]←PC(H), [SP-2]←PC(L),
[SP-3]←SC, SP←SP-3, PC(L)←[00kk],
PC(H)←[00kk+1], CB←NB

<MODEL2/3, Maximum mode>

[SP-1]←CB, [SP-2]←PC(H),
[SP-3]←PC(L), [SP-4]←SC,
SP←SP-4, PC(L)←[00kk],
PC(H)←[00kk+1], CB←NB

Executes the software interrupt routine that makes the 00kk address of the program memory the vector address, following evacuation of the top address +2 value of this instruction and system condition flag (SC) to the stack.

In the maximum mode of the MODEL2/3, the currently selected bank address (content of the CB) is also evacuated upon evacuation of the return address.
Upon branching in the MODEL2/3, the bank address set in the NB is loaded into the CB and the bank is also changed.
The content of the EP register becomes the page address of the data memory.

The vector field is fixed at page 0.

Note You should use the "RETE" instruction that also returns the content of the SC for return from an interrupt routine executed by an "INT [kk]" instruction.

Code

MSB							LSB	
1	1	1	1	1	1	0	0	FCH
			k	k				kk

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode 8-bit indirect

Example When NB = 02H in the MODEL2/3, it executes the "INT [20H]" instruction in the physical address 9000H.

	NB	CB	PC(logical addr.)	EP	M(000020H)	SP
Before execution	02H	01H	9000H	03H	ABCDH	0000H
		↓	↓			
After execution	02H	02H	ABCDH	03H	ABCDH	FFFCH

EP is disregarded and the vector area (000000H–0000FFH) is specified.
In the above example it branches to the physical address 012BCDH.
Since there are no NB and CB in the MODEL0/1, it branches to the physical address ABCDH.

Stack content after execution

(1) MODEL2/3 (maximum mode)

00FFFCH	Content of SC
00FFFDH	02H (PC(L))
00FFFEH	90H (PC(H))
00FFFFH	01H (CB)

(2) MODEL2/3 (minimum mode), MODEL0/1

00FFFDH	Content of SC
00FFFEH	02H (PC(L))
00FFFFH	90H (PC(H))

JP HL *Indirect Jump using HL reg.* *2 cycles*

Function **<MODEL0/1>**

PC ← HL

<MODEL2/3>

PC ← HL, CB ← NB

Loads the content of the HL register into the program counter (PC) then unconditionally branches.

Upon branching in the MODEL2/3, the bank address set in the NB is loaded into the CB and the bank is also changed.

Code

MSB							LSB	
1	1	1	1	0	1	0	0	F4H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Register direct

Example When NB = 02H and HL = 8765H in the MODEL2/3, it executes the "JP HL" instruction in the physical address 9000H.

	NB	CB	PC(logical addr.)	HL
Before execution	02H	01H	9000H	8765H
After execution	02H	02H	8765H	8765H

In the above example it branches to the physical address 010765H.
Since there are no NB and CB in the MODEL0/1, it branches to the physical address 8765H.

JP [kk] *Indirect Jump using vector* *4 cycles*

Function **<MODEL0/1>**

PC(L) ← [00kk], PC(H) ← [00kk+1]

<MODEL2/3>

CB ← NB, PC(L) ← [00kk],
PC(H) ← [00kk+1]

It makes the kk the 8-bit indirect address, then load the vector written into the 00kk and 00kk+1 address of the program memory into the program counter (PC), and then unconditionally branches.

Upon branching in the MODEL2/3, the bank address set in the NB is loaded into the CB and the bank is also changed.

The vector field is fixed at page 0.

Code

MSB							LSB	
1	1	1	1	1	1	0	1	FDH
			k	k				kk

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode 8-bit indirect

Example When NB = 02H in the MODEL2/3, it executes the "JP [20H]" instruction in the physical address 9000H.

	NB	CB	PC(logical addr.)	EP	M(000020H)
Before execution	02H	01H	9000H	03H	ABCDH
After execution	02H	02H	ABCDH	03H	ABCDH

EP is disregarded and the vector area (000000H–0000FFH) is specified.
In the above example it branches to the physical address 012BCDH.
Since there are no NB and CB in the MODEL0/1, it branches to the physical address ABCDH.

JRL qqrr *Jump to relative location qqrr* *3 cycles*

Function **<MODEL0/1>**

PC ← PC + qqrr + 2

<MODEL2/3>

PC ← PC + qqrr + 2, CB ← NB

Adds the 16-bit relative address qqrr (-32768 to 32767) to the program counter (PC) as an offset from the top address +2 of this instruction and unconditionally branches to this address.

Upon branching in the MODEL2/3, the bank address set in the NB is loaded into the CB and the bank is also changed.

Code

MSB							LSB	
1	1	1	1	0	0	1	1	F3H
			r	r				rr
			q	q				qq

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Signed 16-bit PC relative

Example When NB = 02H in the MODEL2/3, it executes the "JRL $+2000H" instruction in the physical address 9000H.

	NB	CB	PC(logical addr.)
Before execution	02H	01H	9000H
			9002H + (2000H - 2)
After execution	02H	02H	B000H

In the above example it branches to the physical address 013000H.
Since there are no NB and CB in the MODEL0/1, it branches to the physical address B000H.

JRL cc1, qqrr *|||||||||| Jump to relative location qqrr if condition cc1 is true ||||||||||||||| 3 cycles |||*

Function **<MODEL0/1>**

```
If cc1 is true
  then  JRL qqrr
  else  PC ←PC + 3
```

<MODEL2/3>

```
If cc1 is true
  then  JRL qqrr
  else  PC ← PC + 3, NB ← CB
```

When the condition cc1 has been established, the CPU executes the "JRL qqrr" instruction and when a condition has not been established, it executes the following instruction.
See "JRL qqrr" instruction.

In the MODEL2/3, when a condition has not been established, the content of the NB specifying the branch destination bank returns to the current bank address (content of the CB).

Condition cc1 is of the below 4 types.

cc1	Condition	
C	Carry	(Carry flag C = 1)
NC	Non Carry	(Carry flag C = 0)
Z	Zero	(Zero flag Z = 1)
NZ	Non Zero	(Zero flag Z = 0)

Code

MSB							LSB	
1	1	1	0	1	1	cc1		ECH–EFH✻
			r	r				rr
			q	q				qq

✻

cc1		Mnemonic	Code
C	00	JRL C,qqrr	ECH
NC	01	JRL NC,qqrr	EDH
Z	10	JRL Z,qqrr	EEH
NZ	11	JRL NZ,qqrr	EFH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Signed 16-bit PC relative

Example At the time of condition establishment, operates the same as the "JRL qqrr" instruction. When a condition has not been established, the operation of the "JRL cc1,qqrr" in the physical address 9000H is as indicated below.

	NB	CB	PC(logical addr.)
Before execution	02H	01H	9000H
After execution	01H	01H	9003H

There are no NB and CB in the MODEL0/1.

JRS rr ||||||||||||||||||||||||||||| *Jump to relative location rr* ||||||||||||||||||||||||||||||||||||||||||||||||||||| *2 cycles* |||

Function **<MODEL0/1>**

PC ← PC + rr + 1

<MODEL2/3>

PC ← PC + rr + 1, CB ← NB

Adds the 8-bit relative address rr (-128 to 127) to the program counter (PC) as an displacement from the top address +1 of this instruction and unconditionally branches to this address.

Upon branching in the MODEL2/3, the bank address set in the NB is loaded into the CB and the bank is also changed.

Code

MSB							LSB	
1	1	1	1	0	0	0	1	F1H
			r	r				rr

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Signed 8-bit PC relative

Example When NB = 02H in the MODEL2/3, it executes the "JRL $+20H" instruction in the physical address 9000H.

	NB	CB	PC(logical addr.)
Before execution	02H	01H	9000H
			9001H + (20H - 1)
After execution	02H	02H	9020H

In the above example it branches to the physical address 011020H.
Since there are no NB and CB in the MODEL0/1, it branches to the physical address 9020H.

JRS cc1, rr ||||||||||||||||| *Jump to relative location rr if condition cc1 is true* ||||||||||||||||||| *2 cycles* |||

Function

```
<MODEL0/1>
  If cc1 is true
    then   JRS rr
    else   PC ← PC + 2

<MODEL2/3>
  If cc1 is true
    then   JRS rr
    else   PC ← PC + 2, NB ← CB
```

When the condition cc1 has been established, the CPU executes the "JRS rr" instruction and when a condition has not been established, it executes the following instruction.

See "JRS rr" instruction.

In the MODEL2/3, when a condition has not been established, the content of the NB specifying the branch destination bank returns to the current bank address (content of the CB).

Condition cc1 is of the below 4 types.

cc1	Condition	
C	Carry	(Carry flag C = 1)
NC	Non Carry	(Carry flag C = 0)
Z	Zero	(Zero flag Z = 1)
NZ	Non Zero	(Zero flag Z = 0)

Code

MSB						LSB	
1	1	1	0	0	1	cc1	E4H–E7H✻
		r	r				rr

✻

cc1		Mnemonic	Code
C	00	JRS C,qqrr	E4H
NC	01	JRS NC,qqrr	E5H
Z	10	JRS Z,qqrr	E6H
NZ	11	JRS NZ,qqrr	E7H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Signed 8-bit PC relative

Example At the time of condition establishment, operates the same as the "JRS rr" instruction. When a condition has not been established, the operation of the "JRS cc1,rr" in the physical address 9000H is as indicated below.

	NB	CB	PC(logical addr.)
Before execution	02H	01H	9000H
After execution	01H	01H	9002H

There are no NB and CB in the MODEL0/1.

JRS cc2, rr ||||||||||||||||| *Jump to relative location rr if condition cc2 is true* ||||||||||||||||||| *3 cycles* |||

Function

```
<MODEL0/1>
  If cc2 is true
    then   PC ← PC + rr + 2
    else   PC ← PC + 3

<MODEL2/3>
  If cc2 is true
    then   PC ← PC + rr + 2, CB ← NB
    else   PC ← PC + 3, NB ← CB
```

When the condition cc2 has been established, it adds the 8-bit relative address rr (-128 to 127) to the program counter (PC) as an offset from the top address +2 of this instruction and branches to that address.
When a condition has not been established, it executes the following instruction.

Upon branching in the MODEL2/3, the bank address set in the NB is loaded into the CB and the bank is also changed. When a condition has not been established, the content of the NB specifying the branch destination bank returns to the current bank address (content of the CB).

JRS cc2, rr

Condition cc2 is of the below 16 types.

cc2	Condition	
LT	Less Than	([N ∀ V] = 1)
LE	Less or Equal	(Z ∨ [N ∀ N] = 1)
GT	Greater Than	(Z ∨ [N ∀ N] = 0)
GE	Greater or Equal	([N ∀ N] = 0)
V	Overflow	(V = 1)
NV	Non Overflow	(V = 0)
P	Plus	(N = 0)
M	Minus	(N = 1)
F0	F0 is set	(F0 = 1)
F1	F1 is set	(F1 = 1)
F2	F2 is set	(F2 = 1)
F3	F3 is set	(F3 = 1)
NF0	F0 is reset	(F0 = 0)
NF1	F1 is reset	(F1 = 0)
NF2	F2 is reset	(F2 = 0)
NF3	F3 is reset	(F3 = 0)

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	1	1	0	cc2				E0H–EFH✻
r				r				rr

✻

cc2		Mnemonic	Code
LT	0000	JRS LT, rr	E0H
LE	0001	JRS LE, rr	E1H
GT	0010	JRS GT, rr	E2H
GE	0011	JRS GE, rr	E3H
V	0100	JRS V, rr	E4H
NV	0101	JRS NV, rr	E5H
P	0110	JRS P, rr	E6H
M	0111	JRS M, rr	E7H
F0	1000	JRS F0, rr	E8H
F1	1001	JRS F1, rr	E9H
F2	1010	JRS F2, rr	EAH
F3	1011	JRS F3, rr	EBH
NF0	1100	JRS NF0, rr	ECH
NF1	1101	JRS NF1, rr	EDH
NF2	1110	JRS NF2, rr	EEH
NF3	1111	JRS NF3, rr	EFH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Signed 8-bit PC relative

Example At the time of condition establishment when NB = 02H in the MODEL2/3, operation of the "JRS cc2,$+20" in the physical address 9000H is as indicated below.

	NB	CB	PC(logical addr.)
Before execution	02H	01H	9000H
			9002H + (20H - 2)
After execution	02H	02H	9020H

In the above example it branches to the physical address 011020H. In the MODEL0/1, since there are no NB and CB, it branches to the physical address 9020H.

When a condition has not been established, the operation of the "JRS cc2,rr" in the physical address 9000H is as indicated below.

	NB	CB	PC(logical addr.)
Before execution	02H	01H	9000H
After execution	01H	01H	9003H

There are no NB and CB in the MODEL0/1.

LD r, r' ||||||||||||||||||||||||||||| *Load r' reg. into r reg.* |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *1 cycle* |||

Function **r ← r'**
Loads the content of the r' register (A/B/L/H) into the r register (A/B/L/H).

Code

MSB					LSB	
0	1	0	r	0	r'	✻

✻

r \ r'	A (0,0)	B (0,1)	L (1,0)	H (1,1)
A(0,0)	40H	41H	42H	43H
B(0,1)	48H	49H	4AH	4BH
L(1,0)	50H	51H	52H	53H
H(1,1)	58H	59H	5AH	5BH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register direct
Dst: Register direct

Example

Set Value		Result					
r	r'	r	r'	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

(r≠r')

LD A, BR ||||||||||||||||||||| *Load BR reg. into A reg.* |||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *2 cycles* |||

Function **A ← BR**
Loads the content of the BR register into the A register.

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	1	0	0	0	0	0	0	C0H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register direct
Dst: Register direct

Example

Set Value		Result					
A	BR	A	BR	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

LD A, SC ||||||||||||||||||||| *Load SC. into A reg.* ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *2 cycles* |||

Function **A ← SC**
Loads the content of the system condition flag (SC) into the A register.

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	1	0	0	0	0	0	1	C1H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register direct
Dst: Register direct

Example

Set Value		Result					
A	SC	A	SC	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

LD A, er *Load er reg. into A reg.* *2 cycles*

Function **A ← er**

Loads the content of the er register (NB/EP/XP/YP) into the A register.

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	1	0	0	1	0	er		C8H–CBH✻

✻

er		Mnemonic	Code
NB	00	LD A, NB	C8H
EP	01	LD A, EP	C9H
XP	10	LD A, XP	CAH
YP	11	LD A, YP	CBH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register direct
Dst: Register direct

Example

Set Value		Result					
A	er	A	er	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

Note This instruction cannot be used in the MODEL0/1.

LD BR, A *Load A reg. into BR reg.* *2 cycles*

Function **BR ← A**

Loads the content of the A register into the BR register.

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	1	0	0	0	0	1	0	C2H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register direct
Dst: Register direct

Example

Set Value		Result					
BR	A	BR	A	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

LD SC, A *Load A reg. into SC* *3 cycles*

Function **SC ← A**

Sets the content of the A register into the system condition flag (SC).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	1	0	0	0	0	1	1	C3H

Flag

I1	I0	U	D	N	V	C	Z
↕	↕	↕	↕	↕	↕	↕	↕

Mode Src: Register direct
Dst: Register direct

Example

Set Value		Result									
SC	A	SC	A	SC							
				I1	I0	U	D	N	V	C	Z
5AH	42H	42H	42H	0	1	0	0	0	0	1	0

LD er, A ||||||||||||||||||||||||| *Load A reg. into er reg.* |||||||||||||||||||||||||||||||||||||||||||||||||||||| *2 or 3 cycles* |||

Function **er ← A**

Loads the content of the A register into the er (NB/EP/XP/YP) register.

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	1	0	0	1	1	er		CCH–CFH✻

✻

er		Mnemonic	Code	
NB	00	LD NB, A	CCH	(3 cycles)
EP	01	LD EP, A	CDH	(2 cycles)
XP	10	LD XP, A	CEH	(2 cycles)
YP	11	LD YP, A	CFH	(2 cycles)

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register direct
Dst: Register direct

Example

Set Value		Result					
er	A	er	A	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

Note This instruction cannot be used in the MODEL0/1.

LD [BR:ll], r |||||||||||||||| *Load r reg. into location [BR:ll]* |||||||||||||||||||||||||||||||||||||||||||||||| *3 cycles* |||

Function **[BR:*ll*] ← r**

Loads the content of the r register (A/B/L/H) into the data memory. The data memory address has been specified by the content of the BR register (upper byte specification) and the 8-bit absolute address *ll* (lower byte specification).
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
0	1	1	1	1	0	r		78H–7BH✻
			l	*l*				*ll*

✻

r		Mnemonic	Code
A	00	LD [BR:*ll*], A	78H
B	01	LD [BR:*ll*], B	79H
L	10	LD [BR:*ll*], L	7AH
H	11	LD [BR:*ll*], H	7BH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register direct
Dst: 8-bit absolute

Example

Set Value		Result					
[BR:*ll*]	r	[BR:*ll*]	r	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

LD [hhll], r ||||||||||||||||||| *Load r reg. into location [hhll]* |||||||||||||||||||||||||||||||||||||||||||||||||| *5 cycles* |||

Function **[hh*ll*] ← r**

Loads the content of the r register (A/B/L/H) into the data memory that has been address specified by the 16-bit absolute address hh*ll*.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Mode Src: Register direct
Dst: 16-bit absolute

Example

Set Value		Result					
[hh*ll*]	r	[hh*ll*]	r	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	1	0	1	0	1	r		D4H–D7H✻
			l	*l*				*ll*
			h	h				hh

✻

r		Mnemonic	Code
A	00	LD [hh*ll*], A	D4H
B	01	LD [hh*ll*], B	D5H
L	10	LD [hh*ll*], L	D6H
H	11	LD [hh*ll*], H	D7H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

LD [HL], r |||||||||||||||||||| *Load r reg. into location [HL]* ||||||||||||||||||||||||||||||||||||||||||||||||||| *2 cycles* |||

Function **[HL] ← r**

Loads the content of the r register (A/B/L/H) into the data memory that has been address specified by the HL register.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Mode Src: Register direct
Dst: Register indirect

Example

Set Value		Result					
[HL]	r	[HL]	r	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

Code

MSB							LSB	
0	1	1	0	1	0	r		68H–6BH✻

✻

r		Mnemonic	Code
A	00	LD [HL], A	68H
B	01	LD [HL], B	69H
L	10	LD [HL], L	6AH
H	11	LD [HL], H	6BH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

LD [IX], r ||||||||||||||||||||||| *Load r reg. into location [IX]* |||||||||||||||||||||||||||||||||||||||||||||||||| *2 cycles* |||

Function **[IX] ← r**
Loads the content of the r register (A/B/L/H) into the data memory that has been address specified by the IX register.
The content of the XP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
0	1	1	0	0	0	r		60H–63H✻

✻

r		Mnemonic	Code
A	00	LD [IX], A	60H
B	01	LD [IX], B	61H
L	10	LD [IX], L	62H
H	11	LD [IX], H	63H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register direct
Dst: Register indirect

Example

Set Value		Result					
[IX]	r	[IX]	r	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

LD [IY], r ||||||||||||||||||||||| *Load r reg. into location [IY]* |||||||||||||||||||||||||||||||||||||||||||||||||| *2 cycles* |||

Function **[IY] ← r**
Loads the content of the r register (A/B/L/H) into the data memory that has been address specified by the IY register.
The content of the YP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
0	1	1	1	0	0	r		70H–73H✻

✻

r		Mnemonic	Code
A	00	LD [IY], A	70H
B	01	LD [IY], B	71H
L	10	LD [IY], L	72H
H	11	LD [IY], H	73H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register direct
Dst: Register indirect

Example

Set Value		Result					
[IY]	r	[IY]	r	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

LD [IX+dd], r |||||||||||||| *Load r reg. into location [IX + dd]* |||||||||||||||||||||||||||||||||||||||||||| *4 cycles* |||

Function **[IX+dd] ← r**

Loads the content of the r register (A/B/L/H) into the data memory that has been address specified by the sum of the content of the IX register and the displacement dd.
The displacement dd is handled as signed data and the range is -128 to 127.
The content of the XP register becomes the page address of the data memory (MODEL2/3).

Code

MSB ... LSB

1	1	0	0	1	1	1	0	CEH
0	1	0	r	r	1	0	0	44H/4CH/54H/5CH✻
d	d	d	d	d	d	d	d	dd

✻

r		Mnemonic	Code
A	00	LD [IX+dd], A	44H
B	01	LD [IX+dd], B	4CH
L	10	LD [IX+dd], L	54H
H	11	LD [IX+dd], H	5CH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register direct
Dst: Register indirect with displacement

Example

Set Value		Result					
[IX+dd]	r	[IX+dd]	r	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

LD [IY+dd], r |||||||||||||| *Load r reg. into location [IY + dd]* |||||||||||||||||||||||||||||||||||||||||||| *4 cycles* |||

Function **[IY+dd] ← r**

Loads the content of the r register (A/B/L/H) into the data memory that has been address specified by the sum of the content of the IY register and the displacement dd.
The displacement dd is handled as signed data and the range is -128 to 127.
The content of the YP register becomes the page address of the data memory (MODEL2/3).

Code

MSB ... LSB

1	1	0	0	1	1	1	0	CEH
0	1	0	r	r	1	0	1	45H/4DH/55H/5DH✻
d	d	d	d	d	d	d	d	dd

✻

r		Mnemonic	Code
A	00	LD [IY+dd], A	45H
B	01	LD [IY+dd], B	4DH
L	10	LD [IY+dd], L	55H
H	11	LD [IY+dd], H	5DH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register direct
Dst: Register indirect with displacement

Example

Set Value		Result					
[IY+dd]	r	[IY+dd]	r	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

LD [IX+L], r |||||||||||||||| *Load r reg. into location [IX + L]* ||||||||||||||||||||||||||||||||||||||||||||| *4 cycles* |||

Function **[IX+L] ← r**
Loads the content of the r register (A/B/L/H) into the data memory that has been address specified by the sum of the content of the IX register and the content of the L register.
The content of the L register is handled as signed data and the range is -128 to 127.
The content of the XP register becomes the page address of the data memory (MODEL2/3).

Mode Src: Register direct
Dst: Register indirect with index register

Example

Set Value		Result					
[IX+L]	r	[IX+L]	r	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

Code

MSB LSB

1	1	0	0	1	1	1	0	CEH
0	1	0	r		1	1	0	46H/4EH/56H/5EH✻

✻

r		Mnemonic	Code
A	00	LD [IX+L], A	46H
B	01	LD [IX+L], B	4EH
L	10	LD [IX+L], L	56H
H	11	LD [IX+L], H	5EH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

LD [IY+L], r |||||||||||||||| *Load r reg. into location [IY + L]* ||||||||||||||||||||||||||||||||||||||||||||| *4 cycles* |||

Function **[IY+L] ← r**
Loads the content of the r register (A/B/L/H) into the data memory that has been address specified by the sum of the content of the IY register and the content of the L register.
The content of the L register is handled as signed data and the range is -128 to 127.
The content of the YP register becomes the page address of the data memory (MODEL2/3).

Mode Src: Register direct
Dst: Register indirect with index register

Example

Set Value		Result					
[IY+L]	r	[IY+L]	r	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

Code

MSB LSB

1	1	0	0	1	1	1	0	CEH
0	1	0	r		1	1	1	47H/4FH/57H/5FH✻

✻

r		Mnemonic	Code
A	00	LD [IY+L], A	47H
B	01	LD [IY+L], B	4FH
L	10	LD [IY+L], L	57H
H	11	LD [IY+L], H	5FH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

LD r, #nn *Load immediate data nn into r reg.* *2 cycles*

Function **r ← nn**
Loads the 8-bit immediate data nn into the r register (A/B/L/H).

Code

MSB							LSB	
1	0	1	1	0	0	r		B0H–B3H✻
			n	n				nn

✻

r		Mnemonic	Code
A	00	LD A, #nn	B0H
B	01	LD B, #nn	B1H
L	10	LD L, #nn	B2H
H	11	LD H, #nn	B3H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result					
				SC			
r	nn	r	nn	N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

LD BR, #hh *Load immediate data hh into BR reg.* *2 cycles*

Function **BR ← hh**
Loads the 8-bit immediate data hh into the BR register.

Code

MSB							LSB	
1	0	1	1	0	1	0	0	B4H
			h	h				hh

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result					
				SC			
BR	hh	BR	hh	N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

LD SC, #nn *Load immediate data nn into SC* *3 cycles*

Function **SC ← nn**
Sets the 8-bit immediate data nn into the system condition flag (SC).

Code

MSB							LSB	
1	0	0	1	1	1	1	1	9FH
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
↕	↕	↕	↕	↕	↕	↕	↕

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result									
				SC							
SC	nn	SC	nn	I1	I0	U	D	N	V	C	Z
5AH	42H	42H	42H	0	1	0	0	0	0	1	0

LD NB, #bb ||||||||||||||||| *Load immediate data bb into NB reg.* ||||||||||||||||||||||||||||||||||||||||| *4 cycles* |||

Function **NB ← bb**

Loads the 8-bit immediate data bb into the new code bank register NB.

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	1	0	0	0	1	0	0	C4H
			b	b				bb

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result					
NB	bb	NB	bb	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

Note This instruction cannot be used in the MODEL0/1.

LD EP, #pp ||||||||||||||||| *Load immediate data pp into EP reg.* ||||||||||||||||||||||||||||||||||||||||| *3 cycles* |||

Function **EP ← pp**

Loads the 8-bit immediate data pp into the expand page register EP.

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	1	0	0	0	1	0	1	C5H
			p	p				pp

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result					
EP	pp	EP	pp	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

Note This instruction cannot be used in the MODEL0/1.

LD XP, #pp ||||||||||||||||| *Load immediate data pp into XP reg.* ||||||||||||||||||||||||||||||||||||||||| *3 cycles* |||

Function **XP ← pp**

Loads the 8-bit immediate data pp into the expand page register XP.

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	1	0	0	0	1	1	0	C5H
			p	p				pp

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result					
XP	pp	XP	pp	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

Note This instruction cannot be used in the MODEL0/1.

LD YP, #pp ||||||||||||||||| Load immediate data pp into YP reg. |||||||||||||||||||||||||||||||||||||||| 3 cycles |||

Function **YP ← pp**
Loads the 8-bit immediate data pp into the expand page register YP.

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	1	0	0	0	1	1	1	C7H
			p	p				pp

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result					
YP	pp	YP	pp	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

Note This instruction cannot be used in the MODEL0/1.

LD [BR:ll], #nn |||||||| Load immediate data nn into location [BR:ll] |||||||||||||||||||||||||| 4 cycles |||

Function **[BR:*ll*] ← nn**
Loads the 8-bit immediate data nn into the data memory. The data memory address has been specified by the content of the BR register (upper byte specification) and the 8-bit absolute address *ll* (lower byte specification).
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	1	1	1	0	1	DDH
			l	*l*				*ll*
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Immediate data
Dst: 8-bit absolute

Example

Set Value		Result					
[BR:*ll*]	nn	[BR:*ll*]	nn	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

LD [HL], #nn ||||||||||||| Load immediate data nn into location [HL] ||||||||||||||||||||||||||||| 3 cycles |||

Function **[HL] ← nn**
Loads the 8-bit immediate data nn into the data memory that has been address specified by the HL register.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	0	1	1	0	1	0	1	B5H
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Immediate data
Dst: Register indirect

Example

Set Value		Result					
[HL]	nn	[HL]	nn	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

LD [ir], #nn ||||||||||||||||| *Load immediate data nn into location [ir reg.]* |||||||||||||||||||||||||||| *3 cycles* |||

Function **[ir] ← nn**

Loads the 8-bit immediate data nn into the data memory that has been address specified by the ir register (IX/IY).
The content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory (MODEL2/3).

Mode Src: Immediate data
Dst: Register indirect

Example

Set Value		Result					
				SC			
[ir]	nn	[ir]	nn	N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

Code

MSB							LSB	
1	0	1	1	0	1	1	ir	B6H/B7H✻
n	n							nn

✻

ir		Mnemonic	Code
IX	0	LD [IX], #nn	B6H
IY	1	LD [IY], #nn	B7H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

LD r, [BR:ll] |||||||||||||||| *Load location [BR:ll] into r reg.* ||||||||||||||||||||||||||||||||||||||||||||||||| *3 cycles* |||

Function **r ← [BR:*ll*]**

Loads the content of the data memory into the r register (A/B/L/H). The data memory address has been specified by the content of the BR register (upper byte specification) and the 8-bit absolute address *ll* (lower byte specification).
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Mode Src: 8-bit absolute
Dst: Register direct

Example

Set Value		Result					
				SC			
r	[BR:*ll*]	r	[BR:*ll*]	N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

Code

MSB						LSB	
0	1	0	r	1	0	0	44H/4CH/54H/5CH✻
l	*l*						*ll*

✻

r		Mnemonic	Code
A	00	LD A, [BR:*ll*]	44H
B	01	LD B, [BR:*ll*]	4CH
L	10	LD L, [BR:*ll*]	54H
H	11	LD H, [BR:*ll*]	5CH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

LD [HL], [BR:ll] ||||||| *Load location [BR:ll] into location [HL]* |||||||||||||||||||||||||||||||||||| *4 cycles* |||

Function **[HL] ← [BR:*ll*]**

Loads the content of the data memory [BR:*ll*] into the data memory [HL]. The data memory [BR:*ll*] address has been specified by the content of the BR register (upper byte specification) and the 8-bit absolute address *ll* (lower byte specification). The data memory [HL] address has been specified by the HL register.

The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
0	1	1	0	1	1	0	0	6CH
			l	*l*				*ll*

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: 8-bit absolute
Dst: Register indirect

Example

Set Value		Result					
[HL]	[BR:*ll*]	[HL]	[BR:*ll*]	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

LD [ir], [BR:ll] ||||||||||| *Load location [BR:ll] into location [ir reg.]* ||||||||||||||||||||||||||||||| *4 cycles* |||

Function **[ir] ← [BR:*ll*]**

Loads the content of the data memory [BR:*ll*] into the data memory [ir]. The data memory [BR:*ll*] address has been specified by the content of the BR register (upper byte specification) and the 8-bit absolute address *ll* (lower byte specification). The data memory [ir] address has been specified by the ir register (IX/IY).

The content of the EP register becomes the page address of the data memory [BR:*ll*] and the content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory [ir] (MODEL2/3).

Code

MSB							LSB	
0	1	1	ir	0	1	0	0	64H/74H✻
			l	*l*				*ll*

✻

ir		Mnemonic	Code
IX	0	LD [IX], [BR:*ll*]	64H
IY	1	LD [IY], [BR:*ll*]	74H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: 8-bit absolute
Dst: Register indirect

Example

Set Value		Result					
[ir]	[BR:*ll*]	[ir]	[BR:*ll*]	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

LD r, [hhll] Load location [hhll] into r reg. 5 cycles

Function **r ← [hh*ll*]**

Loads the content of the data memory that has been address specified by the 16-bit absolute address hh*ll* into the r register (A/B/L/H).

The content of the EP register becomes the page address of the data memory (MODEL2/3).

Mode Src: 16-bit absolute
Dst: Register direct

Example

Set Value		Result					
r	[hh*ll*]	r	[hh*ll*]	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	1	0	1	0	0	r		D0H–D3H✻
			l	*l*				*ll*
			h	h				hh

✻

r		Mnemonic	Code
A	00	LD A, [hh*ll*]	D0H
B	01	LD B, [hh*ll*]	D1H
L	10	LD L, [hh*ll*]	D2H
H	11	LD H, [hh*ll*]	D3H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

LD r, [HL] Load location [HL] into r reg. 2 cycles

Function **r ← [HL]**

Loads the content of the data memory that has been address specified by the HL register into the r register (A/B/L/H).

The content of the EP register becomes the page address of the data memory (MODEL2/3).

Mode Src: Register indirect
Dst: Register direct

Example

Set Value		Result					
r	[HL]	r	[HL]	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

Code

MSB						LSB	
0	1	0	r	1	0	1	45H/4DH/55H/5DH✻

✻

r		Mnemonic	Code
A	00	LD A, [HL]	45H
B	01	LD B, [HL]	4DH
L	10	LD L, [HL]	55H
H	11	LD H, [HL]	5DH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

LD [BR:ll], [HL] ||||||| *Load location [HL] into location [BR:ll]* ||||||||||||||||||||||||||||||||||| *4 cycles* |||

Function **[BR:*ll*] ← [HL]**

Loads the content of the data memory [HL] into the data memory [BR:*ll*]. The data memory [BR:*ll*] address has been specified by the content of the BR register (upper byte specification) and the 8-bit absolute address *ll* (lower byte specification). The data memory [HL] address has been specified by the HL register.

The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
0	1	1	1	1	1	0	1	7DH
			l	*l*				*ll*

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register indirect
Dst: 8-bit absolute

Example

Set Value		Result					
[BR:*ll*]	[HL]	[BR:*ll*]	[HL]	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

LD [HL], [HL] |||||||||||| *Load location [HL] into location [HL]* |||||||||||||||||||||||||||||||||||||| *3 cycles* |||

Function **[HL] ← [HL]**

Loads the content of the data memory that has been address specified by the HL register into the same address. As a result, only three cycles are consumed.

The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
0	1	1	0	1	1	0	1	6DH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register indirect
Dst: Register indirect

Example

Set Value	Result				
[HL]	[HL]	SC			
		N	V	C	Z
42H	42H	–	–	–	–

LD [ir], [HL] |||||||||||||||| *Load location [HL] into location [ir reg.]* ||||||||||||||||||||||||||||||||| *3 cycles* |||

Function **[ir] ← [HL]**

Loads the content of the data memory [HL] into the data memory [ir]. The data memory [HL] address has been specified by the HL register. The data memory [ir] address has been specified by the ir register (IX/IY).

The content of the EP register becomes the page address of the data memory [HL] and the content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory [ir] (MODEL2/3).

Code

MSB							LSB	
0	1	1	ir	0	1	0	1	65H/75H✻

✻

ir		Mnemonic	Code
IX	0	LD [IX], [HL]	65H
IY	1	LD [IY], [HL]	75H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register indirect
Dst: Register indirect

Example

Set Value		Result					
[ir]	[HL]	[ir]	[HL]	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

LD r, [IX] ||||||||||||||||||||||| *Load location [IX] into r reg.* |||||||||||||||||||||||||||||||||||||||||||||||||||| *2 cycles* |||

Function **r ← [IX]**

Loads the content of the data memory that has been address specified by the IX register into the r register (A/B/L/H).
The content of the XP register becomes the page address of the data memory (MODEL2/3).

Code

MSB						LSB	
0	1	0	r	1	1	0	46H/4EH/56H/5EH✻

✻

r		Mnemonic	Code
A	00	LD A, [IX]	46H
B	01	LD B, [IX]	4EH
L	10	LD L, [IX]	56H
H	11	LD H, [IX]	5EH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register indirect
Dst: Register direct

Example

Set Value		Result					
r	[IX]	r	[IX]	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

LD [BR:ll], [IX] ||||||||| *Load location [IX] into location [BR:ll]* |||||||||||||||||||||||||||||||||||| *4 cycles* |||

Function **[BR:*ll*] ← [IX]**

Loads the content of the data memory [IX] into the data memory [BR:*ll*]. The data memory [BR:*ll*] address has been specified by the content of the BR register (upper byte specification) and the 8-bit absolute address *ll* (lower byte specification). The data memory [IX] address has been specified by the IX register.
The content of the EP register becomes the page address of the data memory [BR:*ll*] and the content of the XP register becomes the page address of the data memory [IX] (MODEL2/3).

Code

MSB							LSB	
0	1	1	1	1	1	1	0	7EH
l				*l*				*ll*

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register indirect
Dst: 8-bit absolute

Example

Set Value		Result					
[BR:*ll*]	[IX]	[BR:*ll*]	[IX]	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

LD [HL], [IX] |||||||||||||| *Load location [IX] into location [HL]* ||||||||||||||||||||||||||||||||||||| *3 cycles* |||

Function **[HL] ← [IX]**

Loads the content of the data memory [IX] into the data memory [HL]. The data memory [HL] address has been specified by the HL register. The data memory [IX] address has been specified by the IX register.
The content of the EP register becomes the page address of the data memory [HL] and the content of the XP register becomes the page address of the data memory [IX] (MODEL2/3).

Code

MSB							LSB	
0	1	1	0	1	1	1	0	6EH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register indirect
Dst: Register indirect

Example

Set Value		Result					
[HL]	[IX]	[HL]	[IX]	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

LD [ir], [IX] ||||||||||||||||| *Load location [IX] into location [ir reg.]* |||||||||||||||||||||||||||||||||| *3 cycles* |||

Function **[ir] ← [IX]**

Loads the content of the data memory [IX] into the data memory [ir]. The data memory [IX] address has been specified by the IX register. The data memory [ir] address has been specified by the ir register (IX/IY).
The content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory (MODEL2/3).

Mode Src: Register indirect
Dst: Register indirect

Example

Set Value		Result					
				SC			
[IY]	[IX]	[IY]	[IX]	N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

Code

MSB LSB

0	1	1	ir	0	1	1	0

66H/76H✻

✻

ir		Mnemonic	Code
IX	0	LD [IX], [IX]	66H
IY	1	LD [IY], [IX]	76H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

LD r, [IY] |||||||||||||||||||||| *Load location [IY] into r reg.* |||||||||||||||||||||||||||||||||||||||||||||||||| *2 cycles* |||

Function **r ← [IY]**

Loads the content of the data memory that has been address specified by the IY register into the r register (A/B/L/H).
The content of the YP register becomes the page address of the data memory (MODEL2/3).

Mode Src: Register indirect
Dst: Register direct

Example

Set Value		Result					
				SC			
r	[IY]	r	[IY]	N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

Code

MSB LSB

0	1	0	r		1	1	1

47H/4FH/57H/5FH✻

✻

r		Mnemonic	Code
A	00	LD A, [IY]	47H
B	01	LD B, [IY]	4FH
L	10	LD L, [IY]	57H
H	11	LD H, [IY]	5FH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

LD [BR:ll], [IY] |||||||| Load location [IY] into location [BR:ll] ||||||||||||||||||||||||||||||||||| 4 cycles |||

Function **[BR:*ll*] ← [IY]**

Loads the content of the data memory [IY] into the data memory [BR:*ll*]. The data memory [BR:*ll*] address has been specified by the content of the BR register (upper byte specification) and the 8-bit absolute address *ll* (lower byte specification). The data memory [IY] address has been specified by the IY register.

The content of the EP register becomes the page address of the data memory [BR:*ll*] and the content of the YP register becomes the page address of the data memory [IY] (MODEL2/3).

Code

MSB							LSB	
0	1	1	1	1	1	1	1	7FH
			l	*l*				*ll*

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register indirect
Dst: 8-bit absolute

Example

Set Value		Result					
				SC			
[BR:*ll*]	[IY]	[BR:*ll*]	[IY]	N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

LD [HL], [IY] |||||||||||||| Load location [IY] into location [HL] |||||||||||||||||||||||||||||||||||||| 3 cycles |||

Function **[HL] ← [IY]**

Loads the content of the data memory [IY] into the data memory [HL]. The data memory [HL] address has been specified by the HL register. The data memory [IY] address has been specified by the IY register.

The content of the EP register becomes the page address of the data memory [HL] and the content of the YP register becomes the page address of the data memory [IY] (MODEL2/3).

Code

MSB							LSB	
0	1	1	0	1	1	1	1	6FH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register indirect
Dst: Register indirect

Example

Set Value		Result					
				SC			
[HL]	[IY]	[HL]	[IY]	N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

LD [ir], [IY] ||||||||||||||||| Load location [IY] into location [ir reg.] ||||||||||||||||||||||||||||||||| 3 cycles |||

Function **[ir] ← [IY]**

Loads the content of the data memory [IY] into the data memory [ir]. The data memory [IY] address has been specified by the IY register. The data memory [ir] address has been specified by the ir register (IX/IY).

The content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory (MODEL2/3).

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register indirect
Dst: Register indirect

Example

Set Value		Result					
				SC			
[IX]	[IY]	[IX]	[IY]	N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

Code

MSB							LSB	
0	1	1	ir	0	1	1	1	67H/77H✻

✻

ir		Mnemonic	Code
IX	0	LD [IX], [IY]	67H
IY	1	LD [IY], [IY]	77H

LD r, [IX+dd] *Load location [IX + dd] into r reg. 4 cycles*

Function **r ← [IX+dd]**

Loads the content of the data memory that has been address specified by the sum of the content of the IX register and the displacement dd into the r register (A/B/L/H).
The displacement dd is handled as signed data and the range is -128 to 127.
The content of the XP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	1	0	r		0	0	0	40H/48H/50H/58H✻
			d	d				dd

✻

r		Mnemonic	Code
A	00	LD A, [IX+dd]	40H
B	01	LD B, [IX+dd]	48H
L	10	LD L, [IX+dd]	50H
H	11	LD H, [IX+dd]	58H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register indirect with displacement
Dst: Register direct

Example

Set Value		Result					
r	[IX+dd]	r	[IX+dd]	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

LD [HL], [IX+dd] *Load location [IX + dd] into location [HL] 5 cycles*

Function **[HL] ← [IX+dd]**

Loads the content of the data memory [IX+dd] into the data memory [HL]. The data memory [HL] address has been specified by the HL register. The data memory [IX+dd] address has been specified by the sum of the content of the IX register and the displacement dd.
The displacement dd is handled as signed data and the range is -128 to 127.
The content of the EP register becomes the page address of the data memory [HL] and the content of the XP register becomes the page address of the data memory [IX+dd] (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	1	1	0	0	0	0	0	60H
			d	d				dd

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register indirect with displacement
Dst: Register indirect

Example

Set Value		Result					
[HL]	[IX+dd]	[HL]	[IX+dd]	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

LD [ir], [IX+dd] *Load location [IX + dd] into location [ir reg.]* *5 cycles*

Function **[ir] ← [IX+dd]**

Loads the content of the data memory [IX+dd] into the data memory [ir]. The data memory [ir] address has been specified by the ir register (IX/IY). The data memory [IX+dd] address has been specified by the sum of the content of the IX register and the displacement dd.
The displacement dd is handled as signed data and the range is -128 to 127.
The content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	1	1	ir	1	0	0	0	68H/78H✻
			d	d				dd

✻

ir		Mnemonic	Code
IX	0	LD [IX], [IX+dd]	68H
IY	1	LD [IY], [IX+dd]	78H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register indirect with displacement
Dst: Register indirect

Example

Set Value		Result					
[ir]	[IX+dd]	[ir]	[IX+dd]	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

LD r, [IY+dd] *Load location [IY + dd] into r reg.* *4 cycles*

Function **r ← [IY+dd]**

Loads the content of the data memory that has been address specified by the sum of the content of the IY register and the displacement dd into the r register (A/B/L/H).
The displacement dd is handled as signed data and the range is -128 to 127.
The content of the YP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	1	0	r	r	0	0	1	41H/49H/51H/59H✻
			d	d				dd

✻

r		Mnemonic	Code
A	00	LD A, [IY+dd]	41H
B	01	LD B, [IY+dd]	49H
L	10	LD L, [IY+dd]	51H
H	11	LD H, [IY+dd]	59H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register indirect with displacement
Dst: Register direct

Example

Set Value		Result					
r	[IY+dd]	r	[IY+dd]	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

LD [HL], [IY+dd] ||||| Load location [IY + dd] into location [HL] |||||||||||||||||||||||||||||| 5 cycles |||

Function **[HL] ← [IY+dd]**

Loads the content of the data memory [IY+dd] into the data memory [HL]. The data memory [HL] address has been specified by the HL register. The data memory [IY+dd] address has been specified by the sum of the content of the IY register and the displacement dd.

The displacement dd is handled as signed data and the range is -128 to 127.

The content of the EP register becomes the page address of the data memory [HL] and the content of the YP register becomes the page address of the data memory [IY+dd] (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	1	1	0	0	0	0	1	61H
			d	d				dd

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register indirect with displacement
Dst: Register indirect

Example

Set Value		Result					
[HL]	[IY+dd]	[HL]	[IY+dd]	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

LD [ir], [IY+dd] |||||||| Load location [IY + dd] into location [ir reg.] |||||||||||||||||||||||||| 5 cycles |||

Function **[ir] ← [IX+dd]**

Loads the content of the data memory [IY+dd] into the data memory [ir]. The data memory [ir] address has been specified by the ir register (IX/IY). The data memory [IY+dd] address has been specified by the sum of the content of the IY register and the displacement dd.

The displacement dd is handled as signed data and the range is -128 to 127.

The content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	1	1	ir	1	0	0	1	69H/79H✻
			d	d				dd

✻

ir		Mnemonic	Code
IX	0	LD [IX], [IY+dd]	69H
IY	1	LD [IY], [IY+dd]	79H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register indirect with displacement
Dst: Register indirect

Example

Set Value		Result					
[ir]	[IY+dd]	[ir]	[IY+dd]	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

LD r, [IX+L] |||||||||||||||| *Load location [IX +L] into r reg.* ||||||||||||||||||||||||||||||||||||||||||||| *4 cycles* |||

Function **r ← [IX+L]**
Loads the content of the data memory that has been address specified by the sum of the content of the IX register and the content of the L register into the r register (A/B/L/H).
The content of the L register is handled as signed data and the range is -128 to 127.
The content of the XP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	1	0	r		0	1	0	42H/4AH/52H/5AH✻

✻

r		Mnemonic	Code
A	00	LD A, [IX+L]	42H
B	01	LD B, [IX+L]	4AH
L	10	LD L, [IX+L]	52H
H	11	LD H, [IX+L]	5AH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register indirect with index register
Dst: Register direct

Example

Set Value		Result					
r	[IX+L]	r	[IX+L]	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

LD [HL], [IX+L] |||||||| *Load location [IX + L] into location [HL]* ||||||||||||||||||||||||||||||||| *5 cycles* |||

Function **[HL] ← [IX+L]**
Loads the content of the data memory [IX+L] into the data memory [HL]. The data memory [HL] address has been specified by the HL register. The data memory [IX+L] address has been specified by the sum of the content of the IX register and the content of the L register.
The content of the L register is handled as signed data and the range is -128 to 127.
The content of the EP register becomes the page address of the data memory [HL] and the content of the XP register becomes the page address of the data memory [IX+L] (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	1	1	0	0	0	1	0	62H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register indirect with index register
Dst: Register indirect

Example

Set Value		Result					
[HL]	[IX+L]	[HL]	[IX+L]	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

LD [ir], [IX+L] *Load location [IX + L] into location [ir reg.]* *5 cycles*

Function **[ir] ← [IX+L]**

Loads the content of the data memory [IX+L] into the data memory [ir]. The data memory [ir] address has been specified by the ir register (IX/IY). The data memory [IX+L] address has been specified by the sum of the content of the IX register and the content of the L register.
The content of the L register is handled as signed data and the range is -128 to 127.
The content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	1	1	ir	1	0	1	0	6AH/7AH✻

✻

ir		Mnemonic	Code
IX	0	LD [IX], [IX+L]	6AH
IY	1	LD [IY], [IX+L]	7AH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register indirect with index register
Dst: Register indirect

Example

Set Value		Result					
[ir]	[IX+L]	[ir]	[IX+L]	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

LD r, [IY+L] *Load location [IY +L] into r reg.* *4 cycles*

Function **r ← [IY+L]**

Loads the content of the data memory that has been address specified by the sum of the content of the IY register and the content of the L register into the r register (A/B/L/H).
The content of the L register is handled as signed data and the range is -128 to 127.
The content of the YP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	1	0	r	r	0	1	1	43H/4BH/53H/5BH✻

✻

r		Mnemonic	Code
A	00	LD A, [IY+L]	43H
B	01	LD B, [IY+L]	4BH
L	10	LD L, [IY+L]	53H
H	11	LD H, [IY+L]	5BH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register indirect with index register
Dst: Register direct

Example

Set Value		Result					
r	[IY+L]	r	[IY+L]	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

LD [HL], [IY+L] |||||||| *Load location [IY + L] into location [HL]* |||||||||||||||||||||||||||||||| *5 cycles* |||

Function **[HL] ← [IY+L]**

Loads the content of the data memory [IY+L] into the data memory [HL]. The data memory [HL] address has been specified by the HL register. The data memory [IY+L] address has been specified by the sum of the content of the IY register and the content of the L register.
The content of the L register is handled as signed data and the range is -128 to 127.
The content of the EP register becomes the page address of the data memory [HL] and the content of the YP register becomes the page address of the data memory [IY+L] (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	1	1	0	0	0	1	1	63H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register indirect with index register
Dst: Register indirect

Example

Set Value		Result					
[HL]	[IY+L]	[HL]	[IY+L]	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

LD [ir], [IY+L] |||||||||||| *Load location [IY + L] into location [ir reg.]* ||||||||||||||||||||||||||| *5 cycles* |||

Function **[ir] ← [IY+L]**

Loads the content of the data memory [IY+L] into the data memory [ir]. The data memory [ir] address has been specified by the ir register (IX/IY). The data memory [IY+L] address has been specified by the sum of the content of the IY register and the content of the L register.
The content of the L register is handled as signed data and the range is -128 to 127.
The content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	1	1	ir	1	0	1	1	6BH/7BH✻

✻

ir		Mnemonic	Code
IX	0	LD [IX], [IY+L]	6BH
IY	1	LD [IY], [IY+L]	7BH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register indirect with index register
Dst: Register indirect

Example

Set Value		Result					
[ir]	[IY+L]	[ir]	[IY+L]	SC			
				N	V	C	Z
5AH	42H	42H	42H	–	–	–	–

LD rp, rp' *Load rp' reg. into rp reg.* *2 cycles*

Function **rp ← rp'**
Loads the content of the rp' register (BA/HL/IX/IY) into the rp register (BA/HL/IX/IY).

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
1	1	1	0	rp		rp'		✻

✻

rp \ rp'	BA (0,0)	HL (0,1)	IX (1,0)	IY (1,1)
BA(0,0)	E0H	E1H	E2H	E3H
HL(0,1)	E4H	E5H	E6H	E7H
IX(1,0)	E8H	E9H	EAH	EBH
IY(1,1)	ECH	EDH	EEH	EFH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register direct
Dst: Register direct

Example

Set Value		Result					
rp	rp'	rp	rp'	SC			
				N	V	C	Z
3521H	E964H	E964H	E964H	–	–	–	–

(rp≠rp')

LD BA, PC *Load PC into BA reg.* *2 cycles*

Function **BA ← PC + 2**
Loads the content of the program counter (PC) into the BA register. The value that has been loaded is top address of this instruction + 2.

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
1	1	1	1	1	0	0	1	F9H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register direct
Dst: Register direct

Example

Set Value		Result					
BA	PC	BA	PC	SC			
				N	V	C	Z
3521H	E964H	E964H	E964H	–	–	–	–

LD BA, SP *Load SP into BA reg.* *2 cycles*

Function **BA ← SP**
Loads the content of the stack pointer (SP) into the BA register.

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
1	1	1	1	1	0	0	0	F8H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register direct
Dst: Register direct

Example

Set Value		Result					
BA	SP	BA	SP	SC			
				N	V	C	Z
3521H	E964H	E964H	E964H	–	–	–	–

LD HL, PC — *Load PC into HL reg.* — *2 cycles*

Function **HL ← PC + 2**

Loads the content of the program counter (PC) into the HL register. The value that has been loaded is top address of this instruction + 2.

Code

MSB LSB

1	1	0	0	1	1	1	1	CFH
1	1	1	1	0	1	0	1	F5H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register direct
Dst: Register direct

Example

Set Value		Result					
HL	PC	HL	PC	SC			
				N	V	C	Z
3521H	E964H	E964H	E964H	–	–	–	–

LD HL, SP — *Load SP into HL reg.* — *2 cycles*

Function **HL ← SP**

Loads the content of the stack pointer (SP) into the HL register.

Code

MSB LSB

1	1	0	0	1	1	1	1	CFH
1	1	1	1	0	1	0	0	F4H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register direct
Dst: Register direct

Example

Set Value		Result					
HL	SP	HL	SP	SC			
				N	V	C	Z
3521H	E964H	E964H	E964H	–	–	–	–

LD IX, SP — *Load SP into IX reg.* — *2 cycles*

Function **IX ← SP**

Loads the content of the stack pointer (SP) into the IX register.

Code

MSB LSB

1	1	0	0	1	1	1	1	CFH
1	1	1	1	1	0	1	0	FAH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register direct
Dst: Register direct

Example

Set Value		Result					
IX	SP	IX	SP	SC			
				N	V	C	Z
3521H	E964H	E964H	E964H	–	–	–	–

LD IY, SP — *Load SP into IY reg.* — *2 cycles*

Function **IY ← SP**

Loads the content of the stack pointer (SP) into the IY register.

Code

MSB LSB

1	1	0	0	1	1	1	1	CFH
1	1	1	1	1	1	1	0	FEH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register direct
Dst: Register direct

Example

Set Value		Result					
IY	SP	IY	SP	SC			
				N	V	C	Z
3521H	E964H	E964H	E964H	–	–	–	–

LD SP, rp |||||||||||||||||||| Load rp reg. into SP |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 2 cycles |||

Function **SP ← rp**

Loads the content of the rp register (BA/HL/IX/IY) into the stack pointer (SP).

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
1	1	1	1	0	0	rp		F0H–F3H✻

✻

rp		Mnemonic	Code
BA	00	LD SP, BA	F0H
HL	01	LD SP, HL	F1H
IX	10	LD SP, IX	F2H
IY	11	LD SP, IY	F3H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register direct
Dst: Register direct

Example

Set Value		Result					
				SC			
SP	rp	SP	rp	N	V	C	Z
3521H	E964H	E964H	E964H	–	–	–	–

LD [hhll], rp |||||||||||||| Load rp reg. into location [hhll] |||||||||||||||||||||||||||||||||||||||||||||||| 5 cycles |||

Function **[hh*ll*] ← rp(L), [hh*ll*+1] ← rp(H)**

Stores the lower byte of the rp register (BA/HL/IX/IY) in the address hh*ll* of the data memory and stores the upper byte in the following address hh*ll*+1.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	0	1	1	1	1	rp		BCH–BFH✻
			l	*l*				*ll*
			h	h				hh

✻

rp		Mnemonic	Code
BA	00	LD [hh*ll*], BA	BCH
HL	01	LD [hh*ll*], HL	BDH
IX	10	LD [hh*ll*], IX	BEH
IY	11	LD [hh*ll*], IY	BFH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register direct
Dst: 16-bit absolute

Example

Set Value	Result						
				SC			
rp	[hh*ll*]	[hh*ll*+1]	rp	N	V	C	Z
E964H	64H	E9H	E964H	–	–	–	–

LD [hhll], SP |||||||||||| *Load SP into location [hhll]* |||||||||||||||||||||||||||||||||||||||||||||||||| *6 cycles* |||

Function **[hh*ll*] ← SP(L), [hh*ll*+1] ← SP(H)**
Stores the lower byte of the stack pointer (SP) in the address hh*ll* of the data memory and stores the upper byte in the following address hh*ll*+1.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	1	1	1	1	1	0	0	7CH
			l	*l*				*ll*
			h	h				hh

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register direct
Dst: 16-bit absolute

Example

Set Value	Result						
SP	[hh*ll*]	[hh*ll*+1]	SP	SC			
				N	V	C	Z
E964H	64H	E9H	E964H	–	–	–	–

LD [HL], rp |||||||||||||||| *Load rp reg. into location [HL]* |||||||||||||||||||||||||||||||||||||||||||||||| *5 cycles* |||

Function **[HL] ← rp(L), [HL+1] ← rp(H)**
Stores the lower byte of the rp register (BA/HL/IX/IY) in the address of the data memory that has been specified by the content of the HL register and stores the upper byte in the following address.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
1	1	0	0	0	1	rp		C4H–C7H✻

✻

rp		Mnemonic	Code
BA	00	LD [HL], BA	C4H
HL	01	LD [HL], HL	C5H
IX	10	LD [HL], IX	C6H
IY	11	LD [HL], IY	C7H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register direct
Dst: Register indirect

Example

Set Value	Result						
rp	[HL]	[HL+1]	rp	SC			
				N	V	C	Z
E964H	64H	E9H	E964H	–	–	–	–

LD [IX], rp |||||||||||||||||| *Load rp reg. into location [IX]* |||||||||||||||||||||||||||||||||||||||||||||||||| *5 cycles* |||

Function **[IX] ← rp(L), [IX+1] ← rp(H)**

Stores the lower byte of the rp register (BA/HL/IX/IY) in the address of the data memory that has been specified by the content of the IX register and stores the upper byte in the following address.
The content of the XP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
1	1	0	1	0	1	rp		D4H–D7H*

*

rp		Mnemonic	Code
BA	00	LD [IX], BA	D4H
HL	01	LD [IX], HL	D5H
IX	10	LD [IX], IX	D6H
IY	11	LD [IX], IY	D7H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register direct
Dst: Register indirect

Example

Set Value	Result						
rp	[IX]	[IX+1]	rp	SC			
				N	V	C	Z
E964H	64H	E9H	E964H	–	–	–	–

LD [IY], rp |||||||||||||||||| *Load rp reg. into location [IY]* |||||||||||||||||||||||||||||||||||||||||||||||||| *5 cycles* |||

Function **[IY] ← rp(L), [IY+1] ← rp(H)**

Stores the lower byte of the rp register (BA/HL/IX/IY) in the address of the data memory that has been specified by the content of the IY register and stores the upper byte in the following address.
The content of the YP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
1	1	0	1	1	1	rp		DCH–DFH*

*

rp		Mnemonic	Code
BA	00	LD [IY], BA	DCH
HL	01	LD [IY], HL	DDH
IX	10	LD [IY], IX	DEH
IY	11	LD [IY], IY	DFH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register direct
Dst: Register indirect

Example

Set Value	Result						
rp	[IY]	[IY+1]	rp	SC			
				N	V	C	Z
E964H	64H	E9H	E964H	–	–	–	–

LD [SP+dd], rp |||||||| *Load rp reg. into location [SP + dd]* |||||||||||||||||||||||||||||||||||||||||| *6 cycles* |||

Function **[SP+dd] ← rp(L), [SP+dd+1] ← rp(H)**

Stores the lower byte of the rp register (BA/HL/IX/IY) in the address of the data memory that has been specified by the sum of the content of the SP and the displacement dd, and stores the upper byte in the following address.
The displacement dd is handled as signed data and the range is -128 to 127.

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	1	1	1	0	1	rp		74H–77H✻
d	d	d	d	d	d	d	d	dd

✻

rp		Mnemonic	Code
BA	00	LD [SP], BA	74H
HL	01	LD [SP], HL	75H
IX	10	LD [SP], IX	76H
IY	11	LD [SP], IY	77H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register direct
Dst: Register indirect with displacement

Example

Set Value	Result						
rp	[SP+dd]	[SP+dd+1]	rp	SC			
				N	V	C	Z
E964H	64H	E9H	E964H	–	–	–	–

LD rp, #mmnn ||||||||| *Load immediate data mmnn into rp reg.* |||||||||||||||||||||||||||||||||||| *3 cycles* |||

Function **rp ← mmnn**

Loads the 16-bit immediate data mmnn into the rp register (BA/HL/IX/IY).

Code

MSB							LSB	
1	1	0	0	0	1	rp		C4H–C7H✻
n	n	n	n	n	n	n	n	nn
m	m	m	m	m	m	m	m	mm

✻

rp		Mnemonic	Code
BA	00	LD BA, #mmnn	C4H
HL	01	LD HL, #mmnn	C5H
IX	10	LD IX, #mmnn	C6H
IY	11	LD IY, #mmnn	C7H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result					
rp	mmnn	rp	mmnn	SC			
				N	V	C	Z
3521H	E964H	E964H	E964H	–	–	–	–

LD SP, #mmnn *Load immediate data mmnn into SP* *4 cycles*

Function **SP ← mmnn**
Loads the 16-bit immediate data mmnn into the stack pointer (SP).

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result					
SP	mmnn	SP	mmnn	SC			
				N	V	C	Z
3521H	E964H	E964H	E964H	–	–	–	–

Code

MSB … LSB

1	1	0	0	1	1	1	1	CFH
0	1	1	0	1	1	1	0	6EH
			n	n				nn
			m	m				mm

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

LD rp, [hhll] *Load location [hhll] into rp reg.* *5 cycles*

Function **rp(L) ← [hh*ll*], rp(H) ← [hh*ll*+1]**
Loads the content of the data memory that has been address specified by the 16-bit immediate data hh*ll* into the rp register (BA/HL/IX/IY) as the lower byte and loads the content of the following address as the upper byte.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Mode Src: 16-bit absolute
Dst: Register direct

Example

Set Value			Result				
rp	[hh*ll*]	[hh*ll*+1]	rp	SC			
				N	V	C	Z
3521H	64H	E9H	E964H	–	–	–	–

Code

MSB … LSB

1	0	1	1	1	0	rp	B8H–BBH✻
			l	*l*			*ll*
			h	h			hh

✻

rp		Mnemonic	Code
BA	00	LD BA, [hh*ll*]	B8H
HL	01	LD HL, [hh*ll*]	B9H
IX	10	LD IX, [hh*ll*]	BAH
IY	11	LD IY, [hh*ll*]	BBH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

LD SP, [hhll] — *Load location [hhll] into SP* — *6 cycles*

Function **SP(L) ← [hh*ll*], SP(H) ← [hh*ll*+1]**
Loads the content of the data memory that has been address specified by the 16-bit immediate data hh*ll* into the stack pointer (SP) as the lower byte and loads the content of the following address as the upper byte.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	1	1	1	1	0	0	0	78H
			l	*l*				*ll*
			h	h				hh

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: 16-bit absolute
Dst: Register direct

Example

Set Value			Result				
SP	[hh*ll*]	[hh*ll*+1]	SP	SC			
				N	V	C	Z
3521H	64H	E9H	E964H	–	–	–	–

LD rp, [HL] — *Load location [HL] into rp reg.* — *5 cycles*

Function **rp(L) ← [HL], rp(H) ← [HL+1]**
Loads the content of the data memory that has been address specified by the HL register into the rp register (BA/HL/IX/IY) as the lower byte and loads the content of the following address as the upper byte.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
1	1	0	0	0	0	rp		C0H–C3H✻

✻

rp		Mnemonic	Code
BA	00	LD BA, [HL]	C0H
HL	01	LD HL, [HL]	C1H
IX	10	LD IX, [HL]	C2H
IY	11	LD IY, [HL]	C3H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register indirect
Dst: Register direct

Example

Set Value			Result				
rp	[HL]	[HL+1]	rp	SC			
				N	V	C	Z
3521H	64H	E9H	E964H	–	–	–	–

LD rp, [IX] *Load location [IX] into rp reg.* *5 cycles*

Function **rp(L) ← [IX], rp(H) ← [IX+1]**

Loads the content of the data memory that has been address specified by the IX register into the rp register (BA/HL/IX/IY) as the lower byte and loads the content of the following address as the upper byte.
The content of the XP register becomes the page address of the data memory (MODEL2/3).

Mode Src: Register indirect
Dst: Register direct

Example

Set Value			Result				
rp	[IX]	[IX+1]	rp	SC			
				N	V	C	Z
3521H	64H	E9H	E964H	–	–	–	–

Code

MSB … LSB

1	1	0	0	1	1	1	1	CFH
1	1	0	1	0	0	rp		D0H–D3H✻

✻

rp		Mnemonic	Code
BA	00	LD BA, [IX]	D0H
HL	01	LD HL, [IX]	D1H
IX	10	LD IX, [IX]	D2H
IY	11	LD IY, [IX]	D3H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

LD rp, [IY] *Load location [IY] into rp reg.* *5 cycles*

Function **rp(L) ← [IY], rp(H) ← [IY+1]**

Loads the content of the data memory that has been address specified by the IY register into the rp register (BA/HL/IX/IY) as the lower byte and loads the content of the following address as the upper byte.
The content of the YP register becomes the page address of the data memory (MODEL2/3).

Mode Src: Register indirect
Dst: Register direct

Example

Set Value			Result				
rp	[IY]	[IY+1]	rp	SC			
				N	V	C	Z
3521H	64H	E9H	E964H	–	–	–	–

Code

MSB … LSB

1	1	0	0	1	1	1	1	CFH
1	1	0	1	1	0	rp		D8H–DBH✻

✻

rp		Mnemonic	Code
BA	00	LD BA, [IY]	D8H
HL	01	LD HL, [IY]	D9H
IX	10	LD IX, [IY]	DAH
IY	11	LD IY, [IY]	DBH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

LD rp, [SP+dd] |||||||| *Load location [SP + dd] into rp reg.* |||||||||||||||||||||||||||||||||||||||||| *6 cycles* |||

Function **rp(L) ← [SP+dd], rp(H) ← [SP+dd+1]**

Loads the content of the data memory that has been address specified by the sum of the content of the SP and the displacement dd into the rp register (BA/HL/IX/IY) as the lower byte and loads the content of the following address as the upper byte.
The displacement dd is handled as signed data and the range is -128 to 127.

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	1	1	1	0	0	rp		70H–73H✻
			d	d				dd

✻

rp		Mnemonic	Code
BA	00	LD BA, [SP+dd]	70H
HL	01	LD HL, [SP+dd]	71H
IX	10	LD IX, [SP+dd]	72H
IY	11	LD IY, [SP+dd]	73H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Src: Register indirect with displacement
Dst: Register direct

Example

Set Value			Result				
rp	[SP+dd]	[SP+dd+1]	rp	SC			
				N	V	C	Z
3521H	64H	E9H	E964H	–	–	–	–

MLT |||||||||||||||||||||||||||||||||||||| *Multiply* |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *12 cycles* |||

Function **HL ← L * A**

Multiplies the content of the L register by the content of the A register and stores the result in the HL register.

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	1	0	1	1	0	0	0	D8H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	0	0	↕

Mode Implide (Register direct)

Example

Set Value		Result				
L	A	HL	SC			
			N	V	C	Z
00H	64H	0000H	0	0	0	1
64H	58H	2260H	0	0	0	0
A5H	93H	5EBFH	0	0	0	0
C8H	A5H	80E8H	1	0	0	0

Note This instruction cannot be used in the MODEL0/2.

NEG r ||||||||||||||||||||||||||||||| *Negate r reg.* |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *3 cycles* |||

Function **r ← 0 - r**

Subtracts the content of the r register (A/B) from 0 (creates two compliment) and stores the result in the r register.

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	0	0	1	0	r	A4H/A5H✻

✻

r		Mnemonic	Code
A	0	NEG A	A4H
B	1	NEG B	A5H

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Register direct

Example

	Set Value	Result				
	r	r	SC			
			N	V	C	Z
• *D=0, U=0*	57H	A9H	1	0	1	0
	00H	00H	0	0	0	1
	2BH	D5H	1	0	1	0
	80H	80H	1	1	1	0
• *D=1, U=0*	57	43	0	0	1	0
• *D=1, U=1*	57	03	0	0	1	0

NEG [BR:ll] |||||||||||||||| *Negate location [BR:ll]* ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *5 cycles* |||

Function **[BR:*ll*] ← 0 - [BR:*ll*]**

Subtracts the content of the data memory from 0 (creates two compliment) and stores the result in the same address. The data memory address has been specified by the content of the BR register (upper byte specification) and the 8-bit absolute address *ll* (lower byte specification).
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	0	0	1	1	0	A6H
			l	*l*				*ll*

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode 8-bit absolute

Example

	Set Value	Result				
	[BR:*ll*]	[BR:*ll*]	SC			
			N	V	C	Z
• *D=0, U=0*	57H	A9H	1	0	1	0
	00H	00H	0	0	0	1
	2BH	D5H	1	0	1	0
	80H	80H	1	1	1	0
• *D=1, U=0*	57	43	0	0	1	0
• *D=1, U=1*	57	03	0	0	1	0

NEG [HL] |||||||||||||||||||||| *Negate location [HL]* ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *4 cycles* |||

Function **[HL] ← 0 - [HL]**

Subtracts the content of the data memory that has been address specified by the HL register from 0 (creates two compliment) and stores the result in that address.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	0	0	1	1	1	A7H

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Register indirect

Example

	Set Value	Result				
	[HL]	[HL]	SC			
			N	V	C	Z
• *D=0, U=0*	57H	A9H	1	0	1	0
	00H	00H	0	0	0	1
	2BH	D5H	1	0	1	0
	80H	80H	1	1	1	0
• *D=1, U=0*	57	43	0	0	1	0
• *D=1, U=1*	57	03	0	0	1	0

NOP — *No Operation* — *2 cycles*

Function **No Operation**

Expends 2 cycles without doing an operation that otherwise exerts an affect. The program counter (PC) is incremented (+1).

Code

MSB							LSB	
1	1	1	1	1	1	1	1	FFH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

OR A, r — *Logical OR of r reg. and A reg.* — *2 cycles*

Function **A ← A ∨ r**

Takes a logical sum of the content of the r register (A/B) and the content of the A register and stores the result in the A register.

Code

MSB							LSB	
0	0	1	0	1	0	0	r	28H/29H✻

✻

r		Mnemonic	Code
A	0	OR A, A	28H
B	1	OR A, B	29H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Register direct
Dst: Register direct

Example

Set Value		Result				
A	B	A	SC			
			N	V	C	Z
32H	6CH	7EH	0	–	–	0
86H	41H	C7H	1	–	–	0

OR A, #nn — *Logical OR of immediate data nn and A reg.* — *2 cycles*

Function **A ← A ∨ nn**

Takes a logical sum of the 8-bit immediate data nn and the content of the A register and stores the result in the A register.

Code

MSB							LSB	
0	0	1	0	1	0	1	0	2AH
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result				
A	nn	A	SC			
			N	V	C	Z
32H	6CH	7EH	0	–	–	0
86H	41H	C7H	1	–	–	0

OR A, [BR:ll] *Logical OR of location [BR:ll] and A reg. 3 cycles*

Function **A ← A ∨ [BR:*ll*]**

Takes a logical sum of the content of the data memory and the content of the A register and stores the result in the A register. The data memory address has been specified by the content of the BR register (upper byte specification) and the 8-bit absolute address *ll* (lower byte specification).
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB ... LSB

0	0	1	0	1	1	0	0	2CH
			l	*l*				*ll*

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: 8-bit absolute
Dst: Register direct

Example

Set Value		Result	SC			
A	[BR:*ll*]	A	N	V	C	Z
32H	6CH	7EH	0	–	–	0
86H	41H	C7H	1	–	–	0

OR A, [hhll] *Logical OR of location [hhll] and A reg. 4 cycles*

Function **A ← A ∨ [hh*ll*]**

Takes a logical sum of the content of the data memory that has been address specified by the 16-bit absolute address hh*ll* and the content of the A register and stores the result in the A register.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB ... LSB

0	0	1	0	1	1	0	1	2DH
			l	*l*				*ll*
			h	h				hh

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: 16-bit absolute
Dst: Register direct

Example

Set Value		Result	SC			
A	[hh*ll*]	A	N	V	C	Z
32H	6CH	7EH	0	–	–	0
86H	41H	C7H	1	–	–	0

OR A, [HL] *Logical OR of location [HL] and A reg. 2 cycles*

Function **A ← A ∨ [HL]**

Takes a logical sum of the content of the data memory that has been address specified by the HL register and the content of the A register and stores the result in the A register.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB ... LSB

0	0	1	0	1	0	1	1	2BH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Register indirect
Dst: Register direct

Example

Set Value		Result	SC			
A	[HL]	A	N	V	C	Z
32H	6CH	7EH	0	–	–	0
86H	41H	C7H	1	–	–	0

OR A, [ir] |||||||||||||||||||||| *Logical OR of location [ir reg.] and A reg.* ||||||||||||||||||||||||||||||||| *2 cycles* |||

Function **A ← A ∨ [ir]**

Takes a logical sum of the content of the data memory that has been address specified by the ir register (IX/IY) and the content of the A register and stores the result in the A register.
The content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
0	0	1	0	1	1	1	ir	2EH/2FH✻

✻

ir		Mnemonic	Code
IX	0	OR A, [IX]	2EH
IY	1	OR A, [IY]	2FH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Register indirect
Dst: Register direct

Example

Set Value		Result				
A	[Ir]	A	SC			
			N	V	C	Z
32H	6CH	7EH	0	–	–	0
86H	41H	C7H	1	–	–	0

OR A, [ir+dd] |||||||||||| *Logical OR of location [ir reg. + dd] and A reg.* ||||||||||||||||||||||| *4 cycles* |||

Function **A ← A ∨ [ir+dd]**

Takes a logical sum of the content of the data memory and the content of the A register and stores the result in the A register. The data memory address has been specified by the sum of the content of the ir register (IX/IY) and the displacement dd.
The displacement dd is handled as signed data and the range is -128 to 127.
The content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	1	0	1	0	0	ir	28H/29H✻
			d	d				dd

✻

ir		Mnemonic	Code
IX	0	OR A, [IX+dd]	28H
IY	1	OR A, [IY+dd]	29H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Register indirect with displacement
Dst: Register direct

Example

Set Value		Result				
A	[Ir+dd]	A	SC			
			N	V	C	Z
32H	6CH	7EH	0	–	–	0
86H	41H	C7H	1	–	–	0

OR A, [ir+L] |||||||||||||||| *Logical OR location [ir reg. + L] and A reg.* |||||||||||||||||||||||||||||| *4 cycles* |||

Function **A ← A ∨ [ir+L]**

Takes a logical sum of the content of the data memory and the content of the A register and stores the result in the A register. The data memory address has been specified by the sum of the content of the ir register (IX/IY) and the content of the L register.
The content of the L register is handled as signed data and the range is -128 to 127.
The content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	1	0	1	0	1	ir	2AH/2BH✻

✻

ir		Mnemonic	Code
IX	0	OR A, [IX+L]	2AH
IY	1	OR A, [IY+L]	2BH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Register indirect with index register
Dst: Register direct

Example

Set Value		Result				
A	[Ir+L]	A	SC			
			N	V	C	Z
32H	6CH	7EH	0	–	–	0
86H	41H	C7H	1	–	–	0

OR B, #nn |||||||||||||||||||| *Logical OR of immediate data nn and B reg.* ||||||||||||||||||||||||||||||| *3 cycles* |||

Function **B ← B ∨ nn**

Takes a logical sum of the 8-bit immediate data nn and the content of the B register and stores the result in the B register.

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	1	0	1	0	0	B4H
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result				
B	nn	B	SC			
			N	V	C	Z
32H	6CH	7EH	0	–	–	0
86H	41H	C7H	1	–	–	0

OR L, #nn |||||||||||||||||||| *Logical OR of immediate data nn and L reg.* |||||||||||||||||||||||||||||||| *3 cycles* |||

Function **L ← L ∨ nn**

Takes a logical sum of the 8-bit immediate data nn and the content of the L register and stores the result in the L register.

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	1	0	1	0	1	B5H
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result				
L	nn	L	SC			
			N	V	C	Z
32H	6CH	7EH	0	–	–	0
86H	41H	C7H	1	–	–	0

OR H, #nn *Logical OR of immediate data nn and H reg. 3 cycles*

Function **H ← H ∨ nn**

Takes a logical sum of the 8-bit immediate data nn and the content of the H register and stores the result in the H register.

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	1	0	1	1	0	B6H
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result				
H	nn	H	SC			
			N	V	C	Z
32H	6CH	7EH	0	–	–	0
86H	41H	C7H	1	–	–	0

OR SC, #nn *Logical OR of immediate data nn and SC 3 cycles*

Function **SC ← SC ∨ nn**

Takes a logical sum of the 8-bit immediate data nn and the content of the system condition flag (SC) and sets the result in the system condition flag (SC).

Code

MSB							LSB	
1	0	0	1	1	1	0	1	9DH
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
↑	↑	↑	↑	↑	↑	↑	↑

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result							
SC	nn	SC							
		I1	I0	U	D	N	V	C	Z
32H	6CH	0	1	1	1	1	1	1	0
86H	41H	1	1	0	0	0	1	1	1

OR [BR:ll], #nn *Logical OR of immediate data nn and location [BR:ll] 5 cycles*

Function **[BR:*ll*] ← [BR:*ll*] ∨ nn**

Takes a logical sum of the 8-bit immediate data and the content of the data memory and stores the result in that address. The data memory address has been specified by the content of the BR register (upper byte specification) and the 8-bit absolute address *ll* (lower byte specification).
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	1	1	0	0	1	D9H
			l	*l*				*ll*
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Immediate data
Dst: 8-bit absolute

Example

Set Value		Result				
[BR:*ll*]	nn	[BR:*ll*]	SC			
			N	V	C	Z
32H	6CH	7EH	0	–	–	0
86H	41H	C7H	1	–	–	0

OR [HL], A |||||||||||||||||| *Logical OR of A reg. and location [HL]* |||||||||||||||||||||||||||||||||||| *4 cycles* |||

Function **[HL] ← [HL] ∨ A**

Takes a logical sum of the content of the A register and the data memory that has been address specified by the HL register and stores the result in that address.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	1	0	1	1	0	0	2CH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Register direct
Dst: Register indirect

Example

Set Value		Result				
[HL]	A	[HL]	SC			
			N	V	C	Z
32H	6CH	7EH	0	–	–	0
86H	41H	C7H	1	–	–	0

OR [HL], #nn |||||||||||| *Logical OR of immediate data nn and location [HL]* |||||||||||||||| *5 cycles* |||

Function **[HL] ← [HL] ∨ nn**

Takes a logical sum of the 8-bit immediate data nn and the data memory that has been address specified by the HL register and stores the result in that address.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	1	0	1	1	0	1	2DH
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Immediate data
Dst: Register indirect

Example

Set Value		Result				
[HL]	nn	[HL]	SC			
			N	V	C	Z
32H	6CH	7EH	0	–	–	0
86H	41H	C7H	1	–	–	0

OR [HL], [ir] ||||||||||||||| *Logical OR of location [ir reg.] to location [HL]* |||||||||||||||||||| *5 cycles* |||

Function **[HL] ← [HL] ∨ [ir]**

Takes a logical sum of the content of the data memory that has been address specified by the ir register (IX/IY) and the data memory that has been address specified by the HL register and stores the result in data memory [HL].
The content of the EP register becomes the page address of the data memory [HL] and the content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory [ir] (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	1	0	1	1	1	ir	2EH/2FH✻

✻

ir		Mnemonic	Code
IX	0	OR [HL], [IX]	2EH
IY	1	OR [HL], [IY]	2FH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

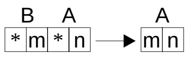
Mode Src: Register indirect
Dst: Register indirect

Example

Set Value		Result				
[HL]	[ir]	[HL]	SC			
			N	V	C	Z
32H	6CH	7EH	0	–	–	0
86H	41H	C7H	1	–	–	0

PACK |||||||||||||||||||||||||||||| *Pack BA reg. to A reg.* ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *2 cycles* |||

Function



Packs the content of the BA register and stores it in the A register. The upper 4 bits of the A register are substituted for the content of the lower 4 bits of the B register.

Code

MSB LSB

1	1	0	1	1	1	1	0	DEH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Implide (Register direct)

Example

Set Value	Result					
			SC			
BA	A	B	N	V	C	Z
38C4H	84H	38H	–	–	–	–

POP r |||||||||||||||||||||||||||||| *Pop top of stack into r reg.* ||||||||||||||||||||||||||||||||||||||||||||||||||||||| *3 cycles* |||

Function **r ← [SP], SP ← SP + 1**

Loads the content of the address indicated by the stack pointer (SP) into the r register (A/B/L/H) and increments (+1) the SP.

Code

MSB LSB

1	1	0	0	1	1	1	1	CFH
1	0	1	1	0	1	r		B4H–B7H✻

✻

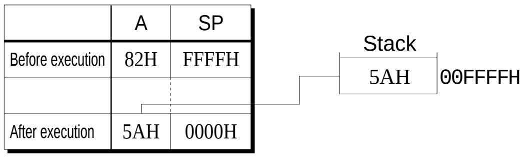
r		Mnemonic	Code
A	00	POP A	B4H
B	01	POP B	B5H
L	10	POP L	B6H
H	11	POP H	B7H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Register direct

Example Executes "POP A"



	A	SP
Before execution	82H	FFFFH
After execution	5AH	0000H

POP rp |||||||||||||||||||||||||||| *Pop top of stack into rp reg.* ||||||||||||||||||||||||||||||||||||||||||||||||||||| *3 cycles* |||

Function **rp(L) ← [SP], rp(H) ← [SP+1],**
SP ← SP + 2

Loads the content of the 2 bytes from the address indicated by the stack pointer (SP) in the sequence of the lower byte, then the upper byte of the rp register (BA/HL/IX/IY) and adds 2 to the SP.

Code

MSB LSB

1	0	1	0	1	0	rp		A8H–ABH✻

✻

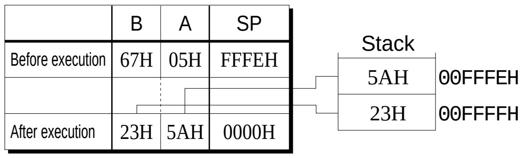
rp		Mnemonic	Code
BA	00	POP BA	A8H
HL	01	POP HL	A9H
IX	10	POP IX	AAH
IY	11	POP IY	ABH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Register direct

Example Executes "POP BA"



	B	A	SP
Before execution	67H	05H	FFFEH
After execution	23H	5AH	0000H

POP BR |||||||||||||||||||||||||| *Pop top of stack into BR reg.* |||||||||||||||||||||||||||||||||||||||||||||||||||||| *2 cycles* |||

Function **BR ← [SP], SP ← SP + 1**

Loads the content of the address indicated by the stack pointer (SP) into the BR register and increments (+1) the SP.

Code

MSB							LSB	
1	0	1	0	1	1	0	0	ACH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Register direct

Example

	BR	SP
Before execution	82H	FFFFH
After execution	5AH	0000H

Stack	
5AH	00FFFFH

POP EP |||||||||||||||||||||||||| *Pop top of stack into EP reg.* |||||||||||||||||||||||||||||||||||||||||||||||||||||| *2 cycles* |||

Function **EP ← [SP], SP ← SP + 1**

Loads the content of the address indicated by the stack pointer (SP) into the EP register and increments (+1) the SP.

Code

MSB							LSB	
1	0	1	0	1	1	0	1	ADH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Register direct

Example

	EP	SP
Before execution	82H	FFFFH
After execution	5AH	0000H

Stack	
5AH	00FFFFH

Note This instruction cannot be used in the MODEL0/1.

POP IP |||||||||||||||||||||||||| *Pop top of stack into IP reg.* |||||||||||||||||||||||||||||||||||||||||||||||||||||| *3 cycles* |||

Function **YP ← [SP], XP ← [SP+1], SP ← SP + 2**

Loads the content of the 2 bytes from the address indicated by the stack pointer (SP) sequentially into the YP register and XP register and adds 2 to the SP.

Code

MSB							LSB	
1	0	1	0	1	1	1	0	AEH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Register direct

Example Executes "POP IP"

	XP	YP	SP
Before execution	67H	05H	FFFEH
After execution	23H	5AH	0000H

Stack	
5AH	00FFFEH
23H	00FFFFH

Note This instruction cannot be used in the MODEL0/1.

POP SC |||||||||||||||||||||||||| *Pop top of stack into SC* |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *2 cycles* |||

Function **SC ← [SP], SP ← SP + 1**

Sets the content of the address indicated by the stack pointer (SP) to the system condition flag (SC) and increments (+1) the SP.

Code

MSB							LSB	
1	0	1	0	1	1	1	1	AFH

Flag

I1	I0	U	D	N	V	C	Z
↕	↕	↕	↕	↕	↕	↕	↕

Mode Register direct

Example

	SC	SP
Before execution	82H	FFFFH
After execution	5AH	0000H

Stack	
5AH	00FFFFH

POP ALE *Pop all registers including expand registers* *14 cycles*

Function **POP IP, EP, BR, IY, IX, HL, BA**

Pops the content of the address indicated by the stack pointer (SP) in the sequence of IP (XP/YP), EP, BR, IY, IX, HL and BA register. It can once return the content of the register that has been batch evacuated by the "PUSH ALE" instruction. 12, which corresponds to the number of bytes that have been popped, is added to the SP.

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
1	0	1	1	1	1	0	1	BDH

Flag

I1	I0	U	D	N	V	C	Z
↕	↕	↕	↕	↕	↕	↕	↕

Mode Implide (Register direct)

Example In case of SP = FFF4H, when "POP ALE" is executed, the content of the stack as shown in the below figure returns to the respective register and becomes SP = 0000H.

00FFFAH	IX(L)	00FFF4H	YP
00FFFBH	IX(H)	00FFF5H	XP
00FFFCH	L	00FFF6H	EP
00FFFDH	H	00FFF7H	BR
00FFFEH	A	00FFF8H	IY(L)
00FFFFH	B	00FFF9H	IY(H)

Note This instruction cannot be used in the MODEL0/1.

POP ALL *Pop all registers* *11 cycles*

Function **POP BR, IY, IX, HL, BA**

Pops the content of the address indicated by the stack pointer (SP) in the sequence of BR, IY, IX, HL and BA register. It can once return the content of the register that has been batch evacuated by the "PUSH ALL" instruction. 9, which corresponds to the number of bytes that have been popped, is added to the SP.

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
1	0	1	1	1	1	0	0	BCH

Flag

I1	I0	U	D	N	V	C	Z
↕	↕	↕	↕	↕	↕	↕	↕

Mode Implide (Register direct)

Example In case of SP = FFF7H, when "POP ALL" is executed, the content of the stack as shown in the below figure returns to the respective register and becomes SP = 0000H.

00FFFCH	L	00FFF7H	BR
00FFFDH	H	00FFF8H	IY(L)
00FFFEH	A	00FFF9H	IY(H)
00FFFFH	B	00FFFAH	IX(L)
		00FFFBH	IX(H)

PUSH r ||||||||||||||||||||||||||| *Push r reg. onto stack* ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *3 cycles* |||

Function **[SP-1] ← r, SP ← SP - 1**

Stores the content of the r register (A/B/L/H) in the address indicated by the content resulting from the subtraction of 1 from the stack pointer (SP).
The SP is decremented (-1).

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
1	0	1	1	0	0	r		B0H–B3H✻

✻

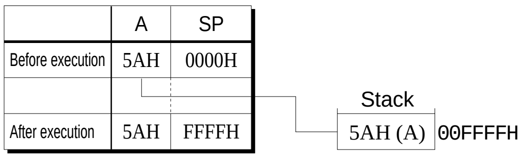
r		Mnemonic	Code
A	00	PUSH A	B0H
B	01	PUSH B	B1H
L	10	PUSH L	B2H
H	11	PUSH H	B3H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Register direct

Example Executes "PUSH A"



	A	SP
Before execution	5AH	0000H
After execution	5AH	FFFFH

PUSH rp |||||||||||||||||||||||| *Push rp reg. onto stack* ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *4 cycles* |||

Function **[SP-1] ← rp(L), [SP-2] ← rp(H),**
SP ← SP - 2

Stores the content of the rp register (BA/HL/IX/IY) in the address indicated by the content resulting from the subtraction of 1 from the stack pointer (SP) and in the one preceding address in the sequence of the upper byte, then the lower byte.
2 is subtracted from the SP.

Code

MSB							LSB	
1	0	1	0	0	0	rp		A0H–A3H✻

✻

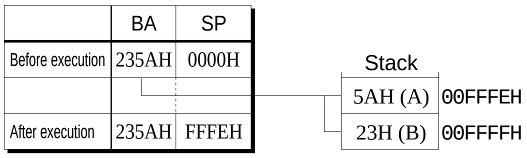
rp		Mnemonic	Code
BA	00	PUSH BA	A0H
HL	01	PUSH HL	A1H
IX	10	PUSH IX	A2H
IY	11	PUSH IY	A3H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Register direct

Example Executes "PUSH BA"



	BA	SP
Before execution	235AH	0000H
After execution	235AH	FFFEH

PUSH BR |||||||||||||||||||| *Push BR reg. onto stack* |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *3 cycles* |||

Function **[SP-1] ← BR, SP ← SP - 1**

Stores the content of the BR register in the address indicated by the content resulting from the subtraction of 1 from the stack pointer (SP).
The SP is decremented (-1).

Code

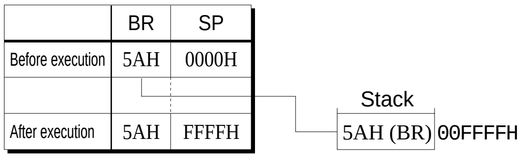
MSB							LSB	
1	0	1	0	0	1	0	0	A4H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Register direct

Example



	BR	SP
Before execution	5AH	0000H
After execution	5AH	FFFFH

Stack

5AH (BR)	00FFFFH

PUSH EP |||||||||||||||||||| *Push EP reg. onto stack* |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *3 cycles* |||

Function **[SP-1] ← EP, SP ← SP - 1**

Stores the content of the EP register in the address indicated by the content resulting from the subtraction of 1 from the stack pointer (SP).
The SP is decremented (-1).

Code

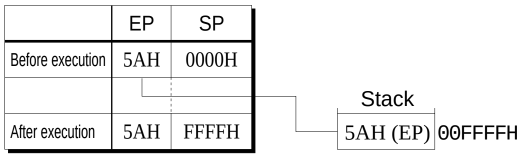
MSB							LSB	
1	0	1	0	0	1	0	1	A5H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Register direct

Example



	EP	SP
Before execution	5AH	0000H
After execution	5AH	FFFFH

Stack

5AH (EP)	00FFFFH

Note This instruction cannot be used in the MODEL0/1.

PUSH IP |||||||||||||||||||||| *Push IP reg. onto stack* |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *4 cycles* |||

Function **[SP-1] ← XP, [SP-2] ← YP, SP ← SP - 2**

Stores the content of the XP register and the YP register in the address indicated by the content resulting from the subtraction of 1 from the stack pointer (SP), and the one preceding address.
2 is subtracted from the SP.

Code

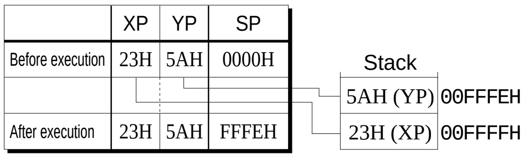
MSB							LSB	
1	0	1	0	0	1	1	0	A6H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Register direct

Example Executes "PUSH IP"



	XP	YP	SP
Before execution	23H	5AH	0000H
After execution	23H	5AH	FFFEH

Stack

5AH (YP)	00FFFEH
23H (XP)	00FFFFH

Note This instruction cannot be used in the MODEL0/1.

PUSH SC |||||||||||||||||||| *Push SC onto stack* |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *3 cycles* |||

Function **[SP-1] ← SC, SP ← SP - 1**

Stores the content of the system condition flag (SC) in the address indicated by the content resulting from the subtraction of 1 from the stack pointer (SP).
The SP is decremented (-1).

Code

MSB LSB

1	0	1	0	0	1	1	1	A7H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Register direct

Example

	SC	SP
Before execution	5AH	0000H
After execution	5AH	FFFFH

Stack

5AH (SC)	00FFFFH

PUSH ALE ||||||||||||||||| *Push all registers including expand registers* ||||||||||||||||||||||||| *15 cycles* |||

Function **PUSH BA, HL, IX, IY, BR, EP, IP**

Pushes the content of the registers in the sequence of BA, HL, IX, IY, BR, EP and IP (XP/YP) from the address indicated by the stack pointer (SP) toward the lower address. 12, which corresponds to the number of bytes pushed, is subtracted from the SP.
Evacuates the registers for MODEL2/3 use.

Code

MSB LSB

1	1	0	0	1	1	1	1	CFH
1	0	1	1	1	0	0	1	B9H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Implide (Register direct)

Example In case of SP = 0000H, when "PUSH ALE" is executed, the registers are stacked as shown in the below figure and becomes SP = FFF4H.

00FFFAH	IX(L)	00FFF4H	YP
00FFFBH	IX(H)	00FFF5H	XP
00FFFCH	L	00FFF6H	EP
00FFFDH	H	00FFF7H	BR
00FFFEH	A	00FFF8H	IY(L)
00FFFFH	B	00FFF9H	IY(H)

Note This instruction cannot be used in the MODEL0/1.

PUSH ALL ||||||||||||||||| *Push all registers* |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *12 cycles* |||

Function **PUSH BA, HL, IX, IY, BR**

Pushes the content of the registers in the sequence of BA, HL, IX, IY and BR from the address indicated by the stack pointer (SP) toward the lower address. 9, which corresponds to the number of bytes pushed, is subtracted from the SP.
Evacuates the registers for MODEL0/1 use.

Code

MSB LSB

1	1	0	0	1	1	1	1	CFH
1	0	1	1	1	0	0	0	B8H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Mode Implide (Register direct)

Example In case of SP = 0000H, when "PUSH ALL" is executed, the registers are stacked as shown in the below figure and becomes SP = FFF7H.

00FFFCH	L	00FFF7H	BR
00FFFDH	H	00FFF8H	IY(L)
00FFFEH	A	00FFF9H	IY(H)
00FFFFH	B	00FFFAH	IX(L)
		00FFFBH	IX(H)

RET |||||||||||||||||||||||||||||||||||| *Return from subroutine* |||||||||||||||||||||||||||||||||||||||||| *3(MIN)/4(MAX) cycles* |||

Function **<MODEL0/1, MODEL2/3-minimum>**
PC(L)←[SP], PC(H)←[SP+1], SP←SP+2

<MODEL2/3-maximum>
PC(L)←[SP], PC(H)←[SP+1],
CB←[SP+2], NB←CB, SP←SP+3

Loads the 2 bytes from the address indicated by the stack pointer (SP) in the sequence of the lower byte then the upper byte of the program counter (PC), then returns from the subroutine.
In the maximum mode of the MODEL2/3 it loads the following 1 byte into the CB as the bank address and returns it to the originally called bank. It simultaneously also resets that bank address to NB.
The returned number of bytes (minimum mode: 2 bytes, maximum mode: 3 bytes) are added to the SP.

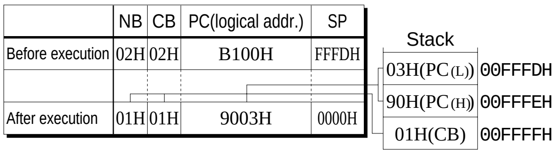
Code

MSB							LSB	
1	1	1	1	1	0	0	0	F8H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Example In the maximum mode of the MODEL2/3, it executes the "RET" instruction in the physical address 013100H.



	NB	CB	PC(logical addr.)	SP
Before execution	02H	02H	B100H	FFFDH
After execution	01H	01H	9003H	0000H

In the above example it returns to the physical address 009003H.
Since CB is not stacked in the minimum mode of the MODEL2/3, CB and NB are not changed.
There are no NB and CB in the MODEL0/1.

Note Use the following "RETE" instruction to return from an exception processing routine.

RETE |||||||||||||||||||||||||||||||| *Return from exception processing routine* |||||||||||||| *4(MIN)/5(MAX) cycles* |||

Function **<MODEL0/1, MODEL2/3-minimum>**
SC←[SP], PC(L)←[SP+1],
PC(H)←[SP+2], SP←SP+3

<MODEL2/3-maximum>
SC←[SP], PC(L)←[SP+1],
PC(H)←[SP+2], CB←[SP+3], NB←CB,
SP←SP+4

Loads the 3 bytes from the address indicated by the stack pointer (SP) in the sequence of the system condition flag (SC) then the lower byte and the upper byte of the program counter (PC), then returns from the subroutine.
In the maximum mode of the MODEL2/3 it loads the following 1 byte into the CB as the bank address and returns it to the originally called bank. It simultaneously also resets that bank address to NB.
The returned number of bytes (minimum mode: 3 bytes, maximum mode: 4 bytes) are added to the SP.

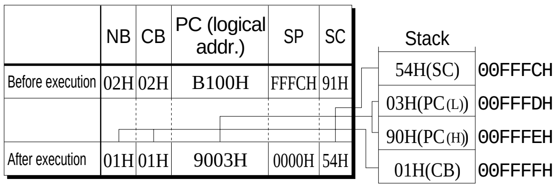
Code

MSB							LSB	
1	1	1	1	1	0	0	1	F9H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Example In the maximum mode of the MODEL2/3, it executes the "RETE" instruction in the physical address 013100H.



	NB	CB	PC (logical addr.)	SP	SC
Before execution	02H	02H	B100H	FFFCH	91H
After execution	01H	01H	9003H	0000H	54H

In the above example it returns to the physical address 009003H.
Since CB is not stacked in the minimum mode of the MODEL2/3, CB and NB are not changed.
There are no NB and CB in the MODEL0/1.

RETS |||||||||||||||||||||||||||||||| *Return from subroutine then skip 2 bytes* ||||||||||||||| *5(MIN)/6(MAX) cycles* |||

Function **<MODEL0/1, MODEL2/3-minimum>**
PC(L)←[SP], PC(H)←[SP+1], SP←SP+2,
PC←PC+2

<MODEL2/3-maximum>
PC(L)←[SP], PC(H)←[SP+1],
CB←[SP+2], NB←CB, SP←SP+3,
PC←PC+2

Skips the 2-byte instruction of the returned address, following execution of the "RET" instruction.

Code

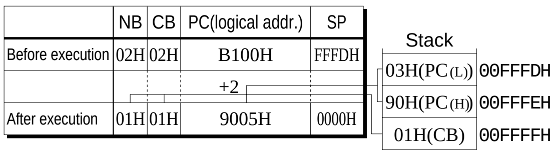
MSB LSB

1	1	1	1	1	0	1	0	FAH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

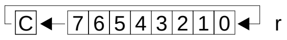
Example In the maximum mode of the MODEL2/3, it executes the "RETS" instruction in the physical address 013100H.



In the above example it returns to the physical address 009005H.
Since CB is not stacked in the minimum mode of the MODEL2/3, CB and NB are not changed.
There are no NB and CB in the MODEL0/1.

RL r |||||||||||||||||||||||||||||||||| *Rotate left r reg. with carry* |||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *3 cycles* |||

Function



Rotates the content of the r register (A/B) the equivalent of 1 bit to the left, including the carry (C). The content of the carry (C) moves to bit 0 of the register and bit 7 of the register moves to the carry (C).

Code

MSB LSB

1	1	0	0	1	1	1	0	CEH
1	0	0	1	0	0	0	r	90H/91H✻

✻

r		Mnemonic	Code
A	0	RL A	90H
B	1	RL B	91H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	↕	↕

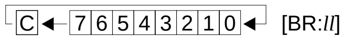
Mode Register direct

Example

Set Value		Result				
r	C	r	SC			
			N	V	C	Z
83H	0	06H	0	–	1	0
4CH	0	98H	1	–	0	0
A2H	1	45H	0	–	1	0

RL [BR:ll] Rotate left location [BR:ll] with carry 5 cycles

Function



Rotates the content of the data memory the equivalent of 1 bit to the left, including the carry (C). The data memory address has been specified by the content of the BR register (upper byte specification) and the 8-bit absolute address *ll* (lower byte specification). The content of the carry (C) moves to bit 0 of the data and bit 7 of the data moves to the carry (C).
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	1	0	0	1	0	92H
			l	*l*				*ll*

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	↕	↕

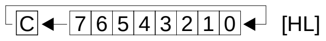
Mode 8-bit absolute

Example

Set Value		Result				
[BR:*ll*]	C	[BR:*ll*]	SC			
			N	V	C	Z
83H	0	06H	0	–	1	0
4CH	0	98H	1	–	0	0
A2H	1	45H	0	–	1	0

RL [HL] Rotate left location [HL] with carry 4 cycles

Function



Rotates the content of the data memory that has been address specified by the HL register the equivalent of 1 bit to the left, including the carry (C). The content of the carry (C) moves to bit 0 of the data and bit 7 of the data moves to the carry (C).
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	1	0	0	1	1	93H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	↕	↕

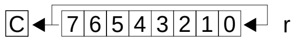
Mode Register indirect

Example

Set Value		Result				
[HL]	C	[HL]	SC			
			N	V	C	Z
83H	0	06H	0	–	1	0
4CH	0	98H	1	–	0	0
A2H	1	45H	0	–	1	0

RLC r Rotate left r reg. circular 3 cycles

Function



Rotates the content of the r register (A/B) the equivalent of 1 bit to the left. Bit 7 of the register moves to bit 0 and carry (C).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	1	0	1	0	r	94H/95H✻

✻

r		Mnemonic	Code
A	0	RLC A	94H
B	1	RLC B	95H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	↕	↕

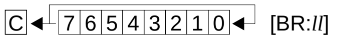
Mode Register direct

Example

Set Value		Result				
r	C	r	SC			
			N	V	C	Z
E3H	0	C7H	1	–	1	0
3BH	1	76H	0	–	0	0

RLC [BR:ll] ||||||||||||||||| *Rotate left location [BR:ll] circular* ||||||||||||||||||||||||||||||||||||||||||||| *5 cycles* |||

Function



Rotates the content of the data memory the equivalent of 1 bit to the left. The data memory address has been specified by the content of the BR register (upper byte specification) and the 8-bit absolute address *ll* (lower byte specification).
Bit 7 of the data moves to bit 0 and carry (C).
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	1	0	1	1	0	96H
			l	*l*				*ll*

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	↕	↕

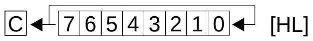
Mode 8-bit absolute

Example

Set Value		Result				
[BR:*ll*]	C	[BR:*ll*]	SC			
			N	V	C	Z
E3H	0	C7H	1	–	1	0
3BH	1	76H	0	–	0	0

RLC [HL] ||||||||||||||||||||||| *Rotate left location [HL] circular* ||||||||||||||||||||||||||||||||||||||||||||||| *4 cycles* |||

Function



Rotates the content of the data memory that has been address specified by the HL register the equivalent of 1 bit to the left. Bit 7 of the data moves to bit 0 and carry (C).
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	1	0	1	1	1	97H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	↕	↕

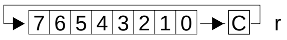
Mode Register indirect

Example

Set Value		Result				
[HL]	C	[HL]	SC			
			N	V	C	Z
E3H	0	C7H	1	–	1	0
3BH	1	76H	0	–	0	0

RR r ||||||||||||||||||||||||||||||||||||| *Rotate right r reg. with carry* ||||||||||||||||||||||||||||||||||||||||||||||||||||||| *3 cycles* |||

Function



Rotates the content of the r register (A/B) the equivalent of 1 bit to the right, including the carry (C). The content of the carry (C) moves to bit 7 of the register and bit 0 of the register moves to the carry (C).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	1	1	0	0	r	98H/99H✻

✻

r		Mnemonic	Code
A	0	RR A	98H
B	1	RR B	99H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	↕	↕

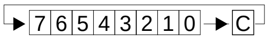
Mode Register direct

Example

Set Value		Result				
r	C	r	SC			
			N	V	C	Z
7EH	0	3FH	0	–	0	0
51H	0	28H	0	–	1	0
D4H	1	EAH	1	–	0	0

RR [BR:ll] |||||||||||||||||||| *Rotate right location [BR:ll] with carry* |||||||||||||||||||||||||||||||||||| *5 cycles* |||

Function



[BR:*ll*]

Rotates the content of the data memory the equivalent of 1 bit to the right, including the carry (C). The data memory address has been specified by the content of the BR register (upper byte specification) and the 8-bit absolute address *ll* (lower byte specification). The content of the carry (C) moves to bit 7 of the data and bit 0 of the data moves to the carry (C).
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	1	1	0	1	0	9AH
			l	*l*				*ll*

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	↕	↕

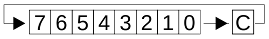
Mode 8-bit absolute

Example

Set Value		Result				
			SC			
[BR:*ll*]	C	[BR:*ll*]	N	V	C	Z
7EH	0	3FH	0	–	0	0
51H	0	28H	0	–	1	0
D4H	1	EAH	1	–	0	0

RR [HL] ||||||||||||||||||||||||| *Rotate right location [HL] with carry* |||||||||||||||||||||||||||||||||||||| *4 cycles* |||

Function



[HL]

Rotates the content of the data memory that has been address specified by the HL register the equivalent of 1 bit to the right, including the carry (C). The content of the carry (C) moves to bit 7 of the data and bit 0 of the data moves to the carry (C).
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	1	1	0	1	1	9BH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	↕	↕

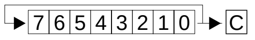
Mode Register indirect

Example

Set Value		Result				
			SC			
[HL]	C	[HL]	N	V	C	Z
7EH	0	3FH	0	–	0	0
51H	0	28H	0	–	1	0
D4H	1	EAH	1	–	0	0

RRC r |||||||||||||||||||||||||||||| *Rotate right r reg. circular* ||||||||||||||||||||||||||||||||||||||||||||||||||||||| *3 cycles* |||

Function


r

Rotates the content of the r register (A/B) the equivalent of 1 bit to the right. Bit 0 of the register moves to bit 7 and carry (C).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	1	1	1	0	r	9CH/9DH✻

✻

r		Mnemonic	Code
A	0	RRC A	9CH
B	1	RRC B	9DH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	↕	↕

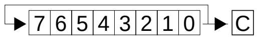
Mode Register direct

Example

Set Value		Result				
r	C	r	SC			
			N	V	C	Z
C6H	1	63H	0	–	0	0
D7H	0	EBH	1	–	1	0

RRC [BR:ll] |||||||||||||||| *Rotate right location [BR:ll] circular* ||||||||||||||||||||||||||||||||||||||| *5 cycles* |||

Function


[BR:*ll*]

Rotates the content of the data memory the equivalent of 1 bit to the right. The data memory address has been specified by the content of the BR register (upper byte specification) and the 8-bit absolute address *ll* (lower byte specification).
Bit 0 of the data moves to bit 7 and carry (C).
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	1	1	1	1	0	9EH
			l	*l*				*ll*

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	↕	↕

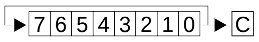
Mode 8-bit absolute

Example

Set Value		Result				
[BR:*ll*]	C	[BR:*ll*]	SC			
			N	V	C	Z
C6H	1	63H	0	–	0	0
D7H	0	EBH	1	–	1	0

RRC [HL] |||||||||||||||||||| *Rotate right location [HL] circular* |||||||||||||||||||||||||||||||||||||||||||| *4 cycles* |||

Function


[HL]

Rotates the content of the data memory that has been address specified by the HL register the equivalent of 1 bit to the right. Bit 0 of the data moves to bit 7 and carry (C).
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	1	1	1	1	1	9FH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	↕	↕

Mode Register indirect

Example

Set Value		Result				
[HL]	C	[HL]	SC			
			N	V	C	Z
C6H	1	63H	0	–	0	0
D7H	0	EBH	1	–	1	0

SBC A, r ||||||||||||||||||||||||| *Subtract with carry r reg. from A reg.* ||||||||||||||||||||||||||||||||||||||| *2 cycles* |||

Function **A ← A - r - C**
Subtracts the content of the r register (A/B) and carry (C) from the A register.

Code

MSB							LSB	
0	0	0	1	1	0	0	r	18H/19H✻

✻

r		Mnemonic	Code
A	0	SBC A, A	18H
B	1	SBC A, B	19H

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Register direct
Dst: Register direct

Example

	Set Value			Result				
	A	B	C	A	SC			
					N	V	C	Z
• *D=0, U=0*	A8H	42H	1	65H	0	1	0	0
	36H	5AH	1	DBH	1	0	1	0
• *D=1, U=0*	88	39	1	48	0	0	0	0
• *D=1, U=1*	88	39	1	08	0	0	1	0

SBC A, #nn |||||||||||||||| *Subtract with carry immediate data nn from A reg.* ||||||||||||||||||| *2 cycles* |||

Function **A ← A - nn - C**
Subtracts 8-bit immediate data nn and carry (C) from the A register.

Code

MSB							LSB	
0	0	0	1	1	0	1	0	1AH
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Immediate data
Dst: Register direct

Example

	Set Value			Result				
	A	nn	C	A	SC			
					N	V	C	Z
• *D=0, U=0*	A8H	42H	1	65H	0	1	0	0
	36H	5AH	1	DBH	1	0	1	0
• *D=1, U=0*	88	39	1	48	0	0	0	0
• *D=1, U=1*	88	39	1	08	0	0	1	0

SBC A, [BR:ll] |||||||||| *Subtract with carry location [BR:ll] from A reg.* ||||||||||||||||||||||| *3 cycles* |||

Function **A ← A - [BR:*ll*] - C**
Subtracts the content of the data memory and the carry (C) from the A register. The data memory address has been specified by the content of the BR register (upper byte specification) and the 8-bit absolute address *ll* (lower byte specification).
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
0	0	0	1	1	1	0	0	1CH
			l	*l*				*ll*

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: 8-bit absolute
Dst: Register direct

Example

	Set Value			Result				
	A	[BR:*ll*]	C	A	SC			
					N	V	C	Z
• *D=0, U=0*	A8H	42H	1	65H	0	1	0	0
	36H	5AH	1	DBH	1	0	1	0
• *D=1, U=0*	88	39	1	48	0	0	0	0
• *D=1, U=1*	88	39	1	08	0	0	1	0

SBC A, [hhll] *Subtract with carry location [hhll] from A reg.* *4 cycles*

Function **A ← A - [hh*ll*] - C**

Subtracts the content of the data memory and the carry (C) from the A register. The data memory address has been specified by the 16-bit absolute address hh*ll*.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
0	0	0	1	1	1	0	1	1DH
			l	*l*				*ll*
			h	h				hh

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: 16-bit absolute
Dst: Register direct

Example

	Set Value			Result				
	A	[hh*ll*]	C	A	SC			
					N	V	C	Z
• *D=0, U=0*	A8H	42H	1	65H	0	1	0	0
	36H	5AH	1	DBH	1	0	1	0
• *D=1, U=0*	88	39	1	48	0	0	0	0
• *D=1, U=1*	88	39	1	08	0	0	1	0

SBC A, [HL] *Subtract with carry location [HL] from A reg.* *2 cycles*

Function **A ← A - [HL] - C**

Subtracts the content of the data memory and the carry (C) from the A register. The data memory address has been specified by the HL register.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
0	0	0	1	1	0	1	1	1BH

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Register indirect
Dst: Register direct

Example

	Set Value			Result				
	A	[HL]	C	A	SC			
					N	V	C	Z
• *D=0, U=0*	A8H	42H	1	65H	0	1	0	0
	36H	5AH	1	DBH	1	0	1	0
• *D=1, U=0*	88	39	1	48	0	0	0	0
• *D=1, U=1*	88	39	1	08	0	0	1	0

SBC A, [ir] *Subtract with carry location [ir reg.] from A reg.* *2 cycles*

Function **A ← A - [ir] - C**

Subtracts the content of the data memory and the carry (C) from the A register. The data memory address has been specified by the ir register (IX/IY).
The content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
0	0	0	1	1	1	1	ir	1EH/1FH✻

✻

ir		Mnemonic	Code
IX	0	SBC A, [IX]	1EH
IY	1	SBC A, [IY]	1FH

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Register indirect
Dst: Register direct

Example

	Set Value			Result				
	A	[ir]	C	A	SC			
					N	V	C	Z
• *D=0, U=0*	A8H	42H	1	65H	0	1	0	0
	36H	5AH	1	DBH	1	0	1	0
• *D=1, U=0*	88	39	1	48	0	0	0	0
• *D=1, U=1*	88	39	1	08	0	0	1	0

SBC A, [ir+dd] *Subtract with carry location [ir reg. + dd] from A reg. 4 cycles*

Function **A ← A - [ir+dd] - C**

Subtracts the content of the data memory and the carry (C) from the A register. The data memory address has been specified by the sum of the content of the ir register (IX/IY) and the displacement dd.
The displacement dd is handled as signed data and the range is -128 to 127.
The content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory (MODEL2/3).

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Register indirect with displacement
Dst: Register direct

Example

	Set Value			Result				
	A	[ir+dd]	C	A	SC			
					N	V	C	Z
• *D=0, U=0*	A8H	42H	1	65H	0	1	0	0
	36H	5AH	1	DBH	1	0	1	0
• *D=1, U=0*	88	39	1	48	0	0	0	0
• *D=1, U=1*	88	39	1	08	0	0	1	0

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	1	1	0	0	ir	18H/19H✻
			d	d				dd

✻

ir		Mnemonic	Code
IX	0	SBC A, [IX+dd]	18H
IY	1	SBC A, [IY+dd]	19H

SBC A, [ir+L] *Subtract with carry location [ir reg. + L] from A reg. 4 cycles*

Function **A ← A - [ir+L] - C**

Subtracts the content of the data memory and the carry (C) from the A register. The data memory address has been specified by the sum of the content of the ir register (IX/IY) and the content of the L register.
The content of the L register is handled as signed data and the range is -128 to 127.
The content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory (MODEL2/3).

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Register indirect with index register
Dst: Register direct

Example

	Set Value			Result				
	A	[ir+L]	C	A	SC			
					N	V	C	Z
• *D=0, U=0*	A8H	42H	1	65H	0	1	0	0
	36H	5AH	1	DBH	1	0	1	0
• *D=1, U=0*	88	39	1	48	0	0	0	0
• *D=1, U=1*	88	39	1	08	0	0	1	0

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	1	1	0	1	ir	1AH/1BH✻

✻

ir		Mnemonic	Code
IX	0	SBC A, [IX+L]	1AH
IY	1	SBC A, [IY+L]	1BH

SBC [HL], A ||||||||||||||| *Subtract with carry A reg. from location [HL]* |||||||||||||||||||||||||| *4 cycles* |||

Function **[HL] ← [HL] - A - C**

Subtracts the content of the A register and the carry (C) from the data memory that has been address specified by the HL register.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	1	1	1	0	0	1CH

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Register direct
Dst: Register indirect

Example

	Set Value			Result				
					SC			
	[HL]	A	C	[HL]	N	V	C	Z
• *D=0, U=0*	A8H	42H	1	65H	0	1	0	0
	36H	5AH	1	DBH	1	0	1	0
• *D=1, U=0*	88	39	1	48	0	0	0	0
• *D=1, U=1*	88	39	1	08	0	0	1	0

SBC [HL], #nn |||||||||| *Subtract with carry immediate data nn from location [HL]* ||||||| *5 cycles* |||

Function **[HL] ← [HL] - nn - C**

Subtracts the 8-bit immediate data nn and the carry (C) from the data memory that has been address specified by the HL register.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	1	1	1	0	1	1DH
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Immediate data
Dst: Register indirect

Example

	Set Value			Result				
					SC			
	[HL]	nn	C	[HL]	N	V	C	Z
• *D=0, U=0*	A8H	42H	1	65H	0	1	0	0
	36H	5AH	1	DBH	1	0	1	0
• *D=1, U=0*	88	39	1	48	0	0	0	0
• *D=1, U=1*	88	39	1	08	0	0	1	0

SBC [HL], [ir] ||||||||||| Subtract with carry location [ir reg.] from location [HL] ||||||||| 5 cycles |||

Function **[HL] ← [HL] - [ir] - C**

Subtracts the content of the data memory that has been address specified by the ir register (IX/IY) and the carry (C) from the data memory that has been address specified by the HL register.
The content of the EP register becomes the page address of the data memory [HL] and the content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory [ir] (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	1	1	1	1	ir	1EH/1FH✻

✻

ir		Mnemonic	Code
IX	0	SBC [HL], [IX]	1EH
IY	1	SBC [HL], [IY]	1FH

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Register indirect
Dst: Register indirect

Example

	Set Value			Result				
	[HL]	[ir]	C	[HL]	SC			
					N	V	C	Z
• *D=0, U=0*	A8H	42H	1	65H	0	1	0	0
	36H	5AH	1	DBH	1	0	1	0
• *D=1, U=0*	88	39	1	48	0	0	0	0
• *D=1, U=1*	88	39	1	08	0	0	1	0

SBC BA, rp |||||||||||||||| Subtract with carry rp reg. from BA reg. |||||||||||||||||||||||||||||||||| 4 cycles |||

Function **BA ← BA - rp - C**

Subtracts the content of the rp register (BA/HL/IX/IY) and the carry (C) from the BA register.

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	0	0	0	1	1	rp		0CH–0FH✻

✻

rp		Mnemonic	Code
BA	00	SBC BA, BA	0CH
HL	01	SBC BA, HL	0DH
IX	10	SBC BA, IX	0EH
IY	11	SBC BA, IY	0FH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Register direct
Dst: Register direct

Example

Set Value			Result				
BA	rp	C	BA	SC			
				N	V	C	Z
63C2H	2125H	1	429CH	0	1	0	0
205CH	7120H	1	AF3BH	1	0	1	0

(rp≠BA)

SBC BA, #mmnn ||| *Subtract with carry immediate data mmnn from BA reg.* |||||||||| *4 cycles* |||

Function **BA ← BA - mmnn - C**
Subtracts the 16-bit immediate data mmnn and the carry (C) from the BA register.

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	1	1	0	0	0	1	0	62H
			n	n				nn
			m	m				mm

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Immediate data
Dst: Register direct

Example

Set Value			Result				
				SC			
BA	mmnn	C	BA	N	V	C	Z
63C2H	2125H	1	429CH	0	1	0	0
205CH	7120H	1	AF3BH	1	0	1	0

SBC HL, rp |||||||||||||||| *Subtract with carry rp reg. from BA reg.* |||||||||||||||||||||||||||||||||| *4 cycles* |||

Function **HL ← HL - rp - C**
Subtracts the content of the rp register (BA/HL/IX/IY) and the carry (C) from the HL register.

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	0	1	0	1	1	rp		2CH–2FH✻

✻

rp		Mnemonic	Code
BA	00	SBC HL, BA	2CH
HL	01	SBC HL, HL	2DH
IX	10	SBC HL, IX	2EH
IY	11	SBC HL, IY	2FH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Register direct
Dst: Register direct

Example

Set Value			Result				
				SC			
HL	rp	C	HL	N	V	C	Z
63C2H	2125H	1	429CH	0	1	0	0
205CH	7120H	1	AF3BH	1	0	1	0

(rp≠HL)

SBC HL, #mmnn |||| *Subtract with carry immediate data mmnn from HL reg.* |||||||||| *4 cycles* |||

Function **HL ← HL - mmnn - C**
Subtracts the 16-bit immediate data mmnn and the carry (C) from the HL register.

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	1	1	0	0	0	1	0	63H
			n	n				nn
			m	m				mm

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

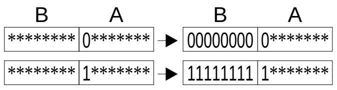
Mode Src: Immediate data
Dst: Register direct

Example

Set Value			Result				
				SC			
HL	mmnn	C	HL	N	V	C	Z
63C2H	2125H	1	429CH	0	1	0	0
205CH	7120H	1	AF3BH	1	0	1	0

SEP |||||||||||||||||||||||||||||||||||||| *Sign expand A reg. to BA reg.* ||||||||||||||||||||||||||||||||||||||||||||||||||||| *3 cycles* |||

Function



Expands the code bit (bit 7) of the 8-bit data stored in the A register into the B register and makes it 16-bit data handled as the BA register.
When the value of the A register is positive (bit 7 is '0') the B register becomes 00H and when it is negative (bit 7 is '1') it becomes FFH.

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	0	1	0	0	0	A8H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

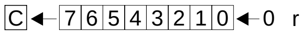
Mode Implide (Register direct)

Example

Set Value		Result				
B	A	BA	SC			
			N	V	C	Z
5CH	76H	0076H	–	–	–	–
42H	A5H	FFA5H	–	–	–	–

SLA r ||||||||||||||||||||||||||||||| *Shift r reg. left arithmetic* ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *3 cycles* |||

Function



Shifts the content of the r register (A/B) 1 bit to the left. Bit 7 of the register moves to the carry (C) and '0' enters bit 0 of the register.
The same result as for the "SLL" instruction is obtained, but the "SLA" instruction also changes the overflow (V) flag due to the arithmetic shift.

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	0	0	0	0	ir	80H/81H✻

✻

r		Mnemonic	Code
A	0	SLA A	80H
B	1	SLA B	81H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Register direct

Example

Set Value	Result				
r	r	SC			
		N	V	C	Z
00111100	01111000	0	0	0	0
10010000	00100000	0	1	1	0

SLA [BR:ll] |||||||||||||||||| *Shift location [BR:ll] left arithmetic* |||||||||||||||||||||||||||||||||||||||||||| *5 cycles* |||

Function $\boxed{C}\leftarrow\boxed{7|6|5|4|3|2|1|0}\leftarrow 0$ [BR:*ll*]

Shifts the content of the data memory 1 bit to the left. The data memory address has been specified by the content of the BR register (upper byte specification) and the 8-bit absolute address *ll* (lower byte specification). Bit 7 of the data moves to the carry (C) and bit 0 of the data becomes '0'.
The same result as for the "SLL" instruction is obtained, but the "SLA" instruction also changes the overflow (V) flag due to the arithmetic shift.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	0	0	0	1	0	82H
			l	*l*				*ll*

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode 8-bit absolute

Example

Set Value	Result				
[BR:*ll*]	[BR:*ll*]	SC			
		N	V	C	Z
00111100	01111000	0	0	0	0
10010000	00100000	0	1	1	0

SLA [HL] |||||||||||||||||||||| *Shift location [HL] left arithmetic* |||||||||||||||||||||||||||||||||||||||||||||| *4 cycles* |||

Function $\boxed{C}\leftarrow\boxed{7|6|5|4|3|2|1|0}\leftarrow 0$ [HL]

Shifts the content of the data memory that has been address specified by the HL register 1 bit to the left. Bit 7 of the data moves to the carry (C) and bit 0 of the data becomes '0'.
The same result as for the "SLL" instruction is obtained, but the "SLA" instruction also changes the overflow (V) flag due to the arithmetic shift.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	0	0	0	1	1	83H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Register indirect

Example

Set Value	Result				
[HL]	[HL]	SC			
		N	V	C	Z
00111100	01111000	0	0	0	0
10010000	00100000	0	1	1	0

SLL r *Shift r reg. left logical* *3 cycles*

Function C ◄— 7 6 5 4 3 2 1 0 ◄— 0 r

Shifts the content of the r register (A/B) 1 bit to the left. Bit 7 of the register moves to the carry (C) and '0' enters bit 0 of the register.
The same result as for the "SLA" instruction is obtained, but the overflow (V) flag does not change due to the logical shift.

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	0	0	1	0	ir	84H/85H✻

✻

r		Mnemonic	Code
A	0	SLL A	84H
B	1	SLL B	85H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	↕	↕

Mode Register direct

Example

Set Value	Result				
r	r	SC			
		N	V	C	Z
00111100	01111000	1	–	0	0
10010000	00100000	0	–	1	0

SLL [BR:ll] *Shift location [BR:ll] left logical* *5 cycles*

Function C ◄— 7 6 5 4 3 2 1 0 ◄— 0 [BR:*ll*]

Shifts the content of the data memory 1 bit to the left. The data memory address has been specified by the content of the BR register (upper byte specification) and the 8-bit absolute address *ll* (lower byte specification).
Bit 7 of the data moves to the carry (C) and bit 0 of the data becomes '0'.
The same result as for the "SLA" instruction is obtained, but the overflow (V) flag does not change due to the logical shift.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	0	0	1	1	0	86H
			l	*l*				*ll*

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	↕	↕

Mode 8-bit absolute

Example

Set Value	Result				
[BR:*ll*]	[BR:*ll*]	SC			
		N	V	C	Z
00111100	01111000	1	–	0	0
10010000	00100000	0	–	1	0

SLL [HL] |||||||||||||||||||||| *Shift location [HL] left logical* |||||||||||||||||||||||||||||||||||||||||||||||| *4 cycles* |||

Function $\boxed{C} \leftarrow \boxed{7|6|5|4|3|2|1|0} \leftarrow 0$ [HL]

Shifts the content of the data memory that has been address specified by the HL register 1 bit to the left. Bit 7 of the data moves to the carry (C) and bit 0 of the data becomes '0'.
The same result as for the "SLA" instruction is obtained, but the overflow (V) flag due to the logical shift.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	0	0	1	1	1	87H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	↕	↕

Mode Register indirect

Example

Set Value	Result				
[HL]	[HL]	SC			
		N	V	C	Z
00111100	01111000	1	–	0	0
10010000	00100000	0	–	1	0

SLP |||||||||||||||||||||||||||||||||||| *Set CPU to SLEEP mode* ||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *3 cycles* |||

Function **SLEEP**

Puts the CPU in the SLEEP status.
In the SLEEP status, the peripheral circuits including the CPU and the oscillation circuit stop operating and power consumption is substantially reduced.
From the SLEEP status, an interrupt outside the MCU will return it to the normal program execution status.

See Section 3.7.2, "SLEEP status".

Code

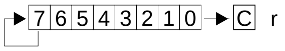
MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	0	1	1	1	1	AFH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

SRA r *Shift r reg. right arithmetic* *3 cycles*

Function



Shifts the content of the r register (A/B) 1 bit to the right. Bit 0 of the register moves to the carry (C) and bit 7 of the register does not change.
The overflow (V) flag is reset to '0'.

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	0	1	0	0	r	88H/89H✻

✻

r		Mnemonic	Code
A	0	SRA A	88H
B	1	SRA B	89H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	0	↕	↕

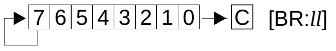
Mode Register direct

Example

Set Value	Result				
r	r	SC			
		N	V	C	Z
01000100	00100010	0	0	0	0
10111001	11011100	1	0	1	0

SRA [BR:ll] *Shift location [BR:ll] right arithmetic* *5 cycles*

Function



Shifts the content of the data memory 1 bit to the right. The data memory address has been specified by the content of the BR register (upper byte specification) and the 8-bit absolute address *ll* (lower byte specification). Bit 0 of the data moves to the carry (C) and bit 7 of the data does not change.
The overflow (V) flag is reset to '0'.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	0	1	0	1	0	8AH
			l	*l*				*ll*

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	0	↕	↕

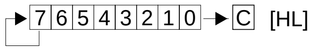
Mode 8-bit absolute

Example

Set Value	Result				
[BR:*ll*]	[BR:*ll*]	SC			
		N	V	C	Z
01000100	00100010	0	0	0	0
10111001	11011100	1	0	1	0

SRA [HL] *Shift location [HL] right arithmetic* *4 cycles*

Function



Shifts the content of the data memory that has been address specified by the HL register 1 bit to the right. Bit 0 of the data moves to the carry (C) and bit 7 of the data does not change.
The overflow (V) flag is reset to '0'.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Mode Register indirect

Example

Set Value	Result				
		SC			
[HL]	[HL]	N	V	C	Z
01000100	00100010	0	0	0	0
10111001	11011100	1	0	1	0

Code

MSB … LSB

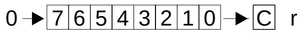
1	1	0	0	1	1	1	0	CEH
1	0	0	0	1	0	1	1	8BH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	0	↕	↕

SRL r *Shift r reg. right logical* *3 cycles*

Function



Shifts the content of the r register (A/B) 1 bit to the right. Bit 0 of the register moves to the carry (C) and bit 7 of the register becomes '0'.

Mode Register direct

Example

Set Value	Result				
		SC			
r	r	N	V	C	Z
01000100	00100010	0	–	0	0
01101101	00110110	0	–	1	0

Code

MSB … LSB

1	1	0	0	1	1	1	0	CEH
1	0	0	0	1	1	0	r	8CH/8DH✻

✻

r		Mnemonic	Code
A	0	SRL A	8CH
B	1	SRL B	8DH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	0	–	↕	↕

SRL [BR:*ll*] Shift location [BR:ll] right logical 5 cycles

Function 0 → [7|6|5|4|3|2|1|0] → [C] [BR:*ll*]

Shifts the content of the data memory 1 bit to the right. The data memory address has been specified by the content of the BR register (upper byte specification) and the 8-bit absolute address *ll* (lower byte specification). Bit 0 of the data moves to the carry (C) and bit 7 of the data becomes '0'.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	0	1	1	1	0	8EH
			l	*l*				*ll*

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	0	–	↕	↕

Mode 8-bit absolute

Example

Set Value	Result				
[BR:*ll*]	[BR:*ll*]	SC			
		N	V	C	Z
01000100	00100010	0	–	0	0
01101101	00110110	0	–	1	0

SRL [HL] Shift location [HL] right logical 4 cycles

Function 0 → [7|6|5|4|3|2|1|0] → [C] [HL]

Shifts the content of the data memory that has been address specified by the HL register 1 bit to the right. Bit 0 of the data moves to the carry (C) and bit 7 of the data becomes '0'.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	0	1	1	1	1	8FH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	0	–	↕	↕

Mode Register indirect

Example

Set Value	Result				
[HL]	[HL]	SC			
		N	V	C	Z
01000100	00100010	0	–	0	0
01101101	00110110	0	–	1	0

SUB A, r ||||||||||||||||||||||||| *Subtract r reg. from A reg.* ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *2 cycles* |||

Function **A ← A - r**
Subtracts the content of the r register (A/B) from the A register.

Code

MSB							LSB	
0	0	0	1	0	0	0	r	10H/11H*

*

r		Mnemonic	Code
A	0	SUB A, A	10H
B	1	SUB A, B	11H

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Register direct
Dst: Register direct

Example

	Set Value		Result				
	A	B	A	SC			
				N	V	C	Z
• D=0, U=0	A8H	42H	66H	0	1	0	0
	36H	5AH	DCH	1	0	1	0
• D=1, U=0	88	39	49	0	0	0	0
• D=1, U=1	88	39	09	0	0	1	0

SUB A, #nn ||||||||||||||||| *Subtract immediate data nn from A reg.* ||||||||||||||||||||||||||||||||||||| *2 cycles* |||

Function **A ← A - nn**
Subtracts 8-bit immediate data nn from the A register.

Code

MSB							LSB	
0	0	0	1	0	0	1	0	12H
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Immediate data
Dst: Register direct

Example

	Set Value		Result				
	A	nn	A	SC			
				N	V	C	Z
• D=0, U=0	A8H	42H	66H	0	1	0	0
	36H	5AH	DCH	1	0	1	0
• D=1, U=0	88	39	49	0	0	0	0
• D=1, U=1	88	39	09	0	0	1	0

SUB A, [BR:ll] |||||||||| *Subtract location [BR:ll] from A reg.* ||||||||||||||||||||||||||||||||||||||||| *3 cycles* |||

Function **A ← A - [BR:*ll*]**
Subtracts the content of the data memory from the A register. The data memory address has been specified by the content of the BR register (upper byte specification) and the 8-bit absolute address *ll* (lower byte specification).
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
0	0	0	1	0	1	0	0	14H
			l	*l*				*ll*

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: 8-bit absolute
Dst: Register direct

Example

	Set Value		Result				
	A	[BR:*ll*]	A	SC			
				N	V	C	Z
• D=0, U=0	A8H	42H	66H	0	1	0	0
	36H	5AH	DCH	1	0	1	0
• D=1, U=0	88	39	49	0	0	0	0
• D=1, U=1	88	39	09	0	0	1	0

SUB A, [hhll] *Subtract location [hhll] from A reg.* *4 cycles*

Function **A ← A - [hh*ll*]**

Subtracts the content of the data memory that has been address specified by the 16-bit absolute address hh*ll* from the A register.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
0	0	0	1	0	1	0	1	15H
			l	*l*				*ll*
			h	h				hh

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: 16-bit absolute
Dst: Register direct

Example

	Set Value		Result				
	A	[hh*ll*]	A	SC			
				N	V	C	Z
• *D=0, U=0*	A8H	42H	66H	0	1	0	0
	36H	5AH	DCH	1	0	1	0
• *D=1, U=0*	88	39	49	0	0	0	0
• *D=1, U=1*	88	39	09	0	0	1	0

SUB A, [HL] *Subtract location [HL] from A reg.* *2 cycles*

Function **A ← A - [HL]**

Subtracts the content of the data memory that has been address specified by the HL register from the A register.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
0	0	0	1	0	0	1	1	13H

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Register indirect
Dst: Register direct

Example

	Set Value		Result				
	A	[HL]	A	SC			
				N	V	C	Z
• *D=0, U=0*	A8H	42H	66H	0	1	0	0
	36H	5AH	DCH	1	0	1	0
• *D=1, U=0*	88	39	49	0	0	0	0
• *D=1, U=1*	88	39	09	0	0	1	0

SUB A, [ir] *Subtract location [ir reg.] from A reg.* *2 cycles*

Function **A ← A - [ir]**

Subtracts the content of the data memory that has been address specified by the ir register (IX/IY) from the A register.
The content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
0	0	0	1	0	1	1	ir	16H/17H✻

✻

ir		Mnemonic	Code
IX	0	SUB A, [IX]	16H
IY	1	SUB A, [IY]	17H

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Register indirect
Dst: Register direct

Example

	Set Value		Result				
	A	[ir]	A	SC			
				N	V	C	Z
• *D=0, U=0*	A8H	42H	66H	0	1	0	0
	36H	5AH	DCH	1	0	1	0
• *D=1, U=0*	88	39	49	0	0	0	0
• *D=1, U=1*	88	39	09	0	0	1	0

SUB A, [ir+dd] *Subtract location [ir reg. + dd] from A reg.* *4 cycles*

Function **A ← A - [ir+dd]**

Subtracts the content of the data memory from the A register. The data memory address has been specified by the sum of the content of the ir register (IX/IY) and the displacement dd.

The displacement dd is handled as signed data and the range is -128 to 127.

The content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	1	0	0	0	ir	10H/11H✻
			d	d				dd

✻

ir		Mnemonic	Code
IX	0	SUB A, [IX+dd]	10H
IY	1	SUB A, [IY+dd]	11H

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Register indirect with displacement
Dst: Register direct

Example

	Set Value		Result				
	A	[ir+dd]	A	SC			
				N	V	C	Z
• *D=0, U=0*	A8H	42H	66H	0	1	0	0
	36H	5AH	DCH	1	0	1	0
• *D=1, U=0*	88	39	49	0	0	0	0
• *D=1, U=1*	88	39	09	0	0	1	0

SUB A, [ir+L] *Subtract location [ir reg. + L] from A reg.* *4 cycles*

Function **A ← A - [ir+L]**

Subtracts the content of the data memory from the A register. The data memory address has been specified by the sum of the content of the ir register (IX/IY) and the content of the L register.

The content of the L register is handled as signed data and the range is -128 to 127.

The content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	1	0	0	1	ir	12H/13H✻

✻

ir		Mnemonic	Code
IX	0	SUB A, [IX+L]	12H
IY	1	SUB A, [IY+L]	13H

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Register indirect with index register
Dst: Register direct

Example

	Set Value		Result				
	A	[ir+L]	A	SC			
				N	V	C	Z
• *D=0, U=0*	A8H	42H	66H	0	1	0	0
	36H	5AH	DCH	1	0	1	0
• *D=1, U=0*	88	39	49	0	0	0	0
• *D=1, U=1*	88	39	09	0	0	1	0

SUB [HL], A ||||||||||||||| *Subtract A reg. from location [HL]* |||||||||||||||||||||||||||||||||||||||||||||| *4 cycles* |||

Function **[HL] ← [HL] - A**

Subtracts the content of the A register from the data memory that has been address specified by the HL register.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	1	0	1	0	0	14H

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Register direct
Dst: Register indirect

Example

	Set Value		Result				
				SC			
	[HL]	A	[HL]	N	V	C	Z
• *D=0, U=0*	A8H	42H	66H	0	1	0	0
	36H	5AH	DCH	1	0	1	0
• *D=1, U=0*	88	39	49	0	0	0	0
• *D=1, U=1*	88	39	09	0	0	1	0

SUB [HL], #nn |||||||||| *Subtract immediate data nn from location [HL]* |||||||||||||||||||||||| *5 cycles* |||

Function **[HL] ← [HL] - nn**

Subtracts the 8-bit immediate data nn from the data memory that has been address specified by the HL register.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	1	0	1	0	1	15H
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Immediate data
Dst: Register indirect

Example

	Set Value		Result				
				SC			
	[HL]	nn	[HL]	N	V	C	Z
• *D=0, U=0*	A8H	42H	66H	0	1	0	0
	36H	5AH	DCH	1	0	1	0
• *D=1, U=0*	88	39	49	0	0	0	0
• *D=1, U=1*	88	39	09	0	0	1	0

SUB [HL], [ir] |||||||||||| *Subtract location [ir reg.] from location [HL]* ||||||||||||||||||||||||||| *5 cycles* |||

Function **[HL] ← [HL] - [ir]**

Subtracts the content of the data memory that has been address specified by the ir register (IX/IY) from the data memory that has been address specified by the HL register.
The content of the EP register becomes the page address of the data memory [HL] and the content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory [ir] (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	1	0	1	1	ir	16H/17H✻

✻

ir		Mnemonic	Code
IX	0	SUB [HL], [IX]	16H
IY	1	SUB [HL], [IY]	17H

Flag

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

Mode Src: Register indirect
Dst: Register indirect

Example

	Set Value		Result				
	[HL]	[ir]	[HL]	SC			
				N	V	C	Z
• *D=0, U=0*	A8H	42H	66H	0	1	0	0
	36H	5AH	DCH	1	0	1	0
• *D=1, U=0*	88	39	49	0	0	0	0
• *D=1, U=1*	88	39	09	0	0	1	0

SUB BA, rp |||||||||||||||| *Subtract rp reg. from BA reg.* ||||||||||||||||||||||||||||||||||||||||||||||||||||| *4 cycles* |||

Function **BA ← BA - rp**

Subtracts the content of the rp register (BA/HL/IX/IY) from the BA register.

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	0	0	0	1	0	rp		08H–0BH✻

✻

rp		Mnemonic	Code
BA	00	SUB BA, BA	08H
HL	01	SUB BA, HL	09H
IX	10	SUB BA, IX	0AH
IY	11	SUB BA, IY	0BH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Register direct
Dst: Register direct

Example

Set Value		Result				
BA	rp	BA	SC			
			N	V	C	Z
63C2H	2125H	429DH	0	0	0	0
C261H	5A32H	682FH	0	1	0	0
205CH	7120H	AF3CH	1	0	1	0

(rp≠BA)

SUB BA, #mmnn *Subtract immediate data mmnn from BA reg.* *3 cycles*

Function **BA ← BA - mmnn**

Subtracts the 16-bit immediate data mmnn from the BA register.

Mode Src: Immediate data
Dst: Register direct

Code

MSB							LSB	
1	1	0	1	0	0	0	0	D0H
			n	n				nn
			m	m				mm

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Example

Set Value		Result				
BA	mmnn	BA	SC			
			N	V	C	Z
63C2H	2125H	429DH	0	0	0	0
C261H	5A32H	682FH	0	1	0	0
205CH	7120H	AF3CH	1	0	1	0

SUB HL, rp *Subtract rp reg. from HL reg.* *4 cycles*

Function **HL ← HL - rp**

Subtracts the content of the rp register (BA/HL/IX/IY) from the HL register.

Mode Src: Register direct
Dst: Register direct

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	0	1	0	1	0	rp		28H–2BH✻

✻

rp		Mnemonic	Code
BA	00	SUB HL, BA	28H
HL	01	SUB HL, HL	29H
IX	10	SUB HL, IX	2AH
IY	11	SUB HL, IY	2BH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Example

Set Value		Result				
HL	rp	HL	SC			
			N	V	C	Z
63C2H	2125H	429DH	0	0	0	0
C261H	5A32H	682FH	0	1	0	0
205CH	7120H	AF3CH	1	0	1	0

(rp≠HL)

SUB HL, #mmnn *Subtract immediate data mmnn from HL reg.* *3 cycles*

Function **HL ← HL - mmnn**

Subtracts the 16-bit immediate data mmnn from the HL register.

Mode Src: Immediate data
Dst: Register direct

Code

MSB							LSB	
1	1	0	1	0	0	0	1	D1H
			n	n				nn
			m	m				mm

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Example

Set Value		Result				
HL	mmnn	HL	SC			
			N	V	C	Z
63C2H	2125H	429DH	0	0	0	0
C261H	5A32H	682FH	0	1	0	0
205CH	7120H	AF3CH	1	0	1	0

SUB IX, rp ||||||||||||||||||| *Subtract rp reg. from IX reg.* ||||||||||||||||||||||||||||||||||||||||||||||||||||||| *4 cycles* |||

Function **IX ← IX - rp**

Subtracts the content of the rp register (BA/HL) from the IX register.

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	1	0	0	1	0	0	rp	48H/49H✻

✻

rp		Mnemonic	Code
BA	0	SUB IX, BA	48H
HL	1	SUB IX, HL	49H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Register direct
Dst: Register direct

Example

Set Value		Result				
IX	rp	IX	SC			
			N	V	C	Z
63C2H	2125H	429DH	0	0	0	0
C261H	5A32H	682FH	0	1	0	0
205CH	7120H	AF3CH	1	0	1	0

SUB IX, #mmnn |||||| *Subtract immediate data mmnn from IX reg.* |||||||||||||||||||||||||||||| *3 cycles* |||

Function **IX ← IX - mmnn**

Subtracts the 16-bit immediate data mmnn from the IX register.

Code

MSB							LSB	
1	1	0	1	0	0	1	0	D2H
			n	n				nn
			m	m				mm

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result				
IX	mmnn	IX	SC			
			N	V	C	Z
63C2H	2125H	429DH	0	0	0	0
C261H	5A32H	682FH	0	1	0	0
205CH	7120H	AF3CH	1	0	1	0

SUB IY, rp ||||||||||||||||||| *Subtract rp reg. from IY reg.* ||||||||||||||||||||||||||||||||||||||||||||||||||||||| *4 cycles* |||

Function **IY ← IY - rp**

Subtracts the content of the rp register (BA/HL) from the IY register.

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	1	0	0	1	0	1	rp	4AH/4BH✻

✻

rp		Mnemonic	Code
BA	0	SUB IY, BA	4AH
HL	1	SUB IY, HL	4BH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Mode Src: Register direct
Dst: Register direct

Example

Set Value		Result				
IY	rp	IY	SC			
			N	V	C	Z
63C2H	2125H	429DH	0	0	0	0
C261H	5A32H	682FH	0	1	0	0
205CH	7120H	AF3CH	1	0	1	0

SUB IY, #mmnn |||||| *Subtract immediate data mmnn from IY reg.* |||||||||||||||||||||||||||||| *3 cycles* |||

Function **IY ← IY - mmnn**

Subtracts the 16-bit immediate data mmnn from the IY register.

Mode Src: Immediate data
Dst: Register direct

Code

MSB							LSB	
1	1	0	1	0	0	1	1	D3H
			n	n				nn
			m	m				mm

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Example

Set Value		Result				
IY	mmnn	IY	SC			
			N	V	C	Z
63C2H	2125H	429DH	0	0	0	0
C261H	5A32H	682FH	0	1	0	0
205CH	7120H	AF3CH	1	0	1	0

SUB SP, rp |||||||||||||||||| *Subtract rp reg. from SP* |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *4 cycles* |||

Function **SP ← SP - rp**

Subtracts the content of the rp register (BA/HL) from the stack pointer (SP).

Mode Src: Register direct
Dst: Register direct

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	1	0	0	1	1	0	rp	4CH/4DH✻

✻

rp		Mnemonic	Code
BA	0	SUB SP, BA	4CH
HL	1	SUB SP, HL	4DH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Example

Set Value		Result				
SP	rp	SP	SC			
			N	V	C	Z
63C2H	2125H	429DH	0	0	0	0
C261H	5A32H	682FH	0	1	0	0
205CH	7120H	AF3CH	1	0	1	0

SUB SP, #mmnn |||| *Subtract immediate data mmnn from SP* |||||||||||||||||||||||||||||||||||||| *4 cycles* |||

Function **SP ← SP - mmnn**

Subtracts the 16-bit immediate data mmnn from the stack pointer (SP).

Mode Src: Immediate data
Dst: Register direct

Code

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	1	1	0	1	0	1	0	6AH
			n	n				nn
			m	m				mm

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

Example

Set Value		Result				
SP	mmnn	SP	SC			
			N	V	C	Z
63C2H	2125H	429DH	0	0	0	0
C261H	5A32H	682FH	0	1	0	0
205CH	7120H	AF3CH	1	0	1	0

SWAP A |||||||||||||||||||||||| *Swap high-order and low-order of A reg.* |||||||||||||||||||||||||||||||||| *2 cycles* |||

Function **A(H) ↔ A(L)**

Replaces the content of the lower 4 bits with the upper 4 bits of the A register.

Mode Register direct

Code

MSB							LSB	
1	1	1	1	0	1	1	0	F6H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Example

Set Value	Result				
A	A	SC			
		N	V	C	Z
4CH	C4H	–	–	–	–
62H	26H	–	–	–	–

SWAP [HL] ||||||||||||||||| *Swap high-order and low-order of location [HL]* |||||||||||||||||||||| *3 cycles* |||

Function **[HL](H) ↔ [HL](L)**

Replaces the content of the lower 4 bits with the upper 4 bits of the data memory specified by the HL register.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Mode Register indirect

Code

MSB							LSB	
1	1	1	1	0	1	1	1	F7H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Example

Set Value	Result				
[HL]	[HL]	SC			
		N	V	C	Z
4CH	C4H	–	–	–	–
62H	26H	–	–	–	–

UPCK |||||||||||||||||||||||||||||||| *Unpack A reg. to BA reg.* |||||||||||||||||||||||||||||||||||||||||||||||||||||||||| *2 cycles* |||

Function

$\overset{A}{\boxed{m|n}} \longrightarrow \overset{B}{\boxed{0|m}}\overset{A}{\boxed{0|n}}$

Unpacks the content of the A register and stores it in the BA register. The lower 4 bits of the B register are substituted for the content of the upper 4 bits of the A register and both of the upper 4 bits of the A register and B register become '0'.

Mode Implide (Register direct)

Code

MSB							LSB	
1	1	0	1	1	1	1	1	DFH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	–	–	–	–

Example

Set Value	Result				
A	BA	SC			
		N	V	C	Z
84H	0804H	–	–	–	–

XOR A, r ||||||||||||||||||||||||| *Exclusive OR r reg. and A reg.* |||||||||||||||||||||||||||||||||||||||||||||||||||| *2 cycles* |||

Function **A ← A ∀ r**

Takes an exclusive OR of the content of the r register (A/B) and the content of the A register and stores the result in the A register.

Code

MSB							LSB	
0	0	1	1	1	0	0	r	38H/39H✻

✻

r		Mnemonic	Code
A	0	XOR A, A	38H
B	1	XOR A, B	39H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Register direct
Dst: Register direct

Example

Set Value		Result				
A	B	A	SC			
			N	V	C	Z
2CH	41H	6DH	0	–	–	0
7AH	B6H	CCH	1	–	–	0

XOR A, #nn |||||||||||||||||| *Exclusive OR immediate data nn and A reg.* |||||||||||||||||||||||||||||||| *2 cycles* |||

Function **A ← A ∀ nn**

Takes an exclusive OR of the 8-bit immediate data nn and the content of the A register and stores the result in the A register.

Code

MSB							LSB	
0	0	1	1	1	0	1	0	3AH
n	n							nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result				
A	nn	A	SC			
			N	V	C	Z
2CH	41H	6DH	0	–	–	0
7AH	B6H	CCH	1	–	–	0

XOR A, [BR:ll] ||||||||||| *Exclusive OR location [BR:ll] and A reg.* ||||||||||||||||||||||||||||||||||||| *3 cycles* |||

Function **A ← A ∀ [BR:*ll*]**

Takes an exclusive OR of the content of the data memory and the content of the A register and stores the result in the A register. The data memory address has been specified by the content of the BR register (upper byte specification) and the 8-bit absolute address *ll* (lower byte specification).
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
0	0	1	1	1	1	0	0	3CH
l	*l*							*ll*

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: 8-bit absolute
Dst: Register direct

Example

Set Value		Result				
A	[BR:*ll*]	A	SC			
			N	V	C	Z
2CH	41H	6DH	0	–	–	0
7AH	B6H	CCH	1	–	–	0

XOR A, [hhll] |||||||||||| Exclusive OR location [hhll] and A reg. ||||||||||||||||||||||||||||||||||| 4 cycles |||

Function **A ← A ∀ [hh*ll*]**

Takes an exclusive OR of the content of the data memory that has been address specified by the 16-bit absolute address hh*ll* and the content of the A register and stores the result in the A register.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
0	0	1	1	1	1	0	1	3DH
			l	*l*				*ll*
			h	h				hh

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: 16-bit absolute
Dst: Register direct

Example

Set Value		Result				
A	[hh*ll*]	A	SC			
			N	V	C	Z
2CH	41H	6DH	0	–	–	0
7AH	B6H	CCH	1	–	–	0

XOR A, [HL] |||||||||||||| Exclusive OR location [HL] and A reg. ||||||||||||||||||||||||||||||||||||| 2 cycles |||

Function **A ← A ∀ [HL]**

Takes an exclusive OR of the content of the data memory that has been address specified by the HL register and the content of the A register and stores the result in the A register.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
0	0	1	1	1	0	1	1	3BH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Register indirect
Dst: Register direct

Example

Set Value		Result				
A	[HL]	A	SC			
			N	V	C	Z
2CH	41H	6DH	0	–	–	0
7AH	B6H	CCH	1	–	–	0

XOR A, [ir] ||||||||||||||||| Exclusive OR location [ir reg.] and A reg. ||||||||||||||||||||||||||||||| 2 cycles |||

Function **A ← A ∀ [ir]**

Takes an exclusive OR of the content of the data memory that has been address specified by the ir register (IX/IY) and the content of the A register and stores the result in the A register.
The content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
0	0	1	1	1	1	1	ir	3EH/3FH*

*

ir		Mnemonic	Code
IX	0	XOR A, [IX]	3EH
IY	1	XOR A, [IY]	3FH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Register indirect
Dst: Register direct

Example

Set Value		Result				
A	[ir]	A	SC			
			N	V	C	Z
2CH	41H	6DH	0	–	–	0
7AH	B6H	CCH	1	–	–	0

XOR A, [ir+dd] ||||||||| *Exclusive OR location [ir reg. + dd] and A reg.* ||||||||||||||||||||||||| *4 cycles* |||

Function **A ← A ∀ [ir+dd]**

Takes an exclusive OR of the content of the data memory and the content of the A register and stores the result in the A register. The data memory address has been specified by the sum of the content of the ir register (IX/IY) and the displacement dd.
The displacement dd is handled as signed data and the range is -128 to 127.
The content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	1	1	1	0	0	ir	38H/39H✻
			d	d				dd

✻

ir		Mnemonic	Code
IX	0	XOR A, [IX+dd]	38H
IY	1	XOR A, [IY+dd]	39H

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Register indirect with displacement
Dst: Register direct

Example

Set Value		Result				
A	[ir+dd]	A	SC			
			N	V	C	Z
2CH	41H	6DH	0	–	–	0
7AH	B6H	CCH	1	–	–	0

XOR A, [ir+L] |||||||||||| *Exclusive OR location [ir reg. + L] and A reg.* ||||||||||||||||||||||||||| *4 cycles* |||

Function **A ← A ∀ [ir+L]**

Takes an exclusive OR of the content of the data memory and the content of the A register and stores the result in the A register. The data memory address has been specified by the sum of the content of the ir register (IX/IY) and the content of the L register.
The content of the L register is handled as signed data and the range is -128 to 127.
The content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	1	1	1	0	1	ir	3AH/3BH✻

✻

ir		Mnemonic	Code
IX	0	XOR A, [IX+L]	3AH
IY	1	XOR A, [IY+L]	3BH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Register indirect with index register
Dst: Register direct

Example

Set Value		Result				
A	[ir+L]	A	SC			
			N	V	C	Z
2CH	41H	6DH	0	–	–	0
7AH	B6H	CCH	1	–	–	0

XOR B, #nn ||||||||||||||| *Exclusive OR immediate data nn and B reg.* ||||||||||||||||||||||||||||||| *3 cycles* |||

Function **B ← B ∀ nn**

Takes an exclusive OR of the 8-bit immediate data nn and the content of the B register and stores the result in the B register.

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	1	1	0	0	0	B8H
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result				
B	nn	B	SC			
			N	V	C	Z
2CH	41H	6DH	0	–	–	0
7AH	B6H	CCH	1	–	–	0

XOR L, #nn |||||||||||||||| *Exclusive OR immediate data nn and L reg.* ||||||||||||||||||||||||||||||| *3 cycles* |||

Function **L ← L ∀ nn**

Takes an exclusive OR of the 8-bit immediate data nn and the content of the L register and stores the result in the L register.

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	1	1	0	0	1	B9H
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result				
L	nn	L	SC			
			N	V	C	Z
2CH	41H	6DH	0	–	–	0
7AH	B6H	CCH	1	–	–	0

XOR H, #nn ||||||||||||||| *Exclusive OR immediate data nn and H reg.* ||||||||||||||||||||||||||||||| *3 cycles* |||

Function **H ← H ∀ nn**

Takes an exclusive OR of the 8-bit immediate data nn and the content of the H register and stores the result in the H register.

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	1	1	0	1	0	BAH
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result				
H	nn	H	SC			
			N	V	C	Z
2CH	41H	6DH	0	–	–	0
7AH	B6H	CCH	1	–	–	0

XOR SC, #nn |||||||||||| *Exclusive OR immediate data nn and SC* |||||||||||||||||||||||||||||||||| *3 cycles* |||

Function **SC ← SC ∀ nn**
Takes an exclusive OR of the 8-bit immediate data nn and the content of the system condition flag (SC) and sets the result in the system condition flag (SC).

Code
MSB LSB

1	0	0	1	1	1	1	0	9EH
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
↕	↕	↕	↕	↕	↕	↕	↕

Mode Src: Immediate data
Dst: Register direct

Example

Set Value		Result							
SC	nn	SC							
		I1	I0	U	D	N	V	C	Z
2CH	41H	0	1	1	0	1	1	0	1
7AH	B6H	1	1	0	0	1	1	0	0

XOR [BR:ll], #nn |||| *Exclusive OR immediate data nn and location [BR:ll]* |||||||||||||| *5 cycles* |||

Function **[BR:*ll*] ← [BR:*ll*] ∀ nn**
Takes an exclusive OR of the 8-bit immediate data and the content of the data memory and stores the result in that address. The data memory address has been specified by the content of the BR register (upper byte specification) and the 8-bit absolute address *ll* (lower byte specification).
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code
MSB LSB

1	1	0	1	1	0	1	0	DAH
			l	*l*				*ll*
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Immediate data
Dst: 8-bit absolute

Example

Set Value		Result				
[BR:*ll*]	nn	[BR:*ll*]	SC			
			N	V	C	Z
2CH	41H	6DH	0	–	–	0
7AH	B6H	CCH	1	–	–	0

XOR [HL], A |||||||||||||| *Exclusive OR A reg. and location [HL]* ||||||||||||||||||||||||||||||||||||| *4 cycles* |||

Function **[HL] ← [HL] ∀ A**
Takes an exclusive OR of the content of the A register and the data memory that has been address specified by the HL register and stores the result in that address.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code
MSB LSB

1	1	0	0	1	1	1	0	CEH
0	0	1	1	1	1	0	0	3CH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Register direct
Dst: Register indirect

Example

Set Value		Result				
[HL]	A	[HL]	SC			
			N	V	C	Z
2CH	41H	6DH	0	–	–	0
7AH	B6H	CCH	1	–	–	0

XOR [HL], #nn ||||||||| *Exclusive OR immediate data nn and location [HL]* |||||||||||||||||| *5 cycles* |||

Function **[HL] ← [HL] ∀ nn**

Takes an exclusive OR of the 8-bit immediate data nn and the data memory that has been address specified by the HL register and stores the result in that address.
The content of the EP register becomes the page address of the data memory (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	1	1	1	1	0	1	3DH
			n	n				nn

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Immediate data
Dst: Register indirect

Example

Set Value		Result				
[HL]	nn	[HL]	SC			
			N	V	C	Z
2CH	41H	6DH	0	–	–	0
7AH	B6H	CCH	1	–	–	0

XOR [HL], [ir] |||||||||||| *Exclusive OR location [ir reg.] and location [HL]* ||||||||||||||||||| *5 cycles* |||

Function **[HL] ← [HL] ∀ [ir]**

Takes an exclusive OR of the content of the data memory that has been address specified by the ir register (IX/IY) and the data memory that has been address specified by the HL register and stores the result in data memory [HL].
The content of the EP register becomes the page address of the data memory [HL] and the content of the XP register (at time of IX specification) or the YP register (at time of IY specification) becomes the page address of the data memory [ir] (MODEL2/3).

Code

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	1	1	1	1	1	ir	3EH/3FH✻

✻

ir		Mnemonic	Code
IX	0	XOR [HL], [IX]	3EH
IY	1	XOR [HL], [IY]	3FH

Flag

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	–	–	↕

Mode Src: Register indirect
Dst: Register indirect

Example

Set Value		Result				
[HL]	[ir]	[HL]	SC			
			N	V	C	Z
2CH	41H	6DH	0	–	–	0
7AH	B6H	CCH	1	–	–	0

APPENDIX A Operation Code Map

Operation code map (1/3)

1st operation code

L \ H	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	ADD A,A	SUB A,A	AND A,A	CP A,A	LD A,A	LD L,A	LD [IX],A	LD [IY],A	INC A	INC BA	PUSH BA	LD A,#nn	ADD BA,#mmnn	SUB BA,#mmnn	CARS C,rr	CARS rr
1	ADD A,B	SUB A,B	AND A,B	CP A,B	LD A,B	LD L,B	LD [IX],B	LD [IY],B	INC B	INC HL	PUSH HL	LD B,#nn	ADD HL,#mmnn	SUB HL,#mmnn	CARS NC,rr	JRS rr
2	ADD A,#nn	SUB A,#nn	AND A,#nn	CP A,#nn	LD A,L	LD L,L	LD [IX],L	LD [IY],L	INC L	INC IX	PUSH IX	LD L,#nn	ADD IX,#mmnn	SUB IX,#mmnn	CARS Z,rr	CARL qqrr
3	ADD A,[HL]	SUB A,[HL]	AND A,[HL]	CP A,[HL]	LD A,H	LD L,H	LD [IX],H	LD [IY],H	INC H	INC IY	PUSH IY	LD H,#nn	ADD IY,#mmnn	SUB IY,#mmnn	CARS NZ,rr	JRL qqrr
4	ADD A,[BR:ll]	SUB A,[BR:ll]	AND A,[BR:ll]	CP A,[BR:ll]	LD A,[BR:ll]	LD L,[BR:ll]	LD [IX],[BR:ll]	LD [IY],[BR:ll]	INC BR	BIT A,B	PUSH BR	LD BR,#hh	LD BA,#mmnn	CP BA,#mmnn	JRS C,rr	JP HL
5	ADD A,[hhll]	SUB A,[hhll]	AND A,[hhll]	CP A,[hhll]	LD A,[HL]	LD L,[HL]	LD [IX],[HL]	LD [IY],[HL]	INC [BR:ll]	BIT [HL],#nn	PUSH EP	LD [HL],#nn	LD HL,#mmnn	CP HL,#mmnn	JRS NC,rr	DJR NZ,rr
6	ADD A,[IX]	SUB A,[IX]	AND A,[IX]	CP A,[IX]	LD A,[IX]	LD L,[IX]	LD [IX],[IX]	LD [IY],[IX]	INC [HL]	BIT A,#nn	PUSH IP	LD [IX],#nn	LD IX,#mmnn	CP IX,#mmnn	JRS Z,rr	SWAP A
7	ADD A,[IY]	SUB A,[IY]	AND A,[IY]	CP A,[IY]	LD A,[IY]	LD L,[IY]	LD [IX],[IY]	LD [IY],[IY]	INC SP	BIT B,#nn	PUSH SC	LD [IY],#nn	LD IY,#mmnn	CP IY,#mmnn	JRS NZ,rr	SWAP [HL]
8	ADC A,A	SBC A,A	OR A,A	XOR A,A	LD B,A	LD H,A	LD [HL],A	LD [BR:ll],A	DEC A	DEC BA	POP BA	LD BA,[hhll]	EX BA,HL	AND [BR:ll],#nn	CARL C,qqrr	RET
9	ADC A,B	SBC A,B	OR A,B	XOR A,B	LD B,B	LD H,B	LD [HL],B	LD [BR:ll],B	DEC B	DEC HL	POP HL	LD HL,[hhll]	EX BA,IX	OR [BR:ll],#nn	CARL NC,qqrr	RETE
A	ADC A,#nn	SBC A,#nn	OR A,#nn	XOR A,#nn	LD B,L	LD H,L	LD [HL],L	LD [BR:ll],L	DEC L	DEC IX	POP IX	LD IX,[hhll]	EX BA,IY	XOR [BR:ll],#nn	CARL Z,qqrr	RETS
B	ADC A,[HL]	SBC A,[HL]	OR A,[HL]	XOR A,[HL]	LD B,H	LD H,H	LD [HL],H	LD [BR:ll],H	DEC H	DEC IY	POP IY	LD IY,[hhll]	EX BA,SP	CP [BR:ll],#nn	CARL NZ,qqrr	CALL [hhll]
C	ADC A,[BR:ll]	SBC A,[BR:ll]	OR A,[BR:ll]	XOR A,[BR:ll]	LD B,[BR:ll]	LD H,[BR:ll]	LD [HL],[BR:ll]	Undefined Code	DEC BR	AND SC,#nn	POP BR	LD [hhll],BA	EX A,B	BIT [BR:ll],#nn	JRL C,qqrr	INT [kk]
D	ADC A,[hhll]	SBC A,[hhll]	OR A,[hhll]	XOR A,[hhll]	LD B,[HL]	LD H,[HL]	LD [HL],[HL]	LD [BR:ll],[HL]	DEC [BR:ll]	OR SC,#nn	POP EP	LD [hhll],HL	EX A,[HL]	LD [BR:ll],#nn	JRL NC,qqrr	JP [kk]
E	ADC A,[IX]	SBC A,[IX]	OR A,[IX]	XOR A,[IX]	LD B,[IX]	LD H,[IX]	LD [HL],[IX]	LD [BR:ll],[IX]	DEC [HL]	XOR SC,#nn	POP IP	LD [hhll],IX	Expansion Code	PACK	JRL Z,qqrr	Undefined Code
F	ADC A,[IY]	SBC A,[IY]	OR A,[IY]	XOR A,[IY]	LD B,[IY]	LD H,[IY]	LD [HL],[IY]	LD [BR:ll],[IY]	DEC SP	LD SC,#nn	POP SC	LD [hhll],IY	Expansion Code	UPCK	JRL NZ,qqrr	NOP

Operation code map (2/3)

2nd operation code (1st operation code = CE)

L \ H	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	ADD A,[IX+dd]	SUB A,[IX+dd]	AND A,[IX+dd]	CP A,[IX+dd]	LD A,[IX+dd]	LD L,[IX+dd]	LD [HL],[IX+dd]		SLA A	RL A	CPL A	AND B,#nn	LD A,BR	LD A,[hh*ll*]	JRS LT,rr	CARS LT,rr
1	ADD A,[IY+dd]	SUB A,[IY+dd]	AND A,[IY+dd]	CP A,[IY+dd]	LD A,[IY+dd]	LD L,[IY+dd]	LD [HL],[IY+dd]		SLA B	RL B	CPL B	AND L,#nn	LD A,SC	LD B,[hh*ll*]	JRS LE,rr	CARS LE,rr
2	ADD A,[IX+L]	SUB A,[IX+L]	AND A,[IX+L]	CP A,[IX+L]	LD A,[IX+L]	LD L,[IX+L]	LD [HL],[IX+L]		SLA [BR:*ll*]	RL [BR:*ll*]	CPL [BR:*ll*]	AND H,#nn	LD BR,A	LD L,[hh*ll*]	JRS GT,rr	CARS GT,rr
3	ADD A,[IY+L]	SUB A,[IY+L]	AND A,[IY+L]	CP A,[IY+L]	LD A,[IY+L]	LD L,[IY+L]	LD [HL],[IY+L]		SLA [HL]	RL [HL]	CPL [HL]	Undefined Code	LD SC,A	LD H,[hh*ll*]	JRS GE,rr	CARS GE,rr
4	ADD [HL],A	SUB [HL],A	AND [HL],A	CP [HL],A	LD [IX+dd],A	LD [IX+dd],L			SLL A	RLC A	NEG A	OR B,#nn	LD NB,#bb	LD [hh*ll*],A	JRS V,rr	CARS V,rr
5	ADD [HL],#nn	SUB [HL],#nn	AND [HL],#nn	CP [HL],#nn	LD [IY+dd],A	LD [IY+dd],L	Undefined Code Area		SLL B	RLC B	NEG B	OR L,#nn	LD EP,#pp	LD [hh*ll*],B	JRS NV,rr	CARS NV,rr
6	ADD [HL],[IX]	SUB [HL],[IX]	AND [HL],[IX]	CP [HL],[IX]	LD [IX+L],A	LD [IX+L],L			SLL [BR:*ll*]	RLC [BR:*ll*]	NEG [BR:*ll*]	OR H,#nn	LD XP,#pp	LD [hh*ll*],L	JRS P,rr	CARS P,rr
7	ADD [HL],[IY]	SUB [HL],[IY]	AND [HL],[IY]	CP [HL],[IY]	LD [IY+L],A	LD [IY+L],L			SLL [HL]	RLC [HL]	NEG [HL]	Undefined Code	LD YP,#pp	LD [hh*ll*],H	JRS M,rr	CARS M,rr
8	ADC A,[IX+dd]	SBC A,[IX+dd]	OR A,[IX+dd]	XOR A,[IX+dd]	LD B,[IX+dd]	LD H,[IX+dd]	LD [IX],[IX+dd]	LD [IY],[IX+dd]	SRA A	RR A	SEP	XOR B,#nn	LD A,NB	MLT	JRS F0,rr	CARS F0,rr
9	ADC A,[IY+dd]	SBC A,[IY+dd]	OR A,[IY+dd]	XOR A,[IY+dd]	LD B,[IY+dd]	LD H,[IY+dd]	LD [IX],[IY+dd]	LD [IY],[IY+dd]	SRA B	RR B		XOR L,#nn	LD A,EP	DIV	JRS F1,rr	CARS F1,rr
A	ADC A,[IX+L]	SBC A,[IX+L]	OR A,[IX+L]	XOR A,[IX+L]	LD B,[IX+L]	LD H,[IX+L]	LD [IX],[IX+L]	LD [IY],[IX+L]	SRA [BR:*ll*]	RR [BR:*ll*]		XOR H,#nn	LD A,XP		JRS F2,rr	CARS F2,rr
B	ADC A,[IY+L]	SBC A,[IY+L]	OR A,[IY+L]	XOR A,[IY+L]	LD B,[IY+L]	LD H,[IY+L]	LD [IX],[IY+L]	LD [IY],[IY+L]	SRA [HL]	RR [HL]	Undefined Code Area		LD A,YP		JRS F3,rr	CARS F3,rr
C	ADC [HL],A	SBC [HL],A	OR [HL],A	XOR [HL],A	LD [IX+dd],B	LD [IX+dd],H			SRL A	RRC A		CP B,#nn	LD NB,A	Undefined	JRS NF0,rr	CARS NF0,rr
D	ADC [HL],#nn	SBC [HL],#nn	OR [HL],#nn	XOR [HL],#nn	LD [IY+dd],B	LD [IY+dd],H	Undefined Code Area		SRL B	RRC B		CP L,#nn	LD EP,A	Code Area	JRS NF1,rr	CARS NF1,rr
E	ADC [HL],[IX]	SBC [HL],[IX]	OR [HL],[IX]	XOR [HL],[IX]	LD [IX+L],B	LD [IX+L],H			SRL [BR:*ll*]	RRC [BR:*ll*]	HALT	CP H,#nn	LD XP,A		JRS NF2,rr	CARS NF2,rr
F	ADC [HL],[IY]	SBC [HL],[IY]	OR [HL],[IY]	XOR [HL],[IY]	LD [IY+L],B	LD [IY+L],H			SRL [HL]	RRC [HL]	SLP	CP BR,#hh	LD YP,A		JRS NF3,rr	CARS NF3,rr

Operation code map (3/3)

2nd operation code (1st operation code = CF)

L \ H	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	ADD BA,BA		ADD HL,BA		ADD IX,BA		ADC BA,#mmnn	LD BA,[SP+dd]				PUSH A	LD BA,[HL]	LD BA,[IX]	LD BA,BA	LD SP,BA
1	ADD BA,HL		ADD HL,HL		ADD IX,HL		ADC HL,#mmnn	LD HL,[SP+dd]				PUSH B	LD HL,[HL]	LD HL,[IX]	LD BA,HL	LD SP,HL
2	ADD BA,IX	Undefined	ADD HL,IX		ADD IY,BA		SBC BA,#mmnn	LD IX,[SP+dd]				PUSH L	LD IX,[HL]	LD IX,[IX]	LD BA,IX	LD SP,IX
3	ADD BA,IY	Code Area	ADD HL,IY		ADD IY,HL		SBC HL,#mmnn	LD IY,[SP+dd]				PUSH H	LD IY,[HL]	LD IY,[IX]	LD BA,IY	LD SP,IY
4	ADC BA,BA		ADC HL,BA		ADD SP,BA			LD [SP+dd],BA				POP A	LD [HL],BA	LD [IX],BA	LD HL,BA	LD HL,SP
5	ADC BA,HL		ADC HL,HL		ADD SP,HL			LD [SP+dd],HL				POP B	LD [HL],HL	LD [IX],HL	LD HL,HL	LD HL,PC
6	ADC BA,IX		ADC HL,IX		Undefined			LD [SP+dd],IX				POP L	LD [HL],IX	LD [IX],IX	LD HL,IX	Undefined
7	ADC BA,IY		ADC HL,IY		Code Area			LD [SP+dd],IY		Undefined		POP H	LD [HL],IY	LD [IX],IY	LD HL,IY	Code Area
8	SUB BA,BA	CP BA,BA	SUB HL,BA	CP HL,BA	SUB IX,BA		ADD SP,#mmnn	LD SP,[hh*ll*]		Code Area		PUSH ALL		LD BA,[IY]	LD IX,BA	LD BA,SP
9	SUB BA,HL	CP BA,HL	SUB HL,HL	CP HL,HL	SUB IX,HL							PUSH ALE		LD HL,[IY]	LD IX,HL	LD BA,PC
A	SUB BA,IX	CP BA,IX	SUB HL,IX	CP HL,IX	SUB IY,BA		SUB SP,#mmnn							LD IX,[IY]	LD IX,IX	LD IX,SP
B	SUB BA,IY	CP BA,IY	SUB HL,IY	CP HL,IY	SUB IY,HL									LD IY,[IY]	LD IX,IY	
C	SBC BA,BA		SBC HL,BA		SUB SP,BA	CP SP,BA	CP SP,#mmnn	LD [hh*ll*],SP				POP ALL		LD [IY],BA	LD IY,BA	Undefined Code Area
D	SBC BA,HL	Undefined	SBC HL,HL		SUB SP,HL	CP SP,HL						POP ALE		LD [IY],HL	LD IY,HL	
E	SBC BA,IX	Code Area	SBC HL,IX				LD SP,#mmnn							LD [IY],IX	LD IY,IX	LD IY,SP
F	SBC BA,IY		SBC HL,IY											LD [IY],IY	LD IY,IY	Undefined Code

APPENDIX B Instruction List by Addressing Mode

Instruction list by addressing mode (1/12)

8-bit Arithmetic and Logic Operation

Mnemonic	Immediate			Register Direct			Register Indirect			Register Indirect with Displacement			Register Indirect with Index Register			8-bit Absolute			16-bit Absolute		
	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle
ADD	A,#nn	2	2	A,A	1	2	A,[HL]	1	2	A,[IX+dd]	3	4	A,[IX+L]	2	4	A,[BR:*ll*]	2	3	A,[hh*ll*]	3	4
				A,B	1	2	A,[IX]	1	2	A,[IY+dd]	3	4	A,[IY+L]	2	4						
							A,[IY]	1	2												
	[HL],#nn	3	5	[HL],A	2	4	[HL],[IX]	2	5												
							[HL],[IY]	2	5												
ADC	A,#nn	2	2	A,A	1	2	A,[HL]	1	2	A,[IX+dd]	3	4	A,[IX+L]	2	4	A,[BR:*ll*]	2	3	A,[hh*ll*]	3	4
				A,B	1	2	A,[IX]	1	2	A,[IY+dd]	3	4	A,[IY+L]	2	4						
							A,[IY]	1	2												
	[HL],#nn	3	5	[HL],A	2	4	[HL],[IX]	2	5												
							[HL],[IY]	2	5												
SUB	A,#nn	2	2	A,A	1	2	A,[HL]	1	2	A,[IX+dd]	3	4	A,[IX+L]	2	4	A,[BR:*ll*]	2	3	A,[hh*ll*]	3	4
				A,B	1	2	A,[IX]	1	2	A,[IY+dd]	3	4	A,[IY+L]	2	4						
							A,[IY]	1	2												
	[HL],#nn	3	5	[HL],A	2	4	[HL],[IX]	2	5												
							[HL],[IY]	2	5												
SBC	A,#nn	2	2	A,A	1	2	A,[HL]	1	2	A,[IX+dd]	3	4	A,[IX+L]	2	4	A,[BR:*ll*]	2	3	A,[hh*ll*]	3	4
				A,B	1	2	A,[IX]	1	2	A,[IY+dd]	3	4	A,[IY+L]	2	4						
							A,[IY]	1	2												
	[HL],#nn	3	5	[HL],A	2	4	[HL],[IX]	2	5												
							[HL],[IY]	2	5												
AND	A,#nn	2	2	A,A	1	2	A,[HL]	1	2	A,[IX+dd]	3	4	A,[IX+L]	2	4	A,[BR:*ll*]	2	3	A,[hh*ll*]	3	4
	B,#nn	3	2	A,B	1	2	A,[IX]	1	2	A,[IY+dd]	3	4	A,[IY+L]	2	4						
	L,#nn	3	2				A,[IY]	1	2												
	H,#nn	3	2																		
	SC,#nn	2	2																		
	[BR:*ll*],#nn	3	5	[HL],A	2	4	[HL],[IX]	2	5												
	[HL],#nn	3	5				[HL],[IY]	2	5												
OR	A,#nn	2	2	A,A	1	2	A,[HL]	1	2	A,[IX+dd]	3	4	A,[IX+L]	2	4	A,[BR:*ll*]	2	3	A,[hh*ll*]	3	4
	B,#nn	3	3	A,B	1	2	A,[IX]	1	2	A,[IY+dd]	3	4	A,[IY+L]	2	4						
	L,#nn	3	3				A,[IY]	1	2												
	H,#nn	3	3																		
	SC,#nn	2	3																		
	[BR:*ll*],#nn	3	5	[HL],A	2	4	[HL],[IX]	2	5												
	[HL],#nn	3	5				[HL],[IY]	2	5												

Instruction list by addressing mode (2/12)

8-bit Arithmetic and Logic Operation

	Immediate			Register Direct			Register Indirect			Register Indirect with Displacement			Register Indirect with Index Register			8-bit Absolute			16-bit Absolute		
Mnemonic	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle
XOR	A,#nn	2	2	A,A	1	2	A,[HL]	1	2	A,[IX+dd]	3	4	A,[IX+L]	2	4	A,[BR:*ll*]	2	3	A,[hh*ll*]	3	4
	B,#nn	3	3	A,B	1	2	A,[IX]	1	2	A,[IY+dd]	3	4	A,[IY+L]	2	4						
	L,#nn	3	3				A,[IY]	1	2												
	H,#nn	3	3																		
	SC,#nn	2	3																		
	[BR:*ll*],#nn	3	5	[HL],A	2	4	[HL],[IX]	2	5												
	[HL],#nn	3	5				[HL],[IY]	2	5												
CP	A,#nn	2	2	A,A	1	2	A,[HL]	1	2	A,[IX+dd]	3	4	A,[IX+L]	2	4	A,[BR:*ll*]	2	3	A,[hh*ll*]	3	4
	B,#nn	3	3	A,B	1	2	A,[IX]	1	2	A,[IY+dd]	3	4	A,[IY+L]	2	4						
	L,#nn	3	3				A,[IY]	1	2												
	H,#nn	3	3																		
	BR,#nn	3	3																		
	[BR:*ll*],#nn	3	4	[HL],A	2	3	[HL],[IX]	2	4												
	[HL],nn	3	4				[HL],[IY]	2	4												
BIT	A,#nn	2	2	A,B	1	2															
	B,#nn	2	2																		
	[BR:*ll*],#nn	3	4																		
	[HL],#nn	2	3																		
INC				A	1	2	[HL]	1	3							[BR:*ll*]	2	4			
				B	1	2															
				L	1	2															
				H	1	2															
				BR	1	2															
DEC				A	1	2	[HL]	1	3							[BR:*ll*]	2	4			
				B	1	2															
				L	1	2															
				H	1	2															
				BR	1	2															
CPL				A	2	3	[HL]	2	4							[BR:*ll*]	3	5			
				B	2	3															
NEG				A	2	3	[HL]	2	4							[BR:*ll*]	3	5			
				B	2	3															

Instruction list by addressing mode (3/12)

8-bit Transfer

Mnemonic	Immediate			Register Direct			Register Indirect			Register Indirect with Displacement			Register Indirect with Index Register			8-bit Absolute			16-bit Absolute		
	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle
LD	A,#nn	2	2	A,A	1	1	A,[HL]	1	2	A,[IX+dd]	3	4	A,[IX+L]	2	4	A,[BR:*ll*]	2	3	A,[hh*ll*]	4	5
				A,B	1	1	A,[IX]	1	2	A,[IY+dd]	3	4	A,[IY+L]	2	4						
				A,L	1	1	A,[IY]	1	2												
				A,H	1	1															
				A,BR	2	2															
				A,SC	2	2															
				A,NB	2	2															
				A,EP	2	2															
				A,XP	2	2															
				A,YP	2	2															
	B,#nn	2	2	B,A	1	1	B,[HL]	1	2	B,[IX+dd]	3	4	B,[IX+L]	2	4	B,[BR:*ll*]	2	3	B,[hh*ll*]	4	5
				B,B	1	1	B,[IX]	1	2	B,[IY+dd]	3	4	B,[IY+L]	2	4						
				B,L	1	1	B,[IY]	1	2												
				B,H	1	1															
	L,#nn	2	2	L,A	1	1	L,[HL]	1	2	L,[IX+dd]	3	4	L,[IX+L]	2	4	L,[BR:*ll*]	2	3	L,[hh*ll*]	4	5
				L,B	1	1	L,[IX]	1	2	L,[IY+dd]	3	4	L,[IY+L]	2	4						
				L,L	1	1	L,[IY]	1	2												
				L,H	1	1															
	H,#nn	2	2	H,A	1	1	H,[HL]	1	2	H,[IX+dd]	3	4	H,[IX+L]	2	4	H,[BR:*ll*]	2	3	H,[hh*ll*]	4	5
				H,B	1	1	H,[IX]	1	2	H,[IY+dd]	3	4	H,[IY+L]	2	4						
				H,L	1	1	H,[IY]	1	2												
				H,H	1	1															
	BR,#hh	2	2	BR,A	2	2															
	SC,#nn	2	3	SC,A	2	3															
	[BR:*ll*],#nn	3	4	[BR:*ll*],A	2	3	[BR:*ll*],[HL]	2	4												
				[BR:*ll*],B	2	3	[BR:*ll*],[IX]	2	4												
				[BR:*ll*],L	2	3	[BR:*ll*],[IY]	2	4												
				[BR:*ll*],H	2	3															
				[hh*ll*],A	4	5															
				[hh*ll*],B	4	5															
				[hh*ll*],L	4	5															
				[hh*ll*],H	4	5															

Instruction list by addressing mode (4/12)

8-bit Transfer

Mnemonic	Immediate			Register Direct			Register Indirect			Register Indirect with Displacement			Register Indirect with Index Register			8-bit Absolute			16-bit Absolute		
	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle
LD	[HL],#nn	2	3	[HL],A	1	2	[HL],[HL]	1	3	[HL],[IX+dd]	3	5	[HL],[IX+L]	2	5	[HL],[BR:*ll*]	2	4			
				[HL],B	1	2	[HL],[IX]	1	3	[HL],[IY+dd]	3	5	[HL],[IY+L]	2	5						
				[HL],L	1	2	[HL],[IY]	1	3												
				[HL],H	1	2															
	[IX],#nn	2	3	[IX],A	1	2	[IX],{HL]	1	3	[IX],[IX+dd]	3	5	[IX],[IX+L]	2	5	[IX],[BR:*ll*]	2	4			
				[IX],B	1	2	[IX],[IX]	1	3	[IX],[IY+dd]	3	5	[IX],[IY+L]	2	5						
				[IX],L	1	2	[IX],[IY]	1	3												
				[IX],H	1	2															
	[IY],#nn	2	3	[IY],A	1	2	[IY],[HL]	1	3	[IY],[IX+dd]	3	5	[IY],[IX+L]	2	5	[IY],[BR:*ll*]	2	4			
				[IY],B	1	2	[IY],[IX]	1	3	[IY],[IY+dd]	3	5	[IY],[IY+L]	2	5						
				[IY],L	1	2	[IY],[IY]	1	3												
				[IY],H	1	2															
				[IX+dd],A	3	4															
				[IX+dd],B	3	4															
				[IX+dd],L	3	4															
				[IX+dd],H	3	4															
				[IY+dd],A	3	4															
				[IY+dd],B	3	4															
				[IY+dd],L	3	4															
				[IY+dd],H	3	4															
				[IX+L],A	2	4															
				[IX+L],B	2	4															
				[IX+L],L	2	4															
				[IX+L],H	2	4															
				[IY+L],A	2	4															
				[IY+L],B	2	4															
				[IY+L],L	2	4															
				[IY+L],H	2	4															
	NB,#bb	3	4	NB,A	2	3															
	EP,#pp	3	3	EP,A	2	2															
	XP,#pp	3	3	XP,A	2	2															
	YP,#pp	3	3	YP,A	2	2															
EX				A,B	1	2	A,[HL]	1	3												
SWAP				A	1	2	[HL]	1	3												

Instruction list by addressing mode (5/12)

Rotate/Shift

	Immediate			Register Direct			Register Indirect			Register Indirect with Displacement			Register Indirect with Index Register			8-bit Absolute			16-bit Absolute		
Mnemonic	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle
RL				A	2	3	[HL]	2	4							[BR:*ll*]	3	5			
				B	2	3															
RLC				A	2	3	[HL]	2	4							[BR:*ll*]	3	5			
				B	2	3															
RR				A	2	3	[HL]	2	4							[BR:*ll*]	3	5			
				B	2	3															
RRC				A	2	3	[HL]	2	4							[BR:*ll*]	3	5			
				B	2	3															
SLA				A	2	3	[HL]	2	4							[BR:*ll*]	3	5			
				B	2	3															
SLL				A	2	3	[HL]	2	4							[BR:*ll*]	3	5			
				B	2	3															
SRA				A	2	3	[HL]	2	4							[BR:*ll*]	3	5			
				B	2	3															
SRL				A	2	3	[HL]	2	4							[BR:*ll*]	3	5			
				B	2	3															

Instruction list by addressing mode (6/12)

16-bit Arithmetic Operation

Mnemonic	Immediate			Register Direct			Register Indirect			Register Indirect with Displacement			Register Indirect with Index Register			8-bit Absolute			16-bit Absolute		
	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle
ADD	BA,#mmnn	3	3	BA,BA	2	4															
				BA,HL	2	4															
				BA,IX	2	4															
				BA,IY	2	4															
	HL,#mmnn	3	3	HL,BA	2	4															
				HL,HL	2	4															
				HL,IX	2	4															
				HL,IY	2	4															
	IX,#mmnn	3	3	IX,BA	2	4															
				IX,HL	2	4															
	IY,#mmnn	3	3	IY,BA	2	4															
				IY,HL	2	4															
	SP,#mmnn	4	4	SP,BA	2	4															
				SP,HL	2	4															
ADC	BA,#mmnn	4	4	BA,BA	2	4															
				BA,HL	2	4															
				BA,IX	2	4															
				BA,IY	2	4															
	HL,#mmnn	4	4	HL,BA	2	4															
				HL,HL	2	4															
				HL,IX	2	4															
				HL,IY	2	4															
SUB	BA,#mmnn	3	3	BA,BA	2	4															
				BA,HL	2	4															
				BA,IX	2	4															
				BA,IY	2	4															
	HL,#mmnn	3	3	HL,BA	2	4															
				HL,HL	2	4															
				HL,IX	2	4															
				HL,IY	2	4															
	IX,#mmnn	3	3	IX,BA	2	4															
				IX,HL	2	4															
	IY,#mmnn	3	3	IY,BA	2	4															
				IY,HL	2	4															
	SP,#mmnn	4	4	SP,BA	2	4															
				SP,HL	2	4															

Instruction list by addressing mode (7/12)

16-bit Arithmetic Operation

	Immediate			Register Direct			Register Indirect			Register Indirect with Displacement			Register Indirect with Index Register			8-bit Absolute			16-bit Absolute		
Mnemonic	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle
SBC	BA,#mmnn	4	4	BA,BA	2	4															
				BA,HL	2	4															
				BA,IX	2	4															
				BA,IY	2	4															
	HL,#mmnn	4	4	HL,BA	2	4															
				HL,HL	2	4															
				HL,IX	2	4															
				HL,IY	2	4															
CP	BA,#mmnn	3	3	BA,BA	2	4															
				BA,HL	2	4															
				BA,IX	2	4															
				BA,IY	2	4															
	HL,#mmnn	3	3	HL,BA	2	4															
				HL,HL	2	4															
				HL,IX	2	4															
				HL,IY	2	4															
	IX,#mmnn	3	3																		
	IY,#mmnn	3	3																		
	SP,#mmnn	4	4	SP,BA	2	4															
				SP,HL	2	4															
INC				BA	1	2															
				HL	1	2															
				IX	1	2															
				IY	1	2															
				SP	1	2															
DEC				BA	1	2															
				HL	1	2															
				IX	1	2															
				IY	1	2															
				SP	1	2															

Instruction list by addressing mode (8/12)

16-bit Transfer

	Immediate			Register Direct			Register Indirect			Register Indirect with Displacement			Register Indirect with Index Register			8-bit Absolute			16-bit Absolute		
Mnemonic	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle
LD	BA,#mmnn	3	3	BA,BA	2	2	BA,[HL]	2	5	BA,[SP+dd]	3	6							BA,[hh*ll*]	3	5
				BA,HL	2	2	BA,[IX]	2	5												
				BA,IX	2	2	BA,[IY]	2	5												
				BA,IY	2	2															
				BA,SP	2	2															
				BA,PC	2	2															
	HL,#mmnn	3	3	HL,BA	2	2	HL,[HL]	2	5	HL,[SP+dd]	3	6							HL,[hh*ll*]	3	5
				HL,HL	2	2	HL,[IX]	2	5												
				HL,IX	2	2	HL,[IY]	2	5												
				HL,IY	2	2															
				HL,SP	2	2															
				HL,PC	2	2															
	IX,#mmnn	3	3	IX,BA	2	2	IX,[HL]	2	5	IX,[SP+dd]	3	6							IX,[hh*ll*]	3	5
				IX,HL	2	2	IX,[IX]	2	5												
				IX,IX	2	2	IX,[IY]	2	5												
				IX,IY	2	2															
				IX,SP	2	2															
	IY,#mmnn	3	3	IY,BA	2	2	IY,[HL]	2	5	IY,[SP+dd]	3	6							IY,[hh*ll*]	3	5
				IY,HL	2	2	IY,[IX]	2	5												
				IY,IX	2	2	IY,[IY]	2	5												
				IY,IY	2	2															
				IY,SP	2	2															
	SP,#mmnn	4	4	SP,BA	2	2													SP,[hh*ll*]	4	6
				SP,HL	2	2															
				SP,IX	2	2															
				SP,IY	2	2															
				[hh*ll*],BA	3	5															
				[hh*ll*],HL	3	5															
				[hh*ll*],IX	3	5															
				[hh*ll*],IY	3	5															
				[hh*ll*],SP	4	6															

Instruction list by addressing mode (9/12)

16-bit Transfer

Mnemonic	Immediate			Register Direct			Register Indirect			Register Indirect with Displacement			Register Indirect with Index Register			8-bit Absolute			16-bit Absolute		
	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle
LD				[HL],BA	2	5															
				[HL],HL	2	5															
				[HL],IX	2	5															
				[HL],IY	2	5															
				[IX],BA	2	5															
				[IX],HL	2	5															
				[IX],IX	2	5															
				[IX],IY	2	5															
				[IY],BA	2	5															
				[IY],HL	2	5															
				[IY],IX	2	5															
				[IY],IY	2	5															
				[SP+dd],BA	3	6															
				[SP+dd],HL	3	6															
				[SP+dd],IX	3	6															
				[SP+dd],IY	3	6															
EX				BA,HL	1	3															
				BA,IX	1	3															
				BA,IY	1	3															
				BA,SP	1	3															

Instruction list by addressing mode (10/12)

Branch

	8-bit Indirect			16-bit Indirect			Sined 8-bit PC Relative			Sined 16-bit PC Relative			Register Direct			Others		
Mnemonic	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle	Operand	Byte	Cycle
JRS							rr	2	2									
							C,rr	2	2									
							NC,rr	2	2									
							Z,rr	2	2									
							NZ,rr	2	2									
							LT,rr	3	3									
							LE,rr	3	3									
							GT,rr	3	3									
							GE,rr	3	3									
							V,rr	3	3									
							NV,rr	3	3									
							P,rr	3	3									
							M,rr	3	3									
							F0,rr	3	3									
							F1,rr	3	3									
							F2,rr	3	3									
							F3,rr	3	3									
							NF0,rr	3	3									
							NF1,rr	3	3									
							NF2,rr	3	3									
							NF3,rr	3	3									
JRL										qqrr	3	3						
										C,qqrr	3	3						
										NC,qqrr	3	3						
										Z,qqrr	3	3						
										NZ,qqrr	3	3						
JP	[kk]	2	4										HL	1	2			
DJR							NZ,rr	2	4									

Instruction list by addressing mode (11/12)

Cycle (Min.: Minimum mode, Max.: Maximum mode, Skip: Not branched)

Branch	8-bit Indirect				16-bit Indirect				Sined 8-bit PC Relative					Sined 16-bit PC Relative					Others			
			Cycle				Cycle				Cycle					Cycle					Cycle	
Mnemonic	Operand	Byte	Min.	Max.	Operand	Byte	Min.	Max.	Operand	Byte	Min.	Max.	Skip	Operand	Byte	Min.	Max.	Skip	Operand	Byte	Min.	Max.
CARS									rr	2	4	5	–									
									C,rr	2	4	5	2									
									NC,rr	2	4	5	2									
									Z,rr	2	4	5	2									
									NZ,rr	2	4	5	2									
									LT,rr	3	5	6	3									
									LE,rr	3	5	6	3									
									GT,rr	3	5	6	3									
									GE,rr	3	5	6	3									
									V,rr	3	5	6	3									
									NV,rr	3	5	6	3									
									P,rr	3	5	6	3									
									M,rr	3	5	6	3									
									F0,rr	3	5	6	3									
									F1,rr	3	5	6	3									
									F2,rr	3	5	6	3									
									F3,rr	3	5	6	3									
									NF0,rr	3	5	6	3									
									NF1,rr	3	5	6	3									
									NF2,rr	3	5	6	3									
									NF3,rr	3	5	6	3									
CARL														qqrr	3	5	6	–				
														C,qqrr	3	5	6	3				
														NC,qqrr	3	5	6	3				
														Z,qqrr	3	5	6	3				
														NZ,qqrr	3	5	6	3				
CALL					[hh*ll*]	3	7	8														
RET																				2	3	4
RETE																				1	4	5
RETS																				1	5	6
INT	[kk]	2	7	8																		

Instruction list by addressing mode (12/12)

Multiplication and Division

	Implide		
Mnemonic	Operand	Byte	Cycle
MLT		2	12
DIV		2	13

Auxiliary Operation

	Implide		
Mnemonic	Operand	Byte	Cycle
PACK		1	2
UPCK		1	2
SEP		2	3

System Control

Mnemonic	Operand	Byte	Cycle
NOP		1	2
HALT		2	3
SLP		2	3

Stack Control

	Register Direct			Implide		
Mnemonic	Operand	Byte	Cycle	Operand	Byte	Cycle
POP	A	2	3	ALL	2	11
	B	2	3			
	L	2	3			
	H	2	3			
	BR	1	2	ALE	2	14
	SC	1	2			
	EP	1	2			
	IP	1	3			
	BA	1	3			
	HL	1	3			
	IX	1	3			
	IY	1	3			
PUSH	A	2	3	ALL	2	12
	B	2	3			
	L	2	3			
	H	2	3			
	BR	1	3	ALE	2	15
	SC	1	3			
	EP	1	3			
	IP	1	4			
	BA	1	4			
	HL	1	4			
	IX	1	4			
	IY	1	4			

APPENDIX C Instruction Index

LD: 8-bit Load

LD: 8-bit Load

XOR: Exclusive OR

EPSON International Sales Operations

AMERICA

EPSON ELECTRONICS AMERICA, INC.

- HEADQUARTERS -
1960 E. Grand Avenue
El Segundo, CA 90245, U.S.A.
Phone: +1-310-955-5300 Fax: +1-310-955-5400

- SALES OFFICES -

West
150 River Oaks Parkway
San Jose, CA 95134, U.S.A.
Phone: +1-408-922-0200 Fax: +1-408-922-0238

Central
1450 East American Lane, Suite 1550
Schaumburg, IL 60173, U.S.A.
Phone: +1-847-517-7667 Fax: +1-847-517-7601

101 Virginia Street, Suite 290
Crystal Lake, IL 60014, U.S.A.
Phone: +1-815-455-7630 Fax: +1-815-455-7633

Northeast
301 Edgewater Place, Suite 120
Wakefield, MA 01880, U.S.A.
Phone: +1-781-246-3600 Fax: +1-781-246-5443

Southeast
4300 Six Forks Road, Suite 430
Raleigh, NC 27609, U.S.A.
Phone: +1-919-781-7667 Fax: +1-919-781-6778

1700 Spinnaker Drive
Alpharetta, GA 30005, U.S.A.
Phone: +1-770-754-4872 Fax: +1-770-753-0601

EUROPE

EPSON EUROPE ELECTRONICS GmbH

- HEADQUARTERS -
Riesstrasse 15
80992 Muenchen, GERMANY
Phone: +49-(0)89-14005-0 Fax: +49-(0)89-14005-110

- GERMANY -
SALES OFFICE
Breidenbachstrasse 46
D-51373 Leverkusen, GERMANY
Phone: +49-(0)214-83070-0 Fax: +49-(0)214-83070-10

- UNITED KINGDOM -
UK BRANCH OFFICE
G6 Doncastle House, Doncastle Road
Bracknell, Berkshire RG12 8PE, ENGLAND
Phone: +44-(0)1344-381700 Fax: +44-(0)1344-381701

- FRANCE -
FRENCH BRANCH OFFICE
1 Avenue de l' Atlantique, LP 915 Les Conquerants
Z.A. de Courtaboeuf 2, F-91976 Les Ulis Cedex, FRANCE
Phone: +33-(0)1-64862350 Fax: +33-(0)1-64862355

ASIA

- HONG KONG, CHINA -
EPSON HONG KONG LTD.
20/F., Harbour Centre, 25 Harbour Road
Wanchai, HONG KONG
Phone: +852-2585-4600 Fax: +852-2827-4346
Telex: 65542 EPSCO HX

- CHINA -
SHANGHAI EPSON ELECTRONICS CO., LTD.
4F, Bldg., 27, No. 69, Gui Jing Road
Caohejing, Shanghai, CHINA
Phone: 21-6485-5552 Fax: 21-6485-0775

- TAIWAN, R.O.C. -
EPSON TAIWAN TECHNOLOGY & TRADING LTD.
10F, No. 287, Nanking East Road, Sec. 3
Taipei, TAIWAN, R.O.C.
Phone: 02-2717-7360 Fax: 02-2712-9164
Telex: 24444 EPSONTB

HSINCHU OFFICE
13F-3, No. 295, Kuang-Fu Road, Sec. 2
HsinChu 300, TAIWAN, R.O.C.
Phone: 03-573-9900 Fax: 03-573-9169

- SINGAPORE -
EPSON SINGAPORE PTE., LTD.
No. 1 Temasek Avenue, #36-00
Millenia Tower, SINGAPORE 039192
Phone: +65-337-7911 Fax: +65-334-2716

- KOREA -
SEIKO EPSON CORPORATION
KOREA OFFICE
50F, KLI 63 Bldg., 60 Yoido-Dong
Youngdeungpo-Ku, Seoul, 150-010, KOREA
Phone: 02-784-6027 Fax: 02-767-3677

- JAPAN -
SEIKO EPSON CORPORATION
ELECTRONIC DEVICES MARKETING DIVISION

Electronic Device Marketing Department
IC Marketing & Engineering Group
421-8, Hino, Hino-shi, Tokyo 191-8501, JAPAN
Phone: +81-(0)42-587-5816 Fax: +81-(0)42-587-5624

ED International Marketing Department I
(Europe & U.S.A.)
421-8, Hino, Hino-shi, Tokyo 191-8501, JAPAN
Phone: +81-(0)42-587-5812 Fax: +81-(0)42-587-5564

ED International Marketing Department II
(Asia)
421-8, Hino, Hino-shi, Tokyo 191-8501, JAPAN
Phone: +81-(0)42-587-5814 Fax: +81-(0)42-587-5110

ENERGY
SAVING
EPSON

In pursuit of **“Saving” Technology**, Epson electronic devices.
Our lineup of semiconductors, liquid crystal displays and quartz devices
assists in creating the products of our customers’ dreams.
Epson IS energy savings.

EPSON

SEIKO EPSON CORPORATION

ELECTRONIC DEVICES MARKETING DIVISION

■ Electronic devices information on Epson WWW server

http://www.epson.co.jp

First issue MARCH 1993, Printed APRIL 1999 in Japan Ⓜ B