Pin 4 (SOD) : It is the output pin for Serial Output Data (SOD).

Pin 5 (SID) : It is the input pin for Serial Input Data (SID).

Pins 6 to 10 : These pins represent the input pins for five interrupt signals, namely, TRAP, RST 7.5, RST 6.5, RST 5.5, and INTR. The TRAP signal is a nonmaskable interrupt, and has the greatest priority. The signals RST 5.5, RST 6.5, and RST 7.5 restart, interrupt and transfer the program control to specific memory locations. They have higher priorities than the INTR (Interrupt Request) which is a general-purpose interrupt. The priority increases from 5.5 to 7.5.

Pin 11 : The microprocessor acknowledges an interrupt by the $\overline{\text{INTA}}$ (Interrupt Acknowledgement) signals appearing at this pin.

Pins 12 to 19 : The lines AD_0 to AD_7 are used as lower order address bus as well as data bus.

Pin 20 (V_{SS}) : It acts as the system ground.

Pins 21 to 28 : The lines A_8 to A_{15} are used as higher order address bus.

Pins 29, 33, and 34 (S_0, S_1, and IO/$\overline{M}$) : They carry 'status signals' which identify different operations.

Pin 30 (ALE) : The signal at this pin indicates the Address Latch Enable (ALE) and is mainly used to latch the low-order address from the multiplexed bus.

Pin 31 ($\overline{\text{WR}}$) : This pin carries a low $\overline{\text{WR}}$ control signal which activates a write operation. The bar indicates an active low signal.

Pin 32 ($\overline{\text{RD}}$) : It carries a low $\overline{\text{RD}}$ control signal which enables a read operation. The bar represents an active low signal.

Pin 35 (READY) : Slower peripherals are synchronized with the microprocessor with the signal at this pin.

Pin 36 ($\overline{\text{RESET IN}}$) : If the signal at this pin is low, the program counter in the microprocessor is reset.

Pin 37 [CLK (OUT)] : The signal at this pin is a clock output and can be used for other devices.

Pins 38 and 39 (HLDA and HOLD) : When HOLD is excited at the pin 39, the microprocessor dictates the external peripheral to use the buses. The HLDA signal appearing at pin 38 acknowledges the HOLD request.

Pin 40 (V_{CC}) : The +5V power supply is connected to this pin.

Programming a microprocessor

A **program** contains a set of instructions written in a logical manner to direct a microprocessor to perform the specific operations in a particular sequence. The set of instructions can be divided into five groups: data transfer operation, arithmetic operation, logic operation, branching operation, and Stack, I/O and machine control operations.

An **instruction** is a command to the microprocessor to do the desired job on given data. An instruction has two parts: the **operation code** (abbreviated 'opcode') and the **operand.** The opcode contains the job to be performed and the operand contains the data to be operated on. The instruction can be written in a hexagonal code, and as a memory aid, a symbolic code, called a **mnemonic** is given for each instruction.

When programs are written in mnemonics, the medium of communication with the computer is said to be in the **assembly language.** If the instructions for communication with the computer are in binary form specifically for each computer, the medium of communication is said to be in the **machine language.** An **assembler** is a computer program that translates a program written in the assembly language into the machine language.

The program format used in the assembly language has five columns, viz., memory address, machine code, instruction mnemonic (opcode and operand), and comments.

Memory address : These are 16-bit addresses of the user read/write (R/W) memory storing the machine code of the program. Each address contains 4 digits. The first two digits show the page number of the R/W memory, and the last two digits indicate the line number. For example, if the user memory begins with 0200, the page number is 02H. Every instruction has a memory address.

Machine code : The hexadecimal machine code for each instruction is found from the 8085 instruction set to translate the assembly language program into machine language. The machine codes are the hexadecimal numbers which are entered in the memory addresses through the key board. The monitor program stored in the Read Only Memory (ROM) converts these hexadecimal numbers into binary forms which are stored in the R/W memory.

Mnemonics : The mnemonic in an instruction contains an opcode and an operand. The mnemonics are used to write programs in the 8085 assembly language and are translated manually into the machine code as outlined above.

Comments : They explain the instructions and help in understanding the program logic. Comments are separated by a semicolon from the instruction.

Set of instructions for the Intel 8085

The complete set of mnemonics and the corresponding opcodes for the Intel 8085 microprocessor is shown in Tabel 3.MC 1. Out of these instructions we give below some commonly used instructions and the tasks performed by them.

(i) **Some data transfer operations**

MOV B, C : This mnemonic gives the instruction for data movement from the register C to the register B. This instruction commands the microprocessor to copy the contents of register C into the register B. The previous contents of the register B are lost, and the contents of the register C remains unchanged. Here MOV is the opcode and B, C is the operand. The Hex code for this instruction is 41.

Instructions like MOV C, D; MOV C, B; MOV B, D etc. have similar meanings, but they have different machine (Hex) codes.

MOV E, M : This instruction asks the microprocessor to copy the contents of the memory location M into the register E. The memory location is indicated by the contents of the HL register. The Hex code for the instruction is 5E.

Example : Let the binary number in the register H be 0010 0000. Its hexadecimal equivalent is 20H, where H signifies that the number is in the hexadecimal system. If the contents of the register L are 0100 0000, i.e., 40H, the address of the memory M is 2040, i.e., the contents of the HL register. Suppose that the memory location 2040 contains the number 1101 1001. Then the contents of the register E after the performance of the MOV E, M instruction will be 1101 1001. The HL register pair is the **address pointer.** The register H holds the higher 8-bits and the register L holds the lower 8-bits of the 16-bit address of the memory.

Instructions like MOV B, M; MOV C, M; MOV D, M etc. have similar significances, but they have other Hex (machine) codes.

MOV M, B : This instruction causes the data to be copied from the register B into the memory whose address is given by the contents of the HL register. The Hex code for this instruction is 70. Instructions such as MOV M, C; MOV M, D; MOV M, H etc. are similar.

Mnemonic
ACI byte
ADC A
ADC B
ADC C
ADC D
ADC E
ADC H
ADC L
ADC M
ADD A
ADD B
ADD C
ADD D
ADD E
ADD H
ADD L
ADD M
ADI byte
ANA A
ANA B
ANA C
ANA D
ANA E
ANA H
ANA L
ANA M
ANI byte
CALL address
CC address
CM address
CMA
CMC
CMP A
CMP B
CMP C
CMP D
CMP E
CMP H
CMP L
CMP M
CNC address
CNZ address
CP address
CPE address
CPI byte
CPO address
CZ address
DAA
DAD B
DAD D
DAD H

Table 3.MC 1
List of Instructions for Intel 8085

Mnemonic	Machine code	Mnemonic	Machine code	Mnemonic	Machine code	Mnemonic	Machine code	Mnemonic	Machine code
ACI byte	CE	DAD SP	39	MOV A,H	7C	MOV M,A	77	RST 3	DF
ADC A	8F	DCR A	3D	MOV A,L	7D	MOV M,B	70	RST 4	E7
ADC B	88	DCR B	05	MOV A,M	7E	MOV M,C	71	RST 5	EF
ADC C	89	DCR C	0D	MOV B,A	47	MOV M,D	72	RST 6	F7
ADC D	8A	DCR D	15	MOV B,B	40	MOV M,E	73	RST 7	FF
ADC E	8B	DCR E	1D	MOV B,C	41	MOV M,H	74	RZ	C8
ADC H	8C	DCR H	25	MOV B,D	42	MOV M,L	75	SBB A	9F
ADC L	8D	DCR L	2D	MOV B,E	43	MVI A,byte	3E	SBB B	98
ADC M	8E	DCR M	35	MOV B,H	44	MVI B,byte	06	SBB C	99
ADD A	87	DCX B	0B	MOV B,L	45	MVI C,byte	0E	SBB D	9A
ADD B	80	DCX D	1B	MOV B,M	46	MVI D,byte	16	SBB E	9B
ADD C	81	DCX H	2B	MOV C,A	4F	MVI E,byte	1E	SBB H	9C
ADD D	82	DCX SP	3B	MOV C,B	48	MVI H,byte	26	SBB L	9D
ADD E	83	DI	F3	MOV C,C	49	MVI L,byte	2E	SBB M	9E
ADD H	84	EI	FB	MOV C,D	4A	MVI M,byte	36	SBI byte	DE
ADD L	85	HLT	76	MOV C,E	4B	NOP	00	SHLD address	22
ADD M	86	IN byte	DB	MOV C,H	4C	ORA A	B7	SIM	30
ADI byte	C6	INR A	3C	MOV C,L	4D	ORA B	B0	SPHL	F9
ANA A	A7	INR B	04	MOV C,M	4E	ORA C	B1	STA address	32
ANA B	A0	INR C	0C	MOV D,A	57	ORA D	B2	STAX B	02
ANA C	A1	INR D	14	MOV D,B	50	ORA E	B3	STAX D	12
ANA D	A2	INR E	1C	MOV D,C	51	ORA H	B4	STC	37
ANA E	A3	INR H	24	MOV D,D	52	ORA L	B5	SUB A	97
ANA H	A4	INR L	2C	MOV D,E	53	ORA M	B6	SUB B	90
ANA L	A5	INR M	34	MOV D,H	54	ORI byte	F6	SUB C	91
ANA M	A6	INX B	03	MOV D,L	55	OUT byte	D3	SUB D	92
ANI byte	E6	INX D	13	MOV D,M	56	PCHL	E9	SUB E	93
CALL address	CD	INX H	23	MOV E,A	5F	POP B	C1	SUB H	94
CC address	DC	INX SP	33	MOV E,B	58	POP D	D1	SUB L	95
CM address	FC	JC address	DA	MOV E,C	59	POP H	E1	SUB M	96
CMA	2F	JM address	FA	MOV E,D	5A	POP PSW	F1	SUI byte	D6
CMC	3F	JMP address	C3	MOV E,E	5B	PUSH B	C5	XCHG	EB
CMP A	BF	JNC address	D2	MOV E,H	5C	PUSH D	D5	XRA A	AF
CMP B	B8	JNZ address	C2	MOV E,L	5D	PUSH H	E5	XRA B	A8
CMP C	B9	JP address	F2	MOV E,M	5E	PUSH PSW	F5	XRA C	A9
CMP D	BA	JPE address	EA	MOV H,A	67	RAL	17	XRA D	AA
CMP E	BB	JPO address	E2	MOV H,B	60	RAR	1F	XRA E	AB
CMP H	BC	JZ address	CA	MOV H,C	61	RC	D8	XRA H	AC
CMP L	BD	LDA address	3A	MOV H,D	62	RET	C9	XRA L	AD
CMP M	BE	LDAX B	0A	MOV H,E	63	RIM	20	XRA M	AE
CNC address	D4	LDAX D	1A	MOV H,H	64	RLC	07	XRI byte	EE
CNZ address	C4	LHLD address	2A	MOV H,L	65	RM	F8	XTHL	E3
CP address	F4	LXI B, dble	01	MOV H,M	66	RNC	D0		
CPE address	EC	LXI D, dble	11	MOV L,A	6F	RNZ	C0		
CPI byte	FE	LXI H, dble	21	MOV L,B	68	RP	F0		
CPO address	E4	LXI SP, dble	31	MOV L,C	69	RPE	E8		
CZ address	CC	MOV A,A	7F	MOV L,D	6A	RPO	E0		
DAA	27	MOV A,B	78	MOV L,E	6B	RRC	0F		
DAD B	09	MOV A,C	79	MOV L,H	6C	RST 0	C7		
DAD D	19	MOV A,D	7A	MOV L,L	6D	RST 1	CF		
DAD H	29	MOV A,E	7B	MOV L,M	6E	RST 2	D7		

MVI B, 48H : This mnemonic stands for the Move Immediate instruction. It is a 2-byte instruction asking the microprocessor to load the register B with the data 48H. Here the opcode is MVI and the operand is B, 48H. In a 2-byte instruction, the first byte denotes the operation code and the second byte indicates the operand. So the Hex code here is 06 48H.

Similar instructions are MVI A, 32H having the Hex code 3E 32H, MVI L, 54H having the Hex code 2E 54H; and so on. In Table 3.MC 1, the data in the instructions with opcode MVI are denoted by 'byte'.

MVI M, A5 : This is another Move Immediate instruction which causes the movement of the data A5 to the memory whose address is contained in the HL register pair.

LXI R_p, 16-bit (dble) : This instruction commands the microprocessor to load 16-bit data in a register pair R_p. Only the high order is mentioned here; for example, LXI H means HL register pair.

STA address : The task implied by this instruction is to copy the data from the accumulator A in the memory location specified by the 16-bit address.

LDA address : Copy the data into the accumulator A from the memory whose location is specified by the 16-bit address.

STAX R_p : Copy the data from the accumulator into the memory whose address is specified by the register pair R_p (BC or DE).

LDAX R_p : Copy the data into the accumulator A from the memory location specified by the register pair R_p (BC or DE).

XCHG : Exchange the contents of the register pair HL with those of the register pair DE.

(ii) Some arithmetic operations

ADD R : This mnemonic asks the microprocessor to add the contents of the register R to the contents of the accumulator A. Here the opcode is ADD and the operand is R. The result of the addition is stored in A. In Table 3.MC 1, R denotes A, B, C, D, E, H, L.

ADD M : Add the contents of the memory location whose address is stored in the HL register pair to the contents of the accumulator A and store the result in A.

ADI byte : This mnemonic is Add Immediate Instruction. It is a two byte instruction asking the microprocessor to add the 8-bit data (represented by 'byte') to the contents of the accumulator A. The result is stored in A.

ADC R : Add the contents of the register R with carry status (given by a previous operation) to the contents of the accumulator A, and store the result in A. In Table 3.MC 1, the general symbol R denotes A, B, C, D, E, H, and L.

ADC M : Add the data in the memory M whose address is specified by the HL pair to the contents of the accumulator A with carry.

ACI byte : Add Immediate the data (represented by 'byte') to the contents of the accumulator A with carry.

SUB R : Subtract the contents of the register R (i.e., A, B, C, D, E, H, L) from the contents of the accumulator A, and store the result in A.

SUB M : Subtract the contents of the memory location specified by the register pair HL from the accumulator contents. Store the result in A.

SUI byte : Subtract immediate the 8-bit data denoted by 'byte' from the contents of the accumulator A and store the result in A.

SBB R	:	Subtract the contents of the register R (i.e., A, B, C, D, E, H, L) from the contents of A with borrow. If the carry flag is set by a previous operation then subtract 1 plus the contents of R from the contents of A.
SBI byte	:	Subtract Immediate the 8-bit data (denoted by 'byte') from the contents of A with borrow.
INR R	:	Increment the contents of the register R (i.e., A, B, C, D, E, H, L) by one.
INR M	:	Increment the contents of the memory location whose address is specified by the HL pair.
INX R_p	:	Increment the contents of a register pair R_p (i.e., BC, DE, HL, SP).
DCR R	:	Decrement the contents of the register R (i.e., A, B, C, D, E, H, L) by one.
DCR M	:	Decrement the contents of the memory location whose address is specified by the HL pair by one.
DCX R_p	:	Decrement the contents of the register pair R_p (i.e., BC, DE, HL, SP) by one.
DAA	:	This is the mnemonic for Decimal Adjust Accumulator. Its task is to adjust the result of addition of BCD numbers.

(iii) Some logic operations

ANA R	:	Logically AND bitwise the contents of the register R (i.e., A, B, C, D, E, H, L) with the contents of the accumulator. Store the result in the accumulator.
ANA M	:	Logically AND bitwise the contents of the memory location whose address is specified by the HL pair with the contents of the accumulator. Store the result in the accumulator.
ANI byte	:	Logically AND Immediate the 8-bit data (byte), bitwise with the contents of the accumulator. The result is in the accumulator.
ORA R	:	Logically OR bitwise the contents of the register R (i.e., A, B, C, D, E, H, L) with those of the accumulator. The result is contained in the accumulator.
ORA M	:	Logically OR bitwise the contents of the memory location whose address is given by the HL pair with the contents of the accumulator. The result is placed in the accumulator.
ORI byte	:	Logically OR Immediate bitwise the 8-bit data (byte) with the contents of the accumulator, and keep the result in the accumulator.
XRA R	:	Exclusive-OR logically bitwise the contents of the register R (i.e., A, B, C, D, E, H, L) with the contents of the accumulator. Hold the result in the accumulator. Note that the instruction XRA A clears the accumulator contents.
XRA M	:	Exclusive-OR logically bitwise the contents of the memory location specified by the HL pair with the contents of the accumulator. Store the result in the accumulator.
XRI byte	:	Exclusive-OR Immediate bitwise the 8-bit data (byte) with the contents of the accumulator. Store the result in the accumulator.
CMA	:	Complement the contents of the accumulator, and store the result in the accumulator.

CMP R/M	:	Compare the contents of the register memory with those of the accumulator for less than, equal to, or greater than.
		If (A) < (R/M), CY flag is set and Zero flag reset.
		If (A) = (R/M), Zero flag is set and CY flag reset.
		If (A) > (R/M), CY and Zero flags are reset.

RLC : Rotate the accumulator to the left. Each bit in A is shifted to the immediate left position; bit D_7 is placed in the position of D_0 and also in the CY flag, as shown in Fig. 3.MC 1(v).

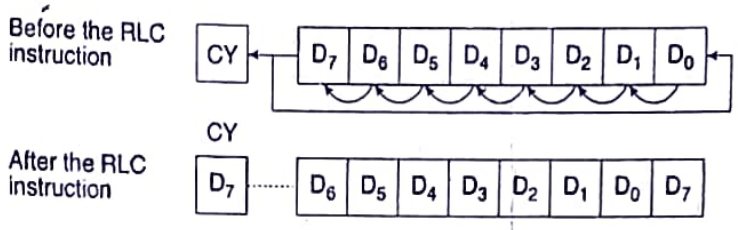

Fig. 3.MC 1(v)

RRC : Rotate the accumulator to the right. Each bit in A is shifted to the adjacent right position. Bit D_0 is placed in the position of D_7 and also in the CY flag [Fig. 3.MC 1(vi)].

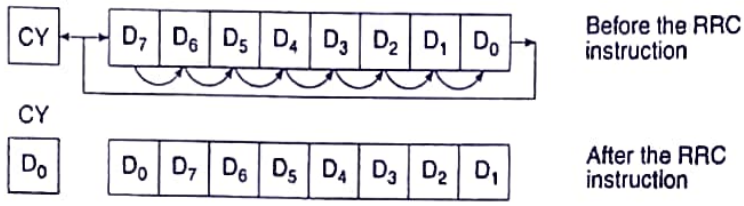

Fig. 3.MC 1(vi)

RAL : Rotate A including the carry to the left. Each bit in A is shifted to the adjacent left position; D_7 is placed in CY and CY is placed in D_0 [Fig. 3.MC 1(vii)].

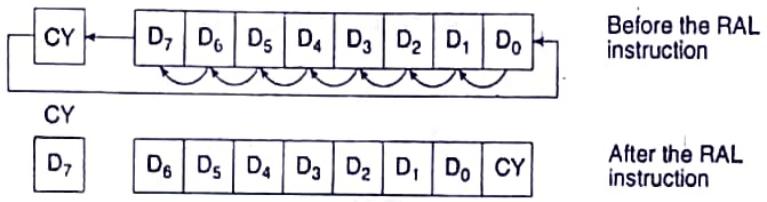

Fig. 3.MC 1(vii)

RAR : Rotate A including the carry to the right. Each bit in A is shifted to the adjacent right position; D_0 is placed in CY, and CY in D_7 [Fig. 3.MC 1(viii)].

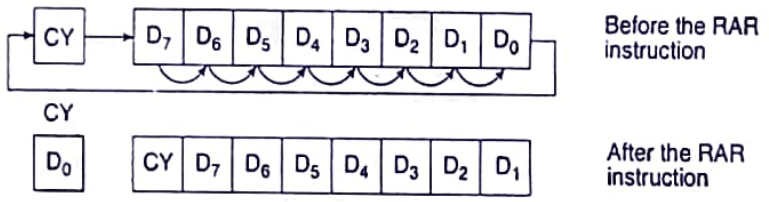

Fig. 3.MC 1(viii)

(iv) Some branching operations

The branch instructions ask the microprocessor to change the sequence of a program unconditionally or subject to some conditions.

JMP addres

Conditional

These instr
specified by the
arithmetic and l
decide whether

The condit
(line number) m
address. The fol
sequence to the

Ins
opcode
JC
JNC
JZ
JNZ
JP
JM
JPE
JPO

Subroutine

If a functi
called a subrou
repetition of th
To implement
appearing in th
the subroutine
explained belo

shifted to the
D_0 and also in

shifted to the
D_7 and also in

shifted to the
ed in D_0 [Fig.

shifted to the
in D_7 [Fig.

f a program

- JMP address : This is an unconditional Jump instruction. It is a 3-byte instruction. The opcode is JMP and the operand is a 16-bit memory address. The second byte specifies the low-order and the third byte specifies the high-order memory address. For example, the instruction asking the microprocessor to go to the memory location 3000H, the mnemonics and the machine code entries are:

Mnemonics	Machine code
JMP 3000H	C3
	00
	30

The 16-bit memory address of the jump location is entered in the opposite order. First, the low-order byte (00H) is entered, and then the high-order byte (30H) is entered.

Conditional Jumps

These instructions cause the microprocessor to take decisions depending on some conditions specified by the flags. The flags are set or reset reflecting the data condition on the completion of arithmetic and logic operations. The conditional Jump instructions, after checking the flag conditions, decide whether the sequence of a program is to be changed.

The conditional Jump instructions are 3-byte instructions. The second byte gives the low-order (line number) memory address whereas the third byte specifies the high-order (page number) memory address. The following conditional Jump instructions ask the microprocessor to transfer the program sequence to the memory location indicated for given conditions.

Instructions		Function
opcode	operand	
JC	16-bit address	Jump when there is a Carry (CY = 1)
JNC	16-bit address	Jump when there is No Carry (CY = 0)
JZ	16-bit address	Jump when the result is Zero (Z = 1)
JNZ	16-bit address	Jump when the result is Not Zero (Z = 0)
JP	16-bit address	Jump when the result is Plus (D_7 = 0 and S = 0)
JM	16-bit address	Jump when the result is Minus (D_7 = 1 and S = 1)
JPE	16-bit address	Jump if the Parity is Even (P = 1)
JPO	16-bit address	Jump if the Parity is Odd (P = 0)

Subroutine Instructions

If a function occurs repeatedly in the main program, it can be performed by a set of instructions, called a **subroutine**, written separately from the main program. The use of the subroutine avoids the repetition of the same instruction. The subroutine can be called by the main program when required. To implement subroutines, two instructions, namely, CALL and RET, are used. The CALL instruction appearing in the main program calls a subroutine, while the RET instruction appearing at the end of the subroutine commands the microprocessor to return to the main program. These instructions are explained below.

Instructions		
opcode	operand	Function
CALL	16-bit memory address of a subroutine	This is a 3-byte instruction transferring the program sequence to a subroutine address unconditionally. The memory location of the subroutine is specified by the second and the third bytes. The second byte denotes the line number and the third byte the page number. For example, the instruction CALL 2050H calls the subroutine located at 2050H.
RET		Return from the subroutine unconditionally. This is a 1-byte instruction.

Restart (RST) Instructions

These are 1-byte call instructions transferring the program execution to a particular location on page 00H. The eight RST instructions are listed below.

Mnemonic	Hex code	Function
RST 0	C7	Call 0000H
RST 1	CF	Call 0008H
RST 2	D7	Call 0010H
RST 3	DF	Call 0018H
RST 4	E7	Call 0020H
RST 5	EF	Call 0028H
RST 6	F7	Call 0030H
RST 7	FF	Call 0038H

Conditional Call and Return Instructions

Such instructions use the flag conditions. If the condition is satisfied, the conditional Call instruction transfers the program to the subroutine. Otherwise, the main program is continued. A conditional Return instruction returns the sequence to the main program when the condition is satisfied. Otherwise, the subroutine is continued. For a conditional Call instruction in the main program, the Return instruction in the subroutine may be conditional or unconditional.

Conditional Call Instructions

Mnemonic		Function
CC	16-bit address	Call subroutine located at the specified 16-bit address if Carry flag is set (CY = 1).
CNC	16-bit address	Call subroutine located at the specified 16-bit address if the Carry flag is reset (CY = 0).
CZ	16-bit address	Call subroutine located at the specified 16-bit address if the Zero flag is set (Z = 1).

Mnemonic	Function
CNZ 16-bit address	Call subroutine located at the specified 16-bit address if the Zero flag is reset (Z = 0).
CM 16-bit address	Call subroutine located at the specified 16-bit address if the number is negative, i.e., if Sign flag is set (S = 1).
CP 16-bit address	Call subroutine located at the specified 16-bit address if the number is positive, i.e., if Sign flag is reset (S = 0).
CPE 16-bit address	Call subroutine located at the specified 16-bit address if the parity is even, i.e., if the Parity flag is set (P = 1).
CPO 16-bit address	Call subroutine located at the specified 16-bit address if the parity is odd, i.e., if the Parity flag is reset (P = 0).

Conditional Return Instructions

Mnemonic	Function
RC	Return if the Carry flag is set (CY = 1)
RNC	Return if the Carry flag is reset (CY = 0)
RZ	Return if the Zero flag is set (Z = 1)
RNZ	Return if the Zero flag is reset (Z = 0)
RM	Return if the number is negative, i.e., if the Sign flag is set (S = 1).
RP	Return if the number is positive, i.e., if the Sign flag is reset (S = 0)
RPE	Return for even parity, i.e., if the Parity flag is set (P = 1).
RPO	Return for odd parity, i.e., if the Parity flag is reset (P = 0).

I/O and Machine Control Instructions

For data transfer between the microprocessor and the I/O device, IN and OUT instructions are used. The instruction IN transfers data inputs from the input device (e.g., a key board) into the accumulator. The instruction OUT transfers the contents of the accumulator to an output device (e.g., an LED display). These are 2-byte instructions where the second byte indicates the address or the port number of the I/O device. The instruction IN is explained below.

Memory address	Machine code	Mnemonics	Memory contents
2050	DB	IN64H; 2050	1101 1011 (= DBH)
2051	64	2051	0110 0100 (= 64H)

Here the memory locations 2050H and 2051H are chosen arbitrarily. In this instruction, the microprocessor is asked to first read the bytes stored in the memory locations 2050H and 2051H, and then read the switch positions at port 64H.

The instruction OUT is illustrated as follows:

Memory address	Machine code	Mnemonics	Memory contents
2060	D3	OUT 23H; 2060	1101 0011 (= D3H)
2061	23	2061	0010 0011 (= 23H)

Here also the memory locations 2060 and 2061 are chosen arbitrarily. If the output port with address 23H refers to an LED display, the execution of the instruction OUT displays the contents of the accumulator at the port.

HLT : It is a 1-byte machine control instruction having the Hex Code 76. It asks the microprocessor to stop processing and wait.

NOP : This is also a 1-byte machine control instruction having the Hex Code 00. It instructs the microprocessor not to perform any operation. This instruction can be used with advantage to replace a redundant instruction. When an error appears in a program due to a wrong instruction it is better to eliminate the instruction by substituting NOP rather than to reassemble the entire program.

Direct and Indirect Addressing

Addressing instructions are commands to the microprocessor to copy 8-bit data into a destination from a source. The destination is a register or an output port, and the source is a register, an input port, or an 8-bit number (from 00H to FFH).

Let the memory location 2000H contain the data byte D5H. One can copy this data byte from the memory location 2000H to the accumulator A as follows:

Machine code	Instruction
21	LXI H, 2000H
00	
20	
7E	MOV A, M

Here the register pair HL is first loaded with the data byte address 2000H. Then the data byte (D5H) from the memory address 2000H is shifted to the accumulator A. This is an example of **indirect addressing**.

The same operation can also be performed in the following manner:

Machine code	Instruction
3A	LDA 2000H
00	
20	

Here the data byte (D5H) is copied into the accumulator A from the memory location specified by the 16-bit address 2000H. This is an example of direct addressing mode; the instruction LDA gives the memory address 2000H directly as a part of its operand. Thus, in direct addressing mode, the address of the operand (data) is contained in the instruction itself. But, in indirect addressing mode, the address of the operand is not directly contained in the instruction; it is contained elsewhere, e.g., in the HL pair.

Microprocessor trainer kit

The trainer kit is used in the laboratory to execute an assembly language program on an Intel 8085 microprocessor. The manufacturer provides a manual containing detailed information regarding the installation, operation, and maintenance of the microprocessor system. The user must consult this manual, whenever necessary, particularly in executing a program. In a student laboratory, SDA-85, which is a single-board microcomputer kit, is commonly used. Some other kits are ESA 85, AEFC, and ILC V2.