Assembly Language Programming

Instruction format

An instruction is a command to the microprocessor to perform a given task on a particular data.

The 8085 have 74 basic instructions and 246 total instructions. The instruction set of 8085 is defined by the manufacturer Intel Corporation. Each instruction of 8085 has 1 byte opcode. With 8-bit binary code, we can generate 256 different binary codes. In this, 246 codes have been used for operational code.

The instruction format is a format by which we can give an instruction to the microprocessor. Each instruction format is of two parts: Opcode and Operand. In some instructions, the operand is implicit.

Opcode

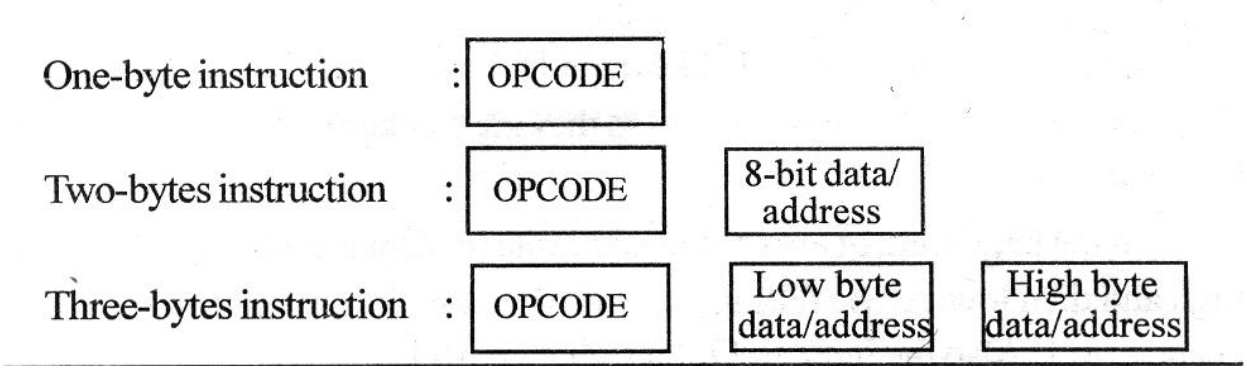
An opcode is short for 'Operation Code'. It is a single instruction that can be executed by the CPU. In machine language it is a binary or hexadecimal value such as 'B6' loaded into the instruction register. It is designed during the manufacturing of the chip. This means you cannot change the opcode unless you change the HARDWARE design of the processor.

Mnemonic

The term mnemonic goes hand-in-hand with opcode, and is simply a friendly term used to describe an opcode. It is usually used by assembly language programmers to remember the "OPERATIONS" a machine can do, like "ADD", "MOV" etc. This is assembler specific.

Operand

An operand is the second part of the instruction, which tells the computer where to find or store the data or instructions. It is the "argument" or "parameter" that directly follow a mnemonic. Generally speaking, depending upon the mnemonic used, there can be 0-3 operands following the mnemonic.



The size of 8085 instructions can be 1 byte, 2 bytes or 3 bytes.

One-byte instructions

A one-byte instruction includes an opcode and an operand in the same byte. Operand(s) are internal registers and are in the instruction in the codes. If there is no numeral present in the instruction then that instruction will be of one-byte, for example, "MOV C, A", "RAL", "ADD B", etc.

Two-byte instructions

In a two-byte instruction, the first byte specifies the operation code and the 2nd byte specifies the operand. Here, the operand may be a data or an address. For example, in MVI A, 35H and IN 29H, etc. In a two-byte instruction, the first byte will be the opcode and the second byte will be for the numeral present in the instruction.

Three-byte instructions

In a three-byte instruction, the first byte specifies the opcode, and the following two bytes specify the 16-bit operand. While storing the 3 bytes instruction in memory, the sequence of storage is, opcode first followed by low byte of address or data and then high byte of address or data. For example, in LXI H,3500H and STA 2500H, etc.

8085 INSTRUCTION SETS

a) Data Transfer Instructions
b) Arithmetic Instructions
c) Logical Instructions
d) Rotate Instructions
e) Branching Instructions
f) Control Instructions

a) Data Transfer instructions

Mnemonics	**Description**	**Example**
MOV Rd, Rs	• Copies the content of source register Rs into destination register Rd • Rs and Rd can be A, B, C, D, E, H, L	MOV A, B
MOV Rd, M	• Copies the content of memory location M into the destination register Rd • Rd can be A, B, C, D, E, H, L • The memory location M is specified by HL pair	MOV A, M
MOV M, Rs	• Copies the content of register Rs into memory location M • Rs can be A, B, C, D, E, H, L • The memory location M is specified by HL pair	MOV M, A
MVI Rd, 8-bit	• The 8-bit data is stored in the destination register Rd • Rd can be A, B, C, D, E, H, L	MVI A, 32H
MVI M, 8-bit	• The 8-it data is stored in memory location M • M is specified by HL pair	MVI M, 32H
LDA 16-bit (Load Accumulator Direct)	• Copies the content of memory location specified by 16-bit address into A	LDA 2015H (A← [2015])

STA 16-bit (Store Accumulator Direct)	• Copies the content of A into 16-bit memory address	STA 2015H (A→[2015])
LDAX Rp (Load Accumulator Indirect)	• Copies the content of memory location specified by register pair Rp into A • Rp can be B or D i.e. BC pair or DE pair	LDAX B
STAX Rp (Store Accumulator Indirect)	• Copies the content of A into 16-bit memory address specified by register pair Rp • Rp can be B or D i.e. BC pair or DE pair	STAX B
LXI Rp, 16-bit (Load Register Pair)	• Loads 16-bit data into register pair • Rp can be B, D or H i.e. BC pair, DE pair or HL pair	LXI H, 2015H L←15 H←20
IN 8-bit	• The data from i/p port specified by 8-bit address is transferred into A	IN 40H A←[40] 40H is address of input port
OUT 8-bit	• The data of A is transferred into output port specified by 8-bit address	OUT 10H A→[10] 10H is address of output port
PUSH rp	• Stores contents of register pair rp by pushing it into two locations above the top of the stack • rp = BC, DE, HL, or PSW	PUSH B
POP rp	• Pop out 2-Bytes from the top of the stack through rp i.e. register pair e.g. BC, DE, HL or AF	POP B
XCHG	• Exchange the content of HL pair with DE pair i.e. the content of H and D are exchanged whereas content of L and E are exchanged	XCHG H←→D L←→E

Example

MOV E, M . It is a 1-Byte instruction. Suppose E register content is DBH, H register content is 40H, and L register content is 50H. Let us say location 4050H has the data value AAH. When the 8085 executes this instruction, the contents of E register will change to AAH, as shown below.

	Before	**After**
(E)	DBH	AAH
(HL)	4050H	4050H
(4050H)	AAH	AAH

b) Arithmetic Instructions

Mnemonics	Description	Example
ADD R/M (add register/memory)	• The content of register /memory (R/M) is added to the A and result is stored in A • The memory M is specified by HL pair	ADD B A←A+B ADD M A←A+M
ADC R/M (add with carry)	• The content of register /memory (R/M) is added to the A along with carry flag CF and result is stored in A • The memory M is specified by HL pair	ADC B A←A+B+CF ADC M A←A+M+CF
ADI 8-bit (add immediate)	• The 8-bit data is added to A and result is stored in A	ADI 32H A←A+32
ACI 8-bit (add immediate with carry)	• The 8-bit data is added to A along with carry flag CF and result is stored in A	ACI 32H A←A+32+CF
SUB R/M (subtract register/memory)	• The content of register /memory (R/M) is subtracted from A and result is stored in A • The memory M is specified by HL pair	SUB B A←A-B SUB M A←A-M
SBB R/M (subtract with borrow)	• The content of register /memory (R/M) is subtracted from A along with borrow flag BF and result is stored in A • The memory M is specified by HL pair	SBB B A←A-B-BF SBB M A←A-M-BF
SUI 8-bit (subtract immediate)	• The 8-bit data is subtracted from A and result is stored in A	SUI 32H A←A-32
INR R/M (Increment Register/Memory)	• Increment the content of register/memory by 1 • Memory is specified by HL pair	INR B B←B+1 INR M M←M+1
DCR R/M (Decrement Register/Memory)	• Decrement the content of register/memory by 1 • Memory is specified by HL pair	DCR B B←B-1 DCR M M←M-1
INX Rp (Increment Register Pair)	• Increment the content of register pair Rp by 1	INX H HL←HL+1
DCX Rp (Decrement Register Pair)	• Decrement the content of register pair Rp by 1	DCX H HL←HL-1

c) Logical Instructions

Mnemonics	Description	Example
CMP R/M (Compare Register/Memory)	• Compares the content of register/memory with A • The result of comparison is: If A< R/M : Carry Flag CY=1	CMP B CMP M

	If A= R/M : Zero Flag Z=1 If A> R/M : Carry Flag CY=0	
CPI 8-bit (Compare Immediate)	• Compares 8-bit data with A • The result of comparison is: If A< 8-bit : Carry Flag CY=1 If A= 8-bit : Zero Flag Z=1 If A> 8-bit : Carry Flag CY=0	CPI 32H
ANA R/M (logical AND register/memory)	• The content of A are logically ANDed with the content of register/memory and result is stored in A • Memory M must be specified by HL pair	ANA B A←A.B ANA M A←A.M
ANI 8-bit (AND immediate)	• The content of A are logically ANDed with the 8-bit data and result is stored in A	ANI 32H A←A.32H
ORA R/M (logical OR register/memory)	• The content of A are logically ORed with the content of register/memory and result is stored in A • Memory M must be specified by HL pair	ORA B A←A or B ORA M A←A or M
ORI 8-bit (OR immediate)	• The content of A are logically ORed with the 8-bit data and result is stored in A	ORI 32H A←A or 32H
XRA R/M (logical XOR register/memory)	• The content of A are logically XORed with the content of register/memory and result is stored in A • Memory M must be specified by HL pair	XRA B A←A xor B XRA M A←A xor M
XRI 8-bit (XOR immediate)	• The content of A are logically XORed with the 8-bit data and result is stored in A	XRI 32H A←A xor 32H

d) Rotate Instructions

Mnemonics	**Description**	**Example**
RLC (Rotate Accumulator Left)	• Each bit of A is rotated left by one bit position. • Bit D7 is placed in the position of D0.	RLC
RRC (Rotate Accumulator Right)	• Each bit of A is rotated right by one bit position. • Bit D0 is placed in the position of D7.	RRC
RAL (Rotate Accumulator Left with Carry)	• Each bit of A is rotated left by one bit position along with carry flag CY • Bit D7 is placed in CY and CY in the position of D0.	RAL
RAR (Rotate Accumulator Right with Carry)	• Each bit of A is rotated right by one bit position along with carry flag CY. • Bit D7 is placed in CY and CY in the position of D0.	RLC

e) Branching Instructions

Mnemonics	Description	Example
JMP 16-bit address/label (Unconditional Jump)	The program sequence is transferred to the memory location specified by 16-bit address	JMP C000H
JC 16-bit address/label	Jump on Carry (CY=1)	
JNC 16-bit address/label	Jump No Carry (CY=0)	
JP 16-bit address/label	Jump on Positive (S=0)	
JM 16-bit address/label	Jump on Negative (S=1)	
JZ 16-bit address/label	Jump on Zero (Z=1)	
JNZ 16-bit address/label	Jump No Zero (Z=0)	
JPE 16-bit address/label	Jump on Parity Even (P=1)	
JPO 16-bit address/label	Jump on Parity Odd (P=0)	
CALL 16-bit address/label (Unconditional call)	The program sequence is transferred to the subroutine at memory location specified by the 16-bit address	CALL C000H
CC 16-bit address/label	Call if carry flag is 1	CC 2050
CNC 16-bit address/label	Call if carry flag is 0	CNC 2050
CZ 16-bit address/label	Calls if zero flag is 1	CZ 2050
CNZ 16-bit address/label	Calls if zero flag is 0	CNZ 2050
CPE 16-bit address/label	Calls if parity flag is 1	CPE 2050
CPO 16-bit address/label	Calls if parity flag is 0	CPO 2050
CM 16-bit address/label	Calls if sign flag is 1	CM 2050
CP 16-bit address/label	Calls if sign flag is 0	CP 2050
RET (Unconditional return)	The program sequence is transferred from the subroutine program to calling program	RET
RC	Return from the subroutine if carry flag is 1	RC
RNC	Return from the subroutine if carry flag is 0	RNC
RZ	Return from the subroutine if zero flag is 1	RZ
RNZ	Return from the subroutine if zero flag is 0	RNZ
RPE	Return from the subroutine if parity flag is 1	RPE
RPO	Return from the subroutine if parity flag is 0	RPO
RM	Returns from the subroutine if sign flag is 1	RM
RP	Returns from the subroutine if sign flag is 0	RP

f) Control Instructions

Mnemonics	Description	Example
NOP	No operation is performed	NOP
HLT	The CPU finishes executing the current instruction and stops any further execution	HLT