UNIT 4: INSTRUCTION SETS AND ADDRESSING MODES

UNIT STRUCTURE

4.1 Learning Objectives
4.2 Introduction
4.3 Instruction Format
4.4 Types of Instruction
4.5 Addressing Modes of 8085 Microprocessor
4.6 Execution of Instruction
4.7 Assembly Language Notation
4.8 Let Us Sum Up
4.9 Further Readings
4.10 Answer to Check Your Progress
4.11 Model Questions

4.1 LEARNING OBJECTIVES

After going through this unit you will be able to:

- describe the different instruction format of 8085
- describe the different types of instructions
- define addressing modes of 8085
- describe how instructions are executed
- define different assembly language notations

4.2 INTRODUCTION

In the previous unit, we have discussed the different types of data representation like binary number system, decimal number system, BCD etc. In this unit, we will discuss the different types of instruction format. We will also discuss the different types of addressing modes in this unit. In the next unit, we will explore the different concepts related to input output organization.

4.3 INSTRUCTION FORMAT

We know a computer performs a task when it is instructed to do so. A program is a sequence of instructions, where every instruction is recognized by the computer and executed by it to perform the specified job. The set of instructions to which a particular microprocessor responds, is fixed during the time of its design. The understanding of different instruction formats, types of instructions, and the mode of addressing will help us to visualize the internal working of a microprocessor and, in turn, the working of a computer. For 8085, the set of instructions are designed to perform various data movements, arithmetic, logical, branch control operations etc. There are 74 basic and 246 total instructions for INTEL 8085. The set has been defined by its manufacturer INTEL Corporation of USA. These instructions are having three different formats. They are:

a) One - byte instructions
b) Two- byte instructions
c) Three- byte instructions

These three formats are shown in figure 4.1. The first byte of all the instructions is always OPCODE which indicates the operation to be performed.

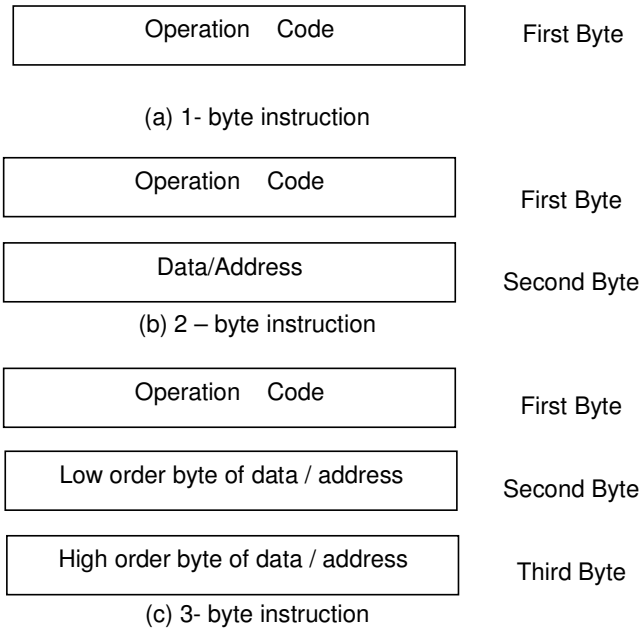


Figure 4.1: Different instruction format of 8085

a) One byte Instructions: This type of instructions contains both the operation as well as the data in a single byte called the OPCODE. INTEL 8085 has many such types of single-byte instruction. For example:

MOV A, B; Move the content of B to A; here, A and B are registers

ADD B; Add the content of B with A; HERE, A is the accumulator and B is a processor register

SUB C; Subtract the content of C from A

INR A; Increase the content of A by 1

DCR A; Decrease the content of A by 1

b) Two- byte Instructions: In these types of instructions, the first byte contains the OPCODE and the second byte contains an 8-bit data or address. For example:

ADI 05; Add immediate

SUI 07; Subtract immediate

CPI 12; Compare immediate

c) Three- byte Instructions: Here, the first byte contains the OPCODE and the second and the third bytes contain a 16- bit data or address. The second byte contains the list significant or lower order byte and third byte contains the most significant or high order byte of the 16-bit data or address. For example, if the 16-bit data is FA25H, then it is coded in the instruction as 25, FA in the second and third bytes respectively. Some more examples :

LXI H, 2500; Load H-L pair immediate by 2500 (H-L pair means high order and low order byte)

LDA 2825; Load accumulator from memory location 2825

4.4 TYPES OF INSTRUCTION

The INTEL 8085 has 74 different basic instructions and 246 opcodes in all. The instructions set of 8085 can be grouped under five different categories as shown below:

1. Data transfer Group
2. Arithmetic Group

3. Logical Group
4. Branch Group
5. Stack, 1/0 and Machine control Group

1. Data transfer Group:

The instructions in this group are used to move data from one register to another or from register to memory and vice-versa. Some of these instructions load data given immediately to the register. Data transfer instructions can be further subdivided on the basis of their mode of addressing – immediate, direct, register, and register indirect.

a) Immediate data transfer Instructions: One of these instructions is of the form:

MVI r, data (Move Immediate)

(r)← (byte 2)

The data contents of the second byte are immediately moved to the register r.

In the other form of immediate data transfer a 16-bit data or address is transferred to a register pair. The instruction format is –

LXI rp , 16-bit data (Load register pair Immediate)

(rh) ← (byte 3)

(rl) ← (byte 2)

b) Direct Data Transfer Instructions: Here the data is transferred to and from memory. The address of the memory location is given in the instruction. For example:

LDA addr (Load Accumulator Direct)

(A) ← ((byte 3)(byte 2))

The meaning of this instruction is load the accumulator by the data stored in memory location specified by byte 2 and byte 3.

(c) Register Data Transfer Instruction: Many 8085 instructions fall under this group. These instructions transfer an 8-bit data between registers. General format is given below.

MOV r1, r2 (Move register)

(r1) ← (r2)

The content of register r2 is copied to register r1.

(d) Register Indirect Data Transfer Instructions: These instructions move data between register and memory. The address of memory location must be loaded in H-L register pair. The format is given below.

MOV r, m (Move data from memory to register)

(r) ← ((H) (L))

and

MOV m, r (Move data from register to memory)

((H) (L)) ← (r)

2) Arithmetic Group:

This group includes various instructions for 8-bit addition, subtraction, 16-bit addition, increment, decrement of the content of register and register pair.

Some instructions of these groups require two operands. In such cases, one of the operand must be in accumulator. After the arithmetic operation, the result is stored in the accumulator. Format for some instructions are –

i) ADD r (Add register)

(A) ← (A) + (r)

ii) ADD M (Add memory)

(A) ← (A) + ((H)(L))

iii) ADI 8-bit data (Add immediate)

(A) ← (A) + (byte2)

iv) INR r (increment register)

(r)← (r) + 1

3) Logical Group:

These instructions perform various logical operations on data stored in register, memory etc. These instructions include comparing, ORing, ANDing, EXORing, complementing and rotating of data. Here one operand is stored in the accumulator and the result is also stored in it.

Generalized format of some of these instructions are –

i) ORA r (Perform OR operation between register r and the accumulator)

(A) ← (A) (r)

The content of register r is ORed with the contents of the accumulator and the result is placed in the accumulator.

Similarly,

ii) ANA r (AND register)

(A) ← (A) (r)

iii) ANI data (AND immediate)

(A) ← (A) $\wedge$ (byte2)

iv) XRA r (Exclusive OR register)

(A) ← (A) (r)

v) CMP r (Compare register)

(A)- (r)

The meaning of example v) is that the contents of register r is subtracted from the contents of accumulator and the accumulator remains unchanged. The condition flags are set as a result. The Z flag is set to 1 if (A) =(r). The CY flag is set to 1 if (A) < (r). Here, Z flag indicates zero, i.e. Z=1 when (A) – (r) = 0, otherwise Z=0, and CY means carry.

vi) RLC (Rotate left)

(An +1)← (An)

(CY), (Ao) ← (A7)

The content of the accumulator is shifted left by one bit. The low order bit and the carry are both set to the value shifted out from the A7 bit. Similarly,

vii) RRC (Rotate right)

(An)←(An+1)

(CY),(A7)←(Ao)

viii) RAL (Rotate left through carry)

(An+1)←(An)

(Ao)←(CY)

(CY)←(A7)

ix) RAR (Rotate right through carry)

(An)←(An+1)

(A7)←(CY)

(CY)←(Ao)

(A)←(Ā)

x) CMA (Complement accumulator)

xi) CMC (Complement carry)

(CY)←($\overline{CY}$)

xii) STC (Set carry)

(CY)← 1

Branch Group:

Normally a computer executes a program from top to bottom sequentially. But sometimes the sequential flow of execution is required to be broken and the execution needs to alter its path by jumping from one place to another place in the program. This is known as branching. In 8085 instruction set, there are two basic forms of branching instructions: JUMP and CALL instructions.

The JUMP instruction causes the program execution to jump from one location to any other location in the program from where computer continues to execute thereafter.

The CALL instruction transfers the control of execution from the main program to a sub-program temporarily. Execution of program returns back to the main program when the RETURN instruction is encounted at the last of the sub-program.

There are *two* types of JUMP, CALL and RETURN instructions—*conditional* and *unconditional*. The generalized forms of some of these instructions are as follows.

i) JUMP addr

ii) CALL addr

iii) RETURN addr

These are unconditional jump, call and return instructions.

On the other hand, JZ, JNZ, JC, JNC, CZ, CNZ, RZ, RNZ etc are some of the conditional jump, call and return instructions, which jump, call and return when the condition is satisfied.

5. Stack, I/O and Machine Control Group:

This group contains miscellaneous instructions. These are I/O, stack manipulation, interrupt control instructions.

i) IN port address

(A) ← (Data)

ii) OUT port address

(Data) ← (A)

These two are I/O instructions through I/O port.

iii) PUSH r (Push register in to the top of stack)

iv) POP r (Pop register from the top of stack)

v) PUSH PSW (Push Program Status Word)

vi) POP PSW (Pop Program Status Word)

These four are stack manipulation instructions.

vii) EI (Enable Interrupt)

viii) DI (Disable Interrupt)

ix) SIM (Set Interrupt)