Chapter Four Instructions Set

Instructions set

Instructions set 8086 has 117 instructions, these instructions divided into 6 groups:

1. Data transfer instructions
2. Arithmetic instructions
3. Logic instructions
4. Shift instructions
5. Rotate instructions
6. Advance instructions

Data Transfer Instructions

Data Transfer Instructions The microprocessor has a group of data transfer instructions that are provided to move data either between its internal registers or between an internal register and a storage location in memory. Some of these instructions are:

- **MOV:** use to transfer a byte or a word of data from a source operand to a destination operand. These operands can be internal registers and storage locations in memory. Notice that the MOV instruction cannot transfer data directly between a source and a destination that both reside in external memory. For instance, flag bits within the microprocessors are not modified by execution of a MOV instruction.

EXAMPLES

1. MOV DX, CS where DX = 0100H

DX = CS = 0100H

2.MOV SUM, AX DS = 0200H SUM = 1212H

PA = 02000H +1212H = 03212H

AL Memory location 03212H

AH Memory location 03213H

3.If DS contain 1234H what is the effect of executing the instruction MOV CX,[0ABCDH]

CL loaded with the content of Memory location

1234H + ABCDH =1CF0DH And CH is loaded with the content of Memory location 1234H + ABCDH +1 = 1CF0EH

- **XCHG:** in MOV instruction the original contents of the source location are preserved and the original contents of the destination are destroyed. But **XCHG** instruction can be used to swap data between two general purpose register or between a general purpose register and storage location in memory.

EXAMPLES:

1.XCHG AX, DX (AX) (DX)

2.XCHG SUM, BX

(DS (0) + SUM) BX

DS = 02000 H + 1234 H = 03234 H

(3234) (BL)

(3235) (BH)

2.Arithmetic Instructions

Arithmetic instructions includes instructions for the addition, subtractions can be performed on numbers expressed in a variety of numeric data formats. The status that results from the execution of an arithmetic instruction is recoded in the flags of the microprocessor. The flags that are affected by arithmetic instructions are CF, AF, SF, ZF, and PF.

- **Addition:** ADD, ADC, and INC

ADD AX,BX

AX= AX+BX

EXAMPLE:

AX= 1100H, BX=0ABCH ADD AX, BX

1100H+ 0ABCH = 1BBCH = AX

- **ADC AX, BX AX**

= AX+BX+CF

- **INC AH AH**

= AH +1

EXAMPLE:

The original contents of AX, BL, memory location SUM, and CF are AX=1234H, BL= ABH, Sum=00CDH and CF=0 respectively, describe the result of execution the following sequence of instruction:

ADD AX, SUM A

DC BL, 05H

INC SUM

1. AX= 1234H + 00CDH = 4301H CF=0

2. BL= ABH +05H +0=B0H CF=0

3. SUM=00CDH + 1=00CEH CF=0

Instructions	AX	BL	SUM	CF
Initial state	1234H	ABH	00CDH	0
ADD AX,	4301H	ABH	00CDH	0
SUM ADC BL,	4301H	B0H	00CDH	0
05H INC SUM	4301H	B0H	00CEH	0

➢ **<u>Subtraction:</u>** SUB, SBB, DEC, and NEG

- SUB AX, BX

AX = AX – BX

- SBB AX, BX

AX = AX - BX – CF

<u>EXAMPLE:</u>

BX = 1234H, CX = 0123H, CF = 0

SBB BX, CX

BX = 1234H - 0123H - 0 = 1111H

- DEC subtract 1 from its operand
- NEG BX (2's complement)

00H – BX 0000 + 2's complement of BX

EXAMPLE:

BX = 3A H NEG BX 0011 1010

0000 H +FFC6H 1100 0101 +

1

1100 0110

C 6

➢ **Multiplication and Division MUL, DIV**

- **MUL CL**

(AX)= AL* CL

- **MUL CX**

(DX, AX) = AX * CX

- **DIV CL**

(AH), (AL) = AX/CL 51

And AL the quotient

Where AH is the reminder

- **DIV CX**

  DX, AX= (DX,AX)/CX

  AX contain the Quotient

  DX contain the reminder

<u>**EXAMPLE:**</u>

MUL CL where AL=-1 CL= -2

AX = FF H * FE H = FD02 H

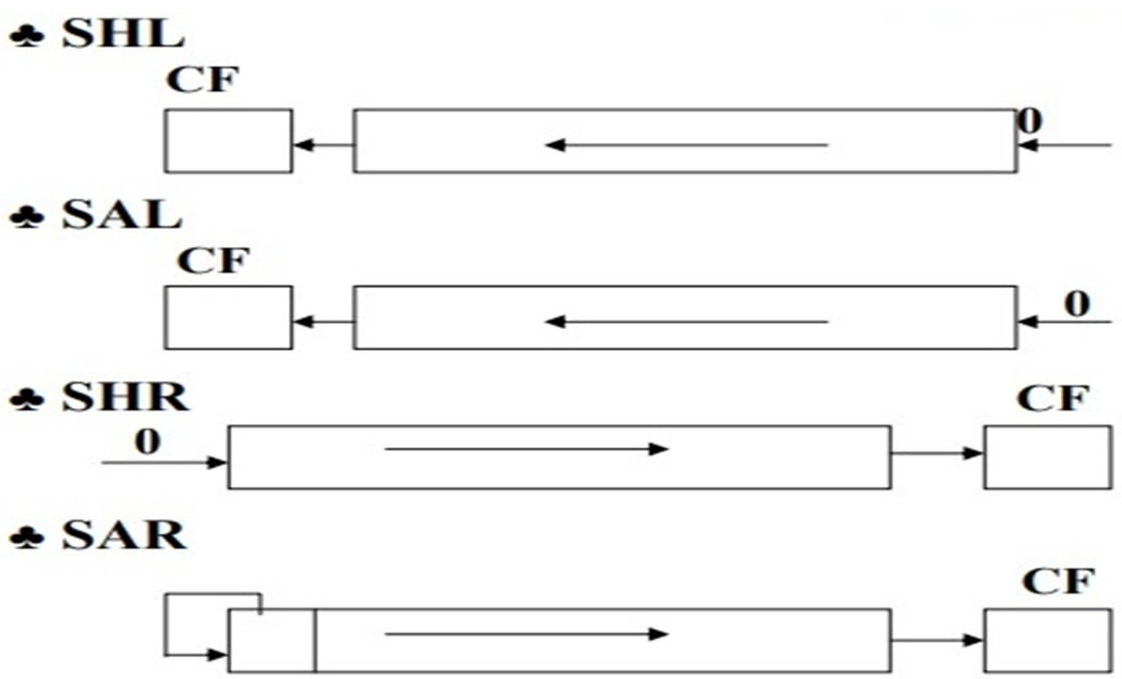
3.Logical Instructions (AND, OR, XOR, NOT)

Instructions	AL
MOV AL, 0101 0101B	0101 0101 B
AND AL,0001 1111B	0001 0101 B
OR AL,1100 0000B	1101 0101 B
XOR AL,0000 1111B	1101 1010 B
NOT AL	0010 0101 B

4. Shift Instructions

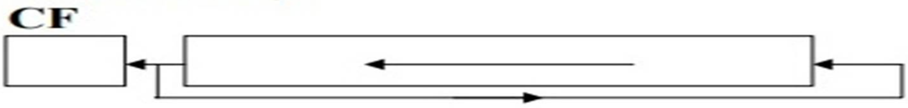
The four types of shift instructions can perform two basic types of shift operations. They are the logical shift and arithmetic shift. Each of these operations can be performed to the right or to the left.

Instructions	Meaning	format	Operation	Flags affected
SAL/SHL	Shift arithmetic left/shift logical left	SAL/SHL D, Count	Shift the D left by the number of bit positions equal to count and fill the vacated bits positions on the right with zeros	OF, CF
SHR	Shift logical right	SHR D, Count	Shift the D right by the number of bit position equal to count and fill the vacated bit positions on the left with zeros	OF, CF
SAR	Shift arithmetic right	SAR D, Count	Shift the D right by the number of bit positions equal to count and fill the vacated bit positions on the left with the original most significant bit	OF,SF, ZF, AF, PF, CF

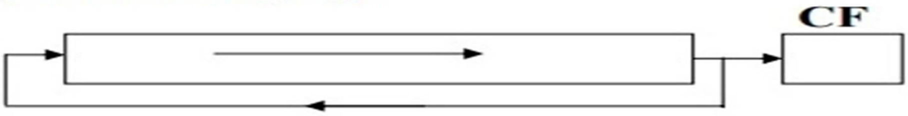


5.Rotate Instructions

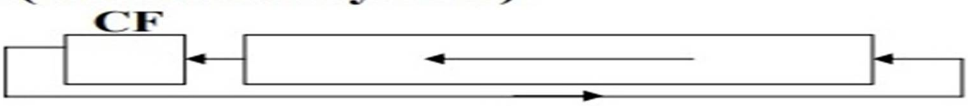
♣ **ROL (Rotate Left)**



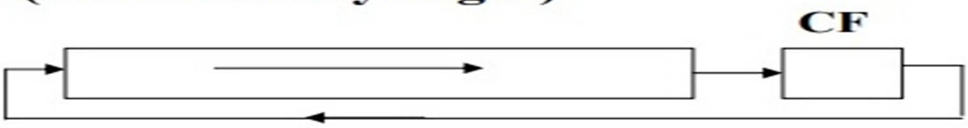
♣ **ROR (Rotate Right)**



♣ **RCL (Rotate Carry Left)**



♣ **RCR (Rotate Carry Right)**



6. Advance instruction (Program and Control Instruction)

In this section many of instructions that can be executed by the 8086 microprocessor are described, furthermore, these instructions use to write simple programs. The following topics are discussed in this section:

1. Flag control instructions
2. Compare instruction
3. Jump instructions
4. Push and POP Instruction
5. String instruction

1.Flag Control Instruction

The 8086 microprocessor has a set of flags which either monitor the status of executing instruction or control options available in its operation. The instruction set includes a group of instructions which when execute directly affect the setting of the flags. The instructions are:

LAHF: load AH from flags

SAHF: store AH into flags

CLC: clear carry, CF=0

STC: set carry, CF=1

CMC: complement carry, CF= CF

CLI: clear interrupt, IF=0

STI: set interrupt, IF=1

EXAMPLE:-

Write an instruction to save the current content of the flags in memory location MEM1 and then reload the flags with the contents of memory location MEM2

Solution:

LAHF

MOV MEM1, AH

MOV AH, MEM2

SAHF

2.Compare Instruction

There is an instruction included instruction set which can be used to compare two 8-bit number or 16-bit numbers. It is the compare (CMP) instruction. The operands can reside in a storage location in memory, a register within the MPU.

Instruction	Meaning	Format	Operation	Flag affected
CMP	Compare	CMP D,S	D-S	CF,AF,OF,PF,SF

Destination	Source
Register	Register
Register	Memory
Memory	Register
Register	Immediate
Memory	Immediate
Accumulator	Immediate

The process of comparison performed by the CMP instruction is basically a subtraction operation. The source operand is subtracted from the destination operand. However the result of this subtraction is not saved. Instead, based on the result the appropriate flags are set or reset.

EXAMPLE: lets the destination operand equals 100110012 and that the source operand equals 000110112. Subtraction the source from the destination, we get

١٠٠١١٠٠١

\- ٠٠٠١١٠١١

Replacing the destination operand with its 2's complement and adding

10011001

\+ 11100101

011111102

1. No carry is generated from bit 3 to bit 4, therefore, the auxiliary carry flag AF is at logic 0.
2. There is a carry out from bit 7. Thus carry flag CF is set.
3. Even through a carry out of bit 7 is generated; there is no carry from bit 6 to bit 7. This is an overflow condition and the OF flag is set.

4. There are an even number of 1s, therefore, this makes parity flag PF equal to 1.
5. Bit 7 is zero and therefore sign flag SF is at logic 0.
6. The result that is produced is nonzero, which makes zero flag ZF logic 0.

3.JUMP Instruction

The purpose of a jump instruction is to alter the execution path of instructions in the program. The code segment register and instruction pointer keep track of the next instruction to be executed. Thus a jump instruction involves altering the contents of these registers. In this way, execution continues at an address other than that of the next sequential instruction. That is, a jump occurs to another part of the program.

There two type of jump instructions:

a. Unconditional jump.
b. Conditional jump.

In an **unconditional jump**, no status requirements are imposed for the jump to occur. That is, as the instruction is executed, the jump always takes place to change the execution sequence. See Figure 16.

Instruction	Meaning	Format	Operation	Flag affected
JMP	Unconditional jump	JMP operand	Jump is to the address specified by operand	None

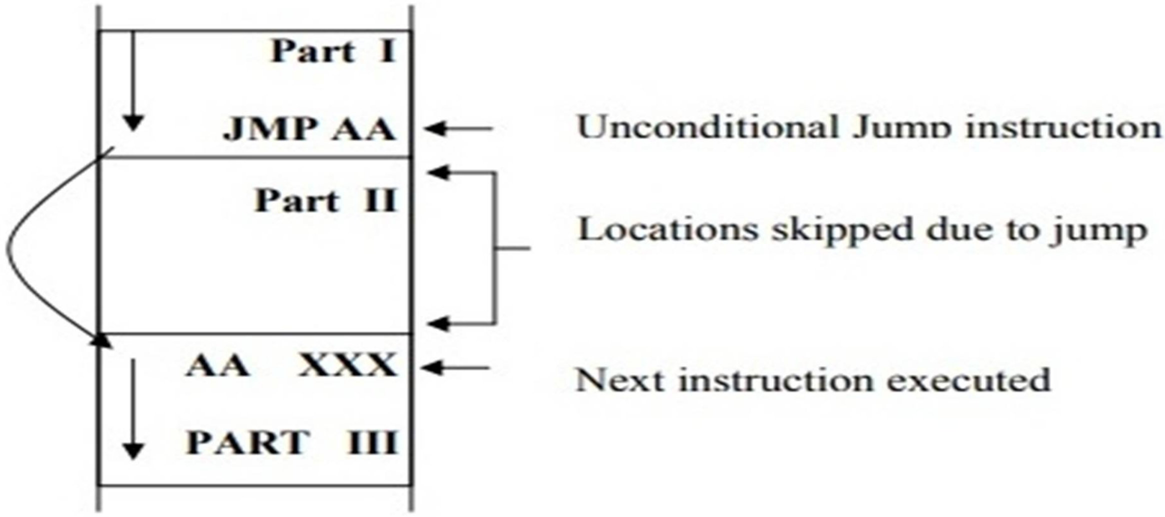


Figure 16: Unconditional jump program sequence.

On the other hand, for a **conditional jump** instruction, status conditions that exist at the moment the jump instruction is executed decide whether or not the jump will occur. If this condition or conditions are met, the jump takes place, otherwise execution continues with the next sequential instruction of the program. The conditions that can be referenced by a conditional jump instruction are status flags such as carry (CF), parity (PF), and overflow (OF). See Figure 17

Instruction	Meaning	Format	Operation	Flags affected
JCC	Conditional jump	Jcc operand	If the specific condition cc is true, the jump to the address specified by the operand is initiated, otherwise the next instruction is executed	None

The following table lists some of the conditional jump instructions

Instruction	Meaning
JAE/JNB	Jump if above or equal jump if not below
JB/JNAE	Jump if below/jump if not above or equal
JC	Jump if carry
JCXZ	Jump if CX is zero
JE/JZ	Jump if equal/jump if zero
JNC	Jump if not carry
JNE/JNZ	Jump if not equal/ jump if not zero
JNO	Jump if not overflow
JNP/JPO	Jump if parity/jump if parity odd
JNS	Jump if not sign
JO	Jump if overflow
JP/JPE	Jump if parity/jump if parity Even
JS	Jump if sign

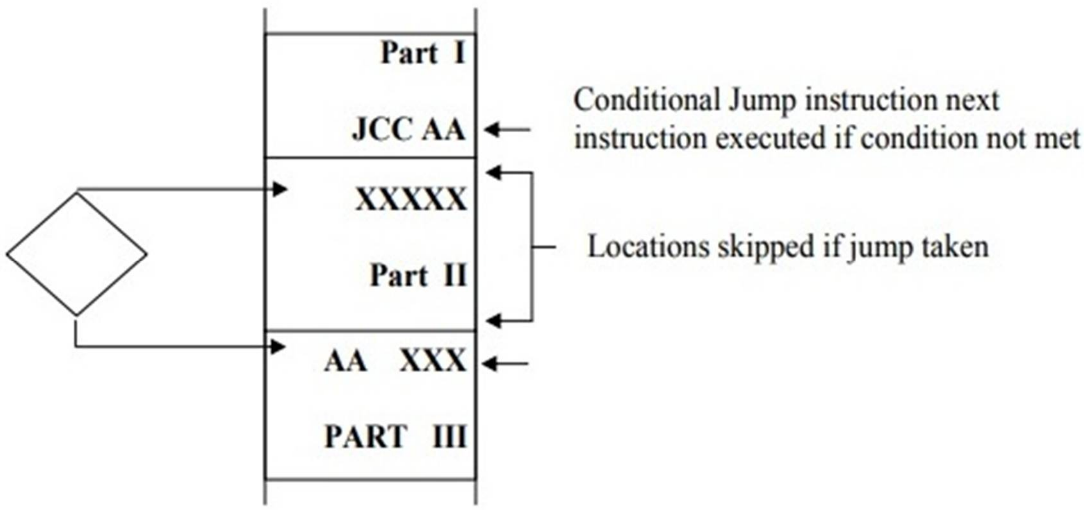


Figure 17: Conditional jump program sequence.

EXAMPLE: write a program to move a block of N bytes of data starting at offset address BLK1ADDR to another block starting at offset address BLK2ADDR. Assume that both blocks are in the same data segment, whose starting point is defined by the data segment address DATASEGMADDR.

```
MOV AX, DATASEGADDR
MOV DS, AX
MOV SI, BLK1ADDR
MOV DI, BLK2ADDR
MOV CX, N
NXTPT: MOV AH, [SI]
MOV [DI], AH
INC SI
INC DI
DEC CX
JNZ NXTPT
HLT
```

4.Push and POP Instruction

It is necessary to save the contents of certain registers or some other main program parameters. These values are saved by pushing them onto the stack. Typically, these data correspond to registers and memory locations that are used by the subroutine.

The instruction that is used to save parameters on the stack is the push (PUSH) instruction and that used to retrieve them back is the pop (POP) instruction. Notice a general-purpose register, a segment register (excluding CS), or a storage location in memory as their operand.

- Execution of a PUSH instruction causes the data corresponding to the operand to be pushed onto the top of the stack. For instance, if the instruction is **PUSH AX** the result is as follows:

$((SP)-1) \leftarrow (AH)$

$((SP)-2) \leftarrow (AL)$

$(SP) \leftarrow (SP)-2$

This shows that the two bytes of the AX are saved in the stack part of memory and the stack pointer is decrement by 2 such that it points to the new top of the stack.

- On the other hand, if the instruction is <u>POP AX</u> Its execution results in

(AL) ((SP))

(AH) ((SP) + 1)

(SP) (SP)+2

The saved contents of AX are restored back into the register.

- We also can save the contents of the flag register and if saved we will later have to restore them. These operations can be accomplished with the push flags **<u>(PUSHF)</u>** and pop flags **<u>(POPF)</u>** instructions, respectively. Notice the **<u>PUSHF</u>** save the contents of the flag register on the top of the stack. On the other hand, **<u>POPF</u>** returns the flags from the top of the stack to the flag register.

Instruction	Meaning	Operation	Flags affected
PUSHF	Push flags onto stack	$((SP)) \leftarrow (flag)$	None
POPF	Pop flags from stack	$(flag) \leftarrow ((SP))$	OF, DF, IF, TF, SF, ZF, AF, PF, CF

5.String Instructions

The microprocessor is equipped with special instructions to handle string operations. By "string" we mean a series of data words or bytes that reside in consecutive memory locations. There are five basic string instructions in the instruction set of the 8086, these instruction are:

a. Move byte or work string (MOVS, MOVSB, and MOVSW).
b. Compare string (CMPS).
c. Scan string (SCAS).
d. Load string (LODS)
e. Store string (STOS).

They are called the basic string instructions because each defines and operations for one element of a string.