2- Arithmetic instruction

The instruction set of the 8086 microprocessor contains a variety of arithmetic instructions. They include instructions for the addition, subtraction, multiplication, and division operations. These operations can be performed on numbers expressed in a variety of numeric data formats.

These formats include unsigned or signed binary bytes or words, unpacked or packed decimal bytes, or ASCII numbers. Remember that by packed decimal we mean that two BCD digits are packed into a byte-size register or a memory location.

Unpacked decimal numbers are stored one BCD digit per byte. The BCD numbers are unsigned decimal numbers. ASCII numbers are expressed in ASCII code and stored one number per byte.

The status that results from the execution of an arithmetic instruction is recorded in the flags of the microprocessor. The flags that are affected by the arithmetic instructions are carry flag (CF), auxiliary flag (AF), sign flag (SF), zero flag (ZF), parity flag (PF), and overflow flag (OF).

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

Figure -1-(a) shows the form of each of the addition instructions

Fig. 1-(b) shows. the allowed operand variations for all but the INC instruction; and Fig. 1-(c) shows the allowed operands for the INC instruction.

ADD instruction:

The ADD instruction is use to add the contents of a source operand to the contents of the destination operand. The result is put into the location of the destination operand.

In general, the result of executing ADD is expressed as:

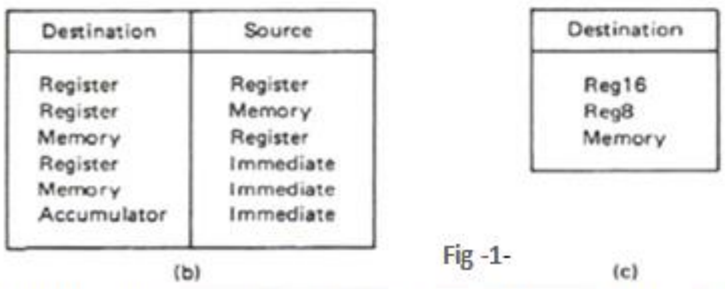
(D) + (S) → (D)

- The 8086 microprocessor can perform addition operation between any two registers except segment register (**CS, DS, ES,** and **SS**) and instruction pointer (**IP**).
- Addition must occur between similar sizes
- Addition can occur between register and memory
- The destination on all addition instructions cannot be immediate number
- No memory to memory can be added.

Mnemonic	**Meaning**	**Format**	**Operation**	**Flags affected**
ADD	Addition	ADD D,S	(S) + (D) → (D) carry → (CF)	ALL
ADC	Add with carry	ADC D,S	(S) + (D) + (CF) → (D) carry → (CF)	ALL
INC	Increment by one	INC D	(D) + 1 → (D)	ALL but CY
DAA	Decimal adjust for addition	DAA	Adjust AL for decimal Packed BCD	ALL

(a)

Fig-1-



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

(b)

Destination
Reg16
Reg8
Memory

(c)

Fig -1-

(a) Addition instructions (b) Allowed operands for ADD and ADC.
(c) Allowed operands for INC instruction

Ex: For the figure below,

- What is the result of executing the following instruction?
- What is the addressing mode for this instruction?

ADD AX , [DI + BX +2H]

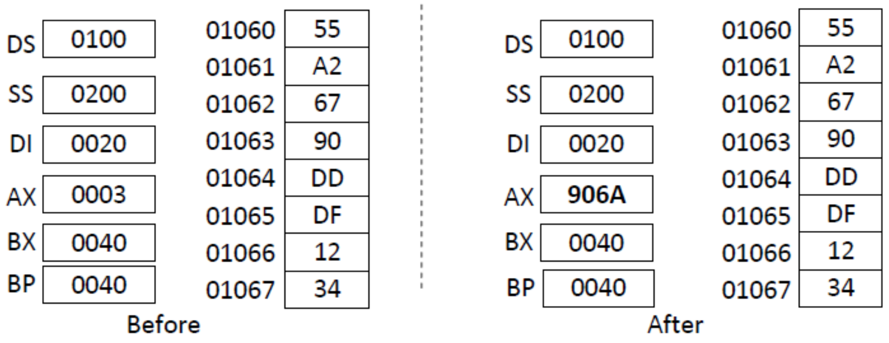
Solution:

- EA= [DI+ BX +2H] =[0020 + 0040 + 02H]= 0062H

  PA = (DS × 10H) + EA = 1000H +0062H= 1062H

  Memory word stored at location 1062H is 9067 H

  AX = AX + 9067H = 906A H



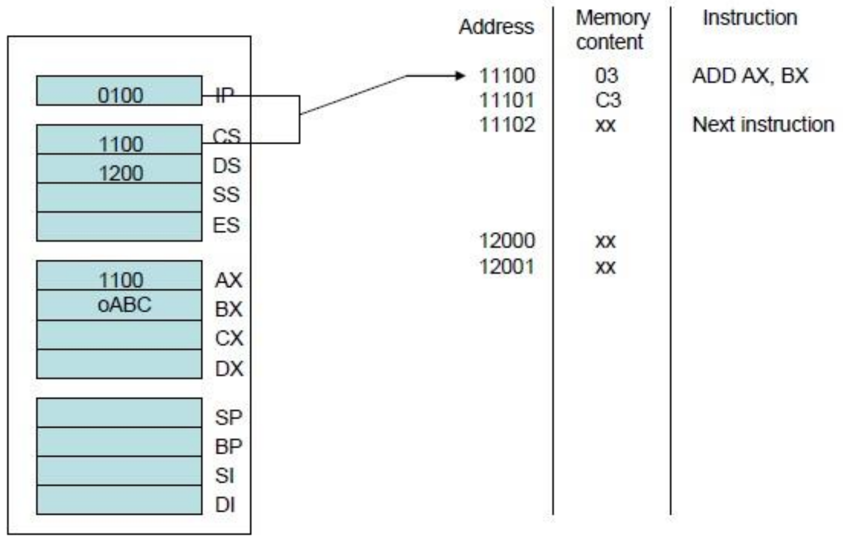
- The addressing mode for this instruction is **Based Indexed** mode.

Ex: Assume that AX and BX registers contain 1100H and 0ABCH, respectively. What is the result of executing the instruction

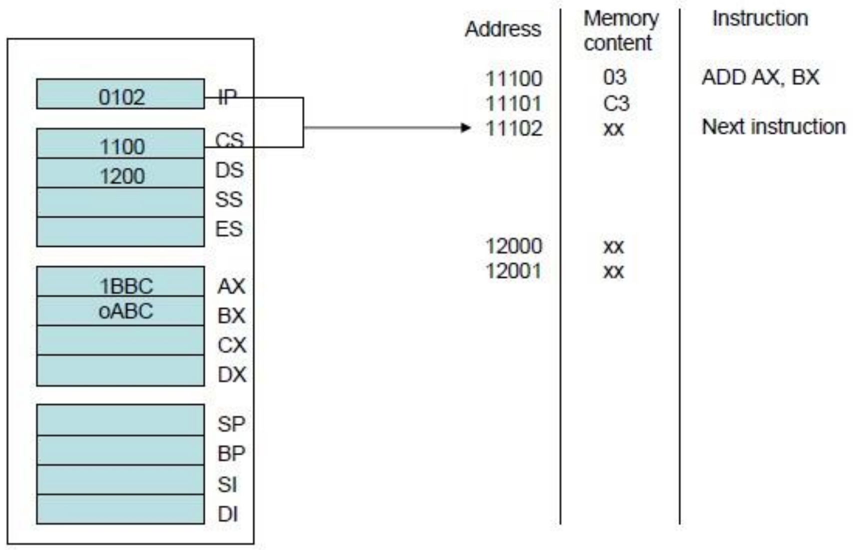
ADD AX, BX

The content of the source (BX) will be added to the content of the destination (AX) to give 0ABC + 1100 = 1BBC in the destination AX. The process of executing this instruction is shown in the following figures.

ADD instruction before execution



After execution of ADD instruction



ADC INSTRUCTION:

The instruction add with carry (ADC) works similarly to ADD. But in this case. The content of the carry flag is also added
that is:

(S)+(D)+(CF)→ (D)

The ADC instruction is used for multiword add operation.

INC instruction:

As shown in Fig.1-(c), its operands can be the contents of a l6-bit internal register, an 8-bit internal register, or a storage location in memory. Execution of the INC instruction adds 1 to the specified operand. An example of an instruction that increments the high byte of AX is

INC AH

This instruction is typically used to increment the values of a count or address.

Ex:- The original contents of AX=1234H, BL=ABH, word content of memory location with PA of DS:1234H=[DS:1234H] → 00CDH
and carry flag CF=NC=0
Describe the results of executing the following sequence of instructions:

```
ADD  AX, [SUM]
ADC   BL, 05H
INC   [SUM]
```

SOLUTION:

Executing the first instruction adds the word in the accumulator and the word in the memory location pointed to by address SUM. The result is placed in the accumulator. That is:

(AX)+([DS:1234H]) = 1234H + 00CDH = 1301H → (AX)

The carry flag remains reset.

The second instruction adds to the lower byte of the base register (BL) the immediate operand 5H and the carry flag, which is 0H. This gives

(BL) + imm8 + (CF) = ABH + 5H + 0 = B0H → (BL)

Since no carry is generated, CF stays reset.

The last instruction increments the contents of memory location SUM by one.

That is,

(SUM) + 1H = OOCDH+ 1H = OOCEH → (SUM)

These results are summarized in Figure below:

instruction	**(AX)**	**(BL)**	**(SUM)**	**(CF)**
Initial state	**1234**	**AB**	**00CD**	**0**
ADD AX,[SUM]	**1301**	**AB**	**00CD**	**0**
ADC BL,05H	**1301**	**B0**	**00CD**	**0**
INC [SUM]	**1301**	**B0**	**00CE**	**0**

Ex:- Perform the following 32-bit binary add operation

(DX,CX) ← (DX,CX) + (BX,AX)

For the following data in the registers

(DX,CX) = FEDCBA98H

(BX,AX) = 01234567H

The least 16 significant bits of the 32 bit numbers are added

ADD CX, AX

To add the most significant 16 bits we must account for the possibility Of a carry out from the addition of the lower 16 bits. Therefore, ADC Instruction must be used

ADC DX, BX

Ex:- For the figure below, what is the result of executing the following instructions?

INC WORD [0040]

INC BYTE [0042]

Solution:

- WORD[0040]→ 03FF H

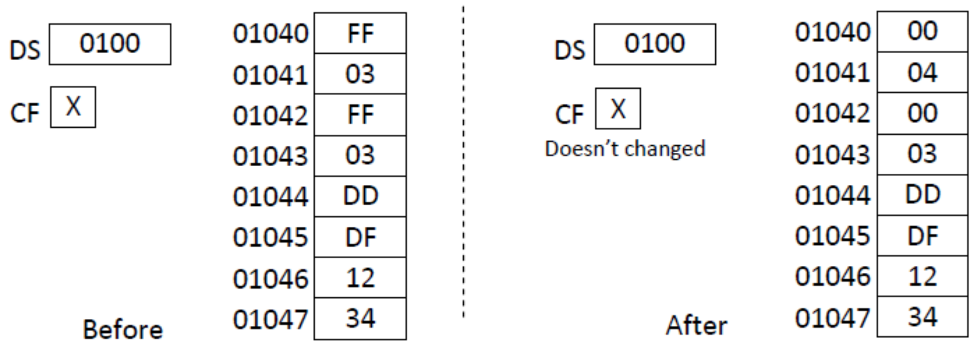
Execute INC instruction

WORD [0040] → 03FFH+1H= 0400H

- BYTE[0042] → FFH

Execute INC instruction

BYTE [0042] → FFH+1H= 00H



Subtraction of Binary numbers:

The way of performing binary subtraction is to use the 2's complement method. Using this method, the difference of two binary numbers is found by an addition process instead of directly through subtraction. The process requires that the value of the minuend be replaced by its 2's complement and then this value is added to the subtrahend to produce the difference. That is, the subtraction

(Subtrahend) — (Minuend) = Difference

is performed as

(Subtrahend) + (2's complement of Minuend) = Difference

Let us first review how to form the 2's complement of a binary number. To form the 2's complement of a number, first change all 1s in the number to 0s and all 0s to ls; then 1 is added to the least significant bit.

Ex: Subtract 32 H (0011 0010) from 45 H (0100 0101)

```
        32 H                 =   0011 0010
2's complement of 32 H  =   1100 1110
                         +
        45 H                 =   0100 0101
                      ___________________________
                   | 1 |    0001 0011 = 13 H
                    CF
```

The carry from the most significant bit is ignored

Ex: subtract 45 H (0100 0101) from 32 H (0011 0010)

```
        45 H               =   0100 0101
  2's complement of 45 H  =   1011 1011
                       +
        32 H               =   0011 0010
                           ________________________

                               1110 1101 = ED H
```

The result is negative and it is expressed in 2's complement, this can be verified by taking the 2's complement of result, the 2's complement of the result should be 13 H

Result ED H = 1110 1101

2's complement of ED H = 0001 0011 = 13 H

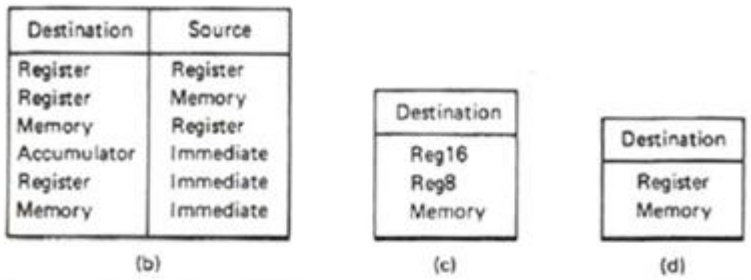
Subtraction instruction: SUB, SBB, DEC, DAS, NEG

Figure -2-(a) shows the form of each of the subtraction instructions, Fig. 2-(b) shows. the allowed operand variations for SUB & SBB,. Fig. 2-(c) shows the allowed operands for the DEC instruction and Fig. 2-(d) shows the allowed operands for the NEG instruction.

Mnemonic	Meaning	Format	Operation	Flags affected
SUB	Subtract	SUB D,S	(D) - (S) → (D) Borrow → (CF)	All
SBB	Subtract with borrow	SBB D,S	(D) - (S) - (CF) → (D)	All
DEC	Decrement by one	DEC D	(D) - 1 → (D)	All but CF
NEG	Negate	NEG D		All
DAS	Decimal adjust for subtraction	DAS	Convert the result in AL to packed decimal format	All

(a)

Fig-2-



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

(b)

Destination
Reg16
Reg8
Memory

(c)

Destination
Register
Memory

(d)

(a) Subtraction instructions (b) Allowed operands for SUB and SBB.
(c) Allowed operands for DEC instruction (d) Allowed operands for NEG instruction

SUB instruction:

The subtract (SUB) instruction is used to subtract the value of a source operand from a destination operand. The result of this operation in general is given as

(D) - (S) → (D)

SBB instruction:

The subtract with borrow is similar to SUB but also subtract the carry flag From the destination.

(D) - (S) – (CF) → (D)

EX: Assume the contents of registers BX and CX are 1234H and 0123H, respectively, and the carry flag is 0, what is the result of executing the following instruction

SBB BX, CX

The instruction implements the operation

(BX) –(CX) –(CF) → (BX)

Therefore, (BX) = 1234H –0123H –0 = 1111H

The carry flag remains cleared since no borrow is needed.

DEC instruction:

The decrement instruction is used to subtract 1 from its operand.

It does not affect the carry flag.

Ex:

DEC AL	Subtracts 1 from the contents of AL.
DEC C BX	Subtracts 1 from the contents of BX.

Note : The carry flag CF is not affected. If the contents of 8-bit register are 00H and 16-bit register are 0000H, after DEC instruction contents of registers wilt be FFH and FFFFH respectively without affecting carry flag.

DEC BYTE [BX]	Decrement byte at offset of BX in DS.
DEC WORD [BX]	Decrement word at offset of BX in DS.

NEG instruction:

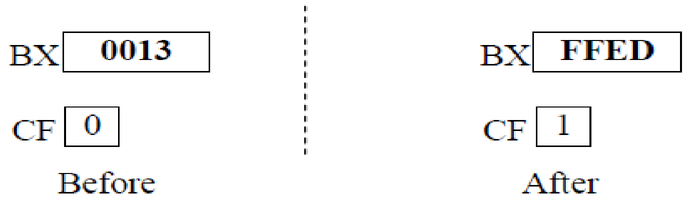
The negate (NEG) instruction This instruction replaces the number in a destination with the 2's complement of that number. The destination can be a register or a memory location. This instruction can be implemented by inverting each bit and adding 1 to it.

The carry flag will be become 1 as a result of this instruction.

Ex:- what is the result of executing the following instruction sequence?

NEG BX

Solution :



Ex: - Assume (BX)=003AH, what is the result of executing the instruction

NEG BX

(BX)=0000000000111010

2’s comp =1111111111000110 = FFC6H and (CF) = 1

BCD Arithmetic:

The 8086 provides two instructions to support BCD arithmetic. They correct result of a BCD addition and a BCD subtraction. The DAA (decimal adjust after addition) instruction follows BCD addition, and the DAS (decimal adjust after subtraction) follows BCD subtraction. Both instructions correct the result of the addition or subtraction so that it is a BCD number.

DAA Instruction : Decimal Adjust Accumulator.

This instruction is used to make sure the result of adding two packed BCD numbers is adjusted to be a legal BCD number.

Instruction works as follows :

1. If the value of the low-order four bits (D_3-D_0) in the AL is greater than 9 or if AF is set, the instruction adds 6 (06) to the low-order four bits.

2. If the value of the high-order four bits (D_7-D_4) in the AL is greater than 9 or if carry flag is set, the instruction adds 6 (60) to the high-order four bits.

Examples:-

1- ; AL = 0011 1001 = 39 **BCD** , CL = 0001 0010 = 12 **BCD**

Add AL, CL ; AL = 0100 1011 = 4B**H**

DAA ; add 0110 Because 1011 > 9

; AL = 0101 0001 = 51 **BCD**

2-

; AL = 1001 0110 = 96 **BCD** , BL 0000 0111= 07 **BCD**

ADD AL, BL ; AL= 1001 1101= 9D**H**

DAA ; add 0110 Because 1101 > 9

; AL = 1010 0011 = A3**H**

; 1010 > 9 so add 0110 0000

; AL = 0000 0011 = 03 **BCD**, CF 1. The result is **103**.

The instruction updates the AF, CF, PF, and ZF. The OF is undefined after DAA instruction.

Note : only works for AL.

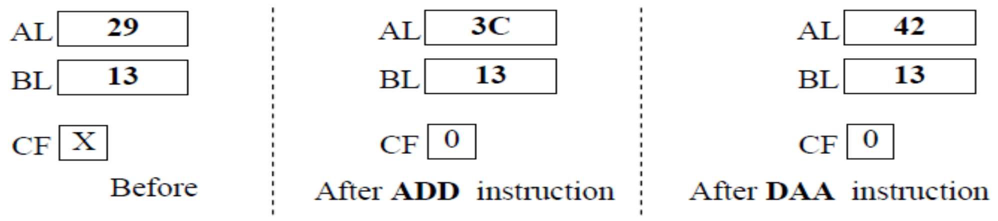
Ex: what is the result of executing the following instruction sequence?

ADD AL , BL

DAA

Assume that AL contains 29H (the BCD code for decimal number 29), BL contain 13H (the BCD code for decimal number 13) , and AH has been cleared.

Sol:



DAS Instruction : Decimal Adjust After Subtraction.

This instruction is used after subtracting two packed BCD numbers to make sure the result is correct packed BCD. **Instruction works as follows :**

1- If the value of the low-order four bits (D_3-D_0) in the AL is greater than 9 or if AF is set; the instruction subtracts 6 (06) from the low-order four bits.

2- If the value of the high-order four bits (D_7-D_4) in the AL is greater than 9 or if carry flag is set, the instruction subtracts 6 (60) from the high-order four bits.

Examples:

```
1.                  ;  AL = 0011 0010 = 32 BCD , CL = 0001 0111 = 17 BCD
SUB AL, CL          ;  AL = 0001 1011 = 1BH
DAS                 ;  Subtract 0110 Because 1011 > 9
                    ;  AL = 0001 0101 = 15 BCD


2.                  ;  AL = 0010 0011= 23 BCD ,  CL = 0101 1000 = 58 BCD
SUB AL, CL          ;  AL = 1100 1011 = CB H ,CF = I
DAS                 ; Subtract 0110 (6) because 1011 > 9
                    ;  AL = 1100 0101 = C5H
```

; Subtract 0110 0000 because 1100 > 9

; AL = 0110 0101 = 65 **BCD** , CF = 1

CF = 1 means borrow is needed means number is negative (-65).

The DAS instruction updates the AF, CF, PF, and ZF. The OF flag is undefined after DAS instruction.

Note : DAS only works for AL