2- Arithmetic Instructions:

The bulk of the arithmetic instructions found in any microprocessor include **addition, subtraction, and comparison**. 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,OF, and PF.**

Addition

Addition (ADD) appears in many forms in the microprocessor:

- ✓ **ADD** instruction for 8-, 16- binary addition.
- ✓ **Add-with-carry ,** is introduced with the **ADC** instruction.
- ✓ The increment instruction **(INC).** Increment is a special type of addition that adds 1 to a number.

The following Table illustrates the addressing modes available to the ADD instruction.

Assembly Language	**Operation**
ADD AL,BL	AL = AL + BL
ADD CX,DI	CX = CX + DI
ADD CL,44_H	CL = CL + 44_H
ADD BX,$245F_H$	BX = BX + $245F_H$
ADD [BX],AL	AL adds to the byte contents of the data segment memory location addressed by BX with the sum stored in the same memory location.
ADD CL,[BP]	The byte contents of the stack segment memory location addressed by BP add to CL with the sum stored in CL.
ADD BX,[SI+2]	The word contents of the data segment memory location addressed by SI + 2 add to BX with the sum stored in BX.

ADD [BX+D],DL	DL adds to the byte contents of the data segment memory location addressed by BX + DI with the sum stored in the same memory location.

Register Addition.

The following example shows a simple sequence of instructions that uses register addition to add the contents of several registers. In this example, the contents of AX, BX, CX, and DX are added to form a 16-bit result stored in the AX register.

Example:

ADD AX,BX

ADD AX,CX

ADD AX,DX

Whenever arithmetic instructions execute, the contents of the flag register change. Note that the contents of the interrupt, trap, and other flags do not change due to arithmetic . Only the flags located in the rightmost 8 bits of the flag register and the overflow flag change. These rightmost flags denote the result of the arithmetic or a logic operation.

Any ADD instruction modifies the contents of the sign, zero, carry, auxiliary carry, parity, and overflow flags.

Immediate Addition.

Immediate addition is employed whenever constant or known data are added. An 8-bit immediate addition appears in the next Example. In this example, DL is first loaded with 12_H by using an immediate move instruction. Next, 33_H is added to the 12_H in DL by an immediate addition instruction.

After the addition, the sum (45_H) moves into register DL and the flags change, as follows:

Z = 0 (result not zero)

O = 0 (no overflow)

P = 0 (odd parity)

S = 0 (result positive)

A = 0 (no half-carry)

C = 0 (no carry)

Example:

```
MOV DL,12H
ADD DL,33H
```

- **Memory-to-Register Addition:** Suppose that an application requires memory data to be added to the AL register. The following shows an example that adds two consecutive bytes of data, stored at the data segment offset locations NUMB and , to the AL register.

Example

```
MOV DI,OFFSET NUMB ;address NUMB
MOV AL,0 ;clear sum
ADD AL,[DI] ;add NUMB
ADD AL,[DI+1] ;add NUMB+1
```

The first instruction loads the destination index register (DI) with offset address NUMB. The DI register, used in this example, addresses data in the data segment beginning at memory location NUMB. After clearing the sum to zero, the ADD AL,[DI] instruction adds the contents of memory location NUMB to AL. Finally, the ADD AL,[ ] instruction adds the contents of memory location NUMB plus 1 byte to the AL register. After both ADD instructions execute, the result appears in the AL register as the sum of the contents of NUMB plus the contents of .

Increment Addition.

Increment addition (INC) adds 1 to a register or a memory location. The INC instruction adds 1 to any register or memory location, except a segment register. The following Table illustrates some of the possible forms of the increment instructions available to the 8086 processors. As with other instructions presented thus far, it is impossible to show all variations of the INC instruction because of the large number available.

With indirect memory increments, the size of the data must be described by using the BYTE PTR, WORD PTR, directives.

Assembly Language	**Operation**
INC BL	BL = BL + 1
INC SP	SP = SP + 1
INC AX	AX = EAX + 1
INC BYTE PTR[BX]	Adds 1 to the byte contents of the data segment memory location addressed by BX
INC WORD PTR[SI]	Adds 1 to the word contents of the data segment memory location addressed by SI

Example

```
MOV DI,OFFSET NUMB ;address NUMB
MOV AL,0 ;clear sum
ADD AL,[DI] ;add NUMB
INC DI ;increment DI
ADD AL,[DI] ;add NUMB+1
```

Addition-with-Carry.

An addition-with-carry instruction (ADC) adds the bit in the carry flag (C) to the operand data. This instruction mainly appears in software that adds numbers that are wider than 16 bits in the 8086.

The following Table lists several add-with-carry instructions, with comments that explain their operation. Like the ADD instruction, ADC affects the flags after the addition.

Assembly Language	**Operation**
ADC AL,AH	AL = AL + AH + carry
ADC CX,BX	CX = CX + BX + carry
ADC BX, DX	BX = BX + DX + carry
ADC DH,[BX]	The byte contents of the data segment memory location addressed by BX add to DH with the sum stored in DH
ADC BX,[BP+2]	The word contents of the stack segment memory location addressed

Example

ADD AX,CX

ADC BX,DX

Example

The original contents of AX, BL, memory location SUM, and CF are AX=1234$_H$, BL= AB$_H$, Sum=00CD$_H$ and CF=0 respectively, describe the result of execution the following sequence of instruction:

ADD AX, SUM

ADC BL, 05$_H$

INC SUM

Solution

1. AX= $1234_H + 00CD_H = 4301_H$ CF=0

2. BL= $AB_H +05_H +0=B0_H$ CF=0

3. SUM=$00CD_H + 1=00CE_H$ CF=0

Instructions	AX	BL	SUM	CF
Initial state	1234_H	AB_H	$00CD_H$	0
ADD AX, SUM	4301_H	AB_H	$00CD_H$	0
ADC BL, 05_H	4301_H	$B0_H$	$00CD_H$	0
INC SUM	4301_H	$B0_H$	$00CE_H$	0

Exercises:

1- Assume AX=1100_H, BX=$0ABC_H$, CF=1 what is the content of AX and carry flag after executing the following instruction:

ADC AX,BX

2- Assume AX=$4F3D_H$, BX=$FD81_H$, CF=1, what is the results after executing the following instruction:

ADC AX,BX

What is the states of state flags?

Subtraction

Many forms of subtraction **(SUB)** appear in the instruction set. These forms use any addressing mode with 8-, 16-, or 32-bit data.

- ✓ A special form of subtraction (decrement, or **DEC**) subtracts 1 from any register or memory location.
- ✓ As with addition, numbers that are wider than 16 bits must occasionally be subtracted. The **subtract-with-borrow instruction** (**SBB**) performs this type of subtraction.

Table below lists some of the many addressing modes allowed with the subtract instruction (SUB).

About the only types of subtraction not allowed are **memory-to-memory and segment register subtractions**. Like other arithmetic instructions, the subtract instruction affects the flag bits.

Register Subtraction.

Example below shows a sequence of instructions that perform register subtraction. This example subtracts the 16-bit contents of registers CX and DX from the contents of register BX. After each subtraction, the microprocessor modifies the contents of the flag register. The flags change for most arithmetic and logic operations.

Example

```
SUB BX,CX
SUB BX,DX
```

Immediate Subtraction.

As with addition, the microprocessor also allows immediate operands for the subtraction of constant data. Example below presents a short sequence of instructions that subtract 44_H from 22_H.

Example

MOV CH,22_H

SUB CH,44_H

Here, we **first** load the 22_H into CH using an immediate move instruction. **Next,** the SUB instruction, using immediate data 44_H, subtracts 44_H from the 22_H.

After the subtraction, the difference ($0DE_H$) moves into the CH register. The flags change as follows for this subtraction:

Z = 0 (result not zero)

O = 0 (no overflow)

P = 1 (even parity)

S = 1 (result negative)

A = 1 (half-borrow)

C = 1 (borrow)

✓ **Note:**

Both carry flags (C and A) hold borrows after a subtraction instead of carries, as after an addition. Notice in this example that there is no overflow. This example subtracted (44_H) from (22_H), resulting in a ($0DE_H$). Because the correct 8-bit signed result is , there is no overflow in this example. An 8-bit overflow occurs only if the signed result is greater than or less than .

Some of the many addressing modes allowed with the subtract instruction (SUB).

Assembly Language	**Operation**
SUB CL,BL	CL = CL – BL
SUB AX,SP	AX = AX – SP
SUB DH,$6F_H$	DH = DH – $6F_H$
SUB AX,$0CCCC_H$	AX = AX – $0CCCC_H$

SUB SI,2300_H	SI = SI – 2300_H
SUB [DI],CH	Subtracts CH from the byte contents of the data segment memory addressed by DI and stores the difference in the same memory location
SUB CH,[BP]	Subtracts the byte contents of the stack segment memory location addressed by BP from CH and stores the difference in CH

Decrement Subtraction.

Decrement subtraction **(DEC)** subtracts 1 from a register or the contents of a memory location. Table below lists some decrement instructions that illustrate register and memory decrements.

The decrement indirect memory data instructions require BYTE PTR, WORD PTR, because the assembler cannot distinguish a byte from a word when an index register addresses memory. **For example,** **DEC [SI]** is vague (مبهم) because the assembler cannot determine whether the location addressed by SI is a byte, word. Using **DEC BYTE PTR[SI], DEC WORD PTR[DI]** reveals(يَكْشِفُ) the size of the data to the assembler.

Assembly Language	**Operation**
DEC BH	BH = BH – 1
DEC CX	CX = CX – 1
DEC BYTE PTR[DI]	Subtracts 1 from the byte contents of the data segment memory location addressed by DI.
DEC WORD PTR[BP]	Subtracts 1 from the word contents of the stack segment memory.

Subtraction-with-Borrow.

A subtraction-with-borrow (**SBB**) instruction functions as a regular subtraction, except that the carry flag (C), which holds the borrow, also subtracts from the difference.

The most common use for this instruction is for subtractions that are wider than 16 bits in the 8086 microprocessors. Wide subtractions require that borrows propagate through the subtraction, just as wide additions propagate the carry. Table below lists several SBB instructions with comments that define their operations.

Like the SUB instruction, SBB affects the flags. Notice that the immediate subtract from memory instruction in this table requires a BYTE PTR, WORD PTR.

Assembly Language	**Operation**
SBB AH,AL	AH = AH – AL – carry
SBB AX,BX	AX = AX – BX – carry
SBB CL,2	CL = CL – 2 – carry
SBB BYTE PTR[DI],3	Both 3 and carry subtract from the data segment memory location addressed by DI.
SBB [DI],AL	Both AL and carry subtract from the data segment memory location addressed by DI.
SBB DI,[BP+2]	Both carry and the word contents of the stack segment memory.

Comparison

The comparison instruction (CMP) is a subtraction that changes only the flag bits; the destination operand never changes. A comparison is useful for checking the entire contents of a register or a memory location against another value. A

CMP is normally followed by a conditional jump instruction, which tests the condition of the flag bits.

Table below lists a variety of comparison instructions that use the same addressing modes as the addition and subtraction instructions already presented. Similarly, the only **disallowed** forms of compare are **memory-to-memory and segment register compares**.

Assembly Language	**Operation**
CMP CL,BL	CL – BL
CMP AX,SP	AX – SP
CMP AX,2000_H	AX – 2000_H
CMP [DI],CH	CH subtracts from the byte contents of the data segment memory location addressed by DI.
CMP CL,[BP]	The byte contents of the stack segment memory location addressed by BP subtracts from CL.
CMP AL,[DI+SI]	The byte contents of the data segment memory location addressed by DI plus SI subtracts from AL.

Example:

If AX=0400_H , what is the result after executing the following instruction :

SUB AX,$03F8_H$

What is the states of state flags?

Solution:

AX	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

2's complement 03F8	1	1	1	1	1	1	0	0	0	0	0	0	1	0	0	0

0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

AX=0008$_H$

ZF=0

PF=0

SF=0

AF=1 (لانه لايوجد تحميل من النصف الادنى للاعلى في البايت الاسفل.. هذا يعني يوجد استعاره)

CF=0 (لانه يوجد باقي هذا يعني لايوجد استعارة لذلك لم يتفعل هذا العلم)

OF=0

Exercises:

1- If AX=1400$_H$, CF=1 , what is the result after executing the following instruction :

SBB AX,03F8$_H$

What is the states of state flags?

2- If BX=AB00$_H$, what is the result after executing the following instruction :

DEC BX

What is the states of state flags?