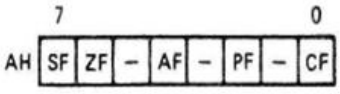
5- Flag – Control Instruction:

The 8086 microprocessor has a set of flags that either monitors the state of executing instructions or controls options available in its operation.
The instruction set includes a group of instructions that. when executed, directly affect the state of the flags. These instructions, shown in Fig. (a).

Mnemonic	Meaning	Operation	Flags Affected
LAHF	Load AH from flags	(AH) ← (Flags)	None
SAHF	Store AH into flags	(Flags) ← (AH)	SF, ZF, AF, PF, CF
CLC	Clear carry flag	(CF) ← 0	CF
STC	Set carry flag	(CF) ← 1	CF
CMC	Complement carry flag	(CF) ← $(\overline{CF})$	CF
CLI	Clear interrupt flag	(IF) ← 0	IF
STI	Set interrupt flag	(IF) ← 1	IF

Fig (a)



SF = Sign flag
ZF = Zero flag
AF = Auxiliary
PF = Parity flag
CF = Carry flag
– = Undefined (do not use)

Fig (b)

- **LAHF & SAHF instructions**:

The first two instructions, LAHF ***load AH from flags*** & SAHF ***store AH into flags*** can be used either to read the flags or to change them, respectively. Notice that the data transfer that takes place is always between the **AH** register and the flag (**status**) register.

Figure (b) shows the format of the flag information in AH. Notice that **bits l, 3, and 5** are undefined, depended on the emulator or kits of 8086 for example the emulator 8086 take bit 1 equal to 1, bit 3 equal to 0 & bit 5 equal to 0 .

If we may want to start an operation with certain flags set or reset. Assume that we want to preset all flags to logic 1. To do this, we can first load AH with FF H and then execute the SAHF instruction.

Ex: Write an instruction sequence to save the current contents of the 8086's flags in the memory location at offset MEM1 of the current data segment and then reload the flags with the contents of the storage location at offset MEM2.

Solution:

LAHF
MOV [MEM1] , AH
MOV AH,[MEM2]
SAHL

To save the current flags, we must first load them into the AH register and then move them to the location MEM1. The instructions that do this are

LAHF
MOV [MEM1] , AH

Similarly, to load the flags with the contents of MEM2, we must First copy the contents of MEM2 into AH and then store the contents of AH into the flags. The instructions for this are

MOV AH,[MEM2]
SAHL

- **CLC, STC & CMC INSTRUCTIONS:**

The next three instructions, CLC ***clear carry***, STC ***set carry***, and CMC ***complement carry***, as shown in Fig. (a). are used to manipulate the carry flag and permit CF to be cleared, set, or complemented, respectively.
For example, if CF is 1 and the CMC instruction is executed, it becomes 0.

- **CLI & STI INSTRUCTION:**

The last two instructions CLI ***clear interrupt*** & STI ***set interrupt*** are used to manipulate the interrupt flag.
Executing the clear interrupt **CLI** instruction sets **IF** to logic 0 and disables the interrupt interface. On the other hand, executing the **STI** instruction sets **IF** to 1, and the microprocessor is enabled to accept interrupts from that point on.

Ex:
Of the three carry flag instructions CLC, STC, and CMC, only one is really an independent instruction—that is, the operation that it provides cannot be performed by a series of the other two instructions. Determine which one of the carry instructions is the independent instruction.

Solution:

Let us begin with the CLC instruction.

The clear-carry operation can be performed by an STC instruction followed by a CMC instruction. Therefore, CLC is not an independent instruction.

The operation of the set-carry (STC) instruction is equivalent to the operation performed by a CLC instruction, followed by a CMC instruction. Thus, STC is also not an independent instruction.

On the other hand, the operation performed by the last instruction, complement carry (CMC), cannot be expressed in terms of the CLC and STC instructions. Therefore, it is the independent instruction.

6- Compare instruction

An instruction is included in the instruction set of the 8086 that can be used to compare two 8-bit or 16-bit numbers. It is the compare (CMP) instruction shown in Fig. (a).

Mnemonic	Meaning	Format	Operation	Flags Affected
CMP	Compare	CMP D,S	(D) − (S) is used in setting or resetting the flags	CF, AF, OF, PF, SF, ZF

(a)

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

(b)

The compare operation enables us to determine the relationship between two numbers that is, whether they are equal or unequal, and when they are unequal, which one is larger.

Figure (b) shows that the operands for this instruction . The source and destination both must be byte or word.

The source may be an immediate number, a register, or a memory location.

The destination may be a register or a memory location.

However the source and destination both **can't be memory locations**.

The comparison is done by subtracting the source byte or word from the destination byte or word. But **the result is not stored in the destination**.

Source and destination remain unchanged. only **flags are updated**.

Flags : The AF, OF, SF, ZF, PF and CF are updated by the CMP instruction.

The new logic state of these flags can be used by the instructions that follow to make a decision whether or not to alter the sequence in which the program executes.

The importance of the flags lies in the fact that they lead us to an understanding of the relationship between the two numbers.

Ex:

if 5 is compared to 7 by subtracting 5 from 7, the ZF and CF both become logic 0. These conditions indicate that a smaller number was compared to a larger one.

7 > 5 → 7-5 = 2 → ZF= 0 & CF = 0

On the other hand, if 7 is compared to 5, we are comparing a larger number to a smaller number. This comparison result in ZF and CF equal to 0 and 1, respectively.

5 > 7 → 5 -7 = -2 → ZF= 0 & CF =1

If two equal numbers—for instance. 5 and 5 are compared, ZF is set to 1 and CF cleared to 0 to indicate the equal condition.

5 > 5 → 5 -5 = 0 → ZF = 1 & CF = 0

So to check if two numbers are equal by checking the ZF. To check two numbers relationship which is are larger or smaller by checking the CF.

Ex:

let us assume that the destination operand equals 10011001B = -103Dec. and that the source operand equals 00011011B = +27Dec.

Subtracting the source operand from the destination operand, we get

```
  10011001 =  -103
-
  00011011 =  -(+27)
-------------------
  01111110 =  +126
```

In the process of subtraction, we get the **status** that follows:

1. A borrow is needed from bit 4 to bit 3; therefore, the auxiliary carry flag, AF, is set
2. There is no borrow to bit 7. Thus, carry flag, CF, is reset.

3. There is an even number of 1's in the result; therefore, this sets the parity flag. PF.
4. Bit 7 of the result is zero, so the sign flag, SF, is reset.
5. The result that is produced is nonzero, which resets the zero flag, ZF.
6. Note that the 8-bit result of binary subtraction is not what the subtraction of the signed numbers should produce. The overflow flag OF having been set indicates this condition.

Ex:

Describe what happens to the status flags as the sequence of instructions that follow is executed.

MOV AX, 1234 H
MOV BX, ABCD H
CMP AX, BX

Assume that flags ZF, SF, CF, AF, OF, and PF are all initially reset.

Solution:

The first instruction loads AX with 1234 H. No status flags are affected by the execution of a MOV instruction. The second instruction puts ABCD H into the BX register. Again, status is not affected. Thus, after execution of these two move instructions, the contents of AX and BX are **(AX) = 1234 H = 0001 0010 0011 0100** and **(BX) = ABCD H = 1010 1011 1100 1101**

The third instruction is a 16-bit comparison with AX representing the destination and BX the source. Therefore, the contents of BX are subtracted from that of AX:

(AX) — (BX) = 0001 0010 0011 0100 — 1010 1011 1100 1101
= 0110 0110 0110 0111 = 6667 H

ZF = 0 , SF = 0 , OF = 0 , CF = 1 , AF = 1 & PF = 0

The flags are either set or reset based on the result of this subtraction. Note that the result is nonzero and positive. This makes ZF and SF equal to zero. Moreover, the overflow condition has not occurred. Therefore, OF is also at logic 0. The carry and auxiliary carry conditions occur and make CF and AF equal 1. Finally, the result has odd parity; therefore, PF is 0. Table below summarizes these results.

instruction	ZF	SF	CF	AF	OF	PF
Initial state	0	0	0	0	0	0
MOV AX,1234 H	0	0	0	0	0	0
MOV BX, ABCD H	0	0	0	0	0	0
CMP AX,BX	0	0	1	1	0	0