Chapter-5

The Processor Status and the FLAGS Register

Overview

The circuits in the CPU can perform simple decision making based on the current state of the processor. For the 8086 processor, the processor state is implemented as nine individual bits called **flags**. Each decision made by the 8086 is based on the values of these flags.

A flag is a flip-flop that indicates some condition produced by the execution of an instruction. It also controls certain operations of the execution unit. A 16-bit flag register in the execution unit contains nine active flags.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				OF	DF	IF	TF	SF	ZF		AF		PF		CF

Fig.5.1: The FLAGS Register.

5.1 The FLAGS Register

Figure 5.1 shows the FLAGS register. The status flags are located in bits 0,2,4,6,7, and 11 and the control flags are located in bits 8,9, and 10. The bits have no significance.

The Status Flags

Six of the nine flags are called status flags and used to indicate some conditions produced by execution of some instructions. The six conditional flags (status flags) are:

CF	Carry Flag
PF	Parity Flag
AF	Auxiliary carry Flag
ZF	Zero Flag
SF	Sign Flag
OF	Overflow Flag

These flags are set or reset by the execution unit on the basis of results of some arithmetic or logic operations.

Carry Flags (CF)

CF = 1 if there is a carry out from the most significant bit (msb) on addition, or there is a borrow into the msb on subtraction; otherwise, it is 0. CF is also affected by shift and rotate instructions.

Parity Flag (PF)

PF = 1 if the low byte of a result has an even number of one bits (even parity). It is 0 if the low byte has odd parity. For example, if the result of a word addition is FFFEH, then the low byte contains 7 one bits, so PF = 0.

Auxiliary Flag (AF)

AF = 1 if there is a carry out from bit 3 on addition, or a borrow into bit 3 on subtraction. AF is used in binary-coded decimal (BCD) operations.

Zero Flag (ZF)

ZF = 1 for a zero result, and ZF = 0 for a nonzero result.

Sign Flag (SF)

SF = 1 if the msb of a result is 1; it means the result is negative for a signed interpretation. SF = 0 if the msb is 0.

Overflow Flag (OF)

OF = 1 if signed overflow occurred, otherwise it is 0.

The three remaining flags of the flags register are used to control certain operation of the processor and hence they are call control flags. They are:

TF	Trap Flag
IF	Interrupt Flag
DF	Direction Flag

The TF is used for single stepping through a program. IF is used for interruption and DF is used in string operations.

5.2 Overflow

When we perform an arithmetic operation such as addition, there are four possible outcomes: (1) no overflow (2) signed overflow only, (3) unsigned overflow only, and (4) both signed and unsigned overflows.

As an example of unsigned overflow but not signed overflow, suppose AX contains FFFFH, BX contains 0001H, and ADD AX,BX is executed. The binary result is

```
     1111111111111111
   + 0000000000000001
     --------------------------
   1 0000000000000000
```

If we are giving an unsigned interpretation, the correct answer is 10000H=65536, but this is out of range for a word operation. A 1 is carried out of the msb and the answer stored in AX, 0000H, is wrong, so unsigned overflow occurred.

But the stored answer is correct as a signed number, for FFFFH = -1, 0001H = 1, FFFFH + 0001H = -1 + 1 = 0, so signed overflow did not occur.

As an example of signed but not unsigned overflow, suppose AX and BX both contain 7FFFH, and we execute ADD AX,BX. The binary result is

```
     0111111111111111
  +  0111111111111111
    --------------------------
    1111111111111110 = FFFEH
```

The signed and unsigned decimal interpretation of 7FFFH is 32767. Thus 7FFFH + 7FFFH = 32767 + 32767 = 65534. It is not unsigned overflow but signed overflow, because FFFEH is -2.

How the Processor Indicates Overflow

The processor sets OF = 1 for signed overflow and CF = 1 for unsigned overflow. It is then up to program to take appropriate action.

In determining overflow, the processor does not interpret the result as either signed or unsigned. The action it takes is to use both interpretations for each operation and to turn on CF or OF for unsigned overflow and signed overflow, respectively.

Unsigned Overflow

On addition, the unsigned overflow occurs when there is a carry out of the msb. This means that the correct answer is larger than the biggest unsigned number; that is FFFFH for word and FFH for a byte. On subtraction, unsigned overflow occur when there is a borrow into the msb . This means that the correct answer is smaller than 0.

Signed Overflow

On addition of numbers with the same sign, signed overflow occurs. In the preceding example, when we were adding 7FFFH and 7FFFH (two positive numbers), but we get FFFEH (a negative number).

Subtraction operation A-B can be performed as operation A+ (-B). OF=1 if the result has a different sign than expected.

Overflow Flag

The rules for turning on the overflow flag in binary/integer math are two:
1. If the sum of two numbers with the sign bits off yields a result number with the sign bit on, the "overflow" flag is turned on.

0100 + 0100 = 1000 (overflow flag is turned on)

2. If the sum of two numbers with the sign bits on yields a result number with the sign bit off, the "overflow" flag is turned on.
1000 + 1000 = 0000 (overflow flag is turned on)

Otherwise, the overflow flag is turned off. * 0100 + 0001 = 0101 (overflow flag is turned off)
* 0110 + 1001 = 1111 (overflow flag is turned off)
* 1000 + 0001 = 1001 (overflow flag is turned off)
* 1100 + 1100 = 1000 (overflow flag is turned off)

5.3 How Instructions Affect the Flags

In general, each time the processor executes an instruction, the flags are altered to reflect the result. However, some instructions don't affect any of the flags, affect only some of them, or may leave them undefined.

Instructions	**Affects Flags**
MOV/XCHG	non
ADD/SUB	all
INC/DEC	all except CF
NEG	all (CF=1 unless result is 0.
	OF=1 if word operand is 8000H
	or byte operand is 80H)

Example 5.1: ADD AX,BX , where AX contains FFFFH, BX contains FFFFH.

Solution:

```
       FFFFH
       FFFFH
      ---------
    1  FFFEH
```

The result store in AX is FFFEH = 1111 1111 1111 1110

SF = 1 because the msb is 1.

PF = 0 because the low byte contains odd number of 1.

ZF = 0 because the result is nonzero.

CF = 1 because there is carry out of the msb.