SF, the Sign Flag. Binary representation of signed numbers uses the most significant bit as the sign bit. After arithmetic or logic operations, the status of this sign bit is copied into the SF, thereby indicating the sign of the result.
TF, the Trap Flag. When this flag is set it allows the program to single-step, meaning to execute one instruction at a time. Single-stepping is used for debugging purposes.
IF**, Interrupt Enable Flag**. This bit is set or cleared to enable or disable only the external mask able interrupt requests.
DF**, the Direction Flag**. This bit is used to control the direction of string operations.
OF**, the Overflow Flag**. This flag is set whenever the result of a signed number operation is too large, causing the high-order bit to overflow into the sign bit. In general, the carry flag is used to detect errors in unsigned arithmetic operations. The overflow flag is only used to detect errors in signed arithmetic operations.

Flag register and ADD instruction

In this section we examine the impact of the ADD instruction on the flag register as an example of the use of the flag bits. The flag bits affected by the ADD instruction are CF (carry flag), PF (parity flag), AF (auxiliary carry flag), ZF (zero flag), SF (sign flag), and OF (overflow flag).

Example 1-10

Show how the flag register is affected by the addition of 38H and 2FH.

Solution:

```
MOV   BH,38H      ;BH= 38H
ADD   BH,2FH      ;add 2F to BH, now BH=67H

      38          0011  1000
+     2F          0010  1111
      67          0110  0111
```

CF = 0 since there is no carry beyond d7
PF = 0 since there is an odd number of 1s in the result
AF = 1 since there is a carry from d3 to d4
ZF = 0 since the result is not zero
SF = 0 since d7 of the result is zero

Example 1-11

Show how the flag register is affected by

```
MOV   AL,9CH      ;AL=9CH
MOV   DH,64H      ;DH=64H
ADD   AL,DH       ;now AL=0
```

Solution:

```
        9C        1001  1100
+       64        0110  0100
        00        0000  0000
```

CF=1 since there is a carry beyond d7
PF=1 since there is an even number of 1s in the result
AF=1 since there is a carry from d3 to d4
ZF=1 since the result is zero
SF=0 since d7 of the result is zero

The same concepts apply for 16-bit addition, as shown in Examples 1-12 and 1-13. It is important to notice the differences between 8-bit and 16-bit operations in terms of their impact on the flag bits. The parity bit only counts the lower 8-bits of the result and is set accordingly. Also notice the CF bit. The carry flag is set if there is a carry beyond bit d15 instead of bit d7.

Example 1-12

Show how the flag register is affected by

```
MOV   AX,34F5H    ;AX= 34F5H
ADD   AX,95EBH    ;now AX= CAE0H
```

Solution:

```
        34F5      0011  0100  1111  0101
+       95EB      1001  0101  1110  1011
        CAE0      1100  1010  1110  0000
```

CF = 0 since there is no carry beyond d15
PF = 0 since there is an odd number of 1s in the lower byte
AF = 1 since there is a carry from d3 to d4
ZF = 0 since the result is not zero
SF = 1 since d15 of the result is one

Example 1-13

Show how the flag register is affected by

```
MOV   BX,AAAAH    ;BX= AAAAH
ADD   BX,5556H    ;now BX= 0000H
```

Solution:

```
        AAAA      1010  1010  1010  1010
+       5556      0101  0101  0101  0110
        0000      0000  0000  0000  0000
```

CF = 1 since there is a carry beyond d15
PF = 1 since there is an even number of 1s in the lower byte
AF = 1 since there is a carry from d3 to d4
ZF = 1 since the result is zero
SF = 0 since d15 of the result is zero

Notice the zero flag (ZF) status after the execution of the ADD instruction. Since the result of the entire 16-bit operation is zero (meaning the contents of BX), ZF is set to high. Do all instructions affect the flag bits? The answer is no; some instructions such as data transfers (MOV) affect no flags. As an exercise, run these examples on DEBUG to see the effect of various instructions on the flag register.

Example 1-14

Show how the flag register is affected by

```
MOV   AX,94C2H     ;AX=94C2H
MOV   BX,323EH     ;BX=323EH
ADD   AX,BX        ;now AX=C700H
MOV   DX,AX        ;now DX=C700H
MOV   CX,DX        ;now CX=C700H
```

Solution:

```
      94C2      1001  0100  1100  0010
+     323E      0011  0010  0011  1110
      C700      1100  0111  0000  0000
```

After the ADD operation, the following are the flag bits:
CF = 0 since there is no carry beyond d15
PF = 1 since there is an even number of 1s in the lower byte
AF = 1 since there is a carry from d3 to d4
ZF = 0 since the result is not zero
SF = 1 since d15 of the result is 1

Running the instructions in Example 1-14 in DEBUG will verify that MOV instructions have no effect on the flag. How these flag bits are used in programming is discussed in future chapters in the context of many applications.

Use of the zero flag for looping

One of the most widely used applications of the flag register is the use of the zero flag to implement program loops. The term loop refers to a set of instructions that is repeated a number of times. For example, to add 5 bytes of data, a counter can be used to keep track of how many times the loop needs to be repeated. Each time the addition is performed the counter is decremented and the zero flag is checked. When the counter becomes zero, the zero flag is set (ZF = I) and the loop is stopped. The following shows the implementation of the looping concept in the program, which adds 5 bytes of data. Register CX is used to hold the counter and BX is the offset pointer (SI or or could have been used instead). AL is initialized before the start of the loop. In each iteration, ZF is checked by the JNZ instruction.

JNZ stands for "Jump Not Zero" meaning that if ZF = 0, jump to a new address. If ZF = 1, the jump is not performed and the instruction below the jump will be executed. Notice that the JNZ instruction must come immediately after the instruction that decrements CX since JNZ needs to check the affect of "DEC CX" on the zero flag. If any instruction were placed between them, that instruction might affect the zero flag.