

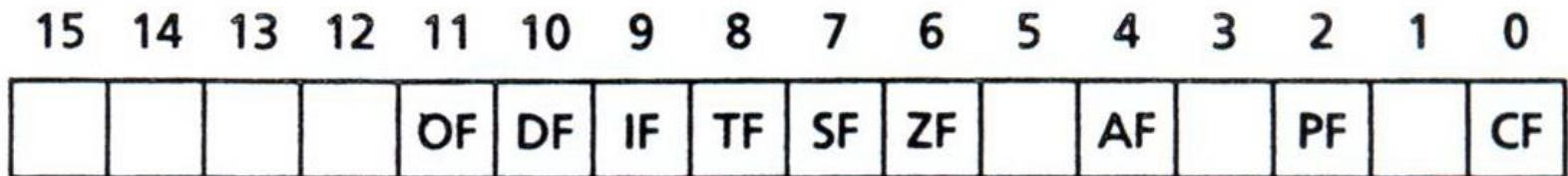
CHAPTER 5

THE PROCESSOR STATUS AND FLAG REGISTERS

Outline

- The FLAG Register
- Overflow
- How Instruction Affect the Flags

The FLAGS Register



The Status Flags

- The processor uses the status flags to reflect the result of an operation.
- Bits 0, 2, 4, 6, 7, 11
- If SUB AX, AX is executed, the zero flag becomes 1, thereby indicating that a zero result was produced.

Carry Flag (CF)

- $CF = 1$ if there is a carry out of msb on addition, or there is a borrow into msb on subtraction; otherwise, $CF = 0$.

Parity Flag (PF)

- $PF = 1$ if the low byte of a result has an even number of one bits (even parity).
- $PF = 0$ if the low byte has odd parity.
- If the result of a word addition is `FFFEh`, then the low byte contains 7 one bits, so $PF = 0$.

Auxiliary Carry Flag (AF)

- $AF = 1$ if there is a carry out from bit 3 on addition, or a borrow into bit 3 on subtraction.

Zero Flag (ZF)

- $ZF = 1$ for a zero result.
- $ZF = 0$ for a nonzero result.

Sign Flag (SF)

- $SF = 1$ if the msb of a result is 1; it means the result is negative if you are giving a signed interpretation.
- $SF = 0$ if the msb is 0.

Overflow Flag (OF)

- $OF = 1$ if signed overflow occurred, otherwise $OF = 0$.

Unsigned Overflow: ADD AX, BX

Unsigned Interpretation

1111 1111 1111 1111	65535	AX = FFFFh
+ 0000 0000 0000 0001	1	BX = 0001h
<u>1 0000 0000 0000 0000</u>	0	AX = 0000h

Signed Interpretation

1111 1111 1111 1111	-1	AX = FFFFh
+ 0000 0000 0000 0001	1	BX = 0001h
<u>1 0000 0000 0000 0000</u>	0	AX = 0000h

Signed Overflow: ADD AX, BX

Unsigned Interpretation

0111 1111 1111 1111	32767	AX = 7FFFh
+ 0111 1111 1111 1111	32767	BX = 7FFFh
<u>1111 1111 1111 1110</u>	65534	AX = FFFEh

Signed Interpretation

0111 1111 1111 1111	32767	AX = 7FFFh
+ 0111 1111 1111 1111	32767	BX = 7FFFh
<u>1111 1111 1111 1110</u>	-2	AX = FFFEh

How the Processor Determines that Unsigned Overflow Occurred

- $CF = 1$
- Addition
 - The correct answer is larger than the biggest unsigned number (FFFFh and FFh).
- Subtraction
 - The correct answer is smaller than 0.
 - There is a borrow into the msb.

How the Processor Determines that Signed Overflow Occurred

- $OF = 1$
 - There is a carry into the msb but no carry out.
 - There is a carry out but no carry in.
- Addition
 - The sum has a different sign.
- Subtraction
 - The result has a different sign than expected.
 - $A - (-B) = A + B$
 - $-A - B = -A + -B$
- Addition of Numbers with Different Signs
 - Overflow is impossible.
 - $A + (-B) = A - B$

How Instructions Affect the Flags

Instruction	Affects Flags
MOV/XCHG	none
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)

ADD AX, BX where AX contains FFFFh and BX contains FFFFh.

FFFFh	1111 1111 1111 1111	
+ FFFFh	+ 1111 1111 1111 1111	
<u>1 FFFEh</u>	<u>1 1111 1111 1111 1110</u>	AX = FFFEh

SF = 1 because the msb is 1.

PF = 0 because there are 7 (odd number) of 1 bits in the low byte of the result.

ZF = 0 because the result is nonzero.

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

OF = 0 because the sign of the stored result is the same as that of the numbers being added (as a binary addition, there is a carry into the msb and also a carry out).

ADD AL, BL where AL contains 80h and BL contains 80h.

80h	1000 0000	
<u>+ 80h</u>	<u>+ 1000 0000</u>	
± 00h	± 0000 0000	AL = 00h

SF = 0 because the msb is 0.

PF = 1 because all the bits in the result are 0.

ZF = 1 because the result is 0.

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

OF = 1 because the numbers being added are both negative, but the result is 0 (as a binary addition, there is no carry into the msb but there is a carry out).

SUB AX, BX where AX contains 8000h and BX contains 0001h.

8000h	1000 0000 0000 0000
<u>- 0001h</u>	<u>- 0000 0000 0000 0001</u>
7FFFh	0111 1111 1111 1111 AX = 7FFFh

SF = 0 because the msb is 0.

PF = 1 because there are 8 (even number) one bits in the low byte of the result.

ZF = 0 because the result is nonzero.

CF = 0 because a smaller unsigned number is being subtracted from a larger one.

OF = 1 because in a signed sense we are subtracting a positive number from a negative one, which is like adding two negatives but the result is positive (the wrong sign).

INC AL where AL contains FFh.

FFh	1111 1111	
+ 1h	+ 0000 0001	
<u> </u>	<u> </u>	
± 00h	± 0000 0000	AL = 00h

SF = 0, PF = 1, ZF = 1.

CF is unaffected by INC.

If CF = 0 before the execution of the instruction, CF will still be 0 afterward.

OF = 0 because numbers of unlike sign are being added (there is a carry into the msb and also a carry out).

MOV AX, -5

AX = FFFBh

None of the flags are affected by MOV.

NEG AX where AX contains 8000h.

$$\begin{array}{rcll} 8000h & = & 1000\ 0000\ 0000\ 0000 & \\ \text{one's complement} & = & 0111\ 1111\ 1111\ 1111 & \\ & & \underline{\hspace{10em} +1} & \\ & = & 1000\ 0000\ 0000\ 0000 & = 8000h \end{array}$$

SF = 1, PF = 1, ZF = 0.

CF = 1 because for NEG CF is always 1 unless the result is 0.

OF = 1 because the result is 8000h; when a number is negated, we would expect a sign change, but because 8000h is its own two's complement, there is no sign change.