UMBC CMSC211

This page was developed by Jeff Walton, a former student, and I dedicate this page to him, in that hope that some day he will decide it is finally time to graduate.

All about the Flags Register

The Flag Register is covered in our Textbook in Chapter 5, page 98. Chapter 5 is a bit ahead of binary arithmetic, so to simplify matters lets look at what interests us right now. As you can tell, there's a lot of background material to cover before we can sink our teeth into programming in assembly language. The first few weeks are difficult because of topics like this - the Flags Register is covered in Chapter 5, but we are just starting Chapter 3. Yet we need a knowledge of the Flags Register now. So, here's a quick primer on the Flags Register.

The 8088/8086 Flags Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				O	D	I	T	S	Z		A		P		C

What Do the Letters Mean?

C	Carry
P	Parity
A	Auxillary Carry
Z	Zero
S	Sign
T	Trap
I	Interrupt
D	Direction
O	Overflow

And how about the numbers?

Bit position. The Flags Register may be represented as a value, but this value really isn't good for anything. The Flags Register is a bit masked value. The various bits that are set are of interest to us.

The Bits

Carry Bit

Indicates carry after addition or a borrow after subtraction. The carry flag also indicates an error condition after some programs and procedures.

Parity Bit

The parity bit is a logical 0 for odd parity and a logical 1 for even parity. Parity is a count of the 1s and 0s and is expressed as Even or Odd. For example, if a number contains 3 binary 1 bits, it has Odd parity. If a number contains zero 1 bits, it is said to have Even parity.

Auxillary Carry Bit

Holds a carry after addition or a borrow after subtraction between bit positions 3 and 4 of the ***result***. This **highly specialized** flag bit is tested by DAA and DAS instructions to adjust the value of AL after a BCD (binary coded decimal) addition or subtraction. Otherwise, the **A** flag is not used by the microprocessor.

Zero Bit

Indicates that the ***result*** of an **arithmetic** or **logic** operation is zero. If **Z** = 1, the ***result*** is zero. If **Z** = 0, the ***result*** is not zero.

Sign Bit

Indicates the arithmetic sign of the ***result*** after an addition or subtraction. If **S** = 1, the ***result*** is negative. If S = 0, the ***result*** is positive.

Trap Bit

An onchip debugging feature of the microprocessor. Not discussed in this paper.

Interrupt Bit

An interrupt controls operation of the **INTR** (interrupt request) input pin. If **I** = 1, the **INTR** pin is enabled. If **I** = 0, the **INTR** pin is disabled. The state of the **I** flag bit is controlled by the **STI** (set **I** flag) and **CLI** (clear **I** flag) instruction.

Direction Bit

Controls the selection of increment and decrement for the **DI** and **SI** registers during string instructions. If **D** = 1, the registers are automatically decremented. If **D** = 0, the registers are automatically incremented. The **D** flag is set with **STD** (sexually transmitted disease) and cleared with **CLD** (penicillin) instructions.

Overflow Bit

A condition that can occur when **signed** numbers are added or subtracted. A ***overflow*** condition indicates that a ***result*** has exceeded the capacity of the machine. For example, if **7F**h (+127) is added to **01**h (+1), the ***result*** is **80**h (-128). This ***result*** represents an overflow condition indicated by the overflow flag for **signed** addition. For **unsigned** operations, ignore the overflow flag.

Other Flag Bits

With the advent of the 80286 and above, additional bit values are available. The 80286 Flag Register is 16 bit, while later processors are 32 bit. The Flagsocessors are 32 bit. The Flags Registers of the various processors are upwardly compatible. Additional bits include **IOPL** (Input/Output Priviledge Level), **NT** (Nested Task), **RF** (Resume - used with debugging), **VM** (Virtual Mode - protected mode DOS virtual machine selection), and **AC** (Alignment Check - addressing a **WORD** or **DWORD** on a non **WORD** or **DWORD** boundary). Basically, be aware they exist, but don't concern yourselves with them.

Examples

Example 1:

92	0101 1100 (5Ch)	MSB not Set
+ 69	0100 0101 (45h)	MSB not Set
161	1010 0001 (A1h)	MSB is set

Flags:	O:	1	S:	1	Z:	0	C:	0

(pseudo coding: al=92; bl=69; add al,bl)

Example 2:

92	0101 1100 (5Ch)	MSB not Set
+ 197	1100 0101 (C5h)	MSB is Set
289	0010 0001 (21h)	Is this 289?

Flags:	O:	0	S:	0	Z:	0	C:	1

(pseudo coding: al=92; bl=197; add al,bl)

Example 3:

92	0101 1100 (5Ch)	MSB not Set
+ -69	1100 0101 (BBh)	MSB is Set
23	0001 0111 (17h)	MSB not Set

Flags:	O:	0	S:	0	Z:	0	C:	1

(pseudo coding: al=92; bl=-69; add al,bl)

Example 4:

92	0101 1100 (5Ch)	MSB not Set
- +69	0100 0101 (45h)	MSB not Set
23	0001 0111 (17h)	MSB not Set

Flags:	O:	0	S:	0	Z:	0	C:	0

(pseudo coding: al=92; bl=69; sub al,bl)

Example 5:

-92	1010 0100 (A4h)	MSB is Set
+ -69	1011 1011 (BBh)	MSB is Set
-161	0101 1111 (5Fh)	Bad blowout here. This is 95 (5Fh). What we want is FF5Fh. We're missing something

Flags:	O:	1	S:	0	Z:	0	C:	1

(pseudo coding: al=-92; bl=-69; add al,bl)

Example 6:

-92	1010 0100 (A4h)	MSB is Set
- -69	1011 1011 (BBh)	MSB is Set
-23	1110 1001 (E9h)	MSB is Set

Flags:	O:	0	S:	1	Z:	0	C:	1

(pseudo coding: al=-92; bl=-69; sub al,bl)

Example 7:

92	0101 1100 (5Ch)	MSB not Set
- +92	0101 1100 (5Ch)	MSB not Set
0	0000 0000 (00h)	See Flags

Flags:	O:	0	S:	0	Z:	1	C:	0

(pseudo coding: al=92; bl=92; sub al,bl)

Example 8:

92	0101 1100 (5Ch)	MSB not Set
+ -92	1010 0100 (A4h)	MSB is Set
0	0000 0000 (00h)	See Flags

Flags:	O:	0	S:	0	Z:	1	C:	1

(pseudo coding: al=92; bl=-92; add al,bl)

I still don't understand the Overflow bit

O.K. - I think I got every overflow question wrong on every Quiz and Test myself. So here's the secret (we'll revisit some of the above examples) - I wish somebody would have made it so easy for me!. All the examples are of the general form A + B = C. Whenever the MSB of A does not equal the MSB of C, the Overflow flag is set. Put differently (and more correct): all examples are of the form A = A + B. Whenever the MSB of A (before the operation) is different from the MSB of A (after the operation), Overflow may have occured. Please also note, a carry or borrow (which sets the Carry Bit) is a separate event. And also note how tightly coupled the Sign Bit is to ***result's*** MSB.

Example 1:

```
   92                0xxx xxxx
+  69                0xxx xxxx
  161                1xxx xxxx
```

Overflow Bit is Set

Example 2:

```
   92                0xxx xxxx
+ 197                1xxx xxxx
  289                0xxx xxxx
```

Overflow Bit is NOT Set

Example 3:

```
  -92                1xxx xxxx
- -69                1xxx xxxx
  -23                1xxx xxxx
```

Overflow Bit is NOT set

Question 4:

```
    ??               1xxx xxxx
+/- ??               1xxx xxxx
    ??               0xxx xxxx
```

You guessed it... the Overflow Bit is set

And 16 bit (WORD) math:

```
    ????             1xxx xxxx xxxx xxxx
+/- ????             0xxx xxxx xxxx xxxx
    ????             1xxx xxxx xxxx xxxx
```

Yep... the Overflow Bit is NOTset

And what you see in CodeView

These values can be observed through the Register Window.

Flag	Set (= 1)	Not Set (= 0)

Carry	CY	NC
Parity	PE	PO
Auxillary Carry	AC	NA
Zero	ZR	NZ
Sign	NG	PL
Trap	---	---
Interrupt	EI	DI
Direction	DN	UP
Overflow	OV	NV
