The Flags Register

Basic Assembly

Assembly language programming
By xorpd

xorpd.net

Objectives

- We will learn about the **Flags** register.
- We will learn about some specific flags inside the flags register:
  - The **Zero** flag.
  - The **Sign** flag.
  - The **Carry** flag.
  - The **Overflow** flag.
- In the future we will use the values of those flags to make branches and decisions in our code.

The Flags Register

- A 32-bit register inside the x86 processor.
  - Has 64 bit extension for long-mode.
  - There is no usual direct access to this register.
- Every bit in this register is “a flag”:
  - It means that it represents True or False.
- Contains bits with values that reflect on the result of the last calculation.
  - Has the last calculation resulted in zero?
  - Has the last calculation resulted in a negative number?
  - Has the last calculation resulted in a number that doesn’t fit into a 32 bit register?
- Contains other system related bits.

The Flags register (Cont.)

Bit number	Short name	Description
0	CF	Carry flag
1	1	Reserved
2	PF	Parity flag
3	0	Reserved
4	AF	Auxiliary Carry flag
5	0	Reserved
6	ZF	Zero flag
7	SF	Sign flag
8	TF	Trap flag
9	IF	Interrupt enable flag
10	DF	Direction Flag
11	OF	Overflow flag
More bits …		

The Flags register (Cont.)

Bit number	Short name	Description
0	**CF**	**Carry flag**
1	1	Reserved
2	PF	Parity flag
3	0	Reserved
4	AF	Auxiliary Carry flag
5	0	Reserved
6	**ZF**	**Zero flag**
7	**SF**	**Sign flag**
8	TF	Trap flag
9	IF	Interrupt enable flag
10	DF	Direction Flag
11	**OF**	**Overflow flag**
More bits …		

Changes in the Flags register

- Every instruction can have certain effects on some bits of the flags register.
  - There are some general rules of thumb regarding how each flag is changed by different instructions.
  - With some experience, you will be able to get a feeling of how the flags are going to change.
- The Flags Register is like the “mood” of the processor.
  - It changes when things happen.

The Zero Flag (ZF)

- The zero flag is set (to 1) whenever the last calculation had the result of zero.
- It will be cleared (to 0) whenever the last calculation had a nonzero result.
- Example:

```
mov       eax,3h
mov       ecx,3h
sub       eax,ecx
```

- Right after the execution of the sub instruction, the zero flag will be set to 1.
- In this example, right after the execution of the add instrucion, the zero flag will be cleared to 0.

```
mov       eax,3h
mov       ecx,3h
add       eax,ecx
```

The Sign flag (SF)

- The sign equals the most significant bit of the last result.
  - 0 if the result is positive in the two’s complement representation.
  - 1 if the result is negative in the two’s compliment representation.
- Examples:

```
mov edx,0
dec edx

; edx == 0xffffffff
; Sign flag is 1
```

```
mov edx,0
inc edx

; edx == 1
; Sign flag is 0
```

The Carry flag (CF)

- The carry flag "understands" unsigned addition and subtraction.
- The carry flag is set if the addition of two numbers causes a carry out of the most significant bits (Leftmost bits).
- Example:

```
mov eax,0ffffffffh
add eax,1

; eax == 0
; Carry flag is set.
```

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
+	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- Tells you that the result you got adding two unsigned numbers, is wrong.

The Carry flag (CF)

- The carry flag “understands” unsigned addition and subtraction.
- The carry flag is set if the addition of two numbers causes a carry out of the most significant bits (Leftmost bits).
- Example:

```
mov eax,0ffffffffh
add eax,1

; eax == 0
; Carry flag is set.
```

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
+	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- Tells you that the result you got adding two unsigned numbers, is wrong.

The Carry flag (Cont.)

- The carry flag is also set if the subtraction of two numbers requires a borrow into the most significant bits.
- Example:

```
mov ecx,0
mov edx,11b
sub ecx,edx

; ecx == 0xfffffffd
; Carry flag is 1.
```

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1

The Carry flag (Cont.)

- The carry flag is also set if the subtraction of two numbers requires a borrow into the most significant bits.
- Example:

```
mov ecx,0
mov edx,11b
sub ecx,edx

; ecx == 0xfffffffd
; Carry flag is 1.
```

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1

The Carry flag (Examples)

- Otherwise, the carry flag is cleared (0).
- Examples:

```
mov eax,f0h
mov ecx,35h
add eax,ecx

; eax == 0x125
; Carry flag is 0.
```

```
mov eax,f0h
mov ecx,35h
add al,cl

; al == 0x25
; Carry flag is 1.
```

```
mov eax,f0h
mov ecx,35h
sub eax,ecx

; eax == 0xbb
; Carry flag is 0.
```

```
mov eax,f0h
mov ecx,35h
sub cl,al

; cl == 0x45
; Carry flag is 1.
```

The Overflow flag (OF)

- The overflow flag “understands” signed addition and subtraction.
  - According to the two’s complement representation.
- Addition:
  - Set if the addition of two positive numbers has a negative result.
  - Set if the addition of two negative numbers has positive result.
- Subtraction:
  - Set if “positive – negative” has a negative result.
  - Set if “negative – positive” has a positive result.
- Tells you if something went wrong with your signed arithmetic calculation.

The Overflow flag (Cont.)

- How does it work?
  - The processor looks on the msb of the two operands and the msb of the result.
    - The msb (Most significant bit) is the sign of the number.
  - If the result of the addition/subtraction has a “reasonable” sign, the overflow flag is cleared. If not, it is set.
- “positive + negative” can never set the overflow flag.
  - Because there is no reasonable prediction. The result could be both positive or negative.
  - Same for “positive – positive”, “negative – negative”.

The Overflow flag (Example)

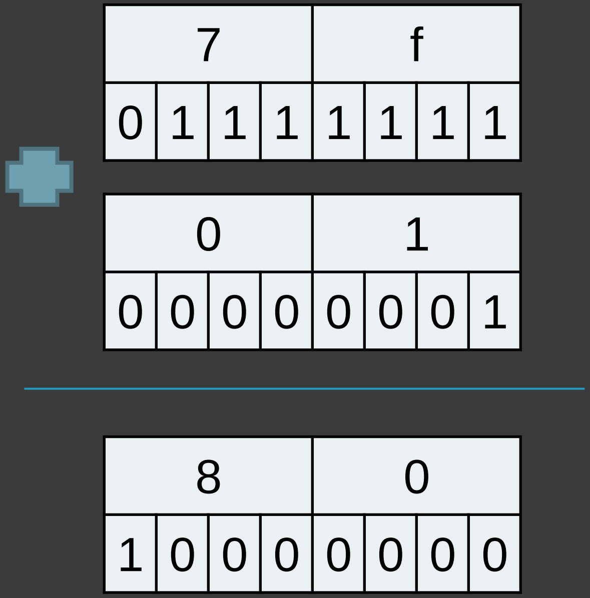
```
mov al,7fh
mov cl,1h
add al,cl

; al == 0x80
; Overflow flag
; is set.
```

The Overflow flag (Example)

```
mov al,7fh
mov cl,1h
add al,cl

; al == 0x80
; Overflow flag
; is set.
```



7				f			
0	1	1	1	1	1	1	1

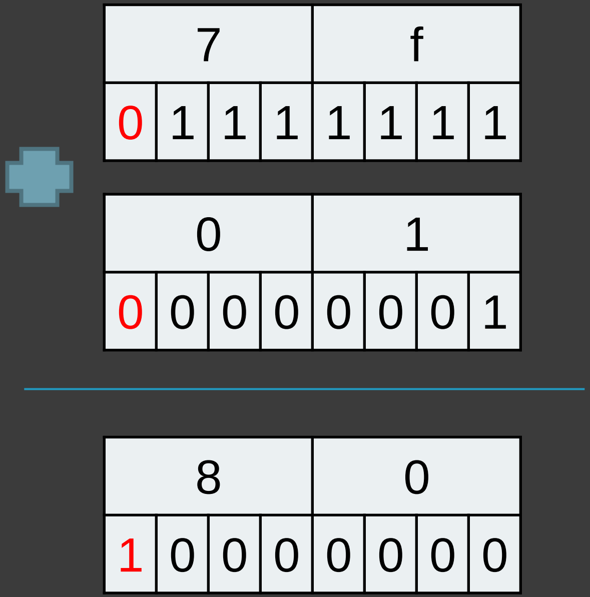
0				1			
0	0	0	0	0	0	0	1

8				0			
1	0	0	0	0	0	0	0

The Overflow flag (Example)

```
mov al,7fh
mov cl,1h
add al,cl

; al == 0x80
; Overflow flag
; is set.
```



7				f			
0	1	1	1	1	1	1	1

+

0				1			
0	0	0	0	0	0	0	1

8				0			
1	0	0	0	0	0	0	0

The Overflow flag (Example)

```
mov al,7fh
mov cl,1h
add al,cl

; al == 0x80
; Overflow flag
; is set.
```

7				f			
0	1	1	1	1	1	1	1

+

0				1			
0	0	0	0	0	0	0	1

8				0			
1	0	0	0	0	0	0	0

- “Positive + Positive = Negative”
  - Wrong singed result → Overflow flag is set.

The Overflow flag (Cont.)

- More examples:

```
mov eax,7fffffffh
mov edx,1h
add eax,edx

; eax == 0x80000000
; Overflow flag is set.
```

```
mov dx,6342h
mov cx,2000h
add cx,dx

; cx == 0x8342
; Overflow flag is set.
```

```
mov eax,7fffffffh
mov edx,1h
sub eax,edx

; eax == 0x7ffffffe
; Overflow flag is cleared.
```

```
mov esi,0ffffffffh
mov edi,0ffffffffh
add esi,edi

; esi == 0xfffffffe
; Overflow flag is cleared.
```

Overflow and Carry flags

- Both of the flags will change in every arithmetic operation.
  - The processor doesn’t care about your interpretation of the bits.
  - It is your responsibility as a programmer to look at the relevant flags.
- Which flag to look at?
  - Depending on how you interpret your numbers.
  - If you work with unsigned numbers, you only care about the **carry** flag.
  - If you work with signed numbers, you only care about the **overflow** flag.

Overflow, Carry Comparison

Code	Carry flag	Overflow flag
mov eax,0x0 sub eax,1		
mov dl,0x7f add dl,0x1		
mov ax,0x5 mov si,0x4 add si,ax		
mov cl,0x80 mov dl,0x80 add cl,dl		

Overflow, Carry Comparison

Code	Carry flag	Overflow flag
mov eax,0x0 sub eax,1 **; eax == 0xffffffff**	1	0
mov dl,0x7f add dl,0x1		
mov ax,0x5 mov si,0x4 add si,ax		
mov cl,0x80 mov dl,0x80 add cl,dl		

Overflow, Carry Comparison

Code	Carry flag	Overflow flag
mov eax,0x0 sub eax,1 **; eax == 0xffffffff**	1	0
mov dl,0x7f add dl,0x1 **; dl == 0x80**	0	1
mov ax,0x5 mov si,0x4 add si,ax		
mov cl,0x80 mov dl,0x80 add cl,dl		

Overflow, Carry Comparison

Code	Carry flag	Overflow flag
mov eax,0x0 sub eax,1 ; **eax == 0xffffffff**	1	0
mov dl,0x7f add dl,0x1 ; **dl == 0x80**	0	1
mov ax,0x5 mov si,0x4 add si,ax ; **si == 0x9**	0	0
mov cl,0x80 mov dl,0x80 add cl,dl		

Overflow, Carry Comparison

Code	Carry flag	Overflow flag
mov eax,0x0 sub eax,1 **; eax == 0xffffffff**	1	0
mov dl,0x7f add dl,0x1 **; dl == 0x80**	0	1
mov ax,0x5 mov si,0x4 add si,ax **; si == 0x9**	0	0
mov cl,0x80 mov dl,0x80 add cl,dl **; cl == 0x0**	1	1

Summary

- The **Zero flag** is 1 iff the last result was zero.
- The **Sign flag** is 1 iff the last result was negative.
- The **Carry flag** is 1 iff the result (considering unsigned numbers) is wrong.
- The **Overflow flag** is 1 iff the result (Considering signed numbers) is wrong.