

# ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

---

## Introdução à linguagem assembly do x86 - Sintaxe – continuação

### O registrador de sinalizadores (FLAGS)

- **Flags de Status e Flags de Controle**
  - indicam o estado do microprocessador após a execução de cada instrução;
  - conjunto de bits individuais, cada qual indicando alguma propriedade;
  - subdividem-se em: Flags de Estado (*status*) e Flags de Controle.

#### Organização

1 registrador de 16 bits

6 FLAGS de estado

3 FLAGS de controle

7 bits não utilizados (sem função)

# ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

## Introdução à linguagem assembly do x86 - Sintaxe – continuação

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	

### Flags de estado

Nome	Símbolo	Função/característica
Carry Flag	CF	Indicador de "vai-um"
Parity Flag	PF	Indicador de número PAR de 1's no byte inferior
Auxiliary Carry	AF	Indicador de "vai-um" para operações em BCD
Zero Flag	ZF	Indicador de "zero" na última operação
Sign Flag	SF	Indicador de resultado negativo
Overflow Flag	OF	Indicador de erro de transbordamento

Obs.: o emprego dos **Flags de Controle** será discutido juntamente com operações com *arrays* e interrupções.

# ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

---

## Introdução à linguagem assembly do x86 - Sintaxe – continuação

- **Overflow (erro de transbordamento)**

**Overflow** → ocorre porque a representação dos números está limitada a uma certa faixa

Tipos	8 bits	16 bits
Não-sinalizado	0 a 255	0 a 65.535
Sinalizado (C2)	-128 a +127	- 32.768 a + 32.767

- Qualquer operação aritmética que tenha como resultado um número fora da faixa de representação, estará produzindo **Overflow**.
- O resultado armazenado no registrador destino estará truncado e terá, portanto, um valor incorreto.

# ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

---

## Introdução à linguagem assembly do x86 - Sintaxe – continuação

- Tem-se dois Flags que podem indicar *overflow*: CF e OF

CF → indica se há um vai-um para fora do MSB (*most significant bit*)

do número → números não sinalizados

se CF = 0 → não ocorreu overflow

se CF = 1 → ocorreu overflow

OF → testa o vem-um que chega e o vai-um gerado no MSB:

se iguais (0 e 0 ou 1 e 1) → OF = 0

se diferentes → OF = 1

# ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

## Introdução à linguagem assembly do x86 - Sintaxe – continuação

- Exemplos de operações com 8 bits:

**ADD AL,BL        ;AL = FFh e BL = 01h**

		repres. não-sinalizada	repres. sinalizada
FFh	11111111b	255	-1
01h	<u>+ 00000001b</u>	<u>+ 1</u>	<u>+1</u>
	1 00000000b	→ 256	0

**Logo após a execução da instrução:**

**CF = 1 - existência de vai 1 no MSB**

**OF = 0 , pois no MSB o "vem-um" é igual ao "vai-um" (ambos 1).**

# ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

## Introdução à linguagem assembly do x86 - Sintaxe – continuação

- **ADD AL,BL** ;ambos AL e BL contém 7Fh

		repres. não-sinalizada	repres. sinalizada
7Fh	0111 1111b	127	+ 127
7Fh	<u>+ 0111 1111b</u>	<u>+ 127</u>	<u>+ 127</u>
	0 1111 1110b →	254	+ 254

Logo após a execução da instrução:

**CF = 0**

**OF = 1** , pois no MSB o "vem-um" = 1 é diferente do "vai-um" = 0.

**Portanto:**

representação não-sinalizada	--> Flag CF indica <i>overflow</i> ;
representação sinalizada	--> Flag OF indica <i>overflow</i> .

# ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

---

## Introdução à linguagem assembly do x86 - Sintaxe – continuação

- **Como as instruções afetam os Flags**
  - **Algumas instruções, imediatamente após a sua execução:**
    - afetam todos os Flags;
    - afetam apenas alguns;
    - não afetam nenhum.

Instrução	Flags afetados
MOV	nenhum
XCHG	nenhum
LEA	nenhum
ADD/SUB	todos
INC/DEC	todos, exceto CF que não é afetado
NEG	todos, CF=1 se o resultado não for zero

# ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

---

Introdução à linguagem assembly do x86 - Sintaxe – continuação

Exemplos:

1) **ADD AX,BX** ;onde ambos AX e BX valem 0FFFFh

**CF, OF, SF, PF, AF, ZF ?**

2) **INC AL** ;onde AL contem 0FFh

**CF, OF, SF, PF, AF, ZF ?**



# ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

---

**Exemplos:**

**ADD AX,BX            ;onde ambos AX e BX valem FFFFh**

**FFFFh            1111 1111 1111 1111 b**

**FFFFh            +    1111 1111 1111 1111 b**

**FFFEh            1 1111 1111 1111 1110 b**

**Como resultado:    CF = 1            AF = 1            ZF = 0**  
**PF = 0            SF = 1            OF = 0**

**INC AL                ;onde AL contem FFh**

**FFh                1111 1111 b**

**01h                + 0000 0001 b**

**100h               1 0000 0000 b**

**Como resultado:    CF = não afetado    AF = 1            ZF = 1**  
**PF = 1                SF = 0            OF = 0**

# ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

---

## Introdução à linguagem assembly do x86 - Sintaxe – continuação

**Acesso ao TD:**

**C:\ tasm nome\_arquivo.asm /zi → montagem**

**C:\ tlink nome\_arquivo.obj /v → linkagem**

**C:\ nome\_arquivo.exe → execução**

**C:\ td nome\_arquivo.exe → debugger**

# ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

---

## Introdução à linguagem assembly do x86 - Sintaxe – continuação

- O programa DEBUG

O programa TD depuração de programas em Linguagem Montadora e permite acompanhar a modificação do conteúdo de registradores (inclusive o de Flags).

Escrevendo um programa de teste e verificação dos Flags:

```
TITLE PROGRAMA PARA VERIFICACAO DOS FLAGS
;
;usado no DEBUG para verificar o registradores de Flags
;
.MODEL SMALL
.STACK 100H
.CODE
    MOV AX,4000H      ;AX = 4000h - valor inicial de AX
    ADD AX,AX         ;AX = 8000h (4000h + 4000h = 8000h)
    SUB AX,0FFFFH     ;AX = 8001h (8000h - FFFFh = 8001h)
    NEG AX            ;AX = 7FFFh (C2 de 8001h)
    INC AX            ;AX = 8000h (7FFFh + 0001h = 8000h)
    MOV AH,4CH
    INT 21H           ;saida para o DOS
    END
```

# ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

## Introdução à linguagem assembly do x86 - Sintaxe – continuação

### Simbologia usada para os Flags no Programa Debug

	Símbolo quando 1	Símbolo quando 0
<b>Flag de Estado</b>		
CF	CY (carry)	NC (no carry)
PF	PE (parity even - PAR)	PO (parity odd - IMPAR)
AF	AC (auxiliary carry)	NA (no aux. carry)
ZF	ZR (zero)	NZ (no zero)
SF	NG (negativo)	PL (plus - positivo)
OF	OV (overflow)	NV (no overflow)
<b>Flag de Controle</b>		
DF	DN (down - para baixo)	UP (up - para cima)
IF	EI (permite interrupção)	DI (desabilita interup.)

# EXEMPLO INTERRUPTÕES

```
ADD AX,BX  
MOV AX, 0  
JZ SALTA
```

```
SUB AL,BL ; AL = 5 E BL = -1  
ZF = 0  
CF = 0  
SF = 0  
PF = 0  
AF = 0  
OF = 0
```

INSTRUÇÃO QUE ALTERA\* FLAGS  
JGE SALTA

ZF = 1 ou  
ZF = 0 e SF = 0

\* INSTRUÇÃO ARITMÉTICA OU LÓGICAS

```
CMP AL,BL ; AL = 4 E BL = 7  
JGE SALTA
```

ZF = 0  
SF = 1

NÃO SALTA