Arhitectura Sistemelor de Calcul

Lect. Dr. Șotropa Diana
diana.sotropa@ubbcluj.ro

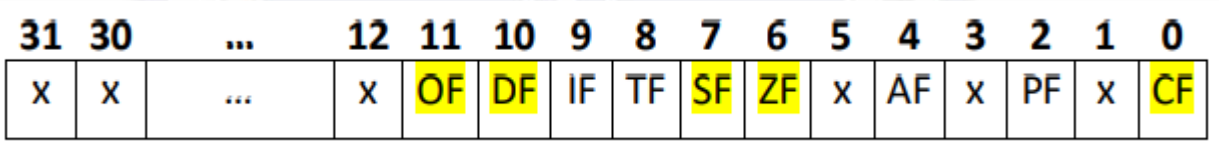
Facultatea de Matematică și Informatică
Universitatea Babeș-Bolyai


FACULTATEA DE MATEMATICA SI INFORMATICA
ROMÂNIA
BABEŞ-BOLYAI

FLAG-urile

Flag-urile

- Un flag este un indicator reprezentat pe un bit.
- O configuraţie a registrului de flaguri indică un rezumat sintetic a execuţiei fiecărei instrucţiuni.
- Pentru x86 registrul EFLAGS *(the status register)* are 32 biţi dintre care sunt folosiţi uzual numai 9.



31	30	...	12	11	10	9	8	7	6	5	4	3	2	1	0
x	x	...	x	OF	DF	IF	TF	SF	ZF	x	AF	x	PF	x	CF

Flagurile

$$\begin{array}{r} 1001\ 0011b + \\ 0111\ 0011b \\ \hline 1\ 0000\ 0110b \end{array}$$

- **CF** *(Carry Flag)* - flagul de transport
  - are valoarea 1 în cazul în care în cadrul ultimei operatii efectuate (UOE) s-a efectuat transport în afara domeniului de reprezentare a rezultatului si valoarea 0 in caz contrar

Adunare:

```
add regd, regs; regd ← regd + regs
add reg, mem; reg ← reg + mem
add mem, reg; mem ← mem + reg
add reg/mem, con; reg/mem ← reg/mem + con
```

$$\begin{array}{r} 1001\ 0011b + \\ 0111\ 0011b \\ \hline 1\ 0000\ 0110b \end{array}$$

Exemplu:

```
MOV AL, 10010011b
ADD AL, 01110011b; AL = 00000110b, CF = 1
```

Flagurile

31	30	...	12	11	10	9	8	7	6	5	4	3	2	1	0
x	x	...	x	OF	DF	IF	TF	SF	ZF	x	AF	x	PF	x	CF

- **CF** *(Carry Flag)* - flagul de transport
  - are valoarea 1 în cazul în care în cadrul ultimei operatii efectuate (UOE) s-a efectuat transport în afara domeniului de reprezentare a rezultatului si valoarea 0 in caz contrar

$1001\ 0011b +$	$147 +$	$93h +$
$0111\ 0011b$	115	$73h$
$\mathbf{1}\ 0000\ 0110b$	262	$106h$
rezultă un transport de cifră seminificativă și valoarea 1 este depusă automat în CF	**(fără semn)**	**(hexa)**

Flagul CF semnalează depăşirea în cazul interpretării FĂRĂ SEMN

Flagurile

31	30	...	12	11	10	9	8	7	6	5	4	3	2	1	0
x	x	...	x	OF	DF	IF	TF	SF	ZF	x	AF	x	PF	x	CF

- **PF** *(Parity Flag)*
  - valoarea lui se stabileşte a.î. împreună cu numărul de biţi 1 din octetul cel mai puțin semnificativ al reprezentarii rezultatului UOE să rezulte un număr impar de cifre 1

Scădere:

```
sub regd, regs; regd ← regd - regs
sub reg, mem; reg ← reg - mem
sub mem, reg; mem ← mem - reg
sub reg/mem, con; reg/mem ← reg/mem - con
```

Exemplu:

```
MOV AL, 10001100b
ADD AL, 00000010b; AL = 10001110b, PF = 1
SUB AL, 10000000b; AL = 00001110b, PF = 0
```

Flagurile

31	30	...	12	11	10	9	8	7	6	5	4	3	2	1	0
x	x	...	x	OF	DF	IF	TF	SF	ZF	x	AF	x	PF	x	CF

- **AF** *(Auxiliary Flag)*
  - indică valoarea transportului de la bitul 3 la bitul 4 al rezultatului UOE
  - de exemplu, în adunarea de mai jos transportul este 0

Exemplu:

```
MOV AL, 10010011b
ADD AL, 01110011b; AL = 00000110b, AF = 0
```

$$\begin{array}{r} 93h + \\ 73h \\ \hline 106h \end{array}$$

(hexa)

Flagurile

31	30	...	12	11	10	9	8	7	6	5	4	3	2	1	0
x	x	...	x	OF	DF	IF	TF	SF	ZF	x	AF	x	PF	x	CF

- **ZF** *(Zero Flag)*
  - primeşte valoarea 1 dacă rezultatul UOE este egal cu zero şi valoarea 0 la rezultat diferit de zero.

Incrementare / Decrementare:

```
INC reg/mem; reg/mem = reg/mem + 1
DEC reg/mem; reg/mem = reg/mem - 1
```

Exemplul 1:

```
MOV ECX, 1
SUB ECX, 1; ECX = 0, ZF = 1
```

Exemplul 2:

```
MOV EAX, 0FFFFFFFFh
INC EAX ; EAX = 0, ZF = 1
INC EAX ; EAX = 1, ZF = 0
DEC EAX ; EAX = 0, ZF = 1
```

Flagurile

31	30	...	12	11	10	9	8	7	6	5	4	3	2	1	0
x	x	...	x	OF	DF	IF	TF	SF	ZF	x	AF	x	PF	x	CF

- **SF** *(Sign Flag)*
  - primeşte valoarea 1 dacă rezultatul UOE este un număr strict negativ şi valoarea 0 în caz contrar.

Exemplu:

```
MOV BL, 1; BL = 01h
SUB BL, 2; BL = -1 = FFh, SF = 1
```

Flagul SF semnalează care e semnul numărului în cazul interpretării CU SEMN

Flagurile

31	30	...	12	11	10	9	8	7	6	5	4	3	2	1	0
x	x	...	x	OF	DF	IF	TF	SF	ZF	x	AF	x	PF	x	CF

- **TF** *(Trap Flag)* - flag de depanare
  - dacă are valoarea 1, atunci maşina se opreşte după fiecare instrucţiune
- **IF** *(Interrupt Flag)* - flag de întrerupere
- **DF** *(Direction Flag)*
  - pt operare asupra şirurilor de octeţi sau de cuvinte.
  - dacă are valoarea 0, atunci deplasarea în şir se face de la început spre sfârşit, iar dacă are valoarea 1 este vorba de deplasări de la sfârşit spre început.

Cât e TF, IF, DF în urma execuției unei adunări?

Flagurile

31	30	...	12	11	10	9	8	7	6	5	4	3	2	1	0
x	x	...	x	OF	DF	IF	TF	SF	ZF	x	AF	x	PF	x	CF

- **OF** *(Overflow Flag)* - flag pentru depăşire CU SEMN
  - dacă rezultatul ultimei instrucţiuni în interpretarea CU SEMN a operanzilor nu a încăput în spaţiul rezervat operanzilor (intervalul de reprezentare admisibil), atunci acest flag va avea valoarea 1, altfel va avea valoarea 0.

Exemplu:

```
MOV AL, -109; AL = 10010011b
ADD AL, 115; AL = 06 = 00000110b, OF = 0
```

Flagurile

31	30	...	12	11	10	9	8	7	6	5	4	3	2	1	0
x	x	...	x	OF	DF	IF	TF	SF	ZF	x	AF	x	PF	x	CF

- **OF** *(Overflow Flag)* - flag pentru depăşire CU SEMN
  - dacă rezultatul ultimei instrucţiuni în interpretarea CU SEMN a operanzilor nu a încăput în spaţiul rezervat operanzilor (intervalul de reprezentare admisibil), atunci acest flag va avea valoarea 1, altfel va avea valoarea 0.

$$\begin{array}{r} 1001\ 0011b + \\ 0111\ 0011b \\ \hline 1\ 0000\ 0110b \end{array}$$

$$\begin{array}{r} -109 + \\ 115 \\ \hline 06 \end{array}$$

(cu semn)

$$\begin{array}{r} 93h + \\ 73h \\ \hline 106h \end{array}$$

(hexa)

Nu rezultă un transport de cifră seminificativă deci OF = 0

Flagul OF semnalează depăşirea în cazul interpretării CU SEMN

Flagurile

31	30	...	12	11	10	9	8	7	6	5	4	3	2	1	0
x	x	...	x	OF	DF	IF	TF	SF	ZF	x	AF	x	PF	x	CF

- **OF** *(Overflow Flag)* - flag pentru depăşire CU SEMN
  - dacă rezultatul ultimei instrucţiuni în interpretarea CU SEMN a operanzilor nu a încăput în spaţiul rezervat operanzilor (intervalul de reprezentare admisibil), atunci acest flag va avea valoarea 1, altfel va avea valoarea 0.

$$\begin{array}{r} 1001\ 0011b + \\ 0111\ 0011b \\ \hline 1\ 0000\ 0110b \end{array}$$

$$\begin{array}{r} -109 + \\ 115 \\ \hline 06 \end{array}$$

(cu semn)

Nu rezultă un transport de cifră seminificativă deci OF = 0

$$\begin{array}{r} 93h + \\ 73h \\ \hline 106h \end{array}$$

(hexa)

$$\begin{array}{r} 147 + \\ 115 \\ \hline 262 \end{array}$$

(fără semn)

rezultă un transport de cifră seminificativă deci CF = 1

6∈ [-128, 127] => OF = 0

REPREZENTARE vs INTERPRETARE

Categorii de flag-uri

a) flag-uri ale căror valori sunt setate ca efect direct al execuţiei Ultimei Operaţii Efectuate (UOE) : **CF, PF, AF, ZF, SF şi OF**
(unde si CUM se tine cont in program de aceste valori ?)
ADC, salturi CONDITIONATE (cea mai frecventa utilizare)

Adunare cu transport:

```
ADC regd, regs; regd ← regd + regs + CF
ADC reg, mem; reg ← reg + mem + CF
ADC mem, reg; mem ← mem + reg + CF
ADC reg/mem, con; reg/mem ← reg/mem + con + CF
```

Adunare de quadwords: EDX:EAX + ECX:EBX

```
ADD EAX, EBX; EAX = dublucuvant low
ADC EDX, ECX; EDX = dublucuvant high
```

Categorii de flag-uri

a) flag-uri ale căror valori sunt setate ca efect direct al execuţiei Ultimei Operaţii Efectuate (UOE) : **CF, PF, AF, ZF, SF şi OF**
(unde si CUM se tine cont in program de aceste valori ?)
ADC, SBB, salturi CONDITIONATE (cea mai frecventa utilizare)

Scadere cu imprumut:

```
SBB regd, regs; regd ← regd - regs - CF
SBB reg, mem; reg ← reg - mem - CF
SBB mem, reg; mem ← mem - reg - CF
SBB reg/mem, con; reg/mem ← reg/mem - con - CF
```

Scadere de quadwords: EDX:EAX - ECX:EBX

```
SUB EAX, EBX; EAX = dublucuvant low
SBB EDX, ECX; EDX = dublucuvant high
```

Categorii de flag-uri

b) flag-uri ale căror valori sunt setate de către programator în vederea influenţării modului de operare al instrucţiunilor care urmează acestor setări : **CF, TF, DF şi IF**
*(CUM le setam => **prin instructiuni specifice**)*

CLC – efectul fiind **CF=0**
STC – efectul fiind **CF=1**

CMC – complementarea valorii din CF

CLD – avand ca efect **DF=0**
STD – avand ca efect **DF=1**

CLI – avand ca efect **IF=0**
STI – avand ca efect **IF=1**

Având în vedere riscul major de setare accidentală a valorii din TF precum şi rolul său absolut special în dezvoltarea de depanatoare, NU există disponibile instrucţiuni de acces direct la valoarea din TF !!

Operații efectuate de ALU (Arithmetic and Logic Unit)

- Operații aritmetice
  - Adunări și scăderi bit cu bit
  - Înmulțirea și împărțirea sunt operații mai costisitoare, și pot fi înlocuite cu adunări / scăderi repetate
- Operații de shiftare pe biți
  - Implică shiftarea locației unui bit spre stânga sau dreapta cu un anumit număr de poziții
  - Se folosesc pentru operații de înmulțire și împărțire
- Operații logice
  - AND, OR, XOR, NOT

Operații efectuate de ALU (Arithmetic and Logic Unit)

Instrucțiunea MOVZX (move with zero-extend):

```
MOVZX reg32, reg/mem8
MOVZX reg32, reg/mem16
MOVZX reg16, reg/mem8
```

Exemple

```
byteVal db 10001111b

MOVZX AX, [byteVal]; AX = 00000000 10001111b

MOV BX, 0A69Bh; BH = A6h și BL = 9Bh
MOVZX EAX, BX; EAX = 0000A69Bh
MOVZX EDX, BL; EDX = 0000009Bh
MOVZX CX, BL; CX = 009Bh
```

Operații efectuate de ALU (Arithmetic and Logic Unit)

Instrucțiunea MOVSX (move with sign-extend):

```
MOVSX reg32, reg/mem8
MOVSX reg32, reg/mem16
MOVSX reg16, reg/mem8
```

Exemple

```
byteVal db 10001111b

MOVSX AX, [byteVal]; AX = 11111111 10001111b

MOV BX, 0F69Bh; BH = F6h și BL = 9Bh
MOVSX EAX, BX; EAX = FFFFA69Bh
MOVSX EDX, BL; EDX = FFFFFF9Bh
MOVSX CX, BL; CX = FF9Bh
```

Operații efectuate de ALU (Arithmetic and Logic Unit)

Instrucțiunea XCHG (exchange data):

```
XCHG reg, reg
XCHG reg, mem
XCHG mem, reg
```

Exemple

```
XCHG AX, BX; interschimbă valorile regiștrilor pe 16 biți
XCHG AH, AL; interschimbă valorile regiștrilor pe 8 biți
XCHG [var1], BX
XCHG EAX, EBX

MOV AX, [val1]
XCHG AX, [val2]
MOV [val1], AX; interschimbă două valori din memorie
```

Operații efectuate de ALU (Arithmetic and Logic Unit)

Instrucțiunea MUL (înmulțire în interpretarea fără semn):

```
MUL op

op = reg/mem8 => MUL reg/mem8 => AL*reg/mem8 = AX
op = reg/mem16 => MUL reg/mem16 => AX*reg/mem16 = DX:AX
op = reg/mem32 => MUL reg/mem32 => EAX*reg/mem32 = EDX:EAX
```

Exemplul 1

```
; 3*4
MOV AL, 3
MOV BL, 4
MUL BL ; AL*BL=AX
```

Exemplul 2

```
; c*4, c - word
MOV AX, 4
MUL word [c]
; AX*[c]=DX:AX
```

Exemplul 3

```
; a*b, a,b - doubleword
MOV EAX, [a]
MUL dword [b]
; EAX*[b]=EDX:EAX
```

Operații efectuate de ALU (Arithmetic and Logic Unit)

Instrucțiunea DIV (împărțire în interpretarea fără semn):

```
DIV op

op = reg/mem8 => DIV reg/mem8 => AX / reg/mem8 = AL rest AH
op = reg/mem16 => DIV reg/mem16 => DX:AX / reg/mem16 = AX rest DX
op = reg/mem32 => DIV reg/mem32 => EDX:EAX / reg/mem32 = EAX rest EDX
```

Exemplul 1

```
; m/n, m – word, n - byte
MOV AX, [m]
DIV byte [n]
; AX / [n] = AL (cat)
; AX % [n] = AH (rest)
```

Exemplul 2

```
; 5 / r, r – word
MOV AX, 5
MOV DX, 0
DIV word [r]
; DX:AX / [r] = AX
; DX:AX % [r] = DX
```

Exemplul 3

```
; EDX:EAX / 12345678h
MOV ECX, 12345678h
DIV ECX
; EDX:EAX / ECX = EAX
; EDX:EAX % ECX = EDX
```

Operații efectuate de ALU (Arithmetic and Logic Unit)

Instrucțiunea IMUL (înmulțire în interpretarea cu semn):

```
IMUL op

op = reg/mem8 => IMUL reg/mem8 => AL*reg/mem8 = AX
op = reg/mem16 => IMUL reg/mem16 => AX*reg/mem16 = DX:AX
op = reg/mem32 => IMUL reg/mem32 => EAX*reg/mem32 = EDX:EAX
```

Exemplul 1

```
; 3*(-4)
MOV AL, 3
MOV BL, -4
IMUL BL ; AL*BL=AX
```

Exemplul 2

```
; c*(-2), c - word
MOV AX, -2
IMUL word [c]
; AX*[c]=DX:AX
```

Exemplul 3

```
; a*b, a,b - doubleword
MOV EAX, [a]
IMUL dword [b]
; EAX*[b]=EDX:EAX
```

Operații efectuate de ALU (Arithmetic and Logic Unit)

Instrucțiunea IDIV (împărțire în interpretarea fără semn):

```
IDIV op

op = reg/mem8 => IDIV reg/mem8 => AX / reg/mem8 = AL rest AH
op = reg/mem16 => IDIV reg/mem16 => DX:AX / reg/mem16 = AX rest DX
op = reg/mem32 => IDIV reg/mem32 => EDX:EAX / reg/mem32 = EAX rest EDX
```

Exemplul 1

```
; m/n, m – word, n - byte
MOV AX, [m]
IDIV byte [n]
; AX / [n] = AL (cat)
; AX % [n] = AH (rest)
```

Exemplul 2

```
; (-5) / r, r – word
MOV AX, -5
MOV DX, 0
IDIV word [r]
; DX:AX / [r] = AX
; DX:AX % [r] = DX
```

Exemplul 3

```
; EDX:EAX / 0A2345678h
MOV ECX, 0A2345678h
IDIV ECX
; EDX:EAX / ECX = EAX
; EDX:EAX % ECX = EDX
```

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
UNIVERSITATEA BABEȘ-BOLYAI

Str. Mihail Kogălniceanu nr. 1
Cluj-Napoca, Cluj, România

www.cs.ubbcluj.ro