

# Organisasi Sistem Komputer

## Bab 4. Operasi Aritmatika dan Perbandingan

### 4.1 Instruksi Penjumlahan

### 4.2 Instruksi Perkalian

### 4.3 Instruksi Pembagian



# Pembahasan:

- Instruksi INC dan DEC
- Instruksi NEG
- Instruksi ADD dan SUB

# Instruksi INC dan DEC

- Sintaks:

INC *destination*

DEC *destination*

- Instruksi INC (inkrementasi) menambah nilai 1 ke operand destination
- Instruksi DEC (dekrementasi) mengurangi nilai 1 dari operand destination
- Operand *destination* bisa berupa **register** atau **memori**
- Contoh:

```
segment .data  
myWord DW 1000h
```

```
segment .text  
inc word [myWord]  
mov bx, myWord  
dec bx
```

myWord = 1001h

BX = 1000h



# Instruksi NEG

- Sintaks:

`NEG destination`

- Instruksi `NEG` (negasi) mengkonversi nilai operand `destination` ke komplemen dua-nya
- Operand `destination` bisa berupa *register* atau *memori*
- Contoh:

```
segment .text
```

```
mov     eax, 23d
```

```
neg     eax
```

EAX = 00000017h

EAX = FFFFFFFE9h

# Instruksi ADD dan SUB

- Sintaks:

ADD      *dst, src*

SUB      *dst, src*

- Instruksi ADD menjumlahkan operand *dst* ke operand *src* dan menyimpan hasilnya di operand *dst* ( $dst = dst + src$ )
- Instruksi SUB mengurangi operand *dst* dengan operand *src* dan menyimpan hasilnya di operand *dst* ( $dst = dst - src$ )
- Kedua operand harus mempunyai ukuran sama dan dapat berupa **register**, **memori**, atau **immediate value** (namun keduanya tidak dapat berupa memori)
- Instruksi ADD dan SUB mengubah flag Carry, Overflow, Zero, atau Sign berdasarkan hasilnya

# Contoh ADD dan SUB

- Misalkan kita ingin menghitung ekspresi berikut:

$$Rval = -Xval + (Yval - Zval)$$

```
segment .data
    Xval    DD    26
    Yval    DD    30
    Zval    DD    40

segment .bss
    Rval    RESD    1
```

Suku pertama  $-Xval$ : pindahkan nilai  $Xval$  ke EAX lalu negasi

Suku kedua  $(Yval - Zval)$ : pindahkan nilai  $Yval$  ke EBX lalu kurangi dengan  $Zval$

Tambahkan kedua suku, lalu simpan di  $Rval$

```
segment .text

; suku pertama: -Xval
mov     eax, [Xval]
neg     eax                ; EAX = -26

; suku kedua: (Yval - Zval)
mov     ebx, [Yval]
sub     ebx, [Zval]        ; EBX = -10

; tambahkan kedua suku dan simpan:
add     eax, ebx
mov     [Rval], eax        ; EAX = -36
```

# ADD dan SUB pada Bilangan Bertanda

- Instruksi ADD dan SUB dapat digunakan untuk pasangan bilangan bertanda dan pasangan bilangan tidak bertanda
- Contoh:

mov al, 0A3h = 163

add al, 17h = 23

$$\begin{array}{r} 10100011 = 163 \\ + 00010111 = 23 \\ \hline 10111010 = 186 \end{array}$$

mov bl, 0A3h = -93

add bl, 17h = +23

$$\begin{array}{r} 10100011 = -93 \\ + 00010111 = +23 \\ \hline 10111010 = -70 \end{array}$$

- Pada contoh di atas, CPU menjumlahkan dua bilangan biner yang sama dan hasilnya valid jika kita mengartikannya konsisten (penjumlahan bil. tidak bertanda menghasilkan nilai tidak bertanda dan penjumlahan bertanda menghasilkan nilai bertanda)

# Flag yang Dipengaruhi ADD dan SUB

- Instruksi `ADD` dan `SUB` men-set nilai-nilai bit status pada register `EFLAGS`, berdasarkan nilai dari hasil instruksi
- Bit flag status yang diubah antara lain:
  - Bit flag zero (= 1, jika hasil sama dengan 0)
  - Bit flag sign (= 1, jika hasil negatif)
  - Bit flag carry
  - Bit flag overflow
- Ketika apa bit carry dan bit overflow di-set oleh instruksi `ADD` dan `SUB`?



# Flag Carry

- Flag carry di-set, ketika operasi penjumlahan atau pengurangan **bilangan tidak bertanda** menghasilkan nilai di luar jangkauan
- Misal penjumlahan dalam 1 byte:

```
mov  al, 0FEh  = 254
add  al, 02h    = 254 + 2
```



Penjumlahan menghasilkan **carry** yang dibawa keluar kapasitas bit, maka penjumlahan ini disebut **overflow**

Flag Carry yang di-set pada operasi penjumlahan bilangan tidak bertanda mengindikasikan hasil **tidak valid**

# Flag Carry

- Misal pengurangan dalam 1 byte:

```
mov  al, 01h = 1
sub  al, 02h = 1 - 2
```



Pada bit ke-1, kita harus “*borrow*” meminjam nilai dari bit-bit di kiri. Namun karena semua bit di kiri sama dengan 0, maka kita meminjam dari bit di luar kapasitas bit (*overflow*)

Flag Carry yang di-set pada operasi pengurangan bilangan tidak bertanda mengindikasikan hasil *tidak valid*

# Flag Overflow

- Flag overflow di-set ketika operasi penjumlahan dan pengurangan **bilangan bertanda** menghasilkan nilai di luar jangkauan
- Misal pada penjumlahan dan pengurangan bil. bertanda 1 byte (-128 s.d +127):
  - Penjumlahan bil. bertanda 1 byte:

```
mov  al, +127  
add  al, +1
```



OF

1

Penjumlahan menghasilkan +128 yang diluar jangkauan bilangan bertanda, sehingga Flag Overflow di-set. Ini menandakan hasil **tidak valid**

- Pengurangan bil. bertanda 1 byte:

```
mov  al, -128  
add  al, +1
```



OF

1

Pengurangan menghasilkan -129 yang diluar jangkauan bilangan bertanda, sehingga Flag Overflow di-set. Ini menandakan hasil **tidak valid**

# Flag Carry dan Overflow

- Flag **carry** yang ter-set pada penjumlahan atau pengurangan **bilangan tidak bertanda** menandakan hasil yang **tidak valid**
- Flag **overflow** yang ter-set pada penjumlahan atau pengurangan **bilangan bertanda** menandakan hasil yang **tidak valid**

# Ringkasan

- Instruksi `INC` untuk inkrementasi dan instruksi `DEC` untuk dekrementasi
- Instruksi `NEG` untuk negasi (mendapatkan nilai komplemen dua)
- Instruksi `ADD` dan `SUB` untuk penjumlahan dan pengurangan, dan men-set flag:
  - Flag carry di-set jika hasil `ADD` dan `SUB` bilangan tidak bertanda tidak valid
  - Flag overflow di-set jika hasil `ADD` dan `SUB` bilangan bertanda tidak valid