Organisasi Sistem Komputer

Bab 4. Operasi Aritmatika

4.1 Instruksi Penjumlahan

4.2 Instruksi Perkalian

4.3 Instruksi Pembagian

Pembahasan:

- Instruksi `MUL` dan `IMUL`

Perkalian

- Terdapat dua instruksi untuk perkalian:
  - Perkalian bilangan tidak bertanda: `MUL`
  - Perkalian bilangan bertanda: `IMUL`
- Kenapa kita membutuhkan dua instruksi yang berbeda?
  - Misalkan mengalikan `FF` dengan `FF`
    - Jika kita mengasumsikan bilangan tidak bertanda, maka `FF` x `FF` berarti $255 \times 255 = 65035 =$ `FE0Bh`
    - Jika kita mengasumsikan bilangan bertanda, maka `FF` x `FF` berarti $-1 \times -1 = 1 =$ `0001h`

Instruksi MUL

- Sintaks:

```
MUL     src
```

- Instruksi `MUL` digunakan untuk perkalian bilangan tidak bertanda
- Hanya memerlukan satu operand sebagi pengali, nilai terkali diasumsikan berada dalam register `AL`, `AX`, dan `EAX` tergantung dari besar operand src:

src (pengali)	Terkali	Aksi
reg/mem8	AL	AX = AL x src
reg/mem16	AX	DX:AX = AX x src
reg/mem32	EAX	EDX:EAX = EAX x src

mengalikan register 8 bit atau memori 8 bit dengan `AL` dan hasilnya disimpan di `AX`

mengalikan register 16 bit atau memori 16 bit dengan `AX` dan hasilnya disimpan di `DX:AX` (16 bit teratas di `DX` dan 16 bit terbawah di `AX`)

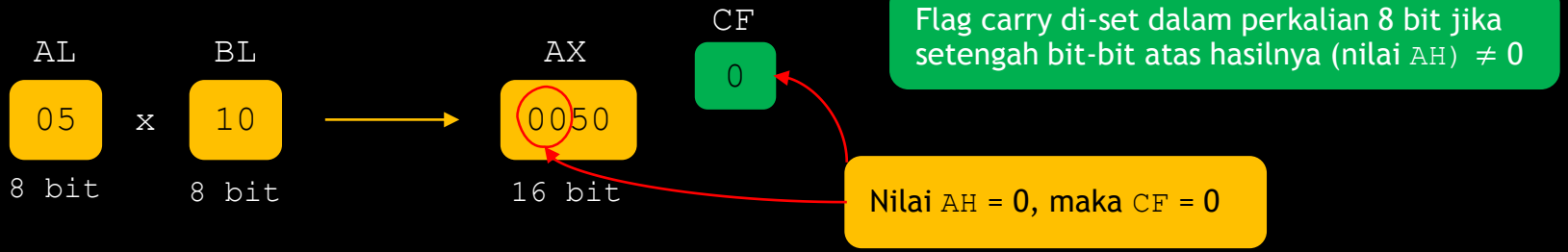
mengalikan register 32 bit atau memori 32 bit dengan `AX` dan hasilnya disimpan di `EDX:EAX` (32 bit teratas di `EDX` dan 32 bit terbawah di `EAX`)

Flag Carry pada Instruksi MUL

- Instruksi `MUL` men-set flag Carry jika setengah atas bit-bitnya tidak sama dengan 0
- Contoh: *Perkalian 8 bit*

```
mov    al, 5h      ; simpan nilai terkali 5h ke al
mov    bl, 10h     ; simpan nilai pengali 10h ke bl
mul    bl          ; ax = al x bl = 5h x 10h = 50h
```

src (pengali)	Terkali	Aksi
reg/mem8	AL	AX = AL x src
reg/mem16	AX	DX:AX = AX x src
reg/mem32	EAX	EDX:EAX = EAX x src



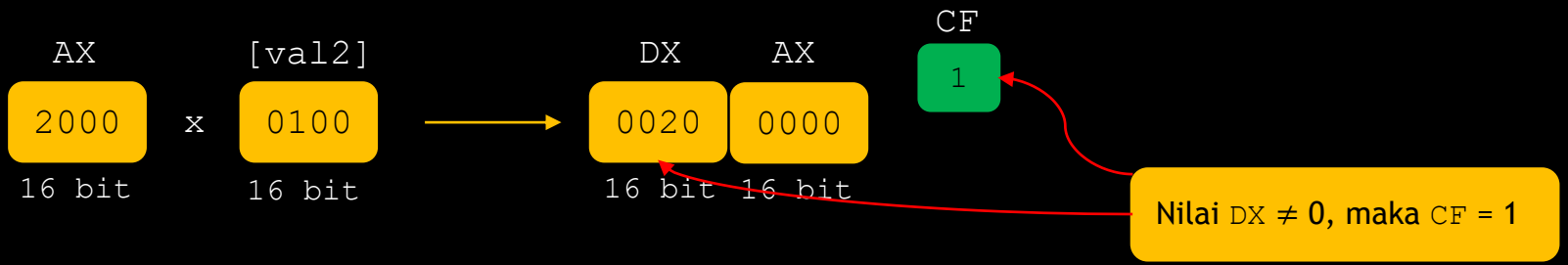
Contoh Instruksi MUL 16-bit

- Contoh: *Perkalian 16 bit*

src (pengali)	Terkali	Aksi
reg/mem8	AL	AX = AL x src
reg/mem16	AX	DX:AX = AX x src
reg/mem32	EAX	EDX:EAX = EAX x src

```
segment .data
    val1     DW       2000h
    val2     DW       0100h

segment .text
    mov    ax, [val1]   ; simpan nilai terkali 2000h ke ax
    mul    [val2]       ; DX:AX = AX x [val1] = 2000h x 0100h = 00200000h
                        ; DX = 0020, AX = 0000
```

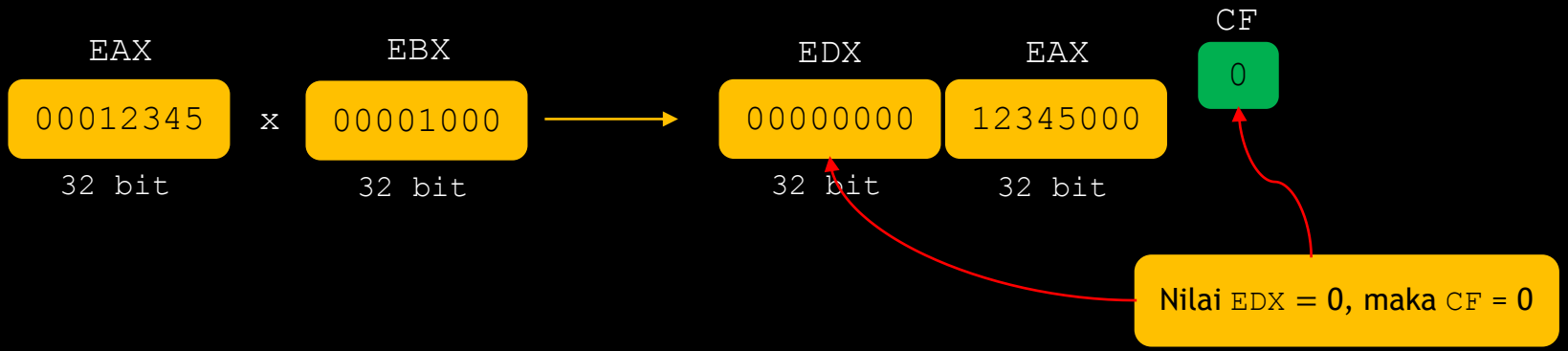


Contoh Instruksi MUL 32-bit

- Contoh: *Perkalian 32 bit*

src (pengali)	Terkali	Aksi
reg/mem8	AL	AX = AL x src
reg/mem16	AX	DX:AX = AX x src
reg/mem32	EAX	EDX:EAX = EAX x src

```
mov    eax, 12345h   ; simpan nilai terkali 12345h ke eax
mov    ebx, 10000h   ; simpan nilai pengali 10000h ke ebx
mul    ebx           ; edx:eax = eax x ebx = 00012345h x 00001000h
                     ; edx = 00000000, eax = 123450000
```



Instruksi IMUL

- Instruksi `IMUL` mempunyai tiga varian:

  `IMUL` *src1*

  `IMUL` *dest, src1*

  `IMUL` *dest, src1, src2*

- Flag Overflow di-set:
  - Pada varian dengan 1 operand: jika setengah bit-bit teratas dari hasil perkalian bukan ektensi tanda (tidak mungkin terjadi overflow)
  - Pada varian dengan 2 dan 3 operand: jika hasil perkalian melebihi kapasitas bit penyimpanan hasil (overflow, sehingga nilai hasil perkalian tidak valid)

Catatan: Pada instruksi `MUL` dan `IMUL` satu operand, hasil perkalian tidak mungkin overflow karena hasil perkalian disimpan dalam register dengan kapasitas dua kali lebih besar dari pengali dan terkali

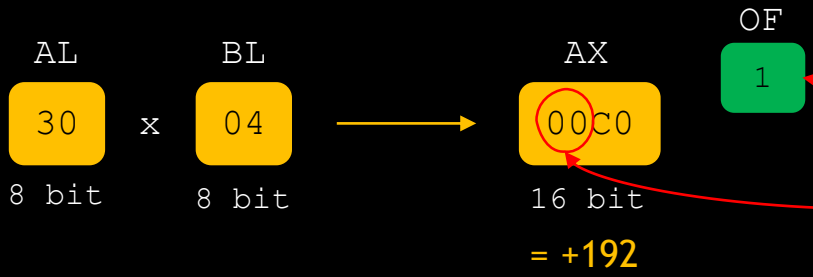
dest	src1	src2	Aksi
	reg/mem8		AX = AL x src1
	reg/mem16		DX:AX = AX x src1
	reg/mem32		EDX:EAX = EAX x src1
reg16	reg/mem16		dest = dest x src1
reg32	reg/mem32		dest = dest x src1
reg16	imm8		dest = dest x imm8
reg32	imm8		dest = dest x imm8
reg16	imm16		dest = dest x imm16
reg32	imm32		dest = dest x imm32
reg16	reg/mem16	imm8	dest = src1 x src2
reg32	reg/mem32	imm8	dest = src1 x src2
reg16	reg/mem16	imm16	dest = src1 x src2
reg32	reg/mem32	imm32	dest = src1 x src2

Contoh Instruksi IMUL

- Contoh #1: +48 × +4

```
mov    al, +48
mov    bl, +4
imul   bl          ; AX = 00C0h, OF = 1
```

dest	src1	src2	Aksi
	reg/mem8		AX = AL x src1
	reg/mem16		DX:AX = AX x src1
	reg/mem32		EDX:EAX = EAX x src1



Flag overflow yang di-set pada IMUL dengan 1 operand tidak menandakan hasil yang tidak valid, namun hanya menandakan jika setengah bit-bit atas dari hasilnya (nilai AH) bukan ekstensi tanda. Ini berarti kita dapat mengabaikan nilai dalam AH

Nilai AH = 0 yang berarti bukan berupa ekstensi tanda, maka OF = 1

Contoh Instruksi IMUL

- Contoh #2: $-3200 \times +2$

```
mov   ax, -32000
imul  ax, 2          ; AX = FFF0h, OF = 1
```

dest	src1	src2	Aksi
reg16	reg/mem16		dest = dest x src1
reg32	reg/mem32		dest = dest x src1
reg16	imm8		dest = dest x imm8
reg32	imm8		dest = dest x imm8
reg16	imm16		dest = dest x imm16
reg32	imm32		dest = dest x imm32

Flag overflow yang di-set pada `IMUL` 2 atau 3 operand berarti hasil perkalian tidak valid

Overflow terjadi karena hasil perkalian melebihi kapasitas bit

Ringkasan

- Dua instruksi perkalian:
  - `MUL` untuk perkalian bilangan tidak bertanda
  - `IMUL` untuk perklaian bilangan bertanda
- Instruksi `MUL` :
  - Hanya satu jenis instruksi
  - flag carry yang ter-set mengindikasikan nilai pada setengah bit-bit dari hasil tidak sama dengan nol, namun hasilnya selalu valid
- Instruksi `IMUL`:
  - terdapat tiga varian instruksi berdasarkan banyak operand, yaitu 1, 2, dan 3 operand
  - flag overflow yang ter-set pada `IMUL` 1 operand mengindikasikan nilai pada setengah bit-bit dari hasil bukan ekstensi tanda, namun hasilnya selalu valid
  - flag overflow yang ter-set pada `IMUL` 2 dan 3 operand mengindikasikan hasil perkalian melebih kapasitas operand destination, sehingga hasil perkalian tidak valid