Kuliah 2:

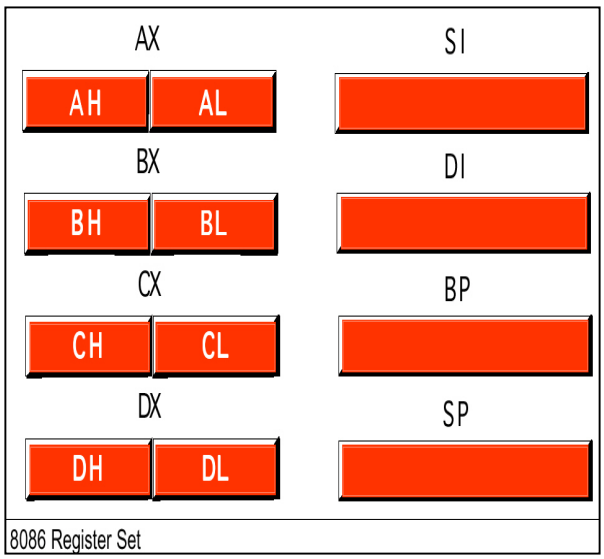
2.1. Struktur Register
2.2. Mode Pengalamatan
2.3. Set Instruksi

2.1. Struktur Register

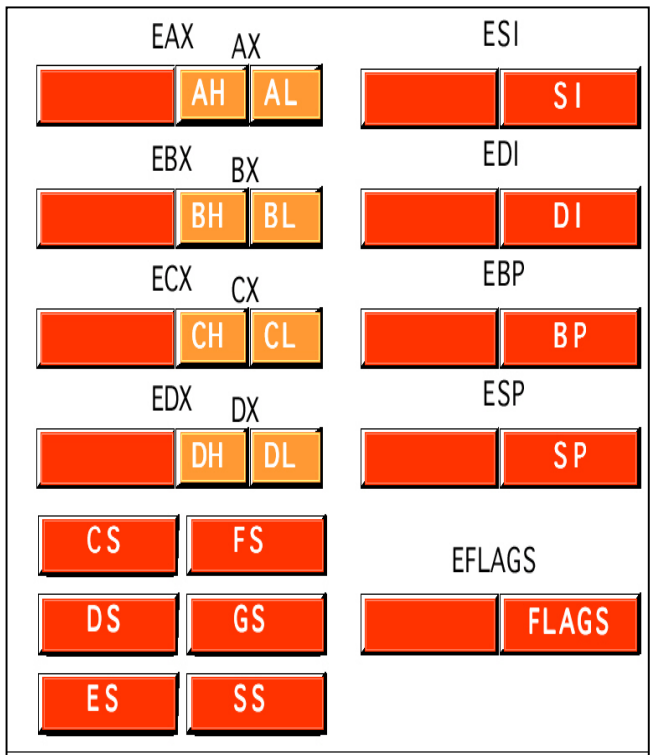
Register:

- Lokasi memori yang sangat khusus terkonstruksi dari Flip-Flop
- Didesain u/ menampung data, data tsb. dapat diakses dan diakses dalam berbagai operasi dgn kecepatan tinggi.
  - U/ prosessor 8088/8086, 80188/80186, 80286 → register 16 bit
  - U/ prosessor 80386/80486/80586/Pentium → register 32 bit
  - Optional u/ general purpose register → not independent 8 bit registers u/ High Order Byte dan Low Order Byte
- **Jenis-Jenis Register:**
  - *General-purpose registers* (*data registers*):
    - 16 bit: AX, BX, CX, DX
    - 8 bit : AH, AL, BH, BL, CH, CL, DH, DL
  - *Segment registers* : CS, DS, SS, ES
  - *Index register*: SI, DI, IP
  - *Pointer register*: IP, SP
  - *Flags registers*: Overflow, Direction, Interrupt, Trap, Sign, Zero, Auxiliary Carry, Parity, Carry
    - 16 bit, tetapi hanya 9 bit yang digunakan

2.1. Struktur Register (cont'd)



8086 Register Set



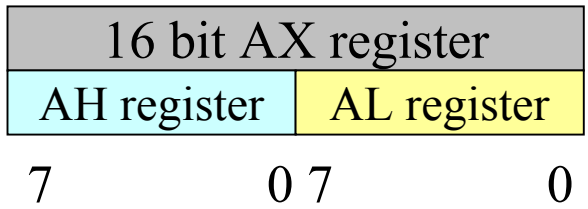
80386 Registers (Application Programmer Visible)

2.1.1 General-purpose register (data register)

- Digunakan u/ operasi aritmatik dan data movement
- Setiap register dapat dialamatkan sebagai suatu nilai 16 bit atau 8 bit
  - Contoh: AX (16 bit) → upper 8 bit AH dan lower 8 bit AL

2.1. Struktur Register (cont'd)

- Posisi bit selalu di nomori dari kiri kekanan, dimulai dgn 0:



- Instruksi dpt beralamat 16-bit atau 8-bit data registers:

AX		BX		CX		DX	
AH	AL	BH	BL	CH	CL	DH	DL

- Jika suatu 16-bit register dimodifikasi → otomatis terkait ke 8-bit-nya
  - Contoh: andaikan AX berisi 0000h. Jika suatu nilai data 126Fh di-move ke AX, maka AL beurbah menjadi 6Fh
- Atribut khusus GP-register:
  - AX (*accumulator*): operasi aritmatik, (operasi lain juga lebih effisien jk dilakukan di resgiter ini)
  - BX (*base*):
    - Operasi aritmatik dan data movement
    - Kemampuan pengalamatan khusus; yaitu berisi alamat memori yang menunjuk ke variabel lain.
    - Catatan: SI, DI, dan BP juga memiliki kemampuan khusus ini

2.1. Struktur Register (cont'd)

- CX (*counter*): counter u/ instruksi repeating atau looping
  - Instruksi ini otomatis mengulang dan menurunkan (decrement) CX dan "quit" jika CX → 0
- DX (*data*): berperan khusus dalam operasi perkalian dan pembagian. Dalam perkalian, sabgai contoh, DX berisi high 16-bits dari produk.

2.1.2. Segment Registers

- Digunakan sebagai base locations u/ intruksi-2 program, data, dan stack.
- Segment-2 register:
  - CS (*code segment*): berisi base location seluruh instruksi/kode yang dapat dieksekusi (executable) dalam suatu program
  - DS (*data segment*): default base location u/ variabel-2 memori.
    - CPU menghitung "offsets" variabel menggunakan nilai DS yang sedang berlaku (current value of DS)
  - SS (*stack segmen*): berisi base location u/ current program stack
  - ES (extra segment): additional base location u/ variabel-2 memori

2.1. Struktur Register (cont'd)

2.1.3. Indeks Registers

- Berisi offsets variabel-2
  - Offset merujuk pada jarak suatu variabel, label, atau instruksi dari base segment-nya.
- U/ speed up pemrosesan strings, arrays, dan struktur data lainnya yang mengandung multiple elements.
- Index-2 register:
  - SI (*source index*):
    - 8088's string movement instruction, dimana string sumber dituju oleh register SI biasanya mengandung suatu nilai offset dari register DS, namun ia dapat beralamat variabel manapun
  - DI (*destination index*):
    - destination u/ 8088's string movement instruction.
    - Biasanya berisi suatu offset dari register ES, namun ia dapat beralamat variabel manapun
  - BP (*base pointer*):
    - Berisi offset yang diasumsikan dari register SS → bekerja seperti stack pointer
    - Digunakan oleh suatu subroutine u/ menempatkan variabel-2 yang di-passed ke stack melalui suatu "calling" program

2.1. Struktur Register (cont'd)

2.1.4. Pointer Registers

- Antara lain:
  - IP (*instruction pointer*):
    - Berisi lokasi dari instruksi berikut yang akan diekskusi.
    - CS dan IP berkombinasi membentuk alamat instruksi berikut yang akan diekskusi
  - SP (*stack pointer*):
    - Berisi offset atau jarak (distance) dari awal stack ke top stack.
    - SS dan SP berkombinasi membentuk "complete top-of stack address"

2.1.5. Flags Regsiters

- Register 16-bit khusus dgn posisi bit tunggal yang dipekerjakan (assigned) u/ menunjukkan status CPU atau hasil-hasil operasi aritmatik.
- Dari 16 bit posisi hanya 9 bit posisi yang digunakan dan diberi nama

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	**X**	**X**	**X**	**O**	**D**	**I**	**T**	**S**	**Z**	**X**	**A**	**X**	**P**	**X**	**C**

2.1. Struktur Register (cont'd)

O = Overflow	**S = Sign**
D = Direction	**Z = Zero**
I = Interrupt	**A = Auxiliary Carry**
T = Trap	**P = Parity**
X = undefined	**C = Carry**

- Catatan: posisi diatas tidak perlu diingat, karena terdapat instruki yng didesain u/ mentes dan memanipulasi flags
- Sebuah Flags atau bit → set jika nilainya 1, clear jk nilainya 0
- CPU menset flags melalui "turning" pada bit-2 indvidual dlm register flags
- Terdapat dua tipe flags dasar:
  - Control flags
  - Status flags

Control Flags:

- Fungsinya u/ mengontrol operasi CPU (diset oleh programmer)
- Antara lain:
  - **Direction**:
  - Mengontrol arah yang digunakan pada pemrosesan string
  - Nilai : up (1) dan down (0) (diset melalui instruski STD dan CLD

2.1. Struktur Register (cont'd)

- **Interrupt flags:**
- Memungkinan external interrupt berlaku
- Disebabkan oleh devais HW misalkan keyboard, disk drive, system clock timer
- Nilai: 1 enable(EI), 0 disable (DI) (dikontrol melalui instruksi CLI dan STI)
- **Trap flags:**
- Menentukan apakah CPU perlu dihentikan setelah setiap instruksi dilaksanakan
- Digunakan pada saat Debugging program → TRACING

Status Flags:

- Merefleksikan outcome dari operasi aritmatik dan logika
- Antara lain
  - **Carry Flags:**
  - Diset jika hasil op. aritmatik terlalu besar u/ di fit-kan ke destination
  - Contoh: Jika nilai 200 dn 56 dijumlahkan, kemudian diletakkan ke suatu 8-bit destination, 256 terlalu besar sehingga CF ini harus diset → 1 (CY) atau 0 (No Carry(NC))
  - **Overflow Flags:**
  - Diset jika hasil operasi aritmatik yang bertanda terlalu besar u/ di-fitkan ke destination area
  - Nilai: 1 (OV), 0 (NV)

2.1. Struktur Register (cont'd)

- **Sign Flag**
- Diset jika hasil operasi aritmatik atau logika → negatif
- Karena bil. negatif memiliki suatu 1 pada posisi bit tertinggi, sign flag selalu merupakan suatu copy dari bit tanda destination
- Nilai: negatif (NG) dan positif (PL)
- **Zero Flag**
- Diset jika hasil operasi aritmatik atau logika → zero
- Terutama digunakan pada instruksi-2 *jump* dan *loop*, supaya memungkinkan percabangan ke suatu lokasi baru dalam suatu program yang berbasis "perbandingan antara dua nilai"
- Nilai: zero (ZR), Not Zero (NZ)
- **Auxilairy Flag**
- Diset jika suatu operasi mengakibatkan suatu Carry, dari bit 3 ke bit 4, (atau Borrow dari bit 3 ke bit 4) dari sebuah operand
- Nilai: Aux Carry (AC), No Aux Carry (NA)
- **Parity Flag**
- Merefleksi jumlah bit hasil dari operasi aritmatik yang diset.
  - Jika terdapat jumlah bit genap (even) → PE
  - Jika terdapat jumlah bits ganjil (odd) → PO

2.1. Struktur Register (cont'd)

- Digunakan
  - oleh SO u/ memverifikasi integritas memori dan
  - oleh Software komunikasi u/ memverifikasi kebenaran transmisi data

2.1.6. Perhitungan Alamat

- Prosessor 8086/8088 → kapasitas memori 1 MB = 2^{20} = 1048576 Byte

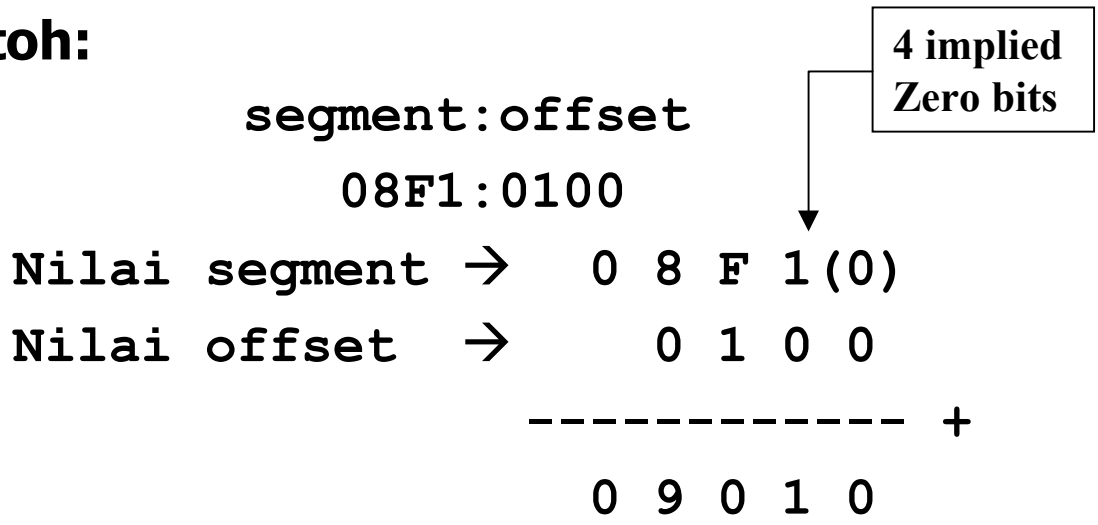
  0000 → terendah

  :

  FFFF → tertinggi
- **Alamat** merujuk pada suatu lokasi memori 8 bit !!
- Alamat dieskpresikan dalam 1 diantara 2 format hexdesimal berikut:
  - Alamat ***segment:offset*** 16-bit → kombinasi suatu base location (segment) dgn offset u/ merepresntasikan **actual location**
    - Segmet register: digit terendah (16^1) tertinggi (16^4)
    - Offset register: digit terendah (16^0) tertinggi (16^3)
  - Alamat ***absolute*** 20-bit → merujuk ke suatu lokasi memori yang eksak (alamat fisik)

2.1. Struktur Register (cont'd)

- **Catatan:**
  - digunakan ***segment:offset*** u/ mengatasi dilema: bhw CPU dapat beralamat 1048576 byte (20 bits) sedangkan register hanya 16 bits
  - Dimengert: segment register memiliki "4 implied zero bits to right
- **Contoh:**



Alamat fisik (H)	segment:offset	Alamat fisik (D)
00000	0000 : 0000	0
:	:	:
7FFFF	7000 : FFFF	524288
:	:	:
FFFFD	F000 : FFFD	1048573
:	:	:
FFFFF	F000 : FFFF	1048575

2.2. Mode Pengalamatan

- Ada beberapa cara (mode) untuk mengakses operand; Operand dapat terdapat di *register*, dalam *instruksi*, dalam *memori* atau *I/O*
- 8088 mempunyai sekitar 24 addressing mode → dikelompok menjadi 7:
  - **Register Addressing**
  - **Immediate Addressing**
  - **Direct Addressing**
  - **Register Indirect Addressing**
  - **Base Relative Addressing**
  - **Direct Indexed Addressing**
  - **Base Indexed Addressing**

2.2.1. Register dan Immediate Addressing

- Register Add.: Operand disimpan/diambil dari register

```
MOV AX,CX ;copy 16 bit isi CX (count regs)
   ke AX (acc. regs)
```

- Immediate Add.: source operand dapat berbentuk konstanta 8/16 bit; Konstanta ini terdapat dalam instruksi.

```
MOV CX,500
MOV CL,-30 ; simpan –30 ke reg. AL
```

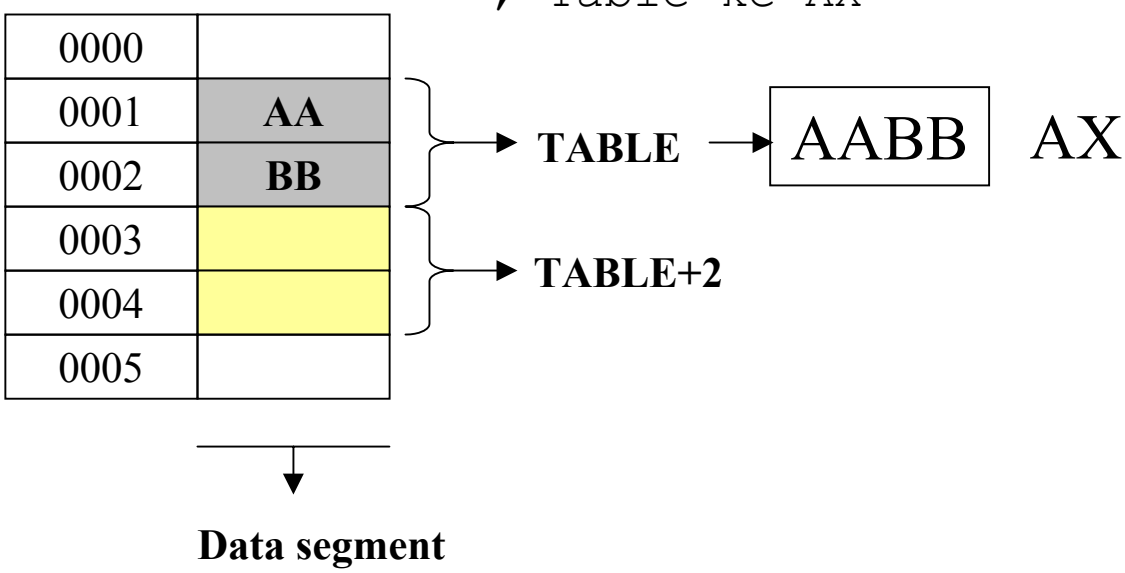
2.2. Mode Pengalamatan (cont'd)

```
K EQU 1024          ; source operand dapat
                    ; berbentuk simbol yg
                    ; didefine dgn EQU
```

- **INGAT:**
  - 8 bit signed number: 127(7H) s/d –128(80H)
  - 16 bit signed number: 32767(7FFFH) s/d –32768(8000H)
  - Max 8 bit unsigned #: 255 (0FFH)
  - Max 16 bit unsigned #: 65535 (0FFFFH)

2.2.2 Direct Addressing

```
MOV AX, TABLE       ; load isi lokasi memori
                    ; Table ke AX
```



2.2. Mode Pengalamatan (cont'd)

2.2.3. Register Indirect Addressing

- Offset: effectives address (EA) → jarak letak operand, dalam byte terhitung dari awal segment
- Address memory = offset + isi DS digeser dgn 0000

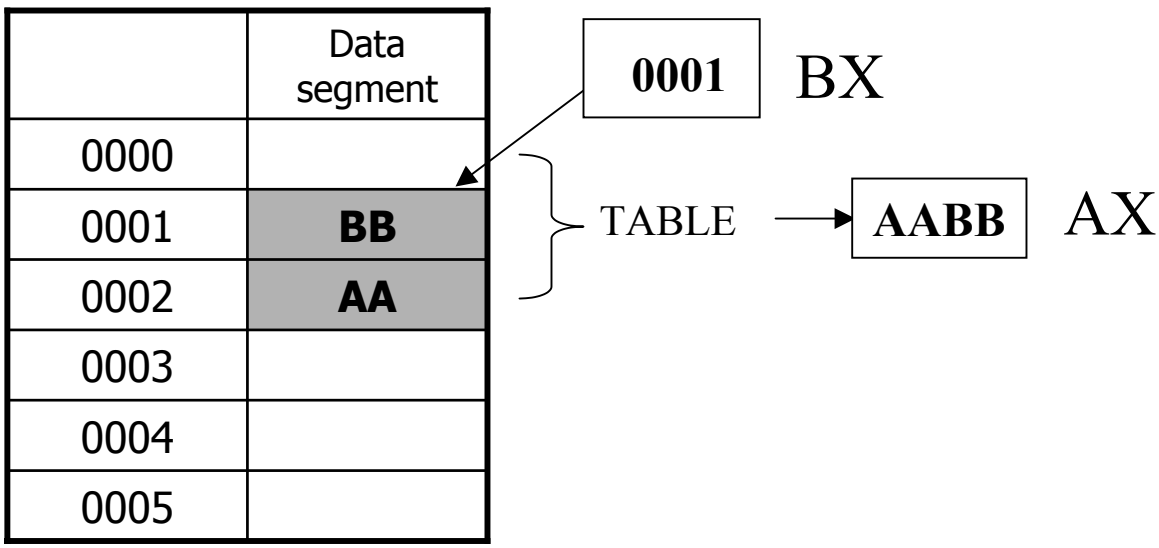
```
MOV AX,[BX]        ; load isi memory yang
                   ; ditunjuk BX ke AX
```

- Pertanyaan: bagaimana meletekkan EA ke BX?

```
MOV BX,OFFSET TABLE
```

```
MOV BX,OFFSET TABLE
MOV AX,[BX]
```

} = `MOV AX,TABLE`

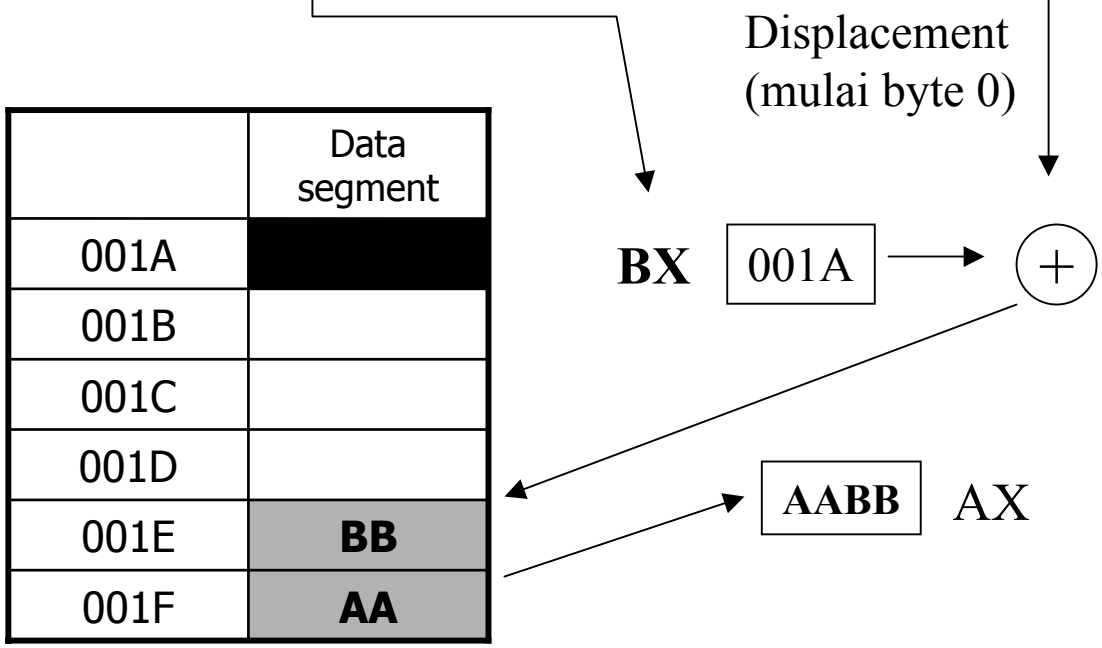


2.2. Mode Pengalamatan (cont'd)

2.2.4. Base Relative Addressing

- EA dihitung: Displacment + isi BX atau BP
  - BX cocok dipakai jika data yang hendak diakses terdapat di beberapa lokasi memori (terpisah)

```
MOV AX,[BX]+4     ;Load isi field record yang
                  ;terdapat di byte ke 5 dan 6,
                  ;dimana awal address record di
                  ;BX ke AX
```

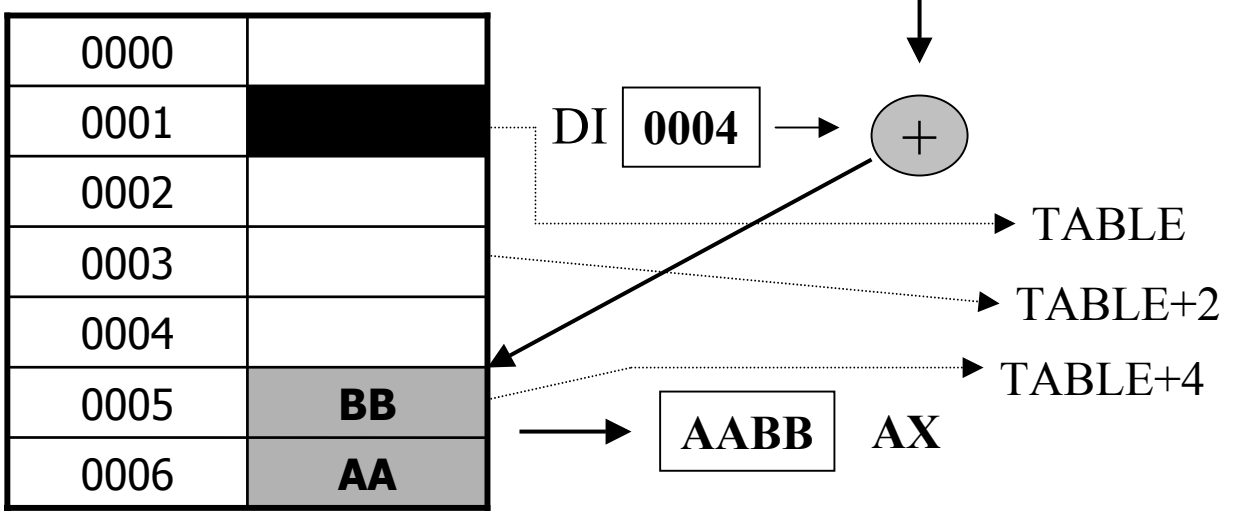


2.2. Mode Pengalamatan (cont'd)

2.2.5. Direct Indexed Addressing

- EA = Displacement + Index Reg DI atau SI
  - DI atau SI cocok untuk akses elemen dalam Tabel
  - Displacement ke awal TABLE
  - Index reg: ke → elemen di dalam Tabel

```
MOV DI,4
MOV AX,TABLE[DI] ;Load elemen ke3 ke AX
                 ;TABLE adalah Tabel 1 Byte
```

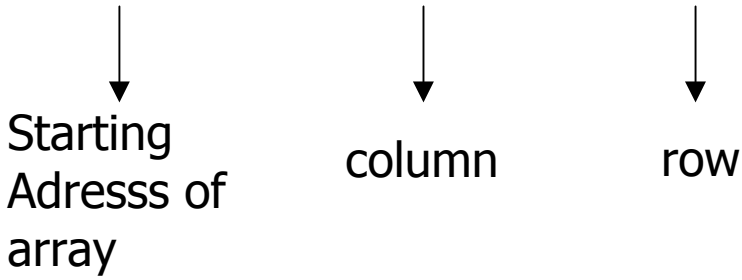


Catatan:
Diandaikan kita punya tabel dg

2.2. Mode Pengalamatan (cont'd)

2.2.6. Base Indexed Addressing

- Cocok untuk akses "two dimensional array"
- EA = Base regs + Index regs + Displ.



- Contoh:

```
MOV AX,[BX+2+DI] ;operand dapat ditulis
MOV AX,[DI+BX+2] ;dlm [],urutan sembarang
MOV AX,[BX+2][DI];displacemnent dpt digabung
MOV AX,[BX][DI+2];dgn salah satu register
```

2.3. Set Instruksi

- **Tipe-tipe instruksi:**
  - Data Transfer (Register, Memory, I/O port)
  - Operasi Aritmatik (bil. Binary atau Bil. BCD)
  - Manipulasi Bit (op. SHIFT, ROTATE, Logical dalam regs dan memory)
  - Control Transfer
  - String
  - Interrupt
  - Process Control

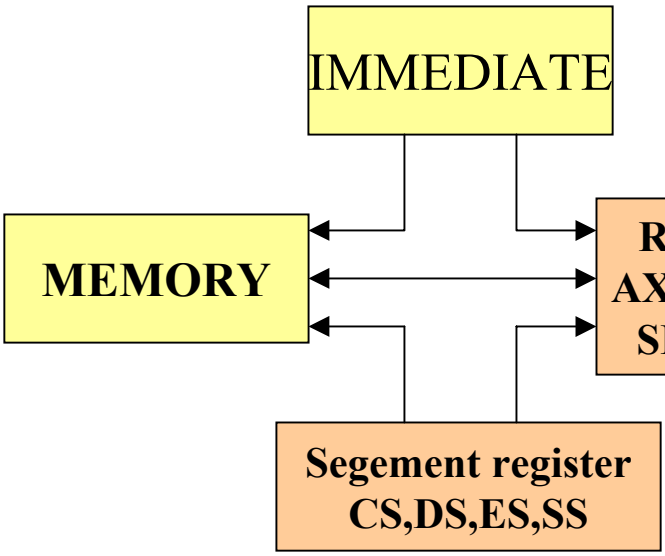
2.3.1 Data Transfer

- Berfungsi memindahkan (mengkopi) data dan address diantara regs dan lokasi memori atau I/O port
- Terbagi menjadi:
  - General Purpose : MOV, PUSH, POP, XCHG, XLAT
  - I/O : IN,OUT
  - Address Transfer : LEA, LDS, LES
  - Flag Transfer : LAHF, SAHF, PUSHF, POPF

2.3. Set Instruksi (cont'd)

General-Purpose Instructions

1. MOV → format: MOV destination,Source



Transfer Data: 1 Byte/1 Word
Antara
Reg ←→ Memory
Reg ←→ Reg
Immediate → reg/memory
data value

- Contoh:

```
MOV AX.TABLE      ;dari mem → reg
MOV TABLE,AX      ;dari reg → mem
MOV ES:[BX],AX    ;reg→segment reg (override)
MOV DS,AX         ; 16 bit reg → 16 seg. Reg
MOV BL,AL         ; 8 bit reg → 8 bit reg
MOV CL,-30        ;konstanta → reg
MOV DEST,25H      ;konstanta → mem
```

2.3. Set Instruksi (cont'd)

- **Perkecualian/tidak dapat dikerjakan:**
  - Move data antar 2 lokasi memori secara langsung → tidak kangsung via general purpose reg.

```
MOV AX,POUNDS
MOV WEIGHT,AX
```

  - Load, immediate value ke segment reg. Via GP reg.

```
MOV AX, DATA_SEG        ; load nomor seg. dari
MOV DS, AX              ; data seg ke DS menunjukan
                        ; letak data  seg.
```

  - Move isi 1 seg.reg ke seg-reg yag lain secara langsung → tidak langsung via GP reg.

```
MOV AX,ES     ;Membuat DS menunjuk ke
              ;seg. yang di
MOV DS,AX     ;tunjuk oleh ES
```

  - Segement register:
    - CS tidak dapat dipakai sbg destination

2. PUSH dan POP

- Stack
  - **dipakai u/ menyimpan address → instruksi Call dan return**
  - **Sbg penyimpan sementara dari operand: data-reg atau memori**

2.3. Set Instruksi (cont'd)

- **Format**

```
PUSH source;simpan 1 word reg. operand
           ;atau 1 word memori operand
           ;di topstack
POP destination    ;biasnya PUSH dan POP
                   ;dipakai bersama
```

- **Contoh:**

```
PUSH SI            ;kedalam stack dpt simpan
PUSH DS            ;GP reg./seg.reg/CS
PUSH CS
PUSH COUNTER       ;dapat pula lokasi memori
PUSH TABLE[BX][DI];
```

STACK		**STACK**		**STACK**	
	SS:01F8		SS:01F8		SS:01F8
	SS:01FA		SS:01FA		SS:01FA
	SS:01FC	**SP →(AX)**	SS:01FC	**(AX)**	SS:01FC
SP	SS:01FE		SS:01FE	**SP**	SS:01FE
	Berfore PUSH AX	**After PUSH AX**		**After POP AX**	

```
PUSH AX; transfer 1 word yang ditunjuk SP ke operand
   AX
POP AX; SP menunjuk address, byte yang semula kosong
   dipakai untuk menyimpan isi AX
```

2.3. Set Instruksi (cont'd)

- PUSH dan POP dipakai u/ copy 1 seg-reg ke seg-reg langsung

```
PUSH ES    ; copy ES ke DS → lambat jika
POP DS     ; dibandingkan dgn inst. MOV
```

- Stack biasa dipakai diawal subroutine

3. EXCHANGE (XCHG)

- Instruksi u/ mempertukarkan isi source operand dgn destination operand, dalam ukuran byte atau word
- Dapat mempertukarkan isi 2 memori dan isi suatu reg dan lokasi memori
- Isi segemetn reg. tidak dapat dipertukarkan (dgn instruksi XCHG)

```
XCHG AX,BX        ;pertukaran 2 word reg
XCHG AL,BH        ;pertukaran 2 Byte reg
XCHG WORD_LOC,DX ;pertukaran dgn lok. Mem
XCHG DL,BYTE_LOC ;pertukaran dgn reg.
```

4. TRANSLATE (XLAT)

- Instruksi u/ membaca/memeriksa nilai-2 (byte) yang terdpt dalam suatu TABEL dan kemudian mengkopi (load) ke dalam register AL (ukuran max TABEIL=256 byte)
- Sebelum eksekusi XLAT → starting address daripada tabelnya harus disimpan(load) ke BX dan index dari byte yang dimaksud ke dalam AL

2.3. Set Instruksi (cont'd)

- **Format:** `XLAT source_table`
- Urutan instruksi u/ memeriksa byte ke 10 dari tabel S_TAB:

```
MOV AL,10              ;load harga index AL
MOV BX,OFFSETS_TAB     ;load starting add. S_TAB
                       ;ke BX
XLAT S_TAB             ;ambil harga dlm tabel ke
                       ;AL
```

- **XLAT** cocok u/ konversi yang membutuhkan waktu perhitungan yang lama, misalkan konversi ASCII display code dari hexadesimal digit

INPUT/OUTPUT instructions

- Dipakai u/ berkomunikasi dgn peripheral devices
- **Format:** IN accumulator, port
  OUT port, accumulator
  - Accumulator → AL = byte trans.; AX= Word transf**.**
  - PORT → bil desimal(0-255) yang menunjukkan walah satu dari 256 device dalam sistem
- Reg DX
  - Dapat dipakai sbg PORT operand (64 K port)
  - Mudah merubah nomor prot dan dapt mengirim 1 data je beberapa port yang berbeda

2.3. Set Instruksi (cont'd)

- **CONTOH:**

```
IN AL,200           ; input dari port 200 dgn
                    ;ukuran 1 byte atau dari
IN AL,Port_val      ;nama portnya.
OUT 30H,AX          ;output 1 word ke port 30H
OUT DX, AX          ;atau ke port yang si
                    ;tunjuk DX
```

Address Transfer Instructions

- Transfer address suatu variabel → load address ke reg mirip dgn instruksi move
- **LEA (Load Effective Address)**

  Mis: DI = 5

```
LEA BX,TABLE[DI]; load offset address
   dari TABLE+5 ke BX
```

- **LDS (Load Pointe using DS)**
  - Membaca 32 bit dari memori, kemudian load 16 bit pertama ke reg tertentu dan 16 bit ke dua ke DS

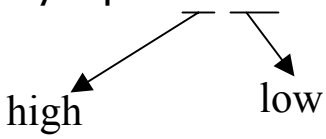
    LDS reg16, mem32
- **LES (load pointer using ES)**
  - Idem LDS, tetapi tidak ke DS tetapi ke ES

2.3. Set Instruksi (cont'd)

2.3.2. Instruksi Aritmatik

- 8088 dapat melaksanakan operasi aritmatik terhadap:
  - **Bil. Binary (bertanda/tidak)**
  - **Bil. Desimal tak bertanda (packed/unpacked)**
- Penyimpanan bil dalam memori:
  - **Low data-lowa Address;**
  - **High data-high Address**
  - **Contoh:**

Menyimpan 1234H dilokasi memori NUM



34 H di NUM
12 H di NUM+1

- **ADD (adding), ADC (add with carry)**
  - ADD AX,CX ; terbatas hanya s/d 16 bit
  - ADC BX,DX ; bila hanya >16bit → bit ke 17 di CF
    dest=dest+source+carry
  - ADD AX,MEM_Word
  - ADD AL,10 ; konstanta ke reg

ADD dan ADC memperngaruhi Flag register:
CF, PF, AF, ZF, SF, OF

2.3. Set Instruksi (cont'd)

- **SUB (subtract), SBB(subract with borrow)**
  - SUB AX,BX ; dest=dest - source
  - SBB BX,DX ; dest=dest-soruce-carry
- **Perintah lainnya:**
  - DEC (decrement destination by one)
  - MUL, IMUL (integer mult.)
  - DIV (divide, unsigned), IDIB (integer divide signed)