Organisasi Sistem Komputer

Bab 3. Perpindahan Data

3.1 Instruksi MOV

3.2 Instruksi MOVZX dan MOVSX

Pembahasan:

- Instruksi MOV dan ketentuannya

Instruksi MOV

- Sintaks:

```
MOV    destination, source
```

- Memindahkan (move) data dari operand *source* (asal) ke operand *destination* (tujuan)
- Kombinasi operand destination dan source:
  - `MOV reg, reg`
  - `MOV mem, reg`
  - `MOV reg, mem`
  - `MOV mem, imm`
  - `MOV reg, imm`

Ketentuan Instruksi MOV

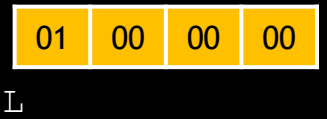
- Ketika memindahkan data dari register ke register, kedua register harus berukuran sama:
  - `MOV eax, ecx` — BENAR — ukuran eax = ecx = 32 bit
  - `MOV bx, ax` — BENAR — ukuran bx = ax = 16 bit
  - `MOV ax, ebx` — ERROR — ukuran ebx (32 bit) > ukuran ax (16 bit)
- Ketika memindahkan immediate value ke register, ukuran immediate value yang dipindahkan diasumsikan berdasarkan ukuran tujuan register:
  - `MOV al, 12 ; ax = 0Ch` — karena ukuran al = 8 bit, maka 12 diasumsikan 0C
  - `MOV bl, 0BCAFh ; bl = AFh` — BCAF mempunyai ukuran 16 bit, karena bl hanya 8 bit maka 8 bit terbawah dari nilai langsung yang dipindahkan
- Kurung kotak digunakan untuk memindahkan isi dari memori:
  - `MOV eax, [L1]` — Memindahkan isi dari memori sebesar 4 byte dengan byte pertama pada alamat yang ditunjuk L1
  - `MOV eax, L1` — Tanpa kurung kotak, kita memindahkan alamat L1 ke eax

Ketentuan Instruksi MOV

- Besar ukuran data yang dipindahkan diasumsikan dari ukuran register tujuan atau asal
  - `MOV  eax, [L1]` — Karena ukuran eax = 32 bit = 4 byte, maka pindahkan isi memori sebesar 4 byte yang dimulai dari alamat yang ditunjuk oleh L1
  - `MOV  [L1], ax` — Pindahkan nilai ax (sepanjang 2 byte) ke alamat memori yang ditunjuk L1
- Kita tidak dapat memindahkan data dari alamat memori ke alamat memori lain
  - `MOV [L1], [L2]` — ERROR
- Untuk memindahkan immediate value ke memori, kita memerlukan size specifier
  - `MOV [L] , 1` — ERROR
  - `MOV dword [L], 1` — BENAR — dword adalah size specifier yang berarti double word (4 byte)

Memindahkan Immediate Value

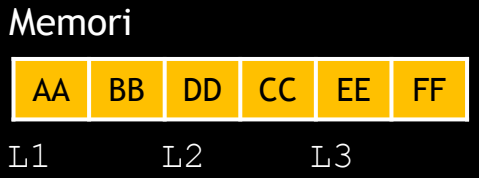
- Misalkan instruksi berikut: `MOV [L], 1`
- Assembler akan memberikan error: “operation size not specified!”
- Ini karena assembler tidak mengetahui apakah yang kita maksud dengan “1” adalah 01h, 0001h, 00000001h, dst.
  - Label tidak mempunyai tipe data
- Assembler menyediakan cara untuk menentukan ukuran dari operand immediate value: size specifier
- Lima size specifier: byte (1 byte), word (2 byte), dword (4 byte), qword (8 byte), tword (10 word)
- Contoh:
  - `MOV dword [L], 1`
  - Nilai 1 disimpan dalam 4-byte



Memindahkan Data dari/ke Memori

- Misalkan kita mempunyai deklarasi berikut pada segment data:

```
L1    DB    0AAh, 0BBh
L2    DW    0CCDDh
L3    DB    0EEh, 0FFH
```



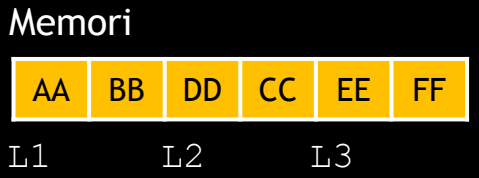
- Pada segment text:

```
MOV    EAX, [L2]
MOV    AX,  [L3]
MOV    [L1], EAX
```

Memindahkan Data dari/ke Memori

- Misalkan kita mempunyai deklarasi berikut pada segment data:

```
L1    DB    0AAh, 0BBh
L2    DW    0CCDDh
L3    DB    0EEh, 0FFH
```



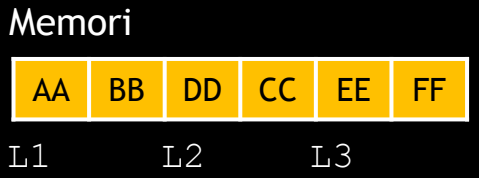
- Pada segment text:

```
MOV    EAX, [L2]
MOV    AX,  [L3]
MOV    [L1], EAX
```

Memindahkan Data dari/ke Memori

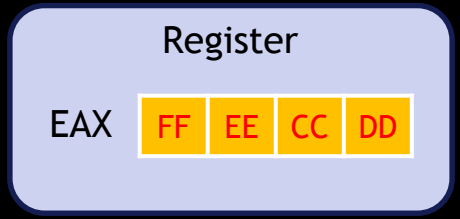
- Misalkan kita mempunyai deklarasi berikut pada segment data:

```
L1    DB    0AAh, 0BBh
L2    DW    0CCDDh
L3    DB    0EEh, 0FFH
```



- Pada segment text:

```
MOV    EAX, [L2]
MOV    AX,  [L3]
MOV    [L1], EAX
```

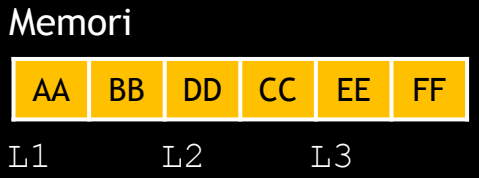


`MOV EAX, [L2]` berarti "Pindahkan isi memori sebesar 4 byte (karena ukuran `EAX` = 4 byte) dimulai dengan isi byte dari alamat yang ditunjuk oleh label `L2`"

Memindahkan Data dari/ke Memori

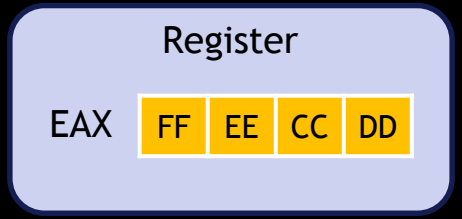
- Misalkan kita mempunyai deklarasi berikut pada segment data:

```
L1    DB    0AAh, 0BBh
L2    DW    0CCDDh
L3    DB    0EEh, 0FFH
```



- Pada segment text:

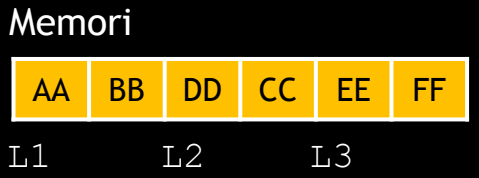
```
MOV    EAX, [L2]
MOV    AX,  [L3]
MOV    [L1], EAX
```



Memindahkan Data dari/ke Memori

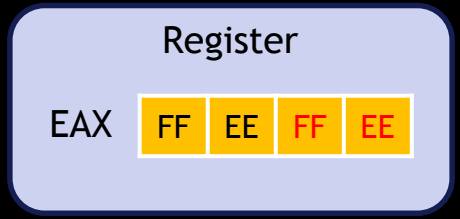
- Misalkan kita mempunyai deklarasi berikut pada segment data:

```
L1    DB    0AAh, 0BBh
L2    DW    0CCDDh
L3    DB    0EEh, 0FFH
```



- Pada segment text:

```
MOV    EAX, [L2]
MOV    AX,  [L3]
MOV    [L1], EAX
```

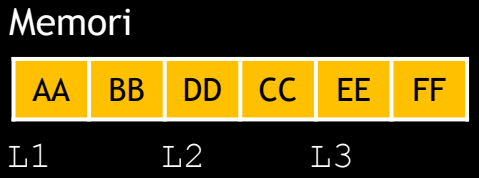


`MOV AX, [L3]` berarti "Pindahkan isi memori sebesar 2 byte (karena ukuran `AX` = 2 byte) dimulai dengan isi byte dari alamat yang ditunjuk oleh label `L3`"

Memindahkan Data dari/ke Memori

- Misalkan kita mempunyai deklarasi berikut pada segment data:

```
L1    DB    0AAh, 0BBh
L2    DW    0CCDDh
L3    DB    0EEh, 0FFH
```



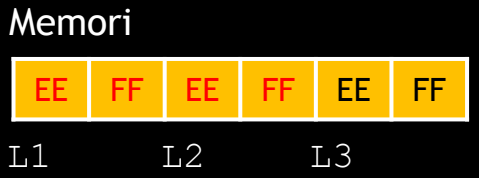
- Pada segment text:

```
MOV    EAX, [L2]
MOV    AX,  [L3]
MOV    [L1], EAX
```

Memindahkan Data dari/ke Memori

- Misalkan kita mempunyai deklarasi berikut pada segment data:

```
L1    DB    0AAh, 0BBh
L2    DW    0CCDDh
L3    DB    0EEh, 0FFH
```



- Pada segment text:

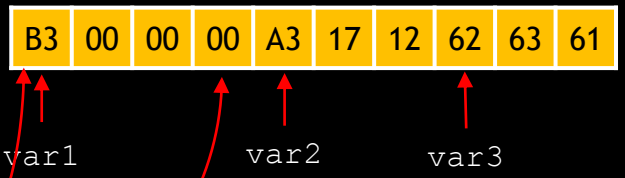
```
MOV    EAX, [L2]
MOV    AX,  [L3]
MOV    [L1], EAX
```

`MOV [L1], EAX` berarti "pindahkan isi dari `EAX` sebesar 4 byte ke alamat yang ditunjuk oleh `L1`"

Kurung Kotak

- `MOV  eax, [L]`
  - Memindahkan isi dari address `L` ke register `eax`
  - Memindahkan 32 bit (4 byte) isi, karena eax adalah register 32 bit
- `MOV  eax, L`
  - Memindahkan address `L` ke `eax`
  - Memindahkan 32-bit address `L` ke `eax`
- `MOV  ebx, [eax]`
  - Memindahkan isi dari memori pada alamat memori yang berada dalam `eax` (=L) ke `ebx`

Contoh Kurung Kotak



Dalam heks = B3

```
var1        dd      179
var2        db      0A3h, 017h, 012h
var3        db      "bca"
```

```
mov     eax, var1
add     eax, 3
mov     ebx, [eax]
add     ebx, 5
mov     [var1], ebx
```

eax = alamat var1

eax

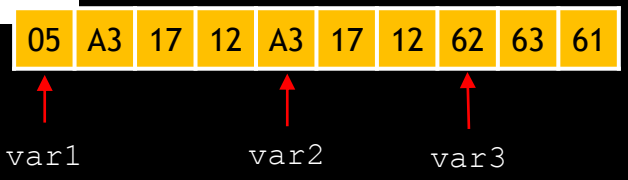
eax = alamat var1+3

eax

ebx = 12 17 A3 00

ebx = 12 17 A3 05

Pindahkan isi ebx ke isi memori alamat var1



Demo (3.1a)

```
   ; directive include
%include "asm_io.inc"

segment .data
   ; directive Dx
    L1  db  0AAh, 0BBh
    L2  dw  0CCDDh
    L3  db  0EEh, 0FFh

segment .bss
   ; directive RESx


segment .text
   global _main
   _main:
      ; Routine "setup"
      enter  0, 0
      pusha

      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      ;;  Tuliskan kode program Anda di bawah      ;;
      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

      dump_mem  1, L1, 0   ; print layout memori sebelum perpindahan data
      mov  eax, [L2]       ; eax = FF EE DD CC

      mov  ax, [L3]        ; eax = FF EE FF EE
      mov  [L1], eax
      dump_mem  3, L1, 0   ; print layout memori setelah perpindahan data

      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      ; Routine "cleanup"
      popa
      mov    eax, 0
      leave
      ret
```

sebelum perpindahan data

```
Memory Dump # 1 Address = 00405004
00405000 00 00 00 00 AA BB DD CC EE FF 00 00 25 69 00 25 "????????????%i?%"
Memory Dump # 3 Address = 00405004
00405000 00 00 00 00 EE FF EE FF EE FF 00 00 25 69 00 25 "????????????%i?%"
```

sesudah perpindahan data

Demo (3.1b)

```
var1        dd      179
var2        db      0A3h, 017h, 012h
var3        db      "bca"
```

```
mov     eax, var1
add     eax, 3
mov     ebx, [eax]
add     ebx, 5
mov     [var1], ebx
```