Organisasi Sistem Komputer

Bab 2. Pengenalan ke Program Assembly (NASM)

2.1 Arsitektur x86

2.2 Assembler dan Linker

2.3 Menulis Hello World dalam Bahasa Assembly

2.4 Struktur Program NASM

Pembahasan:

- Struktur Program NASM
- `segment .data`
- `segment .bss`
- `segment .text`

Struktur Program NASM

```
segment .data
   ; directive Dx

segment .bss
   ; directive RESx

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

      ; Program Anda di bawah

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

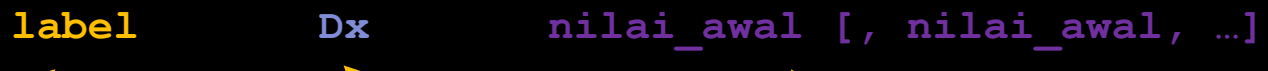
segment **data** - tempat mendeklarasikan variabel terinisialisasi (dengan nilai awal)

segment **bss** - tempat mendeklarasikan variabel tidak terinisialisasi (tanpa nilai awal)

segment **text** - tempat menuliskan kode program

Segment .data

- Perintah deklarasi data terinisialisasi:



label

- Digunakan sebagai nama variable
- Bisa ditentukan apasaja selama tidak sama dengan *reserved words*
- Case sensitive
- Penanda (pointer) ke address memori byte pertama dari data
- Opsional

nilai_awal

- Bisa berupa nilai konstan, karakter, atau string
- Bisa lebih dari satu (mis. untuk deklarasi array)

Directive **Dx**

perintah assembler untuk besar alokasi ruang memori

Segment .data

```
label        Dx   nilai_awal [, nilai_awal, …]
```

- Directive *Dx*:
- nilai_awal:
  - Nilai konstan
    - Format: `(+/-)digit[radiks]`
    - Radiks:
      - `d`: decimal (default)
      - `b`: biner
      - `h`: heksadesimal
      - `o`: octal
    - Contoh: `26, 26d, 11010011b, 42o, 1Ah, 0A3h, 0xFE`
  - Character/String:
    - Ditulis dalam tanda kutip tunggal atau kutip ganda
    - Contoh: `'A', "Hello"`

Directive	Kegunaan	Ruang Memori
DB	Define Byte	1 byte
DW	Define Word	2 byte
DD	Define Double Word	4 byte
DQ	Define Quad Word	8 byte
DT	Define Ten Byte	10 byte

Angka heksadesimal juga dapat ditulis tanpa radiks `h`, namun harus diawali dengan `0x`

Angka heksadesimal yang diawali dengah huruf harus ditulis dengan awalan `0`

Contoh Deklarasi Variabel .data

```
var1      DW      12345
var2      DB      17o
var3      DB      110101b
bil_neg   DW      -12345
L1        DB      2Fh
bil_hex   DD      0FFFF1A92h

L2        DB      "A"
```

← 2 byte, bernama var1, di-inisialisasi ke `12345`

← 1 byte, bernama var2, di-inisialisasi ke oktal `17`

← 1 byte, bernama var3, di-inisialisasi ke biner `110101`

← 2 byte, bernama bil_neg, di-inisialisasi ke desimal negatif `12345`

← 1 byte, bernama L1, di-inisialisasi ke heksadesimal 2FH

← 1 byte, bernama bil_hex, di-inisialisasi ke heksadesimal FFFF1A92 (**catatan**: heksadesimal yang dimulai dengan huruf harus ditambahkan 0 di depannya)

← 1 byte, bernama L2, di-inisialisasi ke kode ASCII untuk "A" (65d)

Kode ASCII

Karakter "A" direpresentasikan dalam decimal 65 atau heksadesimal 41

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Dx dengan Lebih dari Satu Nilai Awal

- `L3  DB  0, 1, 2, 3`
  - Mendefinisikan 4 bytes, di-inisialisasi ke 0, 1, 2, dan 3
  - `L3` adalah pointer ke byte pertama
- `L4  DB  "w", "o", "r", "d", 0`
  - Mendefinisikan string ***null-terminated***, di-inisialisasi ke "`word\0`"
  - `L4` adalah pointer ke byte awal string
- `L5  DB  "word", 0`
  - Ekuivalen dengan di atas, lebih mudah ditulis

Dx dengan kualifier *times*

- Misal, kita ingin mendeklarasikan 100 bytes yang kesemuanya di-inisialisasi ke 0
- Kita harus menuliskan:

```
L6    DB     0, 0, 0, …, 0    (0 ditulis sebanyak 100 kali)
```

- NASM menyediakan *shortcut* untuk melakukan ini: kualifier *times*
- Dengan kualifier *times*:

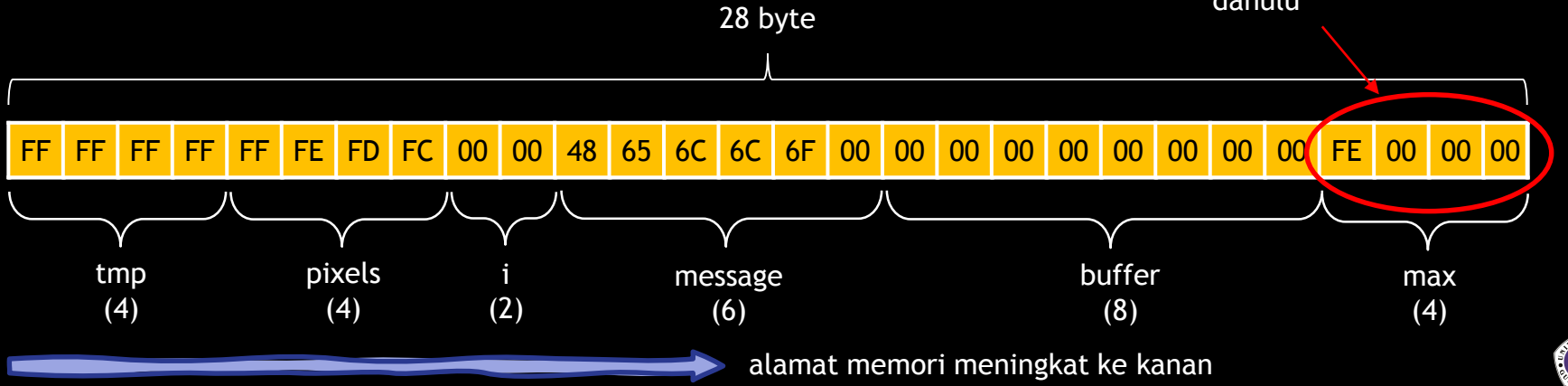
```
L6    times  100    DB 0
```

Contoh Segment .data

```
segment .data
    ; directive Dx
    tmp             DD      -1
    pixels          DB      0FFh, 0FEh, 0FDh, 0FCh
    i               DW      0
    message         DB      "H", "e", "llo", 0
    buffer          times   8   DB   0
    max             DD      254
```

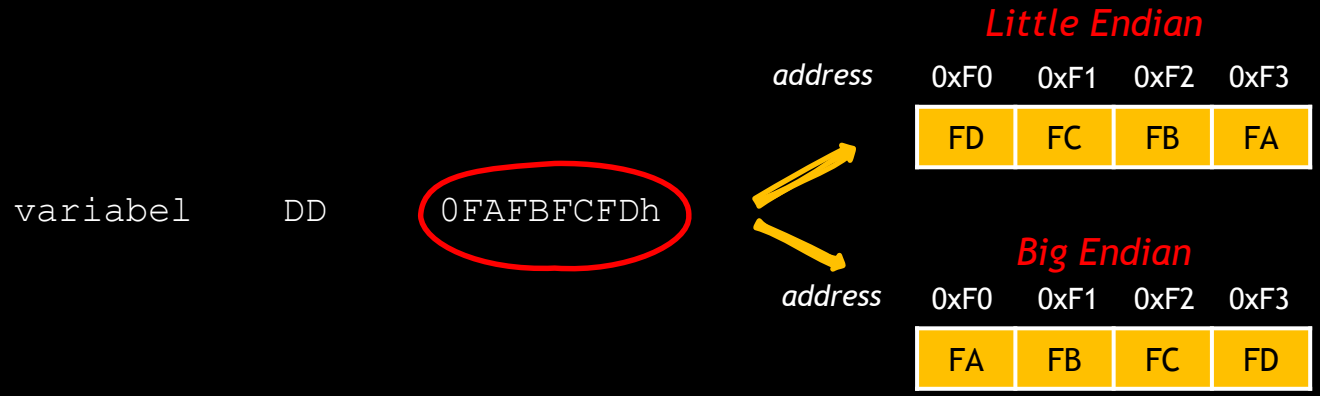
254 dalam 4 byte = `00 00 00 FE`

Little Endian
Data multi-byte disimpan dengan urutan byte terkecil dahulu



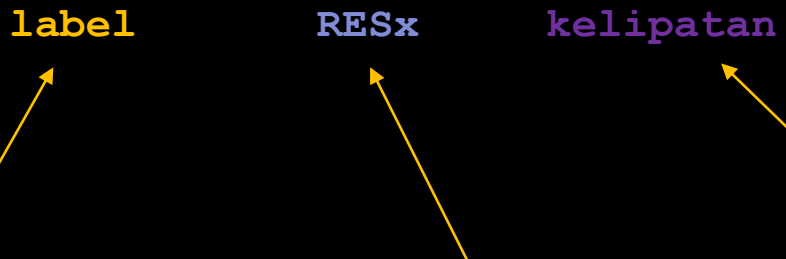
Big Endian dan Little Endian

- Terdapat dua metode pengurutan byte-byte dari data multi-byte dalam memori:
  - Little Endian: digunakan oleh prosesor Intel dan AMD
  - Big Endian: digunakan oleh prosesor IBM



Segment .bss

- Perintah deklarasi data tidak terinisialisasi:



`label`

- Digunakan sebagai nama variable
- Bisa ditentukan apasaja selama tidak sama dengan *reserved words*
- Case sensitive
- Penanda (pointer) ke address memori byte pertama dari data
- Opsional

Directive **`RESx`**

keyword untuk besar alokasi ruang memori tidak terinisialisasi

`kelipatan`

- Pengali banyaknya ruang memori dari yang didefinisikan pada RESx
- Berupa angka konstan atau ekspresi yang menghasilkan konstan

Segment .bss

```
label     RESx    kelipatan
```

- Directive *RESx*:
- kelipatan:
  - Pengali ruang memori dari unit yang didefinisikan RESx
  - Berupa nilai konstan atau ekspresi yang menghasilkan konstan
- Contoh:

Directive RESx	Kegunaan	Ruang Memori
RESB	Reserve Byte	1 byte
RESW	Reserve Word	2 byte
RESD	Reserve Double Word	4 byte
RESQ	Reserve Quad Word	8 byte
REST	Reserve Ten Byte	10 byte

```
temp        RESB    1
reserve     RESW    100
```

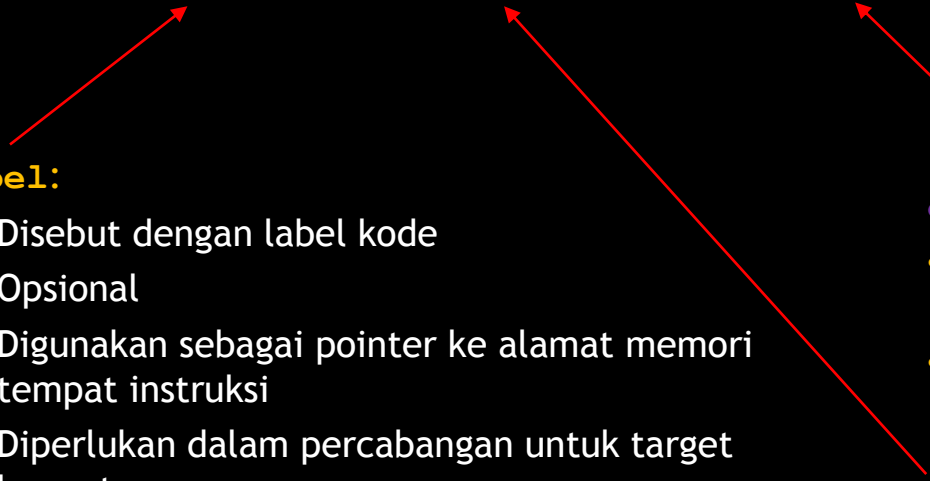
← Mengalokasikan 1 x 1 byte (= 1 byte) ruang memori dengan nama temp

← Mengalokasikan 100 x 2 byte (= 200 byte) ruang memori dengan nama reserve

Segment .text

- Format sintaks instruksi:

```
label:     mnemonic     operand [, operand, …]
```



`label:`

- Disebut dengan label kode
- Opsional
- Digunakan sebagai pointer ke alamat memori tempat instruksi
- Diperlukan dalam percabangan untuk target lompatan
- Dapat dinamakan apa saja selama tidak sama dengan *reserved words* dan harus diakhiri titik dua (:)

`operand`

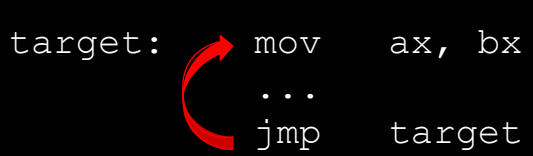
- Berupa register, alamat/isi memori, immediate value (nilai langsung)
- Bisa lebih dari satu, tergantung dari mnemonic

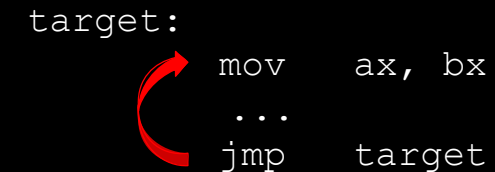
`mnemonic`

Keyword untuk instruksi operasi CPU

Segment .text

- Untuk apa label pada instruksi ?
  - CPU mempunyai instruksi untuk mengubah alur instruksi (untuk percabangan) yaitu JMP
  - Label digunakan sebagai target dari instruksi JMP



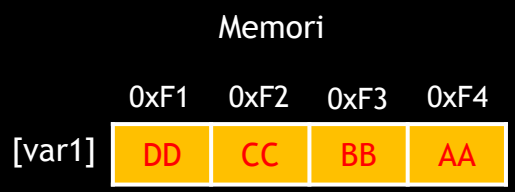


Instruksi `JMP target`: instruksi selanjutnya diarahkan ke instruksi dengan label target

Umumnya dituliskan tidak satu baris dengan instruksi untuk memudahkan membacanya

Operand

- Jenis operand-operand:
  - **Register**: nama register
    - `ADD eax, ebx` ← berarti eax = eax + ebx
  - **Memori**: alamat/isi memori
    - Tanda kurung kotak [ ] digunakan untuk mereferensi isi memori
    - `MOV eax, [var1]` ← berarti pindahkan isi memori dari alamat var1 ke register eax; eax = AABBCCDD
    - `MOV ecx, var1` ← berarti pindahkan alamat memori var1 ke register ecx; ecx = 000000F1
  - **Immediate Value**: nilai langsung berupa angka konstan
    - `ADD eax, 2` ← berarti eax = eax + 2



Ringkasan

- Struktur NASM: segment .data, segment .bss, dan segment .text
- Segment .data untuk mendeklarasikan data terinisisalisasi
  - `label  Dx  nilai_awal [, nilai_awal, …]`
- Segment .bss untuk mendeklarasikan data tidak terinisialisasi
  - `label  RESx  kelipatan`
- Segment .data untuk menuliskan kode-kode instruksi
  - `label:  mnemonic  operand [, operand, …]`