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:

- Menulis program “Hello, World!” menggunakan bahasa assembly

“Hello, World!” dalam Assembly

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

segment .data
    ; directive Dx
        hello    db    "Hello, World!",0

segment .bss
    ; directive RESx

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

        ; Instruksi-instruksi
        mov     eax, hello      ; pindahkan alamat hello ke eax
        call    print_string    ; panggil fungsi print_string

        ; Routine “cleanup”
        popa
        mov     eax, 0
        leave
        ret
```

Area untuk menyertakan file ekternal dengan directive `%include`

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

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

segment **text** - tempat menuliskan kode program

Comment

- Bahasa assembly tidak mudah untuk dibaca, penulisan comment sangatlah penting
- Comment pada NASM diawali dengan ‘;’ (titik koma)
- Contoh:

```
add eax, ebx        ; y = y + b
```

Directive

- Directive adalah perintah ke assembler untuk melakukan sesuatu saat proses assembly namun bukan perintah yang diterjemahkan menjadi instruksi kode mesin
  - Antara lain: `%define` dan `%include`
- Kita menggunakan directive `%define` untuk mendefinisikan constant
  - Misalkan kode kita sering munggunakan angka 100 untuk suatu hal tertentu, misalkan untuk besar ukuran
  - Kita dapat menuliksan dalam kode NASM : `%define SIZE 100`
  - Saat proses assembly, NASM akan mensubtitusi semua kata `SIZE` dalam program dengan nilai 100
- Kita menggunakan directive `%include` untuk menyertakan macro kode assembly pada file lain ke kode :
  - `%include "nama_file.inc"`

Reserved Words

- *Reserved Words* adalah kata-kata yang mempunyai arti tertentu dan harus digunakan dalam konteks yang sesuai
- Reserved words antara lain:
  - Instruction Mnemonic; seperti: `MOV`, `ADD`, dan `MUL`
  - Nama Register: `EAX`, `EBX`, dst.
  - Directive;
  - Specifier: untuk informasi ukuran data dari variable dan operand; contoh: `BYTE` dan `WORD`

“Setup” dan “Clean-up”

- Sebelum dan setelah menjalankan instruksi pada program, kita perlu melakukan routine “setup” dan “clean-up”
- Kita akan memahami ini nanti, untuk saat ini kita akan menganggap segment text kita akan selalu seperti ini:
- Semua kode instruksi yang kita tulis berada setelah routine “setup” dan sebelum routine ”clean-up”

```
segment .text
    global _main
    _main:
        ; Routine “setup”
        enter   0, 0
        pusha

        ; Instruksi-instruksi



        ; Routine “clean-up”
        popa
        mov     eax, 0
        leave
        ret
```

Input dan Output

- Menulis program assembly yang menerima input dan mengeluarkan output merupakan hal yang sulit
- Kita perlu memahami system call dari sistem operasi (berbeda untuk setiap sistem operasi)
- Untuk memudahkan, kita akan memanfaatkan dua file: `asm_io.inc` dan `asm_io.asm`, yang berisi fungsi-fungsi dan macro-macro yang dapat digunakan untuk input dan output
- Kita menyertakan file asm_io.inc pada kode program dengan menuliskan perintah include pada bagian awal kode:

```
%include "asm_io.inc"
```

- Kita juga akan menyertakan `asm_io.asm` pada proses link

Fungsi I/O dalam asm_io

- `print_char` : print karakter yang sesuai dengan kode ASCII yang disimpan dalam register `AL`
- `print_string` : print isi dari string yang disimpan dalam alamat dalam `EAX`
- `print_nl` : print baris baru
- `read_int` : membaca sebuah integer dari keyboard dan menyimpannya dalam `EAX`
- `read_char` : membaca sebuah karakter dari keyboard dan menyimpannya dalam `AL`

Macro-macro dalam asm_io.inc

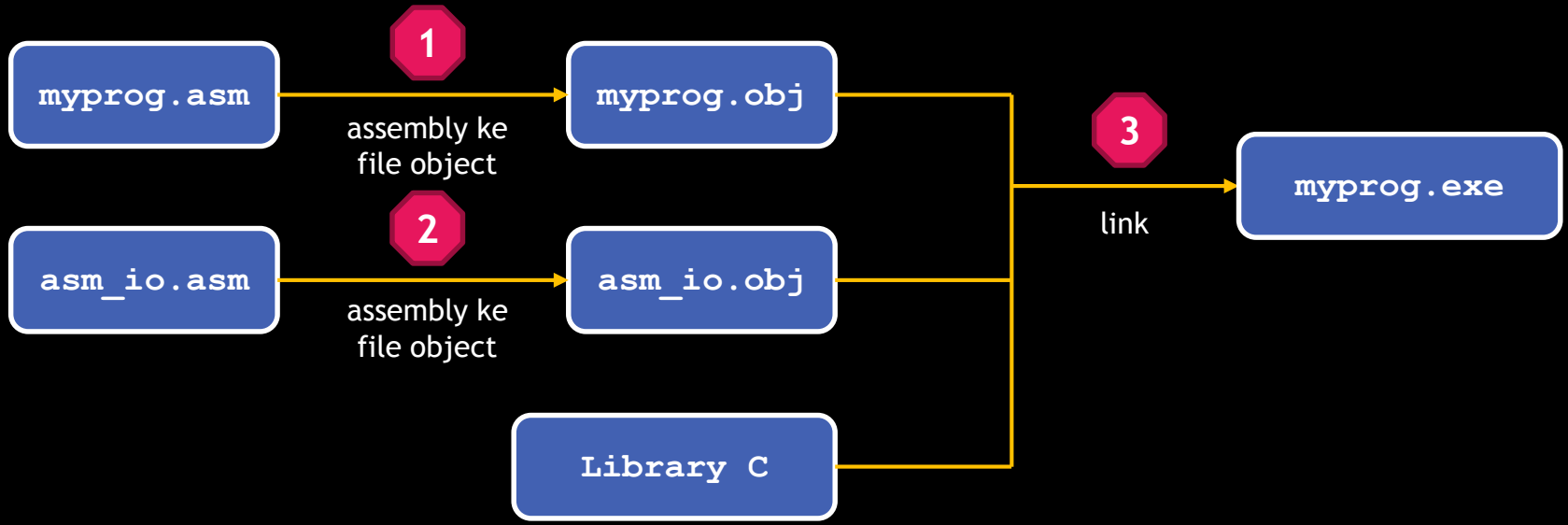
- `dump_regs` : print byte-byte yang disimpan dalam register dan juga bit-bit dalam register `EFLAGS`
- `dump_memory` : print byte-byte yang disimpan dalam memori

File Skeleton

- Kita akan menggunakan file skeleton sebagai template program

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

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
```

Proses Assembly (dengan asm_io)



Proses Assembly (dengan asm_io)

!!!! Pastikan file `asm_io.inc`, `asm_io.asm`, dan program assembly Anda dalam satu folder/direktori

1. Assembly `<nama_prog>.asm` menjadi `<nama_prog>.obj`:

```
> nasm -f win32 <nama_prog>.asm
```

2. Assembly `asm_io.asm` menjadi `asm_io.obj`:

```
> nasm -f win32 asm_io.asm
```

3. Link `<nama_prog>.obj`, dan `asm_io.obj`:

```
> gcc -m32 -o <nama_program>.exe <nama_prog>.obj asm_io.obj
```

Selanjutnya….

Menulis Program Pertama dalam Assembly