Kuliah 3:

3.1. Debugging dan TASM

3.2. Interrupt

3.1.1. Debugging

- **Proses Pembuatan Program**

Terdapat 5 langkah:

- **Desain Algoritma:**
  - Penetapan masalah
  - Pengusulan solusi yang terbaik
  - Pembuatan diagram skematik yang digunakan agar usulan solusi lebih baik →flowchart
- **Pengkodean algoritma**
  - Penulisan program dalam bahasa pemrograman tertentu
- **Penerjemaahan ke bahasa mesin**
  - Pembuatan program obyek, atau dgn kata lain, program ditulis dalam suatu deret/urutan 0 dan 1 yang dapat diinterpretasi oleh prosessor
- **Test program**

  Eksekusi program pada mesin komputer
- **Mengeliminasi kesalahan yang terdeteksi**
  - Koreksi kesalahan umumnya memerlukan pengulangan seluruh langkah-2 dari langkah 1 atau 2

Untuk membuat program dlm bahasa rakitan tersedia dua option:

- Mengunakan Turbo Assembler (TASM, Borland)
- Debugger

3.1.1. Debugging (cont'd)

- **Debug:**
  - Hanya dapat membuat program dgn extension *.COM → karena karakteristik sifat program ini →
    - program tidak lebih besar 64 K dan
    - Program harus dimulai dgn displacment, offset, atau 0100H memory direction didalam segment tertentu (spesific)
  - Kumpulan perintah debug:
    - A: *merakit intruksi simbolik → kode mesin*
    - D: *menampilkan isi suatu daerah memori*
    - E: *memasukan data ke memori, dimulai pada lokasi tertentu*
    - G: *run executable program ke memori*
    - N: *menamai program*
    - P: *ekskusi sekumpulan instruksi yang terkait*
    - Q: quit
    - R: *menampilkan isi satu atau lebih registers*
    - T: *trace isi sebuah instruksi*
    - U: *unassembled kode mesin ke kode simbolik*
    - W: *menulis program ke disk*
  - Dengan debug → visualisasi nilai registers internal dari CPU
- **Memulai debug:**

```
C:/>debug[enter]
```

3.1.1. Debugging (cont'd)

```
-r[enter]
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000
SI=0000 DI=0000 DS=0D62 ES=0D62 SS=0D62 CS=0D62
IP=0100 NV EI PL NZ NA PO NC 0D62:0100 2E
CS:0D62:0101 803ED3DF00 CMP BYTE PTR [DFD3],00
CS:DFD3=03
```

→ Seluruh isi registers internal ditampilkan

Atau dgn menggunakan parameter nama register,mis:

```
-rbx
BX 0000
:
```

→ akan ditampikan isi register BX dan indikator debug berubah dari "-" → ":" (ini berarti kita dapat mengisi nilai baru dari register BX (masukan nilai + enter), atau kita tetap mempertahankan nilai lama (tekan langsung enter tanpa mengetik nilai)

- **Pembuatan program assembler dasar**
  - Menginisialisai Debug, yaitu mengetik `debug[enter]` pada prompt OS
  - Assemble suatu program dgn perintah "a" yaitu a parameter[enter]
    - Parameter → alamat awal proses perakitan
    - Jika tanpa parameter → diinisialisasi pada posisi (locality) yang dispesifikasikan oleh CS:IP (biasanya 100h yaitu locality dimana sebuah program dgn extension *.CON diinisialisasi)

3.1.1. Debugging (cont'd)

- Rekomendasi: lebih baik alamat awal ditulis u/ menghindari masalah-2 yang terjadi segera setelah register CS:IP digunakan

→ dgn demikian kita ketikan:

```
a 100[enter]
mov ax,0002[enter]
mov bx,0004[enter]
add ax,bx[enter]
nop[enter][enter]
```

- Keterangan:
  - move nilai 0002 ke AX,
  - move nilai 0004 ke BX,
  - tambahkan nilai bx ke ax,
  - Intruksi nop → finish the program
- Kemudian:

```
C:\>debug
-a 100
0D62:0100 mov ax,0002
0D62:0103 mov bx,0004
0D62:0106 add ax,bx
0D62:0108 nop
0D62:0109
```

`-t` (execute setiap instruksi dari program)

AX=0002 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000
SI=0000 DI=0000 DS=0D62 ES=0D62 SS=0D62
CS=0D62 IP=0103 NV EI PL NZ NA PO NC 0D62:0103 BB0400
MOV BX,0004

3.1.1. Debugging (cont'd)

```
-t
AX=0002 BX=0004 CX=0000 DX=0000 SP=FFEE BP=0000
SI=0000 DI=0000 DS=0D62 ES=0D62 SS=0D62
CS=0D62 IP=0106 NV EI PL NZ NA PO NC
0D62:0106 01D8 ADD AX,BX
-t
AX=0006 BX=0004 CX=0000 DX=0000 SP=FFEE BP=0000
SI=0000 DI=0000 DS=0D62 ES=0D62 SS=0D62 CS=0D62
IP=0108 NV EI PL NZ NA PE NC 0D62:0108 90 NOP
```

- **Menyimpan program**
  - Cek panjang prorgram dgn perintah "h"

    ```
    -h 10a 100 (alamat akhir 10a)
    020a 000a
    ```

  - Menamai prorgram

    ```
    -n test.com
    ```

  - Gunakan perintah "rcx" untuk mengubah nilai registes CX dgn nilai yg diperoleh dari ukuran file → 000a

    ```
    -rcx
    CX 0000
    :000a
    ```

  - Terakhir, gunakan perintah "w" u/ menyimpan

    ```
    -w
    Writing 000A bytes
    ```

3.1.1. Debugging (cont'd)

- **Meload program**
  - Berikan nama file yang akan diload
  - Load dgn perintah "l"

```
-n test.com
-l
-u 100 109
0C3D:0100 B80200 MOV AX,0002
0C3D:0103 BB0400 MOV BX,0004
0C3D:0106 01D8 ADD AX,BX
0C3D:0108 CD20 INT 20
```

  - Perintah "u" digunakan u/
    - memverifikasikan bhw program telah diloaded ke memori. Ini berarti dgn perintah ini: kode di unassemble, kemudian meanmpilakn hasil unassemble
    - Parameter mengindikasi Debug dari mana dan disassamble ke mana.
  - Debug selalu meload program ke memori pada alamat 100h, kecuali digunakan paramater lain.

3.1.2. TASM

- **Membangun Program Assembler**
  - Software yang diperlukan
    - Editor → pembuatan source program
    - Compiler → menerjemahkan source program ke program obyek (→ TASM)
    - Linker → menggenerasi executable program dari program obyek (→ TLINK)
- **Pemrograman dgn TASM**
  - Directives berikut penting untuk dimasukan
    - .Model SMALL → mendefinisikan model memori yg digunakan dlm prog.
    - .CODE → mendefinisikan instruksi prog.
    - .STACK → reserves suatu ruang memori bagi instruksi-2 prog. ke stack
    - END → finishes prog.
  - Langkah-langkah
    - Gunakan prog. Editor u/ membuat source prog. Contoh:

```
.MODEL SMALL    ;memory model
.STACK          ;memory space for program
                ;instructions in the stack
.CODE           ;the following lines are
                ;program instructions
```

3.1.2. TASM (cont'd)

```
mov ah,1h       ;moves the value 1h to
                ;register ah
mov cx,07h      ;moves the value 07h to
                ;register cx
int 10h         ;10h interruption
mov ah,4ch      ;moves the value 4 ch to
                ;register ah
int 21h         ;21h interruption
END             ;finishes the program code
```

- Save program, misal contoh1.asm (disave dalam ASCII format !!)
- Gunakan TASM u/ membuat prog. Obyek

```
C:\>tasm contoh1.asm
```

- Gunakan TLINK u/ membangun excutable prog.

```
C:\>tlink contoh1.obj
```

- Eksekusi executable prog.

```
C:\>contoh1[enter]
```

- **Proses assembly** → manajemen segmen dan perbandingan dgn tabel simbol (dalam hal token-token dari instruksi)

3.1.2. TASM (cont'd)

– Contoh lain: (contoh2.asm)

```
.model small
.stack
.code
mov ah,2h   ;moves the value 2h to register ah
mov dl,2ah  ;moves de value 2ah to register
            ;dl
            ;(Its the asterisk value in ;ASCII
            ;format)
int 21h     ;21h interruption
mov ah,4ch  ;4ch function, goes to operating
            ;system
int 21h     ;21h interruption
end         ;finishes the program code
```

➢ Dengan proses yang sama seperti contoh sebelumnya, program akan menghasilkan karakter asterik "*" pada layar monitor

3.2 Interrupts

- **Jenis-Jenis Interrupts**
  - **Internal HW interruptions**
    - Ditimbulkan/digenerasi oleh peristiwa tertentu yang terjadi pada waktu/selama eksekusi program
    - Diatur oleh HW dan tidak mungkin diubah
    - Contoh: tipe interrupt u/ counter clock internal; HW call interrupt ini u/ memaintance "time to date"
  - **External HW interupstions**
    - ✓ Ditimbulkan/digenerasi oleh devais peripheral, mis keyboard, printers, dsb.
    - ✓ Biasa juga ditimbulkan/digenerasi oleh Co-prosessor
    - ✓ Tidak mungkin mendeaktivekan
    - ✓ Tidak dikirim langsung ke CPU, melainkan ke IC yang memiliki fungsi u/ menghandle secara eksklusive interrupts ini. (IC → PIC8259A, yg dikontrol oleh CPU dgn menggunakan urutan komunikasi → PATH Control
  - **Software interruptions**
    - Diaktifkan langsung oleh assembler melalui sejumlah interuosi yg diharapkan dgn instruksi INT

3.2 Interrupts (cont'd)

- Terdapat dua jenis:
  - DOS interruptions
  - BIOS interupstions
- ➢ Perbedaannya:
- → DOS int. lebih mudah digunakan, namun lebih lambat, karena int. jenis ini menggunakan BIOS
- → BIOS int. lebih cepat, namun banyak kerugiannay karena BIOS bagian HW dan HW-specific

- Pemilihan interupts tergantung pada karakteristik yang akan kita berikan pada program: SPEED → BIOS int; PORTABILITY → DOS int.

- **Interrupts yang sering digunakan**
  - Int 21H (DOS interruption) Multiple class to DIS functions
  - Int 10H (BIOS) Video I/O
  - Int 16H (BIOS) Keyboard I/O
  - Int 17H (BIOS) Printer I/O