Pemrograman Bahasa Tingkat Rendah

(Turbo Assembler)

Mata Kuliah Teknik Informatika

Disusun oleh:
Cecep Suwanda, S.Si., M.Kom.

Program Studi Teknik Informatika
Fakultas Teknologi Informasi
Universitas Bale Bandung

30 September 2025

Kata Pengantar

Buku ini disusun sebagai bahan ajar untuk mata kuliah Pemrograman Bahasa Tingkat Rendah dengan fokus pada Turbo Assembler. Buku ini mencakup 15 pertemuan yang dirancang untuk memberikan pemahaman komprehensif tentang pemrograman assembly pada mikroprosesor Intel 8086.

Penulis berharap buku ini dapat membantu mahasiswa dalam memahami konsep-konsep dasar pemrograman assembly dan mengembangkan kemampuan praktis dalam menggunakan Turbo Assembler.

Daftar Isi

Bab 1

Pengenalan Bahasa Rakitan dan Bahasa Tingkat Rendah; Sistem Bilangan (Biner, Heksadesimal)

1.1 Tujuan Pembelajaran

Setelah mengikuti pertemuan ini, mahasiswa diharapkan dapat:

- Memahami konsep bahasa rakitan dan bahasa tingkat rendah
- Mengetahui perbedaan antara bahasa tingkat tinggi dan bahasa tingkat rendah
- Memahami sistem bilangan biner dan heksadesimal
- Mampu melakukan konversi antar sistem bilangan

1.2 Materi Pembelajaran

1.2.1 Pengenalan Bahasa Rakitan dan Bahasa Tingkat Rendah

- Definisi bahasa rakitan (assembly language)
- Karakteristik bahasa tingkat rendah
- Perbandingan dengan bahasa tingkat tinggi
- Keunggulan dan kelemahan bahasa rakitan
- Aplikasi bahasa rakitan dalam pemrograman

1.2.2 Sistem Bilangan

- Sistem bilangan biner (basis 2)
- Sistem bilangan heksadesimal (basis 16)
- Konversi antar sistem bilangan
- Operasi aritmatika dalam sistem bilangan biner
- Representasi data dalam komputer

1.3 Contoh Soal dan Latihan

1. Konversikan bilangan desimal 255 ke biner dan heksadesimal
2. Konversikan bilangan biner 11010110 ke desimal dan heksadesimal
3. Konversikan bilangan heksadesimal A5F ke desimal dan biner
4. Lakukan operasi penjumlahan biner: 1011 + 1101

1.4 Tugas

- Buat tabel konversi bilangan 0-15 dalam format desimal, biner, dan heksadesimal
- Jelaskan mengapa sistem bilangan heksadesimal sering digunakan dalam pemrograman
- Berikan contoh aplikasi bahasa rakitan dalam kehidupan sehari-hari

1.5 Referensi

- Hyde, Randall. *The Art of Assembly Language*, 2nd ed., No Starch Press, 2010.
- Susanto. *Belajar Pemrograman Bahasa Assembly*, Elex Media Komputindo, 1995.

Bab 2

Arsitektur Mikroprosesor Intel 8086: Register, Segmentasi Memori, Mode Pengalamatan; Struktur File ASM

2.1 Tujuan Pembelajaran

Setelah mengikuti pertemuan ini, mahasiswa diharapkan dapat:

- Memahami arsitektur mikroprosesor Intel 8086
- Mengetahui fungsi dan penggunaan register-register utama
- Memahami konsep segmentasi memori
- Mengenal berbagai mode pengalamatan
- Memahami struktur file ASM

2.2 Materi Pembelajaran

2.2.1 Arsitektur Mikroprosesor Intel 8086

- Sejarah dan karakteristik Intel 8086
- Arsitektur internal mikroprosesor
- Bus sistem dan kontrol
- Mode operasi (real mode)

2.2.2 Register-Register Utama

- Register umum (AX, BX, CX, DX)
- Register indeks (SI, DI)
- Register pointer (SP, BP)
- Register segmen (CS, DS, SS, ES)

- Register flag (FLAGS)
- Register instruksi (IP)

2.2.3 Segmentasi Memori

- Konsep segmentasi memori
- Register segmen dan offset
- Perhitungan alamat fisik
- Batasan memori dalam mode real

2.2.4 Mode Pengalamatan

- Pengalamatan register
- Pengalamatan langsung
- Pengalamatan memori langsung
- Pengalamatan register tidak langsung
- Pengalamatan berbasis indeks
- Pengalamatan berbasis register

2.2.5 Struktur File ASM

- Format dasar file assembly
- Direktif assembler
- Komentar dalam kode
- Organisasi kode program

2.3 Contoh Soal dan Latihan

1. Jelaskan fungsi register AX, BX, CX, dan DX
2. Hitung alamat fisik jika CS = 2000h dan IP = 0100h
3. Berikan contoh penggunaan mode pengalamatan register
4. Tuliskan struktur dasar file ASM

2.4 Tugas

- Buat diagram arsitektur Intel 8086
- Jelaskan perbedaan antara register segmen dan register umum
- Berikan contoh penggunaan berbagai mode pengalamatan
- Buat template struktur file ASM untuk program sederhana

2.5 Referensi

- Brey, Barry B. *Mikroprosesor Intel 8086/8088 dsb.*, edisi terjemahan, Penerbit Informatika.
- Intel Corporation. *Intel 64 and IA-32 Architectures Software Developer's Manual*, Intel.

Bab 3

Instalasi Turbo Assembler dan Lingkungan Pengembangan; Struktur Program COM/ Direktif Dasar (ORG, END)

3.1 Tujuan Pembelajaran

Setelah mengikuti pertemuan ini, mahasiswa diharapkan dapat:

- Menginstal dan mengkonfigurasi Turbo Assembler
- Memahami lingkungan pengembangan Turbo Assembler
- Mengetahui perbedaan struktur program COM dan EXE
- Menggunakan direktif dasar ORG dan END
- Mampu membuat program assembly sederhana

3.2 Materi Pembelajaran

3.2.1 Instalasi Turbo Assembler

- Persyaratan sistem
- Proses instalasi
- Konfigurasi lingkungan
- Pengaturan path dan direktori

3.2.2 Lingkungan Pengembangan

- Editor Turbo Assembler
- Kompiler (TASM)
- Linker (TLINK)
- Debugger (TD)
- File bantuan dan dokumentasi

3.2.3 Struktur Program COM

- Karakteristik program COM
- Format file COM
- Batasan ukuran program
- Penggunaan memori
- Contoh struktur program COM

3.2.4 Struktur Program EXE

- Karakteristik program EXE
- Format file EXE
- Header file EXE
- Segmentasi program
- Contoh struktur program EXE

3.2.5 Direktif Dasar

- Direktif ORG (Origin)
- Direktif END
- Direktif lainnya (TITLE, PAGE, etc.)
- Penggunaan direktif dalam program

3.3 Praktikum

1. Instalasi Turbo Assembler
2. Konfigurasi lingkungan pengembangan
3. Membuat program "Hello World" sederhana
4. Kompilasi dan eksekusi program
5. Penggunaan direktif ORG dan END

3.4 Contoh Kode

```
; Program sederhana menggunakan direktif dasar
TITLE Program Sederhana
PAGE 60,132

.MODEL SMALL
.STACK 100h
```

```
.DATA
    pesan DB 'Hello World!$'

.CODE
    ORG 100h
    START:
        MOV AX, @DATA
        MOV DS, AX

        MOV DX, OFFSET pesan
        MOV AH, 09h
        INT 21h

        MOV AH, 4Ch
        INT 21h
    END START
```

3.5 Tugas

- Instal Turbo Assembler di komputer
- Buat program COM sederhana yang menampilkan nama Anda
- Jelaskan perbedaan antara program COM dan EXE
- Buat dokumentasi proses instalasi dan konfigurasi

3.6 Referensi

- Borland. *Turbo Assembler (TASM) User's Guide*, Borland International.
- Nopi, Arif. *Tutorial Bahasa Assembly (x86)*, Jasakom, Edisi Online.

Bab 4

Instruksi Dasar: Perpindahan Data (MOV), Operasi Aritmatika (ADD, SUB, MUL, DIV)

4.1 Tujuan Pembelajaran

Setelah mengikuti pertemuan ini, mahasiswa diharapkan dapat:

- Menggunakan instruksi MOV untuk perpindahan data
- Melakukan operasi aritmatika dengan ADD, SUB, MUL, DIV
- Memahami pengaruh operasi terhadap flag register
- Mampu menulis program dengan operasi aritmatika dasar

4.2 Materi Pembelajaran

4.2.1 Instruksi Perpindahan Data (MOV)

- Sintaks instruksi MOV
- Aturan penggunaan MOV
- Perpindahan data antar register
- Perpindahan data dari/ke memori
- Perpindahan data dengan konstanta
- Contoh penggunaan MOV

4.2.2 Operasi Aritmatika

- Instruksi ADD (penjumlahan)
- Instruksi SUB (pengurangan)
- Instruksi MUL (perkalian)

- Instruksi DIV (pembagian)
- Pengaruh terhadap flag register
- Penanganan overflow dan underflow

4.2.3 Flag Register dan Operasi Aritmatika

- Flag Carry (CF)
- Flag Zero (ZF)
- Flag Sign (SF)
- Flag Overflow (OF)
- Flag Parity (PF)
- Flag Auxiliary Carry (AF)

4.3 Praktikum

1. Program perpindahan data antar register
2. Program penjumlahan dua bilangan
3. Program pengurangan dengan penanganan flag
4. Program perkalian dan pembagian
5. Program kalkulator sederhana

4.4 Contoh Kode

```
; Program operasi aritmatika dasar
TITLE Operasi Aritmatika
.MODEL SMALL
.STACK 100h

.DATA
    bil1 DW 15
    bil2 DW 7
    hasil DW ?

.CODE
START:
    MOV AX, @DATA
    MOV DS, AX

    ; Penjumlahan
    MOV AX, bil1
    ADD AX, bil2
    MOV hasil, AX
```

```
    ; Pengurangan
    MOV AX, bil1
    SUB AX, bil2

    ; Perkalian
    MOV AX, bil1
    MUL bil2

    ; Pembagian
    MOV AX, bil1
    DIV bil2

    MOV AH, 4Ch
    INT 21h
END START
```

4.5 Latihan

1. Buat program yang menghitung (A + B) - (C + D)
2. Buat program yang menghitung (X * Y) / Z
3. Jelaskan pengaruh operasi aritmatika terhadap flag register
4. Buat program yang menampilkan status flag setelah operasi

4.6 Tugas

- Buat kalkulator sederhana dengan operasi +, -, *, /
- Implementasikan penanganan error untuk pembagian dengan nol
- Buat program yang menghitung rata-rata dari 5 bilangan
- Dokumentasikan penggunaan flag register dalam operasi aritmatika

4.7 Referensi

- Hyde, Randall. *The Art of Assembly Language*, 2nd ed., No Starch Press, 2010.
- Susanto. *Belajar Pemrograman Bahasa Assembly*, Elex Media Komputindo, 1995.

Bab 5

Instruksi Logika (AND, OR, XOR, NOT); Output Teks ke Layar (INT 10h fungsi 02h)

5.1 Tujuan Pembelajaran

Setelah mengikuti pertemuan ini, mahasiswa diharapkan dapat:

- Menggunakan instruksi logika AND, OR, XOR, NOT
- Memahami operasi bitwise dan penggunaannya
- Menggunakan interupsi INT 10h untuk output teks
- Mampu menampilkan teks di layar dengan kontrol posisi

5.2 Materi Pembelajaran

5.2.1 Instruksi Logika

- Instruksi AND (operasi logika AND)
- Instruksi OR (operasi logika OR)
- Instruksi XOR (operasi logika XOR)
- Instruksi NOT (operasi logika NOT)
- Operasi bitwise dan aplikasinya
- Pengaruh terhadap flag register

5.2.2 Aplikasi Instruksi Logika

- Manipulasi bit
- Masking dan unmasking
- Enkripsi sederhana

- Operasi set dan clear bit
- Perbandingan bit

5.2.3 Interupsi BIOS INT 10h

- Fungsi 02h: Set Cursor Position
- Fungsi 09h: Write Character and Attribute
- Fungsi 0Ah: Write Character Only
- Fungsi 0Eh: Write Character in TTY Mode
- Parameter dan penggunaan

5.2.4 Output Teks ke Layar

- Kontrol posisi kursor
- Atribut karakter (warna, background)
- Mode teks dan grafik
- Penanganan karakter khusus

5.3 Praktikum

1. Program operasi logika dasar
2. Program manipulasi bit
3. Program enkripsi sederhana dengan XOR
4. Program output teks dengan kontrol posisi
5. Program menampilkan teks berwarna

5.4 Contoh Kode

```
; Program operasi logika dan output teks
TITLE Operasi Logika dan Output
.MODEL SMALL
.STACK 100h

.DATA
    pesan DB 'Hello Assembly!$'
    nilai1 DW 0F0F0h
    nilai2 DW 0FF00h

.CODE
START:
    MOV AX, @DATA
```

```
    MOV DS, AX

    ; Operasi logika
    MOV AX, nilai1
    AND AX, nilai2    ; Masking
    OR AX, 000Fh      ; Set bit terakhir
    XOR AX, 0FFFFh    ; Inversi
    NOT AX            ; Komplemen

    ; Output teks ke layar
    MOV AH, 02h       ; Set cursor position
    MOV BH, 00h       ; Page 0
    MOV DH, 10        ; Row 10
    MOV DL, 20        ; Column 20
    INT 10h

    ; Tampilkan karakter
    MOV AH, 09h       ; Write character
    MOV AL, 'A'       ; Character 'A'
    MOV BH, 00h       ; Page 0
    MOV BL, 1Eh       ; Yellow on blue
    MOV CX, 1         ; Count
    INT 10h

    MOV AH, 4Ch
    INT 21h
END START
```

5.5 Latihan

1. Buat program yang melakukan masking pada 8 bit terakhir

2. Buat program enkripsi sederhana menggunakan XOR

3. Buat program yang menampilkan teks di berbagai posisi layar

4. Buat program yang menampilkan teks dengan berbagai warna

5.6 Tugas

- Implementasikan operasi logika untuk manipulasi data
- Buat program yang menampilkan menu dengan kontrol posisi
- Buat program enkripsi/dekripsi teks menggunakan XOR
- Dokumentasikan penggunaan interupsi INT 10h

5.7 Referensi

- Hyde, Randall. *The Art of Assembly Language*, 2nd ed., No Starch Press, 2010.
- Partoharsojo, Hartono. *Tuntunan Praktis Pemrograman Assembly*, Penerbit Informatika.

Bab 6

Input dari Keyboard (INT 16h); Penanganan Keyboard Buffer

6.1 Tujuan Pembelajaran

Setelah mengikuti pertemuan ini, mahasiswa diharapkan dapat:

- Menggunakan interupsi INT 16h untuk input keyboard
- Memahami fungsi-fungsi INT 16h
- Menangani keyboard buffer
- Mampu membuat program interaktif dengan input keyboard

6.2 Materi Pembelajaran

6.2.1 Interupsi BIOS INT 16h

- Fungsi 00h: Read Key (Blocking)
- Fungsi 01h: Check for Key (Non-blocking)
- Fungsi 02h: Get Keyboard Flags
- Parameter dan return values
- Perbedaan blocking dan non-blocking

6.2.2 Keyboard Buffer

- Konsep keyboard buffer
- Cara kerja buffer keyboard
- Penanganan buffer penuh
- Flushing keyboard buffer
- Status keyboard buffer

6.2.3 Input Karakter dan String

- Input karakter tunggal
- Input string dengan echo
- Input string tanpa echo (password)
- Validasi input
- Penanganan karakter khusus

6.2.4 Karakter Khusus

- Scan code dan ASCII code
- Function keys (F1-F12)
- Arrow keys
- Control keys (Ctrl, Alt, Shift)
- Special keys (Enter, Escape, Backspace)

6.3 Praktikum

1. Program input karakter tunggal
2. Program input string dengan echo
3. Program input password (tanpa echo)
4. Program penanganan keyboard buffer
5. Program interaktif dengan menu

6.4 Contoh Kode

```
; Program input keyboard dan penanganan buffer
TITLE Input Keyboard
.MODEL SMALL
.STACK 100h

.DATA
    prompt DB 'Masukkan nama: $'
    nama DB 50 DUP('$')
    pesan DB 'Halo, $'
    buffer DB 50 DUP(?)

.CODE
START:
    MOV AX, @DATA
    MOV DS, AX
```

```
    ; Tampilkan prompt
    MOV AH, 09h
    MOV DX, OFFSET prompt
    INT 21h

    ; Input string dengan echo
    MOV AH, 0Ah
    MOV DX, OFFSET buffer
    INT 21h

    ; Input karakter tunggal (blocking)
    MOV AH, 00h
    INT 16h
    ; AL = ASCII code, AH = Scan code

    ; Check for key (non-blocking)
    MOV AH, 01h
    INT 16h
    JZ no_key
    ; Key available, read it
    MOV AH, 00h
    INT 16h

no_key:
    ; Flush keyboard buffer
    MOV AH, 0Ch
    MOV AL, 00h
    INT 21h

    MOV AH, 4Ch
    INT 21h
END START
```

6.5 Latihan

1. Buat program yang menunggu input dari keyboard
2. Buat program yang menampilkan scan code dari key yang ditekan
3. Buat program input password dengan masking karakter
4. Buat program yang menangani keyboard buffer penuh

6.6 Tugas

- Buat program login dengan validasi username dan password
- Implementasikan menu interaktif dengan navigasi menggunakan arrow keys
- Buat program yang menangani input dengan timeout

- Dokumentasikan penggunaan interupsi INT 16h

6.7 Referensi

- Hyde, Randall. *The Art of Assembly Language*, 2nd ed., No Starch Press, 2010.
- Susanto. *Belajar Pemrograman Bahasa Assembly*, Elex Media Komputindo, 1995.

Bab 7

Percabangan dan Loop: CMP, JE, JNE, JL, JG, LOOP; Interupsi Mouse (INT 33h)

7.1 Tujuan Pembelajaran

Setelah mengikuti pertemuan ini, mahasiswa diharapkan dapat:

- Menggunakan instruksi CMP untuk perbandingan
- Menggunakan instruksi percabangan (JE, JNE, JL, JG)
- Menggunakan instruksi LOOP untuk perulangan
- Memahami interupsi mouse INT 33h
- Mampu membuat program dengan struktur kontrol

7.2 Materi Pembelajaran

7.2.1 Instruksi Perbandingan (CMP)

- Sintaks instruksi CMP
- Cara kerja CMP
- Pengaruh terhadap flag register
- Perbandingan dengan operasi aritmatika

7.2.2 Instruksi Percabangan

- Instruksi JE (Jump if Equal)
- Instruksi JNE (Jump if Not Equal)
- Instruksi JL (Jump if Less)
- Instruksi JG (Jump if Greater)

- Instruksi lainnya (JLE, JGE, JB, JA, etc.)
- Conditional jumps dan flag register

7.2.3 Instruksi Perulangan (LOOP)

- Instruksi LOOP
- Instruksi LOOPZ/LOOPE
- Instruksi LOOPNZ/LOOPNE
- Penggunaan register CX
- Kontrol perulangan

7.2.4 Interupsi Mouse INT 33h

- Fungsi 00h: Initialize Mouse
- Fungsi 01h: Show Mouse Cursor
- Fungsi 02h: Hide Mouse Cursor
- Fungsi 03h: Get Mouse Position and Button Status
- Fungsi 04h: Set Mouse Cursor Position
- Fungsi 05h: Get Button Press Information

7.3 Praktikum

1. Program percabangan sederhana
2. Program perulangan dengan LOOP
3. Program kombinasi percabangan dan perulangan
4. Program interaksi dengan mouse
5. Program game sederhana dengan mouse

7.4 Contoh Kode

```
; Program percabangan, loop, dan mouse
TITLE Percabangan dan Loop
.MODEL SMALL
.STACK 100h

.DATA
    pesan1 DB 'Bilangan sama$'
    pesan2 DB 'Bilangan berbeda$'
    pesan3 DB 'Loop selesai$'
    bil1 DW 10
```

```
    bil2 DW 10

.CODE
START:
    MOV AX, @DATA
    MOV DS, AX

    ; Percabangan
    MOV AX, bil1
    CMP AX, bil2
    JE sama
    ; Jika tidak sama
    MOV AH, 09h
    MOV DX, OFFSET pesan2
    INT 21h
    JMP lanjut

sama:
    MOV AH, 09h
    MOV DX, OFFSET pesan1
    INT 21h

lanjut:
    ; Perulangan
    MOV CX, 5
ulang:
    ; Kode yang diulang
    DEC CX
    JNZ ulang

    ; Inisialisasi mouse
    MOV AX, 00h
    INT 33h
    CMP AX, 0
    JE no_mouse

    ; Tampilkan cursor mouse
    MOV AX, 01h
    INT 33h

    ; Baca posisi mouse
    MOV AX, 03h
    INT 33h
    ; BX = button status, CX = X position, DX = Y position

no_mouse:
    MOV AH, 4Ch
    INT 21h
END START
```

7.5 Latihan

1. Buat program yang membandingkan dua bilangan
2. Buat program yang menghitung faktorial menggunakan LOOP
3. Buat program yang menampilkan angka 1-10 menggunakan perulangan
4. Buat program yang merespon klik mouse

7.6 Tugas

- Buat program kalkulator dengan menu percabangan
- Implementasikan algoritma sorting sederhana dengan perulangan
- Buat program game tebak angka dengan mouse
- Dokumentasikan penggunaan instruksi percabangan dan perulangan

7.7 Referensi

- Hyde, Randall. *The Art of Assembly Language*, 2nd ed., No Starch Press, 2010.
- Partoharsojo, Hartono. *Tuntunan Praktis Pemrograman Assembly*, Penerbit Informatika.

Bab 8

Instruksi Stack: PUSH, POP, CALL, RET; Subrutin dan Parameter Sederhana

8.1 Tujuan Pembelajaran

Setelah mengikuti pertemuan ini, mahasiswa diharapkan dapat:

- Memahami konsep stack dan operasinya
- Menggunakan instruksi PUSH dan POP
- Menggunakan instruksi CALL dan RET
- Membuat subrutin dengan parameter sederhana
- Mampu memodularisasi program

8.2 Materi Pembelajaran

8.2.1 Konsep Stack

- Definisi stack (tumpukan)
- Prinsip LIFO (Last In, First Out)
- Register stack pointer (SP)
- Register stack segment (SS)
- Operasi stack dalam assembly

8.2.2 Instruksi Stack

- Instruksi PUSH (memasukkan data ke stack)
- Instruksi POP (mengambil data dari stack)
- PUSH/POP register

- PUSH/POP memori
- PUSH/POP flag register
- PUSH/POP immediate value

8.2.3 Instruksi Subrutin

- Instruksi CALL (memanggil subrutin)
- Instruksi RET (kembali dari subrutin)
- CALL near dan CALL far
- RET dengan parameter
- Penanganan return address

8.2.4 Parameter dalam Subrutin

- Parameter melalui register
- Parameter melalui stack
- Parameter melalui memori
- Return value dari subrutin
- Konvensi pemanggilan

8.3 Praktikum

1. Program demonstrasi operasi stack
2. Program subrutin sederhana
3. Program subrutin dengan parameter
4. Program subrutin dengan return value
5. Program modular dengan multiple subrutin

8.4 Contoh Kode

```
; Program demonstrasi stack dan subrutin
TITLE Stack dan Subrutin
.MODEL SMALL
.STACK 100h

.DATA
    pesan1 DB 'Hello from main!$'
    pesan2 DB 'Hello from subrutin!$'
    nilai1 DW 10
    nilai2 DW 20
```

```
    hasil DW ?

.CODE
START:
    MOV AX, @DATA
    MOV DS, AX

    ; Demonstrasi stack
    MOV AX, 1234h
    PUSH AX
    MOV BX, 5678h
    PUSH BX

    ; Ambil dari stack (urutan terbalik)
    POP CX  ; CX = 5678h
    POP DX  ; DX = 1234h

    ; Panggil subrutin
    CALL tampilkan_pesan

    ; Subrutin dengan parameter
    PUSH nilai1
    PUSH nilai2
    CALL tambahkan
    ADD SP, 4  ; Bersihkan parameter dari stack

    MOV AH, 4Ch
    INT 21h

; Subrutin tanpa parameter
tampilkan_pesan PROC
    PUSH AX
    PUSH DX

    MOV AH, 09h
    MOV DX, OFFSET pesan2
    INT 21h

    POP DX
    POP AX
    RET
tampilkan_pesan ENDP

; Subrutin dengan parameter
tambahkan PROC
    PUSH BP
    MOV BP, SP

    PUSH AX
    PUSH BX
```

```
    MOV AX, [BP+6]  ; Parameter kedua
    MOV BX, [BP+4]  ; Parameter pertama
    ADD AX, BX
    MOV hasil, AX

    POP BX
    POP AX
    POP BP
    RET
tambahkan ENDP

END START
```

8.5 Latihan

1. Buat program yang menggunakan stack untuk menyimpan dan mengambil data
2. Buat subrutin yang menghitung kuadrat dari sebuah bilangan
3. Buat subrutin yang menukar nilai dua variabel
4. Buat program yang menggunakan multiple subrutin

8.6 Tugas

- Implementasikan subrutin untuk operasi aritmatika dasar
- Buat program kalkulator modular menggunakan subrutin
- Implementasikan subrutin untuk manipulasi string
- Dokumentasikan penggunaan stack dan subrutin

8.7 Referensi

- Hyde, Randall. *The Art of Assembly Language*, 2nd ed., No Starch Press, 2010.
- Susanto. *Belajar Pemrograman Bahasa Assembly*, Elex Media Komputindo, 1995.

Bab 9

Array dan String: Penyimpanan Array, Instruksi String (MOVS, CMPS, SCAS, LODS, STOS)

9.1 Tujuan Pembelajaran

Setelah mengikuti pertemuan ini, mahasiswa diharapkan dapat:

- Memahami konsep array dan string dalam assembly
- Menyimpan dan mengakses data array
- Menggunakan instruksi string MOVS, CMPS, SCAS, LODS, STOS
- Mampu memanipulasi string dan array
- Mengoptimalkan operasi string dengan prefix REP

9.2 Materi Pembelajaran

9.2.1 Konsep Array dan String

- Definisi array dan string
- Penyimpanan array dalam memori
- Pengalamatan elemen array
- String sebagai array karakter
- Null-terminated string

9.2.2 Penyimpanan Array

- Array satu dimensi
- Array multi dimensi
- Pengalamatan array dengan indeks

- Perhitungan offset elemen array
- Array dengan ukuran berbeda

9.2.3 Instruksi String

- Instruksi MOVS (Move String)
- Instruksi CMPS (Compare String)
- Instruksi SCAS (Scan String)
- Instruksi LODS (Load String)
- Instruksi STOS (Store String)
- Penggunaan prefix REP

9.2.4 Register untuk Operasi String

- SI (Source Index)
- DI (Destination Index)
- CX (Counter)
- AL/AX (Accumulator)
- Direction Flag (DF)

9.3 Praktikum

1. Program demonstrasi operasi array
2. Program copy string menggunakan MOVS
3. Program perbandingan string menggunakan CMPS
4. Program pencarian karakter menggunakan SCAS
5. Program manipulasi string lengkap

9.4 Contoh Kode

```
; Program demonstrasi array dan string
TITLE Array dan String
.MODEL SMALL
.STACK 100h

.DATA
    array DW 10, 20, 30, 40, 50
    string1 DB 'Hello World$'
    string2 DB 20 DUP('$')
    string3 DB 'Assembly$'
```

```
    string4 DB 'Assembly$'
    karakter DB 'l'
    panjang EQU $ - string1

.CODE
START:
    MOV AX, @DATA
    MOV DS, AX
    MOV ES, AX

    ; Akses elemen array
    MOV BX, 2        ; Indeks elemen ke-2
    MOV AX, array[BX] ; AX = 30

    ; Copy string menggunakan MOVS
    LEA SI, string1
    LEA DI, string2
    MOV CX, panjang
    CLD              ; Clear direction flag
    REP MOVSB

    ; Perbandingan string menggunakan CMPS
    LEA SI, string3
    LEA DI, string4
    MOV CX, 8
    CLD
    REPE CMPSB
    JE sama
    ; String berbeda
    JMP lanjut

sama:
    ; String sama

lanjut:
    ; Pencarian karakter menggunakan SCAS
    LEA DI, string1
    MOV AL, karakter
    MOV CX, panjang
    CLD
    REPNE SCASB
    JE ditemukan
    ; Karakter tidak ditemukan
    JMP selesai

ditemukan:
    ; Karakter ditemukan di posisi CX

selesai:
    ; Load string menggunakan LODS
```

```
    LEA SI, string1
    MOV CX, 5
    CLD
    LODSB  ; Load byte ke AL

    ; Store string menggunakan STOS
    LEA DI, string2
    MOV AL, 'X'
    MOV CX, 5
    CLD
    REP STOSB

    MOV AH, 4Ch
    INT 21h
END START
```

9.5 Latihan

1. Buat program yang mengakses elemen array dengan indeks
2. Buat program yang menyalin string dari satu lokasi ke lokasi lain
3. Buat program yang membandingkan dua string
4. Buat program yang mencari karakter dalam string

9.6 Tugas

- Implementasikan fungsi untuk menghitung panjang string
- Buat program yang membalik string
- Implementasikan fungsi untuk menggabungkan dua string
- Buat program yang mengurutkan array bilangan

9.7 Referensi

- Hyde, Randall. *The Art of Assembly Language*, 2nd ed., No Starch Press, 2010.
- Partoharsojo, Hartono. *Tuntunan Praktis Pemrograman Assembly*, Penerbit Informatika.

Bab 10

Pemrograman Modular: PROC/ENDP, Penggunaan Makro Sederhana (MACRO)

10.1 Tujuan Pembelajaran

Setelah mengikuti pertemuan ini, mahasiswa diharapkan dapat:

- Memahami konsep pemrograman modular
- Menggunakan PROC/ENDP untuk membuat prosedur
- Membuat dan menggunakan makro sederhana
- Membedakan antara prosedur dan makro
- Mampu merancang program modular

10.2 Materi Pembelajaran

10.2.1 Konsep Pemrograman Modular

- Definisi pemrograman modular
- Keuntungan pemrograman modular
- Prinsip modularitas
- Reusability dan maintainability
- Organisasi kode program

10.2.2 Prosedur dengan PROC/ENDP

- Sintaks PROC/ENDP
- Deklarasi prosedur
- Parameter prosedur
- Local variables dalam prosedur
- Return value dari prosedur
- Konvensi pemanggilan prosedur

10.2.3 Makro (MACRO)

- Definisi makro
- Sintaks MACRO/ENDM
- Parameter makro
- Makro dengan multiple parameters
- Makro bersyarat
- Perbedaan makro dan prosedur

10.2.4 Organisasi Program Modular

- Struktur file program
- Include files
- Library routines
- Module dependencies
- Code organization

10.3 Praktikum

1. Program dengan prosedur sederhana
2. Program dengan prosedur berparameter
3. Program dengan makro sederhana
4. Program dengan makro berparameter
5. Program modular lengkap

10.4 Contoh Kode

```
; Program demonstrasi prosedur dan makro
TITLE Pemrograman Modular
.MODEL SMALL
.STACK 100h

.DATA
    pesan1 DB 'Hello from procedure!$'
    pesan2 DB 'Hello from macro!$'
    nilai1 DW 15
    nilai2 DW 25
    hasil DW ?

; Makro untuk menampilkan pesan
tampilkan_pesan MACRO pesan
```

```
    PUSH AX
    PUSH DX
    MOV AH, 09h
    MOV DX, OFFSET pesan
    INT 21h
    POP DX
    POP AX
ENDM

; Makro untuk pertukaran nilai
tukar_nilai MACRO var1, var2
    PUSH AX
    MOV AX, var1
    XCHG AX, var2
    MOV var1, AX
    POP AX
ENDM

.CODE
START:
    MOV AX, @DATA
    MOV DS, AX

    ; Panggil prosedur
    CALL tampilkan_hello

    ; Gunakan makro
    tampilkan_pesan pesan2

    ; Pertukaran nilai menggunakan makro
    tukar_nilai nilai1, nilai2

    ; Panggil prosedur dengan parameter
    PUSH nilai1
    PUSH nilai2
    CALL tambahkan
    ADD SP, 4

    MOV AH, 4Ch
    INT 21h

; Prosedur tanpa parameter
tampilkan_hello PROC
    PUSH AX
    PUSH DX

    MOV AH, 09h
    MOV DX, OFFSET pesan1
    INT 21h
```

```
    POP DX
    POP AX
    RET
tampilkan_hello ENDP

; Prosedur dengan parameter
tambahkan PROC
    PUSH BP
    MOV BP, SP

    PUSH AX
    PUSH BX

    MOV AX, [BP+6]  ; Parameter kedua
    MOV BX, [BP+4]  ; Parameter pertama
    ADD AX, BX
    MOV hasil, AX

    POP BX
    POP AX
    POP BP
    RET
tambahkan ENDP

END START
```

10.5 Latihan

1. Buat prosedur untuk menghitung faktorial
2. Buat makro untuk menampilkan karakter
3. Buat prosedur untuk mengurutkan array
4. Buat makro untuk operasi aritmatika

10.6 Tugas

- Implementasikan library prosedur untuk operasi string
- Buat makro untuk debugging dan logging
- Implementasikan program kalkulator modular
- Dokumentasikan perbedaan antara prosedur dan makro

10.7 Referensi

- Hyde, Randall. *The Art of Assembly Language*, 2nd ed., No Starch Press, 2010.
- Borland. *Turbo Assembler (TASM) User's Guide*, Borland International.

Bab 11

Pemrograman Grafik Dasar: Menggambar Piksel dan Garis (Mode Grafik INT 10h)

11.1 Tujuan Pembelajaran

Setelah mengikuti pertemuan ini, mahasiswa diharapkan dapat:

- Memahami konsep mode grafik
- Menggunakan interupsi INT 10h untuk mode grafik
- Menggambar piksel di layar
- Menggambar garis menggunakan algoritma
- Mampu membuat program grafik sederhana

11.2 Materi Pembelajaran

11.2.1 Konsep Mode Grafik

- Perbedaan mode teks dan grafik
- Resolusi layar
- Color palette
- Video memory organization
- Pixel addressing

11.2.2 Interupsi INT 10h untuk Grafik

- Fungsi 00h: Set Video Mode
- Fungsi 0Ch: Write Pixel
- Fungsi 0Dh: Read Pixel

- Fungsi 0Fh: Get Video Mode
- Mode grafik yang tersedia

11.2.3 Menggambar Piksel

- Koordinat piksel (X, Y)
- Color value
- Perhitungan alamat memori video
- Optimasi akses memori video

11.2.4 Algoritma Menggambar Garis

- Algoritma DDA (Digital Differential Analyzer)
- Algoritma Bresenham
- Implementasi algoritma garis
- Optimasi algoritma

11.2.5 Koordinat dan Transformasi

- Sistem koordinat layar
- Transformasi koordinat
- Clipping
- Viewport

11.3 Praktikum

1. Program mengatur mode grafik
2. Program menggambar piksel tunggal
3. Program menggambar garis horizontal dan vertikal
4. Program menggambar garis diagonal
5. Program menggambar bentuk geometri sederhana

11.4 Contoh Kode

```
; Program demonstrasi grafik dasar
TITLE Pemrograman Grafik
.MODEL SMALL
.STACK 100h

.DATA
    x1 DW 100
    y1 DW 100
    x2 DW 200
    y2 DW 150
    color DB 15  ; Putih

.CODE
START:
    MOV AX, @DATA
    MOV DS, AX

    ; Set mode grafik 320x200, 256 warna
    MOV AH, 00h
    MOV AL, 13h
    INT 10h

    ; Gambar piksel tunggal
    MOV AH, 0Ch
    MOV AL, color
    MOV BH, 00h
    MOV CX, 160  ; X coordinate
    MOV DX, 100  ; Y coordinate
    INT 10h

    ; Gambar garis horizontal
    MOV CX, 50   ; X start
    MOV DX, 50   ; Y coordinate
    MOV BX, 100  ; X end

gambar_horizontal:
    MOV AH, 0Ch
    MOV AL, color
    MOV BH, 00h
    INT 10h
    INC CX
    CMP CX, BX
    JLE gambar_horizontal

    ; Gambar garis vertikal
    MOV CX, 50   ; X coordinate
    MOV DX, 50   ; Y start
    MOV BX, 100  ; Y end
```

```
gambar_vertikal:
    MOV AH, 0Ch
    MOV AL, color
    MOV BH, 00h
    INT 10h
    INC DX
    CMP DX, BX
    JLE gambar_vertikal

    ; Gambar garis diagonal menggunakan DDA
    CALL gambar_garis_dda

    ; Tunggu input keyboard
    MOV AH, 00h
    INT 16h

    ; Kembali ke mode teks
    MOV AH, 00h
    MOV AL, 03h
    INT 10h

    MOV AH, 4Ch
    INT 21h

; Prosedur menggambar garis menggunakan DDA
gambar_garis_dda PROC
    PUSH AX
    PUSH BX
    PUSH CX
    PUSH DX

    MOV AX, x2
    SUB AX, x1
    MOV BX, y2
    SUB BX, y1

    ; Hitung jumlah steps
    CMP AX, BX
    JGE steps_x
    MOV CX, BX
    JMP hitung_delta
steps_x:
    MOV CX, AX

hitung_delta:
    ; Delta X dan Delta Y
    MOV DX, AX
    SAR DX, 8  ; Delta X = (x2-x1)/steps
    MOV AX, BX
```

```
    SAR AX, 8  ; Delta Y = (y2-y1)/steps

    ; Inisialisasi
    MOV BX, x1
    MOV DX, y1

gambar_loop:
    PUSH AX
    PUSH BX
    PUSH CX
    PUSH DX

    MOV AH, 0Ch
    MOV AL, color
    MOV BH, 00h
    MOV CX, BX  ; X
    MOV DX, DX  ; Y
    INT 10h

    POP DX
    POP CX
    POP BX
    POP AX

    ADD BX, DX  ; X += Delta X
    ADD DX, AX  ; Y += Delta Y

    LOOP gambar_loop

    POP DX
    POP CX
    POP BX
    POP AX
    RET
gambar_garis_dda ENDP

END START
```

11.5 Latihan

1. Buat program yang menggambar kotak

2. Buat program yang menggambar segitiga

3. Buat program yang menggambar lingkaran

4. Buat program yang menggambar pola geometri

11.6 Tugas

- Implementasikan algoritma Bresenham untuk menggambar garis
- Buat program yang menggambar grafik fungsi matematika
- Implementasikan program paint sederhana
- Dokumentasikan perbedaan antara algoritma DDA dan Bresenham

11.7 Referensi

- Hyde, Randall. *The Art of Assembly Language*, 2nd ed., No Starch Press, 2010.
- Partoharsojo, Hartono. *Tuntunan Praktis Pemrograman Assembly*, Penerbit Informatika.

Bab 12

Pemrosesan File Dasar: Membuka, Menutup, Membaca, Menulis File (INT 21h fungsi 3Ch/3Fh/40h)

12.1 Tujuan Pembelajaran

Setelah mengikuti pertemuan ini, mahasiswa diharapkan dapat:

- Memahami konsep file handling
- Menggunakan interupsi INT 21h untuk operasi file
- Membuka dan menutup file
- Membaca dan menulis data ke file
- Menangani error dalam operasi file

12.2 Materi Pembelajaran

12.2.1 Konsep File Handling

- Definisi file dan file system
- File handle dan file descriptor
- Mode akses file (read, write, append)
- File attributes dan permissions
- Error handling dalam operasi file

12.2.2 Interupsi INT 21h untuk File

- Fungsi 3Ch: Create File
- Fungsi 3Dh: Open File
- Fungsi 3Eh: Close File

- Fungsi 3Fh: Read File
- Fungsi 40h: Write File
- Fungsi 41h: Delete File

12.2.3 Operasi File Dasar

- Membuat file baru
- Membuka file yang sudah ada
- Menutup file
- Membaca data dari file
- Menulis data ke file
- Menghapus file

12.2.4 File Handle dan Error Handling

- File handle management
- Error codes dan penanganannya
- Validasi operasi file
- Resource cleanup

12.2.5 Buffer dan Data Transfer

- File buffer
- Block transfer
- Sequential access
- Data formatting

12.3 Praktikum

1. Program membuat file baru
2. Program menulis data ke file
3. Program membaca data dari file
4. Program menyalin file
5. Program dengan error handling

12.4 Contoh Kode

```
; Program demonstrasi operasi file dasar
TITLE Pemrosesan File
.MODEL SMALL
.STACK 100h

.DATA
    nama_file DB 'test.txt', 0
    data_tulis DB 'Hello, World!', 13, 10
    data_baca DB 100 DUP(?)
    file_handle DW ?
    pesan_sukses DB 'File berhasil dibuat!$'
    pesan_error DB 'Error dalam operasi file!$'
    pesan_baca DB 'Data dari file: $'

.CODE
START:
    MOV AX, @DATA
    MOV DS, AX

    ; Buat file baru
    MOV AH, 3Ch
    MOV CX, 0        ; File attributes (normal)
    MOV DX, OFFSET nama_file
    INT 21h
    JC error_handler
    MOV file_handle, AX

    ; Tampilkan pesan sukses
    MOV AH, 09h
    MOV DX, OFFSET pesan_sukses
    INT 21h

    ; Tulis data ke file
    MOV AH, 40h
    MOV BX, file_handle
    MOV CX, 15       ; Jumlah byte yang ditulis
    MOV DX, OFFSET data_tulis
    INT 21h
    JC error_handler

    ; Tutup file
    MOV AH, 3Eh
    MOV BX, file_handle
    INT 21h
    JC error_handler

    ; Buka file untuk dibaca
    MOV AH, 3Dh
```

```
    MOV AL, 0        ; Mode read
    MOV DX, OFFSET nama_file
    INT 21h
    JC error_handler
    MOV file_handle, AX

    ; Baca data dari file
    MOV AH, 3Fh
    MOV BX, file_handle
    MOV CX, 100      ; Jumlah byte yang dibaca
    MOV DX, OFFSET data_baca
    INT 21h
    JC error_handler

    ; Tampilkan data yang dibaca
    MOV AH, 09h
    MOV DX, OFFSET pesan_baca
    INT 21h

    ; Tambahkan null terminator
    MOV BX, OFFSET data_baca
    ADD BX, AX
    MOV BYTE PTR [BX], '$'

    MOV AH, 09h
    MOV DX, OFFSET data_baca
    INT 21h

    ; Tutup file
    MOV AH, 3Eh
    MOV BX, file_handle
    INT 21h

    JMP selesai

error_handler:
    MOV AH, 09h
    MOV DX, OFFSET pesan_error
    INT 21h

selesai:
    MOV AH, 4Ch
    INT 21h
END START
```

12.5 Latihan

1. Buat program yang menulis teks ke file
2. Buat program yang membaca dan menampilkan isi file

3. Buat program yang menyalin isi file ke file lain
4. Buat program yang menghitung jumlah karakter dalam file

12.6 Tugas

- Implementasikan program text editor sederhana
- Buat program yang menyimpan dan memuat data konfigurasi
- Implementasikan program backup file
- Dokumentasikan error codes dan penanganannya

12.7 Referensi

- Hyde, Randall. *The Art of Assembly Language*, 2nd ed., No Starch Press, 2010.
- Susanto. *Belajar Pemrograman Bahasa Assembly*, Elex Media Komputindo, 1995.

Bab 13

Pemrosesan File Lanjutan: Manipulasi Pointer (INT 21h fungsi 42h), Penyimpanan Data Terformat

13.1 Tujuan Pembelajaran

Setelah mengikuti pertemuan ini, mahasiswa diharapkan dapat:

- Menggunakan fungsi 42h untuk manipulasi file pointer
- Memahami konsep random access file
- Menyimpan dan memuat data terformat
- Mengimplementasikan database sederhana
- Mampu membuat program file management

13.2 Materi Pembelajaran

13.2.1 Manipulasi File Pointer

- Konsep file pointer dan position
- Fungsi 42h: LSEEK (Set File Pointer)
- Mode seek (beginning, current, end)
- Random access file
- File positioning dan navigation

13.2.2 Fungsi LSEEK (42h)

- Sintaks fungsi 42h
- Parameter AL (seek mode)
- Parameter BX (file handle)

- Parameter CX:DX (offset)
- Return value dalam DX:AX

13.2.3 Data Terformat

- Konsep data terformat
- Record-based data
- Fixed-length records
- Variable-length records
- Data serialization

13.2.4 Database Sederhana

- Struktur record
- Index file
- CRUD operations (Create, Read, Update, Delete)
- Data validation
- File locking

13.2.5 Optimasi File Access

- Buffer management
- Caching strategies
- Batch operations
- Error recovery

13.3 Praktikum

1. Program demonstrasi LSEEK
2. Program random access file
3. Program penyimpanan data terformat
4. Program database sederhana
5. Program file management

13.4 Contoh Kode

```
; Program demonstrasi manipulasi file pointer dan data terformat
TITLE Pemrosesan File Lanjutan
.MODEL SMALL
.STACK 100h

.DATA
    nama_file DB 'data.dat', 0
    file_handle DW ?

    ; Struktur record
    record_size EQU 20
    nama DB 15 DUP(' ')
    umur DB ?
    gaji DW ?

    ; Data untuk disimpan
    data1 DB 'John Doe       ', 25, 1000
    data2 DB 'Jane Smith     ', 30, 1500
    data3 DB 'Bob Johnson    ', 35, 2000

    ; Buffer untuk membaca
    buffer DB record_size DUP(?)

    pesan_sukses DB 'Operasi berhasil!$'
    pesan_error DB 'Error dalam operasi!$'

.CODE
START:
    MOV AX, @DATA
    MOV DS, AX

    ; Buat file baru
    MOV AH, 3Ch
    MOV CX, 0
    MOV DX, OFFSET nama_file
    INT 21h
    JC error_handler
    MOV file_handle, AX

    ; Tulis record pertama
    MOV AH, 40h
    MOV BX, file_handle
    MOV CX, record_size
    MOV DX, OFFSET data1
    INT 21h
    JC error_handler

    ; Tulis record kedua
```

```
MOV AH, 40h
MOV BX, file_handle
MOV CX, record_size
MOV DX, OFFSET data2
INT 21h
JC error_handler

; Tulis record ketiga
MOV AH, 40h
MOV BX, file_handle
MOV CX, record_size
MOV DX, OFFSET data3
INT 21h
JC error_handler

; Baca record kedua (random access)
MOV AH, 42h
MOV AL, 0        ; Seek from beginning
MOV BX, file_handle
MOV CX, 0
MOV DX, record_size  ; Offset ke record kedua
INT 21h
JC error_handler

; Baca record
MOV AH, 3Fh
MOV BX, file_handle
MOV CX, record_size
MOV DX, OFFSET buffer
INT 21h
JC error_handler

; Update record kedua
MOV AH, 42h
MOV AL, 1        ; Seek from current position
MOV BX, file_handle
MOV CX, 0
MOV DX, -record_size  ; Kembali ke awal record
INT 21h
JC error_handler

; Modifikasi data
MOV buffer[15], 31    ; Update umur
MOV WORD PTR buffer[16], 1600  ; Update gaji

; Tulis record yang dimodifikasi
MOV AH, 40h
MOV BX, file_handle
MOV CX, record_size
MOV DX, OFFSET buffer
```

```
    INT 21h
    JC error_handler

    ; Baca semua record
    MOV AH, 42h
    MOV AL, 0        ; Seek to beginning
    MOV BX, file_handle
    MOV CX, 0
    MOV DX, 0
    INT 21h
    JC error_handler

    ; Baca dan tampilkan semua record
    MOV CX, 3        ; Jumlah record
baca_loop:
    PUSH CX

    MOV AH, 3Fh
    MOV BX, file_handle
    MOV CX, record_size
    MOV DX, OFFSET buffer
    INT 21h
    JC error_handler

    ; Tampilkan nama (15 karakter pertama)
    MOV buffer[15], '$'
    MOV AH, 09h
    MOV DX, OFFSET buffer
    INT 21h

    ; Tampilkan newline
    MOV AH, 02h
    MOV DL, 13
    INT 21h
    MOV DL, 10
    INT 21h

    POP CX
    LOOP baca_loop

    ; Tutup file
    MOV AH, 3Eh
    MOV BX, file_handle
    INT 21h

    JMP selesai

error_handler:
    MOV AH, 09h
    MOV DX, OFFSET pesan_error
```

```
    INT 21h

selesai:
    MOV AH, 4Ch
    INT 21h
END START
```

13.5 Latihan

1. Buat program yang membaca record tertentu dari file
2. Buat program yang mengupdate record di posisi tertentu
3. Buat program yang menghapus record (mark as deleted)
4. Buat program yang mencari record berdasarkan kriteria

13.6 Tugas

- Implementasikan program address book dengan file
- Buat program inventory management sederhana
- Implementasikan program student database
- Dokumentasikan teknik optimasi file access

13.7 Referensi

- Hyde, Randall. *The Art of Assembly Language*, 2nd ed., No Starch Press, 2010.
- Partoharsojo, Hartono. *Tuntunan Praktis Pemrograman Assembly*, Penerbit Informatika.

Bab 14

Program Residen (TSR): Konsep Terminate-and-Stay-Resident; Contoh Implementasi Sederhana

14.1 Tujuan Pembelajaran

Setelah mengikuti pertemuan ini, mahasiswa diharapkan dapat:

- Memahami konsep program TSR (Terminate-and-Stay-Resident)
- Mengetahui cara kerja program residen
- Mengimplementasikan TSR sederhana
- Menggunakan interupsi untuk komunikasi TSR
- Mampu membuat program background

14.2 Materi Pembelajaran

14.2.1 Konsep Program TSR

- Definisi TSR (Terminate-and-Stay-Resident)
- Perbedaan program normal dan TSR
- Keuntungan dan kelemahan TSR
- Aplikasi program TSR
- Memory management untuk TSR

14.2.2 Arsitektur Program TSR

- Initialization routine
- Resident routine
- Interrupt handler

- Memory allocation
- Program termination

14.2.3 Interupsi untuk TSR

- INT 21h fungsi 31h: Terminate and Stay Resident
- INT 21h fungsi 25h: Set Interrupt Vector
- INT 21h fungsi 35h: Get Interrupt Vector
- Custom interrupt untuk komunikasi

14.2.4 Implementasi TSR

- Program structure
- Memory management
- Interrupt hooking
- Communication mechanism
- Error handling

14.3 Praktikum

1. Program TSR sederhana
2. Program TSR dengan interrupt handler
3. Program TSR dengan komunikasi
4. Program TSR dengan memory management
5. Program TSR dengan error handling

14.4 Contoh Kode

```
; Program TSR sederhana
TITLE Program TSR
.MODEL SMALL
.STACK 100h

.DATA
    pesan_install DB 'TSR berhasil diinstall!$'
    pesan_uninstall DB 'TSR berhasil diuninstall!$'
    old_int21h DD ?
    tsr_installed DB 0

.CODE
START:
```

```
    MOV AX, @DATA
    MOV DS, AX

    ; Cek apakah TSR sudah diinstall
    MOV AH, 0FFh
    INT 21h
    CMP AL, 0AAh
    JE sudah_install

    ; Install TSR
    CALL install_tsr

    ; Tampilkan pesan
    MOV AH, 09h
    MOV DX, OFFSET pesan_install
    INT 21h

    ; Terminate and Stay Resident
    MOV AH, 31h
    MOV AL, 0
    MOV DX, 1000h  ; Ukuran program dalam paragraph
    INT 21h

sudah_install:
    ; Uninstall TSR
    CALL uninstall_tsr

    ; Tampilkan pesan
    MOV AH, 09h
    MOV DX, OFFSET pesan_uninstall
    INT 21h

    MOV AH, 4Ch
    INT 21h

; Prosedur install TSR
install_tsr PROC
    ; Simpan interrupt vector lama
    MOV AH, 35h
    MOV AL, 21h
    INT 21h
    MOV WORD PTR old_int21h, BX
    MOV WORD PTR old_int21h[2], ES

    ; Set interrupt vector baru
    MOV AH, 25h
    MOV AL, 21h
    MOV DX, OFFSET new_int21h
    INT 21h
```

```
    RET
install_tsr ENDP

; Prosedur uninstall TSR
uninstall_tsr PROC
    ; Restore interrupt vector lama
    MOV AH, 25h
    MOV AL, 21h
    MOV DX, WORD PTR old_int21h
    MOV DS, WORD PTR old_int21h[2]
    INT 21h

    RET
uninstall_tsr ENDP

; Interrupt handler baru
new_int21h PROC
    CMP AH, 0FFh
    JNE call_old_int

    ; Handler untuk komunikasi TSR
    MOV AL, 0AAh  ; Tanda TSR aktif
    IRET

call_old_int:
    ; Panggil interrupt handler lama
    JMP DWORD PTR old_int21h

new_int21h ENDP

END START
```

14.5 Latihan

1. Buat TSR yang menampilkan waktu di layar
2. Buat TSR yang menangani hotkey
3. Buat TSR yang memantau aktivitas keyboard
4. Buat TSR yang melakukan backup otomatis

14.6 Tugas

- Implementasikan TSR untuk system monitoring
- Buat TSR untuk keyboard macro
- Implementasikan TSR untuk file watcher
- Dokumentasikan teknik memory management untuk TSR

14.7 Referensi

- Hyde, Randall. *The Art of Assembly Language*, 2nd ed., No Starch Press, 2010.
- Susanto. *Belajar Pemrograman Bahasa Assembly*, Elex Media Komputindo, 1995.

Daftar Pustaka

1. Hyde, Randall. *The Art of Assembly Language*, 2nd ed., No Starch Press, 2010.
2. Susanto. *Belajar Pemrograman Bahasa Assembly*, Elex Media Komputindo, 1995.
3. Partoharsojo, Hartono. *Tuntunan Praktis Pemrograman Assembly*, Penerbit Informatika.
4. Brey, Barry B. *Mikroprosesor Intel 8086/8088 dsb.*, edisi terjemahan, Penerbit Informatika.
5. Borland. *Turbo Assembler (TASM) User's Guide*, Borland International.
6. Intel Corporation. *Intel 64 and IA-32 Architectures Software Developer's Manual*, Intel.
7. Nopi, Arif. *Tutorial Bahasa Assembly (x86)*, Jasakom, Edisi Online.