



Nama Irfan Arif Maulana
NPM 1906379270

Kode Asisten QH
Modul 3 & 4

Part 1 – Analisis Program

```
.MODEL SMALL
.DATA

    TEXT db 'Hello, how are you doing today?', 13, 10, '$'
    TEXT2 db 'INPUT (happy = 1/sad = 0): ', '$'
    TEXT3 db 13, 10, "That's great to hear!", '$'
    TEXT4 db 13, 10, 'I hope your day gets better..', '$'

.CODE
.STARTUP

    mov DX, offset TEXT
    mov AH, 09h
    int 21h

INPUT:

    mov DX, offset TEXT2
    mov AH, 09h
    int 21h

    mov AH, 01H
    int 21H

HAPPY: ber

    cmp AL, 31h
    jne SAD

    mov DX, offset TEXT3
    mov AH, 09h
    int 21h

    jmp STOP

SAD:

    cmp AL, 30h
    jne INPUT

    mov DX, offset TEXT4
    mov AH, 09h
    int 21h

STOP:
.EXIT
END
```

1. Bagian berwarna hijau merupakan interrupt yang berfungsi untuk ...

Interrupt yang diimplementasikan pada program bagian warna hijau berfungsi untuk menampilkan string yang alamatnya disimpan pada register DX (berupa alamat DS:DX). Hal ini dapat dilakukan sebab register AH diisi oleh nilai 09H.

String yang ditampilkan merupakan string yang tersimpan pada variabel TEXT pada data segment. Alamat pertama alias huruf pertama dari string TEXT disimpan pada register DX sebagai persiapan print string TEXT ke standard output. Kemudian perintah interrupt dengan nilai 09H pada AH berfungsi untuk menampilkan string yang ada pada alamat DS:DX tersebut secara berurutan hingga bertemu tanda '\$'.

2. Bagian berwarna biru merupakan interrupt yang berfungsi untuk ...

Interrupt yang diimplementasikan pada program bagian warna biru berfungsi untuk melakukan permintaan input keyboard. Dengan input yang dibaca sebanyak satu buah karakter. Hal ini mungkin untuk dilakukan sebab register AH diisi dengan nilai 01H.

3. Bagian berwarna kuning merupakan ... Jelaskan masing-masing instruksi dalam bagian tersebut!

Ketiga bagian program yang diwarnakan kuning mempunyai fungsi yang sama, yaitu untuk melakukan *jump* atau lompatan perintah pada label yang telah ditentukan pada program. Ketiga jump ini memanfaatkan prinsip *conditional jump*. Yaitu, sebelum perintah jump dilakukan, akan dicek terlebih dahulu suatu kondisi. Jika kondisi tersebut terpenuhi, maka perintah jump akan dilakukan. Begitu pula sebaliknya. Pada kasus ini, ketiga jump menggunakan kondisi dari perintah jump berjenis JNE atau *Jump if Not Equal* yang mana akan mengecek kesamaan nilai dengan yang ada pada Zero Flag (ZF). ZF sendiri bisa mendapat nilai dari proses aritmatika ataupun komparasi. Pada JNE, perintah jump akan dilakukan jika tidak senilai dengan apa yang ada pada ZF.

Secara berurutan, perintah JNE pertama berfungsi untuk melakukan jump ke label SAD jika hasil komparasi AL (input keyboard) tidak sama dengan 1 (ASCII: 31). JNE kedua berfungsi untuk mengakhiri program setelah label HAPPY selesai dieksekusi agar label SAD tidak perlu dieksekusi. JNE terakhir berfungsi untuk lompat kembali ke label INPUT jika AL tidak bernilai 0 (ASCII: 30) sebagai bentuk *error handling*.

4. Jelaskan masing-masing fungsi dari nilai 13, 10, dan simbol '\$' yang diberikan pada semua teks yang didefinisikan!

Ketiga nilai dan simbol tersebut berfungsi sebagai perintah dalam cara penulisan teks pada bahasa assembly.

Spesifiknya, nilai 13 disebut sebagai *Carriage Return* memiliki fungsi untuk menempatkan kembali posisi *cursor* pada terminal (standard output) ke kolom pertama pada baris sekarang. Ada pun nilai 10 disebut sebagai *Line Feed* memiliki fungsi untuk menempatkan *cursor* pada baris baru pada terminal. Atau dalam kata lain, membuat baris baru dengan letak *cursor* di kolom terakhir di mana *cursor* tersebut ditempatkan. Sehingga penggunaan keduanya bersamaan berfungsi untuk membuat baris baru pada terminal sekaligus menempatkan *cursor* pada awal baris tersebut (kolom paling kiri).

Ada pula simbol '\$' memiliki fungsi yang berbeda dengan nilai 13 dan 10 yang fungsinya sebagai kontrol posisi *cursor*. Simbol '\$' berfungsi sebagai penanda akhirnya dari suatu teks atau string. Dengan begitu, program akan mengetahui kapan akhir dari sebuah string dari masing-masing variabel TEXT. String yang diprint pada terminal merupakan yang terletak sebelum simbol '\$'. Sedangkan string yang terletak setelah simbol '\$' tidak akan diprint.

5. Jalankan koding diatas, kemudian jelaskan apa yang dilakukan oleh program tersebut (secara *step-by-step*)!

Sederhananya, program tersebut mendemonstrasikan permintaan input keyboard pada terminal dan penampilan output pada terminal berdasarkan evaluasi nilai input yang diberikan.

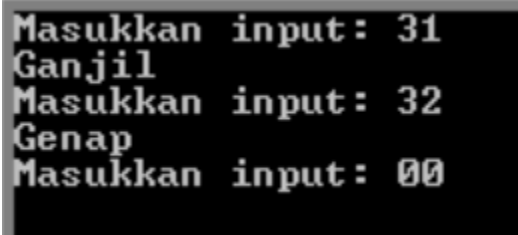
Pra emulasi, program tersebut mendeklarasikan terlebih dahulu 4 buah variabel string yang nilainya disimpan pada data segment sebagai persiapan string yang nantinya akan diprint ke terminal sebagai output.

Kemudian selama program diemulasi, jalannya program dapat dibagi menjadi 3 bagian utama. Yaitu pertama menampilkan terlebih dahulu string TEXT. Kedua, program menampilkan string TEXT2 dan diikuti dengan meminta satu karakter input keyboard. Lalu terakhir, input keyboard ini akan dievaluasi nilainya sebagai penentu perintah mana yang akan dieksekusi oleh program. Jika input bernilai 1, maka program akan terus dilanjutkan dengan mem-print string TEXT3 dan melakukan jump ke akhir program (jmp STOP). Ada pun jika input tidak bernilai 1, maka program akan loncat ke label SAD dan mengevaluasi kembali nilai input. Jika input bernilai 0, maka program akan berlanjut ke penampilan TEXT4 dan berakhir ke STOP. Jika pada evaluasi kedua ini nilai input tidak sama dengan 0, maka program lompat kembali ke label INPUT untuk melakukan permintaan input kembali sampai *user* hanya memasukkan 0 atau 1 dan mengeksekusi kembali HAPPY atau SAD sampai program menuju ke STOP.

Part II - Programming

1. Buatlah program
 - a. Meminta satu buah input berupa bilangan dua digit yang dilakukan beserta sebuah *prompt text* (contohnya “masukkan input: ” seperti di bawah).
 - b. Jika input adalah bilangan genap, maka outputnya berupa teks “Genap”, jika input adalah bilangan ganjil, maka outputnya berupa prompt text “Ganjil”.
 - c. Program mengulangi proses tersebut sampai diberikan input 00 dari *user*
 - d. Gunakan minimal 1 *procedure* untuk meminta input dan 1 *macro* untuk memberi *output & prompt text*
 - e. Di save dalam bentuk ASM, dikumpulkan bersamaan dengan file ini

Contoh :



```
Masukkan input: 31
Ganjil
Masukkan input: 32
Genap
Masukkan input: 00
```

Tambahan :

Cara buat baris baru: <https://stackoverflow.com/questions/15832893/printing-a-new-line-in-assembly-language-with-ms-dos-int-21h-system-calls>

DOS Interrupts (int 21h): <http://spike.scu.edu.au/~barry/interrupts.html#ah02>