Game Tebak Angka

Akhmad Muntako
Teknik Komputer 2013
1306368646
Kelompok 65
Akmad.muntako@ui.ac.id

Muhammad Zaini
Teknik Komputer 2013
1306368652
Kelompok 65
Muhammad.zaini@ui.ac.id

Limas Baginta
Teknik Komputer 2013
1306368690
Kelompok 67
Limas.baginta@gmail.com

Fransiska Dyah Ayu
Teknik Komputer 2013
1306368691
Kelompok 67
Fransiska.dyah@ui.ac.id

***Abstrak*—Bahasa Assembly merupakan bahasa pemograman terendah kedua setelah bahasa mesin. Sedangkan bahasa C, C++, Java, dll merupakan bahasa pemrograman tingkat tinggi yang penulisannya lebih bisa dimengerti oleh manusia.**

Kata Kunci—assembly, mikroprosesor, instruksi, pengalamatan, program.

I. PENDAHULUAN

Pada proyek Ujian Tengah Semester Praktikum Mikrokontroller kali ini, kami membuat suatu program yang bernama game tebak angka dalam bahasa assembly. Input pada program ini menggunakan keyboard dan input yang digunakan maksimal tiga digit angka. Pemain 1 memasukkan nilai yang akan ditebak, sedangkan pemain 2 akan menebak nilainya. Saat pemain 2 mencoba menebak angka dan jika angka yang ditebaknya tidak sama dengan angka yang dimasukkan oleh pemain 1, maka akan ditampilkan petunjuk yang akan menunjukkan input yang dimasukkan pemain 2 lebih besar atau lebih kecil dari input yang dimasukkan oleh pemain 1.

II. DASAR TEORI

Bahasa Assembly merupakan bahasa pemrograman diatas bahasa mesin, namun berada dibawah bahasa tingkat tinggi seperti bahasa C, C++, Java, dll. Keuntungan menggunakan bahasa assembly adalah bisa mengakses register atau memori secara langsung dan ukuran filenya lebih kecil. Pada kali ini akan digunakan mikroprosesor 8086.

8086 meupakan suatu mikrokontroller yang berukuran 16-bit. Sehingga untuk mereferensi ukuran data di 8086 ada tiga jenis yakni byte yang berukuran 8-bit, word yang berukuan 16-bit, dan doubleword yang berukuran 32-bit.

8086 menggunakan format *little endian* yang artinya *least significant value* disimpan terlebih dahulu di memori.

8086 menggunakan *segmented memory* yang terdiri dari dua angka yang biasanya dituliskan dalam bentuk heksadesimal dan dipisahkan oleh *colon* yang mempresentasikan *segment* dan *offset* yang biasanya disebut *logical address*.

Kesamaan yang dimiliki oleh keluarga 8086 adalah sebagai berikut:

- *General Purpose Register*

  CPU meliki empat buah *general purpose* yakni AX (*Accumulator*), BX (*Base*), CX (*Count*), dan DX (*Data*) yang mempunyai ukuran 16-bit. Register tersebut bisa dipisah menjadi dua bagian yakni *High* (H) dan *Low* (L) yang mempunyai ukuran 8-bit.

- *Index Register*

  CPU memiliki dua buah *index register* yakni SI (*Source index*) dan DI(*Destination index*).

- *Pointer Register*

  CPU memiliki tiga buah *pointer register* yakni IP (*Instruction pointer*), SP (*Stack pointer*), dan BP (*Base pointer*).

- *Segment Register*

  Ada empat buah *segment register* yakni CS (*Code segment*), DS (*Data segment*), SS (*Stack segment*), dan ES (*Extra segment*).

- *Flag Register*

  Flag register akan berisikan flag yang merupakan laporan status dari CPU.

8086 juga mempunyai *data addressing mode* yakni *MOV instruction* yang berfungsi untuk menyalin sebuah data dari sumber ke tujuan, *register addressing* yang berfungsi untuk menyalin data dari suatu register ke register lainnya,

Selain itu, pada 8086 juga terdapat instruksi aritmatika, logika, dan *program control*. Pada instruksi aritmatika terdapat operasi penambahan, pengurangan, perkalian, pembagian, perbandingan, dan *BCD and ASCII Arithmetic*. Pada instruksi logika terdapat operasi *and, or,exclusive or, complement,* dan *shift and rotate*. Pada instruksi *program control* terdapat operasi *jump* yang terdiri dari *conditional jump, conditional jump, loop,* dan *controlling the flow of an Assembly Language Program*.

Kemudian terdapat pula instruksi *procedure* dan *macro*. *Procedure* merupakan suatu fungsi subrutin yang terdiri dari sekumpulan instruksi untuk menjalankan satu jenis tugas dan dapat digunakan berulang-ulang dalam suatu program. Instruksi ini menggunakan instruksi *CALL* dan RETURN. Sedangkan *macro* hamper sama seperti *procedure*, namun pada *macro* tidak menggunakan fungsi *CALL*. Ada pula *interrupt* yang dibagi menjadi dua jenis yakni *hardware-generated CALL* dan *software-generated CALL*. *Interrupt* yang sering digunakan yakni INT 21 yang merupakan *DOS function*.

III. PROGRAM

```
.model small
.stack 100h
.data
    CR          equ 13d
    LF          equ 10d
    judul       db  CR,LF,'GAME TEBAK ANGKA$'
    angkarahasia db CR,LF,'Masukkan maksimal 3 digit angka rahasia(0-255): $'
    tampilanLayar   db  CR, LF,'Temukan aku, Masukkan maksimal 3 digit angka(0-255) : $'
    kurang      db  CR, LF,'Aku berada dibawah angka masukanmu ','$'
    lebih       db  CR, LF,'Aku berada diatas angka masukanmu ', '$'
    sama        db  CR, LF,'Ahhh, kau menemukanku!', '$'
    salahMasukan db  CR, LF,'Salah memasukan input, cobalagi!', '$'
    retry       db  CR, LF,'Ulangi [y/n] ? ' ,'$'
    angka       db  0d
    inputUser   db  0d
    cekError    db  0d
    parameter   label Byte

.code
MACRO tampil a
    MOV ah, 9h
    INT 21h
    endm
proc input

jumlahinput:
    CMP     cl, 3d
    JG      selesaiinput
    MOV     ah, 7h
    INT     21h
    CMP     al, 0Dh
    JE      selesaiinput
    cmp     al, 39h
    JG      start
    cmp     al, 30h
    JL      start
    SUB     al, 30h
    MOV     dl, al
    PUSH    dx
    INC     cl
    JMP jumlahinput

selesaiinput:
    DEC cl
    CMP cl, 02h
    JG  over
    MOV BX, OFFSET cekError
    MOV BYTE PTR [BX], cl
    MOV cl, 0h
```

```
wile2:
    CMP cl,cekError
    JG endwile2
    POP dx
    MOV ch, 0h
    MOV al, 1d
    MOV dh, 10d

wile3:
    CMP ch, cl
    JGE endwile3
    MUL dh
    INC ch
    JMP wile3

endwile3:
    MUL dl
    JO  over
    MOV dl, al
    ADD dl, angka
    mov angka,dl
    JC  over
    MOV BX, OFFSET angka
    MOV BYTE PTR [BX], dl
    INC cl
    JMP wile2

endwile2:
    MOV ax, @data
    MOV ds, ax
    JMP game
    over:
    MOV dx, offset salahMasukan
    tampil dx
    JMP start
input endp


proc tebak
while:
    CMP     cl, 5d
    JG      endwhile
    MOV     ah, 1h
    INT     21h
    CMP     al, 0Dh
    JE      endwhile
    cmp     al, 39h
    JG      start
    cmp     al, 30h
    JL      start
    SUB     al, 30h
    MOV     dl, al
    PUSH    dx
    INC     cl
    JMP while

endwhile:
    DEC cl
    CMP cl, 02h
    JG  overflow
    MOV BX, OFFSET cekError
    MOV BYTE PTR [BX], cl
    MOV cl, 0h
tebak endp
```

```
proc cek
while2:
    CMP cl,cekError
    JG endwhile2
    POP dx
    MOV ch, 0h
    MOV al, 1d
    MOV dh, 10d

while3:
    CMP ch, cl
    JGE endwhile3
    MUL dh
    INC ch
    JMP while3

endwhile3:
    MUL dl
    JO  overflow
    MOV dl, al
    ADD dl, inputUser
    JC  overflow
    MOV BX, OFFSET inputUser
    MOV BYTE PTR [BX], dl
    INC cl
    JMP while2

endwhile2:
    MOV ax, @data
    MOV ds, ax
    MOV dl, angka
    MOV dh, inputUser
    CMP dh, dl
    JC greater
    JE equal
    JG lower

equal:
    MOV dx, offset sama
    tampil dx
    call ulang

greater:
    MOV dx, offset lebih
    tampil dx
    JMP game

lower:
    MOV dx, offset kurang
    tampil dx
    JMP game

overflow:
    MOV dx, offset salahMasukan
    tampil dx
    JMP game
RET
exit:

cek endp

proc ulang
    retry_while:
    MOV dx, offset retry
    tampil dx
    MOV ah, 1h
```

```
    INT 21h
    CMP al, 6Eh
    JE return_to_DOS
    CMP al, 79h
    JE restart
    CMP al, 4Eh
    JE return_to_DOS
    CMP al, 59h
    JE restart
    JMP retry_while

restart:
    JMP start
return_to_DOS:
    MOV ax, 4c00h
    INT 21h
    ret
    ulang endp
proc set
    MOV ax, 0h
    MOV bx, 0h
    MOV cx, 0h
    MOV dx, 0h
    MOV BX, OFFSET inputUser
    MOV BYTE PTR [BX], 0d
    MOV BX, OFFSET cekError
    MOV BYTE PTR [BX], 0d
    ret
    set endp

.startup
start:
    call set
    mov angka,0
    mov dx,offset judul
    tampil dx
    mov dx,offset angkarahasia
    tampil dx
    mov cl,0h
    mov dl,0h
    call input

game:
    call set
    MOV ax, @data
    MOV ds, ax
    MOV dx, offset tampilanLayar
    tampil dx
    MOV cl, 0h
    MOV dx, 0h
    call tebak
    call cek
.exit
end
```

IV. ANALISIS PROGRAM

Pada game tebak angka ini bisa dimainkan oleh 2 pemain. Dimana pemain pertama bertugas untuk menuliskan angka yang akan ditebak oleh pemain kedua.

- Definisi string dan variable yang akan digunakan

```
.data

;Deklarasi digunakan untuk me
CR          equ 13d
LF          equ 10d

;Pesan string digunakan dalam
judul       db  CR,LF,'GAME T
angkarahasia db CR,LF,'Masukk
tampilanLayar       db  CR, LF
kurang      db  CR, LF,'Aku be
lebih       db  CR, LF,'Aku be
sama        db  CR, LF,'Ahhh,
salahMasukan db  CR, LF,'Sala
retry       db  CR, LF,'Ulang

angka       db  0d      ;vari
inputUser   db  0d      ;vari
cekError    db  0d      ;vari

parameter       label Byte
```

- Macro untuk menampilkan string yang di definisikan di data

```
MACRO tampil a

    MOV ah, 9h
    INT 21h
    endm
```

- *Procedure* yang benama input berfungsi untuk pemain 1 memasukkan angka yang akan ditebak oleh pemain 2.

```
jumlahinput:

    CMP     cl, 3d
    JG      selesaiinput

    MOV     ah, 7h
    INT     21h

    CMP     al, 0Dh
    JE      selesaiinput
    cmp     al, 39h
    JG      start
    cmp     al, 30h
    JL      start
    SUB     al, 30h
    MOV     dl, al
    PUSH    dx
    INC     cl

    JMP jumlahinput
```

Label jumlahinput berfungsi untuk membatasi input yang dimasukkan maksimal tiga digit yang dilakukan dengan cmp cl, 3d. Bila cl lebih dari 3, maka akan jump ke label selesaiinput. Membaca input dari pemain 1 dengan int 21h, 07. Dengan int 21h, 07 pemain 1 bisa memasukkan input namun tidak keluar di layar. Membandingkan al dengan 0DH(Enter) untuk mendeteksi saat pemain selesai memasukkan input dan jump ke label selesaiinput bila al sama ddengan 0DH, hal ini mendeteksikan juga bahwa input 0 dapat dilakukan dengan menekan enter. Untuk mendeteksi bahwa input yang dimasukkan pemain 1 merupakan angka, maka register al akan dibandingkan dengan 39h (9) dan 30h (0), bila input yang dimasukkan diluar interval tersebut, maka akan jump ke label start. Kemudian mengurangi al dengan 30 agar mendapatkan nilai yang diinginkan dan menyalinnya ke register dl. Increment cl untuk pengisian digit berikutnya.

```
selesaiinput:
; END reading user input

    DEC cl

    CMP cl, 02h
    JG  over

    MOV BX, OFFSET cekError
    MOV BYTE PTR [BX], cl

    MOV cl, 0h
```

Pada label selesaiinput, cl di decrement, kemudian membandingkan cl dengan 02, bila cl>2 maka akan jump ke over yang mendefinisikan bahwa input yang dimasukkan salah. Kemudian kita akan mendapatkan offset dari cekError dan akan di salin ke register bx.

```
wile2:
    CMP cl,cekError
    JG endwile2

    POP dx

    MOV ch, 0h
    MOV al, 1d
    MOV dh, 10d
```

Label wile2 berfungsi untuk membuat numeric yang actual untuk representasi dari angka yang dimasukkan pemain 1

```
wile3:

    CMP ch, cl
    JGE endwile3

    MUL dh

    INC ch
    JMP wile3
```

Pada label wile3 berfungsi untuk membuat perkalian 10 untuk membuat posisi relasi dari digit. Membandingkan registerch dan cl, bila hasilnya lebih besar atau sama maka akan jump ke labelendwile3. Register dh dikalikan, meng-increment register ch dan jump ke label wile3.

```
endwile3:
; END loop perkalian untu

    ; sekarang AL memiliki

    MUL dl

    JO  over            ;

    MOV dl, al
    ADD dl, angka
    mov angka,dl

    JC  over

    MOV BX, OFFSET angka
    MOV BYTE PTR [BX], dl

    INC cl

    JMP wile2
```

Pada label endwile3 akan dilihat apakah ada kesalahan input dari pemain 1. Bila terdapat kesalahan, akan jump ke variable over. Kemudian input dari pemain 1 yang awalnya di register dl, akan disimpan di variabel angka.

```
endwile2:
; END proses masukan dari user

    MOV ax, @data               ; mer
    MOV ds, ax                  ; me

    JMP game
    over:

    MOV dx, offset salahMasukan
    tampil dx
    JMP start           ; melakukan

input endp
```

Label endwile2digunakan untuk jump ke variable game yang akan digunakan untuk memulai game dengan pemain 2 memasukkan angka tebaknnya. Variabel over berfungsi untuk mengindikasikan bahwa input yang

dimasukkan pemain 1 salah dan akan jump ke start yakni awal game.

- *Procedure* yang bernama tebak befungsi agar pemain 2 memasukkan angka tebakannya.

```
proc tebak
    ; MULAI membaca m
while:

    CMP     cl, 5d
    JG      endwhile

    MOV     ah, 1h
    INT     21h

    CMP     al, 0Dh
    JE      endwhile
    cmp     al, 39h
    JG      start
    cmp     al, 30h
    JL      start
    SUB     al, 30h
    MOV     dl, al
    PUSH    dx
    INC     cl

    JMP while
```

Pada label while fungsinya sama seperti label jumlahinput yang ada di *procedure* input.

```
endwhile:
; END reading user input

    DEC cl

    CMP cl, 02h
    JG  overflow

    MOV BX, OFFSET cekError
    MOV BYTE PTR [BX], cl

    MOV cl, 0h

tebak endp
```

Pada label endwhile fungsinya sama seperti label selesaiinput pada *procedure* input.

- *Procedure yangbernama* cek berfungsi untuk mengecek apakah pemain 2 berhasil menebak angka yang dimasukkan oleh pemain 1.

Terdapat label while2, while3, endwhile3, dan endwhile2 yang fungsinya sama seperti label wile2, wile3, endwile2, dan endwile3 pada *procedure* input.

```
equal:

    MOV dx, offset sama
    tampil dx
    call ulang
```

Pada label equal berfungsi untuk menampilkan string variable sama bila angka yang dimasukkan pemain 2 sama dengan angka yang dimasukkan pemain 1 dan akan memanggil *procedure* ulang.

```
greater:

    MOV dx, offset lebih
    tampil dx
    JMP game
```

Pada label greater berfungsi untuk menampilkan string variable lebih bila angka yang dimasukkan pemain 2 lebihbesarl dari angka yang dimasukkan pemain 1 dan akan jump ke label game.

```
lower:

    MOV dx, offset kurang
    tampil dx
    JMP game
```

Pada label lower berfungsi untuk menampilkan string variable kurang bila angka yang dimasukkan pemain 2 lebih kecil dari angka yang dimasukkan pemain 1 dan akan jump ke label game.

```
overflow:

    MOV dx, offset salahMasukan
    tampil dx
    JMP game                    ; m


RET
exit:

cek endp
```

Pada label overflow berfungsi untuk menampilkan string variable salahMasukan bila pemain 2 salah menebak angka yang dimasukkan pemain 1 dan akan jump ke label game.

- *Procedure* yang bernama ulang berfungsi untuk menanyakan ke pemain apakah mereka ingin memulai permainan game tebak angka lagi. Bila pemain menekan tombol ‘y’ atau ‘Y’ maka akan mengulang game dari awal. Sedangkan bila pemain menekan tombol ‘n’ atau’N’ maka program akan berhenti.

```
proc ulang
    retry_while:

    MOV dx, offset retry
    tampil dx

    MOV ah, 1h
    INT 21h

    CMP al, 6Eh
    JE return_to_DOS

    CMP al, 79h
    JE restart

    CMP al, 4Eh
    JE return_to_DOS

    CMP al, 59h
    JE restart


    JMP retry_while

restart:
    JMP start
return_to_DOS:
    MOV ax, 4c00h
    INT 21h

    ret
    ulang endp
```

Pada label retry_while berfungsi untuk menampilkan string variable retry. Dengan menggunakan int 21h, 01 berfungsi untuk mengambil input dari pemain. Kemudian akan dibandingkan apakah input user ‘n’ (6Eh), ‘y’ (79h), ‘N’ (4Eh), dan ‘Y’ (59h). Bila iya, maka akan jump ke label restart atau return_to_DOS. Bila input yang dimasukkan bukan yang diatas, maka akan jump ke label retry_while lagi.

- *Procedure* yang bernama set berfungsi untuk menyimpan variable menuju 0h

```
proc set
     ; MULAI menyimpan varia
     MOV ax, 0h
     MOV bx, 0h
     MOV cx, 0h
     MOV dx, 0h

     MOV BX, OFFSET inputUser
     MOV BYTE PTR [BX], 0d

     MOV BX, OFFSET cekError
     MOV BYTE PTR [BX], 0d
     ; END restart
     ret
     set endp
```

- Saat di startup, pada label start berfungsi untuk menampilkan string variable judul dan angkarahasia serta memanggil *procedure* input dimana pemain 1 akan meninput angka untuk ditebak oleh pemain 2.

```
.startup
start:
     call set
     mov angka,0
     mov dx,offset judul
     tampil dx
     mov dx,offset angkarahasia
     tampil dx
     mov cl,0h
     mov dl,0h
     call input
```

- Pada label game berfungsi untuk memanggil *procedure* set dan menampilkan string variable tampilanLayar. Kemudian akan dipanggil *procedure* tebak dan cek.

```
game:
     call set
     MOV ax, @data              ; me
     MOV ds, ax                 ; me
     MOV dx, offset tampilanLayar
     tampil dx

     MOV cl, 0h                 ; me
     MOV dx, 0h                 ; me
     call tebak
     call cek

.exit
end
```

V. SIMULASI PROGRAM

- Bila pemain 1 memasukkan input bukan angka

  Pemain 1 memasukkan input a

```
GAME TEBAK ANGKA
Masukkan maksimal 3 digit angka rahasia(0-255):
GAME TEBAK ANGKA
Masukkan maksimal 3 digit angka rahasia(0-255):
```

- Pemain 1 memasukkan input angka diatas tiga digit

  Pemain 1 memasukkan input 1234

```
GAME TEBAK ANGKA
Masukkan maksimal 3 digit angka rahasia(0-255):
Salah memasukan input, cobalagi!
GAME TEBAK ANGKA
Masukkan maksimal 3 digit angka rahasia(0-255):
```

- Pemain 1 memasukkan input tiga digit

  Pemain 2 memasukkan input 123

```
GAME TEBAK ANGKA
Masukkan maksimal 3 digit angka rahasia(0-255):
Temukan aku, Masukkan maksimal 3 digit angka(0-255):
```

- Pemain 2 menebak angka

```
Temukan aku, Masukkan maksimal 3 digit angka(0-255): 120
Aku berada diatas angka masukanmu
Temukan aku, Masukkan maksimal 3 digit angka(0-255): 124
Aku berada dibawah angka masukanmu
Temukan aku, Masukkan maksimal 3 digit angka(0-255): 123
Ahhh, kau menemukanku!
Ulangi [y/n] ?
```

- Bila pemain menekan tombol selain n, y, N, atau Y

```
Ulangi [y/n] ? r
Ulangi [y/n] ?
```

- Bila pemain menekan tombol y atau Y

```
Ulangi [y/n] ? y
GAME TEBAK ANGKA
Masukkan maksimal 3 digit angka rahasia(0-255):
```

- Bila pemain menekan tombol n atau N

```
Ulangi [y/n] ? n
PROGRAM HAS RETURNED CONTROL
TO THE OPERATING SYSTEM
```

REFERENCES

[1] Modul 2-Praktikum Mikrokontroller

[2] Modul 3-Praktikum Mikrokontroller

[3] Modul 4-Praktikum Mikrokontroller