

Daftar Isi

Daftar Isi	1
Pertemuan 2: Pengenalan Debug dan Interrupt.....	2
P2.1. Teori	3
Pengenalan DEBUG.....	3
Perbedaan Program COM dan EXE.....	3
Perintah-perintah Dasar DEBUG	4
Perintah-perintah Dasar Assembler	12
Looping	14
Interrupt.....	14
P2.2. Studi Kasus.....	15
P2.3. Latihan.....	Error! Bookmark not defined.
P2.4. Daftar Pustaka	18

Pertemuan 2

Pengenalan Debug

Objektif:

1. Mahasiswa mengetahui perangkat lunak DEBUG
2. Mahasiswa mengetahui perintah-perintah dasar DEBUG
3. Mahasiswa dapat membuat program assembler sederhana menggunakan DEBUG
4. Mahasiswa mengetahui perintah-perintah Interrupt dan fungsinya
5. Mahasiswa dapat membuat program menggunakan Interrupt

P2.1 Teori

PENGENALAN DEBUG

DEBUG adalah alat bantu dalam perancangan peralatan berbasis mikro-prosesor, karena dapat mencapai tingkat perangkat keras yang paling dalam dari suatu komputer, misal menulis informasi ke dalam boot sector, direktori, FAT, menjalankan interupsi BIOS atau DOS.

Hal-hal penting dalam Debug:

- Hanya mengenal dan selalu bekerja dengan bilangan-bilangan heksadesimal
- Bekerja dengan penunjukan ke alamat-alamat memori memakai format **segment : offset**
- Kemampuan mengakses daerah "very low level access" (software/hardware).

Setiap jenis komputer (mainframe, minicomputer, microcomputer) memiliki sarana debugging berbeda.

PERBEDAAN PROGRAM .COM DAN .EXE

Secara umum perbedaan antara program yang ber-extension .com dan .exe adalah ukuran luas file.

Kelebihan dan Kekurangan File .COM

- Lebih pendek dari file EXE - Lebih cepat dibanding file EXE
- Hanya dapat menggunakan 1 segment
- Ukuran file maksimum 64 KB (ukuran satu segment)
- sulit untuk mengakses data atau procedure yang terletak pada segment yang lain.
- 100h byte pertama merupakan PSP(Program Segment Prefix) dari program tersebut.
- Bisa dibuat dengan DEBUG

Kelebihan dan Kekurangan File .EXE

- Lebih panjang dari file COM - Lebih lambat dibanding file COM
- Bisa menggunakan lebih dari 1 segmen
- Ukuran file tak terbatas sesuai dengan ukuran memory.
- mudah mengakses data atau procedure pada segment yang lain.
- Tidak bisa dibuat dengan DEBUG

-

PERINTAH-PERINTAH DASAR DEBUG

Untuk menjalankan utility debug, cukup ketikkan debug pada command prompt. Apabila jika ingin membuka file langsung untuk dioperasikan debug maka cukup menambahkan nama file sebagai command tail seperti contoh berikut ini:

```
C:> DEBUG COBA.COM
```

-

Untuk memulai menggunakan debug, ketikkan perintah seperti berikut:

```
C:>DEBUG (enter)
```

A (Assembler)

Perintah assembler berguna untuk tempat menulis program assembler. Seperti contoh berikut:

```
-A100
0FD8:100
```

Terlihat pada sebelah kiri bawah huruf A terdapat angka yang merupakan pernyataan segment dan offset dimana program ditempatkan.

N (Name)

Perintah ini digunakan untuk membuat nama program. Adapun tata cara penulisan namanya adalah *N [Drive]:[nama program]*. Contoh dapat dilihat di bawah ini:

```
-N C: COBA.COM
```

RCX (Register CX)

Perintah RCX adalah perintah untuk mengetahui dan memperbaharui isi register CX yang merupakan tempat penampungan panjang program yang sedang aktif sebelum dijalankan.

Contoh:

```
-RCX
CX 0000
:
```

RIP (Register IP)

Perintah yang digunakan untuk memulai proses program dari titik tertentu. Pada program Debug selalu dimulai dengan 100 hexa, seperti contoh berikut:

```
-RIP
IP 0100
:
```

W (Write)

Setelah program selesai dibuat serta RCX sudah ditentukan dan RIP dari program yang dibuat, maka langkah berikutnya adalah menulis program sebelum memprosesnya.

```
-W
Writing 0008 bytes
```

G (Go)

Perintah G digunakan untuk melakukan proses (running) di dalam Debug. Lihat contoh di bawah ini, huruf A merupakan hasil (output) dari program yang dibuat.

```
-G
A
Program terminated normally
```

T (Trace)

Perintah ini digunakan untuk memproses sebaris program saja. Untuk menjalankan, cukup ketik T kemudian Enter.

Contoh:

- T (enter)

Maka akan muncul:

```
AX=0200 BX=0000 CX=0008 DX=0000 SP=CE2E BP=0000 SI=0000 DI=0000 DS=29E7
DS=0FD8 ES=0FD8 SS=0FD8 CS=0FD8 IP=0102 NV UP DI PL NZ NA PO NC
0FD8:0102 B241 MOV DI, 41
```

R (Register)

Untuk menampilkan informasi komposisi register-register di dalam mikroprosesor, alamat memori, serta isi dari alamat memori tersebut yang mungkin berupa instruksi yang akan dilaksanakan oleh komputer atau data.

Contoh :

- R (enter)

Maka akan muncul:

```
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000 DS=29E7 ES=29E7
SS=29E7 CS=29E7 IP=0100 NV UP EI PL NZ NA PO NC 29E7:0100 0114 ADD[SI],DX
DS:0000=20CD
```

AX, BX, CX, DX, SP, BP, SI, DI, DS, ES, SS, CS, dan IP adalah register internal mikroprosesor yang dipakai dalam CPU.

NV, UP, EI, PL, NZ, NA, PO, dan NC adalah output dari sebuah register yang disebut register status atau register flag.

Angka **29E7:0100** adalah alamat lokasi memori dengan format segment:offset. Kedua nilai tersebut merupakan kombinasi antara register CS (Code Segment) dengan IP (Instruction Pointer).

0114 adalah isi alamat memori bersangkutan, byte ke-1 berisi nilai '01', dan byte ke-2 berisi nilai '14'. Karena setiap alamat memori berisi satu byte, tentunya nilai 01 itulah yang berdiam pada alamat offset 0100, sedangkan nilai 14 ada di alamat 0101 ADD[SI],DX adalah terjemahan intruksi dari alamat memori bersangkutan.

Untuk mengubah nilai-nilai register internal dapat menggunakan perintah-perintah seperti

berikut:

Tabel 2.1 Perintah Ubah Nilai Register

Nama Register	Keterangan
RAX	Mengubah Nilai register AX
RBX	Mengubah Nilai register BX
RCX	Mengubah Nilai register CX
RDX	Mengubah Nilai register DX
RSP	Mengubah Nilai register SP
RBP	Mengubah Nilai register BP
RSI	Mengubah Nilai register SI
RDI	Mengubah Nilai register DI
RDS	Mengubah Nilai register DS
RES	Mengubah Nilai register ES
RSS	Mengubah Nilai register SS
RCS	Mengubah Nilai register CS
RIP	Mengubah Nilai register IP

Sebagai contoh untuk mengubah nilai register AX dari nilai '0000' ke nilai '1111' yaitu:

```
C:>Debug (enter)
-RAX (enter)
AX 0000
: 1111 (diisi setelah ':' dan enter)
```

Contoh di atas juga berlaku untuk register internal lainnya dan untuk diperhatikan bahwa angka-angka yang dimasukkan kedalam register harus nilai heksadesimal.

D (Dump)

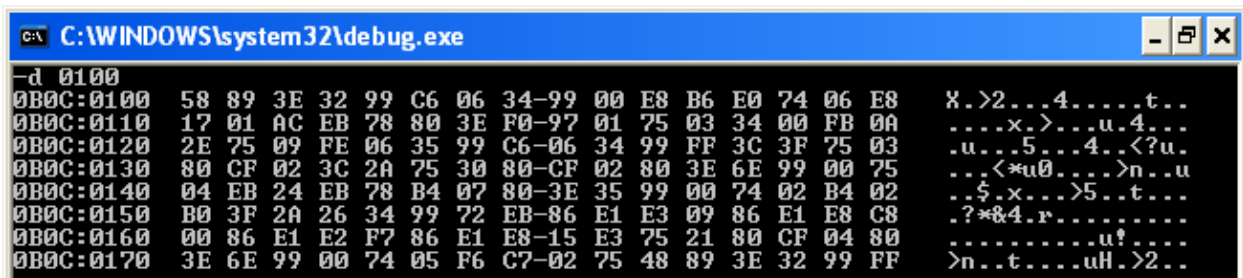
Berfungsi untuk melihat isi blok memori.

Contoh :

C:>Debug (enter)

-d 0100 (enter)

Hasil yang diperoleh:



Gambar 2.1 Melihat isi blok memori

Dari tampilan tersebut, terbagi menjadi 3 bagian, yaitu :

- Bagian kiri,
Menampilkan alamat-alamat memori dengan format segment:offset
- Bagian tengah,
Menampilkan angka-angka dalam heksadesimal sebagai isi dari alamat-alamat memori
- Bagian kanan,
Menampilkan kode-kode karakter ASCII sebagai terjemahan dari angka heksadesimal tersebut.

Debug hanya akan memperlihatkan 96 jenis karakter ASCII tercetak (*printable character*), mulai dari nilai desimal 33-127. Di luar nilai itu, karakter yang ditampilkan hanyalah berupa tanda titik (dot atau period).

Beberapa parameter yang dapat digunakan:

- L (Length/Panjang)
Memiliki arti menampilkan data sepanjang 2 byte, bila parameter 'L' tidak diberikan, maka otomatis akan ditampilkan 128 byte data.
-D 0100 L 2 (enter)
- Alamat awal - alamat akhir

-D 0100 01FF (enter)

- Alamat segment:offset

-D FFFF:0000 (enter)

- Alamat segment:offset sampai segment:offset

-D F000:E000 F000:E000 (enter)

U (Unassemble)

Berfungsi untuk menampilkan listing program bahasa mesin. Sintaks penggunaan perintah Unassemble:

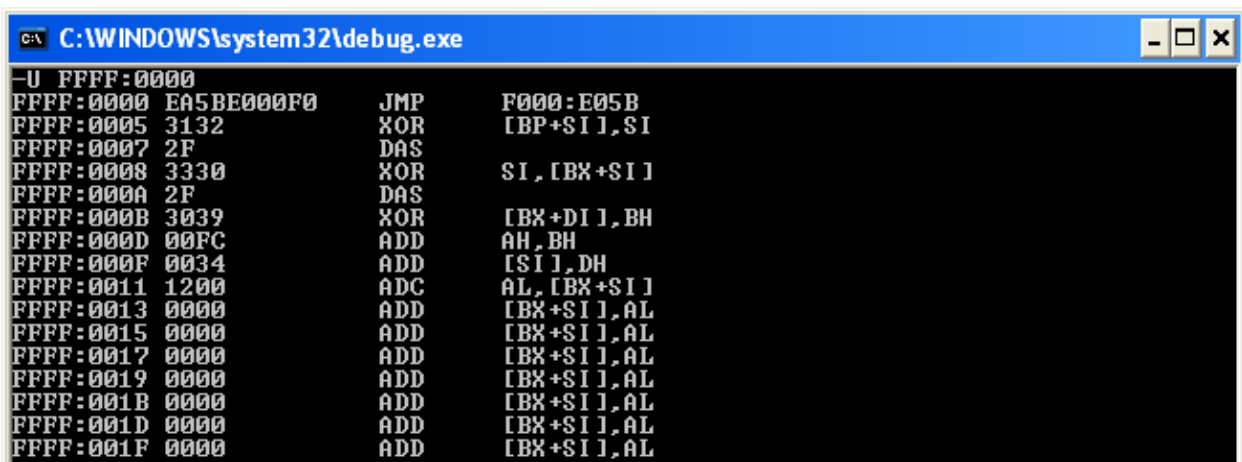
U[tempat memulai Unassemble] L[Panjang program]

Contoh :

C:>Debug (enter)

-U FFFF:0000 (enter)

Hasil :



```
C:\WINDOWS\system32\debug.exe
-U FFFF:0000
FFFF:0000 EA5BE000F0 JMP F000:E05B
FFFF:0005 3132 XOR [BP+SI],SI
FFFF:0007 2F DAS
FFFF:0008 3330 XOR SI,[BX+SI]
FFFF:000A 2F DAS
FFFF:000B 3039 XOR [BX+DI],BH
FFFF:000D 00FC ADD AH,BH
FFFF:000F 0034 ADD [SI],DH
FFFF:0011 1200 ADC AL,[BX+SI]
FFFF:0013 0000 ADD [BX+SI],AL
FFFF:0015 0000 ADD [BX+SI],AL
FFFF:0017 0000 ADD [BX+SI],AL
FFFF:0019 0000 ADD [BX+SI],AL
FFFF:001B 0000 ADD [BX+SI],AL
FFFF:001D 0000 ADD [BX+SI],AL
FFFF:001F 0000 ADD [BX+SI],AL
```

Gambar 2.2 Menampilkan listing program bahasa mesin

Beberapa bentuk perintah 'U' :

- -U F000:E05B (enter)
- -U 100 L 100 (enter)

E (enter)

Berfungsi untuk mengisi atau mengubah data dalam memori dalam jumlah kecil.

Contoh :

C:>Debug (enter)

-E 0100 (enter)

Hasil :

-E 0100

29E7:0100 01.

Setelah angka 01, dapat dimasukkan nilai untuk mengganti angkut tersebut langsung di belakangnya yang diakhir dengan menekan ENTER atau menekan SPACE BAR untuk berpindah ke alamat berikutnya atau menekan tanda '-' (Hyphen) untuk mundur ke lokasi sebelumnya.

F (Fill)

Berfungsi untuk mengisi lokasi memori. Perbedaan dengan perintah 'E (Enter)', yang menawarkan modifikasi memori secara satu alamat demi satu alamat, sedang 'F (Fill)' untuk mengubah isi alamat memori dalam jumlah besar, sesuai dengan rentang (range) yang dikehendaki.

Contoh :

C:>Debug (enter)

-F 0100 017F 58 (enter)

Berarti isilah mulai alamat offset 0100 sampai offset 017F dengan nilai heksa 58 (karakter ASCII 'x').

atau

- F F 0100 017F 'x' (enter)

Beberapa bentuk perintah 'F' :

- F 0100 L 1 41 (enter)

Arti : mulai offset 0100 sebanyak 1 byte diisi dengan nilai heksa 41 (karakter ASCII 'A').

C (Compare)

Berfungsi untuk membandingkan isi sebuah blok memori dengan isi blok memori lainnya. Format perintah: C alamat1 panjang alamat2

Contoh :

C:>Debug (enter)

-C 0100 L 10 0200 (enter)

Artinya, mulai offset 0100 sebanyak 16 byte (10 heksa) bandingkan dengan offset 0200. Hasil yang dimunculkan hanyalah nilai-nilai yang berbeda setiap alamat. Apabila layar tidak memberikan reaksi apapun selain kembali ke prompt '-' atau hyphen, berarti kedua blok memori persis sama.

S (Search)

Berfungsi untuk mencari data baik yang berupa karakter maupun untaian karakter (string) di dalam suatu blok memori tertentu. Apabila dalam pencarian, data yang dicari diketemukan, maka akan ditampilkan semua alamat dari data tersebut lengkap dengan nilai segment dan offsetnya, sebaliknya bila tidak diketemukan akan kembali ke prompt '-'.

Format perintah : S alamat awal panjang alamat akhir

Contoh :

C:>Debug (enter)

-S F000:E000 L FF "IBM" (enter)

Berarti mulai alamat F000:E000 sebanyak FFh byte cari string "IBM".

-S F000:E000 L FF "A" (enter)

Berarti mulai alamat F000:E000 sebanyak FFh byte cari string "A"..

M (Move)

Berfungsi untuk memindahkan atau menyalin data yang ada di suatu lokasi memori ke alamat memori lainnya.

Format perintah: M alamat asal panjang alamat tujuan

Contoh :

C:>Debug (enter)

-M 0100 L 7F 0200 (enter)

Berarti mulai alamat offset 0100 sebanyak 7Fh byte isi memorinya pindahkan atau kopikan ke offset 0200.

Dari pembahasan perintah-perintah DEBUG dapat disimpulkan bahwa DEBUG dibuat dengan tujuan untuk dapat mengeksplorasi program-program yang sudah dibuat berikut segala dampaknya terhadap sistem dan aplikasi, sedangkan ASSEMBLY diadakan dengan untuk mempermudah seorang programmer dalam menyusun instruksi-instruksi pada sebuah program yang sedang dibuat.

PERINTAH-PERINTAH DASAR ASSEMBLER

Instruksi Bahasa Assembly

Secara fisik, kerja dari sebuah komputer dapat dijelaskan sebagai siklus pembacaan instruksi yang tersimpan di dalam memori. komputer menentukan alamat dari memori program yang akan dibaca, dan melakukan proses baca data di memori. Data yang dibaca diinterpretasikan sebagai instruksi. Alamat instruksi disimpan oleh komputer di register, yang dikenal sebagai program counter. Instruksi ini misalnya program aritmatika yang melibatkan 2 register.

Dalam bahasa Assembly mempunyai 3 tipe intruksi dasar yaitu:

- Mnemonic,
- Operand1 dan operand2
- Kometar

<i>MOV</i>	<i>AH,</i>	<i>#30H</i>	<i>; kirim 30H ke akumulator A</i>
↓	↓	↓	↓
Mnemonic	operand1	operand2	komentar
(opcode)			

Mnemonic atau opcode ialah kode yang akan melakukan aksi terhadap operand . Operand

ialah data yang diproses oleh opcode. Sebuah opcode bisa membutuhkan 1 ,2 atau lebih operand, kadang juga tidak perlu operand. Sedangkan komentar dapat kita berikan dengan menggunakan tanda titik koma (;).

Berikut adalah perintah-perintah dasar assembler yang digunakan dalam listing program menggunakan DEBUG.

MOV

Perintah MOV adalah perintah untuk mengisi, memindahkan, memperbaharui isi dari suatu register, variable ataupun suatu lokasi memori. Format perintah:

MOV [operand A], [operand B]

Dengan ketentuan:

Operand A berupa: register, variable, lokasi memory

Operand B berupa: register, variable, lokasi memori ataupun bilangan.

Operand B merupakan bilangan asal yang akan diisikan ke operand A. Operand A merupakan tujuan pengisian atau penduplikasian dari operand B.

Contoh:

MOV AH, AL

Artinya:

Operand A dari perintah di atas adalah register AH

Operand B dari perintah di atas adalah register AL maka,

Isi register AL diisi/diduplikat/dipindahkan ke register AH

Contoh:

MOV AH, 02

Artinya:

Operand A dari perintah di atas adalah register AH

Operand B dari perintah di atas adalah bilangan 02

Perintah di atas adalah memasukkan 02 ke register AH.

LOOP

Perintah ini adalah perintah pengulangan yang mempunyai tata penulisan sebagai berikut:

Loop [lokasi memori]

Syarat untuk menggunakan operasi Loop adalah diharuskannya mengisi register CX untuk setiap kali pengulangan. Berikut contoh penggunaan Loop pada program:

```
:0100 MOV CX, 05
:0103 MOV AH, 02
:0105 MOV DL, 41
:0107 INT 21
:0109 LOOP 0103
:0108 INT 20
```

Hasil program tersebut adalah mencetak huruf A sebanyak lima kali. Hal ini karena perintah Loop yang ditujukan ke arah lokasi memory *segment:0103* yaitu tempat yang berisikan perintah MOV AH, 02.

Untuk setiap kali looping dilakukan, komputer akan mengurangi isi register CX sampai dengan CX menunjukkan nilai nol (0) dan komputer akan melanjutkan ke baris berikutnya di bawah Loop. Apabila perintah Loop diarahkan ke lokasi memori *segment:0100* (MOV CX, 05) maka yang akan terjadi adalah program tidak akan berhenti karena CX akan melakukan looping terus-menerus dan tidak akan berhenti.

INTERRUPT

Subroutine yang dapat dipanggil saat menggunakan perintah interrupt terdiri dari dua jenis, yaitu:

1. *Bios Interrupt* yaitu interrupt yang disediakan oleh BIOS (Basic Input Output System). Interrupt yang termasuk ke dalam interrupt BIOS adalah int 0 hingga int 1F hexa.
2. *DOS Interrupt* yaitu interrupt yang disediakan DOS (Disk Operating System). Interrupt yang termasuk dalam interrupt DOS adalah interrupt di atas Int 1F hexa, misalkan Interrupt 20 hexa, interrupt 21 hexa, dan lain-lain.

Macam-macam Interrupt:

- **Int 20h**

Perintah Int 20h merupakan salah satu dari DOS interrupt. Perintah ini bertugas untuk memberhentikan proses computer terhadap suatu program COM. Bila pada setiap program COM yang dibuat tidak terdapat Int 20h, maka computer akan hang dikarenakan computer tidak menemukan perintah pemberhentian proses.

- **Int 21h**

Perintah Int 21h adalah salah satu Int yang termasuk DOS Interrupt karena Int 21h mempunyai banyak sekali tugas, sehingga tugasnya dibagi-bagi ke dalam beberapa bagian. Untuk menggunakan bagian-bagian tersebut maka Nomor bagian atau yang disebut dengan *Service Number* nya harus disertakan.

- **Int 21h Service 02h**

Untuk mencetak sebuah huruf ke layar digunakan fungsi kedua dari Int 21h. Untuk menjalankan fungsi Int 21h service 02h harus memenuhi beberapa syarat, yaitu:

- ✓ Register AH, harus berisi service number dari Int 21h yang akan dijalankan (02h)
- ✓ Register DL, harus berisi bilangan hexa dari karakter ASCII yang akan dicetak.

P2.2 Studi Kasus

Bagaimana mencetak huruf **A** menggunakan Debug.com

Caranya adalah:

1. Ketik DEBUG pada prompt seperti contoh di bawah ini:

```
C:> Debug
```

```
- A100
```

```
0FD8:0100
```

2. Ketik program dibawah ini:

```
MOV AH, 02
```

```
MOV DL, 41
```

Int 21

Int 20

Hingga terlihat pada layar sebagai berikut:

-A100

0FD8:0100 MOV AH, 02

0FD8:0102 MOV DL, 41

0FD8:0104 INT 21

0FD8:0106 INT 20

0FD8:0108

3. Kemudian periksa panjang program dengan perintah RCX, dan akan terlihat sebagai berikut:

-RCX

CX 0000

: 8

Isi kursor dengan angka 8 kemudian Enter. Beri nama program dengan perintah N (Name) yang bernama Cetak.com, seperti berikut:

-NCetak.com

4. Lihat kembali awal program dengan perintah RIP, apakah berisi 0100, jika tidak maka isilah dengan angka tersebut:

-RIP

IP 0102

:100

5. Gunakan perintah W (Write) untuk menulis ke disk computer, seperti contoh dibawah ini:

- W

Writing 0008 bytes

6. Untuk memproses program (compile) gunakan perintah G (Go).

- G

A

Program terminated normally

Output program tersebut adalah huruf A. Dijelaskan pula, bahwa program berhenti secara normal.

Penjelasan program perbarisnya adalah:

- MOV AH, 02

Hasil pelaksanaan perintah adalah berubahnya isi register AX, khususnya pada *high byte* nya. Dapat dilihat di utility Debug yang memproses setiap barisnya dengan menggunakan perintah T (trace) sebagai berikut:

-T

```
AX=0200 BX=0000 CX=0008 DX=0000 SP=CE2E BP=0000 SI=0000 DI=0000
DS=0FD8 ES=0FD8 SS=0FD8 CS=0FD8 IP=0102 NV UP DI PL NZ NA PO NC
0FD8:0102 B241 MOV DL, 41
```

- MOV DL, 41

Setelah mengubah isi register AH kemudian mengisi register DL dengan kode ASCII 41 hexa yaitu huruf A besar.

```
AX=0200 BX=0000 CX=0008 DX=0041 SP=CE2E BP=0000 SI=0000 DI=0000
DS=0FD8 ES=0FD8 SS=0FD8 CS=0FD8 IP=0102 NV UP DI PL NZ NA PO NC
0FD8:0104 CD21 INT 21
```

Kemudian hasilnya akan dicetak yaitu berupa huruf A menggunakan perintah interrupt 21 dengan service 02h

-T

A

```
AX=0241 BX=0000 CX=0008 DX=0041 SP=CE2E BP=0000 SI=0000 DI=0000
DS=0FD8 ES=0FD8 SS=0FD8 CS=0FD8 IP=0102 NV UP DI PL NZ NA PO NC
0FD8:0106 CD20 INT 20
```

Mengakhiri program dengan perintah `int 20h` dan akan memberikan pesan “*Program terminated normally*” yang menandakan bahwa program berhenti semestinya.

P2.3 Latihan

1. Buatlah program yang menghasilkan output huruf B menggunakan DEBUG!
2. Buatlah program yang menghasilkan output berikut: ABCDEFGHIJ

P2.4 Daftar Pustaka

Lukito, Ediman. Dasar-dasar Pemrograman dengan Assembler 8088. PT. Elex Media Komputindo. Jakarta. 1990.

Pemrograman Bahasa Assembly Edisi Online Versi 1.0. Agustus 2011.

<http://www.scribd.com/doc/46495287/Sto-Assembly>