

P E R C O B A A N 1

- INSTRUKSI DASAR
- MNEMONIC CODE DAN OPERATION CODE
- OPERASI STRING
- OPERASI PERCABANGAN
- STATUS LED

Tujuan :

1. Mengetahui dan mengenalkan pemrograman bahasa assembler (*assembly language*) yang berbasis sistem minimum BGC-8088.
2. Mengetahui perbedaan *mnemonic code* dan *opcode (operation code)*.
3. Penggunaan Interrupt 84 untuk menampilkan string di sistem minimum BGC-8088.
4. Memahami dasar-dasar instruksi untuk tujuan mengkomunikasikan mikroprosesor dengan perangkat yang terhubung dengannya.

Teori Dasar :

A. PENGERTIAN ASSEMBLER

Program pada suatu unit mikroprosesor (MPU) adalah bahasa mesin, yaitu suatu kode-kode instruksi yang memerintahkan MPU untuk melakukan perhitungan atau fungsi tertentu. Kode-kode tersebut ditulis dalam bilangan heksadesimal seperti contohnya: 74 02 75 F0 A3 dan seterusnya, tetapi yang tersimpan dalam bentuk fisik di memori adalah kode '1' dan '0' sebagai berikut: 01110100b(74h), 00000010b(02h), 01110101b(75h), 11110000b(F0h), 10100011b(A3h). Alangkah sulitnya mengerti atau membuat program bahasa mesin, karena hanya berupa kode-kode heksadesimal dan untuk menterjemahkannya diperlukan 'kamus' yang menterjemahkan apa itu kode 74h dan seterusnya.

Guna mempermudah pemrograman, diciptakanlah penterjemah atau dalam istilah software disebut kompiler (*compiler*), yang mengkonversi bahasa yang lebih tinggi (yang dimengerti manusia) ke bahasa mesin. Setingkat lebih tinggi dari bahasa mesin adalah bahasa assembler, selanjutnya di atas bahasa assembler adalah bahasa tingkat tinggi seperti BASIC, Pascal, C dan sebagainya. Makin tinggi tingkatnya makin mempermudah manusia untuk menulis program, contoh saat ini Borland Delphi bekerja pada sistem operasi Windows, merupakan software pemrograman visual berbasis orientasi objek, sangat mudah untuk memprogramnya. Untuk assembler, programmer harus berusaha berpikir lebih keras untuk proses-proses yang pada bahasa tingkat tinggi begitu mudah penulisannya.

Contoh perintah assembler adalah seperti berikut ini:

MOV DX,FF13

Yang tersimpan pada memori bukanlah “MOV DX,FF13” tapi merupakan bahasa mesin berupa bilangan heksadesimal “BA 13 FF”. MOV disebut mnemonic code, sedangkan BA adalah opcode (*operation code*) untuk mnemonic code MOV DX. Jadi setiap mnemonic code mempunyai opcode tersendiri.

Opcode (*operation code*) adalah bahasa mesin berupa kode-kode heksa yang dapat dijalankan oleh mikroprosesor. Sedangkan mnemonic code adalah bahasa yang mempermudah programmer melakukan programming mikroprosesor. Untuk mempelajari pemrograman mikroprosesor pada praktikum ini digunakan sistem minimum mikroprosesor bernama BGC-8088. Seperti yang telah dijelaskan sebelumnya bahwa diperlukan suatu kompiler untuk menterjemahkan bahasa assembler ke dalam bahasa mesin, pada BGC-8088 sendiri kompilernya sudah terintegrasi dalam program Monitor yang terprogram sebelumnya. Program monitor ini hampir mirip dengan program DEBUG yang dijalankan pada sistem operasi DOS (*Disk Operating System*).

B. SISTEM MINIMUM BGC-8088

Langkah awal dari pengoperasian BGC-8088 adalah memberikan catu daya. Setelah catu daya diberikan, pada panel LCD BGC-8088 tampak sebagai berikut:

BGC-8088 MICROENGINEER MONITOR V 3.10

*_
_

Ini menyatakan bahwa perangkat BGC-8088 telah siap untuk menerima dan menjalankan intruksi. Instruksi-instruksi dasar pemrograman, editing dan pengisian memori atau register dapat dilihat pada bagian di bawah ini.

1. Memasukkan program dalam bahasa assembler (*assemble*)

Instruksi : **A** <alamat_awal_program> + **CR** (*carriage return*)

Contoh : A100

2. Melihat program yang telah ditulis (*unassemble*)

Instruksi : **U** <alamat_awal_program_yang_telah_diassemble> + **CR**

3. Menyisipkan Program (*insert*)


Instruksi : **I** <alamat_tujuan> + **CR**

4. Melihat isi Register (register)

Instruksi : **R <nama_register> + CR**

BGC-8088 mempunyai **14 register 16-bit** yang masing-masing memiliki fungsi khusus. Beberapa diantaranya dapat dipecah menjadi dua register 8 bit yang didefinisikan sebagai register *low* dan *high*. Misalnya register 16-bit AX seperti di bawah ini:


Register AX = 7A41h



Bit ke-	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Biner	0	1	1	1	1	0	1	0	0	1	0	0	0	0	0	1


Dapat dipecah menjadi dua register AH (high) dan AL (low) yang masing-masing berkapasitas 8-bit.

Register AH = 7Ah



Bit ke-	7	6	5	4	3	2	1	0
Biner	0	1	1	1	1	0	1	0

Register AL = 41h



Bit ke-	7	6	5	4	3	2	1	0
Biner	0	1	0	0	0	0	0	1

Register-register dapat dibagi dalam lima golongan, yaitu:

a. General purpose register

terdiri dari :

- **AX (AH + AL)** = Accumulator Register
- **BX (BH + BL)** = Base Register
- **CX (CH + CL)** = Counter Register
- **DX (DH + DL)** = Data Register

b. Segment Register

- **CS** = Code Segment Register
- **DS** = Data Segment Register
- **SS** = Stack Segment Register
- **ES** = Extra Segment Register

c. Pointer Register

- **IP** = Instruction Pointer Register
- **SP** = Stack Pointer Register
- **BP** = Base Pointer Register

d. Index Pointer

- **SI** = Source Index Register
- **DI** = Destination Register

e. FLAG Register

Register flag ini adalah register 16 bit, fungsi register ini ialah mencatat tanda yang berkaitan dengan operasi khusus tentang kerja mikroprosesor yaitu :

- *Overflow flag (OF)*
- *Direction Flag (DF)*
- *Interrupt flag (IF)*
- *Trap flag (TF)*

Sedangkan tanda yang berkaitan dengan kerja mikroprosesor akibat operasi aritmatika dan logika yaitu :

- *Sign flag (SF)*
- *Zero flag (ZF)*
- *Auxiliary carry flag (AF)*
- *Parity flag (PF)*
- *Carry flag (CF)*

Posisi tiap bit pada register flag adalah

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLAG	XX	XX	XX	XX	OF	DF	IF	TF	SF	ZF	XX	AF	XX	PF	XX	CF

5. Mengisi sederet memori dengan data yang sejenis (fill)

Instruksi : **F** (alamat awal), (alamat akhir), (data)

6. Mengganti isi memori satu persatu (enter)

Instruksi : **E** (alamat awal) + **CR**

7. **Melihat isi memori (dump)**
Instruksi : **D** (alamat awal) + **CR**
8. **Meng-eksekusi- program step-by-step (trace)**
Instruksi : **T** = (alamat awal) + **CR**
Instruksi tersebut akan mengeksekusi satu line program, dan ditandai dengan adanya tampilan isi register. Untuk mengeksekusi line program berikutnya, tekan **T**.
9. **Mengeksekusi Program (go)**
Instruksi : **G** = (alamat awal) + **CR**
10. **Memindahkan blok memori atau menghapus isi blok (move)**
Instruksi : **M** (alamat awal),(alamat akhir),(alamat awal tujuan) + **CR**
11. **Konversi bilangan Desimal ke Hexa**
Instruksi : **J** (bilangan desimal) + **CR**
12. **Konversi bilangan Desimal ke Biner**
Instruksi : **B** (bilangan desimal) + **CR**
13. **Menghitung jumlah dan selisih dua bilangan heksa**
Instruksi : **H** (bil.1) , (bil.2) + **CR**

C. PERINTAH-PERINTAH DASAR BAHASA ASSEMBLER

Di bawah ini adalah beberapa perintah bahasa assembler yang dapat dijalankan pada sistem minimum BGC-8088. Penulisan perintah-perintah ini tidak membedakan antara huruf besar dan huruf kecil (*non case sensitive*).

- **MOV**

Biasanya digunakan untuk menyimpan nilai pada suatu register, selain itu, MOV dapat digunakan untuk menyalin isi suatu register ke register lainnya.

Sintaks:

```
MOV <register>, <nilai>
MOV <register>, [<alamat_memori>]
MOV <register1>, <register2>
MOV [<alamat_memori>], register
```

Contoh:

```
MOV AX,47A5    → menyimpan nilai 47A5h ke register AX
MOV DX,[200]   → menyalin isi memori pada alamat 200 ke register AL
MOV BX,AX      → menyalin isi register AX ke dalam register BX
MOV [300],AX   → mengisi alamat memori 300 dengan nilai yang
                terdapat pada register AX.
```

- **JMP**

Merupakan lompatan non-kondisional yang digunakan untuk melakukan lompatan (*jumping*) ke alamat memori tertentu.

Sintaks:

JMP <alamat_CS>:<alamat_IP>

JMP <alamat_IP>

Contoh:

JMP 0100:0100 → melompat ke instruksi dengan alamat CS = 100 dan IP = 100

JMP 100:100 → sama seperti perintah di atas

JMP 104 → melompat ke instruksi pada alamat IP = 100, alamat CS mengikuti isi register CS sebelumnya.

- **INC dan DEC**

INC (*increment*) digunakan untuk penambahan 1 pada suatu register.

DEC (*decrement*) digunakan untuk pengurangan 1 pada suatu register.

Sintaks:

INC <register>

DEC <register>

Contoh:

INC AX

DEC SI

- **NOP**

NOP (*no operation*) adalah perintah untuk memerintahkan mikroprosesor tidak melakukan apa-apa selama 1 siklus mesin. **NOP** sering digunakan untuk tujuan jeda atau tunda (*delay*).

- **INT 85**

Interrupt 85 digunakan untuk mengakhiri program.

- **INT 84**

Digunakan untuk mencetak suatu karakter yang kode ASCII-nya tersimpan di register AL.

ASCII (*American Standard Code for Information Interchange*)

ASCII adalah bentuk baku (internasional) yang digunakan untuk menyandikan karakter-karakter pada suatu mesin (komputer). Penyandian dapat berupa bilangan desimal ataupun heksadesimal.

Contoh :

Karakter = 0 1 2 3 A B c d j K

kode ASCII = 30 31 32 33 41 41 63 64 6A 6B ← Dalam Hex

Catatan : Spasi = 20h

Pindah baris = 0Dh

- **LOOP**

Adalah perintah yang digunakan untuk fungsi perulangan. LOOP menggunakan register CX untuk menentukan jumlah pengulangan yang akan dilakukan. Saat perintah LOOP dijalankan, prosesor akan mengurangi nilai CX dengan 1 dan memeriksa apakah CX bernilai 0? Jika Ya, maka eksekusi akan dilanjutkan pada perintah berikutnya, jika tidak maka akan terjadi lompatan ke alamat memori yang ditunjuk.

Sintaks:

```
LOOP <alamat_CS>:<alamat_IP>
LOOP <alamat_IP>
```

Contoh:

```
0100:0100  MOV CX,9
0100:0103  MOV AL,31
0100:0105  INT 84
0100:0107  INC AL
0100:0109  LOOP 105
0100:010B  INT 85
```

Output:

123456789

D. OPERASI PERCABANGAN DAN STATUS PORT

Merupakan suatu keharusan bagi suatu perangkat komputer untuk dapat berkomunikasi dengan dunia luar. Berbagai alat dapat dijadikan alat komunikasi, seperti layar peraga, keyboard, LED dsb.

BGC menyediakan beberapa sarana komunikasi antara lain layar peraga (LCD, Liquid Crystal Display), keyboard, led status (status port). Disamping itu juga tersedia beberapa instruksi yang berhubungan dengan hal ini, antara lain:

Keyboard driver :

INT 81H,

AH = 0 (service number 0) → membaca 1 karakter penekanan keyboard tanpa penekanan CR.

AH = 1 (service number 1) → membaca karakter dari keyboard dengan penekanan CR.

Komunikasi Port :

- **OUT DX,AL**

adalah perintah yang digunakan untuk mengirimkan data yang berada di register AL ke port yang alamatnya tersimpan di register DX.

- **IN AL,DX**

adalah perintah yang digunakan untuk membaca data dari port yang alamatnya tersimpan di register DX, data yang terbaca disimpan di register AL.

Percabangan :

- **CMP**

Digunakan untuk membandingkan dua buah operand dan hasil perbandingan tersebut memiliki status yang disimpan pada register flag yang bersangkutan. Operand dapat berupa register, alamat memori ataupun bilangan heksadesimal langsung, catatan, BGC-8088 tidak bisa melakukan perbandingan antara dua alamat memori.

Sintaks: **CMP <operand1>,<operand2>**

Contoh:

CMP AL,BL

CMP BL,[200]

CMP AX,56F2

Di bawah ini adalah perintah-perintah untuk lompatan bersyarat (*conditional jump*) yang biasanya mengikuti perintah CMP.

Sintaks:

<conditional_jump> <alamat_memori>

Contoh:

CMP AL,41

JE 100

- **JE (Jump If Equal)**

JE akan membuat mikroprosesor melakukan lompatan ke suatu alamat memori tertentu apabila 2 buah operand yang dibandingkan pada perintah CMP mempunyai nilai yang sama.

- **JNE (Jump If Not Equal)**

JNE akan membuat mikroprosesor melakukan lompatan ke suatu alamat memori apabila 2 buah operand yang dibandingkan pada perintah CMP tidak mempunyai nilai yang sama.

- **JG (Jump If Greater Than)**

JG akan membuat mikroprosesor melakukan lompatan ke suatu alamat memori apabila **operand1** lebih besar dibandingkan **operand2**.

- **JB (Jump if Below Than)**

JB akan membuat mikroprosesor melakukan lompatan ke suatu alamat memori apabila **operand1** lebih kecil dibandingkan **operand2**.

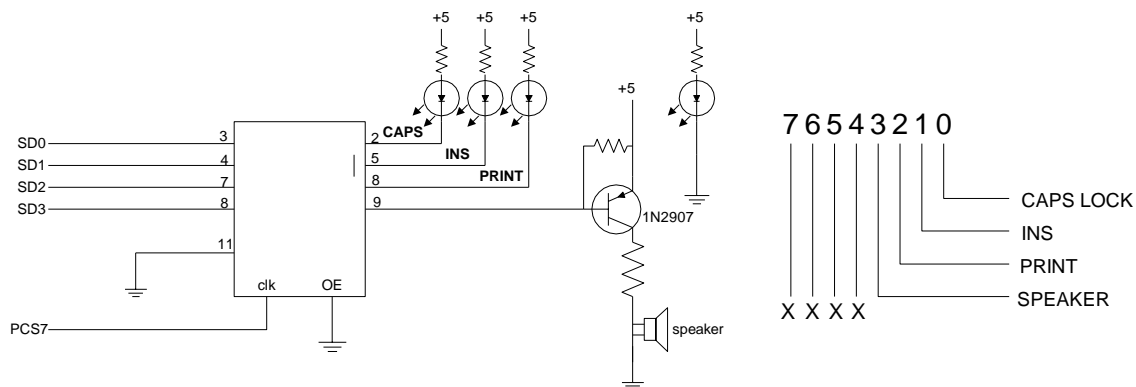
Dan masih banyak lagi perintah-perintah lompatan bersyarat lainnya :

- | | | | |
|--------------|---------------|--------------|--------------|
| • JGE | • JNBE | • JPE | • JLE |
| • JZ | • JNG | • JO | • JNP |
| • JNZ | • JNGE | • JNO | • JNC |
| • JS | • JBE | • JC | • JNL |
| • JNS | • JP | • JL | • JNB |

E. STATUS LED PORT

Status LED merupakan rangkaian yang terdiri dari 3 buah led yang berfungsi menandakan aktif atau tidaknya **caps**, **ins** dan **print**. Led tersebut dapat dinyalakan dengan menggunakan operasi manual, yaitu dengan penekanan tombol print, caps lock atau insert pada keyboard BGC-8088, atau dengan operasi software dengan memanfaatkan **alamat port FF70 - FF7F**. Konfigurasi Status LED adalah sebagai berikut:

D7	D6	D5	D4	D3	D2	D1	D0
x	x	x	x	Speaker	Print	Insert	Caps



Gambar 1. Status LED

-- dw --

[illegible]

Alat-alat :

- 1 Set BGC-8088
- Kertas kosong untuk catatan (bila perlu).

Prosedur percobaan :

Bagian 1

Modul 1.1A. Mnemonic dan Operation Code

1. Assemble program di bawah ini dengan menggunakan perintah **A100 + CR**.
2. Catat CS dan IPnya. Tanyakan asisten apabila ada yang kurang jelas.

CS	IP	Line Program
		Mov al,61
		Int 84
		Int 85

3. Jalankan program tersebut dengan menggunakan perintah **G=100 +CR**. Catat tampilan pada LCD BGC-8088.

--

4. Unassemble program yang telah dituliskan pada alamat memori 100 dengan menggunakan perintah **U100 + CR**. Catat **Operation Code** dari setiap perintah.

CS	IP	Opcode	Line Program
			Mov al,61
			Int 84
			Int 85

5. Salin Opcode dari program no. 4 ke alamat memori 200 dengan menggunakan perintah **E<alamat_memori>,<kode>**. Pisahkan antar 2 digit opcode dari program di atas dengan menggunakan koma (,).

Contoh:

* E200,B0,41,... + CR

6. Jalankan Opcode yang telah dituliskan tersebut dengan menggunakan perintah **G=200 +CR**. Apa yang terjadi? Buat kesimpulan dari percobaan yang anda lakukan. Apa yang dimaksud dengan Mnemonic Code dan Operation Code?

.....

Modul 1.1B. INT 84 LCD Driver

1. Assemble program dibawah ini pada sistem minmum BGC-8088 dengan menggunakan perintah **A<alamat program anda> +CR**.
2. Catat **CS : IP** program anda ketika anda mengetikannya di BGC.
(tanyakan pada asisten jika ada yang kurang jelas)

CS	IP	Line Program
		Mov al,61
		Mov cx,a
	**	Push cx
		Int 84
		Mov cx,7fff
	*	Loop *
		Pop cx
		Inc al
		Loop **
		Int 85

Program 1.1b. Program menampilkan karakter

3. Eksekusi program secara langsung dengan menggunakan perintah **G=<alamat program anda>**.

Output program pada LCD BGC-8088:

Kesimpulan :

.....

Modul 1.1C. Operasi String dengan memanfaatkan INT 84

1. Assemble program contoh berikut ini pada sistem minimum BGC-8088. Jangan lupa untuk menuliskan **CS:IP** program anda pada data pengamatan.
2. Isi alamat memori **200** dengan data – **Laboratorium Mikroprosesor** -- menggunakan perintah
e<alamat memori>,'<data yang diminta>'.
3. Sebagai contoh anda diminta mengisi alamat 200 dengan data string -- **Laboratorium Mikroprosesor** – maka penulisannya adalah
e200,'Laboratorium Mikroprosesor'+CR.
(tanyakan pada asisten jika kurang jelas).

CS	IP	Line Program
		Mov si,200
		Mov cx,1A
	**	Push cx
		Mov al, [si]
		Int 84
		Mov cx, 5fff
	*	Loop *
		Pop cx
		Inc si
		Loop **
		Int 85

Program 1.1c. Int 84 untuk Menampilkan String.

4. Eksekusi program secara langsung dengan menggunakan perintah
G=<alamat program anda>.

Output program pada LCD BGC-8088:

Kesimpulan :

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Modul 1.1D. Operasi percabangan dan Int 81 (keyboard driver)

1. Assemble line program di bawah ini ke dalam BGC-8088 dengan menggunakan alamat awal 100. Dan catat **CS:IP** program pada kolom **CS** dan **IP**.

CS	IP	Line Program
	A	Mov si,200
		Mov di,300
		Mov ah,00
		Int 81
		Cmp al,61
		Je B
		Cmp al,62
		Je C
		Jmp A
	B	Mov cx,15
	D	Mov al,[si]
		Int 84
		Inc si
		Loop D
		Mov al,0d
		Int 84
		Jmp A
	C	Mov cx,d
	E	Mov al,[di]
		Int 84
		Inc di
		Loop E
		Mov al,0d
		Int 84
		Jmp A
		Int 85

Program 1.d. Program Operasi Percabangan dan Int 81.

2. Setelah selesai isi alamat memori 200 dengan menggunakan perintah **E<alamat memori>,<data>** dengan data :
 “Universitas Gunadarma”
dan alamat memori 300 dengan data:
 “Mikroprosesor”

3. Setelah selesai eksekusi program secara langsung dengan menggunakan perintah **G<alamat program anda>**, lalu lakukan penekanan **tombol A dan B** pada keyboard BGC dan berikan kesimpulan anda mengenai hasil eksekusi program.
(tanyakan pada asisten jika ada yang kurang jelas)

Output program jika tombol A ditekan:

.....

Output program jika tombol B ditekan:

.....

Kesimpulan dari cara kerja program:

.....
.....
.....
.....

Bagian 2

Modul 1.2A. Status LED Port

1. Buktikan kebenaran jawaban laporan pendahuluan no. 4 anda dengan mengirimkan data-data ke alamat Status LED Port (**FF70 s/d FF7F**). Dengan mengetikkan perintah **O<alamat_status_port>,<data>**. O adalah huruf O, bukan angka nol. **Lingkari Status LED yang menyala.**

Contoh : *OFF73,7 + CR

a. Caps

Perintah :

LED Status Port yang menyala [**Caps** | Insert | Print]

b. Insert

Perintah :

LED Status Port yang menyala [**Caps** | Insert | Print]

c. Print

Perintah :

LED Status Port yang menyala [**Caps** | Insert | Print]

d. Caps dan Insert

Perintah :

LED Status Port yang menyala [**Caps** | Insert | Print]

2. Setelah terbukti assemble program dengan menggunakan perintah **A<alamat program anda>**, jangan lupa untuk mencatat CS dan IP-nya.

CS	IP	Line Program	Keterangan
		Mov dx,<alamat status port;FF70-FF7F>	
		Mov cx,f	
	D	Push cx	
		Mov al,<data untuk Print>	
		Out dx,al	
		Mov cx, 7fff	
	A	Loop A	
		Mov al,<data untuk Ins>	
		Out dx,al	
		Mov cx,7fff	
	B	Loop B	
		Mov al,<data untuk Caps>	
		Out dx,al	
		Mov cx,7fff	
	C	Loop C	
		Pop cx	
		Loop D	
		Int 85	

Program 1.e. Program Status LED Port

Output Program saat dijalankan:

Print	Ins	Caps

Berulang sebanyak : Kali.

Kesimpulan :

.....

.....

.....

.....

.....

.....

.....

.....

This image shows a full page of a document template. It consists of approximately 30 horizontal rows. Each row is defined by two parallel dotted lines, creating a series of uniform gaps for writing. The entire page is otherwise blank, with no margins, text, or other markings.

