**LAPORAN**

**PEMOGRAMAN ASSEMBLER**

**MEMORY ACCSESS**



Nama : Syarif Hidayatullah

NIM : 3.34.12.0.21

No.Job : 2

Kelas : IK 2 A

**PROGRAM STUDI TEKNIK INFORMATIKA**

**JURUSAN TEKNIK ELEKTRO**

**POLITEKNIK NEGERI SEMARANG**

**TAHUN AJARAN 2013/2014**

**BAB 1**

**MEMORI AKSES**

1. **Tujuan Instruksi Khusus**

Setelah menyelesaikan praktek ini mahasiswa dapat :

1. Memahami dan mengetahui struktur bahasa Assembly
2. Memahasmi konsep dari memori akses
3. **Pokok Bahasan**

**Memory Access**

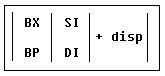
Ada 4 register yang dapat digunakan untuk mengakses memory yaitu BX, SI, DI, BP , register register tersebut dapat dikombinasikan didalam symbol tanda kurung kotak [], kita bisa mendapatkan lokasi memori yang berbeda. Kombinasi ini didukung dengan mode pengalamatan ( addressing modes) sebagai berikut :

|  |  |  |
| --- | --- | --- |
| [BX+SI] [BX+DI] [BP+SI]  [BP+DI] | [SI] [DI] d16 ( variabel offset only )  [BX] | [BX+SI+d8]  [BX+DI+d8]  [BP+SI+d8]  [BP+DI+d8] |
| [SI+d8]  [DI+d8]  [BP+d8]  [BX+d8] | [BX+SI+d16]  [BX+DI+d16]  [BX+SI+d16]  [BX+DI+d16] | [SI+d16]  [DI+d16]  [BP+d16]  [BX+d16] |

**d8**-merupakan penggantian nilai yang terdekat untuk 8 bit signed ( contoh : 22,55h,-1,etc...)

**d16**-merupakan penggantian nilai yang terdekat untuk 16 bit signed ( contoh : 300, 5517h, -259, etc...)

Penggantian dapat menjadi nilai yang terdekat atau offset dari variabel atau keduanya . Jika ada beberapa nilai , assembler mengevaluasi semua nilai tersebut dan mengkalkulasi sebagai nilai tunggal . Penggantian di dalam atau diluar symbol [] , maka assembler men-generates kode mesin yang sama dengan kedua cara tersebut . Penggantian nilai signed negative . Umumnya compiler memperhatikan perbedaan antara d8 dan d16, dan men-generate kode mesin yang diperlukan . contoh , diasumsikan DS = 100, BX =30, SI =70. Mode pengalamatannya adalah: [BX+SI] + 25 yaitu dikalkulasi oleh processor ke physical addres 100\*16+30+70+25=1725 . Defaultnya segmen register DS adalah digunakan untuk semua mode kecuali register BP , untuk hal tersebut segmen register SS digunakan . Jadi ada cara mudah untuk mengingat semua kemungkinan tersebut dengan menggunakan bagan di bawah ini :



Kita dapat membentuk semua kombinasi yang valid dengan mengambil hanya satu item tiap kolom atau melewati kolom tersebut dengan tidak mengambil darinya , seperti yang kita lihat BX dan BP tidak pernah bersama-sama. SI dan DI demikian juga , contoh disini merupakan mode pengalamatan yang valid : [BX+5] , [BX+SI] , [DI+BX-4], nilai dalam purpose register (CS,DS,SS,ES) disebut segment, dan nilai dalam purpose register (BX,SI,DI,BP) disebut offset. Saat DS berisi nilai 1234h dan SI berisi nilai 7890h hal tersebut dapat ditulis sebagai 1234:7890. Physical address akan menjadi 1234h\*10h+7890h = 19BD0h. Jika nol ditambahkan ke angka decimal lalu di kalikan 10, maka 10h=16, jadi jika nol ditambahkan pada nilai hexadecimal , hal tersebut dikalikan dengan 16, contoh:

7h = 7

70h = 7\*16=112

Hal tersebut dapat dikatakan bahwa dalam pandangan compiler terhadap tipe data , prefix yang digunakan adalah :

byte ptr- untuk byte.

word ptr-untuk word (2 byte)

Contoh:

byte ptr [BX] ; mengakses byte.

word ptr [BX] ; mengakses word.

Assembler mendukung prefix seperti dibawah:

b.-untuk byte ptr

w.-untuk word ptr

Dalam banyak kasus dapat mengkalkulasi tipe data secara otomatis.

1. **Alat dan Bahan**
2. Komputer dengan OS min Windows XP
3. Emu8086
4. **Instruksi Kerja**

**Instruksi MOV**

* Mengutip operan ke dua ( source ) ke operand pertama (destination)
* Operan source dapat suatu nilai terdekat, general-purpose register atau lokasi memory
* Register destination dapat suatu general-purpose register, atau lokasi memory.
* Kedua operand harus mempunyai ukuran yang sama , baik byte atau word

|  |
| --- |
| These types of operands are supported:  MOV REG, memory  MOV memory, REG  MOV REG,REG  MOV memory , immediate  MOV REG,immediate  **REG** : AX, BX, CX, DX, AH, AL, BL , BH, CH, CL, DH, DL, DI, BP,SP.  **Memory** : [BX],[BX+SI+7], variabel , etc ...  **Immediate** : 5,-24,3Fh,10001101b, etc ... |

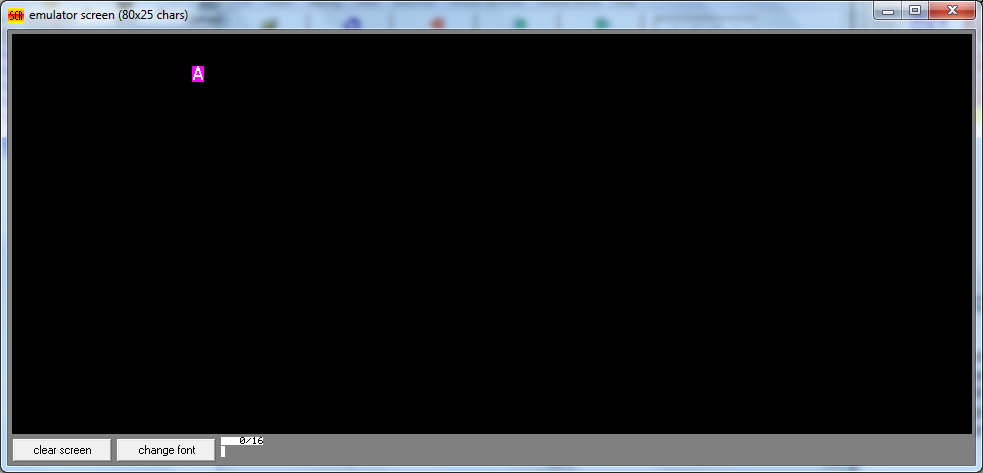
|  |
| --- |
| For segment registers only these types of **MOV** are supported:  MOV SREG , memory  MOV memory, SREG  MOV REG,SREG  MOV SREG,REG  **SREG :** DS,ES,SS,CX,DX,AH,AL,BL,BH,CH,CL,DH,DL,DI,SI,BP,SP.  **Memory** : [BX],[BX+SI+7], variabel , etc |

Instruksi MOV tidak dapat digunakan untuk mengeset nilai register CS dan IP

Contoh program mendemontrasikan penggunaan instruksi MOV:

|  |
| --- |
| ORG 100h ; directive yg diperlukan utk 1 segment .com program  MOV AX, 0b800h ; set AX ke nilai hex dr B800h.  MOV DS,AX ; copy nilai AX ke DS.  MOV CL,’A’ ; set CL ke kode ASCII ‘A’ , yaitu 41h.  MOV CH,1101\_1111b ; set CH ke nilai binary  MOV BX,15Eh ;set BX ke 15Eh.  MOV [BX],CX ; copy isi CX ke memory pada alamat B800:015E  RET ; kembali ke OS |

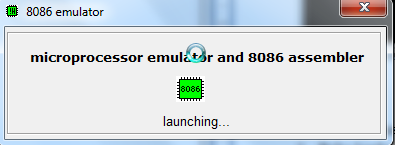
Setelah mengetikan kode diatas tekan tombol [Compile and Emulate] atau F5. Kemudian Window emulator akan terbuka dengan program yang di loaded, lalu click tombol [Single Step] dan amati nilai register . Tanda “;” adalah digunakan untuk membuat komentar program dan akan diabaikan oleh compiler.. Keluaran dari program diatas seharusnya seperti ini:



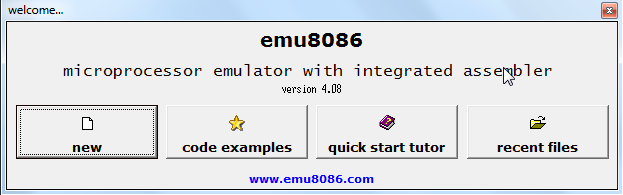
**PERCOBAAN**

**Percobaan**

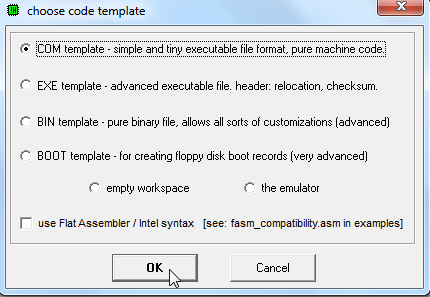
1. Buka aplikasi emu 8086



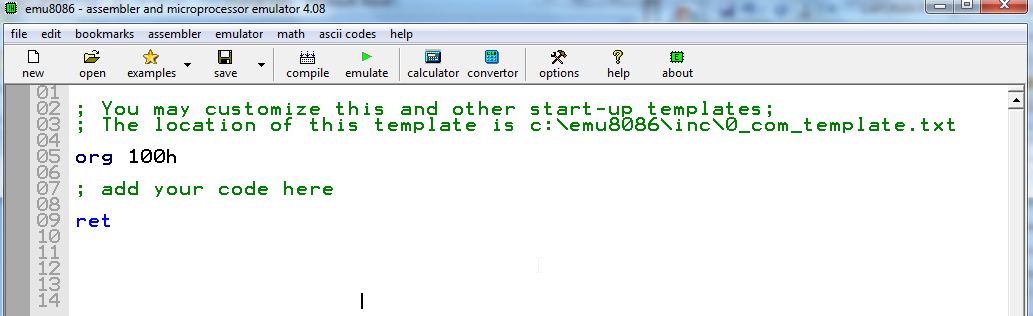
1. Buka lembar kerja baru , dengan cara klik New



1. Pilih COM template , lalu klik OK



1. Kemudian akan tampil layar kerja berikut , dan ketikan source kode yang diinginkan



1. Setelah mengetikkan kode yang diinginkan simpan program tersebut
2. Jalankan program tersebut dengan meilih icon



1. Untuk menelisik hasil pilih

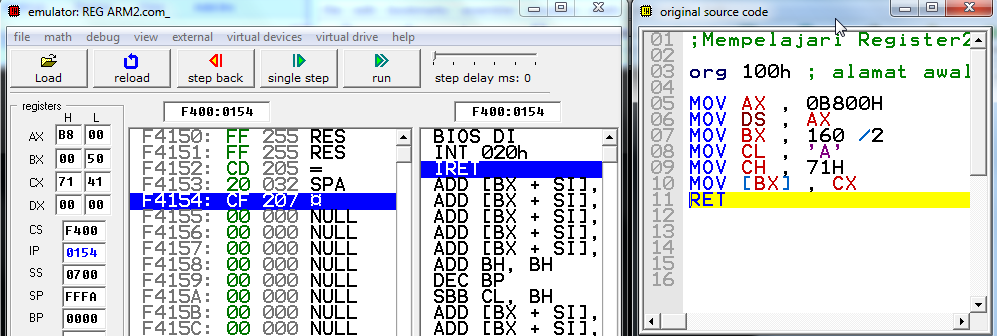


1. Untuk melihat hasilnya klik RUN

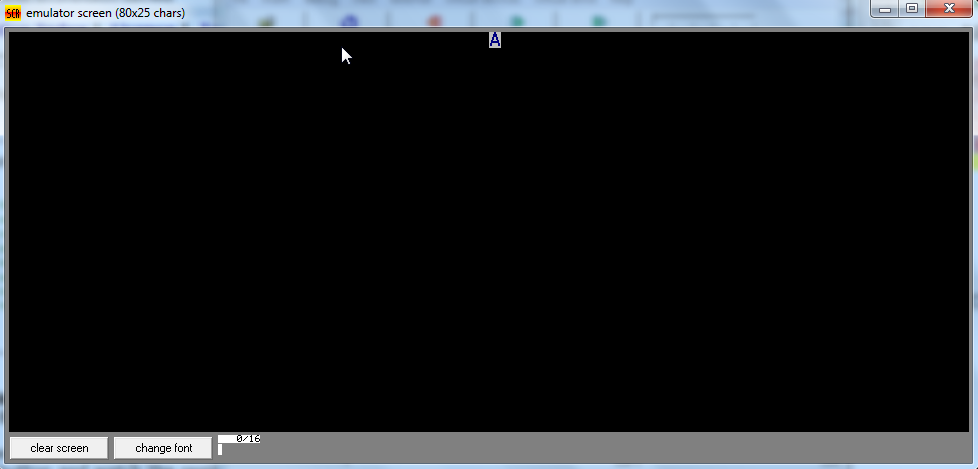


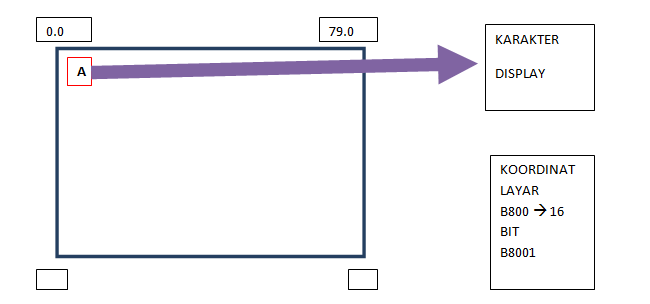
1. **Latihan**

* **Percobaan satu**

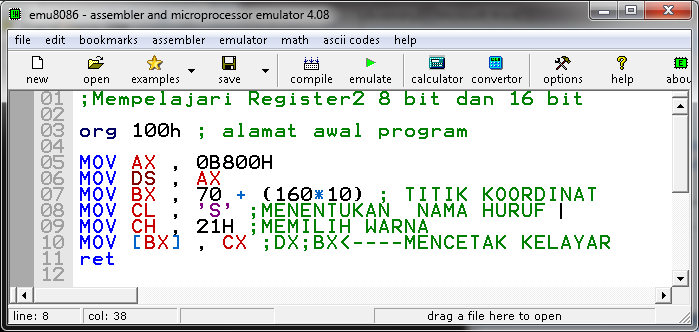


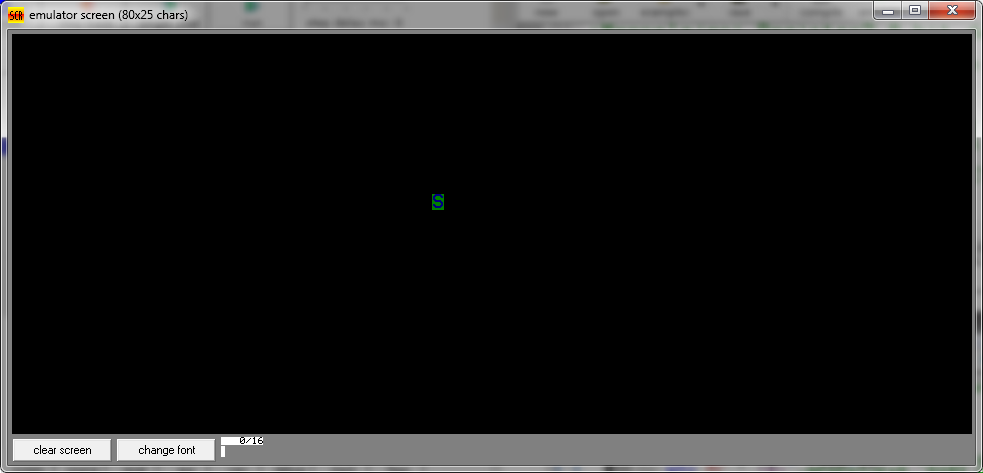
**Hasil**

****

****

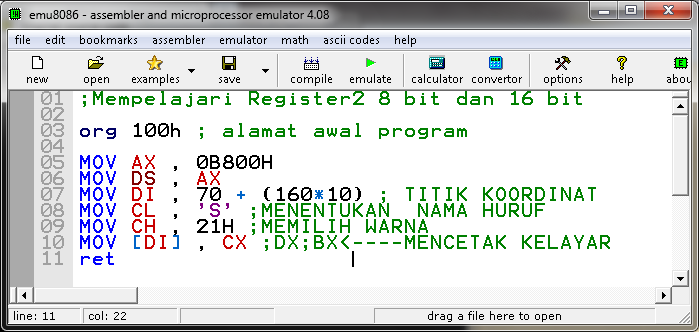
|  |
| --- |
| ;Mempelajari Register2 8 bit dan 16 bit  org 100h ; alamat awal program  MOV AX , 0B800H  MOV DS , AX  MOV BX , 70 + (160\*10) ; TITIK KOORDINAT  MOV CL , 'S' ;MENENTUKAN NAMA HURUF  MOV CH , 21H ;MEMILIH WARNA  MOV [BX] , CX ;DX;BX<----MENCETAK KELAYAR  ret |

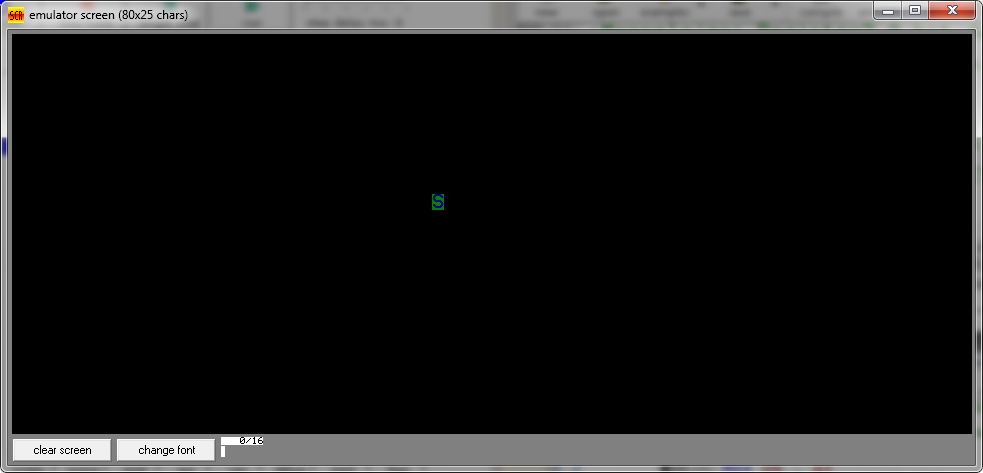
****

****

* **Percobaan dua**

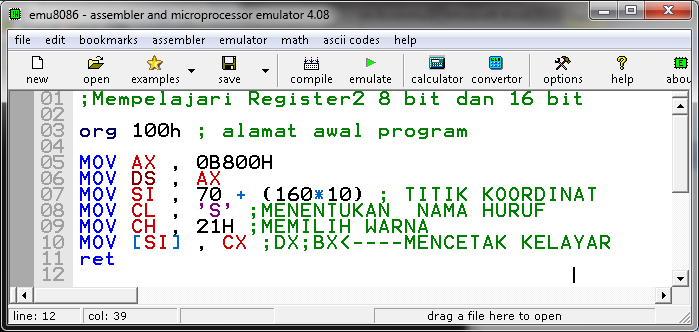
|  |
| --- |
| ;Mempelajari Register2 8 bit dan 16 bit  org 100h ; alamat awal program  MOV AX , 0B800H  MOV DS , AX  MOV DI , 70 + (160\*10) ; TITIK KOORDINAT  MOV CL , 'S' ;MENENTUKAN NAMA HURUF  MOV CH , 21H ;MEMILIH WARNA  MOV [DI] , CX ;DX;BX<----MENCETAK KELAYAR  ret |

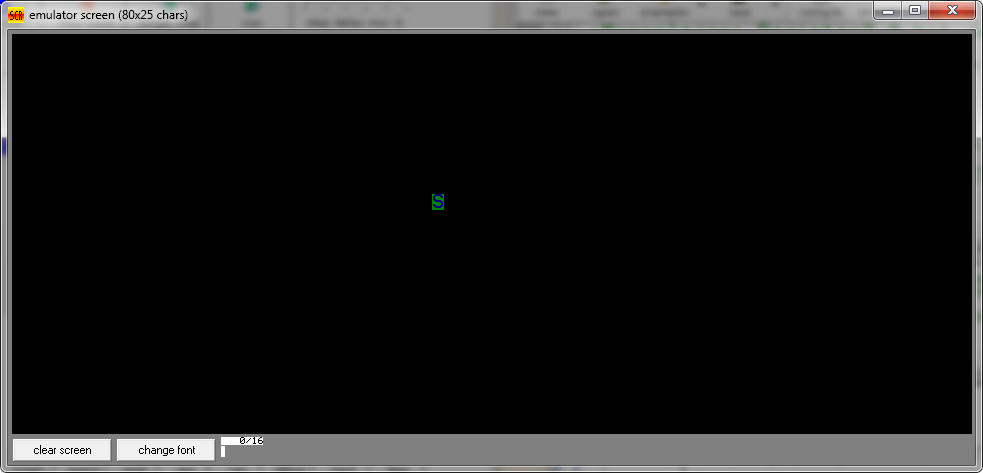
****

****

* **Percobaan Tiga**

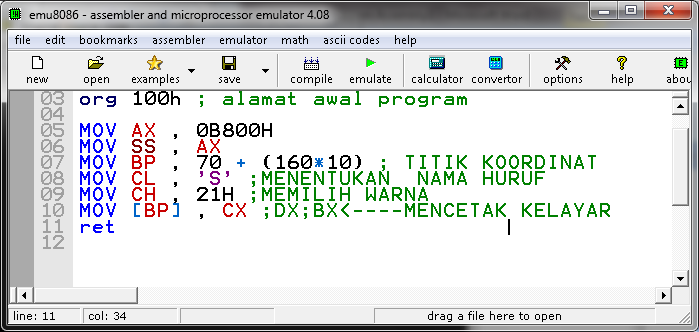
|  |
| --- |
| ;Mempelajari Register2 8 bit dan 16 bit  org 100h ; alamat awal program  MOV AX , 0B800H  MOV DS , AX  MOV SI , 70 + (160\*10) ; TITIK KOORDINAT  MOV CL , 'S' ;MENENTUKAN NAMA HURUF  MOV CH , 21H ;MEMILIH WARNA  MOV [SI] , CX ;DX;BX<----MENCETAK KELAYAR  ret |

****

****

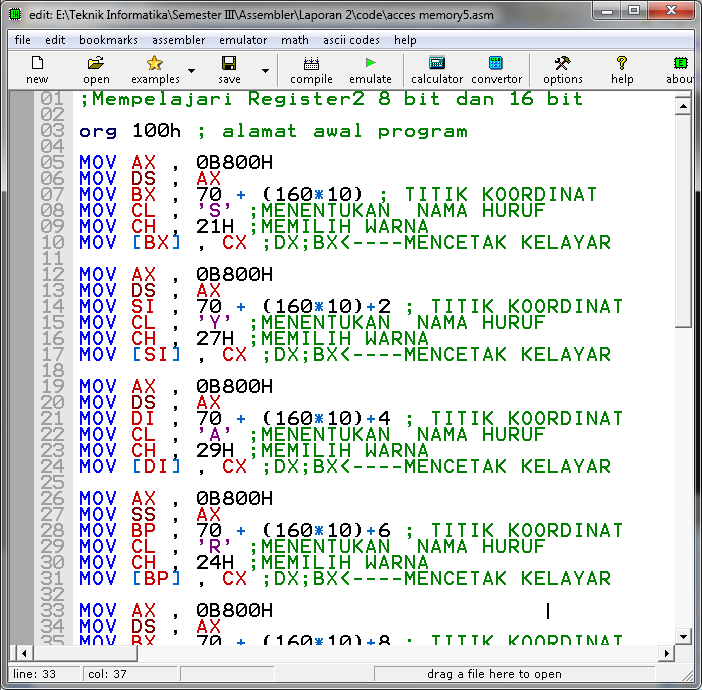
* **Percobaan Empat**

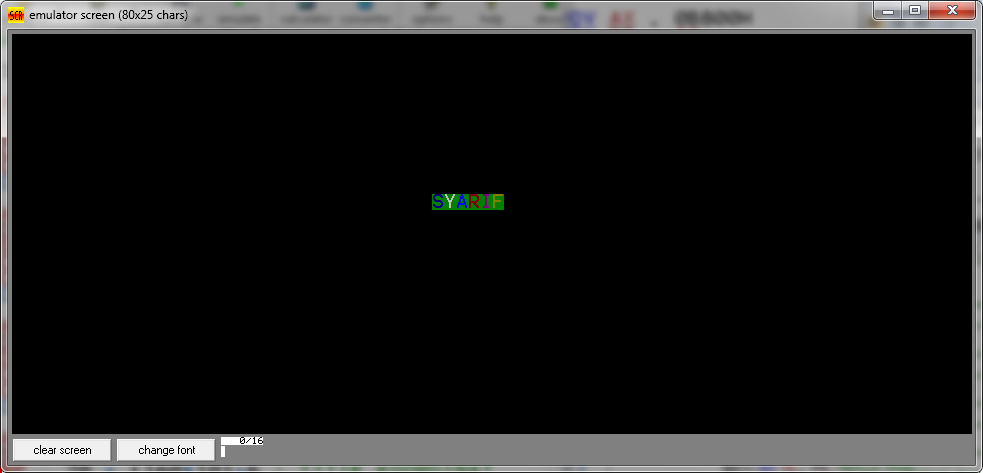
|  |
| --- |
| ;Mempelajari Register2 8 bit dan 16 bit  org 100h ; alamat awal program  MOV AX , 0B800H  MOV SS , AX  MOV BP , 70 + (160\*10) ; TITIK KOORDINAT  MOV CL , 'S' ;MENENTUKAN NAMA HURUF  MOV CH , 21H ;MEMILIH WARNA  MOV [BP] , CX ;DX;BX<----MENCETAK KELAYAR  ret |

****

* **Percobaan Lima**

|  |
| --- |
| ;Mempelajari Register2 8 bit dan 16 bit  org 100h ; alamat awal program  MOV AX , 0B800H  MOV DS , AX  MOV BX , 70 + (160\*10) ; TITIK KOORDINAT  MOV CL , 'S' ;MENENTUKAN NAMA HURUF  MOV CH , 21H ;MEMILIH WARNA  MOV [BX] , CX ;DX;BX<----MENCETAK KELAYAR  MOV AX , 0B800H  MOV DS , AX  MOV SI , 70 + (160\*10)+2 ; TITIK KOORDINAT  MOV CL , 'Y' ;MENENTUKAN NAMA HURUF  MOV CH , 27H ;MEMILIH WARNA  MOV [SI] , CX ;DX;BX<----MENCETAK KELAYAR  MOV AX , 0B800H  MOV DS , AX  MOV DI , 70 + (160\*10)+4 ; TITIK KOORDINAT  MOV CL , 'A' ;MENENTUKAN NAMA HURUF  MOV CH , 29H ;MEMILIH WARNA  MOV [DI] , CX ;DX;BX<----MENCETAK KELAYAR  MOV AX , 0B800H  MOV SS , AX  MOV BP , 70 + (160\*10)+6 ; TITIK KOORDINAT  MOV CL , 'R' ;MENENTUKAN NAMA HURUF  MOV CH , 24H ;MEMILIH WARNA  MOV [BP] , CX ;DX;BX<----MENCETAK KELAYAR  MOV AX , 0B800H  MOV DS , AX  MOV BX , 70 + (160\*10)+8 ; TITIK KOORDINAT  MOV CL , 'I' ;MENENTUKAN NAMA HURUF  MOV CH , 25H ;MEMILIH WARNA  MOV [BX] , CX ;DX;BX<----MENCETAK KELAYAR  MOV AX , 0B800H  MOV DS , AX  MOV BX , 70 + (160\*10)+10; TITIK KOORDINAT  MOV CL , 'F' ;MENENTUKAN NAMA HURUF  MOV CH , 26H ;MEMILIH WARNA  MOV [BX] , CX ;DX;BX<----MENCETAK KELAYAR  RET |





1. **KESIMPULAN**
2. Ada 4 register yang dapat digunakan untuk mengakses memory yaitu BX, SI, DI, BP , register register tersebut dapat dikombinasikan didalam symbol tanda kurung kotak [], kita bisa mendapatkan lokasi memori yang berbeda. Kombinasi ini didukung dengan mode pengalamatan ( addressing modes)