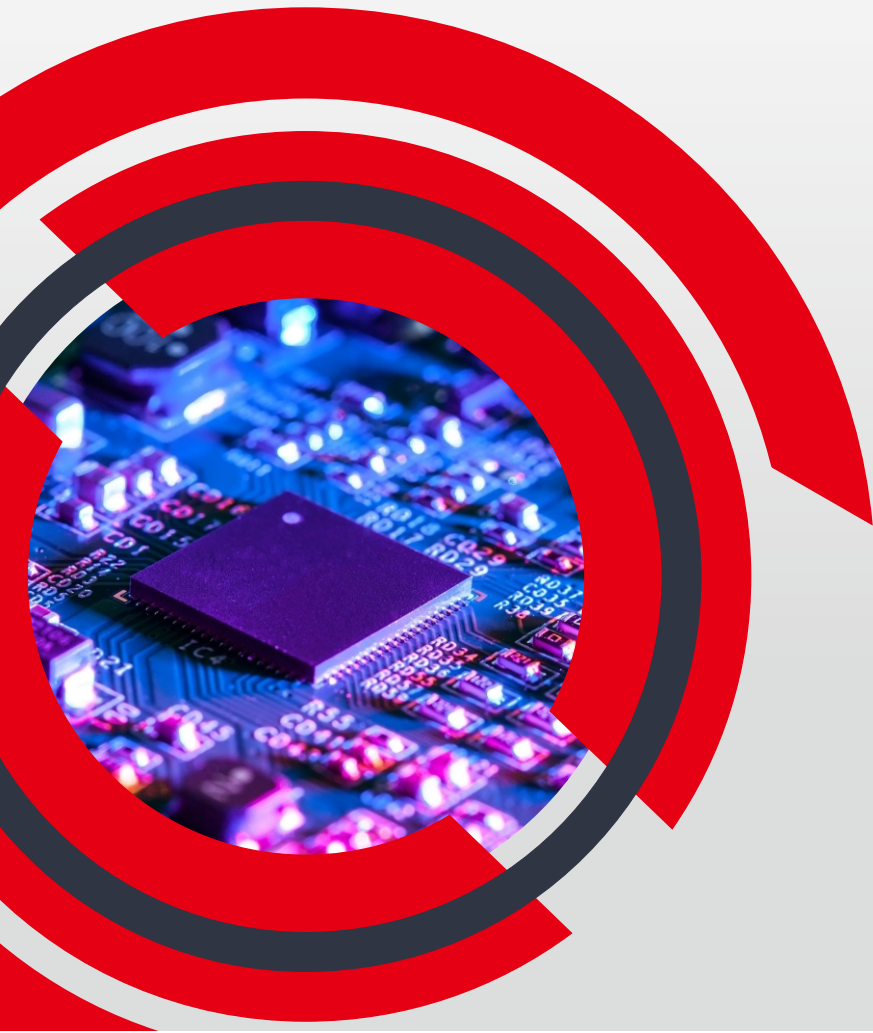


Dr. Unang Achlison, S.T., M.Kom

# Dasar Mikrokontroler I

**Eksperimen AT89C51 menggunakan  
TopView Simulator**



YAYASAN PRIMA AGUS TEKNIK

# Dasar Mikrokontroler I

**Eksperimen AT89C51 menggunakan  
TopView Simulator**

**Dr. Unang Achlison, S.T., M.Kom**



**YAYASAN PRIMA AGUS TEKNIK**  
Jl. Majapahit No. 605 Semarang  
Telp. (024) 6723456. Fax. 024-6710144  
Email : [penerbit\\_ypat@stekom.ac.id](mailto:penerbit_ypat@stekom.ac.id)

# **Dasar Mikrokontroler I: Eksperimen AT89C51 menggunakan TopView Simulator**

## **Penulis :**

Dr. Unang Achlison, S.T., M.Kom

**ISBN : 9 786236 141588**

## **Editor :**

Dr. Mars Caroline Wibowo. S.T., M.Mm.Tech

## **Penyunting :**

Dr. Joseph Teguh Santoso, M.Kom.

## **Desain Sampul dan Tata Letak :**

Irdha Yuniyanto, S.Ds., M.Kom

## **Penebit :**

Yayasan Prima Agus Teknik

## **Redaksi :**

Jl. Majapahit no 605 Semarang

Telp. (024) 6723456

Fax. 024-6710144

Email : [penerbit\\_ypat@stekom.ac.id](mailto:penerbit_ypat@stekom.ac.id)

## **Distributor Tunggal :**

### **Universitas STEKOM**

Jl. Majapahit no 605 Semarang

Telp. (024) 6723456

Fax. 024-6710144

Email : [info@stekom.ac.id](mailto:info@stekom.ac.id)

Hak cipta dilindungi undang-undang

Dilarang memperbanyak karya tulis ini dalam bentuk dan dengan cara apapun tanpa ijin dari penulis

## **KATA PENGANTAR**

Segala puji bagi Allah SWT atas segala nikmat, rahmat, dan karunia-Nya sehingga penulis dapat menyelesaikan buku berjudul Dasar Mikrokontroler I: Eksperimen AT89C51 menggunakan TopView Simulator.

Isi buku ini disajikan secara praktis dan lengkapi contoh eksperimen. Cakupan bidang yang dibahas buku ini sangat membantu dan berperan sebagai sumbangsih pemikiran dalam mendukung pemecahan permasalahan yang muncul pada mikrokontroler ATMEL AT89C51, karakterisasi dan aplikasi dalam bidang kendali elektronik.

Buku ini disusun secara integratif antar disiplin ilmu yaitu pengetahuan mikrokontroler, elektronika analog dan pemrograman. Buku diktat ini bisa dijadikan bahan belajar mikrokontroler ATMEL AT89C51. Buku ini masih banyak kekurangan dan jauh dari kesempurnaan. penulis sangat terbuka menerima kritik dan saran membangun sehingga secara bertahap penulis dapat memperbaikinya.

Semoga buku diktat ini bermanfaat dan bisa memberikan pencerahan bagi pembaca untuk memahami mikrokontroler ATMEL AT89C51.

Penulis

**Dr. Unang Achlison, S.T., M.Kom**

# DAFTAR ISI

KATA PENGANTAR .....	iii
DAFTAR ISI .....	iv
DAFTAR TABEL .....	v
DAFTAR GAMBAR .....	vi
BAB I Mikrokontroler .....	1
A. Pendahuluan .....	1
B. Mikrokontroler AT89C51 .....	3
1. Arsitektur AT89S51 .....	3
2. Fungsi Pin pada AT89S51 .....	5
3. Bahasa Pemrograman pada AT89S51 .....	6
C. Rangkuman .....	10
D. Latihan .....	11
E. Rujukan .....	33
F. Bacaan yang dianjurkan .....	33
BAB II Mikrokontroler Simulator .....	34
A. Pendahuluan .....	34
B. TopView Simulator .....	34
1. Tahapan menginstal .....	34
2. Buka Program .....	36
3. Penulisan File Awal .....	37
4. Simulasi Mikrokontroler .....	37
C. Rangkuman .....	46
D. Latihan .....	46
E. Rujukan .....	60
F. Bacaan yang dianjurkan .....	60

## DAFTAR TABEL

Tabel 1.1. Data Display 7 Segmen .....	13
Tabel 1.2 Pin dan Fungsi .....	15
Tabel 1.3 Peta Karakter .....	17
Tabel 1.4 Data Resolusi ADC .....	21

## DAFTAR GAMBAR

Gambar 1-1 Susunan pin dan arsitektur dari keluarga 8051 .....	3
Gambar 1-2 Rangkaian Sistem Minimum untuk AT89S51 .....	4
Gambar 1-3 Diagram Pin pada AT89S51 .....	5
Gambar 1-4 Langkah-langkah pemrograman mikrokontroler .....	6
Gambar 1.5. Rangkaian Interface Push Button dan LED .....	11
Gambar 1.6. Rangkaian 7 Segmen .....	12
Gambar 1.7 Modul 7 Segment tunggal .....	12
Gambar 1.8. Modul LCD Karakter 2x16 .....	14
Gambar 1.9. Rangkaian LCD Karakter 2x16 .....	14
Gambar 1.10. Lokasi memori display LCD Karakter .....	16
Gambar 1.11 Konfigurasi pin ADC0804 .....	19
Gambar 1.12 Rangkaian ADC0804 .....	12
Gambar 1.13 Rangkaian interface keypad 4x4 .....	26
Gambar 1.14 Motor Stepper .....	31
Gambar 1.15. Ilustrasi sebuah kompas dengan elektromagnet .....	31
Gambar 2.1. Instalasi software Topview Simulator .....	34
Gambar 2.2. Didirektori akan diletakkan file program .....	35
Gambar 2.3. Instalasi software pada progress bar .....	35
Gambar 2.4. Select Device .....	36
Gambar 2.5. Program Topview Simulator .....	36
Gambar 2.6. Program Topview Simulator .....	37
Gambar 2.7. Menu Select Device .....	38
Gambar 2.8. Menu Utama Topview Simulator .....	39
Gambar 2.9. Menu membuka file new.asm .....	39
Gambar 2.10. Sumber kode assembler .....	40
Gambar 2.11. Mentimpan file sumber kode assembler .....	40
Gambar 2.12. Kompilasi sumber kode assembler .....	41
Gambar 2.13. Proses kompilasi sumber kode assembler .....	41
Gambar 2.14. Hasil kompilasi sumber kode assembler .....	42
Gambar 2.15. External Modul Setting .....	43
Gambar 2.16. LED Modul Port Setting .....	43
Gambar 2.17. Menampilkan LED Modul .....	44
Gambar 2.18. Tampilkan modul LED .....	44
Gambar 2.19. Modul LED activating levels 1 .....	45

Gambar 2.20. Modul LED activating levels 0 .....	45
Gambar 2.21. Eksperimen Button dan LED .....	47
Gambar 2.22. Eksperimen Button OFF dan LED OFF .....	48
Gambar 2.23. Eksperimen Button ON dan LED ON .....	48
Gambar 2.24. Eksperimen Dekade Counter pada LCD .....	51
Gambar 2.25. Eksperimen Button OFF maka LCD = 0 .....	52
Gambar 2.26. Eksperimen Button OFF maka LCD = 2 .....	52
Gambar 2.27. Eksperimen Dekade Counter pada 7-Segment .....	54
Gambar 2.28. Button OFF dan Counter pada 7-Segmen = 1 .....	54
Gambar 2.29. Button OFF dan Counter pada 7-Segmen = 2 .....	55
Gambar 2.30. Eksperimen Sensor Suhu pada LCD .....	59
Gambar 2.31. Eksperimen Sensor Suhu LM35 .....	59
Gambar 2.32. Eksperimen Sensor Suhu LM35 dan Solder .....	60



## BAB I

### Mikrokontroler

#### A. Pendahuluan

Sistem berbasis mikroprosesor telah berkembang mencakup berbagai macam jenis sistem kontrol, yang secara umum disebut sebagai komputer. Terdapat bermacam tingkatan komputer, tiga jenis yang paling banyak dipergunakan adalah komputer personal (Personal Computer), mikrokontroler, dan Programmable Logic Controller (PLC). Ketiganya mempunyai penggunaan yang berbeda-beda.

Komputer personal banyak dipergunakan sebagai pengolah kata, spreadsheet, pengolahan basis data, drafting, dan desain grafis. Mikrokontroler dapat dianggap komputer dengan ukuran dan kapasitas kecil banyak dipakai dalam berbagai macam produk sebagai sistem yang sering disebut sebagai *embeded system*. PLC lebih banyak penggunaannya dalam aplikasi industri karena kemudahan dalam pemrograman dan wiring, serta ketahanan terhadap kondisi yang terdapat pada lingkungan mesin industri.

Mikrokontroler pada dasarnya adalah komputer dalam satu chip, yang di dalamnya terdapat mikroprosesor, memori, jalur Input/Output (I/O) dan perangkat pelengkap lainnya. Kecepatan pengolahan data pada mikrokontroler lebih rendah jika dibandingkan dengan PC. Pada PC kecepatan mikroprosesor yang digunakan saat ini telah mencapai orde GHz, sedangkan kecepatan operasi mikrokontroler pada umumnya berkisar antara 1 – 16 MHz. Begitu juga kapasitas RAM dan ROM pada PC yang bisa mencapai orde Gbyte, dibandingkan dengan mikrokontroler yang hanya berkisar pada orde byte/Kbyte. Meskipun kecepatan pengolahan data dan kapasitas memori pada mikrokontroler jauh lebih kecil jika dibandingkan dengan komputer personal, namun kemampuan mikrokontroler sudah cukup untuk dapat digunakan pada banyak aplikasi terutama karena ukurannya yang kompak. Mikrokontroler sering digunakan pada sistem yang tidak terlalu kompleks dan tidak memerlukan kemampuan komputasi yang tinggi.

Sistem yang menggunakan mikrokontroler sering disebut sebagai *embeded system* atau *dedicated system*. *Embeded system* adalah sistem pengendali yang tertanam pada suatu produk, sedangkan *dedicated system* adalah sistem pengendali yang dimaksudkan hanya untuk suatu fungsi tertentu. Sebagai contoh printer adalah suatu *embeded system*

karena di dalamnya terdapat mikrokontroler sebagai pengendali dan juga dedicated system karena fungsi pengendali tersebut berfungsi hanya untuk menerima data dan mencetaknya. Hal ini berbeda dengan suatu PC yang dapat digunakan untuk berbagai macam keperluan, sehingga mikroprosesor pada PC sering disebut sebagai general purpose microprocessor (mikroprosesor serba guna). Pada PC berbagai macam software yang disimpan pada media penyimpanan dapat dijalankan, tidak seperti mikrokontroler hanya terdapat satu software aplikasi.

Penggunaan mikrokontroler antara lain terdapat pada bidang-bidang berikut ini:

- otomotif : Engine Control Unit, Air Bag, fuel control, Antilock Braking System, sistem pengaman alarm, transmisi otomatis, hiburan, pengkondisi udara, speedometer dan odometer, navigasi, suspensi aktif.
- perlengkapan rumah tangga dan perkantoran : sistem pengaman alarm, remote control, mesin cuci, microwave, pengkondisi udara, timbangan digital, mesin foto kopi, printer, mouse.
- pengendali peralatan di industri.
- robotika.

Saat ini mikrokontroler 8 bit masih menjadi jenis mikrokontroler yang paling populer dan paling banyak digunakan. Maksud dari mikrokontroler 8 bit adalah data yang dapat diproses dalam satu waktu adalah 8 bit, jika data yang diproses lebih besar dari 8 bit maka akan dibagi menjadi beberapa bagian data yang masing-masing terdiri dari 8 bit. Contoh mikrokontroler 8 bit antara lain keluarga Motorola 68HC05/11, Intel 8051, Microchip PIC 16, dan yang akhir-akhir ini mulai populer keluarga Atmel AVR. Masing-masing mikrokontroler mempunyai cara dan bahasa pemrograman yang berbeda, sehingga program untuk suatu jenis mikrokontroler tidak dapat dijalankan pada jenis mikrokontroler lain.

Pemilihan jenis mikrokontroler yang cocok dengan aplikasi yang dibuat terdapat tiga kriteria yaitu:

- memenuhi kebutuhan secara efektif & efisien. Hal ini menyangkut kecepatan, kemasan/packaging, konsumsi daya, jumlah RAM dan ROM, jumlah I/O dan timer, harga per unit.
- Bahasa pemrograman yang tersedia.
- Kemudahan dalam mendapatkannya

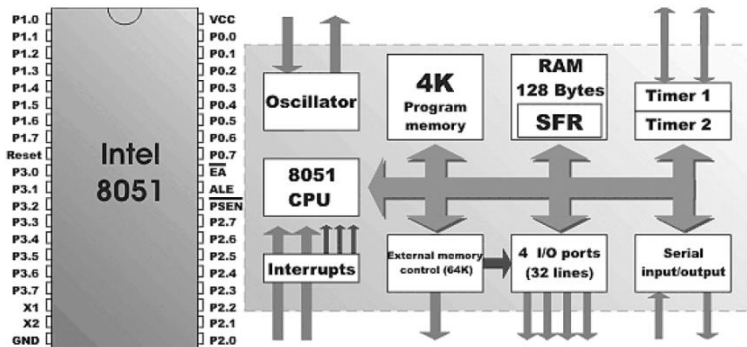
## B. Mikrokontroler AT89C51

Mikrokontroler seri AT89C51 produksi ATMEL adalah pengembangan dari seri 8051 dari Intel. Selain ATMEL terdapat beberapa perusahaan lain yang juga mengembangkan seri 8051 ini seperti Phillips, Siemens, dan Dallas Semiconductor.

### 1. Arsitektur AT89S51

AT89S51 merupakan mikrokontroler 8 bit 40 pin dengan kapasitas RAM 128 byte dan ROM 4 kbyte serta mempunyai dua pewaktu/pencacah (timer/counter) 16 bit, satu port serial, enam sumber interupsi dan empat port input/output (I/O). Masing-masing port I/O terdiri dari 8 pin jalur data sehingga jumlah total pin I/O adalah 32 (Atmel 2001).

Keluarga AT89S mempunyai beberapa seri lain selain seri AT89S51 yaitu jenis 40 pin (AT89S52/53/55) dan jenis 20 pin (AT89S2051/4051).

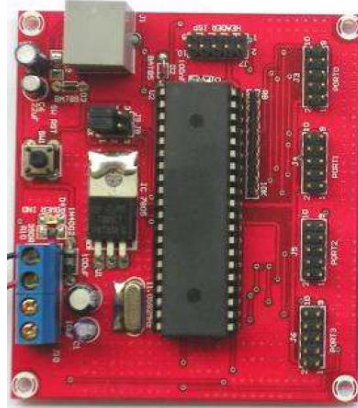


Gambar 1-1 Susunan pin dan arsitektur dari keluarga 8051.

Keluarga AT89S adalah penerus dari keluarga AT89C dengan tambahan kemampuan untuk dapat diprogram secara serial. Hal ini mendukung fasilitas ISP (In System Programming) yang memungkinkan mikrokontroler untuk diprogram secara langsung tanpa memerlukan rangkaian pemrogram khusus. Hal ini berbeda dengan keluarga AT89C yang memerlukan rangkaian pemrogram tersendiri dengan dua level tegangan, yaitu 5 dan 12 Volt.

Beberapa istilah yang merujuk pada rangkaian mikrokontroler adalah rangkaian programmer/downloader dan sistem minimum.

Sistem minimum adalah rangkaian dasar yang diperlukan mikrokontroler untuk dapat bekerja. Rangkaian programmer /downloader adalah rangkaian yang digunakan untuk memprogram mikrokontroler. Untuk seri AT89S51 rangkaian sistem minimumnya dapat sekaligus sebagai rangkaian programmer/downloader.



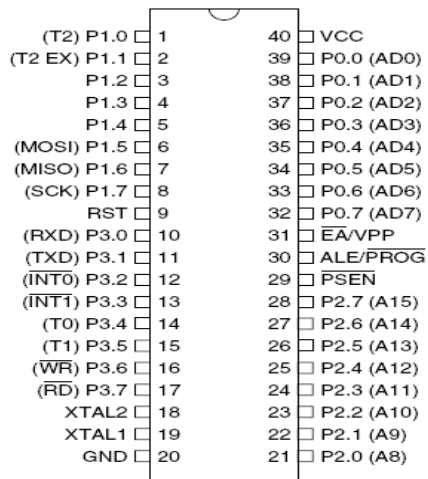
Gambar 1-2 Rangkaian Sistem Minimum untuk AT89S51.

Beberapa komponen pelengkap pada sistem minimum AT89S51 adalah :

- regulator untuk menjaga tegangan catu daya 5 V yang diperlukan mikrokontroler.
- kristal sebagai sumber detak (clock)
- rangkaian reset
- soket untuk koneksi ke peranti I/O
- koneksi ke port paralel untuk pemrograman dari PC

Untuk memprogram suatu mikrokontroler terdapat banyak bahasa pemrograman yang dapat digunakan. Bahasa pemrograman yang biasa digunakan dalam pemrograman mikrokontroler adalah Assembly. File bahasa Assembly (ASM) dapat dituliskan menggunakan sembarang pengolah kata (misal Notepad), untuk kemudian dikompilasi menggunakan aplikasi TopView Simulator untuk mendapatkan file HEX. File HEX inilah yang dimasukkan ke mikrokontroler menggunakan Microcontroller ISP Software dari Atmel (ataupun software ISP lain) melalui perantara kabel paralel.

## 2. Fungsi Pin pada AT89S51



Gambar 1-3 Diagram Pin pada AT89S51

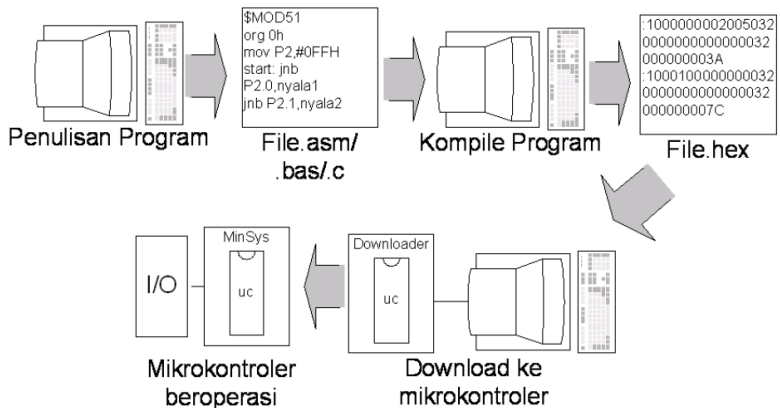
Mikrokontroler AT89S51 buatan ATMEL yang merupakan pengembangan mikrokontroler standard MCS-51 memiliki fitur: PEROM (Programmable Erasable Read Only Memory) 4 kbyte untuk program, RAM internal 128 byte, In-System Programming, 4 buah port I/O masing-masing 8 bit (P0 – P3), 2 buah Timer/counter 16 bit, 5 buah sumber interupsi, Sebuah port serial full duplex, Tiga level penguncian program (menghindari pembajakan program), Memiliki operasi daya rendah. Fungsi Pin pada AT89S51 sebagai berikut:

- VCC = Tegangan positif +5V
- GND = Tegangan negative 0V
- P0 – P3 = Port / pin-pin Input / Output yang akan berhubungan dengan perangkat luar ( tombol, LED, LCD, dll )
- RST = Reset ( untuk mereset sitem )
- XTAL 1,2 = Pin untuk memasang X'tal sebagai sumber denyut untuk Oscilator
- EA/VPP = Pin untuk menentuakn lokasi memori.  
Jika menggunakan memori Internal EA harus diberi VCC  
Jika menggunakan memori Eksternal, EA haurs diberi GND
- ALE dan PSEN = 2 pin ini berfungsi untuk mengakses memori Eksternal. Jika menggunakan internal memori, pin ini tidak digunakan.

- T0-T1 = Pin untuk Timer
- MOSI,MISO,SCK = PIN untuk memprogram mikrokontroler.
- RXD dan TXD = komunikasi serial dengan perangkat / komputer lain.
- INT0 dan INT1 = Pin Interupsi, jika pin ini aktif maka, prosesor akan menjalankan program khusus interupsi.

### 3. Bahasa Pemrograman pada AT89S51

Memprogram suatu mikrokontroler terdapat banyak bahasa pemrograman yang dapat digunakan. Bahasa pemrograman yang biasa digunakan dalam pemrograman mikrokontroler adalah Assembly. File bahasa Assembly (ASM) dapat dituliskan menggunakan sembarang pengolah kata (misal Notepad), untuk kemudian dikompile menggunakan Assembler (misal ASM51) untuk mendapatkan file HEX. File HEX inilah yang dimasukkan ke mikrokontroler menggunakan Microcontroller ISP Software dari Atmel (ataupun software ISP lain) melalui perantaraan kabel paralel.



Gambar 1-4 Langkah-langkah pemrograman mikrokontroler.

Telah dikembangkan beberapa kompiler untuk beberapa bahasa pemrograman tingkat tinggi yang dapat digunakan pada pemrograman mikrokontroler AT89S51 antara lain bahasa assembler dari aplikasi TopView Simulator.

a. Instruksi copy data

Kode dasar untuk kelompok ini adalah MOV, singkatan dari MOVE yang artinya memindahkan, meskipun demikian lebih tepat dikatakan perintah ini mempunyai makna pengcopy-an data. Hal ini bisa dijelaskan berikut : setelah instruksi MOV A,R7 dikerjakan, Akumulator A dan register serba guna R7 berisikan data yang sama, yang asalnya tersimpan di dalam R7.

Perintah MOV dibedakan sesuai dengan jenis memori AT89Cx051. Perintah ini pada memori data dituliskan menjadi MOV, misalkan :

```
MOV A,$20
```

```
MOV A,@R1
```

```
MOV A,P1
```

```
MOV P3,A
```

Untuk pemakaian pada memori program, perintah ini dituliskan menjadi MOVC, hanya ada 2 jenis instruksi yang memakai MOVC, yakni:

```
MOVC A,@A+DPTR ; DPTR sebagai register indirect
```

```
MOVC A,@A+PC ; PC sebagai register indirect
```

Selain itu, masih dikenal pula perintah MOVX, yakni perintah yang dipakai untuk memori data eksternal (X singkatakan dari External). Perintah ini hanya dimiliki oleh anggota keluarga MCS51 yang mempunyai memori data eksternal. Hanya ada 6 macam instruksi yang memakai MOVX, instruksi-instruksi tersebut adalah:

```
MOVX A,@DPTR
```

```
MOVX A,@R0
```

```
MOVX A,@R1
```

```
MOVX @DPTR,A
```

```
MOVX @R0,A
```

```
MOVX @R1,A
```

b. Instruksi Aritmatika

- Perintah ADD dan ADDC

Isi Akumulator A ditambah dengan bilangan 1 byte, hasil penjumlahan akan ditampung kembali dalam Akumulator. Dalam operasi ini bit Carry (C flag dalam PSW – Program Status Word) berfungsi sebagai penampung limpahan hasil penjumlahan. Jika hasil penjumlahan tersebut melimpah (nilainya lebih besar dari 255) bit Carry akan

bernilai '1', kalau tidak bit Carry bernilai '0'. ADDC sama dengan ADD, hanya saja dalam ADDC nilai bit Carry dalam proses sebelumnya ikut dijumlahkan bersama. Bilangan 1 byte yang ditambahkan ke Akumulator, bisa berasal dari bilangan konstan, dari register serba guna, dari memori data yang nomor memorinya disebut secara langsung maupun tidak langsung, seperti terlihat dalam contoh berikut :

ADD A,R0 ; register serba guna

ADD A,#\$23 ; bilangan konstan

ADD A,@R0 ; no memori tak langsung

- Perintah MUL AB

Bilangan biner 8 bit dalam Akumulator A dikalikan dengan bilangan biner 8 bit dalam register B. Hasil perkalian berupa bilangan biner 16 bit, 8 bit bilangan biner yang bobotnya lebih besar ditampung di register B, sedangkan 8 bit lainnya yang bobotnya lebih kecil ditampung di Akumulator A. Bit OV dalam PSW (Program Status Word) dipakai untuk menandai nilai hasil perkalian yang ada dalam register B. Bit OV akan bernilai '0' jika register B bernilai \$00, kalau tidak bit OV bernilai '1'.

MOV A,#10

MOV B,#20

MUL AB

- Perintah DIV AB

Bilangan biner 8 bit dalam Akumulator A dibagi dengan bilangan biner 8 bit dalam register B. Hasil pembagian berupa bilangan biner 8 bit ditampung di Akumulator, sedangkan sisa pembagian berupa bilangan biner 8 bit ditampung di register B. Bit OV dalam PSW (Program Status Word) dipakai untuk menandai nilai sebelum pembagian yang ada dalam register B. Bit OV akan bernilai '1' jika register B asalnya bernilai \$00.

c. Instruksi Lompatan

- Jump

Jump adalah instruksi agar program melompat ke alamat program tertentu. Ada empat jenis

jump diantaranya:

SJMP dengan maximum lompatan sejauh 256 byte

AJMP dengan maximum lompatan sejauh 2 kb



LJMP dengan maximum lompatan sejauh 64 kb

JMP bisa digunakan menggantikan SJMP, AJMP, dan LJMP

- Instruksi pemanggilan sub-rutin

Instruksi sub-rutin yang dimaksudkan adalah proses pemanggilan suatu fungsi dalam program. Beberapa program memiliki suatu fungsi yang sama, oleh karena itu demi mengefisiensi memori program digunakan suatu fungsi yang digunakan bersama. Dalah satu contoh adalah fungsi delay. Instruksi pemanggilan ini ada tiga jenis:

ACALL dengan pemanggilan sejauh maximum 2 Kb

LCALL dengan pemanggilan maximum sejauh 64 Kb

CALL bisa digunakan mewakili ACALL atau LCALL

- xzfdsf

d. Instruksi Lompatan Bersyarat

Lompatan bersyarat hanya bisa digunakan untuk melompat maximum sejauh 256 Byte. Di luar jarak tersebut haruslah dibuat program penghubung. Di antaranya lompatan bersyarat adalah:

- Instruksi JZ / JNZ

Instruksi JZ (Jump if Zero) dan instruksi JNZ (Jump if not Zero) adalah instruksi JUMP

bersyarat yang memantau nilai Akumulator A.

- Instruksi JC / JNC

Instruksi JC (Jump on Carry) dan instruksi JNC (Jump on no Carry) adalah instruksi jump bersyarat yang memantau nilai bit Carry di dalam Program Status Word (PSW). Bit Carry merupakan bit yang banyak sekali dipakai untuk keperluan operasi bit, untuk menghemat pemakaian memori-program disediakan 2 instruksi yang khusus untuk memeriksa keadaan bit Carry, yakni JC dan JNC.

- Instruksi JB / JNB / JBC

Instruksi JB (Jump on Bit Set), instruksi JNB (Jump on not Bit Set) dan instruksi JBC (Jump on Bit Set Then Clear Bit) merupakan instruksi Jump bersyarat yang memantau nilai-nilai bit tertentu. Bit-bit tertentu bisa merupakan bit-bit dalam register status maupun kaki input mikrokontroler MCS51. Contoh pemakaian instruksi JB dan JNB sebagai berikut :

JB P1.1,\$

JNB P1.1,\$

Instruksi-instruksi di atas memantau keadaan kaki IC MCS51 Port 1 bit 1. Instruksi pertama memantau P1.1, jika P1.1 bernilai '1' maka MCS51 akan mengulang instruksi ini, (tanda \$ mempunyai arti jika syarat terpenuhi kerjakan lagi instruksi bersangkutan). Instruksi berikutnya melakukan hal sebaliknya, yakni selama P1.1 bernilai '0' maka MCS51 akan tertahan pada instruksi ini.

- Instruksi proses dan test

Instruksi-instruksi Jump bersyarat yang dibahas di atas, memantau kondisi yang sudah terjadi yang dicatat MCS51. Ada dua instruksi yang melakukan dulu suatu proses baru kemudian memantau hasil proses untuk menentukan apakah harus Jump. Kedua instruksi yang dimaksud adalah instruksi DJNZ dan instruksi CJNE.

- e. Pemrograman pada AT89S51

Pemrograman pada AT89S51 dilakukan dengan membuat file kode sumber (source code) yang ditulis dalam bahasa assembly (file berakhiran \*.asm) dengan langkah-langkah sebagai berikut:

1. Ketik Program dengan bantuan NotePad dan Save as dengan nama yang diakhiri dengan ekstensi .asm (misal: mikro.asm)
2. Compile sumber kode tersebut dengan bantuan aplikasi TopView Simulator.
3. Setelah tidak ada error, file ekstensi .asm (misal: mikro.asm) kemudian diubah ke bentuk file.hex.
4. Kemudian File Hex ini siap dituliskan (download) ke EPROM dalam mikrokontroler menggunakan program uploader (Downloader ISP).

### C. Rangkuman

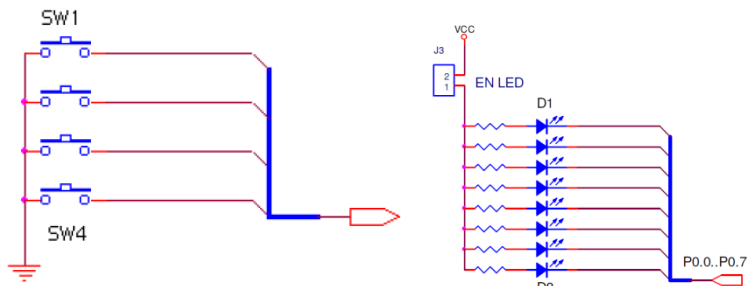
Mikrokontroler pada dasarnya adalah komputer dalam satu chip, yang di dalamnya terdapat mikroprosesor, memori, jalur Input/Output (I/O) dan perangkat pelengkap lainnya. Bahasa pemrograman yang biasa digunakan dalam pemrograman mikrokontroler adalah Assembly. File bahasa Assembly (ASM) dapat dituliskan menggunakan sembarang pengolah kata (misal Notepad), untuk kemudian dikompilasi menggunakan aplikasi TopView Simulator. Rangkaian programmer /downloader adalah rangkaian yang digunakan untuk memprogram mikrokontroler.

## D. Latihan

### 1. Saklar Push Button

Tujuan:

- memahami rangkaian mikrokontroller dengan interface ke saklar
- memahami program assembly untuk mengambil data saklar dan mengeluarkan data ke LED.
- memahami beberapa instruksi assembly dasar, MOV, Setb, Clr, RL dan RR.



Gambar 1.5. Rangkaian Interface Push Button dan LED

Pada gambar 1.5. tersebut tampak rangkaian push button, bila saklar ditekan maka port sesuai dengan bit tersebut akan mendapat logika low '0' dan sebaliknya bila saklar tidak ditekan maka port tersebut akan mendapat logika high '1'.

Percobaan Ambil Data Saklar

- Pada percobaan ini, LED akan nyala bila saklar ditekan sesuai dengan bit tersebut.
- Ketik program berikut ini (beri nama ledbutton.asm):

```
ORG 0H
```

```
MAIN :
```

```
MOV P3,#0FFH ; memberikan logika high ke port3
```

```
MOV A,P3 ; apakah ada penekanan tombol di port 3
```

```
CALL DELAY ; waktu delay mengurangi efek bouncing
```

```
MOV P0,A ; mengeluarkan kondisi pembacaan ke port 1
```

```
SJMP MAIN ; lompat ke awal program
```

```
DELAY : MOV R4,#0 ; isi r4 dengan 0h
```

```
DELAY1 : MOV R5,#1H ; isi r5 dengan 1h
```

```
DJNZ R5,$ ; tunggu r5 sampai 0
```

```
DJNZ R4,DELAY1 ; kurangi r4 bila tidak 0 lompat kedelay1
```

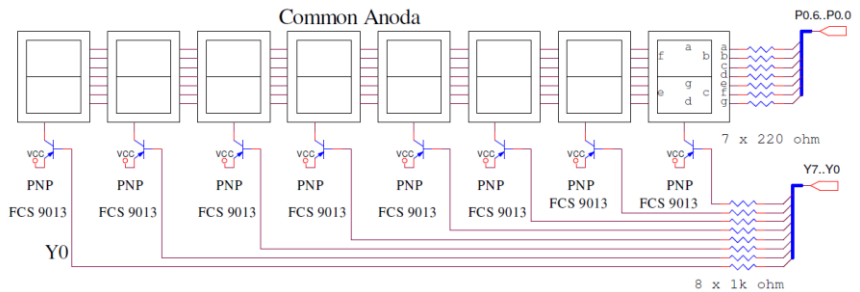
```
RET
```

```
END
```

## 2. Display 7 Segmen

Tujuan:

- memahami rangkaian interface mikrokontroller dengan 7 segmen
- memahami program assembly untuk menampilkan data ke 7 segmen
- memahami beberapa instruksi assembly dasar, MOV, Setb, Clr, RLC, RRC dan waktu tunda.



Gambar 1.6. Rangkaian 7 Segmen

Pada gambar tersebut seven segment common anoda dikendalikan dengan menggunakan transistor PNP, apabila ada logika low pada basis transistor, maka 7 segment akan nyala dan sebaliknya akan padam.



Gambar 1.7 Modul 7 Segment tunggal

Tabel 1.1. Data Display 7 Segmen

P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	Display
DOT	g	f	e	d	c	b	a	
1	1	0	0	0	0	0	0	0
1	1	1	1	1	0	0	1	1
1	0	1	0	0	1	0	0	2
1	0	1	1	0	0	0	0	3
1	:	:	:	:	:	:	:	:
1	0	0	0	1	0	0	0	A
1	0	0	0	0	0	1	1	b

Pada tabel tersebut tampak bahwa untuk menghidupkan sebuah segmen, harus dikirimkan data logika low "0" dan sebaliknya untuk mematikan segmen, harus dikirimkan data logika high "1".

#### Percobaan Menulis Sebuah Karakter pada 7 Segmen

Pada percobaan ini, karakter 'A' akan ditampilkan pada 7 Segmen Display 1. Untuk melakukan percobaan ini ketikkan program ini (beri nama 7segmen.asm):

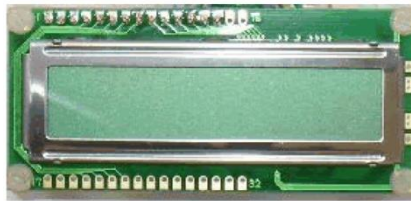
```
SEVENS EQU P1
ORG 0H
MAIN :
    CLR P0.0
    MOV SEVENS,#0A0H ; karakter 'A'
    SJMP MAIN
END
```

### 3. LCD ( Liquid Crystal Display)

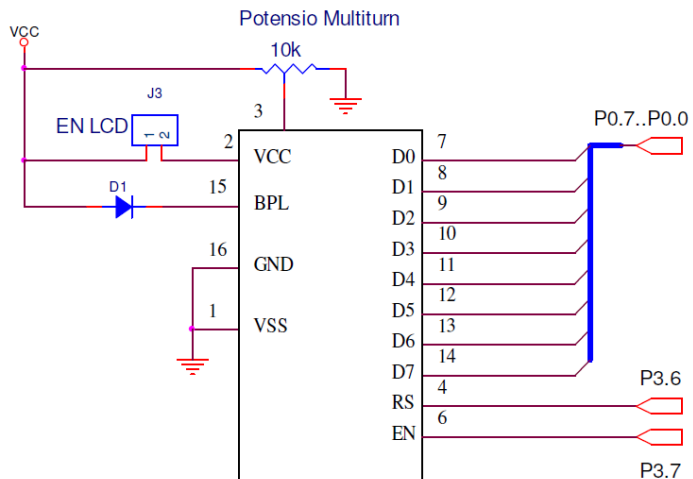
Tujuan:

- memahami rangkaian interface mikrokontroller dengan LCD Karakter 2x16.
- memahami program assembly untuk menampilkan data ke LCD Karakter 2x16.
- Peserta memahami beberapa instruksi assembly dasar, MOV, Setb, Clr, dan waktu tunda.
- memahami mencetak karakter pada posisi baris dan kolom tertentu

Modul LCD Character dapat dengan mudah dihubungkan dengan mikrokontroller seperti AT89S51. LCD mempunyai lebar display 2 baris 16 kolom atau biasa disebut sebagai LCD Character 2x16, dengan 16 pin konektor, yang didefinisikan sebagai berikut:



Gambar 1.8. Modul LCD Karakter 2x16



Gambar 1.9. Rangkaian LCD Karakter 2x16

Tabel 1.2 Pin dan Fungsi

PIN	Name	Function
1	V <sub>SS</sub>	Ground voltage
2	V <sub>CC</sub>	+5V
3	V <sub>EE</sub>	Contrast voltage
4	RS	Register Select 0 = Instruction Register 1 = Data Register
5	R/W	Read/ Write, to choose write or read mode 0 = write mode 1 = read mode
6	E	Enable 0 = start to lacht data to LCD character 1= disable
7	DB0	LSB
8	DB1	-
9	DB2	-
10	DB3	-
11	DB4	-
12	DB5	-
13	DB6	-
14	DB7	MSB
15	BPL	Back Plane Light
16	GND	Ground voltage

Display karakter pada LCD diatur oleh pin EN, RS dan RW:

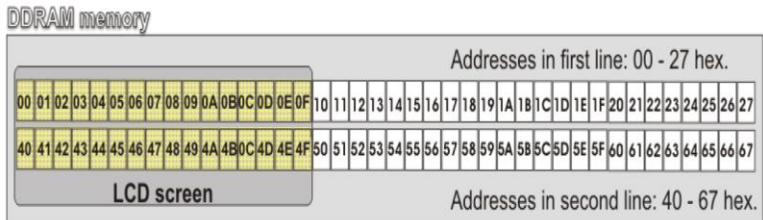
Jalur EN atau Enable yang digunakan untuk memberitahu LCD bahwa sedang mengirimkan data. EN harus dibuat logika low “0” dan high “1” pada dua jalur kontrol yang lain RS dan RW. Ketika dua jalur yang lain telah siap, set EN dengan logika “1” dan tunggu untuk sejumlah waktu tertentu dan set EN ke logika low “0” lagi.

Jalur RS atau Register Select saat berlogika low “0”, data akan dianggap sebagai sebuah perintah atau instruksi khusus (seperti clear screen, posisi kursor dll). Ketika RS berlogika high “1”, data yang dikirim adalah data text yang akan ditampilkan pada display LCD. Jalur RW adalah jalur kontrol Read/ Write. Ketika RW berlogika low (0), maka informasi pada bus data akan dituliskan pada layar LCD. Ketika RW berlogika high ”1”, maka program akan melakukan pembacaan memori dari LCD. Sedangkan pada aplikasi umum pin RW selalu diberi logika low ”0”. Pada akhirnya, bus data terdiri dari 4 atau 8 jalur ( bergantung pada mode operasi yang dipilih oleh user ). Pada kasus bus data 8 bit, jalur diacukan sebagai DB0 s/d DB7.

## Memori LCD

### a. DDRAM ( Display Data RAM )

Memori DDRAM digunakan untuk menyimpan karakter yang akan ditampilkan. Semua teks yang kita tuliskan ke modul LCD adalah disimpan didalam memory ini, dan modul LCD secara berturutan membaca memory ini untuk menampilkan teks ke modul LCD itu sendiri.



Gambar 1.10. Lokasi memori display LCD Karakter

Pada peta memori tersebut, daerah yang berwarna kuning (00 s/d 0F dan 40 s/d 4F) adalah display yang tampak. Sebagaimana yang anda lihat, jumlahnya sebanyak 16 karakter per baris dengan dua baris. Angka pada setiap kotak adalah alamat memori yang bersesuaian dengan posisi dari layar. Demikianlah karakter pertama di sudut kiri atas adalah menempati alamat 00h. Posisi karakter berikutnya adalah alamat 01h dan seterusnya. Akan tetapi, karakter pertama dari baris 2 sebagaimana yang ditunjukkan pada peta memori adalah pada alamat 40h. Demikianlah kita perlu untuk mengirim sebuah perintah ke LCD untuk mengatur letak posisi kursor pada baris dan kolom tertentu. Instruksi Set Posisi Kursor adalah 80h. Untuk ini kita perlu menambahkan alamat lokasi dimana kita berharap untuk menempatkan kursor. Sebagai contoh, kita ingin menampilkan kata "World" pada baris ke dua pada posisi kolom ke sepuluh. Sesuai peta memori, posisi karakter pada kolom 11 dari baris ke dua, mempunyai alamat 4Ah, sehingga sebelum kita tulis kata "World" pada LCD, kita harus mengirim instruksi set posisi kursor, dan perintah untuk instruksi ini adalah 80h ditambah dengan alamat  $80h + 4Ah = 0Cah$ . Sehingga dengan mengirim perintah Cah ke LCD, akan menempatkan kursor pada baris kedua dan kolom ke 11 dari DDRAM.



b. CGROM ( Character Generator ROM )

Sebuah peta karakter yang dapat ditampilkan, sesuai dengan lokasi dari masing-masing karakter.

Tabel 1.3 Peta Karakter

		4 higher bits in address																											
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111												
4 lower bits in address	xxxx0000	CG RAM (1)			0	1	2	3	4	5	6	7	8	9	A	B	-	0	1	2	3	4	5	6	7	8	9	A	B
	xxxx0001	(2)			!	1	A	Q	a	q								。	ア	チ	△	ä	q						
	xxxx0010	(3)			"	2	B	R	b	r								「	イ	ツ	×	β	θ						
	xxxx0011	(4)			#	3	C	S	c	s								」	ウ	テ	ε	ε	∞						
	xxxx0100	(5)			\$	4	D	T	d	t								、	エ	ト	ト	μ	Ω						
	xxxx0101	(6)			%	5	E	U	e	u								・	オ	ナ	1	ε	Ü						
	xxxx0110	(7)			&	6	F	V	f	v								ヲ	カ	ニ	ヨ	ρ	Σ						
	xxxx0111	(8)			'	7	G	W	g	w								フ	キ	ヌ	ラ	g	π						
	xxxx1000	(1)			<	8	H	X	h	x								イ	ク	ネ	リ	∫	×						
	xxxx1001	(2)			)	9	I	Y	i	y								ウ	ケ	ル	ル	'	γ						
	xxxx1010	(3)			*	:	J	Z	j	z								エ	コ	ン	レ	j	〒						
	xxxx1011	(4)			+	;	K	[	k	{								オ	サ	ヒ	ロ	*	斤						
	xxxx1100	(5)			,	<	L	¥	l									カ	シ	フ	ワ	φ	円						
	xxxx1101	(6)			-	=	M	]	m	}								ユ	ズ	ヘ	シ	も	÷						
	xxxx1110	(7)			.	>	N	^	n	→								ヨ	セ	ホ	°	ñ							
	xxxx1111	(8)			/	?	O	_	o	€								ッ	ソ	マ	°	ö	■						

Percobaan Menulis Karakter 'A' pada LCD, ketikkan program ini (beri nama LCD.asm):

```

LCD_RS bit P1.0
LCD_CS bit P1.1
org 0h
call init_LCD
MAIN: mov R1,#80h ; Lokasi Display RAM, Row=1 Col=1
call wr_inst
mov A,#'A' ; Cetak Karakter A
call write_data
stop : sjmp stop
;=====
; Inisialisasi LCD
;=====
INIT_LCD : mov A,#03Fh ;FUNCTION SET 0011 1111
call wr_inst
call wr_inst
mov A,#0Ch ;0000 1101
call wr_inst
mov A,#06h ;0000 0110
call wr_inst
mov A,#01h ;0000 0001
call wr_inst
RET
;=====
; Routine untuk menulis instruksi ke LCD
;=====
wr_inst:
clr LCD_RS ;INSTRUKSI
mov P2,A ;intruksi ke LCD
SETB LCD_CS ;module
CALL DELAY
CLR LCD_CS
call delay
ret
;=====
; Routine untuk menulis data ke LCD
;=====

```

```

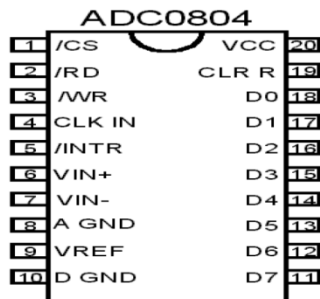
wr_data:
setb LCD_RS
;setb LCD_CS
mov P2,A ;data ke LCD
SETB LCD_CS ;module
CALL DELAY
CLR LCD_CS
call delay
ret ;
;=====
; ROUTINE DELAY
;=====
delay: mov R0,#0
delay1:mov R7,#0fh
djnz R7,$
djnz R0,delay1
ret
end

```

#### 4. Analog To Digital Converter (ADC)

Tujuan:

- memahami rangkaian interface mikrokontroller dengan ADC 0804
- memahami setting tegangan referensi Vref ADC0804
- memahami perhitungan tegangan resolusi ADC0804
- memahami program assembly untuk menampilkan data ADC ke LCD Karakter 2 x 16



Gambar 1.11 Konfigurasi pin ADC0804

Diagram konfigurasi pin ADC0804 ditunjukkan pada gambar 5.2. Pin 11 sampai 18 (keluaran digital) adalah keluaran tiga keadaan, yang dapat dihubungkan langsung dengan bus data bilamana diperlukan. Apabila CS (pin 1) atau RD (pin2) dalam keadaan high ("1"), pin 11 sampai 18 akan mengambang (high impedance), apabila CS dan RD rendah keduanya, keluaran digital akan muncul pada saluran keluaran.

Sinyal mulai konversi pada WR (pin 3). Untuk memulai suatu konversi, CS harus rendah. Bilamana WR menjadi rendah, konverter akan mengalami reset, dan ketika WR kembali kepada keadaan high, konversi segera dimulai.

Konversi detak konverter harus terletak dalam daerah frekuensi 100 sampai 800kHz. CLK IN (pin 4) dapat diturunkan dari detak mikrokontroller, sebagai kemungkinan lain, kita dapat mempergunakan pembangkit clock internal dengan memasang rangkaian RC antara CLN IN (pin 4) dan CLK R (pin 19). Pin 5 adalah saluran yang digunakan untuk INTR, sinyal selesai konversi. INTR akan menjadi tinggi pada saat memulai konversi, dan akan aktif rendah bila konversi telah selesai. Tepi turun sinyal INTR dapat dipergunakan untuk menginterupsi sistem mikrokontroller, supaya mikrokontroller melakukan pencabangan ke subroutine pelayanan yang memproses keluaran konverter.

Pin 6 dan 7 adalah masukan diferensial bagi sinyal analog. A/D ini mempunyai dua ground, A GND (pin 8) dan D GND (pin10). Kedua pin ini harus dihubungkan dengan ground. Pin 20 harus dihubungkan dengan catu daya +5V A/D ini mempunyai dua buah ground, A GND (pin 8) dan D GND (pin 10). Keduanya harus dihubungkan dengan catu daya, sebesar +5V.

Pada A/D 0804 REF V merupakan tegangan referensi yang digunakan untuk offset suatu keluaran digital maksimum. Dengan persamaan sebagai berikut:

$$-V_{REF} = \frac{1}{2} V_{INmaks}$$

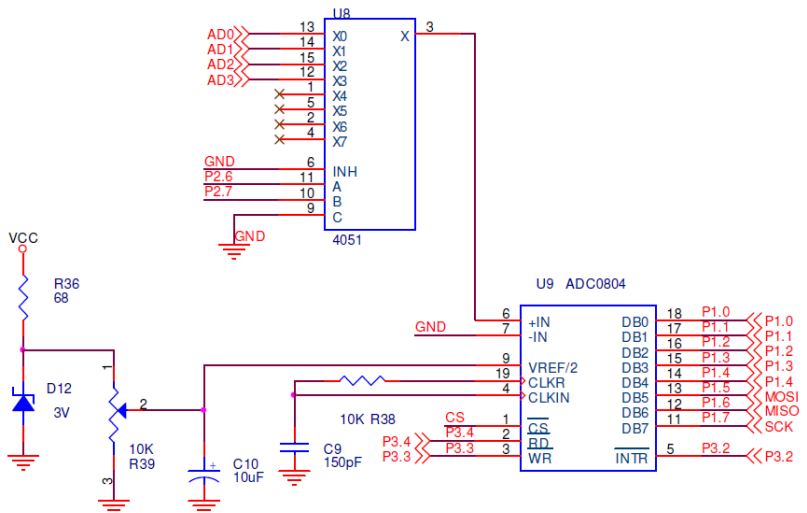
$$V_{RESOLUSI} = \frac{V_{INmaks}}{255}$$

Resolusi ini mempunyai arti sebagai berikut:

Tabel 1.4 Data Resolusi ADC

No	$V_{IN}$ ( volt )	Data Digital ( biner )
1	0,000	0000 0000
2	0,0156	0000 0001
3	0,0313	0000 0010
4	:	
5	4	1111 1111

A/D ini dapat dirangkai untuk menghasilkan konversi secara kontinu. Untuk melaksanakannya, kita harus menghubungkan CS, dan RD ke ground dan menyambungkan WR dengan INTR seperti pada gambar dibawah ini. Maka dengan ini keluaran digital yang kontinu akan muncul, karena sinyal INTR menggerakkan masukan WR. Pada akhir konversi INTR berubah menjadi low, sehingga keadaan ini akan mereset konverter dan mulai konversi.



Gambar 1.12 Rangkaian ADC0804

Konverter A/D tersedia secara komersial sebagai rangkaian terpadu dengan resolusi 8 bit sampai dengan 16 bit. Pada percobaan ini akan memperkenalkan ADC0801, yaitu sebagai sebuah konverter A/D 8 bit yang mudah diinterfacedengan sistem mikrokontroler. A/D ini menggunakan metode aproksimasi berturut-turut untuk mengkonversikan masukan analog (0-5V) menjadi data digital 8 bit yang ekuivalen.

ADC0801 mempunyai pembangkit clock internal dan memerlukan catu daya +5V dan mempunyai waktu konversi optimum sekitar 100us. Percobaan Data ADC0804 dalam desimal akan ditampilkan pada LCD Karakter 2x16 pada Baris 1, Colom 1, 2 dan 3, menampilkan data ratusan, puluhan dan satuan. Ketikkan program ini (beri nama ADC.asm):

```

org 0h
ratusan equ 30h
puluhan equ 31h
satuan equ 32h
;
org 0h
call init_LCD
call write_char
start: call ADC
call Bin2Dec
call Write2LCD
sjmp start
;
;=====
;Subrutin ini digunakan untuk mengambil data ADC MUX X0
;=====
ADC: clr P2.6
clr P2.7
clr P3.3
nop
nop
nop
setb P3.3
eoc: jb P3.2,eoc
clr P3.4
mov A,P1
setb P3.4
ret
;=====
;Subrutin untuk menampilkan data ke LCD character 2 x16
;pada DDRAM 0C9 0CA 0CB untukratusan, puluhan, and satuan
;=====

```

```

Write2LCD:
mov r1,#0c9h
call write_inst
mov a,ratusan
add a,#30h
mov r1,a
call write_data
;
mov r1,#0cah
call write_inst
mov a,puluhan
add a,#30h
mov r1,a
call write_data
;
mov r1,#0cbh
call write_inst
mov a,satuan
add a,#30h
mov r1,a
call write_data
ret

```

```

;=====
;Subrutin ini untuk merubah data biner ke desimal
;menjadi 3 digit = ratusan-puluhan-satuan
;=====

```

```

Bin2Dec:
mov b,#100d
div ab
mov ratusan,a
mov a,b
mov b,#10d
div ab
mov puluhan,a
mov satuan,b

```

```

;=====
;Subrutin untuk menampilkan tulisan Data ADC0804
;pada baris 1
;=====

```

```

write_char:
mov dptr,#word1 ;DPTR = [ address word1 ]
mov r3,#16 ;R3=16,number character to be display
mov r1,#80h ;R1=80h,address DDRAM start position
acall write_inst
;
write1:clr a ; A = 0
movc a,@a+dptr ; A = [A+ DPTR]
mov r1,A ; R1 = A
inc dptr ; DPTR = DPTR +1
acall write_data ;
djnz r3,write1 ; R3 = R3-1,
ret
;
Init_lcd:
mov r1,#00000001b ;Display clear
call write_inst
mov r1,#00111000b ;Function set,Data 8 bit,2 line font 5x7
call write_inst
mov r1,#00001100b ;Display on, cursor off,cursor blink off
call write_inst
mov r1,#00000110b ;Entry mode, Set increment
call write_inst
ret
;
write_inst:
clr P3.6 ; RS = P2.0 = 0, write mode instruction
mov P0,R1 ; D7 s/d D0 = P0 = R1
setb P3.7 ; EN = 1 = P2.1
call delay ; call delay time
clr P3.7 ; EN = 0 = P2.1
ret
Write_data:
setb P3.6 ; RS = P2.0 = 1, write mode data
mov P0,R1 ; D7 s/d D0 = P0 = R1
setb P3.7 ; EN = 1 = P2.1
call delay ; call delay time
clr p3.7 ; EN = 0 = P2.1
ret

```



```

;
delay: mov R0,#0
delay1:mov R2,#0fh
dijnz R2,$
dijnz R0,delay1
ret
;
word1: DB ' Data ADC0804 '
end

```

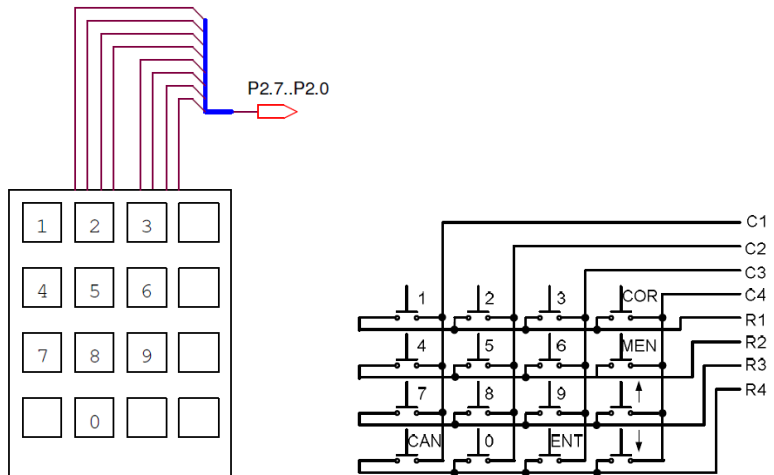
## 5. Keypad 4x4

Tujuan:

- a. memahami rangkaian interface keypad 4x4 dengan mikrokontroller
- b. memahami bahasa assembly untuk pengambilan data keypad
- c. memahami bahasa assembly untuk pengambilan data keypad dan mengeluarkan ke LED
- d. memahami bahasa assembly untuk pengambilan data keypad dan mengeluarkan ke 7 Segmen
- e. memahami bahasa assembly untuk pengambilan data keypad dan mengeluarkan ke LCD Karakter

Keypad serig digunakan sebagai suatu input pada beberapa peralatan yang berbasis mikroprocessor atau mikrokontroller. Keypad sesungguhnya terdiri dari sejumlah saklar, yang terhubung sebagai baris dan kolom dengan susunan seperti yang ditunjukkan pada gambar 1.13.

Agar mikrokontroller dapat melakukan scan keypad, maka port mengeluarkan salah satu bit dari 4 bit yang terhubung pada kolom dengan logika low “0” dan selanjutnya membaca 4 bit pada baris untuk menguji jika ada tombol yang ditekan pada kolom tersebut. Sebagai konsekuensi, selama tidak ada tombol yang ditekan, maka mikrokontroller akan melihat sebagai logika high “1” pada setiap pin yang terhubung ke baris.



Gambar 1.13 Rangkaian interface keypad 4x4

Percobaan scan data keypad 4x4 dan mengeluarkan ke LCD. Ketikkan program ini (beri nama KeyPad.asm):

```
col4 bit P2.0
col3 bit P2.1
col2 bit P2.2
col1 bit P2.3
row1 bit P2.4
row2 bit P2.5
row3 bit P2.6
row4 bit P2.7
keydata equ 70h
keybounc equ 71h
keyport equ P2
org 0h
mov P2,#11111111b
call Init_LCD
start: call keypad4x4 ;calling subroutine keypad4x4
Mov A,keydata ;A = keydata
Cjne A,#0FFh,WrLCD;
sjmp start ;LOOPING FOREVER PART 1
;
```

```

WrLCD: Mov R1,#80h ;Pick DDRAM 1st row and 1st col
call write_inst
Mov R1,#30h
Add A,R1
Mov R1,A
call write_data ;write data
Sjmp start ;LOOPING FOREVER PART 2;
; Init_lcd:
mov r1,#00000001b ;Display clear
call write_inst ;
mov r1,#00111000b ;Function set, Data 8 bit,2 line font 5x7
call write_inst ;
mov r1,#00001100b ;Display on, cursor off,cursor blink off
call write_inst
mov r1,#00000110b ;Entry mode, Set increment
call write_inst
ret
;
Write_inst:
clr P3.6 ; P3.6 = RS =0
mov P0,R1 ; P0 = D7 s/d D0 = R1
setb P3.7 ; P3.7 =EN = 1
call delay ; call delay time
clr P3.7 ; P3.7 =EN = 0
ret
;
Write_data:
setb P3.6 ; P3.6 = RS =1
mov P0,R1 ; P0 = D7 s/d D0 = R1
setb P3.7 ; P3.7 =EN = 1
call delay ; call delay time
call delay ; call delay time
clr p3.7 ; P3.7 =EN = 0
ret

```

```

;=====
; subroutine scan keypad 4x4
;=====
Keypad4x4:
mov keybounc,#50 ;keybounc = 50
mov keyport,#0FFh ;keyport=P2= FF
clr col4 ;col4 = 0
;
keyCOR: jb row4,keyMEN ; Key COR
djnz keybounc,KeyCOR
mov keydata,#0Ah ;Data Output
ret
;
keyMEN: jb row3,keyUpA ; Key MEN
djnz keybounc,keyMEN
mov keydata,#0bh ;Data Output
ret
keyUpA: jb row2,keyDnA ; Key Up Arrow
djnz keybounc,keyUpA
mov keydata,#0ch ;Data Output
ret
;
keyDnA: jb row1,key3 ; Key Down Arrow
djnz keybounc,keyDnA
mov keydata,#0dh ;Data Output
ret
;=====
key3: setb col4
clr col3
jb row4,key6
djnz keybounc,key3 ; Key 3
mov keydata,#03h ;Data Output
ret
;
key6: jb row3,key9
djnz keybounc,key6 ; Key 6
mov keydata,#06h ;Data Output
ret
;

```

```

key9: jb row2,keyENT
djnz keybounc,key9 ; Key 9
mov keydata,#09h ;Data Output
ret
;
keyENT: jb row1,key2
djnz keybounc,keyENT ; Key ENT
mov keydata,#0eh ;Data Output
ret
;=====
key2: setb col3
clr col2
jb row4,key5
djnz keybounc,key2
mov keydata,#02h ;Data Output
ret
;
key5: jb row3,key8
djnz keybounc,key5
mov keydata,#05h ; Data Output
ret
;
key8: jb row2,key0
djnz keybounc,key8
mov keydata,#08h ;Data Output
ret
;
key0: jb row1,key1
djnz keybounc,key0
mov keydata,#00h ;Data Output
ret
;=====
key1: setb col2
clr col1
jb row4,key4
djnz keybounc,key1
mov keydata,#01h ;Data Output
ret

```

```
key4: jb row3,key7
djnz keybounc,key4
mov keydata,#04h ;Data Output
ret
key7: jb row2,keyCAN
djnz keybounc,key7
mov keydata,#07h ;Data Output
ret
keyCAN: jb row1,Nokey
djnz keybounc,keyCAN
mov keydata,#0Fh ;Data Output
ret
Nokey: mov keydata,#0FFh
ret
;=====
;The end of Keypad 4x4 subroutine
;=====
delay: mov R0,#0
delay1: mov R2,#50
djnz R2,$
djnz R0,delay1
ret
end
```

## 6. Motor Stepper

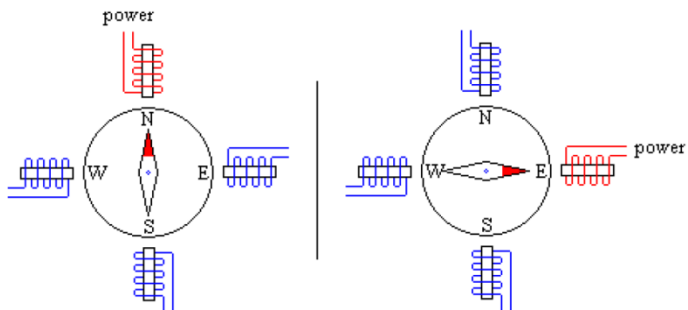
### Tujuan

- mamahami rangkaian interface mikrokontroller dengan motor stepper
- memahami rangkaian driver motor stepper ULN2003
- memahami bahasa assembly untuk mengatur arah putaran motor stepper
- memahami bahasa assembly untuk mengatur arah putaran motor stepper dengan menggunakan saklar.



Gambar 1.14 Motor Stepper

Motor stepper bila diberikan tegangan pada motor ini, maka motor akan berada dalam keadaan diam, agar motor dapat berputar, maka harus merubah tegangan yang masuk ke motor. Sebagai ilustrasi sebuah kompas dengan elektromagnet disekitarnya. Sebagaimana digambarkan pada gambar 1.15, apabila tegangan yang diberikan pada elektromagnet dirubah, maka akan merubah posisi jarum dari kompas.



Gambar 1.15. Ilustrasi sebuah kompas dengan elektromagnet

Dengan empat buah elektromagnet maka gerakan akan melompat secara kasar. Sekarang bayangkan susunan yang sama dengan 100 elektromagnet yang mengitari kompas. Dengan mengatur energi yang mengalir pada setiap elektromagnet dalam berurutan, maka jarum akan memerlukan sebanyak 100 langkah untuk melakukan satu kali putaran.

Percobaan Menggerakkan Motor Stepper Searah Jarum Jam. Ketikkan program ini (beri nama Stepper.asm):

```
org 0h
start: call StepCW
sjmp start
;
StepCW:
mov P0,#11101111b ; Turn on driver 1. isi P0 dg 11101111b
call delay ; call delay time
mov P0,#11011111b ; Turn on driver 2. isi P0 dg 11011111b
call delay ; call delay time
mov P0,#10111111b ; Turn on driver 3. isi P0 dg 10111111b
call delay ; call delay time
mov P0,#01111111b ; Turn on driver 4. isi P0 dg 01111111b
call delay ; call delay time
ret
StepCCW:
mov P0,#01111111b ; Turn on driver 1
call delay ; call delay time
mov P0,#10111111b ; Turn on driver 2
call delay ; call delay time
mov P0,#11011111b ; Turn on driver 3
call delay ; call delay time
mov P0,#11101111b ; Turn on driver 4
call delay ; call delay time
ret
delay: mov R0,#255
delay1:mov R2,#255
dijnz R2,$
dijnz R0,delay1
ret
end
```



E. Rujukan

Malvino, A.P, 1996, Prinsip – Prinsip Elektronika Edisi ke 3, Erlangga, Jakarta.

Tim Gramedia, 1991, Rangkaian – Rangkaian Elektronika, PT. Elex Media Komputindo, Jakarta.

Nalwan, P. A., 2003, Panduan Praktis Teknik Antarmuka dan Pemrograman Mikrokontroler AT89C51, Elex Media Komputindo, Jakarta

F. Bacaan yang dianjurkan

Eko Putra, Agfianto, 2002, Belajar Mikrokontroller AT89C51/52/55 ( Teori dan Aplkasi ), Gava Media, Yogyakarta.

Malik, I. A., 1997, Bereksperimen Dengan Mikrokontroler 8031, PT Elex Media Komputindo: Jakarta.

## BAB II

### Mikrokontroler Simulator

#### A. Pendahuluan

Mikrokontroler Simulator diperlukan bila seorang programmer belum mempersiapkan perangkat untuk mem-flash mikrokontroler untuk menguji fungsionalitas program dalam fase eksperimental. Untuk ini, ada software simulator yang digunakan untuk mensimulasikan mikrokontroler bekerja tanpa mikrokontroler itu sendiri. Simulator biasanya tidak memiliki koneksi ke dunia nyata, semua operasi disimulasikan dalam perangkat lunak. Simulator mikrokontroler adalah model program yang meniru pekerjaan mikrokontroler. Simulator modern sekarang mensimulasikan operasi aritmatika dan operasi I / O dan bahkan periferal seperti timer, ADC, USART, I2C, dan sebagainya. Dalam banyak kasus, dimungkinkan untuk mempersiapkan seluruh proyek menggunakan simulator dan kemudian merekam sumber kode yang dikompilasi ke mikrokontroler yang sebenarnya. Mikrokontroler Simulator memfasilitasi: debugging pada level kode sumber; mengikuti waktu operasi dalam gerakan lambat tetapi dengan nilai dunia nyata; dan menghubungkan sinyal stimulus seperti sinyal dunia nyata.

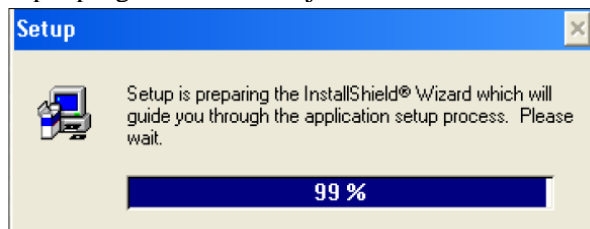
#### B. TopView Simulator

Program Topview simulator merupakan software yang dirancang khusus untuk belajar mikrokontroler khususnya keluarga MCS-51. Sedangkan jenis IC yang banyak digunakan pada software ini lebih banyak produksi dari ATMEL (AT89C51 sampai AT89S252).

Walaupun software ini lebih banyak untuk aplikasi mikrokontroler produksi ATMEL kita tidak perlu khawatir karena jenis ini ada banyak sekali dipasaran dan harga relative lebih murah.

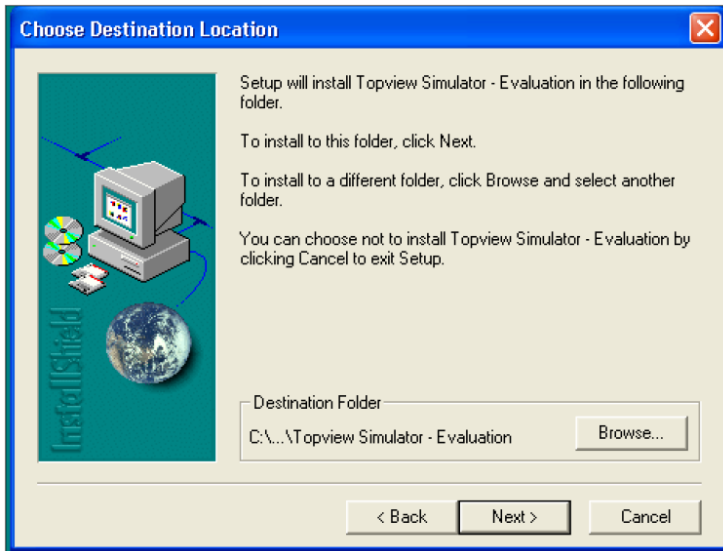
##### 1. Tahapan menginstal

Buka software TopviewSimulator, double klik ikon SETUP dan biarkan sampai progress bar menunjukkan nilai 100%.

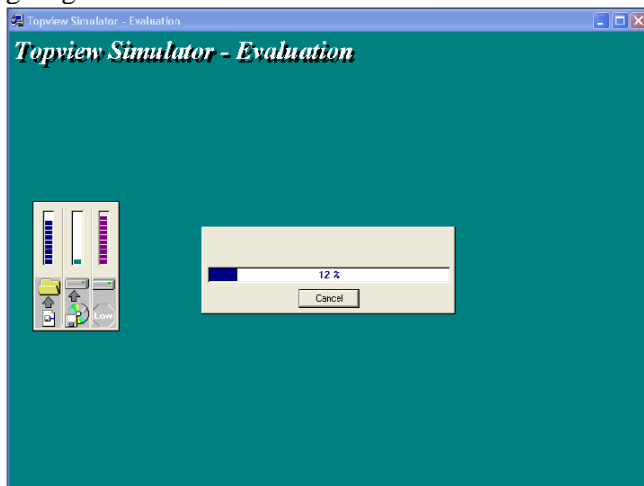


Gambar 2.1. Instalasi software TopviewSimulator

Klik Next untuk proses selanjutnya,



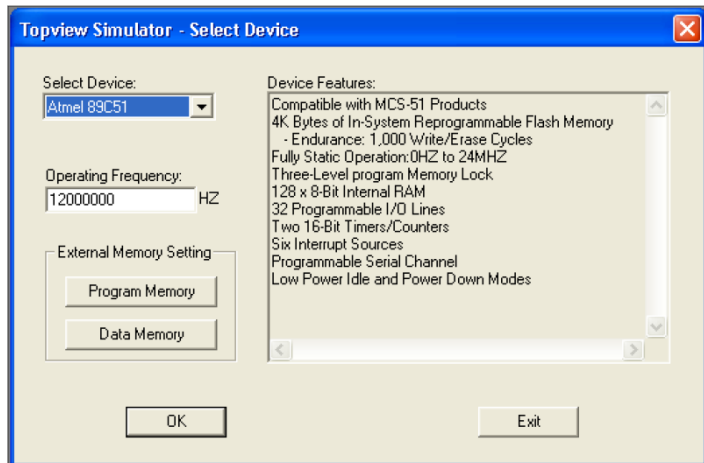
Gambar 2.2. Didirektori akan diletakkan file program  
Menentukan didirektori akan diletakkan file program  
Topview Simulator. Karena sudah dipilihkan suatu direktori maka  
bisa langsung klik Next.



Gambar 2.3. Instalasi software pada progress bar  
Proses diatas biarkan sampai progress bar menunjukkan angka 100%.

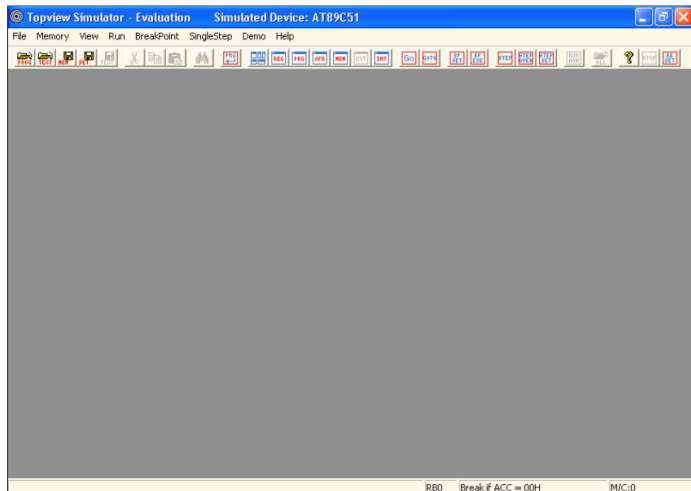
## 2. Buka Program

Untuk melihat hasil meinstal/ menjalankan langsung program Topview Simulator yaitu dengan cara : Pada windows → menu Start (klik) → Program → Topview Simulator (klik). Maka akan keluar tampilan (Select Device) dan pilih OK.



Gambar 2.4. Select Device

Setelah tampilan Select Device dan diklik OK, maka akan keluar tampilan berikut:

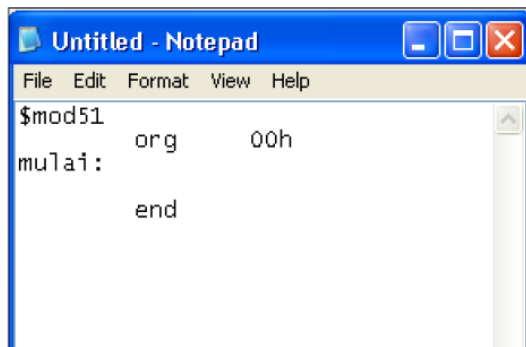


Gambar 2.5. Program Topview Simulator

### 3. Penulisan File Awal

Penulisan file awal disini buka menulis program yang akan digunakan sebagai program dari mikrokontroler tetapi hanya sebagai awalan penulisan program yang selanjutnya diedit dan disimpan kembali dengan nama yang lain. Jadi program atau file dengan nama lain inilah yang akan digunakan sebagai program dari mikrokontroler. Cara penulisan file mulai dari awal :

- a. Pertama kali buka program NOTEPAD, kemudian tulis program dibawah ini.



Gambar 2.6. Program Topview Simulator

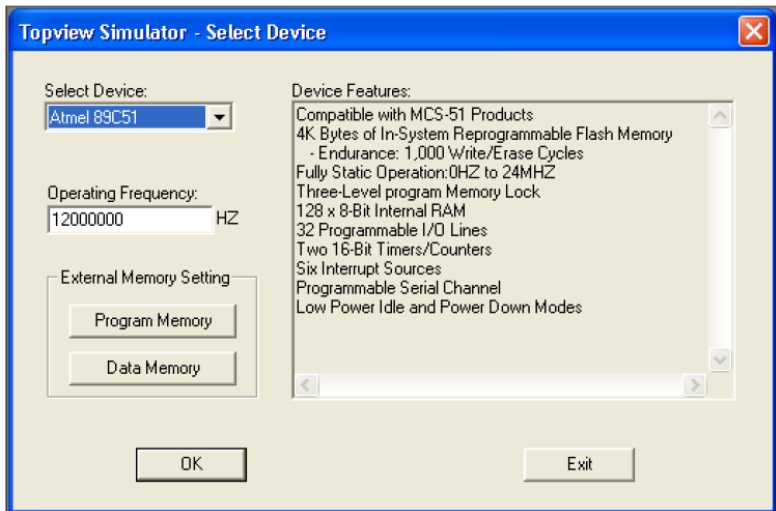
- b. Simpan program tersebut dengan nama new.asm yang sebenarnya namanya adalah new (penulisan nama tidak boleh hanya new kemudian ditekan tombol enter, karena akan menghasilkan nama file new.txt /berextension TXT bukan ASM). Tetapi sebelum menyimpan file tersebut dengan nama new.asm buat direktori khusus menyimpan file-file belajar mikrokontroler, sehingga tidak bercampur dengan file lainnya. Misal nama direktori adalah dengan nama topview di drive D:\topview. Jadi setelah penyimpanan maka file akan berada D:\topview>new.asm (.asm → Assembler, .TXT → Text). Jadi setelah proses 1 dan 2 selesai maka akan mempunyai 1 direktori di drive D dengan nama topview (di drive D tersebut bila hardisk terpartisi, namun bila hardisk tidak terpartisi maka dapat dibuat direktori topview di drive C), dan didalam direktori tersebut ada 1 file dengan nama new.asm. proses 1 dan 2 ini harus bisa karena bila tidak bisa maka proses selanjutnya tidak bisa dilakukan.

#### 4. Simulasi Mikrokontroler

Mencoba membuat program simulasi sederhana dengan aplikasi pada LED. Sedangkan untuk melihat dari respon yang ditimbulkan oleh program yang dibuat, maka bisa melihatnya dengan menampilkan modul externalnya (LED).

Buka program Topview Simulator.

Pada windows → menu Start (klik) → Program → Topview Simulator (klik). Maka akan keluar tampilan (Select Device) :



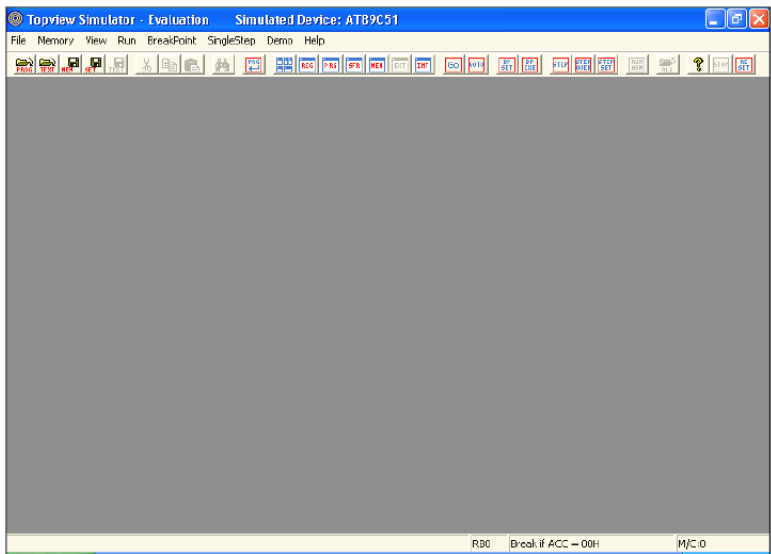
Gambar 2.7. Menu Select Device

Select Device : Atmel 89C51

Operating Frequency : 12000000 HZ

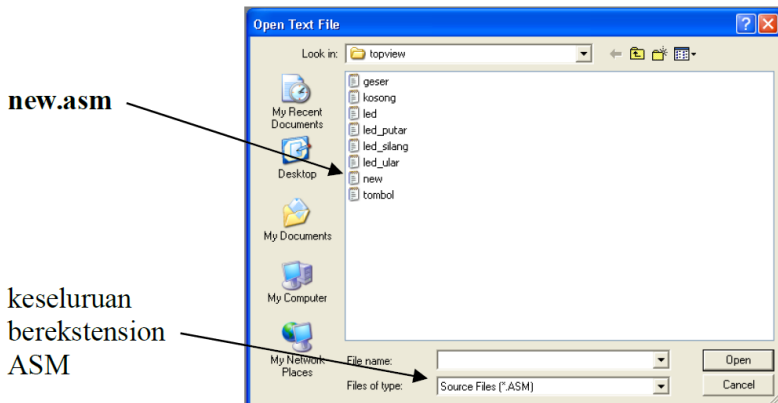
External Memory Setting : -

Setelah tampilan Select Device dan klik OK, maka akan keluar tampilan berikut:



Gambar 2.8. Menu Utama Topview Simulator

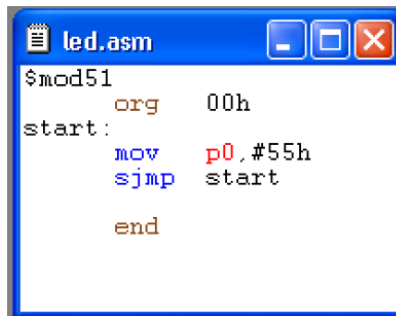
- a. Membuka file new.asm (file hasil proses nomor 1 dan 2).
  - Klik Menu File → Load Text File. Maka akan keluar tampilan berikut :



Gambar 2.9. Menu membuka file new.asm

- Cari direktori D:\topview dan pilih new.asm sebagai awal penulisan program. (file new.asm berfungsi untuk penulisan atau lembar awal penulisan program yang selanjutnya kita edit dan disimpan dengan nama lain/ Save TextFile As).

b. Tulis program aplikasi led berikut :

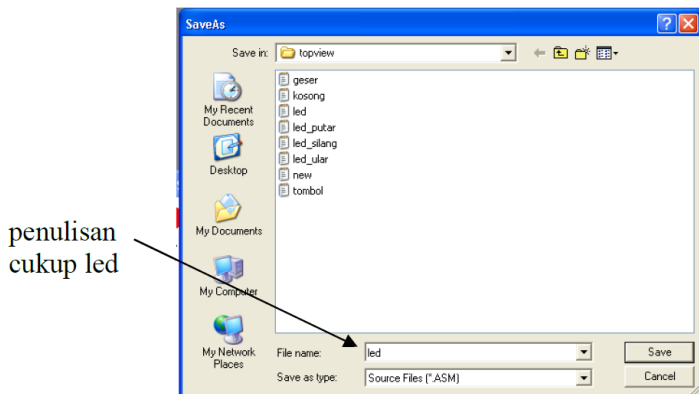


```

led.asm
$mod51
    org    00h
start:
    mov    p0, #55h
    sjmp   start
end
  
```

Gambar 2.10. Sumber kode assembler

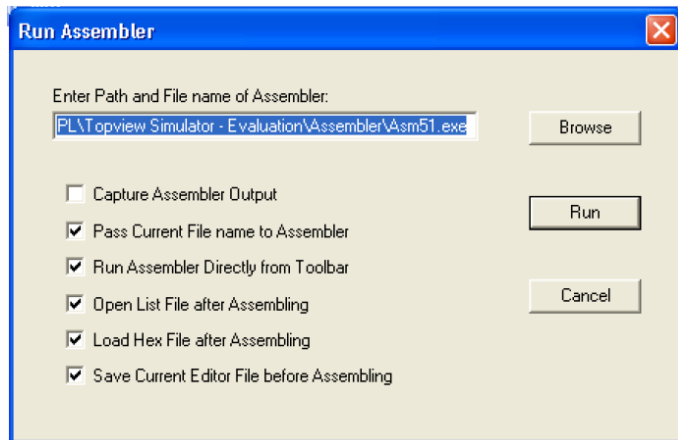
- Setelah selesai simpan dengan nama lain new.asm → led.asm  
→ Klik menu File → Save TextFile As...



Gambar 2.11. Mentimpan file sumber kode assembler

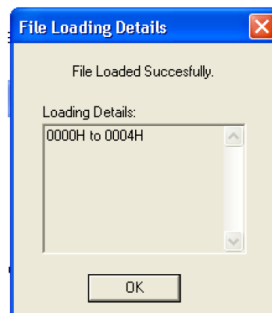
- Penyimpana file sebaiknya pada direktori yang sama (D:\topview)
  - Klik Save
- c. Kompilasi sumber kode assembler
- Tampilkan dahulu file led.asm (klik lembar file led.asm)
  - Klik menu Command → Run Assembler. Maka akan keluar tampilan Run Assembler





Gambar 2.12. Kompilasi sumber kode assembler

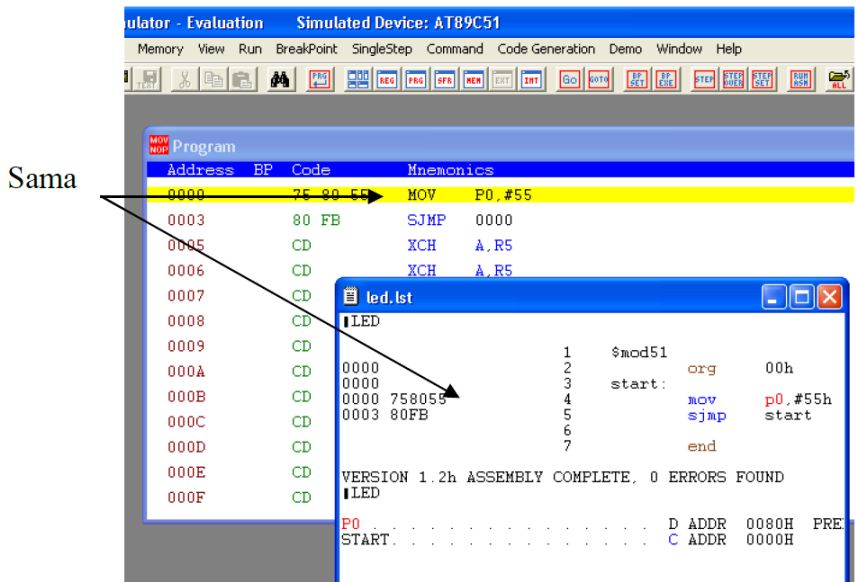
- Hilangkan tanda centang pada Option Capture Assembler Output (untuk menghilangkan tampilan output pada saat assembler), kemudian klik RUN untuk menjalankan proses compiler (assembler)
- Dari proses diatas akan keluar 2 tampilan yaitu File Loaded Succesfully dan File LST



Gambar 2.13. Proses kompilasi sumber kode assembler

- File LST → led.lst adalah merupakan file perpaduan yaitu file Heksa dan Assembler yang jadi satu, serta penunjuk dari kesalahan sintak (penulisan program) berupa tanda anak panah.

- d. Melihat isi memori mikrokontroler
- Pada tampilan diatas menyatakan bahwa program yang kita tulis (led.asm → led.hex) akan menempati memori pada alamat 0000h sampai 0004h. Klik OK
  - Pembuktian alamat 0000h sampai 0004h adalah dengan menampilkan jendela Program Window, caranya klik Menu View → klik Program Window

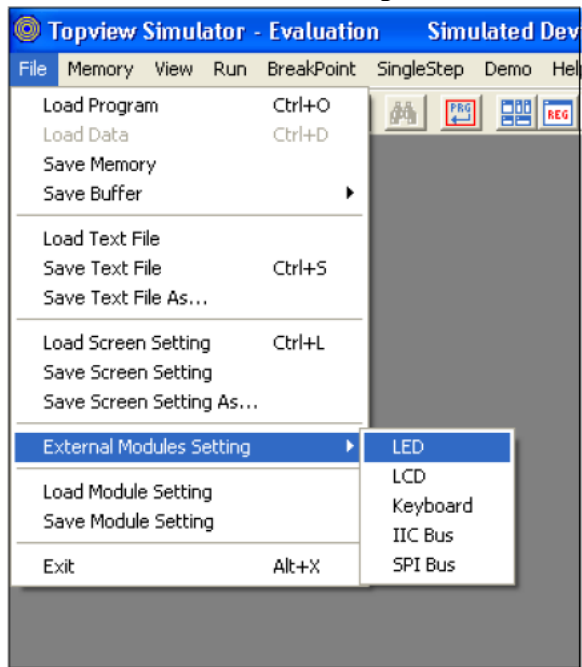


Gambar 2.14. Hasil kompilasi sumber kode assembler

Keterangan :

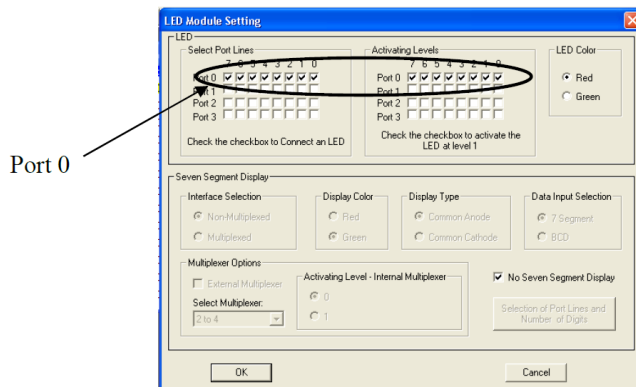
- Address : Alamat dari program
- Code : Program yang berupa kode (bilangan) hexsa /bahasa mesin
- Mnemonics : Program yang kita tulis sebelumnya (penulisan nama label diganti dengan nomor address, misal tunda → 0010)

- e. Menampilkan/setting modul led  
 klik menu File → External Modul Setting → LED



Gambar 2.15. External Modul Setting

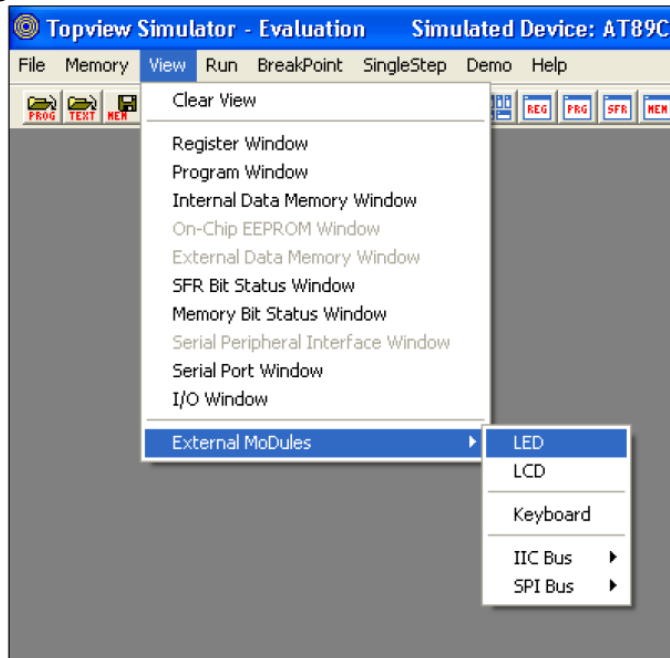
Cara mensetting :



Gambar 2.16. LED Modul Port Setting

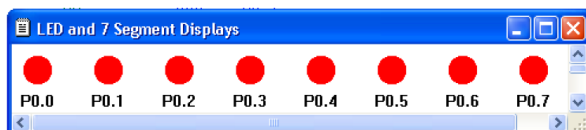
- Beri tanda centang (√) bit 0 – 7 pada Select Port Lines
- Pada Activating Levels masing-masing bit beri tanda centang, agar bila diberi logika 1 (high) led akan menyala dan sebaliknya (bila tanpa tanda centang maka diberi logika 1 akan padam/ logika 0 low akan menyala).
- Led Color yaitu untuk menentukan jenis warna dari led Red (merah) atau Green (Hijau).
- Beri tanda centang pada No Seven Segment Display
- Klik OK

Setelah melakukan setting, maka tampilkan modul led tersebut dengan cara :



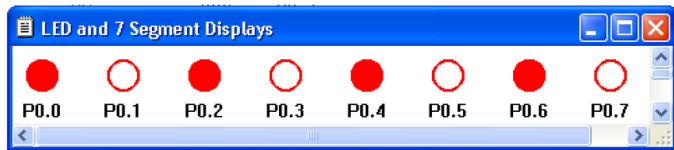
Gambar 2.17. Menampilkan LED Modul

Klik menu View → External Moduls → LED

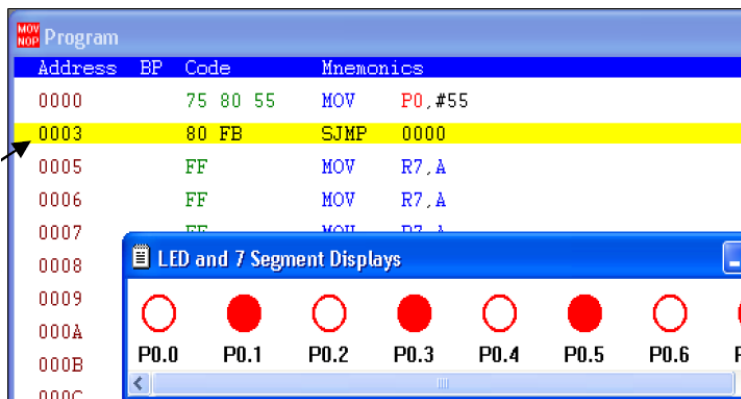


Gambar 2.18. Tampilkan modul LED

- f. Menjalankan simulasi dengan mengklik ikon GO  
Activating levels 1 (diberi tanda centang)/ kondisi active high:



Gambar 2.19. Modul LED activating levels 1  
Activating levels 0(tidakdiberi tanda centang)/kondisi active low:



Gambar 2.20. Modul LED activating levels 0

Jalanya program dapat dilihat sesuai dengan gerakan blok warna kuning pada window program.

- g. Menghentikan simulasi dengan mengklik ikon STOP  
sedangkan fungsi dari ikon RESET adalah untuk mengembalikan jalanya (Runningnya) program ke alamat awal (0000h)

Catatan :

Usahakan sebelum melakukan setiap simulasi untuk melakukan reset terhadap mikrokontroler, dengan cara mengklik RESET untuk mengembalikan seluruh nilai register ke nilai awalnya.

- h. Pembahasan program led.asm

Alamat program akan dimulai alamat 0000h pada memori start:

```
mov p0,#55h ; Port 0 diberi nilai 55h → 01010101B
sjmp start ; kembali ke start untuk berulang terus menerus.
```

end

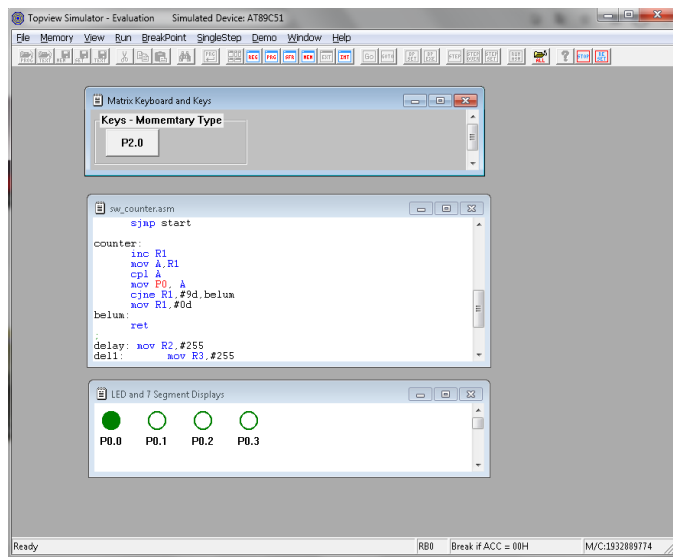


```

cpl A
mov P0, A
cjne R1,#9d,belum
mov R1,#0d
belum:
ret
;
delay: mov R2,#255
del1:   mov R3,#255
del2:   djnz R3,del2
        djnz R2,del1
        ret
end

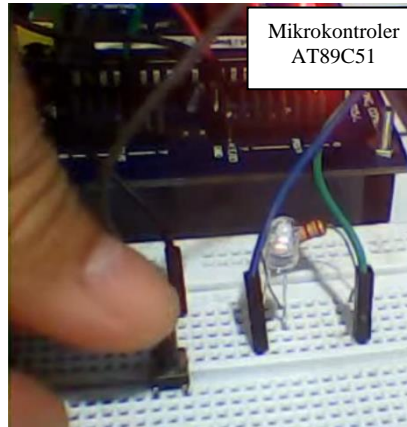
```

## b. TopView Simulator



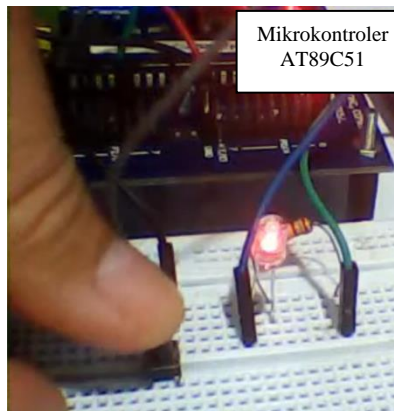
Gambar 2.21. Eksperimen Button dan LED

- c. Mikrokontroler AT89C51  
- Button OFF maka LED OFF



Gambar 2.22. Eksperimen Button OFF dan LED OFF

- Button ON maka LED ON



Gambar 2.23. Eksperimen Button ON dan LED ON





```

    call delay
    call delay
    call delay
usai:
    sjmp start
counter:
    inc R3
    mov A,R3
    mov DPTR,#DataDesimal
    movc A,@A+DPTR
    mov r1,#88h
    acall write_inst
    mov R1,A
    call write_data
    cjne R3,#9d,belum
    mov R3,#0d
belum:
    ret
Init_lcd:
    mov r1,#00000001b ;Display clear
    call write_inst
    mov r1,#00111000b ;Function set, Data 8 bit,2 line font 5x7
    call write_inst
    mov r1,#00001100b ;Display on, cursor off,cursor blink off
    call write_inst
    mov r1,#00000110b ;Entry mode, Set increment
    call write_inst
    ret
;
Write_inst:
    clr P2.6 ; P2.6 = RS =0
    clr P2.5 ; P2.6 = R/W =0
    mov P0,R1 ; copy R1 ke P0
    setb P2.7 ; P3.7 =EN = 1
    call delay ; call delay time
    clr P2.7 ; P3.7 =EN = 0
    ret
;

```

Write\_data:

```
setb P2.6 ; P3.6 = RS = 1
clr P2.5 ; P2.6 = R/W = 0
mov P0,R1 ; copy R1 ke P0
setb P2.7 ; P3.7 =EN = 1
call delay ; call delay time
clr p2.7 ; P3.7 =EN = 0
ret
```

delay: mov R0,#0

delay1:mov R7,#0fh

djnz R7,\$

djnz R0,delay1

ret

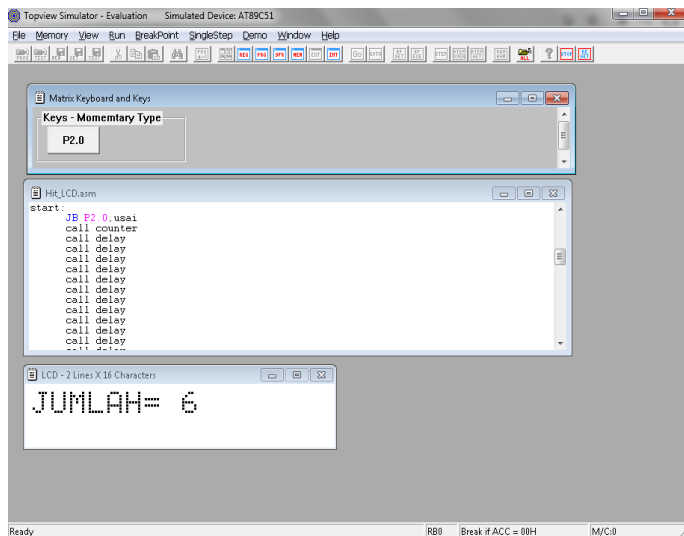
;

DataDesimal:

```
db '0','1','2','3','4','5','6','7','8','9'
```

end

## b. TopView Simulator



Gambar 2.24. Eksperimen Dekade Counter pada LCD

c. Mikrokontroler AT89C51

- Button OFF dan Counter sementara pada LCD = 0



Gambar 2.25. Eksperimen Button OFF maka LCD = 0

- Button ON dan Counter bertambah pada LCD = 2



Gambar 2.26. Eksperimen Button OFF maka LCD = 2

### 3. Dekade Counter pada 7-Segment

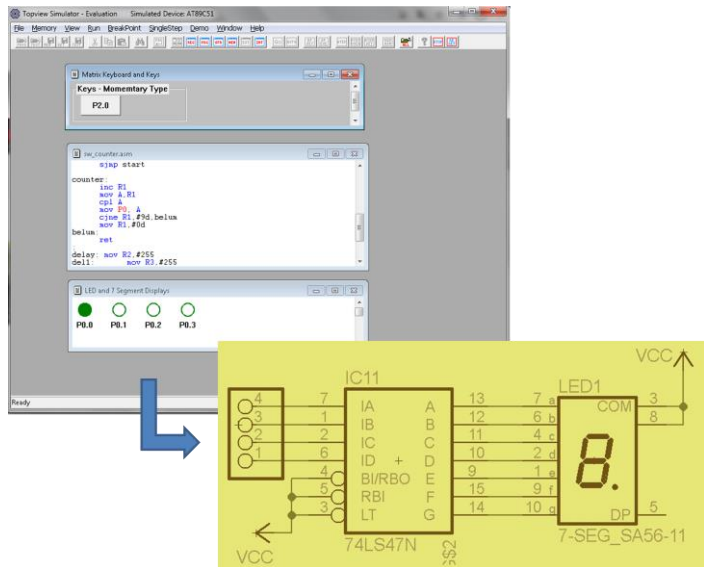
#### a. Asembler

```

$mod51
ORG 000H
START:
    MOV A,#00001001B
    MOV B,A
    MOV R0,#0AH
LABEL:
    JB P2.0,usai
    MOV A,B
    INC A
    MOV B,A
    MOVC A,@A+PC
    MOV P0,A
    ACALL DELAY
    DEC R0
    MOV A,R0
usai: JZ START
    SJMP LABEL
    DB 3FH ;digit drive pattern for 0
    DB 06H ;digit drive pattern for 1
    DB 5BH ;digit drive pattern for 2
    DB 4FH ;digit drive pattern for 3
    DB 66H ;digit drive pattern for 4
    DB 6DH ;digit drive pattern for 5
    DB 7DH ;digit drive pattern for 6
    DB 07H ;digit drive pattern for 7
    DB 7FH ;digit drive pattern for 8
    DB 6FH ;digit drive pattern for 9
DELAY:
    MOV R4,#05H ;subroutine for delay
WAIT1: MOV R3,#00H
WAIT2: MOV R2,#00H
WAIT3: DJNZ R2,WAIT3
        DJNZ R3,WAIT2
        DJNZ R4,WAIT1
    RET
END

```

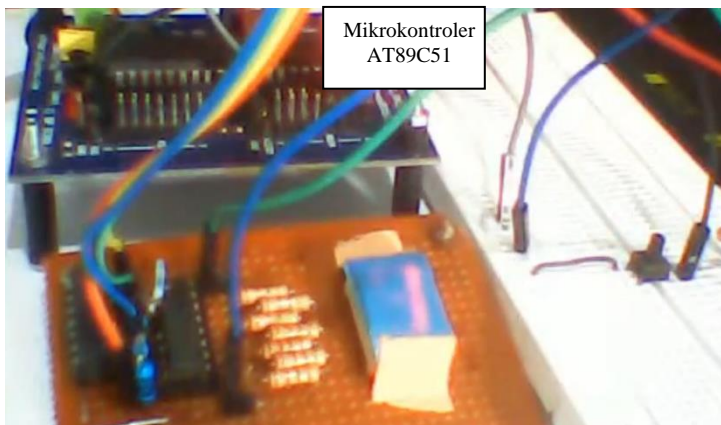
## b. TopView Simulator



Gambar 2.27. Eksperimen Dekade Counter pada 7-Segment

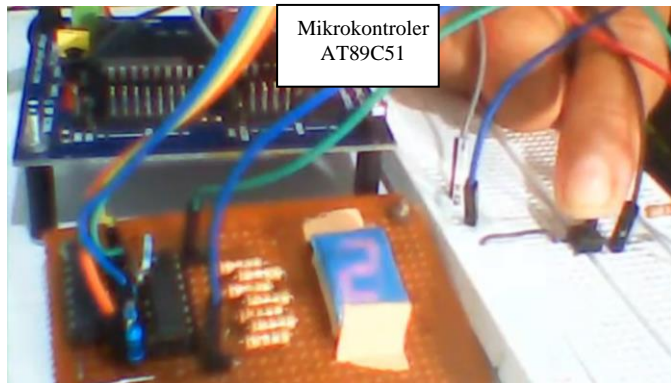
## c. Mikrokontroler AT89C51

- Button OFF dan Counter sementara pada 7-Segmen = 1



Gambar 2.28. Button OFF dan Counter pada 7-Segmen = 1

- Button OFF dan Counter bertambah pada 7-Segmen = 2



Gambar 2.29. Button OFF dan Counter pada 7-Segmen = 2

#### 4. Sensor Suhu pada LCD

##### a. Assembler

\$mod51

```
org 0h
ratusan equ 30h
puluhan equ 31h
satuan equ 32h
```

```
;
```

```
org 0h
call init_LCD
call write_char
start: call ADC
call Bin2Dec
call Write2LCD
sjmp start
```

```
;
```

```
=====
;Subrutin ini digunakan untuk mengambil data ADC MUX X0
=====
```

```
ADC:
clr P3.3
```

```

nop
nop
nop
setb P3.3
eoc: jb P3.2,eoc
clr P3.4
mov A,P1
setb P3.4
ret

```

```
;
```

```
=====
==
```

```

;Subrutin untuk menampilkan data ke LCD character 2 x16
;pada DDRAM 0C9 0CA 0CB untukratusan, puluhan, and satuan

```

```
=====
==
```

```

Write2LCD:
mov r1,#0c5h
call write_inst
mov a,ratusan
add a,#30h
mov r1,a
call write_data
;
mov r1,#0c6h
call write_inst
mov a,puluhan
add a,#30h
mov r1,a
call write_data
;

```

```

mov r1,#0c7h
call write_inst
mov a,satuan
add a,#30h
mov r1,a
call write_data

```



```

mov r1,#0c9h
acall write_inst
mov r1,#11011111b
acall write_data

```

```

mov r1,#0cah
acall write_inst
mov r1,#'C'
acall write_data

```

```
ret
```

```
;
```

```

;=====
;Subrutin ini untuk merubah data biner ke desimal
;menjadi 3 digit = ratusan-puluhan-satuan
;=====

```

```
Bin2Dec:
```

```

mov b,#100d
div ab
mov ratusan,a
mov a,b
mov b,#10d
div ab
mov puluhan,a
mov satuan,b
ret

```

```
;
```

```

;=====
;Subrutin untuk menampilkan tulisan Data ADC0804
;pada baris 1
;=====

```

```
write_char:
```

```

mov dptr,#word1 ;DPTR = [ address word1 ]
mov r3,#16 ;R3=16,number character to be display
mov r1,#80h ;R1=80h,address DDRAM start position
acall write_inst

```

```
;
```

```

write1:clr a ; A = 0
movc a,@a+dptr ; A = [A+ DPTR]

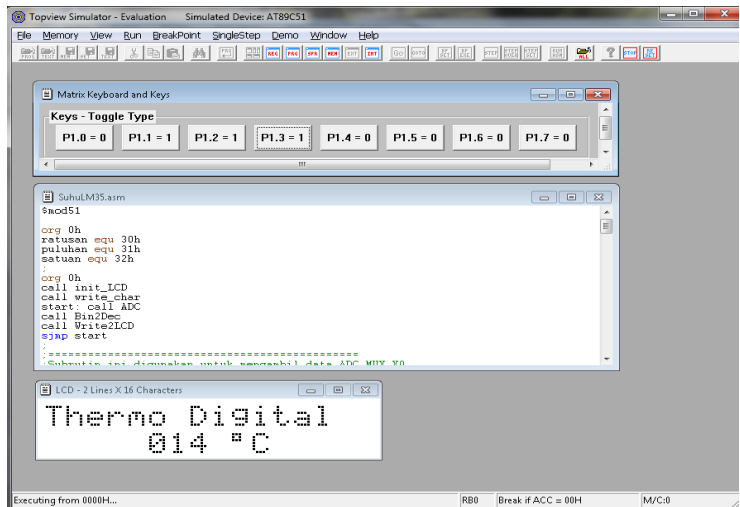
```

```

mov r1,A ; R1 = A
inc dptr ; DPTR = DPTR +1
acall write_data ;
djnz r3,write1 ; R3 = R3-1,
ret
;
Init_lcd:
mov r1,#00000001b ;Display clear
call write_inst
mov r1,#00111000b ;Function set,Data 8 bit,2 line font 5x7
call write_inst
mov r1,#00001100b ;Display on, cursor off,cursor blink off
call write_inst
mov r1,#00000110b ;Entry mode, Set increment
call write_inst
ret
write_inst:
clr P2.5 ; RW
clr P2.6 ; RS = P2.0 = 0, write mode instruction
mov P0,R1 ; D7 s/d D0 = P0 = R1
setb P2.7 ; EN = 1 = P2.1
call delay ; call delay time
clr P2.7 ; EN = 0 = P2.1
ret
Write_data:
clr P2.5 ; RW
setb P2.6 ; RS = P2.0 = 1, write mode data
mov P0,R1 ; D7 s/d D0 = P0 = R1
setb P2.7 ; EN = 1 = P2.1
call delay ; call delay time
clr p2.7 ; EN = 0 = P2.1
ret
delay: mov R0,#0
delay1:mov R2,#0fh
djnz R2,$
djnz R0,delay1
ret
word1: DB 'Thermo Digital '
end

```

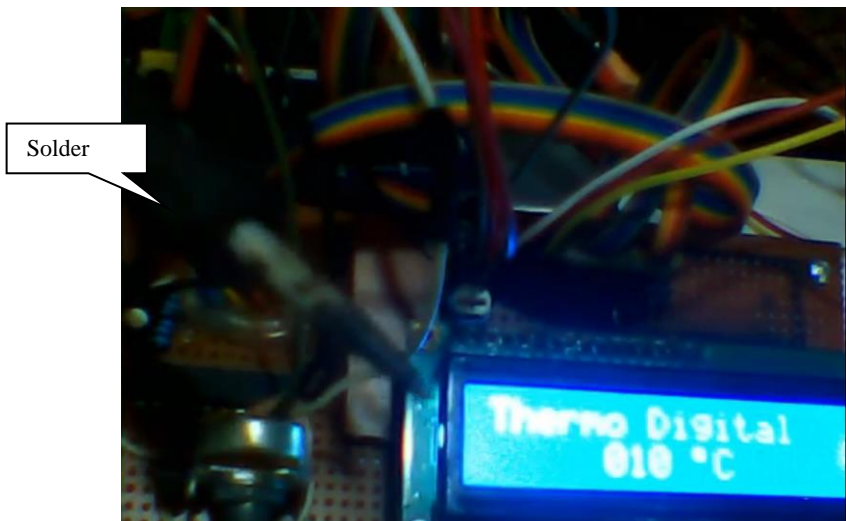
## b. TopView Simulator



Gambar 2.30. Eksperimen Sensor Suhu pada LCD

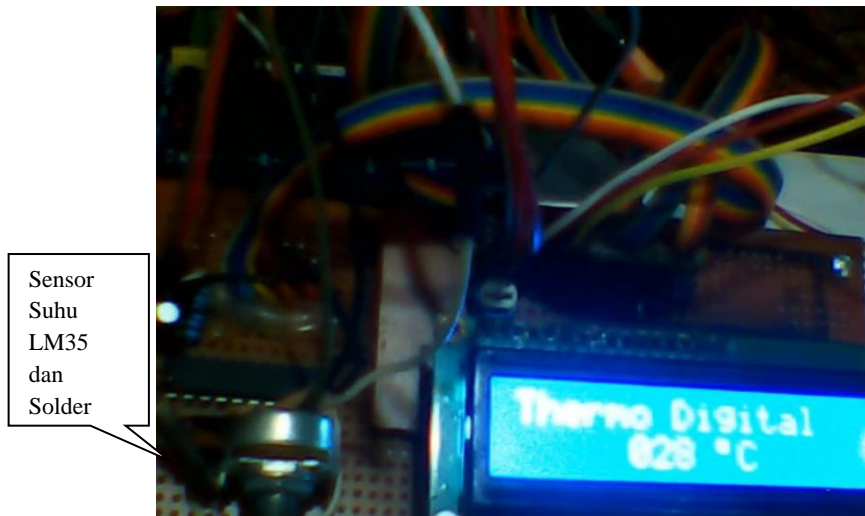
## c. Mikrokontroler AT89C51

- Mikrokontroler AT89C51, Sensor Suhu LM35



Gambar 2.31. Eksperimen Sensor Suhu LM35

- Mikrokontroler AT89C51, Sensor Suhu LM35 dan Solder



Gambar 2.32. Eksperimen Sensor Suhu LM35 dan Solder

E. Rujukan

Usman, (2008), Teknik Antarmuka + Pemrograman Mikrokontroler AT89s52, Yogyakarta : Andi.

Syahrul, (2014), Pemrograman Mikrokontroler AVR Bahasa Assembly dan C, Bandung: Informatika.

F. Bacaan yang dianjurkan

Paulus Andi Nalwan, (2003), Panduan Praktis Teknik Antarmuka dan Pemrograman Mikrokontroler AT89c51, Jakarta: PT. Elex Media Komputindo

# Dasar Mikrokontroler I

**Eksperimen AT89C51 menggunakan  
TopView Simulator**

**Dr. Unang Achlison, S.T., M.Kom**



**YAYASAN PRIMA AGUS TEKNIK**

**YAYASAN PRIMA AGUS TEKNIK**  
Jl. Majapahit No. 605 Semarang  
Telp. (024) 6723456. Fax. 024-6710144  
Email : [penerbit\\_ypat@stekom.ac.id](mailto:penerbit_ypat@stekom.ac.id)

ISBN 978-623-6141-58-8 (PDF)

