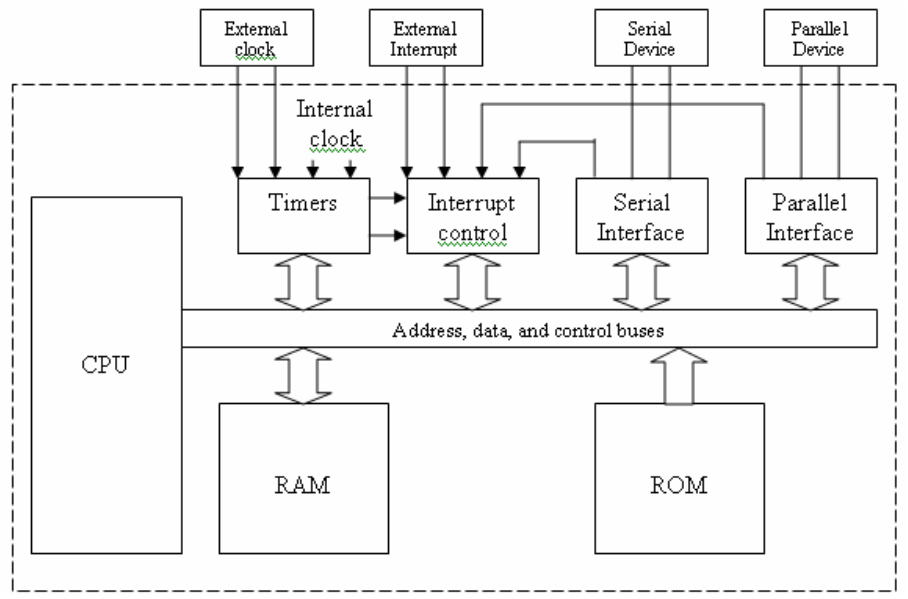
BAB II
LANDASAN TEORI

2.1 Arsitektur Mikrokontroller 8051

2.1.1 Mikroprosessor dan Mikrokontroller

Mikrokontroller secara harfiah merupakan pengatur berukuran mikro. Berbeda dengan mikroprosessor yang merupakan CPU (*Central Processing Unit*) chip tunggal yang digunakan dalam mikrokomputer, mikrokontroller memiliki sebuah CPU serta fitur rangkaian yang terdapat pada mikrokomputer yang terintegrasi dalam sebuah IC (*Integrated Circuit*), sehingga dalam satu IC mikrokontroller terdapat sebuah CPU, RAM (*Random Access Memory*), ROM (*Read Only Memory*), sebuah antarmuka serial, sebuah antarmuka parallel, *timer*, dan *interrupt scheduling circuity* (rangkaian penjadwalan interupsi).



Gambar 2.1 Blok diagram sistem mikrokomputer

Namun kapasitas *on-chip* RAM (RAM internal) tidak sebesar RAM pada mikrokomputer saat ini, oleh karenanya mikrokontroller hanya dipergunakan

untuk aplikasi tertentu. Gambar 2.1 memperlihatkan sebuah sistem mikrokomputer yang terintegrasi dalam sebuah IC mikrokontroller. Fitur penting lainnya dari mikrokontroller adalah sistem *interrupt* terintegrasi. Sebagai perangkat berorientasi kontrol, mikrokontroller sering digunakan untuk sebuah sistem yang merespon *external stimuli* (interupsi) secara *real time*. Mikrokontroller harus dapat melakukan tugas *switching* dengan cepat, menunda suatu proses sementara waktu mengeksekusi yang lain ketika merespon sebuah "*event*". Misal pada produk microwave berbasis mikrokontroller, membuka pintu microwave merupakan contoh "*event*" yang mengakibatkan sebuah *interrupt* terhadap mikrokontroller.

Mikroprosessor umumnya digunakan sebagai CPU dalam sistem mikrokomputer. Inilah tujuan utama mikroprosessor didesain, dan ini merupakan keutamaan yang mikroprosessor miliki. Sedangkan mikrokontroller didesain untuk mengakomodasi desain minumum yang memiliki ukuran kecil dan komponen minimal serta berorientasi ke aktivitas bersifat kontrol. Desain sistem dengan menggunakan ribuan IC dapat direduksi dengan penggunaan sebuah mikrokontroller. Yang dibutuhkan hanyalah sebuah mikrokontroller terprogram dalam ROM dan sejumlah kecil komponen pendukung. Mikrokontroller sangat cocok untuk mengontrol perangkat I/O dengan desain sistem minimalis, sedangkan mikroprosessor cenderung untuk penggunaan pemrosesan informasi dalam sistem komputer.

2.1.2 Fitur Standar MCS-51

Keluarga 8051 atau biasa disebut MCS-51 pertama kali dikembangkan oleh Intel Corporation pada tahun 80-an, sehingga dapat dibilang usia MCS-51 sudah mencapai 26 tahun. MCS-51 merupakan salah satu keluarga mikrokontroller yang sampai sekarang masih banyak dikembangkan oleh berbagai produsen semacam Atmel Corp., Philips Semiconductors, Cygnal Integrated Products, Inc., dan Winbond Electronics Corp. Berbagai kemampuan / fitur yang dimiliki MCS-51 masih terus ditingkatkan. Arsitektur dasarnya merupakan derivasi dari mikrokontroller 8048, yang diperkenalkan pada tahun 1976.

Meskipun banyak produsen lain yang menggunakan arsitektur ini, 8051 memiliki fitur original sebagai berikut :

- Frekuensi operasi sampai 12 MHz
- Pin I/O sebanyak 32 (tersusun atas 4 port, dengan aransemen 8 bit per port nya)
- RAM sebesar 128 byte
- Ada 3 versi dengan opsi memori program yang berbeda :
  - Tidak memiliki memori program (membutuhkan eksternal memori, misal 8031)
  - 4K x 8 bit *internal mask-programmed* ROM (8051)
  - 4K x 8 bit UV-*erasable* EPROM (8751)
- *Timer* / *Counter* 16 bit dua buah (*Timer* 0 dan *Timer* 1)
- 5 Sumber *interrupt* (2 external) dengan 2 prioritas level
- Port serial *full-duplex*

MCS-51 memiliki pin sebanyak 40 pin, 32 pin merupakan 4 port I/O 8 bit, sisanya adalah pin Vcc, *Ground*, EA, ALE, PSEN, XTAL (X1 dan X2), dan *reset*. Fitur original 8051 dengan 4 Kb ROM tidak terlalu kecil dan tidak terlalu besar, 128 byte RAM (termasuk SFR) dapat memenuhi keperluan standar, 4 port berjumlah 32 jalur I/O sangat cukup memenuhi aplikasi standar. Keluraga MCS-51 telah berkembang pesat dan memiliki berbagai macam varian. Perbedaan antar varian keluarga MCS-51 dapat dikelompokkan menjadi 8 bagian, yaitu :

1. Tegangan kerja

   Tegangan kerja AT89LV55 buatan Atmel mampu beroperasi dengan tegangan kerja 2.7 – 6 V. Seri P89LPC9xx buatan Philips memiliki tegangan kerja 2.4 – 3.6 V.
2. Memori dan Pemrogram

   Memori proram internal yang dimiliki AT89C51 sebesar 4Kb sedangkan memori data internal yang dimilikinya sebesar 256 byte. Tipe C8051F12x buatan Cygnal memiliki memori program internal

128 Kb, sedangkan memori data internal yang dimilikinya sebesar 8448 byte.

Tidak hanya kapasitas memori, tipe memori yang digunakan pun bervariasi, yaitu :

- OTP (*One Time Programmable*) / *Mask* ROM (*Read Only Memory*)
- MTP (*Multiple Time Programmable*) UVEPROM (*Ultra-Violet Erasable Programmable* ROM)
- MTP *Flash* / EEPROM.

Cara penulisan memori program internal pun bervariasi, antara lain : *In Circuit Programming* (ICP), *In Application Programming* (IAP), *In System Programming* (ISP), dan *Parallel Programming* (PP).

3. Kecepatan

   Dalam hal kecepatan, ada dua satuan yang digunakan. Beberapa produsen menggunakan satuan MHz (*MegaHertz*), sedangkan yang lain menggunakan sautan MIPS (*Million Instructions Per Second*). AT83C5111 memiliki frekuensi maksimum 66 MHz sedangkan C8051F120 buatan cygnal mampu memproses 100 MIPS. *Clock rate* umumnya bernilai 12, yang berarti *internal clock* beroperasi 1/12 dari frekuensi sumber *clock* atau 1 *cycle* membutuhkan 12 pulsa *clock*. IC buatan Cygnal memiliki *clock rate* 1 dimana 1 *cycle* membutuhkan 1 pulsa *clock*.

4. *Timer* / *Counter*

   Timer / Counter yang dimiliki varian MCS-51 dapat berjumlah hingga 5 buah, contohnya P89LPC932 buatan Philips. Beberapa varian bahkan memiliki fasilitas *Pulse Width Modulation* (PWM), *Programmable Counter Array* (PCA), dan *WatchDog Timer*.

5. *Interrupt*

   Tipe P89LPC932 buatan Philps memiliki 2 *interrupt source* : *external interrupt* 0 dan 1, *timer* 0 dan 1, *serial port* Tx, *serial port* Rx, *combined serial port* Rx/Tx, *brownout detect*, *watchdog/Real-Time*

clock, I^2C (*Inter Integrated Circuit*), keyboard, *comparator* 1 dan 2. SPI (*Serial Peripheral Interface*), CCU (*Capture/Compare Unit*), dan data EEPROM *write completion*. Tipe tersebut memiliki *interrupt level* / *priority* hingga 4 tingkat.

6. *Serial Interface* (Antarmuka Serial)

   Serial Interface yang banyak didukung keluarga MCS-51 adalah *Universal Asynchronous Receiver Transmitter* (UART). Namun varian yang lebih baru juga memiliki interface I^2C atau *Two Wire Interface*, SPI, CAN (*Control Area Network*), bahkan USB (*Universal Serial Bus*).

7. I/O

   Jumlah pin yang berfungsi sebagai digital input / output pada IC buatan Cygnal dapat berjumlah mulai 8, 16, 32, hingga 64 pin.

8. Fungsi khusus

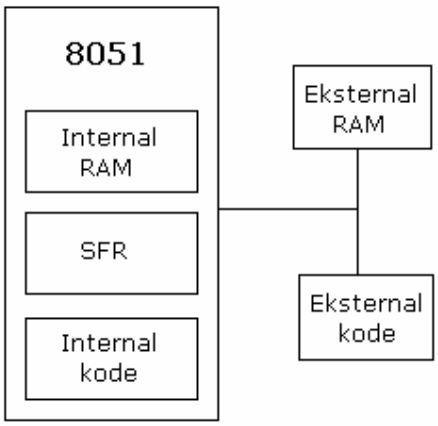
   Selain perbedaan kapasitas memori, kecepatan, jumlah *timer*, *interface*, dan jalur input / output, varian MCS-51 juga memiliki kemampuan spesifik. Beberapa kemampuan yang terdapat pada varian MCS-51 antara lain :

   - ADC (*Analog to Digital Converter*) hingga 12 bit 32 *channel* pada C8051F20 buatan Cygnal
   - 8 *Keyboard Interrupt*, *Power-On Reset* dan *Brown-Out Detect* pada P87LPC768 buatan Philips

- W925E / C240 dan W925E / C625 buatan Winbond memiliki DTMF (*Dual Tone Multiple Frequency*) *generator* / *receiver* dan FSK (*Frequency-Shift Keying*) *generator* / *receiver* yang terintegrasi. Adapun 40 pin pada mikrokontroller 8051 original masih memiliki fungsi yang sama pada mikrokontroller turunannya.

2.1.3 Tipe Memori

8051 secara umum memiliki 3 tipe memori, yaitu : *On-Chip* memori, memori kode eksternal dan RAM (*Random Access Memory*) eksternal.



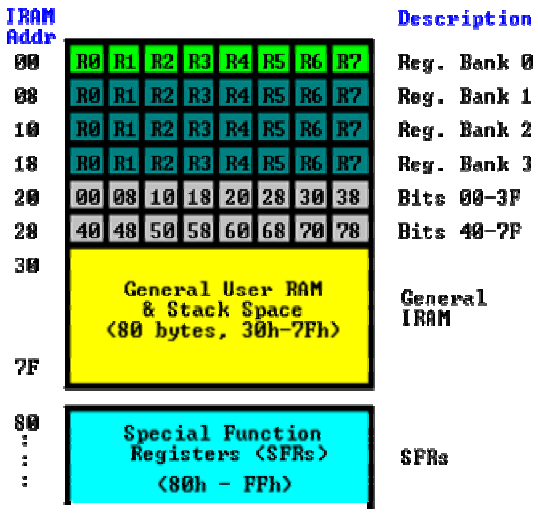
Gambar 2.2 Tipe memori 8051

On-Chip memori merupakan memori (Kode dan RAM) yang secara fisik terintegrasi dalam mikrokontroller itu sendiri. Memori kode eksternal merupakan memori program yang terpisah dari mikrokontroller. Jenisnya, umumnya, berupa eksternal EEPROM. RAM eksternal merupakan RAM yang terpisah dari mikrokontroller.

Memori kode merupakan memori yang menampung program yang akan dijalankan. Memori ini terbatas pada 64 Kb dan hadir dalam beragam bentuk dan ukuran. Memori kode dapat berupa *on-chip*, yang tertanam dalam mikrokontroller sebagai ROM (*Read Only Memory*) atau EPROM (*Erasable Programmable Read Only Memory*). Kode program juga dapat disimpan secara *off-chip* dalam sebuah eksternal ROM atau EPROM.

Eksternal RAM merupakan RAM secara *off-chip*. Karena memori secara *off-chip*, maka tidak fleksibel dalam pengaksesannya dan juga sangat lambat. Sebagai contoh, untuk menaikkan lokasi satu internal RAM, dibutuhkan cukup 1 instruksi dan 1 siklus instruksi. Sedangkan untuk menaikkan satu byte pada eksternal RAM dibutuhkan 4 instruksi dan 7 siklus instruksi. Maka Eksternal RAM lebih lambat 7 kali lipat. Namun keuntungannya, jumlah ruang memori juga berlipat. Internal RAM terbatas sampai 128 byte (256 byte untuk 8052) dan 8051 mendukung eksternal RAM sampai 64 Kb.

On-Chip memori terbagi menjadi 2 tipe, yaitu : Internal RAM dan *Special Function Register* (SFR). Layout dari internal RAM 8051 digambarkan dalam peta memori berikut :



Gambar 2.3 Peta Memori on-chip RAM 8051

Seperti diilustrikan pada peta memori di atas, 8051 memiliki internal RAM sebesar 128 byte. RAM internal ini merupakan fitur *on-chip* dari 8051, jadi merupakan RAM tercepat yang tersedia, fleksibel dalam hal pembacaan, penulisan dan modifikasi isinya. Internal RAM bersifat *volatile*, jadi saat *reset* memori ini akan dibersihkan. Sebesar 128 byte dari internal RAM terbagi menjadi beberapa bagian. 8 byte pertama (00h-07h) merupakan register bank 0. Dengan memanipulasi isi dari SFR, program dapat memilih untuk menggunakan register bank 0, 1, 2 atau 3. Register bank ini terletak dalam internal RAM mulai dari alamat 08 sampai 1F. Bit memori juga merupakan bagian dari internal RAM, dimulai dari alamat 20h sampai 2Fh. 80 byte berikutnya, dari alamat 30h sampai 7Fh, dapat digunakan oleh pengguna untuk ruang variabel yang sering diakses atau pengaksesan yang membutuhkan kecepatan tinggi. Ruang ini juga digunakan oleh mikrokontroller sebagai ruang penyimpanan dari operasi *stack*. Kenyataan ini membatasi *stack* dari 8051, karena ruang yang disediakan untuk *stack* hanya

sebesar 80 byte, dan biasanya kurang dari 80 byte, karena harus berbagi antara *stack* dan variabel yang didefinisikan oleh pengguna.

Mikrokontroller 8051 memiliki kemampuan untuk mengakses sejumlah variabel bit. Variabel seperti ini dapat bernilai 1 atau 0. Ada 128 bit variabel yang tersedia untuk pengguna, mulai dari 00h-7Fh. Pengguna dapat menggunakan variabel ini dengan perintah `SETB` dan `CLR`. Sebagai contoh, untuk menset variabel bit 24h ke 1 maka instruksi berikut dapat digunakan :

```
SETB 24h
```

Bit memori merupakan bagian dari internal RAM, dalam kenyataannya 128 bit variabel menggunakan 16 byte internal RAM dari alamat 20h - 2Fh. Jadi, jika menuliskan FFh ke internal RAM alamat 20h, maka secara efektif akan menset bit 00h sampai 07h. Dengan begitu maka instruksi

```
MOV 20h,#0FFh
```

Akan sama dengan :

```
SETB 00h
SETB 01h
SETB 02h
SETB 03h
SETB 04h
SETB 05h
SETB 06h
SETB 07h
```

Seperti diilustrasikan di atas, bahwa bit memori bukan merupakan memori tipe baru. Bit memori hanya merupakan subset dari internal RAM. Namun karena 8051 menyediakan instruksi khusus untuk mengakses 16 byte memori ini secara

bit per bit, maka dapat dikatakan seperti halnya tipe memori yang berbeda.. Jika dalam program tidak menggunakan variabel bit ini, maka lokasi 20h – 20Fh dapat digunakan oleh pengguna secara bebas. Dalam penggunaannya, variabel bit (20h – 20Fh) dapat saja melakukan penimpaan nilai dari bit-bit yang didefinisikan oleh pengguna atau bank register, untuk itu pengguna harus memperhatikan *range* 128 bit serta pemberian nilai langsung ke dalam alamat 20h – 2Fh. Variabel bit dari 00h-7Fh dapat digunakan secara bebas oleh pengguna. Namun bit variabel 80h dan diatasnya sebenarnya digunakan untuk mengakses beberapa SFR secara bit. Contoh, jika jalur output P0.0 – P0.7 berlogika 0 (*clear*) dan pengguna ingin memberi logik 1 pada jalur output P0.0, maka dapat digunakan instruksi berikut :

```
MOV P1, #01h ;atau SETB 80h
```

Kedua instruksi di atas mengerjakan hal yang sama. Bagaimanapun juga, dengan menggunakan instruksi `SETB` hanya akan menyalakan jalur P0.0 tanpa member efek terhadap pin P0 lainnya. Namun perintah `MOV` secara efektif akan mematikan pin P0 lainnya. Secara *default*, 8051 akan menginisialisasi *Stack Pointer* (SP) ke 07h ketika mikrokontroller melakukan *booting*. Hal ini berarti bahwa *stack* akan dimulai pada alamat 08h dan terus ke alamat diatasnya. Jika pengguna menggunakan register bank alternatif (bank 1, 2 atau 3), maka pengguna perlu menginisialisasi *stack pointer* ke alamat di atas alamat teratas dari register bank yang digunakan, bila tidak maka *stack* dapat mengacaukan isi dari register bank. Sama halnya terhadap penggunaan variabel bit, pengguna perlu menginisialisasi *stack pointer* ke alamat di atas 2Fh untuk menjamin variabel bit terlindungi dari *stack*.

2.1.4 Jenis Register 8051

2.1.4.1 Bank Register

8051 menggunakan 8 register “R” yang diberi nama R0-R7. Register ini secara umum digunakan untuk membantu dalam operasi manipulasi nilai dan

pemindahan data dari satu lokasi memori ke lokasi lainnya. Sebagai contoh, untuk menambah nilai dari R4 ke *accumulator*, instruksi berikut dapat digunakan :

```
ADD  A, R4
```

Jika *accumulator* (A) berisi nilai 6 dan R4 berisi nilai 3, maka *accumulator* menjadi 9 setelah instruksi tersebut dieksekusi. Sebagai terlihat pada peta memori di atas, register R (R4) merupakan bagian dari internal RAM, yaitu pada alamat 04h. Maka instruksi di atas sama dengan instruksi berikut :

```
ADD A, 04h
```

Instruksi ini akan menambah nilai *accumulator* dengan nilai yang terkandung pada alamat 04h, dan hasilnya disimpan dalam *accumulator*. Namun 8051 memiliki 4 *bank* register berbeda. Ketika pertama kali 8051 *booting*, register bank 0 secara *default* akan digunakan. Bagaimanapun juga, dalam program, dimungkinkan untuk menginstruksikan 8051 untuk menggunakan salah satu dari register *bank* yang ada, register *bank* 1, 2 atau 3. Pada kasus seperti ini, R4 tidak lagi berada pada lokasi 04h. Misal, jika dalam program digunakan register bank 3, maka register R (R4) akan menempati lokasi internal RAM alamat 1Ch. Konsep dari register bank menambah tingkat fleksibilitas dalam 8051, khususnya jika berhadapan dengan *interrupt*.

2.1.4.2 *Special Function Register* (SFR)

2.1.4.2.1 Penjelasan SFR

SFR merupakan sejumlah ruang memori yang mengontrol fungsi-fungsi khusus dari prosesor 8051. Contoh, 4 register SFR dapat mengakses 32 jalur I/O 8051. SFR lainnya dapat dimanfaatkan oleh program untuk membaca atau menulis serial port dari 8051. SFR lainnya dapat digunakan oleh pengguna untuk menset baud rate serial, mengontrol dan mengakses timer, dan mengkonfigurasi

sistem interupsi dari 8051. Dalam memprogram, SFR merupakan ilusi dari bentuk internal memori.

8051 merupakan mikrokontroller yang fleksibel dengan sejumlah mode operasi yang saling berhubungan. Program yang dibuat pengguna dapat saja mengatur mode operasi dari 8051 hanya dengan memanipulasi nilai SFR 8051. SFR dapat diakses seperti halnya pengaksesan internal memori (alamat 00h – 7Fh), namun register SFR terdapat dalam alamat 80h – FFh. Setiap register SFR memiliki sebuah alamat dan sebuah nama. Tabel berikut mempresentasikan SFR dari 8051 beserta alamat dan namanya :

Tabel 2.1 Lokasi SFR

80	P0	SP	DPL	DPH				PCON	87
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
90	P1								97
98	SCON	SBUF							9F
A0	P2								A7
A8	IE								AF
B0	P3								B7
B8	IP								B9
C0									C7
C8									CF
D0	PSW								D7
D8									DF
E0	ACC								E7
E8									EF
F0	B								F7
F8									FF

Seperti terlihat pada tabel di atas, meskipun *range* alamat dari 80h – FFh seharusnya memberikan total 128 alamat, pada kenyataanya hanya ada 21 SFR dalam standar 8051. Alamat lainnya dalam range alamat SFR merupakan alamat yang tidak sah. Penulisan atau pembacaan pada alamat yang tidak sah tersebut dapat menghasilkan nilai tidak terdefinisi. Hal ini juga mengakibatkan program tidak akan kompatibel dengan turunan 8051 lainnya.

2.1.4.2.2 Tipe SFR

Seperti yang dijelaskan pada tabel SFR di atas, SFR dengan warna background biru merupakan SFR yang berhubungan dengan port I/O. 8051 memiliki 4 port I/O 8 bit, sehingga total ada 32 jalur I/O. Apapun nilai yang diberikan ke jalur I/O, logik 1 atau 0, nilai ini dapat dibaca dengan mengontrol SFR warna backgroung hijau. SFR dengan warna background kuning merupakan pengontrol operasi atau sebagai pengkonfigurasi dari beberapa aspek 8051. Contoh, TCON dapat digunakan untuk mengontrol timer dan SCON untuk mengontrol serial port. Sisanya, warna background hijau, merupakan pemabantu operasi SFR lainnya, dalam pengertian register ini tidak secara langsung mengkonfigurasi 8051 tapi 8051 tidak dapat beroperasi tanpa register ini. Contoh, saat serial port telah dikonfigurasi dengan menggunakan SCON, program dapat membaca atau menulis ke serial port dengan menggunakan register SBUF. SFR dengan namanya berwarna merah dapat diakses dengan operasi bit (misal dengan SETB dan CLR).

2.1.4.2.3 Register Pada SFR

P0 (Port 0, Alamat 80h, dapat dialamati secara bit)

Merupakan I/O untuk port 0. Setiap bit dari register ini berkoresponedensi terhadap pin-pin port 0 pada mikrokontroller. Contoh, bit 0 dari P0 merupakan pin 0.0, bit 7 merupakan pin P0.7. Menulis 1 ke register ini akan mengirim logik 1 ke pin I/O bersangkutan.

SP (Stack Pointer)

SFR ini mengindikasikan dimana nilai berikutnya yang akan diambil dari stack akan dibaca. Jika pengguna mem-push sebuah nilai ke stack, nilai tersebut akan dituliskan ke alamat SP+1. Jadi jika SP berisi 07h, instruksi PUSH akan mem-push nilai ke stack pada alamat 08h. Register SFR ini dapat termodifikasi oleh semua instruksi stack, seperti PUSH, POP, LCALL, RET, RETI serta ketika adanya interupsi dari mikrokontroller. Pada saat startup, SFR SP akan diinisialisasi ke 07h, yang berarti stack akan dimulai pada alamat 08h dan alamat

teratas berikutnya pada internal RAM. Karena register bank alternative (bank 1, 2, atau 3) dan juga variabel bit terletak pada alamat 08h – 2Fh, maka sangat penting untuk menginisialisasi SP ke alamat yang berada di atas bank register dan variabel bit (jika pengguna menggunakannya).

DPL / DPH (Data Pointer Low / High, alamat 87h)

Register SFR DPL dan DPH bekerja bersama untuk mempresentasikan sebuah nilai 16 bit yang disebut data pointer. Data pointer digunakan dalam operasi terhadap akses eksternal RAM dan beberapa instruksi yang melibatkan memori kode. Nilainya bertipe unsigned 2 byte integer, jadi dapat berupa nilai dari 0000h – FFFFh (dalam decimal, 0 – 65.535)

PCON (Power Control, alamat 87h)

SFR PCON digunakan untuk mengontrol mode control power 8051. Mode operasi dari 8051 dianataranya adalah mode sleep, dimana pada mode tersebut power yang dibutuhkan lebih sedikit. Mode operasi seperti ini dapat diatur dalam register PCON. Selain itu PCON juga digunakan untuk salah satu bit pada PCON juga digunakan untuk mendouble efektifitas dari baud rate serial port pada 8051.

TCON (Timer Control, alamat 88h)

SFR TCON digunakan untuk mengkonfigurasi dan memodifikasi operasi dari 2 timer 8051. SFR ini mengontrol berkerjanya / berhentinya timer dan terdapat bit flag untuk mengindikasikan apakah timer mengalami overflow. Selain itu bit pada TCON juga digunakan untuk mengkonfigurasi pengaktifan eksternal interrupt dan juga menganding flag eksternal interrupt yang akan set apabila terjadi eksternal interrupt.

TMOD (Timer Mode, alamat 89h)

SFR TMOD digunakan untuk mengkonfigurasi mode operasi setiap timer. Dengan menggunakan register TMOD, pengguna dapat mengkonfigurasi timer untuk beroperasi pada mode 16 bit. 8 bit autoreload, 13 bit atau 2 timer terpisah.

Selain itu, pengguna juga dapat mengkonfigurasi timer hanya untuk menghitung ketika sebuah pin eksternal aktif atau untuk menghitung event yang terindikasikan pada eksternal pin.

TL0 / TH0 (Timer 0 Low / High, alamat 8Ah / 8Ch)

Kedua SFR ini mempresentasikan timer 0. Bagaimana register ini bekerja tergantung bagaimana timer dikonfigurasi oleh register TMOD. Namun timer ini selalu menhitung naik nilai. Yang dikonfigurasikan disini adalah bagaimana dan kapan timer menaikkan nilai.

TL1 / TH1 (Timer 0 Low / High, alamat 8Bh / 8Dh)

Kedua SFR ini mempresentasikan timer 1. Bagaimana register ini bekerja tergantung bagaimana timer dikonfigurasi oleh register TMOD. Namun timer ini selalu menhitung naik nilai. Yang dikonfigurasikan disini adalah bagaimana dan kapan timer menaikkan nilai.

P1 (Port 1, Alamat 90h, dapat dialamati secara bit)

Merupakan I/O untuk port 1. Setiap bit dari register ini berkoresponedensi terhadap pin-pin port 1 pada mikrokontroller. Contoh, bit 0 dari P1 merupakan pin 1.0, bit 7 merupakan pin P1.7. Menulis 1 ke register ini akan mengirim logik 1 ke pin I/O bersangkutan.

SCON (Serial Control, alamat 99h)

SFR SCON digunakan untuk mengkonfigurasi cara kerja serial port 8051. Register ini mengontrol baud rate dari port serial, apakah port serial diaktifkan untuk menerima data atau mengirim data. Selain itu, SCON juga memiliki flag yang akan set (bernilai 1) apabila sebuah byte berhasil terkirim atau diterima.

SBUF (Serial Buffer, alamat 99h)

SFR SBUF digunakan untuk menampung data yang dikirim atau diterima via port serial. Nilai yang dituliskan ke SBUF akan dikirimkan ke pin TXD. Dan

juga nilai yang diterima oleh pin RXD akan dikirimkan ke program pengguna via SBUF. Dengan kata lain, SBUF berperan sebagai port output ketika data ditulis ke SBUF, dan SBUF berperan sebagai port input manakala SBUF dibaca.

P2 (Port 2, Alamat A0h, dapat dialamati secara bit)

Merupakan I/O untuk port 2. Setiap bit dari register ini berkoresponedensi terhadap pin-pin port 2 pada mikrokontroller. Contoh, bit 0 dari P2 merupakan pin 2.0, bit 7 merupakan pin P2.7. Menulis 1 ke register ini akan mengirim logik 1 ke pin I/O bersangkutan.

IE (Interrupt Enable, alamat A8h)

SFR IE digunakan untuk mengaktifkan atau menonaktifkan interupsi tertentu. Bit terendah dari IE digunakan untuk mengaktifkan/menonaktifkan interrupt tertentu, sedangkan 4 bit tertinggi digunakan untuk mengaktifkan/menonaktikan semua interrupt. Maka jika bit tertinggi pada IE bernilai 0, semua interupsi akan tidak aktif

P3 (Port 3, alamat B0h)

Merupakan I/O untuk port 3. Setiap bit dari register ini berkoresponedensi terhadap pin-pin port 3 pada mikrokontroller. Contoh, bit 0 dari P3 merupakan pin 3.0, bit 7 merupakan pin P3.7. Menulis 1 ke register ini akan mengirim logik 1 ke pin I/O bersangkutan.

IP (Interrupt Priority, alamat, alamat B8h)

SFR ini digunakan untuk menentukan prioritas dari setiap interupsi

PSW (Program Status Word, alamat D0h)

PSW digunakan untuk menyimpan sejumlah bit penting yang terset (1) atau terclear (0) oleh instruksi dari 8051. SFR PSW mengandung carry flag, auxilarry carry flag, overflow flag dan parity flag. Selain itu PSW juga

mengandung register bank select flag yang digunakan untuk memiliki register bank “R” yang aktif.

ACC / A (Accumulator, alamat E0h)

Accumulator merupakan register SFR 8051 yang sering digunakan karena banyak instruksi yang melibatkannya. Register ini dapat menampung sebuah nilai 8 bit (1 byte). Lebih dari setengah keselurahan instruksi 8051 menggunakan atau memanipulasi register ini. Contoh, untuk menambah angka 10 dengan 20, hasilnya 30 akan disimpan di register Accumulator. Setelah register ini memegang nilai, pengguna dapat mengolah lagi nilai tersebut atau menyimpannya di lokasi memori lainnnya.

B (B Register, alamat F0)

Register B hampir sama dengan register A, yaitu sama-sama dapat menampung nilai 8 bit. Register ini digunakan dalam 2 jenis instruksi, yaitu operasi perkalian (MUL) dan pembagian (DIV). Namun register B juga sering digunakan oleh programmer sebagai register penyimpan sementara.

2.1.5 Mode Pengalamatan

Mode pengalamatan adalah bagaimana cara pengalamatan terhadap sebuah lokasi memori. Pengalamatan dalam 8051 ini meliputi :

- **Pengalamatan segera (*Immediate*) :** `MOV A, #20h`
- **Pengalamatan langsung (*Direct*) :** `MOV A, 30h`
- **Pengalamatan tak langsung (*Indirect*) :** `MOV A, @R0`
- **Eksternal langsung (*External Direct*) :** `MOVX A, @DPTR`
- **Kode tak langsung (*Code Indirect*) :** `MOVC A, @A+DPTR`

2.1.5.1 Pengalamatan segera (*Immediate*)

Sesuai namanya, pengalamatan segera merupakan pengalamatan dimana sebuah nilai segera diberikan ke sebuah lokasi memori. Instruksi dengan

pengalamatan seperti ini mendikte nilai yang akan diberikan setelah lokasi memori yang ditunjuk. Contohnya adalah instruksi beikut :

```
MOV A,#20h
```

Instruksi ini merupakan pengalamatan segera karena register Accumulator akan segera menampung nilai yang mengikutinya, yaitu 20 hexa. Pengalamatan seperti sangat cepat karena nilai yang akan ditampung diikutkan dalam instruksi.

2.1.5.2 Pengalamatan langsung (*Direct*)

Pengalamatan secara langsung (*Direct Addressing*) merupakan pengalamatan dimana nilai yang akan ditampung dalam memori diperoleh secara langsung dari lokasi memori lainnya. Contohnya adalah :

```
MOV A, 30h
```

Instruksi ini akan membaca data pada internal RAM lokasi 30h dan menyimpan data tersebut pada Accumulator. Meskipun nilai yang akan ditampung tidak diikutkan dalam instruksi, pengalamatan seperti ini juga termasuk cepat, karena data diperoleh dari internal RAM 8051. Pengalamatan secara langsung (direct) juga lebih fleksibel dibandingkan dengan pengalamatan segera (immediate), karena nilai yang akan ditampung dapat bervariasi pada lokasi memori. Yang perlu diperhatikan adalah pengalamatan langsung menggunakan alamat 00h-7Fh maka merujuk ke internal RAM, sedangkan alamat 80h-FFh merujuk ke SFR.

2.1.5.3 Pengalamatan tak langsung (*Indirect Addressing*)

Pengalamatan tak langsung merupakan mode pengalamatan yang serba guna dan fleksibel. Pengalamatan ini merupakan satu-satunya cara mengakses 128 byte ekstra memori yang terdapat internal RAM 8052. Contoh pengalamatan ini sebagai berikut :

```
MOV A, @R0
```

Instruksi ini akan menyebabkan 8051 untuk menganalisa nilai dari register R0. Lalu 8051 akan menampung nilai pada internal RAM yang alamatnya ditunjuk oleh R0 ke register Accumulator. Misalkan R0 memegang nilai 40h, dan internal RAM alamat 40h menampung nilai 67h. Ketika intruksi di atas dieksekusi, pertama-tama 8051 akan mencek nilai dari R0. Karena R0 memerang nilai 40h, 8051 akan mendapatkan nilai dari internal RAM alamat 40h (yaitu 67h) dan menyimpan nilai tersebut ke register accumulator. Pada akhir instruksi tersebut, accumulator akan memegang nilai 67h. Pengalamatan secara tak langsung ini selalu merujuk ke internal RAM, tidak pernah merujuk ke SFR. Perhatikan instruksi berikut :

```
MOV R0, #99h     ;99h merupakan alamat SFR SBUF
MOV @R0, #01h    ;mengirim 1 ke SBUF untuk port serial
```

Instruksi di atas untuk 8051 tidak bisa digunakan. Karena pengalamatan tak langsung tidak pernah merujuk ke SFR. Namun instruksi ini akan memberi nilai 01h ke internal RAM alamat 99h pada mikrokontroller 8052, dimana internal RAM nya mencapai 256 byte.

2.1.5.4 Eksternal langsung (*External Direct*)

Memori eksternal diakses dengan menggunakan mode pengalamatan ini. Pengalamatan ini mirip dengan pengalamatan secara langsung, dimana sebuah lokasi memori akan menampung nilai dari data pada lokasi memori lainnya. Namun memori lainnnya ini merupakan memori eksternal.Hanya ada 2 intruksi untuk mede pengalamatan ini, yaitu :

```
MOVX A,@DPTR
MOVX @DPTR,A
```

Kedua instruksi ini menggunakan DPTR, dimana DPTR memegang alamat lokasi memori eksternal yang ingin dibaca atau ditulisi. Saat DPTR sudah memegang alamat memori eksternal yang valid, instuksi pertama di atas akan memberikan nilai yang terkandung pada alamat memori eksternal tersebut ke register accumulator. Instruksi kedua merupakan kebalikannya, yaitu akan menulis nilai pada accumulator ke alamat pada eksternal memori yang ditunjuk oleh DPTR.

2.1.5.5 Eksternal tak langsung (*External Indirect*)

Selain dengan pengalamatan eksternal langsung, eksternal memori juga dapat diakses menggunakan mode pengalamatan eksternal tak langsung. Bentuk pengalamatan seperti ini biasanya digunakan dalam aplikasi yang menggunakan RAM eksternal ukuran kecil. Contoh pengalamatan eksternal tak langsung :

```
MOVX @R0, A
```

Pertama-tama 8051 akan membaca nilai dari R0 dan nilai pada accumulator akan dituliskan pada alamat memori eksternal yang ditunjuk oleh nilai pada R0.

2.1.6 Set Instruksi dan Aliran Program

Secara keseluruhan MCS-51 mempunyai 255 macam instruksi, yang dibentuk dengan mengkombinasikan instruksi dan operan. Dalam pemrograman mikrokontroller, operan adalah data yang tersimpan di dalam memori, register dan input / output (I/O). Instruksi yang dikenal secara umum dikelompokkan menjadi beberapa kelompok, yaitu instruksi pemindahan data, aritmatika, operasi logika dan pengaturan aliran program.

2.1.6.1 Kelompok Pemindahan Data

Instruksi dasar untuk kelompok ini adalah `MOV` (Move) yang berarti memindahkan atau menyalin data. Contoh : `MOV A, R7`. Setelah instruksi ini

dieksekusi, accumulator (A) dan R7 akan berisi data yang sama, yang awalnya tersimpan dalam R7. Untuk pemakaian pada memori kode / program, instruksi MOV dituliskan menjadi MOVC. Hanya ada 2 jenis instruksi yang menggunakannnya yaitu :

```
MOVC A,@A+DPTR          ;DPTR sebagai register pointer
MOVC A,@A+PC            ;PC sebagai register pointer
```

Selain itu masih dikenal pula instruksi MOVX, yakni instruksi yang dipakai untuk mengakses memori eksternal.

2.1.6.2 Instruksi Aritmatika

Instruksi-instrksi dalam kelompok aritmatik selalu melibatkan accumulator dan beberapa juga melibatkan register lainnya. Secara garis besar operasi dalam instruksi aritmatik adalah : penjumlahan (ADD dan ADDC), pengurangan (SUBB), perkalian (MUL), pembagian (DIV) dan penaikkan / penurunan satu (INC / DEC).

2.1.6.3 Operasi Logika

Kelompok instruksi ini dipakai untuk melakukan operasi logika, yaitu operasi *AND* (instruksi ANL), operasi *OR* (instruksi ORL), operasi *exclusive-OR* / *XOR* (XRL), operasi *clear* (instruksi CLR), instruksi negasi / komplemen (instruks CPL), operasi penggeseran kanan / kiri (instruksi RR, RRC, RL dan RLC) serta operasi penukaran data (instruksi SWAP). Data yang dipakai dalam operasi ini bisa berupa data yang berada dalam *accumulator* atau data yang berada dalam RAM.

2.1.6.4 Pengaturan Aliran Program

Ketika pertama kali 8051 diinisialisasi, 8051 akan mereset PC (*Program Counter*) ke 0000h. 8051 akan memulai mengeksekusi instruksi pada memori secara *sequential* kecuali jika sebuah instruksi pada program menyebabkan PC

untuk bertindak lain. Ada beberapa instruksi yang dapat menyebabkan nilai PC termodifikasi, khusunya instruksi percabangan, pengecekan kondisi, lompat langsung (*direct jump*), pemanggilan (*call*), dan kembali (*return*) dari subrutin. Selain itu, aktifnya interrupt dapat menyebabkan program berjalan menyimpang dari aturan *sequential*.

2.1.6.4.1 Kondisi Percabangan (*Conditional Branching*)

8051 mengandung sekumpulan instruksi yang tergolong sebagai instruksi percabangan. Instruksi ini menyebabkan eksekusi program berjalan tidak secara sequential jika suatu kondisi tertentu terpenuhi. Contohnya adalah instruksi `JB`, yang berarti *Jump if Bit Set*

```
JB 45h, HELLO
NOP
HELLO: ...
```

8051 akan menganalisa bit yang terkandung pada alamat 45h. Jika bit tersebut set (1), maka program akan segera melompat ke label `HELLO`, melewatkan instruksi `NOP`. Jika bit tidak terset (0) maka kondisi percabangan intruksi tidak terpenuhi dan eksekusi berjalan sesuai biasanya, mengeksekusi instruksi berikutnya, `NOP`. Pada bahasa pemrograman level tinggi ini mirip dengan "*IF… THEN…*". Yang perlu diperhatikan adalah program hanya akan melakukan percabangan terhadap intruksi yang terletak dalam *range* 128 byte sebelum atau 127 byte sesudah instruksi percabangan. Hal itu berarti bahwa label `HELLO` harus ditulis +/- 128 byte dari lokasi memori dimana instruksi kondisi percabangan terletak. Selain itu ada juga instruksi JZ (*Jump if Zero*), JNZ (*Jump if Not Zero*), JC (*Jump on Carry*), dan JNC (*Jump on No Carry*).

2.1.6.4.2 Pengecekan Kondisi dan Pemrosesan

Instruksi-instruksi kondisi percabangan yang dibahas sebelumnya, memantau kondisi yang sudah terjadi dan tercatat di 8051. Ada dua instruksi yang

melakukan terlebih dahulu suatu proses, kemudian memantau hasil proses untuk menentukan apakah harus lompat atau tidak. Kedua instruksi tersebut adalah `DJNZ` (*Decrement and Jump if Not Zero*) dan `CJNE` (*Compare and Jump if Not Equal*).

Instruksi DJNZ merupakan instrukusi yang akan mengurangi 1 nilai register serbaguna (R0 – R7) atau RAM dan akan melompat ke alamat yang dituju jika ternyata pengurangan 1 tersebut hasilnya tidak nol. Contoh :

```
MOV R0,#23h
DJNZ R0,$
```

Instruksi MOV R0,#23h memberi nilai 23h ke R0, selanjutnya setiap kali instruksi DJNZ R0,$ dikerjakan, 8051 akan mengurangi nilai R0 dengan 1, jika R0 belum menjadi nol maka 8051 akan mengulang instruksi tersebut. Instruksi CJNE membandingkan 2 nilai yang disebutkan dan 8051 akan lompat ke alamat yang dituju jika kedua nilai tersebut tidak sama. Contoh :

```
     MOV A,P1
     CJNE A,#0Ah,TidakSama
     ...
     SJMP EXIT
TidakSama:
     NOP
     ...
```

Instruksi `MOV A,P1` membaca nilai masukan dari Port 1, instruksi `CJNE A,#0Ah,TidakSama` memeriksa apakah nilai Port 1 yang sudah disimpan di A sama dengan 0Ah, jika tidak maka lompat ke label `TidakSama`.

2.1.6.4.3 Lompat Langsung (*Direct Jump*)

Tidak seperti kondisi percabangan, direct jump akan langsung melakukan lompatan ke lokasi memori yang diberikan tanpa harus memenuhi suatu kondisi. Dalam bahasa BASIC, instruksi ini sama dengan "*GOTO*". Dalam 8051 digunakan instruksi lompat langsung dan pemanggilan (*Direct Jump and Call*) untuk melakukan hal ini, seperti `LJMP, AJMP` atau `SJMP`. Ketiga instruksi ini sebenarnya melakukan hal yang sama, yang membedakan hanyalah jangkauannya saja. Instruksi `SJMP` hanya akan melompat pada alamat dengan jamgkauan +/- 128 byte dari pengeksekusian intruksi `SJMP`. Intruksi `AJMP` hanya dapat melompat ke alamat dalam 2Kb blok memori yang sama. Jika intruksi `AJMP` terletak pada alamat 650h, maka hanya bisa melompat ke alamat 0000h-07FFh. Sedangkan `LJMP` dapat melompat ke lokasi memori manapun. Namun intruksi `LJMP` membutuhkan 3 byte kode memori, sedangkan `SJMP` dan `AJMP` hanya membutuhkan 2 byte.

2.1.6.4.4 Pemanggilan Langsung (*Direct Call*)

Operasi lainnya adalah pemanggilan langsung dengan instruksi `LCALL`, dalam bahasa BASIC mirip dengan perintah *GOSUB*. Ketika 8051 mengeksekusi intrukksi `LCALL`, maka secara langsung 8051 akan mempush PC saat itu ke dalam *stack* dan kemudian melanjutkan eksekusi kode pada alamat yang ditunjuk oleh instruksi `LCALL`.

2.1.6.4.5 Kembali dari Rutin (*return*)

Struktur lainnya yang dapat mengubah aliran program adalah intruksi Kembali dari Rutin, yaitu `RET`. Ketika instruksi `RET` dieksekusi, program akan kembali ke alamat setelah instruksi pemanggilan rutin, alamat yang sebelumnya tersimpan dalam *stack* ketika pemanggilan rutin.

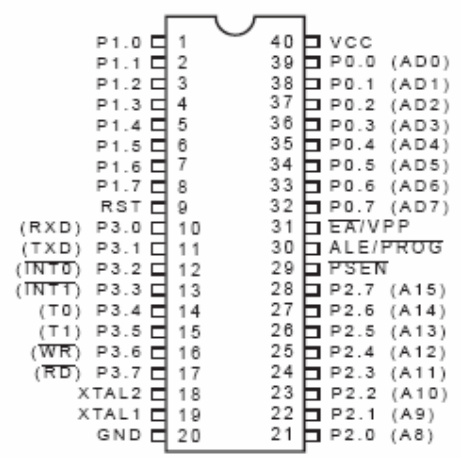
2.1.6.4.6 Interupsi

Interupt merupakan fitur spesial dari 8051. Interupt dipicu ketika adanya event, saat ini terjadi 8051 akan meninggalkan program yang sedang berjalan dan mengerjakan bagian kode lainnya yang dikenal sebagai interrupt handler. Interupt handler melakukan fungsi khusus yang dibutuhkan untuk menangani terjadinya event dan kemudian mengembalikan program yang ditinggalkannya saat interrupt handler selesai. Jelas terlihat bahwa interrupt dapat mengubah aliran program berubah.

2.2 Mikrokontroller AT89C51

2.2.1 Fitur AT89C51

AT89C51 merupakan mikrokontroller CMOS 8 bit dengan kapasitas PEROM (Progrmmable Erasable Read Only Memory) sebesar 4 Kb yang memiliki performa tinggi serta konsumsu daya yang rendah. Komponen ini dimanufaktur dengan menggunakan teknologi memori *high-density* milik atmel serta kompatibel dalam hal instruksi set dan susunan pin standar industri MCS 51. Fitur flash on-chip memberikan kemampuan memori kode / program dapat diprogram berulang kali secara ISP (In System Programming) atau melalui programmer memori nonvolatile. Dengan mengkombinasikan sebuah CPU 8-bit dan chip flash monolithic, AT89C51 merupakan mikrokontroller yang kokoh, fleksibel dan biaya terjangkau untuk solusi aplikasi kontrol emebedded. AT89C51 hadir dalam bentuk PQFP / TQFP, PDIP dan PLCC. Berikut merupakan susunan pin AT89C51 dalam kemasan PDIP :



Gambar 2.4 Konfigurasi pin AT89C51

AT89C51 memiliki fitur standar, anatara lain : 4Kb *Flash*, 128 byte RAM, 32 jalur I/O, 2 buah *timer / counter* 16-bit, 5 *vector interrupt*, *serial port full duplex, on-chip oscillator* dan rangkaian *clock*. Selain itu, AT89C51 didesain dengan *static logic* untuk operasi frekuensi sampai 0Hz dan mendukung mode operasi yang menghemat daya, yaitu *Idle Mode* dan *Power Down Mode*. Mode *idle* akan memberhentikan proses CPU (*sleep*), namun tetap memfungsikan fitur *on-chip* lainnya, seperti : RAM, *timer / counter*, *serial port* dan sistem interupsi. Pada mode *idle*, isi internal RAM tidak akan berubah. Untuk keluar dari mode *idle* dapat dilakukan melalui interupsi atau dengan *reset*. Mode *power down* akan menyimpan isi RAM dan SFR, namun menahan oskilator untuk mendisable fungsi chip lainnya sampai adanya reset. Karena oskilator berhenti maka instruksi terakhir yang dieksekusi adalah instruksi yang meng-invoke mode *power down*. Berikut adalah tabel status pin selama terjadinya mode operasi *idle* dan *power down*.

Tabel 2.2 Status pin saat mode operasi idle dan power down

Mode	Memori kode / program	ALE	PSEN	PORT0	PORT1	PORT2	PORT3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	Eksternal	1	1	*Float*	Data	Alamat	Data
Power Down	Internal	0	0	Data	Data	Data	Data
Power Down	Eksternal	0	0	*Float*	Data	Data	Data

2.2.2 Penguncian Bit Pada Memori Kode / Program

Fitur on-chip menyediakan 3 mode penguncian bit yang dapat dibiarkan tidak terprogram (U / *Unprogrammed*) atau dapat juga diprogram (P / *Programmed*) untuk mendapatkan sebuah fitur proteksi yang dideskripsikan pada tabel berikut :

Tabel 2.3 Fitur proteksi Lock Bit pada AT89C51

Program Lock Bit				Jenis Proteksi
	LB1	LB2	LB3	
1	U	U	U	Tidak ada fitur penguncia program
2	P	U	U	Instruksi MOVC yang dieksekusi dari memori program eksternal tidak berfungsi dalam memfetch byte kode dari memori internal, EA akan di*sampling* dan di*latch* pada saat *reset*, dan memprogram flash selanjutnya tidak bisa.
3	P	P	U	Sama seperti mode 2, namun *verify* juga tidak berfungsi
4	P	P	P	Sama seperti mode 3, namun eksekusi eksternal tidak berfungsi

Ketika lock bit 1 (LB1) deprogram, level logik pada pin EA akan disampling dan dilacth pada saat reset. Jika chip tidak memiliki reset, latch akan

menginisialisasi ke nilai acak dan akan memegang nilai tersebut sampai reset aktif.

2.2.3 Memprogram *On-chip Flash*

AT89C51 umumnya diproduksi dengan array memori flash on-chip pada kondisi terhapus / *erased* (berisi data FFh) dan siap untuk diprogram. *Interface programming* menerima sinyal *program enable* baik *high-voltage* (12 V) maupun *low-voltage* (Vcc). Mode pemrograman secara *low-voltage* memberikan cara yang mudah untuk memprogram AT89C51 dalam sistem pengguna, sedangkan mode *high-voltage* merupakan cara kompatibe dengan third part konvensional seperti *programmer Flash* atau EPROM. AT89C51 dipasarkan dengan mode *programming low-voltage* yang terintegrasi atau *high-voltage* saja. Tanda pada bagian atas dan kode *signature* pada IC menandakan mode *programming*nya.

Tabel 2.4 Kode signature AT89C51 untuk pemrogrmannya

	Vpp = 12 V	Vpp = 5 V
Tanda bagian atas	AT89C51 xxxx yyww	AT89C51 xxxx-5 yyww
Kode *signature*	(030H)=1Eh (031H)=5Eh (032H)=FFh	(030H)=1Eh (031H)=5Eh (032H)=05h

Memori program pada AT89C51 diprogram secara byte per byte dalam kedua mode pemrogramannya. Untuk memprogram byte pada *on-chip flash* memori yang sudah terisi, seluruh memori harus dihapus dengan menggunakan mode *Chip Erase*.

2.3 LCD HD48770

LCD (Liquid Crystal Display) memiliki standarisasi, seperti halnya MCS-51, yang merupakan *chip* kontroler bertipe Hitachi 44780 yang menjadi acuan

bagi *vendor* LCD dalam membuat modul LCD. Standar 44780 memerlukan 3 jalur kontrol ditambah 4 atau 8 jalur I/O untuk jalur data. Pengguna dapat memilih apakah LCD dioperasikan secara 4-bit *data bus* atau 8-bit *data bus*. Jika menggunakan 4-bit *data bus* maka total koneksi ke LCD ada 7 jalur data (3 untuk jalur kontrol dan 4 untuk jalur data). Jika menggunakan 8-bit *data bus* maka total koneksi ke LCD ada 11 jalur data (3 untuk jalur kontrol dan 8 untuk jalur data). Tiga jalur kontrol ke LCD adalah EN (*Enable*), RS (*Register Select*) dan R/W (*Read/Write*). Berikut ini adalah susunan umum pin LCD 44780 :

Tabel 2.5 Deskripsi pin (14) LCD

Pin	**Deskripsi**
1	*Ground*
2	Vcc
3	Pengatur kontras
4	"RS" Instruction/Register Select
5	"R/W" Read/Write LCD Registers
6	"EN" Enable clock
7 - 14	Data I/O Pins

Sebagaimana terlihat pada kolom deskripsi, *interface* LCD merupakan sebuah *parallel bus*, dimana hal ini sangat memudahkan dan sangat cepat dalam membaca / menulis data dari / ke LCD. Kode ASCII yang ditampilkan sepanjang 8 bit dikirim ke LCD secara 4 atau 8 bit pada satu waktu. Jika mode 4 bit yang digunakan, maka 2 nibble data dikirim untuk membuat sepenuhnya 8 bit (pertama dikirim 4 bit MSB lalu 4 bit LSB dengan pulsa clock “EN” setiap nibblenya).

Jalur kontrol EN digunakan untuk memberitahu LCD bahwa mikrokontroller mengirimkan data ke LCD. Untuk mengirim data ke LCD program harus menset EN ke kondisi *high* (1) dan kemudian menset dua jalur kontrol lainnya (RS dan R/W) atau juga mengirimkan data ke jalur *data bus*. Saat jalur lainnya sudah siap, EN harus diset ke 0 dan tunggu beberapa saat (tergantung

pada *datasheet* LCD), dan set EN kembali ke high (1). Ketika jalur RS berada dalam kondisi *low* (0), data yang dikirimkan ke LCD dianggap sebagai sebuah perintah atau instruksi khusus (seperti bersihkan layer, posisi kursor dll). Ketika RS dalam kondisi *high* atau 1, data yang dikirimkan adalah data ASCII yang akan ditampilkan dilayar. Misal, untuk menampilkan huruf “A” pada layar maka RS harus diset ke 1. Jalur kontrol R/W harus berada dalam kondisi *low* (0) saat informasi pada *data bus* akan dituliskan ke LCD. Apabila R/W berada dalam kondisi *high* (1), maka program akan melakukan query (pembacaan) data dari LCD. Instruksi pembacaan hanya satu, yaitu *Get LCD status* (membaca status LCD), lainnya merupakan instruksi penulisan. Jadi hampir setiap aplikasi yang menggunakan LCD, R/W selalu diset ke 0. Jalur data (*data bus*) dapat terdiri 4 atau 8 jalur (tergantung mode yang dipilih pengguna), merrka dinamakan DB0, DB1, DB2, DB3, DB4, DB5, DB6 dan DB7.

Mengirim data secara parallel baik 4 atau 8 bit merupakan 2 mode operasi primer. Untuk membuat sebuah aplikasi interface LCD, menentukan mode operasi merupakan hal yang paling penting. Mode 8 bit sangat baik digunakan ketika kecepatan menjadi keutamaan dalam sebuah aplikasi dan setidaknya minimal tersedia 11 pin I/O (3 pin untuk kontrol, 8 pin untuk data). Sedangkan mode 4 bit minimal hanya membutuhkan 7 bit (3 pin untuk kontrol, 4 untuk data).

Bit “RS” digunakan untuk memilih apakah data atau instruksi yang akan ditransfer antara mikrokontroller dan LCD. Jika bit ini di set (RS = 1), maka byte pada posisi kursor LCD saat itu dapat dibaca atau ditulis. Jika bit ini di reset (RS = 0), bisa merupakan instruksi yang dikirim ke LCD atau status eksekusi dari instruksi terakhir yang dibaca.

Macam-macam instruksi yang tersedia untuk standar LCD 44780 ditunjukkan pada tabel dibawah :

Tabel 2.6 Instruksi LCD HD44780

R/S	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Instruksi/Deskripsi
4	5	14	13	12	11	10	9	8	7	Pin
0	0	0	0	0	0	0	0	0	1	Bersihkan tampilan
0	0	0	0	0	0	0	0	1	*	Kursor kembali posisi awal
0	0	0	0	0	0	0	1	ID	S	Pengaturan arah pergerakan kursor
0	0	0	0	0	0	1	D	C	B	Pengaktifan Layar / Kursor
0	0	0	0	0	1	SC	RL	*	*	Gerak kursor / geser layar
0	0	0	0	1	DL	N	F	*	*	Pengesetan panjang *interface*
0	0	0	1	A	A	A	A	A	A	Pindahkan kursor ke dalam CGRAM
0	0	1	A	A	A	A	A	A	A	posisi kursor / alamat karakter
0	1	BF	*	*	*	*	*	*	*	Status *Busy Flag*
1	0	D	D	D	D	D	D	D	D	Menulis karakter pada posisi kursor saat itu
1	1	D	D	D	D	D	D	D	D	Membaca karakter dari layar pada posisi kursor saat itu

keterangan :

- * berarti *don't care*, bisa 1 atau 0
- Pengaturan arah pergerakan kursor :
    - ID – atau *Increment Direction*, berarati cursor naik satu setiap akhir penulisan byte ke layar jika bit ini diset
    - S – atau *Shift*, berarti geser layar ketika byte ditulis ke layar
- Pengaktifan Layar / Kursor
    - D – (*Display*) menyalakan layar atau on jika diset 1. Mematikan layar jika 0

    - C – (*Cursor*) Mengaktifkan kursor jika diset 1, jika 0 kursor tidak aktif
    - B atau *Blink*, jika diset 1 maka cursor kelap-kelip
- Gerak kursor / geser layar
    - SC Aktifkan geser layar jika 1
    - RL, Arah geser, 1 ke kanan, 0 ke kiri
- Pengesetan panjang *interface*
    - DL (Data Length), 8 bit jika diset 1, 4 bit jika diset 0
    - N – Jumlah baris pada layar. Jika 1 baris maka diset 0, jika 2 baris maka diset 1
    - F atau Font, bit ini untuk menset ukuran karakter Font. Jika diset 1 maka ukuran font 5x10, jika diset 0 maka 5x7
- Busy Flag
    - BF – bit ini akan bernilai 1 ketika LCD sedang memproses

A – Address / Alamat

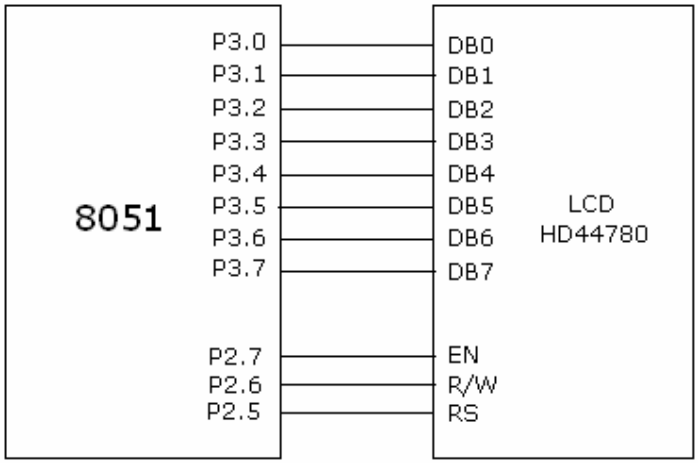
- Baca / tulis ASCII ke layar
- Gerak kursor ke CGRAM / layar

Data – Data

Kursor untuk 44780 dapat diaktifkan dengan bentuk *underscore* dengan instruksi “Pengaktifan Layar / Kursor” dan menset bit C. Pada operasi normal, setelah karakater dikirim ke LCD, cursor akan bergerak satu karakter ke kanan. Instruksi "Bersihkan tampilan" dan "Kursor kembali posisi awal" digunakan untuk mereset posisi kursor pada pojok kanan atas layar. Untuk menggerakkan kursor , instruksi "Posisi Kursor / Alamat Karakter" dapat digunakan. Untuk instruksi ini, bit ke-7 dari byte instruksi harus diset dan 7 bit sisanya digunakan sebagai alamat karakter / posisi pada LCD. 7 bit ini menyediakan 128 alamat yang merupakan jumlah maksimal alamat karakter LCD yang tersedia.

Untuk mode 8-bit dibutuhkan 8 - 11 jalur data untuk berkomunikasi dengan LCD. Misalkan jalur data akan dihubungkan ke port 3 (P3.0 – P3.7) dan

jalur kontrol dihubungkan ke P2.5, P2.6 dan P2.7. Gambar berikut menunjukkan skematik koneksi LCD mode 8-bit ke mikrokontroller 8051.

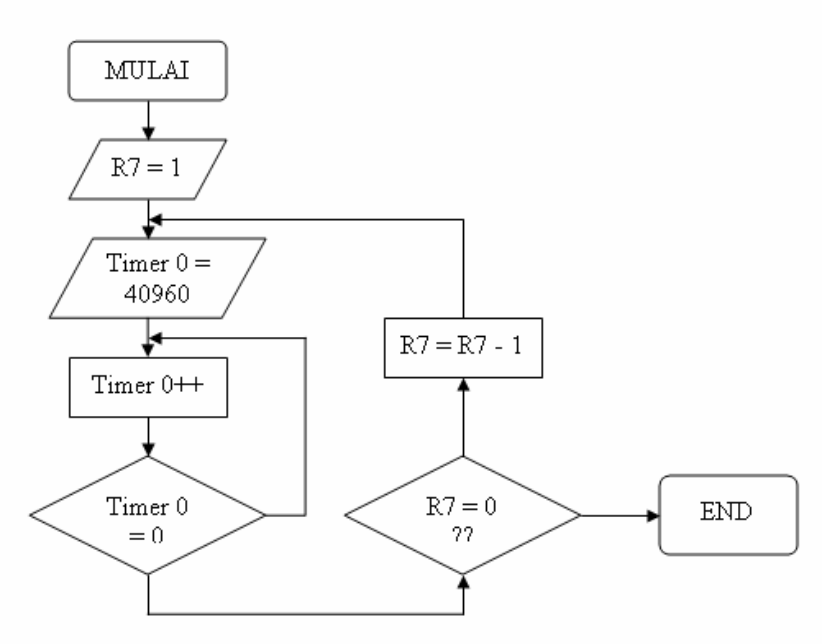


Gambar 2.5 Skema koneksi LCD

Dari skema dapat dilihat koneksi dari 8051 ke LCD merupakan jalur satu sama lain yang terhubung antara pin. Dalam penulisan program assembler dapat dideklarasikan konstanta untuk memudahkan. Misalkan :

```
DB0  EQU  P3.0
DB1  EQU  P3.1
DB2  EQU  P3.2
DB3  EQU  P3.3
DB4  EQU  P3.4
DB5  EQU  P3.5
DB6  EQU  P3.6
DB7  EQU  P3.7
EN   EQU  P2.7
RW   EQU  P2.6
RS   EQU  P2.5
DATA EQU  P1
```

Dengan menggunakan konstanta di atas, pengguna dapat merujuk jalur I/O 8051 sesuai dengan nama pin pada LCD 44780. Contoh, untuk menset RW ke 1, maka cukup dengan instruksi : SETB RW. Sebagaimana telah dijelaskan, bahwa jalur EN digunakan untuk memberitahu LCD bahwa instruksi yang dikirim lewat jalur kontrol dan jalur data siap dieksekusi. Jalur EN harus berada dalam transisi 0-1/1-0 sebelum/setelah instruksi atau data dikirim ke LCD. Singkat kata, pengguna harus memanipulasi EN ketika berkomunikasi dengan LCD. Sebelum berinteraksi dengan LCD, pengguna perlu menset jalur EN ke 1 : SETB EN. Setelah selesai menyiapkan instruksi pada jalur kontrol dan jalur data, EN harus berada dalam kondisi low (0). Jalur EN harus berada dalam kondisi low untuk beberapa saat (tergantung waktu yang tercantum pada datasheet). Penulis menggunakan LCD dengan maksimal waktu pengeksekusian intruksi ±50ms dan ±20ms. Untuk itu dibutuhkan rutin penundaan hingga LCD selesai mengeksekusi instruksi yang dikirim. Untuk menghasilkan delay ±50ms dapat digunakan timer dengan mode 16 bit. Diagram alur serta rutin program ini akan menjelaskan bagaimana timer menghitung hingga ±50ms :



Gambar 2.6 Diagram alur timer 0 mode 16 bit mencacah selama ±50ms

Dalam 8051, timer dengan mode 16 bit dapat menjangkau nilai sampai 65535. Timer selalu menhitung naik ke atas dan akan mengalami overflow bila mencapai nilai maksimalnya. Apabila timer mengalami overflow, maka timer akan kembali lagi ke 0. Bit register pencacat overflow adalah TFx Timer mode 16 bit adalah mode umum yang sering digunakan karena mudah dan sesuai untuk penggunaan delay. Penaikkan 1 pada timer memakan waktu 1 µs / 1 siklus mesin. Untuk menghitung banyaknya bilanganan (desimal) yang dicacah oleh timer dapat digunakan formula berikut :

TH0 * 256 + TL0

TH0 dan TL0 merupakan bit register pada SFR TMOD yang menunjukkan pengaturan pada Timer 0. Untuk mencacah sampai sampai 1000 maka register TH0 dan TL0 perlu disikan nilai berikut :

TH0 * 256 + TL0 = 1000
3 * 256 + 232 = 1000
TH0 = 3 dan TL0 = 232

Sedangkan untuk menghitung waktu sampai ± 50µs dapat digunakan rutin di bawah ini (perhatikan komentar program setelah tanda titik-koma).

```
DELAY1:
PUSH 7
MOV R7,#1
SUB_DELAY1Z:
MOV TMOD,#00000001B      ; Timer 0 mode 16 bit
MOV TH0,#0A0H            ; 160 * 256 + 0 = 40960
MOV TL0,#00H             ; 65536 - 40960 = 24576us
                         ; = 24,5 ms
SETB TR0                 ; Timer 0 jalan
TF0??:
```

```
JNB TF0,TF0??            ; overflow gak?
CLR TR0                  ; matikan timer 0
CLR TF0                  ; reset flag overflow
                         ; 24.5 x 2 = 49 ms
                         ; +/- 50 ms
DJNZ R7,SUB_DELAY1Z      ; r7 = 0 gak?
POP 7
RET
```

Sebelum menggunakan LCD, pengguna harus menginisialisasi dan mengkonfigurasi LCD tersebut, yaitu dengan mengirimkan sejumlah instruksi inisialisasi ke LCD. Instruksi pertama yang harus dikirim adalah menset jalur data LCD, apakah mode 8-bit atau mode 4-bit. Selain itu, ukuran font karakter harus diset ke 5x7 dot. Dua instruksi ini dapat dilakukan dengan mengirimkan data 38h sebagai suatu perintah. Oleh karenanya RS harus berada dalam kondisi low (0). Instruksi berikut dapat digunakan untuk menginsialisasi 2 instruksi tersebut :

```
INITIALIZE:
SETB LCD_EN
CLR LCD_RW
CLR LCD_RS
MOV DATA,#00110000B  ;30h
CLR LCD_EN
LCALL DELAY1
SETB LCD_EN
```

Instruksi selanjutnya yang perlu dilakukan adalah mengaktifkan layar, yaitu dengan memberikan data 0Ch.

```
CLR LCD_RW
CLR LCD_RS
```

```
MOV DATA,#00001100B  ;0Ch
CLR LCD_EN
LCALL DELAY1
SETB LCD_EN
```

Agar arah kursor bergeser satu setelah penulisan byte pada LCD, maka perlu diinisialisasi Increment Direction nya.

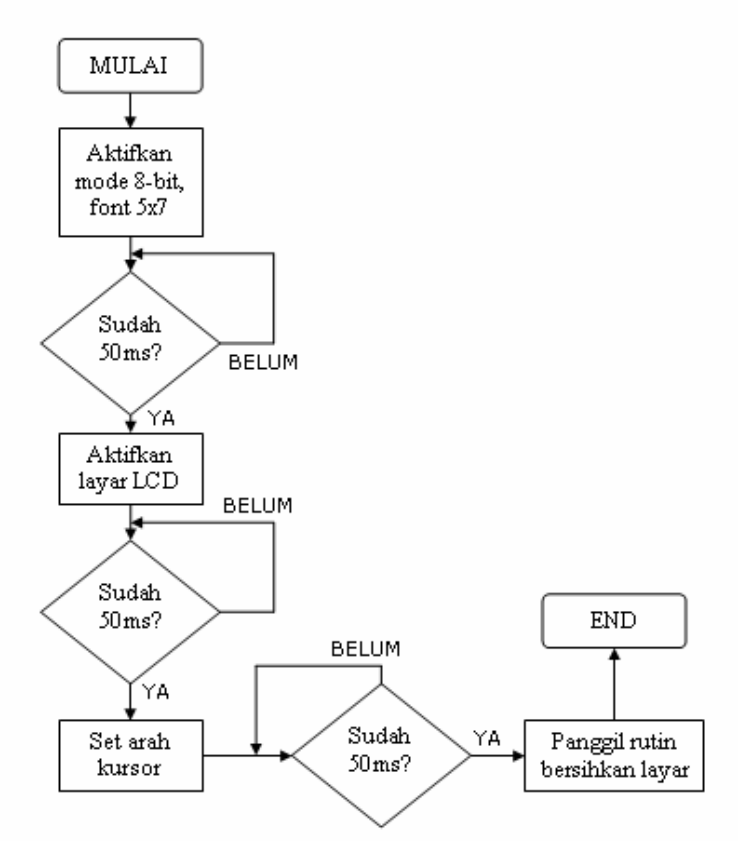
```
CLR LCD_RW
CLR LCD_RS
MOV DATA,#00000110B
CLR LCD_EN
LCALL DELAY1
RET
```

Semua instruksi di atas dapat digunakan sebagai rutin insialisasi ke LCD. Ketika LCD pertama kali diinisialisasi, layar LCD secara otomatis akan kosong. Namun untuk memastikan agar tampilan yang dikirim ke LCD sesuai harapan, tidak ada salahnya untuk membersihkan layar terlebih dahulu sebagai operasi pertama setelah inisialisasi. Instruksi untuk membersihkan layar adalah dengan memberikan data 01h. Rutin berikut dapat digunakan untuk membersihkan layar LCD.

```
CLEAR_SCREEN:
SETB LCD_EN
CLR LCD_RS
CLR LCD_RW
MOV DATA,#00000001B
LCALL DELAY0
CLR LCD_EN
LCALL DELAY0
```

```
RET
```

Inisialisasi LCD dan permbersihan layar merupakan operasi pertama yang diberikan ke LCD. Secara kseluruhan alur operasi tersebut sebagai berikut :



Gambar 2.7 Diagram alur operasi pertama pada LCD

Setelah melakukan inisialisasi dan pembersihan layar, LCD dapat dikirimi byte ASCII untuk menampilkan karakter yang diinginkan. Penulisan per karakter merupakan hal yang akan terus berulang dilakukan. Untuk itu membuat rutin penulisan karakter akan mengefisiensi panjang program. Rutin berikut dapat digunakan untuk menulis karakter ke LCD.

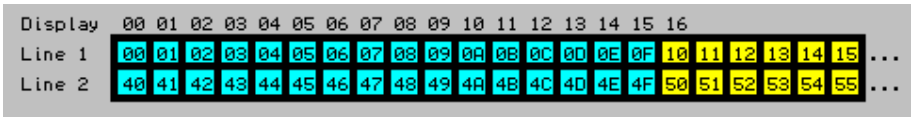
```
WRITE_ON:
MOV DATA,A
SETB LCD_EN
CLR LCD_RW
SETB LCD_RS
LCALL DELAY0
CLR LCD_EN
LCALL DELAY0
LCALL DELAY0
RET
```

Untuk menampilkan karakter ASCII ke layar LCD, pada program ASCII karakter harus di masukkan ke accumulator kemudian panggil rutin WRITE_ON. Contoh, untuk menuliskan text “TEST” ke layar LCD :

```
LCALL INITIALIZE
LCALL CLEAR_SCREEN
MOV A,#’T’
LCALL WRITE_ON
MOV A,#’E’
LCALL WRITE_ON
MOV A,#’S’
LCALL WRITE_ON
MOV A,#’T’
LCALL WRITE_ON
```

Kata “TEST” akan tercetak di pojok kiri atas layar. Namun untuk menampilkan karakter pada posisi yang diinginkan, program perlu ditambahkan. Standar 44780 mengandung sejumlah memori yang berhubungan dengan layar. Semua karakter yang ditulis ke LCD disimpan dalam memori ini, dan 44780

membaca memori ini untuk menampilkan karakter itu. Memori ini digambarkan dalam peta memori berikut :



Gambar 2.8 Alamat karakter LCD 44780

Untuk LCD 2x16, maka daerah yang berwarna biru merupakan alamat memori yang menunjukkan posisi pada layar. Karakter pertama berada pada alamat 00h, karakter ke-2 01h, dan seterusnya hingga karakter ke-16 yang berada pada alamat 0Fh. Untuk baris ke-2, karakter pertama berada pada alamat 40h hingga karakter ke-16 pada alamat 4Fh. Apabila kursor sudah mencapai alamat 0Fh, karakter ke-16 baris ke-1, maka penulisan berikutnya tidak akan menunjuk ke posisi karakter pertama baris ke-2. Karena setelah alamat 0Fh adalah 10h, sedangkan karakter pertama baris ke-2 beralamat 40h. Untuk mencetak karakter pada posisi yang diinginkan, LCD perlu dikirimi perintah dahulu, yaitu dengan instruksi Set Cursor Position, 80h + [alamat karakter]. Pengguna dapat membuat rutin yang memberikan instruksi ke LCD untuk memposisikan kursor ke alamat karakter yang diinginkan. Rutin berikut ini dapat digunakan :

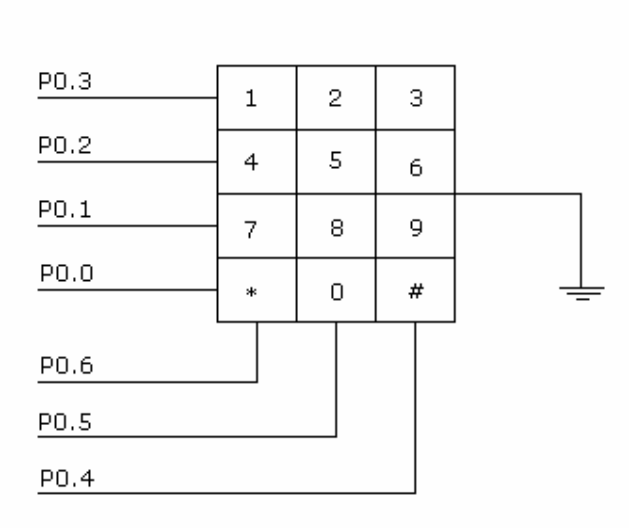
```
ADDRESS:
SETB LCD_EN
CLR LCD_RW
CLR LCD_RS
MOV DATA,A
LCALL DELAY0
CLR LCD_EN
LCALL DELAY0
RET
```

Untuk menulis kata “TEST” dimulai dari alamat 45h, baris ke-2, maka penulisan program menjadi :

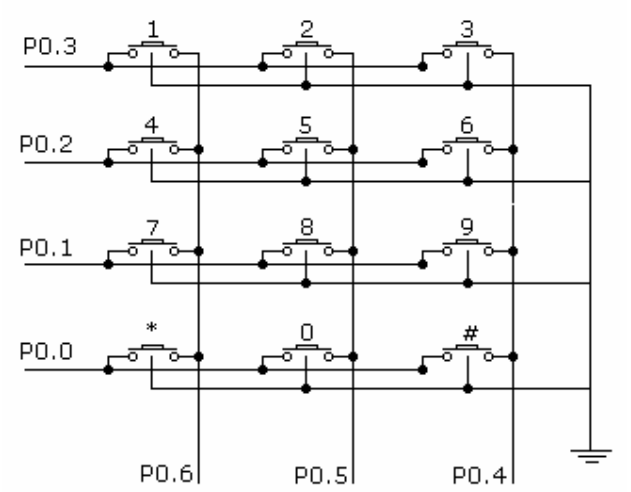
```
LCALL INITIALIZE
LCALL CLEAR_SCREEN
MOV A,#0C5h          ;80h + 45h
LCALL ADRESS
MOV A,#’T’
LCALL WRITE_ON
MOV A,#’E’
LCALL WRITE_ON
MOV A,#’S’
LCALL WRITE_ON
MOV A,#’T’
LCALL WRITE_ON
```

2.4 Scanning Keypad

Teknik *scanning keypad* merupakan sebuah teknik yang meminimalisasi jumlah pemakaian port untuk tombol *keypad* yang banyak. Umumnya *keypad* yang digunakan merupakan *keypad matrix* 4x3, 4x4 atau 4x5. Penulis menggunakan *keypad matrix* 4x3, dimana ada 12 tombol (4 baris dan 3 kolom). Tentunya menggunakan 12 port hanya untuk pemakaian pertombol sangat tidak efisien. Dengan teknik *scanning keypad* cukup digunakan 6 port I/O pada mikrokontroller. Gambar 2.9 dan 2.10 akan menunjukkan bagaimana keypad terhubung ke mikrokontroller beserta rangkaiannya.



Gambar 2.9 Susunan keypad matrix 4x3



Gambar 2.10 Rangkaian keypad matrix 4x3

Dari rangkaian keypad pada gambar 2.10 di atas, dapat diketahui bahwa bila tidak ada penekanan tombol pada keypad matrix maka kondisi pada P0.0 sampai P0.6 adalah high atau 1. Jika terjadi penekanan pada tiap tombol pada keypad, pin port 0 pada baris dan kolom akan terhubung ke ground sehingga baris dan kolom yang terhubung akan berlogika low atau 0. Untuk membuat program assembler, pengambilan data melalui keypad pada prinsipnya dilakukan dengan cara membandingkan data awal sebelum terjadi penekanan keypad dengan data

setelah terjadi penekanan keypad. Dari rangkaian keypad di atas, data yang dibandingkan adalah data yang terletak pada alamat port 0 mikrokontroller. Berikut adalah tabel konversi dari tiap penekanan tombol pada keypad.

Tabel 2.7 Konversi hexa tombol keypad

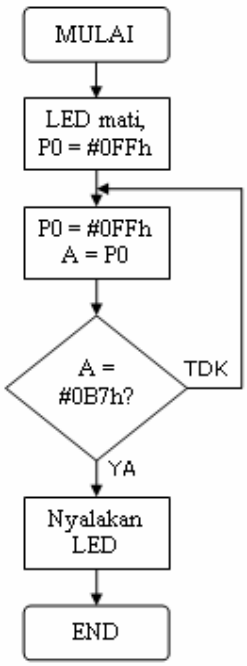
Tombol	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0	Hexa
1	1	0	1	1	0	1	1	1	B7
2	1	1	0	1	0	1	1	1	D7
3	1	1	1	0	0	1	1	1	E7
4	1	0	1	1	1	0	1	1	BB
5	1	1	0	1	1	0	1	1	DB
6	1	1	1	0	1	0	1	1	EB
7	1	0	1	1	1	1	0	1	BD
8	1	1	0	1	1	1	0	1	DD
9	1	1	1	0	1	1	0	1	ED
*	1	0	1	1	1	1	1	0	BE
0	1	1	0	1	1	1	1	0	DE
#	1	1	1	0	1	1	1	0	EE

P0.7 akan selalu berada dalam kondisi high. Bila digunakan rangkaian keypad pada gambar xx, maka tabel konversi di atas dapat digunakan. Misalkan P2.0 terhubung seri antara LED dan Vcc. Dengan memberi logik 0 pada P2.0 maka LED akan menyala. Penekanan tombol 1 pada keypad akan menyebabkan LED menyala, sementara penekanan tombol lainnya tidak akan berpengaruh, berikut adalah programnya :

```
ORG 00h
MOV P0, #0FFh  ;Jalur keypad dalam kondisi high
SETB P2.0      ;LED mati
AmbilData:
MOV P0, #0FFh
```

```
MOV A, P0      ;baca data P0, masukkan ke A
CJNE A, #0B7h,AmbilData  ;Bila    tidak    terjadi
                         ;penekanan ulang kembali
                         ;ke label AmbilData
CLR P2.0  ;Nyalakan LED
END
```

Tiga baris awal hanya untuk memastikan bahwa LED mati dan jalur keypad berada dalam kondisi awal / high yang berarti belum terjadi penekanan. Baris setelah label AmbilData merupakan perintah untuk membaca data pada port 0 dan membandingkannya dengan nilai heksa B7 (tombol ‘1’). Bila tidak sama kembali ulangi ke label AmbilData. Apabila tombol 1 ditekan maka baris akan mengeksekusi baris ke-2 dari terakhir, CLR P2.0, yang akan menyalakan LED. Berikut ini diagram alur bagaimana program berkerja :



Gambar 2.11Diagram alur contoh program sederhana keypad