BAB II

TINJAUAN PUSTAKA

2.1. Sistem Kontrol Otomatis

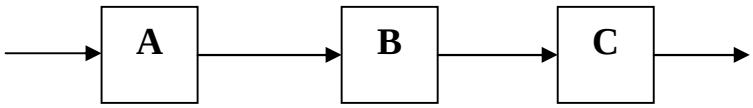
Kontrol otomatis memegang peranan sangat penting dalam perkembangan ilmu dan teknologi. Di samping sangat diperlukan pada pesawat ruang angkasa, peluru kendali, sistem pengemudian pesawat, dan sebagainya, kontrol otomatis telah menjadi bagian yang peting dan terpadu dari proses-proses pabrik dan industri modern. Sebagai contoh, kontrol otomatis sangat diperlukan dalam operasi-operasi di industri untuk mengontrol tekanan, temperature, viskositas, dan aliran dalam industri proses; pengerjaan dengan mesin perkakas, penanganan, dan perakitan bagian-bagian mekanik dalam industi manufaktur, dan sebagainya.

Dalam proses industri sering dibutuhkan besaran-besaran yang memerlukan kondisi atau persyaratan yang khusus seperti ketelitian yang tinggi, harga yang konstan untuk selang waktu tertentu, harga yang bervariasi dalam suatu rangkuman tertentu, perbandingan yang tetap antara dua varibel/besaran atau suatu besaran sebagai fungsi dari pada besaran lainnya.

Jelas semuanya ini tidak cukup dilakukan hanya dengan pengukuran saja, tetapi juga memerlukan suatu cara pengontrol agar syarat-syarat tersebut dapat dipenuhi karena inilah diperkenalkan suatu konsep pengontrolan yang disebut *sistem pengontrolan, teknik pengaturan, atau sistem kendali*. Istilah yang popular adalah *Pengontrolan secara otomatis*. Instrumentasi dan pengontrolan merupakan bidang ilmu yang saling menunjang.

2.2. Elemen Sistem kontrol

Setiap proses kontrol terdiri dari unit yang membentuknya yang disebut elemen sistem dan selanjutnya elemen ini terdiri dari komponen-komponen yang ditunjukkan pada blok diagram di bawah ini



Gambar 2.1 Elemen-elemen sistem kontrol

Keterangan :

1. kotak A adalah *masukan (inputan)*
2. Kotak B adalah *pengontrol*
3. Kotak C adalah *sistem (proses)*

2.3. Jenis Komponen Sistem Kontrol

Sesuai dengan fungsi pengontrolan secara menyeluruh, maka komponen sistem pengaturan dapat dibagi dalam 4 (empat) kelompok yaitu ;

a. Sensor

   Sensor digunakan sebagai elemen yang langsung mengadakan kontak dengan yang diukur.

b. Error detektor

   Mengukur kesalahan yang terjadi antara keluaran actual dan keluaran yang diinginkan

c. Penggerak

Alat ini berfungsi untuk aliran energi ke sistem yang di kontrol. Alat ini disebut juga *elemen pengontrol akhir (final control elemen)*, misalnya : motor listrik, Katup pengontrol, pompa, silinder hidrolik, dan lain-lain.

d. Penguat (Amplifier)

Unit ini dibutuhkan karena daya dari *“ error detector “* tidak cukup kuat untuk menggerakkan elemen keluaran. Karena fungsi pengontrolan adalah untuk mengendalikan keluaran agar kasalahan mendekati nol, maka diperlukan penguat daya (power amplifier)

2.4. Mikrokontroler

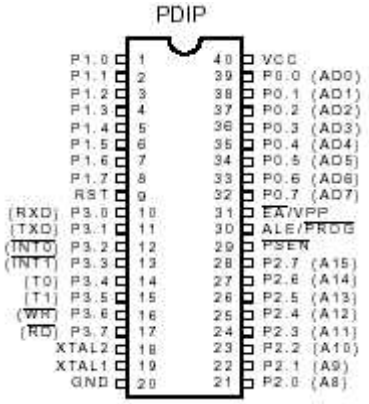
Suatu sistem dapat dikatakan sebagai mikrokontroler jika di dalamnya telah tersedia tiga bagian utama yang mendukung sebuah komputer, yaitu *central processing unit* (CPU), memori (RAM atau ROM) dan *pheriperal input* dan *output* (I/O). Mikrokontroler merupakan suatu *divais* elektronik yang termasuk general *purepose device* yang cocok untuk banyak aplikasi.

Mikrokontroler adalah suatu komputer yang dibangun pada satu *chip. Input* dan *output*nya sera sistem memori dibangun di dalam mikrokontroler, maka memungkinkan *divais* ini untuk *diinterfacekan* dengan *hardware.*

Dan fungsi kontrol. Beberapa bagian dalam mikrokontroler adalah: parallel port, serial port, *timer, conter, interup kontrol,* RAM, ROM, dan bahkan ada yang mempunyai ADC dan DAC.

2.5. Aksitektur AT89C51

Mikrokontroler AT89C51 adalah keluarga dari MSC-51 dari ATMEL Inc., yang paling populer untuk mikroprosesor kelas 8 bit. Divais ini dikemas dalam bentuk QUAD dan DIP. Mempunyai 40 pin yang terdiri dari 4 buah port 8-bit yang sangant fleksibel penggunaannya. Pada gambar di bawah ini diperlihatkan konfigurasi pin-pin dari mikrokontroler AT89C51. Fungsi pin pada mikrokontroler AT89C51 dapat dikelompokan menjadi pin sumber tegangan, pin kristal, pin kontrol, pin *input/output*, dan pin *interupsi*.



Gambar.2.2. Pin AT89C51[1]

Spesifikasi teknik dari mikrokontroler AT89C51 adalah sebagai berikut:

1. Termasuk dalam keluarga mikrokontroler MCS-51.
2. Memiliki 4 kilobyte *flash memory* yang dapat diprogram ulang sebanyak 1000 kali.

[1] " Belajar Mikrokotroler AT89C51", Gava Media, Hal 84

3. Frekuensi *clock* maksimal 24 MHz.
4. Memiliki 128 x 8-bit internal RAM.
5. Memiliki 32 jalur I / O yang dapat diprogram.
6. Memiliki 16-bit *timer/ counter*.
7. Memiliki 6 sumber *interrupt*.
8. Memiliki port serial yang dapat diprogram.
9. Terdapat mode *Low Power Idle* dan *Down Power.*

2.6. VCC

Tegangan *supply*, dihubungkan dengan tegangan 5 volt

2.7. GND

Ground dihubungkan dengan Vss atau *ground*.

2.8. RST

Masukan *reset*. Kondisi logika ‘1’ selama 2 siklus mesin saat osilator bekerja akan me*reset* mikrokontroler yang bersangkutan.

2.9. ALE/ $\overline{PROG}$

Keluaran ALE atau *Adreess Latch Enable* menghasilkan pulsa-pulsa untuk mengancing *byte* rendah (*low byte*) alamat selama pengaksesan memori *ekstenal*. Kaki ini juga berfungsi sebagai masukan pulsa program (*the program pulsa input*) atau selama pengisian *flash* PEROM. Pada operasi normal, ALE akan berpulsa

dengan laju 1/6 dari frekuensi kristal dan dapat digunakan sebagai pewaktuan *(timing*) atau pendetakan (*clocking*) rangkaian *eksternal*. Catatan, ada satu pulsa yang dilompati selama pengaksesan memori data *eksternal*. Jika dikehendaki, operasi ALE bisa dimatikan dengan cara mengatur bit 0 dari SFR lokasi 8Eh. Jika isinya '1', ALE hanya akan aktif selama dijumpai instruksi MOVX atau MOVC. Selain itu, kaki ini akan secara lemah di *pulled high*. Mematikan bit ALE tidak akan ada efeknya jika mikrokontroler mengeksekusi program secara *eksternal.*

2.10. Port I/O

Port I/O pada mikrokontroler AT89C51 ada empat buah, dimana setiap port terdiri dari delapan bit. Masing-masing bisa di akses sendiri-sediri maupun secara bersama-sama. Pada port 3 ada penambahan fungsi khusus yaitu untuk *interrupt*, *timer*, serial dan $\overline{RD}$ serta $\overline{WR}$ untuk akses memori *eksternal.*

2.10.1. Port 0

Port 0 merupakan port parallel 8 bit *open drain* dua arah. Bila digunakan untuk mengakses memori luar, port ini akan memultipleks alamat memori dengan data. Port 0 juga memerlukan kode byte pada saat pemrograman *flash*, dan kode byte *output* selama pembuktian program.

2.10.2. Port 1

Port 1 adalah sebuah port *input output* 8 bit *bidirectional* dengan *pull-up internal*. *Output buffer* port 1 dapat menjadi sumber dari 4 TTL input. Ketika '1'

ditulis di pin-pin port 1, mereka akan di *pull high* oleh *pull up* internal dan dapat digunakan sebagai input. Sebagai *input*, pin port 1 *dipull* rendah secara eksternal. Port 1 juga menerima perintah alamat byte rendah selama pemrograman *flash* dan pembuktian.

2.10.3. Port 2

Port 2 adalah sebuah port *input output* 8 bit *directiona*l dengan *internal pull up.* Buffer output port 2 dapat berfungsi sebagai sumber 4 *input* TTL. Ketika ‘1’ dituliskan di pin port 2 mereka di *pull high* oleh *internal pull up* dan dapat digunakan sebagiai *input*.

Port 2 mengeluarkan perintah alamat byte selam pengambilan dari program memori *eksternal* dan selama pemasukan ke data memori *eksternal* yang menggunakan pengalamatan 16 bit (MOVX @DPTR). Dalam aplikasi ini menggunakan *internal pull up* yang kuat ketika menggunakan ‘1’. Port 2 juga menerima perintah alamat bit tinggi dan beberapa sinyal kontrol selama pemrograman *flash* dan pembuktian.

2.10.4. Port 3

Port 3 merupakan port I/O dua arah dengan dilengkapi *pull up internal*. Penyangga keluaran port 3 mampu memberikan / menyerap arus empat masukan TTL (sekitar 1.6 mA).

Jika ‘1’ dituliskan ke kaki-kaki port 3, maka masing-masing kaki akan *dipulled high* dengan *pull up internal* sehinggah dapat digunakan sebagai

masukan. Sebagai masukan, jika kaki-kaki port 3 dihubungtkan ke *ground* (di *pulled low*), maka masing-masing kaki akan memberikan arus (*source*) karena di *pulled high* secara *internal.*

Port 3 juga memiliki fungsi-fungsi alternaif sebagaimana ditunjukan pada tabel3.1,antara lain menerima sinyal-sinyal kontrol selama pengisian program dan *verifikasi falsh.*

Tabel 3.1. Fungsi-fungsi khusus pada AT89C51

Kaki Port	Funsi alternatif	Keterangan
P3.0	RXD	Saluran masukan serial
P3.1	TXD	Saluran keluaran serial
P3.2	$\overline{INT0}$	*Interupsi eksternal* 0
P3.3	$\overline{INT1}$	*Interupsi eksternal* 1
P3.4	T0	Masukan *eksterna*l pewaktu/pencacah 0
P3.5	T1	Masukan *eksterna*l pewaktu/pencacah 1
P3.6	$\overline{WR}$	Sinyal tanda baca memori data *eksternal*
P3.7	$\overline{RD}$	Sinyal tanda tulis memori data eksternal

Sumber: Belajar Mikrokontroler AT89C51, hal 87

2.11. Interupt

Terdiri dari sepasang *interupt eksternal* yaitu INT0 dan INT1. Untuk mengaktifkan kedua *interupt* ini maka pada *register* IE (alamat 0A8H) diberi angka aktif 1 pada bit pertama dan yang ketiga. Prioritas dari *interup*t ini diatur

oleh register IP (alamat 0B8H), juga pada bit pertama dan ketiga. Untuk mematikan kerja *interupt* sementara maka *register* EA diberi nilai 0 atau dengan program clr EA.

2.12. Timer

Selain *interupt eksternal* port kontrol ini terdapat sepasang *interupt timer* T0 dan T1. untuk mngaktifkan kedua *timer* harus melalui beberapa prosedur dahulu. Pertama adalah penentuan jenis *timer* dengan register TMOD. Register ini tidak dapat di akses bit per bit, biasanya seting dilakukan pada awal inisialisasi program. Setelah seting TMOD maka *timer* siap untuk dijalankan, dihentikan atau lain-lainnya dengan mengakses register *timer* khusus yaitu SFR (*special function Registers*).

Timer ada 3 mode yaitu mode 0 (13 bit *timer*), dimana 5 bit dihandle oleh register TLx dan 8 bit dihandle oleh register THx serta *overflow* dihandle oleh TFx. Yang kedua adalah mode 1 yaitu 16 bit *timer*, dimana TLx dan THx sama-sama 8 bit. Pada mode 2 adalah 8 bit *Auto Reload Timer*, artinya TLx akan menghitung dari 00H sampai FFH kemudian saat kembali ke 00H maka nilai THx akan diletakan ke TLx. Hal itu dilakukan berulang-ulang (*Auto Reload*) sampai TFx di set *overtlow* yang mana tergantung inisialisasi TMOD dan THx.

Sumber pewaktu dari *timer* bisa dari dalam *(internal)* maupun dari luar (*eksternal)*. Untuk kasus sumber timer eksternal (disebut juga kondisi *counter)* maka yang mengaktifkan adalah bit C/D pada TMOD jika bit 1 maka sumber *internal* sebagai *interval timing* dan *eksterna*l sebagai *event counting* atau

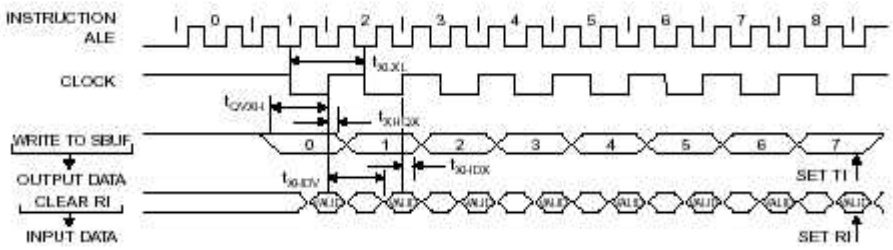
menghitung sinyal yang datang dari luar. Jika bit = 0, maka *timer* bersifat kontinyu. Frekuensi *timer eksternal* maksial adalah 500 kHz dengan *clock* osilator 12 MHz. secarea software *timer* diaktifkan dengan perintah SETB TR0 dan dihentikan dengan perintah CLR TR0.

2.13. Serial

Pada port kontrol (3) ada sepasang pin serial, dimana digunakan untuk komunikasi data serial. Mode komunikasi serial ini ada empat macam yang kesemuanya diatur pada *register* SCON (*Serial Kontrol Register*) pada alamat 099H.

Mode pertama yaitu mode 0, dimana aktif bila bit SM0 dan SM1 bernilai 0. hal ini mengakibatkan serial menjadi *shift register* mode dimana pin TxD sebagai *clock* dan RxD sebagai data *input* maupun *output* 8 bit yang dilewatkan satu persatu dengan LSB yang pertama dikirim.

Mode kedua yaitu mode 1, 8-bit UART dengan *Baud Rate* bisa diatur, artinya kecepatan pengiriman data per detik bisa diubah. Hal itu disebabkan pengiriman data dilewatkan TxD dan penerimaan data lewat RxD, sedangkan *clock* diambildari *timer internal*, yaitu melihat TFx dari *timer*.



Gambar 2.3.*Timming* Diagram Mode 0 Serial[2]

Semuanya itu mudah dilakukan sebab lebar data tetap yaitu 10 bit, bit pertama yang dikirim disebut start bit dengan kondisi harus selalu 0, kemudian bit kedua sampai dengan bit kesembilan adalah data yang sebenarnya dimulai dari LSB ke MSB (D) dulu dikirim sampai dengan D7) dan terakhir adalah bit *stop* yang harus berkondisi *high*. Begitu *stop* bit ini sudah muncul di TxD untuk dikirim maka *flag regist*er T1 akan diset secepat mungkin sehinggah bisa mengirim 8 bit data serial selanjutnya.

Pada sisi penerima (RxD), setelah menerima perubahan dari 1 ke 0, maka counter pembagi 16 mulai menghitung dari awal, RxD mengambil data 0 dulu sebanyak delapan kali, bila tidak sampai delapan kali pengambilan terjadi perubahan dari 0 ke 1 (data tidak 0) maka dianggap bahwa penerima telah di *trigger* oleh *noise* dan bukan dimasukan sebagai *input* yang *valid*.

Mode ketiga yaitu mode 2, 9-bit UART dengan *Baud Rate* tetap, yaitu pengiriman dengan baud rate 1/32 atau 1/64 dari osilator. Jumlah bit yang dikirim adalah 11 bit, yaitu bit start yang pertama, kemudian delapan bit data yang

[2] Ibid, hal 174

sebenarnya, setelah itu ada bit ke sembilan yang dapat diprogram apakah 1 atau 0 tergantung kita, dan yang terakhir adalah bit *stop*. Adanya bit yang kesembilan dimana dapoat kita atur apakah 1 atau 0 hampir sama dengan cara kerja *parity* bit.

Mode keempat yaitu mode 3, 9-bit UART dengan *Baud Rate* bisa diatur. Mode ini sama dengan mode 2 hanya bedanya pada *baud rate* yang bisa diatur.

Untuk penginisialisasiannya dengan perintah SETB REN atau MOV SCON,#xxx1xxxxxxB. *Baud rate* pada serial komunikasi ini diatur berdasarkan kristal osilator. Untuk kemudahan pemrograman jika memakai serial maka kristal osilator dipasang 11,059 MHz.

Contoh inisialisasinya adalah sebagai berikut :

```
org 00H
mov SCON, #52H ;serial port mode 1
mov TMOD,#20H ;timer 1 mode 2
TH1,#-13                    ;reload counter untuk baund 2400
Setb TR1            ;timer 1 mulai menghitung
End
```

2.14. /RD dan /WR

Sepasang pin yang terakhir adalah untuk pembacaan dan penulisan memori *eksternal* yaitu /RD dan /WR. Kedua pin ini aktif *low*. Kerja pin-pin ini bersama-sama dengan pin kontrol ALE dan /PSEN.

2.15. PSEN

Program *Store Enable* adalah membaca *strobe* ke memori program *eksterna*l. Saat AT89C51 mengeksekusi kode dari memori program *eksterna*l, PSEN teraktifkan dua kali setiap putaran mesin (*machine cycle*), kecuali bila pengaktifan dua PSEN diloncati selama tiap akses ke memori data *eksternal*.

2.16. EA/Vpp

External Access Enable. EA harus terjalin ke *ground* agar komponen enable, untuk fetch kode mulai dari 0000H sampai FFFFH di lokasi memori program *eksternal*. Catatan, bagaimanapun bahwa jika *clock* bit 1 terprogram, EA akan ter*latch* dibagian dalam pada *reset.* EA seharusnya terjalin ke Vcc untuk eksekusi-eksekusi program *interna*l.

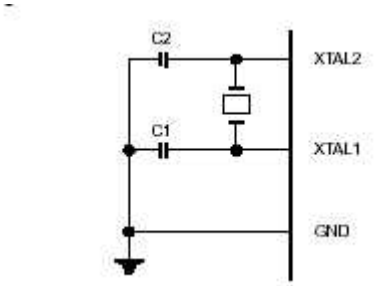
Pin ini juga menerima 12 volt *programming enable voltage* (Vpp) selama pemrograman *flash*, untuk bagian-bagian yang membutuhkan Vpp 12 volt.

2.17. Pin Osilator

Sepasang pin ini diberi kristal eksternal yang di *bypass* dengan dua buah kapasitor kira-kira 30 pF ± 10%. Kristal yang dipakai umumnya 12 MHz, tetapi apabila ada komunikasi serial lebih mudah menggunakan kristal dengan ukuran 11,059 MHz dalam penentuan *Baud Rate*nya. XTAL1 adalah input ke pembalikan penguat oscillator (*inverting Oscillator amplifier*) dan input ke *clock internal* pengoprasian rangkaian. Sedangkan XTAL2 adalah *output* dari pembalikan penguat *oscillator*.

2.18. Karakteristik Oscilator

XTAL1 dan XTAL2 adalah *input* dan *output*, berturut-turut dari *inverting amplifier* yang mana dapat dikonfigurasi sebagai missal pada *chip* oscillator. Salah satu kristal kwarsa atau resonator kramik bisa digunakan. Untuk menjalankan alat dari sumber *clock eksternal* (*exsternal clock source*), XTAL2 seharusnya tidak terhubung kiri saat XTAL1 dijalankan. Dimana tidak membutuhkan *duty cycle* di sinyal clock *eksternal*, semenjak input ke *internal clocking circuitry* melalui satu dibagi dua flip-flop, tetapi spesifikasi waktu *low* dan *high* tegangan maksimum atau minimum harus terpantau.



Gambar 2.4. Rangkaian Kristal AT89C51[3]

2.19. Mode Kerja Mikrokontroler AT89C51

Pada mikrokontroler AT89C51 dalam register PCON terdapat penambahan 4 bit kontrol. Dua bit merupakan general *purpose flag* bit dan 2 bit lainnya yaitu bit PD (*power down*) dan bit IDL (*idle*).

[3] " AT89 Series Hardware Description ", Atmel Inc, Hal 32

2.20. Pemrograman Bahasa Assembly MCS

Bahasa assembly menggantikan kode-kode biner dari bahasa mesin dengan " *mnemonik* " yang muda diingat. Misalnya, sebuah instruksi penambahan dalam basaha mesin disajikan dengan kode " 10110011 " yang dalam bahasa assembly dapat disajikan dalam *mnemonik* ADD, sehingga mudah diingat dibanding sederetan angka 0 dan 1.

Tidak hanya itu, perintah penambahan membutuhkan suatu operan baik berupa data langsung maupun suatu lokasi yang menyimpan data yang bersangkutan. Dengan demikian kode untuk ADD bisa berbeda-beda tergantung kebutuhan atau jenis operan dalam bentuk yang berbeda-beda.

Program bahasa assembly adalah sebuah program yang terdiri atas label-label, *mnemonik* dan lain sebagainya. Masing-masing pernyataan berhubungan dengan suatu instruksi mesin. Bahasa assembly, sering juga disebut kode sumber (*Source code*) atau kode simbolik (*symbolic code*) tidak dapat dijalankan oleh prosesor.

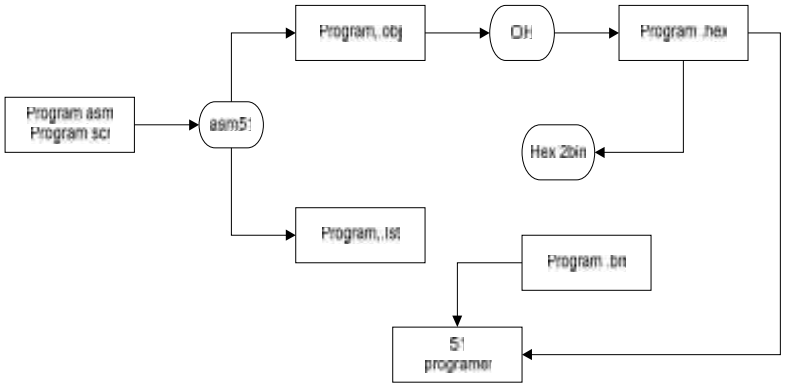
Assembler adalah suatu program yang dapat menterjamahkan program bahasa assembly ke program bahasa mesin. Program bahasa mesin ini dapat berbentuk " *absolute*" atau " *relocatable* ". Berikutnya dilakukan " *linking* " untuk mengatur alamat absolute agar program dapat dijalankan oleh prosesor yang bersangkutan.

2.21. Operasi Assembler

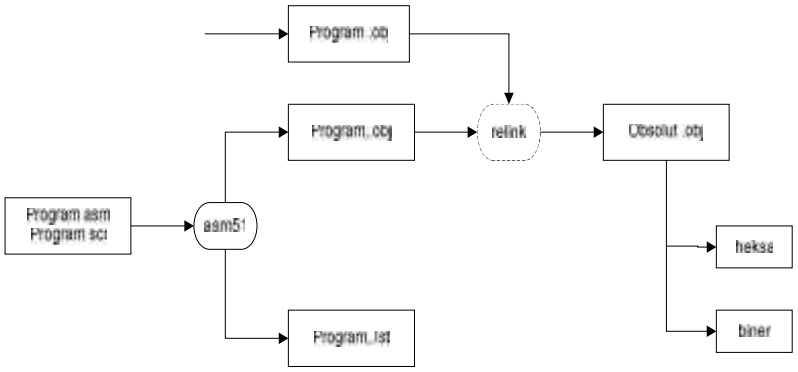
Banyak program-program assembler yang digunakan untuk mengembangkan aplikasi mikrokontroler MCS51. Progaram assembler dari Intel untuk mikrokontroler MCS51, ASM51.

Program assembler, ASM51, dapat dijalankan pada komputer PC dengan sistem operasi DOS atau Windows. Karena komputer yang digunakan prosesor bukan keluarga MCS51, maka ASM51 sering dikenal sebagai pemrogram " cross assembler ". Program bisa diketik dengan menggunakan sembarang editor DOS maupun Windows da harus disimpan dalam format teks (bukan rtf atau sejenisnya). Kemudian di assembly menggunakan ASM51 sehingga menghasilkan berkas objek, tetapi tidak bisa dijalankan dengan komputer yang bersangkutan, karena prosesornya bukan kelurga MCS51. Karena tidak bisa dijalankan langsung, maka bisa digunakan perangkat lunak lain untuk melakukan emulasi atau simulasi program, misalnya TS 8051 Emulator.

Simulasi program MCS51 bersifat opsional, artinya boleh dilakukan maupun tidak tergantung pada kebutuhan, jika program MCS51 berkerja sesuai dengan keinginan, maka langkah berikutnya yaitu mengubah jadi format heksa atau biner yang siap untuk dikirim / disimpan dalam AT89C51 / 52 /55 melalui pemrogram (programmer). Proses selengkapnya ditunjukkan pada gambar 2.5. dan gambar 2.6



Gambar 2.5. Proses Pembuatan program aplikasi modul tunggal[4]



Gambar 2.6. Proses pembuatan program aplikasi sistem modular[5]

Cara menggunakan ASM51 sebagai berikut :

ASM51 berkas_program (kontrol_program)

Berkas_program merupakan nama berkas yang di assemblykan dan

Kontrol_assembler menentukan efek assembly yang dijalankan.

[4] Ibid, hal 63.
[5] Ibid, hal 64.

2.22. Format Program Bahasa Assembly

Program bahasa assembly berisikan:

1. Instruksi-instruksi mesin:
2. Pengarah-pengarah assembler;
3. Kontrol-kontrol assembler:
4. Komentar-komentar.

Instruksi-instruksi mesin merupakan mnemonik yang menyatakan suatu instruksi yang bisa dijalankan (misalnya MOV). Pengarah assembler (*assembler directive*) merupakan unstruksi ke program assembler yang mendefinisikan struktur program, simbol-simbol, data, konstanta, dan lain-lain (misalnya: ORG). Kontrol-kontrol assembler mengatur (menentukan) mode-mode assembler dan aliran assembler langsung (misalnya $TITLE). Komentar perlu dituliskan agar program muda dibaca, tidak harus perinstruksi bisa juga sekumpulan instruksi yang mengerjakan suatu operasi

Baris-baris program yang mengandung instruksi mesin atau pengarah assembler harus mengikuti aturan program assembler ASM51. Masing-masing baris atas beberapa *field* yang dipisahkan dengan sepasi atau tabulasi. Format umumnya:

[label :] mnemonik [operan] [,operan] […] [; komentar]

2.22.1. Alamat Data

Banyak intruksi yang mengakses lokasi-lokasi memori menggunakan pengalamatan langsung dan membutuhkan alamat memori data internal (00h

hingga 7Fh) atau alamat SFR (8 0 h hingga FFh) pada operan. Simbol yang telah telah baku dalam SFR dapat digunakan, misalnya:

```
MOV     A, 45H ;
MOV     A, SBUF; sama seperti MOV  A, 99 h
```

2.22.2. Alamat Bit

Salah satu kelebihan mikrokontroler 51 adalah kemampuannya bisa mengakses alamat-alamat per-bit tanpa menggunakan cara khusus. Instruksi yang melibatkan lokasi -lokasi yang teralamati-bit (bit addressable) harus menyediakn suatu alamat bit dalam memori data internal (00 h hingga 7 Fh) atau alamat bit di ruang SFR (80 h hingga FFh). Cara menuliskannya ada tiga cara:

1. Secara eksplisit menggunakan alamatnya langsung (misal : SETB 0EH);
2. Menggunakan tanda titik (.) antara alamat byte dan posisi bit (misal : SETB ACC.7)
3. Menggunakan simbol yang baku (misal: JNB TI, $).

2.22.3. Alamat Kode

Suatu alamat kode digunakan dalam operan instruksi lompatan, termasuk lompatan relative (SJMP dan lompatan bersyarat), lompatan dan *call absolute* (ACALL, AJMP) dan lompatan *call far* (LJMP, LCALL). Alamat kode tersebut biasanya diwakili dalam bentuk suata label, misalnya:

```
DISINI:  . . .

         . . .

         SJMP DISINI
```

ASM51 akan menentukan alamat kode dengan benar kemudian menyisipkan kedalam instruksi sebagai *offset* 8-bit bertanda, alamat halaman 11-bit atau alamat panjang 16-bit sesuai dengan kasusnya.

2.32. Jump Dan Call Umum

ASM51 membolehkan kita untuk menggunakan mnemonik JMP atau CALL yang umum; mnemonik JMP digunakan sebagai wakiil dari SJMP, AJMP atau LJMP, sedangkan mnemonik CALL mewakili ACALL atau LCALL.

Assembler akan menkonversi mnemonik umum ini menjadi instruksi yang sesungguhnya mengikuti beberapa aturan sederhana.

1. Diubah ke SJMP jika tidak ada dalam acuan alamat di depan (tujuan lompatan sebelum instruksi JMP yang bersangkutan) dan jangkauan lompatan berada dalam 128 byte (lokasi).
2. Diubah ke bentuk AJMP jika tidak ada acuan lompatan di depan dan tujuan lompatan masih berada di dalam blok 2K yang sama;
3. Jika aturan 2 dan 3 tidak dipenuhi maka akan diubah ke bentuk LJMP.

Tidak selamanya konversi merupakan cara pemrograman yang baik. Misalnya, tujuan lompatan beberapa di depan (setelah instruksi JMP yang bersangkutan) maka JMP yang umum tersebut akan diubah ke bentuk LJMP, walau SJMP lebih cocok. Perhatikan contoh berikut:

```
        ORG     0h
MULAI : INC     A
        JMP     MULAI
        ORG     MULAI + 200
        JMP     MULAI
        JMP     SELESAI
SELESAI : INC   A
        END
```