BAB II

LANDASAN TEORI

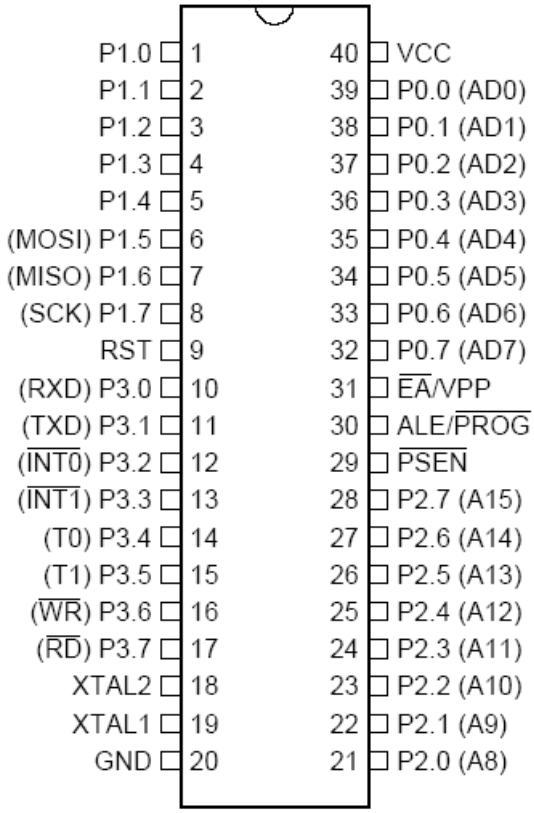
2.1. Perangkat Keras

Dalam merancang sebuah peralatan yang cerdas, diperlukan suatu perangkat keras (*hardware*) yang dapat mengolah data, menghitung, mengingat dan mengambil pilihan. Mikrokontroler merupakan salah satu jawabannya. Vendor dari mikrokontroler ini ada beberapa macam, diantaranya yang paling terkenal adalah Atmel, Motorola dan Siemens.

2.1.1. Mikrokontroler AT89S51

AT89S51 merupakan mikrokontroler 8 bit dengan 4 K*byte Flash* PEROM *(Programmable and Erasable Read Only Memory)* yang konfigurasi dan instruksinya kompatibel dengan standar 80C51 dan 80C52. AT89S51 mampu ditulis dan dihapus sebanyak 1.000 kali. AT89S51 memiliki 128 x 8-bit RAM internal, 32 jalur I/O *Programmable*, dua buah *Timer/Counter* 16 bit, tujuh sumber interupsi dan kanal *programmable* serial. Selain itu AT89S51 memiliki mode *Low–power Idle*, *Power down* dan tiga tingkat pengunci program memory. AT89S51 dapat beroperasi statis dari 0-24 MHz.

2.1.2. Konfigurasi Pin AT89S51



Gambar 2.1. Mikrokontroler AT89S51 dan Konfigurasi 40 pin

Mikrokontroler AT89S51

Penjelasan fungsi pin-pin mikrokontroller AT89S51 adalah sebagai berikut:

a. Port 0 (pin 32-39)

Adalah 8 bit *bidirectional* port I/O. port 0 akan memultipleks data dan alamat ketika mengakses program eksternal dan mode port 0 mempunyai *pull-ups* internal.

b. Port 1 (pin 1-8)

Adalah 8 bit *bidirectional* port I/O dengan *pull-ups* internal.

c. Port 2 (pin 21-28)

Adalah 8 bit *bidirectional* port I/O dengan *pull-ups* internal. Port 2 dapat digunakan untuk mengeluarkan alamat 8 bit teratas ketika mengakses memori eksternal.

d. Port 3 (pin 10-17)

Adalah 8 *bitbidirectional* port I/O dengan *pull-ups* internal. Selain itu port 3 memiliki fungsi lain yaitu:

Tabel 2.1 Fungsi lain pin port 3 pada AT89S51

Pin Port	Fungsi lain
P3.0	RXD (port input serial)
P3.1	TXD (port output serial)
P3.2	INT0 (interupsi eksternal 0)
P3.3	INT1 (interupsi ekternal 1)
P3.4	T0 (masukan eksternal pewaktu/ pencacah 0)
P3.5	T1 (masukan eksternal pewaktu/ pencacah 1)
P3.6	WR (sinyal *write* pada data memori eksternal)
P3.7	RD (sinyal *read* pada data memori eksternal)

e. RST (pin 9)

Adalah input *reset* (aktif logika *high*). Pulsa transisi dari *low* ke *high* akan me-*reset* AT89S51. Pin ini dihubungkan dengan rangkaian *power reset.*

f. ALE / PROG (*Address Latch Enable / Programmed*) (pin 30)

Pin ini digunakan untuk menahan alamat memori eksternal selama pelaksanaan instuksi.

g. PSEN (*Program Store Enable*) (pin 29)

Pin ini berfungsi pada saat mengeksekusi program yang terletak pada memori eksternal. PSEN akan aktif dua kali setiap *cycle*.

h. EA / VPP (*External Access Enable*) (pin 31)

Pin ini berfungsi untuk menentukan mikrokontroller mengeksekusi program eksternal atau internal . Apabila *External Access Enable* (EA) diberi logika *low* maka mikrokontroller akan mengeksekusi program eksternal dan apabila diberi logika *high* mikrokontroler akan mengeksekusi program internal.

i. XTAL1 (pin 19)

Adalah pin input pada rangkaian osilator internal .

j. XTAL2 (pin 18)

Adalah pin output dari rangkaian osilator internal .

k. VCC (pin 40)

Adalah input catu daya sebesar +5 V.

l. GND (pin 20)

Adalah input *ground.*

2.1.3. Struktur Memori

AT89S51 mempunyai struktur memori yang terdiri dari :

1. RAM internal, merupakan memori sebesar 128 byte yang biasanya digunakan untuk menyimpan variabel atau data yang bersifat sementara.

2. *Special Function Register (SFR),* merupakan memori yang berisi register-register yang mempunyai fungsi-fungsi khusus yang disediakan oleh mikrokontroler, seperti *timer* serial dan lain-lain.
3. *Flash* PEROM, merupakan memori yang digunakan untuk menyimpan instruksi-instruksi MCS51

AT89S51 mempunyai struktur memori yang terpisah antara RAM internal dan *Flash* PEROM. RAM internal dialamati oleh RAM *Address Register* sedangkan *Flash* PEROM yang menyimpan instuksi-instruksi MCS51 yang dialamati oleh program *Addres Register*.

RAM internal terdiri atas:

a. Register Banks

AT89S51 mempunyai 8 buah register yang terdiri dari R0 hingga R7. Register-register tersebut selalu terletak pada alamat 00H hingga 07H pada setiap kali di-*reset*. Posisi R0 hingga R7 dapat dipindah ke *Bank* 1 (08H hingga 0FH), *Bank* 2 (10H hingga 17H), dan *Bank* 3 (18H hingga 1FH) dengan mengatur bit RS0 dan RS1.

b. Bit *Addressable* RAM

RAM pada alamat 20H hingga 2FH dapat diakses secara pengalamatan bit sehingga hanya sebuah instruksi setiap bit dapat di-*set, clear, AND* dan *OR*.

c. RAM keperluan umum

RAM pada alamat 30H hingga 7FH dapat diakses dengan pengalamatan langsung maupun tak langsung.

Special Function Register (SFR) yang dimiliki oleh AT89S51 sebanyak 21 SFR yang terletak pada alamat 80H hingga FFH. Beberapa dari SFR mampu dialamatkan dengan pengalamatan bit. Dibawah ini beberapa register pada SFR, yaitu:

1. *Accumulator*

   Register ini terletak pada alamat E0H. *Accumulator* banyak digunakan untuk operasi aritmatika dan operasi logika. Register ini juga diperlukan pada proses pengambilan dan pengiriman data ke memori eksternal.

2. *Port*

   AT89S51 mempunyai 4 buah *port*, yaitu *port* 0, *port* 1, *port* 2, dan *port* 3 yang terletak pada alamat 80H, 90H, A0H dan B0H. Semua *port* ini dapat diakses dengan pengalamatan bit.

3. *Stack Pointer*

   Stack Pointer adalah suatu *register* yang menunjuk pada *stack*, nilai pada *stack pointer* akan bertambah jika data disimpan pada *stack* melalui perintah *PUSH, CALL* atau rutin interupsi dilaksanakan .

4. *Data Pointer*

   Register ini merupakan *register* 16 bit yang terdiri atas *register* Data Pointer Low (DPL) dan Data Pointer High (DPH).

5. *Register Timer*

   AT89S51 mempunyai 2 buah 16 bit *Timer /Counter*, yaitu *Timer* 0 dan *Timer* 1.

Program yang ada pada *Flash* PEROM akan dijalankan jika pada sistem di-*reset*, pin External Access (EA) berlogika high sehingga mikrokontroller aktif berdasarkan program pada *Flash* PEROM.

2.1.4. Antarmuka Serial AT89S51

Pada port serial AT89S51 penerimaan dan pengiriman data port serial melalui register SBUF. Penulisan ke *Serial Buffer* (SBUF) berarti mengisi register pengiriman ke SBUF, sedangkan pembacaan dari *Serial Buffer* (SBUF) berarti membaca register penerimaan SBUF. *Port* serial pada AT89S51 bisa digunakan dalam 4 mode kerja yang berbeda, terdiri dari 1 mode bekerja secara sinkron dan 3 lainnya bekerja secara asinkron.

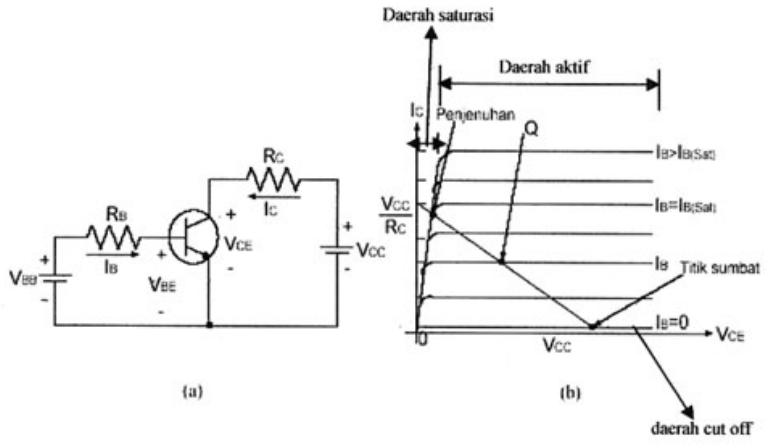
Adapun mode kerja dari port serial, antara lain yaitu :

1. *Mode 0 :* Mode ini bekerja secara sinkron, data serial dikirim dan diterima malalui kaki P3.0 (Rxd), sedangkan kaki P3.1 (Txd) digunakan untuk menyalurkan detak pendorong data serial yang dibangkitkan oleh AT89S51. Data dikirim dan diterima 8 bit sekaligus dimulai dari bit *Less Significant Bit* (LSB) dan diakhiri dengan bit *Most Significant Bit* (MSB). Kecepatan *boud rate* 1/12 frekuensi kristal yang digunakan.
2. *Mode 1 :* Pada mode ini data dikirim melalui kaki P3.1 (Txd) dan diterima melalui kaki P3.0 (Rxd) secara asinkron (juga mode 2 dan 3). Pada mode 1 data dikirim atau diterima 10 bit sekaligus, diawali dengan 1 bit *start*, disusul dengan 8 bit data yang dimulai dari bit *Less Significant Bit* (LSB) dan diakhiri dengan 1 bit *stop*. Pada AT89S51/52 yang berfungsi sebagai penerima bit *stop* adalah RB8 dalam register SCON. Kecepatan *boudrate*

bisa diatur sesuai dengan keperluan dengan menggunakan *timer*. Mode 2 dan 3 yang umum dikenal dengan UART.

3. *Mode 2 :* Data dikirim atau diterima 11 bit sekaligus, diawali dengan 1 bit *start*, disusul 8 bit data, kemudian bit ke 9 yang bisa diatur lebih lanjut, diakhiri dengan 1 bit *stop*. Pada AT89S51/52 yang berfungsi sebagai pengirim, bit 9 tersebut berasal dari bit TB8 dalam register SCON dan yang berfungsi sebagai penerima, bit 9 ditampung pada bit RB8 dalam register *Serial Control* (SCON), sedangkan bit *stop* diabaikan tidak ditampung. *Boudrate* bisa dipilih antara 1/32 atau 1/64 frekuensi kristal yang digunakan.
4. *Mode 3 :* Mode ini sama dengan mode 2 hanya saja *boudrate*-nya bisa diatur sesuai dengan keperluan seperti mode1.

2.1.5. Transistor Sebagai Saklar



Gambar 2.2. (a) Bias basis. (b) Garis beban dc.

Gambar 2.2.a memperlihatkan rangkaian *bias basis*. Sebuah sumber tegangan V_{BB} membias maju dioda emiter melalui resistor R_B yang juga berfungsi membatasi arus. Penjumlahan tegangan di sekitar loop input memberikan :

$$I_B R_B + V_{B\ E} - V_{B\ B} = 0 \quad \text{..... (2.1.)}$$

sehingga arus bias pada basis adalah :

$$I_B = \frac{V_{BB} - V_{BE}}{R_B} \quad \text{..... (2.2.)}$$

Dengan V_{BE} = 0,7 V untuk transistor silikon dan 0,3 V untuk germanium. Dalam rangkaian kolektor, sumber tegangan *Vcc* membias balik dioda kolektor melalui *RC*.

Persamaan tegangan kolektor emitter dapat diperoleh melalui hukum ohm, yaitu :

$$V_{C\ C} + V_{C\ C} + I_C R_C = 0 \quad \text{..... (2.3.)}$$

$$I_C = \frac{V_{C\ C} - V_{C\ E}}{R_C} \quad \text{..... (2.4.)}$$

$$V_{C\ E} = V_{C\ C} - I_C R_C \quad \text{..... (2.5.)}$$

Dalam rangkaian bias basis yang diperlihatkan Gambar 2.2.a, V_{CC} dan R_C adalah konstan. Pada persamaan 2.5. apabila disederhanakan akan dapat ditentukan besarnya arus *Ic*, seperti terlihat pada persamaan 2.6.

$$I_C = \frac{V_{C\ C} - V_{C\ E}}{R_C} \quad \text{..... (2.6.)}$$

Ratio harga $\beta_{dc} = \frac{I_C}{I_B}$ ..... (2.7.)

Gambar 2.2.b menunjukkan grafik dari persamaan (2.6.) memotong kurva dari kolektor. Perpotongan vertikal adalah pada V_{CC}/R_C dan perpotongan horizontal pada V_{CC}. Garis ini disebut *garis beban dc* karena garis ini menyatakan semua titik operasi yang mungkin. Perpotongan dari garis beban *dc* dengan arus basis adalah titik operasi kerja dari transistor. Titik dimana garis beban memotong kurva $I_B = 0$ disebut *titik sumbat* (*cutoff*). Pada titik ini, arus basis nol dan arus kolektor sangat kecil, sehingga dapat diabaikan (hanya ada arus bocor I_{CEO}). Pada titik sumbat, dioda emiter tidak lagi dibias maju, dan transistor kehilangan kerja normalnya. Untuk itu digunakan suatu pendekatan, bahwa tegangan kolektor-emiter adalah :

$$V_{C(cut\ off)} \cong V_{CC} \quad \text{..... (2.8.)}$$

Perpotongan dari garis beban dan kurva $I_B = I_{B(sat)}$ disebut saturasi. Pada titik ini, arus basis sama dengan $I_{B(sat)}$ dan arus kolektor adalah maksimum. Pada saturasi, dioda kolektor tidak lagi dibias balik, dan transistor kehilangan kerja normalnya. Untuk itu digunakan suatu pendekatan, bahwa arus kolektor pada saturasi adalah seperti di perlihatkan pada persamaan 2.5.

$$I_{C(sat)} \cong \frac{V_{CC}}{R_C} \quad \text{..... (2.9.)}$$

$$I_{MAX} \cong \frac{V_{CC}}{R_C + R_E} \quad \text{..... (2.10.)}$$

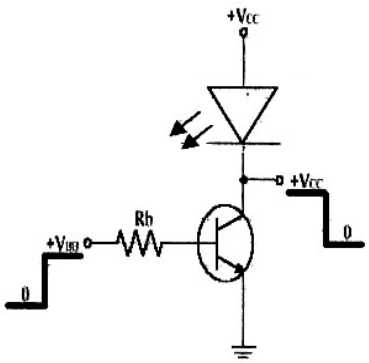
dan arus basis yang tepat menimbulkan saturasi adalah seperti di perlihatkan pada persamaan 2.11.

$$I_{B(sat)} = \frac{I_{C(sat)}}{\beta_{dc}} \quad \text{..... (2.11.)}$$

Dengan $ß_{dc}$ merupakan penguatan arus. Salah satu penggunaan dari transistor adalah sebagai *switch* atau saklar, artinya bahwa mengoperasikan transistor pada salah satu dari saturasi atau titik sumbat, tetapi tidak di tempat-tempat sepanjang garis beban. Jika sebuah transistor berada dalam keadaan saturasi, transistor tersebut seperti sebuah *switch* yang tertutup dari kolektor ke emiter. Jika transistor tersumbat (*cut off*), transistor seperti sebuah *switch* yang terbuka. Dalam transistor dikenal istilah aturan disain *Soft saturation* dan *Hard saturation*. *Soft saturation* berarti kita membuat transistor hampir saturasi, dimana arus basis hanya cukup untuk mengoperasikan transistor pada ujung atas dari garis beban. *Soft saturation* tidak dapat diandalkan pada produksi masal karena adanya perubahan-perubahan pada $ß_{dc}$ dan $I_{B(sat)}$. *Soft saturation* akan mengacu pada rancangan dimana transistor akan jenuh secara terbatas, dalam hal ini penguatan arus hanya sedikit lebih kecil daripada penguatan arus aktif.

Pada kondisi *Hard saturation*, berarti terdapat arus basis yang cukup kuat untuk membuat transistor saturasi pada semua harga dari $ß_{dc}$. Untuk keadaan yang paling buruk dari temperatur dan arus, hampir semua transistor silikon sinyal kecil mempunyai $ß_{dc}$ lebih besar daripada 10. Karena itu, suatu pedoman disain untuk *hard saturation* adalah mempunyai arus basis kira-kira sepersepuluh dari harga saturasi arus kolektor, ini menjamin *hard saturation* pada semua kondisi kerja. Sebagai contoh, jika ujung atas garis beban mempunyai arus kolektor sebesar 10 mA, maka akan didapatkan arus basis sebesar 1 mA. Hal ini menjamin keadaan saturasi untuk semua transistor, arus, temperatur dan sebagainya. Di gunakan aturan 10 : 1 dalam proses mendisain rangkaian *switching* transistor, kecuali jika di tentukan lain. Ingat, ini hanya suatu pedoman. Jika nilai tahanan standar

menghasilkan perbandingan I_C / I_B sedikit lebih besar daripada 10, hampir setiap transistor sinyal kecil akan menuju keadaan *hard saturation*.



Gambar 2.3. Contoh transistor yang digunakan sebagai switch.

Gambar 2.3. menunjukkan sebuah rangkaian *switching* transistor yang digerakkan oleh tegangan step. Jika tegangan *input* nol, transistor tersumbat (*cut off*). Dalam hal ini, transistor kelihatannya seperti sebuah *switch* yang terbuka. Dengan tidak adanya arus yang mengalir ke kolektor melalui LED, maka tegangan *output* sama dengan $+V_{BB}$. Rangkaian ini disebut sebuah *LED driver*, karena transistor mengendalikan LED. Jika tegangan *input* rendah (*low*), transistor akan tersumbat (*cut off*) dan LED padam. Jika tegangan *input* tinggi (*high*), transistor saturasi dan LED menyala.

2.1.6. Penyearah (Rectifier)

Penyearah adalah proses merubah atau menyearahkan arus bolak-balik menjadi arus searah. Arus bolak balik ini berasal dari tegangan jala-jala. Secara umum penyearah dibagi menjadi tiga kategori yaitu :

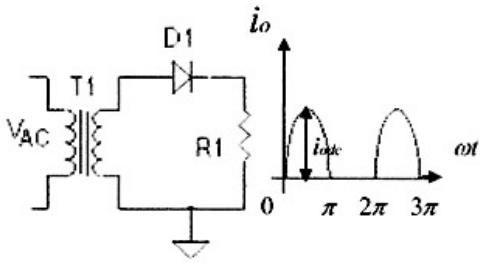
1. Penyearah setengah gelombang
2. Penyearah gelombang penuh dengan trafo *center tap* (ct)

3. Penyearah gelombang penuh sistem jembatan (*bridge rectifiers*) memakai filter (penyaring) dengan kapasitor (sebagai perata / penyaring)

Komponen utama yang diperlukan dalam penyearahan adalah transformator, dioda dan kapasitor elektrolit.

2.1.7. Penyearah Setengah Gelombang

Penyearah ini bekerja dengan menggunakan satu dioda, sehingga hanya pulsa positif yang dapat terambil. Penyerah ini praktis sederhana, tetapi kekurangannya adalah bahwa gelombang keluaran bukan gelombang penuh sehingga rentan sekali akan *ripple.*



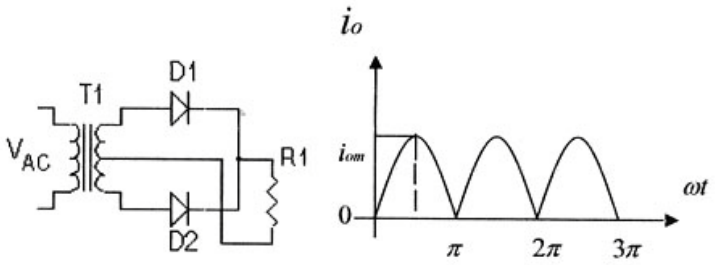
Gambar 2.4. Penyearah setengah gelombang

2.1.8. Penyearah Gelombang Penuh Sistem CT

Penyearah ini menggunakan transformator jenis ct dengan dua buah dioda sebagai penyearah. Dioda bekerja secara bergantian pada D1 menghasilkan

gelombang untuk setengah siklus (cycle) dan D2 juga menghasilkan setengah siklus sehingga untuk satu perioda dihasilkan gelombang penuh yaitu :

$0 < \omega t < \pi$ dan $\pi < \omega t < 2\pi$.



Gambar 2.5. Penyearah sistem CT

Misalnya tegangan input berbentuk sinusoida :

$$V_i = V_M \sin \omega t \quad \text{..... (2.12.)}$$

Maka arus output :

$$I_O = I_{O\,M} \sin \omega t \text{ , untuk } 0 < \omega t < \pi \quad \text{..... (2.13.)}$$

$$I_O = -I_{O\,M} \sin \omega t \text{ , untuk } \pi < \omega t < 2\pi \quad \text{..... (2.14.)}$$

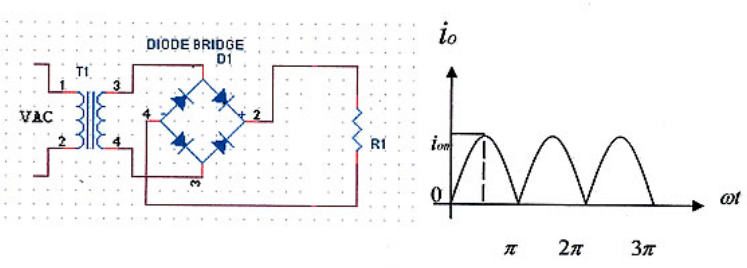
Arus output maksimum I_{OM} yang mengalir pada rangkaian :

$$I_{O\,M} = \frac{V_M}{R_f + R_L} \quad \text{..... (2.15.)}$$

2.1.9. Penyearah Gelombang Penuh Sistem Jembatan

Penyearah ini menggunakan 4 buah dioda sebagai penyearah. Pada siklus pertama dua dioda bekerja untuk menyearahkan atau mengambil pulsa positif. Siklus selanjutnya dua dioda berikutnya yang bekerja untuk mengambil pulsa negatif. Keuntungan penyearah ini adalah bahwa keluaran berupa gelombang

penuh dan jika salah satu dioda rusak, maka dioda yang satunya akan tetap bekerja.



Gambar 2.6. Penyearah sistem jembatan

2.2. PERANGKAT LUNAK

Pada perancangan ini digunakan bahasa assembly keluarga MCS-51 yang kompatibel dengan mikrokontroler AT89S51.

2.2.1. Bahasa Assembly MCS-51

Bahasa yang digunakan untuk memprogram IC mikrokontroler AT89S51 adalah bahasa assembly untuk MCS-51. angka 51 merupakan jumlah instruksi pada bahasa ini hanya ada 51 instruksi. Dari 51 instruksi, yang sering digunakan orang hanya 10 instruksi. Instruksi –instruksi tersebut antara lain :

1. Instruksi *MOV*

   Perintah ini merupakan perintah untuk mengisikan nilai ke alamat atau register tertentu. Pengisian nilai dapat secara langsung atau tidak langsung.

   Contoh pengisian nilai secara langsung

   MOV R0,#20h

   Perintah di atas berarti : isikan nilai 20 Heksadesimal ke register 0 (R0).

   Tanda # sebelum bilangan menunjukkan bahwa bilangan tersebut adalah nilai.

Contoh pengisian nilai secara tidak langsung

MOV 20h,#80h

.....

.....

MOV R0,20h

Perintah di atas berarti : isikan nilai yang terdapat pada alamat 20 Heksadesimal ke register 0 (R0).

Tanpa tanda # sebelum bilangan menunjukkan bahwa bilangan tersebut adalah alamat.

2. Instruksi *DJNZ*

Decreament Jump if Not Zero (DJNZ) ini merupakan perintah untuk mengurangi nilai register tertentu dengan 1 dan lompat jika hasil pengurangannya belum nol. Contoh ,

```
        MOV  R0,#80h
Loop:   .....
        .....
        DJNZ R0,Loop
        .....
```

R0 -1, jika belum 0 lompat ke *loop*, jika R0 = 0 maka program akan meneruskan ke perintah pada baris berikutnya.

3. Instruksi *ACALL*

Instruksi ini berfungsi untuk memanggil suatu rutin tertentu. Contoh :

```
        .....
        ACALL  TUNDA
        .....
TUNDA:
        .....
```

4. Instruksi *RET*

Instruksi *RETURN (RET)* ini merupakan perintah untuk kembali ke rutin pemanggil setelah instruksi *ACALL* dilaksanakan. Contoh,

```
    ACALL  TUNDA
    .............
TUNDA:
    .................
    RET
```

5. Instruksi *JMP (Jump)*

   Instruksi ini merupakan perintah untuk lompat ke alamat tertentu. Contoh,

```
Loop:
    .................
    .............
    JMP Loop
```

6. Instruksi *JB (Jump if Bit)*

   Instruksi ini merupakan perintah untuk lompat ke alamat tertentu, jika pin yang dimaksud berlogika *high* (1). Contoh,

```
Loop:
    JB   P1.0,Loop
    .................
```

7. Instruksi *JNB (Jump if Not Bit)*

   Instruksi ini merupakan perintah untuk lompat ke alamat tertentu, jika pin yang dimaksud berlogika *Low* (0). Contoh,

```
Loop:
    JNB   P1.0,Loop
    .................
```

8. Instruksi *CJNE (Compare Jump if Not Equal)*

Instruksi ini berfungsi untuk membandingkan nilai dalam suatu register dengan suatu nilai tertentu. Contoh,

Loop:

.....

CJNE R0,#20h,Loop

.....

Jika nilai R0 tidak sama dengan 20h, maka program akan lompat ke rutin *Loop*. Jika nilai R0 sama dengan 20h,maka program akan melanjutkan instruksi selanjutnya..

9. Instruksi *DEC (Decreament)*

Instruksi ini merupakan perintah untuk mengurangi nilai register yang dimaksud dengan 1. Contoh,

MOV R0,#20h R0 = 20h

.....

DEC R0 R0 = R0 – 1

.....

10. Instruksi *INC (Increament)*

Instruksi ini merupakan perintah untuk menambahkan nilai register yang dimaksud dengan 1. Contoh,

MOV R0,#20h R0 = 20h

.....

INC R0 R0 = R0 + 1

.....

11. Dan lain sebagainya