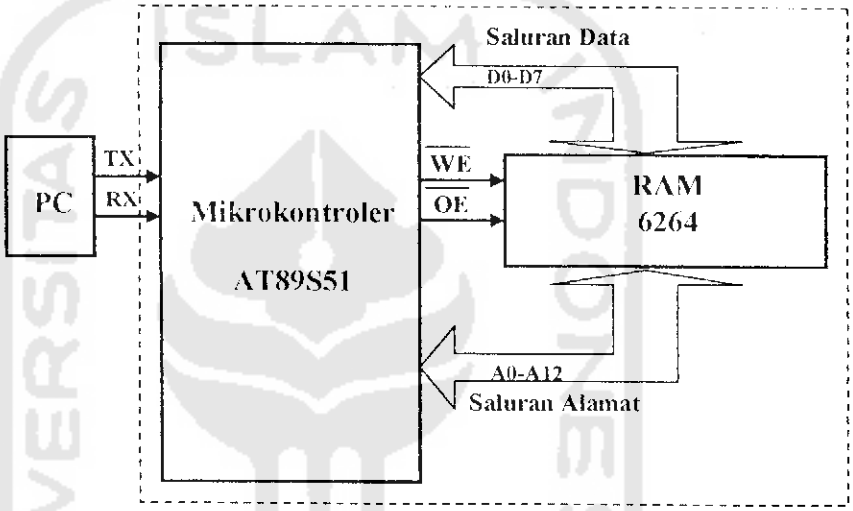
BAB III

PERANCANGAN SISTEM

3.1. Diagram Blok Sistem

Secara umum diagram blok *emulator* mikrokontroler AT89S51 seperti pada Gambar 3.1 berikut :



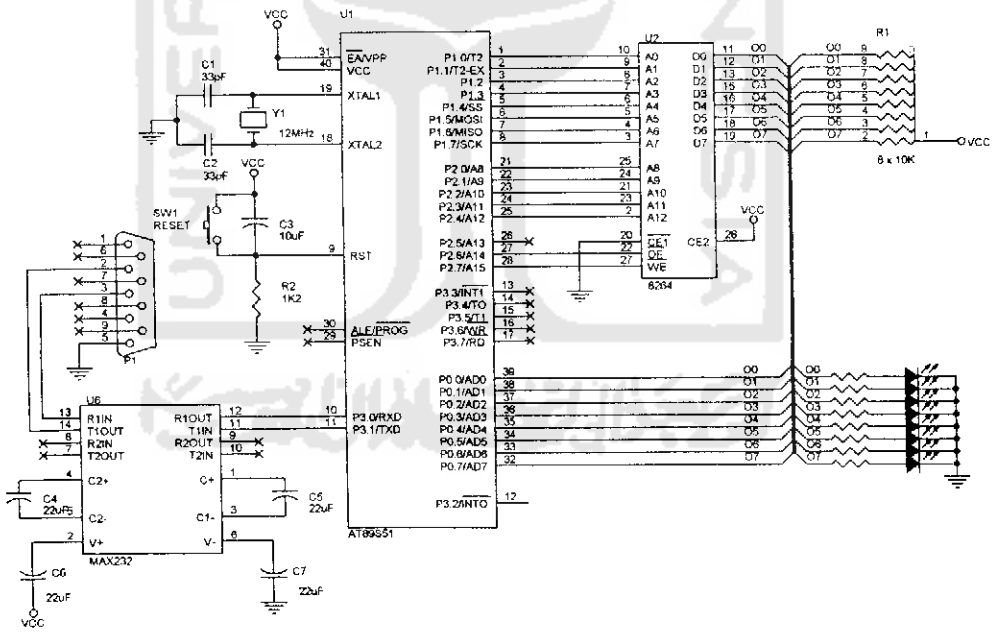
Gambar 3. 1. Blok diagram *emulator* mikrokontroler AT89S51

Pada dasarnya *emulator* mikrokontroler AT89S51 adalah sistem memori yang dibangun dengan RAM (*Random Access Memory*) yang mempunyai 2 mode kerja, mode pertama adalah pengisian memori dan mode kedua adalah mode pemakaian memori. Pada rancang bangun program emulator mikrokontroler AT89S51 program dirancang di komputer, pada saat akan diuji coba program tersebut dikirim ke *emulator* dan selanjutnya diisikan ke RAM. Untuk keperluan itu pada mode pertama, saluran data (*data bus*) D0-D7, saluran alamat (*address bus*) A0-A12 dan WE (*Write Enable)* dari RAM dikendalikan oleh *port* paralel mikrokontroler pengendali *emulator*.

Pada mode pengisian memori data diambil dari komputer oleh mikrokontroler kemudian dengan mengaktifkan pin WE data dituliskan ke RAM melalui saluran data. Sedangkan pada mode pemakaian memori, mikrokontroler mengambil data dari RAM kemudian menjalankannya dan menampilkan data tersebut kembali ke komputer.

3.2. Perancangan Perangkat Keras

Rangkaian lengkap *emulator* mikrokontroler AT89S51 terlihat pada Gambar 3.2. Dalam gambar tersebut AT89S51 (U1) berfungsi menerima data dari PC lewat saluran RS232, selanjutnya mengisikan data tersebut ke RAM 6264 (U2). Mikrokontroler AT89S51 juga berfungsi sebagai prosesor pemakai RAM



Gambar 3. 2. Rangkaian *emulator* mikrokontroler AT89S51

C1, C2 dan Kristal Y1 membentuk rangkaian osilator mikrokontroler AT89S51. C3 dan R2 membentuk rangkaian *power on reset*. Kedua rangkaian ini

merupakan rangkaian sistem minimum dan selalu dijumpai dalam semua rangkaian dengan mikrokontroler MCS51.

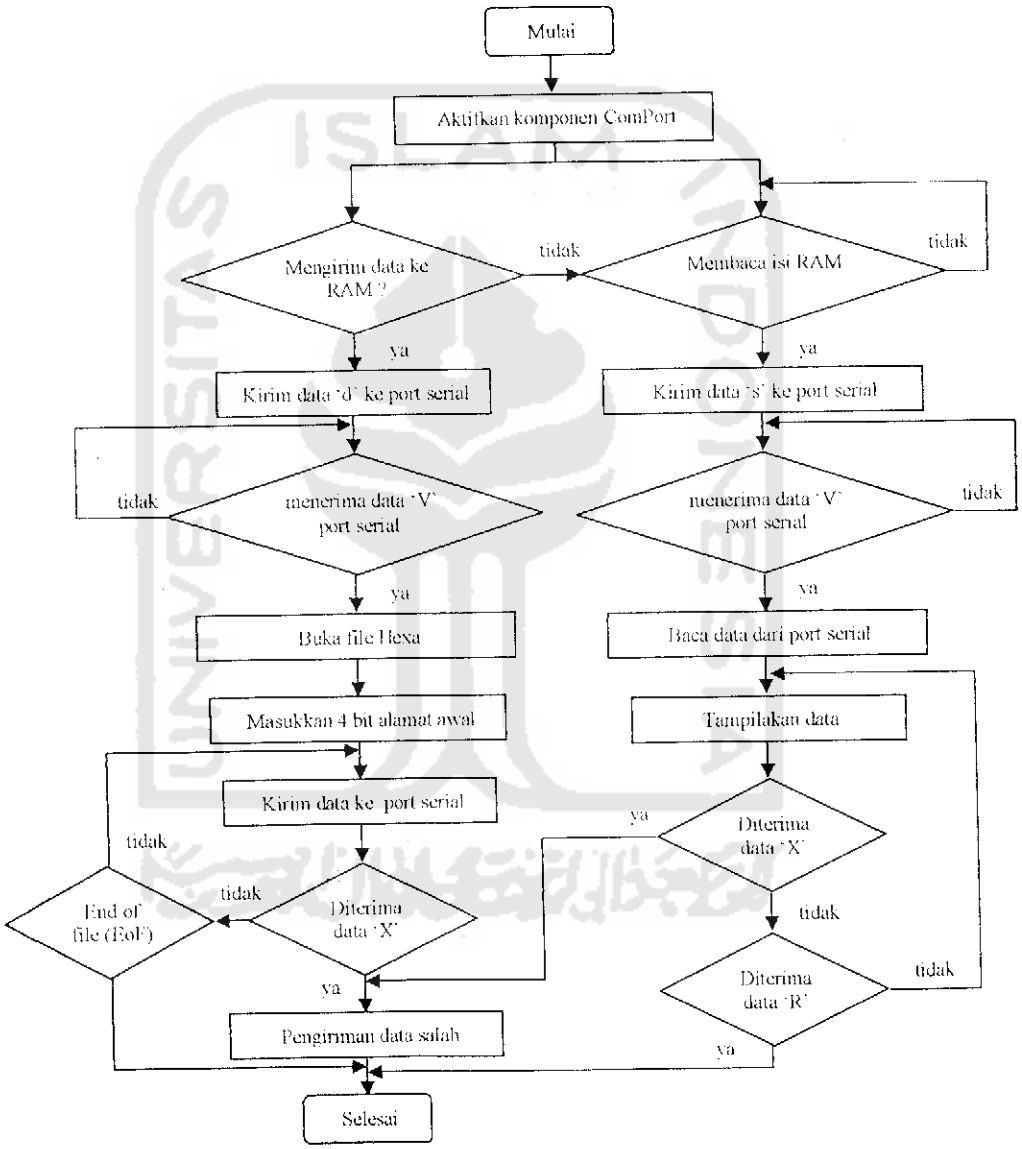
Komunikasi ke PC dilakukan dengan menggunakan sarana UART yang ada di dalam AT89S51, sinyal UART tersebut dikirim lewat TXD (kaki nomor 11) dan diterima di RXD (kaki nomor 10). Kedua kaki tersebut bekerja dengan level tegangan TTL, sedangkan *serial port PC* bekerja dengan level tegangan RS232, agar kedua sistem ini bisa dihubungkan secara elektronik, diperlukan rangkaian tambahan untuk menyesuaikan level tegangan. IC MAX232 berikut dengan kapasitor C4, C5, C6 dan C7 bertugas menyesuaikan kedua level tegangan yang berbeda tersebut, dengan demikian alat ini bisa langsung dihubungkan ke *port* seri PC. Konektor P1 merupakan konektor DB9, konektor standar RS232, seperti yang biasa dipakai untuk mouse dari PC.

Hubungan *port-port* AT89C51 dengan pin-pin RAM 6264 menggunakan sistem data bus yang mana saluran data RAM (D0 - D7) langsung dihubungkan ke port 0 yang merupakan saluaran data bus AT89S51 dan saluran alamat RAM (A0 – A12) dihubungkan ke P1.0 – P1.7 dan P2.0 – P2.4 yang merupakan saluran alamat mikrokontroler AT89S51. Proses pengisian dan pembacaan RAM diatur melalui pin OE dan WE RAM yang masing-masing dihubungkan ke P2.6 dan P2.7 mikrokontroler AT89S51. Jika OE diberi logika "0" maka data keluaran dari RAM dapat diambil melalui saluran data sedangkan jika pin WE diaktifkan dengan memberi logika '0' maka data dari mikrokontroler disimpan ke RAM.

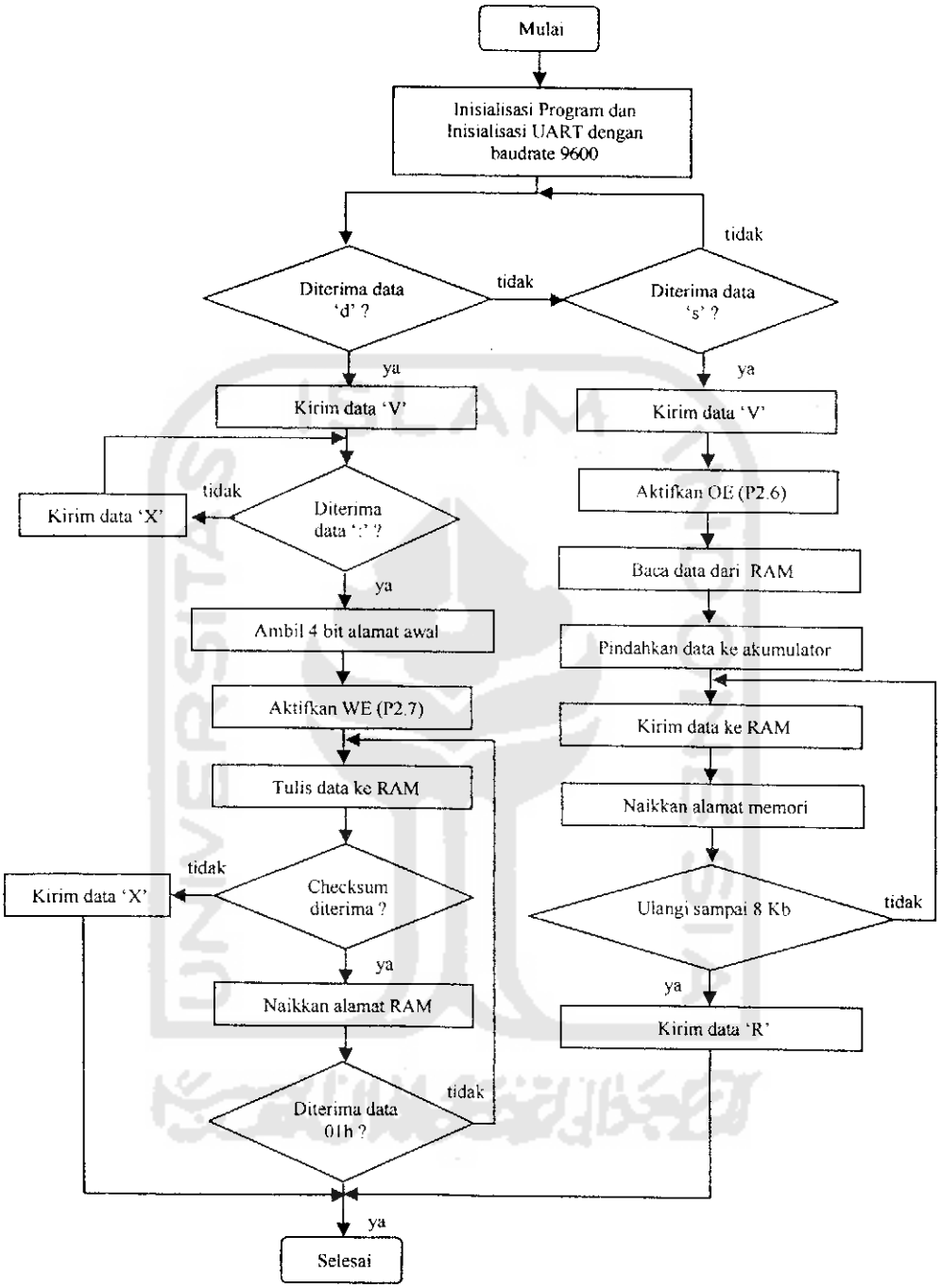
3.2. Perancanggan Perangkat Lunak

3.2.1. Diagram alir sistem

Diagram alir program Delphi ditunjukkan pada Gambar 3.3 dan diagram alir program *assembler* ditunjukkan pada Gambar 3.4 berikut



Gambar 3. 3. Diagram alir program pengiriman dan pembacaan data oleh PC



Gambar 3. 4. Diagram alir program pengiriman dan pembacaan data oleh mikrokontroler

Pada diagram alir program *emulator* menggunakan Delphi seperti Gambar 3.3 diatas, pemilihan mode kerja *emulator* dilakukan setelah mengaktifkan komponen *comport* yang merupakan komponen untuk mengatur komunikasi serial PC dengan piranti luar seperti mikrokontroler.

Ada 2 mode kerja yang dimiliki oleh *emulator* yaitu mode untuk menyimpan data ke RAM yang ditandai dengan pengiriman data ASCII 'd' ke port serial dan mode pemakaian memori yang ditandai dengan pengiriman data ASCII 's' ke port serial. Setelah pemilihan mode kerja, kemudian komputer akan menunggu penerimaan data 'V' dari mikrokontroler melalui port serial. Jika data 'V' tidak diterima berarti komunikasi antara PC dengan mikrokontroler belum tersambung.

Pada mode penyimpanan data ke RAM, setelah data 'V' diterima, program akan meminta file hexa dan alamat awal dalam hal ini adalah 0000. setelah itu kemudian program akan mengirimkan data dalam format hexa satu persatu sampai data terakhir yang disebut EoF (*End of File*).

Jika pada proses pengiriman data komputer menerima data 'X' dari mikrokontroler berarti pengiriman data salah dan proses pengiriman data dihentikan.

Sedangkan pada mode pemakaian memori setelah data 'V' diterima, data langsung dibaca dari mikrokontroler dan kemudian ditampilkan ke monitor PC. Pembacaan data terus dilakukan berulang-ulang sampai diterima data 'R' yang dikirim dari mikrokontroler. Sama seperti pada proses pengiriman data jika diterima data 'X' berarti pembacaan data salah.

Pada Gambar 3.4 diatas dapat dilihat bahwa jika data yang diterima dari PC adalah data 'd' berarti mode penyimpanan data ke RAM aktif sedangkan jika data yang diterima 's' maka mode pemakaian memori yang aktif

Pada saat penyimpanan data ke RAM pertama kali mikrokontroler menunggu data ':', jika data ':' tidak diterima, (data ':' merupakan data pertama dalam file dengan format heksa intel) berarti data yang dikirim salah atau bukan file heksa. Kemudian mikrokontroler akan mengirimkan data 'X' ke komputer. Setelah data ':' diterima maka data selanjutnya adalah alamat, setelah alamatnya tersimpan maka baca data selanjutnnya jika data yang diterima adalah 00h berarti data selanjutnya adalah data heksa. Untuk melakukan penyimpanan data ke RAM terlebih dulu diaktifkan pin WE RAM (P2.7 dibuat rendah '0') baru kemudian datanya dapat dituliskan ke RAM sesuai dengan alamat yang sudah ditentukan.

Pengiriman data dilakukan perbaris, setiap akhir baris merupakan data checksum jika data *checksum* yang diterima tidak sama dengan kebalikan (*inverter*) dari penjumlahan data yang diterima setelah tanda ':' berarti penyimpanan data salah dan mikrokontroler mengirimkan data 'X' ke komputer. Proses penyimpanan data selesai jika data yang diterima adalah 01 yang merupakan data Eof (*End of file*) yaitu akhir dari data.

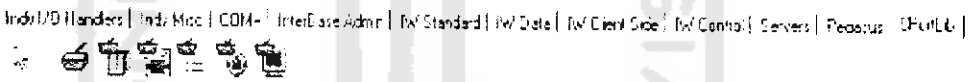
Sedangkan pada saat pemakaian memori data diambil dari RAM oleh mikrokontroler dengan mengaktifkan pin OE. Setelah pin OE aktifkan (P2.6 dibuat rendah '0') maka data dari Ram dapat dibaca oleh mikrokontroler. Pembacaan data akan berhenti setelah membaca 8 kbyte data dan kemudian mikrokontroler akan mengirimkan data 'R' ke PC.

3.2.2. Inisialisasi Port Serial pada Delphi 7

Untuk dapat mengakses perangkat lunak melalui port serial harus diketahui terlebih dahulu konfigurasi dari perangkat luar tersebut antara lain:

1. Port yang akan digunakan, komputer yang sekarang terdapat 2 port serial (Com1 dan Com2).
2. *Baud rate*, kecepatan data yang akan digunakan
3. Data bit, bit data
4. Stop bit, bit untuk memisahkan data
5. Parity, bit untuk validasi data yang telah dikirim
6. *Flow control*

Delphi tidak dapat langsung digunakan untuk mengakses data luar melalui port serial sehingga perlu ditambahkan komponen baru yaitu ComPort. Tampilan komponen ComPort seperti pada Gambar 3.5 berikut

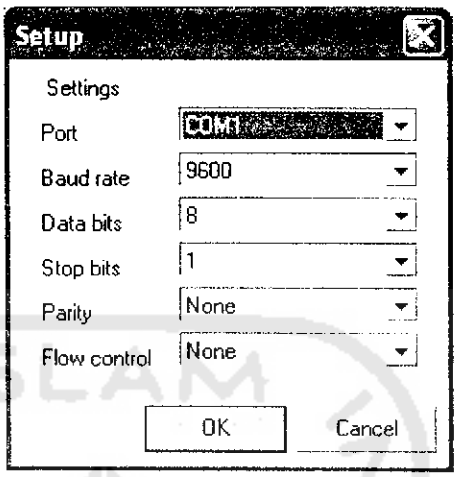


Gambar 3. 5. Tampilan komponen ComPort

Untuk mengaktifkan komponen comport terlebih dulu ditampilkan comport setup dialog untuk mengeset komunikasi serial yang diinginkan. Berikut contoh program untuk menampilkan comport setup dialog saat penekanan button1.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
    ComPort.ShowSetupDialog;
end;
```

Tampilan *comport* setup dialog seperti pada Gambar 3.6 berikut



Gambar 3. 6. Tampilan *comport* setup dialog

Untuk menerima data dari perangkat luar dapat diambil pada *event* RXChart seperti pada contoh program berikut :

```
procedure TFrmEmulator.ComPortRxChar(Sender: TObject; Count: Integer);
begin
if not pilihan then begin
 ComPort.ReadStr(Str, Count);
 Edit1.Text:=Str;
 If Str ='O' then begin
    MessageDlg ('Transfer Selesai.', mtInformation, [mbOK], 0);
    TabSheet1.Show;
 end;
 if Str='V' then begin
 end;
 if Str='X' then begin
    MessageDlg ('Transfer gagal, Reset mikro dan coba lagi.', mtInformation,
[mbOK], 0);
    application.Terminate;
  End;
End;
```

Program diatas adalah kejadian (*Event*) saat menerima data dari perangkat luar. Jika data yang diterima adalah 'O' maka ditampilkan informasi 'Transfer selesai', jika data yang diterima adalah data 'X' maka ditampilkan informasi 'Transfer gagal, Reset mikro dan coba lagi'.

3.2.3. Inisialisasi komunikasi serial pada AT89S51

Listing program berikut ini merupakan contoh program untuk inisialisasi serial

Init_serial:

```
MOV   SCON,#52H    ; Mode 1, Receive Enable aktif
MOV   TMOD,#20H    ; Timer 1 Mode 2, Timer0 mode 1
MOV   TH1,#0FDH    ; Baudrate 9600 bps
setb  TR1          ; aktifkan timer1
Clr   RI           ; Hapus Flag Receive
Clr   TI           ; Hapus Flag Transmitter
ret
```

Dari program di atas dapat dilihat bahwa untuk inisialisai serial diperlukan beberapa register seperti SCON, TMOD dan TH1. Register SCON digunakan untuk menentukan mode serial yang digunakan jika register SCON diisi data 52H berarti komunikasi serial yang digunakan adalah mode 1 (UART 8 bit) dengan *Receive Enable aktif.* Sedangkan register TMOD dan TH1 digunakan untuk menghitung *baudrate* (kecepatan pengiriman data).

Dengan persamaan 2.2 pada bab II, jika diinginkan kecepatan pengiriman datanya sebesar 9600 bps dan osilator kristal yang digunakan adalah 11,0592 MHz maka nilai TH1nya adalah

$$TH1 = 256 - \frac{K \times f_{osilator}}{12 \times 32 \times Boudrate}$$

$$TH1 = 256 - \frac{1 \times 11{,}0592.10^{6}}{12 \times 32 \times 9600}$$

$$TH1 = 256 - 3 = 253\,(FDH)$$

3.2.4. Program assembler membaca data dari RAM

Listing program berikut ini adalah program untuk membaca data dari RAM melalui jalur data bus pada *port* 0 AT89S51 dan jalur alamat A0 – A12.

```
Read:
    MOV   P1,DPL          ;alamat nibble rendah dikirim ke P1
    MOV   B,DPH
    ANL   B,#$1F
    ORL   B,#$C0          ;WE high, OE high, CS low
    MOV   P2,B            ;A8-A12 adalah jalur alamat nible tinggi
    CLR   P2.6            ;Enable-kan OE
    NOP
    MOV   A,P0            ;Simpan data di Akumulator
    SETB  P2.6            ;Disable OE
    ACALL SerialOut       ;kirim data ke port serial
    MOV   A,DPH           ;Check 8 KB
    CJNE  A,#$1F,NextRD
    MOV   A,DPL
    CJNE  A,#$FF,NextRD
    LJMP  End
NextRD:                   ;alamat selanjutnya
    INC   DPTR
    SJMP  Read
```

Dari listing program diatas dapat dilihat bahwa jalur alamat A0 sampai A7 dihubungkan ke P1 dan alamat A8 sampai A12 dihubungkan ke P2.0 sampai P2.4.

untuk memulai pembacaan data dari RAM sebelumnya pin OE dari RAM harus di nolkan dulu dengan mengenolkan P2.6 karena pin OE aktif *low*. Setelah pin OE di aktifkan maka data dapat diambil dari jalur data *port* 0 dan disimpan pada akumulator. Ulangi proses tersebut sampai semua data pada RAM terbaca.

Kapasitas RAM 6264 dapat diketahui dari jumlah alamatnya. RAM 6264 memiliki alamat 13 bit yaitu A0 sampai A12 dengan demikian kapasitasnya sebesar 2^n, dimana n adalah jumlah bit alamat sehingga jika jumlah bit alamatnya = 13 bit berarti kapasitas RAMnya adalah 2^{13} = 8192 byte = 8 Kbyte.

3.2.5. Program assembler menulis data ke RAM

Listing program berikut ini adalah program untuk menulis data ke RAM data ditulis melalui jalur data bus pada *port* 0 AT89S51 dan jalur alamat A0 – A12.

```
SaveData:
    MOV   P1,DPL          ;alamat nibble rendah dikirim ke P1
    MOV   B,DPH
    ANL   B,#$1F
    ORL   B,#$C0          ;WE high, OE high, CS low
    MOV   P2,B            ;A8-A12 adalah jalur alamat nible tinggi
    MOV   P0,A            ;kirimkan data ke P0
    NOP
    CLR   P2.7            ;Enable WE
    NOP
    SETB  P2.7            ;Disable WE
    INC   DPTR            ;alamat selanjutnya
    DJNZ  R0,SaveData     ;ulangi sampai R0=0
```

Sama seperti pada pembacaan data dari RAM jalur data pada *port* 0 dan jalur alamat pada *port* 1 dan P2.0 sampai P2.4. Untuk mulai menuliskan data ke RAM terlebih dulu P2.7 dibuat berlogika rendah '0' untuk membuat pin WE dari RAM aktif, sehingga data dapat dituliskan ke RAM lewat jalur data *port* 0. ulangi proses tersebut sampai sampai R0= 0. R0 adalah register yang digunakan untuk menyimpan kapasitas data.