Bahasa Pemrograman Assembler 8051

A. Struktur Bahasa Assembler

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 bias dijalankan (misalnya MOV). Pengarah *assembler* (*assembler directive*) merupakan instruksi ke program *assembler* yang mendefinisikan struktur program, simbol-simbol, data, konstanta dan lain-lain.(misalnya :ORG). adapun komentar perlu dituliskan agar program mudah dibaca.

Baris-baris program yang mengandung instruksi mesin atau pengarah *assembler* harus mengikuti aturan program *assembler* ASM51, ataupun compiler lainnya, seperti *macro assembler*, dan lain-lain. Masing-masing baris terdiri atas beberapa *field* yang dipisahkan dengan spasi atau tabulasi. Adapun format umumnya sebagai berikut :

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

Label

Sebuah label mewakili sebuah alamat dari instruksi (atau data) yang mengikat. Label ini digunakan sebagai sebagai operan pada instruksi - instruksi percabangan (misalnya SJMP Lanjut). Dalam mikrokontroller MCS51 terdapat perbedaan antara simbol dan label. Apabila label menggunakan titik dua ‘ : ‘, sedangkan simbol tidak..

Var EQU 200 *; “Var” adalah suatu symbol dari nilai 200*
Mulai : MOV P0,#0FEH *; “mulai” adalah label yang menunjuk pada lokasi instruksi MOV.*

Mnemonik

Mnemonik instruksi atau pengarah *assembler* dimasukkan ke dalam *“mnemonic field”* yang mengikuti label *mnemonic*. *Mnemonic* instruksi misalnya DD, MOV, INC dan lain-lain. Sedangkan pengarah *assembler*, misalnya ORG, EQU, DB dan lain – lain.
Adapun pengelompokannya ialah sebagai berikut :

1) Kontrol kondisi *assembler*

2) Definisi simbol

3) Pemesanan inisialisasi penyimpan

4) Rantai (*linkage*) program

5) Pemilihan segmen

contoh penggunaanya, misalnya :

ORG 100H ; *awal program di 100h / set nilai PC*

END ; pernyataan paling akhir dari program

Operan

Operan ditulis setelah mnemonik, bisa berupa alamat atau data yang digunakan instruksi yang bersangkutan. Bisa juga berupa label yang mewakili alamat suatu data atau berupa simbol yang mewakili suatu data konstanta. Operan dapat bersifat sebagai tujuan (*destination*) dari hasil operasi, maupun sumber (*source*). Contohnya yakni :

MOV A,#255

MOV A,@R0

ADD A,@R0

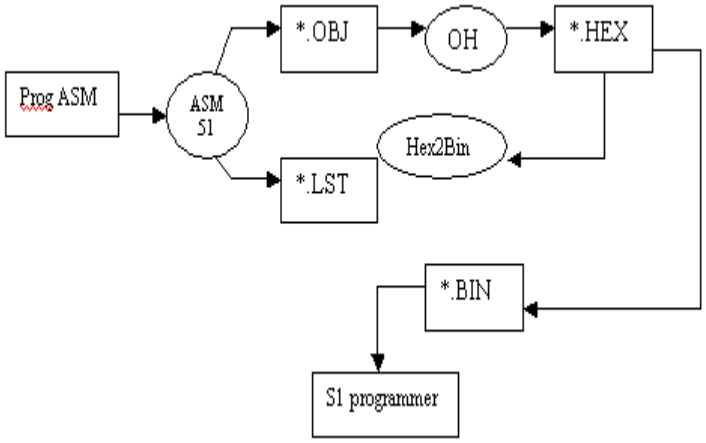
Ketiga instruksi diatas melibatkan dua buah operan yang masing-masing berbeda fungsinya, ada yang berupa nilai (#255), ada juga yang berupa pengalamatan tak langsung (*immediate addressing)* , serta *register* tujuan *(destination).*

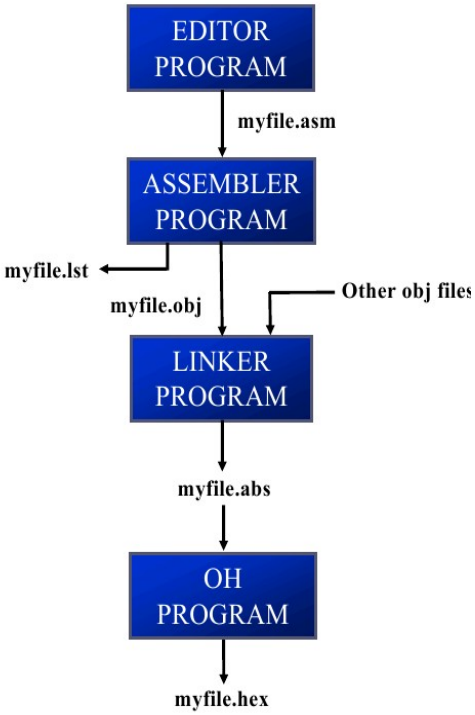
Komentar

Komentar harus diawali dengan titik koma (;). Sub-rutin dari program besar yang mengerjakan suatu operasi biasanya diawali dengan blok komentar yang menjelaskan fungsi sub-rutin atau bagian dasar program tersebut.

B. Operasi Assembler MCS51

Untuk melakukan simulasi program MCS51 dilalui melalui beberapa tahap, dimulai dari perancangan *script* pada *text editor*, compilasi, simulasi dan *upload* ke *chip*. Proses tersebut menjadi suatu prosedur standar dalam perancangan program aplikasi *Assembler* khususnya dalam ruang lingkup pemrograman mikrokontroller MCS51.





Setelah user melakukan pembuatan program pada text editor, maka compiler akan menghasilkan objek file dan file listing, yang memuat secara detil alokasi / penempatan tiap instruksi di memori, compiler melalui listing file (*.LST) akan menunjukkan alokasi pada ROM dari setiap instruksi yang dibuat. Baik itu opcode maupun address dari setiap baris nstruksi. B erikut ialah contoh dari format listing file yang dihasilkan compiler :

```
1 0000          ORG 0H          ;start (origin) at 0
2 0000  7D25    MOV R5,#25H     ;load 25H into R5
3 0002  7F34    MOV R7,#34H     ;load 34H into R7
4 0004  7400    MOV A,#0        ;load 0 into A
5 0006  2D      ADD A,R5        ;add contents of R5 to A
                                ;now A = A + R5
6 0007  2F      ADD A,R7        ;add contents of R7 to A
                                ;now A = A + R7
7 0008  2412    ADD A,#12H      ;add to A value 12H
                                ;now A = A + 12H
8 000A  80EF HERE: SJMP HERE;stay in this loop
9 000C          END             ;end of asm source file
```

ROM Address	Machine Language	Assembly Language
0000	7D25	MOV R5, #25H
0002	7F34	MOV R7, #34H
0004	7400	MOV A, #0
0006	2D	ADD A, R5
0007	2F	ADD A, R7
0008	2412	ADD A, #12H
000A	80EF	HERE: SJMP HERE

dalam 8051 instruction set in numerical order. Dapat kita ketahui bahwa masing-masing format instruksi memiliki nilai konversi opcode yang berbeda. Sebagaimana contoh uraian diatasdapat kita simpulkan bahwa perubahan ROM address dipengaruhi oleh jumlah opcode (*machine language)* dari tiap baris instruksi.

Sedangkan setelah masuk kedalam chip (ROM), maka kode heksa hasil konversi program assembler yang dibuat akan tersusun sebagai berikut :

Address	Code
0000	7D
0001	25
0002	7F
0003	34
0004	74
0005	00
0006	2D
0007	2F
0008	24
0009	12
000A	80
000B	FE

adapun format kompilasi dari 8051 compiler seperti (macro assembler), ialah sebagai berikut :

Kompilasi Program Assembler MCS51

Program simulator ini terkoneksikan dengan *compiler Macro Assembler,* yang akan melakukan proses kompilasi program Assembler yang telah dibuat di teks editor hingga menjadi 3 jenis file dengan ekstensi berbeda, diantaranya ialah :

1. File Heksadesimal (*.Hex), merupakan file yang akan di *upload* ke chip mikrokontroler.

2. File listing program (*.Lst), merupakan kumpulan listing program yang dilengkapi dengan informasi tentang kode program yang dibuat, seperti jumlah *error* pada program, penggunaan variable memori serta pengaksesan ke alamat memori. dalam file listing program akan ditampilkan lokasi alamat dari tiap instruksi beserta kode operannya. Adapun lokasi memori dari tiap instruksi merupakan perubahan nilai dari PC (*program counter)* dari instruksi sebelumnya.

3. File objek (*.obj) merupakan, kumpulan operan dari tiap baris instruksi.

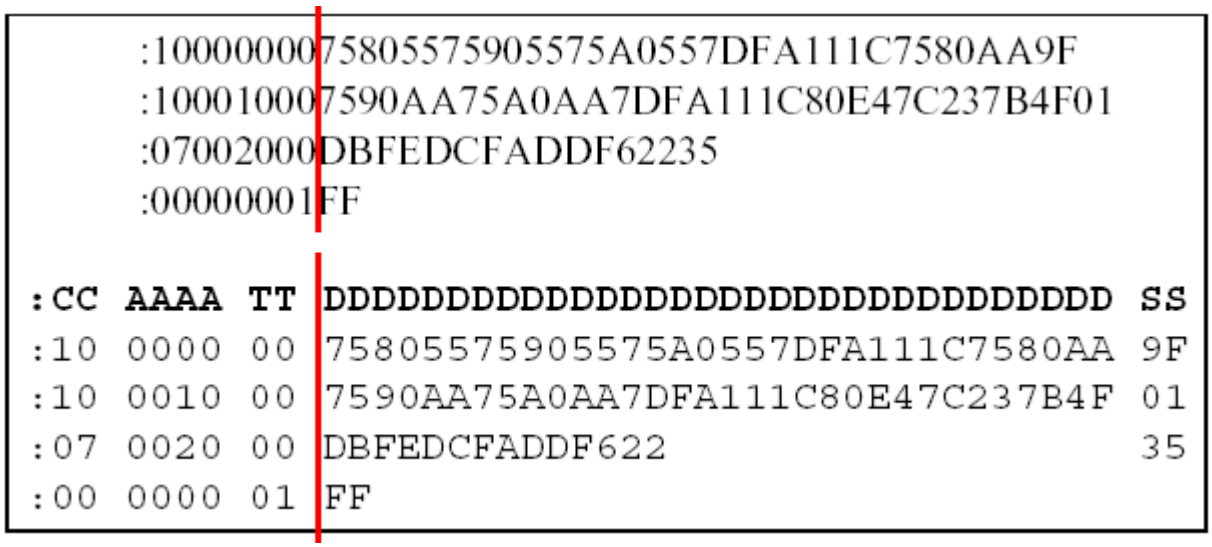
Adapun proses yang dilakukan dalam proses kompilasi program hingga akhirnya akan menghasilkan *hex file* yang akan di *upload* ke chip. Misalkan instruksi yang akan di *compile* ialah sebagai berikut :

```
LOC   OBJ        LINE
0000             1 ORG 0H
0000 758055      2 MAIN: MOV P0,#55H
0003 759055      3 MOV P1,#55H
0006 75A055      4 MOV P2,#55H
0009 7DFA        5 MOV R5,#250
000B 111C        6 ACALL MSDELAY
000D 7580AA      7 MOV P0,#0AAH
0010 7590AA      8 MOV P1,#0AAH
0013 75A0AA      9 MOV P2,#0AAH
0016 7DFA        10 MOV R5,#250
0018 111C        11 ACALL MSDELAY
001A 80E4        12 SJMP MAIN
                 13 ;250 ms delay
                 14 MSDELAY:
001C 7C23        15 TIGA: MOV R4,#35
001E 7B4F        16 DUA: MOV R3,#79
0020 DBFE        17 SATU: DJNZ R3,SATU
0022 DCFA        18 DJNZ R4,DUA
```

```
0024 DDF6        19 DJNZ R5,TIGA
0026 22          20 RET
                 21 END
```

File heksadesimal, biasanya terdiri atas beberapa penysun diantaranya :

1. Informasi nomor *byte* dari operan yang ada pada *disasembler code.*
2. Alamat awal, yang mendefinisikan penempatan dari informasi nomor *byte* maupun konversi heksadesimal dari *mnemonic* dan operan instruksi-nya.
4. berikut ini ialah hasil dari kompilasi kode program diatas :



hasil kompilasi file dibagi ke dalam 4 bagian strukturan penyusunnya, diantaranya ialah :

1. Jumlah *byte operan* pada rentang alamat memori tertentu sebanyak 0-16 kumpulan bilangan 8 *bit*, dilambangkan dengan CC
2. *Range address* atau lokasi memori dari tiap baris instruksi , dilambangkan dengan AAAA
3. Apabila setelah 16 operan 8 bit pertama ditampilkan, masih terdapat operan yang akan di tampilkan, maka nilainya = 00, sedangkan apabila sudah tidak ada operan 8 bit yang ditampilkan pada baris berikutnya nilainya = 01 dilambangkan dengan TT
4. File 8 bit operan yang merupakan konversi dari tiap instruksi, misalnya : MOV P1,#55H, maka konversi heksadesimal-nya ialah sebagai berikut 759055, 75 menunjukan tipe pengalamatan atau operasi, 90 menunjukkan alamat dari P1 sebagai *destination*, sedangkan 55 menunjukan nilai (*source)* yang disalin ke P1. bagian ini disimbolkan dengan DDDDDDDDDDDDDDD

5. Bagian ini merupakan *checksum* dari 4 bagian sebelumnya (CC, AAAA, TT,DDDDDDDDD), misalkan pada baris pertama hasil kompilasi program ialah sebagai berikut : 10000000758055759055 75A0557DFA111C7580AA, maka pengecekkan dilakukan dengan

melakukan penjumlahan per tiap 8 bit bilangan, yakni :

10+00+00+00+75+80+55+75+90+55+75+A0+55+7D+FA+11+1C+75+80+AA = 761H, *carry* 7 dihilangkan. Sehingga menjadi 61H, kemudian dilakukan komplemen2 maka hasilnya ialah sebagai berikut :

FFH

61H

_____ -

9EH

1 H

_____ +

9FH