MODUL 4
Menggunakan Bahasa Pemrograman Assembly

BAGIAN 1
Menggunakan Bahasa Pemrograman Assembly

Tujuan Pembelajaran Umum:

1. Mahasiswa trampil menggunakan bahasa pemrograman assembly

Tujuan Pembelajaran Khusus:

1. Mahasiswa memahami konstruksi program assembly
2. Mahasiswa memahami proses kerja assembly
3. Mahasiswa dapat menggunakan berbagai jenis mnemonik
4. Mahasiswa dapat menggunakan assembler directive

Bahasa Assembly adalah bahasa komputer yang kedudukannya di antara bahasa mesin dan bahasa level tinggi misalnya bahasa C, C++, Pascal, Turbo Basic, Java, dan sebagainya. Bahasa C atau Pascal dikatakan sebagai bahasa level tinggi karena memakai kata-kata dan pernyataan yang mudah dimengerti manusia, meskipun masih jauh berbeda dengan bahasa manusia sesungguhnya. Assembler adalah program yang bekerja membantu penulisan instruksi dalam format bahasa inggris sehingga mudah dibaca dan dipahami.

```
MOV R0, #02h
MOV A, #03h
ADD A, R0
```

Perintah baris pertama bekerja menjalankan proses pengisian register R0 dengan data 02h. Perintah baris kedua bekerja menjalankan proses pengisian register A dengan data 03h. Kemudian proses penjumlahan data pada register A dengan data pada register R0 dijalankan menggunakan perintah ADD A,R0 dan menghasilkan data 05h tersimpan di register A.

Perintah MOV dan ADD adalah mnemonik atau singkatan dari perintah MOVE dan ADD. Mnemonik dari perintah lainnya dapat dirangkum dalam tabel 8 berikut.

Tabel 8 Mnemonik Perintah Assembly AT89S51

No	PERINTAH	MNEMONIK
1.	ADD	**ADD**
2.	ADD WITH CARRY	**ADC**
3.	SUB WITH BORROW	**SBB**
4.	INCREMENT	**INC**
5.	DECREMENT	**DEC**
6.	MULTIPLY	**MUL**
7.	DEVIDE	**DIV**
8.	AND LOGIC	**ANL**
9.	OR LOGIC	**ORL**
10.	EXLUSIVE OR LOGIK	**XRL**
11.	DECIMAL ADJUST ACCUMULATOR	**DAA**
12.	CLEAR ACCUMULATOR	**CLR A**
13.	COMPLEMENT ACCUMULATOR	**CPL A**
14.	ROTATE ACCUMULATOR LEFT	**RLA**
15.	ROTATE ACCUMULATOR LEFT THROUGH CARRY	**RLCA**
16.	ROTATE ACCUMULATOR RIGHT	**RRA**
17.	ROTATE ACCUMULATOR RIGHT THROUGH CARRY	**RRCA**
18.	SWAPP NIBBLE WITHIN ACCUMULATOR	**SWAP**
19.	PUSH DIRECT BYTE KE STACK	**PUSH**
20.	POP DIRECT BYTE DARI STACK	**POP**
21.	JUMP IF CARRY SET C=1	**JC**
22.	JUMP IF CARRY NOT SET C = 0	**JNC**
23.	JUMP IF DIRECT BIT SET	**JB**
24.	JUMP IF DIRECT BIT NOT SET	**JNB**
25.	JUMP IF DIRECT BIT SET & CLEAR BIT	**JBC**
26.	ABSOLUTE CALL	**ACALL**
27.	LONG CALL	**LCALL**
28.	RETURN	**RET**
29.	RETURN FROM INTERRUPT	**RETI**
30.	ABSOLUTE JUMP	**AJMP**
31.	LONG JUMP	**LJMP**
32.	SHORT JUMP	**SJMP**
33.	JUMP INDIRECT	**JMP**
34.	JUMP IF ACCUMULATOR ZERRO	**JZ**
35.	JUMP IF ACCUMULATOT NOT ZERRO	**JNZ**
36.	COMPARE AND JUMP IF NOT EQUAL	**CJNE**
37.	DECREAMENT AND JUMP IF NOT ZERO	**DJNZ**
38.	NO OPERATION	**NOP**

MODUL 4
Menggunakan Bahasa Pemrograman Assembly

Bahasa mesin adalah kumpulan kode biner yang merupakan instruksi yang bisa dijalankan oleh komputer. Di dalam mikrokontroler instruksi disimpan dalam kode heksa sehingga sulit dibaca dan dipahami maknanya. Sedangkan bahasa assembly memakai kode mnemonik untuk menggantikan kode biner, agar lebih mudah diingat sehingga lebih memudahkan dalam penulisan program.

No	OPERATION CODE	ASSEMBLY
1.	78 02	MOV R0, #02h
2.	74 03	MOV A, #03h
3.	28	ADD A, R0

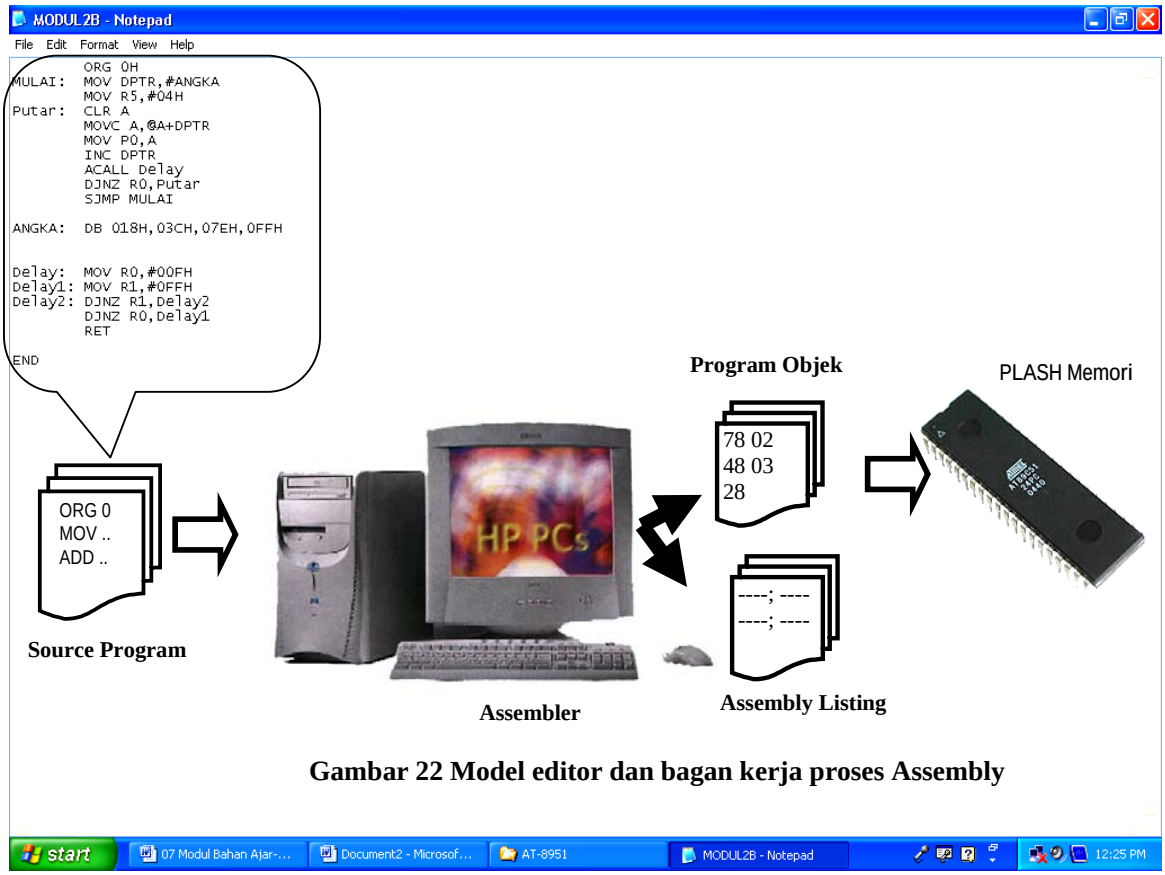
Kode bahasa mesin atau sering disebut dengan operation code dari perintah MOV R0,#02h adalah 78 02. Untuk MOV A,#03h kode operasinya dalah 74 03 dan 28 adalah kode operasi dari perintah ADD A, R0. Kode operasi untuk setiap perintah dapat dibaca pada lembar instruction set.

Program yang ditulis dengan bahasa assembly terdiri dari label; kode mnemonik, operand 1, operand 2, keterangan, dan lain sebagainya. Program ini disebut sebagai program sumber (Source Code). Source code belum bisa diterapkan langsung pada prosesor untuk dijalankan sebagai program. Source code harus diterjemahkan dulu menjadi bahasa mesin dalam bentuk kode biner atau operasi.

Source code ditulis dengan program editor biasa, misalnya Note Pad pada Windows atau SideKick pada DOS, TV demo, lalu source code diterjemahkan ke bahasa mesin dengan menggunakan program Assembler. Proses menterjemahkan source code menjadi bahasa mesin disebut dengan proses *assembled.* Hasil kerja program Assembler adalah “program objek” dan juga “assemly listing”.

Program Objek berisikan kode kode operasi bahasa mesin. Biasanya file program objek menguanakan ekstensi .HEX. Kode-kode operasi bahasa mesin inilah yang dituliskan ke memori-program prosesor. Dalam dunia mikrokontroler biasanya program objek ini diisikan ke UV EPROMatau EEPROM dan khusus untuk mikrokontroler buatan Atmel, program ini diisikan ke dalam Flash PEROM yang ada di dalam chip mikrokontroler AT89S51 atau AT89C2051.

Assembly Listing merupakan naskah yang berasal dari program sumber, dalam naskah tersebut pada bagian sebelah setiap baris dari program sumber diberi tambahan hasil terjemahan program Assembler. Tambahan tersebut berupa nomor memori-program berikut dengan kode yang akan diisikan pada memori-program bersangkutan. Naskah ini sangat berguna untuk dokumentasi dan sarana untuk menelusuri program yang ditulis. Gambar 22 menunjukkan model editor program assembly dan bagan kerja proses assembly.



Gambar 22 Model editor dan bagan kerja proses Assembly

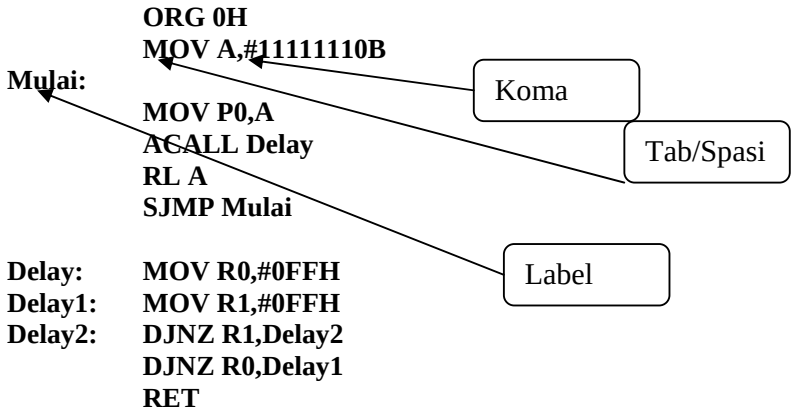
Yang perlu diperhatikan adalah setiap prosesor mempunyai konstruksi yang berlainan, instruksi untuk mengendalikan masing-masing prosesor juga berlainan, dengan demikian bahasa Assembly untuk masing-masing prosesor juga berlainan, yang sama hanyalah pola dasar cara penulisan program Assembly saja.

Konstruksi Program Assembly

Source program dalam bahasa Assembly menganut prinsip 1 baris untuk satu perintah tunggal. Setiap baris perintah tersebut bisa terdiri atas beberapa bagian (field), yakni bagian Label, bagian mnemonik, bagian operand yang bisa lebih dari satu dan terakhir bagian komentar. Untuk membedakan masing-masing bagian tersebut dibuat ketentuan sebagian berikut:

1. Masing-masing bagian dipisahkan dengan spasi atau TAB, khusus untuk operand yang lebih dari satu masing-masing operand dipisahkan dengan koma.

2. Bagian-bagian tersebut tidak harus semuanya ada dalam sebuah baris, jika ada satu bagian yang tidak ada maka spasi atau TAB sebagai pemisah bagian tetap harus ditulis.
3. Bagian Label ditulis mulai huruf pertama dari baris, jika baris bersangkutan tidak mengandung Label maka label tersebut digantikan dengan spasi atau TAB, yakni sebagai tanda pemisah antara bagian Label dan bagian mnemonik.



Label mewakili nomor memori-program dari instruksi pada baris bersangkutan, pada saat menulis instruksi JUMP, Label ini ditulis dalam bagian operand untuk menyatakan nomor memori-program yang dituju. Dengan demikian Label selalu mewakili nomor memori-program dan harus ditulis dibagian awal baris instruksi.

Disamping Label dikenal pula **Symbol**, yakni satu nama untuk mewakili satu nilai tertentu dan nilai yang diwakili bisa apa saja tidak harus nomor memori-program. Cara penulisan Symbol sama dengan cara penulisan Label, harus dimulai di huruf pertama dari baris instruksi.

Mnemonik (artinya sesuatu yang memudahkan diingat) merupakan singkatan perintah, dikenal dua macam mnemonik, yakni manemonic yang dipakai sebagai instruksi mengendalikan prosesor, misalnya **ADD**, **MOV**, **DJNZ** dan lain sebagainya. Ada pula mnemonik yang dipakai untuk mengatur kerja dari program Assembler misalnya **ORG**, **EQU** atau **DB**, mnemonik untuk mengatur kerja dari program Assembler ini dinamakan sebagai 'Assembler Directive'.

Operand adalah bagian yang letaknya di belakang bagian mnemonik, merupakan pelangkap bagi mnemonik. Kalau sebuah instruksi di-ibaratkan sebagai kalimat perintah,

maka mnemonik merupakan subjek (kata kerja) dan operand merupakan objek (kata benda) dari kalimat perintah tersebut.

Tergantung pada jenis instruksinya, operand bisa berupa berbagai macam hal. Pada instruksi **JUMP** operand berupa Label yang mewakili nomor memori-program yang dituju misalnya **LJMP** Start, pada instruksi untuk pemindahan/pengolahan data, operand bisa berupa Symbol yang mewakili data tersebut, misalnya **ADD A,#Offset**. Banyak instruksi yang operandnya adalah register dari prosesor, misalnya **MOV A,R1**. Bahkan ada pula instruksi yang tidak mempunyai operand, misalnya **RET**.

Komentar merupakan bagian yang sekedar sebagai catatan, tidak berpengaruh pada prosesor juga tidak berpengaruh pada kerja program Assembler, tapi bagian ini sangat penting untuk keperluan dokumentasi.

Assembler Directive

Seperti sudah dibahas di atas, bagian Mnemonik dari sebuah baris perintah bisa merupakan instruksi untuk prosesor, maupun berupa Assembler Directive untuk mengatur kerja dari program Assembler. Mnemonik untuk instruksi prosesor, sangat tergantung pada prosesor yang dipakai, sedangkan mnemonik untuk Assembler Directive tergantung pada program Assembler yang dipakai. Meskipun demikian, terdapat beberapa Assembler Directive yang umum, yang sama untuk banyak macam program Assembler.

Assembler Directive yang bersifat umum tersebut, antara lain adalah

ORG – singkatan dari *ORIGIN*, untuk menyatakan nomor memori yang dipakai setelah perintah itu, misalnya **ORG 0000h** maka memori berikutnya yang dipakai Assembler adalah **0000h**. **ORG** berlaku untuk memori program maupun memori-data. Dalam hal penomoran memori, dikenal tanda sebagai awalan untuk menyatakan nomor memori dari baris bersangkutan. Misalnya :

```
        ORG 0H
        MOV A,#11111110B
Mulai:
        MOV P0,A
```

EQU – singkatan dari *EQUATE*, dipakai untuk menentukan nilai sebuah Symbol.
Misalnya **Angka88 EQU 88** memberi nilai **88** pada Symbol **Angka88**, atau **CR EQU 0D** mempunyai makna kode ASCII dari **CR** (Caarriage Return) adalah **08**.

DB – singkatan dari *DEFINE BYTE*, dipakai untuk memberi nilai tertentu pada memori-program. Nilai tersebut merupakan nilai 1 byte, bisa berupa angka ataupun kode ASCII. **DB** merupakan Assembler Directive yang dipakai untuk membentuk teks maupun tabel.

DW – singkatan dari *DEFINE WORD*, dipakai untuk memberi nilai 2 byte ke *memori-program* pada baris bersangkutan. Assembler Directive ini biasa dipakai untuk membentuk suatu tabel yang isinya adalah nomor-nomor *memori-program*.

DS – singkatan dari *Define Storage*, Assembler Directive ini dipakai untuk membentuk variable. Sebagai variabel tentu saja memori yang dipakai adalah *memori-data* (RAM) bukan *memori-program* (ROM). Hal ini harus benar-benar dibedakan dengan Assembler Directive **DB** dan **DW** yang membentuk kode di *memori-program*. Dan karena **DS** bekerja di RAM, maka **DS** hanya sekedar menyediakan tempat di memori, tapi tidak mengisi nilai pada memori bersangkutan.(Budhy Sutanto)

.

BAGIAN 2
PETUNJUK KERJA

A. PETUNJUK PRE-TEST

1. Kerjakan soal dan latihan pre-test yang ada pada Modul 4 dengan mengisi tanda cek.
2. Isi dengan sebenarnya sesuai keadaan saudara
3. Jika saudara telah memiliki kompetensi seperti yang dinyatakan dalam pre test kerjakan soal-soal Post-Test
4. Jika saudara belum memiliki kompetensi seperti yang dinyatakan dalam pre test pelajari materi pada bagian satu dari Modul ini

B. PETUNJUK POST-TEST

I. UMUM

Dalam tugas ini, pada akhirnya saudara akan memiliki kompetensi terkait dengan :

1. Memahami konstruksi program assembly
2. Memahami proses kerja assembly .
3. Menggunakan berbagai jenis mnemonik
4. Menggunakan assembler directive

II. KHUSUS

1. Kumpulkan dan pelajari contoh-contoh program kemudian mencermati konstruksi program, jenis mnemonik dan assembler directive yang digunakan.

MODUL 4
Menggunakan Bahasa Pemrograman Assembly

BAGIAN 3
PRE-TEST

Subkompetensi	Pernyataan	Saya memiliki kompetensi ini	
		Tidak	Ya
4. Menggunakan Bahasa pemrograman Asembly	4.1. Saya memahami konstruksi program assemly secara baik		
	4.2. Saya memahami Proses kerja asembly secara benar		
	4.3. Saya memahami semus jenis-jenis mnemonik mikrokontroler AT89S51		
	4.4. Saya menguasai assembler directive		

BAGIAN 4
POST-TEST

A. Pilihlah salah satu jawaban yang saudara anggap paling benar

1. Bahasa Assembly adalah
   a. Bahasa high level
   b. Bahasa mesin
   c. Bahasa diantara bahasa mesin dan bahasa high level
   d. Bahasa Low level
2. Untuk mengisikan data 64h ke register A maka perintah yang benar
   a. MOV A, #64h
   b. MOVE A,#64h
   c. MOV A,64h
   d. MOVE A,64h
3. Mnemonik CLR A mewakili perintah
   a. Complement Accumulator
   b. Clear Accumulator
   c. Complement Register Accumulator
   d. Clear Register Accumulator
4. Mnemonik JNC mewakili perintah
   a. Jump if carry not set
   b. Jump if carry set
   c. Jum if non carry
   d. Jump if carry
5. Pernyataan berikut yang benar
   a. Source program dapat dijalankan langsung pada mikrokontroler
   b. Listing program dapat dijalankan pada mikrokontroler
   c. Program objek dapat dijalankan pada mikrokontroler
   d. Assembler dapat dijalankan pada mikrokontroler
6. Konstruksi program assembly berikut yang salah
   a. Satu baris satu perintah dimulai dengan spasi atau tab
   b. Kolom pertama setiap baris merupakan lokasi Label
   c. Operand pada setiap perintah dipisahkan dengan tanda koma
   d. Operand pada setiap perintah diakhiri dengan tanda titik
7. Pada mikrokontroler AT89S51 program awal harus ditulis pada original 0000h karena
   a. Vektor interupsi beralamat 0000h
   b. Vektor reset beralamat 0000h
   c. Vektor control beralamat 0000h
   d. Vektor booting beralamat 0000h

BAGIAN 5
KUNCI JAWABAN

A. Pilihan ganda

1. c
2. a
3. b
4. a
5. c
6. d
7. b