

P E R C O B A A N 2

- TEORI DASAR MIKROKONTROLER MCS-51
- SIMULASI MIKROKONTROLER

Tujuan:

1. Memahami pengertian mikrokontroler
2. Dapat melakukan pemrograman mikrokontroler MCS-51 menggunakan bahasa assembler pada program simulasi MIDE-51.

Teori Dasar:

1. MIKROKONTROLER MCS-51

Pada awal perkembangannya, mikroprosesor dibuat menurut kebutuhan aplikasi yang lebih spesifik, dalam hal ini mikroprosesor dibagi menjadi beberapa jenis, yaitu :

- Mikroprosesor RISC (*Reduced Instruction Set of Computing*) dan CISC (*Complex Instruction Set of Computing*). Jenis ini yang digunakan untuk pengolahan informasi dengan perangkat lunak yang rumit dan digunakan untuk kebanyakan PC (*Personal Computer*) saat ini.
- Pengolah Sinyal Digital, DSP (*Digital Signal Processor*). Memiliki perangkat lunak dan perangkat keras yang ditujukan untuk mempermudah proses pengolahan sinyal-sinyal digital. DSP digunakan pada perangkat audio dan video modern seperti VCD, DVD, *home theatre* dan juga pada kartu-kartu multimedia di komputer.
- Mikrokontroler, adalah mikroprosesor yang dikhususkan untuk instrumentasi dan kendali. Contoh penggunaannya adalah sebagai pengendali motor, berperan seperti PLC (*Programmable Logic Controller*), pengaturan pengapian dan injeksi bahan bakar pada kendaraan bermotor atau alat pengukur otomatis suatu besaran seperti suhu, tekanan, kelembaban dan lain-lain.

Sebuah mikroprosesor yang digabungkan dengan *input-output (I/O)* dan memori (*Random Access Memory/Read Only Memory*) akan membentuk sebuah sistem mikrokomputer. Dari pemikiran CPU yang dapat dikonstruksi dalam sebuah IC tunggal, maka sebuah mikroprosesor, I/O dan memori dapat pula dibangun dalam tingkatan IC. Konstruksi ini menghasilkan Single Chip Microcomputer (SCM). SCM inilah yang disebut sebagai mikrokontroler.

Tahun 1976 Intel meluncurkan mikrokontroler pertama yang disebut seri MCS-48 yang berisi lebih dari 17.000 transistor, hingga saat ini seri ini masih banyak digunakan untuk aplikasi khusus. Seiring perkembangan mikroprosesor, mikrokontroler juga mengalami perkembangan pesat seperti turunan MCS-51, 68HC11, mikrokontroler PIC, Fujitsu dan sebagainya.

Intel 8048 adalah SCM yang pertama, dipasarkan pada tahun 1976, ini merupakan cikal bakal dari mikrokontroler. Keluarga dari 8048 adalah 8021, 8022, 8048, dan 8049 yang hingga saat ini masih digunakan pada alat-alat

kedokteran modern dan digunakan pada keyboard IBM PC untuk *scanning* tombol-tombolnya. Versi 8748 memiliki EPROM 1 kByte untuk menyimpan programnya. Keluarga mikrokontroler pertama ini dikenal dengan nama MCS-48. Generasi kedua mikrokontroler 8 bit adalah keluarga mikrokontroler 8051 di tahun 1980, dengan nama MCS-51 dan diklaim sebagai standar mikrokontroler untuk industri yang menguasai lebih dari 60% pasar mikrokontroler dan menjadi inti bagi terciptanya mikrokontroler produk lainnya. Generasi ketiga adalah mikrokontroler 16 bit, seri MCS-96 yang dapat melakukan operasi 16 bit serta penambahan kemampuan dan kecepatan proses yang ditingkatkan. Kini jutaan chip telah digunakan di seluruh dunia untuk pengendalian proses-proses dan instrumentasi.

Dalam perkembangan yang begitu cepat, batasan-batasan tersebut menjadi kabur, seperti definisi mini, mikro dan mainframe komputer. Beberapa mikrokontroler disebut *embedded processor*, atau *embedded processor* adalah mikrokontroler, artinya prosesor yang diberikan program khusus yang selanjutnya diaplikasikan untuk akuisisi data dan kendali khusus, juga bisa diprogram ulang. Beberapa mikrokontroler modern juga sudah dilengkapi dengan DSP atau terdapat pula mikrokontroler yang tergolong RISC seperti mikrokontroler AVR (*Alf and Vegard's Risc processor*).

Mikrokontroler adalah suatu chip yang dibuat dengan ciri khasnya, umumnya adalah :

- Memiliki memori yang relatif sedikit. Penggunaan mikrokontroler untuk keperluan instrumentasi khusus membuatnya tidak efisien jika menggunakan memori yang besar namun tidak terpakai.
- Memiliki unit I/O langsung. Berbeda dengan mikrokomputer yang unit I/O-nya dapat dikonfigurasi lebih lanjut, mikrokontroler mempunyai unit I/O yang terintegrasi dan berhubungan langsung dengan mikroprosesornya.
- Program atau perangkat lunaknya relatif sederhana. Sesuai fungsi yang dibuat untuk tujuan khusus, mikrokontroler hanya membutuhkan program yang sederhana untuk menjalankan fungsinya.
- Pemroses bit, ketimbang byte. Dengan memori yang sedikit dan implementasi perangkat lunak yang sederhana, mikrokontroler lebih cenderung digunakan untuk memproses bit (*binary digit*) dibandingkan byte (8 bit), untuk kemudian setiap bit disalurkan ke setiap jalur keluaran I/O pada pin-pin yang dimilikinya.
- Beberapa varian memiliki memori yang tidak hilang bila catu padam didalamnya untuk menyimpan program.

Sedangkan dalam hal aplikasi, sistem mikrokontroler memiliki karakteristik sebagai berikut :

- Memiliki program khusus yang disimpan dalam memori untuk aplikasi tertentu, tidak seperti PC yang multifungsi karena mudahnya memasukkan program. Program mikrokontroler relatif lebih kecil daripada program-program pada PC.
- Konsumsi daya kecil.

- Rangkaian sederhana dan kompak.
- Murah, karena komponen yang digunakan sedikit.
- Unit I/O yang sederhana, misalnya keypad, LCD, LED, latch.
- Lebih tahan terhadap kondisi lingkungan ekstrim misalnya temperatur, tekanan, kelembaban dan sebagainya.

Seri MCS-51 sederhana, murah dan mudah didapat di pasaran, cukup untuk aplikasi sederhana bagi pencinta elektronik maupun aplikasi di industri. Chip ini kemudian dikembangkan menjadi beberapa seri dengan berbagai kemampuan seperti pada 8031, 80C31, 8051AH dan 8751.

Beberapa perusahaan produsen semikonduktor membuat *variannya* atas lisensi Intel, yaitu suatu chip yang dapat menjalankan bahasa dan fitur 8051 ditambah dengan kemampuan dan kemudahan khusus. Perusahaan tersebut antara lain AMD, Atmel, Dallas, Matra, OKI, Philips, Siemens, ISS. Produk Philips memberikan tambahan adanya ADC (*Analog to Digital Converter*) dan generator PWM (*Pulse Width Modulation*), sedangkan Dallas mempercepat detak (*clock*) dan siklus mesin, Atmel membuat mikrokontroler yang menggunakan memori *Flash* dan harganya relatif murah.

2. PERINTAH DASAR MCS-51

Perintah dasar yang biasa digunakan pada uController MCS-51 adalah sebagai berikut:

1. clr (clear)

format :

clr a

(mereset atau memberi nilai 00h pada akumulator)

clr rx

(mereset atau memberi nilai 00h pada register x)

Contoh: **clr r0**

clr py

(mereset atau memberi nilai 00h pada port y)

Contoh: **clr p1**

clr <alamat 8 bit>

(mereset atau memberi nilai 00h pada alamat tertentu)

Contoh: **clr 4ah**

2. mov

format :

mov a, px

(menyalin isi data pada port x ke dalam akumulator)

Contoh: **mov a, p3**

mov px, #<nilai 8 bit>

(menyalin suatu nilai 8-bit ke port x)

Contoh: **mov p0, #0feh**

mov px, ry

(menyalin isi data yang nilainya terdapat pada register y ke dalam port x)

Contoh: `mov p3, r5`

3. **setb (set bit)**

format : `setb px.y`

(menset atau memberikan logika 1 pada port x.y)

Contoh: `setb p1.0`

4. **call**

Call terbagi menjadi dua format yaitu *acall (absolute call)* dan *lcall (long call)*, perbedaannya hanya pada kemampuan jauh dekatnya pemanggilan subrutin. Seandainya penggunaan *acall* hanya mampu memanggil sampai alamat 100h maka untuk *lcall* dapat lebih dari itu, namun juga untuk penggunaan *lcall* membutuhkan memori dan siklus mesin yang lebih banyak.

Saat perintah *call* dijalankan, isi register PC (*Program Counter*) akan disimpan ke dalam stack dan digantikan dengan alamat subrutin yang dipanggil. Saat subrutin berakhir dengan ditandai perintah *ret (return)* register PC akan diisi kembali oleh isi dari stack, dan mikrokontroler akan menjalankan perintah di bawah baris perintah *call* tadi.

Format : `acall <label subrutin>`

(perintah untuk memanggil program pada subrutin)

Contoh: `acall cinta`

`lcall <label subrutin>`

(perintah untuk memanggil program pada subrutin)

Contoh: `lcall komputer`

Cat.: Penggunaan subrutin sebaiknya menggunakan kata, untuk kata-katanya sesuka pemrogram boleh menggunakan nama sendiri, nama kota ataupun nama-nama lainnya.

5. **jmp (jump)**

Jmp juga terbagi menjadi dua format yaitu *sjmp (short jump)* dan *ljmp (long jump)*, untuk penggunaannya sama seperti format *call* pada penjelasan di atas, hanya saja *jump* merupakan lompatan sederhana yang tidak dapat mengembalikan nilai register PC seperti perintah *call*.

Format : `sjmp <label subprogram>`

(lompat atau jalankan langsung program yang berada pada label suatu subprogram)

Contoh: `sjmp kamu`

`sjmp <alamat memori>`

(lompat atau jalankan langsung program yang berada pada suatu alamat memori)

Contoh: `ljmp 100h`

6. **djnz (decrement and jump if not zero)**

format : **djnz rx, <label subprogram>**
(kurangi nilai isi data pada register x dan bila nilainya belum mencapai 0 maka akan dilakukan lompatan ke label subprogram)

Contoh: **djnz r7, gaul**
(kurangi nilai isi data pada register R7 dan bila nilainya belum mencapai 0 maka dilakukan lompatan ke subprogram dengan label gaul)

7. jnb (jump if not bit set)

format : **jnb px.y, <label subprogram>**
(lompat ke label subprogram bila nilai port x.y berlogika LOW atau mempunyai nilai 0)

Contoh: **jnb p1.0, go**

Cat.: jnb hanya bisa dijalankan dengan operand yang berkapasitas 1 bit.

8. cjne (compare and jump if not equal)

format : **cjne a, xyz, <label subprogram>**
(bandingkan apakah nilai akumulator sama dengan nilai xyz, bila nilainya tidak sama maka lompat ke label subprogram)

Contoh: **cjne a, #0fh, keren**

cjne rv, xyz, <label subprogram>
(bandingkan apakah nilai register v sama dengan nilai xyz, bila nilainya tidak sama maka lompat ke label subprogram)

Contoh: **cjne r1, #0ach, ganteng**

9. rr (rotate right)

rl (rotate left)

format : **rr a**
(memutar ke kanan 1 bit pada isi akumulator)
 rl a
(memutar ke kiri 1 bit pada isi akumulator)
 rr rx
(memutar ke kanan 1 bit pada isi register x)
 rl rx
(memutar ke kiri 1 bit pada isi register x)

10. inc (increment)

dec (decrement)

format : **inc a**
(menambahkan nilai 1 bit pada akumulator)
 dec a
(mengurangi nilai 1 bit pada akumulator)
 inc rx

(menambahkan nilai 1 bit pada register x)

dec rx

(mengurangkan nilai 1 bit pada register x)

Cat : untuk perintah yang menggunakan decrement, increment, rotate, compare hanya dapat dilakukan oleh akumulator maupun register saja. Bila nilai pada suatu port ingin dilakukan perintah diatas maka port tersebut wajib disalin terlebih dahulu kedalam akumulator atau register dengan menggunakan perintah mov.

- Format penulisan standar bahasa assembly MCS-51 (pada M-IDE51) :

```
$mod51
org 0h
;
~ Main Program ~
;
end
```

Keterangan:

- **\$mod51**
Instruksi ini digunakan agar simulator dapat mengidentifikasi program yang dibuat dalam bahasa assembler.
- **org 0h**
mempunyai fungsi untuk menulis program di alamat 0 hexa pada register mikrokontroler
- **Main Program**
Berisi program utama
- **End**
Mengakhiri baris program

[illegible]

Alat-alat :

- 1 Set IBM PC dengan Sistem Operasi Windows dan Software M-IDE51, dan ISP Flash Programmer, dan TOP View Simulator
- 1 Set Modul Basic Training Kit lengkap dengan sistem interkoneksinya.


Prosedur percobaan :

Bagian 1

Modul 2.1 Clear, Mov, dan SetB

1. Bukalah **M-IDE 51** yang ada di desktop kemudian klik New untuk membuat lembaran kerja baru.
2. Ketik Program dan setelah program, Save as program tersebut dalam bentuk **.asm**

```
$mod51
; inisialisasi program
org 0h
; org (origin) awal penulisan program pada alamat 0 hex
mulai : clr a
        mov p2,a
        sjmp mulai
end
```

3. klik **F9** atau klik symbol seperti ini  (Running Program) untuk mengetahui program tersebut terjadi error atau no error.
4. Setelah program berhasil, kemudian bukalah **TOP View Simulator** untuk melihat simulasi program yang telah kita buat.
5. Klik **File**, kemudian pilih menu **EXTERNAL MODULE SETTING > LED** untuk menentukan bagian apa yang digunakan. Setelah itu, buka **View** klik **External Module > LED** sebelumnya kita harus mengetahui dulu program apa yang sedang kita gunakan. Setelah itu Klik **File** Kemudian **Load Program**, Kemudian Klik **GO** untuk menjalankan Simulasi Tersebut dan Klik **Stop** untuk menghentikan simulasinya. **Tanyakan Kepada Asisten Jika Kurang Jelas !!**
6. Catat hasil keluaran pada rangkaian led pada Simulator



7. Ubah program sebelumnya menjadi seperti di bawah ini dan simulasikan.
Catat hasil keluaran setiap port.

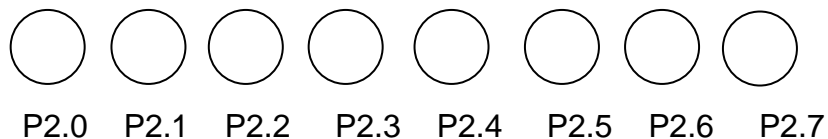
```
$mod51
org 0h
mulai : mov p2,#00h
        jnb p3.0, mati
        sjmp mulai
mati   : setb p2.1
        ;setb digunakan untuk penambahan 1 bit
        sjmp mati

end
```

8. Ulangi Langkah Untuk Melihat Simulasi program pada TOP View Simulator

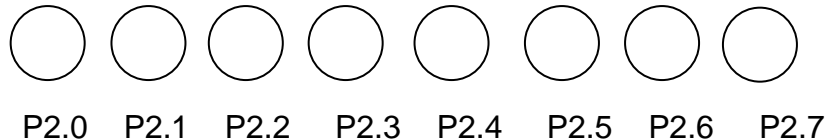
Gambarkan keluaran dari program diatas:

- a. Sebelum P3.0 ditekan :



● Low = 0
○ High=1

- b. Setelah P3.0 ditekan :



Kesimpulan :

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....


.....

.....

.....

.....

Modul2.2 Teknik Delay, Conditional Jump, Rotate Right dan Rotate Left

1. Bukalah **M-IDE 51** yang ada di desktop kemudian klik New untuk membuat lembaran kerja baru.
2. Ketik Program dan setelah program selesai diketik, Save as program tersebut dalam bentuk **.asm**
3. klik **F9** atau klik symbol seperti ini  (Running Program) untuk mengetahui program tersebut terjadi error atau no error.
4. Setelah program berhasil, kemudian bukalah **TOP View Simulator** untuk melihat simulasi program yang telah kita buat.
5. Klik File, kemudian arahkan kursor kearah menu **EXTERNAL MODULE SETTING** untuk menentukan bagian apa yang digunakan. Setelah itu, buka **View** klik **External Module** sebelumnya kita harus mengetahui dulu program apa yang sedang kita gunakan. Setelah itu Klik **File** Kemudian **Load Program**,.Kemudian Klik **GO** untuk menjalankan Simulasi Tersebut dan Klik **Stop** untuk menghentikan simulasinya. *Tanyakan Kepada Asisten Jika Kurang Jelas!!*

Program 2.2.A

```
$mod51
flip equ p0      ;menyatakan "flip" sama dengan p0
org 0h
balik:

    mov flip,#00001111b ; nibble atau 4-bit atas yang nyala
    call delay          ;delay:waktu tunda
    call delay

    mov flip,#11110000b ; nibble atau 4-bit bawah yang nyala
    call delay
    call delay

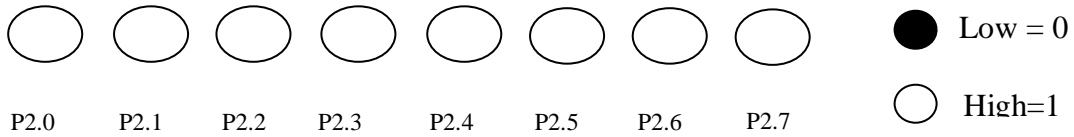
    sjmp balik

delay:mov r0,#07h

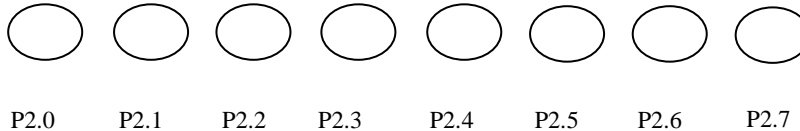
lagi:  djnz r2,lagi
       djnz r1,lagi
       djnz r0,lagi
       ret
       end
```

6. Catat hasil keluaran pada Simulator

a. Keluaran Kondisi 1 :



b. Keluaran Kondisi 2 :



Kesimpulan:

.....

.....

.....

.....

Ketik Program di bawah ini. Simpan, kemudian simulasikan.

Program 2.2.B

PROGRAM	KETERANGAN
<pre> \$mod51 org 0h ; mov a,#0feh ; kiri: mov p2,a acall delay RL A cjne a,#7fh,kiri kanan: mov p2,a acall delay RR A cjne a,#0feh,kanan sjmp kiri delay: mov r0,#07h lagi: djnz r2,lagi djnz r1,lagi djnz r0,lagi ret end </pre>	

Modul 2.3 Register

1. Bukalah **M-IDE 51** yang ada di desktop kemudian klik New untuk membuat lembaran kerja baru.
2. Ketik Program dan setelah program, Save as program tersebut dalam bentuk **.asm**

```
$mod51
org 100h
mov r3, #0fh
mov p2, r3
pil:    jnb p3.0, krng
        jnb p3.7, tmbh
        sjmp pil

krng:   dec r3
        mov p2, r3
        acall delay
        sjmp pil

tmbh:   inc r3
        mov p2, r3
        acall delay
        sjmp pil

delay:  mov r0, #0fh
ulang:  djnz r2, ulang
        djnz r1, ulang
        djnz r0, ulang
        ret
end
```

- a. Bagaimana nilai heksadesimal pada port 2 bila P3.0 mendapat logika LOW?
Mengapa terjadi demikian?

.....
.....
.....
.....

- b. Bagaimana nilai heksadesimal pada port 2 bila P3.7 mendapat logika LOW?
Mengapa terjadi demikian?

.....
.....
.....
.....

1. Buatlah program dengan menggunakan bahasa assembly dengan ketentuan sebagai berikut.
 - a. Jika ditekan **P0.1** maka led akan menyala dari P2.0 berjalan menuju ke P2.7, Kemudian Jika **P0.1** ditekan maka led akan menyala kembali dari P2.7 berjalan menuju ke P2.0 (***dalam penekanan yang sama menghasilkan dua kondisi yang berbeda!!!***)

Program Anda:

This image shows a blank sheet of handwriting practice paper. It features two vertical columns, each containing ten sets of horizontal dashed lines. These lines are designed to help children learn letter formation and alignment. The entire page is enclosed in a simple black border.

Nama Assisten :	
BAGIAN 1	MANDIRI
Tanggal Periksa :	

This image shows a full page of white paper with horizontal dotted lines. The lines are evenly spaced and run across the width of the page, providing a guide for handwriting practice. There are no margins, text, or other markings on the page.

