Sistem Mikrokontroler

Oleh : Hendawan Soebhakti, ST

Program Studi Teknik Elektro
Politeknik Batam
Juni 2007

Pokok Bahasan

Pendahuluan
Arsitektur MCS-51
Operasi Pemindahan Data
Operasi Logika
Operasi Aritmatika
Operasi Percabangan
Operasi Timer/Counter
Aplikasi Seven Segmen
Aplikasi Motor Stepper
Komunikasi Data Serial
Operasi Interupsi

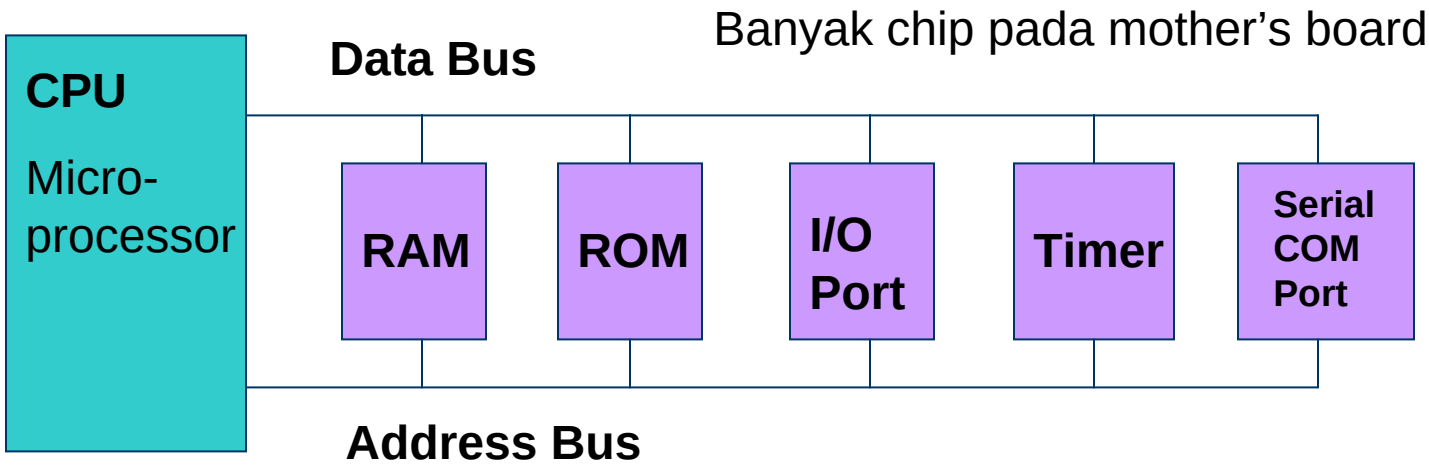
Pendahuluan

Mengapa perlu mikroprosesor/mikrokontroler?

- Mikroprosesor adalah inti dari sistem komputer
- Pada saat ini banyak perangkat elektronika yang dikendalikan oleh mikroprosesor/mikrokontroler
- Dengan mikroprosesor/mikrokontroler penggunaan komponen dapat dikurangi sehingga biaya produksi dapat dikurangi.

Mikroprosesor :

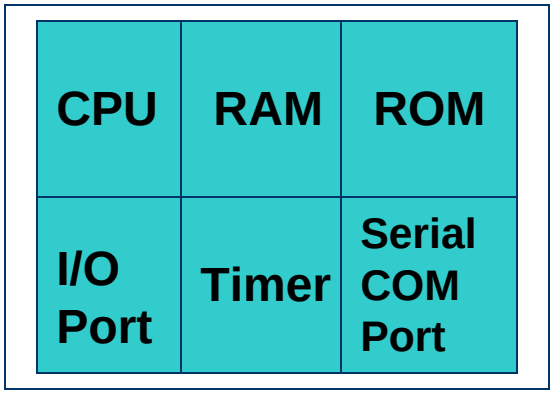
- CPU *(Central Processing Unit)*
- Tidak terdapat RAM, ROM, I/O pada chip CPU
- Contoh : Intel's x86, Motorola's 680x0



Microprocessor System

Mikrokontroler :

- Mini komputer
- CPU, RAM, ROM dan Port I/O dalam satu chip
- Contoh : Motorola’s 6811, Intel’s 8051, Zilog’s Z8 and PIC 16X



A single chip

Microcontroller

Mikroprosesor vs Mikrokontroler

Mikroposesor

- CPU berdiri sendiri, RAM, ROM, I/O dan timer terpisah dalam chip yang berbeda.
- Desainer dapat memilih besarnya kapasitas RAM, ROM dan jumlah port I/O.

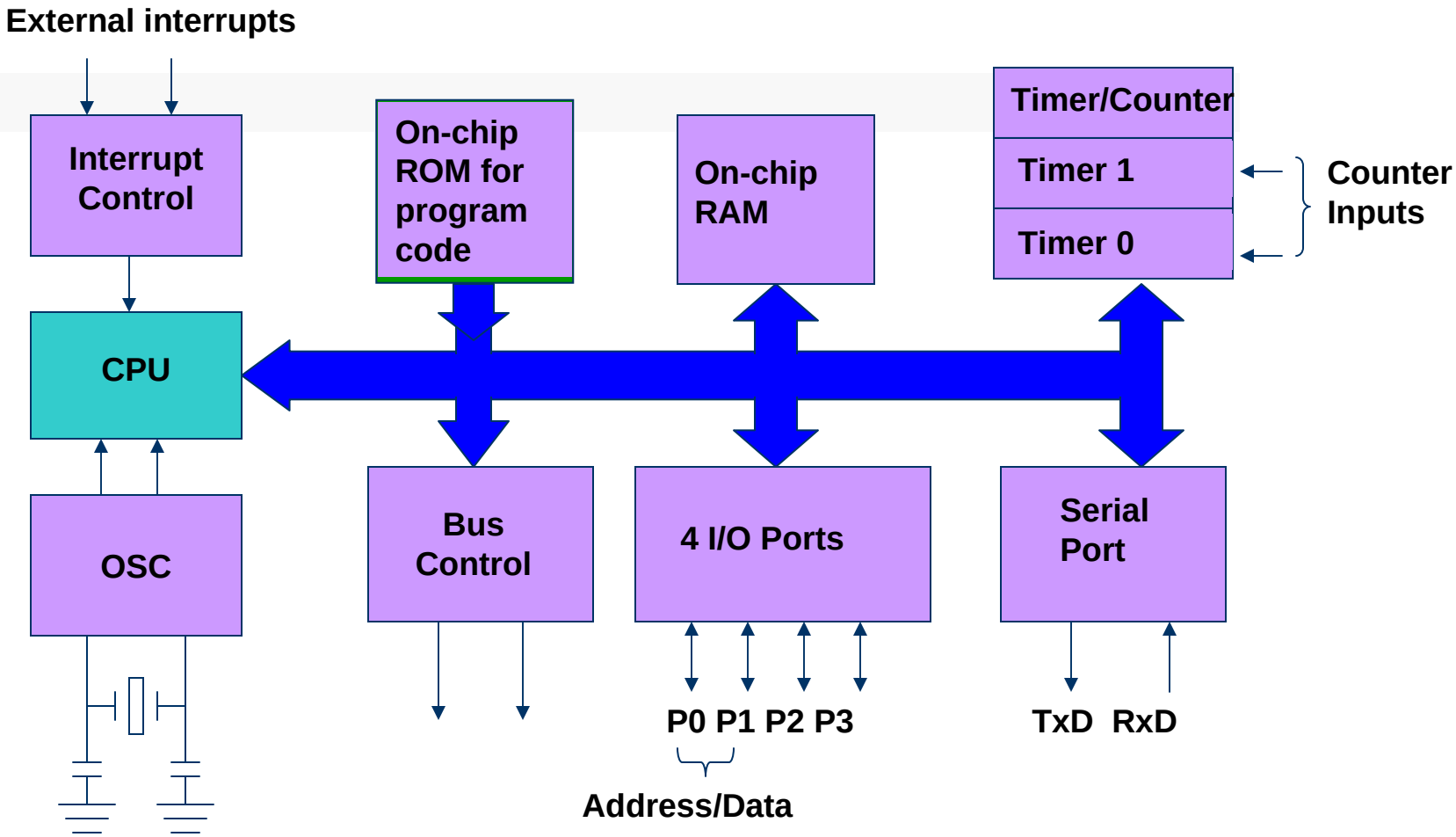
Mikrokontroler

- CPU, RAM, ROM, I/O dan timer menjadi satu pada sebuah chip.
- Besarnya kapasitas ROM, RAM dan Port I/O sudah ditentukan sesuai tipe mikrokontroler.

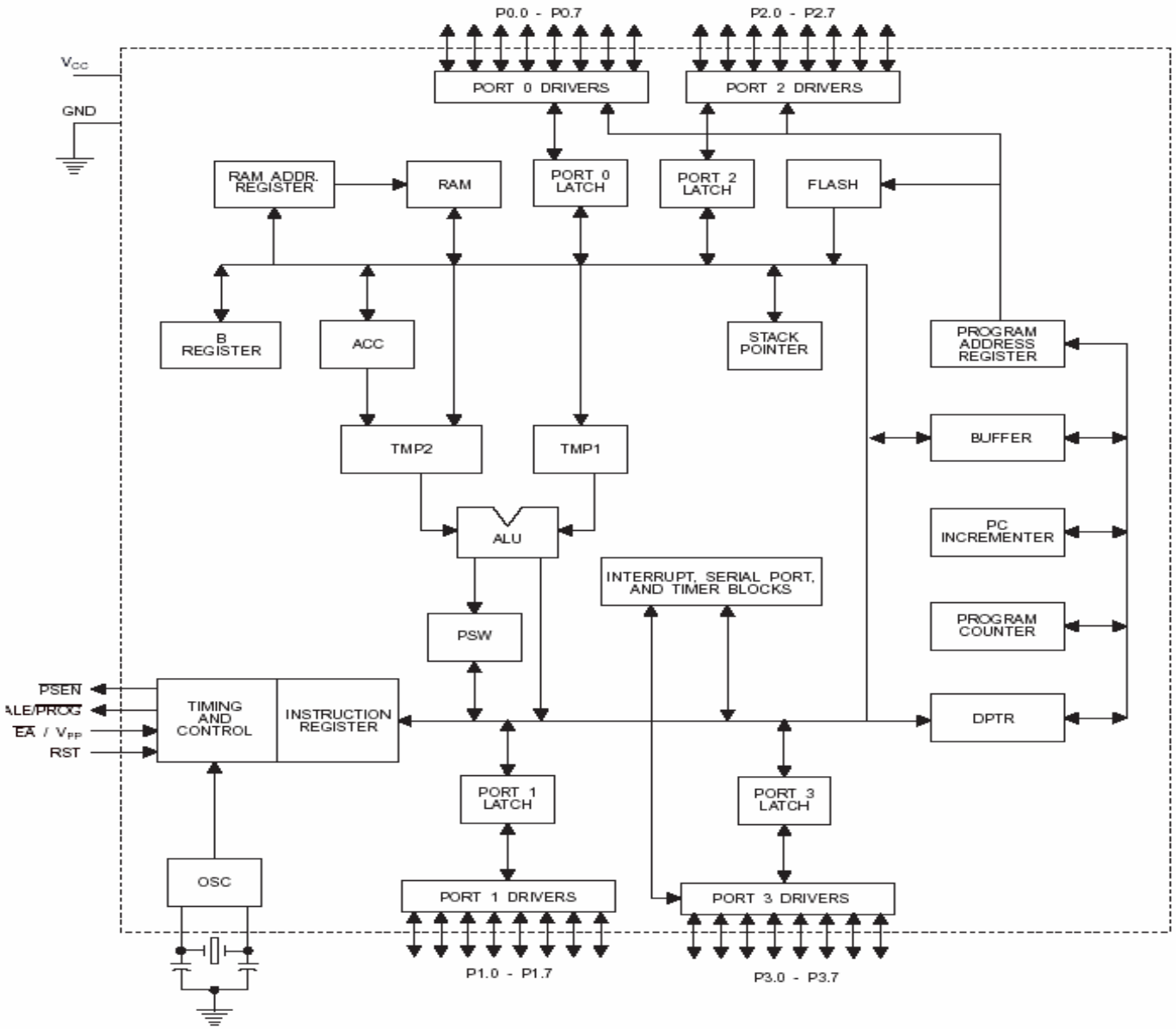
Arsitektur MCS-51

Diagram Blok 89S51
Peta Memori Internal 89S51
Konfigurasi pin AT89S51
Osilator dan Clock
Organisasi RAM Internal
Special Function Register

Diagram Blok 89S51



Block Diagram



Memori Internal 89S51

Memori internal 89S51 terdiri dari 3 bagian yaitu ROM, RAM dan SFR

ROM / *Read Only Memory* adalah memori tempat menyimpan program / *source code*.
Sifat ROM adalah *non-volatile*, data / program tidak akan hilang walaupun tegangan supply tidak ada.
Kapasitas ROM tergantung dari tipe mikrokontroler. Untuk AT89S51 kapasitas ROM adalah 4 KByte.
ROM pada AT89S51 menempati address 0000 s/d 0FFF.

RAM / *Random Access Memory* adalah memori tempat menyimpan data sementara.
Sifat RAM adalah *volatile*, data akan hilang jika tegangan supply tidak ada.
Kapasitas RAM tergantung pada tipe mikrokontroler. Pada AT89S51 RAM dibagi menjadi 2 yaitu :

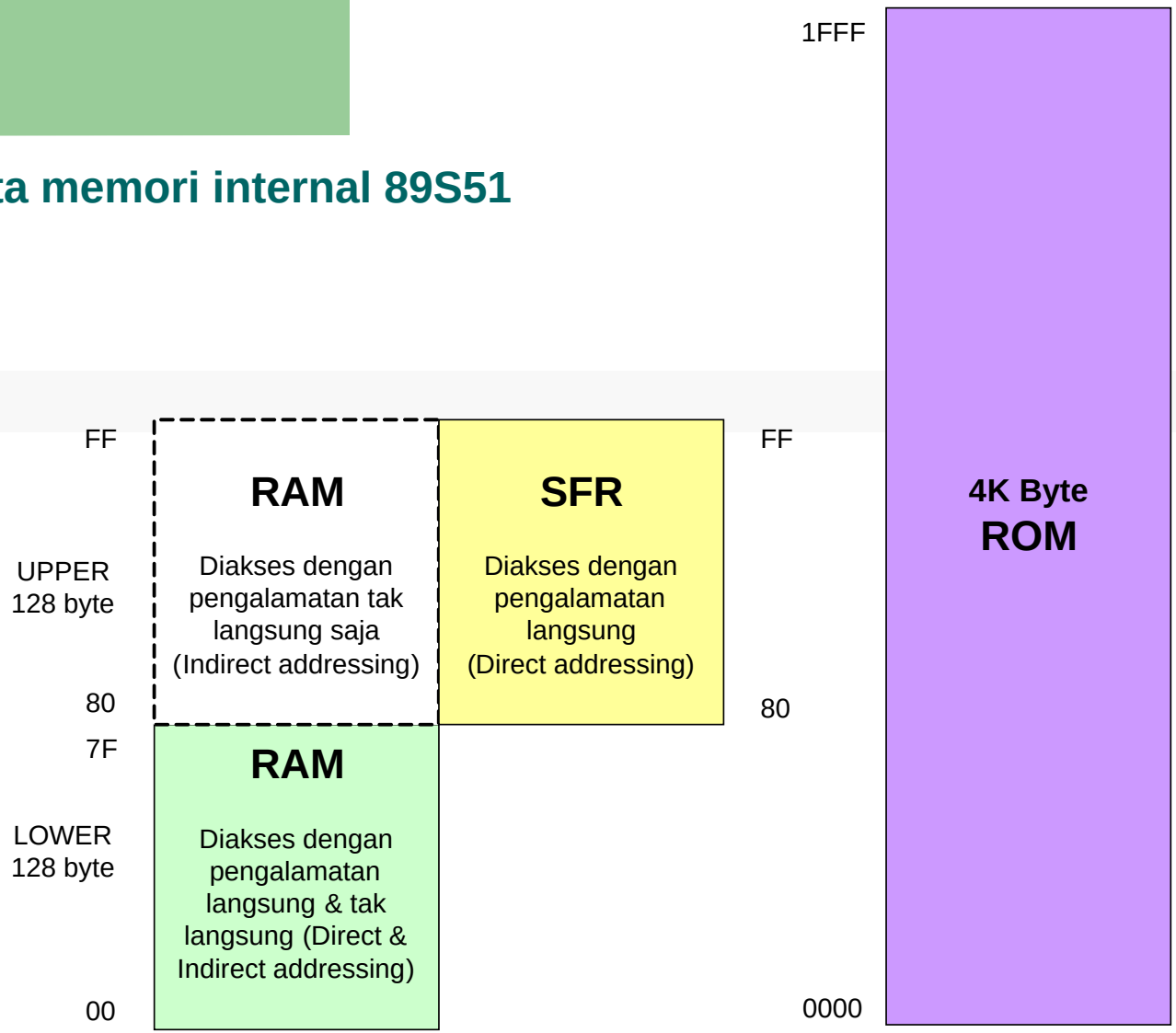
- LOWER 128 byte yang menempati address 00 s/d 7F.
  RAM ini dapat diakses dengan pengalamatan langsung (direct) maupun tak langsung (indirect)
  Contoh :

```
Direct   ->  mov   30h,#120   ; Pindahkan data 120 ke RAM pada address 30h
Indirect ->  mov   R0,#30h    ; Isi Register 0 dengan 30h
             mov   @R0,#120   ; Pindahkan data 120 ke RAM pada address sesuai isi R0
```

- UPPER 128 byte yang menempati address 80 s/d FF. Address ini sama dengan address SFR meski secara fisik benar-benar berbeda.
  RAM ini hanya dapat diakses dengan pengalamatan tak langsung saja.

SFR / *Special Function Register* adalah register dengan fungsi tertentu. Misalnya, register TMOD dan TCON adalah *timer control register* yang berfungsi mengatur fasilitas timer mikrokontroler.
SFR pada AT89S51 menempati address 80 s/d FF.

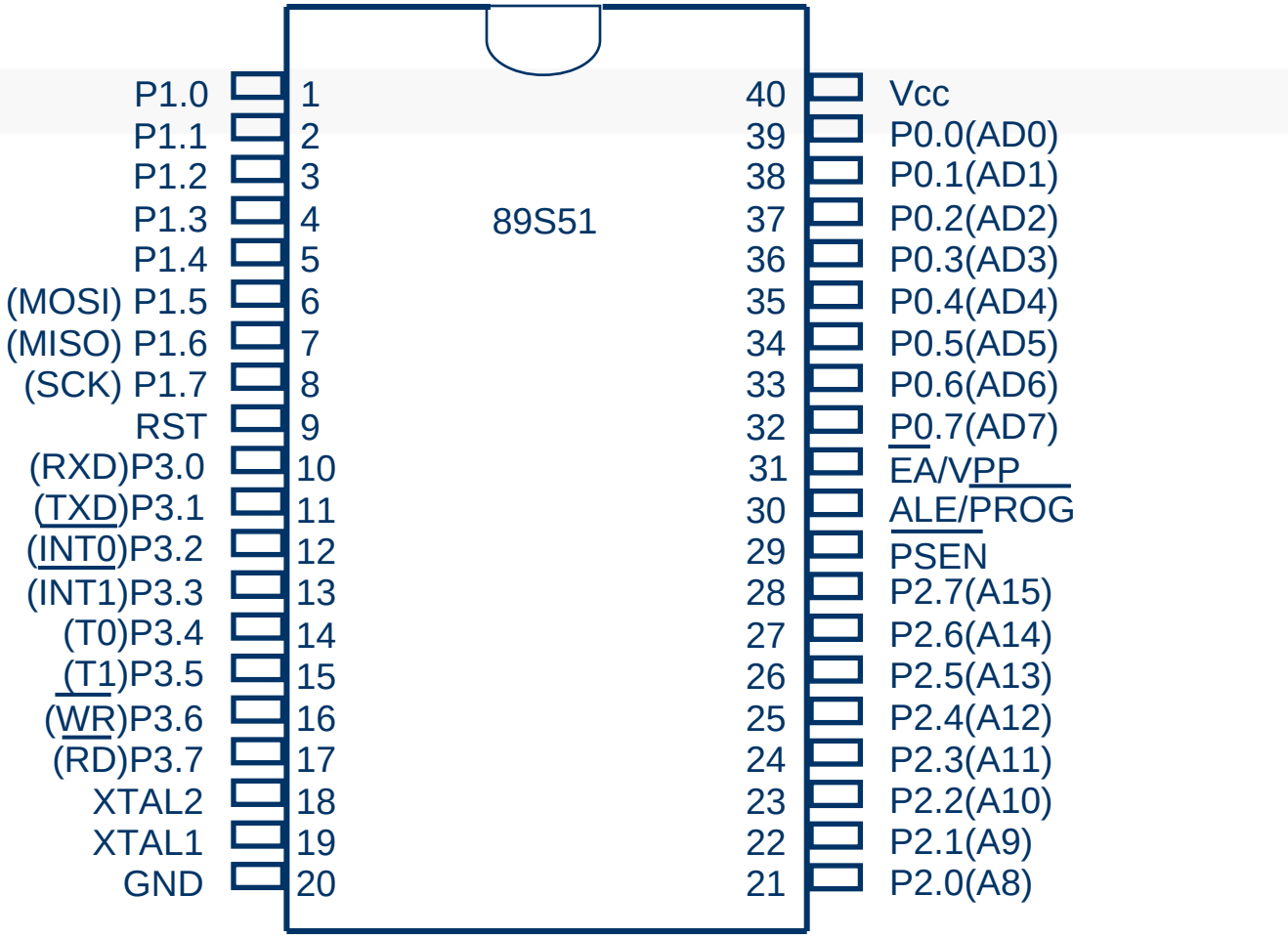
➢Peta memori internal 89S51



Catatan :

Gambar diatas adalah peta memori internal 89S51 yang terdiri dari RAM, SFR dan ROM.
Tampak bahwa ada kesamaan address antara RAM, SFR dan ROM yaitu pada address 00 s/d FF.
Atas pertimbangan inilah maka biasanya source code ditulis setelah address 00FF yaitu 0100 pada ROM
Hal ini dimaksudkan agar data RAM dan SFR tidak terisi oleh byte source code.

Konfigurasi pin AT89S51



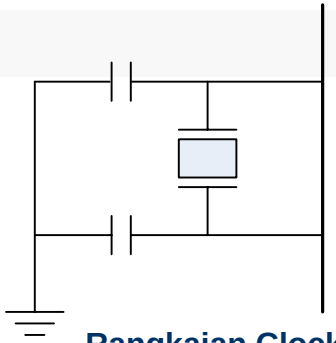
VCC	Tegangan supply +5V
GND	Ground
Port 0	Port 0 merupakan port paralel 8 bit dua arah (bi-directional) yang dapat digunakan untuk berbagai keperluan. Port 0 juga memultipleks alamat dan data jika digunakan untuk mengakses memori eksternal
Port 1	Port 1 merupakan port paralel 8 bit bi-directional dengan internal pull-up. Port 1 juga digunakan dalam proses pemrograman (In System Programming) P1.5 MOSI P1.6 MISO P1.7 SCK
Port 2	Port 2 merupakan port paralel 8 bit bi-directional dengan internal pull-up. Port 2 akan mengirim byte alamat jika digunakan untuk mengakses memori eksternal
Port 3	Port 3 merupakan port paralel 8 bit bi-directional dengan internal pull-up. Port 3 juga bisa difungsikan untuk keperluan khusus yaitu : P3.0 RXD(Receive Data) P3.1 TXD(Transmit Data) P3.2 INT0(Interrupt 0) P3.3 INT1(Interrupt 1) P3.4 T0(Timer 0) P3.5 T1(Timer 1) P3.6 WR(Write Strobe) P3.7 RD(Read Strobe)

RST	Pulsa dari low ke high akan mereset mikrokontroler
ALE/PROG	Address Latch Enable, digunakan untuk menahan alamat memori eksternal selama pelaksanaan instruksi
PSEN	Program Store Enable, merupakan sinyal kendali yang memperbolehkan program memori eksternal masuk ke dalam bus selama proses pengambilan instruksi
EA/VPP	Jika EA=1 maka mikrokontroler akan melaksanakan instruksi dari ROM internal Jika EA=0 maka mikrokontroler akan melaksanakan instruksi dari ROM eksternal
XTAL1	Input ke rangkaian osilator internal
XTAL2	Output dari rangkaian osilator internal

Osilator dan Clock

Agar dapat mengeksekusi program, mikrokontroler membutuhkan pulsa clock. Pulsa ini dapat dihasilkan dengan memasang rangkaian resonator pada pin XTAL1 dan XTAL2. Frekuensi kerja maksimum 89S51 adalah 33 MHz.

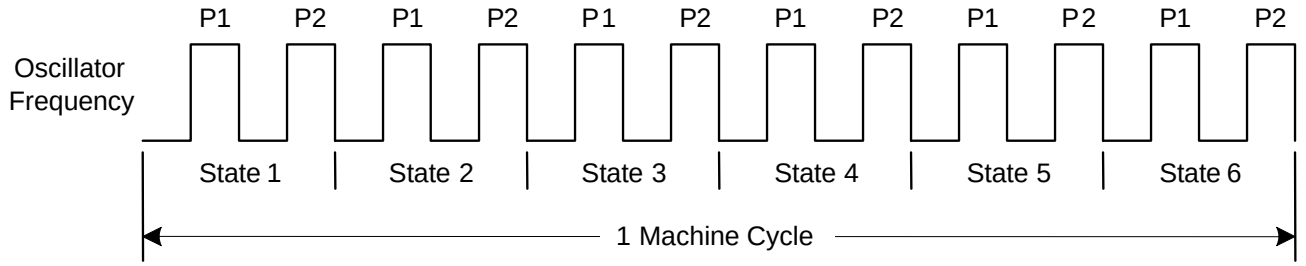
Gambar disamping adalah contoh rangkaian osilator yang bisa digunakan pada mikrokontroler. Komponen utamanya adalah quartz crystal yang dihubungkan dengan kapasitor. Nilai kapasitornya biasanya 33pF.



Rangkaian Clock

Dalam mikrokontroler dikenal istilah *Machine Cycle* (MC) / Siklus Mesin, dimana :

1 MC = 6 state = 12 periode clock
Jika frekuensi crystal yang digunakan adalah 12 MHz maka 1 MC = 12/frekuensi crystal = 12/12 MHz =1uS



➢Waktu Eksekusi Instruksi

Waktu eksekusi sebuah instruksi oleh mikrokontroler tergantung dari jenis instruksi dan frekuensi clock yang digunakan. Setiap instruksi memiliki panjang byte dan jumlah siklus yang berbeda.
Byte instruksi (***Byte***) menandakan jumlah lokasi memori yang dipakai
Siklus instruksi (***Cycle***) menandakan jumlah machine cycle yang dibutuhkan.

Waktu eksekusi dapat dihitung dengan rumus :

$$Tinst = \frac{C \times 12}{frekuensi\ crystal}$$

Dimana :
Tinst : Waktu yang dibutuhkan untuk mengeksekusi 1 instruksi (Secon)
C : Jumlah machine cycle

Contoh :
Diketahui sebuah mikrokontroler dengan frekuensi crystal 12 MHz.
Berapakah waktu yang diperlukan untuk mengeksekusi perintah berikut ini?

```
Mov     A,#30h
```

Jawab :
Dari lembaran data 8051 Operational Code Mnemonics diketahui bahwa instruksi dengan format
Mov A,#n adalah instruksi dengan Byte = 1 dan Cycle = 1

Maka : Tinst = (1x12)/12MHz=1uS

Beberapa contoh Operational Code Mnemonics (Opcode)

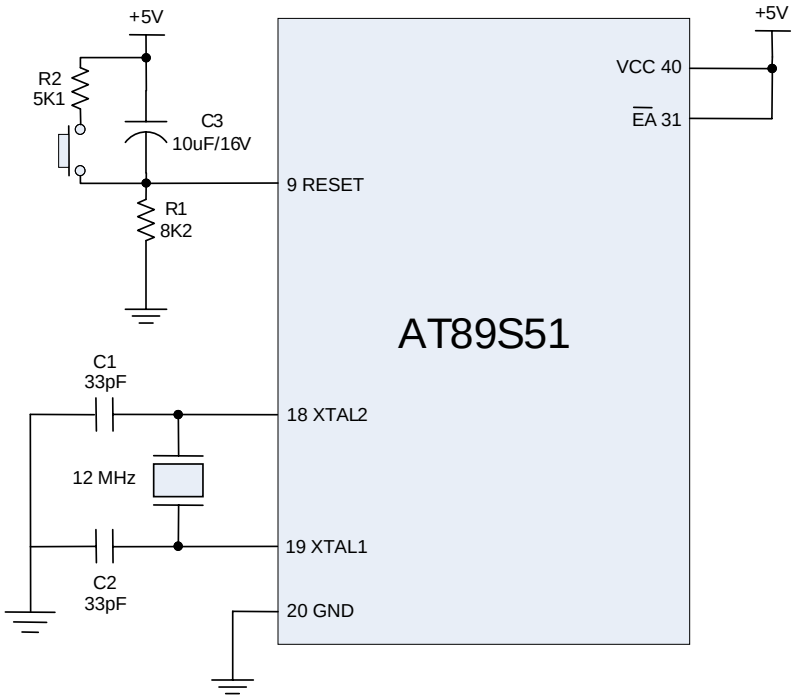
MNEMONICS		BYTE	CYCLE	MNEMONICS		BYTE	CYCLE
ADD	A,Rr	1	1	MOV	add,#n	3	2
ADD	A,add	2	1	MOV	add,@Rp	2	2
ADD	A,@Rp	1	1	MOV	@Rp,A	1	1
ADD	A,#n	2	1	MOV	@Rp,add	2	2
DEC	A	1	1	MOV	@Rp,#n	2	1
DEC	Rr	1	1	MOV	DPTR,#nn	3	2
DEC	add	2	1	CJNE	A,#n,radd	3	2
DEC	@Rp	1	1	DJNZ	Rr.radd	2	2
DIV	AB	1	4	DJNZ	add,radd	3	2
INC	A	1	1	ACALL	sadd	2	2
INC	Rr	1	1	LCALL	ladd	3	2
INC	add	2	1	SJMP	radd	2	2
INC	DPTR	1	2	AJMP	sadd	2	2
MUL	AB	1	4	LJMP	ladd	3	2
SUBB	A,Rr	1	1	JB	b,radd	3	2
SUBB	A,add	2	1	JZ	radd	2	2
SUBB	A,@Rp	1	1	JNZ	radd	2	2
SUBB	A,#n	2	1	RET		1	2
CLR	A	1	1	RETI		1	2
NOP		1	1	SETB	b	2	1
RL	A	1	1	ANL	A,Rr	1	1
RR	A	1	1	ANL	A,add	2	1
SWAP	A	1	1	ANL	A,@Rp	1	1
MOV	A,#n	2	1	ORL	A,Rr	1	1
MOV	Rr,A	1	1	ORL	A,add	2	1
MOV	add,Rr	2	2	ORL	A,@Rp	1	1

Rangkaian Minimum Sistem

Agar mikrokontroler dapat bekerja maka diperlukan komponen tambahan berupa rangkaian reset dan clock.

Mengapa perlu reset ?

Saat power dinyalakan, instruksi yang pertamakali dieksekusi oleh mikrokontroler adalah instruksi yang tersimpan pada address 0000h. Agar Program Counter (PC) dapat menunjuk address 0000h pada saat awal maka mikrokontroler perlu di-reset. Caranya adalah dengan memberikan pulsa high pada pin Reset selama minimal 2 machine cycle (jika f crystal = 12 MHz maka 2MC = 2uS). Setelah itu baru diberikan pulsa low. Kondisi ini dapat dipenuhi dengan memasang rangkaian RC yang akan mensuplai tegangan Vcc ke pin 9 selama kapasitor mengisi muatan / *charging.*
Konstanta waktu pengisian dapat dihitung dengan mengalikan nilai R dan C.
Pada rangkaian dibawah adalah : T=R.C = (8K2).(10uF) = 82mS. Setelah kapasitor terisi, maka pin 9 akan low.



Tombol push button dipasang agar pada saat *running* Mikrokontroler dapat juga di-reset.

Pin EA / External Access harus dihubungkan ke +5V agar mikrokontroler dapat mengambil byte instruksi dari ROM internal mikrokontroler.

Organisasi RAM Internal

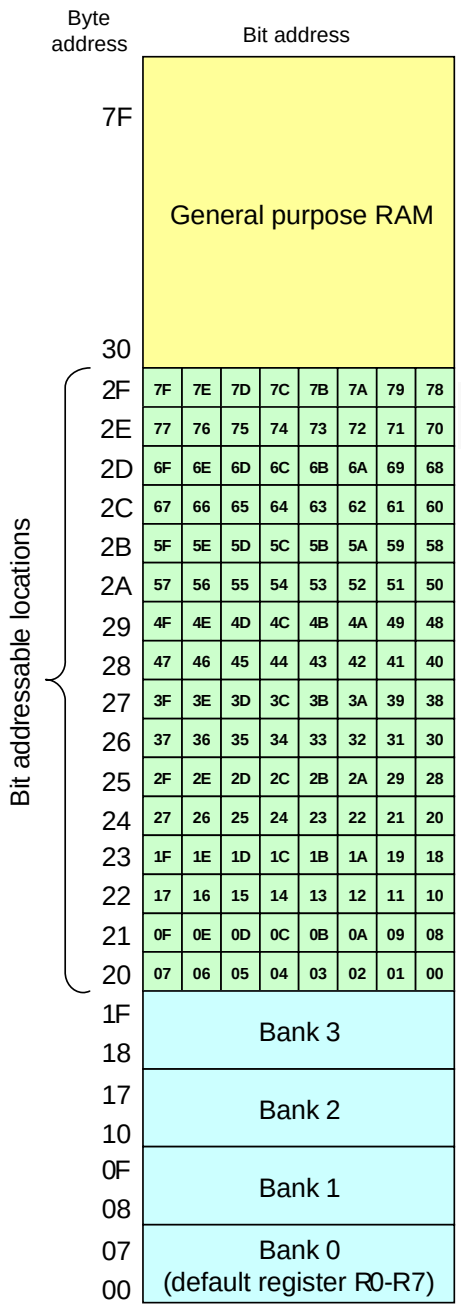
RAM (Random Access Memory) internal AT89S51 berfungsi untuk menyimpan data sementara. Data akan tetap disimpan selama ada supply tegangan ke mikrokontroler.

Pada AT89S51, RAM dibagi menjadi 3 bagian yaitu :

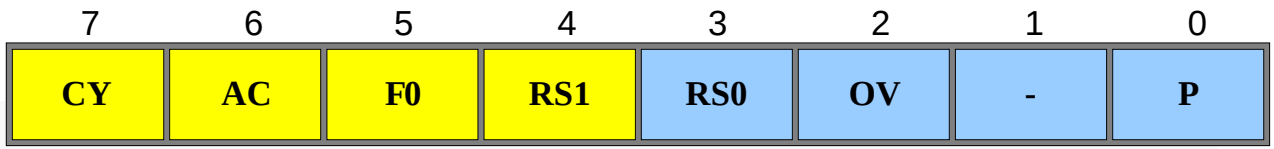
- **Register serba guna**
  Terdiri dari Bank 0, Bank 1, Bank 2, Bank 3.
  Tiap bank register terdiri dari 8 register 8 bit yaitu R0, R1,.. ,R7
  Pemilihan bank register ditentukan pada register PSW
  Rentang address : 00 s/d 1F

- **Bit addressable RAM**
  Adalah RAM yang dapat diakses per bit.
  Ini diperlukan pada saat kita ingin menyimpan data yang panjangnya hanya 1 bit. Setiap bit pada lokasi RAM ini memilil address sendiri-sendiri seperti terlihat pada gambar.
  Rentang address : 20 s/d 2F

- **General purpose RAM**
  Adalah RAM yang dapat diakses per byte.
  Ini diperlukan pada saat kita ingin menyimpan data yang panjangnya 8 bit.
  Rentang address : 30 s/d 7F



Register PSW



PSW / Program Status Word Special Function Register

Bit	Symbol	Fuction
7	CY	Carry flag. Digunakan dalam instruksi aritmatika, JUMP, ROTATE dan BOOLEAN
6	AC	Auxilliary carry flag Digunakan untuk aritmatika BCD
5	F0	User flag 0.
4	RS1	Register bank select bit1.
3	RS0	Register bank select bit0. RS1 RS0 0 0 Select register bank0 0 1 Select register bank1 1 0 Select register bank2 1 1 Select register bank3
2	OV	Over flow flag Digunakan dalam instruksi aritmatika
1	-	Tidak digunakan
0	P	Parity flag. Menunjukkan parity register A1 = Odd parity

Bit addressable as PSW0 to PSW.7

Special Function Register

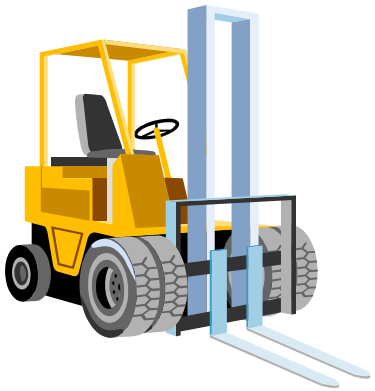
0F8H								0FFH
0F0H	B 00000000							0F7H
0E8H								0EFH
0E0H	ACC 00000000							0E7H
0D8H								0DFH
0D0H	PSW 00000000							0D7H
0C8H								0CFH
0C0H								0C7H
0B8H	IP XX000000							0BFH
0B0H	P3 11111111							0B7H
0A8H	IE 0X000000							0AFH
0A0H	P2 11111111		AUXR1 XXXXXXX0				WDTRST XXXXXXXX	0A7H
98H	SCON 00000000	SBUF XXXXXXXX						9FH
90H	P1 11111111							97H
88H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000	AUXR XXX00XX0	8FH
80H	P0 11111111	SP 00000111	DP0L 00000000	DP0H 00000000	DP1L 00000000	DP1H 00000000		PCON 0XXX0000 87H

Simbol	Nama	Alamat
ACC	Akumulator	E0H
B	B register	F0H
PSW	Program Status Word	D0H
SP	Stack Pointer	81H
DPTR0	Data Pointer 0 16 bit DP0L Byte rendah DP0H Byte tinggi	 82H 83H
DPTR1	Data Pointer 0 16 bit DP1L Byte rendah DP1H Byte tinggi	 84H 85H
P0	Port 0	80H
P1	Port 1	90H
P2	Port 2	A0H
P3	Port 3	B0H
IP	Interrupt Priority Control	B8H
IE	Interrupt Enable Control	A8H
TMOD	Timer/Counter Mode Control	89H
TCON	Timer/Counter Control	88H

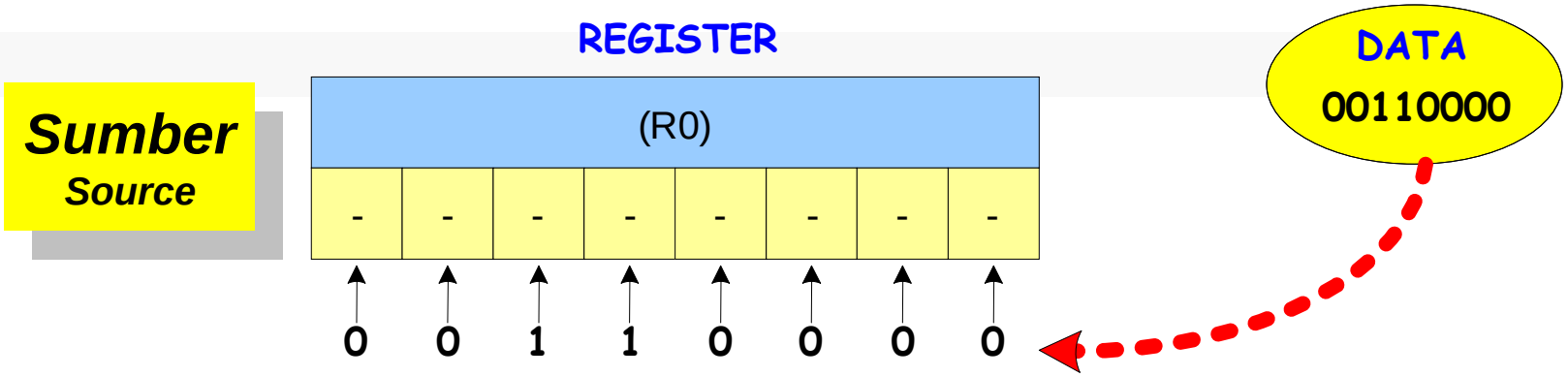
Simbol	Nama	Alamat
TH0	Timer/Counter 0 High Byte	8CH
TL0	Timer/Counter 0 Low Byte	8AH
TH1	Timer/Counter 1 High Byte	8DH
TL1	Timer/Counter 1 Low Byte	8BH
SCON	Serial Control	98H
SBUF	Serial Data Buffer	99H
PCON	Power Control	87H
WDTRST	Watchdog Timer Reset	A6H
AUXR	Auxiliary Register	8EH

Operasi Pemindahan Data

Immediate Addressing Mode
Register Addressing mode
Direct Addressing Mode
Indirect Addressing Mode



Immediate Addressing Mode



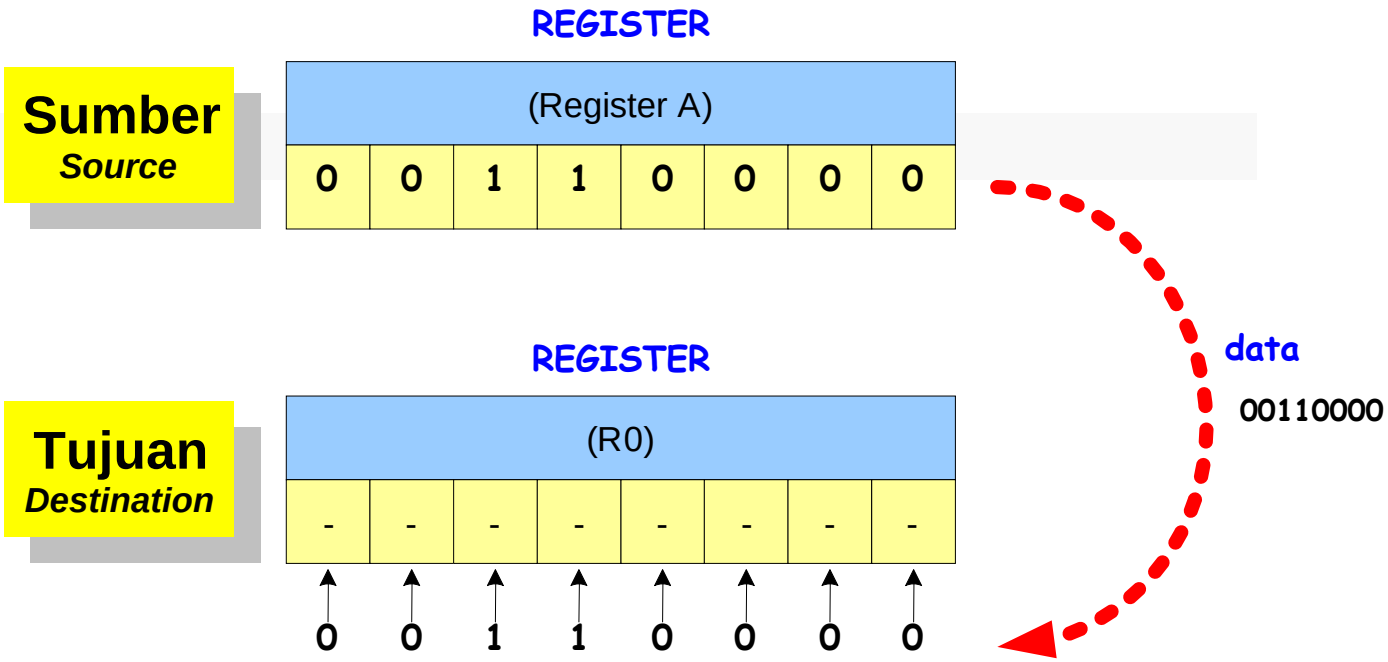
Immediate Addressing Mode

Immediate Addressing : Data langsung dipindahkan ke register
Contoh :

```
MOV     A,#30h          ; Copy the immediate data 30h to A register
MOV     R0,#00110000B   ; Copy the immediate data 30h to R0 register
MOV     DPTR,#48        ; Copy the immediate data 30h to DPTR register
```

Catatan :
30hexa = **00110000b**iner = **48**desimal

Register Addressing mode



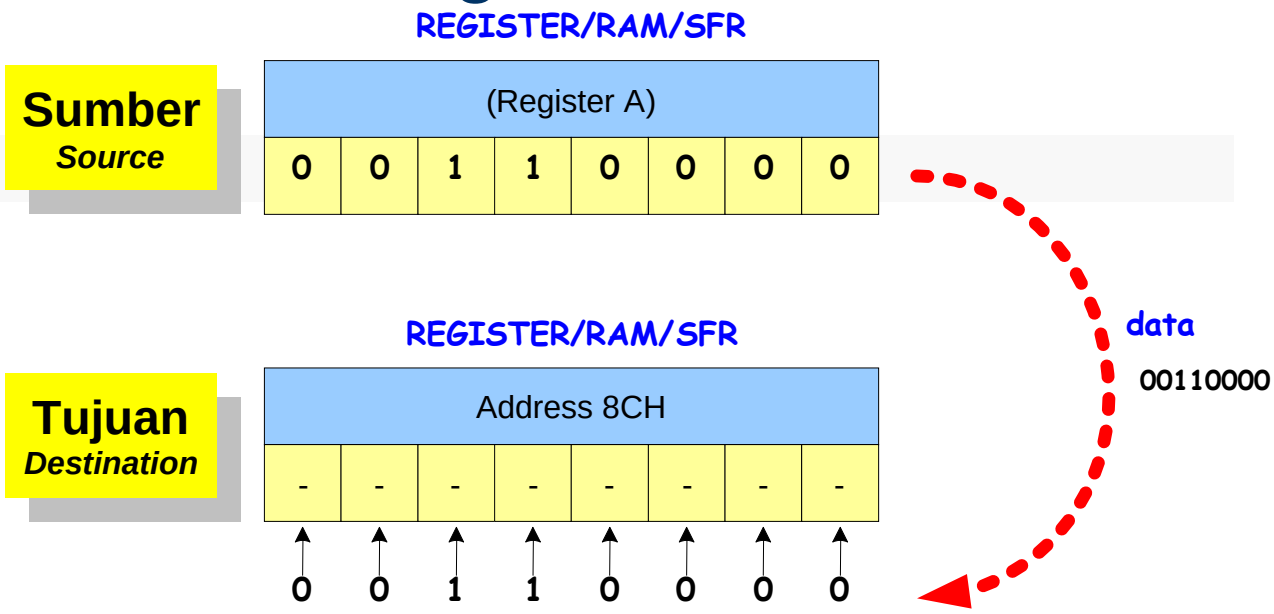
Register Addressing Mode

Register Addressing : Data dari register sumber dipindahkan ke register tujuan

Contoh :

```
MOV     R0,A        ; Copy data from A register to R0 register
MOV     A,R7        ; Copy data from R7register to A register
```

Direct Addressing Mode



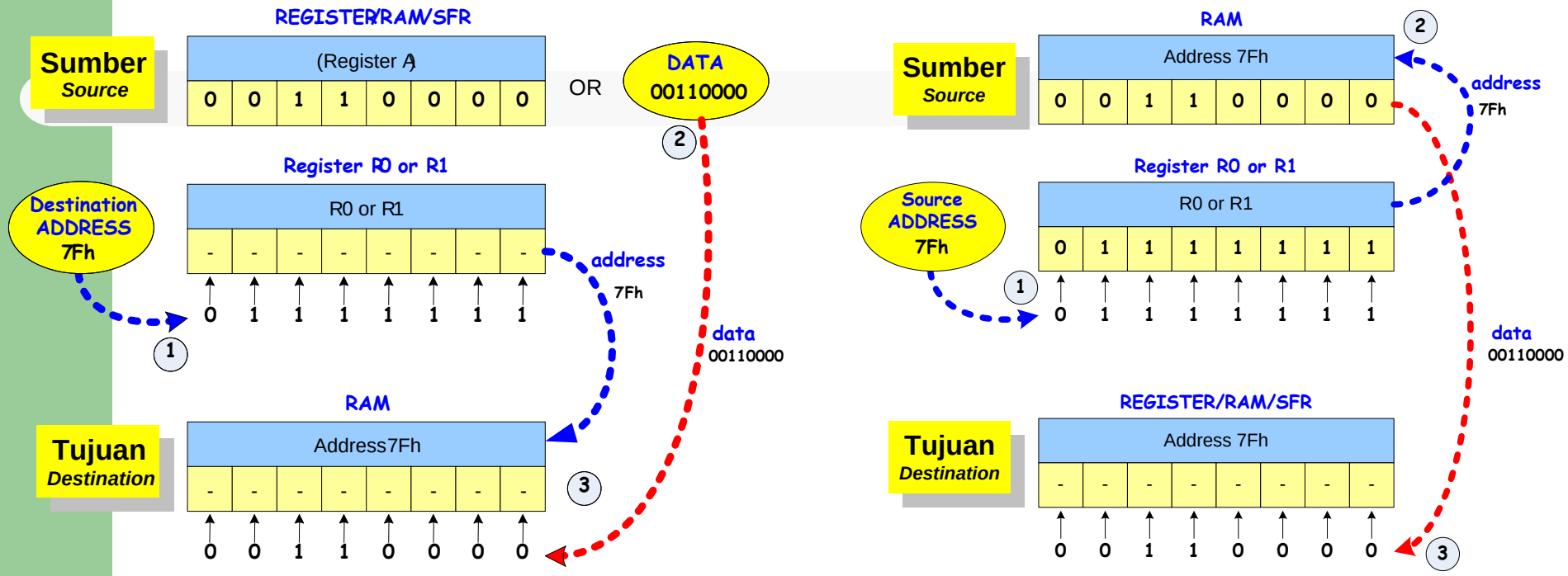
Direct Addressing Mode

Register Addressing : Data dari register/RAM/SFR sumber dipindahkan ke register/RAM/SFR tujuan

Contoh :

```
MOV     A,80h       ; Copy data from port 0 pins to A register
MOV     80h,A       ; Copy data from A register to the port 0
MOV     3Ah,#3Ah    ; Copy immediate data 3Ah to RAM location 3Ah
MOV     R0,12h      ; Copy data from RAM location 12h to R0 register
MOV     8Ch,R7      ; Copy data from R7 register to SFR timer 0 high byte
MOV     5Ch,A       ; Copy data from A register to RAM location 5Ch
MOV     0A8h,77h    ; Copy data from RAM location 77h to IE register
```

Indirect Addressing Mode



R0 atau R1 digunakan untuk menunjukkan Destination Address

```
MOV     @R0,#30h    ; Copy immediate data 30h to the address in R0
MOV     @R0,30h     ; Copy the content of address 30h to the address in R0
MOV     @R0,A       ; Copy the data in A to the address in R0
```

R0 atau R1 digunakan untuk menunjukkan Source Address

```
MOV     7Fh,@R0     ; Copy the content of the address in R0 to address 7Fh
MOV     A,@R0       ; Copy the content of the address in R0 to A
```

Latihan 1

1. Pindahkan data 34h ke R5, R6 dan R7 menggunakan immediate addressing mode.
2. Pindahkan data 35h ke register A kemudian copy data tersebut ke tiap register R5, R6 dan R7 menggunakan register addressing mode.
3. Pindahkan data 36h ke register R5, kemudian copy data tersebut ke tiap register R6 dan R7 menggunakan direct addressing mode.

Operasi Logika

Logika Boolean
Operasi Rotate & Swap

Logika Boolean

BOOLEAN OPERATOR	8051 MNEMONIC
AND	ANL (AND logical)
OR	ORL (OR logical)
NOT	CPL (Complement)
XOR	XRL (Exclusive OR logical)

AND

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

Contoh :

00001111
11110000 ANL
00000000

OR

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

Contoh :

00001111
11111111 ORL
11111111

XOR

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

Contoh :

11000011
11001100 XRL
00001111

NOT

A	/A
0	1
1	0

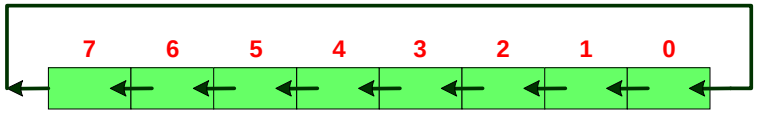
Contoh :

CPL 00001111

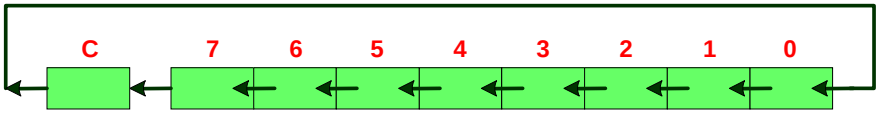
11110000

Operasi Rotate & Swap

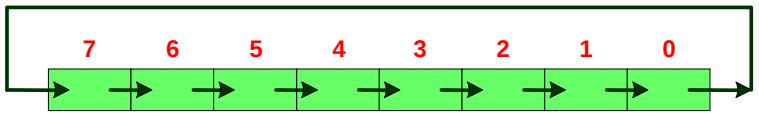
RL	**Rotate Left** : Geser isi register ke kiri satu bit.
RR	**Rotate Right** : Geser isi register ke kanan satu bit.
RLC	**Rotate Left and Carry** : Geser isi register dan carry flag ke kiri satu bit.
RRC	**Rotate Right and Carry** : Geser isi register dan carry flag ke kanan satu bit.
SWAP	**Swap** : Saling tukarkan nibble register, low nibble menjadi high nibble, dan sebaliknya.



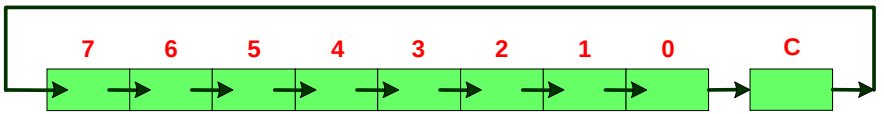
RL A



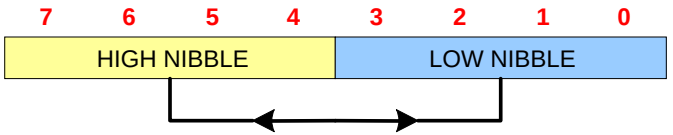
RLC A



RR A



RRC A



SWAP A

- **Contoh 1. Logika Boolean** :

```
MOV   A,#0FFh    ; A = FFh
MOV   R0,#77h    ; R0 = 77h
ANL   A,R0       ; A = 77h
MOV   15h,A      ; 15h = 77h
CPL   A          ; A = 88h
ORL   15h,#88h   ; 15h = FFh
XRL   A,15h      ; A = 77h
XRL   A,R0       ; A = 00h
ANL   A,15h      ; A = 00h
ORL   A,R0       ; A = 77h
CLR   A          ; A = 00h
XRL   15h,A      ; 15h = FFh
XRL   A,R0       ; A = 77h
```

- **Contoh 2. Rotate & Swap** :

```
MOV   A,#0A5h    ; A = 10100101b = A5h
RR    A          ; A = 11010010b = D2h
RR    A          ; A = 01101001b = 69h
RR    A          ; A = 10110100b = B4h
RR    A          ; A = 01011010b = 5Ah
SWAP  A          ; A = 10100101b = A5h
CLR   C          ; C = 0, A =10100101b = A5h
RRC   A          ; C = 1, A = 01010010b = 52h
RRC   A          ; C = 0, A = 10101001b = A9h
RL    A          ; A = 01010011b = 53h
RL    A          ; A = 10100110b = A6h
SWAP  A          ; C = 0, A = 01101010b = 6Ah
RLC   A          ; C = 0, A = 11010100b = D4h
RLC   A          ; C = 1, A = 10101000b = A8h
SWAP  A          ; C = 1, A = 10001010b = 8Ah
```

Operasi Aritmatika

Increment & Decrement
Penjumlahan
Pengurangan
Perkalian
Pembagian

Increment & Decrement

Increment : Tambah satu isi register
Decrement : Turunkan satu isi register

Mnemonic		Operation
INC	A	Tambah satu isi register A
INC	Rr	Tambah satu isi register Rr
INC	add	Tambah satu isi direct address
INC	@Rp	Tambah satu isi address dalam Rp
INC	DPTR	Tambah satu isi register 16 bit DPTR
DEC	A	Kurangi satu isi register A
DEC	Rr	Kurangi satu isi register Rr
DEC	add	Kurangi satu isi dirrect address
DEC	@Rp	Kurangi satu isi address dalam Rp

Contoh

```
MOV     A,#3Ah          ; A = 3Ah
DEC     A               ; A = 39h
MOV     R0,#15h         ; R0 = 15h
MOV     15h,#12h        ; Internal RAM 15h = 12h
INC     @R0             ; Internal RAM 15h = 13h
DEC     15h             ; Internal RAM 15h = 12h
INC     R0              ; R0 = 16h
MOV     16h,A           ; Internal RAM 16h = 39h
INC     @R0             ; Internal RAM 16h = 3Ah
MOV     DPTR,#12FFh     ; DPTR = 12FFh
INC     DPTR            ; DPTR = 1300h
DEC     83h             ; DPTR = 1200h(SFR 83h adalah byte DPH
```

Penjumlahan

Mnemonic		Operation
ADD	A,#n	Tambahkan A dengan angka n, simpan hasilnya di A
ADD	A,Rr	Tambahkan A dengan register Rr, simpan hasilnya di A
ADD	A,add	Tambahkan A dengan isi address, simpan hasilnya di A
ADD	A,@Rp	Tambahkan A dengan isi address dalam Rp, simpan hasilnya di A
ADDC	A,#n	Tambahkan A, angka n dan Carry, simpan hasilnya di A.
ADDC	A,Rr	Tambahkan A, isi register Rr dan Carry, simpan hasilnya di A.
ADDC	A,add	Tambahkan A, isi address dan Carry, simpan hasilnya di A.
ADDC	A,@Rp	Tambahkan A, isi address dalam Rp dan Carry, simpan hasilnya di A.

Catatan :
Carry flag (C) akan 1 jika terdapat carry pada bit ke-7.
Auxilliary Carry flag (AC) akan 1 jika terdapat carry pada bit ke-3.
Over Flow flag (OV) akan 1 jika terdapat carry pada bit ke-7, tapi tidak terdapat carry pada bit ke-6 atau terdapat carry pada bit ke-6 tetapi tidak pada bit ke-7, dimana dapat dieksperikan dengan operasi logika sbb :

OV = C7 XOR C6

Penjumlahan tak bertanda dan bertanda
Unsigned and Signed Addition

Unsigned number : 0 s/d 255d atau 00000000b s/d 11111111b
Signed number : -128d s/d +127d atau 1000000b s/d 01111111b

Penjumlahan unsigned number dapat menghasilkan carry flag jika hasil penjumlahan melebihi FFh, atau borrow flag jika angka pengurang lebih besar dari yang dikurangi.

• **Penjumlahan Tak Bertanda / *Unsigned Addition***

Carry flag dapat digunakan untuk mendeteksi hasil penjumlahan yang melebihi FFh. Jika carry = 1 setelah penjumlahan, maka carry tersebut dapat ditambahkan ke high byte sehingga hasil penjumlahan tidak hilang. Misalnya :

```
  95d      =  01011111b
189d       =  10111101b
284d       1  00011100b
```

C=1 dapat ditambahkan ke byte berikutnya (high byte)

•**Penjumlahan Bertanda / *Signed Addition***

Hasil penjumlahan bertanda tidak boleh melebihi -128d atau +127d. Aturan ini tidak menjadi masalah ketika angka yang dijumlahkan positif dan negatif, misalnya :

```
- 001d     = 11111111b
+027d      = 00011011b
+026d      = 00011010b = + 026d
```

Dari penjumlahan diatas terdapat carry dari bit ke-7, maka C=1. Pada bit ke-6 juga terdapat carry, maka OV=0. Pada penjumlahan ini tidak perlu manipulasi apapun karena hasil penjumlahannya sudah benar.

Jika kedua angka yang dijumlahkan adalah positif, maka ada kemungkinan hasil penjumlahan melebihi +127d, misalnya :

```
+100d      = 01100100b
+050d      = 00110010b
+150d      = 10010110b = - 106d
```

Ada kelebihan 22d dari batas +127d. Tidak ada carry dari bit ke-7, maka C=0, ada carry dari bit ke-6, maka OV=1.

Contoh penjumlahan dua angka positif yang tidak melebihi +127d adalah :

```
+045d    = 00101101b
+075d    = 01001011b
+120d    = 01111000b = + 120d
```

Dari penjumlahan diatas tidak terdapat carry dari bit ke-7 maupun bit ke-6, maka C=0 dan OV=0.

Penjumlahan dua angka negatif yang tidak melebihi -128d adalah sbb :

```
- 030d    = 11100010b
- 050d    = 11001110b
- 080d    = 10110000b = - 080d
```

Terdapat carry dari bit ke-7, maka C=1, ada carry dari bit ke-6, maka OV=0.

Penjumlahan dua angka negatif yang hasilnya melebihi -128d adalah sbb :

```
- 070d    = 10111010b
- 070d    = 10111010b
- 140d    = 01110100b = +116d  (Komplemen 116d = 139d)
```

Ada kelebihan -12d. Ada carry dari bit ke-7, maka C=1, tidak ada carry dari bit ke-6, maka OV=1.
Dari semua contoh diatas, manipulasi program perlu dilakukan sbb :

FLAGS		ACTION
C	OV	
0	0	None
0	1	Complement the sign
1	0	None
1	1	Complement the sign

Pengurangan

Mnemonic		Operation
SUBB	A,#n	Kurangi A dengan angka n dan C flag, simpan hasilnya di A
SUBB	A,Rr	Kurangi A dengan register Rr dan C flag, simpan hasilnya di A
SUBB	A,add	Kurangi A dengan isi address dan C flag, simpan hasilnya di A
SUBB	A,@Rp	Kurangi A dengan isi address dalam Rp dan C flag, simpan hasilnya di A

Catatan :
Carry flag (C) akan 1 jika terdapat borrow pada bit ke-7.
Auxilliary Carry flag (AC) akan 1 jika terdapat borrow pada bit ke-3.
Over Flow flag (OV) akan 1 jika terdapat borrow pada bit ke-7, tapi tidak terdapat borrow pada bit ke-6 atau terdapat borrow pada bit ke-6 tetapi tidak pada bit ke-7, dimana dapat dieksperikan dengan operasi logika sbb :

OV = C7 XOR C6

Pengurangan tak bertanda dan bertanda
Unsigned and Signed Substraction

Unsigned number : Positif 255d (C=0, A=FFh) s/d Negatif 255d (C=1, A=01h)
Signed number : -128d s/d +127d atau 1000000b s/d 01111111b

Oleh karena carry flag selalu disertakan dalam pengurangan, maka carry flag harus di set 0 agar tidak mempengaruhi pengurangan.
Pengurangan unsigned number dapat menghasilkan borrow jika angka pengurang lebih besar dari yang dikurangi.

• **Pengurangan Tak Bertanda / *Unsigned Substraction***

Berikut adalah contoh pengurangan dimana pengurang lebih besar dari yang dikurangi :

```
      015d      =  00001111b
SUBB  100d      =  01100100b
    - 085d      1  10101011b = 171d
```

C=1 dan OV=0. Komplemen 171d adalah 85d

Contoh lain :

```
      100d      =  01100100b
SUBB  015d      =  00001111b
      085d      0  01010101b = 85d
```

C=0 dan OV=0. Hasilnya sudah benar, jadi tidak perlu dikomplemenkan.

•**Pengurangan Bertanda / *Signed Substraction***

Hasil pengurangan bertanda tidak boleh melebihi -128d atau +127d.

Contoh 1 :

```
      +100d     = 01100100b    (Carry flag = 0 sebelum SUBB)
SUBB  +126d     = 01111110b
      - 026d    1 11100110b = - 026d
```

Dari pengurangan diatas terdapat borrow dari bit ke-7 dan ke-6, maka C=1 dan OV=0.Pada pengurangan ini tidak perlu manipulasi apapun karena hasil pengurangannya sudah benar.

Contoh 2:

```
        - 061d      =  11000011b   (Carry flag = 0 sebelum SUBB)
SUBB  - 116d        =  10001100b
        +055d          00110111b = +55d
```

Tidak ada borrow dari bit ke-7 atau bit ke-6, maka C=0 dan OV=0.

Contoh 3 :

```
        - 099d      =  10011101b   (Carry flag = 0 sebelum SUBB)
SUBB  +100d         =  01100100b
        - 199d         00111001b = +057d     (Komplemen 57d = 198d)
```

Tidak ada borrow dari bit ke-7 tapi ada borrow dari bit ke-6, maka C=0 dan OV=1. Hasilnya perlu dimanipulasi.

Contoh 4 :

```
        +087d      =  01010111b   (Carry flag = 0 sebelum SUBB)
SUBB  - 052d       =  11001100b
        +139d         10001011b = -117d     (Komplemen 117d = 138d)
```

Ada borrow dari bit ke-7 dan tidak ada borrow dari bit ke-6, maka C=1 dan OV=1. Hasilnya perlu dimanipulasi.

Catatan : **Jika OV=1 maka komplemenkan hasilnya.**

Perkalian

Mnemonic		Operation
MUL	AB	Kalikan isi A dengan isi B, simpan lower byte di A dan high byte di B.

Catatan :
Over Flow flag (OV) akan 1 jika A X B > FFh.
Carry flag (C) selalu 0.
Nilai maksimum hasil perkalian antara A dan B adalah FE01h, jika A dan B berisi FFh.

Contoh :

```
MOV     A,#7Bh    ; A=7Bh
MOV     B,#02h    ; B=02h
MUL     AB        ; A=F6h dan B=00h; OV=0
MOV     A,#0FEh   ; A=FEh
MUL     AB        ; A=00h dan B=00h; OV=0
```

Pembagian

Mnemonic		Operation
DIV	AB	Bagi isi A dengan isi B, simpan hasilnya di A dan simpan sisanya di B.

Catatan :
Over Flow flag (OV) akan 1 jika terjadi pembagian dengan 0.
Carry flag (C) selalu 0.

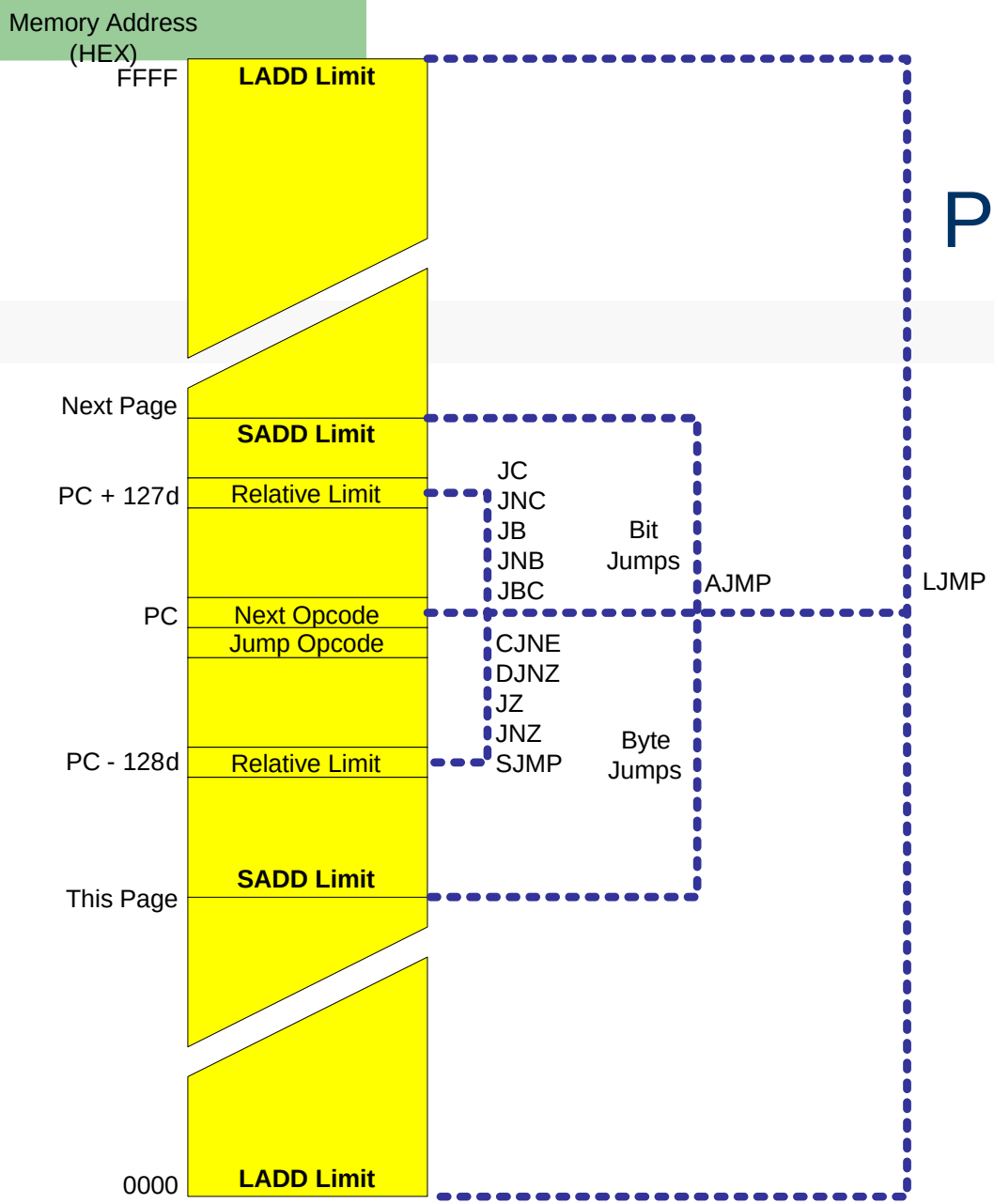
Contoh :

```
MOV       A,#0FFh   ; A=FFh
MOV       B,#2Ch    ; B=2Ch
DIV       AB        ; A=05h dan B=23h
DIV       AB        ; A=00h dan B=05h
DIV       AB        ; A=00h dan B=00h
DIV       AB        ; A=?? Dan B=??, OV=1
```

Operasi Percabangan

Rentang Percabangan
Jump
Call & Subrutin

Rentang Percabangan



Note :
1 page = 2 KByte

Jump

Instruksi JUMP digunakan untuk melompat ke instruksi tertentu.

Bit Jump

Mnemonic		Operation
JC	radd	Jump relative if the carry flag is set to 1
JNC	radd	Jump relative if the carry flag is clear to 0
JB	b,radd	Jump relative if addressable bit is set to 1
JNB	b,radd	Jump relative if addressable bit is clear to 0
JBC	b,radd	Jump relative if addressable bit is set, and clear the addressable bit to 0

Byte Jump

Mnemonic		Operation
CJNE	A,add,radd	Compare content of A register with the content of the direct address, if not equal jump to relative address, C=1 if A< add, C=0 if others.
CJNE	A,#n,radd	Compare content of A register with immediate number n, if not equal jump to relative address, C=1 if A< n, C=0 if others.
CJNE	Rn,#n,radd	Compare content of Rn register with immediate number n, if not equal jump to relative address, C=1 if Rn< n, C=0 if others.
CJNE	@Rp,#n,radd	Compare the content of address contained in register Rp to the number n, if not equal then jump to relative address, C=1 if content of Rn< n, C=0 if others.
DJNZ	Rn,radd	Decrement register Rn by 1 and jump to relative address if the result is not zero, no flag affected.
DJNZ	add,radd	Decrement the dirrect address by 1 and jump to relative address if the result is not zero, no flag affected.
JZ	radd	Jump to the relative address if A is 0.
JNZ	radd	Jump to the relative address if A is not 0.

Unconditional Jump

Mnemonic		Operation
JMP	@A+DPTR	Jump to address formed by adding A to the DPTR.
AJMP	sadd	Jump to absolute short range address sadd.
LJMP	ladd	Jump to absolute long range address ladd.
SJMP	radd	Jump to absolute relative address radd.
NOP		No operation. Do nothing and go to the next instruction.

Contoh 1. Bit Jump

```
        $MOD51
        ORG     0000H
        LJMP    MULAI
        ORG     0100H
MULAI:  MOV     A,#10H
        MOV     R0,A
ADDA:   ADD     A,R0
        JNC     ADDA
;
        MOV     A,#10H
ADDR:   ADD     A,R0
        JNB     CY,ADDR
        JBC     CY,MULAI
        SJMP    MULAI
        END
```

Contoh 2. Byte Jump

```
        $MOD51
        ORG     0000H
        LJMP    BGN
        ORG     0100H
BGN:    MOV     A,#30H
        MOV     50H,#00H
AGN:    CJNE    A,50H,AEQ
        SJMP    NXT
AEQ:    DJNZ    50H,AGN
        NOP
NXT:    MOV     R0,#0FFH
DWN:    DJNZ    R0,DWN
        MOV     A,R0
        JNZ     ABIG
        JZ      AZR0
ABIG:   NOP
;
        ORG     0116H
AZR0:   MOV     A,#08H
        MOV     DPTR,#1000H
        JMP     @A+DPTR
        NOP
        NOP
HERE:   AJMP    AZR0
        END
```

Call & Subrutin

Instruksi CALL digunakan untuk memanggil sub rutin tertentu.

Mnemonic		Operation
ACALL	label	Absolute Call the subroutine named label.
LCALL	label	Long Call the subroutine named label.
RET		Return from subroutine.

Contoh

```
            $MOD51
            ORG     0000H
            LJMP    MULAI
            ORG     0100H
MULAI:      MOV     P0,#0FFH
            CALL    DELAY
            MOV     P0,#00H
            CALL    DELAY
            SJMP    MULAI
;
DELAY:      MOV     R0,#5H
DL0:        MOV     R1,#0FH
DL1:        DJNZ    R1,DL1
            DJNZ    R0,DL0
            RET
            END
```

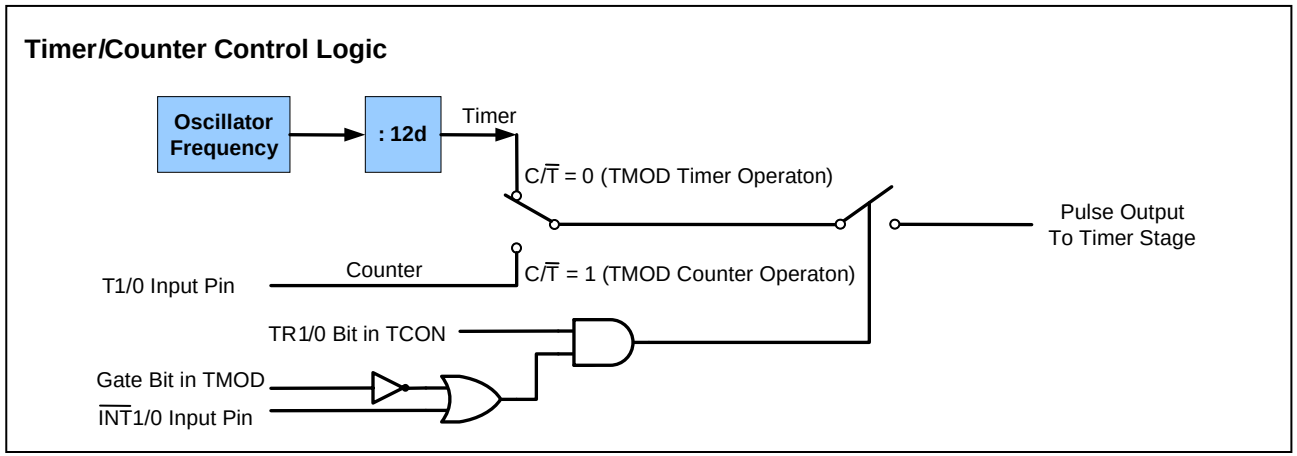
Operasi Timer / Counter

Register TCON
Register TMOD
Timer/Counter Interrupt
Mode Operasi Timer
Menggunakan Counter

Timer sangat diperlukan untuk membuat delay/tundaan waktu. AT89S51 menyediakan fasilitas timer 16 bit sebanyak 2 buah yaitu Timer 0 dan Timer 2. Timer ini juga bisa di fungsikan sebagai counter/pencacah.

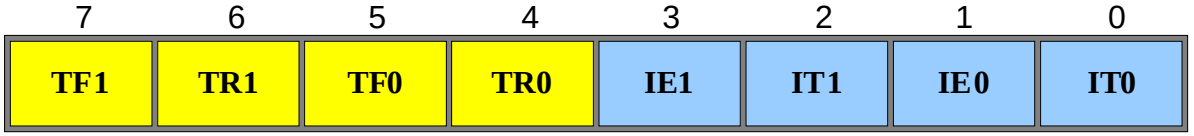
Timer bekerja dengan cara menghitung pulsa clock internal mikrokontroler yang dihasilkan dari rangkaian osilator. Jumlah pulsa clock akan dibandingkan dengan sebuah nilai yang terdapat dalam register timer (TH dan TL). Jika jumlah pulsa clock sama dengan nilai timer, maka sebuah interrupt akan terjadi (ditandai oleh flag TF). Interrupt ini dapat dipantau oleh program sebagai tanda bahwa timer telah overflow.

Counter bekerja dengan cara menghitung pulsa eksternal pada P3.4 (T0) dan P3.5 (T1). Jumlah pulsa ini akan disimpan dalam register timer (TH dan TL).



Timer akan menghitung pulsa clock dari osilator yang sebelumnya telah dibagi 12.
Agar berfungsi sebagai timer maka :
Bit C/T dalam TMOD harus 0 (timer operation)
Bit TRx dalam TCON harus 1 (timer run)
Bit Gate dalam TMOD harus 0 atau pin INTx harus 1.
Counter menghitung pulsa dari pin input T0 dan T1. Agar berfungsi sebagai counter maka :
Bit C/T dalam TMOD harus 1 (counter operation).
Bit TRx dalam TCON harus 1 (timer run)
Bit Gate dalam TMOD harus 0 atau pin INTx harus 1.

Register TCON

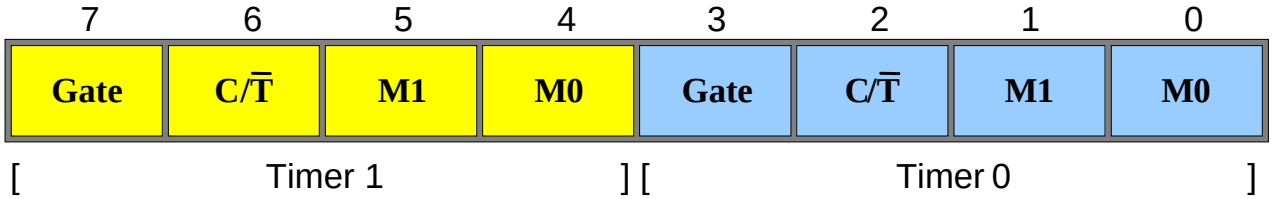


TCON / Timer Control Special Function Register

Bit	Symbol	Fuction
7	TF1	Timer 1 overflow flag. Set saat timer berubah dari satu ke nol. Clear saat prosesor mengeksekusi interrupt service routine pada address 001Bh.
6	TR1	Timer 1 run control bit. Set 1 oleh program agar timer mulai menghitung. Clear oleh program untuk menghentikan timer, bukan me-reset timer.
5	TF0	Timer 0 overflow flag. Set saat timer berubah dari satu ke nol. Clear saat prosesor mengeksekusi interrupt service routine pada address 000Bh.
4	TR0	Timer 0 run control bit. Set 1 oleh program agar timer mulai menghitung. Clear oleh program untuk menghentikan timer, bukan me-reset timer.
3	IE1	External interrupt 1 edge flag. Set 1 pada saat transisi sinyal high ke low diterima oleh port3 pin 3.3 (INT1). Clear saat prosesor mengeksekusi interrupt service routine pada address 0013h.Tidak terkait dengan operasi timer.
2	IT1	External interrupt 1 signal type control bit. Set 1 oleh program untuk mengaktifkan external interrupt 1 yang dipicu oleh sisi turun sinyal (falling edge/transisi high ke low). Clear oleh program untuk mengaktifkan sinyal low pada external interrupt 1 untuk menghasilkan sebuah interrupt.
1	IE0	External interrupt 0 edge flag. Set 1 pada saat transisi sinyal high ke low diterima oleh port3 pin 3.2 (INT0). Clear saat prosesor mengeksekusi interrupt service routine pada address 0003h.Tidak terkait dengan operasi timer.
0	IT0	External interrupt 0 signal type control bit. Set 1 oleh program untuk mengaktifkan external interrupt 0 yang dipicu oleh sisi turun sinyal (falling edge/transisi high ke low). Clear oleh program untuk mengaktifkan sinyal low pada external interrupt 0 untuk menghasilkan sebuah interrupt.

TCON is bit addressable as TCON.0 to TCON.7

Register TMOD



TMOD / Timer Mode Special Function Register

Bit	Symbol	Fuction
7/3	Gate	OR gate enable bit. Mengendalikan RUN/STOP timer 1/0. Set oleh program untuk mengaktifkan timer (RUN) jika bit TR1/0 pada TCON=1 dan sinyal pada pin INT0/1 high. Clear oleh program untuk mengaktifkan timer (RUN) jika bit TR1/0 pada TCON=1.
6/2	C/T	Set oleh program untuk membuat timer 1/0 berfungsi sebagai counter yang akan menghitung pulsa eksternal pada pin 3.5 (T1) atau 3.4 (T0). Clear oleh program untuk membuat timer 1/0 berfungsi sebagai timer yang akan menghitung pulsa clock internal
5/1	M1	Timer/counter operating mode select bit. Set/clear oleh program untuk memilih mode
4/0	M0	Timer/counter operating mode select bit. Set/clear oleh program untuk memilih mode

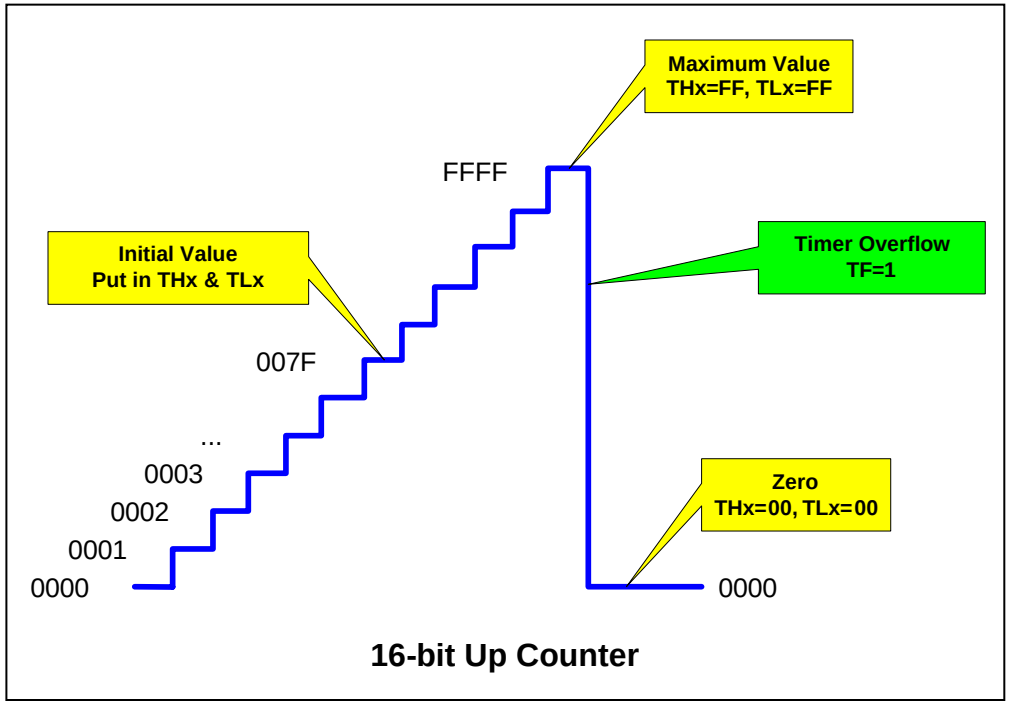
M1	M0	Mode
0	0	0
0	1	1
1	0	2
1	1	3

TMOD is not bit addressable

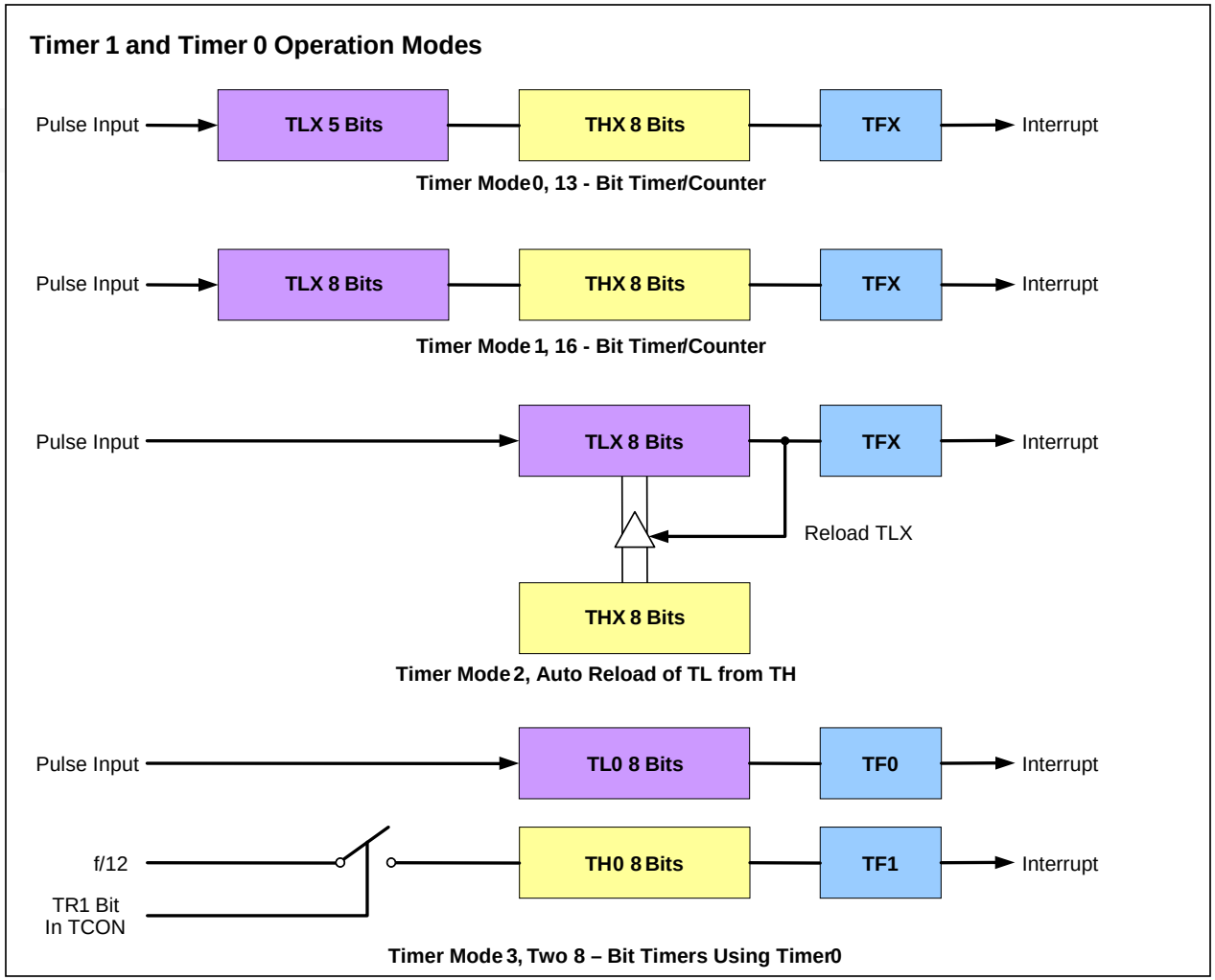
Timer/Counter Interrupt

Timer/Counter pada AT89S51 adalah sebuah Up Counter, nilai counternya akan naik ***(increment)*** dari nilai awalnya sampai nilai maksimumnya dan kembali ke nilai nol. Saat bergulir menjadi nol ***(overflow),*** maka sebuah timer flag akan bernilai 1. Flag ini dapat diuji oleh program untuk menandakan bahwa counter telah selesai menghitung, atau flag tersebut bisa digunakan untuk meng-interrupt program.

Nilai awal timer/counter harus dimasukkan dulu ke dalam timer register ***Timer High (TH)*** dan ***Timer Low (TL)*** sebelum timer/counter dijalankan.



Mode Operasi Timer



Pemilihan mode operasi timer ditentukan pada bit M1 dan M0 dalam register TMOD. Ada 4 mode operasi yaitu :
Mode 0 : 13-bit Timer/Counter
Mode 1 : 16-bit Timer/Counter
Mode 2 : 8-bit Autoreload Timer/Counter
Mode 3 : Two 8 bit Timer/Counter

Timer Mode 0. 13-bit Timer/Counter
Dengan mensetting M1&M0 = 00 dalam TMOD menyebabkan register THx berfungsi sebagai counter 8 bit dan register TLx berfungsi sebagai counter 5 bit. Ketika overflow, TF1x akan 1. Nilai maksimumnya adalah 8191d atau 1FFFh.

Timer Mode 1. 16-bit Timer/Counter
Register THx dan TLx masing-masing berfungsi sebagai counter 8 bit. Ketika overflow, TF1x akan 1 .Nilai maksimumnya adalah 65535d atau FFFFh.

Timer Mode 2. 8-bit Autoreload Timer/Counter
Register TLx berfungsi sebagai counter 8 bit. Register THx berfungsi mengisi ulang / autoreload register TLx ketika terjadi overflow (TFx=1).

Timer Mode 3. Two 8 bit Timer/Counter
Pada mode 3. Timer berfungsi sebagai counter 8 bit yang benar-benar terpisah satu sama lain. Timer 0 berfungsi sebagai timer sekaligus sebagai counter secara terpisah.
TL0 digunakan sebagai counter 8 bit yang menghitung pulsa eksternal, dengan timer flag TF0.
TH0 digunakan sebagai timer 8 bit yang menghitung pulsa clock internal, dengan timer flag TF1.

Pada mode 3, Timer 1 tidak dapat digunakan sebab timer flag TF1 digunakan sebagai timer flag TH0.

Timer/Counter dapat dihidup-matikan secara program dengan mengatur TRx maupun secara hardware dengan memberikan logika 0 pada pin INTx. Berikut adalah tabel nilai TMOD sesuai dengan mode dan kontrol timer/counter.

Timer 0 sebagai timer

Mode	Fungsi Timer 0	TMOD	
		Kontrol Internal	Kontrol Eksternal
0	13 bit timer	00h	08h
1	16 bit timer	01h	09h
2	8 bit autoreload	02h	0Ah
3	Two 8 bit timer	03h	0Bh

Timer 0 sebagai counter

Mode	Fungsi Counter 0	TMOD	
		Kontrol Internal	Kontrol Eksternal
0	13 bit counter	04h	0Ch
1	16 bit counter	05h	0Dh
2	8 bit autoreload	06h	0Eh
3	One 8 bit counter	07h	0Fh

Timer 1 sebagai timer

Mode	Fungsi Timer 1	TMOD	
		Kontrol Internal	Kontrol Eksternal
0	13 bit timer	00h	80h
1	16 bit timer	10h	90h
2	8 bit autoreload	20h	A0h
-	-	-	-

Timer 1 sebagai counter

Mode	Fungsi Counter 1	TMOD	
		Kontrol Internal	Kontrol Eksternal
0	13 bit counter	40h	C0h
1	16 bit counter	50h	D0h
2	8 bit autoreload	60h	E0h
-	-	-	-

Secara umum delay waktu timer (T) dapat dihitung dengan persamaan sebagai berikut :

1. Sebagai timer 8 bit.
   T=(255-TLx)*1uS

2. Sebagai timer 13 bit.
   T=(8191-THxTLx)*1uS

3. Sebagai timer 16 bit.
   T=(65535-THxTLx)*1uS

Dengan catatan frekuensi crystal yang digunakan adalah 12 MHz.

Contoh :
Diinginkan delay waktu 10 mS menggunakan timer 16 bit. Maka nilai THx dan TLx adalah :

T=(65535-THxTLx)*1uS

THxTLx=65535-(T/1uS)
THxTLx=65535-(10mS/1uS)
THxTLx=65535-10000
THxTLx=55535d=D8EFh

Maka **THx=D8h** dan **TLx=EFh**

Contoh 1. Timer Mode 0

```
                $mod51
                ORG         0000H
                LJMP        MULAI
                ORG         0100H
MULAI:          MOV         TMOD,#00H     ; Timer 0 pada mode 0
                MOV         TL0,#17H      ; T=(8191-THx TLx)*1uS
                MOV         TH0,#1CH      ; Delay 1000uS
                SETB        TR0
LOOP:           JBC         TF0,SELESAI
                SJMP        LOOP
SELESAI:
                SJMP        MULAI
                END
```

Contoh 2. Timer Mode 1

```
                $mod51
                ORG         0000H
                LJMP        MULAI
                ORG         0100H
MULAI:          SETB        P0.0
                CALL        DELAY
                CLR         P0.0
                CALL        DELAY
                SJMP        MULAI
;-------Delay 10000uS--------
DELAY:          MOV         TMOD,#01H     ; Timer 0 pada mode 1
                MOV         TH0,#0D8H     ; T=(65535-THx TLx)*1uS
                MOV         TL0,#0EFH     ; THxTLx=(65535-10000)*1uS
                SETB        TR0
LOOP:           JBC         TF0,SELESAI
                SJMP        LOOP
SELESAI:        CLR         TR0
                RET
                END
```

Contoh 3. Timer Mode 2

```
            $mod51
            ORG         0000H
            LJMP        MULAI
            ORG         0100H
MULAI:      MOV         TMOD,#02H     ; Timer 0 pada mode 2
                                      ; T=(255-THx)*1uS
            MOV         TH0,#9BH      ; Delay 100uS
            SETB        TR0
LOOP:       JBC         TF0,SELESAI
            SJMP        LOOP
SELESAI:    SJMP        MULAI
            END
```

Contoh 4. Timer Mode 3

```
            $mod51
            ORG         0000H
            LJMP        MULAI
            ORG         0100H
MULAI:      MOV         TMOD,#03H     ; Timer 0 pada mode 2
                                      ; T=(255-THx)*1uS
            MOV         TH0,#9BH      ; Delay 100uS
            MOV         TL0,#0CDH     ; Delay  50uS
            SETB        TR0
            SETB        TR1
LOOP:       JBC         TF0,OVER1
            JBC         TF1,OVER2
            SJMP        LOOP
OVER1:      MOV         TL0,#0CDH
            SJMP        LOOP
OVER2:      MOV         TH0,#9BH
            SJMP        LOOP
            END
```

Contoh 5. Counter

```
            $mod51
            ORG         0000H
            LJMP        MULAI
            ORG         0100H
MULAI:      MOV         TMOD,#05H     ;COUNTER MODE 1
            SETB        TR0
LOOP:       MOV         A,TL0
            MOV         P1,A
            SJMP        LOOP
            END
```

Latihan

Rancanglah sebuah program untuk menghitung banyaknya pulsa per detik yang hasilnya ditampilkan pada modul MT-LED.

Aplikasi Seven Segmen

Dasar Seven Segment
BCD to 7 Segment
Metoda Scanning

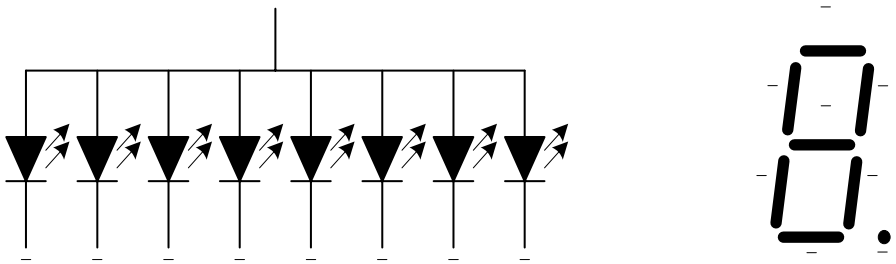
Dasar Seven Segment

Seven segment adalah tampilan angka yang terdiri dari 7 LED yang disusun membentuk angka 8 ditambah 1 LED sebagai titik (dot). Ada dua tipe 7 Segment yaitu : Common Anode dan Common Cathode

Common Anode (CA)

Pada 7 segment CA semua anoda LED dihubungkan menjadi satu dan disebut sebagai Common Anode, sementara katoda LED diberi nama a, b, c, d, e, f, g dan dp (dot/titik).

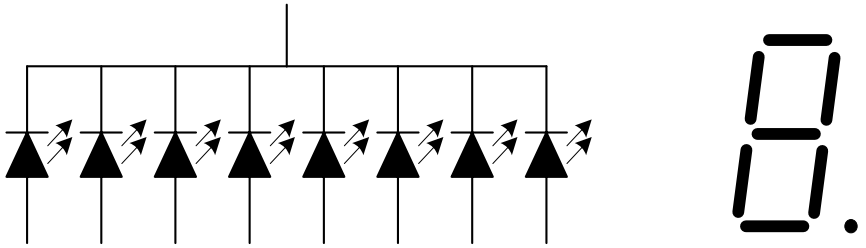
Tanda bar diatas menunjukkan bahwa pin tersebut adalah aktif low. Sebagai contoh untuk membentuk angka 2 maka pin common diberi tegangan + sedangkan pin a, c, d, f dan g diberi tegangan 0 volt.
Besarnya tegangan forward (Vf) LED dapat dilihat dari lembaran data sheet tiap produk seven segment.



Konfigurasi Seven Segment Common Anode

Common Cathode (CC)

Pada 7 segment CC semua katoda LED dihubungkan menjadi satu dan disebut sebagai Common Cathode, sementara katoda LED diberi nama a, b, c, d, e, f, g dan dp (dot/titik). Sebagai contoh untuk membentuk angka 1 maka pin common diberi tegangan 0 volt sedangkan pin a dan b diberi tegangan +.



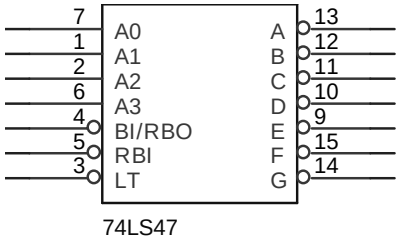
Konfigurasi Seven Segment Common Cathode

Common Cathode

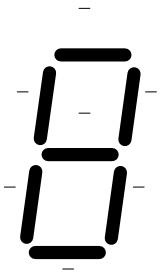
BCD to 7 Segment

BCD (Binary Coded Decimal) to Seven Segment adalah sebuah decoder yang dapat mengubah kode biner menjadi tampilan angka pada seven segment. Ada dua tipe BCD to 7 segment yaitu : 74LS47 dan 74LS48.

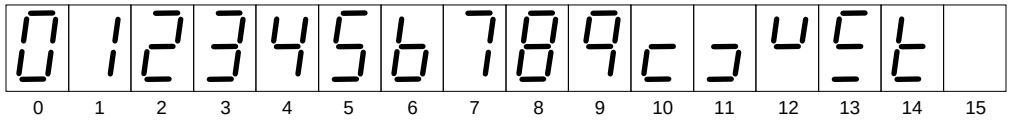
74LS47 (BCD to 7 Segment Common Anode)



74LS47



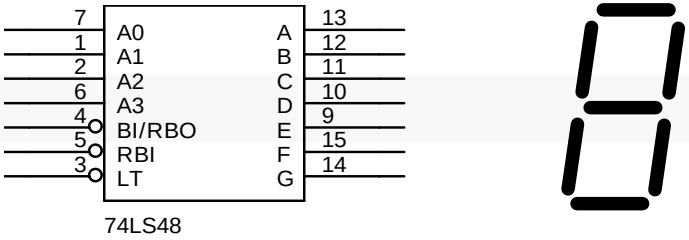
Nama Pin	Deskripsi
A0-A3	Input BCD
$\overline{RBI}$	Ripple Blanking Input (Aktif Low)
$\overline{LT}$	Lamp Test Input (Aktif Low)
$\overline{BI}$ / $\overline{RBO}$	Blanking Input/Ripple Blanking Output (Aktif Low)
$\overline{a}$ - $\overline{g}$	Output Segment



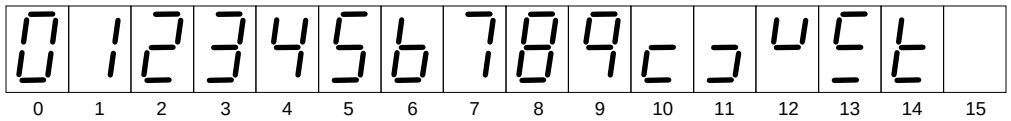
Tabel Kebenaran 74LS47

Tabel Kebenaran															
Desimal / Fungsi	Input							Output							
	$\overline{LT}$	$\overline{RBI}$	A3	A2	A1	A0	$\overline{BI}$ / $\overline{RBO}$	$\overline{a}$	$\overline{b}$	$\overline{c}$	$\overline{d}$	$\overline{e}$	$\overline{f}$	$\overline{g}$
0	1	1	0	0	0	0	1	0	0	0	0	0	0	1
1	1	X	0	0	0	1	1	1	0	0	1	1	1	1
2	1	X	0	0	1	0	1	0	0	1	0	0	1	0
3	1	X	0	0	1	1	1	0	0	0	0	1	1	0
4	1	X	0	1	0	0	1	1	0	0	1	1	0	0
5	1	X	0	1	0	1	1	0	1	0	0	1	0	0
6	1	X	0	1	1	0	1	1	1	0	0	0	0	0
7	1	X	0	1	1	1	1	0	0	0	1	1	1	1
8	1	X	1	0	0	0	1	0	0	0	0	0	0	0
9	1	X	1	0	0	1	1	0	0	0	1	1	0	0
10	1	X	1	0	1	0	1	1	1	1	0	0	1	0
11	1	X	1	0	1	1	1	1	1	0	0	1	1	0
12	1	X	1	1	0	0	1	1	0	1	1	1	0	0
13	1	X	1	1	0	1	1	0	1	1	0	1	0	0
14	1	X	1	1	1	0	1	1	1	1	0	0	0	0
15	1	X	1	1	1	1	1	1	1	1	1	1	1	1
$\overline{BI}$	X	X	X	X	X	X	0	1	1	1	1	1	1	1
$\overline{RBI}$	1	0	0	0	0	0	0	1	1	1	1	1	1	1
$\overline{LT}$	0	X	X	X	X	X	1	0	0	0	0	0	0	0

74LS48 (BCD to 7 Segment Common Cathode)



Nama Pin	Deskripsi
A0-A3	Input BCD
$\overline{RBI}$	Ripple Blanking Input (Aktif Low)
$\overline{LT}$	Lamp Test Input (Aktif Low)
$\overline{BI}$ / $\overline{RBO}$	Blanking Input/Ripple Blanking Output (Aktif Low)
a - g	Output Segment

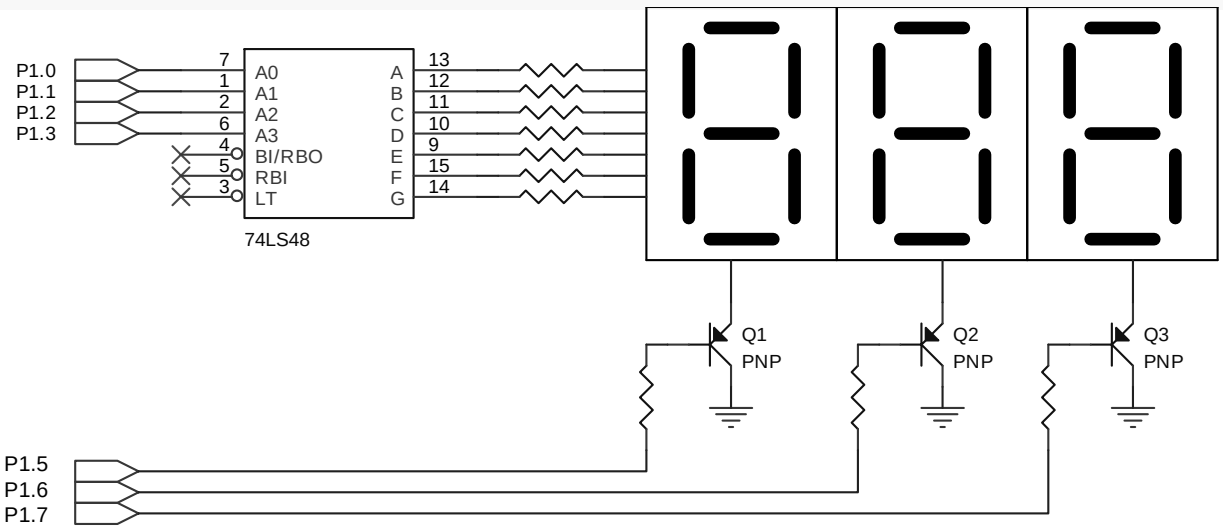


Tabel Kebenaran 74LS48

Tabel Kebenaran															
Desimal / Fungsi	Input							Output							
	$\overline{LT}$	$\overline{RBI}$	A3	A2	A1	A0	$\overline{BI}$ / $\overline{RBO}$	a	b	c	d	e	f	g
0	1	1	0	0	0	0	1	1	1	1	1	1	1	0
1	1	X	0	0	0	1	1	0	1	1	0	0	0	0
2	1	X	0	0	1	0	1	1	1	0	1	1	0	1
3	1	X	0	0	1	1	1	1	1	1	1	0	0	1
4	1	X	0	1	0	0	1	0	1	1	0	0	1	1
5	1	X	0	1	0	1	1	1	0	1	1	0	1	1
6	1	X	0	1	1	0	1	0	0	1	1	1	1	1
7	1	X	0	1	1	1	1	1	1	1	0	0	0	0
8	1	X	1	0	0	0	1	1	1	1	1	1	1	1
9	1	X	1	0	0	1	1	1	1	1	0	0	1	1
10	1	X	1	0	1	0	1	0	0	0	1	1	0	1
11	1	X	1	0	1	1	1	0	0	1	1	0	0	1
12	1	X	1	1	0	0	1	0	1	0	0	0	1	1
13	1	X	1	1	0	1	1	1	0	0	1	0	1	1
14	1	X	1	1	1	0	1	0	0	0	1	1	1	1
15	1	X	1	1	1	1	1	0	0	0	0	0	0	0
$\overline{BI}$	X	X	X	X	X	X	0	0	0	0	0	0	0	0
$\overline{RBI}$	1	0	0	0	0	0	0	0	0	0	0	0	0	0
$\overline{LT}$	0	X	X	X	X	X	1	1	1	1	1	1	1	1

Metoda Scanning

Metoda scanning digunakan untuk melakukan penghematan jalur data yang diperlukan untuk mengendalikan seven segmen yang jumlahnya lebih dari satu buah seperti pada gambar dibawah ini.

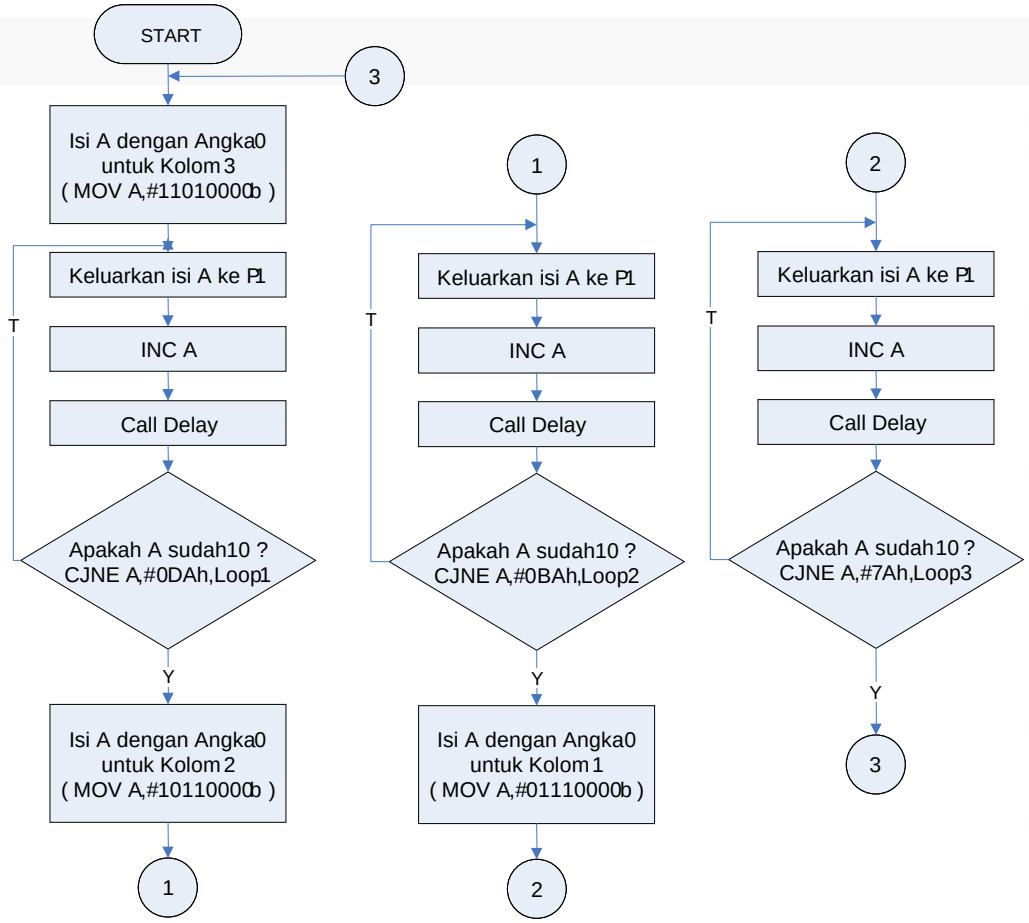


Dengan metoda scanning, semua output segmen dari 74LS48 (a sampai g) dihubungkan ke semua seven segmen. Dengan demikian data akan diterima oleh semua seven segmen secara bersamaan. Yang harus dilakukan selanjutnya adalah memilih common mana yang akan diaktifkan. Dengan mengaktifkan common secara bergantian dan dilakukan dalam frekuensi yang cepat (50 Hz) maka seolah-olah akan dilihat tiga digit angka yang menyala bersamaan.

Contoh 1 :

Tiga digit seven segment didesain untuk menampilkan angka 0-9 pada tiap kolomnya secara bergantian (hanya satu kolom yang menyala). Dimulai dari kolom ketiga, kemudian kolom kedua dan kolom pertama, demikian berulang-ulang.

Flowchart Program :



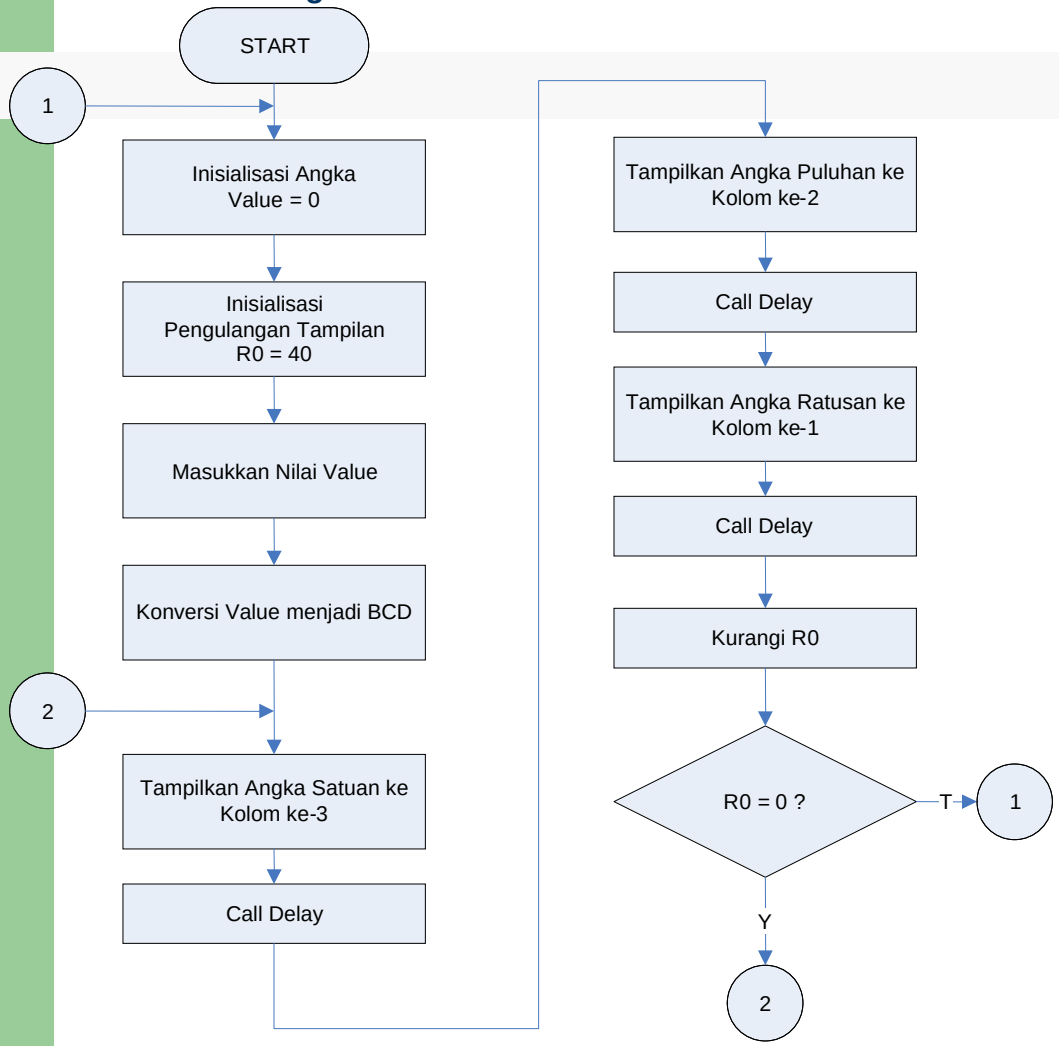
Listing Program :

```
                $mod51
                ORG     0000H
                LJMP    MULAI
                ORG     0100H
MULAI:          MOV     A,#11010000B
LOOP1:          MOV     P1,A
                INC     A
                ACALL   DELAY
                CJNE    A,#0DAH,LOOP1
                MOV     A,#10110000B
LOOP2:          MOV     P1,A
                INC     A
                ACALL   DELAY
                CJNE    A,#0BAH,LOOP2
                MOV     A,#01110000B
LOOP3:          MOV     P1,A
                INC     A
                ACALL   DELAY
                CJNE    A,#7AH,LOOP3
                SJMP    MULAI
;---------------------------------
; DELAY SUBROUTINE
;---------------------------------
DELAY:          MOV     R5,#0FFH
                MOV     R6,#0FFH
                MOV     R7,#5
DLY:            DJNZ    R5,DLY
                DJNZ    R6,DLY
                DJNZ    R7,DLY
                RET
                END
```

Contoh 2 :

Tiga digit seven segment didesain untuk menampilkan angka 000 sampai 255 kemudian berulang kembali dari 000.

Flowchart Program :



Listing Program :

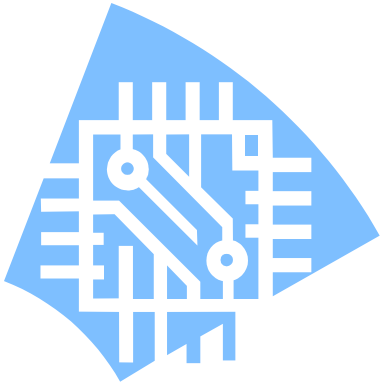
```
;---------- VARIABLES ----------
ANGKA1      EQU     30H
ANGKA2      EQU     31H
ANGKA3      EQU     32H
VALUE       EQU     33H
;---------- MAIN PROGRAM ---------
            ORG     0000H
            LJMP    MULAI
            ORG     0100H
MULAI:      MOV     VALUE,#0
LOOP1:      MOV     R0,#40
LOOP2:      MOV     A,VALUE
            MOV     B,#0AH
            DIV     AB
            MOV     ANGKA3,B
            MOV     B,#0AH
            DIV     AB
            MOV     ANGKA2,B
            MOV     ANGKA1,A
            MOV     A,ANGKA3
            ANL     A,#0FH
            ORL     A,#70H
            MOV     P1,A
            CALL    DELAY
            MOV     A,ANGKA2
            ANL     A,#0FH
            ORL     A,#0B0H
            MOV     P1,A
            CALL    DELAY
            MOV     A,ANGKA1
            ANL     A,#0FH
            ORL     A,#0D0H
            MOV     P1,A
            CALL    DELAY
            DJNZ    R0,LOOP2
            INC     VALUE
            SJMP    LOOP1
```

Latihan

Rancanglah sebuah jam digital 3 digit menggunakan timer dengan ketentuan seperti gambar berikut ini :

Aplikasi Motor Stepper

Dasar Motor Stepper
Operasi Pergeseran Step
Kendali Kecepatan Motor Stepper
Kendali Arah Putaran Motor Stepper
Kendali Posisi
Kendali Torsi



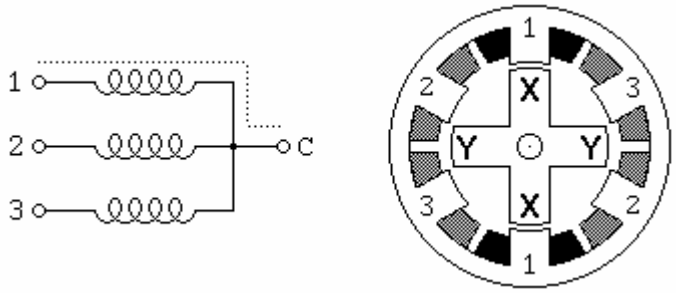
Dasar Motor Stepper

Motor stepper adalah sebuah peralatan elektromekanik yang mengubah pulsa elektrik menjadi pergerakan mekanik. Shaft atau kumparan motor stepper berputar per step ketika pulsa elektrik dimasukkan ke kumparan tersebut dengan urutan yang benar. Urutan pemberian pulsa ke motor stepper akan menyebabkan arah putaran yang berbeda. Sedangkan besarnya frekuensi dari pulsa akan mempengaruhi kecepatan putaran motor stepper.

Secara umum motor stepper terbagi menjadi 2 tipe yaitu :

1. ***Variable-reluctance***
2. ***Permanent-magnet***

A. Variable-reluctance (VR) Stepper Motor



Gambar 2.1 Motor Stepper Tipe Variable Reluctance

Jika anda memiliki motor dengan tiga kumparan, yang ketiga kumparannya dihubungkan seperti pada gambar diatas dengan satu terminal terhubung ke semua kumparan *(common)*, maka motor tersebut adalah motor stepper tipe variable reluctance (VR). Dalam prakteknya, kabel *common* dihubungkan ke terminal positif power supply dan kabel yang lain dihubungkan ke ground secara berurutan agar motor dapat berputar.
Bagian rotor yang berbentuk silang *(cross)* pada gambar 2.1 dapat berputar sebesar 30 derajat per step. Rotor yang memiliki 4 gigi dan stator yang memiliki 6 kutub, dimana setiap kumparan terbagi menjadi 2 kutub yang posisinya berseberangan. Ketika kumparan nomor 1 aktif *(energize)*, gigi rotor yang bertanda X akan tertarik ke kutub kumparan 1. Pada kondisi ini, posisi rotor seperti terlihat pada gambar 2.1. Jika arus listrik yang mengalir pada kumparan 1 dimatikan dan kumparan 2 dinyalakan, rotor akan berputar 30 derajat searah jarum jam sehingga gigi rotor Y akan menghadap kutub 2
Untuk memutar motor secara kontinyu, kita hanya perlu memasukkan power ke 3 kumparan secara berurutan. Dengan asumsi bahwa logika 1 berarti arus mengalir pada kumparan motor, maka berikut adalah urutan yang harus dipenuhi agar motor dapat berputar sebanyak 24 step atau 2 putaran :

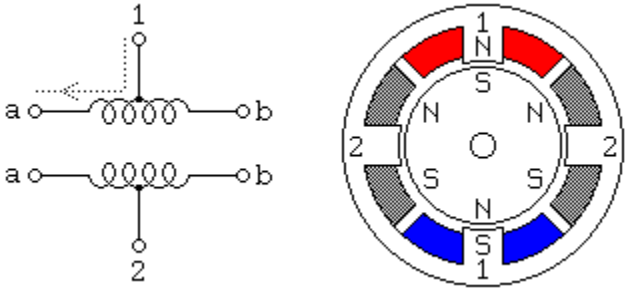
Kumparan 1 100100100100100100100100
Kumparan 2 010010010010010010010010
Kumparan 3 001001001001001001001001
waktu --->

Motor stepper tipe VR ada juga yang terdiri dari 4 dan 5 kumparan, memerlukan 5 atau 6 kabel. Prinsip pengendaliannya sama dengan motor stepper dengan 3 kumparan. Motor stepper pada gambar 2.1 memberikan pergerakan 30 derajat per step, dengan menambah jumlah kutub dan gigi rotor akan didapatkan *step angle* yang lebih kecil.

B. Permanent-magnet (PM) Stepper Motor

Berbeda dengan rotor motor stepper tipe *variable reluctance* yang terbuat dari besi, rotor motor stepper *permanent magnet* terbuat dari magnet permanen dengan dua kutub utara dan selatan.

Motor Stepper Unipolar

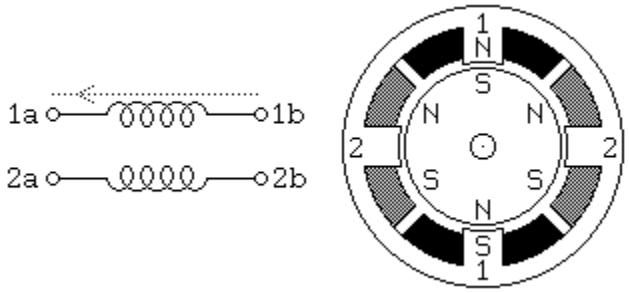


Gambar 2.2 Motor Stepper Unipolar

Motor stepper unipolar baik tipe 5 atau 6 kabel biasanya dihubungkan seperti pada gambar 2.2, dengan sebuah *center tap* pada tiap kumparan. Pada penggunaannya, *center tap* dihubungkan ke supply positif, dan dua ujung kumparan lainnya dihubungkan ke ground.
Bagian rotor motor pada gambar 2.2 dibuat dari magnet permanent dengan 6 kutub, 3 kutub utara dan 3 kutub selatan. Seperti terlihat pada gambar, arus mengalir dari *center tap* kumparan 1 ke terminal a menyebabkan kutub stator yang atas menjadi berkutub utara dan kutub stator yang bawah berkutub selatan. Kondisi ini menyebabkan rotor berada pada posisi seperti gambar 2.2. Jika arus pada kumparan 1 dimatikan dan kumparan 2 dinyalakan, maka rotor akan berputar 30 derajat, atau 1 step.
Untuk berputar secara kontinyu, kita hanya perlu menghubungkan supplay power ke 2 kumparan secara berurutan. Dengan asumsi bahwa logika 1 berarti arus mengalir pada kumparan motor, maka berikut adalah urutan yang harus dipenuhi agar motor dapat berputar sebanyak 24 step atau 2 putaran :

Kumparan 1a 100010001000100010001
Kumparan 1b 001000100010001000100
Kumparan 2a 010001000100010001000
Kumparan 2b 000100010001000100010
waktu --->

Motor Stepper Bipolar



Gambar 2.3 Motor Stepper Bipolar

Motor stepper bipolar dengan magnet permanent pada dasarnya dibentuk dengan mekanisme yang sama dengan motor stepper unipolar, tapi kedua kumparan dihubungkan dengan lebih sedarhana tanpa ada center tap seperti terlihat pada gambar 2.3. Agar motor dapat berputar secara kontinyu, maka berikut ini adalah urutan pemberian polaritas tegangan pada tiap terminal :

```
Terminal 1a +---+---+---+---  ++--++--++--++--
Terminal 1b --+---+---+---+-  --++--++--++--++
Terminal 2a -+---+---+---+--  -++--++--++--++-
Terminal 2b ---+---+---+---+  +--++--++--++--+
                  waktu --->
```

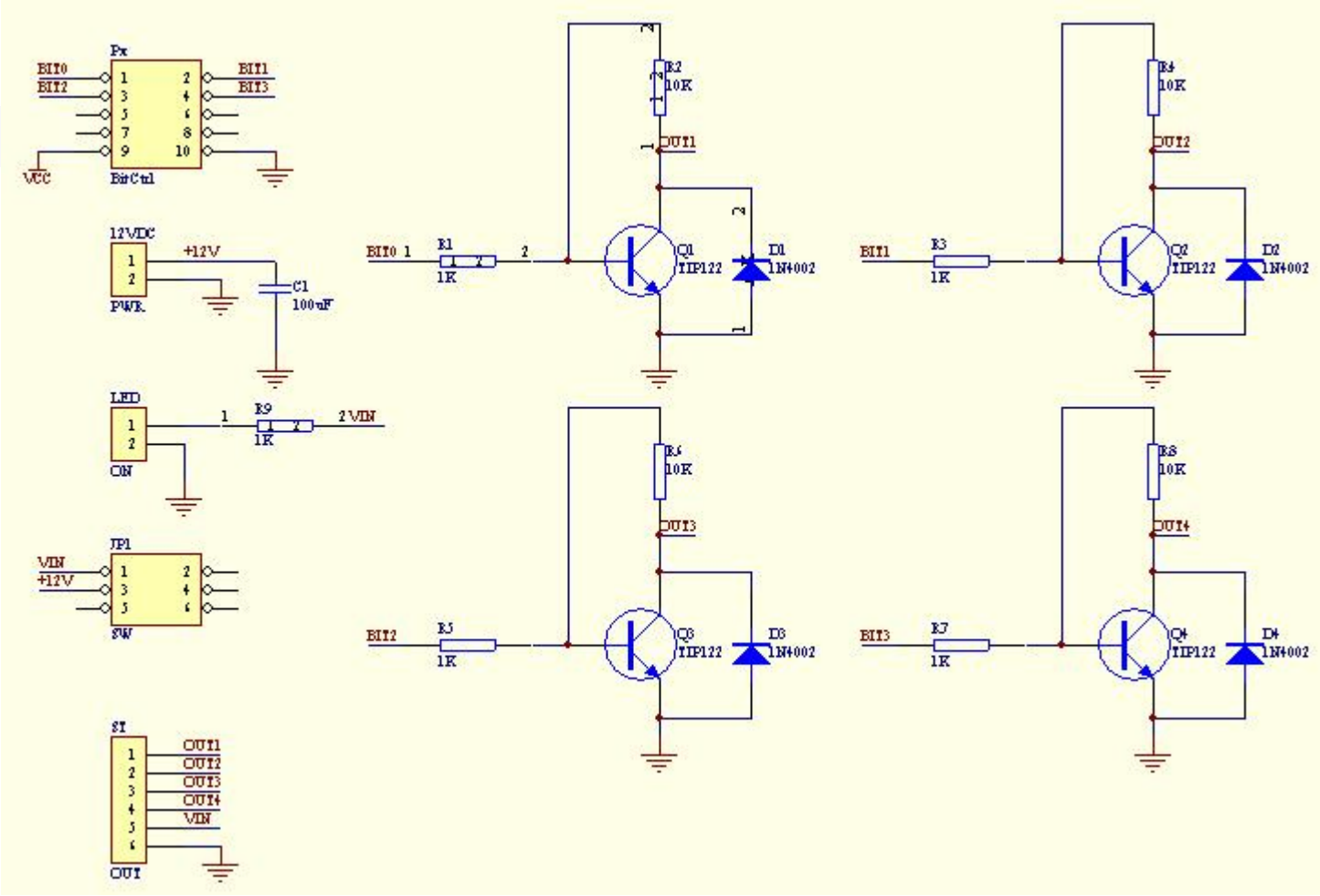
Step Angle / SA

Motor stepper bergerak per step. Setiap bergerak satu step, motor stepper akan berputar beberapa derajat sesuai dengan step anglenya. Step angle tergantung dari jumlah kutub magnet motor stepper. Jumlah putaran yang diperlukan agar motor stepper bergerak 1 putaran penuh (360^0) adalah :

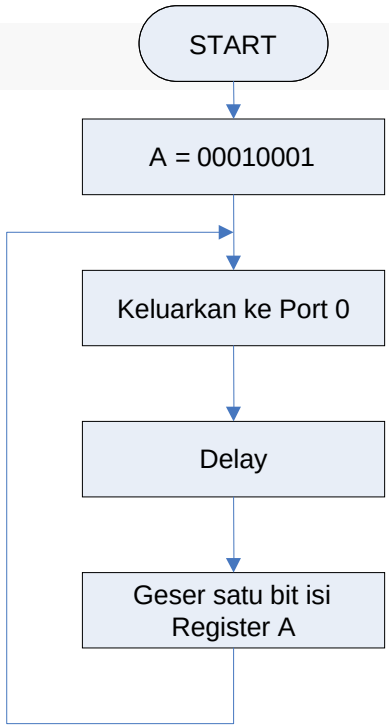
Step = 360^0 / Step Angle

Misalnya, sebuah motor stepper memiliki SA=1,80 maka untuk untuk berputar satu putaran penuh memerlukan jumlah step sebanyak : 360 / 1,8 = 200 step

Skema Rangkaian Motor Stepper



Operasi Pergeseran Step




```
$mod51
ORG         0000H
            LJMP        MULAI
            ORG         0100H
MULAI:      MOV         A,#00010001B
LOOP:       MOV         P0,A
            CALL        DELAY
            RL          A
            SJMP        LOOP
;-----------------------------------
; DELAY SUBROUTINE
;-----------------------------------
DELAY:      MOV         R5,#0FFH
            MOV         R6,#0FH
            MOV         R7,#05H
DLY:        DJNZ        R5,DLY
            DJNZ        R6,DLY
            DJNZ        R7,DLY
            RET
            END
```

Kendali Kecepatan Motor Stepper

Kecepatan motor stepper ditentukan oleh kecepatan aktifasi kumparannya. Secara program, ini bisa dilakukan dengan mengubah delay waktu pergeseran tiap bitnya.
Semakin cepat delay waktunya, kecepatan motor stepper juga akan bertambah.

```
$mod51
ORG       0000H
          LJMP      MULAI
          ORG       0100H
MULAI:    MOV       A,#00010001B
LOOP:     MOV       P0,A
          CALL      DELAY
          RL        A
          SJMP      LOOP
;----------------------------------
; DELAY SUBROUTINE
;----------------------------------
DELAY:    MOV       R5,#0FFH
          MOV       R6,#0FH
          MOV       R7,#05H
DLY:      DJNZ      R5,DLY
          DJNZ      R6,DLY
          DJNZ      R7,DLY
          RET
          END
```

Pada rutin delay, jika instruksi :

MOV R7,#05H diganti menjadi
MOV R7,#01H

Maka kecepatan motor stepper akan bertambah

Kendali Arah Putaran Motor Stepper

Arah putaran motor stepper ditentukan oleh arah urutan aktifasi kumparannya. Secara program, ini bisa dilakukan dengan mengubah arah pergeseran bit.

Jika arah pergesarannya ke kiri, maka motor stepper akan berputar kearah kiri pula (CCW)
Jika arah pergeserannya ke kanan, maka motor stepper akan berputar ke arah kanan (CW)

```
$mod51
ORG         0000H
            LJMP        MULAI
            ORG         0100H
MULAI:      MOV         A,#00010001B
LOOP:       MOV         P0,A
            CALL        DELAY
            RL          A
            SJMP        LOOP
;---------------------------------
; DELAY SUBROUTINE
;---------------------------------
DELAY:      MOV         R5,#0FFH
            MOV         R6,#0FH
            MOV         R7,#05H
DLY:        DJNZ        R5,DLY
            DJNZ        R6,DLY
            DJNZ        R7,DLY
            RET
            END
```

Pada program, jika instruksi :

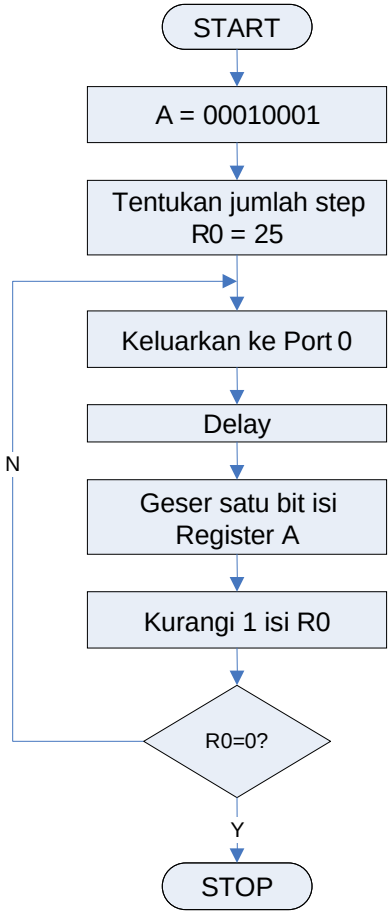
RL **A** diganti menjadi
RR **A**

Maka arah putaran motor stepper akan berubah

Kendali Posisi

Derajat putaran motor stepper ditentukan oleh banyaknya jumlah pergeseran aktifasi kumparannya. Secara program, ini bisa dilakukan dengan mengubah jumlah pergeseran bit.

Pada motor stepper dengan SA=1,8^0 , agar motor stepper bergerak 45^0 maka diperlukan pergeseran step sebanyak : 45/1,8 =25 step



Pada program, jika instruksi :

MOV R0,#25 diganti menjadi
MOV R0,#50

Maka posisi putaran motor stepper akan berubah menjadi 90^0

```
$mod51
        ORG     0000H
        LJMP    MULAI
        ORG     0100H
MULAI:  MOV     A,#00010001B
        MOV     R0,#25
LOOP:   MOV     P0,A
        CALL    DELAY
        RL      A
        DJNZ    R0,LOOP
HERE:   SJMP    HERE
        END
```

Kendali Torsi

Torsi motor stepper ditentukan oleh banyaknya jumlah kumparan yang aktif pada saat yang sama.
Torsi akan bertambah besar jika 2 kumparan aktif pada saat yang sama.
Secara program, ini bisa dilakukan dengan mengubah kondisi bit.

```
$mod51
        ORG     0000H
        LJMP    MULAI
        ORG     0100H
MULAI:  MOV     A,#00110011B
LOOP:   MOV     P0,A
        CALL    DELAY
        RL      A
        SJMP    LOOP
```

Pada program, jika instruksi :

MOV A,#00110011 diganti menjadi
MOV A,#00010001

Maka torsi motor stepper akan berkurang setengahnya

Latihan

Rancanglah pergerakan sebuah motor stepper dengan ketentuan sebagai berikut :
Motor akan berputar searah jarum jam sampai pada posisi 180^0 setelah itu motor akan berputar berlawanan arah jarum jam sampai pada posisi 0^0. Kemudian motor bergerak kembali ke posisi 180^0 .Demikian seterusnya.

Komunikasi Data Serial

Mode Komunikasi Data Serial
Pengiriman Data Serial
Penerimaan Data Serial

Komunikasi Serial

Komunikasi serial memiliki keuntungan dari segi efektifitasnya karena hanya membutuhkan 2 jalur komunikasi, jalur data dan clock. Data dikirim/diterima per bit secara bergantian. Pada MCS-51, data ditampung sementara dalam register SBUF ***(Serial Buffer)*** sebelum dikirim/diterima.
Untuk mengatur mode komunikasi data serial dilakukan oleh register SCON ***(Serial Control register)***.
Untuk mengatur baudrate dilakukan oleh register PCON ***(Power Control register)***.
Pada AT89S51, port serial terdapat pada P3.0(RXD) dan P3.1 (TXD)

Ada 4 mode komunikasi data serial yang bisa dilakukan mikrokontroler AT89S51 yang dapat dipilih pada bit SM0 an SM1 dalam SCON.

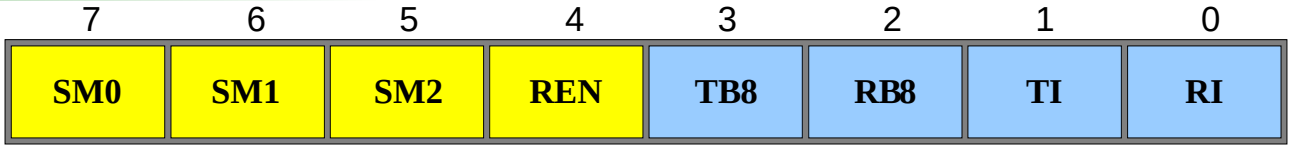
Dalam SCON terdapat flag TI *(Transmit Interrupt)* dan RI *(Receive Interrupt)* yang menandakan sedang terjadi pengiriman atau penerimaan data.

Pengiriman Data

Pengiriman data serial dimulai ketika sebuah byte data dikirimkan ke SBUF. TI akan 1 ketika data telah selesai dikirimkan.

Penerimaan Data

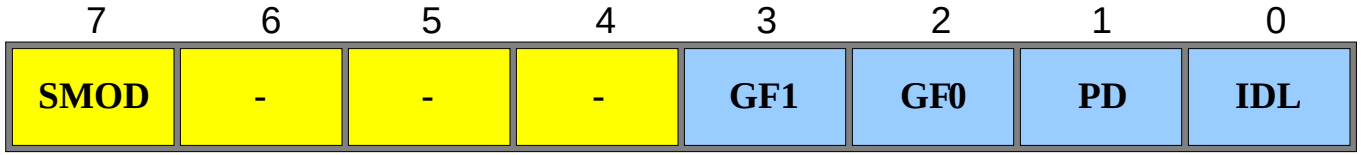
Penerimaan data serial dimulai ketika REN dalam SCON di set 1. RE akan 1 ketika data telah selesai diterima. Data tersebut kemudian disimpan di dalan SBUF



SCON / Serial Port Control Special Function Register

Bit	Symbol	Fuction
7	SM0	Serial port mode bit 0.
6	SM1	Serial port mode bit 1. SM1 SM0 Mode Description 0 0 0 Shift register, baud = f/12 0 1 1 8-bit UART, baud = variable 1 0 2 9-bit UART, baud = f/32 or f/64 1 1 3 9-bit UART, baud = variable
5	SM2	Multiprocessor communications bit Set/clear oleh program untuk mengaktifkan komunikasi multiprosesor pada mode 2 dan 3. Jika di-set 1, sebuah interrupt akan dihasilkan jika bit ke 9 dari data yang diterima adalah 1. Tidak ada interrupt yang dihasilkan jika bit ke 9 adalah 0. Jika di-set 1 pada mode 1, tidak ada interrupt yang dihasilkan kecuali jika sebuah bit stop telah diterima Clear ke 0 jika mode 0 digunakan.
4	REN	Receive enable bit Set 1 untuk mengaktifkan penerimaan, clear ke 0 untuk melumpuhkan penerimaan
3	TB8	Transmitted bit 8. Set/clear oleh program pada mode 2 dan 3.
2	RB8	Received bit 8. Bit ke-8 dari data yang diterima pada mode 2 dan 3 ; stop bit pada mode 1. Tidak digunakan pada mode 0.
1	TI	Transmit interrupt flag Set 1 pada akhir bit ke-7 pada mode 0, dan pada awal bit stop pada mode lain. Harus di-clear oleh program
0	RI	Receive interript flag Set 1 pada akhir bit ke-7 pada mode 0,dan “setengah jalan” pada bit stop pada mode lain. Harus di-clear oleh program

Bit addressable as SCON0 to SCON.7



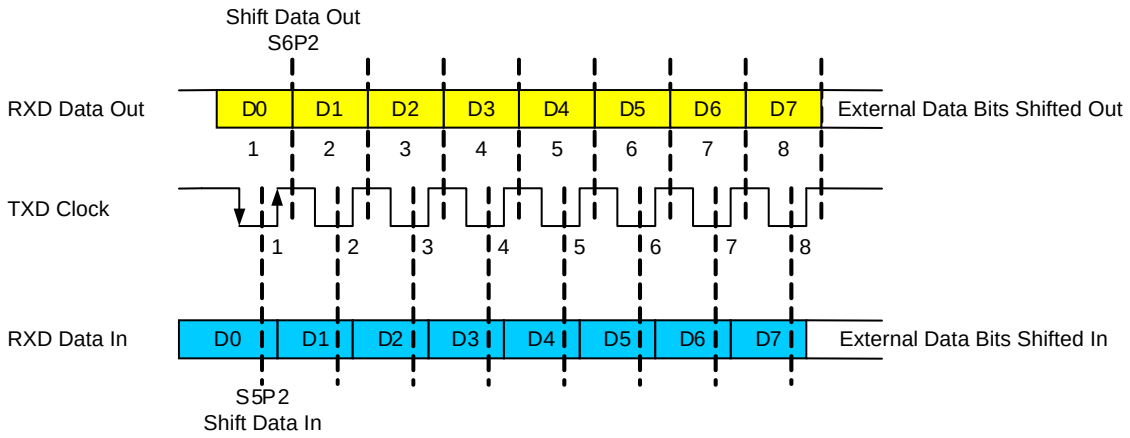
PCON / Power Mode Control Special Function Register

Bit	Symbol	Fuction
7	SMOD	Serial baud rate modify bit . Set 1 oleh program untuk menggandakan baud rate menggunakan timer 1 pada mode 1, 2 dan 3. Clear oleh program untuk menggunakan baud rate timer 1.
6-4	-	Tidak digunakan
3	GF1	General pupose user flag bit 1.
2	GF0	General pupose user flag bit 0.
1	PD	Power down bit . Set 1 oleh program untuk masuk konfigurasi power down .
0	IDL	Idle mode bit . Set 1 oleh program untuk masuk konfigurasi idle mode .

Mode Komunikasi Data Serial

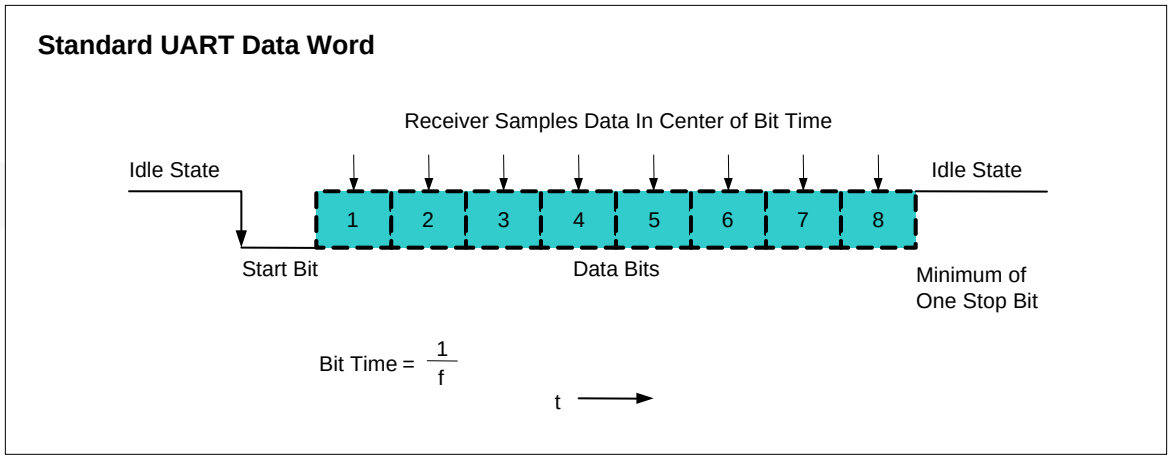
Serial Data Mode 0. Shift Register

Shift Register Mode 0 Timing



Jika bit SM1 & SM0 dalam SCON adalah 00, menyebabkan SBUF dapat menerima atau mengirim data 8 bit melalui pin RXD. Pin TXD digunakan sebagai jalur clock. Baudrate tetap yaitu 1/12 frekuensi osilator.
Ketika mengirim data, data digeser keluar pin RXD setelah satu pulsa clock. Data akan berubah ketika clock dalam fase falling edge / transisi dari haigh ke low.
Ketika menerima data dari pin RXD, data harus disinkronkan dengan pulsa clock yang dihasilkan pada TXD.

Serial Data Mode 1. Standard UART



Ketika SM0 dan SM1 adalah 01, SBUF menjadi 10-bit full-duplex receiver/transmitter yang dapat menerima dan mengirim data pada waktu yang sama. Pin RXD menerima semua data dan Pin TXD mengirim semua data.

Pengiriman data diawali dengan start bit, disusul dengan 8 bit data (Least Significant Bit / LSB terlebih dahulu) dan diakhiri dengan stop bit. Interrupt flag TI akan 1 setiap kali 10 bit dikirim.

Pengiriman data dimulai ketika start bit diterima, disusul dengan 8 bit data dan berakhir dengan diterimanya stop bit. Data 8 bit disimpan dalam SBUF dan stop bit disimpan pada RB8 dalam SCON. Interrupt flag RI akan 1 setiap kali 10 bit diterima.

Mode 1 Baud Rate

Baudrate ditentukan dengan timer 1 yang dioperasikan dalam mode 8 bit autoreload dengan persamaan sbb :

$$f_{baud} = \frac{2^{SMOD}}{32d} x \frac{\text{oscillator frequency}}{12d \text{ x } [256d - (TH1)]}$$

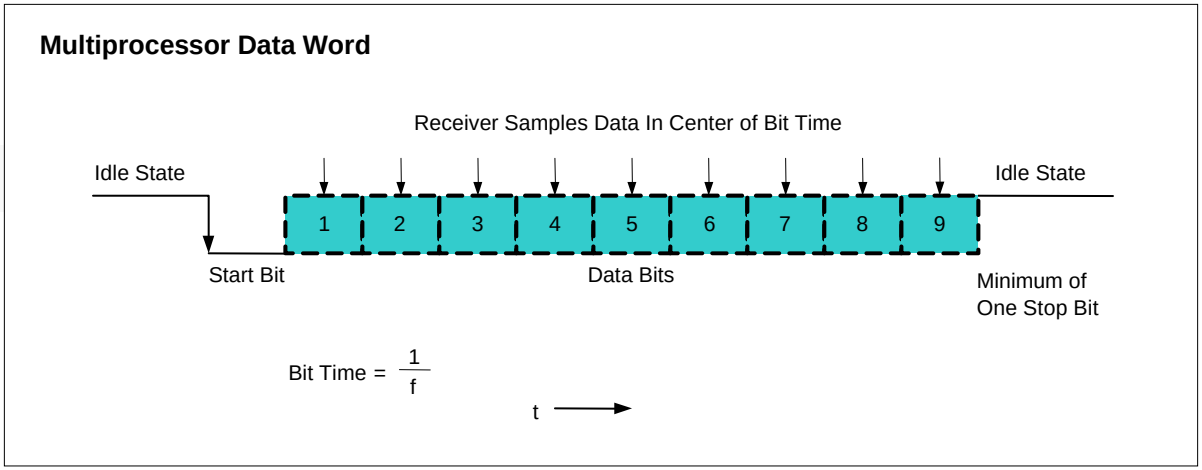
SMOD adalah control bit dalam PCON dan dapat bernilai 0 atau 1.
Jika timer 1 tidak bekerja pada mode 2, maka baudrate adalah :

$$f_{baud} = \frac{2^{SMOD}}{32d} x(\text{timer 1 overflow frequency})$$

Jika diinginkan baudrate-nya standar, maka harus menggunakan crystal dengan frekuensi 11.0592 MHz.
Untuk mendapatkan baudrate standar 9600 hertz maka nilai TH1 dapat dihitung dengan cara :

$$TH1 = 256d - \left(\frac{2^0}{32d} x \frac{11.0592 \times 10^6}{12 \times 9600d} \right) = 253d = 0FDh$$

Serial Data Mode 2. Multiprocessor Mode



Mode 2 sama dengan mode 1, tetapi jumlah data yang dikirim adalah 11 bit, dimulai dengan start bit, 9 bit data dan diakhiri dengan 1 bit stop. Data ke-9 disimpan pada TB8 dalam SCON ketika proses pengirman dan disimpan dalam RB8 ketika proses penerimaan.
Baudrate-nya adalah :

$$f_{baud} = \frac{2^{SMOD}}{64d} \times \text{oscillator frequency}$$

Serial Data Mode 3.

Mode 3 sama dengan mode 2 kecuali baudrate-nya sama dengan pada mode 1.

Contoh 1. Pengiriman Data

```
                $MOD51
                ORG         0000H
                LJMP        MULAI
                ORG         0100H
MULAI:          CALL        INITSERIAL
LOOP:           MOV         R0,#15
                MOV         DPTR,#TEXT
LOOP1:          CLR         A
                MOVC        A,@A+DPTR
                CALL        KIRIM
                CALL        DELAY
                DJNZ        R0,NEXT
                SJMP        LOOP
NEXT:           INC         DPTR
                SJMP        LOOP1
INITSERIAL:
                MOV         SCON,#52H
                MOV         TMOD,#21H
                MOV         TH1,#0FDH
                SETB        TR1
                RET
KIRIM:          JNB         TI,$
                CLR         TI
                MOV         SBUF,A
                RET
DELAY:          MOV         R5,#0FFH
                MOV         R6,#0FFH
                MOV         R7,#3
DLY:            DJNZ        R5,DLY
                DJNZ        R6,DLY
                DJNZ        R7,DLY
                RET
;-------- MESSAGES --------
;           012345678901234
TEXT:           DB          ' HELLO WORLD ! '
                END
```

Contoh 2. Penerimaan Data

```
                $MOD51
                ORG         0000H
                LJMP        MULAI
                ORG         0100H
MULAI:          CALL        INITSERIAL
LOOP:           CALL        TERIMA
                MOV         P1,A
                CALL        DELAY
                SJMP        LOOP
INITSERIAL:
                MOV         SCON,#52H
                MOV         TMOD,#21H
                MOV         TH1,#0FDH
                SETB        TR1
                RET
TERIMA:         JNB         RI,$
                CLR         RI
                MOV         A,SBUF
                RET
DELAY:          MOV         R5,#0FFH
                MOV         R6,#0FFH
                MOV         R7,#3
DLY:            DJNZ        R5,DLY
                DJNZ        R6,DLY
                DJNZ        R7,DLY
                RET
                END
```

Latihan

Buat flow chart dan program komunikasi serial antara mikrokontroler dengan PC dengan ketentuan sebagai berikut :

Ketika tombol angka di keyboard ditekan, maka angka tersebut akan ditampilkan ke seven segmen kemudian dikirim kembali ke PC untuk ditampilkan ke Hyper Terminal.

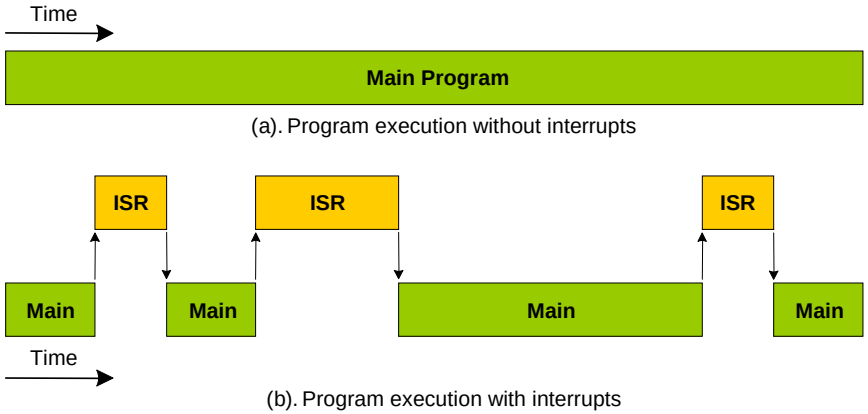
Operasi Interupsi

Register IE
Register IP
Timer Flag Interrupt
Serial Port Interrupt
External Interrupt
Reset
Lokasi Memori Interrupt

Interupsi

Interupsi adalah kondisi yang memaksa mikrokontroler menghentikan sementara eksekusi program utama untuk mengeksekusi rutin interrupt tertentu / ***Interrupt Service Routine (ISR)***

Setelah melaksanakan ISR secara lengkap, maka mikrokontroler akan kembali melanjutkan eksekusi program utama yang tadi ditinggalkan.

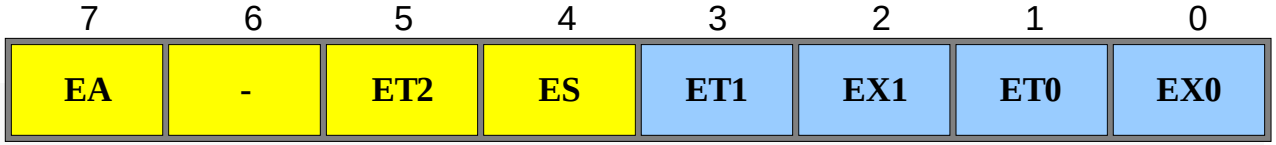


Pada AT89S51, ada 5 sumber interrupt yaitu

1. System reset
2. External 0
3. Timer 0
4. External 1
5. Timer 1
6. Serial Port.

Untuk mengatur kerja interrupt, dapat dilakukan pengaturan pada register ***Interrupt Enable (IE)*** dan ***Interrupt Priority (IP).***

Register IE

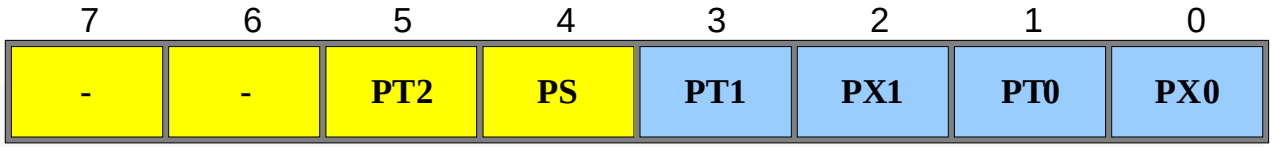


IE / Interrupt Enable Special Function Register

Bit	Symbol	Fuction
7	EA	Enable interrupts bit. Clear ke 0 oleh program untuk melumpuhkan semua interrupt, set 1 untuk mengaktifkan interrupt sesuai enable bit interrupt terkait
6	-	Tidak digunakan.
5	ET2	Reserved for future use
4	ES	Enable serial port interrupt. Set 1 oleh program untuk mengaktifkan serial port interrupt clear untuk melumpuhkan.
3	ET1	Enable timer 1 overflow interrupt. Set 1 oleh program untuk mengaktifkan timer 1 overflow interrupt, clear untuk melumpuhkan.
2	EX1	Enable external interrupt 1. Set 1 oleh program untuk mengaktifkan interrupt 1 (INT1), clear untuk melumpuhkan.
1	ET0	Enable timer 0 overflow interrupt. Set 1 oleh program untuk mengaktifkan timer 0 overflow interrupt, clear untuk melumpuhkan.
0	EX0	Enable external interrupt 0. Set 1 oleh program untuk mengaktifkan interrupt 0 (INT0), clear untuk melumpuhkan.

Bit addressable as IE0 to IE7

Register IP



IP / Interrupt Priority Special Function Register

Bit	Symbol	Fuction
7	-	Tidak digunakan
6	-	Tidak digunakan
5	PT2	Reserved for future use
4	PS	Priority of serial port interrupt
3	PT1	Priority of timer 1 overflow interrupt
2	PX1	Priority of external interrupt1.
1	PT0	Priority of timer 0 overflow interrupt
0	PX0	Priority of external interrupt0.

Bit addressable as IP.0 to IP.7

Serial Port Interrupt

Serial port interrupt terjadi jika **transmit interrupt flag (TI)** atau **receive interrupt flag (RI)** dalam kondisi set (1).

Transmit interrupt terjadi ketika mikrokontroler telah berhasil mengirim data secara lengkap.
Receive interrupt terjadi ketika mikrokontroler telah berhasil menerima data secara lengkap.

Flag TI dan RI harus di-clear (0) oleh program sebab kedua flag ini tidak otomatis clear secara hardware ketika ISR selesai dijalankan.

```
            ORG     0000h
            LJMP    MAIN
            ORG     0023h               ; serial port interrupt vector
            LJMP    SP_ISR              ; jump ke Serial Port ISR
            ORG     0030h
MAIN:       MOV     TMOD,#20h           ; Timer 1, mode 2
            MOV     TH1,#0FDh           ; 1200 baudrate
            SETB    TR1                 ; start Timer
            MOV     SCON,#42h           ; mode 1
            MOV     A,#20h              ; send ASCII space first
            MOV     IE,#90h             ; enable serial port interrupt
            SJMP    $                   ; do nothing
;
SP_ISR:     CJNE    A,#7F,SKIP          ; if finished ASCII set,
            MOV     A,#20h              ; reset to SPACE
SKIP:       MOV     SBUF,A              ; send char. to serial port
            INC     A                   ; increment ASCII code
            CLR     TI                  ; clear interrupt flag
            RETI
            END
```

External Interrupt

External interrupt terjadi jika ***interrupt external flag (IE0 atau IE1)*** pada TCON dalam kondisi set (1).
Interrupt external flag dalam kondisi set jika :

1. Terdapat sinyal low pada Pin INT0 atau INT1atau ;
2. Terdapat perubahan sinyal dari high ke low Pin INT0 atau INT1

Pengaturan kondisi sinyal interrupt ini dilakukan pada bit IT0 dan IT1 dalam register TCON. Jika ITx = 0 maka sinyal interrupt low, jika ITx=1 maka sinyal interrupt dari high ke low. Flag IEx akan kembali 0 jika subrutin interrupt dieksekusi. Untuk kembali dari subrutin interrupt digunakan instruksi RETI ***(RETurn from Interrupt)***

Reset

Reset termasuk interrupt yang tidak dapat dilumpuhkan, sebab reset dipicu secara hardware yaitu pada pin RST dan menyebabkan mikrokontroler mengeksekusi instruksi pada address 0000h. Ketika reset, nilai internal register menjadi sbb :

REGISTER	VALUE(HEX)	REGISTER	VALUE(HEX)
PC	0000	TCON	00
DPTR	0000	TMOD	00
A	00	TH0	00
B	00	TL0	00
SP	07	TH1	00
PSW	00	TL1	00
P0-3	FF	SCON	00
IP	xxx0000b	SBUF	xx
IE	0xx0000b	PCON	0xxxxxxxb

Contoh 1. Interrupt 0

```
            $MOD51
            ORG     0000H
            LJMP    MULAI
            ORG     0003H
            LJMP    INT0_ISR
            ORG     0100H
MULAI:      MOV     IE,#81H     ; Interrupt 0 enable
            MOV     IP,#01H     ; Prioritaskan
LOOP:       MOV     P0,#00
            CALL    DELAY
            MOV     P0,#0FFH
            CALL    DELAY
            SJMP    LOOP
;---------------------------------
; INTERRUPT 0 SERVICE ROUTINE
;---------------------------------
INT0_ISR:   PUSH    05H
            PUSH    06H
            PUSH    07H
            MOV     R0,#16
            MOV     A,#01H
LOOP1:      MOV     P0,A
            CALL    DELAY
            RL      A
            DJNZ    R0,LOOP1
            POP     07H
            POP     06H
            POP     05H
            RETI
;
DELAY:      MOV     R5,#0FFH
            MOV     R6,#0FH
            MOV     R7,#05H
DLY:        DJNZ    R5,DLY
            DJNZ    R6,DLY
            DJNZ    R7,DLY
            RET
            END
```

Contoh 2. Interrupt 1

```
            $MOD51
            ORG     0000H
            LJMP    MULAI
            ORG     0013H
            LJMP    INT1_ISR
            ORG     0100H
MULAI:      MOV     IE,#84H     ; Interrupt 1 enable
            MOV     IP,#04H     ; Prioritaskan
LOOP:       MOV     P0,#00
            CALL    DELAY
            MOV     P0,#0FFH
            CALL    DELAY
            SJMP    LOOP
;---------------------------------
; INTERRUPT 0 SERVICE ROUTINE
;---------------------------------
INT1_ISR:   PUSH    05H
            PUSH    06H
            PUSH    07H
            MOV     R0,#16
            MOV     A,#80H
LOOP1:      MOV     P0,A
            CALL    DELAY
            RL      A
            DJNZ    R0,LOOP1
            POP     07H
            POP     06H
            POP     05H
            RETI
;
DELAY:      MOV     R5,#0FFH
            MOV     R6,#0FH
            MOV     R7,#05H
DLY:        DJNZ    R5,DLY
            DJNZ    R6,DLY
            DJNZ    R7,DLY
            RET
            END
```

Lokasi Memori Interrupt

Interrupt Service Routine / ISR ditempatkan pada address yang berbeda untuk tiap sumber interrupt. Berikut adalah tabel sumber interrupt, flag yang dipengaruhi dan vector address tiap sumber interrupt.

INTERRUPT	FLAG	VECTOR ADDRESS
System reset	RST	0000h
External 0	IE0	0003h
Timer 0	TF0	000Bh
External 1	IE1	0013h
Timer 1	TF1	001B
Serial Port	RI or TI	0023h

Pustaka

[1]. Kenneth J. Ayala, *The 8051 Microcontroller*, Prentice Hall, 1991.
[2]. Moh.Ibnu Malik & Anistardi, Bereksperimen dengan Mikrokontroler 8031, Elex Media Komputindo, 1987.
[3]. Hendawan Soebhakti, *Buku Panduan Praktikum Sistem Mikrokontroler*, Politeknik Batam, 2006.