PETUNJUK PRAKTIKUM

MIKROKONTROLER

(AT89Sxx)

Disusun oleh : Sumarna

***E-mail* : sumarna@uny.ac.id**

JURUSAN PENDIDIKAN FISIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS NEGERI YOGYAKARTA

YOGYAKARTA 2005

PETUNJUK PRAKTIKUM MIKROKONTROLER

Disusun oleh : Sumarna, Jurdik Fisika, FMIPA, UNY
(*E-mail* : sumarna@uny.ac.id)

Pengantar :

Munculnya mikrokontroler berawal dari kebutuhan akan suatu alat khusus yang dapat dijalankan secara otomatis, praktis, dan memiliki kemampuan untuk melaksanakan perintah-perintah yang diinginkan. Dalam hal penggunaannya, mikrokontroler lebih banyak diaplikasikan secara deterministik, yaitu dipakai untuk keperluan khusus (pada umumnya sebagai pengendali). Perkembangan teknologi semikonduktor memungkinkan untuk memenuhi kebutuhan tersebut.

Bagian fungsional utama suatu mikrokontroler adalah **CPU / Mikroprosesor** (yang berisi ALU, unit kendali, register, dan pengkode), **Memori** dan **Sistem I/O**. Ketiga bagian tersebut secara fungsional (dengan fungsi masing-masing) membentuk satu sistem mikrokontroler dan berada di dalam satu chip. Jadi, sebuah mikroprosesor yang digabungkan dengan I/O dan memori (RAM,ROM) dalam satu chip itulah yang dikenal sebagai mikrokontroler.

Mengapa memilih AT89S51 ?

Untuk aplikasi dan keperluan belajar, di pasaran banyak tersedia pilihan chip mikrokontroler. Salah satu pilihan tersebut adalah AT89S51 buatan ATMEL. Beberapa pertimbangan memilih mikrokontroler tersebut antara lain :

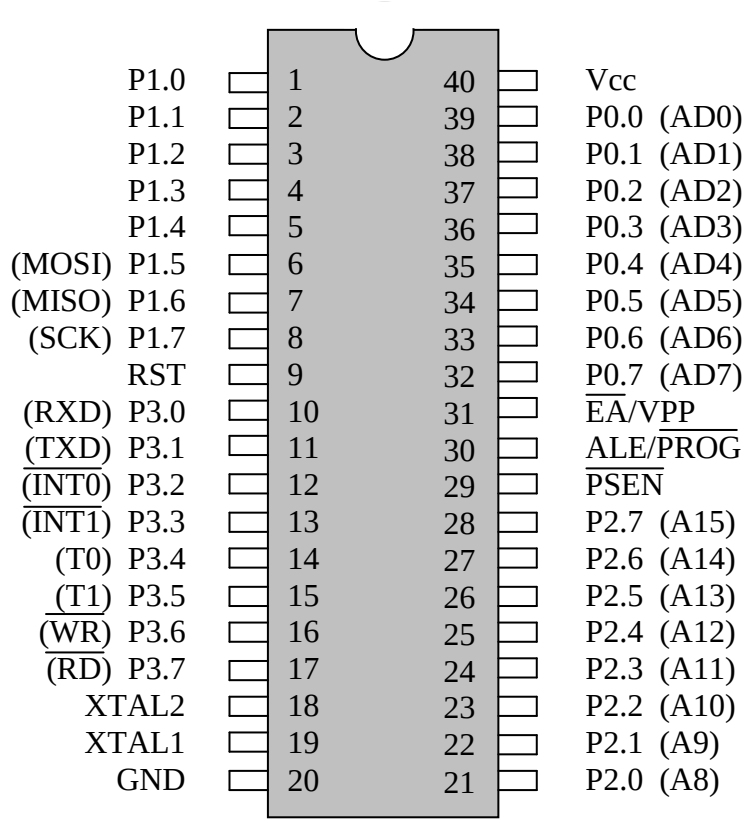
a. Murah.
b. Sebagai model.
c. Populer di kalangan masyarakat khususnya mahasiswa.
d. Cocok untuk menangani data dengan durasi detik.
e. Kompatibel dengan mikrokontroler buatan INTEL MCS-51 dan buatan ATMEL sendiri seperti AT89C51/52/53, AT89S8252.
f. Memiliki Flash memory dengan cara dan alat perograman yang sederhana
g. Aplikatif.

Sekilas mengenai MIKROKONTROLER AT89S51 :

Fasilitas yang terdapat di dalam chip mikrokontroler AT89S51 yang pokok di antaranya adalah :

a. Empat (4) port paralel I/O yang masing-masing berukuran 8 bit (P0, P1, P2, dan P3).
b. Dua (2) sistem Timer/Counter (T0 dan T1)
c. Dua (2) sistem Interupsi (INT0 dan INT1).
d. Sepasang kendali komunikas (RXD dan TXD).
e. RAM 8 x 128 byte dan Flash memory 4 Kbyte.

Letak dan macam pin-pin fungsional chip mikrokontroler AT89S51 yang bertipe DIP-40 tampak pada gambar berikut :



Memori dan Register :

Selain pengetahuan persambungan dalam hardware, pemahaman akan peta memori dan register di dalam chip AT89S51 memiliki peran yang sangat penting dalam proses penyusunan program. Register terletak di dalam CPU (mikroprosesor) dan pada umumnya berguna untuk menampung data sementara. Selain untuk menampung data sementara, ada register yang berfungsi sebagai tempat terjadinya operasi aritmatik dan logik, yaitu register **accumulator** (register A). Memori (berupa ROM atau RAM) berguna sebagai tempat untuk menampung data dan instruksi yang terletak di luar CPU (mikropsosesor).

AT89S51 memiliki struktur memori yang terdiri atas :

1. **RAM Internal**, biasanya digunakan untuk menyimpan variabel atau data yang bersifat sementara. Di dalam RAM internal terdapat 8 (delapan) Bank Register dengan mnemonik R0, R1, R2, R3, R4, R5, R6, dan R7. Delapan buah register pertama terletak pada alamat 00 h hingga 07 h dan membentuk Bank 0 (sebagai default). Posisi R0 s/d R7 dapat dipindahkan ke bank yang lain dengan mengatur bit RS0 dan RS1. Bank 1 beralamatkan

08 h s/d 0F h. Bank 2 terletak pada alamat 10 h s/d 17 h, dan bank 3 menempati alamat 18 h s/d 1F h. R0 dan R1 adalah dua buah register yang dapat digunakan sebagai pointer dari sebuah lokasi memori pada RAM internal tersebut. Di dalam RAM internal pada alamat 20 h hingga 2F h dapat diakses dengan cara pengalamatan bit sehingga hanya dengan sebuah instruksi setiap bit dalam daerah ini dapat di-set, di-clear, di-AND dan di-OR. Di dalam RAM internal juga terdapat RAM untuk keperluan umum yang dimulai dari alamat 30 h hingga 7F h.

2. **SFR** (Special Function Register), berisi register-register yang memiliki fungsi khusus yang disediakan oleh chip AT89S51. SFR terletak pada alamat antara 80 h hingga FF h. Berikut ini disampaikan daftar register di dalam chip AT89S51 dengan fungsi khusus (SFR : Special Function Register).

No.	Register	Mnemonic	Alamat
1.	Akumulator	A atau ACC	E0 h
2.	B	B	F0 h
3.	Port 0	P0	80 h
4.	Port 1	P1	90 h
5.	Port 2	P2	A0 h
6.	Port 3	P3	B0 h
7.	Interupt Enable	IE	A8 h
8.	Stack Pointer	SP	81 h
9.	Data Pointer (total)	DPTR	82 h – 83 h
10.	Data Pointer Low Byte	DPL	82 h
11.	Data Pointer High Byte	DPH	83 h
12.	Power Control	PCON	87 h
13.	Timer/Counter Control	TCON	88 h
14.	Timer/Counter Control Mode	TMOD	89 h
15.	Timer/Counter 0 Low Byte	TL0	8A h
16.	Timer/Counter 1 Low Byte	TL1	8B h
17.	Timer/Counter 0 High Byte	TH0	8C h
18.	Timer/Counter 1 High Byte	TH1	8D h
19.	Serial Port Control	SCON	98 h
20.	Serial Data Port	SBUF	99 h
21.	Interupt Control Priority	IP	B8 h
22.	Program Status Word	PSW	D0 h

3. **Flash PEROM** (Programmable and Erasable ROM), digunakan untuk menyimpan program aplikasi yang disusun oleh pemrogram. Flash PEROM tersebut dapat ditulis atau dihapus berulang-ulang (1000 kali) menggunakan perangkat pemrogram (downloader), misalnya dengan AEC_ISP. Program yang ada dalam Flash PEROM dapat dijalankan jika pada saat sitem di-reset, maka pena EA/VPP berlogika 1 (satu).

Peta Lokasi di dalam RAM AT89**S**51 :

Alamat Byte	Alamat Bit							
7F 30	RAM Keperluan Umum							
2F	7F	7E	7D	7C	7B	7A	79	78
2E	77	76	75	74	73	72	71	70
2D	6F	6E	6D	6C	6B	6A	69	68
2C	67	66	65	64	63	62	61	60
2B	5F	5E	5D	5C	5B	5A	59	58
2A	57	56	55	54	53	52	51	50
29	4F	4E	4D	4C	4B	4A	49	48
28	47	46	45	44	43	42	41	40
27	3F	3E	3D	3C	3B	3A	39	38
26	37	36	35	34	33	32	31	30
25	2F	2E	2D	2C	2B	2A	29	28
24	27	26	25	24	23	22	21	20
23	1F	1E	1D	1C	1B	1A	19	18
22	17	16	15	14	13	12	11	10
21	0F	0E	0D	0C	0B	0A	09	08
20	07	06	05	04	03	02	01	00
1F 18	Bank-3							
17 10	Bank-2							
0F 08	Bank-1							
07 00	Bank-0 Default bank register Untuk R0 – R7							

Lokasi yang dapat dialamati secara bit

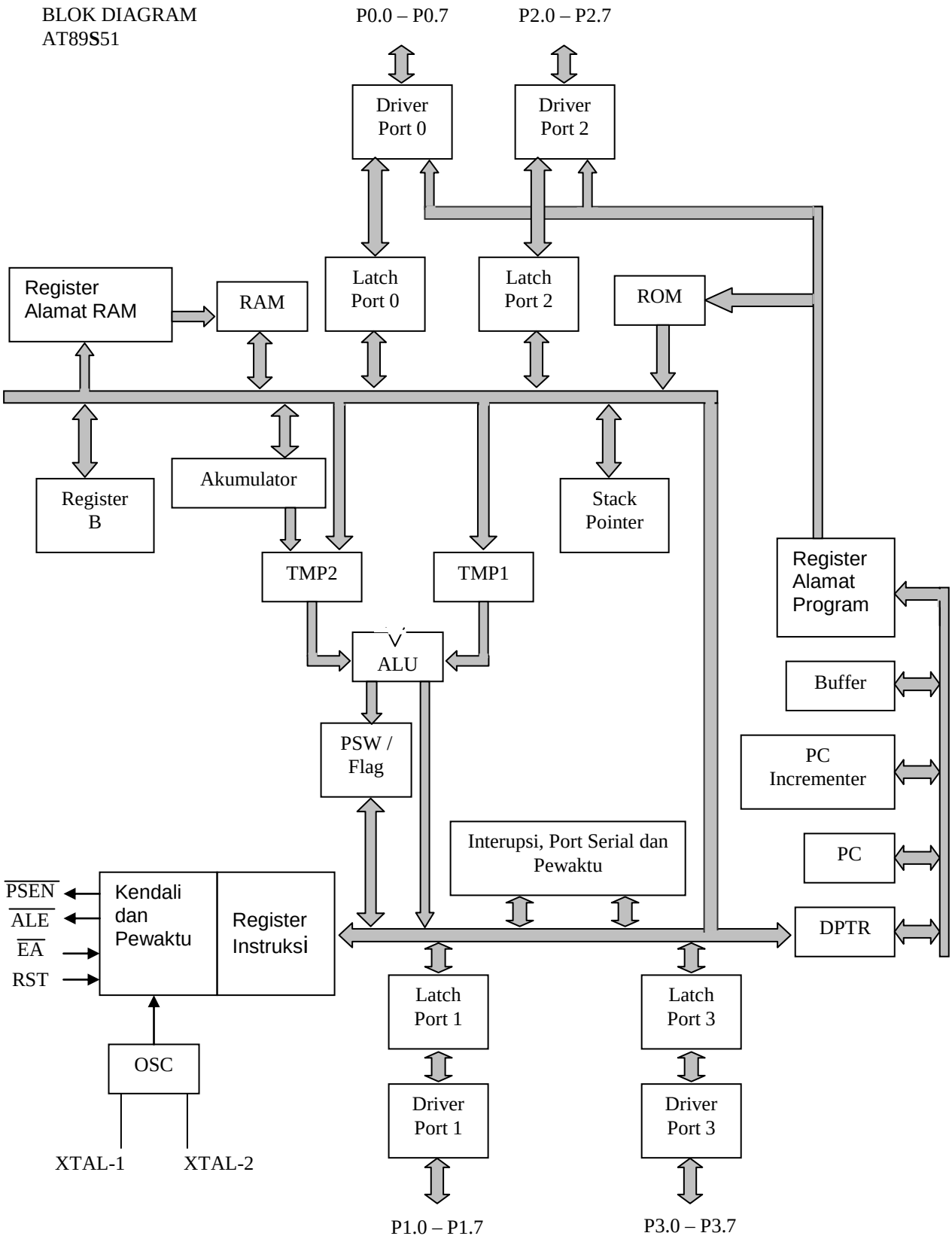
RAM

Peta Lokasi di dalam SFR AT89**S**51 :

Alamat Byte	Alamat Bit								Nama Register
FF									
F0	F7	F6	F5	F4	F3	F2	F1	F0	B
E0	E7	E6	E5	E4	E3	E2	E1	E0	A
D0	D7	D6	D5	D4	D3	D2	-	D0	PSW
B8	-	-	-	BC	BB	BA	B9	B8	IP
B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
A8	AF	-	-	AC	AB	AA	A9	A8	IE
A0	A7	A6	A5	A4	A3	A2	A1	A0	P2
99	Tidak dapat dialamati secara bit								SBUF
98	9F	9E	9D	9C	9B	9A	99	98	SCON
90	97	96	95	94	93	92	91	90	P1
8D	Tidak dapat dialamati secara bit								TH1
8C	Tidak dapat dialamati secara bit								TH0
8B	Tidak dapat dialamati secara bit								TL1
8A	Tidak dapat dialamati secara bit								TL0
89	Tidak dapat dialamati secara bit								TMOD
88	8F	8E	8D	8C	8B	8A	89	88	TCON
87	Tidak dapat dialamai secara bit								PCON
83	Tidak dapat dialamati secara bit								DPH
82	Tidak dapat dialamati secara bit								DPL
81	Tidak dapat dialamati secara bit								SP
80	87	86	85	84	83	82	81	80	P0

Register Fungsi Khusus (SFR)

BLOK DIAGRAM
AT89**S**51



DESKRIPSI SINGKAT INSTRUKSI-INSTRUKSI PADA AT89S51

No.	Instruksi	Deskripsi	Contoh
1.	ADD A,R_n	Menambahkan isi A dan isi R_n dan hasilnya disimpan di A (n : 0 s/d 7)	**ADD A,R7**
2.	ADD A,nH	Menambahkan isi A dengan isi lokasi memori yang alamatnya n dan hasilnya disimpan di A	**ADD A,30H**
3.	ADD A,@R_n	Menambahkan isi A dengan data yang ada di alamat yang ditunjukkan oleh isi R_n dan hasilnya disimpan di A	**ADD A,@R0**
4.	ADD A,#n	Menambahkan isi A dengan sebuah konstanta n dan hasilnya disimpan di A	**ADD A,#05h**
5.	ADDC A,R_n	Menambahkan isi A beserta carry flag dan isi R_n dan hasilnya disimpan di A (n = 0, 1, …, 7)	**ADDC A,R7**
6.	ADDC A,nH	Menambahkan isi A beserta carry flag dan isi lokasi memori yang alamatnya n dan hasilnya disimpan di A	**ADDC A,30H**
7.	ADDC A,@R_n	Menambahkan isi A beserta carry flag dengan data yang ada di alamat yang ditunjukkan oleh isi R_n dan hasilnya disimpan di A	**ADDC A,@R0**
8.	ADDC A,#n	Menambahkan isi A beserta carry flag dengan sebuah konstanta n dan hasilnya disimpan di A	**ADDC A,#05h**
9.	SUBB A,R_n	Mengurangi data di A beserta carry flag-nya dengan isi R_n dan hasilnya disimpan di A	**SUBB A,R0**
10.	SUBB A,nH	Mengurangi data di A beserta carry flag-nya dengan isi lokasi yang alamatnya n, hasilnya disimpan di A	**SUBB A,30H**
11.	SUBB A,@R_n	Mengurangi isi dari A beserta carry flag-nya dengan data yang ada di alamat yang ditunjukkan oleh isi R_n dan hasilnya disimpan di A	**SUBB A,@R0**
12.	SUBB A,#n	Mengurangi data di A beserta carry flag-nya dengan sebuah konstanta n dan hasilnya disimpan di A	**SUBB A,#05h**
13.	INC A	Menambah data yang ada di A dengan 1 dan hasilnya disimpan di A	**INC A**
14.	INC R_n	Menambah data yang ada di R_n dengan 1 dan hasilnya disimpan di R_n	**INC R0**
15.	INC nH	Menambahkan 1 kepada data yang ada di lokasi yang alamatnya n dan hasilnya disimpan di alamat tersebut	**INC 30H**

16.	INC @R_n	Menambahkan 1 kepada data yang ada di lokasi yang alamatnya ditunjukkan oleh isi R_n dan hasilnya disimpan di alamat tersebut	**INC @R5**
17.	INC DPTR	Menambahkan 1 kepada data yang ada di DPTR dan hasilnya disimpan di DPTR	**INC DPTR**
18.	DEC A	Mengurangi data yang ada di A dengan 1 dan hasilnya disimpan di A	**DEC A**
19.	DEC R_n	Mengurangi data yang ada di R_n dengan 1 dan hasilnya disimpan di R_n	**DEC R2**
20.	DEC nH	Mengurangi dengan 1 kepada data yang ada di lokasi yang alamatnya n dan hasilnya disimpan di alamat tersebut	**DEC 30H**
21.	DEC @R_n	Mengurangi dengan 1 kepada data yang ada di lokasi yang alamatnya ditunjukkan oleh isi R_n dan hasilnya disimpan di alamat tersebut	**DEC @R7**
22.	MUL AB	Mengalikan isi A dan isi register B, hasil perkalian disimpan di A untuk byte rendah dan untuk byte tinggi disimpan di B. Jika hasil perkalian melebihi 0FF H, maka flag overflow-nya akan di-set.	**MUL AB**
23.	DIV AB	Membagi data di A dengan data di B yang hasil baginya disimpan di A dan sisa pembagiannya disimpan di B. Flag overflow dan carry akan selalu clear. Jika isi B adalah 0, maka flag overflow di-set.	**DIV AB**
24.	DA A	Mengubah data di A menjadi bentuk BCD. Instruksi ini biasa digunakan sesudah instruksi ADD.	**ADD A,#05h** **DA A**
25.	ANL A,R_n	Melakukan operasi logik AND antara isi A dan isi R_n dan hasilnya disimpan di A	**ANL A,R3**
26.	ANL A,nH	Melakukan operasi AND antara isi A dan isi lokasi yang alamatnya n dan hasilnya disimpan di A	**ANL A,30H**
27.	ANL A,@R_n	Melakukan operasi AND antara isi A dan isi lokasi yang alamatnya ditujukkan oleh isi dari R0 atau R1 dan hasilnya disimpan di A	**ANL A,@R0**
28.	ANL A,#n	Melakukan operasi AND antara isi A dengan suatu konstanta n dan hasilnya disimpan di A	**ANL A,#08h**

29.	ANL nH,A	Melakukan operasi AND antara isi dari lokasi yang alamatnya n dengan isi A dan hasilnya disimpan di lokasi (alamat n) tersebut	**ANL 07H,A**
30.	ANL nH,#n	Melakukan operasi AND antara isi dari lokasi yang alamatnya n dengan suatu konstanta n dan hasilnya disimpan di lokasi (alamat n) tersebut (alamat n dan konstanta n tidak harus sama)	**ANL 07H,#05h**
31.	ORL A,R_n	Melakukan operasi logik OR antara isi A dan isi R_n dan hasilnya disimpan di A	**ORL A,R_n**
32.	ORL A,nH	Melakukan operasi OR antara isi A dan isi lokasi yang alamatnya n dan hasilnya disimpan di A	**ORL A,30H**
33.	ORL A,@R_n	Melakukan operasi OR antara isi A dan isi lokasi yang alamatnya ditujukkan oleh isi dari R0 atau R1 dan hasilnya disimpan di A	**ORL A,@R0**
34.	ORL A,#n	Melakukan operasi OR antara isi A dengan suatu konstanta n dan hasilnya disimpan di A	**ORL A,#05h**
35.	ORL nH,A	Melakukan operasi OR antara isi dari lokasi yang alamatnya n dengan isi A dan hasilnya disimpan di lokasi (alamat n) tersebut	**ORL 30H,A**
36.	ORL nH,#n	Melakukan operasi OR antara isi dari lokasi yang alamatnya n dengan suatu konstanta n dan hasilnya disimpan di lokasi (alamat n) tersebut (alamat n dan konstanta n tidak harus sama)	**ORL 30H,#05h**
37.	XRL A,R_n	Melakukan operasi logik EX-OR antara isi A dan isi R_n dan hasilnya disimpan di A	**XRL A,R2**
38.	XRL A,nH	Melakukan operasi EX-OR antara isi A dan isi lokasi yang alamatnya n dan hasilnya disimpan di A	**XRL A,30H**
39.	XRL A,@R_n	Melakukan operasi EX-OR antara isi A dan isi lokasi yang alamatnya ditujukkan oleh isi dari R0 atau R1 dan hasilnya disimpan di A	**XRL A,@R1**
40.	XRL A,#n	Melakukan operasi EX-OR antara isi A dengan suatu konstanta n dan hasilnya disimpan di A	**XRL A,#05h**
41.	XRL nH,A	Melakukan operasi EX-OR antara isi	**XRL 30H,A**

		dari lokasi yang alamatnya n dengan isi A dan hasilnya disimpan di lokasi (alamat n) tersebut	
42.	XRL mH,#n	Melakukan operasi EX-OR antara isi dari lokasi yang alamatnya m dengan konstanta n dan hasilnya disimpan di lokasi (m) tersebut	**XRL 30H,#05h**
43.	CLR A	Membuat isi akumulator A menjadi 0 (nol).	**CLR A**
44.	CPL A	Mengkomplemen setiap bit isi akumulator A	**CPL A**
45.	RL A	Menggeser ke kiri 1 (satu) bit pada setiap bit isi akumulator A	**RL A**
46.	RLC A	Menggeser ke kiri 1 (satu) bit pada setiap bit isi akumulator A beserta carry flag-nya	**RLC A**
47.	RR A	Menggeser ke kanan 1 (satu) bit pada setiap bit isi akumulator A	**RR A**
48.	RRC A	Menggeser ke kanan 1 (satu) bit pada setiap bit isi akumulator A beserta carry flag-nya	**RRC A**
49.	SWAP A	Melalukan operasi penukaran nibble tinggi dan nibble rendah di dalam akumulator A	**SWAP A**
50.	MOV A,R_n	Memindahkan data dari register R_n ke akumulator A	**MOV A,R3**
51.	MOV A,nH	Memindahkan data dari lokasi yang alamatnya n ke akumulator A	**MOV A,30H**
52.	MOV A,@R_n	Memindahkan data dari lokasi yang alamatnya ditunjukkan oleh isi register R0 atau R1 ke akumulator A	**MOV A,@R0**
53.	MOV A,#n	Mengisi akumulator A dengan suatu konstanta (data) n	**MOV A,#05h**
54.	MOV R_n,A	Memindahkan data dari akumulator A ke register R_n	**MOV R5,A**
55.	MOV R_n,nH	Memindahkan data dari lokasi yang alamatnya n ke register R_n	**MOV R7,30H**
56.	MOV R_n,#n	Mengisi register R_n dengan suatu konstanta (data) n	**MOV R6,#05h**
57.	MOV nH,A	Memindahkan data dari akumulator ke suatu lokasi yang alamatnya n	**MOV 30H,A**
58.	MOV nH,R_n	Memindahkan data dari register R_n ke suatu lokasi yang alamatnya n	**MOV 30H,R6**
59.	MOV mH,nH	Memindahkan data dari suatu lokasi yang alamatnya n ke lokasi lain yang alamatnya m	**MOV 10H,30H**
60	MOV	Memindahkan data dari suatu lokasi	**MOV 30H,@R0**

	nH,@R_n	yang alamatnya ditunjukkan oleh isi register R0 atau R1 ke suatu lokasi yang alamatnya n	
61.	MOV nH,#n	Mengisikan suatu konstanta n (data) ke suatu lokasi yang alamatnya n	**MOV 30H,#05h**
62.	MOV @R_n,A	Memindahkan data dari A ke satu lokasi yang alamatnya ditunjukkan oleh isi register R0 atau R1	**MOV @R1,A**
63.	MOV @R_n,nH	Memindahkan data dari suatu lokasi yang alamatnya n ke lokasi lain yang alamatnya ditunjukkan oleh isi register R0 atau R1	**MOV @R1,30H**
64.	MOV @R_n,#n	Mengisikan suatu konstanta n (data) ke suatu lokasi yang alamatnya ditunjukkan oleh isi register R0 atau R1	**MOV @R_n,#05h**
65.	MOV DPTR,#nn	Mengisi register DPTR dengan suatu konstanta 16 bit	**MOV DPTR,#2000h**
66.	MOVC A,@A+DPTR	Memindahkan data dari memori program yang lokasinya ditunjukkan oleh isi DPTR ditambah indeks akumulator A menuju ke A	**MOVC A,@A+DPTR**
67.	MOVC A,@A+PC	Memindahkan data dari memori program yang lokasinya ditunjukkan oleh isi Program Counter ditambah indeks akumulator A menuju ke A	**MOVC A,@A+PC**
68.	MOVX A,@R_n	Memindahkan data dari memori eksternal yang alamatnya ditunjukkan oleh R_n ke akumulator A	**MOVX A,@R0**
69.	MOVX A,@DPTR	Memindahkan data dari memori eksternal yang alamatnya ditunjukkan oleh DPTR ke akumulator A	**MOVX A,@DPTR**
70.	MOVX @R_n,A	Memindahkan data dari akumulator A ke memori eksternal yang alamatnya ditunjukkan oleh R_n	**MOVX @R0,A**
71.	MOVX @DPTR,A	Memindahkan data dari akumulator A ke memori eksternal yang alamatnya ditunjukkan oleh DPTR	**MOVX @DPTR,A**
72.	PUSH n	Menyimpan data dari suatu register atau memori ke dalam stack	**PUSH A**
73.	POP n	Mengambil data dari dalam stack dan dikembalikan ke suatu register atau memori	**POP B**
74.	XCH A,R_n	Menukar data yang tersimpan di akumulator A dengan data yang ada di register R_n	**XCH A,R7**
75.	XCH A,nH	Menukar data yang tersimpan di akumulator A dengan data yang ada di	**XCH A,70H**

		suatu lokasi yang alamatnya n	
76.	XCH A,@R_n	Menukar data yang tersimpan di akumulator A dengan data yang ada di suatu lokasi yang alamatnya ditunjukkan oleh isi R0 atau R1	**XCH A,@R0**
77.	XCHD A,@R_n	Menukar data nibble rendah di akumulator A dengan data nibble rendah di suatu lokasi yang alamatnya ditunjukkan oleh isi R0 atau R1	**XCHD A,@R0**
78.	CLR C	Mengubah bit carry flag menjadi 0 (nol)	**CLR C**
79.	CLR bit	Mengubah bit pada RAM internal atau register yang dapat dialamati secara bit menjadi 0 (nol)	**CLR P1.2**
80.	SETB C	Mengubah bit carry flag menjadi 1 (satu)	**SETB C**
81.	SETB bit	Mengubah bit pada RAM internal atau register yang dapat dialamati secara bit menjadi 1 (satu)	**SETB A.7**
82.	CPL C	Melakukan komplemen pada bit carry flag	**CPL C**
83.	CPL bit	Melakukan komplemen bit memori atau register yang dapat dialamati secara bit	**CPL A.6**
84.	ANL C,bit	Melakukan operasi logik AND antara bit carry flag dan bit pada register atau memori yang dapat dialamati secara bit	**ANL C,B.5**
85.	ANL C,/bit	Melakukan operasi AND antara bit carry flag dengan komplemen dari bit pada register atau memori yang dapat dialamati secara bit	**ANL C,/A.7**
86.	ORL C,/bit	Melakukan operasi OR antara bit carry flag dengan komplemen dari bit pada register atau memori yang dapat dialamati secara bit	**ORL C,/A.7**
87.	MOV C,bit	Memindahkan bit pada register atau memori yang dapat dialamati secara bit ke bit carry flag	**MOV C,A.0**
88.	MOV bit,C	Memindahkan bit carry flag ke bit pada register atau memori yang dapat dialamati secara bit	**MOV A.1,C**
89.	JC rel	Melompat ke suatu alamat yang didefinisikan oleh rel jika carry flag-nya set	**JC Loop**
90.	JNC rel	Melompat ke suatu alamat yang didefinisikan oleh rel jika carry flag-nya clear	**JNC Loop**

91.	JB bit,rel	Melompat ke suatu alamat yang didefinisikan oleh rel jika bit dari register atau memori yang dapat dialamati secara bit dalam keadaan set	**JB P1.0,Loop**
92.	JNB bit,rel	Melompat ke suatu alamat yang didefinisikan oleh rel jika bit dari register atau memori yang dapat dialamati secara bit dalam keadaan clear	**JNB P1.0,Loop**
93.	JBC bit,rel	Melompat ke suatu alamat yang didefinisikan oleh rel jika bit dari register atau memori yang dapat dialamati secara bit dalam keadaan set, tetapi bit tersebut di-clear setelah lompatan dilakukan	**JBC A.7,Loop**
94.	CALL Addr	Memanggil subroutine yang ditunjuk oleh alamat pada Addr atau tanda label (Otomatis di konversi menjadi ACALL atau LCALL tergantung pada jauhnya lompatan)	**Call Delay**
95.	ACALL Addr	Memanggil subroutine yang ditunjuk oleh alamat pada Addr atau tanda label. Lompatan pemanggilan sejauh 2 K byte	**Acall 35H Acall Loop**
96.	LCALL Addr-16	Memanggil subroutine yang ditunjuk oleh alamat pada Addr atau tanda label. Lompatan pemanggilan sejauh 64 K byte	**LCALL Loop**
97.	RET	Kembali ke alamat yang disimpan dalam SP dan SP-1	**RET**
98.	RETI	Kembali ke alamat yang disimpan dalam SP dan SP-1 dan mengembalikan kondisi flag-flag interrupt agar interupsi berikutnya dengan prioritas yang sama dapat dilakukan	**RETI**
99.	AJMP Addr	Melompat dan menjalankan program yang berada di alamat yang ditentukan oleh Addr (11 bit dari alamat pada Addr dipindahkan ke PC)	**AJMP Loop**
100.	LJMP Addr-16	Melompat dan menjalankan program yang berada di alamat yang ditentukan oleh Addr (16 bit dari alamat pada Addr dipindahkan ke PC)	**LJMP Loop**
101.	SJMP rel	Melompat ke alamat yang ditentukan oleh rel dengan lompatan maksimum 128 byte	**SJMP Awal**

102.	JMP @A+DPTR	Melompat ke alamat yang dihasilkan oleh penjumlahan antara DPTR dan akumulator A	**JMP @A+DPTR**
103.	JZ rel	Melompat ke alamat yang ditentukan oleh rel jika akumulator A sama dengan 0 (nol)	**JZ Loop**
104.	JNZ rel	Melompat ke alamat yang ditentukan oleh rel jika akumulator A adalah tidak sama dengan 0 (nol)	**JNZ Loop**
105.	CJNE A,nH,rel	Membandingkan isi A dengan isi lokasi yang alamatnya n dan melompat ke alamat yang ditentukan oleh rel jika hasil perbandingan tidak sama	**CJNE A,30H,Loop**
106.	CJNE A,#n,rel	Membandingkan isi A dengan suatu konstanta n dan melompat ke alamat yang ditentukan oleh rel jika hasil perbandingan tidak sama	**CJNE A,#05h,Loop**
107.	CJNE R_n,#n,rel	Membandingkan isi register R_n dengan suatu konstanta n dan melompat ke alamat yang ditentukan oleh rel jika hasil perbandingan tidak sama	**CJNE R6,#05h, Loop**
108.	CJNE @R_n,#n,rel	Membandingkan data yang terletak pada suatu lokasi yang alamatnya ditunjukkan oleh isi R0 atau R1 dengan suatu konstanta n dan melompat ke alamat yang ditentukan oleh rel jika hasil perbandingan tidak sama	**CJNE @R0,#05h, Loop**
109.	DJNZ R_n,rel	Mengurangi isi R_n dengan 1 dan melompat ke alamat yang ditentukan oleh rel jika hasilnya belum 0 (nol)	**DJNZ R7,Loop**
110.	DJNZ nH,rel	Mengurangi dengan 1 kepada isi suatu lokasi yang alamatnya n dan melompat ke alamat yang ditentukan oleh rel jika hasilnya belum 0 (nol)	**DJNZ 30H,Loop**
111.	NOP	Melakukan penundaan (delay) pada program sebesar 1 cycle tanpa mempengaruhi isi register-register dan flag.	**NOP**

TATA CARA PEMROGRAMAN AT89S51 :

Yang dimaksud dengan tata cara pemrograman adalah langkah-langkah memasukkan program aplikasi ke dalam chip AT89S51 (download). Sebelum program aplikasi yang disusun oleh seorang programmer dimasukkan ke dalam chip mikrokontroller, maka tata cara berikut harus dilakukan.

1. Program aplikasi disusun dengan bahasa asembler untuk mikrokontroler yang bersangkutan, kemudian ditulis dengan editor teks (seperti EDIT pada MS-DOS, atau NOTEPAD pada Windows), dan disimpan dengan ekstensi **.asm**. Misalkan program itu adalah **Latihan.asm**.

2. Program yang telah diketik dan disimpan tadi kemudian dikompilasi dengan program kompilasi misalnya ASM51 di bawah MS-DOS.
   Caranya :
   >ASM51 **Latihan.asm** atau >ASM **Latihan.asm**

   Setelah menekan ENTER akan diperoleh program yang telah dikompilasi dengan ekstensi otomatis **.lst** (yakni **Latihan.lst**) dan **.obj** (yaitu **Latihan.obj**). Hasil file **.lst** tersebut disertai pesan **no error found** jika tidak ditemukan kesalahan syntax atau **error** jika dijumpai kesalahan dan **harus dibetulkan**.

3. Setelah diperoleh program yang telah dikompilasi (dan tidak ada kesalahan), proses selanjutnya adalah mengubah program tersebut ke format hexa dengan program OH.
   Caranya :
   >OH **Latihan.obj**

   Setelah menekan ENTER akan diperoleh program dalam format hexa yang ditandai dengan ekstensi **.hex** (yakni **Latihan.hex**).

4. Program aplikasi dengan format hex tersebut telah siap dimasukkan (download) ke dalam chip mikrokontroler AT89S51 dengan program downloader antara lain **aec_isp**.
   Caranya :
   >**aec_isp**

   Setelah menekan ENTER akan muncul kotak dialog yang harus di-set lebih dahulu agar sesuai dengan keadaan dan keperluan pemakai.

5. Sorot (pilih) menu E (Programing) dalam dialog untuk memerintahkan komputer men-download program aplikasi ke dalam chip. Jika berhasil, muncul tampilan persentasi pen-download-an (hingga 100%).

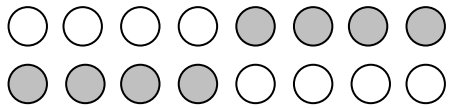
Percobaan-1

Mengendalikan Nyala LED Melalui Port Paralel

Tujuan :

Membuat aplikasi untuk mengendalikan nyala beberapa LED melalui port paralel dengan berbagai macam pola dan kombinasi.

Berikut ini adalah program untuk mengendalikan nyala LED dengan pola flip-flop masing-masing 4 LED, seperti gambar di bawah ini. Port yang dihubungkan ke 8 (delapan) LED adalah P1 (P1.0, P1.1, P1.2, … , P1.7).



Programnya adalah sebagai berikut :

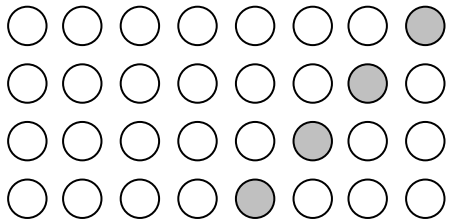
```
            Org   0h
Mulai :     Mov         P1,#00001111B
            Acall       Delay
            Mov         P1,#11110000B
            Acall       Delay
            Sjmp        Mulai

Delay :     Mov   R0,#0FFh
Delay1 :    Mov   R1,#0FFh
Delay2 :    Djnz  R1,Delay2
            Djnz  R0,Delay1
            Ret
            End
```

Ketiklah program di atas pada suatu editor (EDIT di bawah MS-DOS), simpanlah dengan nama **DemoLed1.asm** (nama boleh lain), lakukan kompilasi program hingga menghasilkan berkas dalam format hex, kemudian lakukan download menggunakan program AEC_ISP.

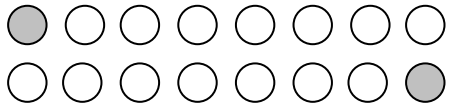
Program berikut berguna untuk mengendalikan nyala LED dengan pola bergeser ke kiri, seperti gambar di bawah. Programnya adalah :

```
            Org   0h
            Mov         A,#11111110B
Mulai :     Mov         P1,A
            Acall       Delay
            RL          A
            Sjmp        Mulai
```

```
Delay :   Mov  R0,#0
Delay1 :  Mov  R1,#0
Delay2 :  Djnz R1,Delay2
          Djnz R0,Delay1
          Ret

          End
```



Dst.



Setelah program ditulis pada editor (EDIT di bawah MS-DOS), simpanlah dengan nama **DemoLed2.asm** lakukan kompilasi program hingga menghasilkan berkas dalam format hex, kemudian lakukan download menggunakan program AEC_ISP.

Selain pola, cara memrogramnya juga dapat bervariasi sesuai dengan kreativitas pemrogramnya. Cobalah program berikut !

```
          Org  0h

Loop :    Mov  Dptr,#DataOut

Loop1 :   Clr  A
          Movc A,@A+Dptr
          Mov  P1,A
          Call Delay
          Inc  Dptr
          Cjne A,#'X',Loop1
          Sjmp Loop

Delay :   Mov  R1,#0
Delay1 :  Mov  R2,#0
          Djnz R2,$
          Djnz R1,Delay1
          Ret

DataOut :
     DB   01111110B,00111100B,00011000B,00000000B
```

```
DB   00011000B,00111100B,01111110B,11111111B,'X'
     End
```

Setelah program ditulis pada editor (EDIT di bawah MS-DOS), simpanlah dengan nama **DemoLed3.asm** lakukan kompilasi program hingga menghasilkan berkas dalam format hex, kemudian lakukan download menggunakan program AEC_ISP.

Tugas-1 :

1. Jelaskan contoh-contoh program di atas ! Apa yang dikerjakan ? Jelaskan maksud pada setiap baris/instruksi pada setiap program !
2. Buatlah diagram alir untuk setiap program di atas !
3. Pola nyala LED dapat diubah-ubah sesuai dengan kehendak pemrogramnya, seperti pola ping-pong, pola geser kanan / kiri, pola menyebar, pola merapat, dll. Susunlah program untuk dapat mengendalikan nyala LED dengan pola-pola tersebut ! Kemudian tunjukkan apakah program itu sesuai dengan yang dimaksudkan.

Menyusun program ibarat membuat karya seni

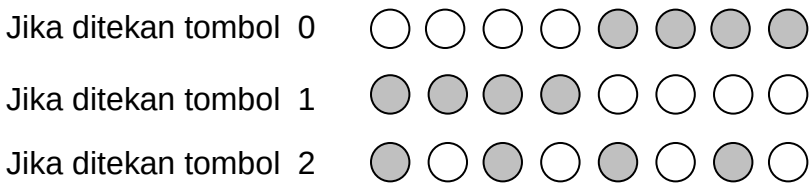
Percobaan-2

Penggunaan Tombol Tekan (Push Button)

Tujuan :

Membuat aplikasi data masukan dan fungsi interupsi pada mikrokontroler AT89S51 melaui penggunaan tombol tekan (push button) untuk memberikan (mengendalikan) status logik pada port paralel dan sebagai indikatornya adalah nyala LED pada port yang bersangkutan.

Berikut ini adalah program untuk menyalakan LED yang dikendalikan melalui penekanan tombol tekan (push button). Dalam program tersebut melibatkan penggunaan data masukan. Pola nyala LED terkait dengan suatu tombol yang ditekan, contohnya sebagai berikut.



Programnya adalah :

```
            Org       0h
Loop :      Mov       A,P3
            Cjne      A,#11111110B,Cek1
            Mov       P1,#0Fh
            Sjmp      Loop
Cek1 :      Cjne      A,#11111101B,Cek2
            Mov       P1,#0F0h
            Sjmp      Loop
Cek2 :      Cjne      A,#11111011B,Loop
            Mov       P1,#10101010B
            Sjmp      Loop

            End
```

Setelah program ditulis pada editor (EDIT di bawah MS-DOS), simpanlah dengan nama **TombLed1.asm** lakukan kompilasi program hingga menghasilkan berkas dalam format hex, kemudian lakukan download menggunakan program AEC_ISP.

Program berikut mengimplementasikan penekanan tombol P3.0 untuk menghidupkan LED pada port 1 dan penekanan tombol P3.1 untuk mematikannya.

```
            Org         0h
Mulai :     Mov         A,P3
            Cjne        A,#0FEh,Terus
            Mov         P1,#0
            Sjmp        Mulai
Terus :     Cjne        A,#0FDh,Mulai
            Mov         P1,#0FFh
            Sjmp        Mulai
            End
```

Setelah program ditulis pada editor (EDIT di bawah MS-DOS), simpanlah dengan nama **TombLed2.asm** lakukan kompilasi program hingga menghasilkan berkas dalam format hex, kemudian lakukan download menggunakan program AEC_ISP.

Program berikut mengimplementasikan tombol P3.0 sebagai saklar toggle untuk menghidupkan dan mematikan LED pada port 1.

```
            Org         0h
Mulai :     Mov         A,P3
            Cjne        A,#0FEh,Mulai
            Cjne        R0,#0,Terus
            Mov         R0,#1
            Mov         P1,#0
Tunggu :    Mov         A,P3
            Cjne        A,#0FFh,Tunggu
            Sjmp        Mulai
Terus :     Mov         R0,#0
            Mov         P1,#0FFh
            Sjmp        Tunggu
            End
```

Setelah program ditulis pada editor (EDIT di bawah MS-DOS), simpanlah dengan nama **TombLed3.asm** lakukan kompilasi program hingga menghasilkan berkas dalam format hex, kemudian lakukan download menggunakan program AEC_ISP.

Pelajari program berikut dengan cermat, kemudian tuliskan dalam editor EDIT, simpan dengan nama **TombLed4.asm**, lakukan kompilasi dan download ke dalam modul mikrokontroler yang tersedia.

```
                Org     0h
                Mov     R3,#11111110B
                Mov     R2,#0
Tombol :        Jnb     P3.0,Jalan
                Call    Nyala
                Sjmp    Tombol
Jalan :         Cjne    R2,#1,Hidupkan
                Mov     R2,#0
                Sjmp    Tombol
Hidupkan :      Mov     R2,#1
                Sjmp    Tombol
Nyala :         Cjne    R4,#1,Mati
                Mov     P1,R3
                Call    Tunggu
                Mov     A,R3
                RL      A
                Mov     R3,A
                Ret
Mati :          Mov     P1,R3
                Call    Tunggu
                Ret
Tunggu :        Mov     R0,#0
Tunggu1 :       Mov     R1,#0
                Djnz    R1,$
                Djnz    R0,Tunggu1
                Ret
                End
```

Program berikut merupakan aplikasi fungsi interupsi eksternal (INT0) sehingga masukan pada pin P3.2 dapat menerima pesan interupsi.

```
          Org     0h
          Sjmp    Awal
          Org     03h
          Mov     P1,#10101010B
          Mov     R5,#0FFh
Ulang :   Djnz    R5,Ulang
          Reti
Awal :    Mov     IE,#81h
          Mov     IP,#01h
```

```
Mulai :   Mov       P1,#00001111B
          Acall     Delay
          Mov       P1,#11110000B
          Acall     Delay
          Sjmp      Mulai

Delay :   Mov       R0,#5
Delay1 :  Mov       R1,#0FFh
Delay2 :  Mov       R2,#0
          Djnz      R2,$
          Djnz      R1,Delay2
          Djnz      R0,Delay1
          Ret

          End
```

Setelah program ditulis pada editor (EDIT di bawah MS-DOS), simpanlah dengan nama **IntEks01.asm** lakukan kompilasi program hingga menghasilkan berkas dalam format hex, kemudian lakukan download menggunakan program AEC_ISP.

Pelajari program berikut dengan cermat, kemudian tuliskan dalam editor EDIT, simpan dengan nama **IntEks02.asm**, lakukan kompilasi dan download ke dalam modul mikrokontroler yang tersedia.

```
          Org       0h
          Sjmp      Mulai
          Org       0003h
          Sjmp      Extr0

Mulai :   Setb      EX0
          Setb      EA
          Setb      TR0

Ulang :   Mov       P1,#0Fh
          Call      Delay
          Mov       P1,#0F0h
          Call      Delay
          Jmp       Ulang

Extr0 :   Mov       A,#11111110B

Masih :   Mov       P1,A
          RL        A
          Call      Delay
          Cjne      A,#11111110B,Masih
          Reti

Delay :   Mov       R0,#0FFh
Delay1 :  Mov       R1,#0h
```

```
Delay2 :  Djnz      R1,Delay2
          Djnz      R0,Delay1
          Ret

          End
```

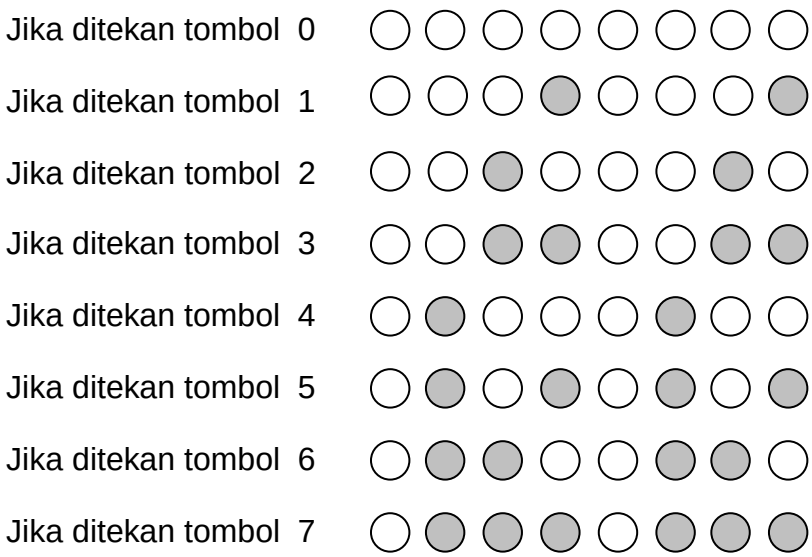
Pelajari program berikut dengan cermat, kemudian tuliskan dalam editor EDIT, simpan dengan nama **In_Tim01.asm**, lakukan kompilasi dan download ke dalam modul mikrokontroler yang tersedia.

```
          Org       0h
          Jmp       Utama
          Org       000Bh
          Jmp       Waktu0
          Lama      Equ       -10000
          Suwi      Equ       -20000

          Mov       R3,#100
          Setb      P1.0

Utama :   Mov       TMOD,#1
          Mov       TH0,#High Suwi
          Mov       TL0,#Low Suwi
          Setb      ET0
          Setb      EA
          Setb      TR0

Ulang :   Setb      P1.7
          Setb      P1.6
          Clr       P1.5
          Clr       P1.4
          Call      Delay
          Clr       P1.7
          Clr       P1.6
          Setb      P1.5
          Setb      P1.4
          Call      Delay
          Jmp       Ulang

Waktu0 :  Mov       TH0,#High Lama
          Mov       TL0,#Low Lama
          Djnz      R3,Terus
          Cpl       P1.0
          Mov       R3,#100

Terus :   Reti

Delay :   Mov       R0,#0FFh
Delay1 :  Mov       R1,#0h
```

```
Delay2 :  Djnz      R1,Delay2
          Djnz      R0,Delay1
          Ret

          End
```

Tugas-2 :

1. Setelah dijalankan, apa yang dihasilkan dari program-program tersebut ?
2. Jelaskan maksud pada setiap instruksi pada program-program di atas !
3. Buatlah diagram alir (flow chart) untuk program-program di atas !
4. Susunlah satu program untuk dapat mengendalikan nyala LED dengan pola yang berbeda jika ditekan tombol yang berbeda ! Kemudian tunjukkan apakah program itu sesuai dengan yang dimaksudkan. Contoh (atau yang lainnya) :



Percobaan-3

Mengendalikan Peraga 7-Segmen

Tujuan :

Membuat aplikasi pada mikrokontroler AT89S51 untuk mengendalikan nyala LED pada peraga 7-segmen. LED-LED pada setiap peraga 7-segmen dapat menampilkan (mengesankan) berbagai karakter (huruf atau angka).

Contoh program berikut untuk mengendalikan penampil 7-Segmen yang dapat menampilkan karakter.

```
            Org         0h

Mulai :     Mov         DPTR,#KARAKTER
            Mov         R6,#08h
            Mov         R1,#7Fh

Ulang :     Clr         A
            Movc        A,@A+DPTR
            Inc         DPTR
            Mov         P0,A
            Mov         A,R1
            Mov         P1,A
            RR          A
            Mov         R1,A
            Mov         R2,#0FFh

Delay :     Djnz        R2,Delay
            Mov         P0,#0FFh
            Djnz        R6,Ulang
            Jmp         Mulai

KARAKTER:   DB    0FDH,70H,24H,0Bah
            DB    77H,62H,0A0H,0FDh

            End
```

Setelah program ditulis pada editor (EDIT di bawah MS-DOS), simpanlah dengan nama **Segmen71.asm** lakukan kompilasi program hingga menghasilkan berkas dalam format hex, kemudian lakukan download menggunakan program AEC_ISP.

Pelajari program berikut dengan cermat, kemudian tuliskan dalam editor EDIT, simpan dengan nama **Segmen72.asm**, lakukan kompilasi dan download ke dalam modul mikrokontroler yang tersedia.

```
            Org         0h

            Mov         Dptr,#Numeric

Ulang :     Clr         A
            MovC        A,@A+Dptr
            Mov         P0.A
            Mov         A,#11111110B

Scan :      Mov         P1,A
            Call        Delay
            RL          A
            Cjne        A,#11111110B,Scan
            Inc         Dptr
            Call        Ldelay
            Jmp         Ulang

Sdelay :    Mov         R7,#0FFh
            Djnz        R7,$
            Ret

Ldelay :    Mov         R6,#0h
Lagi :      Call        Sdelay
            Call        Sdelay
            Djnz        R6,Lagi
            Ret

Numeric :   DB    22h,77h,0A4h,25h,71h
            DB    29h,28h,67h,20h,21h

            End
```

Pelajari program berikut dengan cermat, kemudian tuliskan dalam editor EDIT, simpan dengan nama **Segmen73.asm**, lakukan kompilasi dan download ke dalam modul mikrokontroler yang tersedia.

```
            Org         0h

            Mov         R5,#0
            Mov         R6,#0
            Mov         R7,#0

Mulai :     Mov         Dptr,#Numeric
            Clr         A
            Mov         A,R7

            MovC        A,@A+Dptr
            Mov         P1,#11111101B
            Mov         P0.A
            Call        SDelay
```

```
            Clr       A
            Mov       A,R6

            MovC      A,@A+Dptr
            Mov       P1,#11111011B
            Mov       P0.A
            Call      SDelay

            Djnz      R5,Mulai

            Inc       R7
            Cjne      R7,#10,Mulai
            Mov       R7,#0

            Inc       R6
            Cjne      R6,#10,Mulai
            Mov       R6,#0

            Jmp       Mulai

Sdelay :    Mov       R4,#0FFh
            Djnz      R4,$
            Ret

Numeric :   DB    22h,77h,0A4h,25h,71h
            DB    29h,28h,67h,20h,21h

            End
```

Tugas-3 :

1. Setelah dijalankan, apa yang dihasilkan dari program tersebut ?
2. Jelaskan maksud pada setiap baris/ instruksi !
3. Buatlah diagram alir (flow chart) untuk program di atas !
4. Susunlah program untuk menyalakan 8 buah peraga 7-segmen dengan pola bergiliran yang masing-masing menampilkan angka desimal secara urut dari 0 s/d 7 dari kiri ke kanan.

Percobaan-4

Membaca Key Pad Matrik 3 x 4

Tujuan :

Membuat aplikasi pada mikrokontroler AT89S51 untuk membaca tombol pada Key Pad Matrik dan menampilkan hasilnya pada petaga 7-segmen.

Program berikut digunakan untuk menampilkan angka desimal pada 7-segmen yang dikendalikan melalui penekanan tombol pada Key Pad. Program dasarnya adalah sebagai berikut :

```
KeyData    Equ  50h
KeyBounc   Equ  51h

           Org  0h
           Mov  Dptr,#Numeric

Ulang :    Call KeyPad3x4
           Mov  A,KeyData
           Cjne A,#0FFh,Ditekan
           Jmp  Ulang

Ditekan :  Movc A,@A+Dptr
           Clr  P1.1
           Mov  P0,A
           Jmp  Ulang

KeyPad3x4 :
           Mov  KeyBounc,#50
           Mov  P3,#0FFh
           Clr  P3,0

Ul1 :      Jb   P3.3,Key1
           Djnz KeyBounc,Ul1
           Mov  KeyData,#1
           Ret

Key1 :     Jb   P3.4,Key2
           Djnz KeyBounc,Key1
           Mov  KeyData,#4
           Ret

Key2 :     Jb   P3.5,Key3
           Djnz KeyBounc,Key2
           Mov  KeyData,#7
           Ret

Key3 :     Jb   P3.6,Key4
           Djnz KeyBounc,Key3
```

```
            Mov  KeyData,#0Eh
            Ret

Key4 :      Setb P3.0
            Clr  P3.1
            Jb   P3.3,Key5
            Djnz Keybounc,Key4
            Mov  Keydata,#2
            Ret

Key5 :      Jb   P3.4,Key6
            Djnz KeyBounc,Key5
            Mov  KeyData,#5
            Ret

Key6 :      Jb   P3.5,Key7
            Djnz KeyBounc,Key6
            Mov  KeyData,#8
            Ret

Key7 :      Jb   P3.6,Key8
            Djnz KeyBounc,Key7
            Mov  KeyData,#0
            Ret

Key8 :      Setb P3.1
            Clr  P3.2
            Jb   P3.3,Key9
            Djnz Keybounc,Key8
            Mov  Keydata,#3
            Ret

Key9 :      Jb   P3.4,Key10
            Djnz KeyBounc,Key9
            Mov  KeyData,#6
            Ret

Key10 :     Jb   P3.5,Key11
            Djnz KeyBounc,Key10
            Mov  KeyData,#9
            Ret

Key11 :     Jb   P3.6,Key12
            Djnz KeyBounc,Key11
            Mov  KeyData,#0Fh
            Ret

Key12 :     Mov  KeyData,#0FFh
            Ret

Numeric :   DB   0C0h,0F9h,0A4h,0B0h,99h
            DB   92h,82h,0F8h,80h,90h

            End
```

Setelah program ditulis pada editor (EDIT di bawah MS-DOS), simpanlah dengan nama **KeyPad1.asm**, lakukan kompilasi program hingga menghasilkan berkas dalam format hex, kemudian lakukan download menggunakan program AEC_ISP.

Tugas-4 :

1. Setelah dijalankan, apa yang dihasilkan dari program tersebut ?
2. Jelaskan maksud pada setiap baris/ instruksi dalam program di atas !
3. Buatlah diagram alir (flow chart) untuk program di atas !
5. Susunlah program, jika tombol A ditekan dapat menambah bilangan yang tampil dan jika tombol B yang ditekan dapat menguranginya.

Percobaan-5

Menggenerasi Bunyi pada Loudspeaker

Tujuan :

Membuat aplikasi pada mikrokontroler AT89S51 untuk menggenerasi bunyi melalui loudspeaker dengan frekuensi tertentu.

Program untuk membunyikan loadspeaker dengan frekuensi 500 Hz adalah sebagai berikut :

```
Frek :    Equ  -1000
          Org  0h
          Mov  TMOD,#01h
Ulang :   Mov  TH0,#High Frek
          Mov  TL0,#Low Frek
          Setb TR0
Tunggu :  Jnb  TF0,Tunggu
          Clr  TR0
          Clr  TF0
          CPL  P1.7
          Sjmp Ulang
          End
```

Tugas-5 :

1. Setelah dijalankan, apa yang dihasilkan dari program tersebut ?
2. Jelaskan maksud pada setiap baris/ instruksi dalam program di atas !
3. Buatlah diagram alir (flow chart) untuk program di atas !
4. Susunlah program agar Key-Pad Matrix 3 x 4 dapat difungsikan sebagai tombol suatu organd sederhana. Misalnya ketika tombol 1 ditekan maka luodspeaker akan mengeluarkan bunyi dengan frekuensi nada Do (260 Hz), ketika tombol 2 ditekan maka luodspeaker akan mengeluarkan bunyi dengan frekuensi nada Re (290 Hz), ketika tombol 3 ditekan maka luodspeaker akan mengeluarkan bunyi dengan frekuensi nada Mi (330), dan seterusnya ! (Fa = 350 Hz, Sol = 390 Hz, La = 440 Hz, Si = 490 Hz).

Percobaan-6

Mengendalikan ADC 0804

Program berikut untuk mendemonstrasikan konversi tegangan analog menjadi digital dengan rangkaian ADC-0804. Keluaran digitalnya berupa word biner yang ditampilkan melaui nyala LED pada port-1. Programnya adalah :

```
            ADC_CS    Bit  P2.7
            ADC_RD    Bit  P2.6
            ADC_WR    Bit  P2.5
            ADC_INT   Bit  P2.4

            Org  0h

Mulai :     Clr  ADC_CS
            Clr  ADC_WR
            Setb ADC_WR

EOC_St:     Jb   ADC_INT,EOC_St

Delay :     Djnz R2,$
            Djnz R3,Delay
            ClrADC_RD
            Djnz R3,$
            Mov  A,P3
            Setb ADC_RD
            Setb ADC_CS
            Cpl  A
            Mov  P1,A
            Sjmp Mulai

            End
```

Setelah program ditulis pada editor (EDIT di bawah MS-DOS), simpanlah dengan nama **ADC_04.asm** lakukan kompilasi program hingga menghasilkan berkas dalam format hex, kemudian lakukan download menggunakan program AEC_ISP.

Tugas-6 :

1. Setelah dijalankan, apa yang dihasilkan dari program tersebut ?
2. Jelaskan maksud pada setiap baris/ instruksi dalam program di atas !
3. Buatlah diagram alir (flow chart) untuk program di atas !

Percobaan-7

Mengendalikan Putaran Motor Stepper

Setelah program ditulis pada editor (EDIT di bawah MS-DOS), simpanlah dengan nama **Mo_Step.asm** lakukan kompilasi program hingga menghasilkan berkas dalam format hex, kemudian lakukan download menggunakan program AEC_ISP.

Tugas-7 :

1. Setelah dijalankan, apa yang dihasilkan dari program tersebut ?
2. Jelaskan maksud pada setiap baris/ instruksi dalam program di atas !
3. Buatlah diagram alir (flow chart) untuk program di atas !

Percobaan-8

Pengendali LCD

Percobaan-9

Komunikasi Serial dengan Komputer

Percobaan-10

Tugas Mandiri Berkelompok

Tentukanlah satu tugas/proyek perancangan dan implementasinya mengenai suatu sistem (rangkaian) yang dikendalikan dengan mikrokontroler AT89S51. Misalnya :

1. Alat ukur kecepatan fluida (angin, arus air) yang mencakup laju dan arah.
2. Sistem akusisi data (data logger) dengan ADC.
3. ‘Robot’ pendeteksi suhu.
4. ‘Robot’ pendeteksi cahaya.
5. Alat pengendali suhu oven.
6. Alat pendeteksi getaran mekanik (deteksi dini tsunami, gempa) yang mencakup frekuensi dan intensitas (energi getaran).
7. Alat cacah pil.
8. Alat cacah putaran mekanik.
9. Alarm otomatis terkendali waktu (suhu, cahaya, suara khas, dll.).
10. Alat ukur panjang lembaran kertas/kain yang digulung.
11. Simulasi tegangan analog dengan DAC.
12. Organ.
13. Pola tulisan dinamis pada LCD.
14. Dll.