Ringkasan Set Instruksi

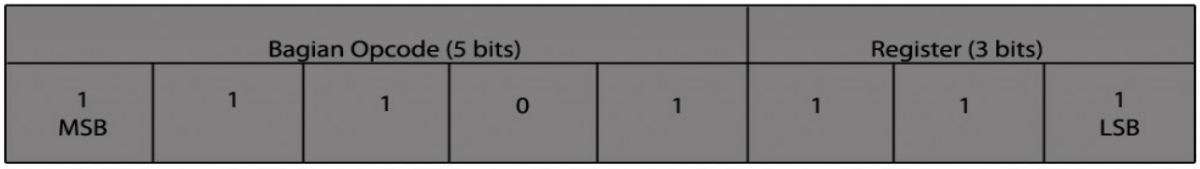
Dan

Mode pengalamatan (Addressing Mode)

Mikroprosessor 8051, sebagaimana terdaftar dalam *8051 set instruction in numerical order* memiliki sekumpulan instruksi yang terintegrasi kedalam beberapa mode pengalamatan, antara lain :

1) Register Addressing

Dalam pengalamatan register, register R0 sampai R7 dari register bank yang telah dipilih, akumulator, B-register, bit pembawa dan DPTR digunakan. Sebuah instruksi MCS-51 menggunakan mode pengalamatan ini menunjukkan register-register R0 sampai R7 opcode-nya sendiri. Bit-bit yang kurang signifikan dari opcode mengindikasikan register mana yang akan digunakan. hal ini ditunjukkan oleh gambar berikut



Opcode dari MOV A, Instruksi R7
(Tiga bit yang kurang signifikan merujuk pada Register yang digunakan)

8051 memiliki 8 buah register kerja (R0 – R7) masing- masing berukuran 16 Bit. Dimana masing-masing register memilik penempatan alamat dalam IRAM (Internal Memory) yang ditunjuk oleh jenis atau tipe bank registernya.

Berikut adalah bank register yang terdapat di mikroprosesor 8051

Bank Register	**RS0 (PSW.4)**	**RS1 (PSW.3)**
Bank 0	0	0
Bank 1	0	1

Bank 2	1	0
Bank 3	1	1

Bank Register	Lokasi (R0-R7) di Internal RAM
Bank 0	(00h – 07h)
Bank 1	(08h – 0fh)
Bank 2	(10h – 17h)
Bank 3	(18h – 1fh)

Adapun **RS0** dan **RS1** ialah nilai dari *program status word* **(PSW),** tepatnya berada pada bit ke 3 dan ke 4 (PSW.3 dan PSW.4)**.**

adapun contoh dari pengalamatan register, misalnya **ADD A,R7**

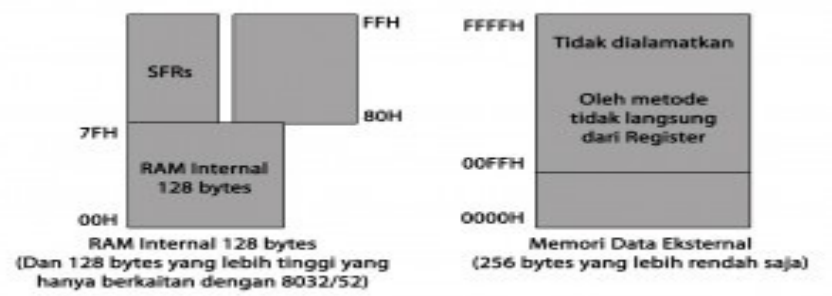
2) Direct Addressing (Pengalamatan Langsung)

Pengalamatan langsung dapat diakses oleh hamper semua register ataupun variable. Misalkan operasi transfer data yang melibatkan penggunaan dua buah register, **MOV P0,A.** Dalam mode pengalamatan langsung, alamat langsung dari operand ditentukan oleh instruksi itu sendiri sebagaimana ditunjukkan di bawah. Mode pengalamatan langsung menggunakan 128 bytes RAM internal yang lebih rendah dan register fungsi khusus (SFR). Sebagai contoh, instruksi MOV A, Direct, menggunakan alamat langsung dari operand sumber. Maka, MOV A, 54H akan mentransfer konten dari lokasi memori on-chip, yang mana alamatnya adalah 54H di akumulator. Dengan cara serupa, untuk membaca konten dari SFR SBUF ke dalam akumulator, kita dapat menggunakan instruksi MOV A, SBUF. Perlu dicatat bahwa SFR SBUF memiliki alamat langsung 99H, yang terletak di 128 bytes RAM on-chip yang lebih lebih tinggi

Pengalamatan Langsung

3) Pengalamatan register tidak langsung

Pengalamatan register tidak langsung menggunakan salah satu dari register-register R0 atau R1, dari register bank yang telah dipilih, sebagai penunjuk ke lokasi di blok memori data sebesar 256 bytes. Hal tersebut dapat merujuk ke 128 bytes RAM internal yang lebih rendah, (dan 128 bytes RAM internal yang lebih tinggi berkaitan dengan 8032/52), atau 256 bytes memori data eksternal yang lebih rendah. Gbr.4.4 menunjukkan alamat memori yang dialamatkan oleh mode pengalamatan tidak langsung. Perlu dicatat bahwa SFRs tidak dialamatkan oleh mode ini. Secara serupa, memori eksternal melampaui 256 bytes yang lebih rendah tidak dialamatkan.



Ruang memori yang dialamatkan oleh Register dengan Mode pengalamatan tidak langsung

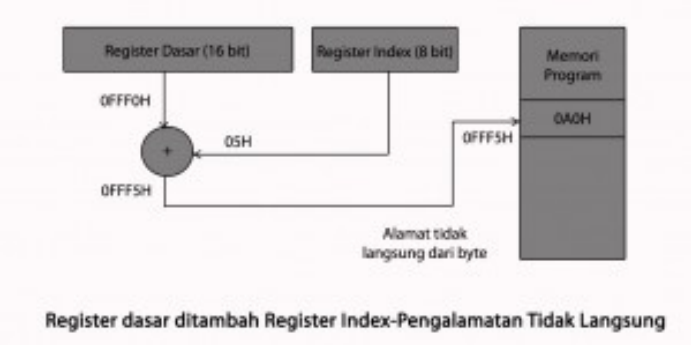
4) Pengalamatan segera (Immediate Addressing)

Pengalamatan segera memungkinkan penggunaan data segera (konstan) sebagai bagian dari instruksi. Sebagai contoh, instruksi MOV A, #45H, akan menyimpan data 45H konstan atau segera di dalam akumulator. Perlu dicatat bahwa symbol # di depan konstan mengindikasikan tipe data segera.

5) Pengalamatan Register dasar ditambah (Register Index)

Pengalamatan register indeks Mode ini memungkinkan byte diakses dari memori program, dimana alamat dihitung sebagai jumlah dari register dasar (DPTR atau PC) dan register indeks,

akumulator. Sebagai contoh, instruksi **MOVC A, @A+DPTR** akan mengambil byte dari memori program, dimana alamatnya dihitung dengan menambahkan konten 8-bit tak bertanda yang asli dari akumulator dan konten 16-bit dari DPTR. Jika DPTR-nya memiliki 0FFF0H dan akumulatornya memiliki 05H, maka byte yang tersimpan di 0FFF5H akan dikopi kedalam akumulator. Metode ini membantu akses tabel pencarian.



Register dasar ditambah Register Index-Pengalamatan Tidak Langsung

2.2.1.2. Set instruksi dalam MCS51

Secara keseluruhan MCS51 memiliki 255 macam instruksi yang dibentuk dengan mengkombinasikan data dan operan.instruksi tersebut dikelompokkan sebagai berikut :

1. Kelompok Penyalinan Data

Instruksi yang mewakili kelompok ini ialah MOV, singkatan dari *move* yang artinya memindahkan. Instruksi MOV dibedakan sesuai dengan jenis memori MCS51. Berikut beberapa aturan baku untuk instruksi MOV, diantaranya :

Instruksi Penyalinan Data	deskripsi
MOV A,Rx	Salin data dari *register* ke akumulator A
MOV A,iramAddress	Salin data pada lokasi RAM ke A
MOV A,@Rx	Salin data pada lokasi Rx ke A
MOV A,#data*Byte*	Isi akumulator dengan suatu nilai *byte*
MOV Rx,A	Salin isi A ke *register*
MOV Rx,iramAddress	Salin isi lokasi memori ke *register*

MOV Rx,#data*Byte*	Isi *register* dengan nilai *byte*
MOV iramAddress,A	Salin isi akumulator ke lokasi memori
MOV iramAddress,Rx	Salin isi *register* ke lokasi memori
MOV iramAddress,iramAddress	Salin isi antar lokasi memori
MOV iramAddress,@Rx	Salin isi lokasi memori Rx ke lokasi memori
MOV iramAddress,#data*Byte*	Isi lokasi memori dengan nilai *byte*
MOV @Rx,A	Salin isi a ke lokasi memori Rx
MOV @Rx,iramAddress	Salin isi lokasi memori ke lokasi memori Rx
MOV @Rx,#data*Byte*	isi lokasi memori dengan nilai *byte*
MOV C,*bit*	salin nilai *bit register* data ke *register* C
MOV *bit*,C	salin nilai *bit* di C ke *bit* di *register*
MOV DPTR,#data16	isi DPTR dengan nilai *byte*
MOVC A,@A+DPTR	DPTR sebagai *register* tak langsung
MOVC A,@A+PC	PC sebagai *register* tak langsung
MOVX A,@Rx	Salin data eksternal lokasi RX ke A
MOVX A,@DPTR	Salin data eksternal lokasi DPTR ke A
MOVX @Rx,A	Salin data eksternal dari A ke lokasi Rx
MOVX @DPTR,A	Salin data eksternal dari A ke lokasi DPTR

2. Kelompok Instruksi Aritmatik

Instruksi dalam kelompok aritmatik selalu melibatkan akumulator dan hanya beberapa yang melibatkan *register* lainnya (seperti DPTR dan lain-lain), berikut akan dibahas satu-persatu instruksi yang termasuk dalam kelompok ini, melalui tabel jenis instruksi beserta deskripsi instruksinya.

Jenis Instruksi Aritmatik	**Format instruksi**	**deskripsi**

Instruksi ADD dan ADDC	ADD A, Rx
	ADD A,iramAddress
	ADD A,@Rx
	ADD A,#data*Byte*
	ADDC A,Rx
	ADDC A,iramAddress
	ADDC A,@Rx
	ADDC A,#data*Byte*
		A <- A+ Rx
		A <- A+ isi lokasi memori
		A <-A+ isi lokasi yg ditunjuk R0
		A <- A + data *byte*
		---------- 	
		Sama dengan instruksii ADD, hanya saja untuk ADDC isi atau nilai *bit* C ikut dijumlahkan.

Instruksi SUBB	SUBB A,Rx
	SUBB A,iramAddress
	SUBB A,@Rx
	SUBB A,#data*Byte*
		A <- A - Rx
		A <- A - isi lokasi memori
		A <- A - [Rx]
		A <- A - data *byte*

Instruksi DA *(decimal adjust)*	DA A	Apabila setelah instruksi ADD atau ADDC,SUBB nilai *flag* AC =1 maka 4 low *bit* A ditambahkan dengan 6H, apabila *flag* CY = 1 maka 4 high *bit* A ditambahkan dengan 6
Instruksi perkalian MUL	MUL AB	Perkalian antara 8 *bit* Acc dengan

		8 *bit register* B akan menghasilkan bilangan 16 *bit*, dimana high *bit* disimpan di B sedangkan low *bit*nya di ACC
Instruksi pembagian DIV	DIV AB	Pembagian antara 8 *bit* ACC dengan 8 *bit register* B akan menghasilkan 8 *bit* bilangan, dimana hasil pembagian di ACC sedangkan sisa *(remainder)* pembagian di *register* B
Instruksi DEC dan INC	DEC A DEC Rx DEC iramAddress DEC @Rx INC A INC Rx INC iramAddress INC @Rx INC DPTR	A <- A – 1 Rx <- Rx – 1 Iramaddress <- Iramaddress – 1 [Rx] <- [Rx] – 1 A <- A +1 Rx <- Rx +1 Iramaddress <- Iramaddress +1 [Rx] <- [Rx] + 1 DPTR <- DPTR + 1

3. Kelompok Instruksi Logika

Kelompok instruksi ini dipakai untuk melakukan operasi logika, yaitu operasi AND (Instruksi ANL), operasi OR (ORL), operasi exclusive OR (Instruksi XRL,operasi *clear* (CLR, instruksi negasi atau komplemen (CPL), operasi pergeseran (RL, RR, RLC, RRC) serta operasi penukaran data (SWAP). Berikut akan dijabarkan melalui tabel instruksi selengkapnya.

Jenis Instruksi Logika	Format instruksi	Deskripsi
Instruksi AND	ANL A,Rx	A <- A AND Rx

	ANL A,iramAddress ANL A,@Rx ANL A,#data*Byte* ANL iramAddress,A ANL iramAddress,#data*Byte* ANL C,*bit* ANL C,/*bit*	A <- A AND isi memori A <- A AND [Rx] A <- A AND data *byte* IRam <- Iram AND A IRam <- Iram AND data C <- C AND *bit register* C <- C AND /*bit*
Instruksi OR	ORL A,Rx ORL A,iramAddress ORL A,@Rx ORL A,#data*Byte* ORL iramAddress,A ORL iramAddress,#data*Byte* ORL C,*bit* ORL C,/*bit*	A <- A OR Rx A <- A OR isi memori A <- A OR [Rx] A <- A OR data *byte* IRam <- Iram OR A IRam <- Iram OR data C <- C OR *bit register* C <- C OR /*bit*
Instruksi Exclusive OR	XRL A,Rx XRL A,iramAddress XRL A,@Rx XRL A,#data*Byte* XRL iramAddress,A XRL iramAddress,#data*Byte* XRL C,*bit* XRL C,/*bit*	A <- A XRL Rx A <- A XRL isi memori A <- A XRL [Rx] A <- A XRL data *byte* IRam <- Iram XRL A IRam <- Iram XRL data C <- C XRL *bit register* C <- C XRL /*bit*
Instrruksi *clear* dan set *bit*	CLR A CLR *bit*	A <- 00 *Bit register* <- 0 C<- 0

	CLR C SETB C SETB *bit*	C <- 1 *Bit register* <- 1
Instruksi negasi atau komplemen	CPL A CPL *bit* CPL C	A <- Neg A *Bit Register* <- Not *Bit* C <- Not C
Instruksi Pergeseran	RL A RLC A RR A RRC A	Melakukan pergeseran *bit* ke kiri dari *bit* yang ada di ACC, apabila RLC akan terjadi set CY apabila nilai di ACC >80H pergeseran *bit* ke kanan dari *bit* yang ada di ACC, apabila RRC akan terjadi set CY apabila pergeseran telah melewati *bit* 0 ACC
Operasi Penukaran data	SWAP A	Menukar 4*bit* High dengan 4*bit* Low pada ACC

Operasi yang dijabarkan pada tabel 2.4 telah mencakup beberapa kajian mengenai operasi *bit* pda MCS51, sebagaimana yang terjabarkan pada gambar 2.5 yang menggambarkan denah memori-*bit* (RAM) lokasi 20-2FH dan SFR, sebanyak 16 *byte* memori bisa dipakai untuk menyimpan 128 *bit* data *boolean* yang diberi alamat per*bit* mulai dengan *bit* lokasi 00-7FH. Selain itu operasi *bit* bisa berlaku di memori data lokasi 80-FFH, sehingga secara keseluruhan operasi *bit* dapat diberlakukan pada 256 lokasi RAM. Operasi *bit* yang dimaksud di sini dapat berupa pemberian nilai data biner, pemindahan data biner / *bit*, operasi logika itu sendiri.

5. Kelompok Instruksi Lompatan

Dalam menjalankan instruksi demi intstruksinya, mikrokontroller mempunya program *counter* (PC) yang selalu menyimpan lokasi dari memori program yang menyimpan instruksi berikutnya.

Kelompok instruksi lompatan terbagi menjadi beberapa kelompok, diantaranya

A) Kelompok Instruksi JUMP

B) kelompok instruksi untuk subrutin

C) kelompok instruksi lompatan bersyarat

D) kelompok Instruksi proses dan *test*

Berikut akan dijabarkan melalui tabel instruksi lompatan beserta format penulisan dan deskripsi instruksi.

Tabel 2.5. table instruksi lompatan

Kelompok Instruksi lompatan	Tipe instruksi	Format instruksi	Deskripsi
Kelompok Instruksi JUMP	LJMP	LJMP address (LJMP *label*)	Melakukan lompatan ke lokasi memori bisa berupa label pada penulisan script maupun alamat dengan ukuran 3 *byte*
	AJMP	AJMP address (AJMP *label*)	Lompatan yang dilakukan berukuran 2 *byte*
	SJMP	SJMP address (SJMP *label*)	Ditandai dengan pergeseran relatif 1 *byte* bilangan 2's complement (-128 - +127)
Kelompok Instruksi	ACALL	**ACALL** *proc*	Memanggil program

untuk Subrutin	LCALL	……. *Proc:* Isi subrutin **RET** **LCALL** *proc* ……. *Proc:* Isi subrutin **RET**	sub-rutin dalam daerah memori-program 2 KB Setara dengan LJMP bisa menjangkau memori program sebanyak 64 KB. RET digunakan untuk mengakhiri sebuah subrutin
Kelompok Instruksi lompatan bersyarat	JZ / JNZ	JNZ relativeAddress	If ACC <> 0 then Goto relativeAddress
		JZ relativeAddress	If ACC =0 then Goto relativeAddress
	JC / JNC	JC relativeAddress	If CY =1 then Goto relativeAddress
		JNC relativeAddress	If CY <> 1 then Goto relativeAddress
	JB / JNB / JBC	JB *bit*Address,relativeAddress	If *bit* set then Goto RelativeAddress If *bit* set then Goto RelativeAddress

		JBC *bit*Address,relativeAddress
		
		JNB *bit*Address,relativeAddress
			Clear bit = 0
			endif
			If *bit* not set then
			Goto
			RelativeAddress

Kelompok Instruksi Proses dan Test	DJNZ
	
	CJNE
		DJNZ Rx,relativeAddress
		
		DJNZ IRAMaddress,rel Address
		
		CJNE A,iramAddress,relAddress
		
		CJNE A,#data*Byte*,relAddress
		
		CJNE Rx,#data*Byte*,relAddress
		
		CJNE @Rx,#data*Byte*,relAdress
			Rx <- Rx – 1
			If Rx <> 0 then
			Goto
			RelativeAddress
			Dec IramAddress
			If Iram <> 0 then
			Goto
			RelativeAddress
			If A <> iramAddress
			then goto relAddress
			If A <> data*Byte*
			then goto relAddress
			If Rx <> data*Byte*
			then goto relAddress
			If [Rx] <> data*Byte*
			then goto relAddress
