Mikrokontroleri

Mikrokontroleri

Poglavlje IV

Skup instrukcija mikrokontrolera familije MCS-51

Uvod

- **Instrukciju računara čine kod operacije (*op-code*) iza koga mogu da slede jedan ili dva operanda**
- ***Op-cod* identifikuje tip operacije koja će biti izvršena nad operandima koji identifikuju izvor i odredište podataka**
- **Operand može biti:**
  - **Podatak koji se koristi u instrukciji**
  - **CPU registar**
  - **Memorijska likacija**
  - **Niki U/I port**
  - **Format instrukcije koja ima dva operanda je uvek oblikia:**

    Instrukcija Odredišni, Izvorni

Načini adresiranja

- Način adresiranja određuje kako će bajt operand bit selektovan
- Osam načina adresiranja

Načini adresiranja	Instrukcija
Registarsko	**MOV A, B**
Neposredno	**ADD A,#80H**
Direktno	**MOV 30H,A**
Indirektno	**ADD A,@R0**
Relativno*	**SJMP AHEAD**
Apsolutno*	**AJMP BACK**
Long*	**LJMP FAR_AHEAD**
Indeksno	**MOVC A,@A+PC**

*** Odnosi se na instrukcije grananja programa**

Registarsko adresiranje

- Razmena podataka između registara

- ***Primer:***

```
MOV  R0, A
MOV  R1, R2
```

- Instrukcija prenosi sadržaj akumulatora u registar R0. Registarska banka (Banka 0, 1, 2 ili 3) je prethodno specificirana

Neposredno adresiranje

- Ovaj način koristi 8 ili 16-bitnu konstantu kao izvorni operand
- Odredišni registar mora imati istu širinu reči kao izvorni operand
- ***Primer:***

```
ADD A,#30h          ;Saberi 8-bitnu
                    ;vrednost 30h sa
                    ;akumulatorom

MOV DPTR,#0FE00h    ;Upisi 16-bitnu
                    ;konstantu FE00
                    ;u DPTR
```

Direktno adresiranje

- Ovaj način omogućava specificiranje operanda njegovom pravom adresom u memoriji (obično u heksadecimalnom formatu) ili simboličkom oznakom

- ***Primer:***

```
MOV  A, P3      ;Prenos sadrzaja porta
                ;P3 u akumulator
MOV  A, 020h    ;Prenos sadrzaja RAM
                ;lokacije 20h u akumulator
```

Indirektno adresiranje

- Ovaj način koristi pointer koji čuva aresu operanda
- Pointeri mogu biti samo registri R0, R1 i DPTR
- R0 i R1 čuvaju 8-bitnu adresu, dod DPTR čuna 16-bitnu adreru

- ***Primer:***

```
MOV  @R0,A       ;Smesti sadrzaj
                 ;akumulatora u memorijsku
                 ;lokaciju na koju ukazuje
                 ;registar R0
MOVX A, @DPTR    ;Prenos sadrzaja RAM
                 ;lokacije na koju ukazuje
                 ;DPTR u akumulator
```

Relativno adresiranje

- Koristi se sa nekim instrukcijama prograskog skoka, kao što je SJMP ili uslovnog skoka kao JNZ
- Ove instrukcije omogućavaju grananje programa na nove lokacije
- Odredišna adresa mora biti u granicama -128 do +127 bajtova od tekuće adrse

- ***Primer:***

```
Nazad: DEC A          ;Dekrementiraj A
       JNZ Nazad      ;Ako A nije nula
                      ;skoči nazad
```

Apsolutno adresiranje

- Adresiranje memorijskih lokacija programske memorije
- Ovo su 2-bajtne instrukcije gde je 11 bitova apsolutne adrese specificirano kao operand
- Viših 5 bitova 16-bitne PC adrese se ne modifikuje. Adresiranje unutar tekuće 2K stranice (2^{11} = 2048)

- ***Primer:***

```
        ACALL INIT
        ...
INIT:   MOV SCON,#50h      ;podprogram
        ...
        ...
```

Dugačko (*Long*) adresiranje

- Adresiranje memorijskih lokacija programske memorije instrukcijama LCALL i LJMP
- Ovo su 3-bajtne instrukcije gde je 16 bitova odredišne adrese specificirano kao operand
- Dozvoljava adresiranje 64 K kodnog prostora
- Program skače na istu lokaciju bez obzira odakle se vrši skok

- ***Primer:***

```
        LCALL INIT
        ...
INIT:   ORL TMOD,#01h      ;podprogram
```

Indeksno adresiranje

- Korisno za čitanje podataka iz tabele
- 16-bitni registar kao bazni registar i akumulator kao indeks - pomeraj
- Zbir ova dva registra formira efektivnu adresu za instrukcije JMP i MOVC

- ***Primer:***

```
MOV   A,#08H          ;Ofset od pocetka tabele
MOV   DPTR,#01F00h    ;Startna adresa tabele
MOVC  A,@A+DPTR       ;Citanje vrednosti iz
                      ;tabele sa adrse
                      ;start adresa + offset
                      ;i smestanje u A
```

Tipovi instrukcija

- **MCS 51 ima instrukcije podeljene u pet funkciionalne grupe:**
  - **Aritmetičke operacije**
  - **Logičke operacije**
  - **Operacije prenosa podataka**
  - ***Boolean* operacije**
  - **Operacije za kontrolu izvršenja programa**

Aritmetičke operacije

- Aritmetičke instrukcije ne vode računa o formatu podataka, da li su to označene ili neoznačene vrednosti, BCD brojvi, ASCII ili slično.
- Operacije postavljaju odgovarajuće bitove u registru PSW na osnovu kojih se u softveru može vršiti dalja obrada podataka

Mnemonic	Description
ADD A, Rn	A = A + [Rn]
ADD A, direct	A = A + [direct memory]
ADD A,@Ri	A = A + [memory pointed to by Ri]
ADD A,#data	A = A + immediate data
ADDC A,Rn	A = A + [Rn] + CY
ADDC A, direct	A = A + [direct memory] + CY
ADDC A,@Ri	A = A + [memory pointed to by Ri] + CY
ADDC A,#data	A = A + immediate data + CY
SUBB A,Rn	A = A - [Rn] - CY
SUBB A, direct	A = A - [direct memory] - CY
SUBB A,@Ri	A = A - [@Ri] - CY
SUBB A,#data	A = A - immediate data - CY
INC A	A = A + 1
INC Rn	[Rn] = [Rn] + 1
INC direct	[direct] = [direct] + 1
INC @Ri	[@Ri] = [@Ri] + 1
DEC A	A = A - 1
DEC Rn	[Rn] = [Rn] - 1
DEC direct	[direct] = [direct] - 1
DEC @Ri	[@Ri] = [@Ri] - 1
MUL AB	Multiply A & B
DIV AB	Divide A by B
DA A	Decimal adjust A

- **[@Ri] ukazuje na sadržaj memorijske lokacije na koju ukazuju registri R0 ili R1**
- **Rn se odnosi na registre R0-R7 selektovane registarse banke**

Logičke operacije

- Logičke instrukcije izvršavaju bulove operacije (AND, OR, XOR i NOT) nad bajtovima podataka po principu bit-po-bit

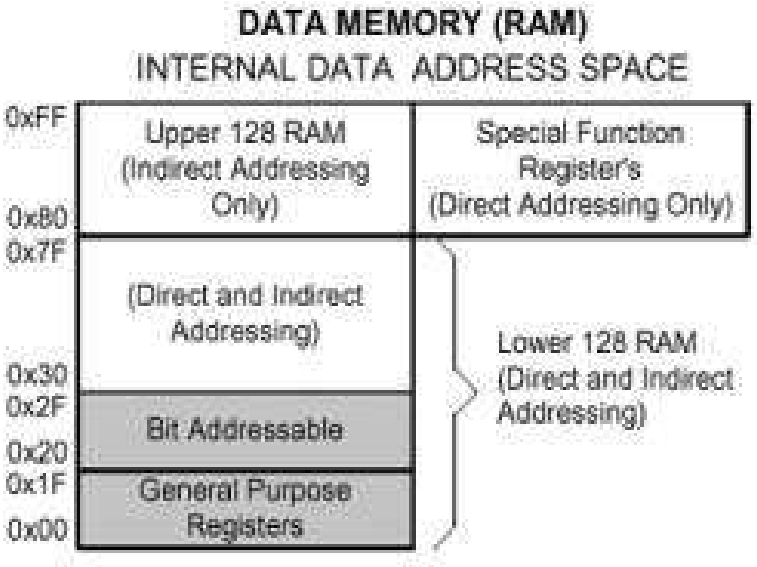
Primer:

```
ANL  A, #02H
     ;Mask bit 1
ORL  TCON, A
     ;TCON=TCON-OR-A
```

Mnemonic	Description
ANL A, Rn	A = A & [Rn]
ANL A, direct	A = A & [direct memory]
ANL A,@Ri	A = A & [memory pointed to by Ri]
ANL A,#data	A= A & immediate data
ANL direct,A	[direct] = [direct] & A
ANL direct,#data	[direct] = [direct] & immediate data
ORL A, Rn	A = A OR [Rn]
ORL A, direct	A = A OR [direct]
ORL A,@Ri	A = A OR [@RI]
ORL A,#data	A = A OR immediate data
ORL direct,A	[direct] = [direct] OR A
ORL direct,#data	[direct] = [direct] OR immediate data
XRL A, Rn	A = A XOR [Rn]
XRL A, direct	A = A XOR [direct memory]
XRL A,@Ri	A = A XOR [@Ri]
XRL A,#data	A = A XOR immediate data
XRL direct,A	[direct] = [direct] XOR A
XRL direct,#data	[direct] = [direct] XOR immediate data
CLR A	Clear A
CPL A	Complement A
RL A	Rotate A left
RLC A	Rotate A left (through C)
RR A	Rotate A right
RRC A	Rotate A right (through C)
SWAP A	Swap nibbles

Operacije prenosa podataka

- Daju mogućnost prenosa podatka između lokacija unutrašnjeg rama i SFR lokacija bez posredstva akumulatora
- Prenos podataka između unutrašnjeg i spoljašnjeg rama korišćenjem indirektnog adresiranja



Mnemonic	Description
MOV @Ri, direct	[@Ri] = [direct]
MOV @Ri, #data	[@Ri] = immediate data
MOV DPTR, #data 16	[DPTR] = immediate data
MOVC A,@A+DPTR	A = Code byte from [@A+DPTR]
MOVC A,@A+PC	A = Code byte from [@A+PC]
MOVX A,@Ri	A = Data byte from external ram [@Ri]
MOVX A,@DPTR	A = Data byte from external ram [@DPTR]
MOVX @Ri, A	External[@Ri] = A
MOVX @DPTR,A	External[@DPTR] = A
PUSH direct	Push into stack
POP direct	Pop from stack
XCH A,Rn	A = [Rn], [Rn] = A
XCH A, direct	A = [direct], [direct] = A
XCH A, @Ri	A = [@Rn], [@Rn] = A
XCHD A,@Ri	Exchange low order digits

Bulove operacije

- CPU može izvršavati jednobitne operacije
- Operacije uključuju *set, clear*, *and, or* i *complement* instrukcije
- Prenos na bit nivou
- Pristup bitovima preko direktnog adresiranja

Mnemonic	Description
CLR C	Clear C
CLR bit	Clear direct bit
SETB C	Set C
SETB bit	Set direct bit
CPL C	Complement c
CPL bit	Complement direct bit
ANL C,bit	AND bit with C
ANL C,/bit	AND NOT bit with C
ORL C,bit	OR bit with C
ORL C,/bit	OR NOT bit with C
MOV C,bit	MOV bit to C
MOV bit,C	MOV C to bit
JC rel	Jump if C set
JNC rel	Jump if C not set
JB bit,rel	Jump if specified bit set
JNB bit,rel	Jump if specified bit not set
JBC bit,rel	if specified bit set then clear it and jump

Instrukcije programskog skoka

- Za kontrolu toka izvršenja programa
- Instrukcije koje daju mogućnost odluke o prograskom skoku (uslovni skokovi)

Mnemonic	Description
ACALL addr11	Absolute subroutine call
LCALL addr16	Long subroutine call
RET	Return from subroutine
RETI	Return from interrupt
AJMP addr11	Absolute jump
LJMP addr16	Long jump
SJMP rel	Short jump
JMP @A+DPTR	Jump indirect
JZ rel	Jump if A=0
JNZ rel	Jump if A NOT=0
CJNE A,direct,rel	Compare and Jump if Not Equal
CJNE A,#data,rel	
CJNE Rn,#data,rel	
CJNE @Ri,#data,rel	
DJNZ Rn,rel	Decrement and Jump if Not Zero
DJNZ direct,rel	
NOP	No Operation