

Univerzitet u Novom Sadu
Fakultet tehničkih nauka
Departman za energetiku, elektroniku i telekomunikacije
Katedra za elektroniku

MIKROPROCESORSKA ELEKTRONIKA

**Beleške sa predavanja
#03**

Pripremio: prof. dr Veljko Malbaša
Novi Sad, oktobar 2007. godine

Skup instrukcija i programiranje mikrokontrolera

Instrukcije prenosa podataka

Instrukcije prenosa podataka, simboličko ime *MOV*, ima sledeći opšti oblik:

$$MOV \text{ dest, source} \quad ; \text{dest} \leftarrow \text{source}$$

Kod izvršenja ove instrukcije, podatak čiju adresu određuje *source* kopira se u registar ili memorijsku lokaciju koja je određena sa *dest*. Podatak koji se prenosi naziva se *operand*. Na primer, sledeće instrukcija kopira sadržaj registra *R3* u akumulator *A*:

$$MOV \text{ A, R3} \quad ; A \leftarrow R3$$

Posle izvršenja ove instrukcije, novi sadržaj registra *A* biće jednak sadržaju registra *R3*, pri čemu prethodni sadržaj *R3* ostaje nepromenjen, a prethodni sadržaj *A* nepovratno je izgubljen. Izvršenje instrukcija prenosa podataka ne menja i ne utiče na sadržaj indikatorskog registra *PSW*.

Sledeći niz instrukcija zamenjuje sadržaje memorijskih lokacija sa adresama *51H* i *52H*:

```
MOV A, 51H
MOV B, 52H
MOV 51H, B
MOV 52H, A
```

Aritmetičke instrukcije

Instrukcija sabiranja, simboličko ime *ADD*, ima sledeći opšti oblik:

$$ADD \text{ A, sabirak} \quad ; A \leftarrow A + \text{sabirak}$$

U toku izvršenja ove instrukcije na prethodni sadržaj registra *A* dodaje se vrednost *sabirka* i suma smešta u registar *A* tako da se prethodni sadržaj registra *A* nepovratno gubi. Instrukcija *ADDC* obavlja operaciju koja je istao kao i *ADD* ali na sumu dodaje vrednost bita za prenos *C*. Instrukcija *ADDC* koristi se za sabiranje brojeva koji imaju više od 8 bita i moraju se smestiti u više od jednog bajta.

Na primer, neka je 24-bitni broj *X* smešten u memorijskim lokacijama sa adresama *41H*, *42H* i *43H*, tako da su značajniji biti smešteni u lokaciji *41H*, a najmanje značajni biti u lokaciji *43H*. Neka je na sličan način broj *Y* smešten u lokacijama *51H*, *52H* i *53H*. Sledeći niz instrukcija sabira brojeve *X* i *Y* i sumu smešta u memorijske lokacije *61H*, *62H* i *63H*,

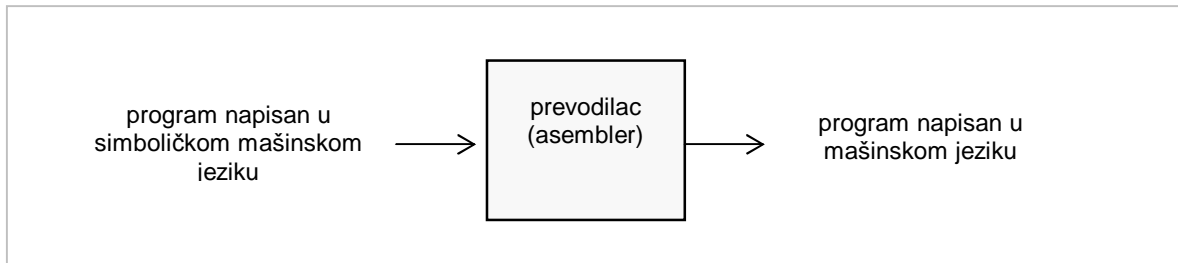
```
MOV A, 43H
ADD A, 53H
MOV 63H, A
```

```
MOV  A, 42H
ADDC A, 52H
MOV  62H, A
```

```
MOV  A, 41H
ADDC A, 51H
MOV  61H, A
```

Asembler

Program napisan u simboličkom mašinskom jeziku sastoji se iz instrukcija i direktiva. Prevodilac koji se naziva asembler prevodi program iz simboličkog mašinskog jezika u mašinski jezik, kao što je ilustrovano na sledećoj slici.



Naredbe u simboličkom mašinskom jeziku mogu biti instrukcije i direktive. Asembler prevodi instrukcije u mašinski jezik tako što jednu instrukciju u simboličkom mašinskom jeziku prevodi u jednu instrukciju u mašinskom jeziku. Direktive se koriste za formatizovanje izvornih programa (programa napisanih u simboličkom mašinskom jeziku) i upravljanje procesom prevođenja i ne prevode se u mašinski jezik.

Naziv 'assembler' često se koristi ne samo za prevodilac, nego i kao sinonim za simbolički mašinski jezik.

Instrukcija u simboličkom mašinskom jeziku (assembleru) u opštem slučaju sastoji se iz sledećih polja:

<labela> operacija <operand1>, <operand2>, <operand3> <;> <komentar>

Labela se koristi za pridruživanje simboličkog imena adresi memorijske lokacije u koju je smeštena instrukcija. Korišćenje labela olakšava programiranje i smanjuje greške, jer programer kod instrukcije programskih skokova ne mora da vodi računa o efektivnim adresama instrukcija, već navodi njihove labele.

Instrukcija može da ima ni jedan, jedan ili dva operanda. Naredba može da se završi komentarom koji je koristan kod dokumentovanja izvornog programa.

Direktive se koriste za formatizovanje izvornih programa i upravljanje procesom prevođenja. Na primer direktiva *END* označava kraj izvornog programa. Direktiva *ORG* postavlja interni brojač adresa memorijskih lokacija na zadatu vrednost. U postupku prevođenja izvornog programa asembler koristi interni brojač adresa memorijskih lokacija za određivanje efektivnih memorijskih adresa u koje smešta prevedene mašinske instrukcije.

Direktive *DS* i *DB* koriste se za rezervisanje memorijskog prostora, na primer za rezervisanje prostora za promenljive. Direktiva *EQU* dodeljuje labeli numeričku vrednost.

Direktive *CSEG*, *DSEG*, *ISEG*, *BSEG* i *XSEG* označavaju memorijski segment (memoriju) za program, interni segment (memoriju) za podatke i promenljive, gornji deo interne memorije za podatke i promenljive, segment za pristup pojedinim bitima i eksternu memoriju, respektivno.

Primeri jednostavnih programa

Primer: Napisati potprogram *suma* koji računa sumu pozitivnih celih 8-bitnih brojeva koji se nalaze u elementima tabele smeštene u memoriji.

Neka je tabela smeštena u memoriji počev od memorijske lokacije sa labelom *tab* i neka je broj elemenata tabele smešten u lokaciji sa labelom *brojel*. Za prenos parametara u potprogram koristićemo registre *B* (za prenos ukupnog broja elemenata koje treba sabrati) i *DPTR* (za prenos početne adrese tabele). Izračunata suma vraća se u registrima *R2* i *R5*, a bit sa adresom *00* koristiće se za javljanje greške.

Primer poziva potprograma:

```
greska BIT    00H          ; rezervisanje bita 0 za javljanje greške, dodela labele 'greska'
                                ; ovom bitu
CLR    greska          ; brisanje bita za grešku
MOV    B, #brojel      ; stavljanje broja elemenata u registar B
MOV    DPTR, #tab      ; pocetna adresa tabele u DPTR
LCALL  suma            ; poziv potprograma
```

Potprogram napisan u assembleru ima sledeci izgled.

```
suma  CJNE  B, #0, jedan   ; proverava da li je broj elemenata jednak 0
      SETB  greska        ; greska ako je broj elemenata jednak 0
      AJMP  kraj          ; kraj ako je greska

jedan  CJNE  B, #1, sabir  ; proverava da li tabela ima samo 1 element
      MOVX  A, @DPTR      ; suma je jednaka prvom elementu
      AJMP  kraj          ; zavrsetak

sabir  MOV   R4, #0        ; suma je u registrima R4 i R5
      MOV   R5,           ; suma je u pocetku 0

opet  MOVX  A, @DPTR      ; A = tab [i]
      INC   DPTR          ; DPTR pokazuje na sledeci element u tabeli
      ADD   A, R5          ; racunanje sume
      MOV   R5, A
      MOV   A, R4
      ADDC  A, #0          ; sabiranje prenosa
      MOV   R4, A
      DJNZ  B, opet       ; da li su sabrani svi elementi?
```

kraj RET ; povratak u glavni program, suma je u R4 i R5

Literatura

H.W. Huang, Using the MCS-51 Microcontroller, Oxford University Press, 2000.