Elektronski fakultet, Niš
Katedra za računarstvo

Arhitektura i organizacija računarskih sistema I

FOND ČASOVA 2+2+1
prijviti se na kurs na CSu

Prof. dr Ivan Milentijević

LITERATURA

1. **Sopstvena sveska sa predavanja**
2. **Sopstvena sveska sa vežbi**
3. **William Stallings, Computer Organization and Architecture, 10th edition, Pearson Education, 2016**
4. **RISC, CISC i DSP procesori, Mile Stojčev, El. fakultet u Nišu, 1997. g.**
5. **The Elements of Computing Systems: Building a Modern Computer from First Principles, Noam Nisan, Shimon Schocken, The MIT Press, 2005**

SINTEZA HIPOTETIČKOG RAČUNARA

- Cilj je da se korak po korak projektuje jedna hipotetička mašina u kojoj će postojanje svakog gradivnog elementa biti opravdano (princip zasnovan na sintezi)

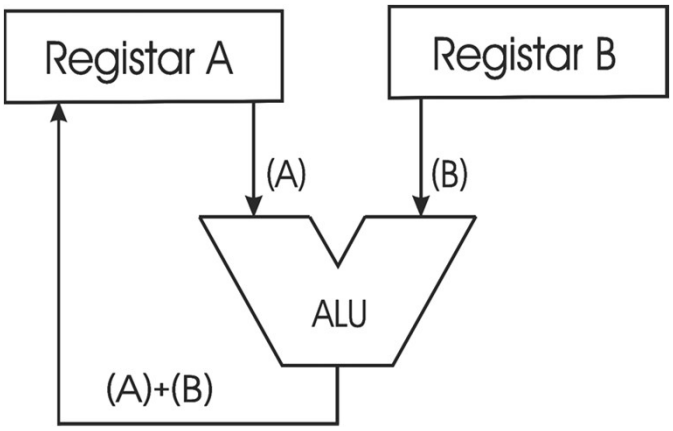
- Neka ta mašina ima ograničen skup od 16 naredbi

	operacija	kod operacije	komentar
	NOP	0000	**op. bez efekta**
aritmetičke naredbe	ADD	0001	**sabiranje**
	ADC	0010	**sabiranje sa pren.**
	SUB	0011	**oduzimanje**
	SBB	0100	**oduzimanje sa pozajm.**
logičke naredbe	OR	0101	**ILI operacija**
	AND	0110	**I operacija**
	NOT	0111	**negacija**
	XOR	1000	**isključivo ILI**
U/ I	IN	1001	**operacija ULAZ**
	OUT	1010	**operacija IZLAZ**

	operacija	kod operacije	komentar
za transfer	MOV	1011	**kopiranje**
	MVI	1100	**neposredno kopiranje**
	HLT	1101	**zaustavljanje**
naredbe grananja	Jxx	1110	**uslovno grananje**
	JMP	1111	**bezuslovno grananje**

aritmetičke i logičke operacije

- Za realizaciju aritmetičkih i logičkih operacija potrebna su dva binarna broja, dva registra (u kojima će brojevi biti zapamćeni) i aritmetičko-logička jedinica (ALU) opšte namene.



Slika 1. Izvršenje operacije
sabiranje sadržaja 2 registra

napomena:

sa (A) se označava sadržaj registra A

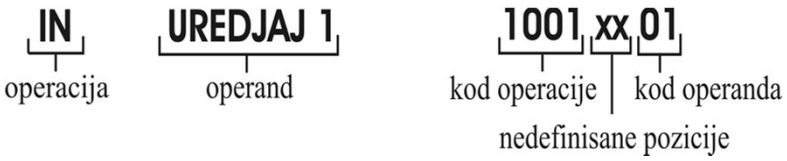
na šemi nedostaje:

- način upisa podataka u registre
- tip operacije koju će obaviti ALU
- upravljački signali pomoću kojih se

vrši vremenska sinhronizacija rada sistema

operacija ULAZ

- Podatak sa kojim treba da se radi mora prethodno da se unese u sistem i to je razlog da se uvede operacija ULAZ.

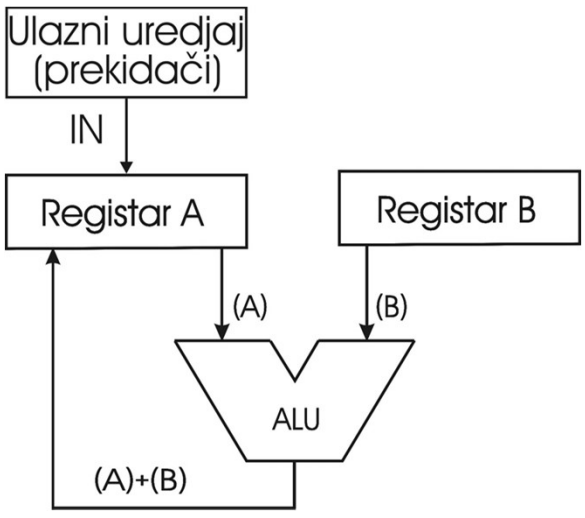


Slika 2. Format mašinske naredbe IN

- **napomena:**

  - format mašinske naredbe IN je 8-bitni

  - polje bitova “kod operanda“ definiše adresu ulaznog uredjaja
  - xx nedefinisano za sada (odredišni operand)

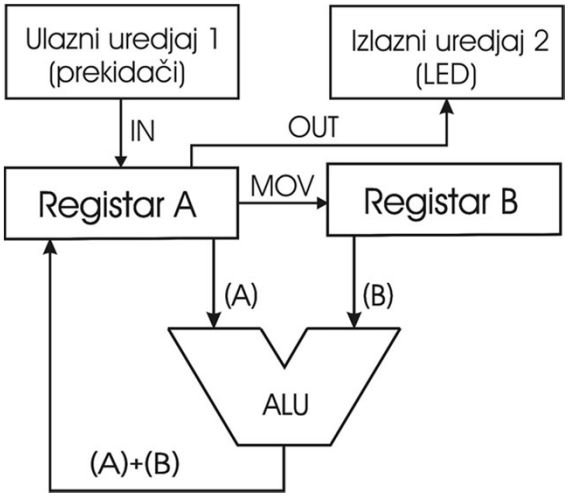


Slika 3. Hardverska struktura pomoću koje se obavlja operacija IN

- **operacija MOV**

Za operaciju sabiranja neophodno je :

- uneti 2 broja u registre A i B
- obezbediti mehanizam za prenos jednog broja u registar B (kopiranje)
- obaviti operaciju sabiranja



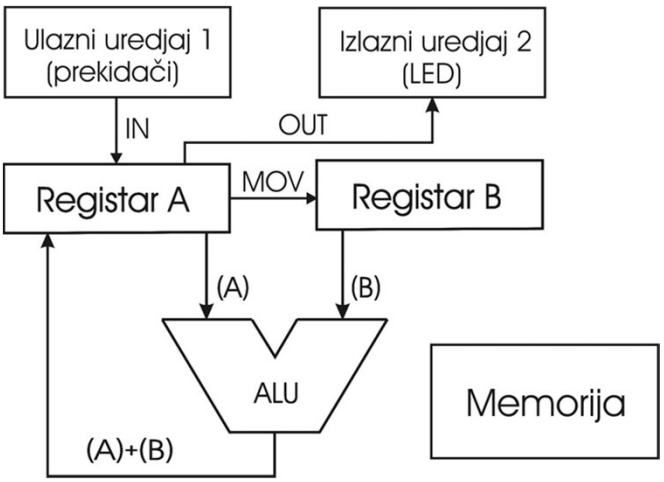
Slika 4. hardver dovoljno dobro definisan za unošenje brojeva u sistem, njihovo sabiranje i generisanje rezultata

```
IN   1
MOV B,A
IN 1
ADD A,B
OUT 2
```

- svaki U/I uredjaj ima svoj broj koji specificira njegovu adresu
- ulaznom uredjaju je dodeljena adresa 1, izlaznom uredjaju adresa 2.

- **MEMORIJA**

Program mašine mora biti smešten u memoriju. Svaka memorijska lokacija ima svoju adresu pomoću koje se specificira ta lokacija. Prvoj memorijskoj lokaciji dodeljena je adresa “0”.



Slika 5. Pozicija memorije

Slika 5.

- Da bi se program izvršio potrebno je da sadržaj memorijskih lokacija bude definisan na sledeći način

memorijska lokacija (adresa)	sadržaj mem.lokacije	naredba
0:	10010001	IN 1
1:	10110100	MOV B,A
2:	10010001	IN 1
3:	00010001	ADD A,B
4:	10101000	OUT 2

a) uzeto je da je svaka instrukcija po 8 bitova i da je memorija bajtovski orijentisana

b) internim registrima A i B dodeljeni su kodovi 00 i 01

c) izvorišni operand A i odredišni B

MOV B, A ; (B)←(A)

1011	01	00
kod oper. mov	odredišni operand A	izvorišni operand B

d) IN 1

1001	00	01
kod operacije IN	odredišni op. A	izvorišni op. (ulazni uredjaj 1)

e) OUT 2

1010	10	00
kod op. OUT	odr.operand (izlazni ur.2)	izvorišni op. reg. A

f) nepotrebno IN A,1 već samo IN 1

g) nepotrebno OUT 2,A već samo OUT 2

h) ADD A,B ; (A)←(A) + (B)

u prvoj fazi izvršenja naredbe sabiranja

registri A i B su izvorišni, a nakon dobijanja rezultata A je odredišni

0001	00	01
kod operacije ADD	odredišni op. u A	izv.operand u B

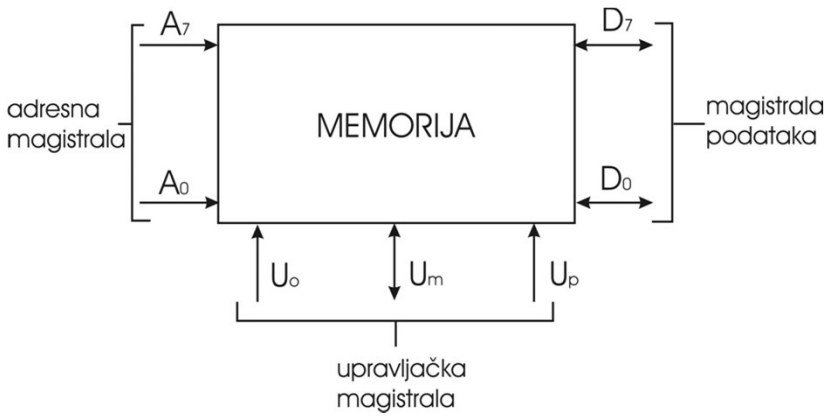
Rad memorije

Da bi se objasnio način rada memorije uvode se sledeći pojmovi :

- - adresna magistrala
- - magistrala podataka
- - upravljačka magistrala

Magistralu čini skup linija po kojima se prenosi informacija.

Uloga i tip informacije koja se prenosi po odredjenoj liniji na magistrali je unapred definisana.



Slika 6. Način povezivanja memorije

- prenos signala po adresnoj magistrali vrši se u jednom smeru (procesor→memorija), a broj adresnih linija odredjuje broj memorijskih lokacija koje se mogu selektovati (npr. 8 linija A0, A1, ... , A7 mogu da adresiraju 2^8= 256)

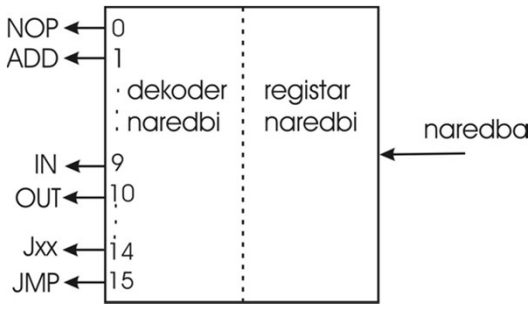
- magistrala podataka je dvosmerna

- nakon adresiranja smer na magistrali podataka odredjen je stanjem signala na upravljačkoj magistrali (sig. RD,WR)

Registar naredbi i dekoder naredbi

- Uzmimo da postoji mehanizam pomoću koga se naredba iz memorije prenosi u jedan registar. Taj registar je **registar naredbi**.
- Njemu se pridružuje **dekoder naredbi**. Oni zajedno čine osnovni deo centralne procesorske jedinice

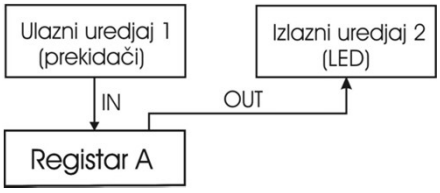
  CPU- a (Central Processing Unit).



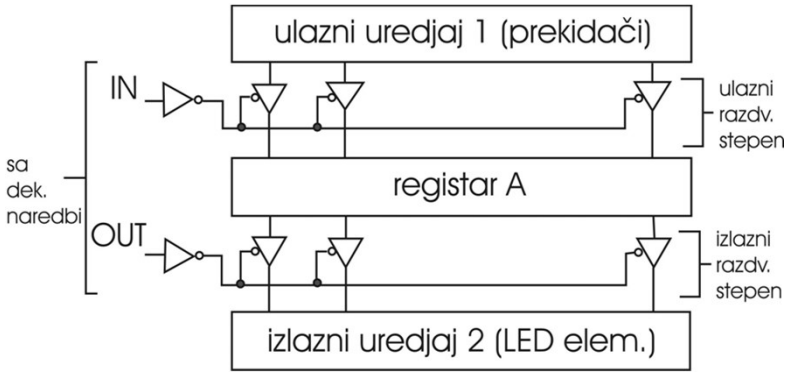
Slika 7. Registar naredbi i dekoder naredbi

Uloga registra naredbi je da prihvati naredbu, a dekoder naredbi u zavisnosti od sadržaja na njegovim ulazima aktivira odgovarajući izlaz.

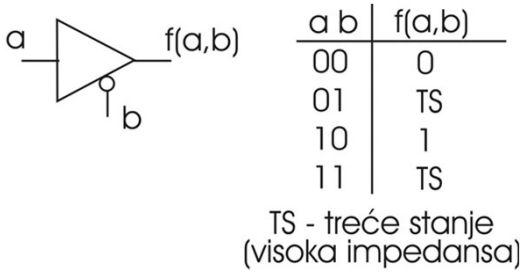
Svaki dekodirani izlaz iz dekodera naredbi odgovara jednoj od 16 definisanih naredbi.



Slika 8.U/I kanali



Slika 9.Efekat dekodiranih izlaza IN i OUT

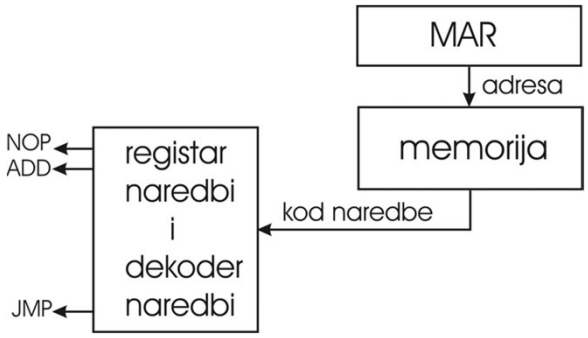


Slika 10. Three state baferi

Kada signali IN i OUT nisu aktivni, izlazi ulaznog i izlaznog razdvojnog stepena su u stanju visoke impedanse (“otkačeni”).

memorijsko- adresni registar

Memorijsko- adresni registar (MAR) ima zadatak da čuva memorijsku adresu u trenutku kada se adresira memorijska lokacija u kojoj je smeštena naredba koju je potrebno pribaviti.

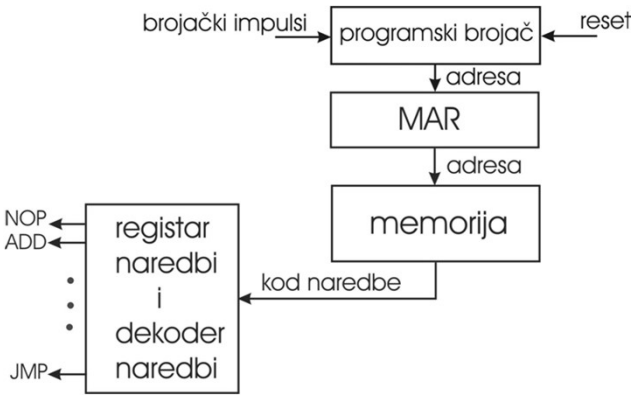


Slika 11. Mesto i uloga memorijsko-adresnog registra

programski brojač

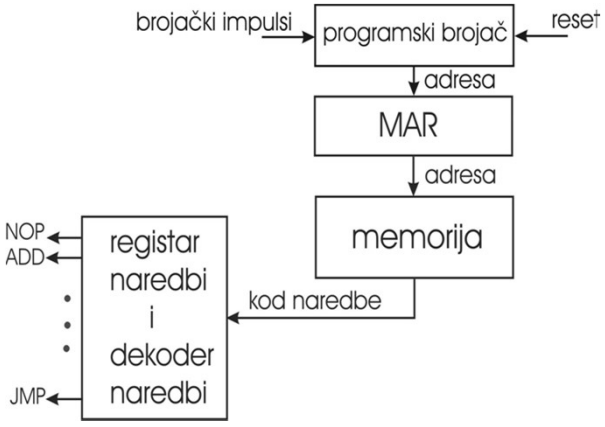
Da bi se neki program izvršio potrebno je memoriji slati sekvencu adresa (u našem primeru od “0” do “4”). Ova aktivnost je slična brojanju, pa je zbog toga neophodno uključiti u sistem blok koji se zove **programski brojač** (program counter – PC).

“0” : IN 1
“1” : MOV B,A
“2” : IN 1
“3” : ADD A,B
“4” : OUT 2



Slika 12. Uvodjenje programskog brojača

Programskom brojaču se moraju dovesti brojački impulsi i reset impuls kojim je omogućeno startovanje programa od početka , od memorijske lokacije sa adresom “0”.

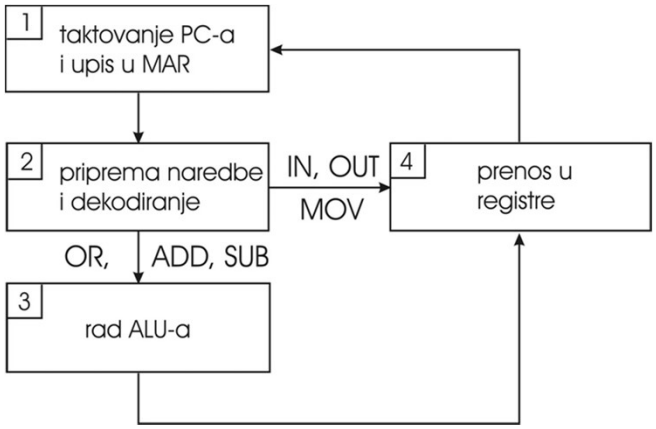


Slika 12. Uvodjenje programskog brojača

Programskom brojaču se moraju dovesti brojački impulsi i reset impuls kojim je omogućeno startovanje programa od početka , od memorijske lokacije sa adresom “0”.

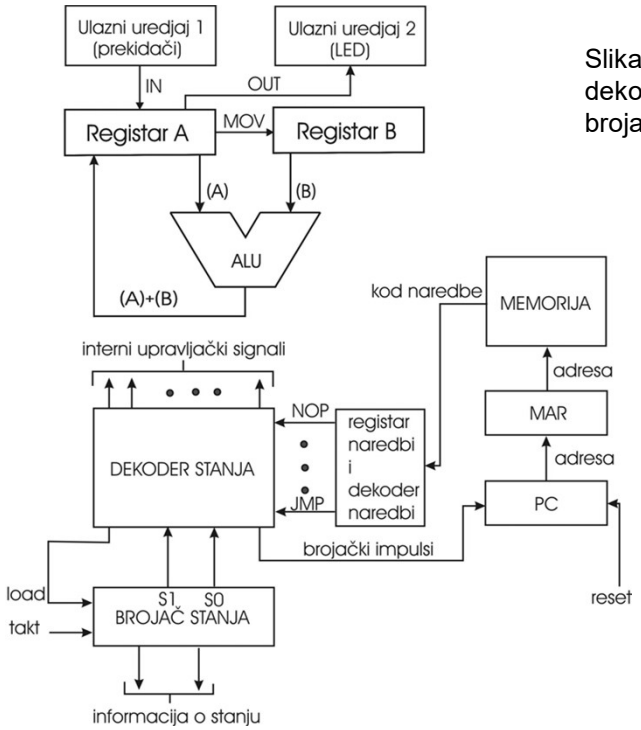
Sinhronizacija sistema i sekvencijalna stanja mašine

- Impulsi za inkrementiranje programskog brojača moraju se dovoditi u vremenskim intervalima koji su u saglasnosti sa trajanjem mašinskih naredbi.
- Da bi se pojednostavilo projektovanje hardvera moguće je uzeti da vreme izvršenja naredbi bude uniformno, ali to onda mora biti vreme potrebno za izvršenje najsporije naredbe. (loše rešenje)
- Uvodi se mašina sa sekvencijalnim stanjima.
- Sledi analiza repetitivnog sekvencijalnog rada mašine za vreme izvršenja svake naredbe.



Slika 13. Dijagram prelaza sekvencijalnih stanja

Da bi se realizovao sekvencijalni rad mašine neophodno je uvesti hardver koji čine **brojač stanja** i **dekoder stanja**.



Slika 14. Mesto i uloga dekodera stanja i brojača stanja

Dekoder stanja na svom izlazu generiše veliki broj upravljačkih signala koji se u okviru mašinske naredbe javljaju u strogo definisanim vremenskim intervalima. Pomoću njih se upravlja funkcijom pojedinih blokova sistema kao što su:

- tip operacije koju obavlja ALU (add, sub...)
- upravlja se čitanjem ili upisom podataka u registre (A, B, MAR)
- upravlja se čitanjem ili upisom podataka u memoriju itd.

Brojač stanja ukazuje u kom se stanju nalazi mašina.U našem slučaju realizovan je kao dvostepeni brojač. Izlazi brojača se vode na ulaz dekodera.

U toku izvršenja naredbi tipa ADD,SUB (odnosno onih koje zahtevaju rad ALU) mašina sekvencijalno prolazi kroz sva 4 stanja.

Kod instrukcija koje kraće traju (mov,nop...), odnosno kod onih koje ne zahtevaju rad ALU, dekoder stanja aktivira signal “ load” koji nalaže da mašina iz stanja 2 predje direktno u stanje 4.

taktna pobuda

Sada već možemo odrediti vreme izvršenja svake naredbe. Ako usvojimo da je pobudna taktna frekvencija sistema 1 MHz, tada možemo izračunati vreme izvršenja našeg programa:

- f =1 MHz **=>** T=1/f = 1μs
-
naredba	stanja
IN 1	3
MOV B, A	3
IN 1	3
ADD A, B	4
OUT 2	3

	∑ = 16
- vreme izvršenja programa = 16 μs.