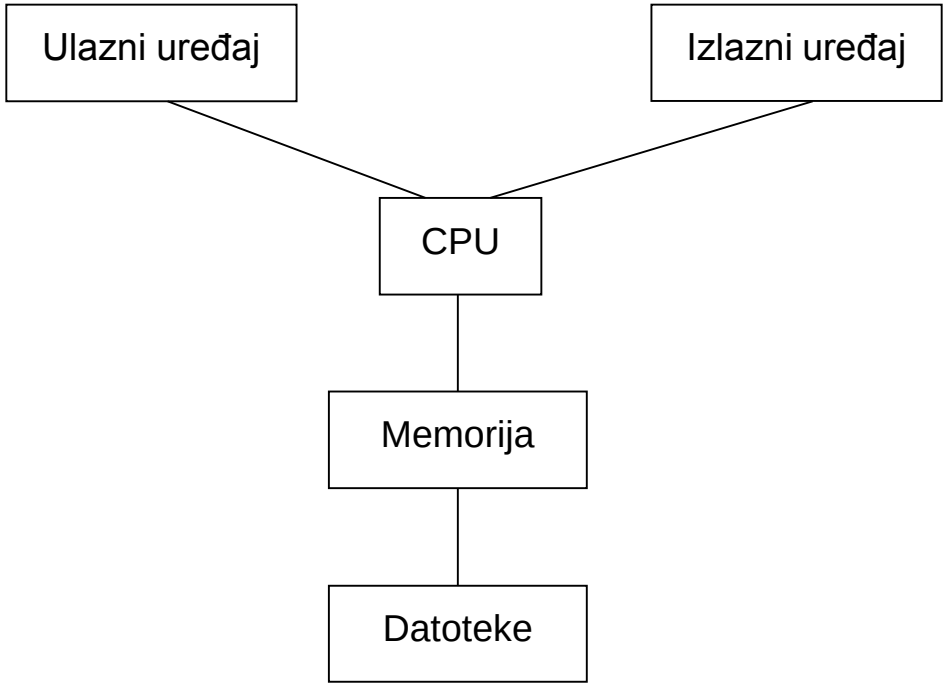
Računarski sistemi

Uvod u mašinski jezik

Gimnazija, 2016

Mašinski jezik

Osnovna struktura



- CPU obavlja kontrolnu funkciju rada računara neophodnu za korektno funkcionisanje čitavog računara. Njome se obezbeđuje da podaci u svakom trenutku budu raspoloživi onom delu računara kojem su potrebni u toku obrade. Ako to nije slučaj tokom celog vremena obrade, rezultat obrade će biti pogrešan. Pored ove kontrolne funkcije, CPU omogućava aritmetičke i logičke operacije nad podacima.

Mašinski jezik

Osnovna struktura

- **Memorija** računara se koristi u dve različite svrhe: prva je da **drži podatke** nad kojima se vrši obrada, a druga da **drži niz instrukcija** (operacija) koje ce se izvršavati nad podacima. Ovaj niz instrukcija se zove **algoritam**. Za algoritam kažemo da se obavlja nad nekom strukturom podataka, gde se pod strukturom podrazumeva forma podataka ali ne i njen trenutni sadržaj.
- Kombinacija algoritma i strukture podataka naziva se program.
- Memorija računara je izdeljena na istovetne **ćelije** ili lokacije tako da svaka ćelija (lokacija) ima svoju posebnu (jedinstvenu) adresu.
- Memorijske ćelije mogu da sadrže ili podatke ili programske instrukcije.

Adrese	Lokacije (ćelije)
0	
1	
2	
N-2	
N-1	
N	

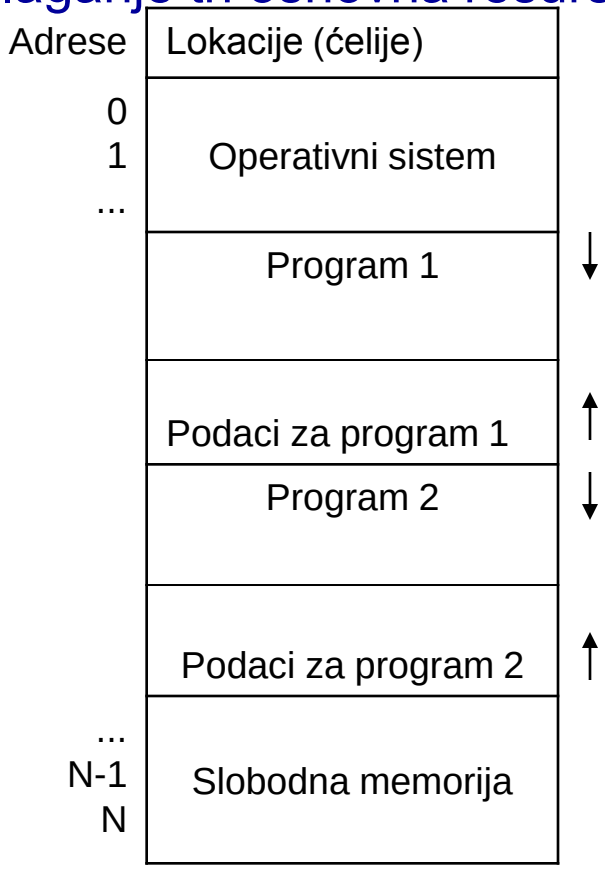
Mašinski jezik

Osnovna struktura

- **Adresibilnost ćelija** omogućava CPU da preko adrese pristupi podatku koji ćelija sadrži.
- Na sličan način CPU pristupa i ćelijama koje sadrže programske operacije. Programske operacije su obično memorisane u uzastopnim ćelijama, jedna za drugom. CPU je u mogućnosti da pristupa ćelijama u proizvoljnom redosledu pristupa. Tokom izvršavanja algoritma operacije sadržane u memorijskim ćelijama mogu se izvršavati jednom ili više puta. Tako CPU tokom procesa obrade može pristupati ćelijama koje sadrže podatke i ćelijama koje sadrže algoritamske operacije u više navrata - sa repeticijom (ponavljanjem).
- Memorija sama po sebi ne može praviti razliku sadržaja svojih lokacija. Drugim rečima, memorija "ne zna", da li se u nekoj njenoj ćeliji nalazi podatak ili instrukcija. Zato je obaveza korisnika računara da raspodelu memorije na deo za podatke i instrukcije izvrši tako da se pravilno odvija proces obrade podataka.

Operativni sistem

- Operativni sistem je specijalan program stalno prisutan u memoriji računara, kojom se kontroliše rad čitavog računarskog sistema.
- Računarski sistem korisniku stavlja na raspolaganje tri osnovna resursa:
  - procesorko vreme,
  - memoriju za smeštaj podataka i algoritama,
  - kao i periferne uređaje.



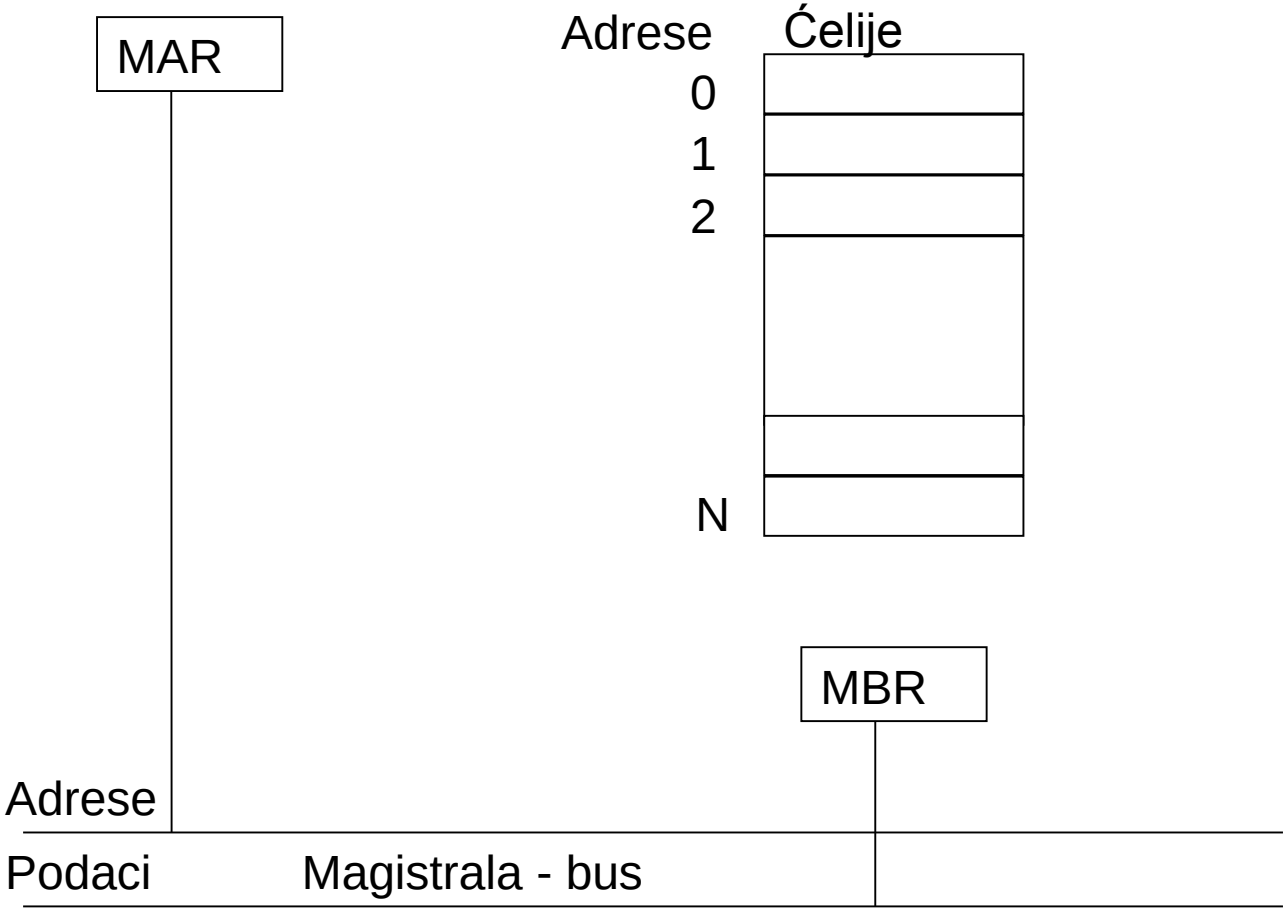
Mašinski jezik

Memorija

- Svaka memorija je izgrađena kao skup memorijskih ćelija, gde svaka ćelija ima svoj naziv to jest adresu. Uobičajeno je da se adrese označavaju celim pozitivnim brojevima. Podatak koji je smešten u memorijsku ćeliju naziva se sadržajem ćelije. Informacije koje predstavljaju sadržaj ćelija se mogu grubo podeliti u dve grupe: instrukcije (naredbe) algoritma obrade i podatke nad kojima se algoritam izvršava.
- CPU računara komunicira sa memorijom uz pomoć dva registra: **memorijskog adresnog registra (MAR) i memorijskog bafer registra (MBR).**
- Da bi CPU obavio komunikaciju sa memorijom u MAR se smešta adresa memorijske ćelije. Adresa se zatim dekodira elektronskim dekoderom koji aktivira memorijsku ćeliju. Pošto se na ovaj način može aktivirati bilo koja od ćelija (zavisno od adrese u MAR-u) u slučajnom (random) redosledu, ovakve memorije se obično nazivaju RAM (Random access memory). Memorijska ćelija može biti procesirana na dva načina.

Mašinski jezik

Memorija



Mašinski jezik

Memorija

- Ćelija se može čitati, to jest sadržaj ćelije se putem memorijske magistrale (bus-a) dovodi u MBR. U memorijsku ćeliju se može upisivati, to jest sadržaj MBR-a se prenosi magistralom do ćelije. Prilikom čitanja sadržaj ćelije se ne menja, već se samo "kopira" iz ćelije u MBR. Međutim, prilikom pisanja sadržaj ćelije se uništava i zamenjuje sadržajem MBR-a. Na taj način CPU i memorija komuniciraju pomoću registara MAR i MBR i magistrale koja povezuje CPU i memoriju.
- Veličina memorijske ćelije može da varira od računara do računara, ali su obično sve ćelije iste veličine. Veličina ćelije obično nije manja od osam binarnih cifara - bitova.
- Bajt je memorijska jedinica od 8 bitova i u nju može biti smešteno jedno slovo ili ceo broj u opsegu 0-255. Tako mala memorijska jedinica nije dovoljna za smeštanje većih brojeva, pa se često veličina memorije meri ne bajtovma već rečima. Memorijska reč je obično veličine 2 ili više bajtova. Veličina memorijske ćelije je u korespondenciji sa veličinom MBR-a, tako da se čitav sadržaj ćelije prenosi u MBR ili obrnuto u jednom prenosu magistralom.

Mašinski jezik

Registri

- Pod registrima se podrazumijevaju posebne memorijske ćelije koje nisu deo glavne memorije i kojima se ne pristupa putem MAR i MBR.
- Pristup registrima se obavlja direktno, pa je i brzina obavljanja operacija sa registrima mnogo veća nego sa standardnim memorijskim ćelijama.
- Obično, registara ima manji broj i imaju specifičnu namenu u računaru, a najčešće su deo CPU-a. Na primer, kada je potrebno sabrati dva broja koja su smeštena u dve ćelije glavne memorije, zbog efikasnosti se najčešće brojevi dovode u interne registre procesora, a rezultat sabiranja smešta ponovo u neki registar pre nego što se pošalje u memorijsku ćeliju. Veličina registra zavisi od njegove namene. Tako se u računarima mogu naći i registri od samo jednog bita poznati i kao flegovi (flag) ili flip-flopovi. U takvim registrima se čuvaju stanja dela računara, pa se testiranjem takvih registara utvrđuje stanje dela računara za koji je registar zadužen. Obično se ovi jednobitni registri stanja grupišu u jedan veći registar sa više bitova u kojem svaki bit prati određeni deo računara.

Mašinski jezik

Mašinski kod

- Glavna memorija sadrži i programske instrukcije i podatke. Većina instrukcija odnosi se na operacije nad podacima koji se nalaze ili u glavnoj memoriji ili u registrima CPU-a. Podaci na koje se instrukcije odnose nazivaju se **operandima**.
- Svaki program se sastoji od niza funkcionalno nezavisnih koraka - instrukcija. Instrukcije obavljaju različite funkcije koje mogu biti grupisane na sledeći način:
  - Transfer podataka između glavne memorije i CPU registara
  - Aritmetičke i logičke operacije sa podacima
  - Upravljanje tokom izvršavanja programa
  - Ulazno/izlazni (I/O) transfer podataka

Mašinski kod

- Sa programerskog aspekta najjednostavniji oblik instrukcije za sabiranje bio bi C:=A+B, gde su A,B,C imena varijabli. Pretpostavimo da su vrednosti ovih varijabli smeštene u memorijske lokacije čije adrese su označene sa A,B,C.

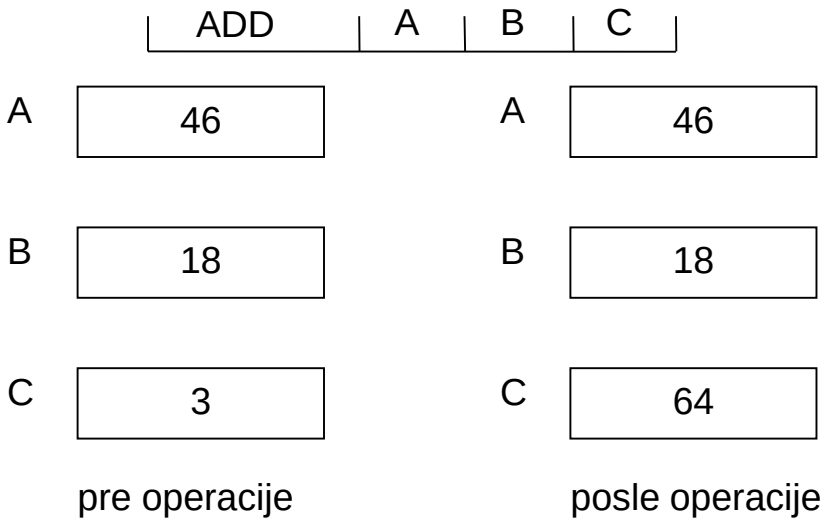
- Izraz C:=A+B ima sledece značenje:

  Vrednosti podataka iz lokacija A i B treba da budu doveden u CPU gde će uz pomoc aritmetičko logičke jedinice (ALU) biti sabrane, a rezultat sabiranja biće smešten u lokaciju C.

Mašinski jezik

Mašinski kod

- Ako ceo gornji izraz treba da bude predstavljen jednom mašinskom instrukcijom, mašinska instrukcija mora imati tri adresna dela za operande A,B,C.
- Tro-adresna instrukcija ADD A,B,C će zahevati veliki broj bitova za smeštanje sve tri adrese. Slika 6.1 prikazuje sadržaj adresa A,B,C pre i posle operacije ADD.



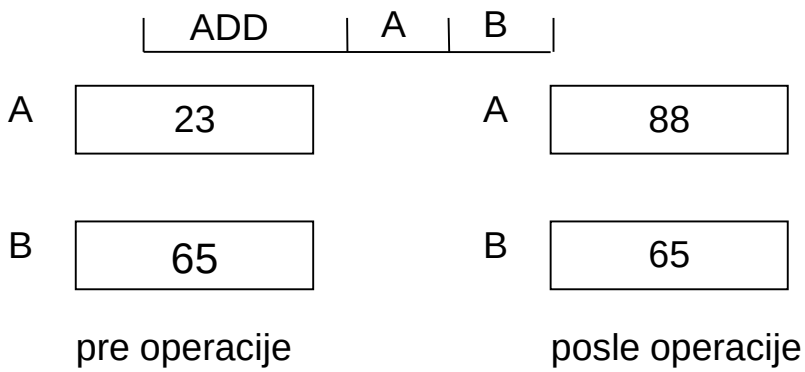
Mašinski jezik

Mašinski kod

- U cilju uštede na dužini instrukcije koristiti se dvo-adresna instrukcija sa dva adresna dela (polja): ADD A,B sa sledećim značenjem: Vrednosti podataka iz A i B treba dovesti u CPU, sabrati ih i rezultat smestiti u lokaciju A.
- U dvo-adresnoj instrukciji originalna vrednost iz lokacije A je uništena operacijom sabiranja, što može predstavljati problem. Ako želimo da vrednost u lokaciji A bude sačuvana moramo za sabiranje koristiti dve instrukcije:
  - MOVE C,B
  - ADD C,A

gde instrukcija MOVE C,B ima:

Kopirati sadržaj lokacije B u lokaciju C.



Mašinski kod

- Predhodno sabiranje se može izvršiti i sa jedno-adresnom instrukcijom. U tom slučaju ćemo koristiti tri instrukcije:
  - LOAD A
  - ADD B
  - STORE C
- Instrukcija LOAD A ima značenje: Kopiraj podatak iz lokacije A u CPU registar, koji se zove akumulator. Instrukcija STORE C ima značenje: Kopiraj sadržaj akumulatora u memorijsku lokaciju C.

Mašinski jezik

Skup instrukcija

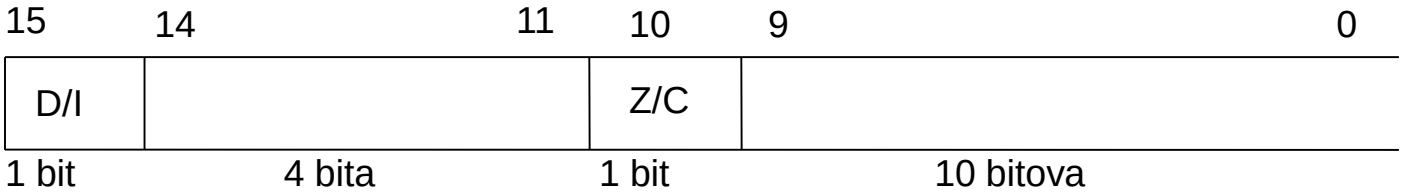
- Sve mašinske instrukcije koje neki računar može da izvršava nazivaju se skupom instrukcija tog računara. Sve instrukcije se mogu svrstati u jednu od sledeće tri kategorije:
  - Memorijske - sa operacijama nad podacima u glavnoj memoriji.
  - Ne-memorijske - sa operacijama bez podataka ili sa podacima u registrima CPU-a.
  - Ulazno/Izlazne - za operacije sa perifernim jedinicama.

Memorijski orjentisane instrukcije

- Ova klasa instrukcija obuhvata sve instrukcije koje se odnose bar na jednu memorijsku lokaciju, pa takva instrukcija mora sadržati adresno polje.
- Podklase memorijskih instrukacija koje se nalaze kod većine danas raspoloživih računara su:
  - Napuni neki od registara sa sadržajem neke memorijske lokacije.
  - Smesti sadržaj registra u neku memorijsku lokaciju.
  - Saberi sadržaj memorijske lokacije sa sadržajem registra.
  - Oduzmi sadržaj memorijske lokacije od sadržaja registra.
  - Uporedi sadržaj registra sa sadržajem memorijske lokacije i preskoči sledeću instrukciju ako nisu jednaki.
  - Povećaj sadržaj memorijske lokacije za 1 i preskoči sledeću instrukciju ako je rezultat 0.
  - Kombinuj sadržaj memorijske lokacije sa sadržajem registra korišćenjem logičkih operacija (AND, OR i sl.).
  - Operacije grananja - bezuslovni skok, ulosvni skok, i skok u potprogram.
  - Instrukcija pomjeranja bitova (shift) - instrukcije kojima se pomeraju svi bitovi u memorijskoj lokaciji ulevo ili udesno.

Memorijski orjentisane instrukcije

- Posmatrajmo jednoadresni racunar koji ima dva registra opšte namene (zvaćemo ih A i B registar), i neka računar ima memorijske lokacije dužine 16 bita. Tipična instrukcija je recimo "Napuni registar A" (napuni registar A sa sadržajem iz neke memorijske lokacije), ili "Dodaj na A" (dodaj sadržaj neke memorijske lokacije na sadržaj registra A).
- Kod jednoadresnih računara osnovni sadržaj instrukcije je kod operacije koju instrukcija vrši i adresa memorijske lokacije koja sadrži podatak potreban operaciji. Na slici za kod operacije je rezervisano 4 bita a za memorijsku adresu 10 bitova. Uloga bitova broj 10 i 15 biće razmotrena nešto kasnije.



Memorijski orjentisane instrukcije

- Ako je adresa memorijske lokacije data sa 10 bitova to znači da je broj memorijskih lokacija koje se mogu adresirati $2^{10} = 1024$. Današnji računari imaju znatno veću memoriju od 1024 16-bitne reči. Ako računar ima, recimo, 1 MB memorije za adresiranje je potrebno 20 bitova, a to s druge strane zahteva i da instrukcija bude znatno duža. Da bi se prevazišao ovaj problem (dužine instrukcije) koriste se razne metode adresiranja.

Grananje (Branching)

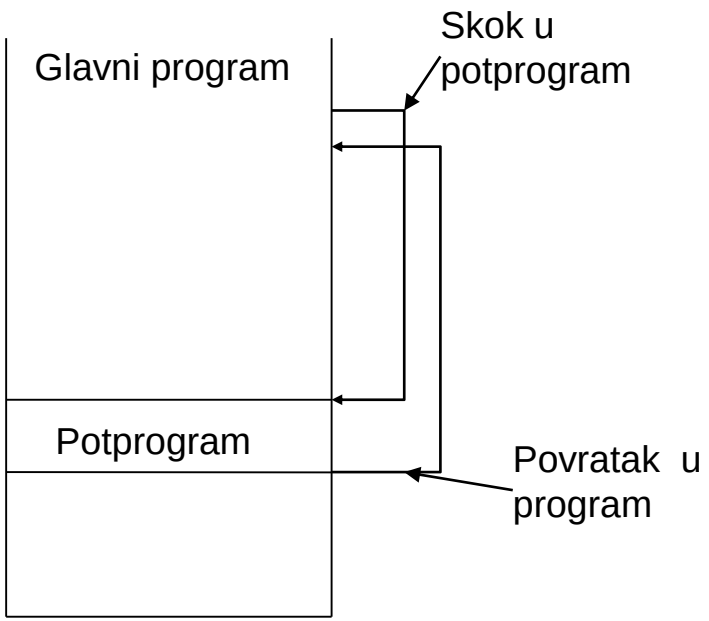
- Neke memorijski orijentisane instrukcije ne koriste podatke u memoriji, već služe kao "putokaz" za redosled izvršavanja instrukcija. Takve instrukcije nazivaju se **instrukcijama grananja programa** (branching).
- Instrukcije se u glavnoj memoriji nalaze jedna iza druge u nizu kako to zahteva algoritam. One se za vreme izvršavanja programa, po pravilu, i izvršavaju jedna za drugom u istom redosledu. Međutim ponekad se pojavljuje potreba da se u zavisnosti od rezultata pojedinih instrukcija menja "normalan" tok izvršavanja instrukcija.

```
          Napuni registar A sa vrednošću n
CIKLUS:   instrukcija 1
          instrukcija 2
                .
          Umanji registar A za 1.
          Skoči na CIKLUS ako je A > 0.
```

- Poslednjom instrukcijom smo zapravo postigli da se instrukcije iz ciklusa ponavljaju sve dok vrednost u registru A ne bude jednaka 0, a zatim se program nastavlja sledećom naredbom koja sledi iza naredbe "Skoči".

Potprogrami

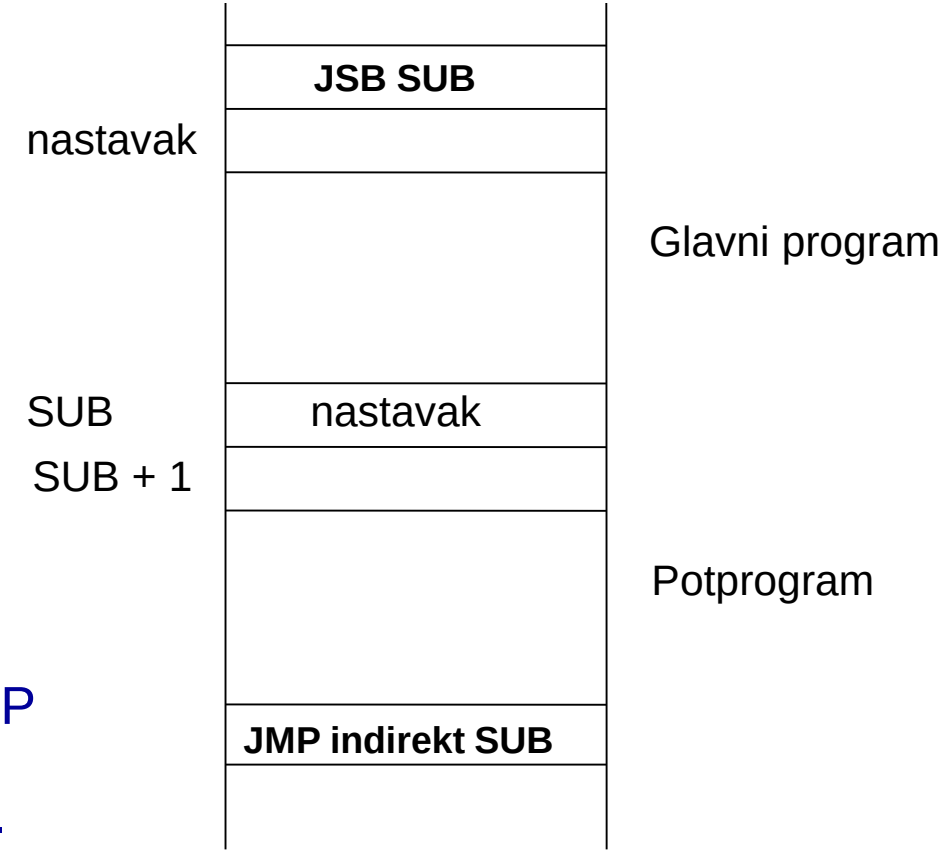
- Specijalan oblik instrukcije grananja je tzv. **skok u potprogram**.
- Pod potprogramom podrazumevamo niz instrukcija koje obavljaju neku opštu funkciju koja se tokom izvršavanja programa može zahtevati više puta i iz različitih delova programa. Posebnost skoka u potprogram se ogleda u tome što se pri grananju mora zapamtiti adresa mesta u programu odakle je "poziv" (skok) na potrpogram učinjen, kako bi se nakon završetka rada potprograma "vratili" na pravo mesto u programu.



Mašinski jezik

Potprogrami

- Kao primer instrukcija koje omogućavaju pozivanje i vraćanje iz potprograma uzećemo instrukcije JSB (jump to subroutine) i RETURN (povratak).
- Instrukcija JSB smešta u memorijsku lokaciju koja se nalazi na prvoj lokaciji potprograma povratnu adresu (adresu prve instrukcije iza instrukcije JSB). Poslednja instrukcija potrograma je instrukcija indirektog skoka (JMP indirect) kojom se vraćamo na sledeću instrukciju iz programa.



Ne-memorijske instrukcije

- Ne-memorijske instrukcije su one koje ne zahtevaju podatak iz memorijske lokacije. Tipičan slučaj su instrukcije koje koriste registre kao operande. Kako računari obično sadrže mali broj registara, za njihovo adresiranje je potrebno svega nekoliko bitova, pa su nememorijske instrukcije znatno kraće od memorijskih. Karakterističan skup ne-memorijskih instrukcija je:
  - Registar - registar aritemetičke i logičke funkcije kao što su ADD, SUBTRACT, MOVE, AND, OR, XOR (ekskluzivno OR).
  - Shift instrukcije kojima se sadržaj registara pomera ulevo ili udesno za jednu ili više pozicija.

Metode adresiranja memorije

- Zbog uštede u dužini instrukcija potrebno je imati više načina adresiranja memorije. Zato se uvodi pojam efektivne adrese koja je zapravo stvarna adresa u memoriji koja se dobija posle različitih transformacija pri raznim metodama adresiranja.

  - **Direktne (apsolutne) adrese**
    Efektivna adresa lokacije data je u samoj instrukciji. Broj bitova u adresnom polju ograničava memorijski prostor koji može biti adresiran na ovaj način. Ako je broj bitova u adresnom polju instrukcije N tada je maksimalna memorija 2^N lokacija.
  - **Indirektne adrese**
    Efektivna adresa se dobija tako što se u instrukciji nalazi adresa memorijske lokacije (ili registra) koja sadrži adresu operanda instrukcije. **Svrha bita 15** u formatu instrukcije je da označi način adresiranja (direktno ili Indirektno). Ako je bit 15 postavljen na nulu koristiće se direktno adresiranje, a ako je postavljen na 1 indirektno. Prednost indirektnog adresiranja je u tome što se čitava memorijska lokacija koristi za adresu (a ne njen deo kao u instrukciji), pa se samim tim može adresirati znatno veća memorija.

Metode adresiranja memorije

- **Neposredno adresiranje**
  Kod neposrednog adresiranja umesto adrese operanda u instrukciji se nalazi sama njegova vrednost. To je pogodno u slučaju fiksnih podataka (konstanti) koje se ne menjaju za vrijeme izvršavanja programa. Vrednost podatka je, naravno, limitirana brojem bitova unutar instrukcije.
- **Indeksno adresiranje**
  Efektivna adresa operanda se izračunava sabiranjem neke vrednosti (indeksa) sa adresom datom u instrukciji. Vrednost indeksa je obično smeštena u posebnom registru CPU tzv. **indeksnom registru**. Ovakva vrsta adresiranja je posebno pogodna u slučajevima kada se instrukcija ponavlja za više operanada koji se nalaze jedan za drugim u memoriji (recimo sabiranje niza brojeva).

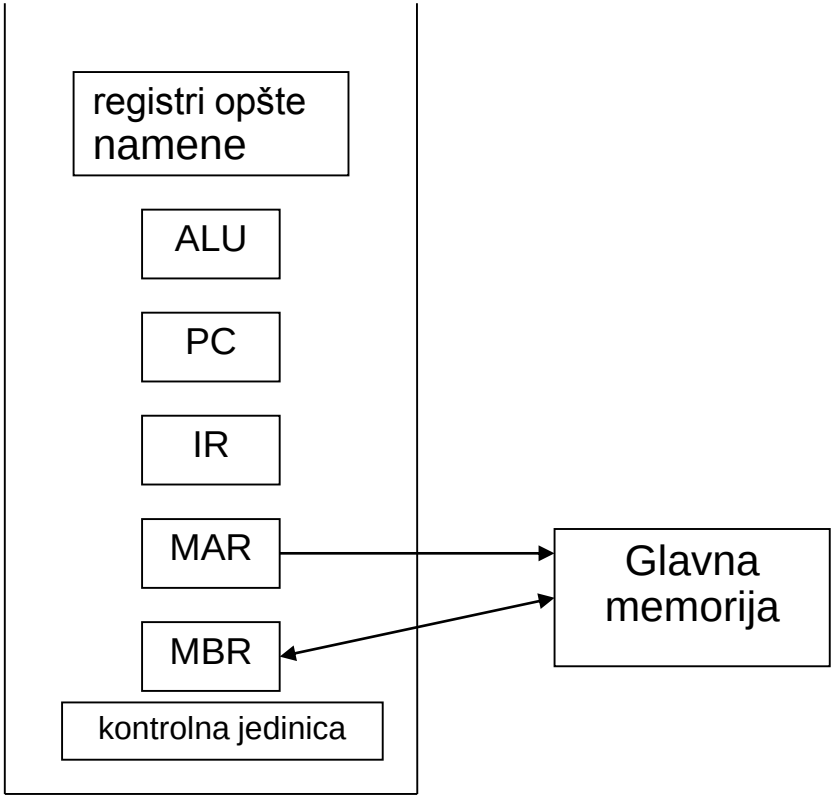
Mašinski jezik

- Za izvršavanje instrukcija, potrebno je da postoji u računaru poseban deo (jedinica), koji uzima instrukcije iz memorije, prepoznaje ih i obavlja akcije u skaldu sa značenjem instrukcije. Takva jedinica naziva se centralnom procesorskom jedinicom (CPU).

- Rešavanje nekog problema uz pomoć računara vrši se definisanjem algoritma. Algoritam se sastoji od niza koraka koji se nazivaju mašinskim instrukcijama. CPU vrši svoj posao na sličan način. Da bi uzeo iz memorije i izvršio instrukciju CPU mora, za svaku instrukciju, da obavi niz elementarnih operacija.

Komponente CPU-a

- CPU možemo posmatrati kao skup komponenti sa različitim funkcijama. One se mogu grupisati ovako:
  - **Specijalni registri** PC, IR, MAR, MBR, kao i skup registrara **opšte namene**.
  - Aritmetičko-logička jedinica **ALU**.
  - Kontrolna jedinica.
- **Specijalni registri** služe za prihvatanje i analizu instrukcija.
- **Registri opšte namene** stoje na raspolaganju programeru da u njih dovodi potrebne podatke.
- **Aritmetičko-logičku jedinicu** možemo posmatrati kao "crnu kutiju" u kojoj se zapravo obavljaju računske i logičke operacije.
- **Kontrolna jedinica** je "nervni centar" računara koja salje upravljačke signale svim ostalim jedinicama.



Mašinski jezik

Interni registri

- Svaki program je sačinjen od niza instrukcija smeštenih u glavnoj memoriji računara. Prilikom izvršavanja programa CPU uzima jednu po jednu instrukciju iz memorije i priprema akcije (operacije) potrebne da se instrukcija obavi. Instrukcije se uzimaju iz memorije redom iz uzastopnih lokacija, osim u slučaju instrukcija grananja.
- Sve komunikacije CPU-a i memorije odvijaju se pomoću dva registra: **memorijskog adresnog registra** (**MAR**) i **memorijskog baferskog registra (MBR)**.
  - Da bi uzeo instrukciju iz memorije CPU najpre smešta njenu adresu u MAR i izvršava operaciju čitanja memorije čiji rezultat je "kopiranje" sadržaja memorijske lokacije u MBR registar.
  - Izvršavanje instrukcije, koja je sada u MBR-u, može zahtevati podatak iz memorije, čija adresa mora biti smeštena u MAR što bi uništilo adresu instrukcije koja se izvršava. CPU bi tako izgubio orjentaciju i ne bi mogao da zna adresu sledeće instrukcije. Kao posledica ovog problema, CPU mora imati poseban registar za čuvanje adrese sledeće instrukcije. Taj registar je poznat kao **PC (program counter - programski brojač).** Na početku izvršavanja programa PC se napuni sa adresom prve instrukcije programa, a zatim nakon uzimanja svake instrukcije iz memorije adresa smeštena u PC-u se automatski uvećava da pokazuje adresu sledece instrukcije (sem u slučaju instrukcije grananja kada se adresa u PC-u modifikuje na drugi način).

Mašinski jezik

Interni registri

- Nakon uzimanja instrukcije iz memorije ona će se naći u MBR registru. Međutim, MBR registar može biti zahtevan za preuzimanje podatka iz memorije koji je potreban za izvršavanje instrukcije. Znači, pojavljuje se sličan problem kao i sa MAR registrom. Zato je potrebno da u CPU-u postoji poseban registar koji će držati instrukciju za vrijeme njenog izvršavanja. Takav registar naziva se **instrukcionim registrom (IR)**.
- Proces izvršavanja programa možemo prikazati kao sledeći niz događaja:
  - **(1) Kopiraj PC sadrzaj u MAR**
  - **(2) Učitaj u MBR memorijsku lokaciju sa adresom u MAR**
  - **(3) Kopiraj sadrzaj MBR-a u IR.**
  - **(4) Dekodiraj sadrzaj IR (prepoznaj instrukciju)**
  - **(5) Izvrši instrukciju**
  - **(6) Ponovi sve od koraka (1)**

Mašinski jezik

Interni registri

- Ovaj proces se moze prikazati simbolickom notacijom:

[PC] ---> MAR
[M] ---> MBR
[PC]+1 ---> PC
[MBR] ---> IR
Dekodiraj IR
Izvrši instrukciju

- Uglaste zagrade označavaju sadržaj memorije (M) ili registra.

- Operacija "Dekodiraj instrukciju" se obavlja pomoću dekodera, koji može biti izveden kao kombinatorno logičko kolo. Operacija "Izvrsi instrukciju" takođe se može sastojati od niza koraka.

Aritmetičko-logička jedinica

- Aritmetičko-logička jedinica (ALU) je deo CPU u kojem se izvršavaju sve aritmetičke i logičke operacije. Gradi se od vrlo brzih elektronskih komponenti i predstavlja složeno "parče" elektronike koje je sposobno da sa podacima koje joj se daju izvrši elementarne aritmetičke i logičke operacije. Tipične operacije koje se obavljaju u ALU-u su:
  - Aritmetičke operacije kao što su ADD (dodavanje) i COMPLEMENT (komplement binarnog broja).
  - Logičke operacije kao što su AND, OR, EXCLUSIVE OR.
  - Manipulacione operacije kao što su SHIFT, TEST.
- Skup aritmetičko logičkih operacija koje mogu da se obavljaju u ALU varira od računara do računara. ALU, najčešće, aritmetičke operacije obavlja samo sa celobrojnim vrednostima. Pa i tada, vrlo ograničen skup operacija se implementira, a složenije operacije se najčešće izvode programskim putem. Tako, recimo, operacije množenja i deljenja, najčesce, nisu realizovane hardverski u ALU-u, već se izvode softverski korišćenjem sabiranja i oduzimanja ili šift operacijama.

Aritmetičko-logička jedinica

- ALU sadrži i niz tzv. test bitova (flip-flop signala) koji signaliziraju rezultat aritmetičkih i logičkih oparacija. Ovi bitovi (flegovi - zastavice) se najčešće označavaju sa N, Z i O. Tako fleg N (Negative) kada je postavljen na 1, označava da je rezultat aritmetičke operacije koja je upravo izvršena u ALU negativan broj, Z (Zero) označava da je rezultat 0, a O označava pojavu rezultata veceg od maksimalnog broja sa kojim računar može da radi (Overflow - prekoračenje).

Floating point jedinica

- ALU najčešće obavlja aritmetičke operacije sa celim brojevima. Međutim brojne aplikacije zahtevaju i rad sa decimalnim brojevima (floating point - pokretni zarez). Naravno, sve operacije sa pokretnim zarezom mogu biti softverski simulirane, ali se pri tome značajno povećava vreme izvršavanja takvih operacija. Zato se, često i za ove operacije gradi posebna elektronska jedinica poznata kao **FPU (Floating point unit)**.

Izvršavanje instrukcija

- Program smešten u glavnoj memoriji računara, izvršava se u sledećim koracima:

  (1) [PC] ---> MAR
  (2) [M] ---> MBR
  (3) [PC]+1 ---> PC
  (4) [MBR] ---> IR
  (5) Dekodiraj IR
  (6) Izvrši instrukciju
  (7) Ponovi od koraka (1)

- Koraci **(1) do (5) se nazivaju fazom uzimanja (FETCH) instrukcije**, a **korak (6) fazom izvršavanja (EXECute)**. Svaki od ovih koraka se naziva mikro-instrukcijom.
- FETCH faza se izvršava uvek na isti način, bez obzira o kojoj je mašinskoj instrukciji reč. Međutim faza izvršavanja biće različita za različite mašinske instrukcije.

Izvršavanje ne-memorijskih instrukcija

- Nememorijske instrukcije su one instrukcije **za čije izvršavanje nisu potrebni podaci koji se nalaze u glavnoj memoriji**. To su najčešće instrukcije koje se izvršavaju sa podacima koji su već u CPU registrima. Za izvršavanje takvih instrukcija potrebno je samo ALU-u saopštiti koji bitovi i iz kojeg registra će biti korišćeni i signalom iz dekodera aktivirati potrebnu operaciju ALU-a. Kada ALU završi operaciju rezultat rada biće ponovo u nekom od registara.

Mašinski jezik

Izvršavanje memorijski zavisnih instrukcija

- Ovde je proces izvršavanja instrukcija nešto složeniji, jer prije izvršavanja operacije mora se iz memorije u CPU dovesti i podatak. Faza izvršavanja instrukcija koje uzimaju podatke iz memorije i nad njima vrše aritmetičku operaciju može biti prikazana na sledeći način:

  $[IR]_{\text{adresno polje}}$ ---> MAR
  [M] ---> MBR
  Izvrši aritmetičko-logičku operaciju korišćenjem ALU i MBR.

- Ako je instrukcija takvog tipa da piše u memoriju, kao npr. operacija "Smesti sadržaj registra A u adresiranu memorijsku lokaciju", sekvenca mikro-instrukcija može biti sledeća:

  $[IR]_{\text{adresno polje}}$ ---> MAR
  [A] ---> MBR
  [MBR] ---> M

Mašinski jezik

Izvršavanje memorijski zavisnih instrukcija

- Neke instrukcije i čitaju i pišu u memoriju. Posmatrajmo npr. instrukciju "Povećaj sadržaj zadate memorijske lokacije za 1 i preskoči sledeću instrukciju ako je rezultat 0". Takva instrukcija se izvršava na sledeći način:

(1) $[IR]_{\text{adresno polje}}$ **---> MAR**
(2) **[M] ---> MBR**
(3) **[MBR]+1 ---> MBR**
(4) **[MBR] ---> M**
(5) **Ako je [MBR] = 0, onda [PC]+1 ---> PC**

- Nakon što je učitan podatak iz memorije u koraku (2), korakom (3) se povećava njegova vrijednost za 1, a korakom (4) upisuje nazad u istu memorijsku lokaciju iz koje je uzet (jer se MAR nije promijenio). Instrukcijom (5) se testira rezultat operacije i u slučaju da je jednak 0, PC se povećava za 1. Time se preskače sledeća instrukcija u memoriji, jer je PC već povećan za 1 za vreme FETCH faze instrukcije.

Mašinski jezik

Instrukcije grananja

- Instrukcije grananja su poseban podskup instrukcija koje se odnose na memoriju i služe za određivanje adrese sledeće instrukcije koja će se izvršavati (menjaju redosled izvršavanja).

- Posmatrajmo tzv. instrukciju bezuslovnog skoka. Ovom instrukcijom se definiše, u adresnom polju, adresa sledeće instrukcije koju treba izvršiti. Za vrijeme FETCH faze instrukcije bezuslovnog skoka PC je povećan za jedan i pokazuje na adresu instrukcije koja se u memoriji nalazi odmah iza instrukcije bezuslovnog skoka. Međutim, sama instrukcija skoka zahteva izvrašavanje sljedeće instrukcije, ne kako je definisano u PC, već kako nalaže adresno polje instrukcije skoka. Zato se tok izvršavanja instrukcije bezuslovnog skoka može prikazati kako ovako:

  $[IR]_{\text{adresno polje}}$ **---> PC**

- Ako je grananje uslovno, pre izvršavanja skoka proverava se neki zadati uslov.

Mašinski jezik

Indirektno adresiranje

- U primerima prikazanim do sada, smatrali smo da je adresiranje bilo direktno, to jest da je adresno polje instrukcije pokazivalo stvarnu (efektivnu) adresu glavne memorije.
- Ako se u instrukciji koristi indirektno adresiranje tada je za izvršavanje instrukcije potrebno još jedno dodatno čitanje memorije.
- Kao primer posmatrajmo instrukciju kojom se indirektnim adresiranjem uzima podatak iz memorije i nad njim izvršava neka aritmetička operacija. Niz koraka je tada sledeći:

$[IR]_{\text{adresno polje}}$ ---> MAR
[M] ---> MBR
[MBR] ---> MAR
[M] ---> MBR
Izvrši aritmetičku oparaciju korišćenjem ALU i MBR.

Mašinski jezik

Indeksno adresiranje

- Kod indeksnog adresiranja, mora postojati i korak u kome se na adresno polje dodaje vrijednost index registra da se dobije efektivna adresa u memoriji. Za primer uzmimo istu instrukciju. Sada će niz koraka biti:

$[IR]_{\text{adresno polje}}$ ---> MAR
[MAR]+[indeksni registar] ---> MAR
[M] ---> MBR
Izvrši aritmetičku operaciju korišćenjem ALU i MBR.