Programiranje u fizici

2. Programski jezici

Prirodno-matematički fakultet u Nišu
Departman za fiziku

- Pod (računarskim) programiranjem obično se podrazumeva čovekova aktivnost (odnosno, aktivnost programera) usmerena ka rešenju nekog konkretnog problema na računaru.
- Problem se obično rešava konstrukcijom algoritma za rešenje problema, a algoritam se zapisuje posredstvom programskog jezika.
- Po jednoj od najopštijih definicija programskog jezika njegova uloga je da obezbedi konstrukcije (i načine) za organizovanje „izračunavanja“ na računaru.
- Organizovano izračunavanje obično nazivamo „programom“ izračunavanja (ili samo programom).
- Po drugoj mogućoj definiciji, programski jezik predstavlja sredstvo namenjeno za komunikaciju između čoveka i računara, a služi za opis algoritma na način „razumljiv“ računaru (direktno, ili posle niza transformacija).
- Program tada predstavlja jedan način zapisa algoritma – rešenja postavljenog problema.

Po stepenu zavisnosti programskog jezika od računara programske jezike delimo na:

- mašinski zavisne (mašinski i simbolički jezik)
- mašinski nezavisne (jezici višeg nivoa)

- Prilikom projektovanja računara CPU (Centralna procesorska jedinica) se projektuje tako da interpretira skup instrukcija koje se nazivaju instrukcijski skup.
- Svaka instrukcija u ovom skupu ima jedinstven binarni kod koji CPU može da interpretira i on se zove mašinski kod instrukcije, a skup svih mašinskih kodova instrukcija se zove mašinski jezik.
- CPU tj. računar može da izvršava samo programe u formi mašinskog jezika a svaki tip CPU-a razume isključivo svoj sopstveni mašinski jezik.

- Veoma je teško programirati direktno u mašinskom jeziku. Sledi primer instrukcije koja izračunava zbir EAX i EBX registara i rezultat smešta u EAX registar: 03 C3

Primer

- Jedan kratak (hipotetički) mašinski program (od tri komande).

00000100111100111010001010

00000101000100100111100010

00101100101010111111001111

- Programi na prvim računarima bili su zapisani mašinskim jezikom što je uslovilo da uzak krug ljudi piše i održava programe.

- Umesto instrukcija pisanih nizom bitova, uvedene su skraćenice za operacije i simboličke oznake podataka npr. naredbom

vrši se sabiranje podataka a i b.

- Na taj način proces programiranja je u znatnoj meri olakšan, ali i dalje zavisi od konkretnog procesora, tj. i dalje je potrebno poznavati tehničke karakteristike konkretnog računara.

Primer.

- Jednostavni (hipotetički) program na simboličkom jeziku koji bi odgovarao naredbi računanja sume dva broja

C := A + B

- Ovde su A, B i C imena za memorijske lokacije.

LOAD A

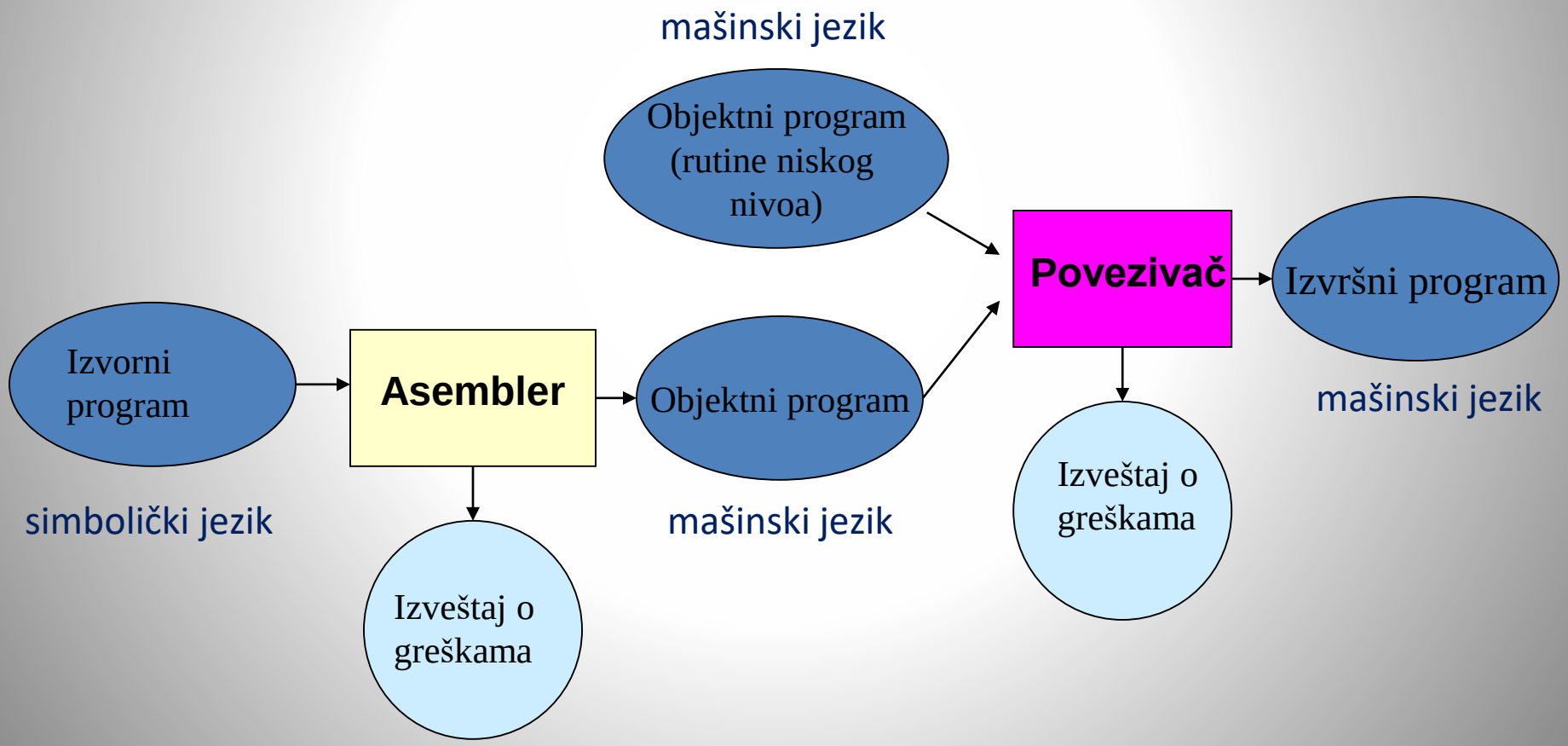
ADD B

STORE C

- Da bi se program napisan na simboličkom jeziku izvršavao na računaru, mora se prethodno prevesti na mašinski jezik.
- Kako svakoj naredbi simboličkog jezika odgovara jedna naredba mašinskog jezika, posao je automatizovan tako što je napisan program koji kao ulaz dobija program napisan u simboličkom jeziku, a kao izlaz odgovarajući program na mašinskom jeziku.
- Program koji vrši prevođenje iz simboličkog u mašinski jezik naziva se asembler.

- Proces prevođenja sa mašinski orijentisanih jezika na mašinski jezik, koji obavljaju asembleri, naziva se ***asembliranje***
- Asembleri prevode izvorni program na mašinski orijentisanom jeziku u međuoblik koji se naziva ***objektni program***
- ***Povezivači*** (linkeri) povezuju objektni program i odgovarajuće rutine niskog nivoa u izvršni program
- Skup naredbi simboličkog jezika zavisi od arhitekture računara, pa program napisan u simboličkom jeziku za jedan računar ne može se koristiti za računar druge arhitekture, već se mora pisati novi program za isti problem.
- Zato za mašinske i simboličke jezike i kažemo da su mašinski zavisni jezici !!!!!!!!!.

Proces prevođenja asemblerskog programa

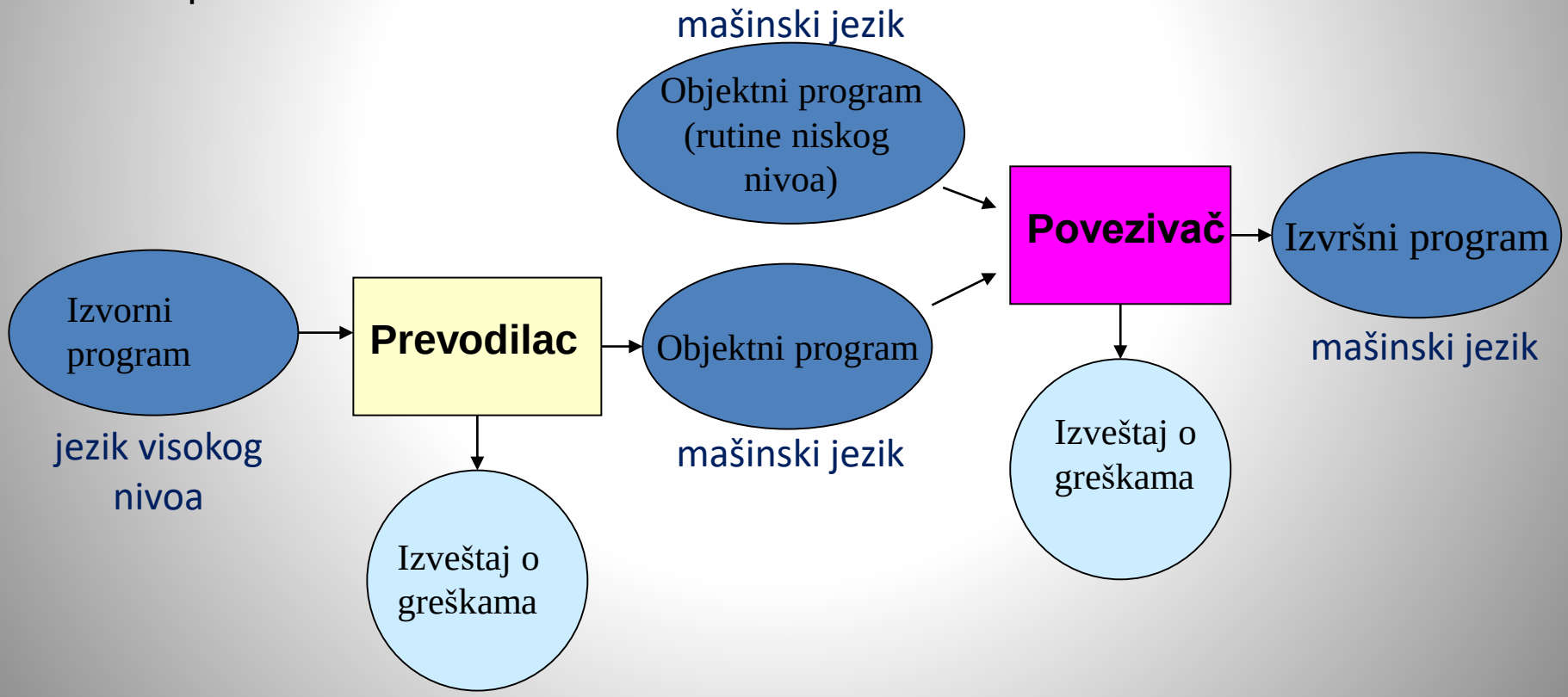


- Programiranje na simboličom jeziku se koristi za pisanje programa tkz. „niskog nivoa“ tj. obično se koriste za programa za interakciju računara sa I/O uređajima:
  - štampačima
  - skenerima
  - uređajima za čuvanje podataka,...
- Njime su pisani programi poznati kao drajveri.
- Razlog: brzina i mogućnost direktnog pristupa resursima računara

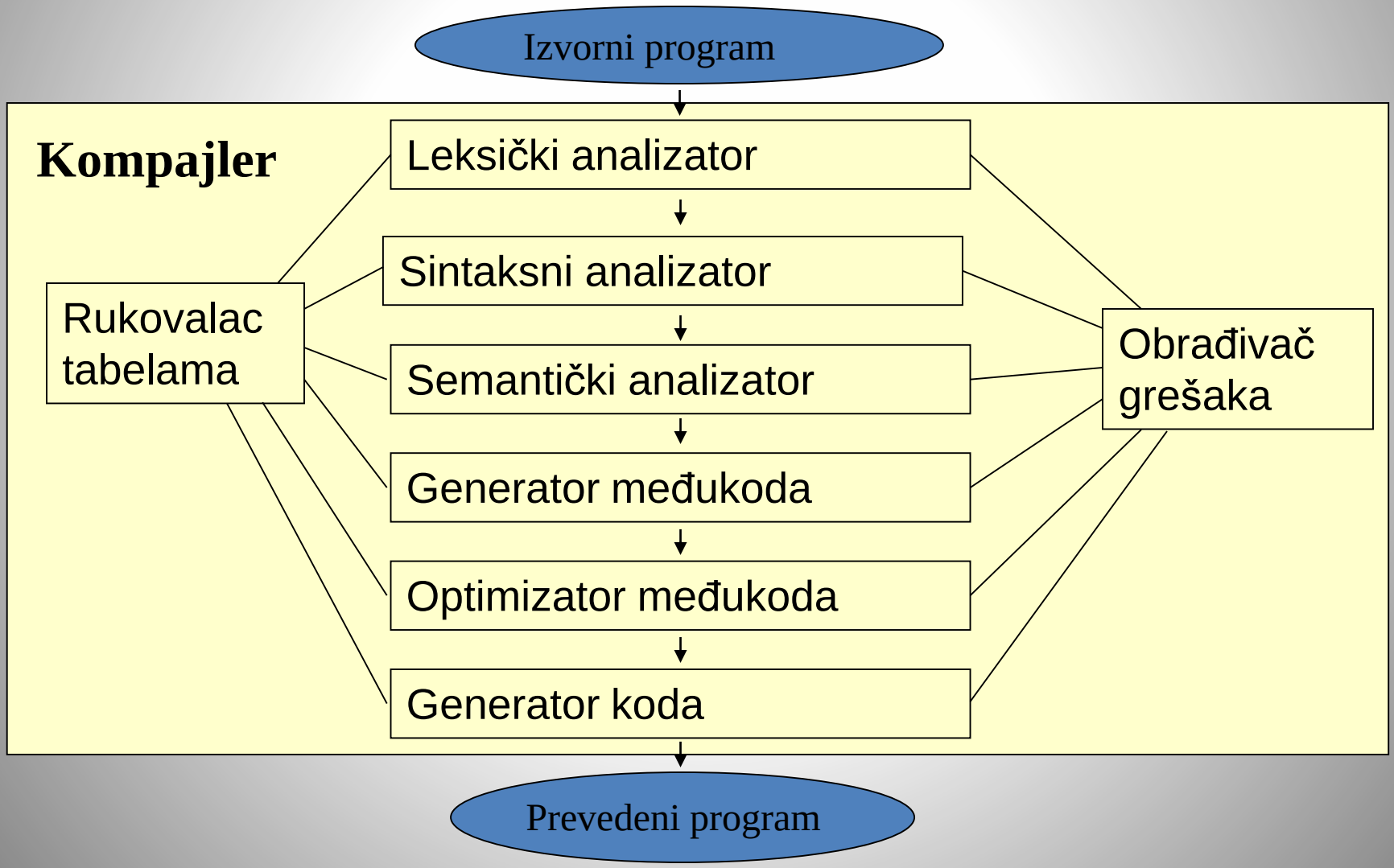
- 50 godina XX veka počinju da se razvijaju mašinski nezavisni jezici, drugim rečima jezici višeg nivoa.
- Korišćenjem jezika višeg nivoa opis naredbi i podataka vrši se na način bliži prirodnom (engleskom) jeziku.
- U ovim jezicima jednoj naredbi odgovara više instrukcija simboličkog jezika.
- S obzirom na to da računar razume samo program napisan na mašinskom jeziku, svaki program pisan jezikom višeg nivoa mora se prevesti na mašinski jezik.

- Na osnovu načina prevođenja i izvršavanja programa, programe za prevođenje delimo na:
  - prevodioce (kompajlere)
  - interpretere.

Najpre su nastali kompilatorski jezici Fortran, Cobol, Algol, PL/I... Kod ovih jezika izgrađuju se programi za prevođenje (kompilatori) kojim se ceo program napisan na višem programskom jeziku prevodi u njemu odgovarajući, mašinski program koji se može više puta izvršavati na računaru.



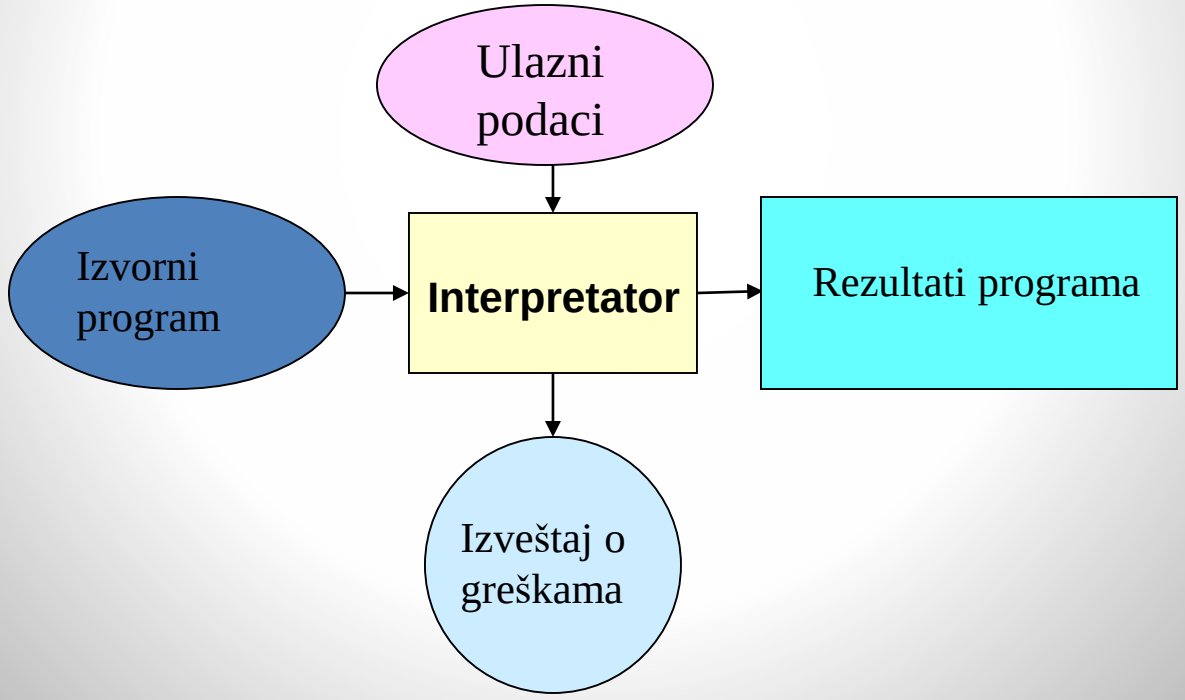
Danas su najpoznatiji jezici ove vrste C i Pascal


Izvorni program
Kompajler
Leksički analizator
Sintaksni analizator
Semantički analizator
Generator međukoda
Optimizator međukoda
Generator koda
Rukovalac tabelama
Obrađivač grešaka
Prevedeni program

- **Leksicka analiza** – u ovoj fazi koristi se leksički analizator (skener) koji prevodi tekst programa u niz tokena (tj. prepoznaje tokene u ulaznom tekstu, na osnovu zadatih leksičkih specifikacija)
- **Sintaksna analiza** – u ovoj fazi parser kao ulaz uzima niz tokena i na osnovu date gramatike, proverava da li je data sekvenca tokena i neterminala ispravna ili ne
- **Semantička analiza** – u ovoj fazi vrši se provera tipova. U ovoj fazi se proverava da li:
  - Sve promenljive u programu su deklarisane pre korišćenja
  - Operandi svakog operatora imaju imaju odgovarajući tip
  - Broj i tip argumenata funkcije odgovara broju i tipu parametara u pozivu funkcije

- **Generisanje memeđukoda** – u ovoj fazi vrši se generisanje jednostavnog međukoda
- **Optimizacija memeđukoda** – u ovoj fazi, (koja je opciona) vrši se optimizacija međukoda u smislu brzine izvršavanja i/ili smanjenja veličine programa
- **Generisanje koda** – generator koda prevodi memeđukod u izvršni kod (mašinski jezik)

- Kod interpreterskih jezika, jedna po jedna instrukcija se prevodi i odmah nakon prevođenja izvršava, faza prevođenja i faza izvršavanja se prepliću.



- Primeri interpreterskih jezika su Lisp, Prolog, Basic, ...

- Važno je napomenuti da jezici višeg nivoa imaju visok stepen nezavisnosti u odnosu na arhitekturu računara i operativni sistem na kojem se izvršavaju.
- Bliži su prirodnom jeziku, čitljiviji i lakši za pisanje programa.
- Skraćeno vreme obuke u programiranju
- Za svaki tip računara postoji program za prevođenje koji isti izvorni kod programa prevodi u odgovarajući mašinski jezik.

- Java – programski jezik visokog nivoa koji je kombinacija prevodioca i interpretera.
- Najpre se izvorni kod napisan u Java programskom jeziku prevodi na mašinski jezik Java virtuelne mašine a zatim se u trenutku pokretanja tako prevedenog programa komande Java virtuelne mašine interpretiraju na ciljnom mikroprocesoru.
- Prednost ovakvog koncepta je da programi napisani u Java programskog jeziku su nezavisni od operativnog sistema i hardverske platforme na kojoj se izvršavaju.