PROGRAMSKI JEZICI

1. O predmetu
2. Razvoj programskih jezika
3. Programske paradigme i podela programksih jezika

Nastavno osoblje

- Nastanik:
  - Suzana Stojković – kabinet 523
- Asistenti:
  - Martin Jovanović – kabinet 523
  - Ivica Marković – kabinet 534
  - Teodora Đorđević – kabinet 533

Sadržaj predmeta

- Predavanja:
  - Teorija programskih jezika
- Vežbe:
  - Programski jezici Java i C#

Način polaganja ispita

- I Kolokvijum (Java) 25
- II Kolokvijum (C#) 25
- Laboratorijska vežba (Windows aplikacije) 10
- Usmeni ispit 40

- Ukupno 100 poena

Literatura

- M. Stanković, *Programski jezici*, Elektronski fakultet, Niš, 2000.

- M. Stanković, S. Stojković, M. Radmanović, I. Petković, *Objektno orijentisani jezici C++ i Java sa rešenim zadacima,* Elektronski fakultet, Niš, 2005.

- Materijal sa predavanja i vežbi dostupan na sajtu predmeta:
  - cs.elfak.ni.ac.rs/nastava

Definicije

Jezik:

- Sistem izražavanja misli koji ima odredjena glasovna i gramatička pravila i služi kao glavno sredstvo za sporazumevanje medju ljudima.
- Način sporazumevanja uopšte.

(Rečnik srpskoga jezika – Matica srpska)

Definicije

- **Prirodni jezici** su jezici nastali spontano u davnim vremenima i njima govore pojedine ljudske zajednice. Njihova pravila su definisana naknadno.
- **Veštački (formalni) jezici** su nastali svesno za konkretnu namenu. Pravila veštačkih jezika su definisana pre njgovog korišćenja.
- **Programski jezici** su formalni jezici namenjeni za kontrolu ponašanja mašina, naročito računara.
- **Programski jezici** služe za komunikaciju sa računarom, ali i da precizno opišu algoritam (pa spadaju i u grupu algoritamskih jezika).

Klasifikacija programskih jezika

Slika 1.1 *Podela programskih jezika*

Mašinski jezici

- Svaka instrukcija mašinskog jezika se sastoji iz:
  - Koda operacije koja treba da se izvede i
  - Adresnog dela koji sadrži:
    - Opranade nad kojima će se operacija izvršiti
    - Adresu gde će se smestiti rezultat izvršene operacije

- Svi elementi mašinske instrukcije se pišu binarnom azbukom

Primer 1.1 *Sekvenca od tri mašinske naredbe od po 32 bita.*

```
0001 1101 1000 0000
0000 0000 1111 1111
0001 1100 0000 0000
0000 0000 1111 1100
0001 1110 0000 0000
0000 0000 0111 0010
```

Asemblerski programski jezici

- **Asemblerski jezici** koriste **simboličke oznake** (mnemoničke kodove) kako za predstavljanje instrukcija mašinskog jezika, tako i za predstavljanje memorijskih adresa gde se nalaze podaci nad kojima se obrada vrši
- Svakoj naredbi asemblerskog jezika odgovara jedna mašinska instrukcija

Primer mašinskih i asemblerskih instruckija

```
0010 0001 0000 1010
0010 0010 0000 0001
0010 0011 0000 1010
```

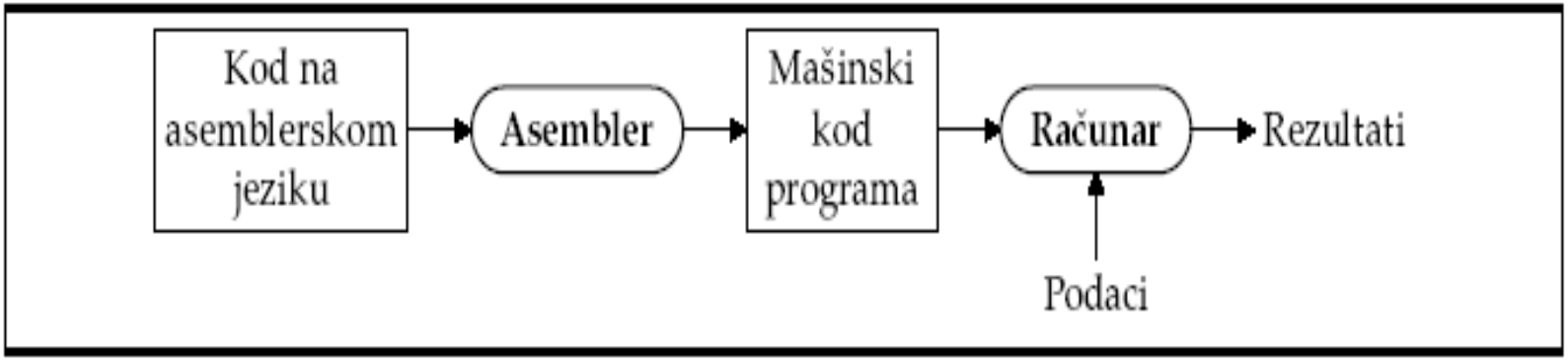
a) Mašinski kod

```
load R1,0A
load R2,01
load R3,0A
```

b) Asemblerski kod

Prevođenje asemblerskih jezika

- Programi pisani asemblerskim programskim jezikom se na mašinski jezik prevode pomoću prevodilaca koji se zovu **asembleri**.



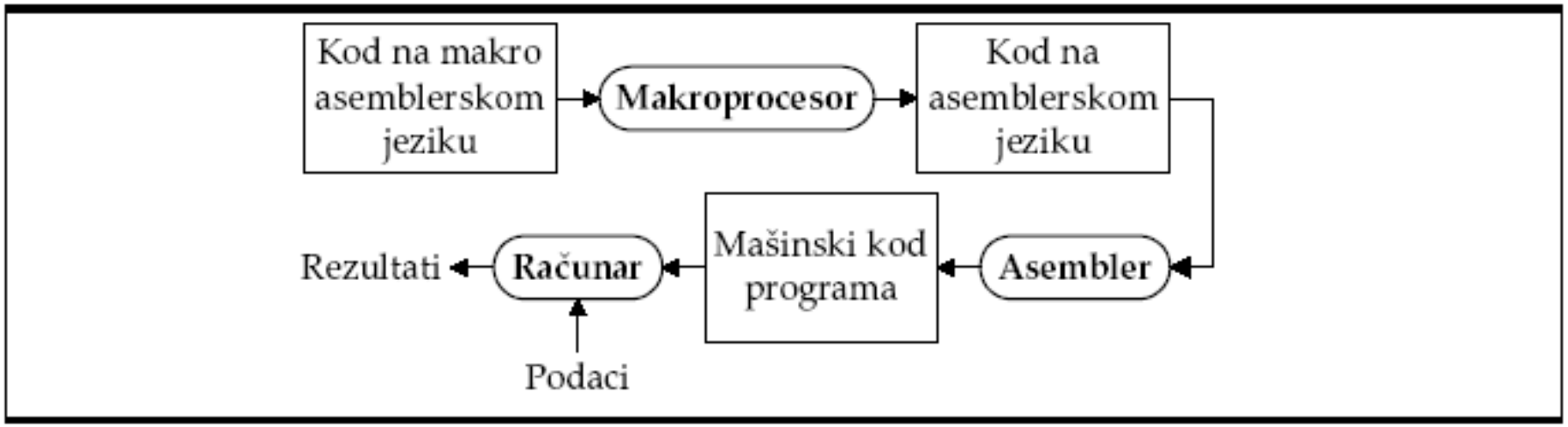
Slika 1.2 Asembleri

Makroasemblerski jezici

- Uvode korišćenje makroinstrukcija
- Makroinstukcije zamenjuju skup instrukcija asemblerskog jezika
- Kreiraju se kad se izvestan skup instukcija ponavlja često u programu.

Prevođenje makroasemblerskih jezika

- Programi pisani makroasemblerskim programskim jezikom se na mašinski jezik prevode pomoću prevodilaca koji se zovu **makroasembleri**.
- Makroasembler se obično sastoji od:
  - **Makroprocesora** – makroinstrukcije zamenjuje skupom asemblerskih naredbi, i
  - **Asemblera** – asemblerski kod prevodi na mašinsi jezik



Slika 1.3 Makroprocesor

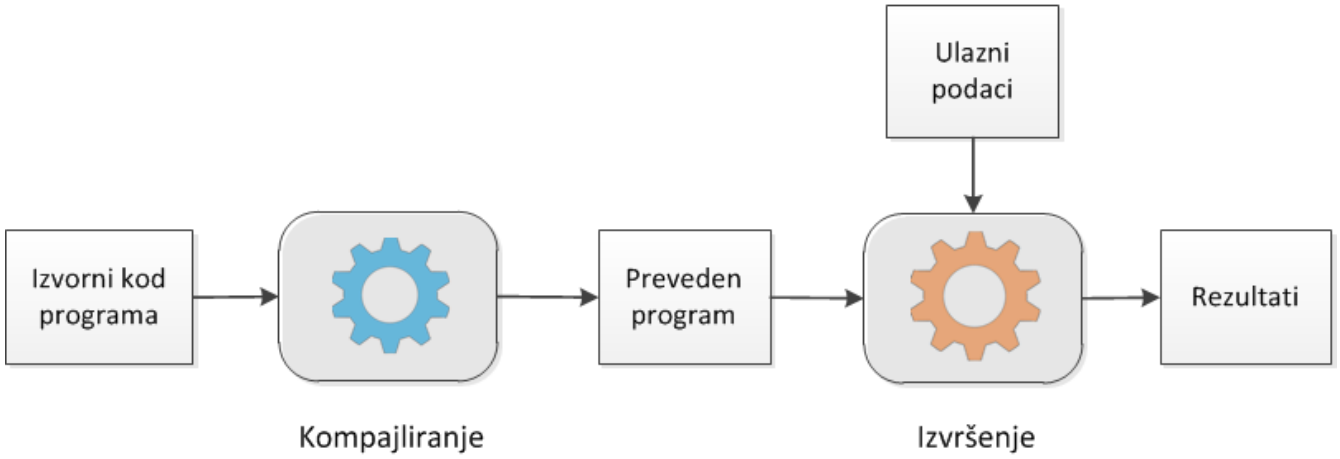
Viši programksi jezici

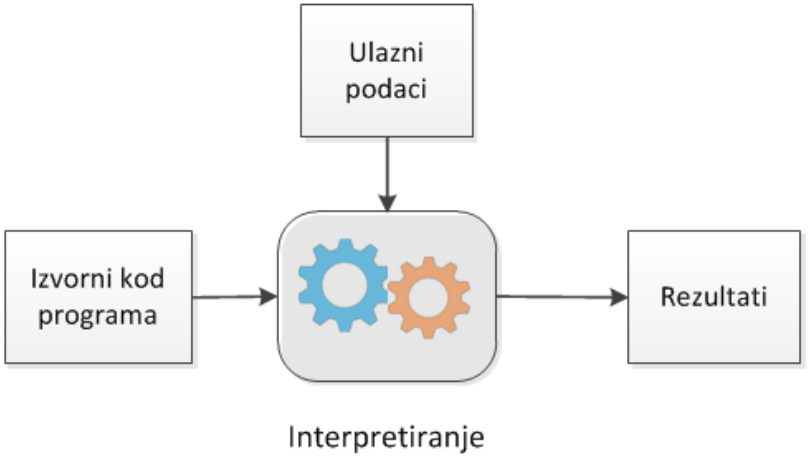
- Naredbe se potpuno približavaju govornom jeziku (engleskom).
- Kod postaje sve razumljiviji korisniku (programeru), a prevodioci postaju sve složeniji

Prevođenje viših programskih jezika

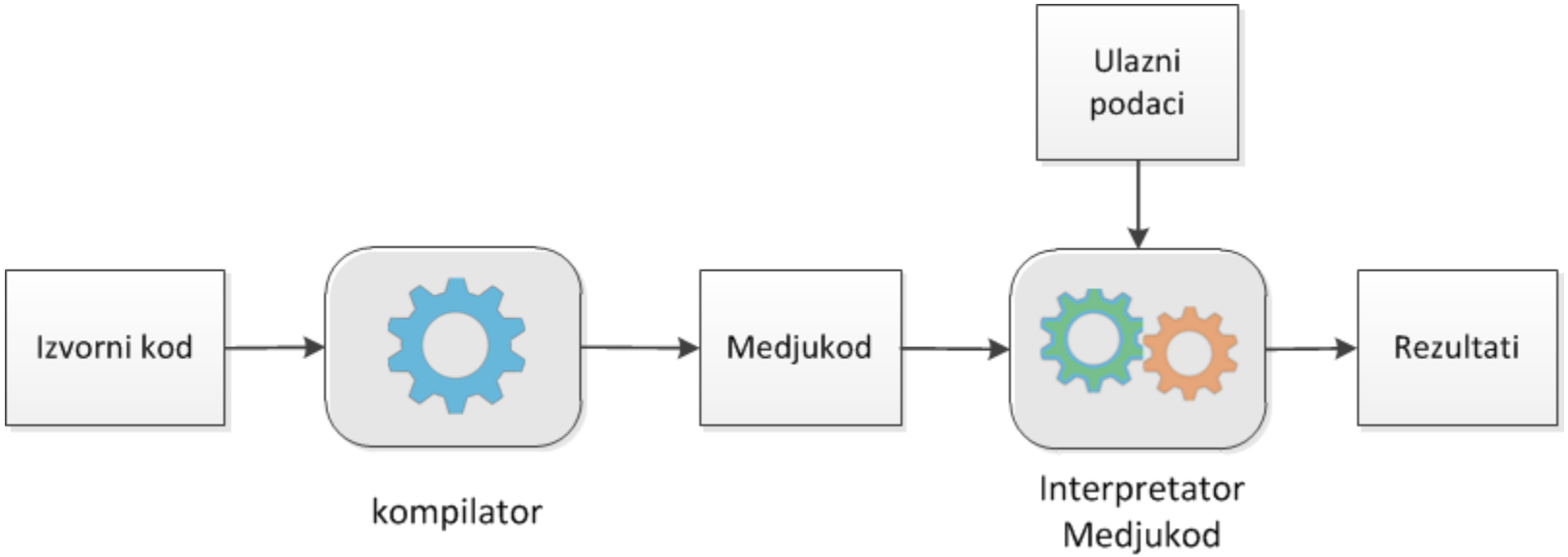
- Viši programski jezici se mogu prevoditi pomoću:
  - Kompilatora – prevode ceo kod i kreiraju izvršnu verziju koja se nakon toga može izvršavati neograničen broj puta bez ponovnog prevođenja,
  - Interpretatora – prevode naredbu po naredbu i svaku prevedenu naredbu odmah izvršavaju,
  - Hibridni prevodioci – najpre se kompilatorom vrši prevođenje do nivoa međukoda koji je jako sličan asemblerskom jeziku, ali potpuno nezavistan od arhitekture računara i operativnog sistema na kojem će se izvršavati. Zatim se medjukod interpretatorom prevodi i izvršava. Jedini deo koji je zavistan od platforme na kojoj će se program izvršavati je taj interpretator međukoda.

Kompilatori i interpretatori

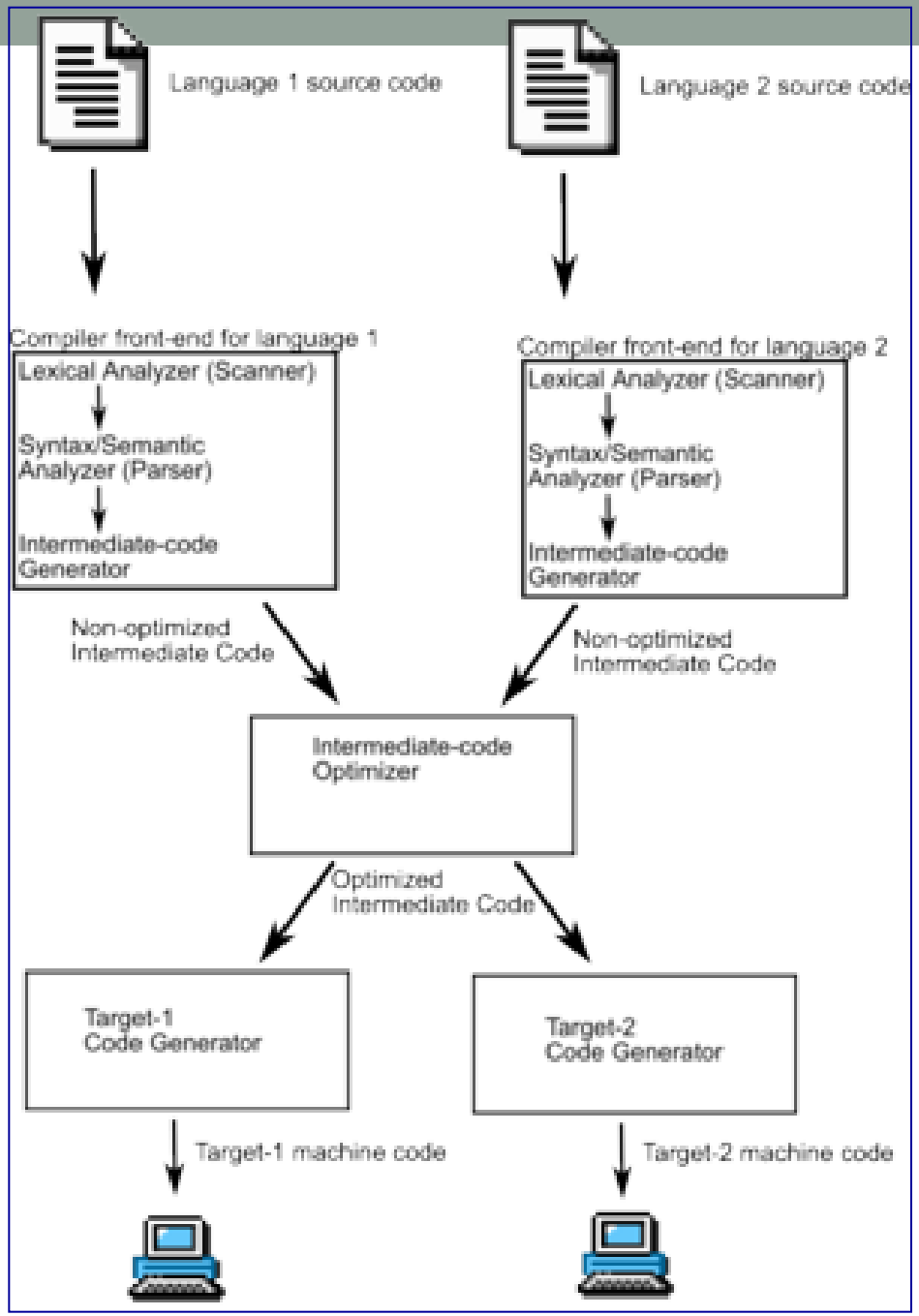




Hibridni prevodioci



Multijezički prevodioci



Podele programskih jezika

- Prema aplikacionom domenu
- Prema paradigmama programiranja
- Prema stepenu zavisnosti od arhitekture računara

Aplikacioni domeni

- Inženjerski (naučni) domen
  - proste strukture podataka (nizovi i matrice), ali se zato zahteva veliki broj preciznih numeričkih računanja
  - FORTRAN – prvi jezik projektovan za precizna numerička računanja
- Poslovni domen
  - Akcenat je na dobrom opisu i velikoj količini podataka koji se obrađuju
  - COBOL – prvi jezik projektovan za poslovni domen
  - Danas se u te svrhe koriste sistemi za upravljanje bazama podataka

Aplikacioni domeni

- Veštačka inteligencija
  - Funkcionalni jezici – Lisp,
  - Logički jezici – Prolog,
- Sistemsko programiranje
  - Zahtevaju veliku preciznost i brzinu rada, mogućnost upravljanja hardverskim resursima
  - C – prvi jezik projektovan za sistemsko programiranje – nastao i premenjen za razvoj operativnog sistema UNIX

Aplikacioni domen

- Internet i Web
  - Jezici za distribuirano programiranje
  - Jezici za formatiranje teksta na Web stranici
    - Murkup jezici – HTML, XML...
  - Jezici za definisanje sadržaja Web stranica
    - Skript jezici – JavaScript, Ruby, PERL, PHP, TypeScript, ...

Programske paradigme

- Programska paradigma definiše stil programiranja

- Najgrublja podela programskih paradigmi:
  - Proceduralne paradigme – cilj je da programer tačno opiše algoritam (način) rešavanja problema.
  - Konceptualne paradigme – zadatak programera je da precizno opiše problem koji se rešava, a mehanizmi programskog jezika treba da obezbede način njegovog rešavanja.

Osnovne programske paradigme

- U literaturi se najčešće 4 paradigme izdvajaju kao osnovne:
  - Imperativna
  - Objektno-orijentisana
  - Funkcionalna
  - Logička

Imperativna programska paradigma

- Nastala zajedno sa Fon-Nojmanovim modelom računara
- Program se sastoji od skupa podataka koji se obrađuju i postupaka (algoritama) kojima se oni obrađuju.
- Koncepti koje imperativni jezici uvode:
  - tipovi podataka – za predstavljanje podataka koji se obrađuju,
  - programske strukture – za predstavljanje algoritama.
- Predstavnici proceduralnih prgramskih jezika:
  - Fortran, Cobol, Algol, PL/1, Basic, Pascal, C …

Objektno-orijentisana programska paradigma

- Najrasprostranjenija programska paradigma
- Softver se posmatra kao mreža objekata koji međusobno komuniciraju razmenom poruka
- Objekti se modeluju korisničkim tipovima podataka - klasama
- Predstavnici objektno-orijentisane paradigme:
  - Simula 67, SmallTalk, C++, Eiffel, Java, C#,...

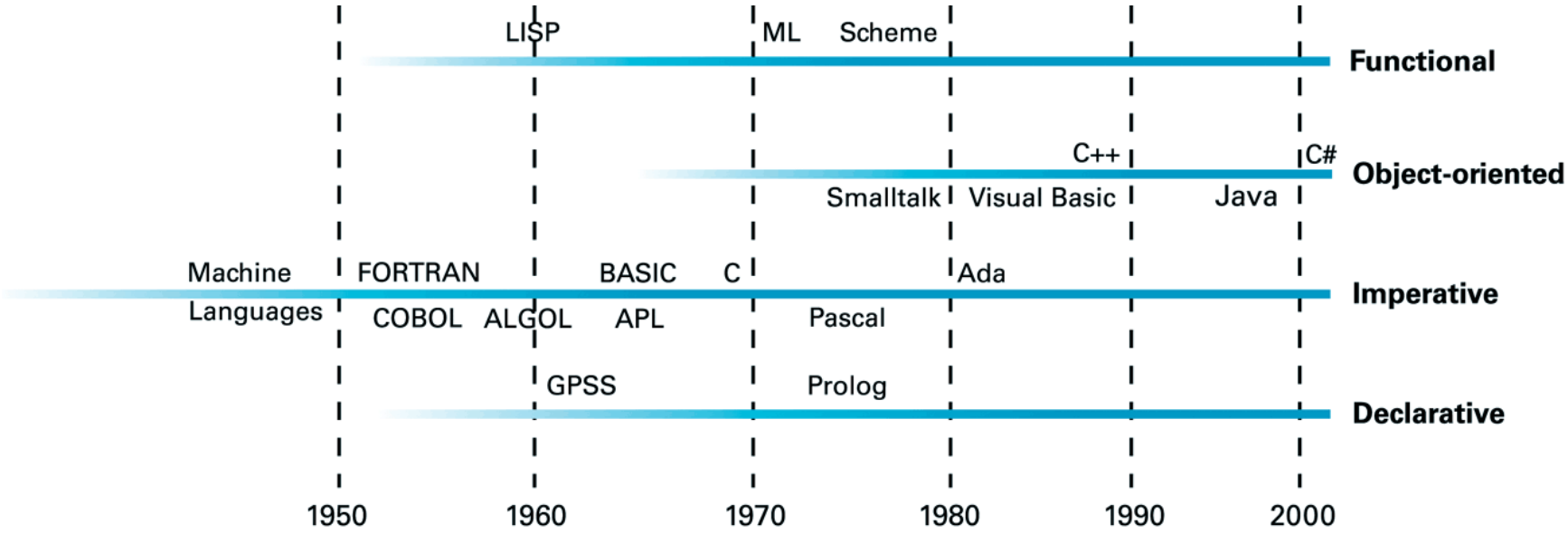
Funkcionalna programska paradigma

- Ceo program se svodi na definisanje i evaluaciju matematičkih funkcija
- Nema bočnih efekata: izlazna vrednost funkcije zavisi isključivo od ulaznih parametara
- Nema programskih struktura – petlje se zamenjuju rekurzivnim funkcijama
- Predstavnici funkcionalnih jezika:
  - Lisp, Scheme, Haskell, ML, Scala, Ocaml...
- Lisp:
  - Jedina struktura podataka koja se koristi su liste
  - Podaci su nepromenljivi

Logička programska paradigma

- Bazirana na principima matematičke logike
- U programu se definišu:
  - Činjenice
  - Pravila zaključivanja
  - Upiti
- Izvršenje programa podrazumeva traženje odgovora na upite korišćenjem datih činjenica i pravila zaključivanja
- Predstavnici logičkih jezika:
  - Prolog, Datalog, CLP, ILOG, Solver, ParLog, LIFE

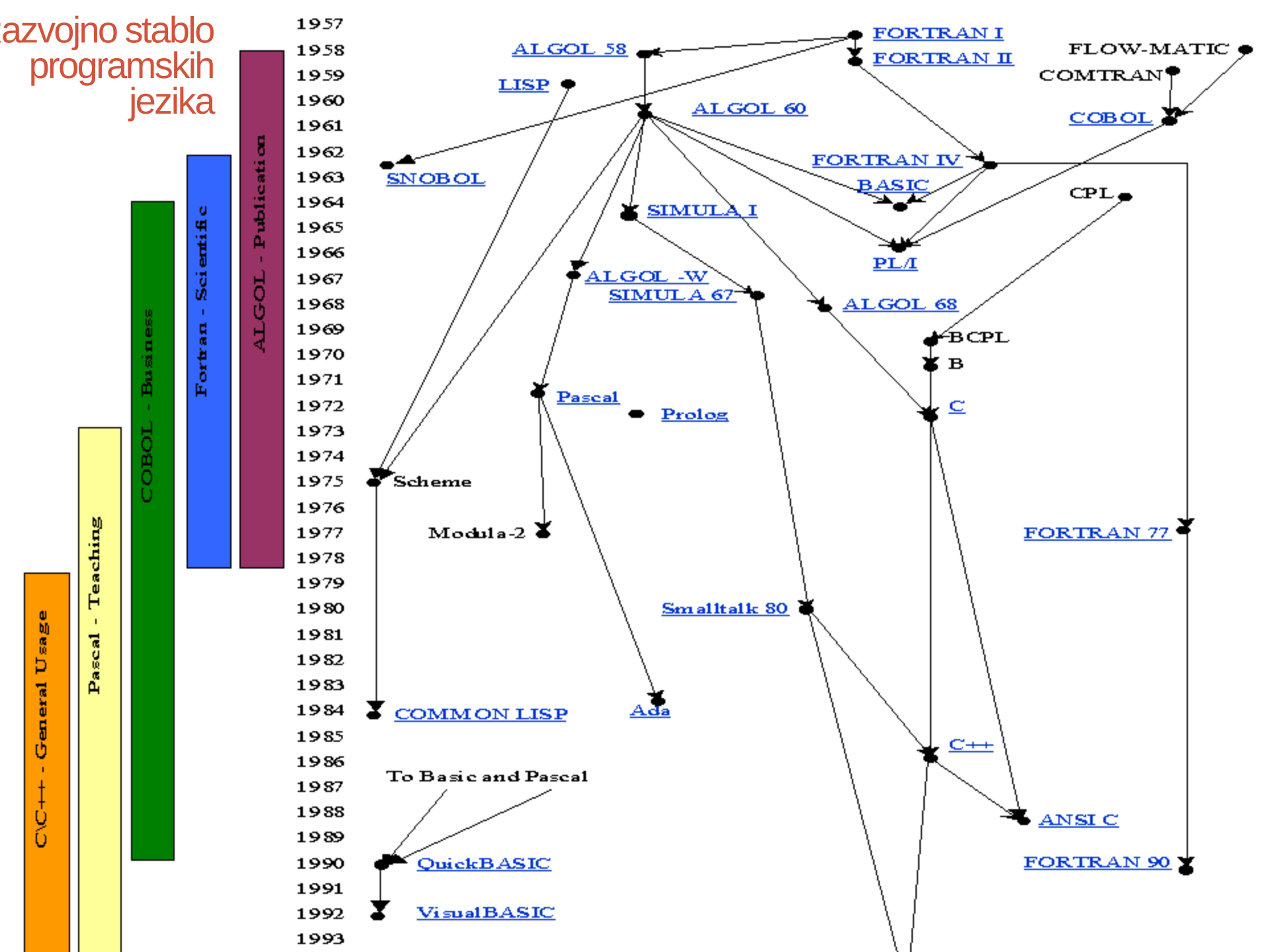
Evolucija paradigmi programiranja



Sporedne programske paradigme

- Predstavljaju kombinaciju osnovnih paradigmi ili podvrstu neke osnovne paradigme
  - Komponentna paradigma
  - Konkurentna paradigma
  - Skript paradigma
  - Generička paradigma
  - Paradigma upitnih jezika
  - Reaktivna paradigma
  - Vizuelna paradigma

Razvojno stablo programskih jezika



Razvoj viših programskih jezika

- **50-te godine XX veka**
  - FORTRAN – FORmula TRANslating system, 1957, John Backus i IBM
    - Imperativni jezik
    - Kompleksna matematička izračunavanja (brzina i tačnost)
    - Statička alokcija memorije – memorijski prostor za ceo kod i sve podatke se rezerviše u fazi prevođenja
  - LISP – LISt Processing, malo posle FORTRANa, 1958, John McCarthy i Paul Graham
    - Funkcionalni jezk
    - Interpretatorskog tipa
  - COBOL – COmmon Business-Oriented language, 1959, Grace Hopper
    - Poslovni domen
    - Efikasno manipulisanje složenim strukturama podataka
    - Efikasan rad sa datotekam

Razvoj viših programskih jezika

- **60-te godine XX veka**
  - ALGOL (58,60,68)
    - Uvodi rekurzije u imperativne jezike,
    - Definisan korišćenjem BNF
  - Simula (Simula I i Simula 67)
    - Prvi objektno-orijentisan jezik
  - Basic
    - Vrlo prosta sintaksa, pogodan za učenje programiranja
  - PL/I
    - FORTRAN + COBOL + SNOBOL+ ... + concurrency + ...

Razvoj viših programskih jezika

- **70-te godine XX veka**
  - C
    - Imperativni jezik za sistemsko programiranje,
  - Pascal
    - Akcenat na strukturnom programiranju (korišćenje programskih struktura sa jednom ulaznom i jednom izlaznom tačkom),
    - Korišćen često za implementaciju kompilatora,
    - Širok spektar strukturnih tipova podataka: zapisi, skupovi, nabrajanja, intervali,...
  - Smalltalk,
    - Potpuno objektno-orijentisani jezik
    - Razvijen tokom 70-ih, prva javna verzija Smalltalk-80
  - Prolog
    - Najistaknutiji predstavnik logičkih jezika

Razvoj viših programskih jezika

- **80-te godine XX veka**
  - C++
  - Objective C
  - Erlang

Razvoj viših programskih jezika

- **90-te godine XX veka**
  - Haskell,
  - Python,
  - Visual Basic,
  - PHP,
  - Ruby,
  - JAVA,
  - PHP,
  - OCaml,
  - Lua,
  - JavaScript...

Razvoj viših programskih jezika

- **XI vek:**
  - Objektno-orijentisani: C#, F#, Groovy, Swift
  - Konkurentni: Scala, Clojure, Go
  - ...