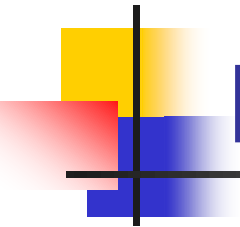


Programski prevodioci 1

Lekcija 1 - Uvod



Kontakt

- Predavač:

- Dragan Bojić
- email: bojic@etf.rs

- Asistenti

- Maja Vukasović (majav@etf.rs)
- Mihajlo Ogrizović (ogrizovic@etf.rs)
- Luka Hrvačević (lukah@etf.rs)



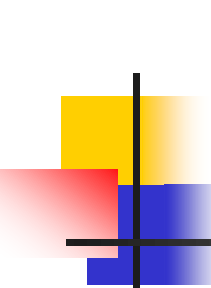
Obaveštenja

- Sajt predmeta:

<http://ir4pp1.etf.rs>

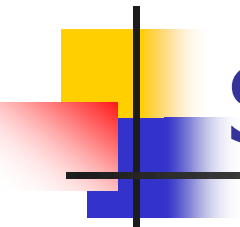
- Mailing lista: 13s114pp1@lists.etf.rs

- Arhiva: <http://lists.etf.rs>



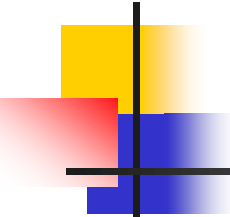
Organizacija predmeta

- (2+2) (P, V) + Domaći zadaci
- Domaći zadatak radi se individualno, obavezan je i uslov je izlaska na ispit
- U svakom ispitnom roku student može polagati ispit koji pokriva celo gradivo i boduje se sa 60 poena. Ukoliko je student polagao kolokvijume (koji se boduju sa 20 poena svaki) uzima se najbolja varijanta zaključivanja ocena:
 - oba kolokvijuma se računaju: $\text{Total} = D1 + D2 + K1 + K2 + \text{ispit} * 1/3$
 - računa se jedan od kolokvijuma: $\text{Total} = D1 + D2 + Kx + \text{ispit} * 2/3$
 - računaju se samo domaći i ispit: $\text{Total} = D1 + D2 + \text{ispit} * 1$
- Uslov za izlazak na ispit je da je $D \geq 20$. Uslov za pozitivnu ocenu je, u sve tri varijante, da broj poena u totalu koji se osvoji kroz kolokvijume i završni ispit ≥ 31 . Granice formiranja ocena su na 51, 61, itd.



Sadržaj predmeta

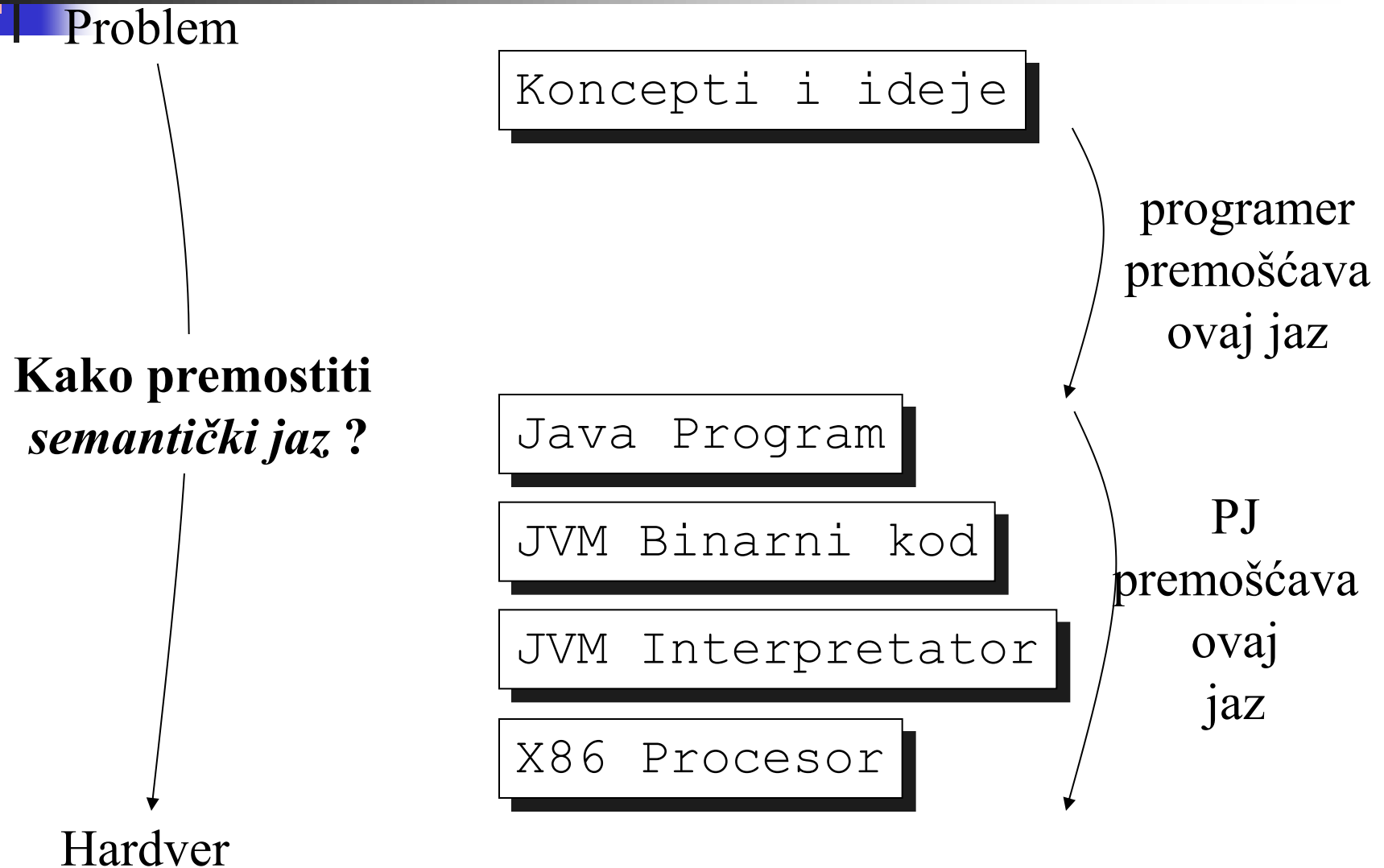
Spisak tema

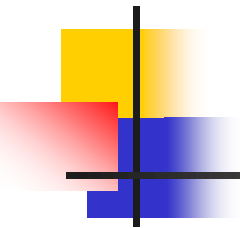


Literatura

- Nastavni materijali sa sajta (P+V)
- Aho, Lam, Sethi, Ullman, Compilers/Principles, Techniques and Tools, 2ed, Addison-Wesley, 2007
- A. W. Appel, Jens Palsberg, Modern Compiler Implementation in Java, Second Edition, Cambridge University Press, 2002
- C. Fischer, R. LeBlanc, Crafting a Compiler, Benjamin-Cummings 1988.

Zašto su nam potrebni programski jezici?





Specifikacija programskog jezika



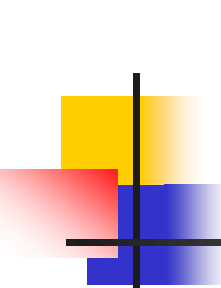
Specifikacija programskog jezika

■ Čemu služi?

- Da se zna kakav je i šta "radi" određeni jezik
- Da se zna kako ga implementirati
- Za sporazumevanje među različitim kategorijama korisnika tog jezika:
 - Projektant jezika, implementator jezika, korisnik-programer...

■ Šta obuhvata specifikacija jezika?

- **Sintaksa** Šta sačinjava "dobro formirani" program
- **Semantika** Šta je značenje (efekat izvršenja) programa



Specifikacija sintakse

Bezkontekstne gramatike (Context Free Grammars, CFG):

- Konačan skup **terminalnih simbola**
- Konačan skup **neterminalnih simbola**
- **Startni (početni) simbol**
- Konačan skup **smena**

Često se CFG pišu u *Bachus Naur Form* (BNF) notaciji.

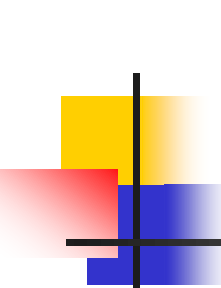
Smena u BNF notaciji piše se kao:

$N ::= \alpha$ gde je N *neterminal*
 a α je sekvenca *terminala* i *neterminala*

$N ::= \alpha \mid \beta \mid \dots$ je skraćenica za nekoliko pravila sa levom stranom N .

Alternativna notacija:

$\langle N \rangle \rightarrow \alpha$



Specifikacija sintakse

CFG definiše skup nizova terminala. Ovaj skup se zove jezik te gramatike.

Primer:

```
Start ::= Letter  
      | Start Letter  
      | Start Digit
```

```
Letter ::= a | b | c | d | ... | z
```

```
Digit  ::= 0 | 1 | 2 | ... | 9
```

P: Koji jezik definiše ova gramatika?

Statička semantika

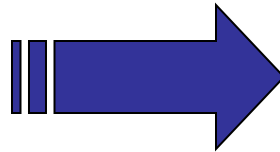
Sintaksna pravila nisu dovoljna za specificiranje izgleda dobro formiranih programa.

Example 1:

```
const m~2
```

```
m + x
```

Nedefinisano



Pravila opsega važenja

Example 2:

```
const m~2 ;
```

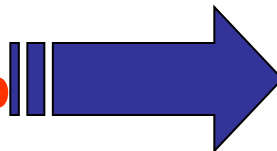
```
var n:Boolean
```

```
begin
```

```
n := m<4;
```

```
n := n+1
```

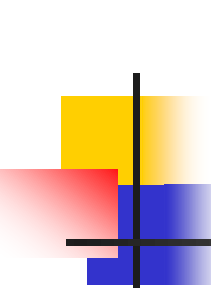
Pogrešan tip



Pravila tipa

```
end
```

Pravila statičke semantike mogu se proveriti bez izvršavanja programa (od strane kompajlera)



Izvršna (engl. run-time) semantika

Specifikacija semantike se odnosi na definisanje “značenja” dobro formiranog programa.

Terminologija:

Izrazi se **izračunavaju** i daju **vrednost** (i mogu ali ne moraju imati bočne efekte)

Iskazi se **izvršavaju** i proizvode **bočne efekte**.

Deklaracije se **elaboriraju** da **proizvedu vezivanja**

Bočni efekti:

- promena vrednosti promenljive
- obavljanje ulaza/izlaza



Izvršna semantika

Primer neformalno specificirane semantike nekih jezičkih konstrukcija:

Iskazi se izvršavaju da ažuriraju promenljive i/ili obave ulaz izlaz.

Iskaz dodele $V := E$ izvršava se na sledeći način:

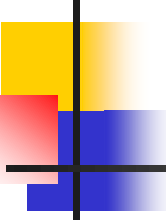
Prvo se izračuna izraz E da dobije vrednost v

potom se v dodeljuje promenljivoj V

Iskaz sekvence $C1; C2$ izvršava se na sledeći način:

Prvo se izvrši iskaz $C1$

potom se izvrši iskaz $C2$

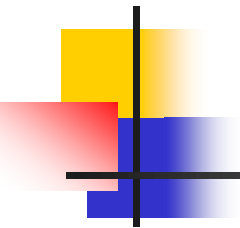


Izvršna semantika

Primer: Semantika deklaracije.

Deklaracija se elaborira da proizvede vezivanje. Takođe može imati sporedni efekt dodele memorije promenljivoj.

*Deklaracija **var** **I**:**T** se elaborira vezujući **I** za novo alociranu promenljivu, čija inicijalna vrednost nije definisana. Promenljiva će biti dealocirana na izlazu iz bloka koji sadrži pomenutu deklaraciju.*



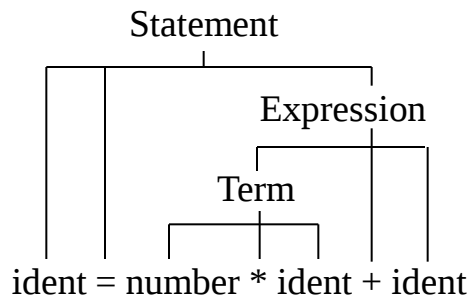
Struktura kompajlera

Niz tokena

Leksička analiza (skeniranje)

Klasa tokena
Vrednost tokena

Sintaksno stablo



međukod



Sintaksno stablo, tabela simbola, ...

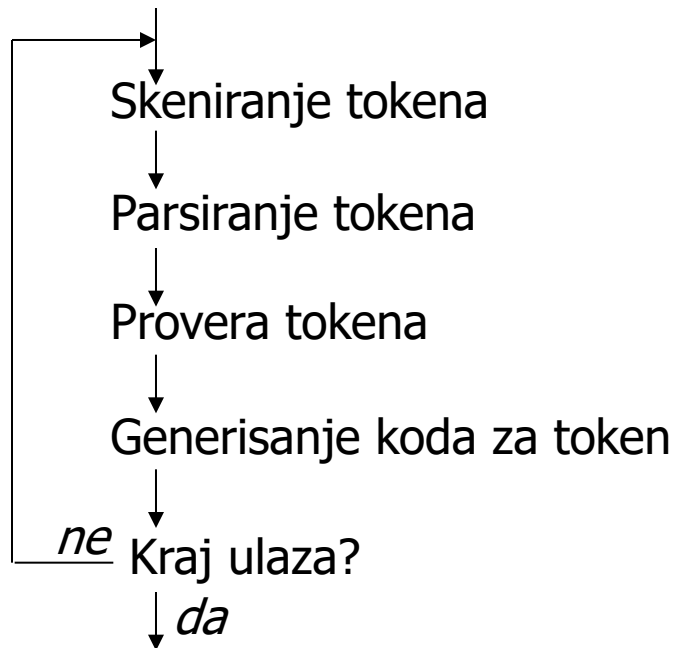
Optimizacija

Generisanje koda

```
ld.i4.s 10
ldloc.1
mul
...
```

Jednoprolazni prevodioci

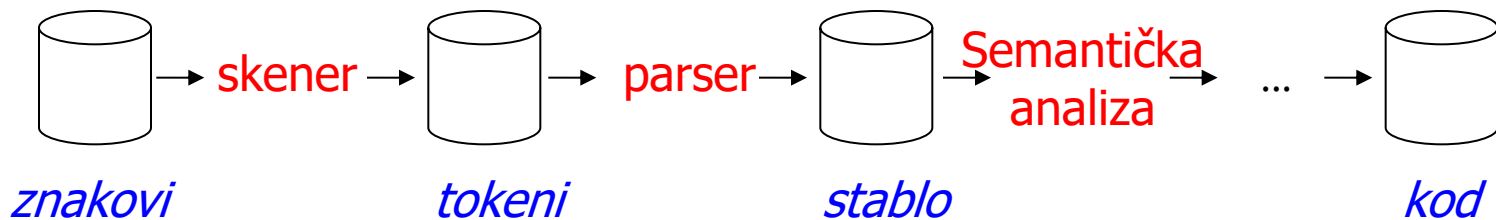
Faze funkcionišu "isprepleteno"



U isto vreme dok se izvorni program čita već se generiše ciljani program.

Višeprolazni prevodioci

Faze su zasebni "programi", koji se izvršavaju sekvencijalno

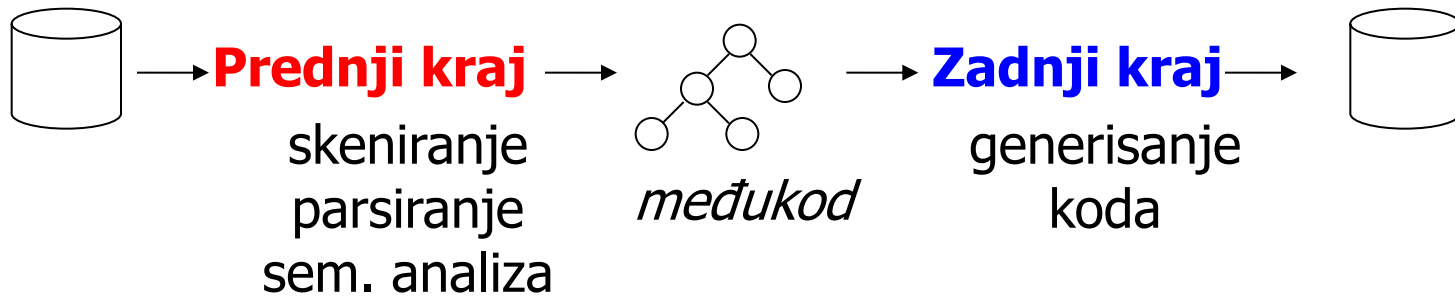


Svaka faza čita iz datoteke i zapisuje u novu datoteku.

Zašto više prolaza?

- Ako je memorija kritična (danas irelevantno)
- Ako je jezik kompleksan
- Ako je važna prenosivost kompajlera

Danas: često dvoprolazni prev.



Jezički zavistan

Java

C

Pascal

Mašinski zavistan

Pentium

PowerPC

SPARC

Sve kombinacije moguće

Prednosti

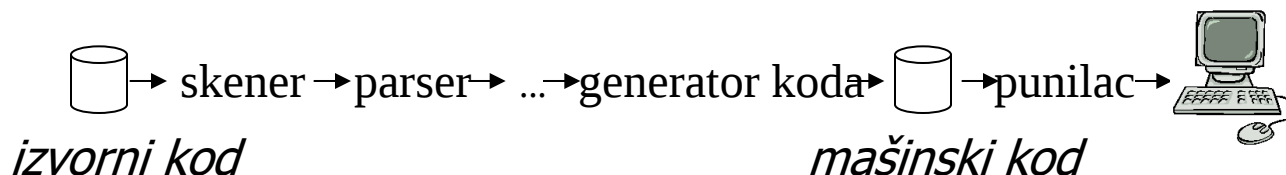
- Bolja prenosivost
- Razne kombinacije između prednjih i zadnjih krajeva moguće
- Optimizacije se lakše vrše na međukodu nego na izvornom kodu

Mane

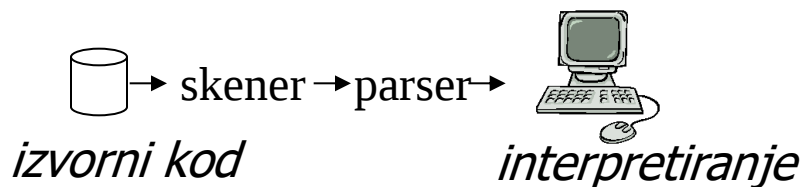
- Sporije prevođenje
- Traži više memorije

Prevodioci i interpretatori

Prevodilac prevodi u mašinski kod

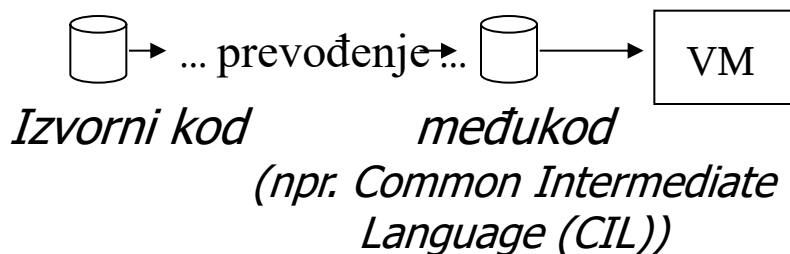


Interpretator "neposredno" izvršava izvorni kod



- Iskazi u petlji se parsiraju nanovo i nanovo

Varijanta: interpretacija međukoda



- Izvorni kod prevodi se u kod *virtuelne mašine* (VM)
- VM interpretira kod simulirajući fizičku mašinu