Programski prevodioci

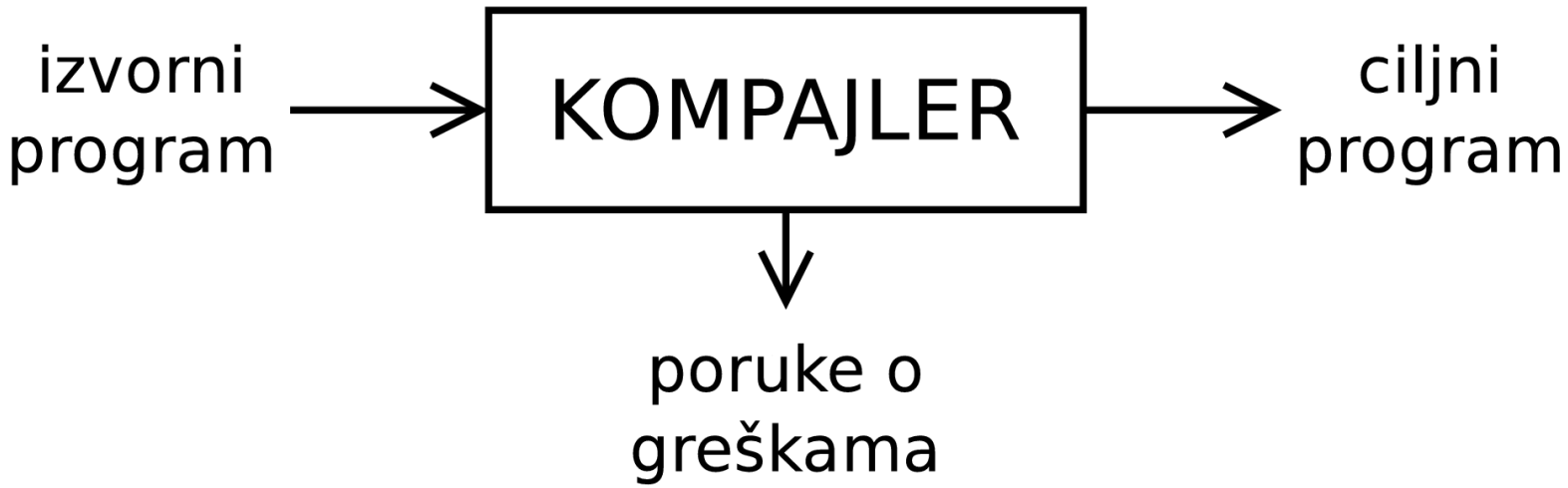
dr Miroslav Hajduković
dr Zorica Suvajdžin
mr Žarko Živanov

Praktični uvod u programske prevodioce
s ciljem predstavljanja tipičnih problema
i načina njihovog rešavanja

Novi Sad 2012.

Uvod

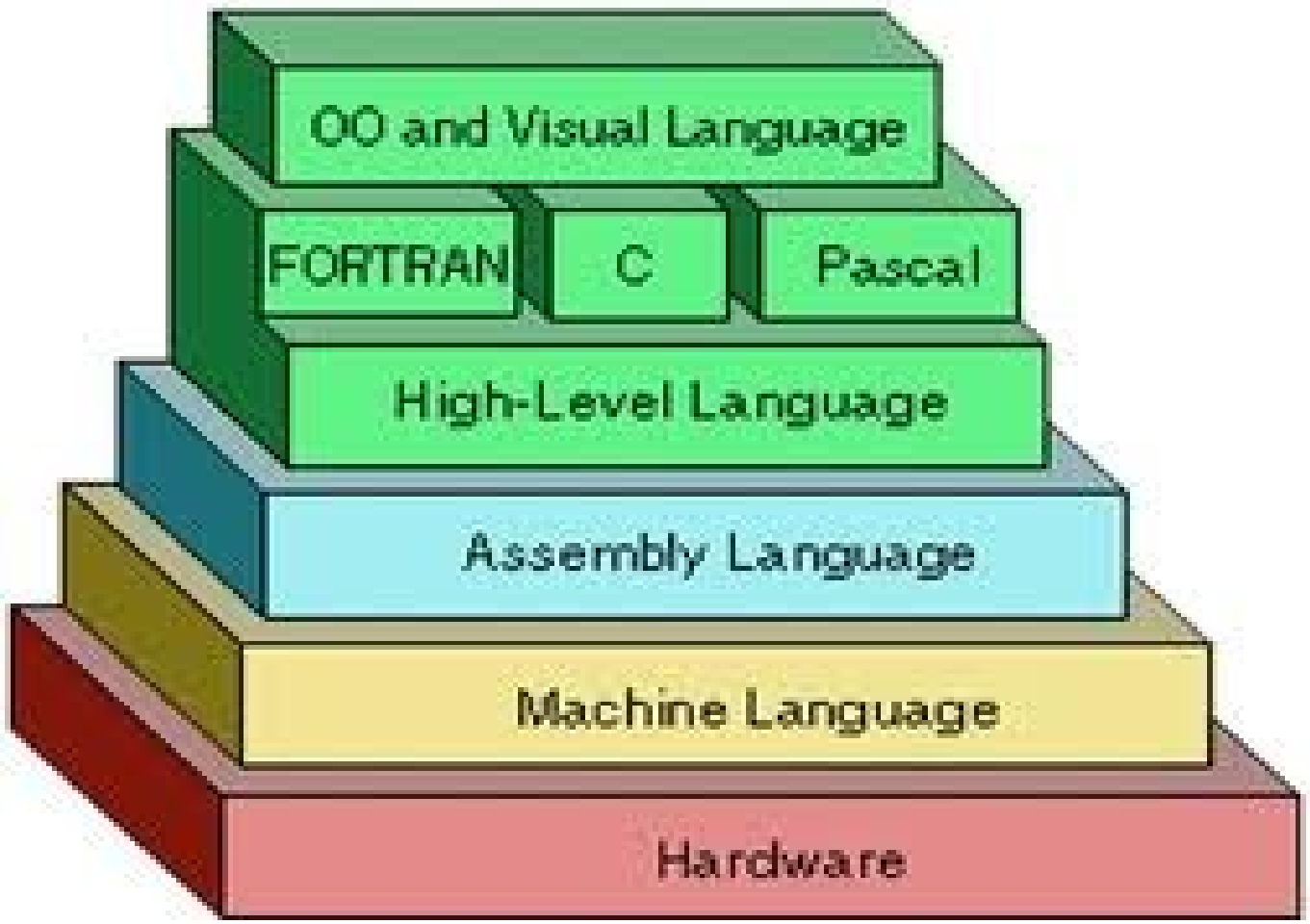
- Zadatak programskih prevodilaca je da:
  - **prevode** programe koji su napisani jednim – **izvornim programskim jezikom** u programe *istog značenja* koji su napisani drugim – **ciljnim programskim jezikom**



Uvod

- Primer izvornog jezika: C, C++, Pascal
- Primer ciljnog jezika: hipotetski asemblerski jezik, Intelov asemblerski jezik, mašinski jezik

Uvod


OO and Visual Language
FORTRAN
C
Pascal
High-Level Language
Assembly Language
Machine Language
Hardware

Mašinski jezik

- Mašinski kod ili mašinski jezik je sistem nedeljivih instrukcija koje izvršava CPU
- Svaka instrukcija izvršava specifičan zadatak
  - operacije nad podacima (u registru ili u memoriji, npr. add ili mov)
  - *jump* operacije (odlučivanje koja će se sledeća instrukcija izvršiti, često uslovna u zavisnosti od rezultata izvršavanja prethodne instrukcije)
- Svaki izvršni (*executable*) program sastoji se od niza ovih atomskih instrukcija

Uvod

- Za prevođenje je potrebno:
  - uspostaviti korespondenciju iskaza (“rečenica”) izvornog i ciljnog programskog jezika
  - prepoznati iskaze (“rečenice”) izvornog programskog jezika i zameniti ih korespondentnim iskazima (“rečenicama”) ciljnog programskog jezika

Primer prevođenja

- Segment programa za računanje najvećeg zajedničkog delioca dva prirodna broja, napisan programskim jezikom *C*:

```
a = 12;
b = 8;
while (a != b)
  if (a > b)
    a = a – b;
  else
    b = b – a;
```

- Prethodni segment programa treba prevesti u segment programa koji je napisan hipotetskim asemblerskim jezikom

Hipotetski asemblerski jezik

Naredba	Značenje
CMP[S,U] op1, op2	Status na osnovu op1 - op2
JMP labela	Bezuslovni skok na adresu
JEQ labela	Skok ako nakon poređenja op1 = op2
JNE labela	Skok ako nakon poređenja op1 != op2
JGT[S,U] labela	Skok ako nakon poređenja op1 > op2
JLT[S,U] labela	Skok ako nakon poređenja op1 < op2
JGE[S,U] labela	Skok ako nakon poređenja op1 >= op2
JLE[S,U] labela	Skok ako nakon poređenja op1 <= op2
PUSH operand	Vrednost operanda na stek
POP operand	Vrednost sa steka u operand
CALL labela	Poziv potprograma
RET	Povratak iz potprograma

Hipotetski asemblerski jezik

Naredba	Značenje
`ADD[S,U] op1, op2, op3`	`op3 <- op1 + op2`
`SUB[S,U] op1, op2, op3`	`op3 <- op1 - op2`
`MUL[S,U] op1, op2, op3`	`op3 <- op1 * op2`
`DIV[S,U] op1, op2, op3`	`op3 <- op1 / op2`
`MOV op1, op2`	`op2 <- op1`
Direktiva	**Značenje**
`WORD n`	`Zauzimanje n memorijskih lokacija`

- Sve memorijske lokacje i registri su 32-bitni (4 bajta)
- Aritmetičke naredbe i skokovi mogu biti označeni (`S`) ili neoznačeni (`U`).
- Radni registri imaju imena od `%0` do `%15`, pri čemu je `%13` registar za povratnu vrednost funkcije, `%14` je pokazivač frejma, dok je `%15` pokazivač vrha steka.
- Konstante se označavaju sa prefiksom `$` (na primer `$15`).
- Labele se pišu navođenjem imena praćenim dvotačkom (na primer `petlja:`). Sistemske labele počinju znakom `@` (na primer `@for:`).

Primer prevođenja

- Korespondencija iskaza programskog jezika *C* i naredbi hipotetskog asemblerskog jezika (podrazumeva se da neoznačenoj celobrojnoj promenljivoj **a** odgovara memorijska lokacija sa labelom **a**, a neoznačenoj celobrojnoj promenljivoj **b** odgovara memorijska lokacija sa labelom **b**)
  - Iskazi pridruživanja

```
a = 12;
b = 8;
a = a - b;
b = b – a;
```

```
MOV   $12,a
MOV   $8,b
SUBU  a,b,a
SUBU  b,a,b
```

Primer prevođenja

- **while** iskaz

```
while (a != b)
  while_telo
```

```
@while0:
        CMPU    a,b
        JEQ     @false0
@true0:
        while_telo
        JMP @while0
@false0:
```

- **if** iskaz

```
if (a > b)
  then_telo
else
  else_telo
```

```
@if1:
            CMPU    a,b
            JLEU    @false1
@true1:
            then_telo
            JMP @exit1
@false1:
            else_telo
@exit1:
```

Primer prevođenja

```
a = 12;
b = 8;
while (a != b)
  if (a > b)
    a = a – b;
  else
    b = b – a;
```

```
MOV   $12,a
...
```

Primer prevođenja

```
a = 12;
b = 8;
while (a != b)
  if (a > b)
    a = a – b;
  else
    b = b – a;
```

```
MOV   $12,a
MOV   $8,b
...
```

Primer prevođenja

```
a = 12;
b = 8;
while (a != b)
  if (a > b)
    a = a – b;
  else
    b = b – a;
```

```
                MOV   $12,a
                MOV   $8,b
@while0:
                CMPU  a,b
                JEQ   @false0
@true0:
                ...
                JMP   @while0
@false0:
```

Primer prevođenja

```
a = 12;
b = 8;
while (a != b)
  if (a > b)
    a = a – b;
  else
    b = b – a;
```

```
                MOV   $12,a
                MOV   $8,b
@while0:
                CMPU  a,b
                JEQ   @false0
@true0:
@if1:
                CMPU  a,b
                JLEU  @false1
@true1:
                ...
                JMP   @exit1
@false1:
                ...
@exit1:
                JMP   @while0
@false0:
```

Primer prevođenja

```
a = 12;
b = 8;
while (a != b)
  if (a > b)
    a = a – b;
  else
    b = b – a;
```

```
                MOV   $12,a
                MOV   $8,b
@while0:
                CMPU  a,b
                JEQ   @false0
@true0:
@if1:
                CMPU  a,b
                JLEU  @false1
@true1:
                SUBU  a,b,a
                JMP   @exit1
@false1:
                ...
@exit1:
                JMP   @while0
@false0:
```

Primer prevođenja

```
a = 12;
b = 8;
while (a != b)
  if (a > b)
    a = a – b;
  else
    b = b – a;
```

```
                MOV   $12,a
                MOV   $8,b
@while0:
                CMPU  a,b
                JEQ   @false0
@true0:
@if1:
                CMPU  a,b
                JLEU  @false1
@true1:
                SUBU  a,b,a
                JMP   @exit1
@false1:
                SUBU  b,a,b
@exit1:
                JMP   @while0
@false0:
```

Analiza prevedenog programa

```
    MOV   $12,a
    MOV   $8,b
@while0:
    CMPU  a,b
    JEQ   @false0
@true0:
@if1:
    CMPU  a,b
    JLEU  @false1
@true1:
    SUBU  a,b,a
    JMP   @exit1
@false1:
    SUBU  b,a,b
@exit1:
    JMP   @while0
@false0:
```

- suvišna druga naredba **CMPU a,b**
- naredbu **JMP @exit1** treba zameniti naredbom **JMP @while0** da bi se izbegla dva uzastopna skoka:
- do neefikasnosti je došlo zbog parcijalnog pristupa prilikom zamene iskaza izvornog jezika iskazima ciljnog jezika i nesagledavanja šireg konteksta u kome se zamena obavlja
- ovim problemom bavi se optimizacija

```
    MOV   $12,a
    MOV   $8,b
@while0:
    CMPU  a,b
    JEQ   @false0
@true0:
@if1:
    JLEU  @false1
@true1:
    SUBU  a,b,a
    JMP   @while0
@false1:
    SUBU  b,a,b
@exit1:
    JMP   @while0
@false0:
```

Osvrt na prevođenje

- Prevođenje je zasnovano na prepoznavanju iskaza (rečenica) izvornog jezika
- Sastoji se od prepoznavanja
  - nizova znakova koji obrazuju simbole (reči)
  - nizova simbola koji obrazuju iskaze (rečenice)
- U toku prevođenja se otkrivaju i greške koje se prijavljuju korisnicima

Osvrt na prevođenje - analiza

- Prepoznavanje iskaza izvornog programskog jezika – **analiza**

SIMBOLI												
a	=	12	;	b	8	while	(	!=	)	if	>	else

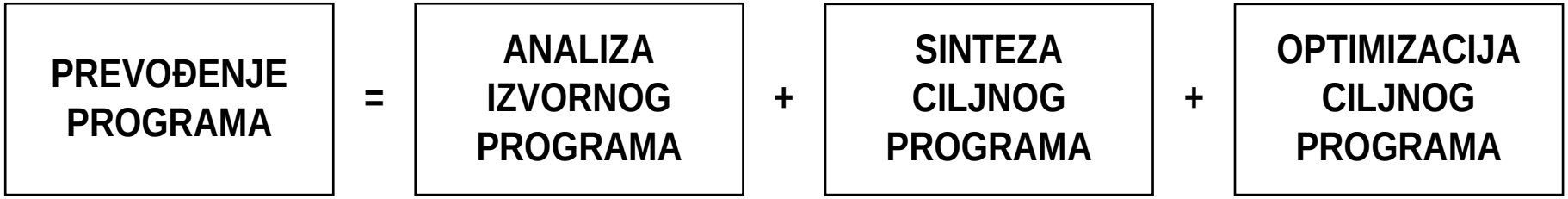
- prepoznavanje **simbola**: **leksička analiza**
- otkrivanje pogrešnih simbola: **leksičke greške**
  - primer: upotreba nedozvoljenih karaktera, npr: **`cou%nt`**
- prepoznavanje iskaza: **sintaksna analiza**
- otkrivanje (formalno) pogrešnih iskaza: **sintaksne greške**
  - primer: izostavljanje otvorene male zagrade iza **`if`**
- prepoznavanje značenja iskaza: **semantička analiza**
- otkrivanje semantički pogrešnih iskaza: **semantičke greške**
  - primer: korišćenje nedefinisane promenljive

Osvrt na prevođenje - analiza

- leksička analiza *(lexical analisys)*
  - leksički ispravno: **`count,  a3, =`**
  - leksička greška: **`cou%nt, 3a, :=`**
- sintaksna analiza *(syntax analisys)*
  - sintaksno ispravno: **`if(a>3) a=1;`**
  - sintaksna greška: **`if(a>3  a=1;`**
- semantička analiza *(semantic analisys)*
  - semantički ispravno: **`int a;  a=3;`**
  - semanička greška: **`int a;  b=3`**;

Osvrt na prevođenje - sinteza

- Generisanje iskaza ciljnog jezika – **sinteza**



- Program prevodilac: **KOMPAJLER**
- Deo kompajlera zadužen za leksičku analizu: **SKENER**
- Deo kompajlera zadužen za sintaksnu analizu: **PARSER**
- Deo kompajlera zadužen za sintezu: **KOD GENERATOR**

- **Optimizacija** izgenerisanog programa (izraz optimizacija se koristi u smislu poboljšanja programa, a ne u smislu pravljenja optimalnog programa)

Leksička i sintaksna analiza teksta

- Sintaksna analiza se oslanja na leksičku analizu
- Leksička i sintaksna analiza omogućuju prepoznavanje iskaza programskog kao i rečenica govornog jezika, pa se mogu primeniti ne samo na programski nego i na obični tekst
- Na primer, leksička i sintaksna analiza običnog teksta su potrebne da bi se odredio
  - ukupan broj reči u tekstu
  - ukupan broj rečenica u tekstu
  - ukupan broj linija

Sintaksa jezika

- Za leksičku i sintaksnu analizu (programskog ili običnog) teksta potrebno je poznavati **sintaksu**, odnosno **gramatiku** (programskog ili govornog) jezika koja određuje pravila pisanja njegovih iskaza
- Gramatika može biti zadana neformalno:
  - reč je niz slova
  - rečenica je niz reči iza kojih dolazi tačka
  - tekst je niz rečenica
- Neformalna gramatika je neprecizna

Formalna gramatika

- Gramatika (programskog) jezika se izražava na formalan način u ***BNF* obliku**: Bakus-Naurova forma (*Backus-Naur form*)
- Ovako izražena gramatika se sastoji od **pravila** koja određuju dozvoljene načineređanja **pojmova** i **simbola** :

  ***pojam* → *pojmovi* i/ili simboli**

- Leva strana pravila (pre znaka →) sadrži pojam koji može biti zamenjen sekvencom pojmova i/ili simbola koje sadrži desna strana pravila (iza znaka →). Jedan od pojmova predstavlja **polazni pojam**.
- Pojmovi su vezani za sintaksnu analizu (*nonterminal symbol*), a simboli su vezani za leksičku analizu (*terminal symbol*)
- primer:

  dodela → ime ”=“ izraz ”;“

Formalna gramatika za običan tekst

text
→ *ε*
→ *text sentence*

sentence
→ capital_word *words* dot

words
→ *ε*
→ *words* word
→ *words* capital_word

word
→ small_letter
→ word small_letter

capital_word
→ capital_letter
→ capital_word small_letter

dot
→ "."

Formalna gramatika za običan tekst

```
capital_letter
    → "A"
    → "B"
    → "C"
      ...
    → "Z"
small_letter
    → "a"
    → "b"
    → "c"
      ...
    → "z"
```

- Napomene
  - Pojmovi su napisani ukošenim slovima (*italikom*)
  - ε označava praznu desnu stranu pravila
  - Razmak i kraj linije imaju funkciju separatora simbola

Formalna gramatika

- **Proširena Bakus-Naurova forma** (*Extended Bacus-Naur Form*) uvodi sledeće elemente
  - na desnoj strani pravila mogu da se koriste:
    - [...] da označe da se sadržaj zagrada može pojaviti nijednom ili jednom
    - { ... } da označe da se sadržaj zagrada može pojaviti nijednom, jednom ili višestruko
    - (...) da označe grupisanje
    - | da označi alternative

Formalna gramatika za običan tekst

text
→ { *sentence* }
sentence
→ capital_word *words* dot
words
→ { (word | capital_word) }

word
→ small_letter { small_letter }
capital_word
→ capital_letter { small_letter }
dot
→ "."

Formalna gramatika za običan tekst

```
capital_letter
   →  "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H"
    | "I" | "J" | "K" | "L" | "M" | "N" | "O" | "P"
    | "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X"
    | "Y" | "Z"
small_letter
   →  "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h"
    | "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p"
    | "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x"
    | "y" | "z"
```

Primer primene formalne gramatike

- Gramatika svakog (programskog) jezika obuhvata:
  - simbole
  - pojmove
  - pravila i
  - polazni pojam
- Gramatika (programskog) jezika je potrebna za proveru ispravnosti (programskog) teksta
- Tokom provere ispravnosti ulaznog teksta, leva strana pravila se zamenjuje desnom sve dok se ne dobije ulazni niz simbola. Ovaj proces zamene se naziva **izvođenje** *(derivation).*
  - Ukoliko se svaki put prvo zameni pojam koji se nalazi krajnje levo u pravilu onda se proces naziva **izvođenje s leva**. Ako se zamenjuje krajnje desni pojam u pravilu, proces se zove **izvođenje s desna**.

Primer primene formalne gramatike

- Primer primene tekst gramatike za dokazivanje da je iskaz

  Ovo je tekst.

  ispravan tekst (tj. da je u skladu sa gramatikom)
  - izvođenje s leva u desno

    text ⇒ *text sentence* ⇒ *ε sentence* ⇒ *sentence* ⇒

    capital_word *words* dot ⇒

    Ovo *words* dot ⇒

    Ovo *words* word dot ⇒*

    Ovo word word dot ⇒

    Ovo je word dot ⇒

    Ovo je tekst dot ⇒

    Ovo je tekst .
  - znak ⇒ označava izvođenje u jednom koraku, a znak ⇒* označava izvođenje u više koraka

    text ⇒* **Ovo je tekst .**

Primer primene formalne gramatike

- prikaz izvođenja u obliku stabla