JUN 2018

ELIMINACIJE

1. Tehnike definisanja novih lingvističkih nivoa su
   - **a. Proširenje DA**
   - **b. Prevođenje DA**
   - c. Kombinacija NE
   - d. Dodavanje NE

2. Data je sledeća gramatika

   S -> get E from S
   S -> begin S L
   S -> print E END
   L -> quit
   L -> ; S L
   E -> num EQ num

   Navedi terminalne I neterminalne simbole

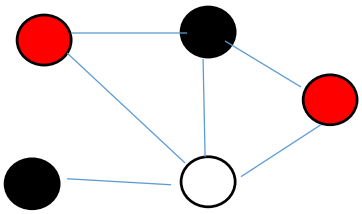
   Terminalni: S, L, E
   Neterminalni: get, from, begin, print, quit, END, ;, num, EQ

3. Sve instrukcije osnovnog bloka osim prve i poslednje predstavljaju njegovu unutrašnjost. Koje od navedenih instrukcija međureprezentacije mogu biti u unutrašnjosti osnovnog bloka.
   - **a. pristup memoriji DA**
   - b. labela NE
   - **c. poziv potprograma DA**
   - d. uslovni skok NE
   - e. bezuslovni skok NE
   - **f. move instrukcija DA**

4. Dato je sledeće parče koda. Označiti od koje do koje linije traje životni vek promenljive **a**?

```
int foo(int x,int y)
{
    int b, z, a = x-2;        <- Početak
    b = 4 + a;
    b += a + y;
    z = 1 + b;
    return z – a;             <- Kraj
}
```

5. Obojiti dati graf sa 3 boje.



Značajan čvor je čvor koji ima veći rang od broja dostupnih boja tj. registara. Vrši se uprošćavanje grafa tako što se sa grafa skidaju redom čvorovi koji su manjeg ranga od broja dostupnih boja I smeštaju se na stek.

Zatim se sa steka redom vraćaju čvorovi I boje se tako da nemaju susede koji su iste boje kao I oni.

6. Zaokružiti instrukcije koje predstavljaju adresno osetljive lokacije (???)

```
main:
      la    $t0, 7
 lab1:
      lw    $t1, 0($t0)
      li    $t0, 5
      b     lab2 + 1
      li    $t2, -5
      blez  lab1
lab2:
      nop
```

ISPIT

1. Zašto se poziva modul asemblera za izračunavanje izraza?

   U prvom prolazu za obradu EQU, u drugom za određivanje vrednosti operanada.

2. Na osnovu datog asemblerskog koda popuniti tabelu simbola. Širina memorije za dati primer je 8 bita, veličina podataka je 24 bita (.word = 24 bita) a veličina svake instrukcije 16 bita.

```
.data
tmp:   .word 100
a:     .word 70
val:   .word 4
.text
main:
       la          $t0, tmp
       lw          $t1, 0($t0)
       li          $t0, 5
       li          $t2, -5
lab:
       add   $t1, $t1, $t0
       addi  $t2, $t2, 1
       blez  $t2, lab
       nop
       la          $t0, a
       sw          $t1, 0($t0)
exit:
       nop
```

SIMBOL	VREDNOST	SEKCIJA
tmp	0000	.data
a	0003	.data
val	0006	.data
main	0009	.text
lab	0011	.text
exit	001D	.text

3. Data je gramatika jezika J. Ispod svake rečenice napisati da li je ta rečenica deo jezika J I ako jeste, napisati redosled primene produkcija koji generiše tu rečenicu. Početni simbol je S.

   1. S -> y += P;
   2. S -> F += x – P;
   3. P -> x – 2
   4. P -> F – 2
   5. F -> z
   6. F -> x – 2

   Rečenice:

   y += x – 2 – 2 - jeste deo gramatike, 1 -> 4 -> 6

   z += x – x – 2 – 2 - jeste deo gramatike, 2 -> 5 -> 4 -> 6

   z += x – p - nije deo gramatike

   Da li je data gramatika analitička ili generativna? GENERATIVNA

4. Napisati regularan izraz koji definiše realne brojeve u jeziku C (53.7, .28 , 4. , -.28)

   “-“? (([0-9]+”.”[0-9]*) | ([0-9]*”.”[0-9]+))

5. Navesti uslove za veoma dobro definisanu petlju, sa stanovišta otkrivanja petlji koje mogu biti fizički podržane (hardverske petlje).
   a. Poznat je naziv iteracione promenljive i njena početna vrednost
   b. Poznat je korak iteracije
   c. Poznat je broj iteracija

6. Poželjne osobine međukoda su:
   Nezavisan od izvornog koda (programskog jezika) I ciljnog procesora. Zato je pogodan za prevođenje u mašinski kod bilo kog odredišnog procesora. Čvorovi stabla međukoda moraju imati jasno I jednostavno značenje zbog lakše kasnije optimizacije koda.

7. Šta je programski trag (engl. Trace)
   Trag predstavlja niz iskaza koji se mogu uzastopno izvršavati tokom izvršavanja programa. Može sadržati i uslovne skokove.

8. Navesti Briggs-ovu strategiju za sigurno spajanje čvorova grafa smetnji.

   Čvorovi a i b se mogu spojiti ukoliko rezultantni čvor ab ima manje od k zajedničkih suseda.

9. Ukratko opisati tehniku kompaktovanja TRAŽI-UNAPRED

   Ova tehnika višeg nivoa koristi primitive ZAMENI I IZBACI. Polazi od početka programskog segmenta i u smeru njegovog izvršenja (unapred) traži NOP instrukciju.

10. Navesti loše osobine BSS punjača.

    Vektor prelaza zauzima prostor u memoriji a koristi se samo za povezivanje.
    Obrađuju segmente procedura, ali ne olakšavaju pristup segmentima deljivih podataka.

JUL 2018

ELIMINACIJE

1. Navesti tehnike za definisanje novih lingvističkih nivoa
    a. **PROŠIRENJE** – nove procedure koriste primitive osnovnog sistema
    b. **PREVOĐENJE** – sa novog jezika na jezik osnovnog sistema (***prevodilac prevodi program napisan na jednom jeziku tj. lingvističkom nivou u program napisan na drugom lingvističkom nivou***)
    c. **INTERPRETACIJA** – faze prevođenja I izvršenja su vremenski zavisne

2. Date su 4 formalne gramatike (skup terminalnih simbola, neterminalnih simbola i početni simbol). Treba zaokružiti ispravne rečenice a precrtati neispravne.

3. Dat je asemblerski kod. Potrebno je podeliti kod u osnovne blokove horizontalnim linijama i konstruisati graf toka upravljanja.

    Po definiciji, osnovni programski blok započinje labelom LABEL, završava se iskazom skoka JUMP ili CJUMP i untar bloka ne postoji LABEL JUMP ili CJUMP

4. Dat je C++ kod. Potrebno je zaokružiti linije koda gde je promenljiva **b** živa. Takođe je potrebno označiti u kojoj liniji počinje, a u kojoj se završava životni vek promenljive.

    Promenljiva je živa ako postoji putanja do čvora u kojem se koristi a da pritom ne prelazi preko čvora u kojem se definiše.
    Za svaku definiciju promenljive a, u čvoru koji nije MOVE, dodaj smetnje između a i svake promenljive žive na izlazu iz tog čvora.
    Za MOVE instrukciju a = c, dodaj smetnje između promenljive a i svake promenljive žive na izlazu čvora koja nije c.

5. Dato je 6 grafova. Ukoliko imamo 3 registra, zaokružiti čvorove koji su značajni, a precrtati one koje nisu.
    Značajan čvor je čvor koji ima veći rang od broja dostupnih registara.

6. Data su dva segmenta asemblerskog koda. Ispod svakog treba napisati simbole koje taj segment definiše i simbole koje taj segment potražuje.

    U .data se samo definiše, dok u .text može I da se definiše I potražuje. Npr. Ako se koristi JUMP u njoj se potražuje labela, ili u add se potražuju operandi itd.

ISPIT

1. Dat je C++ kod. Konstruisati graf smetnji I graf toka upravljanja.
2. Nabrojati 3 poželjne osobine međukoda.
   Nezavisan od izvornog koda (programskog jezika) I ciljnog procesora. Zato je pogodan za prevođenje u mašinski kod bilo kog odredišnog procesora. Čvorovi stabla međukoda moraju imati jasno I jednostavno značenje zbog lakše kasnije optimizacije koda.

3. Nabrojati tehnike kompaktovanja višeg nivoa.
   TRAŽI-UNAPRED, TRAŽI-UNAZAD, KOMPAKTUJ-SA-PRETHODNIM, KOMPAKTUJ-SA-NAREDNIM

4. Napisati formulu za analizu životnog veka čvora n.
   in[n] = use[n] U (out[n] – def[n])
   $out[n] = U_{succ}[n]\ in[s]$

5. Objasniti BNF notaciju.
   Notacija za izražavanje produkcije (generativnih pravila). Svaka produkcija generiše sintaksnu klasu (neterminalni simbol).

6. Dat je asemblerski kod. Treba odrediti koja je vrednost određenih simbola ako je dužina instrukcije 4 bajta a dužina reči takođe 4 bajta.

7. Transformisati datu gramatiku tako da nema konlikata.
   - E -> a b T
     E -> a c G
     Ovde prva dva simbola određuju produkciju

   - E -> a T x
     E -> a T z
     Rešenje je: E -> a T E'
     E' -> x
     E' -> z

   - E -> a T x
     E -> a T
     Rešenje je: E -> a T E'
     E' -> x
     E' -> ε

   - E -> a T b c
     T -> b
     T -> ε
     Rešenje je: E -> a b b c
     E -> a b c

- S -> E $
  E -> E + T T -> T * F F -> id
  E -> E - T T ® T / F F -> num
  E -> T T -> F F -> (E)

  Rešenje: S -> E $
  E -> T E' T -> F T'
  E' -> + T E' T' -> * F T'
  E' -> - T E' T' -> / F T'
  E' -> T' ->
  F – id
  F – num
  F – (E)

8. Dato je 4 for petlje. Potrebno je označiti petlje koje su dobro formirane a za one koje nisu dobro formirane, potrebno je objasniti zašto nisu.

9. Data su 3 objektna modula, M1, M2 I M3. Treba odrediti na kojoj adresi se nalazi funkcija f (nakon punjenja) u povezanom fajlu koja pripada M3, ako punjenje počinje od adrese 0x00800000. Veličina modula M1 je 0x0500, veličina modula M2 je 0x2500, a veličina modula M3 je (neka velicina, data je). Funkcija f se nalazi na udaljenosti 0x0210 od početka M3.

SEPTEMBAR 2018

ELIMINACIJE

1. Zaokruži tačnu tvrdnju a precrtaj netačnu

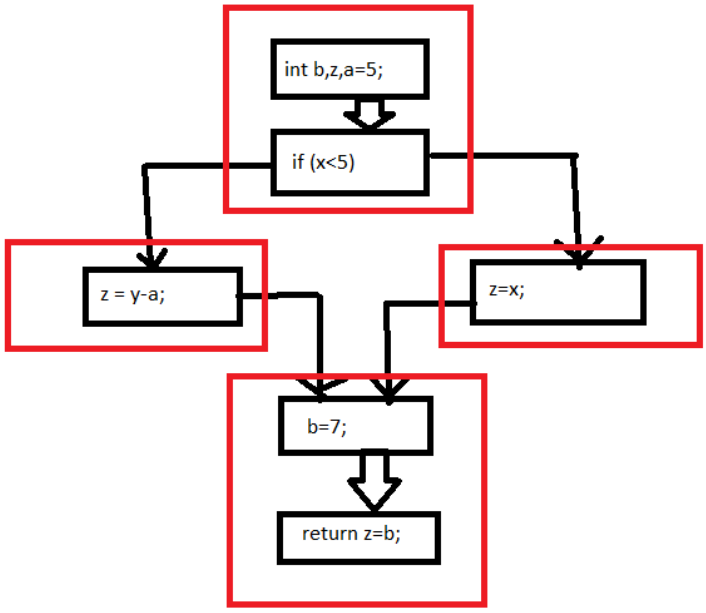
    a. Tabela simbola definiše razliku između mašinske I pseudoinstrukcije
    b. Tabela simbola pridružuje kod operacije I njemu pridruženu vrednost
    c. Popunjena tabela simbola je ulaz u prvi prolaz asemblera (ugrađena je u assembler)
    d. Popunjena tabela simbola je izlaz iz prvog prolaza asemblera

2. Data je gramatika, odredi sve terminalne i neterminalne simbole

    Neterminalni su bili S, E, Q (svi koji su sa leve strane produkcije).
    Terminalni su svi sa desne strane koji nisu neterminalni (uključujući i reči kao EQ, END I sve što je spojeno.)

3. Dat je kod, nacrtati graf toka upravljanja i oznaciti osnovne blokove

```
int foo(int x, int y)
{
      int b, z, a=5;
      if(x<5)
          z=y-a;
      else
          z=x;
      b=7;
      return z-b;
}
```

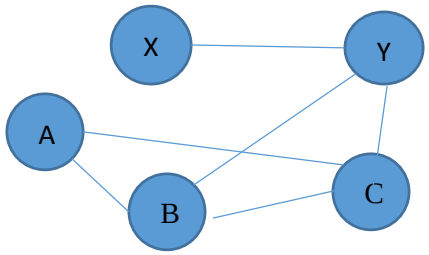


4. Koristeći kod iz prethodnog zadatka, navesti brojeve ispred linija na kojima je **b** živa i linije u kojima počinje i završava se njen životni vek.

```
int foo(int x, int y)
{
        int b, z, a=5;
        if(x<5)
            z=y-a;
        else
            z=x;
        b=7;                <- START
        return z-b;         <- END
}
```

5. Dat je graf smetnji. Zaokruži tačne tvrdnje a precrtaj netačne.

   a. x je u smetnji sa b preko y
   b. a i b su u smetnji
   c. a i y ne mogu se obojiti istom bojom
   d. a i c mogu da se smeste u isti registar



6. Date su tri operacije, jedna od njih (druga) je neka vrsta JUMP-a. Data je tabela sa 3 punjenja, u 3. punjenju treba dodati šta nedostaje (treba skontati da fali adresa labele na koju se skače, i koja je jedina drugačija u svakom punjenju, zato što se u svakom punjenju menja adresa operacija, a tako i labele). Bilo je potrebno upisati 0x0115.

ISPIT

1. Čemu služi modul za izračunavanje izraza. Zašto se poziva u prvom a zašto u drugom prolazu?

2. Na osnovu datog asemblerskog koda popuniti tabelu simbola. Širina memorije za dati primer je 8 bita, veličina podataka je 24 bita (.word = 24 bita) a veličina svake instrukcije 16 bita.

```
.data
tmp:        .word 100
a:          .word 70
val:        .word 4
.text
main:
            la                  $t0, tmp
            lw                  $t1, 0($t0)
            li                  $t0, 5
            li                  $t2, -5
lab:
            add         $t1, $t1, $t0
            addi        $t2, $t2, 1
            blez        $t2, lab
            nop
            la                  $t0, a
            sw                  $t1, 0($t0)
exit:
            nop
```

SIMBOL	VREDNOST	SEKCIJA
tmp	0000	.data
a	0003	.data
val	0006	.data
main	0009	.text
lab	0011	.text
exit	001D	.text

3. Data je gramatika jezika J. Ispod svake rečenice napisati da li je ta rečenica deo jezika J I ako jeste, napisati redosled primene produkcija koji generiše tu rečenicu. Početni simbol je S.

3. S -> y += P;	*3.* P -> x – 2	*5.* F -> z
4. S -> F += x – P;	*4.* P -> F – 2	*6.* F -> x – 2

Rečenice:

y += x – 2 – 2 - jeste deo gramatike, 1 -> 4 -> 6

z += x – x – 2 – 2 - jeste deo gramatike, 2 -> 5 -> 4 -> 6

z += x – p - nije deo gramatike

4. Napisati regularne izraze za realne brojeve

(treba da pokrijes sve, da moze a i ne mora biti minus, da moze a i ne mora biti broj na mestu ispred zareza, isto i za mesto iza zareza, ako na nekom mestu nema broj onda na drugom mestu mora biti, to sve moras da pokrijes)

("-"?[0-9]*"."[0-9]+) | ("-"?[0-9]+"."[0-9]*)

5. Dat je kod. Potrebno je nacrtati graf toka upravljanja i popuniti graf smetnji.

6. Navesti tehnike kompaktovanja višeg reda.

TRAŽI-UNAPRED, TRAŽI-UNAZAD, KOMPAKTUJ-SA-PRETHODNIM, KOMPAKTUJ-SA-NAREDNIM

7. Napisati jednačine životnog veka za čvor n I objasniti sve elemente jednačina

in[n] = use[n] U (out[n] – def[n])

out[n] = U_{succ}[n] in[s]

in – skup promenljivih živih na ulazu
out – skup promenljivih živih na izlazu
def – skup promenljivih koje čvor definiše
use – skup promenljivih koje čvor koristi
succ – skup čvorova naslednika

8. Napisati uslove za dobro definisanu petlju
    a. Poznat je naziv iteracione promenljive i njena početna vrednost
    b. Poznat je korak iteracije

9. Navesti tri dobre osobine dobrog međukoda
   Nezavisan od izvornog koda (programskog jezika) I ciljnog procesora. Zato je pogodan za prevođenje u mašinski kod bilo kog odredišnog procesora. Čvorovi stabla međukoda moraju imati jasno I jednostavno značenje zbog lakše kasnije optimizacije koda.

10. Data su 3 objektna modula, M1, M2 I M3. Treba odrediti na kojoj adresi se nalazi funkcija f (nakon punjenja) u povezanom fajlu koja pripada M3, ako punjenje počinje od adrese 0x00800000. Veličina modula M1 je 0x0500, veličina modula M2 je 0x2500, a veličina modula M3 je (neka velicina, data je). Funkcija f se nalazi na udaljenosti 0x0210 od početka M3.