Potpis Dežurnog: _____

Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Programski Prevodioci 1 (IR4PP1)
Nastavnik: doc. dr Dragan Bojić
Asistent: dipl. inž. Miloš Gligorić
Ispitni rok: Jun 2009.
Datum: 18.06.2009.

Kandidat:* _____

Broj Indeksa:* _____

Ispit traje 3 sata, prvih sat vremena nije dozvoljeno napuštanje ispita. Upotreba literature nije dozvoljena.

Zadatak 1	_____/8	*Zadatak 5*	_____/8
Zadatak 2	_____/8	*Zadatak 6*	_____/8
Zadatak 3	_____/5	*Zadatak 7*	_____/8
Zadatak 4	_____/6	*Zadatak 8*	_____/9

Ukupno na ispitu: **_____/60** ***Ukupno na projektu*:*** **_____/45**

Rok u kome je odbranjen projekat*: **_____** (pr: jun 2009)

Ukupno: _____/105

Ocena: _____ (_____)

Napomena: Ukoliko u zadatku nešto nije dovoljno precizno definisano, student treba da uvede razumnu pretpostavku, da je uokviri (da bi se lakše prepoznala prilikom ocenjivanja) i da nastavi da izgrađuje preostali deo svog odgovora na temeljima uvedene pretpostavke. Kod pitanja koja imaju ponuđene odgovore treba **samo zaokružiti** jedan odgovor. Na ostala pitanja odgovarati **čitko, kratko i precizno**. * popunjava student.

1. [8] Napisati fragment rekurzivnog parsera za prepoznavanje smene:

   $<A>_{x,y,z1} \rightarrow a_q \{c_v\} <A>_{x1,z,u1} <A>_{u,t,y1}$

   $v \leftarrow x - q \quad x1 \leftarrow x \quad y \leftarrow y1 \quad z1 \leftarrow z \quad u \leftarrow 3$

   Napomena: minimizovati broj naredbi dodele vrednosti u kodu.

2. [8] Objasniti naredbe ENTER i LEAVE kod x86 procesora (za korišćenje u Cu, bez pristupa nelokalnim promenljivim).

3. [5] Nacrtati dijagram klasa objektno orijentisane implementacije konačnog automata sa dva stanja A i B prema projektnom uzorku State. Objasniti dijagram.

4. [6] Data je desno-linearna gramatika sa startnim simbolom <M>:

1. <M> → ab<R>
2. <M> → ac
3. <R> → <M>
4. <R> → ε

a) [4] Transformisati datu gramatiku u regularnu gramatiku. Navesti pravila koja se primenjuju pri transformaciji i jasno naznačiti u kom koraku se primenjuje koje pravilo.
b) [2] Konstruisati deterministički automat koji odgovara regularnoj gramatici dobijenoj u a).

5. [8]

c) [2] Napisati gramatiku koja opisuje niz deklaracija celobrojnih promenljivih u Javi (svaka deklaracija je oblika **`int`** `ID {, ID};`).

d) [3] Dodati atribute i akcione simbole u gramatiku iz tačke a) tako da se dobije atributivno-translaciona gramatika koja za svaku deklaraciju promenljivih ispisuje broj promenljivih u deklaraciji.

e) [3] Prikazati stablo izvodjenja i vrednosti atributa u neterminalima za ulaz:

```
int a; int b, c; int r;
```

6. [8] Za programski fragment:

```
for (i = min, j = max; i < len; i++, j--) {
  e = *p;
  ar[j] = e;
  if (i == j) {
    q = ar[j];
    x = a[i];
  }
  e = 0;
}
x++;
```

a) [4] Napisati troadresni kod.
b) [4] Prikazati implementaciju troadresnog koda iz tačke a) putem četvorki i trojki.

7. [8] Dat je sledeći deo programa na C-u.

```
int f2(void) {
     int a;
     a = f3('a');
     return a;
}
int f3(char h) {
     int a = 5;
     a = a + h;
     return a;
}
void f1(int a, int b, int c) {
     int d, r;
     d = a + b;
     f2();
}
void main() {
     f1(1, 2, 3);
}
```

c) [3] Nacrtati izgled steka pre prvog povratka iz funkcije.
d) [5] Napisati 80x86 asemblerski potprogram ekvivalentan datom programu. NE koristiti enter i leave.

8. [9] U jednoj jednostavnoj implementaciji MJ interpretera memorijske oblasti opisane su unutar apstraktne klase Instruction, prikazane sledećim Java kodom (isečak):

```
public abstract class Instruction {
  /* oblast memorije koja sadrzi kod metoda */
  public static byte[] code;
  /* indeks instrukcije koja se izvrsava */
  public static int pc;

  /* staticki/globalni podaci glavnog programa */
  public static int[] staticData;

  /* dinamicki alocirani objekti i nizovi */
  public static int[] heap;
  /* pokazivac na pocetak slobodnog prostora */
  public static int free;

  /* procesorski stek */
  public static int[] procStack;

  /* stek izraza */
  public static int[] exprStack;
  /* indeks poslednje zauzete lokacije exprStack steka */
  public static int esp;

  public abstract void execute();

}
```

Za svaku MJ bytecode instrukciju postoji odgovarajuća podklasa koja implementira promenu stanja MJ VM (u skladu sa instrukcijom), kada se pozove metoda execute.

Potrebno je izvršiti interpretiranje (odgovarajuću promenu stanja) za sledeće instrukcije: *getstatic s* i *aload*. Osnova odgovarajućih klasa dat je u nastavku:

```
public class GETSTATIC extends Instruction {
  // operand instrukcije getstatic s
  private char s;

  // --- IMPLEMENTACIJA ---



}

public class ALOAD extends Instruction {
  // --- IMPLEMENTACIJA ---




}
```

Predpostaviti da su sve memorijske oblasti inicijalizovane kao i data polja (operandi) prikazanih klasa.

Napomena: Detaljno prokomentarisati kod.