**Elektrotehnički fakultet u Beogradu**

Katedra za računarsku tehniku i informatiku

Softversko inženjerstvo

*Nastavnik: dr Dragan Bojić, vanr. prof.*

*Asistenti: dipl. ing. Nemanja Kojić, dipl. ing. Maja Vukasović*

*Školska: 2017/2018.*

*Ispitni rok: Januarsko-februarski ispitni rok*

**Programski prevodioci 1**

Projekat – Kompajler za Mikrojavu –

**Andrija Boricic 0645/2014**

27.05.2018.

Cilj projektnog zadatka je realizacija kompajlera za programski jezik Mikrojavu. Kompajler omogućava prevodjenje sintaksno i semantički ispravnih Mikrojava programa u Mikrojava bajtkod koji se izvršava na virtuelnoj mašini za Mikrojavu.

Programski prevodilac za Mikrojavu ima četiri osnovne funkcionalnosti: leksičku analizu, sintaksnu analizu, semantičku analizu i generisanje koda.

**Leksički analizator** treba da prepoznaje jezičke lekseme i vrati skup tokena izdvojenih iz izvornog koda, koji se dalje razmatraju u okviru sintaksne analize. Ukoliko se tokom leksičke analize detektuje leksička greška, potrebno je ispisati odgovarajuću poruku na izlaz.

**Sintaksni analizator** ima zadatak da utvrdi da li izdvojeni tokeni iz izvornog koda programa mogu formiraju gramatički ispravne sentence. Tokom parsiranja Mikrojava programa potrebno je na odgovarajući način omogućiti i praćenje samog procesa parsiranja na način koji će biti u nastavku dokumenta detaljno opisan. Nakon parsiranja sintaksno ispravnih Mikrojava programa potrebno je obavestiti korisnika o uspešnosti parsiranja. Ukoliko izvorni kod ima sintaksne greške, potrebno je izdati adekvatno objašnjenje o detektovanoj sintaksnoj grešci, izvršiti oporavak i nastaviti parsiranje.

**Semantički analizator** se formira na osnovu apstraktnog sintaksnog stabla koje je nastalo kao rezultat sintaksne analize. Semantička analiza se sprovodi implementacijom metoda za posećivanje čvorova apstraktnog sintaksnog stabla. Stablo je formirano na osnovu gramatike implementirane u prethodnoj fazi. Ukoliko izvorni kod ima semantičke greške, potrebno je prikazati adekvatnu poruku o detektovanoj semantičkoj grešci.

**Generator koda** prevodi sintaksno i semantički ispravne programe u izvršni oblik za odabrano izvršno okruženje Mikrojava VM. Generisanje koda se implementira na sličan način kao i semantička analiza, implementacijom metoda koje posećuju čvorove apstraktnog sintaksnog stabla.

KOMANDE

1. Generisanje klase leksičkog analizatora **(Yylex.java)**

*java -cp lib\JFlex.jar JFlex.Main -d src\ba140645\mjcompiler spec\mjlexer.flex*

Na osnovu specifikacionog fajla mjlexer.flex u folderu src\ba140645d\mjcompiler generiše se klasa Yylex.java koja predstavlja izvorni kod leksičkog analizatora.

1. Generisanje klase LALR(1) parsera **(Yylex.java)** i klase za simbole **(sym.java)**, kao i apstraktnog sintaksnog stabla podržanog ast dodatkom za CUP parser **(klase u paketu ba140645d\mjcompiler\ast)**

**cd src;**

**java –jar ..\lib\cup\_v10k.jar –ast ba140645d.mjcompiler.ast –parser Parser –buildtree ..\spec\mjparser.cup**

Na osnovu specifikacionog fajla mjparser.cup u folderu ba140645d\mjcompiler generiše se klasa MJParser.java koja predstavlja izvorni kod parsera, u istom folderu generiše se i klasa sym.java, dok se u folderu ba140645d\mjcompiler generišu klase apstraktnog sintaksnog stabla na osnovu pomenute specifikacije. U folderu gde se nalazi specifikacija, kreiraće se dodatna ast specifikacija u fajlu mjparser\_astbuild.cup

1. Sama implementacija glavnog programa prevodioca koji uključuje sve pomenute module rešenja nalazi se u folderu **ba140645d\mjcompiler** i nazvana je **Compiler.java**. Sastoji se od niza konsekutivnih koraka koji prvo pokreću prethodno izgenerisani parser, kome kao argument zadaje klasu leksičkog analizatora. Za semantičku analizu koristi klasu iz istog paketa **SemanticAnalyzer.java**, dok za generisanje koda koristi klasu **CodeGenerator.java**

*java –cp src; lib\cup\_v10k.jar; lib\JFlex.jar; lib\log4j-1.2.17.jar; lib\symboltable.jar; lib\mj-runtime.jar rs.ac.bg.etf.pp1.Compiler test/program.mj test/program.obj*

Pokretanje kompajlera, argumenti komandne linije su izvorni kod na jeziku mikrojava koji treba prevesti, kao i objektni fajl u koji treba smestiti rezultat prevođenja.

1. Prevedeni kod se iz objektnog fajla može proveriti korišćenjem disaasembliranja, odnosno klase **rs.etf.pp1.mj.runtime.disasm**

*java -cp lib\mj-runtime.jar rs.etf.pp1.mj.runtime.disasm test\program.obj*

Pokretanje disasemblera, argument komandne linije je objektni fajl sa prevedenim kodom koji treba disasemblirati.

1. Prevedeni program se može pokrenuti na mikrojava virtuelnoj mašini korišćenjem korišćenjem klase **rs.etf.pp1.mj.runtime.Run** koja podiže izvršno okruženje mikrojava virtuelne mašine u okviru koje pokreće prevedeni program.

*java –jar lib\mj-runtime.jar rs.etf.pp1.mj.runtime.Run test\program.obj*

Pokretanje programa, argument komandne linije je objektni fajl sa prevedenim kodom koji treba izvršiti u okviru mikrojava virtuelne mašine.

**Svi navedeni koraci podrazumevaju koreni direktorijum projekta kao tekući radni direktorijum u kojem se pokreću navedene komande.**

Testovi

Postoji ukupno 10 testova, od kojih su 2 za leksicku analizu, 2 za sintaksnu i preostali za semanticku analizu.

Svaki od navedenih testove se moze pokrenuti preko MJTest klasa, u zavisnosti od toga koja funkcionalnost se proverava.

Test klase primaju kao argument putanju do test fajla.