

Elektrotehnički fakultet u Beogradu,

Katedra za računarsku tehniku i informatiku

Programski prevodioci 1 (13S114PP1)

**Projekat**

**-Kompajler za Mikrojavu-**

*Nastavnik:*

prof. dr Dragan Bojić

*Asistenti:*

mast. inž. Maja Vukasović

mast. inž. Kristijan Žiža

*Student:*

Martina Marković, 0672/17

*Januarsko-februarski ispitni rok, školska 2020/2021.*

**Opis postavke zadatka**

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 da formiraju gramatički ispravne sentence. Tokom parsiranja Mikrojava programa potrebno je na odgovarajući način omogućiti i praćenje samog procesa parsiranja. Nakon parsiranja sintaksno ispravnih Mikrojava programa potrebno je obavijestiti korisnika o uspješnosti 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 posjećivanje 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 posjećuju čvorove.

**Opis komandi**

Klasa *Yylex.java* koja predstavlja izvorni kod leksičkog analizatora se generiše na osnovu specifikacionog fajla *mjlexer.flex* dodavanjem novog target-a u *build.xml* fajl:

<target name="lexerGen" depends="delete">

<java jar="lib/JFlex.jar" fork="true">

<arg value="-d"/>

<arg value="./src/rs/ac/bg/etf/pp1"/>

<arg value="spec/mjlexer.flex"/>

</java>

</target>

Klasa *MJParser.java* koja predstavlja izvorni kod parsera se generiše na osnovu specifikacionog fajla *mjparser.cup* dodavanjem novog target-a u *build.xml* fajl:

<target name="parserGen" depends="delete">

<java jar="lib/cup\_v10k.jar" fork="true">

<arg value="-destdir"/>

<arg value="src/rs/ac/bg/etf/pp1"/>

<arg value="-ast"/>

<arg value="src.rs.ac.bg.etf.pp1.ast"/>

<arg value="-parser"/>

<arg value="MJParser"/>

<arg value="-dump\_states"/>

<arg value="-buildtree"/>

<arg value="spec/mjparser.cup"/>

</java>

</target>

Klasa SemanticAnalyzer*.java* se koristi za semantičku analizu,a klasa *CodeGenerator.java* se koristi za generisanje koda.

Pokretanjem main metode klase *Compiler.java* i zadavanjem dva argumenta koji predstavljaju fajl iz kog se čita program napisan na MikroJavi i izlazni .obj fajl izvršavaju se sve četiri osnovne funkcionalnosti prevodioca.

Korišćenjem klase *disasm* iz mj-runtime biblioteke dobijeni izlazni .obj fajl sa prevedenim kodom se može disasemblirati korišćenjem novog target-a u *build.xml* fajlu:

<target name="disasm">

<java classname="rs.etf.pp1.mj.runtime.disasm">

<arg value="test/program.obj"/>

<classpath>

<pathelement location="lib/mj-runtime-1.1.jar"/>

</classpath>

</java>

</target>

Korišćenjem klase *Run* iz mj-runtime biblioteke prevedeni program napisam na MikroJavi se može pokrenuti na MikroJava virtuelnoj mašini unosom sljedećih komandi u terminal :

java –jar lib\mj-runtime-1.1.jar rs.etf.pp1.mj.runtime.Run test\program.obj

**Test primjeri:**

*1. Testiranje rada implementiranog parsera:*

Skup testova sintaksno ispravnih i neispravnih programa u cilju testiranja oporavka od grešaka.

*2. Testiranje rada implementiranog semantičkog analizatora:*

Skup testova sintaksno i semantički ispravnih programa u cilju testiranja implementiranih smjena iz gramatike, kao i neispravnih programa koji sadrže kombinacije semantičkih grešaka.

*3. Javni testovi*

**Novouvedene klase:**

*1. sym\_old.java – leksička analiza*

*2. SemanticAnalyzer.java – semantička analiza*

*3. CounterVisitor.java – obilazak sintaksnog stabla*

*4. CodeGenerator.java – generisanje koda*

*5. CustomDumpSymbolTableVisitor.java – klasa koja proširuje DumpSymbolTableVisitor zbog ispisa bool tipa u tabeli simbola*

*6. MJTest.java – klasa za pokretanje leksičke analize*

*7. Compiler.java – klasa za pokretanje sve četiri faze prevodioca*