**Implementacija kompajlera**

Programski prevodioci

Elektrotehnički fakultet

Autor

Nevena Cvetković 0266/2016

# Postavka zadka

Zadatak ovog domaćeg je da odredjeni program na jeziku MicroJava prevede i uspešno pokrene njegovo izvršavanje na MicroJava virtuelnoj mašini.

Program uzima fajl program.mj i prosledjuje ga lexeru koji vrši leksičku analizu. Lexer kao rezultat vraća detektovane tokene.

Dobijeni tokeni se dalje prosledjuju klasi SemanticPass koja implementira klasu VisitorAdaptor generisanu od strane cup alata. U ovoj fazi se vrši semantička analiza dobijenog koda.

Na kraju se u klasi CodeGenerator obilazi apstraktno stablo semantički ispravnih tokena i na osnovu istog generiše programski kod koji se upisuje u fajl program.obj.

Dobijeni fajl se prosledjuje kao argument alatu koji vrši njegovo dekompajliranje i potom se izvršava liniju po liniju.

# Pokretanje aplikacije

Za build aplikacije se koristi alat ant. Kako bi ant radio neophodno je podesiti sledeće:

1. Search: Edit the system environment variables -> dugme Environment Variables
2. U dobijenom prozoru u User delu treba dodati promenljive ANT\_HOME koja ukazuje na bin folder mesta gde se nalazi ant i JAVA\_HOME koja ukazuje na bin folder instalacije jave
3. U System delu treba u okviru promenljive Path dodati putanje do bin foldera jave i ant-a

Svi pojedinačni alati koji se koriste za generisanje koda se pokreću iz okruženja Eclipse.

U okviru projekta postoji fajl build.xml koji je zadužen za pokretanje svih pojedinačnih alata koji su potrebni za generisanje klasa i izvršavanje koda.

Svaki pojedinačni target je zadužen za odgovarajući alat. Željeni target se može izabrati desnim klikom na build.xml Run As->Ant Build...

Prvo treba štiklirati target compile koji je i podrazumevani target. Ovim se pokreće CUP alat koji generiše LALR(1) gramatiku na osnovu našeg .cup fajla. Kao rezultat se dobija klasa symbol.java kao i klasa MJParser.java i paket rs.bg.ac.etf.pp1.ast u kom se nalaze klase koje predstavljaju pojedinačne delove svake smene.

Potom je potrebno štiklirati i target lexerGen koji pravi lexer na osnovu klase symbol.java i .flex fajla. Kao rezultat se dobija klasa Yylex.java.

Kako bi se pokrenulo generisanje .obj fajla treba pokrenuti main metodu klase Compiler. Kao argumente treba proslediti putanje do program.mj fajla koji sadrži kod koji treba prevesti i izvršiti i do program.obj fajla koji će nastati kao rezultat.( test\program.mj test\program.obj)

Na kraju kako bi se .obj fajl dekompajlirao i izvršio treba pokrenuti runObj target ili pozvati main metodu Run klase iz biblioteke mj-runtime.jar.

Može se podesiti da se konzola ispisuje u fajl. Desni klik na projekat Run As->Run Configurations, tab Common u delu Standard input and output štiklirati Output i navesti putanju do fajla.