Elektrotehnički fakultet Univerziteta u Beogradu

Katedra za računarsku tehniku i informatiku

Projekat

– Kompajler za Mikrojavu –

Predmet: Programski prevodioci 1

Student: Bogdana Veselinović

Profesor: dr Dragan Bojić

Broj indeksa: 375/2012

Asistent: Nemanja Kojić

# Opis 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. Sintaksno i semantički ispravni Mikrojava programi su definisani specifikacijom.

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 dobija proširenjem funkcionalnosti sintaksnog analizatora. Semantička analiza se vrši sprovodi kroz sintaksno-upravljano prevođenje. Osnovnoj gramatici, kojom je specificiran sintaksni analizator, dodaju se atributi i akcije (atributivno-translaciona gramatika).
* **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, kroz nadogradnju sintaksnog analizatora (sintaksno upravljano prevođenje).

# Opis dodatnih klasa

# Opis komandi za prevođenje

Spisak run konfiguracija u Eclipsu za prevođenje programa. Navedene su main klase i argumenti konfiguracije.

Potrebno je da se test fajlovi nalaze u direktorijumu test/test\_files/ u odnosu na root direktorijum projekta i njihova imena prilikom pokretanja se navode bez ekstenzije .mj. Output .obj fajlovi će biti generisani u direktorijumu test/output\_files/ u odnosu na root direktorijum projekta i imaće isti naziv kao i .mj fajl na osnovu koga su generisani.

* **Lexer generator** main klasa: jFlex.Main

argumenti: -d src\compiler spec\mjlexer.flex

Izveštaj jflex-a:

Reading "spec\mjlexer.flex"

Constructing NFA : 201 states in NFA

Converting NFA to DFA :

....................................................................................................................

120 states before minimization, 108 states in minimized DFA

Old file "src\compiler\Yylex.java" saved as "src\compiler\Yylex.java~"

Writing code to "src\compiler\Yylex.java"

* **Lexer test** main klasa: compiler.MJTest

argumenti: test#

* **Parser generator** main klasa: java\_cup.Main

argumenti: -destdir src\compiler

-parser MJParser spec\mjparser.cup

Izveštaj cup-a:

------- CUP v0.11a beta 20060608 Parser Generation Summary -------

0 errors and 0 warnings

45 terminals, 72 non-terminals, and 147 productions declared,

producing 237 unique parse states.

0 terminals declared but not used.

0 non-terminals declared but not used.

0 productions never reduced.

0 conflicts detected (0 expected).

Code written to "MJParser.java", and "sym.java".

---------------------------------------------------- (v0.11a beta 20060608)

* **Parser test** main klasa: compiler.MJParserTest

argumenti: test#

´

* **Pokretanje VM** main klasa: rs.etf.pp1.mj.runtime.Run

argumenti: test/output\_files/test#.obj

* **Disasm** main klasa: rs.etf.pp1.mj.runtime.disasm

argumenti: test/output\_files/test#.obj

* **Compiler** main klasa: compiler.Compiler

argumenti: test#

# Pokretanje iz komandne linije

Potrebno je exportovati konfiguracije u izvršne .jar fajlove (sa ili bez parametara) i smestiti ih u root direktorijum projekta. U komandnoj liniji locirati se u direktorijum gde se nalazi .jar fajl.

* **Lexer test** (od klase compiler.MJTest)

java –jar mjtest.jar test#

[ > test/test\_results/test#.out 2> test/test\_results/test#.err]

* **Parser test** (od klase compiler.MJParserTest)

java –jar mjparsertest.jar test#

[ > test/test\_results/test#.out 2> test/test\_results/test#.err]

* **Compiler** (od klase compiler.Compiler)

java –jar compiler.jar test#

[ > test/test\_results/test#.out 2> test/test\_results/test#.err]