Електротехнички факултет - Универзитет у Београду

Програмски преводиоци 1

Компајлер за Микројаву

Професор:

проф. Др Драган Бојић,

редовни професор

Студент:

Владимир Јанковић

2018/0121

Београд, Фебруар 2023.

Кратак опис поставке задатка

Циљ овог пројекта је конструкција програмског преводиоца за језик Микројаву (Микројава компајлер). Компајлер омогућава превођење синтаксно и семантички исправних Микројава програма у Микројава *bytecode* који се извршава на виртуелној машини за Микројаву. Сви Микројава програми садрже екстензију *.mj* у свом имену. Спецификација за програмски језик Микројава, дефинише синтаксну и семантичку исправност Микројава програма.

Компајлер се састоји из четири основне фазе:

- Лексичка анализа – скенирање и токенизација улаза
- Синтаксна анализа – формирање апстрактног синтаксног стабла
- Семантичка анализа – провера граматике улаза и ажурирање табеле симбола
- Генерисање кода – генерисање машинских инструкција за виртуелну машину

Команде за генерисање Јава кода

Спецификација лексичког анализатора се налази у фајлу са екстензијом *.flex* / *.lex* .Овај фајл је смештен на локацији *MJCompiler/src/spec/mjlexer.flex* . У фајлу *build.xml* се налази циљ за прављење лексера (*build.xml:lexerGen*) коришћењем алата *JFlex.jar* а излаз овог циља је класа *Yylex.java* у директоријуму *MJCompiler/src* и припада пакету *rs.ac.bg.etf.pp1*.

У синтаксној анализи, за конструкцију апстрактног синтаксног стабла је надлежан парсер који се конструише помоћу генератора синтаксних анализатора *AST-CUP*. Генератор је локално развијено проширење алата *CUP* за рад са синтаксним стаблима. Фајл са *.cup* екстензијом се налази на локацији *MJCompiler/src/spec/mjparser.cup*. У фајлу *build.xml* се налази циљ за прављење парсера (*build.xml:parserGen*) коришћењем алата *cup_v10k.jar* а излаз овог циља су:

- Класа *MJParser.java* – за конструкцију LALR(1) парсера за дату граматику из *mjparser.cup* fajla.
- Класа *sym.java* – ова класа садржи вредности токена за терминалне симболе које користи лексички анализатор (*Yylex.java*) у процесу токенизације.
- Јава класе које представљају чворове у апстрактном синтаксном стаблу, смештене у фолдеру *MJCompiler/src* и припадају пакету *rs.ac.bg.etf.pp1.ast*

Циљ *build.xml:repackage* се користи да би се у новонасталим класама пакети исправно преправили, што омогућава да класе буду видљиве другим класама које их користе.

Циљ *build.xml:compile* се користи за компајлирање свих јава класа у *MJCompiler/src* директоријуму. Циљеви су уланчани међусобном зависношћу и тиме би се покретањем једног циља покренуо ланац циљева у оквиру *build.xml* фајла:

delete -> lexerGen

delete -> parserGen -> repackage -> compile

Покретањем класе *Compiler.java* , на стандардни излаз се исписују токени генерисани од стране лексичког анализатора, апстрактно синтаксно стабло генерисано од стране парсера, детекција свих симбола и табела симбола на крају семантичке анализе. На самом крају извршавања се генерише објектни фајл, под претпоставком да се није наишло на грешку током лексичке, синтаксне и семантичке анализе, који се може извршавати над Микројава виртуелном машином. Компајлер извршава превођење фајла *program.mj*.

Циљ *build.xml:disasm* служи да се машински код у објектном фајлу претвори у људски читљив облик у виду асемблерских инструкција и испише на стандардни излаз.

Циљ *build.xml:debug* служи да се на стандардни излаз прикаже стање стека након извршавања сваке наредне инструкције.

Циљ *build.xml:runObj* служи да се објектни фајл изврши на Микројава виртуелној машини. Ови циљеви су међусобно зависни па се покретањем циља *runObj* покрећу и претходна два:

disasm -> debug -> runObj

Тестирање рада компајлера

У наставку се налазе описи приложених тест примера Микројава кода, који служе за тестирање исправности рада компајлера. Сви тестови су смештени у оквиру директоријума *MJCompiler/test/my_tests*.

Тестови за синтаксну анализу – опоравак од грешке:

- *sin_error_test1.mj* – опоравак од грешке при дефиницији глобалне променљиве
- *sin_error_test2.mj* – опоравак од грешке код конструкције исказа доделе
- *sin_error_test3.mj* – опоравак од грешке код декларације формалног параметра функције
- *sin_error_test4.mj* – опоравак од грешке код логичких израза унутар *if (else)* конструкције

Тестови за семантичку анализу:

- *semantic_errors.mj* – тестира све могуће семантичке грешке
- *test01.mj, test02.mj, test03.mj, test04.mj* – тестови који су се користили током израде компајлера ради провере исправности одређених семантичких услова.
- *test301.mj* – јавни тест за А ниво, који проверава исправан рад компајлера код:
  - Декларација глобалних променљивих и дефиниција константи
  - Декларација низова примитивних типова
  - Коришћење примитивних типова (*int, char, bool*)
  - Дефиниција функције *void main()* од које почиње извршавање програма
  - Декларација локалних променљивих
  - Исказ доделе вредности, аритметички изрази
  - Дефинисање новог низа и дохватање елемента низа
  - Додела вредности већем броју елемената помоћу низа
  - Коришћење функција *print* и *read*
- *test302.mj* – јавни тест за Б ниво, који садржи све из А нивоа и додатно тестира:
  - Дефиниција глобалних функција са опционим формалним параметрима
  - Позив глобалне функције као исказ или у оквиру неког израза
  - Условна гранања *if (else)*, условне петље *while* и итерирање по низовима *foreach* петљом
  - Наредба *break* за искакање из окружујуће петље
  - Наредба *continue* за прекидање текуће итерације и скок на наредну итерацију окружујуће петље
  - Наредба *return* за излаз из текуће функције са опционим враћањем повратне вредности
  - Позив глобалних функција *ord, chr* и *len*

Новоуведене класе

Поред главних јава класа за конструкцију компајлера које су кориснички дефинисане (*Compiler*, *SemanticAnalyzer*, *CodeGenerator*, *CounterVisitor*) и програмски генерисане (*Yylex*, *sym*, *MJParser*, класе чворова апстрактног синтаксног стабла) , имплементирана је једна помоћна класа *SymbolTablePrinter.java*.

Класа *SymbolTablePrinter* проширује класу *DumpSymbolTableVisitor* из библиотеке *symboltable-1-1.jar* и редефинише *visitObjNode(Obj)* методу да би се новоуведени тип *bool* могао исписати при испису табеле симбола на крају семантичке анализе. Унутар класе *SymbolTablePrinter* је дефинисана метода *tsdump()* која врши испис табеле симбола и позива се у класи *Compiler* пре почетка фазе генерисања кода.