Универзитет у Београду

Електротехнички факултет



**Програмски преводиоци 1**

**Компајлер за Микројаву**

Вукашин Недељковић 2018/0217

**Кратак опис поставке задатка**

Циљ пројектног задатка је реализација компајлера за програмски језик Микројаву. Компајлер омогућава превођење синтаксно и семантички исправних Микројава програма у Микројава бајткод који се извршава на виртуелној машини за Микројаву. Синтаксно и семантички исправни Микројава програм су дефинисани спецификацијом [MJ].

Програмски преводилац за Микројаву има четири основне функционалности: лексичку анализу, синтаксну анализу, семантичку анализу и генерисање кода.

Лексички анализатор треба да препознаје језичке лексеме и врати скуп токена издвојених из изворног кода, који се даље разматрају у оквиру синтаксне анализе. Уколико се током лексичке анализе детектује лексичка грешка, потребно је исписати одговарајућу поруку на излаз.

Синтаксни анализатор има задатак да утврди да ли издвојени токени из изворног кода програма могу формирају граматички исправне сентенце. Током парсирања Микројава програма потребно је на одговарајући начин омогућити и праћење самог процеса парсирања на начин који ће бити у наставку документа детаљно описан. Након парсирања синтаксно исправних Микројава програма потребно је обавестити корисника о успешности парсирања. Уколико изворни код има синтаксне грешке, потребно је издати адекватно објашњење о детектованој синтаксној грешци, извршити опоравак и наставити парсирање.

Семантички анализатор се формира на основу апстрактног синтаксног стабла које је настало као резултат синтаксне анализе. Семантичка анализа се спроводи имплементацијом метода за посећивање чворова апстрактног синтаксног стабла. Стабло је формирано на основу граматике имплементиране у претходној фази. Уколико изворни код има семантичке грешке, потребно је приказати адекватну поруку о детектованој семантичкој грешци.

Генератор кода преводи синтаксно и семантички исправне програме у извршни облик за одабрано извршно окружење Микројава ВМ. Генерисање кода се имплементира на сличан начин као и семантичка анализа, имплементацијом метода које посећују чворове.

**Генерисање кода**

Генерисање класе лексичког анализатора Yylex.java:

На основу спецификационог фајла mjlexer.flex генерише се класа Yylex.java која представља изворни код лексичког анализатора.

Класа Yylex.java генерише се стављањем следећег таргет-а у build.xml фајл:

<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.lex" />

</java>

</target>

Генерисање класе LALR(1) парсера и класе за симболе:

На основу спецификационог фајла mjparser.cup генерише се класа MJParser.java која представља изворни код парсера, у истом фолдеру генерише се и класа sym.java. У фолдеру где се налази .cup спецификација, креираће се додатна ast спецификација у фајлу mjparser\_astbuild.cup..

Генерисање се покреће додавањем следећег таргет-а у build.xml фајл:

<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="src/spec/mjparser.cup" />

</java>

</target>

Имплементација главног програма :

Сама имплементација главног програма преводиоца названа је Compiler.java. Састоји се од низа сукцесивних корака који прво подразумијевају покретање лексичке анализе, након чега иде синтаксна анализа. Затим слиеди семантичка анализа, за коју се користи класа из истог пакета SemanticAnalyzer.java, док се за генерисање кода користи класа CodeGenerator.java. Први аргумент main функције класе Compiler представља фајл из ког ће се читати микројава код а други аргумент представља излазни .obj фајл.

Дисасемблирање:

Преведени код се из објектног фајла може се дисасемблирати, коришћењем класе rs.etf.pp1.mj.runtime.disasm.

Дисасемблирање се покреће додавањем и засебним покретањем следећег таргет-а у build.xml фајл:

<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>

Покретање преведеног програма:

Преведени програм се може покренути на микројава виртуелној машини коришћењем класе rs.etf.pp1.mj.runtime.Run.

Покретање микројава програма се врши додавањем и засебним покретањем следећег таргет-а у build.xml фајл:

<target name="runObj" depends="disasm" >

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

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

<classpath>

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

</classpath>

</java>

</target>

**Тестови**

* test1-2.1.mj – Овај тест има лексичке и синтаксне грешке од којих врши опоравак
* test2.2.mj – Овај тест садржи све синтаксне грешке и врши се опоравак од свих синтаксних грешака
* test31.mj – Овај тест показује детектовање грешака из спецификације на страни 5 [MJ§A.4p5]
* test32.mj – Овај тест показује детектовање грешака из спецификације на страни 6 [MJ§A.4p6]
* test33.mj – Овај тест показује детектовање грешака из спецификације на страни 7 [MJ§A.4p7]
* test34.mj – Овај тест исписује детектовање: приступ елементу низа, глобалне променљиве, локалне променљиве, глобалне функције, рекорди и коришћење формалног аргумента функције
* testfaktorijel.mj – Овај тест рачуна и исписује факторијел унетог броја
* testnzd.mj – Овај тест рачуна и исписује НЗД два броја
* test30x.mj – Тестови са сајта предмета

**Ново уведене класе**

**TabExtend** – Унутар пакета rs.ac.bg.etf.pp1 и наслеђује класу Tab. Класа је потребна да би се додао bool тип у универзим, као и за потребан испис табеле симбола.

**DumpSymbolTableVisitorExtend** – Унутар пакета rs.ac.bg.etf.pp1 и наслеђује класу DumpSymbolTableVisitor. Класа служи за обраду bool типа.

**CounterVisitor** – Унутар пакета rs.ac.bg.etf.pp1 и наслеђује класу VisitorAdaptor. Класа која се користи за пребројавање локалних параметара функције и локалних променљиви функције.

**MJTest** – Унутар пакета rs.ac.bg.etf.pp1. Класа извршава лексичку анализу програма.

**MJParserTest** – Унутар пакета rs.ac.bg.etf.pp1. Класа извршава лексичку, синтаксну и семантичку анализу улазног програма program.mj и у случају успешног парсирања враћа program.obj који може да се покрене Микројава виртуалном машином.