

Електротехнички факултет, Универзитет у Београду

Катедра за рачунарску технику и информатику

**ПРОЈЕКАТ ИЗ ПРОГРАМСКИХ**

**ПРЕВОДИОЦА 1**

**-Компајлер за Микројаву-**

*Наставник:* др Драган Бојић, редовни професор

*Асистенти:* маст. инж. Маја Вукасовић, студент: Василија Рашић,

дипл. инж. Михајло Огризовић 0465/2018

*Школска:* 2021/2022

*Испитни рок:* Јануарско-фебруарски испитни рок

*Датум:* 26.01.2022.

**ОПИС ПРОЈЕКТНОГ 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 у виртуелној машини за Микројаву.

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 да врати скуп токена издвојених из изворног кода.

Синтаксни анализатор треба да утврди да ли токени добијени лексичком анализом представљају граматички исправне сентенце.

Семантичком анализом формирају се методе за посећивање чворова апстрактног синтаксног стабла формираног приликом синтаксне анализе.

Генерисање кода преводи синтаксно и семантичке исправне програме у извршни облик за одабрано окружење Микројава ВМ.

**КОМАНДЕ ЗА ГЕНЕРИСАЊЕ ЈАВА КОДА**

Приликом израде пројекта потребни су нам помоћни алати и библиотеке за генерисање кода који су нам неопходни за израду.

За генерисање и израду лексера, непходна нам је библиотека “JFlex.jar” која на основу нашег формираног лексера у ‘.lex ’ формату, командом : “-d <dst-path> <src-file>”, у дестинационом фолдеру генерише “Yylex.java” и “sym.java” фајлове, где “sym.java” фајл садржи мапиране токене у јава формату, а “Yylex.java” представља наш генерисани лексички анализатор, који спроводи лексичку анализу и враћа препознате токене.

За реализацију синтаксног анализатора потребна нам је библиотека “cup\_v10k.jar” као и AST-CUP генератор који омогућава рад са синтаксним стаблима. Како би се успешно реализовао наш синтаксни анализатор потребно је да он буде у складу са CUP спецификацијом. Команда која генерише наш синтаксни анализатор у Јава коду је: “-destdir <dst> -ast <dst-klase> -parser <ime-parsera> -buildtree <src> ”. Овом командом се такође формира и апстрактно синтаксно стабло као што и генерише класе на основу чворова стабла које су нам битне за даљу обраду пројекта.

Семантички анализатор се реализује као класа изведена из класе “VisitorAdaptor”, и врши се обиласком апстрактног синтаксног стабла које је реализовано синтаксном анализом. За израду семантичке анализе користимо и библиотеку “symboltable-1-1.jar” која нам пружа табелу симбола као и функције за убацивање типова, променљивих, метода, памћење њихових адреса, локалних променљивих и лакшу реализацију семантичке анализе. Посећивањем чворова синтаксног стабла семантичком анализом се формирају одређена правила у граматици који један програмски код треба да испоштује да би се адекватно превео.

Генерисање кода врши се обиласком апстрактног стабла које је настало као резултат синтаксне анализе и задовољио услове семантичке анализе. Његова имплементација се врши такође помоћу класе која је изведена из генерисане класе “VisitorAdaptor”, само што се на препознате чворове синтаксног стабла генерише одговарајуће инструкције које треба да се извршавају.

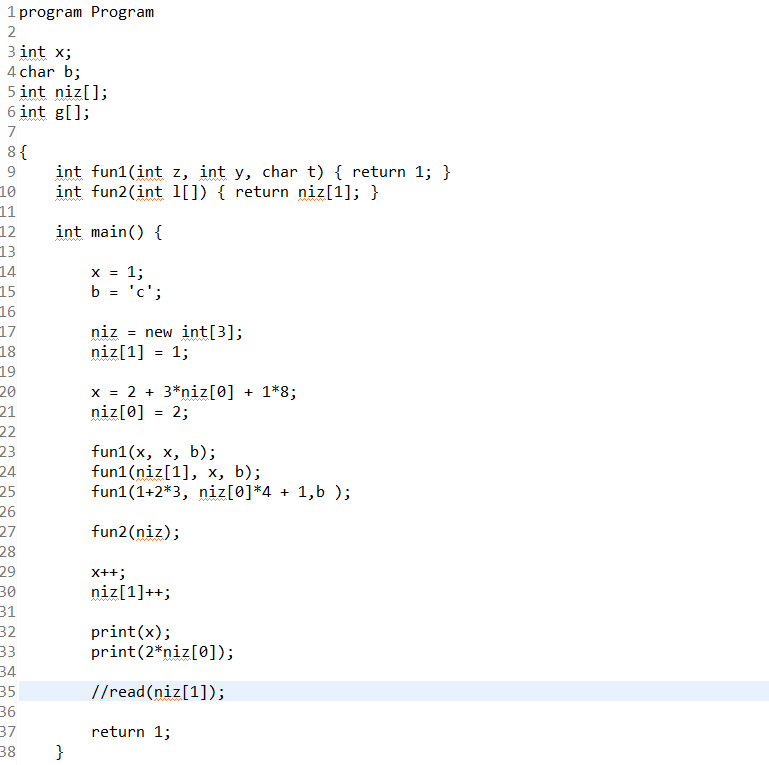
**ТЕСТИРАЊЕ**

За тестирање потребно је за почетак формирати обејкат “Yylex.java” који спороводи лексичку анализу над улазним фајлом. Након успешне лексичке анализе, формира се објекат парсера и врши се синтаксна анализа. Када се обави и синтаксна анализа, формира се објекат семантичког анализатора и над добијеним резултатом синтаксне анализе, врши се семантичка анализа обиласком чворова и уколико је успешна, формира се објекат класе која врши обилазак чворова и генерисање кода и генерише се код у одређени објектни фајл.

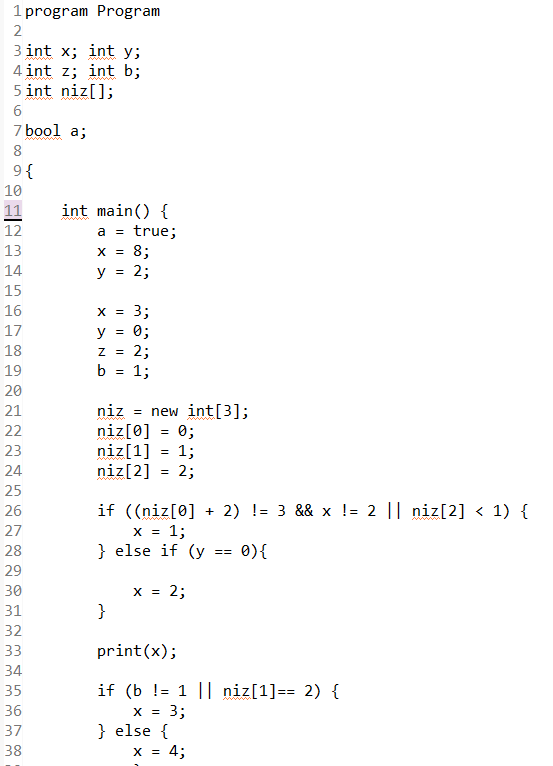
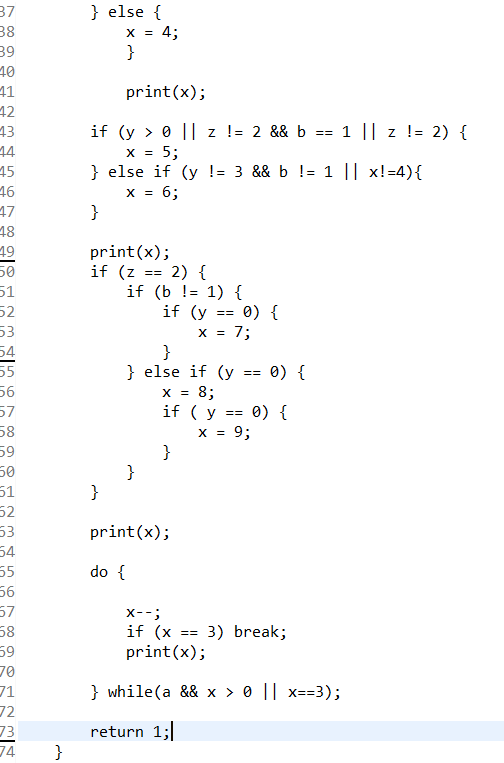
За потребе тестирања постоји и библиотека “mj-runtime-1.1.jar” која нам омогућава командом “disasm ” претварање објектног кода у асемблерски код, који је читљивији и употребљив за потребе дебаговања. Поред тога постоји и класа “Run” која извршава код улазног објектног фајла.

**ТЕСТ ПРИМЕРИ**

1. У овом тест примеру тестиране су основне операције множења, сабирања, додељивања вредности као и рад са низовима, прослеђивање параметара функцији који је елемент низа, и сам низ.



1. У следећем тест примеру циљ је био да се тестира рад са if-else гранама и do-while петљама, све што се тиче скокова унапред и препознавање одређених услова и њихово обрађивање.

1. У последње приложеном тесту, испробаван је рад са рекордима, приступање њиховим пољима, комбинација рекорда и низова.

