

**ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ**

**УНИВЕРЗИТЕТ У БЕОГРАДУ**

***КОМПАЈЛЕР ЗА МИКРОЈАВУ***

Програмски преводиоци 1

Марија Огњеновић,2018/0071

Јануар 2022.

# **Циљ пројекта**

Пројектни задатак „Компајлер за Микројаву“ за циљ је имао упознавање са основама програмских преводилаца и реализацију преводиоца, погодног за програмски језик Микројава. Основна функција оваквог преводиоца била је да омогући превођење синтаксно и семантички исправних програма у бајткод, који би се могао извршавати на виртуелној машини.

Програмски преводилац поседује четири основне функционалности:

* **Лексичку анализу**: Ова фаза подразумева имплементацију лексичког анализатора (**mjlexer.flex**) чији је задатак да препознаје језичке лексеме и врати скуп токена издвојених из изворног кода, који се даље разматрају у оквиру синтаксне анализе. Свака лексичка грешка до које дође, биће исписана на излаз.
* **Синтаксна анализа**: Ова фаза подразумева имплементацију синтаксног анализатора (**mjparser.cup**) чији је задатак да утврди да ли издвојени токени из изворног кода програма могу формирају граматички исправне реченице. Уколико дође до грешке, исписује се на којој линији се десила грешка и која.
* **Семантичка анализа**: Ова фаза подразумева имплементацију семантичког анализатора (**SemanticAnalyzer.java**) који 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 дође до грешке, исписује се на којој линији се десила грешка и која.
* **Генерисање кода**: Ова фаза подразумева имплементацију генератора кода (**CodeGenerator.java**) чији је задатак да преводи синтаксно и семантички исправне програме у извршни облик за одабрано извршно окружење Микројава Виртуелну Машину.

# **Како покренути компајлер?**

* Најпре је неопходно генерисати класу Yylex.java помоћу build targeta „LexerGen“ који се налази у build.xml фајлу која има за циљ да скенира улазни текст и токене.
* Направити класу MJParser.java помоћу build targeta “Parser Gen” који се налази у build.xml фајлу која представља парсер.
* Позвати buil target “repackage” који се налази у build.xml фајл чиме се премештају класе у одговарајући директоријум.
* Превести све класе заједно build targetom "compile", како би се добио .exe фајл за компајлер.
* Покренути програм помоћу build targeta “runObj” који се налази у build.xml фајлу. Уколико је потребно видети садржај користи се алат **disasm** који прима објектни фајл и исписује асемблерске наредбе које се налазе у објектном фајлу.
* Као додатна погодност постоји и циљ „debug“ уз чију помоћ се може пратити стање на стеку стања.

# **Тестови**

У оквиру фолдера тестови се налазе примери тестова.

* Тест1 тестира што је више могуће језичких конструкција
* Тест2 тестира опоравак од грешке у синтаксној анализи
* Тест3 тестира што је више могуће семантички исправних конструкција
* Тест4 тестира препознавање грешака у семнатици по редоследу којим су се догодиле
* Тест5 тестира генерисање кода и сто је више могуће наредби
* Тест301\_А представља јавни тест за ниво А.