Dušan Terzić 0664/18

Mikrojava kompajler

Programski prevodioci 1 (nivo A)

## Opis projekta

Projekat obuhvata pravljenje kompajlera za jezik MikroJava u sklopu predmeta Programski Prevodioci 1.

Projekat se sastoji iz četiri faze, a to su:

* Leksička analiza
* Sintaksna analiza
* Semantička analiza
* Generisanje koda

Leksička analiza obuhvata prepoznavanje simbola i pravljenje leksičkog analizatora. Detektuju se identifikatori, ključne riječi, komentari, konstante i operatori. U slučaju leksičke greške analizator je prijavljuje i nastavlja sa radom.

Sintaksna analiza obuhvata prepoznavanje određenih skupova simbola koji su nađeni u leksičkoj analizi i formiranje sintaksnog stabla. Radi se pomoću alata AST-CUP. Uključuje pronalazak grešaka i oporavak od njih.

Semantička analiza pronalazi semantičke greške u korišćenju sintaksnih struktura dobijenih u prethodnoj fazi (provjera konteksnih uslova). U ovoj fazi se obilaskom sintaksnog stabla formira tabela simbola pomoću koje se kasnije generiše kod.

Generisanje koda je poslednja faza projekta i u toj fazi se pomoću tabele simbola i sintaksnog stabla generiše bajt kod za MJVM (MikroJava Virtuelna Mašina). Izlaz generatora je izvršivi .obj fajl.

## Pokretanje projekta

Projekat se pokreće pomoću alata ANT. U xml fajlu su sadržane komande potrebne za pokretanje projekta. Imamo više targeta u xml fajlu, a to su:

* delete: brisanje prethodno izgenerisanih fajlova i klasa
* lexerGen: pokretanje leksičke analize projekta
* parserGen: generisanje parsera i sintaksnog stabla
* repackage: pomocni target za promjene paketa u klasama
* compile: kompajliranje koda
* disasm: pravljenje .obj fajla
* runObj: pokretanje .obj fajla

Rješenje se testira pokretanjem klase Compiler nad test primjerima.

## Testiranje

Testiranje obuhvata više ispravnih i neispravnih test primjera u jeziku mikrojava (fajlovi sa ekstenzijom .mj). Sadrži test primjer sa sajta predmeta (test301.mj) koji testira A nivo zadatka i obuhvata funkcionalnosti kao što su aritmetičke operacije, deklaracije promjenljivih, konstante, prostore imena, funkciju print i read. Test mytest1 obuhvata jos dodatnih aritmetičkih operacija. Test mytest2 sadrži leksičke greške i pokazuje ispis poruka o njima. Test mytest3 fokusira se na pregled sintaksnih grešaka u kodu i prikaz oporavka od istih grešaka, dok test mytest4 pronalazi semantičke greške i provjerava konteksne uslove.

## Nove klase

Nove klase koje su napravljene tokom pravljenja kompajlera:

* CodeGenerator – služi za generisanje koda
* SemanticAnalyzer – vrši semantičku analizu i formira tabelu simbola
* sym – generisana klasa koja sadrži nazive svih mogućih simbola
* Yylex – generisana klasa koja sadrži leksičke uslove
* MJParser – generisana klasa pomoću cup alata koja provjerava sintaksne uslove i formira stablo
* Klase paketa .ast – generisane klase za svaku smjenu u .cup fajlu
* Compiler – klasa koja pokreće kompajler nad mikrojava fajlom
* MySymbolTableVisitor – klasa koja redefiniše ispis tabele simbola iz biblioteke symboltable.jar

else if (al instanceof Sharp) { // a## => (a mod 2 == 0 ? a^2 : a^3)

if (d.getType().getKind() == Struct.Array) {

Code.put(Code.dup2);

Code.put(Code.aload);// 6

Code.put(Code.dup); // 6 6

Code.put(Code.dup); // 6 6 6

Code.loadConst(2); // 6 6 6 2

Code.put(Code.rem); // 6 6 0

Code.loadConst(0); // 6 6 0 0

Code.putFalseJump(Code.ne, 0); // 6 6 true

int adr = Code.pc - 2;

Code.put(Code.dup); // neparno 6 6 6

Code.put(Code.mul); // 6 36

Code.fixup(adr);

Code.put(Code.mul); // 36

Code.put(Code.astore);

} else {

Code.load(d); // 6

Code.put(Code.dup); // 6 6

Code.put(Code.dup); // 6 6 6

Code.loadConst(2); // 6 6 6 2

Code.put(Code.rem); // 6 6 0

Code.loadConst(0); // 6 6 0 0

Code.putFalseJump(Code.ne, 0); // 6 6 true

int adr = Code.pc - 2;

Code.put(Code.dup); // neparno 6 6 6

Code.put(Code.mul); // 6 36

Code.fixup(adr);

Code.put(Code.mul); // 36

Code.store(d); // store

}

public void visit(DolarDesignator f) { // niz = { 1, 2, 3}

Code.load(f.getDesignator().obj); // 2

Code.put(Code.dup); // 2 2

Code.put(Code.dup); // 2 2 2

Code.put(Code.arraylength); // 2 2 3

Code.loadConst(0); // 2 2 3 0

Code.loadConst(Code.dup2); // 2 2 3 0 3 0

Code.putFalseJump(Code.ne, 0); // 2 2 3 0

int adr = Code.pc - 2;

Code.put(Code.dup\_x2); // 2 0 2 3 0

Code.put(Code.dup\_x1); // 2 0 2 0 3 0

Code.put(Code.pop); // 2 0 2 0 3

Code.put(Code.dup\_x2); // 2 0 3 2 0 3

Code.put(Code.pop); // 2 0 3 2 0

Code.put(Code.aload); // 2 0 3 1

Code.put(Code.dup\_x2); // 2 1 0 3 1

//...

}

**public** **void** visit(ModMax modMax) {

Designator d = modMax,getDesignatorNizModif().getDesignator();

Obj o = d.obj;

**int** kind = d.obj.getType().getElemType().getKind();

Code.loadConst(0);

**if**(kind == 2)

Code.put(Code.baload);

**else** Code.put(Code.aload);

//stek: nulti El.

Code.loadConst(0);

**int** pocetakLoopa = Code.pc;

//krece loop

Code.put(Code.dup); //max 0 0

Code.load(o);

Code.put(Code.arraylength); //max 0 0 duzina

Code.loadConst(1);

Code.put(Code.sub);

**int** krajSkok = Code.mainPc + 1;

Code.putFalseJump(Code.lt, 0); //dosli smo do kraja

Code.put(Code.dup2);//max 0 max 0

Code.loadConst(1);

Code.put(Code.add); //max 0 max 1

Code.load(o);

Code.put(Code.dup\_x1);

Code.put(Code.pop); //max 0 max niz 1

Code.put(Code.aload);

//max 0 max next

Code.put(Code.dup2); //max 0 max next max next

**int** newNotBigger = Code.pc + 1;

Code.putFalseJump(Code.lt, 0); //ako je max veci

//zamjena

//max 0 max nextElem

Code.put(Code.dup\_x2);

Code.put(Code.pop);//max nextElem 0 max

Code.put(Code.pop);

Code.loadConst(1);

Code.put(Code.add); //max nextElem currIter

Code.put(Code.dup\_x2);

Code.put(Code.pop);//currIter max nextElem

Code.put(Code.dup\_x2);

Code.put(Code.pop); //nextElem currIter max

Code.put(Code.pop);

**int** skokNaLoop = Code.pc+1;

Code.putJump(0);

**int** tmp = Code.pc;

Code.pc = pocetakLoopa;

Code.fixup(skokNaLoop);

Code.pc = tmp;

//ostaje stari max

Code.fixup(newNotBigger);

Code.put(Code.pop);

Code.put(Code.pop);

Code.loadConst(1);

Code.put(Code.add); //max nextiter

**int** skokNaLoop2 = Code.pc + 1;

Code.putJump(0);

tmp = Code.mainPc;

Code.pc = pocetakLoopa;

Code.fixup(skokNaLoop2);

Code.pc = tmp;

Code.fixup(krajSkok);

**if**(kind == 2)Code.put(bprint);

**else** Code.put(Code.print);

}