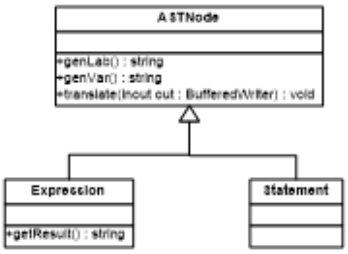
Zadatak 1

Tekst zadatka

ELEKTRONSKI FAKULTET U NIŠU
Katedra za računarstvo *Jun 2008.*

II kolokvijum iz Programskih prevodilaca
I grupa

1. Hijerarhija klasa za predstavljanje čvorova u apstraktnom sintaksnom stablu data je na sledećoj slici:



Dodati klasu za predstavljanje for-petlje u apstraktnom sintaksnom stablu, ako je for-petlja definisana sledećom smenom:

ForStatement → **FOR ID**=*Expression* **TO** *Expression* **DO** *Statement*

Pri čemu je navedeni identifikator broč petlje, prvi navedeni izraz je inicijalna vrednost brojača, a drugi njegova konačna vrednost. Prilikom svakog prolaska kroz petlju vrednost brojača se uvećava za 1.

Definisati zapis for-petlje u medjukodu niskog nivoa i u definisanoj klasi implementirati funkciju *translate* koja čvor AST-a konvertuje u medjukod niskog nivoa u kojem su definisane sledeće komande:

Instrukcija	**Značenje**
`Load_Const Rn, c`	(Rn) = c
`Load_Mem Rn, x`	(Rn) = x
`Store Rn, x`	(x) = (Rn)
`Add Rn,Rm`	(Rn) = (Rn) + (Rm)
`Sub Rn,Rm`	(Rn) = (Rn) - (Rm)
`Mul Rn,Rm`	(Rn) = (Rn) * (Rm)
`Div Rn,Rm`	(Rn) = (Rn) / (Rm)
`Jump lab`	skok na naredbu sa oznakom lab
`JumpIfZero Rn, lab`	skok na naredbu sa oznakom lab ukolikoko je (Rn)=0

Strategija generisanja međukoda

```
IMC <START EXPRESSION>
IMC <END EXPRESSION>
LoadMem R1, Res<START EXPRESSION>
//Zbog čuvanja vrednosti
Store R1,X
Load R2, Res<END EXPRESSION>
//Petlja
loop:
```

```
Compare R1, R2
//Skoci na kraj kada postanu jednaki
JumpIfNotZero R1, kraj
IMC<STATEMENT>
//Zbog toga sto Compare menja vrednost R1
LoadMem R1,X
Add R1,1
Store R1, X
Jump loop
kraj:
```

Java kod

```
public class ForStatement extends Statement{
private Expression starte, ende;
private Statement s;
public vodi translate(BufferedWriter out)
{
String loop=ASTNode.genLab();
String kraj=ASTNode.genLab();
starte.translate(out);
ende.transalte(out);
String current_value=ASTNode.genVar();
starte.genLoad("R1", out);
out.write("Store R1,"+current_value);
ende.genLoad("R2", out);

out.write(loop+":");

out.write("Compare R1,R2");
out.write("JumpIfNotZero R1,"+kraj);
s.translate(out);
out.write("LoadMem R1,"+current_value);
out.write("Add R1,1");
out.write("Store R1,"+current_value);
out.write("Jump"+ loop);

out.write(kraj+":");

}
}
```

Zadatak 2

Tekst zadatka

(*6 poena*) Petlja je u jednom programskom jeziku definisana sledećom smenom:

LoopStatement → **loop** *StatementList* **end loop**

Unutar petlje mora da postoji bar jedna exit naredba koja je definisana smenom:

ExitStatement → **exit when** *Expression* **;**

Definisati klase za predstavljanje ovih naredbi u AST-u. Definisati međukodove niskog nivoa za izvršavanje ovako definisane petlje i implementirati metode za generisanje međukoda.

Strategija generisanja međukoda

```
pocetak:
IMC <STATEMENT1>
...
IMC <STATEMENTk-exit statement>
--------------------------
IMC <STATEMENTk.exit>
LoadMem R1, RESULT<STATEMENTk.exit>
JumpIfNotZero R1, kraj
--------------------------
...
IMC <STATEMENTi>
...
IMC <STATEMENTn>
Jump pocetak
kraj:
```

Java kod

```
public class ExitStatement extends Statement{

	private Expression exit;

	public void translate(BufferedWriter out) throws IOException {

		exit.translate(out);

		genLoad("R1", out);

	}

}
```

```
public class Loop extends Statement{

	private ArrayList<Statement> sl;
```

```
	public void translate(BufferedWriter out) throws IOException{
		String pocetak=ASTNode.genLab();
		String kraj=ASTNode.genLab();
		out.write(pocetak+":");
		out.newLine();
		for(int i=0;i<sl.size();i++){
			Statement current=sl.get(i);
			current.translate(out);
			if(current instanceof ExitStatement){
				out.write("JumpIfNotZero R1,"+kraj);
			}
		}
		out.write("Jump "+pocetak);
		out.newLine();
		out.write(kraj+":");
	}
}
```

Zadatak 3

Tekst zadatka

(*6 poena*) „Select izraz“ u jednom programskom jeziku je definisan na sledeći način:

SelectExpression → **select (** *Expression* **,** *ExpressionList* **)**

Definisati klasu za predstavljanje „select izraza” u apstraktnom sintaksnom stablu.

Definisati međukod niskog nivoa za izračunavanje vrednosti „select izraza“ i u klasi koja ovaj izraz predstavlja u apstraktnom sintaksnom stablu implementirati funkciju za generisanje takvog međukoda.

Značenje izraza je sledeće: izračunava se vrednost prvog izraza u zagradi i to predstavlja redni broj izraza iz liste čija će vrednost biti vraćena kao rezultat.

Strategija generisanja međukoda

```
IMC <USLOV>
```

```
LoadMem R1, RESULT<USLOV>
Store R1, pom
LoadConst R2, 1

IMC<EXPRESSION1>
LoadMem R3, RESULT<EXPRESSION1>
//IMC<EXPRESSIONi> možda promeni sadržaj registra R1, pa zato
ovo radimo
LoadMem R1, pom
JumpIfZero R1, kraj
Sub R1, R2
Store R1, pom

IMC<EXPRESSION2>
LoadMem R3, RESULT<EXPRESSION2>
LoadMem R1, pom
JumpIfZero R1, kraj
Sub R1, R2
Store R1, pom

kraj:
Store R3, RESULT<SELECT EXPRESSION>
```

Java kod

```
public class SelectExpression extends Expression{

      private ArrayList<Expression> el;

      private Expression uslov;


     public void translate(BufferedWriter out) throws IOException{

           String kraj=ASTNode.genLab();

           String pom=ASTNode.genVar();


           uslov.translate(out);

           uslov.genLoad("R1", out);

           out.write("Store R1,"+pom);

           out.write("LoadConst R2,1");

           out.newLine();

           for(int i=0;i<el.size();i++){
```

```
                Expression current=el.get(i);
                current.translate(out);
                current.genLoad("R3",out);
                out.write("LoadMem R1,"+pom);
                out.write("JumpIfZero R1,"+kraj);
                out.newLine();
                out.write("Sub R1, R2");
                out.newLine();
                out.write("Store R1,"+pom);
                out.newLine();
            }
            out.write(kraj+":");
            out.newLine();
            this.result=ASTNode.genVar();
            out.write("Store R3,"+result);
        }
}
```