Klasa ASTNode:

```
import java.io.*;

public abstract class ASTNode {

//  broj kreiranih privremenih promenljivih
    private static int varNo;

    //broj kreiranih labela
    private static int labNo;

    public static String genVar()
    {
        //trebalo bi kreiranu promenljivu upisati
        //u tabelu simbola
        return "$tmp" + varNo++;
    }

    public static String genLab()
    {
        return "lab" + labNo++;
    }

    abstract void translate( BufferedWriter out )
    throws IOException;
}
```

## Klasa Expression:

```
import java.io.*;

public abstract class Expression extends ASTNode{

    protected String result;

    protected void genLoad( String reg,
            BufferedWriter out ) throws IOException
    {
        out.write( "\tLoad_Mem\t" +
            reg + ", " + result );
        out.newLine();
    }
}
```

## Klasa BinaryExpression:

```
import java.io.*;

public abstract class BinaryExpression
extends Expression {
    private Expression left;
    private Expression right;

    public BinaryExpression(Expression l,Expression r)
    {
        left = l;
        right = r;
    }

    protected abstract String opCode();

    public void translate(BufferedWriter out)
    throws IOException
    {
        left.translate( out );
        right.translate( out );
        left.genLoad( "R1", out );
        right.genLoad( "R2", out );
        out.write( "\t" + opCode() + "\t\tR1, R2" );
        out.newLine();
        this.result = ASTNode.genVar();
        out.write( "\tStore\t\tR1, " + this.result );
        out.newLine();
    }
}
```

## Klasa Assignment:

```
import java.io.BufferedWriter;

public class Assignment extends Statement {
    private Variable left;
    private Expression right;

    public Assignment( Variable var, Expression e )
    {
        left = var;
        right = e;
    }

    public void translate( BufferedWriter out )
    throws IOException
    {
        right.translate( out );
        right.genLoad( "R1", out );
        out.write( "\tStore\t\tR1, " + left.name );
        out.newLine();
    }
}
```

## Klasa IfStatement:

```
import java.io.*;

public class IfStatement extends Statement{
    private Statement thenStatement;
    private Statement elseStatement;
    private Expression condition;

    public IfStatement(Expression e, Statement thenS,
            Statement elseS )
    {
        condition = e;
        thenStatement = thenS;
        elseStatement = elseS;
    }

    public void translate( BufferedWriter out )
    throws IOException
    {
        condition.translate( out );
        condition.genLoad( "R1", out );
        String elseLabel = ASTNode.genLab();
        String endLabel = ASTNode.genLab();
        out.write( "\tJumpIfZero\tR1, " + elseLabel );
        out.newLine();
        thenStatement.translate( out );
        out.write( "\tJump\t" + endLabel );
        out.newLine();
        out.write( elseLabel + ":" );
        out.newLine();
        elseStatement.translate( out );
        out.write( endLabel + ":" );
        out.newLine();
    }
}
```

## Klasa Block:

```
import java.io.*;
import java.util.ArrayList;

public class Block extends Statement {
    private ArrayList statements = new ArrayList();

    public void addStatement( Statement newStatement )
    {
        statements.add( newStatement );
    }

    public void translate( BufferedWriter out )
    throws IOException
    {
        for ( int i=0; i<statements.size(); i++ )
        {
            Statement current =
                (Statement) statements.get(i);
            current.translate( out );
        }
    }
}
```

Jedan primer funkcije main:

```
public static void main(String[] args) {

   try
   {
     FileReader file = new FileReader( args[0] );
     MPLexer scanner = new MPLexer( file );
     MPParser parser = new MPParser( scanner );
     ASTNode ast = (ASTNode) parser.parse().value;
     String outFileName = args[0].substring(
         0, args[0].indexOf( ".")+1 );
     outFileName += "ic";
     BufferedWriter writer =
         new BufferedWriter(
             new FileWriter( outFileName ));
     ast.translate( writer );
     writer.close();
   }
   catch( Exception e )
   {
      System.out.println(e);
   }
}
```