Министерство науки и высшего образования Российской Федерации
ФГБОУ ВО «Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»
(ВлГУ)

Кафедра «Информатики и защиты информации»

Курсовая работа на тему:

Разработка компилятора подмножества процедурного языка в ассемблер

Специальность: 10.05.04 – Информационно-аналитические системы безопасности

СОБОЛЕВ Артём Станиславович,
ст. гр. ИСБ-118

г. Владимир 2021

Описание компилятора

Компилятор реализован на языке Python
при помощи библиотеки ply.
Исходный код транслируется в ассемблер MIPS

Ply – инструмент синтаксического анализа, написанный исключительно на python. Ply использует тот же метод анализа что и Lex и Yacc. Так же имеются обширные возможности отладки и отчетов об ошибках.

Mars – эмулятор MIPS ассемблера, имеющий функцию отладки и дизассемблирования

Грамматика

Грамматика представляет из себя вложенные конструкции – правила которым должен соответствовать исходный код. На вход подаются токены из лексера, из которых и строятся блоки, а в дальнейшем и дерево программы.

```
'program :  program_heading semicolon block DOT'

'program_heading :  RESERVED_PROGRAM identifier'

'program_heading :  RESERVED_PROGRAM identifier LPAREN identifier_list RPAREN'

'identifier_list :  identifier_list comma identifier'

'identifier_list :  identifier'

'block :  declaration_part_list statement_part'

'block :  statement_part'
```

Грамматика

Пример построенного дерева для программы:

program Hello_World;
var
a:=integer;
start
a:=5+2;
writeln(a);
stop.

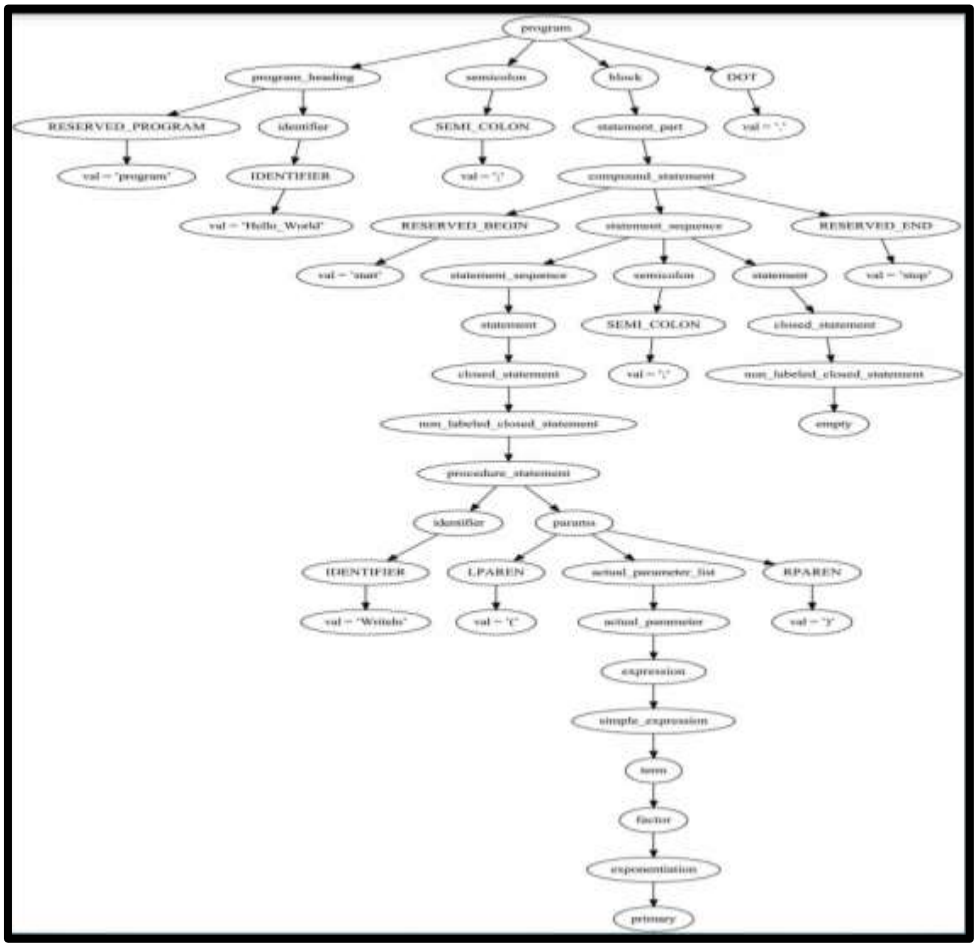


Таблица символов

Таблица символов представлена в виде хэш-таблицы и содержит в себе ключ и значения. Ключом является временная переменная, а её значениями: сдвиг в стеке, размер, тип, имя переменной в программе, путь к переменной и имя временной переменной.

Эти данные пересылаются в генератор промежуточного кода, представляющего собой трёхадресный код типа x:= y op z.

Трансляция в целевой код

Целевым кодом является ассемблер MIPS.

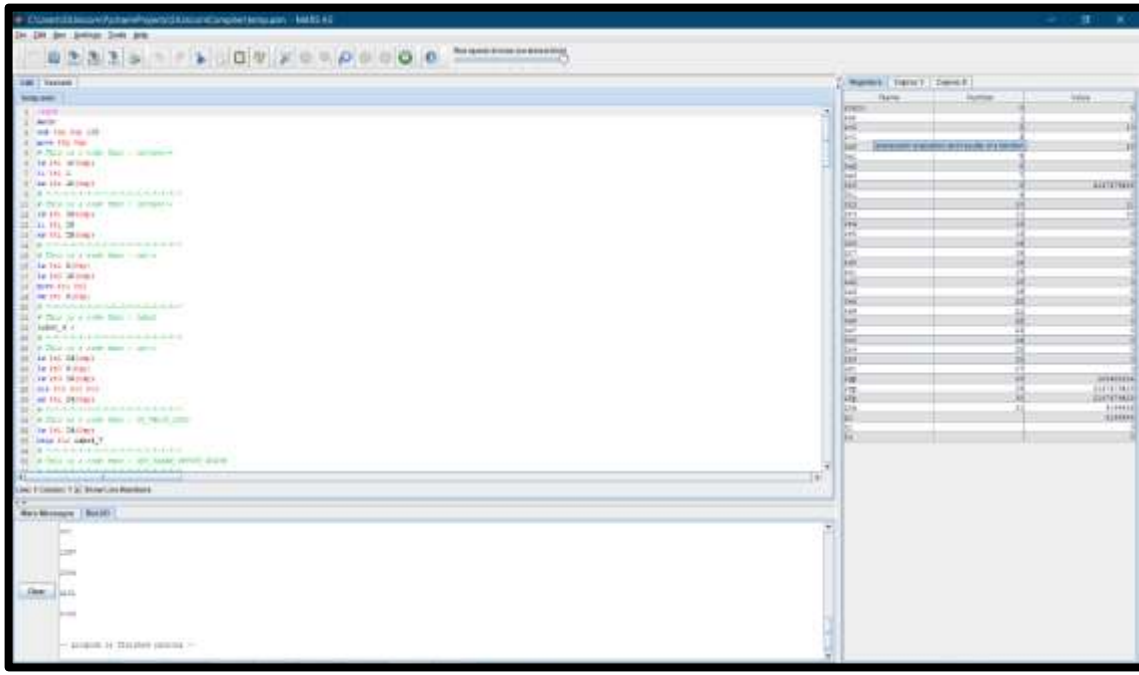
Промежуточный код прогоняется по множеству условий генератора и при соответствии им генерирует соответствующий машинный код.

temp.asm

```
.text
main:
sub $sp $sp 120
move $fp $sp
# This is a code func : integer:=
lw $t1 12($sp)
li $t1 20
sw $t1 12($sp)
# ----------------------
# This is a code func : integer:=
lw $t1 0($sp)
lw $t2 12($sp)
move $t1 $t2
sw $t1 0($sp)
# ----------------------
# This is a code func : integer:=
lw $t1 16($sp)
li $t1 0
sw $t1 16($sp)
# ----------------------
# This is a code func : integer:=
lw $t1 4($sp)
lw $t2 16($sp)
move $t1 $t2
sw $t1 4($sp)
# ----------------------
# This is a code func : integer:=
lw $t1 20($sp)
li $t1 1
sw $t1 20($sp)
# ----------------------
# This is a code func : integer:=
lw $t1 0($sp)
lw $t2 20($sp)
move $t1 $t2
sw $t1 0($sp)
# ----------------------
# This is a code func : label
label_1 :
# ----------------------
# This is a code func : integer:=
lw $t1 24($sp)
li $t1 10
sw $t1 24($sp)
```

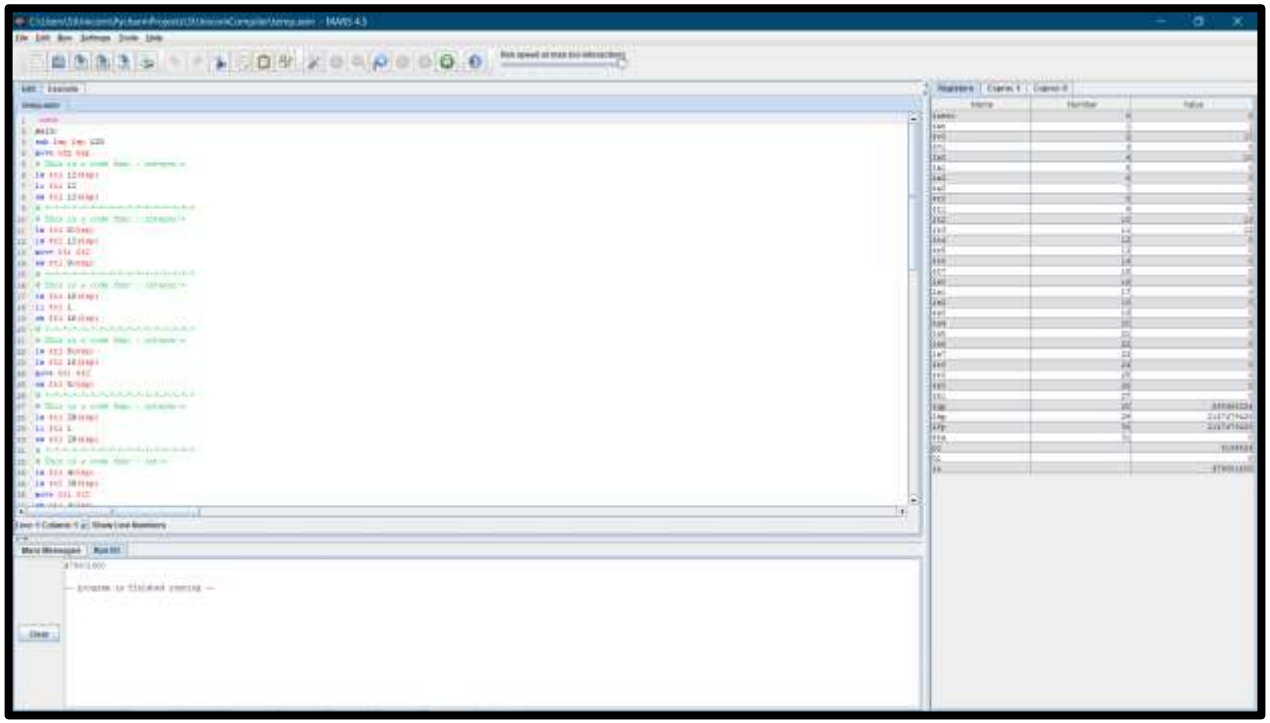
Пример работы компилятора

```
program fibonacci;

func fib(n:integer): integer;
start
    if (n <= 2) then
    start
            fib := 1;
    stop;
    if (n > 2) then
    start
            fib := fib(n-1) + fib(n-2);
    stop;
stop;

var
    i:integer;
    e:integer;
    s:string;
    out : integer;

start
    for i := 1 to 20 do
    start
        out := fib(i);
        writeln(out);
    writeln(' ');
    stop
stop.
```



Пример работы компилятора

```
program factorial;
var
n, i, s: integer;

start
  n := 12;
  s := 1;
  for i := 1 to n do
    start
        s := s * i;
    stop;
writeln(s)
stop.
```



Спасибо за внимание!

СОБОЛЕВ Артём Станиславович,
ст. гр. ИСБ-118