Introduction

This is a project assignment for the undergraduate course Compilers of the Department of Computer Science and Engineering at the University of Ioannina.

Python3 was used to develop the compiler. For the assembly files MIPS was used.

The programming language that compiles is called Scarlet.

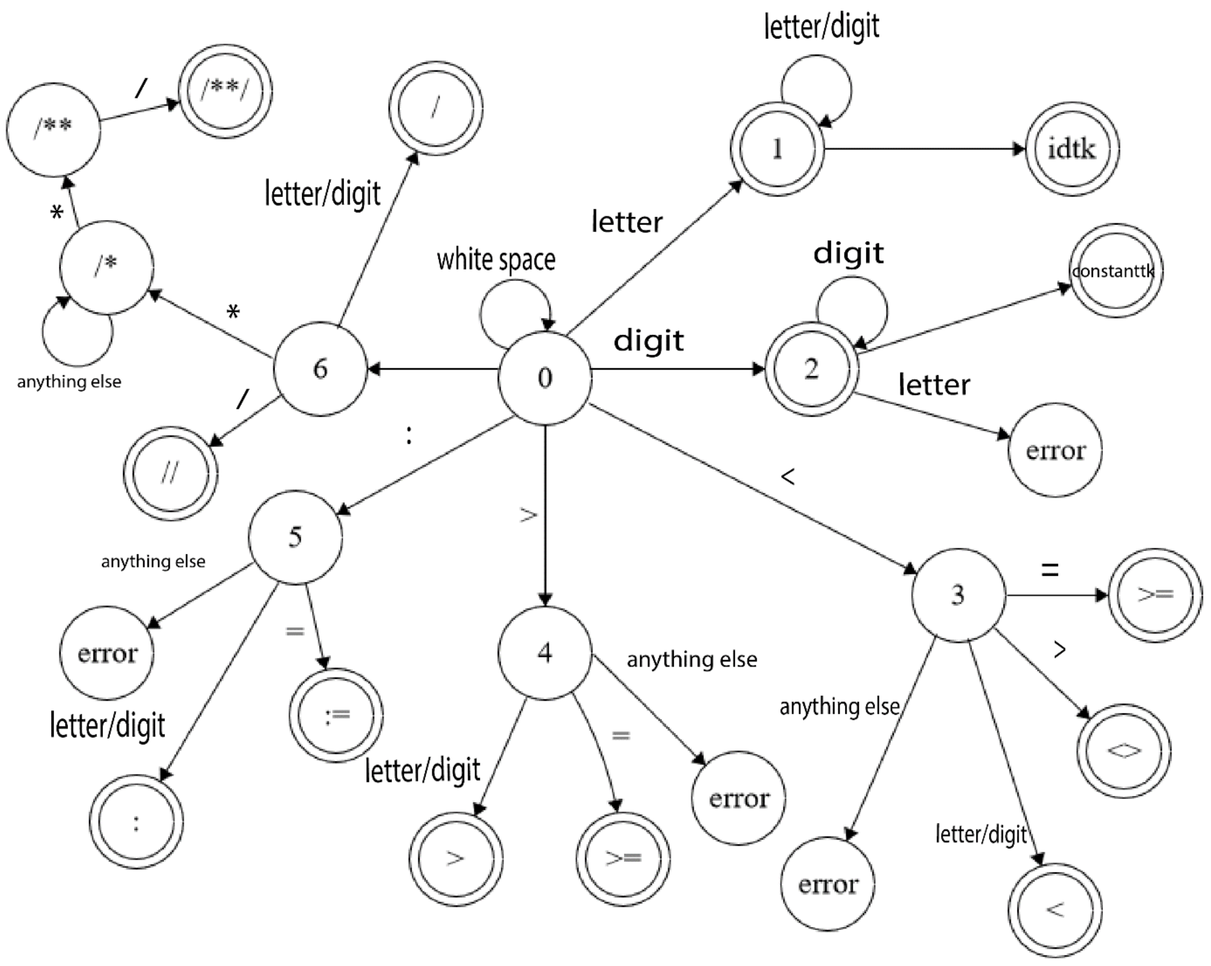
Scarlet

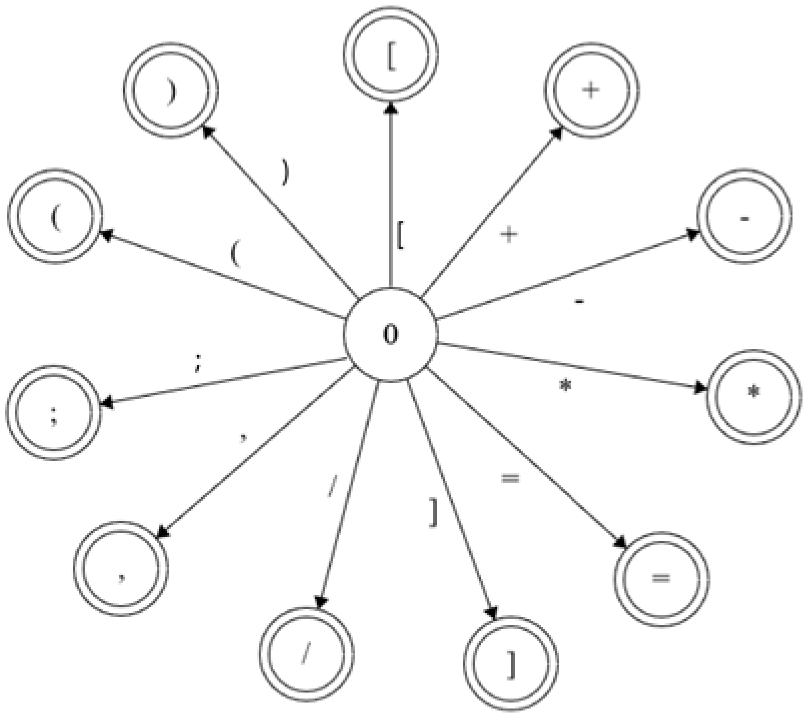
Scarlet contains.

- Letters «A»,…,«Z» and «a»,…,«z»
- Numbers «0»,…,«9»
- Operators «+», «-», «*», «/»
- Comparators «<», «>», «=», «<=», «>=», «<>»
- Assignment «:=»,
- Separators («;», «,», «:»)
- Brackets «(»,«)»,«[»,«]»
- Comments (/* , */ , //)

Reserved words.

program, endprogram
declare
if , then , else , endif
while , endwhile , dowhile , enddowhile
loop , endloop , exit
forcase , endforcase , incase , endincase , when , default , enddefault
function , endfunction , return , in , inout , inandout
and , or , not
input , print


letter/digit
1
idtk
letter
white space
0
digit
2
constanttk
letter
error
6
letter/digit
/
*
/*
/**
/
/**/
anything else
/
//
:
5
anything else
error
=
:=
letter/digit
:
>
4
anything else
=
letter/digit
>
>=
error
<
3
=
>=
>
<>
anything else
error
letter/digit
<


0
[
)
+
(
-
;
*
,
/
]
=

Grammar

```
<program>            ::=  program id <block> endprogram

<block>              ::=  <declarations>  <subprograms>  <statements>

<declarations>       ::=  ( declare  <varlist> ; )*

<varlist>            ::=  ε | id ( , id )*

<subprograms>        ::=  ( <subprogram> ) *

<subprogram>         ::=  function  id  <funcbody>  endfunction

<funcbody>           ::=  <formalpars>  <block>

<formalpars>         ::=  ( <formalparlist> )

<formalparlist>      ::=  <formalparitem>  ( , <formalparitem> )* | ε

<formalparitem>      ::=  in id | inout id | inandout id

<statements>  :      :=  <statement>  ( ; <statement> )*

<statement>          ::=  ε |

                              <assignment-stat> |

                              <if-stat> |

                              <while-stat> |

                              <do-while-stat> |

                              <loop-stat> |

                              <exit-stat> |

                              <forcase-stat> |

                              <incase-stat> |

                              <return-stat> |

                              <input-stat> |

                              <print-stat>

<assignment-stat>    ::=  id := <expression>

<if-stat>            ::=  if   (<condition>)  then  <statements>  <elsepart>  endif

<elsepart>           ::=  ε | else <statements>

<while-stat>         ::=  while  ( <condition> )  <statements>  endwhile

<do-while-stat>      ::=  dowhile  <statements>  enddowhile  ( <condition> )

<loop-stat>          ::=  loop  <statements> endloop
```

<exit-stat>	::= **exit**
<forcase-stat>	::= **forcase**
	(**when** **(** <condition> **)** : <statements>)*
	default : <statements> **enddefault**
	endforcase
<incase-stat>	::= **incase**
	(**when** **(** <condition> **)** : <statements)*
	endincase
<return-stat>	::= **return** <expression>
<print-stat>	::= **print** <expression>
<input-stat>	::= **input** id
<actualpars>	::= **(** <actualparlist> **)**
<actualparlist>	::= <actualparitem> (, <actualparitem>)* | ε
<actualparitem>	::= **in** <expression> | **inout** id | **inandout** id
<condition>	::= <boolterm> (**or** <boolterm>)*
<boolterm>	::= <boolfactor> (**and** <boolfactor>)*
<boolfactor>	::= **not** **[** <condition> **]** | **[** <condition> **]** |
	<expression> <relational-oper> <expression>
<expression>	::= <optional-sign> <term> (<add-oper> <term>)*
<term>	::= <factor> (<mul-oper> <factor>)*
<factor>	::= constant | **(** <expression> **)** | id <idtail>
<idtail>	::= ε | <actualpars>
<relational-oper>	::= **=** | **<=** | **>=** | **>** | **<** | **<>**
<add-oper>	::= **+** | **-**
<mul-oper>	::= ***** | **/**
<optional-sign>	::= ε | <add-oper>

Tests

There are some test used during the assignment to make sure everything was working fine. From the parser generator , syntax analyzer and final assembly code.