```
{
	print("hello")
	boolean x
	x = true
	boolean y
	y = false
	if(x == true) {
	print(y)
	}
}$
```

This Code produces this output when run:

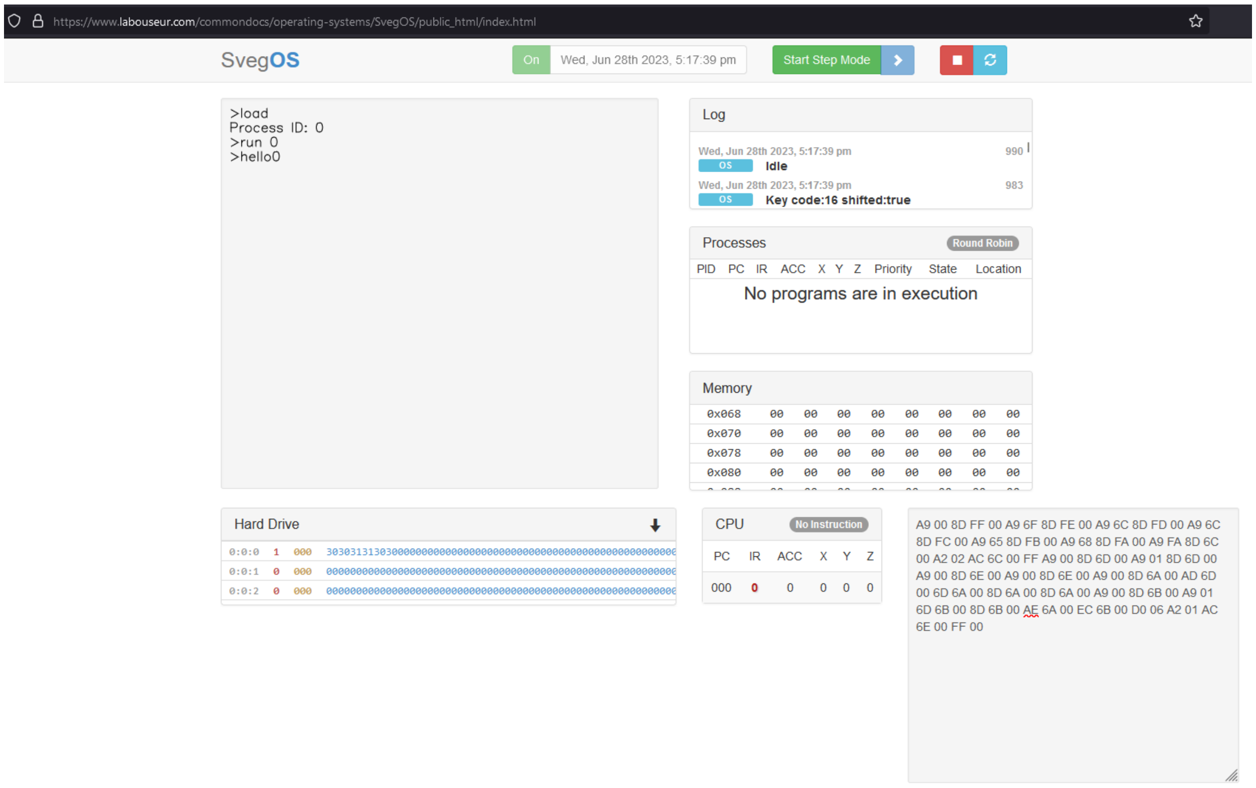
```
$ java source/Compiler.java Progs/codeGen4.txt
INFO  LEXER - Lexing program 1...
DEBUG LEXER - OPEN_BLOCK [ { ] found at (1:1)
DEBUG LEXER - PRINT_STMT [ print ] found at (2:5)
DEBUG LEXER - OPEN_PAREN [ ( ] found at (2:10)
DEBUG LEXER - OPEN_STR [ " ] found at (2:11)
DEBUG LEXER - CHAR_H found at (2:12)
DEBUG LEXER - CHAR_E found at (2:13)
DEBUG LEXER - CHAR_L found at (2:14)
DEBUG LEXER - CHAR_L found at (2:15)
DEBUG LEXER - CHAR_O found at (2:16)
DEBUG LEXER - CLOSE_STR [ " ] found at (2:17)
DEBUG LEXER - CLOSE_PAREN [ ) ] found at (2:18)
DEBUG LEXER - TYPE_BOOL [ boolean ] found at (3:5)
DEBUG LEXER - ID [ x ] found at (3:13)
DEBUG LEXER - ID [ x ] found at (4:5)
DEBUG LEXER - ASSIGN_OP [ = ] found at (4:7)
DEBUG LEXER - BOOL_VAL [ true ] found at (4:9)
DEBUG LEXER - TYPE_BOOL [ boolean ] found at (5:5)
DEBUG LEXER - ID [ y ] found at (5:13)
DEBUG LEXER - ID [ y ] found at (6:5)
DEBUG LEXER - ASSIGN_OP [ = ] found at (6:7)
DEBUG LEXER - BOOL_VAL [ false ] found at (6:9)
DEBUG LEXER - IF_STMT [ if ] found at (7:5)
DEBUG LEXER - OPEN_PAREN [ ( ] found at (7:7)
DEBUG LEXER - ID [ x ] found at (7:8)
DEBUG LEXER - BOOL_OP [ == ] found at (7:10)
DEBUG LEXER - BOOL_VAL [ true ] found at (7:13)
DEBUG LEXER - CLOSE_PAREN [ ) ] found at (7:17)
DEBUG LEXER - OPEN_BLOCK [ { ] found at (7:19)
DEBUG LEXER - PRINT_STMT [ print ] found at (8:9)
DEBUG LEXER - OPEN_PAREN [ ( ] found at (8:14)
DEBUG LEXER - ID [ y ] found at (8:15)
DEBUG LEXER - CLOSE_PAREN [ ) ] found at (8:16)
DEBUG LEXER - CLOSE_BLOCK [ } ] found at (9:5)
DEBUG LEXER - CLOSE_BLOCK [ } ] found at (10:1)
DEBUG LEXER - EOP [ $ ] found at (10:2)
INFO  LEXER - Lex completed successfully

PARSER: Parsing program 1...
Token Stream Received : ( OPEN_BLOCK, PRINT_STMT, OPEN_PAREN, OPEN_STR,
CHAR_H, CHAR_E, CHAR_L, CHAR_L, CHAR_O, CLOSE_STR, CLOSE_PAREN,
TYPE_BOOL, ID, ID, ASSIGN_OP, BOOL_VAL, TYPE_BOOL, ID, ID, ASSIGN_OP,
BOOL_VAL, IF_STMT, OPEN_PAREN, ID, BOOL_OP, BOOL_VAL, CLOSE_PAREN,
OPEN_BLOCK, PRINT_STMT, OPEN_PAREN, ID, CLOSE_PAREN, CLOSE_BLOCK,
CLOSE_BLOCK, EOP )
PARSER: parse()
PARSER: parseProgram()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parsePrintStatement()
PARSER: parseExpression()
PARSER: parseStringExpression()
PARSER: parseCharList()
PARSER: parseChar()
PARSER: parseCharList()
PARSER: parseChar()
PARSER: parseCharList()
PARSER: parseChar()
PARSER: parseCharList()
PARSER: parseChar()
PARSER: parseCharList()
PARSER: parseChar()
PARSER: parseCharList()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseVariableDeclaration()
PARSER: parseType()
PARSER: parseId()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseAssignmentStatement()
PARSER: parseId()
PARSER: parseExpression()
```

```
PARSER: parseBoolValue()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseVariableDeclaration()
PARSER: parseType()
PARSER: parseId()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseAssignmentStatement()
PARSER: parseId()
PARSER: parseExpression()
PARSER: parseBoolValue()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseIfStatement()
PARSER: parseBoolExpression()
PARSER: parseExpression()
PARSER: parseId()
PARSER: parseBoolOperation()
PARSER: parseExpression()
PARSER: parseBoolValue()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parsePrintStatement()
PARSER: parseExpression()
PARSER: parseId()
PARSER: parseStatementList()
PARSER: parseStatementList()
PARSER: Parse completed successfully

CST for program 1...
<Program>
-<Block>
--[{]
--<Statement List>
---<Statement>
----<Print Statement>
-----[print]
-----[(]
-----<Expression>
------<String Expression>
-------["]
-------<Char List>
--------<Char>
---------[h]
--------<Char List>
---------<Char>
----------[e]
---------<Char List>
----------<Char>
-----------[l]
----------<Char List>
-----------<Char>
------------[l]
-----------<Char List>
------------<Char>
-------------[o]
------------<Char List>
-------["]
-----[)]
---<Statement List>
----<Statement>
-----<Variable Declaration>
------<Type>
-------[boolean]
------<Id>
-------[x]
----<Statement List>
-----<Statement>
------<Assign Statement>
-------<Id>
--------[x]
-------[=]
-------<Expression>
--------<Boolean Value>
---------[true]
-----<Statement List>
------<Statement>
-------<Variable Declaration>
--------<Type>
---------[boolean]
--------<Id>
---------[y]
------<Statement List>
-------<Statement>
--------<Assign Statement>
---------<Id>
```

```
----------[y]
---------[=]
---------<Expression>
----------<Boolean Value>
-----------[false]
-------<Statement List>
--------<Statement>
---------<If Statement>
----------[if]
----------<Boolean Expression>
-----------[(]
-----------<Expression>
------------<Id>
-------------[x]
-----------<Boolean Operation>
------------[==]
-----------<Expression>
------------<Boolean Value>
-------------[true]
-----------[)]
----------<Block>
-----------[{]
-----------<Statement List>
------------<Statement>
-------------<Print Statement>
--------------[print]
--------------[(]
--------------<Expression>
---------------<Id>
----------------[y]
--------------[)]
------------<Statement List>
-----------[}]
--------<Statement List>
--[}]
-[$]


SEMANTIC: STARTING SEMANTIC ANALYSIS ON PROGRAM 1.

SEMANTIC: parseBlock()
SEMANTIC: parsePrintStatement()
SEMANTIC: parseVariableDeclaration()
SEMANTIC: parseAssignmentStatement()
```

```
SEMANTIC: parseVariableDeclaration()
SEMANTIC: parseAssignmentStatement()
SEMANTIC: parseIfStatement()
SEMANTIC: parseBlock()
SEMANTIC: parsePrintStatement()

SEMANTIC: AST completed successfully

<Block>
-<Print Statement>
--{hello}
-<Variable Declaration>
--[boolean]
--[x]
-<Assign Statement>
--[x]
--[true]
-<Variable Declaration>
--[boolean]
--[y]
-<Assign Statement>
--[y]
--[false]
-<If Statement>
--[x]
--[true]
--<Block>
---<Print Statement>
----[y]

Name | Type | isInit? | isUsed? | Scope
x|boolean|true|true|0a
y|boolean|true|true|0a

Starting Code Gen...

A9 00 8D FF 00 A9 6F 8D FE 00 A9 6C 8D FD 00 A9 6C 8D FC 00 A9 65 8D FB 00 A9 68 8D
FA 00 A9 FA 8D 6C 00 A2 02 AC 6C 00 FF A9 00 8D 6D 00 A9 01 8D 6D 00 A9 00 8D 6E 00
A9 00 8D 6E 00 A9 00 8D 6A 00 AD 6D 00 6D 6A 00 8D 6A 00 8D 6A 00 A9 00 8D 6B 00 A9
01 6D 6B 00 8D 6B 00 AE 6A 00 EC 6B 00 D0 06 A2 01 AC 6E 00 FF 00
```

Outputted Code:


https://www.labouseur.com/commondocs/operating-systems/SvegOS/public_html/index.html
SvegOS
On
Wed, Jun 28th 2023, 5:17:39 pm
Start Step Mode

```
>load
Process ID: 0
>run 0
>hello0
```

Log

```
Wed, Jun 28th 2023, 5:17:39 pm                  990
OS   Idle
Wed, Jun 28th 2023, 5:17:39 pm                  983
OS   Key code:16 shifted:true
```

Processes — Round Robin

PID	PC	IR	ACC	X	Y	Z	Priority	State	Location

No programs are in execution

Memory

```
0x068   00   00   00   00   00   00   00   00
0x070   00   00   00   00   00   00   00   00
0x078   00   00   00   00   00   00   00   00
0x080   00   00   00   00   00   00   00   00
```

Hard Drive

```
0:0:0  1  000  3030313130300000000000000000000000000000000000000000
0:0:1  0  000  0000000000000000000000000000000000000000000000000000
0:0:2  0  000  0000000000000000000000000000000000000000000000000000
```

CPU — No Instruction

PC	IR	ACC	X	Y	Z
000	0	0	0	0	0

```
A9 00 8D FF 00 A9 6F 8D FE 00 A9 6C 8D FD 00 A9 6C
8D FC 00 A9 65 8D FB 00 A9 68 8D FA 00 A9 FA 8D 6C
00 A2 02 AC 6C 00 FF A9 00 8D 6D 00 A9 01 8D 6D 00
A9 00 8D 6E 00 A9 00 8D 6E 00 A9 00 8D 6A 00 AD 6D
00 6D 6A 00 8D 6A 00 8D 6A 00 A9 00 8D 6B 00 A9 01
6D 6B 00 8D 6B 00 AE 6A 00 EC 6B 00 D0 06 A2 01 AC
6E 00 FF 00
```

hello0