\pash./

Ideal for beginners <3

Angela Tran 180392140
Patrick Mandarino 181466560
Steven Tran 170574740
Haricharan Vinukonda 170618500

Symbol Table

\pash./
Ideal for beginners <3

The symbol table is indexed via the input variable (e.g: x). Its value is the type of the input variable (e.g: int). The values it possesses throughout the program (e.g: 5 or z + 10).

Symbol Table:

ID	Arguments
A	{'2'}
B	{'B + 3'}
...	{. . .}
Z	{'3', 'X + 2'}

Lexer

Input: The source program in PASH.
i.e.

```
func main {
    int i = 2 + 1;
    return i;
}
```

Output: A list/tuple of tokens

I.e. [<*type*, *value*>, <*type*, *value*>, ...]

```
Tokens: [('func_def', 'func'),
('func_decl', 'main'),
('bracket', '{'), ('dataType',
'int'), ('id', 'x'), ('op',
'='), ('int', '2'), ('END',
';'), ('dataType', 'int'),
('id', 'y'), ('op', '='), ('id',
'x'), ('op', '+'), ('int', '1'),
('END', ';'), ('RTRN_STMT',
'return'), ('id', 'y'), ('END',
';'), ('bracket', '}')]
```

\pash./
Ideal for beginners <3

Parser / Syntax Analysis

Input: A list/tuple of tokens (from lexical analysis stage)

I.e. [<*type*, *value*>, <*type*, *value*>, ...]

Returns: True (1) or False (0)

- True meaning list of tokens is formatted correctly
- False meaning list of tokens is formatted poorly

Semantic

Input: A list/tuples of tokens, and the symbol table

I.e:

Tokens: [<*type*, *value*>, <*type*, *value*>, ...]

Symbol Table[y] = y -> ('int', '4', 'y + 5')

Output: True (1) or False (0)

This part of the compiler checks whether the token is classified correctly

Intermediate Code Generation

\pash./
Ideal for beginners <3

Input: list/tuples of tokens, and the symbol table
I.e:

Tokens: [<*type*, *value*>, <*type*, *value*>, ...]

Symbol Table[y] = y -> ('int', '4', 'y + 5')

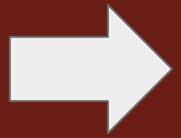
Returns: Intermediate Code (String)

```
ex.
    main: t1 = 0
    t2 = 0
    label1: if t1 >= 3 goto
    END1
    t2 = t2 + 2
    t1 = t1 + 1
    goto label1
    END1: return t2
```

Assembly Code Output: Test 1

\pash./
Ideal for beginners <3

```
func main {
    int x = 2;
    int y = x + 1;
    return y;
}
```



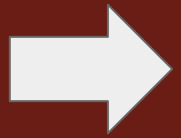
```
main:
LD R1,#2
ADD R2,R1,#1
ST RESULT,R2
```

Assembly Code Output: Test 2

\pash./
Ideal for beginners <3

```
func main {
    int x = 0;
    int y = 0;
    while (x < 3) {
        y = y + 2;
        x = x + 1;
    }

    return y;
}
```

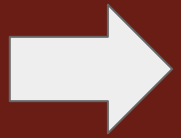


```
main:
LD R1,#0
LD R2,#0
label1:
LD R3,R1
LD R4,#3
SUB R3,R3,R4
BGE R3,R4,END1
ADD R2,R2,#2
ADD R1,R1,#1
B label1
END1:
ST RESULT,R2
```

Assembly Code Output: Test 3

\pash./
Ideal for beginners <3

```
func hello {
    int x = 5;
    int y = 4;
    int z = 10;
    if (x < y) {
      y = y + 5;
      return y;
     }
     elif (z > y) {
     z = z - 3;
     return z;
     }
    else {
      return x;
     }
}
```



```
hello:
LD R1,#5
LD R2,#4
LD R3,#10
LD R4,R1
LD R5,R2
SUB R4,R4,R5
BGE R4,R5,END1
ADD R2,R2,#5
ST RESULT,R2
END1:
LD R4,R3
LD R5,R2
SUB R4,R4,R5
SUB R3,R3,#3
ST RESULT,R3
END2:
ST RESULT,R1
```

Conclusion

\pash./
Ideal for beginners <3

This project gave us a better understanding of a compiler and the different phases involved.

Our experience accomplishing the project goals was we learnt it is not easy to create a compiler and there is a lot of hard work involved :D

Contributions

Angela Tran: 25%

Patrick Mandarino: 25%

Steven Tran: 25%

Haricharan Vinukonda: 25%