my-zhang q1 ed6ad4e 29 seconds ago

1 contributor

81 lines (63 sloc) | 1.22 KB Raw Blame History

Assignment 5

Exercise

Let's consider the following context-free grammar. We are creating a calculator handling identifiers.

```
S -> I; S | ε
I -> ident := E | E | ε
E -> E + T | E - T | T
T -> T * F | T / F | F
F -> ident | const | (E)
```

Question 1

Propose a simple data structure to allow the compiler to represent and handle internally an assembly code.

Tree. We can negerate assembly code through traversing the parse tree. In detail, doing a post-order traversal on the parse tree. When visiting leaf node, print its value(number or variable). On the other hand, print the operation when visiting internal node that represents a arthematical operation.

Question 2

Create an assembly code for stack machine from this grammar.

Implemented in `p1.py`, with lib files in dir `ply`.

Simple example:

```
$ echo "1+2;" | python p1.py
push 1
push 2
add
pop output
[3]
```

A complicated case:

```
$ echo "(1+2)*(3-4); 1+6/3-2;" | python p1.py
push 1
```

```
push 2
add
push 3
push 4
sub
mul
pop output
push 1
push 6
push 3
div
add
push 2
sub
pop output
[-3, 1]
```

With variable:

```
$ echo "a:=1;(a+3)+(2-1);" | python p1.py
push 1
pop a
push a
push 3
add
push 2
push 1
sub
add
pop output
[None, 5]
```