Arithmetic Expression Evaluation

Each time an arithmetic expression is reduced, its value should be evaluated and pushed to a stack data structure. Higher-level expressions can be evaluated hierarchically using the most recent values from the stack.

Sample Arithmetic Expression

$a = 3;$

$b = 4 \times a;$

$var = a \,/\, (-b + 32 \ll 2);$

An Accepted Output

$a = 3;$

$T_0 = 4 \times a;$

$b = T_0;$

$T_1 = -b;$

$T_2 = 32 \ll 2;$

$T_3 = T_1 + T_2;$

$T_4 = a \,/\, T_3;$

$var = T_4$

Reduction Orders

lvalue → a

push(a);

exp → 3

push(3);

assignment → a = 3;

exp = pop();

lvalue = pop();

Code: lvalue = exp;

lvalue → b

push(b);

exp → 4

push(4);

lvalue → a

push(a);

exp → 4 × a

$e_2 = pop();$

$e_1 = pop();$

Code: $T_i = e_1 \times e_2;$

$push(T_i);$

$i = i + 1;$

assignment → b = 4 × a;

exp = pop();

lvalue = pop();

Code: lvalue = exp;

lvalue → var

push(var);

lvalue → a

push(a);

lvalue → b

push(b);

exp → −b

Code: $T_i = -pop();$

$push(T_i);$

$i = i + 1;$

$\text{exp} \rightarrow 32$

push(32);

$\text{exp} \rightarrow 2$

push(2);

$\text{exp} \rightarrow 32 \ll 2$

$e_2 = \text{pop}();$

$e_1 = \text{pop}();$

Code: $T_i = e_1 \ll e_2;$

push(T_i);

$i = i + 1;$

$\text{exp} \rightarrow -b + 32 \ll 2$

$e_2 = \text{pop}();$

$e_1 = \text{pop}();$

Code: $T_i = e_1 + e_2;$

push(T_i);

$i = i + 1;$

$\text{exp} \rightarrow (\ \text{exp}\ )$

No instructions necessary.

$\text{exp} \rightarrow 4 \,/\, (-\text{b} + 32 \ll 2)$

$e_2 = \text{pop}();$

$e_1 = \text{pop}();$

Code: $T_i = e_1 \,/\, e_2;$

$\text{push}(T_i);$

$i = i + 1;$

$\text{assignment} \rightarrow \text{var} = 4 \,/\, (-\text{b} + 32 \ll 2)$

$\text{exp} = \text{pop}();$

$\text{lvalue} = \text{pop}();$

Code: $\text{lvalue} = \text{exp};$