Arithmetic Calculation Language

Nathan Corvino

February 8, 2011

The Artihmetic Calculation Language (ACL) focuses on arithmetic operations, combined with common procedural control structures. It's primary motivation is to explore assembly code generation; as such it is primarily focused on remaining simple to keep this task tractable. At the same time, it does strives to explore as many different code constructs as feasible. It includes variables, conditionals, loops, and function calls.

The syntax is c-like. Statements are terminated with a semicolon, and a list of statements can be enclosed in brackets.

Data Types

ACL will support floats and ints, although it will not necessarily support automatic type coerecion.

Operators

Addition, subtraction, multiplication, and division will be supported, on either ints or floats. Cast operations will be supported for converting between floats and ints.

Comparison operators allowed will be <, >, =, <=, >=, and !=.

Control Structures

ACL will support while loops and if-else structures.

Functions

Function calls will be supported, and at least four parameters will be supported. Return types of int, float, and void—that is, no return type—will be supported.

The program entry point will be the main method which returns an int. Passing command line arguments to it will not be supported.

Example

An example program, that illustrates these constructs in action:

```
int abs(int number)
{
    if (number < 0) {
        return -1 * number;
    } else {
        return number;
    }
}

int power(int base, int exponent)
{
    int total = 1;

    while (exponent > 0) {
        total = base * total;
        exponent = exponent - 1;
    }

    return total;
}

int main()
{
    float x = 4.;
    int y = -3;

    return power((int) x, abs(y));
}
```

This sample program returns 64 when compiled by gcc, as expected; it should do the same when compiled by the ACL compiler.

This example demonstrates ints and floats being used in conjuction with casts, comparisons, looping, branching, and function calls.