Nicolas Viennot

COMS W4115 – Project Proposal

ltc

less than C

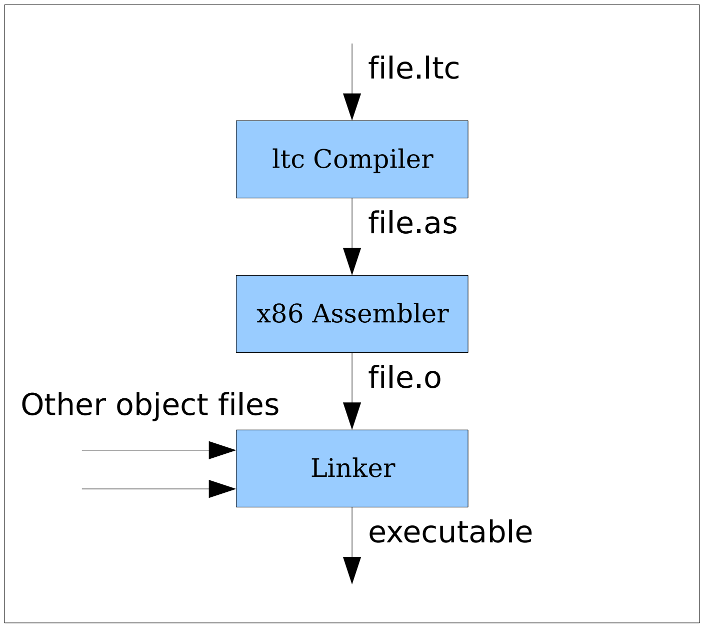
09/24/08

Introduction

The ltc is a very small subset of the C language. The project consists of creating a compiler written in ocaml that translate ltc code into assembly.

Overview

The standard GNU tools `as` and `ld` are used as assembler and linker.



Language features

The ltc syntax is quite similar to C: statements end with “;”, blocks are surrounded with “{ }”. the language has very limited features: arrays and pointers are not supported and there are no variable types (only integer). The major differences between ltc and C are described as follows:

Variables

- variables can only be 32 bits signed integers
- variables don't need to be declared
- global/static variables are not allowed
- variables are accessible in the function scope

Operators

- only the aritmetic, comparison, bitwise and assignment operators are available
- operator precedence is the same as in C.

Functions

- functions can be recursive
- arguments are passed by value
- functions always return an integer and the return value is implicitly given by the result of the last statement
- inline functions are not available

Control

- the do..while construction is not available
- the switch keyword is not available
- labels, goto, if/else, while and for loops, behave like C

Comments

- /* this is a comment */, they can be nested
- // this is also a comment

External calls

External functions can be called by including the right .o object files while linking. No “extern” declaration is needed, the linker will complain if the function doesn't exist.
To communicate with the external world, two functions are provided in the “library”: `input()` and `output()`.

Generated Assembly

The generated assembly is not optimized at all. Temporary values are stored on the stack (not in registers), like every other variables which is very inefficient.

Code sample

```
pow(a, b)
{
      if (b == 0)
            1; /* no return keyword */
      else
            a * pow(a, b-1);  /* recursion */
}

main()
{
      a = b = input();  /* read from stdin */

      b += 2;

      for (i = 1; i <= 5; i++) {
            c = pow(a, i);
            output(2 * (c + b)); /* print to stdout */
      }

      1;
}
```