Compiler Construction (List 1)

Hans de Nivelle

14.10.2015

Aim of this task is that you are able to compile a program into LLVM using clang, and to get some basic understanding of LLVM, and how high-level constructs are tranlated into LLVM.

1. Implement the following functions, and compile them into LLVM:

```
char tolower( char c )
{
   return c >= 'A' && c <= 'Z' ? c - 'A' + 'a' : c;
}

char tolower( char c )
{
   if( c >= 'A' && c <= 'Z' )
      c += 'a' - 'A'; // replace capitalism by lowercasism.
   return c;
}
```

   What does the `nsw` flag mean? Which translation uses Φ-functions?

2. Implement and compile to LLVM:

```
void tolower( char* s )
while( *s != '\0' )
{
   *s = tolower( *s );
   s ++ ;
}
```

   What happens when you replace `s++` by `++s`?

3. Implement and compile to LLVM:

```
size_t length( char* s )
{
```

```
   size_t length = 0;
   while( *( s++ ))
      ++ length;
   return length;
}
```

4. Consider the following matrix multiplication function:

```
#define SIZE 3
void mult( double m1[SIZE][SIZE], double m2[SIZE][SIZE],
           double m3[SIZE][SIZE] )
{
   for( size_t i = 0; i < SIZE; ++ i )
      for( size_t j = 0; j < SIZE; ++ j )
         m3 [i][j] = 0.0;

   for( size_t i = 0; i < SIZE; ++ i )
      for( size_t j = 0; j < SIZE; ++ j )
         for( size_t k = 0; k < SIZE; ++ k )
            m3[i][k] += m1[i][j] * m2[j][k];
}
```

   Translate this function with `clang` and make sure that you understand the resulting LLVM.