Subject: Using LLVM to compile U++
Posted by phirox on Thu, 14 May 2009 18:27:08 GMT
View Forum Message <> Reply to Message

I was intrigued by the move of freebsd to switch from GNU GCC to LLVM. So I installed llvm 2.5 with the clang frontend(llvm-gcc and llvm-g++).

I added a new Build Method named LLVM(under Setup->Build methods). Then set all the flags exactly like with GCC and edit the field Compiler name and put in llvm-g++.

Everything compiled ok, except one little part. In Lang.cpp it gave a segfault. I fixed this error by adding a nonsense variable above the ONCELOCK { line: e.g. "bool llvm_fix;".

Ok the main advantages seem to be compiler speed, better error/warning descriptions, runtime optimilizations which lead to faster execution speeds. The compiler speed is noticeable, I would guess about 30% faster in my case. The messages do seem more verbose. But onto the fun stuff, I tested the speed using the Timing package in reference.

The results for the GCC Speed build(-pipe -O2 -ffunction-section -fomit-frame-pointers):

The restm.Elapsed() = 8242
TIMING Index::FindAdd : 721.85 ms - 721.85 ns (2.01 s / 1000000)
TIMING AsString : 160.85 ms - 160.85 ns (1.45 s / 1000000)
TIMING rand : 0.00 ns - 0.00 ns (1.16 s / 1000000)
The results for the LLVM Speed build(identical flags):
tm.Elapsed() = 8334
TIMING Index::FindAdd : 726.35 ms - 726.35 ns (2.12 s / 1000000)
TIMING AsString : 112.35 ms - 112.35 ns (1.50 s / 1000000)
TIMING rand : 0.00 ns - 0.00 ns (1.14 s / 1000000)

Interesting results as you can see, I will definitely keep using it for now. If not only to keep testing upp's compatibility.

Subject: Re: Using LLVM to compile U++
Posted by mirek on Thu, 14 May 2009 19:08:44 GMT
View Forum Message <> Reply to Message

Nice.

Subject: Re: Using LLVM to compile U++
Posted by cbpporter on Wed, 20 May 2009 11:15:12 GMT
View Forum Message <> Reply to Message

Very interesting. I've been keeping an eye out for LLVM and am waiting for clang to produce a stand alone non GCC based front-end, but it's good to see that the GCC front end is so mature.

I am going to try it out when I have some free time, but until then I have a question: what format are the intermediate files generated during compilation? Do you get LLVM bytecode which is converted to native during final link version, or is the LLVM bytecode phase not visible and after the compilation of a single translation unit you get a native object file? If object files are LLVM, maybe it would be possible to produce a platform independent compiled version of a library. This idea has IMO immense potential.

Also, I didn't know about the intentions of FreeBSD. Quite a bold move. I'll look it up on the Internet. It seems that even FreeBSD might have its uses ! (I am not trying to insult FreeBSD or it's fans here).

Thank you for trying this out.

Subject: Re: Using LLVM to compile U++
Posted by phirox on Thu, 21 May 2009 21:38:56 GMT
View Forum Message <> Reply to Message

With the gcc like frontend(llvm-gcc or llvm-g++), it automatically makes compatible/linkable object binaries like any other compiler. So basically there is no difference in input, arguments or ouput. To create llvm bytecode you have to compile with some extra flags, but you (currently) cannot make a cross-platform binary.

This is the link to the freebsd news about the move.
