You can jump back Home, or to the Blog, Open Source or Research sections.

Getting started with the (recent) LLVM C JIT API

25th September 2018 (6 years ago)

I've been looking into using the LLVM C library to enable me to add a JIT compiler to a small project I'm working on. I came across Paul Smith's excellent blog post describing how to get started with the LLVM C API. In it, he describes how to JIT-compile a simple sum function, but unfortunately the code within doesn't work with recent versions of LLVM. I'm using the version of LLVM as installed by brew (on macOS) as of the time of writing - 7.0.0 - I therefore thought I would update his code to work with this newer version of LLVM.

Getting started, the error I was getting when trying to run `make` against Paul's Makefile was:

```
$ make
clang++ sum.o `llvm-config --cxxflags --ldflags --libs core executionengine jit interpreter analysis native bitwriter --syst
llvm-config: unknown component name: jit
Undefined symbols for architecture x86_64:
  "_LLVMAddFunction", referenced from:
      _main in sum.o
  "_LLVMAppendBasicBlock", referenced from:
      _main in sum.o
  "_LLVMBuildAdd", referenced from:
      _main in sum.o
  "_LLVMBuildRet", referenced from:
       main in sum.o
```



```
  "_LLVMCreateGenericValueOfInt", referenced from:
      _main in sum.o
  "_LLVMDisposeBuilder", referenced from:
      _main in sum.o
  "_LLVMDisposeExecutionEngine", referenced from:
      _main in sum.o
  "_LLVMDisposeMessage", referenced from:
      _main in sum.o
  "_LLVMFunctionType", referenced from:
      _main in sum.o
  "_LLVMGenericValueToInt", referenced from:
      _main in sum.o
  "_LLVMGetParam", referenced from:
      _main in sum.o
  "_LLVMInitializeX86Target", referenced from:
      _LLVMInitializeNativeTarget in sum.o
  "_LLVMInitializeX86TargetInfo", referenced from:
      _LLVMInitializeNativeTarget in sum.o
  "_LLVMInitializeX86TargetMC", referenced from:
      _LLVMInitializeNativeTarget in sum.o
  "_LLVMInt32Type", referenced from:
      _main in sum.o
  "_LLVMLinkInJIT", referenced from:
      _main in sum.o
  "_LLVMModuleCreateWithName", referenced from:
      _main in sum.o
  "_LLVMPositionBuilderAtEnd", referenced from:
      _main in sum.o
  "_LLVMRunFunction", referenced from:
      _main in sum.o
  "_LLVMVerifyModule", referenced from:
      _main in sum.o
  "_LLVMWriteBitcodeToFile", referenced from:
      _main in sum.o
ld: symbol(s) not found for architecture x86_64
clang-7: error: linker command failed with exit code 1 (use -v to see invocation)
make: *** [sum] Error 1
```

The main problem (hinted at in the line `llvm-config: unknown component name: jit`) is that the "Legacy" JIT interface has been removed from LLVM (for versions after 3.5, according to these slides). Instead, we can use the newer MCJIT (Machine Code JIT) execution engine. To do this, we change the Makefile LDFLAGS line:

```
diff --git a/Makefile b/Makefile
index 90af0dd..16a1d2b 100644
--- a/Makefile
+++ b/Makefile
@@ -4 +4 @@ LD=clang++
-LDFLAGS=`llvm-config --cxxflags --ldflags --libs core executionengine jit interpreter analysis native bitwriter --system-li
+LDFLAGS=`llvm-config --cxxflags --ldflags --libs core executionengine mcjit interpreter analysis native bitwriter --system-
```

After changing this, running `make` again gives a different error:

```
$ make
clang -g `llvm-config --cflags` -c sum.c
sum.c:39:5: warning: implicit declaration of function 'LLVMLinkInJIT' is invalid in C99 [-Wimplicit-function-declaration]
    LLVMLinkInJIT();
    ^
1 warning generated.
clang++ sum.o `llvm-config --cxxflags --ldflags --libs core executionengine mcjit orcjit interpreter analysis native bitwrit
Undefined symbols for architecture x86_64:
  "_LLVMLinkInJIT", referenced from:
      _main in sum.o
ld: symbol(s) not found for architecture x86_64
clang-7: error: linker command failed with exit code 1 (use -v to see invocation)
make: *** [sum] Error 1
```

After web-searching for the error, I found reference to a new method, `LinkInMCJIT`, so I changed the `LinkInJIT` line:

```
diff --git a/sum.c b/sum.c
index 7fce959..1335741 100644
--- a/sum.c
+++ b/sum.c
@@ -39 +39 @@ int main(int argc, char const *argv[]) {
-    LLVMLinkInJIT();
+    LLVMLinkInMCJIT();
```

Now, `make` doesn't complain (minor success!), but trying to run the resulting sum executable fails at runtime:

```
$ ./sum 10 20
LLVM ERROR: Target does not support MC emission!
```

…back to searching for the error message. It appears we need to link in another module to help us emit code, `LLVMInitializeNativeAsmPrinter();` apparently does the trick - running again gets us further:

```
$ ./sum 10 20
LLVM ERROR: MCJIT::runFunction does not support full-featured argument passing. Please use ExecutionEngine::getFunctionAddre
```

It turns out that we cannot call arbitrary functions from the engine interface; searching the errors led me to this commit which helped me fix the problem: we should extract a function pointer and call it directly:

```
diff --git a/sum.c b/sum.c
index 6d1f73a..8a82d57 100644
--- a/sum.c
+++ b/sum.c
@@ -59,6 +59,2 @@ int main(int argc, char const *argv[]) {
-    LLVMGenericValueRef args[] = {
-        LLVMCreateGenericValueOfInt(LLVMInt32Type(), x, 0),
-        LLVMCreateGenericValueOfInt(LLVMInt32Type(), y, 0)
-    };
-    LLVMGenericValueRef res = LLVMRunFunction(engine, sum, 2, args);
-    printf("%d\n", (int)LLVMGenericValueToInt(res, 0));
+    int (*sum_func)(int, int) = (int (*)(int, int))LLVMGetFunctionAddress(engine, "sum");
+    printf("%d\n", sum_func(x, y));
```

After re-compiling with this change finally, we can add numbers:

```
$ ./sum 10 20
30
```

success! For the full code, I've pushed a fixed-up repo to Github.

Thanks go to Paul for his original article and all the other linked (and unlinked!) pages that helped me fix the issue.