August 10th 2022

Write JITLink support for new format/architecture Current Progress

Mentors: Vassil Vassilev, Lang Hames, Stefan Gränitz
Student: Sunho Kim

Google
Summer of Code

JITLink

- Do the same job of LLD but just in time
  - Receives object file (“.o file”) and link in memory to an executable form
- Benefit of using object file format
  - Can use the same compilation pipeline with AOT llvm world
  - Not a lot of overhead; no need to store to file system
- Designed to support full features of AOT compiled object files
  - Supports static initializers, thread local storage, and native linking models without hacks – LLVM users don’t have to turn on untested, slow compilation modes in order to use JIT.
  - Have a generic abstraction of linker objects called LinkGraph – which makes it very easy to transform the inputted object file and make new allocations etc.

My project

- Problem: lack of platform/architecture support in JITLink to make it a viable replacement for old JIT infrastructures.

	Linux (ELF)	Mac (MachO)	Windows (COFF)
ARM64	X	O	X
X86_64	O	O	X
RISCV	O	X	X

My project

- Problem: lack of platform/architecture support in JITLink to make it a viable replacement for old JIT infrastructures.

	Linux (ELF)	Mac (MachO)	Windows (COFF)
ARM64	O	O	X
X86_64	O	O	O
RISCV	O	X	X

My project

- Write JITLink backends for
  - ELF/AARCH64 (arm64 gnu linux)
  - COFF/X86_64 (x86_64 msvc windows)
- Deliverables
  - JIT support of the native linking model on ELF/AARCH64 and COFF/X86_64 with no hacks.
  - Bump up the support tier of linux aarch64 target in Julia – fixing random hangs and crashes.
  - Clean and reliable support of msvc c++ target in clang-repl

Progress on ELF/AARCH64

- Fully completed.
- Thread local variables working fully by supporting the thread descriptor model – the natvie TLS model in aarch64 linux.
- Submitted a pull request in Julia to enable JITLink
  - Passes all julia core testcases including complicated backtrace test case which requires correct registration of EH frames.
  - Fixes the crash in a particular test case that have made linux aarch64 3-tier in julia support list.
  - Turned out to fix a code corruption issue that happens under a heavy JIT compilation usage.
- Didn't take a lot of time as a lot of groundworks were already done in the past when adding ELF/X86_64 support.

Progress on COFF/X86_64

- Almost completed – currently polishing and refactoring the code.
- Built COFF support code from scratch.
- Capable of statically linking the windows C runtime (UCRT) and microsoft STL library.
- Support for various COFF specific features: ImageBase relocation, COMDAT symbols, weak external aliases, linker directive, and dllimports.
- Full support of SEH exception handling that clang requires to support c++ exceptions.

COFF/x86_64 demo

- clang-repl targeting x86_64-windows-msvc
  - Hello, world
  - SEH exception (try, catch)
  - Using microsoft STL library
  - Windows message box
  - C++ std::cout lazy load
  - Using real world static library without any modification

Remaining

- Land COFF patches that enables jitlink in LLJIT/clang-repl without hacks
- Work on lazy lexer to enable multiline statements in clang-repl.