My Experience as a first time contributor to GCC's LTO

Rishi Raj
About Me

- I am Rishi Raj
- Undergrad student at IIT Kharagpur, India
- Terminal guy and a frequent distro hopper
- Loves to read, travel and play badminton
What is LTO?

- Optimisation at link-time instead of traditional compile-time
- More optimisation as linker is aware of all the translation unit
- Downside being a longer compile time and more usage of ram during compilation
- In GCC you can use `-flto` or `-ffat-lto-objects` to enable LTO.
ELF file

Format used for storing binaries, libraries and core dumps on Linux and Unix-based systems.

ELF file structure

source: https://en.wikipedia.org/wiki/Executable_and_Linkable_Format
Role of an assembler in producing LTO object file

Section in an lto object file (ELF)

.text
.data
.bss
.gnu.lto_.inline.9ccac84a5a5a48
.gnu.lto_jmpfuncs.9ccac84a5a5a48
.gnu.lto_ipa_modref.9ccac84a5a5a48
.gnu.lto.lto.9ccac84a5a5a48
.gnu.lto_main.0.9ccac84a5a5a48
.gnu.lto_symbol_nodes.9ccac84a5a5a48
.gnu.lto_refs.9ccac84a5a5a48 PROGBITS
.gnu.lto_decls.9ccac84a5a5a48
.gnu.lto_symtab.9ccac84a5a5a48
.gnu.lto_ext_symtab.9ccac84a5a5a48
.gnu.lto_opts
.comment
.note.GNU-stack
.note.gnu.property
.symtab
.strtab
.shstrtab

Section in an lto object file with -g option (ELF)

.text
.data
.bss
.gnu.debuglto_.debug_info
.rela.gnu.debuglto_.debug_info
.gnu.debuglto_.debug_abbrev
.gnu.debuglto_.debug_line
.rela.gnu.debuglto_.debug_line
.gnu.debuglto_.debug_str
.gnu.debuglto_.debug_line_str
.gnu.lto_.inline.9ccac84a5a5a48
.gnu.lto_jmpfuncs.9ccac84a5a5a48
.gnu.lto_ipa_modref.9ccac84a5a5a48
.gnu.lto.lto.9ccac84a5a5a48
.gnu.lto_main.0.9ccac84a5a5a48
.gnu.lto_symbol_nodes.9ccac84a5a5a48
.gnu.lto_refs.9ccac84a5a5a48
.gnu.lto_decls.9ccac84a5a5a48
.gnu.lto_symtab.9ccac84a5a5a48
.gnu.lto_ext_symtab.9ccac84a5a5a48
.gnu.lto_opts
.comment
.note.GNU-stack
.note.gnu.property
.symtab
.strtab
.shstrtab
Bypassing assembler:

- Extending libiberty to output symbol table, string table
- Extending dwarf2out.cc to output debug sections and symbols directly
  - outputting debug sections
  - outputting relocations

For 1000 invocations with bypass:
real   0m14.186s
user   0m10.957s
sys    0m2.424s

While the default path gets:
real   0m21.913s
user   0m13.856s
sys    0m5.705s

With OpenSUSE 13.1 default GCC 4.8.3 build:
real   0m15.160s
user   0m8.481s
sys    0m5.159s

And with clang-3.4:
real   0m30.097s
user   0m22.012s
sys    0m6.649s
Current status and future plan

- The implementation for ELF file format is in testing phase and can be found at "devel/bypass-asm" branch of the gcc repository.
- For relocations we only support X86-64 target as of now.
- Support for other object file’s type is missing.
Google summer of code (GSOC)

- Google Summer of Code is a global, online program focused on bringing new contributors into open source software development.
- Organisers submit their list of projects to Google and after a few days Google announces a list of participating organisations.
- Contributors can submit up to three proposals for different projects.
- Once selected, contributors work with an open source organization on a 12+ week programming project under the guidance of mentors.
People who made it possible

- Google for organising GSOC which introduced me to open source world
- My mentors Jan Hubička and Martin Jambor for helping and guiding me whenever needed
- Thomas Schwinge and David Malcolm for accepting my talk and assistance in getting sponsorship.
- Attendance at FOSDEM was supported by the GNU Toolchain fund, a part of the FSF's Working Together for Free Software Fund: <https://my.fsf.org/civicrm/contribute/transact?reset=1&id=57>.
Thanks for your attention:

https://github.com/rsh-raj

https://www.linkedin.com/in/rsh-raj/