# RISC-V linker relaxation and LLD

MaskRay

https://maskray.me

## MaskRay

- LLVM contributor

## MaskRay

- LLVM contributor
- (Tiny) binutils/GCC/Linux kernel contributor

## MaskRay

- LLVM contributor
- (Tiny) binutils/GCC/Linux kernel contributor
- ccls owner (C++ language server)

## MaskRay

- LLVM contributor
- (Tiny) binutils/GCC/Linux kernel contributor
- ccls owner (C++ language server)
- 退休的算法竞赛 + 超算 +CTF 选手

## Linker relaxation

- Rewrite code sequence (one or more instructions) to improve performance

## Linker relaxation

- Rewrite code sequence (one or more instructions) to improve performance
- "link-time optimization" with a good balance of code size/speed/convenience

```
// x86-64 GOTPCRELX
int var;
int foo(void) { return var; }

movq var@GOTPCREL(%rip), %rax  # load address from GOT
  -> leaq (var-.)(%rip), %rax  # set address relative to RIP
movl (%rax), %eax              # load variable value
retq

// x86-64 General Dynamic to Local Exec
__thread int tls;
int bar(void) { return tls; }

pushq %rax  # ABI required 16-byte stack alignment
data16 leaq tls@TLSGD(%rip), %rdi
  -> movq %fs:0, %rax
data16 data16 rex64 call __tls_get_addr@PLT
  -> leaq tls@TPOFF(%rax), %rax
movl (%rax), %eax
popq %rcx
retq
```

```
// PowerPC64 ELFv2 toc-indirect to toc-relative
int var; int foo(void) { return var; }

addis 2, 12, .TOC.-.Lfunc_gep0@ha
addi 2, 2, .TOC.-.Lfunc_gep0@l
addis 3, 2, .LC0@toc@ha
  -> addis 3, 2, .LC0@toc@ha
ld 3, .LC0@toc@l(3)
  -> addi 3, 3, .LC0@toc@l
lwa 3, 0(3)
blr

// PowerPC64 ELFv2 General Dynamic to Local Exec
__thread int tls; int bar(void) { return tls; }

addis 2, 12, .TOC.-.Lfunc_gep0@ha # TOC setup
addi 2, 2, .TOC.-.Lfunc_gep0@l
mflr 0
std 0, 16(1)                    # save return address
stdu 1, -32(1)                  # push stack frame
addis 3, 2, tls@got@tlsgd@ha  # load tls_index
  -> nop
addi 3, 3, tls@got@tlsgd@l
  -> addis 3, 13, x@tprel@ha
bl __tls_get_addr(tls@tlsgd)   # call __tls_get_addr
  -> nop
nop                             # maybe TOC restore
  -> addi 3, 3, x@tprel@l
lwa 3, 0(3)                     # load variable value
addi 1, 1, 32                   # pop stack frame
ld 0, 16(1)                     # load return address
mtlr 0
blr
```

## Linker relaxation on RISC-V

- `lui a0, %hi(sym); addi a0, a0, %lo(sym)` ⇒ `addi a0, offset(gp)`
- `.L0: auipc a0, %pcrel_hi(sym); addi a0, a0, %pcrel_lo(.L0)` ⇒ `addi a0, offset(gp)`
- GP relaxation: "12-bit offsets ought to be enough for everyone"

## Linker relaxation in RISC-V (cont.)

- `call sym # auipc ra, ..; jalr ra, ..(ra)` ⇒ `jal sym`

```
void ext(void);
void foo(void) {
  ext();
  ext();
  ext();
  ext();
}

0000000000000000 <.text>:
# sh_addralign=4, insert NOP of sh_addrline-2 bytes
       0: 01 00         nop
                0000000000000000:  R_RISCV_ALIGN        *ABS*+0x2

0000000000000002 <foo>:
       2: 41 11         addi    sp, sp, -16
       4: 06 e4         sd      ra, 8(sp)
       6: 97 00 00 00   auipc   ra, 0
                0000000000000006:  R_RISCV_CALL ext
                0000000000000006:  R_RISCV_RELAX        *ABS*
       a: e7 80 00 00   jalr    ra
       e: 97 00 00 00   auipc   ra, 0
                000000000000000e:  R_RISCV_CALL ext
                000000000000000e:  R_RISCV_RELAX        *ABS*
      12: e7 80 00 00   jalr    ra
      16: 97 00 00 00   auipc   ra, 0
                0000000000000016:  R_RISCV_CALL ext
                0000000000000016:  R_RISCV_RELAX        *ABS*
      1a: e7 80 00 00   jalr    ra
      ...
```

```
0000000000000244 <foo>:
     244: 41 11          addi    sp, sp, -16
     246: 06 e4          sd      ra, 8(sp)
     248: ef 00 80 01    jal     24 <ext>
     24c: ef 00 40 01    jal     20 <ext>
     250: ef 00 00 01    jal     16 <ext>
     254: ef 00 c0 00    jal     12 <ext>
     258: a2 60          ld      ra, 8(sp)
     25a: 41 01          addi    sp, sp, 16
     25c: 11 a0          j       4 <ext>
     25e: 00 00          unimp
```

## ELF and DWARF

"Some time after the Elves had awakened at Cuiviénen, the Seven Fathers of the Dwarves were released from their stone chambers."

## Address ranges in DWARF

- `DW_AT_low_pc`
- `DW_AT_high_pc`

```
0x0000002a:   DW_TAG_subprogram [2]
                DW_AT_low_pc [DW_FORM_addr]     (0x0000000000000002 ".text")
                  # R_RISCV_64
                DW_AT_high_pc [DW_FORM_data4]   (0x00000040)
                  # Constant on other architectures
                  # Two relocations: R_RISCV_ADD32 and R_RISCV_SUB32
                  # Neither GCC nor Clang uses DW_FORM_addr (DWARF v3)
                DW_AT_frame_base [DW_FORM_exprloc]      (DW_OP_reg8 X8)
                DW_AT_name [DW_FORM_strp]       ( .debug_str[0x00000020] = "foo")
                DW_AT_decl_file [DW_FORM_data1] ("/tmp/c/a.c")
                DW_AT_decl_line [DW_FORM_data1] (2)
                DW_AT_external [DW_FORM_flag_present]   (true)
```

## .debug_line

DWARF spec: "Most of the instructions in a line number program are special opcodes."

- one entry on another arch: 1 byte
- one entry on RISC-V: 54 bytes

```
// x86-64
            Address             Line   Column File   ISA Flags
            ------------------ ------ ------ ------ --- -------------
0x00000025: 00 DW_LNE_set_address (0x0000000000000000)
0x00000030: 13 address += 0,  line += 1
            0x0000000000000000      2      0      1   0  is_stmt
0x00000031: 05 DW_LNS_set_column (3)
0x00000033: 0a DW_LNS_set_prologue_end
0x00000034: 4b address += 4,  line += 1
            0x0000000000000004      3      3      1   0  is_stmt prologue_end
0x00000035: 75 address += 7,  line += 1
            0x000000000000000b      4      3      1   0  is_stmt
0x00000036: 75 address += 7,  line += 1
            0x0000000000000012      5      3      1   0  is_stmt
0x00000037: 75 address += 7,  line += 1
            0x0000000000000019      6      3      1   0  is_stmt
0x00000038: 75 address += 7,  line += 1
            0x0000000000000020      7      3      1   0  is_stmt
0x00000039: 75 address += 7,  line += 1
            0x0000000000000027      8      3      1   0  is_stmt
```

```
// RISC-V
            Address             Line   Column File   ISA Flags
            ------------------ ------ ------ ------ --- -------------
0x00000025: 00 DW_LNE_set_address (0x0000000000000002)
0x00000030: 13 address += 0,  line += 1
            0x0000000000000002      2      0      1   0  is_stmt
0x00000031: 05 DW_LNS_set_column (3)
0x00000033: 0a DW_LNS_set_prologue_end
0x00000034: 03 DW_LNS_advance_line (3)
0x00000036: 09 DW_LNS_fixed_advance_pc (0x0002)
0x00000039: 01 DW_LNS_copy
            0x0000000000000004      3      3      1   0  is_stmt prologue_end
0x0000003a: 03 DW_LNS_advance_line (4)
0x0000003c: 09 DW_LNS_fixed_advance_pc (0x000e)
0x0000003f: 01 DW_LNS_copy
            0x0000000000000012      4      3      1   0  is_stmt
0x00000040: 03 DW_LNS_advance_line (5)
0x00000042: 09 DW_LNS_fixed_advance_pc (0x0008)
0x00000045: 01 DW_LNS_copy
            0x000000000000001a      5      3      1   0  is_stmt

Relocation section '.rela.debug_line' at offset 0x7e0 contains 17 entries:
    Offset             Info             Type       Symbol's Value  Symbol's Name + Addend
0000000000000028  0000000400000002 R_RISCV_64    0000000000000002 <null> + 0
0000000000000037  0000000600000022 R_RISCV_ADD16 0000000000000004 <null> + 0
0000000000000037  0000000400000026 R_RISCV_SUB16 0000000000000002 <null> + 0
000000000000003d  0000000a00000022 R_RISCV_ADD16 0000000000000012 <null> + 0
000000000000003d  0000000600000026 R_RISCV_SUB16 0000000000000004 <null> + 0
0000000000000043  0000000b00000022 R_RISCV_ADD16 000000000000001a <null> + 0
0000000000000043  0000000a00000026 R_RISCV_SUB16 0000000000000012 <null> + 0
```

## `.gcc_except_table`

- Call-site table: a list of call sites that may throw an exception
- Similar to address ranges in DWARF

```
inline int comdat() {
  try { throw 1; }
  catch (int) { return 1; }
  return 0;
}

  .section        .gcc_except_table,"a",@progbits
  .p2align        2
GCC_except_table1:
.Lexception0:
  .byte   255                           # @LPStart Encoding = omit
  .byte   155                           # @TType Encoding = indirect pcrel sdata4
  .uleb128 .Lttbase0-.Lttbaseref0
.Lttbaseref0:
  .byte   3                             # Call site Encoding = udata4
  .uleb128 .Lcst_end0-.Lcst_begin0
.Lcst_begin0:
# RISC-V -mrelax cannot use .uleb128.
# Worse, label differences are not resolved at assembly time.
  .word   .Lfunc_begin1-.Lfunc_begin1  # >> Call Site 1 <<
  .word   .Ltmp2-.Lfunc_begin1         #   Call between .Lfunc_begin1 and .Ltmp2
  .word   0                            #     has no landing pad
  .byte   0                            #   On action: cleanup
  .word   .Ltmp2-.Lfunc_begin1         # >> Call Site 2 <<
  .word   .Ltmp3-.Ltmp2                #   Call between .Ltmp2 and .Ltmp3
  .word   .Ltmp4-.Lfunc_begin1         #     jumps to .Ltmp4
  .byte   1                            #   On action: 1
  .word   .Ltmp3-.Lfunc_begin1         # >> Call Site 3 <<
  .word   .Lfunc_end1-.Ltmp3           #   Call between .Ltmp3 and .Lfunc_end1
  .word   0                            #     has no landing pad
  .byte   0                            #   On action: cleanup
.Lcst_end0:
  .byte   1                            # >> Action Record 1 <<
  ...
```

## Relocations

- `.eh_frame` has relocations referencing `.text.foo`
- `.eh_frame` has relocations referencing `.gcc_except_table`
- RISC-V -mrelax only: `.gcc_except_table` has relocations referencing `.text.foo`

## Relocations

- `.gcc_except_table` cannot be monolithic on RISC-V. Why?
- If `.text.foo` is discarded (due to `--gc-sections` or other means), the referencing `.gcc_except_table` breaks ELF spec
- ELF spec: "A symbol table entry with `STB_LOCAL` binding that is defined relative to one of a group's sections, and that is contained in a symbol table section that is not part of the group, must be discarded if the group members are discarded. References to this symbol table entry from outside the group are not allowed."
- Teaching LLD `.gcc_except_table` works but is inelegant (against ELF spirit).

## SHF_LINK_ORDER

- Probable fix, if clang knows the linker flavor: LLD.
- SHF_ASSOCIATED: one concern is that defining it now may restrict the design space.
- SHF_LINK_ORDER: supported by GNU ld and Solaris. Exploit/botch it, until we get unhappy and then migrate to SHF_ASSOCIATED...

## LLD

- LLVM Linker
- Mature COFF/ELF/wasm ports. WIP Mach-O port.
- AArch64, ARM(>=v6), PowerPC, PowerPC64, x86-32, x86-64 have production quality. MIPS seems decent too.
- Fast, portable (all-in-one), simple (ELF: 30000+ lines), great LLVM LTO integration

## RISC-V support

Major patches

- Chih-Mao Chen `https://reviews.llvm.org/D39322` [lld] Support RISC-V
- MaskRay `https://reviews.llvm.org/D63076` [ELF] [RISCV] Support PLT, GOT, copy and relative relocations
- MaskRay `https://reviews.llvm.org/D63220` [ELF] [RISCV] Support GD/LD/IE/LE TLS models

## Steps

- Command line options
- Symbol table (input files, `-e`, `-u`, symbol assignments)
- LLVM LTO (bitcode ⇒ object files)
- Input sections
- Split `SHF_MERGE` and `.eh_frame`
- `--gc-secitons`: markLive
- Create synthetic sections (linker generated)
- Move `.eh_frame` from inputSections to synthetic `.eh_frame`
- Linker script SECTIONS
- Identical Code Folding
- Scan relocations
- Finalize synthetic sections
- Layout (addresses, thunks, `SHT_RELR`, symbol assignments)
- Assign file offsets
- Write header and sections

## `riscv_relax_delete_bytes` is difficult in LLD

- Scan relocations
- Finalize synthetic sections
- Layout (addresses, thunks, `SHT_RELR`, symbol assignments)

- Relocation scanning affects address dependent content.
- In GNU ld, (I think) the three steps are in an iterative loop
- Solution: 2-pass relocation scanning (where is the boundary?)
- `finalizeSynthetic()` called more than once

## References


- All Aboard, Part 3: Linker Relaxation in the RISC-V Toolchain
- Toolchain: Compiler Support for Linker Relaxation in RISC-V. Shiva Chen and Hsiangkai Wang
- What makes LLD so fast? Peter Smith