Puzzle for RISC-V ifunc

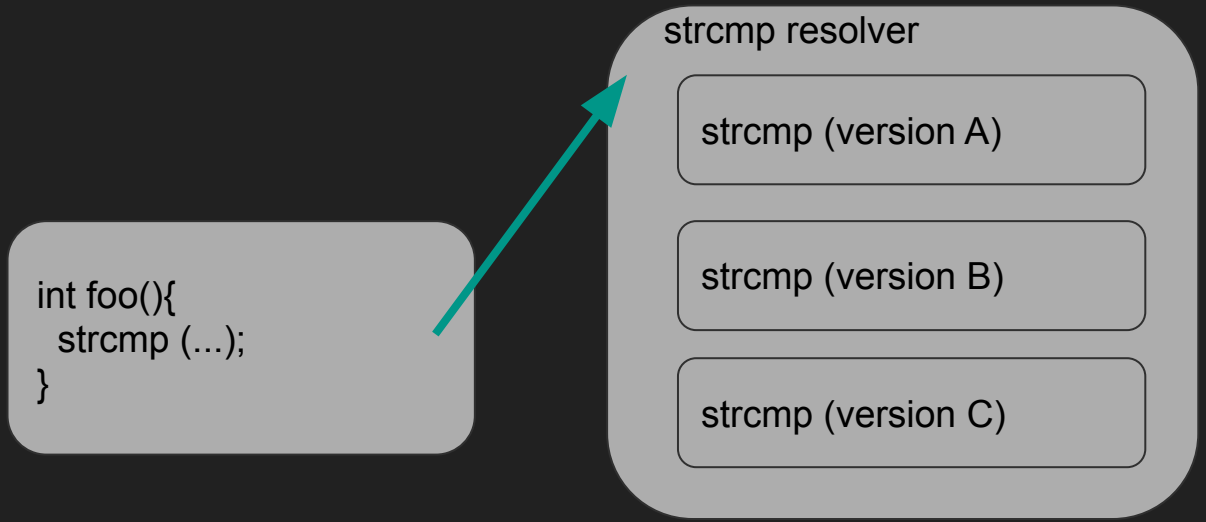
Kito Cheng (SiFive)
Palmer Dabbelt (Google)

Agenda

- What’s IFUNC?
- What’s The Status of RISC-V Support for IFUNC?
- What Are We Missing?
- Dicussion:
  - Design of HWCAP2?

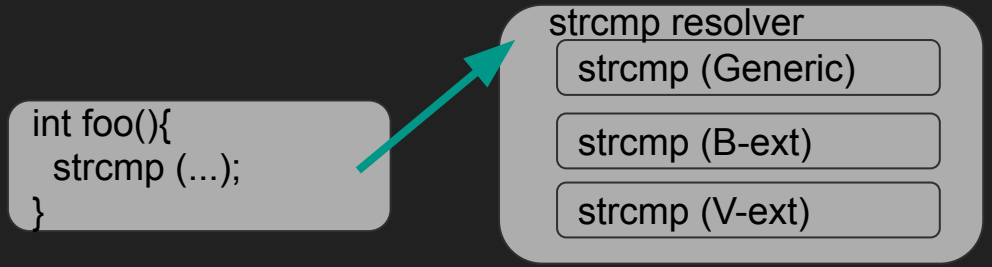
What's IFUNC

- https://sourceware.org/glibc/wiki/GNU_IFUNC
- The GNU indirect function support (IFUNC) is a feature of the GNU toolchain that allows a developer to create multiple implementations of a given function and to select amongst them at runtime using a resolver function



What's The Status of RISC-V Support for IFUNC?

- RISC-V have basic support for ifunc:
  - glibc's infrstracture.
  - ifunc relocation for RISC-V
    - Added at Feb. 2020
      - https://github.com/riscv-non-isa/riscv-elf-psabi-doc/pull/131
  - Workable PoC?
    - SiFive has an internal implmenation, but isn't complete solution yet.
      - That's why there is this session here :p
- The vector, bit manipulation and crypto extension is comming soon!
  - We need optimize serveral funciton via ifunc soon!



What Are We Missing?

- No Mechnish to Disable/Enable Specfic Extension.
- No Function Target Attribute for C/C++
  - e.g. int sse3_func (void) __attribute__ ((__target__ ("sse3")));
- hwcap is Not Enoght to Detect ISA Feature.

No Mechnish to Disable/Enable Specfic Extension

- Why it’s problem?
  - Build glibc with rv64gc
    - Build strcmp_vec.c with rv64gcv
    - Build strcmp_bitmanip.c with rv64gc_zbb
  - How to specifiy the build option?
    - Hardcode CFLAGS -march=rv64gcv for strcmp_vec.c and -march=rv64gc_zbb for strcmp_bitmanip.c?
      - Yes, that’s what we did in our PoC...

No Mechnish to Disable/Enable Specfic Extension (cont.)

- We have .option norvc/rvc to disable / enable RVC extension, but no option for all other extensions…
- -march option need to specify full arch string in ***canonical order***.
  - -march=$(ARCH)_v
    - Not work well due to it might not satify canonical order…
    - ARCH=rv64gc_zbb then we got rv64gc_zbb_v which is invalid ISA string.
  - We might consider to extend the complication option.
    - e.g
      - -march=+v?
      - -march=$(ARCH)+v
        - -march=rv64gc_zbb+v

Proposal for Disable/Enable Specfic Extension

- Extend .option directive.
  - https://github.com/riscv-non-isa/riscv-asm-manual/pull/67
- .option arch, +<ext>
  - .option arch, +v
    - Enable vector extension for folloing code region.
- .option arch, -<ext>
  - .option arch, -c (=== .option norvc)
    - Disable C extension for folloing code region.
- .option arch, =<full-isa-str>
  - .option arch, =rv64gcv
    - Set arch to rv64gcv for folloing code region.
- Work with .option push/.option pop

No Function Target Attribute for C/C++

- Other target has provide target attribute to specify a function compiled with specify target option.
  - int sse3_func (void) __attribute__ ((__target__ ("sse3")));
    - sse3_func will compile with sse3, no matter it’s enabled by CFLAGS or not.
- RISC-V did’t define that yet.
  - That’s OK for short-term, but we must have for long term.
- Note, this feature depend on `.option arch, ` directive.

hwcap is Not Enoght to Detect ISA Feature.

- HWCAP using similar scheme with misa register layout:
  - One bit per extension.
    - 1st bit for A extension
    - 2nd bit for B extension
    - …
    - 22 bit for V extension.
  - But...RISC-V have bunch of multi-letter extension now!!
    - Vector family: v, zve64d, zve64f, zve64x, zve32f, zve32x, zvl128b...and more.
    - Bit manipulation familiy: zba, zbb, zbc, zbs
    - Crypto family: zkn, zbkb, zbkc, zbkx, zknd, zknh...and more.
  - Vendor extension can’t detect by hwcap.
    - X-bit only indicate there is vendor ext., but which vendor ext.?
- We need to define the hwcap2 to detech hardware features, or some other mechnish to detect in user space!
  - That’s major goal of this session, that’s need to coordniate with kernel and glibc folks.

Proposals for hwcap2

- Use hwcap2 as extended hwcap
  - Not work, just extend extra 32 bits...
- Treat hwcap2 as pointer.
- Raw ISA string with explicity version number, like RISCV_Tag_ARCH:
  - rv64i2p0_m2p0_a2p0_f2p0_d2p0_c2p0_v1p0
  - Easy to pass, but pain to parse.
- Preprocessed ISA string linked list:
  - struct rv_hwcap2 {const char *ext; unsinged major; unsinged minor; struct rv_hwcap2 *next;};
  - Easy to parse and traverse.
- Bit vector scheme with variable lenght encoding
  - ULEB128 encoding for hwcap2, and allocating 1 bit per extension.
  - Pain to encoding version and manage the bit, not friendly for vendor extension.
- Leverage configuration-structure, and use DER or PER encoding (defined in ASN.1)
  - Pain to parse…
    - https://github.com/riscv/configuration-structure
- Variant of one of the above, mixture of above or other proposals?
- Or...don’t use hwcap2?

AT_PLATFORM, AT_BASE_PLATFORM or other RISC-V specific AT_* Value?

- AT_PLATFORM and AT_BASE_PLATFORM are
  - https://man7.org/linux/man-pages/man3/getauxval.3.html
    - AT_BASE_PLATFORM
      - A pointer to a string (PowerPC and MIPS only). On PowerPC, this identifies the real platform; may differ from AT_PLATFORM. On MIPS, this identifies the ISA level (since Linux 5.7).
    - AT_PLATFORM
      - A pointer to a string that identifies the hardware platform that the program is running on. The dynamic linker uses this in the interpretation of rpath values.
- Both value are return NULL for RISC-V now.

Special Thanks

- Fangrui Song (MaskRay)@Google for adding ifunc relocation.
- Greentime@SiFive Hu for the kernel part for vector support.
- Vincent Chen@SiFive for RISC-V vector IFUNC implemenation.
- Nelson Chu@SiFive for binutils PoC implemenation.

Few Other Puzzles...

- Mapping Symbol
  - Assist disassmbler to work with ovelapping instruction encoding.
    - https://github.com/riscv-non-isa/riscv-elf-psabi-doc/pull/196
  - Optional for IFUNC but nice to have.
    -

Thanks :)