BPF in the GNU Toolchain and the Linux kernel

Jose E. Marchesi

Oracle Inc.

GNU Toolchain MC @ LPC 2020

ORACLE

Contents

ORACLE

The Project

- **Phase 1:** add BPF target to the toolchain
- **Phase 2:** make the generated programs palatable for the kernel loaders and verifier, and **keep it that way**.
- **Phase 3:** provide other development goodies for BPF developers (simulator, debugger, tracer, etc.)

Port Status

`bpf-unknown-none`

- **binutils port**
  - Upstream since Aug 2019.
  - Debian: `binutils-bpf`
  - Oracle Linux 8: `cross-binutils`.
- **GCC backend**
  - Upstream since Sep 2019.
  - Debian: `gcc-bpf`
  - Oracle Linux 8: `cross-gcc`.
- **GDB port**
  - Upstream since Aug 2020.
- **Simulator**
  - Upstream since Aug 2020.
- **Dejagnu board**
  - `bpf-sim`

ORACLE

xBPF

- Experimental BPF.
- Purpose: compiler testing, BPF debugging, userland.
- `-mxbpf` in GCC and GAS.
- Current extensions:
  - Save/restored callee-saved used registers.
  - Indirect calls: `callr %reg`
- Coming extensions:
  - Signed division instruction.
  - Zero register?
  - Indirect jumps.
  - `%fp` relative addressing.
  - Remove limit on stack frame size.

Support for BTF and CO-RE

- `bpf-unknown-none-gcc -g` should generate BTF, not DWARF $\Rightarrow$ GCC requires reforms in debug hooks.
- Support for `__builtin_preserve_access_index`.

Oracle team working on this.

ORACLE

Some issues raised at LPC 2019

- `-mkernel=VERSION`
- Kernel helpers are no longer implemented with built-ins $\Rightarrow$ GCC internals are now kernel-helper agnostic.
- `jmp32` instructions are now implemented.

A few questions for the kernel hackers

Kernel Helpers in C

- `bpf_helpers.h`

```
static __u32 (*bpf_get_prandom_u32)(void) = (void *) 7;
```

  - -O0, LLVM generates invalid instruction, GCC emits an error.
  - -O2, both LLVM and GCC do the right thing.

- Function attributes

```
static __u32 (*bpf_get_prandom_u32)(void)
  __attribute__ ((kernel_helper (7)));
```

  Work with any optimization level.

Question: LLVM to adopt the `kernel_helper` attribute?

Signed division instructions

- Not supported in eBPF

```
Q: Why there is no BPF_SDIV for signed divide operation?
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
A: Because it would be rarely used. llvm errors in such case
and prints a suggestion to use unsigned divide instead.
```

  - LLVM ICEs "PLEASE submit a bug report".
  - GCC emits a compilation error.

- Breaks C much: problem for testing compiler
- Supported in xBPF

sdiv	OP_CLASS_ALU64=0xe
sdiv32	OP_CLASS_ALU=0xe
smod	OP_CLASS_ALU64=0xf
smod32	OP_CLASS_ALU=0xf

Question: will you reconsider supporting sdiv?

ORACLE

Opcode space reserved for extensions to eBPF

- Would allow variants like xBPF to be kept as superset ISA.
- Available opcodes:

Instruction Class	**Available opcodes**
`ALU`	2
`ALU64`	2
`JMP`	2
`LDX`	8
`STX`	8
`LD`	23
`ST`	28

Question: do you agree to reserve a range for extensions?
Probably split it from the LD or ST classes?