callx instruction

Dave Thaler <dave.thaler.ietf@gmail.com>

Background

- CALL | K: “call helper function by address” in immediate
- CALL | X: “call helper function by address” in register value
  - Was never supported by Linux
  - Was never part of the ISA (so not documented)
  - Clang with –O0 *does* generate callx instructions
  - Gcc with -mxbpf *also* generates callx instructions
  - Interest from other projects (PREVAIL, ebpf-for-windows) in consuming them

Simple example

- Source:

```
int func() { return bpf_get_current_pid_tgid(); }
```

- Compiled with clang using –O1 or –O2:

```
0: 85 00 00 00 0e 00 00 00 call 14
1: 95 00 00 00 00 00 00 00 exit
```

- Compiled with clang using –O0:

```
0: 18 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 r1 = 0 ll
2: 79 11 00 00 00 00 00 00 r1 = *(u64 *)(r1 + 0)
3: 8d 01 00 00 00 00 00 00 callx r1
4: 95 00 00 00 00 00 00 00 exit
```

ELF file with callx

- The `.data` section contains at offset 0000 the value 14 (0xe0)

```
.data: 0000 0e000000 00000000
```

- The `.rel.text` section specifies that at offset 0 in the .text section, to replace the 64-bit placeholder (0) with the actual 64-bit address of the `.data` section

```
RELOCATION RECORDS FOR [.text]:
OFFSET           TYPE                     VALUE
0000000000000000 R_BPF_64_64              .data
```

Program walkthrough

```
0: 18 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 r1 = 0 ll
```

- The placeholder (0) above is replaced with the address of the `.data` section

```
2: 79 11 00 00 00 00 00 00 r1 = *(u64 *)(r1 + 0)
```

- R1 now holds the contents of `.data` at offset 0, i.e., 14

```
3: 8d 00 00 00 01 00 00 00 callx r1
```

- Now call using 14 as the address, same as for “call 14”

```
4: 95 00 00 00 00 00 00 00 exit
```

How/where should we document this?

- Proposal: document it as a separate ISA extension
  - Would use the extension process defined in the ISA spec

- Handling the legacy clang bug
  - All previous clang versions put callx register # in imm instead of dst_reg
  - Clang v.19 fixes that and uses dst_reg like gcc does
  - Proposal: list CALL|X with non-zero imm as deprecated
    - To avoid collisions with potential future assignments
    - Same was done for legacy packet instructions

- Relationship to “code pointer” vs “address”:

```
r1 = code_addr(imm)  ← get code addr at relative offset in imm
callx r1             ← valid or not?
```