Instruction Set Extension Policy

draft-thaler-bpf-isa

Dave Thaler <dthaler@microsoft.com>

WG Charter

- “The BPF working group is initially tasked with … **creating a clear process for extensions**, …”

- Discussed on list in thread “[Bpf] Instruction set extension policy”

Extensions via delta docs, not replacements

- More instructions will be added over time.
- Eventual inclusion in an RFC would be good.
- Propose RFC **per extension** (set of additions)
  - Need not Obsolete (or even Update) original ISA document
- But don’t want to make additions wait for an RFC
- Proposal: allow referencing a non-RFC (e.g., Linux kernel tree file) in the meantime to get code points

Where should registry(s) be?

a) IANA
b) Files in Linux kernel tree

Policy for allocation is mostly orthogonal to where registry resides

Menu of IANA policies in RFC 8126

- Private Use
- Experimental Use
- Hierarchical Allocation
- First Come First Served
- Expert Review
- Specification Required
- RFC Required
- IETF Review
- Standards Action
- IESG Approval

URI Schemes precedent

- https://www.iana.org/assignments/uri-schemes/uri-schemes.xhtml

Range	Registration Procedures
Permanent	Expert Review
Provisional	First Come First Served
Historical	Expert Review

- URI schemes do NOT divide the space by category
- Standardization can reclassify a scheme from provisional to permanent
- RFC 8124 section 4.13 on Provisional Registrations:
  - "… If your registry does not have a practical limit on codepoints, perhaps adding the option for provisional registrations might be right for that registry as well."

Proposed policies for ISA registration

- Historical: Specification required
  - Example: legacy BPF packet access instructions (deprecated)
- Permanent: Standards action
  - Example: everything else in instruction-set.rst
- Provisional: Specification required

Option 1: Multiple key fields for BPF instructions

- BPF instructions are identified by (opcode, src, imm, offset) tuple
  - Where src, imm can be wildcards

Examples:

opcode	src	imm	offset	description
0x07	0x0	any	0	dst += imm
0x0f	any	0x00	0	dst += src
0x30	any	0x00	1	dst = (src != 0) ? (dst s/ src) : 0

Option 2: Mutiple tables

- BPF opcode table
- Separate table per opcode with multiple instructions

Opcodes:

opcode	description
0x17	dst -= imm
0x18	See **64-bit immediate instructions** registry
0x1f	dst -= src
…	

64-bit immediate instructions:

src	description
0x0	dst = imm64
0x1	dst = map_by_fd(imm)
0x2	dst = mva(map_by_fd(imm)) + next_imm
…	

Existing instructions: are all mandatory?

- Immediate instructions for maps & variables (opcode 0x18)
- Atomic instructions (opcode 0xdb)
- Call local
- Call by BTF ID

- Some runtimes don’t yet support the above categories
- Should we define one or more of them as if it were an “extension”?

Questions?

- https://github.com/ietf-wg-bpf/ebpf-docs/pull/33/files
- Contains IANA considerations text posted to mailing list