External RFC ls001.po9 Allocation of new 64-bit Power ISA Encodings </>

- https://git.openpower.foundation/isa/PowerISA/issues/64
- https://bugs.libre-soc.org/show_bug.cgi?id=1046
- Funded by NLnet under the Privacy and Enhanced Trust Programme, EU Horizon2020 Grant 825310, and NGI0 Entrust No 101069594

Severity: Major

Status: New

Date: 29 Apr 2023. v1

Target: v3.2B

Source: v3.0B

Books and Section affected:

```
New Section: 1.6.5
New Definitions: 1.3.1
```

Summary


```
Introduces a new 64-bit encoding similar to EXT1xx in PO1, in which
32 new Primary Opcodes are introduced (EXT2xx), several RESERVED spaces
(57-bit and at least three 32-bit), and the RISC-paradigm Prefixing
concepts are introduced: SVP64 and SVP64Single.
```


Submitter: Luke Leighton (Libre-SOC)

Requester: Libre-SOC

Impact on processor:

```
Addition of new "Zero-Overhead-Loop-Control" DSP-style Vector-style
Encoding concept, introduction of new 64-bit Encodings specifically
designed to be easily identifiable extremely early in Multi-Issue
systems
```

Impact on software:

```
Requires support for new instructions in assembler, debuggers, and related tools.
```

Keywords:

```
64-bit Encoding, Prefixing
```

Motivation

Power ISA Encoding is a finite precious resource that is under pressure. New Primary Opcode areas are needed (beyond those already strictly defined as EXT1xx). New Primary Opcode areas EXT232-263 allows for immediate growth, allowing Power ISA to catch up 12-15 years on Intel and ARM. Also the Simple-V RISC-paradigm “Loop” subsystem based on x86 REP and Zilog Z80 CPIR and LDIR may be cleanly and smoothly introduced. Also several new areas are RESERVED which allows significant future expansion.

Changes

Add the following entries to:

- Section 1.3.1 Book I
- Section 1.6.5 Book I

[[!tag opf_rfc]]

Definitions </>

Add the following Definitions to Section 1.3.1 of Book I

Definition of Simple-V:

In its simplest form, the Simple-V Loop/Vector concept is a Prefixing system (similar to the 8086 `REP` instruction and the Z80 `LDIR`) that both augments its following Defined Word-instruction (Suffix), and also may repeat that instruction with optional sequential register offsets from those given in the Suffix. Register numbers may also be extended (larger register files). More advanced features add predication, element-width overrides, and Vertical-First Mode.

Definition of SVP64 Prefixing:

SVP64 is a well-defined implementation of the Simple-V Loop/Vector concept, in a 32-bit Prefix format, that exploits the following instruction (the Defined Word-instruction) using it as a "template". It requires 24 bits, some of which are common to all Suffixes, and some Mode bits are specific to the Defined Word-instruction class: Load/Store-Immediate, Load/Store-Indexed, Arithmetic/Logical, Condition Register operations, and Branch-Conditional. Anything not falling into those five categories is termed "Unvectorizable".

Definition of Horizontal-First:

Normal Cray-style Vectorization, designated Horizontal-First, performs element-level operations (often in parallel) before moving in the usual fashion to the next instruction. The term "Horizontal-First" stems from naturally visually listing program instructions vertically, and register file contents horizontally, whereupon it is clear that register-elements are prioritised.

Definition of Vertical-First:

Vertical-First executes *one element operation only* then moves on to the next instruction, whereupon if that is also an SVP64-Prefixed instruction the exact same element offset is used. Element offsets are then explicitly advanced by calling a special instruction, `svstep`. The term "Vertical-First" stems from naturally visually listing program instructions vertically and register file contents horizontally, where moving to the next instruction is a clear priority.

Definition of SVP64Single Prefixing:

A 32-bit Prefix in front of a Defined Word-instruction that extends register numbers (allows larger register files), adds single-bit predication, element-width overrides, and optionally adds Saturation to Arithmetic instructions that normally would not have it. *SVP64Single is in Draft only* and is yet to be defined.

Definition of "Unvectorizable":

Any operation that inherently makes no sense if repeated (through SVP64 Prefixing) is termed "Unvectorizable". Examples include `sc` or `sync` which have no registers. `mtmsr` is also classed as Unvectorizable because there is only one `MSR`. Also instructions that simply may not be Prefixed (EXT300-EXT363) are also deemed "Unvectorizable".

Unvectorizable instructions are required to be detected as such if Prefixed (either SVP64 or SVP64Single) and an Illegal Instruction Trap raised.

Hardware Architectural Note: Given that a "pre-classification" Decode Phase is required (identifying whether the Suffix - Defined Word-instruction - is Arithmetic/Logical, CR-op, Load/Store or Branch-Conditional), adding "Unvectorizable" detection to this phase is not unreasonable.

New Prefixed Instruction Encoding space </>

Proposal: Add new Section 1.6.5 to Book I

The following eight new `RESERVED` areas are defined within Primary Opcode 9 (EXT009)

0-5	6-29	30 31	32-37	38-63	Description
PO9	xxxx	x x	010001	xxxx	RESERVED(1)
PO9	xxxx	x x	000001	xxxx	RESERVED(2)
PO9	!ZERO	1 1	!PO9	nnnn	SVP64Single:EXT200-263
PO9	0000	1 1	!PO9	nnnn	Scalar EXT200-263
PO9	SVRM	1 0	!PO9	nnnn	SVP64:EXT200-263
PO9	0000	0 1	DWd	nnnn	32-bit EXT300-363
PO9	!ZERO	0 1	DWd	nnnn	SVP64Single:EXT000-063
PO9	SVRM	0 0	DWd	nnnn	SVP64:EXT000-063

Key:

- **x** - a `RESERVED` encoding. Illegal Instruction Trap must be raised
- **n** - a future specification-defined value (currently `RESERVED`)
- **!PO9** - any 6-bit binary value except Primary Opcode 9 (0b010001)
- **!ZERO** - a non-zero future specification-defined value (currently `RESERVED`)
- **DWd** - a "Defined Word-instruction" - Book I Section 1.6 (Public v3.1 p11)
- **SVRM** - a `RESERVED` encoding
- **SVP64Single**: a future `RESERVED` Prefix encoding
- **SVP64**: a future `RESERVED` Loop-Prefix encoding
- **EXT200-263**: a `RESERVED` encoding for future Scalar instructions (Vectorizable)
- **EXT300-363**: a `RESERVED` encoding for future Scalar instructions (Unvectorizable)

Architectural Resource Allocation Note: Similar to ARM's `MOVPRFX` *instruction and the original x86 REP instruction, despite "influence" over the Suffix, the Suffix is entirely independent of the Prefix. Therefore* ***under no circumstances*** *must different Defined Word-instructions (different from the same* ***Un-Prefixed*** *Defined Word-instruction) be allocated within any* `EXT{zNN}` *prefixed or unprefixed space for a given value of* ***z*** *of 0, 2 or 3: the results would be catastrophic. Even if Unvectorizable an instruction Defined Word-instruction space* ***must*** *have the exact same Instruction and exact same Instruction Encoding in all spaces being RESERVED (Illegal Instruction Trap if Unvectorizable) or not be allocated at all. This is required as an inviolate hard rule governing Primary Opcode 9 that may not be revoked under any circumstances. A useful way to think of this is that the Prefix Encoding is, like the 8086 REP instruction, an independent 32-bit Defined Word-instruction.*

Note a particular consequence of the application of the above paragraph: due to the fact that the Prefix Encodings are independent, **by definition** two new Sandbox areas "come into being" in an **inviolate** manner (i.e. they may not be called anything else, nor may they be revoked rescinded removed or recalled), named `SVP64:EXT022` and `SVP64Single:EXT022`. The **only way** that these two new areas may be revoked is if EXT022 itself is revoked. **All and any** re-definitions modifications enhancements clarifications that apply to EXT022 **also apply to these two new areas** because due to the Prefixes being independent Defined Word-instructions the three areas are actually one and the same area, just as *all* Scalar Defined Word-instructions are.

Notes:

- **PO9**-PO1 Prefixed-Prefixed (96-bit) instructions are prohibited. EXT1xx is thus inherently Unvectorizable as the complexity at the Decoder of recognising `{PO1-word}{PO9-word}{Defined-word-instruction}` becomes too great for High Performance Multi-Issue systems.
- There is however no reason why PO9-PO1 (EXT901?) as an entirely new RESERVED 64-bit Encoding should not be permitted as long as it is clearly marked as Unvectorizable.
- PO1-**PO9** Prefixed-Prefixed (96-bit) instructions are also prohibited for the same reason: Multi-Issue Decode complexity is too great.
- There is however no reason why PO1-PO9 (EXT109) as an entirely new RESERVED 64-bit Encoding should not be permitted as long as it is clearly marked as Unvectorizable.
- EXT100-163 instructions (PO1-Prefixed) are also prohibited from being double-PO1-prefixed (not twice prefixed)
- Considerable care is needed both on Architectural Resource Allocation as well as instruction design itself. All new Scalar instructions automatically and inherently must be designed taking their Vectorizeable potential into consideration *including VSX* in future.
- Once an instruction is allocated in an Unvectorizable area it can never be Vectorized without providing an entirely new Encoding.

[[!tag standards]]
