RFC ls002 Floating-Point Load-Immediate

URLs:

- https://libre-soc.org/openpower/sv/int_fp_mv/#fmvis
- https://libre-soc.org/openpower/sv/rfc/ls002/
- https://bugs.libre-soc.org/show_bug.cgi?id=944

Severity: Major

Status: New

Date: 03 Oct 2022

Target: v3.2

Source: v3.0B

Books and Section affected:

```
Book I Scalar Floating-Point 4.6.2.1
Appendix D Power ISA sorted by opcode
Appendix E Power ISA sorted by version
Appendix F Power ISA sorted by mnemonic
```

Summary

```
Instructions added
fmvis - Floating-Point Move Immediate, Single
fishmv - Floating-Point Immediate, Second-half Move
(Potentially 64-bit prefixed of the same)
```

Submitter: Luke Leighton (Libre-SOC)

Requester: Libre-SOC

Impact on processor:

```
Addition of two new FPR-based instructions
(potentially 3 if EXT001 Prefixed variants added)
```

Impact on software:

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

Keywords:

```
FPR, Floating-point, Load-immediate, BF16, FP32
```

Motivation

Similar to `lxvkq` but extended to a full BF16 with one 32-bit instruction and a full FP32 in two 32-bit instructions these instructions always save a Data Load and associated L1 and TLB lookup. Even clearing an FPR to zero presently requires Load.

Notes and Observations:

1. There is no need for an Rc=1 variant because this is an immediate loading instruction (an FPR equivalent to `li`)
2. There is no need for Special Registers (FP Flags) because this is an immediate loading instruction. No FPR Load Operations alter `FPSCR`, neither does `lxvkq`, and on that basis neither should these instructions.
3. An EXT001 Variant which also save similar Data-Load and Data-TLB lookups are mentioned for completeness but not included as part of this RFC. Another Stakeholder with a vested interest in 64-bit Prefixed instructions may wish to consider submitting them.
4. `fishmv` as a FRS-only Read-Modify-Write (instead of an unnecessary FRS,FRA pair) saves five potential bits, making the difference between a 5-bit XO (VA/DX-Form) and requiring an entire Primary Opcode.

Changes

Add the following entries to the Appendices and instructions of Book I as a new Section 4.6.2.1

Appendices

```
Appendix D Power ISA sorted by opcode
Appendix E Power ISA sorted by version
Appendix F Power ISA sorted by mnemonic
```

Form	Book	Page	Version	mnemonic	Description
DX	I	#	3.0B	fmvis	Floating-point Move Immediate, Single
DX	I	#	3.0B	fishmv	Floating-point Immediate, Second-half Move

Floating-Point Move Immediate

`fmvis FRS, D`

0-5	6-10	11-15	16-25	26-30	31	Form
Major	FRS	d1	d0	XO	d2	DX-Form

Pseudocode:

```
bf16 = d0 || d1 || d2 # create BF16 immediate
fp32 = bf16 || [0]*16 # convert BF16 to FP32
FRS = DOUBLE(fp32)    # convert FP32 to FP64
```

Special registers altered:

```
None
```

Reinterprets `D << 16` as a 32-bit float, which is then converted to a 64-bit float and written to `FRS`. This is equivalent to reinterpreting `D` as a `BF16` and converting to 64-bit float.

Examples:

```
fmvis f4, 0 # writes +0.0 to f4 (clears an FPR)
fmvis f4, 0x8000 # writes -0.0 to f4
fmvis f4, 0x3F80 # writes +1.0 to f4
fmvis f4, 0xBFC0 # writes -1.5 to f4
fmvis f4, 0x7FC0 # writes +qNaN to f4
fmvis f4, 0x7F80 # writes +Infinity to f4
fmvis f4, 0xFF80 # writes -Infinity to f4
fmvis f4, 0x3FFF # writes +1.9921875 to f4
```

Floating-Point Immediate Second-Half Move

`fishmv FRS, D`

DX-Form:

0-5	6-10	11-15	16-25	26-30	31	Form
Major	FRS	d1	d0	XO	d2	DX-Form

Pseudocode:

```
n <- (FRS)                        # read FRS
fp32 <- SINGLE(n)                 # convert to FP32
fp32[16:31] <- d0 || d1 || d2     # replace LSB half
FRS <- DOUBLE(fp32)               # convert back to FP64
```

Special registers altered:

```
None
```

An additional 16-bits of immediate is inserted into `FRS` to extend its accuracy to a full FP32, which is then stored in the usual FP64 Format within the FPR.

This instruction performs a Read-Modify-Write. *FRS is read, the additional 16 bit immediate inserted, and the result also written to FRS. This is strategically similar to how `li` combined with `oris` is used to construct 32-bit Integers. `fishmv` may be macro-op-fused with `fmvis`*

Programmer's note: If a prior `fmvis` instruction had been used to set the upper 16-bits from an FP32 value, `fishmv` may be used to set the lower 16-bits. Example:

```
# these two combined instructions write 0x3f808000
# into f4 as an FP32 to be converted to an FP64.
# actual contents in f4 after conversion: 0x3ff0_1000_0000_0000
# first the upper bits, happens to be +1.0
fmvis f4, 0x3F80 # writes +1.0 to f4
# now write the lower 16 bits of an FP32
fishmv f4, 0x8000 # writes +1.00390625 to f4
```

[[!tag opf_rfc]]