RFC ls015 CR Weird operations </>

- Funded by NLnet under the Privacy and Enhanced Trust Programme, EU Horizon2020 Grant 825310, and NGI0 Entrust No 101069594

- https://libre-soc.org/openpower/sv/rfc/ls015/
- https://git.openpower.foundation/isa/PowerISA/issues/TODO
- https://bugs.libre-soc.org/show_bug.cgi?id=1067

Severity: Major

Status: New

Date: 25 Apr 2023

Target: v3.2B

Source: v3.1B

Books and Section affected:

```
Book I Fixed-Point Instructions
Appendix E Power ISA sorted by opcode
Appendix F Power ISA sorted by version
Appendix G Power ISA sorted by Compliancy Subset
Appendix H Power ISA sorted by mnemonic
```

Summary

```
Instructions added: crweird, crweirder, mfcrfm, mfcrweird, mtcrweird, mtcrrweird
```

Submitter: Luke Leighton (Libre-SOC)

Requester: Libre-SOC

Impact on processor:

```
Addition of new GPR-CR-based instructions
```

Impact on software:

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

Keywords:

```
CR Fields, predication, GPR
```

Motivation

Existing Condition Register operations are somewhat anaemic if to be utilised more extensively as Predicate Masks in a True-Scalable Vector ISA. Merging of multiple CR Fields requires several operations that may be achieved with a single “weird” operation, and transfer between CR Fields and GPR is easier and more powerful. This mitigates the need to add dozens of duplicate Logical Operations.

Notes and Observations:

1. TODO

Changes

Add the following entries to:

- the Appendices of Book I
- Book I 3.3.17 Condition Register Instructions
- Book I 1.6.1 and 1.6.2

Rationale </>

Condition Registers are conceptually perfect for use as predicate masks, the only problem being that typical Vector ISAs have quite comprehensive mask-based instructions: set-before-first, popcount and much more. In fact many Vector ISAs can use Vectors *as* masks, consequently the entire Vector ISA is usually available for use in creating masks (one exception being AVX512 which has a dedicated Mask regfile and opcodes). Duplication of such operations (popcount etc) is not practical for SV given the strategy of leveraging pre-existing Scalar instructions in a minimalist way.

With the scalar OpenPOWER v3.0B ISA having already popcnt, cntlz and others normally seen in Vector Mask operations it makes sense to allow *both* scalar integers *and* CR-Vectors to be predicate masks. That in turn means that much more comprehensive interaction between CRs and scalar Integers is required, because with the CR Predication Modes designating CR *Fields* (not CR bits) as Predicate Elements, fast transfers between CR *Fields* and the Integer Register File is needed.

The opportunity is therefore taken to also augment CR logical arithmetic as well, using a mask-based paradigm that takes into consideration multiple bits of each CR Field (eq/lt/gt/ov). By contrast v3.0B Scalar CR instructions (crand, crxor) only allow a single bit calculation, and both mtcr and mfcr are CR-orientated rather than CR *Field* orientated.

Also strangely there is no v3.0 instruction for directly moving CR Fields, only CR *bits*, so that is corrected here with `mcrfm`. The opportunity is taken to allow inversion of CR Field bits, when copied.

Basic concept:

- CR-based instructions that perform simple AND/OR from any four bits of a CR field to create a single bit value (0/1) in an integer register
- Inverse of the same, taking a single bit value (0/1) from an integer register to selectively target any four bits of a given CR Field
- CR-to-CR version of the same, allowing multiple bits to be AND/OR/XORed in one hit.
- Optional Vectorization of the same when SVP64 is implemented

Purpose:

- To provide a merged version of what is currently a multi-sequence of CR operations (crand, cror, crxor) with mfcr and mtcrf, reducing instruction count.
- To provide a vectorized version of the same, suitable for advanced predication

Useful side-effects:

- mtcrweird when RA=0 is a means to set or clear multiple arbitrary CR Field bits simultaneously, using immediates embedded within the instruction.
- With SVP64 on the weird instructions there is bit-for-bit interaction between GPR predicate masks (r3, r10, r31) and the source or destination GPR, in ways that are not possible with other SVP64 instructions because normal SVP64 is bit-per-element. On these weird instructions the element in effect *is* a bit.
- `mfcrweird` mitigates a need to add `conflictd`, part of {SV Vector ops}, as well as allowing more complex comparisons.

New instructions for CR/INT predication </>

Instruction Formats </>

Add the following entries to Book I 1.6.1 Word Instruction Formats:

CW-FORM </>

```
|0     |6   |9 |11|12   |16   |19  |22   |26   |31|
| PO   | RA    |M |fmsk |BF   |XO  |fmap | XO     |
| PO   | BT    |M |fmsk |BF   |XO  |fmap | XO     |
| PO   | BF |  |M |fmsk |BF   |XO  |fmap | XO     |
```

CW2-FORM </>

```
|0     |6   |9 |11|12   |16   |19  |22   |26   |31|
| PO   | RT    |M |fmsk |BFA  |XO  |fmap | XO  |Rc|
```

Add the following new fields to Book I 1.6.2 Word Instruction Fields:

```
fmap (22:25)
    Field used to specify the CR Field set/clear map for CR Weird
    instructions.
    Formats: CW, CW2

fmsk (12:15)
    Field used to specify the CR Field mask for CR Weird instructions.
    Formats: CW, CW2
```

Add `CW` and `CW2` to the `Formats:` list for all of `RT`, `RA`, `BF`, `BFA` and `Rc`.

Add `CW` to the `Formats:` list for `XO (25:30)`.

Add `CW2` to the `Formats:` list for `XO (25:31)`.

Appendices </>

```
Appendix E Power ISA sorted by opcode
Appendix F Power ISA sorted by version
Appendix G Power ISA sorted by Compliancy Subset
Appendix H Power ISA sorted by mnemonic
```

Form	Book	Page	Version	Mnemonic	Description
CW2	I	#	3.2B	crrweird	
CW2	I	#	3.2B	mfcrweird	
CW	I	#	3.2B	mtcrrweird	
CW	I	#	3.2B	mtcrweird	
CW	I	#	3.2B	crweirder	
CW	I	#	3.2B	mcrfm	

[[!tag opf_rfc]]