RISCV – Decimal Floating-Point Extension Proposal (L)

RISCV ISA specification. Working draft, subject to change.

**Contributors:** Robert T. Finch

# Introduction

Decimal floating-point provides the capacity to maintain rounded decimal numbers as is commonly used in everyday calculations. It has applications that overlap with binary floating-point and may be more desirable for some applications such as finance or engineering. It provides the ability to easily track the number of significant decimal digits. Decimal floating-point is used in calculator type applications.

# Assumptions

There are a limited number of applications requiring decimal floating-point in a system. It is preferable to use binary floating-point to get better application performance when performance is more critical than decimal digit capability.

Decimal floating-point and binary floating-point will not be mixed in the same application.

Opcode space has a cost. Replicating opcodes to support a different floating-point format is wasteful in the author’s opinion.

It may be desirable to perform some decimal floating-point operations using software. Whether an operation is implemented in hardware or software will be invisible to the application. Software implemented operations will be triggered using opcodes that are not implemented with hardware.

There may be a heterogeneous mix of floating-point formats in a multi-core system. It is assumed that a core will support only one format.

Performance of decimal floating-point is less critical than the capacity to use decimal numbers.

Lower precision decimal-floating point operations are not required. Only double or quad precision will be required.

IEEE standard decimal floating-point formats will be used.

# Proposal

We propose:

Reusing the existing floating-point opcodes for decimal floating-point operations where possible. Specifications already state that load and store operations will use existing instructions and floating-point registers. The proposal is to extend the specification to apply to opcodes as well. There will be an indicator in a floating-point control register that indicates the presence of decimal floating-point instead of binary floating-point.

# Rationale

Reusing existing opcodes conserves opcode space. It may also provide leverage for use in toolsets and libraries.

Under the assumption that binary and decimal floating-point are not used in the same application the decimal floating-point indicator allows software to choose what to do if the pertinent floating-point format is not available for the application.

Performing some operations in software may offer a better solution than a pure hardware implementation. If one is using decimal floating-point consideration of performance may have a lower priority than that of binary floating-point applications. If extreme performance is needed then decimal floating-point is likely not the solution.

For instance, decimal floating-point divide may have a large latency depending on the implementation and may be performed almost as fast in software. So that interrupt latency is not adversely affected some operations may be performed in a stepwise fashion using software.

Because of its use in finance and engineering lower precision formats are not generally needed. When one is after decimal floating-point one wants many significant digits.

# Nomenclature

The suggested nomenclature for decimal floating-point operations is to prefix the operation with a ‘DF’ in place of the ‘F’ used as an indicator of floating-point operations.

# Operations

Operations mirror the operations available with binary floating-point. Basic operations include DFADD, DFSUB, DFMUL and DFDIV and others. The opcode formats for all operations are not described in this document instead you are referred to the RISCV specification documentation. Only the operations unique to decimal floating-point are described herein.

# Software Impact

Trap handlers for decimal floating-point instructions implemented with software will need to be written. The exception code 2 – Illegal Instruction Trap would be used.

Hardware Impact