

Computer Organization
Floating Point Coprocessor

Instruction Set Architecture

Guillermo Briceño
Rene Nicklich
Austin Lantz

February 28, 2019

1 Introduction

This document describes the instruction set architecture for a 32-bit floating-point coprocessor. Each instruction is described with its assembler syntax in section 2.

1.1 Instruction Encoding

The twenty-three instructions can be grouped by their instruction encoding formats. The keywords **rd**, **rs1**, and **rs2** stand for the 4-bit register addresses from 0000_2 to 1111_2 , corresponding to a destination register, the first source register, and second source register respectively. In single-source or immediate mode addressing, the keyword **rs** is used instead. Bit 31 is used to specify register or immediate addressing mode for double-operand ALU instructions.

Double-Operand ALU		30	25	21	17	13	0
	r-type:	0	opcode	rd	rs1	rs2	0
	i-type:	1	opcode	rd	rs	18-bit immediate	
Single-Operand ALU	EXP:	0	opcode	rd	22-bit immediate		
		0	opcode	rd	0		
Load, Store, Move		0	opcode	rd	rs	0	
Branching	B:	0	opcode	rd	22-bit immediate		
		0	opcode	rd	0		
No-Operation and Halt							
		0	opcode	0			

2 Instruction Set Specification

NOP No operation

Opcode: 00000
 Syntax: NOP
 Purpose: Perform no operations.

SET Set register to floating-point value

Opcode: 00001
 Syntax r-type: SET rd, #<32-bit FP value>
 Purpose: Assign a 32-bit floating point value to rd.
 Operation: $rd \leftarrow \text{FPvalue}$

LOAD Load value from memory

Opcode: 00010
 Syntax r-type: LOAD rd, rs
 Purpose: Assign rd the value from the memory address in rs.
 Operation: $rd \leftarrow M[rs]$

STORE Store value to memory

Opcode: 00011
 Syntax r-type: STORE rd, rs
 Purpose: Assign memory location specified in rd to value in rs.
 Operation: $M[rd] \leftarrow rs$

MOVE Copy value from a register to another

Opcode: 00100
 Syntax r-type: MOVE rd, rs
 Purpose: Assign rd the value in rs.
 Operation: $rd \leftarrow rs$

FADD

Add 32-bit floating-point value

Opcode:	00101
Syntax r-type:	FADD rd, rs1, rs2
Syntax i-type:	FADD rd, rs1, #<18-bit immediate>
Purpose:	Performs addition on two 32-bit floating-point values from rs1 and rs2, or an immediate in place of rs2, and stores the result in rd.
Operation:	$rd \leftarrow rs1 + rs2$ or $rd \leftarrow rs1 + \text{immediate}$
Condition Codes:	$\begin{array}{c} C \ N \ Z \ V \\ \hline x \ x \ x \ x \end{array}$

FSUB

Subtract 32-bit floating-point values

Opcode:	00110
Syntax r-type:	FSUB rd, rs1, rs2
Syntax i-type:	FSUB rd, rs1, #<18-bit immediate>
Purpose:	Performs subtraction on two 32-bit floating-point values from rs1 and rs2, or an immediate in place of rs2, and stores the result in rd.
Operation:	$rd \leftarrow rs1 - rs2$ or $rd \leftarrow rs1 - \text{immediate}$
Condition Codes:	$\begin{array}{c} C \ N \ Z \ V \\ \hline x \ x \ x \ x \end{array}$

FNEG

Negate a 32-bit floating-point value

Opcode:	00111
Syntax:	FNEG rd, rs
Purpose:	Performs negation on a 32-bit floating-point value from rs and stores the result in rd.
Operation:	$rd \leftarrow -rs$
Condition Codes:	$\begin{array}{c} C \ N \ Z \ V \\ \hline - \ x \ x \ - \end{array}$

FMUL

Multiply two 32-bit floating-point values

Opcode:	01000
Syntax r-type:	FMUL rd, rs1, rs2
Syntax i-type:	FMUL rd, rs1, #<18-bit immediate>
Purpose:	Performs multiplication on two 32-bit floating-point values from rs1 and rs2, or an immediate in place of rs2, and stores the result in rd.
Operation:	$rd \leftarrow rs1 * rs2$ or $rd \leftarrow rs1 * \text{immediate}$
Condition Codes:	$\begin{array}{c} C \ N \ Z \ V \\ \hline x \ x \ x \ x \end{array}$

FDIV

Divide two 32-bit floating-point values

Opcode:	01001
Syntax r-type:	FDIV rd, rs1, rs2
Syntax i-type:	FDIV rd, rs1, #<18-bit immediate>
Purpose:	Performs division on two 32-bit floating-point values from rs1 and rs2, or an immediate in place of rs2, and stores the result in rd.
Operation:	$rd \leftarrow rs1 \div rs2$ or $rd \leftarrow rs1 \div \text{immediate}$
Condition Codes:	$\begin{array}{c} C \ N \ Z \ V \\ \hline x \ x \ x \ x \end{array}$

FLOOR

Compute the floor function

Opcode:	01010
Syntax r-type:	FLOOR rd, rs
Purpose:	Rounds the value in rs to the nearest lowest integer and stores the result in rd.
Operation:	$rd \leftarrow \lfloor rs \rfloor$
Condition Codes:	$\begin{array}{c} C \ N \ Z \ V \\ \hline - \ x \ x \ - \end{array}$

CEIL

Compute the ceiling function

Opcode:	01011
Syntax:	CEIL rd, rs
Purpose:	Rounds the value in rs to the nearest highest integer and stores the result in rd.
Operation:	$rd \leftarrow \lceil rs \rceil$
Condition Codes:	$\begin{array}{c} C \ N \ Z \ V \\ \hline - \ x \ x \ - \end{array}$

ROUND

Round a value

Opcode:	01100
Syntax:	ROUND rd, rs
Purpose:	Rounds the value in rs and stores the result in rd.
Operation:	$rd \leftarrow \text{round}(rs)$
Condition Codes:	$\begin{array}{c} C \ N \ Z \ V \\ \hline - \ x \ x \ x \end{array}$

FABS

Compute the absolute value

Opcode:	01101
Syntax:	FABS rd, rs
Purpose:	Find the absolute value in rs and stores the result in rd.
Operation:	$rd \leftarrow rs $
Condition Codes:	$\begin{array}{c} C \ N \ Z \ V \\ \hline - \ - \ x \ - \end{array}$

MIN

Find the smallest value

Opcode:	01110
Syntax:	MIN rd, rs1, rs2
Purpose:	Finds the smallest value between rs1 and rs2 and stores the result in rd.
Operation:	$rd \leftarrow \min(rs1, rs2)$
Condition Codes:	$\begin{array}{c} C \ N \ Z \ V \\ \hline - \ x \ - \ - \end{array}$

MAX

Find the largest

Opcode:	01111								
Syntax:	MAX rd, rs1, rs2								
Purpose:	Finds the largest value between rs1 and rs2 and store the result in rd.								
Operation:	$rd \leftarrow \max(rs1, rs2)$								
Condition Codes:	<table><tr><td>C</td><td>N</td><td>Z</td><td>V</td></tr><tr><td>-</td><td>x</td><td>-</td><td>-</td></tr></table>	C	N	Z	V	-	x	-	-
C	N	Z	V						
-	x	-	-						

POW

Compute the power

Opcode:	10000								
Syntax:	POW rd, rs, #<22-bit integer-value>								
Purpose:	Finds rs1 raised to an integer value and stores the result in rd.								
Operation:	$rd \leftarrow rs^{\text{integer-value}}$								
Condition Codes:	<table><tr><td>C</td><td>N</td><td>Z</td><td>V</td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	C	N	Z	V	x	x	x	x
C	N	Z	V						
x	x	x	x						

EXP

Compute the exponent

Opcode:	10001								
Syntax:	EXP rd, rs								
Purpose:	Finds e raised to the value in rs and stores the result in rd.								
Operation:	$rd \leftarrow e^{rs}$								
Condition Codes:	<table><tr><td>C</td><td>N</td><td>Z</td><td>V</td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	C	N	Z	V	x	x	x	x
C	N	Z	V						
x	x	x	x						

SQRT

Compute the square root of a value

Opcode:	10010								
Syntax:	SQRT rd, rs								
Purpose:	Finds the square root of the value in rs and stores the result in rd.								
Operation:	$rd \leftarrow \sqrt{rs}$								
Condition Codes:	<table><tr><td>C</td><td>N</td><td>Z</td><td>V</td></tr><tr><td>-</td><td>x</td><td>-</td><td>-</td></tr></table>	C	N	Z	V	-	x	-	-
C	N	Z	V						
-	x	-	-						

B

Branch unconditionally

Opcode:	10011
Syntax:	B rd
Purpose:	Set the program counter to the value in memory addressed by rd.
Operation:	$PC \leftarrow M[rd]$

BZ

Branch if zero

Opcode:	10100
Syntax:	BZ rd, <LABEL>
Purpose:	Branch to the label specified in the assembly program if rd is equal to zero.
Operation:	if (rd == 0): $PC \leftarrow LABEL$

BNBranch if negative

Opcode: 10101

Syntax: BN rd, <LABEL>

Purpose: Branch to the label specified in the assembly program if
rd is less than zero.Operation: if (rd < 0):
PC \leftarrow LABEL**HALT**Stop program

Opcode: 10110

Syntax: HALT

Purpose: Stop the program.