1 Instruction Set Reference List

		31	25	21	17	13	0
Formats	r-type:	opcode	rd	rs1	rs2	0	
	i-type:	opcode	rd	rs	18-bit immediate		
	l-type:	opcode	rd	22-bit immediate			

Instructions	NOP	000000	0			
	SET	000001	rd	rs	18-bit immediate	
	LOAD	000010	rd	22-bit immediate		
	MOV	000011	rd	rs	0	
	FADD	000011	rd	rs1	rs2	0
	FSUB	000100	rd	rs1	rs2	0
	NEG	000101	rd	rs1	0	0

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 ← 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 ← 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] ← 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 ← rs

FADD

Add 32-bit floating-point value

Opcode: `00101`

Syntax r-type: `FADD rd, rs1, rs2`

Syntax i-type: `FADD rd, rs1, #<32-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 ← rs1 + rs2 or rd ← rs1 + immediate

Condition Codes:

N	Z	V
x	x	x

FSUB

Subtract 32-bit floating-point values

Opcode: `00110`

Syntax r-type: `FSUB rd, rs1, rs2`

Syntax i-type: `FSUB rd, rs1, #<32-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 ← rs1 - rs2 or rd ← rs1 - immediate

Condition Codes:

N	Z	V
x	x	x

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 ← -rs

Condition Codes:

N	Z	V
x	x	-

FMUL

Multiply two 32-bit floating-point values

Opcode: `01000`

Syntax r-type: `FMUL rd, rs1, rs2`

Syntax i-type: `FMUL rd, rs1, #<32-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 ← rs1 ∗ rs2 or rd ← rs1 ∗ immediate

Condition Codes:

N	Z	V
x	x	x

FDIV

Divide two 32-bit floating-point values

Opcode: `01001`

Syntax r-type: `FDIV rd, rs1, rs2`

Syntax i-type: `FDIV rd, rs1, #<32-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 ← rs1 ÷ rs2 or rd ← rs1 ÷ immediate

Condition Codes:

N	Z	V
x	x	x

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 ← $\lfloor \text{rs} \rfloor$

Condition Codes:

N	Z	V
x	x	x

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 ← ⌈rs⌉

Condition Codes:

N	Z	V
x	-	x

ROUND

Round a value

Opcode: `01100`

Syntax: `ROUND rd, rs`

Purpose: Rounds the value in rs and stores the result in rd.

Operation: rd ← round(rs)

Condition Codes:

N	Z	V
x	-	x

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 ← |rs|

Condition Codes:

N	Z	V
x	-	x

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 ← min(rs1, rs2)

Condition Codes:

N	Z	V
x	-	x

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 ← max(rs1, rs2)

Condition Codes:

N	Z	V
x	-	x

POW

Compute the power

Opcode: `10000`

Syntax: `POW rd, rs, #<integer-value>`

Purpose: Finds rs1 raised to an integer value and stores the result in rd.

Operation: rd ← $\mathrm{rs}^{\text{integer-value}}$

Condition Codes:

N	Z	V
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 ← e^{rs}

Condition Codes:

N	Z	V
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: $\text{rd} \leftarrow \sqrt{\text{rs}}$

Condition Codes:

N	Z	V
x	-	x

B

Branch unconditionally

Opcode: `10011`

Syntax: `B rd`

Purpose: Set the program counter to the value in memory addressed by rd.

Operation: $\text{PC} \leftarrow \text{M[rd]}$

Condition Codes:

N	Z	V
x	-	x

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 ← LABEL
```

Condition Codes:

N	Z	V
x	-	x

BN

Branch 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 ← LABEL
```

Condition Codes:

N	Z	V
x	-	x

HALT

Stop program

Opcode: `10110`

Syntax: `HALT`

Purpose: Stop the program.