DCC006: Organização de computadores I **(Entrega: 05/10/2020)**

Trabalho Prático #2

Professor: Omar Paranaíba Vilela Neto *Monitor:* Pedro Arthur R. L. Silva

Antes de começar seu trabalho, leia todas as instruções abaixo.

- O trabalho pode ser feito em grupos compostos por até 3 alunos.
- Cópias de trabalho acarretarão em devida penalização às partes envolvidas.
- Entregas após o prazo serão aceitas, porém haverá uma penalização. Quanto maior o atraso maior a penalização.
- O objetivo desse trabalho é te familiarizar com a Linguagem de Descrição de *Hardware* ***Verilog***, utilizando para isso uma versão modificada do simulador *DigitalJS*, que pode ser encontrada através do seguinte link: **150.164.9.98:4200**. Essa versão modificada do simulador já contém a implementação do caminho de dados do RISC-V com ciclo único. As suas tarefas nesse trabalho consistirão em alterar o caminho de dados fornecido a fim de incluir mais operações e módulos.
- Você deve entregar um único arquivo zip, contendo o arquivo *Verilog* com a implementação do caminho de dados com as funções pedidas a seguir. Note que todas as funções devem estar no mesmo caminho de dados, ou seja, o trabalho é incremental, você deve entregar **somente um caminho de dados contendo todas as funções solicitadas**.
- No mesmo arquivo zip contendo o caminho de dados, você deve enviar um relatório, em pdf, explicando suas decisões de projeto e contendo **nome e matrícula de todos os integrantes do grupo.**
- **Cada grupo deve fazer somente uma submissão**. Ou seja, cada grupo terá um aluno responsável por fazer a submissão do trabalho no *Moodle*.

Problema 1: ORI - *Bitwise or immediate* (2.0 pontos)

#Para mais informações sobre o funcionamento dessa instrução confira a documentação do RISC-V
https://github.com/riscv/riscv-isa-manual/releases/latest

Problema 2: SLLI - *Shift Left Logical Immediate* (2.0 pontos)

#Para mais informações sobre o funcionamento dessa instrução confira a documentação do RISC-V
https://github.com/riscv/riscv-isa-manual/releases/latest

Problema 3: LUI - *Load Upper Immediate* (2.0 pontos)

#Para mais informações sobre o funcionamento dessa instrução confira a documentação do RISC-V
https://github.com/riscv/riscv-isa-manual/releases/latest

Problema 4: LWI - *Load With Increment* (2.0 pontos)

Implemente a seguinte instrução:

lwi rd, rs1, rs2

Interpretação: *Reg[rd] = Mem[Reg[rs1] + Reg[rs2]]*

#Atribua os valores que achar necessário para implementação da função, mas lembre-se de documentar bem as suas decisões

#Esse problema é baseado na **questão 4.11 do livro *Computer Organization and Design, RISC - V Edition***, que é parte da bibliografia utilizada na disciplina.

Problema 5: SWAP (2.0 pontos)

Implemente a seguinte instrução:

swap rs1, rs2

Interpretação: *Reg[rs1] = Reg[rs2]; Reg[rs2] = Reg[rs1]*

#Atribua os valores que achar necessário para implementação da função, mas lembre-se de documentar bem as suas decisões

#Esse problema é baseado na **questão 4.12 do livro *Computer Organization and Design, RISC - V Edition***, que é parte da bibliografia utilizada na disciplina.

Problema 6: SS - *Store Sum* (2.0 pontos)

Implemente a seguinte instrução:

ss rs1, rs2, imm

Interpretação: *Mem[Reg[rs1]] = Reg[rs2] + imm*

#Atribua os valores que achar necessário para implementação da função, mas lembre-se de documentar bem as suas decisões

#Esse problema é baseado na **questão 4.13 do livro *Computer Organization and Design, RISC - V Edition***, que é parte da bibliografia utilizada na disciplina.

Problema 7: BLT - *Branch on Less Than* (2.0 pontos)

#Para mais informações sobre o funcionamento dessa instrução confira a documentação do RISC-V **https://github.com/riscv/riscv-isa-manual/releases/latest**

Problema 8: BGE - *Branch on Greater Than or Equal* (2.0 pontos)

#Para mais informações sobre o funcionamento dessa instrução confira a documentação do RISC-V **https://github.com/riscv/riscv-isa-manual/releases/latest**

Problema 9: J - *Jump* (4.0 pontos)

#Para mais informações sobre o funcionamento dessa instrução confira a documentação do RISC-V **https://github.com/riscv/riscv-isa-manual/releases/latest**

31	25	24 20	19 15	14 12	11 7	6 0	
imm[31:12]					rd	opcode	**Type-U**
imm[20|10:1|11|19:12]					rd	opcode	**Type-UJ**
imm[11:0]			rs1	funct3	rd	opcode	**Type-I**
imm[12|10:5]		rs2	rs1	funct3	imm[4:1|11]	opcode	**Type-SB**
imm[11:5]		rs2	rs1	funct3	imm[4:0]	opcode	**Type-S**
funct5	funct2	rs2	rs1	funct3	rd	opcode	**Type-R**

RV32I Base Integer Instruction Set

simm[31:12]					rd	0110111	LUI rd, imm
simm[31:12]					rd	0010111	AUIPC rd, offset
simm[20|10:1|11|19:12]					rd	1101111	JAL rd, offset
simm[11:0]			rs1	000	rd	1100111	JALR rd, rs1, offset
simm[12|10:5]		rs2	rs1	000	simm[4:1|11]	1100011	BEQ rs1, rs2, offset
simm[12|10:5]		rs2	rs1	001	simm[4:1|11]	1100011	BNE rs1, rs2, offset
simm[12|10:5]		rs2	rs1	100	simm[4:1|11]	1100011	BLT rs1, rs2, offset
simm[12|10:5]		rs2	rs1	101	simm[4:1|11]	1100011	BGE rs1, rs2, offset
simm[12|10:5]		rs2	rs1	110	simm[4:1|11]	1100011	BLTU rs1, rs2, offset
simm[12|10:5]		rs2	rs1	111	simm[4:1|11]	1100011	BGEU rs1, rs2, offset
simm[11:0]			rs1	000	rd	0000011	LB rd, offset(rs1)
simm[11:0]			rs1	001	rd	0000011	LH rd, offset(rs1)
simm[11:0]			rs1	010	rd	0000011	LW rd, offset(rs1)
simm[11:0]			rs1	100	rd	0000011	LBU rd, offset(rs1)
simm[11:0]			rs1	101	rd	0000011	LHU rd, offset(rs1)
simm[11:5]		rs2	rs1	000	simm[4:0]	0100011	SB rs2, offset(rs1)
simm[11:5]		rs2	rs1	001	simm[4:0]	0100011	SH rs2, offset(rs1)
simm[11:5]		rs2	rs1	010	simm[4:0]	0100011	SW rs2, offset(rs1)
simm[11:0]			rs1	000	rd	0010011	ADDI rd, rs1, imm
simm[11:0]			rs1	010	rd	0010011	SLTI rd, rs1, imm
simm[11:0]			rs1	011	rd	0010011	SLTIU rd, rs1, imm
simm[11:0]			rs1	100	rd	0010011	XORI rd, rs1, imm
simm[11:0]			rs1	110	rd	0010011	ORI rd, rs1, imm
simm[11:0]			rs1	111	rd	0010011	ANDI rd, rs1, imm
00000	00	shamt[4:0]	rs1	001	rd	0010011	SLLI rd, rs1, imm
00000	00	shamt[4:0]	rs1	101	rd	0010011	SRLI rd, rs1, imm
01000	00	shamt[4:0]	rs1	101	rd	0010011	SRAI rd, rs1, imm
00000	00	rs2	rs1	000	rd	0110011	ADD rd, rs1, rs2
01000	00	rs2	rs1	000	rd	0110011	SUB rd, rs1, rs2
00000	00	rs2	rs1	001	rd	0110011	SLL rd, rs1, rs2
00000	00	rs2	rs1	010	rd	0110011	SLT rd, rs1, rs2
00000	00	rs2	rs1	011	rd	0110011	SLTU rd, rs1, rs2
00000	00	rs2	rs1	100	rd	0110011	XOR rd, rs1, rs2
00000	00	rs2	rs1	101	rd	0110011	SRL rd, rs1, rs2
01000	00	rs2	rs1	101	rd	0110011	SRA rd, rs1, rs2
00000	00	rs2	rs1	110	rd	0110011	OR rd, rs1, rs2
00000	00	rs2	rs1	111	rd	0110011	AND rd, rs1, rs2
0000 pred	pred	pred succ	00000	000	00000	0001111	FENCE pred, succ
0000000		00000	00000	001	00000	0001111	FENCE.I

<table>
<tr><td colspan="3">31 25</td><td>24 20</td><td>19 15</td><td>14 12</td><td>11 7</td><td>6 0</td><td></td></tr>
<tr><td colspan="4">imm[11:0]</td><td>rs1</td><td>funct3</td><td>rd</td><td>opcode</td><td><b>Type-I</b></td></tr>
<tr><td colspan="3">imm[11:5]</td><td>rs2</td><td>rs1</td><td>funct3</td><td>imm[4:0]</td><td>opcode</td><td><b>Type-S</b></td></tr>
<tr><td>funct5</td><td colspan="2">funct2</td><td>rs2</td><td>rs1</td><td>funct3</td><td>rd</td><td>opcode</td><td><b>Type-R</b></td></tr>
</table>

RV64I Base Integer Instruction Set (in addition to RV32I)

<table>
<tr><td colspan="4">simm[11:0]</td><td>rs1</td><td>110</td><td>rd</td><td>0000011</td><td>LWU rd, offset(rs1)</td></tr>
<tr><td colspan="4">simm[11:0]</td><td>rs1</td><td>011</td><td>rd</td><td>0000011</td><td>LD rd, offset(rs1)</td></tr>
<tr><td colspan="3">simm[11:5]</td><td>rs2</td><td>rs1</td><td>011</td><td>simm[4:0]</td><td>0100011</td><td>SD rs2, offset(rs1)</td></tr>
<tr><td>00000</td><td>0</td><td colspan="2">shamt[5:0]</td><td>rs1</td><td>001</td><td>rd</td><td>0010011</td><td>SLLI rd, rs1, imm</td></tr>
<tr><td>00000</td><td>0</td><td colspan="2">shamt[5:0]</td><td>rs1</td><td>101</td><td>rd</td><td>0010011</td><td>SRLI rd, rs1, imm</td></tr>
<tr><td>01000</td><td>0</td><td colspan="2">shamt[5:0]</td><td>rs1</td><td>101</td><td>rd</td><td>0010011</td><td>SRAI rd, rs1, imm</td></tr>
<tr><td colspan="4">simm[11:0]</td><td>rs1</td><td>000</td><td>rd</td><td>0011011</td><td>ADDIW rd, rs1, imm</td></tr>
<tr><td colspan="3">0000000</td><td>shamt[4:0]</td><td>rs1</td><td>001</td><td>rd</td><td>0011011</td><td>SLLIW rd, rs1, imm</td></tr>
<tr><td colspan="3">0000000</td><td>shamt[4:0]</td><td>rs1</td><td>101</td><td>rd</td><td>0011011</td><td>SRLIW rd, rs1, imm</td></tr>
<tr><td colspan="3">0100000</td><td>shamt[4:0]</td><td>rs1</td><td>101</td><td>rd</td><td>0011011</td><td>SRAIW rd, rs1, imm</td></tr>
<tr><td>00000</td><td colspan="2">00</td><td>rs2</td><td>rs1</td><td>000</td><td>rd</td><td>0111011</td><td>ADDW rd, rs1, rs2</td></tr>
<tr><td>01000</td><td colspan="2">00</td><td>rs2</td><td>rs1</td><td>000</td><td>rd</td><td>0111011</td><td>SUBW rd, rs1, rs2</td></tr>
<tr><td>00000</td><td colspan="2">00</td><td>rs2</td><td>rs1</td><td>001</td><td>rd</td><td>0111011</td><td>SLLW rd, rs1, rs2</td></tr>
<tr><td>00000</td><td colspan="2">00</td><td>rs2</td><td>rs1</td><td>101</td><td>rd</td><td>0111011</td><td>SRLW rd, rs1, rs2</td></tr>
<tr><td>01000</td><td colspan="2">00</td><td>rs2</td><td>rs1</td><td>101</td><td>rd</td><td>0111011</td><td>SRAW rd, rs1, rs2</td></tr>
</table>

RV128I Base Integer Instruction Set (in addition to RV64I)

<table>
<tr><td colspan="4">simm[11:0]</td><td>rs1</td><td>111</td><td>rd</td><td>0000011</td><td>LDU rd, offset(rs1)</td></tr>
<tr><td colspan="4">simm[11:0]</td><td>rs1</td><td>010</td><td>rd</td><td>0001111</td><td>LQ rd, offset(rs1)</td></tr>
<tr><td colspan="3">simm[11:5]</td><td>rs2</td><td>rs1</td><td>100</td><td>simm[4:0]</td><td>0100011</td><td>SQ rs2, offset(rs1)</td></tr>
<tr><td>00000</td><td colspan="3">shamt[6:0]</td><td>rs1</td><td>001</td><td>rd</td><td>0010011</td><td>SLLI rd, rs1, imm</td></tr>
<tr><td>00000</td><td colspan="3">shamt[6:0]</td><td>rs1</td><td>101</td><td>rd</td><td>0010011</td><td>SRLI rd, rs1, imm</td></tr>
<tr><td>01000</td><td colspan="3">shamt[6:0]</td><td>rs1</td><td>101</td><td>rd</td><td>0010011</td><td>SRAI rd, rs1, imm</td></tr>
<tr><td colspan="4">simm[11:0]</td><td>rs1</td><td>000</td><td>rd</td><td>1011011</td><td>ADDID rd, rs1, imm</td></tr>
<tr><td colspan="2">000000</td><td colspan="2">shamt[5:0]</td><td>rs1</td><td>001</td><td>rd</td><td>1011011</td><td>SLLID rd, rs1, imm</td></tr>
<tr><td colspan="2">000000</td><td colspan="2">shamt[5:0]</td><td>rs1</td><td>101</td><td>rd</td><td>1011011</td><td>SRLID rd, rs1, imm</td></tr>
<tr><td colspan="2">010000</td><td colspan="2">shamt[5:0]</td><td>rs1</td><td>101</td><td>rd</td><td>1011011</td><td>SRAID rd, rs1, imm</td></tr>
<tr><td>00000</td><td colspan="2">00</td><td>rs2</td><td>rs1</td><td>000</td><td>rd</td><td>1111011</td><td>ADDD rd, rs1, rs2</td></tr>
<tr><td>01000</td><td colspan="2">00</td><td>rs2</td><td>rs1</td><td>000</td><td>rd</td><td>1111011</td><td>SUBD rd, rs1, rs2</td></tr>
<tr><td>00000</td><td colspan="2">00</td><td>rs2</td><td>rs1</td><td>001</td><td>rd</td><td>1111011</td><td>SLLD rd, rs1, rs2</td></tr>
<tr><td>00000</td><td colspan="2">00</td><td>rs2</td><td>rs1</td><td>101</td><td>rd</td><td>1111011</td><td>SRLD rd, rs1, rs2</td></tr>
<tr><td>01000</td><td colspan="2">00</td><td>rs2</td><td>rs1</td><td>101</td><td>rd</td><td>1111011</td><td>SRAD rd, rs1, rs2</td></tr>
</table>

RV32M Standard Extension for Integer Multiply and Divide

00000	01	rs2	rs1	000	rd	0110011	MUL rd, rs1, rs2
00000	01	rs2	rs1	001	rd	0110011	MULH rd, rs1, rs2
00000	01	rs2	rs1	010	rd	0110011	MULHSU rd, rs1, rs2
00000	01	rs2	rs1	011	rd	0110011	MULHU rd, rs1, rs2
00000	01	rs2	rs1	100	rd	0110011	DIV rd, rs1, rs2
00000	01	rs2	rs1	101	rd	0110011	DIVU rd, rs1, rs2
00000	01	rs2	rs1	110	rd	0110011	REM rd, rs1, rs2
00000	01	rs2	rs1	111	rd	0110011	REMU rd, rs1, rs2

31	25	24 20	19 15	14 12	11 7	6 0	
funct5	funct2	rs2	rs1	funct3	rd	opcode	**Type-R**

RV64M Standard Extension for Integer Multiply and Divide (in addition to RV32M)

00000	01	rs2	rs1	000	rd	0111011	MULW rd, rs1, rs2
00000	01	rs2	rs1	100	rd	0111011	DIVW rd, rs1, rs2
00000	01	rs2	rs1	101	rd	0111011	DIVUW rd, rs1, rs2
00000	01	rs2	rs1	110	rd	0111011	REMW rd, rs1, rs2
00000	01	rs2	rs1	111	rd	0111011	REMUW rd, rs1, rs2

RV128M Standard Extension for Integer Multiply and Divide (in addition to RV64M)

00000	01	rs2	rs1	000	rd	1111011	MULD rd, rs1, rs2
00000	01	rs2	rs1	100	rd	1111011	DIVD rd, rs1, rs2
00000	01	rs2	rs1	101	rd	1111011	DIVUD rd, rs1, rs2
00000	01	rs2	rs1	110	rd	1111011	REMD rd, rs1, rs2
00000	01	rs2	rs1	111	rd	1111011	REMUD rd, rs1, rs2

RV32A Standard Extension for Atomic Instructions

00010	aq	rl	00000	rs1	010	rd	0101111	LR.W aqrl, rd, (rs1)
00011	aq	rl	rs2	rs1	010	rd	0101111	SC.W aqrl, rd, rs2, (rs1)
00001	aq	rl	rs2	rs1	010	rd	0101111	AMOSWAP.W aqrl, rd, rs2, (rs1)
00000	aq	rl	rs2	rs1	010	rd	0101111	AMOADD.W aqrl, rd, rs2, (rs1)
00100	aq	rl	rs2	rs1	010	rd	0101111	AMOXOR.W aqrl, rd, rs2, (rs1)
01000	aq	rl	rs2	rs1	010	rd	0101111	AMOOR.W aqrl, rd, rs2, (rs1)
01100	aq	rl	rs2	rs1	010	rd	0101111	AMOAND.W aqrl, rd, rs2, (rs1)
10000	aq	rl	rs2	rs1	010	rd	0101111	AMOMIN.W aqrl, rd, rs2, (rs1)
10100	aq	rl	rs2	rs1	010	rd	0101111	AMOMAX.W aqrl, rd, rs2, (rs1)
11000	aq	rl	rs2	rs1	010	rd	0101111	AMOMINU.W aqrl, rd, rs2, (rs1)
11100	aq	rl	rs2	rs1	010	rd	0101111	AMOMAXU.W aqrl, rd, rs2, (rs1)

RV64A Standard Extension for Atomic Instructions (in addition to RV32A)

00010	aq	rl	00000	rs1	011	rd	0101111	LR.D aqrl, rd, (rs1)
00011	aq	rl	rs2	rs1	011	rd	0101111	SC.D aqrl, rd, rs2, (rs1)
00001	aq	rl	rs2	rs1	011	rd	0101111	AMOSWAP.D aqrl, rd, rs2, (rs1)
00000	aq	rl	rs2	rs1	011	rd	0101111	AMOADD.D aqrl, rd, rs2, (rs1)
00100	aq	rl	rs2	rs1	011	rd	0101111	AMOXOR.D aqrl, rd, rs2, (rs1)
01000	aq	rl	rs2	rs1	011	rd	0101111	AMOOR.D aqrl, rd, rs2, (rs1)
01100	aq	rl	rs2	rs1	011	rd	0101111	AMOAND.D aqrl, rd, rs2, (rs1)
10000	aq	rl	rs2	rs1	011	rd	0101111	AMOMIN.D aqrl, rd, rs2, (rs1)
10100	aq	rl	rs2	rs1	011	rd	0101111	AMOMAX.D aqrl, rd, rs2, (rs1)
11000	aq	rl	rs2	rs1	011	rd	0101111	AMOMINU.D aqrl, rd, rs2, (rs1)
11100	aq	rl	rs2	rs1	011	rd	0101111	AMOMAXU.D aqrl, rd, rs2, (rs1)

31	25	24 20	19 15	14 12	11 7	6 0	
funct5	funct2	rs2	rs1	funct3	rd	opcode	**Type-R**
imm[11:0]			rs1	funct3	rd	opcode	**Type-I**
imm[11:5]		rs2	rs1	funct3	imm[4:0]	opcode	**Type-S**
rs3	funct2	rs2	rs1	funct3	rd	opcode	**Type-R4**

RV128A Standard Extension for Atomic Instructions (in addition to RV64A)

00010	aq	rl	00000	rs1	100	rd	0101111	LR.Q aqrl, rd, (rs1)
00011	aq	rl	rs2	rs1	100	rd	0101111	SC.Q aqrl, rd, rs2, (rs1)
00001	aq	rl	rs2	rs1	100	rd	0101111	AMOSWAP.Q aqrl, rd, rs2, (rs1)
00000	aq	rl	rs2	rs1	100	rd	0101111	AMOADD.Q aqrl, rd, rs2, (rs1)
00100	aq	rl	rs2	rs1	100	rd	0101111	AMOXOR.Q aqrl, rd, rs2, (rs1)
01000	aq	rl	rs2	rs1	100	rd	0101111	AMOOR.Q aqrl, rd, rs2, (rs1)
01100	aq	rl	rs2	rs1	100	rd	0101111	AMOAND.Q aqrl, rd, rs2, (rs1)
10000	aq	rl	rs2	rs1	100	rd	0101111	AMOMIN.Q aqrl, rd, rs2, (rs1)
10100	aq	rl	rs2	rs1	100	rd	0101111	AMOMAX.Q aqrl, rd, rs2, (rs1)
11000	aq	rl	rs2	rs1	100	rd	0101111	AMOMINU.Q aqrl, rd, rs2, (rs1)
11100	aq	rl	rs2	rs1	100	rd	0101111	AMOMAXU.Q aqrl, rd, rs2, (rs1)

RV32S Standard Extension for Supervisor-level Instructions

0000000		00000	00000	000	00000	1110011	ECALL
0000000		00001	00000	000	00000	1110011	EBREAK
0000000		00010	00000	000	00000	1110011	URET
0001000		00010	00000	000	00000	1110011	SRET
0010000		00010	00000	000	00000	1110011	HRET
0011000		00010	00000	000	00000	1110011	MRET
0111101		10010	00000	000	00000	1110011	DRET
00010	00	00100	rs1	000	00000	1110011	SFENCE.VM rs1
0001000		00101	00000	000	00000	1110011	WFI
csr[11:0]			rs1	001	rd	1110011	CSRRW rd, csr, rs1
csr[11:0]			rs1	010	rd	1110011	CSRRS rd, csr, rs1
csr[11:0]			rs1	011	rd	1110011	CSRRC rd, csr, rs1
csr[11:0]			uimm[4:0]	101	rd	1110011	CSRRWI rd, csr, zimm
csr[11:0]			uimm[4:0]	110	rd	1110011	CSRRSI rd, csr, zimm
csr[11:0]			uimm[4:0]	111	rd	1110011	CSRRCI rd, csr, zimm

RV32F Standard Extension for Single-Precision Floating-Point

simm[11:0]			rs1	010	frd	0000111	FLW frd, offset(rs1)
simm[11:5]		frs2	rs1	010	simm[4:0]	0100111	FSW frs2, offset(rs1)
frs3	00	frs2	frs1	rm	frd	1000011	FMADD.S rm, frd, frs1, frs2, frs3
frs3	00	frs2	frs1	rm	frd	1000111	FMSUB.S rm, frd, frs1, frs2, frs3
frs3	00	frs2	frs1	rm	frd	1001011	FNMSUB.S rm, frd, frs1, frs2, frs3
frs3	00	frs2	frs1	rm	frd	1001111	FNMADD.S rm, frd, frs1, frs2, frs3
00000	00	frs2	frs1	rm	frd	1010011	FADD.S rm, frd, frs1, frs2
00001	00	frs2	frs1	rm	frd	1010011	FSUB.S rm, frd, frs1, frs2
00010	00	frs2	frs1	rm	frd	1010011	FMUL.S rm, frd, frs1, frs2
00011	00	frs2	frs1	rm	frd	1010011	FDIV.S rm, frd, frs1, frs2
00100	00	frs2	frs1	000	frd	1010011	FSGNJ.S frd, frs1, frs2
00100	00	frs2	frs1	001	frd	1010011	FSGNJN.S frd, frs1, frs2

31	25	24 20	19 15	14 12	11 7	6 0	
funct5	funct2	rs2	rs1	funct3	rd	opcode	**Type-R**
imm[11:0]			rs1	funct3	rd	opcode	**Type-I**
imm[11:5]		rs2	rs1	funct3	imm[4:0]	opcode	**Type-S**
rs3	funct2	rs2	rs1	funct3	rd	opcode	**Type-R4**

RV32F Standard Extension for Single-Precision Floating-Point contd

00100	00	frs2	frs1	010	frd	1010011	FSGNJX.S frd, frs1, frs2
00101	00	frs2	frs1	000	frd	1010011	FMIN.S frd, frs1, frs2
00101	00	frs2	frs1	001	frd	1010011	FMAX.S frd, frs1, frs2
01011	00	00000	frs1	rm	frd	1010011	FSQRT.S rm, frd, frs1
10100	00	frs2	frs1	000	rd	1010011	FLE.S rd, frs1, frs2
10100	00	frs2	frs1	001	rd	1010011	FLT.S rd, frs1, frs2
10100	00	frs2	frs1	010	rd	1010011	FEQ.S rd, frs1, frs2
11000	00	00000	frs1	rm	rd	1010011	FCVT.W.S rm, rd, frs1
11000	00	00001	frs1	rm	rd	1010011	FCVT.WU.S rm, rd, frs1
11010	00	00000	rs1	rm	frd	1010011	FCVT.S.W rm, frd, rs1
11010	00	00001	rs1	rm	frd	1010011	FCVT.S.WU rm, frd, rs1
11100	00	00000	frs1	000	rd	1010011	FMV.X.S rd, frs1
11100	00	00000	frs1	001	rd	1010011	FCLASS.S rd, frs1
11110	00	00000	rs1	000	frd	1010011	FMV.S.X frd, rs1

RV64F Standard Extension for Single-Precision Floating-Point (in addition to RV32F)

11000	00	00010	frs1	rm	rd	1010011	FCVT.L.S rm, rd, frs1
11000	00	00011	frs1	rm	rd	1010011	FCVT.LU.S rm, rd, frs1
11010	00	00010	rs1	rm	frd	1010011	FCVT.S.L rm, frd, rs1
11010	00	00011	rs1	rm	frd	1010011	FCVT.S.LU rm, frd, rs1

RV32D Standard Extension for Double-Precision Floating-Point

simm[11:0]			rs1	011	frd	0000111	FLD frd, offset(rs1)
simm[11:5]		frs2	rs1	011	simm[4:0]	0100111	FSD frs2, offset(rs1)
frs3	01	frs2	frs1	rm	frd	1000011	FMADD.D rm, frd, frs1, frs2, frs3
frs3	01	frs2	frs1	rm	frd	1000111	FMSUB.D rm, frd, frs1, frs2, frs3
frs3	01	frs2	frs1	rm	frd	1001011	FNMSUB.D rm, frd, frs1, frs2, frs3
frs3	01	frs2	frs1	rm	frd	1001111	FNMADD.D rm, frd, frs1, frs2, frs3
00000	01	frs2	frs1	rm	frd	1010011	FADD.D rm, frd, frs1, frs2
00001	01	frs2	frs1	rm	frd	1010011	FSUB.D rm, frd, frs1, frs2
00010	01	frs2	frs1	rm	frd	1010011	FMUL.D rm, frd, frs1, frs2
00011	01	frs2	frs1	rm	frd	1010011	FDIV.D rm, frd, frs1, frs2
00100	01	frs2	frs1	000	frd	1010011	FSGNJ.D frd, frs1, frs2
00100	01	frs2	frs1	001	frd	1010011	FSGNJN.D frd, frs1, frs2
00100	01	frs2	frs1	010	frd	1010011	FSGNJX.D frd, frs1, frs2
00101	01	frs2	frs1	000	frd	1010011	FMIN.D frd, frs1, frs2
00101	01	frs2	frs1	001	frd	1010011	FMAX.D frd, frs1, frs2
01000	00	00001	frs1	rm	frd	1010011	FCVT.S.D rm, frd, frs1
01000	01	00000	frs1	rm	frd	1010011	FCVT.D.S rm, frd, frs1
01011	01	00000	frs1	rm	frd	1010011	FSQRT.D rm, frd, frs1
10100	01	frs2	frs1	000	rd	1010011	FLE.D rd, frs1, frs2
10100	01	frs2	frs1	001	rd	1010011	FLT.D rd, frs1, frs2
10100	01	frs2	frs1	010	rd	1010011	FEQ.D rd, frs1, frs2

31	25	24 20	19 15	14 12	11 7	6 0	
funct5	funct2	rs2	rs1	funct3	rd	opcode	**Type-R**
imm[11:0]			rs1	funct3	rd	opcode	**Type-I**
imm[11:5]		rs2	rs1	funct3	imm[4:0]	opcode	**Type-S**
rs3	funct2	rs2	rs1	funct3	rd	opcode	**Type-R4**

RV32D Standard Extension for Double-Precision Floating-Point contd

11000	01	00000	frs1	rm	rd	1010011	FCVT.W.D rm, rd, frs1
11000	01	00001	frs1	rm	rd	1010011	FCVT.WU.D rm, rd, frs1
11010	01	00000	rs1	rm	frd	1010011	FCVT.D.W rm, frd, rs1
11010	01	00001	rs1	rm	frd	1010011	FCVT.D.WU rm, frd, rs1
11100	01	00000	frs1	001	rd	1010011	FCLASS.D rd, frs1

RV64D Standard Extension for Double-Precision Floating-Point (in addition to RV32D)

11000	01	00010	frs1	rm	rd	1010011	FCVT.L.D rm, rd, frs1
11000	01	00011	frs1	rm	rd	1010011	FCVT.LU.D rm, rd, frs1
11100	01	00000	frs1	000	rd	1010011	FMV.X.D rd, frs1
11010	01	00010	rs1	rm	frd	1010011	FCVT.D.L rm, frd, rs1
11010	01	00011	rs1	rm	frd	1010011	FCVT.D.LU rm, frd, rs1
11110	01	00000	rs1	000	frd	1010011	FMV.D.X frd, rs1

RV32Q Standard Extension for Quadruple-Precision Floating-Point

simm[11:0]			rs1	100	frd	0000111	FLQ frd, offset(rs1)
simm[11:5]		frs2	rs1	100	simm[4:0]	0100111	FSQ frs2, offset(rs1)
frs3	11	frs2	frs1	rm	frd	1000011	FMADD.Q rm, frd, frs1, frs2, frs3
frs3	11	frs2	frs1	rm	frd	1000111	FMSUB.Q rm, frd, frs1, frs2, frs3
frs3	11	frs2	frs1	rm	frd	1001011	FNMSUB.Q rm, frd, frs1, frs2, frs3
frs3	11	frs2	frs1	rm	frd	1001111	FNMADD.Q rm, frd, frs1, frs2, frs3
00000	11	frs2	frs1	rm	frd	1010011	FADD.Q rm, frd, frs1, frs2
00001	11	frs2	frs1	rm	frd	1010011	FSUB.Q rm, frd, frs1, frs2
00010	11	frs2	frs1	rm	frd	1010011	FMUL.Q rm, frd, frs1, frs2
00011	11	frs2	frs1	rm	frd	1010011	FDIV.Q rm, frd, frs1, frs2
00100	11	frs2	frs1	000	frd	1010011	FSGNJ.Q frd, frs1, frs2
00100	11	frs2	frs1	001	frd	1010011	FSGNJN.Q frd, frs1, frs2
00100	11	frs2	frs1	010	frd	1010011	FSGNJX.Q frd, frs1, frs2
00101	11	frs2	frs1	000	frd	1010011	FMIN.Q frd, frs1, frs2
00101	11	frs2	frs1	001	frd	1010011	FMAX.Q frd, frs1, frs2
01000	00	00011	frs1	rm	frd	1010011	FCVT.S.Q rm, frd, frs1
01000	11	00000	frs1	rm	frd	1010011	FCVT.Q.S rm, frd, frs1
01000	01	00011	frs1	rm	frd	1010011	FCVT.D.Q rm, frd, frs1
01000	11	00001	frs1	rm	frd	1010011	FCVT.Q.D rm, frd, frs1
01011	11	00000	frs1	rm	frd	1010011	FSQRT.Q rm, frd, frs1
10100	11	frs2	frs1	000	rd	1010011	FLE.Q rd, frs1, frs2
10100	11	frs2	frs1	001	rd	1010011	FLT.Q rd, frs1, frs2
10100	11	frs2	frs1	010	rd	1010011	FEQ.Q rd, frs1, frs2
11000	11	00000	frs1	rm	rd	1010011	FCVT.W.Q rm, rd, frs1
11000	11	00001	frs1	rm	rd	1010011	FCVT.WU.Q rm, rd, frs1
11010	11	00000	rs1	rm	frd	1010011	FCVT.Q.W rm, frd, rs1
11010	11	00001	rs1	rm	frd	1010011	FCVT.Q.WU rm, frd, rs1
11100	11	00000	frs1	001	rd	1010011	FCLASS.Q rd, frs1

31	25	24 20	19 15	14 12	11 7	6 0	
funct5	funct2	rs2	rs1	funct3	rd	opcode	**Type-R**

RV64Q Standard Extension for Quadruple-Precision Floating-Point (in addition to RV32Q)

11000	11	00010	frs1	rm	rd	1010011	FCVT.L.Q rm, rd, frs1
11000	11	00011	frs1	rm	rd	1010011	FCVT.LU.Q rm, rd, frs1
11010	11	00010	rs1	rm	frd	1010011	FCVT.Q.L rm, frd, rs1
11010	11	00011	rs1	rm	frd	1010011	FCVT.Q.LU rm, frd, rs1
11100	11	00000	frs1	000	rd	1010011	FMV.X.Q rd, frs1
11110	11	00000	rs1	000	frd	1010011	FMV.Q.X frd, rs1

15 13	12	10	9 7	6 5	4 2	1 0	
funct3	imm8				rd’	op	**Type-CIW**
funct3	imm3		rs1’	imm2	rd’	op	**Type-CL**
funct3	imm3		rs1’	imm2	rs2’	op	**Type-CS**
funct3	imm1	rd/rs1		imm5		op	**Type-CI**
funct3	imm11					op	**Type-CJ**
funct3	imm3		rs1’	imm5		op	**Type-CB**
funct4		rd/rs1		rs2		op	**Type-CR**
funct3	imm6			rs2		op	**Type-CSS**

RV32C Standard Extension for Compressed Instructions

000	nzuimm[5:4|9:6|2|3]				rd’	00	C.ADDI4SPN rd, rs1, imm
001	uimm[5:3]		rs1’	uimm[7:6]	frd’	00	C.FLD frd, offset(rs1)
010	uimm[5:3]		rs1’	uimm[2|6]	rd’	00	C.LW rd, offset(rs1)
011	uimm[5:3]		rs1’	uimm[2|6]	frd’	00	C.FLW frd, offset(rs1)
101	uimm[5:3]		rs1’	uimm[7:6]	frs2’	00	C.FSD frs2, offset(rs1)
110	uimm[5:3]		rs1’	uimm[2|6]	rs2’	00	C.SW rs2, offset(rs1)
111	uimm[5:3]		rs1’	uimm[2|6]	frs2’	00	C.FSW frs2, offset(rs1)
000	0	00000		00000		01	C.NOP
000	nzsimm[5]	rs1/rd≠ 0		nzsimm[4:0]		01	C.ADDI rd, rs1, imm
001	simm[11|4|9:8|10|6|7|3:1|5]					01	C.JAL rd, offset
010	simm[5]	rs1/rd≠ 0		simm[4:0]		01	C.LI rd, rs1, imm
011	nzsimm[9]	rs1/rd= 2		nzsimm[4|6|8:7|5]		01	C.ADDI16SP rd, rs1, imm
011	nzsimm[17]	rd≠ {0, 2}		nzsimm[16:12]		01	C.LUI rd, imm
100	0	00	rs1’/rd’	nzuimm[4:0]		01	C.SRLI rd, rs1, imm
100	0	01	rs1’/rd’	nzuimm[4:0]		01	C.SRAI rd, rs1, imm
100	nzsimm[5]	10	rs1’/rd’	nzsimm[4:0]		01	C.ANDI rd, rs1, imm
100	011		rs1’/rd’	00	rs2’	01	C.SUB rd, rs1, rs2
100	011		rs1’/rd’	01	rs2’	01	C.XOR rd, rs1, rs2
100	011		rs1’/rd’	10	rs2’	01	C.OR rd, rs1, rs2
100	011		rs1’/rd’	11	rs2’	01	C.AND rd, rs1, rs2
100	111		rs1’/rd’	00	rs2’	01	C.SUBW rd, rs1, rs2
100	111		rs1’/rd’	01	rs2’	01	C.ADDW rd, rs1, rs2
101	simm[11|4|9:8|10|6|7|3:1|5]					01	C.J rd, offset
110	simm[8|4:3]		rs1’	simm[7:6|2:1|5]		01	C.BEQZ rs1, rs2, offset
111	simm[8|4:3]		rs1’	simm[7:6|2:1|5]		01	C.BNEZ rs1, rs2, offset
000	0	rs1/rd≠ 0		nzuimm[4:0]		10	C.SLLI rd, rs1, imm
001	uimm[5]	frd		uimm[4:3|8:6]		10	C.FLDSP frd, offset(rs1)
010	uimm[5]	rd≠ 0		uimm[4:2|7:6]		10	C.LWSP rd, offset(rs1)
011	uimm[5]	frd		uimm[4:2|7:6]		10	C.FLWSP frd, offset(rs1)
100	rd”	rs1		00000		10	C.JR rd, rs1, offset
1000		rd≠ 0		rs2≠ 0		10	C.MV rd, rs1, rs2
100	1	00000		00000		10	C.EBREAK
100	rd”	rs1		00000		10	C.JALR rd, rs1, offset
1001		rs1/rd≠ 0		rs2≠ 0		10	C.ADD rd, rs1, rs2
101	uimm[5:3|8:6]			frs2		10	C.FSDSP frs2, offset(rs1)
110	uimm[5:2|7:6]			rs2		10	C.SWSP rs2, offset(rs1)
111	uimm[5:2|7:6]			frs2		10	C.FSWSP frs2, offset(rs1)

15 13	12	10	9 7	6 5	4 2	1 0	
funct3	imm3		rs1’	imm2	rd’	op	**Type-CL**
funct3	imm3		rs1’	imm2	rs2’	op	**Type-CS**
funct3	imm1	rd/rs1		imm5		op	**Type-CI**
funct3	imm3		rs1’	imm5		op	**Type-CB**
funct3	imm6			rs2		op	**Type-CSS**

RV64C Standard Extension for Compressed Instructions (in addition to RV32C)

011	uimm[5:3]		rs1’	uimm[7:6]	rd’	00	C.LD rd, offset(rs1)
111	uimm[5:3]		rs1’	uimm[7:6]	rs2’	00	C.SD rs2, offset(rs1)
001	simm[5]	rs1/rd≠ 0		simm[4:0]		01	C.ADDIW rd, rs1, imm
100	nzuimm[5]	00	rs1’/rd’	nzuimm[4:0]		01	C.SRLI rd, rs1, imm
100	nzuimm[5]	01	rs1’/rd’	nzuimm[4:0]		01	C.SRAI rd, rs1, imm
000	nzuimm[5]	rs1/rd≠ 0		nzuimm[4:0]		10	C.SLLI rd, rs1, imm
011	uimm[5]	rd≠ 0		uimm[4:3|8:6]		10	C.LDSP rd, offset(rs1)
111	uimm[5:3|8:6]			rs2		10	C.SDSP rs2, offset(rs1)

RV128C Standard Extension for Compressed Instructions (in addition to RV64C)

001	uimm[5:4|8]		rs1’	uimm[7:6]	rd’	00	C.LQ rd, offset(rs1)
101	uimm[5:4|8]		rs1’	uimm[7:6]	rs2’	00	C.SQ rs2, offset(rs1)
001	uimm[5]	rd		uimm[4|9:6]		10	C.LQSP rd, offset(rs1)
101	uimm[5:4|9:6]			rs2		10	C.SQSP rs2, offset(rs1)