fadd.d rd, rs1, rs2 f [rd] = f [rs1] + f [rs2]

双精度浮点加*(Floating-point Add, Double-Precision)*. R-type, RV32D and RV64D.

把寄存器f[*rs1*]和f[*rs2*]中的双精度浮点数相加，并将舍入后的和写入f[*rd*]。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0000001 | rs2 | rs1 | rm | rd | 1010011 |

fadd.s rd, rs1, rs2 f [rd] = f [rs1] + f [rs2]

单精度浮点加*(Floating-point Add, Single-Precision)*. R-type, RV32F and RV64F.

把寄存器f[*rs1*]和f[*rs2*]中的单精度浮点数相加，并将舍入后的和写入f[*rd*]。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0000000 | rs2 | rs1 | rm | rd | 1010011 |

fclass.d rd, rs1, rs2 x[rd] = classify*d*(f[rs1])

双精度浮点分类*(Floating-point Classify, Double-Precision)*. R-type, RV32D and RV64D.

把一个表示寄存器f[*rs1*]中双精度浮点数类别的掩码写入x[*rd*]中。关于如何解释写入x[*rd*]的值，请参阅指令**fclass.s**的介绍。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1110001 | 00000 | rs1 | 001 | rd | 1010011 |

fclass.s rd, rs1, rs2 x[rd] = classify*s*(f[rs1])

单精度浮点分类*(Floating-point Classify, Single-Precision)*. R-type, RV32F and RV64F.

把一个表示寄存器f[*rs1*]中单精度浮点数类别的掩码写入x[*rd*]中。x[*rd*]中有且仅有一位被置上，见下表。

|  |  |
| --- | --- |
| x*[rd]*位 | 含义 |
| 0 | f [*rs1*]为。 |
| 1 | f [*rs1*]是负规格化数。 |
| 2 | f [*rs1*]是负的非规格化数。 |
| 3 | f [*rs1*]是-0。 |
| 4 | f [*rs1*]是+0。 |
| 5 | f [*rs1*]是正的非规格化数。 |
| 6 | f [*rs1*]是正的规格化数。 |
| 7 | f [*rs1*]为+。 |
| 8 | f [*rs1*]是信号(signaling)NaN。 |
| 9 | f [*rs1*]是一个安静(quiet)NaN。 |

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1110000 | 00000 | rs1 | 001 | rd | 1010011 |

fcvt.d.l rd, rs1, rs2 f[rd] = f64*s*64(x[rs1])

长整型向双精度浮点转换*(Floating-point Convert to Double from Long)*. R-type, RV64D.

把寄存器x[*rs1*]中的64位二进制补码表示的整数转化为双精度浮点数，再写入f[*rd*]中。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1101001 | 00010 | rs1 | rm | rd | 1010011 |

fcvt.d.lu rd, rs1, rs2 f[rd] = f64*u*64(x[rs1])

无符号长整型向双精度浮点转换*(Floating-point Convert to Double from Unsigned Long)*. R-type, RV64D.

把寄存器x[*rs1*]中的64位无符号整数转化为双精度浮点数，再写入f[*rd*]中。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1101001 | 00011 | rs1 | rm | rd | 1010011 |

fcvt.d.s rd, rs1, rs2 f[rd] = f64*f*32(f[rs1])

单精度向双精度浮点转换*(Floating-point Convert to Double from Single)*. R-type, RV32D and RV64D.

把寄存器f[*rs1*]中的单精度浮点数转化为双精度浮点数，再写入f[*rd*]中。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0100001 | 00000 | rs1 | rm | rd | 1010011 |

fcvt.d.w rd, rs1, rs2 f[rd] = f64*s*32(x[rs1])

字向双精度浮点转换*(Floating-point Convert to Double from Word)*. R-type, RV32D and RV64D.

把寄存器x[*rs1*]中的32位二进制补码表示的整数转化为双精度浮点数，再写入f[*rd*]中。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1101001 | 00000 | rs1 | rm | rd | 1010011 |

fcvt.d.wu rd, rs1, rs2 f[rd] = f64*u*32(x[rs1])

无符号字向双精度浮点转换*(Floating-point Convert to Double from Unsigned Word)*. R-type, RV32D and RV64D.

把寄存器x[*rs1*]中的32位无符号整数转化为双精度浮点数，再写入f[*rd*]中。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1101001 | 00001 | rs1 | rm | rd | 1010011 |

fcvt.l.d rd, rs1, rs2 x[rd] = s64*f*64(f[rs1])

双精度浮点向长整型转换*(Floating-point Convert to Long from Double)*. R-type, RV64D.

把寄存器f[*rs1*]中的双精度浮点数转化为64位二进制补码表示的整数，再写入x[*rd*]中。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1100001 | 00010 | rs1 | rm | rd | 1010011 |

fcvt.l.s rd, rs1, rs2 x[rd] = s64*f*32(f[rs1])

单精度浮点向长整型转换*(Floating-point Convert to Long from Single)*. R-type, RV64F.

把寄存器f[*rs1*]中的单精度浮点数转化为64位二进制补码表示的整数，再写入x[*rd*]中。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1100000 | 00010 | rs1 | rm | rd | 1010011 |

fcvt.lu.d rd, rs1, rs2 x[rd] = u64*f*64(f[rs1])

双精度浮点向无符号长整型转换*(Floating-point Convert to Unsigned Long from Double)*. R-type, RV64D.

把寄存器f[*rs1*]中的双精度浮点数转化为64位无符号整数，再写入x[*rd*]中。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1100001 | 00011 | rs1 | rm | rd | 1010011 |

fcvt.lu.s rd, rs1, rs2 x[rd] = u64*f*32(f[rs1])

单精度浮点向无符号长整型转换*(Floating-point Convert to Unsigned Long from Single)*. R-type, RV64F.

把寄存器f[*rs1*]中的单精度浮点数转化为64位二进制补码表示的整数，再写入x[*rd*]中。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1100000 | 00011 | rs1 | rm | rd | 1010011 |

fcvt.s.d rd, rs1, rs2 f[rd] = f32*f*64(f[rs1])

双精度向单精度浮点转换*(Floating-point Convert to Single from Double)*. R-type, RV32D and RV64D.

把寄存器f[*rs1*]中的双精度浮点数转化为单精度浮点数，再写入f[*rd*]中。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0100000 | 00001 | rs1 | rm | rd | 1010011 |

fcvt.s.l rd, rs1, rs2 f[rd] = f32*s*64(x[rs1])

长整型向单精度浮点转换*(Floating-point Convert to Single from Long)*. R-type, RV64F.

把寄存器x[*rs1*]中的64位二进制补码表示的整数转化为单精度浮点数，再写入f[*rd*]中。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1101000 | 00010 | rs1 | rm | rd | 1010011 |

fcvt.s.lu rd, rs1, rs2 f[rd] = f32*u*64(x[rs1])

无符号长整型向单精度浮点转换*(Floating-point Convert to Single from Unsigned Long)*. R-type, RV64F.

把寄存器x[*rs1*]中的64位的无符号整数转化为单精度浮点数，再写入f[*rd*]中。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1101000 | 00011 | rs1 | rm | rd | 1010011 |

fcvt.s.w rd, rs1, rs2 f[rd] = f32*s*32(x[rs1])

字向单精度浮点转换*(Floating-point Convert to Single from Word)*. R-type, RV32F and RV64F.

把寄存器x[*rs1*]中的32位二进制补码表示的整数转化为单精度浮点数，再写入f[*rd*]中。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1101000 | 00000 | rs1 | rm | rd | 1010011 |

fcvt.s.wu rd, rs1, rs2 f[rd] = f32*u*32(x[rs1])

无符号字向单精度浮点转换*(Floating-point Convert to Single from Unsigned Word)*. R-type, RV32F and RV64F.

把寄存器x[*rs1*]中的32位无符号整数转化为单精度浮点数，再写入f[*rd*]中。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1101000 | 00001 | rs1 | rm | rd | 1010011 |

fcvt.w.d rd, rs1, rs2 x[rd] = sext(s32*f*64(f[rs1]))

双精度浮点向字转换*(Floating-point Convert to Word from Double)*. R-type, RV32D and RV64D.

把寄存器f[*rs1*]中的双精度浮点数转化为32位二进制补码表示的整数，再写入x[*rd*]中。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1100001 | 00000 | rs1 | rm | rd | 1010011 |

fcvt.wu.d rd, rs1, rs2 x[rd] = sext(u32*f*64(f[rs1]))

双精度浮点向无符号字转换*(Floating-point Convert to Unsigned Word from Double)*. R-type, RV32D and RV64D.

把寄存器f[*rs1*]中的双精度浮点数转化为32位无符号整数，再写入x[*rd*]中。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1100001 | 00001 | rs1 | rm | rd | 1010011 |

fcvt.w.s rd, rs1, rs2 x[rd] = sext(s32*f*32(f[rs1]))

单精度浮点向字转换*(Floating-point Convert to Word from Single)*. R-type, RV32F and RV64F.

把寄存器f[*rs1*]中的单精度浮点数转化为32位二进制补码表示的整数，再写入x[*rd*]中。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1100000 | 00000 | rs1 | rm | rd | 1010011 |

fcvt.wu.s rd, rs1, rs2 x[rd] = sext(u32*f*32(f[rs1]))

单精度浮点向无符号字转换*(Floating-point Convert to Unsigned Word from Single)*. R-type, RV32F and RV64F.

把寄存器f[*rs1*]中的单精度浮点数转化为32位无符号整数，再写入x[*rd*]中。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1100000 | 00001 | rs1 | rm | rd | 1010011 |

fdiv.d rd, rs1, rs2 f[rd] = f[rs1] ÷ f[rs2]

双精度浮点除法*(Floating-point Divide, Double-Precision)*. R-type, RV32D and RV64D.

把寄存器f[*rs1*]和f[*rs2*]中的双精度浮点数相除，并将舍入后的商写入f[*rd*]。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0001101 | rs2 | rs1 | rm | rd | 1010011 |

fdiv.s rd, rs1, rs2 f[rd] = f[rs1] ÷ f[rs2]

单精度浮点除法*(Floating-point Divide, Single-Precision)*. R-type, RV32F and RV64F.

把寄存器f[*rs1*]和f[*rs2*]中的单精度浮点数相除，并将舍入后的商写入f[*rd*]。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0001100 | rs2 | rs1 | rm | rd | 1010011 |

fence pred, succ Fence(pred, succ)

同步内存和I/O*(Fence Memory and I/O)*. I-type, RV32I and RV64I.

在后续指令中的内存和I/O访问对外部（例如其他线程）可见之前，使这条指令之前的内存及I/O访问对外部可见。比特中的第3,2,1和0位分别对应于设备输入，设备输出，内存读写。例如**fence** r, rw，将前面读取与后面的读取和写入排序，使用*pred* = 0010和*succ* = 0011进行编码。如果省略了参数，则表示**fence** iorw, iorw，即对所有访存请求进行排序。

31 28 27 24 23 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0000 | pred | succ | 00000 | 000 | 00000 | 0001111 |

fence.i Fence(Store, Fetch)

同步指令流*(Fence Instruction Stream)*. I-type, RV32I and RV64I.

使对内存指令区域的读写，对后续取指令可见。

31 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 000000000000 | 00000 | 001 | 00000 | 0001111 |

feq.d rd, rs1, rs2 x[rd] = f[rs1] == f[rs2]

双精度浮点相等*(Floating-point Equals, Double-Precision)*. R-type, RV32D and RV64D.

若寄存器f[*rs1*]和f[*rs2*]中的双精度浮点数相等，则在x[*rd*]中写入1，反之写0。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1010001 | rs2 | rs1 | 010 | rd | 1010011 |

feq.s rd, rs1, rs2 x[rd] = f[rs1] == f[rs2]

单精度浮点相等*(Floating-point Equals, Single-Precision)*. R-type, RV32F and RV64F.

若寄存器f[*rs1*]和f[*rs2*]中的单精度浮点数相等，则在x[*rd*]中写入1，反之写0。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1010000 | rs2 | rs1 | 010 | rd | 1010011 |

fld rd, offset(rs1) f[rd] = M[x[rs1] + sext(offset)][63:0]

浮点加载双字*(Floating-point Load Doubleword)*. I-type, RV32D and RV64D.

从内存地址x[*rs1*] + *sign-extend*(*offset*)中取双精度浮点数，并写入f[*rd*]。

压缩形式：**c.fldsp** rd, offset; **c.fld** rd, offset(rs1)

31 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| offset[11:0] | rs1 | 011 | rd | 0000111 |

fle.d rd, rs1, rs2 x[rd] = f[rs1] ≤ f[rs2]

双精度浮点小于等于*(Floating-point Less Than or Equal, Double-Precision)*. R-type, RV32D and RV64D.

若寄存器f[*rs1*]中的双精度浮点数小于等于f[*rs2*]中的双精度浮点数，则在x[*rd*]中写入1，反之写0。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1010001 | rs2 | rs1 | 000 | rd | 1010011 |

fle.s rd, rs1, rs2 x[rd] = f[rs1] ≤ f[rs2]

单精度浮点小于等于*(Floating-point Less Than or Equal, Single-Precision)*. R-type, RV32F and RV64F.

若寄存器f[*rs1*]中的单精度浮点数小于等于f[*rs2*]中的单精度浮点数，则在x[*rd*]中写入1，反之写0。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1010000 | rs2 | rs1 | 000 | rd | 1010011 |

fle.d rd, rs1, rs2 x[rd] = f[rs1] < f[rs2]

双精度浮点小于 *(Floating-point Less Than, Double-Precision)*. R-type, RV32D and RV64D.

若寄存器f[*rs1*]中的双精度浮点数小于f[*rs2*]中的双精度浮点数，则在x[*rd*]中写入1，反之写0。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1010001 | rs2 | rs1 | 001 | rd | 1010011 |

fle.s rd, rs1, rs2 x[rd] = f[rs1] < f[rs2]

单精度浮点小于 *(Floating-point Less Than, Single-Precision)*. R-type, RV32F and RV64F.

若寄存器f[*rs1*]中的单精度浮点数小于f[*rs2*]中的单精度浮点数，则在x[*rd*]中写入1，反之写0。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1010000 | rs2 | rs1 | 001 | rd | 1010011 |

flw rd, offset(rs1) f[rd] = M[x[rs1] + sext(offset)][31:0]

浮点加载字*(Floating-point Load Word)*. I-type, RV32F and RV64F.

从内存地址x[*rs1*] + *sign-extend*(*offset*)中取单精度浮点数，并写入f[*rd*]。

压缩形式：**c.flwsp** rd, offset; **c.flw** rd, offset(rs1)

31 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| offset[11:0] | rs1 | 010 | rd | 0000111 |

fmadd.d rd, rs1, rs2, rs3 f[rd] = f[rs1]×f[rs2]+f[rs3]

双精度浮点乘加*(Floating-point Fused Multiply-Add, Double-Precision)*. R4-type, RV32D and RV64D.

把寄存器f[*rs1*]和f[*rs2*]中的双精度浮点数相乘，并将未舍入的积和寄存器f[*rs3*]中的双精度浮点数相加，将舍入后的双精度浮点数写入f[*rd*]。

31 27 26 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| rs3 | 01 | rs2 | rs1 | rm | rd | 1000011 |

fmadd.s rd, rs1, rs2, rs3 f[rd] = f[rs1]×f[rs2]+f[rs3]

单精度浮点乘加*(Floating-point Fused Multiply-Add, Single-Precision)*. R4-type, RV32F and RV64F.

把寄存器f[*rs1*]和f[*rs2*]中的单精度浮点数相乘，并将未舍入的积和寄存器f[*rs3*]中的单精度浮点数相加，将舍入后的单精度浮点数写入f[*rd*]。

31 27 26 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| rs3 | 00 | rs2 | rs1 | rm | rd | 1000011 |

fmax.d rd, rs1, rs2 f[rd] = max(f[rs1], f[rs2])

双精度浮点最大值*(Floating-point Maximum, Double-Precision)*. R-type, RV32D and RV64D.

把寄存器f[*rs1*]和f[*rs2*]中的双精度浮点数中的较大值写入f[*rd*]中。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0010101 | rs2 | rs1 | 001 | rd | 1010011 |

fmax.s rd, rs1, rs2 f[rd] = max(f[rs1], f[rs2])

单精度浮点最大值*(Floating-point Maximum, Single-Precision)*. R-type, RV32F and RV64F.

把寄存器f[*rs1*]和f[*rs2*]中的单精度浮点数中的较大值写入f[*rd*]中。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0010100 | rs2 | rs1 | 001 | rd | 1010011 |

fmin.d rd, rs1, rs2 f[rd] = min(f[rs1], f[rs2])

双精度浮点最小值*(Floating-point Minimum, Double-Precision)*. R-type, RV32D and RV64D.

把寄存器f[*rs1*]和f[*rs2*]中的双精度浮点数中的较小值写入f[*rd*]中。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0010101 | rs2 | rs1 | 000 | rd | 1010011 |

fmin.s rd, rs1, rs2 f[rd] = min(f[rs1], f[rs2])

单精度浮点最小值*(Floating-point Minimum, Single-Precision)*. R-type, RV32F and RV64F.

把寄存器f[*rs1*]和f[*rs2*]中的单精度浮点数中的较小值写入f[*rd*]中。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0010100 | rs2 | rs1 | 000 | rd | 1010011 |

fmsub.d rd, rs1, rs2, rs3 f[rd] = f[rs1]×f[rs2]-f[rs3]

双精度浮点乘减*(Floating-point Fused Multiply-Subtract, Double-Precision)*. R4-type, RV32D and RV64D.

把寄存器f[*rs1*]和f[*rs2*]中的双精度浮点数相乘，并将未舍入的积减去寄存器f[*rs3*]中的双精度浮点数，将舍入后的双精度浮点数写入f[*rd*]。

31 27 26 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| rs3 | 01 | rs2 | rs1 | rm | rd | 1000111 |

fmsub.s rd, rs1, rs2, rs3 f[rd] = f[rs1]×f[rs2]-f[rs3]

单精度浮点乘减*(Floating-point Fused Multiply-Subtarct, Single-Precision)*. R4-type, RV32F and RV64F.

把寄存器f[*rs1*]和f[*rs2*]中的单精度浮点数相乘，并将未舍入的积减去寄存器f[*rs3*]中的单精度浮点数，将舍入后的单精度浮点数写入f[*rd*]。

31 27 26 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| rs3 | 00 | rs2 | rs1 | rm | rd | 1000111 |

fmul.d rd, rs1, rs2 f[rd] = f[rs1] × f[rs2]

双精度浮点乘*(Floating-point Multiply, Double-Precision)*. R-type, RV32D and RV64D.

把寄存器f[*rs1*]和f[*rs2*]中的双精度浮点数相乘，将舍入后的双精度结果写入f[*rd*]中。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0001001 | rs2 | rs1 | rm | rd | 1010011 |

fmul.s rd, rs1, rs2 f[rd] = f[rs1] × f[rs2]

单精度浮点乘*(Floating-point Multiply, Single-Precision)*. R-type, RV32F and RV64F.

把寄存器f[*rs1*]和f[*rs2*]中的单精度浮点数相乘，将舍入后的单精度结果写入f[*rd*]中。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0001000 | rs2 | rs1 | rm | rd | 1010011 |

fmv.d rd, rs1 f[rd] = f[rs1]

双精度浮点移动 *(Floating-point Move)*. 伪指令(Pesudoinstruction), RV32D and RV64D.

把寄存器f[*rs1*]中的双精度浮点数复制到f[*rd*]中，等同于**fsgnj.d** rd, rs1, rs1.

fmv.d.x rd, rs1, rs2 f[rd] = x[rs1][63:0]

双精度浮点移动*(Floating-point Move Doubleword from Integer)*. R-type, RV64D.

把寄存器x[*rs1*]中的双精度浮点数复制到f[*rd*]中。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1111001 | 00000 | rs1 | 000 | rd | 1010011 |

fmv.s rd, rs1 f[rd] = f[rs1]

单精度浮点移动 *(Floating-point Move)*. 伪指令(Pesudoinstruction), RV32F and RV64F.

把寄存器f[*rs1*]中的单精度浮点数复制到f[*rd*]中，等同于**fsgnj.s** rd, rs1, rs1.

fmv.d.x rd, rs1, rs2 f[rd] = x[rs1][31:0]

单精度浮点移动*(Floating-point Move Word from Integer)*. R-type, RV32F and RV64F.

把寄存器x[*rs1*]中的单精度浮点数复制到f[*rd*]中。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1111000 | 00000 | rs1 | 000 | rd | 1010011 |

fmv.x.d rd, rs1, rs2 x[rd] = f[rs1][63:0]

双精度浮点移动*(Floating-point Move Doubleword to Integer)*. R-type, RV64D.

把寄存器f[*rs1*]中的双精度浮点数复制到x[*rd*]中。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1110001 | 00000 | rs1 | 000 | rd | 1010011 |

fmv.x.w rd, rs1, rs2 x[rd] = sext(f[rs1][31:0])

单精度浮点移动*(Floating-point Move Word to Integer)*. R-type, RV32F and RV64F.

把寄存器f[*rs1*]中的单精度浮点数复制到x[*rd*]中，对于RV64F，将结果进行符号扩展。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1110000 | 00000 | rs1 | 000 | rd | 1010011 |

fneg.d rd, rs1 f[rd] = -f[rs1]

双精度浮点取反 *(Floating-point Negate)*. 伪指令(Pesudoinstruction), RV32D and RV64D.

把寄存器f[*rs1*]中的双精度浮点数取反后写入f[*rd*]中，等同于**fsgnjn.d** rd, rs1, rs1.

fneg.s rd, rs1 f[rd] = -f[rs1]

单精度浮点取反 *(Floating-point Negate)*. 伪指令(Pesudoinstruction), RV32F and RV64F.

把寄存器f[*rs1*]中的单精度浮点数取反后写入f[*rd*]中，等同于**fsgnjn.s** rd, rs1, rs1.

fnmadd.d rd, rs1, rs2, rs3 f[rd] = f[rs1]×f[rs2]+f[rs3]

双精度浮点乘取反加*(Floating-point Fused Negative Multiply-Add, Double-Precision)*. R4-type, RV32D and RV64D.

把寄存器f[*rs1*]和f[*rs2*]中的双精度浮点数相乘，将结果取反，并将未舍入的积和寄存器f[*rs3*]中的双精度浮点数相加，将舍入后的双精度浮点数写入f[*rd*]。

31 27 26 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| rs3 | 01 | rs2 | rs1 | rm | rd | 1001111 |

fnmadd.s rd, rs1, rs2, rs3 f[rd] = -f[rs1]\_f[rs2]-f[rs3]

单精度浮点乘取反加*(Floating-point Fused Negative Multiply-Add, Single-Precision)*. R4-type, RV32F and RV64F.

把寄存器f[*rs1*]和f[*rs2*]中的单精度浮点数相乘，将结果取反，并将未舍入的积和寄存器f[*rs3*]中的单精度浮点数相加，将舍入后的单精度浮点数写入f[*rd*]。

31 27 26 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| rs3 | 00 | rs2 | rs1 | rm | rd | 1001111 |

fnmsub.d rd, rs1, rs2, rs3 f[rd] = -f[rs1]\_f[rs2]+f[rs3]

双精度浮点乘取反减*(Floating-point Fused Negative Multiply-Subtract, Double-Precision)*. R4-type, RV32D and RV64D.

把寄存器f[*rs1*]和f[*rs2*]中的双精度浮点数相乘，将结果取反，并将未舍入的积减去寄存器f[*rs3*]中的双精度浮点数，将舍入后的双精度浮点数写入f[*rd*]。

31 27 26 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| rs3 | 01 | rs2 | rs1 | rm | rd | 1001011 |

fnmsub.s rd, rs1, rs2, rs3 f[rd] = -f[rs1]×f[rs2]+f[rs3]

单精度浮点乘取反减*(Floating-point Fused Negative Multiply-Subtract, Single-Precision)*. R4-type, RV32F and RV64F.

把寄存器f[*rs1*]和f[*rs2*]中的单精度浮点数相乘，将结果取反，并将未舍入的积减去寄存器f[*rs3*]中的单精度浮点数，将舍入后的单精度浮点数写入f[*rd*]。

31 27 26 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| rs3 | 00 | rs2 | rs1 | rm | rd | 1001011 |

frcsr rd x[rd] = CSRs[fcsr]

浮点读控制状态寄存器 *(Floating-point Read Control and Status Register)*. 伪指令(Pseudoinstruction), RV32F and RV64F.

把浮点控制状态寄存器的值写入x[*rd*]，等同于**csrrs** rd, fcsr, x0.

frflags rd x[rd] = CSRs[fflags]

浮点读异常标志 *(Floating-point Read Exception Flags)*. 伪指令(Pseudoinstruction), RV32F and RV64F.

把浮点异常标志的值写入x[*rd*]，等同于**csrrs** rd, fflags, x0.

frrm rd x[rd] = CSRs[frm]

浮点读舍入模式 *(Floating-point Read Rounding Mode)*. 伪指令(Pseudoinstruction), RV32F and RV64F.

把浮点舍入模式的值写入x[*rd*]，等同于**csrrs** rd, frm, x0.

fscsr rd, rs1 t = CSRs[fcsr]; CSRs[fcsr] = x[rs1]; x[rd] = t

浮点换出控制状态寄存器 *(Floating-point Swap Control and Status Register)*. 伪指令(Pseudoinstruction), RV32F and RV64F.

把寄存器x[*rs1*]的值写入浮点控制状态寄存器，并将浮点控制状态寄存器的原值写入x[*rd*]，等同于**csrrw** rd, fcsr, rs1。*rd*默认为x0。

fsd rs2, offset(rs1) M[x[rs1] + sext(offset)] = f[rs2][63:0]

双精度浮点存储*(Floating-point Store Doubleword)*. S-type, RV32D and RV64D.

将寄存器f[*rs2*]中的双精度浮点数存入内存地址x[*rs1*] + *sign-extend*(*offset*)中。

31 25 24 20 19 15 14 12 11 7 6 0

压缩形式：**c.fsdsp** rs2, offset; **c.fsd** rs2, offset(rs1)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| offset[11:5] | rs2 | rs1 | 011 | offset[4:0] | 0100111 |

fsflags rd, rs1 t = CSRs[fflags]; CSRs[fflags] = x[rs1]; x[rd] = t

浮点换出异常标志 *(Floating-point Swap Exception Flags)*. 伪指令(Pseudoinstruction), RV32F and RV64F.

把寄存器x[*rs1*]的值写入浮点异常标志寄存器，并将浮点异常标志寄存器的原值写入x[*rd*]，等同于**csrrw** rd, fflags, rs1。*rd*默认为x0。

fsgnj.d rd, rs1, rs2 f[rd] = {f[rs2][63], f[rs1][62:0]}

双精度浮点符号注入*(Floating-point Sign Inject, Double-Precision)*. R-type, RV32D and RV64D.

用f[*rs1*]指数和有效数以及f[*rs2*]的符号的符号位，来构造一个新的双精度浮点数，并将其写入f[*rd*]。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0010001 | rs2 | rs1 | 000 | rd | 1010011 |

fsgnj.s rd, rs1, rs2 f[rd] = {f[rs2][31], f[rs1][30:0]}

单精度浮点符号注入*(Floating-point Sign Inject, Single-Precision)*. R-type, RV32F and RV64F.

用f[*rs1*]指数和有效数以及f[*rs2*]的符号的符号位，来构造一个新的单精度浮点数，并将其写入f[*rd*]。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0010000 | rs2 | rs1 | 000 | rd | 1010011 |

fsgnjn.d rd, rs1, rs2 f[rd] = {~f[rs2][63], f[rs1][62:0]}

双精度浮点符号取反注入*(Floating-point Sign Inject-Negate, Double-Precision)*. R-type, RV32D and RV64D.

用f[*rs1*]指数和有效数以及f[*rs2*]的符号的符号位取反，来构造一个新的双精度浮点数，并将其写入f[*rd*]。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0010001 | rs2 | rs1 | 001 | rd | 1010011 |

fsgnjn.s rd, rs1, rs2 f[rd] = {~f[rs2][31], f[rs1][30:0]}

单精度浮点符号取反注入*(Floating-point Sign Inject-Negate, Single-Precision)*. R-type, RV32F and RV64F.

用f[*rs1*]指数和有效数以及f[*rs2*]的符号的符号位取反，来构造一个新的单精度浮点数，并将其写入f[*rd*]。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0010000 | rs2 | rs1 | 001 | rd | 1010011 |

fsgnjx.d rd, rs1, rs2 f[rd] = {f[rs1][63] ^ f[rs2][63], f[rs1][62:0]}

双精度浮点符号异或注入*(Floating-point Sign Inject-XOR, Double-Precision)*. R-type, RV32D and RV64D.

用f[*rs1*]指数和有效数以及f[*rs1*]和f[*rs2*]的符号的符号位异或，来构造一个新的双精度浮点数，并将其写入f[*rd*]。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0010001 | rs2 | rs1 | 010 | rd | 1010011 |

fsgnjx.s rd, rs1, rs2 f[rd] = {f[rs1][31] ^ f[rs2][31], f[rs1][30:0]}

单精度浮点符号异或注入*(Floating-point Sign Inject-XOR, Single-Precision)*. R-type, RV32F and RV64F.

用f[*rs1*]指数和有效数以及f[*rs1*]和f[*rs2*]的符号的符号位异或，来构造一个新的单精度浮点数，并将其写入f[*rd*]。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0010000 | rs2 | rs1 | 010 | rd | 1010011 |

fsqrt.d rd, rs1, rs2 f[rd] =

双精度浮点平方根*(Floating-point Square Root, Double-Precision)*. R-type, RV32D and RV64D.

将f[*rs1*]中的双精度浮点数的平方根舍入和写入f[*rd*]。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0101101 | 00000 | rs1 | rm | rd | 1010011 |

fsqrt.s rd, rs1, rs2 f[rd] =

单精度浮点平方根*(Floating-point Square Root, Single-Precision)*. R-type, RV32F and RV64F.

将f[*rs1*]中的单精度浮点数的平方根舍入和写入f[*rd*]。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0101100 | 00000 | rs1 | rm | rd | 1010011 |

fsrm rd, rs1 t = CSRs[frm]; CSRs[frm] = x[rs1]; x[rd] = t

浮点换出舍入模式 *(Floating-point Swap Rounding Mode)*. 伪指令(Pseudoinstruction), RV32F and RV64F.

把寄存器x[*rs1*]的值写入浮点舍入模式寄存器，并将浮点舍入模式寄存器的原值写入x[*rd*]，等同于**csrrw** rd, frm, rs1。*rd*默认为x0。

fsub.d rd, rs1, rs2 f[rd] = f[rs1] - f[rs2]

双精度浮点减*(Floating-point Subtract, Double-Precision)*. R-type, RV32D and RV64D.

把寄存器f[*rs1*]和f[*rs2*]中的双精度浮点数相减，并将舍入后的差写入f[*rd*]。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0000101 | rs2 | rs1 | rm | rd | 1010011 |

fsub.s rd, rs1, rs2 f[rd] = f[rs1] - f[rs2]

单精度浮点减*(Floating-point Subtract, Single-Precision)*. R-type, RV32F and RV64F.

把寄存器f[*rs1*]和f[*rs2*]中的单精度浮点数相减，并将舍入后的差写入f[*rd*]。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0000100 | rs2 | rs1 | rm | rd | 1010011 |

fsw rs2, offset(rs1) M[x[rs1] + sext(offset)] = f[rs2][31:0]

单精度浮点存储*(Floating-point Store Word)*. S-type, RV32F and RV64F.

将寄存器f[*rs2*]中的单精度浮点数存入内存地址x[*rs1*] + *sign-extend*(*offset*)中。

31 25 24 20 19 15 14 12 11 7 6 0

压缩形式：**c.fswsp** rs2, offset; **c.fsw** rs2, offset(rs1)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| offset[11:5] | rs2 | rs1 | 010 | offset[4:0] | 0100111 |

j offset pc += sext(offset)

跳转 *(Jump)*. 伪指令(Pseudoinstruction), RV32I and RV64I.

把*pc*设置为当前值加上符号位扩展的*offset*，等同于**jal** x0, offset.

jal rd, offset x[rd] = pc+4; pc += sext(offset)

跳转并链接 *(Jump and Link)*. J-type, RV32I and RV64I.

把下一条指令的地址(*pc+4*)，然后把*pc*设置为当前值加上符号位扩展的*offset*。*rd*默认为x1。

压缩形式：**c.j** offset; **c.jal** offset

31 12 11 7 6 0

|  |  |  |
| --- | --- | --- |
| offset[20|10:1|11|19:12] | rd | 1101111 |

jalr rd, offset(rs1) t =pc+4; pc=(x[rs1]+sext(offset))&~1; x[rd]=t

跳转并寄存器链接 *(Jump and Link Register)*. I-type, RV32I and RV64I.

把*pc*设置为x[*rs1*] + *sign-extend*(*offset*)，把计算出的地址的最低有效位设为0，并将原*pc+4*的值写入f[*rd*]。*rd*默认为x1。

压缩形式：**c.jr** rs1; **c.jalr** rs1

31 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| offset[11:0] | rs1 | 010 | rd | 1100111 |

jr rs1 pc = x[rs1]

寄存器跳转 *(Jump Register)*. 伪指令(Pseudoinstruction), RV32I and RV64I.

把*pc*设置为x[*rs1*]，等同于**jalr** x0, 0(rs1)。

la rd, symbol x[rd] = &symbol

地址加载 *(Load Address)*. 伪指令(Pseudoinstruction), RV32I and RV64I.

将*symbol*的地址加载到x[*rd*]中。当编译位置无关的代码时，它会被扩展为对全局偏移量表(Global Offset Table)的加载。对于RV32I，等同于执行**auipc** rd, offsetHi，然后是**lw** rd， offsetLo(rd);对于RV64I，则等同于**auipc** rd，offsetHi和**ld** rd, offsetLo(rd)。如果offset过大，开始的算加载地址的指令会变成两条，先是**auipc** rd, offsetHi然后是**addi** rd, rd, offsetLo。

lb rd, offset(rs1) x[rd] = sext(M[x[rs1] + sext(offset)][7:0])

字节加载 *(Load Byte)*. I-type, RV32I and RV64I.

从地址x[*rs1*] + *sign-extend*(*offset*)读取一个字节，经符号位扩展后写入x[*rd*]。

31 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| offset[11:0] | rs1 | 000 | rd | 0000011 |

lbu rd, offset(rs1) x[rd] = M[x[rs1] + sext(offset)][7:0]

无符号字节加载 *(Load Byte, Unsigned)*. I-type, RV32I and RV64I.

从地址x[*rs1*] + *sign-extend*(*offset*)读取一个字节，经零扩展后写入x[*rd*]。

31 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| offset[11:0] | rs1 | 100 | rd | 0000011 |

ld rd, offset(rs1) x[rd] = M[x[rs1] + sext(offset)][63:0]

双字加载 *(Load Doubleword)*. I-type, RV32I and RV64I.

从地址x[*rs1*] + *sign-extend*(*offset*)读取八个字节，写入x[*rd*]。

压缩形式：**c.ldsp** rd, offset; **c.ld** rd, offset(rs1)

31 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| offset[11:0] | rs1 | 011 | rd | 0000011 |

lh rd, offset(rs1) x[rd] = sext(M[x[rs1] + sext(offset)][15:0])

半字加载 *(Load Halfword)*. I-type, RV32I and RV64I.

从地址x[*rs1*] + *sign-extend*(*offset*)读取两个字节，经符号位扩展后写入x[*rd*]。

31 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| offset[11:0] | rs1 | 001 | rd | 0000011 |

lhu rd, offset(rs1) x[rd] = M[x[rs1] + sext(offset)][15:0]

无符号半字加载 *(Load Halfword, Unsigned)*. I-type, RV32I and RV64I.

从地址x[*rs1*] + *sign-extend*(*offset*)读取两个字节，经零扩展后写入x[*rd*]。

31 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| offset[11:0] | rs1 | 101 | rd | 0000011 |

li rd, immediate x[rd] = immediate

立即数加载 *(Load Immediate)*. 伪指令(Pseudoinstruction), RV32I and RV64I.

使用尽可能少的指令将常量加载到x[*rd*]中。在RV32I中，它等同于执行**lui**和/或**addi**；对于RV64I，会扩展为这种指令序列**lui, addi, slli, addi, slli, addi ,slli, addi。**

lla rd, symbol x[rd] = &symbol

本地地址加载 *(Load Local Address)*. 伪指令(Pseudoinstruction), RV32I and RV64I.

将*symbol*的地址加载到x[*rd*]中。等同于执行**auipc** rd, offsetHi，然后是**addi** rd, rd, offsetLo。

lr.d rd, (rs1) x[rd] = LoadReserved64(M[x[rs1]])

加载保留双字*(Load-Reserved Doubleword)*. R-type, RV64A.

从内存中地址为x[*rs1*]中加载八个字节，写入x[*rd*]，并对这个内存双字注册保留。

31 27 26 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 00010 | aq | rl | 00000 | rs1 | 011 | rd | 0101111 |

lr.w rd, (rs1) x[rd] = LoadReserved32(M[x[rs1]])

加载保留字*(Load-Reserved Word)*. R-type, RV32A and RV64A.

从内存中地址为x[*rs1*]中加载四个字节，符号位扩展后写入x[*rd*]，并对这个内存字注册保留。

31 27 26 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 00010 | aq | rl | 00000 | rs1 | 010 | rd | 0101111 |

lw rd, offset(rs1) x[rd] = sext(M[x[rs1] + sext(offset)][31:0])

字加载 *(Load Word)*. I-type, RV32I and RV64I.

从地址x[*rs1*] + *sign-extend*(*offset*)读取四个字节，写入x[*rd*]。对于RV64I，结果要进行符号位扩展。

压缩形式：**c.lwsp** rd, offset; **c.lw** rd, offset(rs1)

31 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| offset[11:0] | rs1 | 010 | rd | 0000011 |

lwu rd, offset(rs1) x[rd] = M[x[rs1] + sext(offset)][31:0]

无符号字加载 *(Load Word, Unsigned)*. I-type, RV64I.

从地址x[*rs1*] + *sign-extend*(*offset*)读取四个字节，零扩展后写入x[*rd*]。

31 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| offset[11:0] | rs1 | 110 | rd | 0000011 |

lui rd, immediate x[rd] = sext(immediate[31:12] << 12)

高位立即数加载 *(Load Upper Immediate)*. U-type, RV32I and RV64I.

将符号位扩展的20位立即数*immediate*左移12位，并将低12位置零，写入x[*rd*]中。

压缩形式：**c.lui** rd, imm

31 12 11 7 6 0

|  |  |  |
| --- | --- | --- |
| immediate[31:12] | rd | 0110111 |

mret ExceptionReturn(Machine)

机器模式异常返回*(Machine-mode Exception Return)*. R-type, RV32I and RV64I特权架构

从机器模式异常处理程序返回。将*pc*设置为CSRs[mepc], 将特权级设置成CSRs[mstatus].MPP, CSRs[mstatus].MIE置成CSRs[mstatus].MPIE, 并且将CSRs[mstatus].MPIE为1;并且，如果支持用户模式，则将CSR [mstatus].MPP设置为0。

31 25 24 20 19 15 14 12 11 7 6 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0011000 | 00010 | 00000 | 000 | 00000 | 1110011 |