fadd.d rd，rs1，rs2 f [rd] = f [rs1] + f [rs2]
浮点加，双精度。 R 型，RV32D 和 RV64D。
将寄存器 f [rs1]和 f [rs2]中双精度浮点数相加，和舍入后以双精度形式存入 f [rd]。

fadd.s rd，rs1，rs2 f [rd] = f [rs1] + f [rs2]
浮点加，单精度。 R 型，RV32F 和 RV64F。
将寄存器 f [rs1]和 f [rs2]中单精度浮点数相加，和舍入后以单精度形式存入 f [rd]。

fclass.d rd，rs1，rs2 x [rd] = classifyd（f [rs1]）
浮点分类，双精度。 R 型，RV32D 和 RV64D。
将双精度浮点数 F [RS1]的类型以掩码形式写入 x [rd]。有关写入 x [rd]的值的解释，请参阅 fclass.s 指令的说明。

fclass.s rd，rs1，rs2 x [rd] = classifys（f [rs1]）
浮点分类，单精度。 R 型，RV32F 和 RV64F。
将单精度浮点数 F [RS1]的类型以掩码形式写入 x [rd]。根据下表，根据分类结果，x [rd]中仅一位会被置 1.

x[rd] bit Meaning
0 f[rs1] is ?1.
1 f[rs1] is a negative normal number.
2 f[rs1] is a negative subnormal number.
3 f[rs1] is ?0.
4 f[rs1] is +0.
5 f[rs1] is a positive subnormal number.
6 f[rs1] is a positive normal number.
7 f[rs1] is +1.
8 f[rs1] is a signaling NaN.
9 f[rs1] is a quiet NaN.

fcvt.d.l rd，rs1，rs2 f [rd] = f64s64（x [rs1]）
将 Long 型整数转换为 Double 型浮点数。 R 型，仅限 RV64D。
将 x[rs1]中的 64 位二进制补码整数转换为双精度浮点数，并将其写入 f[rd]。

fcvt.d.lu rd，rs1，rs2 f [rd] = f64u64（x [rs1]）
将无符号长整数转换为为双精度浮点数。 R 型，仅限 RV64D。
将 x[rs1]中的 64 位无符号整数转换为双精度浮点数，并将其写入 f [rd]。

fcvt.d.s rd，rs1，rs2 f [rd] = f64f32（f[rs1]）
单点浮点转换为双精度浮点数。 R 型，RV32D 和 RV64D。
将 f[rs1]中的单精度浮点数转换为双精度浮点数，并将其写入 f[rd]。

fcvt.d.w rd，rs1，rs2 f [rd] = f64s32（x [rs1]）

将整数字转换为双精度浮点数。 R 型，RV32D 和 RV64D。
将 x[rs1]中的 32 位二进制补码整数转换为双精度浮点数，并将其写入 f [rd]。

fcvt.d.wu rd，rs1，rs2 f [rd] = f64u32（x [rs1]）
将无符号整数字转换为双精度浮点数。R 型，RV32D 和 RV64D。
将 x[rs1]中的 32 位无符号整数转换为双精度浮点数，并将其写入 f[rd]。

fcvt.l.d rd，rs1，rs2 x [rd] = s64f64（f[rs1]）
将双精度浮点数转换为长整型数。R 型，仅限 RV64D。
将寄存器 f[rs1]中的双精度浮点数转换为 64 位二进制补码整数并将其写入 x[rd]。

fcvt.l.s rd，rs1，rs2 x[rd] = s64f32（f[rs1]）
将单精度浮点数转换为长整型数。 R 型，仅限 RV64F。
将寄存器 f[rs1]中的单精度浮点数转换为 64 位二进制补码整数并将其写入 x[rd]。

fcvt.lu.d rd，rs1，rs2 x [rd] = u64f64（f [rs1]）
将双精度浮点数转换为无符号长整型数。R 型，仅限 RV64D。
将寄存器 f[rs1]中的双精度浮点数转换为 64 位无符号整数并将其写入 x[rd]。

fcvt.lu.s rd，rs1，rs2 x [rd] = u64f32（f [rs1]）
将单精度浮点数转换为无符号长整型数。R 型，仅限 RV64F。
将寄存器 f[rs1]中的单精度浮点数转换为 64 位无符号整数并将其写入 x[rd]。

fcvt.s.d rd，rs1，rs2 f [rd] = f32f64（f[rs1]）
从双精度浮点数转换为单精度浮点数。R 型，RV32D 和 RV64D。
将 f[rs1]中的双精度浮点数转换为单精度浮点数并将其写入 f[rd]。

fcvt.s.l rd，rs1，rs2 f[rd] = f32s64（x[rs1]）
将长整型数转换为单精度浮点数。R 型，仅限 RV64F。
将 x[rs1]中的 64 位二进制补码整数转换为单精度浮点数，并将其写入 f[rd]。

fcvt.s.lu rd，rs1，rs2 f [rd] = f32u64（x[rs1]）
将无符号长整型数浮点转换为单精度浮点数。R 型，仅限 RV64F。
将 x[rs1]中的 64 位无符号整数转换为单精度浮点数，并把它写到 f[rd]。

fcvt.s.w rd，rs1，rs2 f [rd] = f32s32（x[rs1]）
将整数字转换为单精度浮点数。R 型，RV32F 和 RV64F。
将 x[rs1]中的 32 位二进制补码整数转换为单精度浮点数，并将其写入 f[rd]。

fcvt.s.wu rd，rs1，rs2 f [rd] = f32u32（x[rs1]）
将无符号整数字转换为单精度浮点数。R 型，RV32F 和 RV64F。
将 x[rs1]中的 32 位无符号整数转换为单精度浮点数，并把它写到 f[rd]。

fcvt.w.d rd，rs1，rs2 x [rd] = sext（s32f64（f[rs1]））

将双精度浮点数转换为整数字。R 型，RV32D 和 RV64D。
将寄存器 f[rs1]中的双精度浮点数转换为 32 位二进制补码整数，结果符号扩展后写入 x[rd]。

fcvt.wu.d rd，rs1，rs2 x [rd] = sext（u32f64（f[rs1]））
将双精度浮点数转换为无符号整型数。R 型，RV32D 和 RV64D。
将寄存器 f[rs1]中的双精度浮点数转换为 32 位无符号整数，结果符号扩展后写入 x[rd]。

fcvt.w.s rd，rs1，rs2 x [rd] = sext（s32f32（f[rs1]））
将单精度浮点数转换为整型数。R 型，RV32F 和 RV64F。
将寄存器 f[rs1]中的单精度浮点数转换为 32 位二进制补码整数，结果符号扩展后写入 x[rd]。

fcvt.wu.s rd，rs1，rs2 x [rd] = sext（u32f32（f [rs1]））
将单精度浮点数转换为无符号整型数。R 型，RV32F 和 RV64F。
将寄存器 f[rs1]中的单精度浮点数转换为 32 位无符号整数，结果符号扩展后写入 x [rd]。

fdiv.d rd，rs1，rs2 f [rd] = f [rs1]÷f [rs2]
双精度浮点数除法。R 型，RV32D 和 RV64D。
将寄存器 f[rs1]中的双精度浮点数除以 f[rs2]中的浮点数，商舍入后以双精度形式存入 f[rd]。

fdiv.s rd，rs1，rs2 f [rd] = f [rs1]÷f [rs2]
单精度浮点数除法。R 型，RV32F 和 RV64F。
将寄存器 f[rs1]中的单精度浮点数除以 f[rs2]中的书，商舍入后以单精度形式写入 f [rd]。

栅栏 pred，succ 栅栏（pred，succ）
栅栏内存和 I/O. I 型，RV32I 和 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。

fence.i 围栏 同步（数据存储，取指令）
同步指令流。I 型，RV32I 和 RV64I。
使对内存指令区域的读写，对后续取指令可见。

feq.d rd，rs1，rs2 x[rd] = f[rs1] == f[rs2]
浮点数等于判断，双精度。R 型，RV32D 和 RV64D。
如果 f[rs1]中的双精度浮点数等于 f[rs2]中的浮点数，则将 1 写入到 x[rd]；否则写入 0。

feq.s rd，rs1，rs2 x[rd] = f[rs1] == f[rs2]
浮点等于，单精度。 R 型，RV32F 和 RV64F。
如果 f[rs1]中的单精度浮点数等于 f[rs2]中的浮点数，则写入 1 到 x[rd]；否则，写入 0。

fld rd，offset（rs1）f[rd] = M[x[rs1] + sext（offset）] [63：0]
加载双精度。I 型，RV32D 和 RV64D。

从内存地址 x[rs1] + 偏移量（符号扩展），加载双精度浮点数，并将其写入 f [rd]。

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

fle.d rd，rs1，rs2 x[rd] = f[rs1] ≤f[rs2]
浮点数小于或等于比较，双精度。R 型，RV32D 和 RV64D。
如果 f[rs1]中的双精度浮点数小于或等于 f[rs2]中的数，则将 1 写入到 x[rd]；否则写入 0。

fle.s rd，rs1，rs2 x[rd] = f[rs1] ≤f[rs2]
浮点数小于或等于，单精度。R 型，RV32F 和 RV64F。
如果 f[rs1]中的单精度浮点数小于或等于 f[rs2]中的数字，则将 1 写入到 x[rd]，否则写入 0。

flt.d rd，rs1，rs2 x[rd] = f[rs1] < f[rs2]
浮点数小于比较，双精度。R 型，RV32D 和 RV64D。
如果 f[rs1]中的双精度浮点数小于 f[rs2]中的数，则将 1 写入到 x[rd]，否则写入 0。

flt.s rd，rs1，rs2 x[rd] = f[rs1] < f[rs2]
浮点数小于比较，单精度。R 型，RV32F 和 RV64F。
如果 f[rs1]中的单精度浮点数小于 f[rs2]中的数，则将 1 写入到 x[rd]，否则写入 0。

flw rd，offset（rs1）f[rd] = M[x[rs1] + sext（offset）] [31：0]
浮点数加载字。 I 型，RV32F 和 RV64F。
从内存地址 x[rs1] + 偏移量（符号扩展）处加载单精度浮点数，并将其写入 f[rd]。
压缩形式指令：c.flwsp rd，offset; c.flw rd，offset（rs1）

fmadd.d rd，rs1，rs2，rs3 f [rd] = f [rs1]×f [rs2] + f [rs3]
浮点数乘加，双精度。R4 型，RV32D 和 RV64D。
将 f[rs1]和 f[rs2]中的双精度浮点数相乘，将未舍入的乘积与 f[rs3]中的双精度浮点数相加，结果舍入后以双精度形式存入 f[rd]。

fmadd.s rd，rs1，rs2，rs3 f[rd] = f[rs1] ×f[rs2] + f[rs3]
浮点数乘加，单精度。R4 型，RV32F 和 RV64F。
将 f[rs1]和 f[rs2]中的单精度浮点数相乘，将未舍入的乘积与 f[rs3]中的单精度浮点数相加，结果舍入后以单精度形式存入 f[rd]。

fmax.d rd，rs1，rs2　f[rd] = max（f[rs1]，f[rs2]）
浮点数取最大值，双精度。R 型，RV32D 和 RV64D。
将寄存器 f[rs1]和 f[rs2]中较大的双精度浮点数复制到 f[rd]。

fmax.s rd, rs1, rs2 f[rd] = max(f[rs1], f[rs2])
浮点数取最大值，单精度。R 型，RV32F 和 RV64F。
将寄存器 f[rs1]和 f[rs2]中较大的单精度浮点数复制到 f[rd]。

fmin.d rd, rs1, rs2 f[rd] = min(f[rs1], f[rs2])
浮点数取最小值，双精度。R 型，RV32D 和 RV64D。
将寄存器 f[rs1]和 f[rs2]中较小的双精度浮点数复制到 f[rd]。

fmin.s rd, rs1, rs2 f[rd] = min(f[rs1], f[rs2])
浮点数取最小值，单精度。R 型，RV32F 和 RV64F。
将寄存器 f[rs1]和 f[rs2]中较小的单精度浮点数复制到 f[rd]。

fmsub.d rd, rs1, rs2, rs3 f[rd] = f[rs1]×f[rs2]-f[rs3]
浮点数乘减，双精度。R4 型，RV32F 和 RV64F。
将 f[rs1]和 f[rs2]中的双精度浮点数相乘，用未舍入的乘积减去 f[rs3]中的双精度浮点数，结果舍入后以双精度形式存入 f[rd]。

fmsub.s rd, rs1, rs2, rs3 f[rd] = f[rs1]×f[rs2]-f[rs3]
浮点数乘减，单精度。R4 型，RV32D 和 RV64D。
将 f[rs1]和 f[rs2]中的单精度浮点数相乘，用未舍入的乘积减去 f[rs3]中的单精度浮点数，结果舍入后以单精度形式存入 f[rd]。

fmul.d rd，rs1，rs2 f[rd] = f[rs1] ×f[rs2]
浮点数乘法，双精度。R 型，RV32D 和 RV64D。
将寄存器 f[rs1]和 f[rs2]中的双精度浮点数相乘，结果舍入后以双精度形式写入 f[rd]。

fmul.s rd, rs1, rs2 f[rd] = f[rs1] × f[rs2]
浮点数乘法，单精度。R 型，RV32F 和 RV64F。
将寄存器 f[rs1]和 f[rs2]中的单精度浮点数相乘，结果舍入后以单精度形式写入 f[rd]。

fmv.d rd，rs1 f [rd] = f [rs1]

浮点数搬运指令。伪指令，RV32D 和 RV64D。
将 f[rs1]中的双精度浮点数复制到 f[rd]。扩展成 fsgnj.d rd，rs1，rs1。

fmv.d.x rd，rs1，rs2 f[rd] = x[rs1][63：0]
浮点数搬运指令：将双字从整数寄存器移动到浮点寄存器。R 型，仅限 RV64D。
将寄存器 x[rs1]中的双精度浮点数复制到 f[rd]。

fmv.s rd，rs1 f [rd] = f [rs1]
浮点数搬运指令。伪指令，RV32F 和 RV64F。
将 f[rs1]中的单精度浮点数复制到 f[rd]。扩展为 fsgnj.s rd，rs1，rs1。

fmv.w.x rd，rs1，rs2 f[rd] = x[rs1] [31：0]
浮点数搬运指令：将字从整数寄存器移动到浮点寄存器。R 型，RV32F 和 RV64F。
将寄存器 x[rs1]中的单精度浮点数复制到 f[rd]。

fmv.x.d rd, rs1, rs2 x[rd] = f[rs1][63:0]
浮点数搬运指令：将双字从浮点寄存器移动到整数寄存器。R 型，仅限 RV64D。
将寄存器 f[rd]中的双精度浮点数复制到 x[rs1]。

fmv.x.w rd，rs1，rs2 x[rd] = sext（f[rs1][31：0]）
浮点数搬运指令：将单字从浮点寄存器移动到整数寄存器。R 型，RV64F 以及 RV64F。
将寄存器 f[rd]中的单精度浮点数复制到 x[rs1]。

fneg.d rd，rs1 f [rd] = -f [rs1]
浮点数取反。伪指令，RV32D 和 RV64D。
将 f[rs1]中的双精度浮点数的取反后写入 f[rd]。拓展为 fsgnjn.d rd，rs1，rs1。

fneg.s rd, rs1 f[rd] = -f[rs1]
浮点数取反。伪指令，RV32F 和 RV64F。
将 f[rs1]中的双精度浮点数的取反后写入 f[rd]。拓展为 fsgnjn.s rd, rs1, rs1。

fnmadd.d rd，rs1，rs2，rs3 f [rd] = -f [rs1]×f [rs2] -f [rs3]
浮点数乘加取负，双精度。 R4 型，RV32D 和 RV64D。
将 f[rs1]和 f[rs2]中的双精度浮点数相乘，对结果取反，用未舍入的（取反后的）乘积中减去 f[rs3]中的双精度浮点数，将最终结果舍入后以双精度形式写入 f[rd]。

fnmadd.s rd, rs1, rs2, rs3 f[rd] = -f[rs1]×f[rs2]-f[rs3]
浮点数乘加取负，单精度。 R4 型，RV32F 和 RV64F。
将 f[rs1]和 f[rs2]中的单精度浮点数相乘，对结果取反，用未舍入的（取反后的）乘积中减去 f[rs3]中的单精度浮点数，将最终结果舍入后以单精度形式写入 f[rd]。

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

浮点数乘减取负，双精度。 R4 型，RV32D 和 RV64D。
将 f[rs1]和 f[rs2]中的双精度浮点数相乘，对结果取反，用未舍入的（取反后的）乘积中加上 f[rs3]中的双精度浮点数，将最终结果舍入后以双精度形式写入 f[rd]。

fnmsub.s rd, rs1, rs2, rs3 f[rd] = -f[rs1]×f[rs2]+f[rs3]
浮点数乘减取负，单精度。 R4 型，RV32F 和 RV64F。
将 f[rs1]和 f[rs2]中的单精度浮点数相乘，对结果取反，用未舍入的（取反后的）乘积中加上 f[rs3]中的单精度浮点数，将最终结果舍入后以单精度形式写入 f[rd]。

frcsr rd x [rd] = CSRs [fcsr]
读浮点数控制和状态寄存器。伪指令，RV32F 和 RV64F。
将浮点数控制和状态寄存器复制到 x[rd]。扩展为 csrrs rd，fcsr，x0。

frflags rd x [rd] = CSRs [fflags]
读取浮点数异常标志。伪指令，RV32F 和 RV64F。
将浮点数异常标志复制到 x[rd]。扩展为 csrrs rd，fflags，x0。

frrm rd x[rd] = CSRs [frm]
读取浮点数舍入模式。伪指令，RV32F 和 RV64F。
将浮点数舍入模式复制到 x[rd]。扩展为 csrrs rd，frm，x0。

fscsr rd，rs1 t = CSRs [fcsr]; CSRs [fcsr] = x [rs1]; x [rd] = t
交换浮点数控制和状态寄存器。伪指令，RV32F 和 RV64F。
将 x[rs1]的值复制到浮点控制和状态寄存器，然后将浮点控制和状态寄存器先前的值复制到 x[rd]。扩为为 csrrw rd，fcsr，rs1。如果 rd 省略，假定为 x0。

fsd rs2，offset（rs1）M [x [rs1] + sext（offset）] = f [rs2] [63：0]
浮点存储双字。S 型，RV32D 和 RV64D。
将寄存器 f[rs2]中的双精度浮点数存储到存储器中 x[rs1] + 偏移量（符号扩展）地址处。
压缩形式：c.fsdsp rs2，offset; c.fsd rs2，offset（rs1）

fsflags rd，rs1 t = CSRs [fflags]; CSRs [fflags] = x [rs1]; x [rd] = t
浮点数交换异常标志位。伪指令，RV32F 和 RV64F。
将 x[rs1]复制到浮点异常标志寄存器，然后将浮点异常标志先前的值复制到 x[rd]。扩展为 csrrw rd，fflags，rs1。如果省略 rd，则默认为 x0。

fsgnj.d rd，rs1，rs2 f [rd] = {f [rs2] [63]，f [rs1] [62：0]}
浮点符号注入，双精度。R 型，RV32D 和 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]。

fsgnj.s rd, rs1, rs2 f[rd] = {f[rs2][31], f[rs1][30:0]}
浮点符号注入，单精度。R 型，RV32F 和 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]。

fsgnjn.d rd, rs1, rs2 f[rd] = { ~ f[rs2][63], f[rs1][62:0]}
浮点符号取反注入，双精度。R 型，RV32D 和 RV64D。
用 f[rs1]指数和有效数以及 f[rs2]的符号的符号位（取反后），来构造一个新的双精度浮点数，并将其写入 f[rd]。

fsgnj.s rd, rs1, rs2 f[rd] = {f[rs2][31], f[rs1][30:0]}
浮点符号取反注入，单精度。R 型，RV32F 和 RV64F。
用 f[rs1]指数和有效数以及 f[rs2]的符号的符号位（取反后），来构造一个新的单精度浮点数，并将其写入 f[rd]。

fsgnjx.d rd，rs1，rs2 f [rd] = {f [rs1] [63] ^ f [rs2] [63]，f [rs1] [62：0]}
浮点符号异或注入，双精度。R 型，RV32D 和 RV64D。
浮点符号注入，双精度。R 型，RV32D 和 RV64D。
从 f[rs1]以及 f[rs2]构造出一个新的浮点数，它的指数以及有效数与 f[rs1]相同，符号位为 f[rs1]的符号异或 f [rs2]符号，将这个新的浮点数写入 f [rd]。

fsgnjx.s rd, rs1, rs2 f[rd] = {f[rs1][31] ^ f[rs2][31], f[rs1][30:0]}
浮点符号异或注入，单精度。R 型，RV32F 和 RV64F。
浮点符号注入，单精度。R 型，RV32D 和 RV64D。
从 f[rs1]以及 f[rs2]构造出一个新的浮点数，它的指数以及有效数与 f[rs1]相同，符号位为 f[rs1]的符号异或 f [rs2]符号，将这个新的浮点数写入 f [rd]。

fsqrt.d rd，rs1，rs2 f [rd] = pf [rs1]
浮点数算平方根，双精度。 R 型，RV32D 和 RV64D。
计算寄存器 f[rs1]中的双精度浮点数的平方根，结果舍入后以双精度形式写入 f [rd]。

fsqrt.s rd，rs1，rs2 f [rd] = pf [rs1]
浮点数算平方根，单精度。 R 型，RV32F 和 RV64F。
计算寄存器 f[rs1]中的单精度浮点数的平方根，结果舍入后以单精度形式写入 f [rd]。

fsrm rd，rs1 t = CSRs [frm]; CSRs [frm] = x [rs1]; x [rd] = t
浮点交换舍入模式。伪指令，RV32F 和 RV64F。
将 x[rs1]的值复制到浮点舍入模式寄存器，然后将浮点舍入模式寄存器原来的值复制到 x[rd]。
扩展为 csrrw rd，frm，rs1。如果省略 rd，则默认为 x0。

fsub.d rd，rs1，rs2 f[rd] = f[rs1] - f[rs2]
浮点减法，双精度。 R 型，RV32D 和 RV64D。
用 f[rs1]中的双精度浮点数减去 f[rs2]中的数，差值舍入后以双精度形式写入 f[rd]。

fsub.s rd, rs1, rs2 f[rd] = f[rs1] - f[rs2]
浮点减法，单精度。 R 型，RV32F 和 RV64F。
用 f[rs1]中的单精度浮点数减去 f[rs2]中的数，差值舍入后以单精度形式写入 f[rd]。

fsw rs2，offset（rs1）M [x [rs1] + sext（offset）] = f [rs2] [31：0]
浮点存储字。S 型，RV32F 和 RV64F。
将寄存器 f[rs2]中的单精度浮点数存储到存储器 x[rs1] + 偏移量（符号扩展）地址处。
压缩形式：c.fswsp rs2，offset; c.fsw rs2，offset（rs1）

j offset pc + = sext（offset）
跳转。伪指令，RV32I 和 RV64I。
将 pc 设置为当前 pc 加上偏移量（符号扩展）。扩展为 jal x0，offset。

jal rd，offset x [rd] = pc + 4; pc + = sext（偏移量）
跳转和链接。J 型，RV32I 和 RV64I。
将下一条指令（pc + 4）的地址写入 x[rd]，然后将 pc 设置为当前 pc 加上偏移量（符号扩展）。如果省略 rd，则假定为 x1。
压缩形式：c.j offset; c.jal 抵消

jalr rd, offset(rs1) t =pc+4; pc=(x[rs1]+sext(offset))& ~ 1; x[rd]=t
跳转和链接注册。 I 型，RV32I 和 RV64I。
将 pc 设置为 x[rs1] + 偏移量（符号扩展），将计算地址的最低有效位覆盖为 0，然后将之前 pc 值 + 4 写入 x [rd]。如果省略 rd，则假定为 x1。
压缩形式：c.jr rs1; c.jalr rs1

jr rs1 pc = x[rs1]
寄存器跳转。伪指令，RV32I 和 RV64I。
将 pc 设置为 x[rs1]。扩展为 jalr x0,0（rs1）。

la rd，symbol x [rd] = &symbol
加载地址。伪指令，RV32I 和 RV64I。
将符号的地址加载到 x[rd]中。当被编译成位置无关代码时，它会被扩展为对全局偏移量表的加载：例如对于 RV32I，会生成 auipc rd，offsetHi，然后是 lw rd，offsetLo（RD）;对于 RV64I，则是 auipc rd，offsetHi 和 ld rd，offsetLo（rd）。如果 offst 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]）
加载字节。 I 型，RV32I 和 RV64I。
从内存地址 x[rs1] + 偏移量（符号扩展）处加载一个字节，将该字节符号扩展后存入寄存器 x[rd]。

lbu rd, offset(rs1) x[rd] = M[x[rs1] + sext(offset)][7:0]
无符号加载字节。I 型，RV32I 和 RV64I。
从内存地址 x[rs1] + 偏移量（符号扩展）处加载一个字节，将该字节无符号扩展后存入寄存器 x[rd]。

ld rd，offset（rs1）x [rd] = M [x [rs1] + sext（offset）] [63：0]
加载双字。I 型，仅限 RV64I。
从内存地址 x[rs1]+ 偏移量（符号扩展）处加载 8 个字节并将其写入 X[RD]。
压缩形式：c.ldsp rd，offset; c.ld rd，offset（rs1）

lh rd, offset(rs1) x[rd] = sext(M[x[rs1] + sext(offset)][15:0])
加载半字。 I 型，RV32I 和 RV64I。
从内存地址 x[rs1] + 偏移量（符号扩展）处加载两个字节，将这两个字节符号扩展后存入寄存器 x[rd]。

lhu rd, offset(rs1) x[rd] = M[x[rs1] + sext(offset)][15:0]
无符号加载半字。 I 型，RV32I 和 RV64I。
从内存地址 x[rs1] + 偏移量（符号扩展）处加载两个字节，将这两个字节无符号扩展后存入寄存器 x[rd]。

li rd, immediate x[rd] = immediate
加载立即数。伪指令，RV32I 和 RV64I。
使用尽可能少的指令将常量加载到 x[rd]中。对于 RV32I, 它扩展为 lui 和/或 addi;对于 RV64I, 可能扩展为这种指令序列 lui，addi，slli，addi，slli，addi，slli，addi。(不太理解这个？？？）

lla rd, symbol x[rd] = &symbol
加载本地地址。伪指令，RV32I 和 RV64I。
将符号的地址加载到 x[rd]中。扩展为如下指令序列，先是 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。R 型，仅限 RV64A。
从内存地址 x[rs1]处加载 8 个字节，将它们写入 x[rd]，并对这个内存双字注册保留。

lr.w rd, (rs1) x[rd] = LoadReserved32(M[x[rs1]])
加载保留字。R 型，RV32A 和 RV64A。
从内存地址 x[rs1]处加载 4 个字节，将它们写入 x[rd]，并对这个内存字注册保留。

lw rd，offset（rs1）x [rd] = sext（M [x [rs1] + sext（offset）] [31：0]）
加载字。I 型，RV32I 和 RV64I。
从内存地址 x[rs1] + 偏移量（符号扩展）处加载四个字节并将其写入 x[RD]。对于 RV64I, 加载上来的数被符号扩展后再存入目的寄存器。
压缩形式：c.lwsp rd，offset; c.lw rd，offset（rs1）

lwu rd, offset(rs1) x[rd] = M[x[rs1] + sext(offset)][31:0]
无符号加载字。I 型，RV32I 和 RV64I。
从内存地址 x[rs1] + 偏移量（符号扩展）处加载四个字节并将其写入 x[RD]。对于 RV64I，加载上来的数被无符号扩展后再存入目的寄存器。

lui rd，immediate x [rd] = sext（立即[31:12] << 12）
加载立即数到高位。U 型，RV32I 和 RV64I。
将 20 位立即数符号扩展后，左移 12 位写入 x[rd]，目的寄存器低 12 位清零。
压缩形式：c.lui rd，imm

mret ExceptionReturn（机器）
机器模式异常返回。R 型，RV32I 和 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。