Category		Fmt	RV32I base	machine code(bin)	comment
loads	load byte	I	lb rd, rs1, imm	[31-20,imm][19-15,rs1]000[11-7,rd]0000011	rd=mem[rs1+imm], 8bit数据符号位扩展后返回rd
	load half word	I	lh rd, rs1, imm	[31-20,imm][19-15,rs1]001[11-7,rd]0000011	rd=mem[rs1+imm],16bit数据符号扩展后返回rd,应该保证地址对齐，否则产生异常
	load word	I	lw rd, rs1, imm	[31-20,imm][19-15,rs1]010[11-7,rd]0000011	rd=mem[rs1+imm],32bit数据返回rd，应该保证地址对齐，否则产生异常
	load byte unsinged	I	lbu rd, rs1, imm	[31-20,imm][19-15,rs1]100[11-7,rd]0000011	rd=mem[rs1+imm], 8bit数据高位补0后返回rd
	load half unsinged	I	lhu rd, rs1, imm	[31-20,imm][19-15,rs1]101[11-7,rd]0000011	rd=mem[rs1+imm], 16bit数据高位补0后返回rd
stores	store byte	S	sb rs1, rs2, imm	[31-25,imm[11-5]][24-20,rs2,][19-15,rs1]000[11-7,imm[4-0]]0100011	mem[rs1+imm]=rs2的低8bit数据
	store half word	S	sh rs1, rs2, imm	[31-25,imm[11-5]][24-20,rs2,][19-15,rs1]001[11-7,imm[4-0]]0100011	mem[rs1+imm]=rs2的低16bit数据
	store word	S	sw rs1, rs2, imm	[31-25,imm[11-5]][24-20,rs2,][19-15,rs1]010[11-7,imm[4-0]]0100011	mem[rs1+imm]=rs2的32bit数据
Category		Fmt	RV32I base	machine code(bin)	comment
arithmetric	add	R	add rd, rs1, rs2	0000000[24-20,rs2][19-15,rs1]000[11-7,rd]0110011	rd=rs1+rs2，如果溢出，舍弃高位，保留低32bit
	add immediate	I	addi rd, rs1, imm	[31-20,imm][19-15,rs1]000[11-7,rd]0010011	rd=rs1+imm,12bit立即数会符号位扩展，结果写回rd时如果溢出，则舍弃高位，保留低32bit
	subtract	R	sub rd, rs1, rs2	0100000[24-20,rs2][19-15,rs1]000[11-7,rd]0110011	rd=rs1-rs2，如果溢出，舍弃高位，保留低32bit
	load upper imm	U	lui rd, imm	[31-12,imm][11-7,rd]0110111	load imm to high 20 bits of rd, and place 0 in low 12 bits
	add upper imm to PC	U	auipc rd, imm	[31-12,imm][11-7,rd]0010111	pc=pc+high 20 bits imm&low 12 bits 0, then result save to rd.
logical	xor	R	xor rd, rs1, rs2	0000000[24-20,rs2][19-15,rs1]100[11-7,rd]0110011	rd=rs1^rs2, bitwise operation
	xor immediate	I	xori rd, rs1, imm	[31-20,imm][19-15,rs1]100[11-7,rd]0010011	rd=rs1^imm, i符号扩展12bit数imm，结果放在rd。
	or	R	or rd, rs1, rs2	0000000[24-20,rs2][19-15,rs1]110[11-7,rd]0110011	rd=rs1|rs2, bitwise operation
	or immediate	I	ori rd, rs1, imm	[31-20,imm][19-15,rs1]110[11-7,rd]0010011	rd=rs1|imm，符号扩展12bit数imm，结果放在rd。
	and	R	and rd, rs1, rs2	0000000[24-20,rs2][19-15,rs1]111[11-7,rd]0110011	rd=rs1&rs2, bitwise operation
	and immediate	I	andi rd, rs1, imm	[31-20,imm][19-15,rs1]111[11-7,rd]0010011	rd=rs1&imm, i符号扩展12bit数imm，结果放在rd。
shifts	shift left	R	SLL rd, rs1, rs2	0000000[24-20,rs2][19-15,rs1]001[11-7,rd]0110011	rd=rs1<<rs2，rs1左移值为rs2的低5bit，低位补0
	shift left immediate	I	SLLI rd, rs1, shamt	0000000[24-20,imm][19-15,rs1]001[11-7,rd]0010011	rd=rs1<<shamt，rs1移动5bit立即数，低位补0
	shift right	R	SRL rd, rs1, rs2	0000000[24-20,rs2][19-15,rs1]101[11-7,rd]0110011	rd=rs1>>rs2，高位补0，右移值为rs2中的低5位
	shift right immediate	I	SRLI rd, rs1, shamt	0000000[24-20,imm][19-15,rs1]101[11-7,rd]0010011	rd=r1>>shamt，高位补0，右移值为5bit立即数
	shift right arithmetirc	R	SRA rd, rs1, rs2	0100000[24-20,rs2][19-15,rs1]101[11-7,rd]0110011	rd=rs1>>rs2，高位补符号位，右移值为rs2中的低5位
	shift right arith imm	I	SRAI rd, rs1, shamt	0100000[24-20,imm][19-15,rs1]101[11-7,rd]0010011	右移时，右边补符号位
compare	set <	R	slt rd, rs1, rs2	0000000[24-20,rs2][19-15,rs1]010[11-7,rd]0110011	if rs1< rs2, write 1 to rd, 有符号数比较
	set < immediate	I	slti rd, rs1, imm	[31-20,imm][19-15,rs1]010[11-7,rd]0010011	if rs1<imm, write 1 to rd，imm会符号位扩展，所以进行有符号数比较
	set < unsigned	R	sltu rd, rs1, rs2	0000000[24-20,rs2][19-15,rs1]011[11-7,rd]0110011	if rs1< rs2, write 1 to rd, 无符号数比较
	set < imm unsigned	I	sltiu rd, rs1, imm	[31-20,imm][19-15,rs1]011[11-7,rd]0010011	if rs1<imm, write 1 to rd，imm会符号位扩展，所以进行有符号数比较
Category		Fmt	RV32I base	machine code(bin)	comment
	branch =	SB	beq rs1, rs2,imm	[31-25, imm[12][10:5]][24-20, rs2][19-15, rs1]000[11-7, imm[4:1][11]]1100011	if(rs1==rs2) pc=pc+imm*2,跳转指令

branch	branch <>	SB	bne rs1, rs2,imm	[31-25, imm[12][10:5]][24-20, rs2][19-15, rs1]001[11-7, imm[4:1][11]]1100011	if(rs1!=rs2) pc=pc+imm*2,跳转指令
	branch <	SB	blt rs1, rs2,imm	[31-25, imm[12][10:5]][24-20, rs2][19-15, rs1]100[11-7, imm[4:1][11]]1100011	if(rs1<rs2) pc=pc+imm*2,跳转指令 ， 有符号数
	branch >=	SB	bge rs1, rs2,imm	[31-25, imm[12][10:5]][24-20, rs2][19-15, rs1]101[11-7, imm[4:1][11]]1100011	if(rs1>rs2) pc=pc+imm*2,跳转指令，有符号数
	branch < unsigned	SB	bltu rs1, rs2,imm	[31-25, imm[12][10:5]][24-20, rs2][19-15, rs1]110[11-7, imm[4:1][11]]1100011	if(rs1<rs2) pc=pc+imm*2,跳转指令，无符号数
	branch >=unsigned	SB	bgeu rs1, rs2,imm	[31-25, imm[12][10:5]][24-20, rs2][19-15, rs1]111[11-7, imm[4:1][11]]1100011	if(rs1>rs2) pc=pc+imm*2,跳转指令，无符号数
jump and link	J&L	UJ	JAL rd, imm	[31-12, imm[20][10:1][11][19:12]][11-7,rd]1101111	pc=pc+imm*2, 现在的pc+4写入rd寄存器
	Jump and link register	I	JALR, rd, rs1, imm	[31-20,imm[11:0]][19-15,rs1]000[11-7,rd]1100111	pc=rs1+imm, pc+4写入到rd寄存器