1.

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	7

2.

auipc rd, immediate

x[rd] = pc + sext(immediate[31:12] << 12)

PC 加立即数 (Add Upper Immediate to PC). U-type, RV32I and RV64I.

把符号位扩展的 20 位(左移 12 位)立即数加到 pc 上,结果写入 x[rd]。

31	12 11 7	6 0
immediate[31:12]	rd	0010111

3.

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	

4.

jalr rd, offset(rs1)

t = pc+4; $pc=(x[rs1]+sext(offset))&\sim1$; 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[rsI] + 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 2	0 19 15	14 12	2 11 7	6 0	_
offset[11:0]	rs1	010	rd	1100111	1

5.

beq rs1, rs2, offset

if (rs1 == rs2) pc += sext(offset)

相等时分支 (Branch if Equal). B-type, RV32I and RV64I.

若寄存器 x[rsI]和寄存器 x[rs2]的值相等,把 pc 的值设为当前值加上符号位扩展的偏移 offset。 压缩形式: **c.beqz** rs1, offset

31	25 24	20 19	15 1	14 12	2 11 7	6 0
offset[12 10:5]	rsi	2	rs1	000	offset[4:1 11]	1100011

bne rs1, rs2, offset

if (rs1 \neq rs2) pc += sext(offset)

不相等时分支 (Branch if Not Equal). B-type, RV32I and RV64I.

若寄存器 $\mathbf{x}[rs1]$ 和寄存器 $\mathbf{x}[rs2]$ 的值不相等,把 pc 的值设为当前值加上符号位扩展的偏移 offset。

压缩形式: c.bnez rs1, offset

31	25 24	20 19	15 14	4 12	11 7	6 0
offset[12 10:5]	rs2	1	rs1	001	offset[4:1 11]	1100011

7.

blt rs1, rs2, offset

if $(rs1 <_s rs2)$ pc += sext(offset)

小于时分支 (Branch if Less Than). B-type, RV32I and RV64I.

若寄存器 x[rsI]的值小于寄存器 x[rs2]的值(均视为二进制补码),把 pc 的值设为当前值加上符号位扩展的偏移 offset。

31	25 24	20 19	15	14 12	11 7	6 0	
offset[12 10:5]	rs2		rs1	100	offset[4:1 11]	1100011	

8.

bge rs1, rs2, offset

if (rs1 ≥s rs2) pc += sext(offset)

大于等于时分支 (Branch if Greater Than or Equal). B-type, RV32I and RV64I.

若寄存器 x[rs1]的值大于等于寄存器 x[rs2]的值(均视为二进制补码),把 pc 的值设为当前值加上符号位扩展的偏移 offset。

31	25 24	20 19	15	14 12	2 11 7	6 0
offset[12 10:5]	rs2		rs1	101	offset[4:1 11]	1100011

9.

bltu rs1, rs2, offset

if (rs1 $<_u$ rs2) pc += sext(offset)

无符号小于时分支 (Branch if Less Than, Unsigned). B-type, RV32I and RV64I.

若寄存器 x[rsI]的值小于寄存器 x[rs2]的值(均视为无符号数),把 pc 的值设为当前值加上符号位扩展的偏移 offset。

31	25 24 20	19 15	14 12	2 11 7	6 0
offset[12 10:5]	rs2	rs1	110	offset[4:1 11]	1100011

10.

bgeu rs1, rs2, offset

if (rs1 \geqslant_u rs2) pc += sext(offset)

无符号大于等于时分支 (Branch if Greater Than or Equal, Unsigned). B-type, RV32I and RV64I. 若寄存器 x[rs1]的值大于等于寄存器 x[rs2]的值(均视为无符号数),把 pc 的值设为当前值加上符号位扩展的偏移 offset。

31	25 24	20 19	15	14 12	2 11 7	6	0
offset[12 10:5]	rs	2	rs1	111	offset[4:1 11]	1100011	

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	0 19 1	5 14 1	2 11 7	6 0
	offset[11:0]	rs1	000	rd	0000011

12.

Ih rd, offset(rs1)

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

半字加載 (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	2 11 7	6 0	
	offset[11:0]	rs1	001	rd	0000011	

13.

w 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]	rs	1 010) rd	0000011

14.

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]	rs	1 100	rd	0000011

15.

lhu rd, offset(rs1)

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

无符号半字加載 (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	2 11 7	6 0	_
offset[11:0]	rs1	101	rd	0000011	

sb rs2, offset(rs1)

M[x[rs1] + sext(offset) = x[rs2][7:0]

存字节(Store Byte). S-type, RV32I and RV64I.

将 x[rs2]的低位字节存入内存地址 x[rs1]+sign-extend(offset)。

31	2	5 24	20 19	15 14	12	11 7	6	0
	offset[11:5]	rs2	rs	1	000	offset[4:0]	0100011	

17.

sh rs2, offset(rs1)

$$M[x[rs1] + sext(offset) = x[rs2][15:0]$$

存半字(Store Halfword). S-type, RV32I and RV64I.

将 x[rs2]的低位 2 个字节存入内存地址 x[rs1]+sign-extend(offset)。

1	31 25	24 20	19 15	14 12	11 7	6 0)
	offset[11:5]	rs2	rs1	001	offset[4:0]	0100011	7

18.

SW rs2, offset(rs1)

$$M[x[rs1] + sext(offset) = x[rs2][31:0]$$

存字(Store Word). S-type, RV32I and RV64I.

将 x[rs2]的低位 4 个字节存入内存地址 x[rs1]+sign-extend(offset)。

压缩形式: c.swsp rs2, offset; c.sw rs2, offset(rs1)

3	1 25	24 20	19 15	14 12	11 7	6 0
	offset[11:5]	rs2	rs1	010	offset[4:0]	0100011

19.

addi rd, rs1, immediate

$$x[rd] = x[rs1] + sext(immediate)$$

加立即数(Add Immediate). I-type, RV32I and RV64I.

把符号位扩展的立即数加到寄存器 x[rs1]上,结果写入 x[rd]。忽略算术溢出。

压缩形式: c.li rd, imm; c.addi rd, imm; c.addi16sp imm; c.addi4spn rd, imm

31	20	19 15	14 12	2 11 7	6 ()
immediate[11:0]		rs1	000	rd	0010011	

20.

Slti rd, rs1, immediate

$$x[rd] = (x[rs1] <_s sext(immediate))$$

小于立即数则置位(Set if Less Than Immediate). I-type, RV32I and RV64I.

比较 x[rsI]和有符号扩展的 immediate, 如果 x[rsI]更小,向 x[rd]写入 1,否则写入 0。

31	20	19 15	14 12	2 11 7	6 0
im	mediate[11:0]	rs1	010	rd	0010011

sltiu rd, rs1, immediate

 $x[rd] = (x[rs1] <_u sext(immediate))$

无符号小于立即数则置位(Set if Less Than Immediate, Unsigned). I-type, RV32I and RV64I. 比较 x[rs1]和有符号扩展的 immediate, 比较时视为无符号数。如果 x[rs1]更小,向 x[rd]写入1, 否则写入0。

31 20	19 15	14 12	2 11 7	6 0)_
immediate[11:0]	rs1	011	rd	0010011	1

22.

XOri rd, rs1, immediate

$$x[rd] = x[rs1] ^ sext(immediate)$$

立即数异或(Exclusive-OR Immediate). I-type, RV32I and RV64I. x[rs1]和有符号扩展的 immediate 按位异或,结果写入 x[rd]。

压缩形式: c.xor rd, rs2

31	20	19 15	14 12	2 11 7	6 0
in	nmediate[11:0]	rs1	100	rd	0010011

23.

Ori rd, rs1, immediate

$$x[rd] = x[rs1] | sext(immediate)$$

立即数取或(OR Immediate). R-type, RV32I and RV64I.

把寄存器 x[rs1]和有符号扩展的立即数 immediate 按位取或, 结果写入 x[rd]。

压缩形式: c.or rd, rs2

31	25 24	20	19 15	14 12	11 7	6 0
Immediate[11:	0]	rs2	rs1	110	rd	0010011

24.

andi rd, rs1, immediate

x[rd] = x[rs1] & sext(immediate)

与立即数 (And Immediate). I-type, RV32I and RV64I.

把符号位扩展的立即数和寄存器 x[rsI]上的值进行位与,结果写入 x[rd]。

压缩形式: c.andi rd, imm

31	20	19 15	14 12	2 11 7	6 0
	immediate[11:0]	rs1	111	rd	0010011

25.

slli rd, rs1, shamt

$$x[rd] = x[rs1] \ll shamt$$

立即数逻辑左移(Shift Left Logical Immediate). I-type, RV32I and RV64I.

把寄存器 x[rsI] 左移 shamt 位, 空出的位置填入 0, 结果写入 x[rd]。对于 RV32I, 仅当 shamt [5]=0 时, 指令才是有效的。

压缩形式: c.slli rd, shamt

31	26 25	20 19	15 14	12 11	7 6	0
000000	sham	rs	1 001	l rd	00100)11

srli rd, rs1, shamt

$$x[rd] = (x[rs1] \gg_u shamt)$$

立即数逻辑右移(Shift Right Logical Immediate). I-type, RV32I and RV64I.

把寄存器 x[rs1] 右移 shamt 位, 空出的位置填入 0, 结果写入 x[rd]。对于 RV32I,仅当 shamt [5]=0 时,指令才是有效的。

压缩形式: c.srli rd, shamt

31	26	25 20		14 12	11 7	6 0
	000000	shamt	rs1	101	rd	0010011

27.

srai rd, rs1, shamt

$$x[rd] = (x[rs1] \gg_s shamt)$$

立即数算术右移(Shift Right Arithmetic Immediate). I-type, RV32I and RV64I.

把寄存器 x[rsI] 右移 shamt 位,空位用 x[rsI] 的最高位填充,结果写入 x[rd]。对于 RV32I,仅当 shamt[5]=0 时指令有效。

压缩形式: c.srai rd, shamt

31	26	5 2 5 20	19 15	14 12	11 7	6
	010000	shamt	rs1	101	rd	0010011

28.

add rd, rs1, rs2

$$x[rd] = x[rs1] + x[rs2]$$

ли (Add). R-type, RV32I and RV64I.

把寄存器 x[rs2]加到寄存器 x[rs1]上,结果写入 x[rd]。忽略算术溢出。

压缩形式: c.add rd, rs2; c.mv rd, rs2

J1 2.	5 24 20 .	19 15.	14 12	. 11 /	0 0
0000000	rs2	rs1	000	Rd	0110011

29.

sub rd, rs1, rs2

x[rd] = x[rs1] - x[rs2]

减(Substract). R-type, RV32I and RV64I.

x[rs1]减去 x[rs2], 结果写入 x[rd]。忽略算术溢出。

压缩形式: c.sub rd, rs2

31	25	24 20	19 15	14 12	2 11 7	6 0
	0100000	rs2	rs1	000	rd	0110011

30.

S|| rd, rs1, rs2

$$x[rd] = x[rs1] \ll x[rs2]$$

逻辑左移(Shift Left Logical). R-type, RV32I and RV64I.

把寄存器 $\mathbf{x}[rs1]$ 左移 $\mathbf{x}[rs2]$ 位,空出的位置填入 0,结果写入 $\mathbf{x}[rd]$ 。 $\mathbf{x}[rs2]$ 的低 5 位(如果是 RV64I 则是低 6 位)代表移动位数,其高位则被忽略。

31	25 24	20 19 15	14 12	2 11 7	6 0	_
0000000	rs2	rs1	001	rd	0110011	1

SIt rd, rs1, rs2

 $x[rd] = (x[rs1] <_s x[rs2])$

小于则置位(Set if Less Than). R-type, RV32I and RV64I.

比较 x[rs1]和 x[rs2]中的数,如果 x[rs1]更小,向 x[rd]写入 1,否则写入 0。

31	. 25	24 20	19 15	14 12	11 7	6 0
	0000000	rs2	rs1	010	rd	0110011

32.

sltu rd, rs1, rs2

$$x[rd] = (x[rs1] <_u x[rs2])$$

无符号小于则置位(Set if Less Than, Unsigned). R-type, RV32I and RV64I.

比较 x[rs1]和 x[rs2],比较时视为无符号数。如果 x[rs1]更小,向 x[rd]写入 1,否则写入 0。

31	25	24 20	19 15	14 12	2 11 7	6	0
	0000000	rs2	rs1	011	rd	0110011	

33.

XOT rd, rs1, rs2

 $x[rd] = x[rs1] ^ x[rs2]$

异或(Exclusive-OR). R-type, RV32I and RV64I.

x[rs1]和 x[rs2]按位异或,结果写入 x[rd]。

压缩形式: c.xor rd, rs2

31	. 25	24 20	19 15	14 12	2 11 7	6 0
	0000000	rs2	rs1	100	rd	0110011

34.

srl rd, rs1, rs2

 $x[rd] = (x[rs1] \gg_u x[rs2])$

逻辑右移(Shift Right Logical). R-type, RV32I and RV64I.

把寄存器 x[rs1] 右移 x[rs2] 位,空出的位置填入 0,结果写入 x[rd]。x[rs2] 的低 5 位(如果是 RV64I 则是低 6 位)代表移动位数,其高位则被忽略。

31	25	5 24 20	19 15	14 12	11 7	6 0
	0000000	rs2	rs1	101	rd	0110011

35.

Sra rd, rs1, rs2

 $x[rd] = (x[rs1] \gg_s x[rs2])$

算术右移(Shift Right Arithmetic). R-type, RV32I and RV64I.

把寄存器 x[rs1]右移 x[rs2]位,空位用 x[rs1]的最高位填充,结果写入 x[rd]。x[rs2]的低 5 位(如果是 RV64I 则是低 6 位)为移动位数,高位则被忽略。

3	1 25	24 20	19 15	14 12	11 7	6 ()
	0100000	rs2	rs1	101	rd	0110011	7

Or rd, rs1, rs2

 $x[rd] = x[rs1] \mid x[rs2]$

取或(OR). R-type, RV32I and RV64I.

把寄存器 x[rs1]和寄存器 x[rs2]按位取或,结果写入 x[rd]。

压缩形式: c.or rd, rs2

3	31 2	25 24 2	0 19 15	14 12	2 11 7	6)
	0000000	rs2	rs1	110	rd	0110011	7

37.

and rd, rs1, rs2

x[rd] = x[rs1] & x[rs2]

与 (And). R-type, RV32I and RV64I.

将寄存器 x[rs1]和寄存器 x[rs2]位与的结果写入 x[rd]。

压缩形式: c.and rd, rs2

31	25	5 24 20	19 17	14 12	2 11 7	6 0
	0000000	rs2	rs1	111	rd	0110011