

9. (1) 指令跳转范围为 $\pm 1\text{MB}$ 的地址范围.

(2) $\pm 4\text{KB}$

(3) 可以 `lui rsl, imm32`

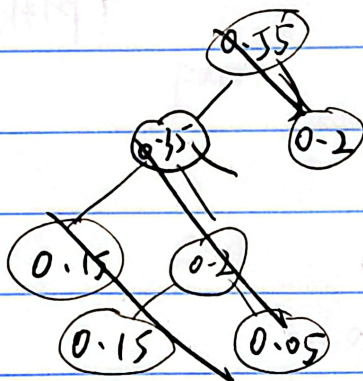
`jalu rd, rsl, imm32`

10. 条件: 仅适用于整数指令集, 仅适用于常见指令, 操作数必须在特定范围内, 需要对齐.

在RVC指令集中, 除了少数例外情况, 各类型的指令都可以使用完整的32个通用寄存器. 例外如: `Move.w` 只能用 `x8, x9, x10, x11`, 要求4字节对齐的指令必须是偶数寄存器.

18.

a_i	p_i
a	0.25
b	0.2
c	0.2
d	0.15
e	0.15
f	0.05



平均长度:

$$\bar{L} = \sum_{i=1}^6 p_i l_i$$

$$= 0.5 + 0.4 + 0.6 + 0.45$$

$$+ 0.45 + 0.15$$

$$= 0.9 + 0.9 + 0.6 + 0.15$$

$$= 1.8 + 0.6 + 0.15$$

$$= 2.4 + 0.15 = 2.55$$

a: 0010

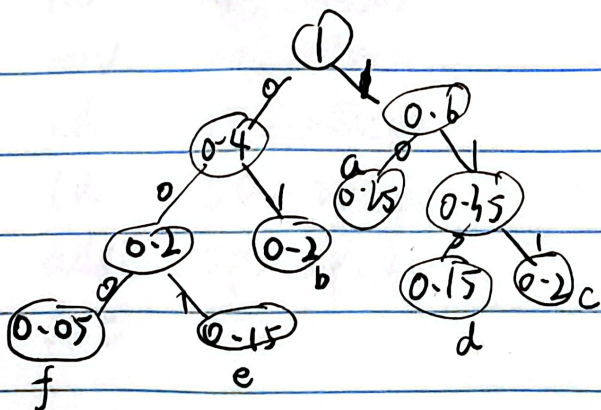
b: 01

c: 111

d: 110

e: 001

f: 000



$$R = 1 - \frac{\sum_{i=1}^6 p_i \log_2 p_i}{2.55} = 0.03296$$

19.

11) 在函数不断地被一层层调用中, 返回地址和局部变量等数据会被压入栈中, 越积越多, 直到超过了栈的容量, 导致栈溢出, 栈溢出可能会导致程序崩溃, 因为栈中数据覆盖了其他重要的数据。

12) 1. 减少函数嵌套的层数 2. 增大栈的容量 3. 使用动态内存分配 4. 使用栈溢出保护机制 5. 优化代码。

20.

ra(F₁)
so(F₁)
para(F₁)
~~Temp~~ para(F₁)
~~Temp so(F₁)~~
ra(F₂)
so(F₂)
para1(F₂)
para2(F₂)
t0(F₂)
t1(F₂)
s1(F₂)