

9. 1) jal 指令可以跳转的地址空间范围为 $\pm 1\text{MB}$

2) 条件分支指令可以跳转的地址空间范围为: $\pm 4\text{KB}$

3) 可以, 先用 `lui` 将 `imm` 左移 12 位作为 `offset[31:12]` 存入 `jalr` 指令将要调用的 `rs1` 寄存器内, `jalr` 指令的立即数则为 `offset[11:0]`. 最后 `jalr` 指令使 $PC = rs1 + imm$ 即将实现 32 位地址跳转。

10. 条件: 1. 立即数或地址偏移量较小

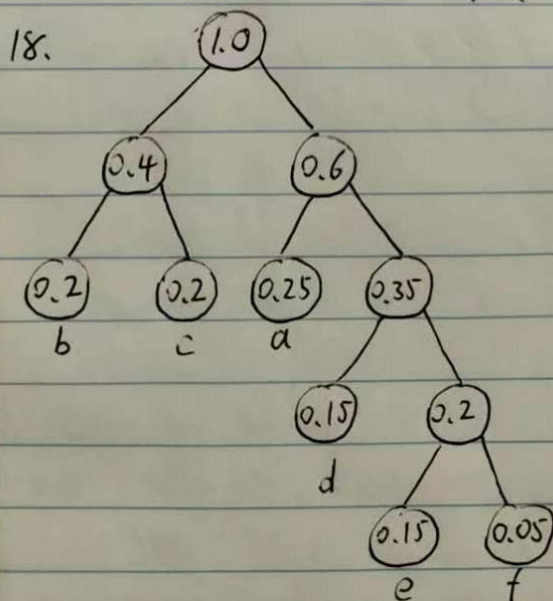
2. 其中一个寄存器为零寄存器(`x0`)或 `x1` 或 `x2`.

3. 目标寄存器与第一个寄存器相同

4. 最常见情况下使用 8 个寄存器

`C1W`, `CL`, `CS`, `CR` 只能使用 `x8~x15` 这 8 个寄存器

`CR`, `CL`, `CSS`, `CJ` 可以使用任何 32 个中的寄存器



$$\sum_{i=1}^6 p_i l_i = 2.55$$

$$H = -\sum_{i=1}^6 p_i \log_2 p_i \approx 2.466$$

$$R = 1 - \frac{H}{\sum_{i=1}^6 p_i l_i} \approx 0.033$$

19. 1) 每次递归就相当于调用一个函数, 每次调用都会将局部数据 (在函数内部定义的变量、参数等, 如 `sp`, `ra`, `so` 等) 放入栈中。这些数据占用的内存直到整个递归结束才能释放, 若递归次数过多, 局部数据也过多, 会使用大量栈内存导致栈溢出。

- 2) ①减少栈空间的需求, 不要定义与用内存过多的变量, 如很大的数组
②函数参数不要传递: 大型结构联合对象尽量传递指针
③减少函数调用次数, 优化递归函数。

20、

ra(F1)

so(F1)

to(F1)

ra(F2)

so(F2)

to(F2)

sl(F2)

tl(F2)