9. 1) jal指令包括20位的J-type，该指令相较当前PC可以跳转的地址空间范围为$\pm 2^{20}$个相对地址，即$\pm 2^{21}$个字节。

2) 条件分支指令包含12位的有符号立即数编码，该指令相较当前PC可以跳转的地址空间范围为$\pm 2^{12}$个相对地址，即$\pm 2^{13}$个字节。

3) 不可以。因为lui指令只能设置一个16位的立即数到寄存器的高16位，而jalr指令只能跳转到寄存器中的相对地址，而不是绝对地址。要实现任意32位绝对地址的操作，需要在程序中设置目标地址的相对地址，然后使用jal指令或者其他跳转指令来完成跳转。

10. 压缩条件：

① 指令只能使用16位的寻址范围。

② 指令不必须使用特定的寄存器，例如x8-x15寄存器

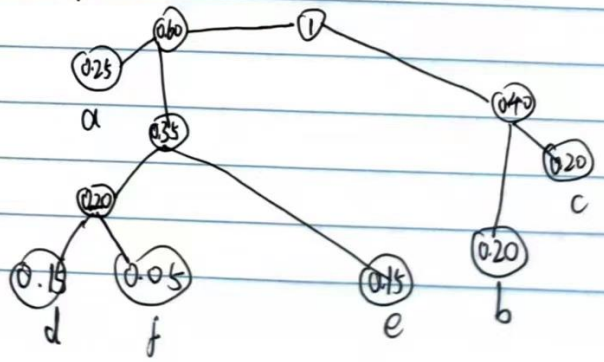
③ 指令不能使用任何立即数或扩展操作，只能使用寄存器之间的操作

④ 指令必须满足压缩指令的格式要求。

并非所有指令都可以使用完整的32个通用整型寄存器。例如，立即数指令只能使用16个寄存器，而分支指令只能使用8个寄存器。

18. 将操作码按出现概率从小到大排列得：f、d、e、b、c、a

画出霍夫曼树：



则编码可为：

a：00

b：10

c：11

d：010

e：011

f：0101

平均长度：$\bar{L}=2\times0.25+4\times(0.15+0.05)$

$+2\times(0.20+0.20)+3\times0.15$

$=2.55$

信息冗余量：$R=1-\frac{-\sum_{i=1}^{n}P_i\log_2 P_i}{\log_2 6}$

≈0.046

19. 1）栈溢出原理：

每当一个函数被调用时，都会在栈中为其开辟一段空间来保存这些信息。当函数执行完毕后，就会把信息弹出栈。如果函数递归调用时没有正确的结束条件，导致递归深度无限增加，最后函数嵌套调用层数过多就会发生溢出。

2）缓解和避免的方法：

① 优化算法，减少函数嵌套调用层数。

② 增加栈的大小

③ 使用尾递归优化技术。

④ 避免使用过多局部变量、数组、大型数据结构和对象。

⑤ 使用异常处理机制捕获栈溢出异常。

20. r_a (F1)

t_0 (F1)

t_1 (F2)

S_0 (F2)

S_1 (F2)