

9.

(1) 20位有符号数可以覆盖 2^{20} 个字节，由于 RISCV64 中一条指令占 4 个字节，所以可以覆盖从当前指令开始的向上 2^7 条和向下 $2^7 - 1$ 条指令的地址空间

会被扩展为 21 位有符号数吗？

(2) 与上题同理，除去一位符号位，可覆盖从当前指令开始的向上 2^9 条和向下 $2^9 - 1$ 条地址空间

(3) 可以。一条 lui 指令可以在寄存器中存入 32 位数的高 20 位，而 jalr 恰好可以取出该数后再加上 12 位组成 32 位地址并跳转。

最低有效位置吗？四字节对齐？

10.

(1) 能够被压缩为 16 位 RVC 指令的条件是

① 立即数或地址偏移量很小

② 使用的寄存器中有 x_0 或 x_1 或 x_2 寄存器

③ 目标寄存器和第一个源寄存器相同

④ 使用的寄存器是常用的 8 个 ($x_8 - x_{15}$)。

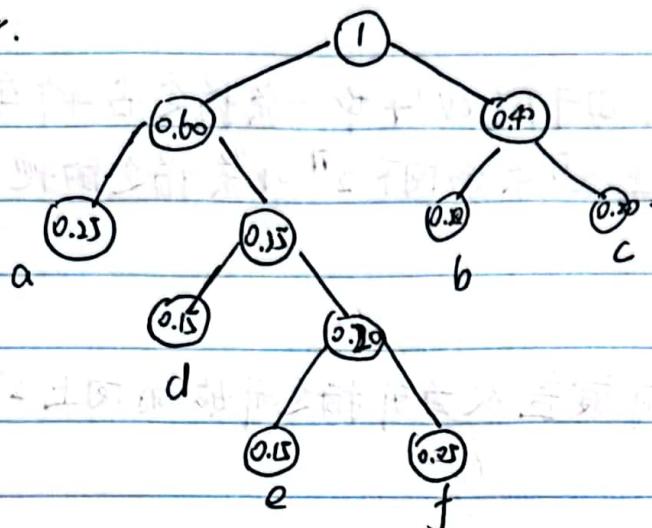
以上条件满足才即可

(2) RVC 指令不同类型的指令都会使用 32 个通用寄存器中的一部分，但是没有某个类型的指令可以使用全部寄存器



扫描全能王 创建

18.



d 010

e 0110

f 0111

$$\text{平均码长} = \sum_{i=1}^6 p_i l_i = 2.55$$

$$\text{最短平均码长 } H = - \sum_{i=1}^6 p_i \log_2 p_i$$

$$\text{信息冗余度} = 1 - \frac{H}{\text{平均码长}}$$

$$= 3.3\%$$



扫描全能王 创建

19.

(1) 递归函数每次都会调用自己作为子函数，而调用函数时，就会执行函数开始时的初始化工作，会将栈指针sp向下移动一段距离（如32.48），开辟一块栈空间供本次函数运算使用。

一般情况下，在函数结束时才会释放栈的这部分空间。但是递归函数会在结束前调用子函数，必须在递归结束后，才可以将每次调用开辟的空间依次释放。这样做也是为了父函数的数据不被覆盖，防止出现找不到返回地址无法返回等问题

所以递归函数死循环或调用次数太多，只开辟栈空间不释放，就有可能将栈的空间全部用完，出现栈溢出的问题

② 方法

① 改变函数实现方法，如不使用递归改用循环结构。

② 减少每次开辟的栈空间，只开辟必要的空间，可以增加固定栈总空间情况下可以调用的次数

③ 减少代码中使用栈的变量，更多使用寄存器存储变量，可以进一步压缩需要开辟的栈空间

④ 将一些变量改为常量，也可以压缩需要的栈空间。

⑤ 在不确定是否会溢出的时候，~~R2L-V~~ R2L-V 可以用栈指针边界检查，（用一条指令检查栈指针位置）或栈空间限制（设置一个比内存小的最大栈大小）来防止栈指针越界，在栈溢出前结束程序的运行。



扫描全能王 创建

5.

首次调用 F3 前: t₀(F1) 重置函数, 将清空所有圆周上储存的圆心与半径
(x_0, y_0, r) s₀(F1) 收集圆周中当前存在的圆心, 并将圆心的坐标存于
t₀(F1) (参数). 用此圆心列表下一次圆心插入操作
之后圆周上多了一个圆心 s₁(F2) 将圆周上所有圆心的半径进行归一化, 并将归一
化后的圆心插入 t₀(F2) 下, 对原有圆心进行移除, 并将新圆心存于 t₁
而过之前, 重复 t₀(F2) (参数) 归一化并移除圆心, 并将新圆心插入
t₁(F2) (参数). 通过暴力循环方式不断更新圆周上的圆心
直到 t₁(F2) (参数) (可能有的计算结果, 因为 F2 会返回一个 int 值) 为 0
即圆周上没有圆心了, 采用将圆周上圆心数减去 1

结束 (4)

所欲而欲用此函数无可避免, 故而此方法速度之慢
实所欲不可得, 但此方法以智取力, 而圆周内之每一点之处理
对外圆周上点不可以影响
而遇此问题, 番多函数都再莫用斯道, 番多函数用其冲出此点后
向圆周间断不连贯
而此方法由单向而通以下, 且带去的是支点一端
故此用此方法前欲用此 V-Sort 语句, 新旧内存交换是必须的
最怕小兵阿山木里身土便身而宝珠失 (是此地村盖地主的第二用)
广运用民外来过府为盗林生, 邪性怪盗林生相来 (小木村大



扫描全能王 创建