

9. 解: (1) 范围是 $-2^{20}+2$ 到 $2^{20}-2$

(2) 范围是 $-2^{12}+2$ 到 $2^{12}-2$

(3) 可写 $|U_i \text{ rsl}, \text{imm}[31,12]$
jalr rd imm[11,0](rsl)

10. 解: 条件: ① 立即数或地址偏移量较小

② 其中 1 个寄存器为 $x0, sp, ra, a0-a5, s0, s1$ 之一

③ 目标寄存器与源寄存器相同

不能使用所有寄存器, 有 m 只能使用 $x8-x15$ 8 个寄存器

18. 解: 根据霍夫曼编码

因此编码为

a 11

b 00

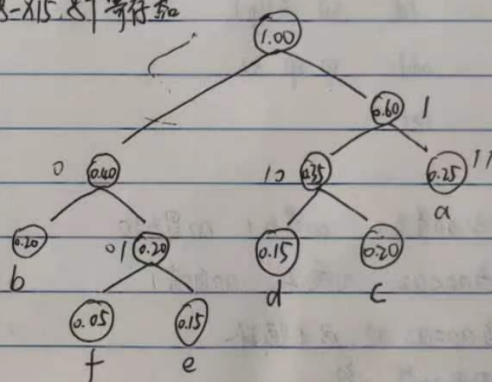
c 101

d 100

e 011

f 010 故平均长度为 $\sum_{i=1}^6 P_i l_i = 2.55$

信息冗余度为 $R = 1 - \frac{\sum_{i=1}^6 P_i \log_2 P_i}{2.55} = 0.033$



19. 解: (1) 在计算机中, 函数调用是通过栈这种数据结构实现的。每当程序执行进入一个函数调用, 栈就会加一层栈帧, 每当函数返回, 栈就减一层栈帧。由于栈的大小不是无限的, 所以, 递归调用次数越多, 就会导致栈溢出。函数的参数是通过 stack 栈来传递的, 在调用中会占用线程的栈资源。递归调用在到达最后的结束点后, 函数才能顺栈清栈, 如果递归调用层数过多, 就可能造成占用的栈资源超过线程的最大值, 从而导致栈溢出。

- (2) 解决方法: ① 减少栈空间的需求, 不要定义占用内存较多的 auto 变量, 应将其修改为指针变量, 从堆空间分配内存
- ② 函数参数中不要传递大规模结构/联合/对象, 应使用引用或指针
- ③ 减少函数调用层次, 慎用递归函数

20. 解:

$ra(F1)$

$to(F1)$

$so(F1)$

$ra(F2)$

$to(F2)$

$tl(F2)$

$so(F2)$

$sl(F2)$