

9.回答以下问题:

1)jal指令包含20位的有符号立即数编码 (J-type) , 该指令相较当前PC可以跳转的地址空间范围为多少?

PC-524288到PC+524287

但是考虑到指令是4位一条, 故实际范围为PC-131,072*4到PC+131,071*4

2)条件分支指令 (如bne) 包含12位的有符号立即数编码 (B-

type) , 这类指令相较当前PC可以跳转的地址空间范围为多少?

PC-2048到PC+2047

但是考虑到指令是4位一条, 故实际范围为PC-512*4到PC+511*4

3)是否可以使用一条lui指令和一条jalr指令的组合完成任意32位绝对地址的跳转操作?

可以

10.调查

RVC压缩指令集的编码, 说明一条常用的32位指令能够被压缩为16位RVC指令的条件是什么? RVC中各类型的指令是否都可以使用完整的32个通用整型寄存器?

条件 (至少满足一个) :

? 立即数或者地址偏移量较小时

? 其中一个寄存器是零寄存器 (x0) 、ABI链接寄存器 (x1) 或者ABI栈寄存器 (x2)

? 目标寄存器和第一个源寄存器相同

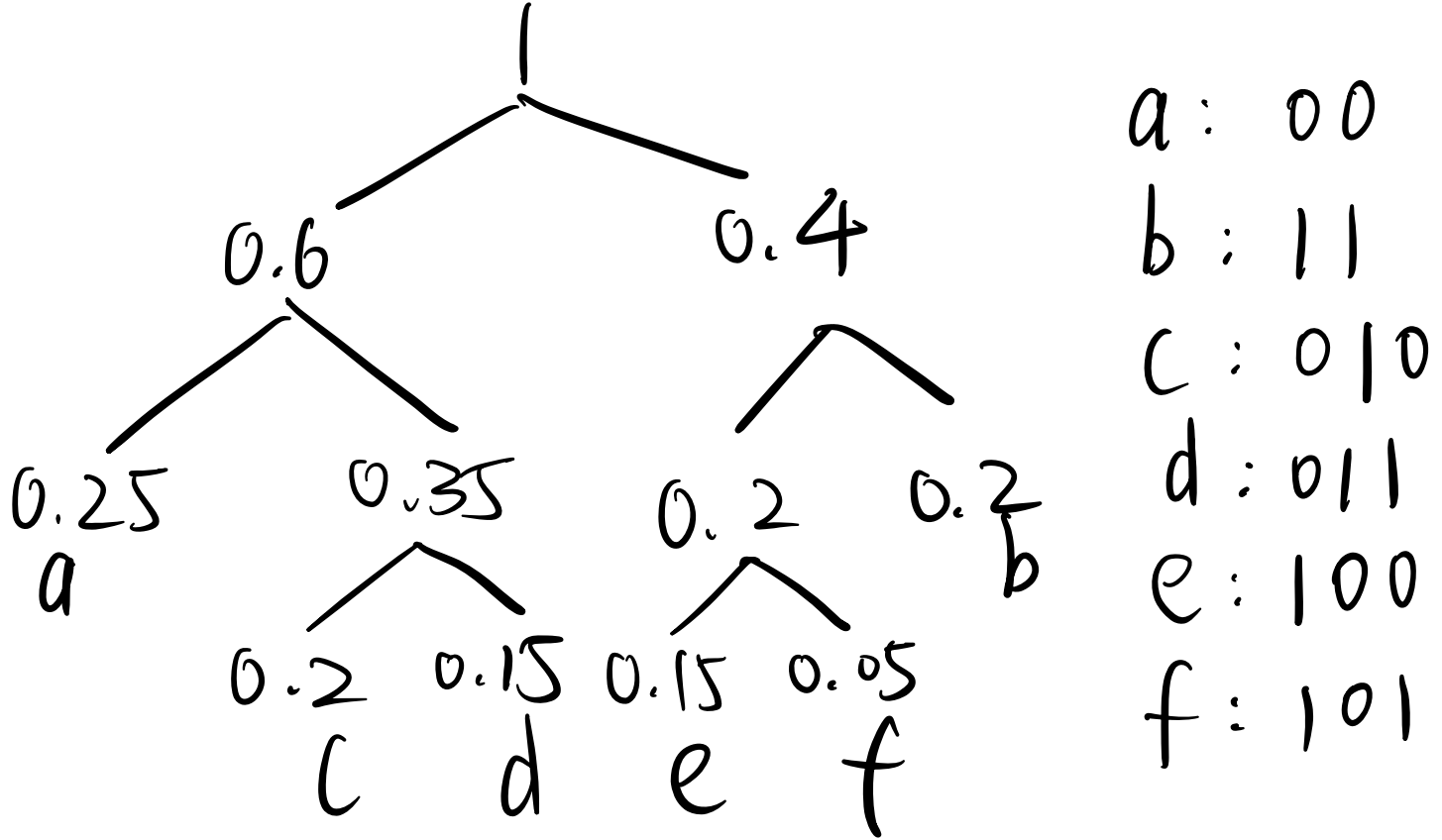
? 最常见情况下使用了8个寄存器

18. 有一组操作码, 它们的出现几率如下表所示。

a_i	p_i
a	0.25
b	0.20

c	0.20
d	0.15
e	0.15
f	0.05

请按照霍夫曼编码对这组操作码进行编码, 计算操作码的平均长度和信息冗余度。



$$H = 2 \times 0.45 + 3 \times 0.55 = 2.55$$

$$R = 1 - \frac{2.55}{3} = 0.15 = 15\%$$

19.回答以下问题:

1)当函数嵌套调用层数过多(例如递归陷入死循环时),可能会造成栈溢出,请简述其原理。

上一层循环所需要调用的栈处于封锁状态,此时往下一层调用,使得每层循环都产生额外的对栈的占用,一旦层数过多则会由于栈的层数是有限的,导致栈溢出。

2)有什么办法可以缓解或避免特定情况下的栈溢出问题?

增大栈的大小: 在程序运行之前, 可以通过调整系统或编译器的参数来增大栈的大小, 以便在函数调用嵌套较深时有足够的空间。

减少局部变量和函数参数的数量: 栈溢出的一个常见原因是在函数中创建过多的局部变量和参数, 可以通过减少这些变量的数量来避免栈溢出。

将递归函数改为迭代函数: 递归函数调用本身就是一种栈操作, 而且它还需要在栈上分配大量的内存空间。将递归函数改写为迭代函数可以避免这种情况。

使用动态内存分配: 通过使用动态内存分配可以避免栈空间的限制, 但是需要手动释放分配的内存, 否则可能会导致内存泄漏的问题。

尽可能避免使用过多的大型对象: 过多的大型对象也会导致栈空间的限制, 可以考虑使用动态内存分配或者全局变量来避免这种情况。

20. 假设有三个函数: F1、F2和F3。其中F1包含1个输入参数, 计算过程使用寄存器t0

和s0; F2包含2个输入参数, 计算过程使用寄存器t0-t1及s0-s1, 返回一个int值。F1

执行过程中会调用F2, F2执行过程中会调用F3。下表模拟了F1执行过程中栈的内容,

其中第一行为F1函数被首次调用时sp寄存器指向的位置。请在表中填入当F2函数首次

调用F3前栈内保存的可能内容, 并在每行的括号内标注该值是被哪个函数所保存的。

第一行的内容已经给出。(可根据需要增删行数)

ra (F1)	
a0 (F1)	输入
t0 (F1)	
s1 (F1)	
ra (F2)	开始调用F2
a0 (F2)	输入1
a1 (F2)	输入2
t0 (F2)	
t1 (F2)	
s0 (F2)	
s1 (F2)	
ra (F3)	开始调用F3