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？
   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？
   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？

1) ±1MB

2) ±4KB

3) 可以 （JALR指令被定义为可使用双指令序列来跳转到32位绝对地址空间内的任何地方，LUI指令将目标地址高20位加载到rs1中，然后JALR指令可以加上低12位）

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？

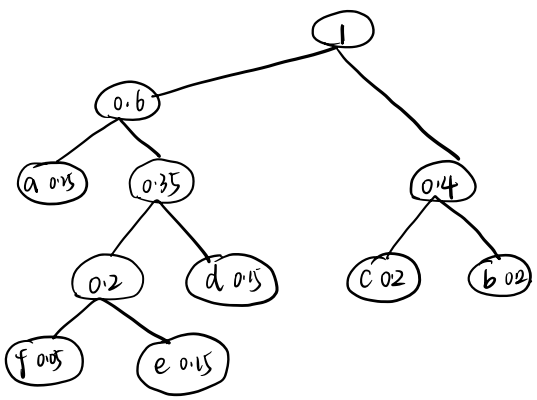
条件：① 立即数或地址偏移量较小时
② 其中一个寄存器为零寄存器（X0），ABI链接寄存器（X1）或ABI栈寄存器（X2）
③ 目标寄存器和第一个源寄存器相同
④ 最常见情况下使用8个寄存器

不能，（CR、CI和CSS指令可使用32个通用整型寄存器，而CIW、CL、CS和CB被限制只能用8个）

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。



a_i	l_i
a	2
b	2
c	2
d	3
e	4
f	4

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

$H = -\sum_{i=1}^{6} p_i \cdot \log_2 p_i = 2.47$

则 $R = 1 - \frac{2.47}{2.55} = 0.031$

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。
    2) 有什么办法可以缓解或避免特定情况下的栈溢出问题？

1) 每当程序执行进入一个函数调用，栈就会多加一层栈帧，向函数返回时，栈会减一层栈帧。由于栈大小有限，当嵌套次数过多时，可能导致占用的栈资源

超过线程最大值，从而导致栈溢出。

2) ① 控制递归深度，例如全用动态规划代替递归函数。
② 修改栈的大小。
③ 可采用尾递归优化or采用循环替代递归。

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)
t0(F1)
s0(F1)
a0(F2)
a1(F2)
t0(F2)
t1(F2)
s0(F2)
s1(F2)
ra(F2)

先保存预设参数

寄存器	ABI名称	说明	保存者
x0	zero	硬件上恒为0	N/A
x1	ra	返回地址	调用者
x2	sp	栈指针	被调用者
x3	gp	全局指针	N/A
x4	tp	线程指针	N/A
x5	t0	临时连接寄存器	调用者
x6-7	t1-2	临时寄存器	调用者
x8	s0/fp	保留寄存器/帧指针	被调用者
x9	s1	保留寄存器	被调用者
x10-11	a0-1	函数参数/返回值	调用者
x12-17	a2-7	函数参数	调用者
x18-27	s2-11	保留寄存器	被调用者
x28-31	t3-6	临时寄存器	调用者
f0-7	ft0-7	浮点临时寄存器	调用者
f8-9	fs0-1	浮点保留寄存器	被调用者
f10-11	fa0-1	浮点函数参数/返回值	调用者
f12-17	fa2-7	浮点函数参数	调用者
f18-27	fs2-11	浮点保留寄存器	被调用者
f28-31	ft8-11	浮点临时寄存器	调用者

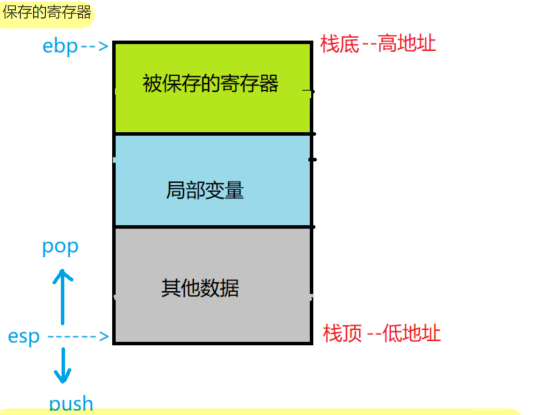
Attention

函数的栈帧

程序在每一次调用函数的时候就会在栈区创建一块空间，这块空间就被称为该函数的栈帧

这块空间中一般包括了下面一些信息：

1. 函数的返回地址和参数
2. 临时变量：包括函数的非静态局部变量以及编译器生成的其他临时变量
3. 保存的寄存器



当运行中的程序调用另一个函数时，就要进入一个新的栈帧，原来函数的栈帧称为调用者的帧，新的栈帧称为当前帧。 被调用的函数运行结束后当前帧全部回收，回到调用者的帧。

例如：当函数A调用函数B的时候，会把返回地址压入栈中，我们把返回地址当做A函数栈帧的一部分，因为它存放的是与A相关的状态

> 注意： esp 寄存器一直指向的是当前栈帧的栈顶（ esp 保存栈顶的地址）。

当函数A调用函数B: