

sw t1, 0(a2)

addi a2, t1, 2

add a2, a2, t0

sw t1, 0(a2)

16. swap: addi t2, t0, 0

此时 t0 中的指针指向原来的 \*b,

addi t0, t1, 0

t1 中的指针指向原来的 \*a;

addi t1, t2, 0

此时仍称 t0 为 a, t1 为 b, 则完成了

17. 计算  $2^{30}$ , 结果存在 a1 中

3.28

9. (1) 以两字节的倍数进行编码, 可跳转的范围是  $\pm 1 \text{ MiB}$

2)  $\pm 4 \text{ KiB}$

3) 可以, 先用 lui 将高 20 位加载到 ra 中, 再用 jalr 将低 12 位加载到 ra 中

lui ra, imm20; jalr ra, imm12(ra)

10. (1) 将 32 位指令压缩为 16 位的条件: (满足其一即可)

① immediate 和 address offset 都很小

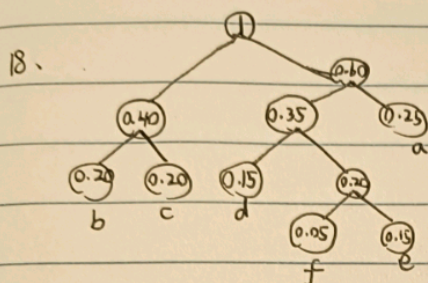
② 一个 register 是 X0 或 ABI link register (X1) 或 ABI stack pointer (X2)

③ 目标寄存器和第一个源寄存器相同

④ the registers used are the 8 most popular ones

(2) 不能。

CR, CI, CSS 格式的压缩指令可以使用所有 32 寄存器;  
但 CIW, CL, CS, CA, CB 只能使用 x8 ~ x15 这 8 个 32 寄存器。



$$\bar{L} = 0.25 \times 2 + 0.2 \times 2 \times 2 + 0.15 \times 3 + (0.15 + 0.05) \times 4$$

$$= 2.55$$

$$H = -\sum_{i=1}^6 P_i \cdot \log_2 P_i = 2.466$$

$$\therefore R = 1 - \frac{H}{\bar{L}} = 0.033 = 3.3\%$$

19. 1) 每次新调用一个函数, 就会开一个临时的栈, 而原来的函数由于没有结束, 原来的栈空间也会保留, 当递归的层数过深, 就会同时存在过多的栈, 导致溢出

2) 解决方法: ① 在函数中不要定义过多的变量, 减小函数对栈空间的需求

② 有时可以考虑用其他算法代替递归算法

③ 函数的参数传递时用指针代替大型数据类型

20.

ra (F1)

to (F1)

~~so (F1)~~

ra (F2)

to (F2)

tl (F2)

so (F2)

sl (F2)

ra (F3)