

3/28

9. 1). $\pm 1 \text{ MiB}$ (2×2^{20})

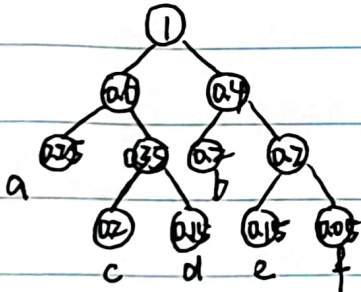
2). $\pm 4 \text{ KiB}$ (2×2^{12})

3). 可以, LUI 指令先将目标地址的高 20 位加载到 rs1, 然后利用 JALR 指令, 将低 12 位地址加上去。

10. 条件: 立即数或地址偏移量较小; 或其中一个寄存器是 x0, x1, x2 之一; 或目标寄存器和第一个源寄存器相同; 或使用寄存器是最常用的 8 个寄存器。

是否: CR, CI, CSS 类型的指令可以使用 32 个通用整型寄存器。但 CLW, CL, CS, CB 只能使用 x8-x15 这 8 个。

18.



| a_i | p_i | l_i | $C(a_i)$ |
|-------|-------|-------|----------|
| a | 0.25 | 3 | 100 00 |
| b | 0.2 | 3 | 110 10 |
| c | 0.2 | 4 | 010 |
| d | 0.15 | 4 | 011 |
| e | 0.15 | 4 | 110 |
| f | 0.05 | 4 | 111 |

平均长度 $\sum_{i=1}^6 p_i l_i = 2.55$

信息冗余度: $R = 1 - \frac{-\sum_{i=1}^6 p_i \log p_i}{2.55} = 3.3\%$

19. 1). 每一次函数调用, 都会在调用栈上维护一个独立的栈帧, 存储包括函数的返回地址、参数以及局部变量, 因此调用的层数越多, 所需栈的空间越大, 可能造成栈的溢出。

2). 办法: 减少函数的调用, 如利用迭代算法通过循环



来实现递归；减少局部变量的使用；优化递归算法，采用尾递归算法，递归调用是函数的最后一个操作，因而不需要保存局部变量，直接让被调用的函数返回到调用者的调用者去，减少栈的使用；使用 malloc 和 free 函数动态分配和释放内存；增加栈空间，为栈设置更大的大小。

20.

| | | |
|--------|---|-------|
| ra(F1) | } | F1 函数 |
| ao(F1) | | |
| to(F1) | | |
| so(F1) | | |
| ra(F2) | } | F2 函数 |
| ao(F2) | | |
| a1(F2) | | |
| to(F2) | | |
| t1(F2) | | |
| so(F2) | | |
| s1(F2) | | |
| ra(F3) | | |

