2022《计算机系统结构》期末考试参考答案

老师说“量大简单”，但好像做完发现只有量大没有简单呢……小坑不少

一、判断题

1. 程序的顺序执行，是造成时间局部性原因之一：错（是空间）
2. 组相联映射是对全相联和直接映射的一种折中，组间采用全相联映射，组内采用直接映像：错，写反了，组间直接映像、组内全相联。
3. 采用多种寻址方式可以减少程序的指令条数，但可能增加处理器实现的复杂度以及指令的CPI：对
4. 流水线属于时间重叠/并行技术：对
5. 数据总线宽度对于高级程序员来说是透明的：对
6. 某型号处理器增加几条媒体处理指令后，不会影响其软件的向后兼容性：对，向后兼容是老的程序能在新的处理器上跑，只增加指令的话确实是可以的。
7. 在顺序发射、顺序完成的指令单流水线中，不存在WAW（写后写）和WAR（读后写）冲突：对
8. 在指令字长确定的情况下，指令系统设计需综合考虑寄存器个数、寻址方式、立即数长度等需求：对
9. 动态多功能流水线通过不同连接方式允许在同一时刻执行不同功能：对
10. 减小CPI是RISC架构的精华，而只有 `Load` 和 `Store` 指令可以访存则是RISC架构的根本：不确定（“只有 `Load` 和 `Store` 指令可以访存”这句话虽然正确，但可能并不能作为RISC架构的根本来看待
11. 从系统结构角度看，增加流水线级数可以提高处理器频率：对
12. 增加Cache组相联度将会减少Cache的冲突缺失次数，但会增加命中时间：对
13. 多级层次存储系统一定要满足包含关系，即上层缓存中的数据一定可以在其下级存储器中找到：对
14. 采用写回（write-back）的Cache，当发生读缺失时，有时会引起写主存操作：对（因为如果读缺失了，则需要写一个进入Cache，此时如果替换掉了Dirty Cache则需要写主存）
15. DRAM的密度比SRAM的密度大：对
16. 在Cache-主存存储层次中，Cache块越大，则强制缺失次数 （compulsory miss）越少：对，但事实上Cache块过度大会导致容量缺失率增加，所以缺失率先降后增
17. 为了支持精确中断，Tomasulo算法需要ROB（Reorder Buffer）支持：对
18. 平均每条指令的执行周期数与程序无关：错
19. Victim Cache是位于CPU和Cache间的又一级Cache：错（它在Cache和下一级存储器调数据的通路之间，用处是存储被替换出去的块，即Victim）
20. 通过寄存器重命名，可以消除WAW和WAR相关：对
21. 关键字优先或早启动（early restart）技术可减少Cache缺失率：错，这两个降低的是Cache缺失的损失（或称代价）
22. 较小而简单的Cache有利于减少命中时间，但会提高缺失率：对
23. INTEL安腾处理器采用的是VLIW架构：错
24. 采用写通过（write-through）策略，需要使用写缓冲器（write buffer）来提高性能，而采用写回策略的Cache则没有必要使用写缓冲器：错，写回也有必要使用（减小CPU停顿）
25. 乱序执行的处理器采用多路组相联Cache时，为了减少功耗， 一般采用Tag比较和数据访问串行的方式，并通过指令乱序执行来弥补由于存储访问延迟造成的性能下降问题：对
26. 伪相联（Pseudo-Associativity）Cache由于变化的命中时间对CPU流水线影响较大，一般适合离CPU远的Cache：对，其中“离CPU远的Cache”例如二级Cache等等
27. 向量指令间存在针对同一向量寄存器的RAR（read after read）冲突。据说是对的

28. 半性能向量长度n1/2等于向量寄存器长度的1/2：错，是向量处理机的性能为其最大性能的一半时所需的向量长度。
29. 通过编译将循环展开成4个循环体时， 一般可减少6条指令（测试和分支）：对
30. 循环展开通过寄存器重命名和指令调度可有效开发指令级并行：对

二、单选题

31. 假定 `Load` 和 `Store` 指令占所有指令的比例为30%，指令Cache的缺失率是2%，数据Cache的缺失率是5%，Cache缺失代价都为100个周期。如果没有任何访存缺失，处理器的CPI为2。如果考虑访存缺失，那么该处理器的CPI是：5.5个周期（$2 + 2\% * 100 + 30\% * 5\% * 100 = 5.5$）
    1. 5.5个周期
    2. 3.5个周期
    3. 6个周期
    4. 3个周期
32. 在执行一段对数组元素求和的循环代码时：以下说法最合适的是：既会体现出数据访问的时间局部性（ `sum` ）和空间局部性（数组），也会体现出代码访问的时间局部性（加法）和空间局部性（指令顺序执行）。
    1. 会体现出数据访问的时间局部性和空间局部性
    2. 会体现出代码访问的时间局部性和空间局部性
    3. 既会体现出数据访问的时间局部性和空间局部性，也会体现出代码访问的时间局部性和空间局部性
    4. 以上说法皆不对

    - 时间局部性：如果程序中的某条指令一旦执行，不久以后该指令可能再次执行；如果某数据被访问过，不久以后该数据可能再次被访问。产生时间局部性的典型原因，是由于在程序中存在着大量的循环操作。
    - 空间局部性：一旦程序访问了某个存储单元，在不久之后，其附近的存储单元也将被访问，即程序在一段时间内所访问的地址，可能集中在一定的范围之内，这是因为指令通常是顺序存放、顺序执行的，数据也一般是以向量、数组、表等形式簇聚存储的。如果一个存储器的位置被引用，那么将来他附近的位置也会被引用。
33. 在发展高性能单处理机过程中，起主导作用的是：时间重叠
    1. 资源重复
    2. 时间重叠/并行
    3. 资源共享
    4. 局部性原理
34. 关于计算机系统的性能指标：执行时间和吞吐率，下列说法错误的是：执行时间短意味着吞吐率高（显然不一定）
    1. 管理员更注重吞吐率
    2. 用户更注重执行时间
    3. 执行时间短意味着吞吐率高
    4. 用户看到的程序执行时间不仅仅是CPU执行该程序的时间
35. 指令系统采用不同寻址方式的目的主要是：缩短指令长度，扩大寻址空间，提高编程灵活性
    1. 实现程序存储与程序控制
    2. 缩短指令长度，扩大寻址空间，提高编程灵活性
    3. 可直接访问外存
    4. 提供扩展操作码的可能并降低指令译码难度
36. 下列关于指令系统说法错误的是：具有相同指令系统的计算机其微架构必然相同（不一定）

1. 指令系统不仅仅是定义了各种指令
2. 具有相同指令系统的计算机其程序可相互兼容
3. 具有相同指令系统的计算机其微架构必然相同
4. 指令系统定义了计算机软硬件接口

37. 关于流水线分类，以下说法错误的是：单功能流水线可分为静态与动态流水线（单功能流水线显然只能是静态的，动态的一定是多功能流水线）

    1. 单功能流水线可分为静态与动态流水线
    2. 流水线可分为单功能与多功能流水线
    3. 流水线可分为线性与非线性流水线
    4. 流水线可分为顺序与乱序流水线

38. 以下哪个指标不是评价流水线性能的常用指标？功耗（此外CPI也可以算常用指标）

    1. 吞吐率
    2. 加速比
    3. 功耗
    4. 效率

39. 各流水段的执行时间并不完全相等的流水线中，最大吞吐率受限于：瓶颈段的执行时长

    1. 排空时间
    2. 瓶颈段的执行时长
    3. 建立时间
    4. 段数

40. 以下关于数据相关的说法错误的是：相关一定会转化为流水线冲突（不一定，甚至同一种相关在某个流水线里会转化为冲突，但放到另一个流水线里又不会）

    1. 相关一定会转化为流水线冲突
    2. 相关是程序固有的属性
    3. 相关分为数据相关、控制相关与名相关
    4. 名相关可分为输出相关与反相关

41. 理论上，以下哪个选项不会影响分支指令性能？分支成功时，分支指令后续指令的处理

    1. 何时算出分支是否成功
    2. 分支成功时，何时算出分支目标地址
    3. 分支成功时，目标指令是否在指令Cache中
    4. 分支成功时，分支指令后续指令的处理

42. 层次存储系统中寄存器容量 << Cache容量 << 主存容量，其主要因素是：不确定1/2，似乎是2

    1. 速度
    2. 成本
    3. 密度
    4. 复杂性

43. 以下哪种结构能够避免取指与访问操作数之间的访存冲突：哈佛结构

    1. 冯诺依曼结构
    2. 哈佛结构（程序和数据完全分开）
    3. 层次存储
    4. 并行存储器

44. 在一个多级存储系统 `M1` 、 `M2` 、…、 `Mn` 中。下面关于存储系统期望达到的目标说法错误的是：整个存储系统的访问时间应该接近于 `Mn` 的访问时间（应该是接近于 `M1` 的时间，因为绝大部分访问都应该直接命中 `M1` ，否则系统也太失败了）

    1. 整个存储系统的容量应该接近于 `Mn` 的容量
    2. 整个存储系统的访问时间应该接近于 `Mn` 的访问时间
    3. 整个存储系统的平均每位价格应该接近于 `Mn` 的平均每位价格

    4. 整个存储系统的访问时间应该接近于 `M1` 的访问时间
45. 采用指令Cache与数据Cache分离的主要目的是：减少指令流水线中Cache的访问冲突

    1. 降低Cache的缺失代价
    2. 提高Cache的命中率
    3. 降低CPU平均访存时间
    4. 减少指令流水线中Cache的访问冲突
46. 下列关于Cache的映像规则中，哪种映像规则的Cache利用率最高？全相联（直接映射最低）

    1. 全相联
    2. 组相联
    3. 直接映射
    4. 段相联
47. 一般情况下，下面哪种并行主存系统的访存效率最高：多体低位交叉存储器

    1. 单体单字存储器
    2. 多体高位交叉存储器
    3. 单体多字存储器
    4. 多体低位交叉存储器
48. 以下说法错误的是：一般说来，用RISC计算机编程比用CISC计算机编程所占程序存储空间更小（RISC对不常用的功能常常需要多个指令组合实现，因此对这些功能效率较低；RISC一般需要更大的内存空间）

    1. 用间址寻址方式可以缩短地址码长度
    2. 用变址寻址方式可以缩短地址码长度
    3. 用寄存器间接寻址方式可以缩短地址码长度
    4. 一般说来，用RISC计算机编程比用CISC计算机编程所占程序存储空间更小
49. RISC计算机 `Load` 和 `Store` 指令平均占总指令数的比例分别是18%和6%，则每执行一条指令，平均访问存储器次数是：1.24（需要访问指令存储）

    1. 1.24
    2. 0.18
    3. 0.06
    4. 0.24
50. 下面哪种说法具有不确定性？增加Cache块大小，会增加缺失代价。（不确定，可能选4/5，倾向于5）

    1. 增加Cache容量，会减小容量缺失，增加命中时间
    2. 增加Cache相联度，会减少冲突缺失，增加命中时间 （假设Cache容量固定）
    3. 增加Cache块大小，会减少强制缺失，增加命中时间 （假设Cache容量固定）
    4. 增加Cache块大小，会增加缺失代价
    5. 增加L1（第一级）Cache相联度，对增加其容量有利
51. 以下哪种技术不能减少Cache缺失代价（miss penalties）？（不确定，应该是4）

    1. 多级Cache
    2. 关键字优先
    3. 写缓冲合并（Merging Write Buffer）
    4. 软件/硬件预取
    5. 让读不命中优先于写
52. 在一个5段的流水线处理机上需经9拍才能完成一个任务，其预约表如下表所示，该预约表对应的原始冲突向量是：禁止集1、5、6、8，冲突向量10110001

功能段	1	2	3	4	5	6	7	8	9
S1	×								×
S2		×	×					×	
S3				×					
S4					×	×			
S5							×		

1. 10110011
2. 10110001
3. 10001101
4. 11001101

53. 如上题所示预约表，通过插入非计算延迟可以实现流水线的无冲突最优调度，此时可以达到的流水线的最大吞吐率是（假设流水线时钟周期为Δt）：可以以3为周期利用预留算法完成安排（S2每条任务要执行3次导致不能以2为周期），此时上表$(S2,8)$和$(S1,9)$都要往后错一位，因此吞吐率为$\frac{n}{(3n+6)\Delta t}$，最大吞吐率为$n\to+\infty$时取到$\frac{1}{3\Delta t}$。

    1. $\frac{1}{2\Delta t}$
    2. $\frac{1}{3\Delta t}$
    3. $\frac{1}{5\Delta t}$
    4. $\frac{2}{7\Delta t}$

54. 一个单处理器采用Cache-主存两层存储结构。指令Cache和数据Cache分离，它们的命中率分别是H_i与H_d。处理机访问两个Cache的时间都是c个时钟周期，一次访问主存储器的延迟是b个时钟周期。在CPU进行的所有访存操作中，访问指令Cache的百分比是F_i。数据Cache采用写回和按写分配（write back&write allocate）策略，当数据Cache访问缺失时，被替换块（dirty位为1）需要写回主存的情况占比是F_d。则该处理器的平均存储器访问时间为：？3，因为F_d的情况下要先访问再写。

    1. $F_i(H_ic+(1-H_i)(b+c))+(1-F_i)(H_dc+(1-H_d)(F_d(b+c)+(1-F_d)(b+c)))$
    2. $F_i(H_ic+(1-H_i)(2b+c))+(1-F_i)(H_dc+(1-H_d)(F_d(2b+c)+(1-F_d)(2b+c)))$
    3. $F_i(H_ic+(1-H_i)(b+c))+(1-F_i)(H_dc+(1-H_d)(F_d(2b+c)+(1-F_d)(b+c)))$
    4. $F_i(H_ic+(1-H_i)(b+c))+(1-F_i)(H_dc+(1-H_d)(F_d(b+c)+(1-F_d)(2b+c)))$
    5. 以上全错

三、填空题

55. 一台单处理机可以以标量方式运行，也可以以向量方式运行。假设向量方式运算速度是标量方式的9倍。某基准程序在此处理机上运行时间为T，向量方式占整个运行时间的25%，其余的时间则以标量方式运行。

    1. 上述程序中向量化代码所占的比例是[$\frac{\frac{x}{9}}{(1-x)}=\frac{25\%}{75\%}\Rightarrow x=\frac{3}{4}$]。
    2. 我们通过硬件优化，使向量方式运算速度加倍，此时可达到的加速比（与全部用标量方式运行相比）是[$\frac{1}{1-75\%+\frac{75\%}{2*9}}=3.42$]。
    3. 如果要达到与第二问硬件优化相同的加速比，采用编译优化的软件方法，用向量化编译器支持同样的基
       准程序，其新的向量化代码所占的比例是[$\frac{1}{1-x+\frac{x}{9}}=3.42\Rightarrow x=79.61\%$]。

56. 假设一个Cache共有4个块，每块大小为1个字节。Cache采用LRU替换策略，初始时为空。当程序执行过程中访存的字节地址序列为：0，1，2，4，1，2，0，4 时，如果采用直接映射方式，Cache的命中次数为[2]次；如果采用 2 路组相连（用地址第 0 位作 index），Cache 的命中次数为[2]次。

Cache Before	Input	Hit	Cache After
xxxx	0	no	0xxx
0xxx	1	no	01xx
01xx	2	no	012x
012x	4	no	412x
412x	1	yes	412x
412x	2	yes	412x
412x	0	no	012x
012x	4	no	412x

Cache Before	Input	Hit	Cache After
xx|xx	0	no	0xxx
0x|xx	1	no	0x|1x
0x|1x	2	no	02|1x
02|1x	4	no	42|1x (LRU, 0 out)
42|1x	1	yes	42|1x
42|1x	2	yes	42|1x
42|1x	0	no	02|1x (LRU, 4 out)
02|1x	4	no	42|1x (LRU, 0 out)

57. 假设有一个两级Cache，L1 Cache的命中时间是2个时钟周期，缺失率为10%；L2 Cache的命中时间为10个时钟周期，局部缺失率（相对于所有L2 Cache访问）为20%，缺失代价为100个时钟周期，则平均存储器访问时间（AMAT）为[$2 + 10\% * (10 + 20\% * 100) = 5$]个时钟周期。

58. 1在CRAY-1机器上计算$D = A \times (B + C)$，其中A、B、C、D都是长度为64的向量，并且A、B、C已经存放在相应的向量寄存器中。CRAY-1的向量寄存器长度为64，所用浮点功能部件的执行时间分别为：相加5拍，相乘7拍，把向量数据元素送往功能部件以及把结果存入向量寄存器都需要1拍时间。不采用向量链接方式，完成该计算所需的最短拍数是[$(1 + 5 + 1 + 1 + 7 + 1) + 2N - 2 = 142$]拍；采用向量链接方式，完成该计算所需的最短拍数是[$1 + 5 + 1 + 1 + 7 + 1 + N - 1 = 79$]拍。

四、简答题

59. 设计一个处理器Cache，该Cache纯数据容量64K字节，采用4路组相联，Cache块大小为64字节，每个块有 1个有效（Valid）位和1个脏（Dirty）位。假设处理器的物理地址总共46位，试计算：

    1. 地址的tag域、index域以及block offset域的位数。（2分）
        - index域：组相联的情况下，index域分为组号和组内块号；每块64字节，说明有1024块，且由于有256组，组号需要8位
        - 块内地址（block offset）需要6位（64字节）
        - 剩下部分都划分给区号（即tag域）：46-8-6=32位
    2. 为实现该Cache，需要SRAM的总的位数（含纯数据Cache和附加信息位）是多少（以Kbits为单位）？（3分）
        - 纯数据64K字节（即512Kbits）
        - 区号存储器每条32+1+1=34位，由于有1024块因此有1024条，则这部分为34Kbits
        - index和块内地址不需要存储在SRAM里。
        - 共计546Kbits
    3. 假定Cache初始为空，整数大小为4字节，数组是Cache块对齐的。考虑以下两种对已访问位置计数的代码，请问哪段代码Cache命中率更高？说明原因，指明是由于哪种类型的Cache缺失造成的？
        - 右图的命中率高，因为整数数组是连续的，每块Cache能放16个整数；当访问1个整数并发现Cache未命中时，该块能放下的16个数都被写入Cache备用，因此剩下的15次访问都会命中；location数组中的每个整数是分散的，因此不仅没有上述机制，还可能导致结构体跟Cache块不对齐进而出现Cache互相覆盖，命中率会下降。
        - 冲突不命中+强制不命中

程序 A：

```
typedef struct {
    ... // other struct members
    int visited;
    int danger;
} location;

location locs[NUM_LOCS];
for (int i = 0; i < NUM_LOCS; i++)
    if (locs[i].visited) count++;
```

程序 B：

```
int visited[NUM_LOCS];
int danger[NUM_LOCS];
for (int i = 0; i < NUM_LOCS; i++)
    if (visited[i]) count++;
```

60. 假设一个RISC-V处理器，采用支持精确中断和推测执行的Tomasulo算法。假设该处理器有2个FP加法器，1个FP乘法/除法器，2个地址加法器，2个整数运算单元。在一个给定的周期内，只能发射（Issue）一条指令，只有一条指令可以访问数据Cache，只有一条指令可以写CDB，只有一条指令可以提交（Commit）。并且：

    - FP加减法运算需要3个周期
    - FP乘法/除法需要10个周期
    - 整数操作和地址计算（`Load`和`Store`指令在EXE段执行地址计算）需要1个周期
    - 访问数据Cache需要1个周期

    在下表中填写每条指令执行时各阶段的时钟周期号（注意：执行周期等于结束和开始的周期差+1），其中第一行和第一列已经给定。需要执行的代码如下：

```
add x3, x3, x2
fdiv.s f4, f2, f8
fadd.s f6, f10, f4
sub x3, x3, x4
fsw f6, 4(x3)
flw f2, 8(x3)
fsub.s f10, f8, f2
fadd.s f14, f6, f8
```

非常不确定，这题基本上一个人一个答案（x

此外，写CDB与下一条指令执行EXE开始之间的前后关系，答案中默认按照写CDB下一周期可以执行EXE进行安排。但按助教描述，似乎也有在写CDB的同一周期就执行的实现方法，具体要参考题目示例。

编号	指令	需要等待的前序指令
1	`add x3, x3, x2`	-
2	`fdiv.s f4, f2, f8`	-
3	`fadd.s f6, f10, f4`	2
4	`sub x3, x3, x4`	1
5	`fsw f6, 4(x3)`	3、4
6	`flw f2, 8(x3)`	2、4
7	`fsub.s f10, f8, f2`	6
8	`fadd.s f14, f6, f8`	5

指令	发射	执行EXE开始	执行EXE结束	读存储器	写CDB	提交
add	**1**	**2**	**2**	**-**	**3**	**4**
fdiv.s	**2**	4	13	-	14	15
fadd.s	**3**	15	17	-	18	19
sub	**4**	5	5	-	6	20
fsw	**5**	19	19	?	20	20（write mem）
flw	**6**	15	15	-	16	21
fsub.s	**7**	16	18	-	19	22
fadd.s	**8**	20	22	-	23	23

61. 使用2-bit饱和计数器实现分支预测。当计数器的值为0或1的时候预测为not taken；计数器的值为2或3的时候预测为taken。实际分支为taken时，计数器+1（最大为 3），否则，实际分支为not taken时，计数器-1（最小为 0）。假定初始状态下饱和计数器的值是 0，多个分支指令对应的饱和计数器相互独立，预测结果互不干扰。现有Fibonacci数列前几项的计算程序如下（左侧为高级语言，右侧为对应的汇编语言）：

```
1 procedure Compute(void):
2     a ← 0
3     b ← 1
4     for j ← 3 downto 0 do
5     begin
6         c ← a + b
7         b ← b + c
8         a ← c
9     end
10 procedure Main(Void):
11    For i ← 100 downto 1 do
12    begin
13        call Compute
14    end
```

```
1 . compute:
2.    mov $0, %eax              ; %eax = 0
3.    mov $1, %ebx              ; %ebx = 1
4.    mov $3, %ecx              ; %ecx = 3
5. loop:
6.      add %eax, %ebx, %edx    ; %edx = %eax + %ebx
7.      add %edx, %ebx, %ebx    ; %ebx = %edx + %ebx
8.      mov %edx, %eax          ; %eax = %edx
9.      sub $1, %ecx            ; %ecx = %ecx - 1
10.     jns loop                ; jump to loop if %ecx >= 0
11.   ret
13. main:
14.   mov $100, %ecx            ; %ecx = 100
15. loop2:
16.       call compute          ; call subroutine
17.       sub $1, %ecx          ; %ecx = %ecx - 1
18.       jnz loop2             ; jump to loop2 if %ecx != 0
19.       ret
```

右侧的汇编代码采用AT&T格式，在函数调用的时候忽视了栈的变化。`$imm`表示一个立即数，`%reg`表示一个通用寄存器。请问在执行右侧汇编语言`Main`例程时，采用2-bit饱和计数器进行分支预测的情况下，右图中第10行`jns`指令位置会发生多少次预测错误？请给出计算依据。

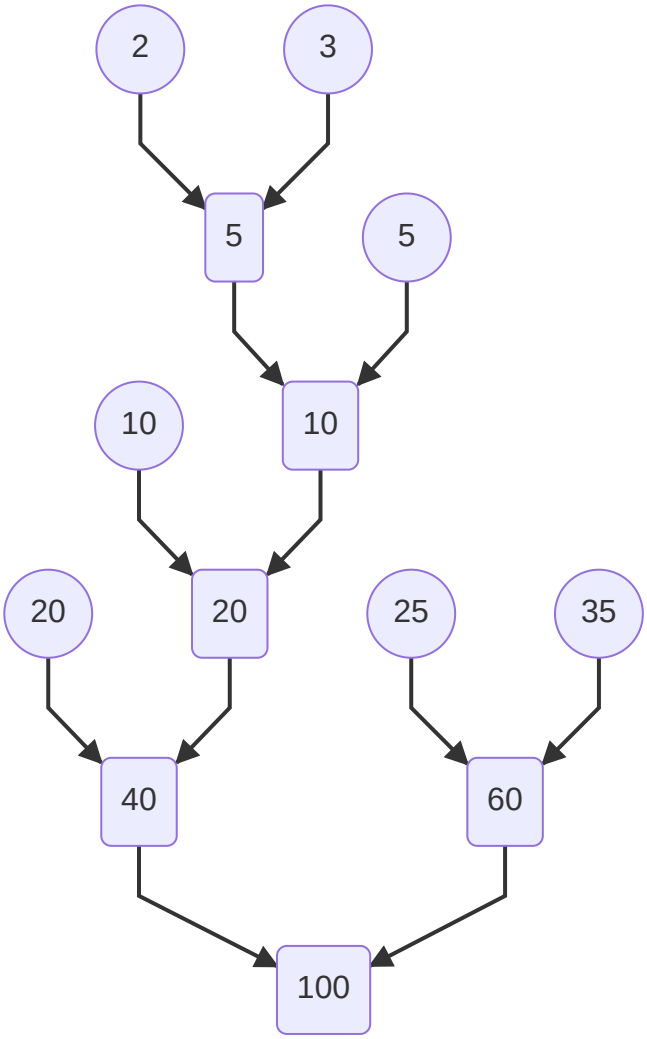
$jBefore$	$jAfter$	$valueBefore$	$tableTaken$	$actuallyTaken$	$valueAfter$
3	2	0	not	taken	1
2	1	1	not	taken	2
1	0	2	taken	taken	3
0	-1	3	taken	not	2
3	2	2	taken	taken	3
2	1	1	taken	taken	3
1	0	3	taken	taken	3
0	-1	3	taken	not	2

故在1-100的循环中，第一次循环会出现3次错误（对应`j`值为0、2、3）；随后每次循环发生1次错误（对应`j`值为0）。一共102次错误。

62. 一台模型机共有7条指令，各指令的使用频率分别为：35%，25%，20%，10%，5%，3%和2%，有8个通用数据寄存器，2个变址寄存器。

    1. 要求操作码的平均长度最短，请设计操作码的编码，并计算所设计操作码的平均长度。

建立Huffman树：



则：

- 2：10011
- 3：10010
- 5：1000
- 10：101
- 20：11
- 25：01
- 35：00
- 平均字长：2.35

2. 设计8字长的寄存器-寄存器型指令3条，16位字长的寄存器-存储器型变址寻址方式指令4条，变址范围不小于±127。请设计指令格式，并给出各字段的长度和操作码的编码。

   - 普通指令：3种指令对应的 `opcode` 占2位；通用寄存器8个，故每个占3位，两个占6位。则寄存器-寄存器型指令3条分别为：
     - 00+3位通用寄存器编码+3位通用寄存器编码
     - 01+3位通用寄存器编码+3位通用寄存器编码
     - 01+3位通用寄存器编码+3位通用寄存器编码
   - 变址寻址方式指令：由于普通指令使用了前两位的00、01和11，因此变址寻址方式指令均应以11开头，占2位；4种指令对应的 `opcode` 占2位；通用寄存器编码1个，占3位；变址寻址寄存器共2个，占1位；范围共255种，占8位。则寄存器-存储器型变址寻址方式指令的编码为：
     - 1100+3位通用寄存器编码+1位变址寻址寄存器编码+8位范围编码

- 1101+3位通用寄存器编码+1位变址寻址寄存器编码+8位范围编码
- 1110+3位通用寄存器编码+1位变址寻址寄存器编码+8位范围编码
- 1111+3位通用寄存器编码+1位变址寻址寄存器编码+8位范围编码