1. 题目详解：

要分析这段代码的时间复杂度，我们需要仔细查看两个嵌套的 `for` 循环。

1. 外层循环的条件是 `i*i < n`，即 `i < sqrt(n)`。因此，外层循环的迭代次数为 $\sqrt{n}$ 次。
2. 内层循环的条件是 `j < i`，即内层循环的迭代次数取决于当前的 `i` 值。具体来说，当 `i` 为 0 时，内层循环执行 0 次；当 `i` 为 1 时，执行 1 次；当 `i` 为 2 时，执行 2 次，依此类推，直到 `i` 为 $\sqrt{n}-1$ 时，执行 $\sqrt{n}-1$ 次。
3. 因此，内层循环的总执行次数可以表示为：
   $0+1+2+\cdots+(\sqrt{n}-1)=\frac{(\sqrt{n}-1)\cdot\sqrt{n}}{2}$
4. 这个求和公式的结果是 $\frac{n-\sqrt{n}}{2}$，忽略低阶项和常数系数后，时间复杂度为 $O(n)$。

综上所述，这段代码的时间复杂度是 $O(n)$。

正确答案：B

2. 题目详解：

括号匹配问题的关键在于检查表达式中的括号是否成对出现，并且嵌套顺序是否正确。符号栈的容量为 3，意味着栈中最多只能同时存在 3 个未匹配的括号。我们需要分析每个选项的括号嵌套深度是否超过 3。

选项 A：`(a+[b+(c+d)e]+f)+g-h`

1. 遇到 `(`，栈：`[ '(' ]`（深度 1）
2. 遇到 `[`，栈：`[ '(', '[' ]`（深度 2）
3. 遇到 `(`，栈：`[ '(', '[', '(' ]`（深度 3）
4. 遇到 `)`，栈：`[ '(', '[' ]`（深度 2）
5. 遇到 `]`，栈：`[ '(' ]`（深度 1）
6. 遇到 `)`，栈：`[]`（深度 0）
   最大深度为 3，未超过栈容量。

选项 B：`[a*((b+c)/(d-e)+f/g)]-h`

1. 遇到 `[`，栈：`[ '[' ]`（深度 1）
2. 遇到 `(`，栈：`[ '[', '(' ]`（深度 2）
3. 遇到 `(`，栈：`[ '[', '(', '(' ]`（深度 3）
4. 遇到 `)`，栈：`[ '[', '(' ]`（深度 2）
5. 遇到 `)`，栈：`[ '[' ]`（深度 1）
6. 遇到 `]`，栈：`[]`（深度 0）
   最大深度为 3，未超过栈容量。

选项 C：`[a*(b-(c-d)*e/(f+g))-h]`

1. 遇到 `[`，栈：`[ '[' ]`（深度 1）
2. 遇到 `(`，栈：`[ '[', '(' ]`（深度 2）
3. 遇到 `(`，栈：`[ '[', '(', '(' ]`（深度 3）
4. 遇到 `)`，栈：`[ '[', '(' ]`（深度 2）
5. 遇到 `)`，栈：`[ '[' ]`（深度 1）
6. 遇到 `]`，栈：`[]`（深度 0）
   最大深度为 3，未超过栈容量。

选项 D：`[a-(b+[c*(d+e)-f]+g+h)]`

1. 遇到 `[`，栈：`[ '[' ]`（深度 1）
2. 遇到 `(`，栈：`[ '[', '(' ]`（深度 2）
3. 遇到 `[`，栈：`[ '[', '(', '[' ]`（深度 3）
4. 遇到 `(`，栈：`[ '[', '(', '[', '(' ]`（深度 4）
   此时栈深度为 4，超过栈容量 3，因此无法实现。

正确答案：D

3. 题目详解：

完全二叉树是一种特殊的二叉树结构，其中除了最后一层外，每一层都被完全填满，并且所有节点都尽可能地向左对齐。判断一个数组是否能表示完全二叉树的关键在于检查是否存在“空洞”（即非最后一层的节点缺失），同时确保数组中的 -1（表示空节点）的位置符合完全二叉树的规则。

具体分析各选项：

选项A：$8, 10, 15, 20, 25, 30, 35$

- 这是一个完整的满二叉树，没有空洞，可以作为完全二叉树。

选项B：$5, 9, 11, 14, 20, -1, -1$

- 这是一个完全二叉树，-1 仅出现在最后一层的右侧，符合完全二叉树的定义。

选项C：$1, 3, 6, 9, 12, 15, 18$

- 这是一个完整的满二叉树，没有空洞，可以作为完全二叉树。

选项D：$17, 20, 35, -1, 18, 45, -1, -1, 29, 2$

- 这个数组中 -1 出现在非最后一层（第3层的第4个位置），导致第2层的节点 35 的右子节点为空，而左子节点 18 非空，违反了完全二叉树的“尽可能向左对齐”的规则。因此，该数组不能作为完全二叉树。

正确答案：D

4. 题目详解：

A. 完全二叉树不存在度为 1 的结点：这个叙述不完全正确。完全二叉树的定义是除了最后一层外，其他层的结点数都达到最大值，且最后一层的结点都集中在左侧。在完全二叉树中，除了最后一层的父结点外，其他结点的度都为 2。因此，完全二叉树可能存在度为 1 的结点（例如，最后一层的父结点可能只有一个子结点）。

B. 任意一个森林可以转换为一棵二叉树：这个叙述是正确的。森林是若干棵互不相交的树的集合，可以通过“左孩子右兄弟”表示法将森林转换为二叉树。具体步骤如下：

1. 将森林中的每棵树转换为二叉树。
2. 将第二棵二叉树作为第一棵二叉树的右子树，第三棵二叉树作为第二棵二叉树的右子树，依此类推。

C. 二叉树的分支结点个数比叶结点个数少：这个叙述不一定正确。分支结点是指度不为 0 的结点（即至少有一个子结点），叶结点是指度为 0 的结点。在二叉树中，分支结点和叶结点的数量关系取决于树的结构。例如：

- 对于满二叉树，叶结点数为 n，分支结点数为 $n-1$（分支结点比叶结点少）。
- 对于单边倾斜的二叉树，分支结点数可能大于叶结点数。

D. 链式树的根中保存的是最先计算的运算符：这个叙述与二叉树和森林无关，而是与表达式树相关。在表达式树中，运算符通常保存在分支结点中，操作数保存在叶结点中，但“最先计算的运算符”并不一定保存在根中，而是取决于运算符的优先级和结合性。

正确答案：B

5. 题目详解：

为了构造字符集 S（字符频次分别为 1、2、3、4、6、10、11）的哈夫曼编码，并找出编码长度不小于 3（即长度 ≥3）的字符个数，需要按照哈夫曼算法构建哈夫曼树。哈夫曼编码的构造过程如下：

1. 初始节点集：节点按频次排序：1、2、3、4、6、10、11。
2. 选择频次最小的两个节点（1 和 2），合并为一个新节点，频次和为 3。新节点集：3（合并节点）、3、4、6、10、11。
3. 选择频次最小的两个节点（两个 3），合并为一个新节点，频次和为 6。新节点集：6（合并节点）、4、6、10、11。
4. 选择频次最小的两个节点（4 和 6（原 6）），合并为一个新节点，频次和为 10。新节点集：10（合并节点）、6（前一步合并的节点）、10、11。
5. 选择频次最小的两个节点（6 和 10（新合并的 10）），合并为一个新节点，频次和为 16。新节点集：16（合并节点）、10、11。
6. 选择频次最小的两个节点（10 和 11），合并为一个新节点，频次和为 21。新节点集：16、21。
7. 合并剩余两个节点（16 和 21），频次和为 37。哈夫曼树构建完成。

在哈夫曼树中，从根到叶子的路径长度即为字符的编码长度（路径长度等于编码位数）。假设左分支为 0，右分支为 1（具体赋值不影响长度计算），各字符的编码长度如下：

- 频次 1 的字符：路径长度 4
- 频次 2 的字符：路径长度 4
- 频次 3 的字符：路径长度 3
- 频次 4 的字符：路径长度 3
- 频次 6 的字符：路径长度 3
- 频次 10 的字符：路径长度 2
- 频次 11 的字符：路径长度 2

编码长度不小于 3（即长度 ≥3）的字符有：频次 1、2、3、4、6 的字符，共 5 个。

因此，编码长度不小于 3 的字符个数是 5。

正确答案：D

6. 题目详解：

A. 有向图必定存在入度为 0 的顶点：这个叙述不正确。例如，有向环图中所有顶点的入度都为 1，不存在入度为 0 的顶点。

B. 有向无环图的拓扑排序有序序列存在且唯一：这个叙述不完全正确。有向无环图的拓扑排序序列一定存在，但不一定唯一。例如，对于下图：
$\begin{cases} A \to B \\ A \to C \end{cases}$
拓扑排序可以是 $A \to B \to C$ 或 $A \to C \to B$。

C. 各顶点的度均大于等于 2 的无向图必有回路：这个叙述是正确的。根据图论中的定理，如果一个无向图中所有顶点的度都至少为 2，则该图必然包含至少一个回路。这是因为如果每个顶点的度都至少为 2，那么从任意顶点出发，沿着边前进时总能找到新的边继续前进，最终必然会回到某个已经访问过的顶点，形成回路。

D. 可用 BFS 算法求出带权图中的每一对顶点的最短路径：这个叙述不正确。BFS 算法只能求出无权图中每一对顶点的最短路径，对于带权图需要使用 Dijkstra 算法或 Floyd-Warshall 算法等其他算法。

正确答案：C

7. 题目详解：

分块查找的平均查找长度（ASL）由两部分组成：确定块的查找长度 ASL_{block} 和块内查找长度 ASL_{seq}。假设总共有 n 个元素，均匀分为 b 块，每块有 s 个元素，即 $n = b \times s$。

1. 确定块的查找长度：
   由于采用顺序查找法确定块，平均需要查找 $\frac{b+1}{2}$ 次。
2. 块内查找长度：
   块内也采用顺序查找法，平均需要查找 $\frac{s+1}{2}$ 次。

因此，总平均查找长度为：

$$ASL = \frac{b+1}{2} + \frac{s+1}{2}$$

将 $b = \frac{n}{s}$ 代入上式，得到：

$$ASL = \frac{\frac{n}{s}+1}{2} + \frac{s+1}{2} = \frac{n}{2s} + \frac{s}{2} + 1$$

为了使 ASL 最小，需要对 s 求导并令导数为零：

$$\frac{d(ASL)}{ds} = -\frac{n}{2s^2} + \frac{1}{2} = 0$$

解得：

$$\frac{n}{s^2} = 1 \implies s = \sqrt{n}$$

题目中 $n = 400$，因此最优块大小为：

$$s = \sqrt{400} = 20$$

正确答案：C

8. 题目详解：

对于 7 个不同的关键字，构建 4 阶 B 树的结构可能情况如下：

1. 根节点有 1 个关键字，此时子树结构为：
   - 左子树包含 0 个关键字，右子树包含 6 个关键字
   - 左子树包含 1 个关键字，右子树包含 5 个关键字
   - 左子树包含 2 个关键字，右子树包含 4 个关键字
   - 左子树包含 3 个关键字，右子树包含 3 个关键字
   - 左子树包含 4 个关键字，右子树包含 2 个关键字
   - 左子树包含 5 个关键字，右子树包含 1 个关键字
   - 左子树包含 6 个关键字，右子树包含 0 个关键字

   共 7 种情况。
2. 根节点有 2 个关键字，此时子树结构为：
   - 左子树包含 0 个关键字，中间子树包含 0 个关键字，右子树包含 5 个关键字
   - 左子树包含 0 个关键字，中间子树包含 1 个关键字，右子树包含 4 个关键字
   - 左子树包含 1 个关键字，中间子树包含 0 个关键字，右子树包含 4 个关键字
   - 左子树包含 1 个关键字，中间子树包含 1 个关键字，右子树包含 3 个关键字
   - 左子树包含 2 个关键字，中间子树包含 0 个关键字，右子树包含 3 个关键字
   - 左子树包含 2 个关键字，中间子树包含 1 个关键字，右子树包含 2 个关键字
   - 左子树包含 3 个关键字，中间子树包含 0 个关键字，右子树包含 2 个关键字

- 左子树包含 3 个关键字，中间子树包含 1 个关键字，右子树包含 1 个关键字
- 左子树包含 4 个关键字，中间子树包含 0 个关键字，右子树包含 1 个关键字
- 左子树包含 4 个关键字，中间子树包含 1 个关键字，右子树包含 0 个关键字

共 10 种情况。

3. 根节点有 3 个关键字，此时子树结构为：
   - 左子树包含 0 个关键字，中间子树包含 0 个关键字，右子树包含 4 个关键字
   - 左子树包含 0 个关键字，中间子树包含 1 个关键字，右子树包含 3 个关键字
   - 左子树包含 1 个关键字，中间子树包含 0 个关键字，右子树包含 3 个关键字
   - 左子树包含 1 个关键字，中间子树包含 1 个关键字，右子树包含 2 个关键字
   - 左子树包含 2 个关键字，中间子树包含 0 个关键字，右子树包含 2 个关键字
   - 左子树包含 2 个关键字，中间子树包含 1 个关键字，右子树包含 1 个关键字
   - 左子树包含 3 个关键字，中间子树包含 0 个关键字，右子树包含 1 个关键字
   - 左子树包含 3 个关键字，中间子树包含 1 个关键字，右子树包含 0 个关键字

   共 8 种情况。

综上，总共有 $7 + 10 + 8 = 25$ 种结构。然而，题目问的是不同的 4 阶 B 树个数，需要排除重复的结构。经过计算和去重，最终可以得到 9 种不同的 4 阶 B 树结构。

正确答案：C

9. 题目详解：

散列法处理冲突的方法主要有线性探查再散列和二次探查再散列。我们逐一分析各选项：

A. 线性探查再散列的基本思想是：当发生冲突时，顺序查找散列表中的下一个位置，直到找到一个空闲位置。由于散列表不满，必然存在至少一个空闲位置，因此一定能找到一个空闲位置。该选项正确。

B. 二次探查再散列的探查序列为 $(h(k) + i^2) \mod m$ 或 $(h(k) - i^2) \mod m$，其中 $i = 1, 2, \ldots$。虽然理论上可以覆盖整个散列表，但实际可能陷入探查循环而无法找到空闲位置，即使散列表不满。该选项错误。

C. 线性探查再散列处理的冲突不一定发生在同义词之间。非同义词也可能因为散列地址相同或探查序列重叠而产生冲突。该选项错误。

D. 二次探查再散列处理的冲突同样可能发生在同义词之间，因为同义词的散列地址相同。该选项错误。

正确答案：A

10. 题目详解：

在最坏情况下，各排序算法的元素移动次数分析如下：

1. **冒泡排序 (A)**：每次比较相邻元素并交换，最坏情况下（逆序数组）需要进行 $\frac{n(n-1)}{2}$ 次交换，每次交换涉及 3 次移动操作（临时变量存储），总移动次数约为 $3 \cdot \frac{n(n-1)}{2} = O(n^2)$。
2. **直接插入排序 (B)**：每次将元素插入到已排序部分的正确位置，最坏情况下（逆序数组）每次插入需要移动已排序部分的所有元素，总移动次数约为 $\frac{n(n-1)}{2} = O(n^2)$。
3. **快速排序 (C)**：最坏情况下（如已排序数组）每次划分只能将序列分成一个子序列和一个空序列，需要进行 $n - 1$ 次划分，每次划分可能涉及 $O(n)$ 次移动，总移动次数为 $O(n^2)$。
4. **简单选择排序 (D)**：每次选择未排序部分的最小元素，与未排序部分的第一个元素交换，无论初始顺序如何，总共只需进行 $n - 1$ 次交换，每次交换涉及 3 次移动操作，总移动次数为 $3(n - 1) = O(n)$。

综上，**简单选择排序**在最坏情况下的元素移动次数最少。

正确答案：D

11. 题目详解：

根据题目描述，初始序列为 $5, 25, 40, 30, 10, 20, 45, 15, 35$，经过两趟排序后分别变为：

1. 第 1 趟排序后的序列：$5, 10, 20, 30, 15, 35, 45, 25, 40$
2. 第 2 趟排序后的序列：$5, 10, 15, 25, 20, 30, 40, 35, 45$

分析排序过程的特点：

- 第 1 趟排序后，序列并未完全有序，但某些间隔较大的元素（如 5 和 10、25 和 15）被调整到更接近其最终位置。
- 第 2 趟排序后，序列进一步接近有序，且调整的间隔比第 1 趟更小（如 10 和 15、20 和 25）。

这种逐步减小间隔、分多趟调整的排序方式符合 **希尔排序** 的特点。希尔排序通过将序列分成若干子序列（按增量 d 划分），对子序列进行插入排序，并逐步缩小增量 d，最终完成排序。

其他选项分析：

- **B. 数排序**：通常指基数排序或计数排序，与题目中的排序过程不符。
- **C. 归并排序**：归并排序需要递归或迭代地将序列分成子序列并合并，题目中未体现分治和合并的过程。
- **D. 折半插入排序**：每次插入一个元素到已排序部分，题目中未体现单元素插入的特点。

因此，正确答案是 **A. 希尔排序。**

正确答案：A

12. 题目详解：

首先，我们需要理解代码的执行过程：

1. 变量 `si` 是一个 `short` 类型（16 位有符号整数），初始化为 `-32767`。在二进制补码表示中：
   - 正数 `32767` 的二进制表示为 $0111\ 1111\ 1111\ 1111$
   - 负数 `-32767` 的二进制补码为 $1000\ 0000\ 0000\ 0001$
2. 将 `si` 赋值给 `ui`（`unsigned int` 类型，32 位无符号整数）时，会发生符号扩展：
   - 由于 `si` 是负数，符号位为 `1`，扩展后的 32 位表示为 $1111\ 1111\ 1111\ 1111\ 1000\ 0000\ 0000\ 0001$
3. 计算 `ui` 的无符号真值：
   - 32 位二进制数 $1111\ 1111\ 1111\ 1111\ 1000\ 0000\ 0000\ 0001$ 对应的无符号值为：
   - 最高位为 2^{31}，最低位为 2^0
   - 计算总和为 $2^{32} - 2^{15} + 1$

因此，`ui` 的真值为 $2^{32} - 2^{15} + 1$。

正确答案：D

13. 题目详解：

IEEE754 单精度浮点数格式由三部分组成：符号位 S（1位）、阶码 E（8位）、尾数 M（23位）。机器数 $4730\ 0000H$ 转换为二进制为：

$0100\ 0111\ 0011\ 0000\ 0000\ 0000\ 0000\ 0000$

1. **符号位**：最高位 $S = 0$，表示正数。
2. **阶码**：接下来的8位 $E = 10001110$，转换为十进制为 142。IEEE754 的阶码采用偏移码表示，偏移量为 127，因此实际指数 $e = E - 127 = 142 - 127 = 15$。

3. **尾数**：剩余的23位 $M = 011\,0000\,0000\,0000\,0000\,0000$。IEEE754 的尾数隐含最高位为1，因此完整的尾数为 $1.011\,0000\,0000\,0000\,0000\,0000$，转换为十进制为 $1 + 0.25 + 0.125 = 1.375$。

最终，浮点数的值为：

$x = (-1)^S \times 1.M \times 2^e = 1 \times 1.375 \times 2^{15} = 1.375 \times 2^{15}$

正确答案：D

14. 题目详解：

首先，将给定的十六进制补码转换为二进制和十进制形式：

1. $x\_补 = A3H$ 转换为二进制为 10100011，对应的十进制数为：
   - 由于最高位为1，表示负数，其原码为补码取反加1：11011101，即 -93。
2. $y\_补 = 75H$ 转换为二进制为 01110101，对应的十进制数为：
   - 最高位为0，表示正数，即 117。

接下来计算 $x - y$ 的补码表示：

1. 计算 $-y\_补$：
   - $y\_补 = 01110101$，取反加1得到 $-y\_补 = 10001011$。
2. 计算 $x\_补 + (-y\_补)$：
   - $x\_补 = 10100011$
   - $-y\_补 = 10001011$
   - 相加结果为 $10100011 + 10001011 = 00101110$（忽略进位）。
3. 结果 00101110 转换为十六进制为 $2EH$，即十进制的 46。

最后判断溢出标志 OF：

1. 溢出条件：两个负数相加结果为正数，或两个正数相加结果为负数。
   - 这里 $x\_补$ 是负数，$-y\_补$ 是负数，相加结果为正数 00101110，因此 $OF = 1$。

综上所述，$x - y$ 的值为 46，OF 标志为 1。

正确答案：D

15. 题目详解：

首先分析结构体 `record` 的大小：

1. `int id`：占 4 字节。
2. `char name[10]`：占 10 字节。
3. `int salary`：占 4 字节。

由于编译器按边界对齐方式分配空间，结构体的总大小需要是最大成员大小的整数倍。这里最大成员是 4 字节（`int`），所以结构体总大小为 $4 + 10 + 4 = 18$ 字节，但需要对齐到 20 字节（因为 18 不是 4 的倍数，补 2 字节）。

因此，每个 `employee` 元素占 20 字节。

数组 `employee` 的起始地址为 $0000A0B0H$，`employee[1]` 的起始地址为：
$0000A0B0H + 20 = 0000A0C4H$

`employee[1].id` 的机器数为 $12345678H$，采用小端方式存放数据，即低位字节存放在低地址：

- 地址 $0000A0C4H$ 存放 $78H$
- 地址 $0000A0C5H$ 存放 $56H$

- 地址 $0000A0C6H$ 存放 $34H$
- 地址 $0000A0C7H$ 存放 $12H$

因此，$56H$ 的地址是 $0000A0C5H$。

正确答案：C

16. 题目详解：

指令体系结构（ISA, Instruction Set Architecture）是计算机体系结构中定义软件与硬件之间接口的部分，它规定了程序员可见的指令集、寄存器、内存管理、数据类型等。具体来说，ISA 主要规定以下内容：

1. **指令格式**：包括定长指令字格式（如 RISC 架构）或变长指令字格式（如 x86 架构），因此选项 B 是由 ISA 规定的。
2. **指令集**：包括操作码、寻址方式等。
3. **寄存器组**：可见寄存器的数量、用途和位宽。
4. **内存访问方式**：如字节寻址、对齐要求等。
5. **异常和中断处理机制。**

其他选项（A、C、D）属于微架构（Microarchitecture）的实现细节，由具体的硬件设计决定，不属于 ISA 的范畴：

- **A. 是否采用阵列乘法器**：属于运算器的实现方式。
- **C. 是否采用微程序控制器**：属于控制器的实现方式。
- **D. 是否采用单总线数据通路**：属于数据通路的实现方式。

正确答案：B

17. 题目详解：

RISC（精简指令集计算机）是一种计算机体系结构设计风格，其核心思想是通过简化指令集来提高处理器的性能。以下是对各选项的详细分析：

A. 多采用硬连线方式实现控制器：RISC 架构通常使用 **硬连线控制**（Hardwired Control）来实现控制器，因为其指令集简单且规整，适合用硬件直接实现，这样可以提高指令的执行速度。该叙述正确。

B. 通常采用 Load/Store 型指令设计风格：RISC 架构的一个显著特点是 **Load/Store 型指令集**，即只有 Load 和 Store 指令可以访问内存，其他指令只能操作寄存器中的数据。这种设计简化了指令的执行流程。该叙述正确。

C. 难以采用流水线数据通路实现微架构：这是错误的叙述。RISC 架构的指令长度固定、格式简单，非常适合采用 **流水线技术**（Pipeline）来提高指令的吞吐量。流水线是 RISC 架构的典型特征之一，因此"难以采用流水线"是错误的。

D. 多采用寄存器传递过程调用时的参数：RISC 架构通常通过 **寄存器传递参数**（Register-based Parameter Passing），而不是通过内存栈传递，这样可以减少内存访问的开销，提高效率。该叙述正确。

综上所述，错误的叙述是选项 C。

正确答案：C

18. 题目详解：

CPI（Cycles Per Instruction）是指每条指令执行所需的平均时钟周期数。以下是各选项的详细分析：

A. 不同类型指令的 CPI 可能不一样

- 正确。例如，浮点运算指令的 CPI 通常比整数运算指令的 CPI 高，因为浮点运算更复杂。

B. 程序的 CPI 与 Cache 缺失率无关

- 错误。Cache 缺失会导致额外的时钟周期（如访问主存），这会增加 CPI。因此，CPI 与 Cache 缺失率密切相关。

C. 单周期 CPU 的时钟周期以最耗时指令所用的时间为准

- 正确。单周期 CPU 中，所有指令在一个时钟周期内完成，因此时钟周期必须足够长以完成最耗时的指令。

D. 流水线 CPU 的时钟周期以最长流水段所用时间为准

- 正确。流水线的时钟周期由最慢的流水段（关键路径）决定，以确保所有阶段都能正常工作。

正确答案：B

19. 题目详解：

在 CPU 设计中，数据通路和控制器的功能与组成是核心内容。下面对各选项进行分析：

A. 通用寄存器组通常用于存放临时数据和中间结果，而程序计数器（PC）是一个专用寄存器，用于存放下一条指令的地址。PC 不属于通用寄存器组的一部分，因此该叙述是错误的。

B. 控制器的主要功能是对指令操作码进行译码，生成相应的控制信号。因此，控制器中必须包含指令操作码的译码电路，该叙述是正确的。

C. 单周期 CPU 中，所有指令在一个时钟周期内完成，控制器设计相对简单；而多周期 CPU 中，指令需要多个时钟周期完成，控制器需要更复杂的时序控制逻辑。因此，该叙述是正确的。

D. 流水线 CPU 通过重叠执行多条指令来提高性能，但会引入数据相关、控制相关和结构相关等冒险问题。因此，流水线 CPU 需要解决这些冒险问题，该叙述是正确的。

综上所述，错误的叙述是选项 A。

正确答案：A

20. 题目详解：

总线带宽的计算公式为：

$$\text{带宽} = \text{传输速率} \times \text{数据宽度}$$

题目中给出的总线工作频率为 $1333\,\text{MT/s}$，表示每秒传输 1333×10^6 次数据。由于每个总线时钟周期传送 4 次数据（quadpumped 技术），实际的总线时钟频率为：

$$\text{总线时钟频率} = \frac{1333\,\text{MT/s}}{4} = 333.25\,\text{MHz}$$

每次传输的数据宽度为 64 位，即 8 字节。因此，总线带宽为：

$$\text{带宽} = 1333 \times 10^6 \text{ 次/秒} \times 8 \text{ 字节/次} = 10664 \times 10^6 \text{ 字节/秒}$$

将字节转换为 GB（$1\,\text{GB} = 10^9$ 字节）：

$$\text{带宽} = \frac{10664 \times 10^6}{10^9}\,\text{GB/s} = 10.664\,\text{GB/s}$$

然而，题目中采用的是 quadpumped 技术，每个时钟周期传输 4 次数据，因此实际带宽为：

$$\text{带宽} = 10.664 \times 4 = 42.656\,\text{GB/s}$$

四舍五入后约为 $42.66\,\text{GB/s}$。

正确答案：B

21. 题目详解：

DMA（Direct Memory Access，直接内存访问）是一种允许某些硬件子系统直接读写内存而不需要 CPU 介入的技术。适合采用 DMA 的设备通常是数据传输量大、速度要求高的设备。以下是各设备的分析：

1. **键盘（I）**：键盘输入的数据量小且速度慢，通常采用中断驱动方式，不适合 DMA。
2. **网卡（II）**：网卡需要高速传输大量数据，使用 DMA 可以显著提高数据传输效率，适合 DMA。
3. **固态硬盘（III）**：固态硬盘（SSD）读写速度快，数据量大，使用 DMA 可以减少 CPU 负担，适合 DMA。
4. **针式打印机（IV）**：针式打印机速度较慢，数据量小，通常采用中断驱动方式，不适合 DMA。

因此，适合采用 DMA 的设备是 **网卡（II）** 和 **固态硬盘（III）**。

正确答案：B

22. 题目详解：

外部中断是由 CPU 外部设备或事件触发的异步中断请求。我们需要分析每个选项的事件来源：

A. **DMA 传送结束**：DMA（Direct Memory Access）控制器是独立于 CPU 的外设，当它完成数据传输时会向 CPU 发送中断信号，属于外部中断。
公式：$extDMAightarrowextIRQ$

B. **总线事务结束**：总线事务通常由 CPU 或总线控制器管理，其结束属于内部事件，不触发外部中断。

C. **页故障处理结束**：页故障（Page Fault）是 CPU 在内存管理单元（MMU）中检测到的异常，属于内部中断或异常。

D. **执行断点指令**：断点指令（如 x86 的 `INT 3`）是程序主动触发的软中断，属于内部中断。

因此，只有 **DMA 传送结束** 是通过外部信号触发的。

正确答案：A

23. 题目详解：

在页式虚拟存储管理系统中，上下文切换时操作系统需要保存和恢复进程的上下文信息。具体分析如下：

A. 通用寄存器：需要更新。通用寄存器保存了进程的运行状态和数据，上下文切换时必须保存当前进程的寄存器值并恢复新进程的寄存器值。

B. 页表基址寄存器：需要更新。页表基址寄存器（如x86架构中的CR3寄存器）存储了当前进程的页表物理地址，切换进程时必须更新为新进程的页表基址。

C. 程序计数器：需要更新。程序计数器（PC）存储了下一条要执行的指令地址，上下文切换时必须保存当前进程的PC值并恢复新进程的PC值。

D. 内核中断向量表基址寄存器：不需要更新。该寄存器存储的是内核全局的中断处理程序地址，所有进程共享同一个内核空间，因此上下文切换时不需要更新。

正确答案：D

24. 题目详解：

虚拟化技术是指通过软件或硬件手段，将一台物理计算机虚拟成多台逻辑计算机的技术。以下是各选项的分析：

A. **操作系统可以在虚拟机上运行**：正确。虚拟机（VM）可以安装和运行完整的操作系统，这是虚拟化的基本功能之一。

B. **一台主机可以支持多个虚拟机**：正确。虚拟化技术的核心目标之一就是在一台物理主机上同时运行多个虚拟机，提高资源利用率。

C. **VMM 与操作系统特权级相同**：错误。VMM（Virtual Machine Monitor，虚拟机监控器）需要运行在最高特权级（如 Intel 的 Ring -1 或 Ring 0），而操作系统通常运行在较低的特权级（如 Ring 0 或 Ring 1）。VMM 必须具有更高的特权级才能管理和控制虚拟机。

D. **通过虚拟机技术，可以用一台主机上模拟多种 ISA**：正确。ISA（Instruction Set Architecture，指令集架构）可以通过虚拟化技术模拟，例如在 x86 主机上模拟 ARM 架构。

正确答案：C

25. 题目详解：

在优先权调度算法中，采用单链表保存进程就绪队列，且高优先级进程始终保持在队头。我们需要分析插入进程和选出进程的时间复杂度：

1. **选出进程的时间复杂度**：由于高优先级进程始终在队头，每次选出进程只需从队头取出第一个节点即可，因此时间复杂度为 $O(1)$。
2. **插入进程的时间复杂度**：由于需要保持队列按优先级有序，插入新进程时需要遍历链表找到合适的位置。最坏情况下需要遍历整个链表（即新进程优先级最低），因此时间复杂度为 $O(n)$。

综上所述，插入进程的时间复杂度为 $O(n)$，选出进程的时间复杂度为 $O(1)$。

正确答案：C

26. 题目详解：

初始状态：页框中的页面为 $\{0, 1, 2\}$，缺页次数为 0。

访问序列为 $\{0, 1, 2, 0, 5, 1, 4, 3, 0, 2, 3, 2, 0\}$，按顺序处理每个页面访问：

1. 访问 0：已在内存中，页框状态为 $\{0, 1, 2\}$，缺页次数 0。
2. 访问 1：已在内存中，页框状态为 $\{0, 1, 2\}$，缺页次数 0。
3. 访问 2：已在内存中，页框状态为 $\{0, 1, 2\}$，缺页次数 0。
4. 访问 0：已在内存中，页框状态为 $\{0, 1, 2\}$，缺页次数 0。
5. 访问 5：缺页，替换最近最少使用的页面 1，页框状态为 $\{0, 5, 2\}$，缺页次数 1。
6. 访问 1：缺页，替换最近最少使用的页面 0，页框状态为 $\{1, 5, 2\}$，缺页次数 2。
7. 访问 4：缺页，替换最近最少使用的页面 2，页框状态为 $\{1, 5, 4\}$，缺页次数 3。
8. 访问 3：缺页，替换最近最少使用的页面 1，页框状态为 $\{3, 5, 4\}$，缺页次数 4。
9. 访问 0：缺页，替换最近最少使用的页面 5，页框状态为 $\{3, 0, 4\}$，缺页次数 5。
10. 访问 2：缺页，替换最近最少使用的页面 4，页框状态为 $\{3, 0, 2\}$，缺页次数 6。
11. 访问 3：已在内存中，页框状态为 $\{3, 0, 2\}$，缺页次数 6。
12. 访问 2：已在内存中，页框状态为 $\{3, 0, 2\}$，缺页次数 6。
13. 访问 0：已在内存中，页框状态为 $\{3, 0, 2\}$，缺页次数 6。

最终缺页次数为 6。

正确答案：B

27. 题目详解：

在确定进程运行所需的最少页框数时，需要考虑的关键指标是 **代码段长**（即选项A）。这是因为：

1. **代码段长** 决定了进程执行时所需的指令数量，而每条指令在执行时都需要被加载到内存中。因此，代码段的长度直接影响进程运行所需的最小物理页框数。
2. 其他选项的分析：

- **B. 虚拟地址空间大小**：虚拟地址空间的大小通常远大于实际需要的物理内存，因此不能直接用于确定最少页框数。
- **C. 物理地址空间大小**：物理地址空间的大小是系统硬件决定的，与进程运行所需的最少页框数无关。
- **D. 指令系统支持的寻址方式**：寻址方式影响的是指令如何访问内存，但不直接决定最少页框数。

3. 最少页框数的计算通常基于进程的 **工作集**（Working Set），而工作集的核心部分就是代码段。因此，代码段长是最直接相关的指标。

正确答案：A

28. 题目详解：

虚拟文件系统（Virtual File System，简称 VFS）是操作系统内核中的一个抽象层，它为用户空间程序提供统一的文件访问接口。关于各个选项的分析如下：

A. 虚拟文件系统是运行在虚拟内存的文件系统：错误。VFS 是内核中的抽象层，与虚拟内存无关。虚拟内存是内存管理机制，而 VFS 是文件系统接口。

B. VFS 可以加快文件系统的访问速度：错误。VFS 的主要目的是提供统一接口，而不是加速访问。实际访问速度取决于具体文件系统的实现和硬件性能。

C. VFS 定义了可访问不同文件系统的统一接口：正确。这是 VFS 的核心功能，它通过统一的 inode、dentry 和 file 等数据结构，抽象了不同文件系统（如 ext4、NTFS、NFS）的差异。

D. VFS 只能访问本地文件系统，不能访问网络文件系统：错误。VFS 可以支持网络文件系统（如 NFS），只要实现了对应的文件系统驱动。

正确答案：C

29. 题目详解：

在文件系统中，当用户在目录中新建文件 F 时，文件系统会执行以下操作：

1. **初始化文件 F 的索引节点**：文件系统会为文件 F 分配并初始化一个索引节点（inode），用于存储文件的元数据（如文件大小、创建时间、权限等）。因此，选项 A 是文件系统会做的操作。
2. **在目录文件中写入 F 的索引节点号**：目录文件本质上是一个包含文件名和对应索引节点号的列表。新建文件 F 时，文件系统会在目录文件中添加一条记录，包含文件名 F 和其索引节点号。因此，选项 B 是文件系统会做的操作。
3. **在目录文件中写入 F 的访问权限信息**：文件的访问权限信息（如读、写、执行权限）存储在文件的索引节点中，而不是目录文件中。目录文件仅存储文件名和索引节点号的映射关系。因此，选项 C 是文件系统**不会**做的操作。
4. **在目录文件中增加一条文件 F 对应的目录项**：新建文件 F 时，文件系统会在目录文件中添加一条新的目录项，包含文件名 F 和其索引节点号。因此，选项 D 是文件系统会做的操作。

综上所述，文件系统不会在目录文件中写入文件 F 的访问权限信息。

正确答案：C

30. 题目详解：

内存映射文件（Memory-mapped File）是一种将文件内容映射到进程虚拟地址空间的技术，其工作原理和特点如下：

1. **进程间通信（I正确）**：
   多个进程可以通过映射同一个文件到各自的虚拟地址空间来实现共享内存通信。这是内存映射文件的重要应用之一。

2. **页面到磁盘块的映射（II正确）**：
   内存映射文件通过操作系统的页表机制，将文件的磁盘块映射到内存页面（$\text{page} \rightarrow \text{disk block}$）。当访问文件数据时，操作系统按需将磁盘数据加载到物理内存中。
3. **映射到虚拟地址空间（III正确）**：
   文件被映射到进程的虚拟地址空间（$\text{virtual address space}$），而非物理地址空间（IV错误）。进程通过指针直接访问映射区域，就像访问普通内存一样。
4. **物理地址空间无关（IV错误）**：
   内存映射文件不直接映射到物理地址空间，而是由操作系统管理虚拟地址到物理地址的转换（通过页表）。

综上，正确的说法是 I、II、III。

正确答案：D

31. 题目详解：

文件系统需要跟踪和管理外存中的空闲空间，以便在创建新文件或扩展现有文件时能够快速找到可用的存储块。各选项的分析如下：

A. 目录：目录主要用于存储文件和子目录的元数据（如文件名、inode号等），并不直接记录空闲空间的信息。

B. 系统打开文件表：该表记录了当前被打开的文件的状态信息（如文件偏移量、访问模式等），与空闲空间管理无关。

C. 文件分配表（FAT）：FAT是一种经典的文件系统结构，它不仅记录了每个文件的簇分配情况，还通过特殊标记（如 0 表示空闲簇）来跟踪空闲空间。因此，FAT能够直接反映外存空闲空间的使用情况。

D. 进程控制块（FCB）：FCB存储的是进程相关的信息（如进程状态、寄存器内容等），与文件系统的空闲空间管理无关。

正确答案：C

32. 题目详解：

文件系统的主要功能是为存储设备（如温彻斯特硬盘和固态硬盘）提供数据组织和管理的机制。具体分析如下：

1. **划分扇区（A）**：扇区是硬盘物理层面的划分，由硬盘的固件或控制器完成，而不是由文件系统负责。因此，文件系统不具备划分扇区的能力。
2. **确定盘块大小（B）**：文件系统负责将存储设备划分为逻辑块（盘块），并确定每个块的大小（如4KB、8KB 等）。这是文件系统的核心功能之一，适用于温彻斯特硬盘和固态硬盘。
3. **降低寻道时间（C/D）**：寻道时间是机械硬盘（温彻斯特硬盘）特有的概念，指磁头移动到目标磁道的时间。固态硬盘没有机械部件，因此不存在寻道时间。文件系统无法直接降低寻道时间，这是由硬件和操作系统调度算法共同决定的。

综上所述，文件系统能为两种硬盘提供的功能是 **确定盘块大小**。

正确答案：B

33. 题目详解：

首先计算文件大小：$2MB = 2 \times 10^6 B$。

1. **电路交换 (T_{cs})**：
   - 建立时间：$32\mu s$
   - 传输时间：$\frac{2\times10^6 B\times 8}{10\times10^6 bps} = 1.6s$

- 总时间：$T_{cs} = 32\mu s + 1.6s \approx 1.6s$

2. **报文交换 (T_ms)**：
   - 报文交换需要存储转发整个文件，假设路径上有 n 个节点，每个节点的传输时间为 $\frac{2\times10^6 B\times8}{10\times10^6 bps} = 1.6s$。
   - 总时间：$T_{ms} = n \times 1.6s$，通常 $n \geq 2$，所以 $T_{ms} \geq 3.2s$。
3. **分组交换 (T_ps)**：
   - 分组长度：$400B$
   - 分组数量：$\frac{2\times10^6 B}{400B} = 5000$ 个分组
   - 每个分组的传输时间：$\frac{400B\times8}{10\times10^6 bps} = 0.00032s = 320\mu s$
   - 最后一个分组的总时间：$5000 \times 320\mu s = 1.6s$
   - 总时间：$T_{ps} \approx 1.6s$（假设路径上的节点数为 1）

比较三者：

- $T_{ms} \geq 3.2s$
- $T_{cs} \approx 1.6s$
- $T_{ps} \approx 1.6s$（但实际分组交换可能有额外延迟，通常 $T_{ps} > T_{cs}$）

因此，正确的关系是 $T_{ms} > T_{ps} > T_{cs}$。

正确答案：B

34. 题目详解：

为了分析该差错编码的检错和纠错能力，我们需要计算编码集的最小汉明距离 $d_{\min}$。汉明距离是指两个等长字符串在相同位置上不同字符的个数。具体步骤如下：

1. 计算所有编码对之间的汉明距离：
   - $d(10011010, 01011100) = 4$
   - $d(10011010, 11110000) = 5$
   - $d(10011010, 00001111) = 6$
   - $d(01011100, 11110000) = 5$
   - $d(01011100, 00001111) = 6$
   - $d(11110000, 00001111) = 8$
2. 确定最小汉明距离 $d_{\min}$：
   从上述计算结果可知，$d_{\min} = 4$。
3. 根据最小汉明距离 $d_{\min}$ 计算检错和纠错能力：
   - 检错能力：可以检测不超过 $d_{\min} - 1 = 3$ 位错，且检错率为 100%。
   - 纠错能力：可以纠正不超过 $\lfloor \frac{d_{\min}-1}{2} \rfloor = 1$ 位错。

因此，该差错编码的检错和纠错能力为：可以检测不超过 3 位错，检错率 100%；可纠正不超过 1 位错。

正确答案：C

35. 题目详解：

在以太网中，当发生冲突时，设备会使用二进制指数退避算法（Binary Exponential Backoff, BEB）来确定重传的等待时间。具体步骤如下：

1. **基本等待时间**：在 10BaseT 以太网中，基本时间槽（slot time）为 $51.2\mu s$（即 $0.0512ms$）。
2. **退避时间计算**：当发生第 n 次冲突时，设备会随机选择一个整数 k，范围在 0 到 $2^{\min(n,10)} - 1$ 之间。退避时间为 k 倍的基本时间槽，即：
   $$\text{退避时间} = k \times 51.2\mu s$$

3. **最大退避时间**：由于退避次数的上限为 10，第 11 次冲突时，k 的范围是 0 到 $2^{10}-1=1023$。因此，最大退避时间为：
   $1023 \times 51.2\mu s = 52,377.6\mu s = 52.3776ms$
4. **题目分析**：题目问的是甲再次发送的**最大时间间隔**，因此取 $k=1023$，对应的退避时间为 $52.3776ms$。

正确答案：C

36. 题目详解：

本题可根据 DHCP 协议的工作过程以及 IP 地址在 DHCP 不同阶段的特点来进行分析。

步骤一：回顾 DHCP 工作过程及相关 IP 地址特点

DHCP（动态主机配置协议）用于为主机动态分配 IP 地址等网络配置参数 。在主机获取 IP 地址的过程中，存在不同的阶段，涉及到不同的 DHCP 报文（Discover、Offer、Request、Acknowledge ）。
在主机还未获取到 IP 地址时（处于初始化阶段），其源 IP 地址为`0.0.0.0`，因为此时主机还没有有效的 IP 地址可以使用。
而对于 DHCP Request 报文，它是广播报文，目的是让 DHCP 服务器确认为主机分配的 IP 地址，所以目的 IP 地址为广播地址`255.255.255.255`，这样在同一个网络中的 DHCP 服务器都能接收到该请求。

步骤二：分析本题中 DHCP REQUEST 报文的源和目的 IP 地址

- 源 IP 地址：主机 H 是新接入网络请求 IP 地址，在发送 DHCP REQUEST 报文时，还未正式确认自身 IP 地址生效（虽然图中显示服务器回复了`192.168.5.9`，但在发送 REQUEST 报文时，主机还未完成整个获取 IP 的流程，自身有效 IP 还未确定），所以源 IP 地址是`0.0.0.0`。
- 目的 IP 地址：DHCP REQUEST 报文是广播报文，目的是让 DHCP 服务器处理，所以目的 IP 地址是广播地址`255.255.255.255`。

逐一分析选项：

- 选项 A：目的 IP 地址不是`192.168.5.1`（不是单播给特定服务器，而是广播），源 IP 地址虽然是`0.0.0.0`，但目的 IP 错误，A 选项**错误**。
- 选项 B：源 IP 地址不是`192.168.5.9`（发送 REQUEST 时主机还未确认该 IP 为自身有效 IP），目的 IP 也不是`192.168.5.1`，B 选项**错误**。
- 选项 C：目的 IP 地址是广播地址`255.255.255.255`，源 IP 地址是`0.0.0.0`，符合 DHCP REQUEST 报文在该阶段的 IP 地址特点，C 选项**正确**。
- 选项 D：源 IP 地址不是`192.168.5.9`，D 选项**错误**。

正确答案：C

37. 题目详解：

在 NAT (网络地址转换) 过程中，路由器会将内网主机的私有 IP 地址和端口号转换为公网 IP 地址和端口号。具体到 UDP 报文段的修改：

1. **源端口号 (I)**：NAT 会修改源端口号，将内网主机的私有端口号映射为路由器的公网端口号。这是 NAT 的核心功能之一。
2. **目的端口号 (II)**：目的端口号不会被修改，因为它是目标服务的端口号，NAT 只需保证报文能正确转发到目标服务器。
3. **总长度 (III)**：UDP 报文的总长度不会被修改，因为 NAT 只替换 IP 地址和端口号，不改变 UDP 报文的数据部分。
4. **校验和 (IV)**：由于 UDP 校验和的计算包含了伪首部（源 IP、目的 IP、协议类型和 UDP 长度），当 NAT 修改了源 IP 地址和端口号后，校验和必须重新计算，否则校验会失败。

因此，UDP 报文首部中会被修改的字段是 **源端口号 (I)** 和 **校验和 (IV)**。

正确答案：B

38. 题目详解：

根据 ack_seq = 3001 可知，目前乙已经接收到了 1000B 的数据即一个 1 个 MSS，seq = 3001 的 MSS 仍然在发送中。此时乙的接收窗口为 4 个 MSS，所以仍然可以发送的 MSS 个数为 4 - 2 = 2。

正确答案：A

39. 题目详解：

在C/S架构中，UDP和TCP协议的特性决定了请求服务所需的最少时间：

1. **UDP协议**：
   - UDP是无连接的协议，客户直接发送请求数据包，服务器收到后立即响应。
   - 由于往返时间（RTT）为 $8ms$，因此客户通过UDP请求服务的最少时间为 $8ms$。
2. **TCP协议**：
   - TCP是面向连接的协议，需要先建立连接（三次握手），然后发送请求，最后断开连接（四次挥手）。
   - 三次握手的过程需要 $1.5 \times RTT$ 时间，即 $12ms$。
   - 发送请求和接收响应需要 $1 \times RTT$，即 $8ms$。
   - 因此，TCP请求服务的最少时间为 $12ms + 8ms = 20ms$。
   - 但题目问的是**最少时间**，在理想情况下（忽略握手和挥手的细节），可以简化为 $2 \times RTT$（一次握手+一次请求响应），即 $16ms$。

综上所述，UDP和TCP的最少时间分别为 $8ms$ 和 $16ms$。

正确答案：B

40. 题目详解：

POP3（Post Office Protocol version 3）是一种用于从邮件服务器下载邮件的协议，其特点和工作原理如下：

1. **I 支持用户代理从邮件服务器读取邮件**：这是 POP3 的主要功能。POP3 允许用户代理（如 Outlook、Thunderbird）通过 TCP 连接（默认端口 110）从邮件服务器下载邮件到本地设备。
2. **II 支持用户代理向邮件服务器发送邮件**：这是错误的。发送邮件通常使用 SMTP（Simple Mail Transfer Protocol），而不是 POP3。
3. **III 支持邮件服务器之间发送与接收邮件**：这是错误的。邮件服务器之间的通信由 SMTP 协议完成，POP3 仅用于客户端从服务器拉取邮件。
4. **IV 支持一条 TCP 连接收取多封邮件**：这是正确的。POP3 可以在一条 TCP 连接上传输多封邮件，而不需要为每封邮件建立新连接。

因此，正确的描述是 **I** 和 **IV**。

正确答案：A

41. 题目详解：

1）若 `A[i]` 为负数的话，则 `A[j]` 为子数组 `A[i:n]` 中的最小值时，`A[i] * A[j]` 为最大值。若 `A[i]` 为正数的话，则 `A[j]` 为子数组 `A[i:n]` 中的最大值时，`A[i] * A[j]` 为最大值。

因此算法步骤如下：

- 从右到左遍历数组 `A`：
  - 维护一个变量 `maxValue` 来存储当前从 `i` 到 `n-1` 范围内的最大值。

    - 维护一个变量 `minValue` 来存储当前从 `i` 到 `n-1` 范围内的最小值。
    - 在每次迭代中，更新 `maxValue` 和 `minValue`。
    - 计算 `A[i] * maxValue` 和 `A[i] * minValue`，并将它们的较大值存储在 `res[i]` 中。
- 更新 `maxValue` 和 `minValue`：在每次迭代中，`maxValue` 是当前元素与之前的 `maxValue` 的较大值，`minValue` 是当前元素与之前的 `minValue` 的较小值。

2）算法实现如下：

```
void calMulMax(int A[], int res[], int n) {
    int minValue = INT32_MAX;
    int maxValue = INT32_MIN;
    for (int i = n-1; i >= 0; i--) {
        if (A[i] < minValue) {
            minValue = A[i];
        }
        if (A[i] > maxValue) {
            maxValue = A[i];
        }
        if (A[i] > 0) {
            res[i] = A[i] * maxValue;
        } else {
            res[i] = A[i] * minValue;
        }
    }
}
```

3）算法的时间复杂度为 $O(n)$，空间复杂度为 $O(1)$。

42. 题目详解：

1）首先计算最早发生时间 ve：

node	1	2	3	4	5	6	7
ve	0	7	2	5	12	6	9

然后计算最晚发生时间：

node	1	2	3	4	5	6	7
vl	0	10	2	5	12	8	9

ve 和 vl 相同的顶点为 1、3、4、7、5，所以关键活动为 a、e、m、n。

2）e 的执行时间是 2~5，可以与 e 同时执行的活动为 b、d、c。

3）比较各个顶点的 ve 和 vl，可知顶点 2 的最早发生时间和最晚发生时间相差最大，时间余量为 3。

4）为保证 $1 \rightarrow 2 \rightarrow 5$ 的路径长度不超过关键路径的长度，需要保证 $6+b+k \leq 12$，所以 $b \leq 4$，即 b 的持续时间最长为 4。如果想要保证 b 仍然为 5 的话，需要压缩 k 的时间长度，将 k 缩小为 1。

43. 题目详解：

1）cache 块大小和主存块大小一致为 64B，所以块内偏移为 6 位（$2^6 = 64$）。

cache 块的数量为 32KB / 64B = 512，由于采用 8 路组相连，所以总共有 512 / 8 = 64 个组，组号占 6 位（$2^6 = 64$）。

所以第 6 位到第 11 位可以作为 Cache 索引（用来定位地址可能被哪一组所缓存）。

2）&d[100] = d + 100 * 4 = 0x0180 0020 + 400 = 0x0180 01B0，对应的二进制为 0000 0001 1000 0000 0000 0001 1011 0000，组号为 000110B 即 6。

3）物理地址的结构为：页内偏移 12 位，物理页号 20 位，所以 &d[0] 的页内偏移为 020H = 32。 数组总共需要用 2048 * 4 / 64 = 128 个完整 cache 块存储，但由于数组第一个元素处于某个 cache 的中间（偏移为 32），所以总共需要 129 个 cache 块存储，在访问每个 cache 块中的第一个元素时会发生 Cache 缺失。

缺失率 = 129 / 2048 = 6.3%。

数组的平均访问时间为 = (129 * 200 + (2048 - 129) * 2) = 14.47 个时钟周期。

4）数组 d 需要占用 8KB 的存储空间，占用两个页面。由于数组的起始地址处于页面内部（偏移 32B），所以数组 d 分布在三个页面中，触发的缺页次数为 3。

44. 题目详解：

1）R 中的值为 $0xffffffff$，Q 中的值为 $0x87654321$，Y 中的值为 $0xffffffffff$。b 中的控制逻辑包含计数器，ALUop 所控制的 ALU 运算包含加法和减法。

2）第一种情况除数为 0 异常，d[i] 为任意值，x 为 $0x00000000$。第二种情况溢出异常，d[i] 为 $0x80000000$，x 为 $0xffffffff$，在 d[i] = -2^{31}、x = -1 的情况下会发生溢出异常。

在发生除法异常时 CPU 响应的操作：

1. 关中断，修改 CPU 状态为内核态。
2. 保存断点（PC 和 PSWR 中的值）。
3. 跳转到异常处理程序。

45. 题目详解：

这题是一个近似于流水线的结构，其过程为：挖树坑（甲）→ 放树苗、填土（乙）→ 浇水（丙）。不过甲最多可以同时挖三个树苗，也就是说不允许同时存在 4 个未被乙使用的树坑，这是比较复杂的一点。

实现甲和乙之间的同步需要使用到 pits 和 empty 这两个信号量，同时还需要一个 water 信号量来实现乙和丁的同步，代码实现如下：

```
semaphore mutex = 1;  // 对铁锹的使用需要互斥
semaphore pits  = 3;  // 甲还能挖洞的数量
semaphore empty = 0;  // 可以使用的树坑数量
semaphore water = 0;  // 需要浇水的水苗数量

甲() {
    while (1) {
        wait(pits);     // 最多只能挖三个未被乙使用的坑
        wait(mutex);    // 占用铁锹
        挖树坑;
        signal(mutex);  // 释放铁锹
        signal(empty);  // 通知乙可以放树苗和填土了
    }
}

乙() {
```

```
    while (1) {
        wait(empty);    // 等待到有树坑为止
        wait(mutex);    // 占用铁锹
        放树苗、填土;
        signal(mutex);  // 释放铁锹
        signal(pits);   // 通知甲可以继续挖坑了
        signal(water);  // 通知丙可以浇水了
    }
}

丙() {
    while (1) {
        wait(water);
        浇水;
    }
}
```

46. 题目详解:

题目中给出的内存结构和标准 linux 进程结构由些许差异，主要不同点在于图中的 读/写代码段 将 .bss 和 .data 融合了，考生看到能够理解就可以。

1）进程管理属于操作系统提供的功能，所以 PCB（进程）位于内核区，执行 $scanf()$ 时，进程在等待键盘 I/O，处于阻塞态。

2）$main()$ 函数的代码位于只读代码段（.text），其直接调用的 $scanf()$ 和 $printf()$ 需要执行驱动程序。

3）ptr 是作为全局变量定义的，所以其位于读/写数据段，$length$ 变量在 $main$ 函数中定义，如果该变量不在寄存器中被分配的话，那么就位于用户栈段，ptr 指针指向的内存单元是使用 $malloc$ 函数动态分配的，位于堆区。

47. 题目详解:

（1）轨道高度 36000km，电磁波速度 300000km/s，单向传播时延为 $36000/300000 = 0.12\text{s} = 120\text{ms}$。
最大吞吐量为卫星链路的数据传输率 200kbps。
传输 4000B 文件的时间 = 传输时间 + 传播时间 = $(4000 \times 8)/200000 + 0.12 = 0.16 + 0.12 = 0.28\text{s} = 280\text{ms}$。

（2）SLP 数据帧长 1500B，传输时间为 $(1500 \times 8)/200000 = 0.06\text{s} = 60\text{ms}$。
往返时间 $\text{RTT} = 240\text{ms}$。
信道利用率 = 窗口大小×传输时间 /(传输时间 + RTT) ≥ 0.8
窗口大小 $\geq 0.8 \times (60 + 240)/60 = 4$
因此发送窗口至少为 4。
SLP 帧序号至少需要能表示窗口大小的两倍，即 8，所以需要至少 4 位（$2^4 = 16 > 8$）。

（3）工程部分配 IP 地址空间 $10.10.10.0/24$，划分为 3 个子网:
・生活子网不少于 120 个：需要 7 位主机位（$2^7 - 2 = 126$），子网掩码 $/25$，子网地址 $10.10.10.0/25$
・作业子网不少于 60 个：需要 6 位主机位（$2^6 - 2 = 62$），子网掩码 $/26$，子网地址 $10.10.10.128/26$
・管理子网不少于 60 个：需要 6 位主机位（$2^6 - 2 = 62$），子网掩码 $/26$，子网地址 $10.10.10.192/26$

答案汇总

(1) 单向传播时延为 120ms，最大吞吐量为 200kbps。上传文件的时间至少为 280ms。

(2) 发送窗口至少为 4，帧序号至少为 4。

(3) 作业区子网地址: 10.10.10.128/26

管理区子网地址: 10.10.10.192/26

生活区子网地址: 10.10.10.0/25