计算机原理 C 卷

一、 选择题 (3 * 10 = 30 分)

1. **关于 IEEE 754 浮点数表示，下列说法中 错误 的是：**
   - A. 非规格化数 (Denormalized numbers) 的主要作用是处理“渐进式下溢”，使得数值能平滑地趋向于 0。
   - B. 在 32 位单精度浮点数中，阶码 (Exponent) 采用 127 的偏置常数，因此阶码字段全 0 表示的指数是 -127。
   - C. 浮点数加法不满足结合律，即 $(a + b) + c$ 不一定等于 $a + (b + c)$，这是由舍入误差导致的。
   - D. 浮点数 0.0 有两种表示：正零 (+0.0) 和负零 (-0.0)，它们在位模式上是不同的。
2. **考虑一个经典的 5 级流水线（取指、译码、执行、访存、写回）。若发生“数据冒险” (Data Hazard)，下列哪种技术不能消除流水线停顿 (Stall)：**
   - A. 转发 (Forwarding/Bypassing) 技术，将执行结果直接送往后续指令的输入。
   - B. 分支预测 (Branch Prediction) 技术。
   - C. 编译器的指令调度 (Code Reordering)。
   - D. 加载-使用冒险 (Load-use Hazard) 时的硬件阻塞。
3. **在现代 Linux 系统中，关于信号 (Signal) 处理的说法 正确 的是：**
   - A. 信号处理程序中调用 `printf()` 是安全的，因为它是标准库函数。
   - B. 正在处理某个信号 SIGINT 时，内核会自动阻塞后续到达的相同类型信号 SIGINT。
   - C. `sigprocmask` 函数的主要作用是删除系统中已经挂起的信号。
   - D. 信号从发出到被进程接收是同步的，进程会在发出信号的瞬间立即跳转到处理程序。
4. **关于 RAID 磁盘阵列，下列说法 错误 的是：**
   - A. RAID 0 提高了读写吞吐量，但没有任何冗余，任何一块磁盘损坏都会导致数据丢失。
   - B. RAID 1 采用镜像技术，磁盘利用率为 50%，具有极高的可靠性。
   - C. RAID 5 采用分布式奇偶校验，允许最多两块磁盘同时损坏而不丢失数据。
   - D. RAID 10 是先做镜像再做条带化，结合了 RAID 1 的可靠性和 RAID 0 的性能。
5. **在 C 语言中，若一个符号在多个目标文件中被定义，链接器处理“弱符号 (Weak Symbol)”的规则是：**
   - A. 允许存在多个同名的强符号。
   - B. 若有一个强符号和多个弱符号，链接器选择强符号。
   - C. 若有多个弱符号且没有强符号，链接器会报错。

- D. 函数名通常被视为弱符号，而未初始化的全局变量被视为强符号。

6. **高速缓存 (Cache) 的一致性协议 (如 MESI) 主要解决了什么问题：**
   - A. 虚拟地址到物理地址的转换冲突。
   - B. 磁盘与主存之间的数据交换速度不匹配。
   - C. 多核处理器中，不同核心的私有 Cache 对同一内存块的数据副本不一致问题。
   - D. 指令 Cache 和数据 Cache 的容量分配问题。
7. **考虑虚拟内存系统，若页面置换算法采用 LRU (最近最久未使用)，在有限的物理页帧下发生“抖动” (Thrashing) 的主要原因是：**
   - A. 物理内存远大于程序所需的局部工作集。
   - B. 程序的空间局部性太好，导致频繁命中。
   - C. 进程频繁访问的页面数超过了分配给它的物理页帧数。
   - D. CPU 主频过高，导致内存访问压力过大。
8. **在 x86-64 架构中，关于 switch 语句的编译优化，说法正确的是：**
   - A. 编译器总是将 switch 翻译成一连串的 if-else。
   - B. 当 case 值比较密集时，编译器通常会生成“跳转表 (Jump Table)”，使跳转复杂度为 O(1)。
   - C. switch 语句的执行效率总是低于 if-else 链。
   - D. 跳转表存储在程序的栈 (Stack) 空间中。
9. **下列关于局部性原理 (Locality) 的描述，错误的是：**
   - A. 步长为 1 的数组访问具有良好的空间局部性。
   - B. 循环体内的指令访问通常具有良好的时间局部性。
   - C. 在多维数组中，按列优先顺序访问按行存储的 C 语言数组，空间局部性极佳。
   - D. 局部性原理是计算机层次化存储结构 (Cache、内存、磁盘) 能够高效工作的理论基础。
10. **当一个进程执行 `longjmp` 函数时，会发生以下哪种情况：**
    - A. 程序立即终止并清理所有资源。
    - B. 类似于 return，但可以跨越多个函数调用栈帧直接跳转回 `setjmp` 设定的位置。
    - C. 它会保存当前的寄存器状态到内存中，以便后续恢复。
    - D. 它只能在同一个函数内部跳转，类似于 `goto`。

二、 填空题 (4 * 4 = 16 分)

1. 在一个 64 位机器上，假设整型 `int` 占用 4 字节。已知十六进制数 `x = 0x80000000`，在 C 语言中将其赋值给 `int` 型变量后，执行表达式 `x >> 1` (算术右移)，得到的结果的十六进制表示为：__。

2. 假设某 CPU 的 L1 数据 Cache 为 32KB，8 路组相联 (8-way set associative)，块大小为 64 字节。则该 Cache 共有 ____ 组 (Sets)，物理地址中用于表示“组索引 (Index)”的位数是 ____ 位。
3. 在静态链接过程中，`.bss` 节 (Section) 通常用于存放 ____ 的全局变量。这类变量在目标文件中不占用实际磁盘空间，仅在运行时由内核初始化为 0。
4. 网络编程中，服务器端在调用 `bind()` 绑定端口后，需要调用 ____ 系统调用将套接字从主动状态转变为被动状态，准备接收客户端连接。

三、 解答题 (50 分)

15. (8分) 浮点数表示与转换

考虑一个 8 位浮点数格式，遵循 IEEE 754 标准：符号位 1 位，阶码 4 位（偏置常数 Bias = 7），尾数 3 位。

(1) 计算该格式下能表示的最大规格化正数（给出十进制值）。

(2) 将十进制数 -1.25 转换为该 8 位格式的位模式。

16. (12分) 汇编逆向分析与栈帧结构

观察以下 x86-64 汇编代码（由 GCC 生成）：

Code snippet:

代码段

```
    # long func(long n)
    # n in %rdi
func:
    cmpq    $1, %rdi
    jg      .L2
    movl    $1, %eax
    ret
.L2:
    pushq   %rbx
    movq    %rdi, %rbx
    leaq    -1(%rdi), %rdi
    call    func
    imulq   %rbx, %rax
    popq    %rbx
    ret
```

(1) 补全对应的 C 语言代码。

(2) 寄存器 %rbx 在此处起什么作用？为什么要进行 pushq 和 popq 操作？

(3) 该函数是否存在栈溢出的风险？在什么情况下会发生？

17. (10分) 多级页表高级计算

在一个 64 位虚拟地址空间的系统中：

- 页大小为 8KB。
- 页表条目 (PTE) 大小为 8 字节。
- 采用 4 级页表结构。

(1) 虚拟地址中，页偏移 (Page Offset) 占用多少位？

(2) 为了使每一级页表恰好能装在一个物理页面内，每一级索引应该占用多少位？该系统能支持的最大有效虚拟地址位数是多少？

18. (10分) Cache 访问模拟

假设一个 2 路组相联 Cache，总容量 128 字节，块大小 16 字节，采用 LRU 替换策略。地址宽度为 8 位。

请分析以下地址序列的访问情况（命中/不命中）：0x10, 0x14, 0x50, 0x18。

(1) 写出每个地址的索引 (Index) 和标记 (Tag)。

(2) 逐步说明 Cache 的内容变化及最终命中情况。

19. (10分) 系统级编程：信号与竞争冒险

考虑以下代码片段：

```c
volatile sig_atomic_t flag = 0;
void handler(int sig) {
    flag = 1;
}

int main() {
    signal(SIGUSR1, handler);
    while (!flag) {
        /* 期待在这里等待信号 */
    }
    printf("Signal received!\n");
    return 0;
}
```

(1) 上述代码在逻辑上存在什么潜在的严重问题（提示：原子性与竞争）？

(2) 如何使用 `sigsuspend` 来修正这个问题？请简述理由。