清华大学本科生考试试题专页纸	
考试课程：操作系统（A卷）	时间：2023年04月17日上午09:50~12:15

系别：_____	班级：_____	学号：_____	姓名：_____

答卷注意事项:	1. 答题前，请先在试题纸和答卷本上写明A卷或B卷、系别、班级、学号和姓名。
	2. 在答卷本上答题时，要写明题号，不必抄题。
	3. 答题时，要书写清楚和整洁。
	4.请回答所有题目。本试卷有1个判断题，8个简答题，共6页。
	5. 考试完毕，必须将试题纸和答卷本一起交回。

一、(30分)判断题

1. [ ] 面向手机的Android是一种微内核架构的操作系统内核。
2. [ ] UNIX操作系统支持以抢占方式来分时复用。
3. [ ] shell命令行程序是通过系统调用来获取用户敲击的字符命令。
4. [ ] 操作系统需要建立和设置好其正常运行所需的堆空间和栈空间。
5. [ ] 本课程实验中的QEMU启动后，CPU将直接跳转到内核入口地址。
6. [ ] linker（链接器）的主要工作是把多个机器码目标文件合并为单个机器码执行文件。
7. [ ] LFU和CLOCK算法都会有Belady异常现象。
8. [ ] 本课程实验中的操作系统内核的二进制代码中包含数据段、代码段和堆栈。
9. [ ] 本课程实验中的操作系统内核的BSS段是指用来存放未初始化的全局变量的一块内存区域，内核执行初始化时需要把BSS段清零。
10. [ ] 本课程实验中的操作系统内核在内核态（S-Mode）处理中断、异常和系统调用。
11. [ ] 本课程实验中，用户态(U-Mode)执行的应用无法执行特权级指令，从而避免了操作系统内核的代码和数据被应用直接访问到。
12. [ ] 本课程实验中，用户态(U-Mode)执行的应用可以通过设置相关CSR寄存器来使能页机制。
13. [ ] 某进程执行exec系统调用时，会出现其执行状态从运行态转到创建态的情况。
14. [ ] 在没有页机制且支持多道程序的操作系统中，不同程序的入口地址要不同且有一定距离，以避免程序之间相互覆盖。
15. [] 本课程实验中，运行的程序从用户态切换到内核态时，除了需要保存所有的通用寄存器外，还要保存部分CSR寄存器。

二、(70分)简答题

1.（6分）

1. 请描述先来先服务（FCFS）调度算法的基本思路和优缺点。
2. 请描述短作业优先（SJF）调度算法的基本思路和优缺点。
3. 请问短作业优先调度算法是否具有最小平均周转时间？请说明理由或给出证明。

2.（6分）

内存目前有5个空闲块，按地址从小到大排列，分别为 16K，4K，10K，48K，12K。现有三个内存分配请求，要求按顺序分配4K，15K，40K三块内存。试分别描述在最先匹配(First-fit)、最佳匹配(Best-fit)、最差匹配(Worst-fit)三种策略下，每个分配请求是否成功，如成功，使用哪个空闲块进行分配。(注：回答的字数在120字以内)

3. (8分)

在实验环境搭建过程中，我们需要安装QEMU软件来模拟一台计算机，并会执行如下的代码，来让QEMU模拟加电启动虚拟计算机，运行操作系统和应用：

```
qemu-system-riscv64 -machine virt -nographic -bios ../bootloader/rustsbi-qemu.bin \
  -device loader,file=target/riscv64gc-unknown-none-elf/release/os.bin,addr=0x80200000
```

或

```
qemu-system-riscv64 -nographic -machine virt -bios ./bootloader/rustsbi-qemu.bin \
  -kernel build/kernel
```

并可得到类似如下内容：

```
.______       __    __      _______.___________.  _______..______   __
|   _  \     |  |  |  |     /       |           | /       ||   _  \ |  |
|  |_)  |    |  |  |  |    |   (----`---|  |----`|   (----`|  |_)  ||  |
|      /     |  |  |  |     \   \       |  |      \   \    |   _  < |  |
|  |\  \----.|  `--'  |.----)   |      |  |  .----)   |   |  |_)  ||__|
| _| `._____| \______/ |_______/       |__|  |_______/    |______/ (__)

[rustsbi] Implementation: RustSBI-QEMU Version 0.0.2
[rustsbi-dtb] Hart count: cluster0 with 1 cores
[rustsbi] misa: RV64ACDFIMSU
[rustsbi] mideleg: ssoft, stimer, sext (0x222)
[rustsbi] medeleg: ima, ia, bkpt, la, sa, uecall, ipage, lpage, spage (0xb1ab)
[rustsbi] pmp0: 0x10000000 ..= 0x10001fff (rwx)
[rustsbi] pmp1: 0x80000000 ..= 0x8fffffff (rwx)
[rustsbi] pmp2: 0x0 ..= 0xffffffffffffff (---)
qemu-system-riscv64: clint: invalid write: 00000004
[rustsbi] enter supervisor 0x80200000
Hello, world!
.text [0x80200000, 0x80202000)
```

请回答如下实验相关问题：（注：每小问回答的字数在20字以内）

1. 请问QEMU模拟的是哪种CPU？该CPU具有哪些硬件属性？

2. QEMU在模拟计算机加电启动执行指令前，把哪些内容加载到它模拟的物理内存中了？
3. QEMU从加电启动到执行操作系统的第一条指令的时间段中，是否出现了特权级切换和进程（任务）切换的情况？
4. 操作系统要执行的第一条指令的物理地址是？

4. (6分)

1. 请描述中断和异常的定义、特征和差别？
2. 请描述操作系统内核对中断和异常的处理过程。
3. 请结合第三章（lab1）课程实验的具体实现描述操作系统内核对中断和异常的特权级切换过程。

5. (8分)

如下的 C 应用程序和Rust 应用程序，可分别运行在第四章（lab2）的uCore和rCore中。请请根据你做的具体实验回答如下问题：

1. 应用程序的输出结果是什么？
2. 应用程序会发出哪些系统调用？
3. 以第四章的uCore或rCore为具体例子，请描述QEMU启动后，操作系统需要完成哪些的自身初始化工作。
4. 以第四章（lab2）的uCore或rCore为具体例子，请描述应用程序在开始执行前，操作系统需要为应用做哪些准备工作。

```c
#include <stdio.h>
#include <unistd.h>
int main() {
   puts("Hello world");
   return 0;
}
```

```rust
#![no_std]
#![no_main]
#[macro_use]
extern crate user_lib;
#[no_mangle]
fn main() -> i32 {
    println!("Hello world");
    0
}
```

6. (8分)

请回答如下在第四章（lab2）的实验相关问题：（注：每小问回答的字数在50字以内）

1. 刷新 TLB 的指令是否是特权指令？
2. 一般在执行哪些操作后，需要执行刷新 TLB 的指令?
3. 更换页表之后需要刷新数据cache 和指令cache吗? 请说明原因。
4. 在内核页表和应用的用户页表中都存在同样虚拟地址的物理页主要包含了哪些内容。这些物理页的作用是什么？
5. 如何只通过sscratch寄存器中转栈指针和页表基址？

7. (7分)

在第四章“地址空间”的实验（lab 2）中，操作系统有了对特权级的支持和页机制的支持。stvec CSR寄存器保存了在产生中断、异常或系统调用时CPU要跳转并执行的地址。如果应用程序运行在用户态时产生了异常，系统调用或中断，导致RISC-V 64 CPU从用户态陷入到了内核态。请问：

(注：每小问回答的字数在40字以内)

1. 如果有特权级机制，用户态的应用程序执行特权指令后会产生异常，CPU硬件会做哪些处理，并跳转到哪个地址执行？
2. 如果有特权级但没有分页机制，用户态的应用程序是否可以直接读写内核态中操作系统的代码或数据？
3. 在RVSC-V中，用户态应用程序通过什么指令实现系统调用并陷入到内核态？
4. 在RVSC-V中，内核通过什么指令实现返回到用户态进程？
5. 如果用户态应用程序通过系统调用让CPU跳转到这个地址并开始准备执行第一条指令时，产生了一个时钟中断。请问CPU接下来要做什么事情？

8. (6分)

以下是 uCore/rCore lab2 的陷入处理函数的一部分，请按自己选择的实验作答：

1. 代码中 “Question 1” 一句起什么作用？如果去掉会发生什么？
2. “Question 2” 中的 `scause` 寄存器包含了什么信息？
3. “Question 3”一句起什么作用？如果去掉会发生什么？

```
#[no_mangle]
pub fn trap_handler() -> ! {
    set_kernel_trap_entry(); // Question 1
    let cx = current_trap_cx();
    let scause = scause::read(); // Question 2
    let stval = stval::read();
    match scause.cause() {
        Trap::Exception(Exception::UserEnvCall) => {
            cx.sepc += 4; // Question 3
            cx.x[10] = syscall(cx.x[17], [cx.x[10], cx.x[11],
cx.x[12]]) as usize;
        }
        ......
```

```
void usertrap()
{
         set_kerneltrap(); // Question 1
         struct trapframe *trapframe = curr_proc()->trapframe;
         ......
         uint64 cause = r_scause(); // Question 2
         if (cause & (1ULL << 63)) {
                  ......
         } else {
                  switch (cause) {
                  case UserEnvCall:
                           trapframe->epc += 4; // Question 3
                           syscall();
                           break;
                  ......
```

9. (15分)

在13个时间单位的时间内，一个程序的虚拟页面访问顺序如下：

1	2	3	4	5	6	7	8	9	10	11	12	13
e	d	a	c	c	d	B	c	e	c	e	a	d

操作系统给这个程序分配了5个物理页帧，初始物理页帧为空。请回答如下问题：

1. 请描述 LRU、FIFO、Clock、工作集、缺页率页面置换算法的基本思路和特征。
2. 为何 LRU 算法难以在实际操作系统中实现？
3. 为何 LRU 算法不会出现 Belady 现象？
4. 如采用工作集页面置换算法，工作集窗口 Δ 的大小 τ = 4，请问在哪些时刻会产生缺页异常，缺页异常次数总共是多少？
5. 如采用缺页率置换算法，容忍的缺页窗口 T=2，请问在哪些时刻会产生缺页异常，缺页异常次数总共是多少？
6. 如果要在 Lab2 中实现 Clock 页面置换算法。请说明 Clock 页面置换算法的执行时机？请描述 Clock 页面置换算法会对页表项中的哪些位做访问操作？这些位起到的作用是什么？这些位会在哪些时机被改写？