CS3601 操作系统

测验试卷

1. 以下为寄存器的初始值及按顺序执行的指令，请将表格补充完整。（前序指令的执行结果会影响后续指令执行）（5 分）

寄存器	初始值
X0	0xffffffff00000000
X1	0x0
X2	0x0

指令	目标寄存器	结果
`asr x0, x0, #0x11`		
`mvn x1, x1`		
`eor x2, x0, x1`		
`add x2, x0, x2`		
`lsr x2, x2, #6`		

2. 不使用栈，将以下函数转换成 ARM64 汇编。（5 分）

（参数 x 通过 x0 寄存器传参；函数返回结果需放在 x0 寄存器中；无需考虑函数调用过程中的寄存器保存/恢复等栈相关操作）

```
int64_t sum(int64_t x) {
    int64_t ret = 0;
    for (int64_t i = 1; i <= x; i++) {
        ret += x;
    }
    return ret;
}
```

3. 阅读以下汇编函数，回答问题：

```
foo:
    stp x19, x20, [sp, #-16]!
    mov x19, #0
    mov x20, #0

loop:
    cmp x20, x2
```

```
    bge done

    ldr x5, [x0, x20, lsl #3]
    ldr x6, [x1, x20, lsl #3]

    mul x5, x5, x6
    add x19, x19, x5

    add x20, x20, #1

    b   loop

done:
    mov x0, x19
    ldp x19, x20, [sp], #16
    ret
```

1) 请分析函数 foo 的作用，参数个数，每个参数的意义。（3 分）
2) 为什么使用 x19 和 x20 寄存器的时候需要保存在栈上，而其他寄存器不需要？（2 分）

4. 局部变量在栈上的顺序和函数调用参数入栈的顺序分别是什么？原因分别是什么？(4 分)

5. 在函数调用过程中：
   1) X0 寄存器是调用者保存还是被调用者保存？为什么？（2 分）
   2) 一个函数如果想要将多个变量作为返回值应该怎么办？给出两种办法。（2 分）

6. 特权级切换时已有硬件保存上下文，为什么还需要软件保存/恢复上下文？给出两个理由。（4 分）

7. 操作系统通过系统调用允许应用调用部分系统能力，请回答以下问题：
   (1) 简述 Flex-SC 执行一个系统调用的流程。（3 分）
   (2) 适合用 Flex-SC 优化的系统调用有哪些特点？至少写出两种。（2 分）
   (3) 什么样的系统调用不适合使用 Flex-SC 优化？给出两种类型。（2 分）
   (4) vDSO 通过将系统调用逻辑放入用户态进行系统调用加速，请分别给出 1 个适合与不适合用 vDSO 加速的系统调用，并分别说明原因。（5 分）

8. 阅读下面的 C 代码，回答问题：

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>

int counter = 1;

int main() {
    int status = 0;
    counter++;
    if (fork() == 0) {
        counter--;
        printf("%d\n", counter);
        return 0;
    }
    counter++;
    waitpid(-1, &status, 0);
    counter++;
    printf("%d\n", counter);
    return 0;
}
```

1) 请写出上述程序所有可能的输出结果。（2 分）

2) 将上述程序编译为 a.out，在 Linux shell 执行./a.out 运行程序。请从 shell 视角介绍，程序是如何被加载的?（哪个进程调用哪个系统调用，每个系统调用的功能/目的是什么）（4 分）

3) 上述程序运行起来后，何时会发生用户-内核上下文切换？请给出至少两种情况。（2 分）

9. 阅读下面的 C 代码，回答问题：

```c
#inc1ude <stdio.h>
#include <unistd.h>
int main()
{
    for (int i = 0; i < 2; i++) {
        fork();
        printf("A\n");
    }

    return 0;
}
```

1) 上面这段程序会输出几个 A？为什么？（3 分）

2) 如果将上述代码的 printf(“A\n”) 改为 printf(“A”)，程序会输出几个 A？为什么？（提示：默认情况下，printf 的输出内容会缓冲在内存中，不会立即输出到屏幕，直到遇到换行符、显式调用 ff1ush 或程序退出）（3 分）

10. 阅读下面的 C 代码，回答问题：

```
#inc1ude <std1ib.h>
#include <stdio.h>

int a;
char b[] = "hel1o";
const char* c;
int const d = 2;

void foo() {
    c = "hello world";
    printf("a: %d\n", a);
    printf("b: %s\n", b);
    printf("c: %s\n", c);
    printf("d: %d\n", d);
}

int main() {
    foo();
    return 0;
}
```

1) 上述代码编译为 ELF 格式文件，a，b，c，d 分别位于 ELF 文件的哪个节中？（4 分）

2) printf(“a:%d\n”, a) 的输出结果是什么？为什么？（3 分）

3) 上述代码并没有实现 printf 函数，为什么可以在运行时调用到 printf 函数？（2 分）

4) 为什么 Linux 进程地址空间需要区分代码段和数据段？（4 分）

11. 同学们在学习了操作系统课程后，对写代码产生了浓厚的兴趣，编写了各种各样的程序。

1) 在 linux 系统中运行如下程序，会导致电脑失去响应吗？为什么？如果会，有什么办法防范这种恶意程序？（3 分）

```
int main() {
    while (1);
    return 0;
}
```

2) 在 linux 系统中运行如下程序，会导致电脑失去响应吗？为什么？如果会，有什么办法防范这种恶意程序？（3 分）

```
#include <unistd.h>

int main() {
    while (1)
        fork();
    return 0;
}
```

3) 有同学在 Linux 系统中运行如下程序，一段时间后发现电脑变慢了，他 Ctr1+C 终止该程序后，发现电脑很快又恢复正常。请写出电脑变慢和恢复正常的具体原因。（4 分）

```
#include <unistd.h>
#inc1ude <stdlib.h>

int main() {
    while (1) {
        if (fork() == 0) {
            exit(0);
        }
        sleep(1); // 程序暂停1秒后继续执行
    }
    return 0;
}
```

12. 在 AArch64 中，如何根据页表项判断 PFN 是否指向大页？（3 分）

13. 有一个 64 位系统，支持 5 级页表，每级页表均支持大页。其虚拟地址索引和偏移量位域如下所示：

虚拟地址索引	偏移量位域
第 1 级	56 ~ 48
第 2 级	47 ~ 39
第 3 级	38 ~ 30
第 4 级	29 ~ 21
第 5 级	20 ~ 12
页偏移量	11 ~ 0

1) 翻译一个虚拟地址所需要的页表大小是多少？（3 分）5 级页表架构能够表示的地址范围是多少？（3 分）第 1 级页表项指向的大页的大小为多少？（3 分）

2) 如果我们希望引入一个 1G 的大页，那么我们应当在哪一级页表引入块描述符？（3 分）

3) 为描述 1536MB 的空间，如果只采用 4KB 页面，那么最少需要多少个页表页？（4 分）如果允许使用大页，那么最少需要多少个页表页？（3 分）

4) 若某 TLB 中最多缓存 64 个 TLB Entry。在每个 Entry 都 valid 的情况下，请计算最多有多大的地址空间的地址翻译可以被缓存？（3 分）最少有多大的地址空间的地址翻译被缓存？（3 分）

14. 64 位操作系统中，OS 往往在启动时通过直接映射（Dircct Mapping）把所有物理内存映射到内核虚拟地址空间，请回答：页表直接映射的好处是什么？（6 分）

15. 请回答下列问题：

1) AArch64 采用基于 4 级页表的虚拟内存机制，每个页表页大小为 4KB，包含 512 个页表项，支持 4KB、2MB、1GB 的页面大小。请问如果考虑全部使用 2M 大页，翻译 3000 个 2M 虚拟页所需要的最小页表大小是多少？（2 分）

2) Linux 操作系统利用 VMA 结构体记录进程已分配的虚拟内存区域，请问：应该使用什么数据结构组织管理 VMA，并简单阐述原因（2 分）

3) 进程创建时，操作系统会为用户栈空间分配虚拟内存区，Linux 初始栈大小通常为 8MB。然而，栈空间无法一次性完全分配，其大小在运行时动态增长（例如递归调用或分配大量的局部变量），可能会超出初始栈的空间，请问：操作系统如何在运行时检测栈空间增长的需求并管理相应的 VMA？（3 分，言之有理即可）

4) 如果应用程序访问虚拟地址时触发缺页异常，请问可能是由哪些原因造成的？操作系统如何区分不同的原因？（6 分）

16. 请回答下列问题：

1) 在执行第 3 行代码时，0S 如何判断应用程序访问的是非法虚拟地址？（3 分）

```
1  int main() {
2      char *ptr = (char *) 0x0;
3      *ptr = 'a';
4  }
```

2) 在执行 fork 系统调用后, 子进程会复制一整份父进程的地址空间，操作系统利用写时拷贝 (copy-on-write) 实现 fork 性能加速，请阐述如何实现写时拷贝（3 分）

3) 换页机制 (swapping) 使我们能够突破物理内存容量限制，把磁盘空间当成内存空间使用，但这也使得缺页异常 + 磁盘操作导致访问延迟增加。请问如何降低换页机制带来的访问延迟开销？（3 分）

17. AArch64 的虚拟内存机制支持包括 4KB 在内的不同的页面大小，通常采用 4 级页表，请回答以下问题：

1) AArch64 中的 ttbr0_el1 和 tbr1_el1 的作用是什么？对于一个虚拟地址硬件如何决定采用哪一个寄存器指向的页表进行地址翻译？（4 分）

2) AArch64 通常支持 4K、2M、1G 三种不同大小的页面，为何页面大小每增加一次是原来的 2^9 倍？使用大页的好处是什么？（4 分）

3) 如果映射虚拟地址地址范围 0x00000000~0x01000000，则每一级页表分别需要多少个页表页（仅考虑 4K 页）?（4 分）

18. 物理内存管理会涉及伙伴系统以及 SLAB 分配器，请回答以下问题：

1) 一个大小为 32K 的物理块，其物理地址为 0x20000000，则其伙伴块的物理地址是多少？（3 分）

2) 操作系统中，伙伴系统和 SLAB 分配器的作用有什么区别？两者在操作系统运行过程中是如何配合的？（3 分）

19. 系统采用**时钟页置换算法**，假设系统共有 4 个物理内存页，且物理内存页初始状态均为空闲。进程 P 访问页号的序列为 0、1、2、7、0、5、3、5、0、2、7、6，请列出**每次访存**后物理内存中的数据页内容（12 分），并给出上述访存序列一共会产生多少次缺页异常？（3 分）

访存：	0	1	2	7	0	5	3	5	0	2	7	6
访存后的物理内存数据												
缺页												