• Don’t forget to write your name on this exam.

• This is an open book, open notes exam. But no online or in-class chatting.

• Ask us if something is confusing in the questions.

• Organize your work, in a reasonably neat and coherent way, in the space provided. Work scattered all over the page without a clear ordering will receive very little credit.

• Mysterious or unsupported answers will not receive full credit. A correct answer, unsupported by explanation will receive no credit; an incorrect answer supported by substantially correct explanations might still receive partial credit.

• If you need more space, use the back of the pages; clearly indicate when you have done this.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Total:</td>
<td>76</td>
<td></td>
</tr>
</tbody>
</table>
1. OS Interfaces
   
   (a) (5 points) Here is a program that uses the UNIX system call API, as described in Chapter 0 of the xv6 book:

   ```c
   #include "param.h"
   #include "types.h"
   #include "user.h"
   #include "syscall.h"

   int main() {
      char * message = "aaa\n";
      int pid = fork();

      if(pid != 0){
         char *echoargv[] = { "echo", "Hello\n", 0 };

         message = "bbb\n";
         exec("echo", echoargv);
         write(1, message, 4);
      }
   }

   write(1, message, 4);
   exit();
   }

   Assume that fork() succeeds, that file descriptor 1 is connected to the terminal when the program starts, and echo program exists. What possible outputs this program can produce (explain your answer)?
(b) (10 points) Write a program that uses the UNIX system call API, as described in Chapter 0 of the xv6 book. The program forks and creates a pipeline of 10 stages. I.e., each stage is a separate process. Each two consecutive stages are connected with a pipe, i.e., the standard output of each stage is connected to the standard input of the next stage. Each stage reads a character from its standard input and sends it to the standard output. The last stage outputs the character it reads from the pipe to the standard output.
2. Basic page tables.

(a) (5 points) Alice wants to construct a page table that maps virtual addresses 0x8000000, 0x80001000 and 0x80002000 into physical addresses 0x0, 0x1000, and 0x2000. Assume that the Page Directory Page is at physical address 0x0, and the Page Table Page is at physical address 0x00001000 (which is PPN 0x00001).

Draw a picture of the page table Alice will construct (or alternatively simply write it down in the format similar to the one below):

Page Directory Page:
PDE 0: PPN=0x1, PTE_P, PTE_U, PTE_W
PDE 1: PPN=0x2, PTE_P, PTE_U, PTE_W
... all other PDEs are zero

The Page Table Page:
PTE 0: PPN=0x3, PTE_P, PTE_U, PTE_W
PTE 1: PPN=0x4, PTE_P, PTE_U, PTE_W
... all other PTEs are zero
3. Stack and calling conventions.

Alice developed an xv6 program that has a function `foo()` that is called directly from `main()`:

```c
int foo(char *p) {
    write(1, "hello\n", 6);
    foo(p);
    return 0;
}

int main() {
    char a[4];
    foo(a);
    exit();
}
```

(a) (5 points) How many times will she see “hello” on her screen? Justify your answer.
(b) (5 points) Now Alice changes her `main()` function like this

```c
static char b[8192];

int foo(char *p) {
    write(1, "hello\n", 6);
    foo(p);
    return 0;
}

int main() {
    char a[8192]
    foo(a);
    write(1, b, 10);
    exit();
}
```

She runs it again. How many times will she see “hello” on the screen (justify your answer).

4. Xv6 process organization.

In xv6, in the address space of the process, what does the following virtual addresses contain?

(a) (3 points) Virtual address 0x0

(b) (3 points) Virtual address 0x80100000

(c) (3 points) What physical address is mapped at virtual address 0x80000000
(d) (5 points) Bob looks at the implementation of the `clearpteu()` function in the xv6 kernel (see below). He is confused about the role of the `walkpgdir()` function.

```c
2021 void
2022 clearpteu(pde_t *pgdir, char *uva)
2023 {
2024     pte_t *pte;
2025
2026     pte = walkpgdir(pgdir, uva, 0);
2027     if(pte == 0)
2028         panic("clearpteu");
2029     *pte &= ~PTE_U;
2030 }
```

Can you explain Bob why `walkpgdir()` is needed here and what purpose it serves?
5. Protection and isolation

(a) (5 points) In xv6 all segments are configured to have the base of 0 and limit of 4GBs, which means that segmentation does not prevent user programs from accessing kernel memory. Nevertheless, user programs can’t read and write kernel memory. How (through what mechanisms) such isolation is achieved?

(b) (5 points) Imagine you plan to run xv6 on the hardware that is identical to x86, but does not provide support for paging. What changes you have to make to the xv6 kernel to make sure that the isolation and protection across the processes and between the process and the kernel is in place.
6. System calls

(a) (5 points) What is the purpose of the line 6138 in the listing below (sys_read() is the xv6 system call that reads data from a file)?

```c
6131 int
6132 sys_read(void)
6133 {
6134 struct file *f;
6135 int n;
6136 char *p;
6137
6138 if(argfd(0, 0, &f) < 0 || argint(2, &n) < 0 || argptr(1, &p, n) < 0)
6139     return 1;
6140 return fileread(f, p, n);
6141 }
```
7. Bob thinks that it’s ok to let user processes register interrupt handlers. He starts with a timer interrupt, i.e., he adds a new system call that takes a pointer to a function that the kernel adds to the IDT as the handler for the timer interrupt (vector32). The rest of the kernel stays unchanged (same fields in the IDT, same CS selector, same kernel stack in the TSS).

(a) (7 points) Bob implements his change and it even works! He sees that his timer interrupt handler is executed several times, but then the system crashes in a mysterious way. Explain why the system works initially, but crashes later?

(b) (7 points) Bob’s friend Alice who is a mature OS hacker tells him that his change is ultimately insecure and breaks isolation guarantees of the xv6 kernel? Can you explain what does Alice mean?
8. 238P organization and teaching
   (a) (3 points) If there is one single most important thing you would like to improve in the
       CS238P class, what would it be?