Name: _____ Entry number:_____

Instructions

- Verify that your exam has pages 1 to 10. Last two pages are the **Project Exam**.
- Please fill the name and entry number on *every page* front and back. Do this first *before* starting the exam. Please keep the institute ID card with you for attendance.
- During the exam, you should not have any electronic equipment such as laptop, mobile phone, calculator, tablets, etc.
- The exam is open notes, open book, open slides. You can keep the printouts with you during the exam.
- Please answer each question in the provided space.

Please sign below:

I will not give or receive aid in the examination. I accept that any act of mine that can be considered to be an *IITD Honour Code* violation will invite disciplinary action.

Signature: _____

Q1 [2 marks] Virtualization

Q1.1 (True/False) [1 mark] One can build a trap-and-emulate hypervisor for hardware that follows the Popek-Goldberg theorem.

_____

Q1.2 (True/False) [1 mark] Hypervisors built using the paravirtualization technique had performance overheads from doing dynamic binary translation.

_____

Q2 [2 marks] Memory model

Q2.1 (True/False) [1 mark] Let us say we run the following multi-threaded program.

CPU 0	**CPU 1**
`x := 1`	`y := 1`
`print(y)`	`print(x)`

Assume that initially x=0 and y=0. If the program prints 1 and 1, then we can infer that the hardware is sequentially consistent.

_____

Q2.2 (Multiple choice) [1 mark] Let us say we run the following multi-threaded program:

CPU 0	**CPU 1**
`lock addl $1, (%eax)`	`addl $1, (%ebx)`
`addl $1, (%ebx)`	`lock addl $1, (%eax)`

In both the threads, %eax points to variable x initialised to 0 and %ebx points to variable y also initialised to 0. Tick mark observable values after the program runs on x86.

a. x=2, y=2 b. x=1, y=2 c. x=2, y=1 d. x=1, y=1

Name: _____ Entry number:_____

Q3 [6 marks] Assembly

Following is an assembly program for a function “foo”:

```
1.    foo:
2.         pushl   %ebx
3.         movl    8(%esp), %ecx
4.         movl    12(%esp), %ebx
5.         movl    $1, %edx
6.         testl   %ecx, %ecx
7.         jle     .L4
8.         xorl    %eax, %eax
9.    .L3:
10.        addl    $1, %eax
11.        imull   %ebx, %edx
12.        cmpl    %eax, %ecx
13.        jne     .L3
14.   .L4:
15.        popl    %ebx
16.        movl    %edx, %eax
17.        ret
```

Q3.1 (Fill in the blanks) [4 marks] This program was compiled using the following C program. Complete the omitted sections of the C code to produce a program that is equivalent to the given assembly code. Note: x is at `8(%esp)` and y is at `12(%esp)`.

```
int foo(int x, int y) {
      int res = (i) _______________;
      for(int j = (ii) ____________; j < (iii) ____________; j++)
            res = (iv) _______________;
      return res;
}
```

Q3.2 [2 marks] Can we safely delete the assembly instructions “`2. pushl %ebx`” and “`15. popl %ebx`”? Why or why not?

Name: _____ Entry number:_____

Q4 [11 marks] Semaphores

Professor Utonium discovered the recipe for creating Powerpuff girls: Sugar, Spice, Everything Nice, and Chemical X. Due to the rise of evil forces, the government set up three factories to manufacture Powerpuff girls at scale. Factory 1 was given an infinite supply of Sugar, Factory 2 was given an infinite supply of Spice, and Factory 3 was given an infinite supply of Everything Nice. After new Powerpuff girls are created, the government watches the crime rate for some time and releases Chemical X and any two out of {Sugar, Spice, Everything Nice} to let factories create more Powerpuff girls.

We represent each factory and the government with separate threads. Powerpuff Girls (ppg), Sugar, Spice, Everything Nice (e_nice) and Chemical X (c_x) are represented using semaphores. Line numbers are added in each function to later ask questions.

```
#include <pthread.h>
#include <semaphore.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

sem_t sugar, spice, e_nice, c_x, ppg;
void* factory1(void* arg) {
1.    while(1) {
2.          sem_wait(&spice);
3.          sem_wait(&e_nice);
4.          sem_wait(&c_x);
5.          printf("Manufactured using Sugar from my supply.\n");
6.          sem_post(&ppg);
7.    }
}
void* factory2(void* arg) {
1.    while(1) {
2.          sem_wait(&e_nice);
3.          sem_wait(&sugar);
4.          sem_wait(&c_x);
5.          printf("Manufactured using Spice from my supply.\n");
6.          sem_post(&ppg);
7.    }
}
void* factory3(void* arg) {
1.    while(1) {
2.          sem_wait(&sugar);
3.          sem_wait(&spice);
4.          sem_wait(&c_x);
5.          printf("Manufactured using Everything Nice from supply.\n");
6.          sem_post(&ppg);
7.    }
}
```

Name: _____ Entry number:_____

```
void* govt(void* arg) {
1.    while(1) {
2.          sem_wait(&ppg);
3.
4.          printf("Crime has risen. Need more Powerpuff Girls!\n");
5.
6.          int r = rand() % 3;
7.          sem_post(&c_x);
8.          if(r != 0)
9.                sem_post(&sugar);
10.         if(r != 1)
11.               sem_post(&spice);
12.         if(r != 2)
13.               sem_post(&e_nice);
14.   }
}
int main() {
1.    pthread_t t[4];
2.    sem_init(&sugar, 0, 0);
3.    sem_init(&spice, 0, 0);
4.    sem_init(&e_nice, 0, 0);
5.    sem_init(&c_x, 0, 0);
6.    sem_init(&ppg, 0, 1);
7.
8.    pthread_create(&t[0], NULL, govt, NULL);
9.    pthread_create(&t[1], NULL, factory1, NULL);
10.   pthread_create(&t[2], NULL, factory2, NULL);
11.   pthread_create(&t[3], NULL, factory3, NULL);
12.
13.   for(int i = 0; i < 4; i ++)
14.         pthread_join(t[i], NULL);
15.   return 0;
}
```

Q4.1 [1 mark] How does the program behave if we delete lines 13 and 14 from main()?
Q4.2 [1 mark] Which thread will get to its print statement first? Justify your answer.

Q4.3 [2 marks] Precisely specify a sequence of events that can lead to a deadlock.

Q4.4 [3 mark] Make changes to the program that keeps the intended behaviour but does not get into deadlocks. To make changes, you cross out different lines and add new lines to the given program.

Q4.5 [4 marks] Even if we change line 6 in the function main to `sem_init(&ppg, 0, 2);` in the original program, the program gets into a deadlock. Specify the sequence of events that lead this modified program to deadlock. Hint: try to expand this table containing values in each semaphore with a sequence of events that leads to deadlock.

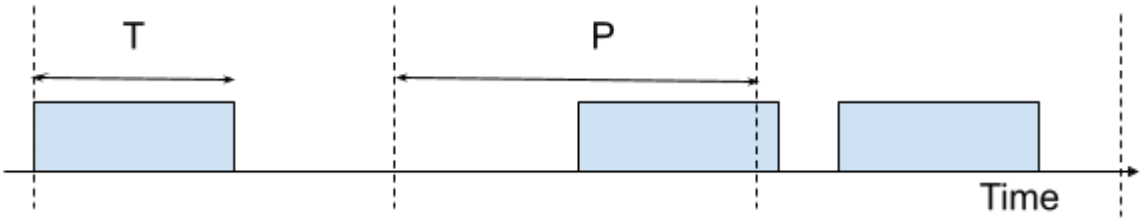
sugar	spice	e_nice	c_x	ppg	Notes
0	0	0	0	2	Initial values

Name: _____ Entry number:_____

Q5 [13 marks] Scheduling

Tasks in hard real-time systems, such as autonomous vehicles and multimedia players, have strict deadlines. Let us assume that tasks are periodic and take a fixed processing time. The period is denoted by P and the processing time is denoted by T with T < P.

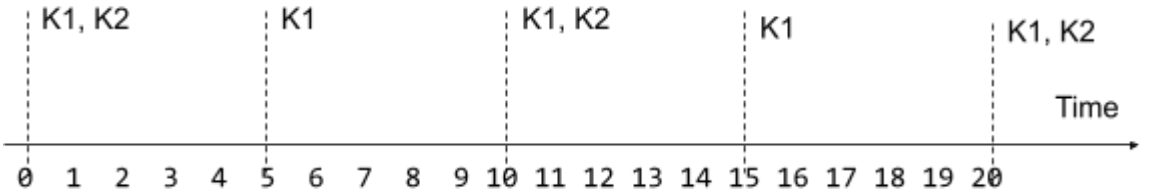
Tasks need to finish before their next period starts. The following shows a schedule with three periods of a task. The task misses its deadline in the second period.



Rate-monotonic scheduler assigns a priority 1/P to each task. It is a preemptive scheduler: if a lower-priority task is running and a higher-priority task becomes available to run, it preempts the lower-priority task.

Q5.1 [1 mark] Given two tasks K1 (T=2, P=5) and K2 (T=4, P=10), which task will have a higher priority?

Q5.2 [2 marks] Draw how the two tasks above will be scheduled by the rate monotonic scheduler during the first 20 seconds. At time 0, tasks K1 and K2 arrive. At time 5, task K1 arrives. At time 10, tasks K1 and K2 arrive, and so on.



Q5.3 [1 mark] What is the average CPU utilisation for the above schedule?

Q5.4 [2 marks] Given N tasks where task i has period P_i and processing time T_i, what will the average CPU utilisation to run these tasks?

Q5.5 [1 marks] Provide a formula using which we can safely conclude that it is impossible for any scheduler to schedule the N tasks above on a single CPU.

Q5.6 [2 marks] Now consider two tasks K1 (T=3, P=5) and K2 (T=3, P=8). Draw how the two tasks will be scheduled by the rate monotonic scheduler during the first 20 seconds. Assume that at time 0, tasks K1 and K2 arrive. At time 5, task K1 arrives. At time 8, task K2 arrives, and so on.

Q5.7 [4 marks] Suggest another scheduling policy such that neither of the two tasks above miss their deadlines. Note that the policy needs to be general-purpose, i.e, it does not assume a certain number of tasks, or a certain (period, processing time) for any task. Also redraw the schedule for the tasks above using your scheduling policy.

Name: _____ Entry number:_____

Q6 [6 marks] Page tables

Let us say we want to expose a virtual address space of size 2^{64} bytes. Our physical memory is of size 16TB and our pages are of size 1GB.

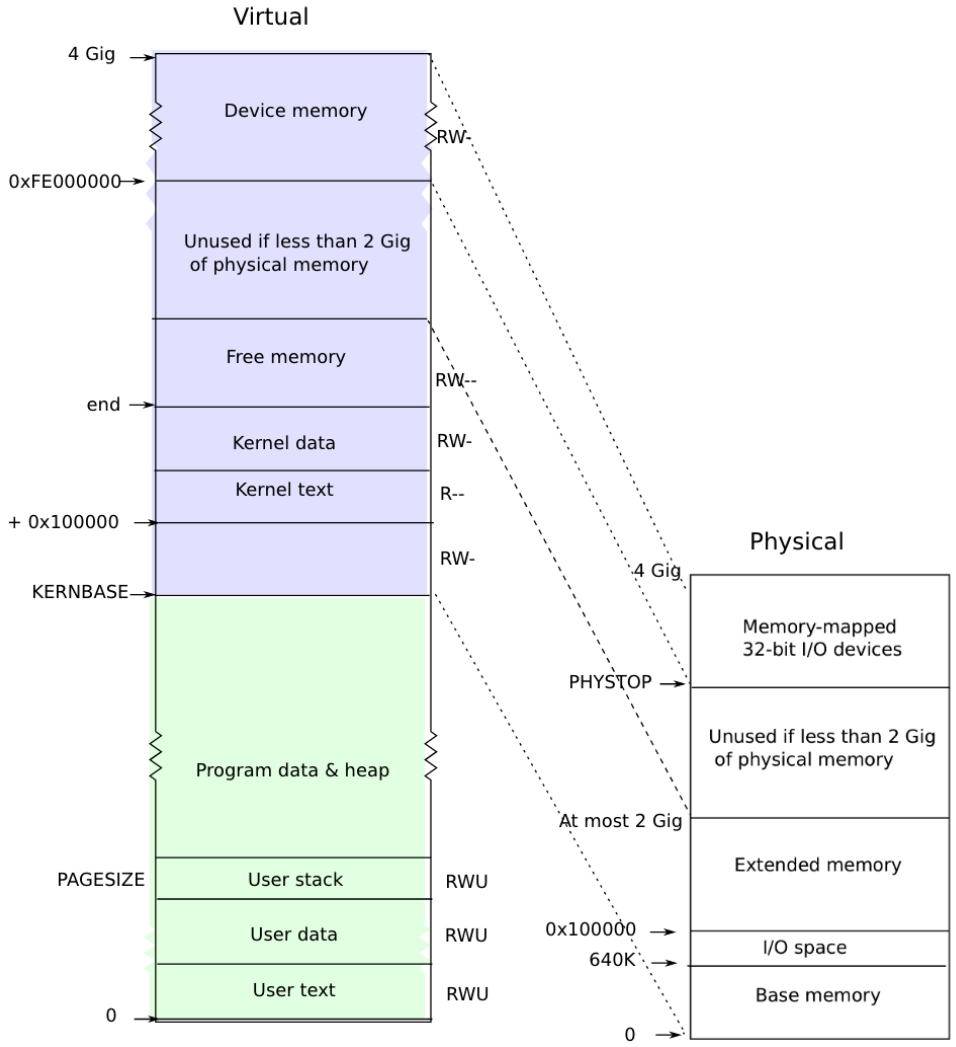
Q6.1 [1 mark] How many page table entries do we need to represent the full virtual address space?
Q6.2 [1 mark] How many physical pages do we have?
Q6.3 [1 mark] How many bytes does each page table entry need for this system? Ignore requirement of bits other than physical page number in the page table entries.
Q6.4 [1 mark] How many page table entries can be stored in one page?
Q6.5 [1 mark] In a single-level page table (i,e, not a multi-level page table), how many pages are required to represent a full virtual address space?
Q6.6 [1 mark] What is the disadvantage of using 1GB pages over using 4KB pages?

Space for rough work

Name: _____ Entry number:_____

Project exam

Let us say we want to add exokernel-like capability to xv6 where we want to let processes read their own page tables. Following shows the xv6 process' virtual memory to physical memory mapping for your reference.



Q1. [2 marks] Conceptually describe how you can add this functionality. You may assume the same available physical memory in xv6 code, i.e, the area in virtual address space from KERNBASE + PHYSTOP (0x80000000 + 0xE000000 = 0x8E000000) to DEVSPACE (= 0xFE000000) is unused.

Name: _____ Entry number:_____

Q2. [4 marks] Describe the xv6 files you will change and how. Try to mention file names and method names as much as you can.