1 Short Answer Problems (13 marks)

And I do mean **short**. If the answer is one word, give me one word.

1.1 Process Timing (8 marks)

Suppose you are faced with the need to cause your program to wait for **one second**. Your colleague, who is less skilled than you, proposes the following function:

```
void Wait()
{
    const int WaitTime=;//Select through experimentation
    for (int i=0;i<WaitTime;i++)
        ;
}
```

Your colleague points out that in a world of 64-bit integers, the loop can count higher than 10^{18}. Since computer clock speeds are limited to a few GHz, it should be possible to find a value for **WaitTime** that makes this loop take exactly one second. Your colleague wants to experiment with different values for **WaitTime** until you can find the right constant.

2

a. (2 marks) On a modern operating system (Linux, Android, Windows, etc) will this scheme work as a method to keep accurate time? Explain your answer.

No, because this process may be moved between running → ready → waiting as it is being "run" by the user so a second may last for longer than a second or however long. One time through the loop may be different from the time it takes through the loop on the i^{th} time.

2

b. (2 marks) What feature(s) would an operating system need to support in order for a scheme like this to keep accurate time?

It would need to allow the user to control whether the function can be blocked or not. It would have to support the user to be able to totally unblock the function to not wait on other threads or processes.

c. (2 marks) Supposing the scheme will keep accurate time, is it a good idea? Why or why not?

2

Although it may work it is a waste of the processing time, since there are functions and libraries that can be used to find the time. ✓

0

d. (2 marks) We have seen a number of better ways to keep time. Identify two. You can refer to system calls in any of the run-time environments we have discussed.

Date.time() X

Time.time() X

- sleep
- wait
- etc...

1.2 Process Management (2 marks)

We define a **context switch** as the change from one running process to another. Give **TWO** examples of information that must be stored and replaced during a context switch.

2

- page table ✓
- stack ✓

1.3 Parallel Processing (3 marks)

2

There are two methods of parallel processing (multi-tasking) that we have studied: threads and processes. As we have seen, the primary difference between them is that multiple **threads** can share a data space, whereas each **process** gets its own data space.
In the space below, provide one example for each technique of a situation in which it would be your preferred method. (ie: "Do I use a thread or a process here?"). Explain your selection.

If I am writing a program where something needs to be fetched, ✓ I would use threads because they are easier to destroy. ✓ Once I have completed the thread function, I can easily kill the thread and move forward with the ✓ program. I would use a process when writing a program ✓ where I am constantly going to require those different processes and they are needed for the duration. (Not destroyed until ✓ program exits). -almost

2 Analysis Problems (20 marks)

2.1 Virtual Memory (6 marks)

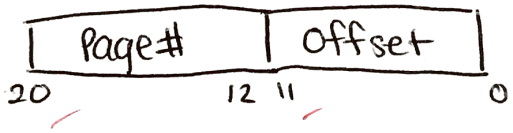
In working through this problem, you may find the following arithmetic identities useful:

$$1K = 2^{10} \quad (1)$$
$$= 1024 \quad (2)$$
$$1000(hex) = 4K \quad (3)$$
$$1M = (1K)^2 \quad (4)$$

Assume a virtual memory system has access to an 2MB (megabyte) address space. The system uses 4K memory pages. It has been installed on a system with 16KB of physical memory. The top 4KB of physical memory is ALWAYS reserved for the operating system.

a. (2 marks) What is the format of logical addresses?

2



$2^x = 2000000$

$x = 21$ bits

offset => $2^x = 4000$

$x = 12$ bits

b. (2 marks) How many entries does the page table for each process contain?

2

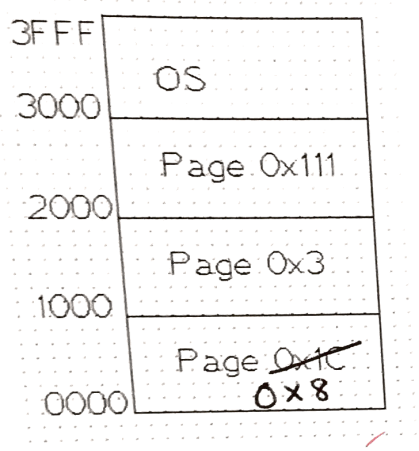
since the page # is 9 bits long

$2^9 = 512$ entries

4

c. (2 marks) At time zero, memory looks like the figure. The most recent three memory accesses (in chronological order from least recent to most recent) have been 0x1CA32, 0x3011, 0x11100A. If the next access issued goes to 0x8004, what **physical address** will be read?

2



Replaces page 0x1C since this is the least recently called. The base is 0, so the physical address will be 0x0004

2

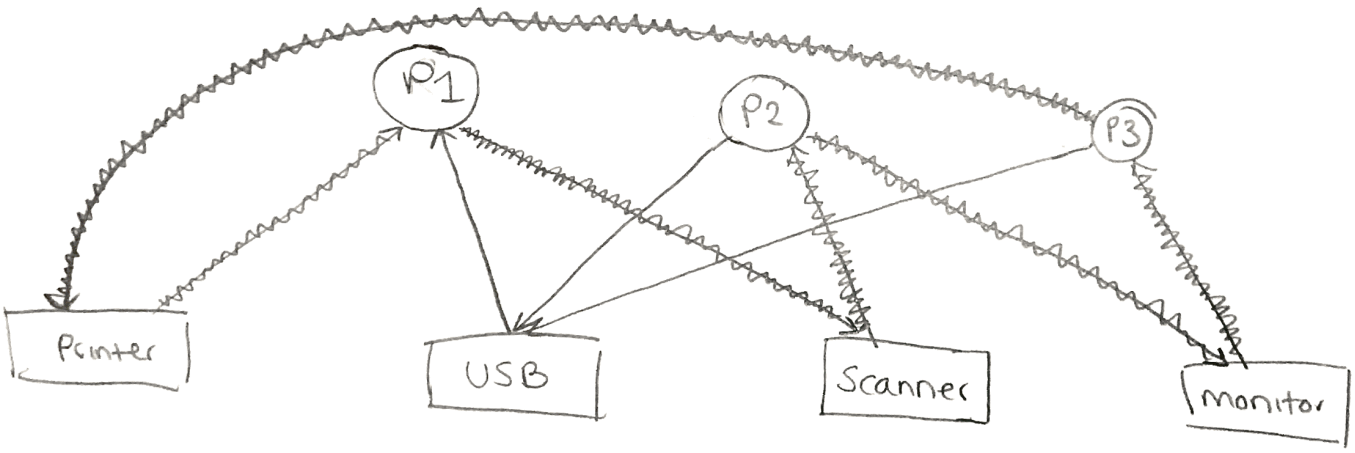
2.2 Semaphore Contention (8 marks)

Assume there are three processes running in a computer system that has four available resources: a usb port, a scanner, a printer and a monitor. Each of the resources is protected by a semaphore of the same name. The code of the three processes is like this:

// Do something
printer.Wait();
USB.Wait();
scanner.Wait();
/* Critical section */
scanner.Signal();
USB.Signal();
printer.Signal();
	// Do something
	scanner.Wait();
	monitor.Wait();
	USB.Wait();
	/* Critical section */
	USB.Signal();
	monitor.Signal();
	scanner.Wait();
		//Do something
		monitor.Wait();
		USB.Wait();
		printer.Wait();
		/*Critical section */
		printer.Signal();
		USB.Signal();
		monitor.Wait()

a. (4 marks) Use a resource allocation graph (or other convincing arguments) to show that deadlock is a possibility in this system.

4



loop = deadlock

4

b. (2 marks) Is it ever possible for this system to run without deadlock? Justify your answer.

2

Yes. If a whole process runs through before any others make their first request, then the process can use and release its resources. This is very unlikely.

c. (2 marks) Re-order the various Wait() and Signal() calls to maintain the resource protection for the critical sections while ensuring that deadlock will not occur.

2

(1) Printer
(2) USB
(3) Scanner
(4) Monitor

//Do something
Printer.Wait()
USB.Wait()
Scanner.Wait()
/*Critical Section*/
Scanner.Signal()
USB.Signal()
Printer.Signal()
	//Do something
	USB.Wait()
	Scanner.Wait()
	Monitor.Wait()
	/*CriticalSection*/
	Monitor.Signal()
	Scanner.Signal()
	USB.Signal()
		//Do something
		Printer.Wait()
		USB.Wait()
		Monitor.Wait()
		/*CriticalSection*/
		Monitor.Wait()
		USB.Wait()
		Printer.Wait()

4

2.3 Deadlock Avoidance (6 marks)

Assume that a multi-tasking system has four types of resources, identified as R1, R2, R3, R4. Before any program runs, there are 3 R1s, 6 R2s, 6 R3s and 5 R4s available. There are also four processes running, P1, P2, P3, P4. At the time of interest, the state of the problem looks like this:

	Worst Case				Allocated			
P1	1	2	2	3	0	2	2	1
P2	1	3	3	1	1	1	1	1
P3	0	3	2	5	0	3	2	1
P4	1	6	1	5	1	0	1	0

Still Need

	R1	R2	R3	R4
P1	1	0	0	2
P2	0	2	2	0
P3	0	0	0	4
P4	0	6	0	5

Left

1, 0, 0, 2

a. (4 marks) Show that this state is safe.

4 P1 can run initially, using resources to leave 0,0,0,0 and then releasing resources to leave 1,2,2,3. Then P2 can run, taking the resources it needs to leave 1,0,0,3. Then it releases its resources to leave 2,3,3,4. Then P3 can run, taking resources and leaving 2,3,3,0. Once finished, it releases its resources and we have 2,6,5,5. Now P4 can run through. Since all processes can run, system is safe

2

b. (2 marks) Suppose that process P3 makes a request for one R4. Is the system still safe?

That would leave the initial leftover resources to be 1,0,0,1. No processes would be able to run, so the system is not safe.

6

3 Software Design Problem (7 marks)

Suppose you are working as part of a design team to create a **event** capability. An **event**, is a synchronization object very like a **binary semaphore**. It can be in two states: **SET** and **RESET**. There are two differences. First, events must be **manually reset**. Second, when an event is signalled, **all threads blocked on that event** will unblock.
A thread can block on an event using **Wait** if it is in the **RESET** state (a call to **Wait** if it is in the **SET** state will simply return). When the event becomes **SET** after a call to **Set**, the block will end, but the event will remain **SET** until explicitly **Reset**.
The operating system currently uses **counted (ie: standard) semaphores** as its only synchronization tool:

```
class Semaphore
{
    // This is a counted semaphore
public:
    Semaphore (int initialCount);
    ~Semaphore();
    void Signal();
    void Wait();
};
```

Figure 1: Existing OS resources

Provide a design for a new class **Event**. As you can see in the figure, **Event** has five public functions: a constructor, **Set**, **Reset**, **Wait** and a destructor. Make sure your answer includes implementations for each of them. As usual, I will accept any plausible format (code, pseudocode, diagrams, stories, etc) BUT make sure you make reference to the features of the existing OS (Figure 1) as you use them so the operation is clear. You can add whatever data members to the class you need to make it work.

```
class Event
{
public:
    enum State {SET,RESET};
public:
    Event(State initialState);
    ~Event();

    void Set();
    void Reset();
    void Wait();
};
```

You may continue your work in this space.

```
Event (State InitialState){
    Threads
    if (      State == SET)
        set()
    else
        reset()
    Semaphore count = new Semaphore;
}
~Event(){
    kill threads
```

create threads and set the event status

Set() = Signal()
Reset changes binary#
Wait() = Wait()

```
Set(){
    for all threads
        count = count+1
        Thread.Wait()
    if count!=0{
        State = SET
}}
```

block threads, make sure that # of threads in Queue != 0, set state to SET

no signal here?

```
Reset(){
    for all threads
        thread.Signal()
        count--;
    if count==0
        State = RESET
}
```

why signal here?

unblock all blocked threads and then count to make sure that there is 0 threads, set state to RESET

2/7

```
Wait(){
    Block thread from continuing
    if(state = SET)
        Return
}
```

check before block

make a thread wait

if thread is already waiting then don't do anything