Quiz 2 - MET CS575 Graduate Operating Systems

Name: Sneha Singh Grade: / 15

1. Consider a logical address space of 256 pages with a 4-KB page size, mapped onto a physical memory of 64 frames. (2 points)

a. **How many bits** are required in the logical address (**show work**)?

1 kb = 1024 bit = 2^10

4 Kb = 4 * 1024 = 2^12

So, we need at least 12 bits for logical address.

Also, 8 bits for each page so we need around 20 bits for the logical address.

Answer: 20 bits

b. **How many bits** are required in the physical address (**show work**)?

There are 64 frames.

64 = 2^6

Hence, we require additional 6 bits.

So, in total **26 bits** are required for the physical address.

2. Explain why implementing synchronization primitives by disabling interrupts is not appropriate in a single-processor system if the synchronization primitives are to be used in **user-level** programs. (1 point)

Since interrupts can, by definition, occur at any time, and because they cannot always be ignored by the kernel, the sections of code affected by interrupts must be guarded from simultaneous use.

If a user-level program uses the synchronization primitives then it is not appropriate to disable interrupts as it can disable the timer interrupt and prevent context switching from taking place, thereby not allowing other processes to execute.

3. Assume that a system has multiple processing cores. For each of the following scenarios, describe which is a better locking mechanism — a Spinlock or a Mutex lock where waiting processes sleep while waiting for the lock to become available. (3 points)

Circle the Spinlock or the Mutex lock for each scenario shown below:

- The lock is to be held for a short duration: Spinlock **or** Mutex lock

  Answer: Spinlock - because it could be faster than a Mutex which needs to suspend then awaken the waiting process

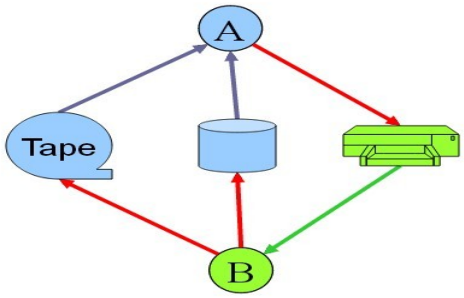
- The lock is to be held for a long duration: Spinlock **or** Mutex lock

  Answer: Mutex lock - as this allows the other processing core to schedule another process while the locked process waits.

- The thread may be put to sleep while holding the lock: Spinlock **or** Mutex lock

  Answer: Mutex lock - as you wouldn't want the waiting process to be spinning while waiting for the other process to wake up.

4. A utility program performs the following operations: copy a file from tape to disk and print the file on printer. The figure shows resource allocation graph for Processes A and B, and Resources Tape, Disk, and Printer.



a. Is there a potential deadlock? Yes or No. Please explain. (2 points)

   Yes , there is a deadlock. Process B is holding the printer so A cannot print

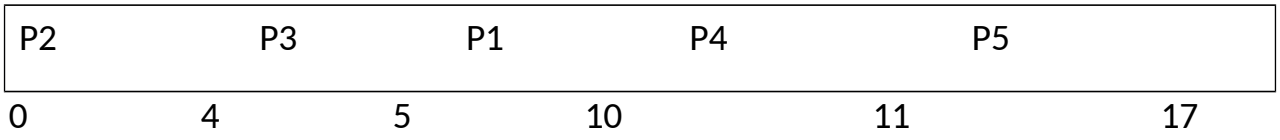
b. Using the symbols in the form of **P->R** and **R->P**, list all of the resource allocations and requests shown in the figure. Hint: there are six. (2 points)

- **tape -> A**
- **A -> printer**
- **B -> tape**
- **Resource -> A**
- **Printer -> B**
- **B -> resource**

5. CPU Scheduling using the **nonpreemptive** SJF scheduling algorithms (Shortest-Job-First Scheduling).

Process	Arrive time	CPU burst time
P1	2	5
P2	0	4
P3	2	1
P4	6	1
P5	5	6

a. Draw the Gantt Charts to illustrate detail execution of the processes using the algorithm mentioned, showing process's name, and arrival/start time according to its burst length. All numbers shown are in milliseconds. (3 points)



b. What is the turnaround time of each process? (2 points)

Hint: Use the formula turnaroundTime = finishTime - arrivalTime

P1 = __10 - _2__ = _8

P2 = __4_ - _0__ = _4__

P3 = ___5 - _2__ = ___3

P4 = __11 - _6__ = __5

P5 =17_ - __5_ = 12_