Problem 1: (9 minutes) Briefly explain the following terms

- TLB miss: A TLB miss occurs when a memory reference is to a page whose page table entry is not in the TLB. At that point, the page table entry has to be fetched from memory (usually this is implemented in software).
- One shot resource allocation: A resource allocation policy that prevents deadlock by forcing processes to acquire all the resources at once at the beginning. It prevents the hold-and-wait ingredient of deadlock as processes are either waiting (at the beginning) or holding (after the start), but never both.
- Protection requirement for memory management: Protection requires that no process can access physical memory belonging to any other process unless that memory is explicitly shared.

Problem 2: (10 minutes) Discuss the any 3 of the following potentially wrong statements.

1. Implementing segmentation for a paged system makes sense just like implementing paging for a segmented system does (e.g., as is used in the Intel Architecture in the last homework)
   This makes little sense for many reasons. Most importantly, paging is more efficient than segmentation due to the fixed power of two size and starting point for each page, and due to the relatively small size of pages. Thus, supporting segmentation with paging from the system side makes sense, but the opposite (making the system use the less efficient dynamic segments) doesn’t.

2. Deadlock detection is the most liberal algorithm that prevents deadlock from occurring
   False. Deadlock detection lets deadlock happen and then recovers from it. Thus, it does not prevent deadlock. Deadlock avoidance is the most liberal algorithm that prevents deadlock.

3. Translating memory addresses at load time makes sharing memory more difficult
   Generally, this is not true. If the addresses are translated then the page has to be mapped to the same location in every process memory (address translation is at load time not dynamic). On the other hand, self-referring code becomes easier to translate since it is in terms of physical addresses. However, given the requirement that the shared memory be at the same place, this is not a real advantage.

4. In virtual memory, when a page is swapped from memory to disk, the cache entries for it must be found and invalidated
   If the cache uses physical addresses, this is absolutely true – the cached physical locations no longer refer to the values stored at that location. If the cache is logically indexed, this is less important since the value in them cannot be mistaken for something else.

5. The TLB holds page table entries for valid pages only
   Valid pages are resident pages (pages whose valid bit in the page table is set) and the phrasing above should have been clearer. Generally, this is true as an invalid page will cause a page fault, so keeping an invalid entry in the TLB makes little sense given the high cost of a page fault. The TLB space is better used for a valid page.

Problem 3: (15 minutes) Show an example that illustrates each of the following:
1. A safe state

<table>
<thead>
<tr>
<th>Process</th>
<th>Claim</th>
<th>Allocated</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R1</td>
<td>R2</td>
</tr>
<tr>
<td>P1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>P2</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

\[ V = [1 \ 1] \]

The above is a safe state since sequence P1, P2 allows finishing without deadlock. Of course, infinitely many solutions are possible here.

2. An unsafe state

<table>
<thead>
<tr>
<th>Process</th>
<th>Claim</th>
<th>Allocated</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R1</td>
<td>R2</td>
</tr>
<tr>
<td>P1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>P2</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

\[ V = [0 \ 1] \]

The above is an unsafe state since no sequence exists that allows the processes to finish should they all make their maximum claim. Of course, infinitely many solutions are possible here.

3. Deadlock avoidance being more liberal than one shot allocation

<table>
<thead>
<tr>
<th>Process</th>
<th>Claim</th>
<th>Allocated</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R1</td>
<td>R2</td>
</tr>
<tr>
<td>P1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>P2</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

\[ V = [3 \ 2] \]

One shot allocation for the example above requires that a process get all the resources before being allowed to go. So, either P1 or P2 can start running, with the other process blocked for insufficient resources. However, deadlock avoidance lets them both start running and blocks them only if a particular request leads to an unsafe state. For example, P1 may ask for resources and release them before P2 does, and neither ever blocks. Alternatively, the safe state in part 1 above can occur with both processes having some resources and being active.

4. Best-fit performing worse than worst-fit for dynamic memory partitioning

Consider a memory that currently has two holes (free areas) of size 400K and 800K, called hole A and hole B respectively.

Suppose the following processes arrive in order: P0 (250K), P1, (550K), P3 (300K). Best fit places P0 in hole A (leaving 150K free), P1 in hole B (leaving 250K free), then cannot place P3. Worst fit places P0 in hole B (leaving 550K free), then P1 in hole B also exhausting hole B). P2 can now be placed in hole A, and worst fit outperforms best fit for this example.
Problem 4: (10 minutes)

(a) Consider an implementation of an OS where processes sometimes do not exit cleanly and
resources they have such as file descriptors are sometimes not released. Eventually, some processes
will ask for resources but there is an insufficient amount. Is this case an example of deadlock?

For these types of questions, you want to identify the resources then show the four ingredients
of deadlock. When it comes to hold and wait, the exited processes are not waiting for anything,
and this is simply a case of resource leak.

(b) After winning his second MVP, Alex Rodriguez is unhappy with his contract and wants
to negotiate a better deal and threatens with a holdout (he will not play unless he is paid more).
News sources report that Rodriguez and the Yankees cannot reach an agreement and are deadlocked.
Does this situation represent deadlock according to the CS definition?

Resources: Alex’s effort (held by Alex, waited on by Yankees), and raise, held by Yankees and
waited on by Alex. Can argue that hold and wait, non-preemption, mutual exclusion and circular
wait all hold.

(c) Assuming that the situation in (b) was an instance of deadlock, consider the following
scenario. The player and the club agree on a new contract, but for half the raise that the player
wanted. What type of solution to deadlock does this represent?

If the Yankees gave ARod the raise, we can argue for preemption (Alex took the money from
the yankees). This doesn’t really change because he changed his demands lower.

Problem 5: (20 minutes)
Paging extends fixed partitioning to allow multiple partitions per process. Similarly, segmentation
does the same for dynamic partitioning. Recall that the buddy system, like fixed and dynamic par-
titioning, is an approach to contiguous memory allocation.

(a) Develop a multiple partition memory management approach based on the Buddy system.
Clearly show how address translation is carried out.

Not going to do the full solution here. The scheme has aspects of both segmentation and paging
although it is mostly like segmentation. The segment table would hold all the buddy segments.
However, instead of base and limit, the entries are like paging entries since the segments are power
of 2 size and start at specific locations in physical memory (If the full memory size is $2^m$, for each
segment of size $2^n$ it can start at any $2^n$ boundary; there are $2^{m-n}$ frames of that size.).

So the segment table entry should include the buddy segment size in bits, and the frame number
among frames of that size. For ease of explanation, assume that we keep one page table for every
segment size, although one page table can be used provided we can do the page lookup of multiple
bit sizes concurrently. When we have a logical address, we look for a match in all the segment tables
concurrently. Exactly one will match, and then we can replace bits m-n with the frame number in
the page table.

(b) Discuss how protection and sharing can be supported for such a system.

Protection is a non-issue just like with paging (cannot access an offset bigger than the frame
size and cannot access a page outside the legal range). Sharing is just like with paging– the buddy
segments have to be of matching size.

(c) Discuss the advantages and disadvantages of such a system compared to paging and segmenta-
tion.
Like paging, this allows easier translation and protection than segmentation. However, it is slightly more complicated than paging because of the variable size segments.

Like segmentation, this allows the declaration of segments from the user perspective. This good for the user, but will result in large size segments which are a drawback when considering placement and virtual memory.

(d) Are there issues in implementing virtual memory for this system?
Yes. The page/frame sizes are variable; this complicates managing virtual memory significantly. Consider, for example, page replacement where finding a victim to replace has to consider either all victims of equal or bigger size, or multiple of segments of smaller size that can be combined into the desired size.

**Problem 6:** (8 minutes) Answer the following questions.
(a) What hit ratio on the TLB is needed to bring the average address translation time to 10% of its value without TLB, assuming TLB accesses are free.

Assume the cost without TLB is $X$. With a TLB, the cost is $(1 - h)X + h(TLBa)$, where $h$ is the TLB hit ratio and $TLBa$ is the TLB access time. Since TLB access time is 0, the translation cost with TLB is $(1-h)X$.

This refers to the time when there is a TLB miss and we have to access memory for the page table at cost $X$. To make the new cost 10% of the old cost, $(1-h)$ should be 10%, or $h$ has to be 0.9 (hit ratio of 90%)

(b) What happens if the size of the TLB is smaller than the number of active pages (memory pages that the program is currently referencing) at any given time?
Frequent TLB misses will occur, causing translation to be slow.

(c) What happens if the size of the allocated physical memory for a process is smaller than the size of the active pages?
Frequent page misses will occur (thrashing) causing the performance to drop dramatically.

**Problem 7:** (15 minutes)
Consider a paged memory system where the frame size is 4Kbytes ($2^{12}$ bytes). The page table size is restricted to $2^{16}$ entries. In addition, the size of the physical memory is $2^{15}$ frames.

(a) What is the maximum memory size (logical) that a process can have?
Number of Page table entries represent the maximum number of pages. Each page is 4K. So, the maximum logical space is $2^{16} \times 2^{12} = 2^{28}$.

(b) How many processes can fit in memory if each of them takes up 1/4 of its maximum size
If each process uses 1/4 of its size, it will need $2^{16}/4 = 2^{14}$ frames. Since the physical memory size is $2^{15}$, two processes can fit.

(c) Consider a computer where the page tables are kept in memory. The cost of accessing memory is 500nsec. A TLB is used to optimize translation; the cost of accessing the TLB is 50nsec. What should the TLB hit rate be to make the average translation time 75nsec?
$(1 - h) \times 500 + h \times 50 = 75$
solve for $h$ to get $\frac{425}{500}$ or 94.4%

**Problem 8:** (25 minutes) An OS uses a multiple level feedback scheduler with 3 round-robin levels. The quantum for the 3 levels are 1, 2, and 4 time units respectively. Assume that we have 6 jobs
that arrive at time 0 with burst lengths of 8, 2, 1, 3, 10, 4. The jobs are listed in the order they arrive (the one with length 8 arrived first).

(a) Show the Gantt chart for these processes. What is the response time and the normalized turnaround time?

No solution provided. Response time is the average length of each wait period. You can add up the waiting time for the process and divide it by the number of wait periods. Or, you can choose to omit the first wait period.

Normalized turnaround time is total time of each process in the system, divided by its run time.

(b) What are the advantages/disadvantages of this scheduler vs. Round Robin scheduling?

It favors shorter processes, and is therefore fairer to I/O processes than round robin. For long processes, when they get a quantum, they experience less context switches. One disadvantage is that long processes may starve if frequent short processes appear.

(c) What are the advantages of Multiple Level Feedback relative to Shortest Job First scheduling; are there any given that SJF minimizes the average wait time?

SJF requires estimating the run time of a process, while multiple level feedback effectively derives that. In addition, SJF strictly favors processes based on run time; whereas MLF is fairer in that it lets processes of similar length alternate using the CPU.

(d) Consider a scheme called selfish round robin scheduling (SRR), which delays newly arrived processes until they catch up in priority to existing processes. Specifically, processes that are waiting build up priority at a rate $x$, while processes that are running build up priority at a rate $y$ where $x$ is bigger than $y$. A new process is only admitted when its priority catches up with the priority of any running process. SRR appears to do the opposite of the multi-level feedback algorithm above, yet both approaches have been proposed. Can you explain this apparent paradox?

The idea behind SRR is to attempt to reduce the context switches. Processes that have been in the system for a while are allowed to continue to run, without competing with newly arriving processes, which have to pay their dues before getting into the system. In summary, it is optimizing utilization by reducing context switches.

MLF on the other hand favors new processes which are often I/O bound processes where response time matters the most. Thus, it favors response time.

**Problem 9:** (bonus; 30 minutes)

This problem is bonus because you will have to read ahead to some stuff we have not covered yet. Consider a system where logical memory is $2^{38}$ bytes, and physical memory is $2^{30}$ bytes. The memory system is paged. There is a total of $2^{16}$ frames in the physical memory. The system is multiprogrammed (more than one process).

(a) What is the page size?

(b) You are trying to decide whether to use an inverted page table or a two level page table for this system. Assume a page table entry is 8 bytes, while an inverted page table entry is 16 bytes. Assume that in a two level page table scheme, every process needs the directory as well as 4 pages of the page table in memory all the time. Typically, you will have 16 processes active in the system. Which page table organization do you pick to minimize the amount of memory taken up by the page tables? Show your work

(c) The system has a TLB and a cache that is indexed on physical addresses and uses a two level page table. Show in painful detail the process of a memory access showing the alternatives at every step. Also, discuss what is implemented in hardware and what is implemented in software.
(d) What needs to be done when a context switch occurs in terms of memory management in the system described in part (c)?