Do not turn this page until you have received the signal to start. 
(In the meantime, please fill out the identification section above, 
print your name and student number on the back of the exam, 
and read the instructions below carefully.)

This midterm test consists of 10 questions on 15 pages (including 
this one). When you receive the signal to start, please make 
sure that your copy of the examination is complete.

If you need more space for one of your solutions, use the last 
page of the test and indicate clearly the part of your work that 
should be marked. We have been careful to leave enough space 
for your answers.

In your written answers, be as specific as possible and ex-
plain your reasoning. Clear, concise answers will be given 
higher marks than vague, wordy answers. Please make your 
handwriting legible!

Please write in pen. Answers written in pencil will not be re-graded.

Marking Guide

# 1: _____/ 8
# 2: _____/ 8
# 3: _____/12
# 4: _____/ 6
# 5: _____/ 8
# 6: _____/12
# 7: _____/10
# 8: _____/10
# 9: _____/ 8
# 10: _____/ 8

TOTAL: _____/90
**Question 1.** Definitions [8 marks]

Define the following terms in the context of this course.

**Part (a)** [2 marks] Emergent property

**Part (b)** [2 marks] Open system

**Part (c)** [2 marks] Quiescent state

**Part (d)** [2 marks] False sharing
Question 2. Systems Design [8 marks]

Part (a) [4 marks] In the first week, we discussed four general techniques for coping with complexity: modularity, abstraction, layering and hierarchy. Which of those four techniques is most evident in the design of the Unix timesharing system? Use at least one specific example to explain your answer.

Part (b) [4 marks] What was the guiding design philosophy of the exokernel? How did the authors of the exokernel paper apply the “end-to-end” argument to motivate this design philosophy?
Question 3. OS Structure [12 marks]

Part (a) [4 marks] Mach and L4 provide the same four basic abstractions and/or mechanisms (although with different names). Identify them and explain why these must be implemented by a microkernel.

Part (b) [4 marks] Explain the difference between Type I and Type II Virtual Machine Monitors, and give one specific example of each.

Part (c) [4 marks] Explain briefly how OS extensions are supported by Mach and Linux, and identify the primary advantage that each approach has over the other.
[This page intentionally left blank for rough work. It will **NOT BE MARKED** unless you **CLEARLY INDICATE** the part of your work that you want us to mark.]
Question 4. Performance Evaluation and Benchmarking [6 MARKS]

Part (a) [4 MARKS] Suppose you need to measure the time for some function f() in an ordinary user-space process. You can use either (i) the hardware cycle counter (continuously incremented on every cycle, without regard to context switches or mode switches), (ii) an interval timer with 10ms resolution that records user time per process, or (iii) some combination of both. The system clock ticks at the same resolution as the interval timer. You expect the runtime of f() to be less than 5ms. Describe how you would measure the time of f() as accurately as possible. (Point form answers are fine)

Part (b) [2 MARKS] After reading some old papers on file system benchmarking, you are inspired to benchmark the file system on your own (modern, high-end) desktop. You download the filesystem benchmark used in the research papers and run it with the default configuration. The benchmark reports that your system has a sequential read bandwidth of 1 GB/s. The spec sheet for your hard drive indicates a sequential read bandwidth of 200 MB/s however. Explain the most likely cause of the discrepancy.
**Question 5.** Optimization [8 marks]

A system runs a workload that spends nearly all its time on disk I/O consisting of roughly 80% random read and 20% sequential write operations (by time). You are asked to upgrade the system from a single hard disk to two disks in a RAID-0 configuration (data is striped across the two disks) with the following properties:

- Reads become slower, at 0.8x of the original (bottlenecked on the slowest drive seek).
- Writes become 2x faster, utilizing the sequential bandwidth of both disks.

**Part (a)** [4 marks] Explain why this upgrade is a bad idea. Show your work. (4 marks)

**Part (b)** [4 marks] Instead of the RAID-O configuration, you propose adding an SSD to act as a read cache. This will achieve 5x faster read performance for the workload, but the time spent on cache management will add 10% overhead (of the original time). What overall speedup can you expect? Show your work.
Question 6. Signals and IPC [12 marks]

Part (a) [4 marks] Explain when and how a user-level signal handler is invoked, including the purpose of the trampoline code placed on the user stack.

Part (b) [4 marks] A monitoring and control process needs to take some action \(a_1\) (e.g. read a temperature sensor and adjust fan speed) every 15ms of real time. It must perform another action \(a_2\) every 1.5s. In between these actions, the program has other compute work to do. The OS provides interval timers with 1ms resolution. The process sets up an interval timer with an initial value of 15ms. The OS decrements the timer in real time, posts SIGALRM to the process when it reaches 0, and resets the timer to the initial value (repeatedly). Thus, the OS will post SIGALRM to the process every 15ms. The process uses a signal handler to catch SIGALRM and (i) performs \(a_1\), (ii) increments a counter, and (iii) when the counter reaches 100, performs \(a_2\) and resets the counter to 0. SIGALRM is an ordinary POSIX signal. Explain what can go wrong in a heavily loaded system.

Part (c) [4 marks] Describe two features of the mbuf data structure used for socket and network communication that are designed to support efficient message handling.
Question 7. Event Notification [10 marks]

Part (a) [2 marks] Why is an event notification mechanism like select() or poll() needed?

Part (b) [4 marks] Briefly explain how kqueue provides efficient and scalable event notification, including the use of the knote data structure.

Part (c) [4 marks] The data in the kqueue paper showed that it was roughly 1.5X more costly to delete an event than to disable an event on 400 descriptors, and that deletion was less scalable as the number of descriptors increased. Suppose you have previously registered an event on some (larger than 400) set of descriptors, and you are no longer interested in that event. Should you delete the event or just disable it? What extra information, if any, would you need before making this decision?
Question 8. Advanced Locking [10 marks]

Part (a) [2 marks] Why is lock acquisition expensive on modern multiprocessors, even if there is no contention for the lock?

Part (b) [2 marks] Suppose we have a process with many more threads than there are CPUs on the system where it executes. What problem can occur with ticket locks that would not occur with ordinary spinlocks?

This question continues on the following page.
Part (c) [6 marks] Code for the MCS lock, as shown in lecture, is reproduced below. When a thread attempts to acquire a lock that is already held, it first sets locked of its own node to TRUE (line 9), and then redirects next of its predecessor to its own node (line 10). Give a scenario to show that the MCS queue lock could deadlock if this order were reversed.

```c
struct qnode {
    int locked;
    struct qnode *next;
}

void acquire(struct qnode **tail, struct qnode *my_node) {
    my_node->next = NULL;
    struct qnode *pred = fetch_and_store(tail, my_node);
    if (pred != NULL) { // queue not empty
        my_node->locked = TRUE;
        pred->next = my_node;
        while (my_node->locked); //spin
    }
}

void release(struct qnode **tail, struct qnode *my_node) {
    if (my_node->next == NULL) {
        if (compare_and_swap(tail, my_node, NULL))
            return;
        while (my_node->next == NULL); //spin
    }
    my_node->next->locked = FALSE; // release next waiter
}
```
Question 9. Transactional Memory [8 marks]

Suppose we have the binary search tree shown below, which has been implemented with transactional memory as shown in the code boxes. Assume the TM system tracks conflicts at a word granularity.

```
typedef struct treenode_s {
    int key;
    int value;
    struct treenode_s *left;
    struct treenode_s *right;
} tnode_t;

int tree_search (tnode_t *root, int k, int *val) {
    int found = 0;
    tnode_t *n;
    atomic {
        n = root;
        while (n != NULL && k != n->key) {
            if (k < n->key)
                n = n->left;
            else
                n = n->right;
        }
        if (n && k == n->key) {
            *val = n->value;
            found = 1;
        }
    }
    return found;
}
```

```
void tree_insert (tnode_t **root, tnode_t *newnode) {
    tnode_t *cur;
    tnode_t *parent;
    atomic {
        cur = *root;
        while (cur != NULL) {
            parent = cur;
            if (newnode->key < cur->key)
                cur = cur->left;
            else
                cur = cur->right;
        }
        if (parent == NULL)
            *root = newnode;
        else {
            if (newnode->key < parent->key)
                parent->left = newnode;
            else
                parent->right = newnode;
        }
    }
}
```

$T_1$ is executing `tree_search(root, 20, &result)`. Concurrently, $T_2$ is executing `tree_insert(&root, mynewnode)` for a new node with key 16. $T_2$ reaches the commit point first and successfully commits its transaction.

**Part (a) [4 marks]** What items are in $T_1$’s read set and write set when it attempts to commit the transaction?

**Part (b) [4 marks]** Will $T_1$’s transaction commit successfully? Explain your answer making reference to $T_1$’s read set and write set, and $T_2$’s transaction.

Write your answers on the following page. (Part (a) and (b) are repeated for your convenience.)
Question 9. Transactional Memory (CONTINUED)

Part (a) [4 marks] What items are in $T_1$’s read set and write set when it attempts to commit the transaction?

Part (b) [4 marks] Will $T_1$’s transaction commit successfully? Explain your answer making reference to $T_1$’s read set and write set, and $T_2$’s transaction.
Question 10. RCU and Scalability [8 marks]

Part (a) [4 marks] Why is memory reclamation challenging for RCU-protected data structures?

Part (b) [2 marks] RCU can be applied to synchronize access to data structures like linked lists. Should all linked lists in a multiprocessor OS kernel be protected by RCU? Briefly justify your answer.

Part (c) [2 marks] In lecture, we looked at a shared counter that was implemented as an array of per-CPU counters with each element padded to occupy a full cache line. Should all counters in a multiprocessor OS use this implementation? Briefly justify your answer.
Print your name and student number in this box.