CS 6210 Spring 2015 Midterm

Name:_______Kishore_________________________GT Number:

Monday, February 23, 2015 (11:05 AM to 11:55 AM)

Note:

1. **Write your name and GT number AT LEAST on the first page.**
2. The test is **CLOSED BOOK** and **NOTES**.
3. Please provide the answers in the space provided. You can use scratch paper (provided by us) to figure things out (if needed) but you get credit **only** for what you put down in the space provided for each answer.
4. For conceptual questions, **concise bullets** (**not wordy sentences**) are preferred. **YOU DON'T HAVE TIME TO WRITE WORDY SENTENCES…**
5. While it is NOT REQUIRED, where appropriate use figures to convey your points (a figure is worth a thousand words!)
6. **Illegible answers are wrong answers.**
7. **DON'T GET STUCK ON ANY SINGLE QUESTION…FIRST PASS: ANSWER QUESTIONS YOU CAN WITHOUT MUCH THINK TIME; SECOND PASS: DO THE REST.**

Good luck!

Question number	Points earned	Running total
1 (0 min) (Max: 1 pts)		
2 (12 min) (Max: 24 pts)		
3 (12 min) (Max: 22 pts)		
4 (12 min) (Max: 28 pts)		
5 (7 min) (Max: 15 pts)		
6 (5 min) (Max: 10 pts)		
Total (48 min) (Max: 100 pts)		

1. (0 min, 1 point) (This is a freebie, you get 1 point regardless)
   The most recent Orange bowl was won by
   (a) Miami Hurricanes
   (b) Stanford Cardinal
   (c) Mississippi State Bulldogs
   (d) Georgia Bulldogs
   [✓ (e) Georgia Tech Yellow Jackets]
   (f) There is a contest for winning a bowl of oranges?

CS 6210 Spring 2015 Midterm

Name:_______Kishore_________________________GT Number:

OS Structures
2. (**12 min**, 24 points)
(a) (4 points) (General)
Enumerate the **four** different types of program discontinuities that can occur during the execution of a process with a **short** sentence explaining what each one of the discontinuities is.

- **Exception - program generated arithmetic errors (e.g., divide by zero) [1 point]**
- **Trap - Syscall [1 point]**
- **Page faults [1 point]**
- **External interrupt - I/O [1 point]**

(b) (4 points) (SPIN)
SPIN relies on programming language support for the implementation of protection domains. Discuss **one** pro and **one** con of this approach (two short bullet points should suffice).

- Pro: No cheating possible (e.g. void*-casting in C) with Modula-3, generic interface provides entry point, implementation hidden, allows for checking for the correct pointer at compile-time, allows for protection domains without incurring border crossing costs due to shared hardware address space [2 points]
Note: if no mention of protection domains, [only 1 point]
- Con: What about drivers/etc. that need H/W-access, need to step outside of language protection, also Modula-3 potentially not too popular, rewriting necessary [2 points]
Note: if not mentioning drivers or H/W-access, [only 1 point]

(c) (8 points) (Exokernel)
A library OS implements a paged virtual memory on top of Exokernel. An application running in this OS encounters a page fault. List the steps from the time the page fault occurs to the resumption of this application. Make any reasonable assumptions to complete this problem (stating the assumptions). (Concise bullets please)
- Fielding by exokernel [2 points]
- exokernel notifies Library OS by calling the entry point in the PE data structure for this library OS [2 points]
- Library OS allocates page frame running page replacement algorithm if necessary to find a free page frame from the set of page frames it already has [2 points]
- Library OS calls exokernel through the API to install translation (faulting page, allocated page frame) in TLB, presenting capability, an encrypted cypher [2 points]

Note: [-2 Points] if not giving enough detail (PE, capabilities, etc.)
Note: [-4 Points] if only considering case of guaranteed mapping

(d) (8 points) (Liedtke's L3 Microkernel)
There are four myths that discourage microkernel-based design of an OS: (1) kernel-user switches (border crossings) are expensive; (2) hardware address space switching is expensive; (3) thread switches are expensive; (4) locality loss in the caches can be expensive when protection domains are implemented using hardware address spaces. How does Liedtke debunk these myths? (Four bullet points - one for each myth - should suffice)

- (1): Proof by construction, 123 processor cycles (incl TLB + cache misses) [2 points]
- (2): Not necessarily, exploit H/W features, e.g. segment registers to pack small protection domains in the same hardware space + explicit costs for large protection domains much smaller than implicit costs (cache no longer warm etc.) [2 points]
- (3): Shown by construction that this is not true, competitive to SPIN and Exokernel [2 points]
- (4): This was true in Mach due to its focus on portability, but if focus on performance this can be overcome, taylor code to use architecture-specific features (e.g. segment-registers on x86) [2 points]

Virtualization

3. (**12 min**, 22 points)
(a) (2 points) (General)
What is the distinction between "physical page" and "machine page" in a virtualized setting?

- **Physical page is the illusory view of the physical memory from the Guest OS MMU. Physical pages are deemed contiguous from the Guest OS point of view. [1 point]**

- **Machine page is the view of the physical memory from the Hypervisor. This refers to the REAL hardware physical memory. The actual "physical memory" given to a specific guest OS maps to some portion of the real machine memory. [1 point]**

(b) (10 points) (Xen)
Recall that in a para virtualized setting, the guest operating system manages its own allocated physical memory. (Note: you don't have to worry about being EXACT in terms of Xen's APIs in answering this question; we are only looking for your general understanding of how para virtualization works)

CS 6210 Spring 2015 Midterm

Name:_______Kishore________________________GT Number:

```
Consider the following scenarios:
(i) A new process starts up in a para-virtualized library OS executing on
top of the Xen hypervisor.  List the interaction between the library OS and
Xen to establish a distinct protection domain for the process to execute in.
```

The distinct protection domain mentioned in the question refers to the page-table for the new process. Following are the steps:

- **Library OS (Linux) allocates memory from its own reserve for a new page table. [1 point]**
- **Library OS registers this memory with the Hypervisor (Xen) as a page table by using a Hypercall. [1 point]**
- **Library OS makes updates to this page table (virtual page, physical page mapping) through hypervisor via batches of updates in a single hypercall. [2 points]**
- **Library OS changes the active page table via the hypervisor thus in effect "scheduling" the new process to run on the processor. [1 point]**

```
(ii) The library OS is "up called" by Xen to deal with a page fault incurred
by its currently running process.  How does the library OS make the process
runnable again?
```

When a page-fault occurs, the hypervisor catches it, and asynchronously invokes the corresponding registered handler. Following are the steps:

Inside the hypervisor:

- **Xen detects the address which caused the page-fault. For example, the faulting virtual address may be in a specific architectural register [1 point]**
- **This register value is copied into a suitable space in the shared data-structure between the hypervisor and library OS. Then the hypervisor does the up-call, which activates the registered page-fault handler. [1 point]**

Inside the library OS page-fault handler:

- **The library OS page fault handler allocates a physical page frame (from its pool of free, i.e., unused pre-allocated physical memory kept in a free-list). [1 point]**
- **If necessary the library OS may run page replacement algorithm to free up some page frames if its pool of free memory falls below a threshold. [1 point]**
- **If necessary the contents of the faulting page will be paged in from the disk. [1 point]**
- **Note that paging in the faulting page from the disk would necessitate additional interactions with the hypervisor to schedule an I/O request using appropriate I/O rings. [1 point]**
- **Once the faulting page I/O is complete, the library OS will establish the mapping in the page table for the process by making a hypervisor call. [1 point]**

(c)   (10 points) (VMware)
In an Intel-like architecture, the CPU has to do address translation on every memory access by accessing the TLB and possibly the page table. In a non-virtualized set up, the page table is set up by the OS and is used by the hardware for address translation through the page table base register (PTBR) set by the OS upon scheduling a process. Recall that in a fully virtualized setting, the page table for a process is an internal data structure maintained by the library OS.

(i) When the library OS schedules a process to run, how does the CPU learn where the page table is for doing address translation on every memory access?

To schedule a process the library OS will do the following:

- **library OS has a distinct PT data structure for each process. [1 point]**
- **dispatching this process to run on the processor involves setting the PTBR (a privileged operation) [1 point]**
- **The library OS will try to execute this privileged operation [1 point]**
- **This will result in a trap into the hypervisor [1 point]**
- **The hypervisor will “emulate” this action by setting the PTBR [1 point]**

Henceforth, the CPU will implicitly use the memory area pointed to by the PTBR as the page table

(ii) When the library OS services a page fault and updates the page table for a given process, how is this mapping conveyed to the CPU so that it can do the address translation correctly for future access to this page by this process?

- **Page fault service involves finding a free physical page frame to map the faulting virtual page to this allocated physical page frame. [1 point]**
- **To establish this mapping, the library OS has to update the page table or the TLB depending on the specifics of the processor architecture assumed by the fully virtualized library OS. Both of these are privileged operations which will result in a trap when the library OS tries to execute either of them. [1 point]**
- **Hypervisor will catch the trap and “emulate” the intended PT/TLB update by the library OS’s into the library OS’s illusion of PT/TLB. [1 point]**
- **More importantly, the hypervisor has a mapping of what machine page (MPN) this physical page of the guest OS refers to in its shadow page table. [1 point]**
- **Hypervisor will establish a direct mapping between the virtual page and the corresponding machine page by entering the mapping into the hardware page table (the shadow page table) or the TLB depending on the specifics of the processor architecture. [1 point]**

CS 6210 Spring 2015 Midterm

Name:_______Kishore________________________GT Number:

Synchronization, Communication, and Scheduling in Parallel Systems

4. (**12 mins**, 28 points)

(a) (2 points)

In a shared memory multiprocessor in which the hardware is enforcing cache coherence, why is a "memory consistency model" necessary?

Memory consistency model serves as a contract between software and hardware to allow the programmer to reason about program behavior.
(2 points if they mention hardware/software contract)
(-1 point for lack of specificity - for full credit, need to mention ordering or sequence)

(b) (4 points) All the variables in the execution shown below are in the shared memory of a cache coherent NUMA multiprocessor. The multiprocessor implements sequential consistency. All variables are initialized to 0 before execution starts.

Execution on processor P1	Execution on processor P2
x = 20	w = y + x
y = 30	z = x+10

Which of the following final values are **impossible** with the above execution? (**circle your choices; +2 for correct choice; -1 for incorrect choice**)

```
(i)   w =  0; z = 10;
(ii)  w =  0; z = 30;
(iii) w = 20; z = 30
(iv)  w = 30; z = 10
(v)   w = 30; z = 30
(vi)  w = 50; z = 30
```

(c) (4 points)

On an Non-Cache-Coherent (NCC) NUMA machine with per-processor caches, would you advocate using the following lock algorithm? Explain why or why not.

```
LOCK(L):
     back: while (L == LOCKED); //spin
                if (Test_and_Set(L) == LOCKED)go back;
UNLOCK(L):
     L = UNLOCKED;
```

No. Since there is no cache consistency Lock release will never be seen by waiting processors.
(1 point for saying "No". 3 points for the correct reason).

CS 6210 Spring 2015 Midterm

Name:________Kishore___________________________GT Number:

(d) (4 points) (Answer True/False with justification)
In a large-scale invalidation-based cache coherent shared memory multiprocessor with rich connectivity among the nodes (i.e., the interconnection network is not a shared bus), the tournament barrier is likely to perform better than the MCS barrier. To jog your memory, MCS algorithm uses a 4-ary arrival and binary wakeup tree. The tournament barrier uses a binary tree for both arrival and wakeup.

True. [1 point]
Tournament algorithm can exploit the rich connectivity for peer-peer signaling among the processors in each round (going up and down the tree). [3 points]

[1 point partial credit for any of the following: there is parallel communication, or messages aren't serialized, or log2(n) vs log4(n), or otherwise thinking about critical path/tree depth]

(e) (4 points)
Light-weight RPC (LRPC) is for cross-domain calls within a single host without going across the network. A specific LRPC call has totally 128 bytes of arguments to be passed to the server procedure, and 4 bytes of return value to be returned to the client. Assuming no programming language level support,
(i) What is the total copy overhead in bytes for the call-return with the LRPC system? Explain why.

Client	→	**A-Stack**	⇒	**128**
A-Stack	→	**Server E-Stack**	⇒	**128**
Server E-Stack	→	**A-Stack**	⇒	**4**
A-Stack	→	**Client**	⇒	**4**

Total = 264 [2 points]
Partial credit (1 point) if the answer says 132 with correct logic

(ii) How much of the copy overhead is due to copying into kernel buffers?

0 bytes [2 points]

(f) (5 points)
In a multiprocessor, when a thread becomes runnable again after a blocking system call completes, conventional wisdom suggests running the thread on the last processor it ran on before the system call.
(i) What is the downside to this idea?

Cache pollution by other threads run after the last time this thread ran on a particular processor. [2 points]
[Partial credit 1 point: for other plausible answers e.g., "poor load balancing"]

Name:_______Kishore________________________GT Number:

(ii) What do you think are important considerations in choosing the “right” processor to schedule this thread?

- **Cache pollution by threads run after the last time a particular thread ran on a processor (1.5 points)**
- **Cache pollution by threads that will be run after a thread is scheduled to run on a particular processor (queue length) (1.5 points)**

(g) (5 points)
Given the following configuration for a chip multiprocessor:

- 4 cores
- Each core is 4-way hardware multithreaded
- 512 MB LLC (Last Level Cache on the chip)

Given the following pool of threads ready to run:

- Group 1: T1-T8 each requiring 8 MB of cache
- Group 2: T9-T16 each requiring 16 MB of cache
- Group 3: T17-24 each requiring 32 MB of cache
- Group 4: T25-32 each requiring 64 MB of cache

Which threads should be run by the scheduler that ensures all the hardware threads are fully utilized **and** maximizes LLC utilization?

4 * 4 = 16 hardware threads
32 software threads
64 + 128 + 256 + 512 = 960 MB required (cumulative cache requirement)

One of many possible feasible schedules that uses all the hardware threads and uses all the available LLC:

- **T25 – T28 on Core 1 = 4 hardware threads & 256 MB**
- **T17 – T20 on Core 2 = 4 hardware threads & 128 MB**
- **T9 – T12 on Core 3 = 4 hardware threads & 64 MB**
- **T13 – T16 on Core 4 = 4 hardware threads & 64 MB**

(Full credit for any feasible schedule that shows complete understanding of hardware threads and LLC utilization)
[Partial credit: -1 if just one thread is chosen wrong (this is normally if they found the optimal schedule that’s less than 512 MB, instead of less than or equal to 512 MB); -2 for each set of up to 4 threads chosen incorrectly]

CS 6210 Spring 2015 Midterm

Name:________Kishore___________________________GT Number:

Parallel System Case Studies

5. (7 mins, 15 points)

(a) (5 points)

An application process starts up on Tornado multiprocessor OS. What are the steps that need to happen before the process actually starts to run?

- **Clustered process objects created one representation per processor on which this process has threads. [2 points]**
- **Clustered region objects created commensurate with the partitioning of the address space. [2 points]**
- **Clustered FCM objects created to support the region objects [1 point]**

(b) (5 points)

A process running on top of Tornado frees up some portion of its memory space (say using a facility such as free()). What does Tornado have to do to cleanup under the cover?

- **Locate the clustered region object that corresponds to this freed address range. This region object has the piece of the page table that corresponds to the address range being freed. [2 points]**
- **Identify the replicas of this region object on all the processors and fix up the mapping of these address ranges in the replicated copies of the page table entries corresponding to the memory being freed. [3 points]**

(c) (5 points) (Answer True/False with justification)

The “region” concept in Tornado and the “address range” concept in Corey are one and the same.

- **False [1 point]**
- **Justification:**
  - **Region is invisible to the application; it is a structuring mechanism inside the kernel to increase concurrency for page fault handling of a multithreaded application since each region object manages a portion of the process address space. [2 points]**
  - **Address range is a mechanism provided to the applications by the kernel for threads of an application to selectively share parts of the process address space. This reduces contention for updating page tables and allows the kernel to reduce the amount of management work to be done by the kernel using the hints from the application (e.g., reduce TLB consistency overhead. [2 points]**

CS 6210 Spring 2015 Midterm

Name:_______Kishore________________________GT Number:

Communication in Distributed Systems

6. (**5 mins**, 10 points)

(a) (5 points)

Assume that messages arrive out-of-order in a distributed system. Does this violate the "happened before" relation? Justify your answer with examples.

- **No. [1 point]**
- **Justification:**
  - **The happened-before relation is only concerned with the sending and receipt of messages ONE at a time. Therefore, messages arriving out of order does not violate the relationship [2 points]**
  - **[2 points] for a reasonable example.**

(b) (5 points)

Recall that in the distributed mutual exclusion algorithm of Lamport, every node acknowledges an incoming lock request message from a peer node by sending an ACK message. One suggested way to reduce the message complexity of the algorithm is to defer ACKs to lock requests. Using an example, show under what conditions a node can decide to defer sending an ACK to an incoming lock request.

If the node is holding the lock, then it can defer sending ACK.
Or if the incoming lock request's timestamp is larger than the timestamp of its own lock request, the node can defer sending ACK. [3 points]

A reasonable example [2 points]