Unit Exam: Virtualization

Total points 156/158

Take this exam alone. It is closed book, closed notes, but feel free to use a single 8.5" x 11" sheet of paper (both sides) of notes. Take the exam only once.

The respondent's email (**parth2@pdx.edu**) was recorded on submission of this form.

✓ "The OS can support many concurrent processes and schedules each of them to use the CPU." Is this an example of a mechanism, a policy or an objective? * 5/5

- (●) mechanism ✓
- ( ) policy
- ( ) objective

✓ "We aim to prevent starvation by ensuring that all processes make forward progress." Is this an example of a mechanism, a policy or an objective? * 5/5

- ( ) policy
- (●) objective ✓
- ( ) mechanism

✓ In the early days of computers, how did they limit direct execution? * 5/5

- ○ Operating Systems limited the direct execution
- ○ only one computer per university
- ○ deep learning
- ◉ only one person used computer at a time ✓

✓ Three processes run concurrently. Each uses a separate portion of physical memory for its code and private data. This is an example of what kind of sharing? * 5/5

- ○ time sharing
- ◉ space sharing ✓

How frequently is each type of limit used, checked or invoked? *

	Multiple times per instruction	Approximately once per instruction	Once per many (variable number of) instructions	Approximately once per 10ms	Score	
MMU address translation	◉	○	○	○	1/1	✓
timer interrupt	○	○	○	◉	1/1	✓
status register mode bit	○	◉	○	○	1/1	✓
system call interface	○	○	◉	○	1/1	✓

✓ When a process is running but then it needs to wait for an I/O operation to complete, the OS transitions the process to which state? * 5/5

- ◉ blocked ✓
- ○ running
- ○ ready

✓ When a process is in the ready state and the OS decides to schedule/run this process, then the OS transitions the process to which state? * 5/5

- (●) running ✓
- ( ) ready
- ( ) blocked

✓ In Linux/Unix, immediately after a process is created, what does it share with its parent process? * 5/5

- ( ) instructions/code
- ( ) register values
- ( ) file descriptors
- ( ) memory contents
- (●) none of the above. the child process has copies of these items but does not actually share any of them with its parent ✓

For the following questions use this information about process arrivals and processing time.

Process ID	Arrival Time	Processing Time
A	0	6
B	2	2
C	3	8
D	4	5
E	5	3

For the Shortest Job First scheduling policy, indicate the Delay for each process. ("Delay" is defined to be the amount of time the process spends in the 'Ready' state) *

	0	2	3	4	5	7	13	other	Score
A	◉	○	○	○	○	○	○	○	1/1
B	○	○	○	◉	○	○	○	○	1/1
C	○	○	○	○	○	○	◉	○	1/1
D	○	○	○	○	○	◉	○	○	1/1
E	○	○	◉	○	○	○	○	○	1/1

For the Shortest Job First with Pre-emption scheduling, indicate the Delay for each process. ("Delay" defined to be the amount of time the process spends in the 'Ready' state). If two processes are ready (or running) and have the same remaining processing time, then schedule the most-recently arrived process first. *

	0	2	3	4	5	7	13	other	Score
A	○	◉	○	○	○	○	○	○	0/1
B	◉	○	○	○	○	○	○	○	1/1
C	○	○	○	○	○	○	◉	○	1/1
D	○	○	○	○	○	◉	○	○	1/1
E	○	○	◉	○	○	○	○	○	0/1

✓ Each process has its own address space * 5/5

- ○ False
- ◉ True ✓

✓ Once a process's data (stack, heap, code, etc.) is loaded into physical memory the OS will not change the location of the data in physical memory. * 5/5

- ○ True
- ◉ False ✓

✓ In a paging-based virtual memory system, all physical page frames are always in use. * 5/5

- ○ True
- ◉ False ✓

✓ In a paging-based virtual memory system, consecutive pages for a process are stored contiguously in physical memory. * 5/5

- ○ True
- ◉ False ✓

✓ The OS reconfigures or flushes (erases) the TLB on each context switch. * 5/5

- ◉ True ✓
- ○ False

✓ In a paging system, the OS needs to choose the page size. What is the benefit of choosing a large page size for a paging-based virtual memory system? * 5/5

- [x] faster address translation ✓
- [ ] reduces external fragmentation
- [x] increased TLB hit rate ✓
- [x] less overhead from page faults ✓
- [ ] reduces internal fragmentation

System with 24 bit addresses and 2K page size

Consider the following system:

physical address space size: 24 bits
page size is 2K

✓ Within every 24 bit memory address, how many bits will be used to determine the page number? * 5/5

- ( ) 11
- ( ) 24
- ( ) 10
- (•) 13 ✓
- ( ) Other: _____

✓ How does the OS allow read-only memory pages to be shared across processes? * 5/5

- ○ it uses multiple CPU cores to achieve parallel processing
- ◉ it configures the appropriate page table entries in multiple processes to refer to a single page frame in physical memory. ✓
- ○ it uses a paging daemon to share the pages
- ○ it duplicates the memory pages in physical memory
- ○ it swaps the shared pages to a shared swap device

✓ Why is COW (copy on write) particularly effective during creation of new processes. * 5/5

- ○ COW is a faster form of exec()
- ○ COW flushes the TLB on every page write
- ◉ COW allows us to avoid most of the memory page copying required by the semantics of the fork() system call. ✓
- ○ COW allows both parent and child to share register state

✓ All processes running on a computer share the same page table. * 5/5

- ○ True
- ◉ False ✓

✓ Pinned pages will not be evicted from memory until they are either unpinned or the process terminates. * 5/5

- (●) True ✓
- ( ) False

✓ Updates to a page cause the OS to write the page to swap space immediately so that data is not lost if the system crashes. * 5/5

- ( ) True
- (●) False ✓

✓ An system can run more efficiently if the OS writes dirty pages to swap space proactively, long before the page needs to be paged out. * 5/5

- (●) True ✓
- ( ) False

✓ Why don't "Conflict" misses happen in TLBs? * 5/5

- [x] because the TLB is fully associative, so there is no hashing or aliasing of page addresses ✓
- [ ] because any TLB cache location can be used to map any page address
- [ ] because they are pre-loaded, thereby eliminating conflicts
- [ ] because the size of the TLB is large enough to avoid the conflicts
- [ ] because the TLB is accessed multiple times per CPU instruction

✓ if we have a 32 bit address space and our page size is 4096 bytes, then how many page addresses will be kept in the TLB? * 5/5

- [ ] 2^20
- [ ] 64
- [x] unknown. the TLB is just a cache and therefore can be any size. ✓
- [ ] 4096

✓ why don't Operating Systems generally use simple single-level page tables? * 5/5

- [x] because they waste memory space, especially when a process's use of its address space is sparse ✓
- [ ] because a single-level page table cannot fit into the TLB
- [ ] because of speed of lookup
- [ ] because they make development of application programs more difficult

What is the meaning of each of the administrative bits in a page table entry? *

	D bit	R bit	W bit	V bit	Score	
indicates that the page has been been modified recently and the modifications have not yet been written to swap space.	◉	○	○	○	1/1	✓
indicates that the page is writeable	○	○	◉	○	1/1	✓
indicates that the page has been referenced (read or written) recently	○	◉	○	○	1/1	✓
indicates that the page is valid, loaded into memory	○	○	○	◉	1/1	✓

Page Replacement Simulation

Consider the following page reference string.

A,E,D,B,C,C,D,D,A,A,D,C

Assume that there are 3 available empty page frames in physical memory and that all three page frames are empty.

Simulate these different page replacement algorithms (FIFO, LRU and OPT), count the page faults, and give your results below.

Indicate the number of page faults for each policy *

	5	6	7	8	9	10	other	Score	
OPT	◉	○	○	○	○	○	○	5/5	✓
LRU	○	◉	○	○	○	○	○	5/5	✓
FIFO	○	○	◉	○	○	○	○	5/5	✓

✓ What is Temporal Locality? * 5/5

- ○ the memory hierarchy of a computer is extremely important for application performance
- ○ reads occur more frequently than writes
- ○ accesses to addresses are usually followed by accesses to nearby addresses
- ◉ that which was accessed recently will be accessed again soon ✓

✓ Why might the OS sometimes suspend/swap out entire processes? * 5/5

- ◉ to reduce likelihood of thrashing ✓
- ○ to allow the remaining processes to have enough physical memory
- ○ to minimize page fault frequency
- ○ to improve temporal locality of page access

Google Forms