CS330 - End Semester Exam (Part I)

Total marks: 20 **Duration: 25 mins**

> The exam is open books, notes, printed materials. You are allowed web access. Consulting/discussing with anyone which includes asking questions in any forum during the exam is strictly prohibited. *Unless specified explicitly, assume 64-bit X86 architecture with multiple CPUs.*
> *All the best!*

1. Select the correct answer(s) for the following questions. Note that, you have to *write all correct choices* to get the credits (no partial marks). No explanation required. $\boxed{2.5 \times 4 = 10}$

(a) Consider two threads T_1 and T_2 of a process P. Which of the following statements are *true*.

A. T_1 can access a pointer declared in the global scope after it is initialized by T_2

B. T_1 can access an address after it is allocated using `mmap` by T_2

C. T_1 can access a variable after it is allocated in the stack area of P

D. T_1 can access a variable after it is allocated in the stack area of T_2

E. T_1 can read from `abc.txt` using `fd` (declared as a global variable) after T_2 successfully executes `fd = open("abc.txt", O_RDWR)`.

(b) Which of the following C/assembly statement(s) *may* result in a user-to-kernel mode switch? Assume that all addresses are virtual addresses and RAX and RCX are general purpose registers.

A. `pop RAX; // Store the value at current SP into RAX`

B. `MOV RAX, RCX; // Move the value in register RAX to RBX`

C. `inc (RAX); // increment the value at memory address in RAX`

D. `add RAX, RCX; // RAX = RAX + RCX`

E. `ret; // Return from the current function`

(c) Which of the following statement(s) is/are true regarding the page fault handler behavior where the page fault handler returns without resulting in any kind of fault and no signals like segmentation fault are raised to the faulting process? Note that, page fault handler can be invoked because of many reasons as discussed in the lectures. Assume a 4-level page table and 48-bit virtual address with a page size of 4KB.

A. *At least one level* of the page table is required to be updated

B. *Exactly one level* of the page table is required to be updated

C. *At most four levels* of the page table is required to be updated

D. *At least one TLB entry* is flushed by the page fault handler

E. *At most one TLB entry* is flushed by the page fault handler

(d) Consider the following pseudocode.

```
sem_t S1, S2;    /* S1 and S2 are initialized to zero */
string(){                               number(){
  printf("A");                          printf("1");
  wait(S1);                             post(S1);
  wait(S2);                             printf("2");
  printf("B");                          post(S2);
  post(S1);                             wait(S1);
  printf("C");                          wait(S2);
  post(S2);                             printf("3");
}                                       }
```

Which of the following can be a correct output if one thread executes `string()` and the other thread executes `number()` concurrently?

A. A12B3C

B. A12BC3

C. 123ABC

D. 1A2BC3

E. 12ABC3

2. True or False with justification. Justification should be a precise argument supporting your verdict. No marks without proper reasoning. $\boxed{2 \times 5 = 10}$

(a) Assuming that the OS is part of the process address space (split-address mode), the OS can invoke a function in the user code segment (by using assembly instructions like `call` or `jmp`).

(b) Allocation of kernel stack for a newly created process using `fork` can be delayed till it is scheduled for the first time.

(c) Context switch overhead of shortest time to completion first (STCF) scheduling policy is always less than round robin (RR) scheduling policy.

(d) If a process invokes `munmap(ptr, 8192)`, the handler of the system call *may* require flushing the TLB entries corresponding to the address range `ptr` to `ptr + 8192`.

(e) In a file system employing write-back page caches (like linux page cache as explained in the lectures), synchronous and unbuffered write to a file (opened with `O_DIRECT` and `O_SYNC` flags) will not require any lookup of the file offset (to be written to) in the page cache.