CS503 Spring 2019 – Final (A)

- Write your **name** in the first page and your **initials in each page**.
- The exam is closed notes, closed book and no collaboration is allowed.
- Please print or write legibly.
- Provide complete but concise answers.
- Answer the questions in the spaces provided. Your answer does not need to take up all the space provided. If you run out of room, continue on the back.
- Provide clear answers and justify statements where required.
- Unless otherwise noted, assume that questions refer to C code, AT&T assembly syntax and the x86 architecture.
- Please check you have all pages (10 pages).
- You have 120 minutes to complete the exam.

Question:	1	2	3	4	5	6	7	8	9	10	11	Total
Points:	20	10	5	5	10	10	10	15	10	5	0	100
Bonus Points:	0	0	0	0	0	0	0	0	0	0	10	10
Score:												

Name (please print): ______________________________

1. State whether the following statements are true of false. **Explain the false statements.**

   (a) (2 points) Application developers do not have to synchronize accesses to shared memory because it is done automatically by the OS.

   ◯ True
   ◯ False. Explanation:

   Solution sketch: False. Application developers may have to synchronize access to shared memory.

   (b) (2 points) A type 2 virtual machine monitor does not require a host OS and does not require a guest OS.

   ◯ True
   ◯ False. Explanation:

   Solution sketch: False. Requires a host and guest.

   (c) (2 points) The Xinu low-level memory manager manages the allocation of stack space but not heap space.

   ◯ True
   ◯ False. Explanation:

   Solution sketch: False. It does both.

   (d) (2 points) In x86, the same vector table is used for external interrupts and exceptions

   ◯ True
   ◯ False. Explanation:

   Solution sketch: True.

   (e) (2 points) To invalidate a page table entry the operating system needs to clear the entire page table entry.

   ◯ True
   ◯ False. Explanation:

   Solution sketch: False. Needs to clear the present bit.

   (f) (2 points) The Xinu *freebuf* function requires exactly two parameters: the buffer address and the pool ID.

   ◯ True
   ◯ False. Explanation:

   Solution sketch: False. Only the buffer address.

   (g) (2 points) Every executed x86 instruction causes at least one memory access that goes through the MMU.

○ True
○ False. Explanation:

Solution sketch: True.

(h) (2 points) A smaller page size leads to smaller page tables

○ True
○ False. Explanation:

Solution sketch: False. Leads to larger page tables

(i) (2 points) A SSD drive does not trigger page faults.

○ True
○ False. Explanation:

Solution sketch: True.

(j) (2 points) Distributed shared memory tends to have lower latency than traditional shared memory.

○ True
○ False. Explanation:

Solution sketch: False. DSM tends to have higher latencies.

2. (10 points) Explain the trade-off between a first-fit and a best-fit memory allocation strategy. Assume there is only one free area and a single free block list.

   Solution sketch: Fist-fit: faster, does not have to search till the end of the free list, Best-fit: reduces fragmentation.

3. (5 points) Explain what happens to an application when the main function of the application returns. Explain how Xinu implements this mechanism.

   Solution sketch: Process terminates. When the OS creates a new process it pre-initializes the stack with certain values. In particular, at the location of the main function return address, it pre-initializes the stack with the address of a kernel function that when executed terminates the currently running process.

4. (5 points) Describe what is temporal locality and what is spatial locality. Explain the importance of locality in computer systems.

Solution sketch: Temporal locality: memory accesses to the same address that are close in time. Spatial locality: memory accesses to nearby addresses that are close in time. Enable caching.

Initials: ____________

5. (10 points) The program bellow selectively adds the elements of two matrix depending on the values of a binary matrix. Describe how you would change the program to increase spacial locality. Describe another to increase temporal locality. Explain how each of the two changes increases locality. Assume the compiler does not optimize the code, in particular, assume that all variables are stored in memory and every reference to a variable causes either a store or a load on the variable address.

```
 1 #include <stdio.h>
 2 #define N 128
 3
 4 int main(){
 5         int matrix1[N][N];
 6         int matrix2[N][N];
 7         int select[N][N];
 8         int sum = 0,i,j;
 9
10         // assume appropriate matrix initialization
11
12         // calculate selective sum
13         for (i = 0; i < N; i++){
14                 for(j = 0; j < N; j++){
15                         if(select[i][j]==1)
16                                 sum += matrix1[i][j];
17                 }
18         }
19
20         for (i = 0; i < N; i++){
21                 for(j = 0; j < N; j++){
22                         if(select[i][j]==1)
23                                 sum += matrix2[i][j];
24                 }
25         }
26         printf("\%d",sum);
27 }
```

Solution sketch: ”For instance, spatial locality: ”int select[][]” -¿ ”char select[][]” (select becomes denser), temporal locality: move statement in line 22 to line 17 (”select[i][j]” is accessed in closer temporal proximity)

6. (10 points) In systems that use segmentation but not paging, physical memory may become fragmented. In such situations, there may not exist a contiguous memory region of a given size despite enough free memory. Explain how can the operating system satisfy a memory allocation request when sufficiently large contiguous memory is not available. Assume protected mode segmentation. Recall that segmentation in protected mode relies on segment descriptors that define the base address and the limit of the segment.

   Solution sketch: The operating system can relocate the segments by moving the contents of the segments and accordingly adjusting the base address of the segments.

7. (10 points) What is an idempotent operation and why are they useful when dealing with a system with failures. Include an example of an idempotent operation in your answer.

   Solution sketch: Operations that when applied multiple times produce the same result as if applied only once. Example op: Write value 1 to address 0x12345.

8. Consider a 32-bit CPU with two-levels of paging (i.e., directory + page table), 4 KB pages, and page directory with 256 entries (i.e., 2^8 entries). Assume directory and page table entries have 4 bytes.

   (a) (5 points) What is the maximum amount of memory used by the entire page table structure?

   Solution sketch: 8 bits to index directory, 12 bits for pages (32-8-12 = 12 bits) to index page tables. Directory size: 256 * 4 bytes = 1KB. Page table size: $2^{12} * 4$ bytes = 16KB. Total: 1 KB + 4 MB (256 page tables).

   (b) (10 points) Explain how you would initialize the page table hierarchy to ensure that every virtual address maps to the same physical address in a memory efficient manner. Explain clearly what is (a) the starting address of each page table and page directory and (b) the contents of each of their entries. You can use formulas to avoid listing individually this information for each table/entry.

   Solution sketch: For instance, starting addresses: PD=0x10000, PT=0; Contents: PD[i]=0, PT[i]=i.

9. (10 points) x86 instructions have variable length, some are 1 byte long but others can reach 10 bytes. `int 03` is used by debuggers to create breakpoints. Explain how debuggers use this instruction in Linux. Does the size of the instruction matter? Explain.

> **Solution sketch:** Debuggers use the instruction to implement breakpoints. Yes, it should have the smallest instruction size to ensure that the debugger can replace any instruction with “int 03” without affecting other program instructions.

10. (5 points) Explain the concept of OS-level virtualization (e.g., containers). Provide two advantages of OS-level virtualization over virtual-machine based virtualization. Explain how you would implement such a service in Xinu.

> **Solution sketch:** Main aspect: applications inside a container run in an isolated environment (e.g., cannot see or interact with other application). Advantages: higher density, faster startup.

11. (10 points (bonus)) Bob and Alice were working on an OS project in a team of two. They wrote a program that only contained three "fork()" calls that printed the process ID after each fork. They were shocked to see that the program produced different outputs when Bob and Alice ran it on their respective machines. Provide three reasons why this could have happened and explain why. Assume that both hardware and software is correct and system configuration is the same.

    Solution sketch: Race on process create (different IDs), print race, stdout flush vs. fork race.