University of Washington – Computer Science & Engineering

Winter 2021 Instructor: Mark Wyse Due: March 17, 2021

CSE 351 Study Guide 3 – Task 3

Last Name:	
First Name:	
UW NetID (username):	
Academic Integrity Statement: All work on these questions is my own. I had no prior knowledge of the questions, and I will not share or discuss my answers with anyone else. Violation of these terms may result in a failing grade. (**please sign**)	

Instructions

- Fill in your name and UW NetID in the table above, then read the Academic Integrity Statement and sign your name in the box to the right of it indicating that you understand and will comply with the statement.
- If able, fill in your responses on a copy of this document and upload your submission to Gradescope when completed.
- When uploading to Gradescope, mark which pages correspond to each question.
- Show scratch work for partial credit but answer in the blanks, boxes, or spaces provided.
- You may use your study guide from Task 1, course lecture slides and Ed Lessons, and course textbooks while completing this task.
- Use of reference materials external to those listed above is not allowed (e.g., Stack Overflow, web searches, etc.)
- These questions should take approximately 30 minutes to answer.
- Refer to the Study Guides webpage for additional information: https://courses.cs.washington.edu/courses/cse351/21wi/guides/

Advice

- Read each question carefully.
- Relax and breathe; you are here to learn.
- Modules: Processes/System Control Flow, Virtual Memory, Memory Allocation. Java and C

Part 1. Processes and System Control Flow

(A) A Process provides two key abstractions. What are they?

(B) Name the type of synchronous exception for each of the following definitions:

a. An unintentional, but possibly recoverable exception caused by the execution of an instruction.

b. An unintentional and unrecoverable exception, caused for example by a machine check, that stops the execution of the current program.

c. An intentional transfer of control to the Operating System (Kernel) to perform some function as requested by the user program.

(C) Provide one possible ordering of the print outputs from the following code (list one output per line in the box provided):

```
void foo() {
  printf("foo\n");
  if (fork() == 0) {
    printf("first\n");
  }
  if (fork() == 0) {
    printf("second\n");
  }
  printf("bye\n");
}
```

Part 2. Virtual Memory

For this part, assume the computer system has the following characteristics:

- 16-bit virtual addresses
- 14-bit physical addresses
- Page size of 512 bytes
- TLB holds 8 entries and is 2-way associative
- Little-endian memory system

Assume the system has the following initial state:

TLB

Set	Tag	Valid	PPN	Tag	Valid	PPN
0	0x1E	1	0x02	0x1F	1	0x01
1	0x03	0	0x0F	0x04	1	0x03
2	0x1C	0	0x11	0x02	1	0x04
3	0x07	1	0x18	0x03	0	0x04

Page Table (first 16 entries only)

PTE	Valid	PPN	PTE	Valid	PPN
0	0	-	**8**	1	0x0A
1	1	0x00	**9**	0	0x0B
2	1	0x10	**10**	1	0x04
3	1	0x0F	**11**	1	0x1D
4	0	-	**12**	0	0x11
5	0	-	**13**	0	-
6	1	0x17	**14**	0	-
7	0	-	**15**	1	0x0E

(A) Computer the number of bits required for each entry in the table below.

VPN	VPO	TLB Tag	TLB Index	PPN	PPO

(B) How many virtual pages and physical pages does this system have?

Virtual Pages	Physical Pages

(C) How many entries are in a single process's page table in this system?

(D) For each of the following virtual addresses, *provide the VPN, VPO, TLBT, and TLBI (in hex)*. Then, *determine if a TLB Hit occurs and if a Page Fault occurs (mark "Y" or "N" for each)*. If possible, *provide the PPN and compute the physical address (in hex)*. If the PPN cannot be determined, leave the PPN and PA entries blank.

VA	VPN	VPO	TLB Tag	TLB Index	TLB Hit (Y/N)	Page Fault (Y/N)	PPN	PA
0x2201								
0xF00D								

(E) True or False: On a context-switch the current process' page table must be invalidated? Circle your answer.

True / False

(F) True or False. On a context-switch the TLB must be invalidated? Circle your answer.

True / False

This space intentionally left blank.

Part 3. Memory Allocation – True/False – Circle your answer

(A) In the context of memory allocators, and compared to an implicit free list, an allocator using an explicit free list is much faster at allocating blocks when most of the memory is free (unused).

True / False

(B) When programming in C, there is no need to explicitly deallocate memory that was dynamically allocated.

True / False

(C) When dynamically allocating memory in C, the malloc() function call will always succeed and return a block of memory to the caller.

True / False

(D) The malloc() routine (memory allocator) in C can freely relocate previously allocated blocks.

True / False