Fall 2015 COMP 3511 Homework Assignment #3 Solution

Handout Date: Oct 28, 2015 Due Date: Nov 11, 2015

Name: _____ ID: _____
E-Mail: _____Section: _____

Please read the following instructions carefully before answering the questions:

- You should finish the homework assignment **individually**.
- There are a total of **4** questions.
- When you write your answers, please try to be precise and concise.
- Fill in your name, student ID, email and Section number at the top of each page.
- Please fill in your answers in the space provided, or you can type your answers in the MS Word file.

- **Homework Collection: the hardcopy** is required and the homework is collected in **collection box #16** (for **L1**) and **collection box #17** (for **L2**). The collection boxes are located outside **Room 4210**, near **Lift 21** (there are labels on the boxes)

1. (20 points) Multiple choices

   1) A mutex lock ____.
   A) is exactly like a counting semaphore
   B) is essentially a boolean variable
   C) is not guaranteed to be atomic
   D) can be used to eliminate busy waiting
   Answe: B

   2) Assume an adaptive mutex is used for accessing shared data on a Solaris system with multiprocessing capabilities. Which of the following statements is not true?
   A) A waiting thread may spin while waiting for the lock to become available.
   B) A waiting thread may sleep while waiting for the lock to become available.
   C) The adaptive mutex is only used to protect short segments of code.
   D) Condition variables and semaphores are never used in place of an adaptive mutex.
   Answer: D

   3) How many philosophers may eat simultaneously in the Dining Philosophers problem with 5 philosophers?
   A) 1
   B) 2
   C) 3
   D) 5
   Answer: B

   4) A cycle in a resource-allocation graph is ____.
   A) a necessary and sufficient condition for deadlock in the case that each resource has more than one instance
   B) a necessary and sufficient condition for a deadlock in the case that each

resource has exactly one instance
C) a sufficient condition for a deadlock in the case that each resource has more than once instance
D) is neither necessary nor sufficient for indicating deadlock in the case that each resource has exactly one instance
Answer: B

5) Which of the following statements is true?
A) A safe state is a deadlocked state.
B) A safe state may lead to a deadlocked state.
C) An unsafe state is necessarily, and by definition, always a deadlocked state.
D) An unsafe state may lead to a deadlocked state.
Answer: D

6) Suppose that there are ten resources available to three processes. At time 0, the following data is collected. The table indicates the process, the maximum number of resources needed by the process, and the number of resources currently owned by each process. Which of the following correctly characterizes this state?

Process	Maximum Needs	Currently Owned
P_0	10	4
P_1	3	1
P_2	6	4

A) It is safe.
B) It is not safe.
C) The state cannot be determined.
D) It is an impossible state.
Answer: B

7) _____ is the method of binding instructions and data to memory performed by most general-purpose operating systems.
A) Interrupt binding
B) Compile time binding
C) Execution time binding
D) Load-time binding
Answer: C

8) _____ is the dynamic storage-allocation algorithm which results in the smallest leftover hole in memory.
A) First fit
B) Best fit
C) Worst fit
D) None of the above
Answer: B

9) Assume a system has a TLB hit ratio of 90%. It requires 15 nanoseconds to access the TLB, and 85 nanoseconds to access main memory. What is the effective memory access time in nanoseconds for this system?
A) 108.5
B) 100

C) 22
D) 176.5
Answer: A

10) Given the logical address 0xAEF9 (in hexadecimal) with a page size of 256 bytes, what is the page number?
A) 0xAE
B) 0xF9
C) 0xA
D) 0x00F9
Answer: A

2. (20 points) Please answer the following questions in a few sentences.

1) (4 points) What are the main differences between deadlock prevention and deadlock avoidance?

Answer: Deadlock prevention is a set of methods for ensuring that at least one of the necessary conditions for deadlock cannot hold. Deadlock avoidance requires that the operating system be given, in advance, additional information concerning which resources a process will request and use during its lifetime.

2) (6 points) One way to eliminate the circular-wait condition is to impose a total ordering of all resource types, for instance it requires that each process requests resources in an increasing order of enumeration – R = {R1, R2, …, Rm}. Please sketch a proof that this ensures no circular-wait (Hint: proof by contradiction)

Answer: Suppose there is a circular-wait, implying that there exists a set $\{P_0, P_1, \ldots, P_n\}$ of waiting processes such that P_0 is waiting for a resource α_0 that is held by P_1, P_1 is waiting for a resource α_1 that is held by P_2, …, P_{n-1} is waiting for a resource α_{n-1} that is held by P_n, and P_n is waiting for a resource α_n that is held by P_0.

Then according the ordering constrain, since P_1 is holding α_0 and requesting α_1, we have $\alpha_0 < \alpha_1$, which means α_0 precedes α_1 in the given enumeration. Similarly we have $\alpha_1 < \alpha_2$, …, $\alpha_{n-1} < \alpha_n$, $\alpha_n < \alpha_0$. Here $\alpha_0 < \alpha_1 < \ldots < \alpha_n$ is contradictory with $\alpha_n < \alpha_0$. So there is no circular-wait.

3) (5 points) How does external fragmentations occur? How could we resolve this problem?

Answer: External fragmentation exists when there is enough total memory space to satisfy a memory request, but the available spaces are not contiguous; storage is fragmented into a large number of small holes, each of which may be insufficient to

satisfy the request (3 points). Compaction technique can be used to solve this problem, but it is costly (2 points)

4) (5 points) What is TLB? How can it improve the performance of memory access?

Answer: A translation lookaside buffer (TLB) is a memory cache that stores recent translations of virtual memory to physical addresses for faster retrieval.

When a virtual memory address is referenced, the operating system will first check if the required page number is in TLB. If yes then the corresponded frame number of the requested page is extracted from the TLB entry, which would save one memory access time. Otherwise the system has to look up the page table to conduct that translation.

3. (20 points) Consider the following snapshot of a system:

	Allocation	Max	Available
	A B C D	A B C D	A B C D
P_0	2 0 0 1	4 2 1 2	3 3 2 1
P_1	3 1 2 1	5 2 5 2	
P_2	2 1 0 3	2 3 1 6	
P_3	1 3 1 2	1 4 2 4	
P_4	1 4 3 2	3 6 6 5	

Answer the following questions using the banker's algorithm:

1) (8 points) Illustrate that the system is in a safe state by demonstrating an order in which the processes may complete.

Answer: A feasible execution sequence: P0, P3, P4, P1, P2.

2) (6 points) If a request from process P1 arrives for (1, 1, 0, 0), can the request be granted immediately?

Answer: Yes.

1) Since request < available, calculate Need = Max-Allocate

Need

	A	B	C	D
P_0	2	2	1	1
P_1	2	1	3	1
P_2	0	2	1	3
P_3	0	1	1	2
P_4	2	2	3	3

2) Since request1 < need1, update available,

	Allocation				Need				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P_0	2	0	0	1	2	2	1	1	2	2	2	1
P_1	4	2	2	1	1	0	3	1				
P_2	2	1	0	3	0	2	1	3				
P_3	1	3	1	2	0	1	1	2				
P_4	1	4	3	2	2	2	3	3				

The current state is safe, a feasible execution sequence is P0, P3, P1, P2, P4

3) (6 points) If a request from process P4 arrives for (0, 0, 2, 0), can the request be granted immediately?

Answer: No.

1) Since request < available, calculate Need = Max-Allocate

	Need			
	A	B	C	D
P_0	2	2	1	1
P_1	2	1	3	1
P_2	0	2	1	3
P_3	0	1	1	2
P_4	2	2	3	3

2) Since request4 < need4, update available,

Allocation	Need	Available

	A B C D	A B C D	A B C D
P_0	2 0 0 1	2 2 1 1	3 3 0 1
P_1	3 1 2 1	2 1 3 1	
P_2	2 1 0 3	0 2 1 3	
P_3	1 3 1 2	0 1 1 2	
P_4	1 4 5 2	2 2 1 3	

The current state is NOT safe, since available resource C is less than the need of any process.

4. (30 points) Memory management

1) (15 points) Consider the following segment table:

Segment	Base	Length
0	90	200
1	500	100
2	3500	150
3	160	10
4	1382	200

What are the physical addresses of the following logical address?

a) 0, 99

b) 1, 101

c) 2, 56

d) 3, 100

e) 4, 2

Answer:

a) 90 + 99 = 189

b) 101 > 100, illegal reference, trap to operating system

c) 3500 + 56 = 3556

d) 100 > 10, illegal reference, trap to operating system

e) 1382+2=1384

2) (15 points) Two-level paging

In a 32-bit machine we subdivide the virtual address into 3 segments as follows:

page number		page offset
10-bit	10-bit	12-bit

We use a two-level page table (in memory) such that the first 10 bits of an address is an index into the first level page table and the next 10 bits are an index into a second level page table. Each page table entry is 32 bits in size.

a) (2 points) What is the page size in such a system?

Answer: Because 12 bits are used to offset into a page, the page size is 2^{12} = 4096 bytes = 4KB.

b) (3 points) How many entries are in the 1st level page table? How many entries are in the 2rd level page table?

Answer: Since an index in the first level or second level page table is 10 bits, there must be 2^{10} =1024 =1K entries per page table (in both 1st and 2rd level page table).

c) (3 points) How much memory do the 1st page table occupy? How much memory do the 2rd page table occupy?

Answer: Since each page table entry is 4 bytes (32-bits), a page table (either first or second level) occupies: 4 bytes * 1K entries = 4KB

d) (7 points) How much space is occupied in memory by the page tables for a process that has 128MB of actual virtual address space allocated? Show your work with detailed explanation.

Answer:

Since the process uses 128MB = 2^{27} bytes and each page is 4KB = 2^{12} bytes, there must be $2^{27} / 2^{12} = 2^{15}$ pages (2 point)

Thus, we need a sufficient number of second level page tables to hold 2^{15} entries. Since each second level page table has 2^{10} entries, we need $2^{15} / 2^{10} = 2^5 = 32$ second level page tables (2 point)

Therefore, because there is one first-level page table and 32 second level page tables, the total space occupied in memory by the page tables is:

(1+32) * 4KB = 33 * 4KB = 132KB (3 point)