NORTHEASTERN UNIVERSITY
COLLEGE OF ENGINEERING
Department of Electrical & Computer Engineering

93

EECE7376: Operating Systems: Interface and Implementation — Dr. Emad Aboelela
Spring 2024 – Quiz 2 — Time Allowed: 1.5 Hour

Student Name: Student ID:

This is a **closed book** and **closed notes** quiz. The quiz has **5 questions**
You **are not allowed** to use any **electronic** device nor **scratch** papers
No answers outside these quiz pages will be graded (you can write on **both sides**)
Make sure to **write your full name** on the empty side of your cheat sheet
You cannot leave the room once the quiz starts (except in case of an emergency)

10

Q1. (10 Points)

Consider the memory segments table below:

Segment	Process	Virtual Base Address	Size	Physical Base Address
0	A	10	20	100
1	A	30	10	200
2	A	50	20	300

Assume all segments are positive growing. For each of the following memory access, fill out the empty slots in the table below. In case of a segmentation fault, enter FAULT in the Segment column.

Process	Virtual Address	Segment	Offset	Physical Address
A	35			
A	45			
A	60			

Q2. (20 Points)

$2^9 = 512$ ~

18

A page-based virtual memory is using the shown **linear page table** where its:

- Virtual address space size = 512 bytes
- Physical memory size = 4096 bytes
- Page size = 16 bytes

VPN	PTE
0	0x80000034
1	0x8000001E
	0x80000021
3 to 28	0x00000000
29	0x80000017
30	0x80000011
31	0x8000002E

The shown **VPN** numbers are in **decimal** while the **PTE** contents are in **hexadecimal** where 4 bytes are used to store each PTE. The leftmost bit in each PTE is the VALID bit. If this bit is 1, then the least significant *x* bits represent the **PFN** and if it is 0, the page is invalid.

5 a) What are the minimum number of bits needed to represent the **VPN**, **PFN**, and the page **offset**?

8 bit ✓

8/10 b) In front of each of the following virtual addresses, write its corresponding VPN in **decimal** and its corresponding physical address in **hexadecimal**. For an invalid virtual address, write FAULT for its physical address.

Virtual address	VPN in decimal	Physical Address in Hexadecimal
0x144		
0x1E7		
0x01C		
0x038		
0x006		

5 c) Given that 4 bytes are needed for every PTE entry in the above linear page table, how many bytes will be saved if a **multi-level page table** (by utilizing a page directory) is used instead? Assume every entry in the directory requires 4 bytes and it is used to represent 4 PTE entries from the above table.

] virtaddr

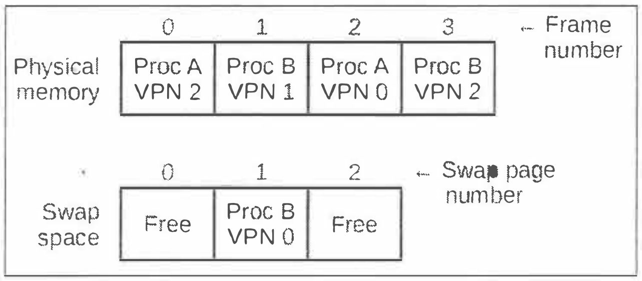
18

Q3. (20 Points)

Given the state of the physical memory and swap space represented on the shown diagram,

6/8 a) Fill in the contents of the linear page tables below for processes ***A*** and ***B***. In the **Valid** and **Present** columns, enter value 0 for non-present and invalid pages. Enter the value 1 for present and valid pages. Enter value -1 for a nonexistent PFN. -1

12 b) The columns for the **Dirty** and **Reference** attributes are not shown on the following table. Explain how the OS utilizes these two attributes to improve the performance of the memory paging technique.



Process	VPN Index	0/1 Valid	1/0/-1 Present	PFN Number
A	0			
A	1			—
A	2			
A	3			
B	0			
B	1			
B	2			
B	3 x		0	—

✓

bit setted.

17

Q4. (20 Points)

a) How many **page faults** occur using the **Least-Recently-Used (LRU)** page replacement policy algorithm for the following sequence of page references using **four-page** physical frames? Compare your result with the **optimal** replacement policy. Show all steps used to find the number of page faults caused by **both policies**. When you have a tie case, apply the **FIFO** policy.

9, 7, 9, 7, 6, 6, 7, 4, 1, 4, 1, 9, 7, 6, 4, 6

b) Repeat part (a) but for **Least-Frequently-Used (LFU)** page replacement policy. Explain why LFU is not a practical policy.

30

Q5. (30 Points)

a) In the page-based virtual memory systems, what is the main goal of utilizing a **translation-lookaside buffer** (**TLB**) and where it is stored?

5

b) How is that goal achieved given that the number of entries in a TLB are much less than those in its corresponding page table?

5

c) In the shown algorithm for virtual address translation using TLB there are two shift operations in lines 1 and 6. How many bits of shift are needed in these operations? Explain

5

```
 1  VPN = (VirtualAddress & VPN_MASK) >> SHIFT
 2  (Success, TlbEntry) = TLB_Lookup(VPN)
 3  if (Success == True) // TLB Hit
 4     if (CanAccess (TlbEntry.ProtectBits) == True)
 5       Offset = VirtualAddress & OFFSET MASK
 6       PhysAddr = (TlbEntry.PFN << SHIFT) | Offset
 7       Register = AccessMemory(PhysAddr)
 8     else RaiseException(PROTECTION_FAULT)
 9  else // TLB Miss
10    PTEAddr = PTBR + (VPN * sizeof(PTE))
11    PTE = AccessMemory(PTEAddr)
12    if (PTE .Valid == False)
13       RaiseException(SEGMENTATION_FAULT)
14    else
15       if (CanAccess(PTE.ProtectBits) == False)
16              RaiseException(PROTECTION_FAULT)
17       else // updates the TLB with the translation
18         TLB_Insert(VPN, PTE.PFN, PTE.ProtectBits)
19         RetryInstruction()
```

d) In the block of the TLB hit, why isn't there a check for segmentation faults the same way it is done on line 12 for a TLB miss?

5

5 e) The shown code does not show the needed formula to locate the entry corresponding to a VPN in the TLB the same way it does in Line 10 to calculate the index of the PTE in the page table. Explain how a VPN entry in the TLB is looked up.

done here!

```
 1  VPN = (VirtualAddress & VPN_MASK) >> SHIFT
 2  (Success, TlbEntry) = TLB_Lookup(VPN)
 3  if (Success == True) // TLB Hit
 4     if (CanAccess (TlbEntry.ProtectBits) == True)
 5       Offset = VirtualAddress & OFFSET MASK
 6       PhysAddr = (TlbEntry.PFN << SHIFT) | Offset
 7       Register = AccessMemory(PhysAddr)
 8     else RaiseException(PROTECTION_FAULT)
 9  else // TLB Miss
10    PTEAddr = PTBR + (VPN * sizeof(PTE))
11    PTE = AccessMemory(PTEAddr)
12    if (PTE .Valid == False)
13       RaiseException(SEGMENTATION_FAULT)
14    else
15       if (CanAccess(PTE.ProtectBits) == False)
16              RaiseException(PROTECTION_FAULT)
17       else // updates the TLB with the translation
18         TLB_Insert(VPN, PTE.PFN, PTE.ProtectBits)
19         RetryInstruction()
```

5 f) Briefly list three advantages of managing the TLB miss using software instead of hardware.

End of the Quiz Questions