


# PRACTICE FINAL EXAM QUESTIONS



May 30, 2024      TCSS422: Operating Systems [Spring 2024]  
School of Engineering and Technology, University of Washington - Tacoma      L19.91

91

## QUESTION 1 – BASE AND BOUNDS

- A computer system uses a simple base/bounds register pair to virtualize address spaces. For each traces fill in the missing values of virtual addresses, physical addresses, base, and/or bounds registers. In some cases, it is not possible to provide an exact value. If so, specify a range (e.g. greater than 100), or value that is not a single number.

Virtual Address	Physical Address		
0	500		
100	600		
300	800	Base?	500
699	1199		
700	[fault]	Bounds?	700

May 30, 2024      TCSS422: Operating Systems [Spring 2024]  
School of Engineering and Technology, University of Washington - Tacoma      L19.92

92

Q1 - 2

**Scenario 2]**

Virtual Address	Physical Address		
300	1500	Base?	<u>1200</u>
1600	2800		
1801	<u>3001</u> ?	Bounds?	<u>&gt; 2801</u>
2801	4001		<u>&gt; 2800</u>

**Scenario 3**

Virtual Address	Physical Address		
<u>0</u>	1000	Base?	<u>1000</u>
<u>100</u>	1100		
<u>1999</u>	2999	Bounds?	<u>2000</u>
<u>≥ 2000</u>	[fault]		

May 30, 2024
TCSS422: Operating Systems [Spring 2024]  
 School of Engineering and Technology, University of Washington - Tacoma
L19.93

93

QUESTION 2 – SINGLE-LEVEL PAGE TABLE

- Consider a computer with 4 GB ( $2^{32}$ ) of physical memory, where the page size is 4 KB ( $2^{12}$ ). For simplicity assume that 1GB=1000MB, 1MB=1000KB, 1KB=1000 bytes
- (a) How many pages must be tracked by a single-level page table if the computer has 4GB ( $2^{32}$ ) of physical memory and the page size is 4 KB ( $2^{12}$ )?  $2^{32}/2^{12} = 2^{20} = 1,000,000$
- (b) How many bits are required for the virtual page number (VPN) to address any page within this 4GB ( $2^{32}$ ) memory space? 20 bits
- (c) Assuming that the smallest addressable unit of memory within a page is a byte (8-bits), how many bits are required for the offset to refer to any byte in the 4 KB page? 12 bits
- (d) Assuming each page table entry (PTE) requires 4 bytes of memory, how much memory is required to store the page table for one process (in MB)?  $2^{20} \text{ entries} \times 4 \text{ bytes} = 2^{22} \text{ bytes} \rightarrow 4 \text{ MB}$

May 30, 2024
TCSS422: Operating Systems [Spring 2024]  
 School of Engineering and Technology, University of Washington - Tacoma
L19.94

94

## Q2 - 2

*4MB PER PAGE TABLE*

- (e) Using this memory requirement, how many processes would fill the memory with page table data on a 4GB computer? *1024* *4GB / 4 MB*

May 30, 2024	TCSS422: Operating Systems [Spring 2024] School of Engineering and Technology, University of Washington - Tacoma	L19.95
--------------	---	--------

95

## QUESTION 3 - TWO-LEVEL PAGE TABLE

- Consider a computer with 1 GB ( $2^{30}$ ) of physical memory, where the page size is 1024 bytes (1KB) ( $2^{10}$ ). We would like to index memory pages using a two level page table consisting of a page directory which refers to page tables which are created on demand to index the entire memory space.
- For simplicity assume that  $1\text{GB}=1000\text{MB}$ ,  $1\text{MB}=1000\text{KB}$ ,  $1\text{KB}=1000$  bytes  *$2^{30}/2^{10} \rightarrow 2^{20}$*
- (a) For a two-level page table, divide the VPN in half. How many bits are required for the page directory index (PDI) in a two-level scheme? *10 BITS*
- (b) How many bits are required for the page table index (PTI)? *10 BITS*
- (c) How many bits are required for an offset to address any byte in the 1 KB page? *10 BITS*

May 30, 2024	TCSS422: Operating Systems [Spring 2024] School of Engineering and Technology, University of Washington - Tacoma	L19.96
--------------	---	--------

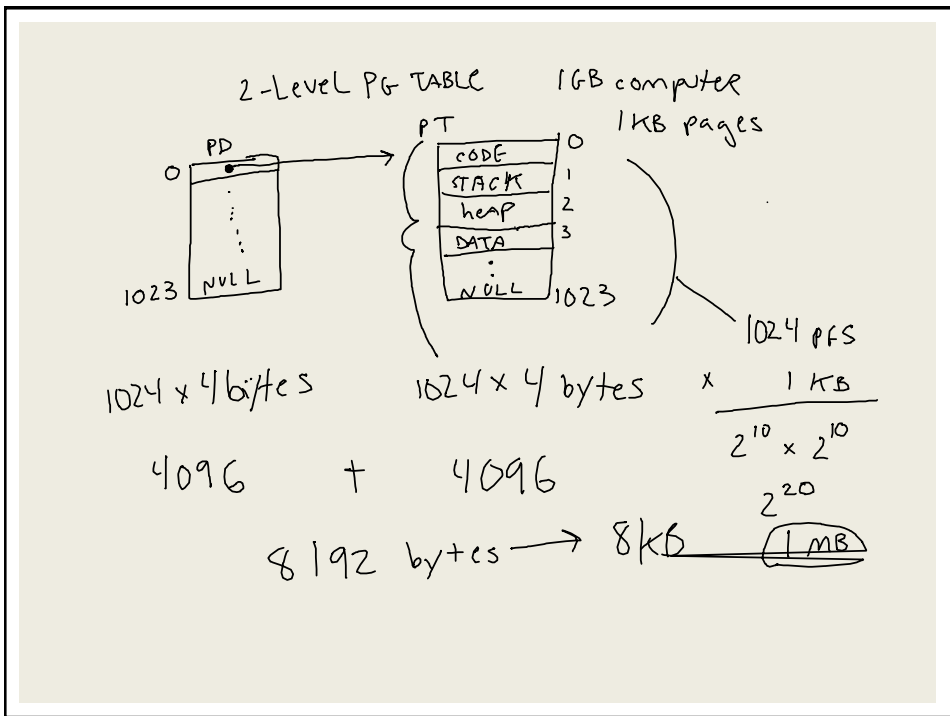
96

Q3 - 2

- (d) Assuming each page table entry (PTE) requires 4 bytes of memory, how many extra bits are available for status bits?  
 $12$        $12 \text{ BITS}$
- (e) HelloWorld.c consists of 4 memory pages. One code page, one heap page, one data segment page, and one stack segment page. How large is the two-level page table in bytes with the structure described above that could index the all 4 memory pages of HelloWorld.c?  
 $8192$   
*Hint: There should be 2 tables, a page directory, and a page table.*
- (f) Assuming the same page table as for HelloWorld.c, using the exact same two-level page table, how large in bytes could the program grow to before needing to expand the page table?  
 $2^{10} \times 2^{10} \rightarrow 2^{20} \rightarrow 1 \text{ MB}$

May 30, 2024
TCSS422: Operating Systems [Spring 2024]  
School of Engineering and Technology, University of Washington - Tacoma
L19.97

97



98

QUESTION 4 – CACHE TRACING

- Consider a 3-element cache with the cache arrival sequences below.
- Determine the number of cache hits and cache misses using each of the following cache replacement policies:

**A. Optimal policy**

Arrival sequence:  
 m m m H H m m H H m m m H H  
 5 3 7 5 3 1 0 7 1 6 4 3 2 1 3

# Hits: \_\_\_\_\_ 6

# Misses: \_\_\_\_\_ 9

Working Cache

Cache 1: ~~5~~ 1

Cache 2: ~~5~~ ~~3~~ 2

Cache 3: ~~5~~ ~~3~~ 3

May 30, 2024

TCSS422: Operating Systems [Spring 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L19.99

99

Q4 - 2

**B. FIFO policy**

Arrival sequence:  
 m m m H H m m H H m m m m m H  
 5 3 7 5 3 1 0 7 1 6 4 3 2 1 3

# Hits: \_\_\_\_\_ 5

# Misses: \_\_\_\_\_ 10

Working Cache

Cache 1: ~~5~~ ~~3~~ ~~4~~ 1

Cache 2: ~~5~~ ~~3~~ 3

Cache 3: ~~5~~ ~~3~~ 2

May 30, 2024

TCSS422: Operating Systems [Spring 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L19.100

100

Q4 - 3

**C. LRU policy**

Arrival sequence:  
 m m m H H m m m H m m m m m H  
 5 3 7 5 3 1 0 7 1 6 4 3 2 1 3

**Working Cache**  
 Cache 1: 7 5 3 1 0 7 1 6 4 3  
 Cache 2: 3 7 5 3 1 0 7 1 6 4  
 Cache 3: 5 3 7 5 3 1 0 7 1 6

# Hits: 4

# Misses: 11

May 30, 2024

TCSS422: Operating Systems [Spring 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L19.101

101

QUESTION 5 – FREE SPACE MANAGEMENT

- Free space management involves capturing a description of the computer's free memory using a data structure, storing this data structure in memory, and OS support to rapidly use this structure to determine an appropriate location for new memory allocations. An efficient implementation is very important when scaling up the number of operations the OS is required to perform.
- Consider the use of a linked list for a free space list where each node is represented by placing the following structure in the header of the memory chunk:
 

```
typedef struct __node_t
            {
                int size;
                struct __node_t *next;
            } node_t;
```

May 30, 2024

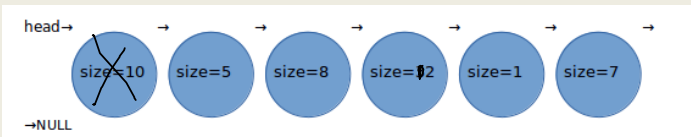
TCSS422: Operating Systems [Spring 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L19.102

102

## Q5 - 2

■ Consider the following free space list:



■ (a) Consider the **next fit** allocation strategy. For this free list above, how many comparison operations must be performed to identify a free chunk of **30-bytes**? 4

■ (b) After the last free space identification, the chunk is split and the remaining free space is returned to the free space list. Now, consider the **next fit** allocation strategy. After finding a free space for the previous request, how many comparisons are required to identify a free chunk of 10-bytes? 4

May 30, 2024	TCSS422: Operating Systems [Spring 2024] School of Engineering and Technology, University of Washington - Tacoma	L19.103
--------------	---	---------

103

## Q5 - 3

■ Now, after the last free space identification the chunk is split and the remaining free space is returned to the free space list. Now consider each of the following free space allocation strategies. How many comparisons are required on the updated free space list to find a free chunk of 2 bytes using:

■ (c) best fit? 5

■ (d) worst fit? 5

■ (e) first fit? 1

May 30, 2024	TCSS422: Operating Systems [Spring 2024] School of Engineering and Technology, University of Washington - Tacoma	L19.104
--------------	---	---------

104