Q1 Virtual Memory Concepts
32 Points

For the questions below, answer true/false and justify your choice.

Q1.1 Memory Compaction
8 Points

Memory compaction is usually used with fixed-size partition memory allocation scheme.

True [✓ False]

Why is this statement true/false?

The whole point of compaction is to reduce the external fragmentation of the variable-sized partition memory allocation scheme. What compaction does is reorder all of the memory so that all of the used memory is in one contiguous block and all of the unused memory is another contiguous block of memory.

The fixed-size partition memory allocation scheme suffers from internal fragmentation, not external fragmentation. The fixed-size scheme has already allocated all of the memory it has (thus no memory gets unallocated). Whether all the memory in those fixed-size partitions is used is a different question (and the answer is almost always no and because not all the memory we have unnecessary waste (this waste is called internal fragmentation)). Because fixed-sized memory allocation already does NOT suffer from external fragmentation, there is no need to compact it.

Q1.2 Paging and Segmentation
8 Points

Paging and Segmentation both use the Page Table for virtual memory since they work the same way in practice.

True [✓ False]

Why is this statement true/false?

Although both the paging and segmentation are similar, they require different tables.

The page table has two parts: The virtual page number (VPN) and the physical frame number (PFN) as shown in figure 7.10 of the textbook.

The segmentation table has two parts: The start address and the size of the segmented block. It carries the size because it needs to do a calculation on whether the offset of the virtual address is a valid offset (if the offset is bigger than the size then it is not a valid address and an Illegal Address Exception is thrown as shown in figure 7.19).

Beacause the two tables have different parts, they cannot be used for both paging and segmentation.

Q1.3 Base and Limit Register
8 Points

There is no special advantage for the base and limit register solution over the bounds register solution for memory management.

True [✓ False]

Why is this statement true/false?

This question is more or less "True or false: there is no reason to use dynamic reallocation (base and limit register) over static reallocation (bounds registers).

Static reallocation "...constrains memory management and leads to poor memory utilization (Ramachandran 7-7)." The best example of this is in the case of swapping out programs. Static reallocation reserves a particular region in memory for the program upon linking it (the linker is the program that reserves the region in memory). The loader then sets the bound values of this program and those bounds never change throughout the lifetime of the program. As an example of the swapping problem, let's say the program is between addresses 15000 and 16000. The bounds are therefore 15000 and 16000. For some reason the memory manager swaps this program out for another one, and when it comes time to swap it back in, another process is using the addresses 15000 to 16000. The swapped-out process now must wait for those addresses to become available again before it can be continued.

With Dynamic Reallocation, the allocated memory of the program is assigned during execution. In the above example for static allocation, with dynamic allocation, the swapped-out process could be swapped back in even if its original memory addresses are in use by another program if there is remaining memory somewhere; it doesn't have to wait for those addresses to become available again if there is available memory somewhere else.

Hardware wise, there is no advantage from using bounds registers or base + limit registers (they both require 2 arithmetic operations). The bounds registers require 2 registers and 2 comparators (a greater than and less than comparator) whilst the base + limit registers require 2 registers and an addition block of hardware and a comparator block of hardware.

Q1.4 External Fragmentation
8 Points

Paged virtual memory can eliminate external fragmentation.

[✓ True] False

Why is this statement true/false?

Paged Virtual Memory is similar to fixed-sized partitions in that it allocates "page-sized" blocks of memory that can be accessed with a page table. Because each block can be reached through the page table, there isn't any memory block that can't be reached and thus paged-virtual memory circumvents external fragmentation since the pages do not need to be contiguous in physical memory. Paged-Virtual memory does suffer from internal fragmentation, however, due to its fixed-size pages.

Q2 Addresses and Page Tables
28 Points

Our operating system uses 32-bit virtual addresses, 20-bit physical addresses, and page sizes of 8KB.

For Q2.1 and Q2.2, complete the layout of the virtual and physical addresses by labelling each field with the correct value, as well as denoting the bit size for each field.

Q2.1 Virtual Address Layout
8 Points

Fill in the layout below with the appropriate fields for the **VIRTUAL address**.

Label 1	Label 2

What part of the virtual address is `Label 1`?

Virtual Page Number (VPN)

What part of the virtual address is `Label 2`?

Page Offset

What is the bit size of `Label 1`?

19

What is the bit size of `Label 2`?

13

Q2.2 Physical Address Layout

8 Points

Fill in the layout below with the appropriate fields for the **PHYSICAL address**.

Label 1	Label 2

What part of the physical address is `Label 1`?

Physical Frame Number (PFN)

What part of the physical address is `Label 2`?

Page Offset

What is the bit size of `Label 1`?

7

What is the bit size of `Label 2`?

13

Q2.3 Page Table Entries

6 Points

How many entries are there in the page table? Format your answer as a base 10 number (standard number). So if your answer is 2^4, write "16".

524288

Q2.4 Page Frames

6 Points

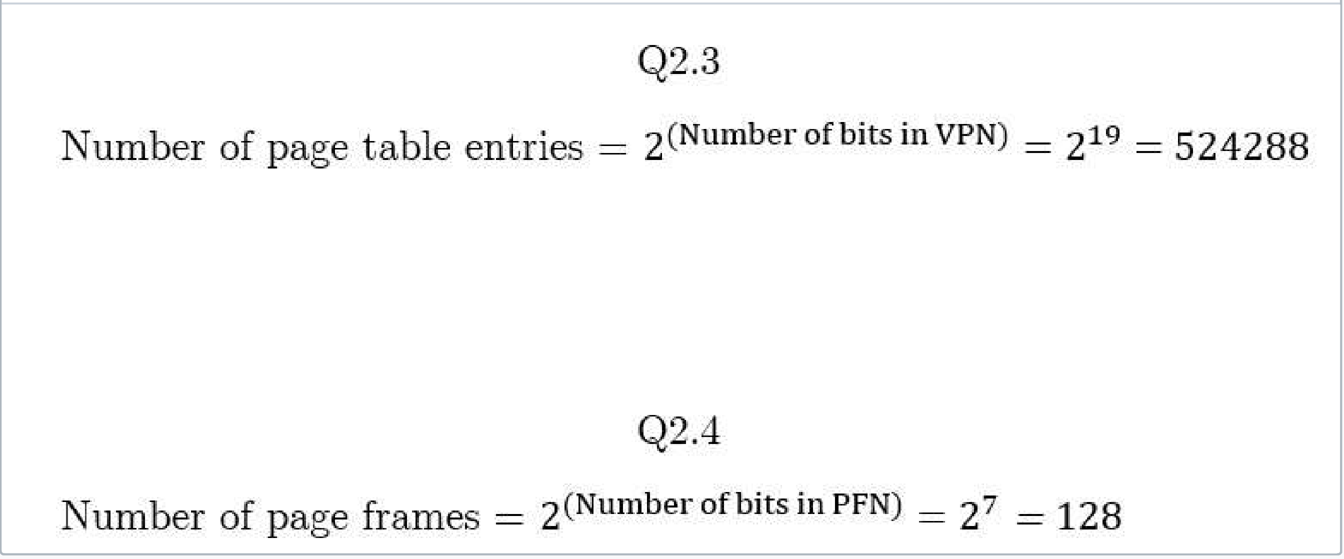
How many page frames does the memory system have? (assume there are no hardware limitations) Again, format your answer as a base 10 number (standard number).

128

Q2.5 Work
0 Points

In order to receive partial credit for an incorrect answer for Q2.3 and Q2.4, please show your work and attach it below. **Incorrect answers with no work shown will receive 0 points.**

hw7 q2.5.jpg Download



Q2.3

$$\text{Number of page table entries} = 2^{(\textbf{Number of bits in VPN})} = 2^{19} = 524288$$

Q2.4

$$\text{Number of page frames} = 2^{(\textbf{Number of bits in PFN})} = 2^{7} = 128$$

See attached image.

Q3 Page Faults
16 Points

Say we have virtual addresses of 32 bits, physical addresses of 28 bits, and a page size of 4KB. Given the following page table entries from the process' page table, answer the following questions.

VPN	PFN	Valid?
35FCA	2A2F	V
47F99	CAFE	V
4ECD5	BEEF	I
50AAC	223D	V

Q3.1 Offset Bits
4 Points

How many bits does the offset in an address have?

12

Q3.2 Address Translation
8 Points

You are given a virtual address of 0x47F99A30. According to the table above, what is its corresponding physical address?

0xCAFEA30

Q3.3 Page Fault
4 Points

Which of the following VPNs is held by a Page Table Entry that may lead to a page fault?

- ○ 35FCA
- ○ 47F99
- [✓ 4ECD5]
- ○ 50AAC
- ○ None of the above

Q4 Second Chance Replacement
24 Points

Provide short answers to the following questions on the Second Chance page replacement algorithm.

Q4.1 Structures Needed
12 Points

What data structures and bookkeeping features are needed to implement Second Chance Replacement? Explain how the algorithm selects a page for eviction.

Because the Second Chance Replacement Algorithm is an extension of the FIFO algorithm, it uses a circular queue as its backing data structure. In addition to the circular queue, it also has a reference bit such that if it is set when the reaches the top of the queue and is selected to be evicted, then the algorithm will set its reference bit back to zero and add it to the back of the queue thus giving it a second chance.

The way the algorithm works is as follows:
1.) The page is added to the queue and initially, its reference bit is 0. At some point, the hardware may decide that it wants the reference bit to be 1.
2.) When the queue fills up and a page needs to be evicted, it goes to the front of the queue and checks the reference bit of that page.
3.) If its reference bit of the selected page is 0, then the page is evicted. If the reference bit is a 1, then it clears the reference bit and moves it to the back of the queue. Then, the algorithm goes to the new page at the front of the queue and checks its reference bit. The process continues until the algorithm finds a page with an unset reference bit.

Q4.2 Degenerates to FIFO
12 Points

Second Chance Replacement can degenerate to FIFO replacement. Explain the circumstances in which this might occur.

If the replacement bit is never set, then the second chance replacement algorithm will act as a FIFO replacement since no page will ever get a second chance and the page evicted will be the page that has been in the queue the longest.

Eric Anders Gustafson

TOTAL POINTS

99 / 100 pts

QUESTION 1

Virtual Memory Concepts **32** / 32 pts

1.1	Memory Compaction	**8** / 8 pts
1.2	Paging and Segmentation	**8** / 8 pts
1.3	Base and Limit Register	**8** / 8 pts
1.4	External Fragmentation	**8** / 8 pts

QUESTION 2

Addresses and Page Tables **28** / 28 pts

2.1	Virtual Address Layout	**8** / 8 pts
2.2	Physical Address Layout	**8** / 8 pts
2.3	Page Table Entries	**6** / 6 pts
2.4	Page Frames	**6** / 6 pts
2.5	Work	**0** / 0 pts

QUESTION 3

Page Faults **16** / 16 pts

3.1	Offset Bits	**4** / 4 pts
3.2	Address Translation	**8** / 8 pts
3.3	Page Fault	**4** / 4 pts

QUESTION 4

Second Chance Replacement **23** / 24 pts

4.1 **Structures Needed** **11** / 12 pts

− 0 pts	Correct
− 3 pts	Did not mention circular queue/circular queue-like function/FIFO (just need to mention one) Ex: frame table or separate queue
− 3 pts	Did not mention referenced bit
− 3 pts	Did not mention clearing referenced bit while going through queue
− 3 pts	Did not mention selecting the first victim without a referenced bit set
− 12 pts	Blank/no answer
− 1 pts	A correction for part 1- the reference bit for a page is not set when the hardware happens to decide that it is- the reference bit is set on a memory access (aka reference) to that page.

4.2 Degenerates to FIFO **12** / 12 pts