CS2106 Operating Systems 2425 Sem 1 28 Nov 2024

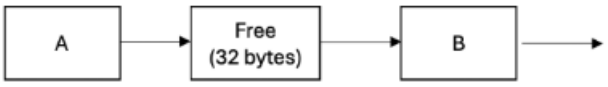
Your Name: _____ **Your ID:** _____
# of Questions: 11
Date and Time of Exam Creation: Mon, Nov 25, 2024 @ 12:02:57
Total Exam Points: 50.00

Question #: 1

We consider a system with 256 bytes of memory and 8-byte allocation units (i.e. memory is always allocated in units of 8 bytes). All addresses are byte addresses. The current state of the memory is shown below. If a partition is allocated to a request, the request name (a single capital letter, e.g. A, B, C etc) is shown. Otherwise, the partition is shown as "Free".

Start Address	End Address	Status
0	63	A
64	79	Free
80	103	B
104	127	Free
128	191	C
192	223	Free
224	231	D
232	255	Free

We use the next-fit allocation algorithm to allocate memory. In this algorithm, we start the search for the first free partition from the beginning of the free list.
For subsequent requests, we continue from where we stopped. If we performed an allocation, we resume our search at the very next free partition from where we complete our allocation. For example, suppose we have the following memory state:



Suppose we allocate 24 bytes in the free partition above to C, we will get:

The search for the next free partition will start at the 8-byte free space shown above. Note that an operation to free a partition **does not** make that partition the next one to be searched.

We have the following five requests:

1. E: Allocate 20 bytes
2. F: Allocate 12 bytes
3. Free memory allocated to D
4. G: Allocate 30 bytes
5. H: Allocate 12 bytes

i. (10 marks of 22): What are the starting addresses of E, F, G and H?

E: 1

F: 2

G: 3

H: 4

1. _____
2. _____
3. _____
4. _____

Item Weight: 10.0

Question #: 2

We consider a system with 256 bytes of memory and 8-byte allocation units (i.e. memory is always allocated in units of 8 bytes). All addresses are byte addresses. The current state of the memory is shown below. If a partition is allocated to a request, the request name (a single capital letter, e.g. A, B, C etc) is shown. Otherwise, the partition is shown as “Free”.

Start Address	End Address	Status
0	63	A
64	79	Free
80	103	B
104	127	Free
128	191	C
192	223	Free
224	231	D
232	255	Free

We use the next-fit allocation algorithm to allocate memory. In this algorithm, we start the search for the first free partition from the beginning of the free list.
For subsequent requests, we continue from where we stopped. If we performed an allocation, we resume our search at the very next free partition from where we complete our allocation. For example, suppose we have the following memory state:

Suppose we allocate 24 bytes in the free partition above to C, we will get:

The search for the next free partition will start at the 8-byte free space shown above. Note that an operation to free a partition **does not** make that partition the next one to be searched.
We have the following five requests:

1. E: Allocate 20 bytes
2. F: Allocate 12 bytes
3. Free memory allocated to D
4. G: Allocate 30 bytes
5. H: Allocate 12 bytes

ii) (1 mark of 22) What is the total internal fragmentation after completing all of the requests above? Do not include memory allocated to A, B and C since we don't know what the original requests were.

Answer: 1 bytes

iii) (1 mark of 22) How much free memory is left after all the requests have been fulfilled?

Answer: 2 bytes

1. _____
2. _____

Item Weight: 2.0

Question #: 3

Q1bi:

We now consider a system again with 256 bytes of memory, but this time with an allocation unit of 1 byte (i.e. we can allocate blocks as small as 1 byte), and buddy allocation. We always allocate from the lowest address block first.
So if we have two buddy blocks of 64 bytes at address 0 and 64, we allocate the block at address 0 first.
The memory is currently in the state shown below. As before, a partition allocated to a request is indicated by a single alphabet, while free partitions are indicated by the word “Free”. To help you, the binary encoding of the starting addresses is shown.

Start Address	End Address	Status
0 0b0000 0000	15	Free
16 0b0001 0000	31	A
32 0b0010 0000	63	Free
64 0b0100 0000	127	B
128 0b1000 0000	159	C
160 0b1010 0000	255	Free

i) (8 marks of 22) We have the following four requests:

1. E: Allocate 10 bytes
2. F: Allocate 92 bytes
3. G: Allocate 28 bytes
4. Free(B)

Fill in the table below showing the state of the memory after fulfilling all of the requests above. Two or more adjacent blocks of free memory must be shown as a single contiguous block; for example, if you have free memory from addresses 64 to 95 and 96 to 127, be shown as a single contiguous block from 64 to 127, or it will be marked as incorrect.

Memory State. In status fill a letter (e,g, A, B, G, etc) of the request if the partition is allocated to a request or “Free” without the quotes if the partition is free in the table shown on Examplify:

(Note: due to a limitation in Examplify, only show the first five rows of the memory state table):

Start Address	End Address	Status
1	2	3
4	5	6
7	8	9
10	11	12
13	14	15

1. _____
2. _____
3. _____
4. _____
5. _____
6. _____
7. _____
8. _____
9. _____
10. _____
11. _____
12. _____
13. _____
14. _____
15. _____

Item Weight: 8.0

Question #: 4

Q1bii.

We now consider a system again with 256 bytes of memory, but this time with an allocation unit of 1 byte (i.e. we can allocate blocks as small as 1 byte), and buddy allocation. We always

allocate from the lowest address block first.

So if we have two buddy blocks of 64 bytes at address 0 and 64, we allocate the block at address 0 first.

The memory is currently in the state shown below. As before, a partition allocated to a request is indicated by a single alphabet, while free partitions are indicated by the word “Free”. To help you, the binary encoding of the starting addresses is shown.

Start Address	End Address	Status
0 0b0000 0000	15	Free
16 0b0001 0000	31	A
32 0b0010 0000	63	Free
64 0b0100 0000	127	B
128 0b1000 0000	159	C
160 0b1010 0000	255	Free

ii) (2 marks of 22): What is the total internal fragmentation after all the requests in part i. are fulfilled? Do not count memory already in use before the requests were made.

Answer: 1 bytes

1. _____

Item Weight: 2.0

Question #: 5

Q2a (2 marks of 12)

We consider a system with a 32-bit virtual address space and a 24-bit physical address space. Pages/frames are 16 KiB (1 KiB $=2^{10}$ bytes) long.

Every page table entry (PTE) consist of a frame number, three access bits (wrx), an “in-memory bit”, and a dirty bit.

a. (2 marks of 12) How large is each page table entry, rounded up to the nearest byte?

Answer: 1 bytes

1. _____

Item Weight: 2.0

Question #: 6

Q2b (3 marks of 12)
We consider a system with a 32-bit virtual address space and a 24-bit physical address space. Pages/frames are 16 KiB (1 KiB $=2^{10}$ bytes) long.
Every page table entry (PTE) consist of a frame number, three access bits (wrx), an “in-memory bit”, and a dirty bit.

b. (3 marks of 12) If we had every entry of the page table in memory, how large would this page table be? State your answer in KiB (1 KiB $=2^{10}$ bytes)

Answer: 1 KiB

1. _____

Item Weight: 3.0

Question #: 7

We consider a system with a 32-bit virtual address space and a 24-bit physical address space. Pages/frames are 16 KiB (1 KiB = 2^{10} bytes) long.
Every page table entry (PTE) consist of a frame number, three access bits (wrx), an “in-memory bit”, and a dirty bit.

c) (4 marks) If we used multilevel page tables, what is the maximum number of levels that we would have if PTEs are the same size at every level, each PTE is 4 bytes long, and we fully make use of a page to store PTEs? The page table directory counts as one level, although it may not occupy an entire page and may have fewer entries than the lower level page tables.

Answer: 1 levels

1. _____

Item Weight: 4.0

Question #: 8

We consider a system with a 32-bit virtual address space and a 24-bit physical address space. Pages/frames are 16 KiB (1 KiB = 2^{10} bytes) long.
Every page table entry (PTE) consist of a frame number, three access bits (wrx), an “in-memory bit”, and a dirty bit.

c) (3 marks of 12) We have an array of size 64 MiB (1 MiB = 2^{20} bytes) starting at address 0x2FCDFB3C. How much memory is used by your page tables, including the page directory, in our multilevel page table? Express your answer in bytes.

Answer: 1 bytes.

1. _____

Item Weight: 3.0

Question #: 9

Consider a system with a single-level page table, and with the following characteristics:

TLB Hit Rate	96%
TLB Access Time	2 ns
Memory Access Time (Read)	50 ns
Cache Hit Rate	97%
Cache Access Time	1 ns
Page Fault Rate	2%
Disk Access Time	25 ms

Note: 1 ns = 10^{-9} s, 1 us = 10^{-6} s, 1 ms = 10^{-3} s

Some important points:

1. Some pages are dirty and may need to be written back to disk before being replaced.
2. The time taken to see if a page table entry is in TLB is 2 ns. If the page table entry is in TLB, the time taken to get the frame number is negligible.
3. In the event of a TLB miss, the TLB is not re-read after remedying the miss.
4. The time taken to check the cache for a hit is 1 ns. If there is a cache hit, the time taken to read/write the cache block is negligible.
5. In the event of a cache miss, the cache needs to be re-read after remedying the cache miss. We ignore the time taken to write the memory block to cache.
6. The page table entries are only cached in the TLB and not in the memory cache.
7. In the event of a page fault, the faulting memory location must be re-accessed after remedying the page fault.

8. For any answer that is not an integer, provide your answer to **three decimal places**.
9. Ensure that you provide your answers in the units shown.

a. (1 mark of 8) What is the worst-case access time in this memory hierarchy? State your answer in ns.

Answer: 1 ns

b. (1 mark of 8) What is the best-case access time in this memory hierarchy? State your answer in nsns

Answer: 2 ns

1. _____
2. _____

Item Weight: 2.0

Question #: 10

Consider a system with a single-level page table, and with the following characteristics:

TLB Hit Rate	96%
TLB Access Time	2 ns
Memory Access Time (Read)	50 ns
Cache Hit Rate	97%
Cache Access Time	1 ns
Page Fault Rate	2%
Disk Access Time	25 ms

Note: 1 ns = 10^{-9} s, 1 us = 10^{-6} s, 1 ms = 10^{-3} s

Some important points:

1. Some pages are dirty and may need to be written back to disk before being replaced.
2. The time taken to see if a page table entry is in TLB is 2 ns. If the page table entry is in TLB, the time taken to get the frame number is negligible.
3. In the event of a TLB miss, the TLB is not re-read after remedying the miss.
4. The time taken to check the cache for a hit is 1 ns. If there is a cache hit, the time taken to read/write the cache block is negligible.

5. In the event of a cache miss, the cache needs to be re-read after remedying the cache miss. We ignore the time taken to write the memory block to cache.
6. The page table entries are only cached in the TLB and not in the memory cache.
7. In the event of a page fault, the faulting memory location must be re-accessed after remedying the page fault.
8. For any answer that is not an integer, provide your answer to **three decimal places**.
9. Ensure that you provide your answers in the units shown.

c. (6 marks of 8) What is the average memory access time in this memory hierarchy, assuming that 25% of the pages chosen for replacement are dirty and must be written back? State your answer in ns.

Answer: 1 ns

1. _____

Item Weight: 6.0

Question #: 11

Consider an inode-based file system with the following characteristics:

Size of each block: 4 KiB (1 KiB = 1024 bytes)
Size of each block pointer: 4 bytes
Number of direct pointers: 12
Number of single indirect pointers: 2
Number of double indirect pointers: 1
Number of triple indirect pointers: 1
Average time taken to read a block: 12 ms (1 ms = 10^{-3} s)

For any answer that is not an integer, give your answer to **three decimal places**.

a. (2 marks) What is the maximum number of blocks possible on this partition? State your answer to two decimal places in billions of blocks (1 billion = 10^{9})

Answer: 1 billion blocks

b. (2 marks) What is the maximum file size possible in this file system? Express your answer to two decimal places in GiB (1 GiB = 2^{40} bytes)

Answer: 2 GiB

In the following parts assume that the inode for the file is already loaded into memory, and we include the time it takes to read the actual data block.

c. (2 marks) What is the best-case average time for reading data from a file? Express your answers in milliseconds (1 ms = 10^{-3} seconds)

Answer: 3 ms

d. (2 marks) What is the worst-case average time for reading data from a file? Express your answers in milliseconds (1 ms = 10^{-3} seconds)

Answer: 4 ms

1. _____
2. _____
3. _____
4. _____

Item Weight: 8.0