

Q1

1. A particular computer system provides a total memory of 4800 words to users. It supports multiprogramming with multiple-partition. At a given moment in time, the memory is occupied by four processes:

Starting Address of Process	Length (words)
1000	1200
3100	700
4000	400
4400	200

1

Q1 (2)

When a new process requests memory, the following method is employed:

- STEP 1: Use the Best-fit algorithm to allocate memory for this new process. Do Step 2 only if there is no hole which fits the memory request.
- STEP 2: Do memory compaction by moving occupied partitions towards lower memory addresses. Compaction stops when a sufficiently large hole is formed for the new memory request. If, at the end of compaction, there is still insufficient space, the process requesting for more memory is blocked.

2

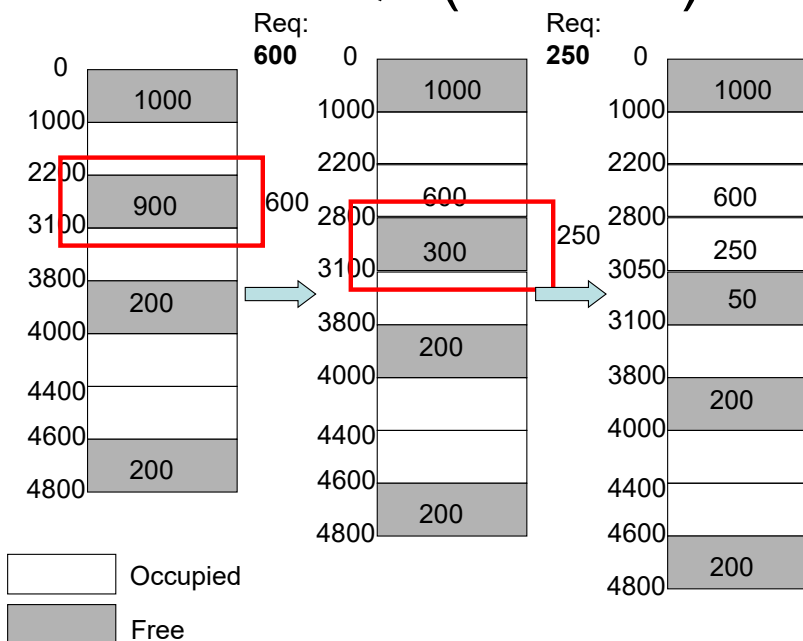
Q1 (3)

The Operating System receives three new memory requirements from processes: 600, 250 and 1050 words (requested in this order).

- Show the memory allocation after all three requests have been serviced.
- If the algorithm in Step 1 had been First-fit, show the memory allocation after all three requests have been serviced.
- In this particular case, which algorithm yields a lower overhead in terms of total relocation of occupied partitions?

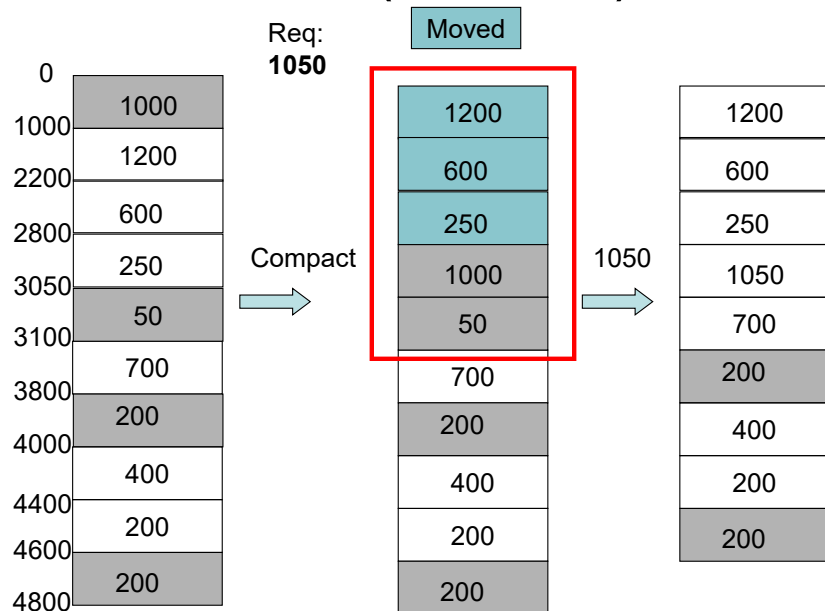
3

Q1 (Best Fit)



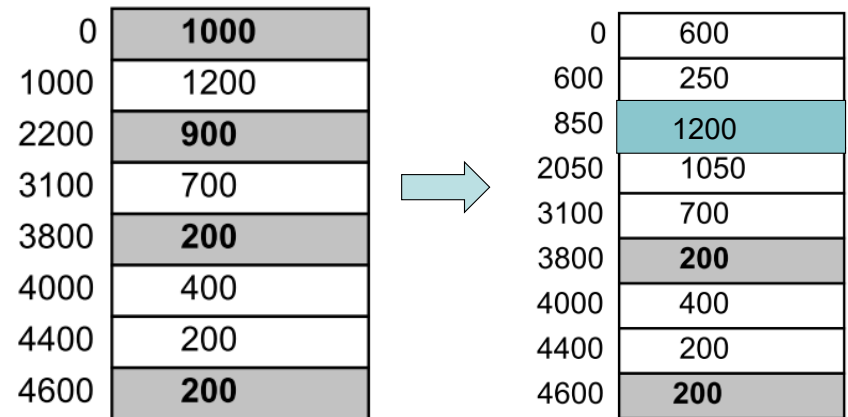
4

Q1 (Best Fit)



5

Q1 (b): First Fit



6

Q1 (c)

The total movement of partitions:

- Best Fit: 2050 words (1200 + 600 + 250)
- First Fit: 1200 words.
- In this particular case, First Fit has lower overhead.

Q2

- Identify memory allocation schemes that suffer from internal fragmentation.
- Explain why memory compaction cannot be performed if absolute address format is used in the code.

7

8

Q2a

2. a) Identify memory allocation schemes that suffer from internal fragmentation.

→ fixed partitioning and paging

9

Q2

2. b) Explain why memory compaction cannot be performed if absolute address format is used in the code.

→ If absolute address is used, the code and data cannot be moved around in the memory (otherwise, addressing error happens).

→ Compaction requires moving code and data in the memory and so it cannot be performed.

10

Q3 Memory Access Time

Consider a paging system with the page table stored in memory.

a) If a memory reference takes 200 nanoseconds, how long does a paged memory reference take?

b) If we add associative registers, and 75 percent of all page-table references are found in the associative registers, what is the effective memory reference time? (Assume that finding a page-table entry in the associative registers takes zero time, if the entry is there.)

11

Q3 (a)

- 400 nanoseconds, two memory accesses:
 - One for the page table
 - One for the word in memory.

12

Q3 (b)

- Access time
 - 200, when the reference in associative registers.
 - 400, otherwise.
- Effective access time = $0.75 \times (200 \text{ nanoseconds}) + 0.25 \times (400 \text{ nanoseconds}) = 250 \text{ nanoseconds}$.

13

Q4

Consider a computer system with a 32-bit logical address and 1-Kbyte page size. The system has 1 Gbytes of physical memory.

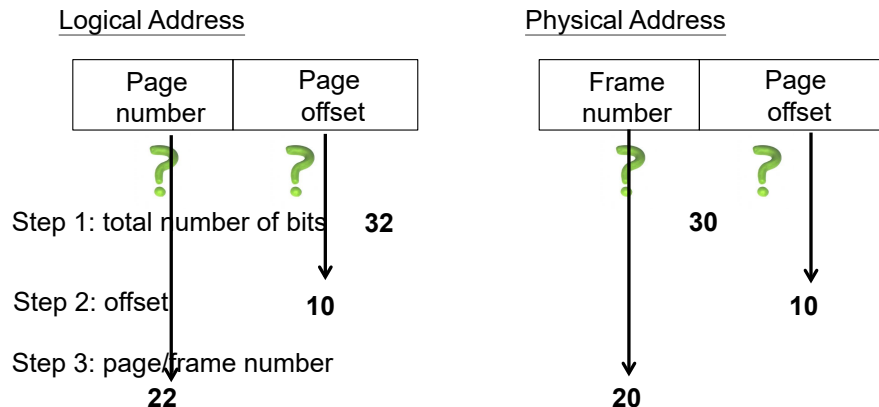
- Give the format of both the logical and physical addresses of this system.
- How many entries are there in a page table?
- If an inverted page table is used, how many entries are there?

14

Q4 (a)

Consider a computer system with a 32-bit logical address and 1-Kbyte page size. The system has 1 GBytes of physical memory.

- Give the format of both the logical and physical addresses of this system.

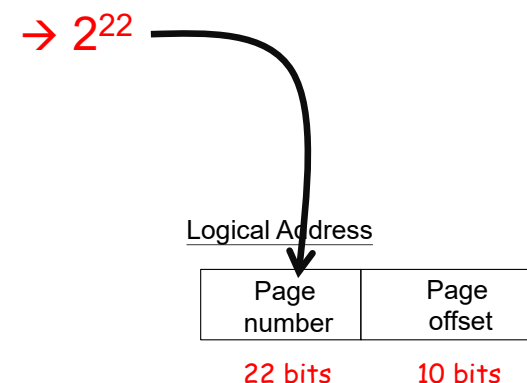


15

Q4 (b)

Consider a computer system with a 32-bit logical address and 1-KByte page size. The system has 1 GBytes of physical memory

- How many entries are there in a page table?



16

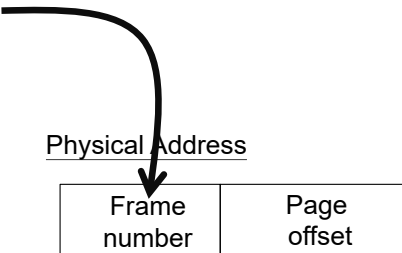
Q4 (c)

Consider a computer system with a 32-bit logical address and 1-Kbyte page size. The system has 1 Gbytes of physical memory

c) If an inverted page table is used, how many entries are there?

→ 2^{20}

Physical Address



A diagram showing a curved arrow pointing from the red text 2^{20} to the 'Frame number' field of a table. The table has two columns: 'Frame number' and 'Page offset'.

Frame number	Page offset
-----------------	----------------

20 bits

10 bits