1a. A logical address is the virtual address generated by the CPU that a user can both see and access, while a physical address is a location in a memory unit that the user can neither see nor access.

1b. The contiguous memory allocation assigns consecutive blocks of memory to a process while the noncontiguous memory allocation assigns separate blocks of memory to a process.

1c.  First-fit placement allocates the first free partition or hole large enough which can accommodate the processes, while best-fit placement allocates the smallest free partition which meets the requirement of the requesting processes.

2. First-fit: 17kb

Best-fit: 17kb

Next-fit: 17kb

Worst-fit: 25kb

3a. Overlays only require part of program code or other data to be placed into internal memory, which reduces memory and time requirement.

3b. In compaction, all the free partitions are made contiguous and all the loaded partitions are brought together, which reduces fragmentation.

4a.

|  |  |
| --- | --- |
| Page # | Frame # |
| 1 | 6 |
| 2 | 7 |
| 3 | 10 |
| 4 | 12 |
| 5 | 18 |
| 6 | 20 |
| 7 | 21 |

4b.

Logical map

|  |  |
| --- | --- |
| Frame | Contents |
| 1 | aa bb cc dd |
| 2 | ee ff gg hh |
| 3 | ii jj kk ll |
| 4 | mm nn oo pp |
| 5 | qq rr ss tt |
| 6 | uu vv ww xx |
| 7 | yy zz |

Physical map

|  |  |
| --- | --- |
| Frame | Contents |
| 1 | aa bb cc dd |
| 2 | ee ff gg hh |
| 3 | ii jj kk ll |
| 4 | mm nn oo pp |
| 5 | qq rr ss tt |
| 6 | uu vv ww xx |
| 7 | yy zz |

4c.

|  |  |
| --- | --- |
| Page # | Frame # |
| 1 | 6 |
| 2 | 7 |
| 3 | 10 |
| 4 | 12 |
| 5 | 18 |
| 6 | 20 |
| 7 | 21 |

4d.

bb: 6\*4+1 = 25

ff: 7\*4+1 = 29

rr: 18\*4+1 = 73

vv: 20\*4+1 = 81

4e. In physical map table frame 21 uses only 2 bytes out of 4 bytes so 2 bytes wasted, so Internal fragmentation = 2

External fragmentation = 0

5a. OPT (X denotes a page fault)

8 7 6 8 5 7 6 4 3

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 8 | 8 | 8 | 8 | 5 | 5 | 5 | 5 | 5 |
| - | 7 | 7 | 7 | 7 | 7 | 7 | 4 | 4 |
| - | - | 6 | 6 | 6 | 6 | 6 | 6 | 3 |
| X | X | X |  | X |  |  | X | X |

Total page fault = 6

Page Fault Rate = (6/9) \* 100 = 66.67%

5b. LRU (X denotes a page fault)

8 7 6 8 5 7 6 4 3

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 8 | 8 | 8 | 8 | 5 | 5 | 5 | 5 | 5 |
| - | 7 | 7 | 7 | 7 | 7 | 7 | 4 | 4 |
| - | - | 6 | 6 | 6 | 6 | 6 | 6 | 3 |
| X | X | X |  | X | X | X | X | X |

Total page fault = 8

Page Fault Rate = (8/9) \* 100 = 88.89%

6.

Installing a faster CPU will not improve utilization because a faster CPU reduces the CPU utilization further since the CPU will spend more time waiting for a process to enter in the ready queue.

Installing a bigger paging disk doesn’t improve utilization because the size of the paging disk does not affect the amount of memory that is needed to reduce the page faults.

Increasing the degree of multiprogramming will not improve utilization because each process would have fewer frames available, and the page fault rate would increase.

Decreasing the degree of multiprogramming will improve utilization because by suspending some of the processes, and the other processes will have more frames to bring their pages in them, hence reducing the page faults.

Installing more main memory will probably improve utilization because more pages can remain resident and don’t require paging to or from the disks.

Increasing the page size will not improve utilization because each process would have fewer frames available, and the page fault rate would increase.

7.

At the start of the execution process, the page fault rate is high because as the necessary number of pages are not yet loaded by the memory.

After the working set of the process is loaded into memory, the page fault is much lower because the necessary number of pages have been loaded into the memory.

8.

To handle the free memory issue, you could either get more physical memory or reclaim pages more aggressively due to the high page fault rate.