1.

a)

$$R = \begin{pmatrix} 0\,0\,0\,0 \\ 0\,7\,5\,0 \\ 1\,0\,0\,2 \\ 0\,0\,2\,0 \\ 0\,6\,4\,2 \end{pmatrix}, A= (1, 5, 2, 0)$$

b)

$$A= (1, 5, 2, 0) \xrightarrow{P_0} (1, 5, 3, 2) \xrightarrow{P_2} (2, 8, 8, 6) \xrightarrow{P_1} (3, 8, 8, 6) \xrightarrow{P_3} (3, 14, 11, 8) \xrightarrow{P_4} (3, 14, 12, 12)$$

c)

Sol)

If granted, the snap shot will be change to

$$R = \begin{pmatrix} 0\,0\,0\,0 \\ 0\,3\,3\,0 \\ 1\,0\,0\,2 \\ 0\,0\,2\,0 \\ 0\,6\,4\,2 \end{pmatrix}, C = \begin{pmatrix} 0\,0\,1\,2 \\ 1\,4\,2\,0 \\ 1\,3\,5\,4 \\ 0\,6\,3\,2 \\ 0\,0\,1\,4 \end{pmatrix}, A = (1\,1\,0\,0)$$

$$A= (1, 1, 0, 0) \xrightarrow{P_0} (1, 1, 1, 2) \xrightarrow{P_2} (2, 4, 6, 6) \xrightarrow{P_1} (3, 8, 8, 6) \xrightarrow{P_3} (3, 14, 11, 8) \xrightarrow{P_4} (3, 14, 12, 12)$$

2.

a.

- Maximum virtual address space = $2^{32} = 2^{22} \times 2^{10} = 2^{22}$ **KB**
- $\therefore$ Maximum # of pages per a process = virtual space / a page size = $2^{22} / 4 = 2^{20}$ pages .
- Maximum size of page table per a process = number of page $\times$ one entry size
  = $2^{20} \times 64$ bits = $(2^{20} \times 64)/8$ Byte= $2^{20} \times 8$ byte= **8 MB**

b.

- need calculate the number of page frame
  # of page frame = size of RAM / size of page
  = 8GB / 4KB = $8 \times 2^{30} / 4 \times 2^{10} = 2^{21}$ page frames
  $\therefore$ **21 bits for page frame number.**

3.

a.

A multilevel page table reduces the number of actual pages of the page table that need to be in memory because of its hierarchic structure. In fact, in a program with lots of instruction and data locality, we only need the top level page table (one page), one instruction page, and one data page.

b.

- Since page size = 16KB $=2^{14}$, 14 bit for offset. That leaves 24 bits for the page fields.
- Since each entry is 4 bytes, one page can hold $2^{14}/4 = 2^{12}$ page table entries and therefore requires 12 bits to index one page. So allocating 12 bits for each of the page fields will address all 2^{38} bytes.
- 12 + 12 + 14

4.

a.

- Mutual exclusion
- Hold-and Wait
- No preemption
- Circular wait

b.

- Ignore
- Detection and recovery
- Avoidance with dynamic allocation
- By attacking one of necessary deadlock condition

c.

Sol) a segment is a logical entity.

- If the segments are large, to keep them in the physical memory might be wasting memory space.
- If a segment's virtual space is larger than physical space, it is not even possible to keep them in the physical memory.

5.

a.

Size of bit-map = $2 \times 2^{10} \times 2^{16}$ Byte = $2 \times 2^{10} \times 2^{16} \times 8$ Bit = 2^{30} bit.
There are 2^{30}blocks
Total disk size = $2^{30} \times 2 \times 2^{10} = 2^{41}$ Byte =2 TB

b.

sol) # of block information per block = (block size / bit used for a block #) – 1
= $8 \times 2 \times 2^{10}$ / 32 bit = $2^9 - 1 = 512 - 1 = 511$
total # block number = 511×2^{20}
total disk size = $511 \times 2^{20} \times 2 \times 2^{10} = 511 \times 2^{31}$ Byte = about $2^9 \times 2^{31}$ Byte = 1 TB.

6.

a. files are cached in the RAM when it is opened.

b.

- In LSF, each i-node is not at a fixed location; they are written to the log.
- LFS uses a data structure called an **i-node map** to maintain the current location of each i-node.
- Opening a file consists of using the map to locate the i-node for the file.

7.

a) Page 3
b) Page 0
c) Page 0
d) Page 1

8.

Sol) since 1 block is 2KB, and 4 Byte per block address, it can save $2 \times 2^{10} / 4 = 2^9 = 512$ block information

Total = 512 + 10 = 522 block information.

Since a block size is 2KB, largest file will be 2KB × 522 =1044 KB

9.

a.

$$Total\ Overhead(P) = Average\ page\ table\ size + the\ wasted\ memory\ in\ th\ last\ page\ of\ process$$
$$= \frac{S}{P} \times E + \frac{P}{2}$$

b.

$$Overhead'(P) = -\frac{SE}{P^2} + \frac{1}{2} = 0$$

$P = \sqrt{2SE}$: *optimal* page size

10.

a.

Sol) 1 block = $2 \times 2^{10} \times 8 = 16384$ bit, 16384 /32 = 512 -1 = 511 block numbers /block
128 GB = $(128 \times 2^{30})/(2 \times 2^{10}) = 2^{26}$ blocks (number of blocks in 128 GB)
Needs $(2^{26})/511$ = 131328.5= 131229 blocks

b.

Sol) 128 GB = $(128 \times 2^{30})/(2 \times 2^{10})$ = 2^{26} blocks (number of blocks in 128 GB)
System need one bit per block, the bit map size is 2^{26} bits.
2^{26} bits = $(2^{26})/8$ = 2^{23}Byte
Since each block size is 2KB, need $(2^{23})/(2 \times 2^{10})$= 4096 blocks to save free block information.

c.

- Since this system use 32bit disk block number, this system support 2^{32} blocks
- Maximum disk size = $2^{32} \times 2 \times 2^{10}$ Byte =8×2^{40} = 8 TB

11.

- Maintains an internal array M that keep track of the state of memory.
- M has as many as virtual memory pages *n*.
- Top *m* entries contain all the pages currently in the memory (page frames).
- Bottom *n* - *m* entries contains all the pages that have been referenced once but have been page out and are not currently in memory

12.

a. **Phase 1 :** begins at the starting directory and examines all the entries in it. For each modified file, its i-node is marked in the bitmap. Each directory is also marked and recursively inspected.
b. **Phase 2:** unmarking any directories that have no modified files or directories in them or under them.
c. **Phase 3:** all marked directory is dumped
d. **Phase 4:** all marked files is dumped

13. (5 pt.)

a)

$$A=(2,1,0,0) \xrightarrow{P_3} (2,2,2,0) \text{ Deadlock}$$

b)

$$A=(0,0,0) \xrightarrow{P_1} (0,1,0) \xrightarrow{P_3} (3,1,3) \xrightarrow{P_2} (5,1,3) \xrightarrow{P_4} (7,2,4) \xrightarrow{P_5} (7,2,6)$$

14.

Sol) In LINUX system, size of a block and number of blocks information are saved in super block in the partition. Since used block information for a file is saved in it's i-node, we can get used blocks information by scanning all i-nodes.

```
Create new bitmap (size of bit = number of block from super block)
        Reset (set as 0)
For each i-node do
        For each entry of i-node do
                Set bitmap to 1 (based on used block information)
```

15.

Solution 1)

Attacking hold and wait,
starvation

Solution 2)

Attacking circular wait,
If a process need two resource at a same time, this solution have problem