"OS Assessment" K180268

Date: - -

Q1: ⓐ The problem in code is that the Threads will deadlock/ interlock among them, as they will acquie lock A, lock B, respectively, but execute at the same time. So each thread needs both of lock A and lock B to be free.

ⓑ The code can be solved, by removing either lock A completely or lock B completely. Now the threads will execute in a way that the one which comes first will be dealt with first, and after it gets executed the other will resume. This is also needed to be done as They acquie the data same as a and b
Make sure that the locks are always taken in same order by threads, so deadlock doesn't occur. Lock ordeing will also be effective.

Q2 :
```
while (true) {
    wait( fork[(i+1)%2])
    // eat now
    signal (fork[(i+1)%2])    }
```

Q3 : 256 pages $= 2^8$

page size $=$ 0268 $\approx$ 256 KB $= 2^8 \times 2^{10} = 2^{18}$

frames $= 128 = 2^7$

ⓐ logical address space bits =

$8 + 18 = 26$ bits

ⓑ Physical address space bits =

$7+18 = 25$ bits

Date: - -

Q4: (1) Segmentation has reduced the problem internal fragmentation. Segments are created, where each has a different virtual address space that directly corresponds to process. Example can be taken, that a program may be having many segments in the logical address space, such as different libraries, mapping onto physical memory is done by limit and base registers. The segments, now, may not be contiguous necessarily, but the exact amount of space that it needs is important, therefore no internal fragmentation problem.

(2) Multithreaded process provides more speed as compared to processes without thread, because it increases performance greatly because CPU can divide simultaneous work amongst multiple threads. The sum of time spent on all processors is higher b/c extra work of coordinating multiple threads. It allows shared resource usage like memory. While it has to do additional synchronization work b/w threads. but finishes faster b/c simultaneous work being done, i.e work on threads of diff type at same time.