# OS-1 Final Exam
# S.V.Harshith – EE19BTECH11018

**Q1)**

If we have DMA then the CPU can do other tasks while performing an I/O operation.

The main use of multi programming is to give the CPU another task while the CPU is waiting for the I/O to complete. If there is no DMA in the multi programming system then the CPU will be fully occupied by I/O tasks and there is no use of multiprogramming.

**Q2)**

Given –

1 nsec to access word from cache

10 nsec to access word from RAM

10 msec to access word from Disk

Cache rate = 95%

Min memory hit rate (After cache miss) = 99%

Implies, Average Time for accessing from RAM –

= (RAM hit rate)*(Ram access time) + (RAM miss rate)*(Disk access time)

= (99/100) * (10 nsec) + (1/100)*(10 msec)

= 0.100099 msec

Therefore, **Average Time for accessing a word** –

= (cache hit rate)*(cache access time) + (cache miss rate)*(average Ram access time)

= (95/100)*(1 nsec) + (5/100)*(0.100099 msec)

**= 5001.445 nsec**

**Q3)**

From Amdahl's Law ,

Maximum Speed up = 1/(S+(1-S)/N)

Given S = 30%

(a) N = 4
    Maximum Speed up = 1/(0.3+(1-0.3)/4)
    = 1/(0.3+0.175)
    = 1/0.475
    = **2.105 times**

(b) N = 8
    Maximum Speed up = 1/(0.3+(1-0.3)/8)
    = 1/(0.3+0.0875)
    = 1/0.3875
    = **2.581 times**

**Q4)**

Given –

Time to process a request = 12 msec

Time for disk operation = 75 msec

Disk operations occur one-third of the time , implies for every three process we get one disk operation.

(a) Single threaded –
    Time for cache hit(2/3 probability) = 12 msec
    Time for cache miss + disk operation(1/3 probability) = 12 msec + 75 msec
    = 87 msec

    Therefore, Average time = (2/3)*12 msec + (1/3)*87 msec = 37msec
    Implies, requests/sec for single thread = 1000/37 = **27 request/sec**
    (Approximated to integer)

(b) Multi-threaded –
    Time for cache hit(2/3 probability) = 12 msec
    Now for cache miss(1/3 probability) , since it is multi-threaded if there is a miss, another thread will take a cache hit and the time for disk operation is not there.
    So, both the requests will take 12 msec
    Implies, requests/sec for single thread = 1000/12 = **83 request/sec**
    (Approximated to integer)

**Q5)** As the textbook says,

(a) The difficulty with cancellation occurs in situations where resources have been allocated to a cancelled thread or where a thread is cancelled while in the midst of updating data it is sharing with other threads. This becomes especially troublesome with asynchronous cancellation. Often, the operating system will reclaim system resources from a cancelled thread but will not reclaim all resources. Therefore, cancelling a thread asynchronously may not free a necessary system-wide resource.

(b) One technique for establishing a cancellation point is to invoke the pthread_testcancel() function. If a cancellation request is found to be pending, the call to pthread_testcancel() will not return, and the thread will terminate; otherwise, the call to the function will return, and the thread will continue to run.

**Q6)**

(a) Suppose a child will ask for a necessary system-wide resource and the resource is freed in the parent , so if the parent is destroyed due to some error and the child is not destroyed then the system-wide resource will not be freed and it can't be accessed by other processes. So, the child should be destroyed after the parent is terminated.

(b) A situation in which destroying a parent should specifically not result in the destruction of its children(orphan process) is when we want to create a process to complete a very long running job without user attention as we know that orphan processes are taken up by init process as a child and it is then terminated there.

**Q7)**

Usually, a multi-threaded server has multiple threads for different cases like receiving the request , responding to request , reading the hard disk etc. which are used for providing faster responses by using the shared memory.

For a simple web server like a server for getting stock prices of different companies which is a floating-point number and suppose it takes a maximum of 32 bits for each number then for even a large number of companies it will just around 32MB of memory which can be easily stored in the server and  a fast search access can be given to it. Multiple threads for this server will only slow down the process and just add unnecessary complexity. So, a single threaded server is better in this case.