1. Circle the relevant role of an operating system as R, I, or G (i.e., acting as a Referee, Illusionist, or Glue):

(a) R / I / G Physical details of a disk, such as sector size, are hidden.

(b) R / I / G Cut and paste commands work across different applications.

(c) R / I / G Prevent users from accessing each other’s files without permission.

2. Match these three terms to the appropriate blanks below: efficiency, response time, throughput

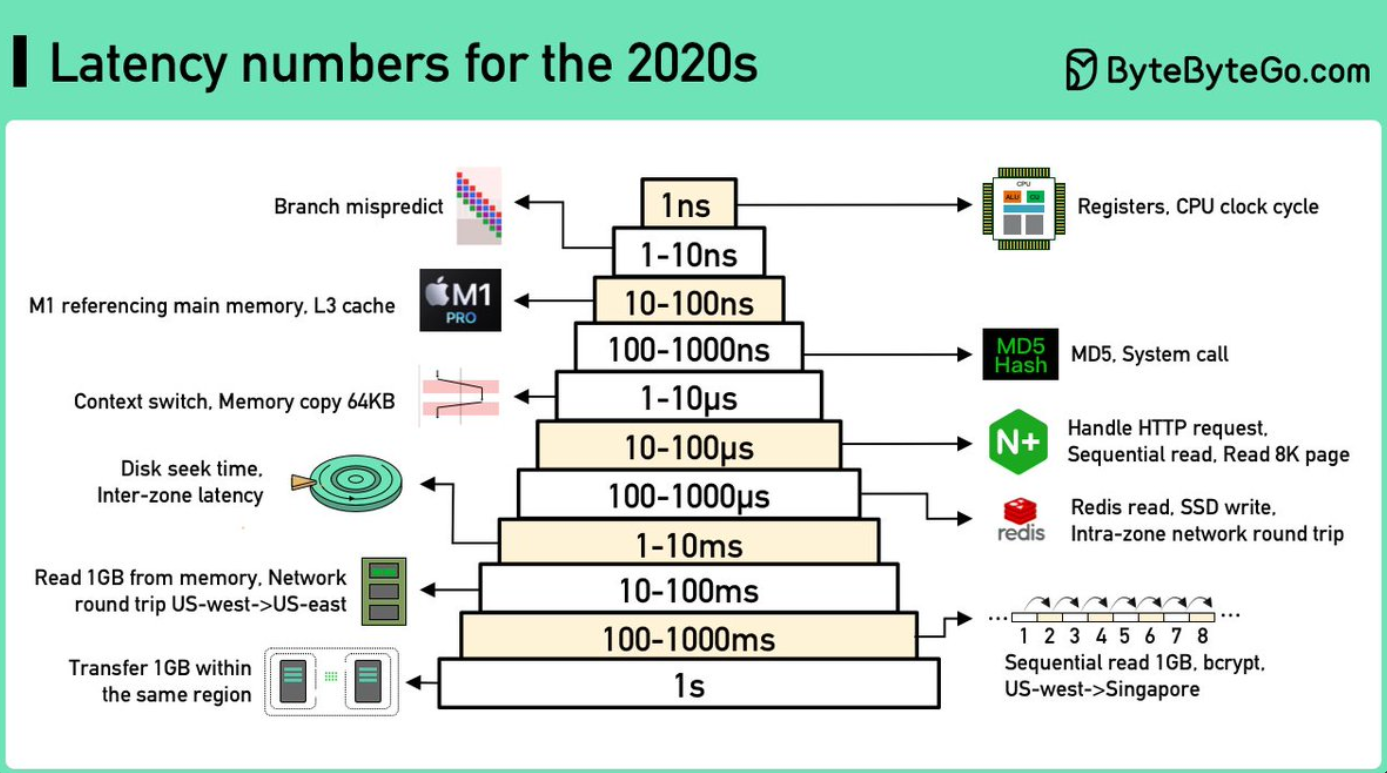
1. The design goal for batch operating systems was high \_\_\_throughput \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.
2. The design goal for time-sharing operating systems was low \_\_\_response time\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.
3. The design goal for virtual machines is high \_\_\_ efficiency\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

3. Operating systems can support communication between applications in multiple ways:

* Through the file system on secondary storage, using open(), read(), write(), close(), etc. system calls.
* With messages passed between the applications, using connect(), send(), receive(), disconnect() system calls.
* Through regions of memory shared between the applications, using load and store instructions.

Which method is likely the most efficient? Use the chart below to help explain why

Shared memory between applications will be the most efficient at the 10-100ns level (processes typically use message passing unless they ask for a shared memory regiment).



Notes: 1 ns = 10-9 seconds

See, Alex Xu, “Latency Numbers Programmer Should Know,” https://www.youtube.com/watch?v=FqR5vESuKe0

1 µs = 10-6 seconds = 1,000 ns

1 ms = 10-3 seconds = 1,000 µs = 1,000,000 ns

4. Consider this statement (from Wikipedia):

In early versions of DOS, printing was accomplished using the copy command: the file to be printed was “copied” to the file representing the print device. Control returned to the user when the print job completed.

Can an OS support printing in the background and thereby allow control to return to a user almost immediately after he or she issues a print command (or to an application after a system call to print is made) rather than waiting until the printing is complete? If so, suggest a way in which this can be done.

Yes, it can. Multiple processes are used – one for the text editor and one for the printing. Historically this was called multiprogramming/multitasking. For printing it was specifically called spooling.