2.

a)

Cache	Memory
Store data for frequently used instructions or future requests	Store data currently being used
Faster	Slower
More expensive	Cheaper
Less capacity	More capacity
CPU reads cache before memory	CPU reads memory before cache

b) Latency = response waiting time while moving the data [ms or ns]

Bandwidth = how much data can be transferred per time unit [Mbits /s or MB/s]

c) Cache can be 10-100 times faster than main memory.

3.

a) RAM is usually bigger than VRAM. This is due to their dedicated purposes: RAM is where all running processes and software are stored; whereas VRAM is only for graphic, video, and display needs. Computers most of the time require RAM to execute the majority of tasks, therefore more RAM is needed compared to VRAM.

b) Intel and AMD have a monopoly over x86 and x86_64 architecture for desktop CPU. With no competitors in the market, these two giants could invest a lot of money into research and development, rapidly advancing caching technology.

4.

a) In dynamically converting memory addresses, the CPU (or MMU, to be exact) converts the program addresses to physical addresses during the execution, allowing instructions to be loaded as the execution proceeds and old instructions to be discarded and replaced with new ones. Amount of memory will not be a “hard” limit (although constant loading and discarding is slow).

In dynamic memory allocation, a sufficient amount of memory is allocated to the process(es). We can change the size of memory areas that we allocate to processes and are able to eliminate (or at least minimize) internal fragmentation.

b)

Paging	Segmentation
In paging, program is divided into fixed or mounted size pages.	In segmentation, program is divided into variable size sections.
For paging operating system is accountable.	For segmentation compiler is accountable.
Page size is determined by hardware.	Here, the section size is given by the user.
It is faster in the comparison of segmentation.	Segmentation is slow.
Paging could result in internal fragmentation.	Segmentation could result in external fragmentation.

In paging, logical address is split into page number and page offset.	Here, logical address is split into section number and section offset.
Paging comprises a page table which encloses the base address of every page.	While segmentation also comprises the segment table which encloses segment number and segment offset.
Page table is employed to keep up the page data.	Section Table maintains the section data.
In paging, operating system must maintain a free frame list.	In segmentation, operating system maintain a list of holes in main memory.
Paging is invisible to the user.	Segmentation is visible to the user.
In paging, processor needs page number, offset to calculate absolute address.	In segmentation, processor uses segment number, offset to calculate full address.
It is hard to allow sharing of procedures between processes.	Facilitates sharing of procedures between the processes.
In paging, a programmer cannot efficiently handle data structure.	It can efficiently handle data structures.
In this protection is hard to apply.	Easy to apply protection in segmentation.

c)

Internal fragmentation = amount of memory allocated to a process is greater than required.

External fragmentation = free memory consists of small slots which are unusable due to their size being too small to fit a process.

External fragmentation can be “cured” by rearranging the memory at time intervals (compaction); internal fragmentation, on the other hand, is harder to tackle.

5.

a)

Programmed I/O: It is due to the result of the I/O instructions that are written in the computer program. Each data item transfer is initiated by an instruction in the program. Usually, the transfer is from a CPU register and memory. In this case it requires constant monitoring by the CPU of the peripheral devices.

Interrupt- initiated I/O: Since in the above case we saw the CPU is kept busy unnecessarily. This situation can very well be avoided by using an interrupt driven method for data transfer. By using interrupt facility and special commands to inform the interface to issue an interrupt request signal whenever data is available from any device. In the meantime, the CPU can proceed for any other program execution. The interface meanwhile keeps monitoring the device. Whenever it is determined that the device is ready for data transfer it initiates an interrupt request signal to the computer. Upon detection of an external interrupt signal the CPU stops momentarily the task that it was already performing, branches to the service program to process the I/O transfer, and then return to the task it was originally performing.

Direct Memory Access: The data transfer between a fast storage media such as magnetic disk and memory unit is limited by the speed of the CPU. Thus, we can allow the peripherals directly communicate with each other using the memory buses, removing the intervention of the CPU. This type of data transfer technique is known as DMA or direct memory access. During DMA the CPU is idle and it has no control over the memory buses. The DMA controller takes over the buses to manage the transfer directly between the I/O devices and the memory unit.

b)

Short I/O wait = the interrupt is handled right after it occurs.

Long I/O wait = the interrupt is handled when the next I/O request arrives.

When long I/O wait is used, there can only be 1 I/O request in line at a time.