1. **What is MMU? Explain its significance**

   A memory management unit (MMU) is a computer hardware component that handles all memory and caching operations associated with the processor. In other words, the MMU is responsible for all aspects of memory management. It is usually integrated into the processor, although in some systems it occupies a separate IC (integrated circuit) chip.
   The work of the MMU can be divided into three major categories:
   - Hardware memory management, which oversees and regulates the processor's use of RAM (random access memory) and cache memory.
   - OS (operating system) memory management, which ensures the availability of adequate memory resources for the objects and data structures of each running program at all times.
   - Application memory management, which allocates each individual program's required memory, and then recycles freed-up memory space when the operation concludes.

2. **What is logical address and physical address? Explain the differences**

   An address generated by the CPU is a logical address whereas address actually available on memory unit is a physical address. Logical address is also known a Virtual address. Virtual and physical addresses are the same in compile-time and load-time address-binding schemes. Virtual and physical addresses differ in execution-time address-binding scheme.

   The set of all logical addresses generated by a program is referred to as a logical address space. The set of all physical addresses corresponding to these logical addresses is referred to as a physical address space. The run-time mapping from virtual to physical address is done by the memory management unit (MMU) which is a hardware device.

3. **What are difference between Volatile / Non-volatile / Hybrid memory?**

   Non-volatile memory is typically used for the task long-term persistent storage (ROM) which will generally use for one time dump. The most widely used form of primary storage today is a volatile form of random access memory (RAM), meaning that when the system power down, anything contained in RAM is lost. Some memory have both properties like, we can store and remove data any number of time. Eg Flash, EPROM etc

4. **Explain the concept of Virtual memory**

   Ref Qns 1, 2 & 5

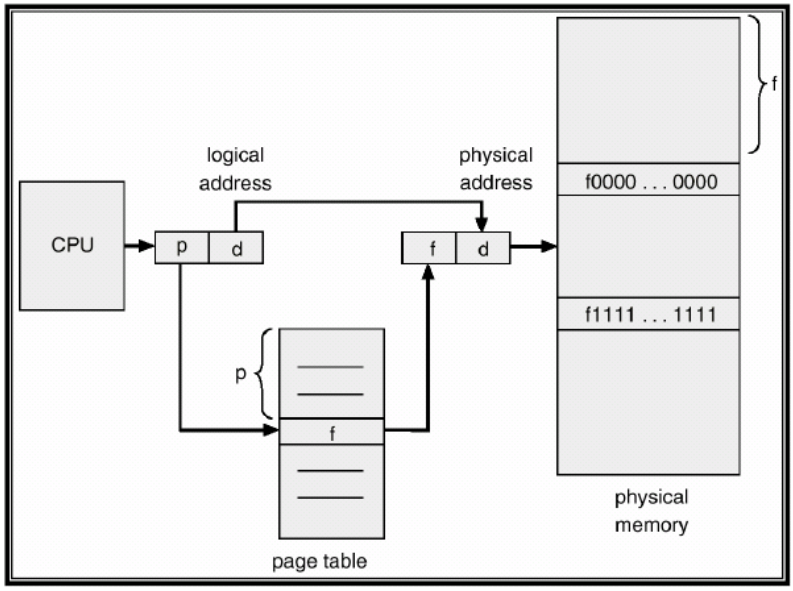
5. **What is paging and page table? What are the advantages?**

   If programs access physical memory, we will face three problems
   - Don't have enough physical memory.
   - Holes in address space (fragmentation).
   - No security (All program can access same memory)

   These problems can be solved using virtual memory.
   - Each program will have their own virtual memory.
   - They separately maps virtual memory space to physical memory.
   - We can even move to disk if we run out of memory (Swapping).

   What is paging?

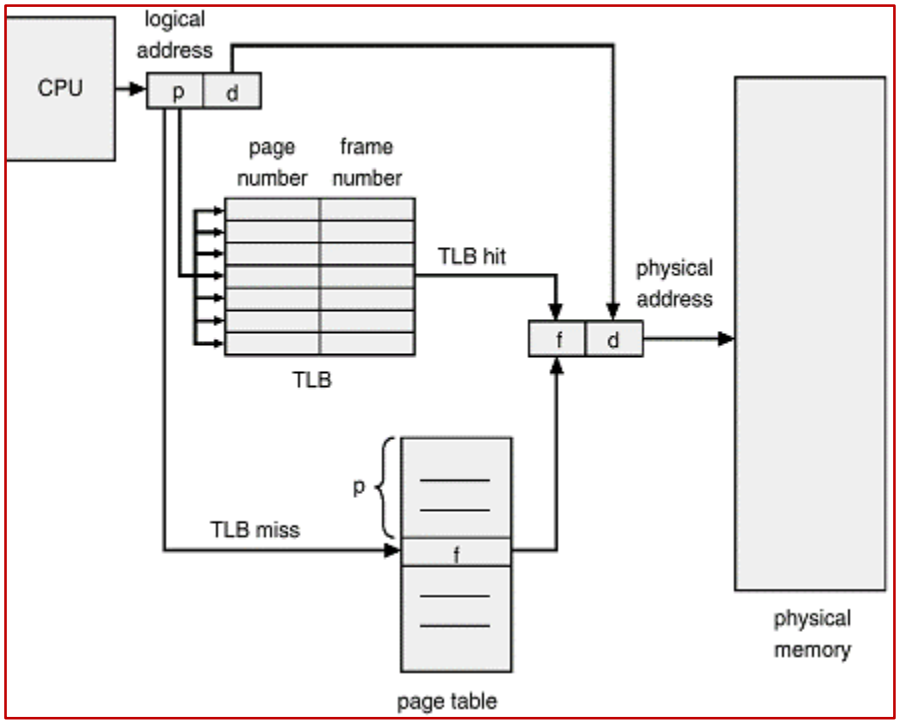
   - Virtual memory divided into small chunks called pages.
   - Similarly physical memory divided into frames.
   - Virtual memory and physical memory mapped using page table.



6. What is TLB? Explain

For faster access page table entries will be stored in CPU cache memory is called TLB.

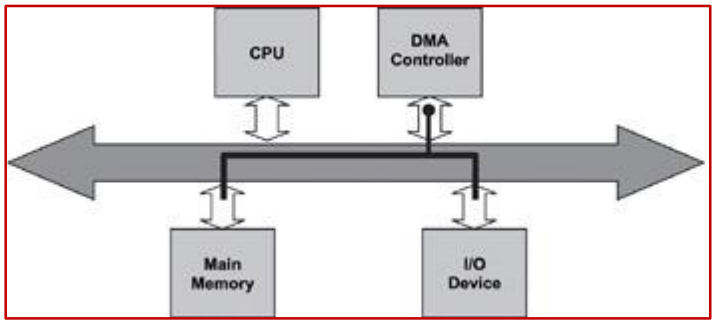
- But limited entries only possible.
- If page entry available in TLB (Hit), control goes to physical address directly (Within one cycle).
- If page entry not available in TLB (Miss), it use page table from main memory and maps to physical address (Takes more cycles compared to TLB).



7. What is DMA?

Direct memory access (DMA) chips or controllers solve this problem by allowing the device to access the memory directly without involving the processor, as shown in Figure. The processor is used to set up the DMA controller before a data transfer operation begins, but the processor is bypassed during data transfer, regardless of whether it is a read or write operation.

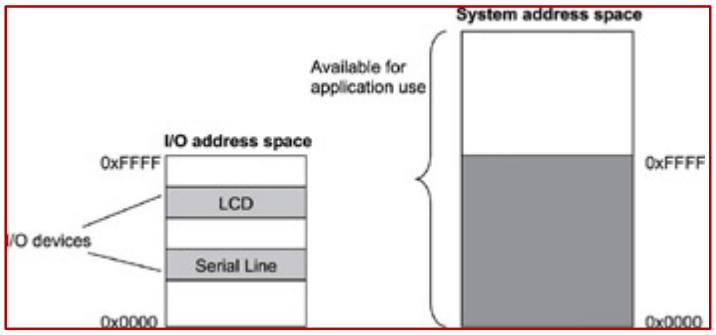
The transfer speed depends on the transfer speed of the I/O device, the speed of the memory device, and the speed of the DMA controller. In essence, the DMA controller provides an alternative data path between the I/O device and the main memory. The processor sets up the transfer operation by specifying the source address, the destination memory address, and the length of the transfer to the DMA controller.



8. Explain differences between memory mapped I/O and port mapped I/O

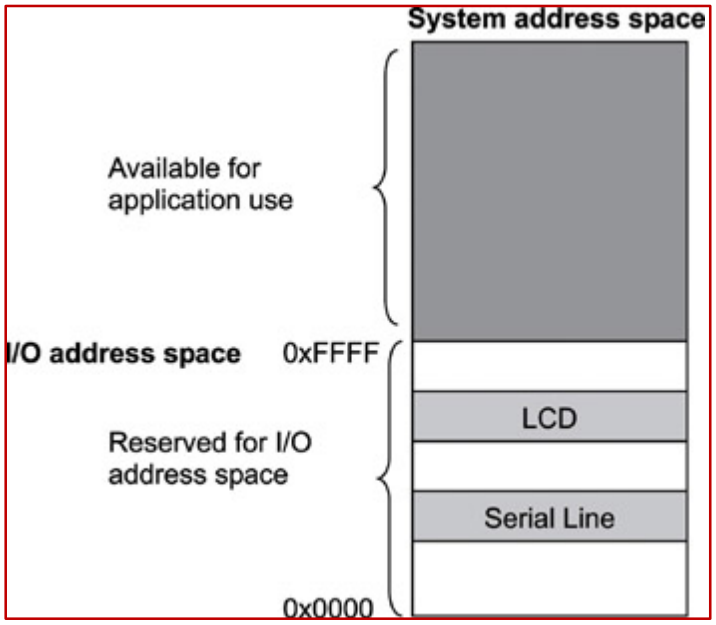
- Port-mapped I/O:

The devices are programmed to occupy a range in the I/O address space. Each device is on a different I/O port. The I/O ports are accessed through special processor instructions, and actual physical access is accomplished through special hardware circuitry. This I/O method is also called isolated I/O because the memory space is isolated from the I/O space, thus the entire memory address space is available for application use.



- Memory mapped I/O

The memory-mapped I/O, the device address is part of the system memory address space. Any machine instruction that is encoded to transfer data between a memory location and the processor or between two memory locations can potentially be used to access the I/O device. The I/O device is treated as if it were another memory location. Because the I/O address space occupies a range in the system memory address space, this region of the memory address space is not available for an application to use.



9. What is cache memory?

A CPU cache is a cache used by the central processing unit (CPU) of a computer to reduce the average time to access data from the main memory. The cache is a smaller, faster memory which stores copies of the data from frequently used main memory locations. Most CPUs have different independent caches, including instruction and data caches, where the data cache is usually organized as a hierarchy of more cache levels (L1, L2, etc.).

When the processor needs to read from or write to a location in main memory, it first checks whether a copy of that data is in the cache. If so, the processor immediately reads from or writes to the cache, which is much faster than reading from or writing to main memory. Most modern desktop and server CPUs have at least three independent caches: an instruction cache to speed up executable instruction fetch, a data cache to speed up data fetch and store, and a translation look-aside buffer (TLB) used to speed up virtual-to-physical address translation for both executable instructions and data.

The data cache is usually organized as a hierarchy of more cache levels (L1, L2, etc.; see also multi-level caches below). However, a TLB cache is part of the memory management unit (MMU) and not directly related to the CPU caches.

10. What is DMA cache coherency? How to solve?

If a CPU has a cache and external memory, then the data the DMA controller has access to (store in RAM) may not be updated with the correct data stored in the cache.
Solutions

- Cache-coherent systems:

External writes are signaled to the cache controller which performs a cache invalidation for incoming DMA transfers or cache flush for outgoing DMA transfers (done by hardware)

- Non-coherent systems:

OS ensures that the cache lines are flushed before an outgoing DMA transfer is started and invalidated before a memory range affected by an incoming DMA transfer is accessed. The OS makes sure that the memory range is not accessed by any running threads in the meantime.

11. What is position dependent code and position independent code in Linux?

The code within a dynamic executable is typically position-dependent, and is tied to a fixed address in memory. Shared objects, on the other hand, can be loaded at different addresses in different processes. Position-independent code is not tied to a specific address. This independence allows the code to execute efficiently at a different address in each process that uses the code.

Position independent code is recommended for the creation of shared objects.

12. What is swapping? Explain with details

Swapping is a mechanism in which a process can be swapped temporarily out of main memory to a backing store, and then brought back into memory for continued execution. Backing store is a usually a hard disk drive or any other secondary storage which fast in access and large enough to accommodate copies of all memory images for all users. It must be capable of providing direct access to these memory images.

Major time consuming part of swapping is transfer time. Total transfer time is directly proportional to the amount of memory swapped. Let us assume that the user process is of size 100KB and the backing store is a standard hard disk with transfer rate of 1 MB per second. The actual transfer of the 100K process to or from memory will take