Assignment No. 4

Q.1] Explain the binding of instructions & data to memory.

→ (1) Usually, a program resides on disk as a binary executable file.
(2) To be executed, the program must be brought into memory and placed within process.
(3) Depending on the memory managment in use, the process may be moved bet^n. disk & memory during its execution.
(4) The process on disk that are waiting to be brought into memory for execution from the input queue.
(5) In most case, a user program goes through several steps-some of which may be optional- before being executed.
(6) Address may be represented in different ways during these steps.
(7) Addresses in the source program are generally symbolic.
(8) A compiler typically binds these symbolic addresses to relocated addresses.
(9) The linker editor / loader in turn binds the relocatable addresses to absolute addresses.
(10) each binding is mapping from one address space to another.

Q.2] Write note -

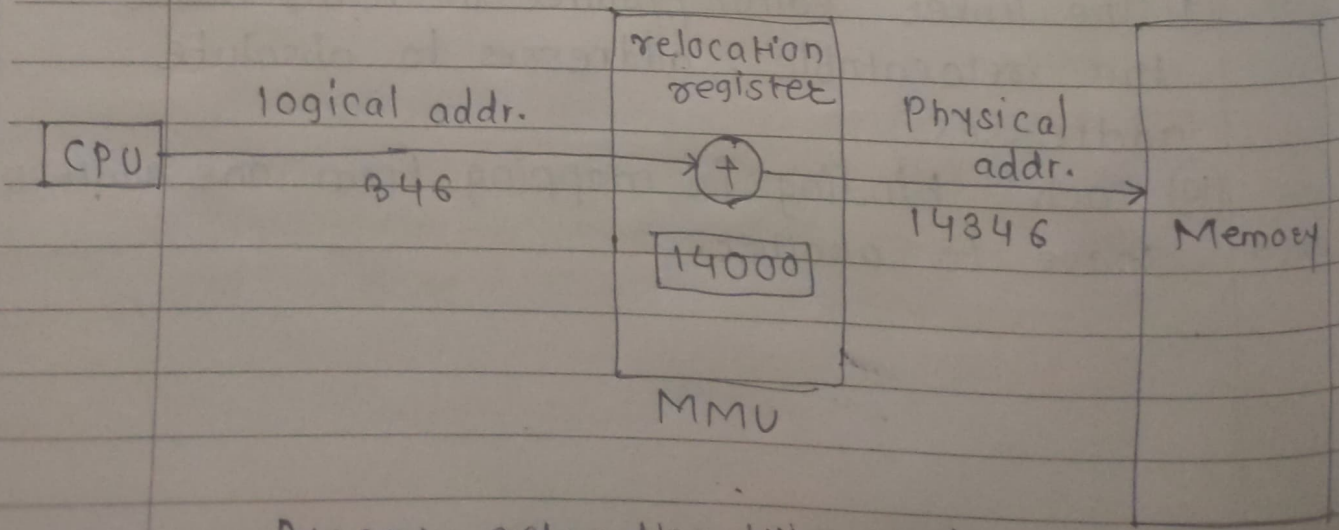
a] logical versus physical address space

→ ① A address generated by CPU is commonly reffered to as logical address, where as an address seen by the memory unit - that is, the one loaded into the memory-address register of the memory - is commonly referred to as physical address.

② The run-time mapping from virtual to physical addresses is done by a hardware device called the memory-managment unit (MMU)

③ The value in the relocation register is added to every address generated by a user process at the time the address is sent to memory.

④ for ex., if base is at 14000, then an attempt by the user to address location 0 is dynamically relocated to location 14000; an access to location 346 is maped to location 14346.



Dynamic relocation using relocation register.

b]

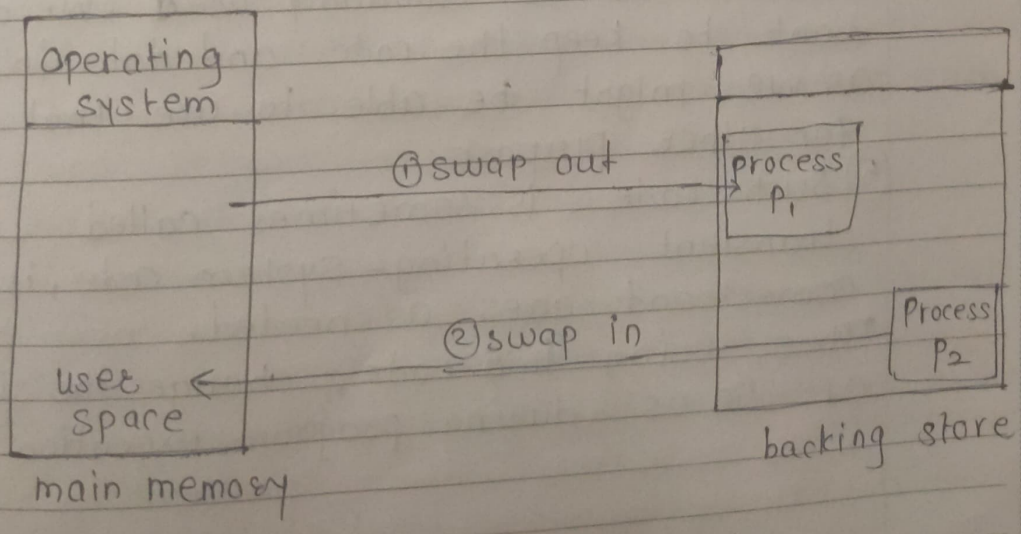
b) swapping :-

→ ① swapping is mechanism in which a process can be swapped temporarily out of main memory to secondary storage & make the memory available to other processes.

② At some later time, the system swaps back the process from the secondary storage to main memory.

③ Though performance is usually affected by swapping process but it helps in running multiple & big processes in paralle and that's the reason swapping is also known as technique for memory compaction.

④ The total time taken by swapping process includes the time it taken to move the entire process to a secondary disk & then to copy the process back to memory, as well as the time the process takes to regain main memory.



Schematic view of swapping

c] Memory protection :-

→ We can prevent a process from accessing memory it does not own by combining
① when the CPU scheduler selects a process for execution, the dispatcher loads the relocation and limit registers with the correct values as part of the context switch.
② Because every address generated by a CPU is checked against these registers, we can protect both the OS & the other user's programs and data from being modified by this running process.
③ The relocation-register scheme provides an effective way to allow the operating system's size to change dynamically.
④ This flexibility is desirable in many situation.
for ex. the OS contains code and buffer space for device drivers. If a divice driver is not commonly used, we do not want to keep the code and data in memory as we might be able to use that space for other purpose.
⑤ Such code is sometimes called transient operating-system code, it comes and goes as needed.
Thus, using this code, changes the size of OS during program execution.

d] Fragmentation :-

→ ① Both the first-fit and best-fit strategies for memory allocation suffer from external fragmentation.

② As processes are loaded and removed from memory, the free memory space is broken into little pieces.

③ External fragmentation :-

- Exist when there is enough total memory space to ~~sto~~ satisfy a request but the available space are not contiguous; storage is fragmented into a large number of small holes.

- This fragmentation problem can be severe. In worst case, we could have a block of free memory bet^n. every two processes.

- If all these small pieces of memory were in one big free block instead, we might be able to run several more processes.

④ Whether we are using the first-fit or best-fit strategy can affect the amount of fragmentation.

⑤ 50-percent rule :- statistical analysis of first fit, for instance, reveals that, even with some optimization, given N allocated blocks, another 0.5N blocks will be lost to fragmention.

⑥ i.e., one third of memory may be unusable. This property known as 50-percent rule.

⑦ Memory fragmentation can be internal as well as external.

① External :- total memory space exist is enough to satisfy a request, but it is not contiguous.

② Internal :- memory block assigned to process is bigger. Some portion of block become unused, as it cannot be used by other process.

e] Translation look-aside Buffer in Paging

→ ① In OS, for each process page table will be created, which will contain Page Table Entry (PTE).

② This Page table entry will tell where in main memory the actual page is residing.

③ Page table entries are place in registers, for each request generated from CPU, it will be matched to appropriate page number of page table, which will now tell where in the main memory that corresponding page resides.

④ but the problem is register size is small and process size may be big hence the required page table will also be big, so registers may not hold all PTE's of page table.

⑤ To overcome size issue, the entire page table was kept in main memory. But the problem here is two main memory references are required:

① To find the frame number

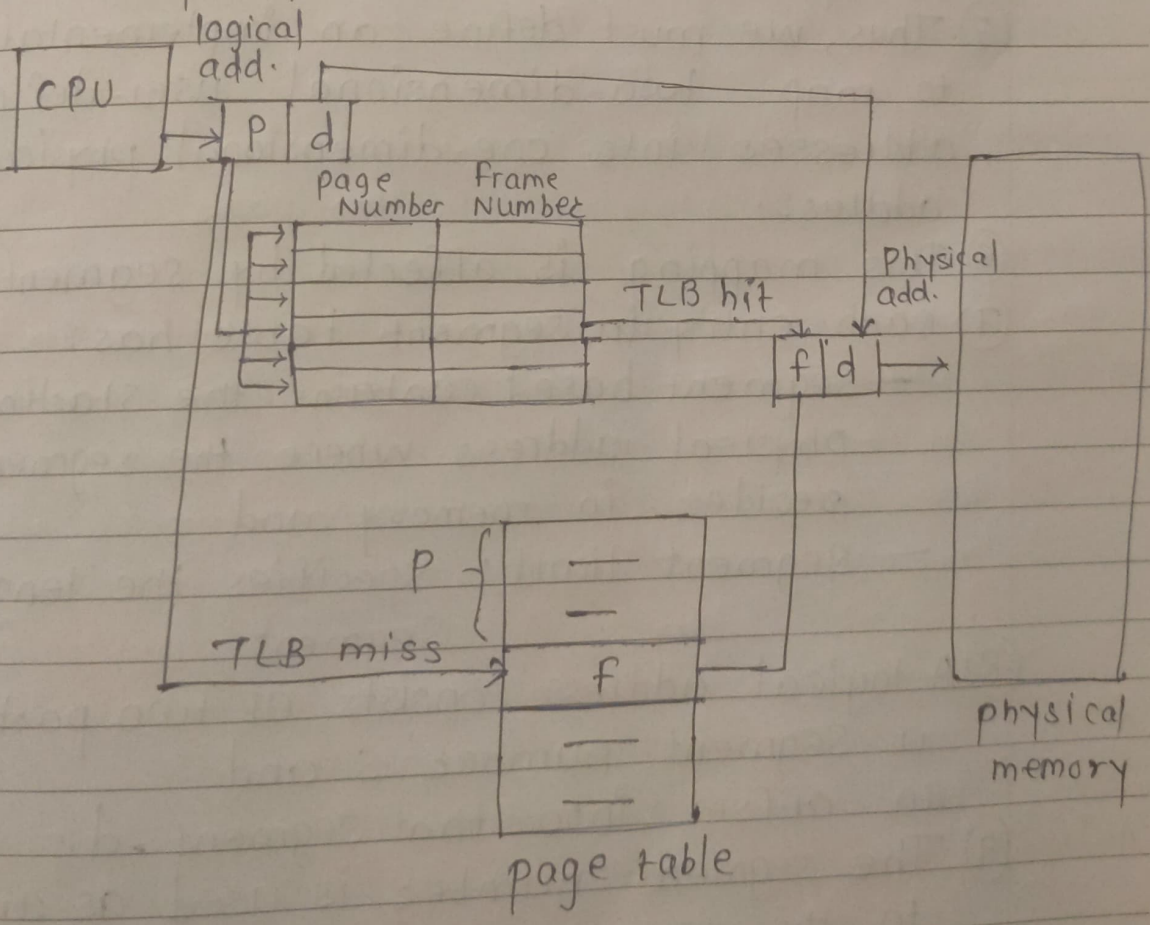
② To go to the address specified by frame number.

⑥ To overcome this problem a high-speed catche is set up for page table entries called Translation Look-aside Buffer (TLB).

⑦ Translation look-aside Buffer is nothing but a special cache used to keep track of recently used transaction. TLB contains page table entries that have been most recently used.

⑧ Steps in TLB hit:

- CPU generates virtual address
- It is checked in TLB.
- Corresponding frame number is retrieved, which now tells where in main memory page lies.



paging hardware with TLB.

Q.4] Explain segmentation with segmentation hardware.

→ (1) Segmentation is memory-managment scheme that supports this programmer view of memory.
(2) A logical address space is a collection of segments.
(3) each segment has a name & a length.
(4) Although, the programmer can now refer to object in the program by a two-dimensional address, the actual physical memory is still one-dimensional sequence of bytes.
(5) Thus, we must define an implementation to map two-dimensional user-defined addresses into one-dimensional physical address.
(6) This mapping is affected by segment table.
(7) each entry in segment table has:
— segment base: contains the starting physical address where the segment resides in memory, and
— segment limit: specifies the length of segment
(8) A logical address consists of two parts:
a segment number s and
an offset into that segment, d.
(9) The segment number is used as an index to the segment table.
(10) The offset d of the logical address must be bet^n 0 and the segment limit.

(11) If it is not, we trap to the operating system when an offset is legal, it is added to segment base to produce the address in physical memory of desired byte.

Q.5] Explain paging with suitable example.

→ (1) Segmentation permits the physical address space of a process to be noncontiguous.

(2) +

(1) Paging is another memory-managment scheme. ~~that offers this~~ Paging avoids external fragmentation and the need for compaction.

(2) Paging is implemented through cooperation bet^n. the operating system and computer hardware.

(3) The basic method for implementing paging involves:

- Breaking physical memory into fixed-size blocks called frames and
- Breaking logical memory into blocks of same size called pages.

(4) Every address generated by CPU is divided into two parts:

(1) Page number (P)

(2) Page offset (d)

(1) Page number is used as an index into a page table. The page table contains the base address of each page in physical memory.

(2) This base address is combined with the page offset to define the physical memory

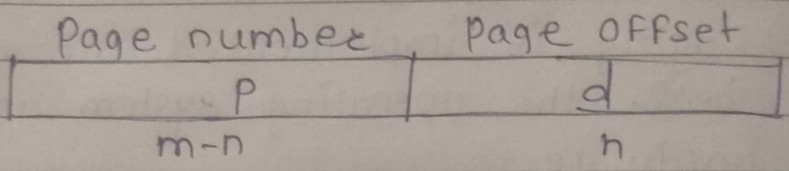
address that is sent to the memory unit.

⑤ The page size is defined by hardware.

⑥ The size of page is power of 2, varying bet^n 512 bytes and 1 GB per page depending on computer arch.

⑦ If the size of logical address space is 2^m and page size is 2^n bytes, then the high-order m-n bits of logical address designate the page number & the n low-order bits designated page offset.

⑧ logical address:-



p → index into page table
d → displacement within page

ex. consider memory in fig 1. Here in logical address, n=2 and m=4. Using a page size of 4 bytes and physical memory of 32 bytes we show how the programmer's view of memory can be mapped into physical memory. Logical address 0 is page 0, offset 0. Indexing into the page table, we find that page 0 is in frame 5.
Thus, logical address 0 maps to physical address 20 [= (5×4) + 0].
logical address 3 (page 0, offset 3) maps to physical address 23 [= (5×4) + 3]

logical address 4 is (page 1, offset 0) ~~thus~~ physical according to page table, page 1 is mapped to fram 6. Thus logical address 4 maps to physical address 24 [=(6×4)+0].

Q.6] What is page replacement & explain any of page replacement technique with ex.

→