1. **Address binding and types**

- **Address Binding** – Process of mapping program addresses to physical memory locations.
- **Compile-Time Binding** – Binding done during compilation when memory location is known in advance.
- **Load-Time Binding** – Binding occurs while loading the program into main memory.
- **Run-Time Binding** – Final address binding happens during program execution.
- **Binding Stages** – Binding can occur at compile time, load time, or run time based on flexibility needed

2. **Differentiate between static and dynamic linking.**

Static linking – system libraries and program code combined by the loader into the binary program image

Dynamic linking –linking postponed until execution time

Keyword	**Static Linking**	**Dynamic Linking**
Integration Time	Linking happens **before execution** during compile time.	Linking occurs **at run time** when the program is executed.
Executable Size	**larger executable** .	**smaller executable**
Library Use	Uses **copy of library** functions i	Uses **shared library**,
Performance	**Faster execution**	**Slightly slower start**

3. **Differentiate between logical and physical address.**

An address generated by the CPU - logical address.

AN address seen by a memory unit - physical address.

Compile time - same for both & run time differs for both

Keyword	**Logical Address**	**Physical Address**
Generated By	Generated by **CPU** during program execution.	Generated by **MMU** for accessing actual memory.
Visibility	Known as **virtual address**, visible to user programs.	**Invisible to users**, used internally by the hardware.

4. **Define swap-in and swap-out operations**.
   Swap-in refers to allocating main memory to a process and bringing it from the secondary storage. Swap-out refers to deallocating a process from the main memory and copying it to the secondary storage.

5. **Differentiate between internal and external fragmentation.**

Internal	Wasted space **within allocated memory blocks** due to fixed-sized partitions or paging.
External	Wasted space **between allocated blocks** due to variable-sized partitions or segmentation.

Internal	**External**
Inside-block	Between-blocks
Fixed-partitions	Variable-partitions
Paging-related	Segmentation-related
Unused-space	Free-space
Not-reusable	Compactable

6. **What is meant by compaction?**
   Compaction is a technique which relocates the memory contents to place all free memory together in one large block.
7. **Define page.---(size 2^n)**

Keyword	**Short Definition**
Page	Fixed-size block of **logical memory** (power of two size).
Mapping	Pages mapped to **frames** via page table.
Fragmentation	Paging removes **external fragmentation**.

8. **How does a page and a frame related?**

The logical pages are loaded in to the same fixed-sized blocks in the physical memory called frames.

9. **Why are page sizes always in powers of 2?**

   Since the main memory capacity is in powers of 2, the page and frame size are always in powers of 2.

10. **Differentiate between a page and a segment.**

Aspect	Page	Segment
Size	Fixed-size block (power of 2).	Variable-sized block.
Unit	Divides memory into equal-sized pages.	Divides memory based on logical units like functions or data structures.
User View	Does **not** match user's logical view.	Matches user's logical view of memory.
Example	Memory split into pages like 1KB, 2KB, etc.	A program split into segments like code, stack, heap, etc.

11. **What is PTBR?**

   PTBR stands for Page Table Base Register.
   PTBR points to the page table in main memory.
   Translation-used to transalte logical to physical address & speed wise -slow

12. **What is inverted page table?**

- ✓ An inverted page table has one entry for each frame of memory.
- ✓ Each entry stores the page’s virtual address and its owning process info. Advantage: Only **one page table** exists in memory for all processes instead of one per process.

13. **What is STBR and STLR?**

- ● STBR stands for Segment Table base Register. STBR points to the segment table in the memory.
- ● STLR stands for Segment Table Length Register. It holds the number of segments available in a process.

14. **Consider a logical address space of eight pages of 1024 (I K) words each, mapped onto a physical memory of 32 frames. How many bits are there in the logical address? How many bits are there in the physical address?**

   Since 32 frames each logical address have 5 bits (since $2^5 = 32$) to represent page number p and since each page of length 1024 10 bits (since $2^{10} = 1024$) to represent displacement d. Hence total number of bits in logical address is 15.

Since the total physical memory size is 32 K, the number bits in physical memory is also 15 bits (Since 2^{15} = 32 K).

15. **Consider a paging system with the page table stored in memory and the memory reference takes 200 nanoseconds,**
    a) **How long a paged memory reference take?**
    200 ns for referring the page table in the main memory to find the frame number and physical address, and 200 ns for actually access the instruction a total of 400 ns.
    b) **If we add associative registers, and 75 percent of all page-table references are found in the associative registers with 20 ns to access associative registers, what is the effective memory reference time?**
    Time to access a page whose entry is in associative registers = 200 + 20 = 220 ns
    Time to access a page whose entry is in main memory= 200 + 200 = 400 ns
    Since 75% of page table in associate register the hit is 0.75 and miss is 0.25
    Effective access time = 0.75 X 220 + 0.25 X 400 = 265 ns.
16. **Consider the following segment table entry, with segment number 1, base 2300 and length 100, what is the physical address for the logical address 1, 10?**
    In logical address 1 stand for segment number and 10 stands for displacement.
    Since the segment starts from address 2300, the physical address is 2300 + 10 = 2310.
17. **What is virtual memory? Mention its advantages.**

✧ Allows execution of processes **not fully in main memory**.
✧ Abstracts physical memory into a **large uniform logical space**.
✧ Separates **logical (user) memory** from **physical memory**.
✧ Enables programs to be **larger than physical memory**.

18. **What is demand paging?**
    Demand paging is a technique where a page is loaded only when it is referenced.

Keyword	**Point**
Page Fault	Missing page triggers a **page fault** to load the page.
Efficiency	Reduces **I/O** by loading fewer pages initially.

Keyword	Point
Performance	May cause **page faults** but improves overall memory use.

19. What is virtual address and virtual address space?

- ✓ Virtual address is a logical address generated by the CPU
- ✓ It can be bound to a physical memory location as the program executes.
- ✓ The collection of virtual addresses that can be used by a process is called virtual address space.

20. **What is page fault?**

In a paging virtual memory system, page fault is an event (usually a trap) indicating that a referenced page is not currently loaded in the main memory.

21. **What is the purpose of valid-invalid bit in a page table?**

The valid-invalid bit for each entry in the page table is used to describe whether a page is in the main memory or in secondary storage.

22. **What is the purpose of dirty bit in a page table?**

- ✧ **Dirty Bit** indicates if a page has been **modified** since loading.
- ✧ If **1**, the page must be **written back** to secondary storage on replacement.
- ✧ If **0**, the page is **clean** and need not be saved again.
- ✧ Helps **optimize** page replacement by avoiding unnecessary writes
- ✧ **Example**: If a page is modified during use (dirty bit = 1), it's saved; otherwise, skipped.

23. **Give the formula for effective access time for demand paging**.

The formula is

Effective access time = (1 - p) X ma + p X page fault time

Where p is the probability of a page fault and ma is the memory access time.

24. What is Belady's anomaly?

Belady's anomaly is one in which a page replacement algorithm may perform worse (leads to more number of page faults) if it has more page frames compared to a lesser number of page frames. This anomaly is found in FIFO page replacement algorithm and hence is also called as FIFO anomaly.

25. What is optimal page replacement algorithm?

Optimal page replacement algorithm is one which replaces a page that will not be used for the longest period of time. An optimal algorithm will never suffer from Belady's anomaly.

26. **Define thrashing. What is the cause for thrashing**

- ✧ **Thrashing**: Excessive paging where pages are swapped in and out repeatedly.
- ✧ **CPU Overhead**: CPU spends most time on swapping, not useful work.

- **Cause**: Insufficient page frames allocated to a process.
- **Effect**: Severe performance degradation.
- **Solution**: Increase allocated frames or reduce multiprogramming level