

DAY 19 - 111 DAYS VERIFICATION CHALLENGE

Topic: DMA, Synchronization & Paging

Skill: Computer Architecture

DAY 19 CHALLENGE:

1. Explain DMA (Direct Memory Access) with diagrams.

Direct Memory Access (DMA) is a feature of computer systems that allows certain hardware subsystems to access main system memory independently of the central processing unit (CPU). DMA is particularly useful for data transfer tasks that would otherwise occupy the CPU, such as transferring data between I/O devices and memory.

Explanation of Direct Memory Access (DMA)

Key Components:

1. **CPU (Central Processing Unit):**
 - Executes program instructions and initiates DMA transfers.
2. **DMA Controller:**
 - A special hardware component that manages the direct memory access process.
 - It takes over the data transfer process from the CPU to reduce its load.
3. **Memory (RAM):**
 - The main system memory where data is stored and accessed.
4. **I/O Devices:**
 - External devices like hard drives, graphics cards, network cards, etc., that need to transfer data to and from memory.

Working of DMA:

1. **Initialization:**
 - The CPU sets up the DMA controller by providing it with the necessary information: the source and destination addresses, the size of the data to be transferred, and the direction of transfer.
2. **Transfer:**
 - The DMA controller takes over the bus and starts transferring data directly between memory and the I/O device without involving the CPU.
3. **Completion:**
 - Once the data transfer is complete, the DMA controller sends an interrupt to the CPU to signal the end of the transfer. The CPU then resumes normal processing.

2. What are advantages & disadvantages of DMA

Advantages:

- Increased efficiency
- Parallelism and concurrent operations
- High throughput
- Reduced latency
- Efficient peripheral handling

Disadvantages:

- Increased complexity
- Potential resource contention
- Limited control during transfers
- Security concerns
- Interrupt overhead
- Compatibility issues

3. What is meant by Synchronization in computer architecture?

In computer architecture, synchronization refers to the coordination of concurrent processes to ensure that they execute in a controlled manner, avoiding conflicts and ensuring correct sequencing. This often involves managing access to shared resources (such as memory, data structures, or I/O devices) to prevent issues like data races, deadlocks, and inconsistent states. Synchronization can be achieved using mechanisms such as locks, semaphores, barriers, and condition variables.

4. What are various synchronization techniques?

Various synchronization techniques are used in computer architecture to manage access to shared resources and ensure correct sequencing of processes. These techniques include:

1. Locks (Mutexes):

- **Mutual Exclusion (Mutex):** Ensures that only one thread can access a critical section of code or a shared resource at a time.
- **Spinlocks:** A type of lock where a thread repeatedly checks for lock availability in a loop (busy-waiting).

2. Semaphores:

- **Binary Semaphores:** Similar to mutexes, they can only take values 0 or 1, used to manage access to a single resource.
- **Counting Semaphores:** Allow access to a fixed number of resources. The semaphore value can be greater than 1.

3. Monitors:

- A high-level synchronization construct that combines mutual exclusion and condition variables. Provides a mechanism for

threads to wait for certain conditions to be met within a critical section.

4. Condition Variables:

- Used with locks to allow threads to wait for certain conditions to be true. Threads can be put to sleep and be notified when the condition changes.

5. Barriers:

- Synchronization points where threads or processes must wait until all participants have reached the barrier before proceeding.

6. Read-Write Locks:

- Allows multiple threads to read a shared resource concurrently while ensuring exclusive access for writing.

7. Atomic Operations:

- Low-level synchronization primitives that ensure a series of instructions execute as a single, indivisible operation. Examples include atomic counters and compare-and-swap instructions.

8. Futures and Promises:

- Used in concurrent programming to represent values that will be available at some point in the future, facilitating synchronization between producer and consumer threads.

9. Message Passing:

- Threads or processes communicate and synchronize by sending messages to each other, avoiding the need for shared memory.

5. What is Paging? What is a Page Table?

Paging

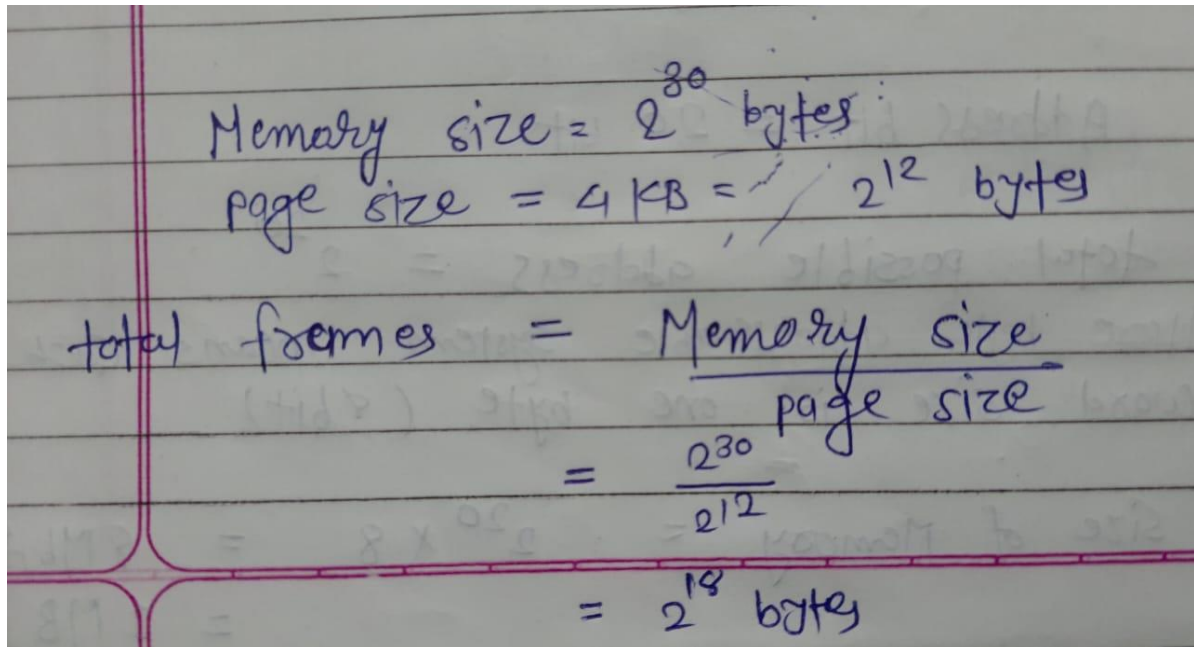
Paging is a memory management scheme that eliminates the need for contiguous allocation of physical memory by dividing both physical and virtual memory into fixed-sized blocks called pages and page frames, respectively. When a program accesses memory, the virtual address is translated into a physical address via a page table. This allows the system to use memory more efficiently and simplifies memory allocation.

Page Table

A page table is a data structure used by the operating system to map virtual addresses to physical addresses. Each process has its own page table, which maintains the mapping of its virtual memory pages to physical memory frames. The page table helps the CPU translate virtual addresses into physical addresses during memory accesses, ensuring correct data retrieval and storage.

6. Find the total number of frames if size of the main memory is 2^{30} bytes, the page size is 4 KB and the size of each page table entry is 32-bit.

(Hint: size of the main memory = Total number of frames * page size)

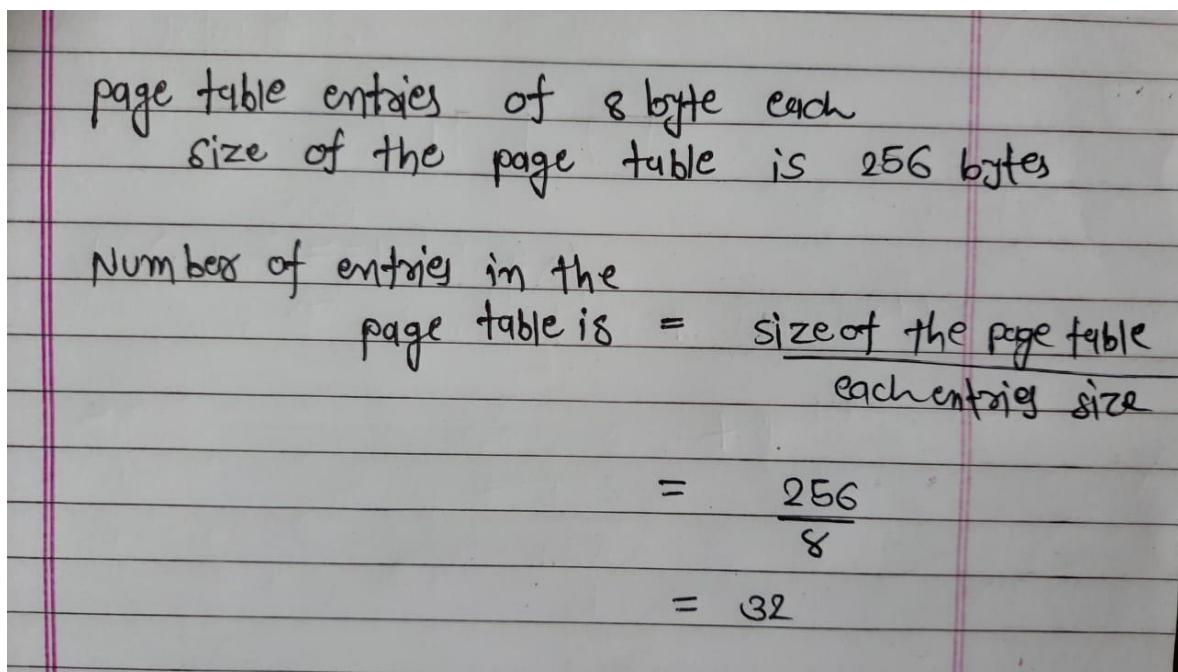


Handwritten solution for Question 6:

$$\begin{aligned}\text{Memory size} &= 2^{30} \text{ bytes} \\ \text{page size} &= 4 \text{ KB} = 2^{12} \text{ bytes} \\ \text{total frames} &= \frac{\text{Memory size}}{\text{page size}} \\ &= \frac{2^{30}}{2^{12}} \\ &= 2^{18} \text{ bytes}\end{aligned}$$

7. Consider a system with page table entries of 8 bytes each. If the size of the page table is 256 bytes, what is the number of entries in the page table?

(Hint: Size of page table = Number of entries in the page table x page table entry size)



Handwritten solution for Question 7:

$$\begin{aligned}\text{page table entries of 8 byte each} \\ \text{size of the page table is 256 bytes} \\ \text{Number of entries in the} \\ \text{page table is} &= \frac{\text{size of the page table}}{\text{each entry size}} \\ &= \frac{256}{8} \\ &= 32\end{aligned}$$

8. Consider a machine with 32-bit logical addresses, 4 KB page size and page table entries of 4 bytes each.

Find the size of the page table in bytes. Assume the memory is byte addressable.

(Hint: Size of page table = Number of entries in page table x Page table entry size)

Number of entries in page table = Process size / Page size)

Handwritten solution on lined paper:

32 bit logical addresses ,
4 KB page size
page entries 4 bytes each

total possible address = 2^{32} bits
page size = 4KB = 2^{12} bytes

Number of entries = $\frac{2^{32}}{2^{12} \text{ byte/page}} = 2^{32-12} = 2^{20}$

Size of the pagetable = no of entries x size of each page table
= $2^{20} \times 4 \text{ bytes}$
= 4MB