DELUXE
PAGE NO.:
DATE :

Assignment - 1

1) What is bus arbitration? Explain centralized arbitration & distributed arbitration in detail.

→ Bus arbitration is the process by which multiple device on a shared bus system. Complete to gain control of the bus for data transfer.

Since only one device can communicate on the bus at a time, arbitration ensure that no two devices attempt to use the bus simulteneously which would cause data curruption or system instability.

Centralized Arbitration: In centralized arbitration, a dedicated device called the bus arbites is responsible for recieving request from various devices that want to use the bus. determining the priority of these requests. and granting access to the highest priority device. centralized arbitration is simple and effective for systems with a limited number of devices.

Distributed Arbitration: In distributed arbitration, there is no central abiter. Instead, each device connected to the bus has its own arbitration is more complex because each device must have capabi-

lity

to participat in process

2) Explain the following term:
i) PCI BUS ii) SCSS BUS

→ (i) PCI BUS (Peripheral Component Interconnect) PCI is a high-performance bead bus standard used to connect internal hardware device to a compliteris CPU and memory The PCI bus allow peripheral lik graphic card, network card, and sound code to communicate with the processor at a high data rate. PCI support both 32-bit & 64-bit data transfer & offer play and play functionallity meaning that devices can be easily added or removed without manually configuring the system.

(ii) SCSI Bus (Small Computer System Interface).
SCSI is a set of standard for connecting and transferring data bet computer and peripheral device, It supports a wide range of device such as has drive, CD-ROM deives, scanner and printers. Unlike PCI, which is primarily used within the computer SCSI can be used to connect. external device, making it a versatile choice for system requiring multiple types of peripheral connections, SCSI can support up to 16 devices on a single bus and offer relatively high data transfer rate making it well suited for services.

3) Explain the concept of Direct memory Access (DMA) and its advantage in I/O operation.

→ Direct memory Access (DMA) is a method of transfering data between memory & peripheral without involving the CPU for every single transaction.
In a typical system, the CPU is responsible for managing all data transfe between devices and memory.
However, this can be inefficient becaus the CPU must pause ir other tasks to handle the I/o operation.
DMA allows devices to transfer data directly to and from memory, by passing the CPU.

Advantage of DMA:
1) By offloading data transfer to DMA controller, the CPU is free to perform other tasks, improving overall system performance.

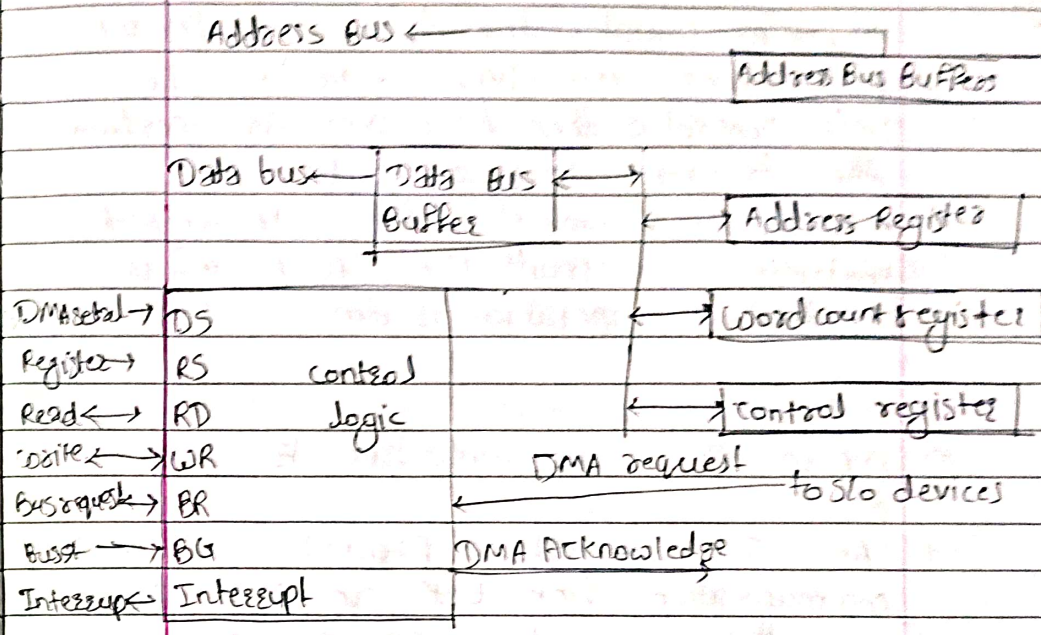
ii) DMA allows data to be transfered in large blocks without CPU intervention, enabling faster I/O operations, especially with high-speed devices.

iii) Since CPU is not involved in the actual data transfer, the overall processing load on the CPU is reduced, making the system more efficient, particularly in I/O intensive operations.

iv) DMA can transfer data with minimal delay which is particularly useful for real-time application that require immediate data handling.

4) What is DMA? Explain two channel DMA controller with block diagram.

→ DMA is mechanism that allows peripheral devices to access system memory directly by passing the CPU. A two-channel DMA controller allows two peripheral to perform data transfer independently, thus improving overall throughput



Control Register:- These define the source and destination address for the data transfer the size of the data to be transferred and other parameter.

Bus interface: This allows the DMA controller to communicate with system bus and memory.

Interrupt logic: The DMA controller can generate interrupt to notify the CPU. When a transfer is complete or if an error occurs.

The CPU initiates the data transfer by programming the DMA controller. The DMA controller then takes over and performs the transfer automatically. Once the transfer is complete, the DMA ~~complete~~ controller interrupts the CPU to inform it that the operation is done.

5) Explain The I/O interface for a input device.

→ An I/O interface provides the necessary communication link between an input device and the computer's CPU and memory.

Consider a keyboard as an input device The I/O device. The I/O interface consists of control and status registers, data buffer and logic unit that handles communication between the device and the system bus.

Input Device
↓
I/O Interface
↓
System Bus
↓
CPU / memory

The I/O interface translate the signals from the input device into a format that the CPU can understand.
It manage the flow of data to prevent overflow or underflow by using buffer to store the data temporarily until the CPU is ready to process it.

6) What do you mean by interrupt? Explain Interrupt nesting and vectored interrupt.

→ An interrupt is the signal sent by a device to the prossor indicating that it needs immediate attention. When an interupt occur. the CPU stops executing its current instruction and jumps to an Interrupt service Routine (ISR) to handle the interupt.

Interrupt Nesting: Interrupt nesting allows a higher-priority Interrupt to interrupt handler. The CPU save the state of the lower-priority handler - priority interrupt. Once the highest priority-interrupt is handled the CPU resumes the lower priority handler. This ensure that critical tasks are handled first improving system responsiveness.

- Vectored Interrupt: In vectored interrupt systems, each interrupt is assigned a unique vector as address when an interrupt occur, the CPU automatically jumps to the ISR corresponding to the interrupts vector, rather than source of the interrupt. This method reduces the overhead and latency involved in processing interrupt.

7) Describe with help of Diagram a serial port interface.

→

A serial post interface is a type of connection that allows data to be transferred one bit at a time over single communication line. It is commonly used for lon-distance communication between a computer and a peripheral device such as modem or a mouse

Serial post use fewer wires and have dimples protocol compared to parallel park, making them suitable. for communication over longer distances.

8) Discuss the challenge and strategis for handling multiple I/o devices concurrently.

→ Following are the challenges for handling multiple I/o devices concurren

i) Resource Contention: Multiple I/o devices may compete for access to shared resources such as memory, bus handewith on CPU time.

ii) Scheduling Conflict: Managging the timing and priority of request from various devices can be difficult, especially in real-time systems.

iii) Data loss or overrune:- Without proper management, buffer overflow or underflows can occur when devices send or eecieve data faster than the system can handle.

Following are the strategies.

(a) Interrupt-driven I/o: Allows the CPU to respond to I/o request dynamically as they occur, reducing the needfor constant polling.

(b) DMA (Direct Memory Access) Allow peripherals to transfer data directly to memory reducing the CPU's involvement in data transfer.

(c) Buffering and Caching: Using buffers to store incoming and outgoing data can smooth out discrepancies in data rate beth devices and the CPU.

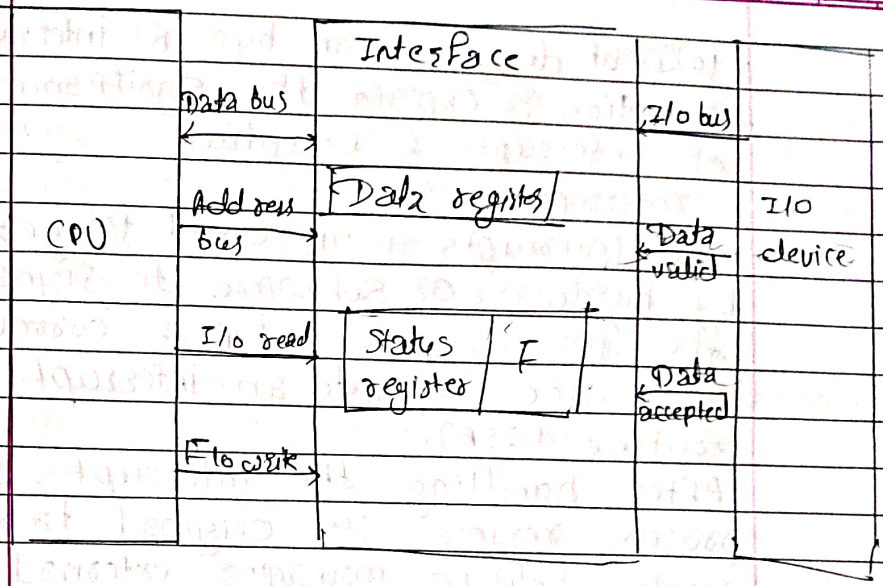
(d) Priority Handling: Assigning priority to devices ensures that critical tasks are serviced first.

(9) Explain programmed I/O and interupt deiven I/O in detail.

→ Programmed I/O:-

In programmed I/O the CPU is responsible for monituring the status of an I/O device and transferring data to and from the device

The CPU continually checks the device to determine it it is ready to send as recieved data. This method inefficient because the CPU must spend a large amount of time, waiting for the device, resulting in wasted processing cycles.



Interrupt-deiven I/O:-

In interrupt-deiven I/O, the CPU. issues a command to the I/O device and then contrida executing other instructions.

When the I/O operation is complet the device sends an interrupt to the CPU which stop its current task and service the I/O request. This method is more efficient than programmed I/O because the CPU. is not idle while waiting for the device to before ready.

10) What do you mean by interrupt & exception? Explain the significance of interrupt & exception

→ Interrupts:

An interrupts is an event triggered by hardware or software to signal the processor to stop it correct task and execute an interrupt service routine (ISR).
After handling the interrupt, the process resume its original task. Interrupts help in managing external or peripheral devices efficiently without constantly polling their status.

Exception:

Exceptions are events that arise due to errors or special conditions in program execution, often due to software issue. like division by zero, invalid memory ~~like division by~~ access, or illegal instruction execution. They are typically caused by the running program and are handled by exception handlers.

Significance of Interrupts & Exceptions:-

Interrupt:

① Interrupts allow the CPU to be used for other tasks while waiting for input / output operations: rather than wasting time in a busy-wait loop.

② Interrupts ensure the system responds quickly to external events, enhancing real-time computing.

③ They facilitate the smooth functioning of multitasking by allowing processes to be paused and resumed.

Exception.

① Exception ensure that error are detected and handled properly, preventing undefined behavior or crash in a system.

② By catching and managing exception system can remain stable and continue functioning after an error occurs.

(11) Differentiate both enabling and disabling interrupt and their significance in I/O operations.

→ Enabling Interrupts:

When interrupts are enabled, the CPU is allowed to respond to hardware or software interrupt request. The processor other tasks, such as handling I/O operations.

Significance⇒

(1) In I/O operation, enabling interrupt allows the system to handle multiple devices effectively, improving system efficiency.

(2) It deminates the need for continous polling, reducing CPU overhead and power consumption.

(3) The CPU can perform other tasks waiting for I/O enchancing multitasking.

Disabling Interrupts:

When interrupts are disabled, the CPU ignore, interrupts requests allowing it to focus slolely on the current task without being preempted by other processes.

Significance⇒

(1) Disabling interrupt can be useful in critical section where it is crucial to avoid interruptions to ensure data integrity or during time-sensitive operation.

(2) However, this can lead to increased response time for peripheral devices, and potentially missing important interrupt signals, degreading system performance in I/O handling.

1. What is cache coherence problem? Discuss the software and hardware approach for cache coherence.

→ The cache coherence problem arises in multiprocessor System where each processor has its own cache. When multiple caches store copies of the same memory location, inconsistencies can occur if one processor updates its cache while others have stale copies.

Software Approaches

1. Lock and Synchronization primitives.
   * Use of mutexes, semaphones or other locking mechanisms ensures that only one processor can moddify shared data at a time.
2. Memory Consistency model.
   * These modeles define the rules for the order of reads and writes across diff-processor. Examples include ~~essential~~ Sequential consistency and weak consistency, which dictate how operations appear to be ordered.
3. Compiler optimizations:
   * Compilers can insert barriers or memory fences that prevent certain optimizations that might violate coherence.

Hardware Approaches

1. Cache Coherence protocols:
   * These are essential mechanisms designed to maintain coherence among caches. common protocols include:-
     * MESI
     * MOESI
     * MSI
2. Snooping
   * In snoopy protocols, caches monitor traffic, a centralized directory keeps track of which caches have copies of each memory block.

~~3~~ 3. Bus-based and Crossbar Interconnects:
   * The underlying architecture can impact Cache Coherence. Bus-based system may experience bottlenecks, while crossbar switches can facilate faster communication and coherence enforcement.

2 What is clustering? What are the sky benefits of clustering? Discuss configuration and operating system design issue.

→

clustering refers to a technique, it's computing a multiple servers or nodes work together as a single system to provide improved performance, reliability & scalability.

Key Benefits of clustering

1. High Availability:
- Clustering can provide redundancy. If one node fails, other can take over its workload, minimizing downtime & ensuring continous services availability

2. Load Balancing
- Distributing workloads across multiple nodes helps optimize resources use & improve response time. This is particularly useful for applications with varying loads.

3. Scalability:
- Clusters can be easily scaled by adding more nodes, This allows systems to grow with increasing demand without major overhauls.

4. Performance Improvement:
- By distributing tasks across multiple nodes, clusters can handle larger volumes of requests. or compute-intensive process more efficiently.

Configure and Operating System Design Issues.

1. Network Configuration:
- A reliable and fast network is crucial for clusters performance. Issues such as network latency, bandwidth, and topology need to be addresed to ensure efficient communication beth nodes.

2. Nodes Communication:
- Mechanism for inter-node communication must be:

3. Load Distribution Algorithms: Deciding how to distribute workloads can affect performance.

4. Data consistency: Maintaing consistency across nodes is critical, especially for distributed databases, Techniques like distributed locking or consensus protocols

5. Fault Detection and Recovery: The system should be able to detected node failure quickly and reroute tasks

6. Resource Management: The operating system must befficiently manage resources like CPU and memory across the cluster

7) Security: Clusters may face unique security challenges, including the need for secure communication b/n nodes and access control.

8) Monitoring and Maintenance: Tools for monitoring the health and performance of the cluster are essential for proactive management and troubleshooting.

3. Define and Differentiate SIMD and MIMD.

→

SIMD (Single Instruction, Multiple Data)
→ SIMD is a parallel computing architecture where a single instructions is executed simultaneously on multiple data points. This is typically used in applications where the same operation needs to be performed on a large set of data, such as in multimedia processing, image processing and scientific computations.

MIMD (MULTIPLE Instruction, Multiple Data)
→ MIMD is a parallel computing architecture where multiple processors execute different instructions on different pieces of data simultaneously. This allows for more flexible in processing as each processor can operate independently on different tasks.

Difference.

Feature	SIMD	MIMD
Instruction Stream	Single instruction for all	Multiple instructions for each
Data Handling	Same operation on multiple data	Different operation on different data
Parallelism Type	Data parallelism	Task parallelism
Flexibility	Less flexible, specific tasks	More flexible, can handle diverse tasks.
complexity	Simpler control logic	More complex due to instruction handling.

5. List and explained designed consideration required for multiprocessor System.

→

Designing a multiprocessor system involves several key considerations to ensure efficient operation, scalability and reliability

1. Architecture Design
- Symmentric vs Asymmetric: Choose betn Symmetric multiprocessor (SMP) System where all proceed share the same resources and asymmetrical systems.
- Shared Memory vs. Distributed Memory:- Decided whether to use a shared memory architecture, which simplifies data sharing but can lead contention.

2. Communication Mechanism:
- Inter processor Communication:- Establish efficient protocol for data exchange betn processor:., such as message passing or shared memory mechanism
- Bandwith and latency and maximize bandwith, essential for performance in a multiprocessor enivornment.

3. Synchronization:
- Locks and Semaphores: Implement Synchronization mechanisms to prevent data races & ensures the processes coordinate access to shared resources.
- Deadlock prevention: Design strategies to avoid deadlock situation where processes become stuck waiting for each other.

4. Cache Coherence.
- Cache Consistency protocol: Design protocols to maintian consistency betn caches in different processors, ensuring that all processors see the same data values.

5. Load Balancing.
- Task Scheduling: Develop algorithms to evenly distribute workloads across processors preventing bottlenecks and ensuring efficient resource utilization.
- Dynamic vs. static Scheduling: Choose between dynamic scheduling

6. Scalability:
- Adding Processors: Ensure that the system can easily incorporate additional processors without significant redesign.
- Performance Scaling: Evaluate how performance scales with the addition of processors and adjust system architecture accordingly.

5. What is Multithreading ? Explain Implicit and Explicit Multithreading.

→ Multithreading is a program technique that allows multiple threads to exist within a single process, enabling concurrent execution of tasks. Each thread represents a separate path of execution, sharing the same memory space and resources of the process while maintaining its own execution context.

Types of Multithreading.

1. Implicite Multithreading.

Implicite multithreading occurs when the system or runtime enivornment automatically manages the distribution of tasks across multiple threads without explicit intervention from the progeammer.

How it works:- The programmer write code that canbe executed in parallel and the compiler or runtime enivornment identifies opportunities for parallelism. The underlying system then handles the details of thread management, such as creating threads and scheduling their execution.

2. Explicite multithreading.

Explicit multithreading requires the programmer to manually create and manage threads. This involve writing code that directly controls the creation, synchroniza and communication beth threads.

How it works:-

The progeammer specifies when to create new thread class in java, programm can create threads, manage their life-cycles, and synchronize them explicitly using mechanism like locks and semaph res.

7) Explain:- i) CUMA ii) Cluster iii) SMP.

→

i) CUMA (Non-Uniform Memory Access)

CUMA is a memory architecture used in multiprocessor systems where the access time to memory varies depending on the memory location relative a processor. In CUMA, each processor has its own local memory and it can access memory attached to other processors, but this access is slower than accessing its own local memory.

ii) Cluster- based SMP (Symmetric Multiprocessing)

Cluster based SMP refers to a system architecture that combines multiple SMP systems into a network of clusters. Each clusters operates independently but can communicate with other clusters.

- CNUMA: Focuses on optimizing memory access patterns for multiprocessor system by using local memory for each processor, improving performance & scalability.

- Cluster based SMP:- Combines ~~memory access~~ multiple SMP system larger account network, allowing for shared memory within cluster.

8. What is pipeling? Explain structural hazard, data hazard and control hazard.

→ Pipeling is a technique used in computer architecture to improve the throughput of instruction processing.

Stages of pipelining.

1. Fetch
2. Decode
3. Execute
4. Memory Access
5. Write Back

This overlapping allows a new instruction to enter the previous instructions have completed, thus increasing the overall instruction throughput.

Types of Hazards in pipelining.

1. Structural Hazard: Occurs when hardware resources required to execute an instruction are insufficient.

2. Data Hazards: Arises when instruction depend on the results of previous instructions that have not yet completed.

3. Control Hazard: Occur due to branch instructions that alter the flow of instruction execution. When a branch is taken, subsequent instruction fetched may be incorrect until the branch decision is resolved

Ex.

```
If R1 == 0 then
    A = B + C
ELSE
    A = D + E
```

DELUXE
PAGE NO.:
DATE :

Resolution Technique:

- Stalling: The pipeline is stalled until the branch outcome is known

- Branch prediction: Predicting the outcome of a branch to continue fetching instructions while the actual condition is evalued.

- Delayed Branched: Rearranging instructions to fill the delay slots with useful work.