</ Operating System Course Project

} /> [

Presented by - Group 14
Tanaya Korhalkar - 76
Shivani Kshirsagar - 77
Mayank Kulkarni - 78
Sanket Kulkarni - 79
Manasi Pandit - 81

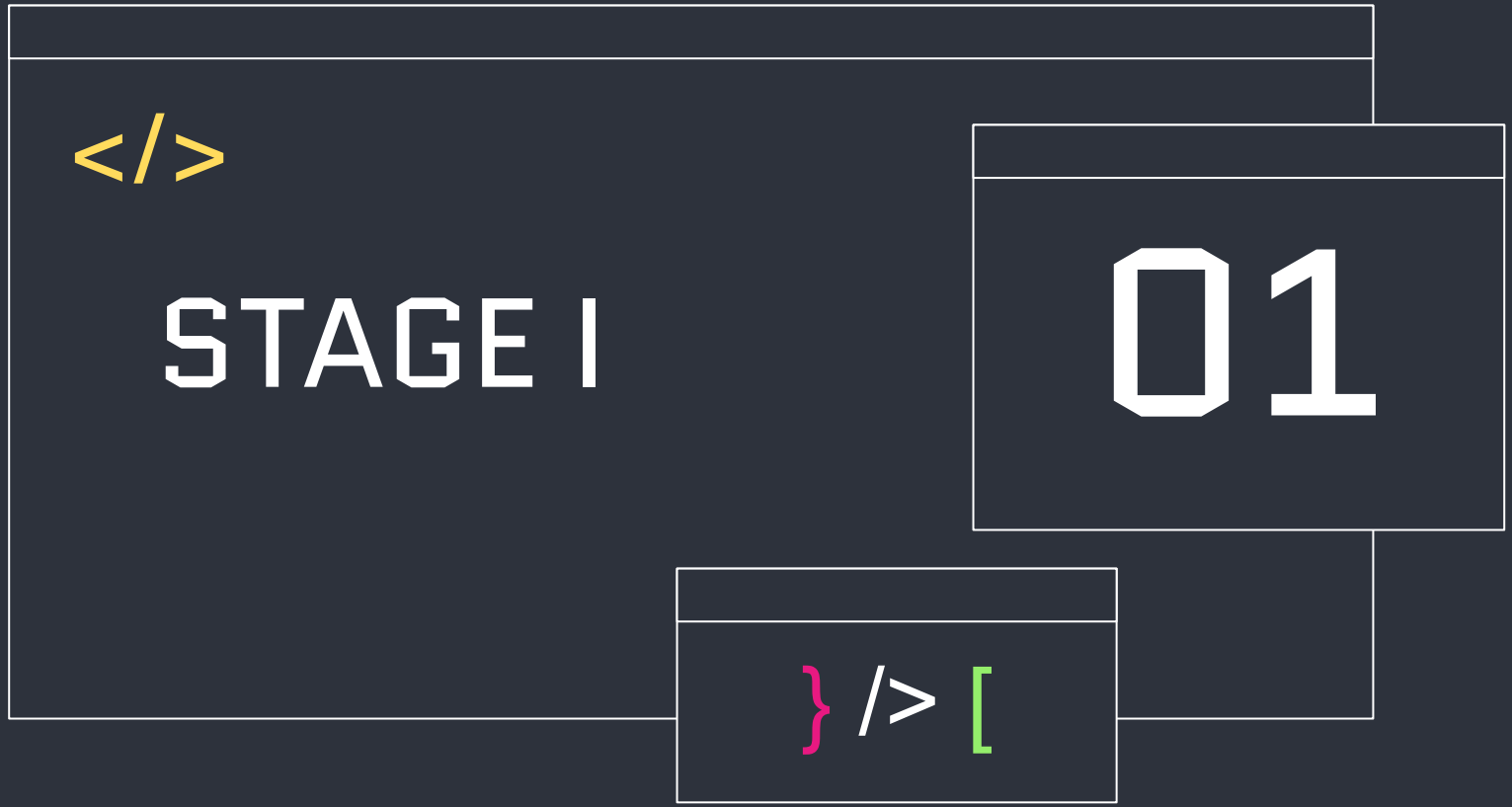
1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

Design and implementation of a Multiprogramming Operating System:

Stage I
- i. CPU/ Machine Simulation
- ii. Supervisor Call through interrupt

Stage II
- i. Paging
- ii. Error Handling
- iii. Interrupt Generation and Servicing
- iv. Process Data Structure

1011 011 01 1011001 10 11011 011 01 110110 110111 1101



1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1

</ Objectives of Stage I

{01}

CPU / Machine simulation

{02}

Supervisor Call Through interrupt

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ CPU/ Machine Simulation

In a multiprogramming system, the CPU plays a crucial role in executing multiple programs efficiently.
To simulate this, we use an OS class, which has several components:

- First, there’s physical memory, which stores data and programs.
- An Instruction Register (IR), which stores the current instruction being executed.
- A General Purpose Register (R) for performing calculations and processing data.
- The Instruction Counter (IC) keeps track of which instruction we are currently on.
- There's also an Interrupt Signal (SI) that triggers different service calls when needed.
- Lastly, a Condition Flag (C) is used to decide if certain instructions should be executed based on specific conditions, and a Buffer that holds input and output data for smoother operations.

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Supervisor Call through Interrupt

- Supervisor Call through Interrupt is a mechanism that allows user programs to request services like reading from or writing to a file.

  The SI variable is used to signal the type of request.
- SI = 1 Read Mode (GD instruction)
- SI = 2 Write Mode(PD instruction)
- SI = 3 Terminate (H instruction)

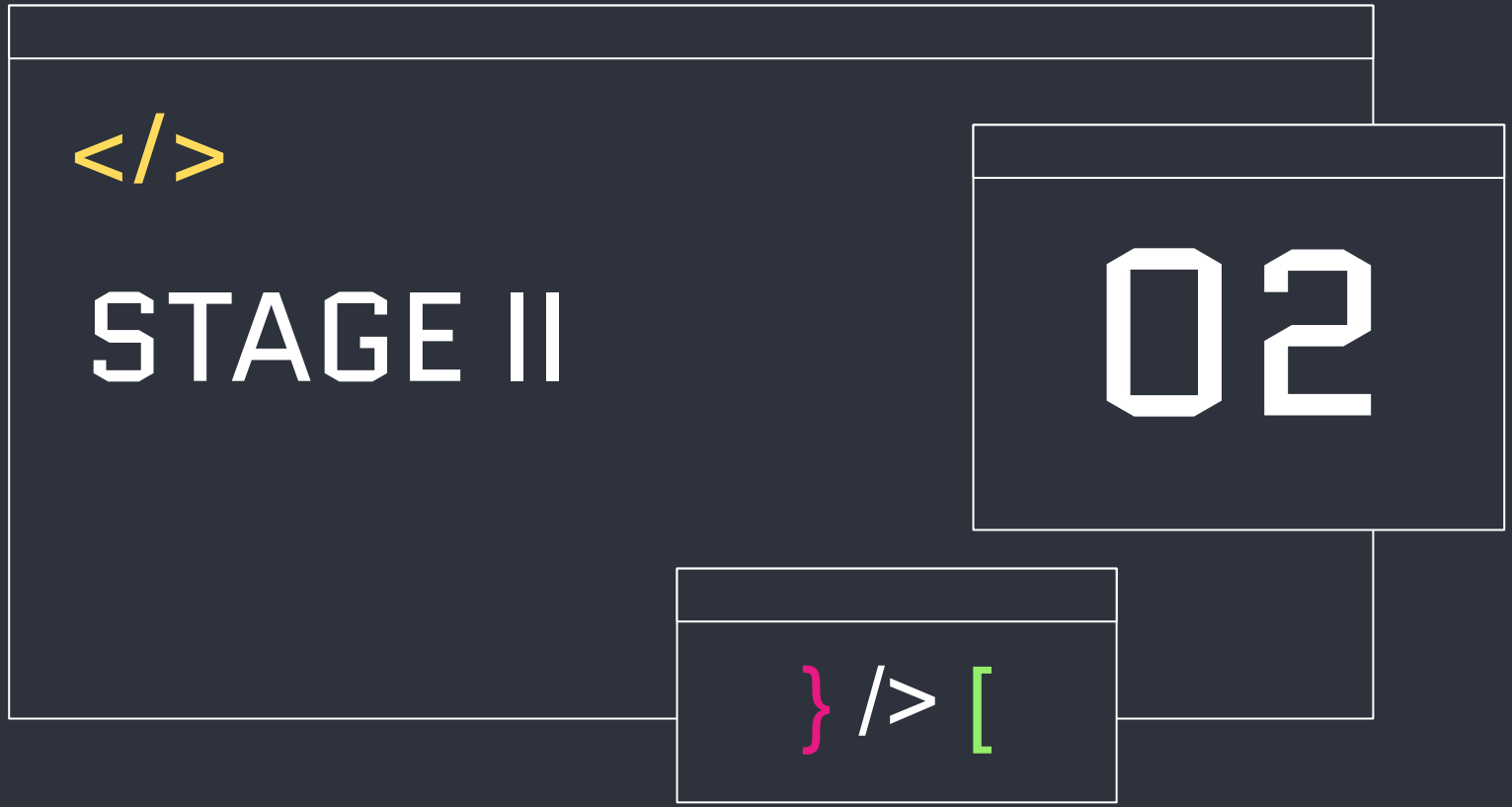
- The MOS (Master Operating System) function handles these requests by performing the necessary input/output operations or halting the system.

- This mechanism allows for multiprogramming by managing service requests efficiently.

- The Interrupt System enforces boundary between user program & system level operations, reflecting real OS behavior.

- Interrupts ensure efficient task handling in multiprogramming.

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1



1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1

</ Objectives of Stage II

{01}

Paging

{02}

Error Handling

{03}

Interrupt Generation and Servicing

{04}

Process Data Structure

1011 011 01 1011001 10 11011 011 01 110110 110111 1101

</ Paging

- Paging is a memory management scheme used by operating systems to manage how data is stored and accessed in memory.
- It divides the physical memory into fixed-sized blocks called frames, and the logical memory (or process memory) into blocks of the same size called pages.
- When a program needs to execute, its pages are loaded into available frames in the physical memory.
- Key Concepts: Pages, Frames, Page table, Logical and Physical addresses.
- Paging is essential for implementing virtual memory, where processes can use more memory than physically available by swapping pages between RAM and disk storage.

</>

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Paging Implementation

Page Table and Frame Allocation:

- The PTR (Page Table Register) serves as the base address for the page table.
- Physical memory (M) is treated as an array where each block (frame) can store instructions or data.
- The allocate() function is used to allocate a new frame for logical page whenever needed.

Address Translation (Logical to Physical):

- The addressMap(int VA) function translates a virtual address (VA) to a real address (RA) by looking up the page table.
- The virtual address is divided into a page number and an offset.
- The page number is used to find the corresponding frame from the page table, and then the offset is added to get the final physical address.

</>

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Paging Implementation

Handling Page Faults:

- When a page fault occurs (i.e., a requested page is not in memory), the system tries to allocate a new frame.
- In MOS(int RA), there is logic for handling page faults by allocating new frames if necessary and updating the page table.

1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1

</ Error Handling

In a multiprogramming operating system, error handling becomes even more critical because multiple processes are sharing the same resources (CPU, memory, I/O devices).

Types of Errors:

- Invalid Instruction
- Memory Bound Violation
- Page Faults
- I/O Errors
- Divide by Zero

Process to handle Errors

- Detection of Error
- Setting the Appropriate Interrupt
- Generating an Error Signal
- Calling the Error Handling Routine
- Reporting the Error
- Graceful Termination of the Program

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Interrupt Generation and Servicing

Interrupts are managed through the SI (Service Interrupt), PI, and TI registers, as well as the
MOS (Master Mode) method.

1. SI Register: Signals service interrupts, e.g., for read/write operations.
2. PI Register: Indicates program interrupts, such as invalid memory access.
3. TI Register: Signals timer interrupts when timing errors occur.
4. MOS Method: Handles interrupts, using an optional real address (RA) for servicing based on the SI, PI, and TI registers.

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Process Data Structure

The process data structure is represented by the PCB (Process Control Block) struct, which contains information about a process, including its state, memory allocation, and other relevant data.

1. PCB Struct: Contains information about a process and is used to manage its execution.
2. Memory Management: The PTR (Page Table Register) manages memory allocation. The addressMap method translates virtual addresses to real addresses using the page table.
3. Process Management: The PCB struct is utilized to manage processes. The load method loads a process into memory, and the execute_user_program method executes the process.

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ THANK YOU />

} /> [

QnA

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 1 1 0 1