Memory Management Unit Simulator Report

Omar Alfarouq Bouhadi - Basma Arnaoui

May 16, 2024

Contents

1 Introduction

The Memory Management Unit (MMU) Simulator is a versatile application designed to illustrate different memory allocation strategies used in operating systems. This tool allows users to interactively simulate the process of allocating and deallocating memory using various strategies such as First Fit, Next Fit, Best Fit, and Worst Fit, providing insights into their operational efficiencies and limitations.

2 Project Overview

2.1 Purpose

The primary purpose of this simulator is to provide an educational tool that helps students and professionals alike understand the complexities involved in dynamic memory management in operating systems.

2.2 Features

- Multiple allocation strategies including First Fit, Next Fit, Best Fit, and Worst Fit.
- An interactive command-line interface for real-time memory management simulation.
- A user-friendly web interface for visual interaction with the memory management system.
- Real-time visualization of memory allocation and deallocation.

3 Technical Details

3.1 Implementation

The simulator is implemented using Python, taking advantage of its powerful libraries to manage the backend logic and Flask to handle the web interface.

3.2 File Structure

```
- memory_manager.py: Contains the core logic for memory management strategies.
- mmu.py: The main entry point of the CLI-based application.
- routes.py: Manages the web interface routing and Flask setup.
```

4 How the App Works

4.1 Command-Line Interface

The CLI allows users to start the simulation by specifying the total memory and the strategy. Users can then execute commands to allocate and deallocate memory, simulate processes, and print the memory map.

4.2 Web Interface

The web interface provides a graphical user interface to interact with the MMU simulator. It allows users to perform all operations available in the CLI in a more intuitive way.

5 Usage

To run the CLI application:

```
python mmu.py <total_memory_in_KB> <strategy_number>
```

To start the web interface:

```
python routes.py
```

Access the web interface at `http://localhost:5000`.

6 Handling Errors and Exceptions

This section illustrates how the MMU handles various errors and exceptions, providing robustness and user feedback.

6.1 Negative Sizes

This subsection describes how the simulator handles attempts to allocate negative sizes, which are not physically plausible.

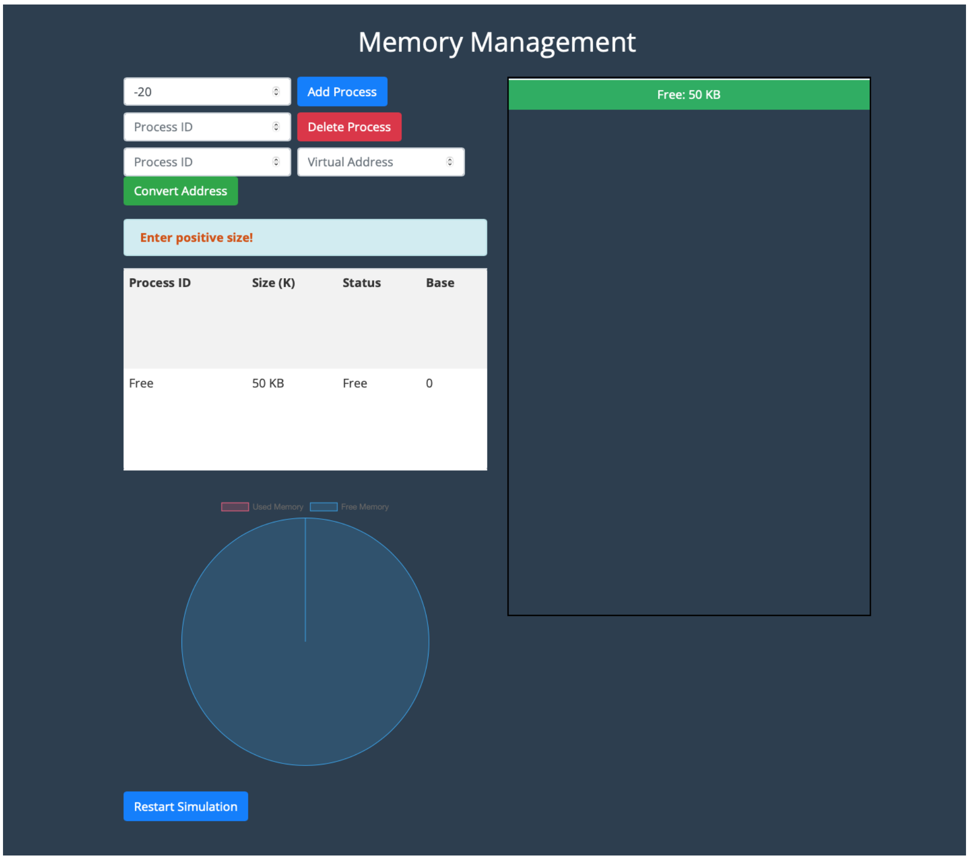


Figure 1: Handling negative sizes input by the user

6.2 Unfound Indices When Deleting

Here, we detail the system's response when a user attempts to delete a process using an ID that does not exist.

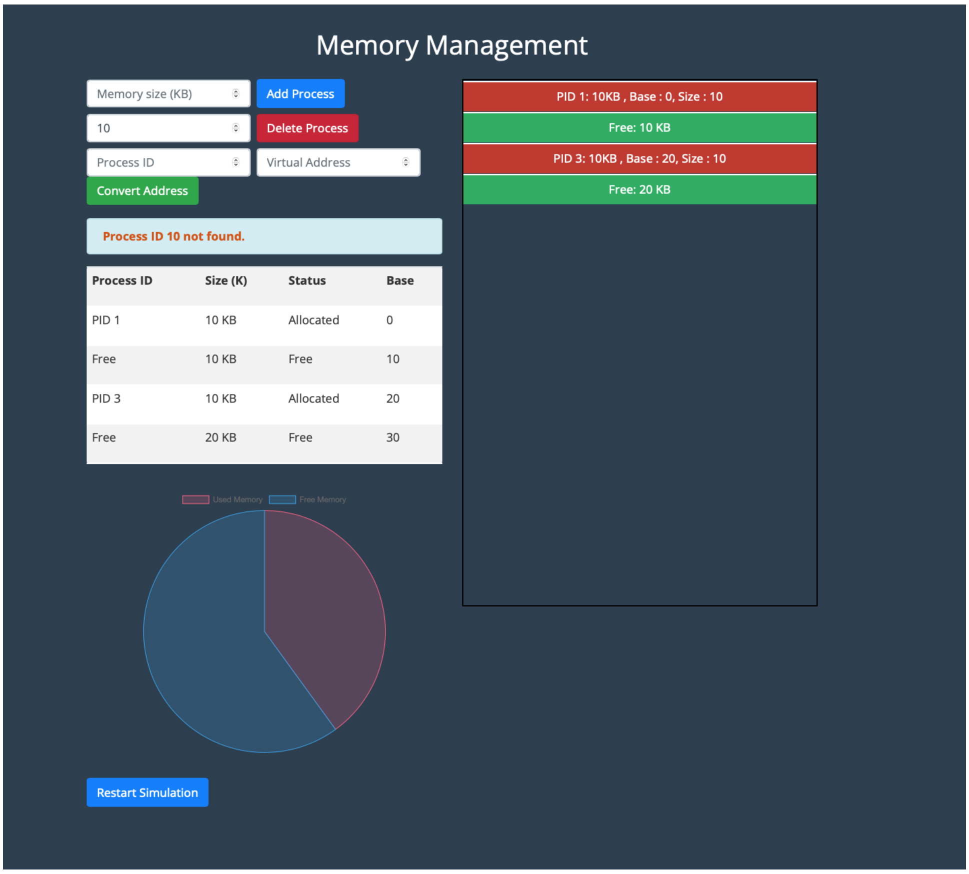


Figure 2: Handling deletion of non-existent process IDs

6.3 Address Out of Address Space

This subsection explains how the simulator reacts when a request is made for a virtual address that exceeds the allocated space.

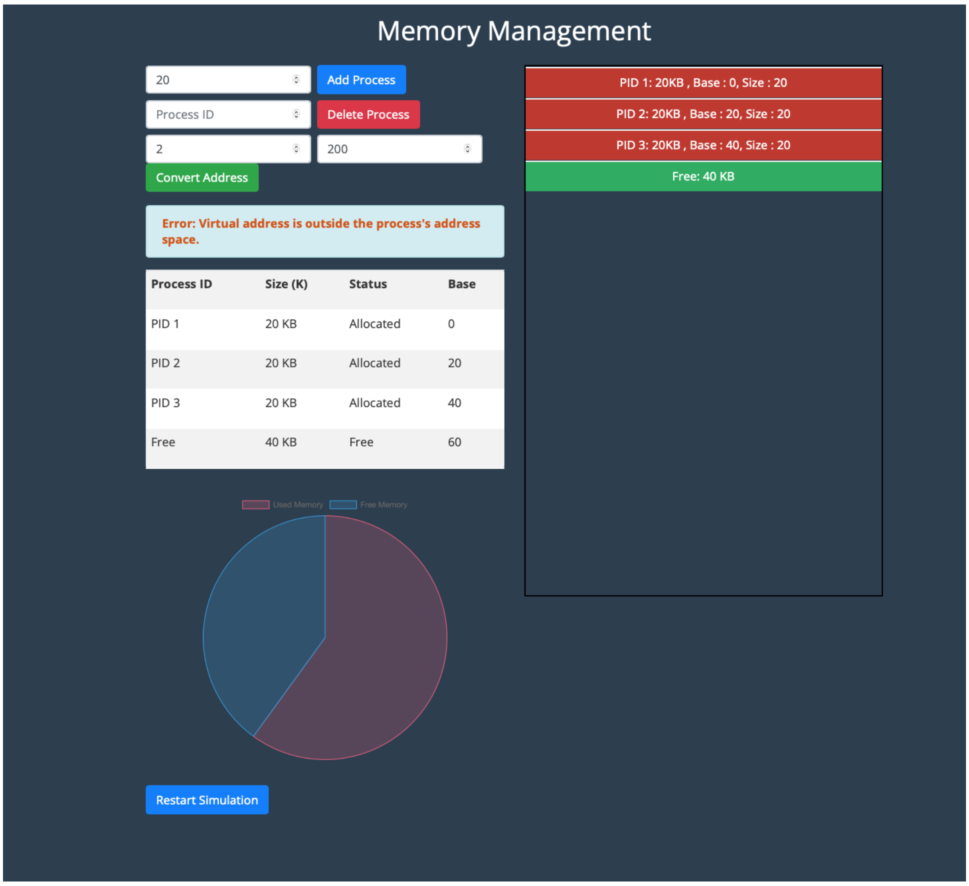


Figure 3: Handling access to a virtual address outside the process's address space

6.4 Not Enough Memory

This section addresses how the simulator notifies the user when there is not enough memory available to fulfill an allocation request.

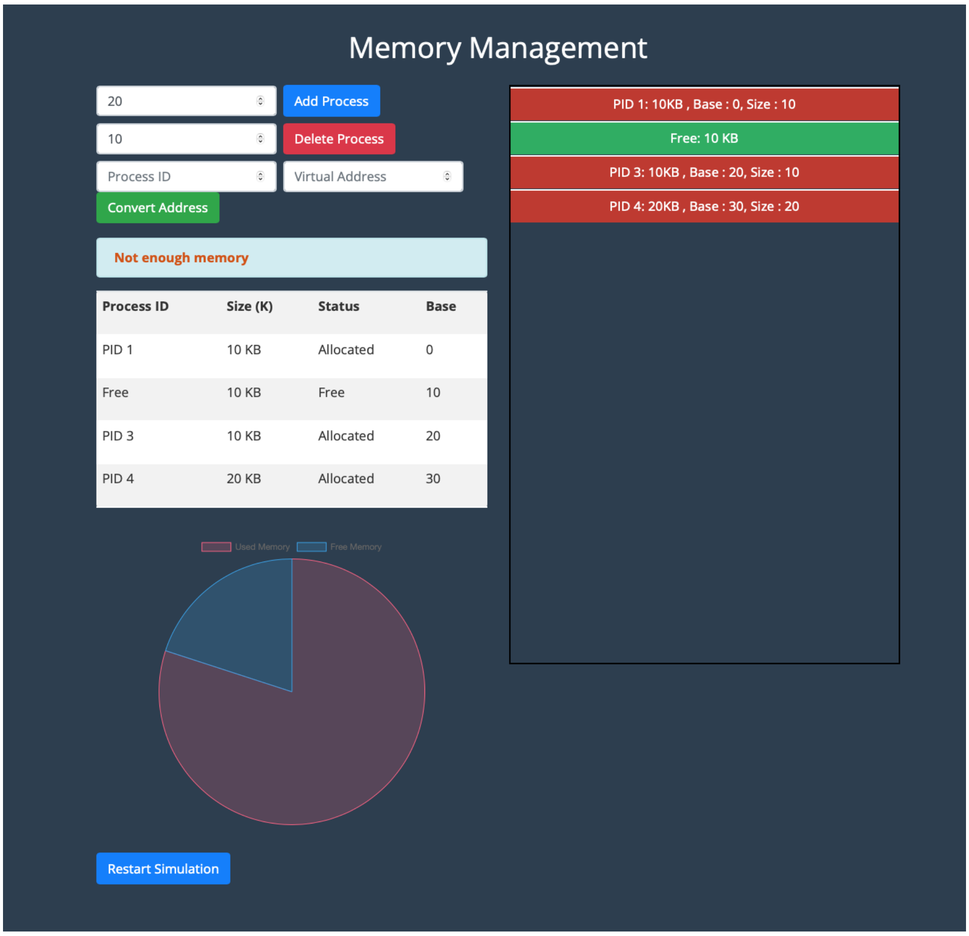
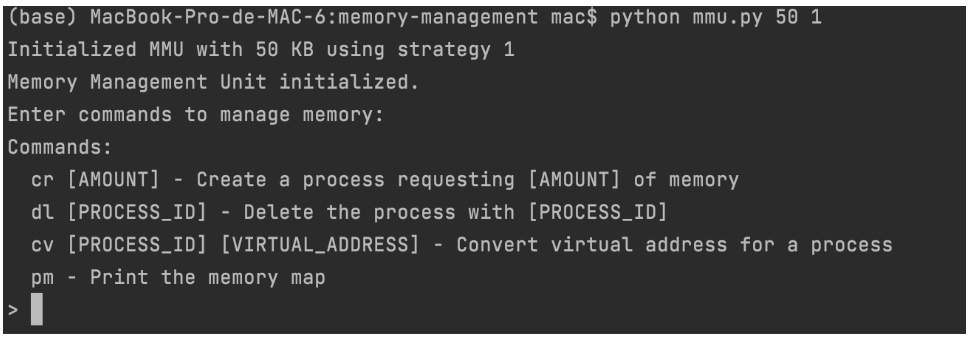


Figure 4: Handling memory allocation failures due to insufficient memory

7 Screenshots

This section contains various screenshots illustrating different aspects and stages of the MMU simulation process.

7.1 Terminal CLI



```
(base) MacBook-Pro-de-MAC-6:memory-management mac$ python mmu.py 50 1
Initialized MMU with 50 KB using strategy 1
Memory Management Unit initialized.
Enter commands to manage memory:
Commands:
  cr [AMOUNT] - Create a process requesting [AMOUNT] of memory
  dl [PROCESS_ID] - Delete the process with [PROCESS_ID]
  cv [PROCESS_ID] [VIRTUAL_ADDRESS] - Convert virtual address for a process
  pm - Print the memory map
>
```

Figure 5: Terminal Command Line Interface

7.2 Terminal Running

```
> cr 10
Created process 1 with Base=0 and Limit(size)=10
> cr 10
Created process 2 with Base=10 and Limit(size)=10
> dl 3
Error: Process not found
> dl 2
Deleted process successfully
> cr 10
Created process 2 with Base=10 and Limit(size)=10
> pm
Memory Map:
Process 1: Base=0, Limit=10, Size=10KB
Process 2: Base=10, Limit=20, Size=10KB
Free: Base=20, Limit=50, Size=30KB
> dl m
Error: Process ID must be an integer.
> s
Invalid command
>
```

Figure 6: Terminal running the MMU simulation

7.3 Landing Page

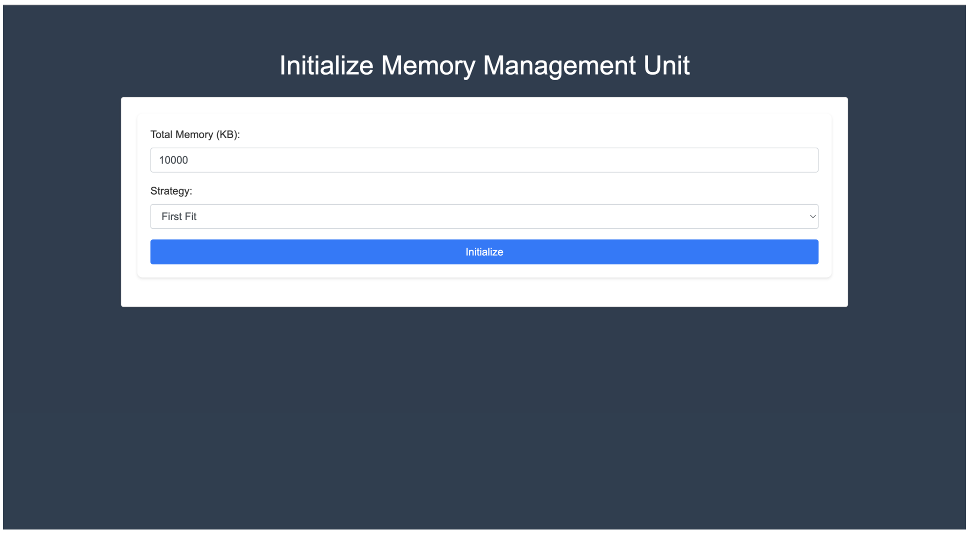


Figure 7: Landing Page - Choice of Total Memory Size + Algorithm

7.4 Creating Processes

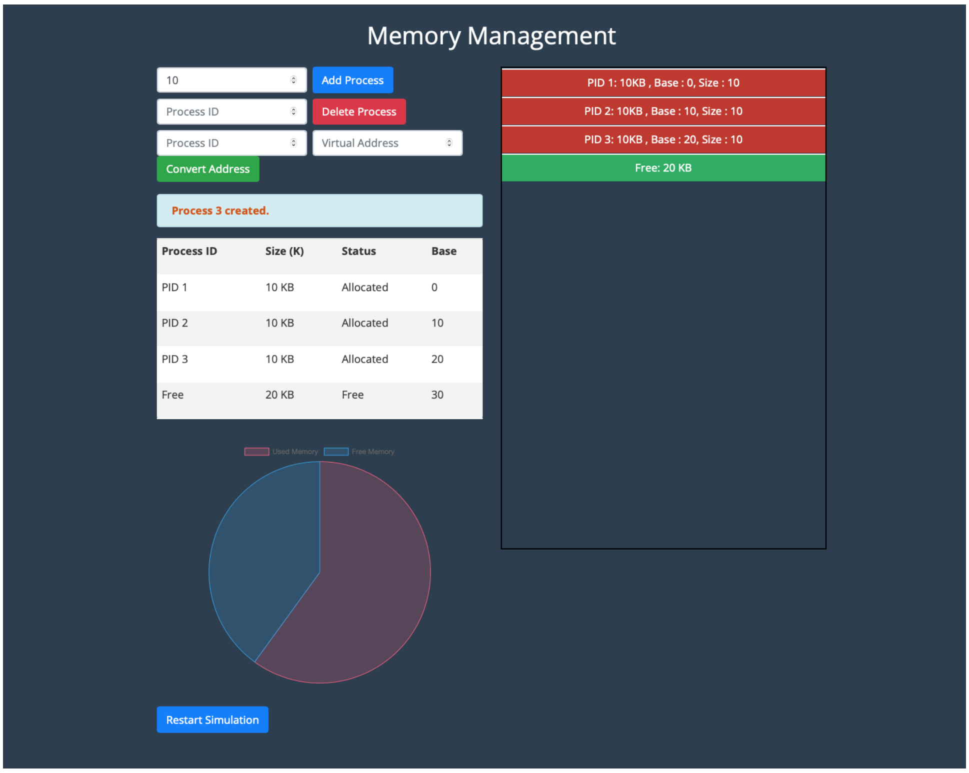


Figure 8: Creating processes in the web interface

7.5 Deleting Processes

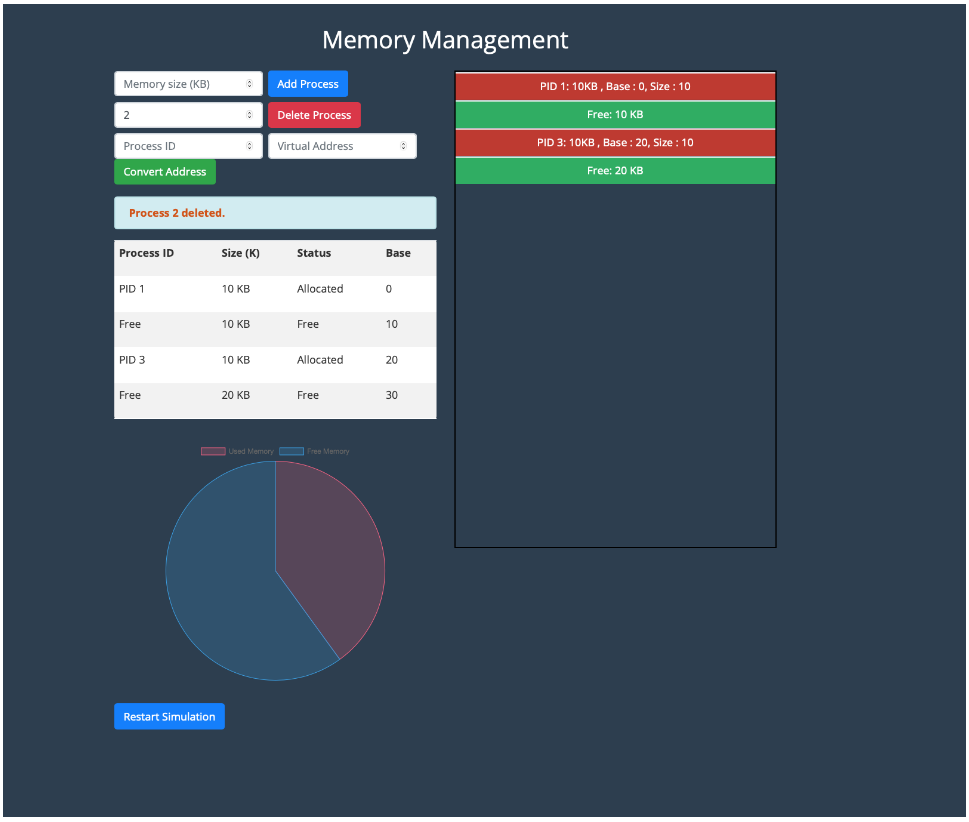


Figure 9: Deleting processes through the web interface

7.6 Converting from Virtual to Physical

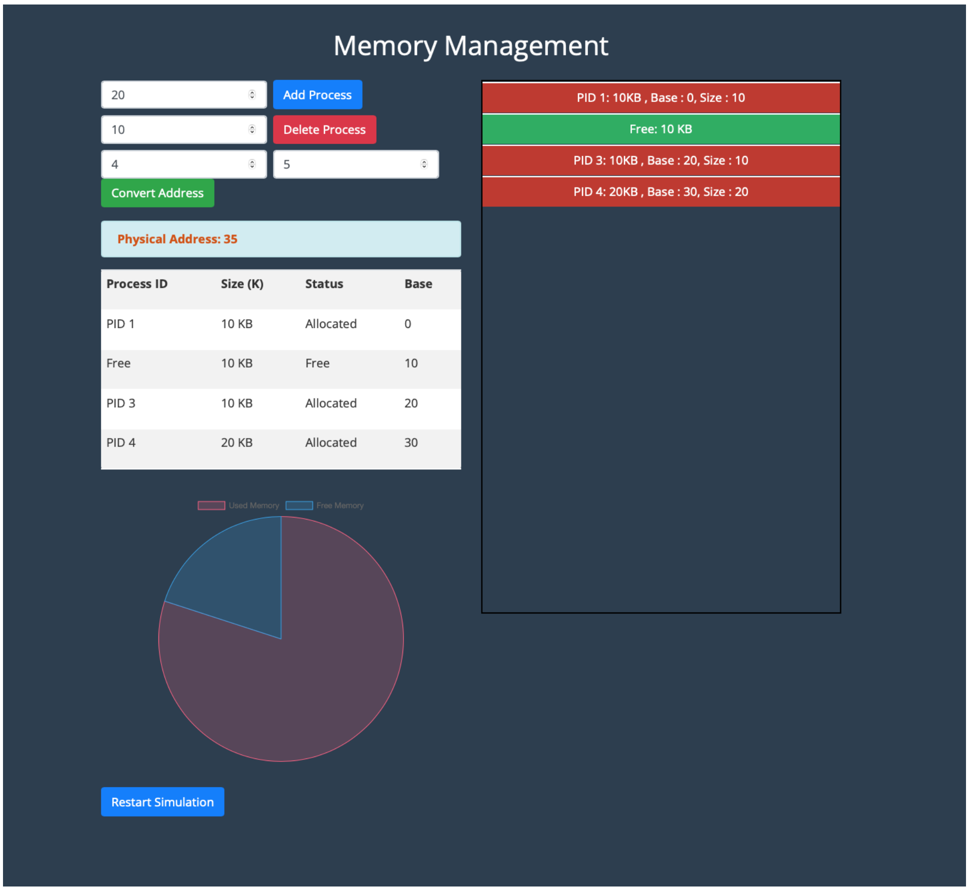


Figure 10: Converting from virtual to physical addresses

8 Conclusion

This report outlines the functionalities and technical details of the MMU Simulator, designed to aid in the understanding of memory management techniques in modern operating systems. It serves as a comprehensive guide to using the simulator effectively, along with troubleshooting common issues encountered during its operation.