2017

Virtuo-ITS: An Interactive Tutoring System to Teach Virtual Memory Concepts of an Operating System

Venkata Krishna Kanth Musunuru
Wright State University

Follow this and additional works at: https://corescholar.libraries.wright.edu/etd_all
Part of the Computer Engineering Commons, and the Computer Sciences Commons

Repository Citation
https://corescholar.libraries.wright.edu/etd_all/1732

This Thesis is brought to you for free and open access by the Theses and Dissertations at CORE Scholar. It has been accepted for inclusion in Browse all Theses and Dissertations by an authorized administrator of CORE Scholar. For more information, please contact corescholar@www.libraries.wright.edu.
VIRTUO - ITS: AN INTERACTIVE TUTORING SYSTEM TO TEACH VIRTUAL MEMORY CONCEPTS OF AN OPERATING SYSTEM

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science

By

VENKATA KRISHNA KANTH MUSUNURU
B.Tech., Jawaharlal Nehru Technological University, India, 2017

2017
Wright State University

____________________________________
Adam R. Bryant, Ph.D.
Thesis Director

____________________________________
Mateen M. Rizki, Ph.D.
Chair, Department of Computer Science and Engineering

Committee on Final Examination

____________________________________
Adam R. Bryant, Ph.D.

____________________________________
Michelle A. Cheatham, Ph.D.

____________________________________
Mateen M. Rizki, Ph.D.

____________________________________
Robert E.W. Fyffe, Ph.D.
Vice President for Research and Dean of the Graduate School
ABSTRACT


Interactive tutoring systems are software applications that help individuals to learn difficult concepts. They can allow students to interact with ideas from essential mathematics to more complicated subjects like software engineering. This thesis concentrates on one such interactive tutoring system (ITS) designed for teaching concepts related to operating system virtual memory.

Operating system concepts can be troublesome to learn without having someone or something to explain them. Even when an instructor is able to provide detailed explanations, it is still exceptionally difficult for students without a computer science foundation to comprehend these concepts. Students require a sophisticated set of mental models to comprehend how various components of the operating system work together. In a lecture, students may find it hard to imagine the various operating system processes or how they work. A tutoring system that visually shows these concepts to students and lets them interact with models of the various components can make learning much easier.

This thesis discusses such an ITS called Virtuo-ITS. The aim of this ITS is to aid individuals in learning virtual memory concepts like paging and virtual address to physical address translation. Virtuo-ITS visually explains concepts of virtual memory and
provides tasks for learners to test their understanding of the concepts. An individual can interact with the system and control the virtual memory processes that are happening to develop a better mental model of each of the concepts. This fulfills the principle point of an ITS, which is to teach difficult concepts simply.

**Keywords:** Interactive tutoring systems, ITS, Virtual memory visualization, Paging, Address transformation, Operating systems.
# Table of Contents

List of Figures ................................................................................................................... vii

List of Tables ..................................................................................................................... ix

ACKNOWLEDGEMENT ...................................................................................................x

1. INTRODUCTION .......................................................................................................1
   1.1. OBJECTIVE: ........................................................................................................1
   1.2. PROBLEM DESCRIPTION: ................................................................................1
   1.3. APPROACH: ........................................................................................................4

2. BACKGROUND .........................................................................................................7
   2.1. INTELLIGENT TUTORING SYSTEMS: .........................................................15
   2.2. VIRTUAL MEMORY OF OPERATING SYSTEMS: ........................................16
       2.2.1. First-In-First-Out(FIFO): .....................................................................17
       2.2.2. OPTIMAL: ............................................................................................18
       2.2.3. Least Recently Used (LRU): ...............................................................19
       2.2.4. Most Frequently Used (MFU): ............................................................20
   2.3. Virtual-to-Physical Address Conversion: .......................................................21

3. DESIGN .....................................................................................................................23
   3.1. UI IMPLEMENTATION: ..................................................................................25
   3.2. Multi-Tasking Background Window: ..............................................................26
   3.3. PAGING ALGORITHMS: ..............................................................................30
       3.3.1. FIFO Page replacement algorithm: ..................................................30
       3.3.2. Optimal Page Replacement Algorithm: .............................................34
       3.3.3. LFU Page Replacement Algorithm: ..................................................37
       3.3.4. MFU Page Replacement Algorithm: ..................................................41
   3.4. Virtual to Physical Address Translation: .......................................................44
   3.5. Nested Page Tables: ......................................................................................47
   3.6. Real-Time Analysis: .......................................................................................48

4. ANALYSIS ................................................................................................................53

5. CONCLUSIONS .......................................................................................................66
   5.1. Summary: .......................................................................................................66
   5.2. Future Work: ..................................................................................................67
6. BIBLIOGRAPHY

..................68
List of Figures

Figure 1: Working of the FIFO page replacement algorithm. .......................................... 18
Figure 2: Working of the Optimal page replacement algorithm...................................... 19
Figure 3: Working of LRU page replacement algorithm.................................................. 20
Figure 4: Working of MFU page replacement algorithm ................................................. 21
Figure 5: Use-case diagram of the ITS system. ............................................................. 25
Figure 6: Multi-Tasking background window. ............................................................... 27
Figure 7: Loading Page with the address found in TLB. .................................................. 28
Figure 8: Load page when the physical address is not present in TLB. ........................... 29
Figure 9: FIFO page replacement window in learning mode. ......................................... 31
Figure 10: FIFO page replacement window in practice mode......................................... 33
Figure 11: Optimal page replacement window in learning mode.................................... 35
Figure 12: Optimal page replacement window in practice window ................................ 36
Figure 13: LFU page replacement window in learning mode. ......................................... 38
Figure 14: LFU page replacement window in practice mode ......................................... 40
Figure 15: MFU page replacement window in learning mode. ....................................... 42
Figure 16: MFU page replacement algorithm in practice mode...................................... 43
Figure 17: Virtual to physical address translation window in learning mode ................. 45
Figure 18: Virtual to physical address translation window in practice mode ................. 46
Figure 19: Nested page table Window ........................................................................ 48
Figure 20: Single process real-time analysis. ................................................................. 49

Figure 21: View of the stats file generated for a single process. ............................... 50

Figure 22: View of the stats file generated for the whole system. ............................ 51

Figure 23: Memory stats of the whole system updated every five seconds. ............... 52
List of Tables

Table 1: Mapping of ITS abilities to Engagement Taxonomy. ........................................ 61
Table 2: Mapping of ITS abilities to Bloom’s taxonomy. ............................................... 61
Table 3: Mapping of categories of Engagement taxonomy on to each window of Virtuo-ITS..................................................................................................................................... 63
ACKNOWLEDGEMENT

This work would not have been possible without the financial support from the Wright State University Department of College of Engineering and Computer Science under Assistantship Award Scholarship. I am greatly indebted to Dr. Ramana Grandhi, Executive Director of International Collaborations and Graduate Programs for the financial support, and Dr. Adam R. Bryant, my thesis director, who have been supportive of my career goals and who worked actively to provide me with the protected academic time to pursue those goals.

I am grateful to all of the members of my Thesis Committee, Dr. Michelle A. Cheatham and Dr. Mateen M. Rizki. They has provided me extensive personal and professional guidance and taught me a great deal about both scientific research and life in general.

I am indebted to the members of my family and my friends. I would like to thank my parents, my sister, and my friend, whose love and guidance are with me in whatever I pursue.
1. INTRODUCTION

1.1. OBJECTIVE:

This thesis describes a tutoring system to visually represent concepts related to operating system virtual memory. The resulting tutoring system provides students with an interactive interface to various objects and processes related to virtual memory. It provides a means to the student through which he/she can learn complex operating system concepts without much effort.

1.2. PROBLEM DESCRIPTION:

The operating systems (OSs) in use today have been through numerous years of development. The first operating systems for use on personal computers were relatively simple, and they lacked the advanced functionalities of contemporary end-user operating systems (Black, 2009). Operating systems have changed a great deal since their introduction. New techniques and optimization of large portions of OS code over the years have led to many new features and dramatic improvements in OS performance, reliability, stability, and security.

Professionals in computing need to know in detail how operating systems function. Students who choose to study computing in most universities are required to complete coursework to gain a solid knowledge of operating systems. Unfortunately, as
one delves deeper into the subject, the concepts are harder to comprehend. For instance, memory management concepts are much more complex than the basic functions of an operating system. Additionally, advanced OS concepts build upon more primary concepts. If students have a visual representation of concepts like memory management, they can build accurate mental models that help them store, retain, and use ideas about memory management more easily.

Many of the major concepts in an OS course are theoretical. Students have a hard time perceiving these concepts and instead prefer practical experience (Hannay, 1992).

“Many of the concepts of virtual memory are dynamic concepts that cannot be easily explained in ... a static medium such as print. While OS texts use images to illustrate concepts, they do not portray the dynamic nature of OS abstractions. Students find it easier to understand when they are grounded to visual objects” (Jones and Newman, 2001).

Visualization is one crucial aspect of tutoring systems and it can help students and teachers in several ways, “from attracting and keeping the attention of students to assisting students in clarifying and understanding complex concepts” (Jones and Newman, 2001). The primary purpose of visualization is to have learners form a solid mental model of the concepts and to have them work on those concepts to gain knowledge of the concepts’ significance (Jones and Newman, 2001).

Especially when dealing with complex subjects, learners should have practical experience that enhances their comprehension of the concepts (Bynum and Camp, 1996). Various fields such as mathematics (Thomas, 1992) and computer programming (Di
Biase and Eisenberg, 1995) have benefited from using visualizations in education. Visualization provides students with environments through which they can interact and use objects in the environment to enhance and test their knowledge of concepts. Concepts that are time-consuming and difficult to be explained by traditional classroom techniques can be learned more easily by using visualizations. They can help students save time, reduce errors and clarify the sequence of particular operations (Lopez, 2004). Particularly, students in operating systems courses can solve programming assignments while aided by the visualization of concepts from their textbooks. In this way, when students execute their programs, they can receive and interpret the program's output as feedback, which helps them understand the concepts better or uncover flaws in their comprehension (Lopez, 2004).

Virtual memory is a fundamental operating systems concept that enables a system to perform multitasking and multiprocessing and to let applications make use of more memory than is currently available in a computer's primary memory store. The operating system uses a portion of the computer's main memory, and rest of the memory will be available for application processes to use. Processes can make use of this free memory to store code, data, and handles to operating system objects needed for their execution. However, when there are many processes to run and limited main memory, the computer cannot allocate enough space for every process. Virtual memory solves this problem.

In virtual memory management schemes, main memory is divided into specific fixed-sized blocks called page frames. Each process is broken down into a number of pages (which are the same size as page frames) and allotted some page frames. When the operating system cannot fit an entire process cannot into its allocated memory space, it
must leave some pages in secondary memory (e.g., a hard disk drive) and bring them into main memory only when they required. Paging is the process of bringing these required pages into memory. Paging is the core concept of virtual memory.

This may appear to seem basic, but there are many decisions and processes involved in paging that a student must comprehend. Several algorithms are involved, such as those that characterize which page needs to be replaced with another page. If the operating system does not choose a page properly, this can cause problems such as thrashing, in which a system spends more time paging than doing actual work. The translation of 32-bit virtual addresses to physical addresses is also a fairly difficult area. A change in a single bit can lead to an entirely different way of looking up a physical address and the rules for translating addresses are not necessarily straightforward.

1.3. APPROACH:

Because of the complexity of virtual memory concepts, an active and interactive approach may help students gain a more detailed understanding than listening to lectures or reading text descriptions. The ITS designed for this thesis focuses on this purpose, to provide individuals with solid mental model of complex concepts associated with virtual memory. An ITS is a software tool that focuses on helping individuals to learn complex concepts without the aid of an instructor. ITSs have the ability to adapt and respond according to how a user interacts with them.

This thesis concentrates on the design and implementation of an ITS for virtual memory of an OS. The ITS discussed in this thesis visualizes the concepts of paging, virtual-to-physical address translation, nested paging, and paging algorithms. Later, the
user has to work on a few tasks through which he will interact with the system to have a better comprehension of these concepts. Virtuo-ITS is designed to be adaptive. It allows a user to complete several tasks, and based on the user's interaction, it will prompt hints to help the user improve his or her performance.

Interaction is the most crucial aspect of this tutoring system. The system involves the user in solving various problems to help him or her comprehend the concepts it presents. The system also allows the user to work in real-time mode to see how a process pages in and out for a specific time. It has a feature to let the user see how a process is accessing main memory at an update interval of five seconds.

This thesis addresses the following questions: 1) How can we design and develop an ITS to consider and overcome the drawbacks of existing systems and tutoring methods? 2) Does the Virtuo-ITS tool adequately visualize the key algorithms of virtual memory concepts in an interactive way? 3) Does this tool adhere to Bloom's taxonomy guidelines for higher level comprehension? 4) Does this interactive tool obtain the satisfactory measures for the visualized virtual memory algorithms as per the Engagement taxonomy?

This thesis is organized into the following chapters: Background (Chapter 2), Design (Chapter 3), Analysis (Chapter 4), and Conclusions (Chapter 5). Chapter 2 gives a review of existing OS simulators and tutoring systems and describes the concepts of memory management of an operating system. It provides an overview of what an ITS is, and discusses its primary elements and features. Later, these are contrasted with the system that we propose. Chapter 3 discusses the ITS design and portrays various
components taken into consideration while designing them. The UI implementation section explains how the UI of our system is composed and how the components are designed. It discusses the factors taken into consideration while designing the system. Chapter 4 describes the analysis and implications of the design in the context of pedagogical theory. Chapter 5 provides conclusions and answers to the questions this thesis aims to address.
Operating systems is a required course for computer science and computer engineering students receiving a bachelor's degree in most educational institutions. In many such institutions, the course involves classroom lectures of theoretical concepts, exams, and programming assignments. Other institutions try a different approach to teach these concepts. These include one of the two following methods: 1) students have access to a fully functional operating system, and have to modify the OS (or write a program that demonstrates one or more key concepts), or 2) students are provided with part of an operating system and have to implement the remaining parts (Mustafa, 2011). In order to do these activities, students first have to understand the concepts very well. Only then can they implement software solutions that take advantage of the concepts.

Virtual memory is a crucial operating system concept. Given the complexity and depth of the topic, it can be very hard to learn in detail. The Virtuo-ITS system discussed in this thesis focuses on the purpose of providing an interactive environment for students to learn virtual memory concepts in detail.

A few operating system simulators exist to help students with these concepts. This section discusses those simulators and how they are used, their benefits, and their drawbacks.
Besim Mustafa, a professor at Edge Hill University, developed such an OS simulator and used it to teach operating system concepts. As part of coursework, students solve programming assignments using this simulator. The assignments give students experience with the concepts of process scheduling, paging, thread management, synchronization, and deadlocks. The paging assignment has students create a process and monitor how pages are swapped in and out of memory. In this way, students can observe paging activity. The simulator has space to hold ten 256-byte pages, so a process can have a maximum of ten pages at the time of its creation. The simulator visualizes how the corresponding pages swap in and out of memory. Students also have the ability to manually invalidate a page table entry, thus forcing a page swap.

According to Mustafa, “An educational resource is of little value if it does not address the educational needs of the students” (Mustafa, 2011). So, he conducted some user experiments with the simulator and found that there is an increase in the learning curve of the students who used the simulators when compared with those students who did not use simulators.

The main downside of this simulator is that it is very weak in the areas of visualization and it does not show all the required elements.

Another group at Columbia University, New York, tried a similar approach; however, they did not use simulators but instead used a real end-user operating system kernel. In this approach, students have to make necessary changes to the OS kernel's code to complete a series of kernel programming projects. The projects focuses on five core OS topics: 1) system calls and processes, 2) synchronization mechanisms, 3) processor
scheduling, 4) virtual memory, and 5) file systems (Laadan, Nieh, and Viennot, 2011). According to the authors, since virtual memory is one of the most complex aspects, it is to be done at the later stage.

In the virtual memory project, students have to measure and visualize how memory pages are managed and used, and then they create programs that produce different memory access patterns (Laadan, Nieh, and Viennot, 2011). The topics covered by the virtual memory project are page table management, page faults, and the copy-on-write mechanism. If students cannot understand the concepts, it is extremely difficult for them to complete these projects. This is the kind of situation in which a tool like Virtuo-ITS may be useful.

Michael D. Black, at American University in Washington D.C., also used a similar approach for teaching operating systems. He followed the approach discussed earlier in which students have access to a part of the OS, and they must implement the remaining functionality. Here, the students have to use functions provided to them in files they are provided and complete the projects using the C programming language. At the end of the course, the OS built by them has five source files: kernel.C, kernel.asm, shell.asm, shell.c and bootload.asm. Out of these five files, students have to program kernel.c and shell.c from scratch as a part of different projects in the course curriculum (Black, 2009).

These projects cover only some of the topics, while leaving other important OS concepts untouched. The projects cover the topics of system calls and interrupts, file systems, and process scheduling. However, the projects only cover round-robin
scheduling and do not include memory management and virtual memory concepts. The author emphasized work on file systems so as to make the OS behave like an OS students would be used to.

Students were assigned a total of five projects, labeled A through E. In project A; students are expected to use kernel functions and write a C program to print “Hello World” to the screen by writing characters directly to video memory (Black, 2009). In project B, the student has to write system call handlers to print the string to the console (Black, 2009). For this project, a new kernel.asm file, which includes an interrupt handler, is available to the student. In project C, the student is expected to write calls to read files from the disk, and also to run programs in the shell (Black, 2009). In project D, the student has to write functions for writing data to files (Black, 2009). In project E, the student has to write programs to make the OS support multiprocessing (Black, 2009).

These projects seem very involved, but according to the author, “providing detailed step-by-step instruction for each stage” is an important factor to make such projects successful (Black, 2009). Students have to complete programming assignments that include making modifications to existing OS code. Trying to learn concepts from multiple sources can be confusing. For an undergraduate student, it could be a daunting task to understand a fully-functional OS and make modifications to it.

One solution for this problem is to use instructional operating systems. An instructional operating system acts as a platform of instruction rather than being an end-user operating system (Anderson and Nguyen, 2004). There are many instructional operating systems that are available to help students learn OS concepts. Although many
of the instructional operating systems are available, students might still find it difficult to learn more complex concepts (Anderson and Nguyen, 2004).

A survey conducted by Anderson reveals the extent to which instructional operating systems are used in some universities. 27% of surveyed universities used a Unix-oriented instructional OS, 17% used Nachos, 14% used the Linux kernel, 8% used a Java-based instructional OS, 7% used Minix, 5% used OS/161, 4% used Xinu, 3% used GeekOS, 3% used Jos, 3% used Yalnix, and the remaining 9% of universities either did not use an instructional OS or the OS they used was represented by under 3% of the instructional operating systems used (Anderson and Nguyen, 2004). The following paragraphs discuss some of these popular instructional operating systems.

Nachos (which stands for "Not Another Completely Heuristic Operating System") is an instructional OS developed at the University of California, Berkeley. This tool is developed in C++ but a Java version is also available since it has advantages in portability. Nachos runs in its own simulator, which simulates the MIPS R2000/3000 instruction set architecture (Anderson and Nguyen, 2004). Nachos falls under the category of a skeleton operating system. Skeleton operating systems consist of only a few functionalities of the end-user operating system (Anderson and Nguyen, 2004). Students have to develop programs for thread management, filesystem, multiprogramming, virtual memory, and networking.

OS/161 is also a skeleton OS. The development of OS/161 considered the drawbacks of Nachos. It was developed at Harvard University and is made to be identical to the BSD Unix OS (Anderson and Nguyen, 2004). The instruction set architecture it
emulates is MIPS R2000. OS/161 is not co-resident with the OS and runs on its own simulator. System/161 is the name given to its simulator. It not only emulates the R2000 instruction set, but it also emulates ports, disks, etc. The assignments given to students using OS/161 are similar to those given to students using Nachos. Students complete assignments on file systems, virtual memory, system calls, and synchronization (Anderson and Nguyen, 2004).

Unlike Nachos, Minix is not a skeleton OS. Minix is a completely functional OS. Minix even has compatibility for Posix. It is Unix like OS, and includes most of the Unix abilities. With most of the functionalities of Unix included in it, the Minix OS is over 30,000 lines of code, with all the comments included (Anderson and Nguyen, 2004). This could prove to be a tiresome task for students. However, the Minix OS consists of a combination of several components. Despite being difficult to understand, students are able to use it, since understanding a separate components is an easier task than understanding an entire operating system.

GeekOS is an instructional operating system developed at the University of Maryland, Collegepark. GeekOS runs on the Bochs emulator and it can also run on Intel-based computers. The source code for GeekOS is not a complete implementation so students must implement source code to add the required functionality (Anderson and Nguyen, 2004). GeekOS is also a skeleton OS. GeekOS is developed in C under Linux. Students' programming assignments relating to Inter Process Communication (IPC), Process scheduling, and File system implementation (Hovemeyer, 2001). The display is a completely text-based display. There is not much visualization involved. Absent from GeekOS are filesystem and virtual memory. To compensate for the lack of disk storage or
filesystem, the GeekOS compiles user programs as data objects linked directly into the kernel (Hovemeyer, 2001). GeekOS has no implementation of virtual memory. Understanding the concepts of virtual memory with a text-based display will be difficult. The text-based display will have the same effect as reading a textbook on a computer screen. Bare metal systems like GeekOS and Minix are difficult to debug, need special tools to develop, and require background knowledge about the machine assembly language (Black, 2009). Implementing advanced concepts like thread management and virtual memory will be a painful task on systems like these.

RCOS.JAVA: RCOS is an instructional operating system. It provides students with animations of the concepts, and students can interact with it to learn about the services, data structures and algorithms involved in operating systems (Jones and Newman, 2001). It is implemented as an MS-DOS application and heavily relies on BIOS calls. This was first developed in C++, later, it was shifted to Java. RCOS has several animator classes for producing animations. The three animators that are available are process scheduler animator, IPC manager animator, and CPU animator. The process scheduler animator displays the Zombie, Ready and Running processes, and it also allows modification to the scheduling algorithm. IPC manager animator displays currently allocated memory, semaphore values, and shared memory (Jones and Newman, 2001). The CPU animator displays CPU register values, stack values, and assembly language instructions (Jones and Newman, 2001).

A Mini Operating System Simulator (MOSS), is an operating system simulator developed entirely using Java. MOSS comprises of various Java classes that implement, rather simulate, the different parts of the operating system (Barnes, 2005). MOSS uses
Java threads to simulate multitasking property of the OS. The downside of this is that it does not simulate the low-level process switching that is present in the assembler. User processes in MOSS are simple Java programs with minimal user interface. MOSS also implements a Unix file system, which is not entirely complete and ignores access rights. The main downside of MOSS is that it just mimics the operating system but does not actually explain anything to the user.

Other OS simulators like JOS, Yalnix, Xinu, OSsim, and topsy are also available. JOS provides Unix-like functions, but its implementation is in exo-kernel style (implemented as a user-level library rather than being built into the kernel) (Anderson and Nguyen, 2004). Yalnix, which is entirely written by students runs on an emulator called UtePC, which does not support disks or demand paging (Anderson and Nguyen, 2004).

Students might find it difficult to learn the concepts of operating systems using simulators. As explained by David Jones, “The simulation may not exhibit exactly the behavior of a typical OS, and as a result, the student must perform the conceptual leap from the behavior of simulation to real OS behavior” (Hovemeyer, 2001). Also, most students find it difficult to learn the concepts of operating systems with real OS. As explained by David Jones, “Using a real OS means that students can experiment with real system and its implementation details. However, the students must climb an even steeper learning curve than involved with the simulated operating system” (Hovemeyer, 2001).

Anderson discovered that there is no single dominant instructional OS. The concept of skeletal operating systems seems to be gaining in popularity over complete
instructional OS (Anderson and Nguyen, 2004). The instructional operating systems that are being developed recently can be executed as applications on computers.

2.1. INTELLIGENT TUTORING SYSTEMS:

Intelligent tutoring systems consist of four components based on a general consensus amongst researchers (Nkambou et al., 2010). These four components are domain, student, tutoring and pedagogical models (Zhen, 2015). Domain model comprises of rules and the steps that should be considered to solve problems. According to Nkambou et al., “It can fulfill several roles: as a source of expert knowledge, a standard for evaluating student’s performance or for detecting errors” (2010). Student model focuses on elucidating the concepts to the students with the help of domain model. The tutor model focuses on how the student is interacting with the system and how he is applying the rules taught to solve the problems. The pedagogical module is accountable for structuring instructional interventions (M.Helander, T.K Landauer, P.Prabhu., 1997). This module operates at two levels. At the curriculum level, it can organize the topics and warrant that a suitable structure is present for learning at each juncture. The other level at which the pedagogical module operates is the problem-solving support level. In this level, it will interfere to instruct the user on problem-solving activities. Towne and Munro (1992) has outlined five various instructional interventions:

1) Performance demonstration: The system illustrates an effective order of actions in a problem-solving environment.

2) Directed step-by-step performance: The system will provide an order of instructions for the user to solve the task.
3) Monitored performance: The user solves the task assigned by the system. The system will interfere when the user makes an error.

4) Goal seeking: The user solves the task assigned by the system. The system monitors the level of abstraction relatively than discrete actions.

5) Free Exploration: The user solves the tasks assigned by the system. However, there is no provision for checking the solution.

Virtuo-ITS system falls under the category of monitored performance. The ITS system discussed in this thesis is a model tracing tutor. Model tracing tutors solve each problem step-by-step along with the student providing assistance as needed. (Helander et al., 1997).

2.2. VIRTUAL MEMORY OF OPERATING SYSTEMS:

Virtual memory is a crucial aspect of an operating system that permits a computer to handle the deficiencies of physical memory by provisionally exchanging pages of data from RAM to a storage disk. With this feature incorporated into the operating system, the programs can address more memory than the actual memory existing on the system. This is the primary benefit with virtual memory. Virtual memory also provides memory protection. Programs only have access to virtual addresses; the translation of virtual-to-physical addresses takes place at the time of execution. Virtual memory supports the execution of multiple processes. There can be cases where the entire program need not be present in memory. Each process is broken down into fixed size pages. The transfer of these pages to and from memory during execution takes place. The user may not use
some features of the program. The pages relating to those features can be loaded into memory when required.

When the physical memory is full, and the system needs to bring pages into memory, then the system removes some pages from the memory and allocates that space for the new pages. The criteria for choosing the pages to be replaced is dependent on page replacement algorithms. Page replacement algorithms determine the page that is to be selected for replacement. There are several page replacement algorithms. The most common page replacement algorithms that professors teach in the operating system lectures are First-In-First-Out (FIFO), Optimal, Least Recently Used (LRU) and Most Frequently Used (MFU). Each of these algorithms is discussed briefly.

2.2.1. First-In-First-Out (FIFO):

The First-In-First-Out (FIFO) page replacement algorithm works on the principle that the page that is in the memory for the longest time interval is the page the algorithm chooses for replacement. This scenario is explained evidently with an example.

Consider the page access sequence to be 236389764352 and the process has only three page frames. The first three frames will be 2, 3, and 6. Later in the page sequence, when the system encounters page 3, which is already present in the memory, there will be no page fault and no need for page replacement. Later, when the system encounters page 8, it is not present in memory. A page fault occurs, and the page has to be brought into memory. Here, there is a need for a page replacement. Among the pages available for that process in memory, the page that is brought first into memory is page 2, so page 8 will
replace page 2. Figure 1 shows working of FIFO page replacement algorithm for the remaining sequence.

![Figure 1: Working of the FIFO page replacement algorithm.](image)

2.2.2. **OPTIMAL:**

As the name optimal suggests, this replacement strategy is the best replacement strategy. However, the drawback of this replacement strategy is that it requires the knowledge of page sequence in advance, which is not the case for most real-time applications. The page that is accessed farthest in the sequence is the pages that the algorithm chooses for replacement. This scenario is explained evidently with an example.

Consider the page access sequence to be 236389764352, and the process has only three page frames. The first three frames will be 2, 3, and 6. Later in the page sequence, when the system encounters page 3, which is already present in the memory, there will be no page fault and no need for page replacement. Later, when the system encounters page
8, it is not present in memory. A page fault occurs, and the page has to be brought into memory. Here, there is a need for a page replacement algorithm. Among the pages available for that process in memory, the page that is accessed farthest in the memory is page 2, so page 8 will replace page 2. Figure 2 shows the working of optimal page replacement algorithm for the remaining sequence.

![Figure 2: Working of the Optimal page replacement algorithm.](image)

2.2.3. Least Recently Used (LRU):

This page replacement strategy is an important strategy used in contemporary operating systems. The pages that are least recently used by the system are the pages that this algorithm chooses for replacement.

Consider the page access sequence to be 236389764352 and the process has only three page frames. The first three frames will be 2, 3, and 6. Later in the page sequence, when the system encounters page 3, which is already present in the memory, there will be no page fault and no need for page replacement. Later, when the system encounters page
8, it is not present in memory. A page fault will occur, and the page has to be brought into memory. Here, there is a need for a page replacement algorithm. Among the pages available for that process in memory, page 2 is the one that is least recently used, so page 8 will replace page 2. Figure 3 shows working of LRU page replacement algorithm for the remaining sequence.

Figure 3: Working of LRU page replacement algorithm

<table>
<thead>
<tr>
<th>Page sequence: 236389764352</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Page faults: 3 3 4 5 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 3 4 5 6</td>
</tr>
</tbody>
</table>

2.2.4. Most Frequently Used (MFU):

This page replacement strategy is one of the many page replacement algorithms that is discussed in operating system courses. The page that is most frequently used is the page that this algorithm chooses for replacement. This is based on the assumption that the page that is most frequently used has its execution part completed, and will not be needed any further.
Consider the page access sequence to be 236389764352 and the process has only three page frames. The first three frames will be 2, 3, and 6. Later in the page sequence, when the system encounters page 3, which is already present in the memory, there will be no page fault and no need for page replacement. Later, when the system encounters page 8, it is not present in memory. A page fault will occur, and the page has to be brought into memory. Among the pages available for that process in memory, the page that is most frequently used is page 3, so page 8 will replace page 2. Figure 4 shows the working of MFU page replacement algorithm for the remaining sequence.

![Page sequence: 236389764352](image)

2.3. Virtual-to-Physical Address Conversion:

As discussed earlier, the programs will have access only to the virtual addresses. During execution, the translation of a virtual address to a physical address takes place. The conversion from virtual-to-physical address uses page tables. The page table will have page table entries which help in the conversion of addresses. However, accessing page
tables and converting the addresses repeatedly can be time-consuming. A Translation Lookaside Buffer (TLB) can be used to decrease the amount of time wasted in conversion. The TLB maintains the list of physical addresses corresponding to respective virtual addresses that. The virtual addresses converted to physical addresses will be present in the TLB. Looking up this buffer before conversion can save execution time when the address that is being converted is present in the TLB.

The contemporary operating systems use two level paging. Having a page table in memory for each process will diminish the amount of physical memory available for the process. Instead, having a page table of page tables will increase the available memory significantly. The first-level page table will contain addresses of the page tables of processes. Having a first-level page table, and another page table relating to the process under execution will guarantee the execution of the process.
3. DESIGN

The design of the Virtuo-ITS system focuses on one single point. It makes the student apply theoretical concepts practically. Having considered the downsides of the current simulators, this system is designed to conquer those issues. This Virtuo-ITS's design ensures that it exemplifies the concepts thoroughly, and tests the students using it to their level of comprehension. The developed ITS system uses swing and AWT in Java. Each window is an instance of JFrame class. The ITS system discussed in this thesis visualizes the following concepts relating to the virtual memory of an operating system.

1. How is multitasking possible in an operating system?
2. What happens in the background when a page fault occurs while multi-tasking?
3. Paging Algorithms- First in First Out (FIFO), Optimal, Least Recently Used (LRU), Most Frequently Used (MFU).
4. How is a virtual address translated to a physical address?
5. How nested page tables work.
6. Real-time page faults of a process and the entire system.

Later, the student performs a few tasks that allow him to interact with the system. The individual's interaction with the system will assess the student’s comprehension of the concept. The system will then adjust in such a way that the more he interacts with it, the deeper the concepts are explained. The tasks require the student to perform page replacement using the knowledge he acquired by clicking on previously created virtual pages. When he does that wrong, then the system prompts him with the right answer and
gives him the correct answer. There are also tasks for the student to convert the virtual address to a physical address using the provided page table. When the student does the conversion right, it implies that he interpreted the concept correctly. If he does it wrong, then Virtuo-ITS provides instructions for the student, guiding him towards the right answer while explaining the concepts again. This will help the student check his understanding of the concepts.

Figure 5 is the use case diagram of the Virtuo-ITS system. It depicts the system's design and how the user has to interact with the system. The user will have to choose one of the topics to learn. Each topic is present in its very own window. There are again two modes available for the student. Learning mode educates about the concepts in a detailed manner by visualizing them to the students. The practice mode lets the user interact with the system and apply the concepts that he/she has learned to solve the given problem.

In practice mode, the user's inputs, while solving the problem, are recorded and are compared against the correct solutions. If the user's response is correct, then the system informs him the same; when the user's response is not right, the system then elucidates the correct solution to the user. This aids the user in self-validation of his perception on the concepts, which is the primary aim of any Interactive Tutoring System. The learning mode is available for all the four page replacement algorithms and also the address translation. The use case diagram in Figure 5 explains clearly what windows are available to the user and how he/she can interact with them.
3.1. **UI IMPLEMENTATION:**

The User Interface (UI) for this system is created using Java. Each concept has a separate window of its own. Each of the windows represents instances of a JFrame class. Considering the downsides in the UIs of current simulators, consideration is taken to ensure a friendly and easily understandable interface of the system. This section describes the UI design of each window. Each concept is designed in its own window, and all of these are joined using a tabbed pane. Thus, the user can learn each concept independently by accessing their respective tabs.
3.2. Multi-Tasking Background Window:

This is the main window that shows up when the tool is active. This window has a module to educate the user about how multi-tasking is conceivable. The user has to select the size of main memory, page size and number of required processes. The user has three choices of main memory sizes, two choices for page sizes, and a maximum number of five processes. The three choices for main memory will be 1GB, 2GB, and 3GB, where GB represents Gigabytes. The two choices for page sizes are 4KB and 8KB, where KB represents Kilobytes.

For better understanding, some anecdotal processes like browser, media player, and game are available. When the user chooses his specifications, and activates the module, then the calculations to determine each process allocated memory will be done based on the specifications selected. The loading of processes into main memory is dependent on these calculations. This window demonstrates to the user how the processes loading into memory takes place, and how the processes are running simultaneously. It also shows how likely the processes are going to generate page faults based on the specifications selected. When there are many processes and less memory it is highly likely that a page fault occurs. In the same screen, the user will likewise have a chance to see how the trap is generated to the system and how user mode switches from user to kernel and looks up page and loads it into memory.

The main memory and virtual memory are represented with different colors to show a distinction between them. All the components in the window will have a separate color of their own. This enables the distinction from one component to another. The operating system occupies some of the main memory, and orange color represents it.
Light green color represents virtual memory, and a green color represents pages of each process. Cyan color represents the page table entries. For the purpose of representation, each process has ten pages. However, a process may have several hundreds or thousands of pages depending on the process. There is a separate blue panel to the right of the window, which depicts what is happening in the system with each click of the NEXT button. Figure 6 shows the background window of the ITS system. For each click of NEXT button, the system will advance a step forward in depicting how the faulted page loads into memory. Small arrow marks are used to illustrate the access of specific part in the system.

Figure 6: Multi-Tasking background window.
When the user selects the required parameters, then the visualizations corresponding to process loading and execution are shown. Depending upon the number of processes the user has selected, the execution also changes. Some processes like games and media players will have more paging activity than other processes like Word and web browsers. The visualization depicting this functionality is also present in the Virtuo-ITS system. When a page fault occurs, then the system searches for the page’s physical address in the translation lookaside buffer (TLB). TLB stores the information of the physical address corresponding to a virtual address. This saves much time when the address is present in TLB, because there will be no need to access the page table and convert the virtual address to a physical address. When the address is present in TLB, then the page is directly loaded into main memory from virtual memory through a disk input/output operation. Figure 7 shows the loading of a faulted page into memory when the page's physical address is readily available from TLB.

![Figure 7: Loading Page with the address found in TLB.](image-url)
This window also visualizes another case where the physical address corresponding to the virtual address is not available in TLB. This case is the most frequent one that happens in the system. When the physical address of the page is not available from the TLB, then the system will access the page table and convert the virtual address to physical address and fetch the page from disk through an input/output operation and restarts from the stage where the process has stopped the execution. Figure 8 shows how a faulted page is loaded into memory when the page's physical address is not found on TLB and has to access the page table and get it from the disk. All the visualized data is present in word format on the right side blue panel. The background window primarily focuses on illustrating the sequence of operations that a system performs when a page fault occurs.

Figure 8: Load page when the physical address is not present in TLB.
3.3. PAGING ALGORITHMS:

There are many algorithms available for page replacement. Out of them, this Virtuo-ITS tool visualizes only the most important ones like First in First out (FIFO), Least Frequently Used(LFU), Most Frequently Used(MFU), and Optimal. In these windows, the user has options to choose either the learning mode or practice mode. Learning mode teaches the user about the functioning of the algorithm, while practice mode assigns a certain problem to the user which he has to solve by using the page replacement strategy that he has learned.

3.3.1. FIFO Page replacement algorithm:

The FIFO tab is partitioned into two panels. One panel is for reenactment of the page replacement algorithm, and the other panel is for providing an exercise to the user. Both the panels are designed in the same way to abstain from confusing the user. In the learning mode, the user can choose the number of pages that can fit into main memory and then input the page sequence of his preference for which he wants to see the reenactment of the FIFO page replacement algorithm. Subsequently inputting the required parameters and clicking GO will generate the pages in virtual memory section according to the input. Blue color pages signify that they are present in the virtual memory. Green color pages signify that they are present in main memory. When the algorithm moves the pages into the main memory, then they change the color to green, indicating that the page is in use in memory. Later with each click of NEXT button, the system propels a step further in the simulation of the algorithm. The whole of this process also appears in word format in the panel to the right side of the window. Figure 9 shows
the visual representation of the learning mode of FIFO page replacement algorithm with all the color scheme described above.

Figure 9: FIFO page replacement window in learning mode.

The steps of how the user interacts with the system in learning mode for the page replacement algorithm are as follows:
1. The user selects FIFO page replacement algorithm in learning mode.

2. He chooses the number of pages that can fit into memory.

3. He enters the page sequence of his preference for which he wishes to see the FIFO page replacement algorithm.

4. The pages are generated in the virtual memory according to the input from the user.

5. Every time he clicks on NEXT button the page replacement algorithm executes on the given sequence, and subsequently, the pages are moved into memory.

6. All of this appears in word format in the blue panel on the right side.

7. If the required page is already present in memory, then there is no need for the page replacement.

Later, the user can test his acquaintance of the FIFO page replacement algorithm in practice mode. The design of panel in this mode is almost indistinguishable from the one in learning mode. The color scheme is likewise precisely analogous to the one in learning mode. Here when the user clicks on START, the system will provide the user with the parameters required for exercising page replacement algorithm. The user has to exercise the page replacement algorithm appropriately for the given constraints. When he makes an error, then the system will alert the user to it and direct him toward the accurate answer while explaining the concepts again. The system will alert the user when he makes an error with a message in the right panel colored in a red color font. The steps of how the user interacts with the system in practice mode are:

1. When the user clicks on START, the Virtuo-ITS will present the user with a page sequence and the other required parameters for FIFO page replacement.
2. The user has to accomplish the FIFO page replacement algorithm by clicking on the appropriate page that has to be present in memory.

3. When the user clicks on the correct page, then the page moves into memory. A prompt in blue color font appears, indicating the user that he clicked on the right page, as shown in Figure 10.

4. When the user clicks on an incorrect page, he will be asked to correct the error, and a prompt showing correct answer will be displayed. Figure 10 represents this process.

Figure 10: FIFO page replacement window in practice mode.
3.3.2. Optimal Page Replacement Algorithm:

The OPTIMAL tab is again partitioned into two panels. One panel is for reenactment of the page replacement algorithm, and the other panel is for providing an exercise to the user. Both the panels are designed in the same way to keep from confusing the user. The user can choose the number of pages that can fit into main memory, and then input the page sequence of his preference for which he wants to see the reenactment of the optimal page replacement algorithm and click GO. This will generate the pages in the virtual memory section in blue color. These pages change color to green when stacked in memory. Later with each click of the NEXT button, the system steps further in the simulation of the algorithm. The whole of this process also appears in text format in the panel to the right side of the window. Figure 11 shows the visual representation of the learning mode of OPTIMAL page replacement algorithm with all the color scheme described above.

The steps of how the user interacts with the system in learning mode for the page replacement algorithm are as follows:

1. The user selects optimal page replacement algorithms in learning mode.
2. He chooses the number of pages that can fit into memory.
3. He enters the page sequence of his preference for which he wishes to see the optimal page replacement algorithm.
4. The pages are generated in the virtual memory according to the input from the user.
5. Every time he clicks on NEXT button the page replacement algorithm executes on the given sequence, and subsequently, the pages are moved into memory.
6. All of this is depicted in word format in the blue panel on the right side.

7. If the required page is already present in memory, then there is no need for the page replacement.

![Optimal page replacement window in learning mode.](image)

Later the user can test his acquaintance of the optimal page replacement algorithm in practice mode. The design of panel in this mode is almost indistinguishable from the one in learning mode. The color scheme is likewise precisely analogous to the one in learning mode. Here when the user clicks on START, Virtuo-ITS will provide the user
with the parameters required for exercising the page replacement algorithm. The user has to exercise the page replacement algorithm appropriately for the given constraints. When he accomplishes something erroneous, then the system will alert the user to it and pilot him in the direction of the accurate answer while elucidating the concepts again. The system will alert the user when he fouls up by delineating a text in the right panel with a red color font.

Figure 12: Optimal page replacement window in practice window.
The steps of how the user interacts with the system in practice mode are:

1. When the user clicks on START, the Virtuo-ITS will present the user with a page sequence and the other required parameters for optimal page replacement.

2. The user has to accomplish the optimal page replacement algorithm by clicking on the appropriate page that has to be present in memory.

3. When the user clicks on the correct page, then the page moves into memory. A prompt in blue color font appears, indicating the user that he clicked on the right page, as shown in Figure 12.

4. When the user clicks on an incorrect page, he will be asked to correct the error, and a prompt showing correct answer will be displayed. Figure 12 is an illustration of the described process.

3.3.3. LFU Page Replacement Algorithm:

The LFU tab is again partitioned into two panels. One panel is for reenactment of the page replacement algorithm, and the other panel is for providing an exercise to the user. Both the panels are designed in the same way to abstain from confusing the user. The user can choose the number of pages that can fit into main memory, and then input the page sequence of his preference for which he wants to see the reenactment of LFU page replacement algorithm and click GO. This will generate the pages in the virtual memory section in blue. These pages change color to green when stacked in memory. Later with each click of the NEXT button, the system propels a step further in the simulation of the algorithm. The whole of this process also appears in word format in the panel to the right.
side of the window. Figure 13 shows the visual representation of the learning mode of LFU page replacement algorithm with all the color scheme described above.

![LFU page replacement window in learning mode](image)

Figure 13: LFU page replacement window in learning mode.

The steps of how the user interacts with the system in learning mode for the page replacement algorithm are as follows:

1. The user selects LFU page replacement algorithms in learning mode.
2. He chooses the number of pages that can fit into memory.
3. He enters the page sequence of his preference for which he wishes to see the LFU page replacement algorithm.

4. The pages are generated in the virtual memory according to the input from the user.

5. Every time he clicks on NEXT button, the page replacement algorithm executes on the given sequence, and subsequently, the pages are moved into memory.

6. All of this appears in word format in the blue panel on the right side.

7. If the required page is already present in memory, then there is no need for the page replacement.

Later the user can test his acquaintance of the LFU page replacement algorithm in practice mode. The design of the panel in this mode almost indistinguishable from the one in learning mode. The color scheme is precisely analogous to the one in learning mode. Here when the user clicks on START, Virtuo-ITS will provide the user with the parameters required for exercising page replacement algorithm. The user has to exercise the page replacement algorithm appropriately for the given constraints. When he makes an error, then the system will make an alert to guide him in the direction of the correct answer while teaching the concepts again. The system will alert the user when he makes an error with red text in the right panel.

The steps of how the user interacts with the system in practice mode are:

1. When the user clicks on START, the Virtuo-ITS will present the user with a page sequence and the other required parameters for LFU page replacement.

2. The user has to accomplish the LFU page replacement algorithm by clicking on the appropriate page that has to be present in memory.
3. When the user clicks on right page, then the page moves into memory. Also, a prompt in blue color font appears, indicating the user that he clicked on the right page, as shown in Figure 14.

4. When the user clicks on an incorrect page, he will be asked to correct the error, and a prompt showing correct answer will be displayed. Figure 14 is an illustration of the described process.

Figure 14: LFU page replacement window in practice mode.
3.3.4. MFU Page Replacement Algorithm:

The MFU tab is again partitioned into two panels. One panel is for reenactment of the page replacement algorithm, and the other panel is for providing an exercise to the user. Both the panels are designed in the same way to abstain from confusing the user. The user can choose the number of pages that can fit into main memory, and then input the page sequence of his preference for which he wants to see the reenactment of MFU page replacement algorithm and click GO. This will generate the pages in the virtual memory section in blue. These pages change color to green when stacked in memory. Later with each click of the NEXT button, the system makes a step further in the simulation of the algorithm. The whole of this process also appears in text format in the panel to the right side of the window. Figure 15 shows the visual representation of the learning mode of MFU page replacement algorithm with the color scheme described above.

The steps of how the user interacts with the system in learning mode for the page replacement algorithm are as follows:

1. The user selects MFU page replacement algorithms in learning mode.
2. He chooses the number of pages that can fit into memory.
3. He enters the page sequence of his preference for which he wishes to see the MFU page replacement algorithm.
4. The pages are generated in the virtual memory according to the input from the user.
5. Every time he clicks on NEXT button, the page replacement algorithm executes on the given sequence, and subsequently, the pages are moved into memory.
6. All of this appears in word format in the blue panel on the right side.
7. If the required page is already present in memory, then there is no need for the page replacement.

![Figure 15: MFU page replacement window in learning mode.](image)

Later the user can test his acquaintance of the MFU page replacement algorithm in practice mode. The design of panel in this mode is almost indistinguishable from the one in learning mode. The color scheme is analogous to the one in learning mode. Here when the user clicks on START, Virtuo-ITS will provide the user with the parameters required for exercising page replacement algorithm. The user has to exercise the page replacement algorithm appropriately for the given constraints. When he makes an error, the system
will alert him so he can correct it and it explains the concepts again. The system will alert the user when he makes an error by delineating a text in the right panel with a red color font (Figure 16).

Figure 16: MFU page replacement algorithm in practice mode.

The steps of how the user interacts with the system in practice mode are:

1. When the user clicks on START, the Virtuo-ITS provides the user with a page sequence and the other required parameters for MFU page replacement.
2. The user has to accomplish the MFU page replacement algorithm by clicking on the appropriate page that has to be present in memory.

3. When the user clicks on the correct page, then the page moves into memory. Also, a prompt in blue color font appears, indicating the user that he clicked on the right page, as shown in Figure 17.

4. When the user clicks on an incorrect page, he will be asked to correct the error, and a prompt showing correct answer will be displayed. This above figure is an illustration of the described process.

3.4. **Virtual to Physical Address Translation:**

The translation from virtual address to physical address is one of the important aspects of virtual memory management of an operating system. This window demonstrates the translation of a 32-bit virtual address to physical address. This visualizes how changes in bits will change the way the memory location is accessed. Later in the practice mode, the developed system will provide the student with a virtual address, and the student has to translate it to the physical address. The prompts will appear in the blue panel according to his interaction with the system.
The virtual address to physical address translation window has two modes. In learning mode, the user will be shown a virtual address represented in a blue dialog box and a green dialog box displays the physical address. Whereas, the page table used for address conversion is colored orange. A virtual address is presented, and it is converted to a physical address in an orderly fashion. With each click of NEXT button, the simulation steps forward. Arrows are animated to represent the appropriate functions that are happening. All of the processes that are happening will also be shown in text format in the panel towards the right end of the window (Figure 18).
The ITS systems discussed visualizes different steps involved in converting a virtual address to physical address in the learning mode. The design of practice mode window is identical to the learning mode. All the elements retain their color from practice mode. This helps the students to easily correlate between them. In practice mode, this tool provides the user with a virtual address and page table entries. Using the provided values, he has to convert the given virtual address to a physical address, and input it into the box provided for the physical address. When the user tries to verify his answer, then the system will validate his answer and will provide the result to him/her. If the user

Figure 18: Virtual to physical address translation window in practice mode.
incorrectly interprets the answer, then he/she will be alerted to it and will be guided towards the correct answer. All of the processes that are happening also appear in text format in the panel towards the right end of the window. The above figure is a demonstration of the described process.

3.5. Nested Page Tables:

All the contemporary operating systems use two-level page tables. The purpose of this window is to visualize the process of nested paging. Different colors are used to depict page tables in the main memory. Each color page table corresponds to a different process. The orange color table represents first-level page table, and the table will contain the addresses of the page tables for each process. With the click of NEXT button, the system will visualize how the page tables get accessed for the address conversion purpose. The appropriate arrows will be shown corresponding to the process page table's color. All of this also appears in word format in the panel towards the right end of the window. Figure 19 shows the design and working of nested page tables window of the Virtuo-ITS.
3.6. Real-Time Analysis:

In this window, the ITS system gets integrated with the real-time operating system. This window allows three major operations:

1. The user can start a single process, and when the user is finished with the process, the tool generates an output file that displays the resources used by the process. It shows the page in and page outs of that process. It also shows how much time the process is running and what amount of processing capacity the process has utilized. This
research considered Firefox process for the purpose of demonstration. Figure 20 depicts the working of the described function.

Figure 20: Single process real-time analysis.

The above figure depicts how a process is opened when the user clicks on the open button. Firefox will be opened, and the user can browse the internet for some time, and when the user is finished with the process and exits the process, then the file containing the statistics of that process will be generated. The file will have the details about the process CPU usage, number of major page faults it generated, number of minor page faults it generated, and some other process relevant
information. When the user clicks on the open file button, then the generated file will be opened. Figure 21 represents a view of the file.

![Command output](image)

Figure 21: View of the stats file generated for a single process.

2. The second feature that the tool incorporates into this window is that the user can now view the paging statistics of the whole system for a specific period of time. These statistics get updated in every one-second interval. The user can click on the view statistics of whole system button, and the data is recorded for every second and the tool logs the data into a file. When the user clicks on the view file generated button, the tool opens the statistics file produced. The file will contain information on the number of faults generated by the entire system during the time interval. Figure 22 depicts the generated statistics file in the time interval of 30 seconds for the system:
3. The third feature that the tool incorporates into this window is that the user can view the change in memory stats of the system while he is using the process. The update interval here is set to five seconds. For every five seconds, the change in memory stats will be displayed to the user. The user can now open two different processes, one with a heavy system usage and another process can be a lightweight process. He can compare how these two process will cause page faults and the rate at which they do it. By clicking on the button, the system will open up a terminal on which the stats of the memory are displayed. These will be updated every five seconds. Figure 23 depicts how this takes place on the system.
Figure 23: Memory stats of the whole system updated every five seconds.
4. ANALYSIS

Visualizations have been used in computer science courses long since the 1980's. During the course of time, a group of best practices has evolved through instructors experience with technology (Dann et al., 2003). Visualization in the field of academics is related to many disciplines, including the procedure of processing data and printing from it (typography), psychology, etc. This makes it arduous to summarize the edifications learned, albeit some broad suggestions about typography and format apply. Some of the commonly accepted suggestions are: (Dann et al., 2003)

The visualization should be able to provide a learner with resources that help them in understanding the visualization. The representation should be good enough to help the learner in understanding the algorithms. The learner may still find it difficult to understand the visualization. The relationship between the components of visualization requires clear explanation by embedding it in text format or by narrating it. The ITS system discussed in this thesis does this by embedding the text in the panel to the right side of the window.

The interactive visualization should be able to adapt to the user's knowledge level. Beginners have a challenging task to grasp the many details or understand all the windows. Conversely, experienced learners may benefit from the extra features like the ability to change the complexity, and ability to have their desired input data set. The ITS system discussed in this thesis does this by alerting the user when he does something wrong and by guiding the user towards the right answer.
The visualization should include performance information. The analysis of the efficiency of the algorithm is a consequential part of understanding the algorithm. So, the visualization should include the performance data of the algorithm. Another way of including the performance of the algorithm is by animating several related algorithms at the same time. The ITS system discussed in this thesis has this ability. Since page faults are the most important data in the performance analysis of the page replacement algorithms, it shows the number of page faults that occurs for each algorithm. It shows this information for all the four algorithms.

The visualization should include the execution history. It is normal for the user to forget the previous steps in the execution, or the user might have incorrectly understood the previous execution step. Having a history of the steps that the system executed will help the user rectify the mistakes. The ITS system discussed has this feature incorporated into it. The ITS system will show the execution history and animation history of each and every step. It will have a designated panel to the right of the window, which shows the execution history.

The visualization should have flexibility in the execution. It should be able to animate the process both forward and in reverse. The ITS system discussed in this thesis has that ability. It can run the animations in both forward and reverse directions in virtual to physical address translation, nested paging, and background windows. The user can click on the next button to advance the animation forward, and similarly, when the user clicks on the back button it traces the backward animation. The execution pauses after each step. This enables the user to check progress at each and every step and continue only after they fully understand what happened earlier.
The visualization should be able to support the custom input data from the user. Having this feature will allow the learner to observe the performance of the algorithm on the provided dataset. This will enable the learner to gain much insight into the working of the algorithm. The ITS system discussed in this thesis has this feature. The user can give his input data set for the FIFO, LRU, MFU, and Optimal page replacement algorithms and observe how the paging happens from in/out of memory. The learner has a display box to view the number of page faults, which is considered to be a performance metric for the page replacement algorithms.

The visualization should support dynamic questions. The users can periodically choose to complete an exercise to test their comprehension of the topic. This will allow the user to learn the faults and rectify the mistakes. The ITS system discussed in this thesis has this ability. There is an exercise for each of the page replacement algorithms. When the user clicks on the start exercise button, the system will provide him/her with a randomly generated page sequence. The user has to perform the page replacement strategy in the provided sequence correctly.

The visualization should also support dynamic feedback. The users should receive a feedback of their activities. This will enable the users to make the necessary corrections at the required stage. Needless to say, this will definitely increase the comprehension of the topic for the user. The ITS system discussed in this thesis has this feature. While performing the exercise, if a user does something wrong, then he will be prompted about the mistake and will be guided towards the right answer.

The visualization should complement the animations with explanations. The learner can better understand the visualization when the animations are supported with
explanations either by providing a text format display or by a background audio file. The ITS system discussed in this thesis has this ability incorporated into it. The panel towards the right side of the window will display the text messages relating to the animation.

Engagement Taxonomy as described by Dann et al. (2003) helps us understand the involvement of a learner in educational situations where tutoring happens through visualization of concepts. There are six defined categories of learner's engagement. This chapter briefly discusses how these engagement categories relate to our ITS system. The six defined categories are as follows:

1. No Viewing
2. Viewing
3. Responding
4. Changing
5. Constructing
6. Presenting

No Viewing: Any intricate concept can be explained using only theory. Therefore, the first category in the engagement taxonomy is No Viewing category. All of the concepts that we intend to teach using the ITS system discussed in this thesis has visualizations involved in them. Thus, this category will not be applicable to our system.

Viewing: viewing is the core form of engagement since every visualization contains some form of viewing (Dann et al., 2003). It is possibly the form of the engagement where we can find many different variations. It is the most passive form of
engagement. All other categories of engagement involve viewing (Dann et al., 2003). The Virtuo-ITS system has different windows that visualize different concepts of virtual memory in an operating system. Viewing type of engagement is present at each stage of the ITS system.

Responding: Responding is the third category in engagement taxonomy. The primary activity in this category is that the users should be able to answers the questions relating to the visualizations they are interacting with. This tool also supports the responding feature. The user has to interact with the system to progress the visualization. The visualization which involves the responding type of engagement will have more educational benefit than the visualization that uses only viewing as the form of engagement (Dann et al., 2003).

Changing: Changing is the fourth category in engagement taxonomy. The key activity in this category of engagement is that the user should be able to change the visualization by providing different input to it. This will enable the user to study the behavior of an algorithm in different scenarios. The ITS system developed has this ability. The user can provide custom input to the page replacement algorithms, thus modifying the visualization. The visualization which involves changing type of engagement will have more educational benefit than the visualization that uses responding as the form of engagement (Dann et al., 2003).

Constructing: Constructing is the fifth category in engagement taxonomy. The main activity in this form of engagement is that the users have to create their own visualization. The ITS system developed entails this type of engagement. The user can specify an input data set and based on the provided data the visualization will be
modified. It is important to note that constructing form of engagement may or may not include coding the algorithm. The visualization which involves constructing type of engagement will have more educational benefit than the visualization that uses changing as the form of engagement (Dann et al., 2003).

Presenting: Presenting is the fifth category in engagement taxonomy. The primary activity in this form of engagement is that the users are presented with an exercise, which they have to complete. The ITS system developed entails this feature. The users will be provided with exercise for the page replacement algorithms, and also for address conversion. The users will have an option to view the real-time statistics while using the system. They can analyze those statistics for a better understanding of their computers. The visualization which involves presenting type of engagement will have more educational benefit than the visualization that uses constructing as the form of engagement (Dann et al., 2003).

A general taxonomy is defined to estimate learner's level of comprehension. It is called Blooms Taxonomy (Huitt, 2011), and the primary idea of this taxonomy is to arrange what educators want students to know in hierarchical levels ranging from less complex to highly complex. The six levels of learner’s understanding defined in blooms taxonomy are as follows:

1. Knowledge level
2. Comprehension level
3. Application level
4. Analysis level
5. Synthesis level
6. Evaluation level

For the purpose of providing examples, the topic of virtual memory is considered. 

**Knowledge level:** In this level, the learner merely recalls the information in the form which they learned. There is no understanding involved at this level. For example, the learner may be asked to define the page replacement algorithms.

**Comprehension level:** In this level the learner understands the information which he/she studied. For example, the learner may be asked to explain the need for page replacement algorithm.

**Application level:** In this level, the learner can apply the concepts that he/she has learned to solve a given problem. For example, the user may be given a problem relating to FIFO page replacement algorithm and be asked to solve it using the knowledge he gained.

**Analysis level:** In this level, the learner can analyze the information and be able to differentiate between different types of information. For example, the user should be able to tell the differences among different page replacement algorithms.

**Synthesis level:** In this level, the student should be able to draw a conclusion from the information he learned, and also be able to design new solutions. For example, the learner should be able to perform the page replacement strategy on the whole given sequence. In the real-time mode of the ITS system designed, the user should be able to draw a conclusion on why there will be a decrease in number of page faults when the same process is killed and immediately opened.
**Evaluation level:** In this level, the learner should be able to compare different ideas by assessing the reasons and values behind those ideas. The learner should be able to decide on an idea based on reasoned arguments (Dann et al., 2003). For example, the learner should be able to evaluate the performance of the page replacement algorithms and decide which is the best procedure to apply to the given strategy.

The ITS system is assessed and evaluated based on student engagement and learning support. To evaluate the use of the ITS system in an educational setting, it is assessed against Engagement taxonomy and Bloom's taxonomy. The Engagement taxonomy has six defined categories of learner engagement with visualization. The Bloom's taxonomy has six hierarchical levels, which define a student's depth of understanding of concepts. The abilities and functionalities of the ITS system are mapped to five of the six categories in Engagement taxonomy, and also they are mapped to the six hierarchical levels of Bloom's taxonomy. The No Viewing category of Engagement taxonomy does not apply to this tool because the ITS system has visualization involved in it.

Table1 represents the mapping of the abilities of the ITS system to Engagement taxonomy. As shown in the table the ITS system involves all levels of engagement taxonomy. The ITS system has visualization involved, and it also requires the learner to interact with it to respond, change and construct the visualizations. It also presents the user with the real-time data from the system. The ITS system has all of the categories of engagement taxonomy in it.
Table 1: Mapping of ITS abilities to Engagement Taxonomy.

<table>
<thead>
<tr>
<th>ENGAGEMENT TAXONOMY</th>
<th>✓</th>
</tr>
</thead>
<tbody>
<tr>
<td>Viewing</td>
<td>✓</td>
</tr>
<tr>
<td>Responding</td>
<td>✓</td>
</tr>
<tr>
<td>Changing</td>
<td>✓</td>
</tr>
<tr>
<td>Constructing</td>
<td>✓</td>
</tr>
<tr>
<td>Presenting</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 2 represents the mapping of the ITS abilities onto Bloom's taxonomy. From Table 2, one can observe that the ITS system involves the learner in the evaluation level, which is the highest level of learning according to Bloom's taxonomy. The ITS system will present the user with the knowledge of the concepts, and the learner has to understand it and apply the knowledge to solve the problem and synthesize a solution for it. Then the solution will be evaluated, and the learner will get to know the correct solution. From this, it infers that the ITS system has all levels of learning of Bloom's taxonomy involved in it.

Table 2: Mapping of ITS abilities to Bloom’s taxonomy.

<table>
<thead>
<tr>
<th>BLOOM'S TAXONOMY</th>
<th>✓</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knowledge</td>
<td>✓</td>
</tr>
<tr>
<td>Comprehension</td>
<td>✓</td>
</tr>
<tr>
<td>Application</td>
<td>✓</td>
</tr>
<tr>
<td>Analysis</td>
<td>✓</td>
</tr>
<tr>
<td>Synthesis</td>
<td>✓</td>
</tr>
<tr>
<td>Evaluation</td>
<td>✓</td>
</tr>
</tbody>
</table>

When a user runs the Virtuo-ITS system, it shows him/her a screen where the instructions to use this tool are present. After the user reads the instructions and clicks on continue, the tool presents him/her with the designed background window. The user will have the option to navigate to any window at any time and continue the execution from the point where he/she left.
The background window of this tool shows the visualization relating to the process that happens when a page fault occurs. The user can select the required parameters and click on NEXT button. The tool then shows the visualizations based on the selected specifications. This satisfies the viewing and responding category of engagement taxonomy. This window does not entail the changing, constructing and presenting categories of engagement taxonomy.

The page replacement algorithm windows of this tool show the visualizations relating to process of page replacements in the main memory according to the selected page replacement algorithm. The user can provide an input sequence to the Virtuo-ITS system and the tool will construct the visualization corresponding to it based on the specified input. In practice mode, the tool presents the user with a randomly generated page sequence, and the user has to construct the correct page replacement algorithm. This window satisfies the viewing, responding, changing, constructing and presenting categories of engagement taxonomy.

The virtual-to-physical address translation window of this tool show the visualization corresponding to the address translation. Virtuo-ITS tool considers a virtual address and translates it to physical address, and shows the visualization corresponding to it. The tool provides the user with a virtual address and asks the user to convert it to a physical address using the provided page table. This window satisfies viewing, responding and changing categories of engagement taxonomy.

The real-time mode window of this tool shows the data obtained from the running operating system. The user can open the browser, and use it for some time and
close it. The tool shows the user the generated file which contains the data recorded from the system. This window satisfies viewing, responding, changing and presenting categories of the engagement taxonomy. Table 3 shows the categories of engagement taxonomy satisfied by each window of the Virtuo-ITS system.

Table 3: Mapping of categories of Engagement taxonomy on to each window of Virtuo-ITS

<table>
<thead>
<tr>
<th>Categories of Engagement Taxonomy</th>
<th>Viewing</th>
<th>Responding</th>
<th>Changing</th>
<th>Constructing</th>
<th>Presenting</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Window</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Background</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FIFO</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Optimal</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>LFU</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MFU</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Virtual-to-Physical address translation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Real-time</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

The following paragraphs discuss the ways in which the Virtuo-ITS system is better than the existing methods for teaching the concepts of virtual memory in operating systems.

The ITS system is unlike the skeleton OS, where the user has to understand the kernel code and program the remainder parts of the operating system. This task proves to
be challenging for a student who has little knowledge about the assembly level language and programming. The developed ITS system does not require the learner to understand complicated code, but rather focuses on teaching the concepts to the learners based on visualization.

The ITS system also does not involve making modifications to the existing OS code. For a learner, to do this task entirely by the knowledge they gained from textual reading could be difficult. The ITS system focuses on teaching the concepts to the user with clear and interactive visualizations along with an added feature to modify them as per the algorithm logic.

The Virtuo-ITS focuses on educating the user by means of animating the concepts and using visualizations, which is unlike Nachos, Minix, GeekOS, OS/161. Minix, Nachos, GeekOS, OS/161 try to simulate the original operating system. The amount of detail these systems present could overwhelm a novice learner quickly. All of these are only simulators, and they do not endeavor in tutoring the concepts. All of these simulators focus on high-level functionality but do not provide anything in the area of low-level functionalities. Virtuo-ITS focuses on tutoring one major concept in great detail. It elucidates even low-level functionalities of each concept.

Most of the simulators that are available do not have a user-friendly interface which makes it difficult for students to learn how to use them. There are only a few tutoring systems developed for virtual memory of an OS. Even the few available virtual memory tutoring systems focus only on paging algorithms and leave out crucial functionalities like virtual-to-physical address translations. Virtuo-ITS has a user-friendly
interface outlined using Java, furthermore, has modules for many concepts which are left out in other systems.

The Virtuo-ITS system improves upon many other OS training systems in the aspects as stated above. The ITS system will be a useful tool for educators to teach the concepts of the operating system in an undergraduate class.
5. CONCLUSIONS

5.1. Summary:

The purpose of this thesis is to present a working prototype of an interactive tutoring system that demonstrates the key concepts of virtual memory in an operating system. The developed tool successfully addressed the hypothetical research problems like the challenges faced by computer science students in learning the concepts of virtual memory, and the problems with existing tutoring tools for teaching the virtual memory concepts. These research problems and software requirements for teaching an OS course in an undergraduate computer science curriculum provided a motivation to perform this research work.

The initial chapters (1-3) briefly introduced the problem statement, and outlined the requirements obtained from the tutoring needs of the course. Further, a literature review addressed the existing teaching methods and OS simulators for teaching undergraduate operating systems course. The design architecture of this tool considered the drawback in existing tutoring methods and the functional requirements defined in these chapters to develop a strategy in building this software. This involved developing features like a practice mode for users. The development of this software made use of the Swing and AWT libraries from Java. The use case diagram in Chapter 3 provided the end-to-end flow of the user interaction with the system across all modules. It described how Virtuo-ITS is unlike skeleton OS as it focuses on explaining the concepts to the student using visualizations rather than having the student program parts of the OS.
Chapter 4 presented the logical analysis of user’s level of understanding using benchmark rules like Bloom’s taxonomy. As seen from Table 1, this software implements the highest level of involvement (Presenting) of the user as per Engagement taxonomy. Similarly, results from Table 2 shows that Virtuo-ITS engages learners in the highest level of user understanding (Evaluation) as per Bloom’s taxonomy.

5.2. Future Work:

In recent years with the advent of 3-dimensional (3D) libraries in computer programming languages provides us an option to expand the visualizations of the virtual memory concepts and algorithms into 3D representations. Existing software frameworks like Unity 3D and Java 3D can provide potential solutions to build the interactive tutoring systems. This research becomes complete when one compares the usefulness of the 2-dimensional and 3-dimensional representations for these visualized concepts.
*Intelligence, 11*(3), 15-16.


*Proceedings of the National Educational Computing Conference, Baltimore, MD*, 129-134.


