An Address Translation Simulator

Steven Robbins
Department of Computer Science
University of Texas at San Antonio
srobbins@cs.utsa.edu

ABSTRACT

Virtual memory is a major topic in undergraduate operating systems courses. One aspect of virtual memory, address translation, is often covered in an abstract way. When examples are given, only a piece of the translation is done, using a small translation lookaside buffer or a small single-level page table. Since most students learn best by doing rather than watching, the topic is best understood by having students do realistic address translations. This is problematic since it involves lookup from several large tables of data which are difficult to fit on a piece of paper. The address translation simulator described here solves this problem by presenting the student with complete page tables in a way that allows simple navigation of these tables. The simulator can be used for both teaching and student evaluation.

Categories and Subject Descriptors

K.3 [**Computers & Education**]: Computer & Information Science Education—*Computer Science Education*

General Terms

Virtual memory, address translation

Keywords

operating systems

1. INTRODUCTION

Virtual memory and address translation are standard topics in an undergraduate operating systems course. Although address translation in modern computers is mostly done by hardware, the topic of address translation is usually taught in the undergraduate operating systems course because that is where virtual memory is typically taught. Operating systems courses that delve into the details of the operating system by having students write (or modify) an operating system are usually good at giving students hands-on experience with the details of operating system issues. Since

it is primarily a hardware topic, address translation is an exception to this.

Although students learn best by doing rather than watching, most presentations of address translation are abstract and cover only the simplest of address translation problems, mainly because of the nature of address translation. Translation requires access to a large amount of data: a translation lookaside buffer and at least one page table. Since this information does not easily fit on a sheet of paper, instructors find it difficult to design problems that students can do. Also, hand manipulation of binary address strings up to 64 bits in length is prone to errors. Even a hexadecimal representation of a 64-bit address is daunting, and hexadecimal is often not suitable because the sizes of bit fields are not always multiples of 4.

This paper discusses a simulator that makes it simple (and fun) to manipulate address bit fields and allows students to practice address translation using a translation lookaside buffer (TLB) and single or 2-level page tables. The simulator can also be used for evaluation of student skills by keeping a log file in HTML format that can be displayed with a standard browser or printed. The simulator is written in Java and will run on any computer system having a Java runtime environment. It is available for download or can be run directly from the web without the need for any installation.

The address translation simulator is part of a suite of simulators that have been developed for the undergraduate operating system curriculum [5]. These simulators have a consistent look and feel so that students who have used one can easily master another.

2. ADDRESS TRANSLATION IN MODERN TEXTBOOKS

Topics in address translation that are typically addressed in modern operating systems textbooks [1, 2, 3, 6, 7, 8] include:

> partitions
> single-level page tables
> multiple-level page tables
> segmentation
> segmentation with paging
> inverted page tables

This paper concerns itself with address translation using a TLB with one and two level page tables. Experience has shown that students do not grasp this concept until they have done problems involving concrete examples. It is particularly difficult to produce problems of this type and some

page table

valid	frame
0	5
1	6
1	4
1	7
0	8
1	9
1	3
. . .	

TLB

page	frame
5	9
3	7
2	4
6	3

Figure 1: A small TLB and the beginning of a page table.

Page Table Levels	2
Page Size	4096
Level 2 Page Table Size	512
Virtual Address Bits	30
Physical Address Bits	23
TLB Entries	16
Page Table Width (bytes)	4

Figure 2: Parameters for the simulator

textbooks do not even try. The exercises involving address translation in some books [2, 3, 6] deal only with questions concerning the location or number of bits in various fields of the virtual address or the size of page tables. Some books do not even have exercises on this topic [1] while others [7, 8] have exercises involving a small (6-16 entries) single-level page table. None of the standard operating systems textbooks have any exercises combining TLB lookup with page table lookup or using multilevel page tables.

It is useful to have students actually perform address translation in some simple cases. I have found that about half of the students in my undergraduate operating systems courses cannot do this without practice, indicating that they really do not understand the process. Address translation requires looking up an entry in a TLB and then possibly looking through one or more page tables. Just displaying a typical TLB requires a lot of space. A full page table will not easily fit on a piece of paper. For the past decade, I have been using a toy problem based on the TLB and page table shown in Figure 1. The page size is 256 bytes. I give my students a few addresses such as the ones below and ask them to determine the page number and the physical address or if a page fault occurs. I also combine this with an access time calculation.

    a) 01010100010
    b) 10001000101
    c) 00110110100

While this type of problem is useful, it is still much too small to test understanding. A similar problem with a 2-level page table would be unmanageable on paper, even if only the page tables needed to do the problem are given. Since part of the problem is to determine which second-level page table to use, giving only the needed page tables defeats the purpose of the exercise. The Address Translation Simulator allows students to experiment with more realistic problems in address translation.

3. GOALS OF THE ADDRESS TRANSLATION SIMULATOR

The goals of the address translation simulator are twofold: practice and testing. In practice mode, help is available for each stage of the translation. In testing mode, help can also be available, but each action of the student is recorded. In either mode, the simulator displays a logical address along with the parameters of the system. Sample parameters are shown in Figure 2.

There are several levels of practice mode available depending on the skill level of the user. Help menus are available for all objects displayed. At the lowest level of skill, a progress list like the one shown in Figure 3 shows the steps the user must take to complete the address translation. As steps are completed, a colored dot appears at the start of the item, indicating that it has been completed. Clicking on any of these items produces a dynamically generated help menu describing in detail what needs to be done. Two possible menus obtained by clicking on the first unchecked item in Figure 3 are shown in Figure 4. Since the menus are dynamically generated, the information shown depends on the current state of the address translation. The upper menu is used if the TLB has not yet been displayed. Clicking on the **do it!** line displays the TLB and changes the menu accordingly. The lower menu corresponds to the case in which the TLB has been displayed but the user has incorrectly determined the sizes of the page number and frame number fields. As the students become more skilled, the progress list and help menus can be removed, requiring the students to know which steps must be completed.


Figure 3: A progress list for a single-level page table.


Figure 4: Two possible help menus for the 4th entry of Figure 3.

In testing mode there are several options as to how much help the student can get, starting with complete hints as

to what to do next as in practice mode up to no indication as to correct procedure until the physical address is found. Testing mode can be set up with a fixed number of *lifelines* in which the user can ask for help a certain number of times.

In testing mode all actions of the user are logged, and written logs can be produced with varying degrees of detail. At the highest level of detail all actions by the user are reported. At the lowest level only the number of correct and incorrect answers are displayed.

The log can be automatically sent to the instructor (or to another account) by email so that the log cannot be modified by the student. Alternatively, the log can be saved to a file in HTML format so that it can be displayed by a browser or printed.

The simulator can be used to enhance and test the following skills necessary to the understanding of address translation:

> Identifying the fields of a logical address
> Identifying the fields of a physical address
> Determining the sizes of the fields in the TLB
> Looking up a page number in a TLB
> Calculating the address of a page table entry
> Looking up an entry in a page table
> Using the valid bit

4. LOOK AND FEEL

The simulator has the same look and feel as other simulators developed for the operating systems curriculum [5]. Three new general purpose widgets have been added: a clipboard, a segmented string and a calculator.

Binary fields are manipulated in the simulator using a familiar copy and paste paradigm. Copying a string representing a binary number moves it into a clipboard. There is only one clipboard, but the contents are displayed in each widget that uses it. A Calculator widget, shown in Figure 5 allows binary numbers to be manipulated in simple ways: add, subtract, multiply, divide and shift. Numbers are manipulated by copying them into the clipboard (the **Select** operation), pasting them (**Paste**) into the **x** or **y** fields of the calculator, performing an operation, and copying the result back into the clipboard (**Select** again).
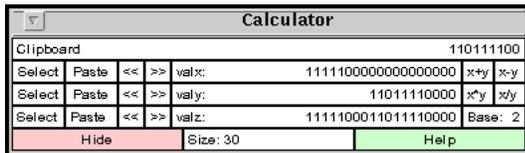
Figure 5: A calculator widget.

An example of a simple operation using the clipboard is the lookup of a page in the TLB. To do this, the user places the appropriate bit field of the logical address in the clipboard. The same clipboard is shared by the TLB, so pushing the **Lookup** button on the TLB locates the corresponding entry. This is explained in more detail in the next section.

Some operations such as the calculation of the memory address of page table entry require the calculator. The start address of the page table is put in the clipboard. The value will then be shown in the clipboard of the calculator (top line of Figure 5). Pushing the **Paste** button of the **x** value

of the calculator (second line of Figure 5) sets the value of **x** in the calculator. Next, select the page number in the logical address and similarly paste it in the **y** field of the calculator. If the page table entries are 4 bytes each, shift the page number left twice by pushing the shift left button (<<) for the **y** value. The add button (**x+y**) puts the sum in the **z** value of the calculator, and the corresponding **Select** button puts the result in the clipboard, making it available to the memory module.

The **Help** button of the calculator displays general information about using the calculator. Depending on the level of help selected, it can display additional information when appropriate, such as *Do not forget to shift the page number to the left.*

5. SINGLE PAGE TABLE

In this section we describe using the address translation simulator in practice mode with a TLB and single-level page table. The user is presented with an address translation problem in a window like the one shown in Figure 6.
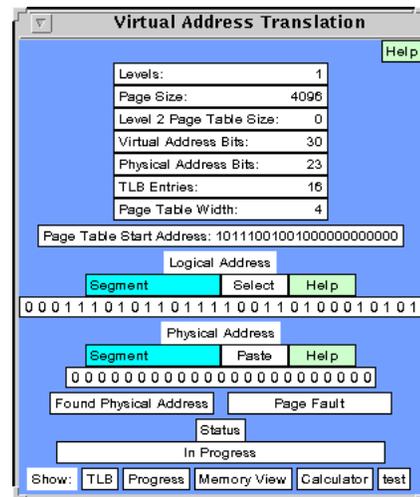

Figure 6: An address translation problem using a single-level page table.

The page size is 4K bytes, the virtual address is 30 bits, and the physical address is 23 bits. Page table entries contain 4 bytes and the TLB has 16 entries. The starting address of the page table is shown along with the logical address of 000111010110111100110100010101. Below the logical address is a place for the user to fill in the corresponding physical address. The buttons below the physical address are for the user to report either having found a physical address or that a page fault will occur. Below this is the status of the physical address determination. When one of the above buttons is pushed, it either reports that the corresponding value was correct or incorrect. At the bottom of the window are some buttons for popping up various windows.

The steps required to solve this problem are shown in Figure 3. Since a single-level page table is being used, the logical and physical addresses each have two parts, an offset and a page or frame number. The first step tested by the simulator is the determination of the boundary between the offset

and the rest of the address. Both the Logical Address and Physical Address widgets are shown in Figure 6 and start in **Segment** mode. In this mode an address is segmented by clicking the mouse in the appropriate place on the address. If done correctly, the step is indicated in the progress list. The logical and physical addresses are displayed in a segmented string widget that allows the string shown (in this case a binary number) to be divided into parts. In this case there is only one dividing line, and if students make a mistake, they can just click again in another place.

Once the students have segmented the logical and physical addresses, they can change the logical address widget to **Select** mode and the physical address to **Paste** mode. Now, clicking on one of the fields of the logical address puts that field into the clipboard. Clicking on a field of the physical address pastes a copy of the clipboard into that field. Only two clicks are needed to copy the offset from the logical address to the physical address.

The next step is the lookup in the TLB shown in Figure 7. First the student segments the TLB into fields containing page and frame numbers. The process is similar to the segmenting of the logical or physical address, except that one click segments all of the entries in the TLB. After segmenting the TLB, students click on the **Lookup** button to look up the page number contained in the clipboard. If the simulator finds a corresponding entry, it highlights the frame number and puts the frame number in the clipboard. Alternatively, if students see a matching page number in the TLB, they can just click on the corresponding frame number after setting the TLB to be in **Select** mode. In either case, if the page is found in the TLB, students can put the corresponding frame in the physical address by clicking on the appropriate field in the physical address. This action pastes the frame address into the physical address from the clipboard.

Figure 7: The TLB after a successful lookup.

If the page is not in the TLB, students must examine the page table. The simulator provides several views of memory. One view is the page table view. For single-level page tables the simulator displays page table entries (with valid bit) for the entire page table as shown in Figure 8. When there is only one page table, students do not have to set the address

of the start of the table. Students can either scroll down to the needed entry (tedious if the page table is large) or can put the entry number in the clipboard and click the **Entry** button on the memory display. A lookup is done by putting the page number into clipboard and clicking the **Entry** button.

Figure 8: The single-level page table view of memory after a successful lookup.

6. TWO-LEVEL PAGE TABLE

There are several ways of implementing 2-level page tables. For this simulator, the top level page table entry contains a valid bit and the frame number of the start of the corresponding second-level page table. The second-level page tables contain valid bits and frame numbers. A page fault can occur at either level. The TLB entries, if used, contain the combined first and second-level page numbers and a frame number. If an entry is present in the TLB, the frame number is directly available and no page table lookups are needed. In this case the procedure is the same as for a single-level page table.

If the combined page number is not in the TLB, students must segment the logical address again so that it is divided into three parts. The high order field is the first-level page number. Students select this to use as an index into the first-level page table. If the corresponding entry is invalid, a page fault occurs. Otherwise a second-level page table lookup is necessary. Figure 9a shows the result of a successful first-level lookup. After the entry is found, the frame number of the start of the second-level page table is in the clipboard. Students now switch to the second-level page table view. In this view students need to indicate the starting frame number of the page table. Since the starting frame number is in the clipboard from the previous step, students should click the **Frame** button. After selecting the second-level page number from the logical address, do an **Entry** lookup in the second-level page table. Figure 9b shows the result of this. The frame number is 1111011.

The address translation simulator assumes that page tables start on frame boundaries, which is almost always the case. When multi-level page tables are used, page tables are often one frame in size. In the first-level page table of Figure 9a, as in the single page table case, the start address is fixed (there is only one such table per process) and each line gives a table entry labeled with the entry number. In the second-level page table shown in Figure 9b, the starting frame number of the page table must be set by the user. Students can do this either by putting the frame number in the clipboard and pushing the **Frame** button or by putting the address of the desired entry in the clipboard and pushing the **Addr** button. In either page table at most 16 entries are displayed on the screen and the students can scroll through the entire page table if they desire.

a)   b)
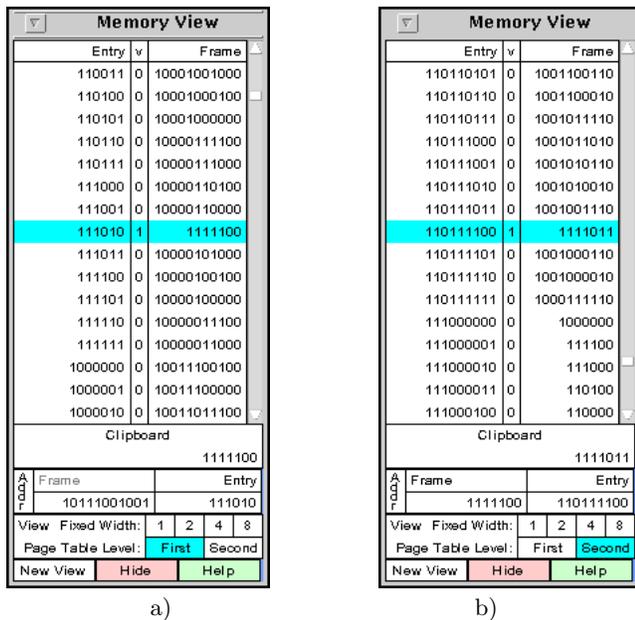
Figure 9: The first-level page table view of memory (left) after a successful lookup and the corresponding second-level page table (right).

Figures 8 and 9 show page table views of memory. In these views, data is arranged in blocks that correspond to the width of the page table and valid bits are shown. Students can also display a frame of physical memory in a more generic way. Figure 10 shows a frame of memory with a data width of 4 bytes. The valid bits are not available from this memory view. A separate valid bit view may be used to determine whether a given page number is valid. This separate view is more appropriate for those machines that store the valid bits in the MMU rather than in the page tables.

7. CONCLUSIONS

The address translation simulator can be used in three ways: for demonstration, for practice and for evaluation. Instructors can perform in-class demonstrations of how address translation is done. After students have seen the demonstration, they can use the simulator to practice performing address translations. The practice problems can be run with varying levels of help, from guiding the students at each step
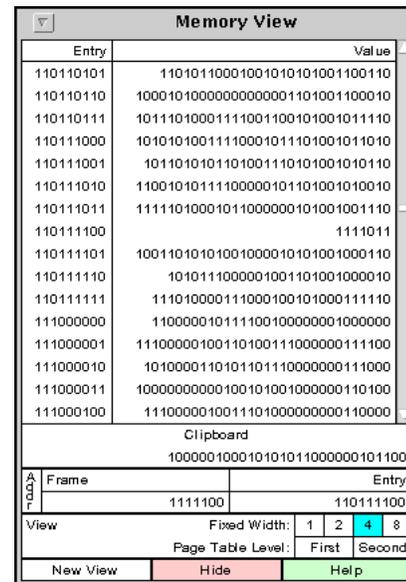
Figure 10: The view of memory arranged in 4-byte words.

to just providing hints if a student gets stuck. A user's manual is available on line.

Lastly, the simulator can be used for testing. The simulator can generate a log for the instructor showing in a settable detail how the student did. Usually, instructors only need to test for completion. That is, instructors should ignore student runs in which errors were made and just check to see that at least one run of each type was computed without error.

The simulator can be run directly from a browser [4] without the need of any installation. If customization is required, the simulator can be downloaded and installed on any host that has a modern Java virtual machine.

8. ACKNOWLEDGMENTS

9. REFERENCES

[1] W. S. Davis and T. M. Rajkumar, *Operating Systems, A Systematic View,* Addison Wesley, 2005.
[2] H. M. Deitel, P. J. Deitel and D. R. Choffnes, *Operating Systems, Third Edition,* Prentice Hall, 2003.
[3] G. Nutt, *Operating Systems, Third Edition,* Addison-Wesley, 2003.
[4] S. Robbins, The Address Translation Simulator, 2004. Online. Internet. Available WWW: **http://vip.cs.utsa.edu/nsf/address/run/**
[5] S. Robbins, Simulators for teaching operating systems, 2003. Online. Internet. Available WWW: **http://vip.cs.utsa.edu/nsf/simulators**
[6] A. Silberschatz, P. B. Galvin and G. Gagne,, *Operating System Concepts, Sixth Edition,* John Wiley and Sons, Inc, 2002.
[7] W. Stallings, *Operating Systems, Fifth Edition,* Prentice Hall, 2004.
[8] A. Tanenbaum, *Modern Operating Systems, Second Edition,* Prentice Hall, 2001.