EXPERIMENT 4

VIRTUAL MEMORY

1. OBJECTIVES

After completing this lab, you will be able to:

- Understand why TLBs (Translation Look-aside Buffer) can provide fast address translation.
- Know how address translation is done by using IPT (Inverted Page Table).
- Understand how page replacement is essential to virtual memory and how to implement a least recently used replacement algorithm.

2. LABORATORY

Software Lab 1 (N4-01a-02)
Software Lab 2 (N4-01c-06)
SPL (N4-B1b-11)

3. EQUIPMENT

Pentium IV PC with Nachos 3.4.

4. MODE OF WORKING

You should be working alone. No group effort.

5. PROBLEM STATEMENT

In this lab, you are required to complete a virtual memory implementation, including how to get a physical frame for a virtual page from the IPT if it exists there, how to put a physical frame/virtual page entry into TLB, and how to implement a least recently used page replacement algorithm.

A software-managed TLB is implemented in Nachos. There is one TLB per machine. There is also an IPT which maps physical frames to virtual pages. Basically, the translation process first examines the TLB to see if there is a match. If so, the matching entry in the TLB will be used for address translation. If there is a miss, the IPT will be looked up. If a matching entry is found in the IPT, the entry will be used to update the TLB. A miss in the IPT means that the page will have to be loaded from disk, and a page in and page out will be performed.

To decide which page to page out, a page replacement policy is used, for example, a least recently used algorithm which will be explained late on. During each lookup process, you need to perform some checking in order to make sure that you are looking up the correct entry and that the entry is valid.

In order to check whether you are referencing the correct entries from the TLB, you have to check the valid bit. The TLB will get updated when an exception is raised and the required page entry isn't in it. In this case, a new entry needs to be inserted into the TLB. The new entry will be inserted into an invalid entry in the TLB or replace an existing entry if it is full. Since the TLB is small, the replacement policy for the TLB is simply FIFO. When there is a context switch between processes, e.g. the main process executing a child process, the entries in the TLB will be cleared by setting all entries to invalid.

The IPT is simply implemented using an array, represented by the `memoryTable` (a mapping of what pages are in memory and their properties). There is one entry for each of the physical frame, and each entry contains the corresponding process id, virtual page number, and the last used field that records the tick value when the page was last accessed. The least recently used algorithms works by iterating through the `memoryTable`, from the beginning, to look for the entry that has been least recently used. If there is an entry that is not valid (i.e., its process is dead), the algorithm will return the index of this invalid entry. Otherwise, the algorithm will return the index of the least recently used entry (that is, the entry with the smallest last used field).

6. EXERCISES

1. Change working directory to Experiment 4 by typing ***cd ~/nachos-exp3-4/vm***.

2. Modify the following methods in file `tlb.cc`. You may need to reference the following files: `tlb.h`, `ipt.h`, `ipt.cc`, `machine/translate.h`, `machine/translate.cc`, `machine/machine.h`, and `machine/disk.h`.

    a. `int VpnToPhyPage(int vpn)` - Gets a physical frame `phyPage` for a virtual page `vpn`, if exists in the IPT.

    b. `void InsertToTLB(int vpn, int phyPage)` - Insert a vpn/phyPage entry into the TLB.

    c. `int lruAlgorithm(void)` – Return the freed physical frame according to the least recently used algorithm.

3. Compile Nachos by typing ***make***. If you see "***ln -sf arch/intel-i386-linux/bin/nachos nachos***" at the end of the compiling output, your compilation is successful. If you encounter any anomalies, type ***make clean*** to remove all object and executable files and then type ***make*** again for a clean compilation.

4. Execute the test program to test whether the virtual memory is working properly by typing ***./nachos -x ../test/vmtest.noff -d > output.txt***. (**Do not cut and paste this command to your termina**l.)

5. Fill the table (examples are shown below) whenever there is a TLB miss (i.e., the matching entry is not found in the TLB, but it could be found in the IPT) or an IPT miss (i.e., the requested page is not in the memory at all; it must be a page fault and must trigger page replacement). Fill as many rows as necessary until Nachos exits.

    The first three columns are the tick that the page fault exception occurred (`tick`), the virtual page number (`vpn`) and the corresponding process id (`pid`). Each process has its own set of virtual page numbers. The next four columns represent each entry in the IPT. Each IPT entry has four values, the process id (`pid`), virtual page number (`vpn`), last accessed tick (last used) and valid flag. The next three columns represent each entry in the TLB. Each TLB entry shows the virtual page number (`vpn`), physical frame number (`phy`), and the valid flag of the entry. The last column records the dirty page that is paged out, if any. You should complete **Table1.csv** file (template is provided) and record down the entries of the IPT and TLB **before** replacement. Please also save given the template as a **pdf** file and **highlight** the entry that is selected to be updated. **Do not highlight**

cells in the csv file and also **do not add any new column header to Table1.csv**.

The first four rows of the table are shown below (any row which does not have a page out is filled with **N**).

tick	vpn	pid	IPT[0]	IPT[1]	IPT[2]	IPT[3]	TLB[0]	TLB[1]	TLB[2]	Page Out
			pid, vpn, last used, valid				vpn, phy, valid			
10	0	0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0,0	0,0,0	0,0,0	0,0,0	N
13	9	0	0,0,12,1	0,0,0,0	0,0,0,0	0,0,0,0	0,0,1	0,0,0	0,0,0	N
15	26	0	0,0,12,1	0,9,15,1	0,0,0,0	0,0,0,0	0,0,1	9,1,1	0,0,0	N
20	1	0	0,0,12,1	0,9,19,1	0,26,17,1	0,0,0,0	0,0,1	9,1,1	26,2,1	N

6. Complete **Table2.csv** file (template is provided) to record page size, the number of physical frames, and the TLB size defined in Nachos as well as the number of pages used by the test program and the number of TLB misses, page faults and page outs that occurred during the execution of the test program.

7. ASSESSMENT

- Assessment of your implementation. Please leave your code, the output file **output.txt** as well as **Table1.csv**, **Table1.pdf** and **Table2.csv** files containing the above information in the **vm** folder for TA/Supervisor to review. Deadline is 1 week after your lab session (e.g., if lab session is from 10AM-12PM on a Monday, then deadline is 9:59AM on the next Monday).
- **Lab Quiz 2**, which is an online multiple-choice quiz, will be administered through NTULearn during week 14.