|  |
| --- |
| Nanyang Technological University School of Computer Engineering |
| CE2005 Operating Systems |
| **EXPERIMENT 3 Report** |
|  |
|  |
| |  | | --- | | **Name: Lu Shengliang** | | **Group: SE2** | | **Martic: U1220521C** | | **Date: 3/29/2014** | |

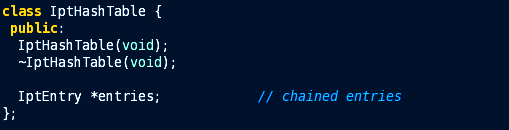
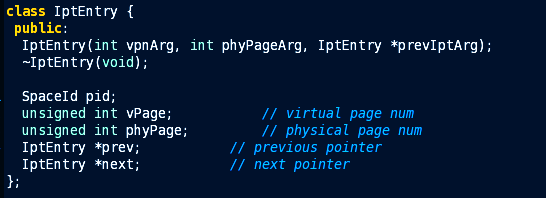
## Introduction

In this lab, it is required to complete a virtual memory implementation, including:

1. Get a physical frame for a logical page from the IPT(Inverted Page Table)

The use of IPT saves memory space and reduces the number of page tables, since many processes will share a single table instead of having their own page tables.

IPT is constructed by a linked collection of IptEntry, which is a hashed table (hashIPT).



1. Put a physical frame/logical page entry into TLB(Translation lookaside buffer)

Similarly, the use of TLB reduces the memory access time, as TLB is a special fast-lookup hardware cache. In this lab, a software-managed TLB is occupied to cope with virtual to physical address translation.

TLB is constructed by an array of TranslationEntry, with size fixed as TLBSize, which is declared in machine.h.



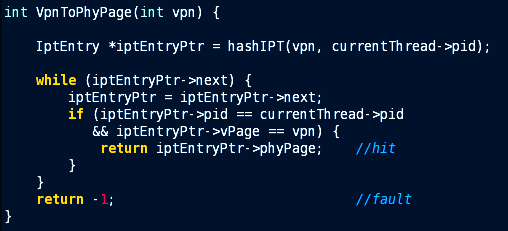
1. Implement a clock page replacement algorithm.

Clock algorithm is an implementation of the Second-Chance algorithm. There is a pointer pointing to the oldest page. When a page fault occurs, currently pointed page will be inspected. If this page is satisfied with a certain criteria, the page will be paged out. Otherwise, a second chance will be provided to it, and it will be cleared to initial state (no more chances). The clock pointer will advance to next page.

In all, clock algorithm is a more efficient version of FIFO than Second-chance because pages don't have to be constantly pushed to the back of the list, but it performs the same general function as Second-Chance. The code will be shown later.

## Method Implementation

#### int VpnToPhyPage(int vpn)

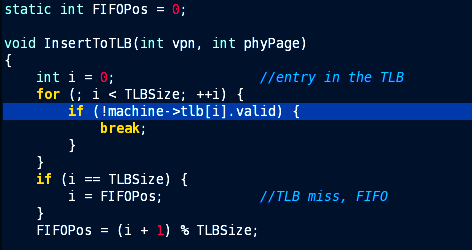
We let iptEntryPtr point to its mapped location in hashIPT. Then the pointer will iterate all elements chained in the linked list. If current iptEntry has the same pid as currentThread, and this entry’s vPage is same as current vpn, then page hit. Otherwise, page fault occurs. 

#### void InsertToTLB(int vpn, int phyPage)

A static FIFO position pointer is occupied in order to show which the oldest entry in TLB is.

We search the TLB from entry 0 until the last entry (*TLBSize*). If the entry is valid, which means the entry is already allocated to others, we will ignore it and continue search.

If we go through the whole TLB, and there is no empty entry, we will put i as the entry who will be evicted now.

Then, we advance FIFO position pointer in a circular manner to ensure it is always pointer at the oldest entry. 

#### int clockAlgorithm(void)

The description of clock algorithm method is shown above. We will settle the replacement criteria here.

1. The entry the clock is on is not valid

* (!memoryTable[clockPos].valid)

1. The entry has indeed been in memory too long without being referenced

* ((!memoryTable[clockPos].dirty) && (memoryTable[clockPos].clockCounter

>= OLD\_ENOUGH))

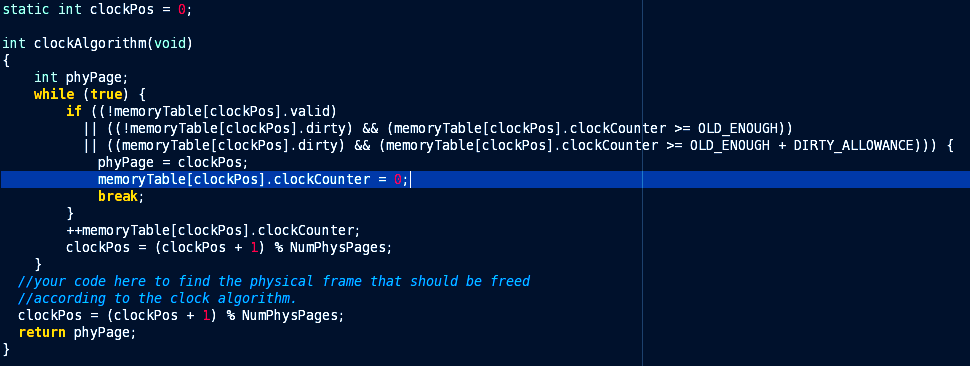
1. the position has been dirtied (changed), and this entry has indeed been in memory too long without being referenced

* ((memoryTable[clockPos].dirty) &&

(memoryTable[clockPos].clockCounter

>= OLD\_ENOUGH + DIRTY\_ALLOWANCE)))

If current entry satisfied any one or more criteria shown above, it is supposed to be evicted. Hence, this function will note the clock position pointer value done, advance the pointer in a circular manner, and return the phyPage value. Before returning, we should set clockCounter back to 0, since this entry is just referenced.

Otherwise, current entry is not referenced, so it will age. And we will advance clock position pointer to check other entries in the memory table.

## Result Analysis

#### Data collection

Since the debug information is hard to analyze, either because too much information are showed or we are not familiar with the format of debug messages.

So we can add printf in several functions to show what we want in our preferred format.

#### Analysis on Page Faults

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sqe | VPN | TLB Miss | Page Fault | Page In | Page Out | Phypage | TLB Entry | Dirty |
| 1 | 0 | Yes | Yes | 0 | - | 0 | 0 | 0 |
| 2 | 9 | Yes | Yes | 1 | - | 1 | 1 | 0 |
| 3 | 27 | Yes | Yes | 2 | - | 2 | 2 | 1 |
| 4 | 1 | Yes | Yes | 3 | - | 3 | 0 | 0 |
| 5 | 0 | Yes | No | - | - | 0 | 1 | 0 |
| 6 | 11 | Yes | Yes | 0 | 0 | 0 | 2 | 0 |
| 7 | 0 | Yes | Yes | 1 | 1 | 1 | 0 | 0 |
| 8 | 9 | Yes | Yes | 3 | 3 | 3 | 1 | 0 |
| 9 | 27 | Yes | No | - | - | 2 | 2 | 1 |
| 10 | 10 | Yes | Yes | 0 | 0 | 0 | 0 | 0 |
| 11 | 0 | Yes | No | - | - | 1 | 1 | 0 |
| 12 | 11 | Yes | Yes | 3 | 3 | 3 | 2 | 0 |
| 13 | 9 | Yes | Yes | 2 | 2 | 2 | 0 | 0 |
| 14 | 27 | Yes | Yes | 1 | 1 | 1 | 1 | 0 |
| 15 | 10 | Yes | No | - | - | 0 | 2 | 0 |
| 16 | 0 | Yes | Yes | 3 | 3 | 3 | 0 | 0 |

As we can see, there are 16 TLB miss, and 12 page faults.

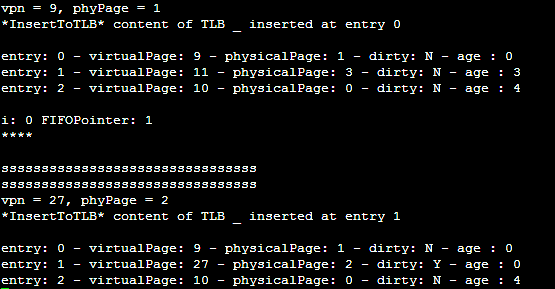
The clock algorithm obviously works at line 8. If we use simple FIFO implementation, page 2 should be Page out and VPN 9 should be put at page 2. But since VPN 27, which is at page 2(line 3), is dirty and not old enough now, clock algorithm leaves it and find page 3 to replace it.



But when it comes to line 13, VPN 9 replaces VPN 27 because *clockCounter* for VPN 27 already exceed the limitation as OLD\_ENOUGH + DIRTY\_ALLOWANCE.

On the contrast, if we set *OLD\_ENOUGH* smaller and *DIRTY\_ALLOWANCE* greater, as shown below, VPN 27 will not be evicted at line 13. VPN 9 will be allocated at Page 1 instead.





## Summary

Different environment configuration values will lead to different results, when the same algorithm is occupied. There is no best solution for chasing good performances when dealing with different data. But via comparison between different algorithm and different configuration values, we can find better solution for a certain situation.

TLB and applying clever algorithm will enhance the performance of virtual memory by reducing the checking time in *memoryTable* or hard disk.