Yecheng Wang – 301540271

CMPT300 Operating Systems I

Assignment 4 – Part 1 Virtual Memory Questions

**Question 1:** A certain computer provides its users with a **virtual-memory space of 2^32 bytes**. The computer has **2^18 bytes of physical memory**. The virtual memory is implemented by **paging**, and the **page size is 4096 bytes**. A user process generates the **virtual address 11123456** (this is in hexadecimal (base 16)). Explain how the system establishes the corresponding physical location. Distinguish between software and hardware operations. Feel free to use a diagram (simple ASCII is fine) if you wish, but that’s not required.

Virtual Memory space = 2^32 bytes = 32 bits to represent

Physical memory space = 2^18 bytes = 18 bits to represent

Page size is 4096 bytes = 2^12 bytes = 12 bits to represent

We have 12 bits of the virtual address determines the offset within the page since page size needs 12 bits to represent, and the remaining bits represent the virtual page number (VPN).

The binary representation of the virtual address `11123456` (hexadecimal) is `00010001000100100011**010001010110**`. Breaking this down:

-VPN: 00010001000100100011

- Offset: **010001010110**

First, we convert the hexadecimal virtual address to binary. Then, we used the page size to determine that we need the lowest 12 bits as the page offset.

Software:

The software operations involve the managing the page tables, including updating the page table when the pages are loaded or evicted, as well as handling page faults. In this next step, the software operating system uses the upper VPN bits to index the page table to find the corresponding physical page number (PPN). If the target page is not currently in the cache, we encounter a page fault, and the operating system must pick a page in a physical frame to evict for the page to load into.

Hardware:

We then move on to the hardware aspects that involve the actual translation of the virtual address to a physical address. The hardware uses the PPN obtained from the page table from the software operations. Now the hardware performs an operation to combine the PPN and the original offset from the virtual address. This forms the physical address.

**Question 2:** Assume we have a **demand-paged memory**. The page table is held in registers. It takes **8 milliseconds to service a page fault** if an empty page is available or if the replaced page is not modified, but **20 milliseconds if the replaced page was modified**. Memory access time is **100 nanoseconds**. Assume that the page to be replaced is **modified 70% of the time**. What is the maximum acceptable page-fault rate for an effective access time of **no more than 200 nanoseconds**? Show your work.

### Part 2: Calculating Maximum Acceptable Page-Fault Rate

For a demand-paged memory with the given parameters:

- Service time for a page fault when an empty page is available or the replaced page is not modified: 8 milliseconds

- Service time for a page fault when the replaced page is modified: 20 milliseconds

- Memory access time: 100 nanoseconds

- The page to be replaced is modified 70% of the time

The effective access time (EAT) is calculated as follows:

EAT= {(1 - Page Fault Rate) \* Memory Access Time} + {Page Fault Rate} \* (Page Fault Overhead)

Where Page Fault Overhead is the weighted average of the service times:

Page Fault Overhead = 0.7 \* 20ms + 0.3 \* 8ms

To find the maximum acceptable page-fault rate for an EAT of no more than 200 nanoseconds, we'll plug in the values and solve for the page-fault rate:

200 = (1−Page Fault Rate) ∗ 100 + Page Fault Rate ∗ (Page Fault Overhead in ns)

Let's calculate the Page Fault Overhead in nanoseconds (note: 1 ms = 1,000,000 ns) and then solve for the Page Fault Rate.

The Page Fault Overhead, considering the weighted average of service times for page faults, is 16,400,000 nanoseconds (or 16.4 milliseconds).

For an effective access time (EAT) of no more than 200 nanoseconds, the maximum acceptable page-fault rate is approximately 6.1×10^(−6) (or 0.0000061). This means that to achieve the target EAT, the system can tolerate a page fault once in every roughly 164,000 memory accesses.

This calculation demonstrates how even a small increase in the page-fault rate can significantly affect the system's overall performance, especially when the cost of handling a page fault is high.