**CSCE 611 600: OPERATING SYSTEMS Homework #3**

**Name:** Rohan Chaudhury

**Email Address:** [rohan.chaudhury@tamu.edu](mailto:rohan.chaudhury@tamu.edu)

**UIN:** 432001358

**List of people I have worked with:** None

**Question 1:** (CSCE 611 – 5 points, CSCE 410 – 7.5 points)

**Consider an MMU with 16 4-KB pages. The incoming virtual address’s lower 12 bits is the offset and the upper 4 bits is the page table index. The middle column lists physical frame number.**

**Compute the** **sixteen-bit physical address (and convert to base 10) given the following page table and incoming virtual address:**

**HINT: Work in binary then covert to decimal.**

**Incoming virtual address: 12296 or 0011 0000 0000 1000**

**Page # Frame # present/absent**

|  |  |  |
| --- | --- | --- |
| **15** | **000** | **0** |
| **14** | **000** | **0** |
| **13** | **000** | **0** |
| **12** | **000** | **0** |
| **11** | **111** | **1** |
| **10** | **000** | **0** |
| **9** | **101** | **1** |
| **8** | **000** | **0** |
| **7** | **000** | **0** |
| **6** | **000** | **0** |
| **5** | **011** | **1** |
| **4** | **100** | **1** |
| **3** | **001** | **1** |
| **2** | **110** | **1** |
| **1** | **001** | **1** |
| **0** | **010** | **1** |

**ANSWER:**

The given incoming virtual address in binary is **0011 0000 0000 1000.** The upper 4 bits of this address is **0011** (**3** in decimal) which is the page table index and the lower 12 bits denoting the offset is **0000 0000 1000**. Corresponding to page table index 3, from the given table above, we can see that the Frame number is **001**. Using this frame number and the lower 12 bits denoting the offset we can obtain the sixteen-bit physical address.

Replacing the upper 4 bits in the virtual address with the frame number we get the sixteen-bit physical address as: **0001 0000 0000 1000 (in binary)**

Converting this to base 10 (decimal) we get the physical address as: **4104**

**Reference:** <https://stackoverflow.com/questions/40292822/translate-virtual-address-to-physical-address>

**Question 2:** (CSCE 611 – 5 points, CSCE 410 – 10 points) **A computer has four page frames. The time of loading, time of last access and the R(referenced) and M(modified) bits are shown below (the times are in clock ticks).**

# Page Loaded Last Ref. (ticks) R bit M bit

**0 126 280 0 0**

**1 230 265 0 1**

**2 140 270 0 0**

**3 110 285 1 0**

1. **Which page will Not Recently Used (NRU) replace?**
2. **Which page will Least Recently Used (LRU) replace?**
3. **Which page will second chance replace?**
4. **Which page is computationally the most expensive to replace?**

**ANSWER:**

1. NRU will replace any page randomly that doesn’t have the R (reference) and M (modified) bit set. So, either **Page 0** or **Page 2** will be replaced by NRU.
2. LRU will replace the page which has the earliest time of last access. Hence **Page 1** will be replaced by LRU.
3. The second chance will replace the page which was loaded the earliest but has its R bit set to 0. According to that **Page 0** will be replaced by second chance.
4. If a page is modified then the modifications have to be stored in memory before it is being replaced, which is computationally more expensive than replacing pages that are not modified. From the given table, we can see that only **Page 1** is modified and hence it will be most computationally expensive to replace.

Question 3: (CSCE 611 – 10 points, CSCE 410 – 12.5 points)

Given reference string R = 0 2 1 3 5 4 6 3 7 4 7 3 3 5 5 3 1 1 1 7 1 3 4 1

and memory M below; where M is has four page frames represented by the heavy outlined boxes. Pages (represented by their page numbers) are in memory if they appear in the upper heavy outlined boxes and have been paged out to disk if they are in the lower boxes.

R=0 2 1 3 5 4 6 3 7 4 7 3 3 5 5 3 1 1 1 7 1 3 4 1

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 2 | 1 | 3 | 5 | 4 | 6 | 3 | 7 | 4 | 7 | 3 | 3 | 5 | 5 | 3 | 1 | 1 | 1 | 7 | 1 | 3 | 4 | 1 |
|  |  | 0 | 2 | 1 | 3 | 5 | 4 | 6 | 3 | 7 | 4 | 7 | 7 | 3 | 3 | 5 | 3 | 3 | 3 | 1 | 7 | 1 | 3 | 4 |
|  |  |  | 0 | 2 | 1 | 3 | 5 | 4 | 6 | 3 | 3 | 4 | 4 | 7 | 7 | 7 | 5 | 5 | 5 | 3 | 3 | 7 | 1 | 3 |
|  |  |  |  | 0 | 2 | 1 | 3 | 5 | 4 | 6 | 6 | 6 | 6 | 4 | 4 | 4 | 7 | 7 | 7 | 5 | 5 | 5 | 7 | 7 |
|  |  |  |  |  | 0 | 2 | 1 | 1 | 5 | 5 | 5 | 5 | 5 | 6 | 6 | 6 | 4 | 4 | 4 | 4 | 4 | 4 | 5 | 5 |
|  |  |  |  |  |  | 0 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
|  |  |  |  |  |  |  | 0 | 0 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
|  |  |  |  |  |  |  |  |  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Answer the following questions:

a) How many page faults occur?

b) What paging algorithm is being used?

c) Is the paging algorithm using a stack algorithm? Why or why not?

d) Briefly describe Belady’s anomaly.

e) Does the paging algorithm in this problem exhibit Belady’s anomaly? Why or why not?

ANSWER:

1. The total number of page faults that have occurred is 11.
2. We can see that if a page hit occurs it is brought to the top of the stack and the least recently used page moves to the bottom of the stack. If a page fault occurs then the least recently used page which is at the bottom of the stack is replaced. Thus the paging algorithm used here is LRU (Least Recently Used).
3. Here the paging algorithm used is LRU which uses a stack.

“A stack-based algorithm is one for which it can be shown that the set of pages in memory for N frames is always a subset of the set of pages that would be in memory with N + 1 frames. For LRU replacement, the set of pages in memory would be the n most recently referenced pages. If the number of frames increases then these n pages will still be the most recently referenced and so, will still be in the memory.

In LRU algorithm every time a page is referenced it is moved at the top of the stack, so, the top n pages of the stack are the n most recently used pages. Even if the number of frames is incremented to n+1, top of the stack will have n+1 most recently used pages.”

We can see a similar kind of behavior in the example given here. Hence the paging algorithm here is using a stack algorithm.

Reference: <https://www.geeksforgeeks.org/beladys-anomaly-in-page-replacement-algorithms/>

1. “Generally, on increasing the number of frames to a process’ virtual memory, its execution becomes faster as fewer page faults occur. Sometimes the reverse happens, i.e. more page faults occur when more frames are allocated to a process. This most unexpected result is termed Belady’s Anomaly.

So, Bélády’s anomaly is the name given to the phenomenon where increasing the number of page frames results in an increase in the number of page faults for a given memory access pattern.”

Reference: <https://www.geeksforgeeks.org/beladys-anomaly-in-page-replacement-algorithms/>

1. “All the stack based algorithms never suffer Belady Anomaly because these type of algorithms assigns a priority to a page (for replacement) that is independent of the number of page frames. Examples of such policies are Optimal, LRU and LFU. Additionally these algorithms also have a good property for simulation, i.e. the miss (or hit) ratio can be computed for any number of page frames with a single pass through the reference string.”

Since LRU is a stack based algorithm that is used in the example given here, it does not exhibit Belady’s anomaly.

Reference: <https://www.geeksforgeeks.org/beladys-anomaly-in-page-replacement-algorithms/>