Question 1. [10 points] why is the protection of processes’ memory space important? Describe a scenario where absence of memory protection leads to problems.

**It helps keep user programs from crashing one another and the OS.**

**Memory being used can be reallocated by another program, which will create errors.**

Question 2. [20 points] **The following table shows the core map of a virtual memory system, which has a page size of 100.**

|  |  |  |
| --- | --- | --- |
| Frame number | Process ID | Page number |
| 0 | 1 | 2 |
| 1 | 1 | 1 |
| 2 | 2 | 1 |
| 3 | 3 | 0 |
| 4 | 1 | 3 |

1. To which physical address does virtual address 130 of process 1 map? If this virtual address does not map to any physical address, write ”does not map”.

Logical address = 130 = pg# (pg Size) + d = 1 (100) + d

THUS, **d = 30, pg# = 1, and so f# = 1**

**Therefore,**

**Physical address = f# (pg Size) + d = 1 (100) + 30 = 130**

1. To which physical address does virtual address 17 of process 2 map? If this virtual address does not map to any physical address, write ”does not map”.

Logical address = 17 = 0 (100) + d

THUS, **d = 17 and pg# = 0**

**Process 2 does not associate with page number 0, therefore**

**DOES NOT MAP**

1. Which virtual address of which process maps to physical address 50?

Physical address = 50 = f# (pg Size) + d = 0 (100) + d

THUS, **d = 50 and f# = 0**

**Therefore, PID = 1 and pg# = 2**

**Logical address = pg# (pg Size) + d = 2 (100) + 50 = 250**

**Process 1 maps to virtual address 250**

Question 3. [30 points] Consider a system where the virtual memory page size is 1KB (1024 bytes), and main memory consists of 4 page frames, which are empty initially. Now consider a process, which requires 8 pages of storage. At some point during its execution, the page table is as shown below:

|  |  |  |
| --- | --- | --- |
| Virtual page # | Physical page # | Valid Flag |
| 0 |  | No |
| 1 |  | No |
| 2 | 2 | Yes |
| 3 | 3 | Yes |
| 4 |  | No |
| 5 |  | No |
| 6 | 0 | Yes |
| 7 | 1 | Yes |

1. [10 points] List the virtual address ranges that will result in a page fault.

**{ [0, 2047], [4096, 6143] }**

**The virtual address range that would result in a page fault would be the set containing the ranges: 0 to 2047 and 4096 to 6143**

1. [20 points] Give the following **ordered** references to the virtual addresses (i) 4500, (ii) 8000, (iii) 3000, (iv) 1100, please calculate the main memory (physical) addresses. If there is a page fault, please use LRU based page replacement to replace the page. How which page will be affected and compute the physical addresses after the page fault. We assume the reference string is

2 4 7 3 0 4 3 0 7 5 0 7 6 0 2 3 6 4 7 6 3 2 before the new reference.

PG SIZE (N) = 1024 AND MAIN MEMORY CONSISTS OF 4 PAGE FRAMES

1. 4500, reference string 7632

4500 = pg# (N) + d = 4 (1024) + d, d = 404 and pg# = 4

Results in a page fault, page replacement needed and will be conducted on pg 7 by LRU. The frame # utilized will be 1.

Therefore,

**Physical address = f# (N) + d = 1 (1024) + 404 = 1428**

**The table now looks like this:**

|  |  |  |
| --- | --- | --- |
| Virtual page # | Physical page # | Valid Flag |
| 0 |  | No |
| 1 |  | No |
| 2 | 2 | Yes |
| 3 | 3 | Yes |
| 4 | **1** | **YES** |
| 5 |  | No |
| 6 | 0 | Yes |
| 7 | **\_\_\_\_\_\_\_\_\_\_\_\_** | **NO** |

1. 8000, reference string 6324

8000 = 7 (1024) + d, d = 832 and pg# = 7

Results in a page fault, page replacement needed and will be conducted on pg 6 by LRU. The frame # utilized will be 0.

Therefore,

**Physical address = f# (N) + d = 0 (1024) + 832 = 832**

**The table now looks like this:**

|  |  |  |
| --- | --- | --- |
| Virtual page # | Physical page # | Valid Flag |
| 0 |  | No |
| 1 |  | No |
| 2 | 2 | Yes |
| 3 | 3 | Yes |
| 4 | 1 | YES |
| 5 |  | No |
| 6 | **\_\_\_\_\_\_\_\_\_\_\_\_** | **NO** |
| 7 | **0** | **YES** |

1. 3000, reference string 3247

3000 = 2 (1024) + d, d = 952 and pg# = 2

There is no page fault because page 2 is valid. The frame number will be 2.

Therefore,

**Physical address = f# (N) + d = 2 (1024) + 952 = 3000**

**The table stays looking like this:**

|  |  |  |
| --- | --- | --- |
| Virtual page # | Physical page # | Valid Flag |
| 0 |  | No |
| 1 |  | No |
| 2 | 2 | Yes |
| 3 | 3 | Yes |
| 4 | **1** | **YES** |
| 5 |  | No |
| 6 | 0 | Yes |
| 7 | **\_\_\_\_\_\_\_\_\_\_\_\_** | **NO** |

1. 1100, reference string 2472, because duplicates remove first 2 to get 3472

1100 = 1 (1024) + d, d = 76 and pg# = 1

Results in a page fault, page replacement needed and will be conducted on pg 3 by LRU. The frame # utilized will be 3.

Therefore,

**Physical address = f# (N) + d = 3 (1024) + 76 = 3148**

**The table now looks like this:**

|  |  |  |
| --- | --- | --- |
| Virtual page # | Physical page # | Valid Flag |
| 0 |  | No |
| 1 | **3** | **YES** |
| 2 | 2 | Yes |
| 3 | **\_\_\_\_\_\_\_\_\_\_\_\_** | **NO** |
| 4 | 1 | Yes |
| 5 |  | No |
| 6 |  | No |
| 7 | 0 | Yes |

Question 4. [20 points] Given a computer system with the following paging based addressing for virtual addresses. Please answer the following questions:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 bits | 5 bits | 5 bits | 5 bits | 7 bits |

TOTAL BITS = 2 + 5 + 5 + 5 + 7 = **24 = m + n**

m = pTotal = 17 and n = d = 7

1. [5 points] What is the size of the virtual address space?

2^Total Bits = 2^24 = **16,777,216 bits**

1. [5 points] What is the page size?

2^n = 2^7 = **128 bits**

1. [5 points] What is the maximum number of pages for a process?

Top(Size/pg Size) = 2^24 / 2^7 = 2^(24-7) = **2^17 = 131072 pages**

1. [5 points] Assume the TLB access time “a”, memory access time “b”, and page fault processing time “c”, given the system has a TLB hit ratio of 99% and page fault rate of 1%. Please formulate the effective memory access time.

4b because of there being 4 slots for P#

EAT = 0.99(a + b) + (a +4b + b)(1 - 0.99) = 0.99a + 0.99b + 0.01(a + 5b)

= .99a +0.99b + 0.01a + 0.05b

= **a + 1.04b**

EAT = (1 – 0.01)b + 0.01c, **sub b with the previous EAT formula**

**EAT = 0.99(a + 1.04b) +0.01c**

Question 5. [20 points] Consider a system with 1MB of available memory and requests for 40KB, 398KB, 15KB, and 20KB. The system is using Buddy Allocation Algorithm.

**a)**. (15 points) Show the amount of memory allocated for each request and the state of memory after each request. Assume there is no memory release.

**b).** (5 points) Why does internal fragmentation occur with buddy allocation? How much internal fragmentation exists in this scenario?

**It happens with buddy allocation because the segmented memory will most likely not get fully utilized. Meaning that there is wasted memory.**

**Amount of Internal Fragmentation =** (64 – 40) + (512 - 398) + (16 - 15) +

(32 - 20)

= **151**

**c).** (5 points) Why does external fragmentation occur with buddy allocation? How much external fragmentation exists in this scenario?

**External Fragmentation occurs in buddy allocation because the available segments cannot accept or allow the requested memory because it is too large. This happens when the last of the available memory total is bigger than the request, but the individual segments of memory are smaller than the requested memory.**

**Amount of External Fragmentation** = 16 + 128 + 256

= **400**