**Reagan Leonard**

**Homework 6: Virtual Memory**

1. Consider a computer with a paged logical address space with 8 pages and each page is 4 Kbytes. The logical address space is mapped into a 256-Kbyte of physical memory space. (30pts)
   1. Draw the fields in the logical and physical addresses and show the number of bits of each field.

A close up of a piece of paper

Description automatically generated

A close up of text on a white background

Description automatically generated

* 1. Draw the page table of a process and show the number of entries in the table and number of bits per entry.

A close up of text on a white background

Description automatically generated

* 1. Populate the page table for process, namely A, which is currently running on the CPU. Several pages of process A is in the physical memory as follows:

|  |  |
| --- | --- |
| #frame 9 | Page 2 of Process A |
| #frame 10 | Page 5 of Process A |
| #frame 11 | Page 4 of Process A |
| #frame 12 | Page 0 of Process A |
| #frame 13 | Page 7 of Process A |
| #frame 14 | Page 3 of Process A |
| #frame 15 | Page 6 of Process A |
| #frame 16 | Page 1 of Process A |

1. Consider paged virtual memory systems. Assume a page size of 256 bytes (28), and that processes in this system can have a maximum virtual address space of 16K bytes (214). The system is currently configured with 8K (213) bytes of physical memory. (30pts)
   1. How many pages are in the virtual address space?

VM / page size = 214 / 28

= 26 = **64 pages**

* 1. How many page frames are in the physical address space?

Physical memory / page frame size

= 213 / 28

= 25 = **32 page frames**

* 1. A user process generates the virtual address 12,345 (0011000000111001 in binary). Explain how the system establishes the corresponding physical address assuming that the hardware memory management unit and transfer lookaside buffer (TLB) is used.

**What is the TLB:** The TLB is part of the memory-management unit (MMU). It’s a memory cache that is used to reduce the time taken to access a user memory location. It is associative, high speed memory. Each entry in the TLB consists of two parts: a key/tag and a value. When the associative memory is presented with an item, it gets compared with all keys at the same time. If the item is found, the corresponding value field is returned.

**How the TLB would help solve this problem:** The TLB contains only a few of the page table entries. When a virtual address gets generated by the CPU (like above), its corresponding page number is given to the TLB. If that page number is found, its frame number is immediately available and is used to access memory. If not, then we get what is called a TLB miss and a memory reference to the page table must be made. Once the frame number is retrieved, we can use it to access memory.

1. Consider a paged virtual memory system with a physical memory that can only contain 4 pages. Assume the execution of a program generates the following address trace

*a b c d d f b e b e*

where *a*, *b*, *c*, *d*, *e*, and *f* are the pages referenced and the page frames are initially empty. (40pts)

* 1. How many page faults occur with first-in-first-out Page Replacement?

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Time* | 1 2 | | 3 4 | | 5 6 | | 7 8 | | 9 10 | |
| *Request* | *a* | *b* | *c* | *d* | *d* | *f* | *b* | *e* | *b* | *e* |
|  | **a** | **a** | **a** | **a** | **a** | **b** | **b** | **c** | **d** | **d** |
|  |  | **b** | **b** | **b** | **b** | **c** | **c** | **d** | **f** | **f** |
|  |  |  | **c** | **c** | **c** | **d** | **d** | **f** | **e** | **e** |
|  |  |  |  | **d** | **d** | **f** | **f** | **e** | **b** | **b** |
| Fault? | **Yes** | **Yes** | **Yes** | **Yes** | **No** | **Yes** | **No** | **Yes** | **Yes** | **No** |

**# of page faults = 7**

* 1. How many page faults occur with LRU Page Replacement?

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Time* | 1 2 | | 3 4 | | 5 6 | | 7 8 | | 9 10 | |
| *Request* | *a* | *b* | *c* | *d* | *d* | *f* | *b* | *e* | *b* | *e* |
|  | **a** | **a** | **a** | **a** | **a** | **b** | **c** | **d** | **d** | **d** |
|  |  | **b** | **b** | **b** | **b** | **c** | **d** | **f** | **f** | **f** |
|  |  |  | **c** | **c** | **c** | **d** | **f** | **b** | **e** | **b** |
|  |  |  |  | **d** | **d** | **f** | **b** | **e** | **b** | **e** |
| Fault? | **Yes** | **Yes** | **Yes** | **Yes** | **No** | **Yes** | **No** | **Yes** | **No** | **No** |

**# of page faults = 6**