**Homework 6: Virtual Memory**

**Due date: 11:59PM Monday April 27**

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.

Logical address:

#logical Page 8 = 2^3 so it needs 3 bits

Offset 4Kb = 2^12 so it needs 12 bits

Physical address:

# page frames = 256K/4k = 64 = 2^6 so it needs 6 bits

Offset 4Kb = 2^12 so it needs 12 bits

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

|  |  |  |
| --- | --- | --- |
| Page Number | #Valid bit | # bit for frame# |
| 0 | 1 | 6 |
| 1 | 1 | 6 |
| 2 | 1 | 6 |
| 3 | 1 | 6 |
| 4 | 1 | 6 |
| 5 | 1 | 6 |
| 6 | 1 | 6 |
| 7 | 1 | 6 |

* 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 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 |
|  | … |

|  |  |  |
| --- | --- | --- |
| Page Number | Valid bit | frame# |
| 0 | 0b1 | 0b001100 = (12) |
| 1 | 0b0 |  |
| 2 | 0b0 |  |
| 3 | 0b0 |  |
| 4 | 0b1 | 0b001011 = (11) |
| 5 | 0b1 | 0b001010 = (10) |
| 6 | 0b0 |  |
| 7 | 0b1 | 0b001101 = (13) |

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?

# of pages = 16K/256 = 2^14 / 2^8 = 2^6 = 64

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

# of frames = 8K/256 = 2^13/ 2^8 = 2^5 = 32

* 1. a user process generates the virtual address 12,345 (0011 0000 0011 1001 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.

for the given address 0b0011 0000 0011 1001, the lowest 8 bits 0b0011 1001 indicate the page offset, and the highest 8 bits 0b0011 0000 indicate the page number.

each entry of TLb has multiple fields. fields related to this question include: valid bit, page# and mapped frame#.

The MMU (memory management unit) looks up in the TLb for an entry with the page number 0b00110000. If an entry is found and the valid bit is 1, the field of frame# indicates the physical page frame. The actual physical address is the concatenation of frame# and the page offset. Otherwise, MMU looks up the frame# in the process page table.

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?

A queue is used to keep track of the references of the four frames. The queue has a fixed size, which is 4.

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

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

A priority queue is used to keep track of the references of the four frames. The priority queue has a fixed size, which is 4.

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