Matt Ritchie

CS420

HW7

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. Explain how the system establishes the corresponding physical location.

In binary that virtual address is:

0001 0001 0001 0010 0011 0100 0101 0110

Since the page size is 2^12, the page table must be 32-12 = 2^20. So the low-order 12 bits (0100 0101 0110) are the displacement into the page while the rest of the bits (0001 0001 0001 0010 0011) are the displacement in the table.

2) When a page fault occurs, the process requesting the page must block while waiting for the page to be brought from disk into physical memory. Assume there exists a process with five user-level threads where the mapping of user threads to kernel threads is many-to-one. If one user thread incurs a page fault while accessing its stack, would the other user threads belonging to the same process also be affected by the page fault (i.e., would they also have to wait for the faulting page to be brought into memory?) Explain.

Yes.

There is only one kernel thread for all user threads, so that kernel thread blocks while waiting for the page fault to be resolved. Since there are no other kernel threads or available user threads, all other user threads in the process are affected by the page fault.

3) The table below is a page table for a system with 12-bit virtual and physical addresses with 256-byte pages. The list of free page frames is D, E, F (that is, D is at the head of the list, E is second, and F is last.)

+-------+------------+

| Page | Page Frame |

+-------+------------+

| 0 | - |

| 1 | 2 |

| 2 | C |

| 3 | A |

| 4 | - |

| 5 | 4 |

| 6 | 3 |

| 7 | - |

| 8 | B |

| 9 | 0 |

+-------+------------+

Given the following virtual addresses, convert them to their equivalent physical addresses in hexadecimal. All numbers are given in hexadecimal. (A dash for a page frame indicates the page is not in memory.)

a) 9EF -> 0EF

b) 111 -> 211

c) 700 -> D00

d) 0FF -> EFF

4) Consider the following page reference string:

1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6

How many page faults would occur for the following replacement algorithms, assuming one, two, three, four, five, six, or seven frames? Remember all frames are initially empty, so your first unique pages will all cost one fault each.

a) LRU replacement

b) FIFO replacement

c) Optimal replacement

|  |  |  |  |
| --- | --- | --- | --- |
| Frames | LRU replacement | FIFO replacement | Optimal replacement |
| 1 | 20 | 20 | 20 |
| 2 | 18 | 18 | 15 |
| 3 | 15 | 16 | 11 |
| 4 | 10 | 14 | 8 |
| 5 | 8 | 10 | 7 |
| 6 | 7 | 10 | 7 |
| 7 | 7 | 7 | 7 |