

1. [Page replacement policies] A computer that has a page size of 1,024 bytes. Consider the following reference string for this computer:

220, 1054, 2320, 4441, 1220, 2321, 5764, 5944, 3200, 10, 11, 2090

What is the number of page faults generated for this string when:

- The page replacement policy is optimal and there are three page frames.
- The page replacement policy is optimal and there are four page frames.
- The page replacement policy is LRU and there are three page frames.
- The page replacement policy is LRU and there are four page frames.
- The page replacement policy is FIFO and there are three page frames.
- The page replacement policy is FIFO and there are four page frames.

For each case, assume that all page frames are vacant when the first address in the reference string is referenced. In addition, indicate which reference results in a page fault.

Answer:

Ref	Virtual address	Page #	Opt+3	Opt+4	LRU+3	LRU+4	FIFO+3	FIFO+4
220	11011100	0	F	F	F	F	F	F
1054	1 0000011110	1	F	F	F	F	F	F
2320	10 0100010000	2	F	F	F	F	F	F
4441	100 0101011001	4	F	F	F	F	F	F
1220	1 0011000100	1						
2321	10 0100010001	2						
5764	101 1010000100	5	F	F	F	F	F	F
5944	101 1100111000	5						
3200	11 0010000000	3	F	F	F	F	F	F
10	1010	0	F		F	F	F	F
11	1011	0						
2090	10 0000101010	2			F		F	F
Page Fault #			7	6	8	7	8	8

2. [From last winter's final] Consider the following fragment of a page table:

page number	frame	valid	permissions	referenced	dirty
12	40	1	r	1	0
13	32	0	rx	0	0
14	1	1	rw	0	0
15	11	1	rw	0	0
16	17	1	rx	0	0
17	6	1	r	0	0

The valid bit indicates whether or not the page frame listed in the page table entry is, in fact, associated with the virtual page number.

The virtual memory system uses Clock with the clock hand scanning the pages in increasing virtual page number order. The hand is currently point to page 15, and this is the page that Clock will consider next when a page frame is needed.

Assume that the page size (and page frame size) is 100 bytes. Consider the operations given below. They are executed in order. For each operation, indicate what happens. For example, if the operation results in a fault, say what kind of fault it is. If it is a page fault, then say which page frame is used to satisfy the page fault and how the clock hand and page table are updated. If the operation does not result in a fault, then say which physical address results from the virtual address translation, and say how the page table is updated.

- Read virtual address 1280.
- Write virtual address 1444.
- Write virtual address 1745.
- Read virtual address 1310.
- Jump to virtual address 1660.

Answer:

- Read virtual address 1280, no fault. Physical address = 4080, Ref bit set to 1;
- Write virtual address 1444, no fault. Physical address = 144, Ref bit set to 1, Dirty bit set to 1;
- Write virtual address 1745, not a page fault but a protection fault, since page 17 is not permitted to write;
- Read virtual address 1310, a page fault generated. Assuming no free page is available, the clock hand is pointing to page 15 of which the ref bit is 0, page frame 11 will be used for page 13. No need to write the old page to disk since page 15 is clean. The valid, ref, and dirty bits will be set to 1 once the new page is in memory.
- Jump to virtual address 1660, no fault. Physical address = 1760, Ref bit set to 1.

The content of page table after the 5 operations:

page number	frame	valid	permissions	referenced	dirty
12	40	1	r	1	0
13	11	1	rx	1	1
14	1	1	rw	1	1
15	11	0	rw	0	0
16	17	1	rx	1	0
17	6	1	r	0	0

3. [Tanenbaum/Woodhull 4.13] Suppose that a 32-bit virtual address is divided into four fields a , b , c and d . The first three are used for a three-level page table system, and the fourth field d is the offset. Does the number of pages depend on the size of all four fields? If not, which ones matter and which ones do not?

Answer: The number of page depends on the physical memory size and the page size. As the page size is determined by d , the number of pages only depends on d , not a , b , or c (except in the trivial sense that $a + b + c + d = 32$, and so depending on d means that it depends on the sum of a , b , and c).

4. [Tanenbaum/Woodhull 4.28] It has been observed that the number of instructions executed between page faults is directly proportional to the number of page frames allocated to a program. If the available memory is doubled, then the mean interval between page faults is also doubled. Suppose that a normal instruction takes 1 microsecond, but if a page fault occurs, it takes 2 milliseconds to handle the fault. If a program takes 60 seconds to run during which time it gets 15,000 page faults, how long would it take to run if twice as much memory were available?

Answer: Time spent in executing program = $60 - 15000 \cdot 2 \cdot 10^{-3} = 30$ seconds
Time spent in executing each interval = $60/15000 - 2 \cdot 10^{-3} = 0.002$ seconds
of page fault when double the memory = $(30/0.002)/2 = 7500$
New execution time = $30 + 7500 \cdot 2 \cdot 10^{-3} = 45$ seconds

5. [Page fault behavior] Give a reference string that has the following property: both *doubling* and *halving* the page size will result in more page faults. Use an LRU page replacement policy. You can choose whatever amount of physical memory and the initial page size you wish when constructing this reference string (*hint<.i>: choose something small*). Show what page faults result in the three cases: original page size, page size doubled, and page size halved.

Answer: Assuming the physical memory is 8 bytes, let us consider the following reference string: (0 2 8 10)*.

- With a page size of 4 bytes, there are two page frames. This string generates 2 page faults (on the initial references to 0 and 8).
- With a page size of 8 bytes, there is one page frame. This string generates an unbounded number of faults - one for each reference to 0 and one for each reference to 8.
- With a page size of 2 bytes, there are four page frames. This string generates 4 page faults (one for each initial reference).