

# COS214 Tutorial 5

Roberto Togneri, 2000

## SOLUTIONS

1. Discuss two approaches to the replacement of pages in a paged memory system. What information is required to optimise the decisions made in the algorithms you describe? Consider the following sequence of page references: 1 2 3 4 4 2 1 4 1 3 4. Determine how many page faults will occur for each algorithm, assuming there are only 2 physical page frames and that both are initially invalid.

### Answer

Page replacement strategies:

- Optimal (need to predict future page references)
- Random
- FIFO (need a queue for selecting the oldest page)
- Least Recently Used / LRU (need timestamp of last reference)
- Second Chance FIFO (need R bit)
- Clock (need R bit)
- Not Recently Used / NRU (need both R and M bits)
- Not Frequently Used / NFU (need R bit and counter)

Optimal

Page requested	1	2	3	4	4	2	1	4	1	3	4
1	1	1*	3*	4	4*	4	4	4*	4	4	4
2		2	2	2*	2	2*	1*	1	1*	3*	3*
Page fault	p	p	p	p			p			p	

6 page faults

FIFO

Page requested	1	2	3	4	4	2	1	4	1	3	4
1	1	1*	3	3*	3*	2	2*	4	4	4*	4*
2		2	2*	4	4	4*	1	1*	1*	3	3
Page fault	p	p	p	p		p	p	p		p	

8 Page faults

LRU

Page requested	1	2	3	4	4	2	1	4	1	3	4
1	1	1*	3	3*	3*	2	2*	4	4*	3	3*
2		2	2*	4	4	4*	1	1*	1	1*	4
Page fault	p	p	p	p		p	p	p		p	p

9 Page faults

2. A segmented memory has paged segments, with each virtual address having a 2-bit segment number, a 2-bit page number and an 11-bit address within the page. The main memory contains 32Kb. Each segment is either read-only (ro), read execute (rx), read-write (rw) or read-write-execute (rwx). The page table and protection is as follows:

Seg 0		Seg 1		Seg 2		Seg 3	
ro		rx		rwx		rw	
Virtual Page	Page Frame	Virtual Page	Page Frame	Virtual Page	Page Frame	Virtual Page	Page Frame
0	9	0	disk	Page table not in memory		0	14
1	3	1	0		1	1	
2	disk	2	15		2	6	
3	12	3	8		3	disk	

- (a) How many virtual pages are there in the address space?  
 (b) How many physical pages can fit in main memory?  
 (c) For each of the following indicate what physical address is computed from the virtual address, if any. If a segment, page or protection fault occurs specify which one.

No	Access	Seg	Page	Offset
1	read	0	1	1
2	read	1	1	10
3	read	3	3	2047
4	write	0	1	4
5	write	2	1	2
6	write	1	0	14
7	jump to	1	3	100
8	read	0	2	50
9	read	2	0	5
10	jump to	3	0	60

### Answer

- (a) 4 segments with 4 virtual pages → 16 virtual pages  
 (b) 11 bits for offset → Page Size =  $2^{11} = 2048(2K) \rightarrow 32K/2K = 16$  physical pages  
 (c)

No	access	seg	page	offset	frame	phys addr	fault
1	read	0	1	1	3	$3 \times 2K + 1 = 6145$	
2	read	1	1	10	0	$0 \times 2K + 10 = 10$	
3	read	3	3	2047	-		page fault
4	write	0	1	4	-		Protection
5	write	2	1	2	-		Segment
6	write	1	0	14	-		protection
7	jump to	1	3	100	8	$8 \times 2K + 100 = 16484$	
8	read	0	2	50	-		page fault
9	read	2	0	5	-		segment
10	jump to	3	0	60	-		protection

3. Page replacement algorithms work best with hardware support. For example, many systems provide *Referenced (R)* and *Modified (M)* flags in hardware. Explain how these flags give the name *Second-Chance FIFO*. Next, explain how the flags could be simulated at only a small amount of expense in software.

**Answer**

In the modified FIFO algorithm the reference ( R ) bit is used to give a page a “second chance” in the FIFO.

The bits can be simulated as follows. When a process is started up, all of its page table entries are marked as not in memory. As soon as a page is referenced, a page fault will occur. The OS then sets the R bit (in its internal tables), changes the page table entry to point to the correct page, with mode READ ONLY, and restarts the instruction. If the page is subsequently written on, another page fault will occur, allowing the OS to set the M bit and change the page’s mode to READ/WRITE.

4. A computer provides each process with 64K of address space divided into pages of 4K. A particular program has a text size of 32,768 bytes, a data size of 16,386 bytes, and a stack size of 15,870 bytes. Will this program fit in the address space? If the page size were 512 bytes, would it fit? Explain! Remember that a page may not contain parts of two different segments.

**Answer**

The text is 8 pages, the data is 5 pages (16,386 = 16K + 2 bytes → 4 + 1) and the stack requires 4 pages → 17 pages  
There are 16 pages in physical memory so the program does not fit.

With 512 byte pages, the text is 64 pages, the data is 33 pages (16,386 = 16K + 2 bytes = 32 + 1) and the stack requires 31 pages → 128 pages  
There are 128 pages in physical memory so the program now fits. A smaller page size results in reduced internal fragmentation and allows the program to fit in physical memory

5. Which of the following programming techniques and structures are “good” for a demand-paged environment? Which are “not good”? Explain your answers.
- Stack
  - Hashed symbol table
  - Sequential search
  - Binary search
  - Pure code
  - Vector operations
  - Indirection

**Answer**

Demand paging is good in techniques or data structures which reference data in the same general vicinity (i.e. hit ratio on the same page is high), e.g. sequential access to an array versus random access.

The following techniques and data structures are good: (a), (c), (e), (f) but the following are not so good: (b), (d), (g)

6. An operating system supports a paged virtual memory, using a central processor with a cycle time of 1 microsecond. It costs an additional 1 microsecond to access a page other than the current one. Pages have 1000 words, and the paging device is a drum that rotates at 3000 revolutions per minute, and transfers 1 million words per second. The following statistical measurements were obtained from the system:

- 1 percent of all instructions executed accessed a page other than the current page.
- Of the instructions that accessed another page, 80 percent accessed a page already in memory.
- When a new page was required, the replaced page was modified 50 percent of the time.

Calculate the effective instruction time on this system, assuming that the system is running one process only, and that the processor is idle during drum transfers. Assume that the drum rotational delay is half a revolution and seek time is negligible.

**Answer**

Time to read/write one page from swap = (rotational delay) + (transfer time)  
3000 rev/min = 50 rev/sec = 0.02 sec/rev → average rotational delay = 0.01 sec  
1000 words and transfer rate of  $10^6$  words/sec → transfer time = 1 μsec  
∴ Time to read/write page = 0.01 sec + 1 μsec = 11,000 μsec

99% of instructions access current page = 0.99 x (1 μsec)  
1% access another page, 80% of these in memory = 0.008 x (2 μsec)  
other 20%, half don’t need replacement = 0.001 x (11,000 μsec)  
half do need replacing = 0.001 x (2 x 11,000 μsec)

**Effective access time: 34 μsec**

7. Consider the following page table:

Index	f	P	time of last access
0	--	0	--
1	0	1	100
2	--	0	--
3	2	1	99
4	--	0	--
5	--	0	--
6	1	1	80
7	3	1	115

What is the PA if VA = [p = 2 | d = 233] is referenced at time 120 and pages are 1K? Assume local scope LRU page replacement. Describe any changes made to the page table.

**Answer**

With p = 2, we get a page fault and need to perform page replacement of the local process. With LRU the candidate page is virtual page 6 / page frame 1. This page is evicted (virtual page 6 has P set to 0) and virtual page 2 now has f = 1 and P = 1 and time of last access set to 120. The PA has f = 1 and d = 233 → 1024 + 233 = 1257.

8. A 64-bit CPU has a VA address range of  $2^{64} = 4\text{GB} \times 4\text{GB}$  which is some really, really, really huge number. What minimum level of page tables do you think is reasonable assuming a 16K page? Comment on the performance cost of adopting such a scheme and is this an issue?

**Answer**

With a 16K page,  $d = 14$  bits and  $64 - 14 = 50$  bits need to be distributed among the  $n$ -level page table indices. Assuming each page table has 64-bit = 8 byte entries and we would like to fit each page table in an integer number of page frames, then the number of bits used to index the page table should be no less than  $(16\text{K}/8) = 2048$  entries =  $2^{11} \rightarrow 11$  bits. With  $(50/11) = 4.5$ , 5-level page tables are implied, but one of these will require an index  $< 11$  bits. Hence a 4-level page table scheme is the most appropriate with  $\text{VA} = [ p | q | r | s | d ]$  and  $p = 12$ ,  $q = 12$ ,  $r = 13$ ,  $s = 13$  and  $d = 14$  bits.

Is there a serious performance cost? With 4-level page tables the memory access is now 5x as slow(!). However if a large enough TLB is available the TLB hit ratio can be quite high, say 98%, and this mitigate the costs of accessing the 4-level page table under normal operation.