CSC 112: Computer Operating Systems
Lecture 8

Memory System II: Paging
Exercises ANS

Department of Computer Science,

Hofstra University

Q1. Inverted Page Table

- Q: A computer system has a 32-bit virtual address space, 4 KB pages, and 512 MB of physical memory.
  a) How many entries are in a conventional single-level page table?
  b) How many entries are in an inverted page table?

ANS:

- a) 4 KB page size = 2^12 bytes, so 32-12 = 20 bits for the page number.
  Number of entries in a conventional page table = 2^20, for each process.
- b) 512 MB physical memory = 2^29 bytes. Each frame is 4 KB = 2^12 bytes.
  Number of frames = 2^29/2^12=2^17.
  So, the inverted page table has 2^17 entries, for the whole system.

Q2. Inverted Page Table

- Q: For a system with a 64-bit virtual address space and 256 MB physical memory, compare the memory requirements for a conventional page table and an inverted page table with 4 KB pages.

ANS:

- Conventional page table:
  Number of virtual pages = 2^64/2^12=2^52 entries.
  This is extremely large and impractical to store in memory.
- Inverted page table:
  Number of physical frames = 2^28/2^12=2^16 entries.
  Much smaller and manageable.

Q1. Page Replacement

- Consider memory size of 3 frames, and following reference stream of virtual pages:
  - 5, 3, 5, 1, 2, 5, 4, 6, 1
- Fill in the table for FIFO, LRU, and OPT page replacement algorithms, and give the number of page faults for each algorithm.

Ref	5	3	5	1	2	5	4	6	1
F1									
F2									
F3									

Q1. Page Replacement ANS

Ref	5	3	5	1	2	5	4	6	1
Frame 1	5	5	5	5	2	2	2	6	6
Frame 2		3	3	3	3	5	5	5	1
Frame 3				1	1	1	4	4	4

FIFO: 8 page faults

Ref	5	3	5	1	2	5	4	6	1
Frame 1	5	5	5	5	5	5	5	5	1
Frame 2		3	3	3	2	2	2	6	6
Frame 3				1	1	1	4	4	4

LRU: 7 page faults

Ref	5	3	5	1	2	5	4	6	1
Frame 1	5	5	5	5	5	5	4	4	4
Frame 2		3	3	3	2	2	2	6	6
Frame 3				1	1	1	1	1	1

OPT: 6 page faults
(When referencing 4 and 6, you can replace any page, as long it page 1 is not replaced, since only it will be referenced again in the future)

Q2. Page Replacement

- Consider memory size of 3 frames, and following reference stream of virtual pages:
  - 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 1, 2, 0
- Fill in the table for FIFO, LRU, and OPT page replacement algorithms, and give the number of page faults for each algorithm.

Ref	7	0	1	2	0	3	0	4	2	3	0	3	1	2	0
F1															
F2															
F3															

Q2. Page Replacement ANS

Ref	7	0	1	2	0	3	0	4	2	3	0	3	1	2	0
F1	7	7	7	2	2	2	2	4	4	4	0	0	0	0	0
F2		0	0	0	0	3	3	3	2	2	2	2	1	1	1
F3			1	1	1	1	0	0	0	3	3	3	3	2	2

FIFO: 12 page faults

Ref	7	0	1	2	0	3	0	4	2	3	0	3	1	2	0
F1	7	7	7	2	2	2	2	4	4	4	0	0	0	2	2
F2		0	0	0	0	0	0	0	0	3	3	3	3	3	0
F3			1	1	1	3	3	3	2	2	2	2	1	1	1

LRU: 12 page faults

Ref	7	0	1	2	0	3	0	4	2	3	0	3	1	2	0
F1	7	7	7	2	2	2	2	2	2	2	2	2	2	2	2
F2		0	0	0	0	0	0	4	4	4	0	0	0	0	0
F3			1	1	1	3	3	3	3	3	3	3	1	1	1

OPT: 8 page faults

Q2. Page Replacement References

- Page replacement Algorithms | FIFO | Example | OS | Lec-26 | Bhanu Priya, Education 4u
  - https://www.youtube.com/watch?v=16kaPQtYo28
- Page replacement Algorithms | LRU | Example | OS | Lec-27 | Bhanu Priya, Education 4u
  - https://www.youtube.com/watch?v=u23ROrlSK_g
- Page replacement Algorithms | OPTIMAL | Example | OS | Lec-28 | Bhanu Priya, Education 4u
  - https://www.youtube.com/watch?v=jeJIKKQcqpU
  - Note that the reference stream is slightly different from FIFO and LRU videos

Q. Paging

- Consider 12-bit virtual address space. Page size is 1 KB. The table shows the 4 entries in the page table. Recall that the valid bit is 1 if the page is resident in physical memory and 0 if the page is on disk or hasn't been allocated. What is the physical memory address corresponding to the virtual memory address 0xF74, 0x374?
- ANS:
- Page size is 1 KB = 2^10, hence offset is 10 bits. 12-bit virtual address space, hence VPN is 12-10=2 bits.
- Virtual address: 0xF74 = 1111 0111 0100 (binary)
  - VPN: 11; Offset: 11 0111 0100
  - Entry for VPN 11 (3 in decimal): Valid bit = 1, PPN = 2
  - Physical address: 1011 0111 0100 (binary) = 0xB74
- Virtual address: 0x374 = 0011 0111 0100 (binary)
  - VPN: 00; Offset: 11 0111 0100
  - Entry for VPN 00 (0 in decimal): Valid bit = 0. hence the page is not in physical memory, there will be a page fault and OS will bring the page into memory. The current PPN is not valid and shown as "/".

VPN	Valid	PPN
0	0	/
1	1	0
2	0	/
3	1	2

Q. Paging

- Consider 46-bit virtual address space. Consider a machine with physical memory size 8 GB, page size of 8 KB, and a page table entry size of 4 bytes. How many levels of page tables are required if each page table fits into a single page?
- ANS: Page size is 8 KB = 2^13, hence offset is 13 bits. 46-bit virtual address space, hence bits used for VPN indexing is 46-13=33 bits. If we use a one-level page table, then the page table has 2^33 entries, too large to fit within a page.
- Since each PTE is 4 bytes, a page table (that fits within one page of 8 KB) contains at most 8 KB/4 B = 2 KB or 2^11 PTEs, that is, the page table at each level has at most 2^11 rows, so the VPN at each level has at most 11 bits. **Number of levels=ceil(33/11) = 3.**
- Breakdown of the 46-bit Virtual Address:
  - 13 bits: Offset within page
  - 11 bits: Level 1 page table index
  - 11 bits: Level 2 page table index
  - 11 bits: Level 3 page table index
- Summary: 3 levels of page tables are required to map a 46-bit virtual address space with the given parameters, because each level can index 2048 entries (11 bits), and 33 bits are needed for indexing (46 total bits minus 13 offset bits).

Q. Paging

- Q: Without a cache or TLB, how many memory operations are required to read or write a single 32-bit word?
- ANS: A memory access incurs 4 actual memory accesses: 3 page table lookups in addition to the actual memory access.
- Q: With a TLB, how many memory accesses can this be reduced to? Best-case scenario? Worst-case scenario?
- ANS: Best-case scenario: 1 memory access. Hit in TLB, once for actual memory operation.
- Worst-case scenario: 4 memory accesses. Miss in TLB + 3 page table lookups in addition to the actual memory operation.

Q. Paging

- Suppose the virtual and physical memory address spaces are both 32 bits with a 4KB page size.
- 1. Suppose you know that there will only be 4 processes running at the same time, each with a working set size of 256KB, i.e., it uses 256 KB memory most of the time. What is the minimum amount of TLB entries that your system would need to support to be able to map/cache the working set size for one process? What happens if you have more entries? What about if you have fewer entries?
- ANS: A process has a working set size of 256KB which means that the working set fits in 256KB/4KB=64 pages. This means our TLB should have 64 entries. If you have more entries, then performance will increase since the process often has changing working sets, and it should be able to store more in the TLB. If it has less, then it can't easily translate the addresses in the working set and performance will suffer.
- 2. Suppose you run some benchmarks on the system and you see that the system is utilizing over 99% of its paging disk IO capacity, but only 10% of its CPU. What is a combination of the of disk space and memory size that can cause this to occur? Assume you have TLB entries equal to the answer from the previous part.
- ANS: The system is thrashing since there isn't enough memory for the benchmark to run without the system page faulting and having to page in new pages. Since there will be 4 processes that have a working set of 256 KB each, swapping will occur as long as the physical memory size is under 1 MB. This happens regardless of the number of TLB entries and disk size. If the physical memory size is lower than the aggregate working set sizes, thrashing is likely to occur.
- 3. Among 1) increasing TLB size, 2) adding more disk space, and 3) adding more memory, which one would lead to the largest performance increase and why?
- ANS: We should add more memory so that we won't need to page in new pages as often.