Sergio Garcia Tapia
Computer Systems: A Programmer's Perspective, by Bryant and O'Hallaron
Chapter 9: Virtual Memory
May 21, 2024

Practice Problems

Exercise 9.1. Complete the following table, filling in the missing entries, and replacing each question mark with the appropriate integer. Use the following units: K = 2^{10} (kilo), M = 2^{20} (mega), G = 2^{30} (giga), T = 2^{40} (tera), P = 2^{50} (peta), or E = 2^{60} (exa).

Number of VA bits (n)	Number of VA (N)	Largest possible VA
8	_____	_____
_____	$2^{?} = 64$ K	_____
_____	_____	$2^{32} - 1 = ?$G - 1
_____	$2^{?} = 256$ T	_____
64	_____	_____

Solution: A virtual address with $N = 2^n$ addresses is an n-bit address space. Note that

- $2^8 = \frac{1}{2^2}2^{10} = \frac{1}{4}$ K.
- $64 \text{ K} = 2^6 \cdot 2^{10} = 2^{16}$
- If the largest address is $2^{32} - 1$, then $n = 32$. Thus $N = 2^{32} = 2^2 \cdot 2^{30}$, which is 4G.
- $256 \text{ T} = 2^8 \cdot 2^{40}$, so $n = 48$. This is $\frac{1}{4}$ P.
- $2^{64} = 2^4 \cdot 2^{60} = 4$.

Number of VA bits (n)	Number of VA (N)	Largest possible VA
8	$2^8 = \frac{1}{4}$ K	$2^8 - 1 = \frac{1}{4}$ K $-$ 1
16	$2^{16} = 64$ K	$2^{16} - 1 = 64$ K $-$ 1
32	$2^{32} = 4$ G	$2^{32} - 1 = 4$ G - 1
48	$2^{48} = 256$ T	$2^{48} - 1 = 256$ T $-$ 1
64	$2^{64} = 4$ E	$2^{64} - 1 = 4$ E $-$ 1

Exercise 9.2. Determine the number of page table entries (PTEs) that are needed for the following combination of virtual address size (n) and page size (P):

n	$P = 2^p$	Number of PTEs
16	4K	_____
16	8K	_____
32	4K	_____
32	8K	_____

Solution: For $n = 16$, there are 2^{16} virtual addresses. According to Section 9.3, the conceptual arrangement of a virtual memory is as an array of N contiguous byte-size cells stored on disk. A page table size is given in bytes, so 4K means 2^{12} bytes. Since we use byte addressing, an address space with a size of 2^{16} can have $2^{16}/2^{12} = 2^4 = 16$ pages. If a page is 8K bytes in size, we can fit $2^{16}/2^{13} = 8$ pages instead.

If $n = 32$, then a 4K page size means $2^{32}/2^{12} = 2^{20} = 1$ M pages. If page size is 8K, then it's $2^{32}/2^{13} = 2^{19} = 2^9 \cdot 2^{10} = 512$K.

n	$P = 2^p$	Number of PTEs
16	4K	16
16	8K	8
32	4K	1M
32	8K	512K

Exercise 9.3. Given a 32-bit virtual address space and a 24-bit physical address, determine the number of bits in the VPN, VPO, PPN, and PPO for the following page sizes P.

Solution: The address pace uses $n = 32$ bits, so the size of the virtual address space is $N = 2^{32}$. The page is is $P = 2^p$ in bytes, where p bits are used for the VPO, and $n - p$ bits are used for the VPN. We also use the same p bits for the PPO. The number of addresses in the physical address space is given by $M = 2^m$. Then $m - p$ bits are used for the PPN.

In this problem, $n = 32$, and $m = 24$. When the page size is 1 KB, this means $P = 2^{10}$, meaning $p = 10$. Thus we use 10 VPO bits and $n - p = 22$ VPN bits. We also use $p = 10$ PPO bits, and we use $m - p = 14$ PPN bits.

For 2 KB pages, we have $p = 2^{11}$, so we use 11 VPO bits, 21 VPN bits, 10 PPO bits, and 13 PPN bits.

	Number of			
P	VPN bits	VPO bits	PPN bits	PPO bits
1 KB	22	10	14	10
2 KB	21	11	13	11
4 KB	20	12	12	12
8 KB	19	13	11	13

Exercise 9.4. Show how the example memory system in Section 9.6.4 translates a virtual address into a physical address and accesses the cache. For the given virtual address, indicate the TLB entry accessed, physical address, and cache byte value returned. Indicate whether the TLB misses, whether a page fault occurs, and whether a cache miss occurs. If there is a cache miss, enter "—" for "Cache byte returned". If there is a page fault, enter "—" for "PPN" and leave parts C and D blank.

Virtual address: `0x03d7`

Solution:

(a) Virtual address format: In the example, we used $n = 14$ bits for the virtual address space, and 1-byte words. The given address in hex can be translated to the following binary sequence:

`00 0011 1101 0111`

13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	1	0	1	0	1	1	1

(b) Address translation: With virtual addresses that are $n = 14$ bits wide and page sizes that are $P = 64 = 2^6$ bytes in size, we use the lower 6 bits for the VPO and upper 8 bits for the VPN. Therefore, the VPN bits are `0000 1111`, which is hex `0x0f`, and the VPO bits are `01 0111`, meaning the VPO in hex is `0x17`. The lower 2 bits of the VPN are used for the TLBI, and the upper 6 bits are used for the TLBT. Thus, `11` is `0x03` for the TLBI, and `00 0011` is `0x03` also for the TLBT. It is a hit for the TLB, since it returns a PPN of `0x0d`. Thus there is no page fault.

Parameter	Value
VPN	`0x0f`
TLB index	`0x03`
TLB tag	`0x03`
TLB hit? (Y/N)	Yes
Page fault? (Y/N)	No
PPN	`0x0d`

(c) Physical Address Format:

11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	0	0	0	1	1	1

(d) Physical memory reference: Now we concatenate the 6-bit PPN `0x0D`, which is `00 1101` and the 6-bit VPO `0x17`, which is `01 0111`, to create the physical address `0011 0101 0111`, or `0x357`.

The MMU sends the physical address to the cache, which extracts the lowest 2 bits for the block offset (CO), the next 4 bits for the cache index (CI), and the highest 6 bits for the cache tag (CT). Thus, CO is `11` or `0x3`, the CI is `0101` or `0x5`, and the highest 6 bits are `00 1101`, or `0x0d`. The tag for the cache set with index `0x5` is `0x0d`, which does not which matches what we have, we have a cache hit, and the valid bit is set. Thus we use the offset, which is `0x3`, to get the byte in block 3, which is `0x1d`.

Parameter	Value
Byte offset	`0x3`
Cache index	`0x5`
Cache tag	`0x0d`
Cache hit ? (Y/N)	Yes
Cache byte returned	`0x1d`

Exercise 9.5. Write a C program `mmapcopy.c` that uses `mmap` to copy an arbitrary-size disk file to `stdout`. The name of the input file should be passed as a command-line argument.