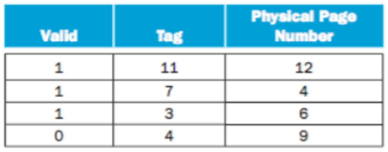
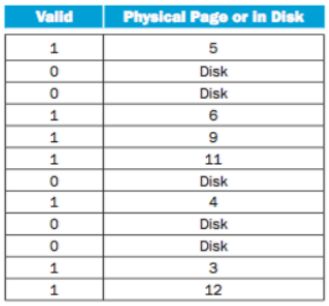
As described in COD Section 5.7 (Virtual memory), virtual memory uses a page table to track the mapping of virtual addresses to physical addresses. This exercise shows how this table must be updated as addresses are accessed. The following data constitutes a stream of virtual addresses as seen on a system. Assume 4 KiB pages, a 4-entry fully associative TLB, and true LRU replacement. If pages must be brought in from disk, increment the next largest page number.

4669, 2227, 13916, 34587, 48870, 12608, 49225
LB:



Valid	Tag	Physical Page Number
1	11	12
1	7	4
1	3	6
0	4	9

Page table:



Valid	Physical Page or In Disk
1	5
0	Disk
0	Disk
1	6
1	9
1	11
0	Disk
1	4
0	Disk
0	Disk
1	3
1	12

(a) For each access shown above, list
- whether the access is a hit or miss in the TLB,
- whether the access is a hit or miss in the page table,
- whether the access is a page fault,
- the updated state of the TLB.

(b) Repeat Part a, but this time use 16 KiB pages instead of 4 KiB pages. What would be some of the advantages of having a larger page size? What are some of the disadvantages?

(c) Repeat Part a, but this time use 4 KiB pages and a two-way set associative TLB.

(d) Repeat Part a, but this time use 4 KiB pages and a direct mapped TLB.

(e) Discuss why a CPU must have a TLB for high performance. How would virtual memory accesses be handled if there were no TLB?

a) First 12 bits are the page offset,
4669 - 123D - Index 1 - Page Fault, Index 1 of page table = 13, TLB puts in value
2227 - 08B3 - Index 0 - Miss in TLB, Hit on the page table
13916 - 365C - Index 3 - Hit in TLB
34587 - 871B - Index 8 - Page fault, index 8 of page table = 14, TLB puts in value index 3 gets evicted
48870 - BEE6 - Index 11 - Hit in TLB
12608 - 3140 - Index 3 - Miss in TLB, Hit in page table, TLB puts in index 3, index 7 gets evicted from TLB
49225 - C049 - Index 12 - Page fault, not enough room in page table

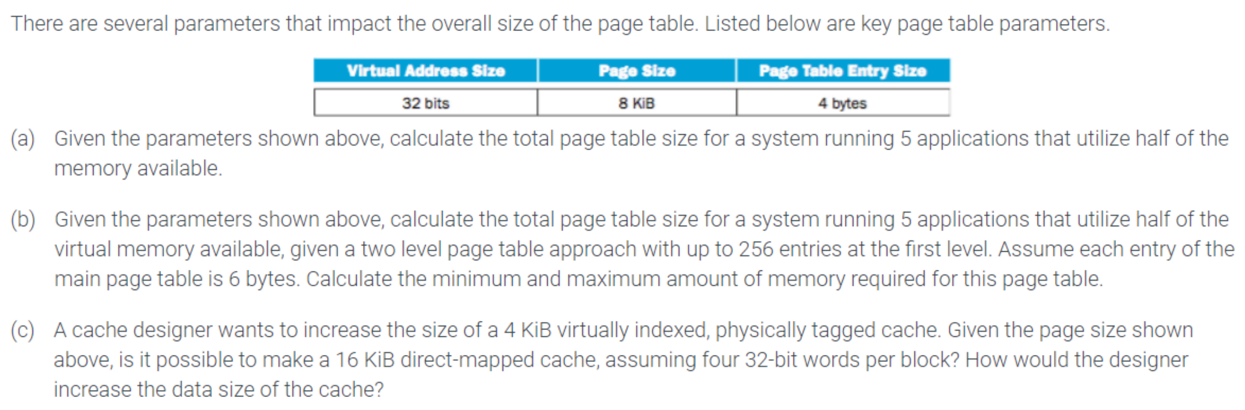
b) First 14 bits now are page offset, advantage is smaller Page Table so you have to store less, but it can take longer as you are moving around more data
4669 - 123D >> 2 - Index 0 -TLB miss, Hit on page table, moves into TLB

2227 - 08B3 >> 2 - Index 0 - TLB hit
13916 - 365C >> 2 - Index 0 - TLB hit
34587 - 871B >> 2 - Index 2 - Page Fault, index 2 is filled with PA 13, TLB puts in index 2, tag 3 gets evicted
48870 - BEE6 >> 2 - Index 2 - TLB hit
12608 - 3140 >> 2 - Index 0 - TLB hit
49225 - C049 >> 2 - Index 3 - TLB miss, Page table hit, TLB put in index 3 evicts index 7

c) Look at the lsb of the page address, that will be the index of the set. Other 3 bits will be the tag bits.
4669 - 123D - 0001, index 1 - TLB index 1 is full, PT miss, page fault, index 1 adds PA 13, tag 3 evicted tag 0 added with PA 13
2227 - 08B3 - 0000, index 0 - Not in TLB, PT hit, TLB has open spot index 0 adds this
13916 - 365C - 0011, index 1 - TLB miss, page table hit, TLB adds tag 1 PA 6 in index 1 evicts tag 11
34587 - 871B - 1000, index 0 - TLB miss, PT miss, Page fault, index 8 of page table gets PA 14, adds this to TLB in index 0
48870 - BEE6 - 1011, index 1 - TLB miss, Page table hit, adds this to TLB evicts tag 0
12608 - 3140 - 0011, index 1 - TLB hit
49225 - C049 - 1100, index 0 - TLB miss, PT miss, Page fault, not in indices of page table so can't add it

d) Look at last 2 bits that will be the index for the set
4669 - 123D - 0001 - TLB miss, PT miss, page fault, index 1 of page table adds PA 13, location 01 (binary) in TLB (tag 3) replaced with tag 0 PA 13
2227 - 08B3 - 0000 - TLB miss, PT hit, evicts tag 11 out of index 00 (binary) and adds tag 0
13916 - 365C - 0011 - TLB miss, PT hit, location 11 (binary) in TLB (unoccupied) filled with tag 0 PA 6
34587 - 871B - 1000 - TLB miss, PT miss, page fault, index 8 of page table adds PA 14, location 00 in TLB replaced with tag 2 and PA 14
48870 - BEE6 - 1011 - TLB miss, PT hit, location 11 (binary) replaced with tag 2 PA 12
12608 - 3140 - 0011 - TLB mis, PT hit, adds location to TLB in index 11 (binary)
49225 - C049 - 1100 - TLB miss, PT miss, page fault

e) Otherwise every single memory operation would require the page to be loaded. Pages leverage temporal locality very well to keep speeds high for virtual memory.


There are several parameters that impact the overall size of the page table. Listed below are key page table parameters.

Virtual Address Size	Page Size	Page Table Entry Size
32 bits	8 KiB	4 bytes

(a) Given the parameters shown above, calculate the total page table size for a system running 5 applications that utilize half of the memory available.

(b) Given the parameters shown above, calculate the total page table size for a system running 5 applications that utilize half of the virtual memory available, given a two level page table approach with up to 256 entries at the first level. Assume each entry of the main page table is 6 bytes. Calculate the minimum and maximum amount of memory required for this page table.

(c) A cache designer wants to increase the size of a 4 KiB virtually indexed, physically tagged cache. Given the page size shown above, is it possible to make a 16 KiB direct-mapped cache, assuming four 32-bit words per block? How would the designer increase the data size of the cache?

a) 2^32 / 2^13 = 2^19 / 2 = 2^18 * 5 applications * 4 bytes
b) Min : 256 * 6 bytes   Max: 2^24 * 4 bytes
c) Yes this is possible as our page size is 8 KiB which is < 16 KiB. We can do this by increasing the associativity or increasing the block size.

Test Question:

Determine write policy of cache. Write through or write back. k denotes a hit.
Load A
Store B

Determine write policy of cache. Write allocate or no-allocate. k denotes a hit.
Load A
Store B
Store A k
Load A k
Load B k
Load B k
Load A k