Exercise 5.10

As described in Section 5.4, 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 table is a stream of virtual addresses as seen on a system. Assume 4 KB pages, a four-entry fully associative TLB, and true LRU replacement. If pages must be brought in from disk, increment the next largest page number.

4095，31272，15789，15000，7193，4096，8912

TLB

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

Page table

Index	Valid	Physical Page or In Disk
0	1	5
1	0	Disk
2	0	Disk
3	1	6
4	1	9
5	1	11
6	0	Disk
7	1	4
8	0	Disk
9	0	Disk
10	1	3
11	1	12

5.10.1 [10] <5.4> Given the address stream in the table, and the initial state of the TLB and page table, show the final state of the system. List for each reference if it is a hit in the TLB, a hit in the page table, or a page fault.

Ans:

Binary address(all hexadecimal), (bits 15-12 virtual pages, 11-0 page offset)

4095 = FFF_{16}, virtual page =0, Miss in TLB, hit in page table

31272= $7A28_{16}$ = virtual page 7, Hit in TLB,

15789 = $3DAD_{16}$,=virtual page 3 Hit in TLB

15000 =$3A98_{16}$ =virtual page 3 Hit in TLB

7193=$1C19_{16}$ =virtual page 1 Page Fault

4096 = 1000_{16} = page 1, Hit in TLB

8912 = $22D0_{16}$ = page 2 Page Fault

H: Hit in TLB, M: Miss in TLB hit in page table, PF: Page Fault
Access stream: 4095 (page 0)，31272 (page 7)，15789 (page 3)，15000 (page 3)，7193(page 1)，4096 (page 1)，8912 (page 2)
TLB

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

After 4095 (page 0)

Valid	Tag	Physical Page Number
1	11	12
1	7	4
1	3	6
1	0	5

After 31272 (page 7) and 15789 (page 3) and 15000 (page 3)

Valid	Tag	Physical Page Number
1	11	12
1	7	4
1	3	6
1	0	5

After 7193 (page 1)

Valid	Tag	Physical Page Number
1	1	13 (=12+1)
1	7	4
1	3	6
1	0	5

After 4096, hit
After 8912 Page Fault

Valid	Tag	Physical Page Number
1	1	13
1	7	4
1	3	6
1	2	14 (=13+1)

0, 7, 3, 3, 1, 1, 2 (M, H, H, H, PF, H, PF)
TLB

Valid	Tag	Physical Page Number
1	1	13
1	7	4

1	3	6
1	2	14 (=13+1)

Page table

Index	Valid	Physical Page or In Disk
0	1	5
1	1	13
2	1	14
3	1	6
4	1	9
5	1	11
6	0	Disk
7	1	4
8	0	Disk
9	0	Disk
10	1	3
11	1	12

5.10.2 [15] <5.4> Repeat Exercise 5.10.1, but this time use 16 KB pages instead of 4 KB pages. What would be some of the advantages of having a larger page size? What are some of the disadvantages?

Ans:

Virtual page number: Address >> 14 bits, and assume no valid entries exist in TLB,

Binary address(all hexadecimal), (bits 15-12 virtual pages, 11-0 page offset)

4095 = FFF_{16}, virtual page =0, Miss in TLB,

31272= $7A28_{16}$ = virtual page 1, Page Fault

15789 = $3DAD_{16}$,=virtual page 0 Hit in TLB

15000 =$3A98_{16}$ =virtual page 0 Hit in TLB

7193=$1C19_{16}$=virtual page 0, Hit in TLB

4096 = 1000_{16} = page 0, Hit in TLB

8912 = $22D0_{16}$ = page 0, Hit in TLB

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

After the access stream

Valid	Tag	Physical Page Number
1	1	13
1	7	4
1	3	6

1	0	5

Index	Valid	Physical Page or In Disk
0	1	5
1	1	13
2	0	Disk
3	1	6
4	1	9
5	1	11
6	0	Disk
7	1	4
f8	0	Disk
9	0	Disk
10	1	3
11	1	12

H: Hit in TLB, M: Miss in TLB hit in page table, PF: Page Fault

0, 1, 0, 0, 0, 0, 0 (M, PF, H, H, H, H, H)

Larger page sizes allow for more addresses to be stored in a single page, potentially decreasing the amount of pages that must be brought in from disk and increasing the coverage of the TLB. However, if a program uses addresses in a sparse fashion (for example randomly accessing a large matrix), then there will be an extra penalty from transferring larger pages compared to smaller pages.

5.10.3 Moved to Backup problem

Exercise 5.10 Part B

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

Virtual Address Size	Page Size	Page Table Entry Size
32 bits	4 KB	4 bytes

5.10.4 [5] <5.4> Given the parameters in the table above, calculate the total page table size for a system running five applications that utilize half of the memory available.

Ans:

4 KB page = 12 offset bits, 20 page number bits

2^{20} = 1 M page table entries

1 M entries × 4 bytes/entry = ~4 MB (2^{22} bytes) page table per application

2^{22} bytes × 5 apps = 20.97 MB total

5.10.5 [10] <5.4> Given the parameters in the table above, calculate the total page table size for a system running five applications that utilize half of the memory available, given a two-level page table approach with 256 entries. Assume each entry of the main page table is 6 bytes. Calculate the minimum and maximum

amount of memory required.

Ans:

4 KB page = 12 offset bits, 20 page number bits

256 entries (8 bits) for first level => 12 bits => 4096 entries per second level

Minimum: 128 first level entries used per app

128 entries × 4096 entries per second level = ~524K (2^{19}) entries

~524K × 4 bytes/entry = ~2 MB (2^{21}) second level page table per app

128 entries × 6 bytes/entry = 768 bytes first level page table per app

~10 MB total for all 5 apps

Maximum: 256 first level entries used per app

256 entries × 4096 entries per second level = ~1M (2^{20}) entries

~1M × 4 bytes/entry = ~4 MB (2^{22}) second level page table per app

256 entries × 6 bytes/entry = 1536 bytes first level page table per app

~20.98 MB total for all 5 apps

5.10.6 Moved to backup problems

Exercise 5.11

In this exercise, we will examine space/time optimizations for page tables. The fo1lowing table shows parameters of a virtual memory system.

Virtual address (bits)	Physical DRAM installed	Page size	PTE size (byte)
32	4 GB	8 KB	4

5.11.1 [10] <5.4> For a single-level page table, how many page table entries (PTEs) are needed? How much physical memory is needed for storing the page table?

Ans

Virtual address 32 bits, physical memory 4 GB

page size 8 KB or 13 bits, page table entry 4 bytes or 2 bits

#PTE = 32 – 13 = 19 bits or 512K entries

PT physical memory = 512K × 4 bytes = 2 MB

5.11.2 [10] <5.4> Using a multilevel page tab1e can reduce the physical memory consumption of page tab1es, by only keeping active PTEs in physical memory. How many levels of page tables will be needed in this case? And how many memory references are needed for address translation if missing in TLB?

Ans:

virtual address 32 bits, physical memory 4 GB

page size 8 KB or 13 bits, page table entry 4 bytes or 2 bits

#PTE = 32 – 13 = 19 bits or 512K entries

8 KB page/4 byte PTE = 2^{11} pages indexed per page

Hence with 2^{19} PTEs will need 2-level page table setup.

Each address translation will require at least 2 physical memory accesses.

5.11.3 Moved to backup problems

Exercise 5.11 Part B

The following tab1e shows the contents of a 4-entry TLB.

Entry-ID	Valid	VA page	Modified	Protection	PA page
1	1	140	1	RW	30
2	0	40	0	RX	34
3	1	200	1	RO	32
4	1	280	0	RW	31

5.11.4 [5] <5.4> Under what scenarios would entry 2's valid bit be set to zero?

Ans. TLB initialization, or process context switch.

5.11.5 [5] <5.4> What happens when an instruction writes to VA page 30?

Ans. TLB miss.

5.11.6 [5] <5.4> What happens when an instruction writes to VA page xxx?

Ans. Write protection exception.

Exercise 5.12

In this exercise, we will examine how replacement policies impact miss rate. Assume a two-way set-associative cache with four blocks. You may find it helpful to draw a table like those found on page 483 to solve the problems in this exercise, as demonstrated below on the address sequence “0,1,2,3,4”.

Address of memory block accessed	Hit or miss	Evicted block	Contents of cache blocks after reference			
			Set 0	Set 0	Set 1	Set 1
0	Miss		Men[0]			
1	Miss		Men[0]		Men[1]	
2	Miss		Men[0]	Men[2]	Men[1]	
3	Miss		Men[0]	Men[2]	Men[1]	Men[3]
4	Miss	0	Men[4]	Men[2]	Men[1]	Men[3]
…						

The following table shows address sequences.

Address sequence
0,2,4,0,2,4,0,2,4

5.12.1 [5] <5.3, 5.5> assuming an LRU replacement policy, how man hits does this address sequence exhibit?

ANS: 0 hits

Address of memory	Hit or miss	Evicted block	Contents of cache blocks after reference			
			Set 0	Set 0	Set 1	Set 1

block accessed				
0	Miss		Men[0]	
2	Miss		Men[0]	Men[2]
4	Miss	Men[0]	Men[4]	Men[2]
0	Miss	Men[2]	Men[4]	Men[0]
2	Miss	Men[4]	Men[2]	Men[0]
4	Miss	Men[0]	Men[2]	Men[4]
0	Miss	Men[2]	Men[0]	Men[4]
2	Miss	Men[4]	Men[0]	Men[2]
4	Miss	Men[0]	Men[4]	Men[2]

5.12.2 [5] <5.3, 5.5> assuming an MRU (most recently used) replacement policy, how many hits does this address sequence exhibit?

ANS: 3 hits

Address of memory block accessed	Hit or miss	Evicted block	Contents of cache blocks after reference			
			Set 0	Set 0	Set 1	Set 1
0	Miss		Men[0]			
2	Miss		Men[0]	Men[2]		
4	Miss	Men[2]	Men[0]	Men[4]		
0	Hit		Men[0]	Men[4]		
2	Miss	Men[0]	Men[2]	Men[4]		
4	Hit		Men[2]	Men[4]		
0	Miss	Men[4]	Men[2]	Men[0]		
2	Hit		Men[2]	Men[0]		
4	Miss	Men[2]	Men[4]	Men[0]		

5.12.3 [5] <5.3, 5.5> Moved to backup problems

5.12.4 [10] <5.3, 5.5> Moved to backup problems

5.12.5 [10] <5.3, 5.5> describe why it is difficult to implement a cache replacement policy that is optimal for all address sequences.

ANS:

The best block to evict is the one that will cause the fewest misses in the future. Unfortunately, a cache controller cannot know the future! Our best alternative is to make a good prediction.

Exercise 5.13 Moved to backup problems