**Homework 5 (Due: Nov 29)**

# **Instructions**

* Complete the following questions to the best of your ability.
* Answers should be clear, concise, and justified with work.
* Please write your name and NetID on the hardcopy of your solution, and bring it to the lecture on the due day. Please submit your solution before the lecture starts.

# **Collaboration**

* Professor/TAs
  + You **may** discuss any content with any professor or TA.
* Students
  + You **may** discuss high-level concepts, techniques, jargon, keywords, related problems, or course content that is relevant to the material.
  + You **may not** discuss particular solutions to any of the questions.
  + If you have substantial conversations with any students, please note their name and, if you feel it necessary, the extent of your collaboration.
* Outside Sources (Internet, books)
  + You **may** use external references for any course content.
  + You **may** use an external tool to verify your solutions where appropriate, but not to generate solutions to any questions.
  + List any outside sources that you use. Formal citations are not necessary; links are fine.

# **Question 1 (50 points)**

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 4KB page size, a 4-entry fully associative TLB, and true LRU replacement. Assume initial LRU way is the one with tag 11. If a page must be brought in from disk, assume the page number is current largest page number plus 1.

**Address stream (in decimal)**: 573, 13920, 6323, 34587, 32776, 20800

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TLB:   |  |  |  | | --- | --- | --- | | **Valid** | **Tag** | **Physical Page Number** | | 1 | 11 | 12 | | 1 | 7 | 4 | | 1 | 3 | 6 | | 0 | 4 | 9 | | Page Table:   |  |  | | --- | --- | | **Valid** | **Physical Page Number** | | 1 | 5 | | 0 | Disk | | 0 | Disk | | 1 | 6 | | 1 | 9 | | 1 | 11 | | 0 | Disk | | 1 | 4 | | 0 | Disk | | 0 | Disk | | 1 | 3 | | 1 | 12 | |

1. Given the address stream shown, and the initial TLB and page table states provided above, show the final state of the system. Also list for each reference if it is a hit in the TLB, a hit in the page table or a page fault.

Address : pageNum

573 : 0

13920 : 3

6323 : 1

34587 : 8

32776 : 8

20800 : 5

When 573:

TLB miss

Page Table hit

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

When 13920:

TLB hit

When 6323:

TLB miss

Page Table miss

Page fault

|  |  |
| --- | --- |
| **Valid** | **Physical Page Number** |
| 1 | 5 |
| 1 | 13 |
| 0 | Disk |
| 1 | 6 |
| 1 | 9 |
| 1 | 11 |
| 0 | Disk |
| 1 | 4 |
| 0 | Disk |
| 0 | Disk |
| 1 | 3 |
| 1 | 12 |

|  |  |  |
| --- | --- | --- |
| **Valid** | **Tag** | **Physical Page Number** |
| 1 | 1 | 13 |
| 1 | 7 | 4 |
| 1 | 3 | 6 |
| 1 | 0 | 5 |

When 34587:

TLB miss

Page Table miss

Page fault

|  |  |
| --- | --- |
| **Valid** | **Physical Page Number** |
| 1 | 5 |
| 1 | 13 |
| 0 | Disk |
| 1 | 6 |
| 1 | 9 |
| 1 | 11 |
| 0 | Disk |
| 1 | 4 |
| 1 | 14 |
| 0 | Disk |
| 1 | 3 |
| 1 | 12 |

|  |  |  |
| --- | --- | --- |
| **Valid** | **Tag** | **Physical Page Number** |
| 1 | 1 | 13 |
| 1 | 8 | 14 |
| 1 | 3 | 6 |
| 1 | 0 | 5 |

When 32776:

TLB hit

When 20800:

TLB miss

Page Table hit

|  |  |  |
| --- | --- | --- |
| **Valid** | **Tag** | **Physical Page Number** |
| 1 | 1 | 13 |
| 1 | 8 | 14 |
| 1 | 3 | 6 |
| 1 | 5 | 11 |

So the final state is:

|  |  |  |
| --- | --- | --- |
| **Valid** | **Tag** | **Physical Page Number** |
| 1 | 1 | 13 |
| 1 | 8 | 14 |
| 1 | 3 | 6 |
| 1 | 5 | 11 |

|  |  |
| --- | --- |
| **Valid** | **Physical Page Number** |
| 1 | 5 |
| 1 | 13 |
| 0 | Disk |
| 1 | 6 |
| 1 | 9 |
| 1 | 11 |
| 0 | Disk |
| 1 | 4 |
| 1 | 14 |
| 0 | Disk |
| 1 | 3 |
| 1 | 12 |

1. Repeat (a), but this time use 8KB pages instead of 4KB pages. What would be some of the advantages of having a larger page size? What are some of the disadvantages?

Address : pageNum

573 : 0

13920 : 1

6323 : 0

34587 : 4

32776 : 4

20800 : 2

When 573:

TLB miss

Page Table hit

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

When 13920:

TLB miss

Page Table miss

Page fault

|  |  |
| --- | --- |
| **Valid** | **Physical Page Number** |
| 1 | 5 |
| 1 | 13 |
| 0 | Disk |
| 1 | 6 |
| 1 | 9 |
| 1 | 11 |
| 0 | Disk |
| 1 | 4 |
| 0 | Disk |
| 0 | Disk |
| 1 | 3 |
| 1 | 12 |

|  |  |  |
| --- | --- | --- |
| **Valid** | **Tag** | **Physical Page Number** |
| 1 | 1 | 13 |
| 1 | 7 | 4 |
| 1 | 3 | 6 |
| 1 | 0 | 5 |

When 6323:

TLB hit

When 34587:

TLB miss

Page Table hit

|  |  |  |
| --- | --- | --- |
| **Valid** | **Tag** | **Physical Page Number** |
| 1 | 1 | 13 |
| 1 | 4 | 9 |
| 1 | 3 | 6 |
| 1 | 0 | 5 |

When 32776:

TLB hit

When 20800:

TLB miss

Page Table miss

Page Fault

|  |  |
| --- | --- |
| **Valid** | **Physical Page Number** |
| 1 | 5 |
| 1 | 13 |
| 1 | 14 |
| 1 | 6 |
| 1 | 9 |
| 1 | 11 |
| 0 | Disk |
| 1 | 4 |
| 0 | Disk |
| 0 | Disk |
| 1 | 3 |
| 1 | 12 |

|  |  |  |
| --- | --- | --- |
| **Valid** | **Tag** | **Physical Page Number** |
| 1 | 1 | 13 |
| 1 | 4 | 9 |
| 1 | 2 | 14 |
| 1 | 0 | 5 |

Advantages:

1. Page table can be smaller
2. Page fault will be reduced

Disadvantages:

1. Taking more time to fetch one page
2. More waste on the page may result in a greater likelihood of unused space
3. Show the final contents of the TLB if it is 2-way set associative. Also show the contents of the TLB if it is direct mapped. Discuss the importance of having a TLB to high performance. How would virtual memory accesses be handled if there were no TLB? You can assume your initial LRU way and initial Index.

2-way set associative:

Set0

Way0: 0->5

Way1: 4->14

Set1

Way0: 2->11

Way1: 0->13

direct mapped:

Set0: 2->14

Set1: 1->11

Set2:

Set3: 0->6

Importance of TLB:

1. Speed up the process of VA->PA translation

If there is no TLB:

1. There will be higher rate of page fault
2. Taking more time to access page table

# **Question 2 (30 points)**

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 | 4 KB | 4 bytes |

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

The POFS in virtual address = log\_2(4KB) = 12 bits

The tags in virtual address = 32bits - 12bits = 20 bits

The pages in a page table = 2^20 = 1M

The pages available = 1M / 2 = 0.5M

The page table is 4B\*0.5M = 2MB

The total page size = 2MB \* 5 = 10MB

1. Given the parameters shown above, calculate the total page table size for a system running 5 applications that utilize half of the memory available, given a two-level page table approach. Assume each entry of the main page table is 6 bytes. Calculate the minimum and maximum amount of memory required.

Same as above, the number of low-level pages is also = 2^20 = 1M

For second-level page(4KB), it can hold 4KB/4B = 1K PTEs

And it can cover 1K \* 4KB = 4MB

Totally, we should have 1M pages, so first-level page table should have 1M/1K = 1K PTEs

The size of first-level page table is 1K \* 6B = 6KB

For minimum, each program hold the virtual memory which can be filled into half of the second-level page tables exactly.

Second-level: 5 \* (2^31B / 4MB) \* 4KB = 10MB

First-level: 5 \* 6KB = 30B

For maximum, all the second-level page tables are used

Second-level: 10MB \* 2 = 20MB

First-level: 30KB \* 2 = 60B

1. A cache designer wants to increase the size of a 4KB virtually indexed, physically tagged cache. Given the page size shown above, is it possible to make a 16KB direct-mapped cache, assuming 2 words per block? How would the designer increase the data size of the cache?

Originally,

Offset = 3 bits

Index bit = log2(4KB / 8B) = 9

So, there is no overlap with VPN

Now,

Index bit = log2(16KB/8B) = 11

Index bit + offset = 14, so it takes [13:0], there is an overlap with VPN([31:12])

So, it is impossible to make a 16KB direct-mapped cache

The designer should use 4 ways in 2^9 sets.

# **Question 3 (15 points)**

For each question, circle the best answer. If none of the selections are appropriate, then choose “e. None of the above”.

B 1. Which of the following is an advantage of interrupts over polling?

a. Interrupts are a simpler mechanism for both the hardware and software.

b. Interrupts allow for more efficient CPU utilization.

c. Interrupts are compatible with virtual memory, while polling is not.

d. Polling causes bad hit rates in the L1 instruction cache.

e. None of the above

E 2. What three operations (in the correct order) are required to read data from the hard disk?

a. jal, sw, jr

b. Wait for rotation, seek, read data as it spins under

c. Interrupt, seek, read data as it spins under

d. Seek, wait for rotation, read data as it spins under

e. None of the above

B 3. Which of the following would not cause an exception?

a. Division by zero.

b. A branch is mis-predicted.

c. Attempting to execute a privileged instruction in a normal application.

d. A load or store does not have a valid virtual to physical translation.

e. None of the above (all cause exceptions).

# **Question 4 (5 points)**

Briefly explain the differences between Exceptions and Interrupts using one or two sentences at most.

Exceptions are typically caused by a program's erroneous behavior or exceptional conditions, while interrupts are external signals that temporarily suspend the normal execution of a program to handle events or perform specific tasks.