CS2106
Introduction to OS

Office Hours, Week 11

Agenda

- Virtual Memory
- Hierarchical Paging
- Inverted Page Tables
- Page Replacement Algorithms
- Miscellaneous

- A few questions remained unanswered
  - Will cover them during revision

Virtual Memory

Is kernel memory always in primary memory (and never in secondary memory)?

- In today's operating systems, yes.
- However, the opposite is possible.

Virtual Memory

When a computer is shut down, what happens to the virtual memory in the storage? Is it wiped or just stays there?

- Virtual memory belongs to a process.
- By the time the computer is off, the process is terminated and does not exist anywhere.
- The swap area in storage should be wiped out for security reasons
  - But not mandatory
- Optionally, the swap partition could be encrypted

Virtual Memory

Why is demand paging not related to caching and hence locality principle?

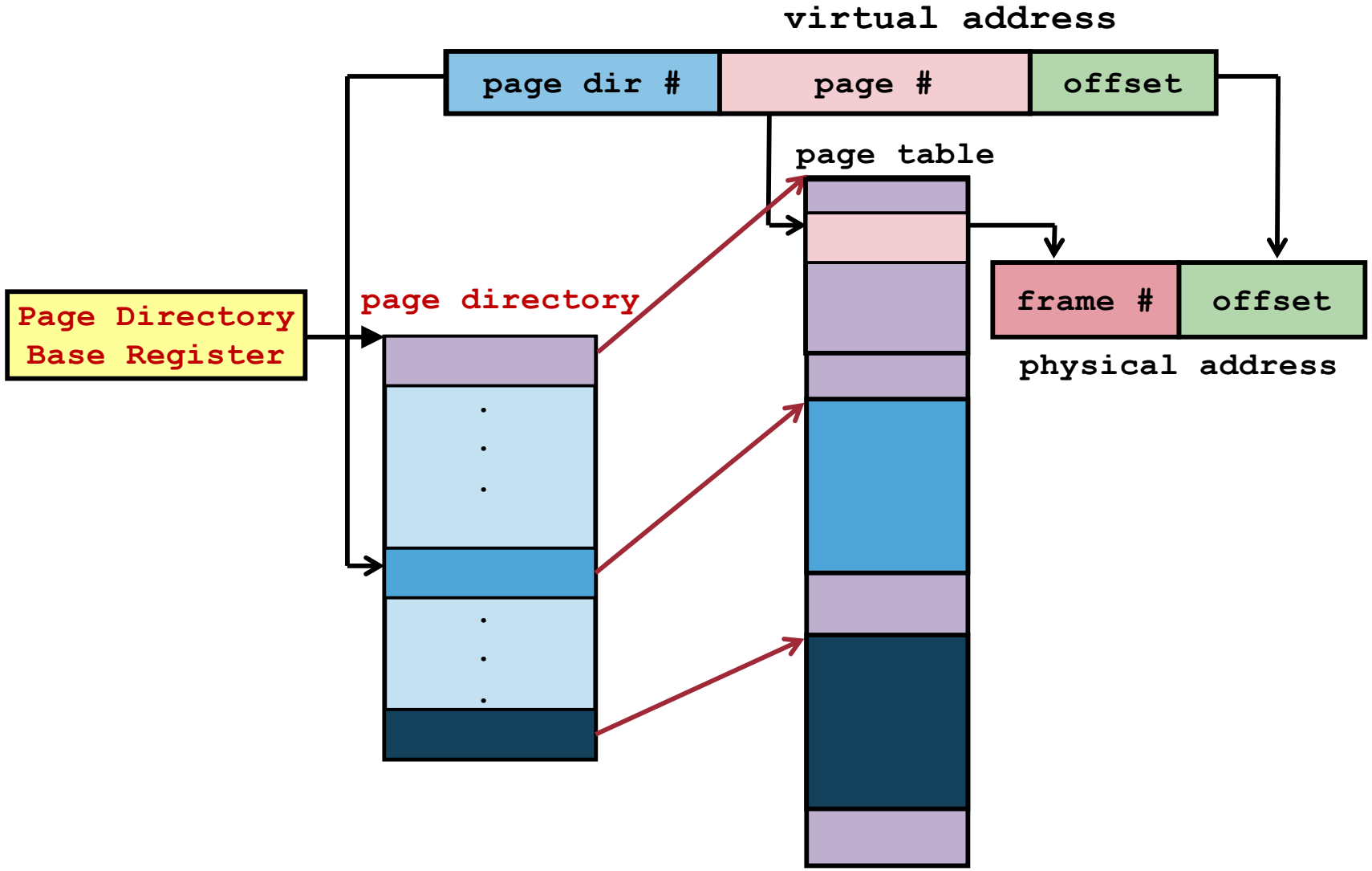
- (in relation to the archipelago quiz)
- Indeed, it is indirectly related.
- Bringing pages lazily would not perform well if the caching functionality of VM was not working properly.
- Good point!

2-Level Page Tables

What does it mean by the page table region is empty/not being used? If we are going to discard the page table regions that are not needed in 2 level paging, why must we still have a null pointer in the page directory? Isn't the region non-existent?

How is it possible that some regions are empty? In the original page table (before splitting), shouldn't every page table entry contain something? How can table entries be empty?

2-level Paging



2-Level Page Tables

- Huge parts of the original single-level page tables are unused
  - The program is not using that part of the virtual address space
  - However, we must keep space for all entries in a directly indexed array so that we can get the mapping in one memory access

- In case of 2-level, we can represent the entire 2^{nd} level table as a NULL pointer in the 1^{st} level to save space

- We now have to keep an empty pointer in the directory
  - Because we are directly indexing the directory!

2-Level Page Tables

In 2-level paging, after splitting page table into regions, if all regions are used, it means we won't remove any of these regions. In this case, we will not save on any space right?

- Correct. Non-empty regions must be there.

Then how should we decide if a page table region should be kept or not?

- Based on whether the page table region corresponds to used or unused part of the virtual address space

Paging of the Page Tables (**IMPORTANT**)

Is it correct to say that since page tables are located in memory, these page tables are also split into pages?

- Yes! Excellent question!
- And this causes an important problem with 1-level page tables (which I haven't seen in OS books):
  - 1-level page tables can easily be bigger than a page
  - However, they must be continuous in physical memory
    - Otherwise, obtaining the mapping in one memory access would not be possible
    - This is despite the use of paging
- The requirement that the OS must keep a giant mostly unused table in contiguous pages of physical memory is super impractical!

Paging of the Page Tables

From slide 21, it was stated that page size is 4KiB. However, it was also stated that this is the page table region size. Why is page size the same as page table region size? Isn't a page and page table region different? Why are their sizes the same?

- Each page table is limited to a page
- Beyond a single page, it requires contiguity in physical memory
- So, all page tables and directories of any level (except the top) are designed to be of the same size (==page size)

During the lecture there was a lot of mention of the word "page", but I don't know if it was referring to page table regions or logical pages.

- See above

Size of Small Page Tables

If the page table must have 2^11 entries, how do we guarantee it won't be < 2^11 entries? for example, I can create program with only 10 entries?

- (Refers to the setup with page size of 4KB, and PTE size 8B)
- Can be less than a full page
  - However, that results in internal fragmentation.
  - The remaining part of the page will be unused unless/until the missing part of the memory region is allocated.

Paging of Directories and so on…

Since the page directory is also stored in memory, is it split into pages as well?

Yes!

If that is the case, does it mean we will have a hierarchical page table?

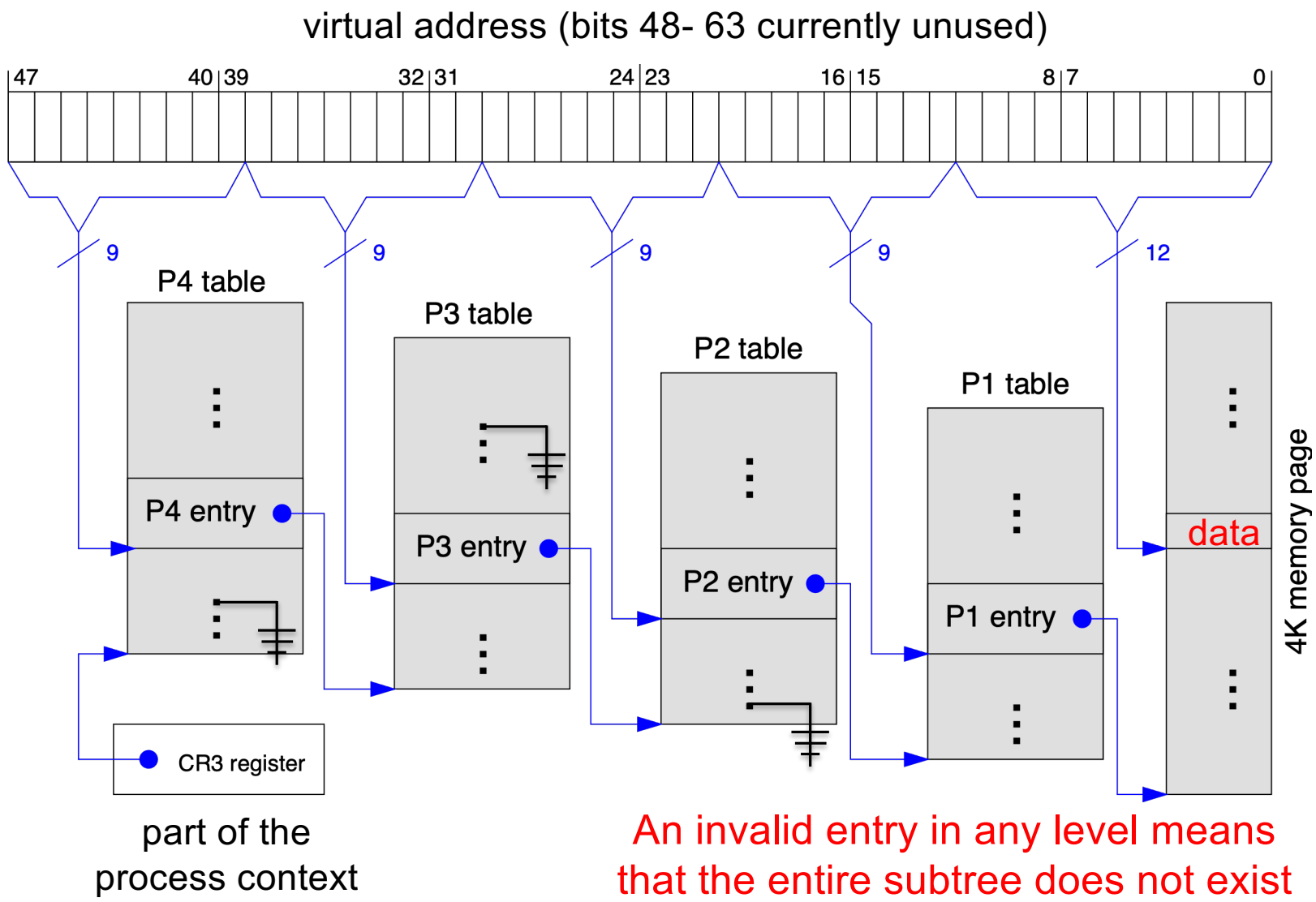
Yes!

Is hierarchical page table a case of splitting the page directory into pages?

Yes!

Great questions!

Hierarchical Page Table – Modern Systems (***Often in Exams***)



Hierarchical Page Tables

In hierarchical page table diagram, is it correct to say that P4, P3, P2 tables are page directories?

- Yes.

And only P1 table is the actual page table region?

- Yes.

Also, is it right to say that only the deepest level (e.g., P1 table) contain page table regions?

- Yes.

Why is there 4 table p1...p4? Is this considered 4 level paging?

- Yes.

Hierarchical Page Tables

- **Is having more layers in hierarchical paging better than having fewer layers? If so how? Since having more layers would mean that we need to keep track of more page directories?**
- **Is here any reason why 4-level paging was chosen, rather than say more than 4 levels?**
  - Excellent questions!
  - There isn't much choice actually: given the size of the logical address, page-size, and size of PTE, everything is fixed.
  - E.g., on x86 only 512 entries can fit in a page (9 bits needed to index)
  - To use 48-bits of the VA, you have to break it into 4 levels

Hierarchical Page Tables

I didn't get how TLB is a solution for 2-level paging? What does TLB do?

- The key problem with 2-level paging is that two extra memory accesses are needed for each data access.
- However, a TLB hits will avoid the extra memory accesses, regardless of how many of them are there.

Inverted Page Tables

Is inverted table mapping from virtual to physical address? And the normal one is from physical to virtual?

The opposite is true:

- Direct ("normal") table: logical (or virtual) → to physical
- Inverted table: physical → logical + process ID

Is inverted page table holding mappings between PID to page number or page number to PID?

- Neither. Physical Frame → Logical Page + PID

Also does it store the frame number or the page number?

- It stores the **logical page number** (+PID); it's looked up by the frame number!

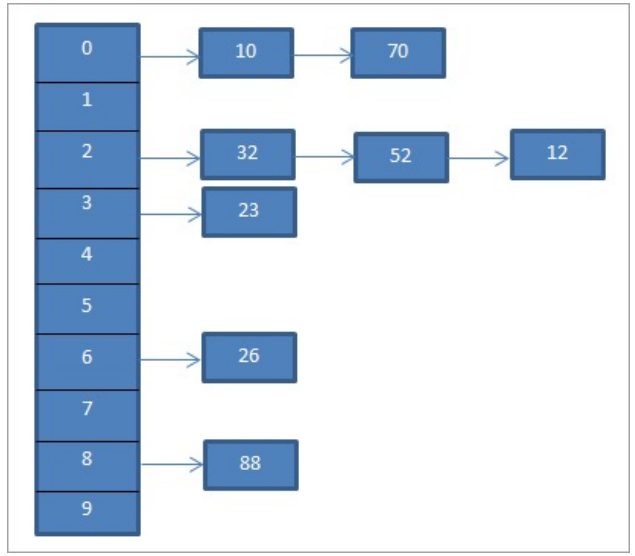
Inverted Page Tables

Is it correct to say if there are N physical frames, there are N entries in the inverted table. Each entry stores process ID and logical page number?

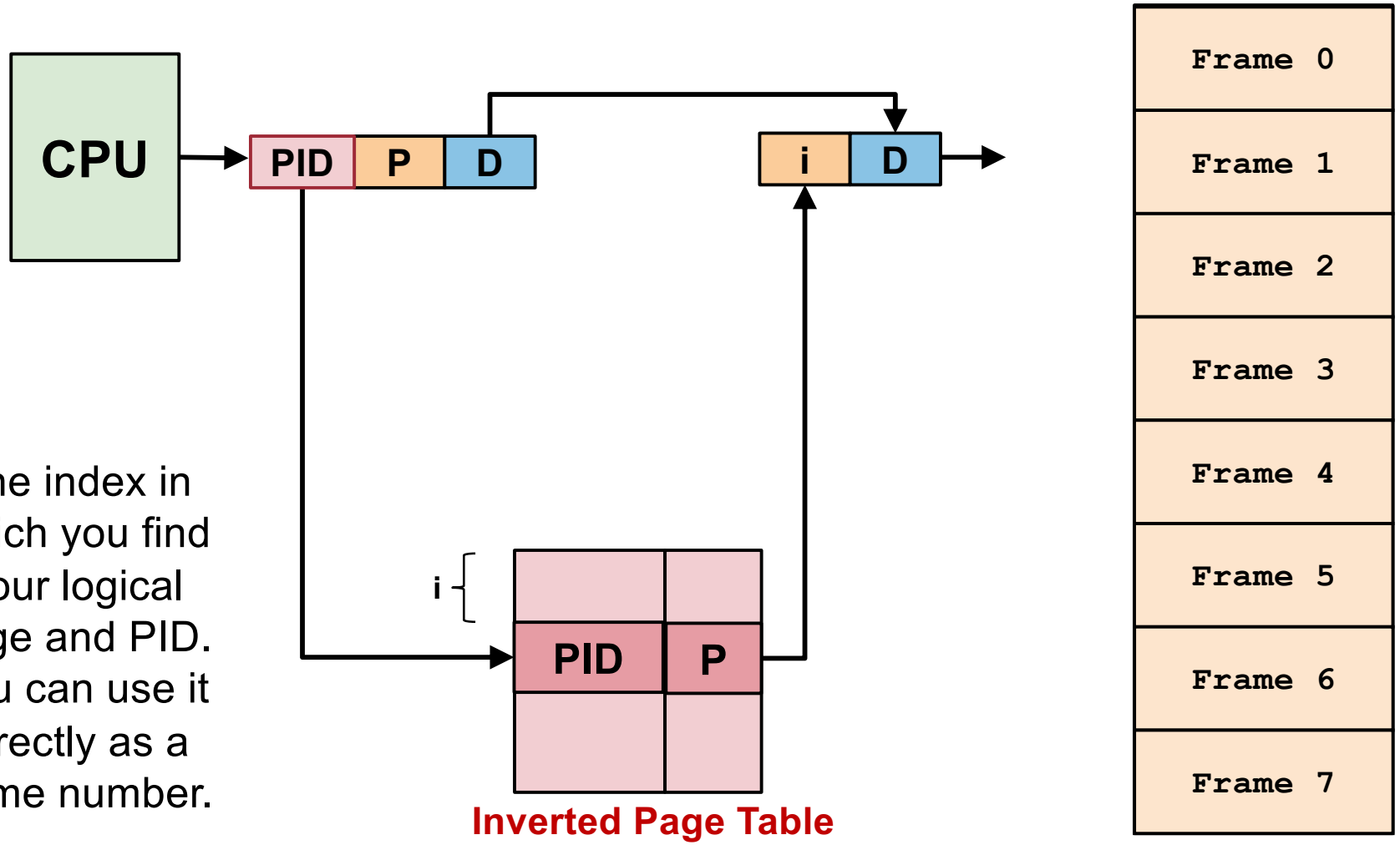
- Yes!

In that case why is inverted table size fixed since each entry can have many processes?

- The main table (what must be pre-allocate)d is fixed
  - Linked list of (logical page, PID) pairs can be appended through *separate chaining*



What is *i* in the inverted table?



The index in which you find your logical page and PID. You can use it directly as a frame number.

Inverted Page Tables

Does inverted page table use more space than hierarchical page table?

- Depends:
  - In an empty system without any processes, it consumes more space
  - In a system with a lot of processes which allocate lots of memory, then the inverted table is smaller.

Inverted Page Tables

It seems like inverted page tables will not work with virtual memory?

- You can extend the inverted page table to cover the swap area too

Inverted Page Tables

Could you please explain how inverted page table is different from normal page table?

- Different purpose
  - Direct tables used for address translation (logical → physical)
  - Inverted table (physical → logical + PID) used for other maintenance tasks
  - However, inverted table can be used to answer logical→ physical too!
    - If a proper hash function is used.
    - You don't have to know that for the exam

Page Replacement Algorithms

Regarding the LRU Implementation details, you mention that it's like taking a plate out of a stack and placing it on top. Do we like keep swapping the plate we want with the one above it until we reach the top or just remove it and place it on the top?

- No, we do not slowly bubble-up a page to the top.
- We just put it directly on top because:
  - it's faster and
  - it's easily doable within a simple linked list

Miscellaneous

Why is page table a contract between hardware and the OS?

- Page Table is a software structure
- However, it is looked up in hardware, e.g., upon TLB misses
- As such, the precise structure of the table must be agreed upon by both hardware and software

Miscellaneous

(Archipelago Qn in Lecture 8 on the number of TLBs in a computer system) Can you explain again ‘why the option 'One for every hardware thread' is wrong?

- It is not entirely wrong
- It is possible to have two TLBs, one for each hardware thread
- it is not necessary, if TLB entries can be distinguished by process ID

Miscellaneous

Are KiB and KB different?

- K is a SI unit prefix in science meaning 1000.
- In computer science, K has been traditionally used to mean 1024.
- To avoid confusion, you should use KiB to mean 1024B.

Miscellaneous

Can you go through Quiz 6 Question 3?

- The question order is randomized, so I don't know what is your question 3.
- Please post on the forum for past quiz question.

Thank you!