CS2106
Introduction to OS

Office Hours, Week 10

Agenda

- General questions
- Paging
- TLBs
- Segmentation
- Virtual Memory

- Mock Midterm Quiz 2 – I'll re-cover it in revision
  - Along with any unanswered questions
  - Feel free to repeat your unanswered question on Archipelago

Cores and Chips

- **What's the difference between processor core and processor chip?**
- **Do we need to know the details for the chips/cores?**
- You **need** to know that today's computers have one or more CPUs.
  - Each CPU is a chip (terms CPU and chip are used interchangeably)
- You **need** to know that most modern CPUs have multiple cores.
- Traditionally, CPUs used to have one core. The only way to have physical parallelism was to have multiple chips, i.e., multiple CPUs. Many OS textbooks were written in this era. Whatever such books say about multiple CPUs, today applies to multiple cores and/or multiple CPUs.
- You **don't need** to know that individual CPU cores may be able to run multiple threads (of the same or different processes) at the same time, if they have two hardware contexts (also knowns as two hardware threads). But it's good to know, as most modern CPUs have this feature.

Slides

- **There was a slide in the lecture that isn't in the uploaded PDF that showed a diagram of the VM + paging illustration.**
- For the virtual memory lecture, I will upload the live slides because they have interesting animations.

Archipelago Quiz

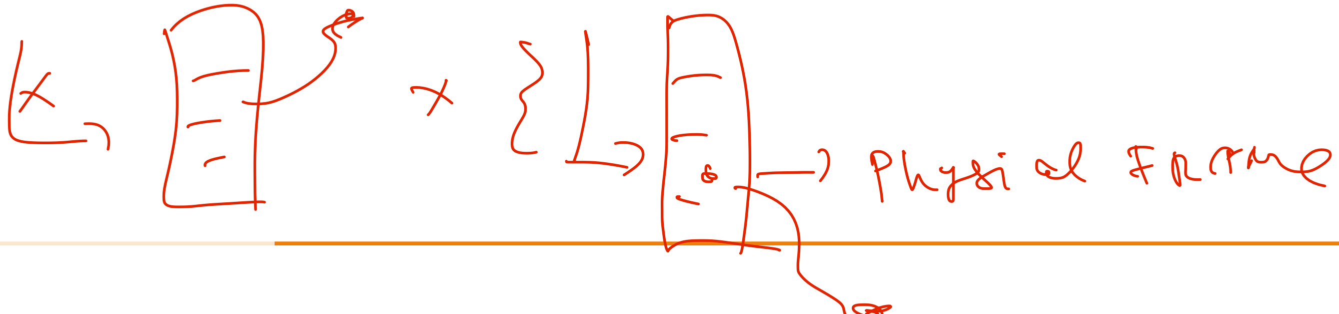
Why is data in the physical or logical memory not saved during context switch?

- There is no need to save the actual data content.
- The data stays where it is, the OS is not going to let another process overwrite it
- Some form of pointers to the data are saved in the PCB

Paging

Is there one page table for one process or one page table for the entire system?

- There is a page table for each process.
- Many processes will have some data at logical page **X**
- The **X-th** entry in their respective page table will give us the actual location (could be anywhere)
- One can ask a different question: which process (or processes, if page is shared) occupies physical frame F?
  - Another structure called inverted page table, one per system, is sometimes used to answer that.



Paging

How does the OS know which page table belongs to which process?

- Every process has a page-table associated with it
- The OS keeps the pointer to the page-table in PCB.

kernel space memory (DRAM)

Paging

✓ ✗

Is the process' page table saved on context switch? Very confused because there was both a tick and a cross on the option.

- In modern systems, page table are not saved in the PCB.
- Just like data content, page table are not moving or going anywhere.
- However, the pointer to the page table is saved in PCB and restored later.

Paging

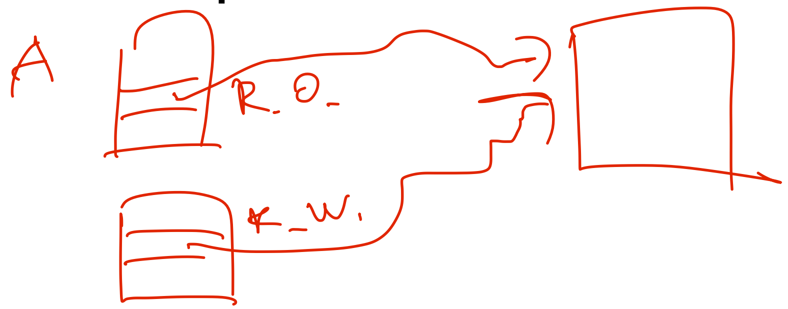
So, memory pages that contain instructions have the access rights the page table set to "executable"? And the data pages in memory have read-write access?

- Correct!
- The access control is done precisely through the page tables!
- Every memory access, be it data or instruction, stack or heap, in cache or not, must go through the page table or TLB.
- Hardware provides the checks and raises exceptions.

Paging

Why does page table entry have information about access right? Isn't access right specific to a process. E.g., Process A can write to page 1 but Process B cannot, How will this info be stored then?

- Page table is the best place to store access rights:
  - Must be accessed (directly or through TLB) upon every access
    - Gives us the mechanism to check every memory access
  - Page tables are private per process, so you can give different permissions to different processes for the same shared page



Paging

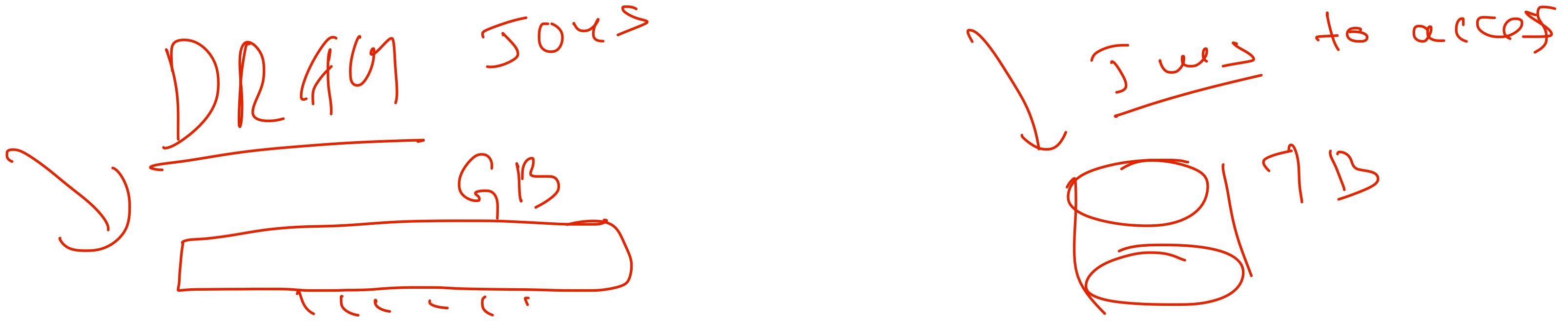
Where is the page table located? It is located in memory?

- Page-tables are located in memory, in the kernel space.
- Some systems allowed page-tables to be on the disk.
- Modern systems (e.g., Linux) keep page-tables always in memory, because they are optimized to be small.

Paging

It seems that each process has a valid bit for each page table entry? This seems like it could be a lot of bits when the number of processes and virtual memory size is large.

- Correct.
- Page tables, as described, are huge.
- On Wednesday we will see how their size can be dramatically reduced on modern systems.



TLBs

How are TLBs different from traditional caches?

- Both have cache-like structure (tags, content, associativity, replacement policy etc)
- Basically, all of them are caches
- However, CPU caches are caching data, or instructions
- TLBs are ***specialized*** to cache page-table entries

TLBs

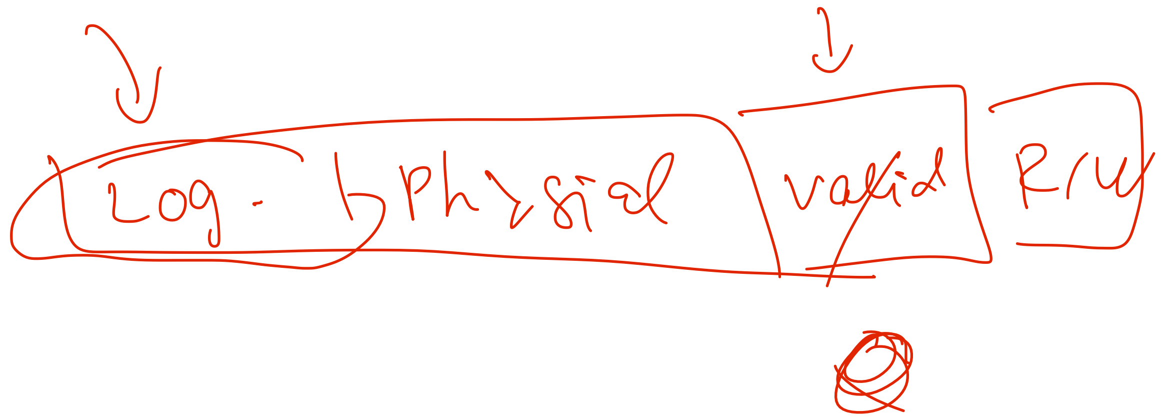
Is it correct to say that TLB and page table are the same just that TLB has faster access?

- Not really.
- TLB caches a number of hot entries from the page table.
- These entries are very fast to access in TLB.

TLBs

What does it mean to invalidate TLB?

- Reset the valid flag of all TLB entries
- Effectively invalidates the TLB content such that the existing mappings cannot be used by a new process



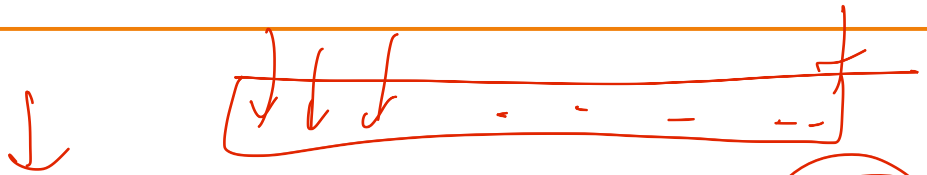
TLBs

16GB V.M.
8GB. phis. mm

For the archipelago question on page faults and fraction of memory accesses, why is it less than 20%? Shouldn't the fraction be 0% since it is already a hit in the TLB which means it should be inside the physical mem?

- The fraction should be 0%.
- One can think of an unlikely case where there is a TLB hit but the page was just evicted. However, most systems would at least invalidate that TLB entry upon eviction, so the TLB hit wouldn't happen.

TLBs



For archipelago question: what if all the 16gb data is only accessed once for the entire program? if so, would TLB not store any of the pages since they are not used after (locality principle)?

- It is possible to traverse your data in such a way that generates no TLB hits
- There will be an optional challenge about it
- And it will be challenging☺
- Note that traversing your data once doesn't mean no TLB hits. Looking at different parts of the same page will result in TLB hits.

Page Walks (TLB miss)

Is accessing the page table to retrieve the mapping done by the OS or the hardware?

- Excellent question!
- The process of traversing the page table and retrieving the entry is called ***page table walk***.
- Page-table walks can be done in **both**:
  - hardware (most of today's systems) or in
  - software, as in some Sun SPARC machines (today's Oracle).
- TLB misses, although less than 1%, are still too frequent
  - Running a software routine every 100 instructions can be slow

Page Walks

If the page table is only accessed by the OS, is the page table located in the kernel memory space?

- Page tables are located in the kernel memory space
- However, the page table is accessed by the kernel **AND** also by the hardware. upon TLB miss
- Hardware is accessing page tables during hardware page-walk.

MMU

ld r1, [1234]

page table?

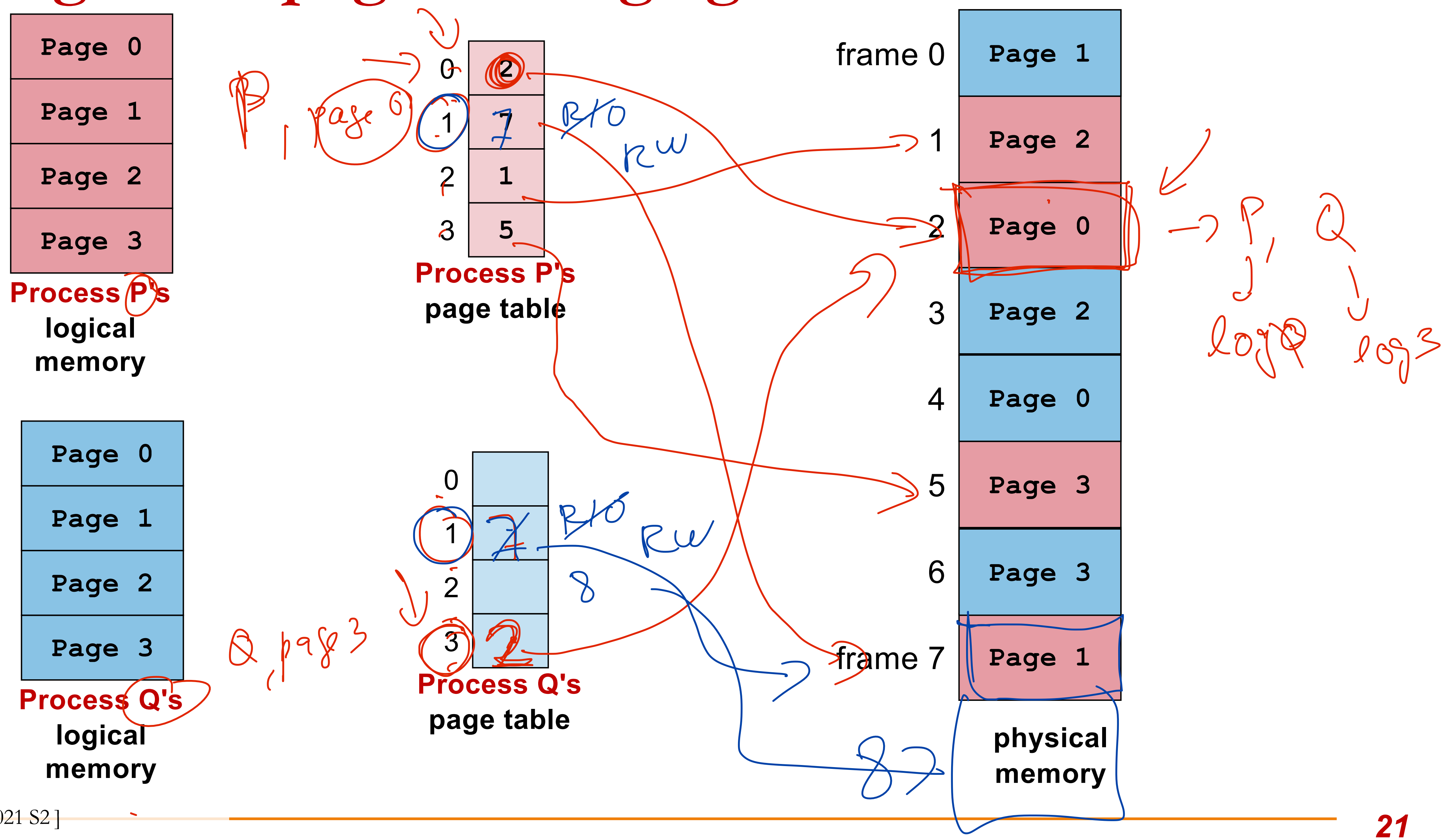
TLB miss

Page Size

Which hardware component is responsible in determining the fixed size of the physical frames in paging?

- The TLB structure must assume a page size.
- Your system may support multiple page sizes, but it must have multiple TLBs, one for each page size
- The page-walking process in hardware also must be aware of the page size.
- Therefore, the CPU designers are determining the page size.

Can you go over page sharing again?



Page Sharing and Copy-on-Write

Page sharing and copy on write: how does a process know whether the page is being shared?

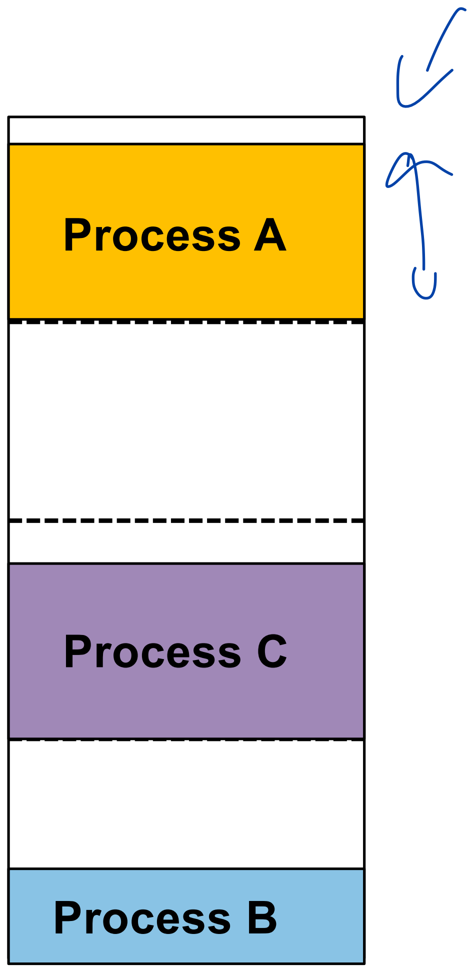
- The OS usually has a separate structure to identify sharers of each physical frame.
- For copy-on-write pages:
  - The OS tags shared pages as read-only
  - Write to them will generate an exception that will be handled by the OS.
  - Then the page will be actually copied and the page tables will be updated.

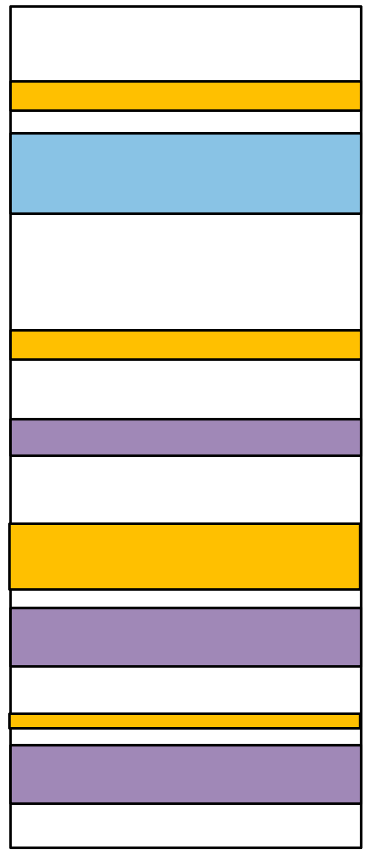
Segmentation

Is segmentation similar to dynamic partitioning?

- Yes.
- Dynamic partitioning: split the memory into contiguous partitions of any size and give a whole contiguous partition to a process.
- Segmentation: further split the process logical memory into partitions of any size, and give the process multiple contiguous regions of physical memory, one for each logical partition.

Dynamic Partitioning (left) vs. Segmentation (right)



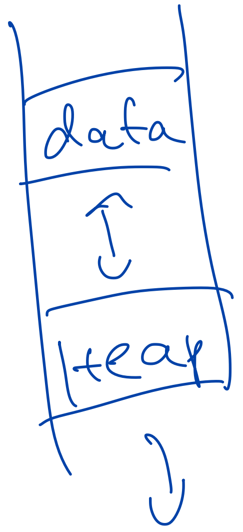


Segmentation

Why do you need a gap between the data and text segments, since both have fixed size?

- We don't!
- But gaps for the heap and ~~data~~ segments are desirable.

stack



Segmentation

If we set a limit for the heap/stack segment, does that mean that is the maximum possible size for heap/stack?

- Yes.
- However, this can be changed later through syscalls.

Segmentation

If there is a segment table per process, does the segment TLB store every segment table of all processes?

- There is one segment table for each process.
- I don’t know of any segment TLB structure.
- (Something similar is now emerging as research prototypes)
- Because there are too few segments, they can be kept in CPU registers and saved and restored upon context switch.
- These registers play the same role that TLB plays in paging

Segmentation

Is Segment table in CPU or memory?

- In theory, segment tables could be in main memory
- In practice, segment tables are so small that they can fit into CPU registers.
- They are saved in PCB and restored during context switch

Is segment table stored in memory or in the kernel space?

- Kernel space is also in memory!
- Segment tables are stored in the CPU because they are small.

Segmentation

It was mentioned that looking up the segment table is done by the hardware instead of the OS. Why is this so and why can't the OS look up the segment table.

- In theory, segment table could be looked up in software
- In practice, I’m not aware of any architecture that does that.
- A process usually has very few segments and their beginning and size can be kept in the CPU registers, which are very fast to access.

Memory or OS Context?

Are page tables and segment tables part of the OS context of a process? Or are they part of the memory context of a process?

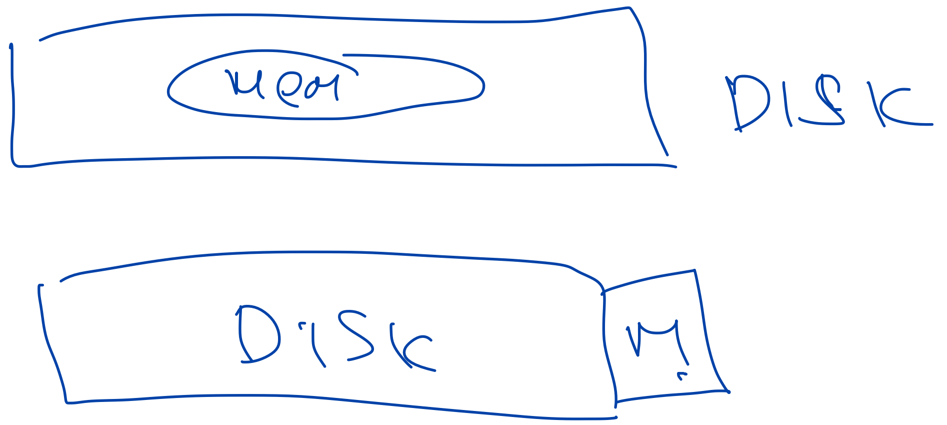
- Great question!
- One could also argue that it’s part of the hardware context
  - The pointer to the page table is stored in CPU registers
- It’s a matter of a definition
- In practice, it doesn’t matter what you call it

Virtual Memory

- **In virtual memory with paging, is the secondary storage the authoritative memory (contains everything) or does it only contain the memory that are not in the physical memory?**

- **When the page is not in the memory resident, it will be loaded from the secondary storage. Is the data in the secondary storage removed when this data is loaded to the memory, or does it stay there?**

- **Excellent questions**



Virtual Memory

- Physical memory is a cache, with hot pages residing there, while cold pages being on in the secondary storage.
- However, there are two types of cache inclusion policies:
  - **Inclusive policy**: the content of the layer below (storage) is a superset of the content in the cache (physical memory)
  - **Exclusive policy**: the contents in the cache and the level below are disjoint
- Both approaches are possible and there's a trade-off:
  - If the cache is inclusive, there is a waste of storage space
    - However, an unmodified page evicted from memory!
  - Exclusive caches save storage space
    - but must always write the page back to the disk upon eviction, even if it's unmodified (aka ***clean***).
- Caching policies that are not strictly inclusive but also not strictly exclusive, are often called non-inclusive

$\boxed{\text{dirty}}$

Virtual Memory vs. Swapping

Is the concept of virtual memory the same as swapping?

- Related but not the same.
- Virtual memory is an abstraction of a huge logical address space.
- Swapping is the process of moving a page from storage to memory and back.
- We say that a page is **swapped out** when we send it to disk
- We say that a page is **swapped in** when it's brought to memory
- In Linux, the part of the storage space allocated for swapped out pages is called **swap**.

Virtual Memory

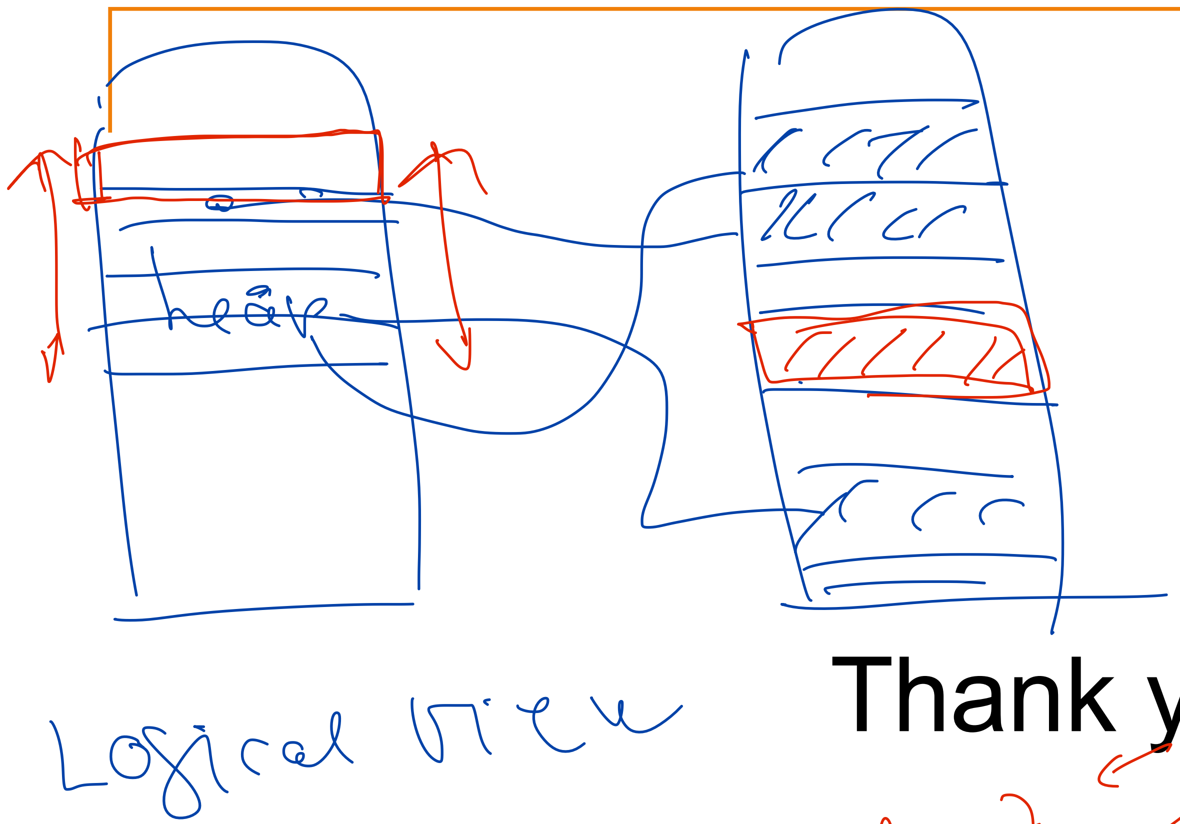
How to determine if a page has higher priority and should be stored in the physical memory?

- Through various page-replacement algorithms and frame allocation policies.
- We will cover this on Wednesday.

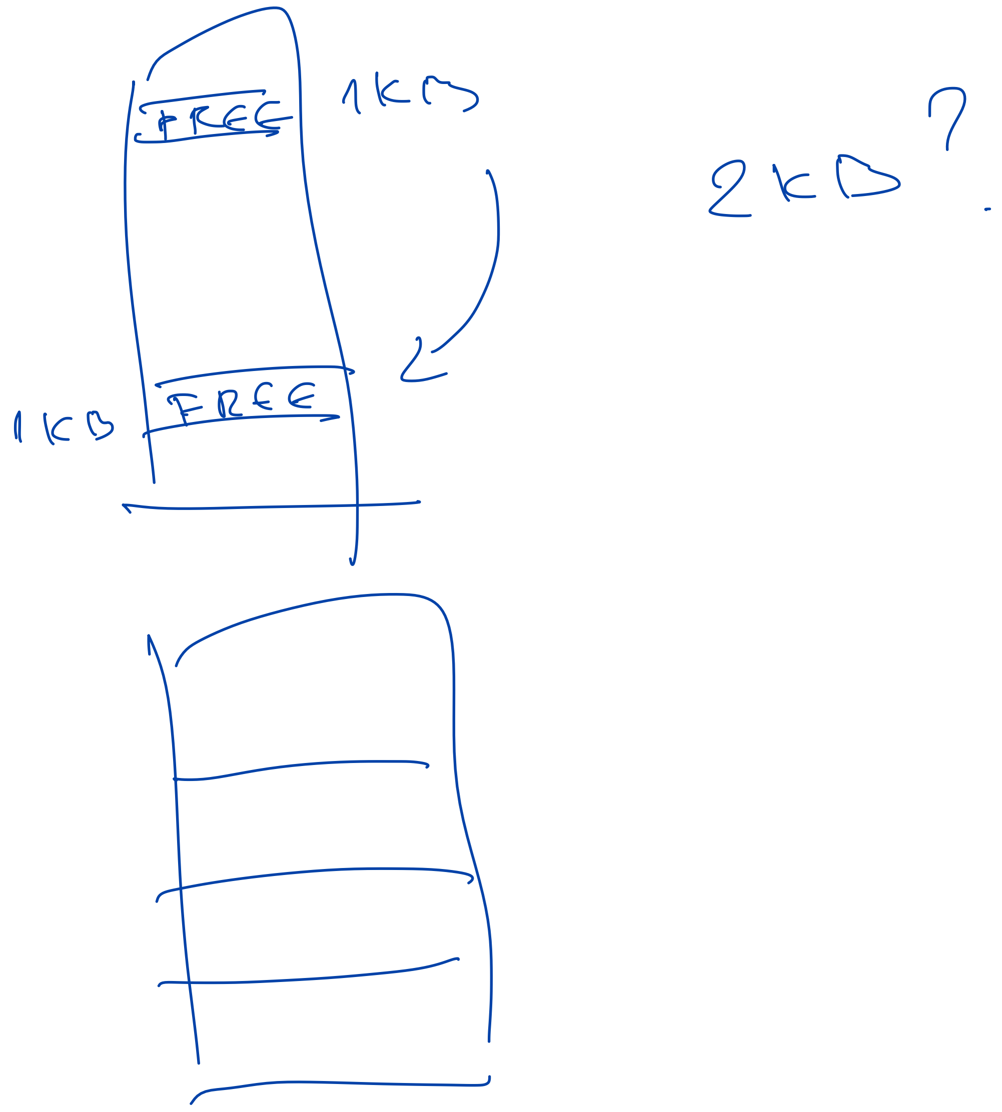
Virtual Memory (non-examinable)

For software like Redis, they claim that they do in-memory storage. Does it mean they don't use the disk at all, and only use DRAM?

- Correct.
- In-memory databases and data storage engines simply use huge amounts of DRAM and store all the data in there.



Thank you!


FREE
1KB
FREE
1KB
2KB ?