Assignment 6

1. Cooperation between hardware and OS

Address translation

Transforms each memory access , changing the **virtual** address provided by the instruction to a **physical** address where the desired information is actually located.

Hardware Support

hardware requirements	notes
Privileged mode	prevent user-mode processes execute privileged operations
Base and bounds	support address translation and check boundary
Privileged instruction(s) to update base/bounds	OS can set these values before running the program
exception handlers	OS can raise a exception code to hardware
Ability to raise exceptions	Some user processes try to access privileged instructions or out-of-bounds address access

OS Support

OS Requirement	Notes
Memory management	allocate memory reclaim memory
Base/bounds management	set base/bounds properly upon context switch
Exception handling	
Exception raising	

Cooperation: Boot

OS (kernel mode)	Hardware
initialize trap table	
	remember addressed of: syscall handler timer handler illegal memory-access and instrucion
start interrupt timer	
	start timer; interrupt
initialize process table	
initialize free list	

Cooperation: Runtime

OS (kernel mode)	Hardware	Program (user mode)
To start process A: allocate entry in process table malloc for process set base/bound registers return-from-trap (into A)		
	restore registers of A move to user mode jump to A's PC	
		Process A runs Fetch

OS (kernel mode)	Hardware	Program (user mode)
	translate virtual address do fetch	
		Execute load/store
	ensure address is legal by base and bound translate virtual address do instruction	
		A runs...
	Timer interrupt move to kernel mode jump to handler	
Handler timer context switch run B		
	restore registers of B move to user mode jump to B's PC	
		Process B runs execute bad load
	Load is out-of-bound exception move to kernel mode jump to trap handler	
Handle the trap kill B reclaim B's memory free B's entry in process table		

2. differences between segmentation and paging

size of chunks

- Segmentation divides the virtual address space into variable-sized segments, where each segment corresponds to a logical unit of the program (e.g., code, data, stack).
- Paging divides the virtual address space into fixed-sized pages, typically ranging from 4KB to 2MB.

management of free space

- Segmentation
    - The space that this library manages is known historically as the heap, and the generic data structure used to manage free space in the heap is some kind of **free list**. This structure contains references to all of the free chunks of space in the managed region of memory.
    - Not regular: we have a number of segments per process, and each segment might be a different size, then there are many free space which have different size.
    - malloc strategies: best-fit, worst-fit, first-fit, buddy algorithm, etc.
    - issue: easy to waste many space because of external
- Paging
    - Also use free list to manage free space.
    - Regular: Paging divides the virtual address space into fixed-sized page
    - Flexibility: with a fully-developed paging approach, the system will be able to support the abstraction of an address space effectively, regardless of how a process uses the address space; we won't, for example, make assumptions about the direction the heap and stack grow and how they are used.
    - simplicity: For example, when the OS wishes to place our tiny 64-byte address space into our eight-page physical memory, it simply finds four free pages; perhaps the OS keeps a free list of all free pages for this, and just grabs the first four free pages off of this list.

context switch overhead

- Segmentation requires saving and restoring the pointers, bases and bounds of the heap, code and stack, which can be expensive in terms of context switch overhead.
- Paging requires only saving and restoring page table pointers, which is much faster.

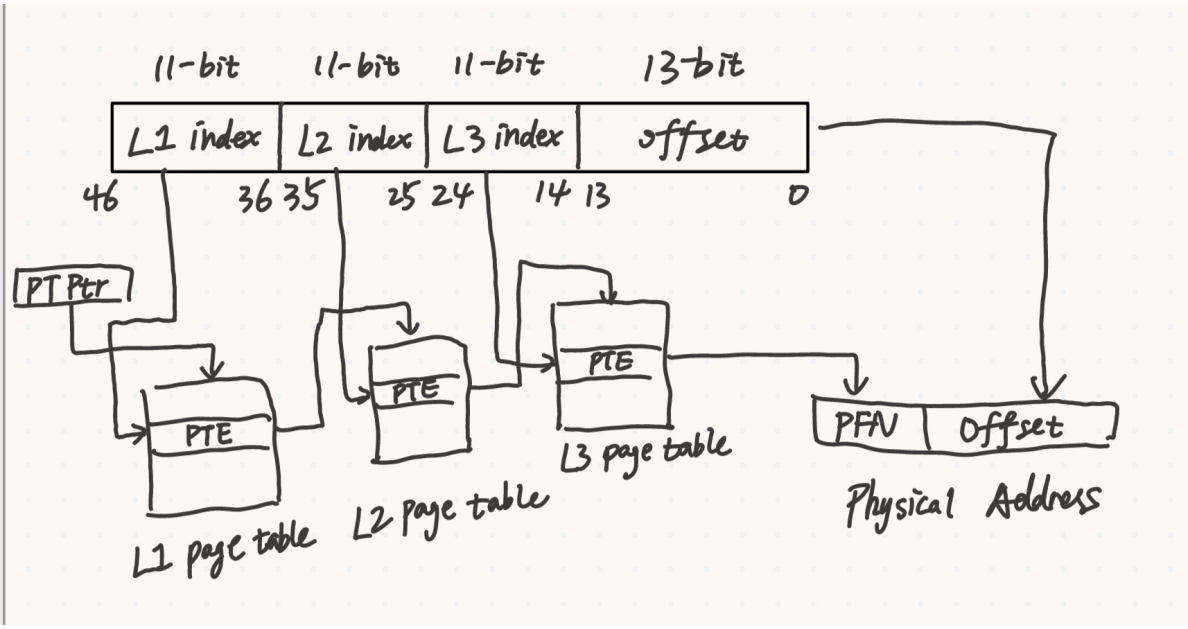
fragmentation

- Segmentation:
    - suffers from external fragmentation, which can lead to a significant amount of memory waste.
    - remedies: compact physical memory by rearranging the existing segments.
- Paging suffers from internal fragmentation, which can also waste some memory but is typically less severe than external fragmentation.

status bits and protect bits

- Segmentation and paging requires a status bit for each page to indicate whether it is currently in memory or not. It also requires a protection bit for each page to control access rights.

3.



Because the Page table entry size is 4 Bytes and page size is 8 KBytes, then the number of PTEs in one page table is 8KB / 4Byte = 2^{11}. Hence, we need 33 / 11 = 3 levels.

4.

(a) page size: 2^{12}=4KB. maximum page table size: 2^{20}=1MB

(b)

- 780, 770
- 614, 1707