Lab Assignment 3: Printing Page Table

Task: Write a function that prints the contents of page table of a process

Define a function called **vmprint().** It should take a **pagetable_t** argument and print that page table in the format described below.

Insert **vmprint(p->pagetable)** in exec.c just before the **return argc**, to print the process's page table of your interest.

Now when you start xv6, it will print output something like this, describing the page table of the first process **init**:

```
page table 0x0000000087f6b000
 ..0: pte 0x0000000021fd9c01 pa 0x0000000087f67000
 .. ..0: pte 0x0000000021fd9801 pa 0x0000000087f66000
 .. .. ..0: pte 0x0000000021fda01b pa 0x0000000087f68000
 .. .. ..1: pte 0x0000000021fd9417 pa 0x0000000087f65000
 .. .. ..2: pte 0x0000000021fd9007 pa 0x0000000087f64000
 .. .. ..3: pte 0x0000000021fd8c17 pa 0x0000000087f63000
 ..255: pte 0x0000000021fda801 pa 0x0000000087f6a000
 .. ..511: pte 0x0000000021fda401 pa 0x0000000087f69000
 .. .. ..509: pte 0x0000000021fdcc13 pa 0x0000000087f73000
 .. .. ..510: pte 0x0000000021fdd007 pa 0x0000000087f74000
 .. .. ..511: pte 0x0000000020001c0b pa 0x0000000080007000
init: starting sh
```

The first line displays the argument to **vmprint(),** which is the physical address of the page frame holding the outermost (level-1) page table of the process. After that, there is a line for each PTE, including PTEs that refer to page-table pages deeper in the paging tree of 3 levels. Each PTE line is indented by a number of " .." that indicates its depth (level) in the tree. Each PTE line shows the PTE index like 0, 1, 2, …., 511 in its page-table page, the pte bits, and the physical address extracted from the PTE. Don't print PTEs that are not valid.

In the above example, the top-level (Level-1) page-table page has mappings for entries 0 and 255. The next level down for entry 0 has only index 0 mapped, and the bottom-level for that index 0 has entries 0, 1, 2, and 3 mapped.

Note: Your code might emit different physical addresses than those shown above. But the number of entries and the virtual addresses should be the same for init process.

Hints:

- Write vmprint() in kernel/vm.c.
- Use the macros at the end of the file kernel/riscv.h.
- Define the prototype for **vmprint()** in kernel/defs.h so that you can call it from exec.c
- Use %p in your printf calls to print out full 64-bit hex PTEs and physical addresses as shown in the example.

Answer the following questions:

Q1. Print the page table of init process
Q2. Print the page table of sh process
Q3. Print the page table of user-level sleep process (refer to Lab Assignment 2)
Q4. What is the size of the page frame, number of entries in a page table and address bits of virtual address space and physical address space in xv6?
Q5. How does xv6 allocate bits of virtual address space for multi-level paging and offsetting?
Q6. Given a pte, how do you determine whether it's valid (present) or not and how do you determine page frame number in pte is of next level page table or user process'? List out some valid pte entries from one of the Q1/Q2/Q3 which fall under above two categories.
Q7. By referring to the pagetables of init/sh/sleep processes, fill out the following table. Please explain your response as remarks, in brief.

	init process	**sh process**	**sleep process**	**Remarks**
No. of page frames consumed by the page table				
Internal fragmentation in the page table(s) in bytes				
No. of page frames allocated for the process				
No. of page frames allocated for TEXT segment of the process and their physical addresses				
No. of page frames allocated for Data/Stack/Heap segments of the process				

and their physical addresses				
Any dirty pages?				
Any kernel mode (controlled) page frames?				

Deliverables:

- **Submit a report with your solution methodology and screenshots of outputs. Also, answer the questions asked above.**
- **Submit the source codes of files modified with proper documentation.**

Zip all the files and name it as LA3-<rollno>.zip. Then, upload it on the Google Classroom.

Please show at least one output (Q1/Q2/Q3) to the TAs before leaving the lab session.

Due by 11:59 PM today. Late submissions attract a penalty of 25% per day!

References:

- https://pdos.csail.mit.edu/6.828/2023/labs/pgtbl.html
- https://pdos.csail.mit.edu/6.828/2023/xv6/book-riscv-rev3.pdf