Operating Systems Practice

Project #2 – 3-level paging

Seonggyun Oh

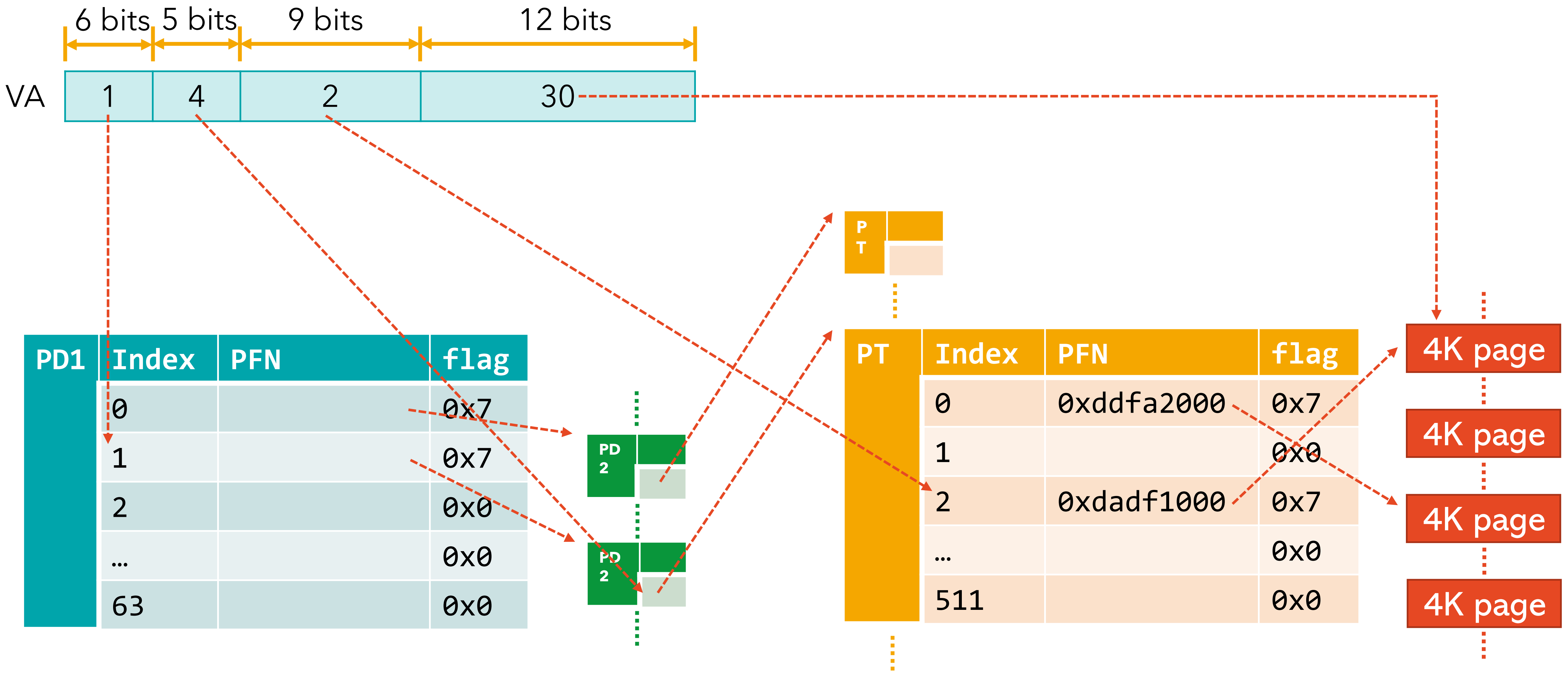
(sungkyun123@dgist.ac.kr)

Before we start…

- Clean-up your Git repository for this project
  - `git fetch && git reset --hard && git clean -fdx && git checkout page-counter`
    - Your modifications will be deleted with this command!

3-Level Paging

- 3-level paging

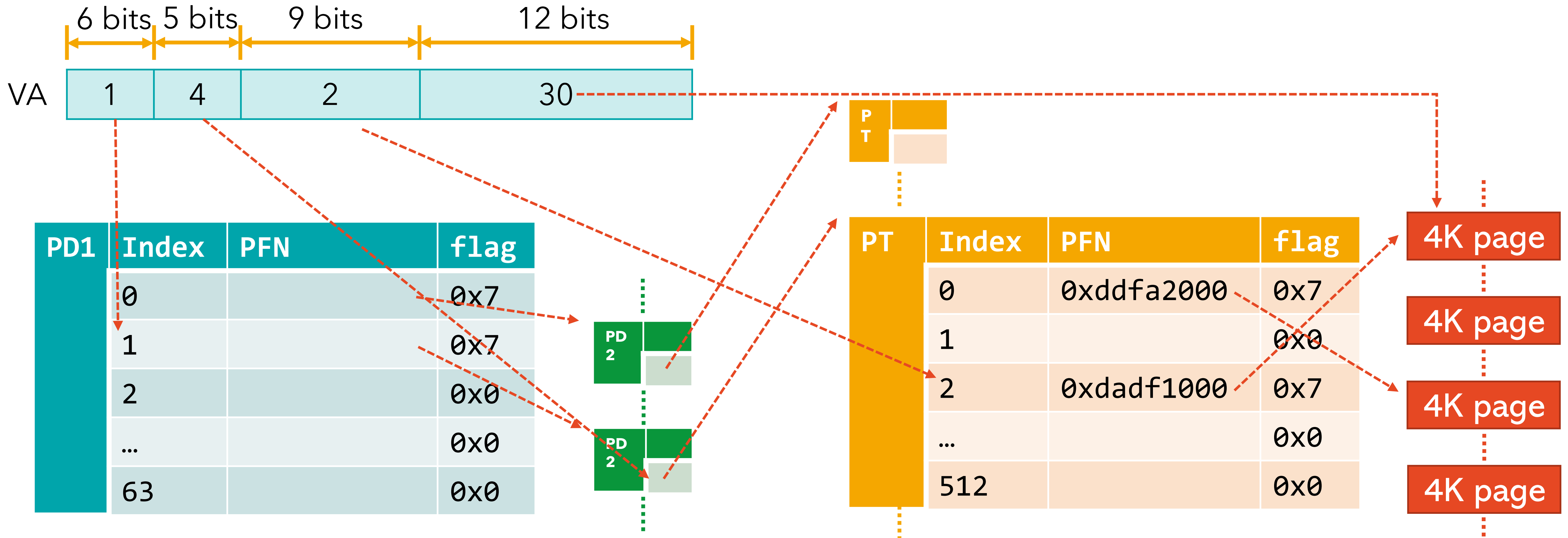


PD1	Index	PFN	flag
	0		0x7
	1		0x7
	2		0x0
	…		0x0
	63		0x0

PT	Index	PFN	flag
	0	0xddfa2000	0x7
	1		0x0
	2	0xdadf1000	0x7
	…		0x0
	511		0x0

Project #2 – Implementing 3-Level Paging

- *pgdir1*: use upper 6 bits in VA (64 entries)
- *pgdir2*: use 5 bits after 6 bits of pgdir in VA (32 entries)
- *pgtab*: use 9 bits after 5 bits of pgdir2 (512 entries)

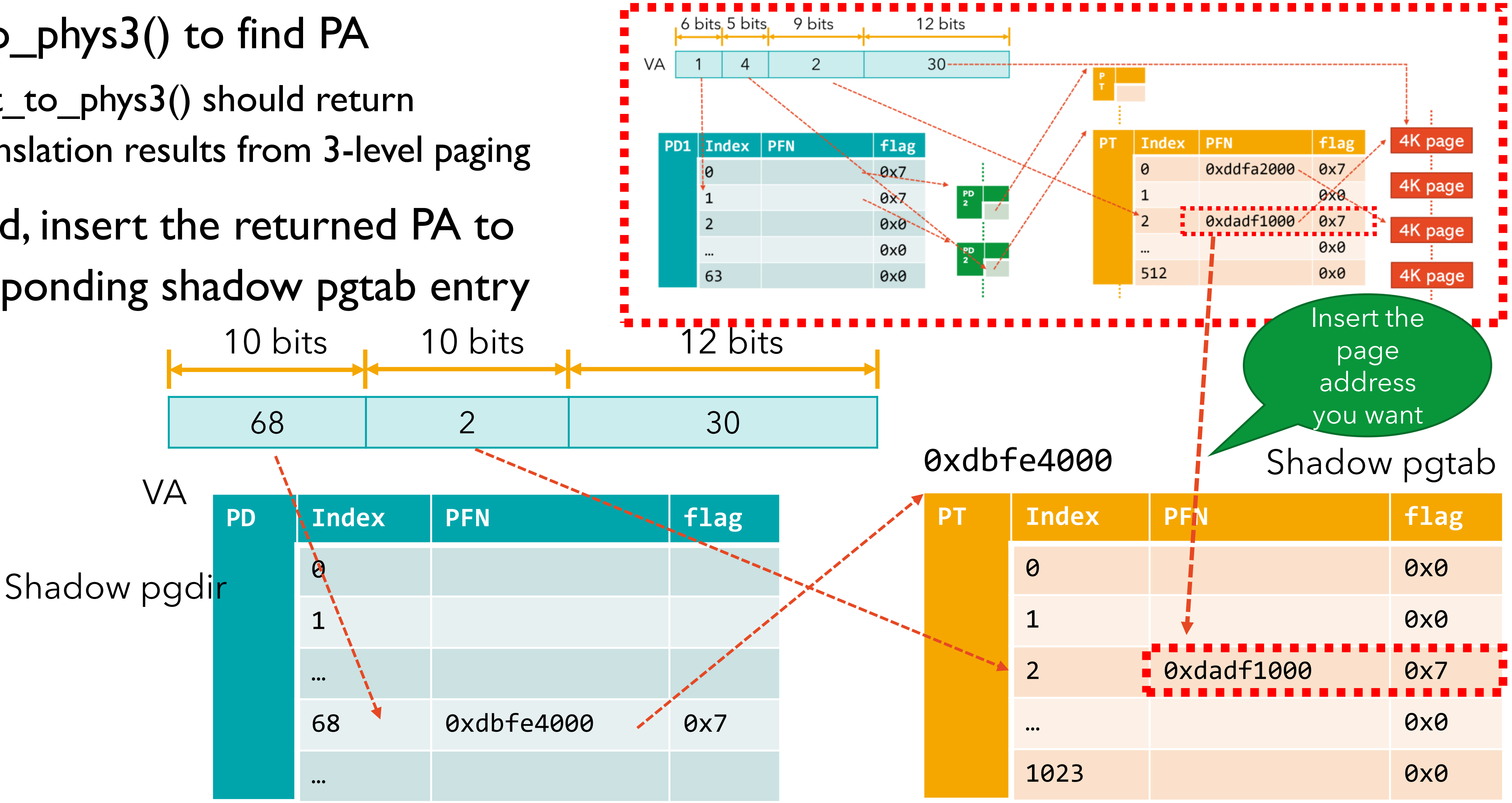


PD1 Index	PFN	flag
0		0x7
1		0x7
2		0x0
…		0x0
63		0x0

PT Index	PFN	flag
0	0xddfa2000	0x7
1		0x0
2	0xdadf1000	0x7
…		0x0
512		0x0

Project #2 – Implementing 3-Level Paging

- If page fault occurs, first call virt_to_phys3() to find PA
  - virt_to_phys3() should return translation results from 3-level paging
- If found, insert the returned PA to corresponding shadow pgtab entry



PD1	Index	PFN	flag
	0		0x7
	1		0x7
	2		0x0
	…		0x0
	63		0x0

PT	Index	PFN	flag
	0	0xddfa2000	0x7
	1		0x0
	2	0xdadf1000	0x7
	…		0x0
	512		0x0

PD	Index	PFN	flag
	0		
	1		
	…		
	68	0xdbfe4000	0x7
	…		

PT	Index	PFN	flag
	0		0x0
	1		0x0
	2	0xdadf1000	0x7
	…		0x0
	1023		0x0

Project #2 – Implementing 3-Level Paging

- **You should modify allocation, deallocation, traversal code to support 3-level paging in context of user's process**
  - walkpgdir(), freevm(), deallocuvm(), mmu.h
- **Use page-fault concept (software TLB) to support 3-level paging**
  - Modify __virt_to_phys3() and pagefault()
- **To evaluate 3-level paging, you should run usertests and memtest**
  - Write the code to output the structure of virtual memory in printvm()
  - Write the code to output the page access count when the process terminates using the leftover space in page directory or page table
- Fill in the code wherever it contains comments written in *//TODO*

Project #2 – Before starting…

- **Because we only consider user processes, you should separate 2-level allocation and 3-level allocation**
  - Kernel's memory paging should be still done in 2-level
  - For 2-level allocation, you should replicate the walkpgdir to k_walkpgdir()
    - shadow_pgdir, kpgdir uses that function (k_walkpgdir())
    - walkpgdir() is only for 3-level allocation and search
  - To do so, we modify setupkvm() so that it can distinguish whether the caller is a user process or the kernel
    - setupkvm(1) should call k_walkpgdir()
    - setupkvm(0) should call walkpgdir()

Project #2 – Modifying allocating part

- **In mmu.h(), you should add various macros to support 3-level paging**
  - For example, if you extract the index of first page directory entry, make PD1X macro that extract the value of upper 6 bits of VA (To do so, the VA must be shifted to 26)
  - Hint
    - Add PD1X, PD2X, PTNX, PG3ADDR (or PG1ADDR and PG2ADDR), PD1XSHIFT, PD2XSHIFT, PTNXSHIFT

- **In walkpgdir(), you should modify 2-level walking to 3-level walking**
  - Rename original pgdir to pgdir1 and add pgdir2 to 3-level
  - pgdir2 should be allocated if it is not allocated yet (like the existing pgtab)
  - Return value should be the same as original

Project #2 – Modifying allocating and traversing part

- **To track the page access count, you should use margin space in page table**
  - We use 9 bits in VA for page table indexing (# of entries per page table: 512 (2^9))
  - xv6 allocate 4 KB pages for page table (# of max entries: 1024 (4KB/4B (sizeof int)))
  - You can use the remaining space (the latter 512 entries) to store page access count
  - When a new page address is faulted (i.e., accessed), increase the count
  - Hint: Write the code in walkpgdir()

PT	Index	PFN	flag
	0	0xddfa2000	0x7
	1		0x0
	…		0x0
	511		0x0
	512		
	513		
	…		
	1023		

Unused!

Project #2 – Modifying deallocating part

- **In freevm(), you should modify 2-level free to 3-level free**
  - freevm() function frees all allocated page table and page directory
  - You should free the (new) second page directory too

- **In deallocuvm(), you should consider the second page directory**
  - deallocuvm() function frees allocated pages by walking the page table
  - Originally, the function only considers the first page directory (refer to PGADDR)
  - PGADDR macro changes the address of input page directory entry into the address of next page directory entry
  - PGADDR only covers 2-level paging, so, you add new macros to support 3-level paging
    - PG3ADDR (or use PG1ADDR and PG2ADDR)
  - If needed, you can make new functions to use in deallocuvm()
  - Hint: You must consider bit-overflow when you traverse valid page to free
    - Otherwise, deallocuvm() will fall into an endless loop

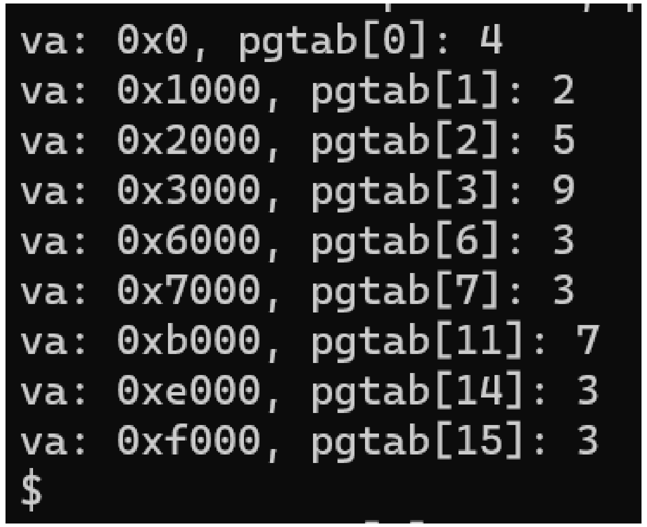
Project #2 – Converting VA to PA using Software TLB

- The page fault handler calls virt_to_phys3() function to convert VA to PA that is mapped to 3-level paging structure
- You should modify the __virt_to_phys3() function to support 3-level paging
- You should modify the pagefault() function to support 3-level paging

Project #2 – Evaluating 3-Level Paging

- Make sure that the shadow page table (2-level) returns the same PA as your own 3-level page table
  - virt_to_phys3(proc->pgdir) should return the same value as virt_to_phys2(proc->shadow_pgdir)!

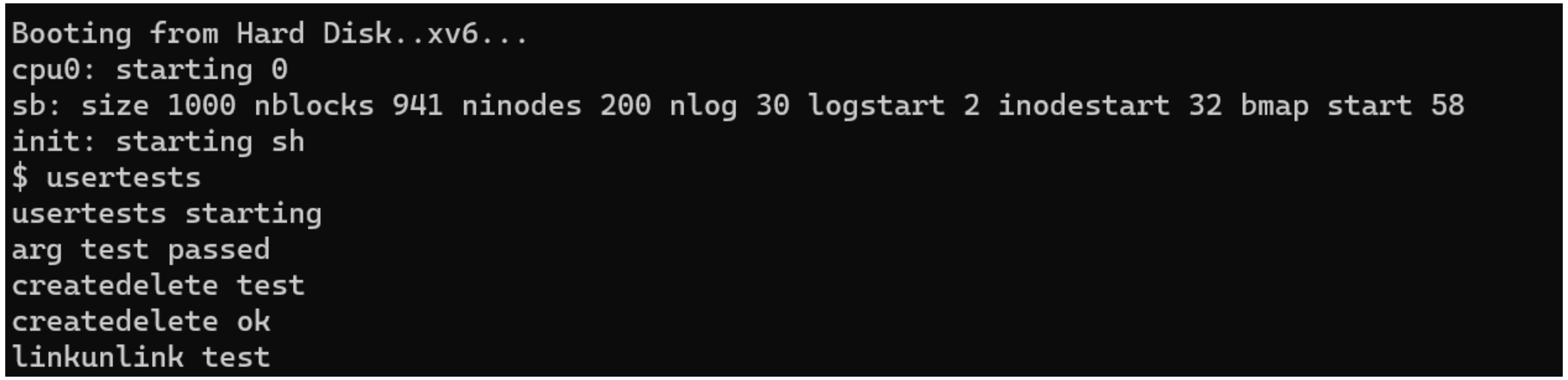
- Print the page access count when the virtual memory of process is freed
  - Implement the code in freevm() function
  - Output



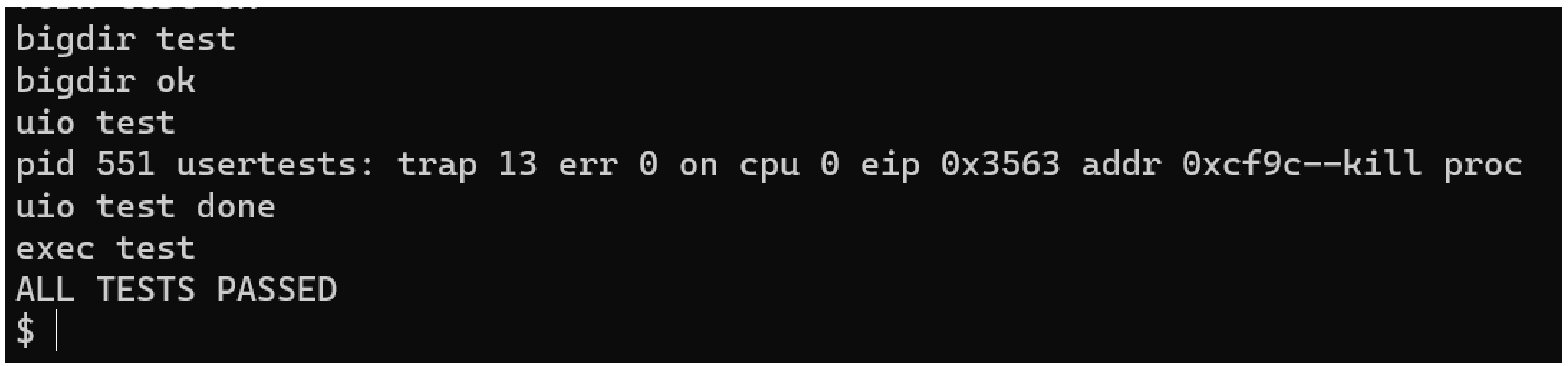
```
va: 0x0, pgtab[0]: 4
va: 0x1000, pgtab[1]: 2
va: 0x2000, pgtab[2]: 5
va: 0x3000, pgtab[3]: 9
va: 0x6000, pgtab[6]: 3
va: 0x7000, pgtab[7]: 3
va: 0xb000, pgtab[11]: 7
va: 0xe000, pgtab[14]: 3
va: 0xf000, pgtab[15]: 3
$
```

Project #2 – Evaluating 3-Level Paging

- To evaluate your implementation of 3-level paging, perform usertests
  - If user code may be correct, "ALL TESTS PASSED" will be printed
  - Output



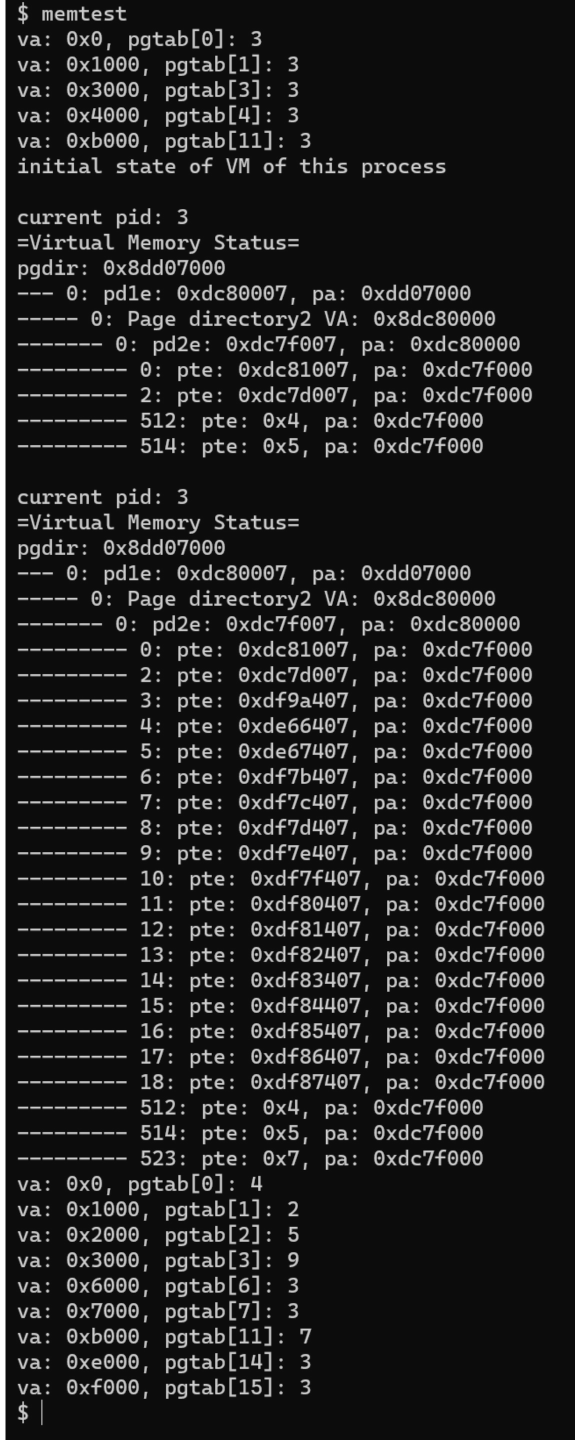
```
Booting from Hard Disk..xv6...
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ usertests
usertests starting
arg test passed
createdelete test
createdelete ok
linkunlink test
```



```
bigdir test
bigdir ok
uio test
pid 551 usertests: trap 13 err 0 on cpu 0 eip 0x3563 addr 0xcf9c--kill proc
uio test done
exec test
ALL TESTS PASSED
$ 
```

Project #2 – Evaluating 3-Level Paging

- Perform memtest
  - You must modify printvm() function in vm.c to enable to print the information of 3-level
  - You should explain the result in report
  - Output (The output may slightly differ)



```
$ memtest
va: 0x0, pgtab[0]: 3
va: 0x1000, pgtab[1]: 3
va: 0x3000, pgtab[3]: 3
va: 0x4000, pgtab[4]: 3
va: 0xb000, pgtab[11]: 3
initial state of VM of this process

current pid: 3
=Virtual Memory Status=
pgdir: 0x8dd07000
--- 0: pd1e: 0xdc80007, pa: 0xdd07000
----- 0: Page directory2 VA: 0x8dc80000
------- 0: pd2e: 0xdc7f007, pa: 0xdc80000
--------- 0: pte: 0xdc81007, pa: 0xdc7f000
--------- 2: pte: 0xdc7d007, pa: 0xdc7f000
--------- 512: pte: 0x4, pa: 0xdc7f000
--------- 514: pte: 0x5, pa: 0xdc7f000

current pid: 3
=Virtual Memory Status=
pgdir: 0x8dd07000
--- 0: pd1e: 0xdc80007, pa: 0xdd07000
----- 0: Page directory2 VA: 0x8dc80000
------- 0: pd2e: 0xdc7f007, pa: 0xdc80000
--------- 0: pte: 0xdc81007, pa: 0xdc7f000
--------- 2: pte: 0xdc7d007, pa: 0xdc7f000
--------- 3: pte: 0xdf9a407, pa: 0xdc7f000
--------- 4: pte: 0xde66407, pa: 0xdc7f000
--------- 5: pte: 0xde67407, pa: 0xdc7f000
--------- 6: pte: 0xdf7b407, pa: 0xdc7f000
--------- 7: pte: 0xdf7c407, pa: 0xdc7f000
--------- 8: pte: 0xdf7d407, pa: 0xdc7f000
--------- 9: pte: 0xdf7e407, pa: 0xdc7f000
--------- 10: pte: 0xdf7f407, pa: 0xdc7f000
--------- 11: pte: 0xdf80407, pa: 0xdc7f000
--------- 12: pte: 0xdf81407, pa: 0xdc7f000
--------- 13: pte: 0xdf82407, pa: 0xdc7f000
--------- 14: pte: 0xdf83407, pa: 0xdc7f000
--------- 15: pte: 0xdf84407, pa: 0xdc7f000
--------- 16: pte: 0xdf85407, pa: 0xdc7f000
--------- 17: pte: 0xdf86407, pa: 0xdc7f000
--------- 18: pte: 0xdf87407, pa: 0xdc7f000
--------- 512: pte: 0x4, pa: 0xdc7f000
--------- 514: pte: 0x5, pa: 0xdc7f000
--------- 523: pte: 0x7, pa: 0xdc7f000
va: 0x0, pgtab[0]: 4
va: 0x1000, pgtab[1]: 2
va: 0x2000, pgtab[2]: 5
va: 0x3000, pgtab[3]: 9
va: 0x6000, pgtab[6]: 3
va: 0x7000, pgtab[7]: 3
va: 0xb000, pgtab[11]: 7
va: 0xe000, pgtab[14]: 3
va: 0xf000, pgtab[15]: 3
$ 
```

Project #2 – 3-Level Paging

- Deadline
  - ~ 2022.11.23 (Wed) 23:59
- Hand-in procedure
  - p2_201812345.patch
  - Run the following command and upload p2_201812345.patch
    - git diff > p2_201812345.patch
  - Check the patch file with Notepad and confirm your modifications are in the patch file
  - Report
    - Submit an 1~3 pages report
      - Free format (Korean/English)
      - Description of your implementation in detail (walkpgdir, deallocuvm, freevm)
      - Explain your test code and answer the following two tests and explain why the results are as follows
      - Insert test code result image

Extra points

- If you found a mistake in the skeleton code
- If you found a different/better method
- Kernel memory leak test has been disabled in usertests
  - https://github.com/dgist-datalab/xv6/commit/63776c9cc
  - If you manage to pass this test (after reverting the commit)

- Write a report for extra points

Finally …

<u>**Do NOT hesitate**</u> to ask questions!

Mini Project #1, #2	Juhyung Park	*arter97@dgist.ac.kr*
Project #1	Minjae Kim	*jwya2149@dgist.ac.kr*
Project #2	Seonggyun Oh	*sungkyun123@dgist.ac.kr*
Project #3	Soyoung Han	*zsally@dgist.ac.kr*

Thank You!