

CS444/544

Operating Systems II

Virtual Memory Layout

Yeongjin Jang

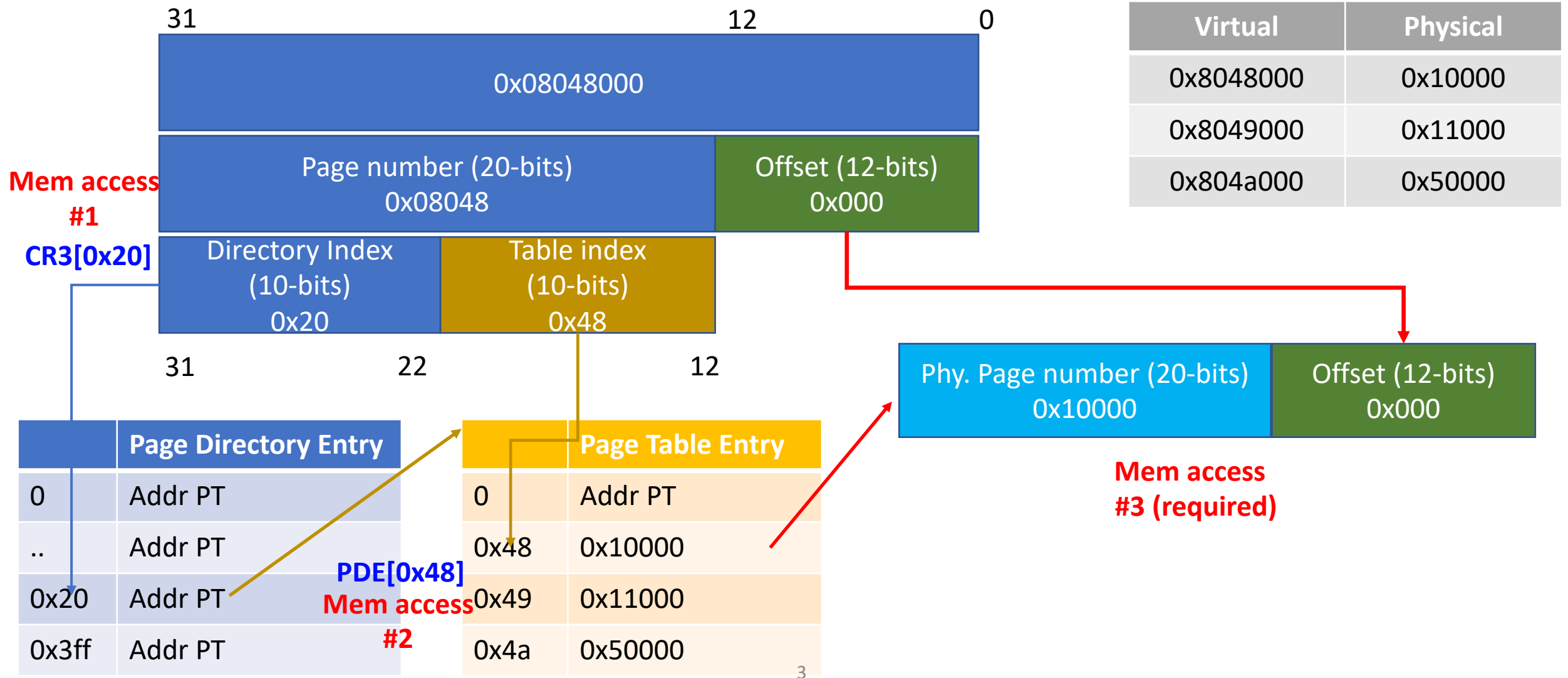


Oregon State
University

Lab 1 Due Passed

- Late submissions: Tokens will be deducted automatically from your token jar based on the tagged commit/push time in the repository
- Please tag your commit accurately
 - Tag name: lab1-final
 - How to delete local/remote tag: <https://devconnected.com/how-to-delete-local-and-remote-tags-on-git/>
- Extra credits for challenge
 - Printing 5 colors and getting back to the original color: +1%

Recap – Page Table & Addr Translation

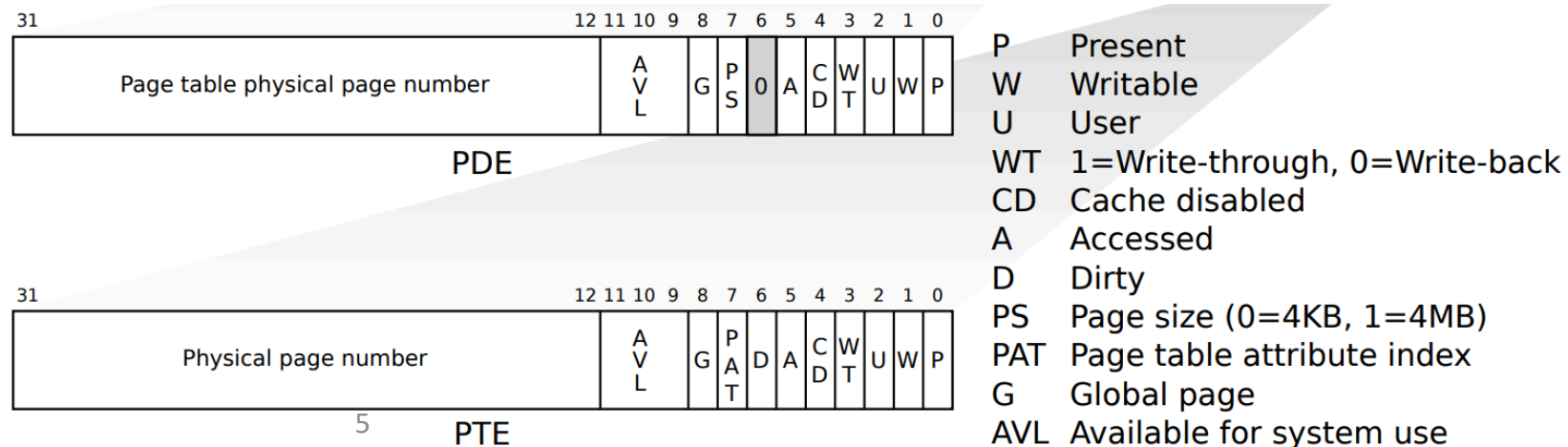


Today's Topic

- Page Permissions
- Virtual Memory Layout
- How JOS Manages Physical Memory?

Page Directory / Table Entry (PDE/PTE)

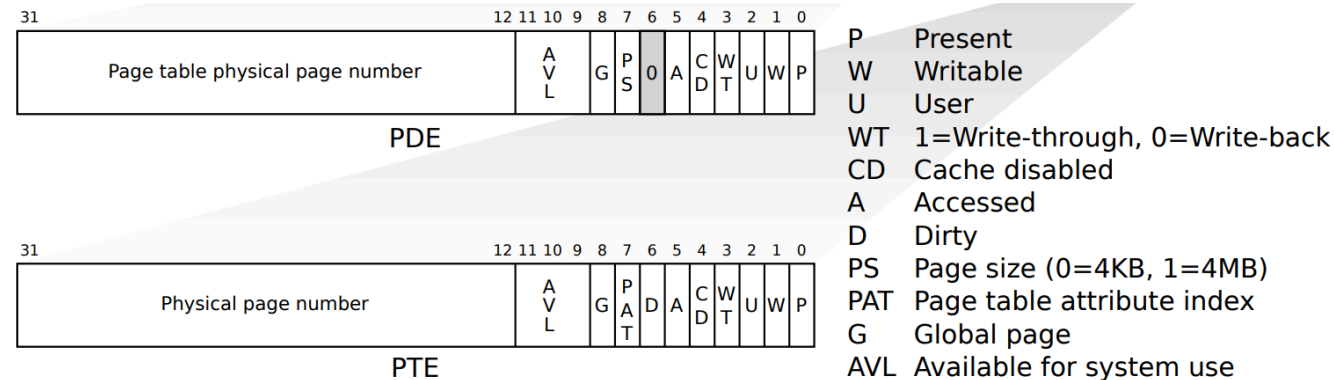
- Top 20 bits: physical page number
 - Physical page number of a page table (PDE)
 - Physical page number of the requested virtual address (PTE)
- Lower 12 bits: some flags
 - Permission
 - Etc.



Permission Flags

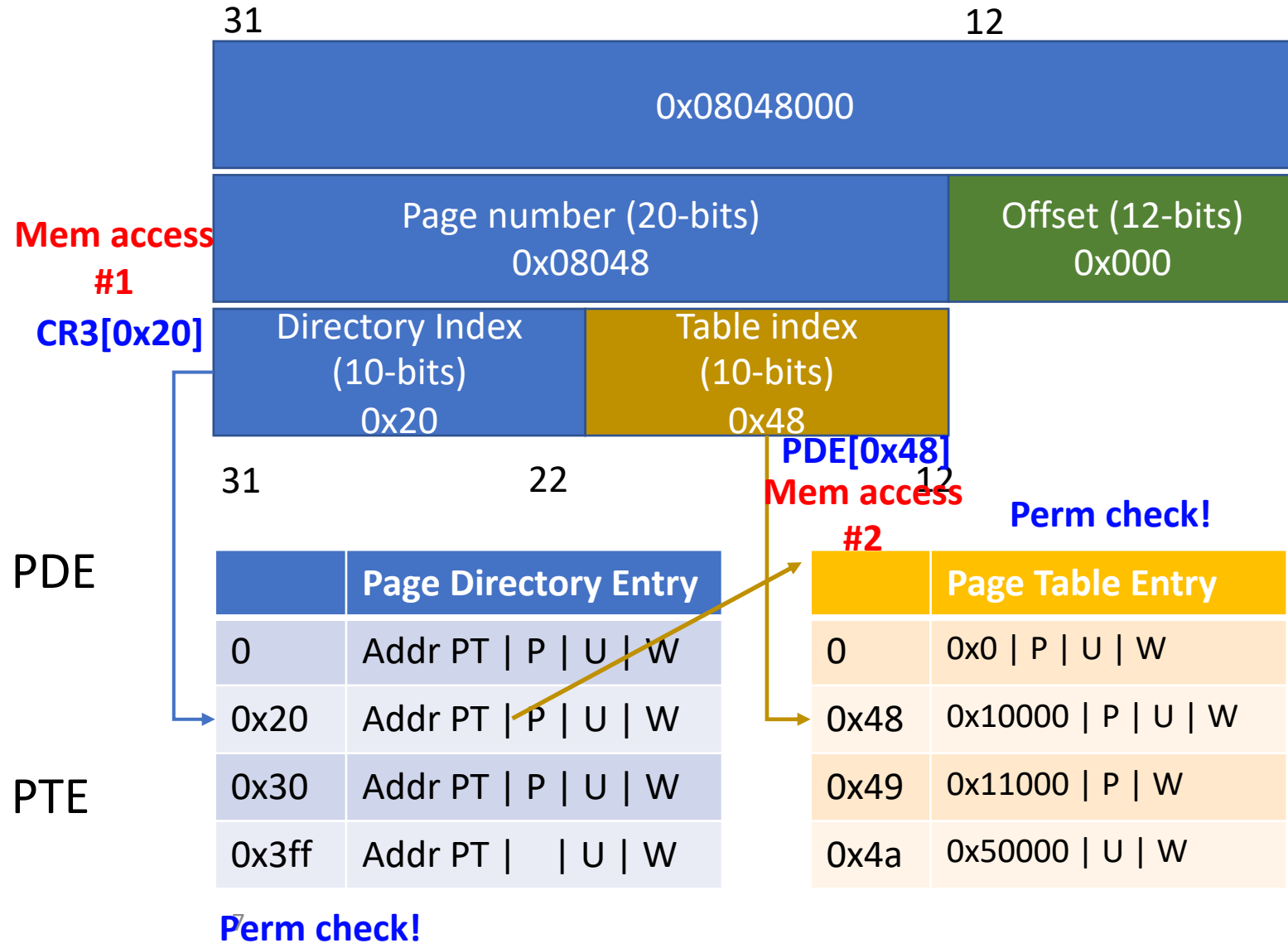
- PTE_P (PRESENT)
 - 0: invalid entry
 - 1: valid entry
- PTE_W (WRITABLE)
 - 0: read only
 - 1: writable
- PTE_U (USER)
 - 0: kernel (only ring 0 can access)
 - 1: user (accessible by ring 3)

	Page Table Entry		
0	Addr PT		
0x48	0x10000 << 12 PTE_U PTE_W		Invalid
0x49	0x11000 << 12 PTE_P PTE_W		Kernel, writable
0x4a	0x50000 << 12 PTE_P PTE_U		User, read-only



When CPU Checks Permission Bits?

- In address translation
- 1. Virtual address
- 2. PDE = CR3[PDX]
 - Checks permission bits in PDE
- 3. PTE = PDE[PTX]
 - Checks permission bits in PTE



CPU checks PDE permission first and then

PTE permission next...

- A virtual memory address is inaccessible if PDE disallows the access
- A virtual memory address is inaccessible if PTE disallows the access
- Both PDE and PTE should allow the access...

PDE/PTE Permission Examples 0

- Virtual address 0x01020304
 - PDE: PTE_P | PTE_W | PTE_U
 - PTE: PTE_P | PTE_W | PTE_U
 - Valid, accessible by ring 3, and writable
- PTE_P (PRESENT)
 - 0: invalid entry
 - 1: valid entry
 - PTE_W (WRITABLE)
 - 0: read only
 - 1: writable
 - PTE_U (USER)
 - 0: kernel (only ring 0 can access)
 - 1: user (accessible by ring 3)

PDE/PTE Permission Examples 1

- Virtual address 0x01020304
 - PDE: PTE_P | PTE_W | PTE_U
 - PTE: PTE_P | PTE_U
 - Valid, accessible by ring 3, but not writable
- PTE_P (PRESENT)
 - 0: invalid entry
 - 1: valid entry
 - PTE_W (WRITABLE)
 - 0: read only
 - 1: writable
 - PTE_U (USER)
 - 0: kernel (only ring 0 can access)
 - 1: user (accessible by ring 3)

PDE/PTE Permission Examples 2

- Virtual address 0x01020304
 - PDE: PTE_P | PTE_U
 - PTE: PTE_P | PTE_W | PTE_U
 - Valid, accessible by ring 3, but not writable
- PTE_P (PRESENT)
 - 0: invalid entry
 - 1: valid entry
 - PTE_W (WRITABLE)
 - 0: read only
 - 1: writable
 - PTE_U (USER)
 - 0: kernel (only ring 0 can access)
 - 1: user (accessible by ring 3)

PDE/PTE Permission Examples 3

- Virtual address 0x01020304
 - PDE: PTE_P | PTE_W | PTE_U
 - PTE: PTE_P
 - valid, inaccessible by ring3, not writable
- PTE_P (PRESENT)
 - 0: invalid entry
 - 1: valid entry
 - PTE_W (WRITABLE)
 - 0: read only
 - 1: writable
 - PTE_U (USER)
 - 0: kernel (only ring 0 can access)
 - 1: user (accessible by ring 3)

PDE/PTE Permission Examples 4

- Virtual address 0x01020304
 - PDE: PTE_P | PTE_W
 - PTE: PTE_P | PTE_U
 - valid, inaccessible by ring3, not writable
- PTE_P (PRESENT)
 - 0: invalid entry
 - 1: valid entry
 - PTE_W (WRITABLE)
 - 0: read only
 - 1: writable
 - PTE_U (USER)
 - 0: kernel (only ring 0 can access)
 - 1: user (accessible by ring 3)

PDE/PTE Permission Examples 5

- Virtual address 0x01020304
 - PDE: PTE_P | PTE_U
 - PTE: PTE_U
 - invalid
- PTE_P (PRESENT)
 - 0: invalid entry
 - 1: valid entry
 - PTE_W (WRITABLE)
 - 0: read only
 - 1: writable
 - PTE_U (USER)
 - 0: kernel (only ring 0 can access)
 - 1: user (accessible by ring 3)

PDE/PTE Permission Examples 6

- Virtual address 0x01020304
 - PDE: PTE_U
 - PTE: PTE_P | PTE_U
 - invalid
- PTE_P (PRESENT)
 - 0: invalid entry
 - 1: valid entry
 - PTE_W (WRITABLE)
 - 0: read only
 - 1: writable
 - PTE_U (USER)
 - 0: kernel (only ring 0 can access)
 - 1: user (accessible by ring 3)

Can you setup a page permission as...

- Kernel: R, User: --
 - PTE_P
- Kernel: R, User: R
 - PTE_P | PTE_U
- Kernel: RW, User: RW
 - PTE_P | PTE_U | PTE_W

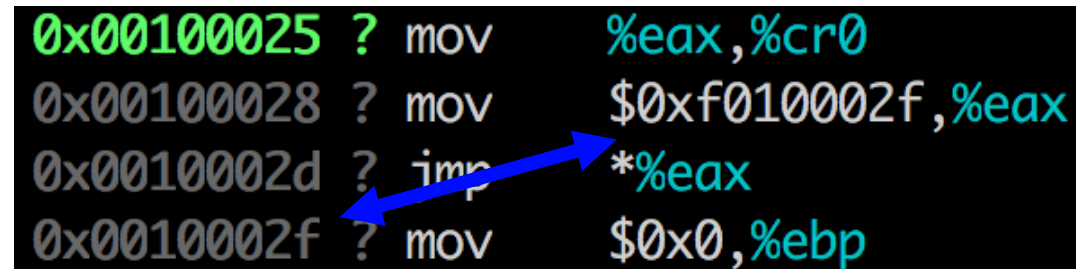
You can't setup a page permission as...

- Kernel: RW, User: R
 - PTE_P | PTE_W | PTE_U -> User RW...
 - PTE_P | PTE_W -> User --
- Kernel: R, User: RW
 - PTE_P | PTE_U | PTE_W -> Kernel RW...
 - PTE_P | PTE_U -> User R...
- Kernel: --, User: RW
 - PTE_P | PTE_U | PTE_W -> Kernel RW...

You can enable such a conflicting permission setup by having N-to-1 mapping

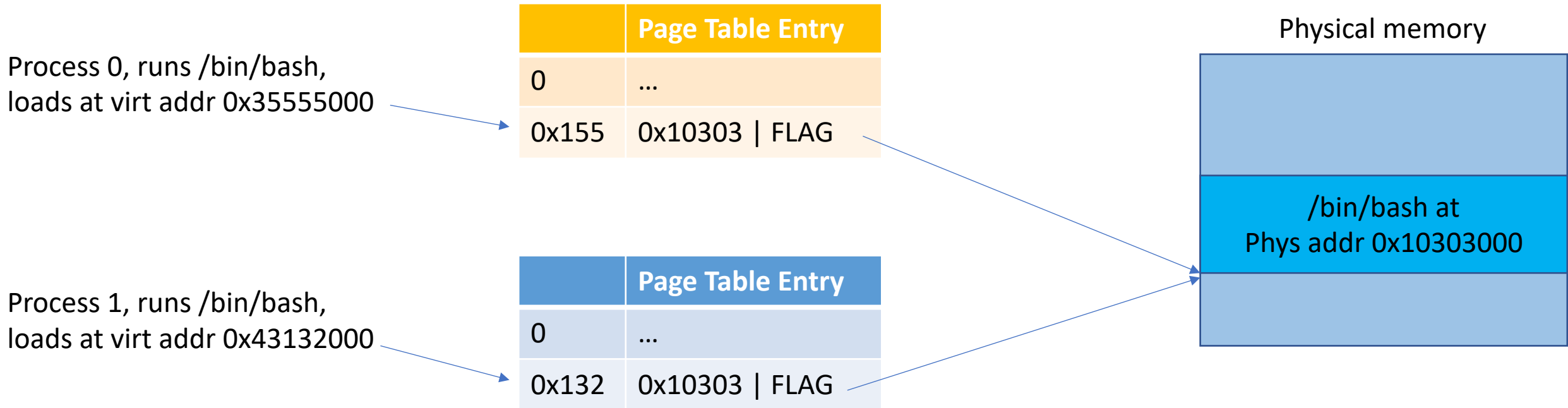
- Virtual to physical address mapping is in N-to-1 relation
 - N number of virtual addresses could be mapped to 1 physical address
- E.g., for a physical address 0x100000
 - JOS maps VA 0x100000 to PA 0x100000
 - JOS maps VA 0xf0100000 to PA 0x100000
- Why?
 - EIP before enabling paging: 0x100025
 - EIP after enabling paging: 0x100028

```
0x00100025 ? mov %eax,%cr0
0x00100028 ? mov $0xf010002f,%eax
0x0010002d ? imp *%eax
0x0010002f ? mov $0x0,%ebp
```

A screenshot of assembly code with a blue arrow pointing from the instruction at 0x0010002d to the instruction at 0x00100028. The code is as follows:
0x00100025 ? mov %eax,%cr0
0x00100028 ? mov \$0xf010002f,%eax
0x0010002d ? imp *%eax
0x0010002f ? mov \$0x0,%ebp

Sharing a Physical Page

- Example: Loading of the same program



2 or more mappings to 0x10303000 is possible!

You can't setup a page permission as...

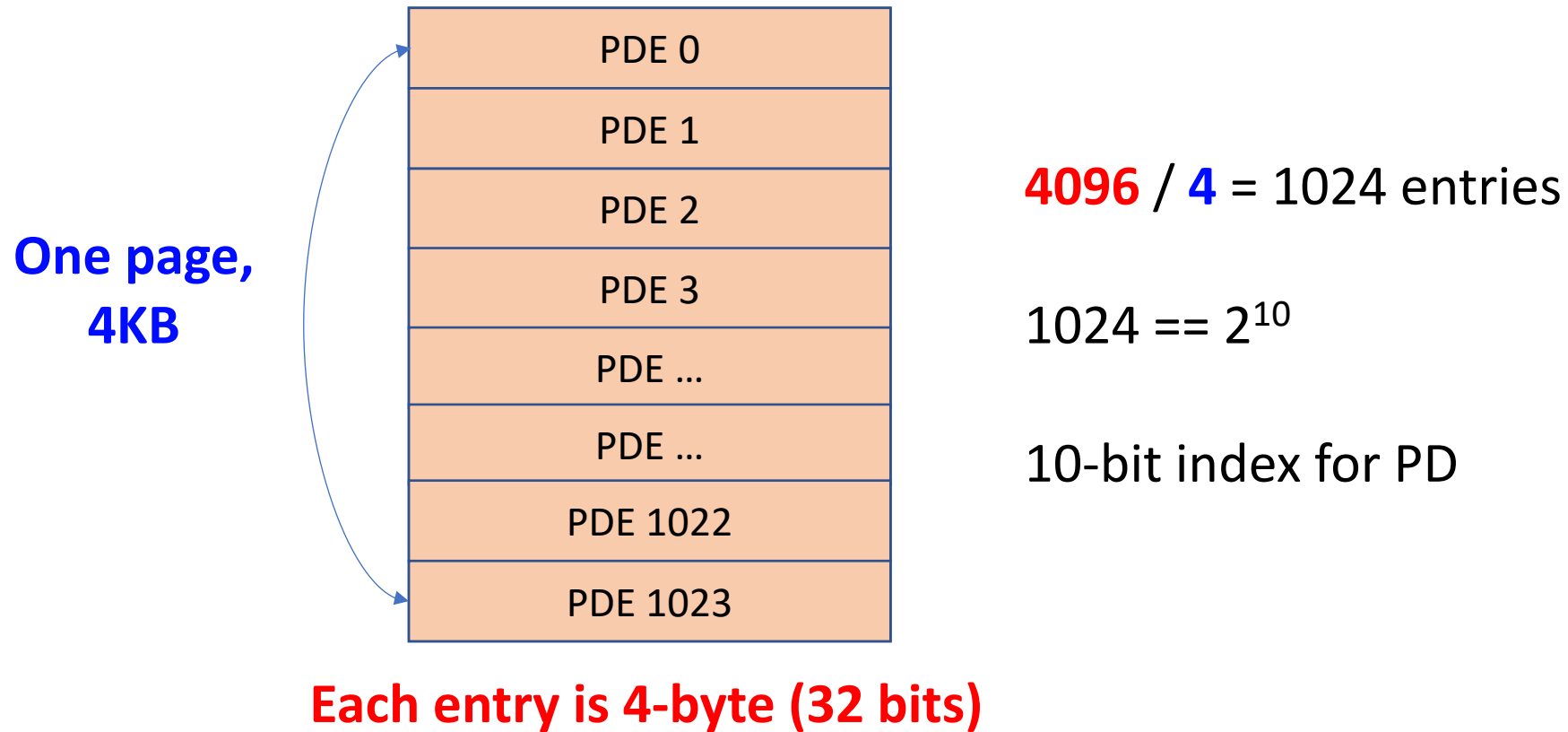
- Kernel: RW, User: R
 - VA 0x00001000 -> PA 0x50000, PTE_P | PTE_U (User R)
 - VA 0xf0050000 -> PA 0x50000, PTE_P | PTE_W (Kernel RW)
- Kernel: R, User: RW
 - VA 0x00002000 -> PA 0x60000, PTE_P | PTE_U | PTE_W (User RW)
 - VA 0xf0060000 -> PA 0x60000, PTE_P (Kernel R)
- Kernel: --, User: RW
 - VA 0x00003000 -> PA 0x70000, PTE_P | PTE_U | PTE_W
 - VA 0xf0070000 -> PA 0x70000, 0 for flag...

PDE/PTE Permissions CAVEAT

- A virtual address access is allowed if both **PDE** and **PTE** entries allows the access...
- General practice: put **a more permissive** permission bits in **PDE**, and **be strict** on setting permission bits in **PTE**
- For a conflicting permission setup for Kernel/User, add **an additional virtual address mapping** can enable such a setup

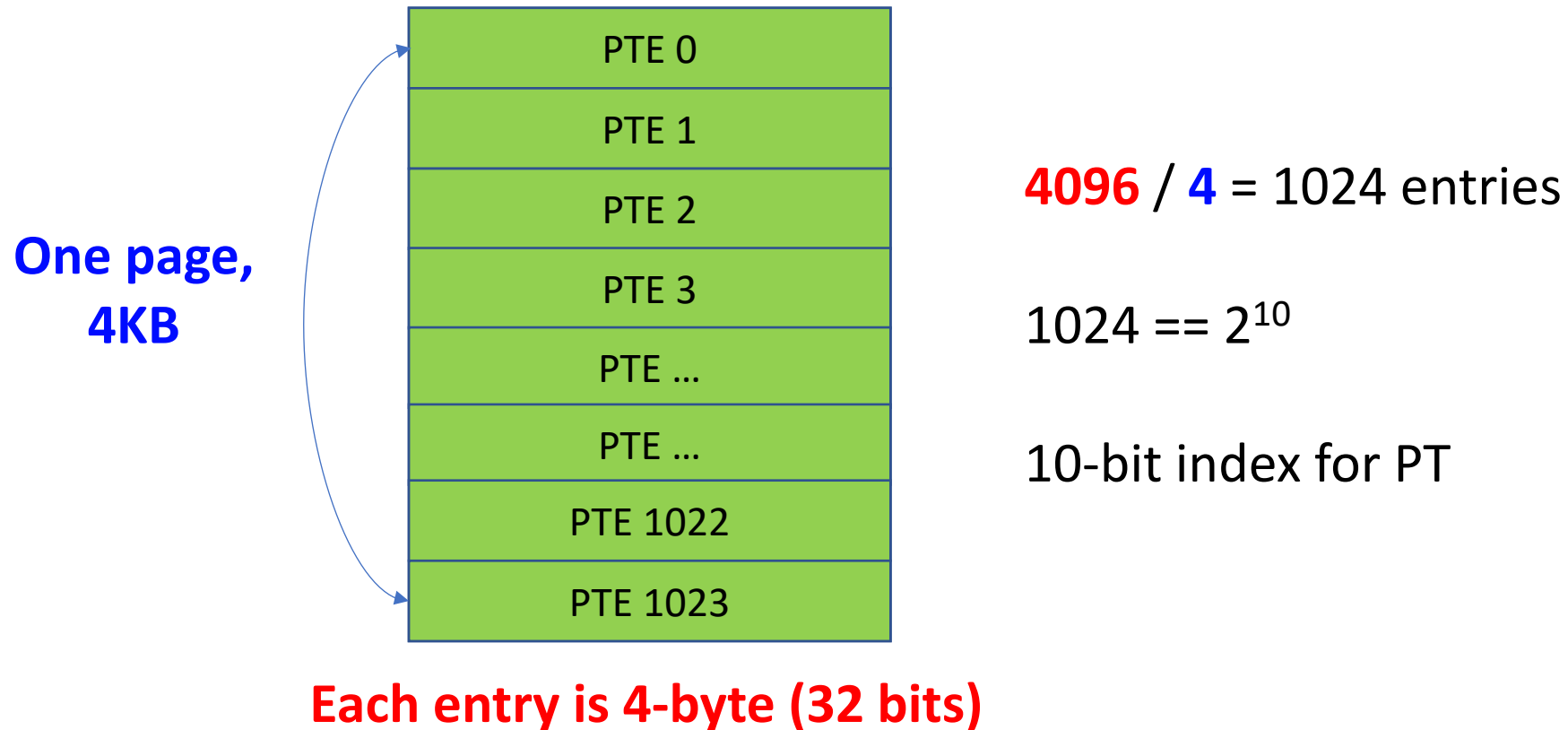
How To Create a Page Directory?

- For a 32-bit Intel processor, we use only 1 page for a Page Directory



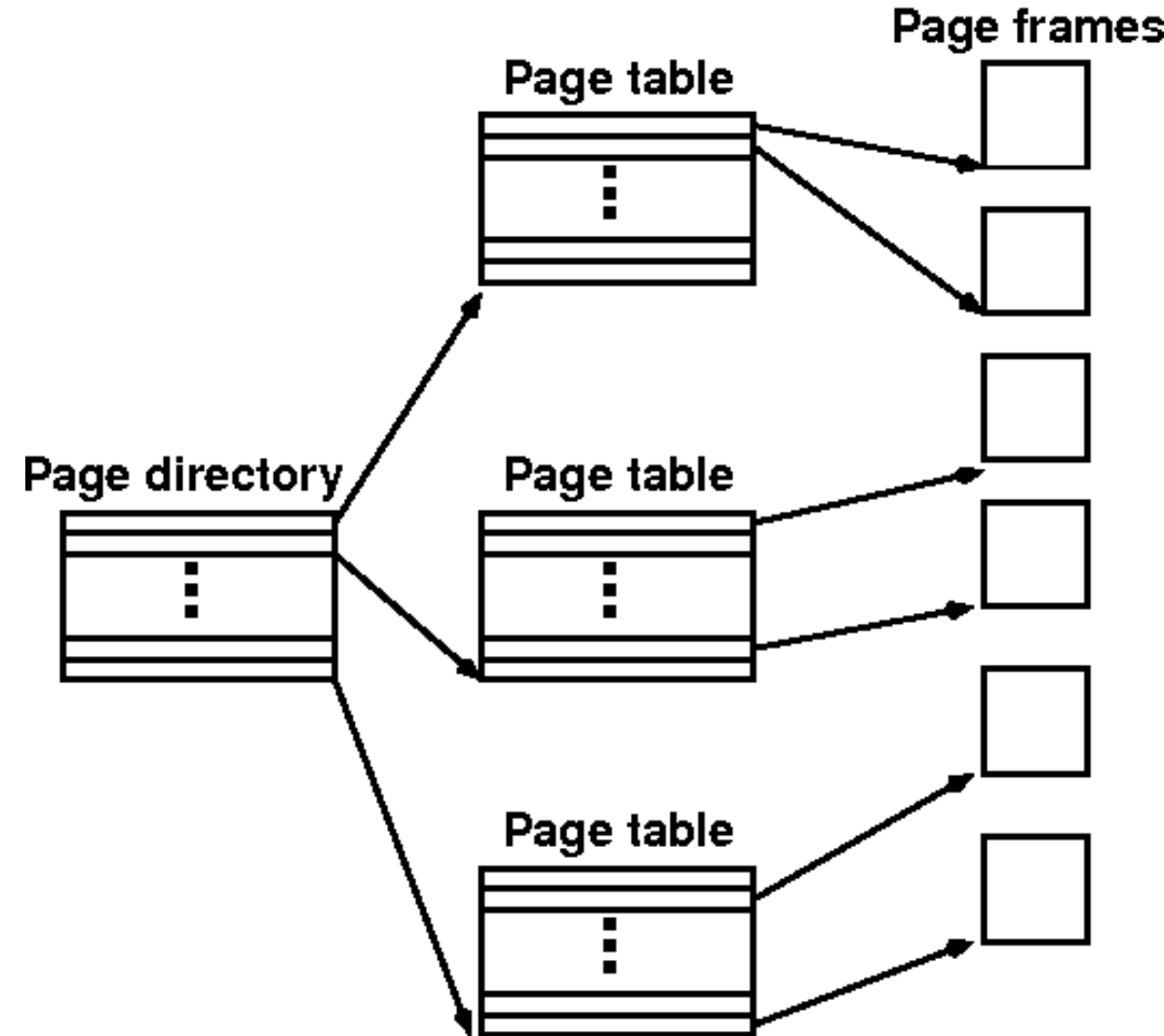
How To Create a Page Table?

- For a 32-bit Intel processor, we use only 1 page for a Page Table

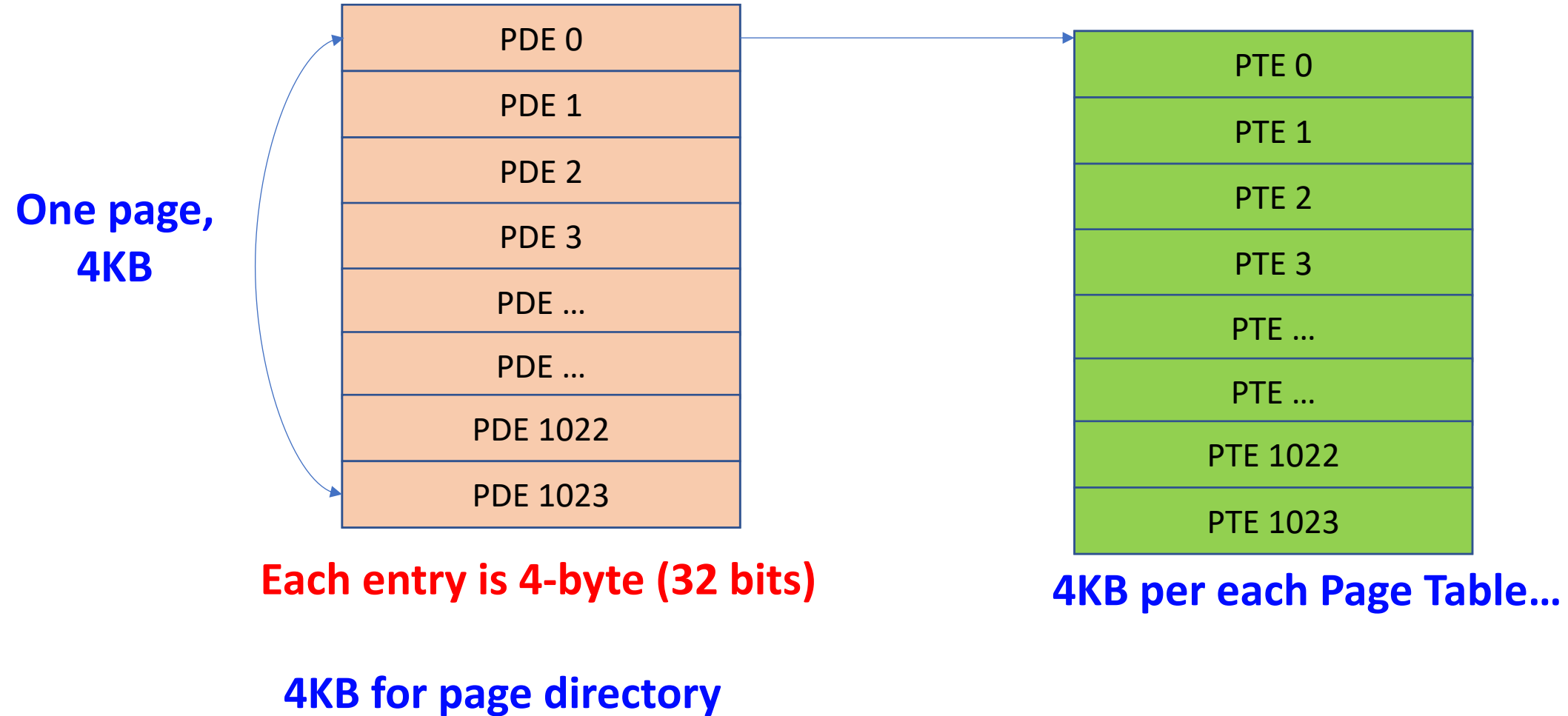


Intel 32-bit Processor uses a 2-level page table

- Virtual address
- Page directory (level 1)
- Page table (level 2)
- Physical page!



What will be the size of full PD/PT?

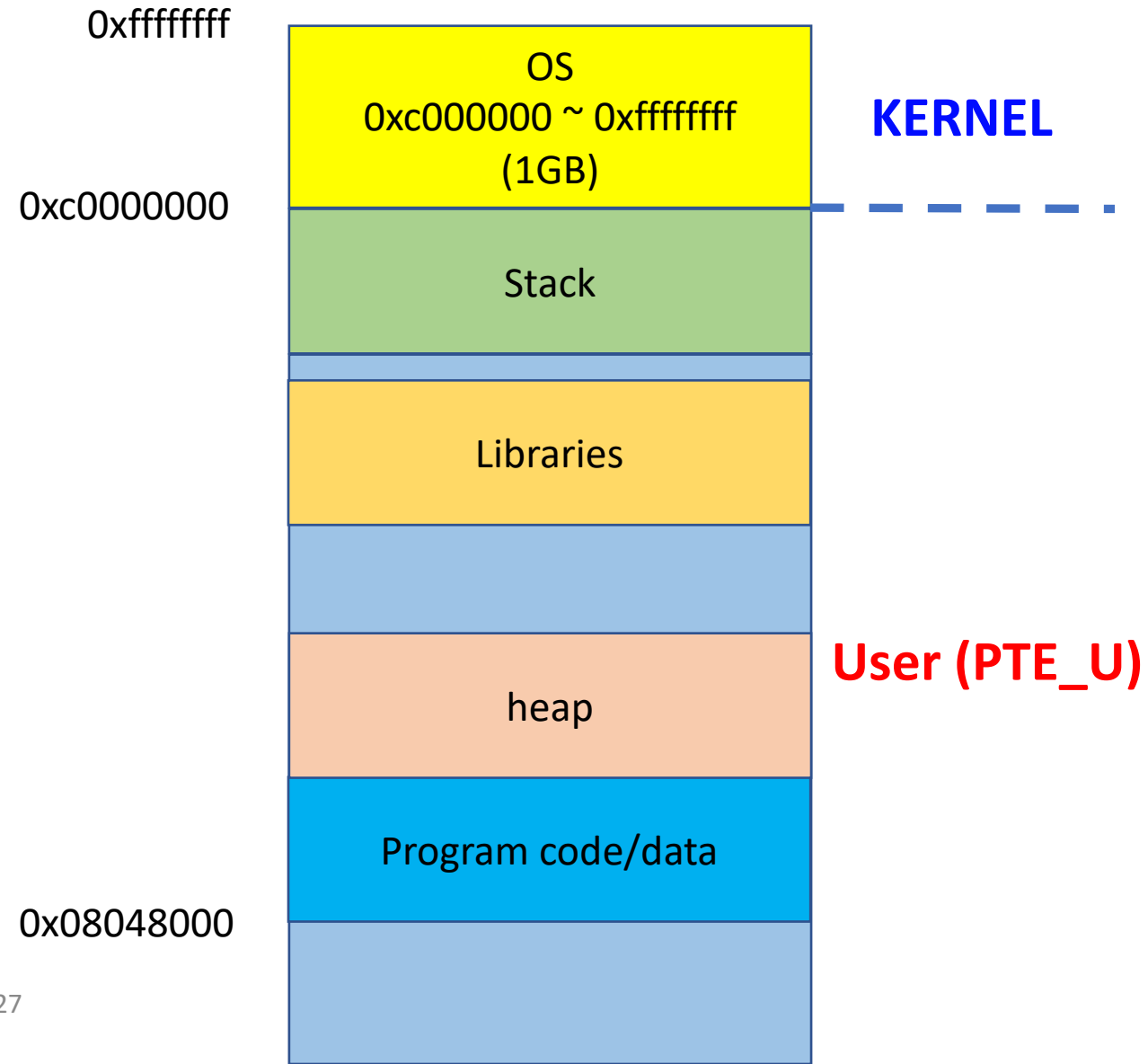


Size of Page Table

- 4 KB for a Page Directory (only one per each process)
- 1024 Page Tables available
 - 4 KB for a Page Table
- 4 KB (PD) + 1024 * 4 KB (PT)
 - 4 KB + 4MB
 - ~ 4MB, 4,198,400 bytes...

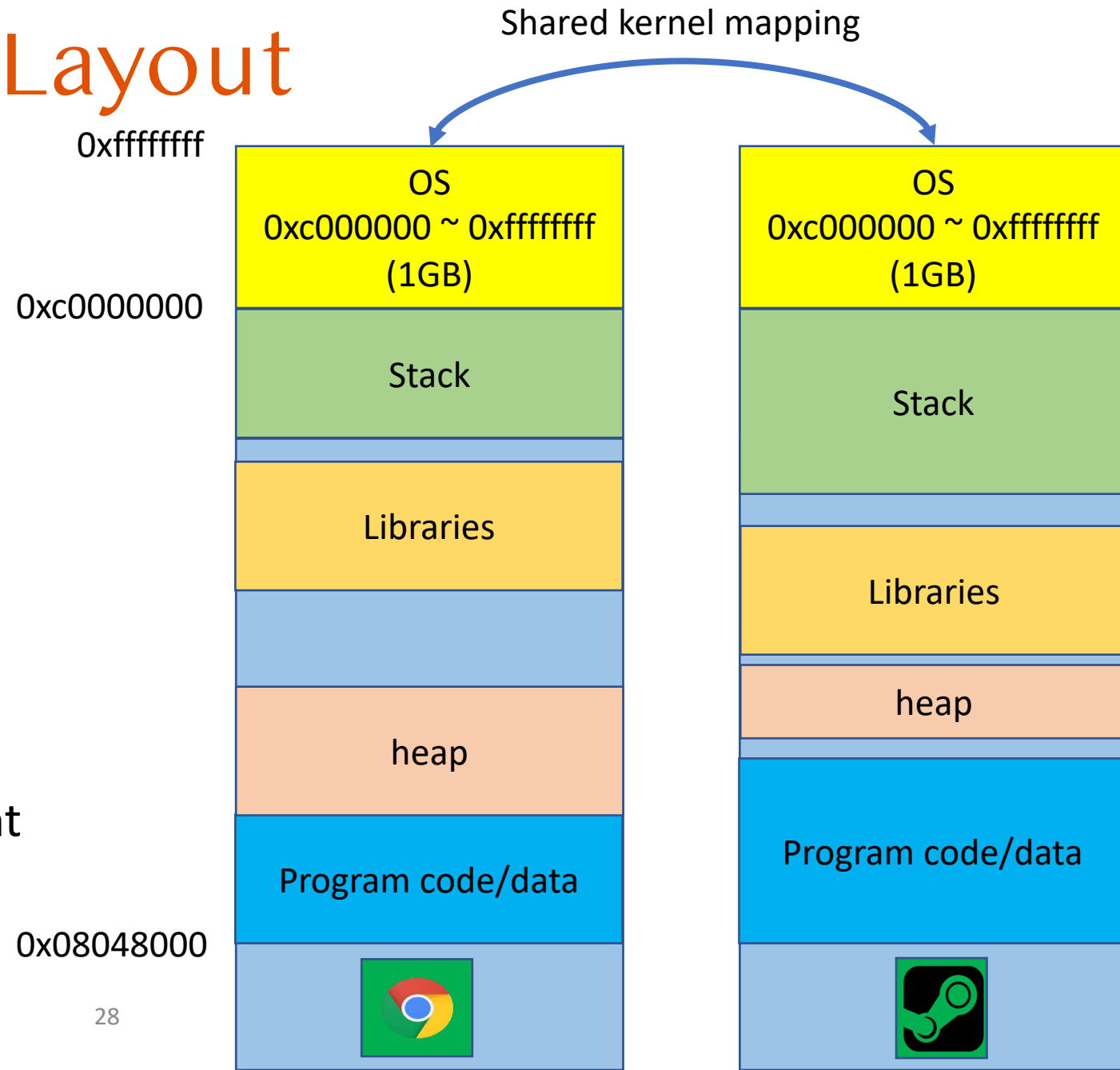
Virtual Memory Layout

- OS allocates a separate virtual memory space to each process
- Transparency
 - Do not have to worry about a system's memory usage status
- Isolation
 - Others can't access my virtual memory space



Virtual Memory Layout

- Each process will have almost the same mapping for the kernel but having a different mapping for userspace
- Why?
 - Kernel is shared among processes
 - Each process could run different apps



Example: Linux on 32-bit Intel

```
blue9057@blue9057-vm-ctf3 ~ $ cat /proc/self/maps
08048000-08054000 r-xp 00000000 08:01 1835032 /bin/cat
08054000-08055000 r--p 0000b000 08:01 1835032 /bin/cat
08055000-08056000 rw-p 0000c000 08:01 1835032 /bin/cat
08910000-08931000 rw-p 00000000 00:00 0 [heap]
b7b55000-b7b77000 rw-p 00000000 00:00 0
b7b77000-b7d77000 r--p 00000000 08:01 3278358 /usr/lib/locale/locale-archive
b7d77000-b7d78000 rw-p 00000000 00:00 0
b7d78000-b7f28000 r-xp 00000000 08:01 2114585 /lib/i386-linux-gnu/libc-2.23.so
b7f28000-b7f2a000 r--p 001af000 08:01 2114585 /lib/i386-linux-gnu/libc-2.23.so
b7f2a000-b7f2b000 rw-p 001b1000 08:01 2114585 /lib/i386-linux-gnu/libc-2.23.so
b7f2b000-b7f2e000 rw-p 00000000 00:00 0
b7f44000-b7f45000 r--p 002d4000 08:01 3278358 /usr/lib/locale/locale-archive
b7f45000-b7f46000 rw-p 00000000 00:00 0
b7f46000-b7f49000 r--p 00000000 00:00 0 [vvar]
b7f49000-b7f4b000 r-xp 00000000 00:00 0 [vdso]
b7f4b000-b7f6e000 r-xp 00000000 08:01 2114571 /lib/i386-linux-gnu/ld-2.23.so
b7f6e000-b7f6f000 r--p 00022000 08:01 2114571 /lib/i386-linux-gnu/ld-2.23.so
b7f6f000-b7f70000 rw-p 00023000 08:01 2114571 /lib/i386-linux-gnu/ld-2.23.so
bf9a9000-bf9ca000 rw-p 00000000 00:00 0 [stack]
```

Example: Linux on 32-bit Intel

```
blue9057@blue9057-vm-ctf3 ~ $ cat /proc/self/maps
08048000-08054000 r-xp 00000000 08:01 1835032 /bin/cat
08054000-08055000 08048000-08051000 r-xp 00000000 08:01 1835015 /bin/more
08055000-08056000 08051000-08052000 r--p 00008000 08:01 1835015 /bin/more
08910000-08931000 08052000-08053000 rw-p 00009000 08:01 1835015 /bin/more
b7b55000-b7b77000 08bb6000-08bd7000 rw-p 00000000 00:00 0 [heap]
b7b77000-b7d77000 b7b85000-b7d85000 r--p 00000000 08:01 3278358 /usr/lib/locale/locale-archive
b7d77000-b7d78000 b7d85000-b7d86000 rw-p 00000000 00:00 0
b7d78000-b7f28000 b7d86000-b7f36000 r-xp 00000000 08:01 2114585 /lib/i386-linux-gnu/libc-2.23.so
b7f28000-b7f2a000 b7f36000-b7f38000 r--p 001af000 08:01 2114585 /lib/i386-linux-gnu/libc-2.23.so
b7f2a000-b7f2b000 b7f38000-b7f39000 rw-p 001b1000 08:01 2114585 /lib/i386-linux-gnu/libc-2.23.so
b7f2b000-b7f2e000 b7f39000-b7f3c000 rw-p 00000000 00:00 0
b7f2e000-b7f44000 b7f3c000-b7f5c000 r-xp 00000000 08:01 2098856 /lib/i386-linux-gnu/libtinfo.so
b7f44000-b7f45000 b7f5c000-b7f5e000 r--p 0001f000 08:01 2098856 /lib/i386-linux-gnu/libtinfo.so
b7f45000-b7f46000 b7f5e000-b7f5f000 rw-p 00021000 08:01 2098856 /lib/i386-linux-gnu/libtinfo.so
b7f46000-b7f49000 b7f75000-b7f76000 r--p 002d4000 08:01 3278358 /usr/lib/locale/locale-archive
b7f49000-b7f4b000 b7f76000-b7f77000 rw-p 00000000 00:00 0
b7f4b000-b7f6e000 b7f77000-b7f7a000 r--p 00000000 00:00 0 [vvar]
b7f6e000-b7f6f000 b7f7a000-b7f7c000 r-xp 00000000 00:00 0 [vdso]
b7f6f000-b7f70000 b7f7c000-b7f9f000 r-xp 00000000 08:01 2114571 /lib/i386-linux-gnu/ld-2.23.so
bf9a9000-bf9ca000 b7f9f000-b7fa0000 r--p 00022000 08:01 2114571 /lib/i386-linux-gnu/ld-2.23.so
b7fa0000-b7fa1000 rw-p 00023000 08:01 2114571 /lib/i386-linux-gnu/ld-2.23.so
bfa73000-bfa94000 rw-p 00000000 00:00 0 [stack]
```

Summary

- Page Directory Entry / Page Table Entry
 - Permission bits (P, W, U)
 - Permission: {bits in PDE} \cap {bits in PTE}
- Virtual memory is N-to-1 mapping
 - Sharing physical page
 - Allowing conflicting permission assignment
 - Kernel RW and User R
- Virtual Memory Layout
 - Shares kernel space (typically at the top of virtual memory space)
 - Can use user space arbitrarily (full transparency and isolation)