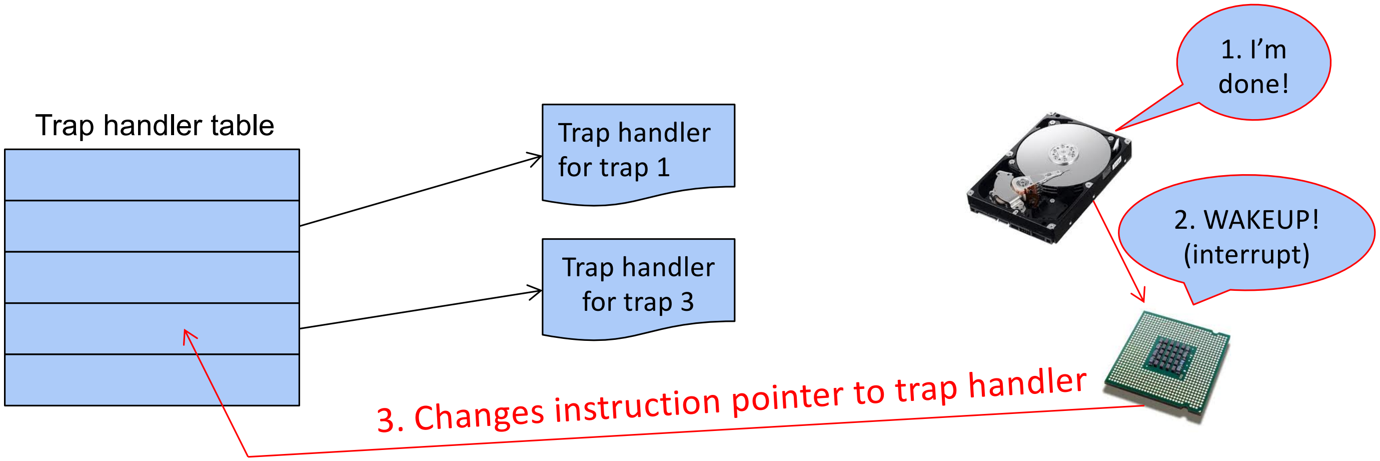
Today

- Roadmap:
  - Recall that when we talked about protected control transfer, I promised we'd come back to it in the context of virtual memory – that's today!
- Learning Outcomes
  - Trace execution through handling of a TLB fault.
  - Identify steps that the operating system must take in the presence of a fault.
  - Differentiate behavior in the presence/absence of page tables in hardware.
- Reading
  - 9.6

Recall: Trap Handling --interrupt



Trap handler table

Trap handler for trap 1

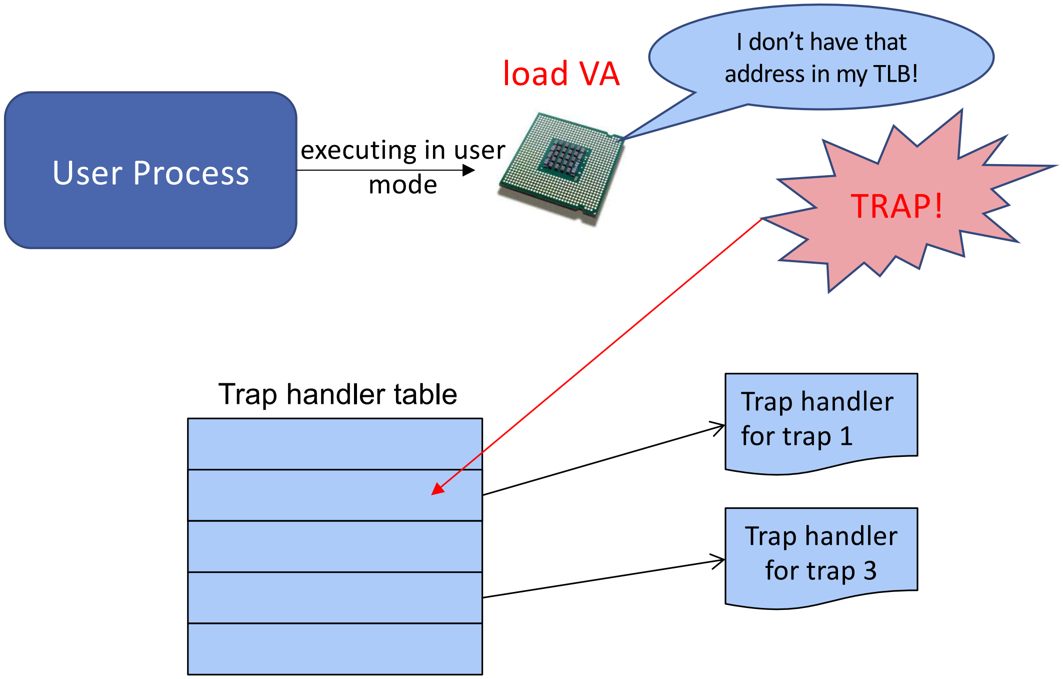
Trap handler for trap 3

1. I'm done!

2. WAKEUP! (interrupt)

3. Changes instruction pointer to trap handler

Recall: Trap Handling -- TLB fault (exception)



Note:

- Some processors handle TLB faults in HW.
- For now, we will assume a processor that **does not** handle TLB faults in HW
- We'll come back to processors that do (i.e., the x86) in class.

Let’s apply this to TLB Faults

- TLB does not have an entry for the VPN.
- Traps into the OS
  - OS looks up entry in page table
    1. Entry is invalid: kill process (report segfault)
    2. Entry is valid and in-memory (present): enter PTE information into the TLB
    3. Entry is valid but not present: read page in from disk and reflect physical page number in the PTE. Then do one of:
       - Restart the instruction (will generate a TLB fault again, but will fall into case 2)
       - Load PTE into the TLB; restart instruction

SW Fault Handling: (1)

1. Interpret the PTE to determine if virtual address is valid.

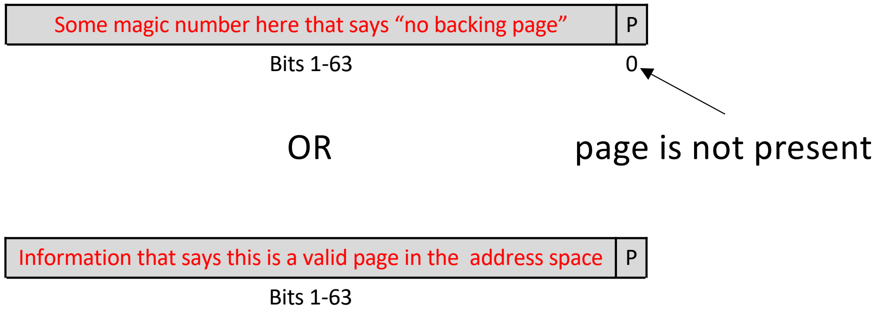
▨ Pages owned by the OS

🟦 Pages owned by process A

🟩 Pages owned by process B

🟪 Pages owned by process C

□ Free Pages



Some magic number here that says "no backing page" | P

Bits 1-63 | 0

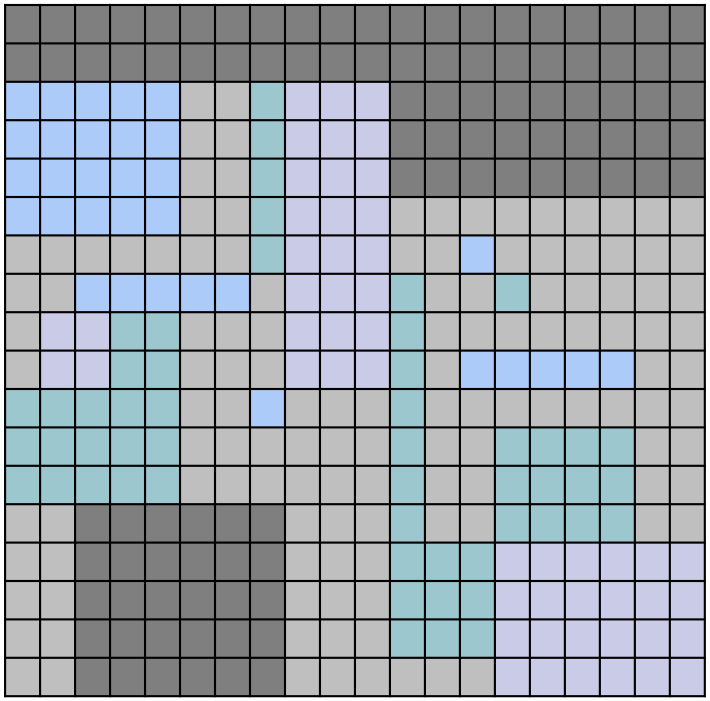
page is not present

OR

Information that says this is a valid page in the address space | P

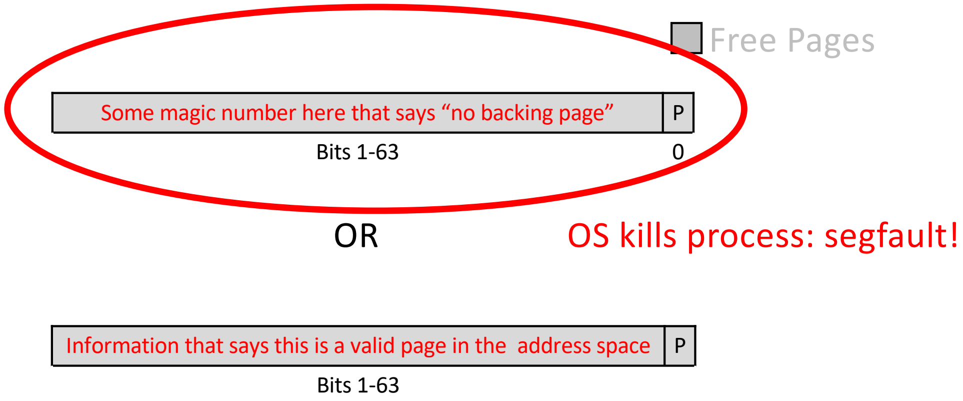
Bits 1-63

DRAM: Each block is a physical page



SW Page Fault Handling: (2)

1. Interpret the PTE to determine if the virtual address is valid.



Some magic number here that says “no backing page” | P

Bits 1-63 | 0

OR

OS kills process: segfault!

Information that says this is a valid page in the address space | P

Bits 1-63

DRAM: Each block is a physical page

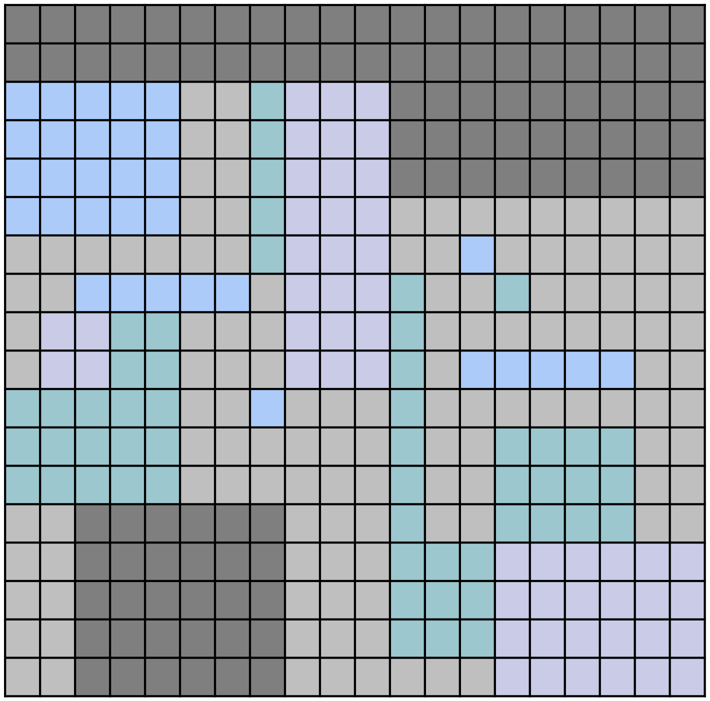
■ Pages owned by the OS

□ Pages owned by process A

■ Pages owned by process B

□ Pages owned by process C

□ Free Pages



SW Page Fault Handling: (3)

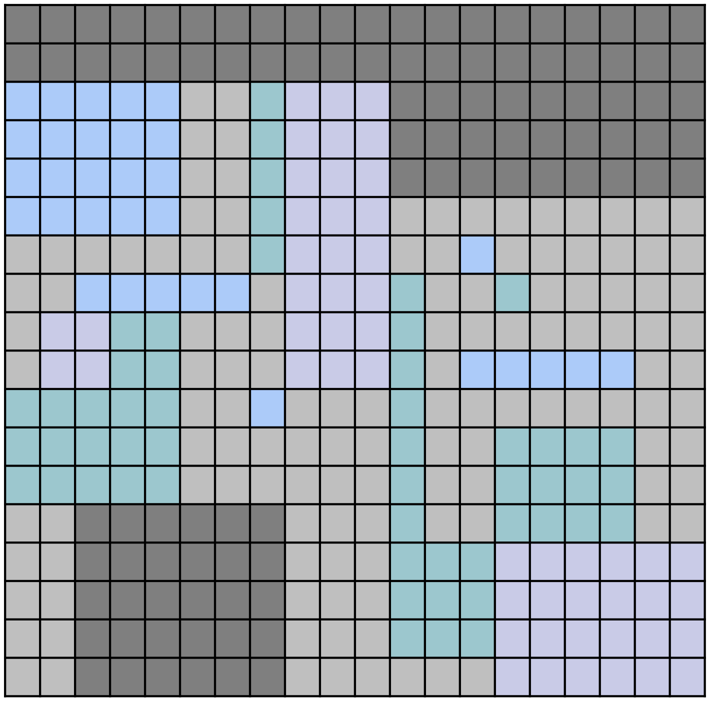
1. Interpret the PTE to determine if the virtual address is valid.
2. Is the page present?

Information here is valid; load into TLB	P
Bits 1-63	1

Information here that says: device/block OR 0-fill	P
Bits 1-63	0

- ■ Pages owned by the OS
- ■ Pages owned by process A
- ■ Pages owned by process B
- ■ Pages owned by process C
- ■ Free Pages

DRAM: Each block is a physical page



SW Page Fault Handling: (4)

1. Interpret the PTE to determine if the virtual address is valid.
2. Is the page present?
3. Find a free physical page.

Information here that says: device/block OR 0-fill	P
Bits 1-63	0

■ Pages owned by the OS

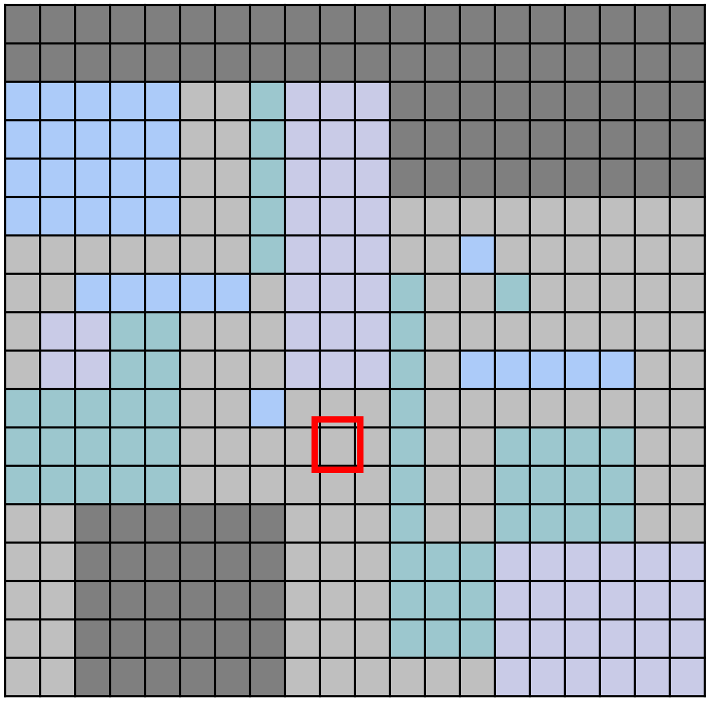
🟦 Pages owned by process A

🟩 Pages owned by process B

🟪 Pages owned by process C

▢ Free Pages

DRAM: Each block is a physical page



SW Page Fault Handling: (5)

1. Interpret the PTE to determine if the virtual address has a backing page.
2. Is the page present?
3. Find a free physical page.
4. Place page contents into page.

Information here that says: device/block OR 0-fill	P
Bits 1-63	0

■ Pages owned by the OS

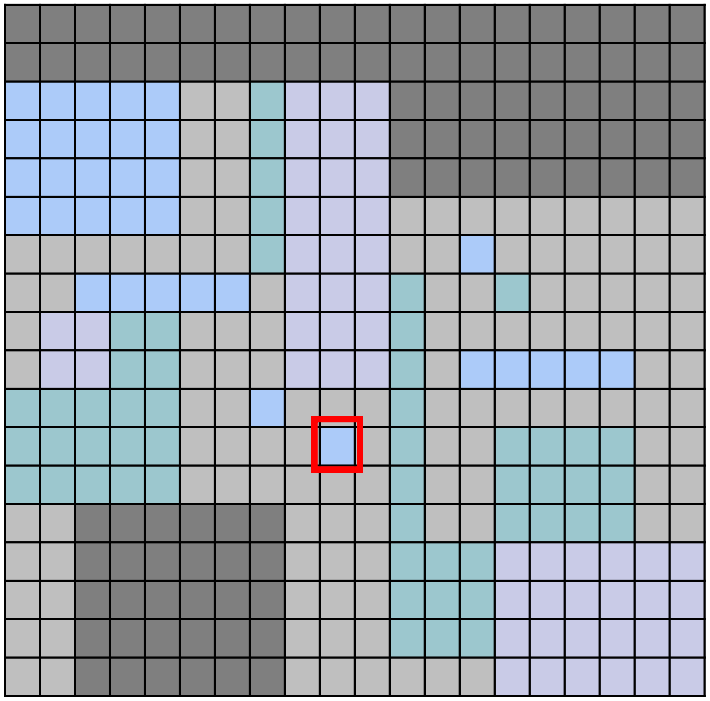
■ Pages owned by process A

■ Pages owned by process B

■ Pages owned by process C

■ Free Pages

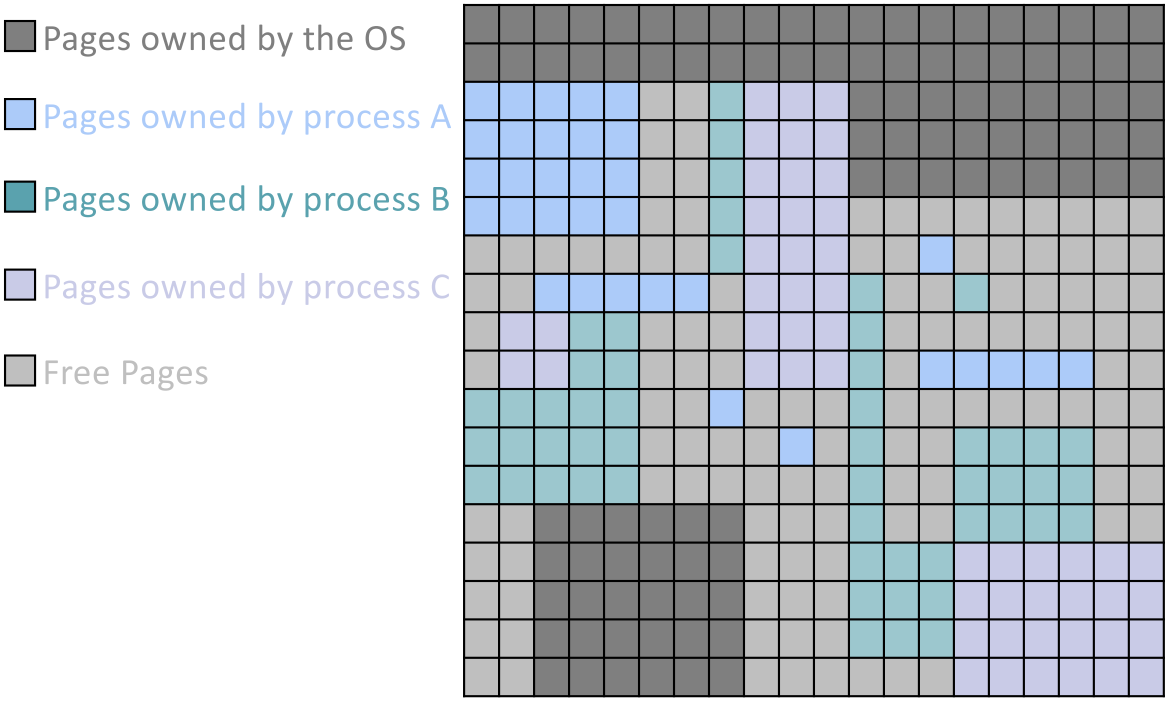
DRAM: Each block is a physical page



SW Page Fault Handling: (6)

1. Interpret the PTE to determine if the virtual address has a backing page.
2. Is the page present?
3. Find a free physical page.
4. Place page contents into page.
5. Fill in PTE
6. Restart instruction

DRAM: Each block is a physical page


■ Pages owned by the OS
■ Pages owned by process A
■ Pages owned by process B
■ Pages owned by process C
■ Free Pages

Literally start execution of the instruction all over!
Return to user mode via some kind of “return from trap instruction”

Fault handling summary

- HW detects that there is no mapping
- Software does some combination of:
  - Determining that there is no way to continue (kill the process)
  - Load the TLB
  - Allocates and initializes a page (either from disk or with 0’s)
  - Updates the page table entry