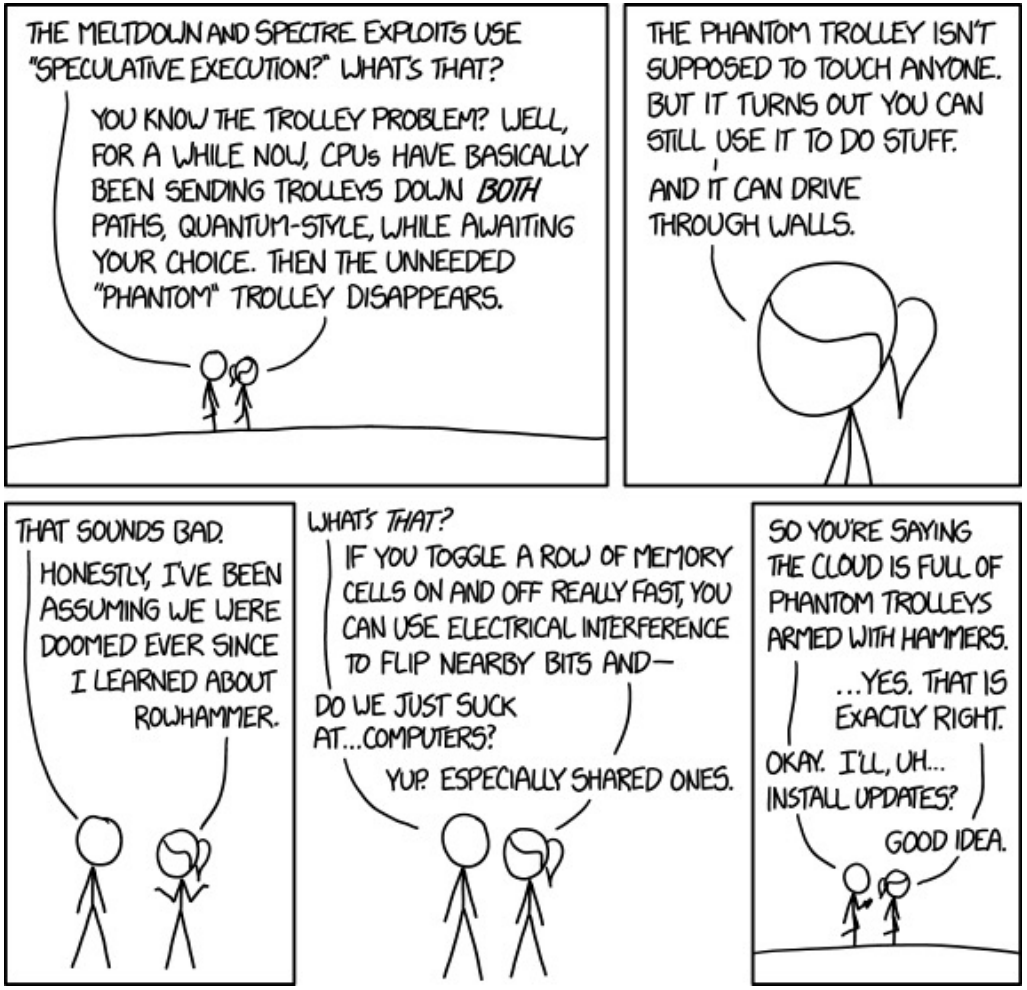
Fantastic Bugs and How To Find Them

CSE 351 Winter 2022

Instructor:
Sam Wolfson

Teaching Assistants:
Angela Xu
Anirudh Kumar
Catherine Guevara
Dara Stotland
Harrison Bay
Ian Hsiao
Kevin Wang
Mara Kirdani-Ryan
Nick Durand
Sanjana Sridhar



https://xkcd.com/1938/

Administrivia

- hw25 due Friday (3/11)
- Lab 5 due Friday (3/11)
  - You can use late days as usual; latest turn in Monday night

- **Final Exam:** 3/15—3/17
  - Similar to midterm: Gilligan's Island Rule
  - Final review section this week (+ VM)
  - Review Session next Monday, location TBD

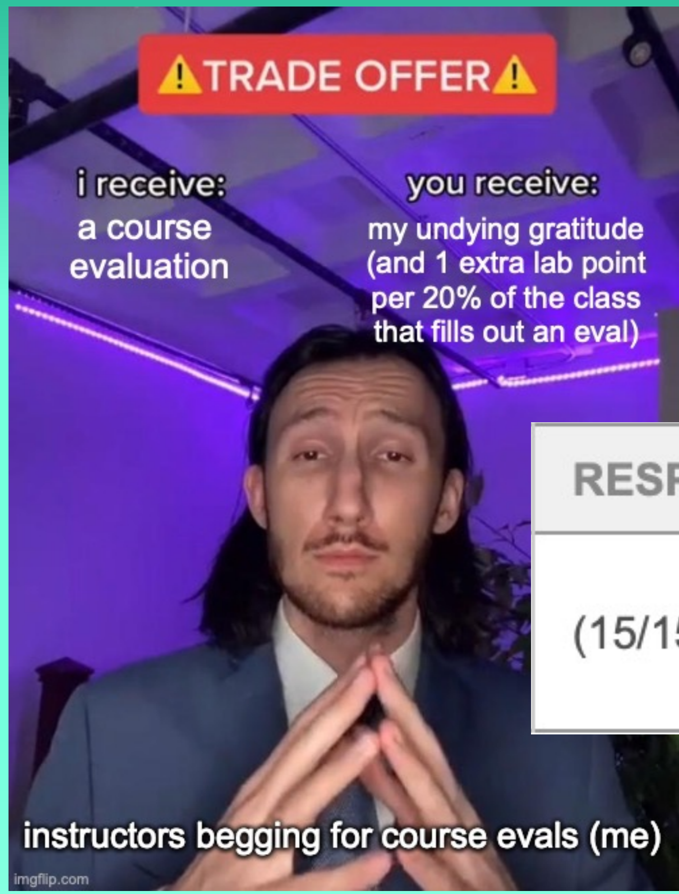
Course Evaluations!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Please fill these out!!!

😂 😂 😂 😂 😂 😂 😂 😂 😂 😂 😂

https://uw.iasystem.org/survey/253745

Course Evaluations!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!



RESPONSES
(15/159) 9% low

https://uw.iasystem.org/survey/253745

Course Evaluations!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

- Please fill these out for your sections as well!
  - Your TAs and I both highly value your feedback.

- AA/BA: Dara & Sanjana
  - https://uw.iasystem.org/survey/255350
- AB/BB: Harrison & Kevin
  - https://uw.iasystem.org/survey/255362
- AC/BC: Mara & Nick
  - https://uw.iasystem.org/survey/255352
- AD/BD: Anirudh & Catherine
  - https://uw.iasystem.org/survey/255354
- AE: Angela & Ian
  - https://uw.iasystem.org/survey/255369

Today: Bugs, Bugs, Bugs 

- Meltdown and Spectre Attacks
- Bugs, Debugging, and Mindfulness

Speculative Execution

The rest of these slides are non-testable

- I haven't quite been honest with you this quarter…
- Modern Intel (and other) CPUs actually work hundreds of instructions ahead of what your program is doing at any given time!
- Branch prediction: CPU tries to predict which branch your program will take, and executes those instructions ahead of time ("speculatively")
  - Based on past program behavior

CPU will eventually figure out that code A is usually executed, and, after a few loop iterations, will speculatively execute code A before the `if` statement is reached!

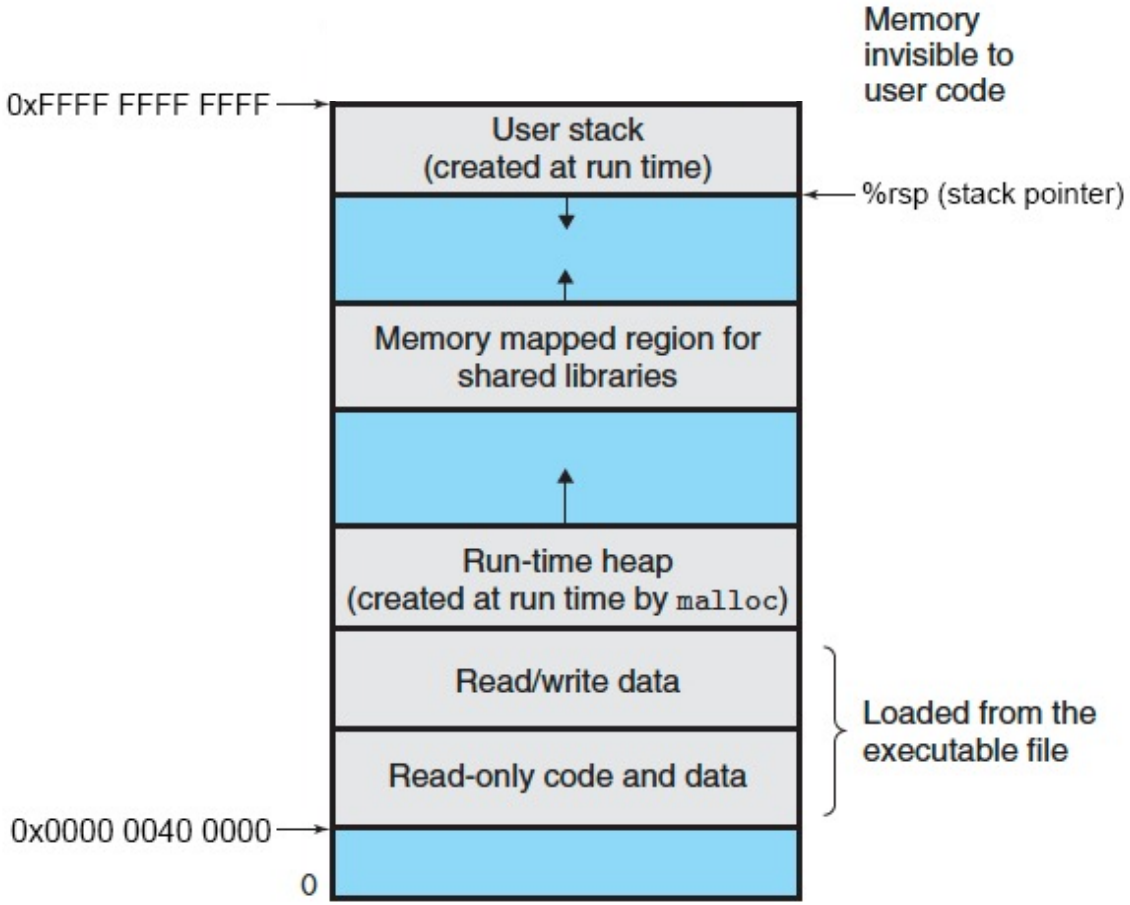
```
while (1) {
  if (rand() > 0.9) {
    // <code A>
  } else {
    // <code B>
  }
}
```

Speculative Execution

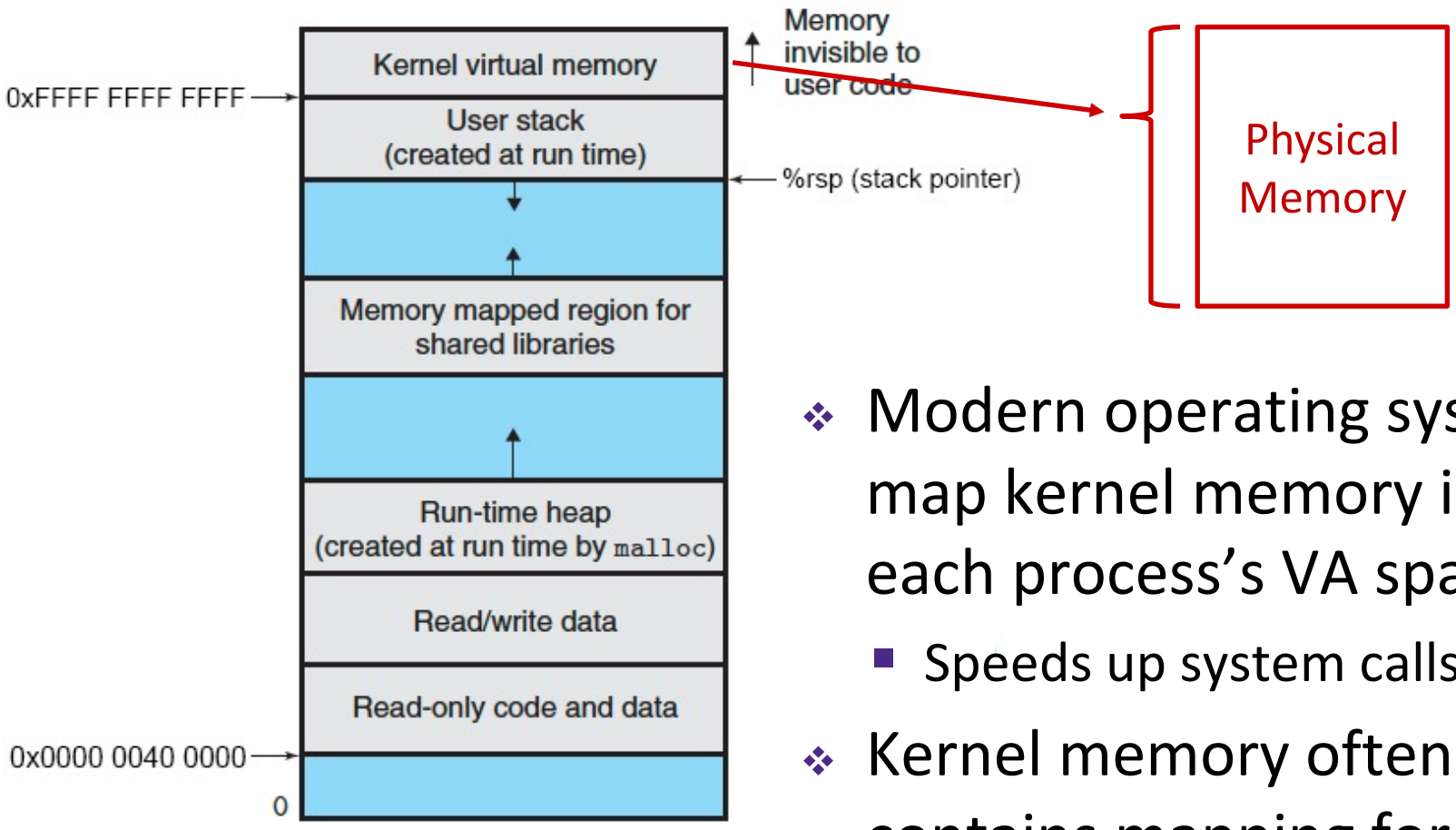
- If the CPU predicts incorrectly, the results are discarded before the program knows they exist.
  - Maintains correctness, just loses a little speed

- However…
  - Instructions executed speculatively **do not trigger exceptions for access to privileged memory locations**
  - And **these instructions affect the cache**

Reminder: Virtual Memory Layout

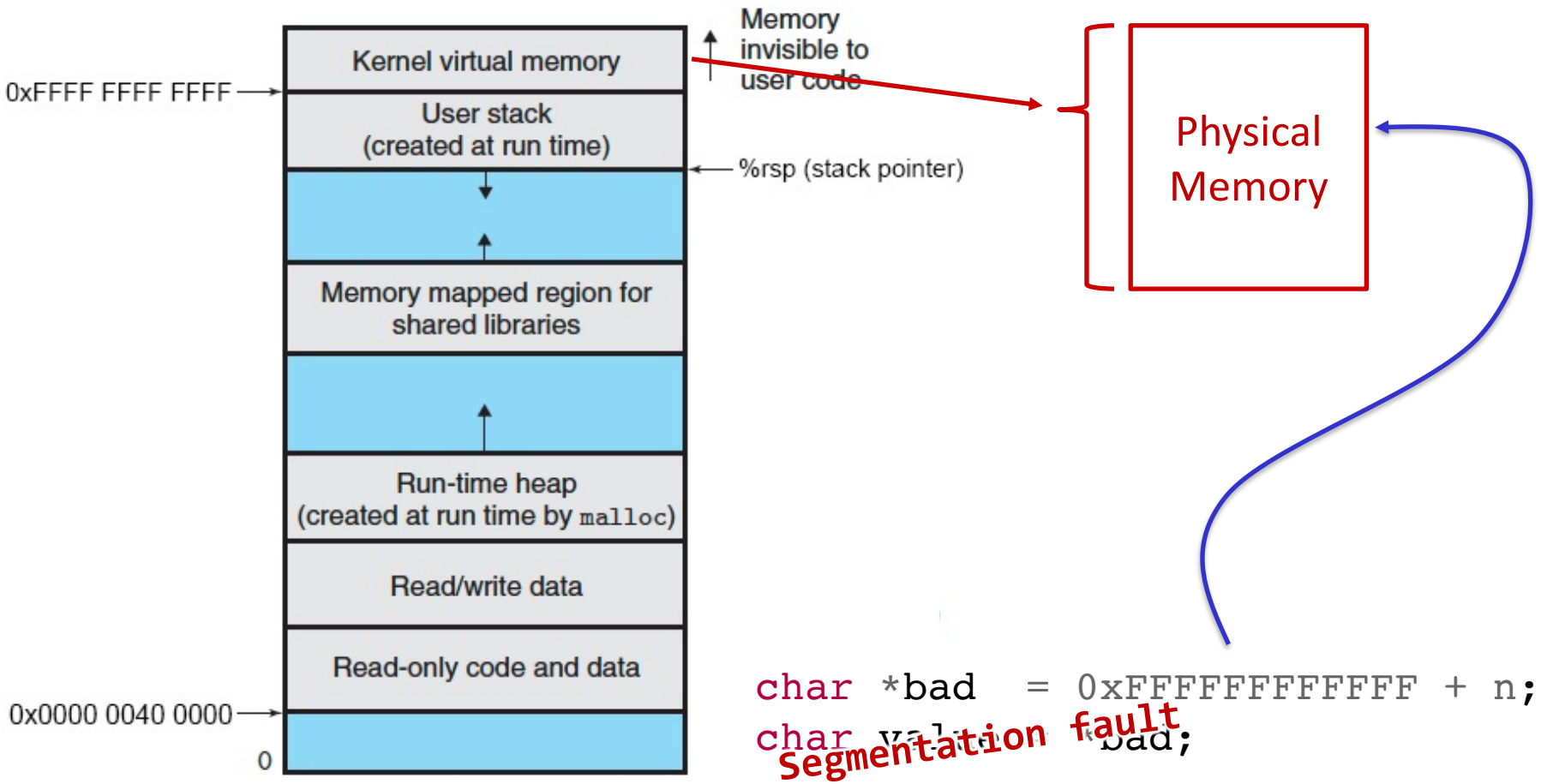


Reminder: Virtual Memory Layout



- Modern operating systems map kernel memory into each process's VA space.
  - Speeds up system calls
- Kernel memory often contains mapping for the computer's **entire physical memory**

Reminder: Virtual Memory Layout




```
char *bad  = 0xFFFFFFFFFFFF + n;
char val = *bad;
```

Segmentation fault

…haha unless??? 🥺

Meltdown Example

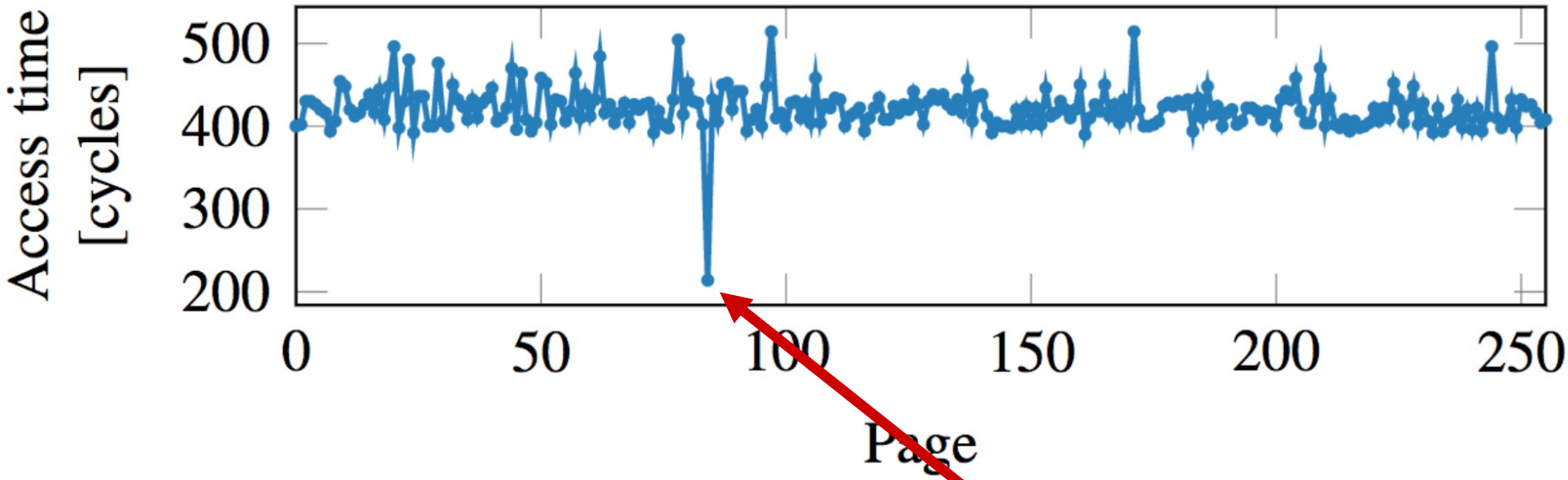
```
#define N <time to access cache>
#define PAGE_SIZE 4096

char array[256 * PAGE_SIZE];

flushCache();

// <train branch predictor>

if (<condition that is always false>) {
  char data = *(<kernel address>);
  char ignored = array[data * PAGE_SIZE];
}

for (int i = 0; i < 255; i++)
  if (accessTime(array[i * PAGE_SIZE]) <= N)
    return i;
```

Recall that kernel memory maps *entire physical memory*!

Only executed speculatively

Meltdown Example

Look for unusually fast accesses to the array




```
for (int i = 0; i < 255; i++)
  if (accessTime(array[i * PAGE_SIZE]) <= N)
    return i;
```

How to Perform An Attack with Meltdown

- Set up an array that is large enough to span 256 pages
  - So we can index into it using a byte
- Speculatively try to access a byte from kernel memory
- Use that as an *index into the array*, multiplied by your system's page size to ensure that adjacent accesses are not cached together
- Try to access each array index (separated by the page size) and time how long it takes.

- The index that took much less time to access corresponds to the secret data!

How to Mitigate Meltdown

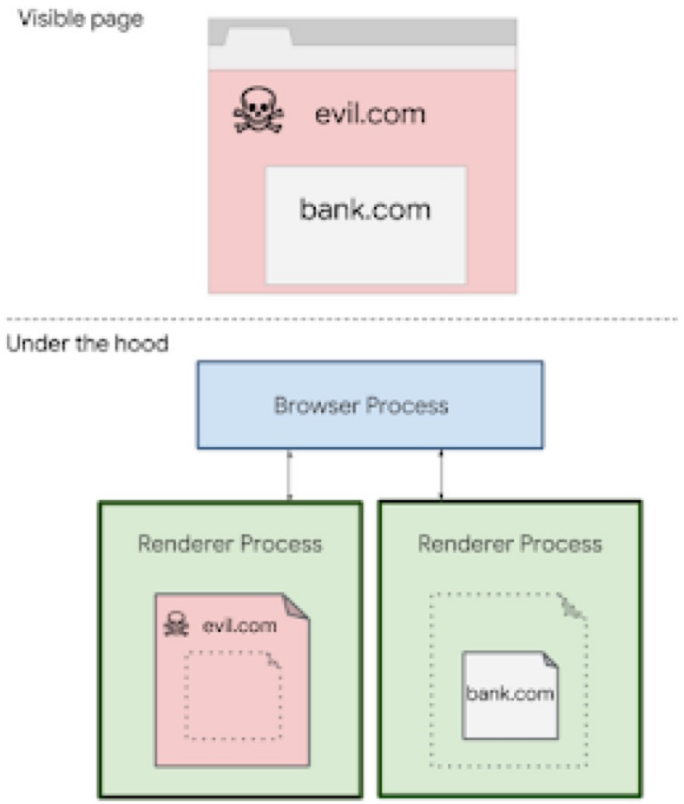
- Don't map kernel memory into user address space
  - This means that system calls are slower, because we need to switch out page tables
- Linux: KAISER
  - Separate page tables for user process and kernel
  - "Shadow" page tables accessible from user processes have extremely limited access to kernel memory, just enough to make system calls
  - https://lwn.net/Articles/738975/
- Further reading
  - https://meltdownattack.com/
  - https://medium.com/@mattklein123/meltdown-spectre-explained-6bc8634cc0c2

Meltdown vs. Spectre

- Spectre works the same way as Meltdown
  - Exploits cache timing and speculative execution
- However, Spectre refers to untrusted code accessing data **within** a process's memory that it doesn't otherwise have a way to access.
  - e.g., JavaScript executed by a web browser that tries to access the browser's private data
- Can't be mitigated in the same way as Meltdown, since it doesn't require kernel memory access

How To Mitigate Spectre

- This must be done by programs themselves (not by the operating system)
  - Harder to mitigate in a universal way – relies on software maintainers
- Site Isolation: each renderer run in a **separate process**
- Reduce granularity of timers that JavaScript can use



Time is a flat circle?

- Prioritizing speed over all else
  - We've seen this before, in buffer overflows!
- We knew this could be a problem!
  - "…we identified several features that, if not properly managed, introduce **previously unreported covert channels and other subtle problems**. These results were surprising; **we did not expect well-defined architectural features to cause undesirable security behavior**. In retrospect, however, the very **complexity of the architecture** suggests that it was bound to include some **unexpected feature interactions**."
  - "…we collected numerous reports of implementation errors claimed to exist in some processor versions…we were surprised to learn of **so many distinct and varied reported implementation errors**."

O. Sibert, P. A. Porras and R. Lindell, "The Intel 80/spl times/86 processor architecture: pitfalls for secure systems," Proceedings 1995 IEEE Symposium on Security and Privacy, **1995**, pp. 211-222, doi: 10.1109/SECPRI.1995.398934.

Time is a flat circle?

- Prioritizing speed over all else
  - We've seen this before, in buffer overflows!
- It's important to be mindful of what you create
  - Understand implications from multiple perspectives
- That being said, nobody is perfect…

You are going to write bugs!

"As soon as we started programming, we found to our surprise that it wasn't as easy to get programs right as we had thought. Debugging had to be discovered. I can remember the exact instant when I realized that a large part of my life from then on was going to be spent in finding mistakes in my own programs."

– *Memoirs of a Computer Pioneer*
by Maurice Wilkes

Debugging Strategies

- You've got to find what works best for you
- Try a lot – your debugging technique should grow over time and some techniques will work better for different domains
  - Print debugging
  - Using a debugger
  - Visualizations
  - Generating thorough test cases/suites
  - Including sensible checks throughout your program
  - etc.
- But this isn't what we're here to talk about now…

Personal debugging tip from Sam: put

```
import pdb; pdb.set_trace()
```

anywhere in your Python code to create a breakpoint on that line!

The Hacker Stereotype

Literally the first result for “hacker” on Google Images

The Hacker Stereotype

- Dark room
- White text on dark background
- Hoodie
- Multiple large monitors

- Obviously this is kind of silly, right?
  - Stereotypes are always oversimplified

- But I still find it tempting to compare myself against this image – why?

Supporting Yourself While Debugging

- CS actively encourages prolonged periods of mental concentration
  - Easy to tune everything else out when you remain immobile just a few feet from your (increasingly large) screen
  - Programmers describe sometimes being "in the zone"
  - Long coding sessions and late nights are socially and culturally encouraged
  - Hackathons are designed this way: encourage you to ignore your bodily needs
    - Why sleep when you can drink Red Bull?
  - Tech companies entice you to stay at work with free food and amenities

Supporting Yourself While Debugging

- When your code doesn't work, it can evoke a lot of different negative emotions
- Strong emotions can impede your thinking ability and scope, which can cause you to spiral
  - Can interact with imposter syndrome, stereotype threat, and other self-esteem issues
- As your mood drops, this can also manifest physically in your body – bad posture, feeling tense, not attending to your needs

Systematic Debugging

- It's easy to get locked into an approach without evaluating others!
  - Maybe if I rearrange these lines?
  - Maybe if I tweak this constant?
  - Maybe if I run it again?
- What if someone periodically asked you to explain and justify what you're doing?
  - *What are you doing?*
  - *Why are you doing it?*
  - *What else have you tried?*
- **Mindfulness:** recognizing your own thinking
  - in other words: *you* be the "someone" ;)

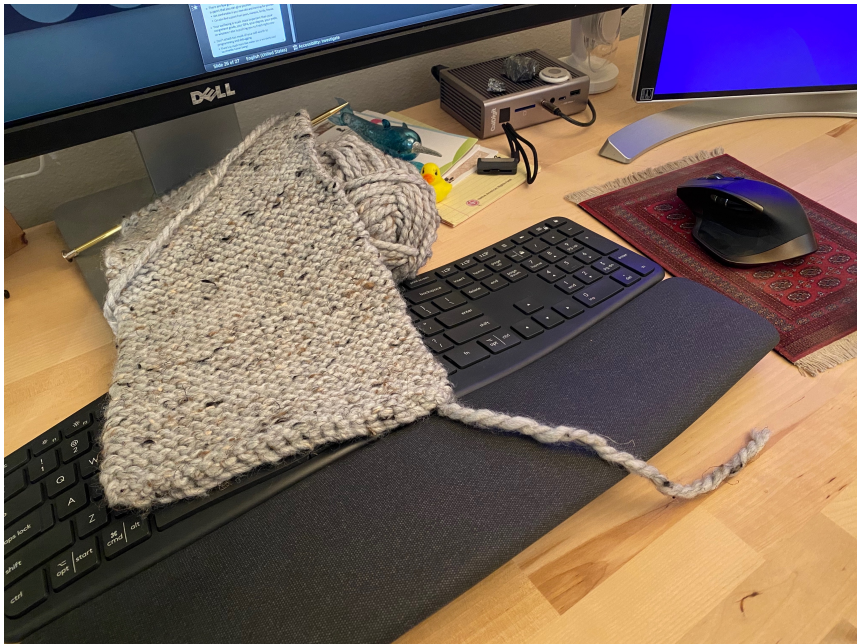
Mindfulness and Debugging

- Mindfulness: "The practice of bringing one's attention in the present moment"
  - Lots of different definitions and nuance, but we'll stick with this broad definition

- While debugging, try to be *mindful* of your emotional and physical state as well as your current approach
  - Am I focused on the task at hand or distracted?
  - Am I calm and/or rested enough to be thinking "clearly?"
  - How is my posture, breathing, and tenseness?
  - Do I have any physical needs that I should address?
  - What approach am I trying and **why**? Are there alternatives?

Supporting Yourself While Debugging

- Try: set a timer for some interval (e.g., 15 minutes) to evaluate your state and approach
  - Like the system timer your OS uses for context switching!
- If you're distracted, feeling negative emotions, tense, or need to address something, ***take a break!***
  - You might even think of a new approach!
  - Knit a scarf, go for a walk, rollerblade, take a nap, cook dinner, bake bread, learn the melodica…



Supporting Yourself

- There are few guarantees for support, besides the support that you can give yourself
  - Get comfortable in your own skin and stand up for yourself
  - Look to your peers, mentors, family, friends! Make use of the support systems that you do have!
- Your wellbeing is much more important than your assignment grade, your GPA, your degree, your pride, or whatever else is pushing you to finish *right now*

Computer Science and You

- Being mindful of how you work best is important
- CS culture tends to perpetuate certain images of what being a programmer "should" look like
  - This might work for you! And that's great!
- But it might not, which is also completely valid
  - **Regardless, you still belong in computer science**
  - Silly personal examples: I use light mode, I prefer lots of ambient (ideally natural) light, I don't use a mechanical keyboard, I've never pulled an all-nighter (at least not to write code), …
- The best approaches are the ones that work for you. And they don't change your value as a computer scientist or as a person.