Paul E. McKenney, Meta Platforms Kernel Team

Kernel Recipes, September 27, 2023

Hardware and its Concurrency Habits

https://mirrors.edge.kernel.org/pub/linux/kernel/people/paulmck/perfbook/perfbook.html Chapter 3

Recette Pour le Codage Simultané

- Une pincée de connaissance des lois de la physique
- Compréhension modeste du matériel informatique
- Compréhension approfondie des exigences
- Conception soignée, y compris la synchronisation
- Validation brutale

https://mirrors.edge.kernel.org/pub/linux/kernel/people/paulmck/perfbook/perfbook.html Chapitre 3

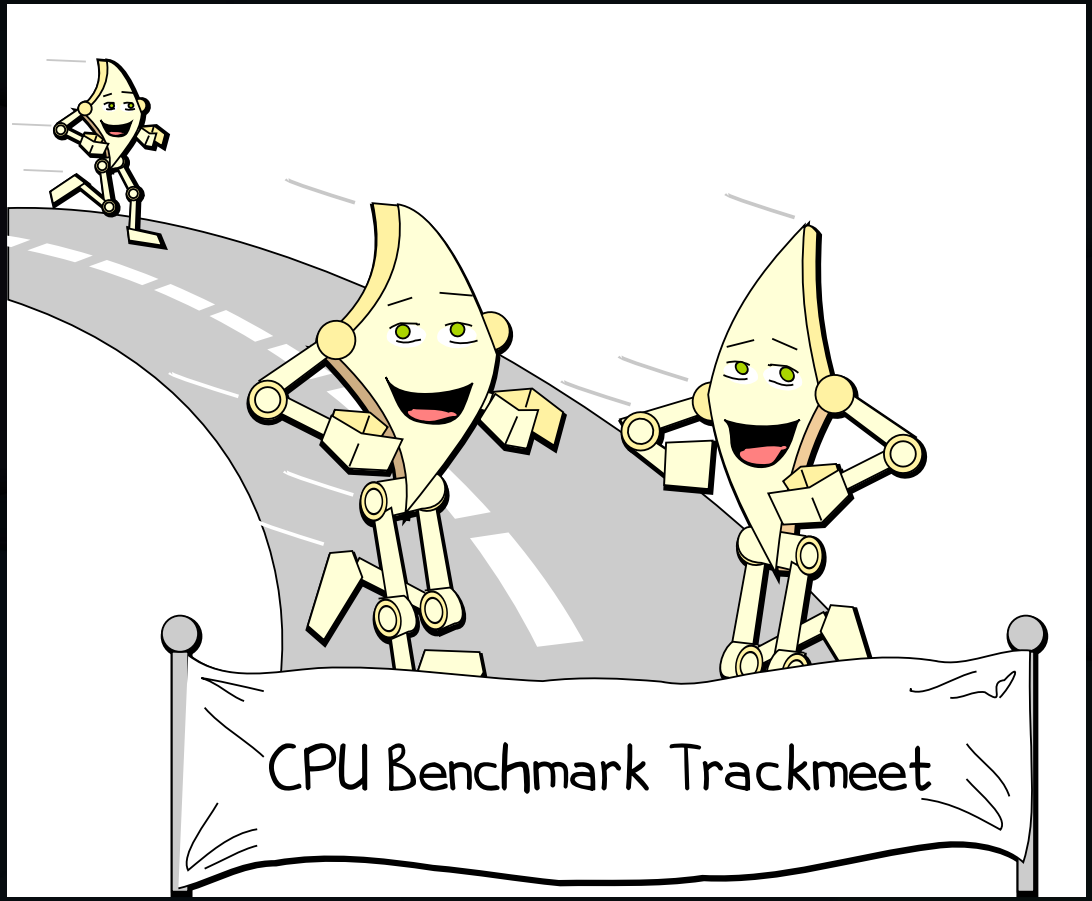
Recette Pour le Codage Simultané

- Une pincée de connaissance des lois de la physique
- Compréhension modeste du matériel informatique
- Compréhension approfondie des exigences
- Conception soignée, y compris la synchronisation
- Validation brutale

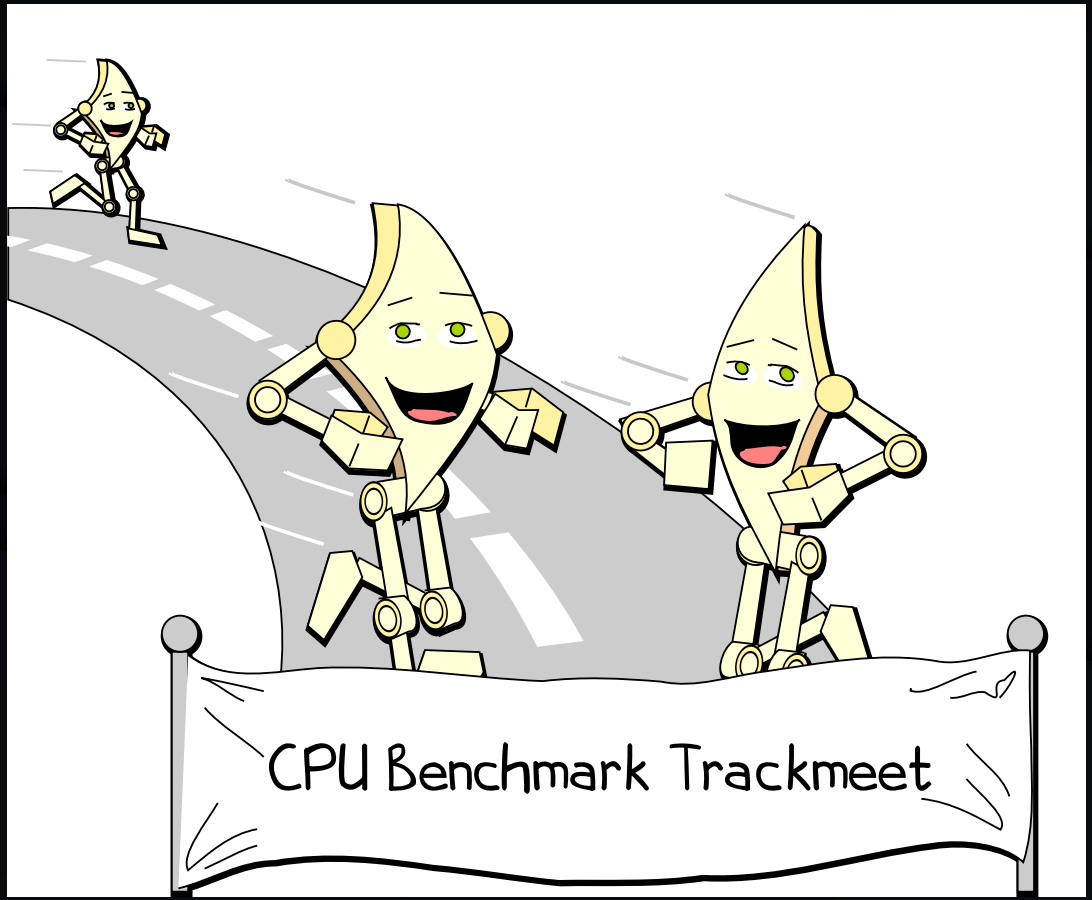
https://mirrors.edge.kernel.org/pub/linux/kernel/people/paulmck/perfbook/perfbook.html Chapitre 3

“Let Them Run Free!!!”

“Let Them Run Free!!!”



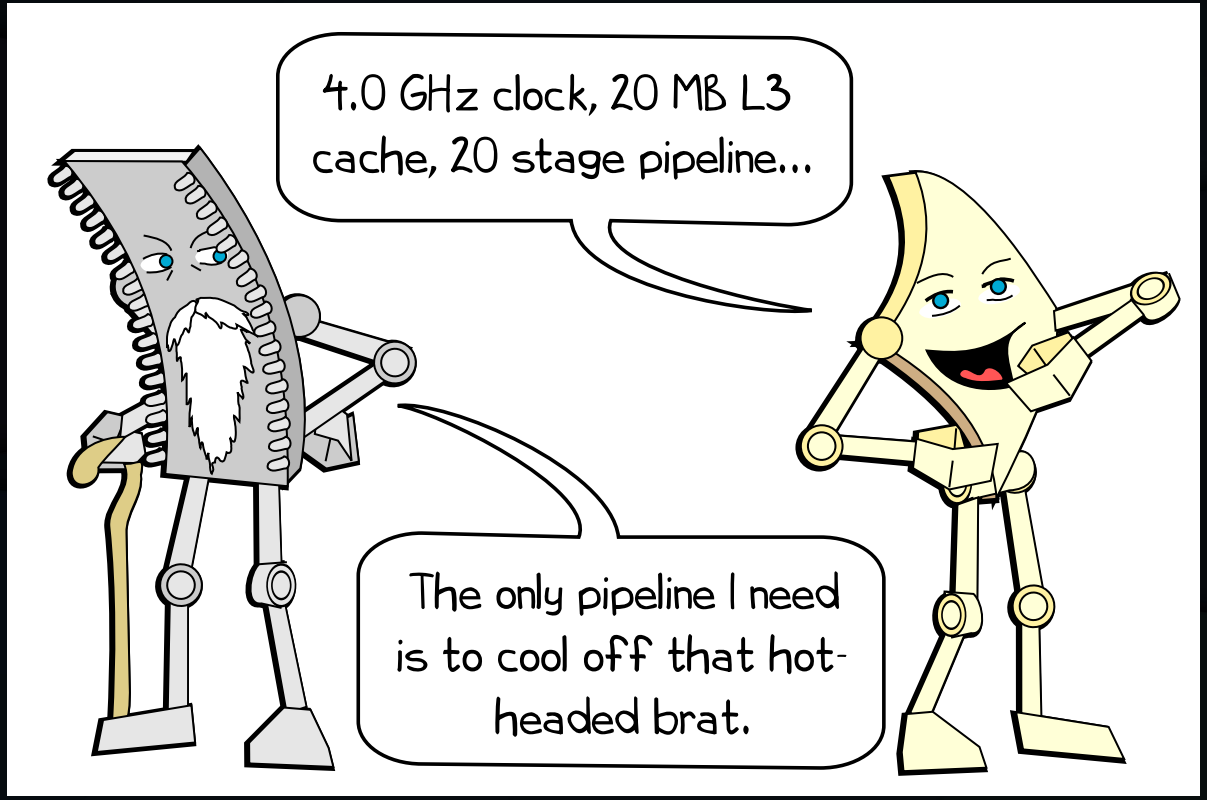
“Let Them Run Free!!!”



Sadly, it is now more of an obstacle course than a track...

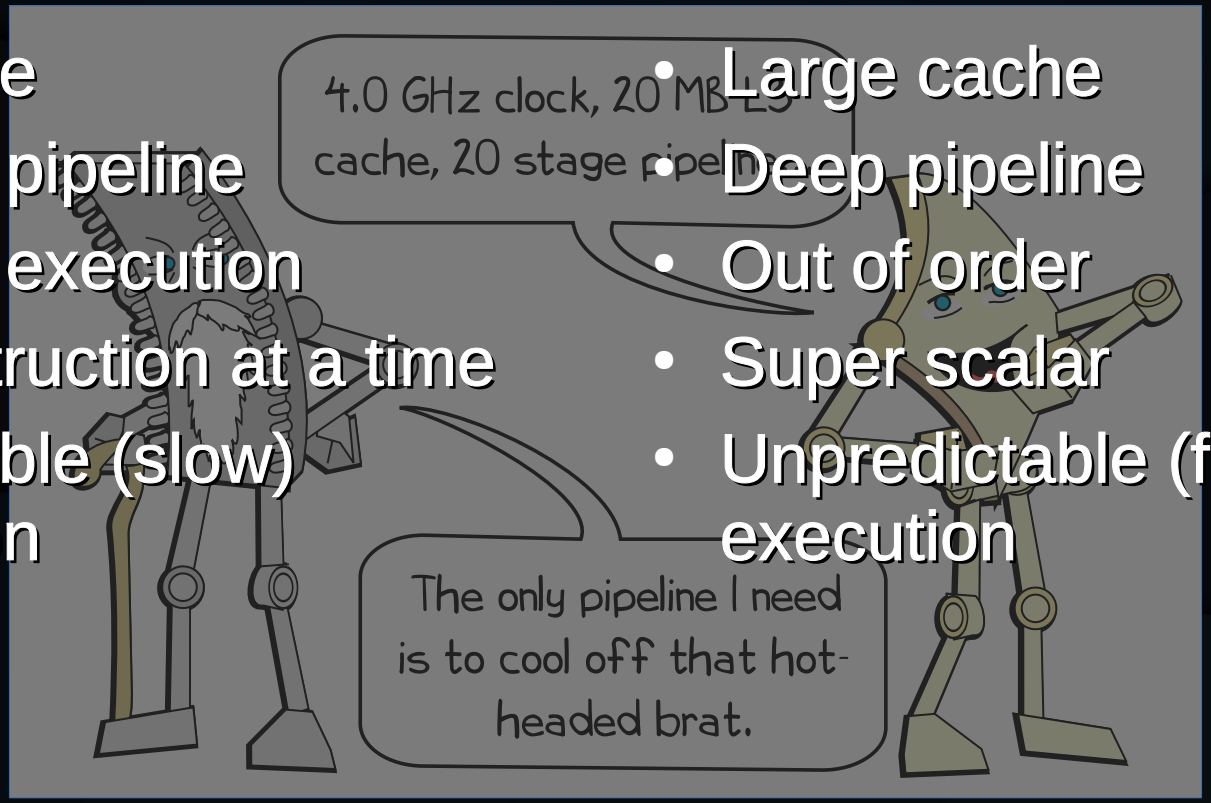
Don’t Make ‘em Like They Used To!

Don’t Make ‘em Like They Used To



Don’t Make ‘em Like They Used To!

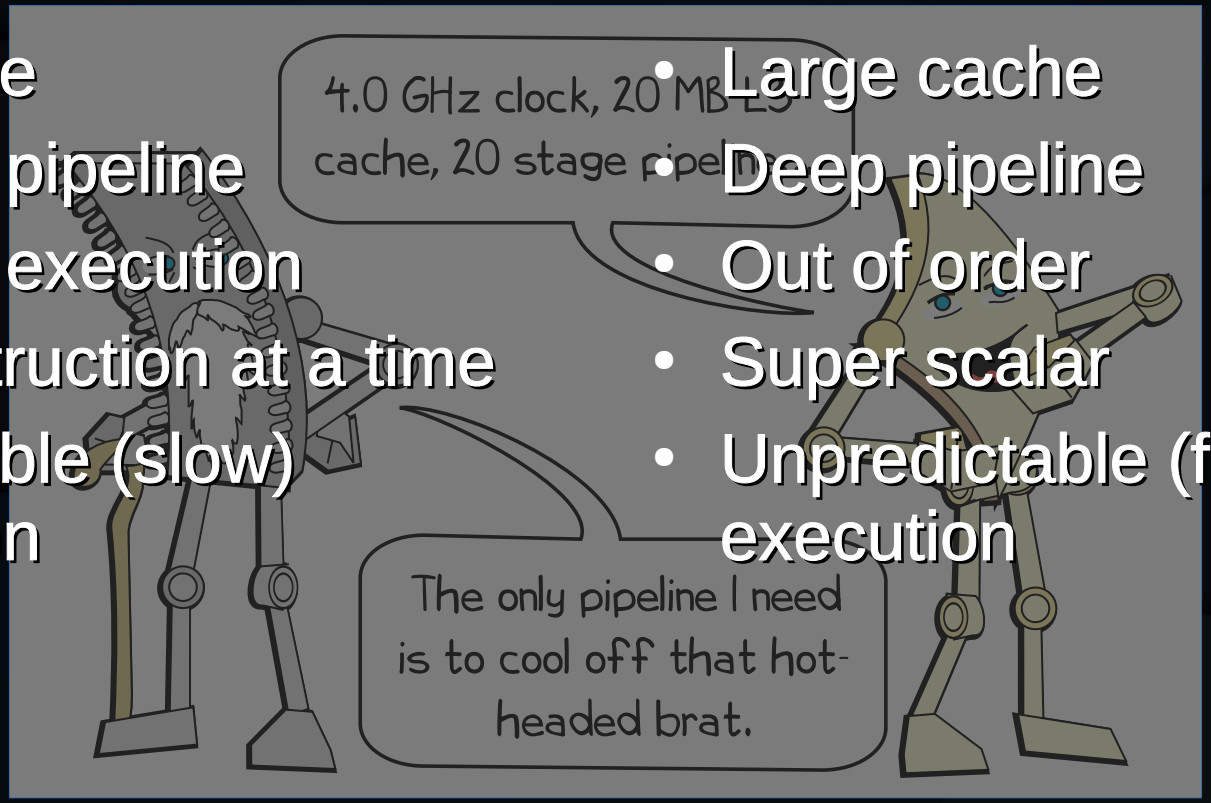
- No cache
- Shallow pipeline
- In-order execution
- One instruction at a time
- Predictable (slow) execution



- Large cache
- Deep pipeline
- Out of order
- Super scalar
- Unpredictable (fast) execution

Don’t Make ‘em Like They Used To!

- No cache
- Shallow pipeline
- In-order execution
- One instruction at a time
- Predictable (slow) execution

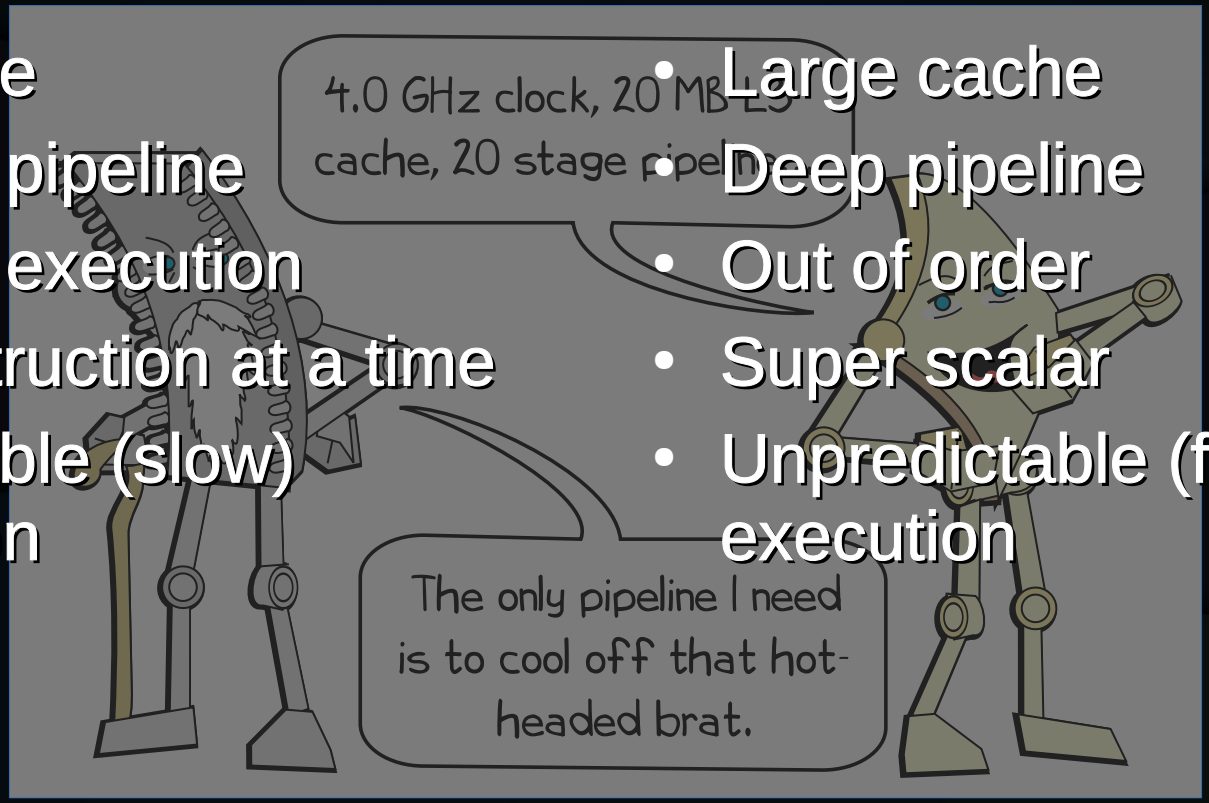


- Large cache
- Deep pipeline
- Out of order
- Super scalar
- Unpredictable (fast) execution

“Tiny Bulldozer” “Semi Tractor-Trailer”

Don’t Make ‘em Like They Used To!

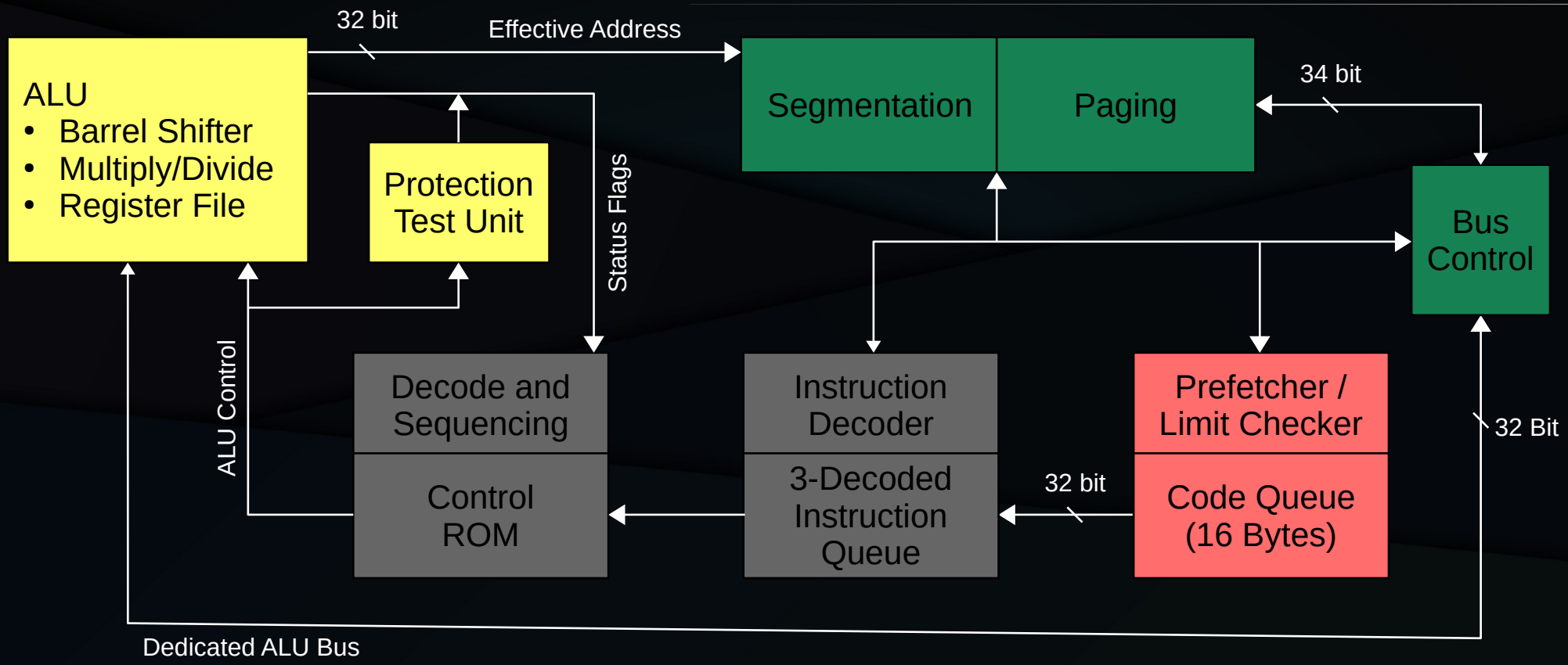
- No cache
- Shallow pipeline
- In-order execution
- One instruction at a time
- Predictable (slow) execution



- Large cache
- Deep pipeline
- Out of order
- Super scalar
- Unpredictable (fast) execution

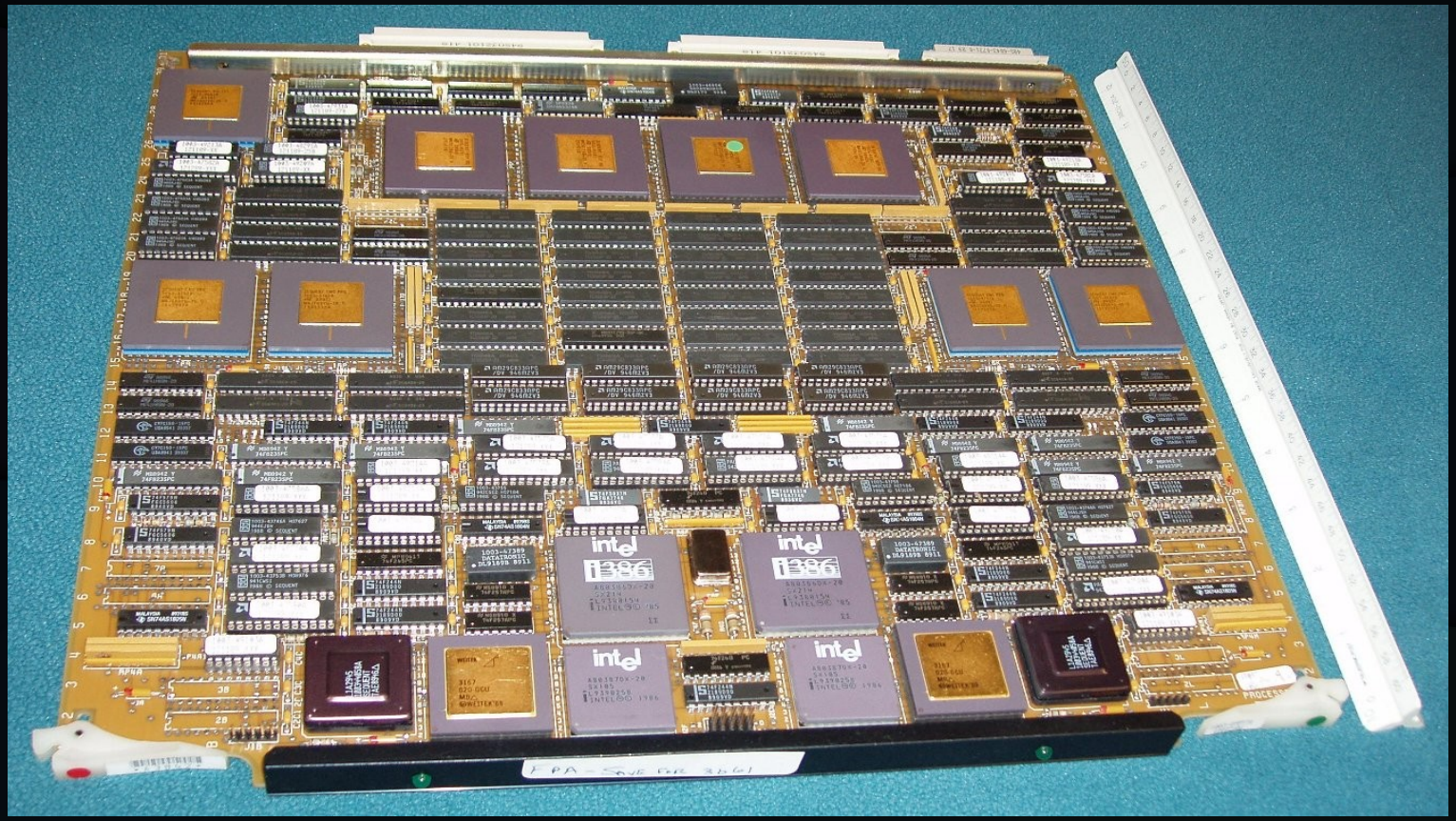
What would be the computing-systems equivalents of a freight train?

“Good Olde Days” CPU Architecture



80386 Architecture (Wikipedia user “Appaloosa” GDFL, simplified and reformatted)

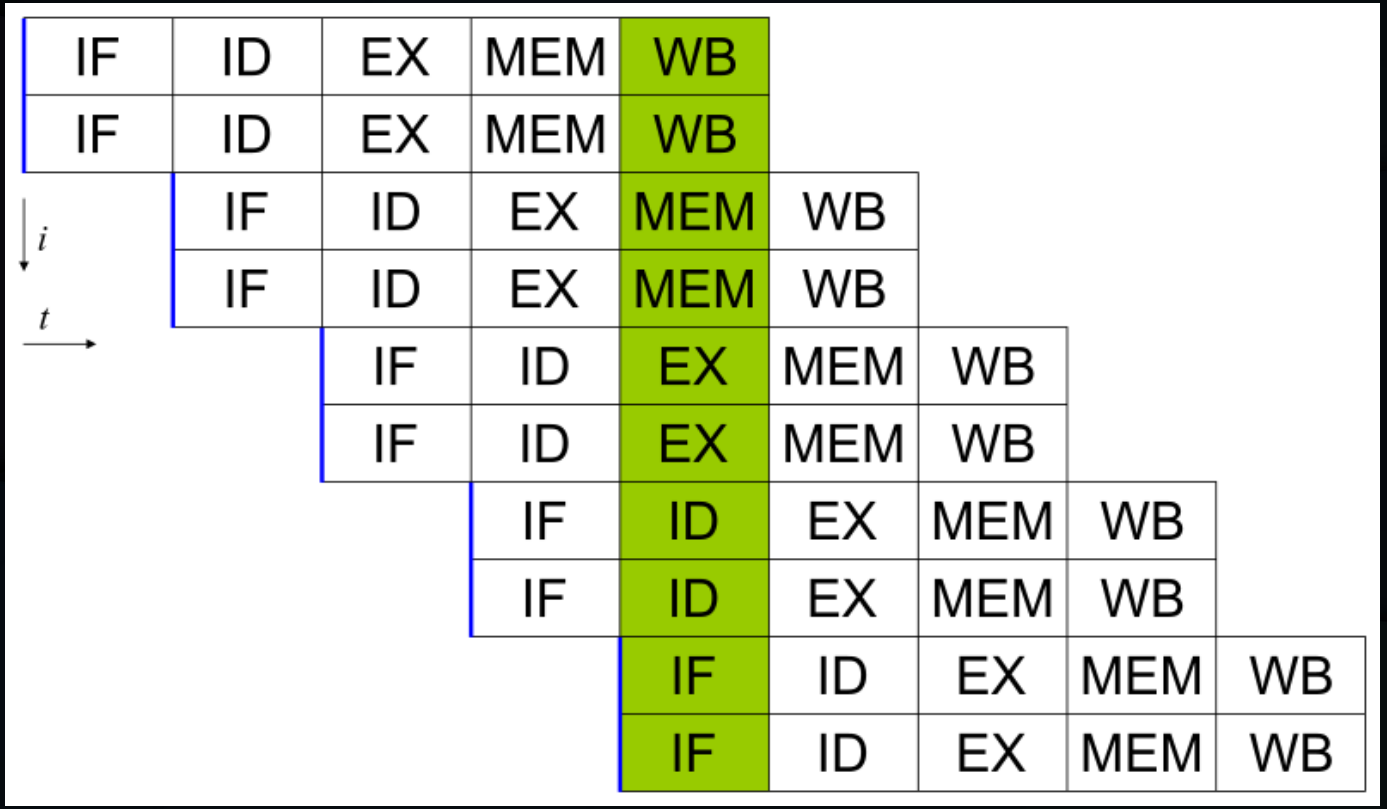
The 80386 Taught Me Concurrency



That and a logic analyzer...

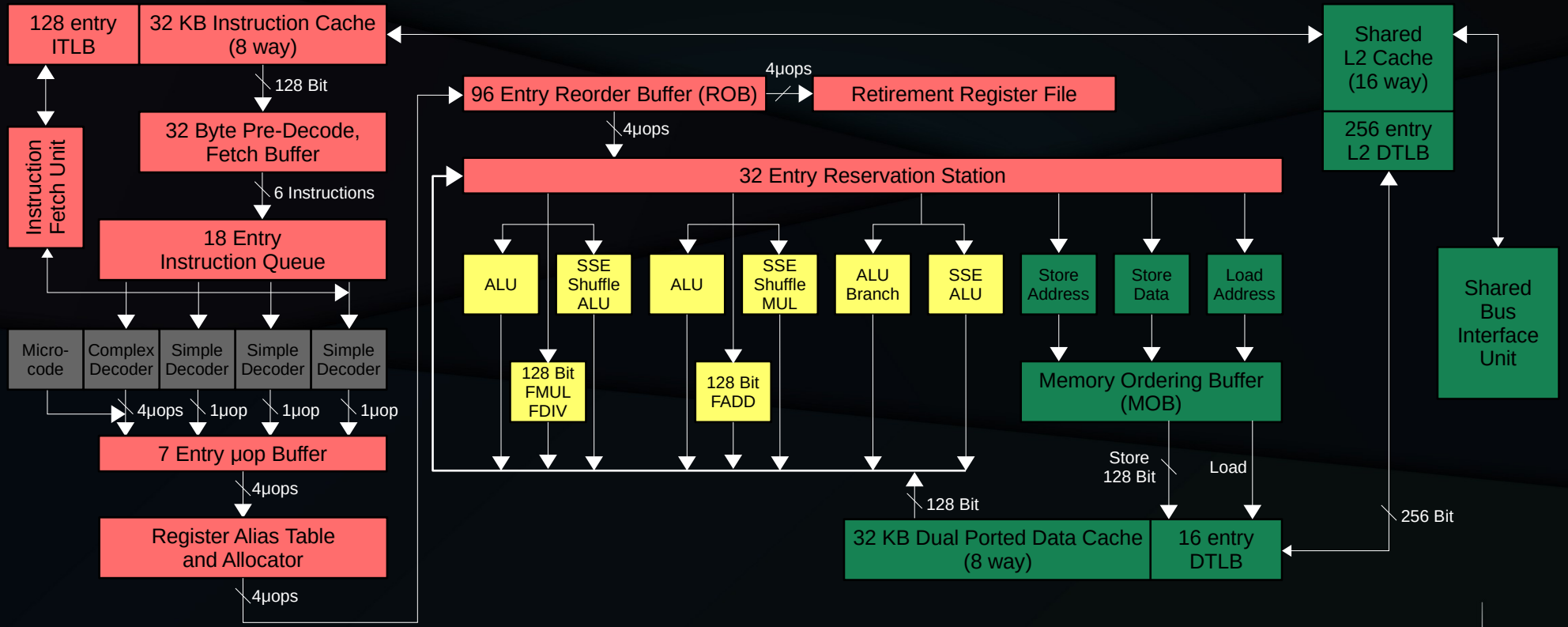
But Instructions Took Several Cycles!

Pipelined Execution For The Win!!!



(Wikipedia user “Amit6” CC BY-SA 3.0, reformatted)

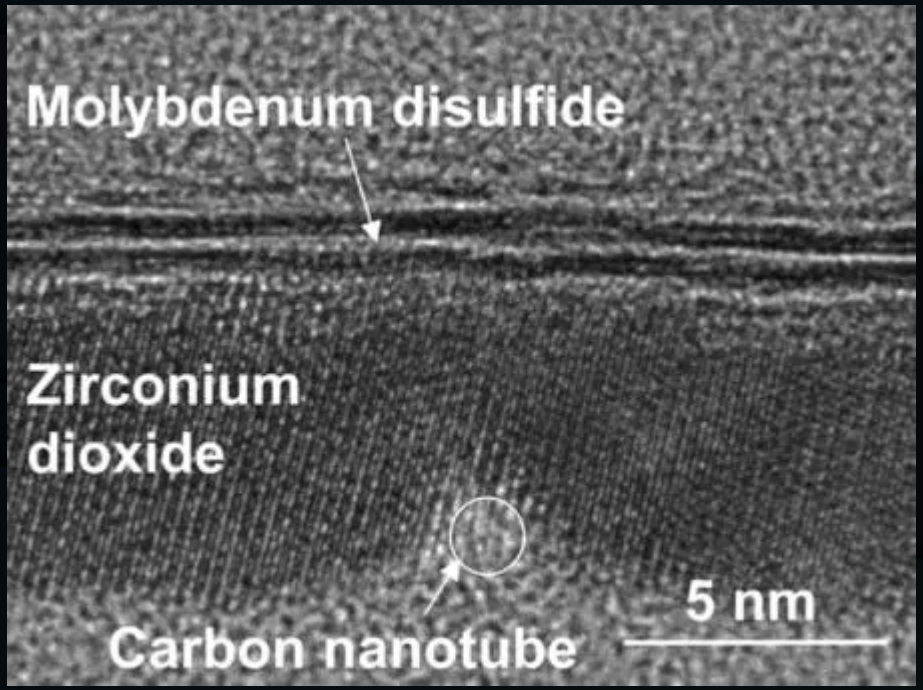
Superscalar Execution For The Win!!!



Intel Core 2 Architecture (Wikipedia user “I, Appaloosa” CC BY-SA 3.0, reformatted)

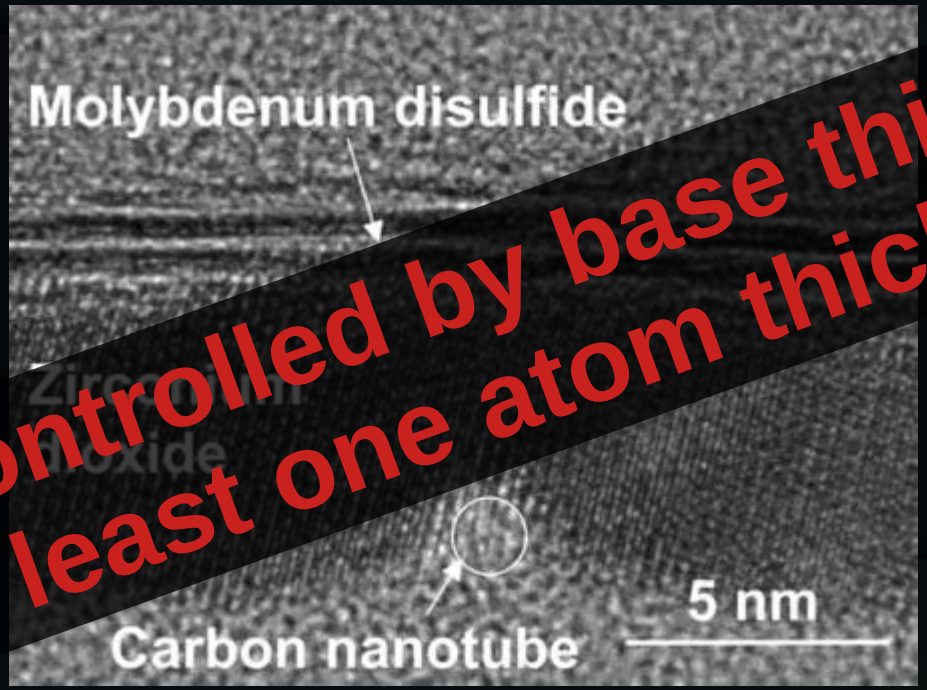
Why All This Hardware Complexity?

Laws of Physics: Atoms Are Too Big!!!



Each spot is an atom. Qingxiao Wang/UT Dallas ca. 2016.

Laws of Physics: Atoms Are Too Big!!!



Speed controlled by base thickness:
At least one atom thick!!!

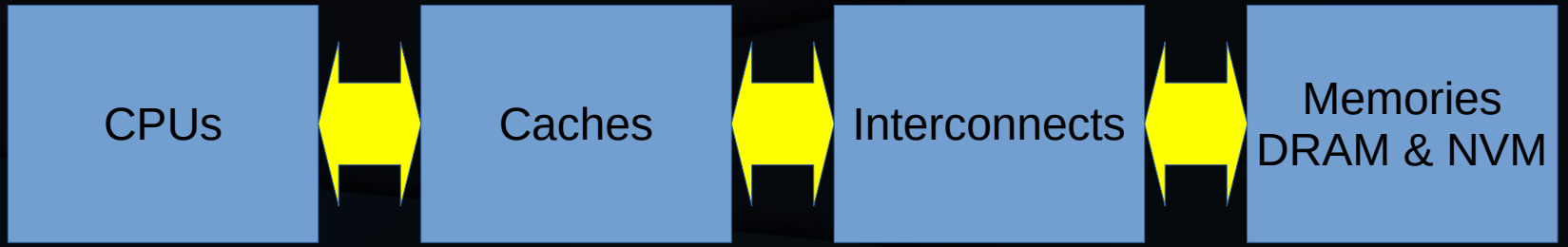
Each spot is an atom. Qingxiao Wang/UT Dallas ca. 2016.

Laws of Physics: Light Is Too Slow!!!

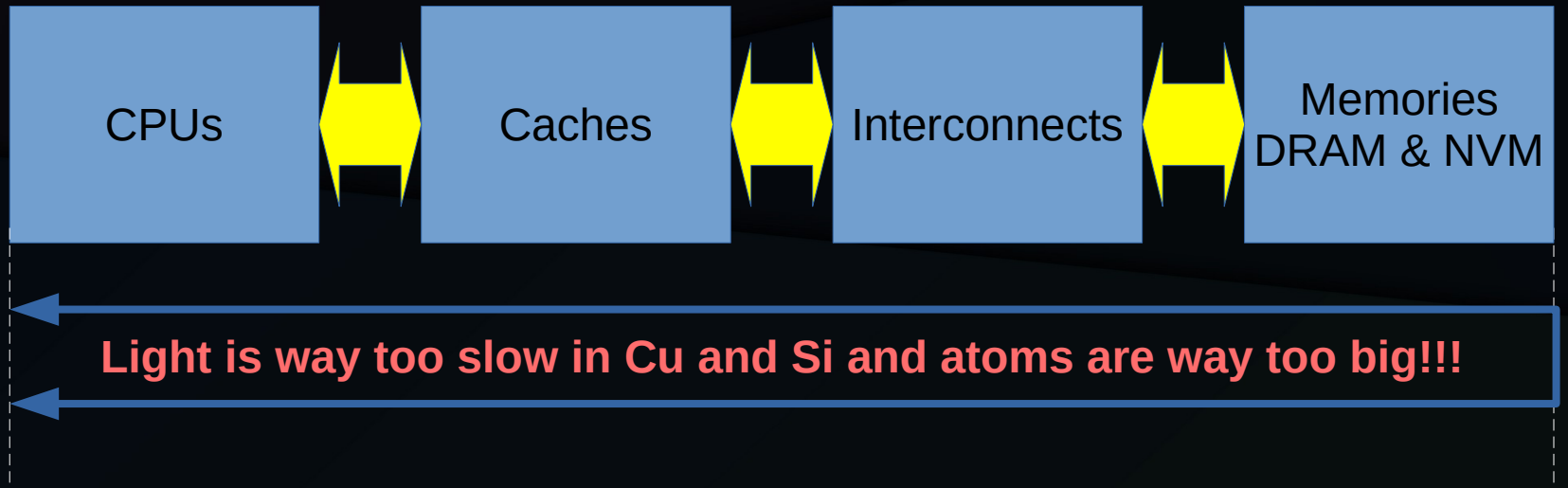
- Following the footsteps of Admiral Hopper:
  - Light goes 11.803 inches/ns in a vacuum
    - Or, if you prefer, 1.0097 lengths of A4 paper per nanosecond
    - Light goes 1 ***width*** of A4 paper per nanosecond in 50% sugar solution
  - But over and back: 5.9015 in/ns
  - But not 1GHz! Instead, ~2GHz: ~3in/ns
  - But Cu: ~1 in/ns, or Si transistors: ~0.1 in/ns
  - Plus other slowdowns: prototols, electronics, ...

“One nanosecond per foot” courtesy of Grace Hopper (https://www.youtube.com/watch?v=9eyFDBPk4Yw)
https://en.wikipedia.org/wiki/List_of_refractive_indices A 50% sugar solution is “light syrup”.

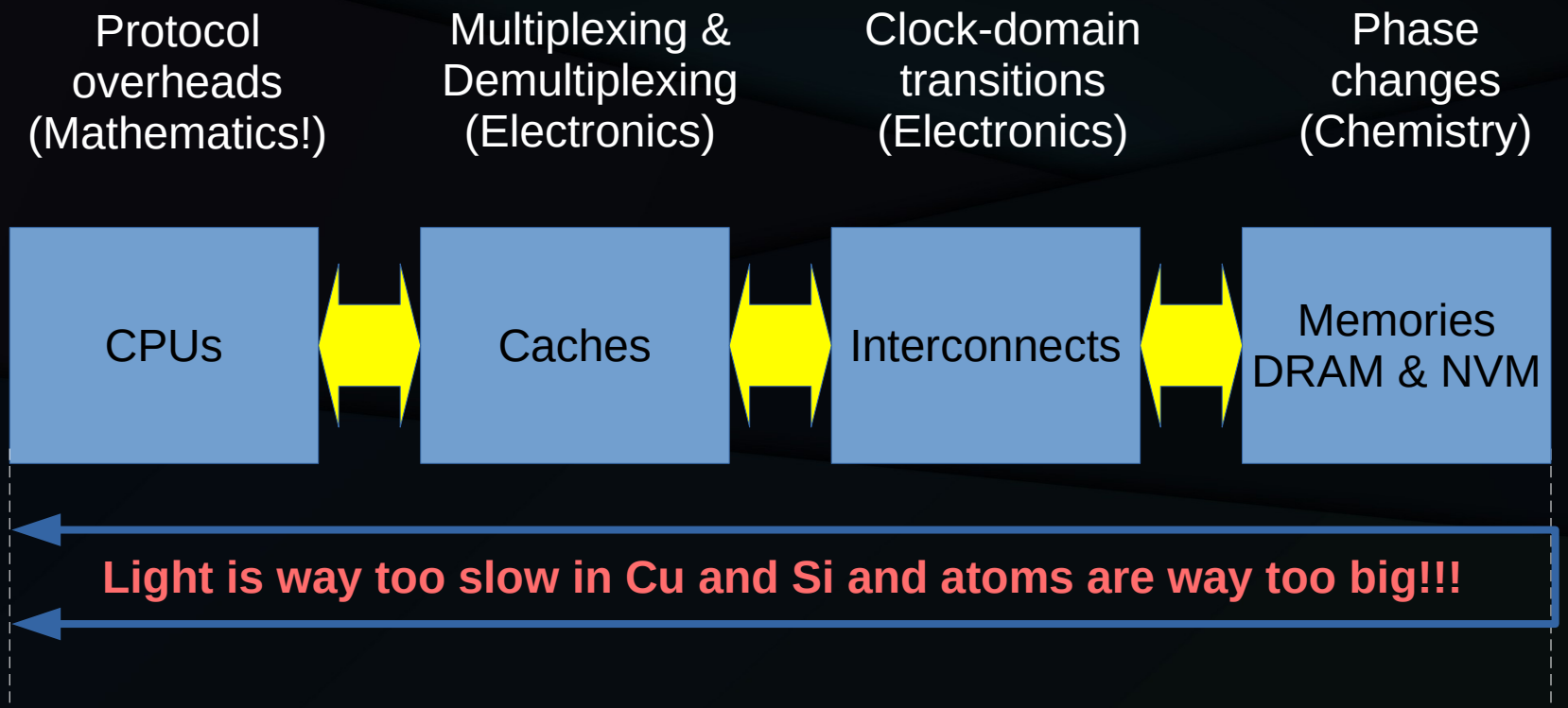
Laws of Physics: Data Is Slower!!!



Laws of Physics: Data Is Slower!!!



Laws of Physics: Data Is Slower!!!



Laws of Physics: Summary

- The speed of light is finite (especially in Cu and Si) and atoms are of non-zero size
- Mathematics, electronics, and chemistry also take their toll
- Systems are fast, so this matters

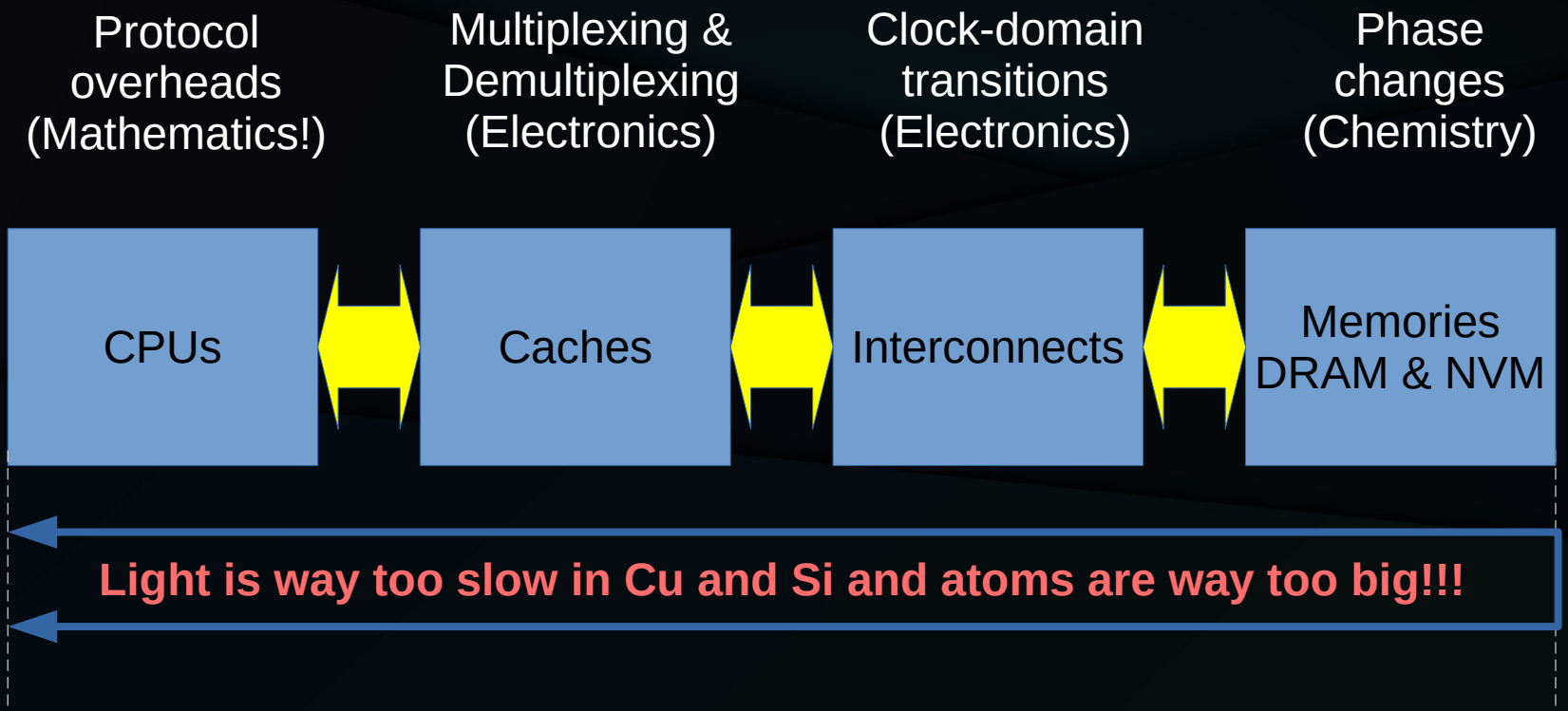
Laws of Physics: Summary

- The speed of light is finite (especially in Cu and Si) and atoms are of non-zero size
- Mathematics, electronics, and chemistry also take their toll
- Systems are fast, so this matters

“Gentlemen, you have two fundamental problems:
(1) the finite speed of light and (2) the atomic nature of matter.” *

* Gordon Moore quoting Stephen Hawking

Why All This Hardware Complexity?

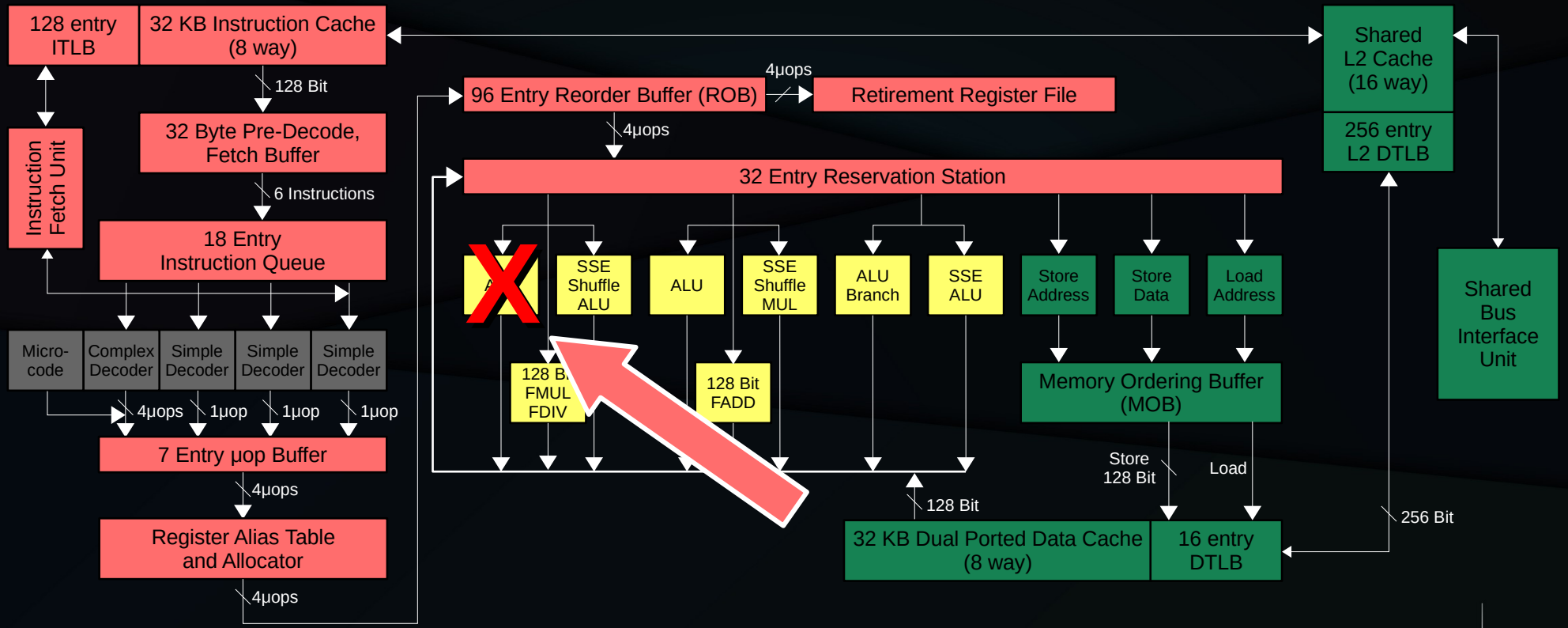


Slow light and big atoms create modern computing obstacle course!!!

Account For All CPU Complexity???

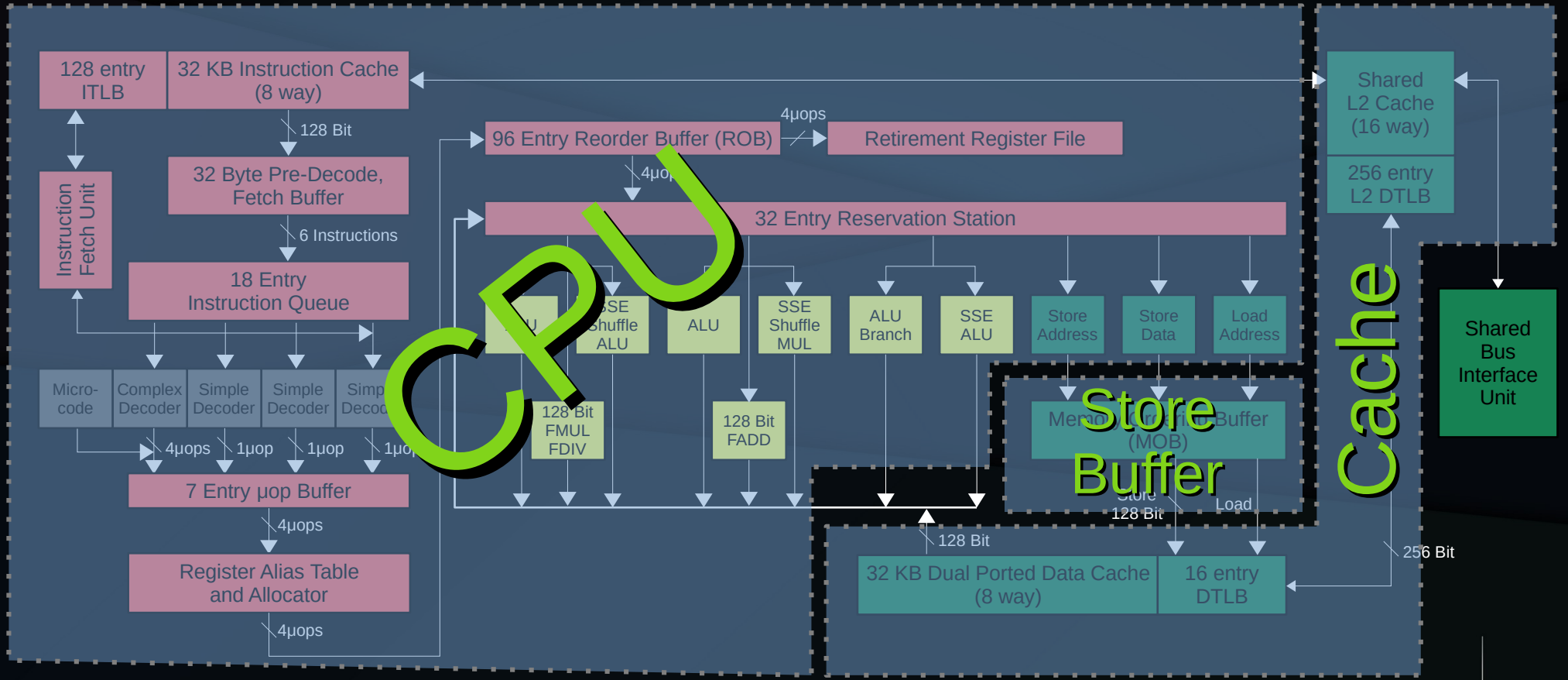
- Sometimes, yes! (Assembly language!)
- But we also need portability: CPUs change
  - From family to family
  - With each revision of silicon
  - To work around hardware bugs
  - As a given physical CPU ages

One of the ALUs Might Be Disabled



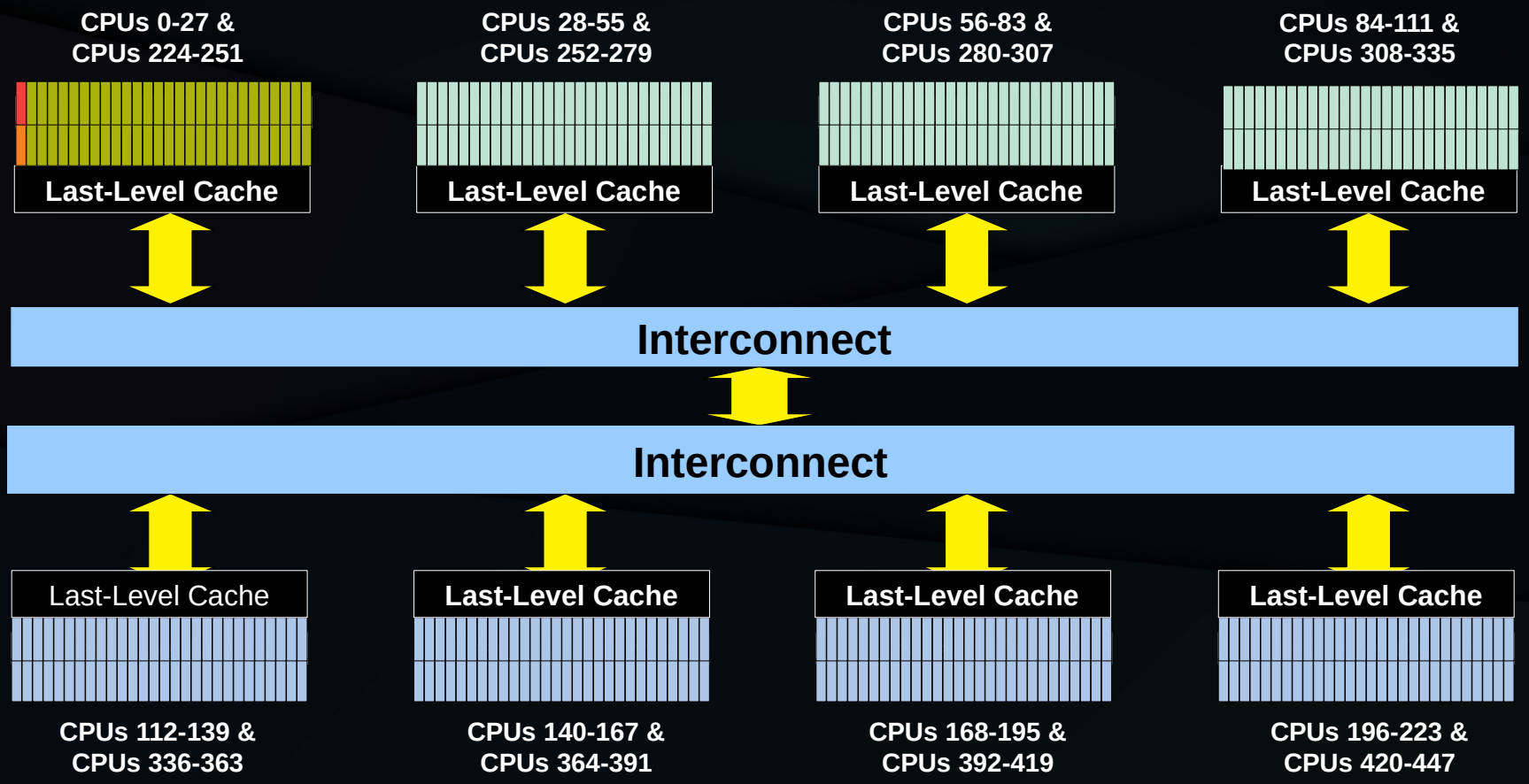
Intel Core 2 Architecture (Wikipedia user “I, Appaloosa” CC BY-SA 3.0, reformatted)

Thus, Simple Portable CPU Model



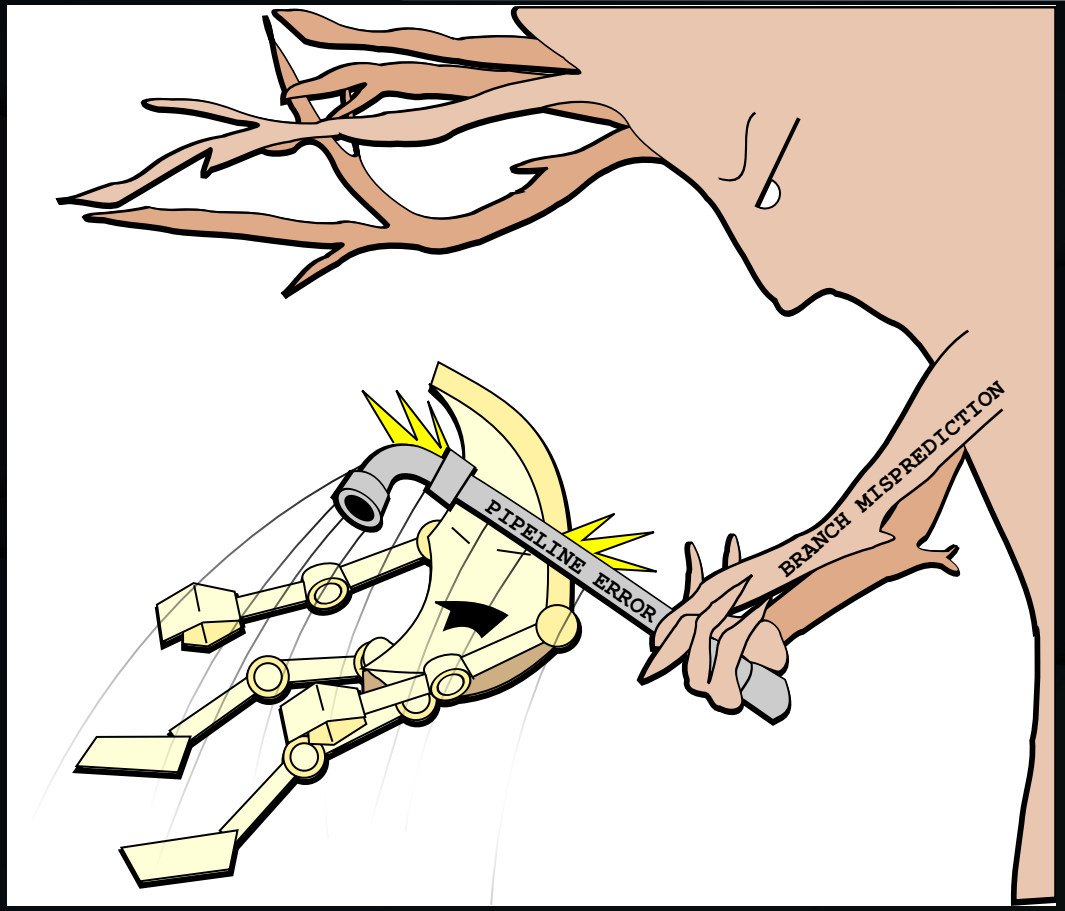
Intel Core 2 Architecture (Wikipedia user “I, Appaloosa” CC BY-SA 3.0, reformatted and remixed)

And Lots Of CPUs Per System!!!



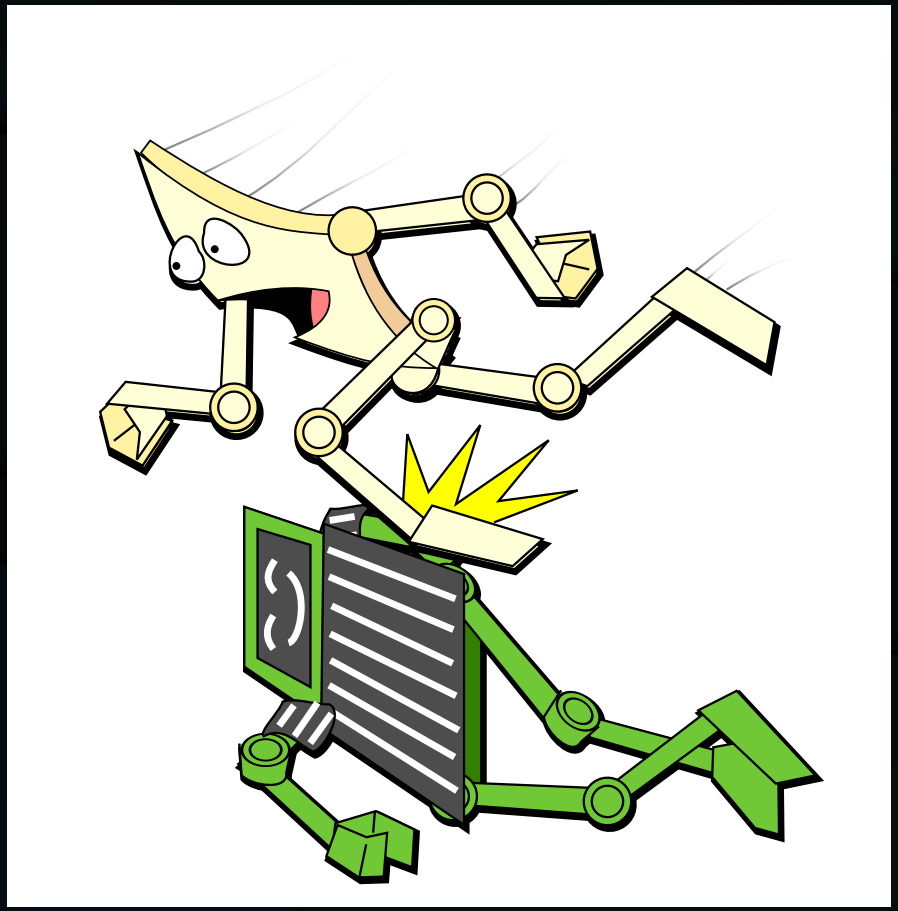
Obstacles for Modern Computers

Obstacle: Pipeline Flush



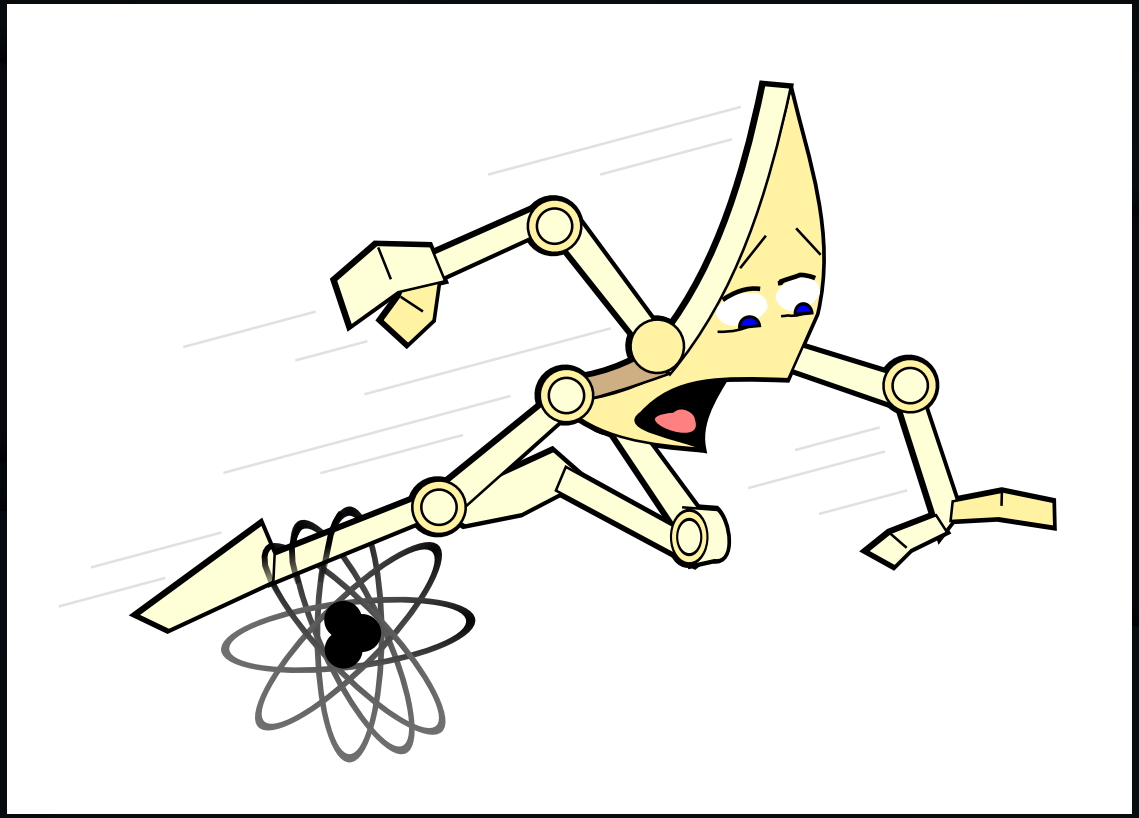
Running at full speed requires perfect branch prediction

Obstacle: Memory Reference



A single fetch all the way from memory can cost hundreds of clock cycles

Obstacle: Atomic Operation

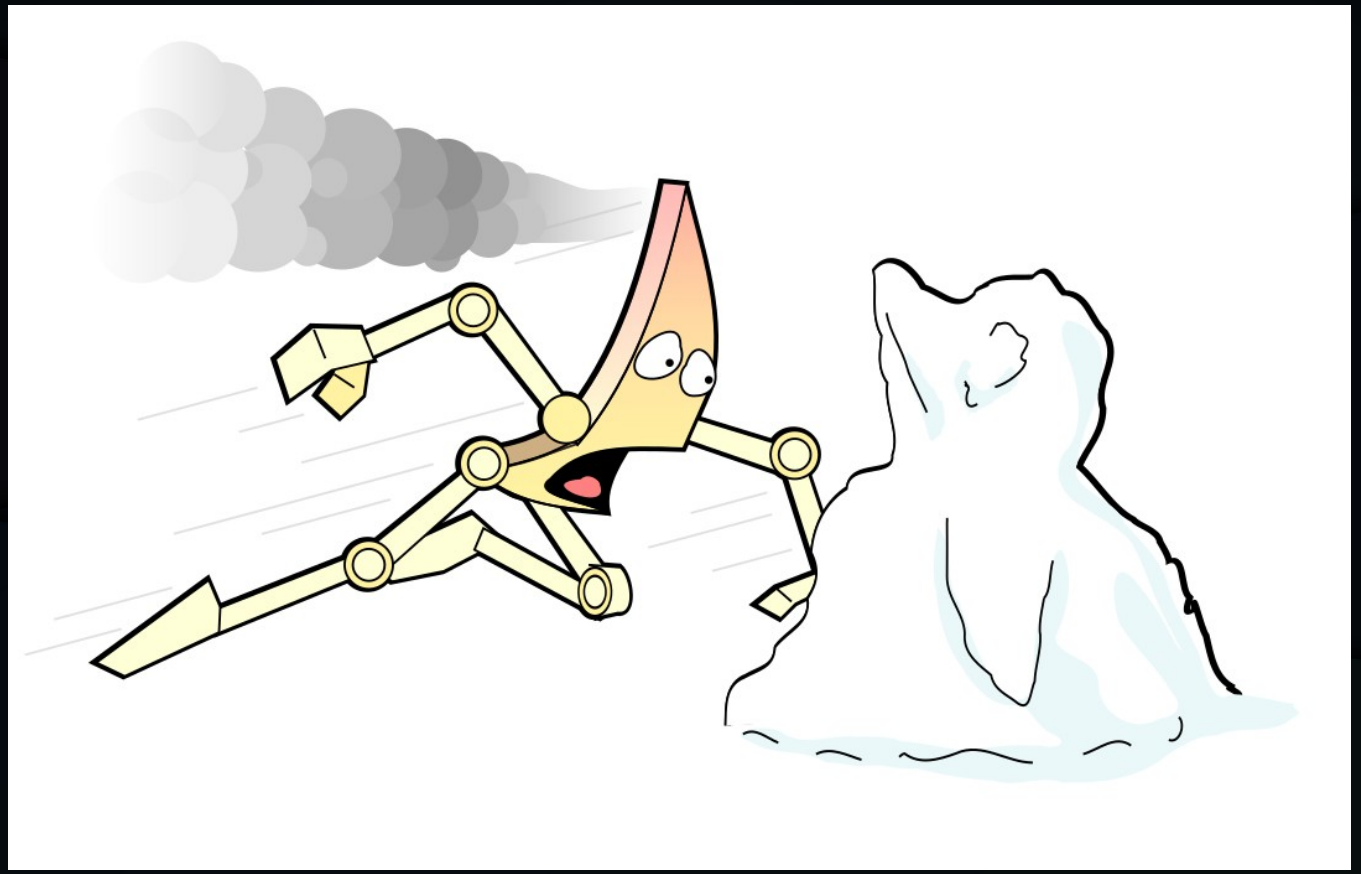


Atomic operations require locking cachelines and/or busses, incurring significant delays

Obstacle: Memory Barrier

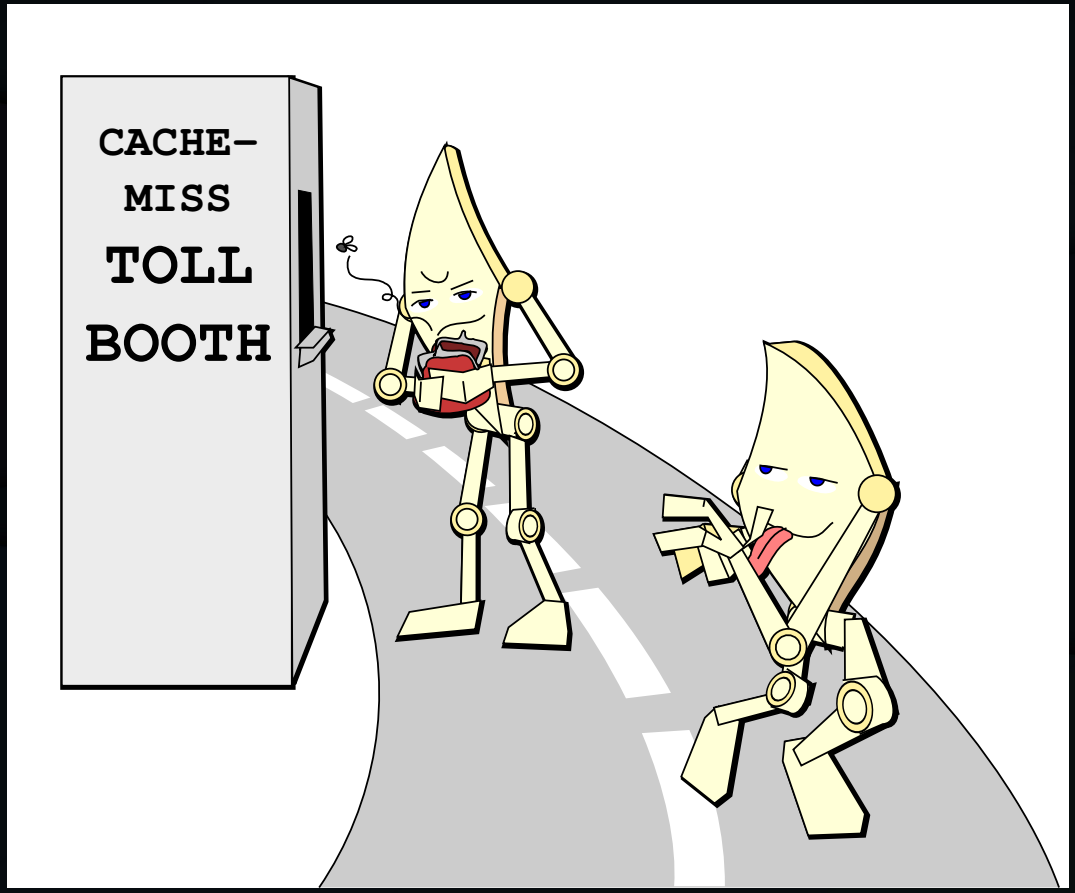
Memory barriers result in stalls and/or ordering constraints, again incurring delays

Obstacle: Thermal Throttling



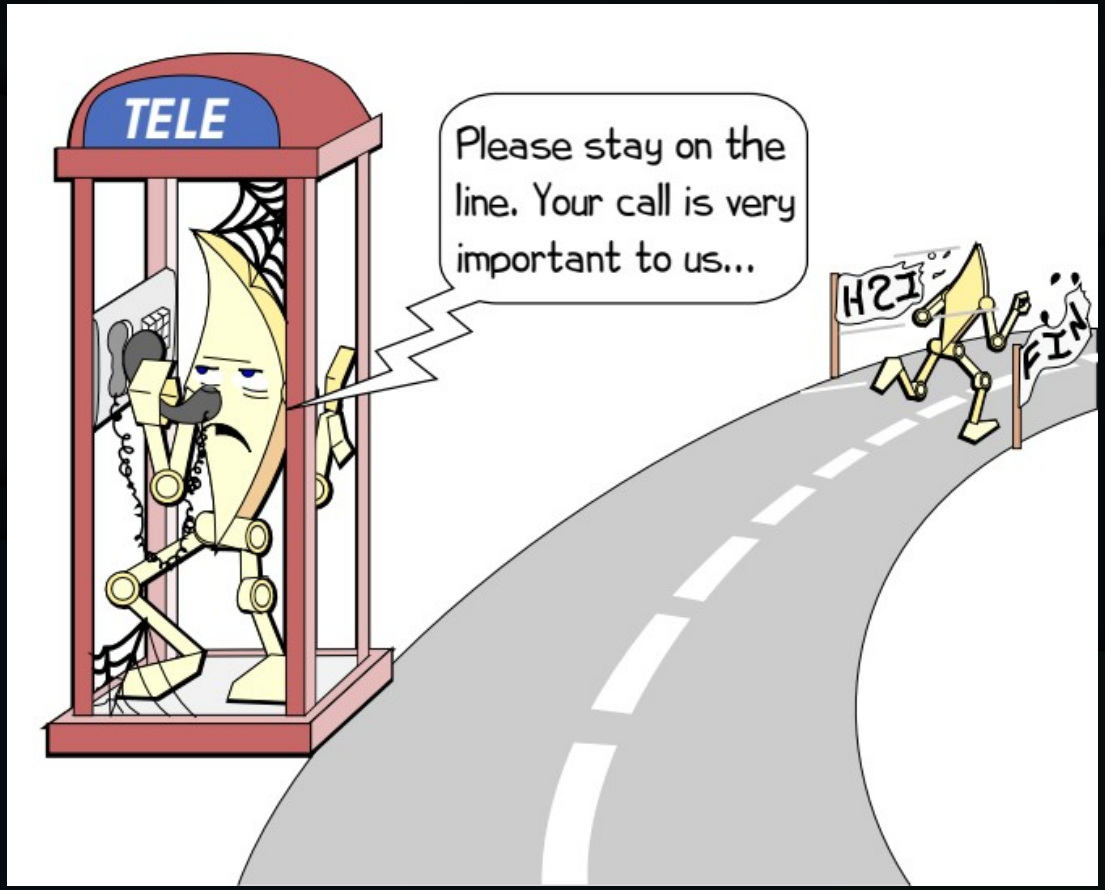
Efficient use of CPU hardware generates heat, throttling the CPU clock frequency

Obstacle: Cache Miss



Cache misses result in waiting for data to arrive (from memory or other CPUs)

Obstacle: Input/Output Operation



And here you thought that cache misses were slow...

Which Obstables To Focus On?

1) I/O operations (but often higher-level issue)
2) Communications cache misses
3) Memory barriers and atomic operations
4) Capacity/geometry cache misses (memory)
5) Branch prediction

Which Obstables To Focus On?

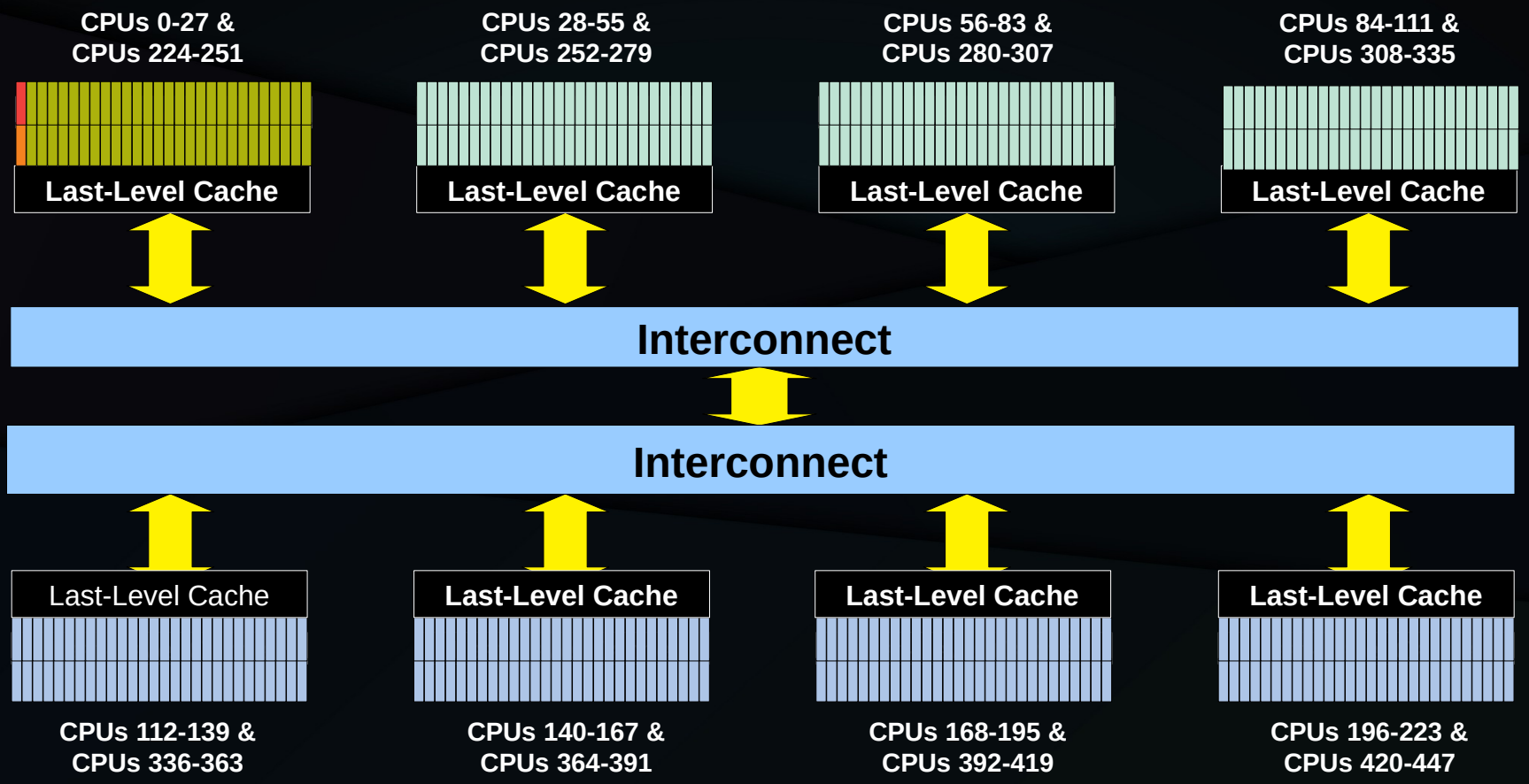
1) I/O operations (but often higher-level issue)
2) **Communications cache misses**
3) **Memory barriers and atomic operations**
4) Capacity/geometry cache misses (memory)
5) Branch prediction

These obstacles can (usually) be overcome in a portable manner.

Xeon Platinum 8176 2.1GHz: CPU 0

Operation	Cost (ns)	Ratio (cost/clock)	CPUs
Clock period	0.5	1.0	
Same-CPU CAS	7.0	14.6	0
Same-CPU lock	15.4	32.3	0
In-core blind CAS	7.2	15.2	224
In-core CAS	18.0	37.7	224
Off-core blind CAS	47.5	99.8	1–27,225–251
Off-core CAS	101.9	214.0	1–27,225–251
Off-socket blind CAS	148.8	312.5	28–111,252–335
Off-socket CAS	442.9	930.1	28–111,252–335
Cross-interconnect blind CAS	336.6	706.8	112–223,336–447
Cross-interconnect CAS	944.8	1,984.2	112–223,336–447

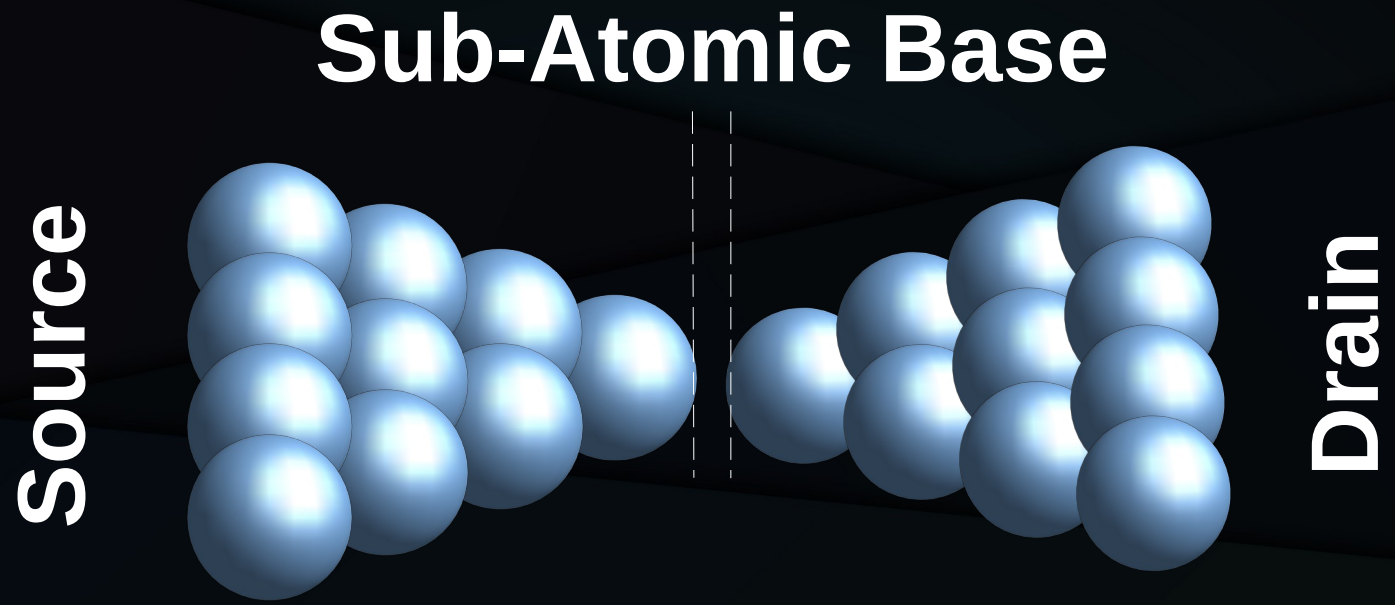
Location Really Matters!!!



Latency Demonstration

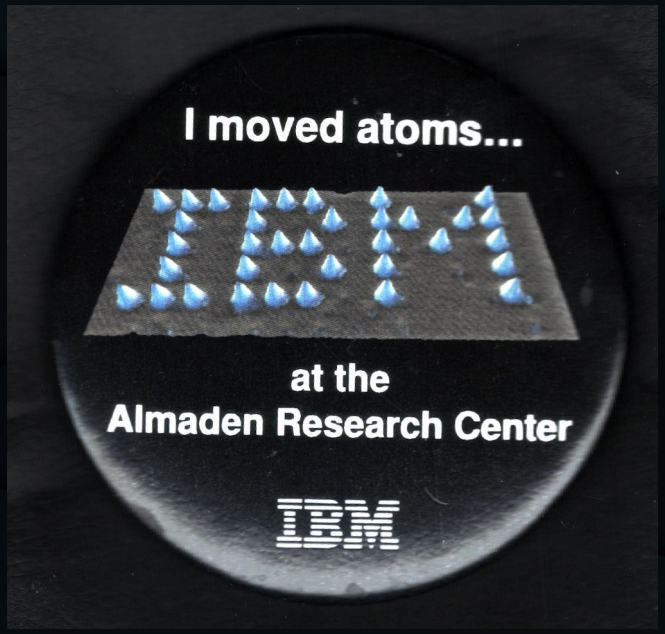
Can Hardware Help???

Can Hardware Help???



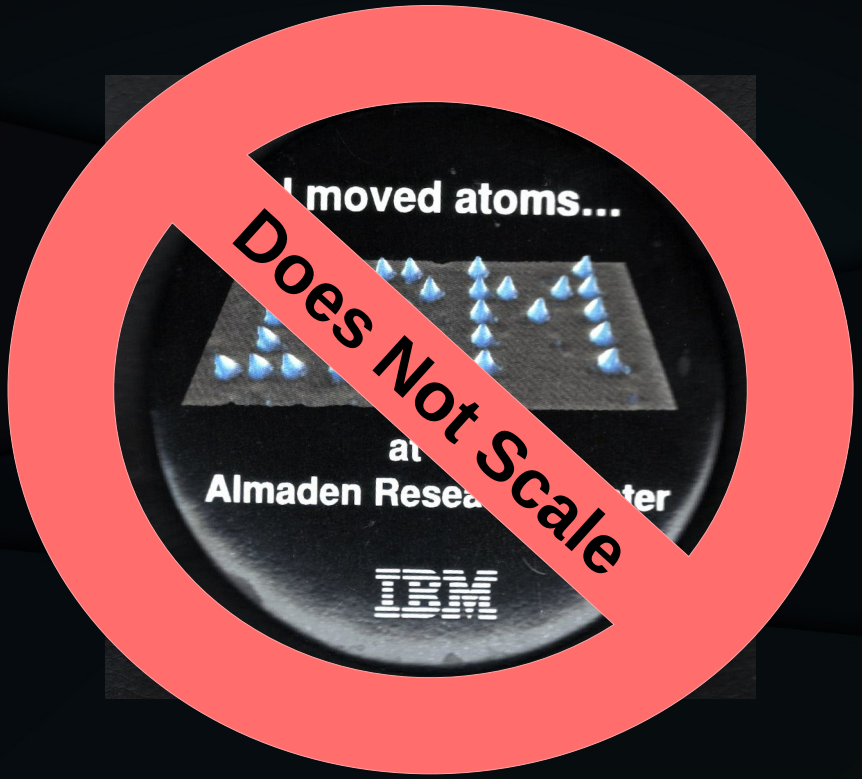
Vacuum-gap transistor: At these scales, the atmosphere is a vacuum!!!

We Really Can Hand-Place Atoms...



Actually a carbon monoxide **molecule** that I moved across a few planes of copper

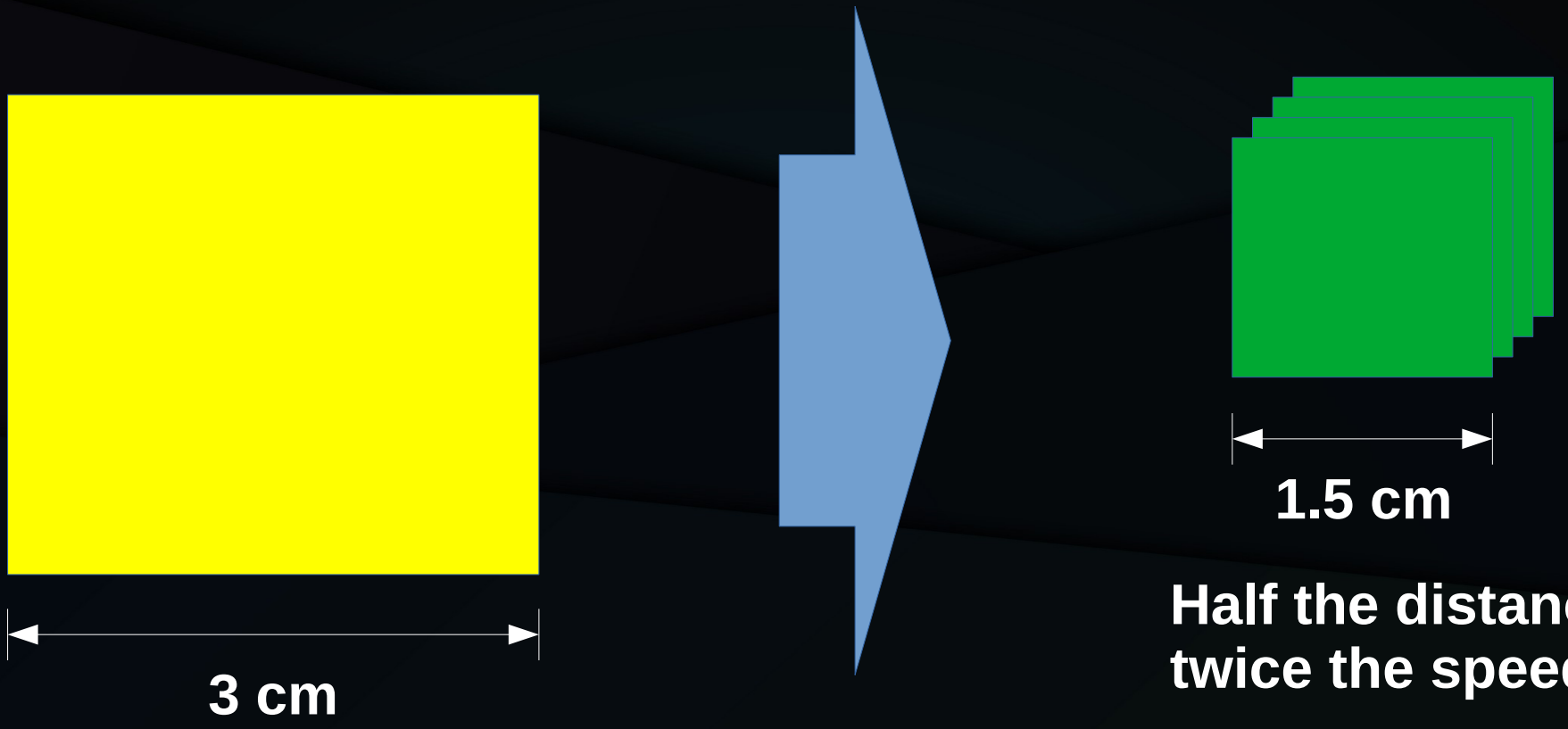
We Really Can Hand-Place Atoms...



But not trillions of them in a cost-effective manner!!!

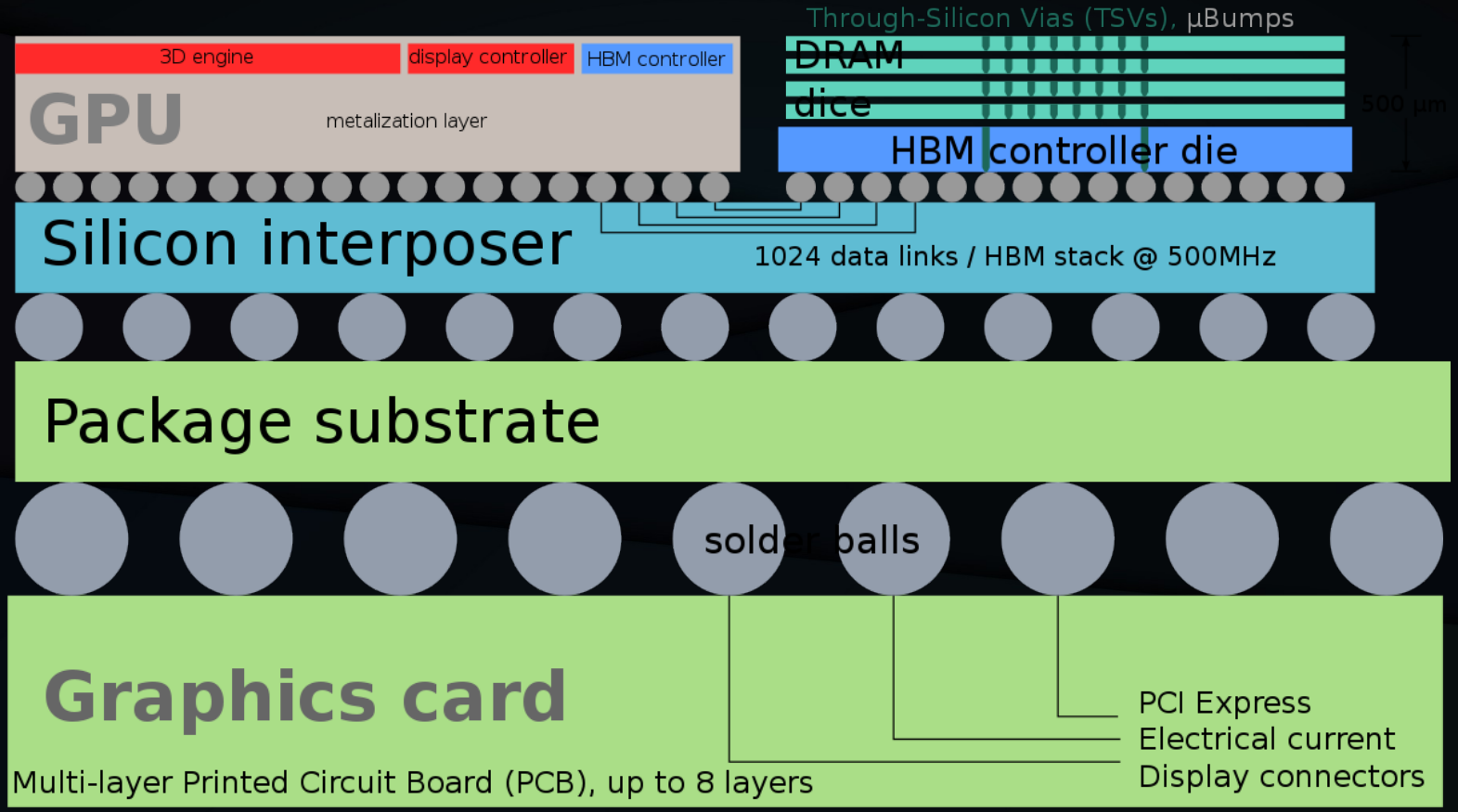
Incremental Help From Hardware

Hardware 3D Integration

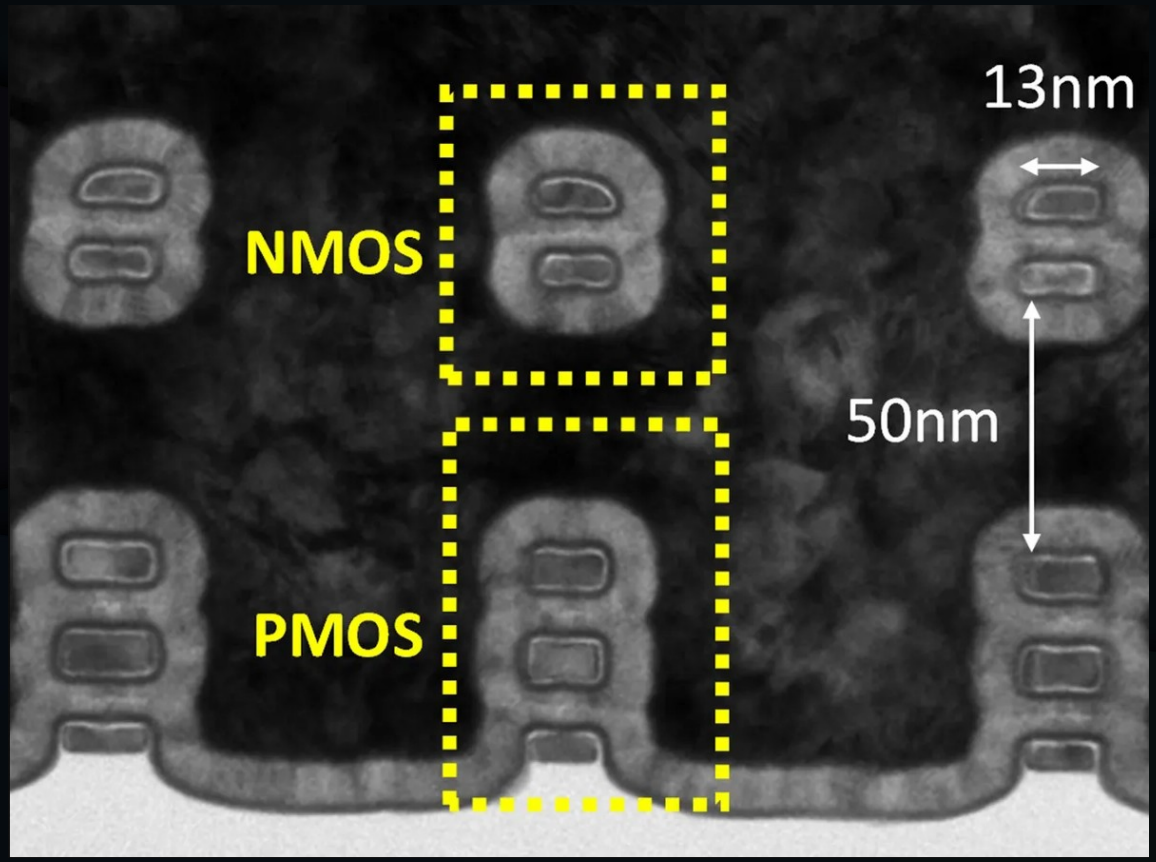


Both stacked chiplets and lithographically stacked transistors.

Stacked Chiplets/Dies

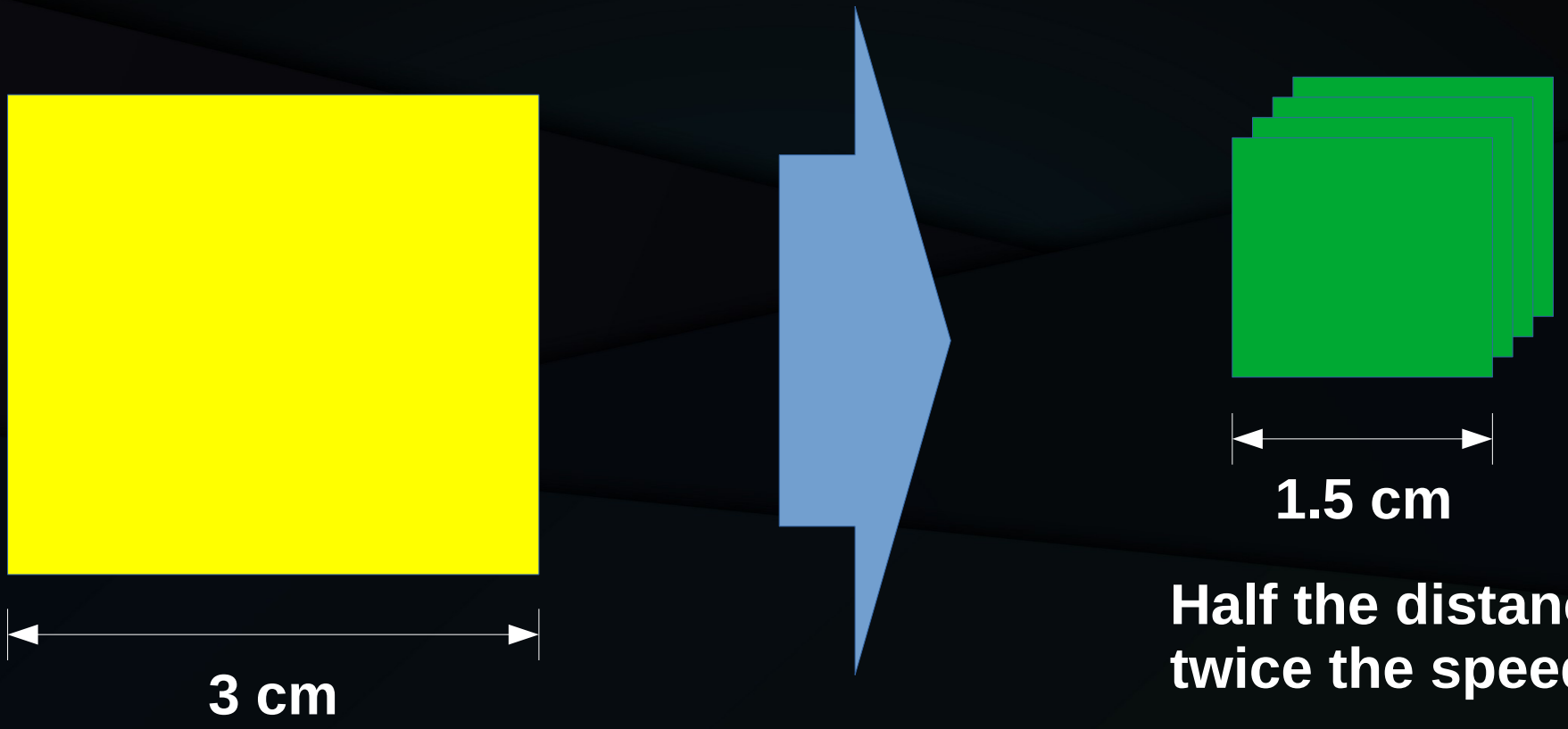


Lithographically Stacked Transistors



https://ieeexplore.ieee.org/document/9976473
https://spectrum.ieee.org/intels-stacked-nanosheet-transistors-could-be-the-next-step-in-moores-law

Hardware 3D Integration



* Give or take issues with power, cooling, alignment, interconnect drivers, and so on.

Hardware Integration is Helping

Operation	Cost (ns)	Ratio (cost/clock)	CPUs
Clock period	0.5	1.0	
Same-CPU CAS	7.0	14.6	0
Same-CPU lock	15.4	32.3	0
In-core blind CAS	7.2	15.2	224
In-core CAS	18.0	37.7	224
Off-core blind CAS	47.5	99.8	1–27,225–251
Off-core CAS	101.9	214.0	1–27,225–251
Off-socket blind CAS	148.8	312.5	28–111,252–335
Off-socket CAS	442.9	930.1	28–111,252–335
Cross-interconnect blind CAS	336.6	706.8	112–223,336–447
Cross-interconnect CAS	944.8	1,984.2	112–223,336–447

Q3 2017: 56 CPUs with ~100ns latencies

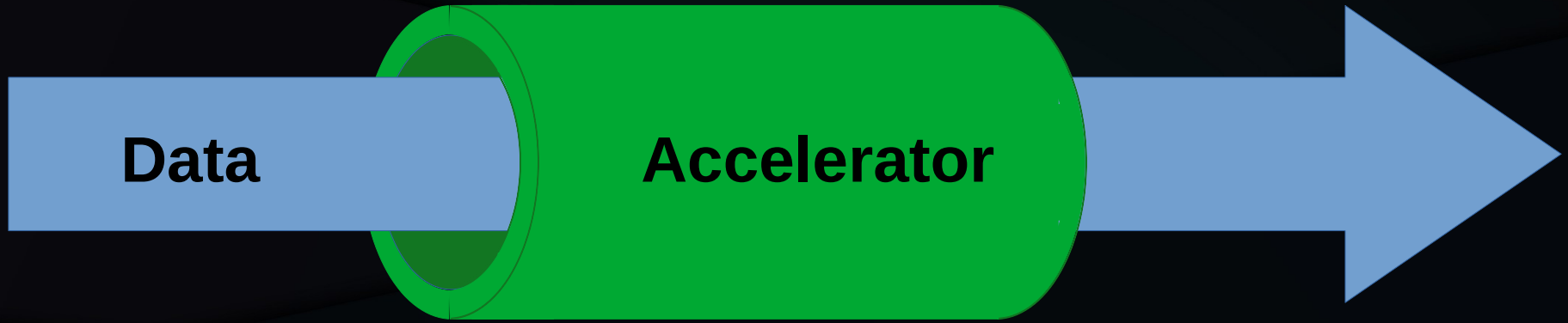
Hardware Integration is Helping

Operation	Cost (ns)	Ratio (cost/clock)
Clock period	0.4	1.0
Same-CPU CAS	12.2	33.8
Same-CPU lock	25.6	71.2
Blind CAS	12.9	35.8
CAS	7.0	19.4
Off-Core		
Blind CAS	31.2	86.6
CAS	31.2	86.5
Off-Socket		
Blind CAS	92.4	256.7
CAS	95.9	266.4

November 2008: 16 CPUs with ~100ns latencies: More than 3x in nine years!!!

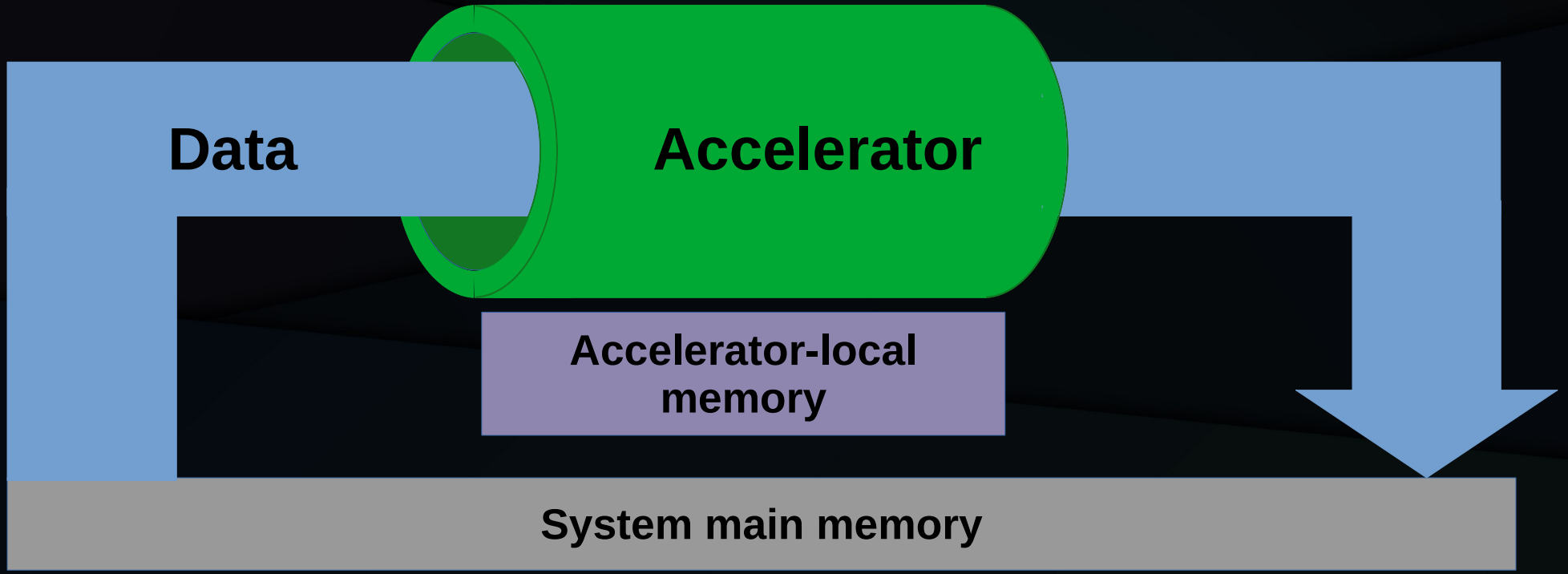
Hardware Accelerators

Hardware Accelerators, Theory



Unidirectional data flow, no out and back, twice the speed!!!

Hardware Accelerators, Practice



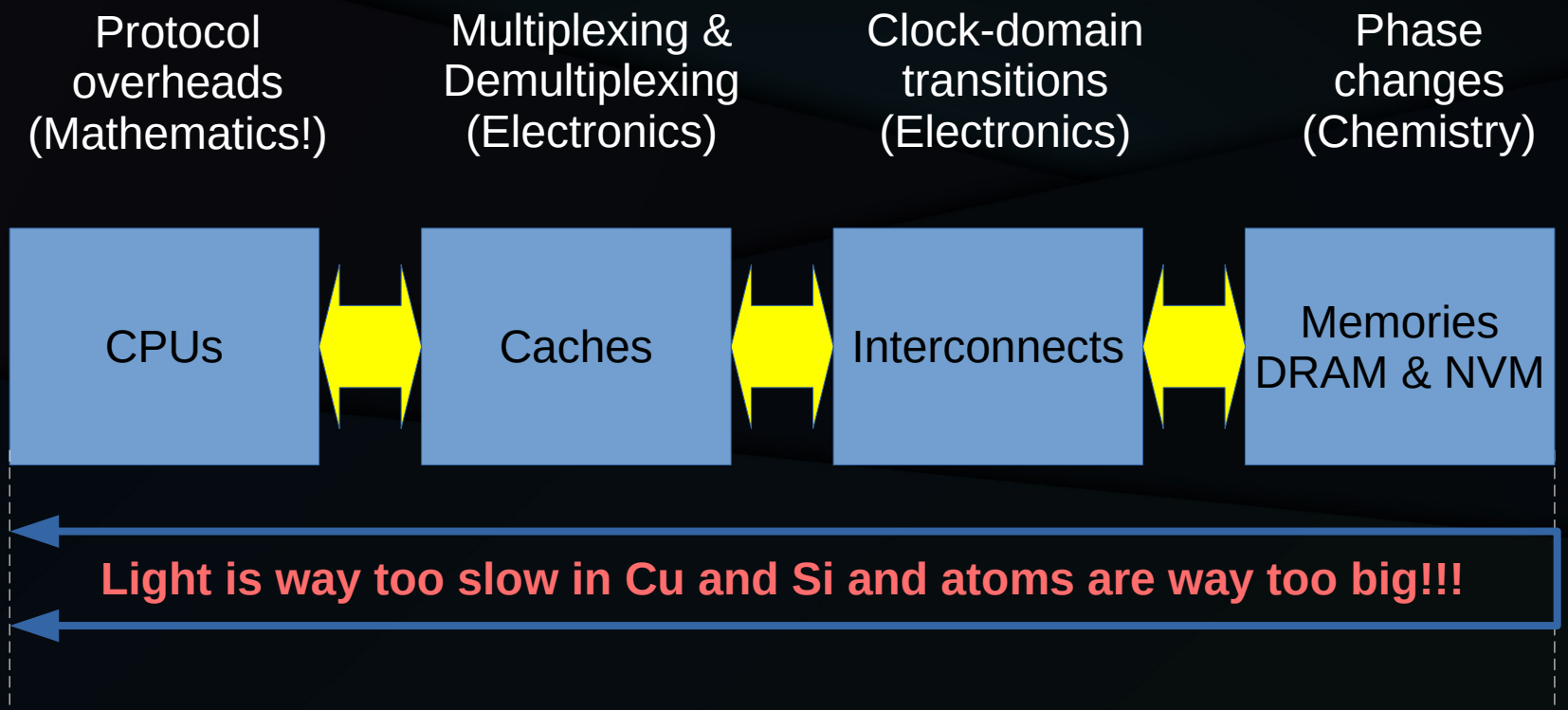
Sadly, back to request-response, but better latency with local memory?

So Why Hardware Accelerators???

- Optimized data transfers (e.g., larger blocks)
- Optimized hard-wired computation
- Better performance per watt
- Better performance per unit capital cost

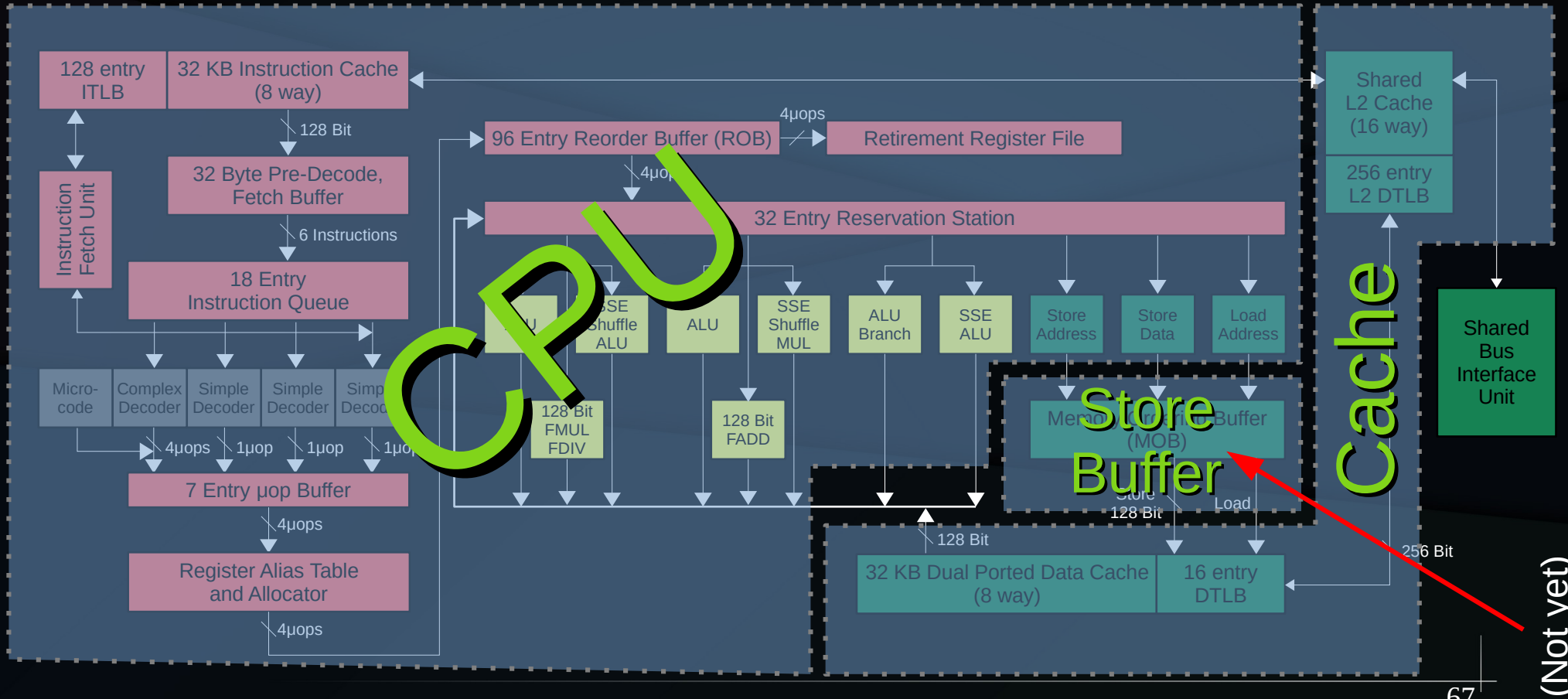
Hardware Has Been Helping All Along

What Hardware Is Up Against



Therefore, Memory Hierarchies!!!

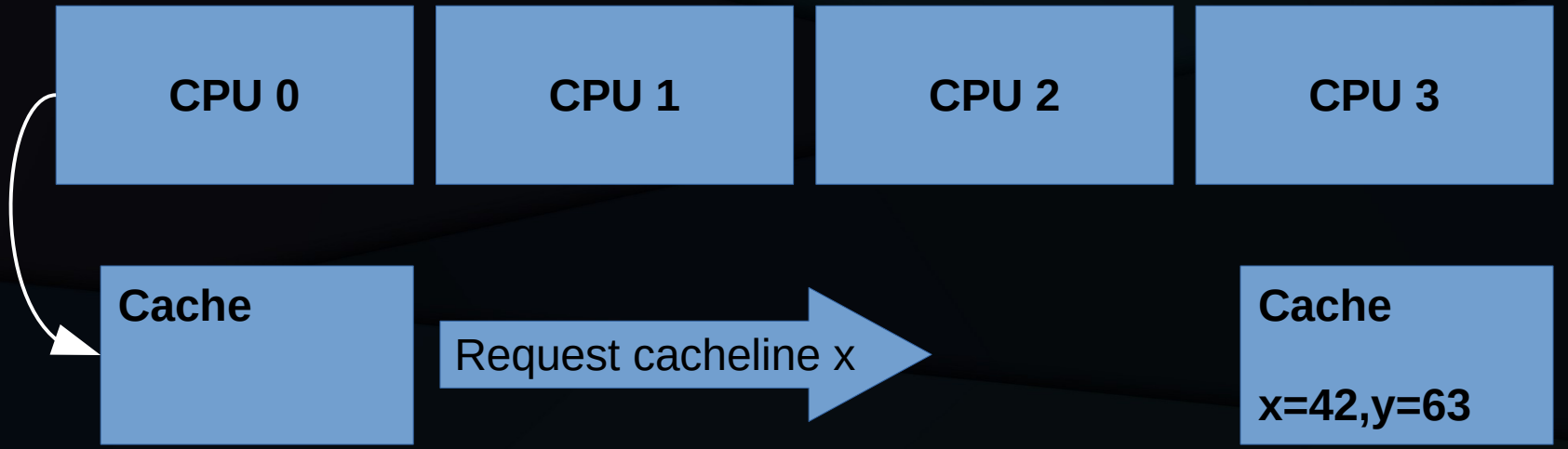
Simple Portable CPU Model Redux



Intel Core 2 Architecture (Wikipedia user “I, Appaloosa” CC BY-SA 3.0, reformatted and remixed)

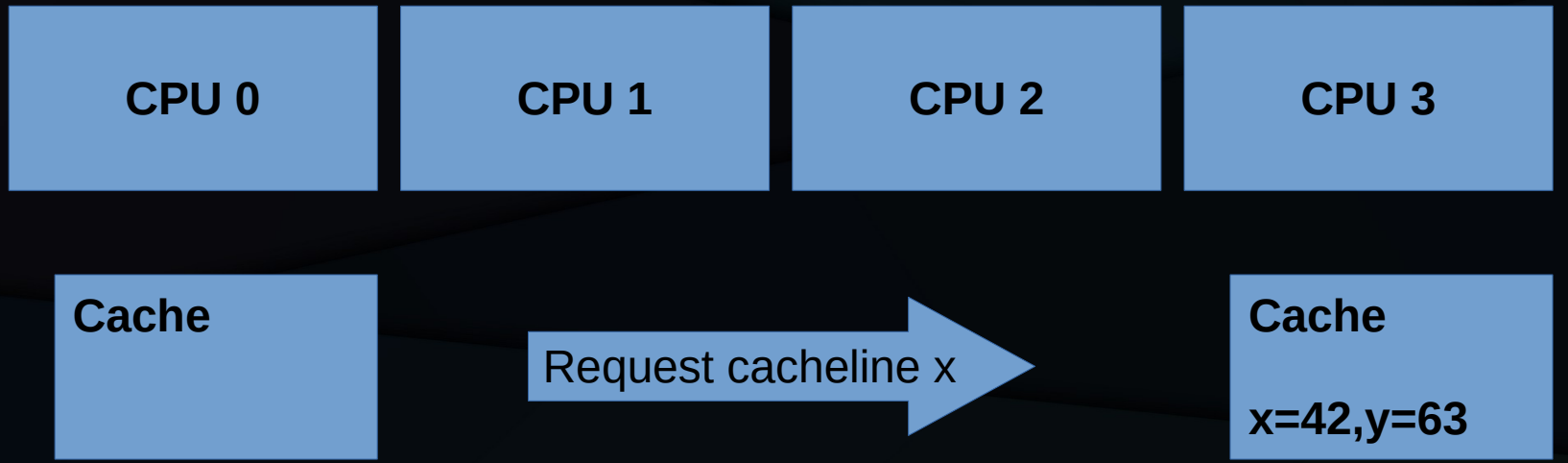
Read-Side Hardware Help (1/7)

```
r1 = READ_ONCE(x)
r2 = READ_ONCE(y)
r3 = READ_ONCE(x)
```



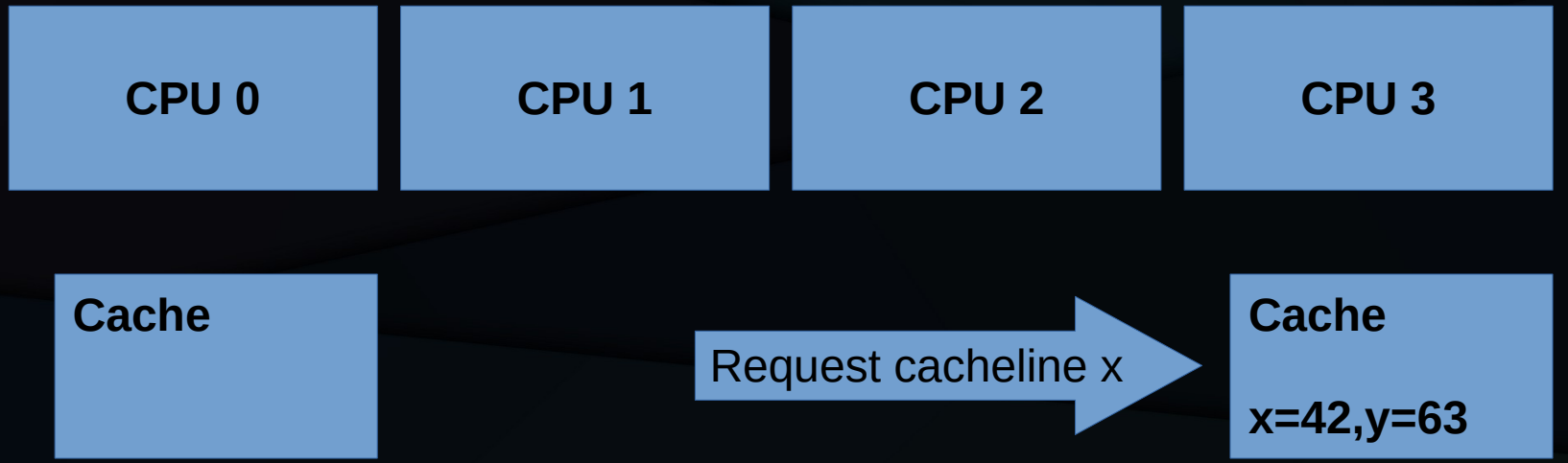
Read-Side Hardware Help (2/8)

```
r1 = READ_ONCE(x)
r2 = READ_ONCE(y)
r3 = READ_ONCE(x)
```



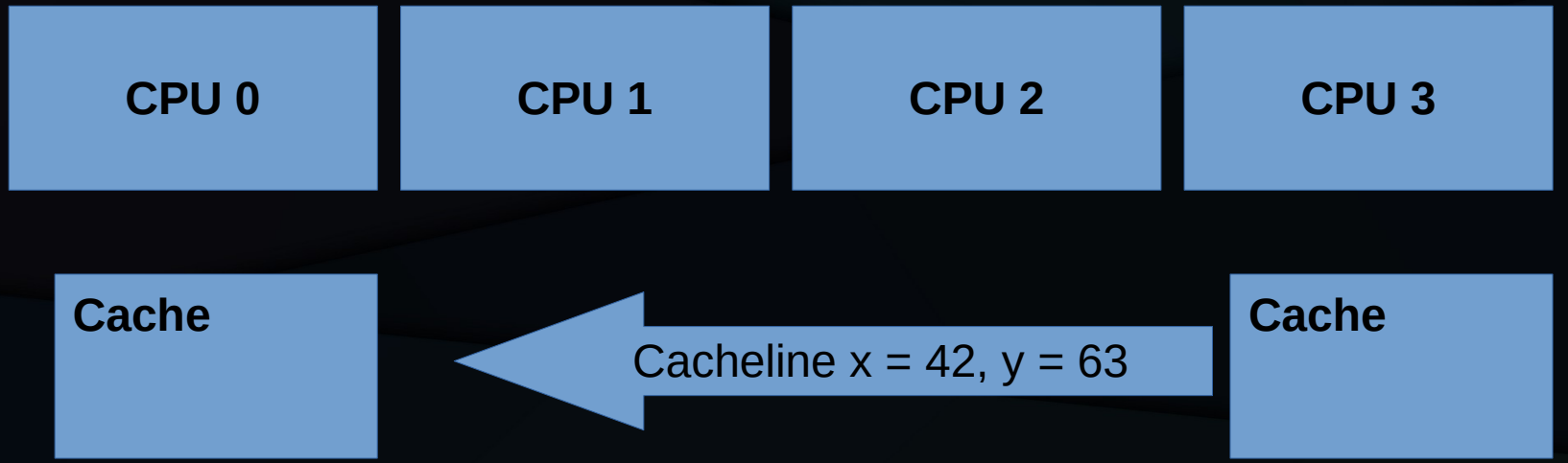
Read-Side Hardware Help (3/8)

```
r1 = READ_ONCE(x)
r2 = READ_ONCE(y)
r3 = READ_ONCE(x)
```



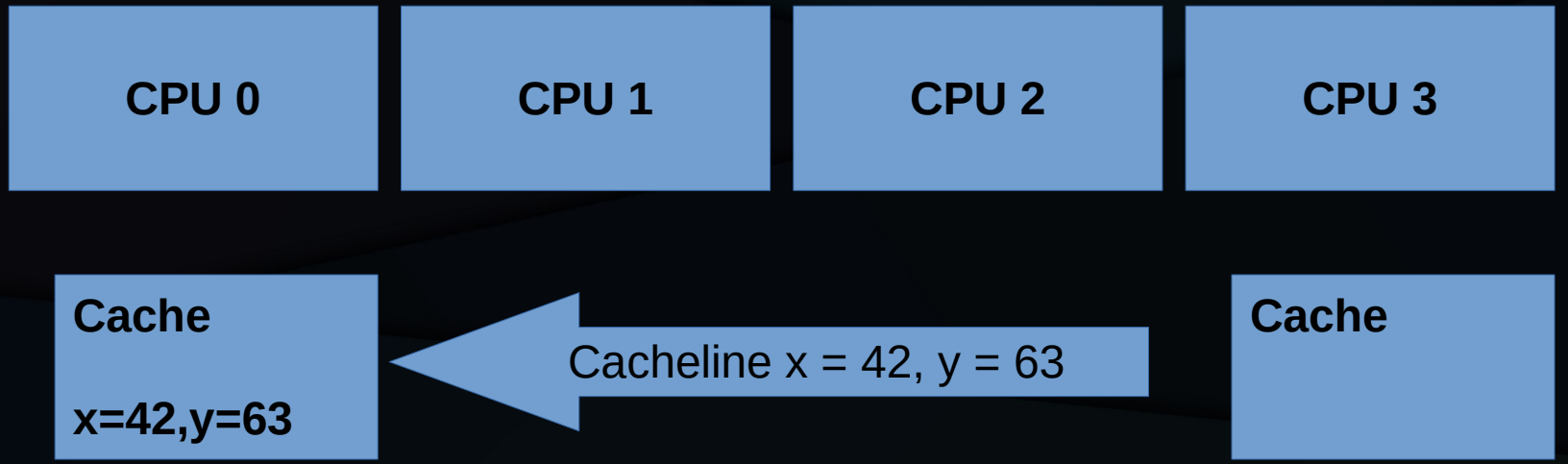
Read-Side Hardware Help (4/8)

```
r1 = READ_ONCE(x)
r2 = READ_ONCE(y)
r3 = READ_ONCE(x)
```



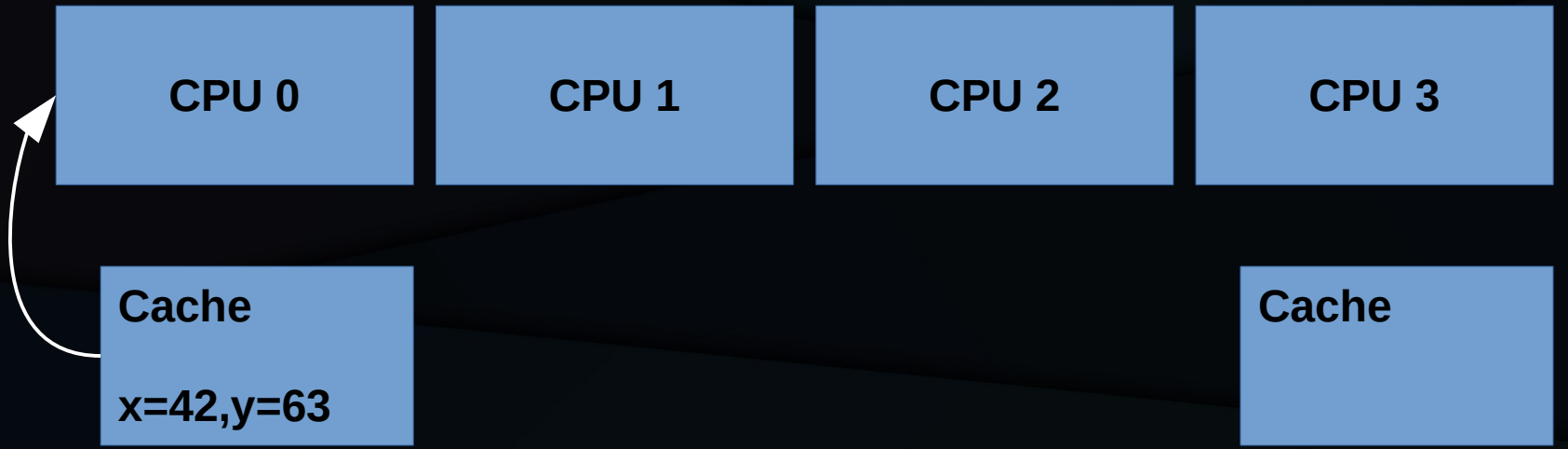
Read-Side Hardware Help (5/8)

```
r1 = READ_ONCE(x)
r2 = READ_ONCE(y)
r3 = READ_ONCE(x)
```



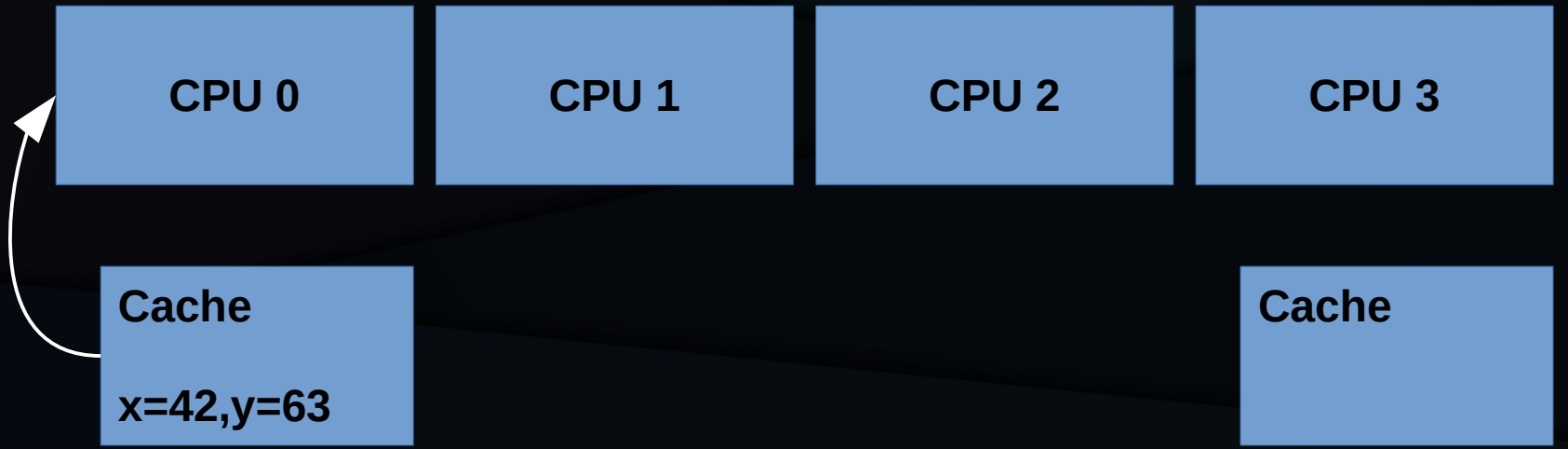
Read-Side Hardware Help (6/8)

```
~~r1 = READ_ONCE(x)~~
r2 = READ_ONCE(y)
r3 = READ_ONCE(x)
```



Read-Side Hardware Help (7/8)

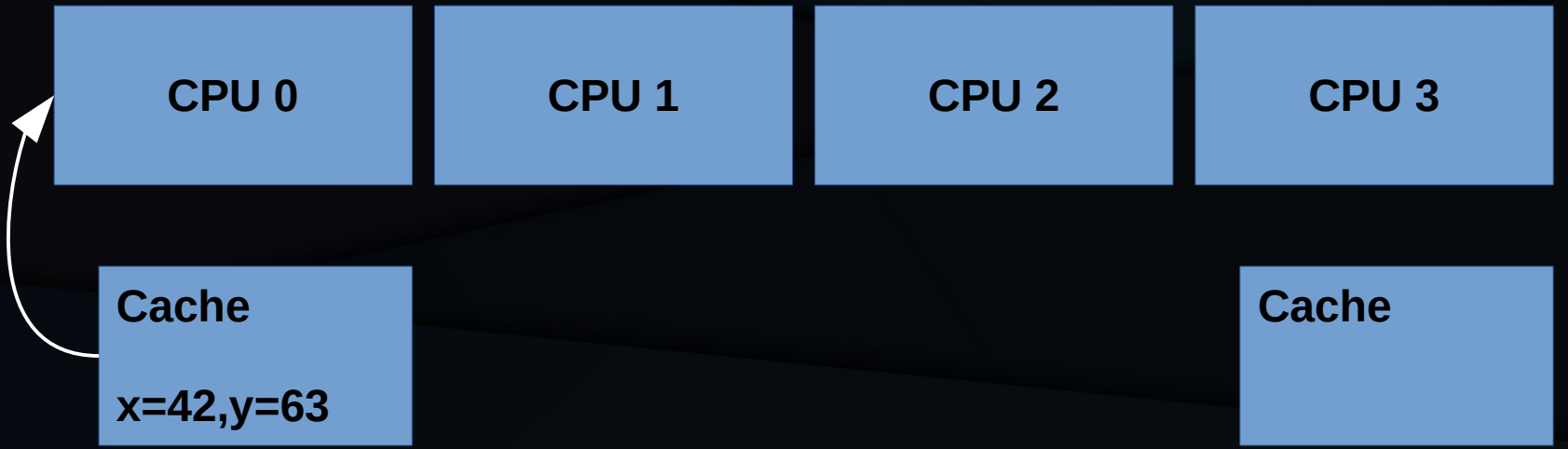
```
r1 = READ_ONCE(x)
r2 = READ_ONCE(y)
r3 = READ_ONCE(x)
```



Caches help beat laws of physics given spatial locality!!!

Read-Side Hardware Help (8/8)

```
r1 = READ_ONCE(x)
r2 = READ_ONCE(y)
r3 = READ_ONCE(x)
```



Caches help beat laws of physics given temporal locality!!!

Levels of Cache on My Laptop

Index	Line Size	Associativity	Size
0	64	8	32K
1	64	8	32K
2	64	4	256K
3	64	16	16,384K

When taking on the laws of physics, don’t be afraid to use a few transistors

Levels of Cache on Large Old Server

Index	Line Size	Associativity	Size
0	64	8	32K
1	64	6	32K
2	64	16	1,024K
3	64	11	39,424K

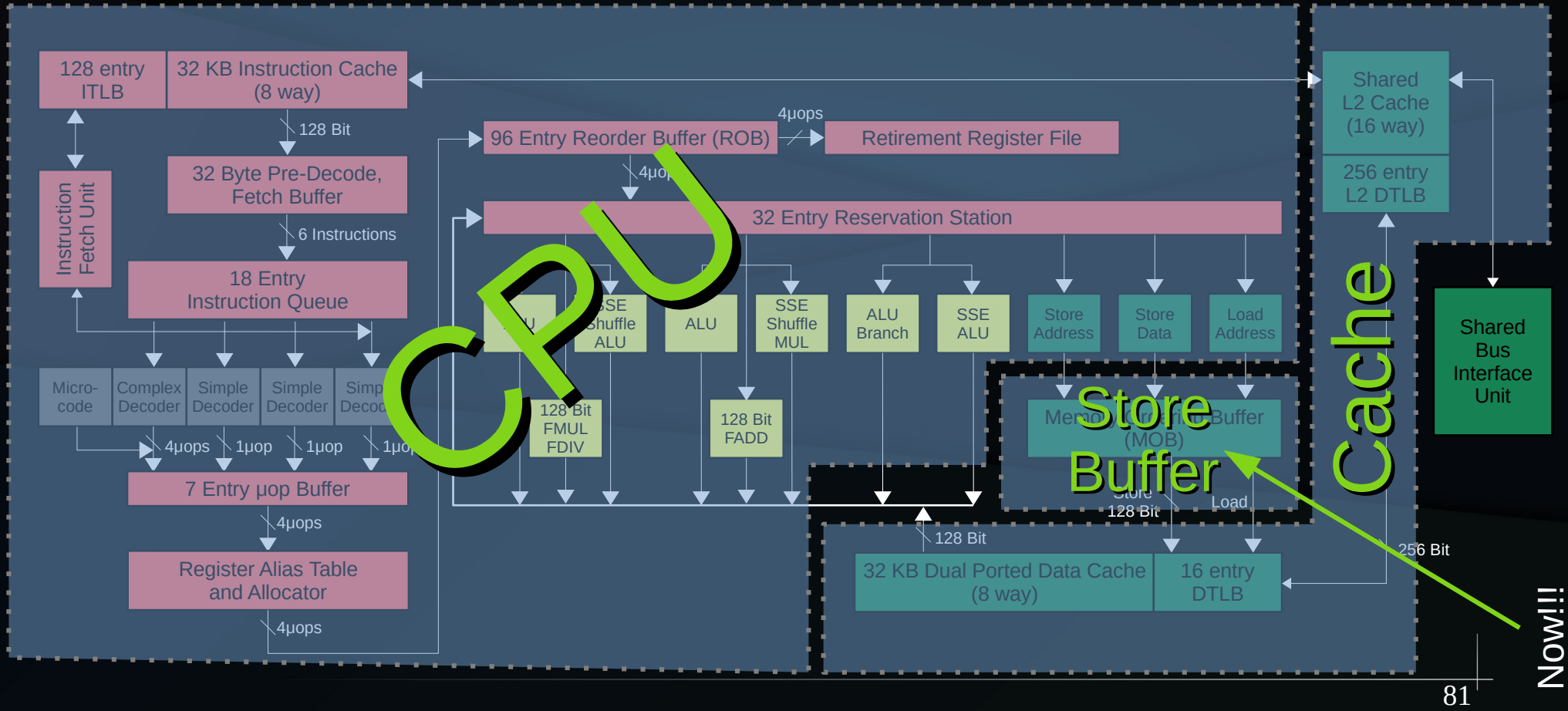
When taking on the laws of physics, don’t be afraid to use lots of transistors

But What About Writes?

Write-Side Hardware Help (Summary)

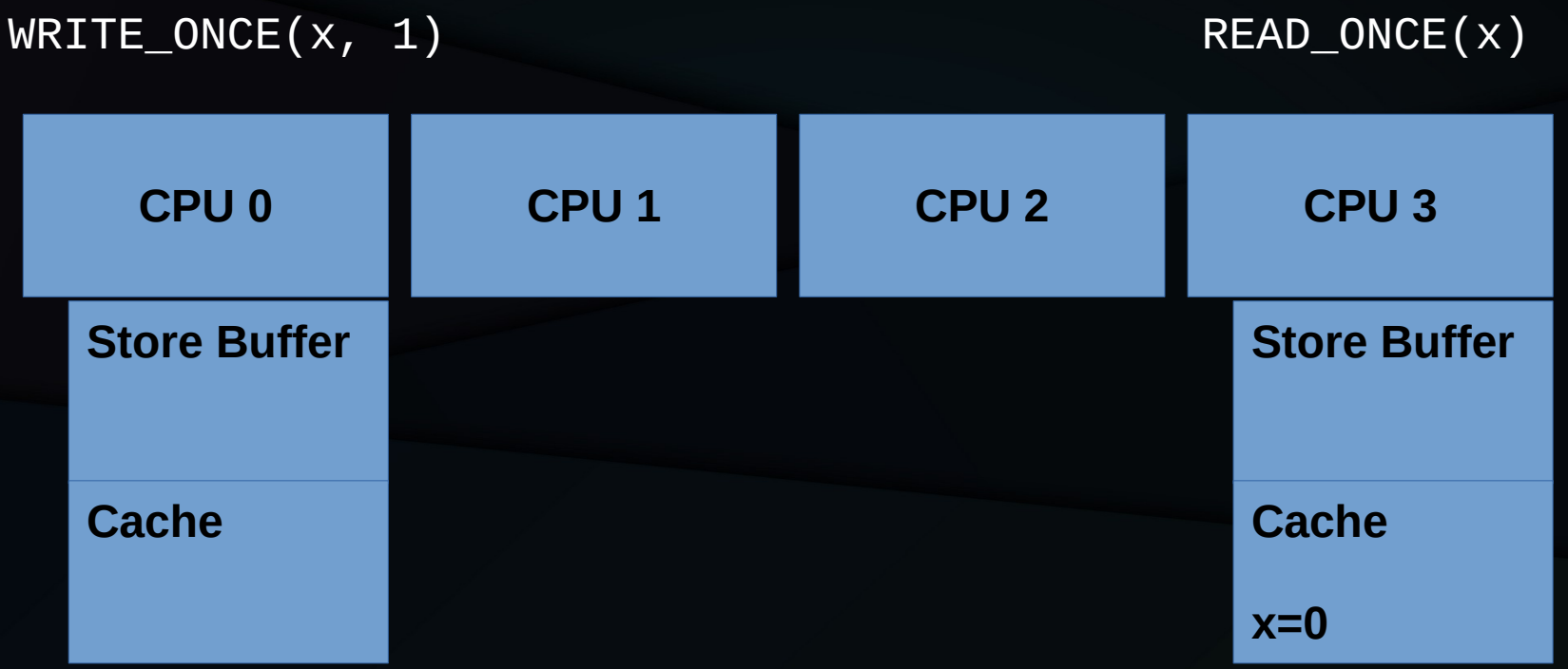
- Store buffers for the win!!! Sort of…
  - Cache line for variable x is initially at CPU 3
  - CPU 0 writes 1 to x, but doesn't have cacheline
    - So holds the write in CPU 0's store buffer
    - And requests exclusive access to the cacheline (which takes time)
  - CPU 3 reads x, obtaining “0” immediately from cacheline
  - CPU 0 receive's x's cacheline
    - And CPU 0's write finally gets to the cacheline
    - Overwriting the value that CPU 3 read, despite the write starting earlier
- Writes only ***appear*** to be instantaneous!!!

Simple Portable CPU Model Redux

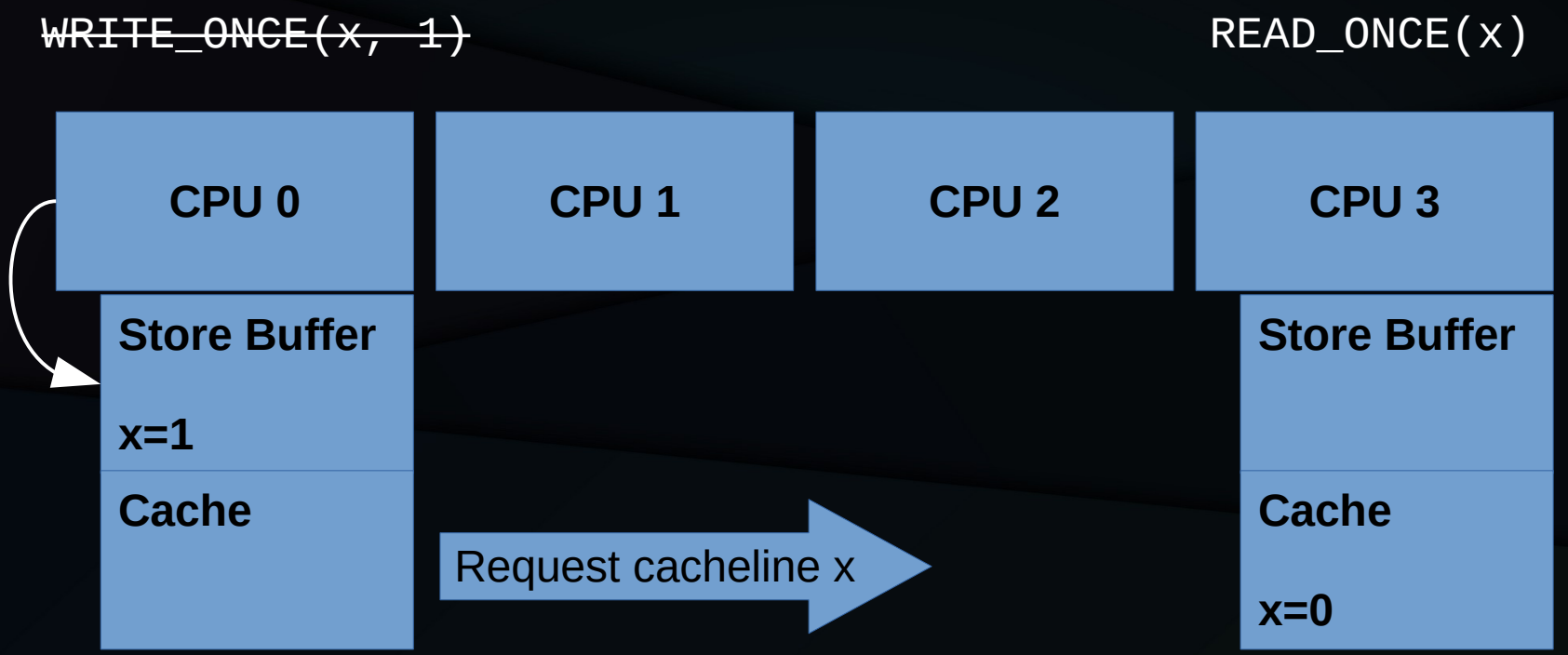


Intel Core 2 Architecture (Wikipedia user “I, Appaloosa” CC BY-SA 3.0, reformatted and remixed)

Write-Side Hardware Help (1/7)

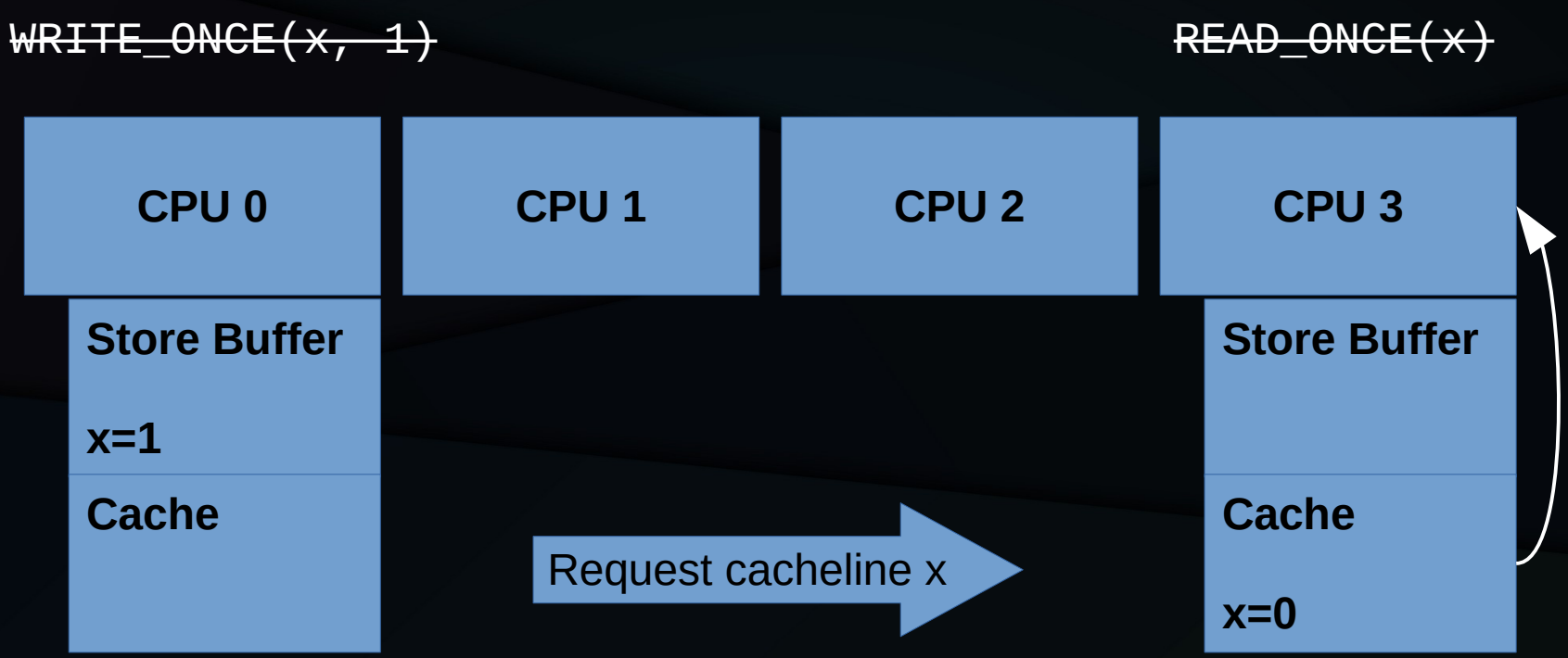


Write-Side Hardware Help (2/7)



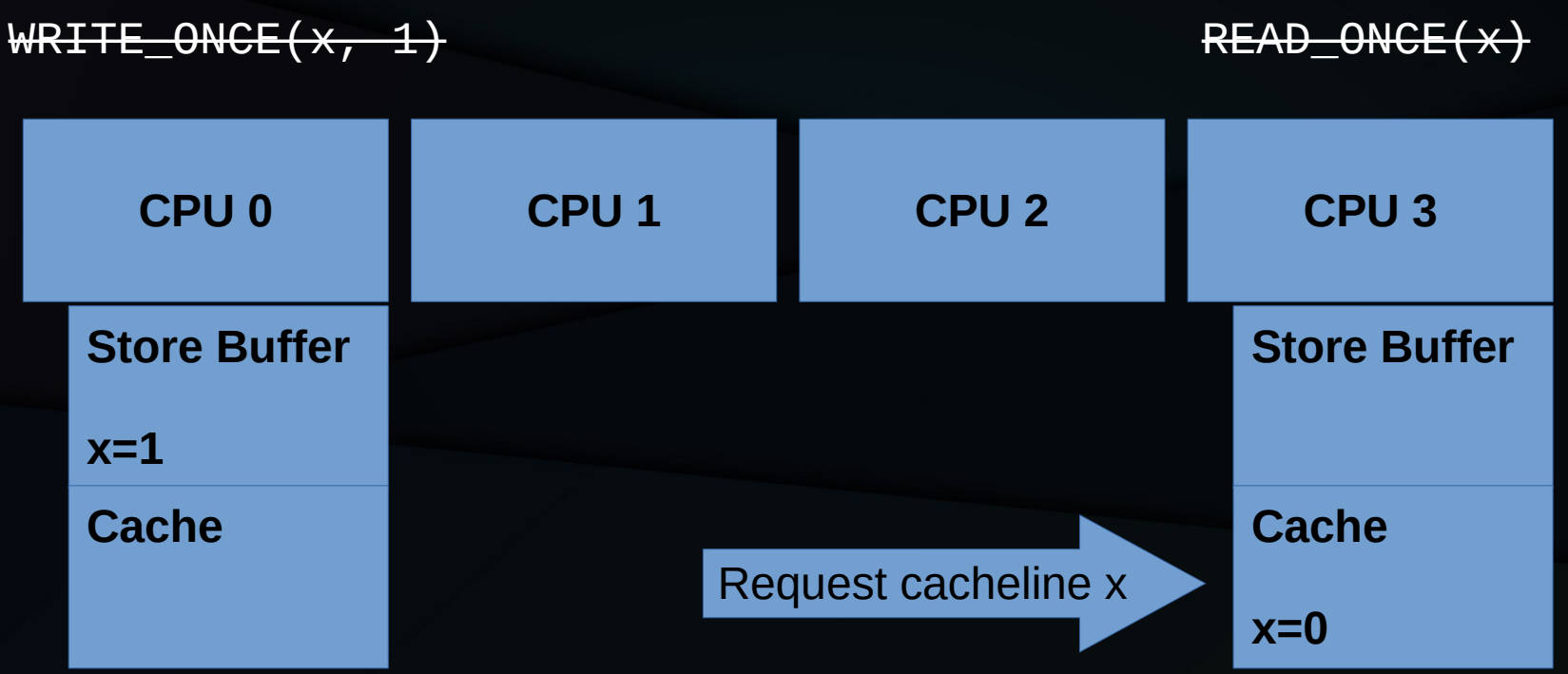
The store buffer allows writes to completes quickly!!! Take that, laws of physics!!!

Write-Side Hardware Help (3/7)



Except that later read gets older value...

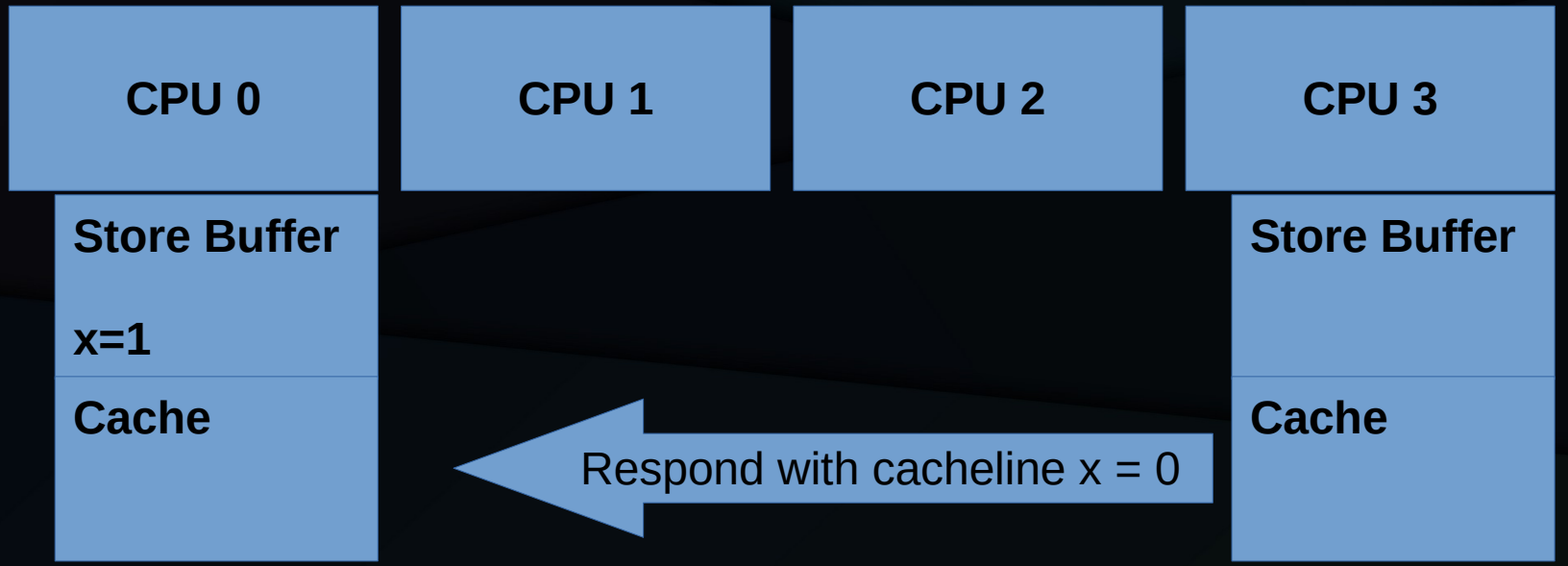
Write-Side Hardware Help (4/7)



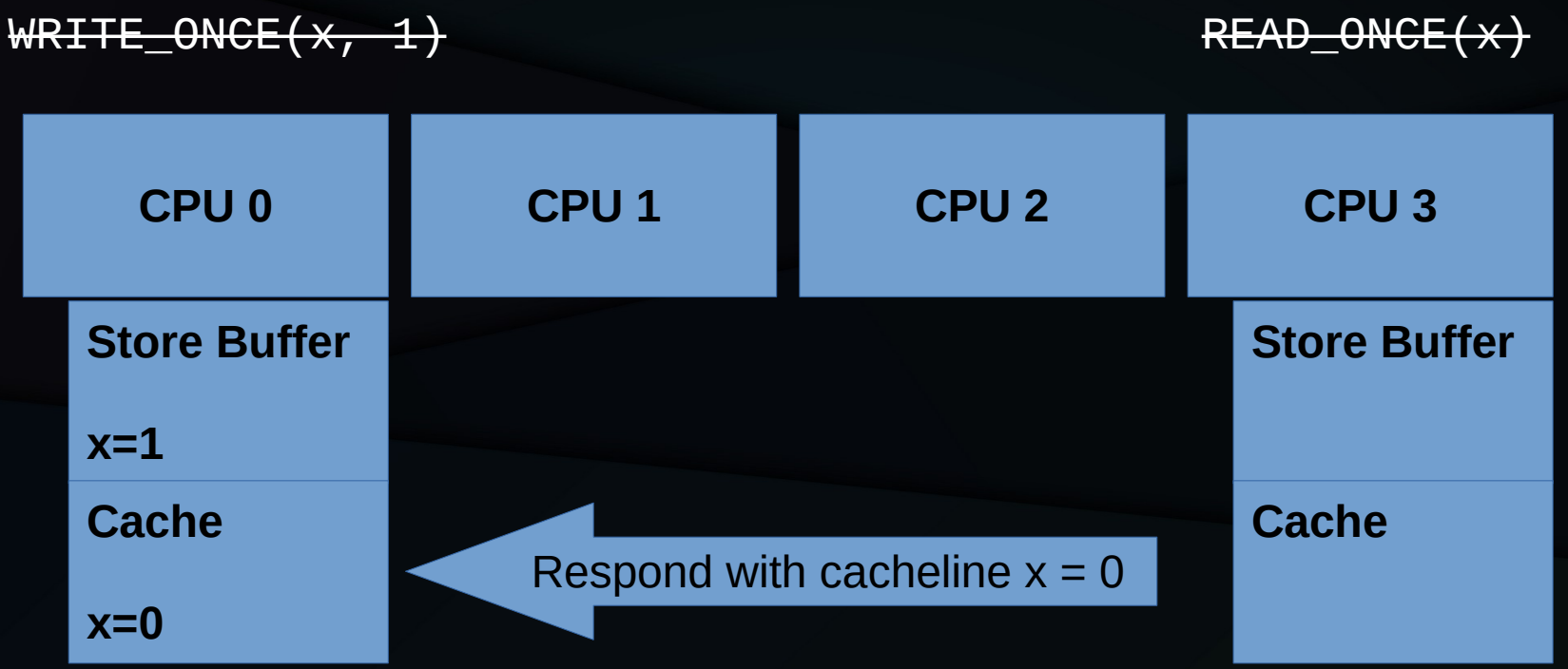
Write-Side Hardware Help (5/7)

~~WRITE_ONCE(x, 1)~~

~~READ_ONCE(x)~~



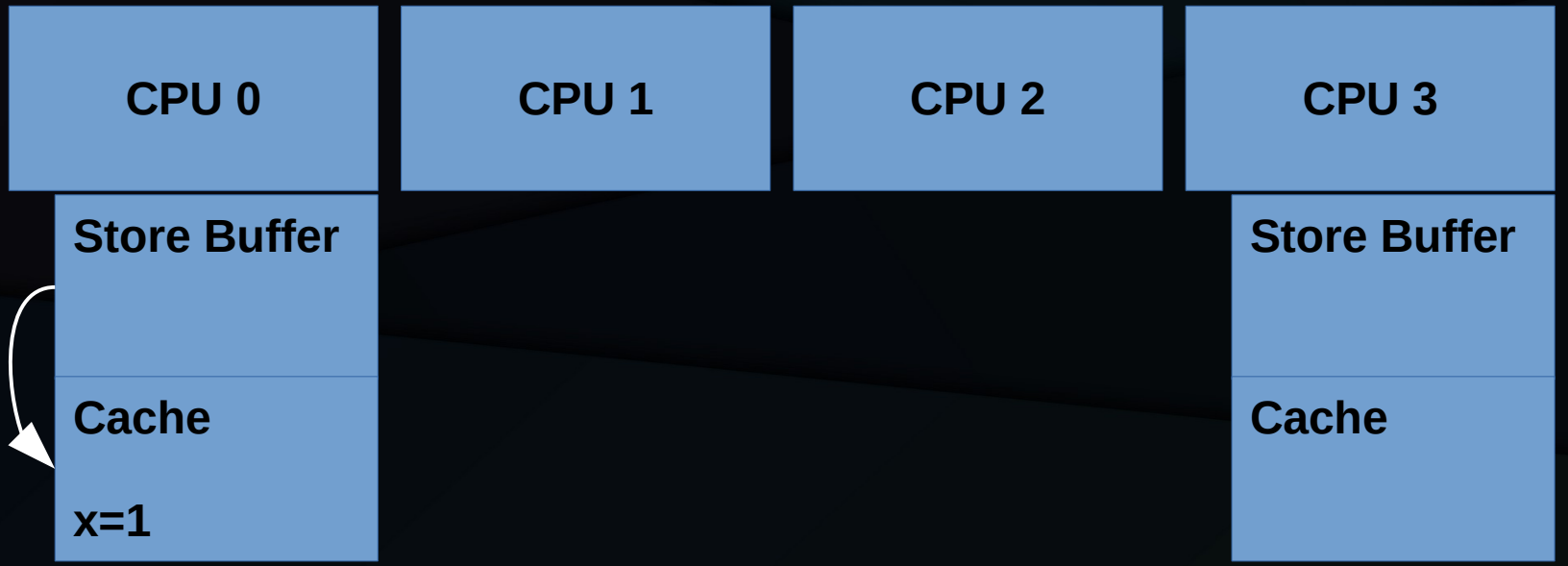
Write-Side Hardware Help (6/7)



Write-Side Hardware Help (7/7)

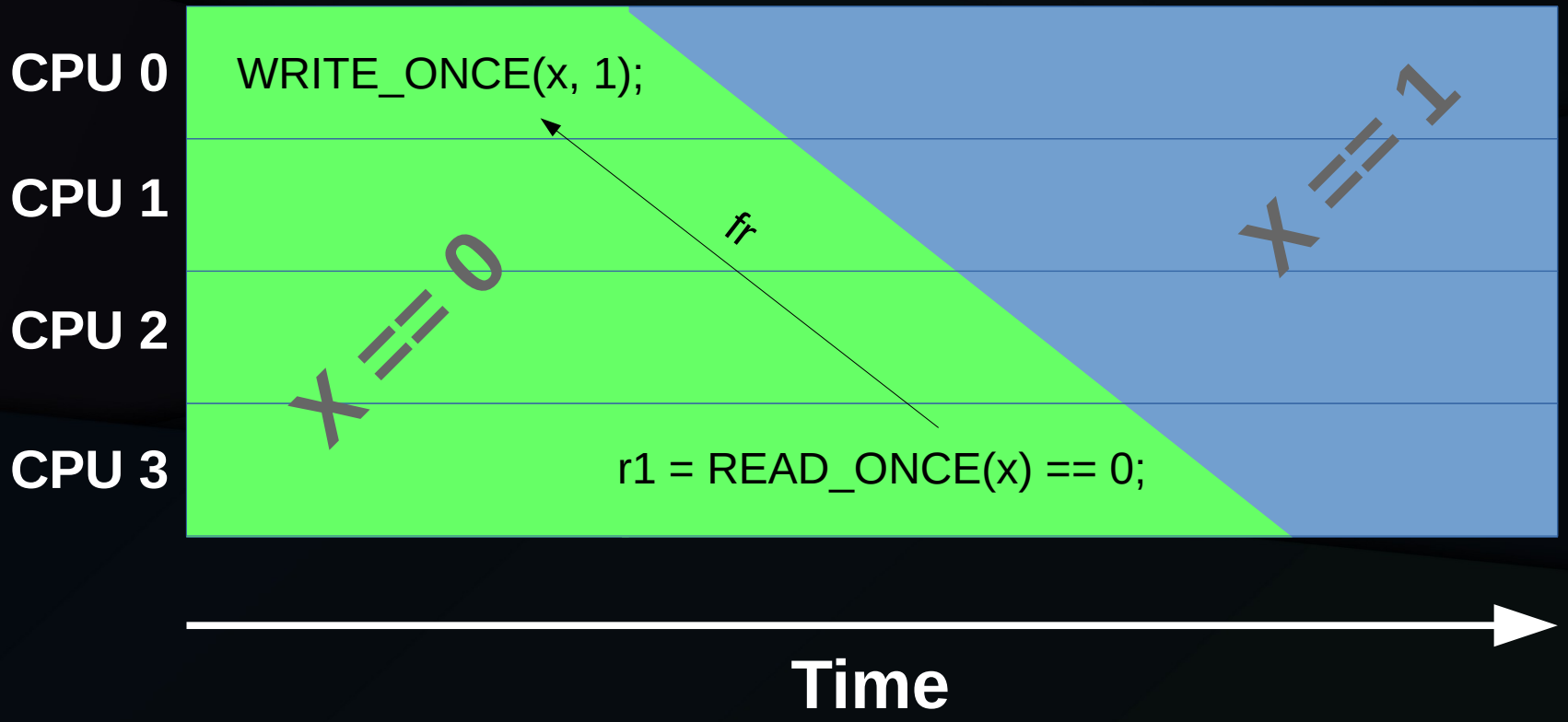
~~WRITE_ONCE(x, 1)~~

~~READ_ONCE(x)~~

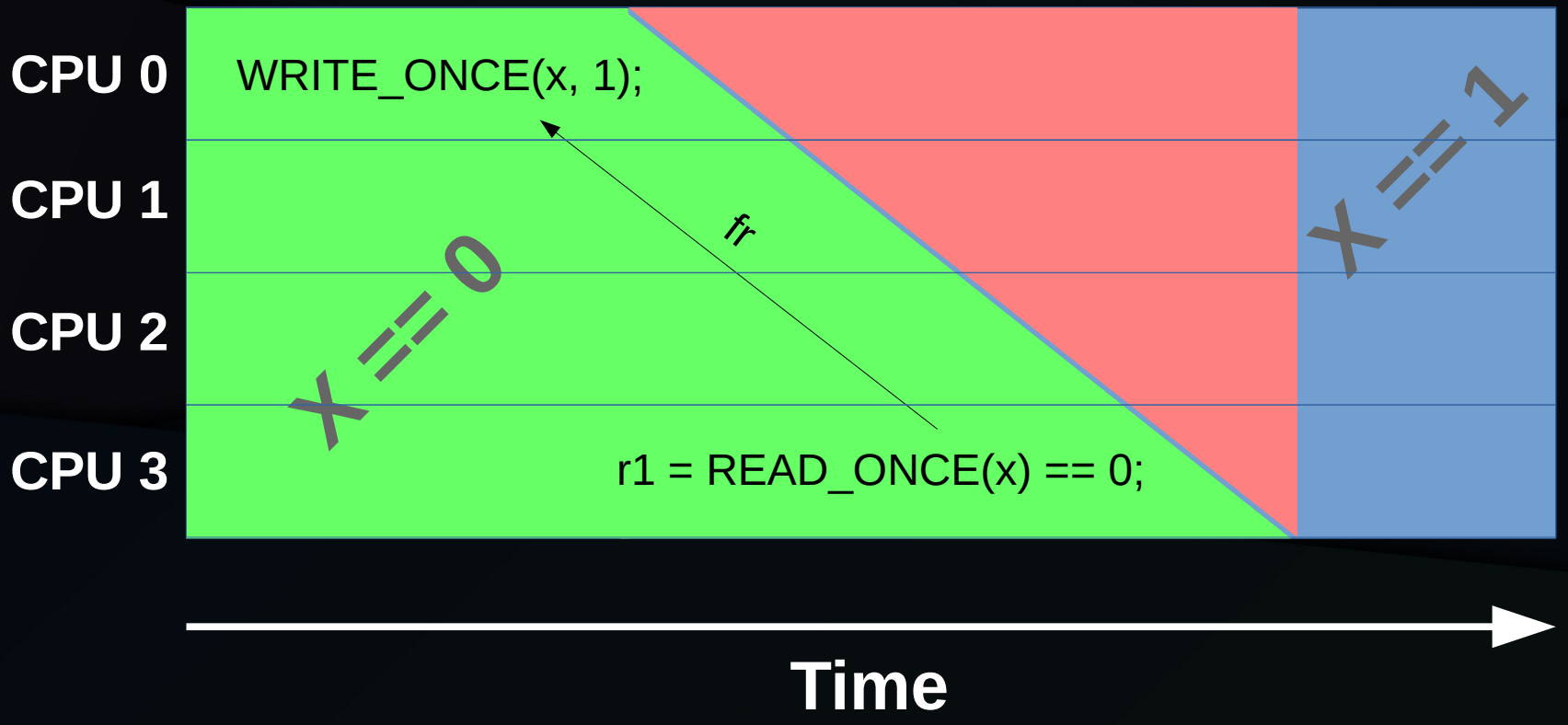


Quick write completion, sort of. Laws of physics: Slow or misordered!!!

Misordering? Or Propagation Delay?



And Careful What You Wish For!!!



Hardware tricks help reduce the red triangle. But too bad about Meltdown and Spectre...

Can Software Help?

Can Software Help?

- Increasingly, yes!!!
- Use concurrent libraries, applications, subsystems, and so on
  - Let a few do the careful coding and tuning
  - Let a great many benefit from the work of a few
- Use proper APIs to deal with memory ordering
  - Chapter 14: “Advanced Synchronization: Memory Ordering”

https://mirrors.edge.kernel.org/pub/linux/kernel/people/paulmck/perfbook/perfbook.html

Summary

Summary

- Modern hardware is highly optimized
  - Most of the time!
  - Incremental improvements due to integration
  - But the speed of light is too slow and atoms too big
- Use concurrent software where available
- Structure your code to avoid the big obstacles
  - Micro-optimizations only sometimes needed

For More Information

- “Computer Architecture: A Quantitative Approach”, Hennessey & Patterson (Recent HW)
- “Parallel Computer Architecture: A Hardware/Software Approach”, Culler et al.
  - Includes SGI Origin and Sequent NUMA-Q
- “Programing Massively Parallel Processors: A Hands-on Approach”, Kirk & Hwu
  - Primarily NVIDIA GPGPUs
- https://developer.nvidia.com/educators/existing-courses
  - List of NVIDIA university courseware
- https://gpuopen.com/professional-compute
  - List of AMD GPGPU-related content
- “Is Parallel Programming Hard, And, If So, What Can You Do About It?”
  - https://mirrors.edge.kernel.org/pub/linux/kernel/people/paulmck/perfbook/perfbook.html

L'antidote de Codage Simultané de la Femme de Paul

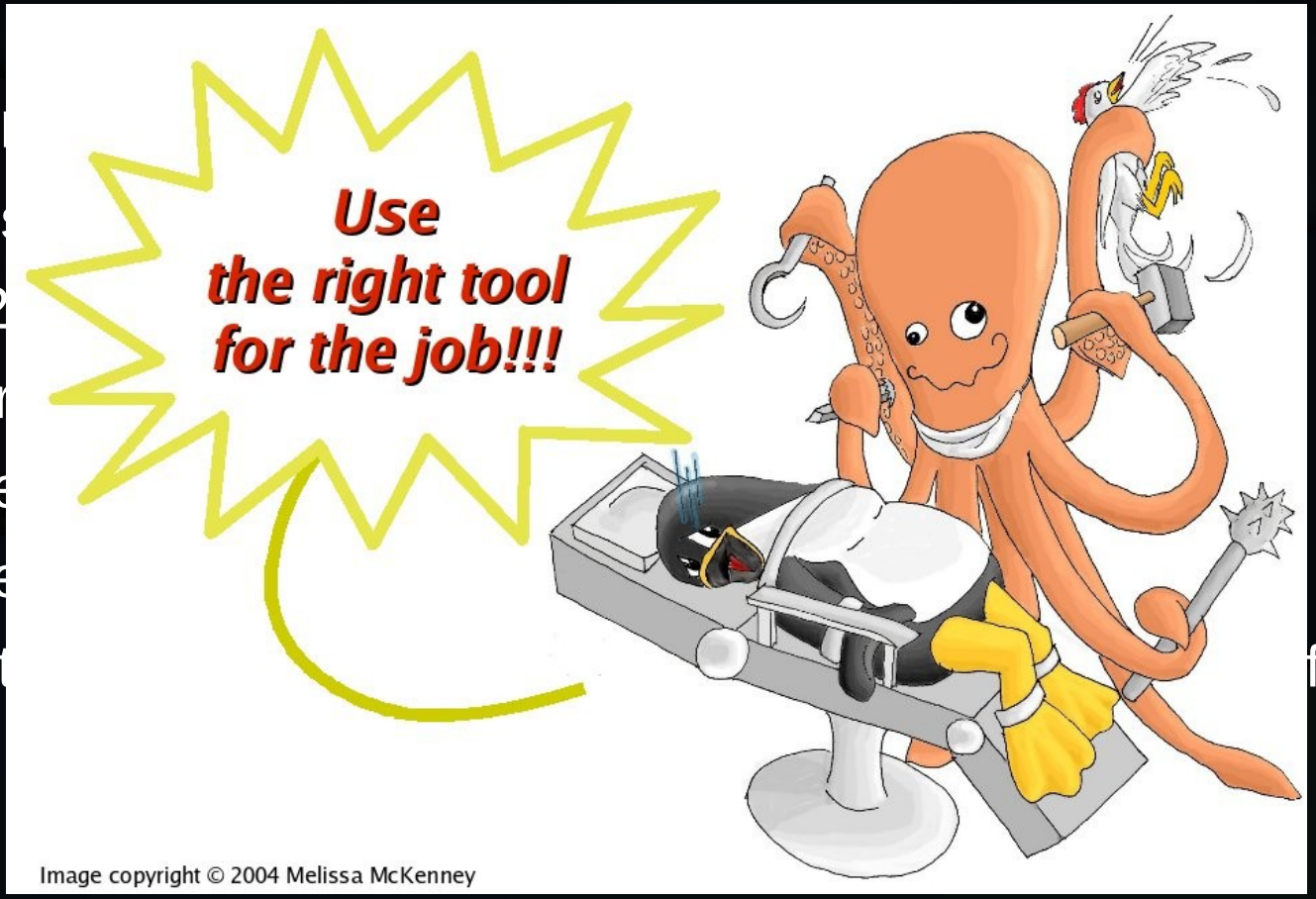
- Cordial de mûres sauvages de l'Himalaya
  - Mettre deux litres de mûres dans un pot de quatre litres
  - Ajouter cinq huitièmes litres de sucre
  - Remplissez le pot de vodka
  - Secouez tous les jours pendant cinq jours
  - Secouez chaque semaine pendant cinq semaines
  - Passer au tamis: Ajoutez des baies à la glace, consommez le liquide filtré comme vous voulez

Paul’s Wife’s Concurrency Antidote

- Wild Himalayan Blackberry Cordial
  - Put 8 cups wild himalayan blackberries in 1 gallon jar
  - Add 2½ cups sugar
  - Fill jar with vodka
  - Shake every day for five days
  - Shake every week for five weeks
  - Pour through sieve: Add berries to ice cream, consume filtered liquid as you wish

Paul's Wife's Questions? Antidote

- Wild Hi
  - 8 cup
  - Add 2
  - Fill jar
  - Shake
  - Shake
  - Pour t filtered liquid



If there is no right tool, invent it!!!