The Beauty and Joy of Computing

Lecture #07
Concurrency

UC Berkeley
Teaching Assistant
Aaron Baker

"KOOMEY'S LAW" – EFFICIENCY 2X EVERY 18

Prof Jonathan Koomey looked at 6 decades of data and found that energy efficiency of computers doubles roughly every 18 months. This is even more relevant as battery-powered devices become more popular. Restated, it says that for a fixed computing load, the amount of battery you need drops by half every 18 months. This was true before transistors!

www.technologyreview.com/computing/38548/

Concurrency: Definition & Examples

Definition: A property of computer systems in which several computations are executing simultaneously, and potentially interacting with each other.

Examples:

- Mouse cursor movement while Snap*!* calculates.
- Screen clock advances while typing in a text.
- Busy cursor spins while browser connects to server, waiting for response

Concurrency & Parallelism

Intra-computer

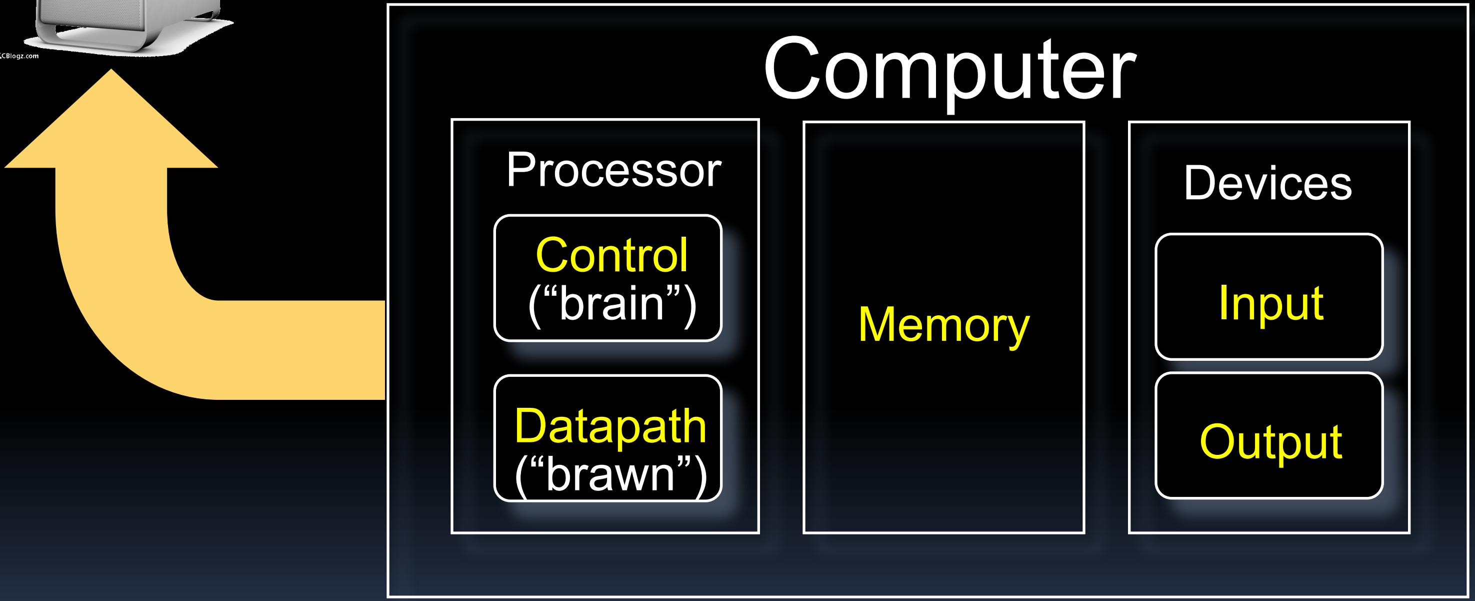
- Today's lecture
- Multiple computing "helpers" are cores <u>within one machine</u>
- Referred to as "multi-core"

Inter-computer

- Multiple computing "helpers" are <u>different machines</u>
- Referred to as "distributed computing"

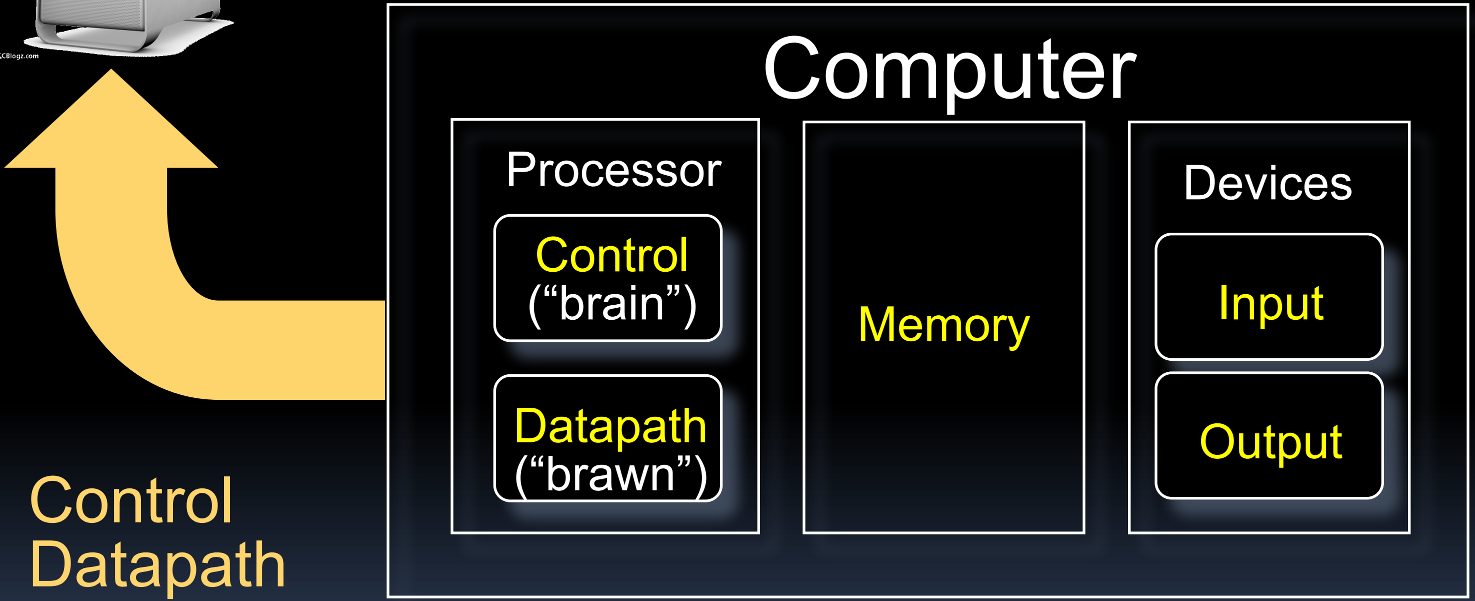
Anatomy: 5 components of any Computer

John von Neumann invented this architecture



Anatomy: 5 components of any Computer

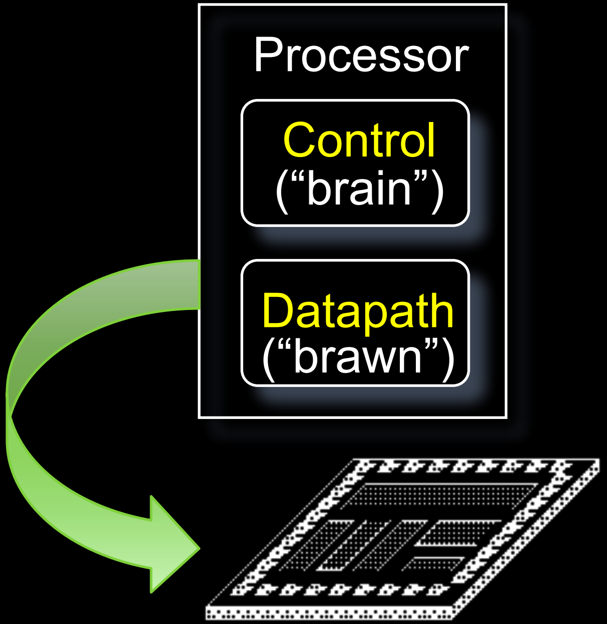
www.yellkey.com/fly



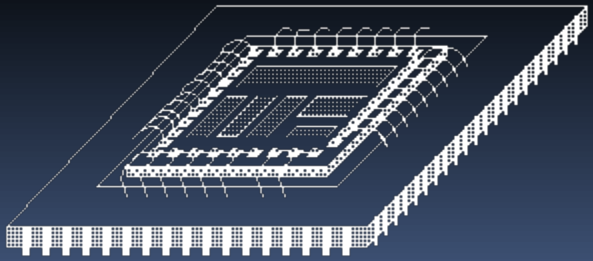
a) Control
b) Datapath
c) Memory
d) Input
e) Output

What causes the most headaches for SW and HW designers with multi-core computing?

But what is INSIDE a Processor?



Bare Processor Die



Chip in Package

- Primarily Crystalline Silicon
- 1 mm – 25 mm on a side
- 2009 "feature size" (aka process) ~ 45 nm = 45 x 10^{-9} m (then 32, 22, and 16 [by yr 2013])
- 100 - 1000M transistors
- 3 - 10 conductive layers
- "CMOS" (complementary metal oxide semiconductor) - most common

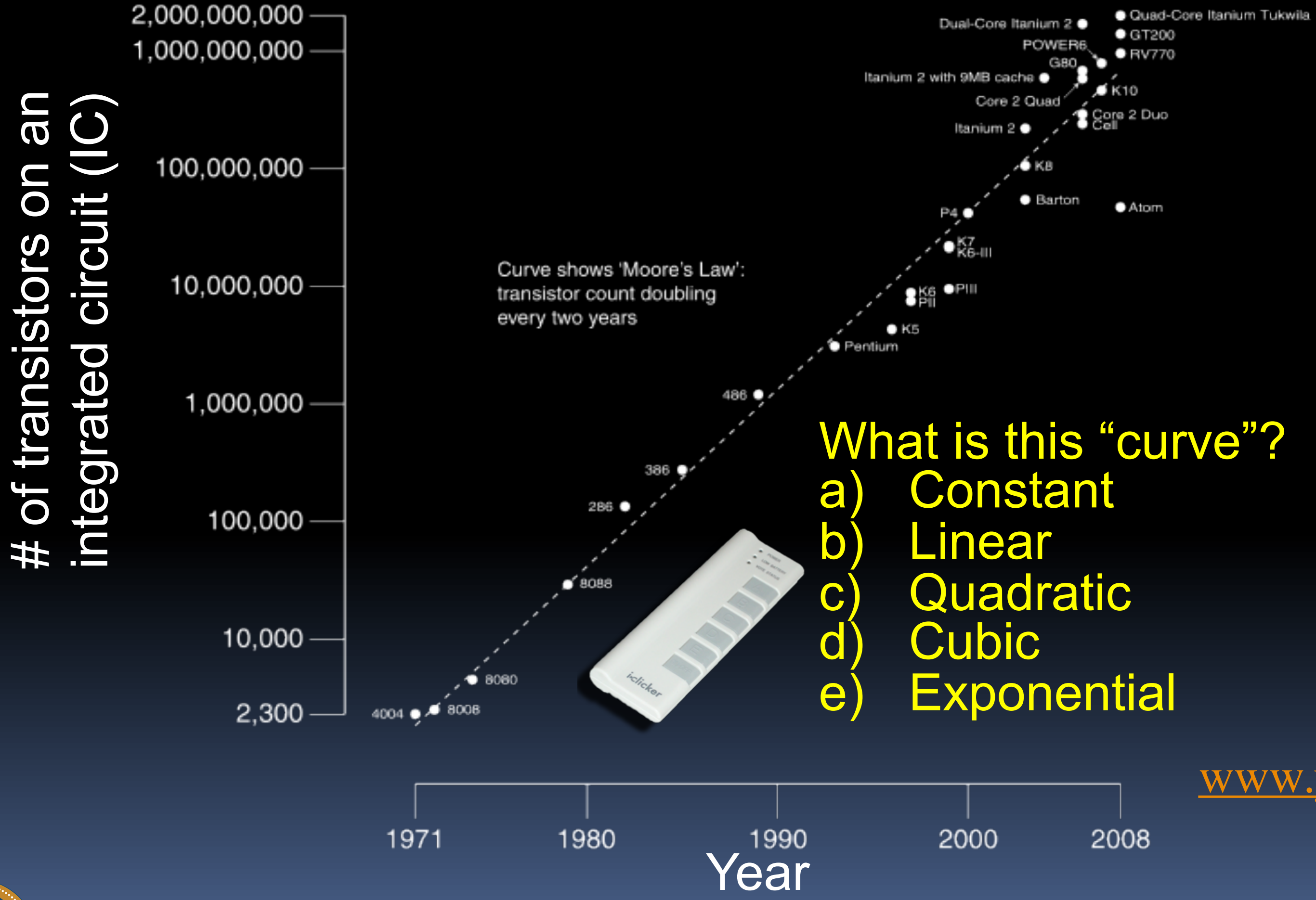
- Package provides:
  - spreading of chip-level signal paths to board-level
  - heat dissipation.
- Ceramic or plastic with gold wires.

Garcia

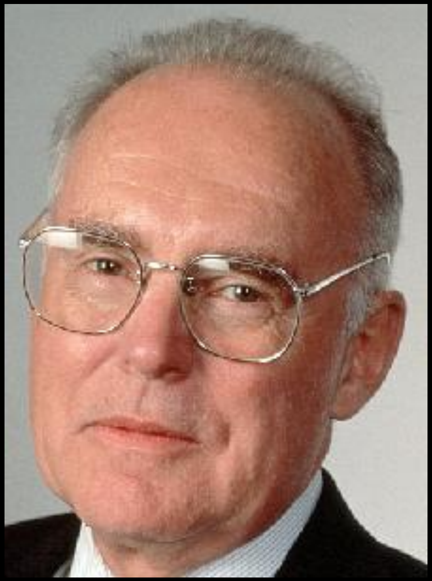
en.wikipedia.org/wiki/Moore's_law

Moore's Law

Predicts: 2X Transistors / chip every 2 years



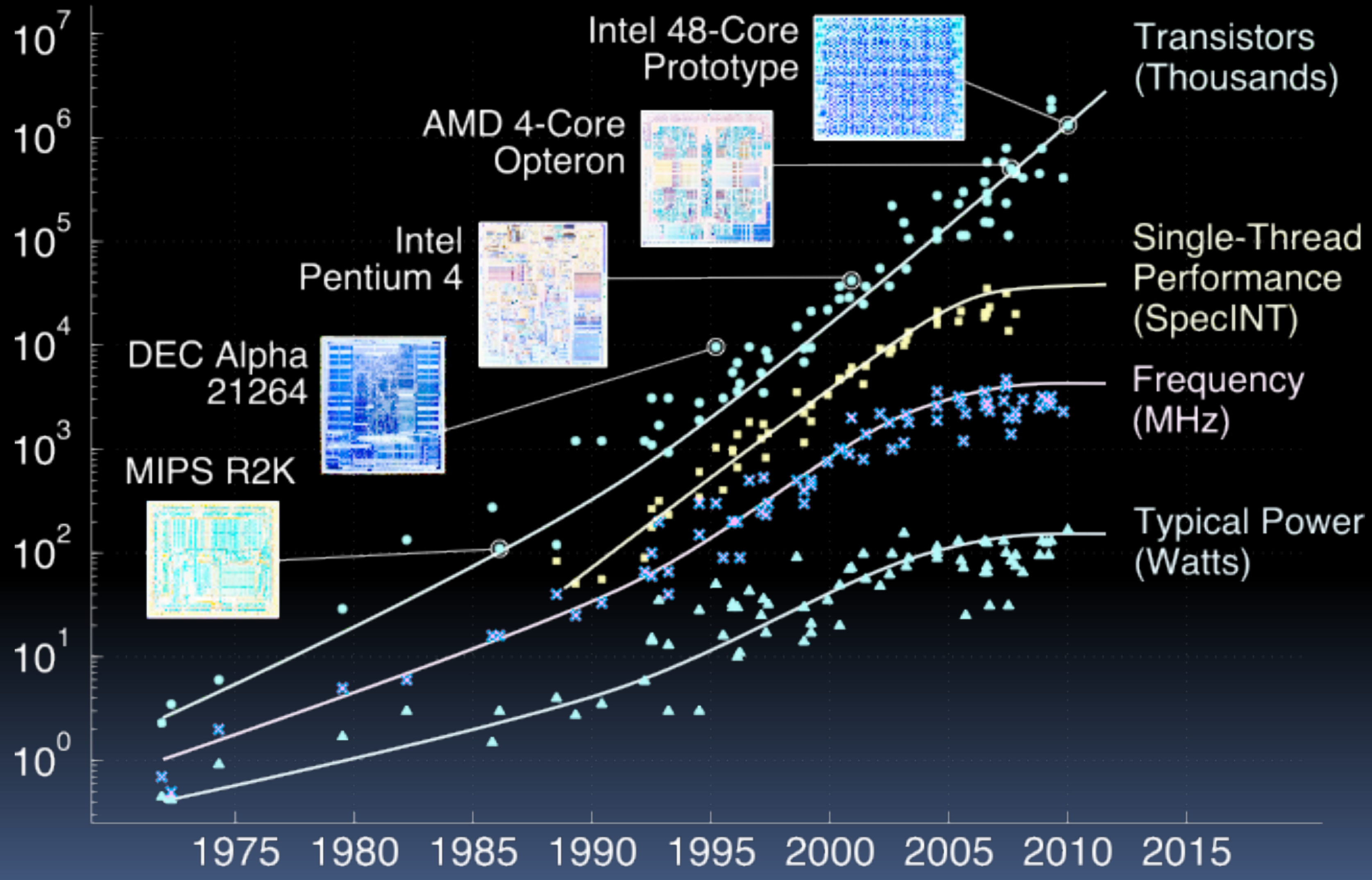
What is this "curve"?
a) Constant
b) Linear
c) Quadratic
d) Cubic
e) Exponential


Gordon Moore
Intel Cofounder
B.S. Cal 1950!

www.yellkey.com/security

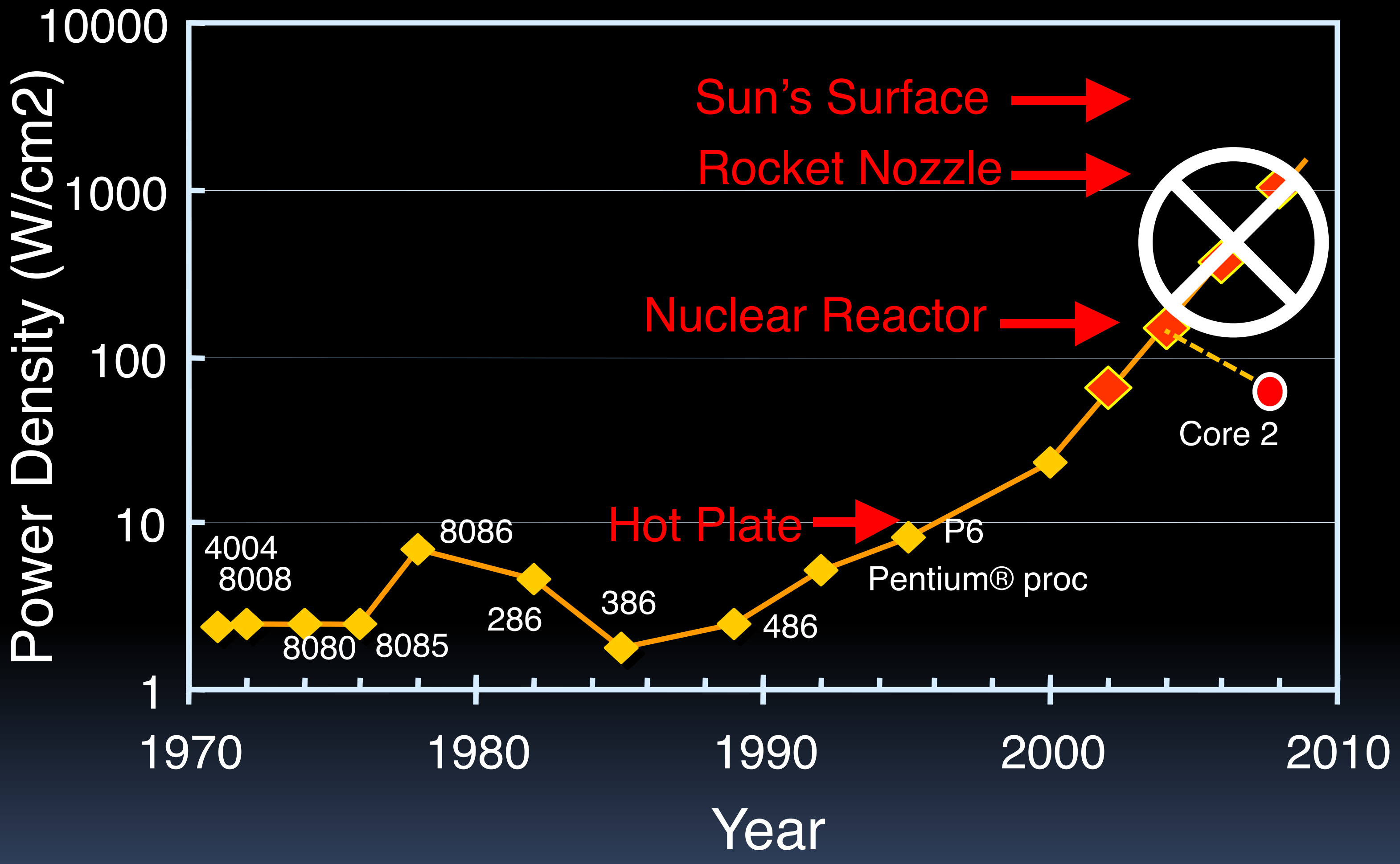
Garcia

Moore's Law and related curves



Data partially collected by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond

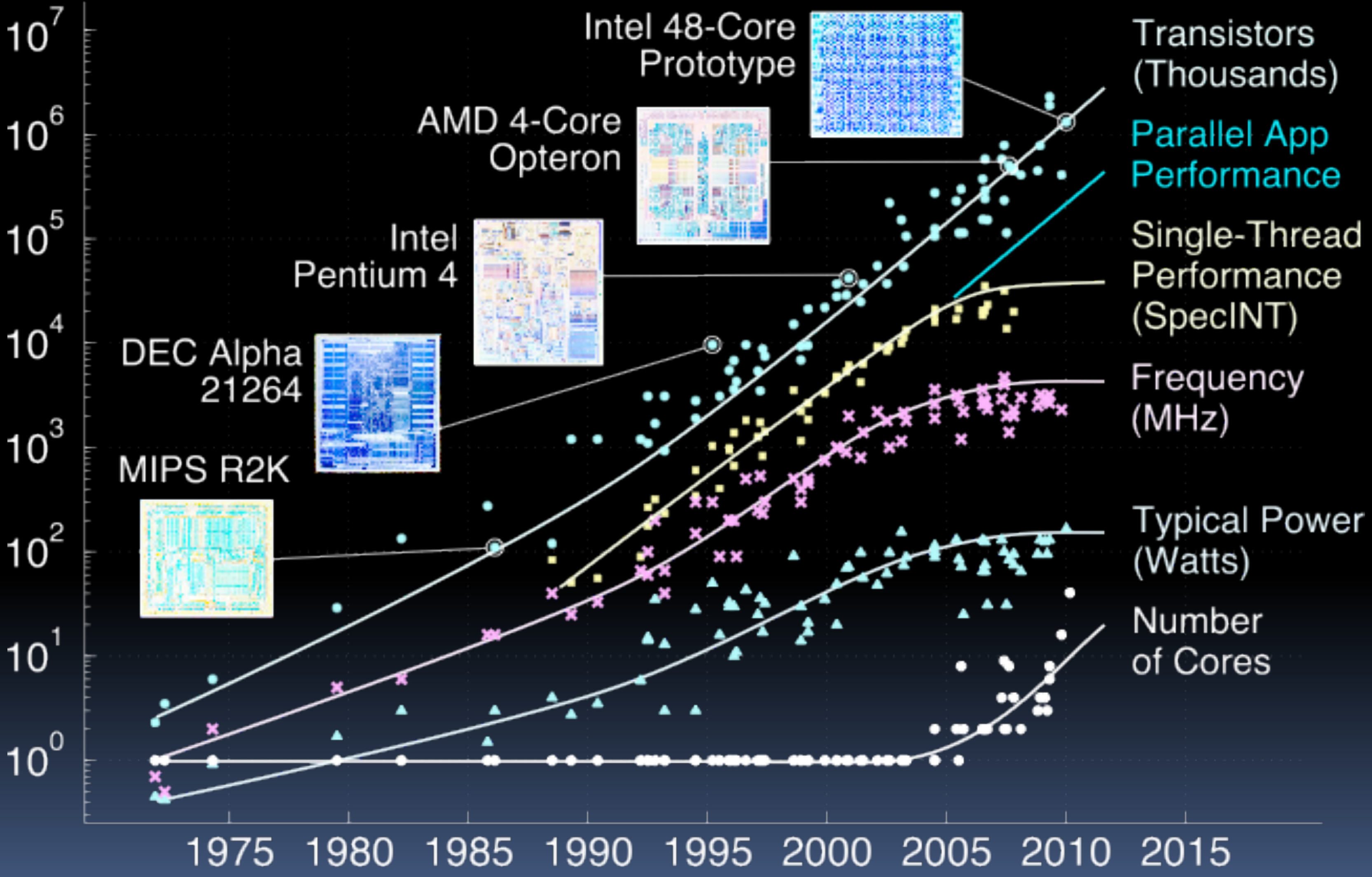
Power Density Prediction circa 2000



Source: S. Borkar (Intel)

Garcia

Moore's Law and related curves

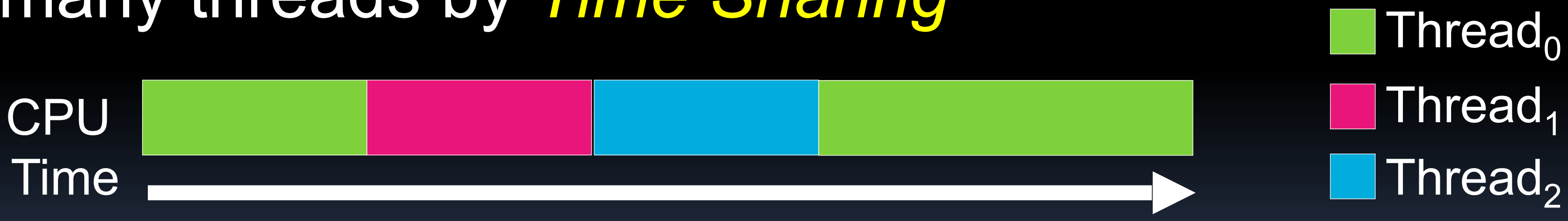


Data partially collected by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond

Garcia

Parallel Computing: Threads

- A *Thread* stands for "thread of execution"; it is a single stream of instructions
  - A program / process can split or fork itself into separate threads, which can (in theory) execute simultaneously.
  - An easy way to describe/think about parallelism
- With a single core, a single CPU can execute many threads by *Time Sharing*



- *Multithreading* is running multiple threads through the same hardware

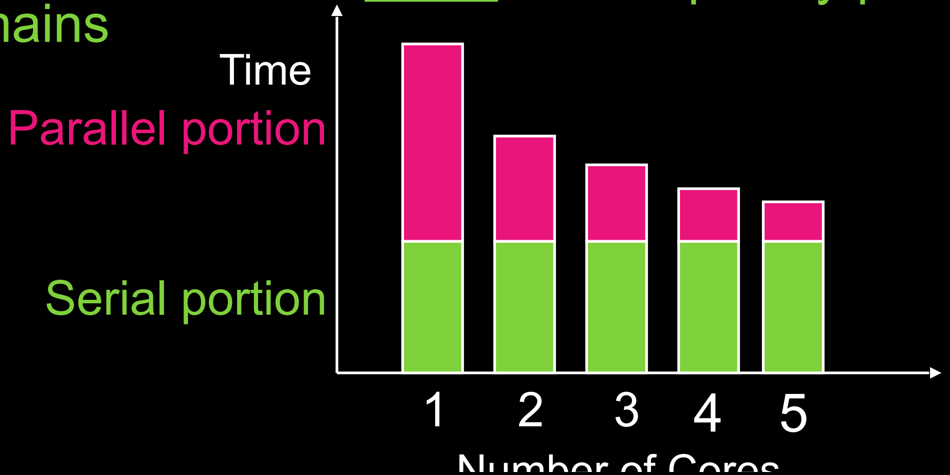
Speed: Making a Pizza

- Tasks:
  - Make the dough: 10 minutes
  - Make the sauce: 10 minutes
  - Prepare the toppings: 10 minutes
  - Assemble the pizza: 10 minutes
  - Bake the pizza: 40 minutes
- How long would this take with just one person?
- With two people? Three people? Four?
- Will adding more people always speed up the process?

en.wikipedia.org/wiki/Amdahl's_law

Speedup Issues : Amdahl's Law

- Applications can almost never be completely parallelized; some serial code remains



- s is serial fraction of program, P is # of cores (was processors)
- **Amdahl's law:**

Speedup(P) = Time(1) / Time(P)

$\leq 1 / (s + [(1-s) / P)]$, and as $P \to \infty$

$\leq 1 / s$

- Even if the parallel portion of your application speeds up perfectly, your performance may be limited by the sequential portion

Garcia

Speedup Issues : Overhead

- Even assuming no sequential portion, there's…
  - Time to think how to divide the problem up
  - Time to hand out small "work units" to workers
  - All workers may not work equally fast
  - Some workers may fail
  - There may be contention for shared resources
  - Workers could overwriting each others' answers
  - You may have to wait until the last worker returns to proceed (the slowest / weakest link problem)
  - There's time to put the data back together in a way that looks as if it were done by one

Life in a multi-core world…

- This "sea change" to multi-core parallelism means that the computing community has to rethink:
  a) Languages
  b) Architectures
  c) Algorithms
  d) Data Structures
  e) All of the above

www.yellkey.com/still

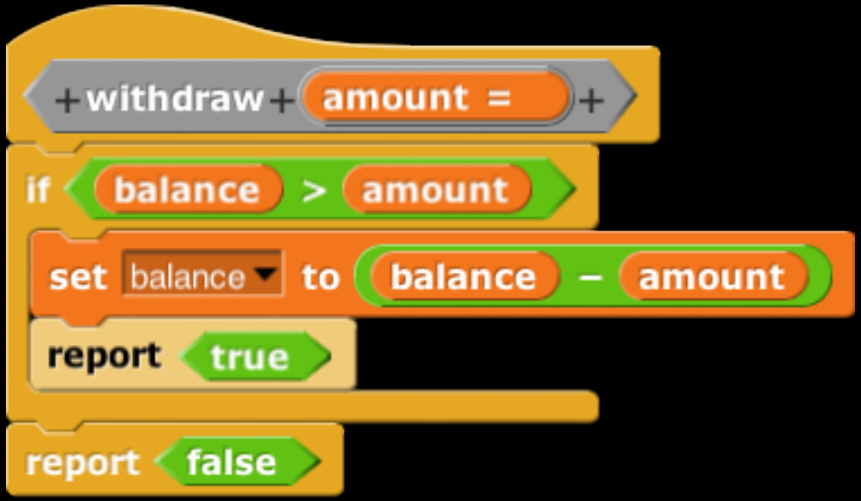
Garcia

BY NC SA

en.wikipedia.org/wiki/Concurrent_computing

Concurrency Issues

- What if two people were calling withdraw <u>at the same time</u>?
  - Assume balance=$100 and two people withdraw $75 each (same account)
  - Computer will try to parallelize this process by executing each call to “withdraw” on a separate core



```
+withdraw+ amount = +
if balance > amount
    set balance to balance − amount
    report true
report false
```

- Can anyone see what the problem could be?
  - Problem: There is only one copy of “balance” in memory. If both threads access “balance” simultaneously, people will withdraw more money than they actually have!
  - Bigger problem: Program <u>may or may not</u> work properly

Garcia

BY NC SA

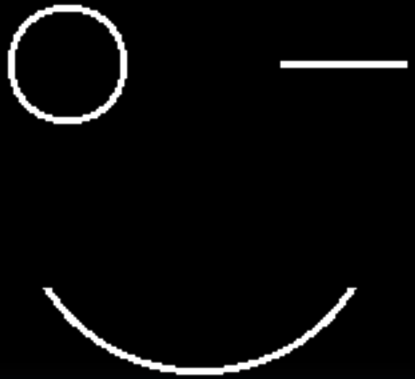
Race Condition

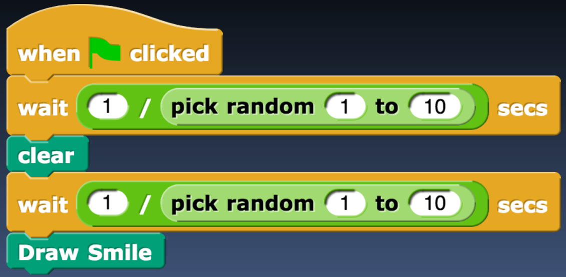
- Two (or more) scripts are running at the same time, BUT we don't know what order they will be run in! This is called a race condition.
- Very common in many programming languages. This will not happen in Snap*!* because it will swap between running script A and script B and won't let both of them run at once. (Phew!)
- …but if you have some explicit calls to "wait random secs" in your code & use these for timers, it can happen!

Possible Outputs?

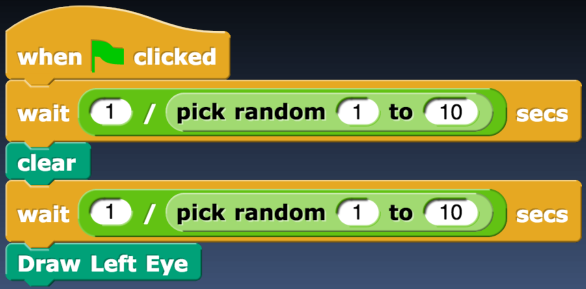
www.yellkey.com/notice

- We want this code to draw a cute winky-face, but there's a problem with parallelizing it!
- Which of the following outputs is <u>not</u> possible?
  - A) No output
  - B) Just left eye
  - C) Just left eye and smile
  - D) Just right eye and smile
  - E) Whole face

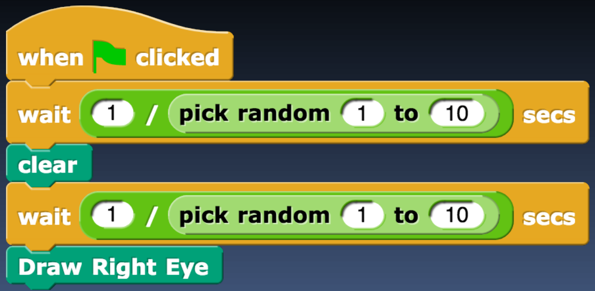




```
when [flag] clicked
wait (1 / pick random 1 to 10) secs
clear
wait (1 / pick random 1 to 10) secs
Draw Smile
```



```
when [flag] clicked
wait (1 / pick random 1 to 10) secs
clear
wait (1 / pick random 1 to 10) secs
Draw Left Eye
```



```
when [flag] clicked
wait (1 / pick random 1 to 10) secs
clear
wait (1 / pick random 1 to 10) secs
Draw Right Eye
```

en.wikipedia.org/wiki/Deadlock

Another concurrency issue… deadlock!

- Deadlock: Threads are waiting on each other before acting, no action
  - Ex: You see an acquaintance walking on campus, don't want to say hi first. Your acquaintance doesn't want to say hi first either. You wait on your acquaintance to say hi first, your acquaintance waits on you, no one says hi
- Livelock also possible: Threads keep taking action without waiting for each other, no progress
  - Ex: Passing someone in a narrow hall, keep moving out of each other's way and blocking each other

Summary

- "Sea change" of computing because of inability to cool CPUs means we're now in multi-core world
- This brave new world offers lots of potential for innovation by computing professionals, but challenges persist