Parallel Programming:
Moore's Law and Multicore

Oregon State
University

Mike Bailey
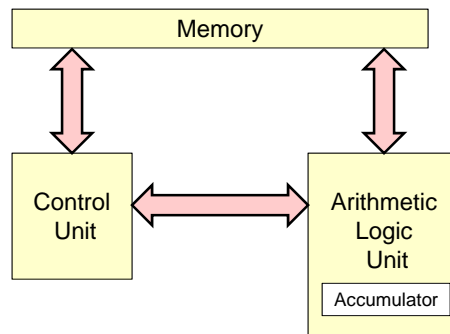
mjb@cs.oregonstate.edu

Oregon State
University
Computer Graphics

moores.law.and.multicore.pptx

mjb – March 15, 2020

Von Neumann Architecture:
Basically the fundamental pieces of a CPU
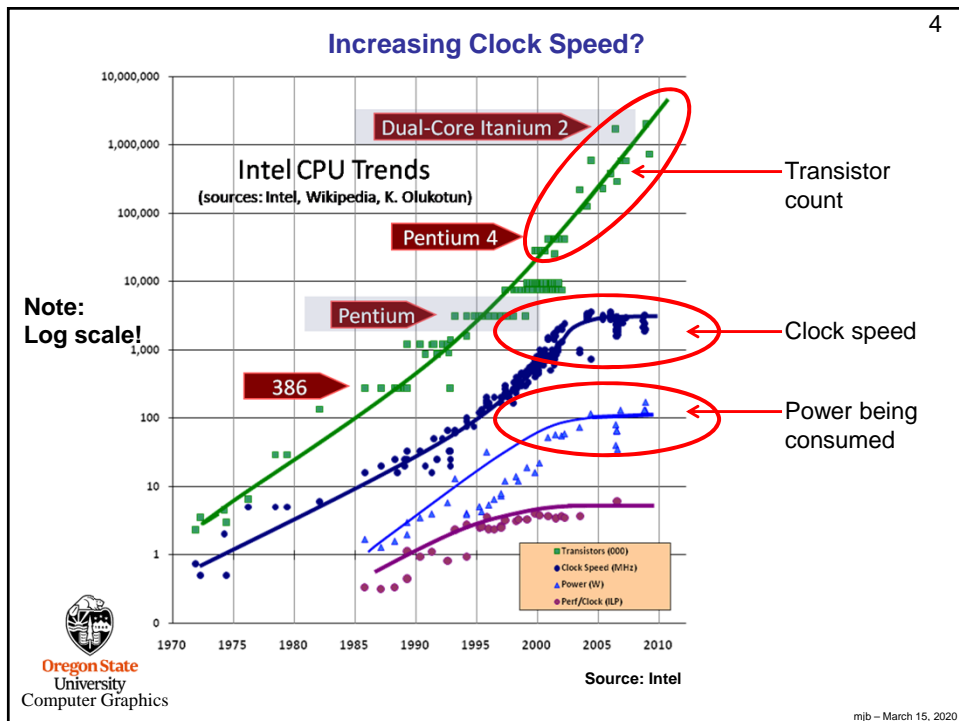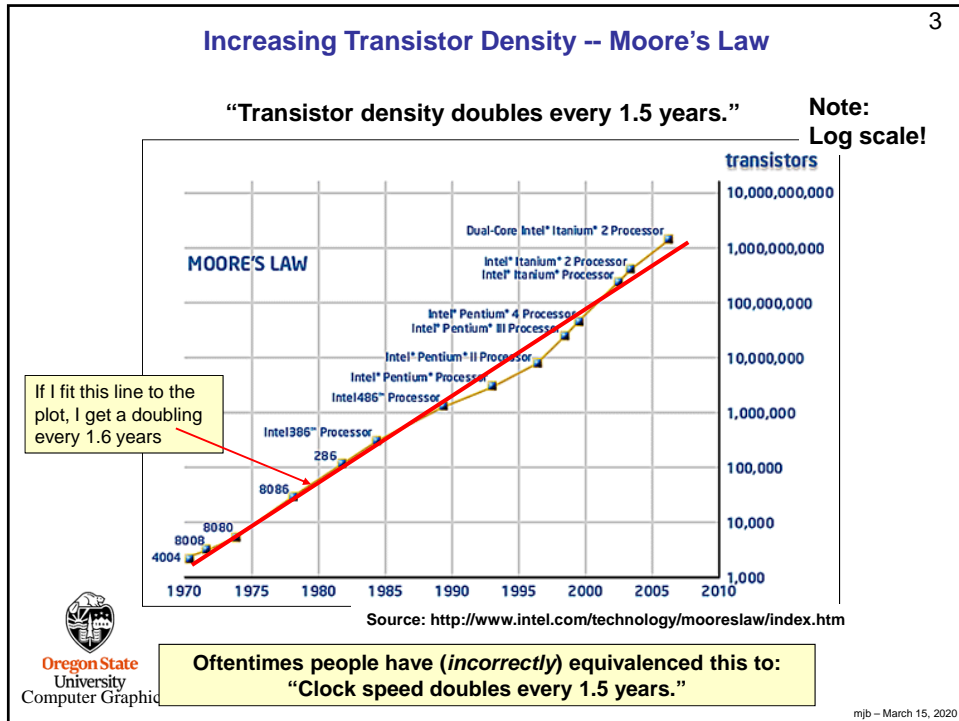# have not changed since the 1960s

Memory

Control Unit

Arithmetic Logic Unit

Accumulator

Other elements:
- **Clock**
- Registers
- Program counter
- Stack pointer

Oregon State
University
Computer Graphics

mjb – March 15, 2020

Increasing Transistor Density -- Moore's Law

"Transistor density doubles every 1.5 years."

**Note:
Log scale!**

transistors

10,000,000,000

1,000,000,000

MOORE'S LAW

Dual-Core Intel® Itanium® 2 Processor

Intel® Itanium® 2 Processor
Intel® Itanium® Processor

100,000,000

Intel® Pentium® 4 Processor
Intel® Pentium® III Processor

Intel® Pentium® II Processor
Intel® Pentium® Processor

10,000,000

Intel486™ Processor

1,000,000

If I fit this line to the plot, I get a doubling every 1.6 years

Intel386™ Processor

286

100,000

8086

10,000

8080
8008
4004

1,000

1970   1975   1980   1985   1990   1995   2000   2005   2010

Source: http://www.intel.com/technology/mooreslaw/index.htm

Oregon State
University
Computer Graphics

**Oftentimes people have (*incorrectly*) equivalenced this to:
"Clock speed doubles every 1.5 years."**

mjb – March 15, 2020

Increasing Clock Speed?

10,000,000

1,000,000

Dual-Core Itanium 2

Intel CPU Trends
(sources: Intel, Wikipedia, K. Olukotun)

100,000

Pentium 4

Transistor
count

10,000

**Note:
Log scale!**

Pentium

1,000

Clock speed

386

100

Power being
consumed

10

1

■ Transistors (000)
● Clock Speed (MHz)
▲ Power (W)
● Perf/Clock (ILP)

0

1970   1975   1980   1985   1990   1995   2000   2005   2010

Oregon State
University
Computer Graphics

Source: Intel

mjb – March 15, 2020

Moore's Law

▪ Fabrication process size ("gate pitch") has fallen from 65 nm, to 45 nm, to 32 nm, to 22 nm, to 16 nm, to 11 nm, to 8 nm. This translates to more transistors on the same size die.

▪ From 1986 to 2002, processor performance increased an average of 52%/year, but then virtually plateaued.

Oregon State
University
Computer Graphics

mjb – March 15, 2020
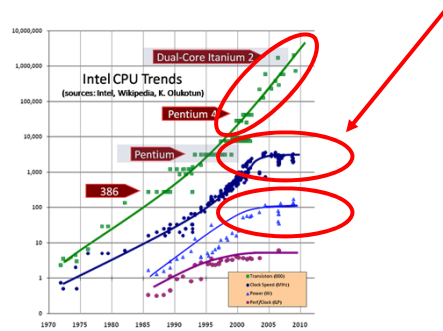
Clock Speed and Power Consumption

1981	IBM PC	5 MHz
1995	Pentium	100 MHz
2002	Pentium 4	3000 MHz (3 GHz)
2007		3800 MHz (3.8 GHz)
2009		4000 MHz (4.0 GHz)

Clock speed has hit a plateau, largely because of power consumption and dissipation.

$$PowerConsumption \propto ClockSpeed^2$$

Yikes!

is-proportional-to

Once consumed, that power becomes *heat*, which much be dissipated somehow. In general, compute systems can remove around 150 $^{watts}/_{cm}$ without resorting to exotic cooling methods.

Oregon State
University
Computer Graphics

mjb – March 15, 2020

And, speaking of "exotic", AMD set the world record for 7
clock speed (8.429 GHz) using a Liquid Nitrogen-cooled CPU
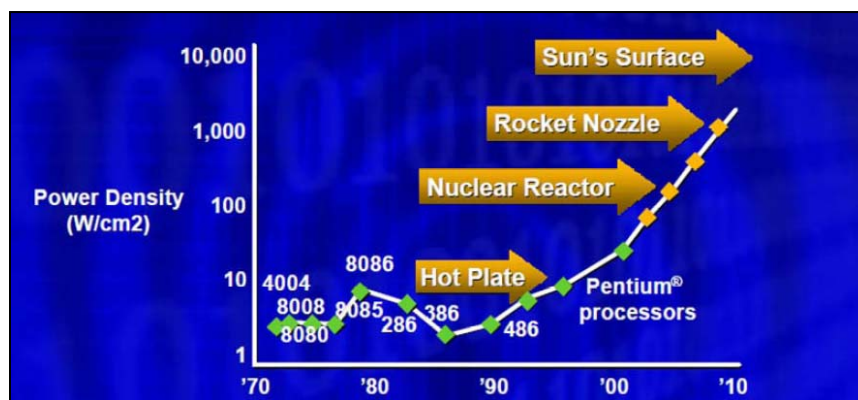

Source: *AMD*

Oregon State
University
Computer Graphics

mjb – March 15, 2020

What Kind of Power Density Dissipation Would it Have Taken 8
to Keep up with Clock Speed Trends?

Source: Intel

Oregon State
University
Computer Graphics

mjb – March 15, 2020

MultiCore -- Multiprocessing on a Single Chip

So, to summarize:

Moore's Law of transistor density is still going, but the "Moore's Law" of clock speed has hit a wall. Now what do we do?

We keep packing more and more transistors on a single chip, but don't increase the clock speed. Instead, we increase computational throughput by using those transistors to pack multiple processors onto the same chip.

This is referred to as **multicore**.

Vendors have also reacted by adding SIMD floating-point units on the chip as well. We will get to that later.

Oregon State
University
Computer Graphics

mjb – March 15, 2020

MultiCore and Multithreading

Multicore, even without multithreading too, is still a good thing. It can be used, for example, to allow multiple programs on a desktop system to always be executing concurrently.

Multithreading, even without multicore too, is still a good thing. Threads can make it easier to logically have many things going on in your program at a time, and can absorb the dead-time of other threads.

But, the big gain in performance is to use *both* to speed up a *single program*. For this, we need *a **combination of both multicore and multithreading***.

Multithreading

Both
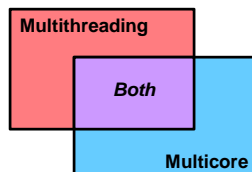
Multicore

Multicore is a very hot topic these days. It would be hard to buy a CPU that doesn't have more than one core. We, as programmers, get to take advantage of that.
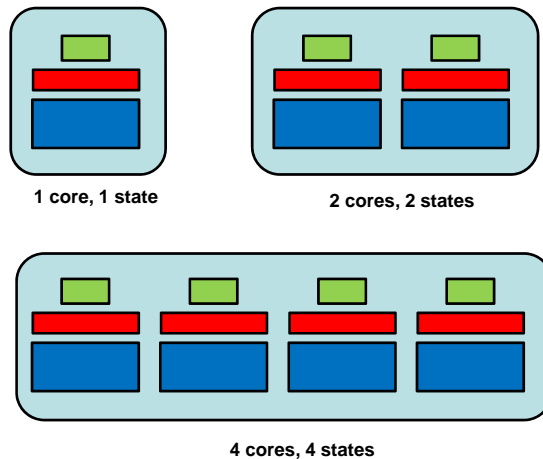
We need to be prepared to convert our programs to run on ***MultiThreaded Shared Memory Multicore*** architectures.

Oregon
University
Computer Graphics

mjb – March 15, 2020

Each of the Multiple Cores keeps its own State

1 core, 1 state

2 cores, 2 states

4 cores, 4 states

State → • Registers
Core   • Program Counter
Cache   • Stack Pointer

Oregon State
University
Computer Graphics

mjb – March 15, 2020

So, if that's what Multicore is about, what is *Hyperthreading*?

1 core, 1 state

1 core, 2 states, with Hyperthreading

2 cores, 2 states

2 cores, 4 states, with Hyperthreading

4 cores, 4 states

State
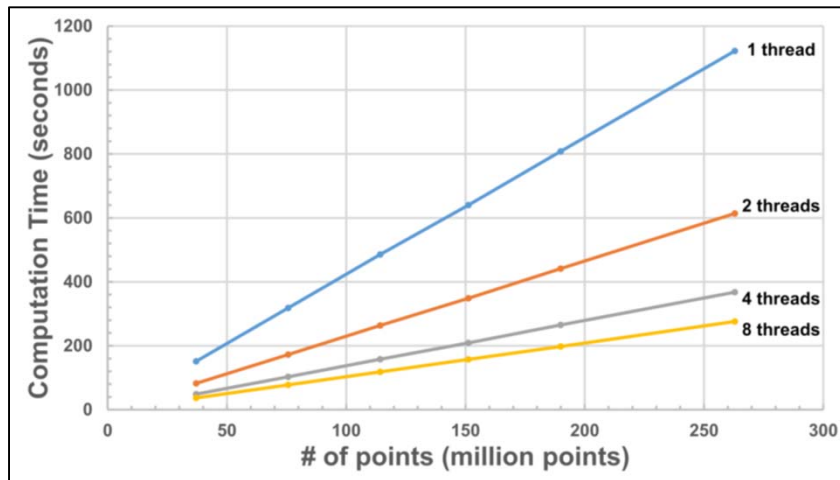Core
Cache

Oregon State
University
Computer Graphics
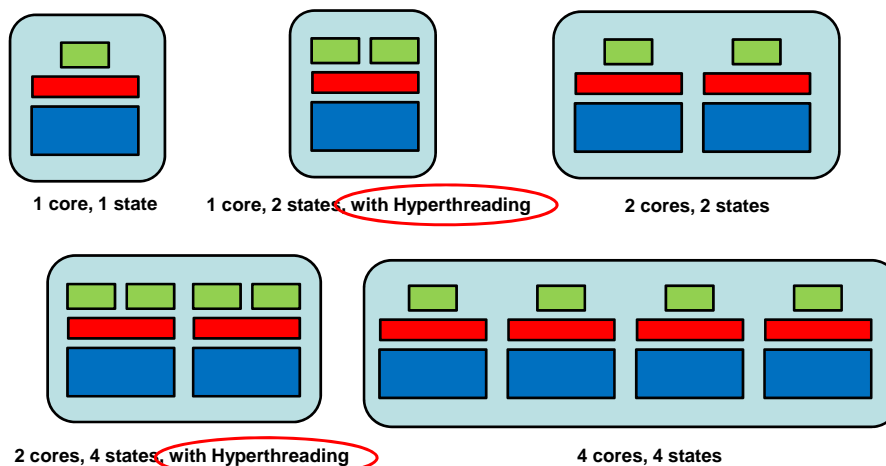
mjb – March 15, 2020

Four Cores with Two Hyperthreads per Core

Source: Erzhuo Che

Note that this is upside-down from our usual convention. Sorry. I got this from someone else.

Oregon State
University
Computer Graphics

mjb – March 15, 2020