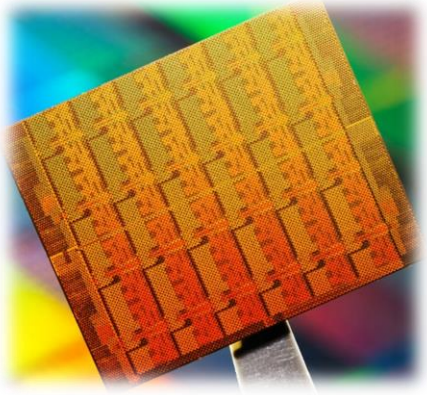
Introduction to Computer Architecture



Computer Architecture

> "Computer architecture is a specification detailing how a set of software and hardware technology standards interact to form a computer system or platform. In short, computer architecture refers to how a computer system is designed and what technologies it is compatible with. "
>
> source: Technopedia

- Computer architecture teaches you
  - how a computer is ***controlled***
  - how a computer is ***built***

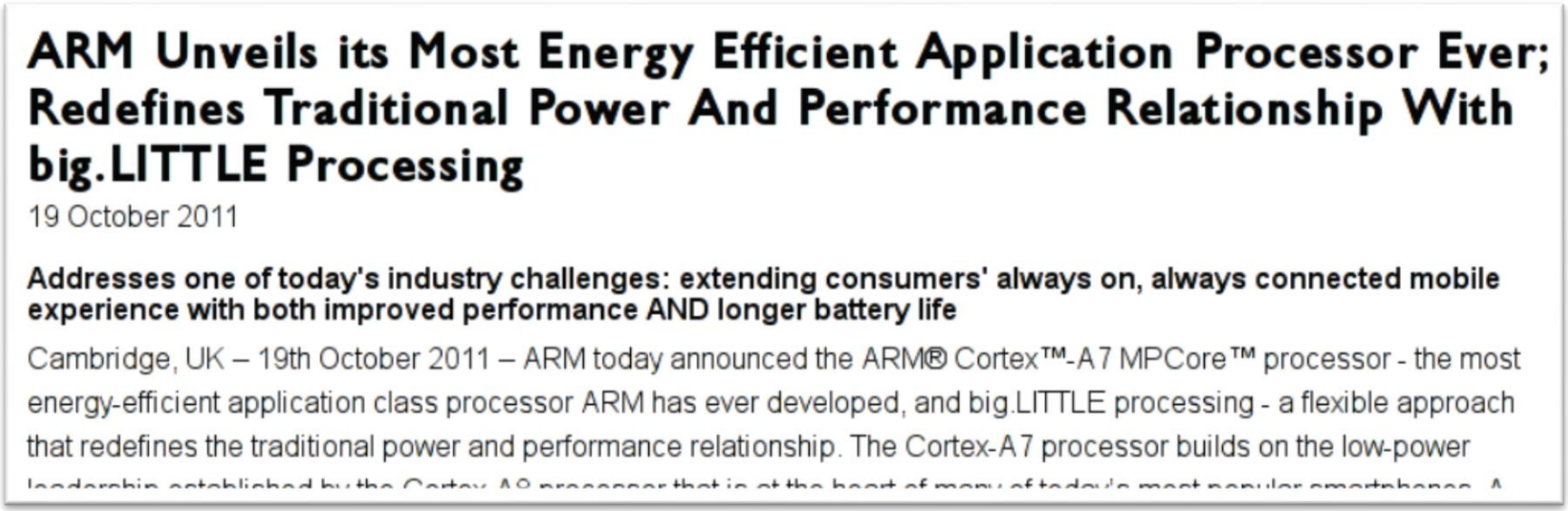
After This Course, You Will…

- understand the **functionality and operation of the basic elements of a computer system** including the processor, memory, and input/output
- understand the **hardware/software interface**
- understand and be able to decode programs written in **assembly language**

Credits For The Entire Course

- Slides and material adapted mainly from
  - the csapp textbook
  - H&P's "Computer Architecture: A Quantitative Approach"
  - Tannenbaum's "Structured Computer Organization"
  - slides and projects from CMU
  - slides from David Black-Schaffer (introduction)

Why Study Computer Architecture?



ARM Unveils its Most Energy Efficient Application Processor Ever; Redefines Traditional Power And Performance Relationship With big.LITTLE Processing

19 October 2011

Addresses one of today's industry challenges: extending consumers' always on, always connected mobile experience with both improved performance AND longer battery life

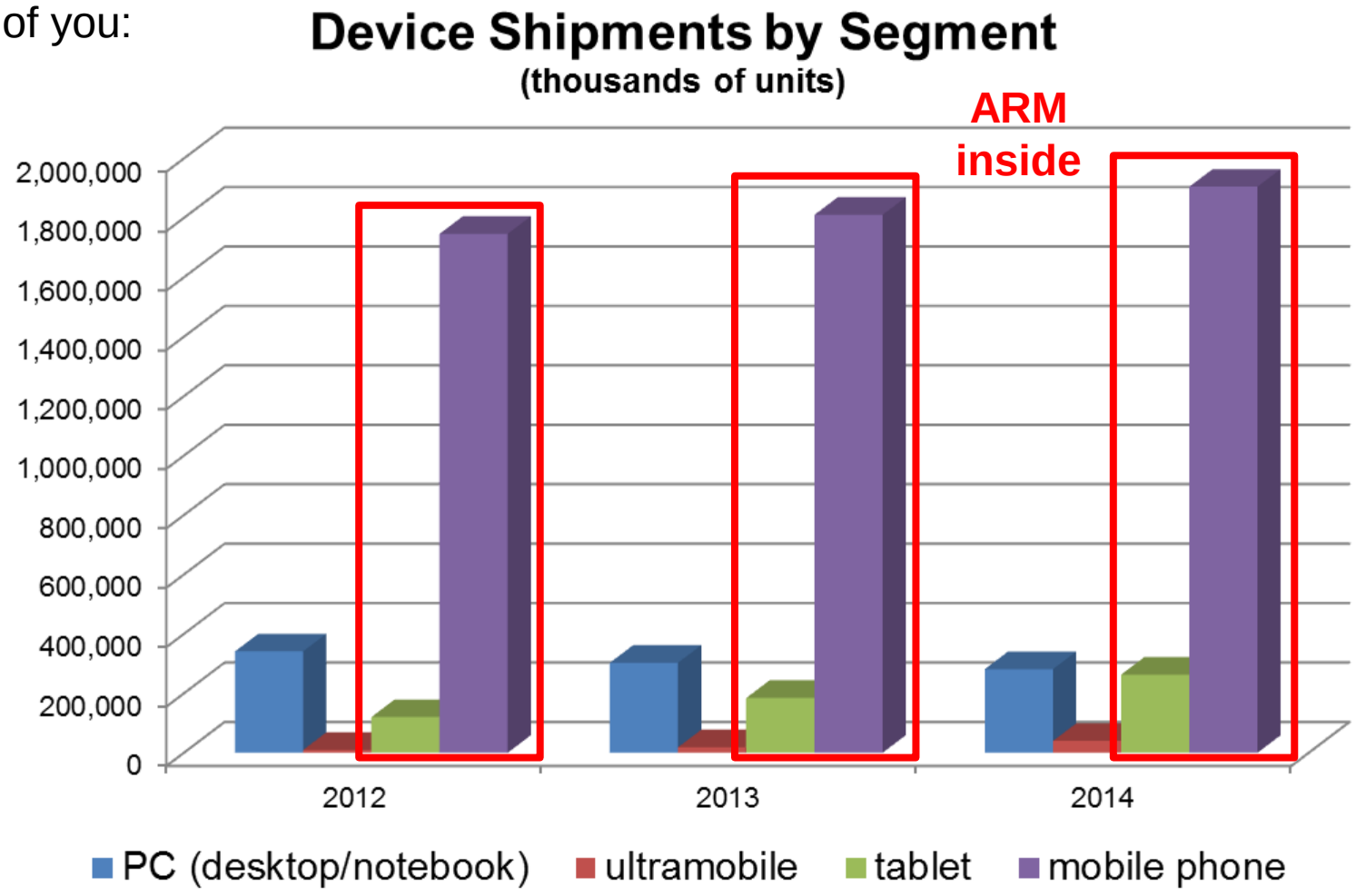
Cambridge, UK – 19th October 2011 – ARM today announced the ARM® Cortex™-A7 MPCore™ processor - the most energy-efficient application class processor ARM has ever developed, and big.LITTLE processing - a flexible approach that redefines the traditional power and performance relationship. The Cortex-A7 processor builds on the low-power

(source: ARM)

- big.LITTLE processing?
- ARM?

Who Owns Device With an ARM Inside?

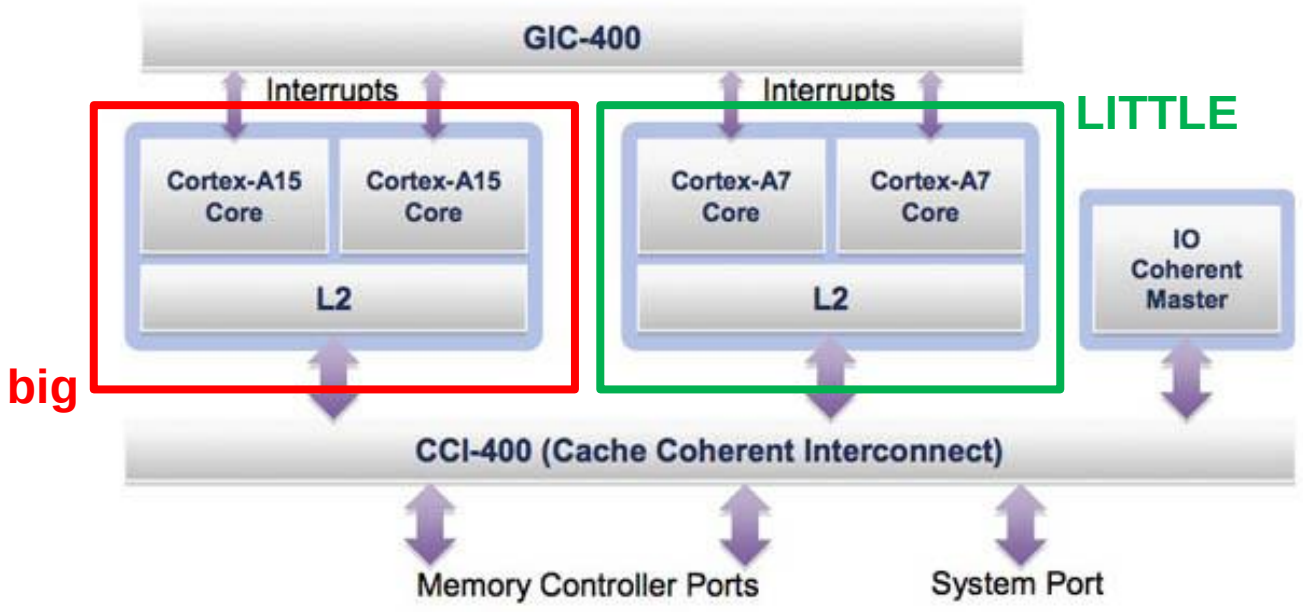
- All of you:



source: http://www.gartner.com/newsroom/id/2610015

What Are They Doing With big.LITTLE?

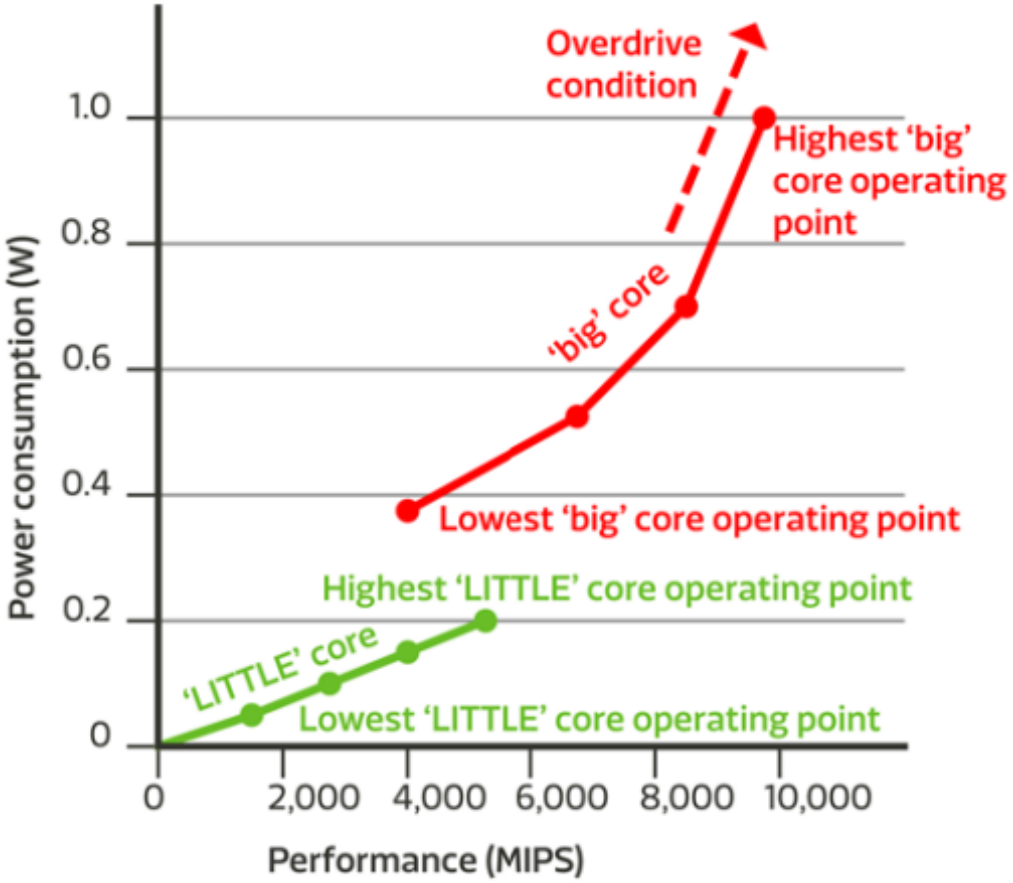
- big.LITTLE
  - big cores for high performance
  - little cores for low performance



source: http://www.eetimes.com/document.asp?doc_id=1279167

And Why Are They Doing It?

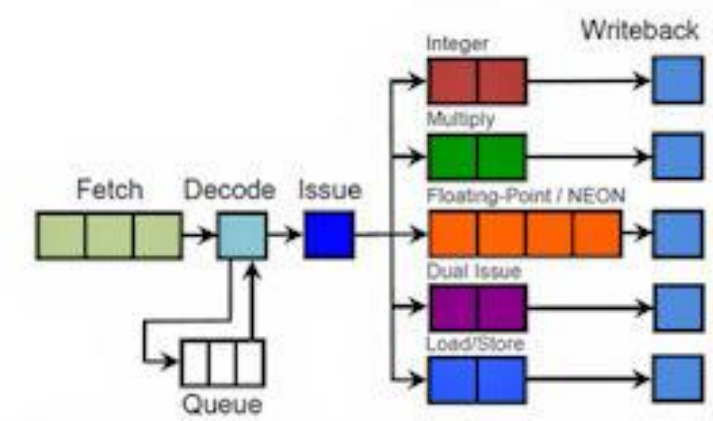
- One reason: **power efficiency = computations / energy**



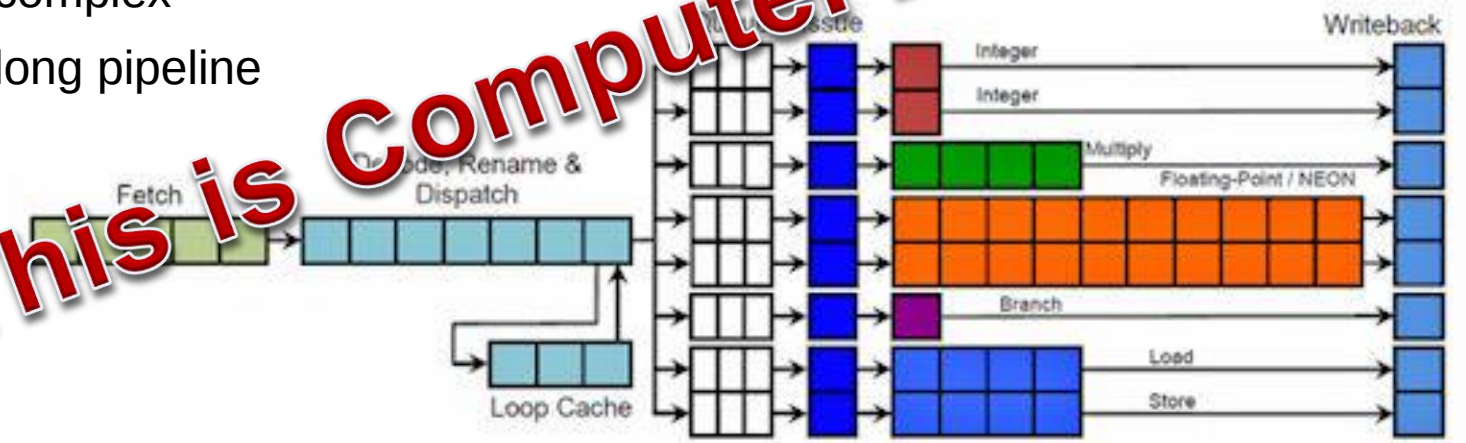
source: http://www.mobilegeeks.com/mediatek-outperforms-qualcomm-welcome-true-octa-core-processing/

The Details

- **LITTLE cores**
  - simple
  - short pipeline
- **big cores**
  - complex
  - long pipeline



ARM Cortex-A7 Pipeline



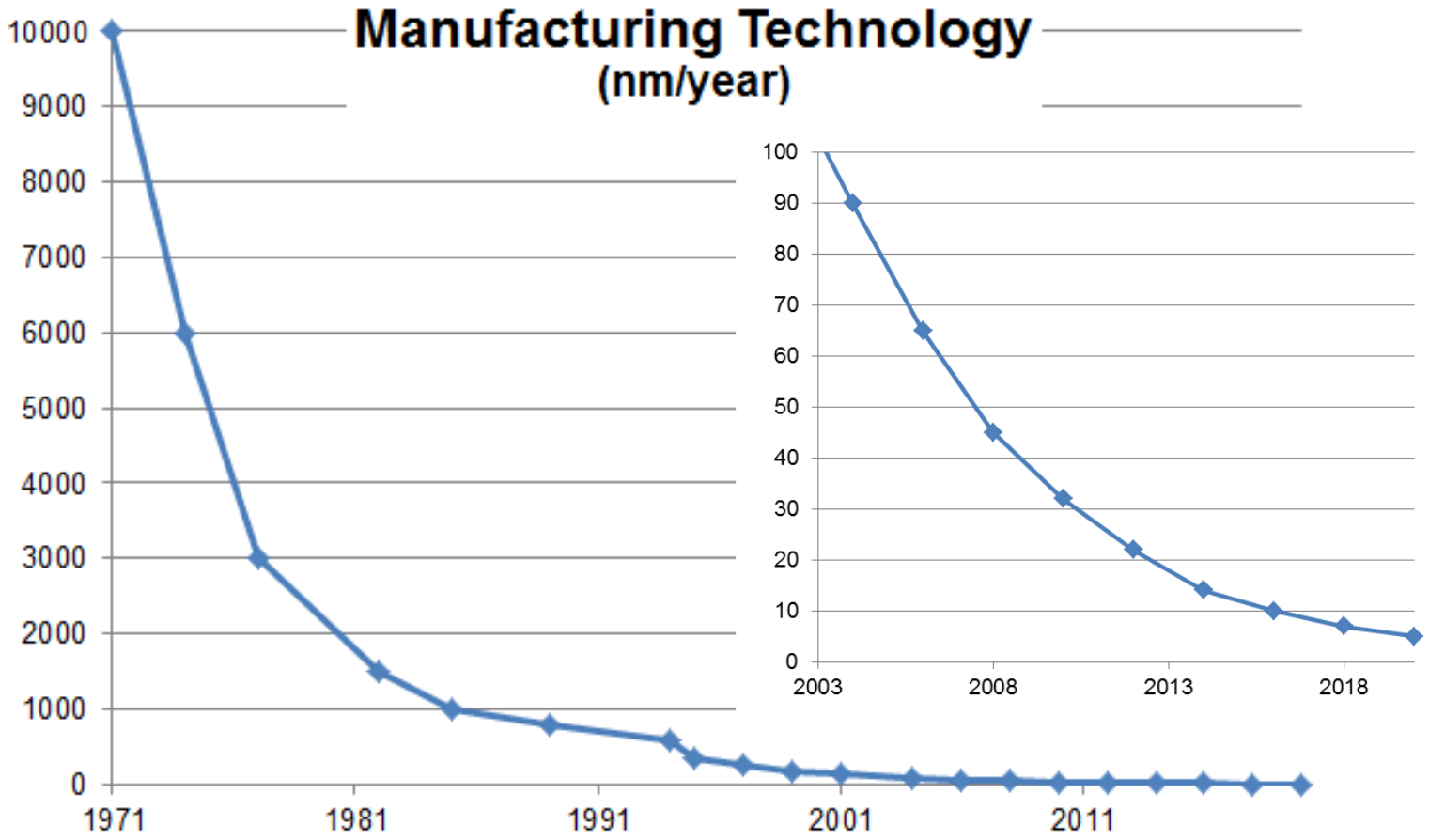
ARM Cortex-A15 Pipeline

source: http://www.eetimes.com/document.asp?doc_id=1279167

This is Computer Architecture

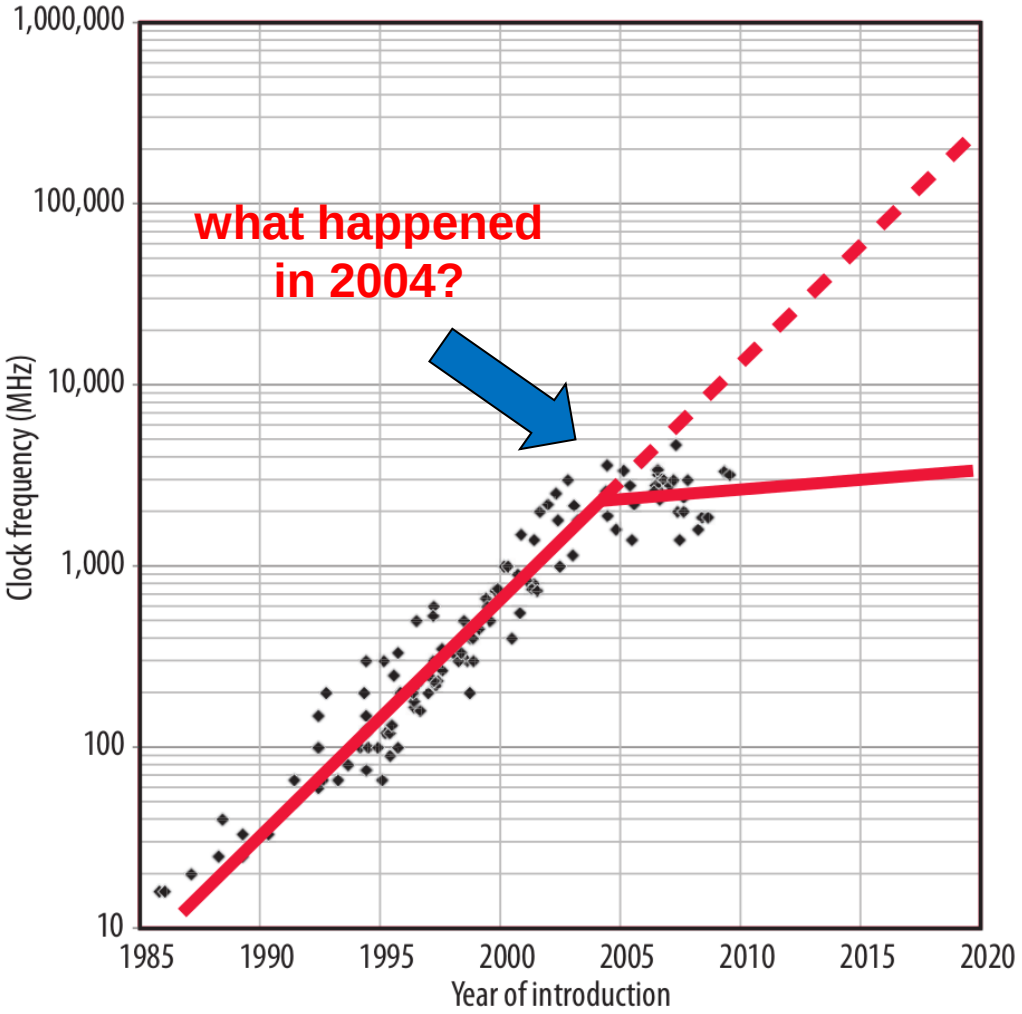
Why Can They Do This?

- Technology scaling: 130nm → 22nm



Why Should They Do This?

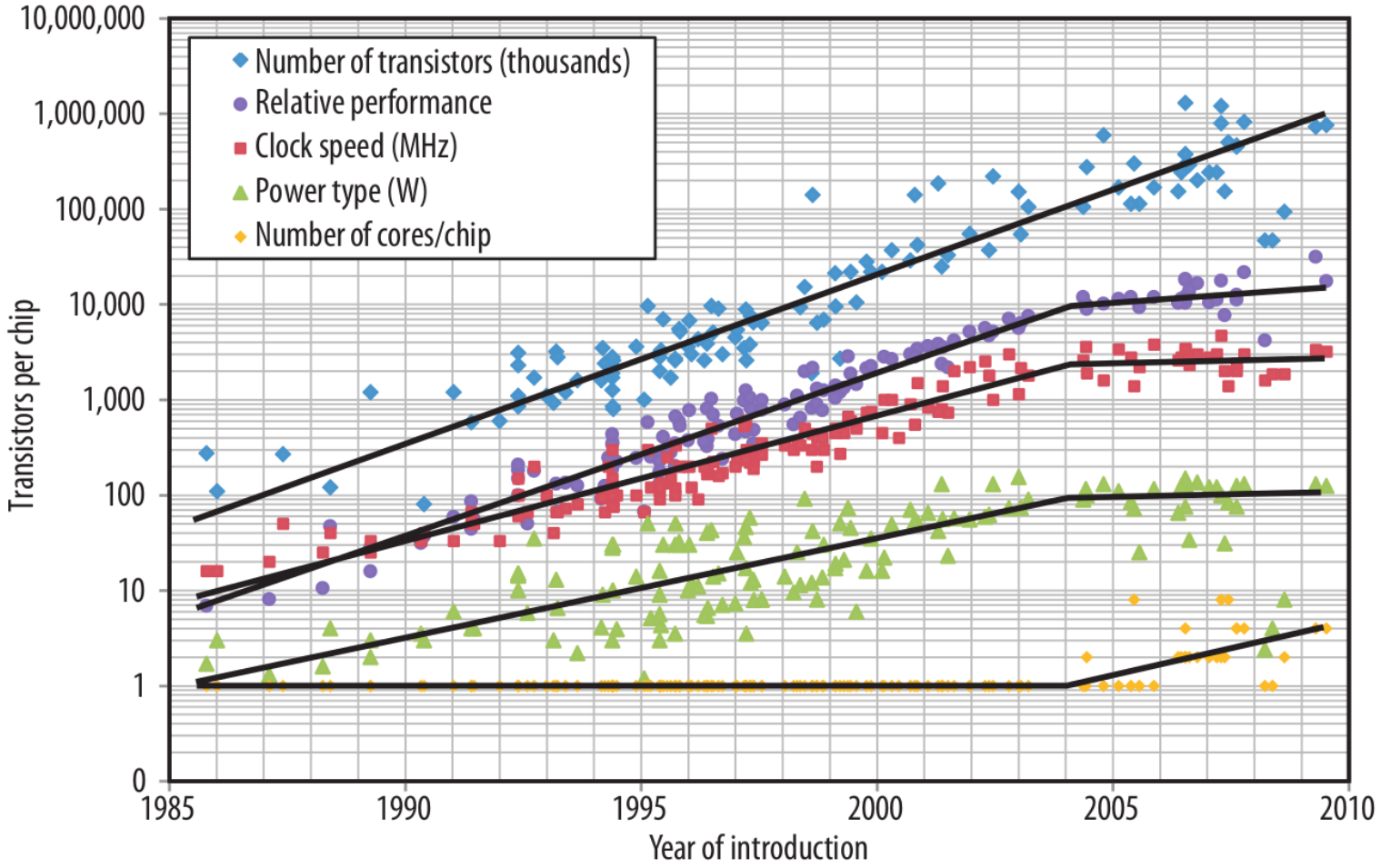
- Single-core performance



source: Fuller et al. Computing Performance: Game Over or Next Level, IEEE, 2011

The Power Wall

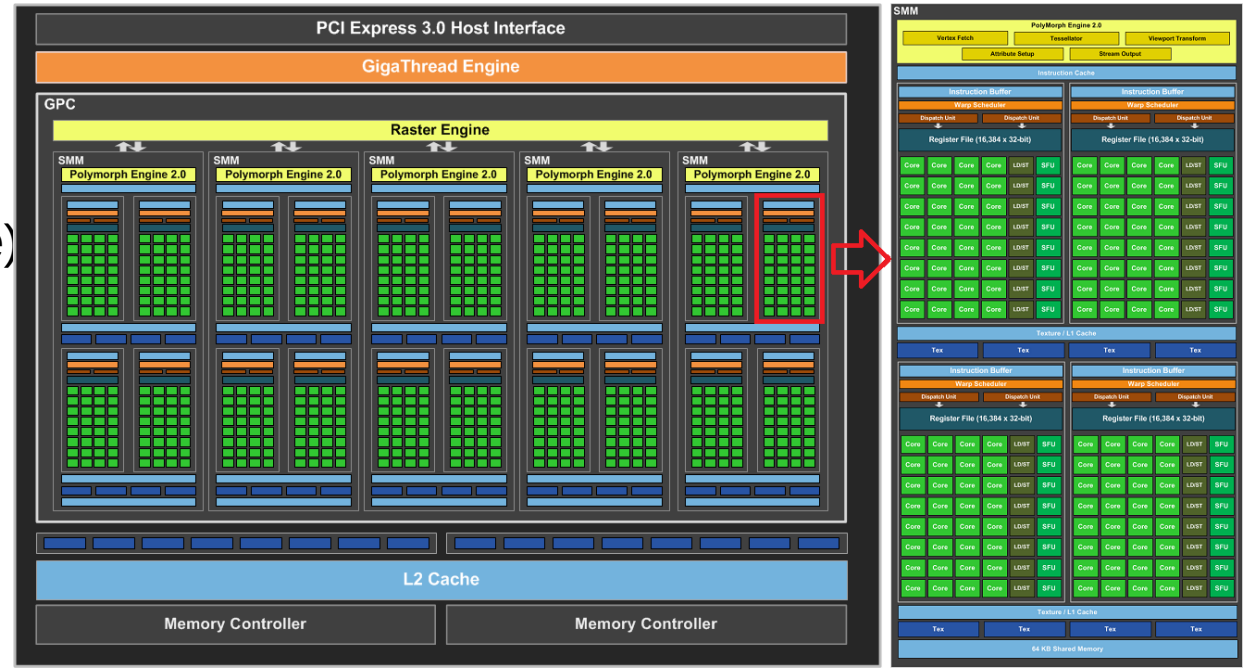
- We can't increase power, need to increase power efficiency



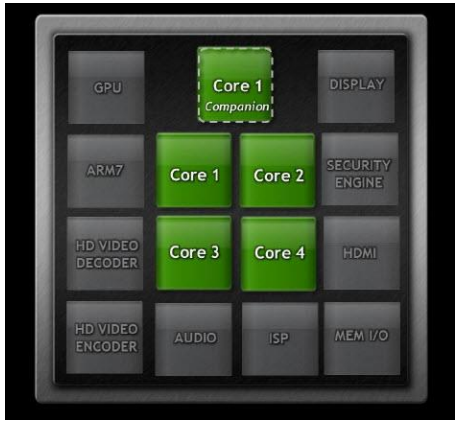
source: Fuller et al. Computing Performance: Game Over or Next Level, IEEE, 2011

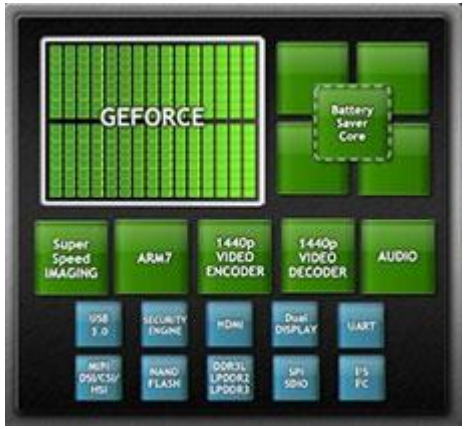
What Are Others Doing?

- NVIDIA
  - GPUs
    (Maxwell architecture)



  - mobile CPUs

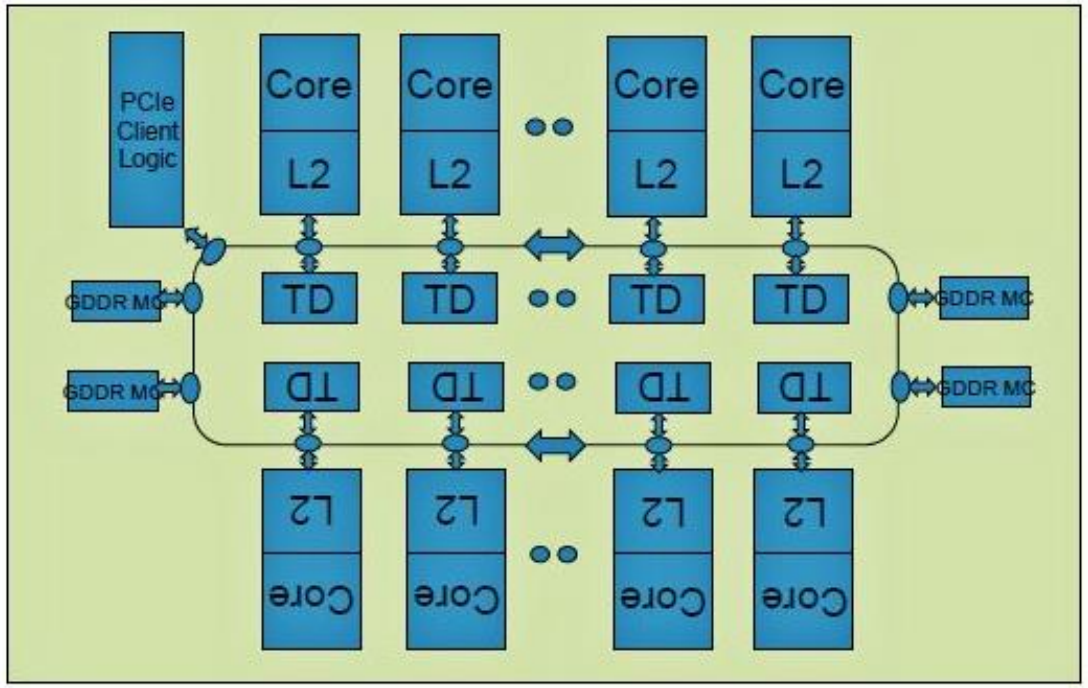




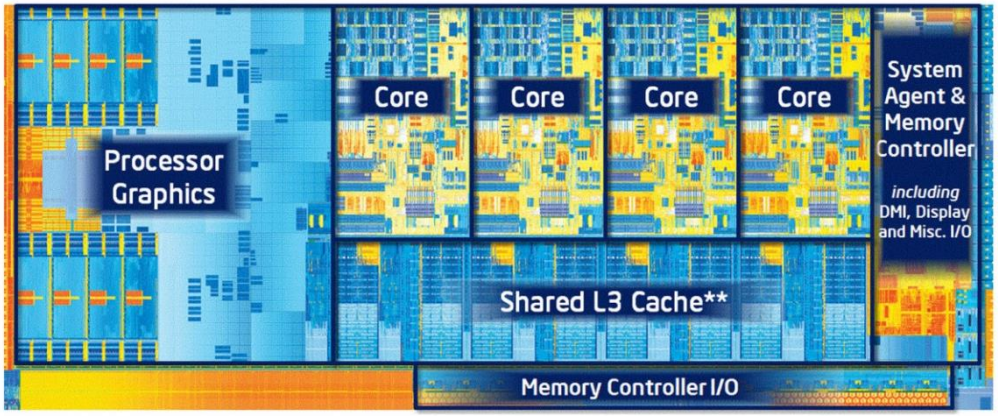
source: NVIDIA

What Are Others Doing?

- Intel
  - MIC
    (Many Integrated Cores)

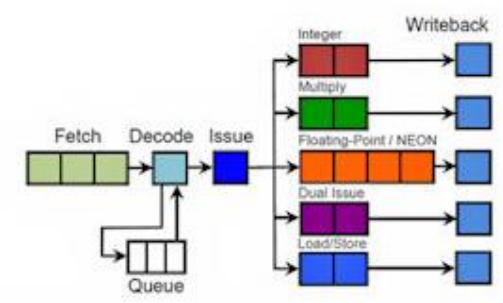


  - desktop/mobile CPUs

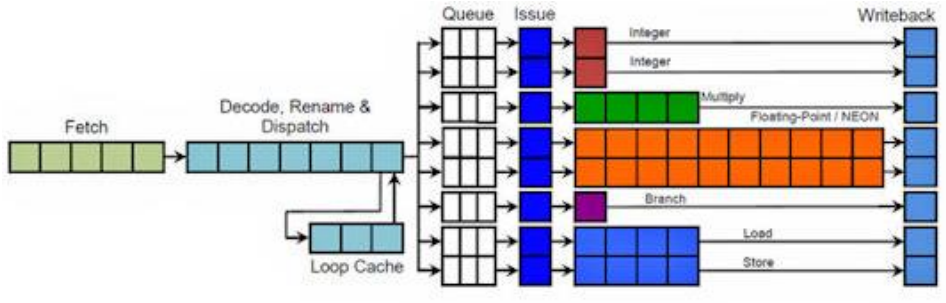


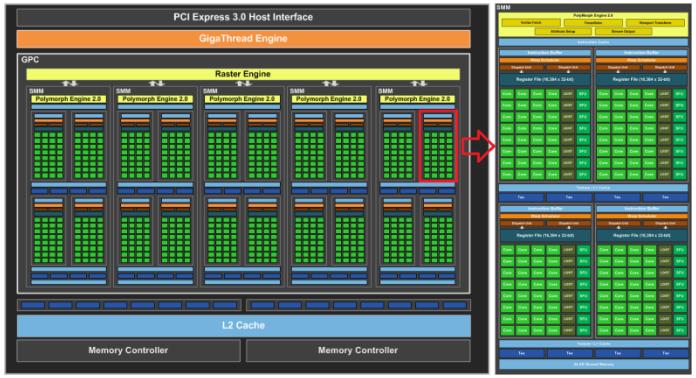
source: Intel

This is Computer Architecture

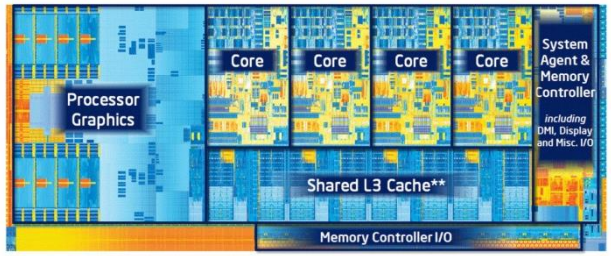


vs





vs



- Understanding the building blocks of processors and computer systems
- Understanding design tradeoffs such as performance vs efficiency
- Building the hardware
- Making it programmable

Why You Should Care...

- …apart from this being a mandatory class (at least for CSE students)?

- Understanding computer architecture is important for many other core subjects in computer science (system programming, OS, compilers, programming models and languages, …)
- Understanding computer architecture will make you a better programmer

```
for (i=0; i<N; i++) {
  for (j=0; j<N; j++) {
    C[i,j] = A[i,j] + B[i,j];
  }
}
```

vs

```
for (j=0; j<N; j++) {
  for (i=0; i<N; i++) {
    C[i,j] = A[i,j] + B[i,j];
  }
}
```

- Understanding assembly and a processor's ISA is still an important skill
- It is actually quite fun!

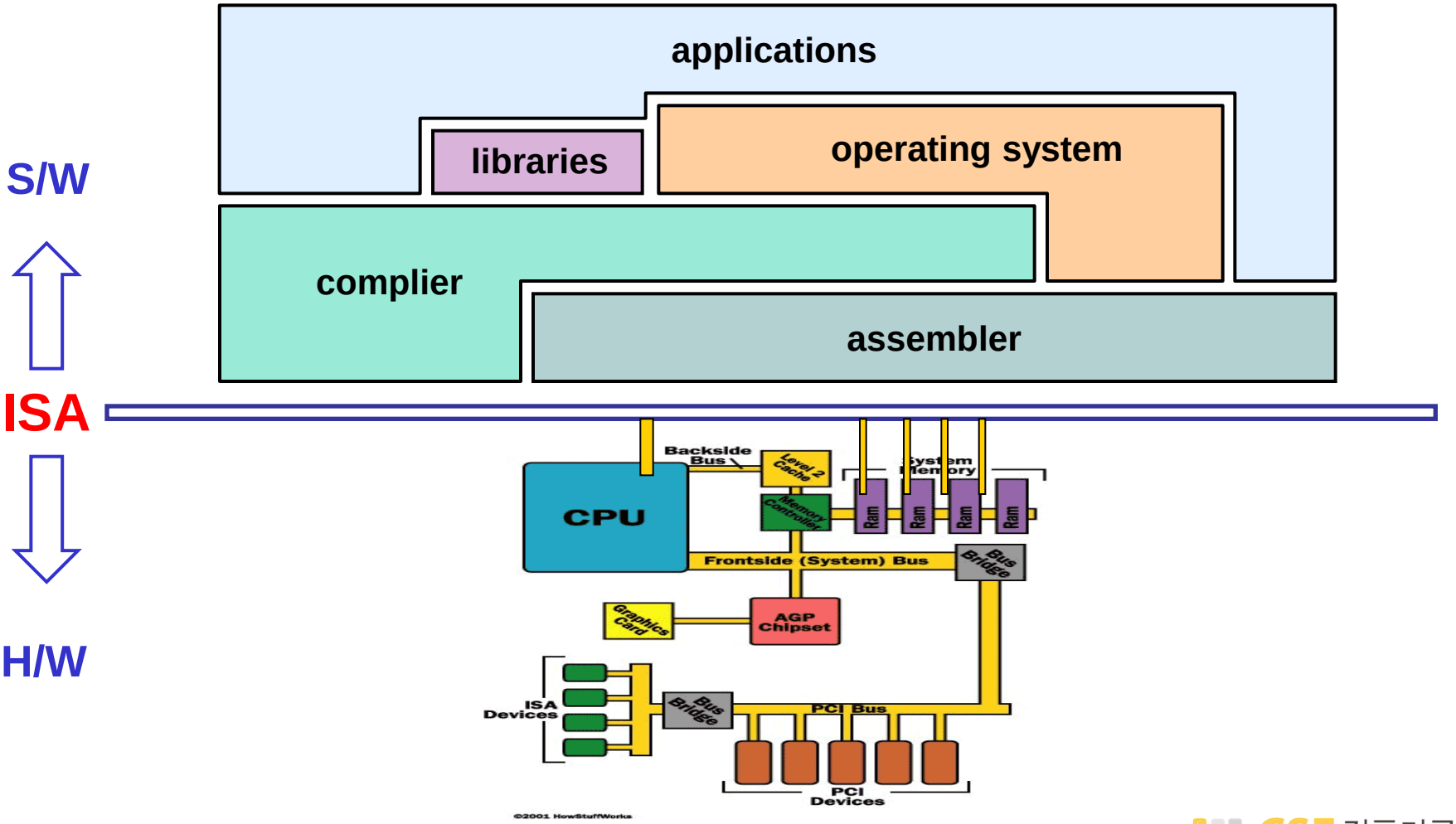
What Are We Going To Do In This Course?

- We will *not*
  - design the next Core i9 with a 30-stage pipeline
  - study how logic gates work (you know that already)
  - write entire programs in assembly

- We will
  - learn an ISA (x86) and how to read and understand assembly programs
  - learn how a simple pipelined processor is built
  - learn about the memory hierarchy in modern computer systems
  - have a quick look at modern state-of-the-art processors

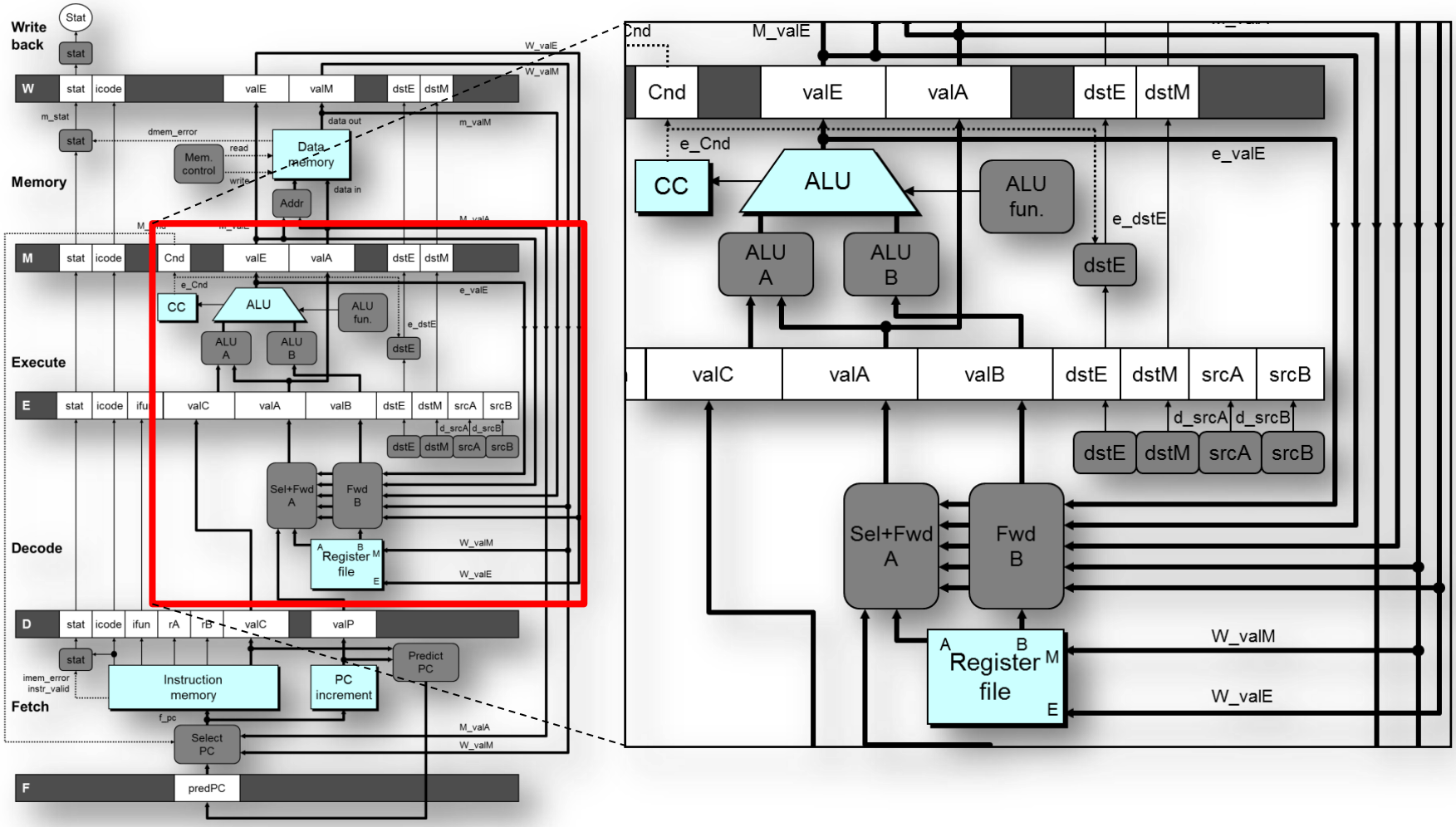
What Are We Going To Do In This Course?

- The ISA (Instruction Set Architecture) as the hardware/software interface



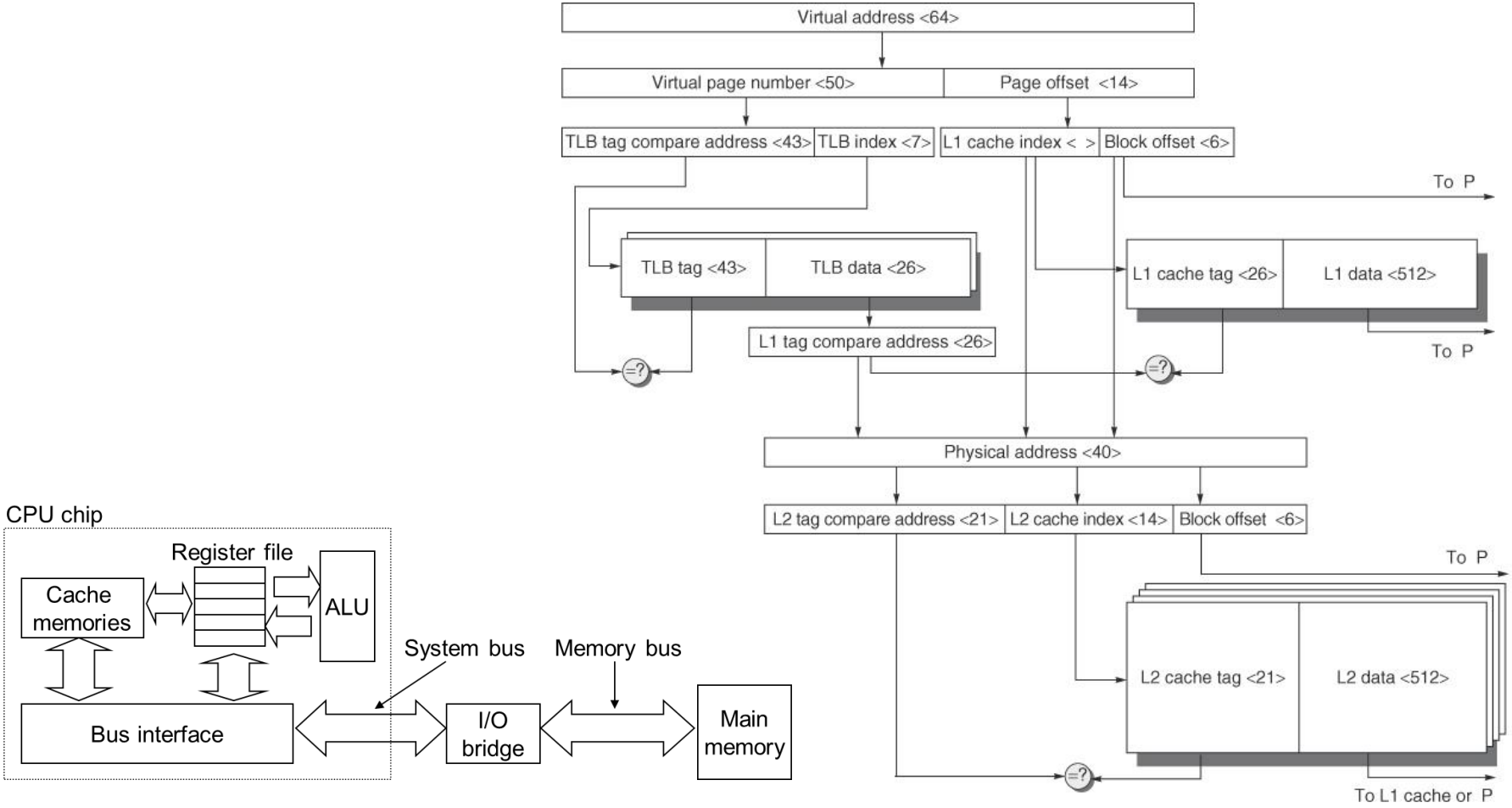
What Are We Going To Do In This Course?

- **Study and modify a ”simple” pipelined processor**



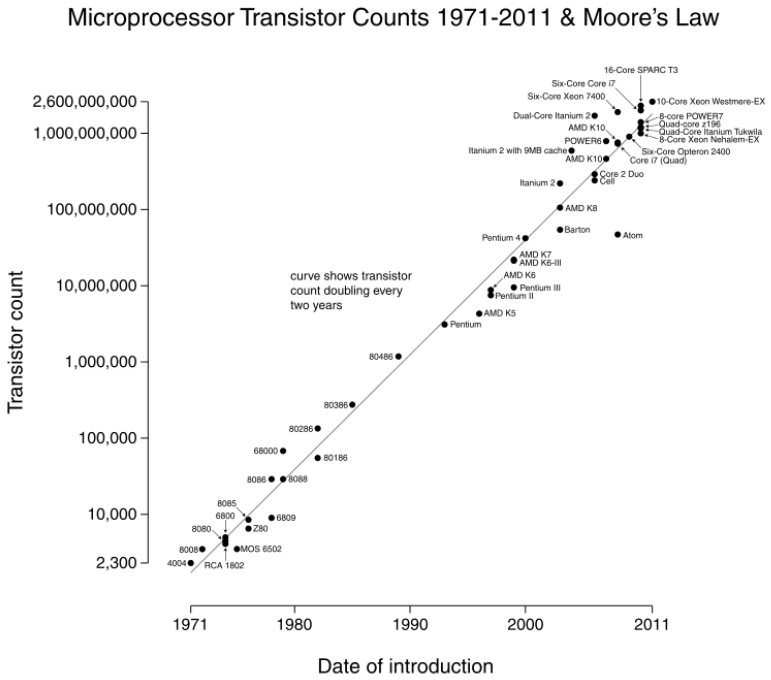
What Are We Going To Do In This Course?

- **Study the memory hierarchy in modern computer systems**



Great Ideas in Computer Architecture

- **Design for Moore's Law**
  - anticipate state of technology when the product ships



source: Wikipedia

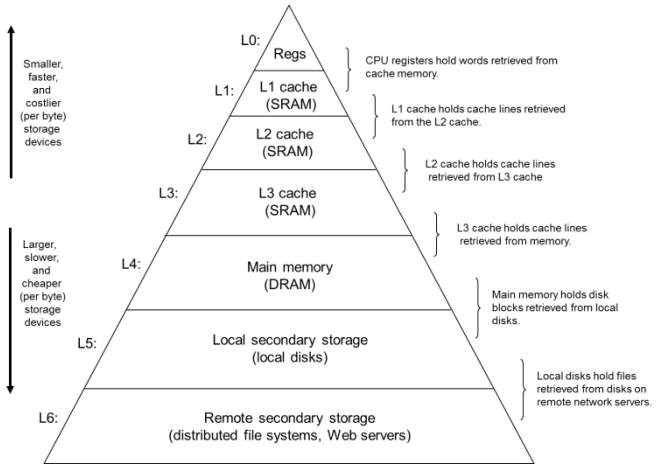
- **Use Abstraction to Simplify Design**
  - abstractions are everywhere

- **Make the Common Case Fast**
  - **Amdahl's Law**

$$speedup = \frac{1}{(1 - parallel) + \frac{parallel}{\#cores}}$$

Great Ideas in Computer Architecture

- **Hierarchy of Memories**
  - exploit spatial and temporal locality (공간/시간 구역성)
- **Performance via Parallelism**
- **Performance via Pipelining**
- **Performance via Prediction**



That's It For Today

- Next class: the ISA and assembly basics

- Late enrollment: students who could not sign up for this class through the online registration system can submit their 초안지 now