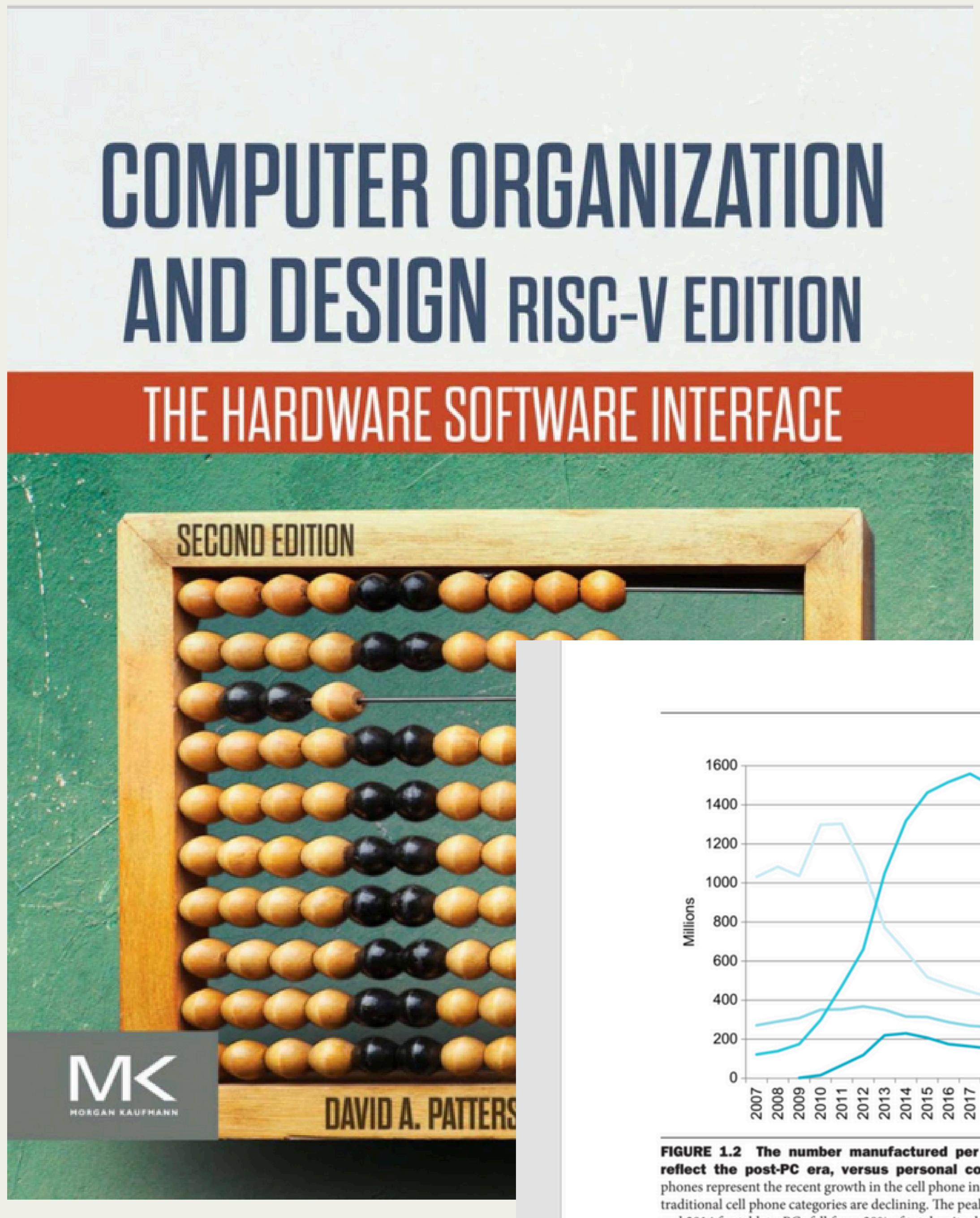
Final Report

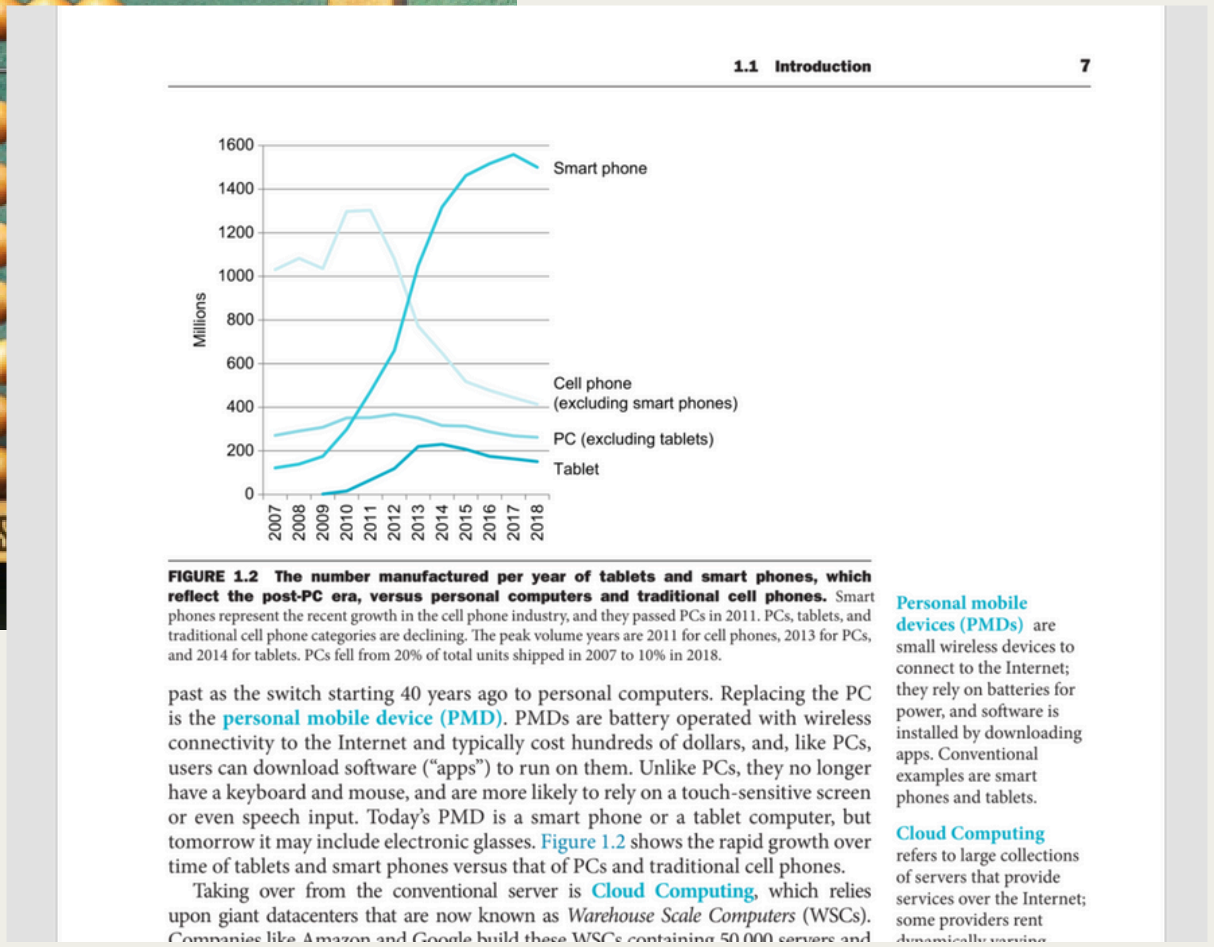
COMPUTER STRUCTURE

資工二
111210552
林小蓮

Computer Organization and Design: RISC-V Edition

by David A. Patterson and John L. Hennessy


COMPUTER ORGANIZATION AND DESIGN RISC-V EDITION
THE HARDWARE SOFTWARE INTERFACE
SECOND EDITION
MK
MORGAN KAUFMANN
DAVID A. PATTERS



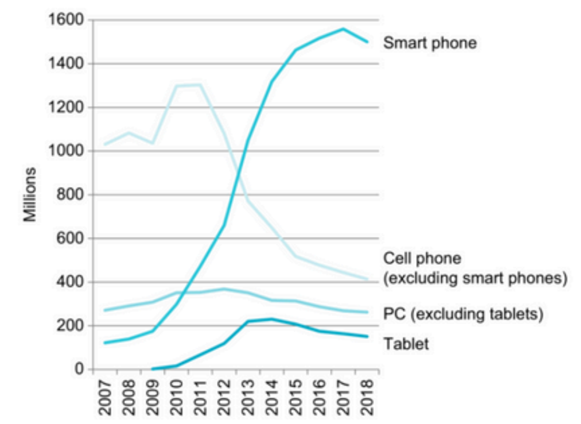


FIGURE 1.2 The number manufactured per year of tablets and smart phones, which reflect the post-PC era, versus personal computers and traditional cell phones. Smart phones represent the recent growth in the cell phone industry, and they passed PCs in 2011. PCs, tablets, and traditional cell phone categories are declining. The peak volume years are 2011 for cell phones, 2013 for PCs, and 2014 for tablets. PCs fell from 20% of total units shipped in 2007 to 10% in 2018.

past as the switch starting 40 years ago to personal computers. Replacing the PC is the **personal mobile device (PMD)**. PMDs are battery operated with wireless connectivity to the Internet and typically cost hundreds of dollars, and, like PCs, users can download software ("apps") to run on them. Unlike PCs, they no longer have a keyboard and mouse, and are more likely to rely on a touch-sensitive screen or even speech input. Today's PMD is a smart phone or a tablet computer, but tomorrow it may include electronic glasses. Figure 1.2 shows the rapid growth over time of tablets and smart phones versus that of PCs and traditional cell phones.

Taking over from the conventional server is **Cloud Computing**, which relies upon giant datacenters that are now known as *Warehouse Scale Computers* (WSCs). Companies like Amazon and Google build these WSCs containing 50,000 servers and

Personal mobile devices (PMDs) are small wireless devices to connect to the Internet; they rely on batteries for power, and software is installed by downloading apps. Conventional examples are smart phones and tablets.

Cloud Computing refers to large collections of servers that provide services over the Internet; some providers rent

CONTENT

BRIEF SUMMARY

Introduction to Computer Organization

- Focuses on the fundamental building blocks of computers.
- Explains the interaction between hardware and software through the RISC-V ISA.
- Introduces the idea of abstraction layers: how high-level software translates into low-level hardware instructions.

Instruction Set Architecture (ISA)

RISC-V ISA:
- A simple, open-source instruction set for teaching and innovation.
- Focused on reduced instruction sets for efficiency.

Importance of ISAs:
- Acts as the bridge between hardware and software.
- Determines how programs interact with the processor.

Processor Design

Key Concepts:
- *Datapath:* The component responsible for executing instructions.
- *Control Unit:* Directs operations within the processor.

Single-Cycle vs. Multi-Cycle Processors:
- *Single-cycle:* Executes one instruction in one clock cycle.
- *Multi-cycle:* Breaks instructions into smaller steps for efficiency.

Pipelining

Definition: Breaking down instruction execution into multiple stages to improve performance.
Stages of Pipelining:
1. Fetch: Get the instruction from memory.
2. Decode: Understand what the instruction does.
3. Execute: Perform the required operation.
4. Memory Access: Read/write data from/to memory.
5. Write Back: Save the result.

Challenges:
- Data Hazards: Dependencies between instructions.
- Control Hazards: Branching (e.g., if/else) causing pipeline delays.

BRIEF SUMMARY

Memory Hierarchy

Why Memory Matters: Memory is slower than the processor, so hierarchy improves speed.

Components:

- Registers: Fastest but smallest memory.
- Caches: Store frequently used data close to the CPU.
- Main Memory (RAM): Larger but slower.
- Storage (HDD/SSD): Long-term, even slower.

Virtual Memory: Allows programs to use more memory than physically available by using disk storage as overflow.

Parallelism

Instruction-Level Parallelism (ILP): Overlapping instructions within the pipeline to improve performance.

Hardware-Level Parallelism:

- Multi-core processors: Multiple CPUs working together.
- GPUs: Specialized processors for parallel tasks like graphics and AI.

Importance in modern systems: Essential for tasks like gaming, AI, and large-scale data processing.

Floating-Point Arithmetic

Floating-Point Numbers: Used to represent very large or very small numbers.

Applications: Scientific computations, graphics, and machine learning.

Challenges:

- Requires special hardware (Floating Point Unit - FPU).
- Precision errors can occur in calculations.

RISC-V's Impact

Why RISC-V?

- Open-source and free to use.
- Modular: Developers can customize it for different applications.
- Widely adopted in IoT, AI, and embedded systems.

Comparison with Other ISAs: Simpler and more flexible than x86 and ARM.

Applications of Concepts

- Optimizing software by understanding hardware limitations.
- Designing efficient systems for specific use cases like AI and IoT.
- Laying the groundwork for careers in hardware and software engineering.

GENERAL IMPRESSION

This book gave me a much deeper understanding of how computers actually work. Before this, I had some basic knowledge, but this material was way more detailed than I was used to. It was cool to see how everything—from simple instructions to full processor designs—fits together. The focus on RISC-V was refreshing since it's modern and open-source, but I'll admit, some parts of the book, like pipelining and memory systems, were pretty tough to understand at first.

WHAT I LEARNED (KEY CONCEPTS)

1. RISC-V and Instruction Sets

I knew computers ran on instructions, but I didn't realize how organized and structured these systems are. RISC-V stood out because it's simple and easy to follow compared to more complicated architectures like x86. It made learning this stuff less intimidating.

2. Processor Design and Pipelining

Pipelining was one of the most interesting concepts I learned. Breaking tasks into smaller steps so the processor can do multiple things at once is really smart. It also made me realize how hard it is to handle problems like data hazards.

WHAT I LEARNED (KEY CONCEPTS)

3. Memory Systems

The chapter on memory systems taught me how important things like caches and virtual memory are for speed and efficiency. I'd never thought about how much effort goes into managing data so that programs don't slow down.

4. Parallelism

I always knew multi-core processors and GPUs were important, but this book helped me understand why. It's crazy how much effort goes into designing systems that can handle multiple tasks at the same time.

OPINIONS ON RISC-V

I really liked how the book focused on RISC-V. It’s simple, easy to learn, and feels like a good starting point for beginners. Knowing it’s open-source made it even more interesting because it’s something anyone can use and improve. I can see why it’s becoming so popular in new technologies like IoT and AI.

APPLICATIONS TO REAL LIFE

This book helped me connect what I learned to the real world. For example:

Understanding how processors handle instructions made me think about how programs are actually run on my laptop or phone.

Learning about memory systems helped me see why upgrading hardware, like adding more RAM or a better CPU, makes such a big difference.

It made me realize that even if I focus on software in the future, understanding hardware is really useful.

CRITIQUES

***The book was super informative**, but some parts felt really dense. For example:*

The chapters on pipelining and floating-point arithmetic were hard to follow the first time. More examples or simpler explanations would’ve helped a lot.

Some parts about Verilog and hardware design felt a bit advanced for students who haven’t worked with that before.

Having more visuals or interactive tools would’ve made it easier to understand the more complicated concepts.

PERSONAL REFLECTION (心得)

This book opened my eyes to how computers really work, beyond just running programs. It was challenging, but it felt rewarding when I started to understand the big picture. It also gave me a new appreciation for how much work goes into designing and optimizing computer systems.

What stuck with me the most is how important simplicity is in design. RISC-V showed me that even simple ideas can be powerful when they’re done right. Overall, this book has inspired me to keep learning about computer systems and how hardware and software work together. It’s definitely made me more curious about how technology shapes the world.

Thank you!