CS211
Advanced Computer Architecture

L01 Introduction

Chundong Wang

September 17th, 2025

Welcome to CS211

- Advanced topics related to computer architecture
  - **Read the syllabus in the system for details.**
  - Prerequisite
    - CS110 (Computer Architecture I) of ShanghaiTech University, or equivalents
  - Reference Textbook
    - 计算机体系结构：量化研究方法（英文版·原书第6版）. 约翰·L. 亨尼斯（John L. Hennessy），戴维·A. 帕特森 （David A. Patterson）、机械工业出版社、2019年07月
- Instructor
  - Chundong Wang (王春东)
  - Office: SIST 1A-504.D
  - Email: wangchd → wangc
  - Website: https://Chundong.Wang
  - Office hour
    - 9:00am-11:00am, every Monday in the Fall semester of 2025
- Teaching assistant (TA) of Fall 2025
  - Mr. Kunchang Guo (郭坤昌)
  - Email: guokch2024
- Online
  - https://toast-lab.sist.shanghaitech.edu.cn/courses/CS211@ShanghaiTech/Fall-2025/
  - Gradescope, Piazza, etc.

Why CS211?

- To become a computer architect
  - Alumni of this class helped design your computer
  - *To make the frequent cases fast and the rare case correct*
  - *To make the most out of resources*
  - … …
- To learn what is *under the hood* of a computer
  - Innate curiosity
  - To better understand when things break
  - To write better code/applications
  - To write better system software (OS, compiler, DB, etc.)
- Because it is intellectually fascinating!
  - What is the most complex man-made single device?

Basic Rules

- No Abuse
  - Abusive words or behaviours are NOT allowed.
  - Be **graceful** and **respectful**.
- No Plagiarism
  - Do not copy **ANYTHING** from **ANYWHERE** at **ANYTIME**.
  - Be yourself.
- No Absence
  - Any one that is absent for three or more quizzes may **fail** CS211 in Fall 2025

Policy on Assignments and Independent Work

上海科技大学 ShanghaiTech University

- With the exception of laboratories and assignments that explicitly permit you to work in groups, all homework and projects are to be YOUR work and your work ALONE.
- PARTNER TEAMS MAY NOT WORK WITH OTHER PARTNER TEAMS
- You can discuss your assignments with other students, and credit will be assigned to students who help others by answering questions (participation), but we expect that what you hand in is yours.
- Level of detail allowed to discuss with other students: Concepts (Material taught in the class/ in the text book)! Pseudocode is NOT allowed!
- Use the Office Hours of the TA and the Prof. if you need help with your homework/ project!
- Rather submit an incomplete homework with maybe 0 points than risking an F!
- It is NOT acceptable to copy solutions from other students.
- You can never look at homework/ project code not by you/your team!
- You cannot give your code to anybody else –> secure your computer when not around it
- It is NOT acceptable to copy (or start your) solutions from the Web.
- It is NOT acceptable to use PUBLIC github/gitlab/gitee archives (giving your answers away)
- We have tools and methods, developed over many years, for detecting this. You WILL be caught, and the penalties WILL be severe.
- **At the minimum F in the course**, and a letter to your university record documenting the incidence of cheating.
- **Both Giver and Receiver are equally culpable and suffer equal penalties**

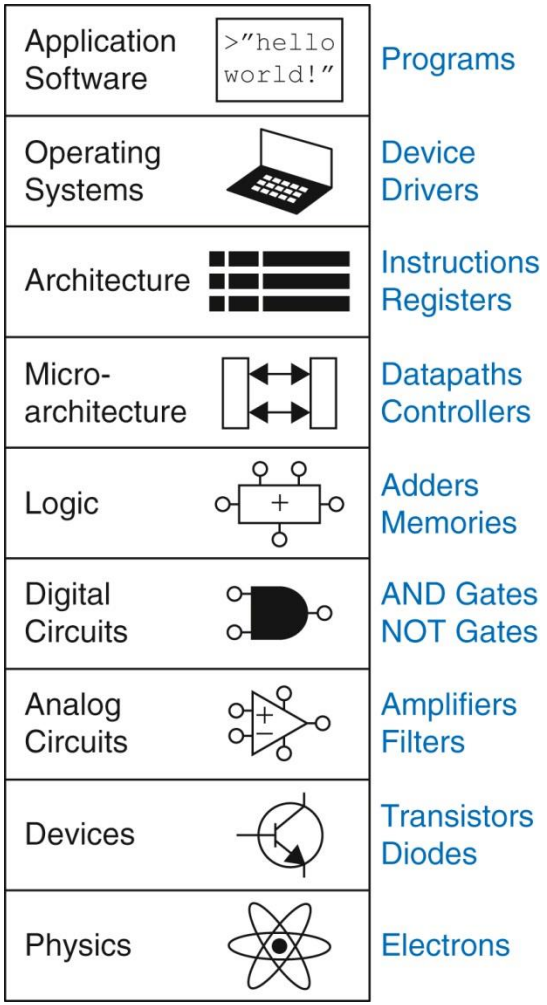
Entry-level Test

- Time
  - 10:20am – 12:00pm (100 minutes), 19 September, 2024
  - Be present before 10:00am
- Venue
  - TBD (tentative)
  - A seating table would be posted
- Contents
  - Everything related to Computer Architecture might be covered
- Closed-book
  - You can take up to three A4-sized cheat sheets, fully handwritten, non-sharing
  - No electronic device is allowed, including but not limited to cell phone, laptop computer, tablet computer, calculator, smart watch, etc.
  - Besides your own pens, pencils, erasers, etc., you can bring a bottle of water or soft drink and snacks/cookies/buns/…

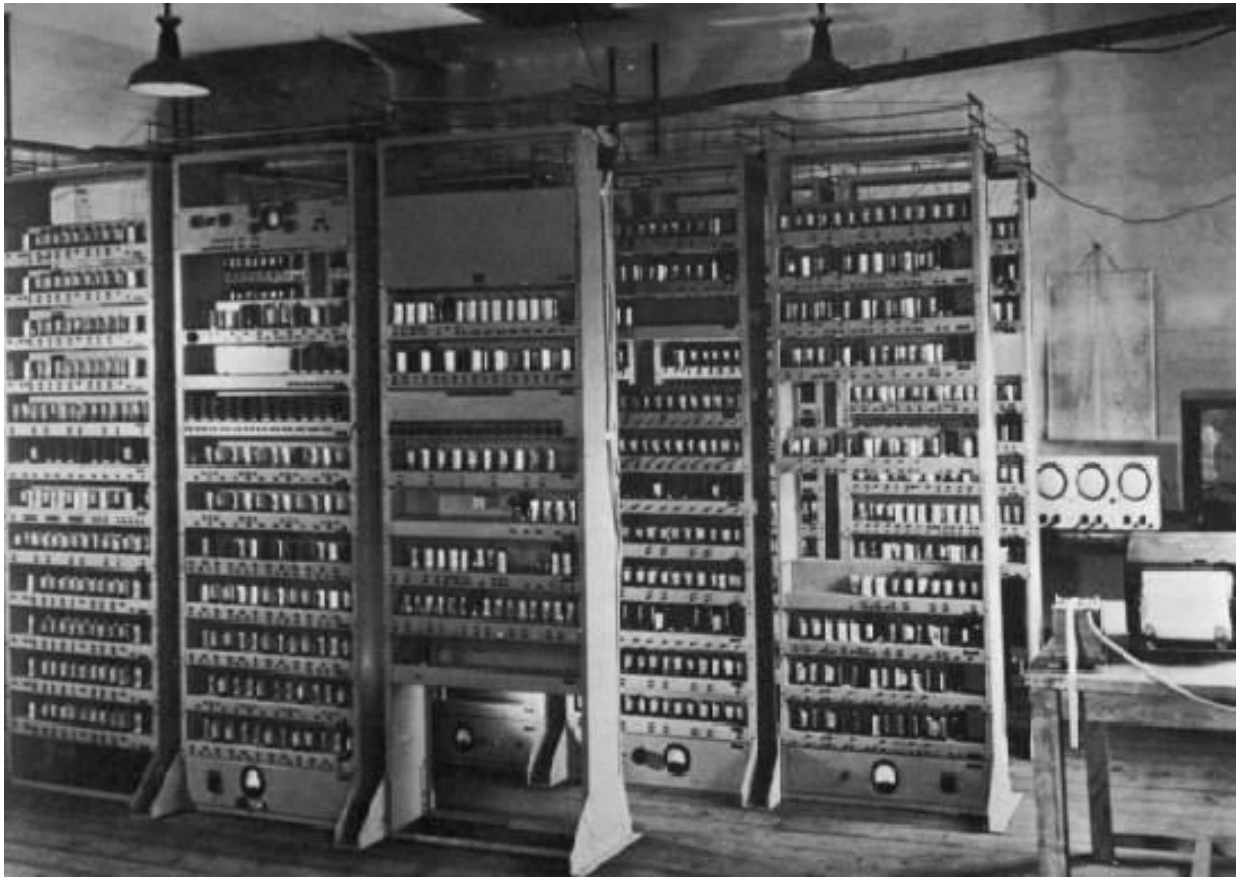
Now, let’s dive into CA.

Computer Architecture

- Application, e.g., WeChat, Meituan
- Algorithm, e.g., KMP, Quicksort
- Programming Language, e.g., C/C++/Python
- Operating System/Virtual Machines
- Instruction Set Architecture (ISA)
- Microarchitecture
- Gates/Register-Transfer Level (RTL)
- Circuits
- Devices
- Physics

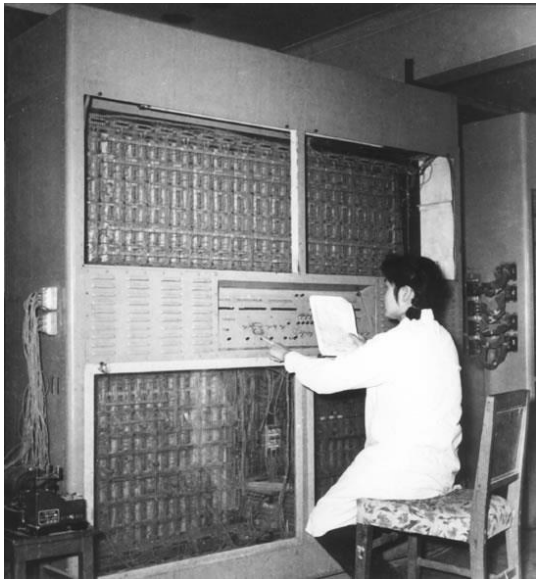


Computing Devices Long Long ago

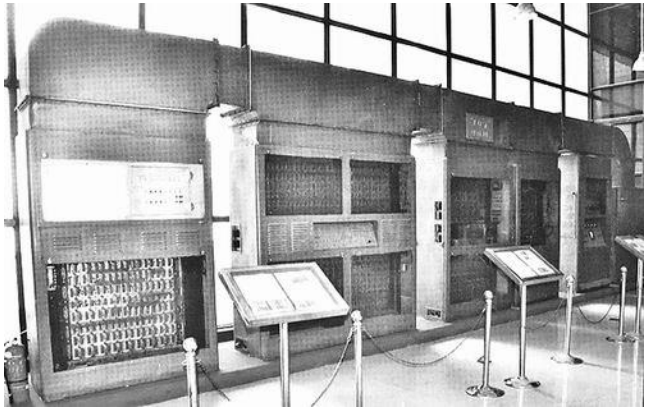


EDSAC (Electronic delay storage automatic calculator), University of Cambridge, UK, May 1949

The 1st computer of China



http://www.ict.cas.cn/kxcb/kxtp/200909/t20090922_2514368.html

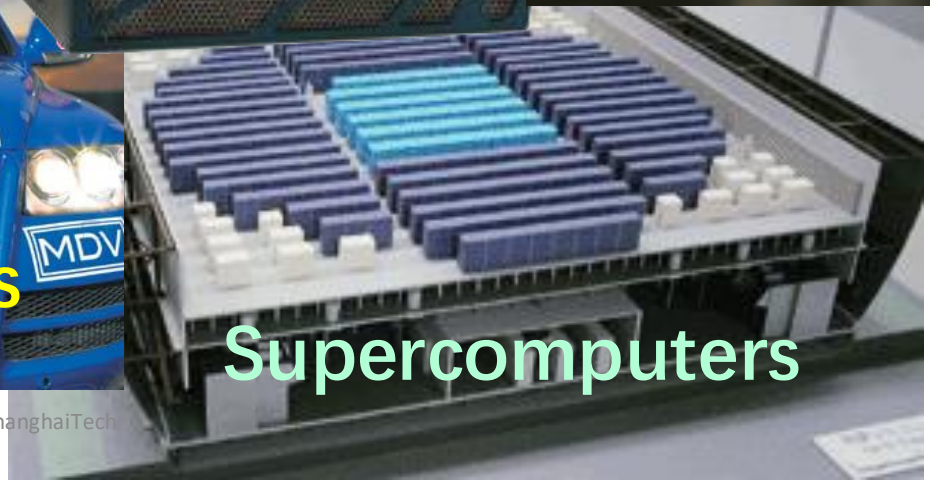


http://news.sciencenet.cn/sbhtmlnews/2019/9/349501.shtm

Computer 103 (103机)

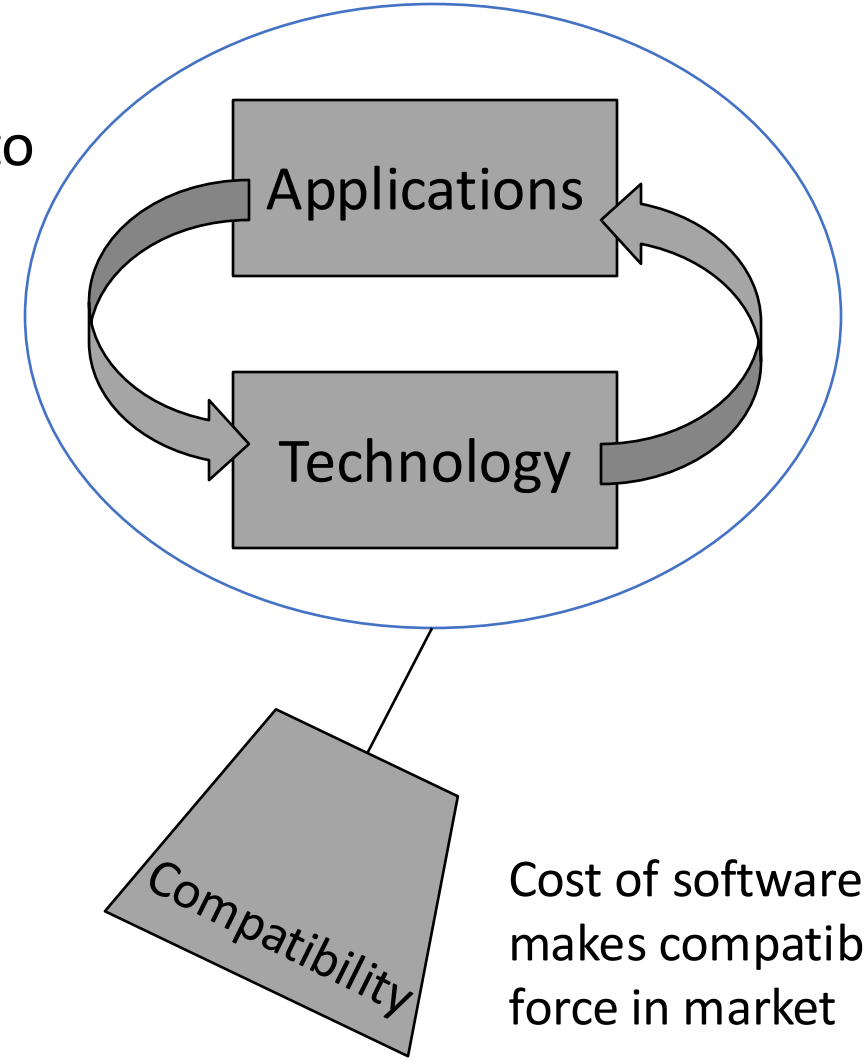
- Kick-off: On August 1st 1958, four instructions were executed on Computer 103 which was with 1KB memory and 30 instructions per second.
- Products: In all, 41 machines were manufactured.
- The remaining intact one: Fudan University (复旦大学) purchased it in 1964. In 1973, Fudan donated it to Qufu Normal University (曲阜师范大学). It was delivered using two trucks with three instructors accompanied. Qufu Normal University built a two-story building for it.

Computing Devices Now



Architecture continually changing

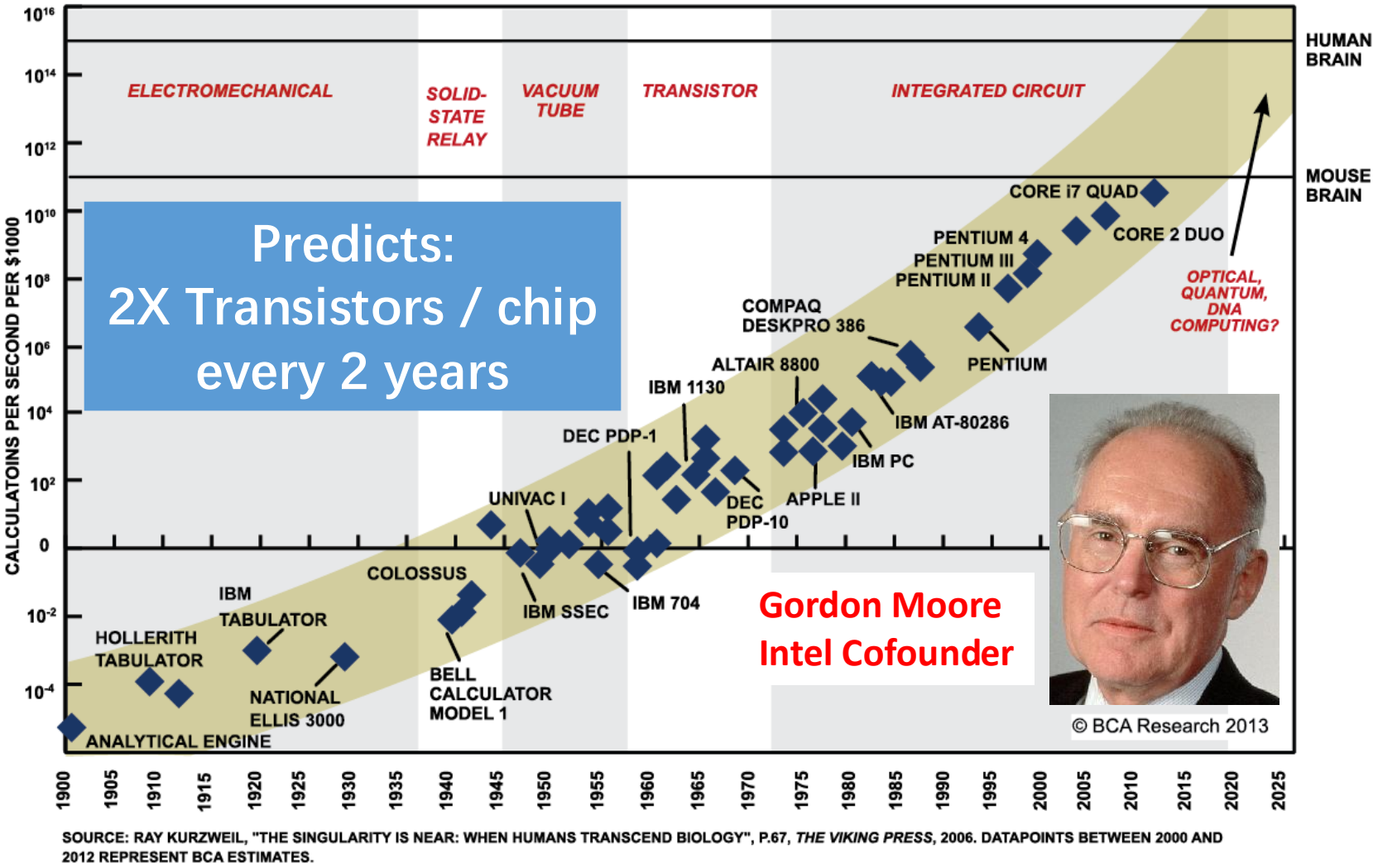
Applications suggest how to improve technology, provide revenue to fund development



Improved technologies make new applications possible

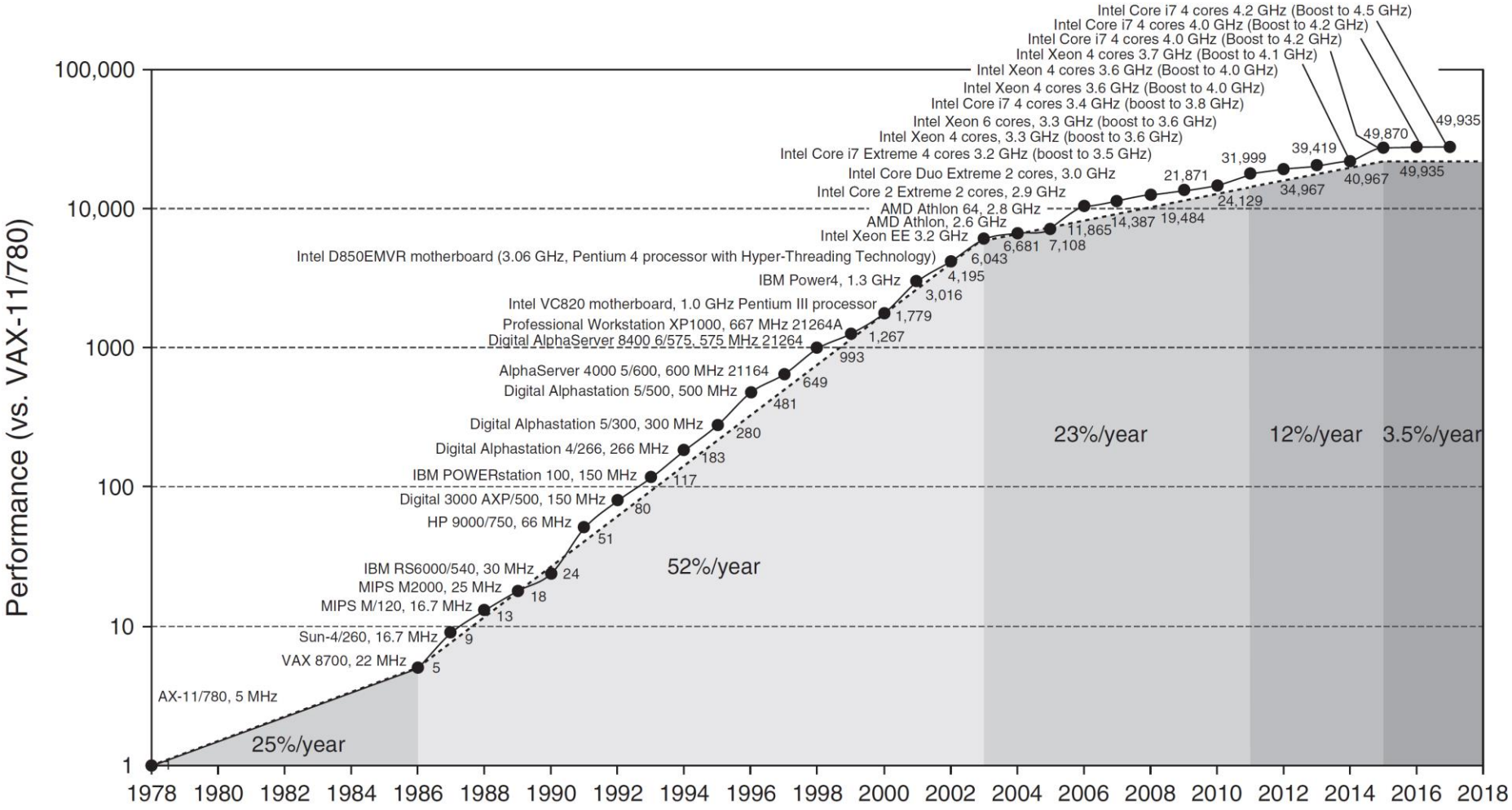
Cost of software development makes compatibility a major force in market

Moore's Law



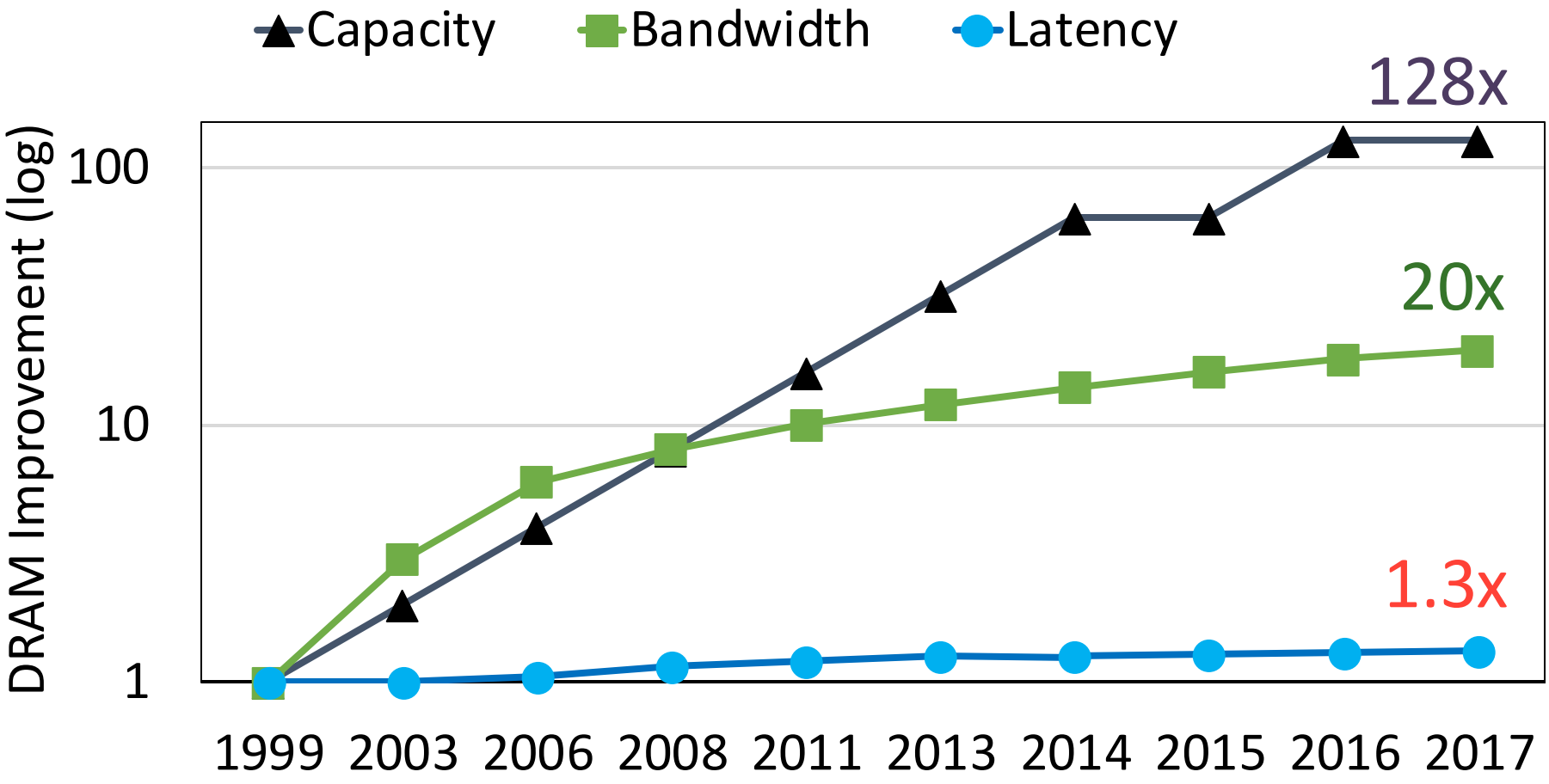
"The number of transistors incorporated in a chip will approximately double every 24 months."—Gordon Moore, Intel co-founder

Single-Thread Processor Performance



Growth in processor performance over 40 years **[Hennessy & Patterson, 2019]**

DRAM Capacity, Bandwidth, Latency Trends



Upheaval in Computer Design

- Most of last 50 years, Moore's Law ruled
  - Technology scaling allowed continual performance/energy improvements without changing software model
- Last decade, technology scaling slowed/stopped
  - Dennard (voltage) scaling over (supply voltage ~fixed)
  - Moore's Law (cost/transistor) over?
  - No competitive replacement for CMOS anytime soon
  - Energy efficiency constrains everything
- No "free lunch" for software developers, must consider:
  - Parallel systems
    - e.g., AMD released their first dual-core processor, the Athlon 64 X2 3800+ (2.0 GHz, 512 KB L2 cache per core), on April 21, 2005.
  - Heterogeneous systems
    - e.g., ARM's big.LITTLE
    - "LITTLE" processors (e.g., Cortex-A53) are designed for maximum power efficiency while "big" processors (e.g., Cortex-A73) are designed to provide maximum compute performance.

Today's Dominant Target Systems

- Mobile (smartphone/tablet)
  - >1 billion sold/year
  - Market dominated by ARM-ISA-compatible general-purpose processor in system-on-a-chip (SoC)
  - Plus sea of custom accelerators (radio, image, video, graphics, audio, motion, location, security, etc.)
- Warehouse-Scale Computers (WSCs)
  - 100,000's cores per warehouse
  - Market dominated by x86-compatible server chips
  - Dedicated apps, plus cloud hosting of virtual machines
  - Now seeing increasing use of GPUs, FPGAs, custom hardware to accelerate workloads
- Embedded computing
  - Wired/wireless network infrastructure, printers
  - Consumer TV/Music/Games/Automotive/Camera/MP3
  - Internet of Things!

A toy example to review CA

```c
// This is a toy example for CS211@ShanghaiTech.
// -- wangchd@shanghaitech.edu.cn

#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include <ctype.h>

#define ARRAY_LENGTH 10

int main(int argc, char **argv) {
        int key_array[ARRAY_LENGTH] = {0, 10, 20, 30, 40, 50, 60, 70, 80, 90};
        int key = 0, i = 0;

        /* Assert that input is truly a number */
        assert(argc == 2);
        for (i = 0; argv[1][i] != 0; ++i)
                assert(isdigit(argv[1][i]) != 0); // Is every char a digit?

        /* Convert the string to a number */
        key = atoi(argv[1]);

        /* Linearly scan the array */
        for (i = 0; i < ARRAY_LENGTH; ++i)
                if (key == key_array[i])
                        break;

        /* Output: if the input is found or not */
        if (i == ARRAY_LENGTH)
                fprintf(stdout, "Key %d is NOT found\n", key);
        else
                fprintf(stdout, "Key %d is found at postion %d\n", key, i);

        return 0;
}

```

What are not covered by CA?

Programming, and language design

Compiling

Process scheduling

Data structure and Algorithm

File operations

```c
// This is a toy example for CS211@ShanghaiTech.
                                    .edu.cn

#include <ctype.h>

#define ARRAY_LENGTH 10

int main(int argc, char **argv) {
        int key_array[ARRAY_LENGTH] = {0, 10, 20, 30, 40, 50, 60, 70, 80, 90};
        int key = 0, i = 0;

        /* Assert that input is truly a number */
        assert(argc == 2);
        for (i = 0; argv[1][i] != 0; ++i)
                assert(isdigit(argv[1][i]) != 0); // Is every char a digit?

        /* Convert the string to a number */
        key = atoi(argv[1]);

        /* Linearly scan the array */
        for (i = 0; i < ARRAY_LENGTH; ++i)
                if (key == key_array[i])
                        break;

        /* Output: if the input is found or not */
        if (i == ARRAY_LENGTH)
                fprintf(stdout, "Key %d is NOT found\n", key);
        else
                fprintf(stdout, "Key %d is found at postion %d\n", key, i);

        return 0;
}
```

What are covered by CA?

```
1  // This is a toy example for CS211@ShanghaiTech.
2  // -- wangchd@shanghaitech.edu.cn
3
10
11 int main(int argc, char **argv) {
12         int key_array[ARRAY_LENGTH] = {0, 10, 20, 30, 40, 50, 60, 70, 80, 90};
13         int key = 0, i = 0;
14
15         /* Assert that input is truly a number *
16         assert(argc == 2);
17                 [i] != 0; ++i)
                   it(argv[1][i]) != 0)
                   g to a number */
23         /* Linearly scan the array */
24         for (i = 0; i < ARRAY_LENGTH; ++i)
25                 if (key == key_array[i])
26                         break;
27
28         /* Output: if the input is found or not */
29         if (i == ARRAY_LENGTH)
                   "Key %d is N
                   "Key %d is f
36
```

Instruction execution: pipeline, in-order or out-of-order, speculation, etc.

Instructions and micro-codes

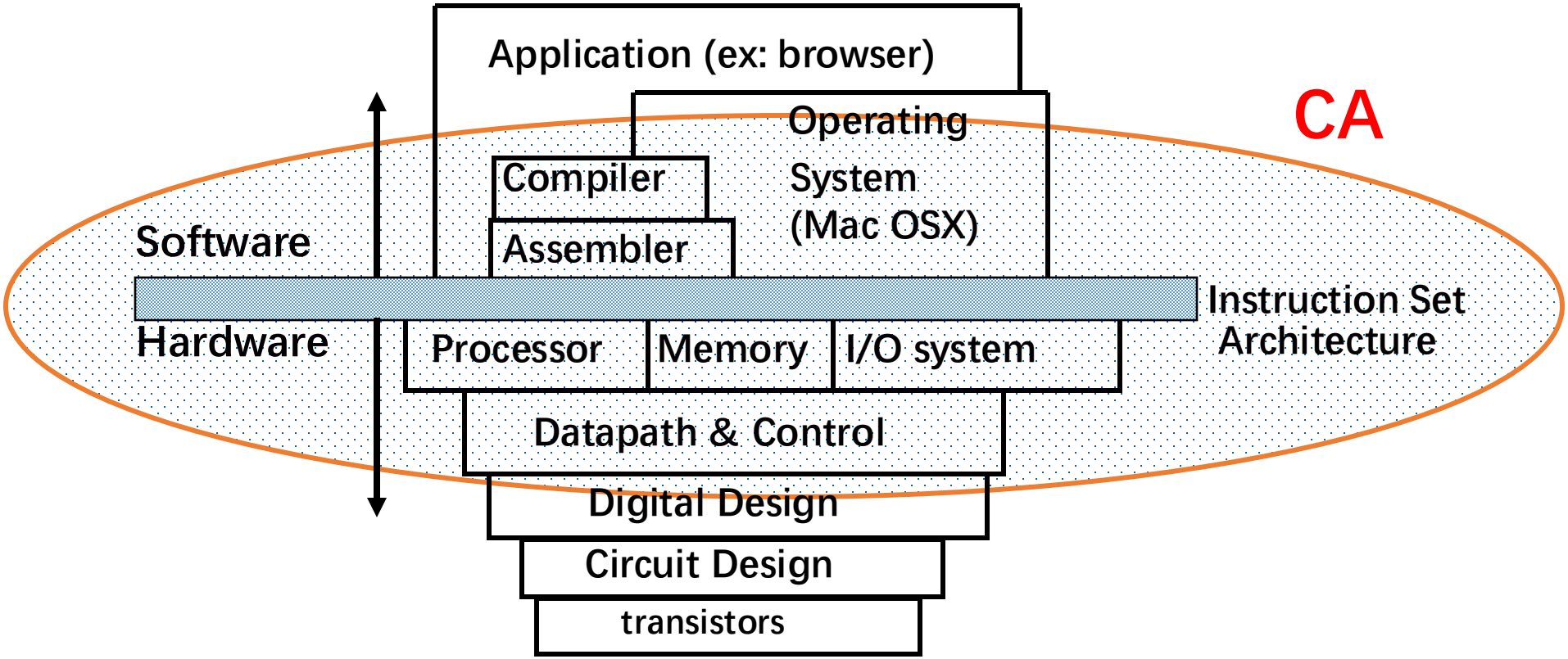
Memory hierarchy: cache, main memory, disk, etc.

Exception, interrupt, etc.

Single-threaded or multi-threaded

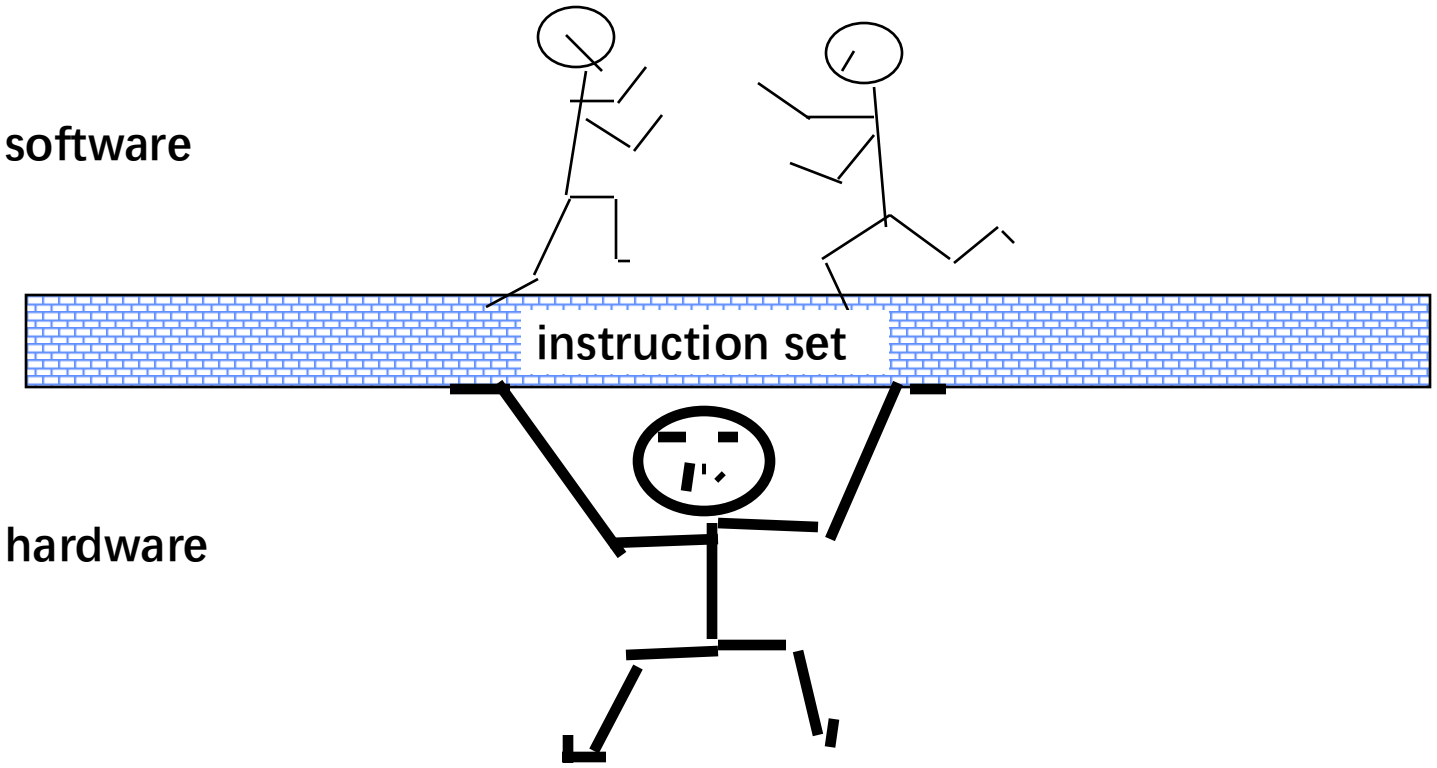
I/O

Conventional Machine Structure



ISA: instruction set architecture

An instruction is a single operation, with an opcode and zero or more operands, of a processor, defined by the processor instruction set.



ISA is the actual programmer-visible instruction set, a critical interface/boundary/contract between software and hardware

An example, `add` of RISC-V

- a
- r
- [
- N
- 3

```
// This is a toy example for CS211@ShanghaiTech.
// -- wangchd@shanghaitech.edu.cn

#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include <ctype.h>

#define ARRAY_LENGTH 10

int main(int argc, char **argv) {
        int key_array[ARRAY_LENGTH] = {0, 10, 20, 30, 40, 50,
        int key = 0, i = 0;

        /* Assert that input is truly a number */
        assert(argc == 2);
        for (i = 0; argv[1][i] != 0; ++i)
                assert(isdigit(argv[1][i]) != 0); // Is every char a digit?

        /* Convert the string to a number */
        key = atoi(argv[1]);

        /* Linearly scan the array */
        for (i = 0; i < ARRAY_LENGTH; ++i)
                if (key == key_array[i]
                        break;

        /* Output: if the input is foun
        if (i == ARRAY_LENGTH)
                fprintf(stdout, "Key %d is NOT found\n", key);
        else
                fprintf(stdout, "Key %d is found at postion %d\n", key, i);

        return 0;
}
```

Data transfer

Arithmetic

Control

31 … 0

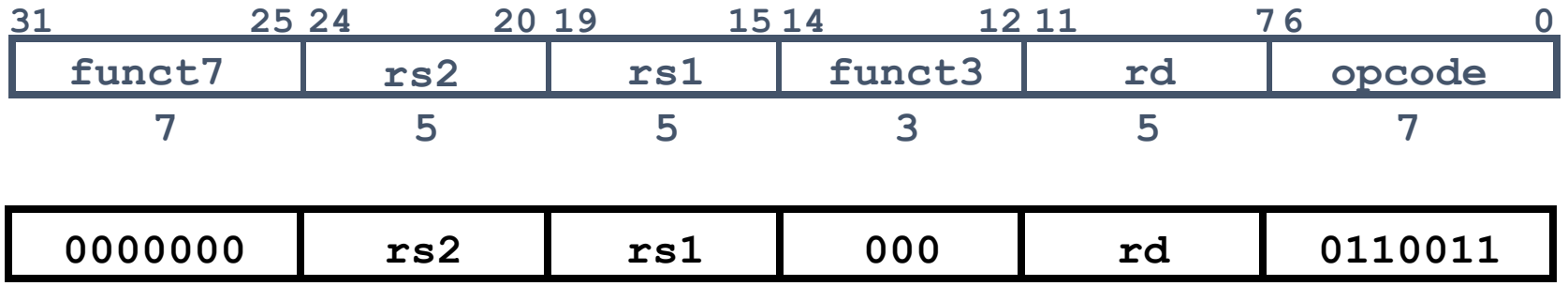
func … ode

7 …

0000(… 011

add … Reg OP

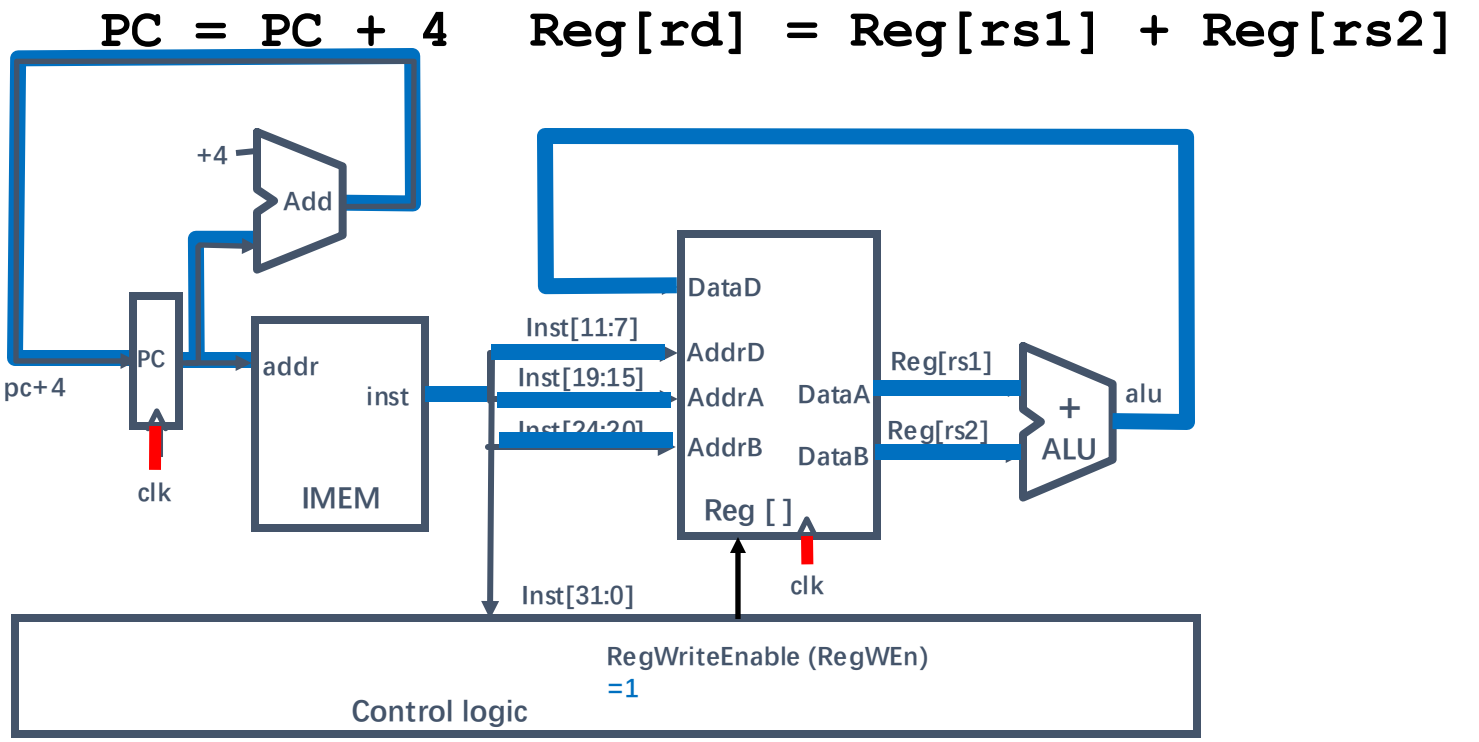
Implementing the **add** instruction of RISC-V

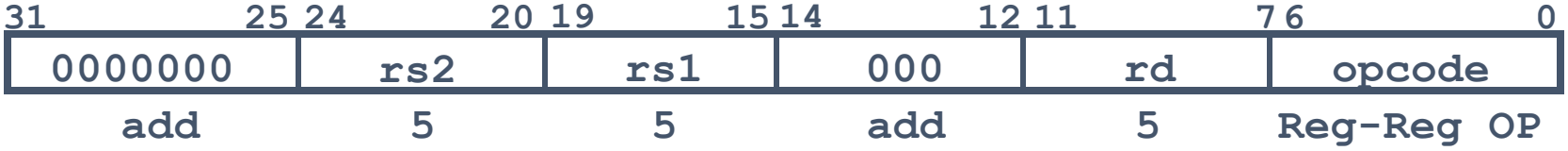


add rd, rs1, rs2

- Instruction makes two changes to machine's state:
  - **Reg[rd] = Reg[rs1] + Reg[rs2]**
  - **PC = PC + 4** %%% To get next instruction

Datapath for `add`

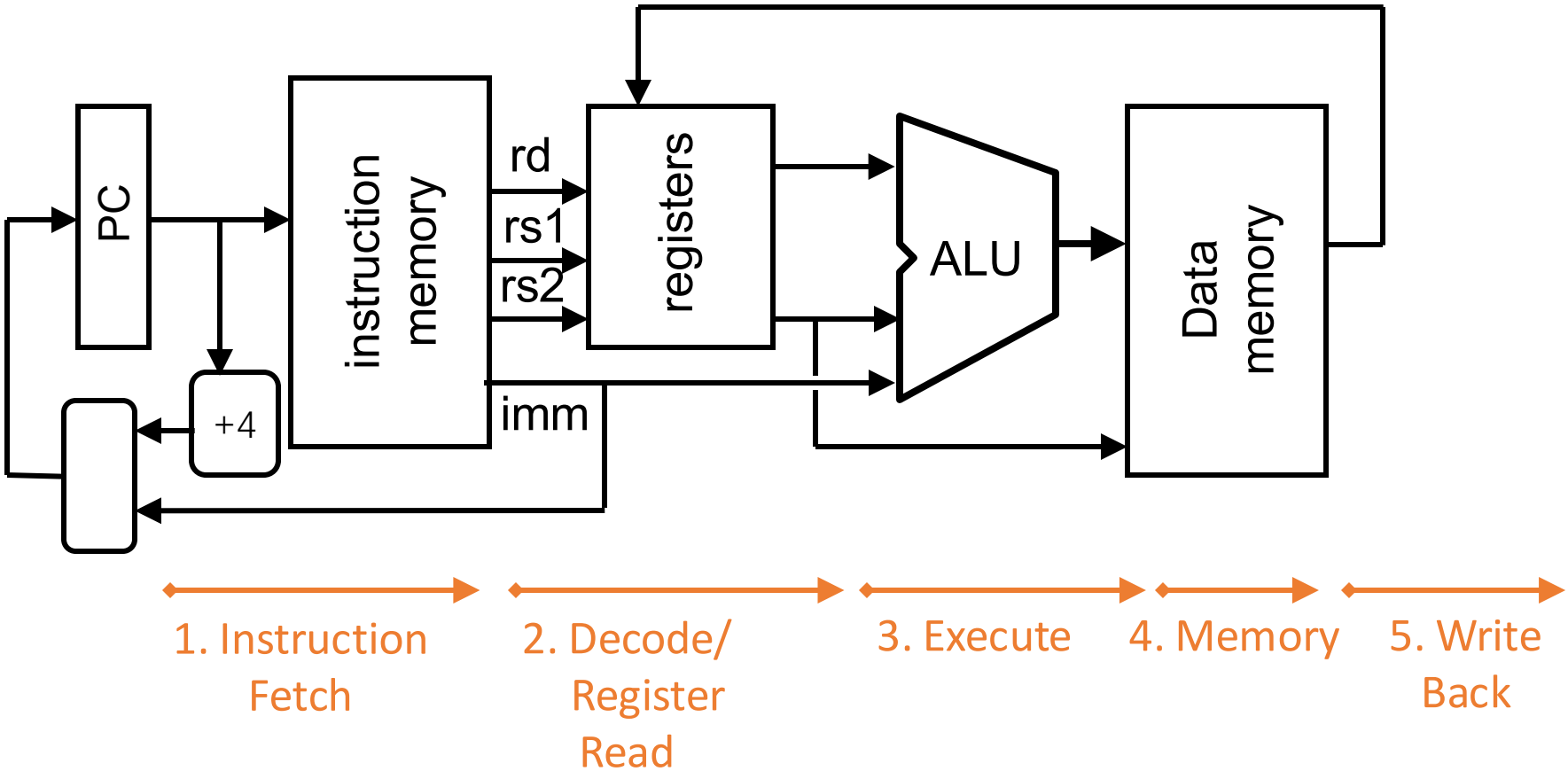




From architecture to microarchitecture

- Instructions are visible to programmers
- Two processors may have the same ISA but different microarchitectures
  - e.g., AMD Opteron and Intel Core i7, with the same 80x86 ISA, have very different pipelines and cache organizations.
- An instruction is partitioned into multiple stages
- Five classic stages of executing an instruction
  - Instruction fetch
  - Instruction decode/register fetch
  - Execute
  - Memory access
  - Write back

Stages of Execution on Datapath



Instruction execution one by one

Inst. No.	1	2	3	4	5	6	7	8	9	10	11	12	13
i (add)	F	D	X	M	W								
i+1						F	D	X					
i+2									F	D	X	M	
i+3													F
i+4													

Low utilization of hardware, and long execution time

Pipeline: instruction-level parallelism

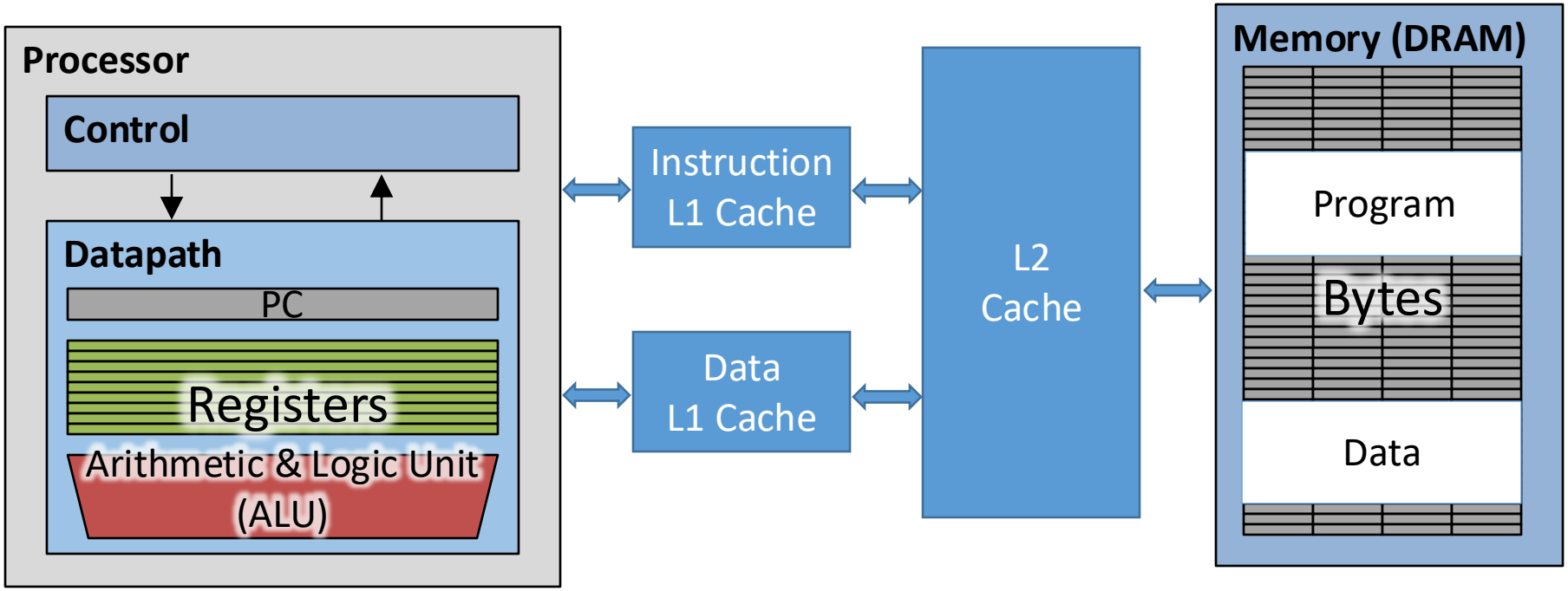
Hardzards may happen

Inst. No.	1	2	3	4	5	6	7	8	9	10	11	12	13
i	F	D	X	M	W								
i+1		F	D	X									
i+2			F	D	X	M							
i+3				F	D	X	M	W					
i+4					F	D	X	M	W				

From single-core to multi-core

- Multi-core is not multi-threading
  - Single core supports multi-threading. Multi-threading is older than multi-core.
  - Intel introduced “hyper-threading” in 2002
    - Virtually, one core becomes two.
- The era of multi-core
  - Intel with Core 2 Duo, AMD with Athlon 64 X2 in 2005/2006.
  - *From Single Core to Multi-Core: Preparing for a new exponential*, in ICCAD ’06
- Multi-core
  - Replicate multiple cores on a single die.
  - Operating systems perceives a core as a separate processor.
- Why multi-core?
  - Difficult to make single-core clock frequencies even higher, i.e., the wall
  - Multi-threading applications, more parallelism demanded
- Problems along with multi-cores?
  - Cache coherence, scheduling, interconnect, etc.

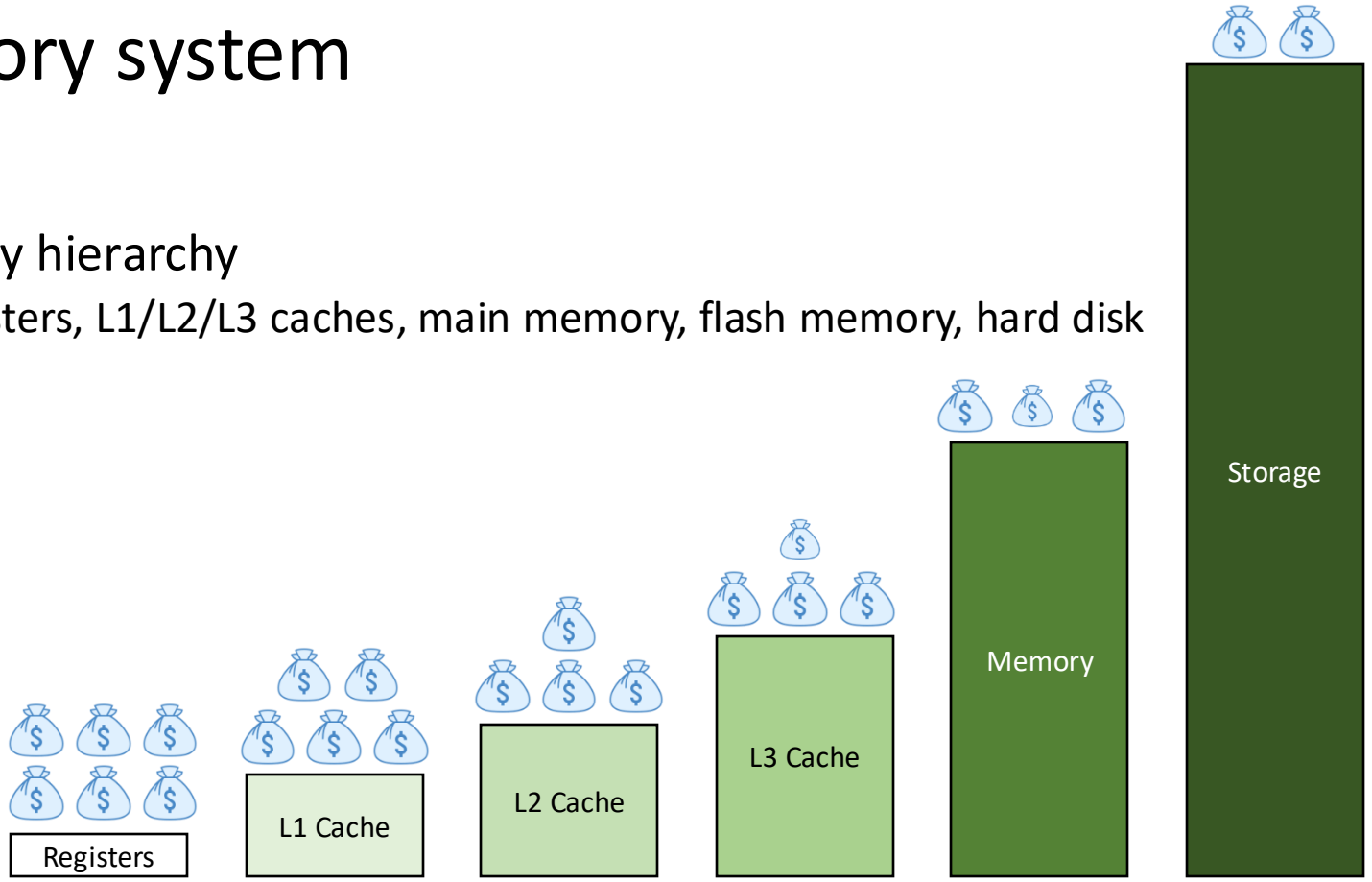
Multi-level cache and main memory



Now L3 cache is very common.
How to manage multi-level caches would be detailed later in this course.

Memory system

- Memory hierarchy
  - Registers, L1/L2/L3 caches, main memory, flash memory, hard disk



	Registers	L1 Cache	L2 Cache	L3 Cache	Memory	Storage
Size	2000 bytes	64KB	256KB	8-32MB	8-64GB	256GB-2TB
Speed	300ps	1ns	3-10ns	10-20ns	50-100ns	50-100us

A typical memory hierarchy for a desktop

Memory system

- Mer
  - R
- Loca
  - S
  - S
  - T

```c
// This is a toy example for CS211@ShanghaiTech.
// -- wangchd@shanghaitech.edu.cn

#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include <ctype.h>

#define ARRAY_LENGTH 10

int main(int argc, char **argv) {
        int key_array[ARRAY_LENGTH] = {0, 10, 20, 30, 40, 50, 60, 70, 80, 90};
        int key = 0, i = 0;

        /* Assert that input is truly a number */
        assert(argc == 2);
        for (i = 0; argv[1][i] != 0; ++i)
                assert(isdigit(argv[1][i]) != 0); // Is every char a digit?

        /* Convert the string to a number */
        key = atoi(argv[1]);

        /* Linearly scan the array */
        for (i = 0; i < ARRAY_LENGTH; ++i)
                if (key == key_array[i])
                        break;

        /* Output: if the input is found or not */
        if (i == ARRAY_LENGTH)
                fprintf(stdout, "Key %d is NOT found\n", key);
        else
                fprintf(stdout, "Key %d is found at postion %d\n", key, i);

        return 0;
}
```

Spatial locality

Temporal locality

Topics covered by CS211

- *Microcode, instructions, ISA*
- *Pipeline*
- *Memory hierarchy*
  - *CPU cache/main memory/disk*
- In-order execution and out-of-order execution
- Speculative execution
  - Branch prediction
- VLIW
  - Very long instruction word
- Multi-threading
- Vectors
- Cache coherence
- Memory consistency
- Synchronization
- Virtual machines
- Security and Privacy
- etc.

Conclusion

- CA is interesting
- CA is challenging
- CA is evolving

Acknowledgements

- These slides contain materials developed and copyright by:
  - Prof. Krste Asanovic (UC Berkeley)
  - Prof. Xuehai Zhou (USTC)
  - Prof. Onur Mutlu (ETH Zurich)
  - Prof. Mikko Lipasti (UW-Madison)
  - Prof. Sören Schwertfeger (ShanghaiTech)
  - Prof. Kenji Kise (Tokyo Tech)
  - Prof. Wenzhi Chen (ZJU)