



University of
Nottingham

UK | CHINA | MALAYSIA

Introduction

Dr. Heng Yu

AY2023-24, Spring Semester
COMP1047: Systems and Architecture
Week 1



Computer Architecture Scope and Definitions



What is Architecture?



Computer Architecture Scope and Definitions

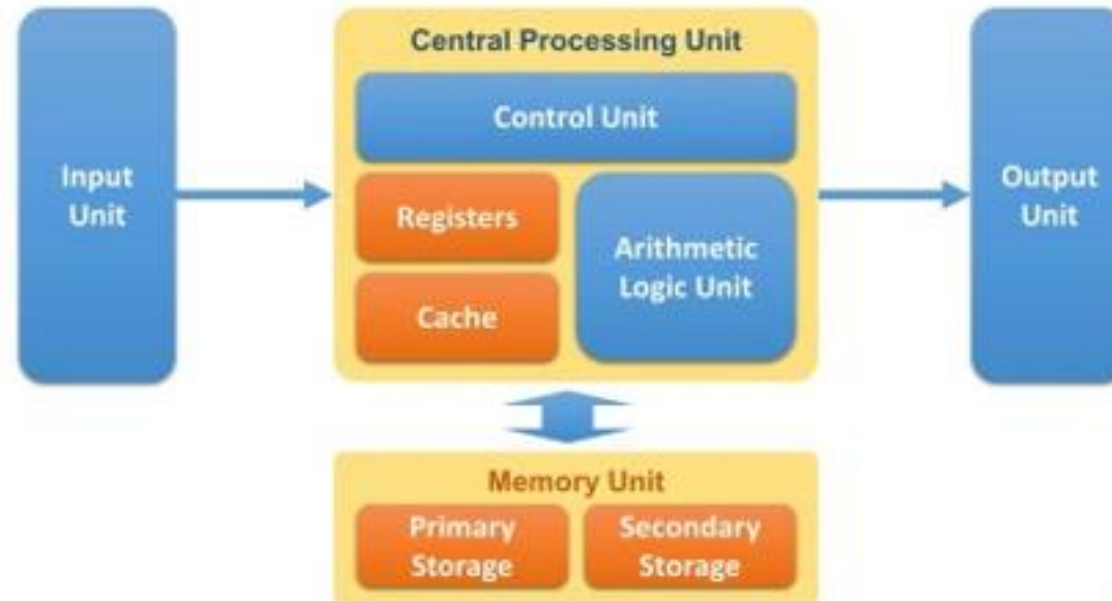
The art or science of designing and creating structures





Computer Architecture Scope and Definitions

Computer Architecture The art or science of designing a Computer



- **Personal Computers (PCs)**

- General purpose use
- Usually execute a variety of third-party software
- Subject to cost/performance tradeoff



- **Servers**

- Typically accessed via a network
- Carry large workloads (single complex application/many small jobs)
- Range from small servers to building sized



- **Embedded Computers**

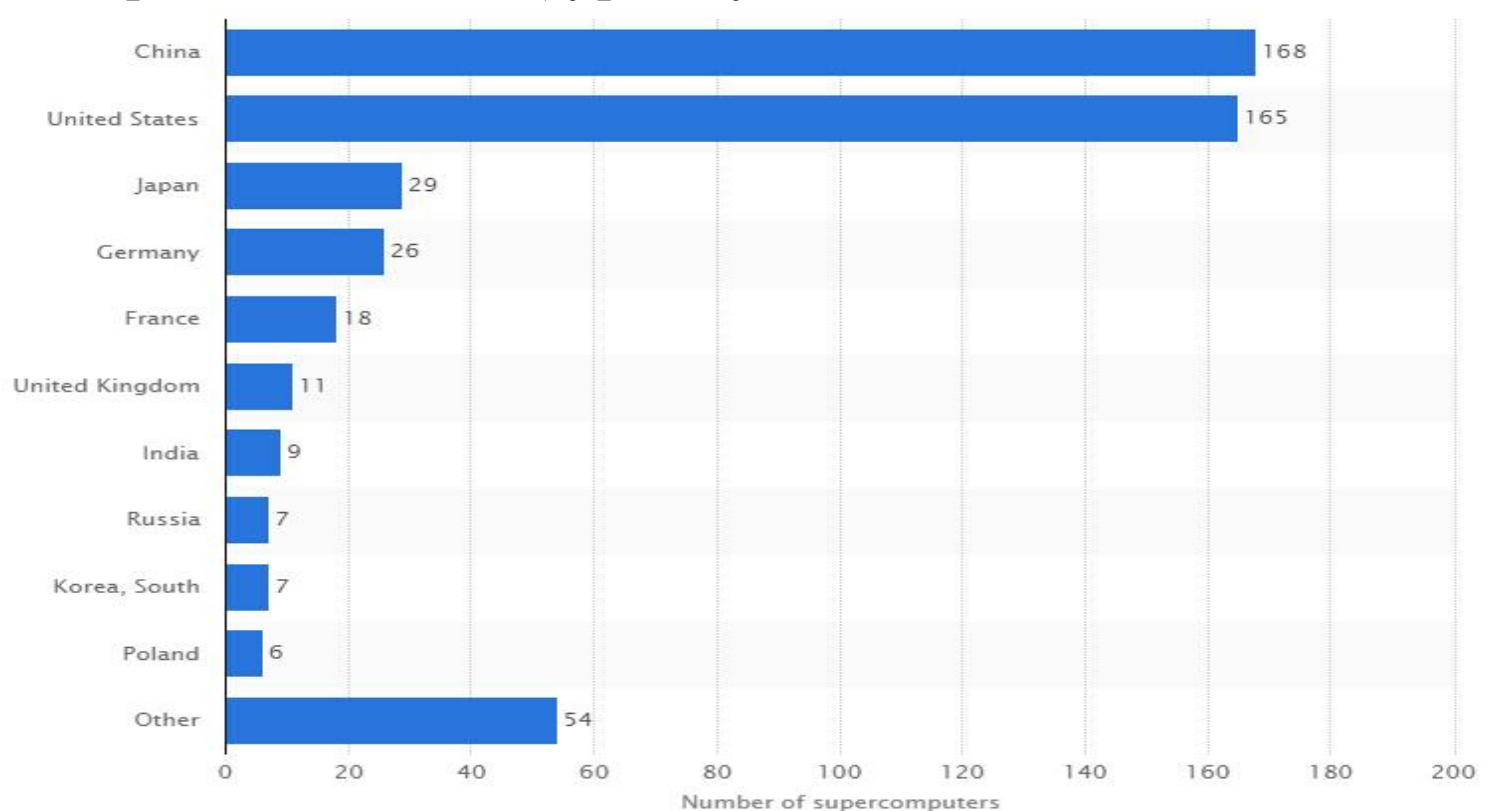
- Hidden as components of systems
- Designed to run one predetermined application or collection of software
- Low cost
- Low tolerance for failure (technologies of redundancy employed)





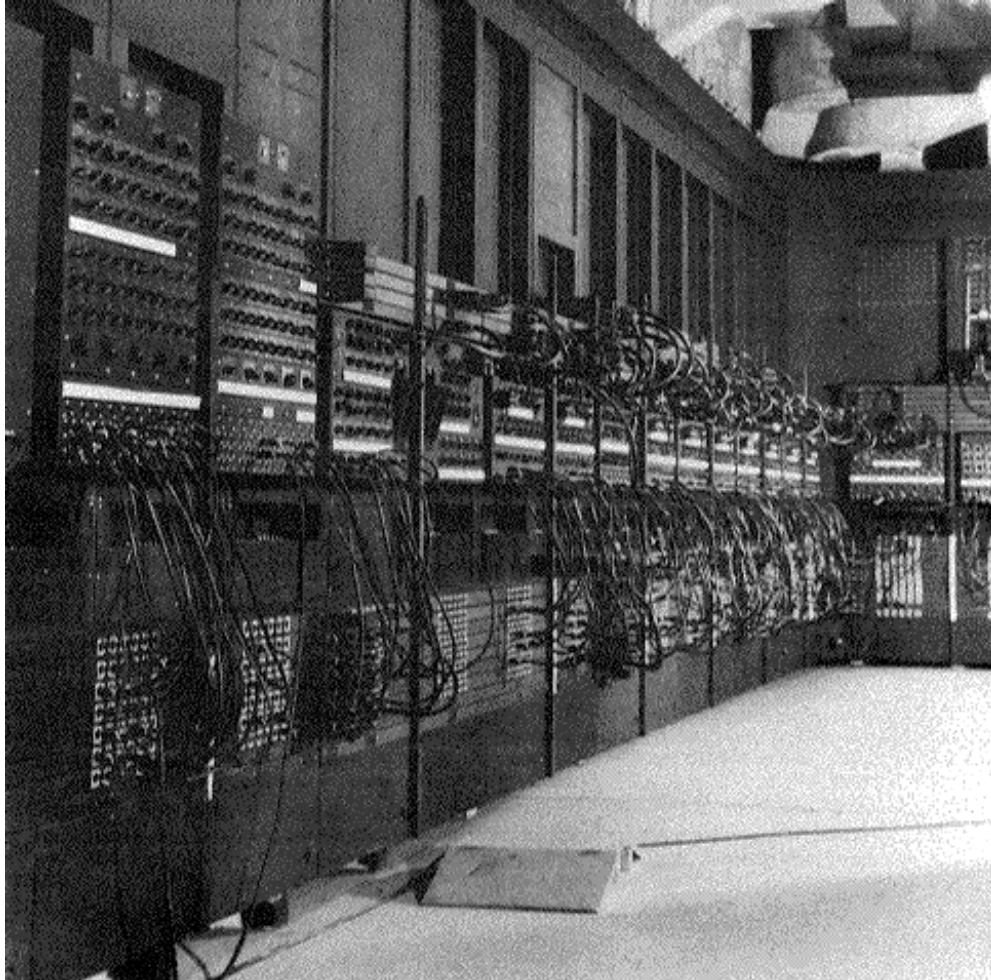
• Supercomputers

- A class of servers with the highest performance
- Perform high-end scientific and engineering calculations
- Expensive to build (typically 10s to 100s of millions of dollars)



Top 500 Supercomputers (June 2016)

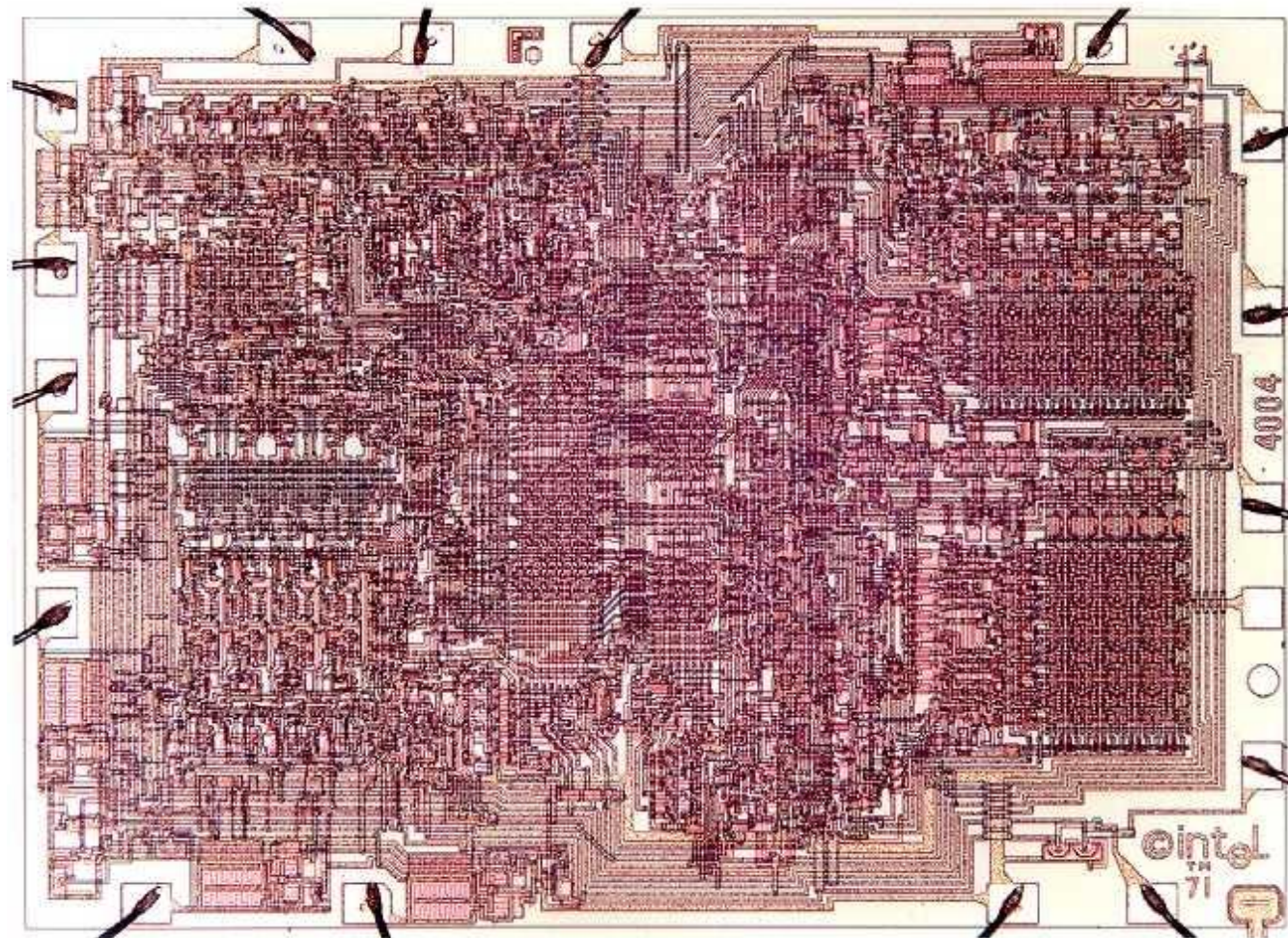
A Bit of History



- Electronic Numerical Integrator And Computer (ENIAC)
- The **first** electronic general-purpose **computer** dedicated at the University of Pennsylvania on February 15, 1946
 - Vacuum tube-based
 - Weighted more than 27 tons, was roughly $2.4\text{m} \times 0.9\text{m} \times 30\text{m}$, occupied 167m^2 and consumed 150 kW of electricity
 - Performance: roughly a few kilo-floating-point operations per second (kflops) (10^3)
- TaihuLight (2016) - 93 petaflops (10^{15})



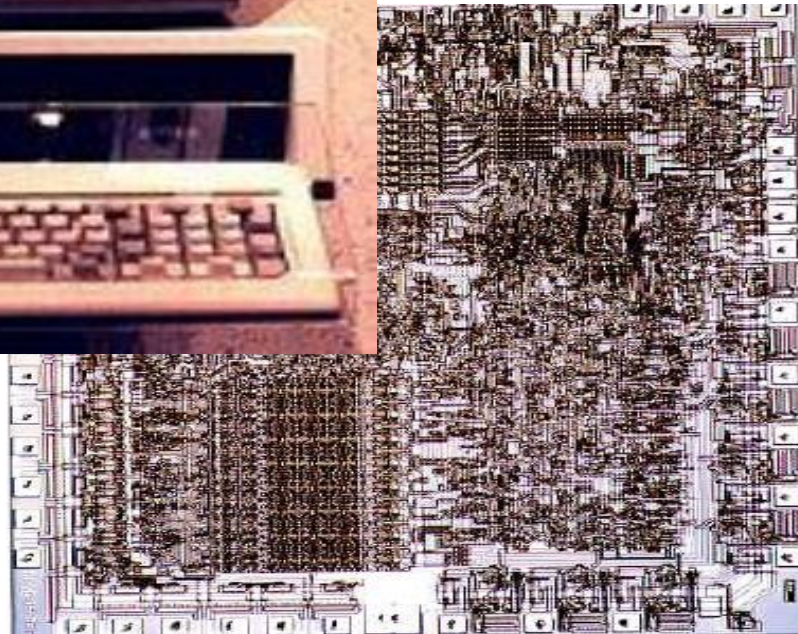
A Bit of History



- Microprocessor
 - Computer CPU on a single chip (IC)
 - Contain a number of transistors connected by wires
- **First commercial microprocessor:** Intel 4004
 - Introduced in 1971
 - 2300 silicon transistors
 - Technology: 10 μm (transistors are 10 μm across)
 - Has roughly the same processing power as ENIAC



A Bit of History



- Intel 8088
 - Use by IBM in its Original PC in 1981
 - 29000 transistors
 - Technology: 3 μm (transistors are 3 μm across)
 - Use operating system (MS-DOS) designed by Microsoft

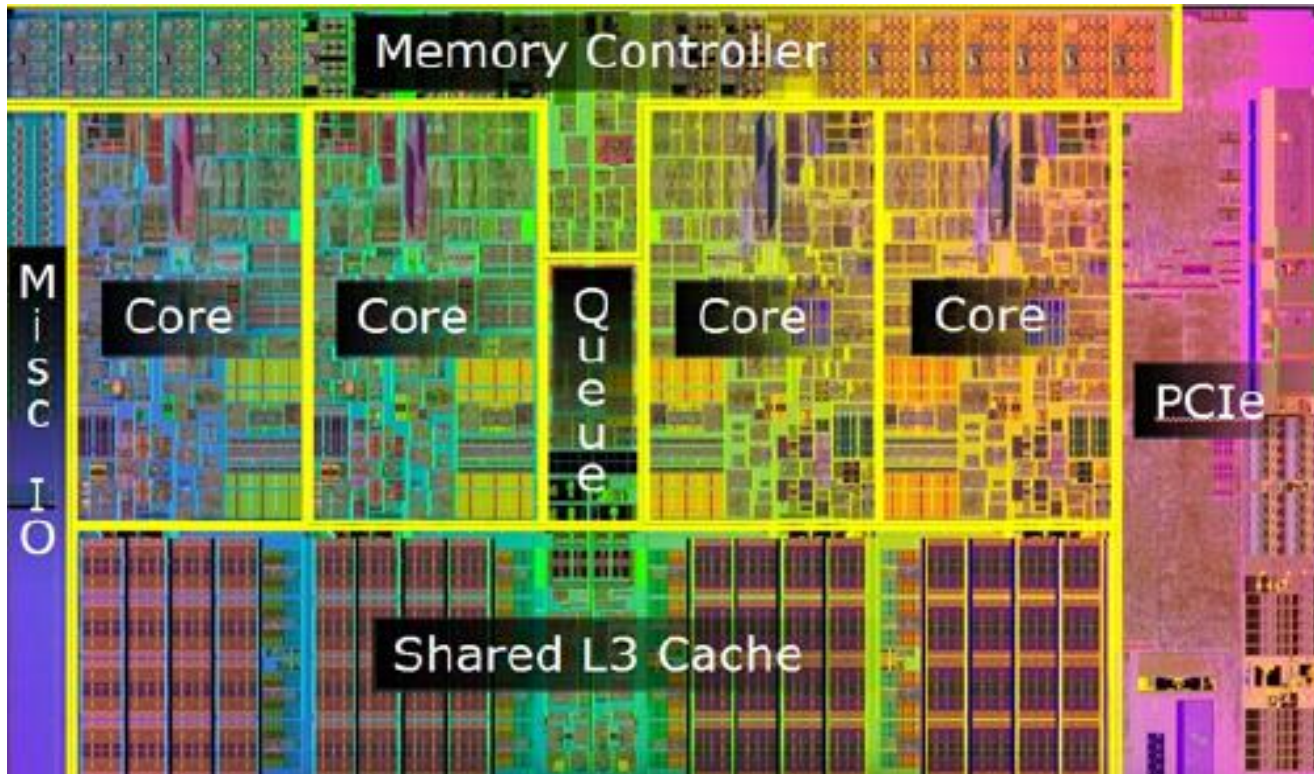


A Bit of History



- Intel Pentium4
 - Introduced in 2003
 - 55M transistors
 - Technology: 90 nm (transistors are 90 nm across)
 - Application: desktop/server

A Bit of History



- Multicore processor
 - A single chip contains more than one microprocessor core
- Intel Core i7
 - Introduced in 2009
 - 774M transistors
 - Technology: 32 nm (transistors are 32 nm across)
 - Four-core multicore
 - Application: desktop/server

[Video: How Microchips are made?](#)



A Bit of History

- Human hair is about 100 μm across
- Intel 4004 transistors are 10 μm across
- Intel 8088 transistors are 3 μm across
- Intel Pentium 4 transistors are 90 nm across
- Intel Core i7 transistors are 32 nm across
- Smaller transistors allow
 - More transistors per chip
 - More processing power
 - Smaller/cheaper chips
- Q: what is the tech scale for mass production as of now in the world? How about China?

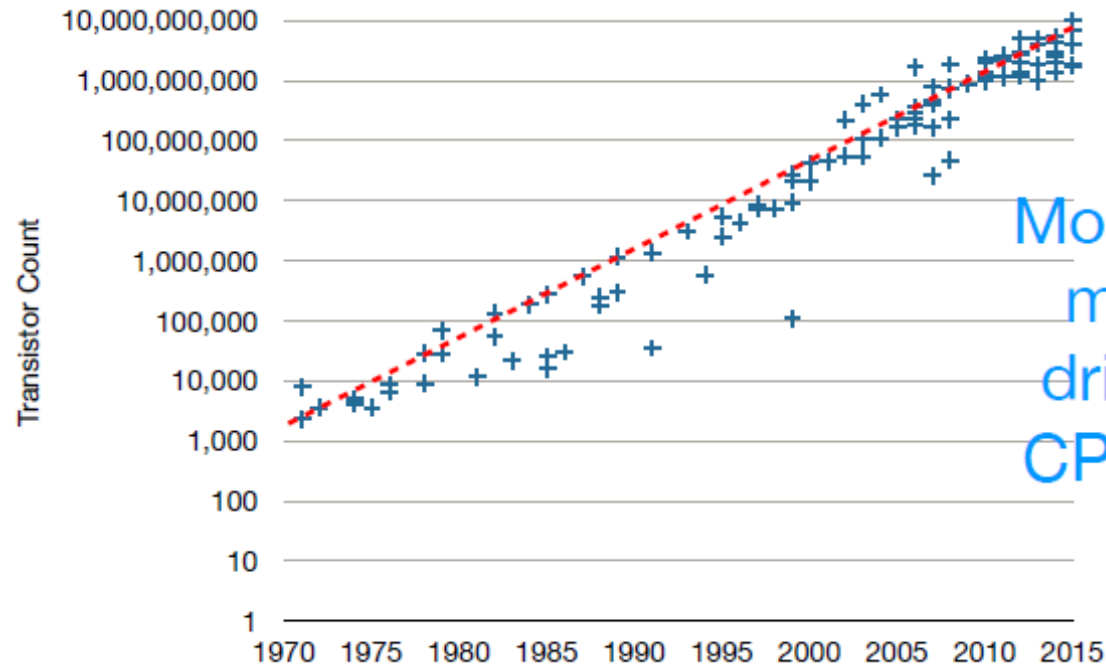


Moore's Law

The number of transistors we can build in a fixed area of silicon doubles every 12 ~ 24 months.

The chip performance would double every two years

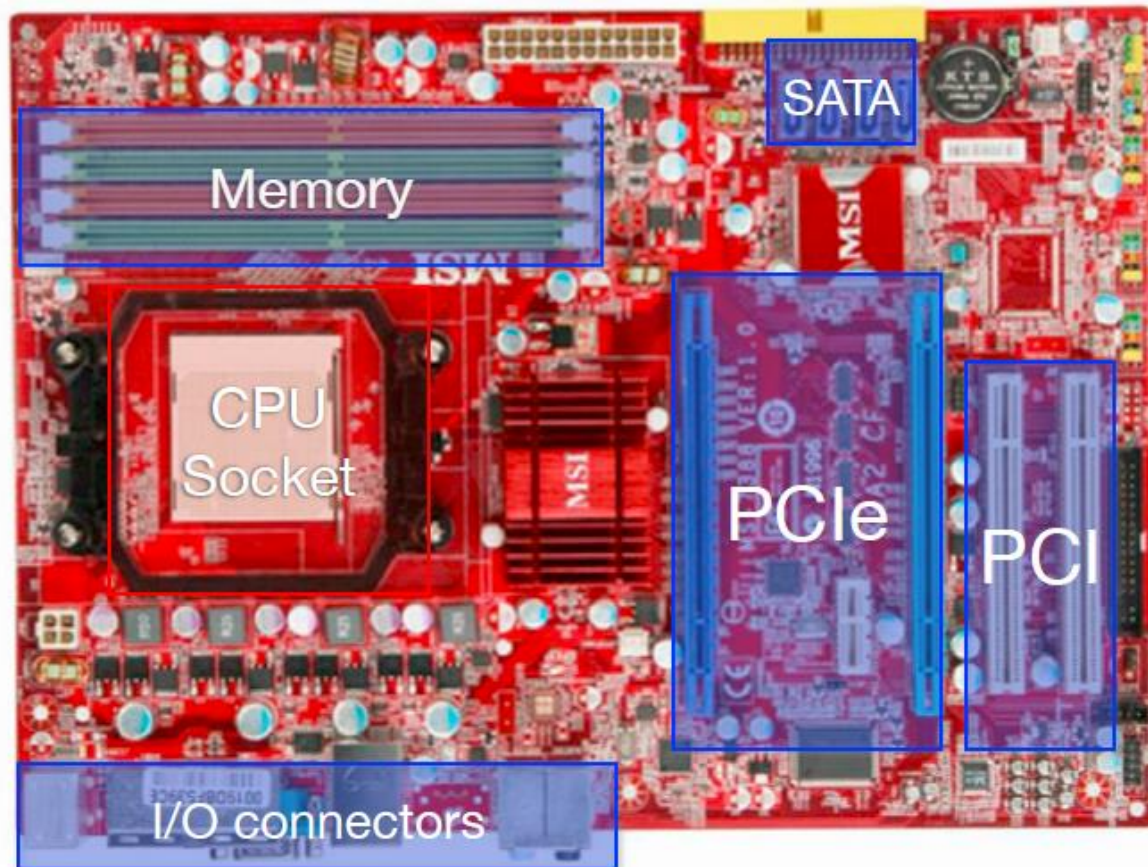
In celebration of ENIAC's 50th Anniversary, the machine was re-implemented using modern integrated circuit technology. The room-sized computer could now fit in the palm of a hand.



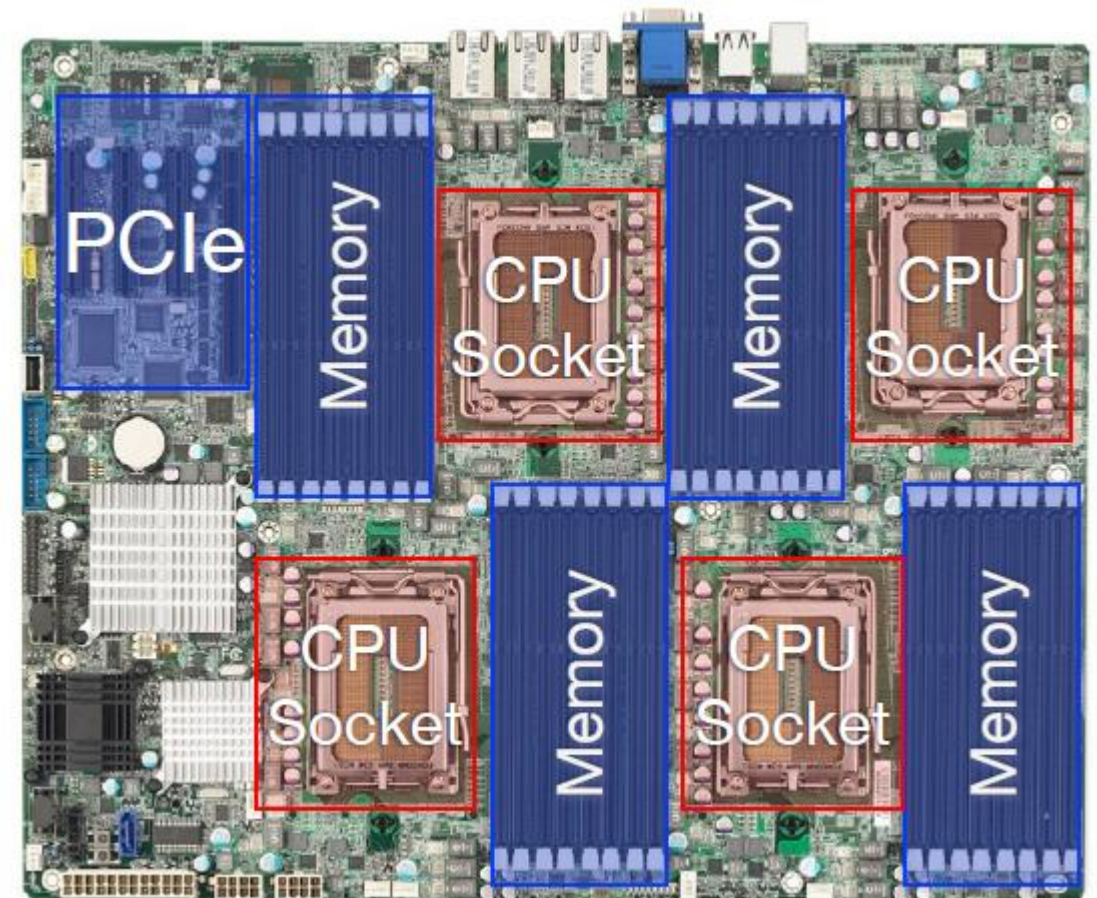
Moore's Law is the most important driver for historic CPU performance gains



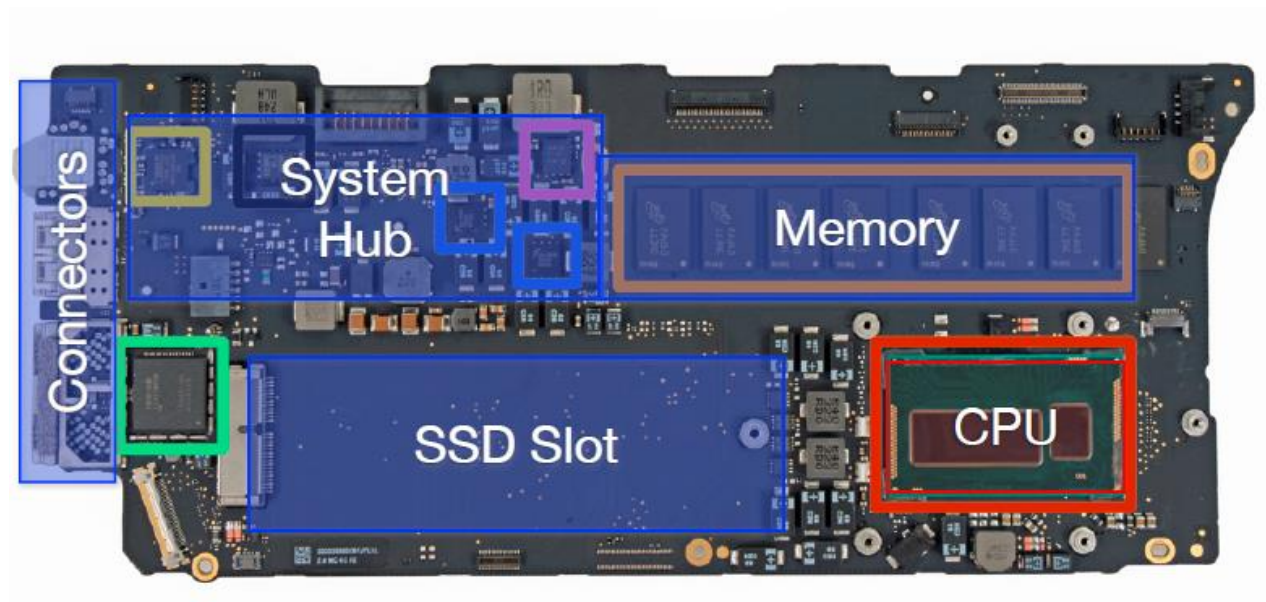
A Desktop PC



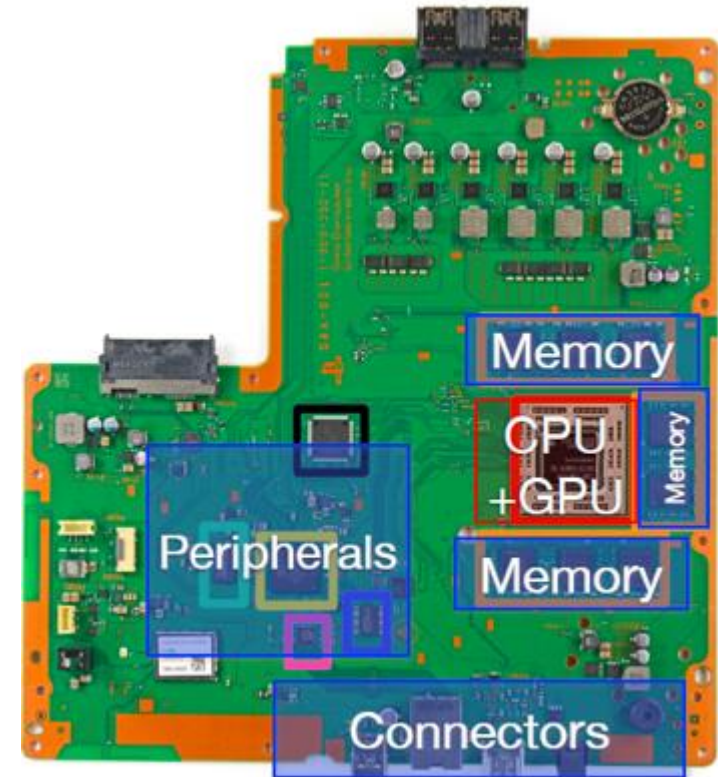
A Server



A MacBook Pro

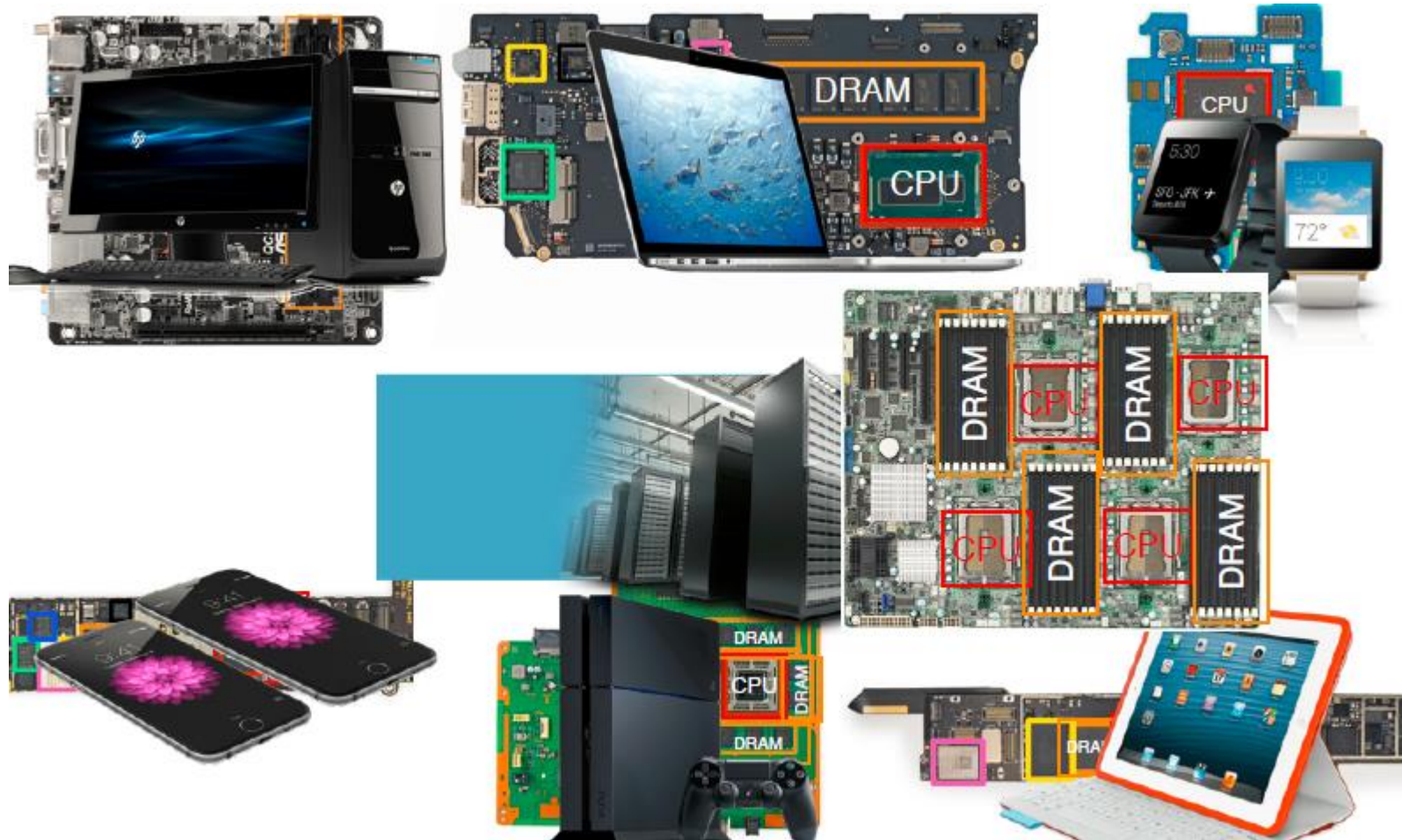


A PlayStation 4





Uncover a Computer



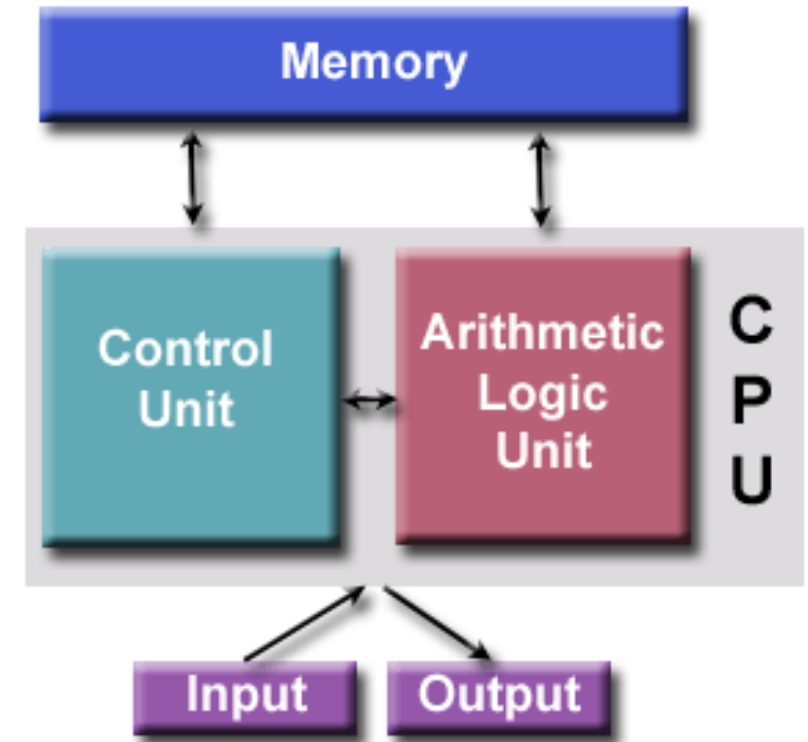
Difference faces, same spirit

Von Neumann architecture (also called Princeton architecture) is still the state-of-the-art computing architecture adopted.

Even in Quantum Computing. Quantum computing differentiates by its bit representation (q-bit), but many of the quantum computers still follow the von Neumann architecture.

Other architectures exist

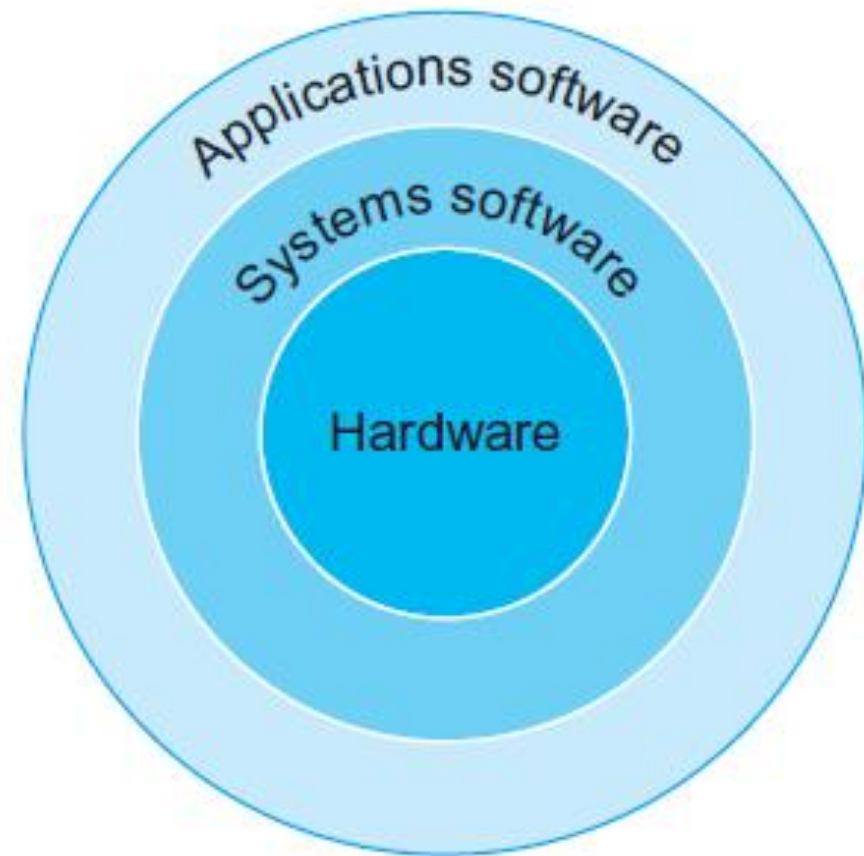
- Harvard architecture
- Dataflow architecture
- Neural computing
- etc.





Below Your Program

- Application software
 - Written in high-level language (HLL)
- System software
 - Compiler
 - Translate HLL code to machine code
 - Operating system
 - Handling basic input/output operations
 - Managing memory and storage
 - Scheduling tasks & sharing resources
- Hardware
 - Processor
 - Memory
 - I/O controllers





- To actually speak to electronic hardware, we need to send electrical signals
- The easiest signal for computers to understand – on/off
 - This corresponds to turning on and off transistors
- Therefore the hardware can only understand **binary representations**
- How can we program the machine?
 - Suppose you write a program in C, how can the machine understand it?
- Solution
 - Translate high-level language code into intermediate-level code (assembly code) which is more human-friendly
 - Then translate the assembly code into the machine's language (binary code).



Programming the Machine

High-level
language
program
(in C)

```
swap(int v[], int k)
{int temp;
  temp = v[k];
  v[k] = v[k+1];
  v[k+1] = temp;
}
```

Compiler

Assembly
language
program
(for MIPS)

```
swap:
  multi $2, $5, 4
  add   $2, $4, $2
  lw    $15, 0($2)
  lw    $16, 4($2)
  sw    $16, 0($2)
  sw    $15, 4($2)
  jr    $31
```

Assembler

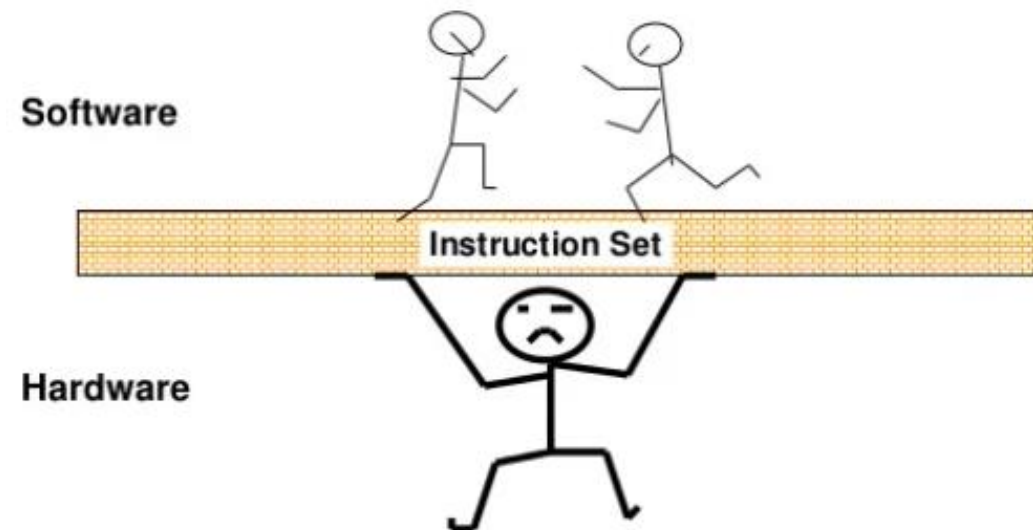
Binary machine
language
program
(for MIPS)

```
000000001010001000000000100011000
00000000100000100001000000100001
10001101111000100000000000000000
100011100001001000000000000000100
101011100001001000000000000000000
101011011110001000000000000000100
00000011111000000000000000001000
```

- **Compiler** – A program that translates high-level language statements into assembly language statements
- **Instruction** – A command that computer hardware understands and obeys
- **Assembler** – A program that translates a symbolic version of instructions into the binary versions
- **Assembly language** – A symbolic representation of machine instructions
- **Machine language** – A binary representation of machine instructions

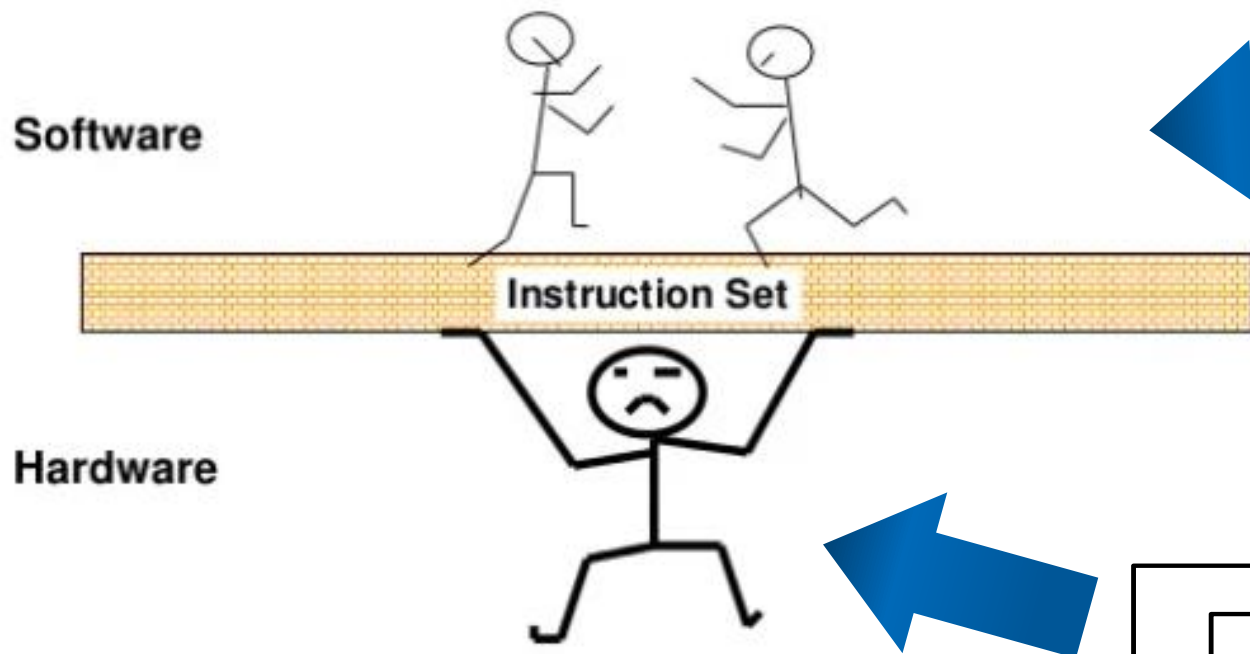
Instruction Set Architecture

- Defines the set of instructions that a computer/processor can execute
- The contract between the hardware and software
 - ➔ “Contract”: given an ISA, your sw and hw must be designed for the ISA! A glue for high and low levels of the system!
- Example ISAs:
 - ➔ x86: intel Xeon, intel Core i7/i5/i3, intel atom, AMD Athlon/Opteron, AMD FX, AMD A-series
 - ➔ ARM: Apple A-Series, Qualcomm Snapdragon, TI OMAP, NVidia Tegra
 - ➔ MIPS: Sony/Toshiba Emotion Engine, MIPS R-4000(PSP)
 - ➔ DEC Alpha, PowerPC, IA-64, SPARC and so on ...



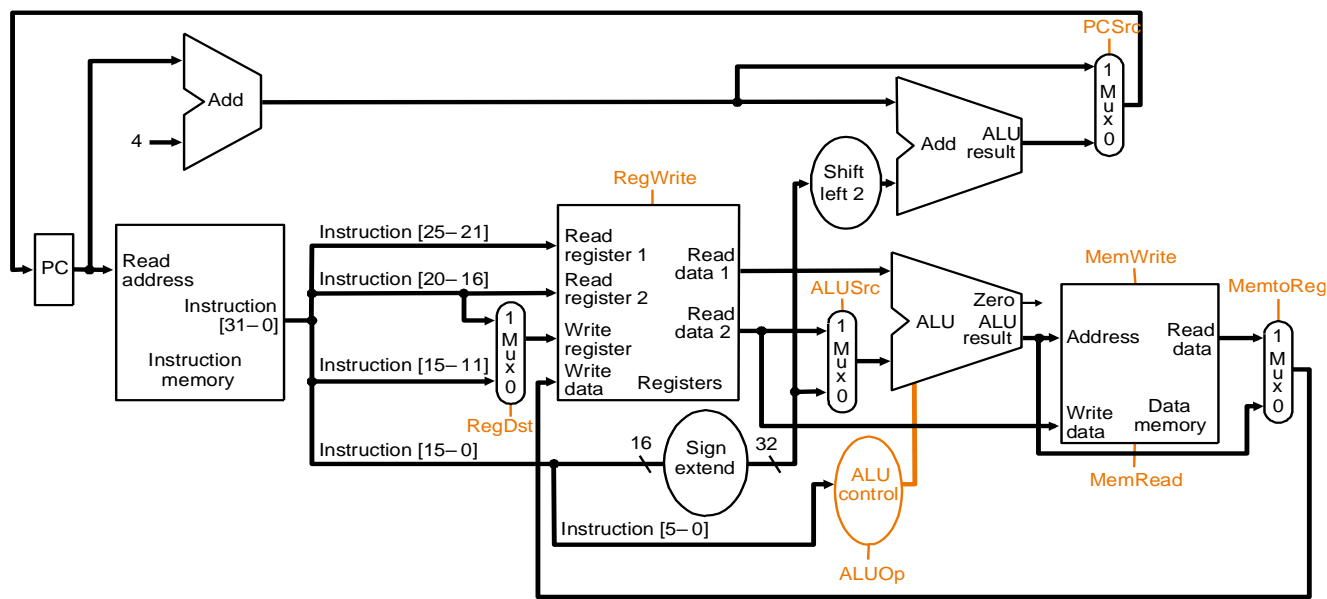


Instruction Set Architecture



```
hanoi:  addi $a0, $a0, -1
        bne  $a0, $zero, hanoi_1
        addi $v0, $zero, 1
        j    return
hanoi_1: jal  hanoi
        sll  $v0, $v0, 1
        addi $v0, $v0, 1
return:  jr   $ra
```

Week No.	Date From	Topic
1	19-Feb	Intro & Computer Performance
2	26-Feb	MIPS Assembly Programming 1
3	4-Mar	MIPS Assembly Programming 2
4	11-Mar	MIPS ISA and ALU Design
5	18-Mar	Single-Cycle CPU Design





What you will learn

- **What determines the performance of your computer**
- **The assembly language**
 - How does software instruct the hardware to perform needed functions
- **How programs are translated into the machine language**
- **How does the hardware execute the machine language**
- **How computers communicate with each other**



University of
Nottingham

UK | CHINA | MALAYSIA

Stay Tuned.