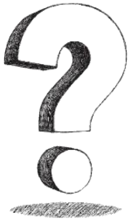
CH 1 Introduction

The Computer Revolution

- Unprecedented speed of progress and innovations in computer systems
  - Analogy to transportation: NY to London in seconds for a penny
- The third revolution of human civilization is made possible
- Enabling new applications
  - Improvement in the cost of computing by factor of 10 → Numerous applications and opportunities appear


Computers in Automobiles


Cell phones


What else?


World wide web

Search engine

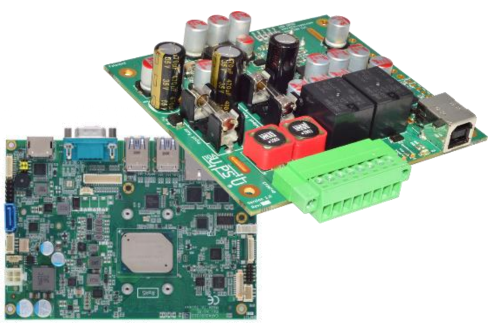

Human genome project

(Traditional) Class of Computers

Personal Computers
- General purpose, variety of software
- Subject to cost/performance tradeoff

Servers
- Network based
- High capacity, performance, reliability

Supercomputers
- High-end scientific and engineering calculations
- Highest capability and performance but represent a small fraction of the overall computer market



Embedded computers
- Hidden as components of systems
- Stringent power/performance/cost constraints

Supercomputer 2020

- Top500 from Wikipedia

Rank	Rmax Rpeak (PFLOPS)	Name	Model	Processor	Interconnect	Vendor	Site country, year	Operating system
1 ▲	415.530 513.855	Fugaku	Supercomputer Fugaku	A64FX	Tofu interconnect D	Fujitsu	RIKEN Center for Computational Science Japan, 2020	Linux (RHEL)
2 ▼	148.600 200.795	Summit	IBM Power System AC922	POWER9, Tesla V100	InfiniBand EDR	IBM	Oak Ridge National Laboratory United States, 2018	Linux (RHEL)
3 ▼	94.640 125.712	Sierra	IBM Power System S922LC	POWER9, Tesla V100	InfiniBand EDR	IBM	Lawrence Livermore National Laboratory United States, 2018	Linux (RHEL)
4 ▼	93.015 125.436	Sunway TaihuLight	Sunway MPP	SW26010	Sunway[25]	NRCPC	National Supercomputing Center in Wuxi China, 2016[25]	Linux (Raise)
5 ▼	61.445 100.679	Tianhe-2A	TH-IVB-FEP	Xeon E5–2692 v2, Matrix-2000[26]	TH Express-2	NUDT	National Supercomputing Center in Guangzhou China, 2013	Linux (Kylin)
6 ▲	35.450 51.721	HPC5	Dell	Xeon Gold 6252, Tesla V100	Mellanox HDR Infiniband	Dell EMC	Eni Italy, 2020	Linux (CentOS)
7 ▲	27.580 34.569	Selene	Nvidia	Epyc 7742, Ampere A100	Mellanox HDR Infiniband	Nvidia	Nvidia United States, 2020	Linux (Ubuntu)
8 ▼	23.516 38.746	Frontera	Dell C6420	Xeon Platinum 8280 (subsystems with e.g. POWER9 CPUs and Nvidia GPUs were added after official benchmarking[10])	InfiniBand HDR	Dell EMC	Texas Advanced Computing Center United States, 2019	Linux (CentOS)
9 ▲	21.640 29.354	Marconi-100	IBM Power System AC922	POWER9, Volta V100	Dual-rail Mellanox EDR Infiniband	IBM	CINECA Italy, 2020	Linux (RHEL)
10 ▼	21.230 27.154	Piz Daint	Cray XC50	Xeon E5-2690 v3, Tesla P100	Aries	Cray	Swiss National Supercomputing Centre Switzerland, 2016	Linux (CLE)

Supercomputer 2021

- Top500 from Wikipedia

Rank (previous)	Rmax Rpeak (PFLOPS)	Name	Model	CPU cores	Accelerator (e.g. GPU) cores	Interconnect	Manufacturer	Site country	Year	Operating system
1	442.010 537.212	Fugaku	Supercomputer Fugaku	158,976 × 48 A64FX @2.2 GHz	0	Tofu interconnect D	Fujitsu	RIKEN Center for Computational Science Japan	2020	Linux (RHEL)
2▼ (1)	148.600 200.795	Summit	IBM Power System AC922	9,216 × 22 POWER9 @3.07 GHz	27,648 × 80 Tesla V100	InfiniBand EDR	IBM	Oak Ridge National Laboratory United States	2018	Linux (RHEL)
3▼ (2)	94.640 125.712	Sierra	IBM Power System S922LC	8,640 × 22 POWER9 @3.1 GHz	17,280 × 80 Tesla V100	InfiniBand EDR	IBM	Lawrence Livermore National Laboratory United States	2018	Linux (RHEL)
4▼ (3)	93.015 125.436	Sunway TaihuLight	Sunway MPP	40,960 × 260 SW26010 @1.45 GHz	0	Sunway[34]	NRCPC	National Supercomputing Center in Wuxi China[34]	2016	Linux (Raise)
5▲ (new)	64.590 89.795	Perlmutter	HP	? × 64 Epyc 7763 @2.45 GHz	? × 108 Ampere A100	Slingshot-10	Nvidia	NERSC United States	2021	Linux (Cray Linux Environment)
6▼ (5)	63.460 79.215	Selene	Nvidia	1,120 × 64 Epyc 7742 @2.25 GHz	4,480 × 108 Ampere A100	Mellanox HDR Infiniband	Nvidia	Nvidia United States	2020	Linux (Ubuntu)
7▼ (6)	61.445 100.679	Tianhe-2A	TH-IVB-FEP	35,584 × 12 Xeon E5–2692 v2 @2.2 GHz	35,584 × 128 Matrix-2000[35]	TH Express-2	NUDT	National Supercomputer Center in Guangzhou China	2013	Linux (Kylin)
8▼ (7)	44.120 70.980	JUWELS (booster module)[36][37]	BullSequana XH2000	1,872 × 24 AMD Epyc 7402 @2.8 GHz	3,744 × 108 Ampere A100	Mellanox HDR Infiniband	Atos	Forschungszentrum Jülich Germany	2020	Linux (CentOS)
9▼ (8)	35.450 51.721	HPC5	Dell	3,640 × 24 Xeon Gold 6252 @2.1 GHz	7,280 × 80 Tesla V100	Mellanox HDR Infiniband	Dell EMC	Eni Italy	2020	Linux (CentOS)
10▼ (9)	23.516 38.746	Frontera	Dell C6420	16,016 × 28 Xeon Platinum 8280 @2.7 GHz (subsystems with e.g. POWER9 CPUs and Nvidia GPUs were added after official benchmarking[10])	0	InfiniBand HDR	Dell EMC	Texas Advanced Computing Center United States	2019	Linux (CentOS)

The Post-PC Era

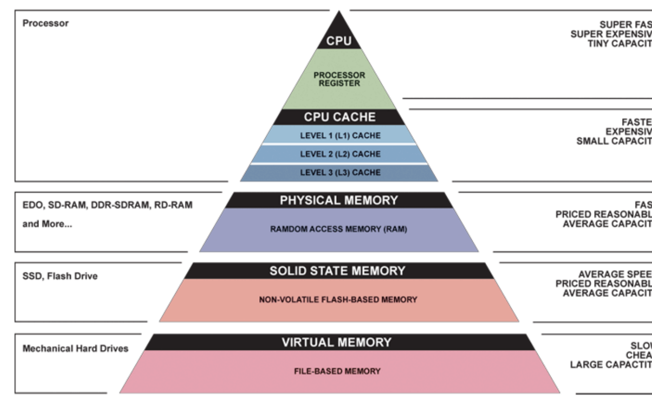
- Personal Mobile Device (PMD)
  - Battery operated
  - Connects to the Internet
  - Smart phones, Tablets, Electronic glasses
- Cloud computing
  - Warehouse Scale Computers (WSCs): 100,000 and more servers
  - Software-as-a-Service (SaaS) model
  - Portions of SW run on PMD and another portion in the Cloud
  - ~~Fog/Edge computing~~

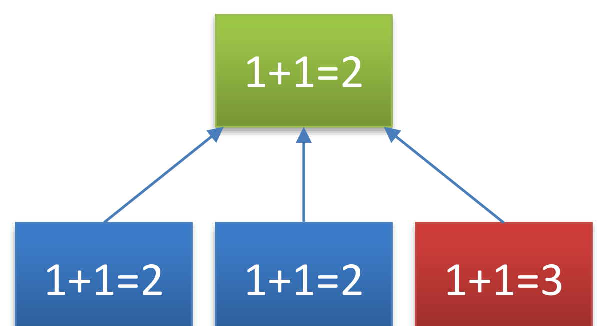
What You Will Learn

- How does high-level language programs run on the hardware?
- What's the interface between S/W and H/W?
- What determines the performance?
- How can we improve the performance?
- What are the reasons for and the consequences of the recent switch from sequential processing to parallel processing?

Important Concepts In Computer Architecture

- Design for Moore's Law
- Use abstraction to simplify design
  - Layers of Representation
- Make the common case fast
  - Related idea: Amdahl's Law
- Performance via:
  - parallelism 
  - pipelining 
  - prediction 
- Hierarchy of memories
- ~~Dependability via Redundancy~~



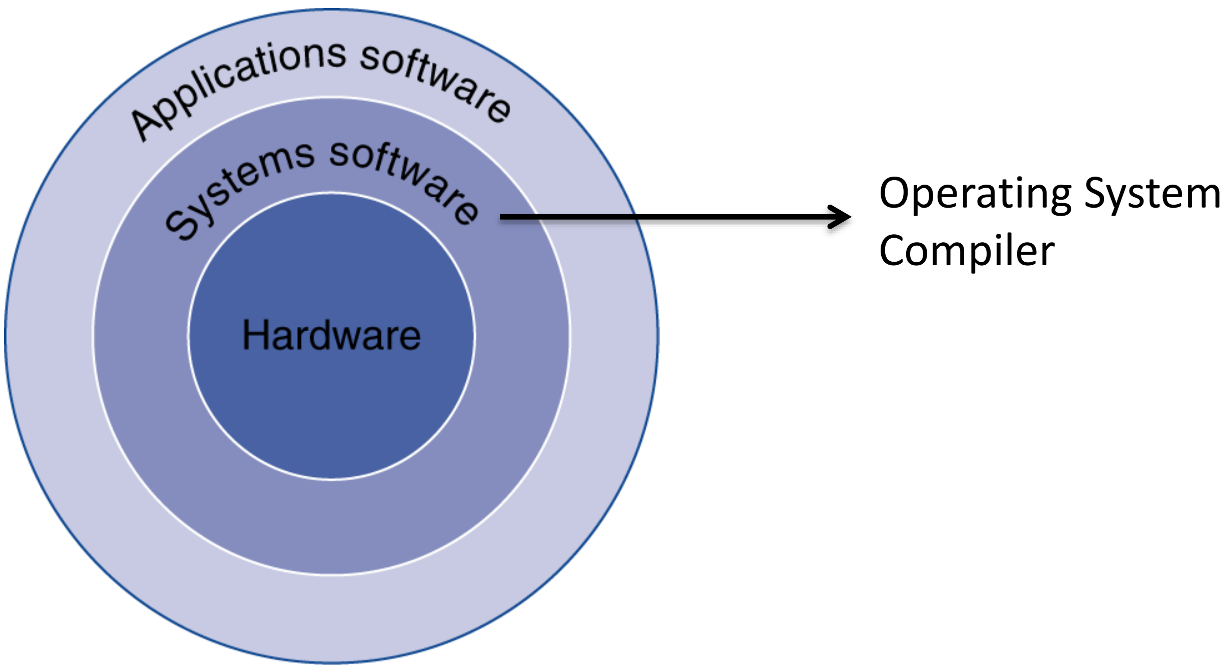


Below Your Program

- Word processor, database, … etc.
  - Millions of lines of code

- Hardware can run only low-level, simple instructions
  - ADD, LOAD, STORE, JUMP … etc

- Several layers between this gap



High-level Language → Instructions

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)

Low level programming language

```
swap:
      muli $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
```

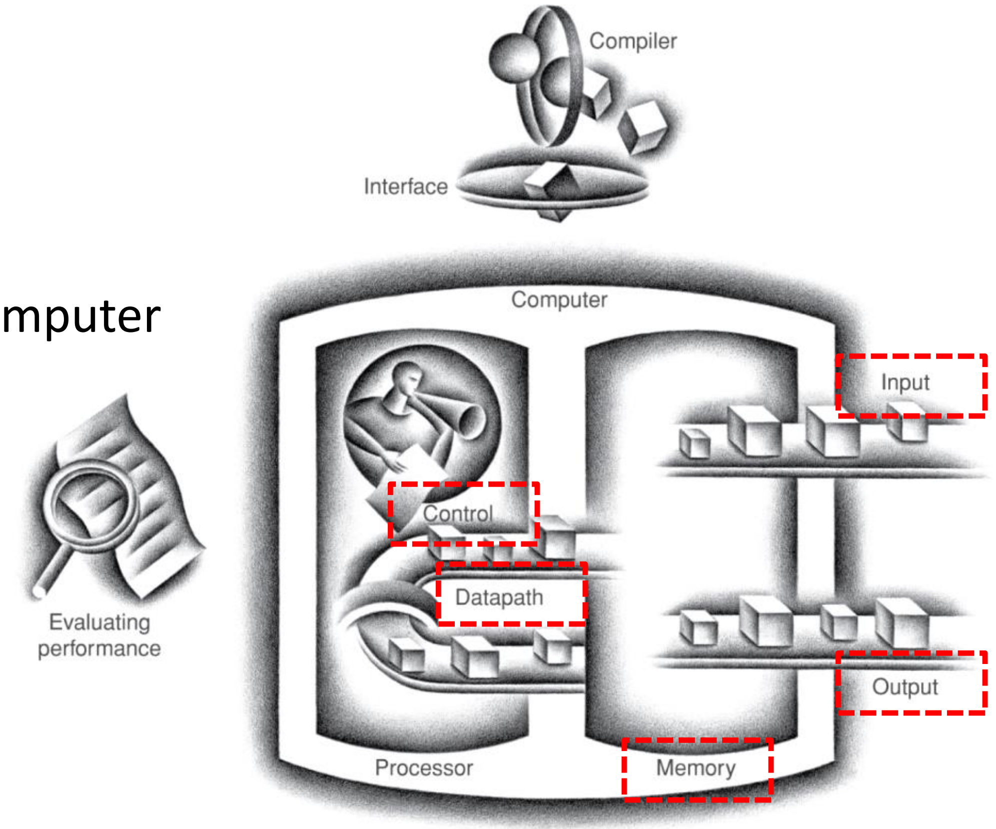
Assembler

Binary machine language program (for MIPS)

1000110010101000**00000010**00110000
00000000101000010000000000011000
00000000000110000001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000

Components of a Computer

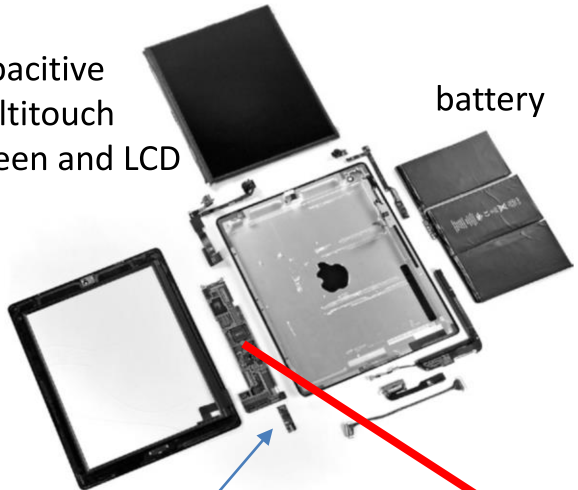
- Basic functions of a Computer
  - Input/Output data
  - Process the data
  - Store the data
- Same components for all kinds of computer
  - Independent of H/W technology
- Processor: Datapath+Control
- Input/output includes
  - User-interface devices
    - Display, keyboard, mouse
  - Storage devices
    - Hard disk, CD/DVD, flash
  - Network adapters
    - For communicating with other computers



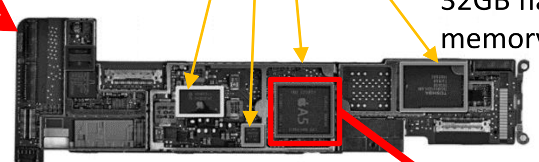
Five classic components of a computer

Opening the Box

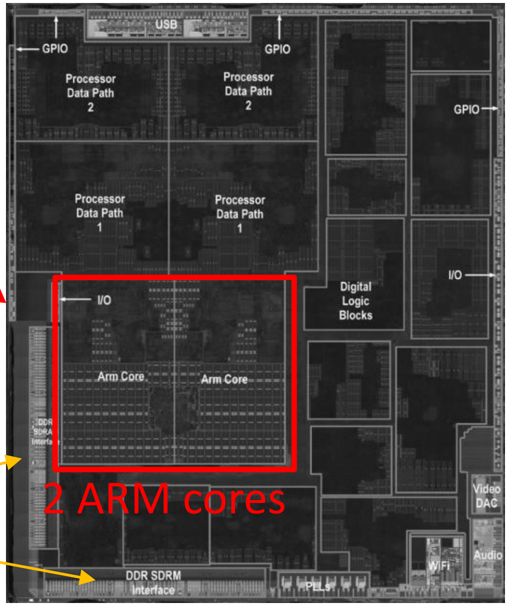
- I/O dominate the Apple iPad 2 tablet
  - I/O: a capacitive multitouch LCD, front-/rear-facing cameras, microphone, speakers, Wi-Fi and Bluetooth networks, etc
  - The datapath, control, and memory are a tiny portion



Integrated circuits (chips)



ARM processor A5



- A5 package includes two memory chips, each with 2 Gb and total 512MB

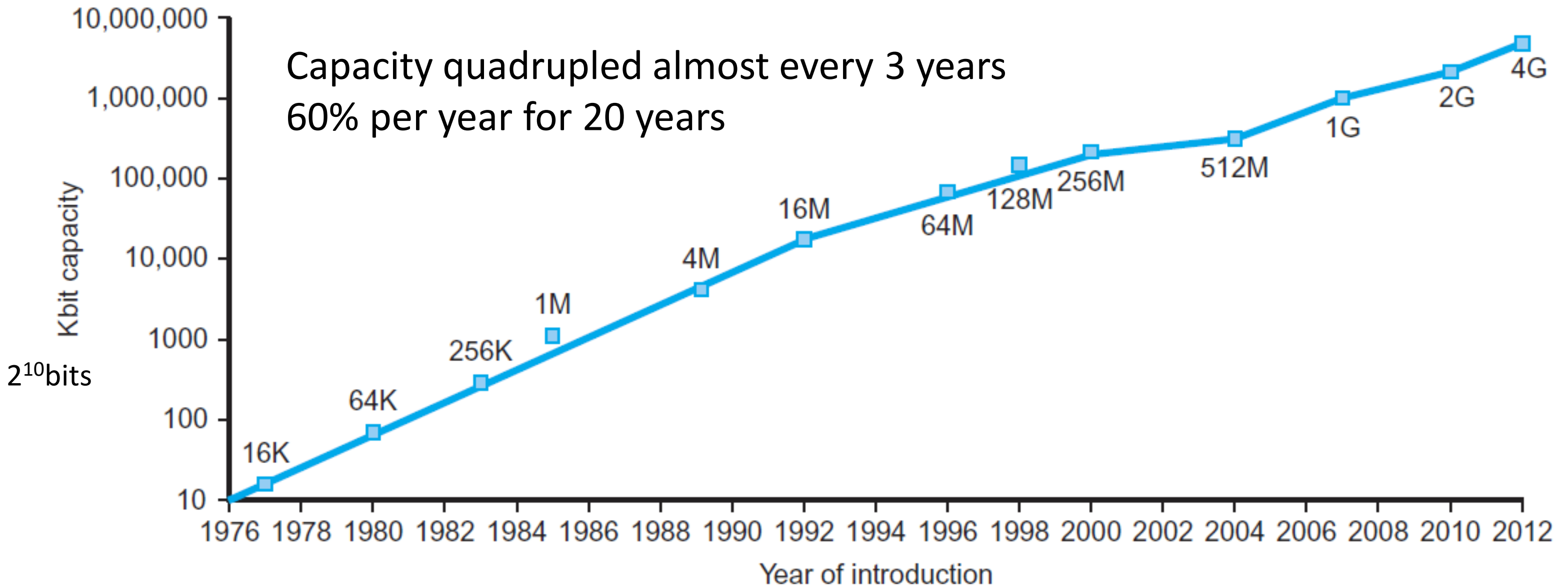
Technology Trends

- Processors and memory have improved at an incredible rate

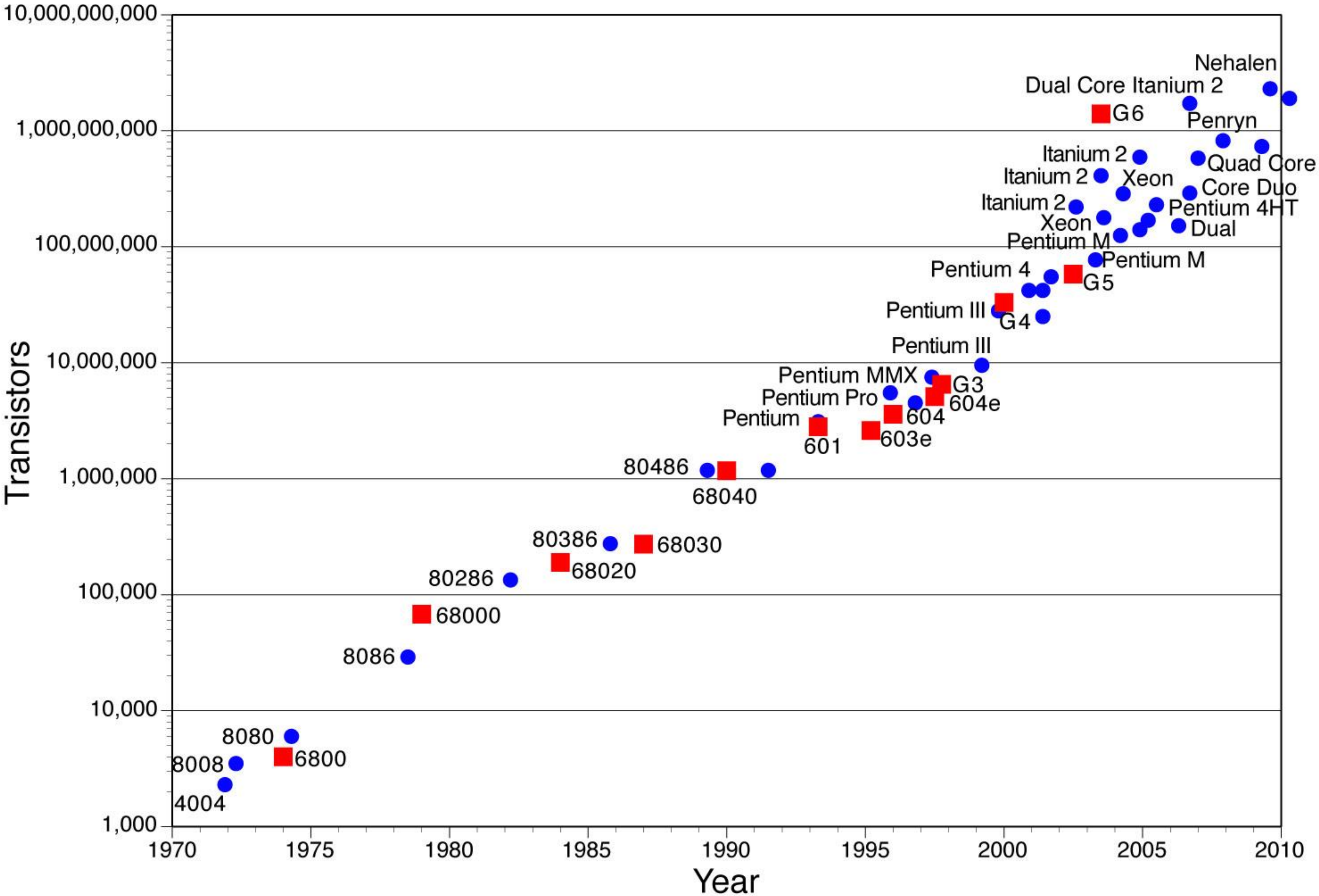
Year	Technology used in computers	Relative performance/unit cost
1951	Vacuum tube	1
1965	Transistor	35
1975	Integrated circuit	900
1995	Very large-scale integrated circuit	2,400,000
2013	Ultra large-scale integrated circuit	250,000,000,000

* A transistor is simply an on/off switch controlled by electricity.

- Growth of DRAM Capacity



Moore's Law Revisited



Performance

- Measuring the performance is challenging!!
  - Why?
    - Are you comparing apple to orange?
    - When money is involved …
- Two key metrics in computer performance
  - Execution time (Response time)
    - The time from the start to end
  - Throughput
    - Total amount of work done in a given time

Defining The Performance

- Focus on the response time for now
- Performance = 1/Execution Time

$P_X > P_Y$

$1/E_X > 1/E_Y$

$E_X < E_Y$

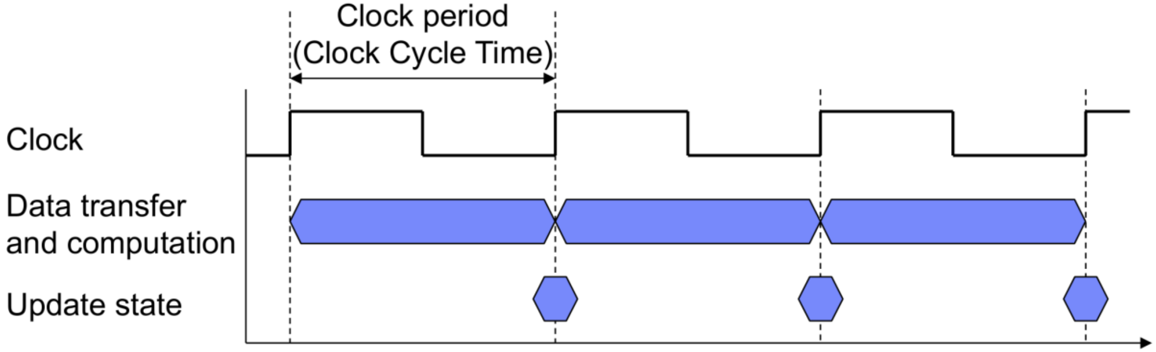
- X is n times faster than Y

$P_X/P_Y = n$

$P_X/P_Y = E_Y/E_X = n$

Measuring Time

- Wall clock time, Elapsed time
  - Total time it took to complete the job (or task)
- CPU time vs. I/O time
- Clock cycle (ticks, clock ticks, cycles …)
  - Operation of digital hardware governed by a constant rate clock



  - Clock frequency (rate): cycles per second
    - e.g., 4.0GHz = 4000MHz = 4.0×10^{9} Hz = $1/(250\times10^{-12})$
  - Clock period (Clock Cycle Time): duration of a clock cycle = 1/clock frequency
    - e.g., 250ps = 0.25ns = 250×10^{-12} s

Classic CPU Performance Equation

- Clock Cycle Time = $\frac{1}{Clock\ Rate}$
- CPU time = **CPU clock cycles** x Clock Cycle Time
- CPU time = **CPU clock cycles** / Clock Rate

> - **IC** (Instruction count)
>   Number of instructions executed by the program
> - **CPI** (Clock Cycles Per Instruction)
>   Average clock cycles per instructions for a program (fragment)
>   CPI = Clock Cycles / Instruction Count

- **CPU clock cycles** = **IC** x **CPI**
- CPU time = IC x CPI x Clock Cycle Time
- CPU time = $\frac{IC \times CPI}{Clock\ Rate}$

Using CPI (Clock cycles Per Instruction)

- Example
  - Computer A has clock cycle time of 250 ps and CPI of 2.0
  - Computer B has clock cycle time of 500 ps and a CPI of 1.2

Which computer is faster?

N: total # of instructions in a program

CPU $time_A$ = N x 2.0 x 250 = 500N ps

CPU $time_B$ = N x 1.2 x 500 = 600N ps

$$\frac{CPU\ performanceA}{CPU\ performanceB} = \frac{Exec\ TimeB}{Exec\ TimeA} = \frac{600N}{500N} = 1.2$$

Computer A is 1.2 times faster than Computer B

Comparing Code Segments

- CPI Information

	CPI for each instruction class		
	A	B	C
CPI	1	2	3

- Code segment Information

Code Sequence	Instruction counts for each instruction class		
	A	B	C
1	200	100	200
2	400	100	100

- Which code sequence executes the most instructions?
  - Segment1: 200 + 100 + 200 = 500 instructions
  - Segment2: 400 + 100 + 100 = 600 instructions
- Which will be faster?
  - Segment1: 200x1 + 100x2 + 200x3 = 1000 cycles
  - Segment2: 400x1 + 100x2 + 100x3 = 900 cycles
- CPI
  - Segment1: 1000/500 = 2.0 CPI
  - Segment2: 900/600 = 1.5 CPI

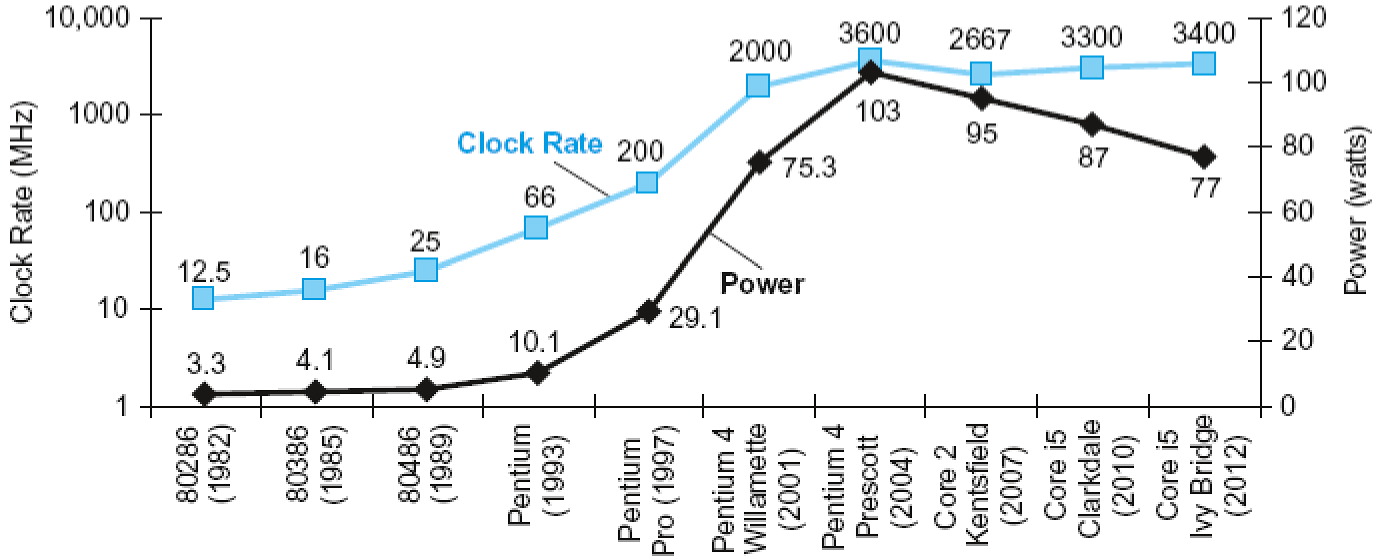
Performance Summary

CPU time = IC x CPI x Clock Cycle Time

$$\text{CPU time} = \frac{Instructions}{Program} \times \frac{Clock\ Cycles}{Instruction} \times \frac{Seconds}{Clock\ Cycle}$$

- Performance difference can arise from:
  - Algorithm
    - affects instruction count, and maybe CPI
  - Programming Language
    - affects instruction count and CPI
  - Compiler
    - affects instruction count and CPI
  - ISA (Instruction Set Architecture)
    - affects instruction count, clock rate, CPI

Power Trend

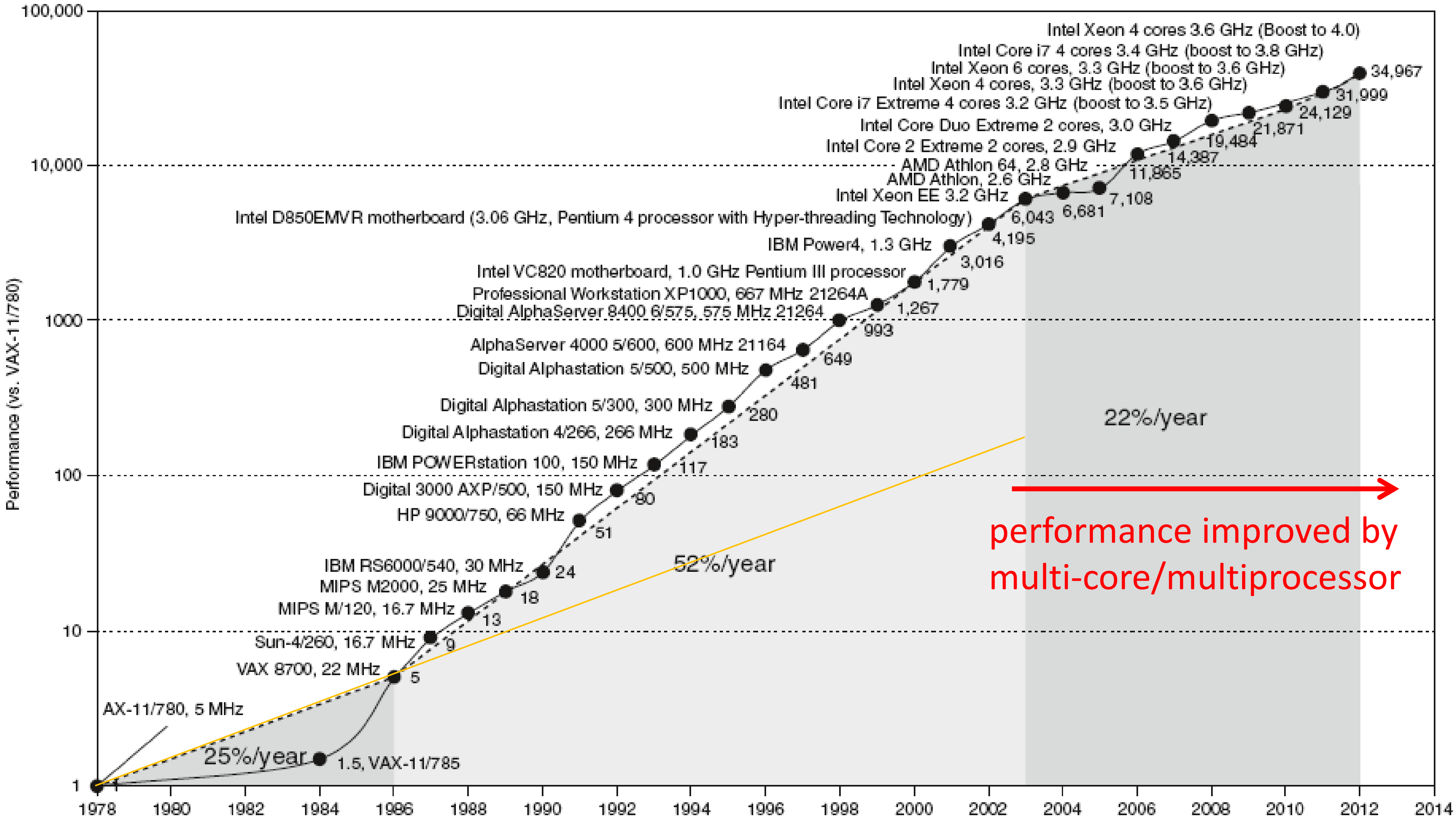


$$\text{Power} \propto \text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency}$$

- Faster clock rate means better performance
- Clock rate and power increase slowed
  - Too much power creates cooling problem
- Reducing required voltage allows the increase of the clock rate
  - → not able to reduce the voltage any more
- What can we do to improve the performance?

From Uniprocessor to Multiprocessor

Growth of processor performance



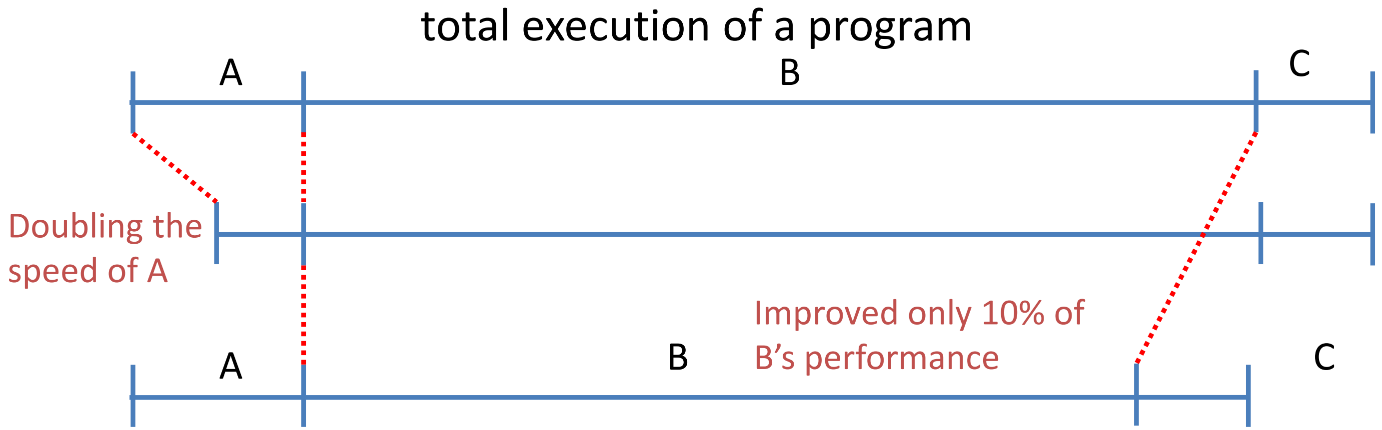
Measured by SPECint benchmark

Implication

- Previously
  - Since H/W performance increased, S/W performance ***automatically*** increased without any change
- Multi-core (=processor) H/W
  - dual-core, quad-core … etc
  - S/W has to be re-written to utilize the multi-core.
    - Parallel programming is challenging
      - load balancing
      - communication overhead
      - synchronization

Amdahl's Law

- Making the common case faster



- Common misbelief
  - Expecting the improvement of one aspect to return the same amount of increase in overall performance
- Amdahl's Law
  - The speedup of a program using multiple processors in parallel computing is limited by the time needed for the sequential fraction of the program.
  - Execution time after improvement

$$= \frac{Execution\ time\ affected\ by\ improvement}{Amount\ of\ improvement} + Execution\ time\ unaffected$$

Amdahl's Law (continued)

- Example
  - A program runs in 100 seconds.
  - Multiply operation consumes 80 seconds.
  - We want to improve the performance 5 times.
  - How much do we need to improve the performance of the multiply operation?

$T = 80/n + (100-80)$

5 times faster = 20 seconds

$20 = 80/n + 20$

$0 = 80/n$ ← Not possible to get 5 times improvement!!!

Pitfalls

- ~~Computers at low utilization use little power.~~
  - At 10% utilization, power consumption is 33% of peak power consumption.
- Designing for performance and designing for energy efficiency are ~~un~~related goals.
- Using a subset of the performance equation as a performance metric.
  - MIPS (million instructions per second)

$$\text{MIPS} = \frac{\text{Instruction count}}{\text{Execution time} \times 10^6} = \frac{\text{Instruction count}}{\frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}} \times 10^6} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}$$

  - Cannot compare different ISA using MIPS
  - MIPS varies by program