Chapter 1: Computer Abstractions and Technology

1.8 - 1.9 and the rest: Multiprocessing and benchmarking, etc

ITSC 3181 Introduction to Computer Architecture

https://passlab.github.io/ITSC3181/

Department of Computer Science

Yonghong Yan

yyan7@uncc.edu

https://passlab.github.io/yanyh/

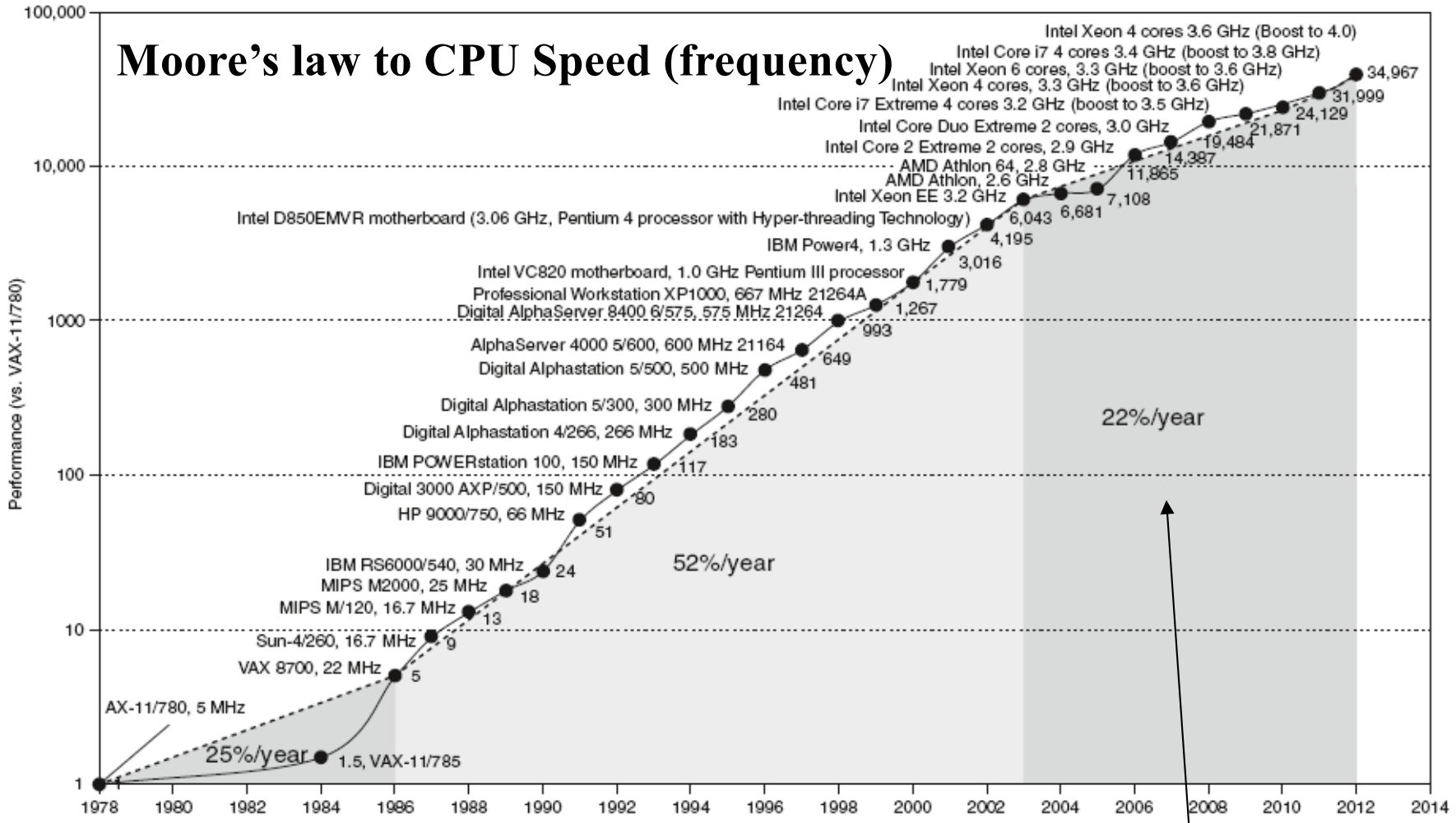
Lectures for Chapter 1 and C Basics
Computer Abstractions and Technology

- **Lecture 01: Chapter 1**
  - **1.1 – 1.4: Introduction, great ideas, Moore's law, abstraction, computer components, and program execution**

- **Lecture 02: C Basics; Memory and Binary Systems**
- **Lecture 03: Number System, Compilation, Assembly, Linking and Program Execution**
- **Lecture 04: Chapter 1**
  - **1.6 – 1.7: Performance, power and technology trends**

☛ **Lecture 05:**
  - **1.8 - 1.9: Multiprocessing and benchmarking**

Uniprocessor Performance

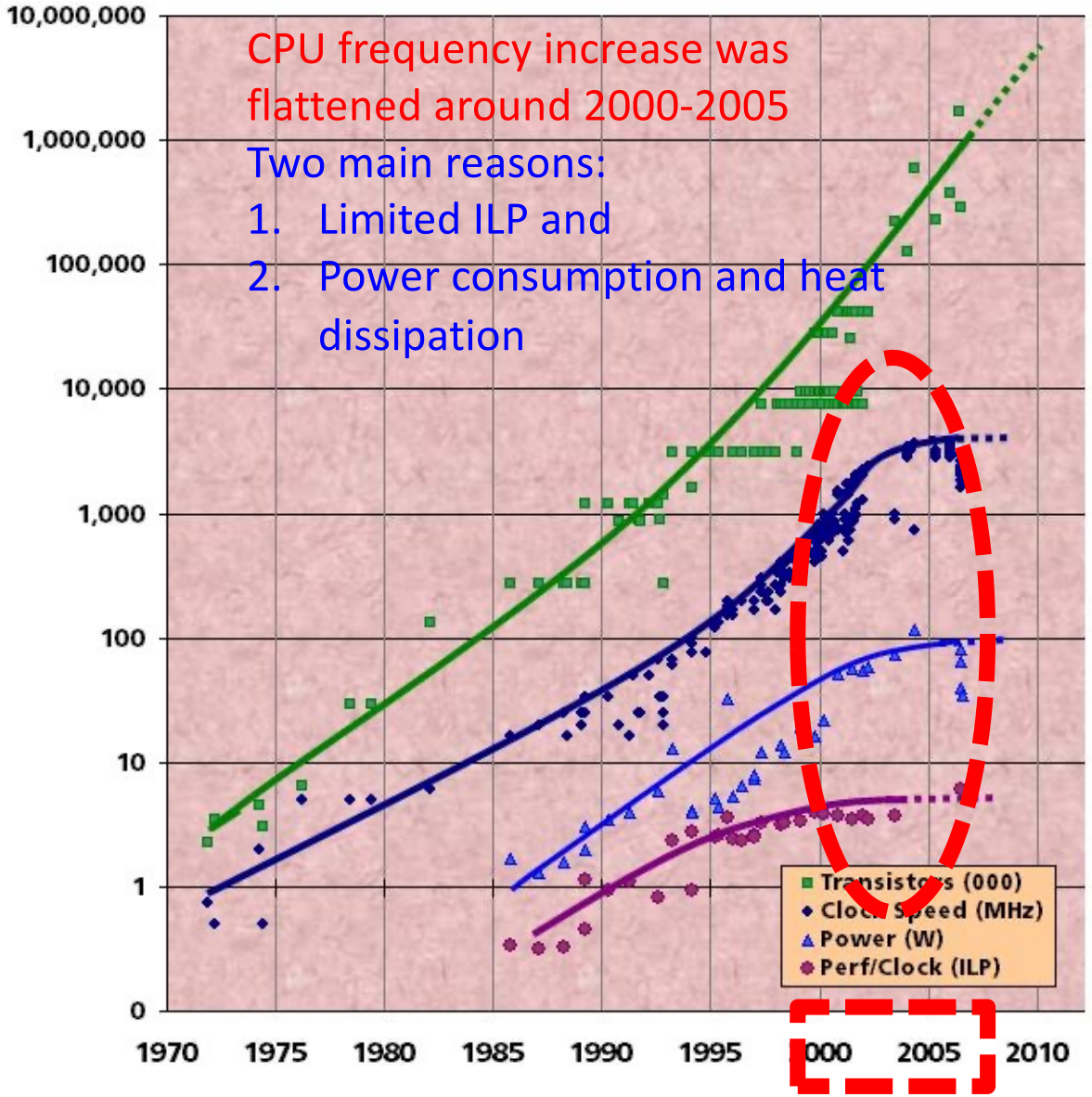


Constrained by power, instruction-level parallelism and memory latency

Moore's Law:

Transistor Density, Frequency, and Multi-cores

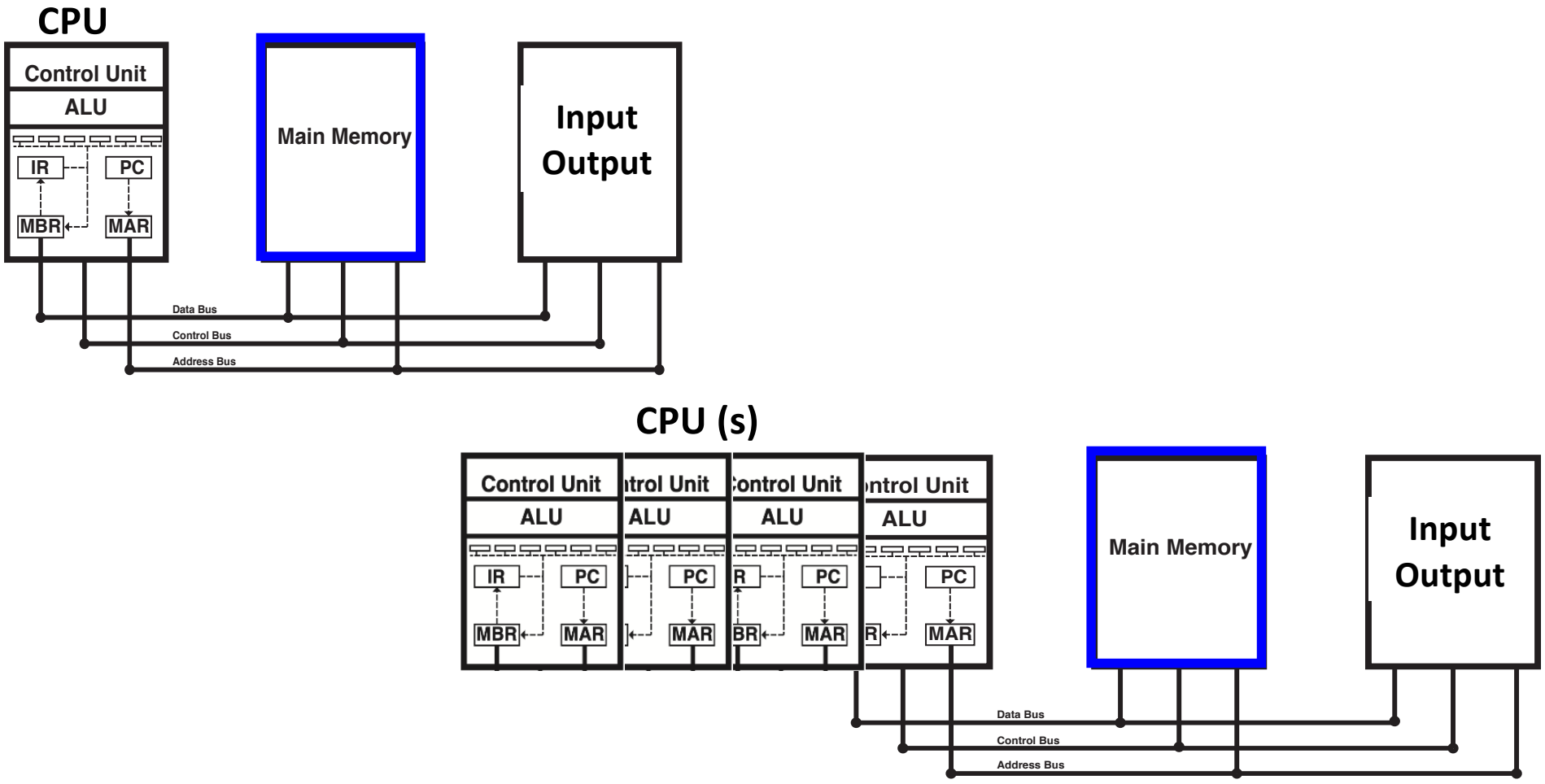
- Moore's Law to processor speed (frequency)



- From 1970s, transistor density doubles every 1.5-2 years till today, but slowing down
  - Latest: 5nm (https://en.wikipedia.org/wiki/7_nm_process)
- CPU frequency (performance) doubles every 2 years till ~2005
- From 2005, transistor density still double every 1.5-2 years, CPU frequency flats
  - Industry moves to multicore, manycore and multiprocessing

Multi-cores or Multi-CPUs

- We cannot increase speed of the CPU, but we can add in a computer more cores or CPUs of the same speed, ~ 2005



History – Past (2000) and Today

The turning away from the conventional organization came in the middle 1960s, when the law of diminishing returns began to take effect in the effort to increase the operational speed of a computer. . . . Electronic circuits are ultimately limited in their speed of operation by the speed of light . . . and many of the circuits were already operating in the nanosecond range.

W. Jack Bouknight et al.
The Illiac IV System (1972)

We are dedicating all of our future product development to multicore designs. We believe this is a key inflection point for the industry.

Intel President Paul Otellini,
describing Intel's future direction at the Intel Developer Forum in 2005

Recent multicore processors

- Sept 13: Intel Ivy Bridge-EP Xeon E5-2695 v2
  - — 12 cores; 2-way SMT; 30MB cache
- March 13: SPARC T5
  - — 16 cores; 8-way fine-grain MT per core
- May 12: AMD Trinity
  - — 4 CPU cores; 384 graphics cores
- Nov 12: Intel Xeon Phi coprocessor
  - — ~60 cores
- Feb 12: Blue Gene/Q
  - — 17 cores; 4-way SMT
- Q4 11: Intel Ivy Bridge
  - — 4 cores; 2 way SMT;
- November 11: AMD Interlagos
  - — 16 cores
- Jan 10: IBM Power 7
  - — 8 cores; 4-way SMT; 32MB shared cache
- Tilera TilePro64

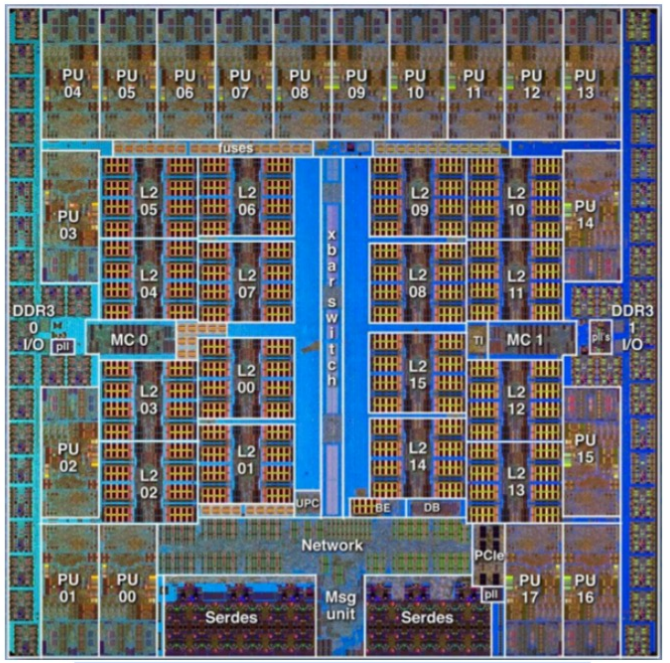
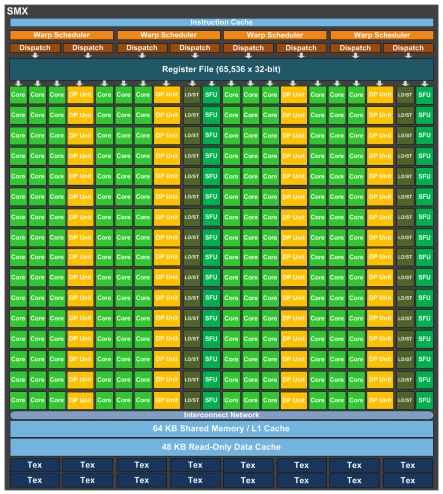


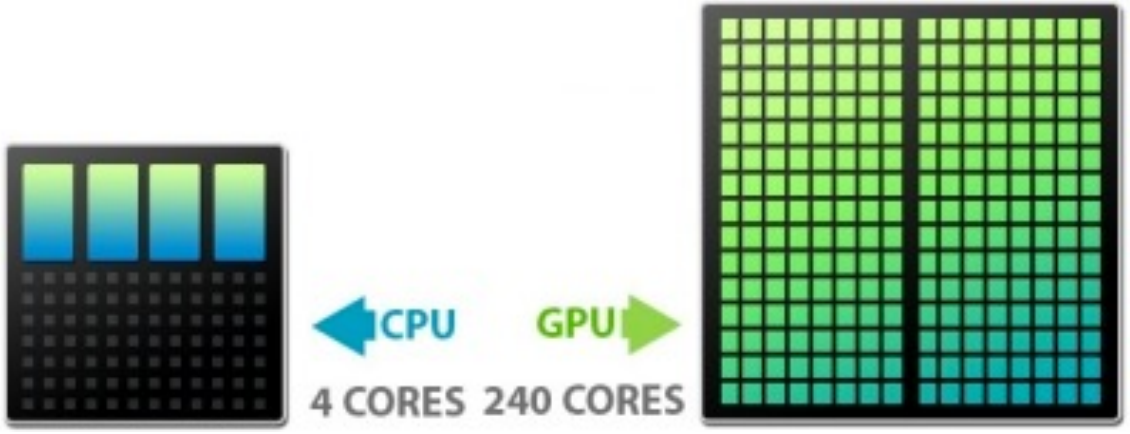
Figure credit: Ruud Haring, Blue Gene/Q compute chip, Hot Chips 23, August, 2011.

Manycore Graphical Processing Unit (GPU) Processors

- ~3000 - 5000 cores of NVIDIA GPUs



SMX: 192 single-precision CUDA cores, 64 double-precision units, 32 special function units (SFU), and 32 load/store units (LD/ST).

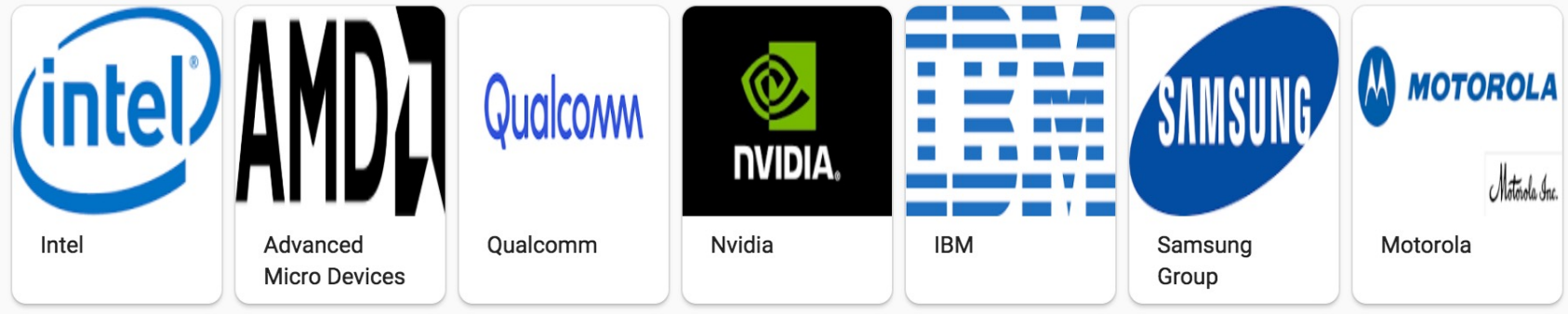


https://www.nvidia.com/en-us/data-center/tesla-p100/

Multiprocessors

- Multicore microprocessors
  - More than one processor per chip
  - Each executes its own instruction sequence
- **Requires explicitly parallel programming**
  - Compare with instruction level parallelism
    - Hardware executes multiple instructions at once
    - Hidden from the programmer
  - Hard to do
    - Programming for performance
    - Load balancing
    - Optimizing communication and synchronization

Major CPU Companies



- Intel, X86 CPUs for desktop, laptop, servers, etc
  - Dell/Apple/HP, etc. Desktop Intel market ~80%, server >98% in 2018
  - Intel will do GPUs based on news from 2019
- AMD, X86 CPU for desktop, laptop, XBOX and PS4
  - The rest of X86 market for desktop and server
  - GPU as well
- Qualcom, Samsung, etc, mostly ARM-based in mobile domain
  - ARM based CPU for smartphone and smartpad (not iphone)
  - Cray, ARM and Fujitsu are building ARM-based server and supercomputers
- IBM, power-based server CPU
  - Server and supercomputers, #1 in top500
    - Summit - IBM Power System, https://www.top500.org/system/179397
- NVIDIA, GPU and ARM-based for mobile
  - #1 in GPU market for graphics, high performance computing and machine learning
- Others
  - Oracle and Fujitsu for SPARC CPU
  - TI, Motorola and Freescale for ARM/power CPU for embedded
  - MIPS CPU vendors

About Supercomputer and HPC

- What Is A Supercomputer?
  - https://www.youtube.com/watch?v=utsi6h7IFPs
- TOP500: https://top500.org/lists/top500/2020/11/

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442,010.0	537,212.0	29,899
2	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148,600.0	200,794.9	10,096
3	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438
4	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway, NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371

Making Use of Multicore CPUs and Multiprocessor Computers

- Adding more processors doesn't help much if programmers aren't aware of them…
  - … or don't know how to use them.
- Serial programs don't benefit from this approach (in most cases).

- **Free lunch of performance provided by Moore's Law is over!**
  - **Do parallel computing ☺**
    - **Check https://passlab.github.io/CSCE569/**

SPEC CPU Benchmark

- Programs used to measure performance
  - Supposedly typical of actual workload
- Standard Performance Evaluation Corp (SPEC)
  - Develops benchmarks for CPU, I/O, Web, …
- SPEC CPU2006
  - Elapsed time to execute a selection of programs
    - Negligible I/O, so focuses on CPU performance
  - Normalize relative to reference machine
  - Summarize as geometric mean of performance ratios
    - CINT2006 (integer) and CFP2006 (floating-point)

$$\sqrt[n]{\prod_{i=1}^{n} \text{Execution time ratio}_i}$$

CINT2006 for Intel Core i7 920

Description	Name	Instruction Count x 10^9	CPI	Clock cycle time (seconds x 10^{-9})	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Interpreted string processing	perl	2252	0.60	0.376	508	9770	19.2
Block-sorting compression	bzip2	2390	0.70	0.376	629	9650	15.4
GNU C compiler	gcc	794	1.20	0.376	358	8050	22.5
Combinatorial optimization	mcf	221	2.66	0.376	221	9120	41.2
Go game (AI)	go	1274	1.10	0.376	527	10490	19.9
Search gene sequence	hmmer	2616	0.60	0.376	590	9330	15.8
Chess game (AI)	sjeng	1948	0.80	0.376	586	12100	20.7
Quantum computer simulation	libquantum	659	0.44	0.376	109	20720	190.0
Video compression	h264avc	3793	0.50	0.376	713	22130	31.0
Discrete event simulation library	omnetpp	367	2.10	0.376	290	6250	21.5
Games/path finding	astar	1250	1.00	0.376	470	7020	14.9
XML parsing	xalancbmk	1045	0.70	0.376	275	6900	25.1
Geometric mean	–	–	–	–	–	–	25.7

SPEC Power Benchmark

- Power consumption of server at different workload levels
  - Performance: ssj_ops/sec
  - Power: Watts (Joules/sec)

$$\text{Overall ssj\_ops per Watt} = \left(\sum_{i=0}^{10} \text{ssj\_ops}_i\right) \Big/ \left(\sum_{i=0}^{10} \text{power}_i\right)$$

ssj: Server Side Java
https://www.spec.org/power/docs/SPECpower_ssj2008-Design_ssj.pdf

SPECpower_ssj2008 for Xeon X5650

Target Load %	Performance (ssj_ops)	Average Power (Watts)
100%	865,618	258
90%	786,688	242
80%	698,051	224
70%	607,826	204
60%	521,391	185
50%	436,757	170
40%	345,919	157
30%	262,071	146
20%	176,061	135
10%	86,784	121
0%	0	80
Overall Sum	4,787,166	1,922
Σssj_ops/Σpower =		2,490

Pitfall: Amdahl's Law

- Improving an aspect of a computer and expecting a proportional improvement in overall performance

$$T_{improved} = \frac{T_{affected}}{improvement\ factor} + T_{unaffected}$$

  - Example: multiply accounts for 80s/100s
    - How much improvement in multiply performance to get 5× overall?

$$20 = \frac{80}{n} + 20$$

      - Can't be done!
  - Corollary: make the common case fast

Concluding Remarks

- Cost/performance is improving
  - Due to underlying technology development
- Hierarchical layers of abstraction
  - In both hardware and software
- Instruction set architecture
  - The hardware/software interface
- Execution time: the best performance measure
- Power is a limiting factor
  - Use parallelism to improve performance

Vision and Wisdom by Experts

- "I think there is a world market for maybe five computers."
  - Thomas Watson, chairman of IBM, 1943.
- "There is no reason for any individual to have a computer in their home"
  - Ken Olson, president and founder of Digital Equipment Corporation, 1977.
- "640K [of memory] ought to be enough for anybody."
  - Bill Gates, chairman of Microsoft,1981.
- "On several recent occasions, I have been asked whether parallel computing will soon be relegated to the trash heap reserved for promising technologies that never quite make it."
  - Ken Kennedy, CRPC Directory, 1994

Linus: The Whole "Parallel Computing Is The Future" Is A Bunch Of Crock.

WEDNESDAY, DECEMBER 31, 2014 AT 9:09AM

http://highscalability.com/blog/2014/12/31/linus-the-whole-parallel-computing-is-the-future-is-a-bunch.html

End of Chapter 01

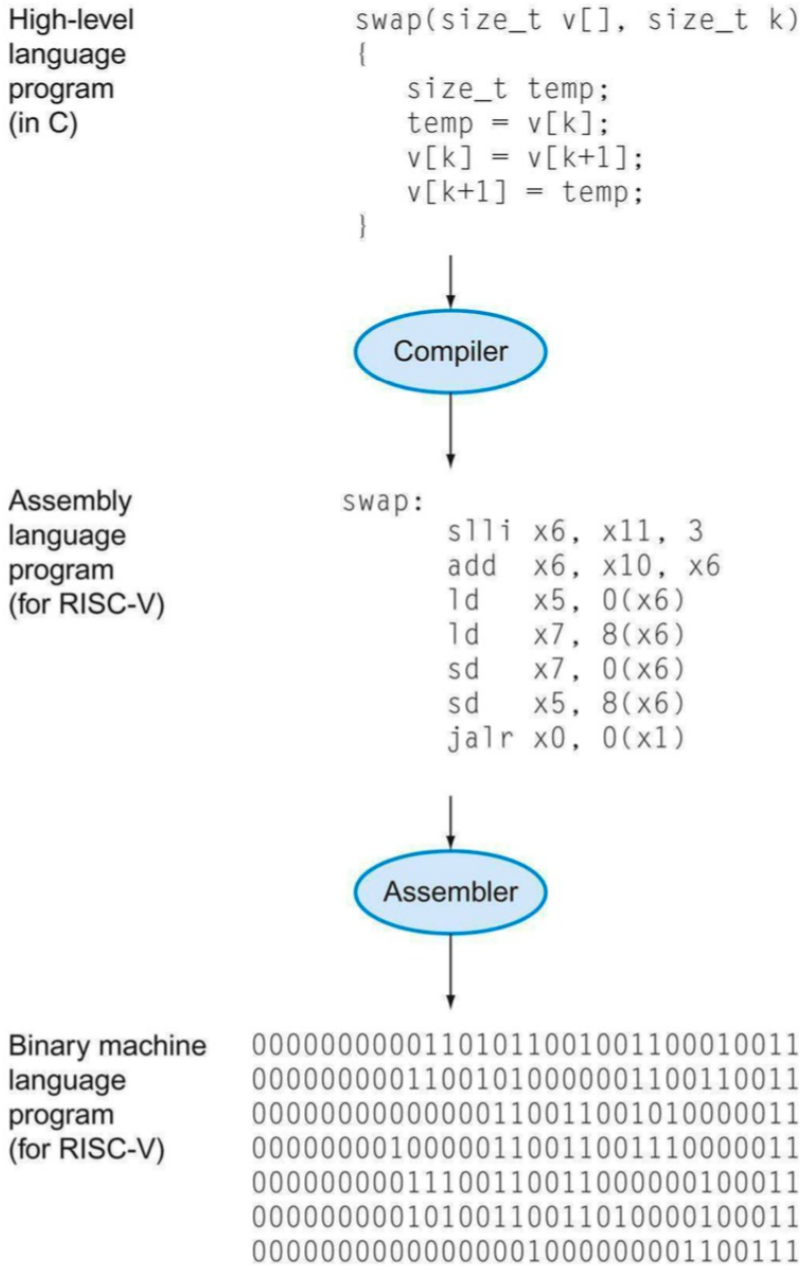
Review for Chapter 1: Three Most Important Topics

- Moore's Law:
  - From 1970s, transistor density doubles every 1.5-2 years till today, but slowing down
    - Latest: 5nm (https://en.wikipedia.org/wiki/7_nm_process)
  - CPU frequency (performance) doubles every 2 years till ~2005
  - From 2005, transistor density still double every 1.5-2 years, CPU frequency flats
    - Industry moves to multicore, manycore and multiprocessing
- Abstraction
  - High-level language, assembly language/ISA as HW/SW interface, binary for computer
  - Software interface, method declaration and definition
- CPU Performance

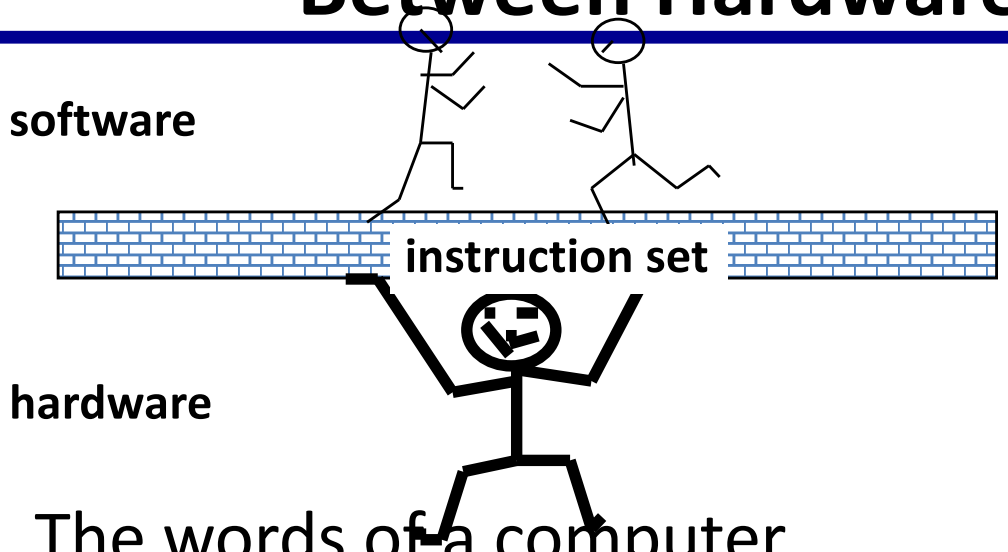
- Power: Linearly proportional to the CPU frequency

Levels of Program Code
Another Great Idea: Abstraction

- High-level language
  - Level of abstraction closer to problem domain
  - Provides for productivity and portability
- Assembly language
  - Textual representation of **instructions**
- Hardware representation
  - Binary digits (bits)
  - Encoded instructions and data



Instruction Set Architecture: The Interface Between Hardware and Software



- The words of a computer language are called *instructions*, and its vocabulary/dictionary is called an *instruction set*
  - lowest software interface, assembly level, to the users or to the compiler writer

Instruction Set Architecture – A type of machine

A language represents a race

High-level language program (in C)

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

Compiler

Assembly language program (for RISC-V)

```
swap:
        slli x6, x11, 3
        add  x6, x10, x6
        ld   x5, 0(x6)
        ld   x7, 8(x6)
        sd   x7, 0(x6)
        sd   x5, 8(x6)
        jalr x0, 0(x1)
```

Assembler

Binary machine language program (for RISC-V)

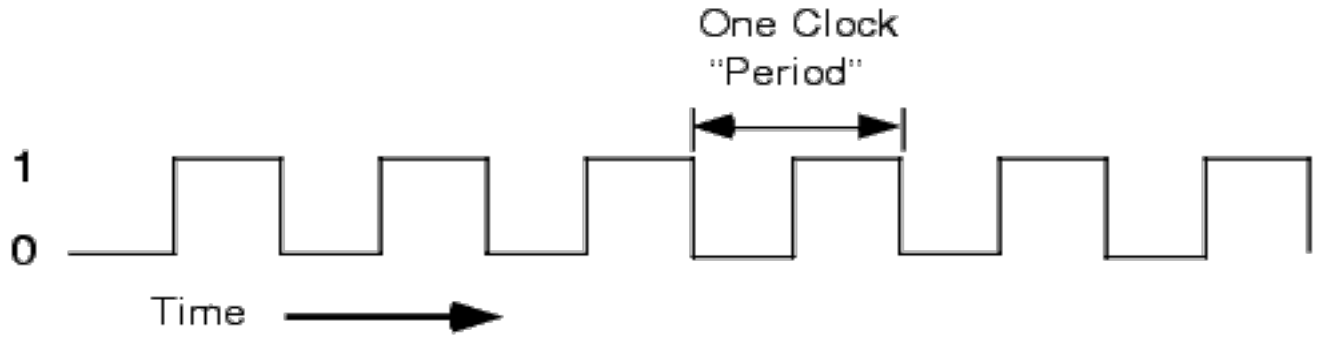
```
00000000001101011001001100010011
00000000011001010000001100110011
00000000000000011001001010000011
00000000100000011001001110000011
00000000011100110011000000100011
00000000010100110011010000100011
00000000000000001000000001100111
```

Review: CPU Time

- Performance improved by
  - Reducing number of clock cycles
  - Increasing clock rate
  - Hardware designer must often trade off clock rate against cycle count

$$\text{CPU Time} = \text{CPU Clock Cycles} \times \text{Clock Cycle Time}$$

$$= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}$$



Review: CPU Time, Instruction Count and CPI

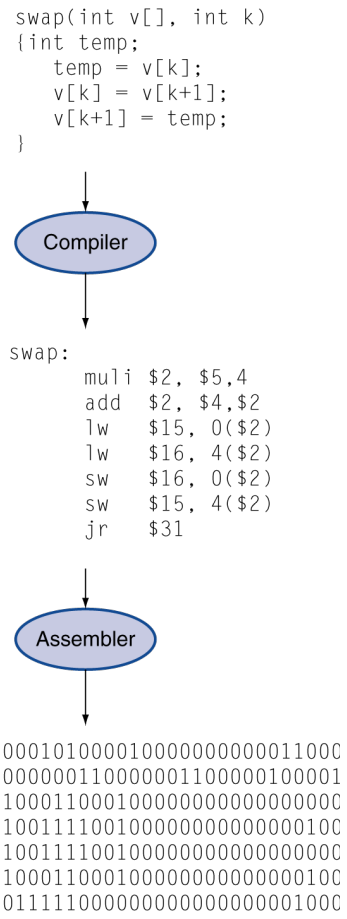
- Hardware/CPU executes a program instruction by instructions

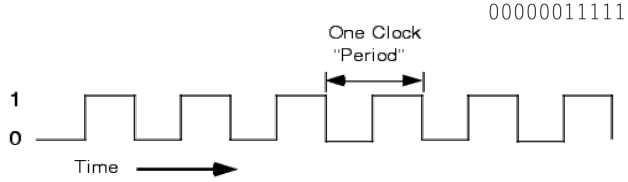
$$\text{Clock Cycles} = \text{Instruction Count} \times \text{Cycles per Instruction}$$

$$\text{CPU Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

$$= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

- Instruction Count for a program
  - Determined by program, ISA and compiler
- Average cycles per instruction
  - Determined by CPU hardware
  - If different instructions have different CPI
    - Average CPI affected by instruction mix





Review: CPI Example

- Alternative compiled code sequences using instructions in classes A, B, C

$$\text{Clock Cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{Instruction Count}_i)$$

Class	A	B	C
CPI for class	1	2	3
IC in sequence 1	2	1	2
IC in sequence 2	4	1	1

- Sequence 1: IC = 5
  - Clock Cycles
    = 2×1 + 1×2 + 2×3
    = 10
  - Avg. CPI = 10/5 = 2.0
- Sequence 2: IC = 6
  - Clock Cycles
    = 4×1 + 1×2 + 1×3
    = 9
  - Avg. CPI = 9/6 = 1.5

End of Review of Chapter 01