Lecture 02: Technology Trends and Quantitative Design and Analysis for Performance

CSCE 513 Computer Architecture

Department of Computer Science and Engineering
Yonghong Yan
yanyh@cse.sc.edu
http://cse.sc.edu/~yanyh

Contents

- Computer components
- Computer architectures and great ideas in computer architectures
- **Trends and Performance**

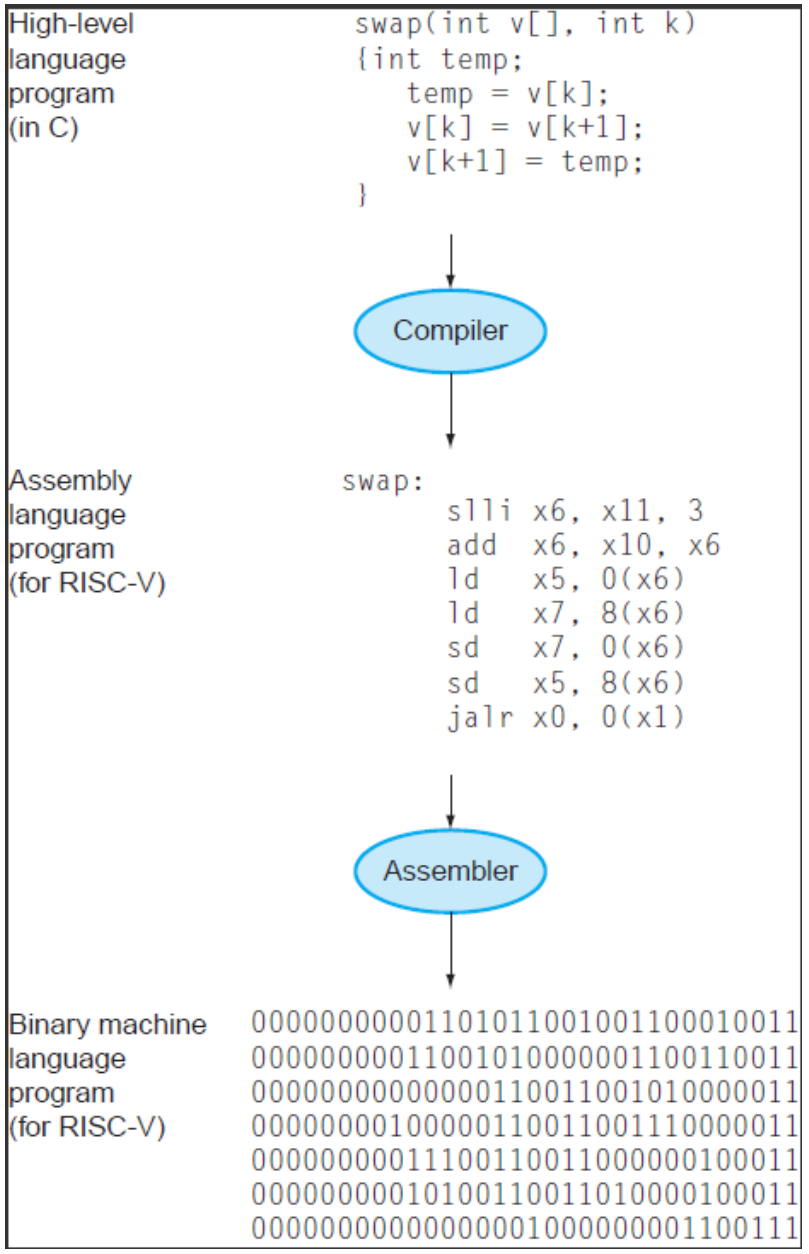
Computer Architecture

Genuine Computer Architecture: Designing the Organization and Hardware to Meet Goals and Functional Requirements

- Covers three aspects of computer design
  - Instruction set architecture
    - Software and hardware interfaces
  - Organization or microarchitecture
    - CPU, memory, cache architecture
  - Hardware
    - Computer systems, e.g. I/O devices

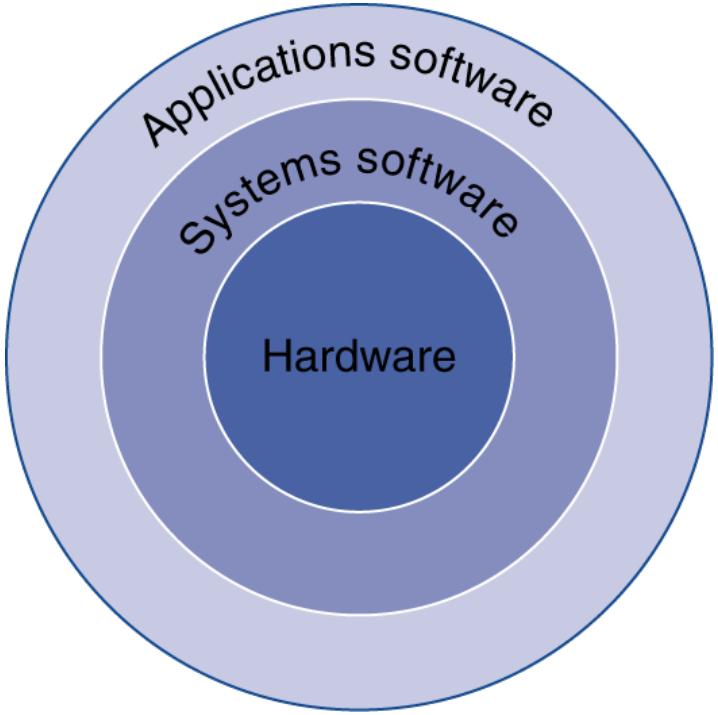
Levels of Program Code

- 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



Below Your Program

- Application software
  - Written in high-level language
- System software
  - Compiler: translates HLL code to machine code
  - Operating System: service code
    - Handling input/output
    - Managing memory and storage
    - Scheduling tasks & sharing resources
- Hardware
  - Processor, memory, I/O controllers



Understanding Performance

- Algorithm
  - Determines number of operations executed
- Programming language, compiler, architecture
  - Determine number of machine instructions executed per operation
- **Processor and memory system**
  - **Determine how fast instructions are executed**
- I/O system (including OS)
  - Determines how fast I/O operations are executed

- **Architecture vs Technology**

Trends in Technology

- Integrated circuit technology (Moore's Law)
  - Transistor density: 35%/year
  - Die size: 10-20%/year
  - Integration overall: 40-55%/year
- DRAM capacity: 25-40%/year (slowing)
  - 8 Gb (2014), 16 Gb (2019), possibly no 32 Gb
- Flash capacity: 50-60%/year
  - 8-10X cheaper/bit than DRAM
- Magnetic disk capacity: recently slowed to 5%/year
  - Density increases may no longer be possible, maybe increase from 7 to 9 platters
  - 8-10X cheaper/bit then Flash
  - 200-300X cheaper/bit than DRAM

Bandwidth and Latency

- Bandwidth or throughput
  - Total work done in a given time
  - 10,000-25,000X improvement for processors
  - 300-1200X improvement for memory and disks

- Latency or response time
  - Time between start and completion of an event
  - 30-80X improvement for processors
  - 6-8X improvement for memory and disks

Measuring Performance

- Typical performance metrics:
  - Response time
  - Throughput
- Speedup of X relative to Y: Execution timeY / Execution timeX
  - Example: time taken to run a program, 10s on X, 15s on Y
  - Speedup: 15s/10s = 1.5, → X is 1.5 faster than Y
- **Execution time**
  - **Wall clock time: includes all system overheads (I/O, swapping, etc)**
  - **CPU time: only computation time**
- Benchmarks
  - Kernels (e.g. matrix multiply)
  - Toy programs (e.g. sorting)
  - Synthetic benchmarks (e.g. Dhrystone)
  - Benchmark suites (e.g. SPEC06fp, TPC-C)

Measuring Execution Time 1/2

- Elapsed time
  - Total response time, including all aspects
    - Processing, I/O, OS overhead, idle time
  - Determines system performance

```
elapsed = read_timer();
REAL result = sum(N, X, a);
elapsed = (read_timer() - elapsed);
```

https://passlab.github.io/CSCE513/exercises/sum/sum_full.c

- CPU time

Measuring Execution Time 2/2

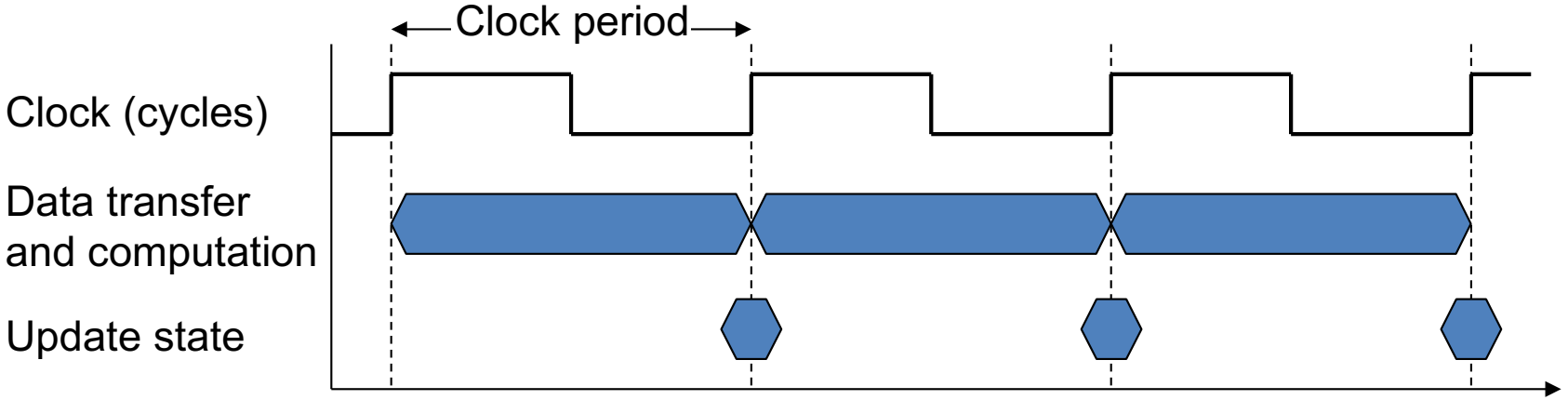
- Elapsed time
- CPU time
  - Time spent processing a given job
    - Discounts I/O time, other jobs' shares
  - Comprises user CPU time and system CPU time
  - Different programs are affected differently by CPU and system
  - "time" command in Linux

```
yanyh@vm:~$ time ./matmul 512 1
================================================================
Matrix Multiplication: A[M][K] * B[k][N] = C[M][N], M=K=N=512,
----------------------------------------------------------------
Performance:              Runtime (ms)     MFLOPS
----------------------------------------------------------------
matmul_base:              628.999949       426.765466
matmul_openmp:            776.000023       345.921969

real    0m1.419s
user    0m1.408s
sys     0m0.008s
```

CPU Clocking

- Operation of digital hardware governed by a constant-rate clock



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

No Excuse About the Unit

- **Should be as clear as we know about thousand/million/billion dollars**

10^{-3} s	**ms**	**millisecond**
10^{-6} s	***μs***	**microsecond**
10^{-9} s	**ns**	**nanosecond**
10^{-12} s	ps	picosecond

10^{3} Hz	**kHz**	**kilohertz**
10^{6} Hz	**MHz**	**megahertz**
10^{9} Hz	**GHz**	**gigahertz**

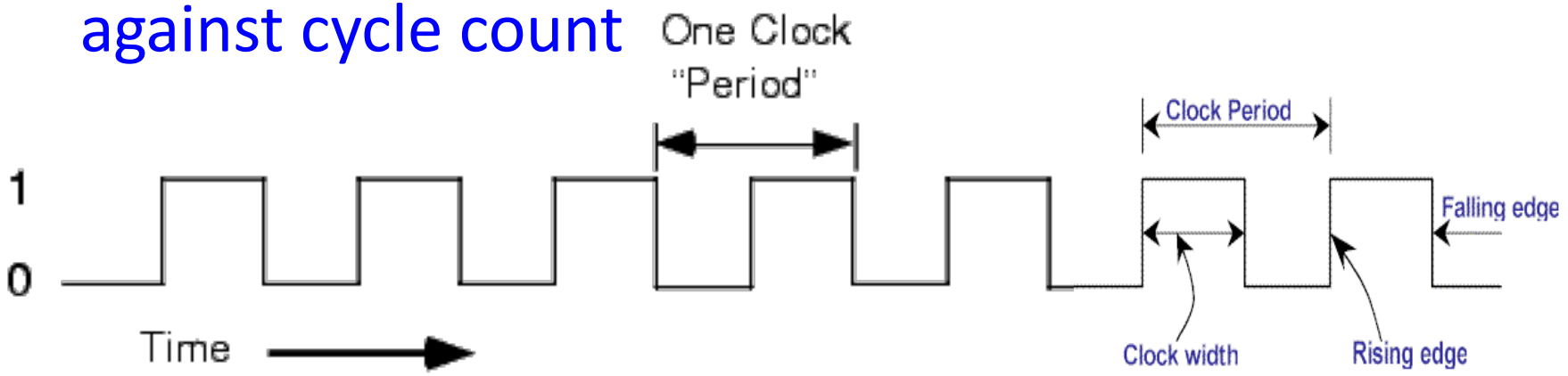
Decimal			Binary		
Value	Metric		Value	IEC	
1000	kB	**kilobyte**	1024	KiB	kibibyte
1000^2	MB	megabyte	1024^2	MiB	mebibyte
1000^3	GB	gigabyte	1024^3	GiB	gibibyte
1000^4	TB	terabyte	1024^4	TiB	tebibyte
1000^5	PB	petabyte	1024^5	PiB	pebibyte
1000^6	EB	exabyte	1024^6	EiB	exbibyte

CPU Time

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

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

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



CPU Time Example

- Computer A: 2GHz clock, 10s CPU time
- Designing Computer B
  - Aim for 6s CPU time
  - Can do faster clock, but causes 1.2 × clock cycles of A
- How fast must Computer B clock be?

$$\text{Clock Rate}_B = \frac{\text{Clock Cycles}_B}{\text{CPU Time}_B} = \frac{1.2 \times \text{Clock Cycles}_A}{6\text{s}}$$

$$\text{Clock Cycles}_A = \text{CPU Time}_A \times \text{Clock Rate}_A$$

$$= 10\text{s} \times 2\text{GHz} = 20 \times 10^9$$

$$\text{Clock Rate}_B = \frac{1.2 \times 20 \times 10^9}{6\text{s}} = \frac{24 \times 10^9}{6\text{s}} = 4\text{GHz}$$

Instruction Count and CPI

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

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

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

- 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

Instr. No.	Pipeline Stage						
1	IF	ID	EX	MEM	WB		
2		IF	ID	EX	MEM	WB	
3			IF	ID	EX	MEM	WB
4				IF	ID	EX	MEM
5					IF	ID	EX
Clock Cycle	1	2	3	4	5	6	7

CPI Example

- Computer A: Cycle Time = 250ps, CPI = 2.0
- Computer B: Cycle Time = 500ps, CPI = 1.2
- Same ISA
- Which is faster, and by how much?

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

$$= I \times 2.0 \times 250\text{ps} = I \times 500\text{ps}$$

A is faster…

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

$$= I \times 1.2 \times 500\text{ps} = I \times 600\text{ps}$$

$$\frac{\text{CPU Time}_B}{\text{CPU Time}_A} = \frac{I \times 600\text{ps}}{I \times 500\text{ps}} = 1.2$$

…by this much

CPI in More Detail

- If different instruction classes take different numbers of cycles

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

- Weighted average CPI

$$\text{CPI} = \frac{\text{Clock Cycles}}{\text{Instruction Count}} = \sum_{i=1}^{n} \left(\text{CPI}_i \times \underbrace{\frac{\text{Instruction Count}_i}{\text{Instruction Count}}}_{\text{Relative frequency}} \right)$$

CPI Example

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

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

Impacts by Components

$$\text{CPU Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

	Inst Count	CPI	Clock Rate
Program	X		
Compiler	X	(X)	
Inst. Set.	X	X	
Architecture	X		X
Technology			X

Processor Performance Equation Summary

$$\text{CPU time} = \text{CPU clock cycles for a program} \times \text{Clock cycle time}$$

$$\text{CPU time} = \frac{\text{CPU clock cycles for a program}}{\text{Clock rate}}$$

$$\text{CPI} = \frac{\text{CPU clock cycles for a program}}{\text{Instruction count}}$$

$$\text{CPU time} = \text{Instruction count} \times \text{Cycles per instruction} \times \text{Clock cycle time}$$

$$\frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}} = \frac{\text{Seconds}}{\text{Program}} = \text{CPU time}$$

$$\text{CPU clock cycles} = \sum_{i=1}^{n} \text{IC}_i \times \text{CPI}_i$$

$$\text{CPU time} = \left(\sum_{i=1}^{n} \text{IC}_i \times \text{CPI}_i \right) \times \text{Clock cycle time}$$

Principles of Computer Design

- Take Advantage of Parallelism
  - e.g. multiple processors, disks, memory banks, pipelining, multiple functional units
- Principle of Locality
  - Reuse of data and instructions
- Focus on the Common Case
  - Amdahl's Law

$$\text{Execution time}_{\text{new}} = \text{Execution time}_{\text{old}} \times \left((1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}} \right)$$

$$\text{Speedup}_{\text{overall}} = \frac{\text{Execution time}_{\text{old}}}{\text{Execution time}_{\text{new}}} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

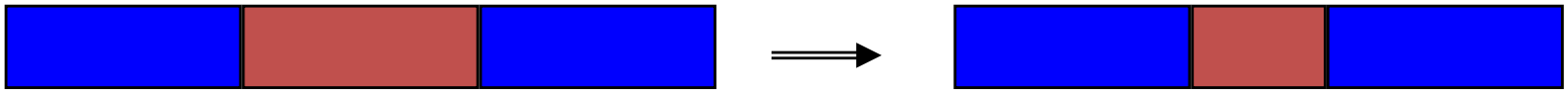
Amdahl's Law

$$ExTime_{new} = ExTime_{old} \times \left[\left(1 - Fraction_{enhanced}\right) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}} \right]$$

$$Speedup_{overall} = \frac{ExTime_{old}}{ExTime_{new}} = \frac{1}{\left(1 - Fraction_{enhanced}\right) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}}}$$

Best you could ever hope to do:

$$Speedup_{maximum} = \frac{1}{\left(1 - Fraction_{enhanced}\right)}$$



Using Amdahl's Law

Overall speedup if we make 90% of a program run 10 times faster.

$F = 0.9 \quad S = 10$

$$\text{Overall Speedup} = \frac{1}{(1-0.9)+\frac{0.9}{10}} = \frac{1}{0.1+0.09} = 5.26$$

Overall speedup if we make 80% of a program run 20% faster.

$F = 0.8 \quad S = 1.2$

$$\text{Overall Speedup} = \frac{1}{(1-0.8)+\frac{0.8}{1.2}} = \frac{1}{0.2+0.66} = 1.153$$

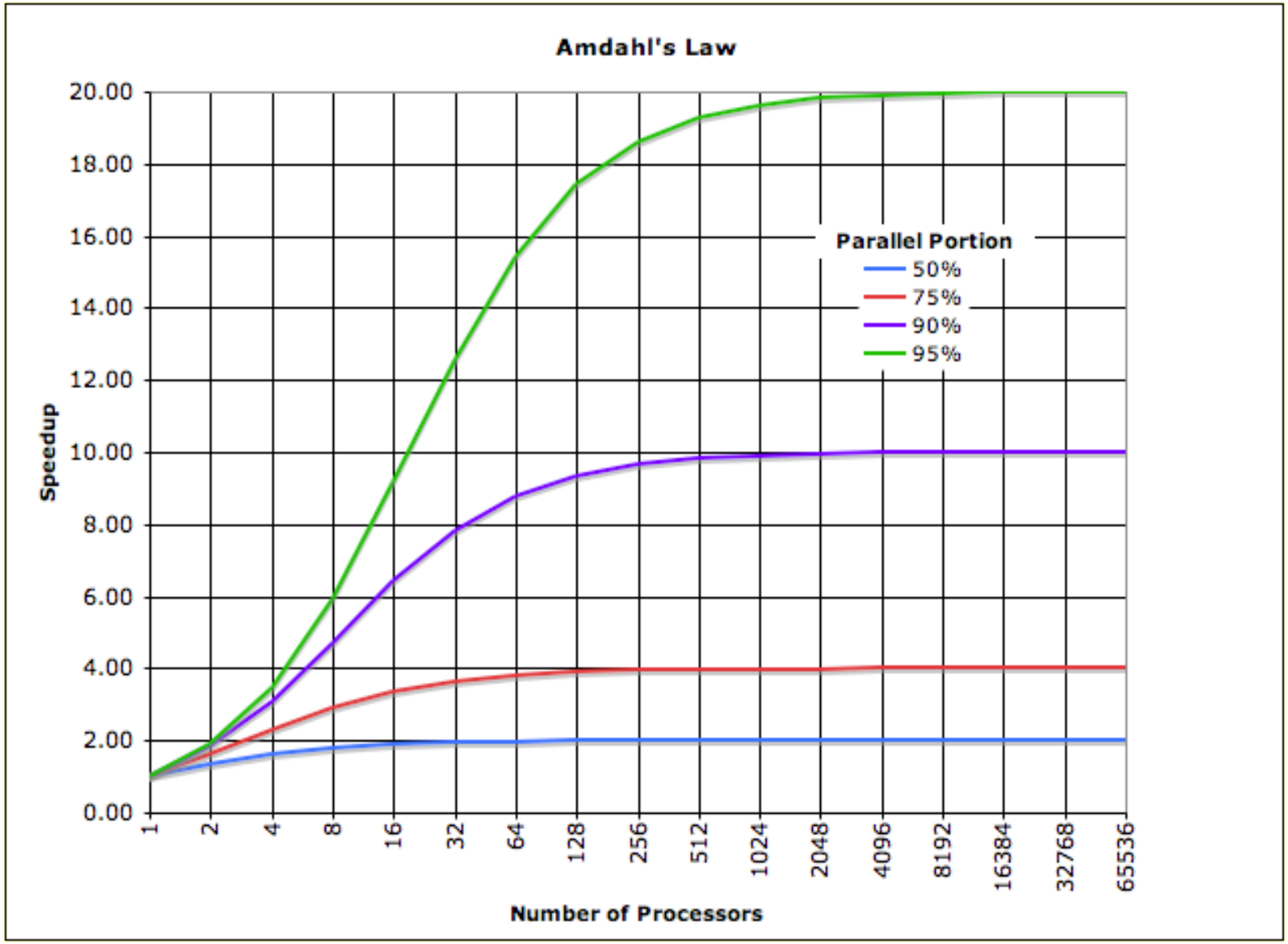
Amdahl's Law for Parallelism

- The enhanced fraction F is through parallelism, perfect parallelism with linear speedup
  - The speedup for F is N for N processors
- Overall speedup

$$S(N) = \frac{T_s}{T_p} = \frac{T_s}{(1-F)*T_s + \frac{F*T_s}{N}} = \frac{1}{1-F+\frac{F}{N}}$$

- Speedup upper bound (when N $\rightarrow \infty$): $S(N) \leq \frac{1}{1-F}$
  - 1-F: the sequential portion of a program

Amdahl's Law for Parallelism



Exercise #1: Amdahl's Law

Suppose that we want to enhance the processor used for Web serving. The new processor is 10 times faster on computation in the Web serving application than the original processor. Assuming that the original processor is busy with computation 40% of the time and is waiting for I/O 60% of the time, what is the overall speedup gained by incorporating the enhancement?

Exercise #1: Amdahl's Law Solution

$$\text{Fraction}_{\text{enhanced}} = 0.4; \text{Speedup}_{\text{enhanced}} = 10;$$

$$\text{Speedup}_{\text{overall}} = \frac{1}{0.6 + \frac{0.4}{10}} = \frac{1}{0.64} \approx 1.56$$

General Amdahl's Law

- F0 30%, no speedup; F1 40%, speedup by 4; F2 30% speedup by 3, what is the overall speedup

- = 1 / (0.3 + 0.4/4 + 0.3/3) = 1/0.5 = 2

Exercise #2: CPU time and Speedup

Suppose we have made the following measurements:

Frequency of FP operations = 25%

Average CPI of FP operations = 4.0

Average CPI of other instructions = 1.33

Frequency of FPSQR = 2%

CPI of FPSQR = 20

Assume that the two design alternatives are to decrease the CPI of FPSQR to 2 or to decrease the average CPI of all FP operations to 2.5. Compare these two design alternatives using the processor performance equation.

Exercise #2: Solution, Textbook Page 54

First, observe that only the CPI changes; the clock rate and instruction count remain identical. We start by finding the original CPI with neither enhancement:

$$\text{CPI}_{\text{original}} = \sum_{i=1}^{n} \text{CPI}_i \times \left(\frac{\text{IC}_i}{\text{Instruction count}} \right)$$

$$= (4 \times 25\%) + (1.33 \times 75\%) = 2.0$$

We can compute the CPI for the enhanced FPSQR by subtracting the cycles saved from the original CPI:

$$\text{CPI}_{\text{with new FPSQR}} = \text{CPI}_{\text{original}} - 2\% \times (\text{CPI}_{\text{old FPSQR}} - \text{CPI}_{\text{of new FPSQR only}})$$

$$= 2.0 - 2\% \times (20 - 2) = 1.64$$

We can compute the CPI for the enhancement of all FP instructions the same way or by summing the FP and non-FP CPIs. Using the latter gives us:

$$\text{CPI}_{\text{new FP}} = (75\% \times 1.33) + (25\% \times 2.5) = 1.625$$

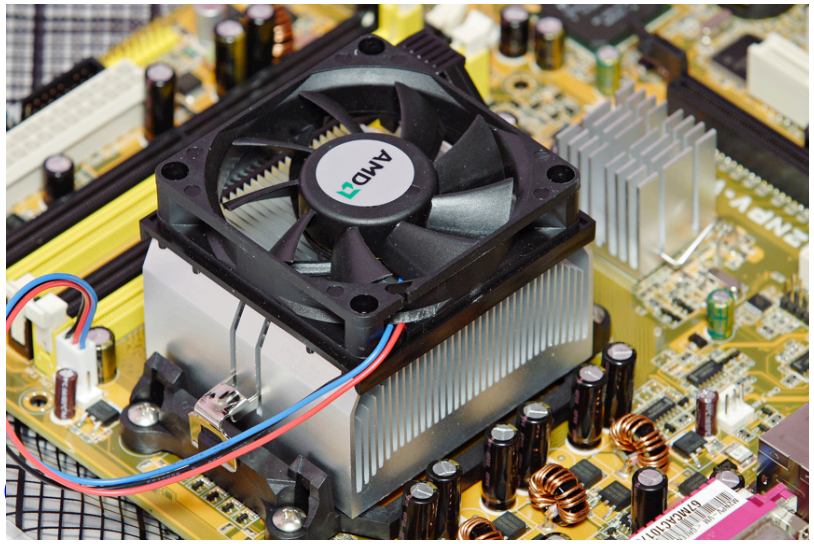
Since the CPI of the overall FP enhancement is slightly lower, its performance will be marginally better. Specifically, the speedup for the overall FP enhancement is

$$\text{Speedup}_{\text{new FP}} = \frac{\text{CPU time}_{\text{original}}}{\text{CPU time}_{\text{new FP}}} = \frac{\text{IC} \times \text{Clock cycle} \times \text{CPI}_{\text{original}}}{\text{IC} \times \text{Clock cycle} \times \text{CPI}_{\text{new FP}}}$$

$$= \frac{\text{CPI}_{\text{original}}}{\text{CPI}_{\text{new FP}}} = \frac{2.00}{1.625} = 1.23$$

Power and Energy

- Problem:
  - Get power in and distribute around
  - get power out: dissipate heat



- Three primary concerns:
  - Max power requirement for a process
  - Thermal Design Power (TDP)
    - Characterizes sustained power consumption
    - Used as target for power supply and cooling system
    - Lower than peak power, higher than average power consumption
  - Energy and energy efficiency

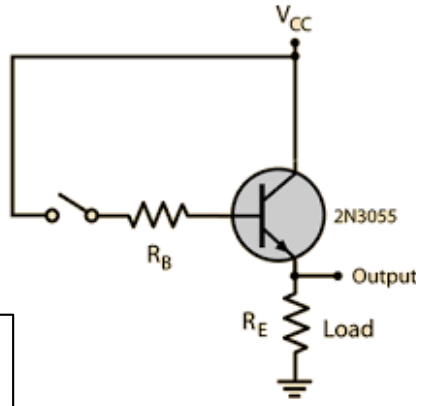
- Clock rate can be reduced dynamically to limit power consumption

Energy and Energy Efficiency

- Power: energy per unit time
  - 1 watt = 1 joule per second
  - Energy per task is often a better measurement

- Processor A has 20% higher average power consumption than processor B. A executes task in only 70% of the time needed by B.
  - So energy consumption of A will be **1.2 * 0.7 = 0.84** of B

Dynamic Energy and Power



- Dynamic energy
  - Transistor switch from 0 -> 1 or 1 -> 0

$$\text{Energy}_{\text{dynamic}} \propto 1/2 \times \text{Capacitive load} \times \text{Voltage}^2$$

- Dynamic power

$$\text{Power}_{\text{dynamic}} \propto 1/2 \times \text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency switched}$$

- Reducing clock rate reduces power, not energy
- The capacitive load:
  - a function of the number of transistors connected to an output and the technology, which determines the capacitance of the wires and the transistors.

An Example from Textbook page #25

Some microprocessors today are designed to have adjustable voltage, so a 15% reduction in voltage may result in a 15% reduction in frequency. What would be the impact on dynamic energy and on dynamic power?

Since the capacitance is unchanged, the answer for energy is the ratio of the voltages since the capacitance is unchanged:

$$\frac{\text{Energy}_{\text{new}}}{\text{Energy}_{\text{old}}} = \frac{(\text{Voltage} \times 0.85)^2}{\text{Voltage}^2} = 0.85^2 = 0.72$$

thereby reducing energy to about 72% of the original. For power, we add the ratio of the frequencies

$$\frac{\text{Power}_{\text{new}}}{\text{Power}_{\text{old}}} = 0.72 \times \frac{(\text{Frequency switched} \times 0.85)}{\text{Frequency switched}} = 0.61$$

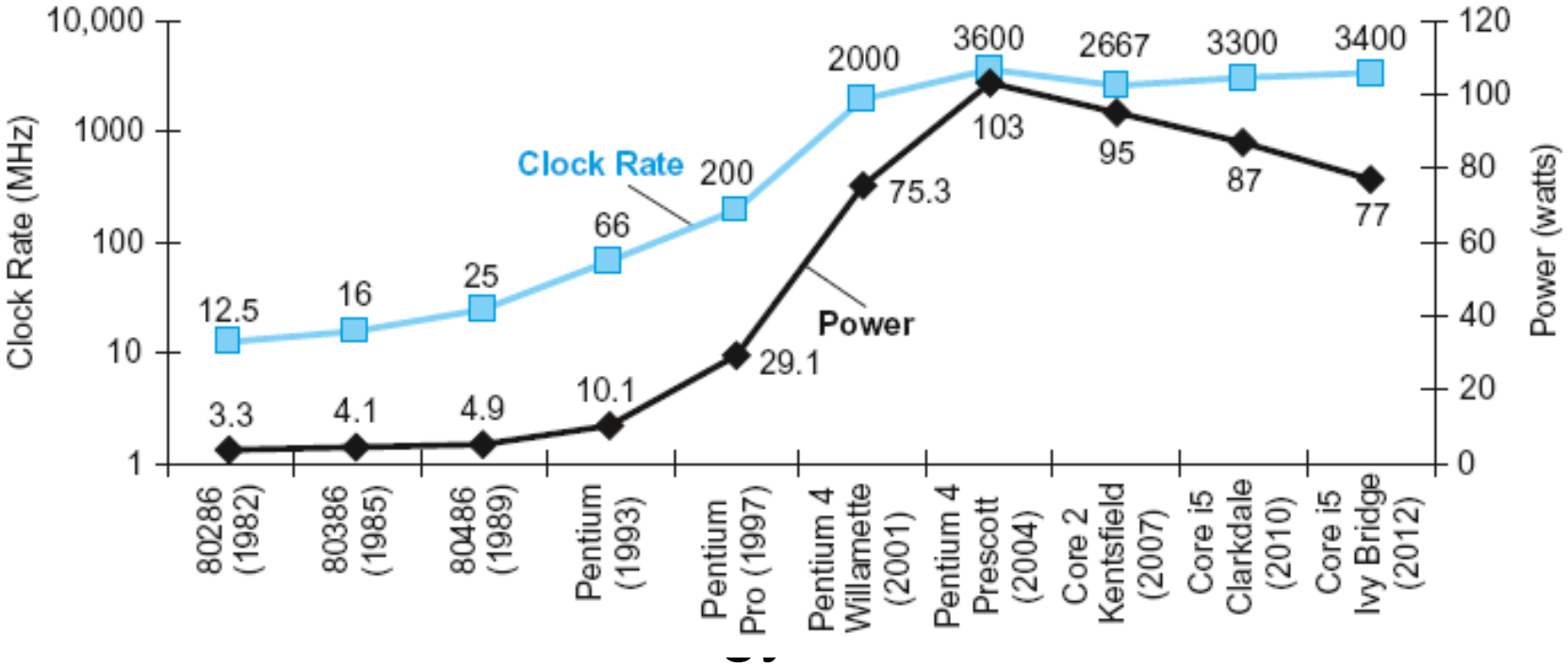
shrinking power to about 61% of the original.

An Example from Textbook

- Suppose a new CPU has
  - **85% of capacitive load of old CPU**
  - 15% voltage and 15% frequency reduction

$$\frac{P_{new}}{P_{old}} = \frac{C_{old} \times 0.85 \times (V_{old} \times 0.85)^2 \times F_{old} \times 0.85}{C_{old} \times V_{old}^{\ 2} \times F_{old}} = 0.85^4 = 0.52$$

Power Trends



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

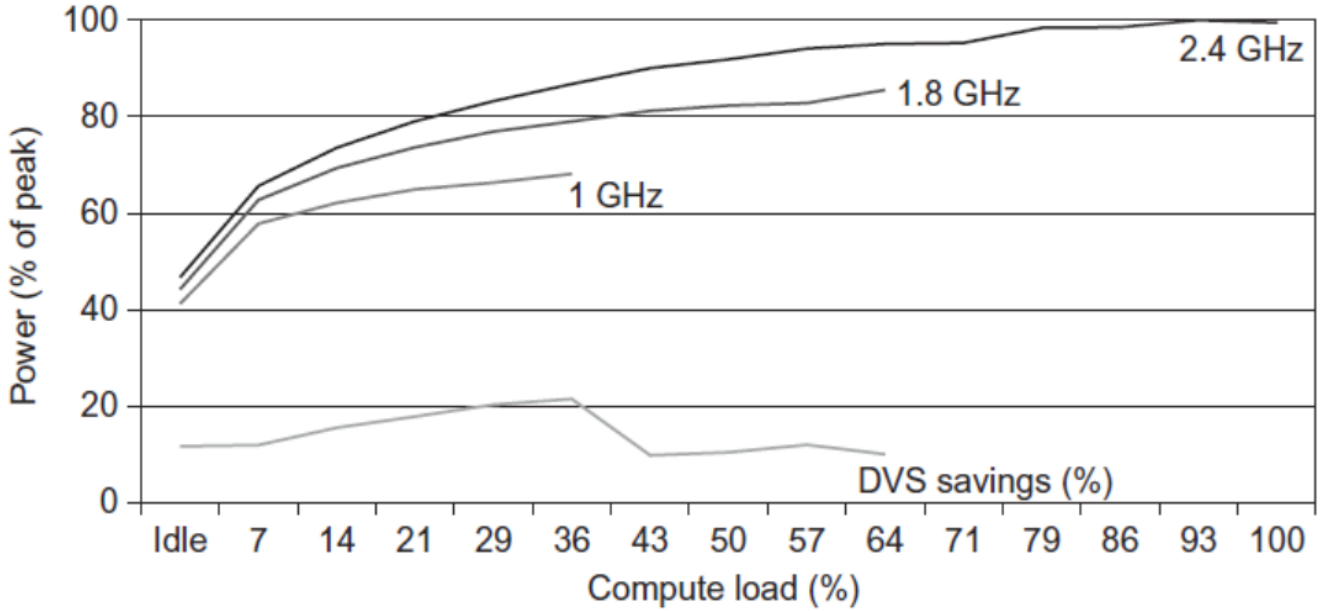
×30

5V → 1V

×1000

Reducing Power

- Techniques for reducing power:
  - Do nothing well
  - Dynamic Voltage-Frequency Scaling



  - Low power state for DRAM, disks
  - Overclocking, turning off cores

Static Power

- Power includes both dynamic power and static power
- Static power consumption
  - 25-50% of total power
  - Scales with number of transistors
  - To reduce: power gating (turn off power of inactive modules)

$$\text{Power}_{\text{static}} \propto \text{Current}_{\text{static}} \times \text{Voltage}$$

Operation:	Energy (pJ)	Area (μm^2)
8b Add	0.03	36
16b Add	0.05	67
32b Add	0.1	137
16b FB Add	0.4	1360
32b FB Add	0.9	4184
8b Mult	0.2	282
32b Mult	3.1	3495
16b FB Mult	1.1	1640
32b FB Mult	3.7	7700
32b SRAM Read (8KB)	5	N/A
32b DRAM Read	640	N/A