"I expect you all to be independent, innovative, critical thinkers who will do exactly as I say!"

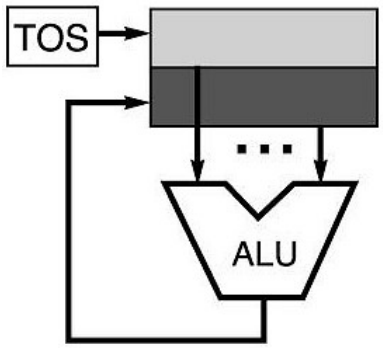
https://s-media-cache-ak0.pinimg.com/236x/e8/c2/5f/e8c25ffa78a9af22cba55729b690657e.jpg

Quiz

1. ISA serves as _____

2. consider stack architecture w/ instruction set:
   - ✓ add, sub, mult, push A, pop A

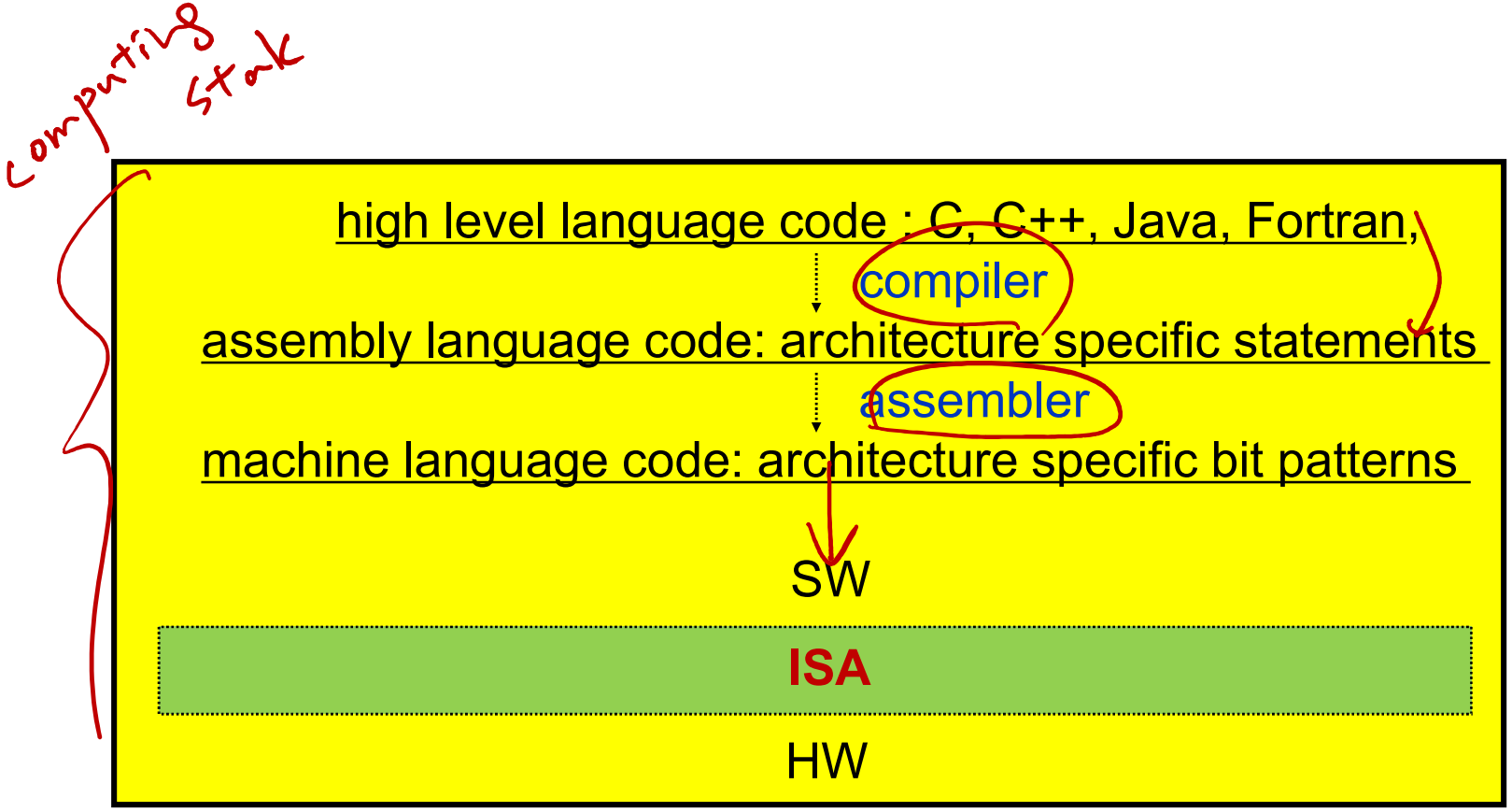
   write assembly code (instruction sequency)
   for the following code: (A+C*B)



Lecture 3:
Performance, Energy, and Power Metric

Review: Instruction Set Architecture (ISA)

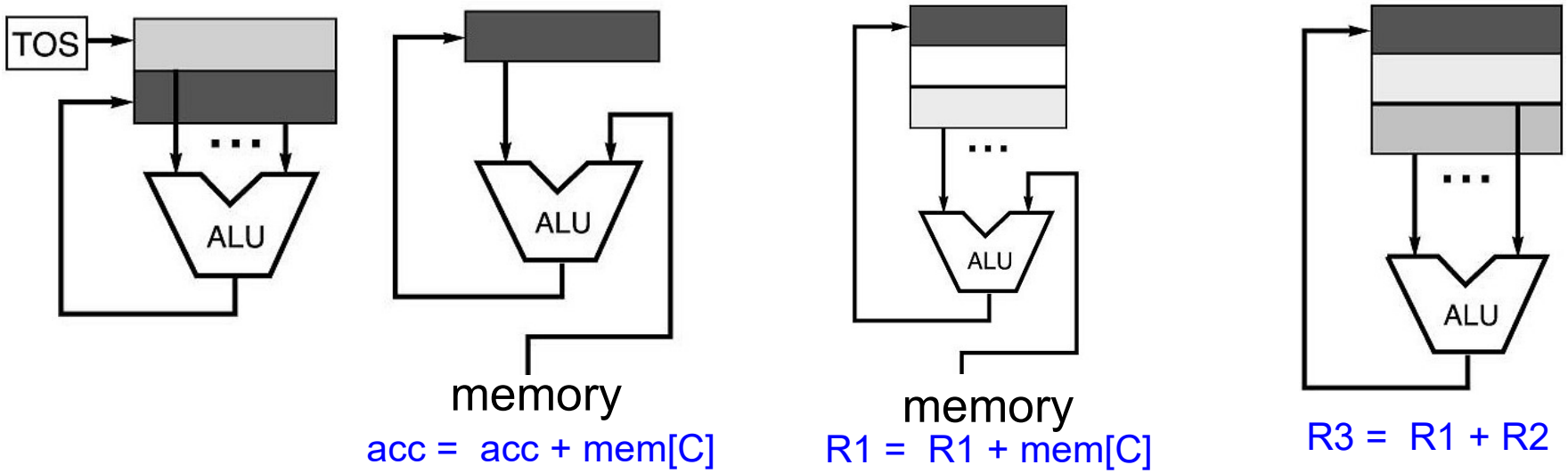
- serves as an interface b/w software and hardware
- provides a mechanism by which the software tells the hardware what should be done



Review: Four Instruction Sets

- code sequence C = A + B

stack	accumulator	register (register-memory)	register (load-store)
push A push B add pop C	load A add B store C	load R1, A add R1, B store C, R1	load R1,A load R2, B add R3, R1, R2 store C, R3



Review: Pros and Cons of ISAs

- accumulator
  - ✓ pros: very low hardware requirements; easy to design and understand
  - ✓ cons: accumulator becomes the bottleneck; little ability for parallelism or pipelining; high memory traffic
- memory-memory
  - ✓ pros: requires fewer instructions; easy to write compilers
  - ✓ cons: very high memory traffic; w/ two operands, more data movements
- memory-register
  - ✓ pros: some data can be accessed without loading first; instruction format easy to encode; good code density
  - ✓ cons: may limit number of registers
- register-register
  - ✓ pros: simple, fixed length instruction encodings; instructions take similar number of cycles; relatively easy to pipeline and make superscalar
  - ✓ cons: higher instruction count; not all instructions need three operands; dependent on good compiler

Computer Performance: time!, time!, time!

- latency (response time)
  - ✓ how long does it take to execute a task?
  - ✓ how long must i wait for the database query?
  - ✓ high-percentile response time (SLO, SLA, etc. at datacenters)
- throughput
  - ✓ how many jobs can the machine complete in a minute?
  - ✓ what is the average execution rate?
  - ✓ how much work is getting done?

CPU Performance

- CPU execution time = seconds / program
  - ✓ time the CPU spends working on a task
  - ✓ does not include time waiting for I/O or running other programs

$$\frac{\text{Instructions}}{\text{program}} \times \frac{\text{cycles}}{\text{Instruction}} \times \frac{\text{seconds}}{\text{cycle}}$$

- programmer
- algorithms
- ISA
- compilers

- microarchitecture
- system architecture

- microarchitecture, pipeline depth
- circuit design
- technology

Performance Metric

- metric #1: time to complete a task (Texe)
  - ✓ execution time, response time, latency
    - X is N time faster than Y means Texe(Y)/Texe(X) = N
  - ✓ major metric used in this course
  - ✓ example:
    - machine A runs a program in 20 seconds
    - machine B runs the same program in 25 seconds
    - A is 25/20 times faster than B?
- metric #2: # of tasks per day, hour, seconds,
  - ✓ throughput or bandwidth
  - ✓ not the same as latency
    - CMPs IMPROVE THROUGHPUT BUT NOT LATENCY
- examples of unreliable metrics
  - ✓ millions instructions per second (MIPS), millions floating-point operations per second (MFLOPS)

How to Improve Performance

- to improve performance (everything else being equal) you can either

Reduce the # of required cycles for a program, or
Reduce the clock cycle time or, said another way,
Increase the clock rate.

$$\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}$$

Performance Related Metrics

- an execution of a given program will require
  - ✓ some number of instructions (machine instructions)
  - ✓ some number of cycles
  - ✓ some number of seconds
- we have a vocabulary of metrics that relate these quantities:
  - ✓ cycle time (seconds per cycle)
  - ✓ clock rate (cycles per second)
  - ✓ CPI (cycles per instruction)
    - ○ a floating point intensive application might have a higher CPI
- instruction set metrics
  - ✓ if two machines have the same ISA which of our quantities (e.g., clock rate, CPI, execution time, # of instructions, MIPS) will likely be identical during a comparison?

Performance Examples

1. supposing we have two implementations of the same ISA, what machine is faster for a given program, and by how much?
   - ✓ machine A w/ a clock cycle time of 10ns and a CPI of 2.5
   - ✓ machine B w/ a clock cycle time of 20ns and a CPI of 1
2. considering two code sequences for a given machine w/ 3 different classes of instructions: class A, B, and C requiring 1, 2, and 3 cycles per instruction, which sequence will be faster?
   - ✓ first code sequence w/ 5 instructions (2 of A, 1 of B, and 2 of C)
   - ✓ second code sequence w/ 6 instructions (4 of A, 1 of B, and 1 of C)

You should try when getting back home

3. considering two compilers and a machine operating at 100MHz w/ 3 classes of instructions: class A, B, and C requiring 1, 2, and 3 cycles per instruction, which compiler generates faster running code?
   - ✓ first compiler's code uses 5M Class A instructions, 1M Class B instructions, and 1M Class C instructions
   - ✓ second compiler's code uses 10M Class A instructions, 1M Class B instructions, and 1M Class C instructions

Effective CPI

- computing the overall effective CPI is done by looking at the different types of instructions and their individual cycle counts and averaging
  - ✓ overall effective CPI $= \sum_{i=1}^{n} (CPI_i \times IC_i)$
    - ✓ where IC_i is the count (percentage) of the number of instructions of class i executed
    - ✓ CPI_i is the (average) number of clock cycles per instruction for that instruction class
    - ✓ n is the number of instruction classes
- overall effective CPI varies by instruction mix – a measure of the dynamic frequency of instructions across one or many programs

Effective CPI: Excercise

op	freq	CPI_i	freq x CPI_i
ALU	50%	1	0.5x1
load	20%	5	0.2x2
store	10%	3	0.1x3
branch	20%	2	0.2x2
overall effective CPI			$\sum$ =

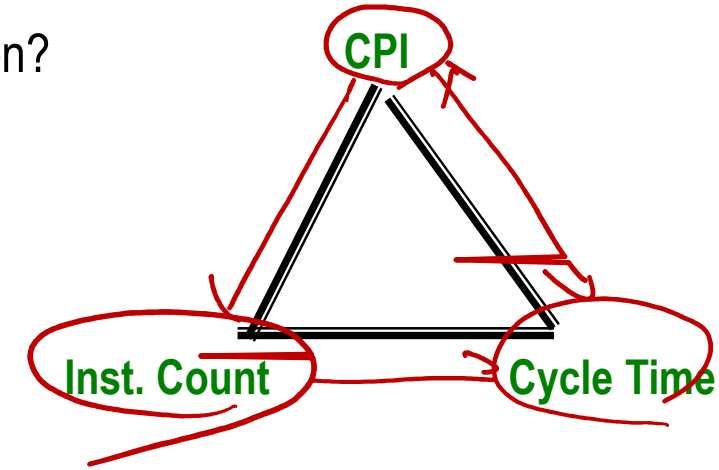
try this later

- how much faster would the machine be if a better data cache reduced the average load time to 2 cycles?
- how does this compare with using branch prediction to shave a cycle off the branch time?
- what if two ALU instructions could be executed at once?

Evaluating ISAs

- design-time metrics:
  - ✓ can it be implemented, in how long, at what cost?
  - ✓ can it be programmed? ease of compilation?
- static Metrics:
  - ✓ how many bytes does the program occupy in memory?
- dynamic metrics:
  - ✓ how many instructions are executed? how many bytes does the processor fetch to execute the program?
  - ✓ how many clocks are required per instruction?
  - ✓ how "lean" a clock is practical?
- best metric: time to execute the program!

depends on the instructions set, the processor organization, and compilation techniques.



Benchmarking

which program to choose?

- real programs
  - ✓ porting problem, complexity, not easy to understand the cause of results
- kernels
  - ✓ computationally intense piece of real programs
- toy benchmarks
  - ✓ QuickSort
- synthetic benchmarks
- benchmark suites
  - ✓ SPEC for scientific, engineering, and general purpose
  - ✓ TPC benchmarks for commercial systems
  - ✓ EEMBC for embedded systems

SPEC Benchmarks

Integer benchmarks		FP benchmarks	
gzip	compression	wupwise	Quantum chromodynamics
vpr	FPGA place & route	swim	shallow water model
gcc	GNU C compiler	mgrid	multigrid solver in 3D fields
mcf	combinatorial optimization	applu	parabolic/elliptic pde
crafty	chess program	mesa	3D graphics library
parser	word processing program	galgel	computational fluid dynamics
eon	computer visualization	art	image recognition (NN)
perlbmk	perl application	equake	seismic wave propagation simulation
gap	group theory interpreter	facerec	facial image recognition
vortex	object oriented database	ammp	computational chemistry
bzip2	compression	lucas	primality testing
twolf	circuit place & route	fma3d	crash simulation fem
		sixtrack	nuclear physics accel
		apsi	pollutant distribution

Reporting Performance

how do we summarize performance for benchmark set w/ a single number?

let Ti be the exe time of program i:

- (weighted) arithmetic mean of execution times:

$$\sum_i T_i / N \qquad \sum_i T_i \times W_i$$

  - ✓ issue is programs w/ the longest execution time will dominate the result

- guiding principle in reporting performance measurements is reproducibility – list everything another experimenter need to duplicate the experiment (version of OS, compiler settings, input set used, specific computer configuration (clock rate, cache sizes and speed, memory size and speed, etc.))

Reporting Performance

- dealing w/ speedup (normalized exe time)
  - ✓ speedup measures the advantage of a machine over a reference machine for a program i
  - ✓ arithmetic, harmonic, and geometric means are used
    - arithmetic mean impacted by choice of reference machine
    - geometric mean for comparison: $\prod(T_i)^{1/n}$ considering it is independent of chosen reference machine but not good metric for total execution time

Reporting Performance

	program a	program b	arithmetic mean	speedup (ref 1)	speedup (ref 2)
machine 1	10 sec	100 sec	55 sec	91.8	10
machine 2	1 sec	200 sec	100.5 sec	50.2	5.5
reference 1	100 sec	10000 sec	5050 sec		
reference 2	100 sec	1000 sec	550 sec		

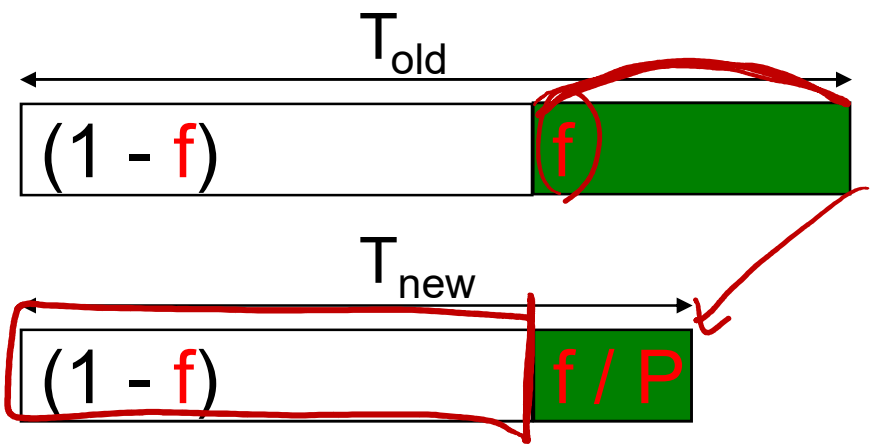
		a	b	am	hm	gm
wrt reference 1	machine 1	10	100	55	18.2	31.6
	machine 2	100	50	75	66.7	70.7
wrt reference 2	machine 1	10	10	10	10	10
	machine 2	100	5	52.5	9.5	22.4

Amdahl's Law

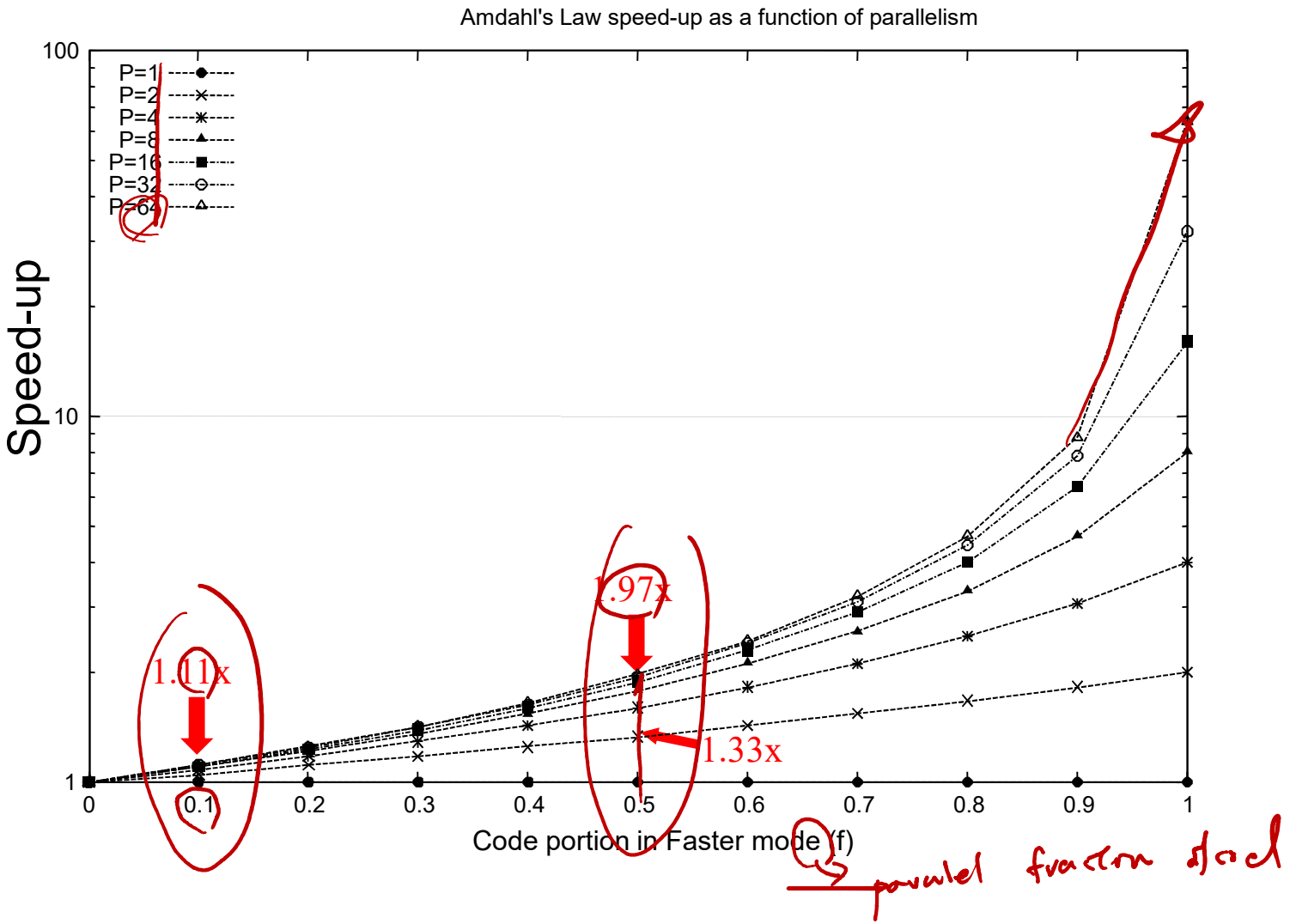
- law of diminishing returns
  - ✓ make the common case faster
  - ✓ speedup = $Perf_{new} / Perf_{old} = T_{old} / T_{new} =$

$$\frac{1}{(1-f)+\frac{f}{P}}$$

- example
  - ✓ supposing floating point instructions are improved to run 2X but comprise only 10% of actual instructions, what's the speedup?

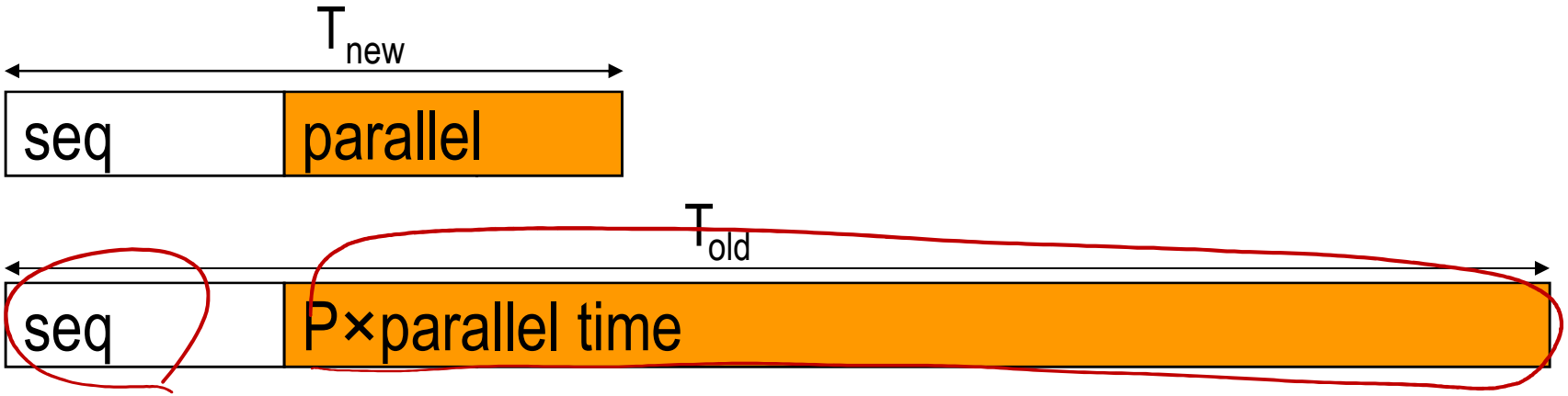


Parallelism vs. Speedup



Gustafson's Law

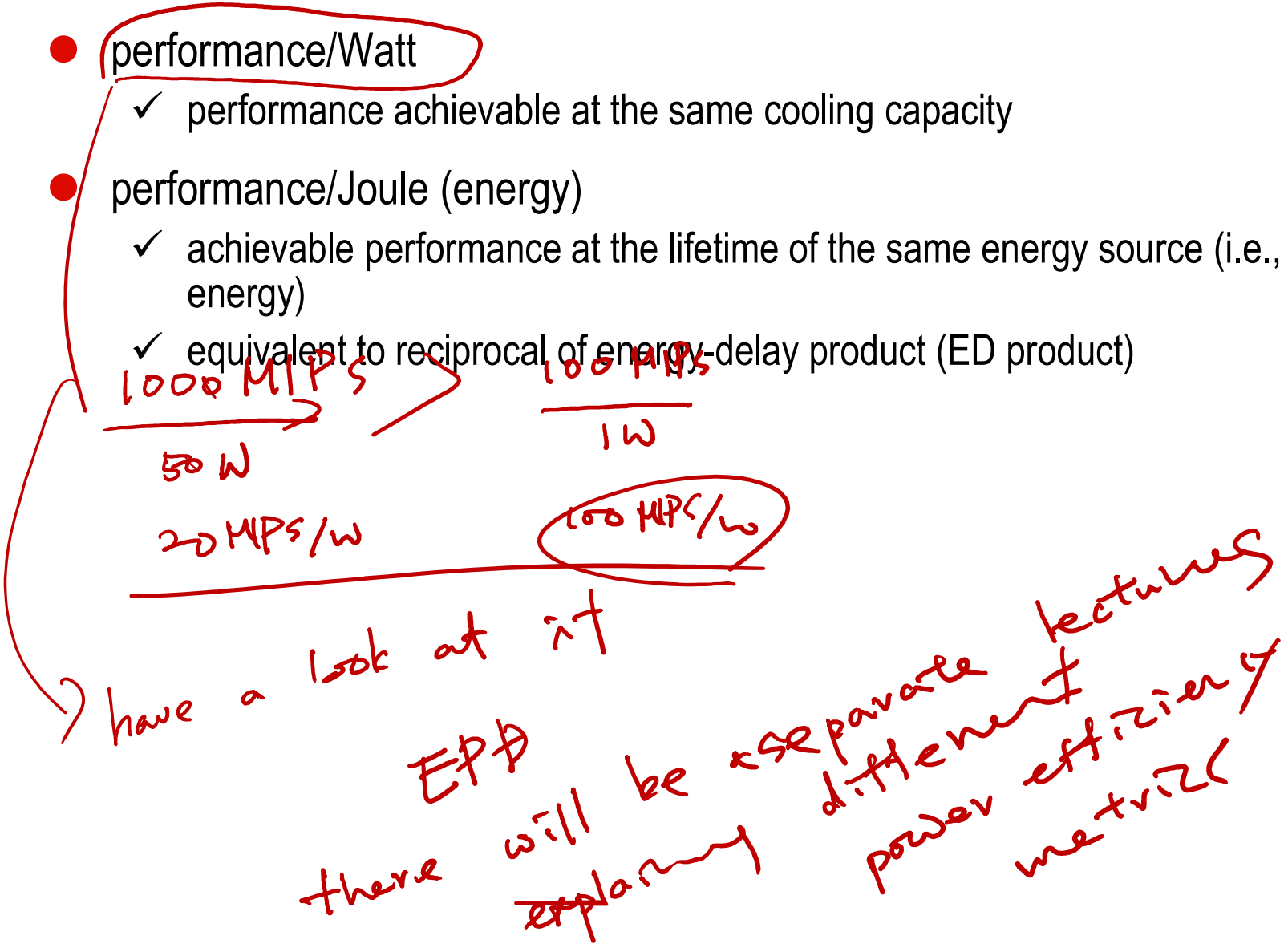
- Amdahl's Law killed massive parallel processing (MPP)
- Gustafson came to rescue
  - ✓ more workloads



  - ✓ assume: seq + parallel = 1 (Tnew)
  - ✓ speedup = seq + p × (1 – seq) where p = parallel factor
  - ✓ if seq diminishes w/ increased problem size, speedup → p

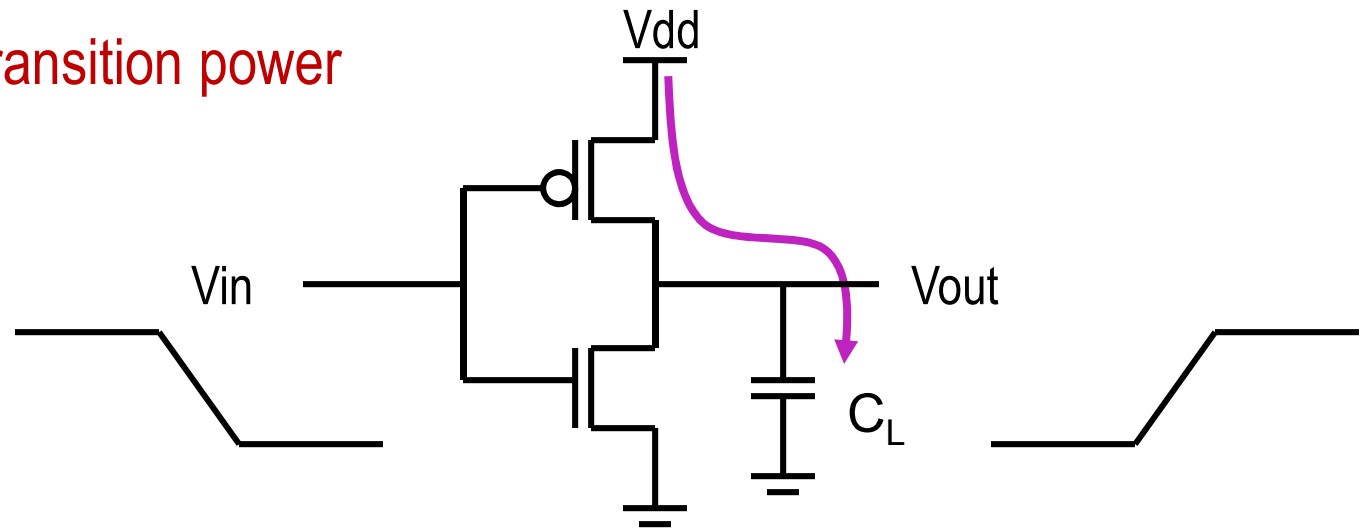
New Breed of Metrics

- performance/Watt
  - ✓ performance achievable at the same cooling capacity
- performance/Joule (energy)
  - ✓ achievable performance at the lifetime of the same energy source (i.e., battery = energy)
  - ✓ equivalent to reciprocal of energy-delay product (ED product)



(Dynamic) Power Dissipation

transition power

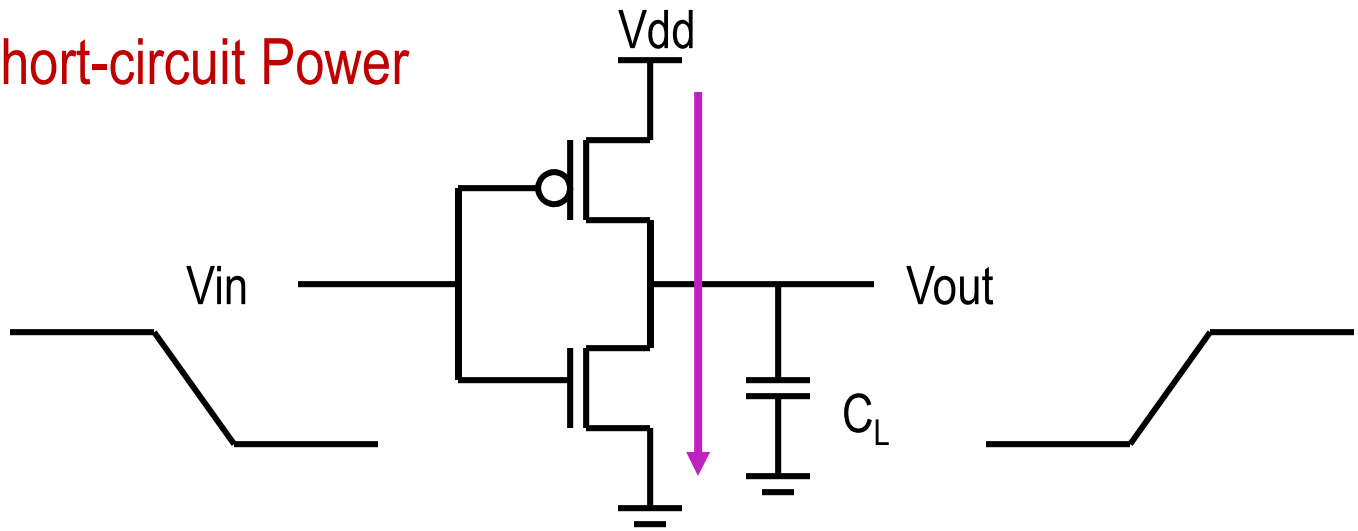


$$\text{energy/transition} = C_L \times V_{DD}^2 \times P_{0/1 \rightarrow 1/0}$$

$$\text{power} = C_L \times V_{DD}^2 \times f$$

(Short Circuit) Power Dissipation

Short-circuit Power



$$\text{Energy/transition} = t_{sc} \times V_{DD} \times I_{peak} \times P_{0/1 \rightarrow 1/0}$$

$$\text{Power} = t_{sc} \times V_{DD} \times I_{peak} \times f$$

CMOS Energy & Power Equations

$$E = C_L V_{DD}^2 P_{0\rightarrow1} + t_{sc} V_{DD} I_{peak} P_{0\rightarrow1} + V_{DD} I_{leakage}$$

$$f_{0\rightarrow1} = P_{0\rightarrow1} * f_{clock}$$

$$P = C_L V_{DD}^2 f_{0\rightarrow1} + t_{sc} V_{DD} I_{peak} f_{0\rightarrow1} + V_{DD} I_{leakage}$$

dynamic power (≈ 40 - 70% today and decreasing relatively)

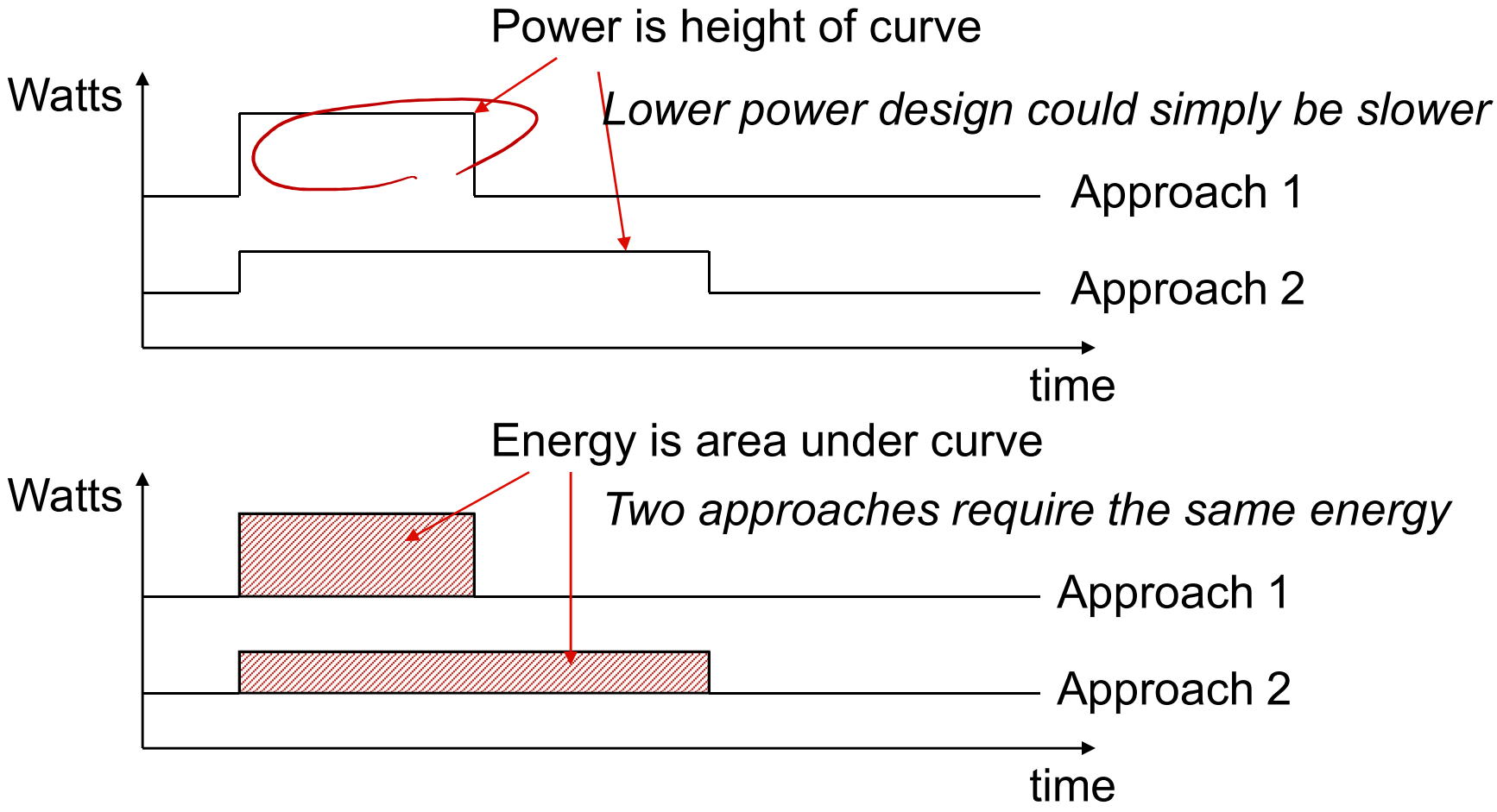
short-circuit power (≈ 10 % today and decreasing absolutely)

leakage power (≈ 20 – 50 % today and increasing)

Power and Energy Figures of Merit

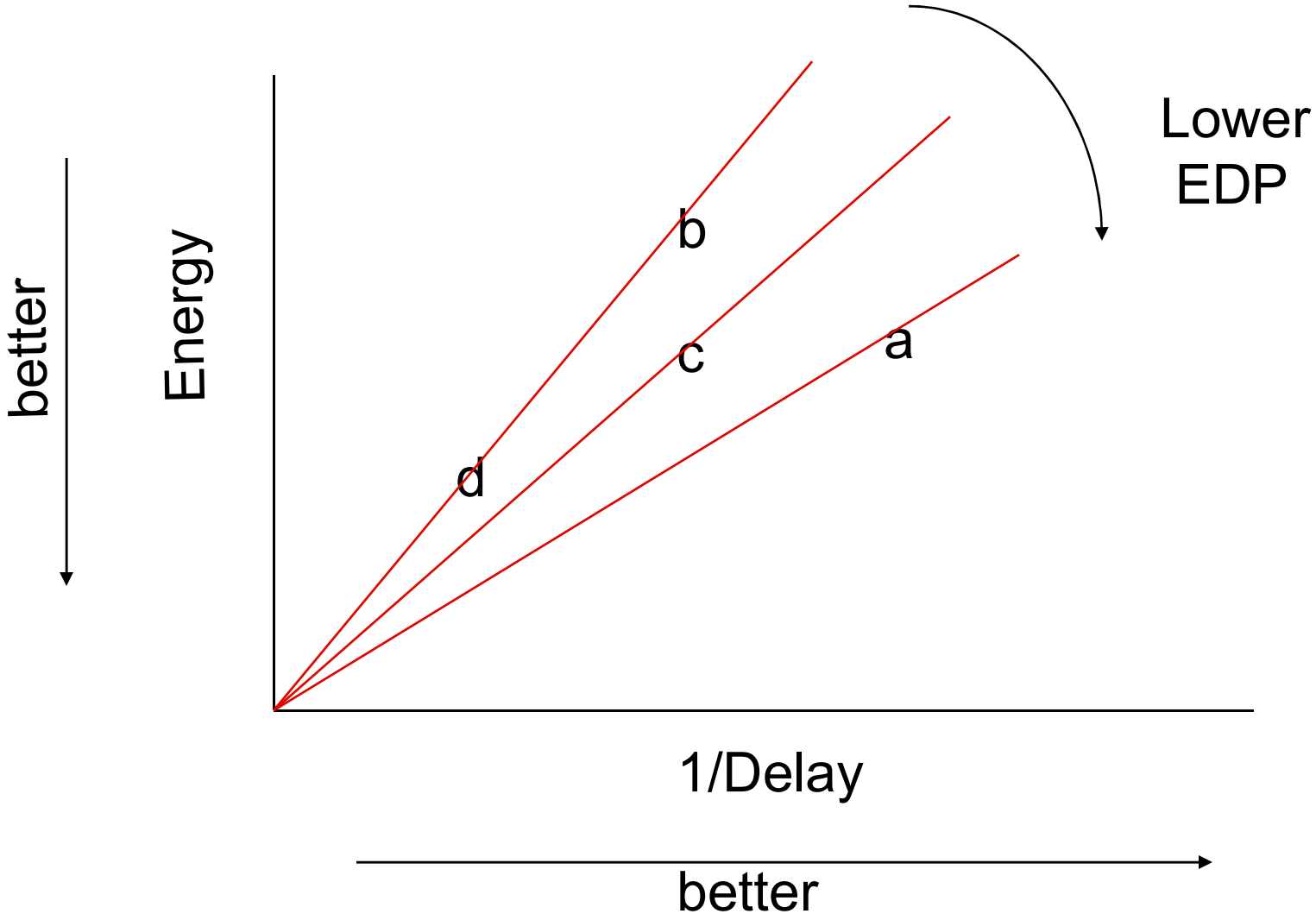
- power in Watts
  - ✓ (energy/time) poses constraints, i.e., processor can only work fast enough to max out the power delivery or cooling solution
  - ✓ peak power
    - determines power ground wiring designs
    - sets packaging/cooling limits
    - impacts signal noise margin and reliability analysis
- energy in Joules
  - ✓ ultimate metric, i.e., the true "cost" of performing a fixed task
  - ✓ energy = power × delay
    - Joules = Watts × seconds
    - lower energy number means less power to perform a computation at the same frequency
- example:
  - ✓ if processor A consumes 1.2× power of processor B, but finishes the task in 30% less time, which processor is more energy efficient and by how much?

Power versus Energy



Understanding Tradeoffs

- Which design is the “best” (fastest, coolest, both) ?



Announcement

- lecture 2 in text book
  - ✓ Appendix A (HP2), Ch. 2.16 – 2.18 (HP1)
- lecture 3 in text book
  - ✓ Ch. 1.6 – 1.7 (HP1) Ch. 1.8 (HP2)
- next lecture: single-cycle processor
  - ✓ Ch. 4.1 – 4.4 (HP1)
- MP assignment
  - ✓ MP0 due on 1/23 5pm