CS152
Computer Architecture and Engineering
Lecture 2: Cost and Performance

January 20, 1995

Dave Patterson (patterson@cs) & Shing Kong (kong@cs)

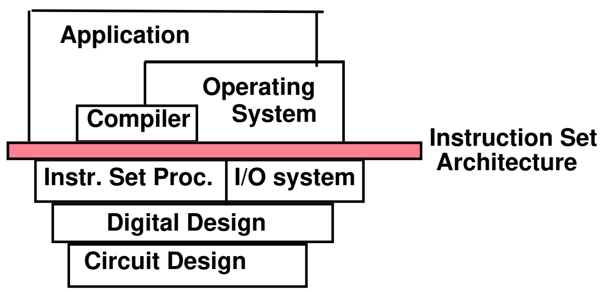
Slides available on http://http.cs.berkeley.edu/~patterson

Overview of Today's Lecture: Cost and Performance

- Review from Last Lecture (2 minutes)
- Cost of Integrated Circuits (20 minutes)
- Administrative Matters (3 minutes)
- Definition and Measures of Performance (25 minutes)
- Break (5 minutes)
- Summarizing Performance and Performance Pitfalls (25 minutes)

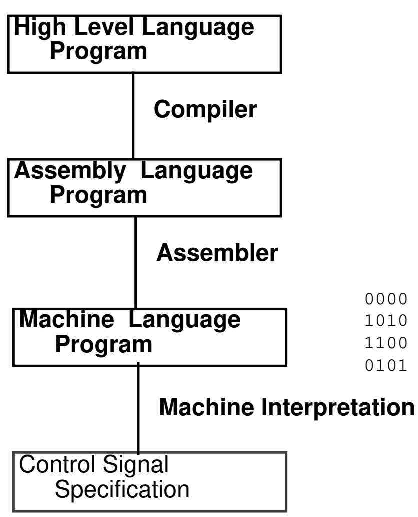
Review: What is "Computer Architecture"

° Co-ordination of ***levels of abstraction***



° Under a set of rapidly changing ***Forces***

Review: Levels of Representation

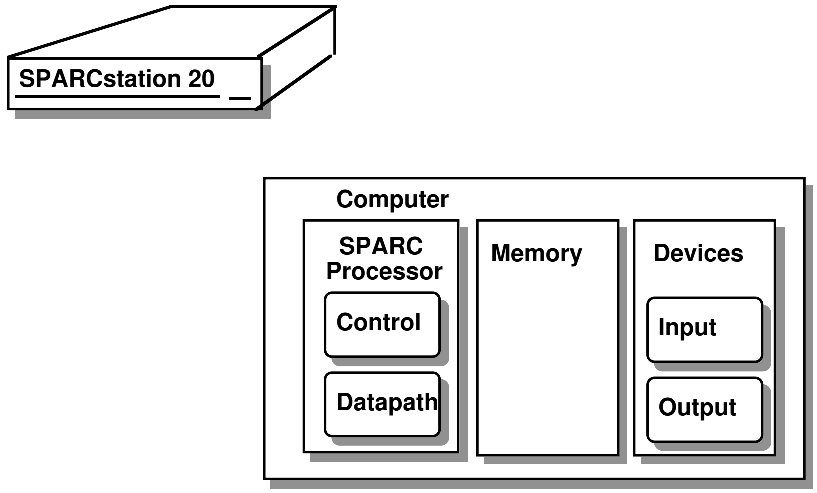



```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

```
lw  $15,   0($2)
lw  $16,   4($2)
sw  $16,   0($2)
sw  $15,   4($2)
```

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```

Review: Levels of Organization



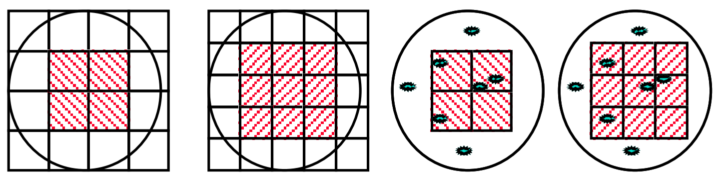
Review: Summary from Last Lecture

- All computers consist of five components
  - Processor: (1) datapath and (2) control
  - (3) Memory
  - (4) Input devices and (5) Output devices
- Not all "memory" are created equally
  - Cache: fast (expensive) memory are placed closer to the processor
  - Main memory: less expensive memory--we can have more
- Input and output (I/O) devices has the messiest organization
  - Wide range of speed: graphics vs. keyboard
  - Wide range of requirements: speed, standard, cost ... etc.
  - Least amount of research (so far)

Integrated Circuits Costs

$$\text{Die cost} = \frac{\text{Wafer cost}}{\text{Dies per Wafer} * \text{Die yield}}$$

$$\text{Dies per wafer} = \frac{\pi * (\text{Wafer\_diam} / 2)^2}{\text{Die Area}} - \frac{\pi * \text{Wafer\_diam}}{\sqrt{2 * \text{Die Area}}} - \text{Test dies} \approx \frac{\text{Wafer Area}}{\text{Die Area}}$$



$$\text{Die Yield} = \text{Wafer yield} * \left\{ 1 + \frac{\text{Defects\_per\_unit\_area} * \text{Die\_Area}}{\alpha} \right\}^{-\alpha}$$

Die Cost is goes roughly with the cube of the area.

Real World Examples

Chip	*Metal layers*	*Line width*	*Wafer cost*	*Defect /cm²*	*Area mm²*	*Dies/ wafer*	*Yield*	*Die Cost*
386DX	2	0.90	$900	1.0	43	360	71%	$4
486DX2	3	0.80	$1200	1.0	81	181	54%	$12
PowerPC 601	4	0.80	$1700	1.3	121	115	28%	$53
HP PA 7100	3	0.80	$1300	1.0	196	66	27%	$73
DEC Alpha	3	0.70	$1500	1.2	234	53	19%	$149
SuperSPARC	3	0.70	$1700	1.6	256	48	13%	$272
Pentium	3	0.80	$1500	1.5	296	40	9%	$417

From "Estimating IC Manufacturing Costs," by Linley Gwennap, *Microprocessor Report*, August 2, 1993, p. 15

Other Costs

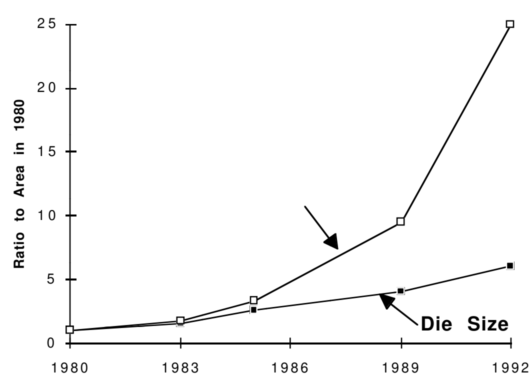
$$\text{IC cost} = \frac{\text{Die cost} + \text{Testing cost} + \text{Packaging cost}}{\text{Final test yield}}$$

Packaging Cost: depends on pins, heat dissipation , ...

Chip	*Die cost*	*Package pins*	*Package type*	*Package cost*	*Test & Assembly*	*Total*
386DX	$4	132	QFP	$1	$4	$9
486DX2	$12	168	PGA	$11	$12	$35
PowerPC 601	$53	304	QFP	$3	$21	$77
HP PA 7100	$73	504	PGA	$35	$16	$124
DEC Alpha	$149	431	PGA	$30	$23	$202
SuperSPARC	$272	293	PGA	$20	$34	$326
Pentium	$417	273	PGA	$19	$37	$473

CMOS improvements

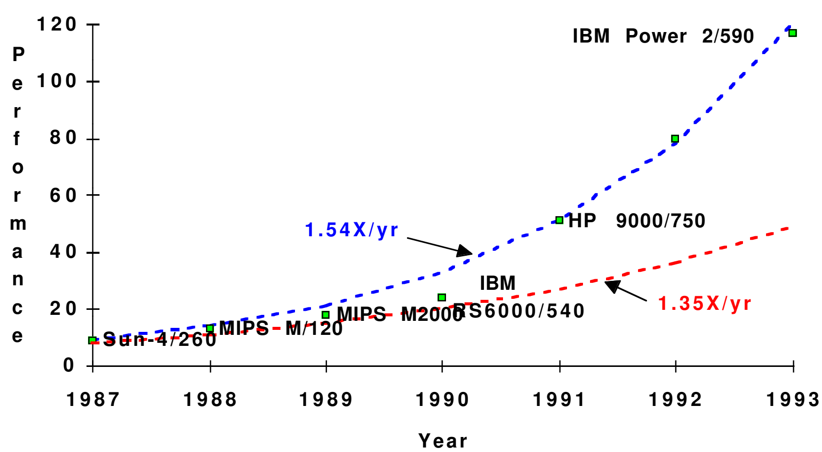
° **Die size 2X / 3 years; Line widths halve / 7 years**



Technology Trends

	Capacity	Speed
Logic	2x in 3 years	2x in 3 years
DRAM	4x in 3 years	1.4x in 10 years
disk	4x in 3 years	1.4x in 10 years

Processor Performance



The bottom line: Performance (and cost)

Plane	DC to Paris	Speed	Passengers	Throughput (pmph)
Boeing 747	6.5 hours	610 mph	470	286,700
BAD/Sud Concodre	3 hours	1350 mph	132	178,200

° **Time to do the task (Execution Time)**
  – execution time, response time, latency
° **Tasks per day, hour, week, sec, ns. .. (Performance)**
  – throughput, bandwidth

The bottom line: Performance (and cost)

" X is n times faster than Y" means

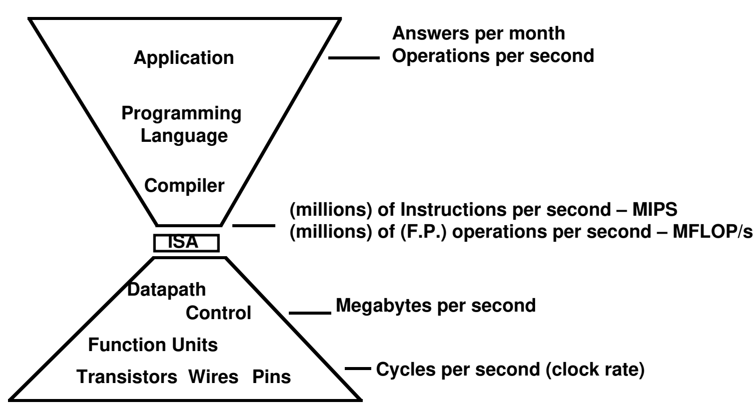
$$\frac{\text{ExTime}(Y)}{\text{ExTime}(X)} = \frac{\text{Performance}(X)}{\text{Performance}(Y)}$$

- Time of Concorde vs. Boeing 747?
- Throughput of Boeing 747 vs. Concodre?

Administrative Matters

- CS152 news group: ucb.class.cs152
- Slides available via Mosaic: http://http.cs.berkeley.edu/~patterson
- Initial Assignment: Read Chapters 1 and 2 of "Computer Organization and Design", Exercises 1.1 to 1.26, 1.50 to 1.52, 2.5 to 2.7, 2.14-2.17, 2.33 + run Linpack program on two machines and measure performance
- Class decided on penalties for cheating
  - Exercises (book): 0 for homework assignment
  - Labs leading to project (groups: penalize individuals in groups)
    - 0 for assignment
  - Exams: 0 for exam
- Book is hardcover version
- Other topics?

Metrics of performance



Relating Processor Metrics

- CPU execution time = CPU clock cycles/pgm X clock cycle time
- or CPU execution time = CPU clock cycles/pgm ÷ clock rate
- CPU clock cycles/pgm = Instructions/pgm X avg. clock cycles per instr.
- or CPI = CPU clock cycles/pgm ÷ Instructions/pgm
- CPI tells us something about the Instruction Set Architecture, the Implementation of that architecture, and the program measured

Aspects of CPU Performance

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

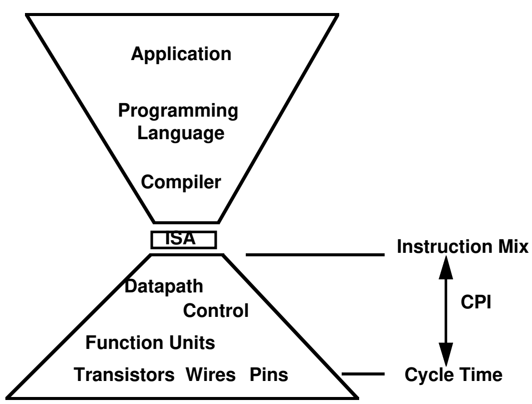
	instr. count	CPI	clock rate
Program			
Compiler			
Instr. Set Arch.			
Organization			
Technology			

Aspects of CPU Performance

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

	instr count	CPI	clock rate
Program	X		
Compiler	X	(x)	
Instr. Set.	X	X	
Organization		X	X
Technology			X

Organizational Trade-offs



CPI

“Average cycles per instruction”

CPI = Instruction Count / (CPU Time * Clock Rate)
= Instruction Count / Cycles

$$\text{CPU time} = \text{CycleTime} * \sum_{i=1}^{n} \text{CPI}_i * I_i$$

"instruction frequency"

$$\text{CPI} = \sum_{i=1}^{n} \text{CPI}_i * F_i \quad \text{where } F_i = \frac{I_i}{\text{Instruction Count}}$$

Invest Resources where time is Spent!

Example

Base Machine (Reg / Reg)

Op	Freq	Cycles	CPI(i)	% Time
ALU	50%	1	.5	33%
Load	20%	2	.4	27%
Store	10%	2	.2	13%
Branch	20%	2	.4	27%
			1.5	

Typical Mix

Marketing Metrics

MIPS = Instruction Count / Time * 10^6

= Clock Rate / CPI * 10^6

•machines with different instruction sets ?

•programs with different instruction mixes ?

- dynamic frequency of instructions

• uncorrelated with performance

MFLOP/S = FP Operations / Time * 10^6

•machine dependent

•often not where time is spent

Why Do Benchmarks?

- How we evaluate differences
  - Different systems
  - Changes to a single system
- Provide a target
  - Benchmarks should represent large class of important programs
  - Improving benchmark performance should help many programs
- For better or worse, benchmarks shape a field
- Good ones accelerate progress
  - good target for development
- Bad benchmarks hurt progress
  - help real programs v. sell machines/papers?
  - Inventions that help real programs don't help benchmark

Programs to Evaluate Processor Performance

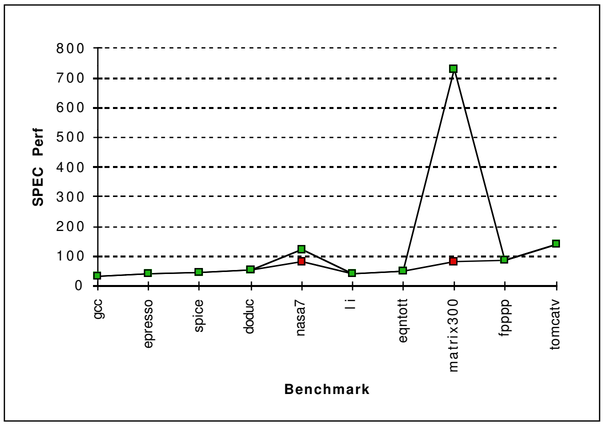
- (Toy) Benchmarks
  - 10-100 line
  - e.g.,: sieve, puzzle, quicksort
- Synthetic Benchmarks
  - attempt to match average frequencies of real workloads
  - e.g., Whetstone, dhrystone
- Kernels
  - Time critical excerpts of real programs
  - e.g., Livermore loops
- Real programs
  - e.g., gcc, spice

Successful Benchmark: SPEC

- 1987 RISC industry mired in “bench marketing”: (“That is 8 MIPS machine, but they claim 10 MIPS!”)
- EE Times + 5 companies band together to perform Systems Performance Evaluation Committee (SPEC) in 1988: Sun, MIPS, HP, Apollo, DEC
- Create standard list of programs, inputs, reporting: some real programs, includes OS calls, some I/O

SPEC first round

- First round 1989; 10 programs, single number to summarize performance
- One program: 99% of time in single line of code
- New front-end compiler could improve dramatically



SPEC Evolution

- Second round; SpecInt92 (6 integer programs) and SpecFP92 (14 floating point programs)
  - Compiler Flags unlimited. March 93 of DEC 4000 Model 610:
  - spice: `unix.c:/def=(sysv,has_bcopy,”bcopy(a,b,c)= memcpy(b,a,c)”`
  - wave5: `/ali=(all,dcom=nat)/ag=a/ur=4/ur=200`
  - nasa7: `/norecu/ag=a/ur=4/ur2=200/lc=blas`
- Add SPECbase: one flag setting for integer programs & 1 for FP
- Third round; 1995; new set of programs
  - “benchmarks useful for 3 years”

How to Summarize Performance

- Arithmetic mean (or weighted arithmetic mean) tracks execution time: SUM(Ti)/n or SUM(Wi*Ti)
- Harmonic mean (or weighted harmonic mean) of rates (e.g., MFLOPS) tracks execution time:
  n/SUM(1/Ri) or n/SUM(Wi/Ri)
- Normalized execution time is handy for scaling performance (e.g., time on reference machine ÷ time on measured machine)
- But do not take the arithmetic mean of normalized execution time, use the geometric mean (prod(Ri)^1/n)
- Alas, geometric mean rewards all improvements equally: program A going from 2 seconds to 1 second as important as program B going from 2000 seconds to 1000 seconds

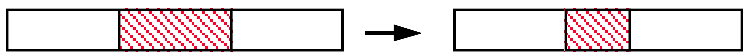
Impact of Means on SPECmark89 for IBM 550

	Ratio to VAX:		Time:		Weighted Time:	
Program	*Before*	*After*	*Before*	*After*	*Before*	*After*
gcc	30	29	49	51	8.91	9.22
espresso	35	34	65	67	7.64	7.86
spice	47	47	510	510	5.69	5.69
doduc	46	49	41	38	5.81	5.45
nasa7	78	144	258	140	3.43	1.86
li	34	34	183	183	7.86	7.86
eqntott	40	40	28	28	6.68	6.68
matrix300	78	730	58	6	3.43	0.37
fpppp	90	87	34	35	2.97	3.07
tomcatv	133	138	20	19	2.01	1.94
Mean	54	72	124	108	54.42	49.99
	Geometric		*Arithmetic*		*Weighted Arith.*	
	Ratio	1.33	Ratio	1.16	Ratio	1.09

Amdahl's Law

Speedup due to enhancement E:

$$\text{Speedup(E)} = \frac{\text{ExTime w/o E}}{\text{ExTime w/ E}} = \frac{\text{Performance w/ E}}{\text{Performance w/o E}}$$



Suppose that enhancement E accelerates a fraction F of the task by a factor S and the remainder of the task is unaffected then,

ExTime(with E) = ((1-F) + F/S) X ExTime(without E)

Speedup(with E) = ExTime(without E) ÷ ((1-F) + F/S) X ExTime(without E)

Cost Summary

- Integrated circuits driving computer industry
- Die costs goes up with the cube of die area

Performance Evaluation Summary

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

- Time is the measure of computer performance!
- Good products created when have:
  - Good benchmarks
  - Good ways to summarize performance
- If not good benchmarks and summary, then choice between improving product for real programs vs. improving product to get more sales=> sales almost always wins
- Remember Amdahl's Law: Speedup is limited by unimproved part of program