EE282
Computer Architecture

Lecture 1: What is Computer Architecture?

September 27, 2001

Marc Tremblay
Computer Systems Laboratory
Stanford University
marctrem@csl.stanford.edu

Goals

- Understand how computer systems are organized and why they are organized that way.
  - instruction-set architecture
  - system-level organization
  - microarchitecture
- Be conversant with measures of computer performance and methods for increasing performance
  - metrics
  - benchmarks
  - performance methods
    - pipelining, multiple issue, branch prediction

Logistics

Lectures	Tues & Thur 1:15-2:30		
Lecturers	Marc Tremblay, Andrew Wolfe, Partha Ranganathan		
TAs	Daniel Wang Other		
Grading	Final Exam	1	35%
	Midterm	1	25%
	Homework	4+2	40%
Text	Hennessy & Patterson *Computer Architecture: A Quantitative Approach (2nd Edition)*		

See Policy Sheet for details

EE282 Online

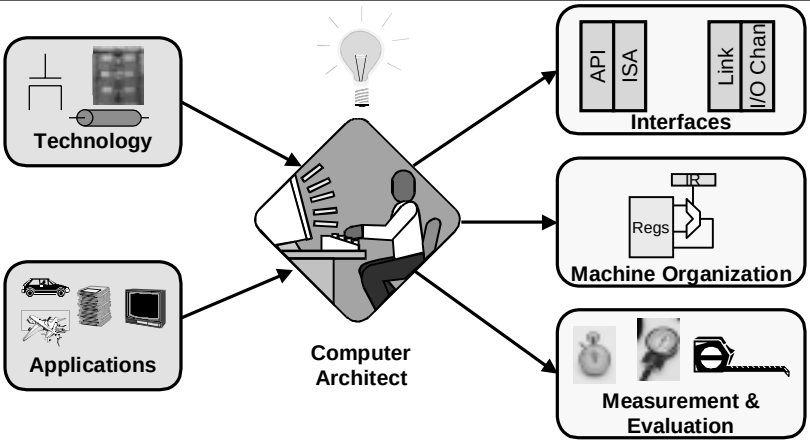
URL: **`http://www.stanford.edu/class/ee282`**

e-mail: **`ee282@lists.stanford.edu`**
`ee282tas@ogun.stanford.edu`

Wanted: A Few Good Graders

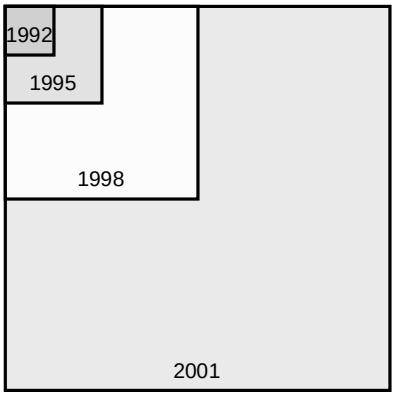
- Grade a portion of every problem set
- In exchange:
  - requirement to take problem sets is waived
  - full credit on all problem sets

What is Computer Architecture?



Technology Constraints

- Yearly improvement
  - Semiconductor technology
    - 60% more devices per chip (doubles every 18 months)
    - 15% faster devices (doubles every 5 years)
    - Slower wires
  - Magnetic Disks
    - 60% increase in density
  - Circuit boards
    - 5% increase in wire density
  - Cables
    - no change



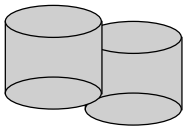
64x more devices since 1992
4x faster devices

Changing Technology leads to Changing Architecture

- 1970s
  - multi-chip CPUs
  - semiconductor memory very expensive
  - microcoded control
  - complex instruction sets (good code density)
- 1980s
  - single-chip CPUs, on-chip RAM feasible
  - simple, hard-wired control
  - simple instruction sets
  - small on-chip caches
- 1990s
  - lots of transistors
  - complex control to exploit instruction-level parallelism
- 2000s
  - even more transistors
  - slow wires
  - limited cooling capacity
  - several CPUs per chip
  - thread level parallelism
  - more to come…

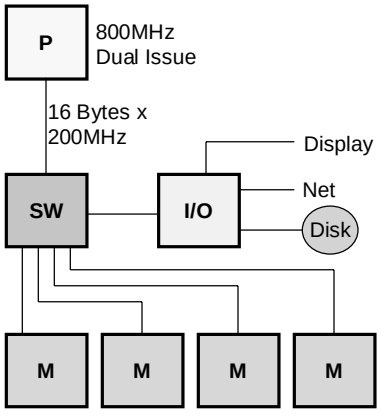
Application Constraints

- Applications drive machine 'balance'
  - Numerical simulations (technical computing)
    - floating-point performance
    - main memory bandwidth
  - Transaction processing
    - I/Os per second
      - Memory bandwidth/latency important
    - integer CPU performance
  - Media processing
    - low-precision 'pixel' arithmetic
    - multiply-accumulate rates
    - bit manipulation
  - Embedded control
    - I/O timing



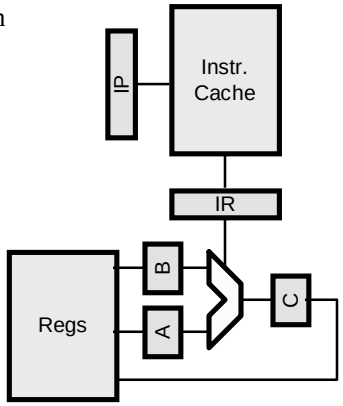
System-Level Organization

- Design at the level of processors, memories, and interconnect.
- More important to application performance than CPU design
- Feeds and speeds
  - constrained by IC pin count, module pin count, and signaling rates
- System balance
  - for a particular application
- Driven by
  - performance/cost goals
  - available components (cost/perf)
  - technology constraints

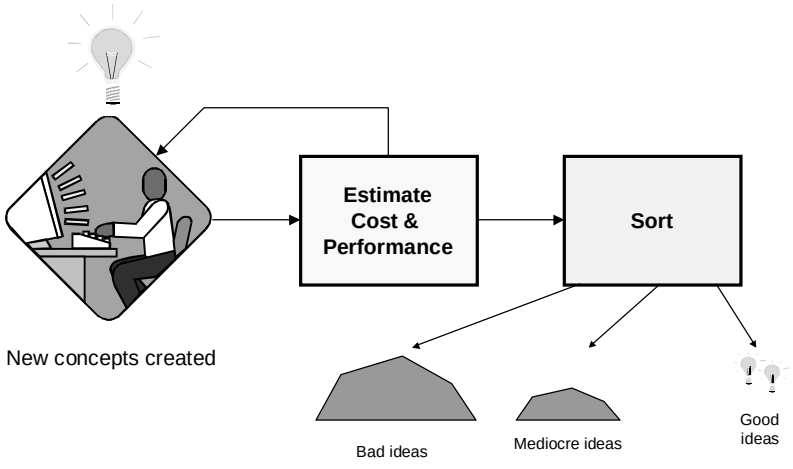


Microarchitecture

- Register-transfer-level (RTL) design
- Implement instruction set
- Exploit capabilities of technology
  - locality and concurrency
- Iterative process
  - generate proposed architecture
  - estimate cost
  - measure performance
- Current emphasis is on overcoming sequential nature of programs
  - deep pipelining
  - multiple issue
  - dynamic scheduling
  - branch prediction/speculation

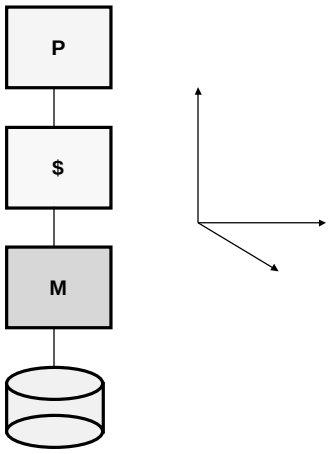


The Architecture Process



Performance Measurement and Evaluation

- Many dimensions to computer performance
  - CPU execution time
    - by instruction or sequence
      - floating point
      - integer
      - branch performance
  - Cache bandwidth
  - Main memory bandwidth
  - I/O performance
    - bandwidth
    - seeks
    - pixels or polygons per second
- Relative importance depends on applications

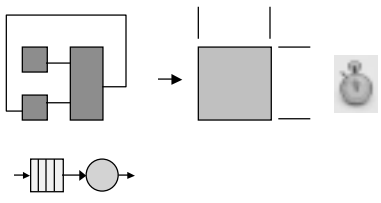


Evaluation Tools

- Benchmarks, traces, & mixes
  - macrobenchmarks & suites
    - application execution time
  - microbenchmarks
    - measure one aspect of performance
  - traces
    - replay recorded accesses
      - cache, branch, register
- Simulation at many levels
  - ISA, cycle accurate, RTL, gate, circuit
    - trade fidelity for simulation rate
- Area and delay estimation
- Analysis
  - e.g., queuing theory

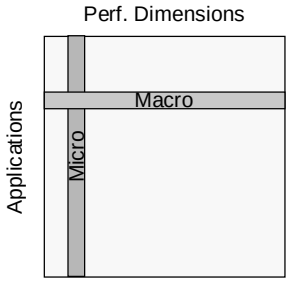
MOVE	39%
BR	20%
LOAD	20%
STORE	10%
ALU	11%

```
LD 5EA3
ST 31FF
....
LD 1EA2
....
```



Benchmarks

- Microbenchmarks
  - measure one performance dimension
    - cache bandwidth
    - main memory bandwidth
    - procedure call overhead
    - FP performance
  - weighted combination of microbenchmark performance is a good predictor of application performance
  - gives insight into the cause of performance bottlenecks
- Macrobenchmarks
  - application execution time
    - measures overall performance, but on just one application



Some Warnings about Benchmarks

- Benchmarks measure the whole *system*
  - application
  - compiler
  - operating system
  - architecture
  - implementation
- Popular benchmarks typically reflect yesterday's programs
  - computers need to be designed for tomorrow's programs
- Benchmark timings often very sensitive to
  - alignment in cache
  - location of data on disk
  - values of data
- Benchmarks can lead to *inbreeding* or positive feedback
  - if you make an operation fast (slow) it will be used more (less) often
    - so you make it faster (slower)
      - and it gets used even more (less)
        - and so on…

Means

Arithmetic mean $\frac{1}{n}\sum_{i=1}^{n} T_i$

Can be weighted. Represents total execution time

Harmonic mean $\frac{n}{\sum_{i=1}^{n} \frac{1}{R_i}}$

Geometric mean $\left(\prod_{i=1}^{n} \frac{T_i}{T_{ri}}\right)^{\frac{1}{n}} = \exp\left(\frac{1}{n}\sum_{i=1}^{n} \log\left(\frac{T_i}{T_{ri}}\right)\right)$

Consistent independent of reference

Performance Basics

- Amdahl's Law
  - speedup fraction p by S

$$T_1 = T_0\left[(1-p) + \frac{p}{S}\right]$$

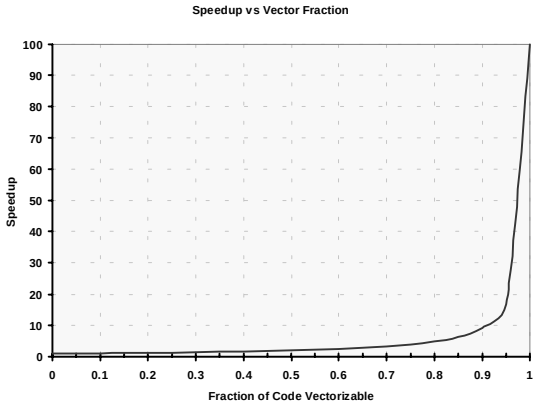
- Performance equation
  - I - instructions executed
  - CPI - clock cycles per instruction
  - t_{cy} - clock cycle
  - Need to consider all three elements when making changes

$T_{cpu} = I \times CPI \times t$

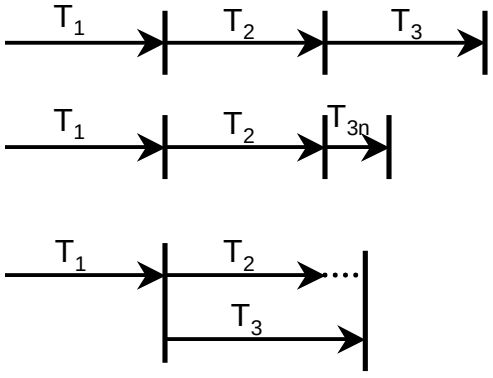
- But remember
  - workloads change depending on machine characteristics

Amdahl's Law

$$T_1 = T_0\left[(1-p) + \frac{p}{S}\right]$$



Amdahl's law assumes serial execution



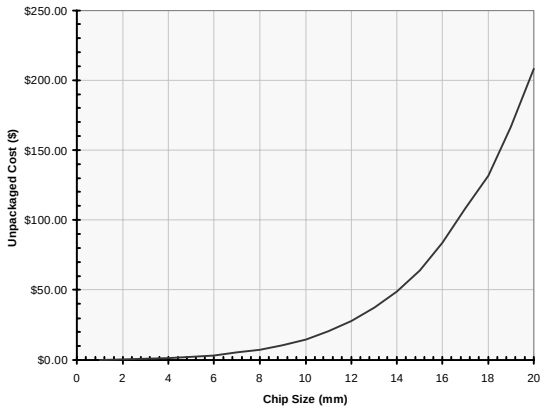
$$T = T_1 + T_2 + T_3$$

Cost

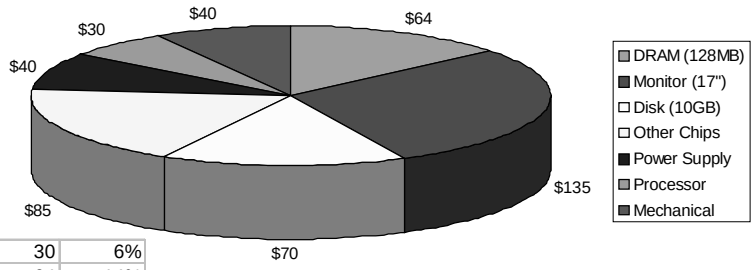
- Chip cost is primarily a function of die area
  - increases faster than linearly due to yield
  - a very powerful processor can be built on a small die (10mm)
  - going larger gives diminishing performance returns
  - but …

Wafer Cost		2,500.00			
Wafer Diameter		200	mm		
Wafer Area		31416	mm^2		
Chip Size	Die Area	Die/Wafer	Yield	Good Die	Cost/Die
1	1	30971	0.98	30351	$0.08
5	25	1167	0.96	1122	$2.23
7.5	56.25	499	0.81	406	$6.16
10	100	269	0.62	166	$15.06
15	225	110	0.35	38	$65.79
17.5	306.25	77	0.27	21	$119.05

Chip cost is a function of size



CPU cost is a small fraction of system cost (today)



Processor	30	6%
DRAM (128MB)	64	14%
Monitor (17")	135	29%
Disk (10GB)	70	15%
Other Chips	85	18%
Power Supply	40	9%
Mechanical	40	9%
Total	434	94%