**Advanced Computer Architecture I Lecture Notes**

*Mark D. Hill, David A. Wood, Gurindar S. Sohi*

Computer Sciences Dept.
1210 West Dayton St.
Univ. of Wisconsin
Madison, WI 53706

{markhill,david,sohi}@cs.wisc.edu
608-262-2196 (Hill)
608-263-7463 (Wood)
608-262-7985 (Sohi)
608-262-9777 (FAX)

These lecture notes were developed for teaching CS/ECE 752, *Advanced Computer Architecture I*, at the University of Wisconsin-Madison with text: Hennessy and Patterson, **Computer Architecture: A Quantitative Approach,** Morgan Kaufmann, Beta 2nd Edition, 1994.

Copyright (C) 1994 by Mark D. Hill, David A. Wood and Gurindar S. Sohi

All rights reserved. Except for use in CS/ECE 752, no part of these notes may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written permission from one of the authors.



## *What is CS/ECE 752?*

We wish to incorporate into the machine--in the form of circuits--only such logical concepts as are either necessary to have a complete system or highly convenient because of the frequency with which they occur and the influence they exert in the relevant mathematical situations.

--*Burks, Goldstine, von Neumann, 1946*

The emphasis on performance rather than aesthetics is deliberate. Without an interest in performance the study of architecture is a sterile exercise, since all computable problems can be solved using trivial architectures, given enough time.

--*Shustek, Ph.D. Thesis, 1978*



## *Chapter 1: Introduction*

Computer Architecture and This Course

What is Performance?

- Latency
- Throughput

Key Ideas

- Pipelining
- Memory Hierarchies

Amdahl's Law

Burks, Goldstine, and von Neumann, 1946



## *Architecture, Organization, Implementation*

Architecture: SW/HW interface

- instruction set
- memory management & protection
- interrupts & traps
- floating-point standard (IEEE)
- I/O buses

Organization: big picture of implementation

- number & location of functional units
- pipeline configuration
- location & configuration of caches

Implementation: Low-Level HW & Circuits

## *Architect's Job*

Evolve Architecture

- Compatibility
- Last through trends

Implement to Obtain

- High Performance
  - Supercomputers (Cray C-90)
  - Massively parallel processors (CM-5)
  - Mainframes (IBM 9000)
- Optimum Cost/Performance
  - Workstations (Sun SPARCstations)
  - Minisupercomputers (Convex, SGI)
  - High-end PCs (Pentium-based)
- Minimum Cost
  - Low-end PCs
  - Personal Digital Assistants (PDAs)



## *Why Study Computer Architecture?*

First electronic computer built 50 years ago.

**Answer #1:** Technology Changes Rapidly.

| Technology | Annual Improvement |
|---|---|
| Transistor Count | 25% |
| Transistor Speed | 20-25% |
| DRAM Density | 60% |
| DRAM Speed | 4% |
| Disk Density | 25% |
| Disk Speed | 4% |

**Goal:** Optimize Cost/Performance

**Answer #2:** User Requirements Change Rapidly

- New technologies inspire new applications



## *Prerequisites*

**CS/ECE 552 or equivalent:** Design correct computer

- Architecture: interface to software
  - (see earlier slide)
- Organization
  - Datapath design
  - Hardwired vs Microcode Control
  - Basic Pipelining
  - Basic Caches
- Implementation



## *Postrequisites*

**CS/ECE 757:** Adv. Computer Arch. II

- Shared-Memory vs. Message-Passing
- Advanced Cache Coherence
- SIMD vs. MIMD
- Dataflow
- Interconnection Networks
- Parallelizing Compilers

**CS/ECE 755:** VLSI System Design

- Introduction to VLSI Design
- VLSI Architectures
- Computer-aided Design

## *Quantitative Computer Architecture*

Design a Conventional Uniprocessor **WELL**

- Emphasis on Cost and Performance
- Quantitative not Qualitative Comparisons

The future for conventional uniprocessors:

- Instruction Sets
- Advanced Memory Systems
  - (caches, memory, virtual memory)
- Advanced Instruction Level Parallelism
  - (pipelining, superscalar, vectors, VLIW)
- Storage Systems (I/O)
- Interconnection Technology
- Basic Parallel Processing



## *What is Performance?*

**Key:** A measure of Speed (Rate)

- Car: Miles driven per hour
- Car Wash: Cars washed per day
- Auto Plant: Cars built per year

Two Metrics

- Latency (a.k.a. response or execution time)
  - time from start to finish of a task
- Throughput (a.k.a. bandwidth)
  - rate of task completion
  - = rate of task initiation
  - = 1 / (time between task completions)

Deterministic vs. average



## *Relative Performance*

Definition: X is n % faster than Y if

$$\frac{\textit{execution rate}_X}{\textit{execution rate}_Y} = \frac{\textit{execution time}_Y}{\textit{execution time}_X} = 1 + \frac{n}{100}$$

Example: X = 1 minute, Y = 2 minutes

$$\frac{2\ \textit{minute}}{1\ \textit{minute}} = 1 + \frac{100}{100}$$

Thus, X is 100 % faster than Y

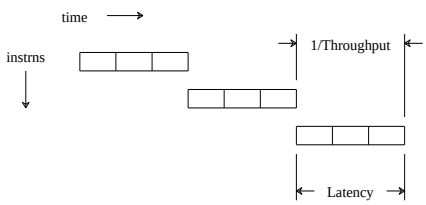
Example: Car Wash that starts one car per minute and holds four cars.

- Latency = four minutes per car
- Throughput = one car per minute
- Throughput > 1/Latency due to overlap
- Key Idea: Pipelining

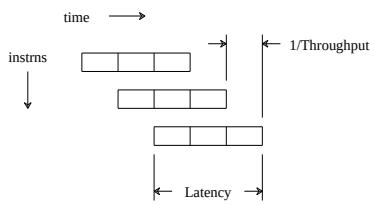


## *Key Idea: Pipelining*

Unpipelined:




Pipelined:




But

- Unequal pieces
- Latch overhead
- Hazards

## *Key Idea: Memory Hierarchy*

Principles of Performance

- Make the common case fast
- Locality for Data and Instrns
  - Temporal (time)—reference same
  - Spatial (space)—reference near

Implementation Facts

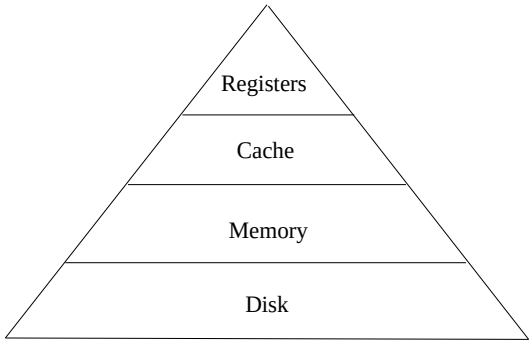
- SRAM is faster than DRAM
- DRAM is faster than Disk

Put it Together: Memory Hierarchy

- Keep recently referenced data as close to the processor as possible.
- Hierarchy of successively larger, slower, and cheaper memories.



## *Memory Hierarchy*




| Type | Size | Speed (ns) | Bandwidth (MB/s) |
|---|---|---|---|
| Registers | <1KB | 10 | 800 |
| Cache | <512KB | 20 | 200 |
| Memory | <512MB | 100 | 133 |
| Disk | >1GB | 20,000,000 | 4 |



## *Amdahl's Law*

(Or why the common case matters most)

Let $Speedup = \frac{new\ rate}{old\ rate} = \frac{old\ latency}{new\ latency}$

Consider an enhancement x that speedups fraction $f_x$ of a task by $S_x$.

$$Speedup_{overall} = \frac{old\ latency}{new\ latency}$$

$$= \frac{[(1-f_x) + f_x] \times old\ latency}{(1-f_x) \times old\ latency + (f_x/S_x) \times old\ latency}$$

Amdahl's Laws gives:

$$Speedup_{overall} = \frac{1}{(1-f_x) + (f_x/S_x)}$$



## *Amdahl's Law, cont.*

Example: $f_x$ = 95% and $S_x$ = 1.10

$$Speedup_{overall} = \frac{1}{(1-0.95) + (0.95/1.10)} = 1.094$$

Example: $f_x$ = 5% and $S_x$ = 10

$$Speedup_{overall} = \frac{1}{(1-0.05) + (0.05/10)} = 1.047$$

Example: $f_x$ = 5% and $S_x \rightarrow \infty$

$$Speedup_{overall} = \frac{1}{(1-0.05) + (0.05/\infty)} \rightarrow 1.052$$

## *Amdahl's Law Corollary*

Since $S_x \rightarrow \infty$ implies

$$Speedup_{overall} \rightarrow \frac{1}{(1-f_x) + (f_x/\infty)}$$

For all real speedups:

$$Speedup_{overall} < \frac{1}{1-f_x}$$

Example

| $f_x$ | $\frac{1}{1-f_x}$ |
|---|---|
| 1% | 1.01 |
| 2% | 1.02 |
| 5% | 1.05 |
| 10% | 1.11 |
| 20% | 1.25 |
| 50% | 2.00 |



## *Burks, Goldstine, and von Neumann, 1946*

''It is evident that the machine must be capable of storing in some manner not only the [data] ... but also the instructions which govern the actual machine.''

''Conceptually we have discussed above two different forms of memory: storage of numbers and storage of orders. ... the memory organ can be used to store both numbers and orders.''

Classical origin of the term ''von Neumann computer''

Many historians actually credit Eckert and Mauchly for this idea.



## *[BGvN46] cont.*

Hardware/Software Trade-offs

> ''We wish to incorporate into the machine--in the form of circuits--only such logical concepts as are either necessary to have a complete system or highly convenient because of the frequency with which they occur and the influence they exert in the relevant mathematical situations.''

Memory Hierarchies

> ''We are therefore forced to recognize the possibility of constructing a hierarchy of memories, each of which has greater capacity than the preceding but which is less quickly accessible.''

Accumulator based machine

> Why not a register machine?



## *[BGvN46]: Arithmetic*

Binary Arithmetic

> Earliest machines were decimal

Two's Complement Representation

Iterative Carry

Iterative Multiply (carry-save adders)

Rounding vs Jamming

Non-restoring division

No Floating-Point. Why?

## *[BGvN46]: Control*

40-bit data

20-bit instructions

- 8-bit instructions
- 12-bit addresses

Basic Instructions

- Conditional and Unconditional Branches
- Data Transfer
- ALU and Shift
- Store into Order (indirect addressing through program modification)

‘‘Hence an order is needed which will tell the computer to stop and to flash a light or ring a bell’’

(C) 1994 by Hill, Wood & Sohi