

# CS184b: Computer Architecture (Abstractions and Optimizations)

Day 1: March 28, 2005  
Architecture Intro



Caltech CS184 Spring2005 -- DeHon

## Today

- This Quarter
- What is Architecture?
  - Why?
- Project Overview

2

Caltech CS184 Spring2005 -- DeHon

## CS184 Sequence

- A - structure and organization
  - raw components, building blocks
  - design space
- B – architectural abstractions and optimization
  - emphasis on abstractions and optimizations including quantification
  - single and multiple threads

3

Caltech CS184 Spring2005 -- DeHon

## Topics this Quarter (1 of 2)

- “Architecture”
- Instruction-Set Architecture (ISA)
  - including pipeline parallelism
- Instruction-Level Parallelism (ILP)
- Memory Architecture and Optimization
  - Caching and Virtual Memory
- Binary Translation

4

Caltech CS184 Spring2005 -- DeHon

## Topics (2 of 2)

- Dataflow
- Multithreaded
- Message Passing
- Shared Memory
- Vector/SIMD
- Multiprocessor Interface/Interconnect
- Defect and Fault Tolerance

5

Caltech CS184 Spring2005 -- DeHon

## Material

- Lots of material – will go fast
- ...probably going to hit exposure over details

6

Caltech CS184 Spring2005 -- DeHon

## Lectures

- Same scheme as last term
  - Schedule MWF
  - Accommodate holes as necessary
  - Currently have 25 lectures on queue

Caltech CS184 Spring2005 -- DeHon

7

## Reading

- Will rely on much more than last term
- Will use textbook (Hennessy and Patterson)
  - chapters 1-6 this term
- Lectures more to complement text than completely overlap
  - going to cover some pretty rich topics
  - ...can't do it in 1.5--3 hours of lecture
- Classic papers

Caltech CS184 Spring2005 -- DeHon

8

## Assignments

- Pull from text
- Help drive working familiarity with conventional architecture techniques and optimizations

Caltech CS184 Spring2005 -- DeHon

9

## Logistics

- Four assignments on single threaded architectures
  - Due Monday 9am (out prev. Mon. class)
  - Still want electronic
    - no handwriting/hand drawing
- Project
  - Weekly milestones, starting next week
  - Sometimes will have both

Caltech CS184 Spring2005 -- DeHon

10

## Themes for Quarter

- Recurring
  - “cached” answers and change
  - merit analysis (cost/performance)
  - dominant/bottleneck resource requirements
  - structure/common case

Caltech CS184 Spring2005 -- DeHon

11

## Themes for Quarter

- New/new focus
  - measurement
  - abstractions/semantics
  - abstractions 0, 1, infinity
  - dynamic data/event handling (vs. static)
  - predictability (avg. vs. worst case)

Caltech CS184 Spring2005 -- DeHon

12

## “Architecture”

What? Why?

## “Architecture”

- “attributes of a system as seen by the programmer”
- “conceptual structure and functional behavior”
- Defines the **visible** interface between the hardware and software
- Defines the semantics of the program (machine code)

## Architecture distinguished from Implementation

- IA32 architecture vs.
  - 80486DX2, AMD K5, Pentium-II-700, P6
- VAX architectures vs.
  - 11/750, 11/780, uVax-II
- PowerPC vs.
  - PPC 601, 604, 630 ...
- Alpha vs.
  - EV4, 21164, 21264, ...
- Admits to many different implementations of single architecture

## Example Distinction: Memory Implementation

- **Abstraction:** large-flat memory
- **Implementation:**
  - multiple-levels of caches, varying sizes
  - virtual memory, with data residing on disk
  - relocation of physical memory placement
- One simple abstraction
  - hides details of implementation/timing
- Many implementations
  - varying costs, performance, technology

## Why ?

- What's the value of this distinction?
- Why do we have it?
  
- What does it cost?

## Value?

- Effort
- Economics
- Software Distribution

## Software Crisis

- Mid 1960's
  - Could build new machines at reasonable pace
  - Could not develop software for new machines fast enough

Caltech CS184 Spring2005 -- DeHon

19

## Historical Anecdotes

- Zuse from *The Computer, My Life*
- Brooks from *Software Pioneers*

Caltech CS184 Spring2005 -- DeHon

20

## Value: Effort

- Reduce/minimize effort necessary to exploit new/different technology
- Number of programmers is small
- Rate of new machine/technology advance is **large**
- Key enabler to riding the technology curve

Caltech CS184 Spring2005 -- DeHon

21

## Value: Economics

- Preserve software investment
  - both uniquely developed and commercial
- Lower barrier to acceptance of new machine
  - all your old code runs...just faster!
- Offer range of scaling:
  - need more power → buy different/better/newer machine
  - have less money → buy the cheaper machine
  - little/no software effort to support

Caltech CS184 Spring2005 -- DeHon

22

## Architecture Benefits

- ISA addressed the “software crisis”
  - Bottleneck to exploiting new machines was the need to write new software suites for them
- Preserve investment in software
  - Programmer education
- Permitted innovation in hardware
  - Use more/less hardware
  - Allow customers buy as much machine as they need
  - New substrates: TTL, ECL, NMOS, CMOS...

Caltech CS184 Spring2005 -- DeHon

23

## Value: Software Distribution

- Vendor not want to sell source
  - “give away” their techniques/technology/IP in a way which can be co-opted/reused

– [pragmatic argument, not fundamental]

Caltech CS184 Spring2005 -- DeHon

24

## Pragmatic: Binary vs. Source Compatibility

- For various software engineering reasons (failures?)
  - source notoriously bad/problematic to port to new machine
  - entire application not all packaged up in one place
    - must find compatible libraries, compiler, compiler options, header files...
    - different (newer) compilers give different results

Caltech CS184 Spring2005 -- DeHon

25

## Pragmatic: Binary vs. Source Compatibility

- For various software engineering reasons (failures?)
- People generally more comfortable with binary compatibility
- ABI/Binary architectural definition smaller/tighter and more well defined?
- **André:** Shouldn't have to be this way...but that's where we are today

Caltech CS184 Spring2005 -- DeHon

26

## Fixed Points

- Architecture requires we “fix” the interface
- Trick is picking what to expose in the interface and fix, and what to hide
- What are the “fixed points?”
  - how you describe the computation
  - primitive operations the machine understands
  - primitive data types
  - interface to memory, I/O
  - interface to system routines?

Caltech CS184 Spring2005 -- DeHon

27

## Abstract Away?

- Specific sizes
  - what fits in on-chip memory
  - available memory (to some extent)
  - number of peripherals
  - where 0, 1, infinity comes in
- Timing
  - individual operations
  - resources (e.g. memory)

Caltech CS184 Spring2005 -- DeHon

28

## Architectural Scalability

- Depends on robustness of fixed-points
  - address space
  - number of registers?
  - operations available
    - right level of abstraction?
  - Adequate primitives
    - e.g. atomic ops
  - sequential assumptions
  - single memory?
  - timing assumptions
    - e.g. branch delay, architectural cycles per op?

Caltech CS184 Spring2005 -- DeHon

29

## Change: Future like the past?

- VM/JIT compilation
- Binary Translation
- More advanced compiler technology and algorithms
- Architectural convergence?
  - Single Threaded ISA Maturity?

Caltech CS184 Spring2005 -- DeHon

30

## Conventional, Single-Threaded Abstraction

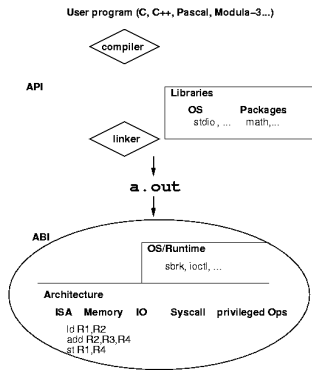
- Single, large, flat memory
- sequential, control-flow execution
- instruction-by-instruction sequential execution
- atomic instructions
- single-thread “owns” entire machine
  - isolation
- byte addressability
- unbounded memory, call depth

## Embodiment

- C+OS-API
  - C+unix-API, C+Windows-API
- Compile to:
  - ISA+OS-ABI
    - e.g. x86+linux-ABI
- Wrap up in standard, executable definition
  - e.g. a.out

## Abstractions

- Model for first half of course
- How support?
- How optimize?
- Remarkable
  - How far implementation can diverge



## Project

## Project: Graph Machine Network

- Look at Inter-PCE network
- For:
  - ConceptNet
  - SMVM
- Answer:
  - When should it be packet-switched vs. time-multiplexed?
  - Characterized cost/benefits of each

## Project Design

- Overlay network for FPGA
- Write VHDL
- Map to Xilinx Component (Virtex2)
- Get Area, Timing

## Project Steps

1. Get familiar with VHDL, build fast SRL queues
2. Build switching primitives
3. Assemble target switches
4. Assemble network, characterize size/density tradeoffs
5. Route/Simulate Traffic on designs and assess route-time (utilization)
6. Defect and Fault support
7. Custom implementation estimation

## Intuitive Tradeoff

- Benefit of Time-Multiplexing?
- Cost of Time-Multiplexing?
  
- Benefit of Packet Switching?
- Cost of Packet Switching?

## Intuitive Tradeoff

- Benefit of Time-Multiplexing?
  - Minimum end-to-end latency
  - No added decision latency at runtime
  - Offline route → high quality route
    - → use wires efficiently
- Cost of Time-Multiplexing?
  - Route task must be static
    - Cannot exploit low activity
  - Need memory bit per switch per time step
    - Lots of memory if need large number of time steps...

## Intuitive Tradeoff

- Benefit of Packet Switching?
  - No area proportional to time steps
  - Route only active connections
  - Avoids slow, off-line routing
- Cost of Packet Switching?
  - Online decision making
    - Maybe won't use wires as well
    - Potentially slower routing?
      - Slower clock, more clocks across net
  - Data will be blocked in network
    - Adds latency
    - Requires packet queues

## Packet Switch Motivations

- SMVM:
  - Long offline routing time limits applicability
  - Route memory exceeds compute memory for large matrices
- ConceptNet:
  - Evidence of low activity for keyword retrieval ... could be important to exploit

## Example

- ConceptNet retrieval
  - Visits 84K nodes across all time steps
  - 150K nodes
  - 8 steps → 1.2M node visits
  - Activity less than 7%

## Dishoom/ConceptNet Estimates

$N_z$	$N_{FPGA}$	$T_{comp}$	$T_{load}$	$T_{lat}$	$T_{step}$
1	64	500	1500	48	2048
2	128	250	750	52	1052
4	256	125	375	60	560
8	512	63	188	76	327

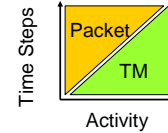
Pushing all nodes, all edges;  
Bandwidth ( $T_{load}$ ) dominates.

Caltech CS184 Spring2005 -- DeHon

43

## Question

- For what activity factor does Packet Switching beat Time Multiplexed Routing?
  - To what extent is this also a function of total time steps?



Caltech CS184 Spring2005 -- DeHon

44

## Big Ideas

- Architectural abstraction
  - define the fixed points
  - stable abstraction to programmer
  - admit to variety of implementation
  - ease adoption/exploitation of new hardware
  - reduce human effort

Caltech CS184 Spring2005 -- DeHon

45