CS 61C: Great Ideas in Computer Architecture (Machine Structures)

Course Summary

Instructor: Michael Greenbaum
Agenda

• Course Summary
• Administrivia/Project 2 Results
• What’s Next?
• Acknowledgements
Review: RISC vs. CISC

• RISC fundamentally changed processor design strategy.
• Lots of man-hours/development funds/transistors - CISC can work
  – Builds off of important RISC concepts (execution engine of CISC processor operates on RISC-like micro-ops)
• You need to understand RISC before you can build CISC!
• RISC still important for mobile/embedded devices.
Old School View of Software-Hardware Interface

- Application (ex: browser)
- Compiler
- Assembler
- Operating System (Mac OSX)
- Processor
- Memory
- I/O system
- Datapath & Control
- Digital Design
- Circuit Design
- transistors

Instruction Set Architecture

CS61C
New-School Machine Structures

- **Parallel Requests**
  - Assigned to computer
  - e.g., Search “Katz”

- **Parallel Threads**
  - Assigned to core
  - e.g., Lookup, Ads

- **Parallel Instructions**
  - >1 instruction @ one time
  - e.g., 5 pipelined instructions

- **Parallel Data**
  - >1 data item @ one time
  - e.g., Add of 4 pairs of words

- **Hardware descriptions**
  - All gates @ one time

---

**Software**

- **Hardware**

  - **Warehouse Scale Computer**
  - **Harness Parallelism & Achieve High Performance**

---

8/10/2011
Summer 2011 -- Lecture #29
6 Great Ideas in Computer Architecture

1. Layers of Representation/Interpretation
2. Moore’s Law
3. Principle of Locality/Memory Hierarchy
4. Parallelism
5. Performance Measurement & Improvement
6. Dependability via Redundancy
#1: Levels of Representation/Interpretation

**High Level Language Program (e.g., C)**

**Compiler**

**Assembler**

**Machine Language Program (MIPS)**

**Machine Interpretation**

**Hardware Architecture Description (e.g., block diagrams)**

**Architecture Implementation**

**Logic Circuit Description (Circuit Schematic Diagrams)**

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

- lw $t0, 0($2)
- lw $t1, 4($2)
- sw $t1, 0($2)
- sw $t0, 4($2)

Anything can be represented as a number, i.e., data or instructions

```
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
```

**Register File**

**ALU**

**Logic Circuit Description (Circuit Schematic Diagrams)**
Call home, we’ve made HW/SW contact!

High Level Language Program (e.g., C)

Assembly Language Program (e.g., MIPS)

Machine Language Program (MiPS)

Compiler

Assembler

Machine Interpretation

Hardware Architecture Description (e.g., block diagrams)

Architecture Implementation

Logic Circuit Description (Circuit Schematic Diagrams)

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

lw $t0, 0($2)
lw $t1, 4($2)
sw $t1, 0($2)
sw $t0, 4($2)

Anything can be represented as a number, i.e., data or instructions

1010 1111 0101 1000
0000 1001 1100 0110
0101 1000 0000 1001
1100 0110 1010 1111
#2: Moore’s Law

Predicts: 2X Transistors / chip every 2 years

Curve shows ‘Moore’s Law’: transistor count doubling every two years

Gordon Moore
Intel Cofounder
B.S. Cal 1950!
#3: Principle of Locality/ Memory Hierarchy

```
Processor

CPU

PROCESSOR REGISTER

CPU CACHE

LEVEL 1 (L1) CACHE

LEVEL 2 (L2) CACHE

LEVEL 3 (L3) CACHE

PHYSICAL MEMORY

RAMDOM ACCESS MEMORY (RAM)

SOLID STATE MEMORY

NON-VOLATILE FLASH-BASED MEMORY

VIRTUAL MEMORY

FILE-BASED MEMORY

SSD, Flash Drive

EDO, SD-RAM, DDR-SDRAM, RD-RAM and More...

SUPER FAST
SUPER EXPENSIVE
TINY CAPACITY

FASTER
EXPENSIVE
SMALL CAPACITY

FAST
PRICED REASONABLY
AVERAGE CAPACITY

AVERAGE SPEED
PRICED REASONABLY
AVERAGE CAPACITY

SLOW
CHEAP
LARGE CAPACITY
```
Typical Memory Hierarchy

- Take advantage of the principle of locality to present the user with as much memory as is available in the cheapest technology at the speed offered by the fastest technology.
Memory Hierarchy

• Caches
  – 3Cs: Compulsory/Capacity/Conflict misses
  – Direct-mapped vs. Set-associative
  – Multi-level caches for fast clock + low miss rate

• Virtual Memory
  – Originally small physical memory that appears to be very large
  – Modern: provide isolation through separated address spaces
#4: Parallelism

Jane
Research  Composing  Typing

Sue
Research  Composing  Typing

Tom
Research  Composing  Typing

URGENT!
Forms of Parallelism

• Instruction Parallelism
  – Processor pipeline: multiple instructions in execution at the same time

• Task Parallelism
  – Synchronization primitives, openMP
  – Modern web services

• Data Parallelism
  – Map-Reduce, SIMD instruction set
  – Data and numerically intensive processing
Transition to Multicore
Parallelism - The Challenge

- Only path to performance is parallelism
  - Clock rates flat or declining

- Key challenge is to craft parallel programs that have high performance on multiprocessors as the number of processors increase – i.e., that scale
  - Scheduling, load balancing, time for synchronization, overhead for communication

- **Project #2**: fastest matrix multiply code on 8 processor (8 cores) computers
  - 2 chips (or sockets)/computer, 4 cores/chip
Great Idea #5: Performance Measurement and Improvement

• Matching application to underlying hardware to exploit:
  – Memory system
  – SIMD, hardware features

• Parallel speedup
  – Amdahl’s law

• Measuring hardware performance
  – AMAT, CPI, performance and power benchmarks.
Great Idea #6: Dependability via Redundancy

- Applies to everything from datacenters to storage to memory
  - Redundant datacenters so that can lose 1 datacenter but Internet service stays online
  - Redundant disks so that can lose 1 disk but not lose data (Redundant Arrays of Independent Disks/RAID)
  - Redundant memory bits so that can lose 1 bit but no data (Error Correcting Code/ECC Memory)
Course Summary

• As the field changes, cs61c has to change too!
• It is still about the software-hardware interface
  – Programming for performance!
  – Understanding the memory hierarchy and its impact on application performance
  – Unlocking the capabilities of the architecture for performance
    • Multicore programming and task parallelism
    • Special instructions
    • Special instruction sequences
Agenda

• Course Summary
• Administrivia/Project 2 Results
• What’s Next?
• Acknowledgements
Administrivia

• Extra OH
  – Come into the SD lab today wherever there’s a blank sign-up time.
  – Sean: After 8pm in the SD lab...

• Project 3 Face-to-Face grading today 200 SD lab. Don’t forget to show up!

• Final Exam - Tomorrow! 9am - 12pm 2050 VLSB
    • Use the back side of your midterm cheat sheet!
Administtrivia

• CS61c Final Feedback Survey
  – Chance to give feedback on this “new” 61c
    • First time it’s been offered during the summer
  – What worked and what didn’t?
  – I’ll send out the link after the final...
Project 2 Results: sgemm-small

![Graph showing the results of Project 2 for sgemm-small]
Project 2 Results: sgemm-all

![Histogram of sgemm-all with x-axis representing GFlops and y-axis representing Groups.]
Project 2 Results: sgemm-openmp

![Bar chart showing performance of sgemm-openmp in Gflops for different groups.](chart.png)
Project 2 EC Winners

• Only 3 extra credit submissions...
  – (EC due at same time as part 2, sorry about that...)
• Serial Winners:
  1. 11.4 Gflop/s Jonathan Dinu, Elan Frenkel
  1. 11.4 Gflop/s Julien Freche, Simon Relet
  3. 11.2 Gflop/s Nils Adiceam, Thibault Hartmann, Victor Degliame
• Didn’t submit, but still fast:
  11.9 Gflop/s Jiajun Wu, Terence Yuen
  11.3 Gflop/s Albert Magyar
  11.2 Gflop/s Edmond Lo, Jiasi Shen
Project 2 EC Winners

• Parallel Winners:
  1. 79.8Gflop/s Jonathan Dinu, Elan Frenkel
  2. 74.9Gflop/s Julien Freche, Simon Relet
  3. 69.2Gflop/s Nils Adiceam, Thibault Hartmann, Victor Degliame

• Didn’t submit, but still fast:
  78.9 Gflop/s John Song, Trung Nguyen
Agenda

- Course Summary
- Administrivia/Project 2 Results
- What’s Next?
- Acknowledgements
What’s Next?

• What classes should I take (now)?
• Take classes from great teachers! (teacher > class)
  – Distinguished Teaching Award (very hard to get)
  – HKN Course evaluations (≥6 is very good)
  – EECS web site has plan for year (up in late spring)
• If have choice of multiple great teachers
  – CS152 Computer Architecture & Engin. (Sp12)
  – CS162 Operating Systems and Systems Programming
  – CS164 Programming Languages and Compilers
  – CS267 Applications of Parallel Computers (Sp12)
What’s Next?

• If you did well in CS3 or 61[ABC] (A- or above) and want to be on staff?
  – Usual path: Lab Assistant => Reader => TA
  – Also: Self-Paced Center Tutor
  – Apply as Lab Assistant by emailing instructor before course
  – Reader/TA Applications available on EECS website
Taking advantage of Cal Opportunities

• Why we are a top university in the WORLD?
  – Research, research, research!
  – Whether you want to go to grad school or industry, you need someone to vouch for you!

• Techniques
  – Find out what you like, do lots of web research (read published papers), hit OH of Prof, show enthusiasm & initiative
Dan’s Opportunities

• **GamesCrafters** (Game Theory R & D)
  – Develop SW, analysis on 2-person games of no chance. (e.g., go, chess, connect-4, nim, etc.)
  – Req: ≥ A- in CS61C, Game Theory / SW Interest

• **UCBUGG** (Recreational Graphics)
  – Develop computer-generated images, animations.
  – Req: 3D interest
  – Taught as a DeCal by UCBUGG veterans
Agenda

• Course Summary
• Administrivia/Project 2 Results
• What’s Next?
• Acknowledgements
“I stand on the shoulders of giants…”

Lecturer SOE
Mike Clancy

Prof
David Culler

TA
Andy Carle
TA
Kurt Meinz
TA
David Jacobs
TA
Jeremy Huddleston

Thanks to these talented folks (& many others) whose contributions have helped make CS61C a really tremendous course!
Extra Special Thanks

Prof Randy Katz

Prof David Patterson

Lecturer SOE Dan Garcia

2010 Summer Instructor Paul Pearce
Special Thanks to the Staff:

• **TAs:**
  Justin Hsia
  Sean Soleyman
  Alvin Yuan

• **Readers:**
  Ken Cheng, Leah Dorner, Amanda Ren

• **Lab Assistants:**
  Kevin Shih, Edwin Liao
The Future for Future Cal Alumni

• What’s The Future?

• New Century, Many New Opportunities: Parallelism, Cloud, Statistics + CS, Bio + CS, Society (Health Care, 3rd world) + CS

• Cal heritage as future alumni
  – Hard Working / Can do attitude
  – Never Give Up (“Don’t fall with the ball!” - cf. The Play)

“The best way to predict the future is to invent it”
  – Alan Kay (inventor of personal computing vision)

The Future is up to you!