

# Advanced Computer Systems Architecture

## Chip-Multiprocessors: Applications and Architectures

CSE 526M

Prof. Patrick Crowley

## Plan for Today

- Comments on next week
  - Next week: Here's what we did on the IXP
  - Subsequent weeks: Here's what we could have done if ... . This would have been harder, this would have been easier, etc.
- Questions
- Today's discussion

## Project Logistics

- Dates
  - Today's date: Apr 8
  - End date: (**1 week later**) Thursday, April 15
- Weekly Milestones

<i>M1</i>	<i>Mar 4</i>	<i>Implementation 1</i>
<i>M2</i>	<i>Mar 18</i>	<i>Implementation 2</i>
<i>M3</i>	<i>Mar 25</i>	<i>Implementation 3</i>
<i>M4</i>	<i>Apr 1</i>	<i>Implementation Wrap-up</i>
<i>M5</i>	<i>Apr 8</i>	<i>Plan future work, Reports</i>
<i>M6</i>	<i>Apr 15</i>	<i>Presentations</i>

## Instruction Sets and Beyond

- Observation: "... RISC has swept away objectivity ... and obscured many important issues"
- Structure
  - Towards defining a RISC
  - Misunderstandings and misleading claims
  - Seeing through the clutter: Register windows
  - Another example: The Intel 432
- Thoughts on architectures and applications

## Observation

- RISC research has
  - Focused attention on important areas
    - ISA design
    - Implementation complexity
  - Ignored other areas
    - Where RISC principles fail
    - Broader system implications
    - Careful RISC/CISC comparisons
- CISC research has
  - Not participated in the debate
- Result: an unbalanced, short-sighted debate

## Defining a RISC

- Single cycle operation
- Load/store design
- Hardwired control
- Relatively few instructions and addressing modes
- Fixed instruction format
- Heavy compiler dependence

## Digression on the 801

- 3 ISA design principles of the IBM 801
- The ISA should consist of only those run-time operations that
  - Could not be moved to compile time
  - Could not be more efficiently executed by object code produced by a compiler that understood the high-level intent of the program
  - Could be implemented in random logic more effectively than the equivalent sequence of software instructions

## Misunderstandings

- The RISC philosophy goes beyond ISAs
  - More generally: *“a willingness to make design tradeoffs freely and consciously across architecture/implementation, hardware/software, and compile-time/run-time boundaries in order to maximize performance as measured in some specific context.”*
- Having two acronyms implies a dichotomy
  - More correctly: RISC and CISC represent points in a continuum

## Fallacies

- F: System = hardware, software and application code
  - Must consider the operating system, its constraints and overheads
- F: Simpler designs, shorter time-to-market
  - DEC's MicroVAX-32 only took several months
  - Sounds right, but had not been validated (academic projects  $\neq$  commercial projects)
  - What about system support? Compilers, libraries, OSes?

## Fallacies (2)

- F: Performance claims
  - Not all benchmarks are representative (e.g., Fibonacci series)
  - One important metric: application-level transactions per second (i.e., throughput)
  - MIPS has some relevance, but so do
    - Reliability
    - Availability
    - Response time

# Register Windows

- Idea: use multiple register sets to reduce/eliminate the need to push registers to memory on a function call
- Point: this is orthogonal to the RISC/CISC debate
- The RISC I and II papers did not factor this out when claiming RISC had an intrinsic performance advantage over the VAX and the 68000

# Effect of Register Windows

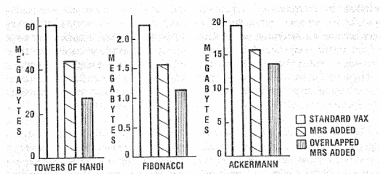


Figure 1. Total processor-memory traffic for benchmarks on the standard VAX and two modified VAX computers, one with multiple register sets and one with overlapped multiple register sets.

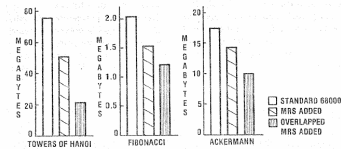


Figure 2. Total processor-memory traffic for benchmarks on the standard 68000 and two modified 68000s, one with multiple register sets and one with overlapped multiple register sets.

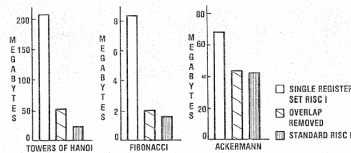


Figure 3. Total processor-memory traffic for benchmarks on the standard RISC I and two modified RISC I's, one with no overlap between register sets and one with only one register set.

## The Point

- *“Thus, any performance claims for RISCs that do not remove the effects due to multiple register sets are inconclusive, at best.”*

## The Intel 432

- A CISC that
  - Was designed to safely and efficiently execute large-scale system code, uses a microcoded, object-oriented run-time
    - E.g., performs checks on all memory references, for boundary conditions and protocol adherence
  - Had serious compiler and implementation flaws, when executing low-level compute-bound benchmarks
    - Non-optimized compiler
    - No caches, addresses and data share 16 pins, slow bus
    - Slow procedure calls (each call got a protected execution context)
    - Instruction decoding was bit-serial

## The Intel 432 (2)

- Analysis results
  - The RISC emphasis on fast decoding and local storage are valid *for low-level compute bound benchmarks*
  - However
    - Minor changes to compiler and implementation would erase these deficiencies
    - Large-scale system code actually benefits from some complex instructions
      - E.g., the SEND instruction for interprocess communication
  - *“Thus, it is wrong to conclude that the 432 supports the general RISC point of view.”*

## Concluding Quote

- *“The focus of the discussion should be on the more general question of the assignment of system functionality to implementation levels within an architecture.”*

## What has changed since these papers were written?

- Consider one metric: worldwide unit sales
- 1980: 724K PCs
- 1986: 9M PCs
- 2002: 130M PCs
- 2002: 3500 Itanium processors
- 2002: 36M game consoles
- 2002: 480M cellular phones

## Intellectual Digestion

- Does the RISC/CISC discussion have relevance today? Explain.
- Are new design principles needed to guide the architectures and implementations of modern systems? Any ideas?
- Will we see a new, dominant ISA in our lifetimes? Why, why not?

## RISC/CISC Relevance

- Yes!
- Given our SOC/CMP future: *“a willingness to make design tradeoffs freely and consciously across architecture/implementation, hardware/software, and compile-time/run-time boundaries in order to maximize performance as measured in some specific context.”*

## New Design Principles

- SOCs and CMPs represent enormous challenges
  - HW/SW interface: where should the function be implemented? What are the metrics?
  - Architecture: will all SOCs be application-specific, or will a *universal SOC* emerge?
  - Software: can our programming paradigms make use of abundant coarse-grain parallelisms? Is there an opportunity for a new way to write software?
  - VLSI: how will power, reliability, and pin counts be managed?

## New, Dominant ISA?

- If so, no one will know except the systems programmers and architects
- APIs and VMs will rule user's attention. See:
  - Microsoft's MRE initiative
  - Java

## Assignment

- Tuesday (4/13):
  - First 3 presentations
    - DDOS Shield
    - ACL-based Firewall
    - Bloom Filter
- Thursday (4/15)
  - Final 3 presentations
    - FAM/JackHMMer
    - FPLib
    - TCP Processor