CS 61C: Great Ideas in Computer Architecture (Machine Structures)
Modern Microprocessor Anatomy + RISC in Retrospect

Instructors:
Randy H. Katz
David A. Patterson
http://inst.eecs.Berkeley.edu/~cs61c/fa10

Review

• Rise of the Warehouse-Scale Computer
  – Mobile end device + backend computing infrastructure
• Cloud Computing
  – “Elastic” pay as you go computing
  – Made possible by VM technology + computing economies of scale
  – VMs enable fine control, allocation, pricing of Cloud Computing

Agenda

• Review
• Intel Nehalem
• Administrivia
• RISC CISC Debate 30 years later
• Summary

Intel Nehalem

• Review entire semester by looking at most recent microprocessor from Intel
• Nehalem is code name for microarchitecture at heart of Core i7 and Xeon 5500 series server chips
• First released at end of 2008
• Die size 263mm² at 45 nm
• 731M transistors

• Figures/Info from Intel, David Kanter at Real World Technologies.
Nehalem System Example:
Apple Mac Pro Desktop 2010

Each chip has three
DRAM channels attached,
each 8 bytes wide at
1.066Gb/s (7.85GB/s).
Can have up to two
DIMMs on each channel
(up to 4GB/DIMM).

Two Nehalem Chips ("Sockets"), each containing
four processors ("cores") running at up to 2.93GHz

QuickPath point-to-point system
Interconnect between CPUs and I/
Up to 25.6 GB/s per
link.

PCI Express connections for
Graphics cards and other extension
boards. Up to 8 GB/s per slot.

Disk drives attached with 3Gb/
s serial ATA link

Slower peripherals (Ethernet, USB, Firewire, WiFi, Bluetooth, Audio)

Core Area
Breakdown

Out of Order Execution Units
Out of Order Scheduling & Instruction Commit
Instruction Decode & Register Renaming
Memory Ordering & Execution
Load Store Queue
L3 cache

Shared L3 Cache

Nehalem Die Photo

Memory Controller
Shared L3 Cache

18.9 mm (0.75 inch)

Core
Core
Core
Core

In-Order Fetch
In-Order Decode and Register Renaming
In-Order Commit
Out-of-Order Execution

2 Threads per Core

Out-of-Order Completion

In-Order Commit

Out of Order
Execution

In-Order Fetch

11/23/10
### Front-End Instruction Fetch & Decode

**128 Entry ITLB (4 way)**
- 128 bits

**Instruction Fetch Unit**
- 16B Pre-Decode, Fetch Buffer
  - 6 instructions
  - 128 bits

**18 Entry Instruction Queue**
- x86 instruction bits

**µOP**
- is Intel name for internal RISC-like instruction, into which x86 instructions are translated

**Complex Decoder**
- 4 µops
- Internal µOP bits

**Simple Decoder**
- 1 µop

**26 Entry µop LSC Buffer**
- Loop Stream Detector (can run short loops out of the buffer)

### x86 Decoding
- Translate up to 4 x86 instructions into µOPS (RISC instructions) each cycle
- Only first x86 instruction in group can be complex (maps to 1-4 µOPS), rest must be simple (map to one µOP)
- Even more complex instructions, jump into microcode engine which spits out stream of µOPS

### Branch Prediction
- Part of instruction fetch unit
- Several different types of branch predictor
  - Details not public
- Two-level Branch Table Buffer
- Loop count predictor
  - How many backwards taken branches before loop exit
- Return Stack Buffer
  - Holds subroutine targets
  - Separate return stack buffer for each SMT thread

### Split x86 in small µOPs, then fuse back into bigger units

**Macro Fusion Example**
- CMP+JMP in 1 clock
- WITH macro fusion
  - INSTRUCTION 3
  - INSTRUCTION 2
  - INSTRUCTION 1
- DECODE
  - COMPLETED INST 3
  - INTERNAL INST 1
  - EXECUTE
  - COMPLETED INST 3
  - COMPLETED INST 2
  - COMPLETED INST 1
- WITHOUT macro fusion
  - INSTRUCTION 3
  - INSTRUCTION 2
  - INSTRUCTION 1
  - DECODE
  - INTERNAL INST 3
  - INTERNAL INST 2
  - EXECUTE
  - COMPLETED INST 3
  - COMPLETED INST 2
  - COMPLETED INST 1

**Advantage**
- Instruction Load Reduced ~ 15%**
- Micro-Op Reduced ~ 10%**
Loop Stream Detectors save Power

Multithreading effects in Out-of-Order Execution Core
- Reorder buffer (remembers program order and exception status for in-order commit) has 128 entries divided statically and equally between both threads
- Reservation stations (instructions waiting for operands for execution) have 36 entries competitively shared by threads

Out-of-Order Execution Engine
- Renaming happens at uOP level (not original macro-x86 instructions)
- 128 Entry Reorder Buffer (ROB)
- Retiree Register File (Program Visible State)
- 36 Entry Reservation Station

Nehalem Memory Hierarchy Overview
- Private L1/L2 per core
- Local memory access latency ~60ns
- 32KB L1 I/C
- 32KB L1 D/C
- 256KB L2
- 1MB Shared L3
- DDR3 DRAM Memory Controllers
- QuickPath System Interconnect
- Each DRAM Channel is 64/72b wide at up to 1.33Gb/s
- Each direction is 20b@6.4Gb/s
- L3 fully inclusive of higher levels (but L2 not inclusive of L1)
- Other sockets' caches kept coherent using QuickPath messages
All Sockets can Access all Data

Lunix doesn’t allocate pages to physical memory after malloc until first access to page. Be sure to touch what you want.

Remote Memory Access

Core’s Private Memory System

Load queue 48 entries
Store queue 32 entries
Divided statically between SMT threads
Up to 16 outstanding misses in flight per core

Cache Hierarchy Latencies

• L1 I & D 32KB 8-way, latency 4 cycles, 64B blocks
  – Note: 4KB Page (12 bits) + 8-way associativity
  (3 bits) means cache index doesn’t need to use virtual part of address, so L1 cache access and TLB lookup can occur in parallel, L1 cache uses physical address tags

• L2 256 KB 8-way, latency <12 cycles
• L3 8 MB, 16-way, latency 30-40 cycles
• DRAM, latency ~180-200 cycles
Nehalem Virtual Memory Details (Review)

- Implements 48-bit virtual address space, 40-bit physical address space
- Two-level TLB
  - I-TLB (L1) has shared 128 entries 4-way associative for 4KB pages, plus 7 dedicated fully-associative entries per SMT thread for large page (2/4MB) entries
  - D-TLB (L1) has 64 entries for 4KB pages and 32 entries for 2/4MB pages, both 4-way associative, dynamically shared between SMT threads
  - Unified L2 TLB has 512 entries for 4KB pages only, also 4-way associative
  - Additional support for system-level virtual machines

What to do with So Many Choices?

- “Introduction to Performance Analysis on Nehalem Based Processors”, 72 pages
  - “Software optimization based on performance analysis of large existing applications, in most cases, reduces to optimizing the code generation by the compiler and optimizing the memory access. Optimizing the code generation by the compiler requires inspection of the assembler of the time consuming parts of the application and verifying that the compiler generated a reasonable code stream. Optimizing the memory access is a complex issue involving the bandwidth and latency capabilities of the platform, hardware and software prefetching efficiencies and the virtual address layout of the heavily accessed variables.”

Administrivia: What’s Left

- Extra Credit Midnight tonight
- Wed 12/1: ½ on Top 3 Extra Credit for SGEMM
  - ½ on Cal Computing History in 1989: Redundant Array of Inexpensive Disks (RAID)
- Project 4 Face-to-Face Grading: Thu 12/2 in Lab
- Please fill out online survey in a lab to help next generation of 61C students!
- Friday 12/3: Course summary, Cal Heritage, HKN course evaluation
- Final Review: Mon 12/6, 2-5PM (10 Evans)
- Final: Mon 12/13 8AM-11AM (220 Hearst Gym)
RISC vs. CISC

- Set up: From 1965 to 1980, virtually all computers implemented instruction sets using microcode (edited Wikipedia entry):
  "Microcode is a layer of hardware-level instructions involved in the implementation of higher-level machine code instructions; it resides in a special high-speed memory and translates machine instructions into sequences of detailed circuit-level operations. It helps separate the machine instructions from the underlying electronics so that instructions can be designed and altered more freely. It also makes it feasible to build complex multi-step instructions while still reducing the complexity of the electronic circuitry compared to other methods. Writing microcode is often called microprogramming and the microcode in a particular processor implementation is sometimes called a microprogram."

- See slides 1 to 16 from RISCtalk1981v6.pdf
  – Unedited transparencies from 1981 RISC talk + RISC I, RISC II die photos

RESOLVING RISC-CISC DEBATE

Products shipped?
2010: 4.0B ARM, 0.3B x86

How USA resolves debates?
We ask celebrities!

Who is the biggest celebrity in the world?

RESOLVING RISC-CISC Debate

- X86 = 1.04 – 1.08x better performance/MHz vs. MIPS, ARM
- MIPS, ARM = 1.4 – 1.5x better energy/MHz vs. x86
- MIPS, ARM = ⅓ to ¼ die area vs. x86

Angelina Jolie as Kate Libby (aka as hacker Acid Burn) in movie “Hackers” (1995)
“And In Conclusion”

- Performance, Intel chip manufacturing
  => x86 ISA dominates Desktops/Servers
  – Speculative execution: branch prediction, out of
    order execution, data prefetching
  – Hardware translation and optimization of
    instruction sequences
  – Opportunistic acceleration (Turbo Mode)
- Cost, energy => RISC ISA dominates mobile
  personal devices, embedded computing, games
- What will the future hold?

Acknowledgements

- These slides contain material developed and
  copyright by:
  – Krste Asanovic (UCB)
- UCB material derived from course CS152