CSE 490/590 Computer Architecture

Putting it all together: Intel Nehalem

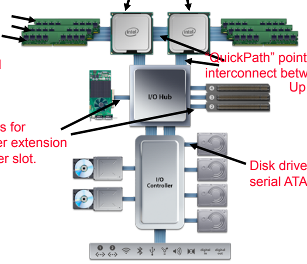
Steve Ko
Computer Sciences and Engineering
University at Buffalo

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

- Figures/Info from Intel, David Kanter at Real World Technologies.

Nehalem System Example: Apple Mac Pro Desktop 2009



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

Each chip has three DRAM channels attached, each 8 bytes wide at 1.066Gb/s (3*8.5GB/s).

Can have up to two DIMMs on each channel (up to 4GB/DIMM)

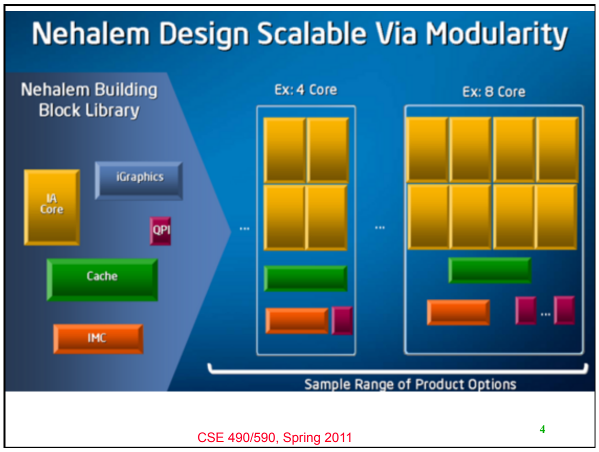
QuickPath" point-point system interconnect between CPUs and I/O. Up to 25.6 GB/s per link.

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

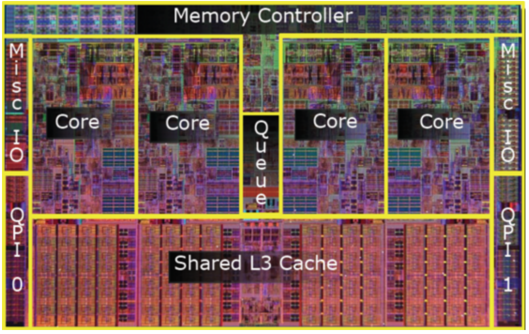
Disk drives attached with 3Gb/s serial ATA link

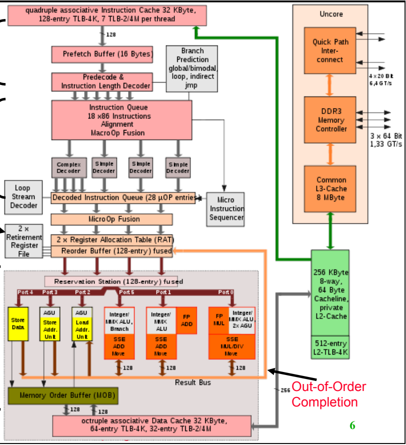
Slower peripherals (Ethernet, USB, Firewire, WiFi, Bluetooth, Audio)

Nehalem Design Scalable Via Modularity


Nehalem Building Block Library
IA Core
iGraphics
QPI
Cache
IMC
Ex: 4 Core
Ex: 8 Core
...
Sample Range of Product Options

Nehalem Die Photo




quadruple associative Instruction Cache 32 KByte, 128-entry TLB-4K, 7 TLB-2/4M per thread
128
Prefetch Buffer (16 Bytes)
Branch Prediction global/bimodal, loop, indirect jmp
Predecode & Instruction Length Decoder
Instruction Queue 18 x86 Instructions Alignment MacroOp Fusion
Complex Decoder
Simple Decoder
Simple Decoder
Simple Decoder
Loop Stream Decoder
Decoded Instruction Queue (28 µOP entries)
Micro Instruction Sequencer
MicroOp Fusion
2 x Retirement Register File
2 x Register Allocation Table (RAT)
Reorder Buffer (128-entry) fused
Reservation Station (128-entry) fused
Port 4
Port 3
Port 2
Port 5
Port 1
Port 0
Store Data
AGU Store Addr. Unit
AGU Load Addr. Unit
Integer/ MMX ALU, Branch
SSE ADD Move
Integer/ MMX ALU
FP ADD
SSE ADD Move
FP MUL
Integer/ MMX ALU, 2x AGU
SSE MUL/DIV Move
128
128
128
Result Bus
Memory Order Buffer (MOB)
128
128
octruple associative Data Cache 32 KByte, 64-entry TLB-4K, 32-entry TLB-2/4M
256
Uncore
Quick Path Inter-connect
4 x 20 Bit 6,4 GT/s
DDR3 Memory Controller
3 x 64 Bit 1,33 GT/s
Common L3-Cache 8 MByte
256 KByte 8-way, 64 Byte Cacheline, private L2-Cache
512-entry L2-TLB-4K

In-Order Fetch

In-Order Decode and Register Renaming

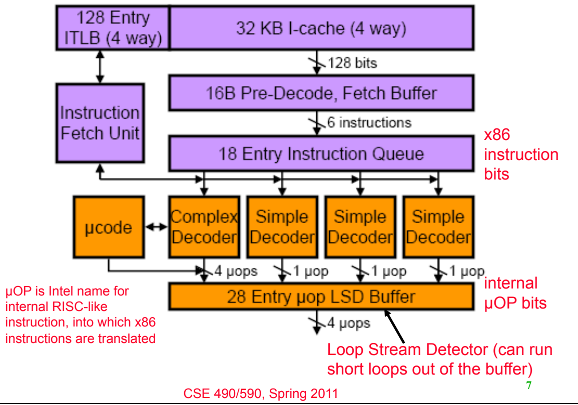
In-Order Commit

Out-of-Order Execution

2 SMT Threads per Core

Out-of-Order Completion

Front-End Instruction Fetch & Decode



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

Loop Stream Detector (can run short loops out of the buffer)

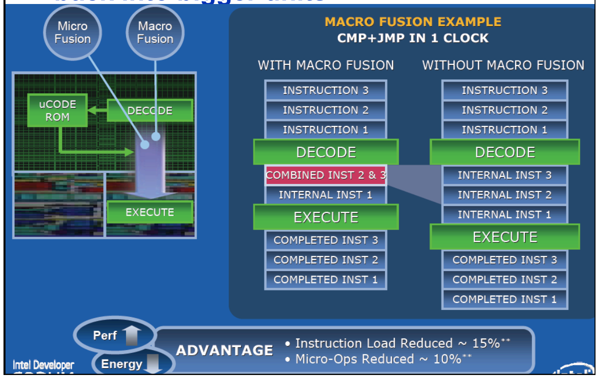
Branch Prediction

- Part of instruction fetch unit
- Several different types of branch predictor
  - Details not public
- Two-level BTB
- Loop count predictor
  - How many backwards taken branches before loop exit
  - (Also predictor for length of microcode loops, e.g., string move)
- Return Stack Buffer
  - Holds subroutine targets
  - Renames the stack buffer so that it is repaired after mispredicted returns
  - Separate return stack buffer for each SMT thread

x86 Decoding

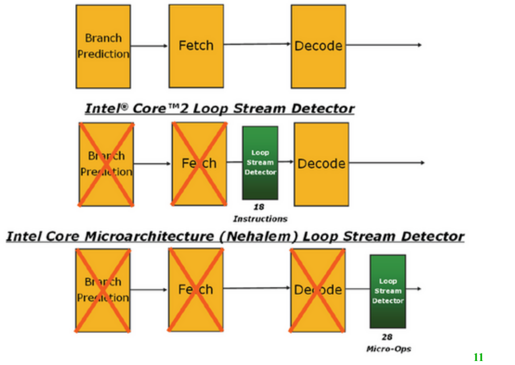
- Translate up to 4 x86 instructions into uOPS each cycle
- Only first x86 instruction in group can be complex (maps to 1-4 uOPS), rest must be simple (map to one uOP)
- Even more complex instructions, jump into microcode engine which spits out stream of uOPS

Split x86 in small uOPs, then fuse back into bigger units

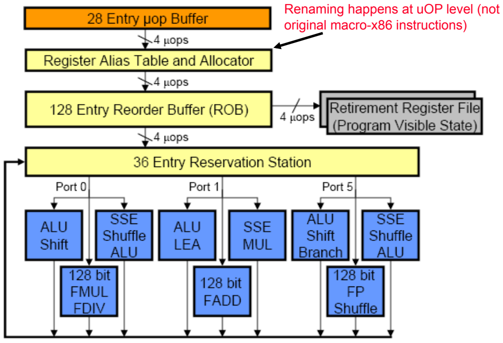


ADVANTAGE
- Instruction Load Reduced ~ 15%**
- Micro-Ops Reduced ~ 10%**

Loop Stream Detectors save Power



Out-of-Order Execution Engine

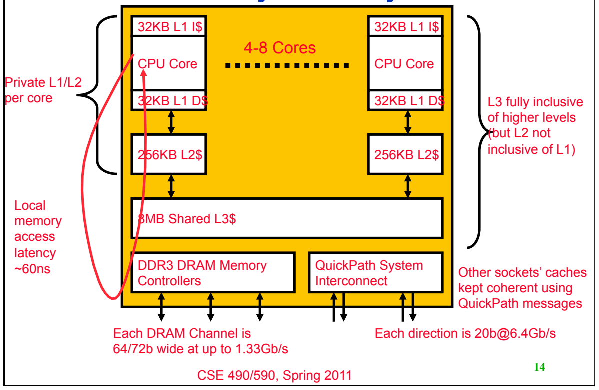


Renaming happens at uOP level (not original macro-x86 instructions)

SMT effects in OoO Execution Core

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

Nehalem Memory Hierarchy Overview



Private L1/L2 per core

Local memory access latency ~60ns

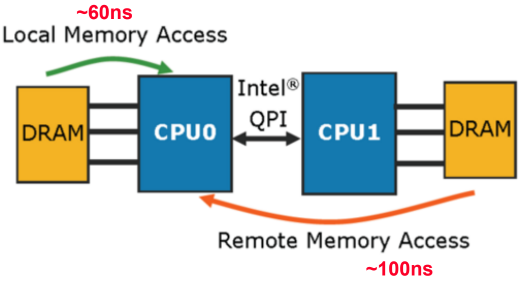
L3 fully inclusive of higher levels (but L2 not inclusive of L1)

Other sockets' caches kept coherent using QuickPath messages

Each DRAM Channel is 64/72b wide at up to 1.33Gb/s

Each direction is 20b@6.4Gb/s

All Sockets can Access all Data



~60ns

Local Memory Access

Remote Memory Access

~100ns

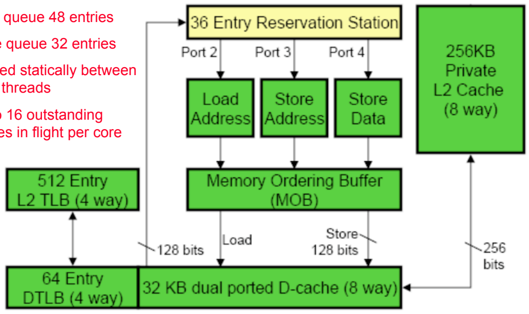
Core's Private Memory System

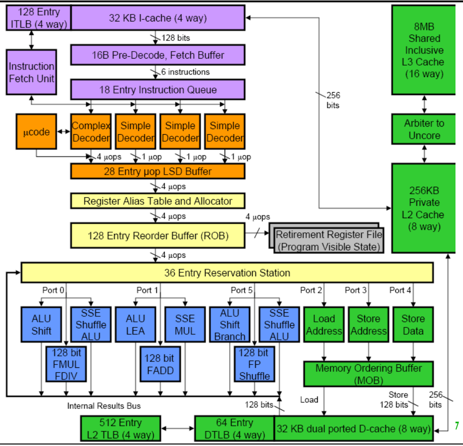
Load queue 48 entries

Store queue 32 entries

Divided statically between SMT threads

Up to 16 outstanding misses in flight per core





Cache Hierarchy Latencies

- L1 32KB 8-way, latency 4 cycles
- L2 256KB 8-way, latency <12 cycles
- L3 8MB, 16-way, latency 30-40 cycles
- DRAM, latency ~180-200 cycles

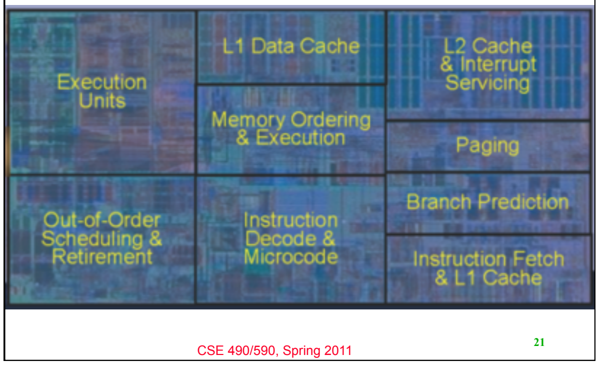
Nehalem Virtual Memory Details

- 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

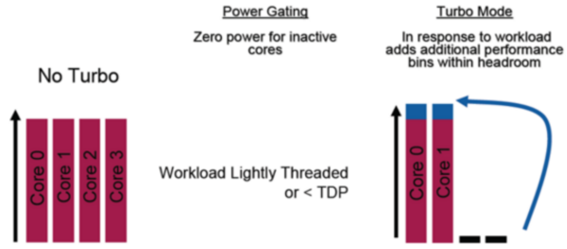
Virtualization Support

- TLB entries tagged with virtual machine and address space ID
  - No need to flush on context switches between VMs
- Hardware page table walker can walk guest-physical to host-physical mapping tables
  - Fewer traps to hypervisor

Core Area Breakdown


Execution Units
L1 Data Cache
L2 Cache & Interrupt Servicing
Memory Ordering & Execution
Paging
Out-of-Order Scheduling & Retirement
Instruction Decode & Microcode
Branch Prediction
Instruction Fetch & L1 Cache

(Nehalem) Turbo Mode


Power Gating

Zero power for inactive cores

Turbo Mode

In response to workload adds additional performance bins within headroom

No Turbo

Workload Lightly Threaded or < TDP

CSE 490/590 Administrivia

- Keyboards available for pickup at my office
- Project 2: less than 2 weeks left (Deadline 5/2)
  - Will have demo sessions
- No class on 5/2 (finish the project!)
- Final exam: Thursday 5/5, 11:45pm – 2:45pm
- Project 2 + Final = 55%

Acknowledgements

- These slides heavily contain material developed and copyright by
  - Krste Asanovic (MIT/UCB)
  - David Patterson (UCB)
- And also by:
  - Arvind (MIT)
  - Joel Emer (Intel/MIT)
  - James Hoe (CMU)
  - John Kubiatowicz (UCB)

- MIT material derived from course 6.823
- UCB material derived from course CS252