CS 61C:
Great Ideas in Computer Architecture
Case Studies: Server and Cellphone microprocessors

Instructors:
Krste Asanovic, Randy H. Katz
http://inst.eecs.Berkeley.edu/~cs61c/fa12

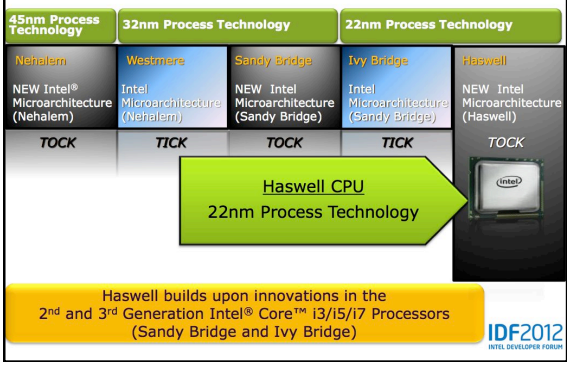
Today: Intel Haswell and smartphone/tablet processors

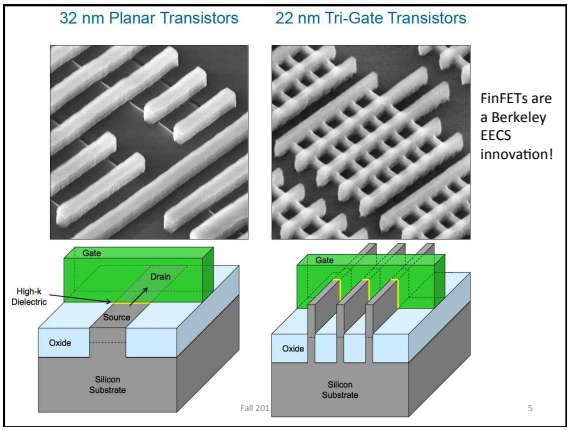
- This material is not on final exam!
- Intended for you to see modern day computer architectures.

Intel Haswell Core

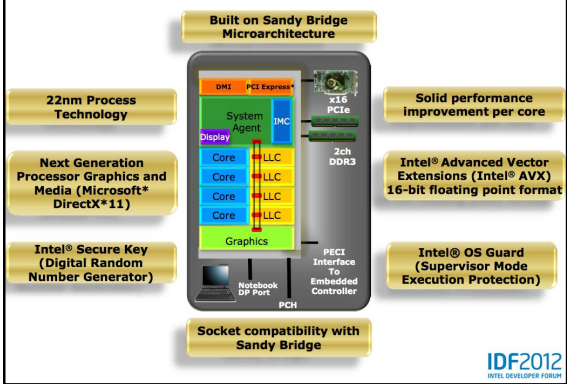
- Not yet in production, the next core after Ivy Bridge!
- Acknowledgements: Slides include material from Intel and David Kanter at Real World Technologies
  - Recommend site **`realworldtech.com`** for reading about new microprocessor architectures

Tick/Tock Development Model





Ivy Bridge (Tick): Recap



How to run x86 code fast?

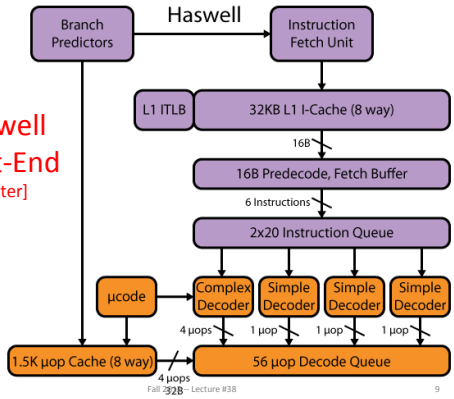
- x86 architecture evolved from 16-bit microprocessor designed for CISC microcoded implementation
  - 8086 introduced in 1978 (34 years old!)
  - Only older widely used ISA is IBM 360 architecture family introduced in 1964 (48 years old!)
- Typical instruction: Reg = Reg op M[Reg]
  - Two-address format
  - Register-memory operations
  - Few general-purpose registers (8 initially, 16 in 64-bit extension)
- Many complex instructions with repeat prefixes
  - String move in one instruction
- Variable-length instructions up to 16 bytes long
- Added one instruction/week over lifetime!!

7

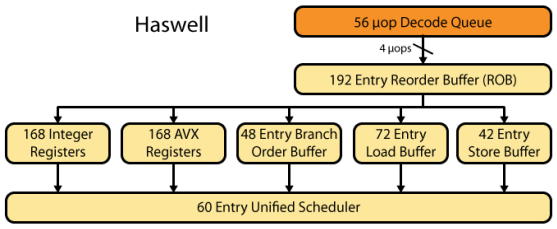
Convert CISC to RISC Dynamically!

- Translate complex x86 instructions into RISC-like micro-operations (µops) during instruction decode
  - e.g., "R ← R op Mem" translates into
  - load T1, Mem # Load from Mem into temp reg
  - R ← R op T1 # Operate using value in temp
- Execute µops using speculative out-of-order superscalar engine with register renaming
  - Both architectural and temporary registers are renamed from same pool
- Reconstruct whole x86 instructions during commit process to report exceptions precisely
- µop translation introduced in Pentium Pro family architecture (P6 family) in 1995, used in all subsequent x86 out-of-order processors

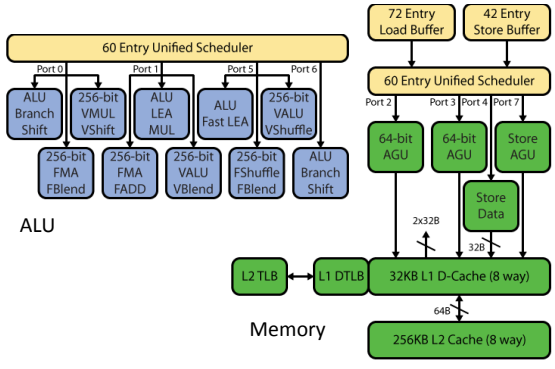
Haswell Front-End [Kanter]



Haswell Rename/Reorder [Kanter]



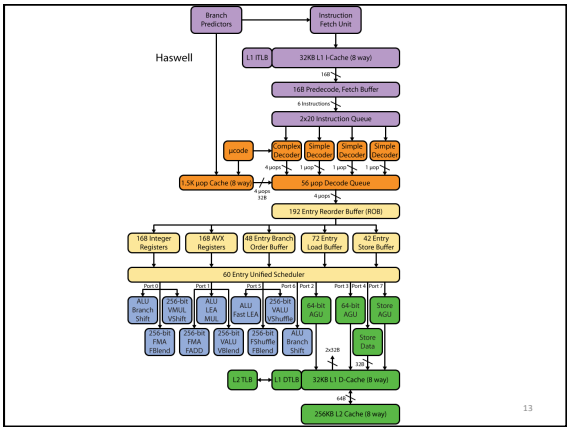
Haswell Execution [Kanter]



Core Cache Size/Latency/Bandwidth

Metric	Nehalem	Sandy Bridge	Haswell
L1 Instruction Cache	32K, 4-way	32K, 8-way	32K, 8-way
L1 Data Cache	32K, 8-way	32K, 8-way	32K, 8-way
Fastest Load-to-use	4 cycles	4 cycles	4 cycles
Load bandwidth	16 Bytes/cycle	32 Bytes/cycle (banked)	64 Bytes/cycle
Store bandwidth	16 Bytes/cycle	16 Bytes/cycle	32 Bytes/cycle
L2 Unified Cache	256K, 8-way	256K, 8-way	256K, 8-way
Fastest load-to-use	10 cycles	11 cycles	11 cycles
Bandwidth to L1	32 Bytes/cycle	32 Bytes/cycle	64 Bytes/cycle
L1 Instruction TLB	4K: 128, 4-way 2M/4M: 7/thread	4K: 128, 4-way 2M/4M: 8/thread	4K: 128, 4-way 2M/4M: 8/thread
L1 Data TLB	4K: 64, 4-way 2M/4M: 32, 4-way 1G: fractured	4K: 64, 4-way 2M/4M: 32, 4-way 1G: 4, 4-way	4K: 64, 4-way 2M/4M: 32, 4-way 1G: 4, 4-way
L2 Unified TLB	4K: 512, 4-way	4K: 512, 4-way	4K+2M shared: 1024, 8-way

All caches use 64-byte lines



New Gather Instructions

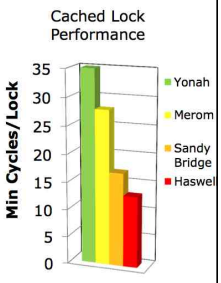
- Gathers 8 (SP/Dword) or 4 (DP/Qword) data elements into a vector register
- Sources
  - Base address
  - Indices (Offsets): vector register giving the offset of each element from the base address (with scale)
  - Mask register: tracks which elements need to be gathered
- Gather is complete when mask is all 0
  - Faults – Gather restarted when fault occurs
- Enables vectorization in new workloads
  - Simplifies code generation

Fundamental building block for sparse, indirect memory accesses, easing vectorization

IDF2012

Synchronization Improvements

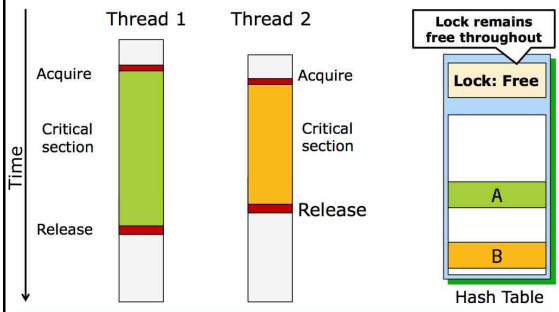
- **Improving existing primitives**
  - Faster LOCK-prefixed instructions
  - A focus in recent generations
- **Locks still limit concurrency**
  - Lock-protected critical sections
  - Needed for threading correctness
  - Tradeoff: correctness vs. performance
- **Intel® TSX**
  - Target lock granularity optimizations
  - Lock Elision
    - Execute without acquiring locks
    - Performance of fine-grained locks with effort of coarse-grained locks



Intel® TSX exposes parallelism through Lock Elision

IDF2012

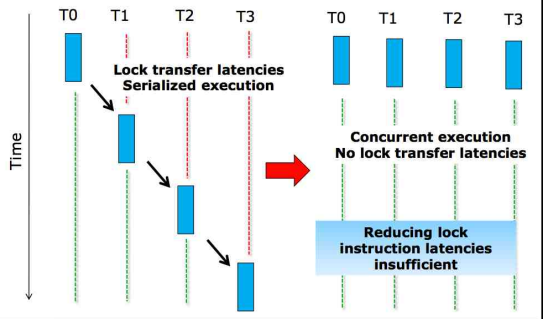
A Canonical Intel® TSX Execution



No serialization and no communication if no conflicts

IDF2012

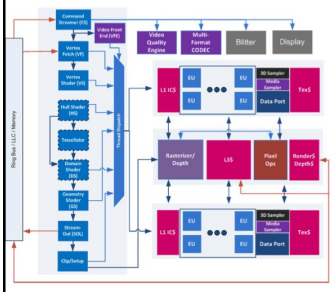
Benefit of Lock Elision



Exposes Concurrency & Eliminates Unnecessary Communication

IDF2012

Haswell Architecture



- **4th Generation Intel Core™ Processor Graphics**
- **Adds support of latest APIs**
  - **Microsoft* DirectX* 11.1**
  - **OpenCL* 1.2**
  - **OpenGL* 4.0**
- **Maintains Similar uArch Partitioning to 3rd Generation Intel Core processor graphics**

Haswell Performance enables new segments & Maximum Performance/Watt

IDF201

Administrivia

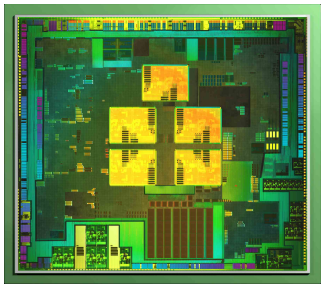
- Final review session with TAs
  - Wednesday December 5
  - 12:00pm-3:00pm
  - 1 Pimentel

Smartphone Processors

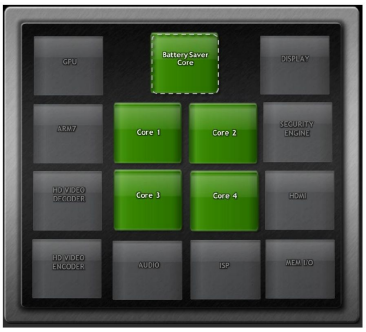
- Many companies and parts but some common features:
  - ARM ISA for application processors
  - Lots of dedicated accelerator blocks, especially image processors for cameras and GPUs for graphics
  - Only ~2W max power dissipation!

NVIDIA Tegra 3

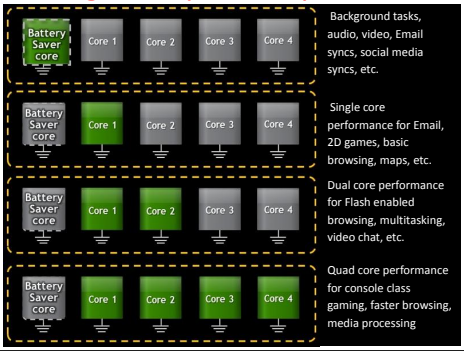
- Used in Nexus 7 and many other phones, tablets, and Audi cars…



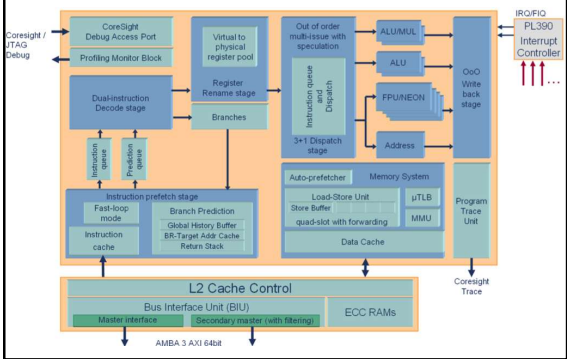
Tegra 3 Block Diagram



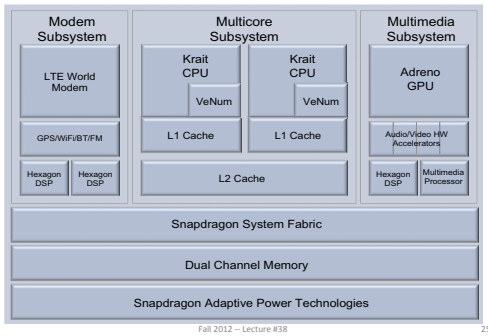
Tegra3 "4plus1" operation



ARM Cortex A9

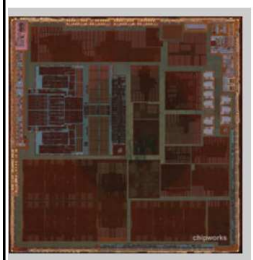


Qualcomm Snapdragon MSM8960

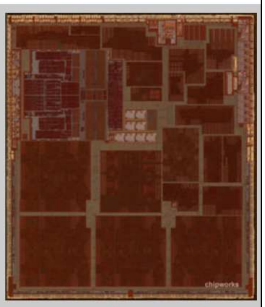


Apple A6 and A6X (32nm)

[Chipworks.com, 2012]



Apple A6 Die at Polysilicon (9.6 mm x 9.8 mm)

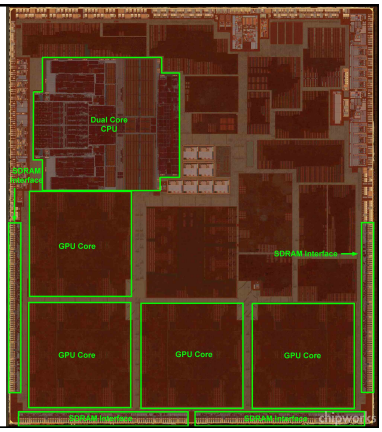


Apple A6X Die at Polysilicon (10.4 mm x 11.9 mm)

Apple A6X

[chipworks.com]

In iPad 4
76.8 GFLOPS Peak!



Summary

- Continual rapid change in architecture
  - Mobile and server processors include large and increasing number of processors on single chip
  - More specialized processors common
  - New architectural concepts (transactional memory)
- Covered basic ideas behind architectures in CS61C, but to learn more take CS152