CPU Architecture Overview

Varun Sampath

CIS 565 Spring 2012

Objectives

- Performance tricks of a modern CPU
  - Pipelining
  - Branch Prediction
  - Superscalar
  - Out-of-Order (OoO) Execution
  - Memory Hierarchy
  - Vector Operations
  - SMT
  - Multicore

What is a CPU anyways?

- Execute instructions
- Now so much more
  - Interface to main memory (DRAM)
  - I/O functionality
- Composed of transistors

Instructions

- Examples: arithmetic, memory, control flow

```
add r3,r4 -> r4
load [r4] -> r7
jz end
```

- Given a compiled program, minimize

$$\frac{cycles}{instruction} \times \frac{seconds}{cycle}$$

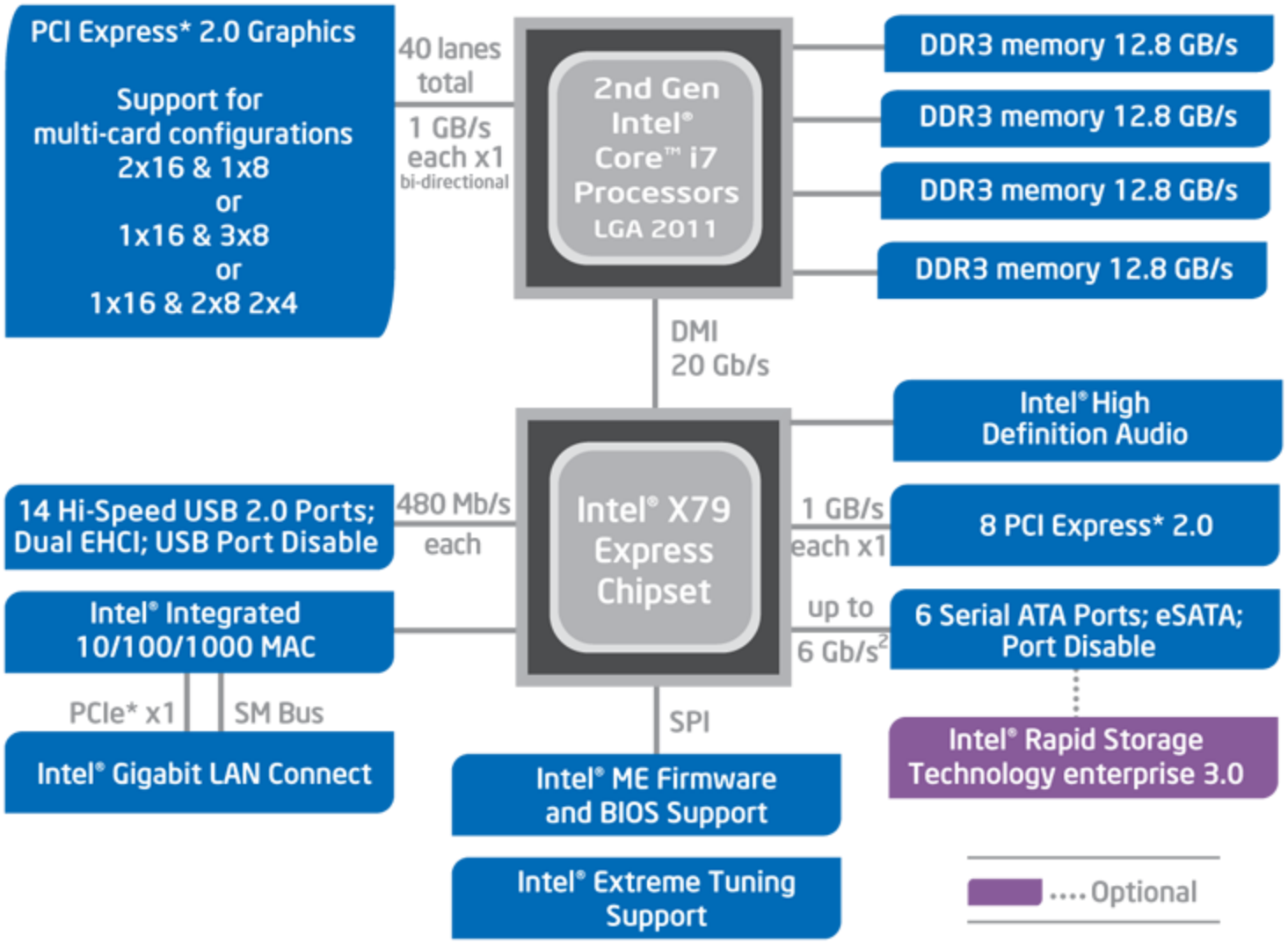
  - CPI (cycles per instruction) & clock period
  - Reducing one term may increase the other

Desktop Programs

- Lightly threaded
- Lots of branches
- Lots of memory accesses

	`vim`	`ls`
Conditional branches	13.6%	12.5%
Memory accesses	45.7%	45.7%
Vector instructions	1.1%	0.2%

Profiled with `psrun` on ENIAC



[1] Theoretical maximum bandwidth
[2] All SATA ports capable of 3 Gb/s. 2 ports capable of 6 Gb/s.

Intel® X79 Express Chipset Block Diagram

Source: intel.com

What is a Transistor?

- Approximation: a voltage-controlled switch
- Typical channel lengths (for 2012): 22-32nm

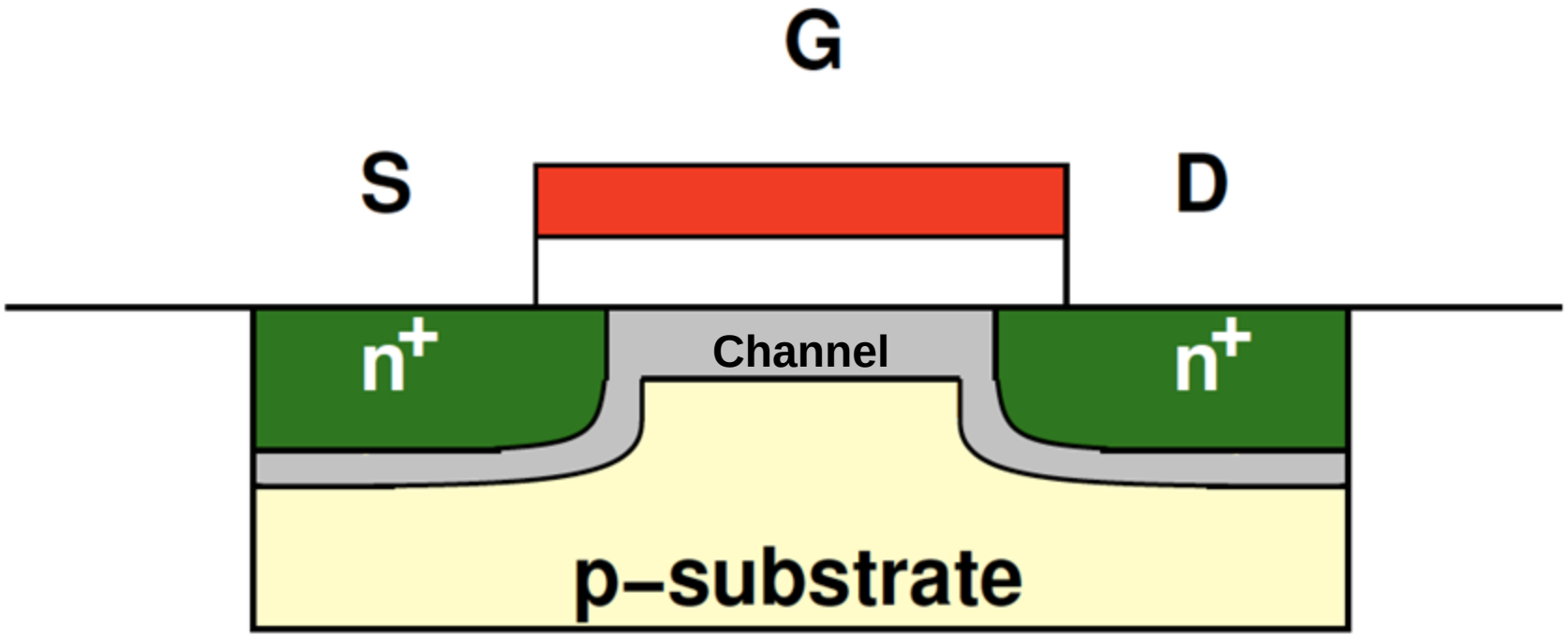
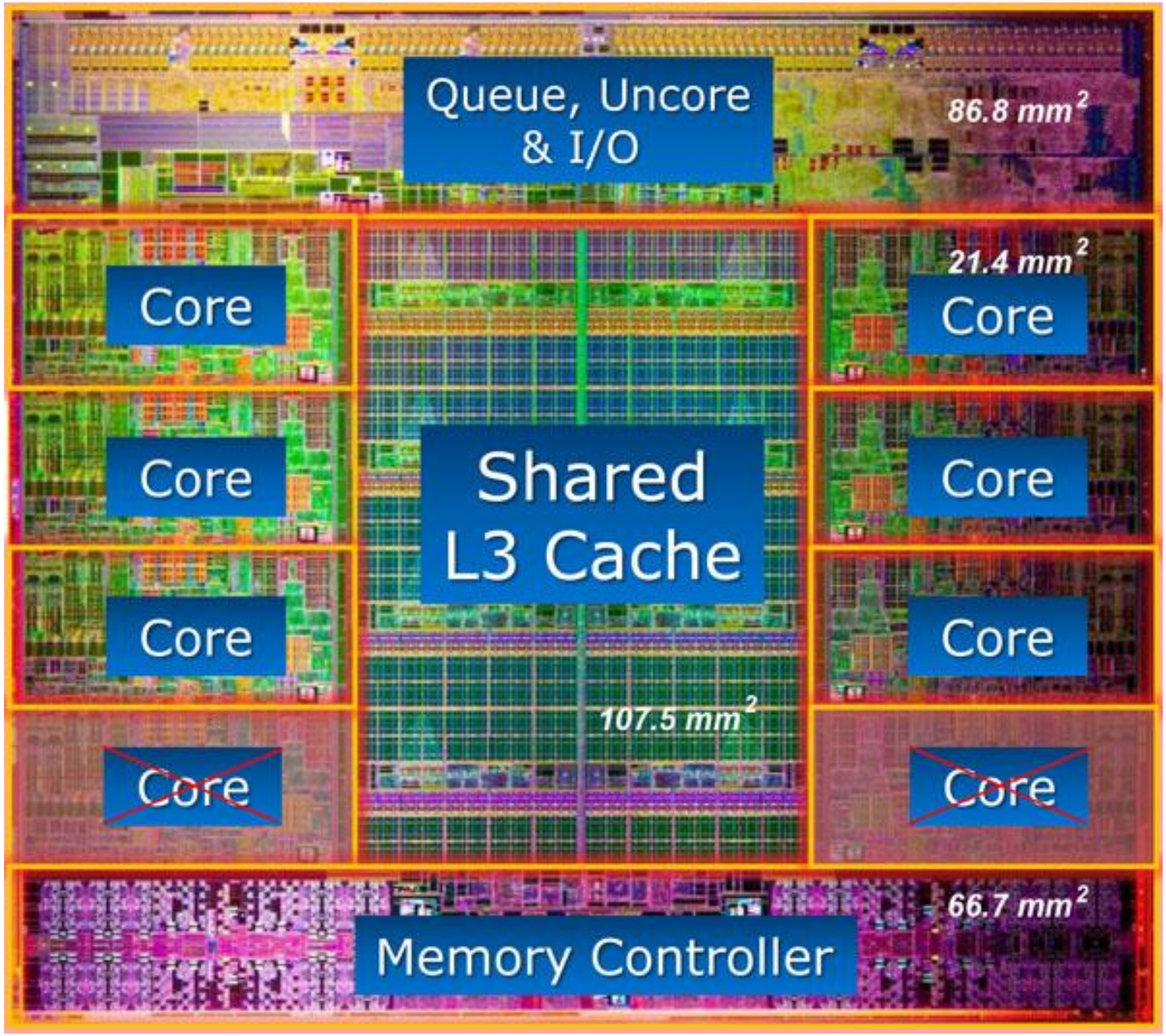


Image: Penn ESE370



Intel Core i7 3960X (Codename Sandy Bridge-E) – 2.27B transistors, Total Size 435mm²

Source: www.lostcircuits.com

A Simple CPU Core

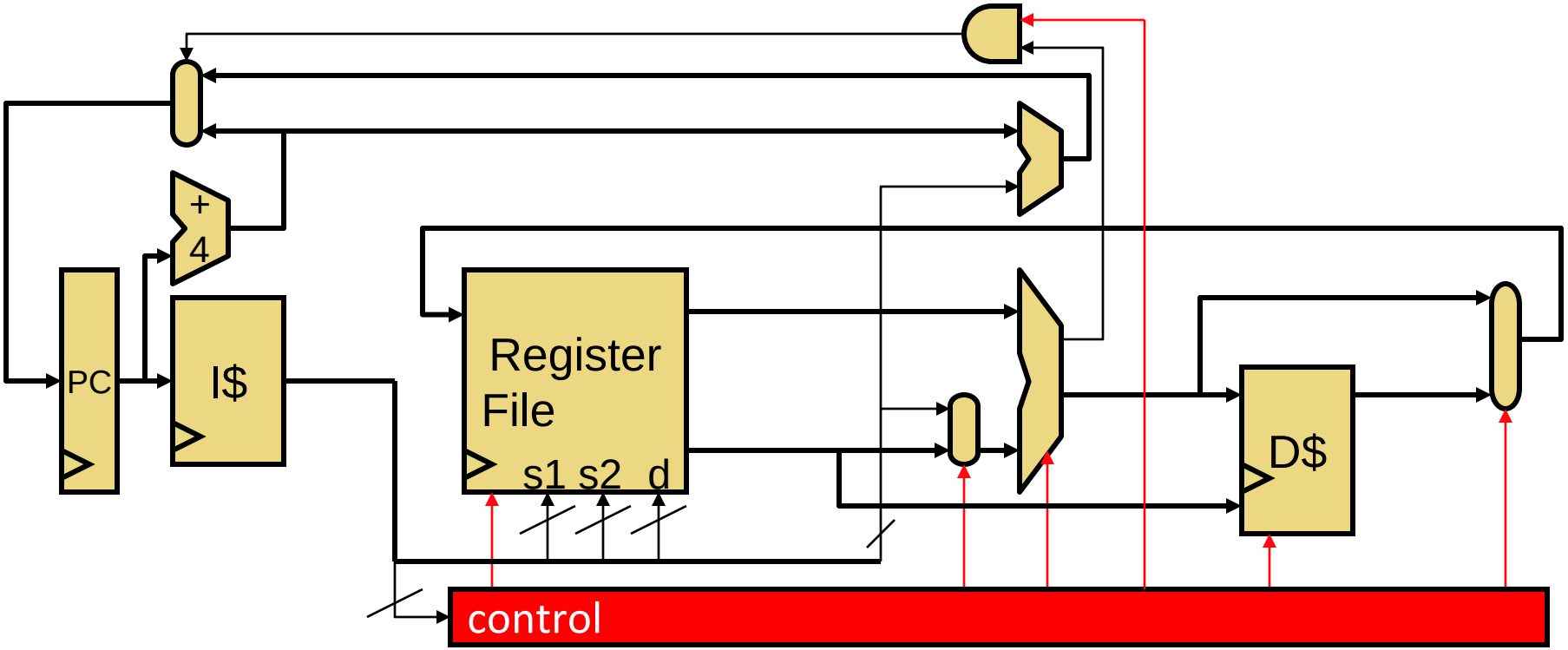
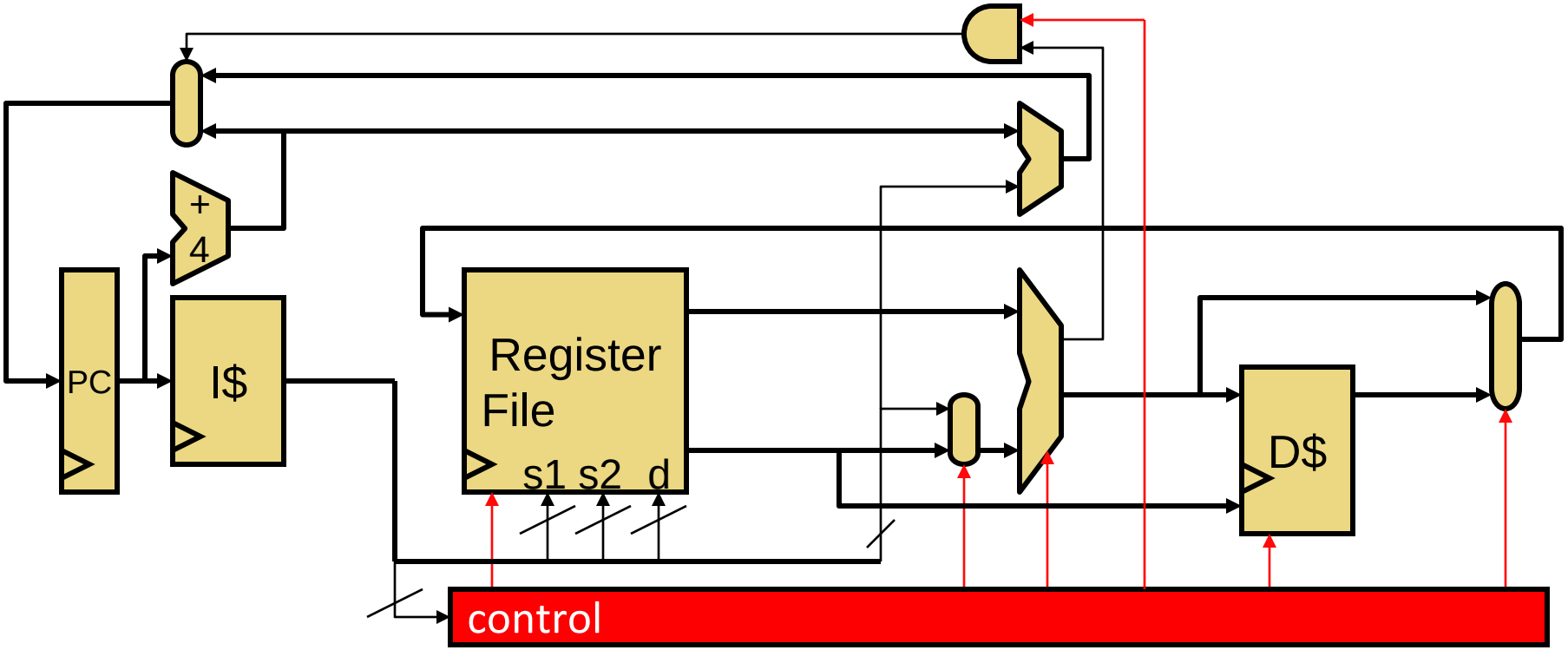


Image: Penn CIS501

A Simple CPU Core



Fetch → Decode → Execute → Memory → Writeback

Image: Penn CIS501

Pipelining

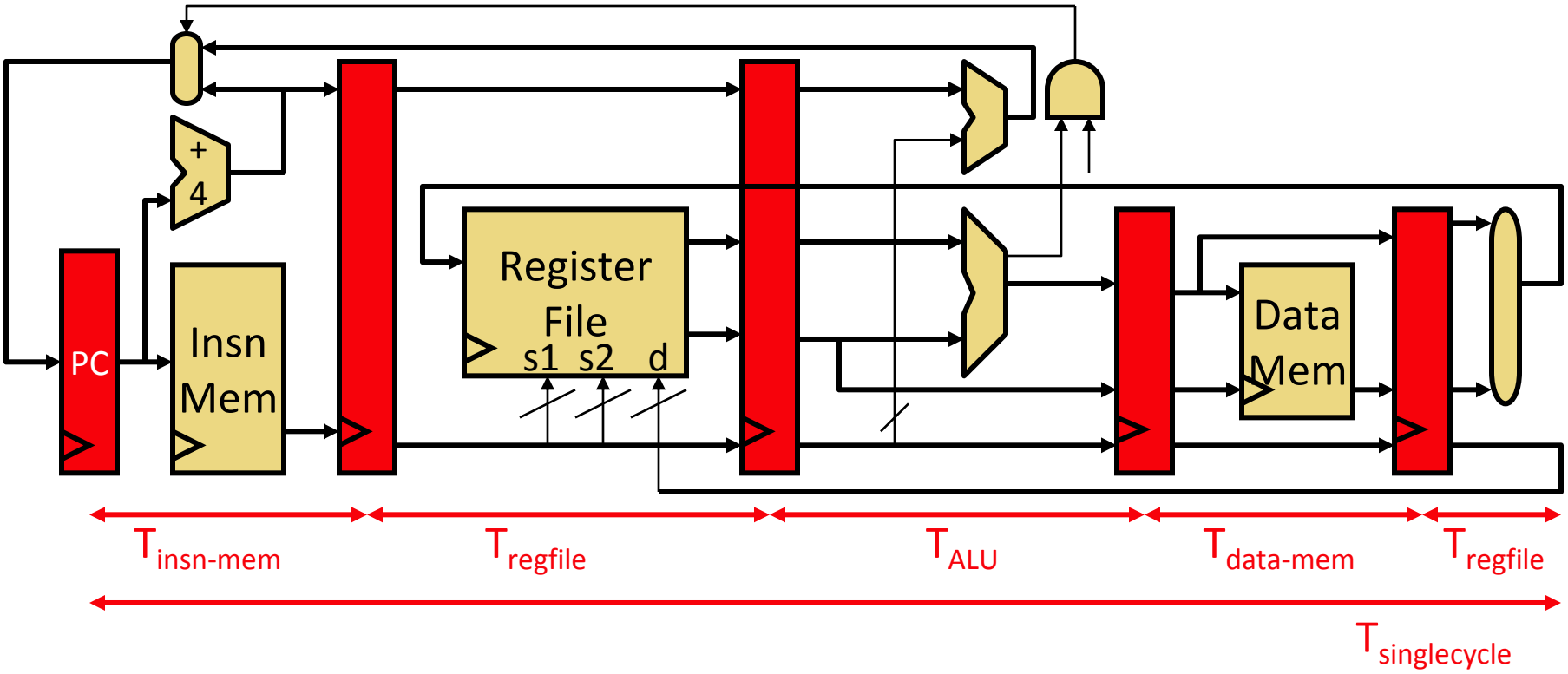


Image: Penn CIS501

Pipelining

- Capitalize on instruction-level parallelism (ILP)

+ Significantly reduced clock period

– Slight latency & area increase (pipeline latches)

? Dependent instructions

? Branches

- Alleged Pipeline Lengths:
  - Core 2: 14 stages
  - Pentium 4 (Prescott): > 20 stages
  - Sandy Bridge: in between

Bypassing

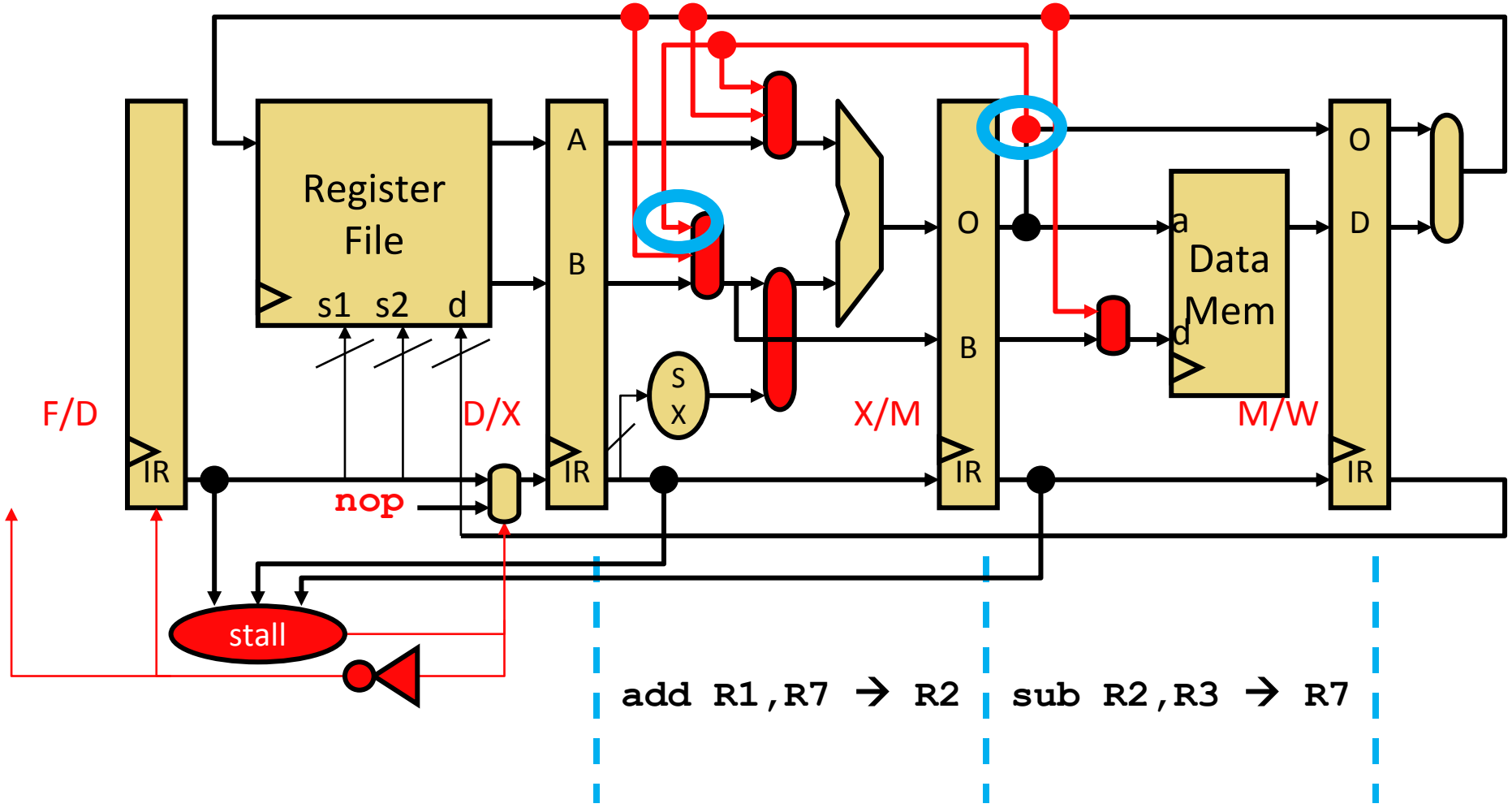


Image: Penn CIS501

Stalls

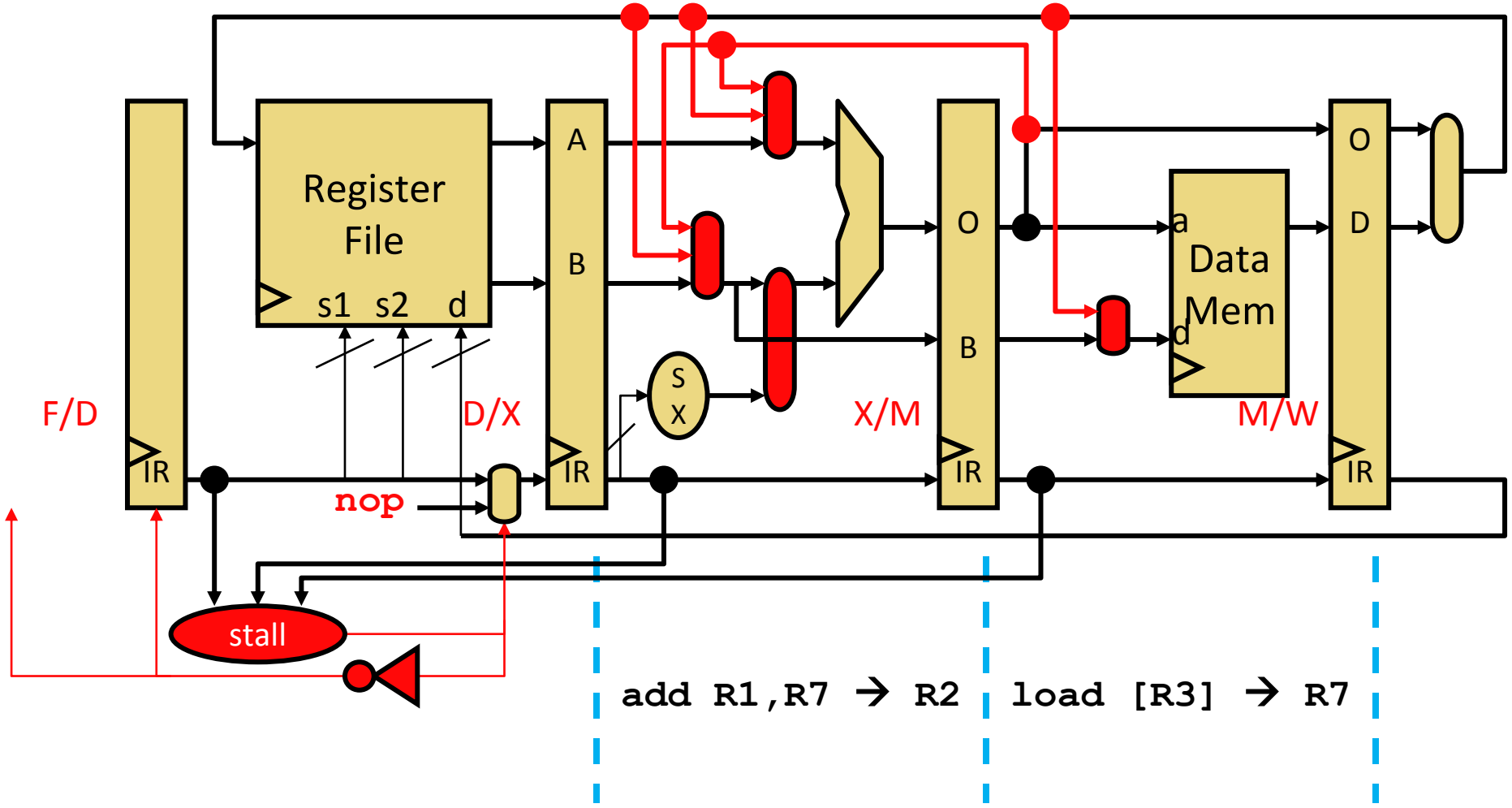


Image: Penn CIS501

Branches

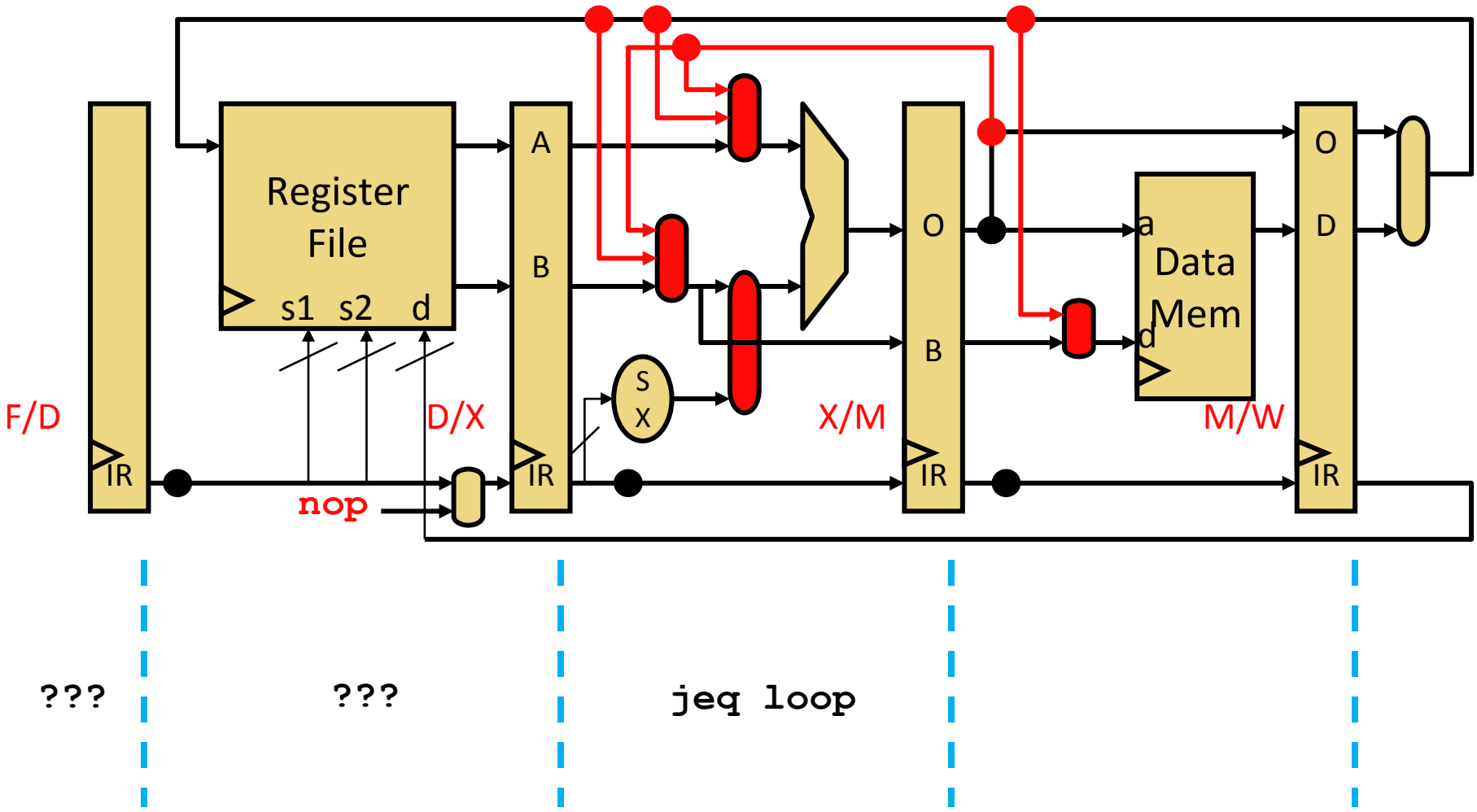


Image: Penn CIS501

Branch Prediction

- Guess what instruction comes next
- Based off branch history
- Example: two-level predictor with global history
  - Maintain history table of all outcomes for M successive branches
  - Compare with past N results (history register)
  - Sandy Bridge employs 32-bit history register

Branch Prediction

+ Modern predictors > 90% accuracy
  - Raise performance *and* energy efficiency (why?)
– Area increase
– Potential fetch stage latency increase

Another option: Predication

- Replace branches with conditional instructions

```
; if (r1==0) r3=r2
cmoveq r1, r2 -> r3
```

+ Avoids branch predictor
  - Avoids area penalty, misprediction penalty

– Avoids branch predictor
  - Introduces unnecessary `nop` if predictable branch

- GPUs also use predication

Improving IPC

- IPC (instructions/cycle) bottlenecked at 1 instruction / clock
- Superscalar – increase pipeline width

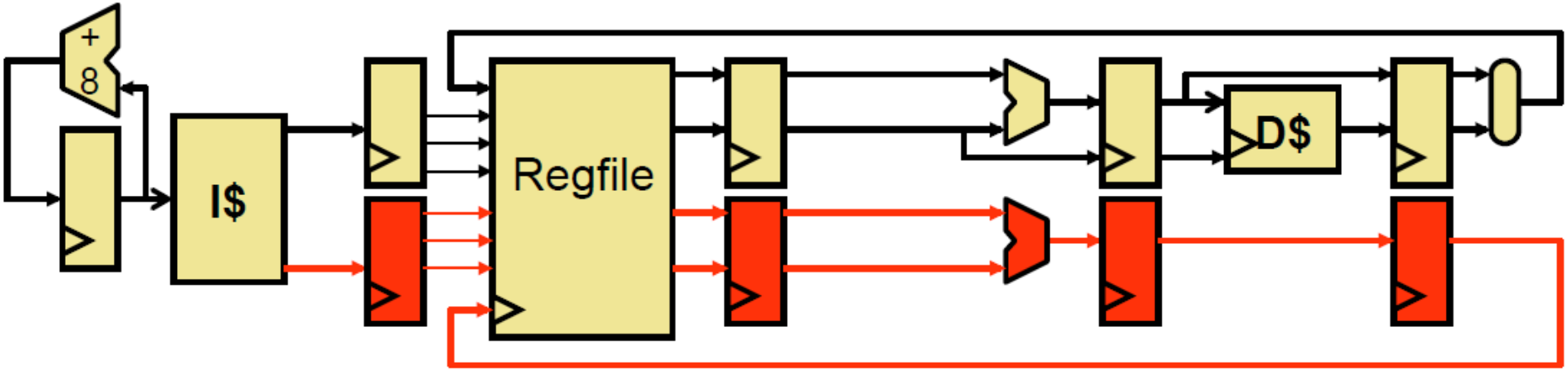


Image: Penn CIS371

Superscalar

+ Peak IPC now at N (for N-way superscalar)
  - Branching and scheduling impede this
  - Need some more tricks to get closer to peak (next)
– Area increase
  - Doubling execution resources
  - Bypass network grows at N^2
  - Need more register & memory bandwidth

Superscalar in Sandy Bridge

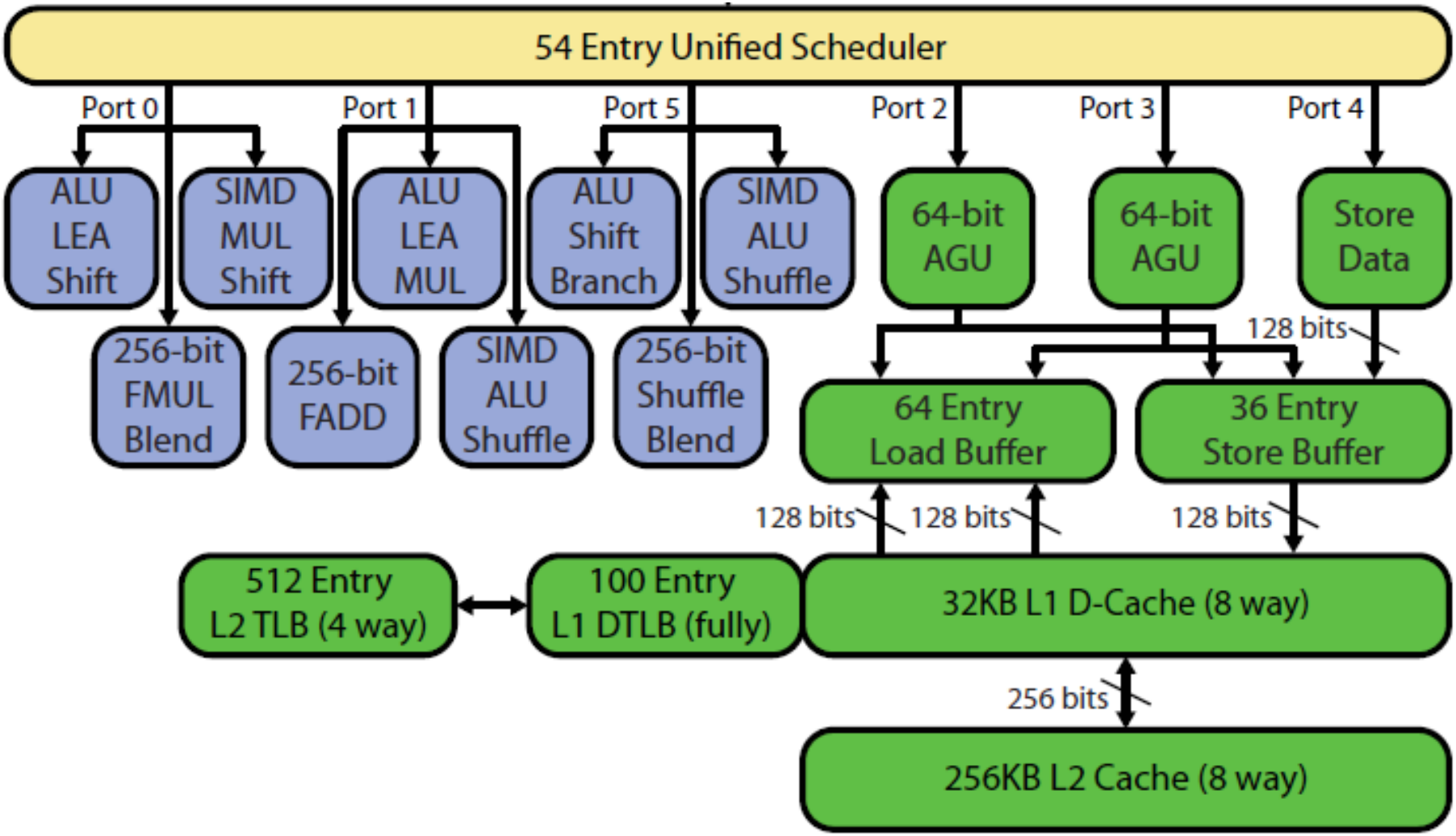


Image © David Kanter, RWT

Scheduling

- Consider instructions:

```
xor r1,r2 -> r3
add r3,r4 -> r4
sub r5,r2 -> r3
addi r3,1 -> r1
```

- `xor` and `add` are dependent (Read-After-Write, RAW)
- `sub` and `addi` are dependent (RAW)
- `xor` and `sub` are *not* (Write-After-Write, WAW)

Register Renaming

- How about this instead:

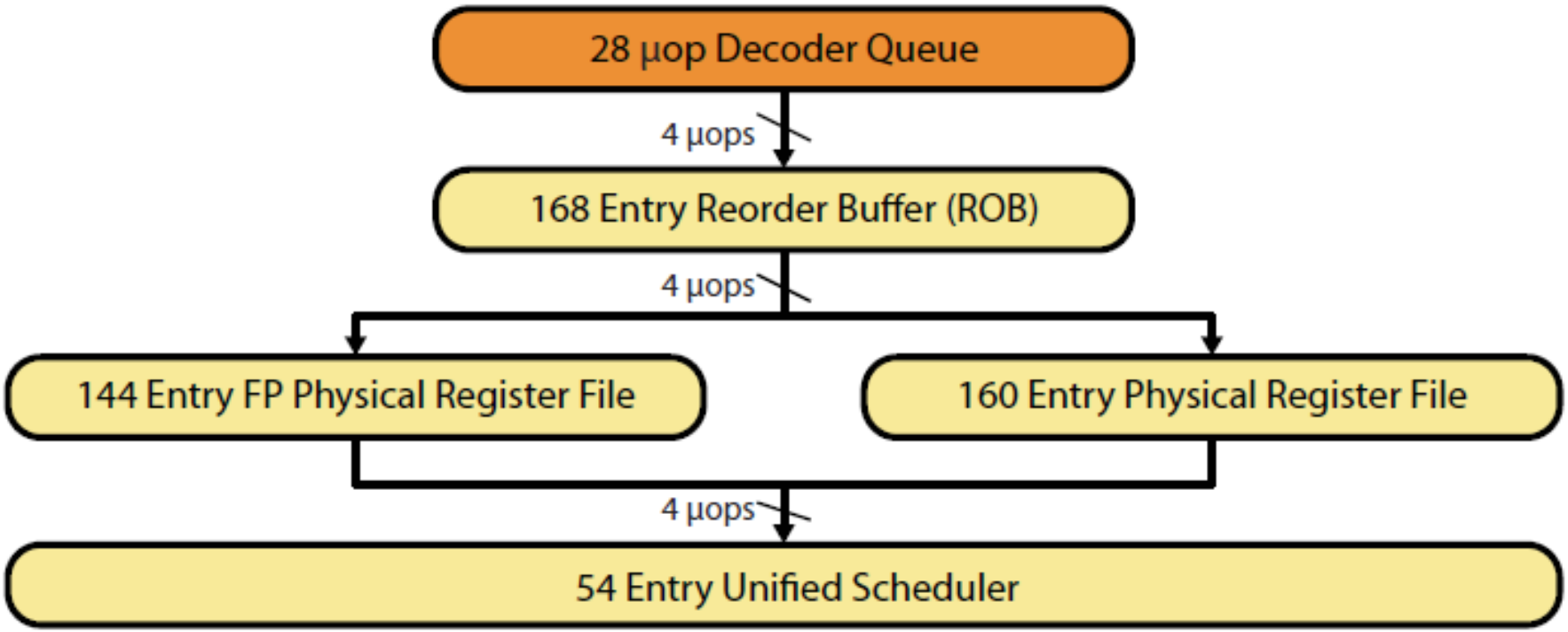
```
xor p1,p2 -> p6
add p6,p4 -> p7
sub p5,p2 -> p8
addi p8,1 -> p9
```

- `xor` and `sub` can now execute in parallel

Out-of-Order Execution

- Reordering instructions to maximize throughput
- Fetch → Decode → Rename → Dispatch → Issue → Register-Read → Execute → Memory → Writeback → Commit
- Reorder Buffer (ROB)
  - Keeps track of status for in-flight instructions
- Physical Register File (PRF)
- Issue Queue/Scheduler
  - Chooses next instruction(s) to execute

OoO in Sandy Bridge



Out-of-Order Execution

+ Brings IPC much closer to ideal
– Area increase
– Energy increase

- Modern Desktop/Mobile In-order CPUs
  - Intel Atom
  - ARM Cortex-A8 (Apple A4, TI OMAP 3)
  - Qualcomm Scorpion
- Modern Desktop/Mobile OoO CPUs
  - Intel Pentium Pro and onwards
  - ARM Cortex-A9 (Apple A5, NV Tegra 2/3, TI OMAP 4)
  - Qualcomm Krait

Memory Hierarchy

- Memory: the larger it gets, the slower it gets
- Rough numbers:

	Latency	Bandwidth	Size
SRAM (L1, L2, L3)	1-2ns	200GBps	1-20MB
DRAM (memory)	70ns	20GBps	1-20GB
Flash (disk)	70-90µs	200MBps	100-1000GB
HDD (disk)	10ms	1-150MBps	500-3000GB

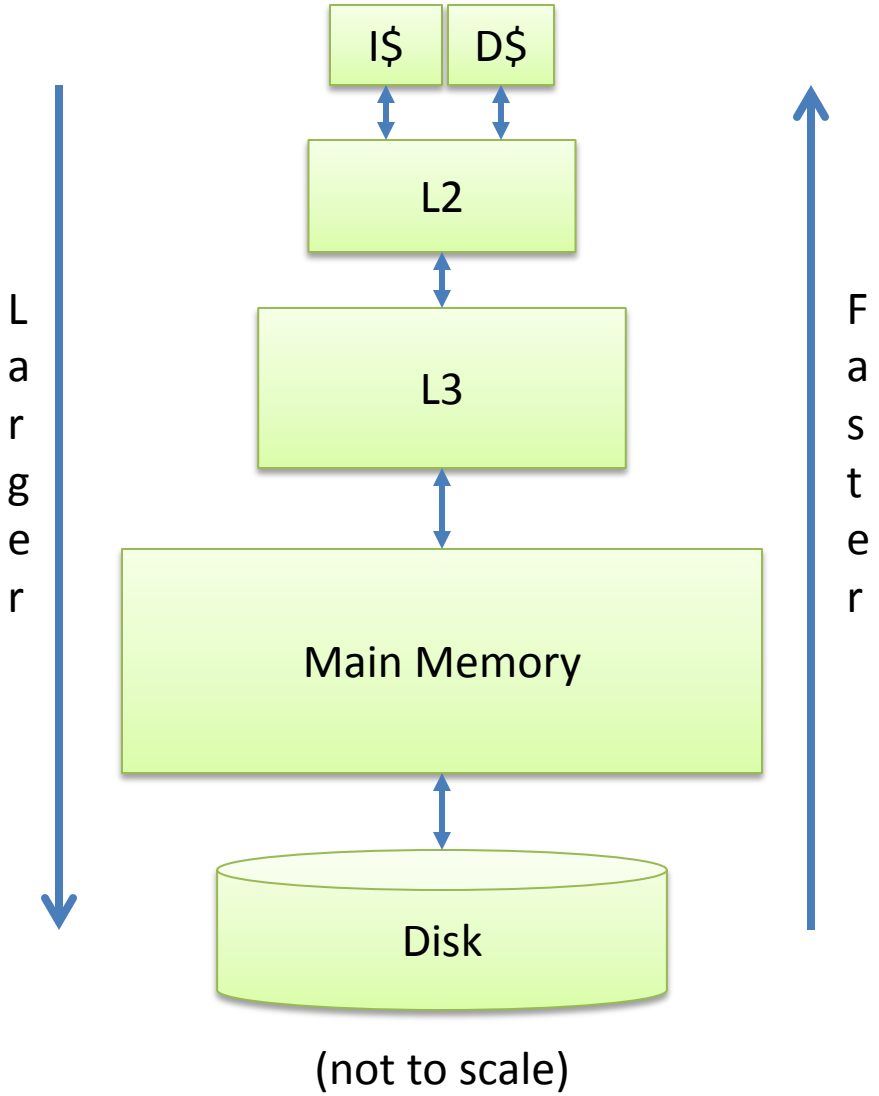
SRAM & DRAM latency, and DRAM bandwidth for Sandy Bridge from Lostcircuits
Flash and HDD latencies from AnandTech
Flash and HDD bandwidth from AnandTech Bench
SRAM bandwidth guesstimated.

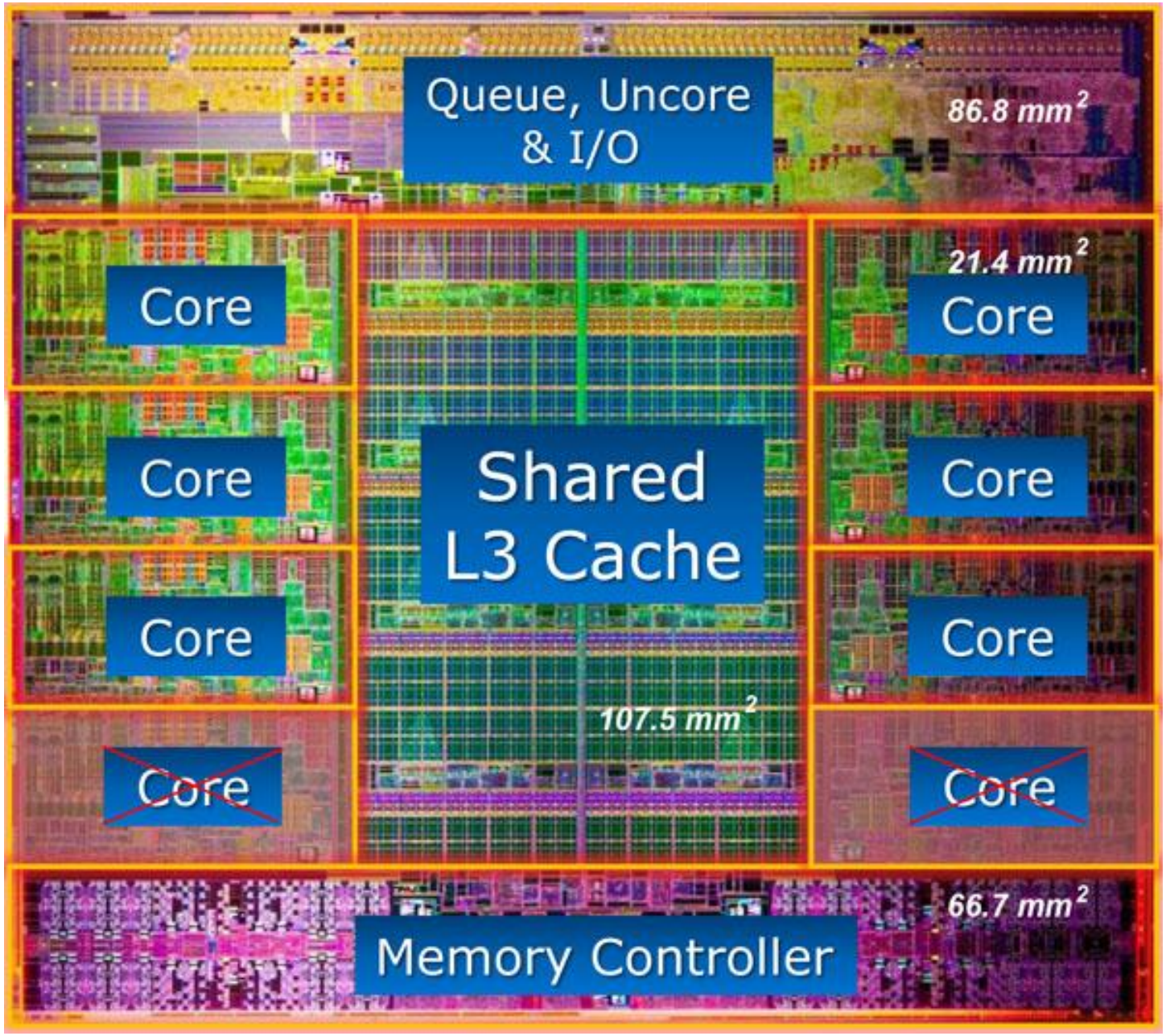
Caching

- Keep data you need close
- Exploit:
  - Temporal locality
    - Chunk just used likely to be used again soon
  - Spatial locality
    - Next chunk to use is likely close to previous

Cache Hierarchy

- Hardware-managed
  - L1 Instruction/Data caches
  - L2 unified cache
  - L3 unified cache
- Software-managed
  - Main memory
  - Disk





Intel Core i7 3960X – 15MB L3 (25% of die). 4-channel Memory Controller, 51.2GB/s total

Source: www.lostcircuits.com

Some Memory Hierarchy Design Choices

- Banking
  - Avoid multi-porting
- Coherency
- Memory Controller
  - Multiple channels for bandwidth

Parallelism in the CPU

- Covered Instruction-Level (ILP) extraction
  - Superscalar
  - Out-of-order
- Data-Level Parallelism (DLP)
  - Vectors
- Thread-Level Parallelism (TLP)
  - Simultaneous Multithreading (SMT)
  - Multicore

Vectors Motivation

```
for (int i = 0; i < N; i++)
    A[i] = B[i] + C[i];
```

CPU Data-level Parallelism

- Single Instruction Multiple Data (SIMD)
  - Let's make the execution unit (ALU) really wide
  - Let's make the registers really wide too

```
for (int i = 0; i < N; i+= 4) {
    // in parallel
    A[i] = B[i] + C[i];
    A[i+1] = B[i+1] + C[i+1];
    A[i+2] = B[i+2] + C[i+2];
    A[i+3] = B[i+3] + C[i+3];
}
```

Vector Operations in x86

- SSE2
  - 4-wide packed float and packed integer instructions
  - Intel Pentium 4 onwards
  - AMD Athlon 64 onwards
- AVX
  - 8-wide packed float and packed integer instructions
  - Intel Sandy Bridge
  - AMD Bulldozer

Thread-Level Parallelism

- Thread Composition
  - Instruction streams
  - Private PC, registers, stack
  - Shared globals, heap
- Created and destroyed by programmer
- Scheduled by programmer or by OS

Simultaneous Multithreading

- Instructions can be issued from multiple threads
- Requires partitioning of ROB, other buffers

+ Minimal hardware duplication

+ More scheduling freedom for OoO

– Cache and execution resource contention can reduce single-threaded performance

Multicore

- Replicate full pipeline
- Sandy Bridge-E: 6 cores

+ Full cores, no resource sharing other than last-level cache

+ Easier way to take advantage of Moore's Law

– Utilization

Locks, Coherence, and Consistency

- **Problem**: multiple threads reading/writing to same data
- A solution: Locks
  - Implement with test-and-set, load-link/store-conditional instructions
- **Problem**: Who has the correct data?
- A solution: cache coherency protocol
- **Problem**: What is the correct data?
- A solution: memory consistency model

Conclusions

- CPU optimized for sequential programming
  - Pipelines, branch prediction, superscalar, OoO
  - Reduce execution time with high clock speeds and high utilization
- Slow memory is a constant problem
- Parallelism
  - Sandy Bridge-E great for 6-12 active threads
  - How about 12,000?

References

- Milo Martin, Penn CIS501 Fall 2011 http://www.seas.upenn.edu/~cis501
- David Kanter, “Intel's Sandy Bridge Microarchitecture.” 9/25/10. http://www.realworldtech.com/page.cfm?ArticleID=RWT091810191937
- Agner Fog, “The microarchitecture of Intel, AMD and VIA CPUs.” 6/8/2011. http://www.agner.org/optimize/microarchitecture.pdf

Bibliography

- Jon Stokes' articles introducing pipelining: 1, 2
- CMOV discussion on Mozilla mailing list
- Herb Sutter, "The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software." link