（1）CPU体系架构概述

周　斌 @ NVIDIA & USTC 2015年8月

看图识芯片

本节课程

- 了解现代CPU的架构和性能优化
  - 流水线 Pipelining
  - 分支预测 Branch Prediction
  - 超标量 Superscalar
  - 乱序执行 Out-of-Order (OoO) Execution
  - 存储器层次 Memory Hierarchy
  - 矢量操作 Vector Operations
  - 多核处理 Multi-Core

什么是CPU?

- 执行指令、处理数据的器件
  - 完成基本的逻辑和算术指令
- 现在增加了复杂功能
  - 内存接口
  - 外部设备接口
- 包含大量晶体管
  - 非常多（上亿...）

指令

举例:

算术: add r3,r4 -> r4

访存: load [r4] -> r7

控制: jz end

对于一个编译好的程序，最优化目标:

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

CPI (每条指令的时钟数) & 时钟周期

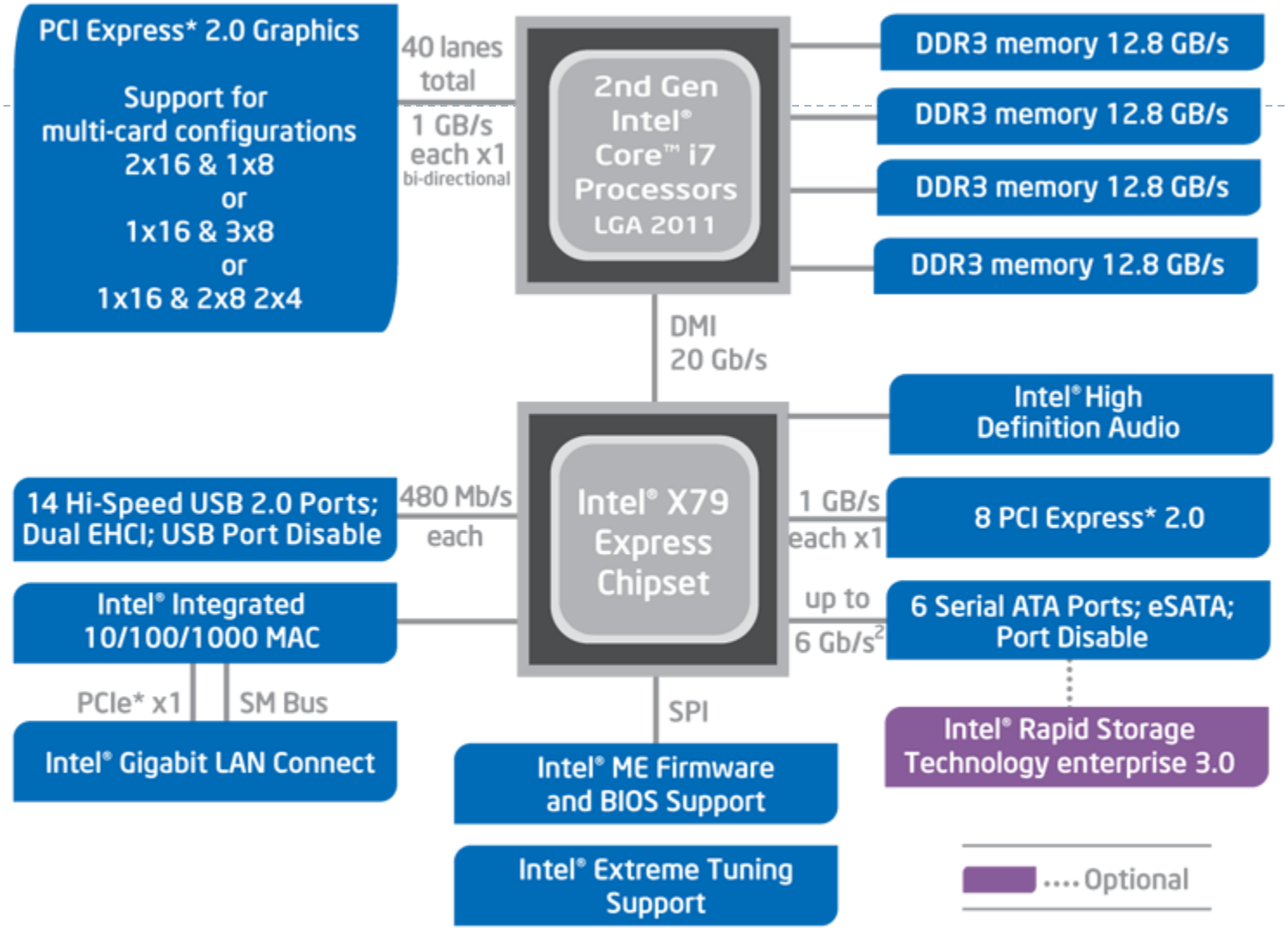
这两个指标彼此并不独立。

桌面应用Desktop Programs

- 轻量级进程，少量线程 Lightly threaded
- 大量分支和交互操作 Lots of branches
- 大量的存储器访问 Lots of memory accesses
- 真正用于数值运算的指令很少

	vim	ls
分支Conditional branches	13.6%	12.5%
访存Memory accesses	45.7%	45.7%
矢量运算Vector instructions	1.1%	0.2%

on Linux



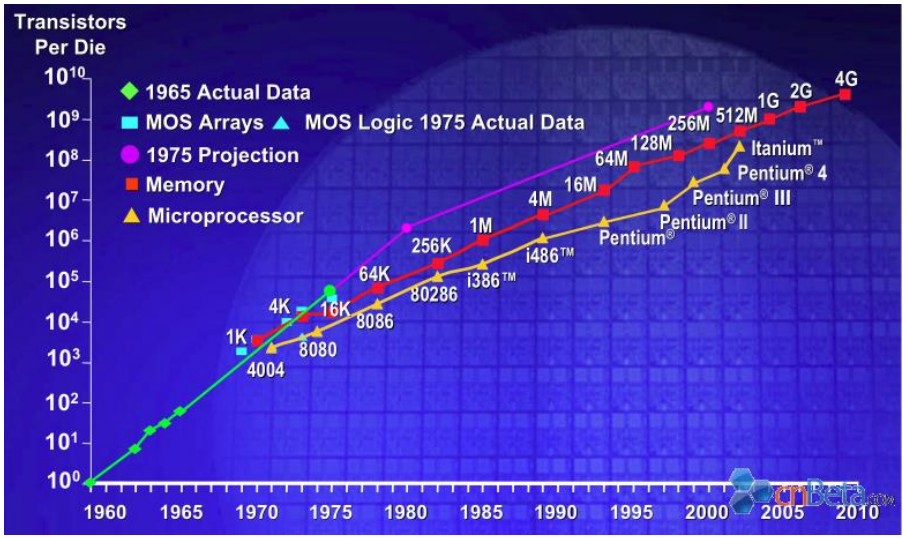
[1] Theoretical maximum bandwidth
[2] All SATA ports capable of 3 Gb/s. 2 ports capable of 6 Gb/s.

Intel® X79 Express Chipset Block Diagram

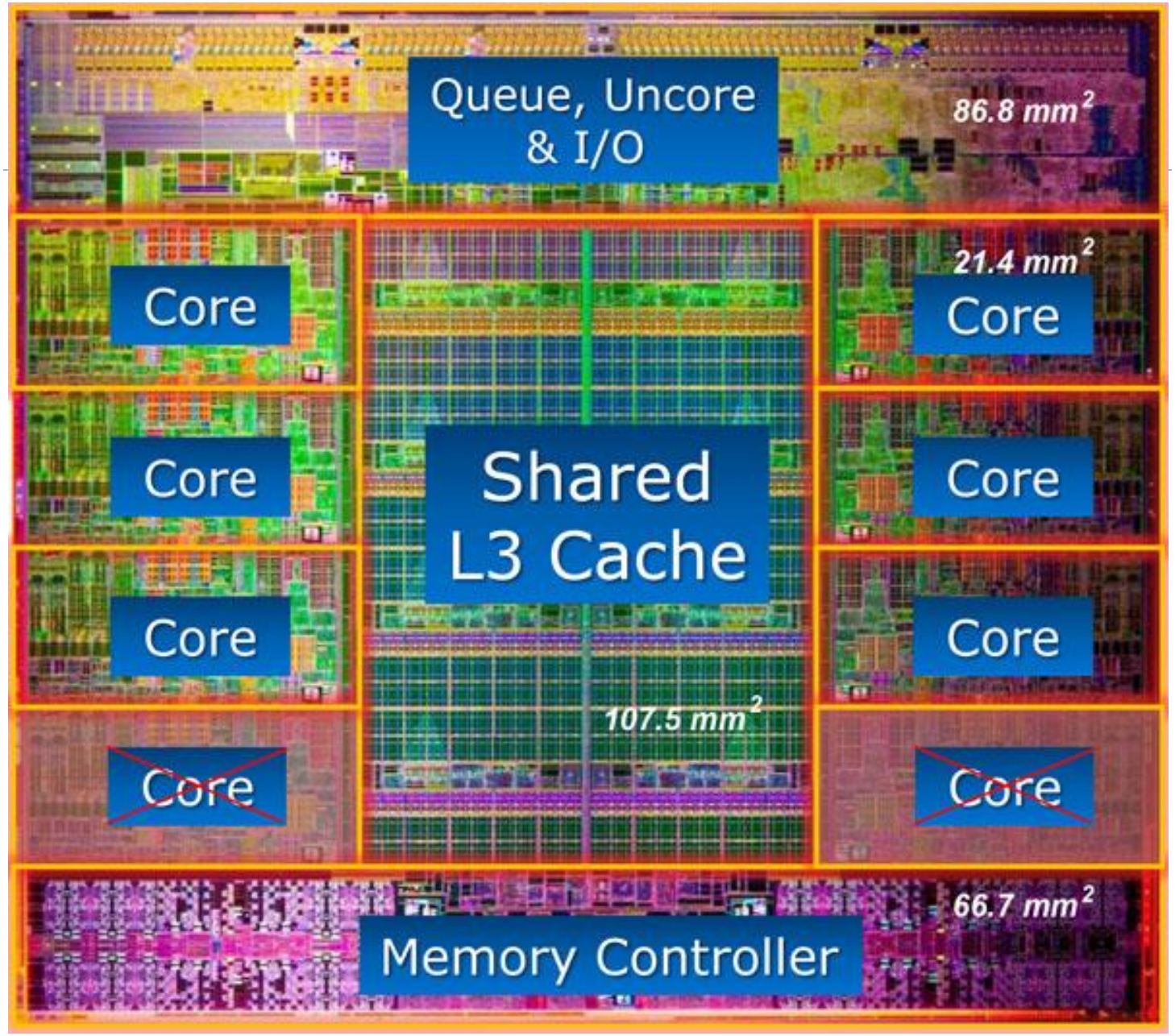
Source: intel.com

摩尔定律Moore’s Law

- 芯片的集成密度每2年翻翻，成本下降一半.
- “The complexity for minimum component costs has increased at a rate of roughly a factor of two per year”
- 这些晶体管都干什么了?
- What do we do with our transistor budget?



Source: intel.com



Intel Core i7 3960X (Codename Sandy Bridge-E) – 2.27B transistors, Total Size 435mm^2

Source: www.lostcircuits.com

简单的CPU结构图

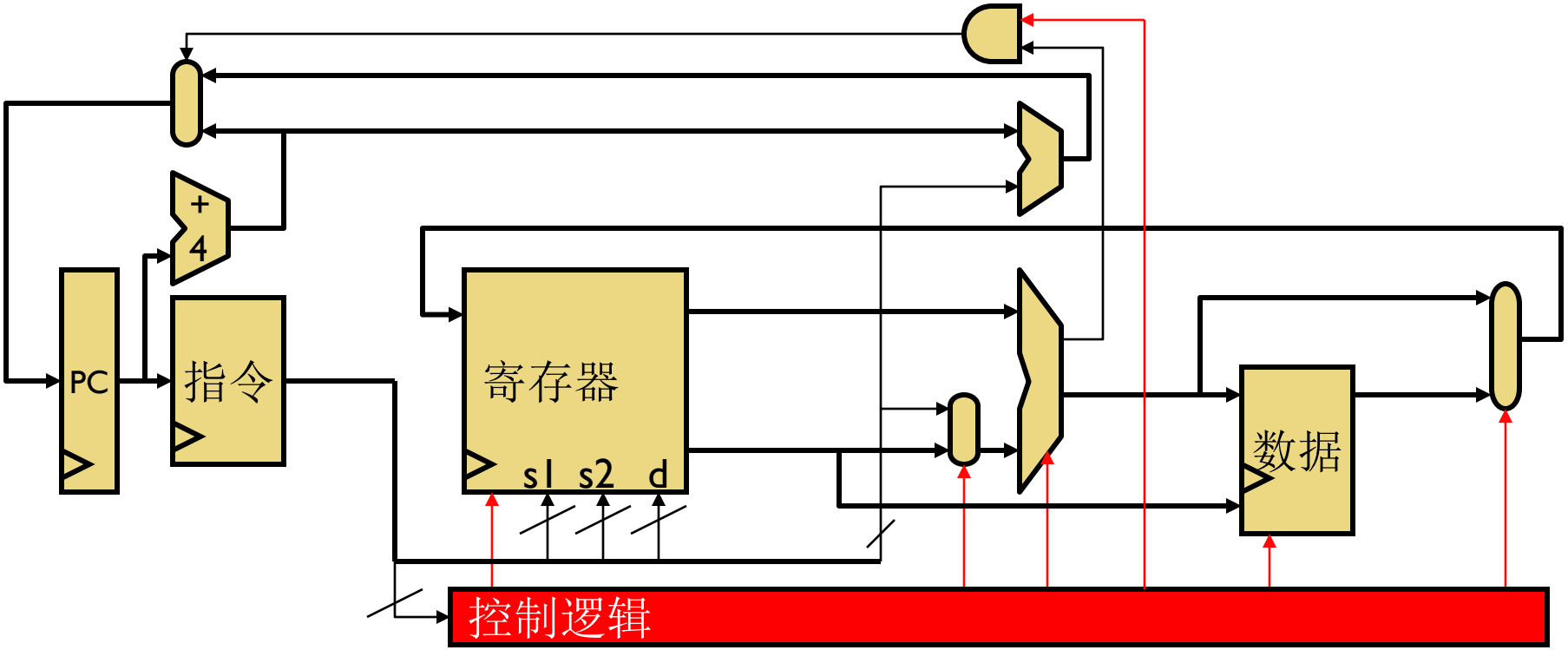
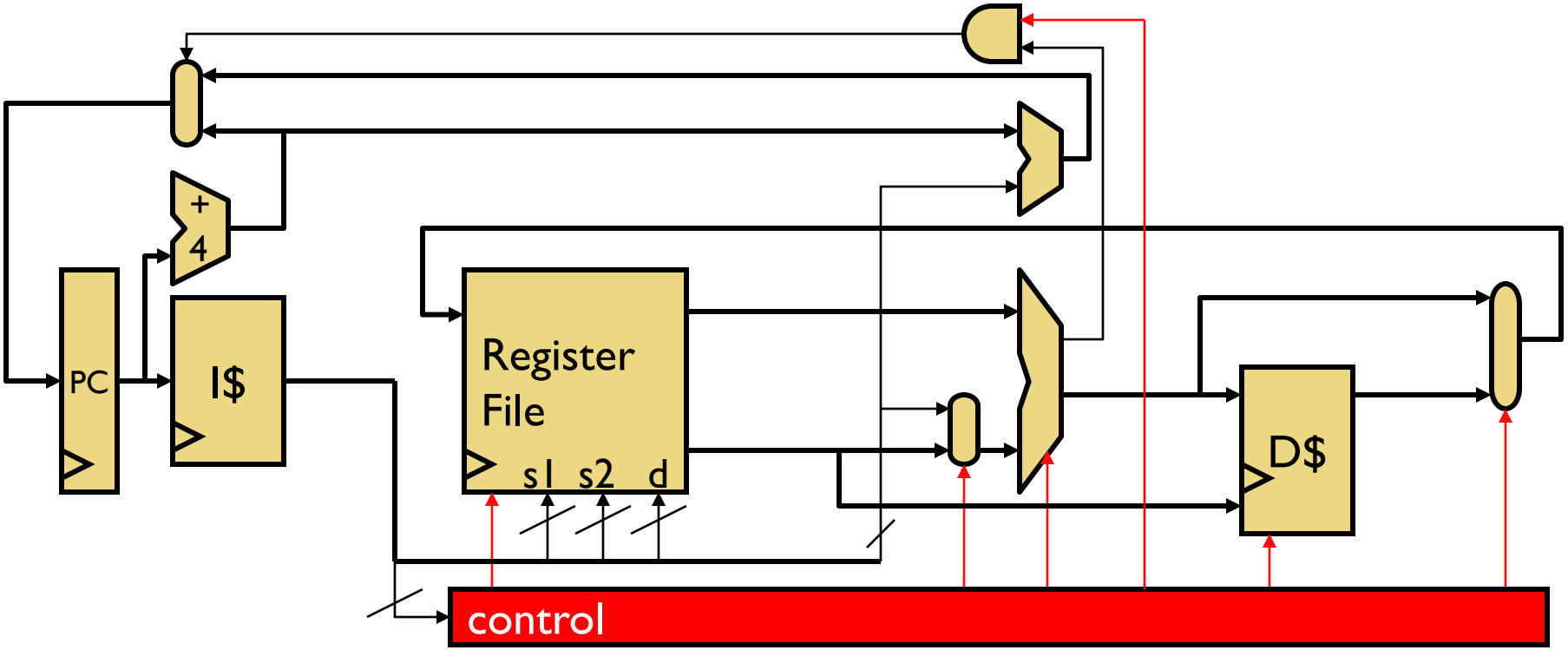


Image: Penn CIS501

简单的CPU



Fetch → Decode → Execute → Memory → Writeback

取指 → 译码 → 执行 → 访存 → 写回

Image: Penn CIS501

流水线 Pipeline

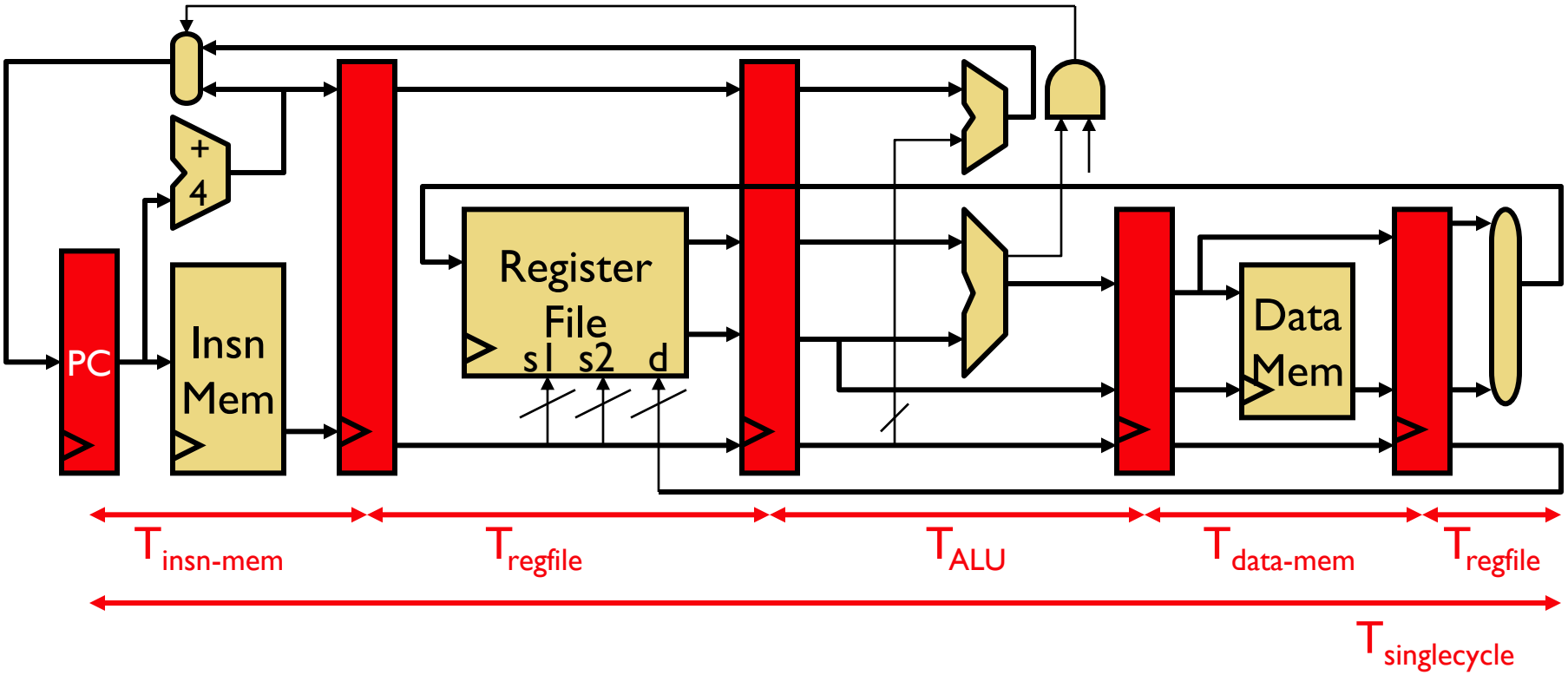


Image: Penn CIS501

Pipelining

- 利用指令级并行 instruction-level parallelism (ILP)

+ 极大的减小时钟周期

Significantly reduced clock period

– 增加一些延迟和芯片面积

Slight latency & area increase (pipeline latches)

带来的问题:

? 具有依赖关系的指令?

分支怎么处理?

- 流水线长度: Alleged Pipeline Lengths:
  - Core 2: 14 级
  - Pentium 4 (Prescott): > 20 级
  - Sandy Bridge: 中间

旁路Bypassing

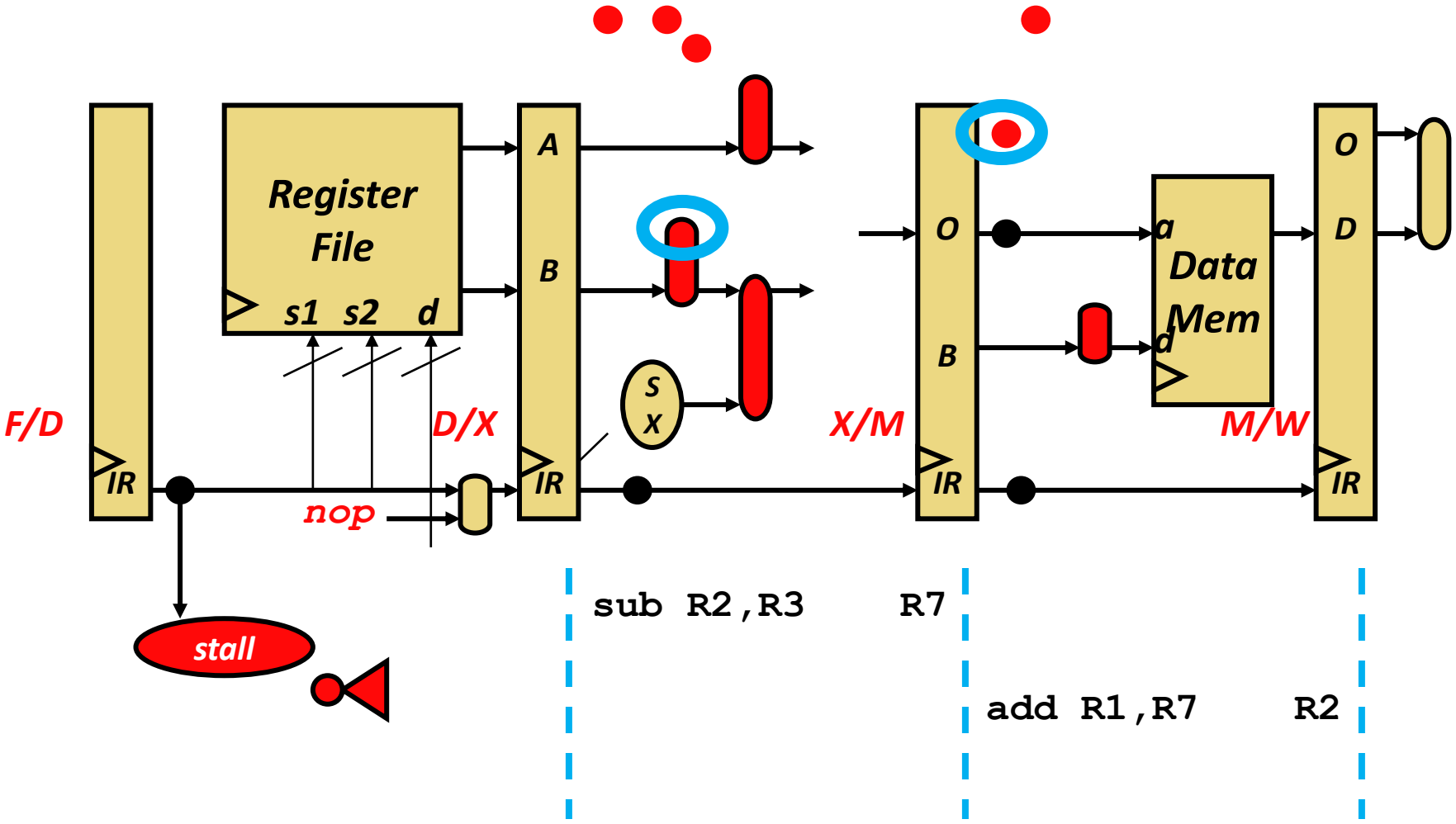


Image: Penn CIS501

停滞 Stalls

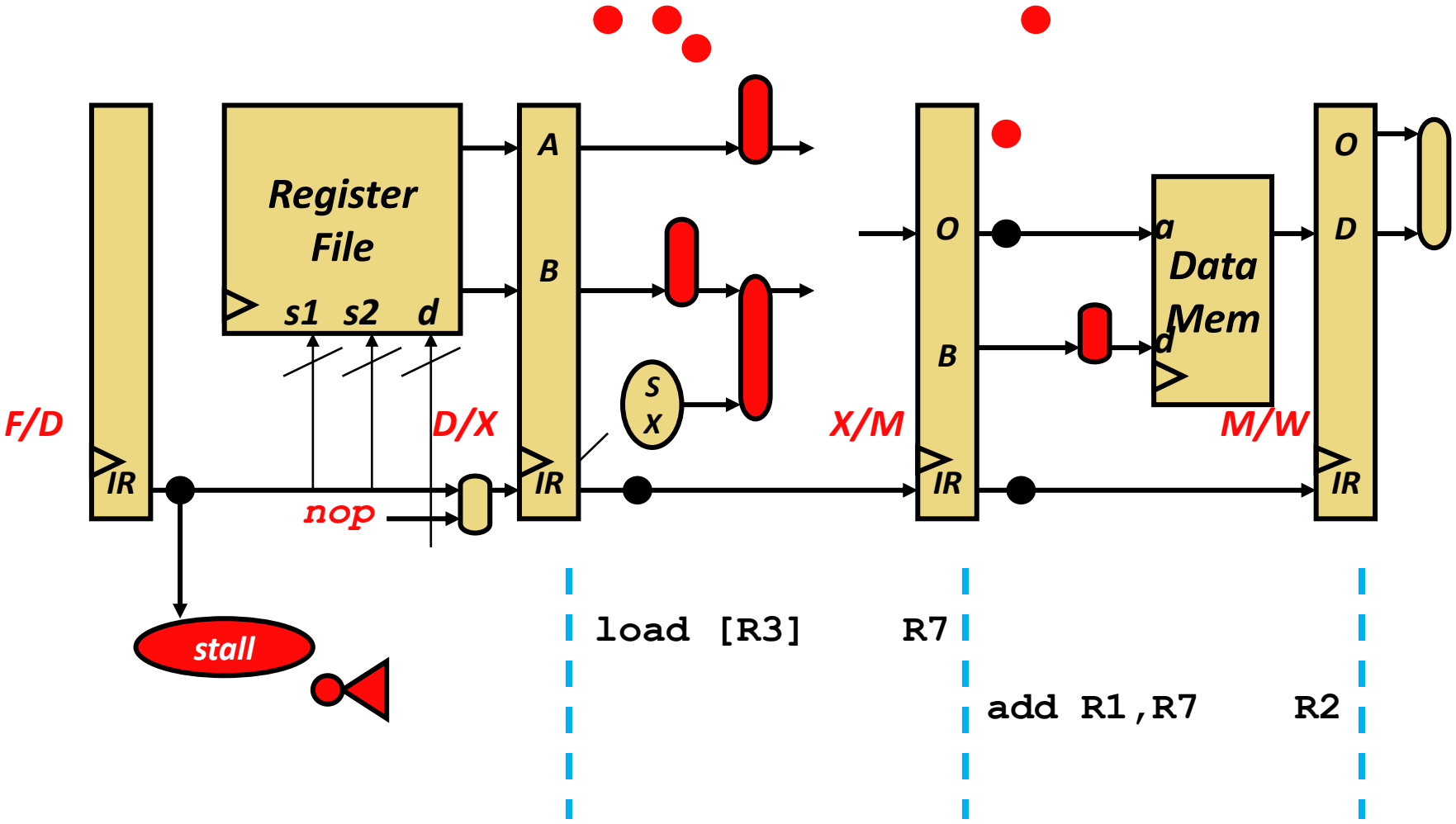


Image: Penn CIS501

分支 Branches

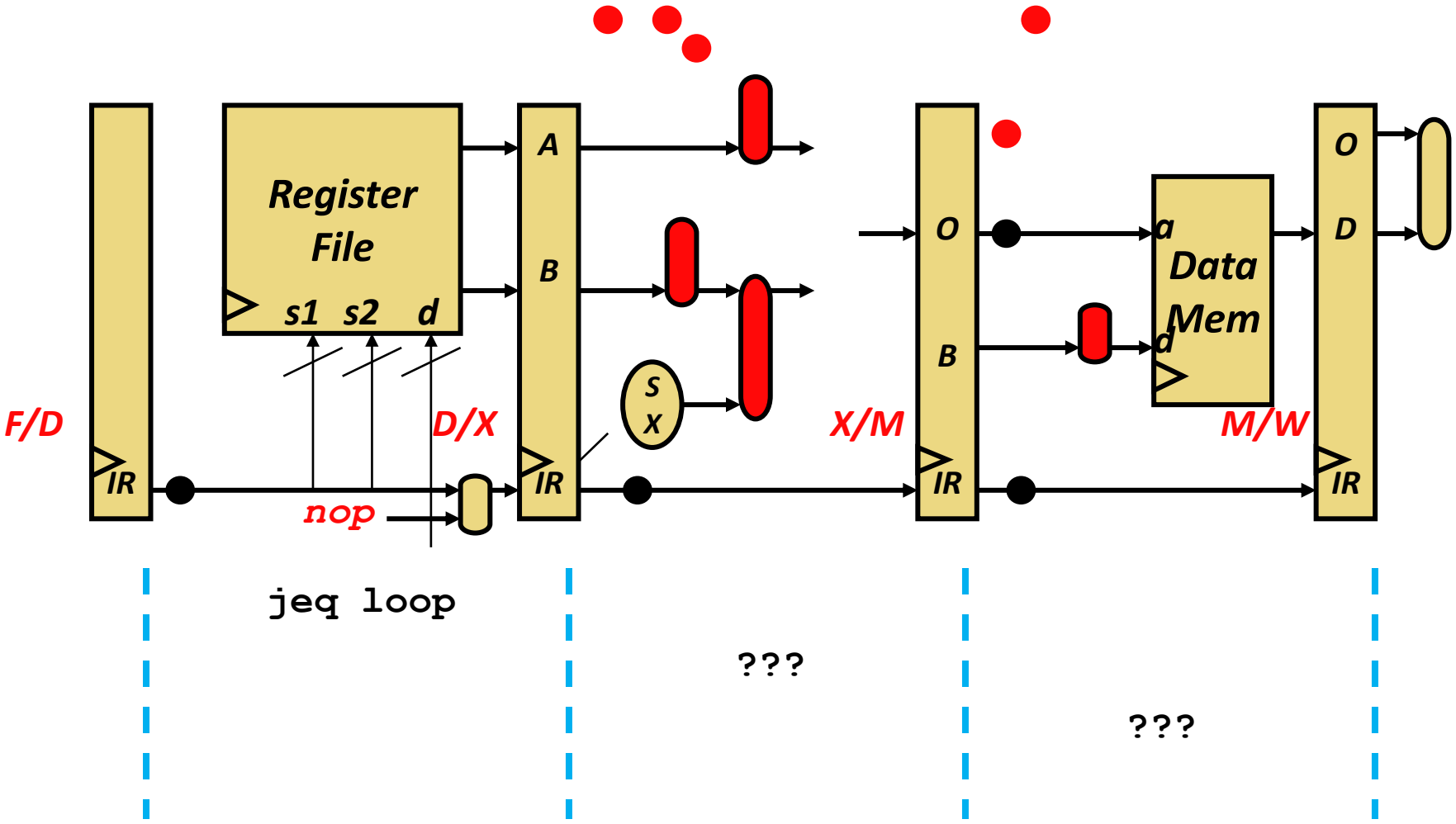


Image: Penn CIS501

分支预测 Branch Prediction

- 猜测下一条指令
- Guess what instruction comes next
- 基于过去的分支记录
- Based off branch history
- 举例：基于全局记录的两层预测
- 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

- 让很多人绞尽脑汁的工作，培养了无数博士

分支预测 Branch Prediction

+ 现代预测器准确度大于90%

Modern predictors > 90% accuracy

- 提升性能及能量效率
- Raise performance *and* energy efficiency (why?)

– Area increase

面积增加

– Potential fetch stage latency increase

可能会增加延迟

分支断定 Another option: Predication

- 用条件语句替换分支
- Replace branches with conditional instructions

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

+ 不使用分支预测器

Avoids branch predictor

  - Avoids area penalty, misprediction penalty
  - 减少面积，减少错误预测

– Avoids branch predictor

  - Introduces unnecessary `nop` if predictable branch

- 在GPU中使用分支断定
- GPUs also use predication

提升 IPC

- 常规 IPC (instructions/cycle) 受限于 1 instruction / clock
- 超标量 Superscalar – 增加流水线宽度

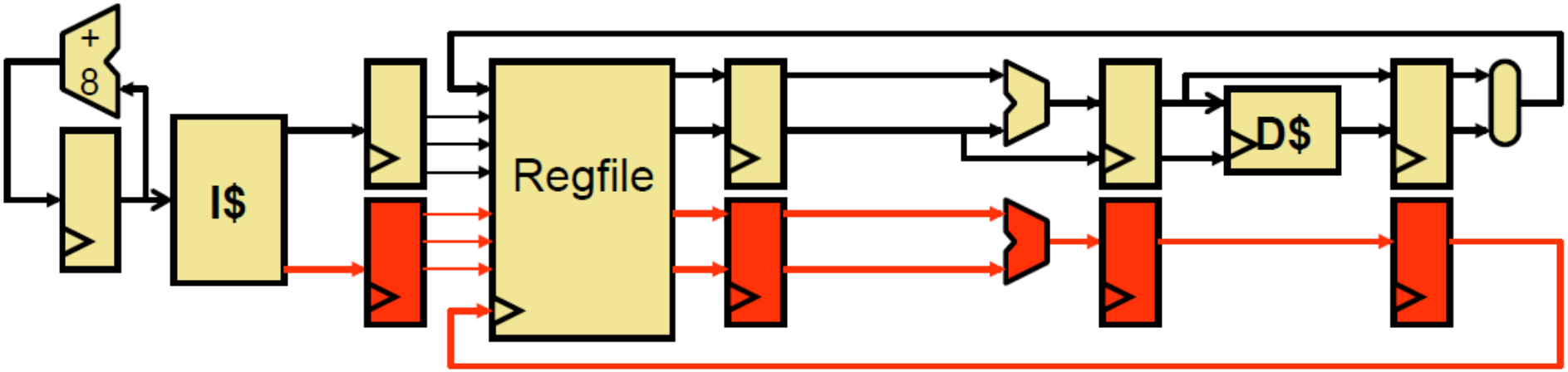


Image: Penn CIS371

超标量 Superscalar

+ 峰值IPC 为 N (N路超标量)

Peak IPC now at N (for N-way superscalar)

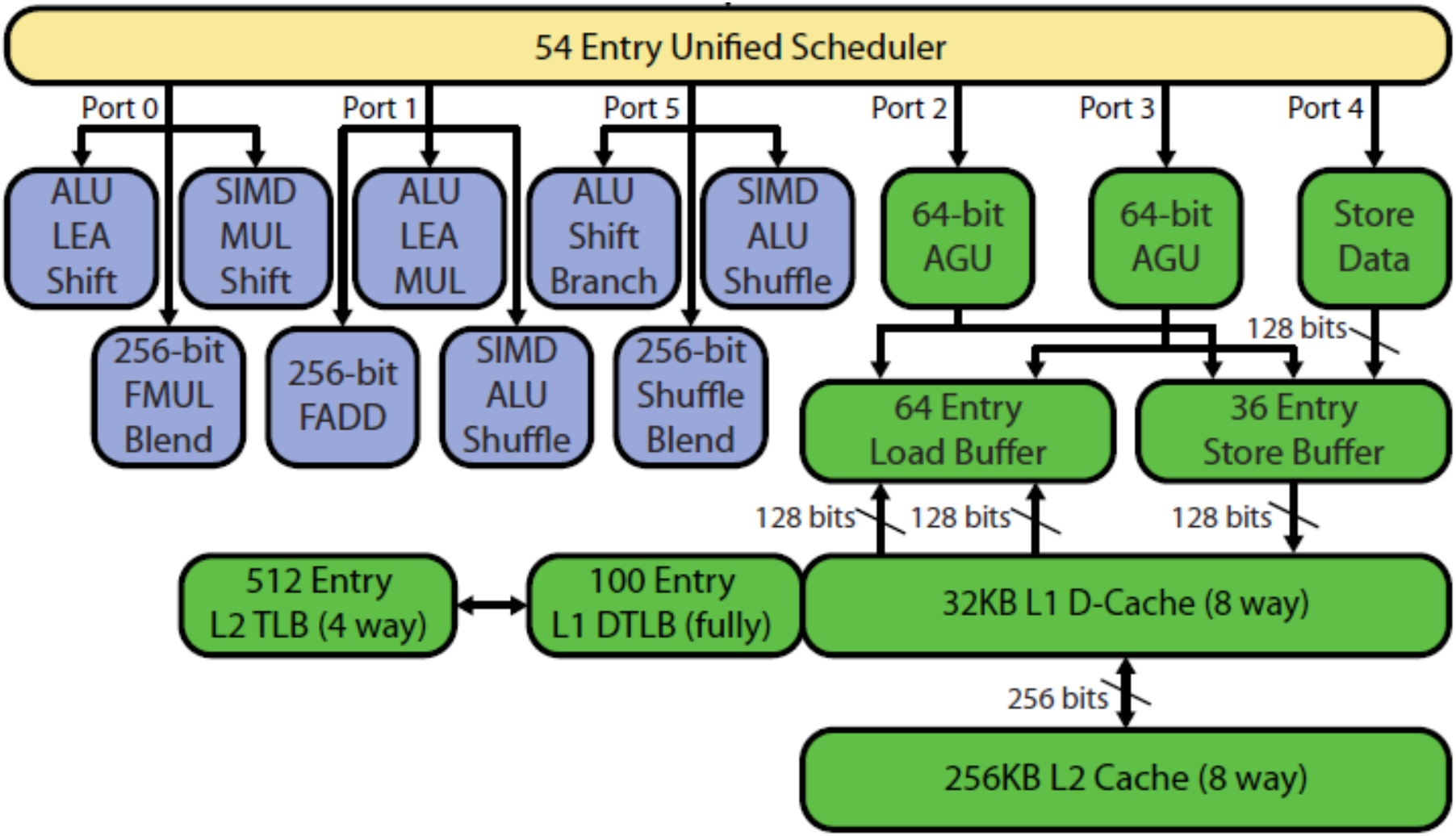
- 分支和调度会产生开销
- 需要一些技巧来逼近峰值

– 增加了面积

Area increase

- N倍资源使用
- 旁路网络 N^2
- 需要更多的寄存器和存储器带宽

Sandy Bridge 超标量



指令调度 Scheduling

- 考虑如下指令

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

- xor and add 互相依赖(Read-After-Write, RAW)
- sub and addi 互相依赖 (RAW)
- xor and sub 不依赖 (Write-After-Write, WAW)

寄存器重命名 Register Renaming

- 替换寄存器

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

- xor and sub 可以并行执行

乱序执行Out-of-Order(OoO) Execution

- 重排指令，获得最大的吞吐率
- Fetch → Decode → Rename → Dispatch → Issue → Register-Read → Execute → Memory → Writeback → Commit

- 重排缓冲区Reorder Buffer (ROB)
  - 记录所有执行中的指令状态

- 发射队列/调度器Issue Queue/Scheduler
  - 选择下一条执行的指令

OoO Sandy Bridge

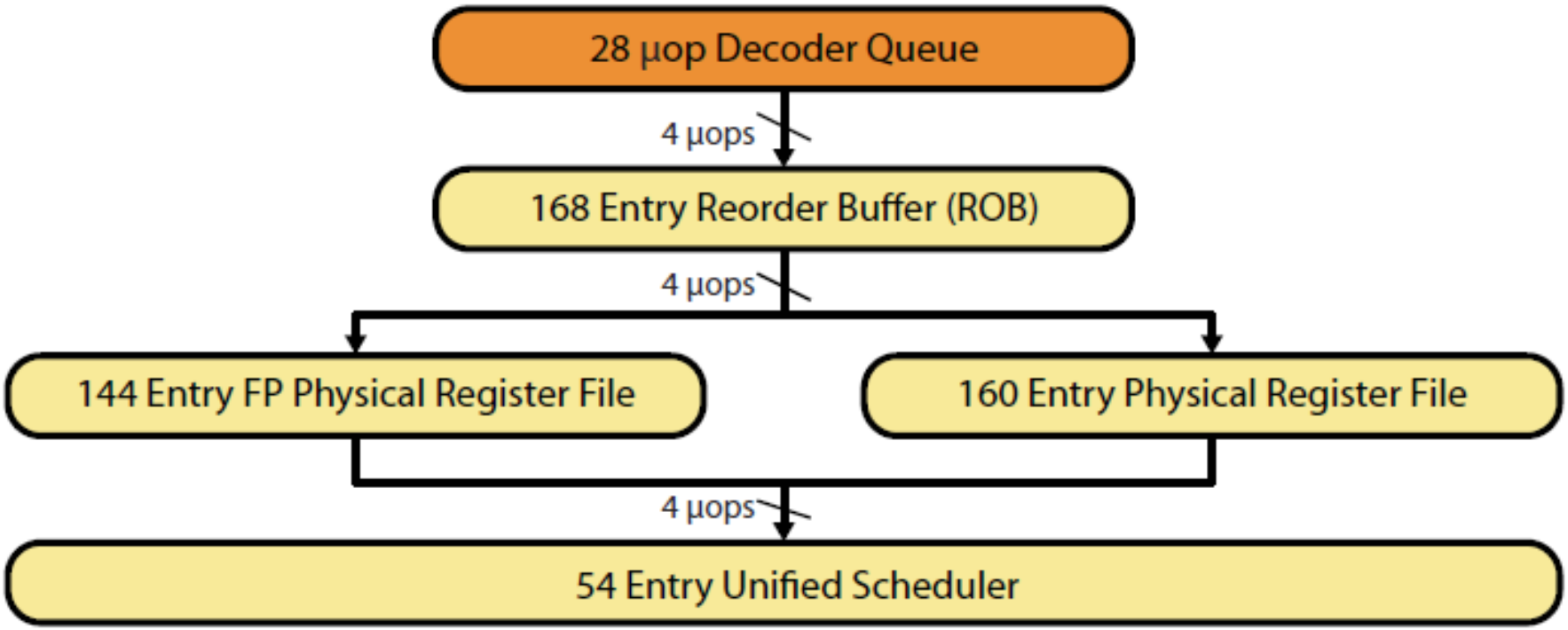


Image © David Kanter, RWT

乱序执行

+ IPC 接近理想状态
– 面积增加
– 功耗增加

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

存储器架构/层次 Memory Hierarchy

- 存储器越大越慢
- 粗略的估计

	延迟	带宽	大小
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

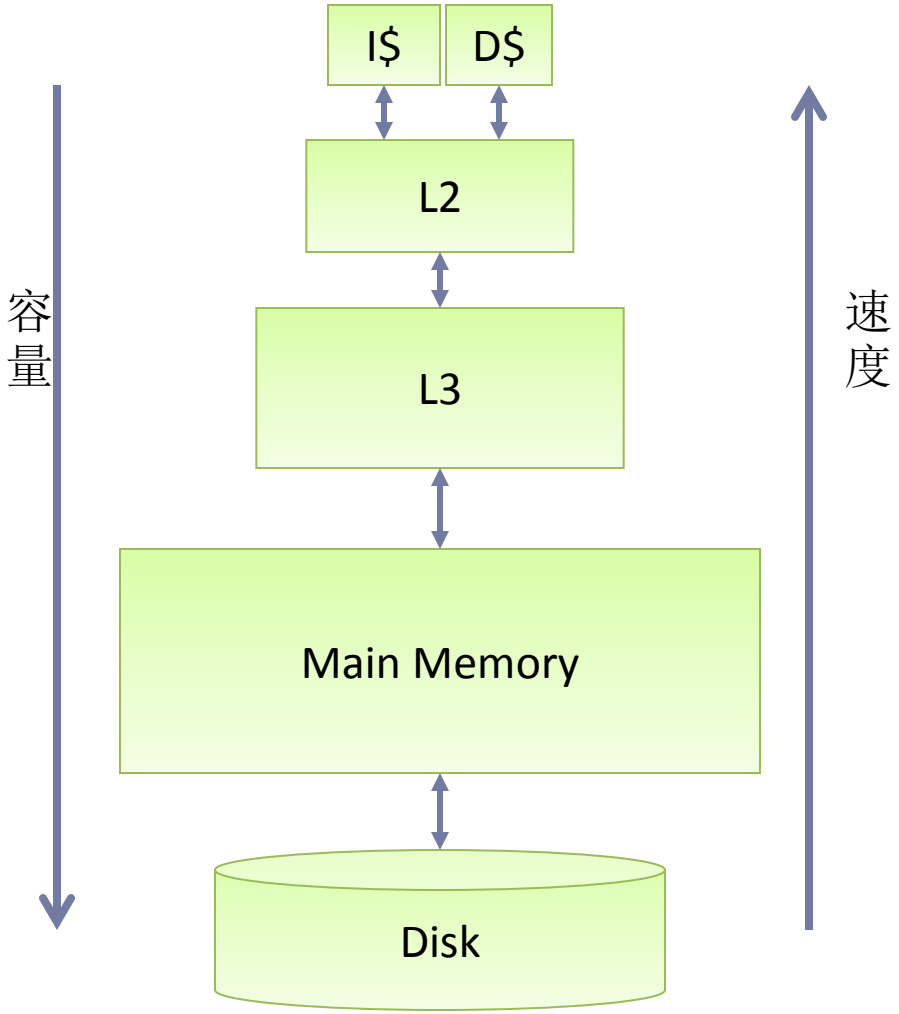
SRAM & DRAM latency, and DRAM bandwidth for Sandy Bridge from Lostcircuits
Flash and HDD latencies from AnandTech
Flash and HDD bandwidth from AnandTech Bench
SRAM bandwidth guesstimated.

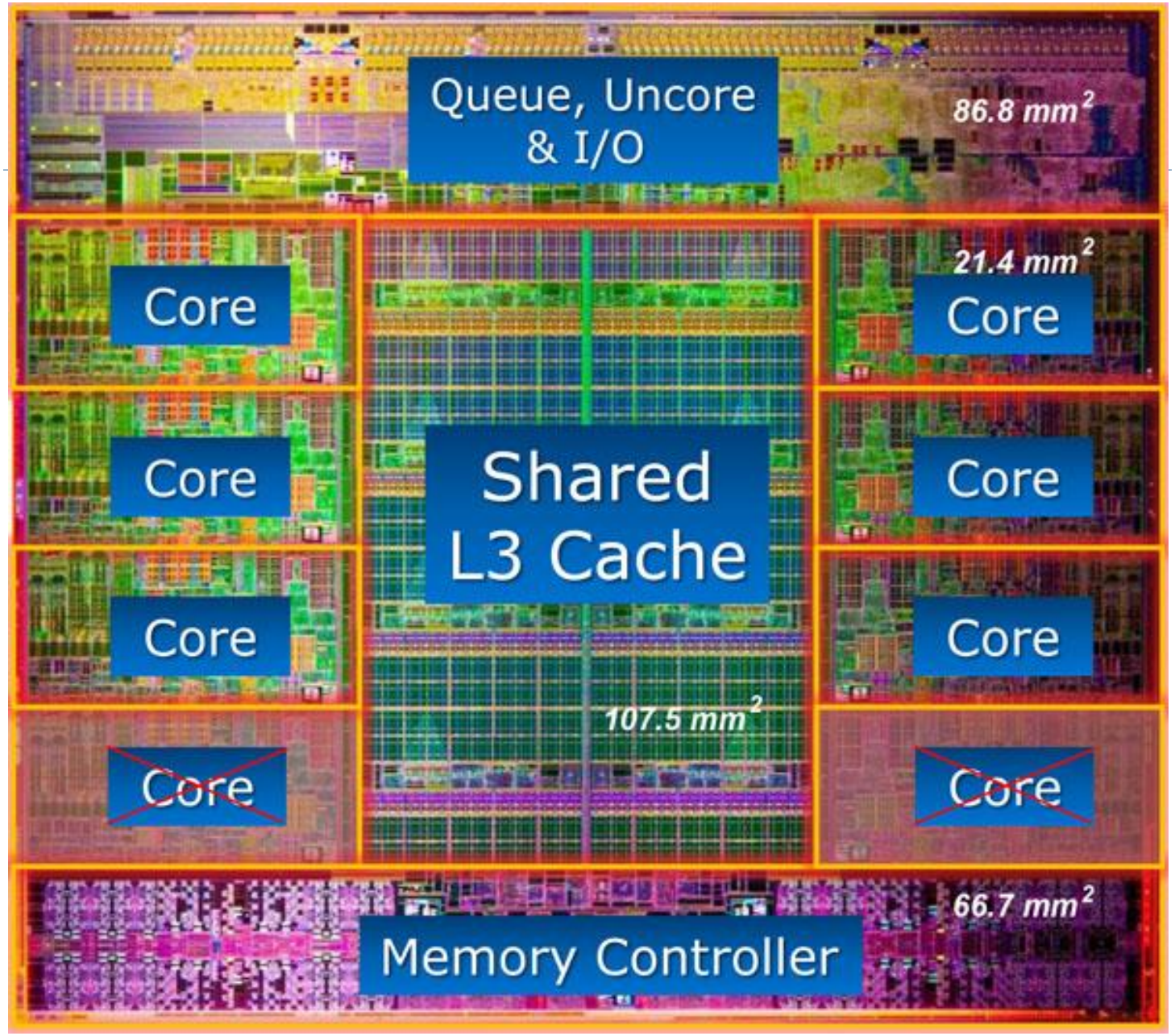
缓存Caching

- 将数据放在尽可能接近的位置
- 利用：
  - 时间临近性
    - 刚刚使用过的数据很可能会被再使用
  - 空间临近性
    - 倾向于使用周围的临近的数据

缓存层次Cache Hierarchy

- 硬件管理
  - L1 Instruction/Data caches
  - L2 unified cache
  - L3 unified cache
- 软件管理
  - Main memory
  - Disk





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

Source: www.lostcircuits.com

存储器的另外设计考虑

- 分区Banking
  - 避免多端口
- 一致性Coherency
- 控制器Memory Controller
  - 多个通道，增加带宽

CPU内部的并行性

- 指令级并行 Instruction-Level (ILP) extraction
  - 超标量Superscalar
  - 乱序执行Out-of-order
- 数据级并行Data-Level Parallelism (DLP)
  - 矢量计算Vectors
- 线程级并行Thread-Level Parallelism (TLP)
  - 同步多线程Simultaneous Multithreading (SMT)
  - 多核 Multicore

向量运算 Vectors Motivation

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

数据级并行 Data-level Parallelism

- 单指令多数据 Single Instruction Multiple Data (SIMD)
  - 执行单元 (ALU) 很宽
  - 寄存器也很宽

```
for (int i = 0; i < N; i+= 4) {
    // in parallel 并行同时计算
    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];
}
```

x86的向量运算

- SSE2
  - 4宽度浮点或整数指令
  - Intel Pentium 4 onwards
  - AMD Athlon 64 onwards
- AVX
  - 8宽度浮点或整数指令
  - Intel Sandy Bridge
  - AMD Bulldozer

线程级并行 Thread-Level Parallelism

- 线程组成
  - 指令流Instruction streams
  - 私有的寄存器，程序计数器，栈Private PC, registers, stack
  - 共享的全局变量，堆Shared globals, heap
- 程序员可以创建和销毁
- 程序员或者OS都可以调度

多核Multicore

- 将流水线完整复制
- Sandy Bridge-E: 6 cores

+完整的核，除了最后一级缓存外，不共享其他资源

+继续保持摩尔定律

– 多核程序和利用率问题

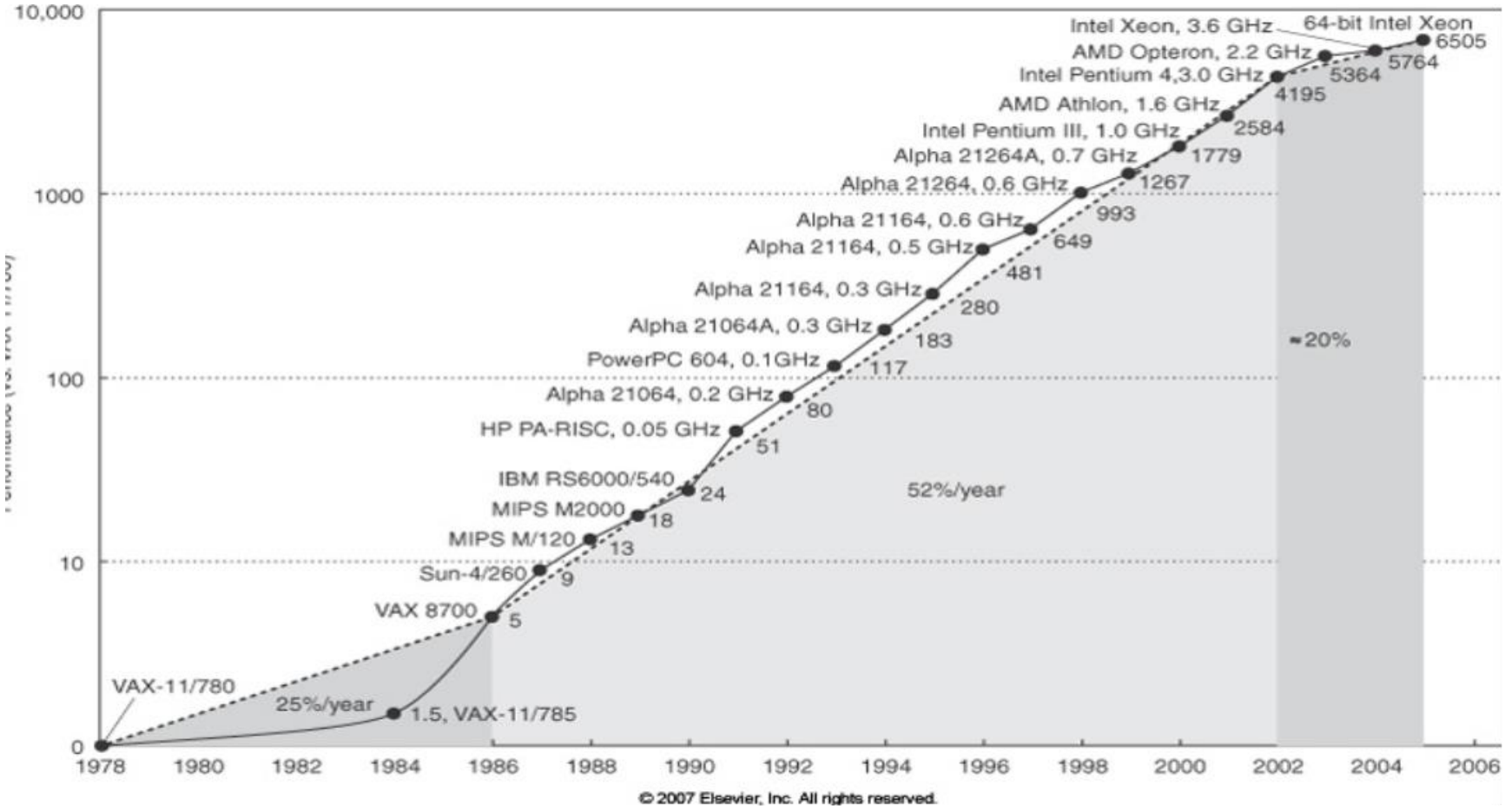
锁、一致性和同一性Locks, Coherence, and Consistency

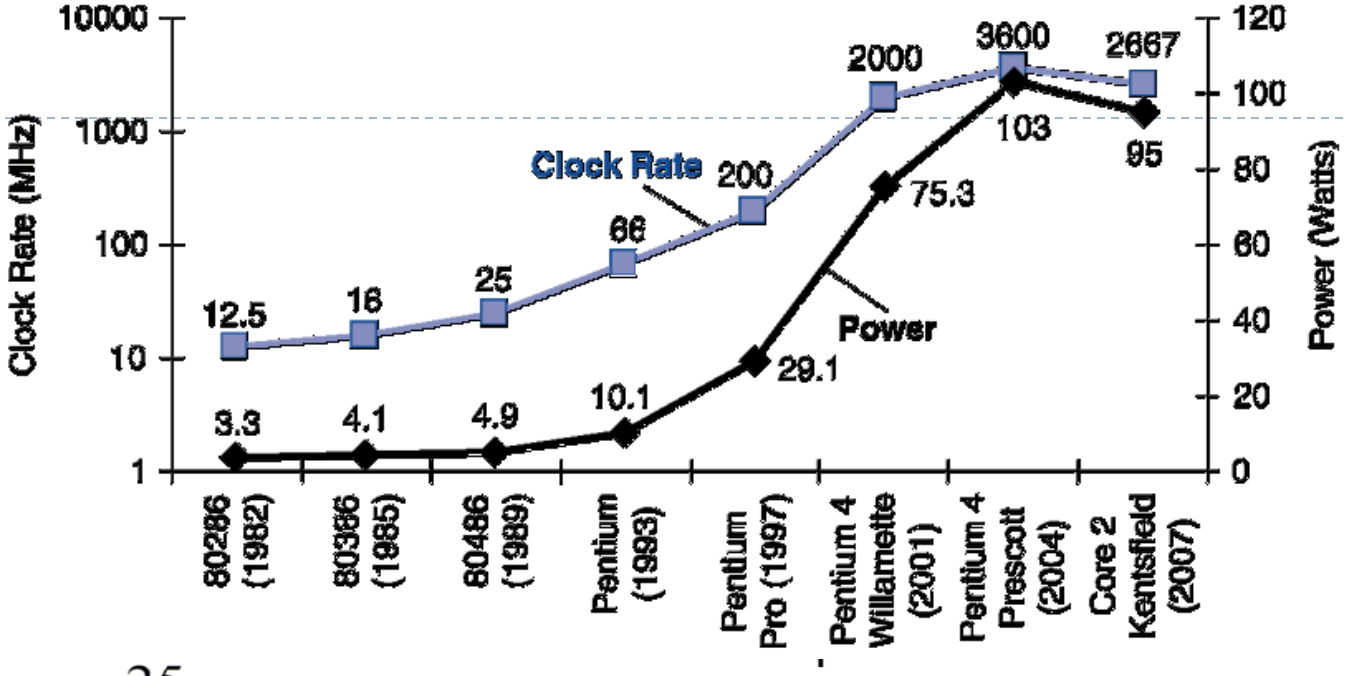
- **问题**: 多线程读写同一块数据
- 解决方法: 加锁

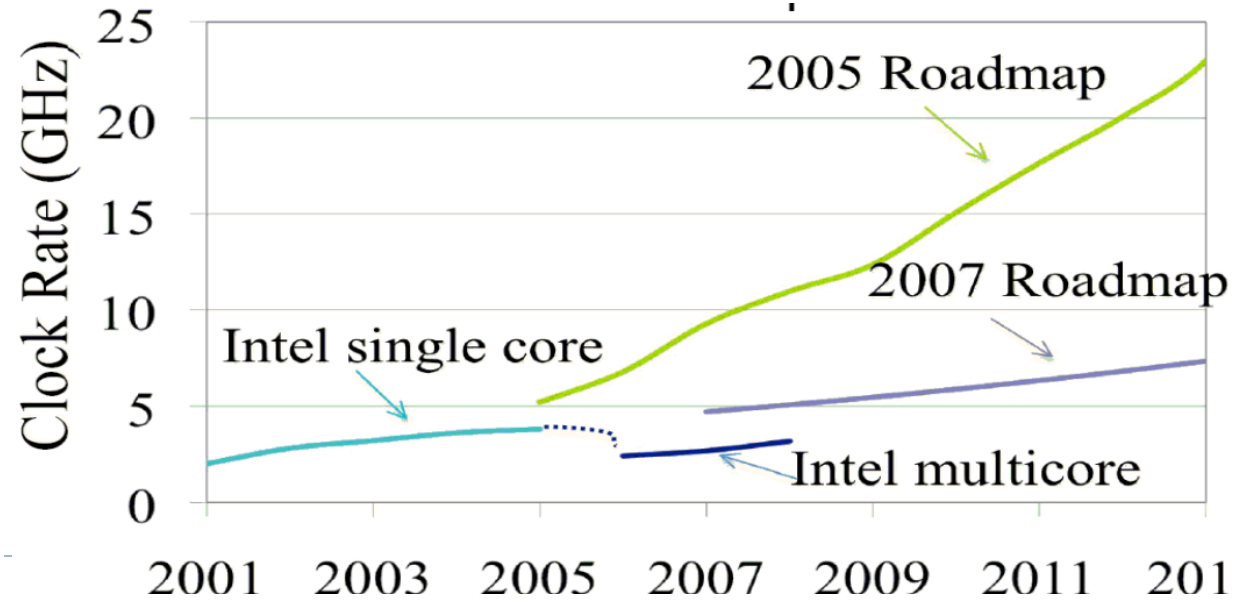
- 问题: 谁的数据是正确的?
- 解决方法: 缓存一致性协议 Coherence

- 问题: 什么样的数据是正确的 Consistency
- 解决方法: 存储器同一性模型

现实的困境：能量墙: Power Wall

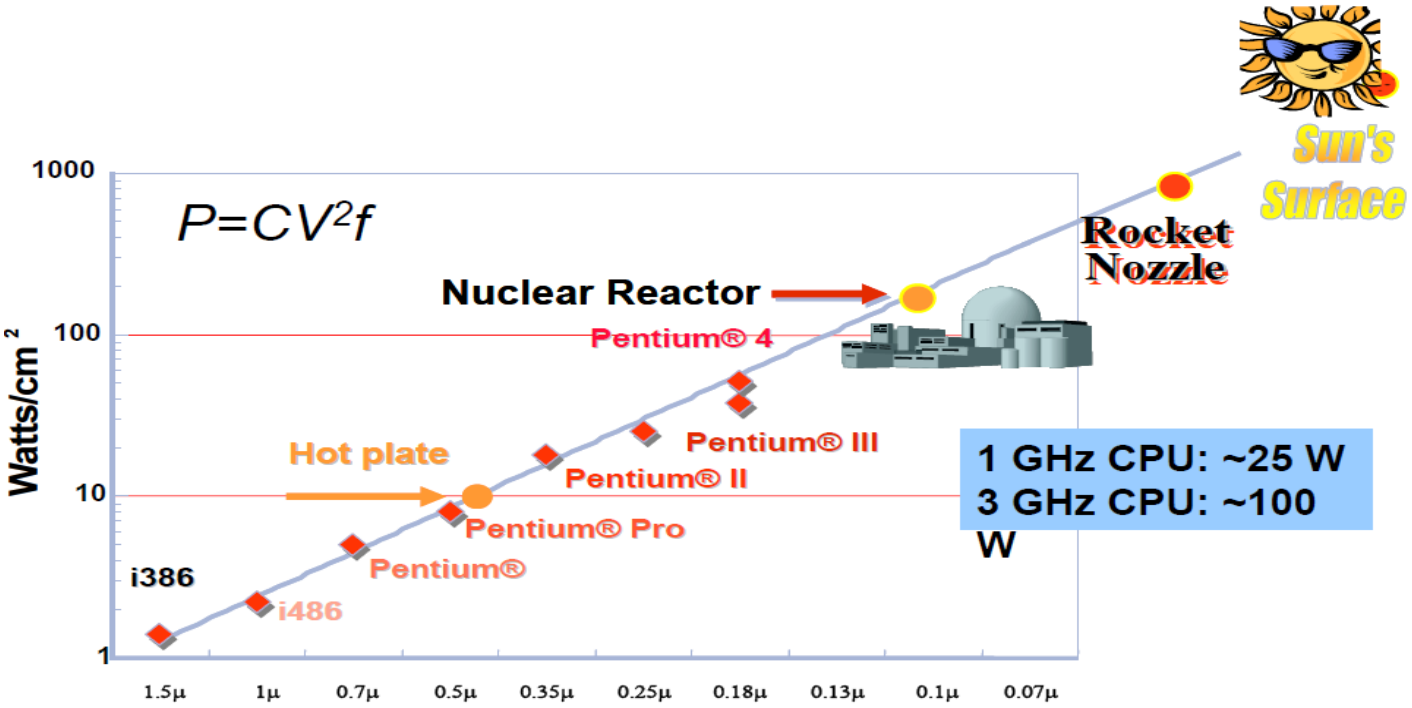



Clock Rate (MHz)
Power (Watts)
10000
1000
100
10
1
120
100
80
60
40
20
0
Clock Rate
Power
12.5
16
25
66
200
2000
3600
2667
3.3
4.1
4.9
10.1
29.1
75.3
103
95
80286 (1982)
80386 (1985)
80486 (1989)
Pentium (1993)
Pentium Pro (1997)
Pentium 4 Willamette (2001)
Pentium 4 Prescott (2004)
Core 2 Kentsfield (2007)


Clock Rate (GHz)
25
20
15
10
5
0
2005 Roadmap
2007 Roadmap
Intel single core
Intel multicore
2001
2003
2005
2007
2009
2011
2013

新时代的计算技术: 并行计算

- 常规传统单核处理器遇到物理约束
  - 时钟频率 (perf/clock) 无法保持线型增长



免费的午餐要消失了

新摩尔定律

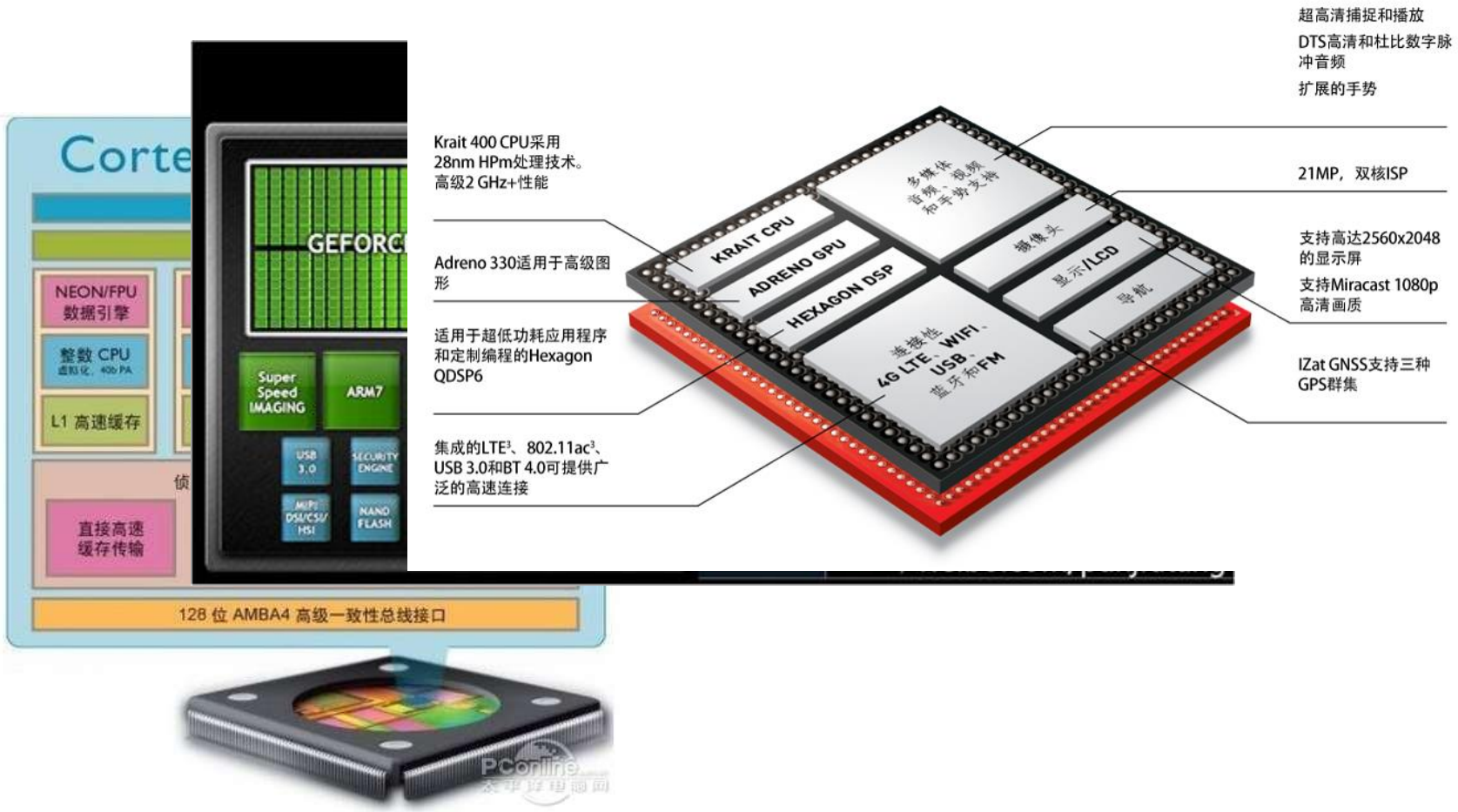
- 处理器越来越胖，核越来越多
- 单核的性能不会大幅度提升

CPU Type	CORES/Freq.	CPU Type	CORES/Freq.
Intel i7 3960X	6/3.3GHz	AMD FX-8150	8/3.6GHz
Intel Xeon E5-2687W	8/3.1GHz	AMD Opteron 6282SE	16/2.6GHz

- 另外一堵墙：存储器墙

处理器的存储器带宽无法满足处理能力的提升

一些新的处理器


超高清捕捉和播放
DTS高清和杜比数字脉冲音频
扩展的手势
Krait 400 CPU采用28nm HPm处理技术。高级2 GHz+性能
21MP，双核ISP
Adreno 330适用于高级图形
支持高达2560x2048的显示屏
支持Miracast 1080p高清画质
适用于超低功耗应用程序和定制编程的Hexagon QDSP6
IZat GNSS支持三种GPS群集
集成的LTE³、802.11ac³、USB 3.0和BT 4.0可提供广泛的高速连接
KRAIT CPU
ADRENO GPU
HEXAGON DSP
多媒体、视频、音频、和手势支持
摄像头
显示/LCD
导航
连接性、4G LTE、WiFi、USB、蓝牙和FM
NEON/FPU 数据引擎
整数 CPU
L1 高速缓存
直接高速缓存传输
128 位 AMBA4 高级一致性总线接口
Super Speed IMAGING
ARM7
USB 3.0
SECURITY ENGINE
NAND FLASH

结论

- CPU 为串行程序优化
  - Pipelines, branch prediction, superscalar, OoO
  - Reduce execution time with high clock speeds and high utilization
- 缓慢的内存带宽（存储器带宽）将会是大问题
- 并行处理是方向
  - Sandy Bridge-E 6-12 活动线程

- 如果有成千上万的活动线程并发执行呢?