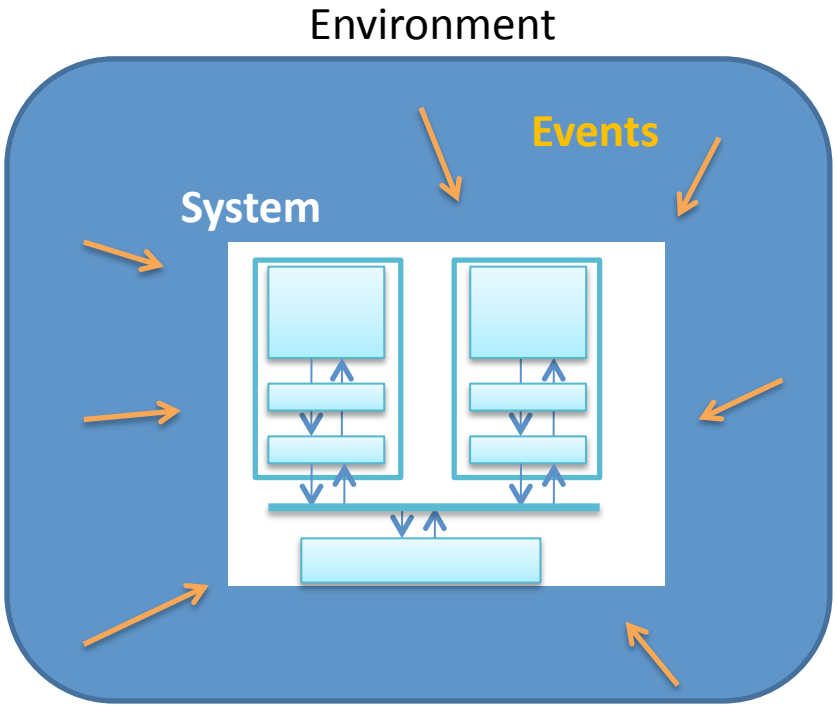
INTRODUCTION

Parallelism and Concurrrency: System and Environment

- *Parallelism*: exploit system resources to speed up computation

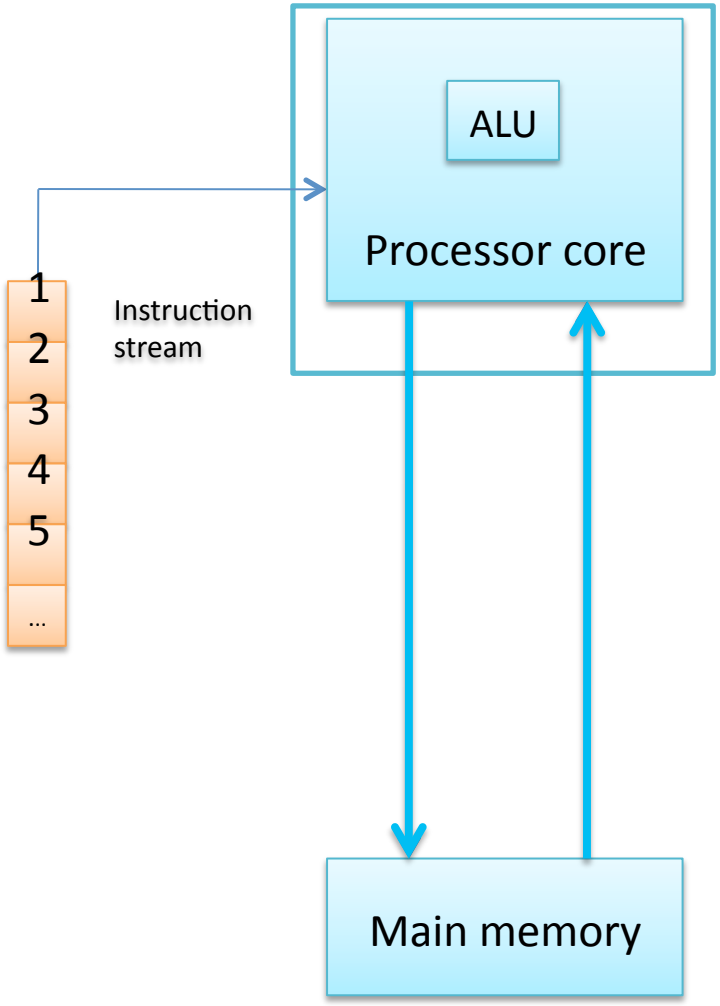
- *Concurrency*: respond quickly/ properly to events
  - from the environment
  - from other parts of system



Concurrency and Parallelism

- Why should you care?

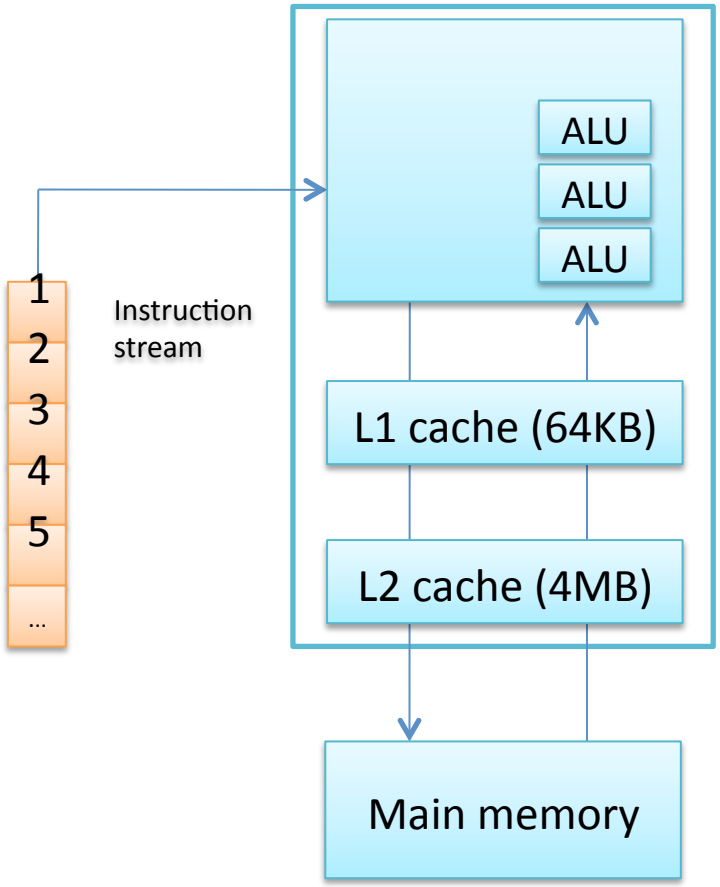
A simple microprocessor model ~ 1985



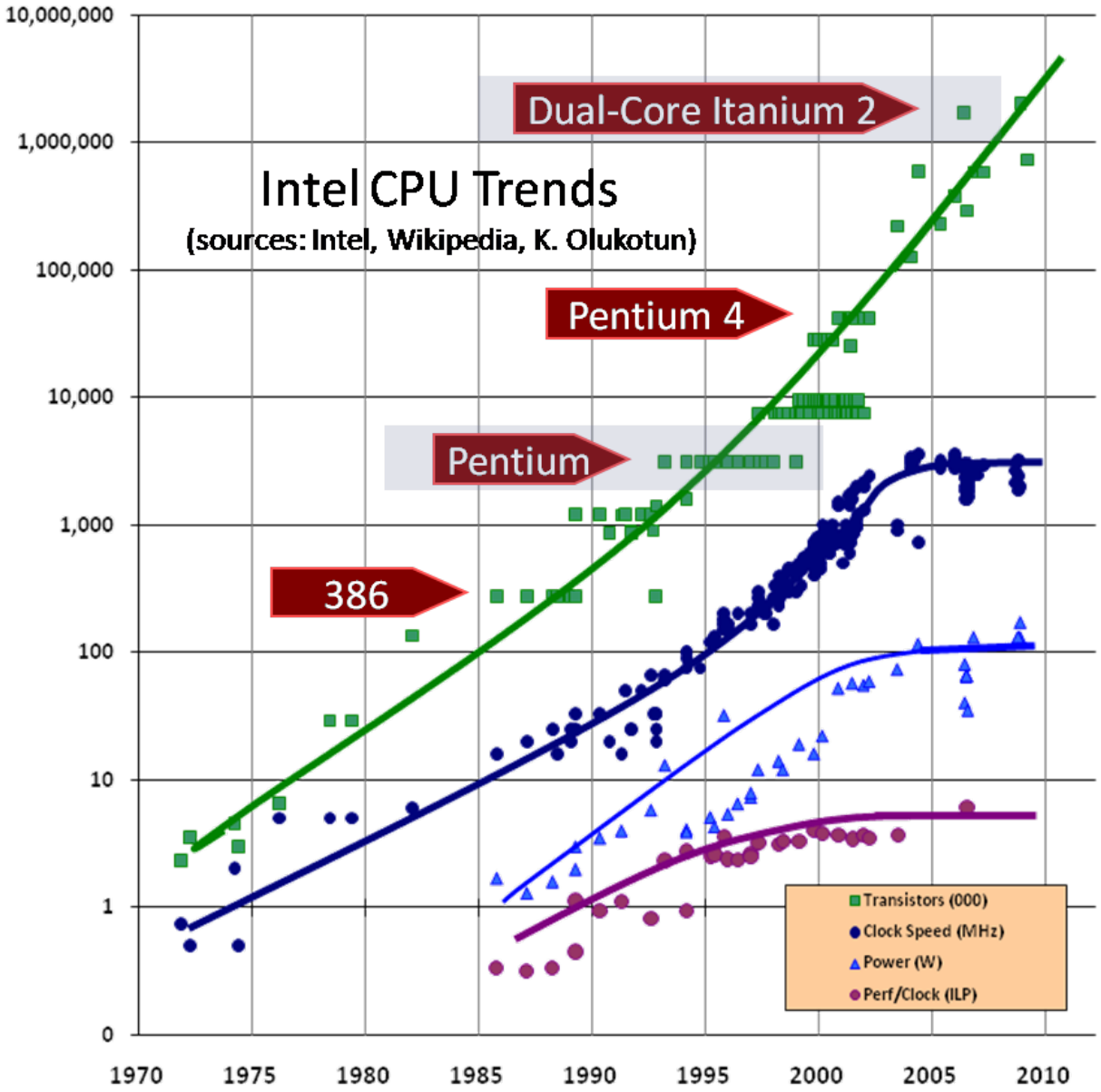
- Single h/w thread
- Instructions execute one after the other
- Memory access time ~ clock cycle time

ALU: arithmetic logic unit

FastFwd Two Decades (circa 2005): Power Hungry Superscalar with Caches



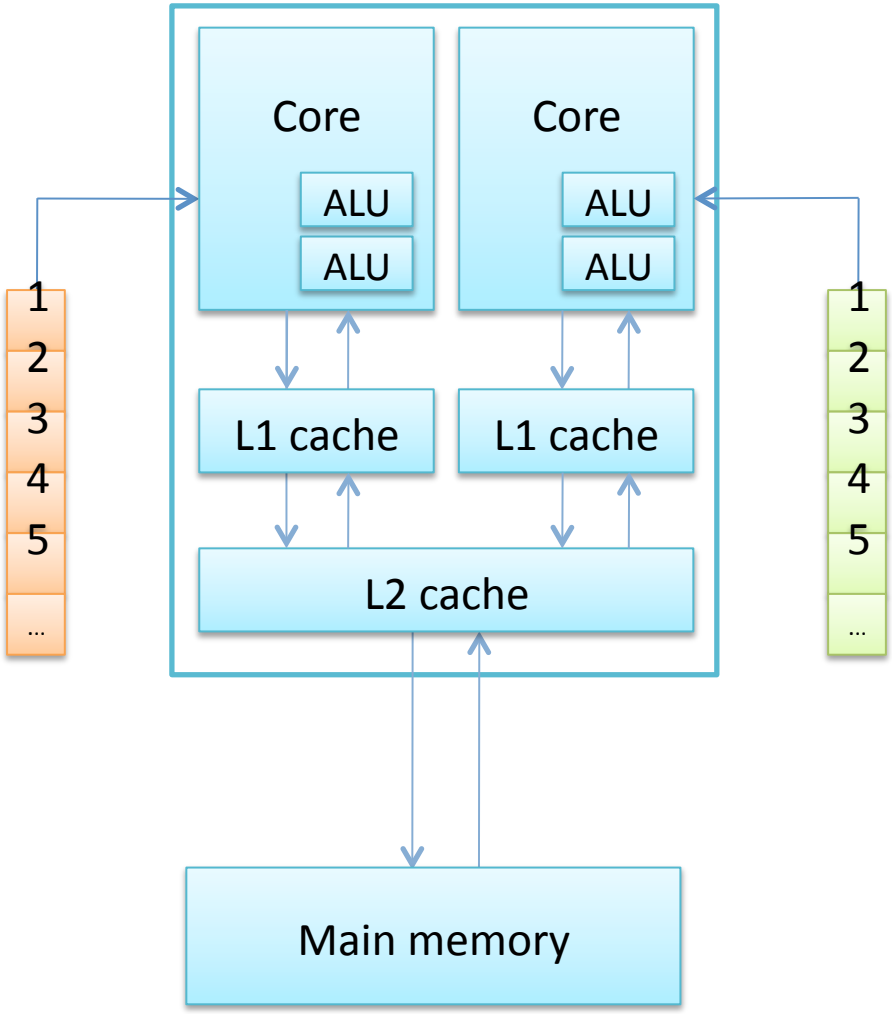
- Caching
- Pipelining
- Out-of-order execution
- Speculation
- …


Intel CPU Trends
(sources: Intel, Wikipedia, K. Olukotun)
Dual-Core Itanium 2
Pentium 4
Pentium
386
Transistors (000)
Clock Speed (MHz)
Power (W)
Perf/Clock (ILP)
10,000,000
1,000,000
100,000
10,000
1,000
100
10
1
0
1970
1975
1980
1985
1990
1995
2000
2005
2010

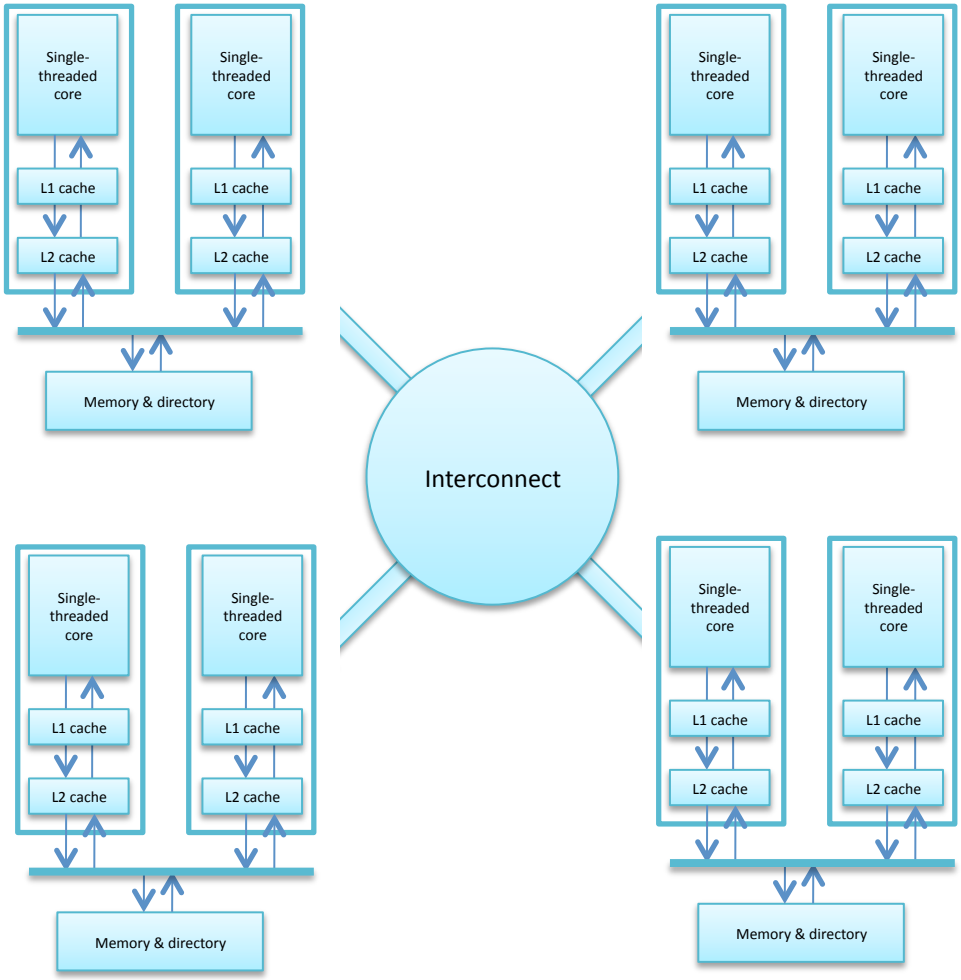
Power wall + Memory wall + ILP wall = **BRICK WALL**

- *Power wall*
  - we can't clock processors faster
- *Memory wall*
  - many workload's performance is dominated by memory access times
- *Instruction-level Parallelism (ILP) wall*
  - we can't find extra work to keep functional units busy while waiting for memory accesses

Multi-core h/w – common L2



NUMA multiprocessor



Technology Trends

- Increasing parallelism within a computer
  - Multi-core CPU
  - Graphical processing unit (GPU)
- Importance of power-efficient computing
  - Mobile phones, tablets
- Increasing disk capacity
  - We are awash in interesting *data*
  - Data-intensive problems require *parallel processing*

Technology Trends (2)

- Increasing *networks and network bandwidth*
  - Wireless, wimax, 3G, …
  - Collection/delivery of massive datasets, plus
  - Real-time responsiveness to asynchronous events
- Increasing *number and variety of computers*
  - Smaller and smaller, and cheaper to build
  - Generating streams of asynchronous events

Application Areas

- Entertainment/games
- Finance
- Science
- Modeling of real-world
- Health care
- Telecommunication
- Data processing
- …

Writing Concurrent and Parallel Code is HARD

- In general: difficult to do multiple things at the same time
- Performance considerations
- Correctness considerations

Performance Considerations

- Speedup
- Responsiveness
- Locality
- Load balance
- Coordination and synchronization

Correctness Concerns

- All those we have for sequential code
  - Assertions, invariants, contracts,
  - buffer overflows, null reference,
  - …
- Plus those related to parallelism/concurrency
  - Data races, deadlocks, livelocks, …
  - Memory coherence/consistency