Parallelism Available

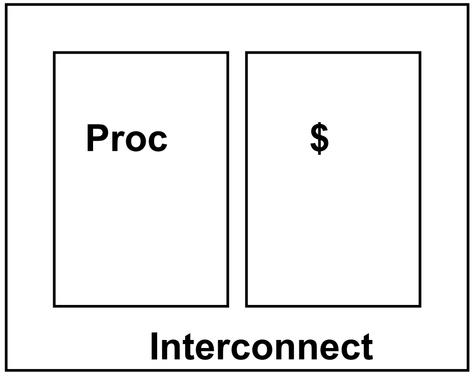
- Bits
- Operations
  - Add, subtract, multiply, …
  - Instruction-level (ILP)
    - How many processor instructions in parallel?
- Thread-level
  - How many threads at once
- Process-level – as above, less used
- Task-level
- Coarse-level: complete programs (or so)

Module 3: Parallel Comp. Drivers

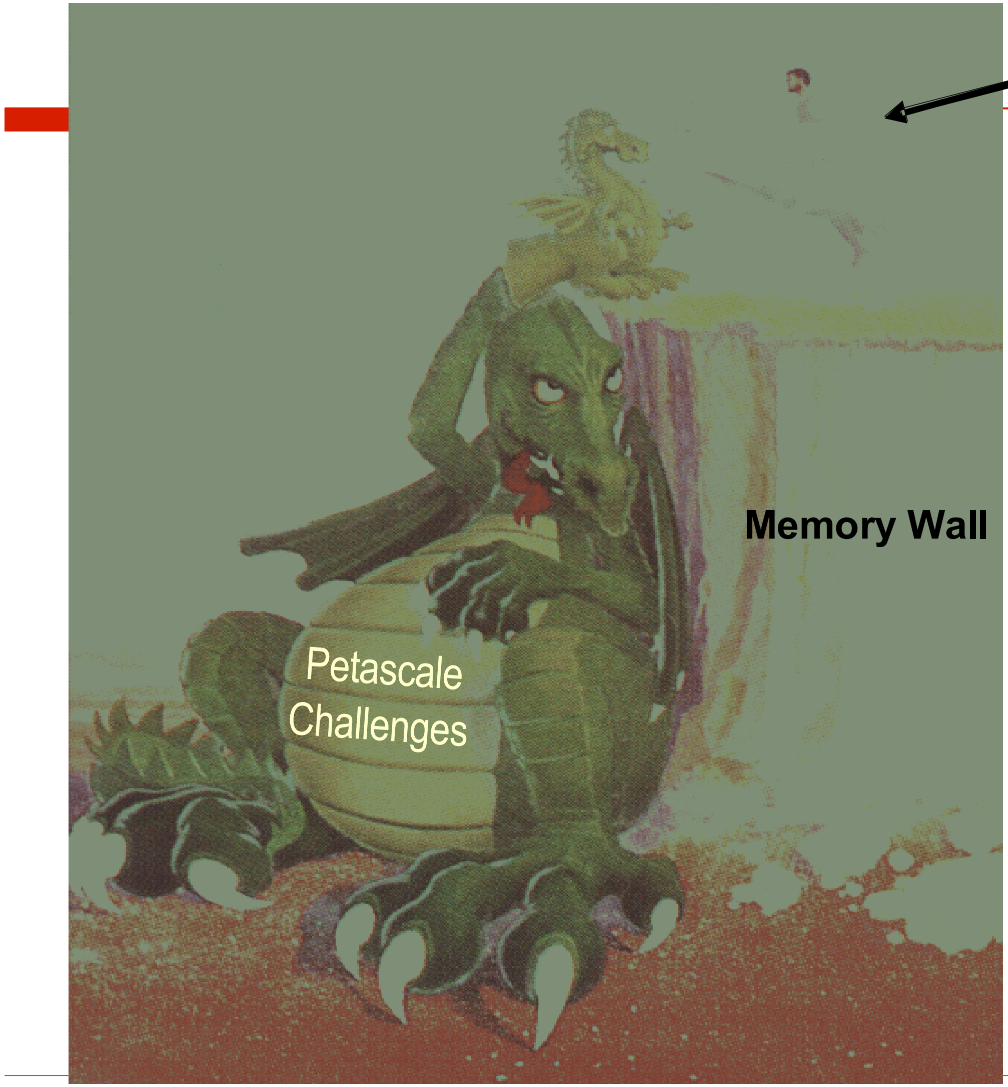
- Technology underpinning
  - Area vs. delay
  - Locality challenge, memory wall
- Sequential architecture trends
- Stages in parallelism exploitation
- ILP limits
- Multithreading
- Parallel computing taxonomies

Technology: A Closer (Rough) Look

- Basic advance is (was?) *decreasing feature size* (λ)
  - Circuits become either faster or lower in power
- Die size is growing too
  - Clock rate improves roughly proportional to λ
  - Number of transistors improves like λ^2 (or faster)
- Performance > 100x per decade
  - clock rate < 10x, rest is transistor count
- *How to use more transistors?*
  - Parallelism in processing
    - multiple operations per cycle reduces CPI
  - Locality in data access
    - avoids latency and reduces CPI
    - also improves processor utilization
  - Both need resources, so tradeoff
- *Fundamental issue- resource distribution, as in uniprocessors*



Challenges: Performance at Scale



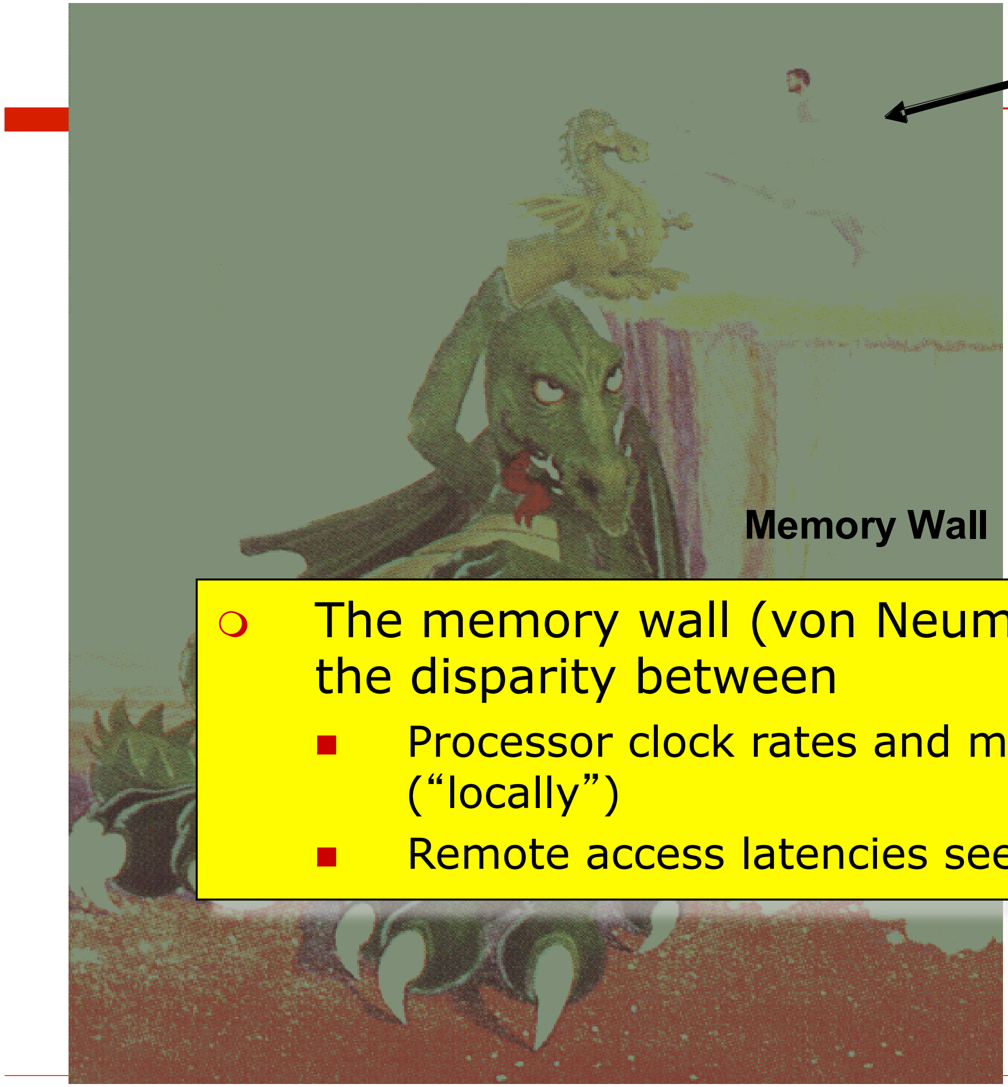
Advanced simulation and modeling apps

Conquering Terascale problems of today

Beware being eaten alive by the petascale problems of tomorrow.

Drawing by Thomas Zacharia (ORNL)

Performance at Scale



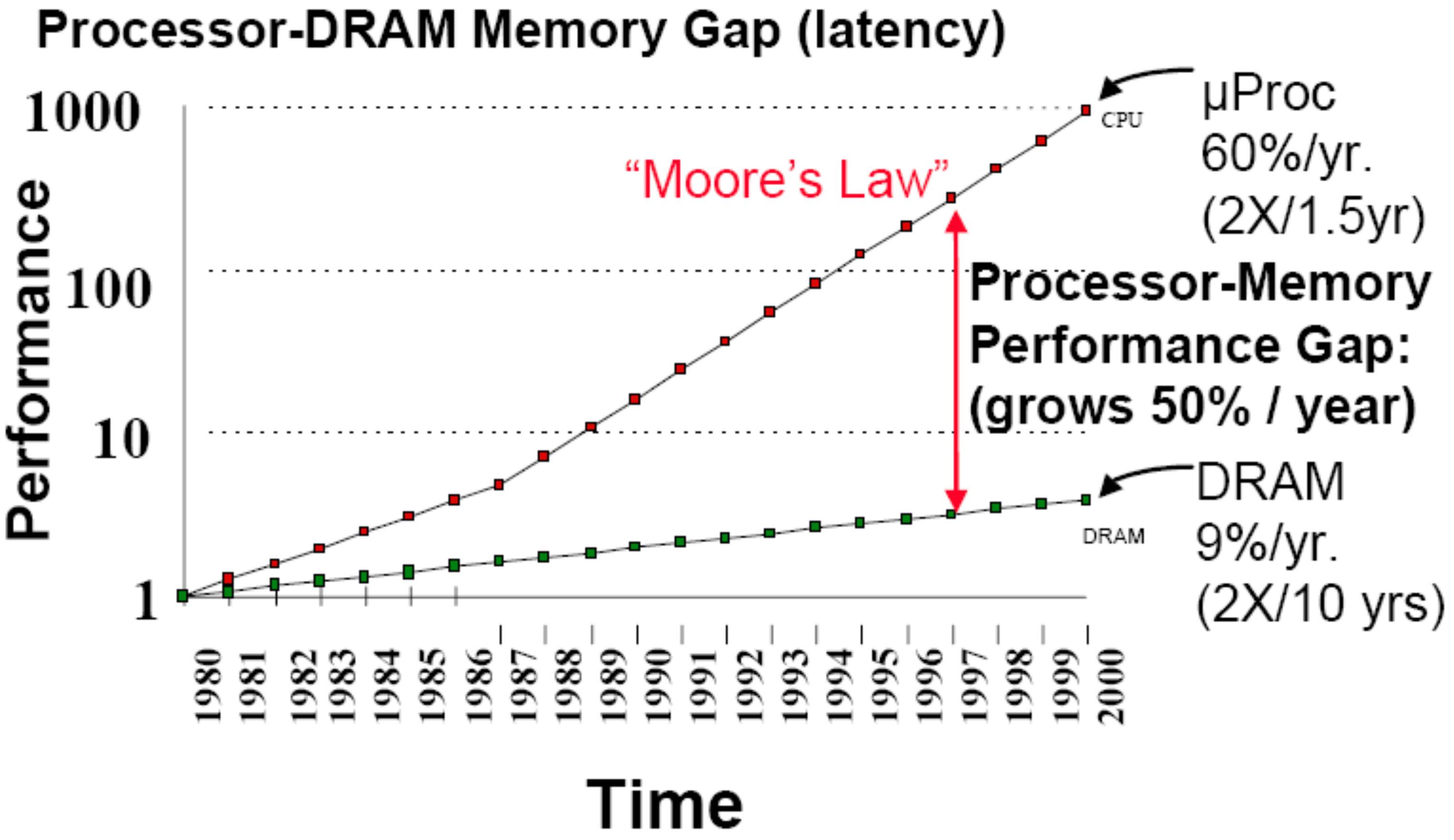
Advanced simulation and modeling apps

Conquering Terascale problems of today

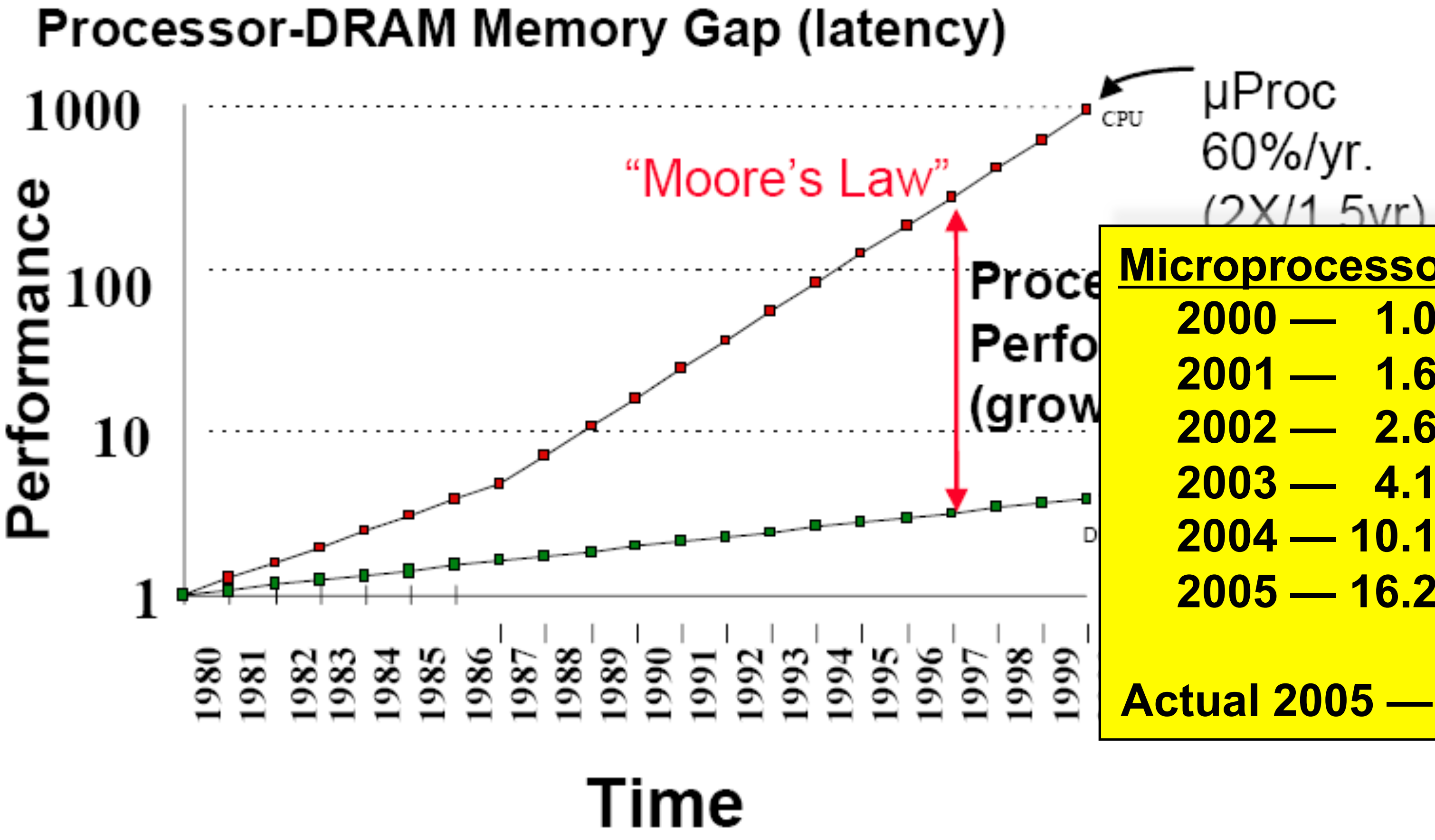
- The memory wall (von Neumann bottleneck) — the disparity between
  - Processor clock rates and memory cycle times (“locally”)
  - Remote access latencies seen “system wide”

Drawing by Thomas Zacharia (ORNL)

The Memory Latency Wall



The Memory Latency Wall

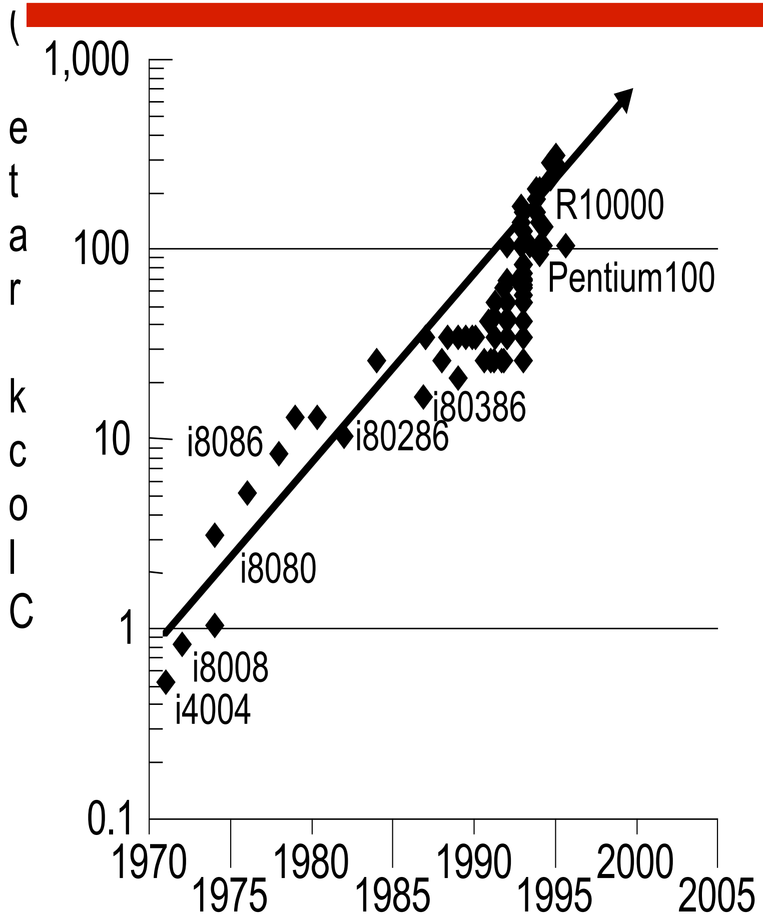


Microprocessor Clock

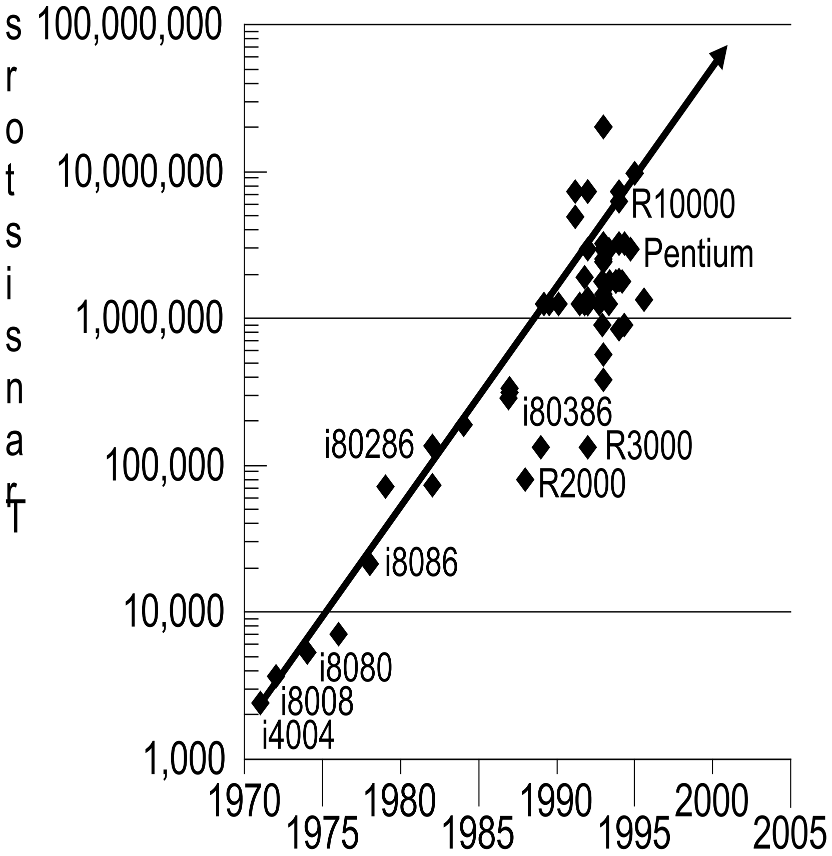
2000 — 1.0 GHz
2001 — 1.6 GHz
2002 — 2.6 GHz
2003 — 4.1 GHz
2004 — 10.1 GHz
2005 — 16.2 GHz

Actual 2005 — 3+ GHz

Growth Rates



- 30% per year



40% per year

Module 3: Arch. Implications

- Sequential architecture trends
- Stages in parallelism exploitation
- ILP limits
- Multithreading
- Parallel computing taxonomies
- Top500

Architectural Trends

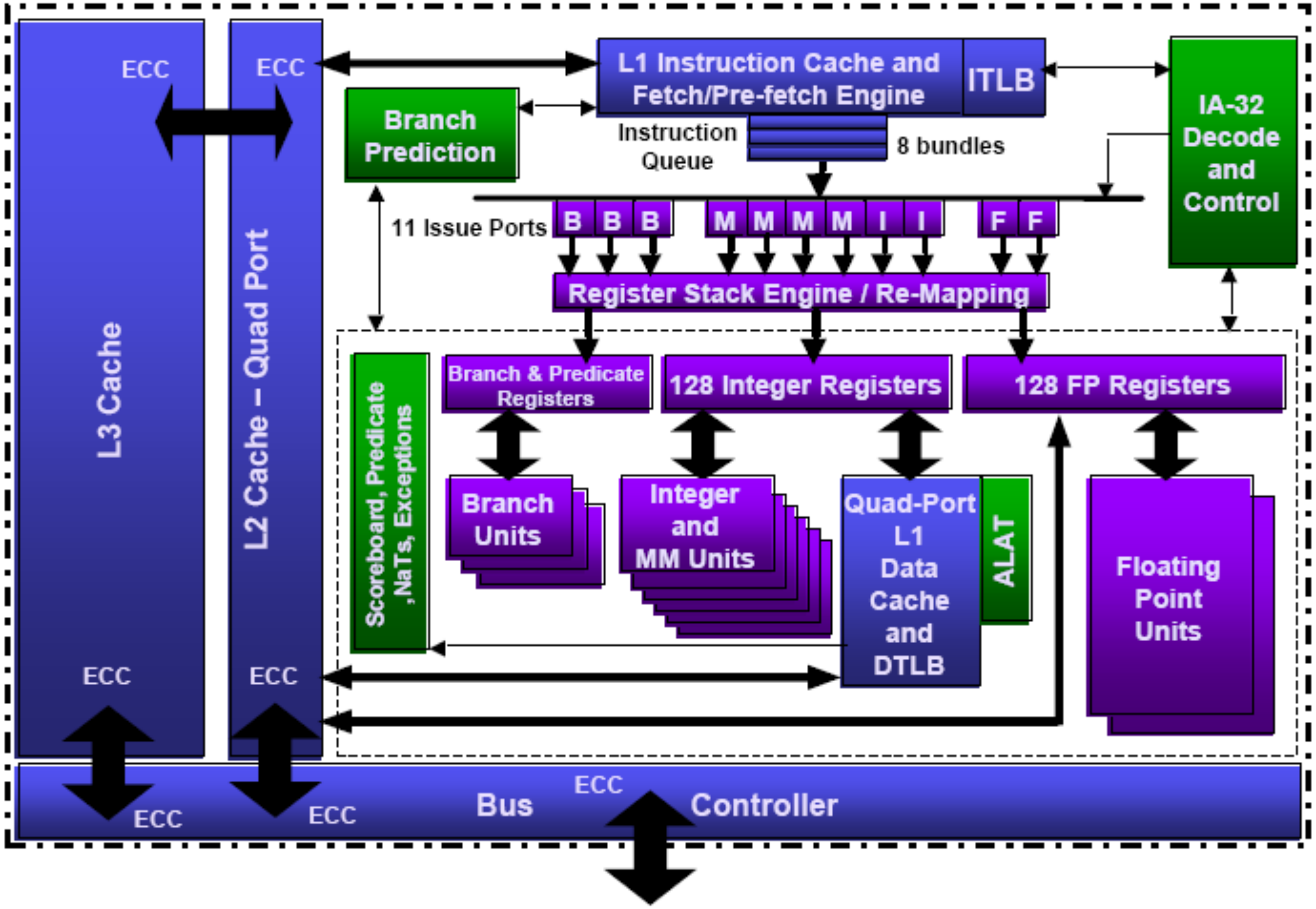
- Architecture: technology gains -> performance and functionality
- Tradeoff between parallelism and locality
  - Past microprocessors: 1/3 compute, 1/3 cache, 1/3 off-chip connect
  - Tradeoffs change with scale and technology advances
    - Most area taken by memories

- Understanding microprocessor architectural trends

  => Build intuition about design issues or parallel machines

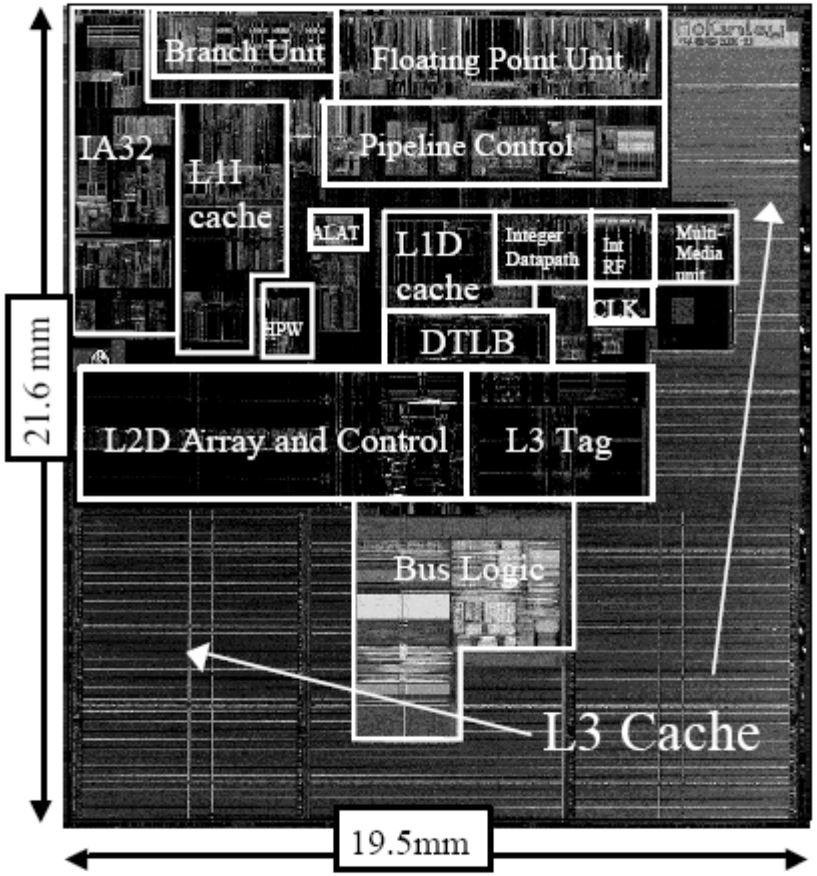
  => Fundamental role of parallelism even in “sequential” computers

Itanium Block Diagram

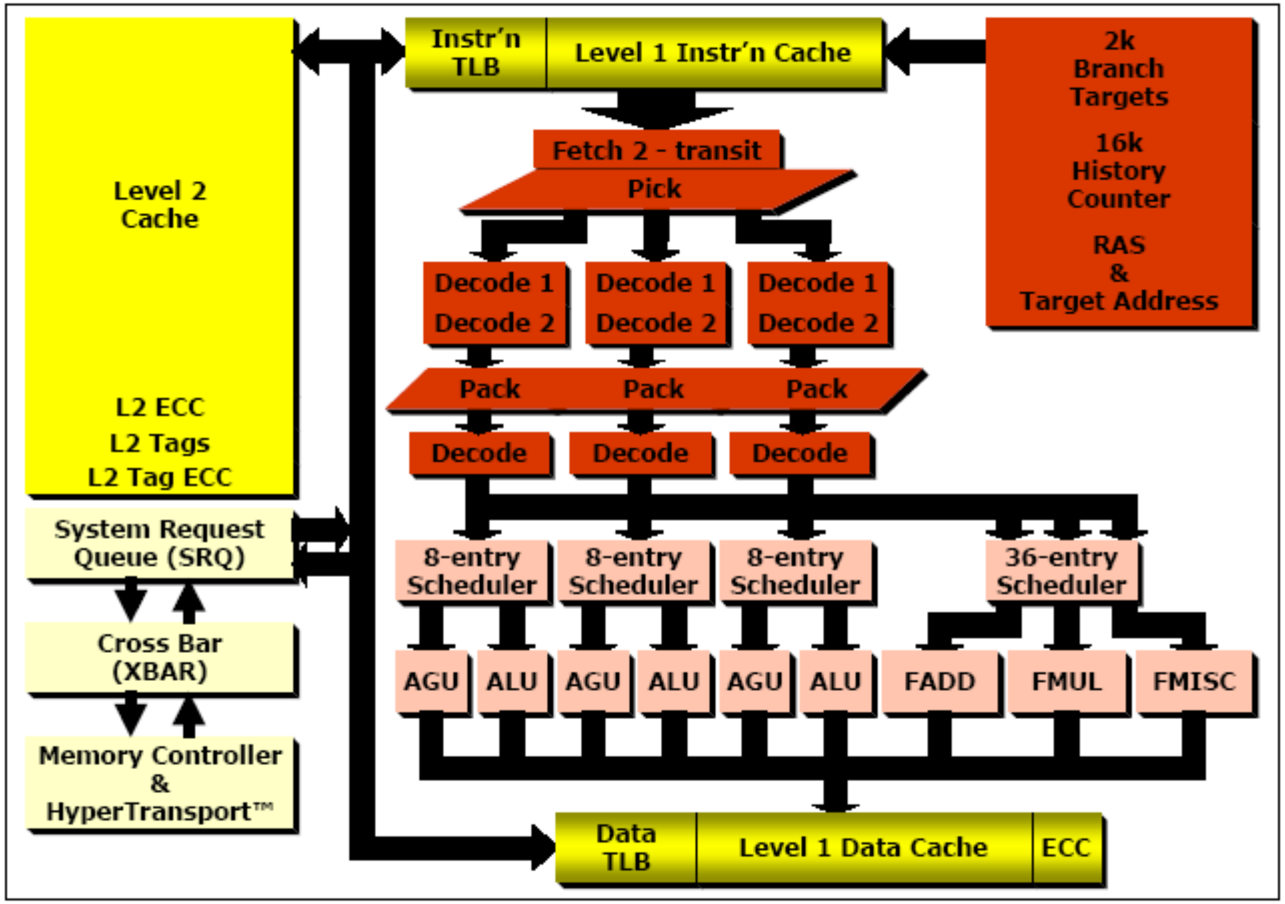


Itanium McKinley – A HPC Processor

- .18µm bulk, 6 layer Al process
- 8 stage, fully stalled in-order pipeline
- Symmetric six integer issue design
- 3 levels of cache on-die totaling 3.3MB
- 221 Million transistors
- 130W @1GHz, 1.5V

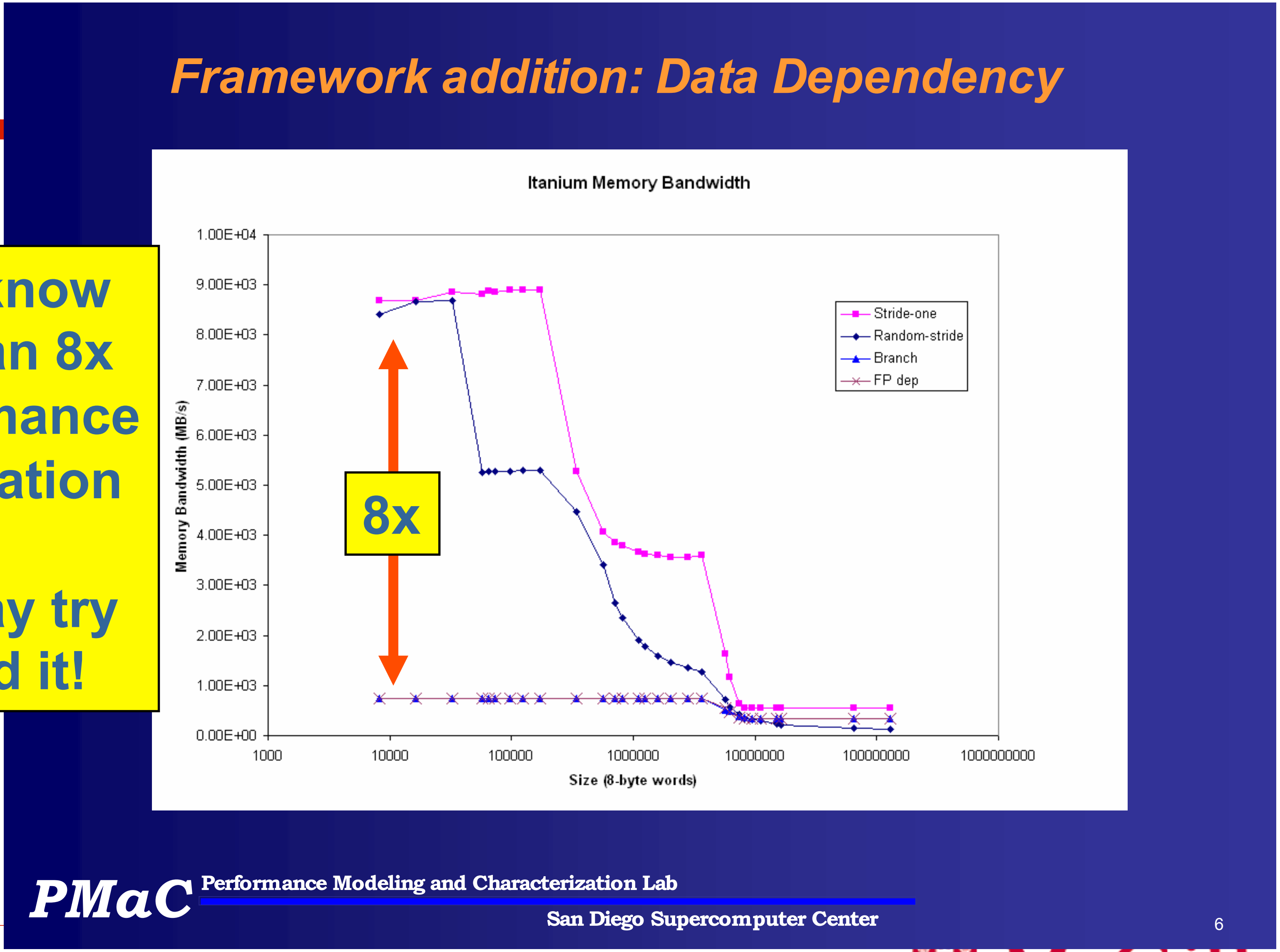


AMD Hammer uArchitecture



- **12-stage integer operation pipeline**
- **17-stage floating point operation pipeline**

Branch/Data Dependency - Itanium



Framework addition: Data Dependency

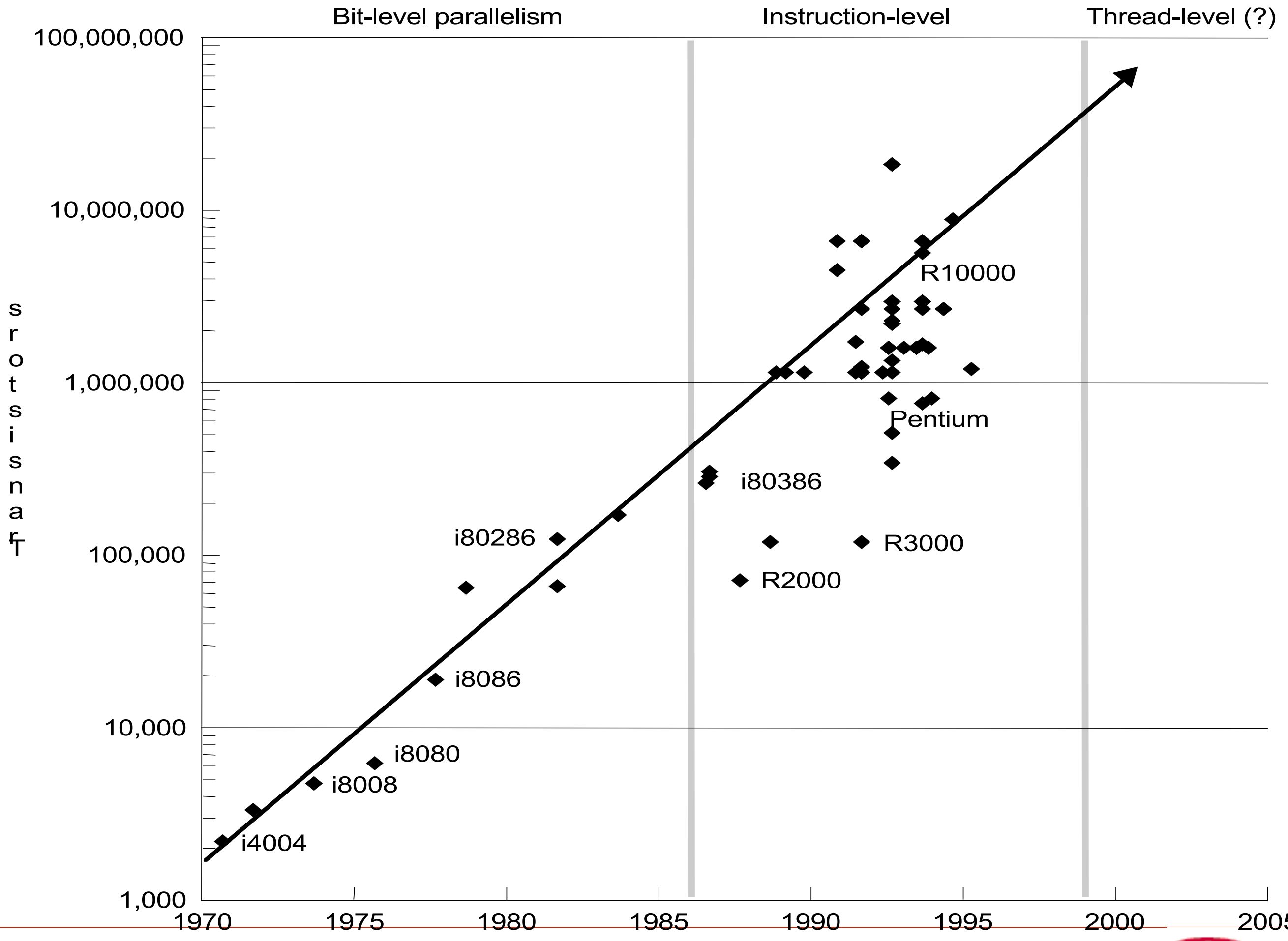
PMaC Performance Modeling and Characterization Lab

San Diego Supercomputer Center

If you know about an 8x performance degradation

You may try to avoid it!

Phases in VLSI Generation

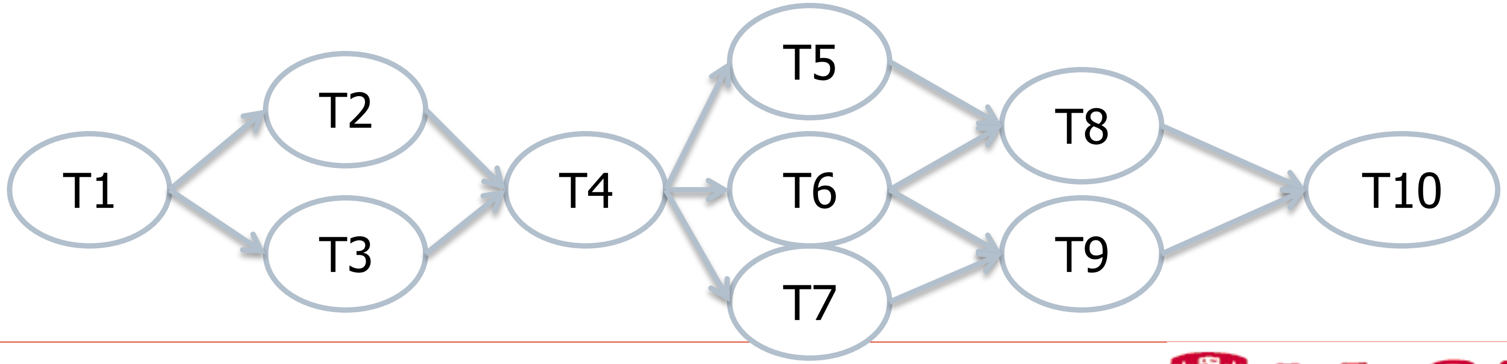


Architectural Trends

- Main trend in VLSI is increase in parallelism
  - Up to 1985: bit level parallelism: 8 bit -> 16-bit
    - Inflection at 32 bit – cache fits on a chip
    - Adoption of 64-bit under way, 128-bit far (no need)
  - Mid 80s to mid 90s: instruction level parallelism
    - Pipelining and RISC instruction sets + compiler
    - On-chip caches and functional units => superscalar
    - Out of order execution, speculation, prediction
      - Deals with control transfer and latency problems
  - Now: thread level parallelism
    - Hardware multi-threaded processors
    - Multi-core processors

Case: Showing Work for Parallel Execution

- Divide work into tasks that can be executed concurrently
- Many different decompositions possible for any computation
- Tasks may be same, different, or even unknown sizes
- Tasks may be independent or have non-trivial order
- Conceptualize tasks and ordering as task dependency graph
  - A *directed acyclic graph (DAG)*
  - Node = task
  - Edge or arc = control dependence



Example: Database Query

- Task: set of elements that satisfy a predicate -> table
- Edge: output of one task serves as input to the next

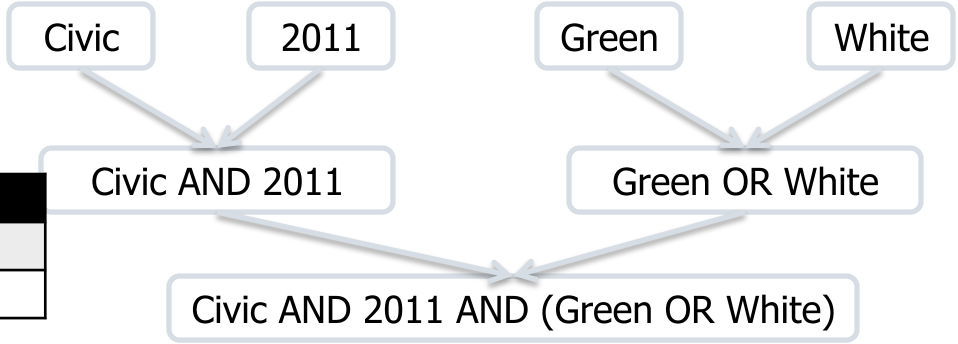
MODEL="CIVIC" AND YEAR=2011 AND (COLOR="GREEN" OR COLOR="WHITE")

ID#	Model
4523	Civic
6734	Civic
4395	Civic
7352	Civic

ID#	Year
7623	2011
9834	2011
6734	2011
5342	2011
3845	2011
4395	2011

ID#	Color
7623	Green
9834	Green
5342	Green
8354	Green

ID#	Color
3476	White
6734	White



ID#	Color
3476	White
7623	Green
9834	Green
6734	White
5342	Green
8354	Green

ID#	Model	Year
6734	Civic	2011
4395	Civic	2011

ID#	Model	Year	Color	Dealer	Price
6734	Civic	2011	White	BC	$17,000