The Intel® Pentium® 4 Processor

Doug Carmean
Principal Architect
Intel Architecture Group

Spring 2002

Intel® PDX

Agenda

- Review
- Pipeline Depth
- Execution Trace Cache
- Data Speculation
- Spec Performance
- Summary

Information in this document is provided in connection with Intel® products. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted by this document. Except as provided in Intel's Terms and Conditions of Sale for such products, Intel assumes no liability whatsoever, and Intel disclaims any express or implied warranty, relating to sale and/or use of Intel® products including liability or warranties relating to fitness for a particular purpose, merchantability, or infringement of any patent, copyright or other intellectual property right. Intel products are not intended for use in medical, life saving, or life sustaining applications.

Intel may make changes to specifications and product descriptions at any time, without notice.

This document contains information on products in the design phase of development. Do not finalize a design with this information. Revised information will be published when the product is available. Verify with your local sales office that you have the latest datasheet before finalizing a design.

Intel processors may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Intel, Pentium, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and foreign countries.

Intel® PDX

Intel® Netburst™ Micro-architecture vs P6

Basic P6 Pipeline

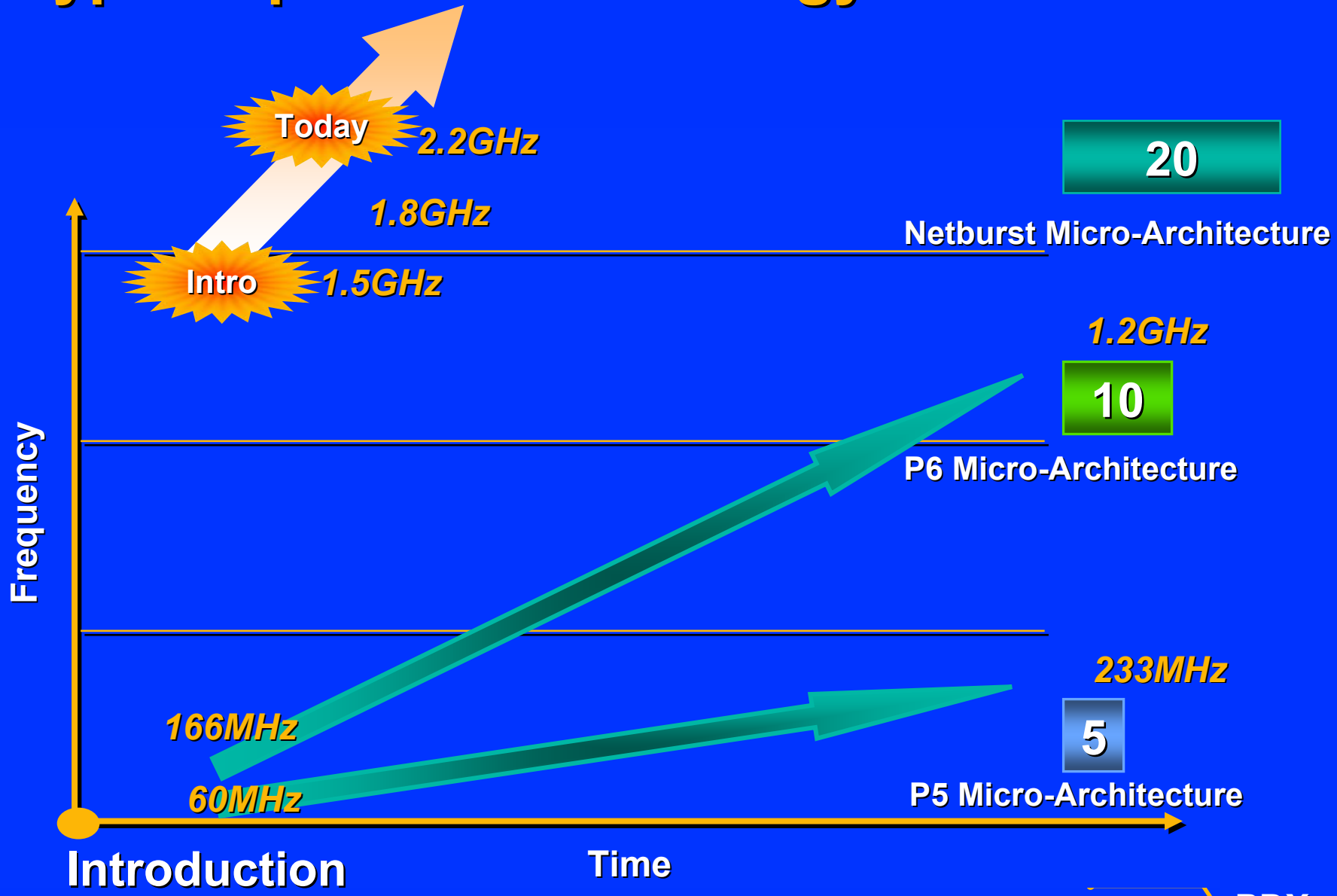
1	2	3	4	5	6	7	8	9	10
Fetch	Fetch	Decode	Decode	Decode	Rename	ROB Rd	Rdy/Sch	Dispatch	Exec

Basic Pentium® 4 Processor Pipeline

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
TC Nxt IP		TC Fetch		Drive	Alloc	Rename		Que	Sch	Sch	Sch	Disp	Disp	RF	RF	Ex	Flgs	Br Ck	Drive

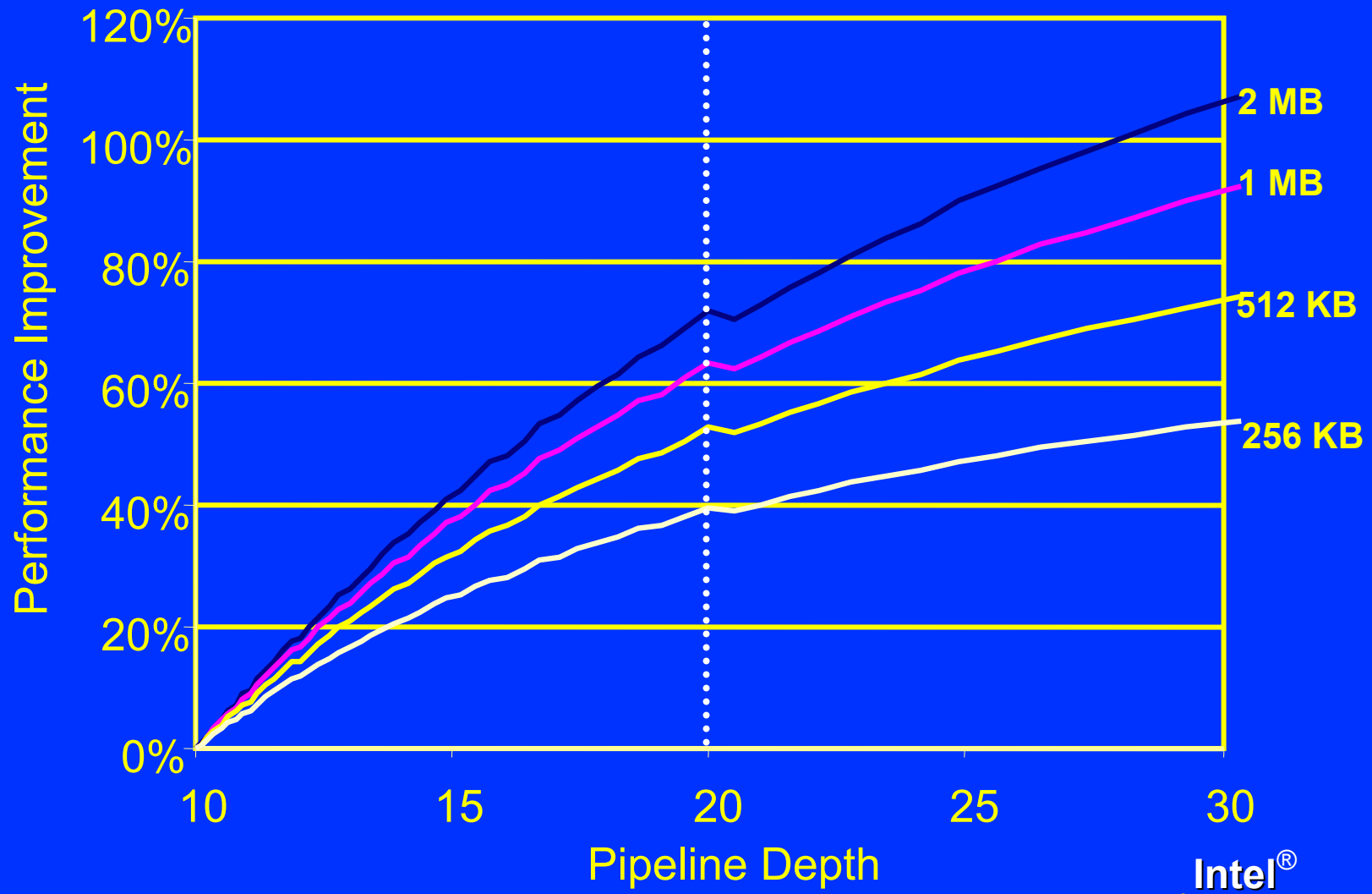
Deeper Pipelines enable higher frequency and performance

Hyper Pipelined Technology



PDX

Deeper Pipelines are Better



Source: Average of 2000 application segments from performance simulations

Intel®

PDX

Why not deeper pipelines?

- Increases complexity
  - Harder to balance
  - More challenges to architect around
  - More algorithms
  - Greater validation effort
  - Need to pipeline the wires

Overall Engineering Effort Increases Quickly as Pipeline depth increases

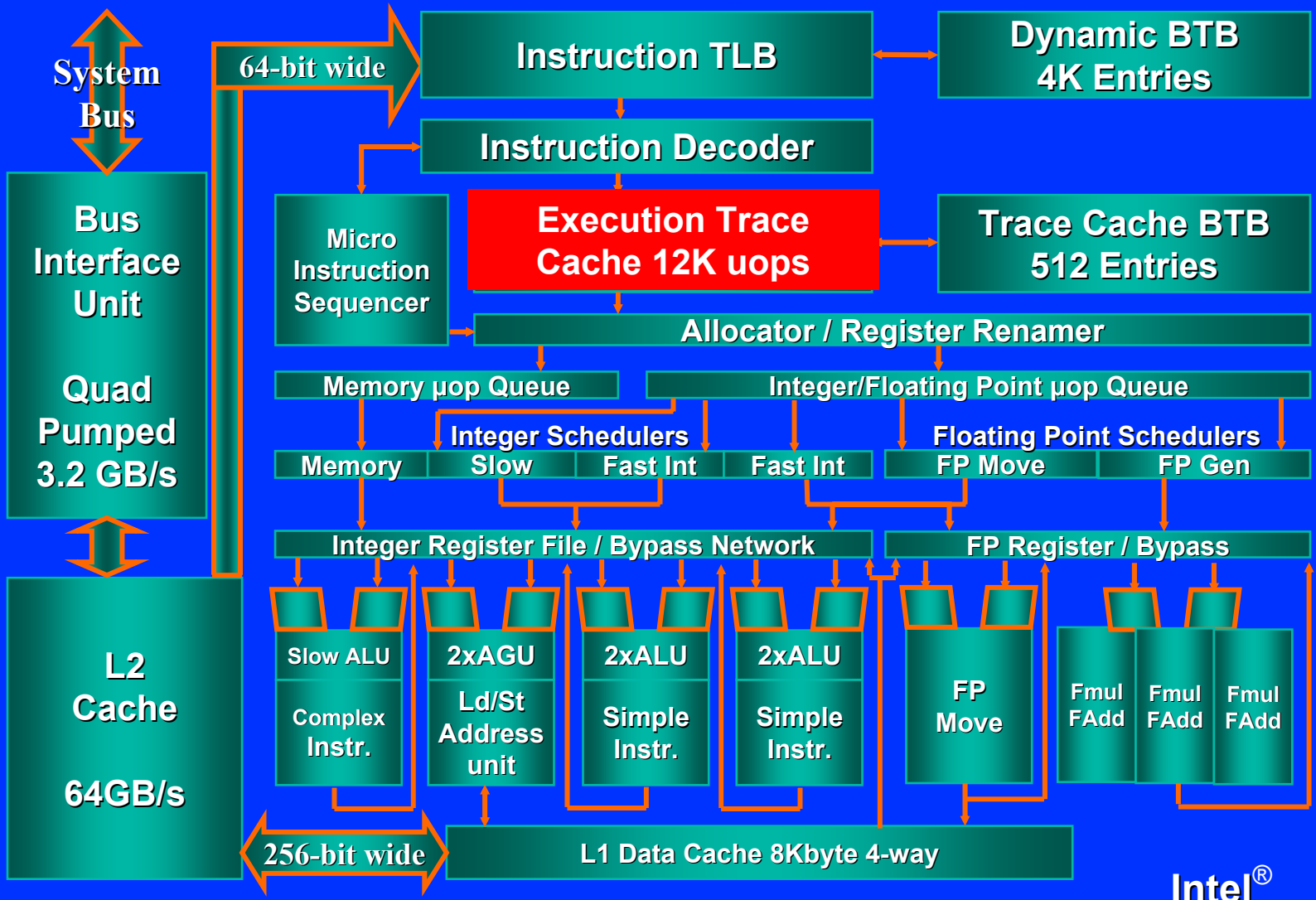
Performance

- High
- I

Higher Frequency increases requirements of front end

- Branch prediction is more important
  - So we improved it
- Need greater uop bandwidth
  - Branches constantly change the flow
  - Need to decode more instructions in parallel

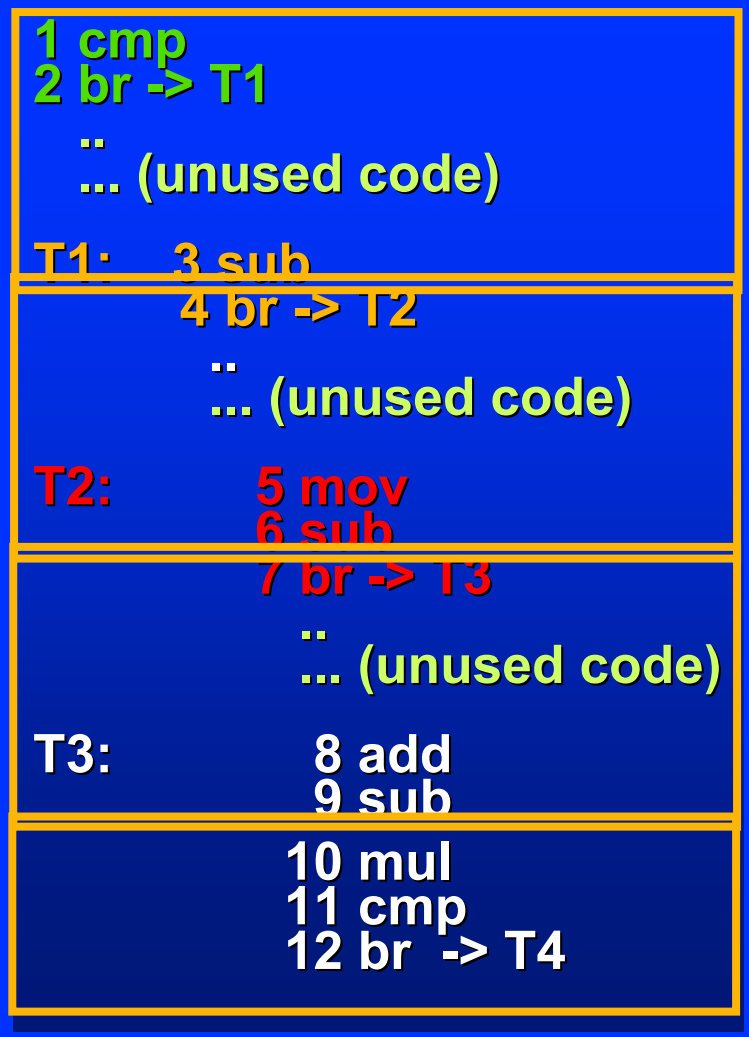
Block Diagram



Intel®

PDX

Execution Trace Cache



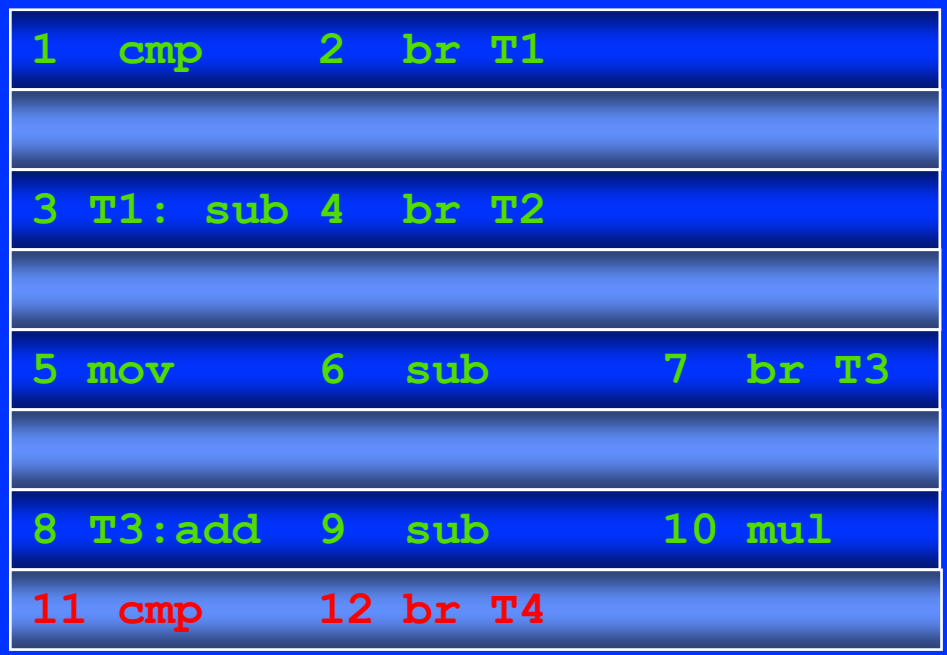
Trace Cache Delivery

1 cmp	2 br T1	3 T1: sub
4 br T2	5 mov	6 sub
7 br T3	8 T3:add	9 sub
10 mul	11 cmp	12 br T4

Intel® PDX

Execution Trace Cache

P6 Microarchitecture



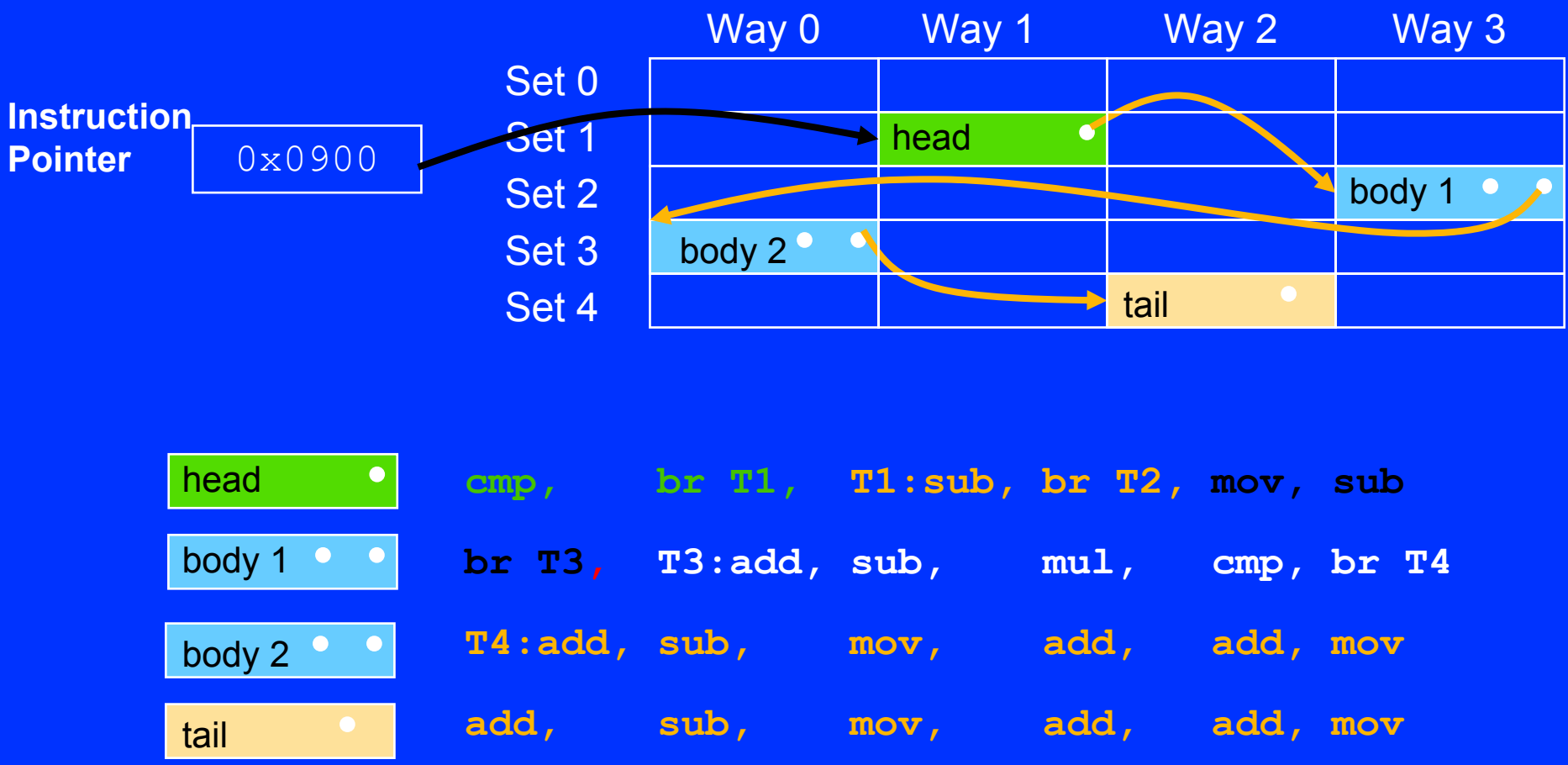
BW = 1.5 uops/ns

Trace Cache Delivery

BW = 6 uops/ns

Intel® PDX

Inside the Execution Trace Cache



Intel® PDX

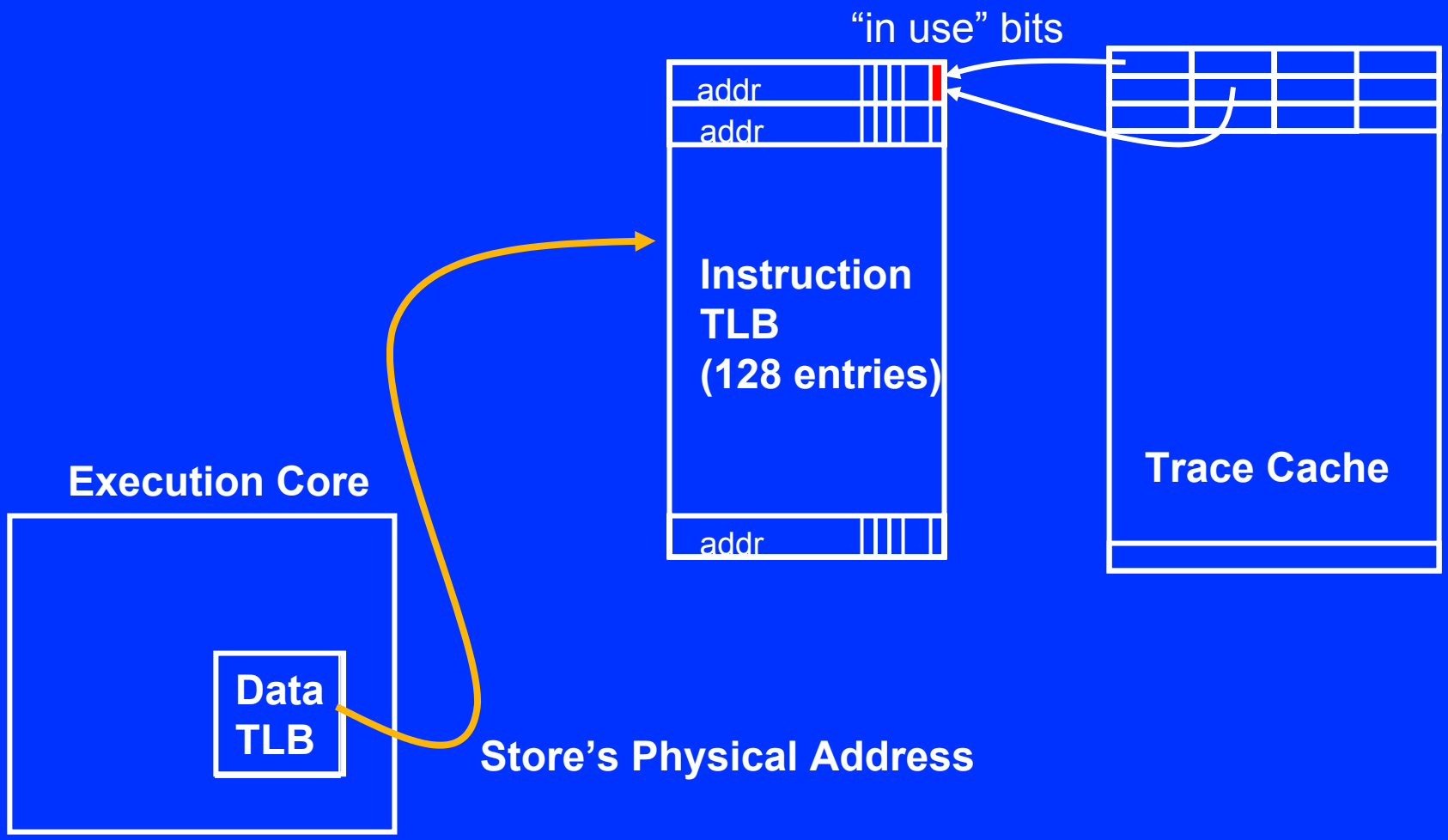
Self Modifying Code

- Programs that modify the instruction stream that is being executed
- Very common in Java* code from JITs
- Requires hardware mechanisms to maintain consistency

Self Modifying Code

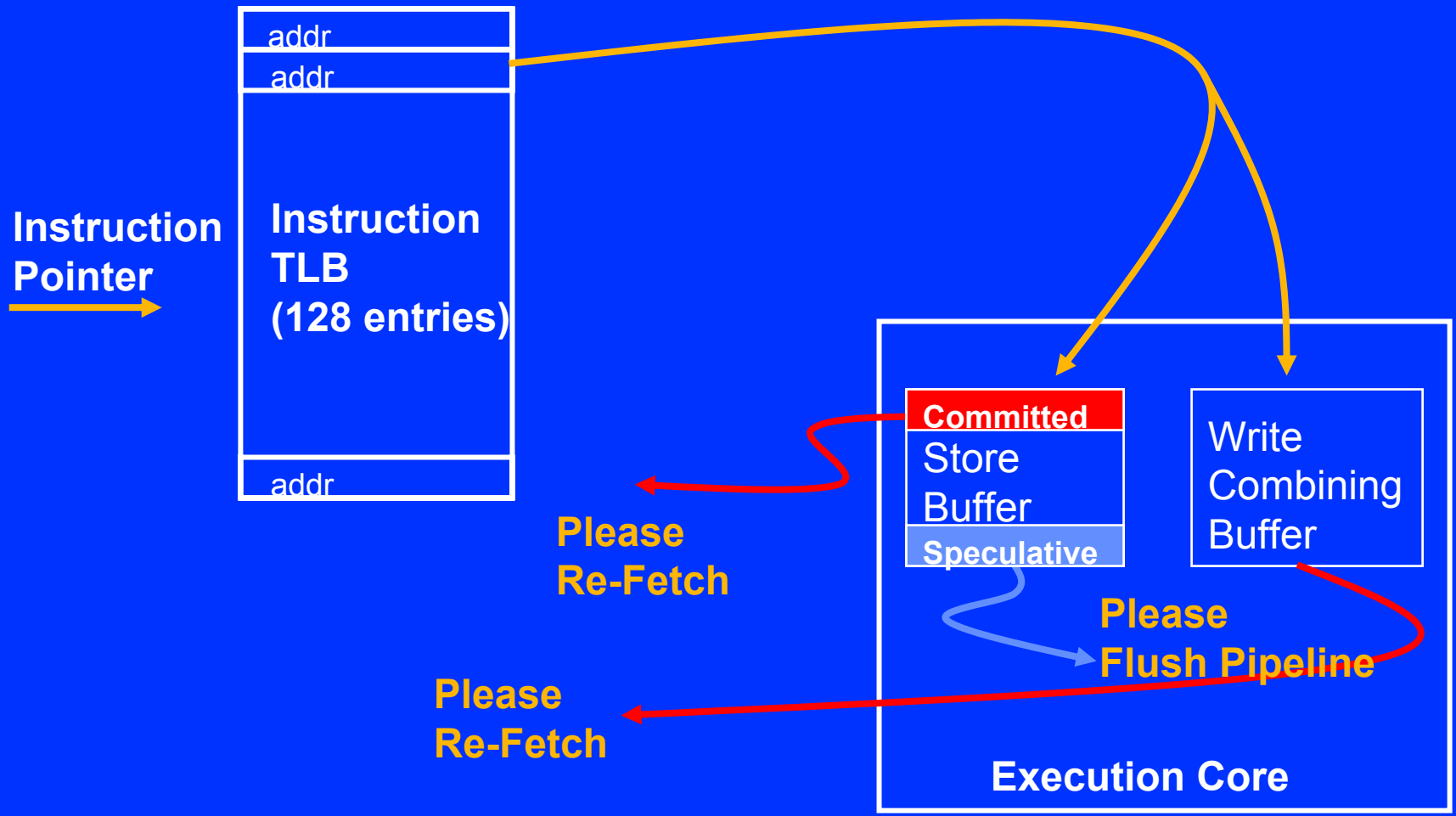
- The hardware needs to handle two basic cases:
  - Stores that write to instructions in the Trace Cache
  - Instruction fetches that hit pending stores
    - Speculative
    - Committed

Case 1: Stores to cached instructions



Intel®
PDX

Case 2: Fetches to pending stores



Intel® PDX

Execution Trace Cache

- Provides higher bandwidth for higher frequency core
- Reduces fetch latency
- Requires new fundamentally new algorithms

Performance

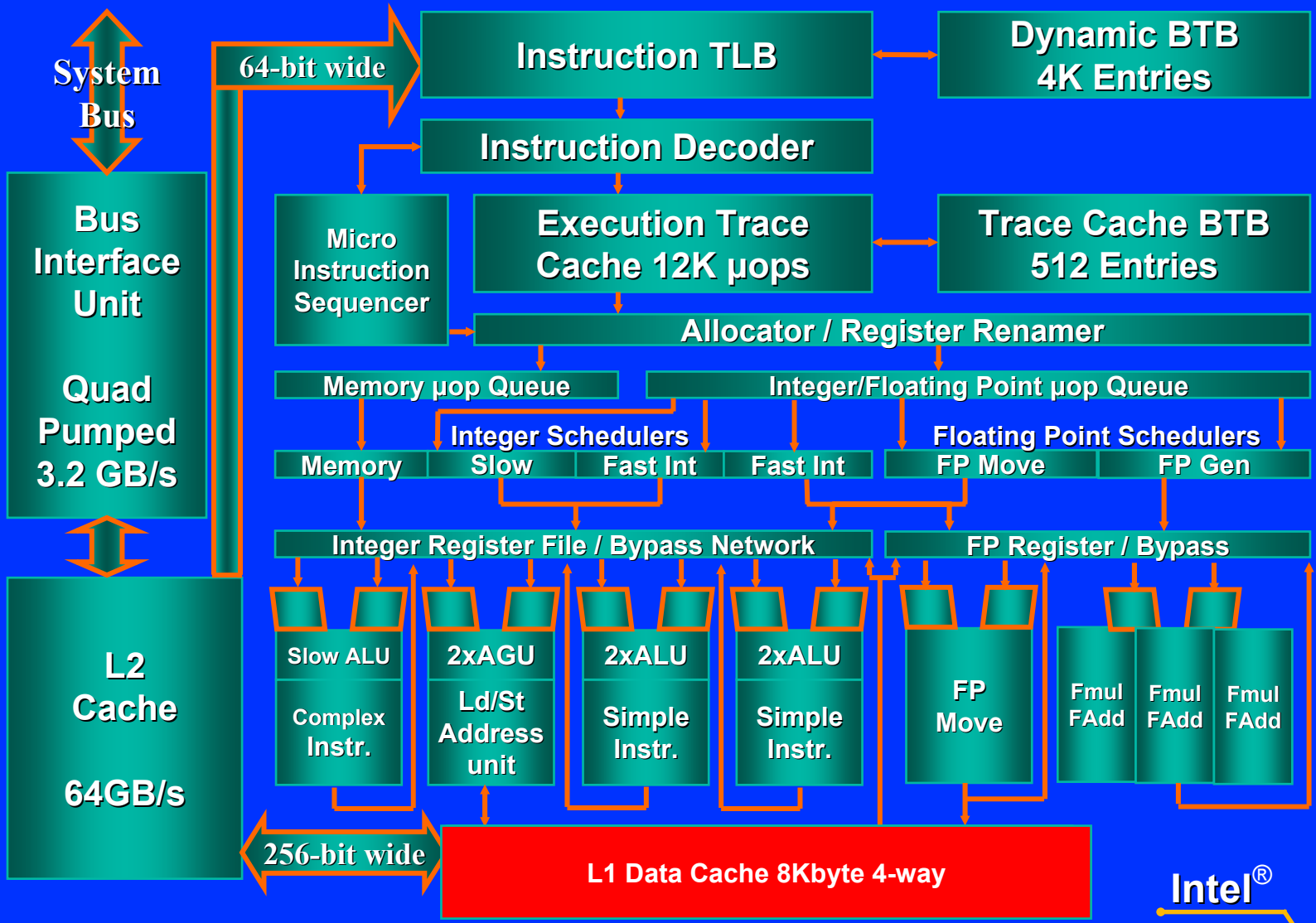
- High bandwidth front end
- Lo

Data Speculation

- Use data before we are sure it is valid
  - Lowers effective LD latency
  - Fast ALUs in Pentium 4 want fast LDs
  - Ratio of LD latency to ADD latency is important if 1 in 5 uops is a LD
- As pipelines get deeper, data speculation gets more important
  - Number of cycles saved /w data speculation increases as pipeline depth increases

Intel® PDX

L1 Data Cache

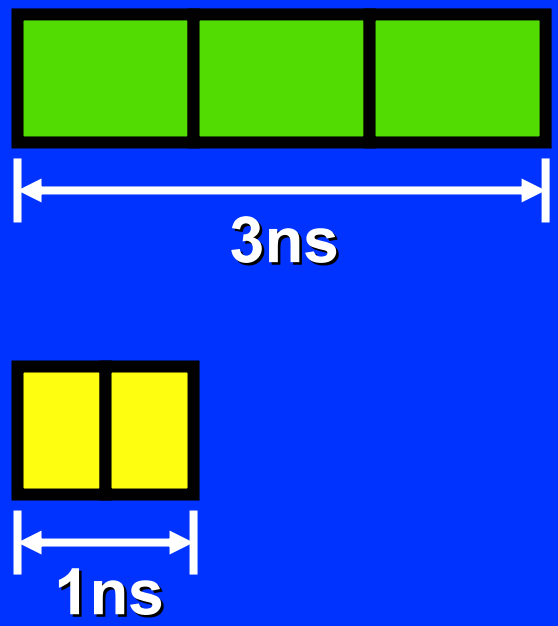


Intel®

PDX

L1 Cache is >3x Faster

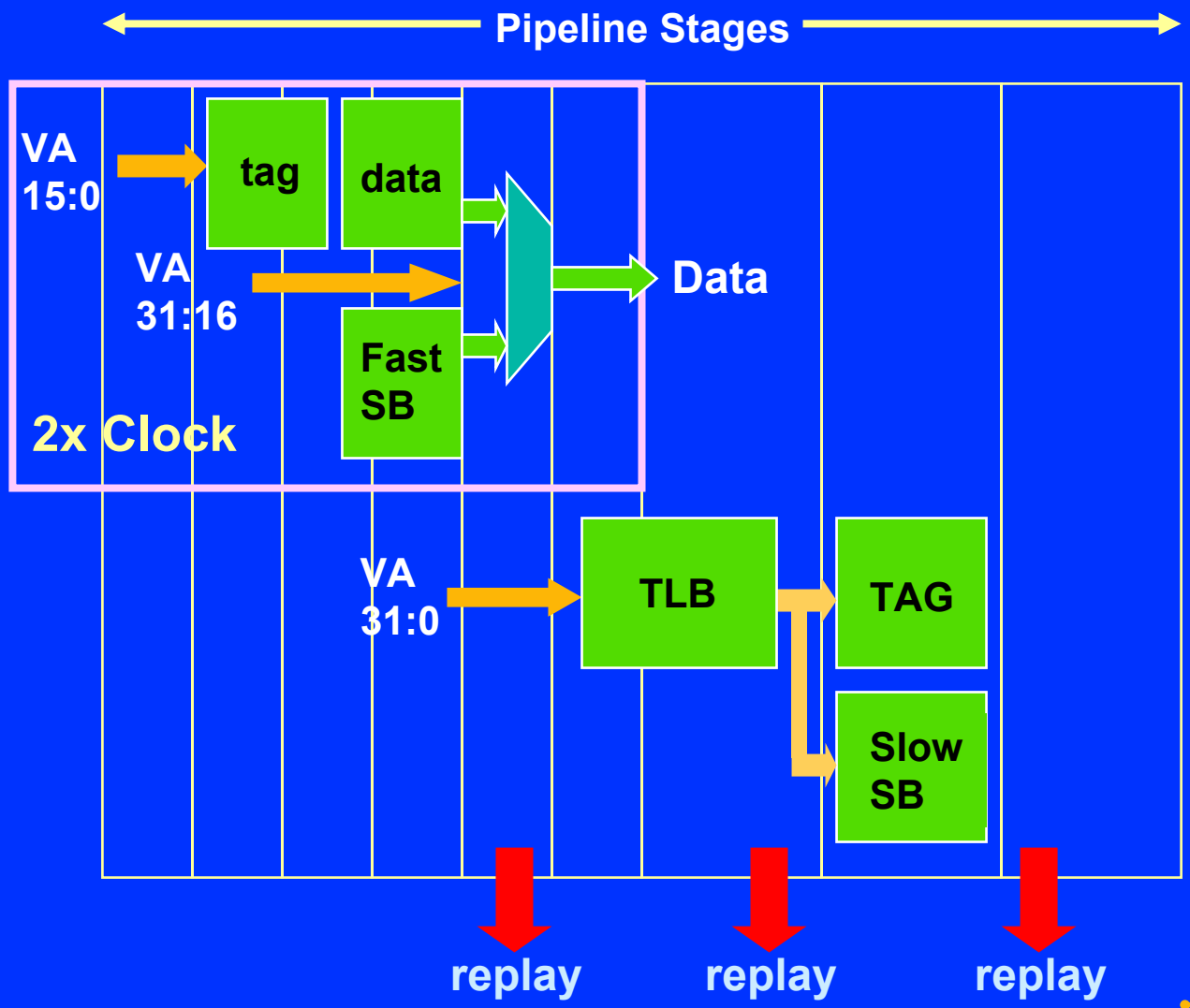
- P6:
  - 3 clocks @ 1GHz
- P4P:
  - 2 clocks @ 2GHz



Lower Latency is Higher Performance

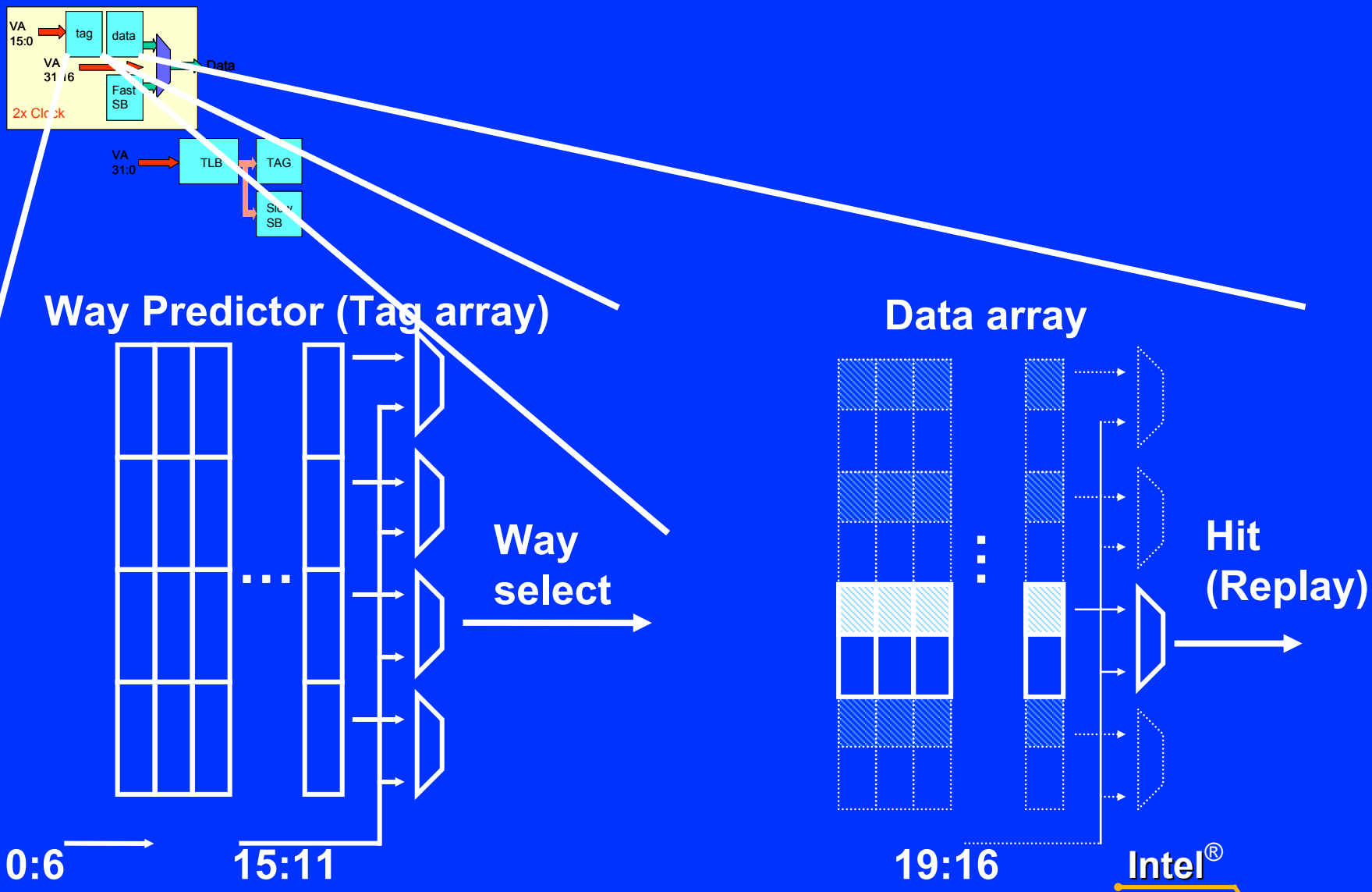
Intel®
PDX

L1 Data Cache



Intel® PDX

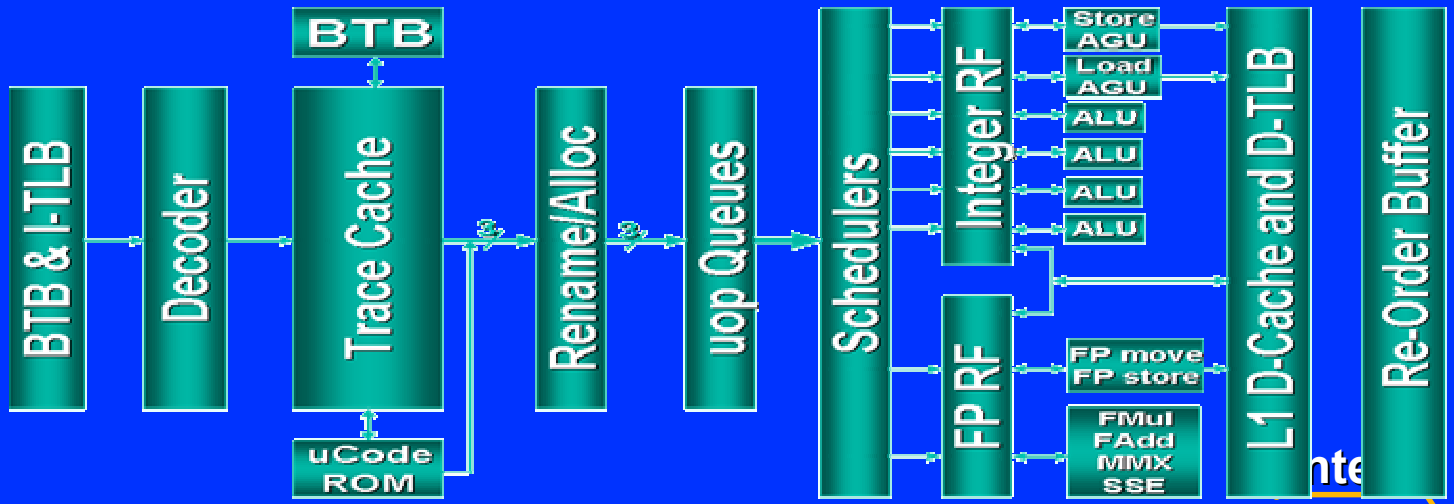
L1 Data Cache



Intel®
PDX

A Digression on Stores

- Two components to a store:
  - STA: address computation
  - STD: data piece
- Hybrid uOP
  - Single uOP in the front, back ends
  - Two uOPs in the middle



PDX

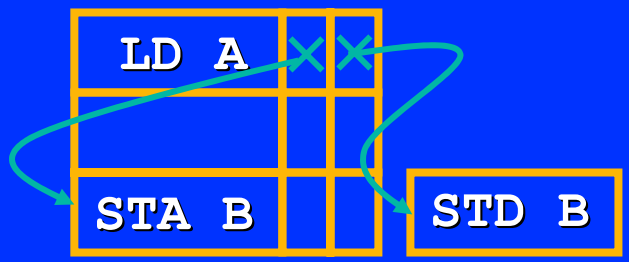
Memory Disambiguation

```
Ld EAX <- AddrA
```

- If the store is older
  - And AddrA = AddrB
  - Then the load must get DataB
- Dependencies can not be resolved until execution

Memory Disambiguation

Option 1



Loads wait for:

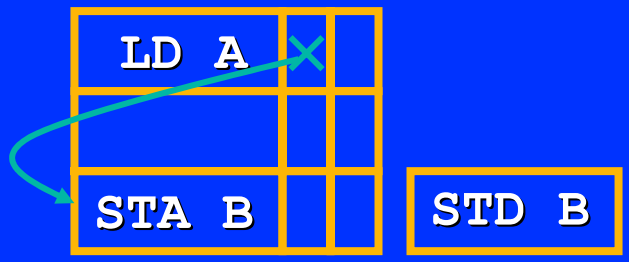
All older STAs AND

All older STDs

No Recovery

K7?

Option 2



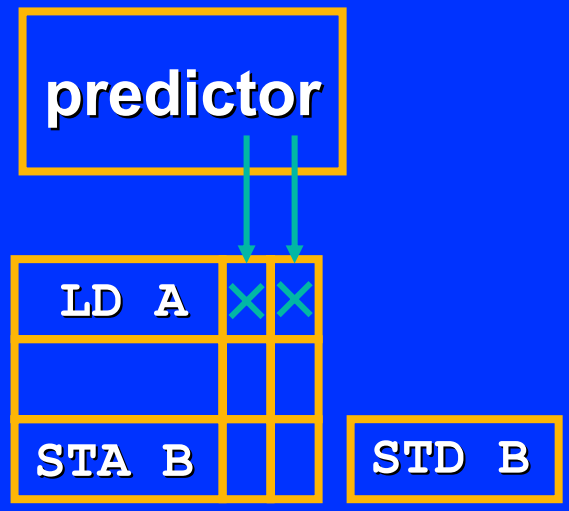
Loads wait for:

All older STAs

STD Recovery

P4

Option 3



Predict

Specific older STAs

Specific older STDs

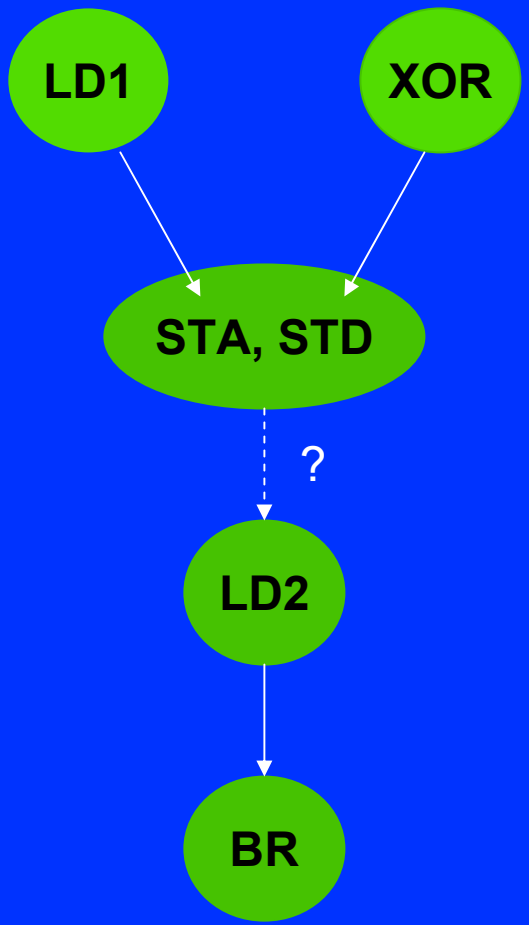
Complex Recovery

EV8

Intel®

PDX

Example



ML=0

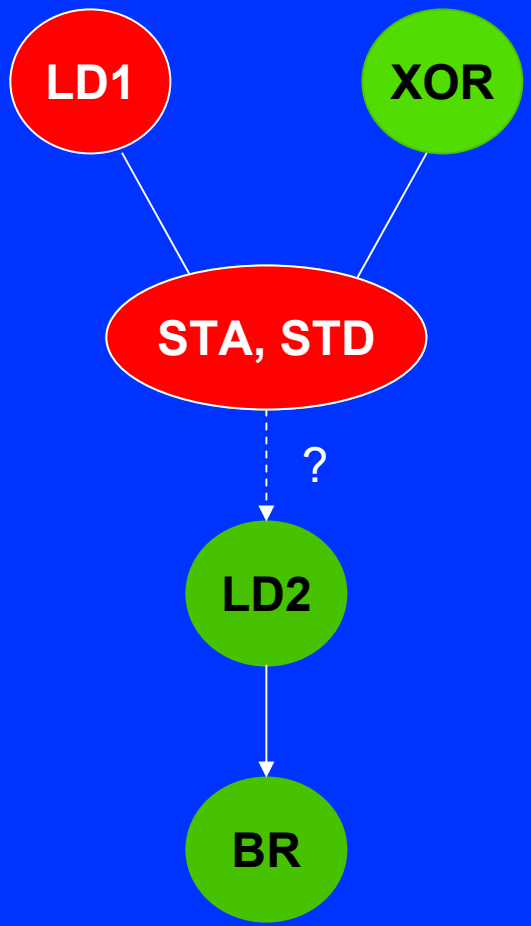
ML=1

JMP if ML=0

- ST writes new value
- LD2 forwards
- Branch resolves based on new value

Intel® PDX

Example



ML=0

LD misses, replays

ML=1

STA dependent, replays

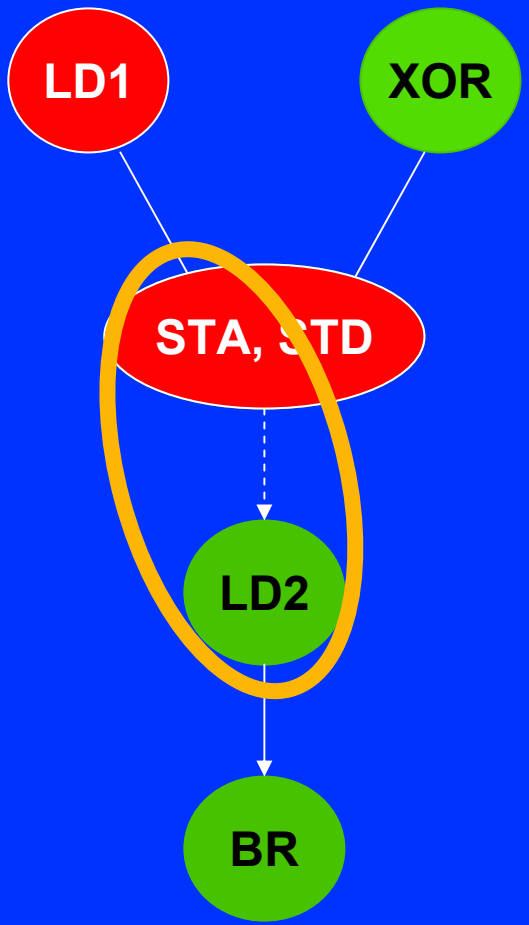
LD hits, gets old value

BR mispredicts

JMP if ML=0

Intel® PDX

Example



- If LD2 depends on the STA, they are usually part of the same dataflow graph
- If the STA replays, the LD usually has an address dependence

Cautious Mode

- Normally, aggressively schedule
- If a large number of problems occur enter Cautious Mode
- In Cautious Mode, branches wait for data to be non-speculative
  - Increases branch misprediction latency
  - Completely eliminates problems

Cautious Mode: Implementation

- A simple state machine cleans up the outliers
  - Out of 2200 traces, 3 traces speedup >20%
  - The other traces are unaffected
  - Average performance improvement < 0.1%

Performance

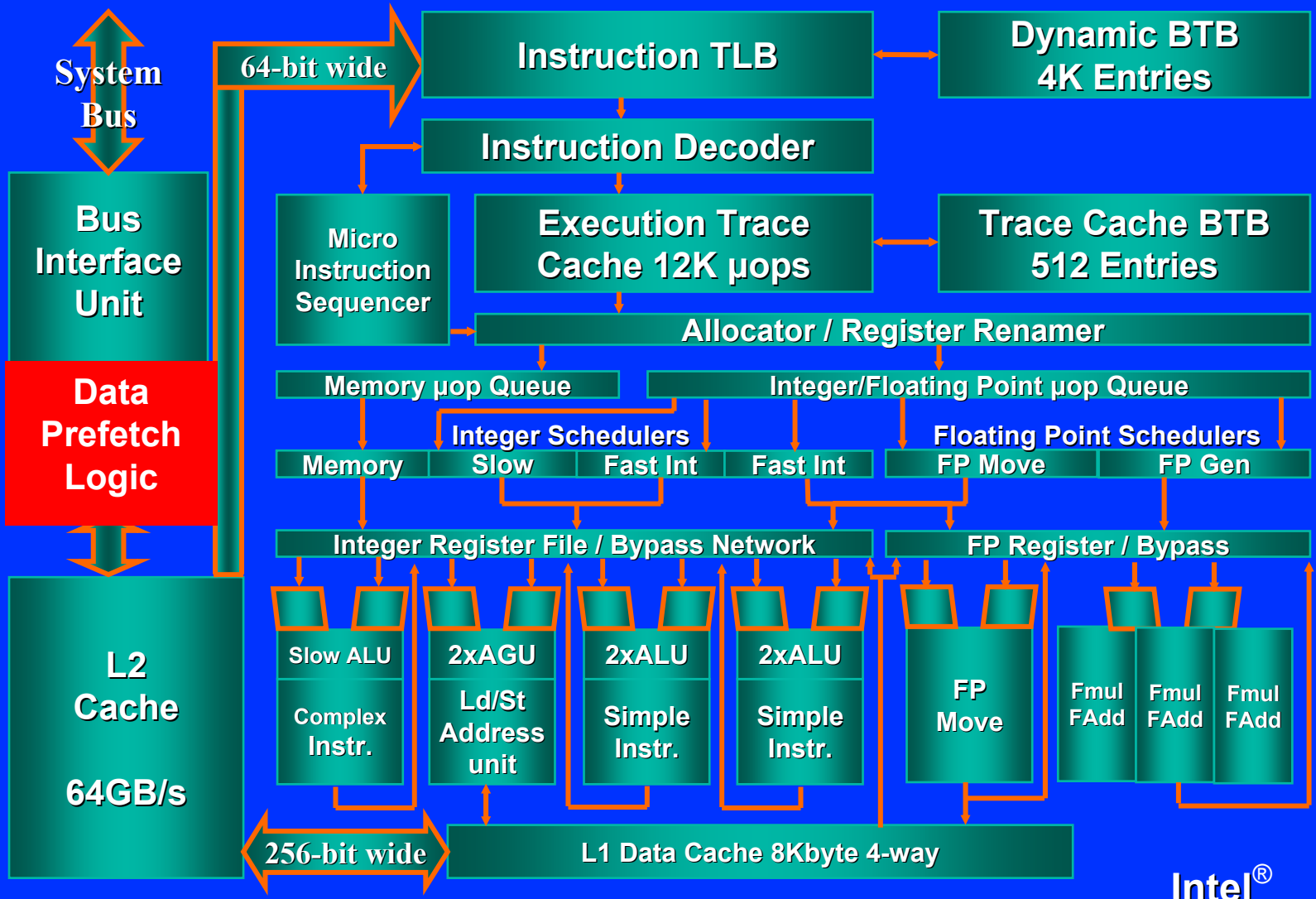
- High bandwidth front end
- Low latency core
- Low

Lower Memory Latency

Reducing Latency

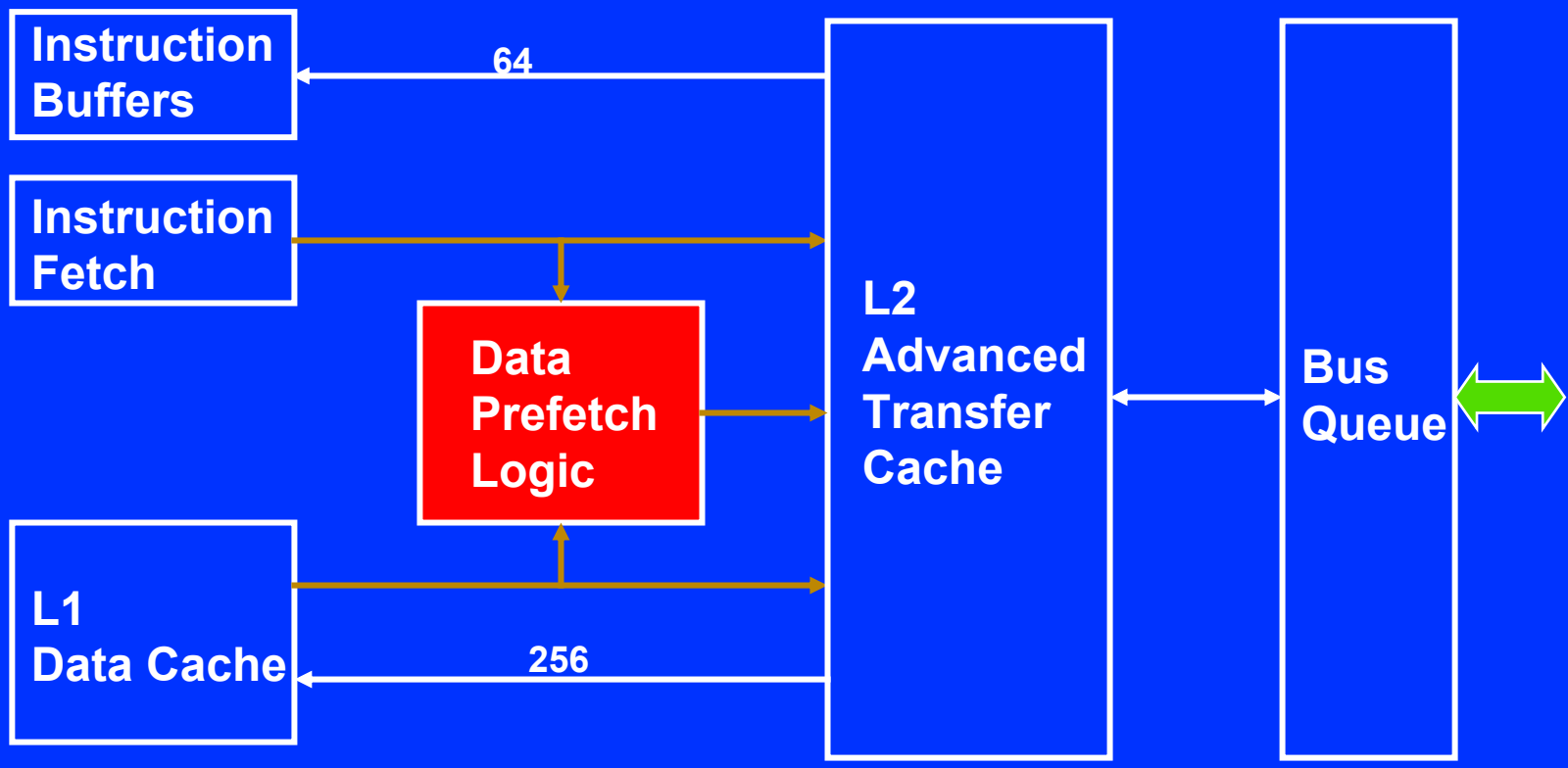
- As frequency increases, it is important to improve the performance of the memory subsystem
- Data Prefetch Logic
  - Watches processor memory traffic
  - Looks for patterns
  - Initiates accesses

Data Prefetch Logic



Intel® PDX

Data Prefetch Logic



Prefetch logic first checks L2 cache and then fetches lines from memory that miss L2 cache.

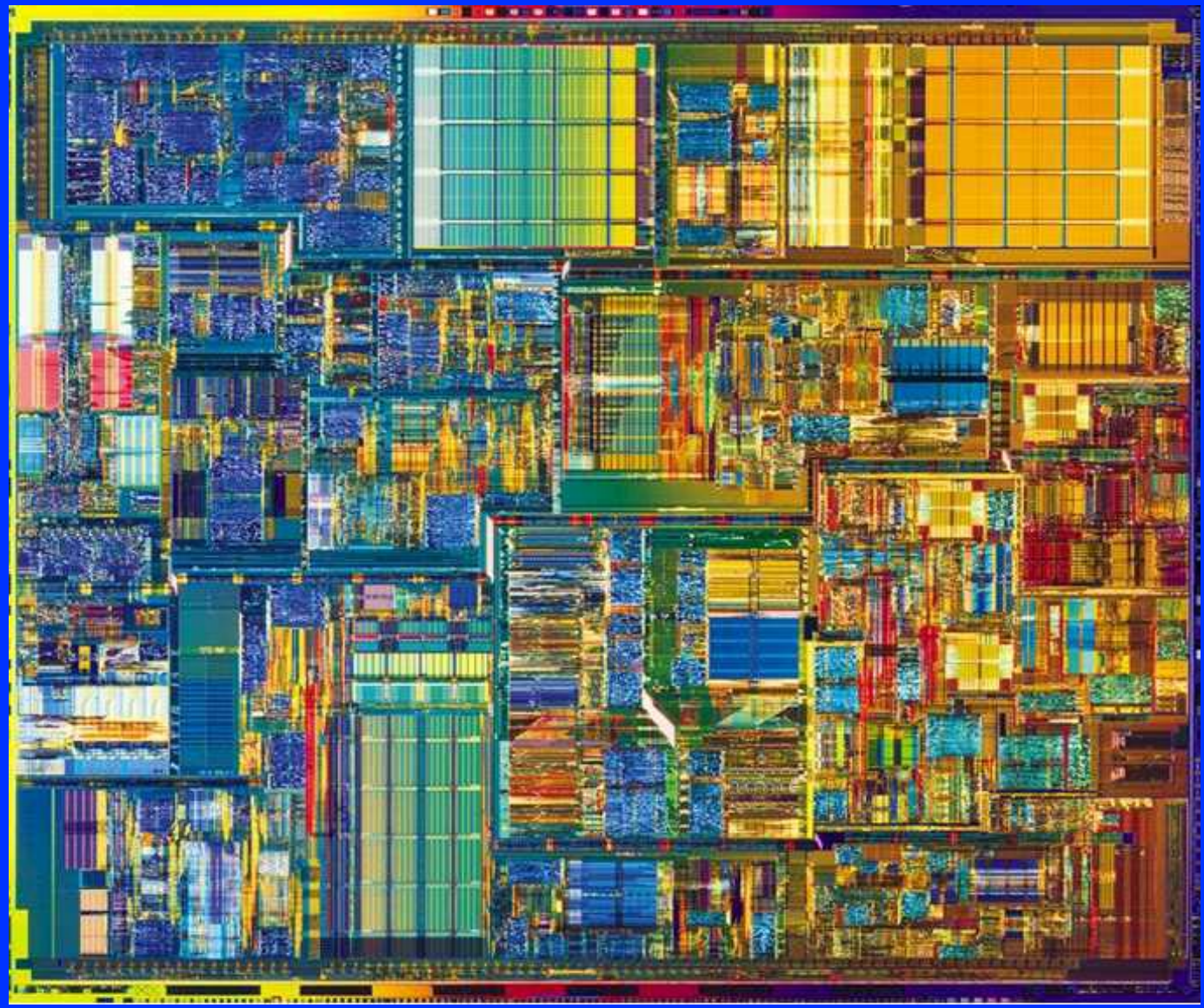
Data Prefetch Logic

- Watches for streaming memory access patterns
  - Can track 8 multiple independent streams
  - Loads, Stores or Instruction
  - Forward or Backward
- Analysis on 32 byte cache line granularity
- Looks for “mostly” complete streams:
  - Access to cache lines 1,2,3,4,5,6 will prefetch
  - Access to cache lines 1,2, 4,5,6 will prefetch
  - 1, ,3, , ,6, , ,9 will not prefetch

Intel® PDX

Performance

- High bandwidth front end
- Low latency core
- Lower memory latency



Intel®

PDX