Hyper-Threading Technology Architecture and Micro-Architecture

Originally Prepared by Tahir Celebi

Istanbul, 2005

Salvatore Venticinque

Università degli Studi della Campania *Luigi Vanvitelli*

Outline

PII, PIII

Sia il Pentium II che il Pentium III si caratterizzano per:

- modifiche in termini di memorie di maggiori dimensioni
- per gli algoritmi di esecuzione dinamica che consentono al processore:
  - di eseguire le istruzioni delle applicazioni in modo molto più efficiente,
  - conservando allo stesso tempo la compatibilità software.

Introduction

- Hyper-Threading technology makes a single processor appear as two logical processors.

- It was first implemented in the Prestonia version of the Pentium® 4 Xeon processor on 02/25/02.

Traditional Approaches (I)

- High requirements of Internet and Telecommunications Industries
- Results are unsatisfactory compared the gain they provide with the cost they cause
- Well-known techniques;
  - Super Pipelining
  - Branch Prediction
  - Super-scalar Execution
  - Out-of-order Execution
  - Fast memories (Caches)

Traditional Approaches (II)

- Super Pipelining:
  - Have finer granularities, execute far more instructions within a second (Higher clock frequencies)
  - Hard to handle cache misses, interrupts and branch mispredictions
- Instruction Level Parallelism (ILP)
  - Mainly targets to increase the number of instructions within a cycle
  - Super Scalar Processors with multiple parallel execution units
  - Execution needs to be verified for out-of-order execution
- Fast Memory (Caches)
  - To reduce the memory latencies, hierarchical units are using which are not an exact solution

Traditional Approaches (III)

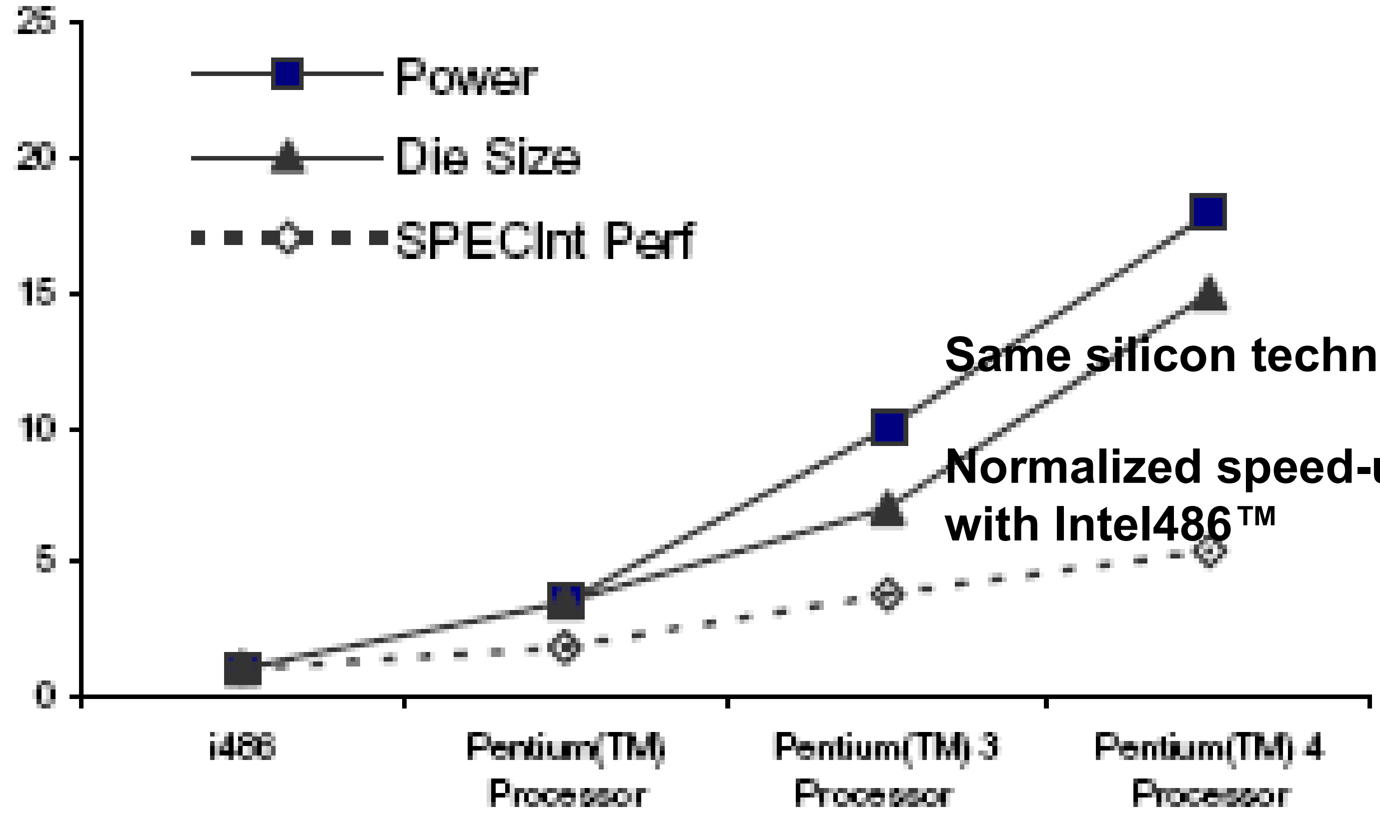


Figure 1: Single-stream performance vs. cost

Thread-Level Parallelism

- Chip Multi-Processing (CMP)
  - Put 2 processors on a single die
  - Processors (only) may share on-chip cache
  - Cost is still high
  - IBM Power4 PowerPC chip

- Single Processor Multi-Threading;
  - Time-sliced multi-threading
  - Switch-on-event multi-threading
  - Simultaneous multi-threading

Hyper-Threading (HT) Technology

- Provides more satisfactory solution
- Single physical processor is shared as two logical processors
- Each logical processor has its own architectu
- Single set of execution units are shared betv
- N-logical PUs are supported
- Have the same gain % with only 5% die-size
- **HT allows single processor to fetch and exe streams simultaneously.**

Figure 2: Processors without Hyper-Threading Tech

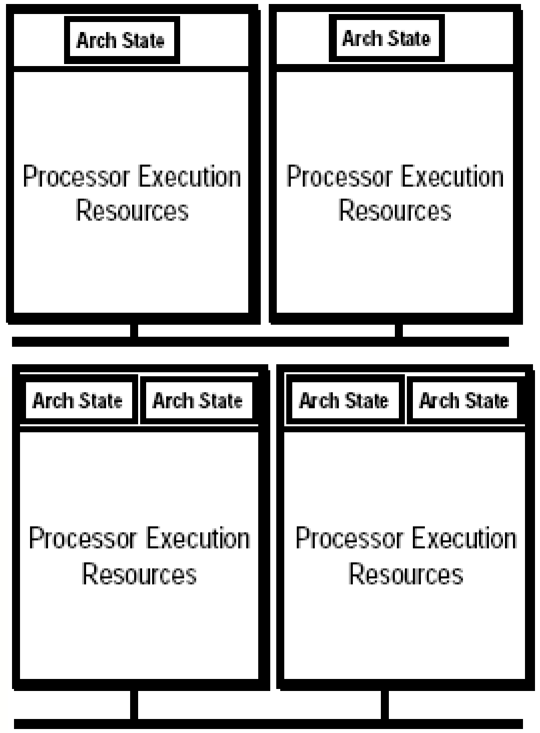
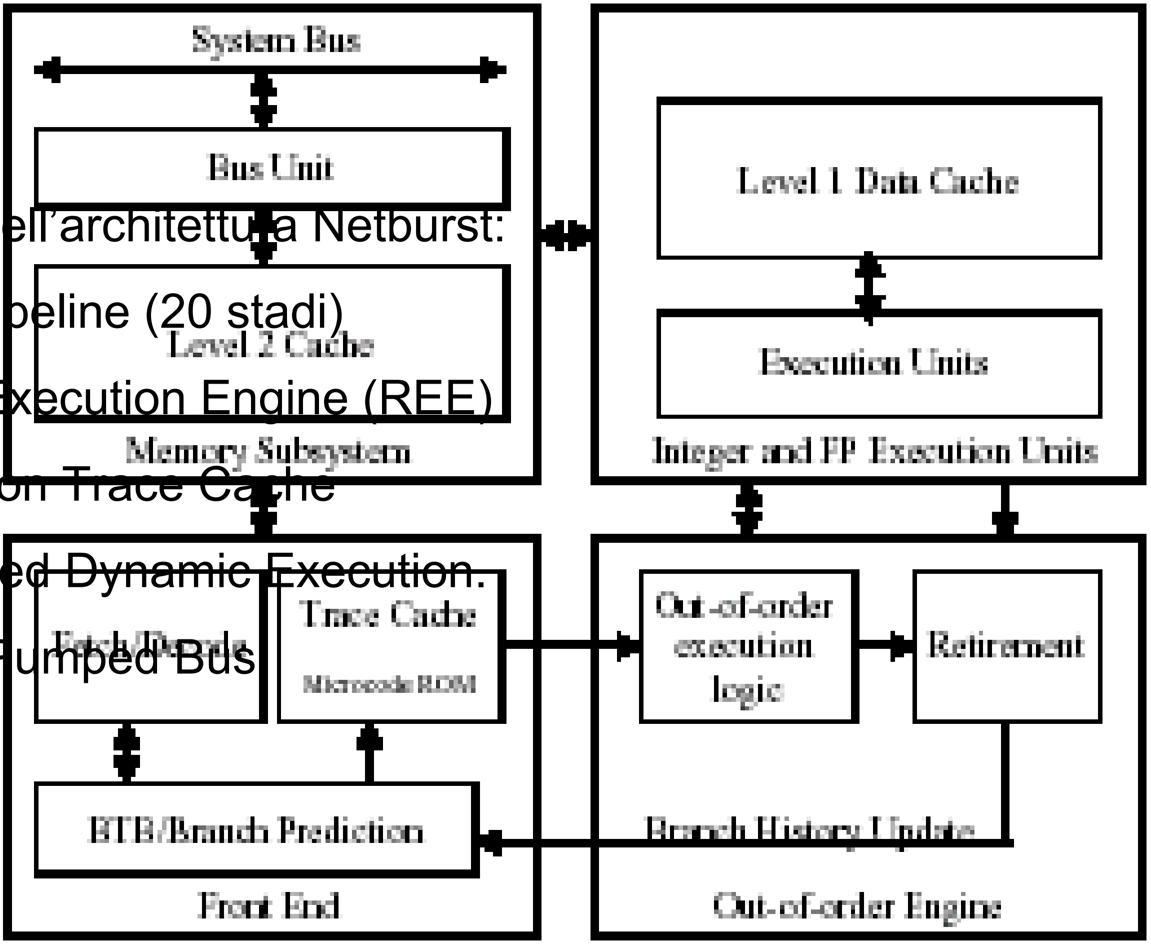

Arch State
Arch State
Processor Execution Resources
Processor Execution Resources
Arch State
Arch State
Arch State
Arch State
Processor Execution Resources
Processor Execution Resources

Figure 3: Processors with Hyper-Threading Technology

HT Resource Types

- Replicated Resources
  - Flags, Registers, Time-Stamp Counter, APIC
- Shared Resources
  - Memory, Range Registers, Data Bus
- Shared | Partitioned Resources
  - Caches & Queues

PIV Architecture



Il cuore dell'architettura Netburst:

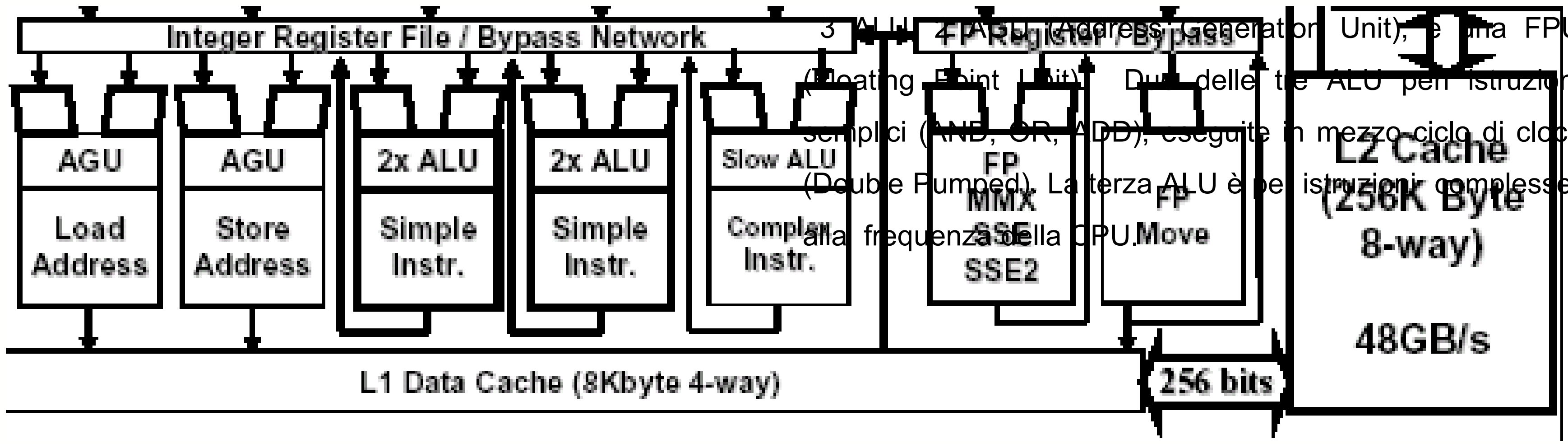
- Hyperpipeline (20 stadi)
- Rapid Execution Engine (REE)
- Execution Trace Cache
- Advanced Dynamic Execution.
- Quad Pumped Bus

HT Pipeline (I)

BUS QUAD PUMPED:

RAPID EXECUTION ENGINE:

3 ALU, 2 AGU (Address Generation Unit), e una FPU (Floating Point Unit). Due delle tre ALU per istruzioni semplici (AND, OR, ADD), eseguite in mezzo-ciclo di clock (Double Pumped). La terza ALU è per istruzioni complesse alla frequenza della CPU.



HT Pipeline (II)

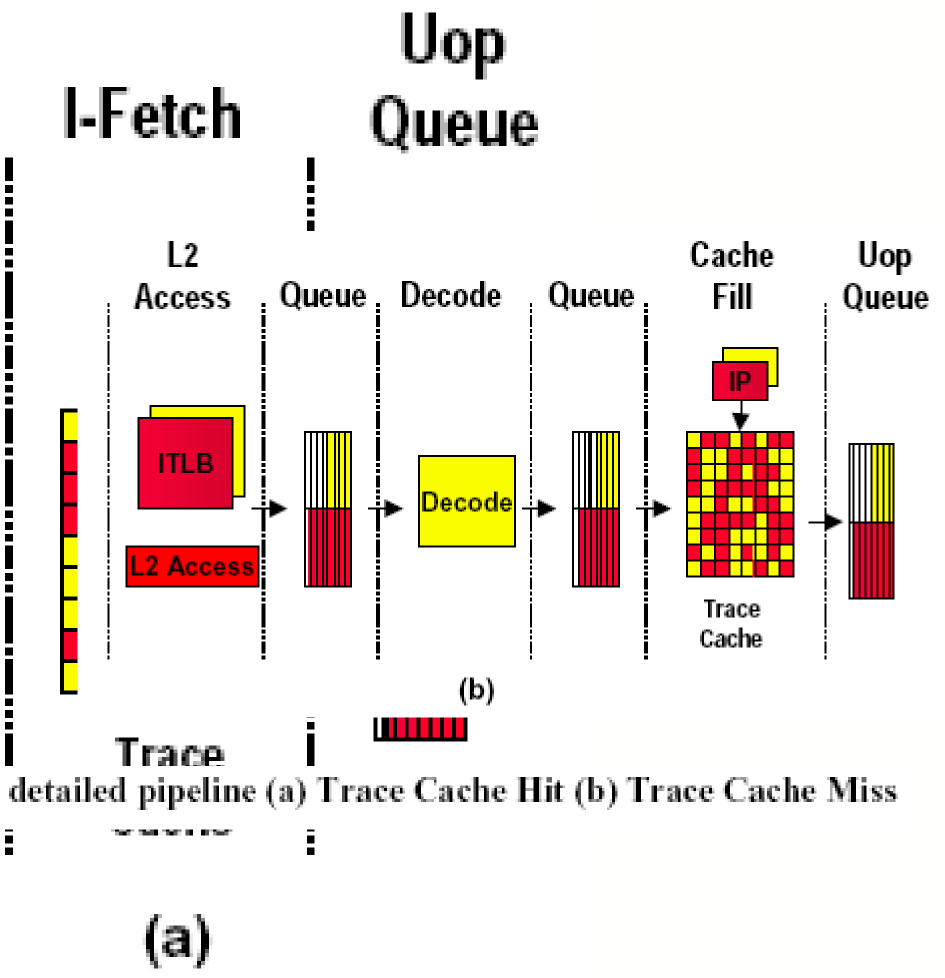


Figure 5: Front-end detailed pipeline (a) Trace Cache Hit (b) Trace Cache Miss

HT Pipeline (III)

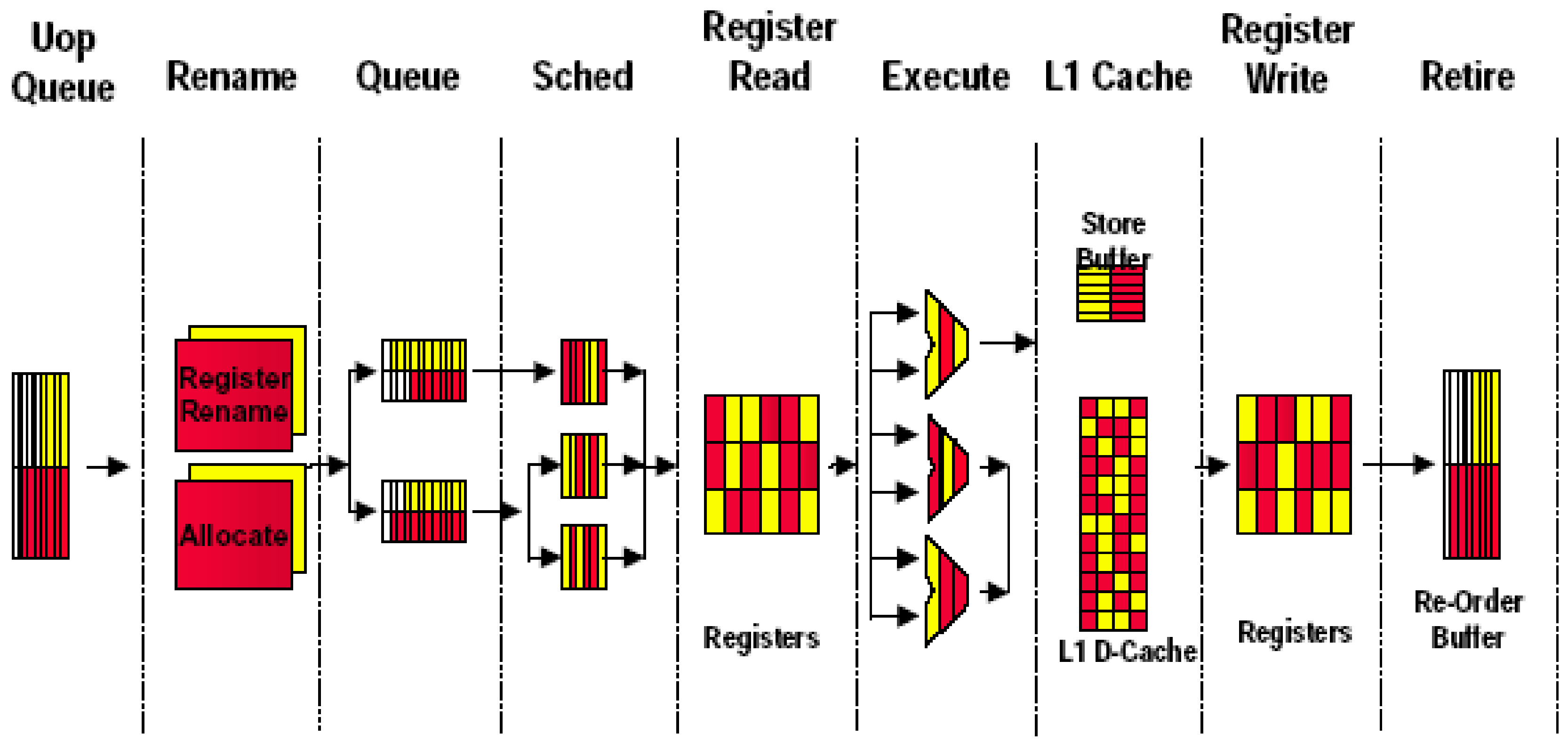
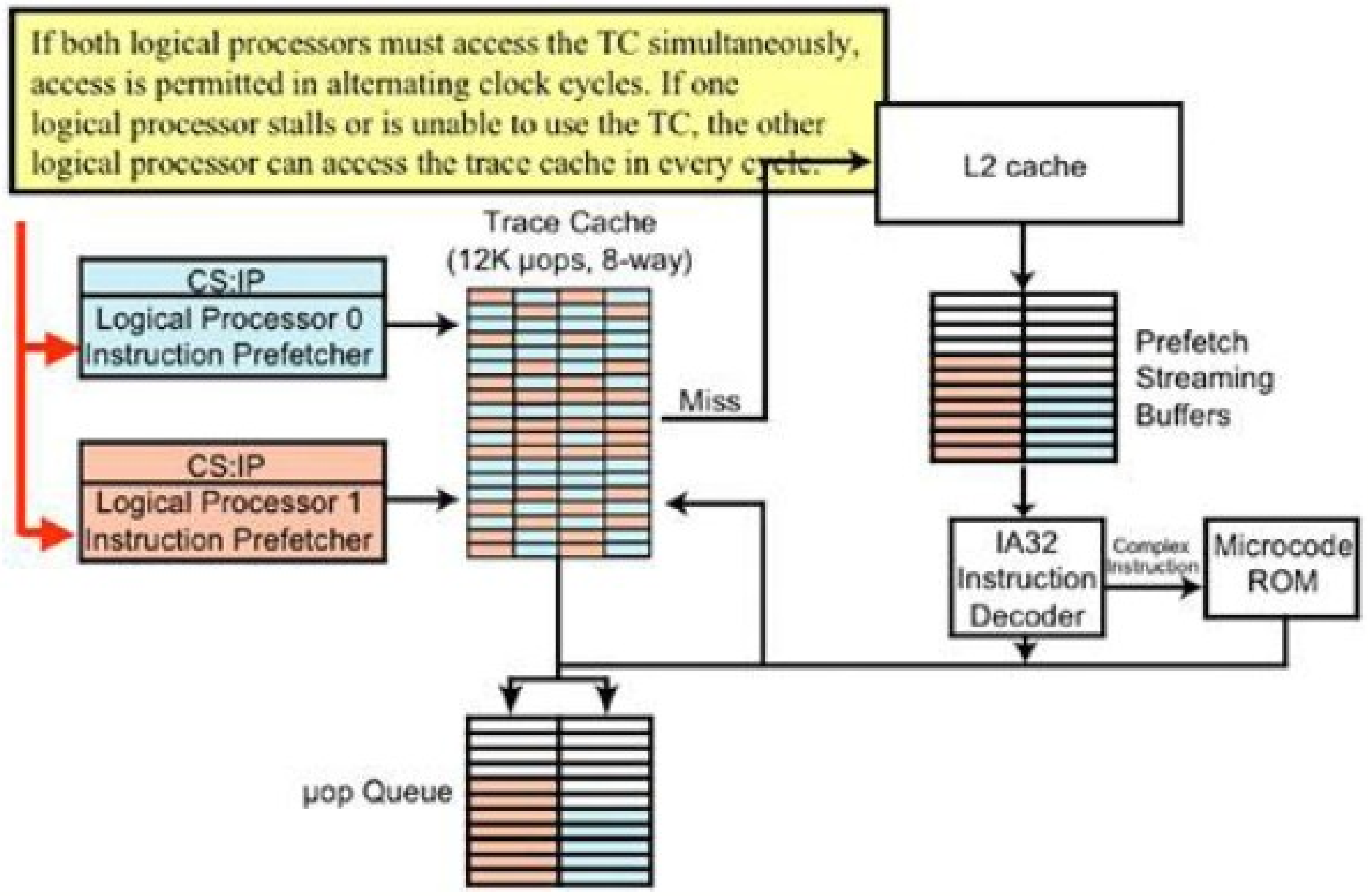


Figure 6: Out-of-order execution engine detailed pipeline

Execution Trace Cache (TC) (I)

- Stores decoded instructions called “micro-operations” or “uops”
- Arbitrate access to the TC using two IPs
  - If both PUs ask for access then switch will occur in the next cycle.
  - Otherwise, access will be taken by the available PU
  - Stalls (stem from misses) lead to switch
- Entries are tagged with the owner thread info
- 8-way set associative, Least Recently Used (LRU) algorithm
- Unbalanced usage between processors

Execution Trace Cache (TC) (I)

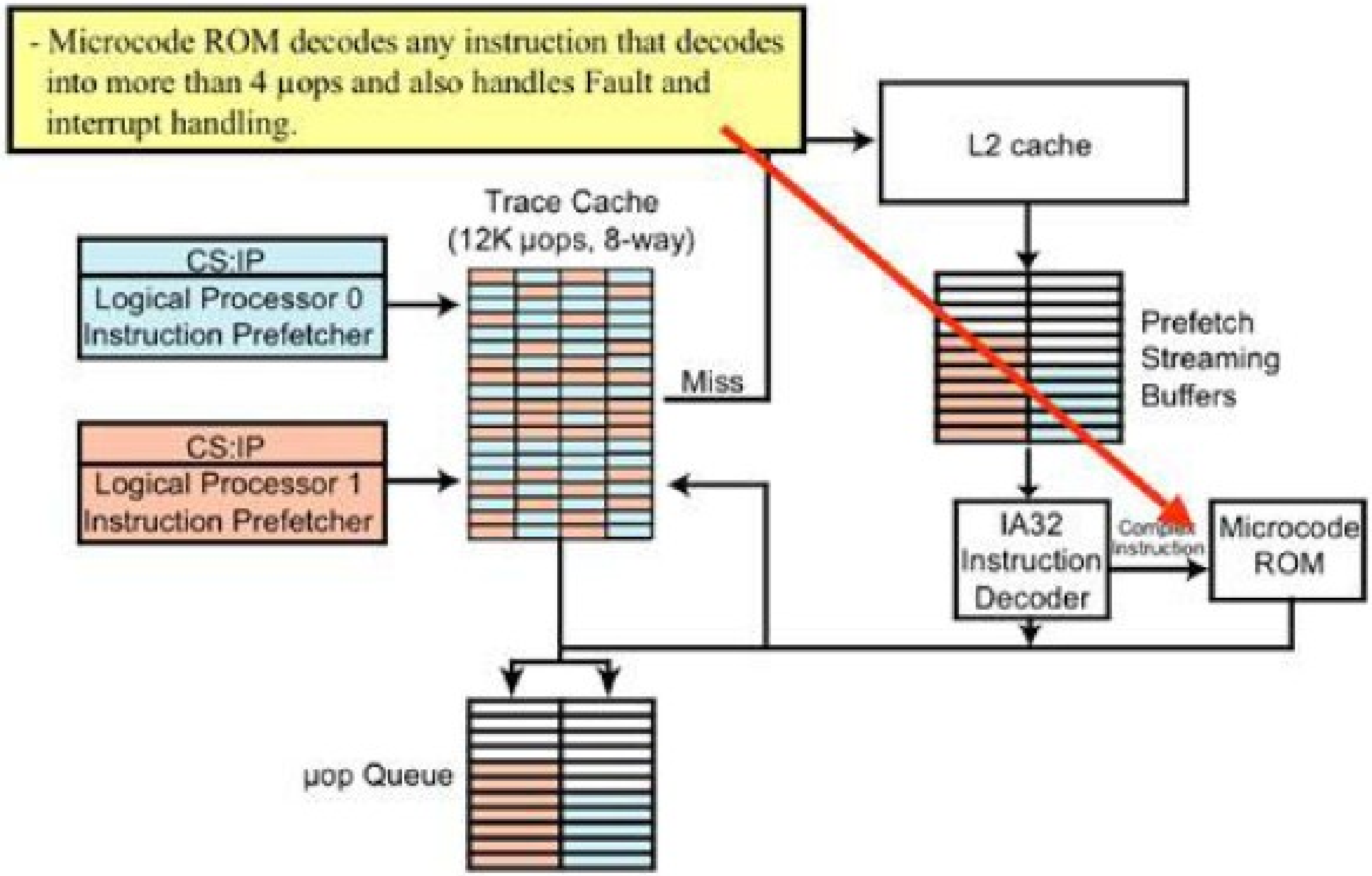


If both logical processors must access the TC simultaneously, access is permitted in alternating clock cycles. If one logical processor stalls or is unable to use the TC, the other logical processor can access the trace cache in every cycle.

Microcode Store ROM (MSROM) (I)

- Complex instructions (e.g. IA-32) are decoded into more than 4 uops
- Invoked by Trace Cache
- Shared by the logical processors
- Independent flow for each processor
- Access to MSROM alternates between logical processors as in the TC

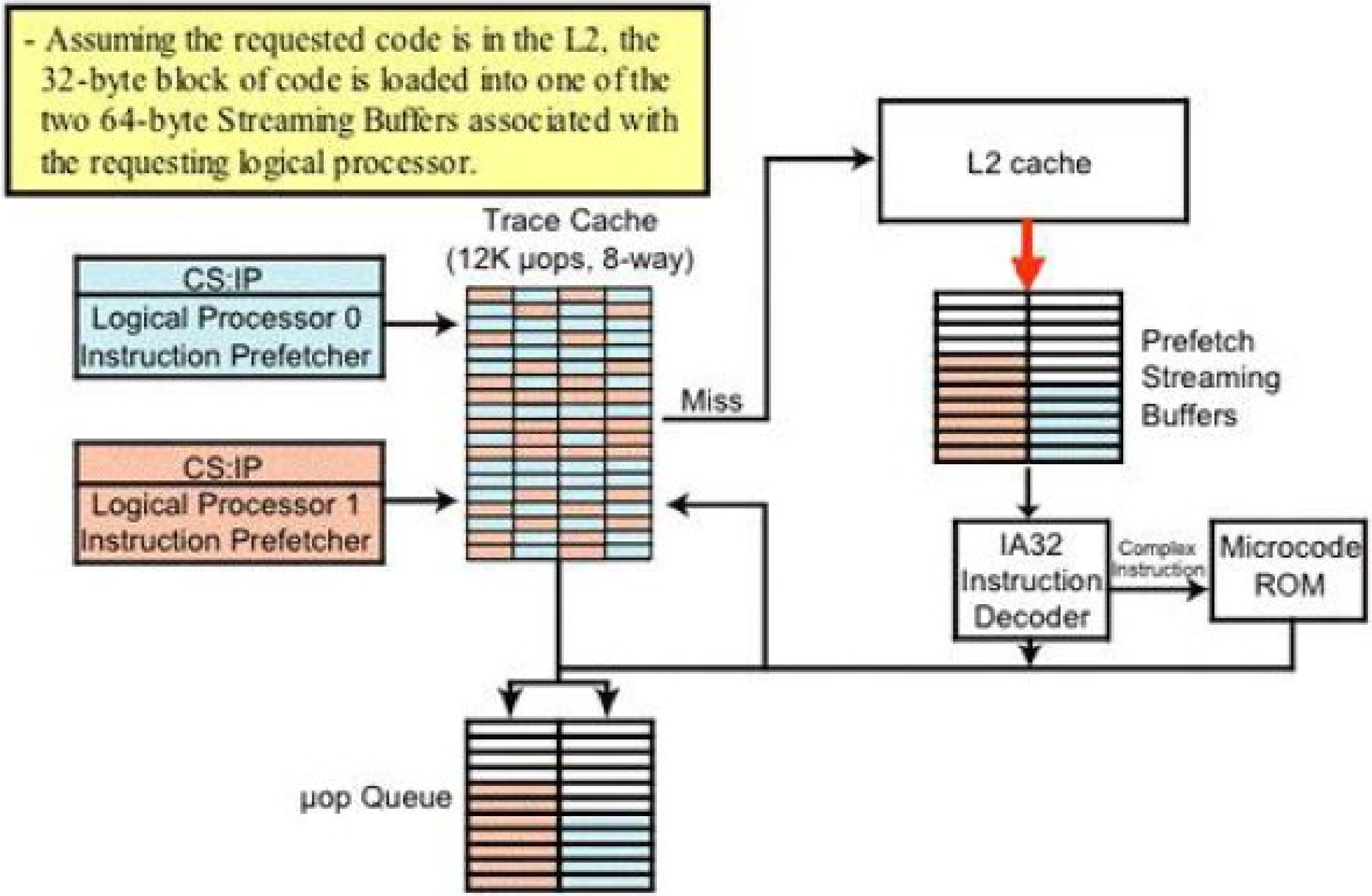
Microcode Store ROM (MSROM) (II)


- Microcode ROM decodes any instruction that decodes into more than 4 μops and also handles Fault and interrupt handling.
L2 cache
Trace Cache (12K μops, 8-way)
CS:IP
Logical Processor 0 Instruction Prefetcher
CS:IP
Logical Processor 1 Instruction Prefetcher
Miss
Prefetch Streaming Buffers
IA32 Instruction Decoder
Complex Instruction
Microcode ROM
μop Queue

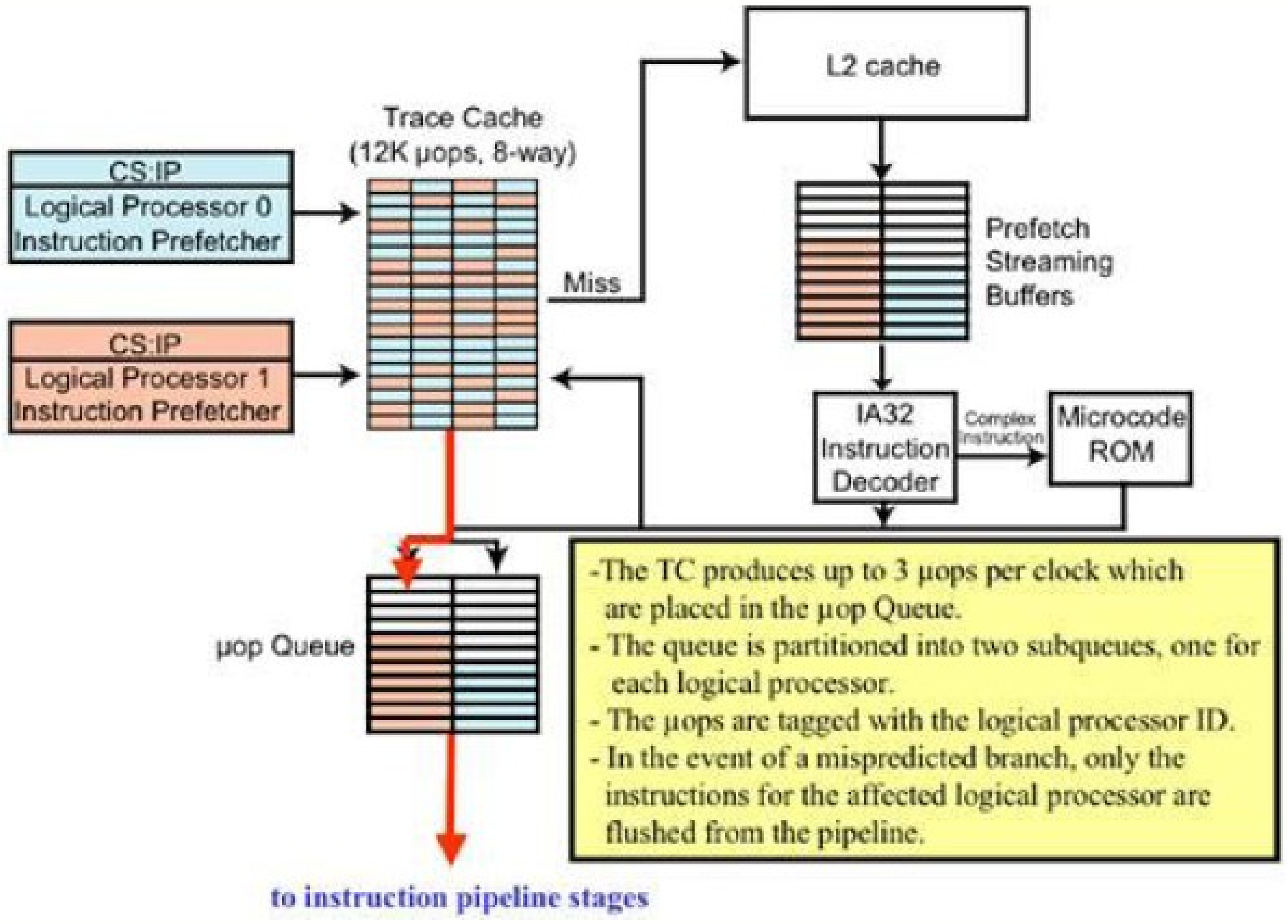
ITLB and Branch Prediction (I)

- If there is a TC miss, bytes need to be loaded from L2 cache and decoded into TC
- ITLB gets the “instruction deliver” request
- ITLB translates next Pointer address to the physical address
- ITLBs are duplicated for processors
- L2 cache arbitrates on first-come first-served basis while always reserve at least one slot for each processor
- Branch prediction structures are either duplicated or shared
  - If shared owner tags should be included

ITLB and Branch Prediction (II)


- Assuming the requested code is in the L2, the 32-byte block of code is loaded into one of the two 64-byte Streaming Buffers associated with the requesting logical processor.

Uop Queue


L2 cache
Trace Cache (12K µops, 8-way)
CS:IP
Logical Processor 0 Instruction Prefetcher
CS:IP
Logical Processor 1 Instruction Prefetcher
Miss
Prefetch Streaming Buffers
IA32 Instruction Decoder
Complex instruction
Microcode ROM
µop Queue
to instruction pipeline stages

-The TC produces up to 3 µops per clock which are placed in the µop Queue.
- The queue is partitioned into two subqueues, one for each logical processor.
- The µops are tagged with the logical processor ID.
- In the event of a mispredicted branch, only the instructions for the affected logical processor are flushed from the pipeline.

HT Pipeline (III) -- Revisited

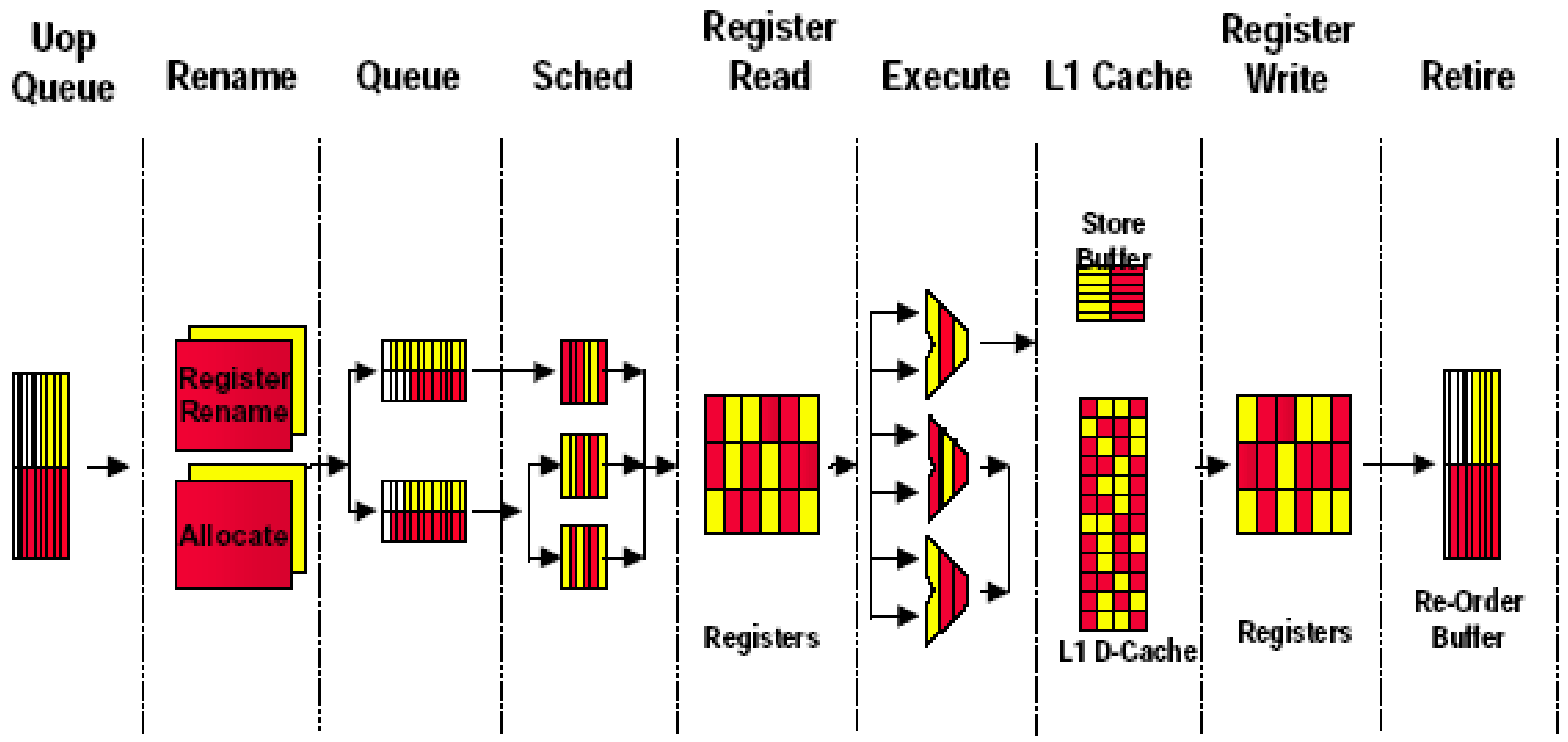


Figure 6: Out-of-order execution engine detailed pipeline

Allocator

- Allocates many of the key machine buffers;
  - 126 re-order buffer entries
  - 128 integer and floating-point registers
  - 48 load, 24 store buffer entries
- Resources shared equal between processors
- Limitation of the key resource usage, we enforce fairness and prevent deadlocks over the Arch.
- For every clock cycle, allocator switches between uop queues
- If there is stall or HALT, there is no need to alternate between processors

Register Rename

- Involves with mapping shared registers names for each processor
- Each processor has its own Register Alias Table (RAT)
- Uops are stored in two different queues;
  - Memory Instruction Queue (Load/Store)
  - General Instruction Queue (Rest)
- Queues are partitioned among PUs

Instruction Scheduling

- Schedulers are at the heart of the out-of-order execution engine
- There are five schedulers which have queues of size 8-12
- Scheduler is oblivious when getting and dispatching uops
  - It ignores the owner of the uops
  - It only considers if input is ready or not
  - It can get uops from different PUs at the same time
  - To provide fairness and prevent deadlock, some entries are always assigned to specific PUs

Execution Units & Retirement

- Execution Units are oblivious when getting and executing uops
    - Since resource and destination registers were renamed earlier, during/after the execution it is enough to access physical registries
- After execution, the uops are placed in the re-order buffer which decouples the execution stage from retirement stage
- The re-order buffer is partitioned between PUs
- Uop retirement commits the architecture state in program order
    - Once stores have retired, the store data needs to be written into L1 data-cache, immediately

Memory Subsystem

- Totally oblivious to logical processors
  - Schedulers can send load or store uops without regard to PUs and memory subsystem handles them as they come
- Memory types;
  - DTLB:
    - Translates addresses to physical addresses
    - 64 fully associative entries; each entry can map either 4K or 4MB page
    - Shared between PUs (Tagged with ID)
  - L1, L2 and L3 caches
    - Cache conflict might degrade performance
    - Using same data might increase performance (more mem. hits)

System Modes

- Two modes of operation;
  - single-task (ST)
    - When there is one SW thread to execute
  - multi-task (MT)
    - When there are more than one SW threads to execute
    - ST0 or ST1 where number shows the active PU
    - HALT command was introduced where resources are combined after the call
  - Reason is to have better utilization of resources

Performance

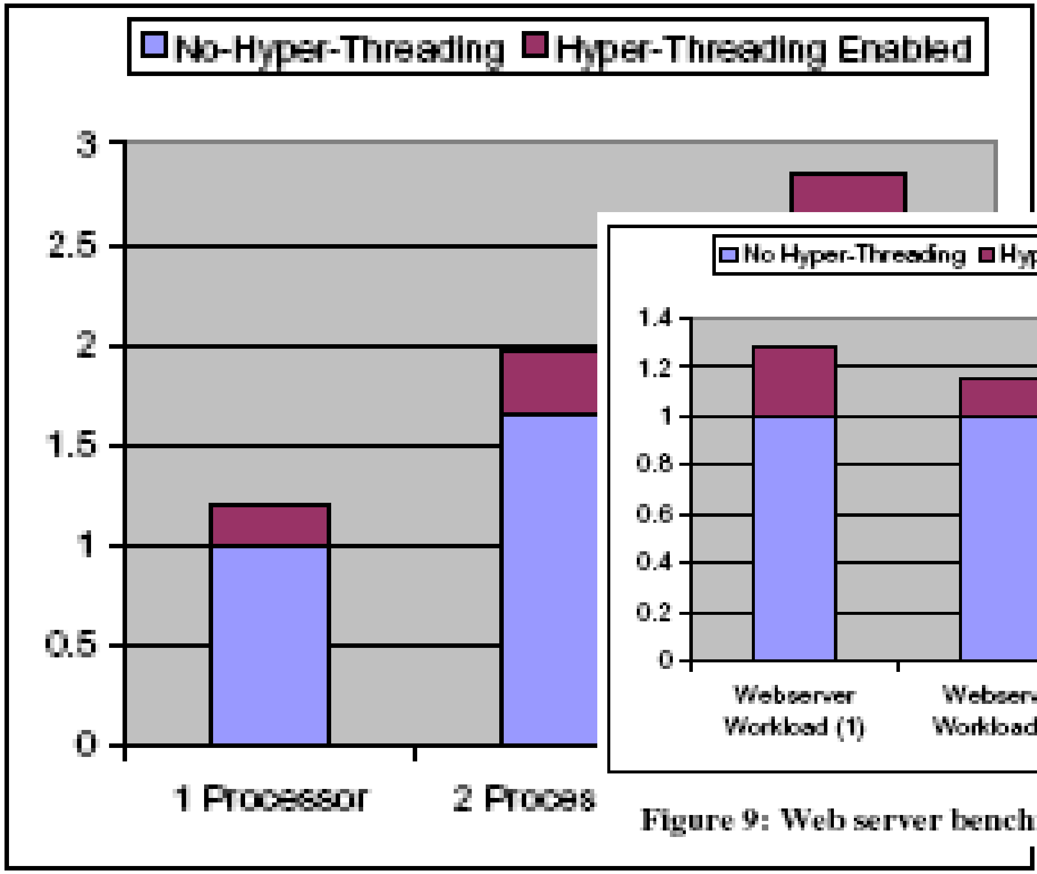


Figure 8: Performance increases from Hyper-Threading Technology on an OLTP workload

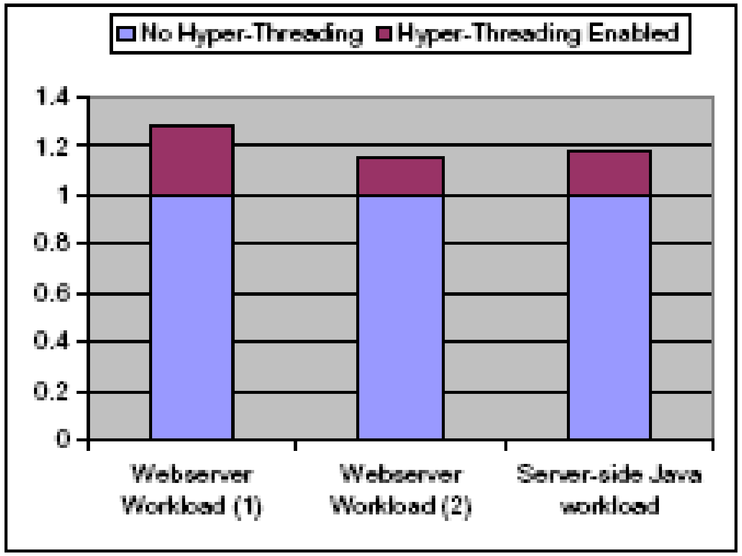


Figure 9: Web server benchmark performance

OS Support for HT

- Native HT Support
  - Windows XP Pro Edition
  - Windows XP Home Edition
  - Linux v 2.4.x (and higher)
- Compatible with HT
  - Windows 2000 (all versions)
  - Windows NT 4.0 (limited driver support)
- No HT Support
  - Windows ME
  - Windows 98 (and previous versions)

Conclusion

- Measured performance (Xeon) showed performance gains of up to 30% on common server applications.
- HT is expected to be viable and market standard from Mobile to server processes.

Questions ?