2008
YEAR

PRESENTATION

The University of Auckland | New Zealand

Computer Science 703

Advance Computer Architecture

2008 Semester 1

Lecture Notes

6Mar08

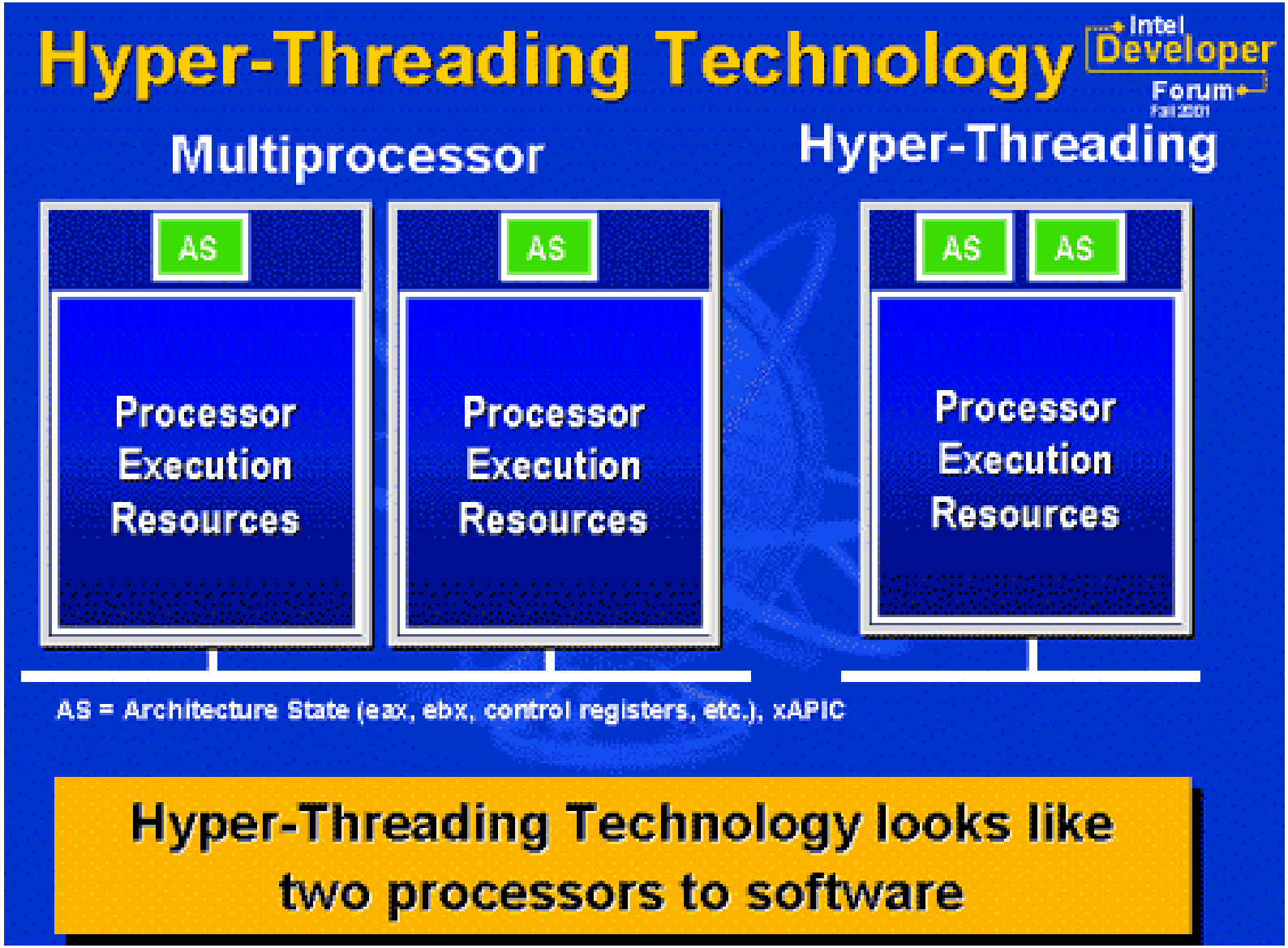
Multiprocessing: Multithreading & Multi-cores

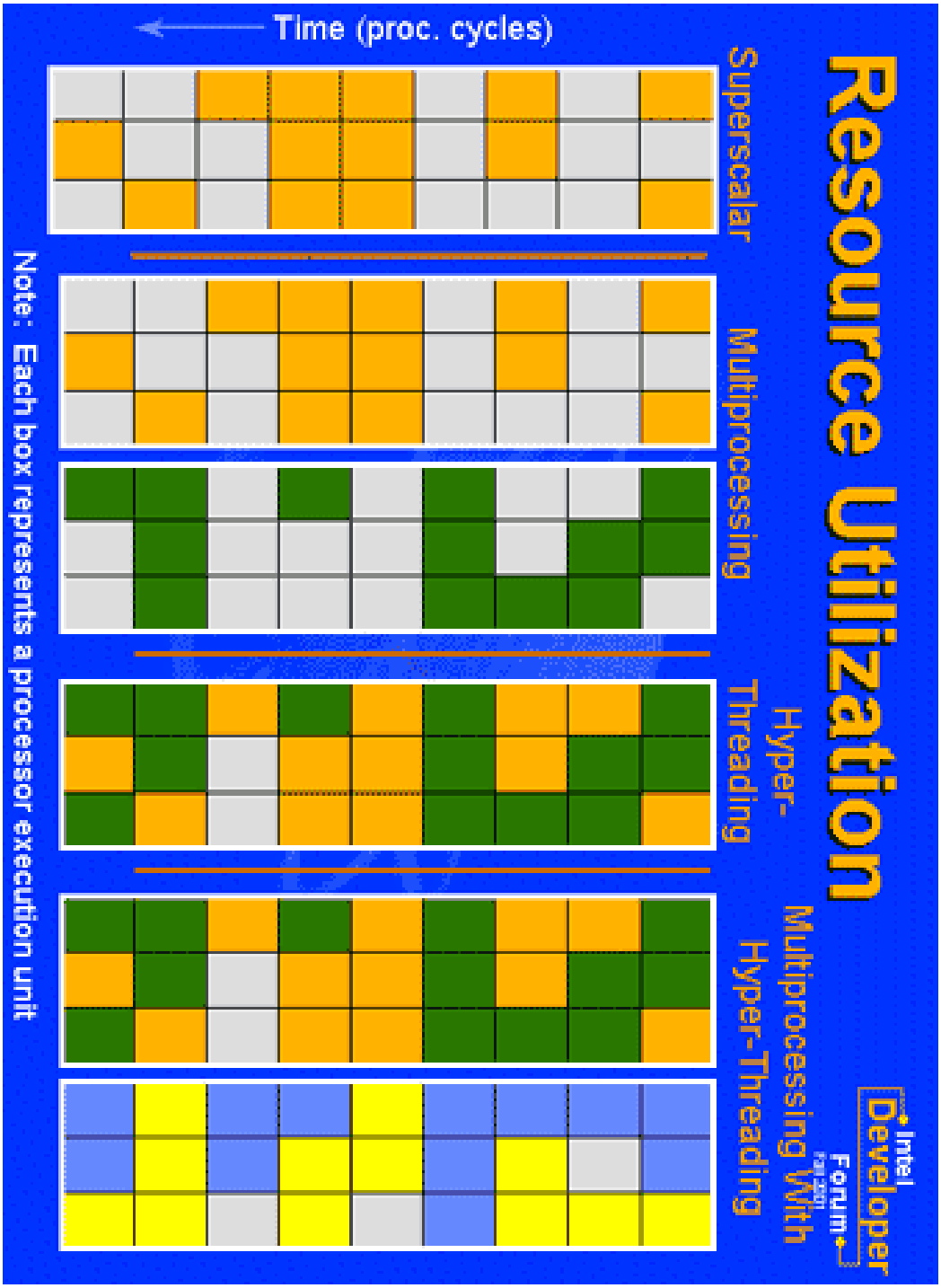
James Goodman

Multiprocessors, Multi-Cores, Multi-threading, and Hyperthreading

Terminology

- Multiprocessors: multiple processors sharing a common memory (SMP, tightly-coupled MP)
- Multi-cores: multiple processors sharing a common silicon die and memory system (CMP)
- Multithreading: a single processor capable of maintaining the state of multiple threads or processes while executing
- Hyperthreading: Intel's term for a certain type of multithreading (SMT)
- Chip Multithreading (CMT): a multi-core die with multithreaded processors


Hyper-Threading Technology
Intel Developer Forum
Multiprocessor
Hyper-Threading
AS
AS
AS
AS
Processor Execution Resources
Processor Execution Resources
Processor Execution Resources
AS = Architecture State (eax, ebx, control registers, etc.), xAPIC
Hyper-Threading Technology looks like two processors to software


Resource Utilization
Intel Developer Forum
Time (proc. cycles)
Superscalar
Multiprocessing
Hyper-Threading
Multiprocessing With Hyper-Threading
Note: Each box represents a processor execution unit

Why Multi-threading

- Resources can be used more effectively
  - Up to 30% more throughput from two threads (Intel)
  - About 5% additional die area for second thread
- Threads can actively share memory data very efficiently

Memory System

- Threads on same core share all memory except registers
- Multi-cores often share L2 cache, not L1

Variations of Multithreading

- Fine-grained multithreading switches between threads on each instruction, causing the execution of multiple threads to be interleaved. This interleaving is often done in a round-robin fashion, skipping any threads that are stalled at that time.

- Coarse-grained multithreading switches threads only on costly stalls, such as level-2 cache misses.

- Simultaneous multithreading (SMT) is a variation on multithreading that uses the resources of a multiple-issue, dynamically scheduled processor to exploit thread-level parallelism (TLP) at the same time it exploits instruction-level parallelism (ILP)

Hennessey &Patterson, pp. 608-609.

Comparison of Multi-threading and True Multiprocessing

- Multi-threading is limited to exploiting wasted resources
- Multi-threading can have faster communication through memory
- Multi-threading can share code in L1 cache (but may also require more cache if code is not shared)

Interesting Multithreading Trade-off

- Multiple threads implies greater tolerance for cache misses

but…

- Multiple threads implies multiple contexts
- Multiple contexts implies larger memory requirements

Multithreading makes sense if throughput is important!