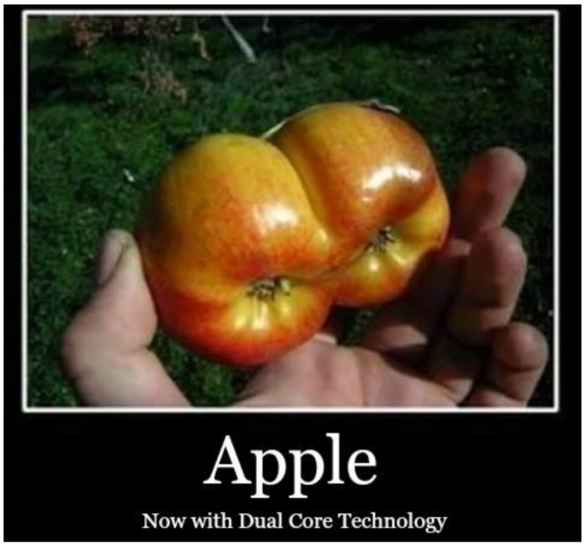
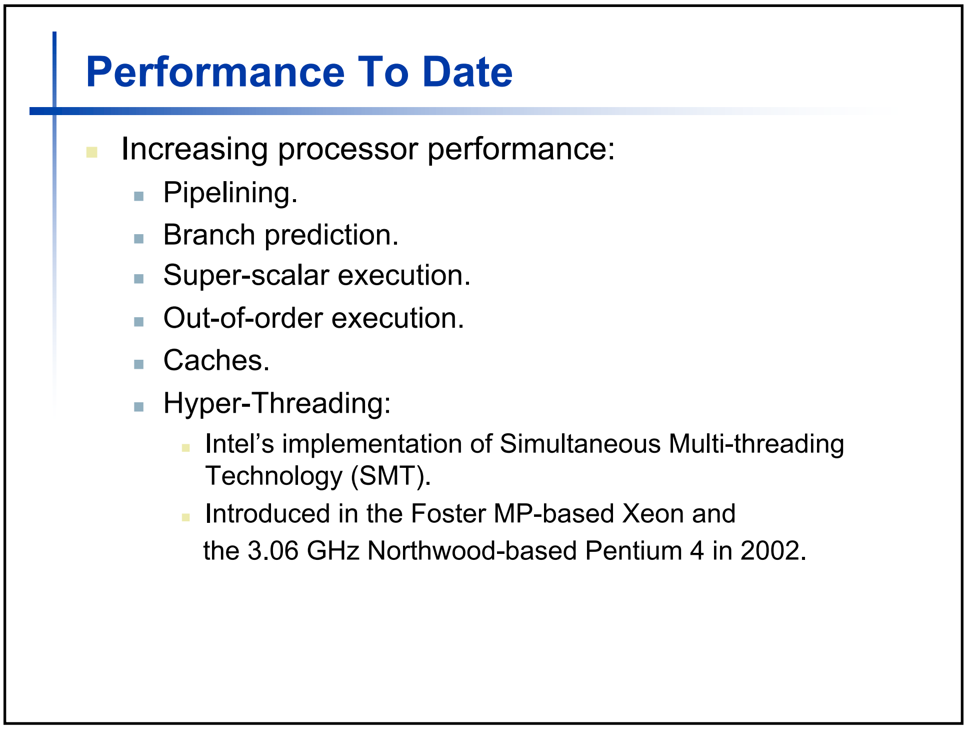
COMPUTER ORGANIZATION AND DESIGN
The Hardware/Software Interface

Multi-Threading

Hyper-, Multi-, and Simultaneous Thread Execution

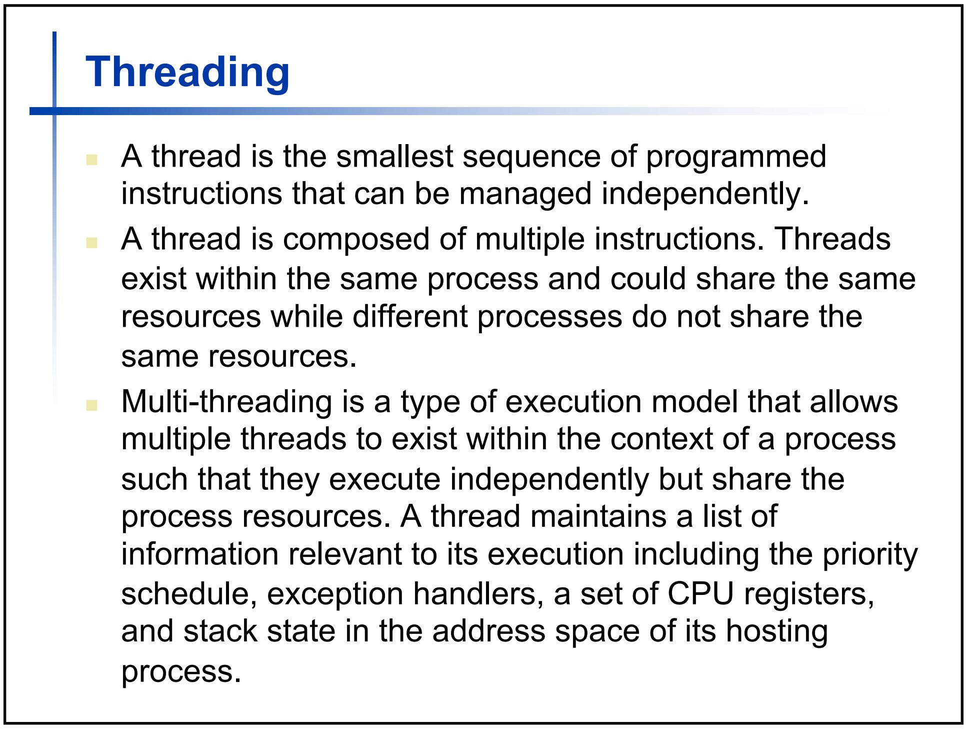
HaHaHaHa





Performance To Date

- Increasing processor performance:
  - Pipelining.
  - Branch prediction.
  - Super-scalar execution.
  - Out-of-order execution.
  - Caches.
  - Hyper-Threading:
    - Intel’s implementation of Simultaneous Multi-threading Technology (SMT).
    - Introduced in the Foster MP-based Xeon and the 3.06 GHz Northwood-based Pentium 4 in 2002.



Threading

- A thread is the smallest sequence of programmed instructions that can be managed independently.
- A thread is composed of multiple instructions. Threads exist within the same process and could share the same resources while different processes do not share the same resources.
- Multi-threading is a type of execution model that allows multiple threads to exist within the context of a process such that they execute independently but share the process resources. A thread maintains a list of information relevant to its execution including the priority schedule, exception handlers, a set of CPU registers, and stack state in the address space of its hosting process.

Multi-threading on A Chip

- Multi-threading helps find a way to “hide” true data dependency stalls, cache miss stalls, and branch stalls by finding instructions (from other process threads) that are **independent** of these stalling instructions.
- Hardware multi-threading – increase the utilization of resources on a chip by allowing multiple processes (threads) to share the functional units of a single processor:
  - Processor must duplicate the state hardware for each thread – a separate register file, PC, instruction buffer, and store buffer for each thread.
  - The caches, TLBs, BHT, BTB, can be shared.
  - The memory can be shared through virtual memory mechanisms.

Overview

- **Multi-threading:**
  - Uses processors resources in a highly efficient way.
  - Consists of two logical processors for every physical core.
  - Separate threads can be executed on each logical processor simultaneously.
  - Here is a lame video
- Multi-threading is enhanced by Multi-core technology which allows the parallel execution of software threads across multiple processor cores.

Simultaneous Multi-threading (SMT)

- A variation on multi-threading that uses the resources of a multiple-issue, dynamically scheduled processor to exploit both program Instruction-level-parallelism (ILP) and thread-level parallelism (TLP):
  - Most superscaler processors have more machine level parallelism than most programs can effectively use (i.e., than have ILP).
  - With register renaming and dynamic scheduling, multiple instructions from independent threads can be issued without regard to dependencies among them:
    - Need separate rename tables for each thread or need to be able to indicate which thread the entry belongs to.
    - Need the capability to commit from multiple threads in one cycle.
- Intel’s Pentium 4 SMT is called hyper-threading:
  - Supports just two threads - doubles the architecture state.

Advantages and Disadvantages

Advantages of Hyper-threading:

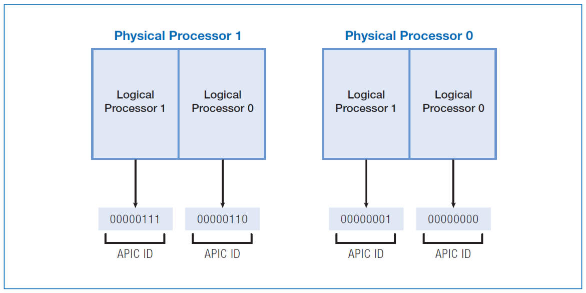
- Performance boost by as much as 30% when compared to an identical processing core without hyper-threading.

Disadvantages of Hyper-threading:

- Possible increase in core size of about 5 percent caused by the duplication of certain sections of the CPU core (Intel's claim).
- Overall power consumption is higher.
- Increases cache thrashing (the repeated displacing and loading of cache blocks), which ARM claims could be as much as 42%.

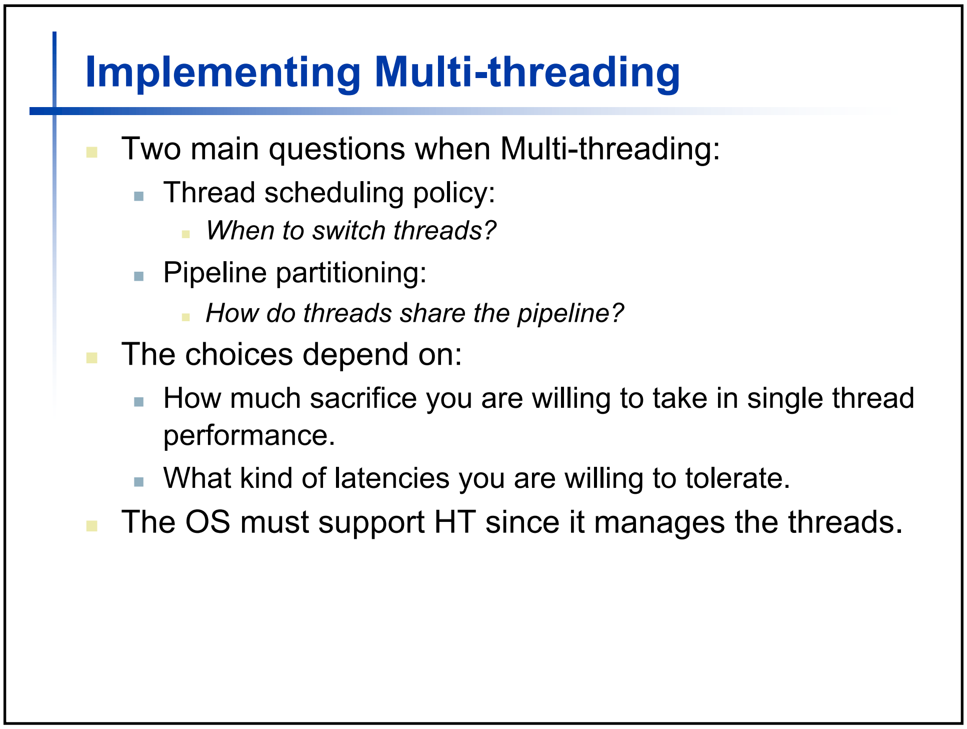
Hyper-Threading Architecture

- Makes a single physical processor appear as multiple logical processors.
- Each logical processor has a copy of architecture state.
- Logical processors share a single set of physical execution resources.
- From an architecture perspective, we have to worry about the logical processors using shared resources:
  - Caches, execution units, branch predictors, control logic, and buses.



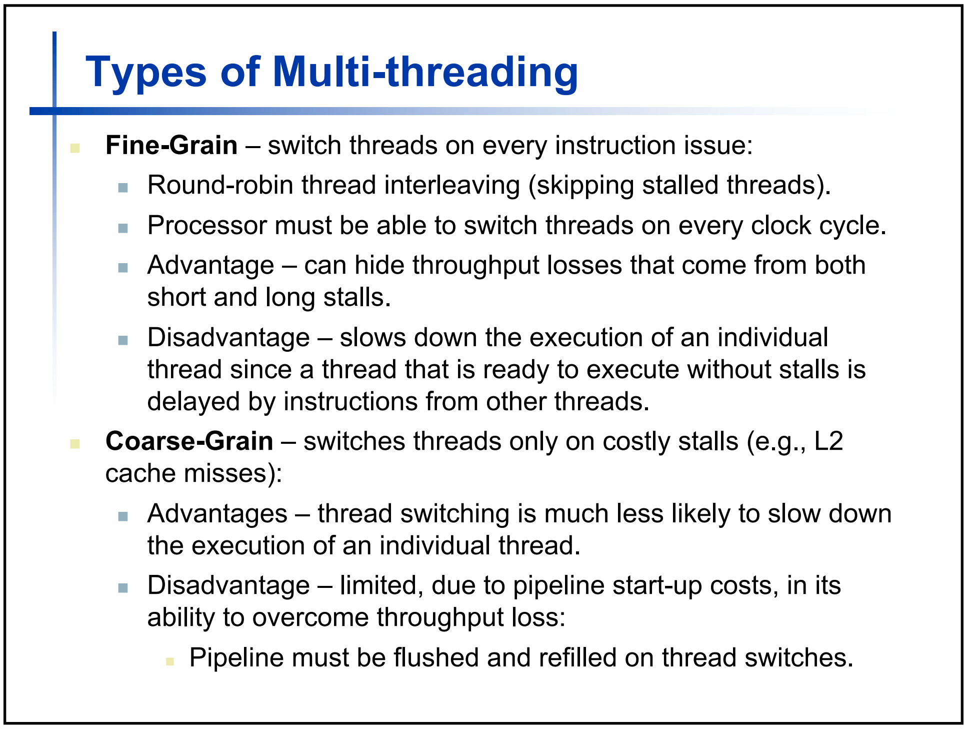
HT vs. Dual Processors - Recap

- Dual Processor:
  - System resources are duplicated.
- Hyper-Threading:
  - Architectural state is duplicated:
    - Data, segment, control, and debug registers.
  - Resources shared by the logical CPU’s:
    - Execution engine, caches, system-bus interface and firmware.



Implementing Multi-threading

- Two main questions when Multi-threading:
  - Thread scheduling policy:
    - *When to switch threads?*
  - Pipeline partitioning:
    - *How do threads share the pipeline?*
- The choices depend on:
  - How much sacrifice you are willing to take in single thread performance.
  - What kind of latencies you are willing to tolerate.
- The OS must support HT since it manages the threads.

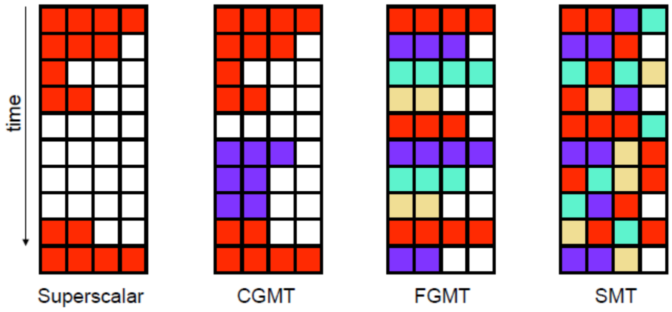


Types of Multi-threading

- **Fine-Grain** – switch threads on every instruction issue:
  - Round-robin thread interleaving (skipping stalled threads).
  - Processor must be able to switch threads on every clock cycle.
  - Advantage – can hide throughput losses that come from both short and long stalls.
  - Disadvantage – slows down the execution of an individual thread since a thread that is ready to execute without stalls is delayed by instructions from other threads.
- **Coarse-Grain** – switches threads only on costly stalls (e.g., L2 cache misses):
  - Advantages – thread switching is much less likely to slow down the execution of an individual thread.
  - Disadvantage – limited, due to pipeline start-up costs, in its ability to overcome throughput loss:
    - Pipeline must be flushed and refilled on thread switches.

Types of Multithreading

- Coarse-grain multithreading (CGMT).
- Fine-grain multithreading (FGMT).
- Simultaneous multithreading (SMT).



Fine-Grain (FGMT)

- Significant sacrifice to single thread performance.
- Tolerates latencies (L2 misses, mis-predicted branches).
- Thread scheduling:
  - Round-robin thread switching after each cycle.
- Pipeline partitioning:
  - No flushing, dynamic.
  - Multiple threads in pipeline at once.
- Lots of threads needed.
- Finding a home in GPU’s.

Software Implementation

- From a software aspect - synchronization of objects is often required when implementing software apps with multi-threading.
- These objects are used to protect memory from being modified by multiple threads at the same time.
- A mutex is one such object. It is a lock which can be locked by a thread, and any successive attempt to lock it by another thread or by the same thread, will be blocked until the mutex is unlocked, thus keeping an item from being accessed more than once at the same time.

Conclusions

- Hardware support for multi-, hyper-, and simultaneous threading exists in all major processors.
- Operating system must support multi-threading.
- Types of multi-threading:
  - Coarse-grain.
  - Fine-grain.
  - Simultaneous.
- Individual latencies are sacrificed for the good of the entire program (or process).