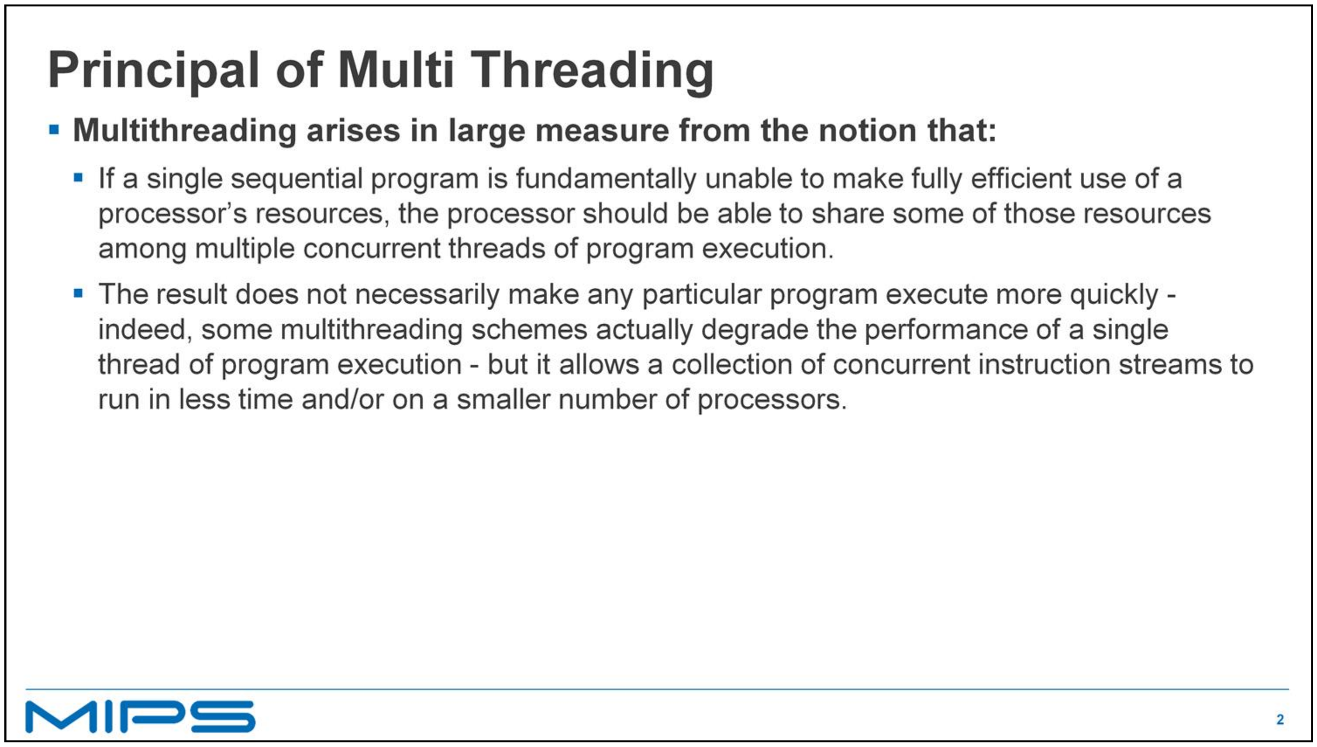
MIPS

MIPS MT Training

Fine Grain Multi Threading

www.mips.com

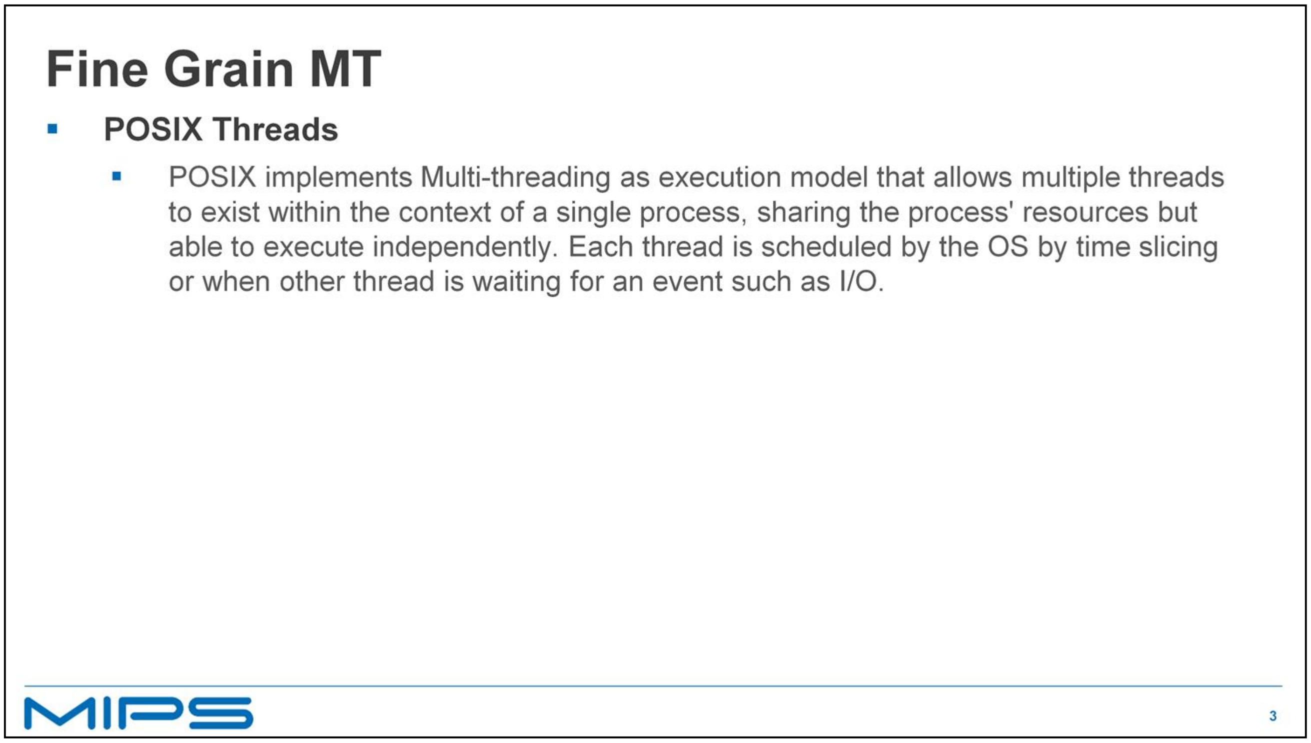
This section covers fine-grained multithreading I will describe what it is as it pertains to the MIPS multithreading architecture.



Principal of Multi Threading

- **Multithreading arises in large measure from the notion that:**
  - If a single sequential program is fundamentally unable to make fully efficient use of a processor's resources, the processor should be able to share some of those resources among multiple concurrent threads of program execution.
  - The result does not necessarily make any particular program execute more quickly - indeed, some multithreading schemes actually degrade the performance of a single thread of program execution - but it allows a collection of concurrent instruction streams to run in less time and/or on a smaller number of processors.

MIPS



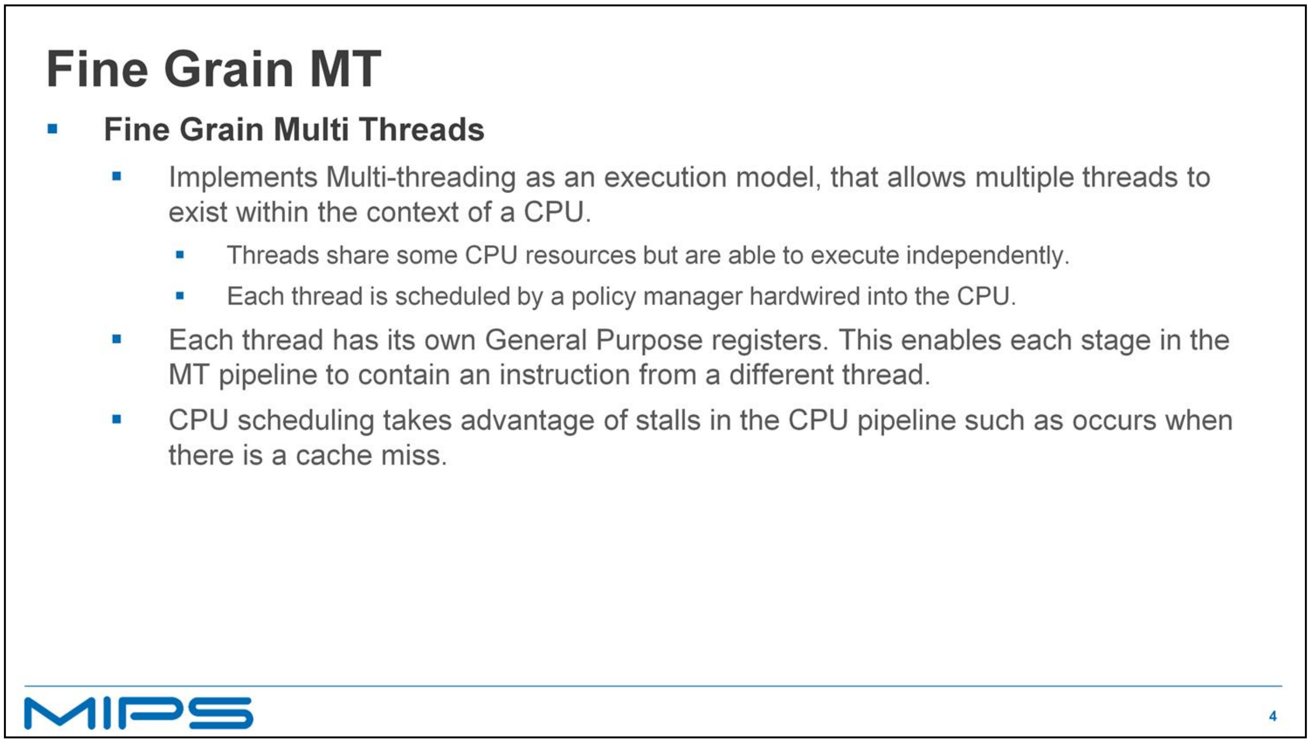
Fine Grain MT

- **POSIX Threads**
  - POSIX implements Multi-threading as execution model that allows multiple threads to exist within the context of a single process, sharing the process' resources but able to execute independently. Each thread is scheduled by the OS by time slicing or when other thread is waiting for an event such as I/O.

3

First we'll talk about POSIX threads so we can get an idea what threading is then I'll go into fine-grained multithreading.

POSIX implements threads as a multithreaded execution model that allows multiple threads to exist in the context of a single process sharing the processes resources but able to execute independently each thread is scheduled by the OS by time slicing or when another thread is waiting for an event such as I/0.



Fine Grain MT

- **Fine Grain Multi Threads**
  - Implements Multi-threading as an execution model, that allows multiple threads to exist within the context of a CPU.
    - Threads share some CPU resources but are able to execute independently.
    - Each thread is scheduled by a policy manager hardwired into the CPU.
  - Each thread has its own General Purpose registers. This enables each stage in the MT pipeline to contain an instruction from a different thread.
  - CPU scheduling takes advantage of stalls in the CPU pipeline such as occurs when there is a cache miss.

MIPS

4

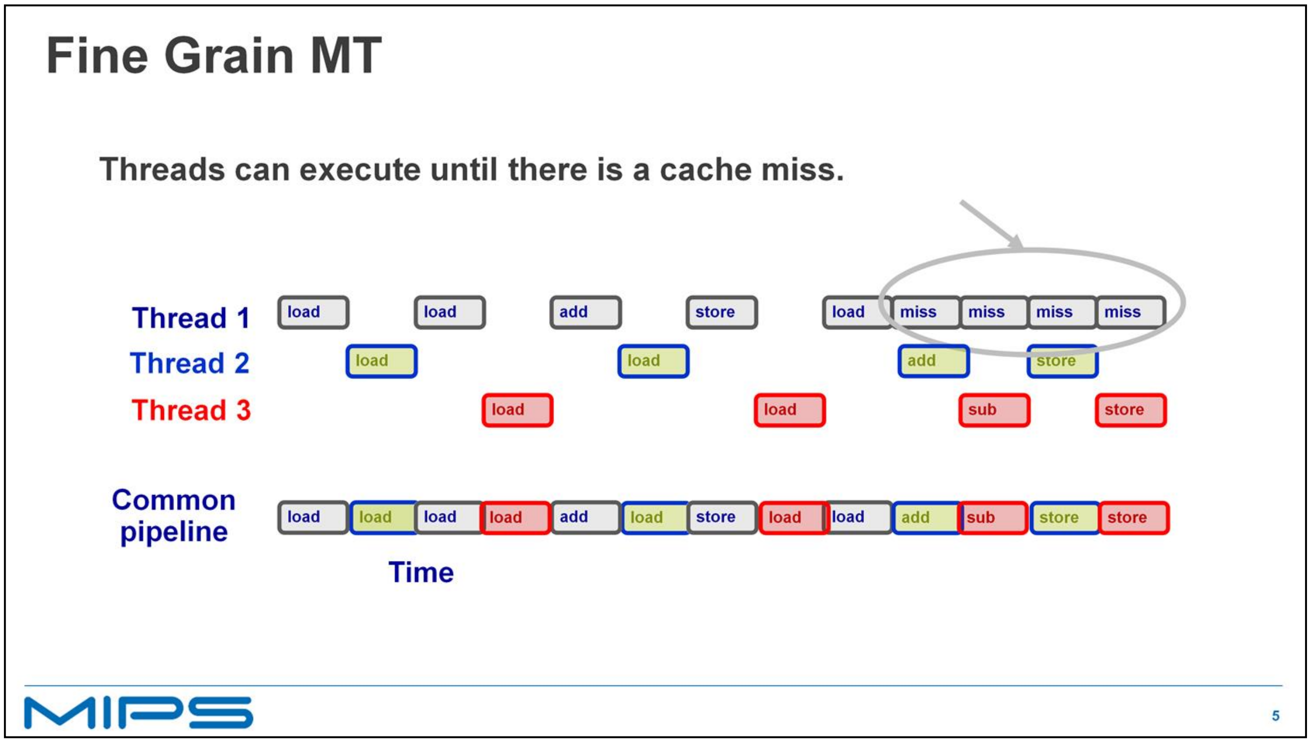
Fine-grained multithreading has executed by the MIPS multithreading architecture implements multithreading as an execution model that allows multiple threads to exist within the context of one CPU

+ Threads share some CPU resources but are able to execute independently

+ Each thread is scheduled by the policy manager this is a way of controlling the priority of each thread.

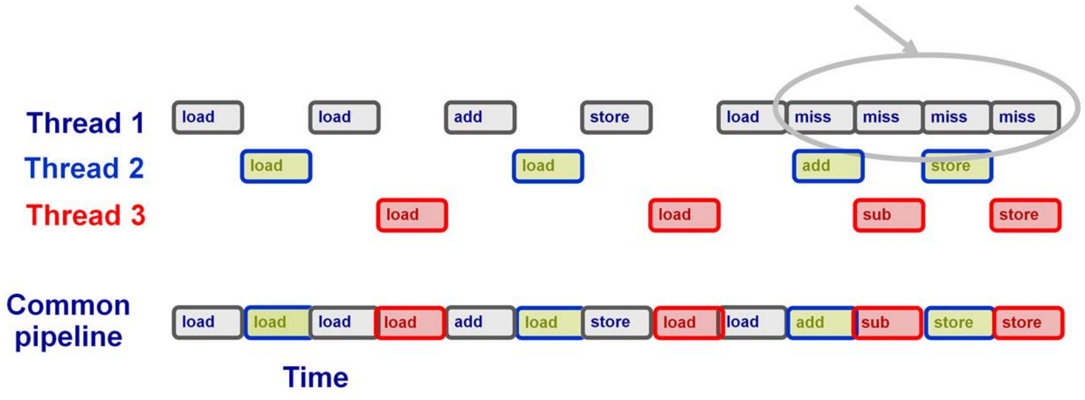
+ Each thread has its own set of general purpose registers. This enables each stage of a multithreaded pipeline to contain instructions from different threads So that execution of those instructions a effects only the registers of the thread the instruction is from.

+ The CPU scheduling through the policy manager takes advantage of stalls in the CPU pipeline such as when the risk cache miss.



Fine Grain MT

Threads can execute until there is a cache miss.



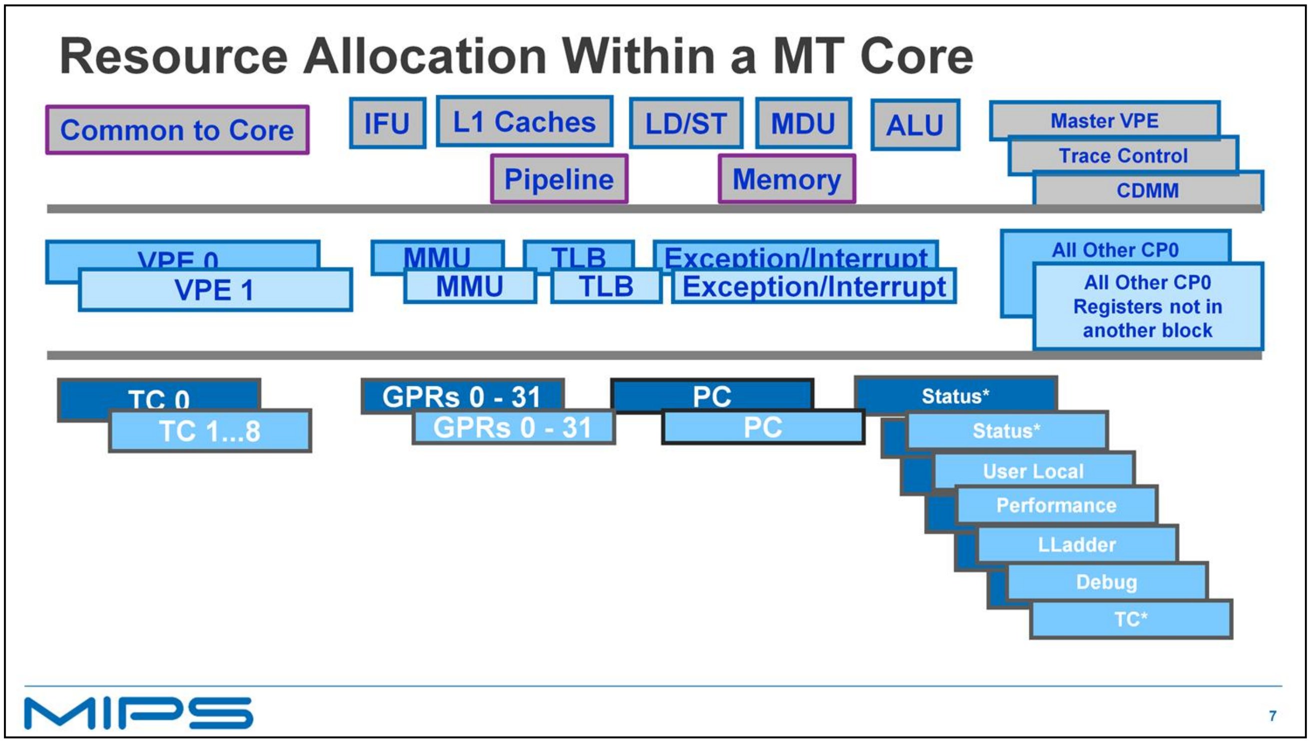
MIPS 5

Here is an illustration of 3 threads sharing the Cores pipeline. Each thread has its own instruction stream. Instructions are dispatched into to the pipe line in a round robin manner until thread 1 has a cache miss. Then instructions from just thread 2 and 3 are dispatched while thread 1s cache miss is handled.

+ Threads to execute in this matter until there is a cache miss. When there is a cache miss in one thread the remaining threads continue to execute.

MIPS MT ASE

- The MIPS MT ASE is an application-specific extension of the MIPS32/MIPS64 instruction set and privileged resource architecture, meaning that it is a true architectural superset.
- A virtual processing element, or VPE, is an instantiation of the full MIPS32/MIPS64 ISA and privileged resource architecture (PRA), sufficient to run a per-processor OS image. A VPE can be thought of as an “exception domain”, as exception state and priority apply globally within a VPE, and only one exception can be dispatched at a time on a VPE.
- A conventional MIPS core embodies a single VPE.
- A thread context, or TC, is the hardware state necessary to support a thread of execution. This includes a set of general purpose registers (GPRs), a program counter (PC), and some coprocessor state.
- A thread of execution, or thread, is a sequential MIPS32/MIPS64 ISA instruction stream. A conventional MIPS processor runs a single thread at a time.


Resource Allocation Within a MT Core

Here is an illustration of how resources are distributed within a MT Core.

+ These are the resources that are common at the Core level are: the Pipeline, the Instruction Fetch Unit, the L1 Caches, the Load Store unit, the multiply divide unit, the arithmetic logic unit and the memory. There are also CP0 registers that are shared by all VPEs.

+ Each VPE has its own MMU, TLB and exception and interrupt logic. All CP0 Register that are not Common to the Core are duplicated for each VPE. In addition to the Standard MIPS CP0 registers there are additional registers are defined to be per-VPE, common for all TCs within the VPE.

+ There can be 2 Virtual Processors per core.

+ Each Thread Contest has it's own General Purpose Registers and internal Program counter. There are also several CP0 registers in each Thread Context.

+ There can be up to 9 Thread Contexts in the core. Each Thread context is associated with a specific VPE.

The all MT specific CP0 register will be discussed more in upcoming sections.