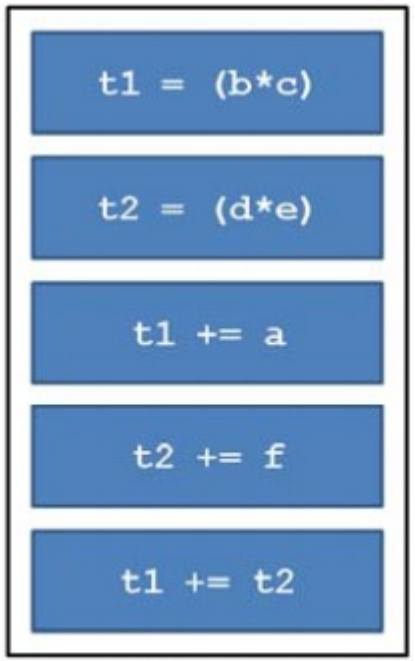
CENG311 Computer Architecture

Parallelism

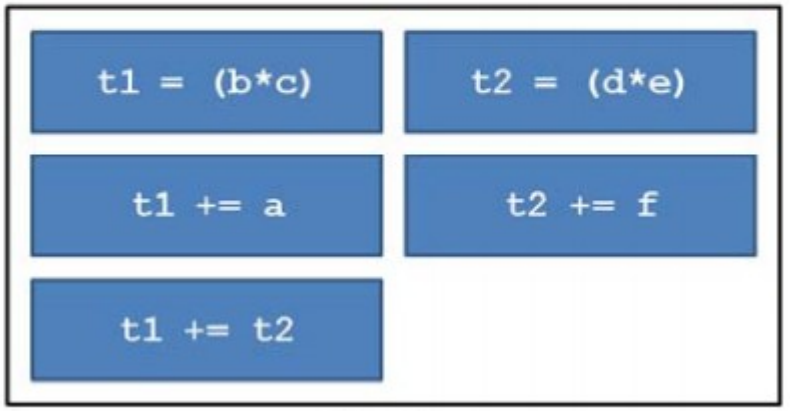
IZTECH, Fall 2023

14 December 2023

Instruction-Level Parallelism



5 cycles



3 cycles

Pipelining

Divide the instruction processing cycle into distinct “stages” of processing

Ensure there are enough hardware resources to process one instruction in each stage

Process a different instruction in each stage

- Instructions consecutive in program order are processed in consecutive stages

Increases instruction processing throughput

Multiple Issue

Static multiple issue

- Compiler groups instructions to be issued together
- Packages them into “issue slots”
- Compiler detects and avoids hazards

Dynamic multiple issue

- CPU examines instruction stream and chooses instructions to issue each cycle
- Compiler can help by reordering instructions
- CPU resolves hazards using advanced techniques at runtime

Speculation

“Guess” what to do with an instruction

- Start operation as soon as possible
- Check whether guess was right
  - If so, complete the operation
  - If not, roll-back and do the right thing

Common to static and dynamic multiple issue

Examples

- Speculate on branch outcome
  - Roll back if path taken is different
- Speculate on load
  - Roll back if location is updated

Static Multiple Issue

Compiler groups instructions into “issue packets”

- Group of instructions that can be issued on a single cycle
- Determined by pipeline resources required

Think of an issue packet as a very long instruction

- Specifies multiple concurrent operations
- Very Long Instruction Word (VLIW)

Dynamic Multiple Issue

“Superscalar” processors

CPU decides whether to issue 0, 1, 2, … each cycle

- Avoiding structural and data hazards

Avoids the need for compiler scheduling

- Though it may still help
- Code semantics ensured by the CPU

Out-of-Order Execution

Allow the CPU to execute instructions out of order to avoid stalls

- But commit result to registers in order
- In order issue, out of order executio, in order commit

Example

```
lw    $t0, 20($s2)
addu  $t1, $t0, $t2
sub   $s4, $s4, $t3
slti  $t5, $s4, 20
```

Can start sub while addu is waiting for lw

Thread-Level Parallelism

Parallelism through the simultaneous execution of different threads

Multiple data from multiple instructions (MIMD)

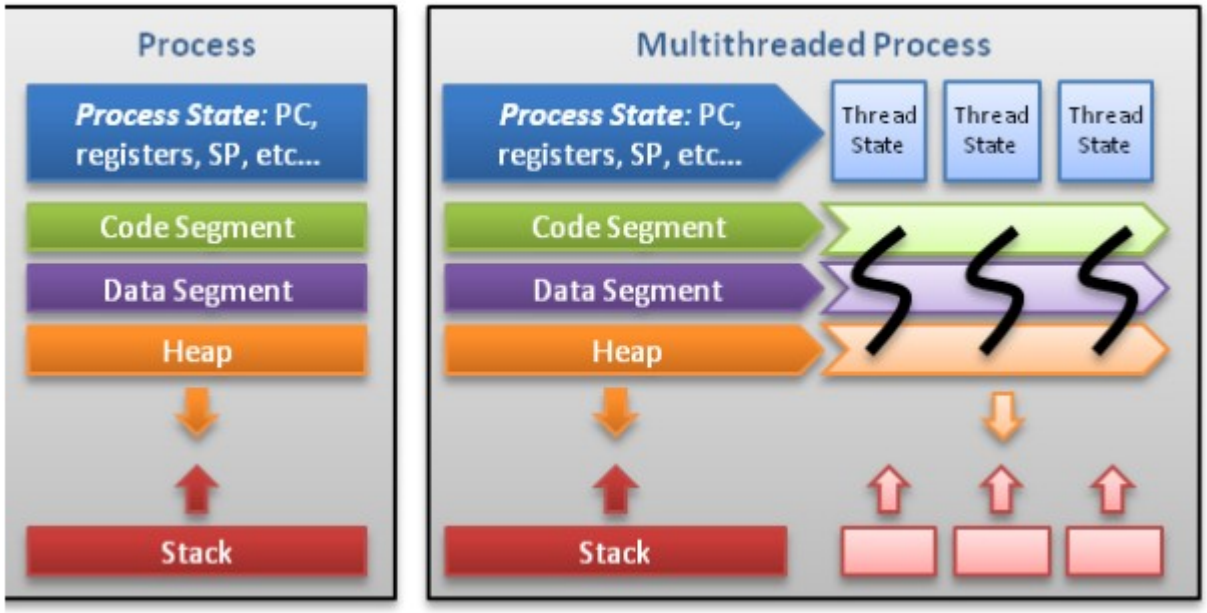
Multiprocessors

- Shared-memory multiprocessors
- Distributed-memory multiprocessors

Threads

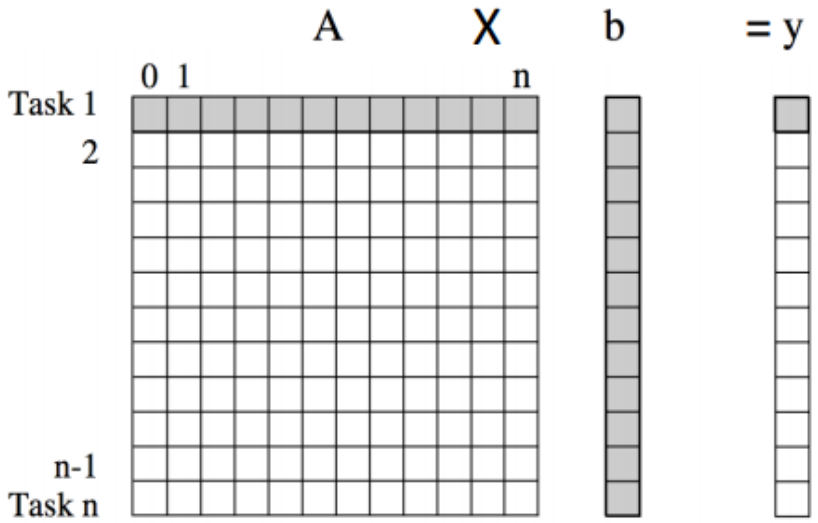
Threads use, and exist within, the process resources

Scheduled and run as independent entities



Thread as Function Instance

```
for (i = 0; i < n; i++)
  y[i] = dot_product(row(A, i),b);
```

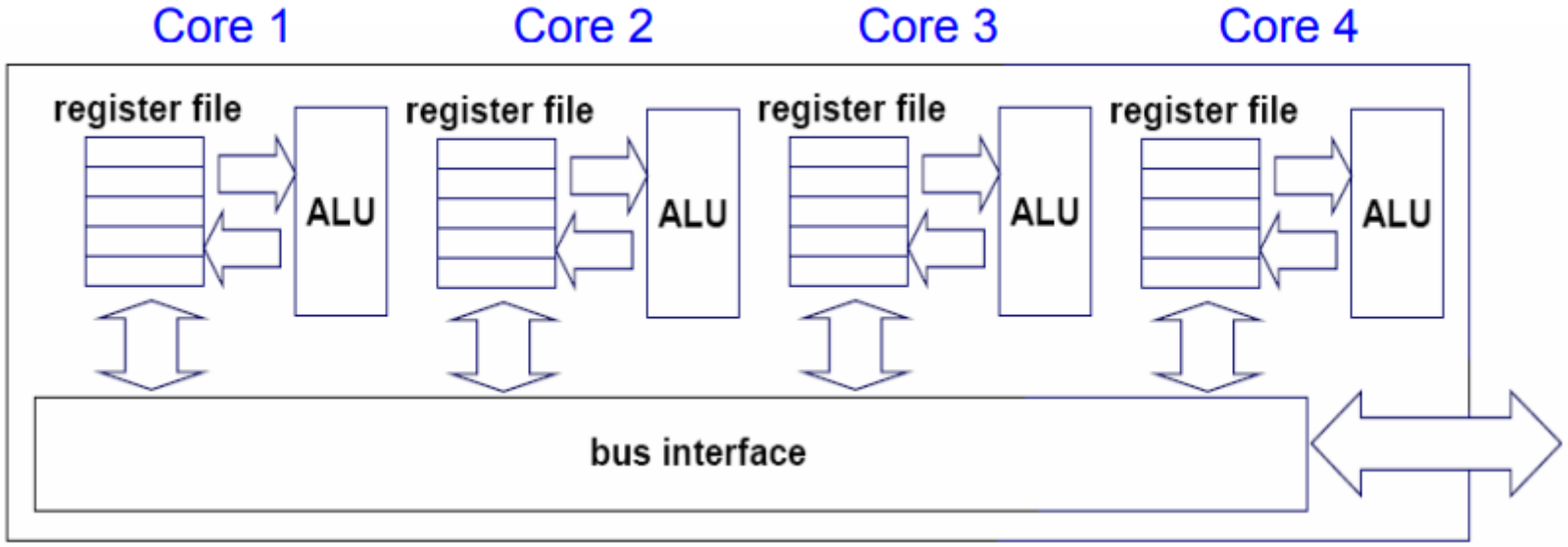


```
for (i = 0; i < n; i++)
    y[i] = create_thread(dot_product(row(A, i), b));
```

Multicore Processors

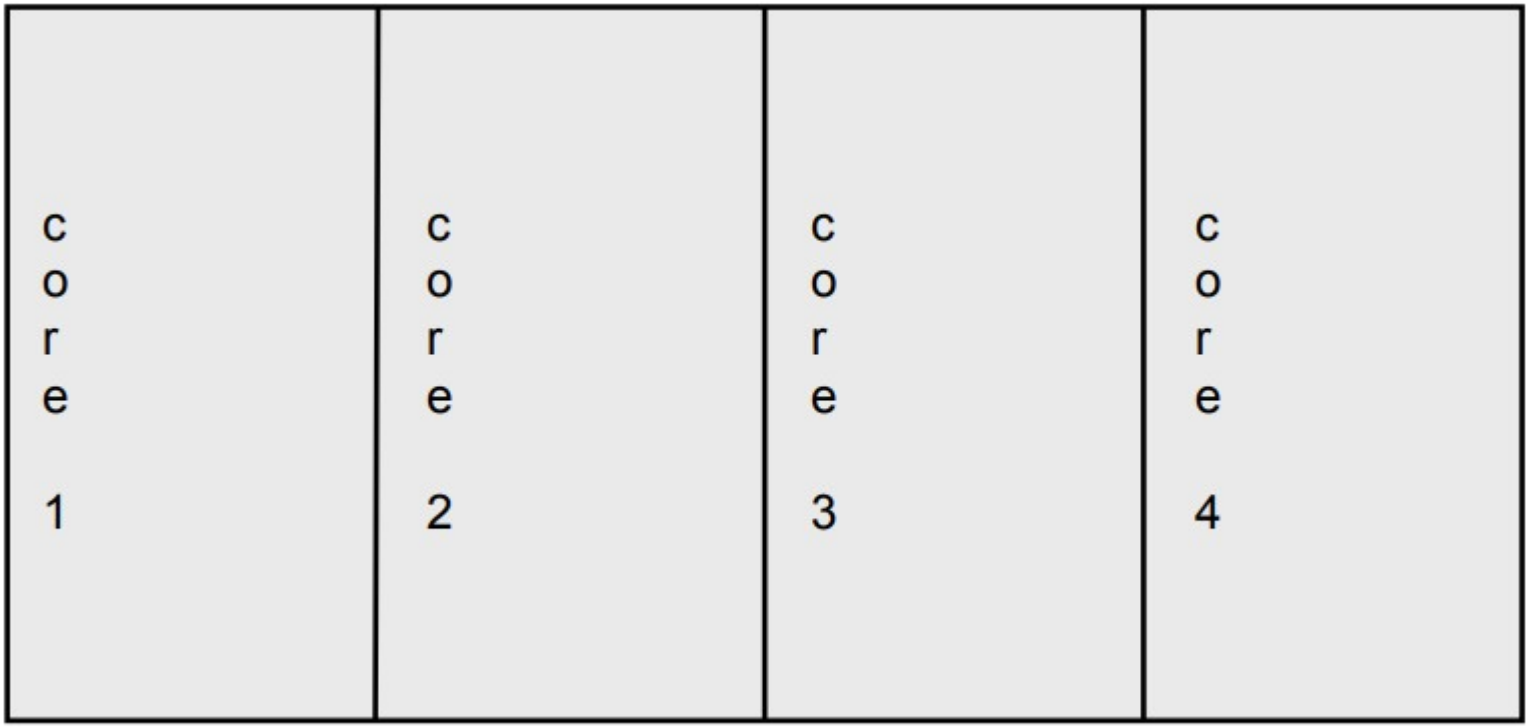
Multiple CPUs on a single chip

The most widely available shared-memory systems use one or more multicore processors

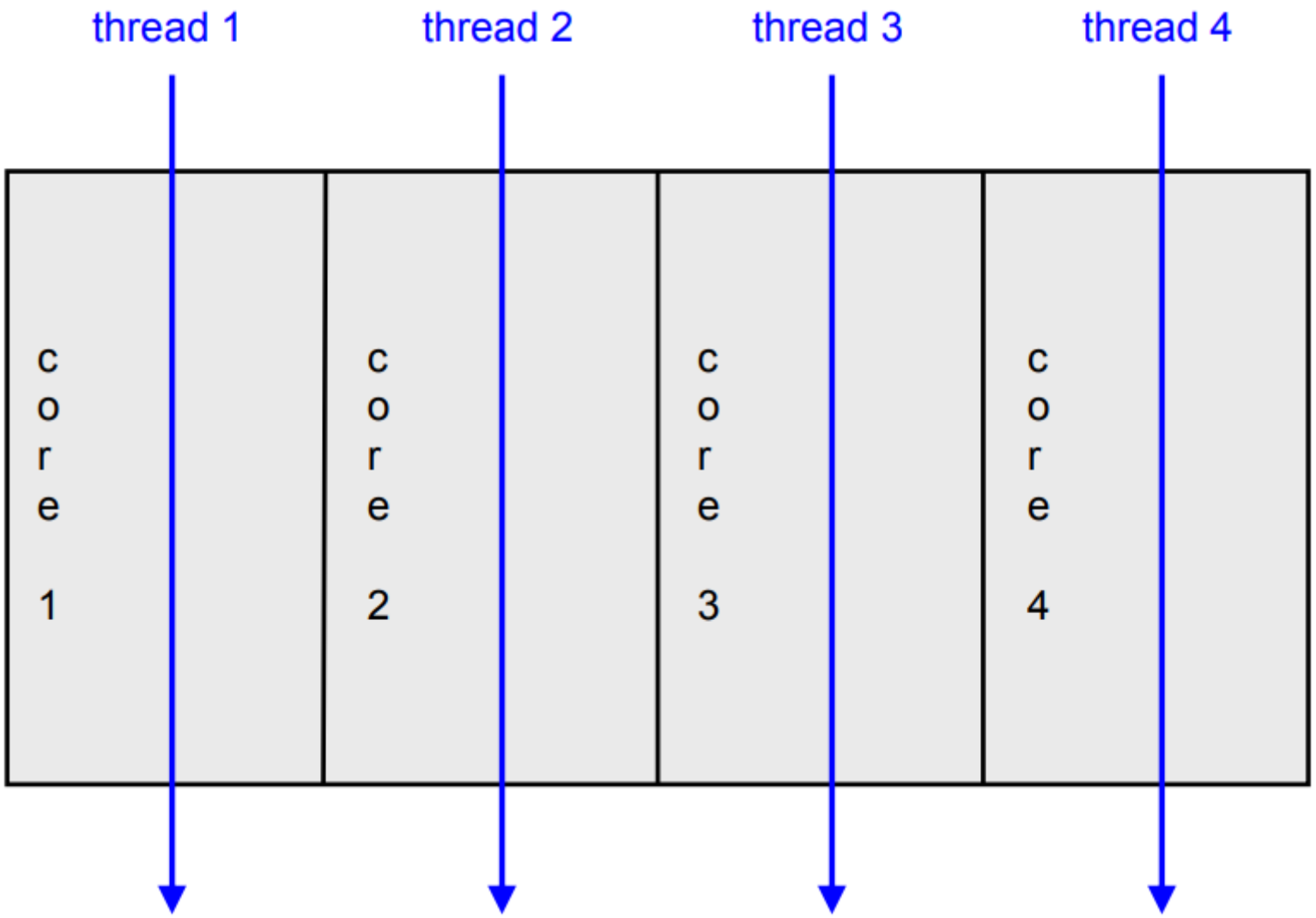


Multicore Processors

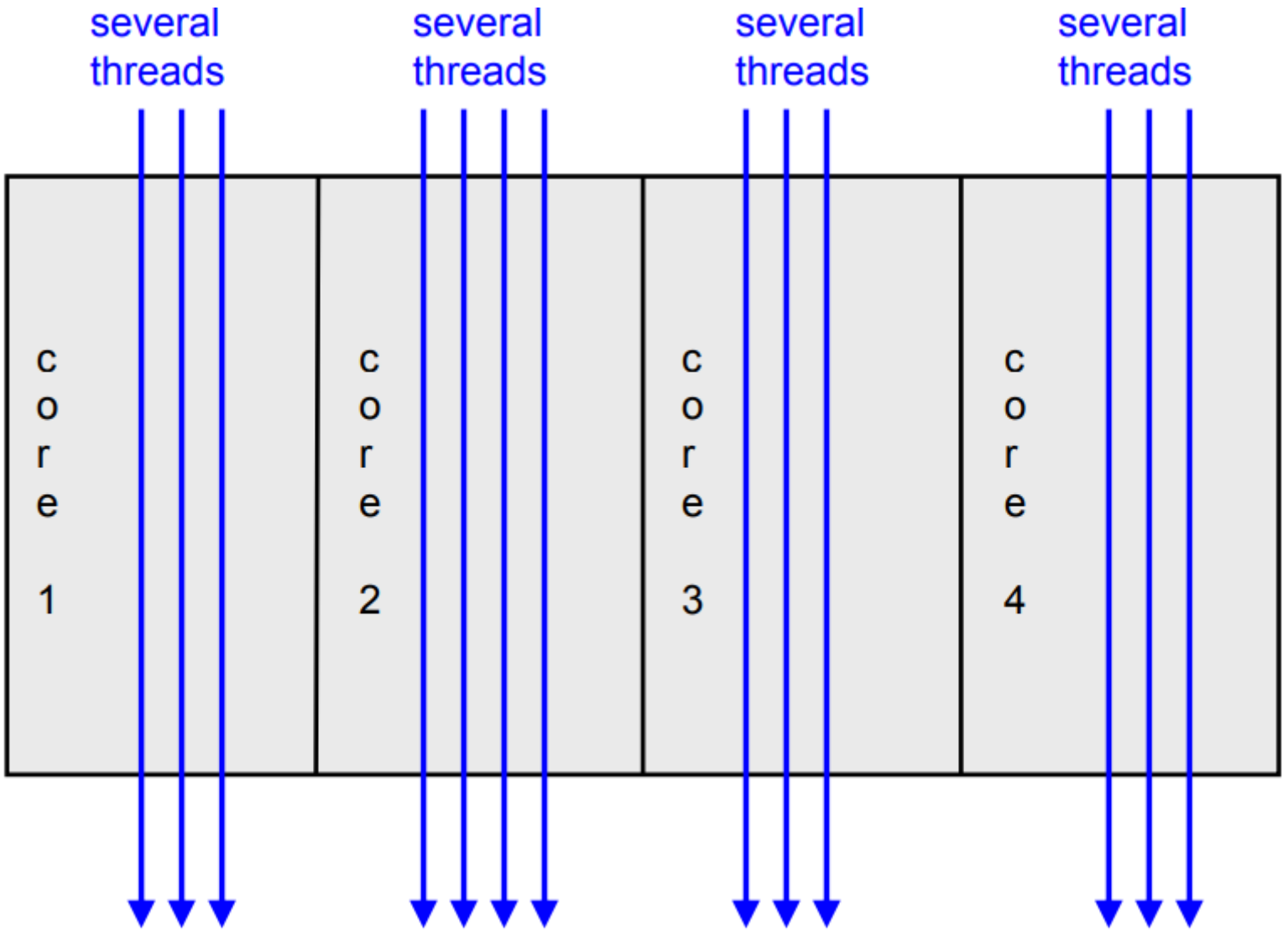
Cores fit in a single processor socket



Parallel Execution in Cores



Simultaneous Multithreading (SMT)



Sequential Performance Tuning

Sequential performance tuning is a time-driven process

Find the thing that takes the most time and make it take less time (i.e., make it more efficient)

May lead to program restructuring

- Changes in data storage and structure
- Rearrangement of tasks and operations

May look for opportunities for better resource utilization

- Cache management – Locality!

May look for opportunities for better processor usage

Parallel Performance Tuning

Parallel performance tuning might be described as conflict-driven or interaction-driven

Find the points of parallel interactions and determine the overheads associated with them

Overheads can be the cost of performing the interactions

- Transfer of data
- Extra operations to implement coordination

Overheads also include time spent waiting

- Lack of work
- Waiting for dependency to be satisfied

Parallel Programming

To use a scalable parallel computer, you must be able to write parallel programs

You must understand the programming model and the programming languages, libraries, and systems software used to implement it, and the platform

Programming Model

How the parallel platforms virtually appears to programmers

- Due to software layers, programming models can diverge from the underlying physical architecture model
- Use programming languages and environments that are close to the architecture

Functionalities provided

- Control
  - How parallel tasks can be created, terminated, and synchronized
- Naming
  - Shared or private data
  - Naming of data/processes/network abstraction
- Operations

OpenMP Overview

A set of compiler directives and library routines for parallel application programmers

Simplifies writing multi-threaded programs in Fortran, C and C++

OpenMP Program

Include header file

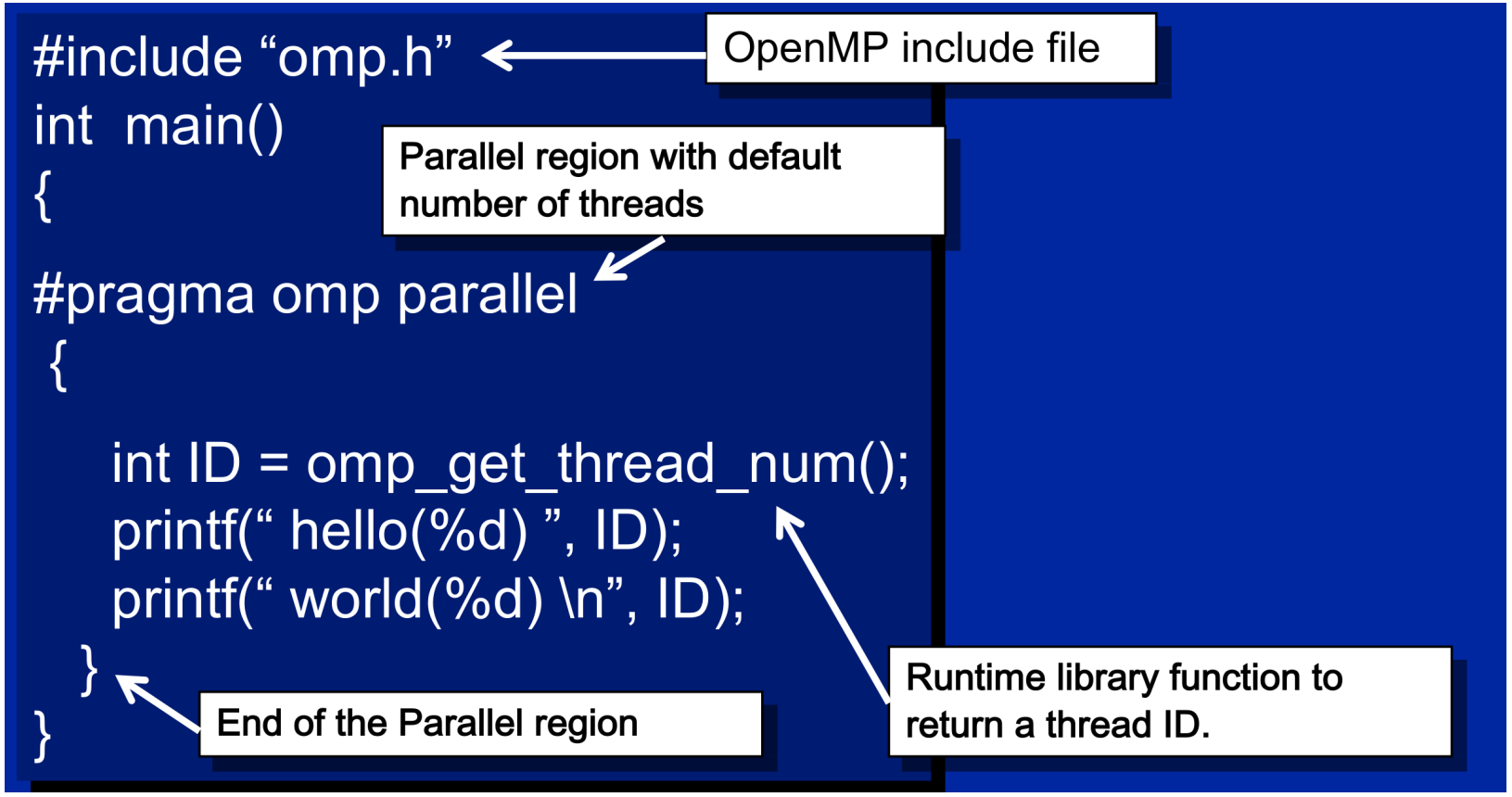
```
#include <omp.h>
```

Compiling

```
gcc -o omp_helloc -fopenmp omp_hello.c
```

Execute as a usual program

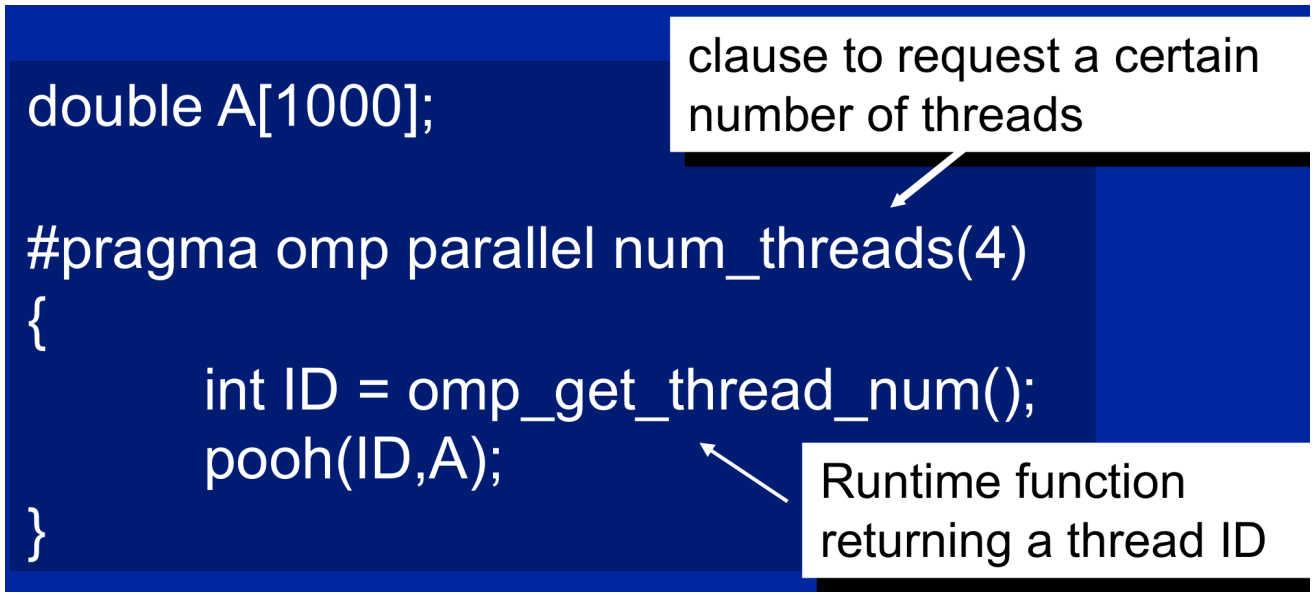
OpenMP HelloWorld



Thread Creation: Parallel Regions

You create threads in OpenMP with the parallel construct

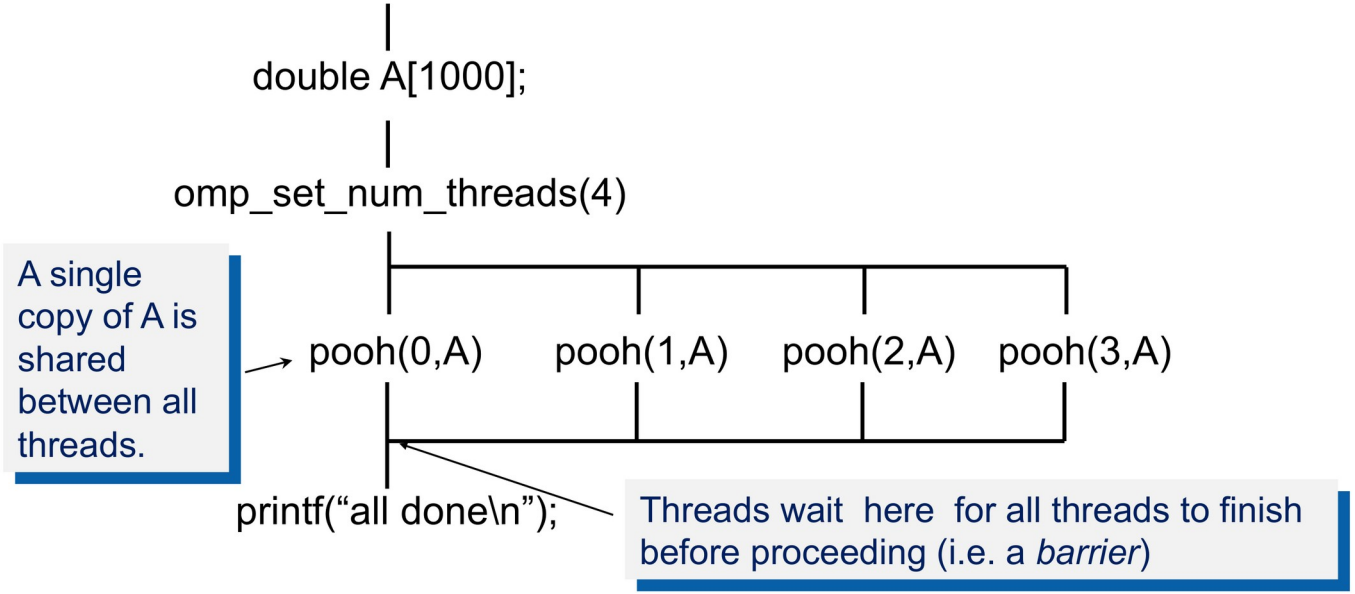
To create a 4 thread Parallel region



Thread Creation: Parallel Regions

Each thread executes the same code

```
double A[1000];
#pragma omp parallel num_threads(4)
{
    int ID = omp_get_thread_num();
    pooh(ID, A);
}
printf("all done\n");
```



Pthreads vs OpenMP

Pthreads is lower level, requires that the programmer explicitly specify the behavior of each thread

OpenMP sometimes allows the programmer to simply state that a block of code should be executed in parallel, and the precise determination of the tasks and which thread should execute them is left to the compiler and the runtime system

References

Chapter 4.10 (Computer Organization and Design: The Hardware/Software Interface by Hennessy/Patterson, 5th edition)

Related videos from Design of Digital Circuits course in ETH Zurich, by Onur Mutlu

https://www.youtube.com/watch?v=f522l7Q-t7g

http://www.youtube.com/watch?v=7XXgZlbBLls

Introduction to OpenMP playlist by Tim Mattson, Intel Labs / Senior Principal Engineer

https://www.youtube.com/watch?v=nE-xN4Bf8XI&list=PLLX-Q6B8xqZ8n8bwjGdzBJ25X2utwnoEG