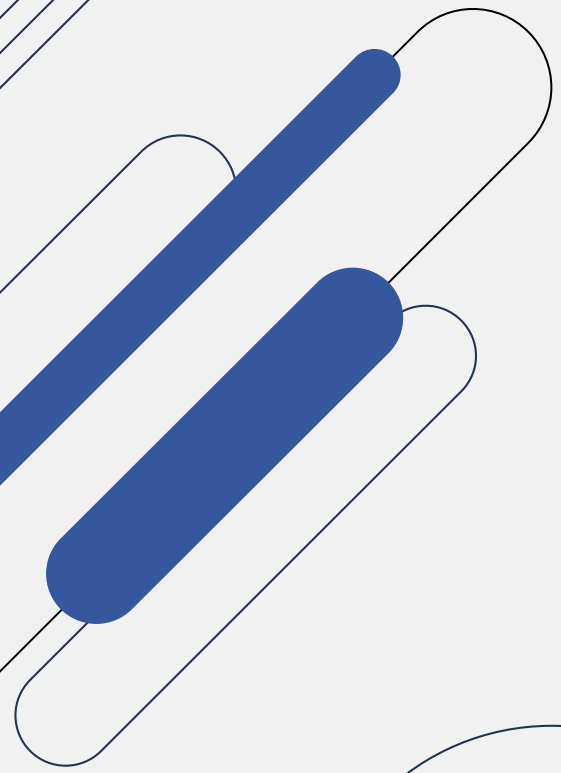
Parallel and Distributed Computing

Dr. Ali Sayyed
Department of Computer Science
National University of Computer & Emerging Sciences



Shared Memory Programming Models

OpenMP

Programming Multicore Systems

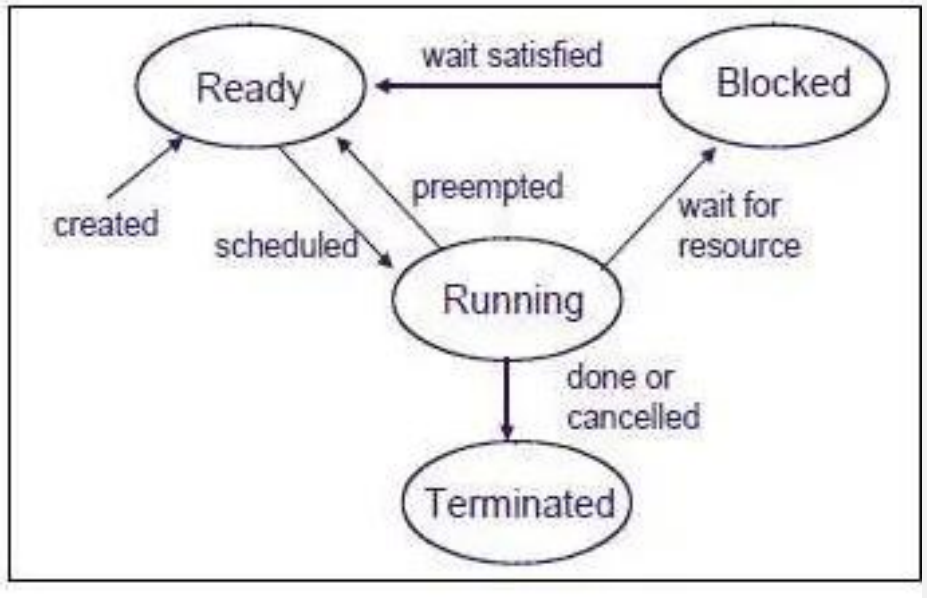
- Most of the **common languages were created prior** to the multicore age.
- As a result, many languages **cannot implicitly** (or **automatically**) employ multicore processors to speed up the execution of a program.
- Instead, **programmers** must specifically write software to leverage the multiple cores on a system.

Expediting Process Execution

- One way to speed up the execution of a single process is to decompose it into lightweight, independent execution flows (a **unit of work**) called **threads**.
- The OS **schedules threads** in the same manner as it schedules processes.
- On a multicore processor, the OS can speed up the execution of a multithreaded program by **scheduling** different threads to run on **separate cores**.
- The **maximum number of threads** that can execute in parallel is equal to the number of physical cores on the system.
    - The excess threads must **wait** their turn to execute
- Threads communicate **implicitly** by writing/ reading shared variables.

Thread Lifecycle

- A thread is “born” by the **create**() function, which places it in the ready state.
- A thread **runs when it is scheduled**.
- The running thread may be **blocked because it must wait** for some resource
- It may be **preempted** either because a higher **priority** thread is ready to run, or its **time** slice has expired.
- When the **resource becomes available** a blocked thread transitions to the Ready state and will eventually be scheduled to run again.
- Finally, a thread is **terminated** when it is done, or another thread requests its cancellation.



Shared Memory Programming

Several Thread Libraries/systems

- **PTHREADS is the POSIX Standard**
  - Relatively low level, possibly slow and relatively heavyweight
  - IEEE
- **OpenMP** standard for application-level programming
  - Support for scientific programming on shared memory
  - openmp.org
- **TBB**: Thread Building Blocks
  - Intel
- **Java threads**
  - Built on top of POSIX threads
  - Object within Java language

OpenMP Overview

- OpenMP (Open Multi-Processing) is a programming API that simplifies writing multi-threaded, shared-memory parallel programs in C, C++, and Fortran.
- **Compiler Directives**: OpenMP uses compiler directives (pragmas) to specify parallel regions of code, which the compiler then translates into threaded code
- **Runtime Library Routines**: OpenMP provides a runtime library that allows developers to manage threads, control the execution environment, and manipulate locks on memory locations.
- **Environment Variables**: OpenMP also uses environment variables to influence the runtime behavior of parallel programs.

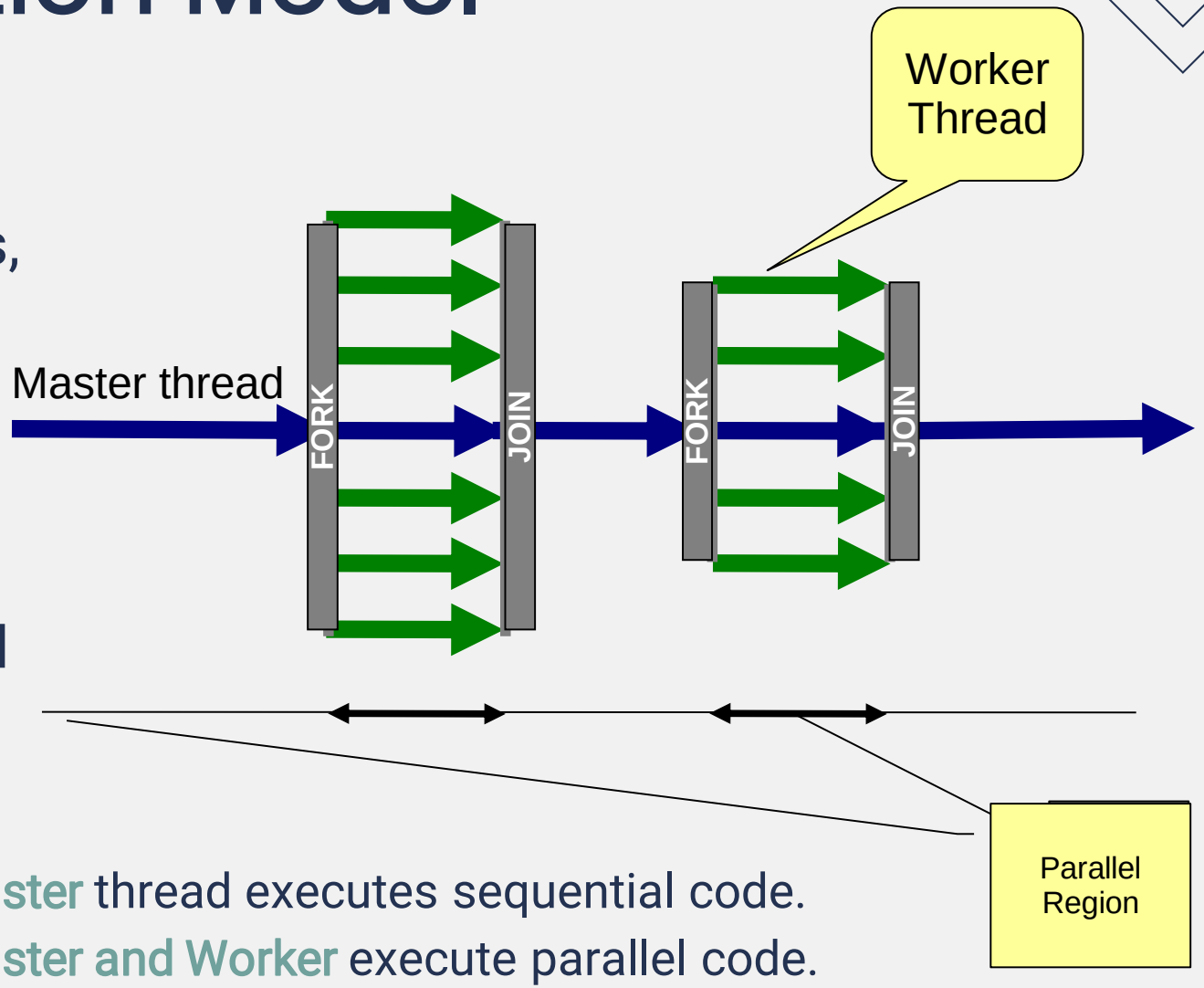
OpenMP API Overview

- Latest Specification is OpenMP 6.0 released in Nov 2024

- Learning more about MP
  - https://hpc.llnl.gov/tuts/openMP/
  - https://www.openmp.org
  - http://tinyurl.com/OpenMP-Tutorial
- A number of Compilers and Tools from various vendors and open source community initiatives implement the OpenMP API.
  - https://www.openmp.org/resources/openmp-compilers-tools/

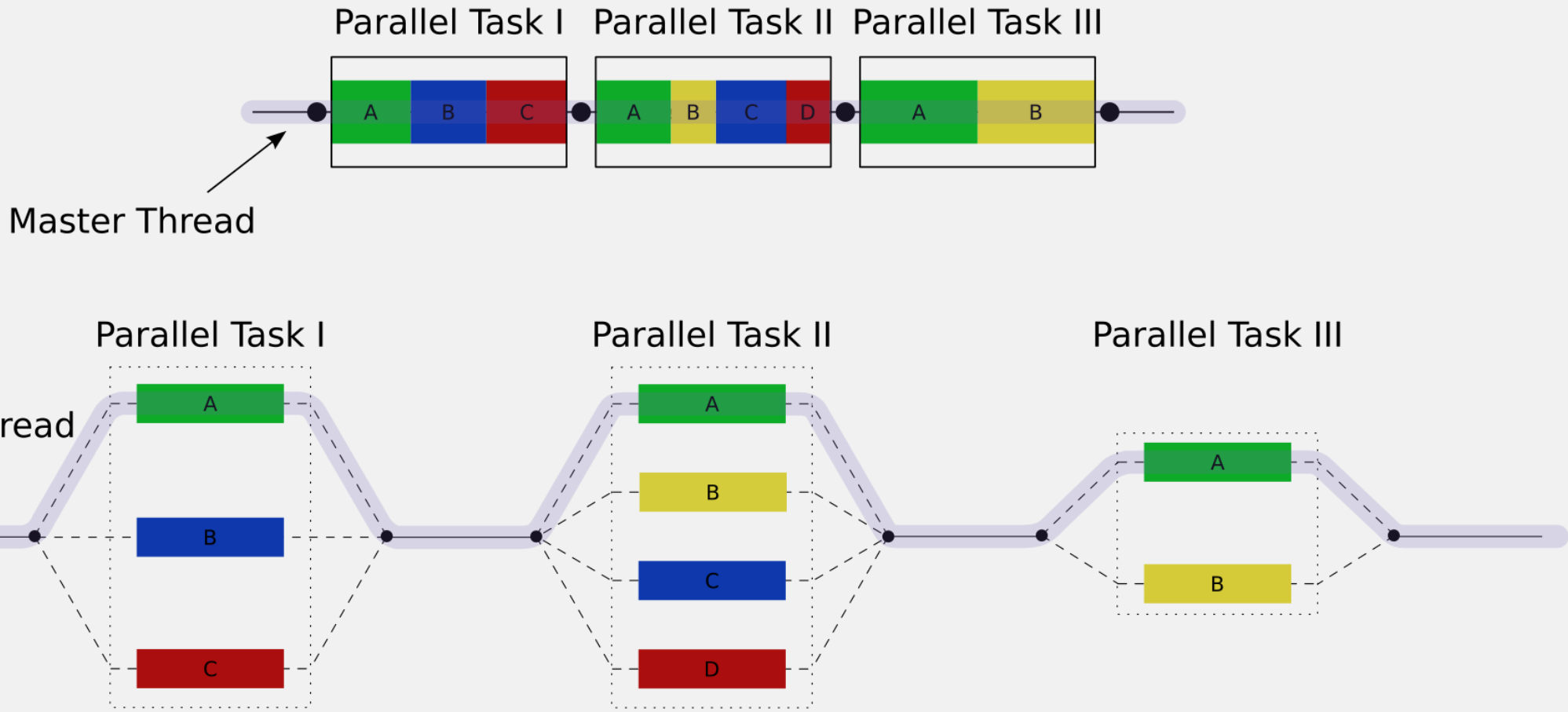
Execution Model

- OpenMP program starts single threaded
- To create additional threads, user starts a parallel region
  - additional threads are launched to create a team
  - original (master) thread is part of the team
- Repeat parallel regions as necessary
  - Fork-join model



- Master thread executes sequential code.
- Master and Worker execute parallel code.

Fork-Join Model



OpenMP API Example

Sequential code:

```
statement1;
statement2;
statement3;
```

Assume we want to execute statement 2 in parallel, and statement 1 and 3 sequentially.

OpenMP API Example

OpenMP parallel code:

```
statement 1;
#pragma <specific OpenMP directive>
statement2;
statement3;
```

Statement 2 (may be) executed in parallel.

Statement 1 and 3 are executed sequentially.

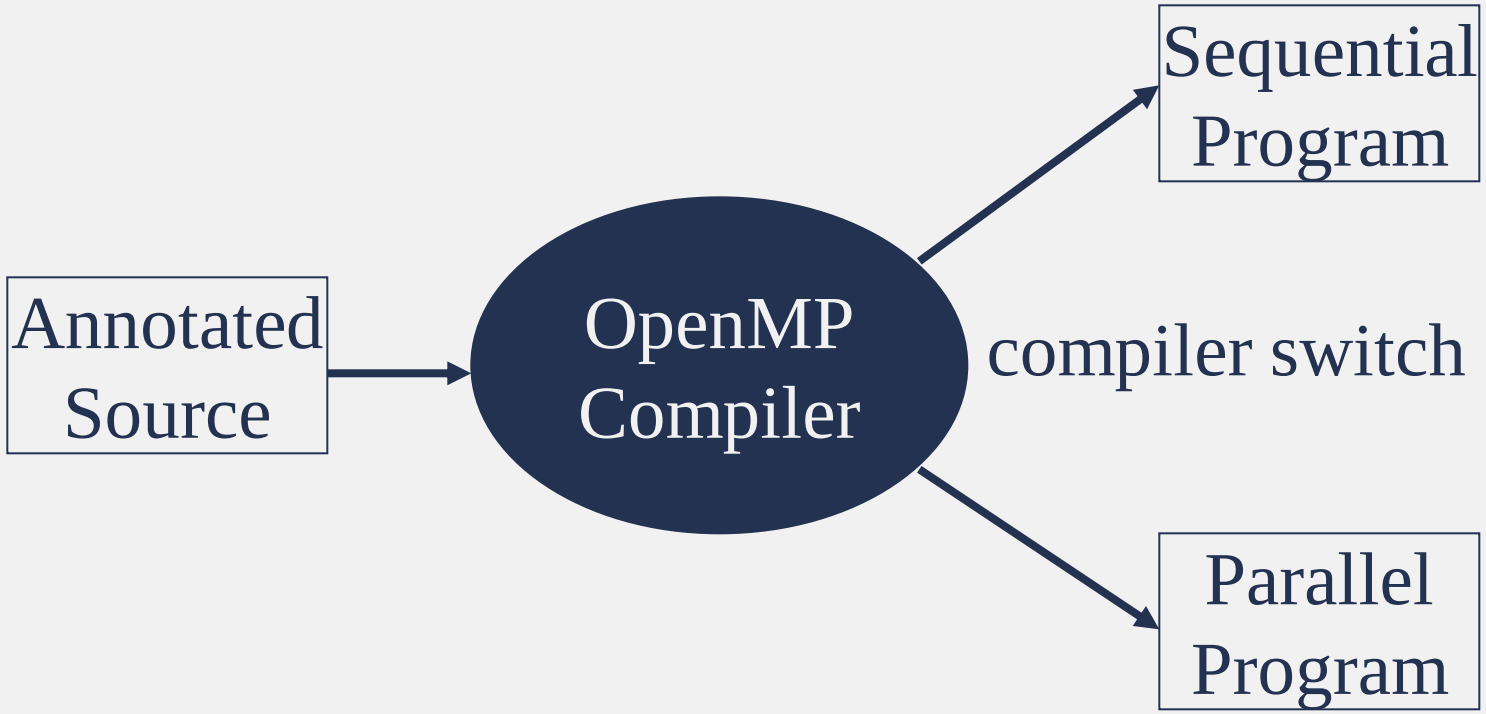
OpenMP Specification

https://www.openmp.org/spec-html/5.0/openmp.html

Important Note

- By giving a parallel directive, the user asserts that the program will remain correct if the statement is executed in parallel.
- OpenMP compiler does not check correctness.
- Some tools exist for helping with that.

OpenMP Example Usage



Compiler Switch

- Compile with -fopenmp to enable OpenMP if using GNU Compilor
- The Microsoft Visual C/C++ compiler supports the OpenMP 2.0 standard with the -openmp switch.
- Compile with -mp to enable OpenMP for multicore CPUs on all platforms, if using NVIDIA HPC compiler.
- Compile with -Qopenmp on Windows, or just -qopenmp or –fiopenmp on Linux or Mac OSX, if using Intel Compiler.

OpenMP Example Usage

- If you give sequential switch,
  - comments and pragmas are ignored.

- If you give parallel switch,
  - comments and/or pragmas are read, and
  - cause translation into parallel program.

- Ideally, one source for both sequential and parallel program (big maintenance plus).

Tools

- **HPE's** Code Parallelization Assistant, a part of the HPE Cray Programming Environment, can be used to identify and exploit parallelism. This tool helps highlight dependencies or bottlenecks during the optimization phase of program development.
- **Intel VTune Profiler** is a low-overhead performance profiling and analysis tool which helps to find and fix performance bottlenecks quickly.
- **Intel Advisor** is a design and analysis tool to help ensure your applications realize full performance potential on modern Intel processors.
- **Intel Inspector** is an easy-to-use memory and threading error debugger for C, C++, and Fortran applications that run on Windows* and Linux
- **NVIDIA Nsight Systems** for Linux is capable of capturing information about OpenMP events.

OpenMP Directives

- Parallelization directives:
  - parallel region
  - parallel for
  - etc.
- Data environment directives:
  - shared, private, threadprivate, reduction,
  - etc.
- Synchronization directives:
  - barrier, critical
  - etc.

General Rules about Directives

- They always apply to the next statement, which must be a structured block.
- Examples
  - ```
    #pragma omp construct [clause …]
    statement
    ```
  - ```
    #pragma omp construct [clause …]
    { statement1; statement2; statement3; }
    ```

OpenMP Parallel Region

- Defines a parallel region, which is code that will be executed by multiple threads in parallel.

```
#pragma omp parallel [clauses]
{
    code_block
}
```

- (Optional) Zero or more clauses.
- Each thread executes the same code.
- Each thread waits at the end.