Introduction to Parallel Computing

Course Instructor: Nausheen Shoaib

Marks Distribution

- Assignment, Project & Quiz: 20%
- Mid Terms : 30%
- Final: 50%

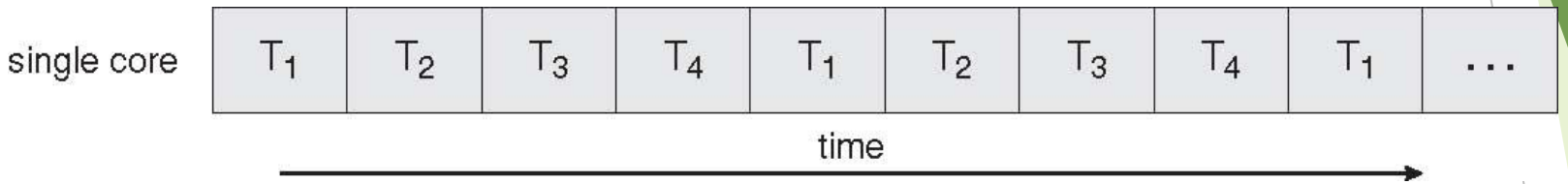
OS Concepts Revision

Process vs. Threads

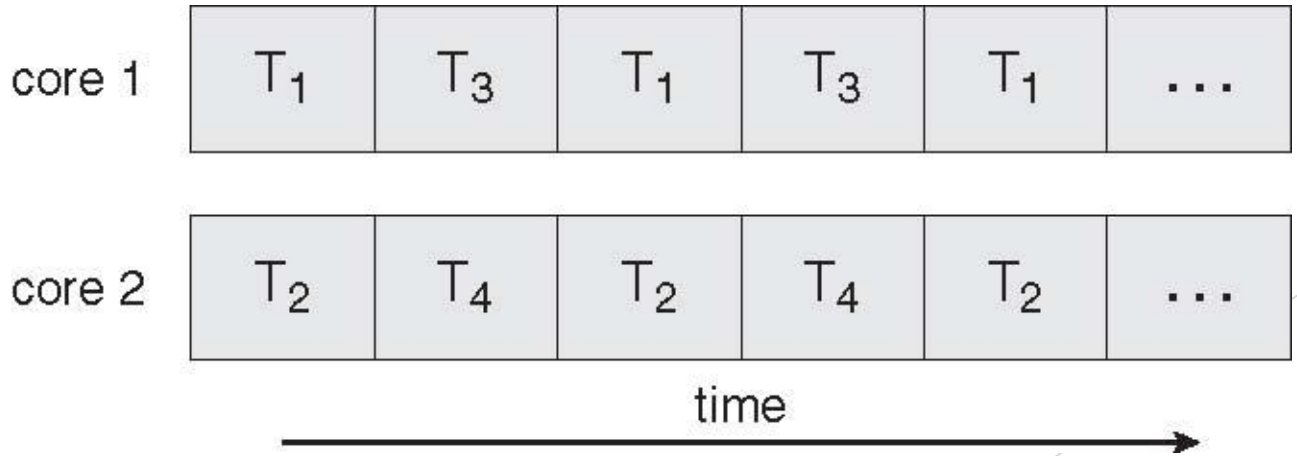
S.N.	Process	Thread
1.	Process is heavy weight or resource intensive.	Thread is light weight taking lesser resources than a process.
2.	Process switching needs interaction with operating system.	Thread switching does not need to interact with operating system.
3.	In multiple processing environments each process executes the same code but has its own memory and file resources.	All threads can share same set of open files, child processes.
4.	If one process is blocked then no other process can execute until the first process is unblocked.	While one thread is blocked and waiting, second thread in the same task can run.
5.	Multiple processes without using threads use more resources.	Multiple threaded processes use fewer resources.
6.	In multiple processes each process operates independently of the others.	One thread can read, write or change another thread's data.

Serial Execution vs. Parallel Execution

Concurrent Execution on a Single-core System

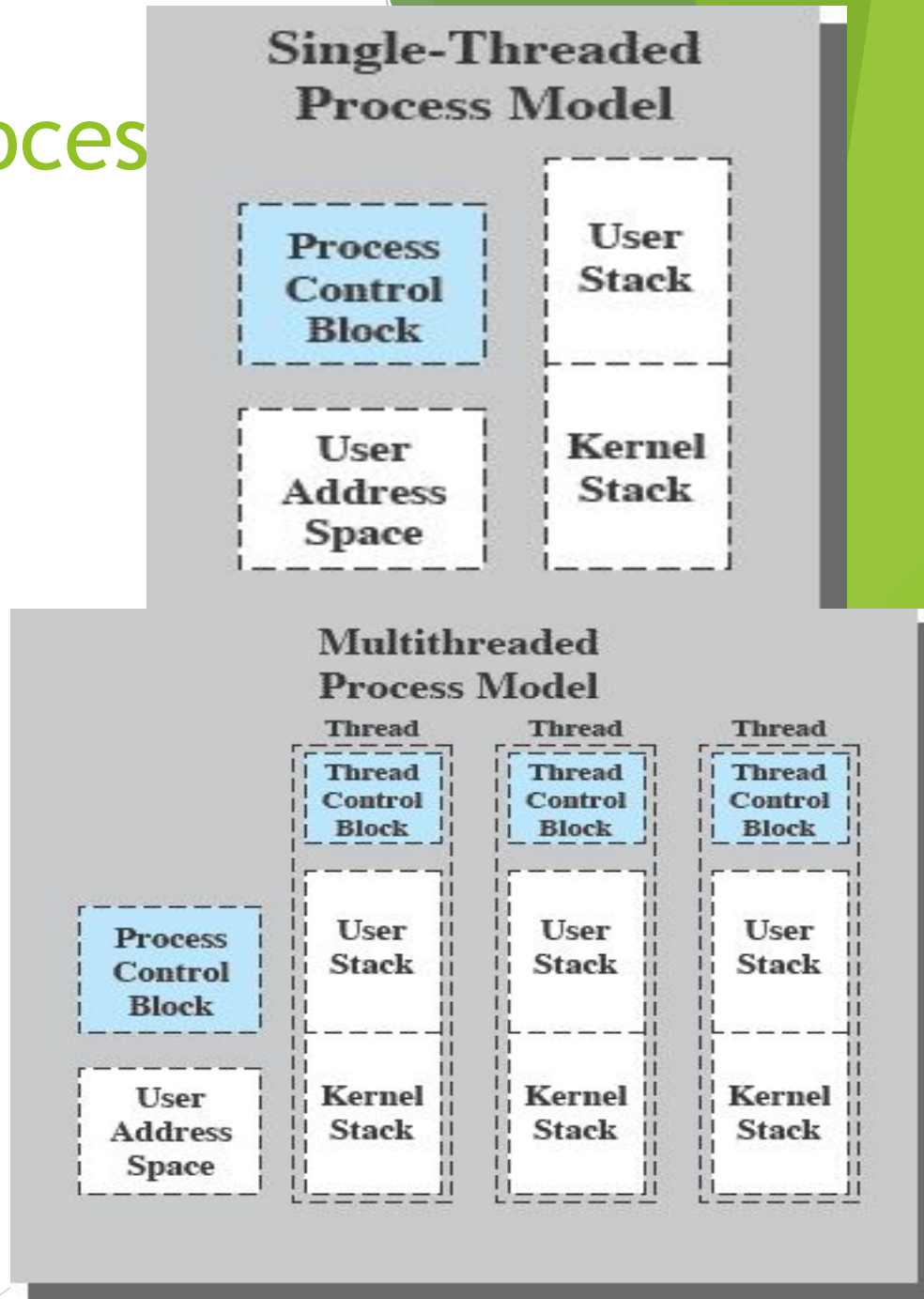


Parallel Execution on a Multicore System



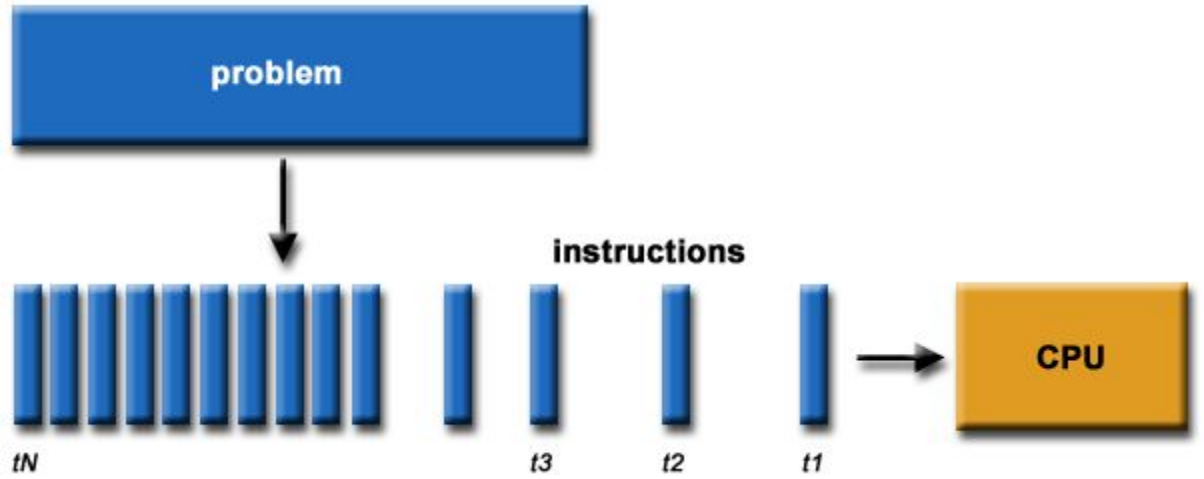
Single & Multithreading Proces

- Each **thread** has
  - An execution state (Running, Ready, etc.)
  - Saved thread context when not running
  - An execution stack
  - Some per-thread static storage for local variables
  - Access to the memory and resources of its process (all threads of a process share this)
- Suspending a process involves suspending all threads of the process
- Termination of a process terminates all threads within the process



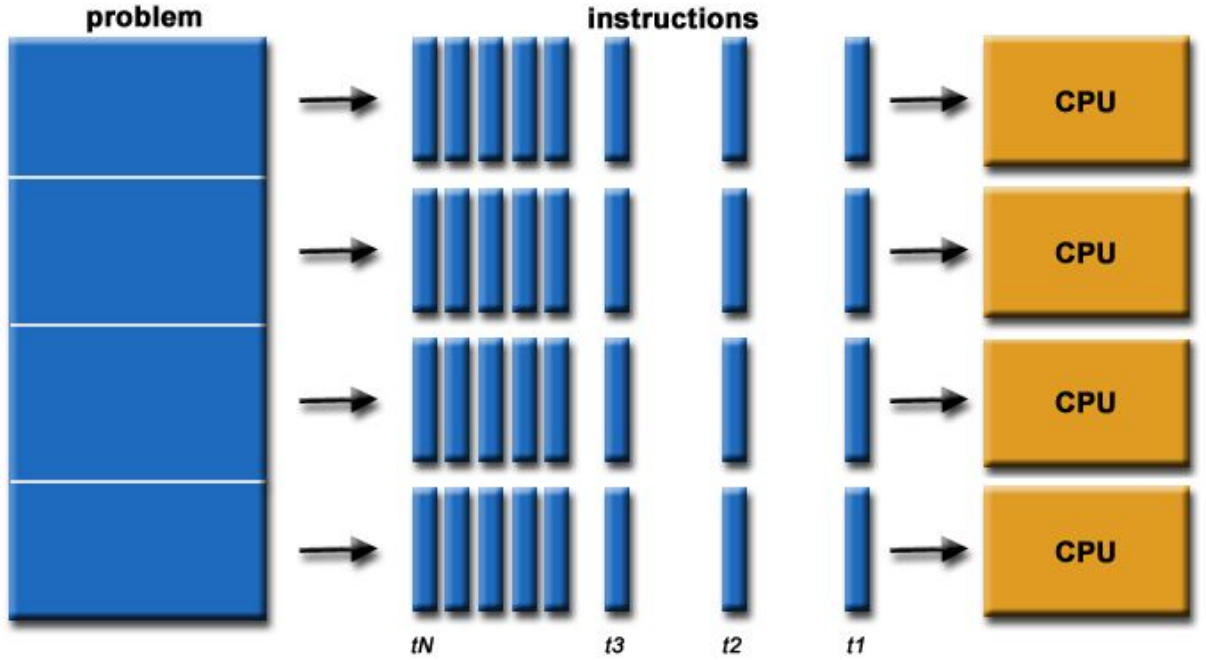
What is Parallel Computing? (1)

- Traditionally, software has been written for ***serial*** computation:
  - To be run on a single computer having a single Central Processing Unit (CPU);
  - A problem is broken into a discrete series of instructions.
  - Instructions are executed one after another.
  - Only one instruction may execute at any moment in time.



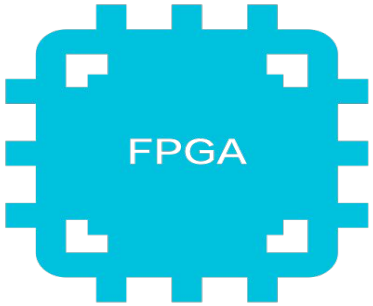
What is Parallel Computing? (2)

- In the simplest sense, ***parallel computing*** is the simultaneous use of multiple compute resources to solve a computational problem.
  - To be run using multiple CPUs
  - A problem is broken into discrete parts that can be solved concurrently
  - Each part is further broken down to a series of instructions
- Instructions from each part execute simultaneously on different CPUs



Parallel Computing: Resources

- The compute resources can include:
  - A single computer with multiple processors;
  - A single computer with (multiple) processor(s) and some specialized computer resources (GPU, FPGA …)
  - An arbitrary number of computers connected by a network;
  - A combination of both.



Parallel Computing: The computational problem

- The computational problem usually demonstrates characteristics such as the ability to be:
  - Broken apart into discrete pieces of work that can be solved simultaneously;
  - Execute multiple program instructions at any moment in time;
  - Solved in less time with multiple compute resources than with a single compute resource.

Parallel Computing: what for? (3)

- Example applications include:
  - parallel databases, data mining
  - web search engines, web based business services
  - computer-aided diagnosis in medicine
  - advanced graphics and virtual reality, particularly in the entertainment industry
  - networked video and multi-media technologies
- Ultimately, parallel computing is an attempt to maximize the infinite but seemingly scarce commodity called time.

Why Parallel Computing?

- This is a legitime question! Parallel computing is complex on any aspect!

- The primary reasons for using parallel computing:
  - Save time - wall clock time
  - Solve larger problems
  - Provide concurrency (do multiple things at the same time)

Limitations of Serial Computing

- Limits to serial computing - both physical and practical reasons pose significant constraints to simply building ever faster serial computers.
- Transmission speeds - the speed of a serial computer is directly dependent upon how fast data can move through hardware. Absolute limits are the speed of light (30 cm/nanosecond) and the transmission limit of copper wire (9 cm/nanosecond). Increasing speeds necessitate increasing proximity of processing elements.
- Economic limitations - it is increasingly expensive to make a single processor faster. Using a larger number of moderately fast commodity processors to achieve the same (or better) performance is less expensive.

Amdahl's Law

- gives the theoretical speedup in latency of the execution of a task at fixed workload that can be expected of a system whose resources are improved

$$speedup \leq \frac{1}{S + \frac{(1-S)}{N}}$$

Where S = portion of program executed serially
N = Processing Cores

Amdahl's Law Example

- we have an application that is 75 percent parallel and 25 percent serial. If we run this application on a system with two processing

  cores?

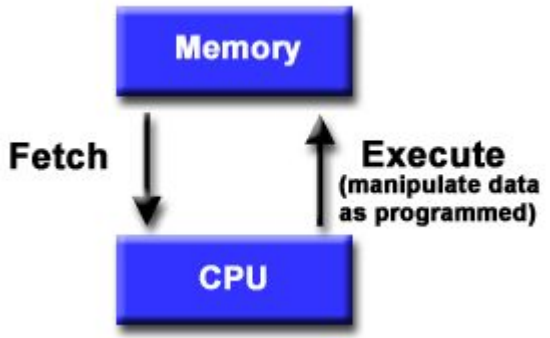
- S=25%=0.25, N= 2

- If we add two additional cores , calculate speedup?

Concepts and Terminology

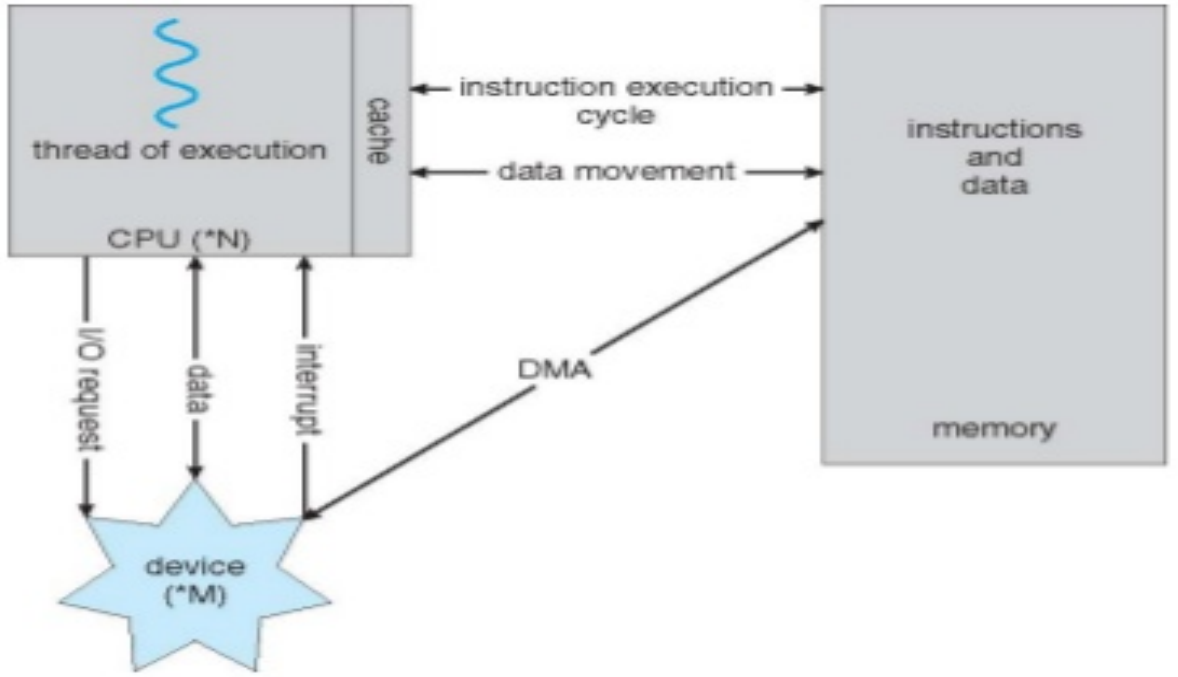
Basic Design

- Basic design
  - Memory is used to store both program and data instructions
  - Program instructions are coded data which tell the computer to do something
  - Data is simply information to be used by the program
- A central processing unit (CPU) gets instructions and/or data from memory, decodes the instructions and then ***sequentially*** performs them.



Von Nueman Architecture

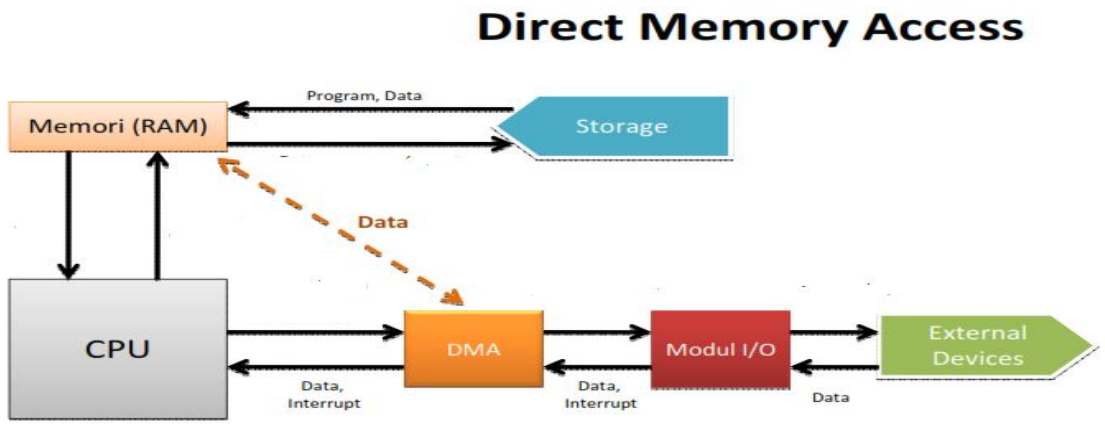
How a Modern Computer Works



A von Neumann architecture

Direct Memory Access Structure

- Used for high-speed I/O devices able to transmit information at close to memory speeds



- Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention

- Only one interrupt is generated per block, rather than the one interrupt per byte

Flynn Taxanomy

- The matrix below defines the 4 possible classifications according to Flynn

SISD **Single Instruction, Single Data**	**SIMD** **Single Instruction, Multiple Data**
MISD **Multiple Instruction, Single Data**	**MIMD** **Multiple Instruction, Multiple Data**