Introduction to Parallel & Distributed Programming

Lec 01 — Course Logistics, Introduction

Subodh Sharma (a.k.a SVS) | Jan 02, 2026

Course Logistics

Course Logistics

- COL 331 is a **co-requisite**

Course Logistics

- COL 331 is a **co-requisite**
  - Those who drop COL331 will be de-registered from COL380 at the end of the add-drop period

Course Logistics

- COL 331 is a **co-requisite**
  - Those who drop COL331 will be de-registered from COL380 at the end of the add-drop period
- Course Discussion: **Piazza**

Course Logistics

- COL 331 is a **co-requisite**
  - Those who drop COL331 will be de-registered from COL380 at the end of the add-drop period
- Course Discussion: **Piazza**
- Course Webpage: **https://subodhvsharma.github.io/course/col380**

Course Logistics

- COL 331 is a **co-requisite**
  - Those who drop COL331 will be de-registered from COL380 at the end of the add-drop period
- Course Discussion: **Piazza**
- Course Webpage: **https://subodhvsharma.github.io/course/col380**
- Audits: **Not allowed**

Course Logistics

- COL 331 is a **co-requisite**
  - Those who drop COL331 will be de-registered from COL380 at the end of the add-drop period
- Course Discussion: **Piazza**
- Course Webpage: **https://subodhvsharma.github.io/course/col380**
- Audits: **Not allowed**
- Attendace: **RollCall (**htpps://rollcall.iitd.ac.in**)**

Course Logistics

- COL 331 is a **co-requisite**
  - Those who drop COL331 will be de-registered from COL380 at the end of the add-drop period
- Course Discussion: **Piazza**
- Course Webpage: **https://subodhvsharma.github.io/course/col380**
- Audits: **Not allowed**
- Attendace: **RollCall (**htpps://rollcall.iitd.ac.in**)**
  - 75% pre-minor required for reminor, 75% in semester for remajor

Course Logistics

- COL 331 is a **co-requisite**
  - Those who drop COL331 will be de-registered from COL380 at the end of the add-drop period
- Course Discussion: **Piazza**
- Course Webpage: **https://subodhvsharma.github.io/course/col380**
- Audits: **Not allowed**
- Attendace: **RollCall (**htpps://rollcall.iitd.ac.in**)**
  - 75% pre-minor required for reminor, 75% in semester for remajor
- 4 Lab Assignments, 2 In-class Quizzes, 1 Lab Exam, Minor, Major; Additional task — **Transcribing!**

Transcribing — Begins from the Next Class!

- Drafting of the entire lecture content **in Latex**
  - **Key learning outcomes** for that content
  - Key takeaways (5-10 bullet items)
  - In **2 to 3 examples to motivate** the topical discussion
  - Comprehensive references
  - **4-8 problems** and their solutions

Academic Dishonesty

Academic Dishonesty

- **Any** act of academic dishonesty will lead to the following:

Academic Dishonesty

- **Any** act of academic dishonesty will lead to the following:
  - **Negative marks** of the total of that evaluation task would be applied

Academic Dishonesty

- **Any** act of academic dishonesty will lead to the following:
  - **Negative marks** of the total of that evaluation task would be applied
  - **1 letter grade drop**

Academic Dishonesty

- **Any** act of academic dishonesty will lead to the following:
  - **Negative marks** of the total of that evaluation task would be applied
  - **1 letter grade drop**
  - **For Grade 1 offence —** (cheating in assessments other than midsem/major)

Academic Dishonesty

- **Any** act of academic dishonesty will lead to the following:
  - **Negative marks** of the total of that evaluation task would be applied
  - **1 letter grade drop**
  - **For Grade 1 offence —** (cheating in assessments other than midsem/major)
    - **Departmental DISCO** (the list of penalties will be available soon)

Academic Dishonesty

- **Any** act of academic dishonesty will lead to the following:
  - **Negative marks** of the total of that evaluation task would be applied
  - **1 letter grade drop**
  - **For Grade 1 offence —** (cheating in assessments other than midsem/major)
    - **Departmental DISCO** (the list of penalties will be available soon)
      - Such as — **Debarred from OCS placements and Internship etc.**

Academic Dishonesty

- **Any** act of academic dishonesty will lead to the following:
  - **Negative marks** of the total of that evaluation task would be applied
  - **1 letter grade drop**
  - **For Grade 1 offence —** (cheating in assessments other than midsem/major)
    - **Departmental DISCO** (the list of penalties will be available soon)
      - Such as — **Debarred from OCS placements and Internship etc.**
  - **For Grade 2 & 3 offence -** (in midsem/major)

Academic Dishonesty

- **Any** act of academic dishonesty will lead to the following:
  - **Negative marks** of the total of that evaluation task would be applied
  - **1 letter grade drop**
  - **For Grade 1 offence —** (cheating in assessments other than midsem/major)
    - **Departmental DISCO** (the list of penalties will be available soon)
      - Such as — **Debarred from OCS placements and Internship etc.**
  - **For Grade 2 & 3 offence -** (in midsem/major)
    - **Institute DISCO**

About the Course

About the Course

- **Learning Goals:**

About the Course

- **Learning Goals:**
  - Write correct and efficient parallel programs; Programming Models

About the Course

- **Learning Goals:**
  - Write correct and efficient parallel programs; Programming Models
    - OpenMP, CUDA, MPI; Profilers and Debuggers

About the Course

- **Learning Goals:**
  - Write correct and efficient parallel programs; Programming Models
    - OpenMP, CUDA, MPI; Profilers and Debuggers
  - Measure parallel performance and analyse correctness

About the Course

- **Learning Goals:**
  - Write correct and efficient parallel programs; Programming Models
    - OpenMP, CUDA, MPI; Profilers and Debuggers
  - Measure parallel performance and analyse correctness
    - Cost and Performance models

About the Course

- **Learning Goals:**
  - Write correct and efficient parallel programs; Programming Models
    - OpenMP, CUDA, MPI; Profilers and Debuggers
  - Measure parallel performance and analyse correctness
    - Cost and Performance models
  - Exposure to parallel algorithms and data structures

About the Course

- **Learning Goals:**
  - Write correct and efficient parallel programs; Programming Models
    - OpenMP, CUDA, MPI; Profilers and Debuggers
  - Measure parallel performance and analyse correctness
    - Cost and Performance models
  - Exposure to parallel algorithms and data structures
    - Map-reduce, fork-join, …

About the Course

- **Learning Goals:**
  - Write correct and efficient parallel programs; Programming Models
    - OpenMP, CUDA, MPI; Profilers and Debuggers
  - Measure parallel performance and analyse correctness
    - Cost and Performance models
  - Exposure to parallel algorithms and data structures
    - Map-reduce, fork-join, …
  - Common distributed systems concepts

About the Course

- **Learning Goals:**
  - Write correct and efficient parallel programs; Programming Models
    - OpenMP, CUDA, MPI; Profilers and Debuggers
  - Measure parallel performance and analyse correctness
    - Cost and Performance models
  - Exposure to parallel algorithms and data structures
    - Map-reduce, fork-join, …
  - Common distributed systems concepts
    - Synchronisations, Logical clocks, Consensus,

Setting Expectations

Setting Expectations

- **Be regular** in attending classes

Setting Expectations

- **Be regular** in attending classes
  - Come prepared reading the supplementary material

Setting Expectations

- **Be regular** in attending classes
  - Come prepared reading the supplementary material
- Program regularly and on your own

Setting Expectations

- **Be regular** in attending classes
  - Come prepared reading the supplementary material
- Program regularly and on your own
- **Speak to me** — I don’t bite!

Setting Expectations

- **Be regular** in attending classes
  - Come prepared reading the supplementary material
- Program regularly and on your own
- **Speak to me** — I don’t bite!
- Check regularly Moodle and Piazza

Setting Expectations

- **Be regular** in attending classes
  - Come prepared reading the supplementary material
- Program regularly and on your own
- **Speak to me** — I don’t bite!
- Check regularly Moodle and Piazza
- It’s a 2-0-2 course, but felt like 2-0-6 — **deal with it** and learn to start early!

Parallel, Distributed, Concurrency

Terms to be used frequently

Parallel, Distributed, Concurrency

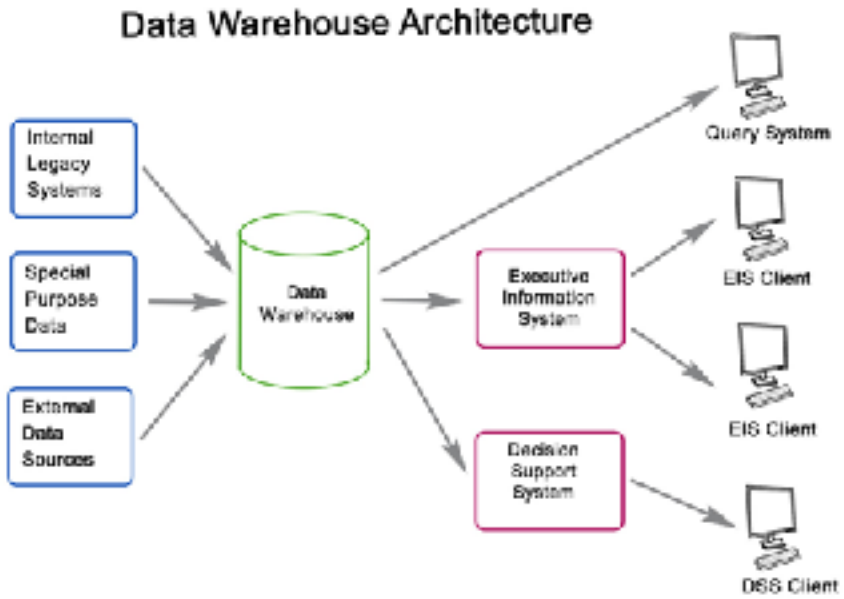
Terms to be used frequently

- **Concurrency:** Multiple task **in progress** simultaneously
- **Parallelism:** Concurrency but **in close coordination**
  - **Eg: Matrix multiplication: Threads may share the same shared memory arrays**
- **Distributed:** Concurrency **but loosely coupled**

Parallel, Distributed, Concurrency

Terms to be used frequently

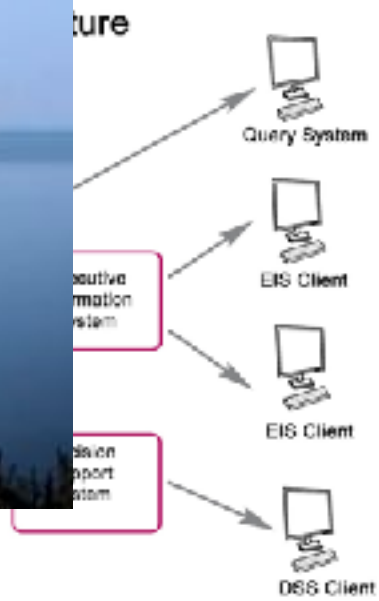
- **Concurrency:** Multiple task **in progress** simultaneously
- **Parallelism:** Concurrency but **in close coordination**
  - **Eg: Matrix multiplication: Threads may share the same shared memory arrays**
- **Distributed:** Concurrency **but loosely coupled**



Parallel, Distributed, Concurrency

Terms to

- **Concur**
  simultan
- **Parallel**
  coordin
  - **Eg: Ma**
    share
- **Distribu**
  coupled



Need for Concurrency in Computing

Courtesy: Nano Banana

Need for Concurrency in Computing

- $P \propto V^2 . f$

Courtesy: Nano Banana

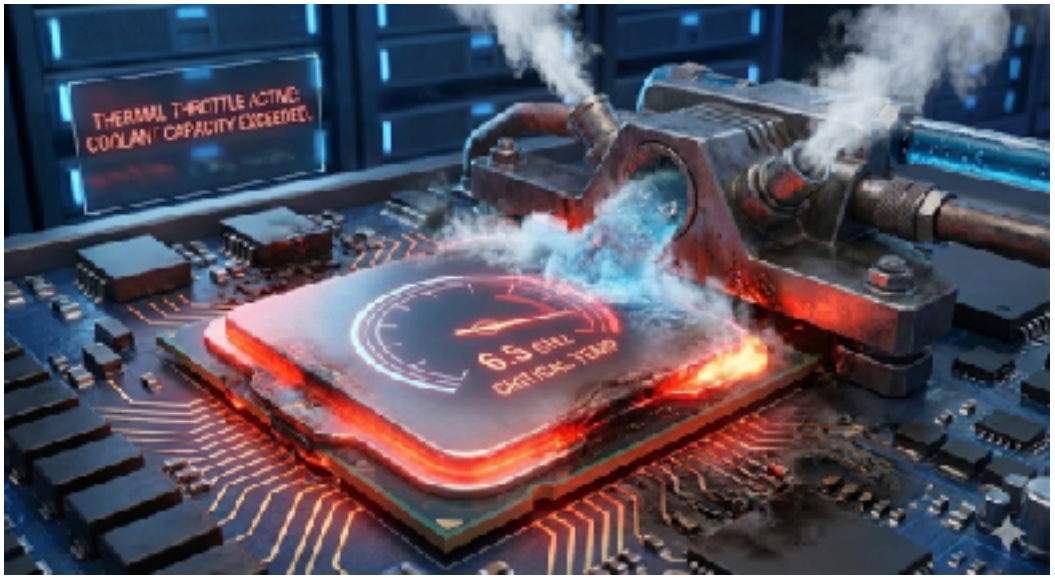
Need for Concurrency in Computing

- $P \propto V^2 . f$
  - Faster processors => Higher power consumption => **Higher heat dissipation => Unreliable processors**

Courtesy: Nano Banana

Need for Concurrency in Computing

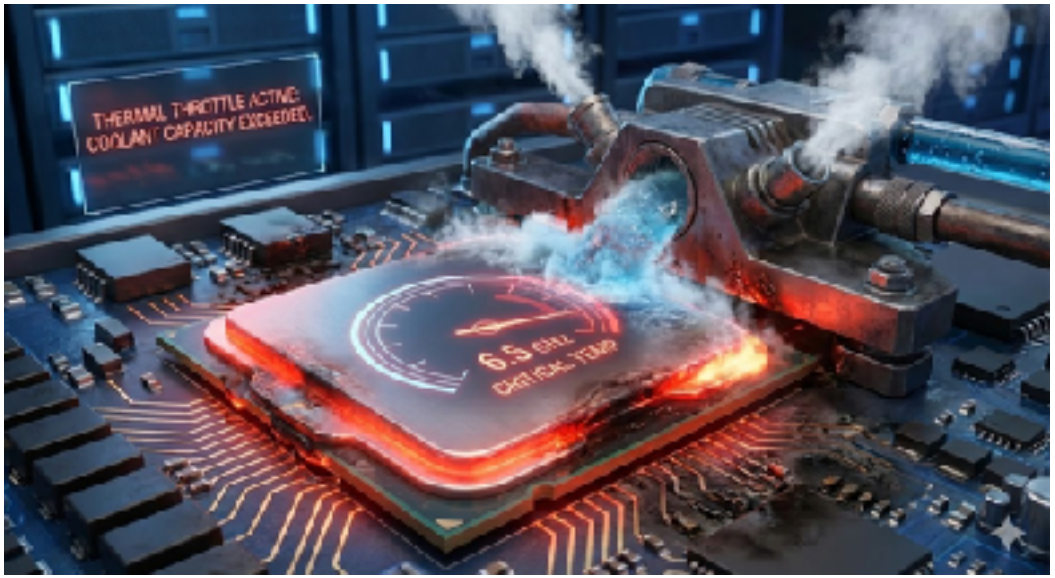
- $P \propto V^2 . f$
  - Faster processors => Higher power consumption => **Higher heat dissipation => Unreliable processors**



Courtesy: Nano Banana

Need for Concurrency in Computing

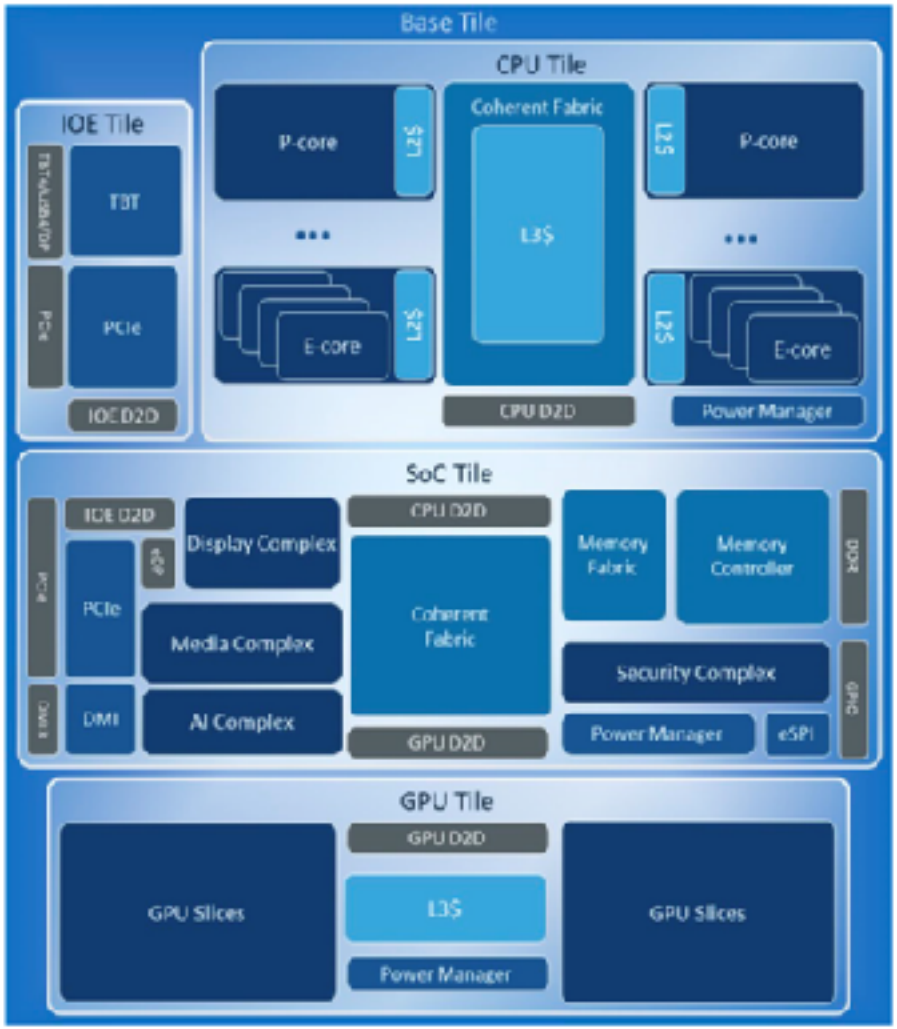
- $P \propto V^2 . f$
  - Faster processors => Higher power consumption => **Higher heat dissipation => Unreliable processors**
- **The V/F curve**: For $f \geq 5Ghz$, the physics break down, even to **get a jump of 100 MHz**, large jump in voltage is required



Courtesy: Nano Banana

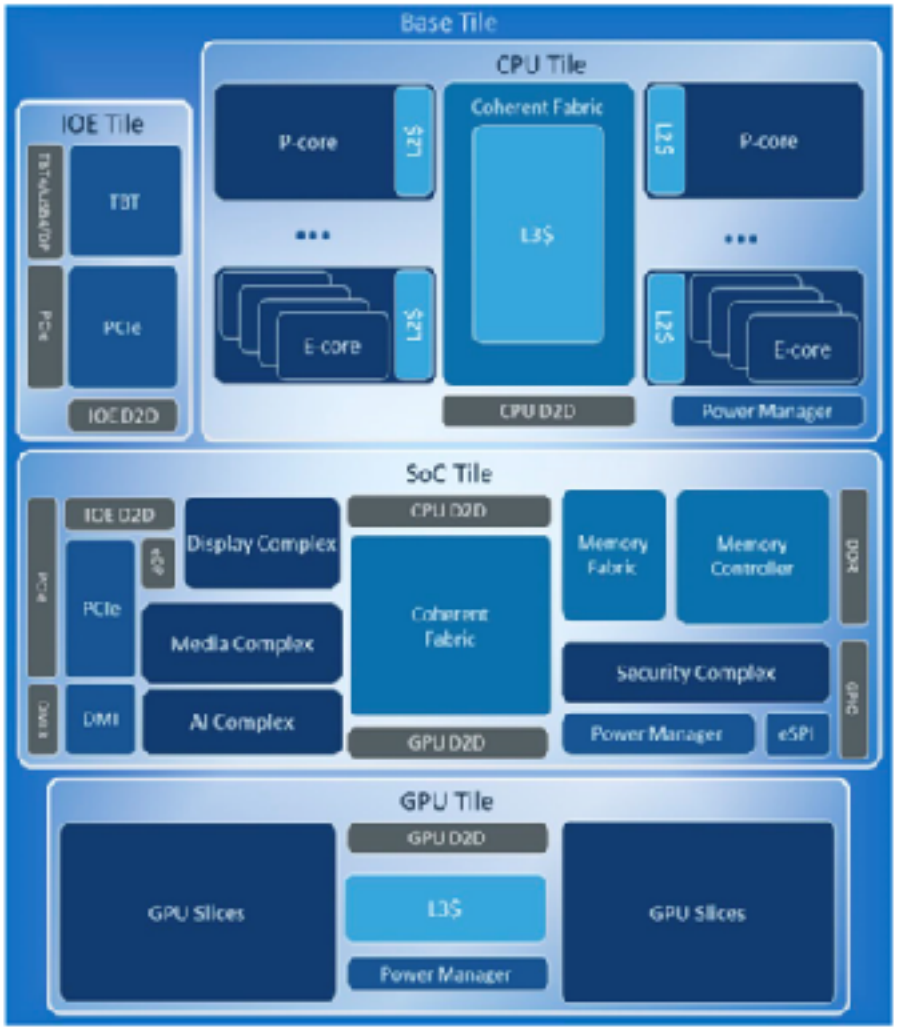
Concurrency in Hardware

Concurrency in Hardware



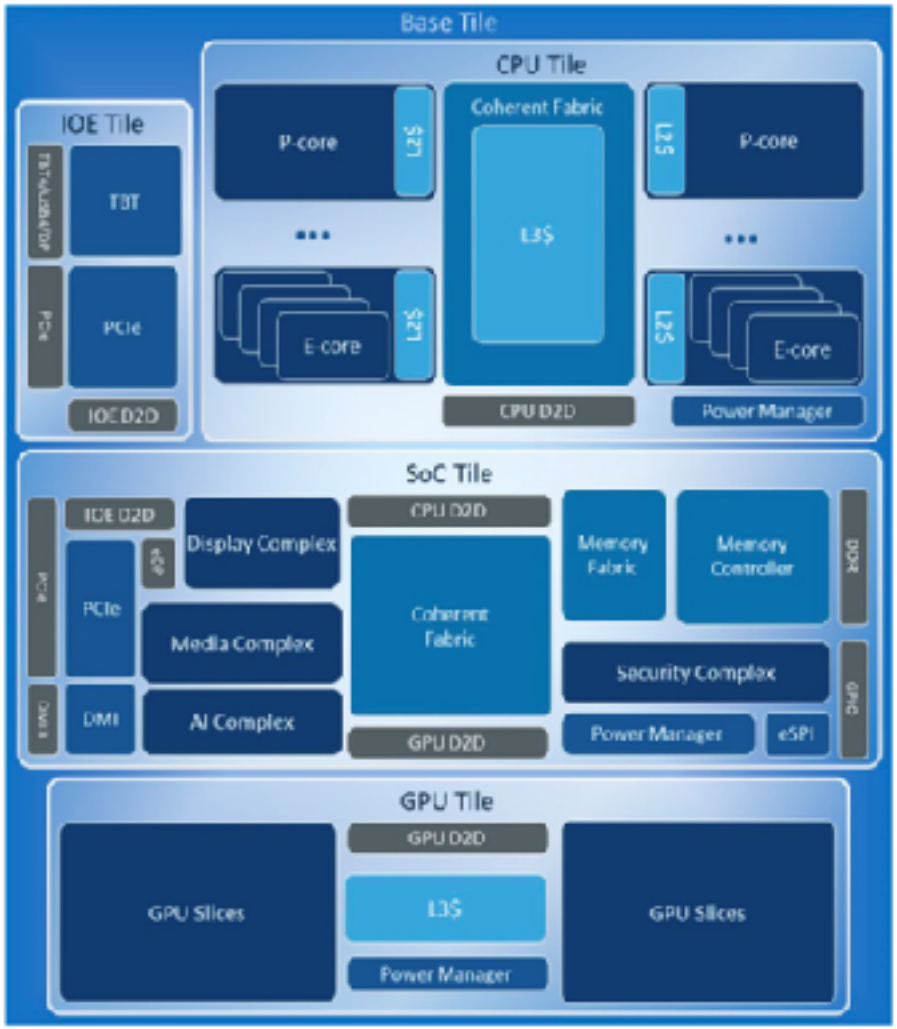
Concurrency in Hardware

- Observe:



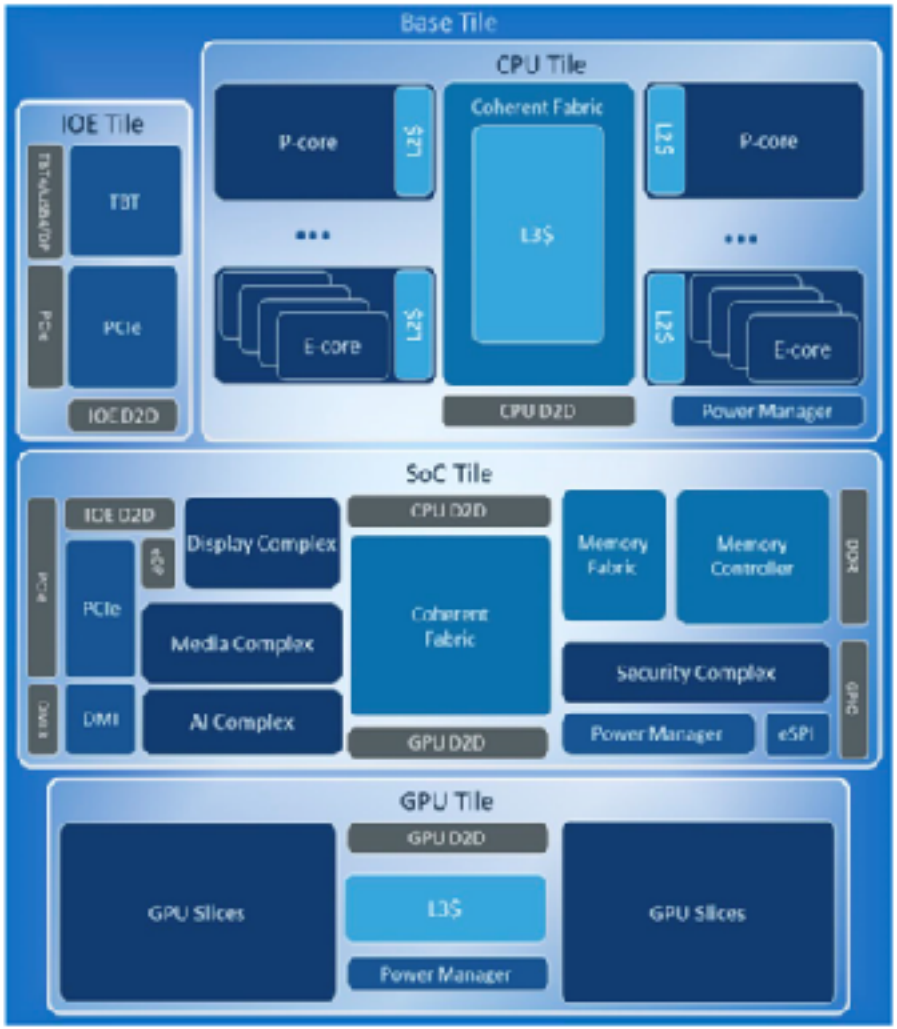
Concurrency in Hardware

- Observe:
  - **Multiple cores** (each with their own L2 cache)



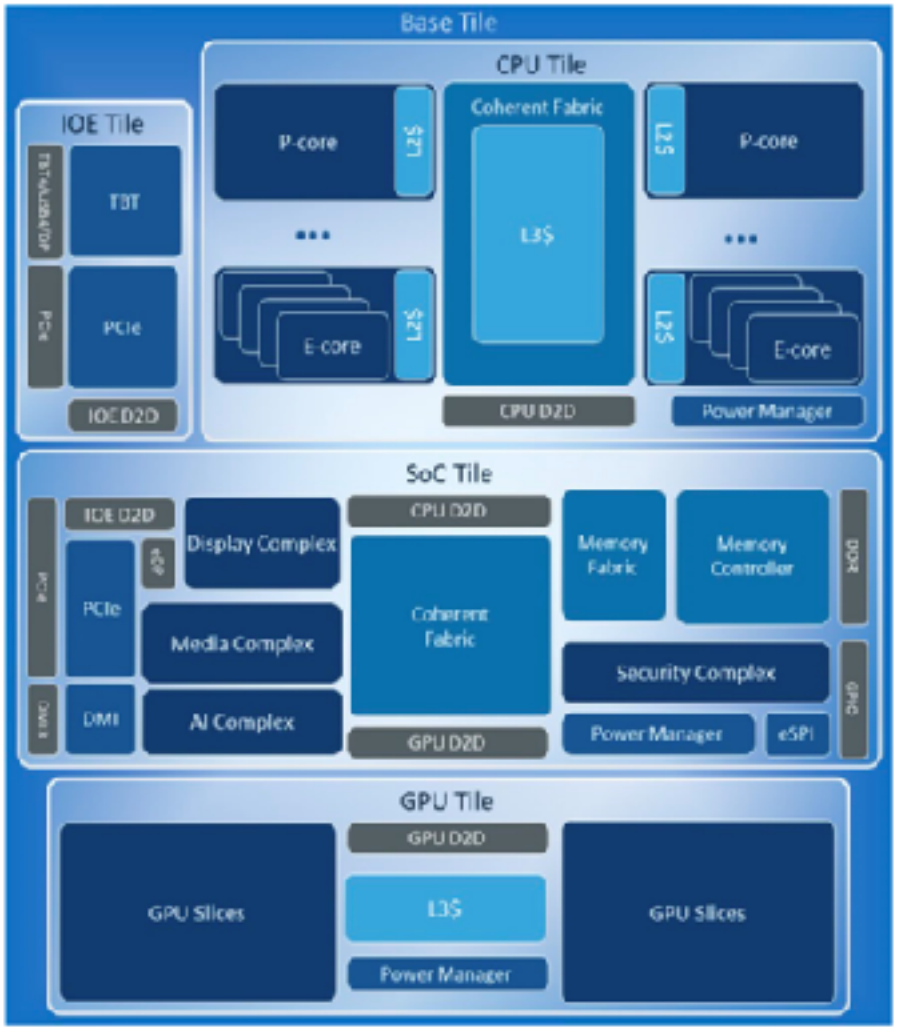
Concurrency in Hardware

- Observe:
  - **Multiple cores** (each with their own L2 cache)
    - Executing many tasks simultaneously



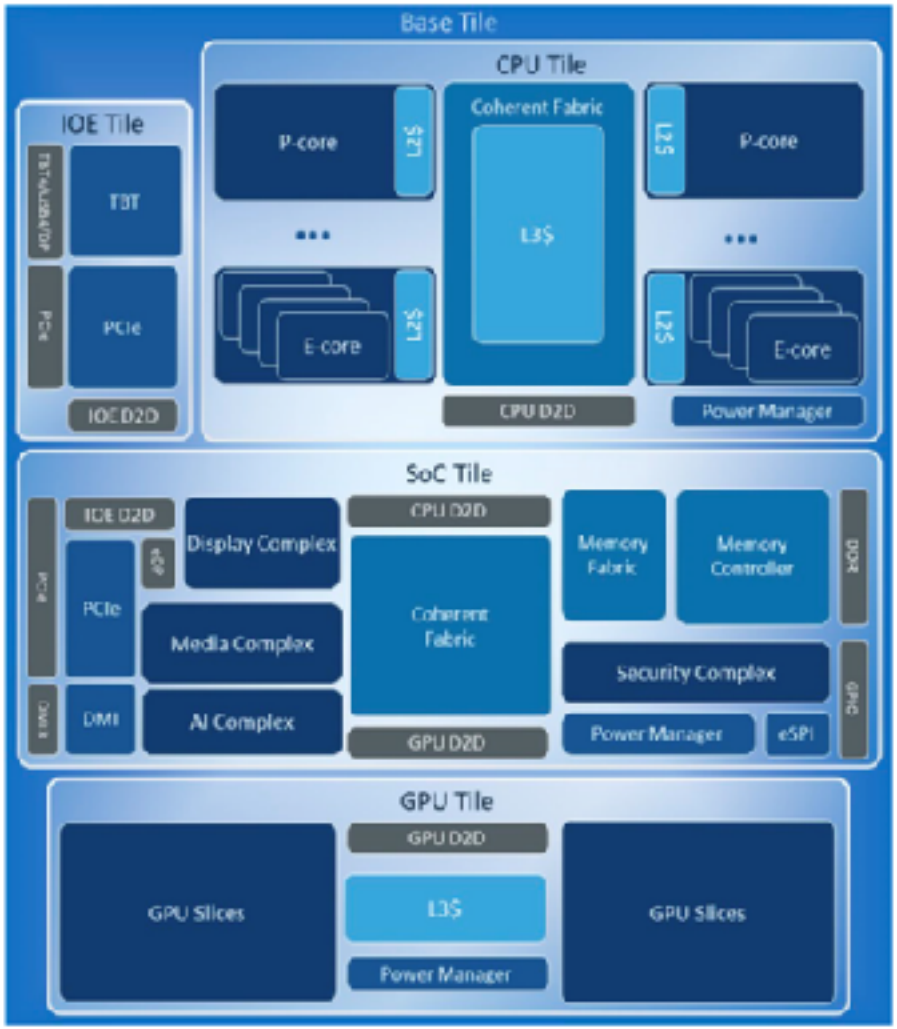
Concurrency in Hardware

- Observe:
  - **Multiple cores** (each with their own L2 cache)
    - Executing many tasks simultaneously
      - Usually work with slower clock but faster interconnects



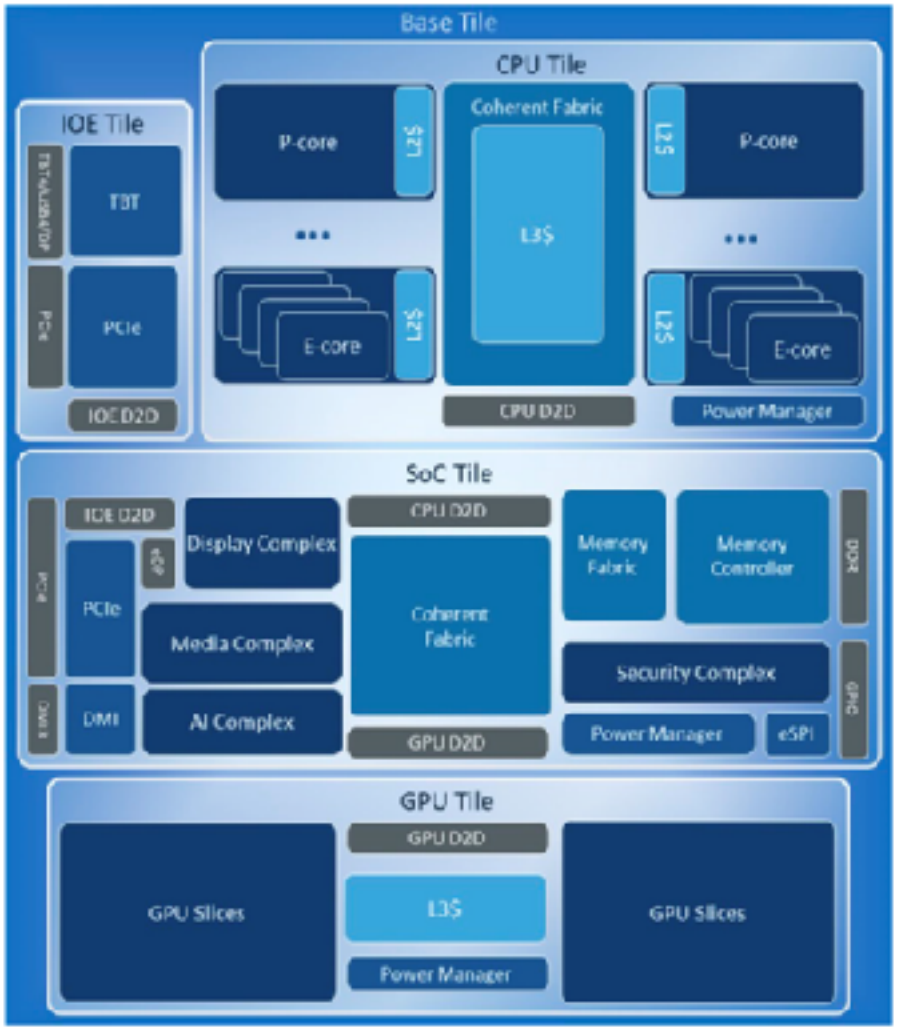
Concurrency in Hardware

- Observe:
  - **Multiple cores** (each with their own L2 cache)
    - Executing many tasks simultaneously
      - Usually work with slower clock but faster interconnects
    - All cores **share** L3 cache



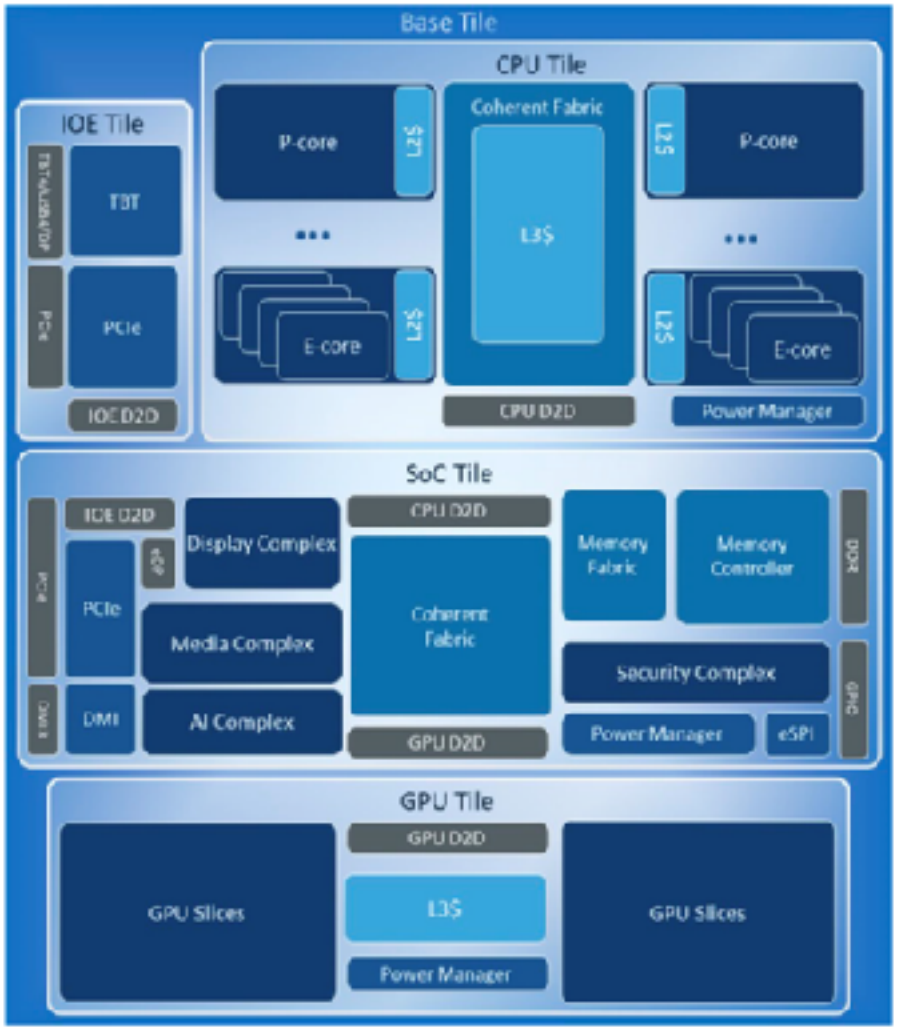
Concurrency in Hardware

- Observe:
  - **Multiple cores** (each with their own L2 cache)
    - Executing many tasks simultaneously
      - Usually work with slower clock but faster interconnects
    - All cores **share** L3 cache
  - Each core is **superscalar**



Concurrency in Hardware

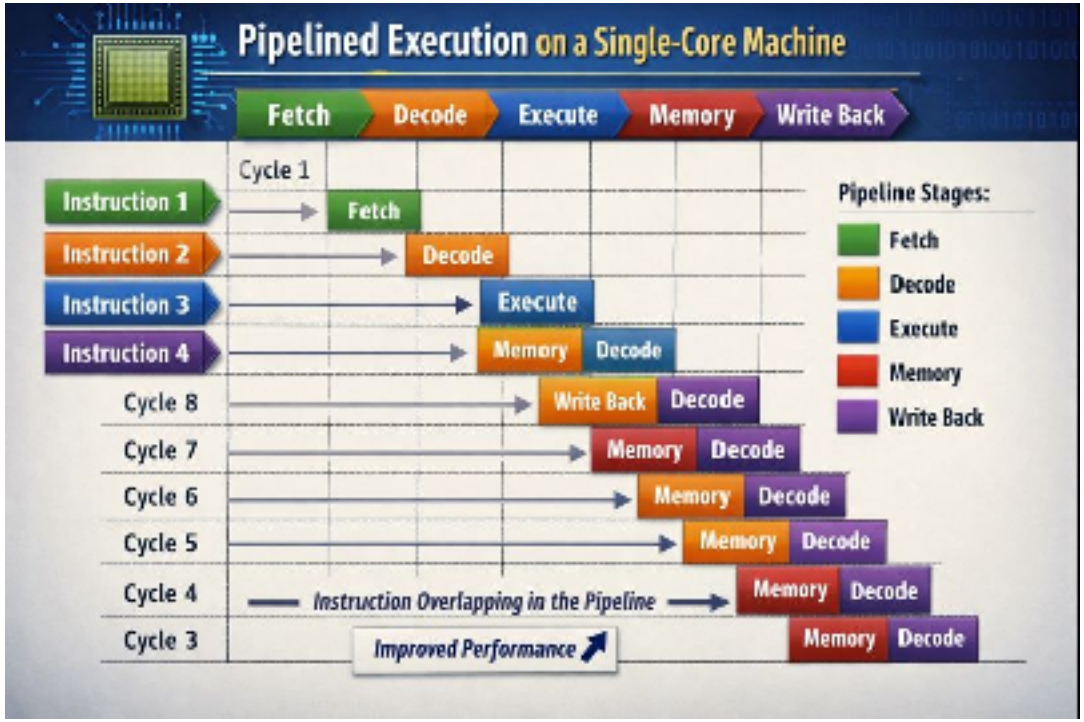
- Observe:
  - **Multiple cores** (each with their own L2 cache)
    - Executing many tasks simultaneously
      - Usually work with slower clock but faster interconnects
    - All cores **share** L3 cache
  - Each core is **superscalar**
    - What does it mean? We will have to look at how a core is set-up!



Concurrency in Hardware

Superscalar Architectures

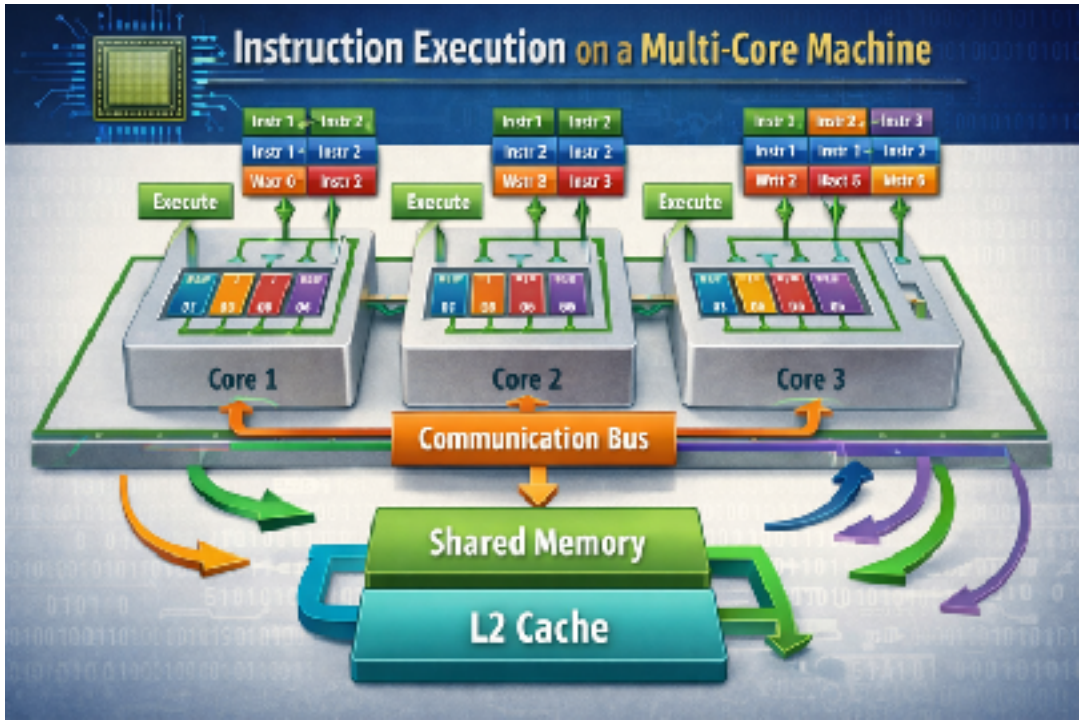
- **In single cores** — pipelining of Instruction execution
- Key take-away - **parallelism due to staged execution**
- **Superscalar:**
  - Multiple instruction streams



Concurrency in Hardware

MultiCore Architectures

- **Multiple cores**
- **Each core may be superscalar**
- **Complex:**
  - DMA/Memory Controllers
  - Memory **coherence & consistency**



Concurrency in Software (User Programs)

Concurrency in Software (User Programs)

```
void multi_transform(int *input, int *out1, int *out2, int *out3, int n)
{
  for (int i = 0; i < n; i++) {
     out1[i] = input[i] * 2; // Double
     out2[i] = input[i] + 100; // Shift
     out3[i] = input[i] * input[i]; // Square
   }
}
```

Concurrency in Software (User Programs)

- Can execute each loop instruction in parallel — Why?

```
void multi_transform(int *input, int *out1, int *out2, int *out3, int n)
{
  for (int i = 0; i < n; i++) {
     out1[i] = input[i] * 2; // Double
     out2[i] = input[i] + 100; // Shift
     out3[i] = input[i] * input[i]; // Square
   }
}
```

Concurrency in Software (User Programs)

- Can execute each loop instruction in parallel — Why?
  - Could also be **vectorized** on a single-core machine

```
void multi_transform(int *input, int *out1, int *out2, int *out3, int n)
{
  for (int i = 0; i < n; i++) {
    out1[i] = input[i] * 2; // Double
    out2[i] = input[i] + 100; // Shift
    out3[i] = input[i] * input[i]; // Square
  }
}
```