Microprocessor & Computer Architecture (µpCA)

UE19CS252

Dr. D. C. Kiran

Department of
Computer Science and Engineering

Microprocessor & Computer Architecture (µpCA)

Unit 5: Advanced Architecture

Dr. D. C. Kiran
Department of Computer Science and Engineering

Microprocessor & Computer Architecture (µpCA)

Syllabus

~~**Unit 1: Basic Processor Architecture and Design**~~

~~**Unit 2: Pipelined Processor and Design**~~

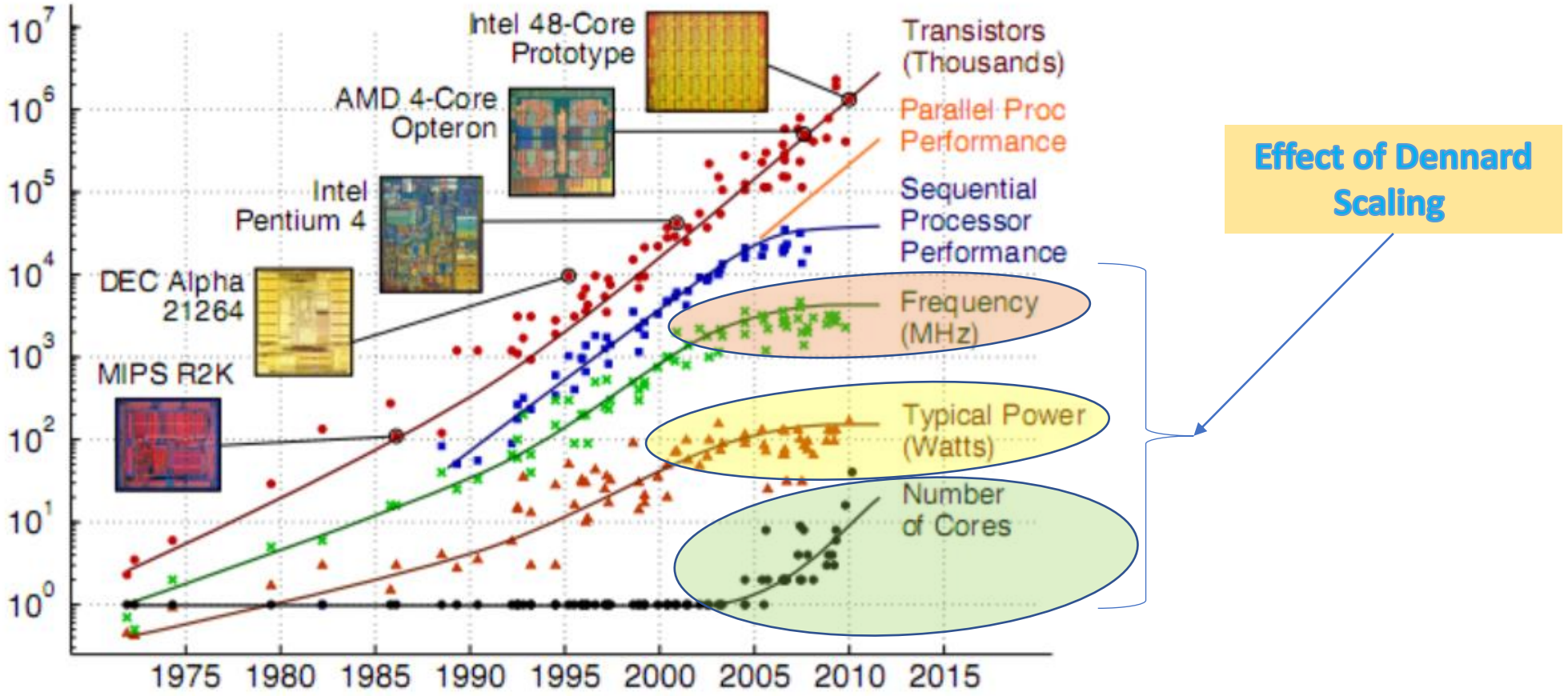
~~**Unit 3: Memory**~~

~~**Unit 4: Input/Output Device Design**~~

Unit 5: Advanced Architecture

- ~~Need for High Performance Computing~~
- ~~Classification of Parallel Architectures~~
- ~~Shared Memory Vs Distributed Memory Programming Paradigm.~~
- ~~Bird Eye View of Parallel Architectures~~
- ~~Parallel Processing~~
- ~~Amdahl's Law & Gustafson's Law~~
- **Multicore Processor**

Why Multicore Processor?



Effect of Dennard Scaling

Limitations Of Single Core

- **The Power Wall**
  - o Limit on the scaling of clock speeds.
  - o Ability to handle on-chip heat has reached a physical limit.

- **The Memory Wall**
  - o Need for bigger cache sizes.
  - o Memory access latency still not in line with processor speeds

- **The ILP Wall**

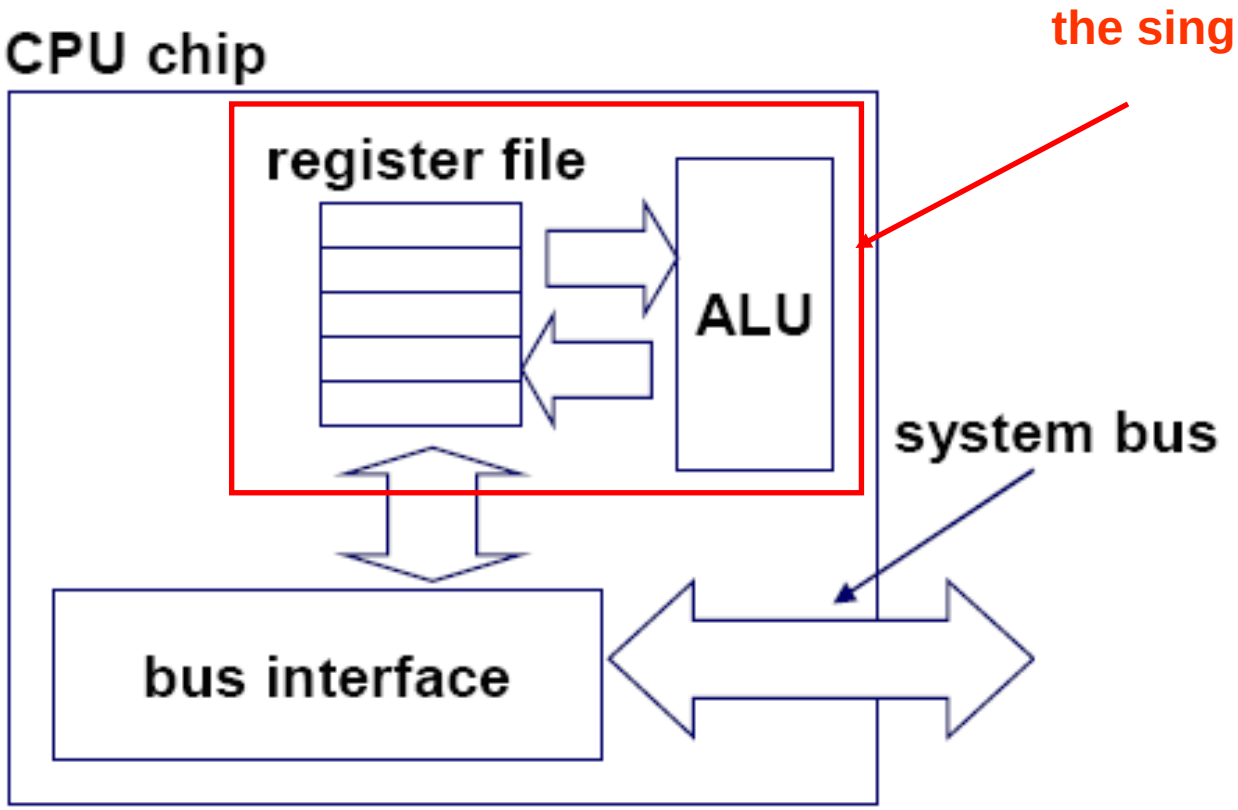
o Identifying Implicit Parallelism within the threads is limited in many Application

o Dependency

o Hardware Restrictions such as, Instruction Window Size (How many instructions can be fetched at a time).

Microprocessor & Computer Architecture (µpCA)

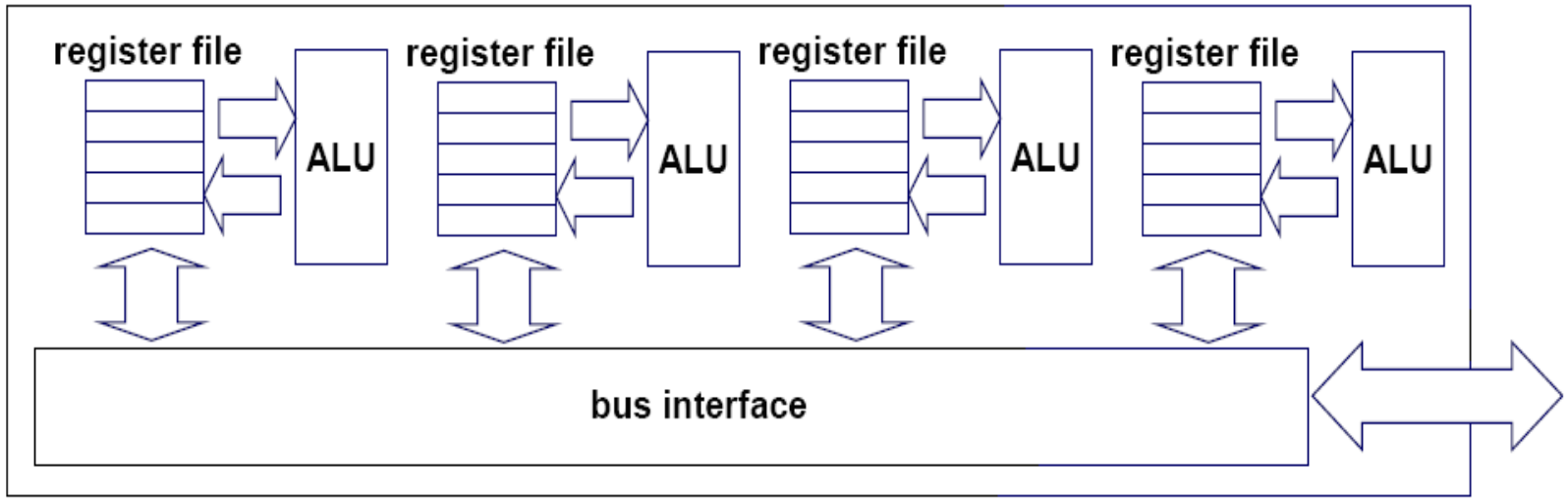
Single-core CPU chip



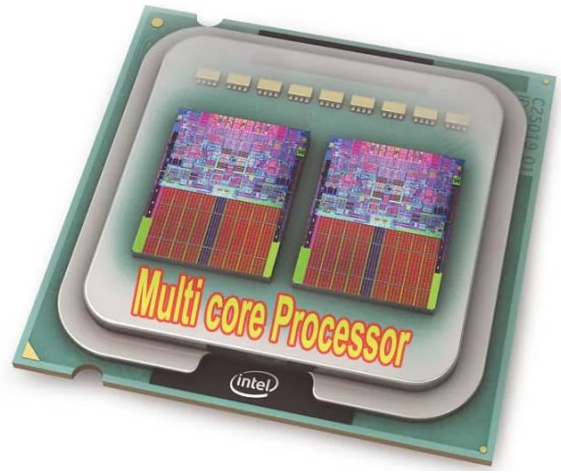
- Consists of
  - Execution state of a program
    - Registers, program counter, stack pointer, etc.
  - Interrupt logic
  - Execution Units
  - Cache
- Think of it as a single threaded processor
- True for Pipelined, Superscalar, VLIW

Microprocessor & Computer Architecture (µpCA)

Multi-core Architectures



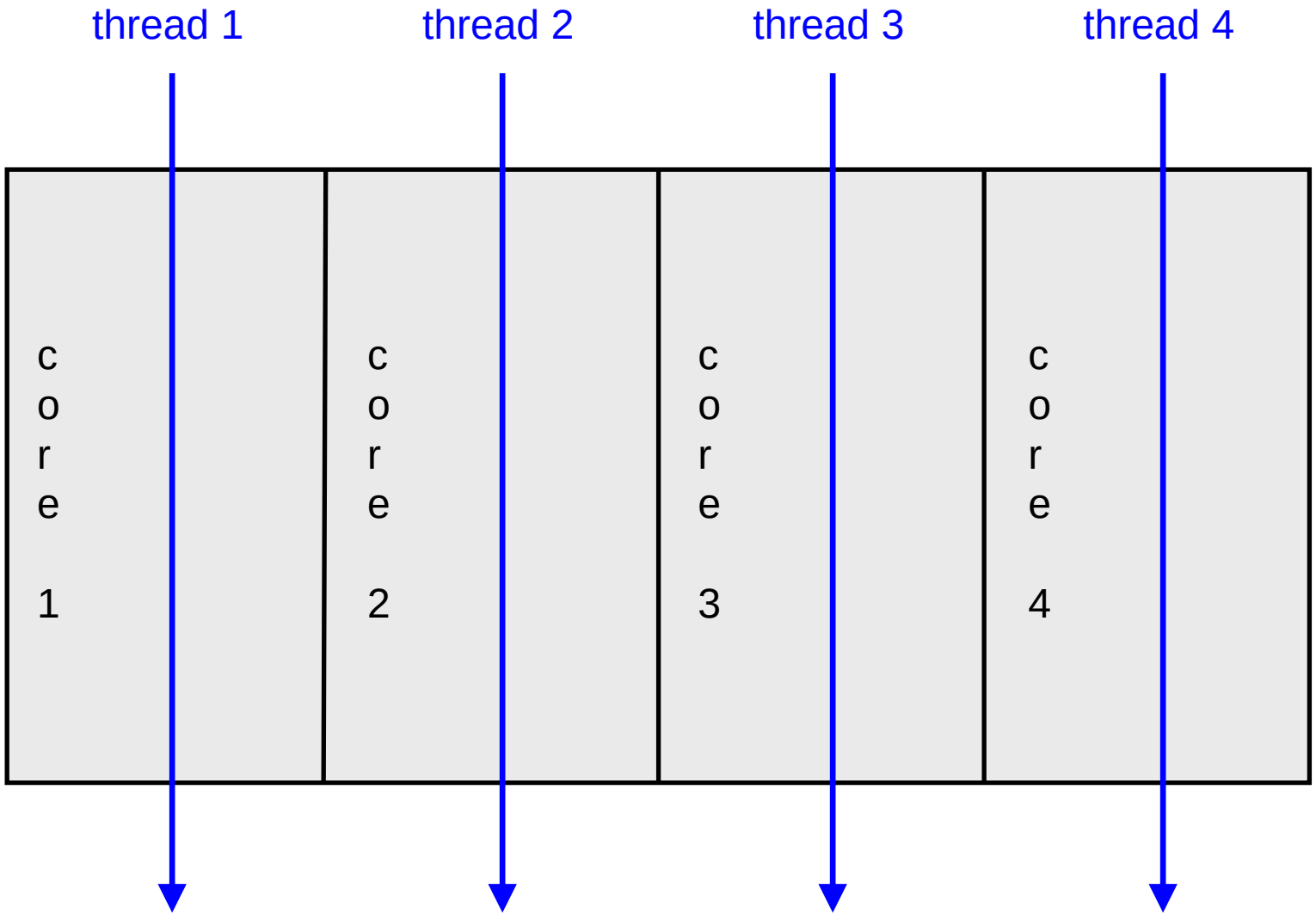
- Chip Multiprocessing (CMP)
- Multiple copies of the processor core
- Multi core with hyperthreading



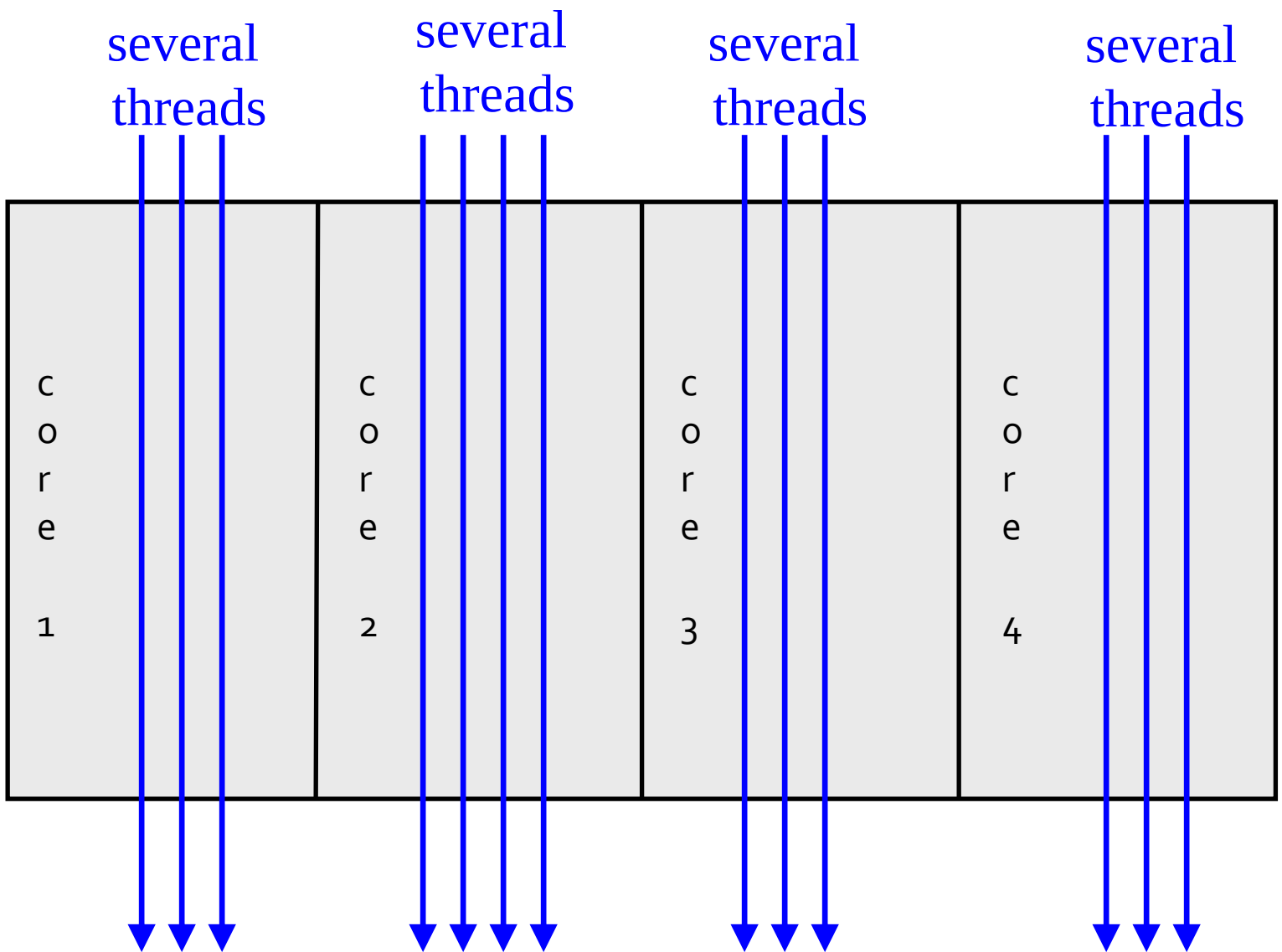
- Multi-core processors are MIMD:
  Different cores execute different threads (Multiple Instructions), operating on different parts of memory (Multiple Data).
- Multi-core is a shared memory multiprocessor:
  All cores share the same memory

Microprocessor & Computer Architecture (µpCA)

Multi-core Architectures



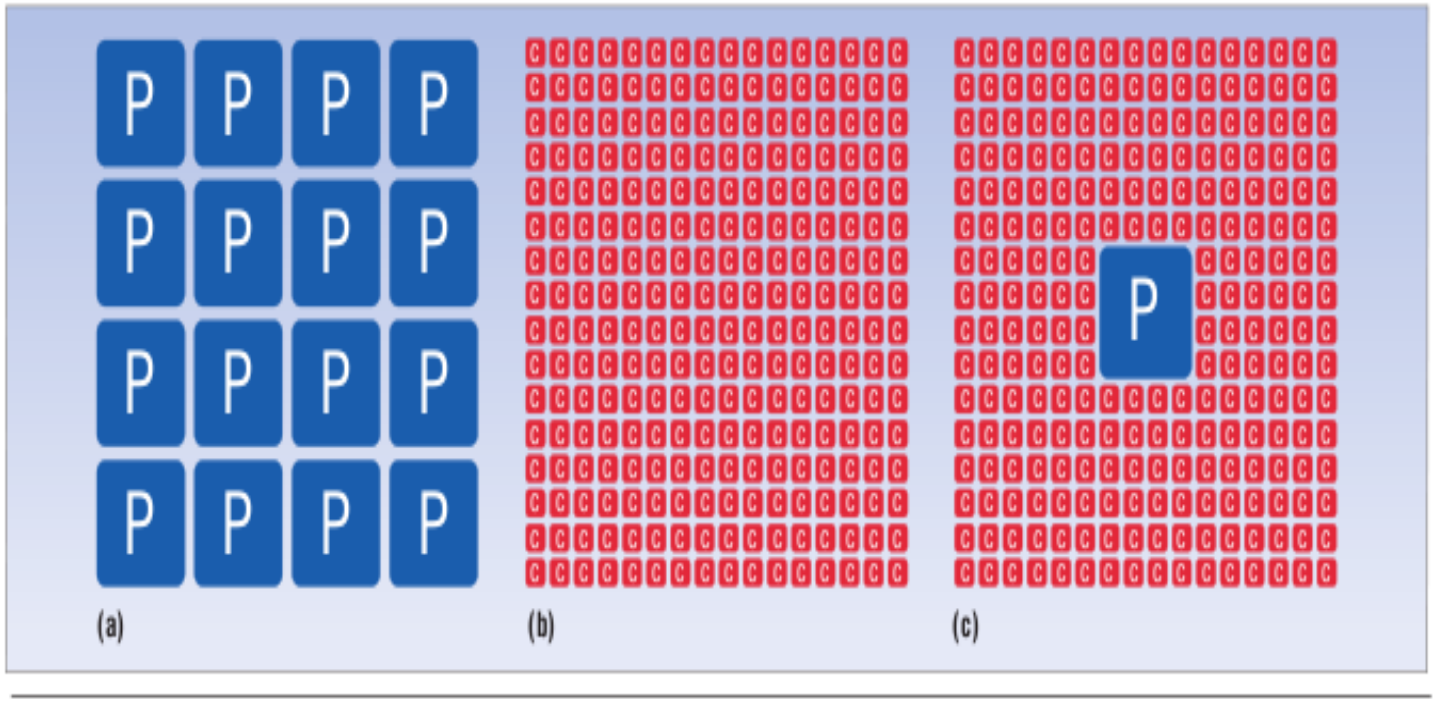
The cores run in parallel (like on a uniprocessor)



Microprocessor & Computer Architecture (µpCA)

Homogeneous : Multiple Core Architecture

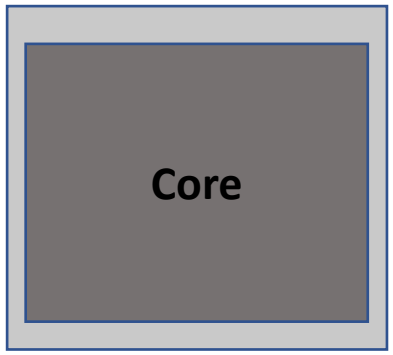
Identical processor cores support same Instruction set Architecture (ISA)



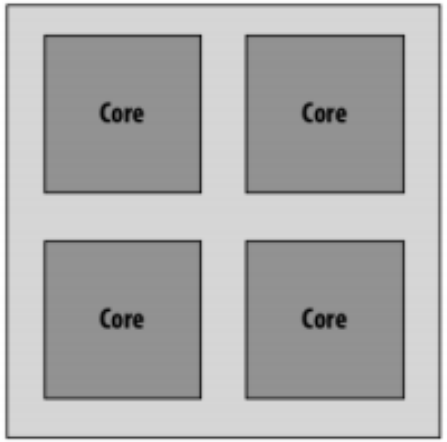
For example – MPC8641, Intel Core Duo

Microprocessor & Computer Architecture (µpCA)

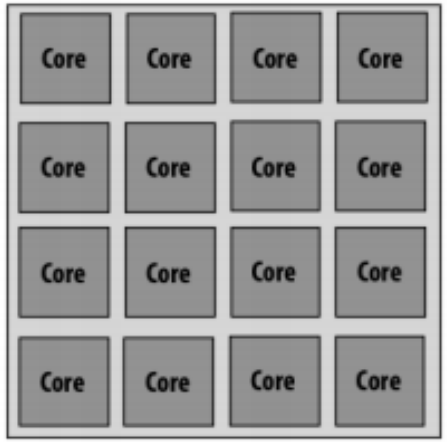
Resource vs Power



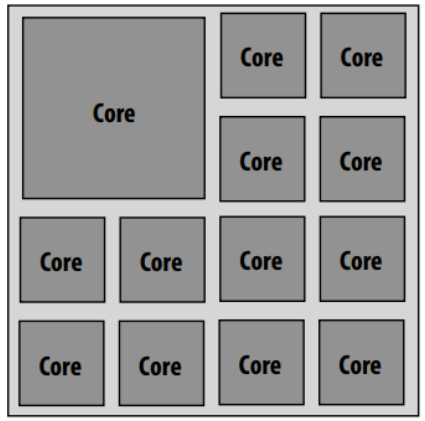
Unicore Processor



Processor A



Processor B



Processor C

P = Total processing resources used to make Uni-core processor (e.g., transistors on a chip).

R= Resources dedicated to each processing core.

Let N=16

- Unicore Processor is made up of ***P=R=16*** resources.
- Each core in Processor A is made of ***R= 4*** resources.
- Each core in Processor B is made of ***R=1*** Resources.
- 1 Core of Processor C is made of ***R=4*** Resources and other 12 cores are made up of ***R=1*** Resources

Microprocessor & Computer Architecture (µpCA)

Amdahl's Law for Multicore Processor

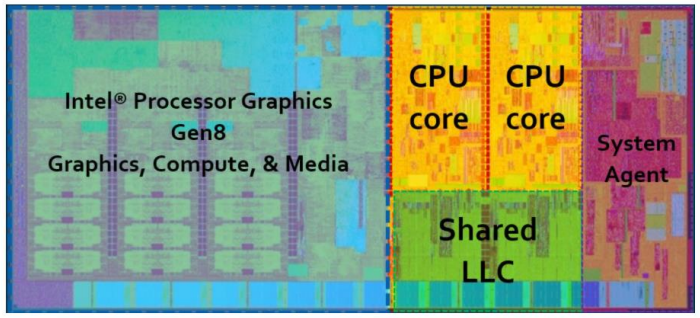
$$\text{Speedup} = \frac{1}{\frac{1 - F}{\text{Perf}(R)} + \frac{F * R}{\text{Perf}(R)*N}}$$

Note: This slide is only to realize that Speed UP measurement is different for Multicore processor over Multi Processor. This need lot of discussion which will ***not be done*** as part of course (MPCA UE18CSE252).

Reference 1 **Reference 2**

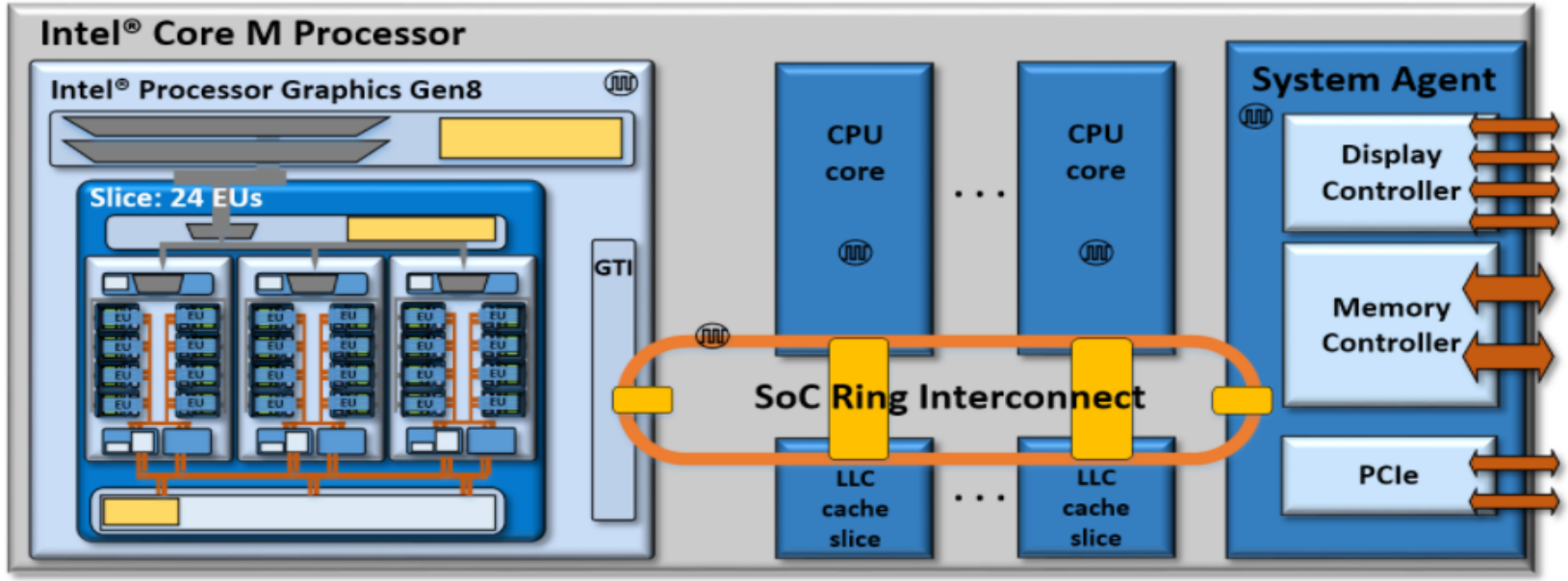
Heterogeneous : Multiple Core Architecture

Non-identical processor cores, support different Instruction Set Architecture (ISA).



An Intel® Core™ M Processor SoC

Reference

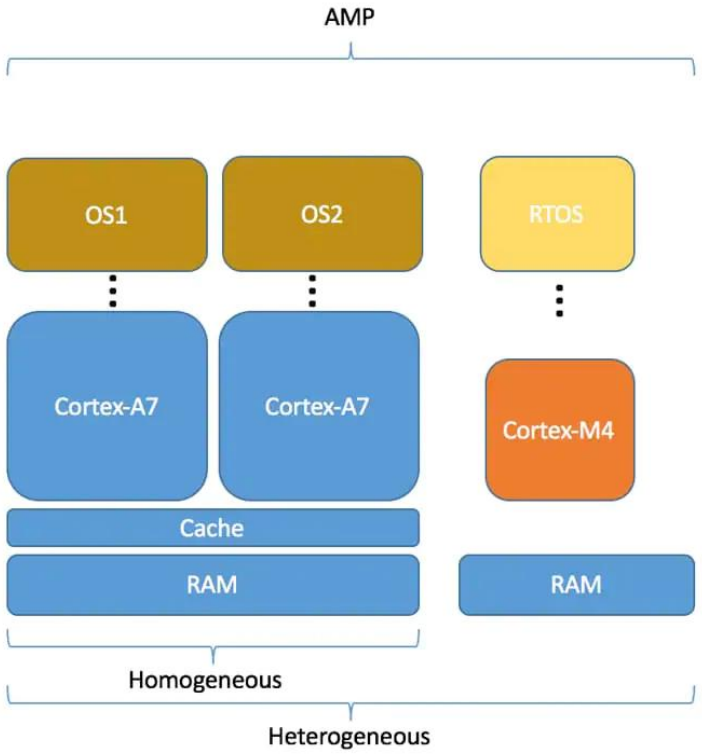


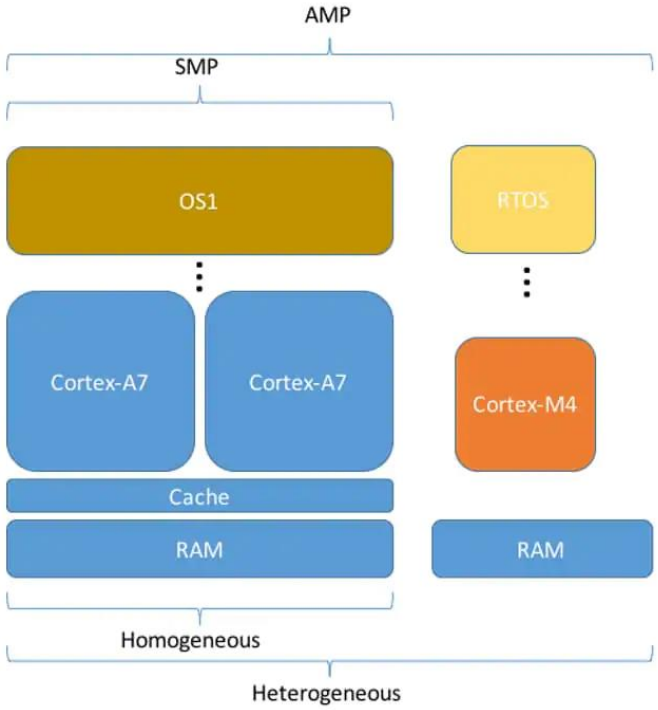
Why?

- Most Efficient Processors are Heterogeneous.
- Power Efficient (Green Computing)

Microprocessor & Computer Architecture (µpCA)

Heterogeneous ARM Processor





Reference

Microprocessor & Computer Architecture (µpCA)

Super Computers are Heterogeneous!

https://www.top500.org/lists/top500/2020/11/

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442,010.0	537,212.0	29,899
2	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148,600.0	200,794.9	10,096
3	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438
4	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway, NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371
5	**Selene** - NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100, Mellanox HDR Infiniband, Nvidia NVIDIA Corporation United States	555,520	63,460.0	79,215.0	2,646

Role of User

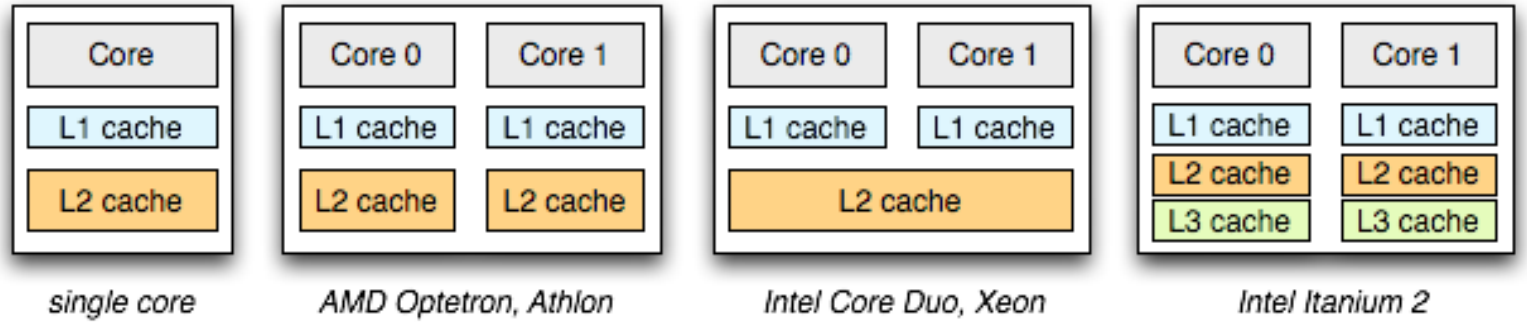
- Programmers must use threads or processes.
- Spread the workload across multiple cores.
- Write parallel algorithms.
- OS will map threads/processes to cores

Role of Operating System

- OS perceives each core as a separate processor
- OS scheduler maps threads/processes to different cores.
- Most major OS support multi-core today

Microprocessor & Computer Architecture (µpCA)

Role of Memory



Need to place Cache, on-Chip to Reduce bandwidth and increase Latency :- Occupy Processor space and Power

Cache Memory in Multicore Reference **1**

Memory Contention: Communication and Computation uses same memory bandwidth. Memory Contention Reference **2**

Cache Coherence: A program running on multiple processors will normally have copies of the same data in several caches. The protocols to maintain coherence for multiple processors are called **cache coherence protocols**. **(Write Policies)**

False Sharing in the shared Cache: If two or more processors are writing data to different portions of the same cache line, then a lot of cache and bus traffic might result for effectively invalidating or updating every cached copy of the old line on other processors. This is called *"false sharing"* or also *"CPU cache line interference"*.

Microprocessor & Computer Architecture (µpCA)

False Sharing

Problem: Updating the Array element by 2

5	4	6	3	12	15	7	9	11	1

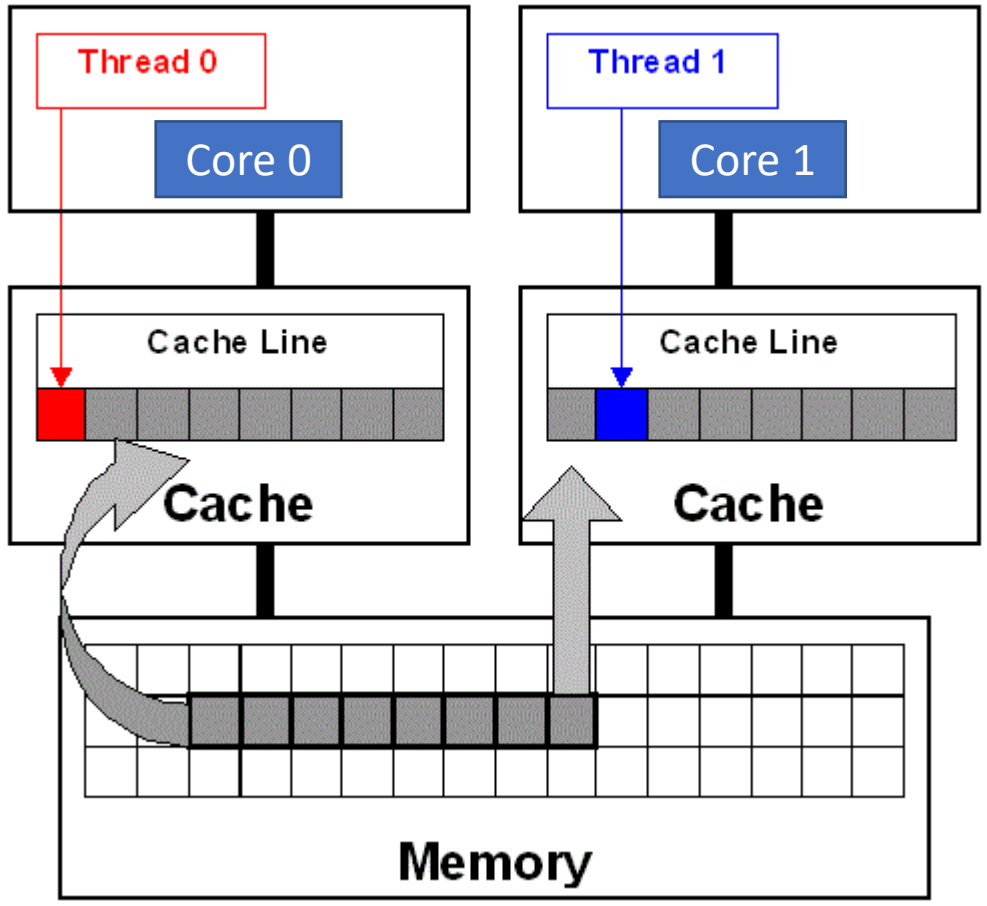
Parallel Solution: Split the array int0 two half to create 2 Threads, Thread 0 and Thread 1

However, two copy of entire array is copied into cache of each core.

False Sharing:

- Core0 update 1st half of the array.
- Core1 update 2nd half of the array.
- The cache line which contain the array is Invalidated, because they are in the same cache line, which lead to unnecessary cache update to maintain cache coherence.

False sharing will have serious effect on performance☹



Cache Coherence False Sharing Reference 3

References

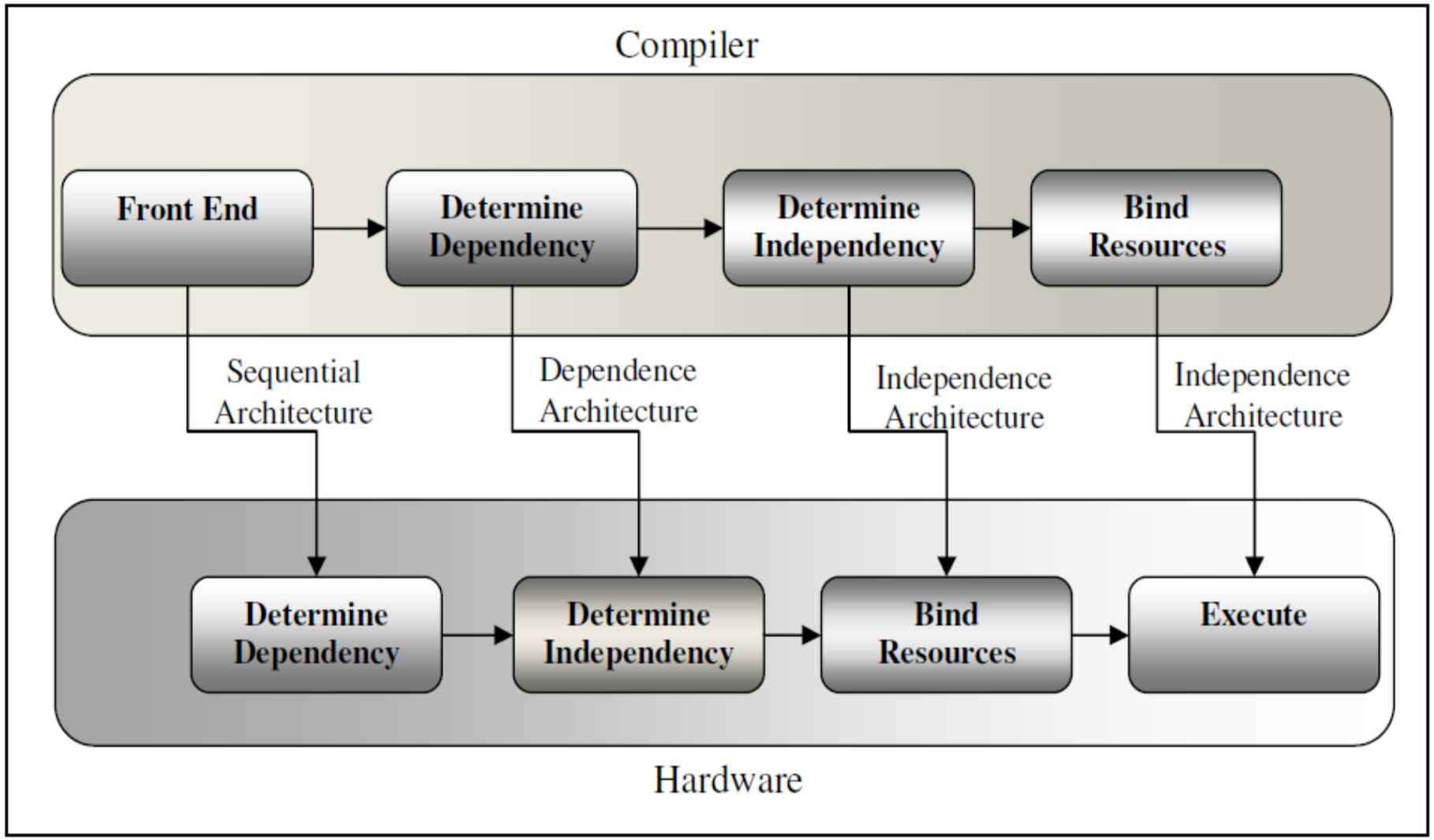
http://cse.unl.edu/~seth/990/Pubs/multicore-review.pdf

https://www.jstage.jst.go.jp/article/ipsjtsldm/8/0/8_51/_pdf/-char/en

https://www.cdac.in/index.aspx?id=pdf_hypack-arm-overview

http://meseec.ce.rit.edu/551-projects/fall2013/3-2.pdf

Conclusion: Parallel Architecture, Compiler, User, OS & Parallelism



THANK YOU

Dr. D. C. Kiran
Department of Computer Science and Engineering

dckiran@pes.edu

9829935135