MAS DSE 230
Scalable Analytics

Computer Systems & Parallelism

Mai H. Nguyen

COMPUTER SYSTEMS & PARALLELISM

- Basics of Computer Systems
  - Hardware & Software
  - Computer Instruction Cycle
  - Memory Hierarchy
  - Virtualization
- Parallelism
  - Parallel Processing
  - Task & Data Parallelism
  - Speedup

COMPUTER SYSTEMS & PARALLELISM

- Basics of Computer Systems
  - Hardware & Software
  - Computer Instruction Cycle
  - Memory Hierarchy
  - Virtualization
- Parallelism
  - Parallel Processing
  - Task & Data Parallelism
  - Speedup

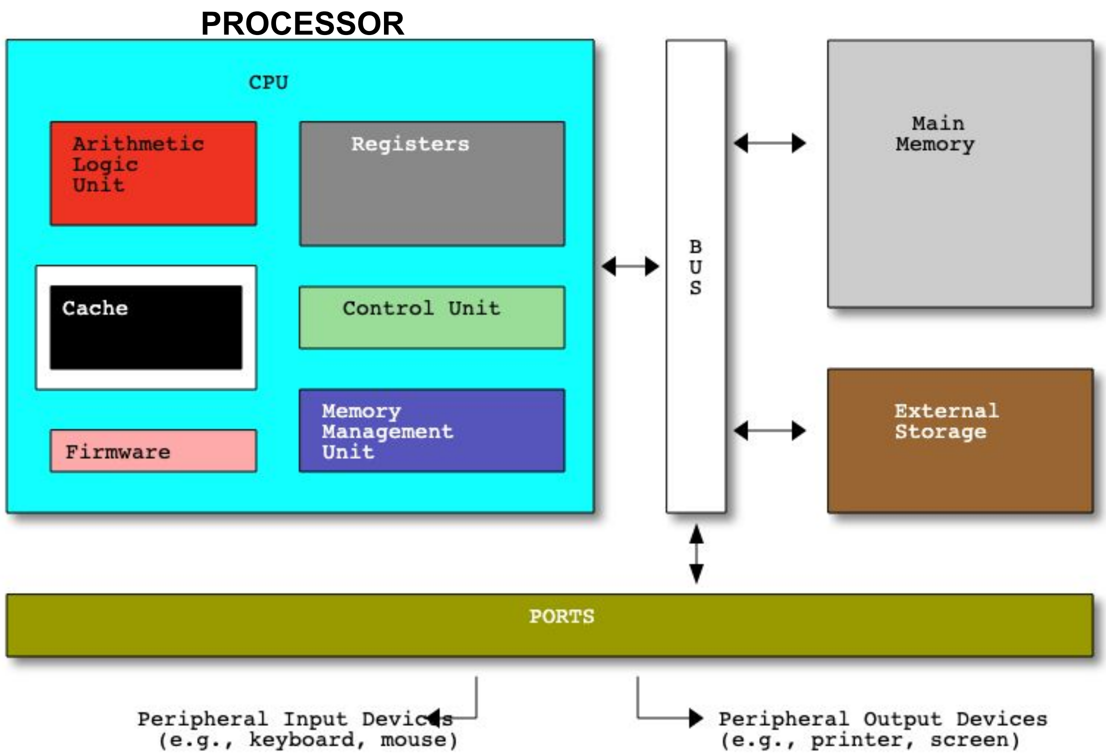
COMPUTER HARDWARE & SOFTWARE

Hardware:
Physical parts of computer

Software:
Programs (instructions) to perform tasks on computer

KEY HARDWARE COMPONENTS

Processor
Executes instructions as specified in program to manipulate data



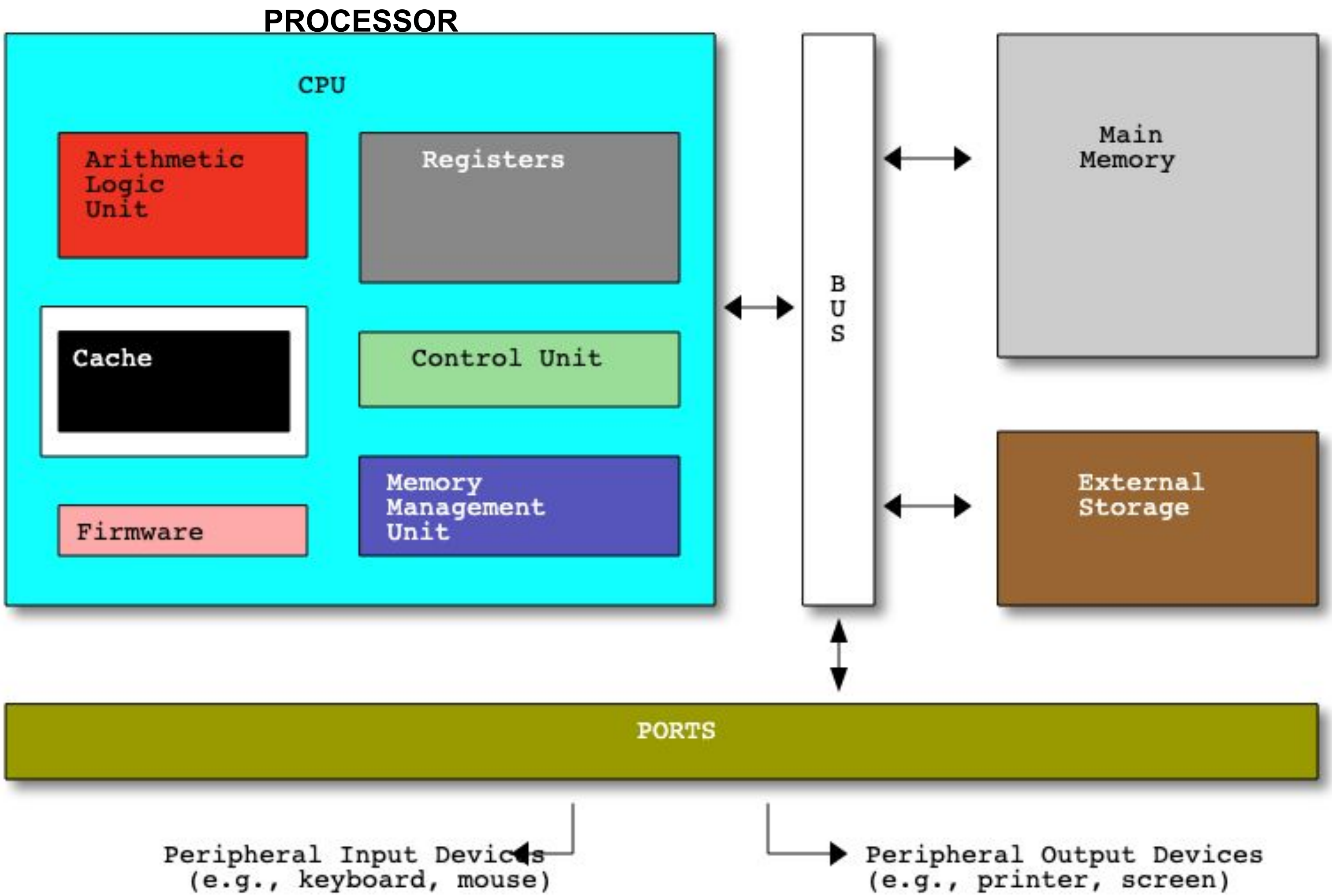
https://www.refsmmat.com/courses/751/notes/architecture.html

Main Memory
Stores data and programs for fast access

External Storage
Stores data and programs; slower but more persistent than Main Memory

Network Interface Controller
Sends/Retrieves data over network to/from interconnected computers/devices

KEY HARDWARE COMPONENTS IN DETAIL

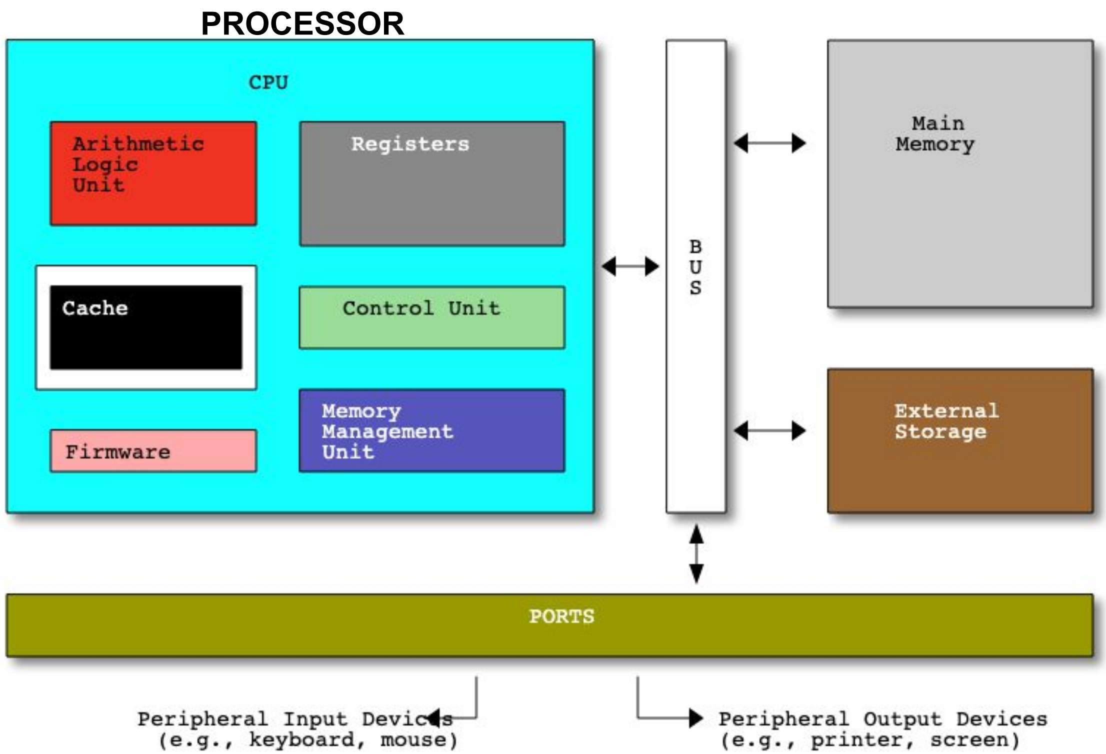


MAIN TYPES OF COMPUTER SOFTWARE

- Firmware
  - Specially designed for device to help control functionality of device
  - e.g.: TV, remote control, appliances
- System Software
  - Controls and manages operations of computer hardware
  - Operating System: Manages computer's resources to enable application software to execute efficiently
    - e.g.: Linux, MacOS, Windows
- Application Software
  - Implements end user applications
  - e.g.: email, spreadsheet, Web browser, communications

KEY HARDWARE COMPONENTS

Processor
Executes instructions as specified in program to manipulate data



https://www.refsmmat.com/courses/751/notes/architecture.html

Main Memory
Stores data and programs for fast access

External Storage
Stores data and programs; slower but more persistent than Main Memory

Network Interface Controller
Sends/Retrieves data over network to/from interconnected computers/devices

MEMORY HIERARCHY

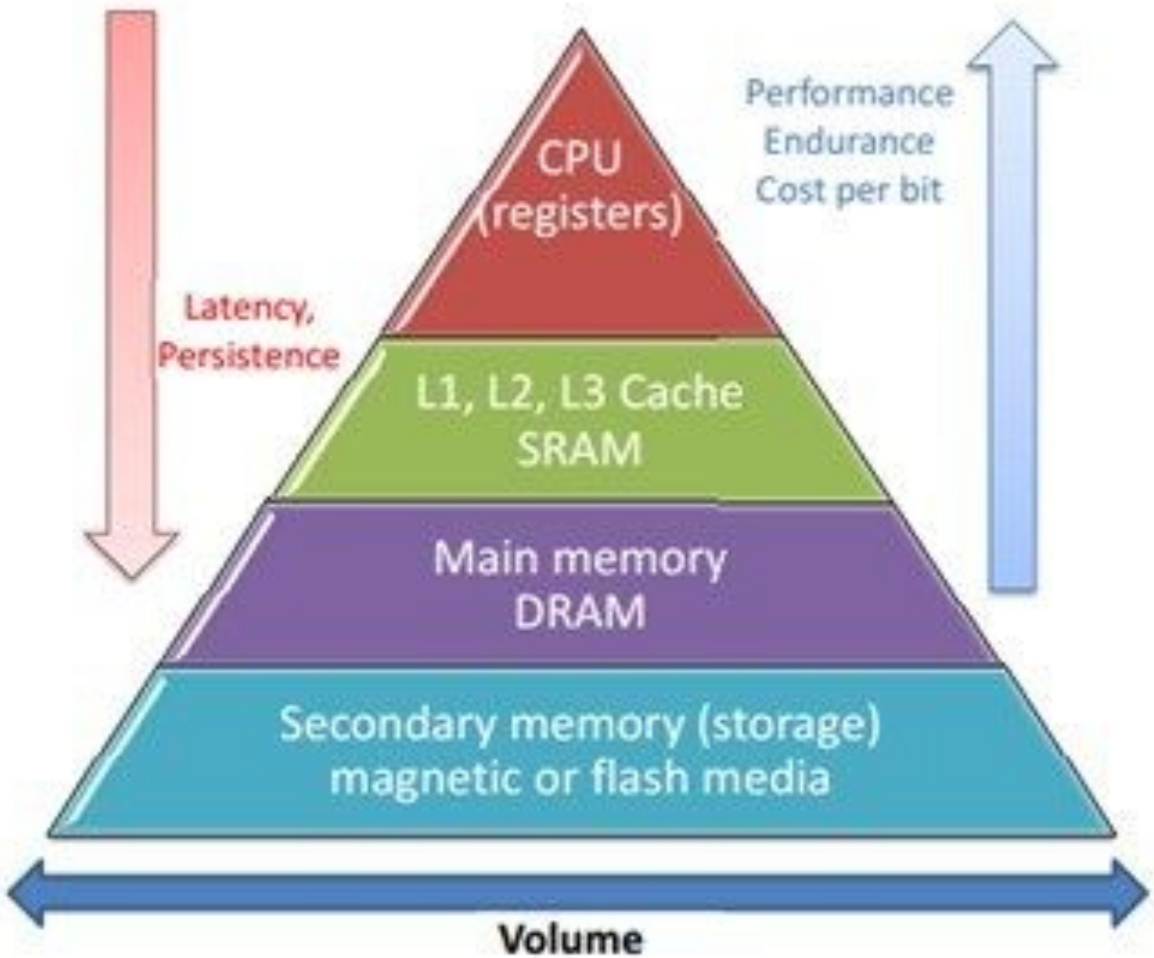
Access Speed

~100GB/s

~10GB/s

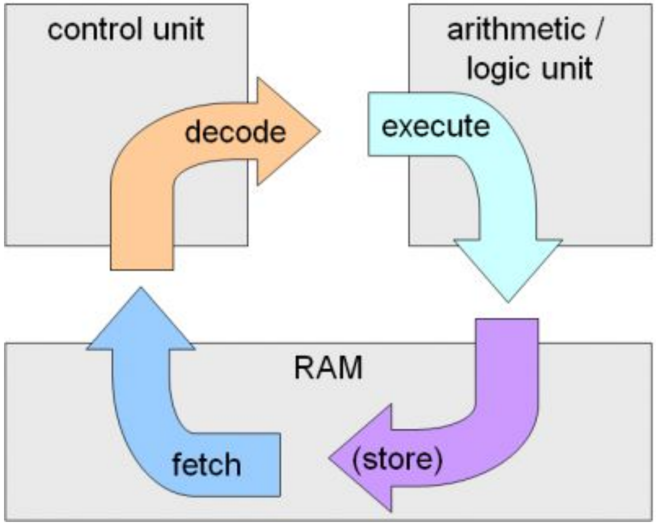
~GB/s

~200MB/s

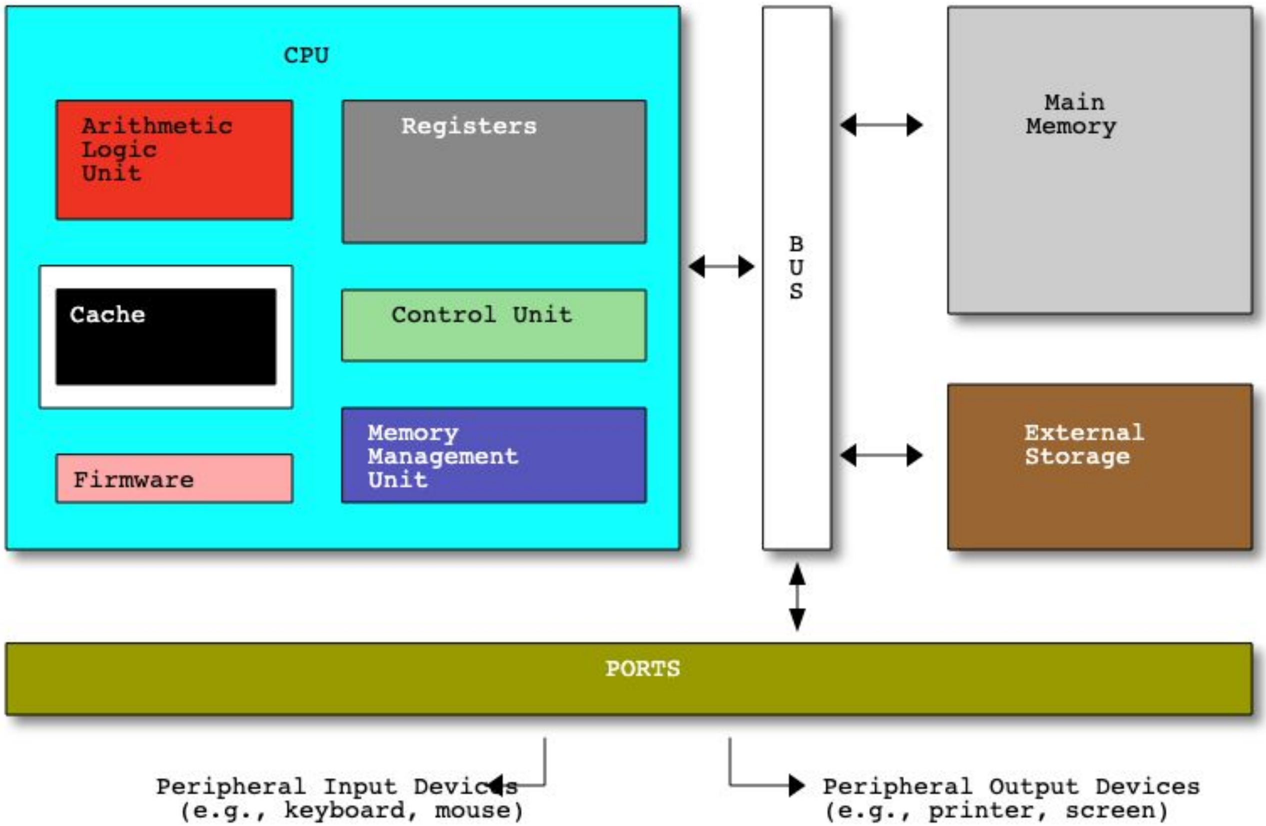


https://www.researchgate.net/figure/The-memory-hierarchy-pyramid_fig1_319529366

COMPUTER INSTRUCTION CYCLE



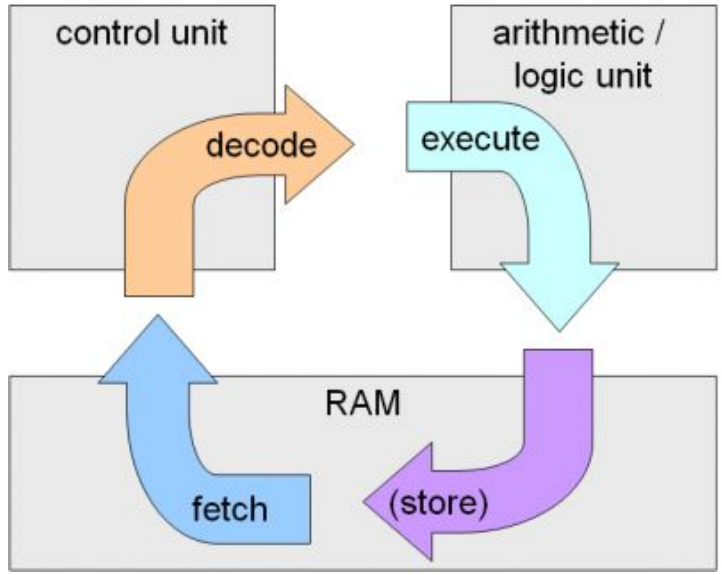
PROCESSOR



https://www.refsmmat.com/courses/751/notes/architecture.html

- Fetch program from external storage to memory
- Fetch instruction from memory
- Decode instruction
- Read data from memory
- Prefetch data into cache
- Load data into registers
- Execute instruction
- Store results in memory
- Write results out to disk

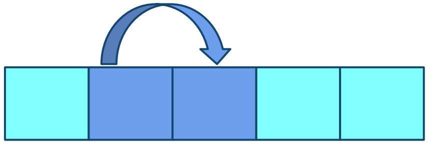
COMPUTER INSTRUCTION CYCLE



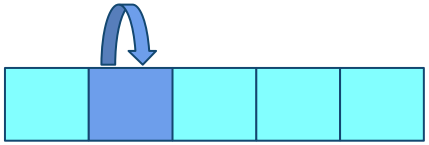
- Modern processors can run millions of instructions per second
- But when data has to be fetched from memory, CU and ALU are idle -> **memory stall**
- Careful use of different levels of memory is essential for overall system performance
  - Want to maximize cache hits to optimize processor utilization

LOCALITY OF REFERENCE

- **Locality of Reference**
  - Many programs tend to access memory locations in a somewhat predictable manner
  - 2 types: spatial and temporal
- **Spatial** locality (locality in space)
  - Items with nearby locations tend to be referenced close together in time



- **Temporal** locality (locality in time)
  - Recently referenced items are likely to be referenced again in the near future



LOCALITY OF REFERENCE

```
sum = 0
for i in range (0,len(ary)):
    sum = sum + ary[i]
print (sum)
```

- Data references
  - Reference array elements in succession — *spatial locality*
  - Reference variable ‘sum’ each iteration — *temporal locality*
- Instruction references
  - Reference instructions in loop — *spatial locality*
  - Cycle through loop repeatedly — *temporal locality*

LOCALITY OF REFERENCE

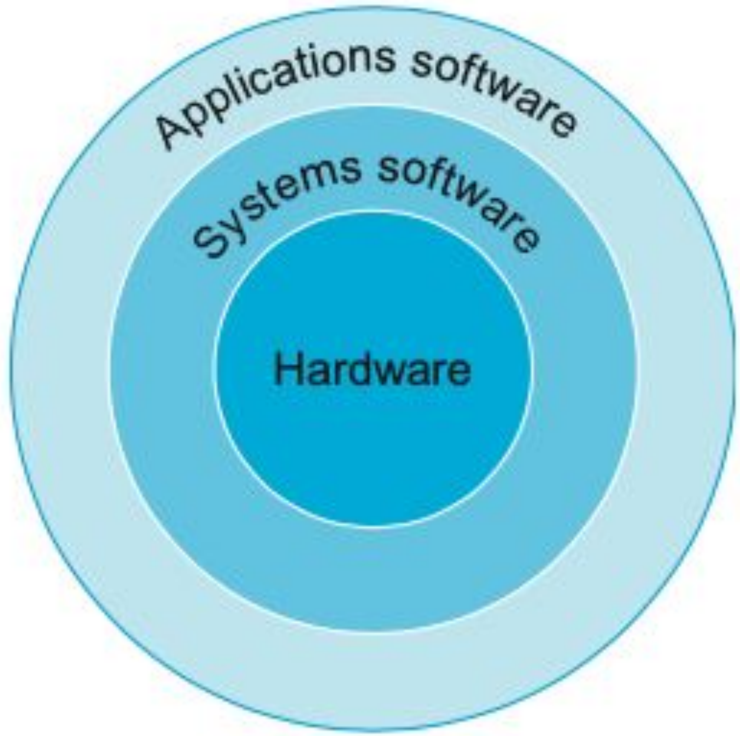
- **Locality of Reference:** Many programs tend to access memory locations in a somewhat predictable manner
  - **Spatial:** Nearby locations will be accessed soon
  - **Temporal:** Same locations accessed again soon
- Locality can be exploited to reduce runtimes using **caching** and/or **prefetching** across all levels in memory hierarchy

MEMORY MANAGEMENT

- **Caching**
  - Buffering instructions and/or data from a lower level at a higher level to exploit locality
- **Prefetching**
  - Preemptively retrieving data from addresses not explicitly requested yet by program
- **Hit**
  - Data needed is already available at higher level
- **Miss**
  - Data needed for program is not yet available at a higher level; need to get it from lower level
- **Replacement Policy**
  - When new data needs to be loaded to higher level, which old data to evict to make room? Many policies exist with different properties

- **Main idea:** Raise cache hits => Reduce memory stalls => Increase performance

OPERATING SYSTEM



Patterson & Hennessy, *Computer Organization and Design*

Systems Software

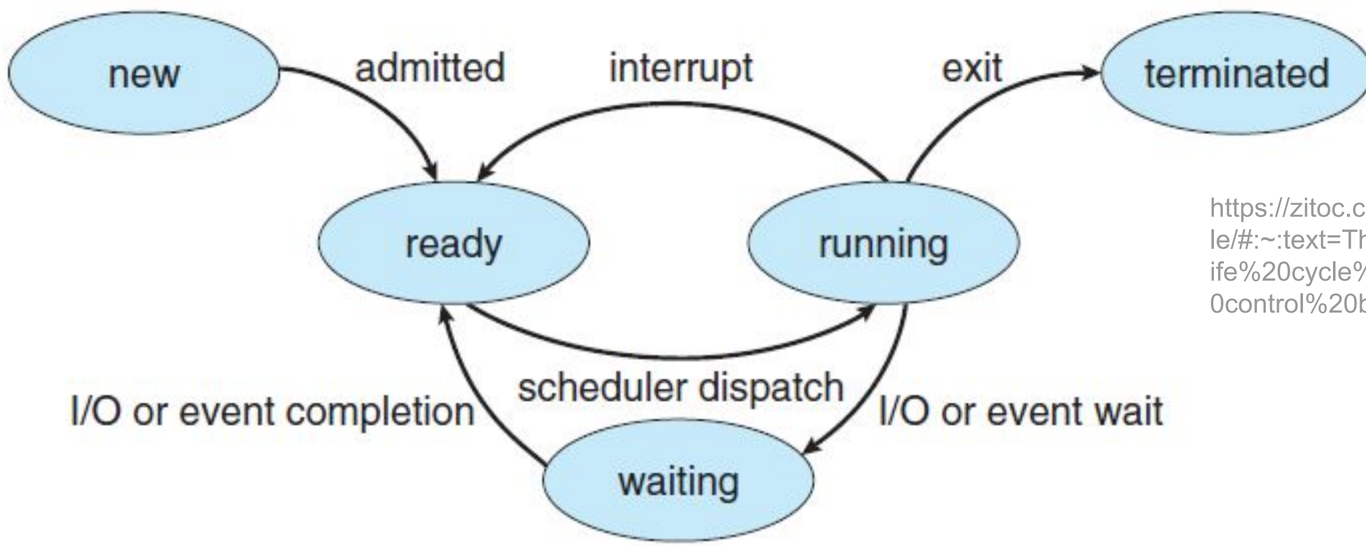
- Operating System
- Compilers
- Assemblers
- Utilities

OPERATING SYSTEM

- Operating System (OS)
  - Systems software that manages hardware and software resources of computer system
  - Provides consistent way for application software to use computer hardware effectively, efficiently, and securely
- Functionality provided
  - Process management
  - Main memory management
  - File management
  - Networking
  - Device management
- Common OS
  - MacOS, Windows, Linux

PROCESS MANAGEMENT BY OS

- Steps taken by OS run process
  - Assign Process ID to process
  - Allocate address space for process. Load code and static data.
  - Start process
  - Update process' state to Ready
  - When process is in Running state, OS hands off control to process
  - Process is put in Waiting/Blocked state when I/O is needed
  - Deletes process' address space when process is Terminated

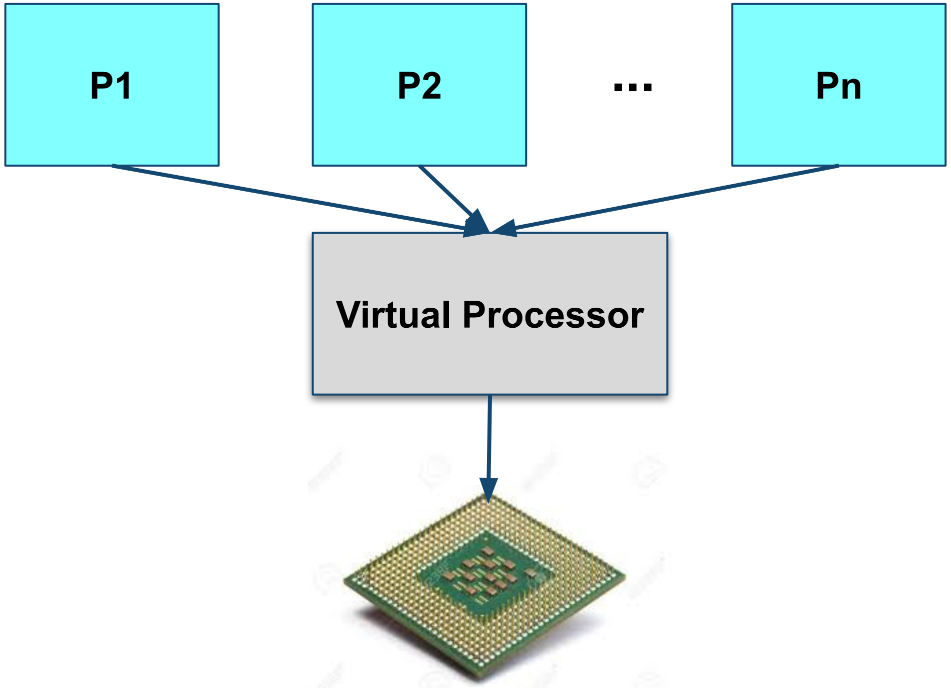


https://zitoc.com/process-life-cycle/#:~:text=The%20process%20life%20cycle%20can,process%20control%20block%20(PCB).

VIRTUALIZATION

- **Virtualization**
  - Using software to provide abstraction of hardware component
- **Process Virtualization**
  - Operating system (OS) time-shares processor among different processes
  - Allows for multiple processes to run on single physical processor at same time
  - OS performs **context switching** to allow different process to run
  - OS creates virtual version of processor

PROCESS VIRTUALIZATION



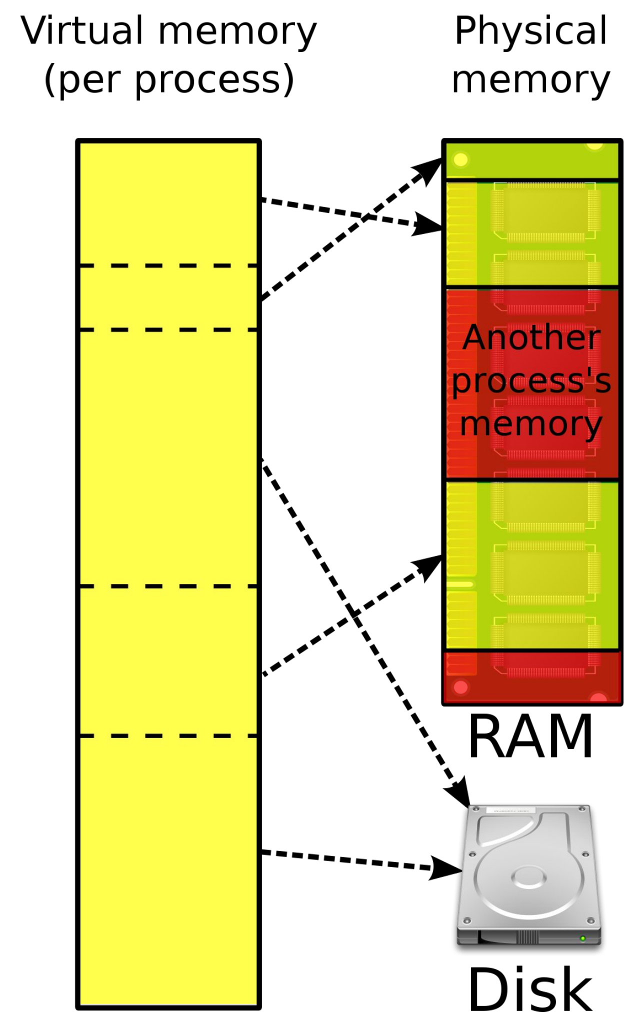
- OS enables process isolation
  - Each process sees its “own” processor
  - Each process is isolated from other processes
- User can run multiple apps at once on single machine

PROCESS SCHEDULING POLICY

- Scheduling policy
  - Controls how OS time-shares processor among processes
- Criteria
  - Throughput: number of processes completed in given time frame
  - Response Time: time process has to wait to execute
  - Turnaround Time: time for process to run, including waiting time
  - Others ...
- Many scheduling policies available
  - FIFO: first in first out
  - SJF: shortest job first
  - RR: round robin
  - MLFQ: multi-level feedback queue
  - Machine-learning-based
  - Others ...

VIRTUAL MEMORY

- Memory (also hardware) can also be virtualized by OS
- **Virtual memory**
  - Allows multiple processes to safely share available memory
  - Allows main memory to be extended through secondary storage
- Virtual memory allows multiple processes to *safely* and *efficiently* share available memory



https://en.wikipedia.org/wiki/Virtual_memory

M. H. Nguyen

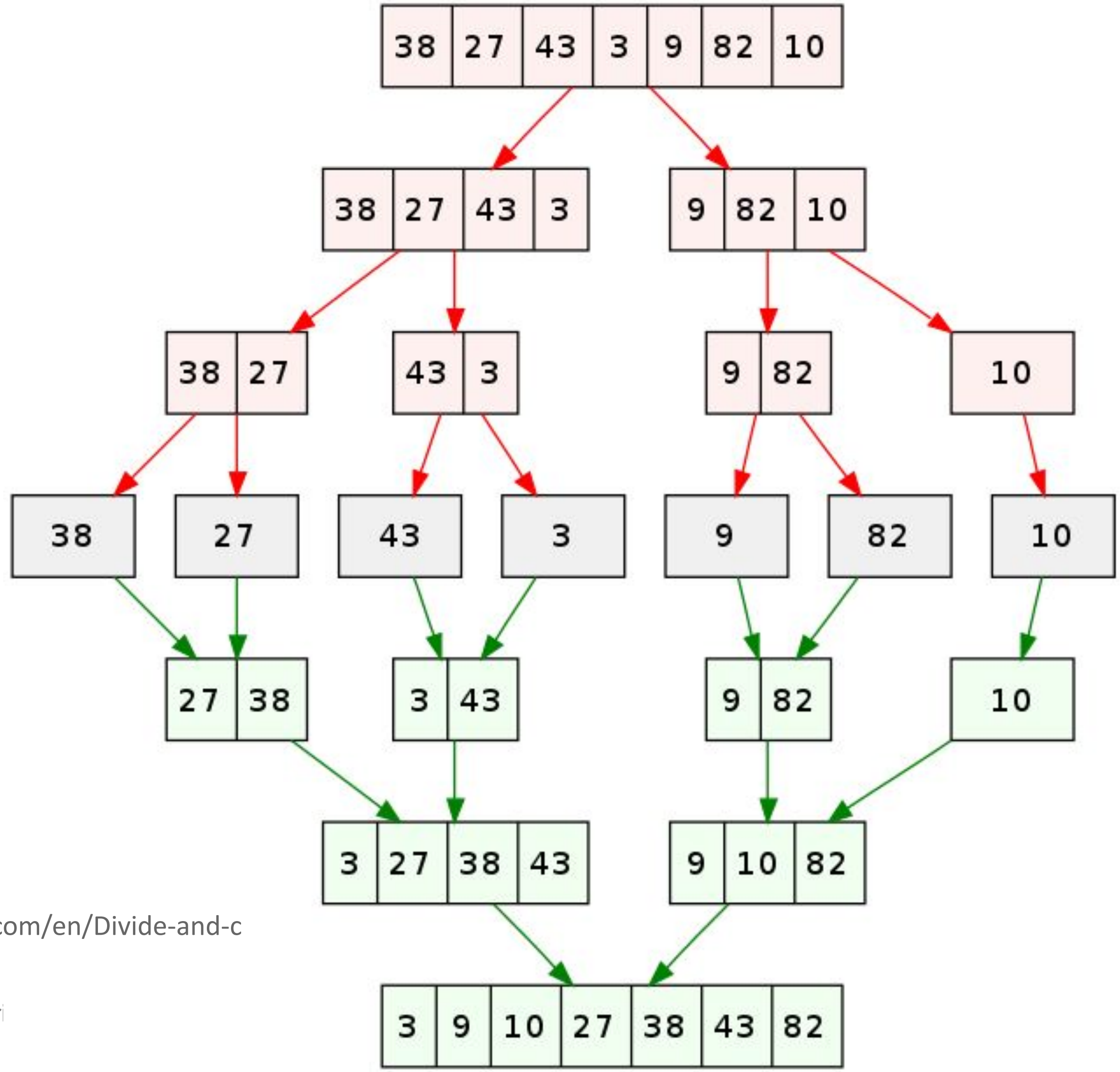
COMPUTER SYSTEMS & PARALLELISM

- Basics of Computer Systems
  - Hardware & Software
  - Computer Instruction Cycle
  - Memory Hierarchy
  - Virtualization
- Parallelism
  - Parallel Processing
  - Task & Data Parallelism
  - Speedup

PARALLEL PROCESSING

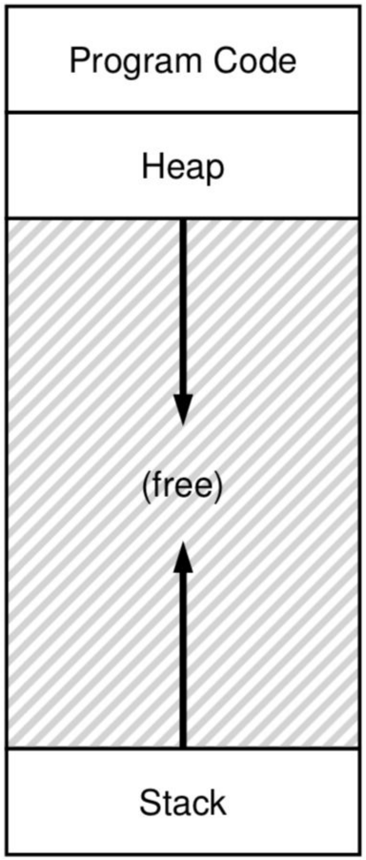
- Idea: Split workload across multiple computing resources in order to speed up processing
- Divide and conquer

DIVIDE AND CONQUER

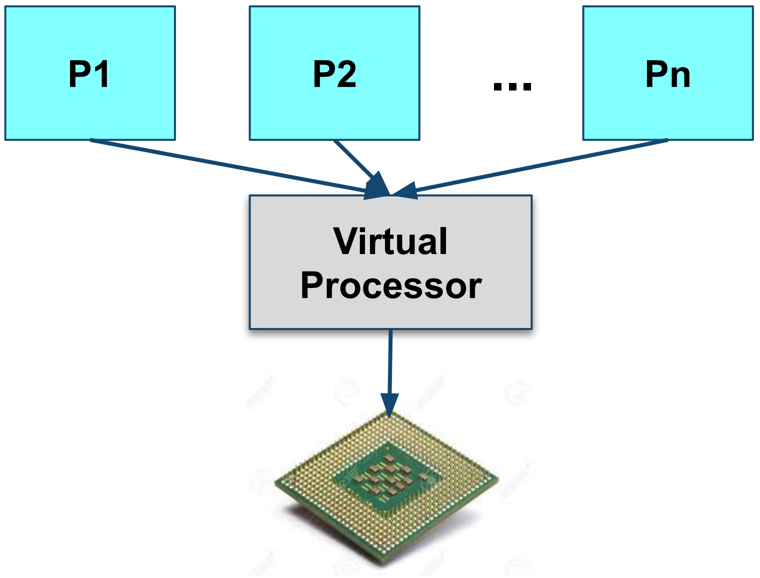


https://www.wikiwand.com/en/Divide-and-conquer_algorithm

PROCESS

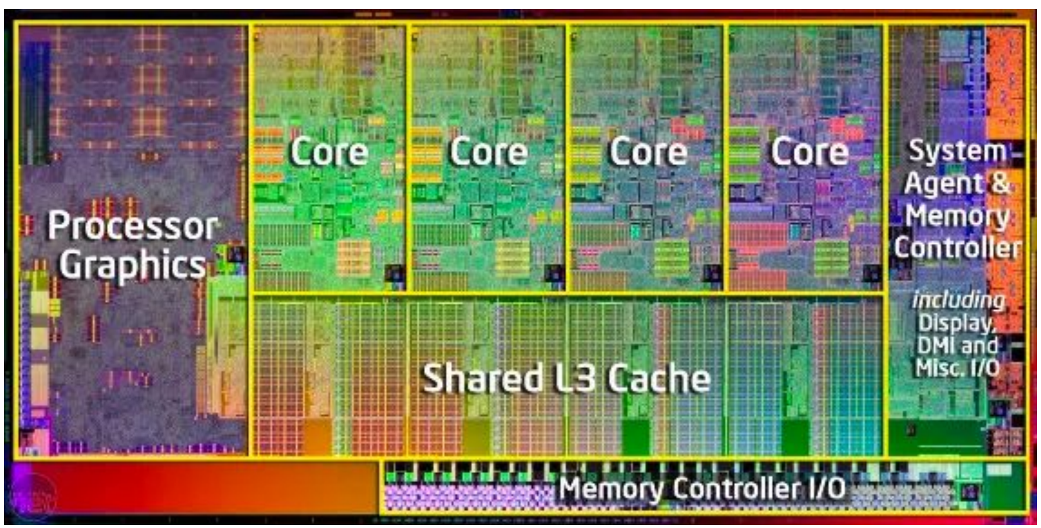


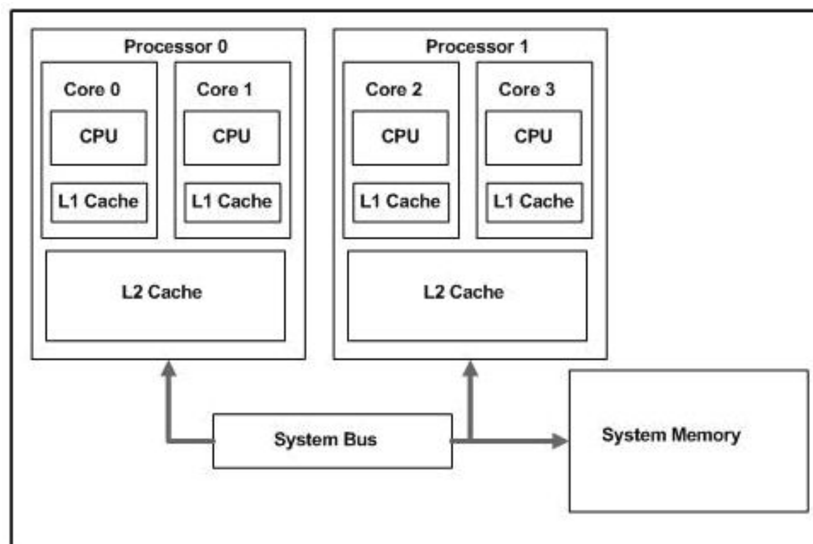
- Program: Text file with code
- Process: Program in execution
  - Process is assigned resources
  - OS time-shares processor among processes



MULTI-PROCESSING

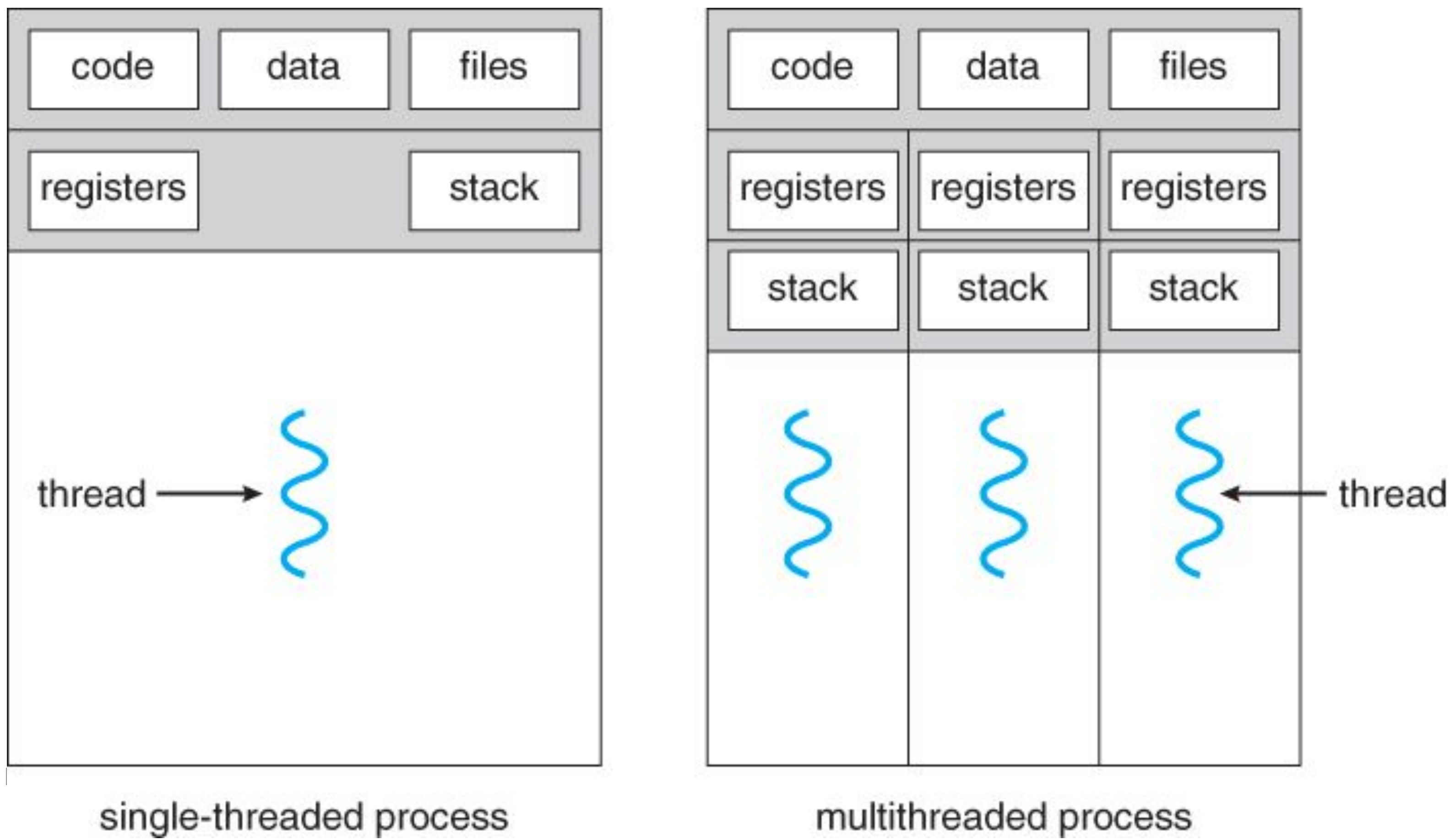
- Modern computers often have multiple *cores* per processor
  - Can also have multiple *processors*
- **Multiprocessing:** Executing multiple processes simultaneously on multiple cores/processors





THREADS

- **Thread**: Unit of execution within a process
- A process can have 1 or many threads
- Some part of process' address space is shared by all threads
- Thread is also called a lightweight process

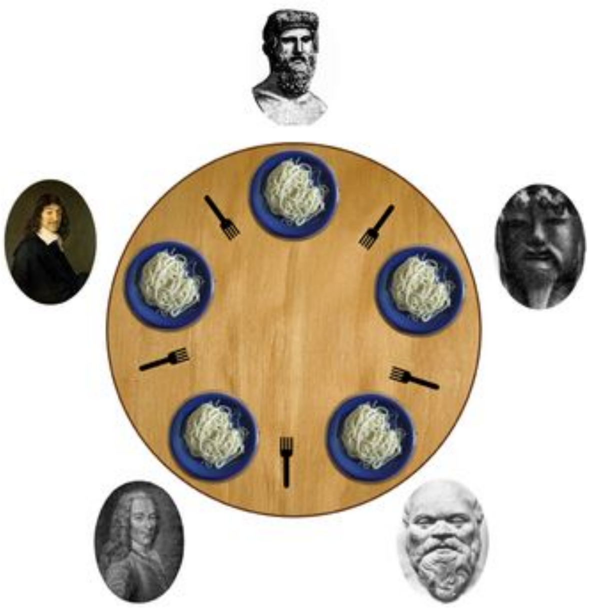


MULTI-THREADING

- Single-threaded process
  - Process has 1 thread. Process and thread are the same
- Multi-threaded process
  - Process has >1 threads
- **Multi-threading**
  - Execution of multiple threads concurrently, with each thread running on separate core
  - Provides way to improve runtime performance
- **Hyper-threading**
  - Single physical core virtualized as 2 logical cores
  - Threads are interleaved and run concurrently

SYNCHRONIZATION ISSUES

- When multiple processes/threads *write* to *shared* data, there are many issues that can arise:
  - Cache coherence
  - Locking
  - Deadlocks

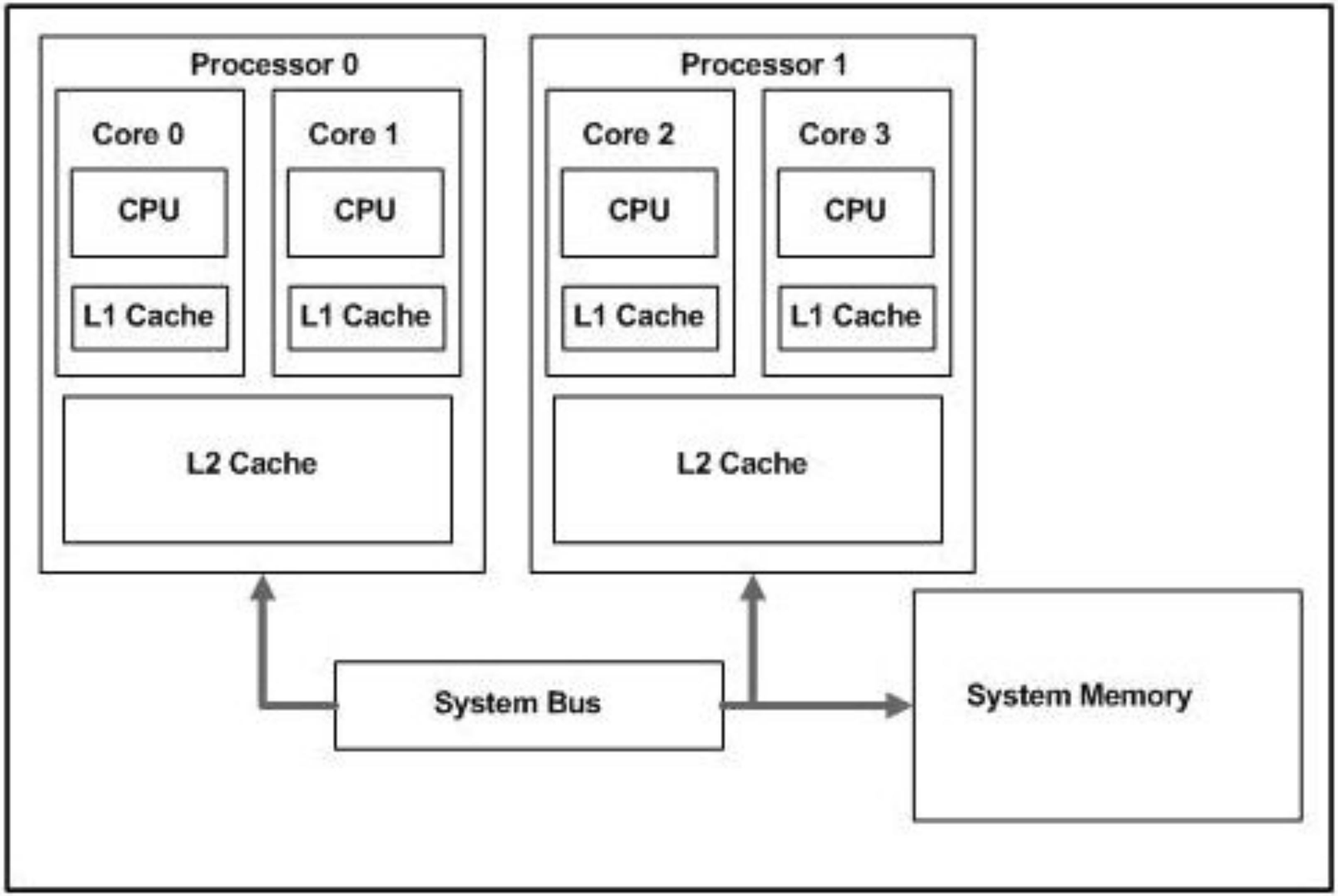


Dining Philosophers Problem

PARALLELISM

- Multi-processing
  - Executing multiple processes simultaneously on multiple cores/processors
- Multi-threading
  - Executing multiple threads on multiple cores/processors
- Distributed processing
  - Executing multiple sets of instructions on multiples systems (‘nodes’)

MULTIPLE PROCESSORS & MULTIPLE CORES

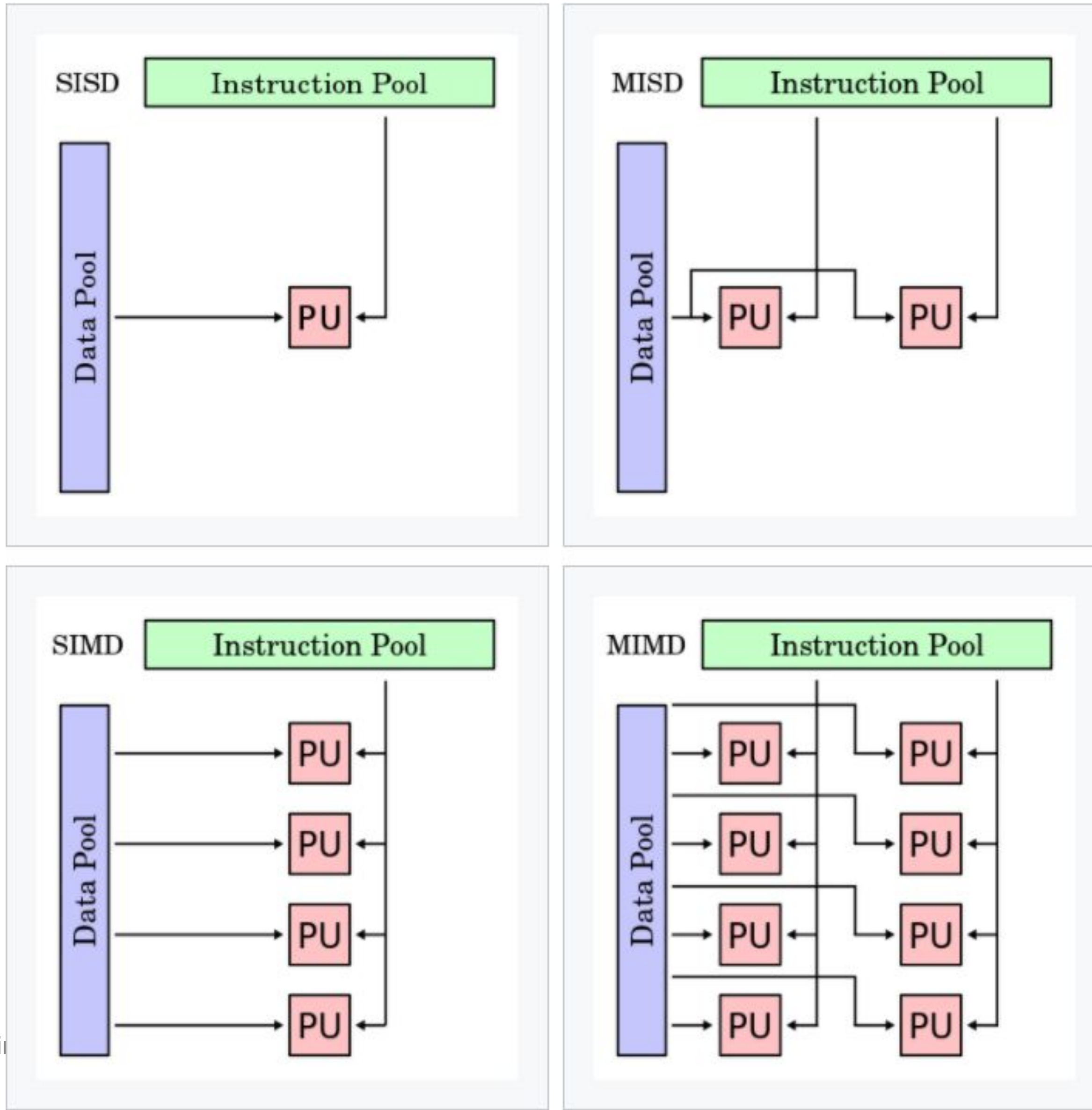


https://software.intel.com/content/www/us/en/develop/articles/software-techniques-for-shared-cache-multi-core-systems.html?wapkw=smart+cache

FLYNN'S TAXONOMY

- Classification of parallel computer architectures based on number of concurrent instruction streams and data streams
- SISD: single instruction, single data
  - sequential computer; no parallelism
- SIMD: single instruction, multiple data
  - same processing instructions executed on different chunks of data (i.e., vector processing)
- MISD: multiple instruction, single data
  - different processing instructions executed on same data
- MIMD multiple instruction, multiple data
  - different processing instructions executed on different data
  - SPMD (single program, multiple data): generalization of SIMD; most common type of parallel programming

FLYNN'S TAXONOMY

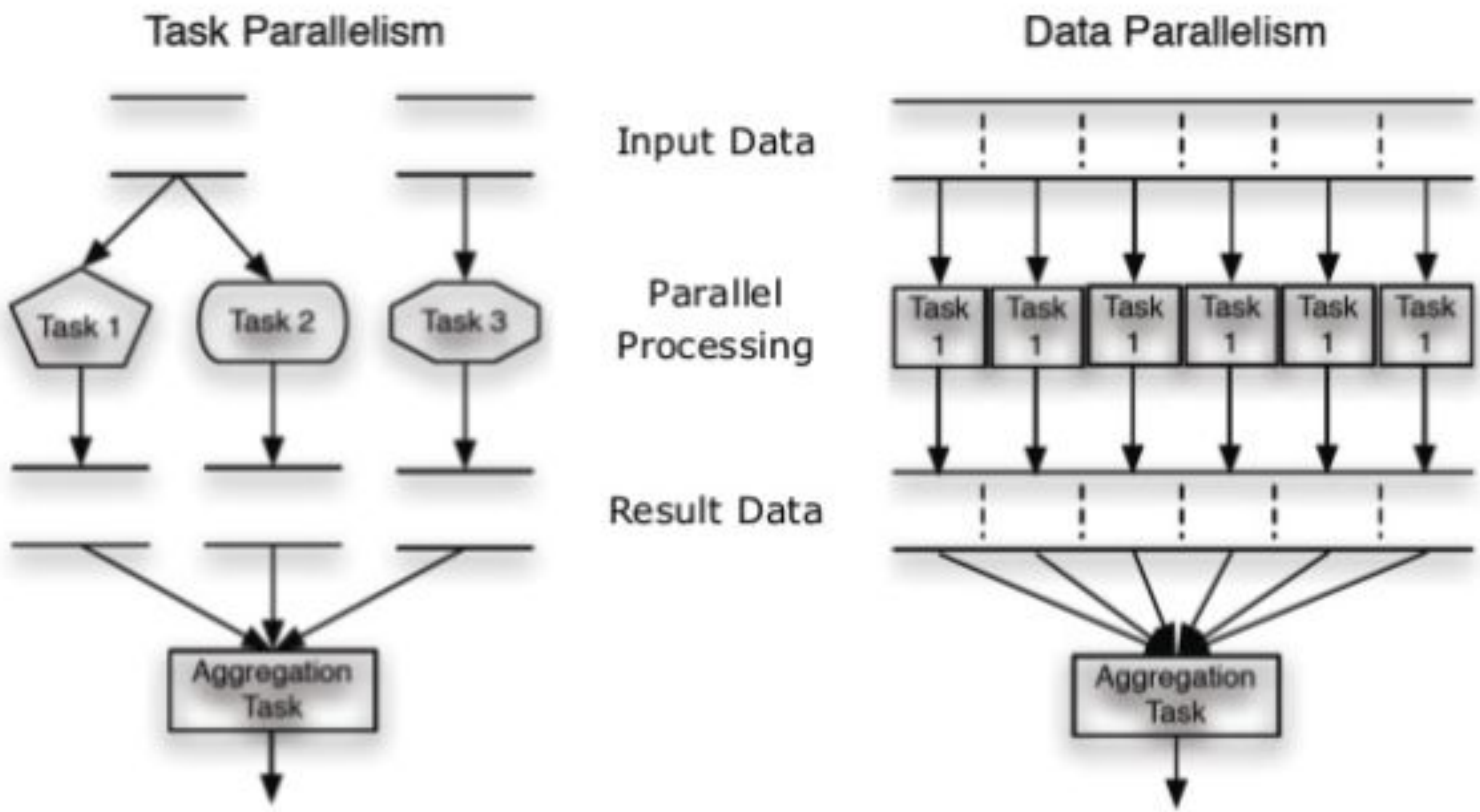


https://en.wikipedia.org/wiki/Flynn%27s_taxonomy

TASK PARALLELISM VS. DATA PARALLELISM

- Execution on multiple cores/processors/nodes
- Task Parallelism
  - Simultaneous execution of different functions (tasks) across same or different datasets
- Data Parallelism
  - Simultaneous execution of same function across subsets of same dataset

TASK PARALLELISM VS. DATA PARALLELISM



https://www.slideshare.net/PeterTrger/openhpi-parallel-programming-concepts-week-6

TASK PARALLELISM VS. DATA PARALLELISM

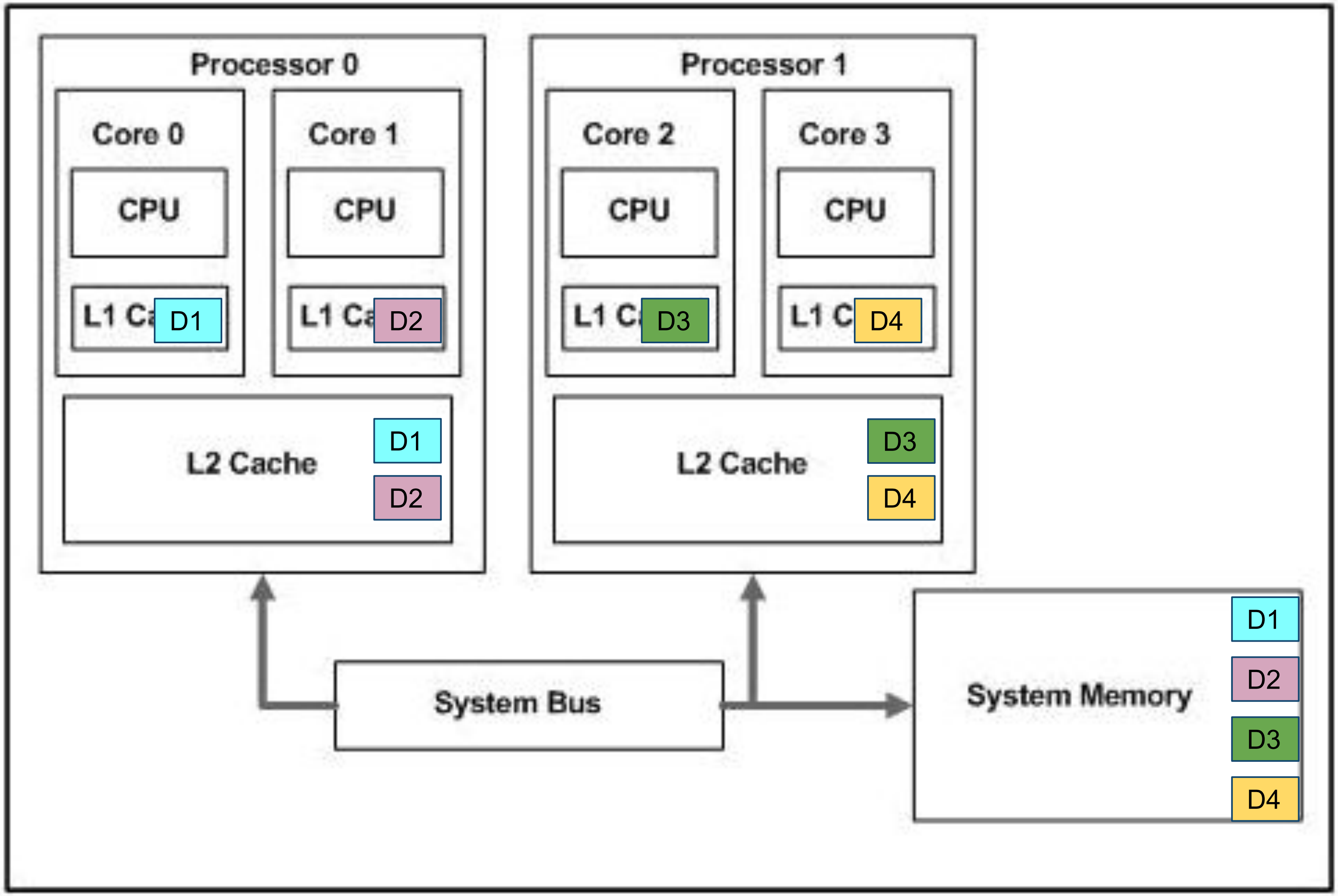
Task Parallelism

- Different operations performed on same or different data
- Asynchronous computation
- Amount of parallelization is proportional to number of independent tasks to be performed

Data Parallelism

- Same operations performed on different subsets of same data
- Synchronous computation
- Amount of parallelization is proportional to input data size

DATA PARALLELISM



https://software.intel.com/content/www/us/en/develop/articles/software-techniques-for-shared-cache-multi-core-systems.html?wapkw=smart+cache

SPEEDUP

- Parallel Computing
  - Processing large-scale data using multiple processors/nodes
- Scaling/Scalability
  - Ability of a computer system to process more data when the amount of resources is increased
- Speedup
  - How much faster a parallel algorithm is compared to a corresponding sequential algorithm

SPEEDUP

- Speedup is one way to quantify the benefit of parallelism

$$\text{Speedup} = \frac{\text{Execution time with 1 core/ processor / worker}}{\text{Execution time with N cores / processors / workers}}$$

AMDAHL'S LAW

- With N processors, do we get a speedup of N?
- **Amdahl's Law:** Formula to upper bound possible speedup
  - A program has 2 parts: one that benefits from parallelism and one that does not
  - Non-parallel part could be for control, I/O, etc.

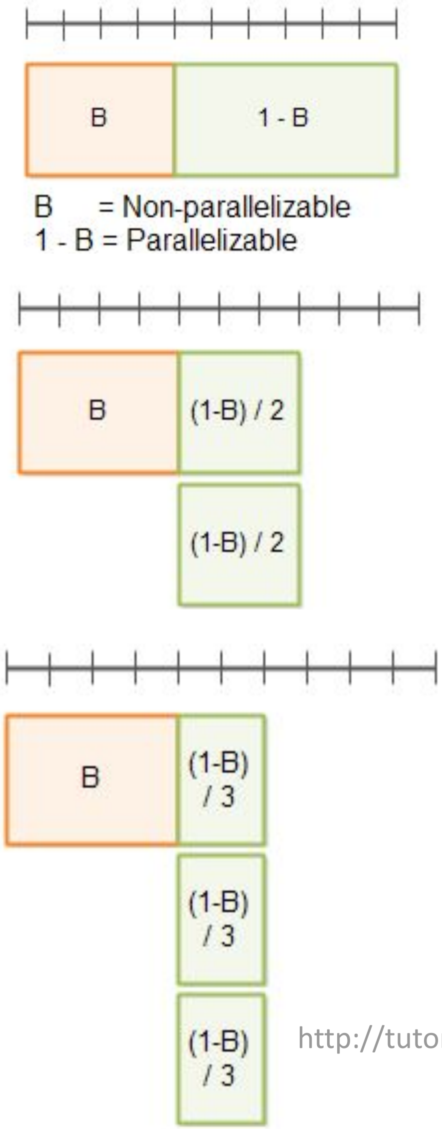
1 processor: n processors:

p $\longrightarrow$ p/n   p = % of execution time spent on parallel part

s $\longrightarrow$ s   s = % of execution time spent on serial part

Speedup = 1 / (s + (p/n))

AMDAHL'S LAW



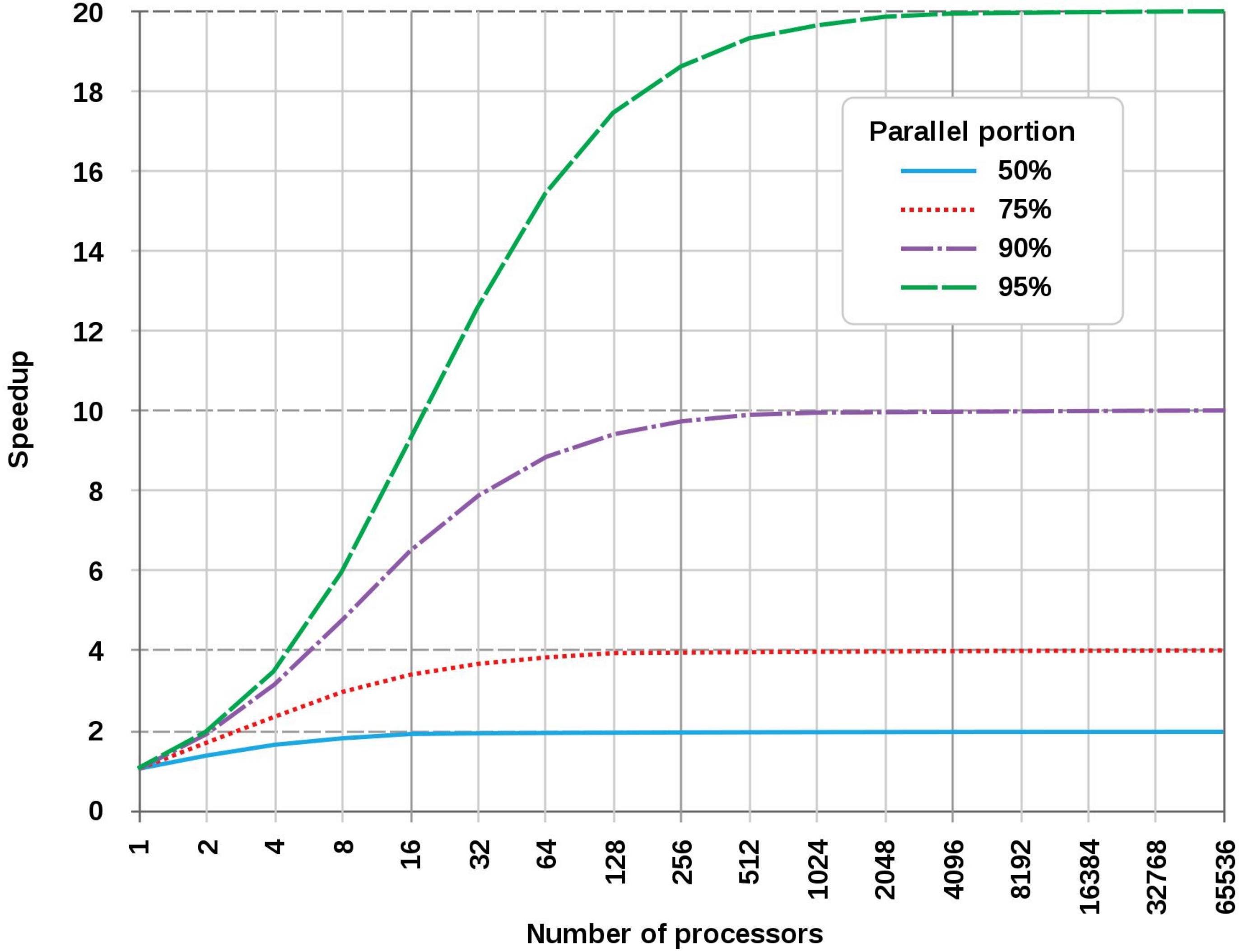
Parallelization factor of 1

Parallelization factor of 2

Parallelization factor of 3

http://tutorials.jenkov.com/java-concurrency/amdahls-law.html

AMDAHL'S LAW



Speedup = 1 / (s + (p/n))

GUSTAFSON'S LAW

- **Amdahl's Law**
  - Gives upper limit of speedup for problem of *fixed* size
  - In practice, problem size scales with amount of available resources

- **Gustafson's Law**
  - Reformulate so that solving larger problem in same amount of time is possible
  - Parallel part scales linearly with amount of resources, and serial part does not increase with respect to problem size

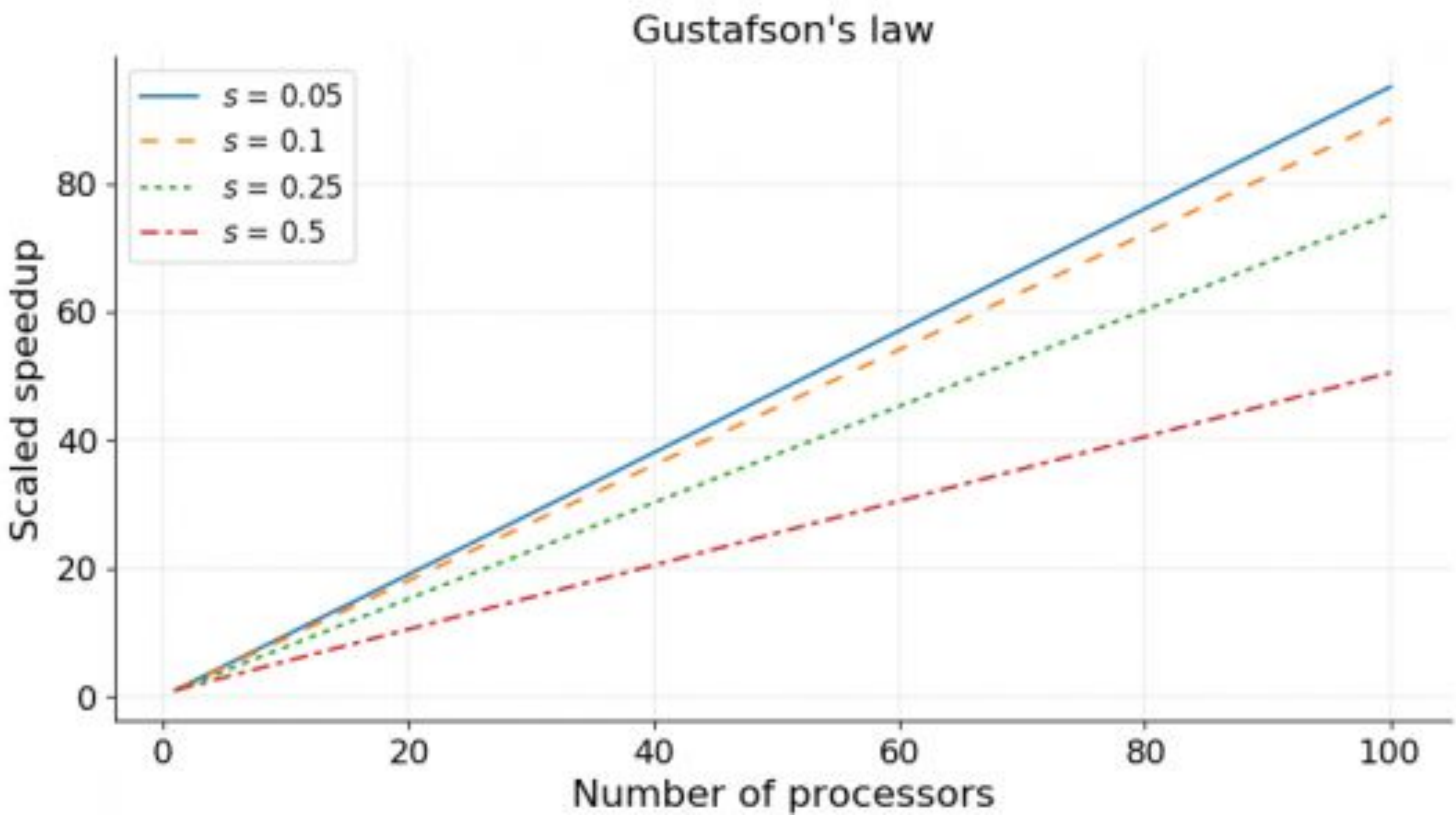
GUSTAFSON'S LAW

- Also provides upper bound on speedup with multiple processors
- Gives upper limit of speedup for problem with a fixed amount of parallel work *per processor*
- Speedup is calculated based on amount of work done for a *scaled* problem size
- Given

  p = % of execution time spent on parallel part
  s = % of execution time spent on serial part
  n = number of processors

Scaled Speedup = s + pn

GUSTAFSON'S LAW



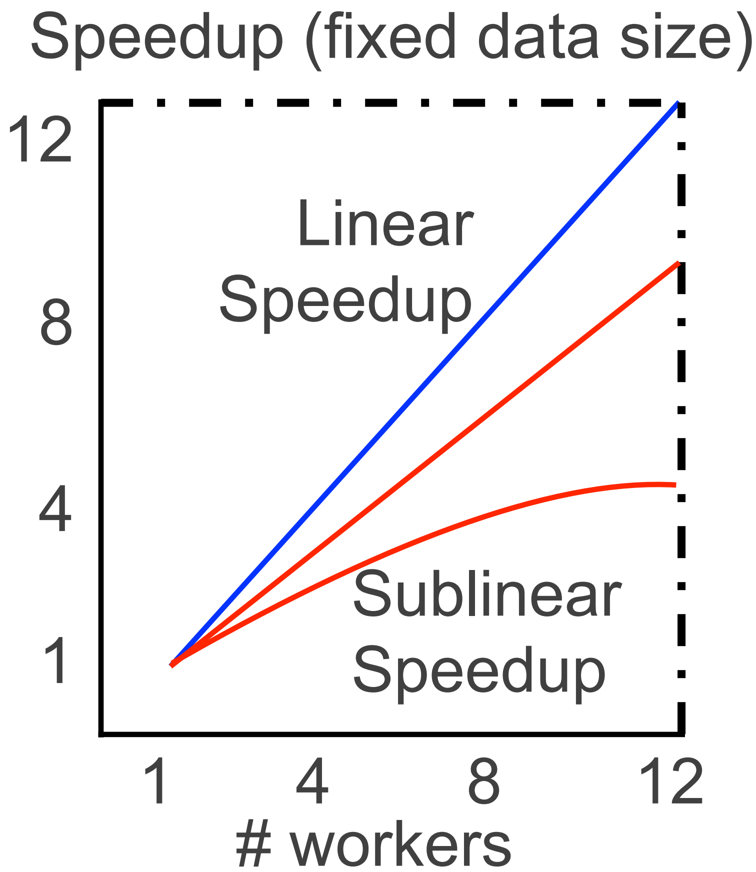
Scaled Speedup = s + pn

https://www.kth.se/blogs/pdc/2018/11/scalability-strong-and-weak-scaling/#:~:text=Strong%20scaling%20concerns%20the%20speedup,is%20governed%20by%20Gustafson's%20law.

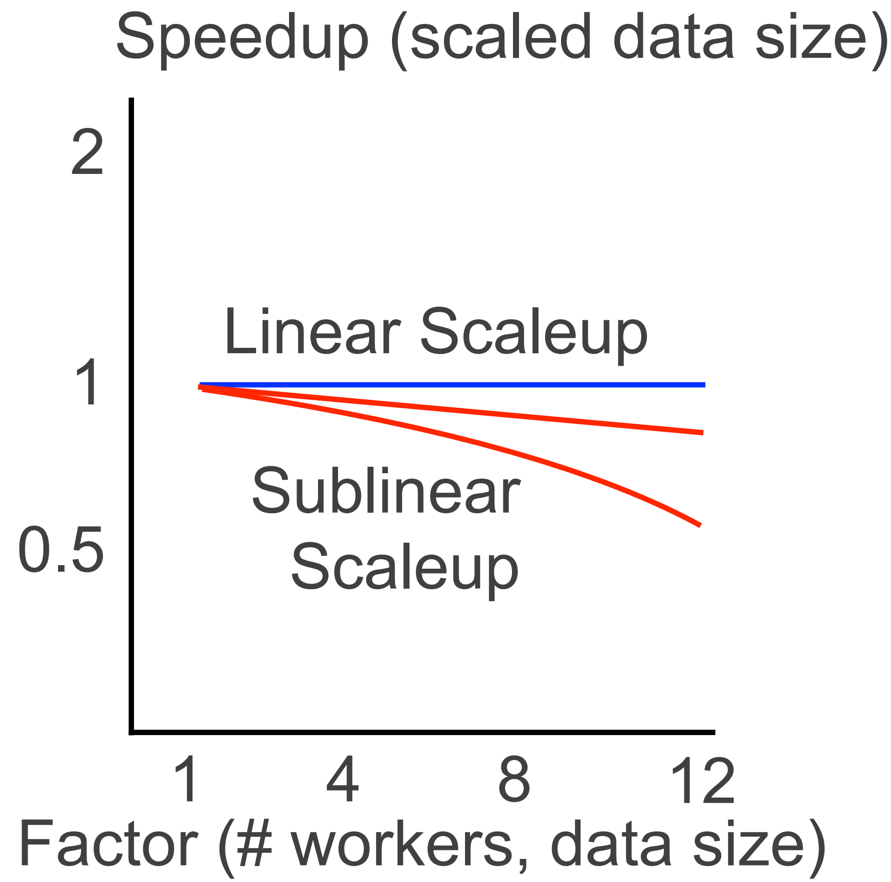
STRONG VS WEAK SCALING

- **Strong Scaling**
  - How execution time varies with number of processors for a fixed *total* problem size
  - Speedup for a *fixed* problem size wrt number of processors
  - How much does parallelism reduce execution time of a fixed problem?
  - Governed by Amdahl's law
- **Weak Scaling**
  - How execution time varies with number of processors for fixed problem size *per processor*
  - Speedup for a *scaled* problem size wrt number of processors
  - How much more data can we process in same amount of time through parallelism?
  - Governed by Gustafson's law

QUANTIFYING PARALLELISM



Speedup plot / **Strong scaling**



Scaleup plot / **Weak scaling**

Arun Kumar, DSC102

REFERENCE MATERIAL

- "Computer Organization and Design" by David A. Patterson and John L. Hennessy
- "Operating Systems: Three Easy Pieces" by Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau

SESSION 1 QUIZ

Q1. What is a memory stall?

A. When the CU and ALU are idle waiting for data to be fetched
B. Results in sub-optimal use of processor
C. Can be avoided by raising cache hits
D. When data needed by process is not available
E. All of the above

SESSION 1 QUIZ

Q2. Which of the following is true about locality of reference?

A. Spatial locality of reference means that same locations will be accessed again soon
B. Temporal locality of reference means that nearby locations will be accessed soon
C. Locality of reference refers to the different access speeds in the memory hierarchy
D. Locality of reference can be exploited to optimize cache hits
E. All of the above

SESSION 1 QUIZ

Q3. Which of these layers of the memory hierarchy typically has the highest access speed?

A. Flash drive
B. Main memory
C. Magnetic disk
D. CPU caches
E. DRAM

SESSION 1 QUIZ

Q4: In process virtualization

A. the OS isolates each process from other processes
B. the OS time shares the processor among multiple processes
C. the OS makes it possible to run multiple apps on a single machine
D. A & C
E. A, B, & C

SESSION 1 QUIZ

Q5: Virtual memory

A. Allows multiple processes to safely share available memory
B. Allows main memory to be extended through secondary storage
C. Allows multiple processes to share each other’s allocated memory
D. A & B
E. A, B, & C

SESSION 1 QUIZ

Q6: In data parallelism

A. Different functions are performed across the same dataset (task parallelism)
B. The same function is performed across subsets of a dataset
C. Operations are performed on subsets of data synchronously
D. B & C
E. A, B, & C

SESSION 1 QUIZ

Q7. In strong scaling

A. The dataset size is fixed as the number of processors is changed
B. Specifies how much parallelism reduces the execution time of a fixed problem
C. Is governed by Amdahl's law
D. Speedup results are shown on a speedup plot
E. All of the above

SESSION 1 QUIZ

Q8: Which of the following is true about parallel processing?

A. In parallel processing, the workload is divided across multiple processors/nodes in order to speed up processing
B. Scalability refers to the ability of a computer system to process more data when the amount of resources is increased
C. Speedup refers to the how much faster a parallel algorithm is compared to the sequential version of the algorithm
D. Speedup = (execution time with 1 processor/node) / (execution time with N processors/nodes)
E. All of the above

COMPUTER SYSTEMS & PARALLELISM

- Basics of Computer Systems
  - Hardware & Software
  - Computer Instruction Cycle
  - Memory Hierarchy
  - Virtualization
- Parallelism
  - Parallel Processing
  - Task & Data Parallelism
  - Speedup