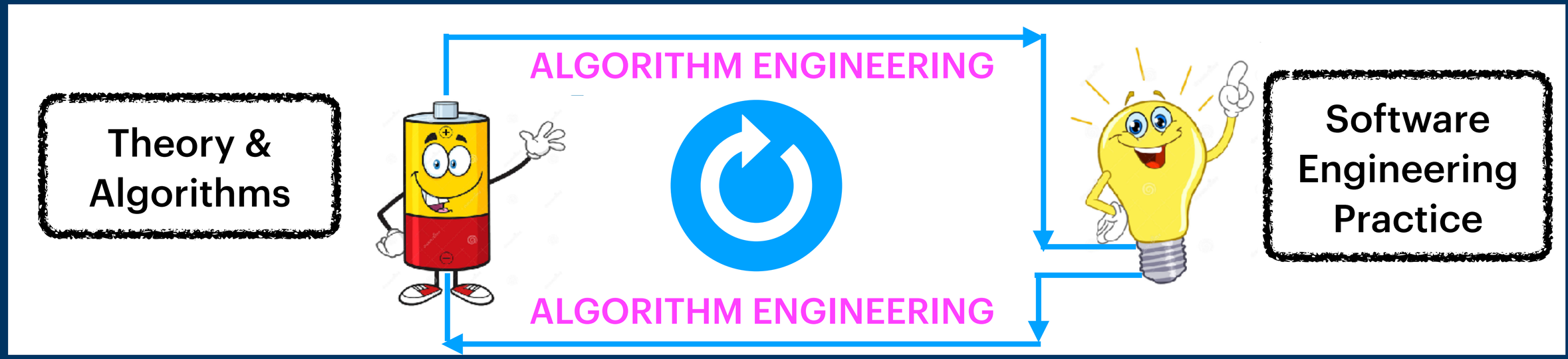


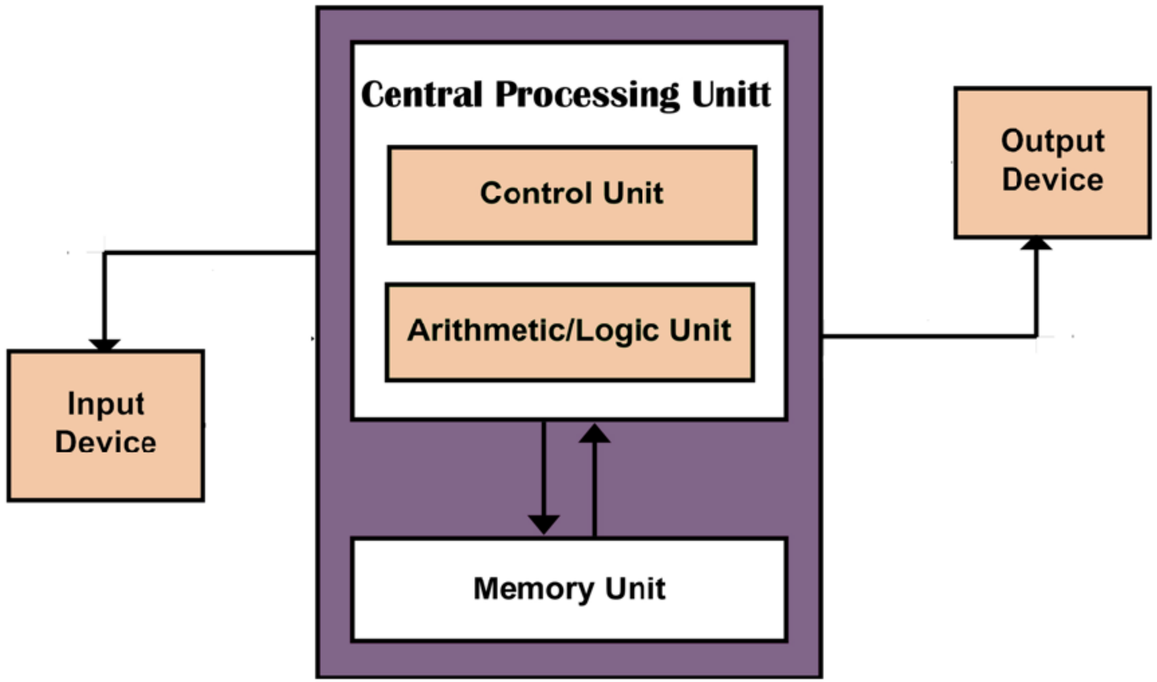
ALGORITHM ENGINEERING

Lecture 2:
Realistic Computing Models

M. Oğuzhan Külekci - kulekci@itu.edu.tr

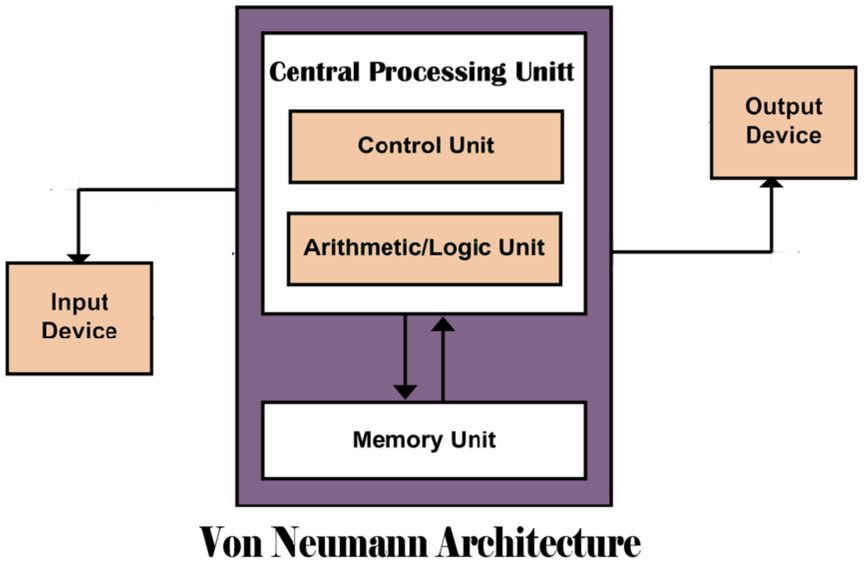
Computing Model

- The architecture of a machine that executes the software (= DS + Algorithms, you know:)).
- Specs are important while devising an algorithm !



Von Neumann Architecture

RAM Model

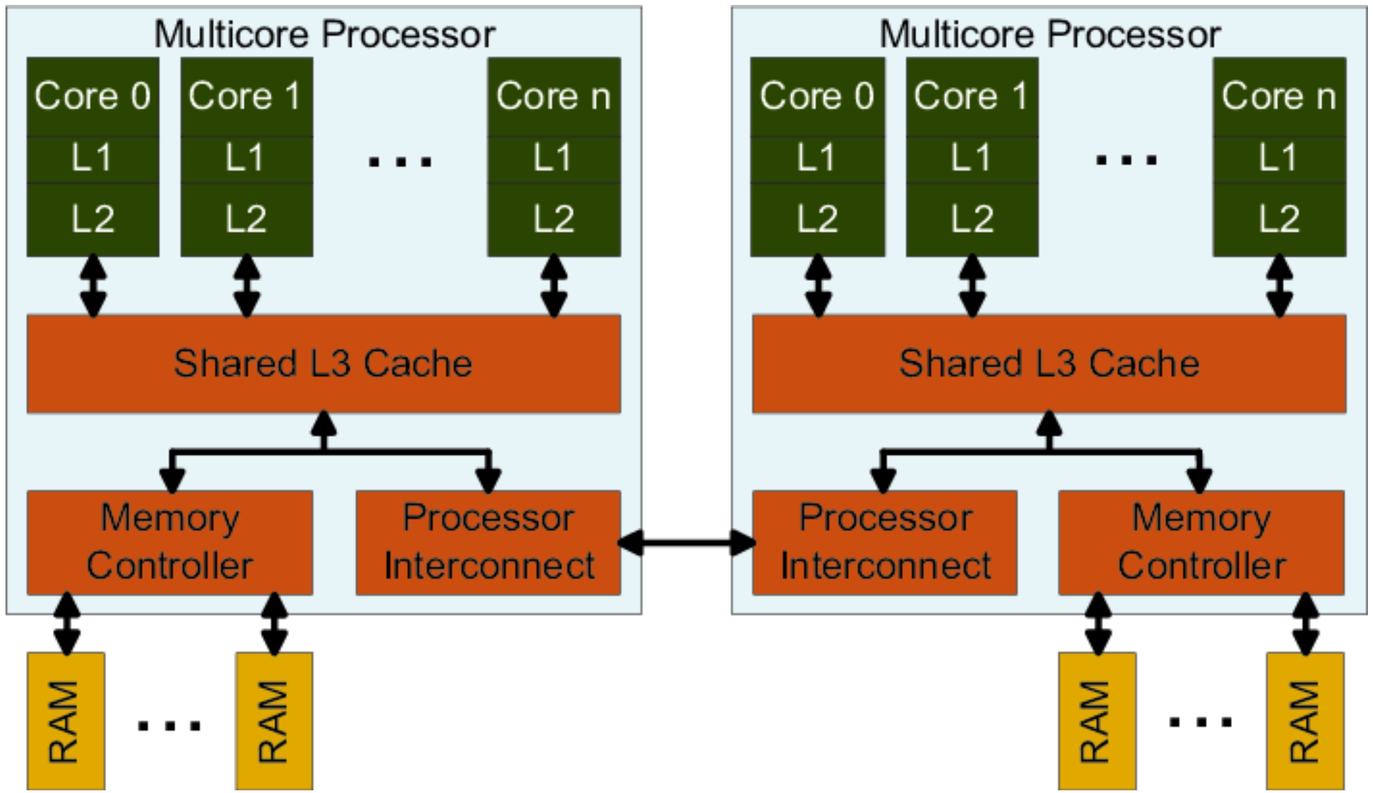


Von Neumann Architecture

- In-order (!) sequential execution of instructions
- Unbounded memory of ***words***
- A word is of size $O(\log n)$, for any n
- Constant-time access to each word in memory
- Equal amount of execution time per instruction

Real computing environments are a bit (!!!) different than these assumptions ?

RAM Model - Theory vs. General Practice

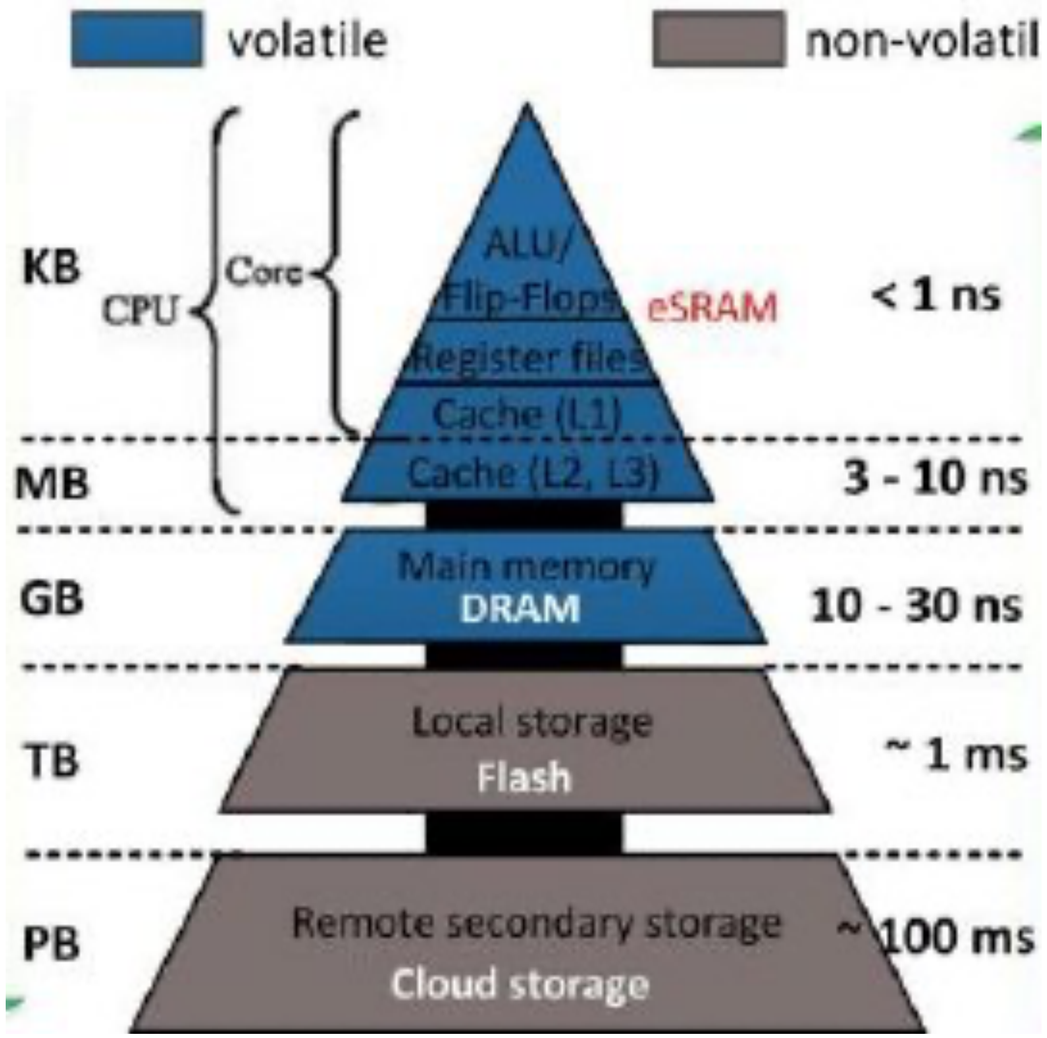


- Sequential execution
- Unbounded memory of ***words***
- A word is of size $O(\log n)$
- Constant-time access to each word in memory
- Equal amount of execution time per instruction

THEORY VS PRACTICE

- **Parallel** execution (out-of-order, multi-core, vectorization, pipelining, etc…)
- **Limited** memory of ***words***
- A **word** is of **constant** number of bits
- **Variable-time** access due to **memory hierarchy**
- **Different** amount of execution time per instruction

MEMORY HIERARCHY



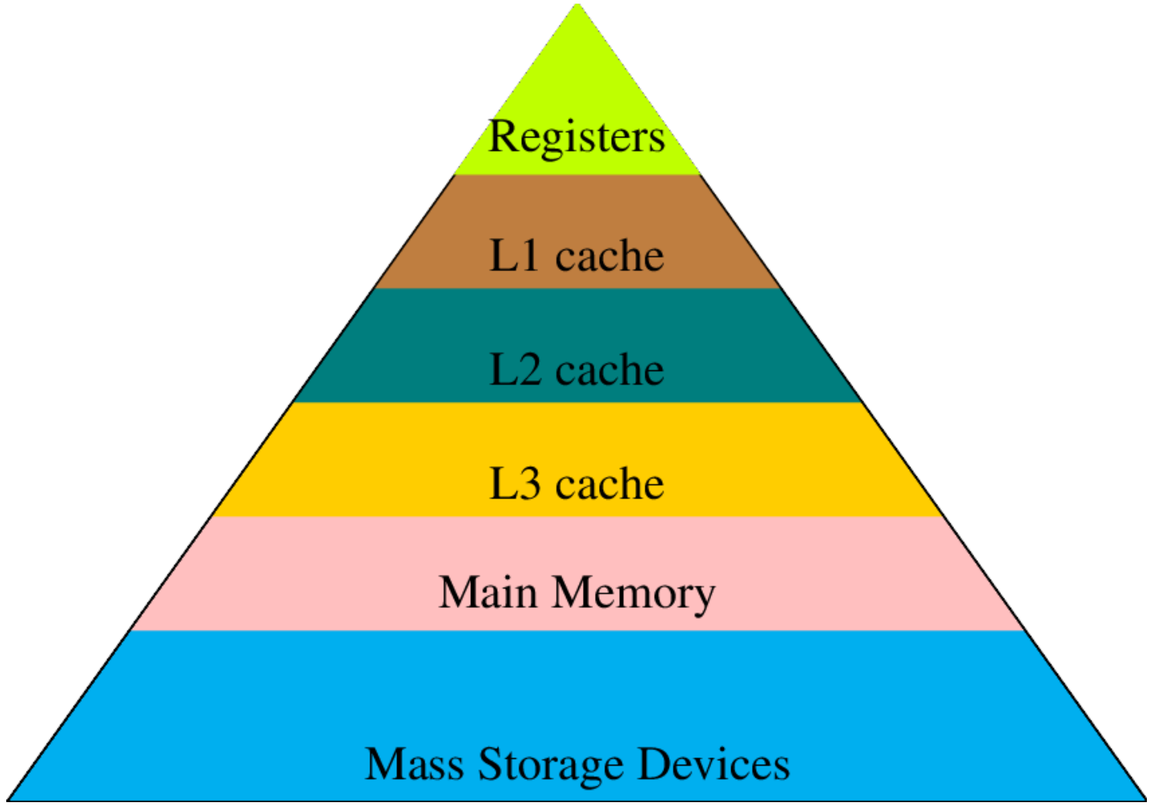
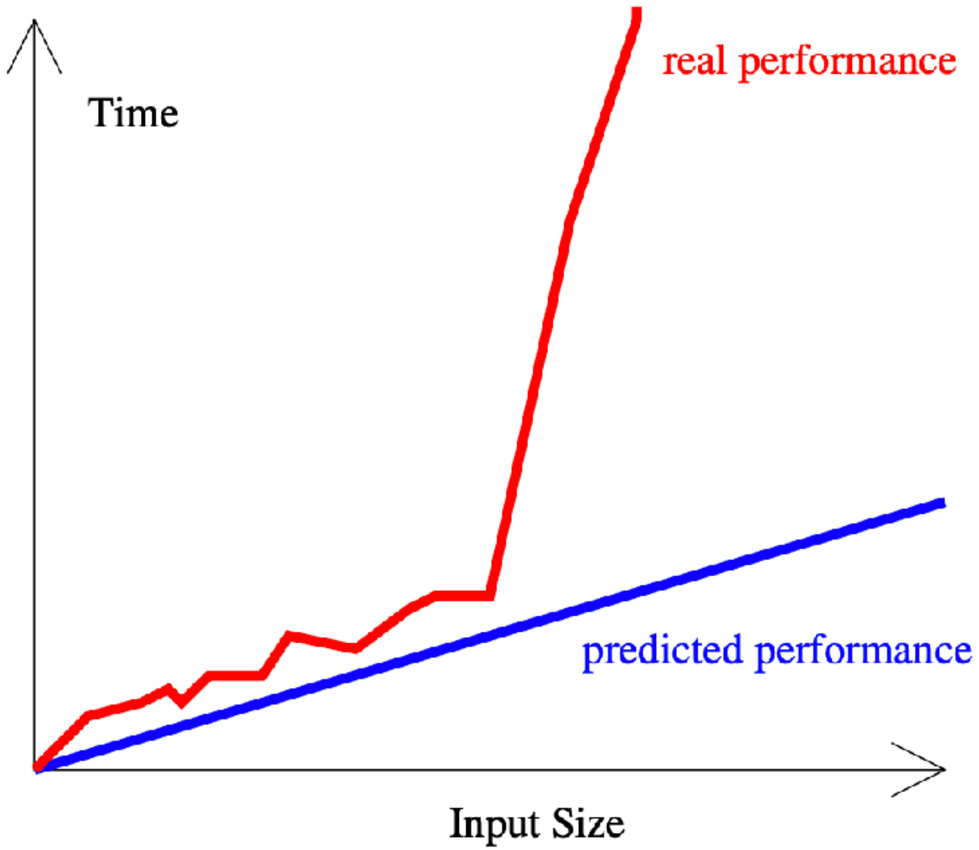


Figure is from Jovanovic et al.. (2015). DOI: 10.2298/FUEE1503465J.

RAM Model Critique

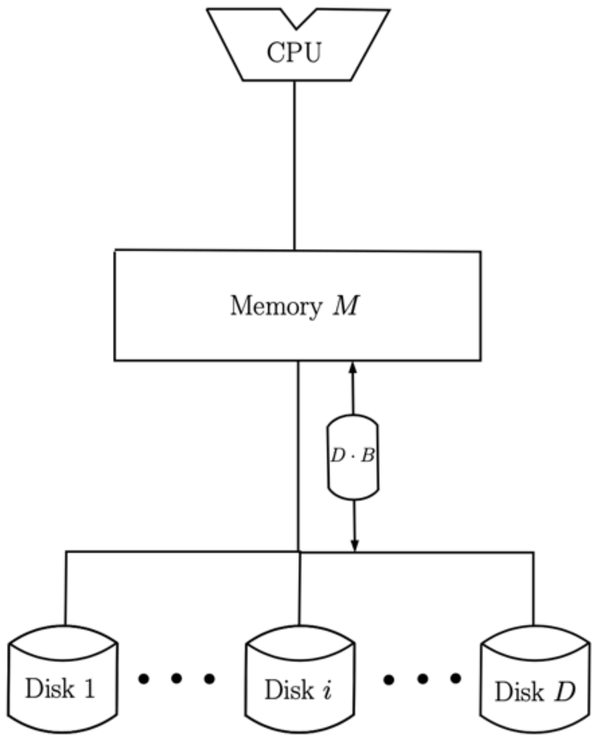


- RAM model hides details, yet enough good for theoretical comparisons.
- The elapsed time in practice highly depends on **I/O** rather than instruction executions, particularly on **large data sets**. Thus, **data access efficiency** matters a lot !
- **Parallel execution** power of multi-core / multi-CPU systems change instruction execution time seriously. Compiler optimizations are nearly automatic.

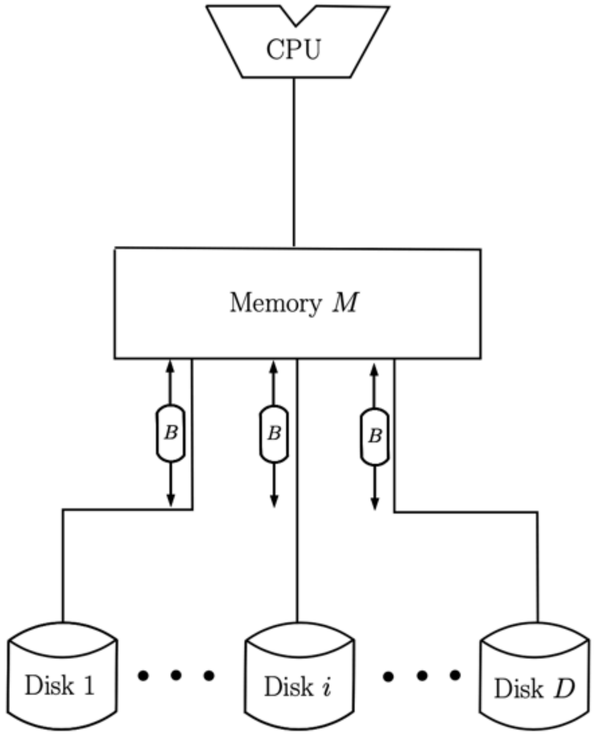
So, there are some other models to consider some of these issues !

Models for Exploiting Memory Hierarchy

External Memory - Parallel Disk Models



EXTERNAL MEMORY MODEL (EM)
(Aggarwal. & Vitter)



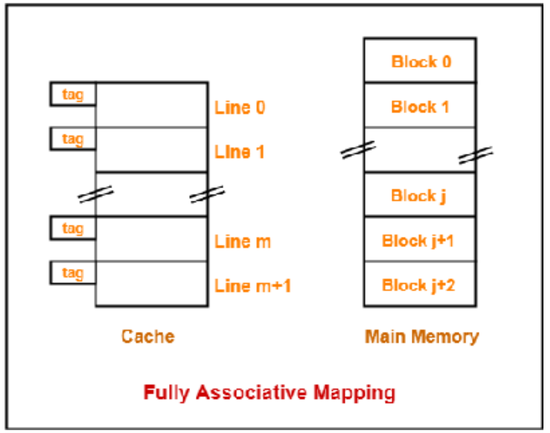
PARALLEL DISK MODEL (PDM)
(only one block can be accessed per disk)
(Vitter & Shivel)

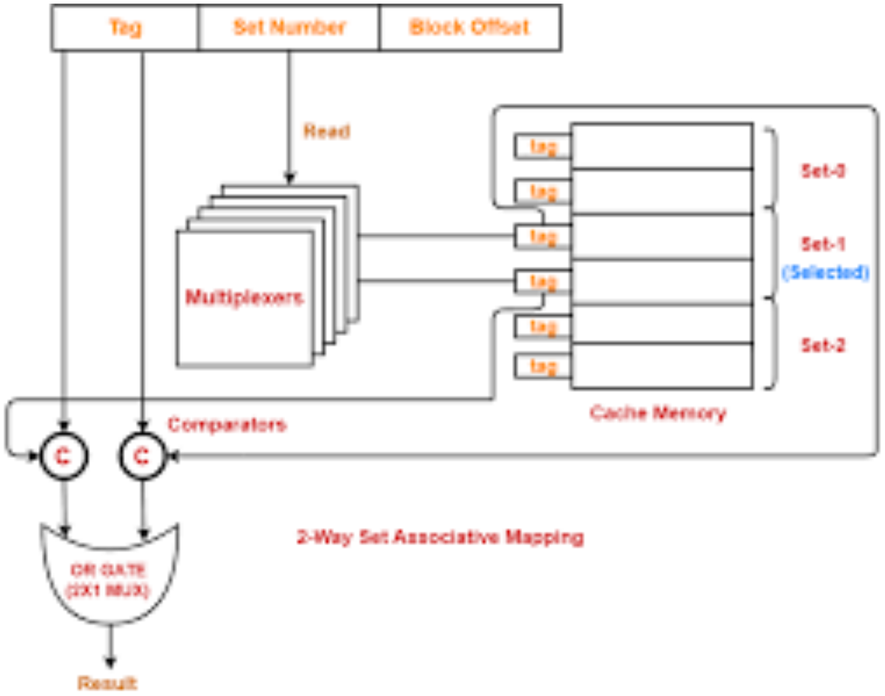
- Two-level memory, fast internal and slow external
- Internal memory has M words; external memory has enough storage much larger than M; data flows from/to disk in chunks of B words.
- Aim is to measure I/O complexity
- Memory-Disk (Fast-Slow), Cache-Memory (Fast-Slow) ?

Models for Exploiting Memory Hierarchy

External Memory - Parallel Disk Models

- Can PDM work between memory and cache ?
- Memory-Disk (Fast-Slow): Assumed fully associative
- Cache-Memory (Fast-Slow): Limited associativity with automated replacement, thus, needs care





Models for Exploiting Memory Hierarchy

Cache-Aware versus Cache-Oblivious

```
for (rowID = 1 to m)
   for (colID = 1 to n)
      a[rowID][colID] = X
```

```
for (colID = 1 to n)
   for (rowID = 1 to m)
      a[rowID][colID] = X
```

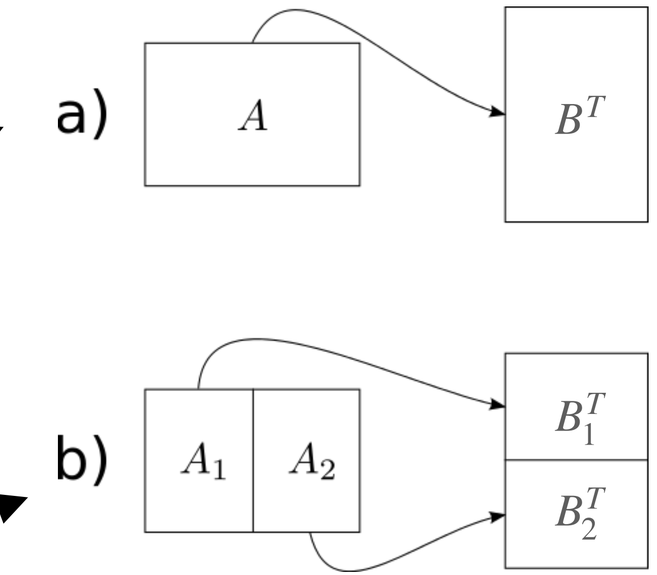
Which one is better ? Why ?

- Cache-aware: Minimize cache-misses while executing the algorithm
- Cache-aware is **specific**, you need to tune according to M and B, but you are free to choose M and B accordingly (easy in theory, but hard in practice)
- **Cache-aware algorithms run very fast in practice even sometimes outperforming alternatives with better time-complexities**

Models for Exploiting Memory Hierarchy

Cache-Aware versus Cache-Oblivious

```
for (rowID = 1 to m)
   for (colID = 1 to n)
      A[rowID][colID] = B[colID][rowID]
```



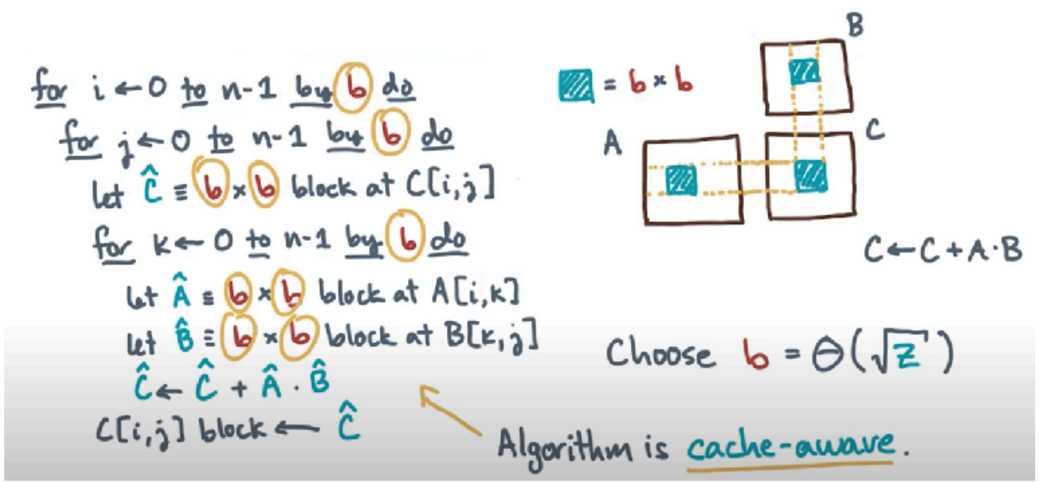
Now, cache-misses are unavoidable !

How to make this cache friendly on any selection of M and B ?

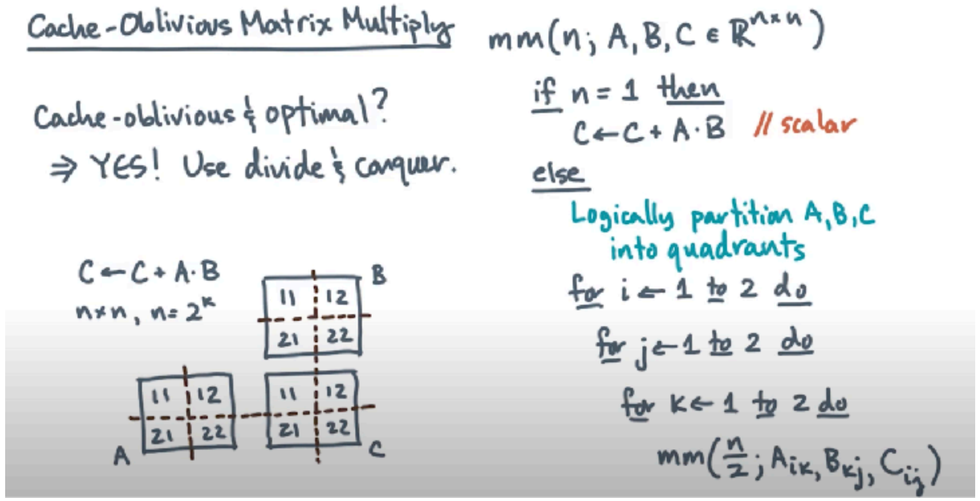
- Cache-oblivious: Minimize cache-misses on **any** selection of M and B
- Cache-oblivious is more general, whatever you choose M and B it provides a performance guarantee.
- **Try designing cache-obviously, implementing cache-aware !**

Models for Exploiting Memory Hierarchy

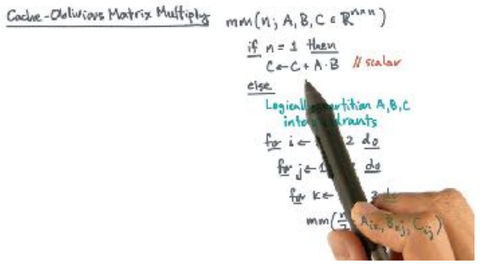
Cache-Aware versus Cache-Oblivious Matrix Multiplication



Parameter b is chosen according to cache-size, and is directly in the algorithm.

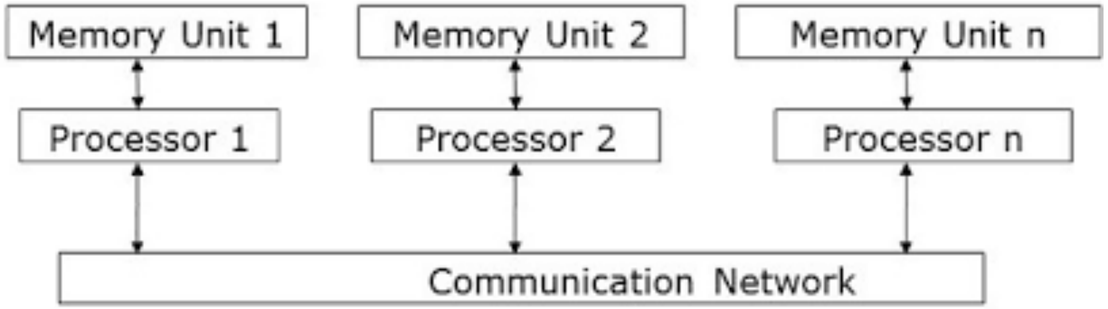


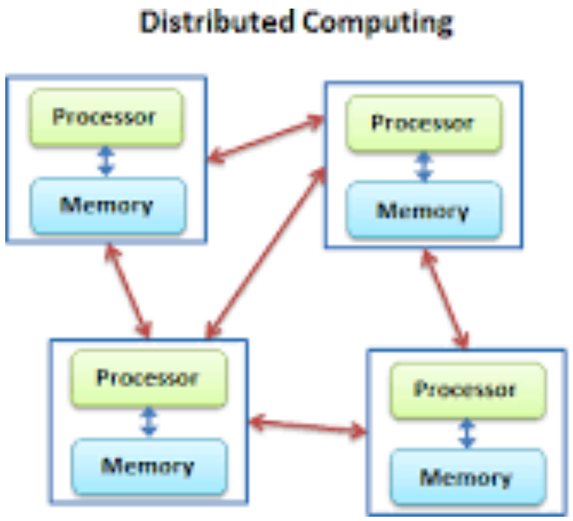
b is not a parameter now. No tuning required, works free of cache-size.

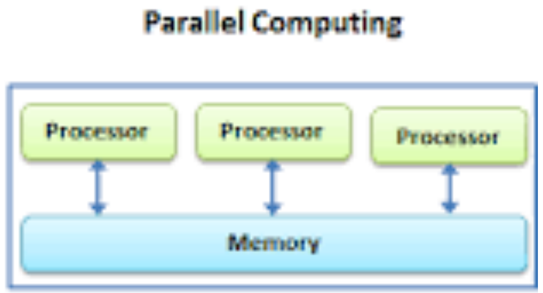


Models for Exploiting Parallelism

Parallel RAM model and Distributed Computing





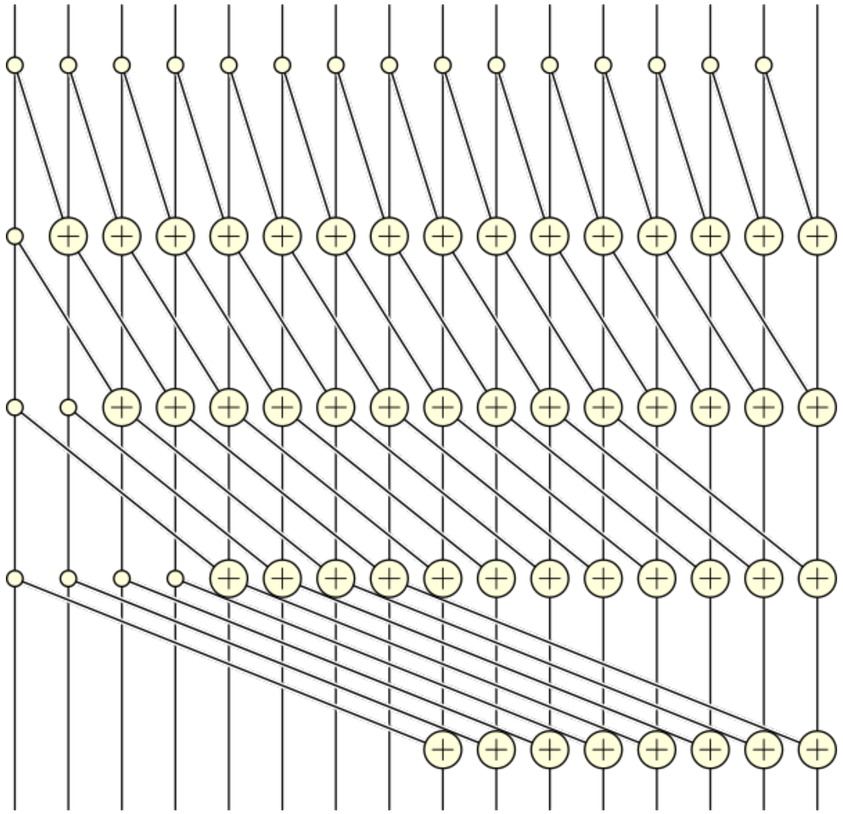


- PRAM: direct extension of RAM model
- There are subdivisions according to read/write from/to shared memory (CRCW, CREW, …)

Distributed (network model) vs. PRAM

Prefix Sum

Parallel Solution Examples



Note: This algorithm is not work-efficient, meaning we do more summation than necessary to increase parallelism.

Hills & Steele Algorithm for Prefix Sum

$O(n \log n) \rightarrow O(n)$ via parallelism

for $i \leftarrow 0$ **to** $\lceil \log_2 n \rceil - 1$ **do**

  for $j \leftarrow 0$ **to** $n - 1$ **do in parallel**

    if $j < 2^i$ **then**

      $x_j^{i+1} \leftarrow x_j^i$

    else

      $x_j^{i+1} \leftarrow x_j^i + x_{j-2^i}^i$

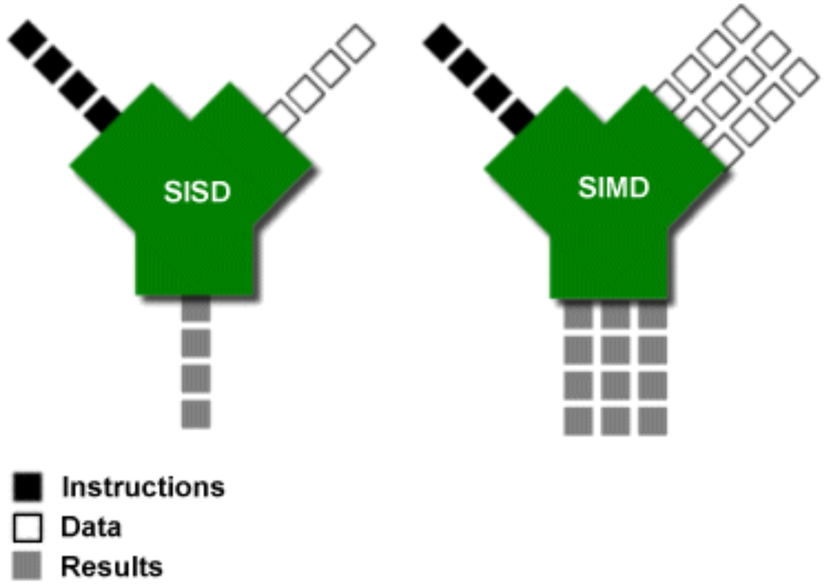
$O(\log n)$

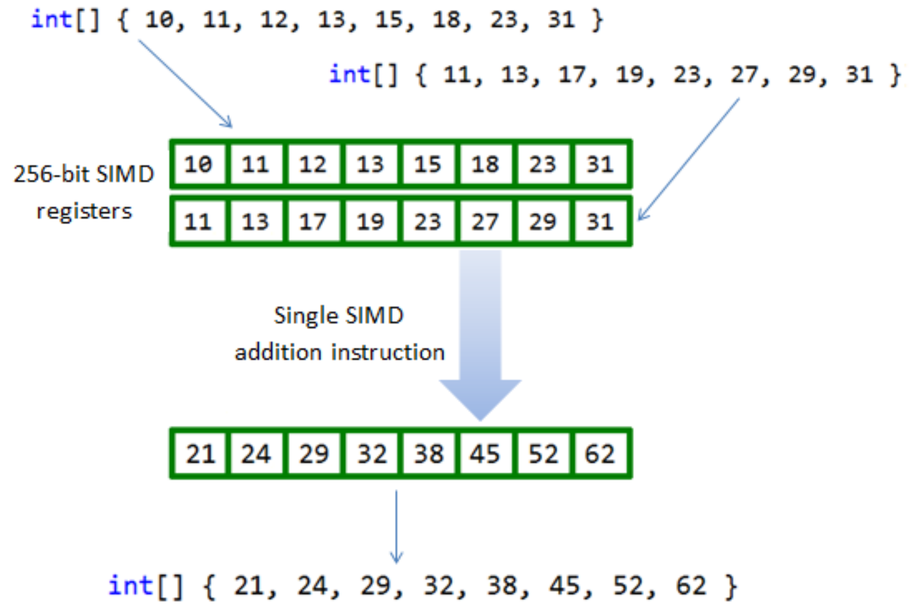
$O(n) \rightarrow O(1)$

Parallel prefix sum an be achieved even more efficient with vectorization (SIMD) support. See http://www.adms-conf.org/2020-camera-ready/ADMS20_05.pdf

Vectorization

Single-Instruction-Multiple-Data (SIMD)





- While designing your solution, never forget the SIMD support in modern CPU's.
- Special instructions that run on multiple data on special long registers.
- Requires great care and also understanding of the underlying hardware architecture.
- Particularly, very useful in massive data processing.

Realistic Computing Models

- I/O efficiency
  - Cache optimization
- Parallel processing

We need all of them in real life to compute fast and cheap, particularly on large data sets.

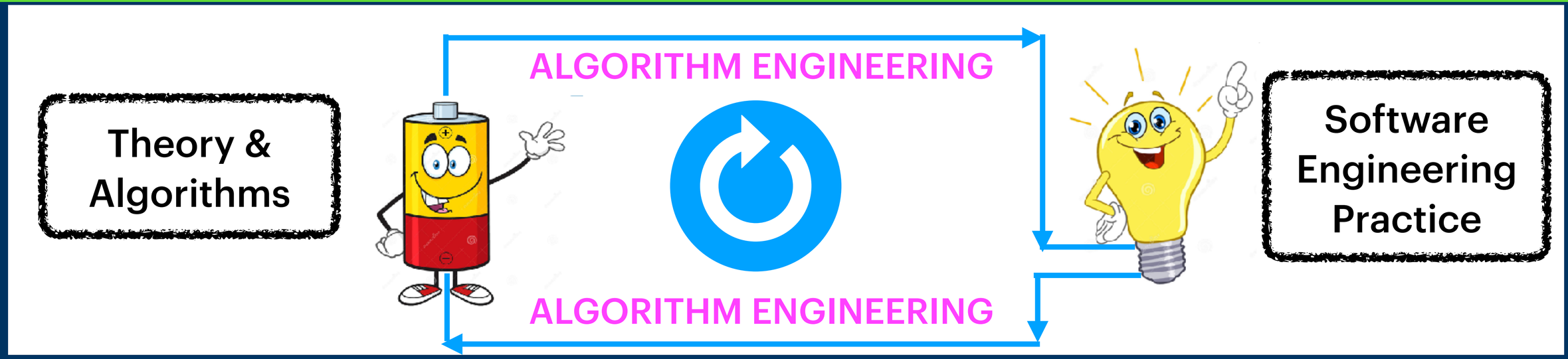
Actually I/O efficiency and PRAM models are similar and can simulate each other. (See section 5.4 on MS book)

For speedy programs, assume closest realistic model, and consider the capabilities of the computing architecture.

Reading Assignment

- Müller & Schira (MS) Chapter 5, Realistic computer models
-

NEXT LECTURE ...



ALGORITHM ENGINEERING

Lecture 4:
Design of Experiments

M. Oğuzhan Külekci - kulekci@itu.edu.tr