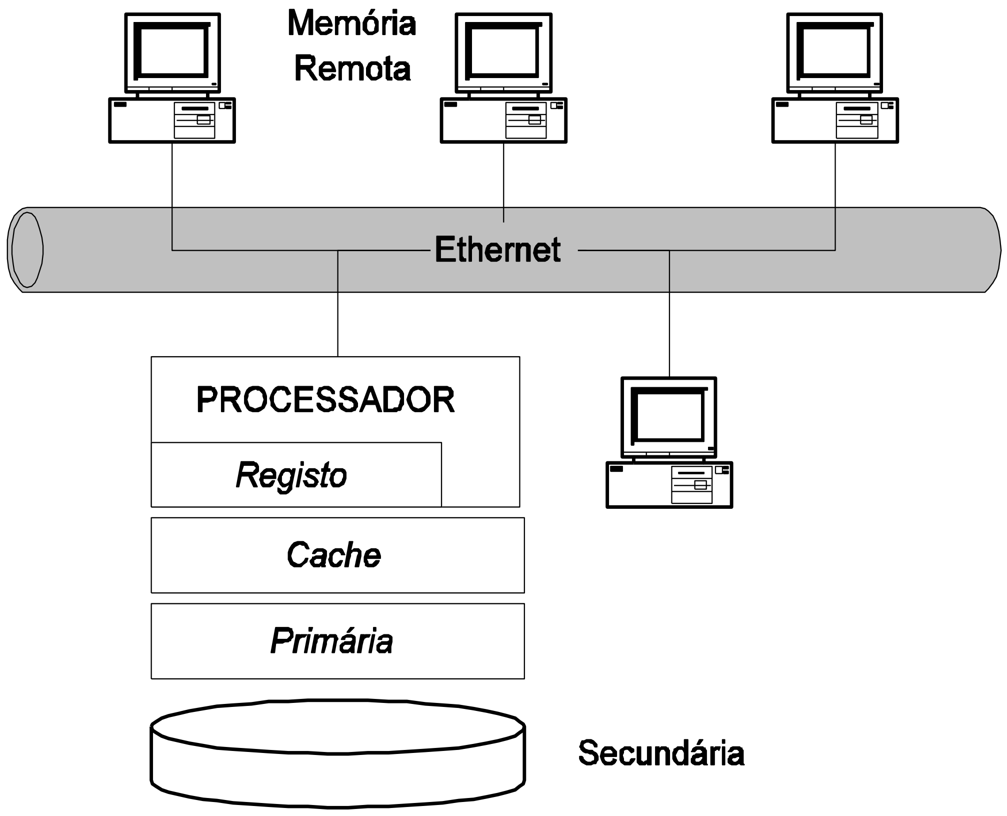
Cache Memory and its impact in the processor performance / data locality

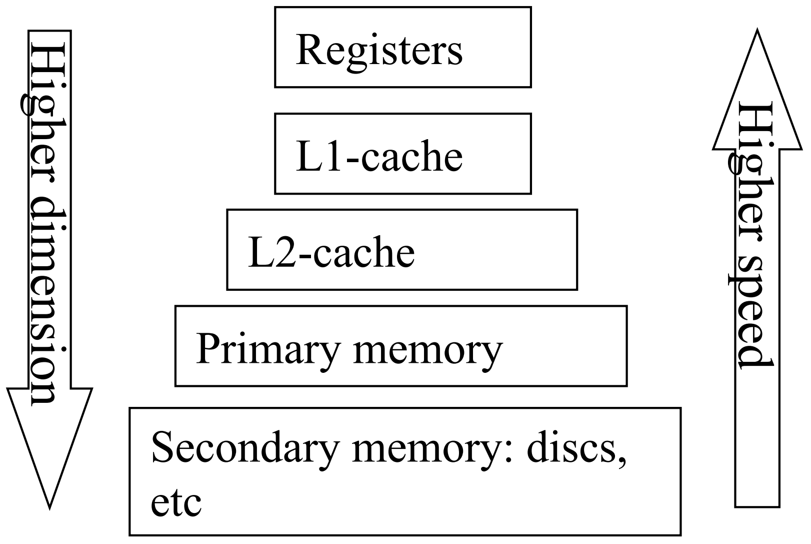
Jorge Barbosa

PCOM

Memory hierarchy

- The *CACHE* memory serves of interface between the processor and the main memory.
- Explores the temporal and spatial location of data.

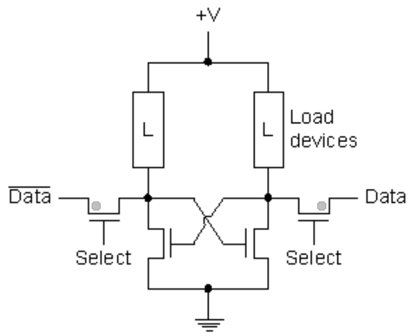




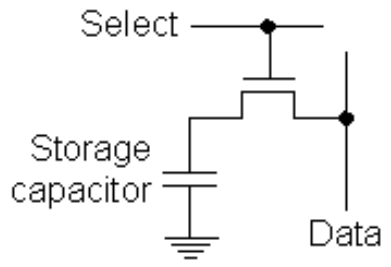
Memory technology

- Faster memories → are more expensive per bit because they require more area per bit

Memory Type	**Typical access time**
SRAM	0.5 – 2.5 ns
DRAM	50 – 70 ns
Magnetic disc	5,000,000 – 20,000,000 ns



Static Random Access Memory



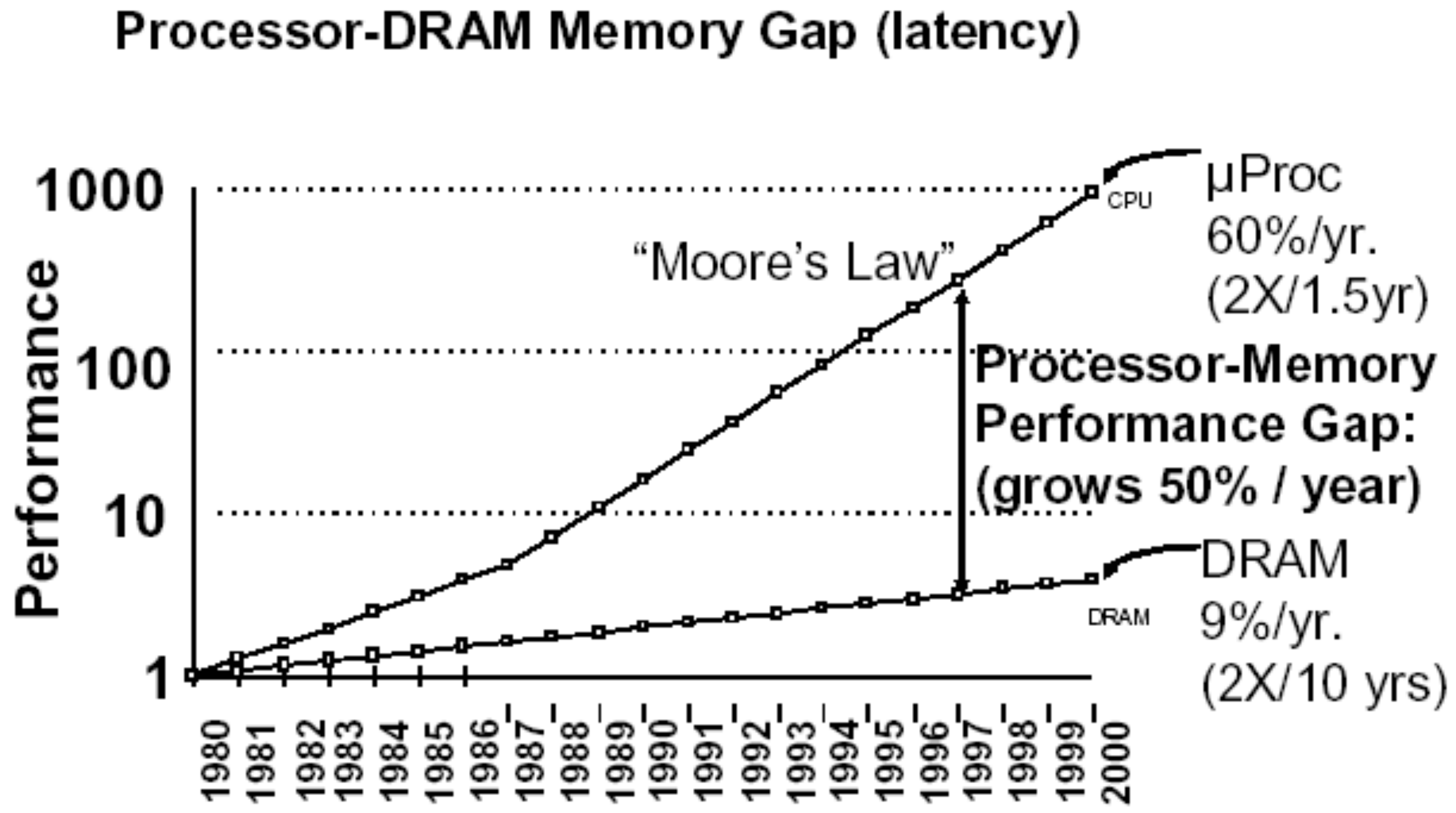
Dynamic Random Access Memory (DRAM, SDRAM, DDR1/2/3)

https://www.usenix.org/legacy/events/sec2001/full_papers/gutmann/gutmann_html/index.html

Cache Memory

- Facts:
  - Execution time = clock cycles executing user code + clock cycles for data transfer between cache and main memory.
  - For many years the increase rate of processor speed was significantly greater than the increase rate of memory speed.
- Therefore, the time to access memory is a bottleneck for the processor performance.
- Cache Memory:
  - Cache memory is on the first level of the memory hierarchy, small dimension, fast access (compared to main memory), and it has the function of decreasing the mean time to access memory.

Processor and memory evolution



Cache memory

- **Cache Hit:** The CPU requests data that are available in the cache. Hit Rate is the percentage of ‘cache hits’ over the total of data accesses.
- **Cache Miss:** The CPU requests data that is not in the cache. The fail time corresponds to the time required to transfer data to cache. It is dependent on the machine architecture.
- **Cache L1 and L2**: L1 Cache is in the same package as the processor and L2 Cache is installed in a separated package. L1 misses are faster to solve than L2 misses.

Cache Memory- concepts

CACHE memory was developed considering the next two key concepts:

- Spatial Location
  - When a data element is requested their neighbors will also be.
  - A cache line is read in a single operation.
  - It is efficient to request data elements from the same cache line.

- Temporal Location
  - When a data element is requested it has high probability to be requested in a short period of time.
  - The user should guaranty that the data that is in the Cache is used with more frequency.

Processor architecture

- Modern processors use a variety of techniques to increase performance:
  - Cache
  - Parallelism
    - Parallel functional units (hyperthreading)
  - Pipelining

What is the purpose of this study about cache memory?

- Compilers are sophisticated and can produce machine code that take advantage of these features and, therefore, can optimize user code;

but they cannot optimize the access pattern to data expressed by the programmer

Cache Coherence

- Multicore processors
  - Cores share a common address space
  - Caches are independent per core

- New problem: shared data

Time step	Event	Cache contents for CPU A	Cache contents for CPU B	Main memory Location X
0				0
1	CPU A reads X	0		0
2	CPU B reads X	0	0	0
3	CPU A stores 1 into X	1	0	1

Cache Coherence

A memory system is coherent if

- A read after a write of a location X from the same processor *P* must return the last write.
- A read by *P* after a write by *Q* must return the last write, if both instructions are sufficiently separated in time and no other write occurs by any processor.
- Writes to the same location are serialized.
  - Two writes from different processors are seen in the same order by all processors.

The *write-invalidate* protocol

- To enforce coherence
  - Ensure that a single processor has exclusive access to a memory location it wants to write.
  - The exclusive access invalidates all copies that may exist in other processors.
  - Other processors that had a copy of that memory location are forced to read the value again.
- Most protocols use blocks of data instead of single memory locations (can lead to *false sharing*).

The *write-invalidate* protocol

Time step	Event	BUS activity	Cache contents for CPU A	Cache contents for CPU B	Main memory Location X
0					0
1	CPU A reads X	Cache miss for X	0		0
2	CPU B reads X	Cache miss for X	0	0	0
3	CPU A stores 1 into X	Invalidation for X	1		0
4	CPU B reads X	Cache miss for X	1	1	1

When the 2nd cache miss occurs for CPU B, CPU A responds canceling the response from memory and updating both the cache of CPU B and main memory.

Ref: Computer Organization and Design, David Pattersson, John Hennessy