Computer Organization and Architecture

K. R. Kaza, IIIT Allahabad.

Memory

- Agenda
  - Memory Terminology
  - Memory Parameters
  - Memory Characteristics
  - Memory Hierarchy
  - Cache Memory
  - Memory Mapping

Memory Terminology

- Memory cell - A circuit that store a bit
- 1 bit memory cells:
- Latches, Flipflops
- Word
- Byte
- Address

Processor-Memory Perfomance Gap

- 1980 - no cache in microprocessors
- 1995 - L2 cache usage
- 2000's - widespread L3 cache usage
- Huge gap between the rate of increase in performance of CPU vs Memory.
- This is a performance bottleneck.

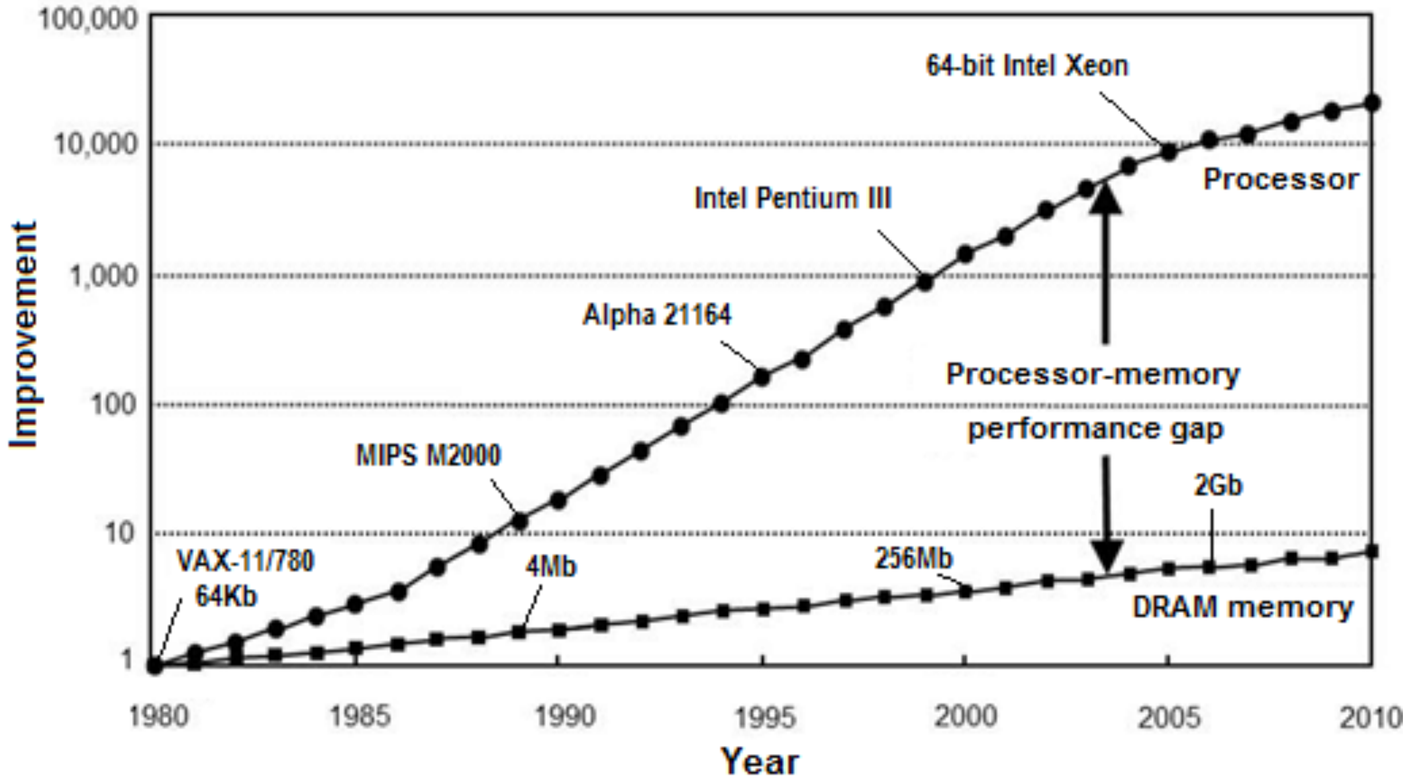


Image Source: Efnusheva et al. "A Survey of Different Approaches for Overcoming the Processor - Memory Bottleneck"

Memory

- Application programmers wish for very large and very fast memory.
- Very expensive and not always available

Memory

- Application programmers wish for very large and very fast memory.
- Very expensive and not always available
- However ...

Memory

- Application programmers wish for very large and very fast memory.
- Very expensive and not always available
- However, an illusion of very large and effective can be created using the following principles

Memory

- Application programmers wish for very large and very fast memory.
- Very expensive and not always available
- However, an illusion of very large and effective can be created using the following principles
- Implement a Memory hierarchy
-

Memory

- Application programmers wish for very large and very fast memory.
- Very expensive and not always available
- However, an illusion of very large and effective can be created using the following principles
- Implement a Memory hierarchy
- Use multiple levels of memory with differing sizes and speeds
-

Memory

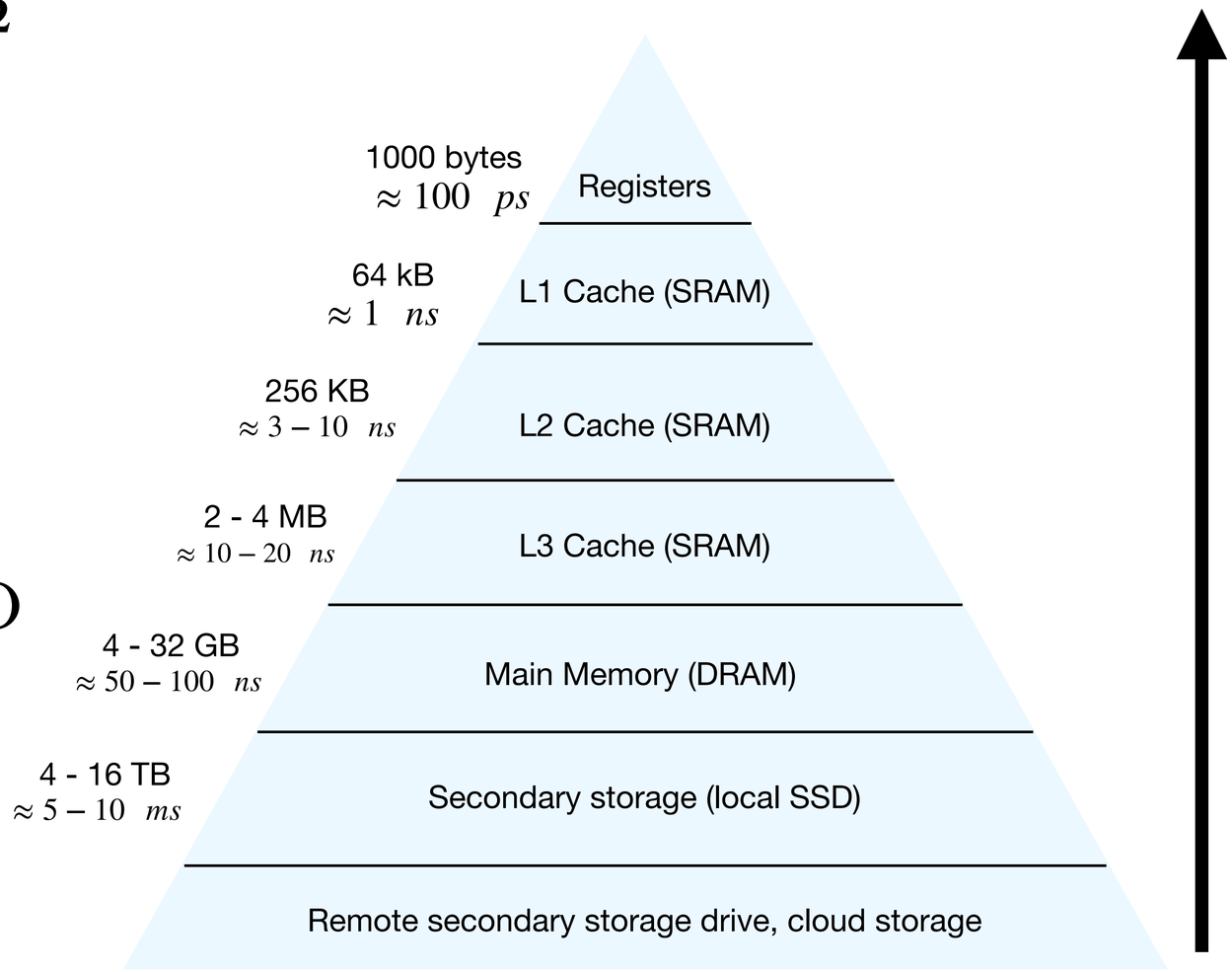
- Application programmers wish for very large and very fast memory.
- Very expensive and not always available
- However, an illusion of very large and effective can be created using the following principles
- Implement a Memory hierarchy
- Use multiple levels of memory with differing sizes and speeds
- Keep the faster and smaller memory closest to the processor
- Entire addressable memory space is available to the largest (slowest) memory type

Memory

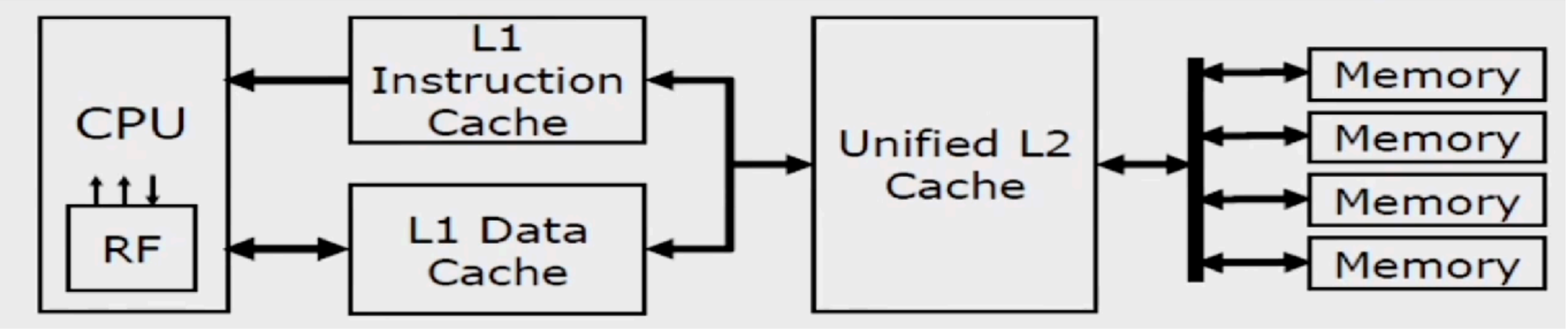
- Application programmers wish for very large and very fast memory.
- Very expensive and not always available
- However, an illusion of very large and effective can be created using the following principles
- Implement a Memory hierarchy
- Use multiple levels of memory with differing sizes and speeds
- Keep the faster and smaller memory closest to the processor
-

Memory Hierarchy

- Last level cache (LLC) can be either L2 or L3
- Registers and Cache are on the processor chip
- Main memory is connected by the memory bus
- Secondary storage is connected by I/O bus



Memory Hierarchy



- Split instruction and data cache at L1
- Unified cache at L2
- Main memory (DRAM) divided into multiple interleaved memory banks

Cache Memory

- A **small and fast** buffer between the processor and main memory
- Old data is be removed, space is made for new values.
-

Cache Memory

- A **small and fast** buffer between the processor and main memory
- Old data is be removed, space is made for new values.
- Using some basic principles:

Cache Memory

- A **small and fast** buffer between the processor and main memory
- Old data is be removed, space is made for new values.
- Using some basic observations/principles:
  - Principle of locality
  - Temporal locality
  - Spatial locality

Cache Memory

- A **small and fast** buffer between the processor and main memory
- Old data is be removed, space is made for new values.
- Using some basic observations/principles:
  - Principle of locality: At any instant, programs access a relatively small portion of their address space.
  - Temporal locality
  - Spatial locality

Cache Memory

- A **small and fast** buffer between the processor and main memory
- Old data is be removed, space is made for new values.
- Using some basic observations/principles:
  - Principle of locality: At any instant, programs access a relatively small portion of their address space.
  - Temporal locality:
  - Spatial locality

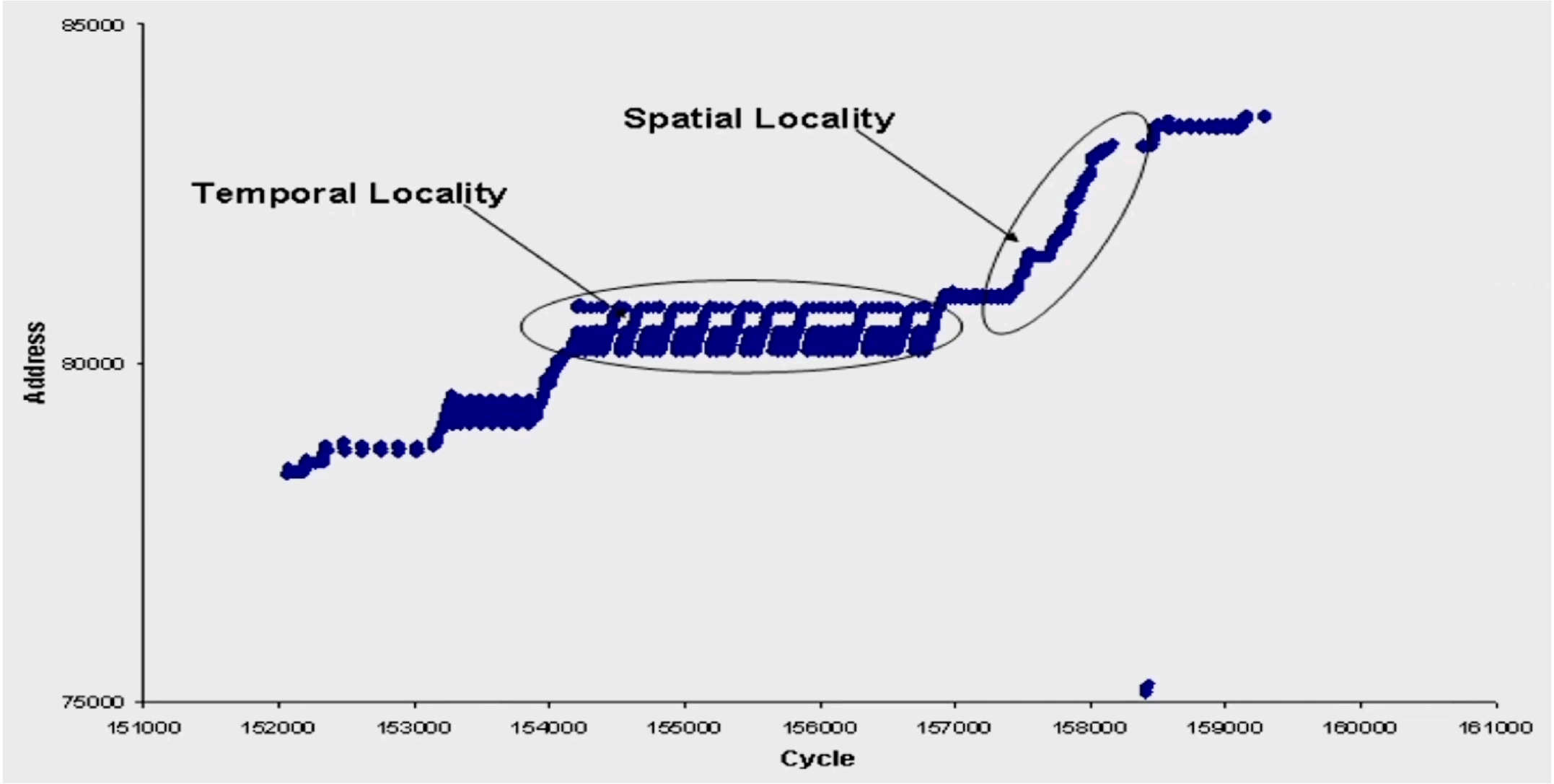
Cache Memory

- A **small and fast** buffer between the processor and main memory
- Old data is be removed, space is made for new values.
- Using some basic observations/principles:
  - Principle of locality: At any instant, programs access a relatively small portion of their address space.
  - Temporal locality: Referenced items tend to be used again soon.
  - Spatial locality: Items close (in address) to referenced items tend to be referenced soon.

Cache Memory

- A **small and fast** buffer between the processor and main memory
- Old data is be removed, space is made for new values.
- Using some basic observations/principles:
  - Principle of locality: At any instant, programs access a relatively small portion of their address space.
  - Temporal locality: Referenced items tend to be used again soon.
  - Spatial locality: Items close (in address) to referenced items tend to be referenced soon.

Memory Access Patterns



Cache Terminology

- Block/Line : Minimum unit of information that can be either present or not in a cache level
- Hit : the requested data is present in the cache

Cache Terminology

- Block/Line : Minimum unit of information that can be either present or not in a cache level
- Hit : the requested data is present in the cache
- Miss : the requested requested by the processor is not present in the cache

Cache Terminology

- Block/Line : Minimum unit of information that can be either present or not in a cache level
- Hit : the requested data is present in the cache
- Miss : the requested requested by the processor is not present in the cache
- Hit Time: Time taken to access the cache memory block and return the data to the processor.
-

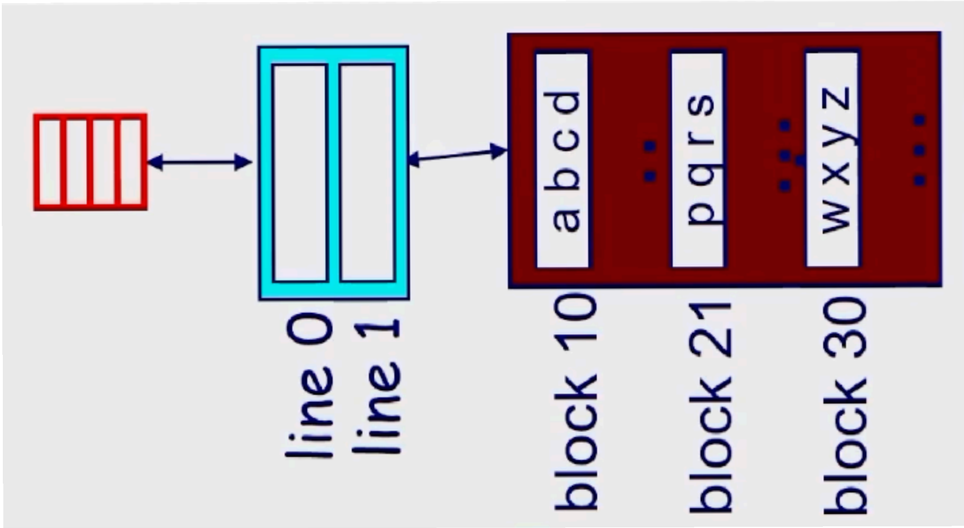
Cache Terminology

- Block/Line : Minimum unit of information that can be either present or not in a cache level
- Hit : the requested data is present in the cache
- Miss : the requested requested by the processor is not present in the cache
- Hit Time: Time taken to access the cache memory block and return the data to the processor.
- Hit rate: Fraction of memory access foundin the cache
- Miss rate: Fraction of memory access not found in the cache
-

Cache Terminology

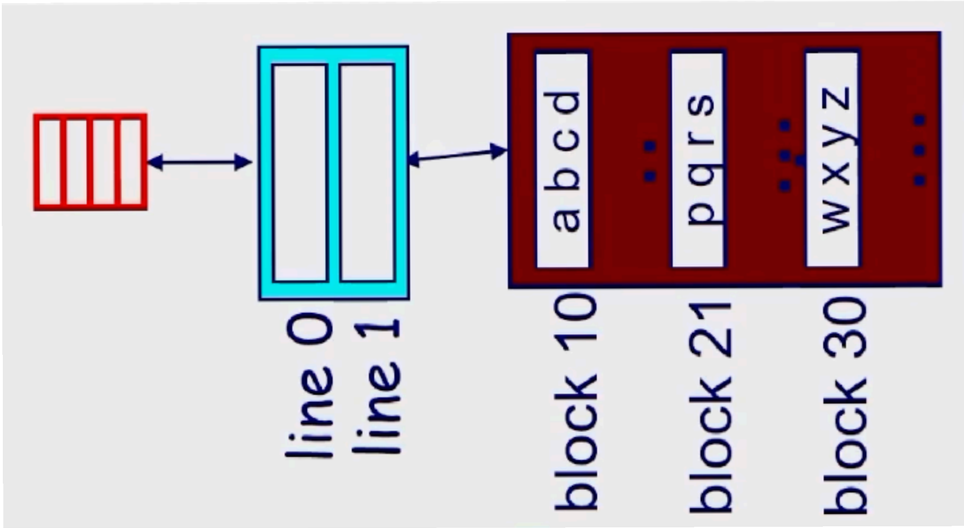
- Block/Line : Minimum unit of information that can be either present or not in a cache level
- Hit : the requested data is present in the cache
- Miss : the requested requested by the processor is not present in the cache
- Hit Time: Time taken to access the cache memory block and return the data to the processor.
- Hit rate: Fraction of memory access foundin the cache
- Miss rate: Fraction of memory access not found in the cache
- Miss penalty : Time to replace a block in the cache with the corresponding block from the next level.

Processor-Cache Interaction



- The transfer unit between the CPU register file and the cache is a 4-byte word
- The transfer unit between the cache and main memory is a 4-word block (16B)

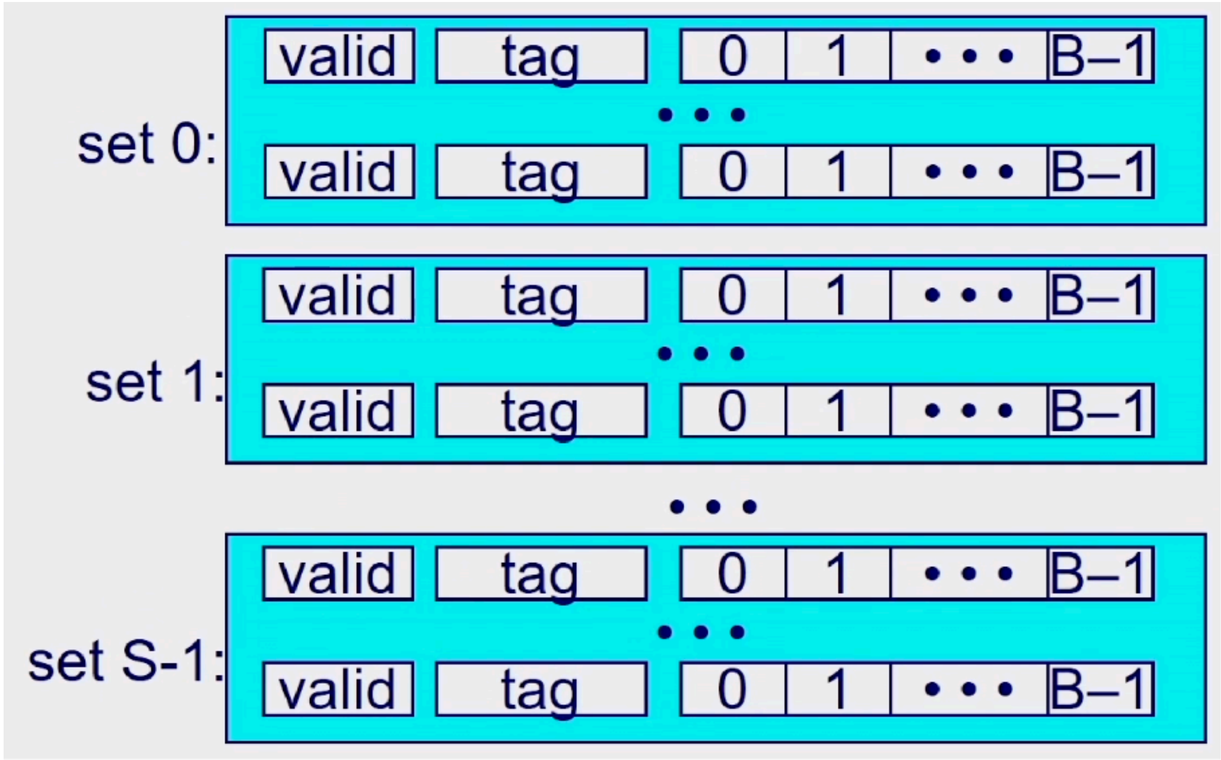
Processor-Cache Interaction



- The transfer unit between the CPU register file and the cache is a 4-byte word
- The transfer unit between the cache and main memory is a 4-word block (16B)
- The tiny, very fast CPU register file has room for four 4-byte words
- The small fast L1 cache has room for two 4-word blocks
- The big slow main memory has room for many 4-word blocks

Cache Organization

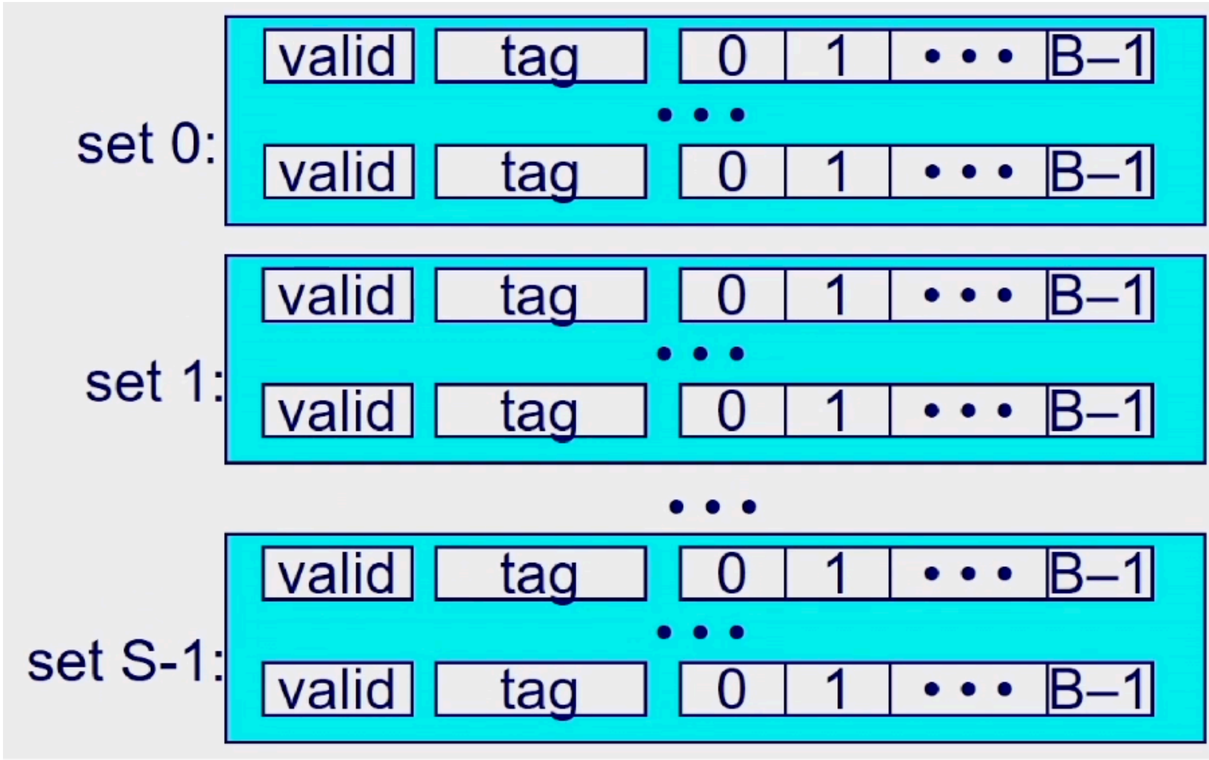
- Cache is an array of **S** sets
- Each set is an array of **E** lines
- Each line has a
  - Valid bit
  - *t* tag bits
  - B $= 2^b$ bytes of memory
- CacheSize : C = B x E x S bytes



Cache Addressing

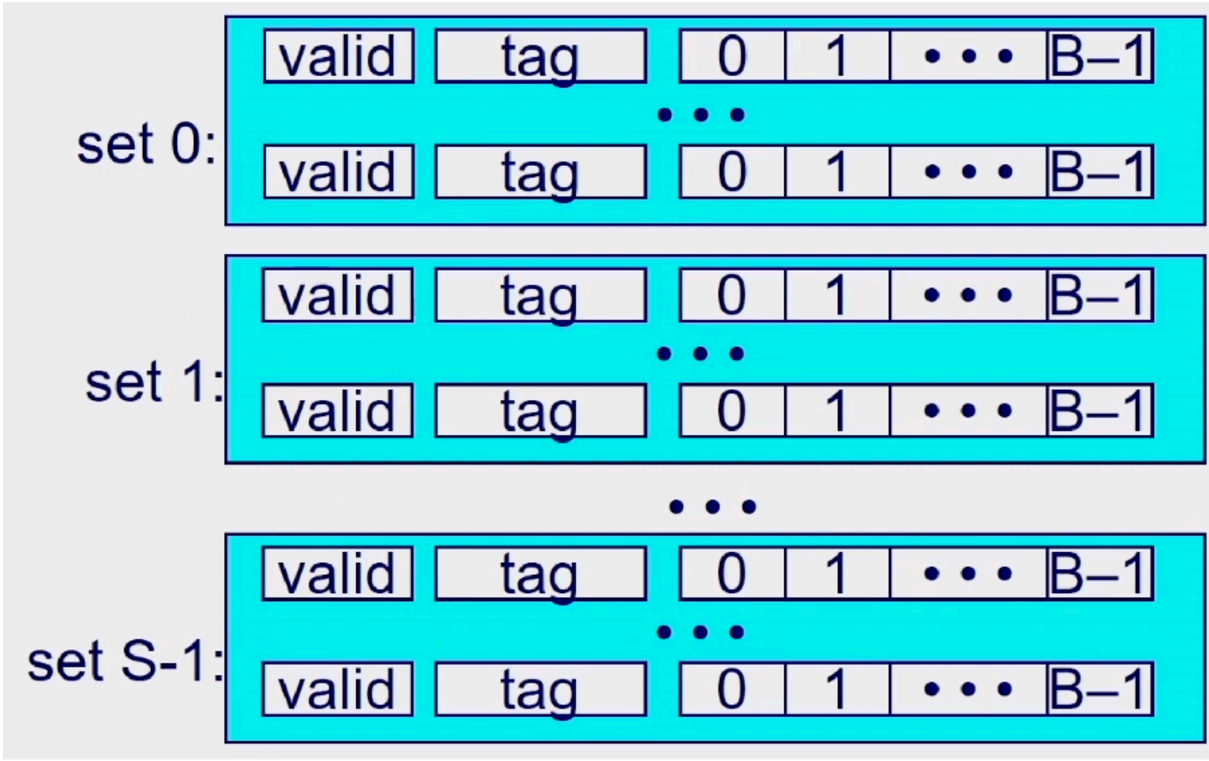
- Suppose, CPU wants Address A
- Address A:
- 

- The word at address A is in the cache if the tag bits in <valid> lines in <set index> matches the <tag>



Cache Addressing

- Suppose, CPU wants Address A
- Locate the set based on <set index>
- Locate the line in the set based on <tag>
- Check that the line is valid
- Locate the data in the line based on <block offset>
-



Numerical Example 1

- A cache has 512 KB capacity, 4B word, 64B block size and 8-way set associative. The system is using 32 bit address. Given the address OX ABC89984, which set of cache will be searched and specify which word of the selected cache block will be forwarded if it is a hit in cache?
-

Numerical Example 1

- A cache has 512 KB capacity, 4B word, 64B block size and 8-way set associative. The system is using 32 bit address. Given the address OX ABC89984, which set of cache will be searched and specify which word of the selected cache block will be forwarded if it is a hit in cache?
- E = 8 blocks in a given set
- # of sets = CacheSize/(BlockSize x E) = $2^{19}/(2^6 \times 2^3) = 2^{10} = 1024$ sets
-

Numerical Example 1

- A cache has 512 KB capacity, 4B word, 64B block size and 8-way set associative. The system is using 32 bit address. Given the address OX ABC89984, which set of cache will be searched and specify which word of the selected cache block will be forwarded if it is a hit in cache?
- E = 8 blocks in a given set
- # of sets = CacheSize/(BlockSize x E) = $2^{19}/(2^6 \times 2^3) = 2^{10} = 1024$ sets
- 1 word = 4B. Hence, 64 byte block has 16 words
- Tag = 16 bits | Index = 10 bits | Offset = 6 bits
- Ox ABC8**9984** = *1001 1001 10*00 0100 → Set *614*, word 1

Memory Interleaving

- Divide the memory into banks
-
- Parallel Access: By splitting data across banks, one bank can be accessed (reading/writing) while others are preparing, greatly increasing bandwidth.
-
- Addressing Mechanism: Memory addresses are mapped to different banks

Thank you !