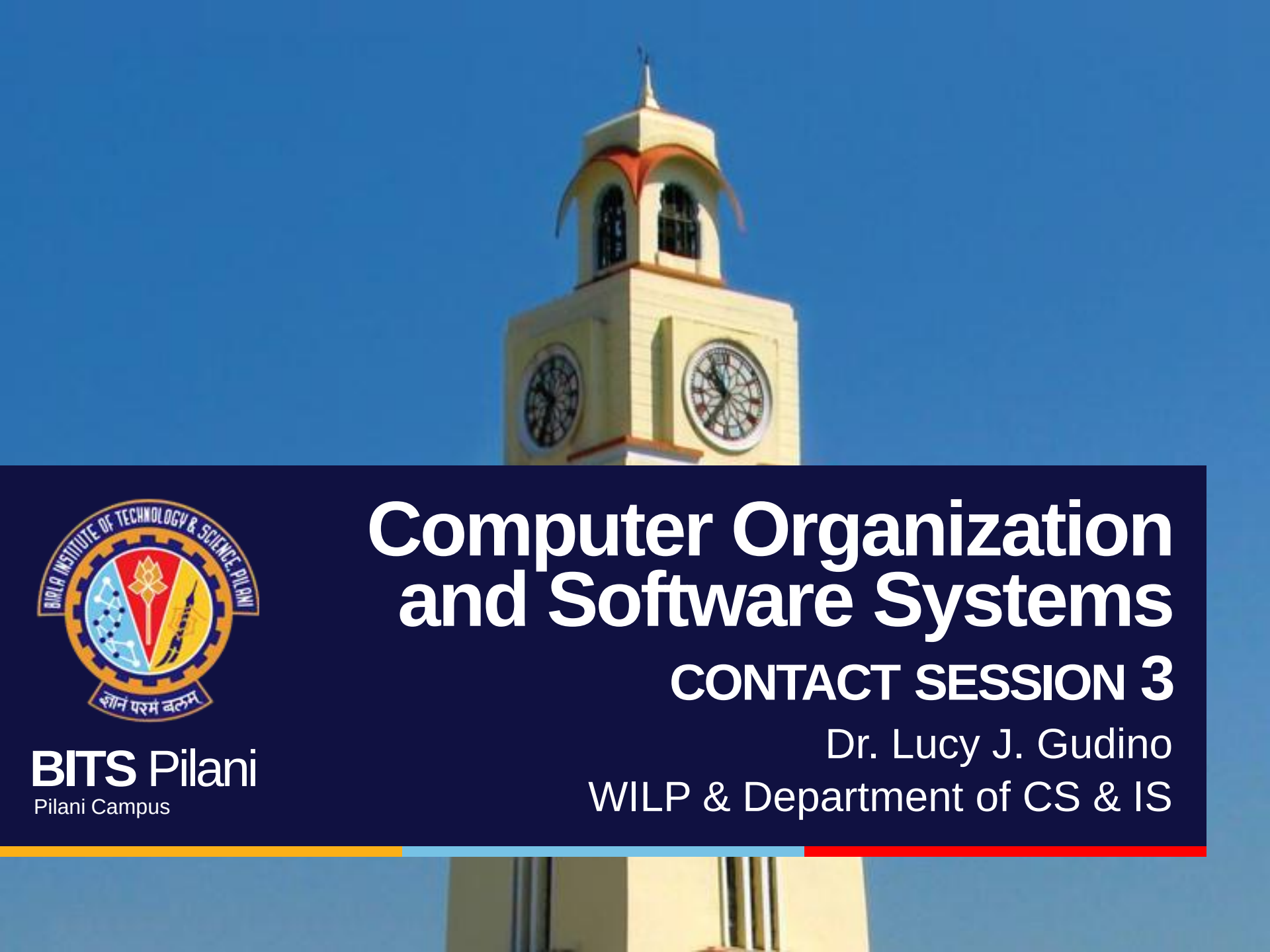


Computer Organization and Software Systems

CONTACT SESSION 3

Dr. Lucy J. Gudino
WILP & Department of CS & IS

BITS Pilani
Pilani Campus

Today's Session

Contact Hour	List of Topic Title	Text/Ref Book/external resource

5-6	**Memory Organization (Contd..)**
	• Solid State Devices
	• Locality
	  • Locality of Reference to Program Data
	  • Locality of instruction fetches
	• Memory Hierarchy
	• Cache Memories
	  • Generic Cache Memory Organization
	  • Direct-Mapped Caches
	  • Fully Associative Caches
		T1

Disk Access Time

- Average time to access some target sector given by :

  $T_{access} = T_{avg\_seek} + T_{avg\_rotation} + T_{avg\_transfer}$

- Seek time (T_{avg_seek})
  - Time to position heads over cylinder containing target sector.
  - Typical T_{avg_seek} is 3—9 ms
- Rotational latency ($T_{avg\_rotation}$)
  - Time required for first bit of target sector to pass under r/w head.
  - $T_{avg\_rotation}$ = 1/2r , where r is rotation Speed in **revolution per Second**
  - $T_{avg\_rotation}$ = 1/2r , given 7200 RPM then r = 7200/60 RPS
- Transfer time ($T_{avg\_transfer}$)
  - Time to read the bits in the target sector.
  - $T_{avg\_transfer}$ = b/rN, where b is the number of bytes to be transferred and N is the average number of bytes on a track

Disk Access Time Example

Given:

- Rotational rate = 7,200 RPM
- Average seek time = 9 ms.
- Avg # sectors/track = 400.
- 512 bytes per sector

Derived:

- Tavg rotation = 1/2r = 1/2 x (60 secs/7200 RPM)
  = 0.00416 = 4.16ms.
- Tavg transfer =b/rN
  = 512 x 60/7200 x 1/(400*512)
  = 0.02 ms
- Taccess = 9 ms + 4.16ms + 0.02 ms

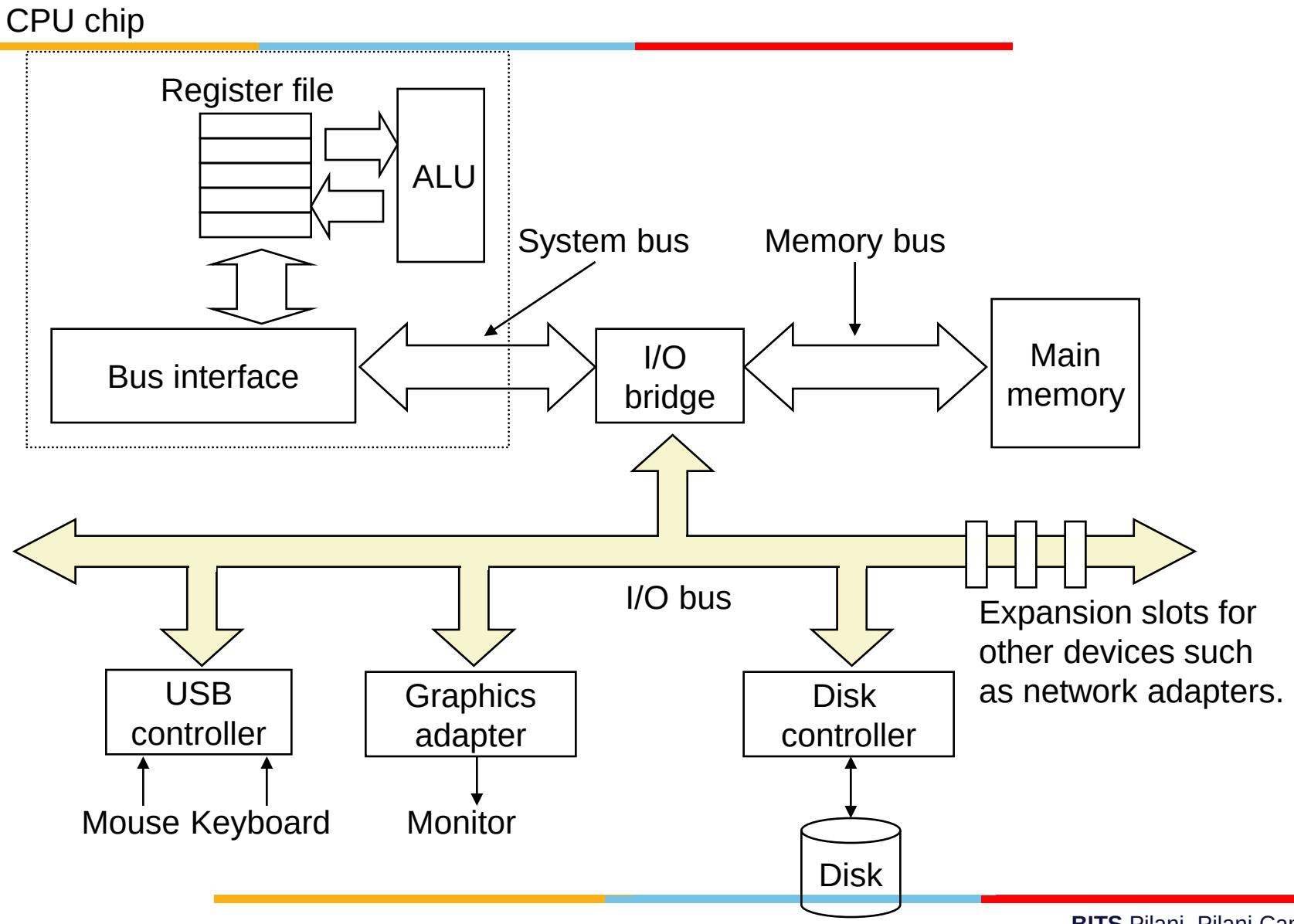
Important points

- Access time dominated by seek time and rotational latency.
- First bit in a sector is the most expensive, the rest are free.
- SRAM access time is about 4 ns/doubleword, DRAM about 60 ns
  - Disk is about 40,000 times slower than SRAM,
  - 2,500 times slower then DRAM.

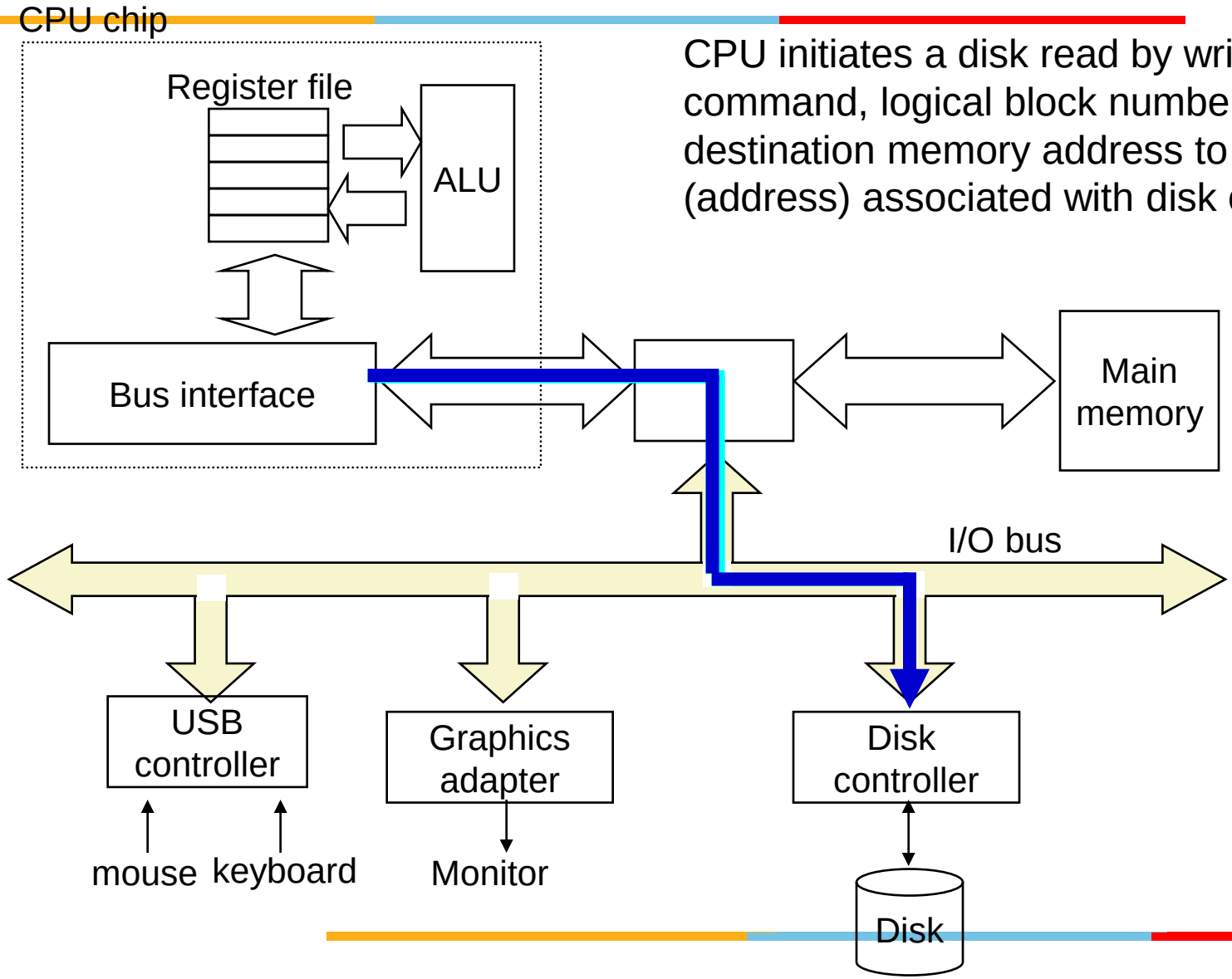
Logical Disk Blocks

- Modern disks present a simpler abstract view of the complex sector geometry:
  - The set of available sectors is modeled as a sequence of logical blocks (0, 1, 2, ...B-1)
- Mapping between logical blocks and actual (physical) sectors
  - Maintained by hardware/firmware device called disk controller.
  - Converts requests for logical blocks into (surface, track,sector) triples.
- Allows controller to set aside spare cylinders for each zone.
  - Accounts for the difference in "formatted capacity" and "maximum capacity".

I/O Bus

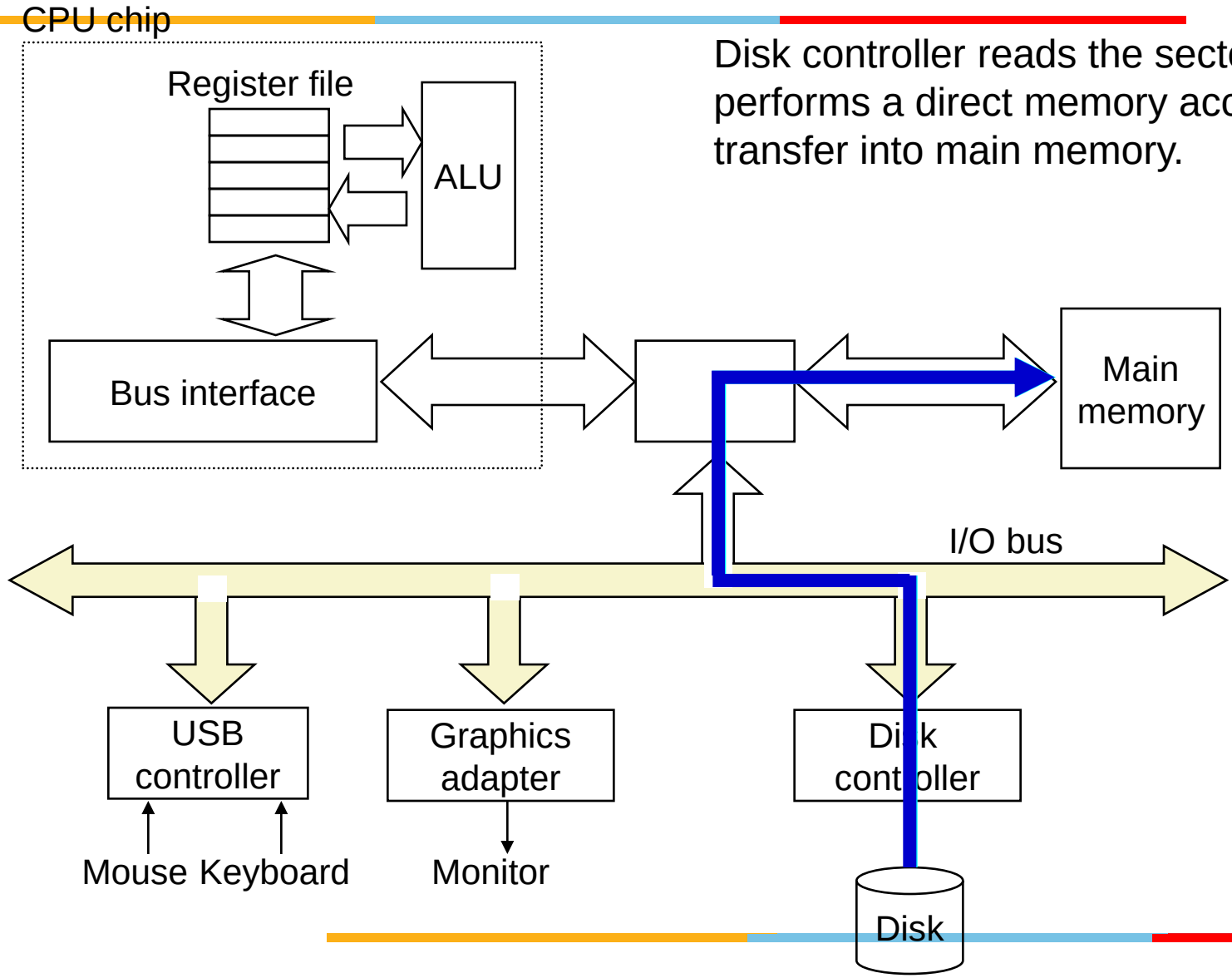

CPU chip
Register file
ALU
System bus
Memory bus
Bus interface
I/O bridge
Main memory
I/O bus
Expansion slots for other devices such as network adapters.
USB controller
Graphics adapter
Disk controller
Mouse
Keyboard
Monitor
Disk

Reading a Disk Sector (1)



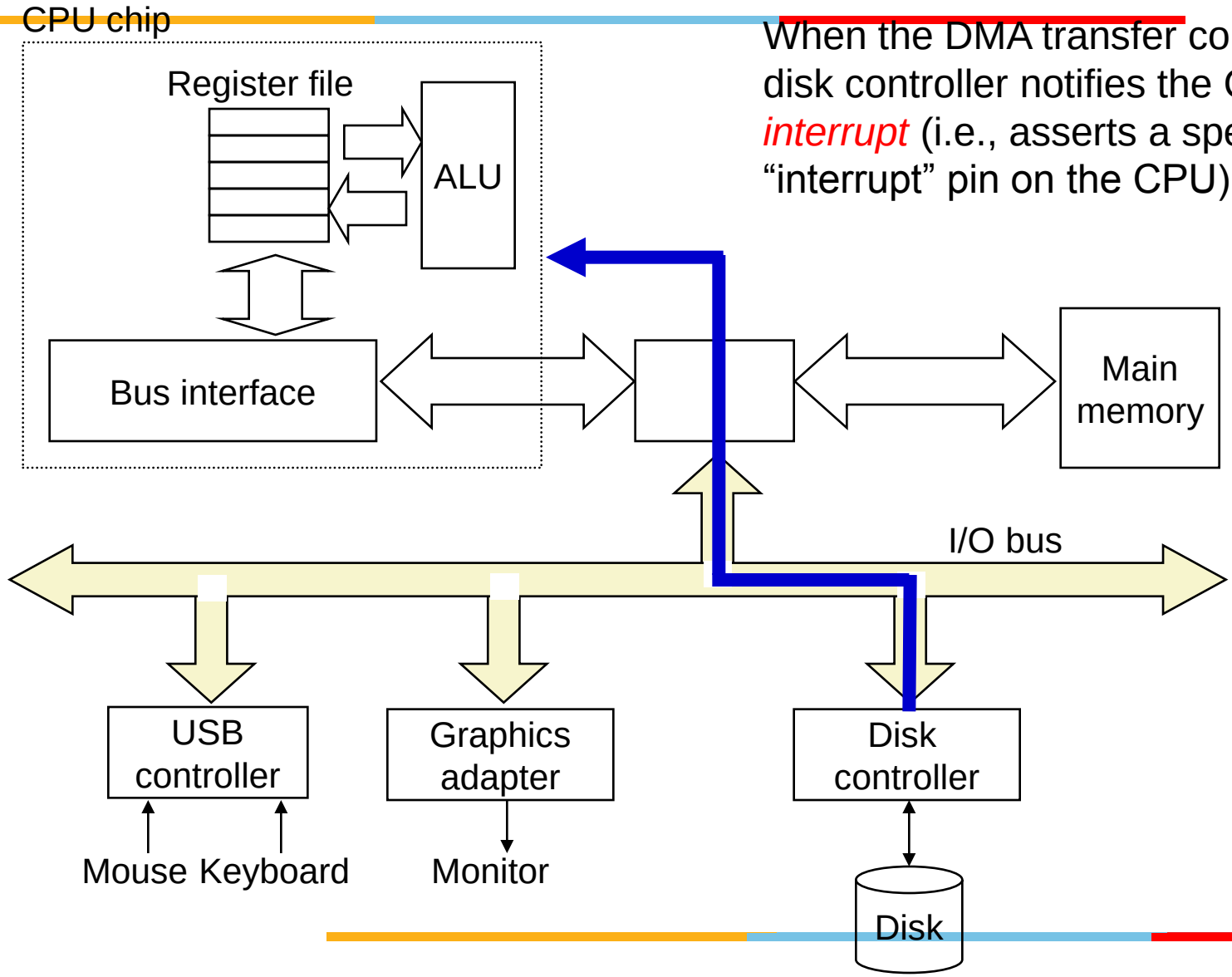
CPU initiates a disk read by writing a command, logical block number, and destination memory address to a port (address) associated with disk controller.

Reading a Disk Sector (2)



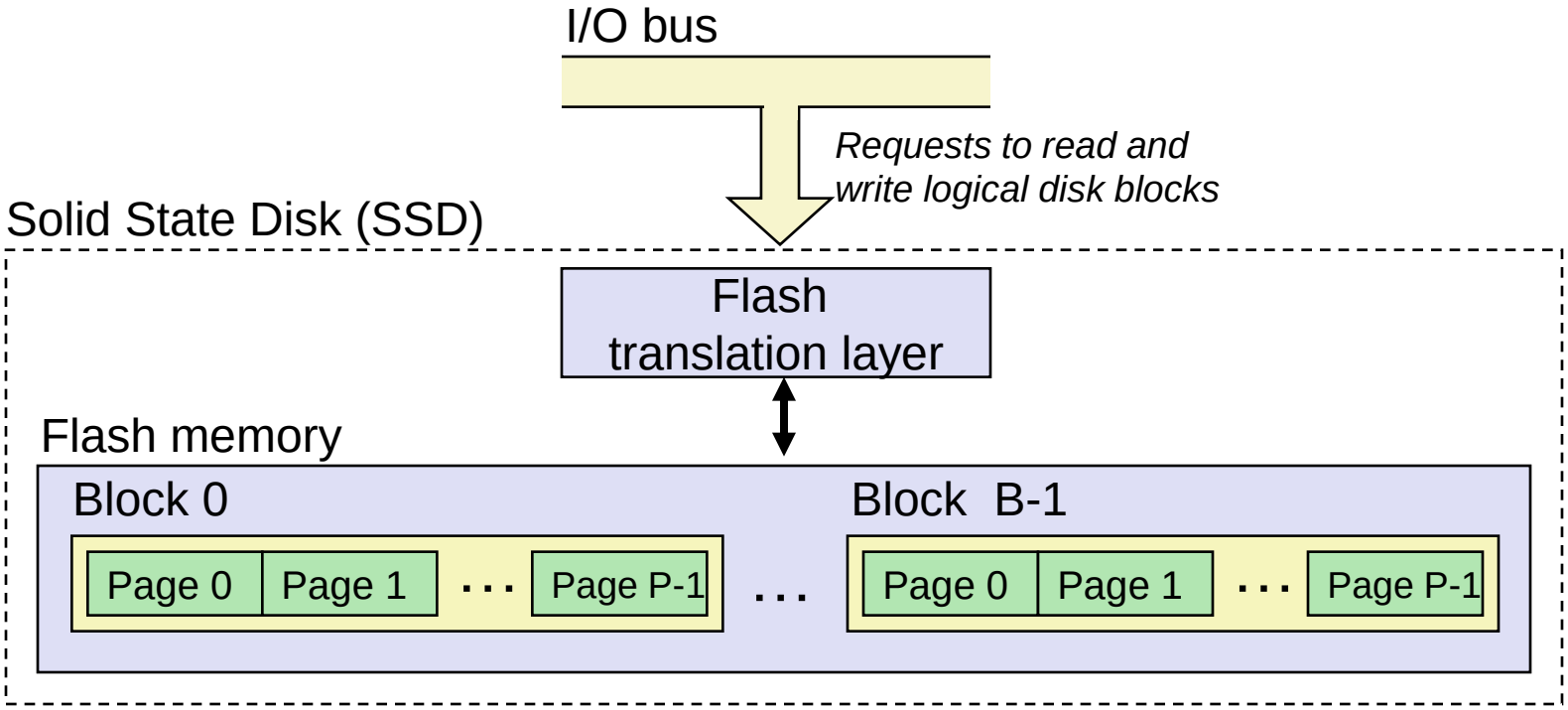
Disk controller reads the sector and performs a direct memory access (DMA) transfer into main memory.

Reading a Disk Sector (3)



When the DMA transfer completes, the disk controller notifies the CPU with an *interrupt* (i.e., asserts a special "interrupt" pin on the CPU)

Solid State Disks (SSDs)



- Pages: 512KB to 4KB, Blocks: 32 to 128 pages
- Data read/written in units of pages.
- Page can be written only after its block has been erased
- A block wears out after about 100,000 repeated writes.

SSD Performance Characteristics

Sequential read tput*	550 MB/s	Sequential write tput	470 MB/s
Random read tput	365 MB/s	Random write tput	303 MB/s
Avg seq read time	50 us	Avg seq write time	60 us

Tput → Throughput

- Sequential access faster than random access
  - Common theme in the memory hierarchy
- Random writes are somewhat slower
  - Erasing a block takes a long time (~1 ms)
  - Modifying a block page requires all other pages to be copied to new block
  - In earlier SSDs, the read/write gap was much larger.

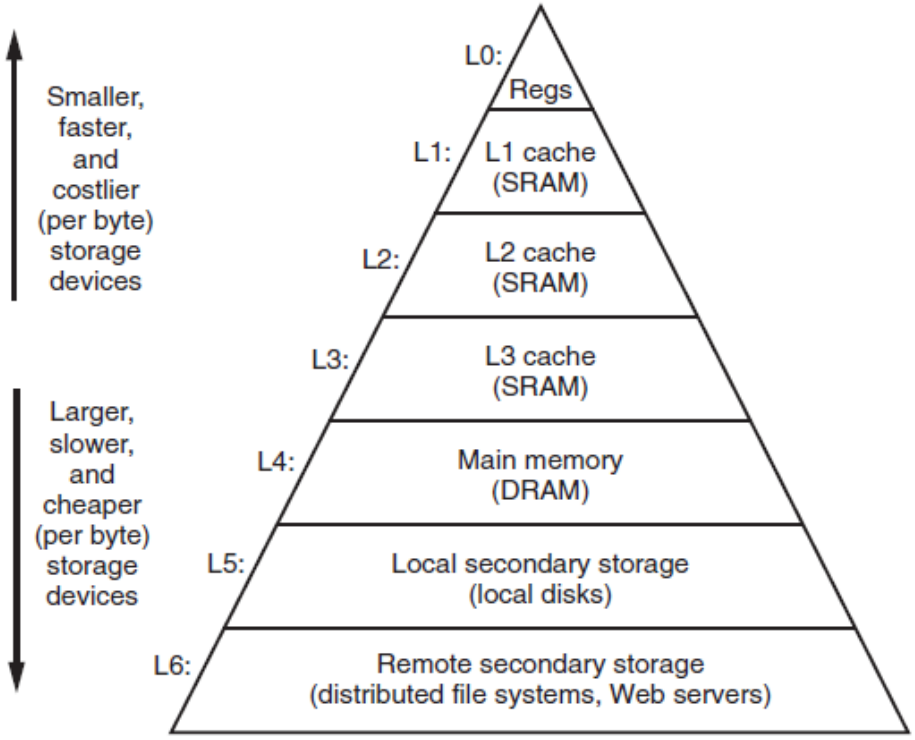
Source: Intel SSD 730 product specification.

SSD Tradeoffs vs Rotating Disks

- Advantages
  - No moving parts → faster, less power, more rugged
- Disadvantages
  - Have the potential to wear out
    - Mitigated by "wear leveling logic" in flash translation layer
    - E.g. Intel SSD 730 guarantees 128 petabyte (128 x 10^{15} bytes) of writes before they wear out
  - In 2015, about 30 times more expensive per byte
- Applications
  - MP3 players, smart phones, laptops
  - Beginning to appear in desktops and servers

The Bottom Line

- How much?
  - Capacity
- How fast?
  - Time is money
- How expensive?



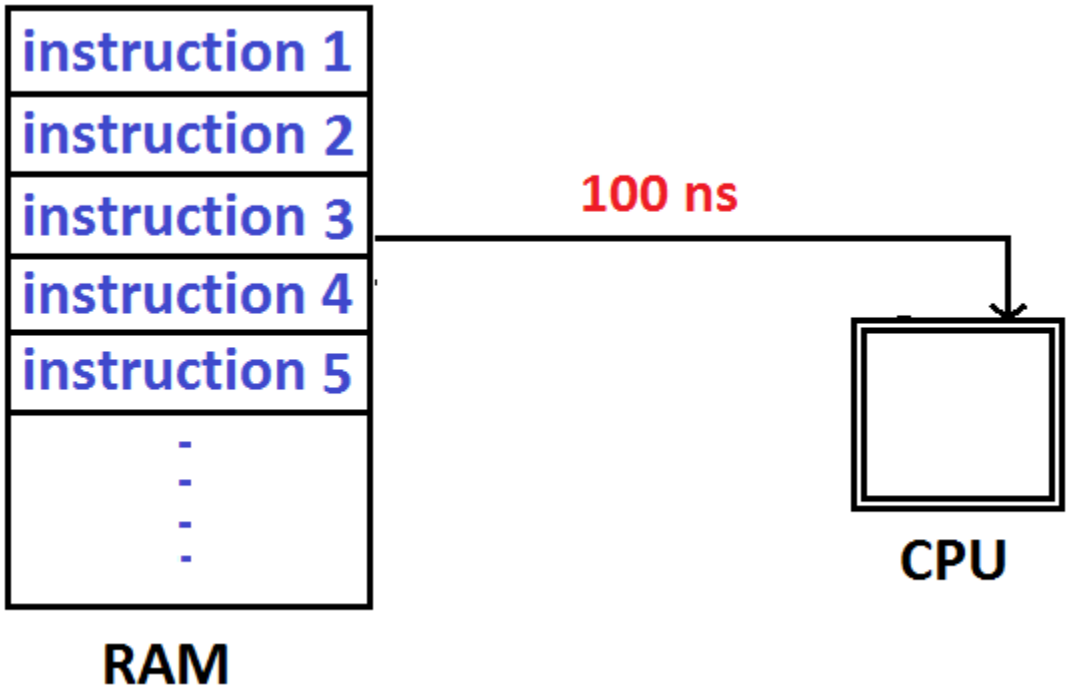
An example of a memory hierarchy.

- Faster access time , ---------- cost per bit
- Greater capacity, ---------- cost per bit
- Greater capacity, ----------access time

Memory Hierarchy

- Registers
  - In CPU
- Internal or Main memory
  - May include one or more levels of cache
  - "RAM"
- External memory
  - Backing store

Performance enhancement - Motivation

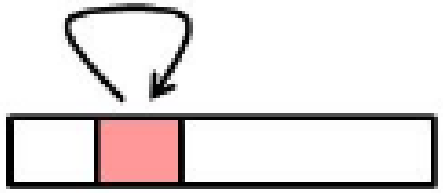


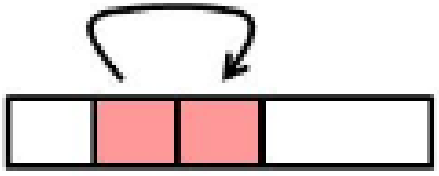
Performance enhancement - Motivation

Locality of Reference

During the course of the execution of a program, memory references tend to cluster

- **Temporal locality:** Locality in time
  - If an item is referenced, it will tend to be referenced again soon
- **Spatial locality:** Locality in space
  - If an item is referenced, items whose addresses are close by will tend to be referenced soon.





Example

```
product  = 1;
for ( i = 0; i < n-1; i++)
      product = product *  a[i] ;
```

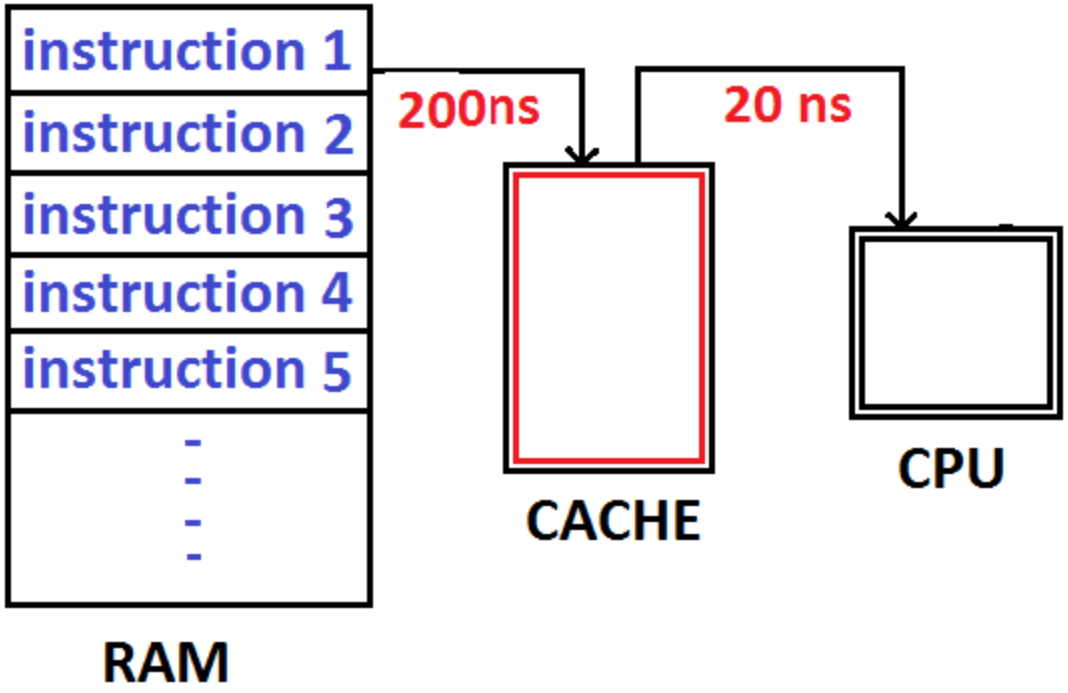
Data :

- Access array elements in succession – spatial locality
- Reference to "product" in each iteration – Temporal locality

Instructions :

- Reference instructions in sequence : Spatial locality
- Looping through : Temporal locality

Performance enhancement - Motivation

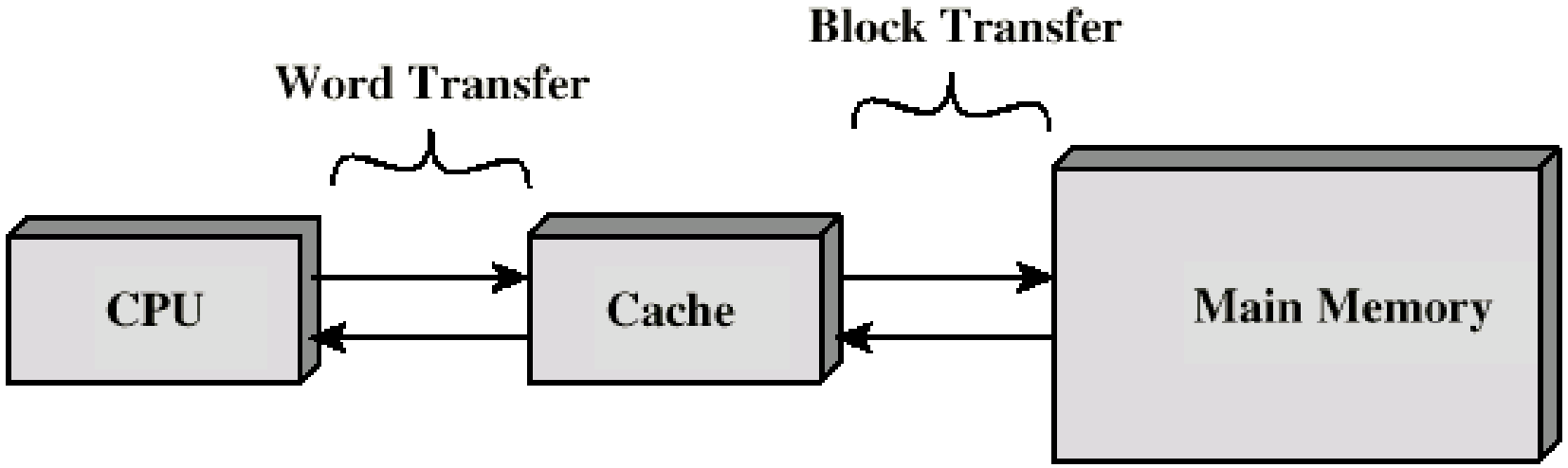


Cache

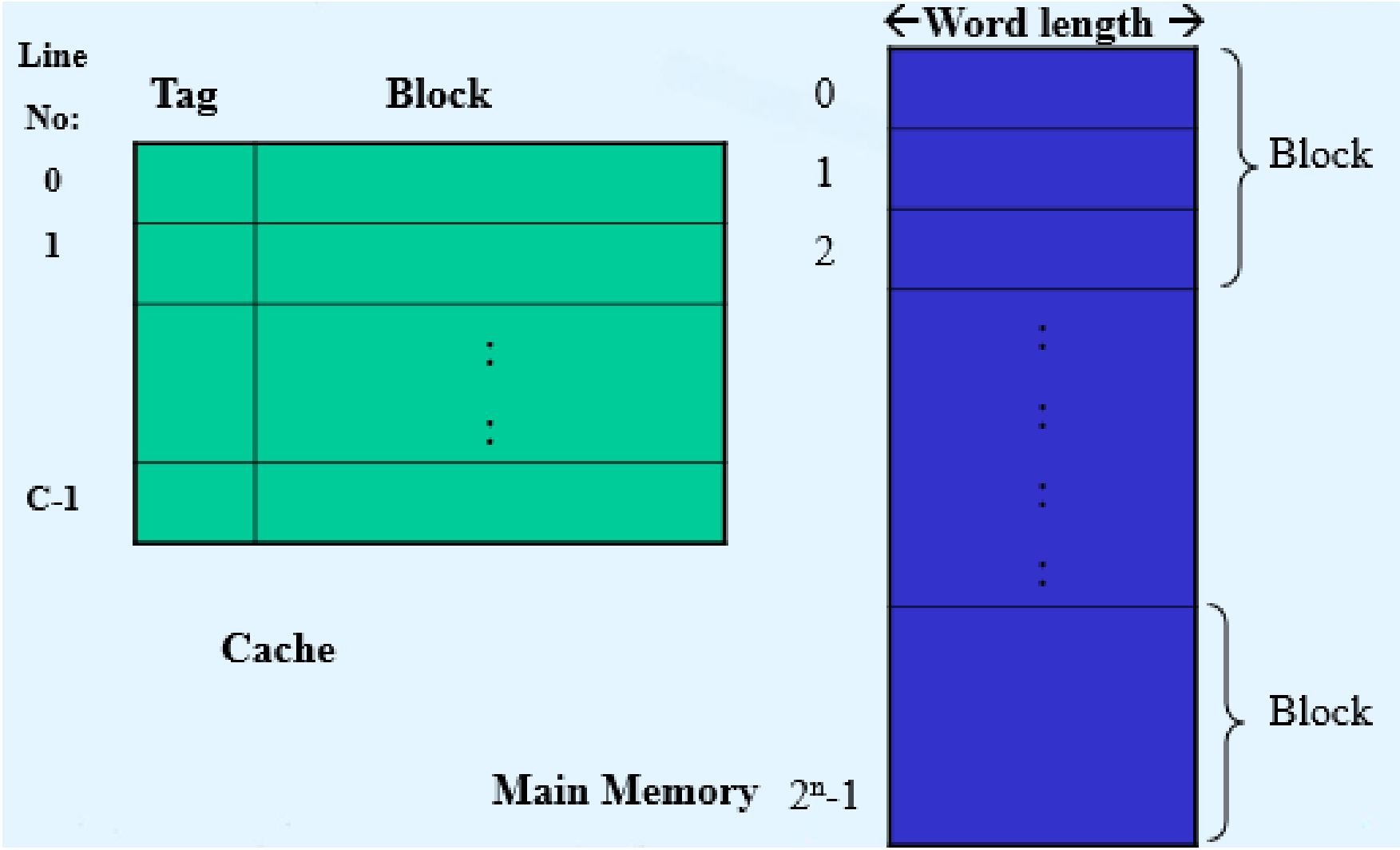
BITS Pilani
Pilani Campus

Cache

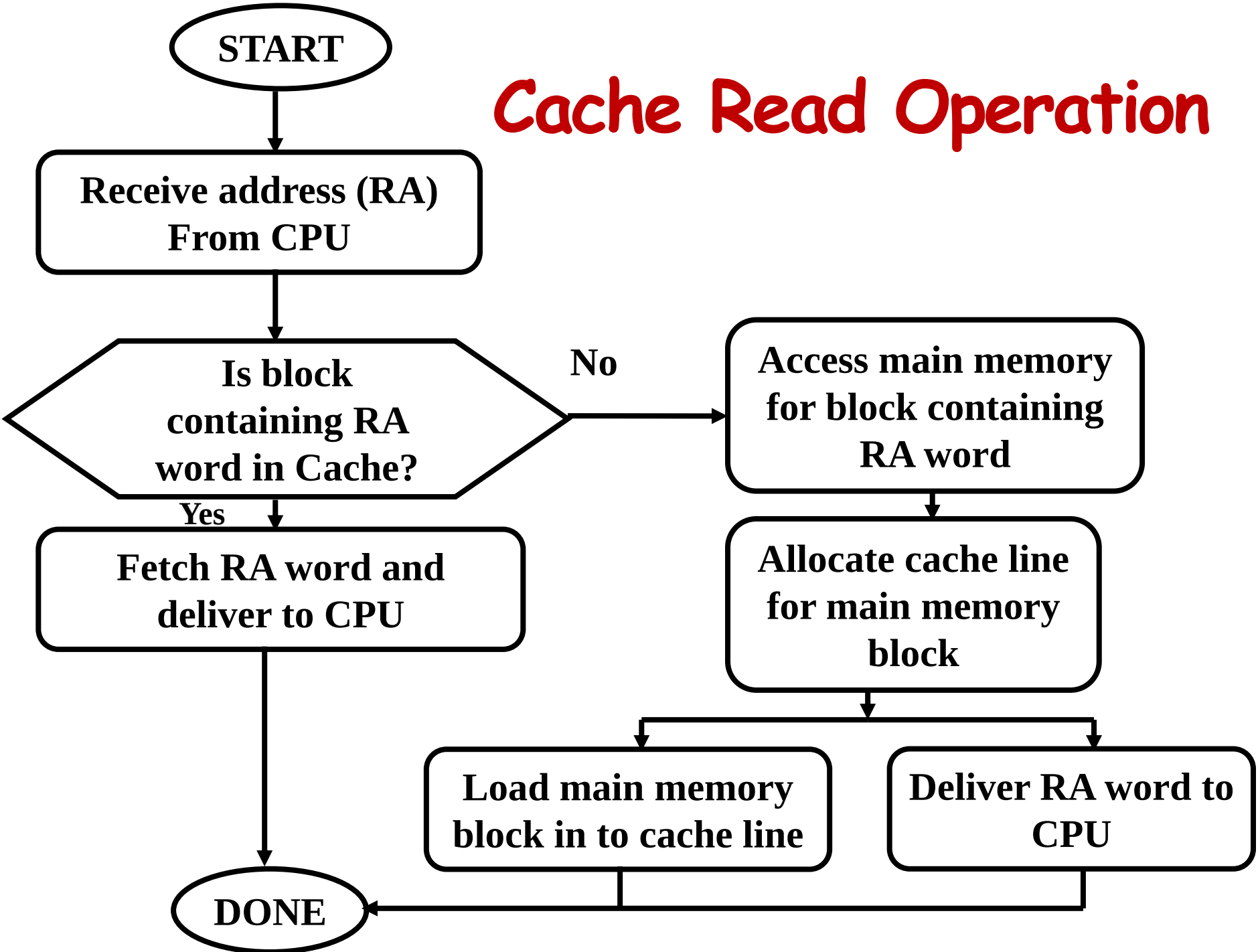
- Small, fast memory
- Sits between normal main memory and CPU
- May be located on CPU chip or separate module



Cache and Main Memory Structure



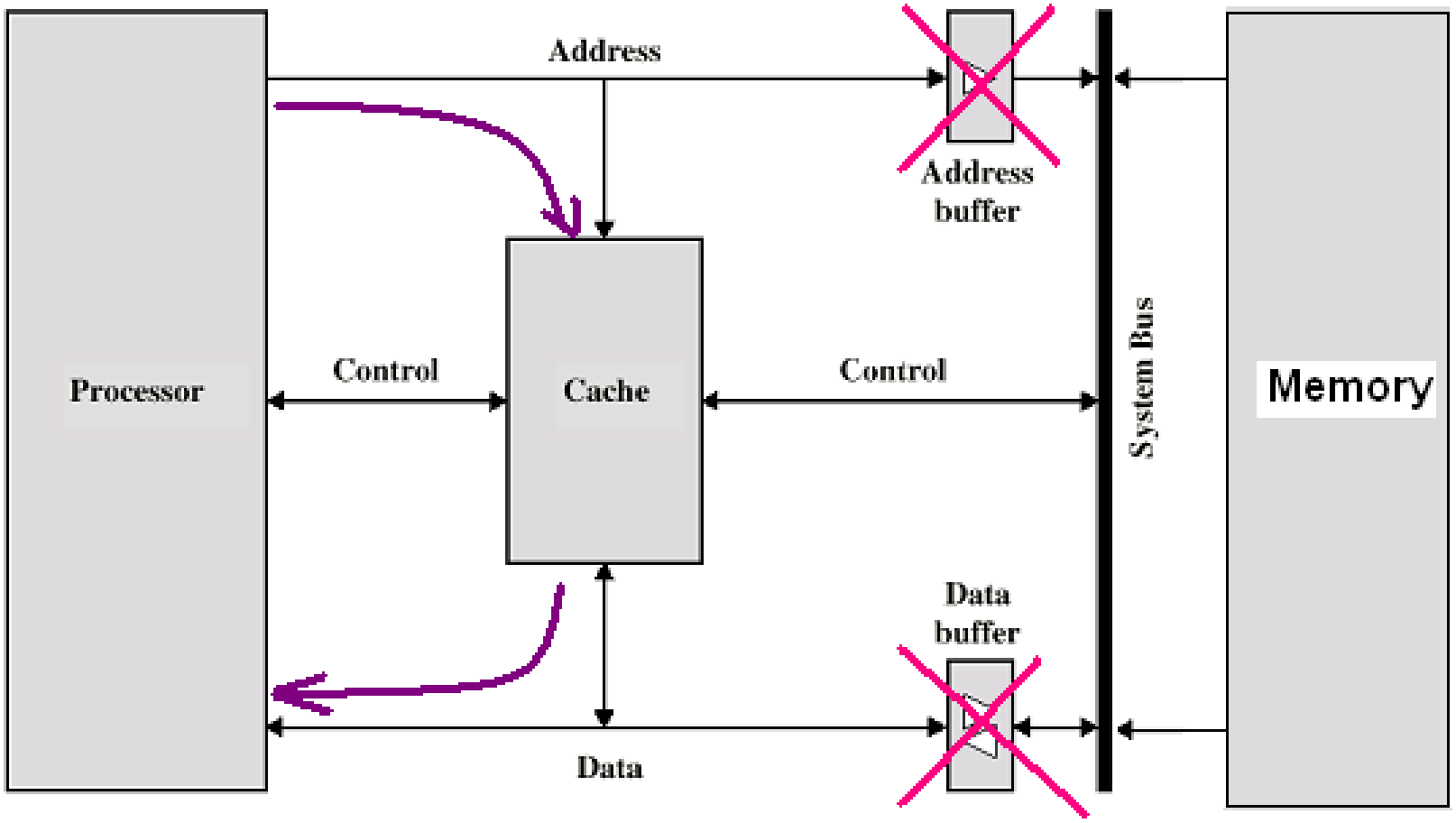
Cache Read Operation



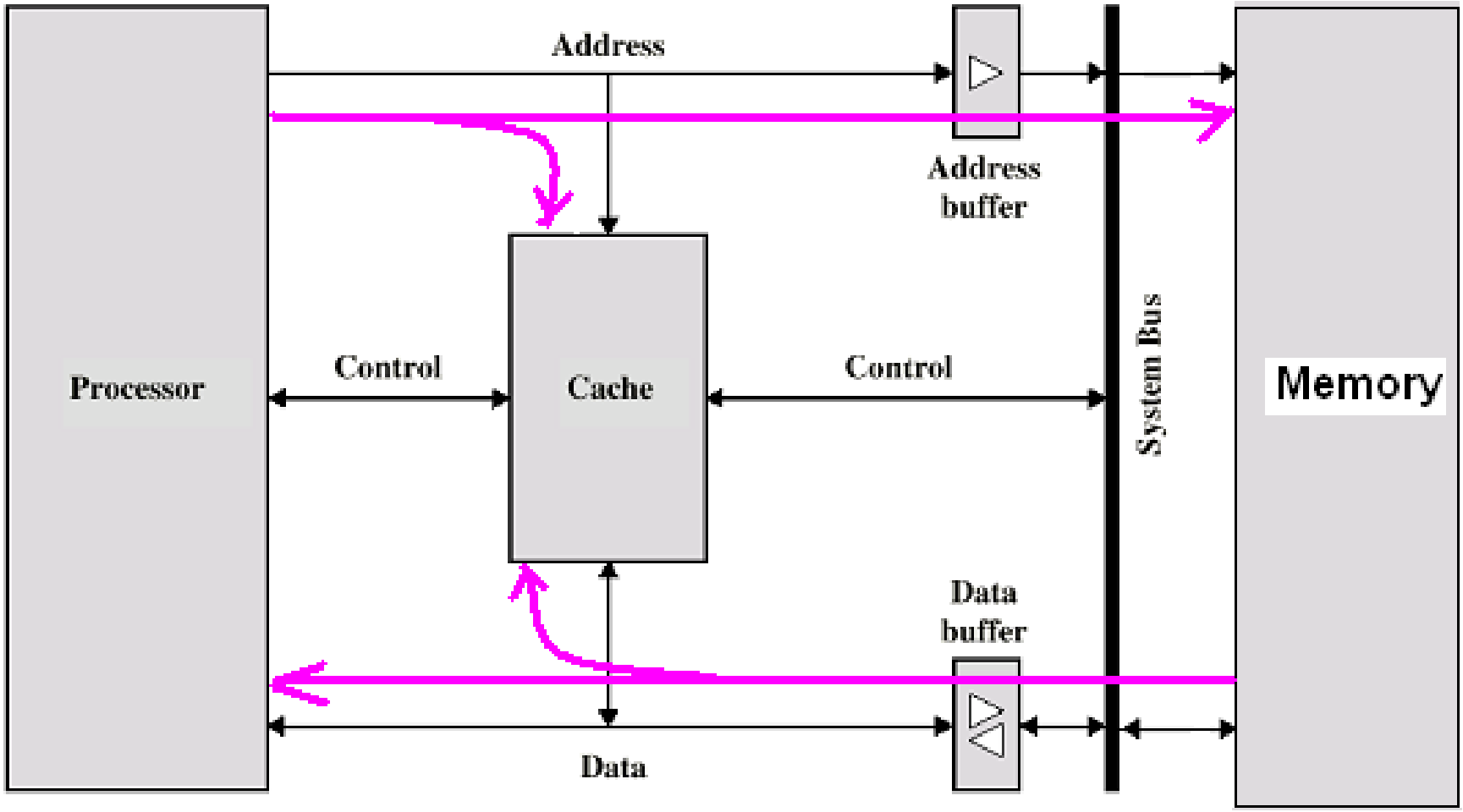
Performance of cache

- Hit ratio : Number of Hits / total references to memory
- Hit
- Miss

Read Hit

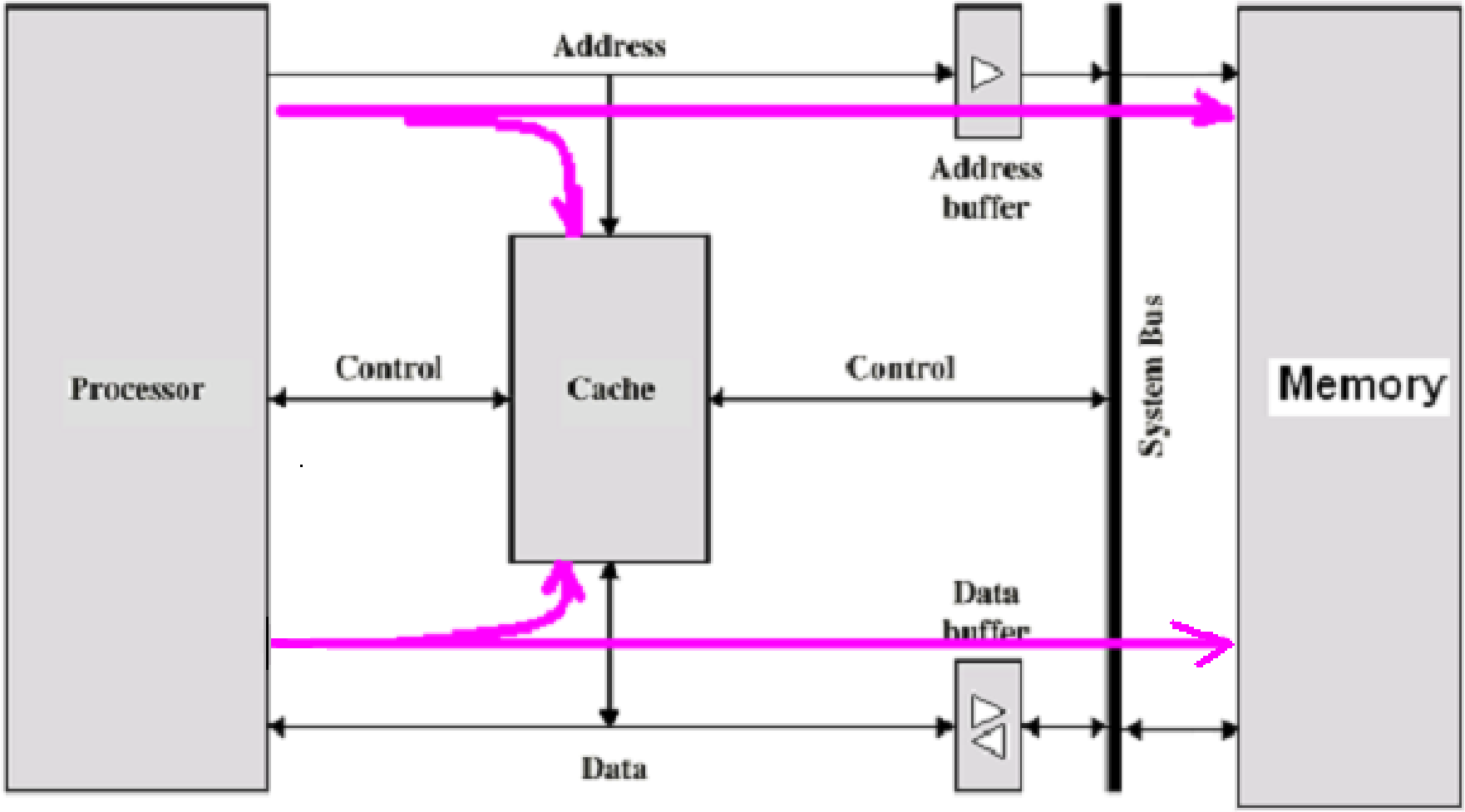

Address
Address buffer
Processor
Control
Cache
Control
System Bus
Memory
Data buffer
Data

Read Miss

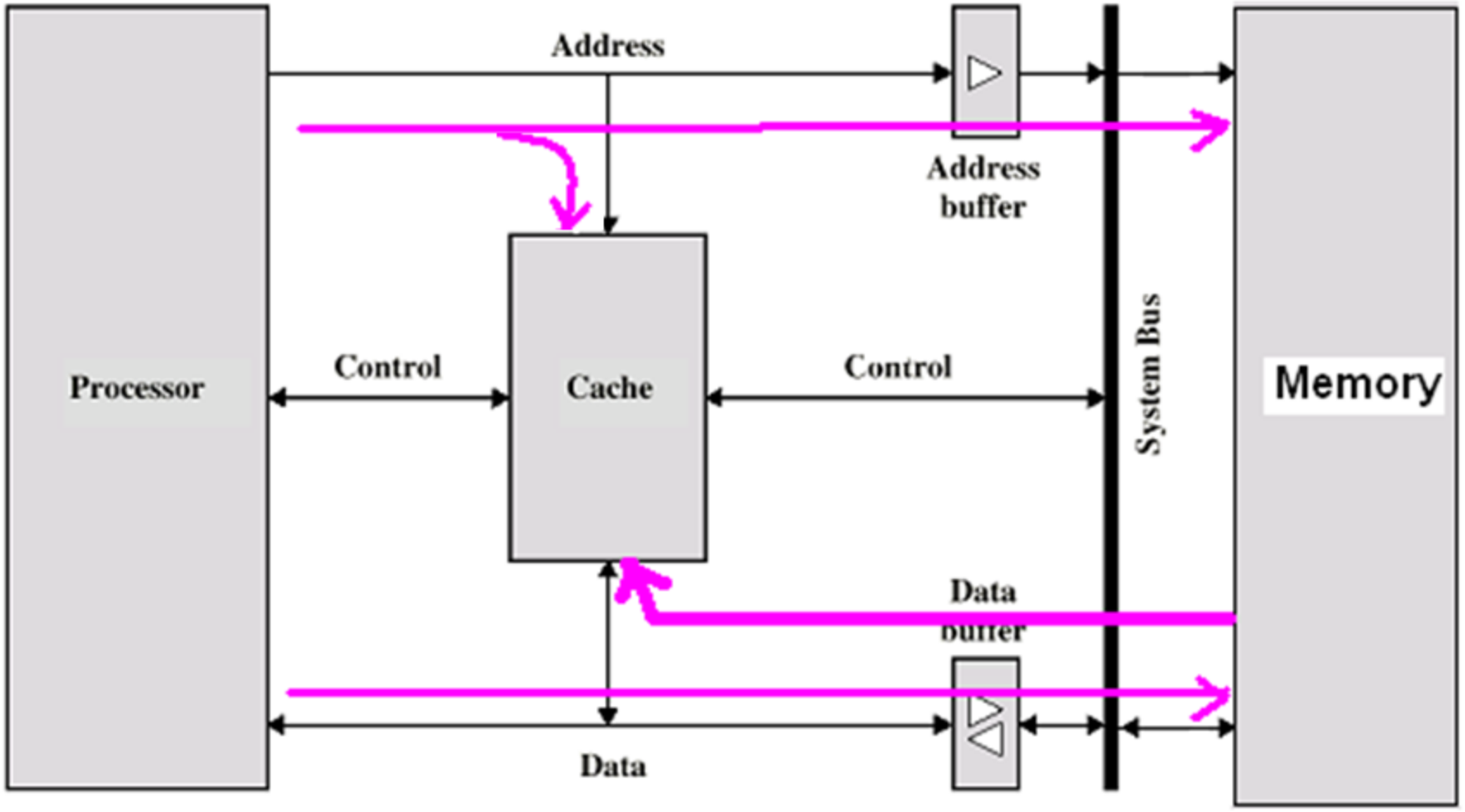


Miss Penalty

Write hit


Address
Address buffer
Processor
Control
Cache
Control
System Bus
Memory
Data buffer
Data

Write miss

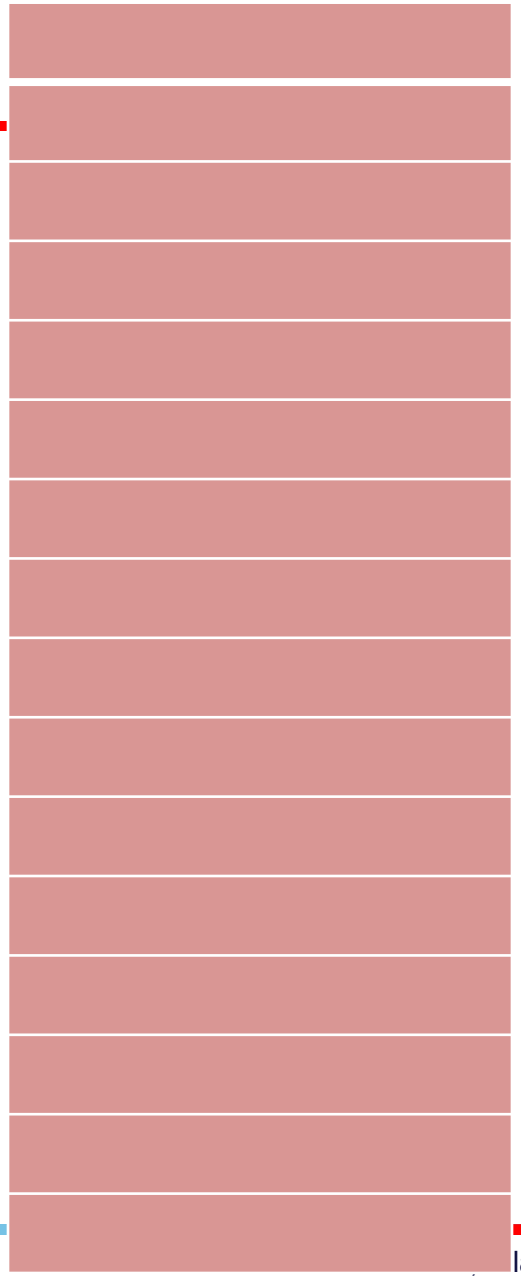

Address
Address buffer
Processor
Control
Cache
Control
System Bus
Memory
Data buffer
Data

Mapping Function

- How memory blocks are mapped to cache lines
- Three types
  - Direct mapping
  - Associative mapping
  - Set Associative mapping

Direct Mapped Cache

- 16 Bytes main memory
  - How many address bits are required?
- Memory block size is 4 bytes
- Cache of 8 Byte
  - How many cache lines?
  - cache contains 2 lines (4 bytes per Line)



Direct Mapped Cache

- Each block of main memory maps to only one cache line
  - **i.e. if a block is in cache, it must be in one specific place**
  - **i = j modulo m**

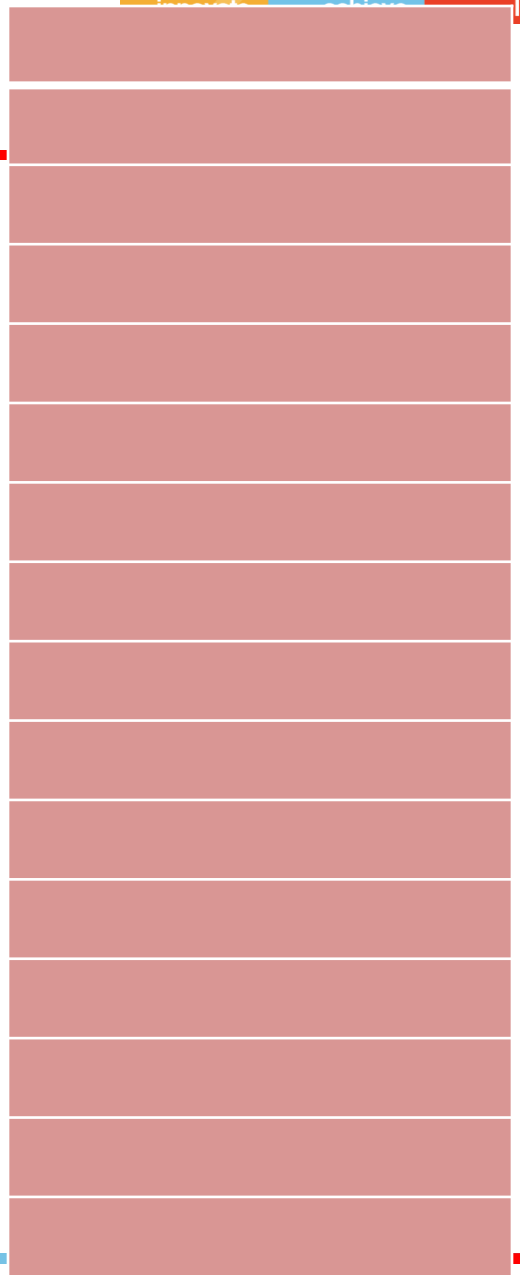
where i = cache line number

j = main memory block no.

m = no.of lines in the cache

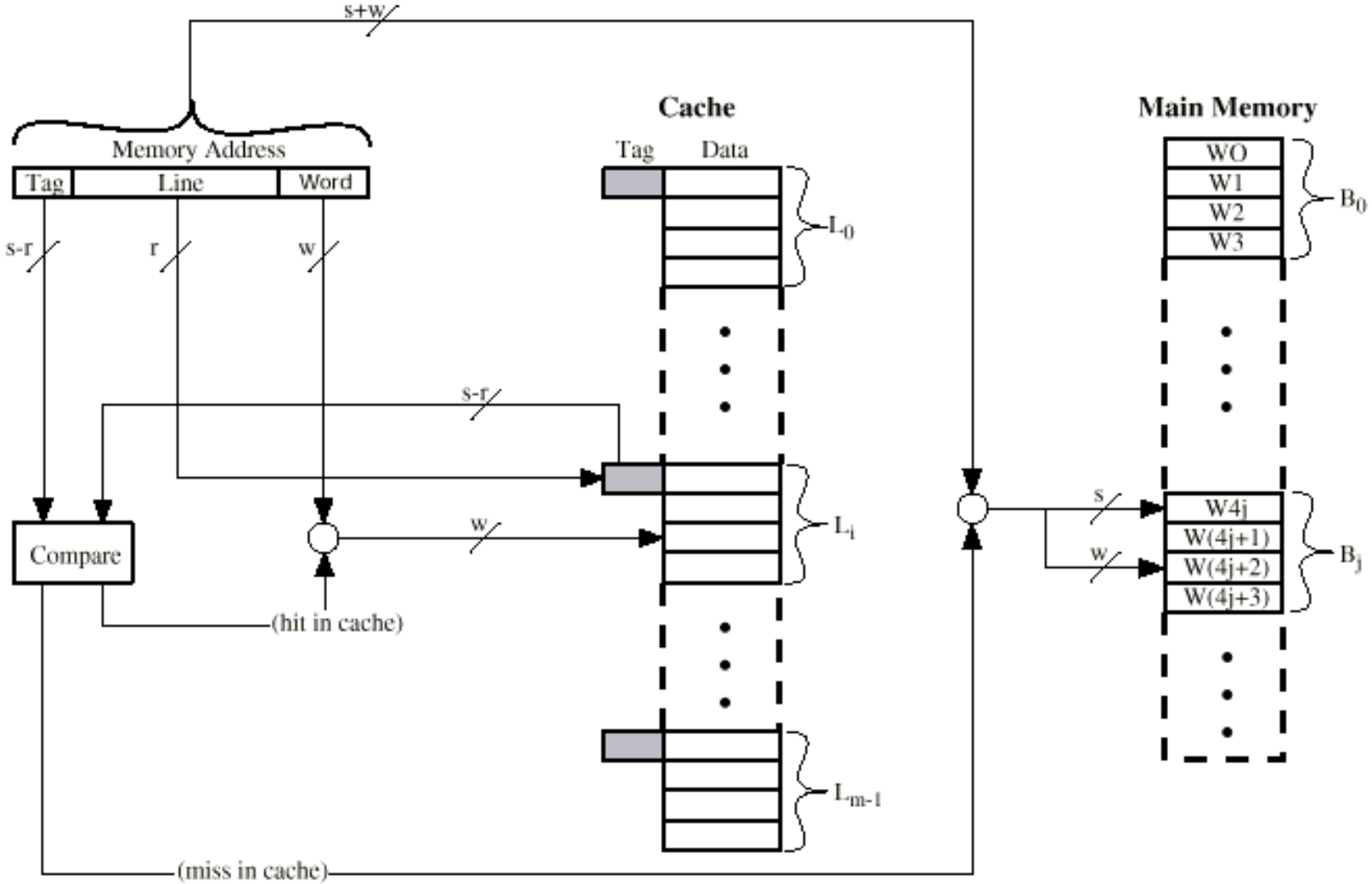
Direct Mapped Cache

i = j modulo m



- Address is split in three parts:
  - Tag
  - Line
  - Word

Direct Mapped Cache Organization



Direct mapped cache- Summary

Address length = (s+w) bits

Number of addressable units = 2^{s+w} words or bytes

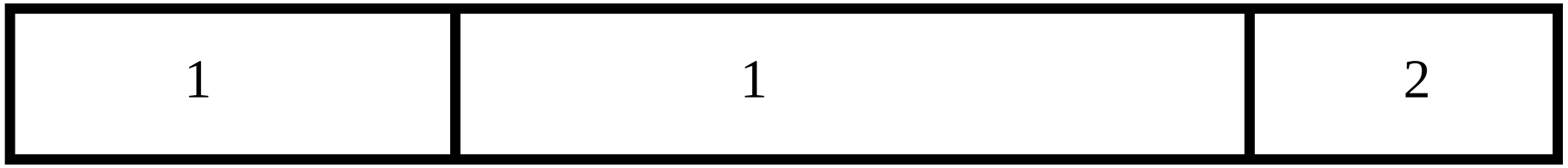
Block size = line size = 2^{w} words or bytes

Number of blocks in main memory = $2^{s+w} / 2^{w} = 2^{s}$

Number of lines in cache = m = 2^{r}

Size of tag =(s-r) bits

Direct mapped cache- Summary



Direct mapped cache-pros & cons

- Simple
- Inexpensive
- Fixed location for given block
  - If a program accesses 2 blocks that map to the same line repeatedly, cache misses are very high

Problem 1 : Direct Mapped Cache

Given :

- Cache of 64KByte, Cache block of 4 bytes
- 16MBytes main memory

Find out

a) Number of bits required to address the main memory
b) Number of blocks in main memory
c) Number of cache lines
d) Number of bits required to identify a word (byte) in a block
e) Number of bits to identify a block
f) Tag, Line, Word

Solution 1

innovate achieve lead

Given :

- **Cache of 64kByte, Cache block of 4 bytes**
- **16MBytes main memory**

Find out

a) Number of bits required to address the main memory

b) Number of blocks in main memory

c) Number of cache lines

Solution 1

Given :

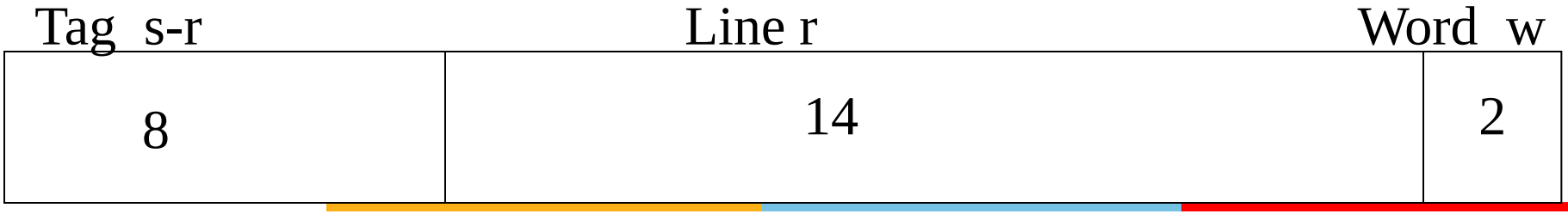
- **Cache of 64kByte, Cache block of 4 bytes**
- **16MBytes main memory**

Find out

d) Number of bits required to identify a word (byte) in a block?

e) Number of bits required to identify a block

f) Tag, Line, Word



Problem 2

Consider a machine with a byte addressable main memory of 2^{16} bytes and block size of 8 bytes. Assume that a direct mapped cache consisting of 32 lines is used with this machine.

a. How is a 16-bit memory address divided into tag, line number, and byte number?
b. Into what line would bytes with each of the following addresses be stored?

0001	0001	0001	1011
1100	0011	0011	0100
1101	0000	0001	1101
1010	1010	1010	1010

c. Suppose the byte with address 0001 1010 0001 1010 is stored in the cache. What are the addresses of the other bytes stored along with it?
d. How many total bytes of memory can be stored in the cache?
e. Why is the tag also stored in the cache?

Solution 2

Consider a machine with a byte addressable main memory of 2^{16} bytes and block size of 8 bytes. Assume that a direct mapped cache consisting of 32 lines is used with this machine.

a. How is a 16-bit memory address divided into tag, line number, and byte number?

b. Into what line would bytes with each of the following addresses be stored?

0001	0001	0001	1011
1100	0011	0011	0100
1101	0000	0001	1101
1010	1010	1010	1010

Solution 2

Consider a machine with a byte addressable main memory of 2^{16} bytes and block size of 8 bytes. Assume that a direct mapped cache consisting of 32 lines is used with this machine.

c. Suppose the byte with address 0001 1010 0001 1010 is stored in the cache. What are the addresses of the other bytes stored along with it?

0001 1010 0001 1**000**
0001 1010 0001 1**001**
0001 1010 0001 1**010** : **given in the problem**
0001 1010 0001 1**011**
0001 1010 0001 1**100**
0001 1010 0001 1**101**
0001 1010 0001 1**110**
0001 1010 0001 1**111**

Solution 2

Consider a machine with a byte addressable main memory of 2^{16} bytes and block size of 8 bytes. Assume that a direct mapped cache consisting of 32 lines is used with this machine.

d. How many total bytes of memory can be stored in the cache?

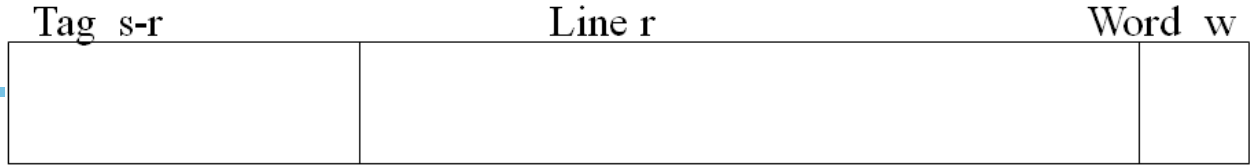
e. Why is the tag also stored in the cache?

Problem 3

Consider a direct-mapped cache with 64 cache lines and a block size of 16 bytes and main memory of 8K (Byte addressable memory) . To what line number does byte address 1200H map?

Hexadecimal	Binary	Decimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

Problem 4



The system uses a L1 cache with direct mapping and 32-bit address format is as follows:

bits 0 - 3 = offset (word)

bits 4 - 14 = index bits (Line)

bits15 - 31 = tag

a) What is the size of cache line?

b) How many Cache lines are there?

c) How much space is required to store the tags in the L1 cache?

d) What is the total Capacity of cache including tag storage?

Problem 5

- 16 Bytes main memory, Memory block size is 4 bytes, Cache of 8 Byte (cache is 2 lines of 4 bytes each)
- Block access sequence :

0 2 0 2 2 0 0 2 0 0 0 2 1

- Find out hit ratio.

Problem 6

Suppose a 1024-byte cache has an access time of 0.1 microseconds and the main memory stores 1 Mbytes with an access time of 1 microsecond. A referenced memory block that is not in cache must be loaded into cache .

Answer the following questions:

a) What is the number of bits needed to address the main memory?

b) If the cache hit ratio is 95%, what is the average access time for a memory reference?

Solution 6

a) 20 bits

b) If the cache hit ratio is 95%, what is the average access time for a memory reference?

Avg access time = hit ratio * cache access +

(1- hit ratio) * (cache access + memory access)