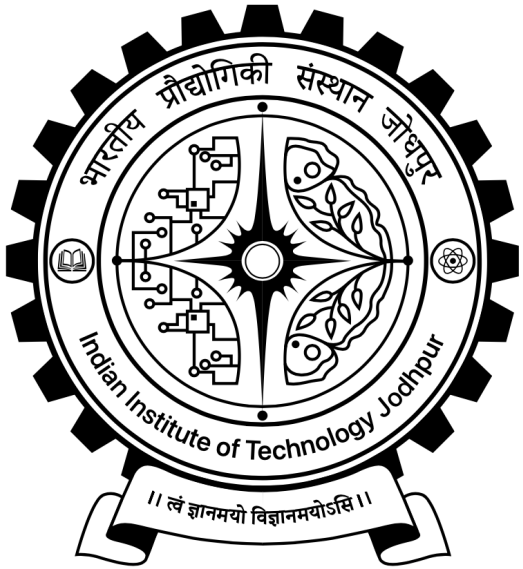


Computer Architecture
CSL3020

Deepak Mishra

http://home.iitj.ac.in/~dmishra/
Department of Computer Science and Engineering
Indian Institute of Technology Jodhpur

Overview of Memory System

Story so far –

Memory is a one large chunk of bytes.

Overview of Memory System

Story so far –

Memory is a one large chunk of bytes.

Memory access (read/write) can be completed within one clock cycle.

Overview of Memory System

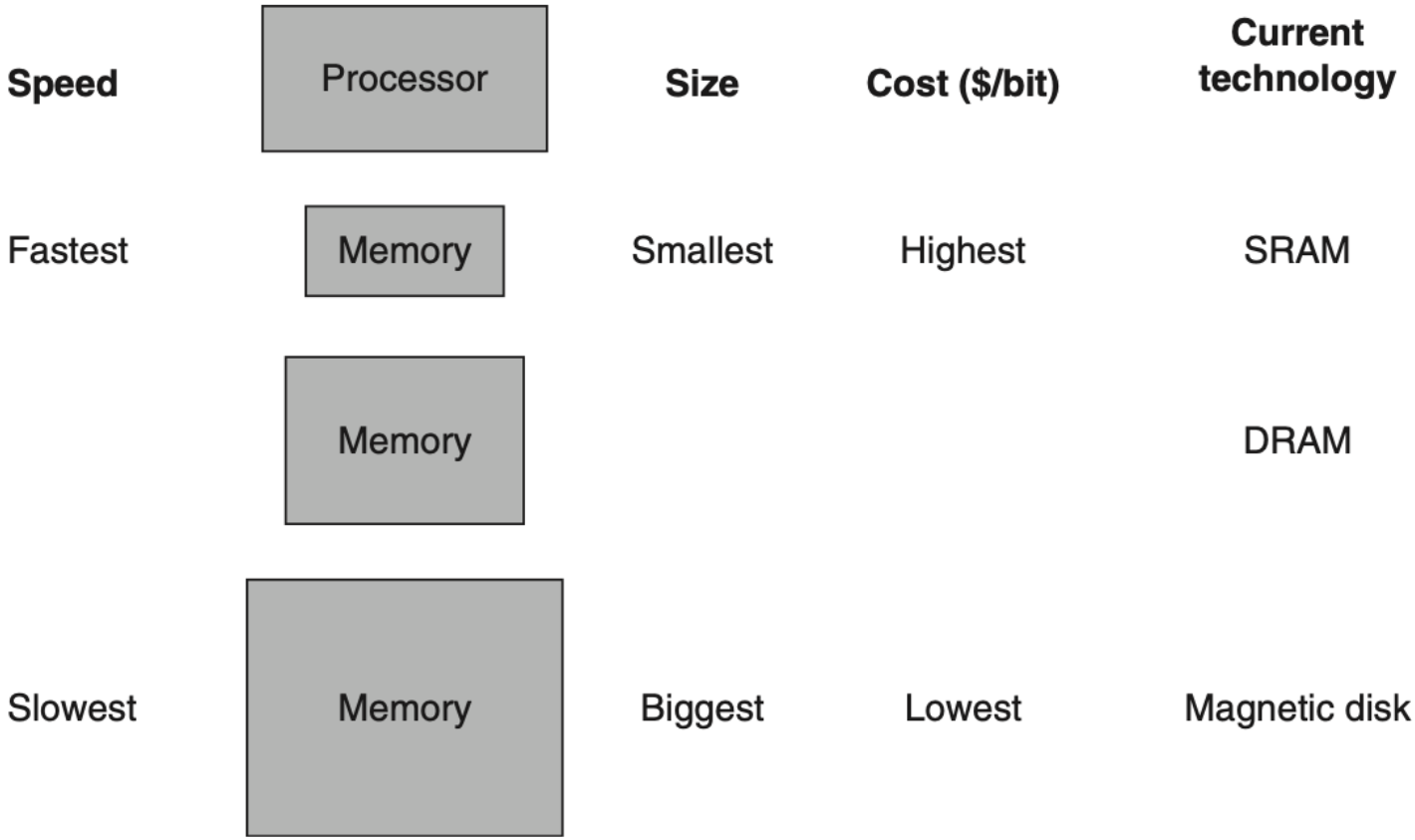
Story so far –

Memory is a one large chunk of bytes.

Memory access (read/write) can be completed within one clock cycle.

All our programs take less than 4 GB of space.

Overview of Memory System



Area, power, latency trade-off

Overview of Memory System

A memory hierarchy can consist of multiple levels, but data is copied between only two adjacent levels at a time.

Overview of Memory System

A memory hierarchy can consist of multiple levels, but data is copied between only two adjacent levels at a time.

Temporal locality: The principle stating that if a data location is referenced then it will tend to be referenced again soon.

Overview of Memory System

A memory hierarchy can consist of multiple levels, but data is copied between only two adjacent levels at a time.

Temporal locality: The principle stating that if a data location is referenced then it will tend to be referenced again soon.

Spatial locality: The principle stating that if a data location is referenced, data locations with nearby addresses will tend to be referenced soon.

Overview of Memory System

How to take the advantage of temporal locality?

Overview of Memory System

How to take the advantage of temporal locality?

Use memory hierarchy

Overview of Memory System

How to take the advantage of temporal locality?

Use memory hierarchy

When the data requested by the processor appears in upper level (cache), it is called a hit otherwise a miss.

Overview of Memory System

How to take the advantage of temporal locality?

Use memory hierarchy

When the data requested by the processor appears in upper level (cache), it is called a hit otherwise a miss.

- Hit rate: The fraction of memory accesses found in the upper level.
- Miss rate = 1 - hit rate
- Miss penalty: It is the time taken to transfer data from lower level to upper level, plus the hit time.

How to take the advantage of spatial locality?

Overview of Memory System

How to take the advantage of spatial locality?

Group memory addresses into blocks and transfer blocks between the levels in memory hierarchy instead of transferring single byte.

- Block: The minimum unit of information that can be either present or not present in the two-level hierarchy is called a block or a line.

Basics of Cache

Using main memory (RAM) for the IM and DM units of the processor we design is inefficient.

Basics of Cache

Using main memory (RAM) for the IM and DM units of the processor we design is inefficient.

Cache represents the level of the memory hierarchy between the processor and main memory.

Basics of Cache

Using main memory (RAM) for the IM and DM units of the processor we design is inefficient.

Cache represents the level of the memory hierarchy between the processor and main memory.

How do we know if a data item is in the cache? If it is, how do we find it?

Basics of Cache

Using main memory (RAM) for the IM and DM units of the processor we design is inefficient.

Cache represents the level of the memory hierarchy between the processor and main memory.

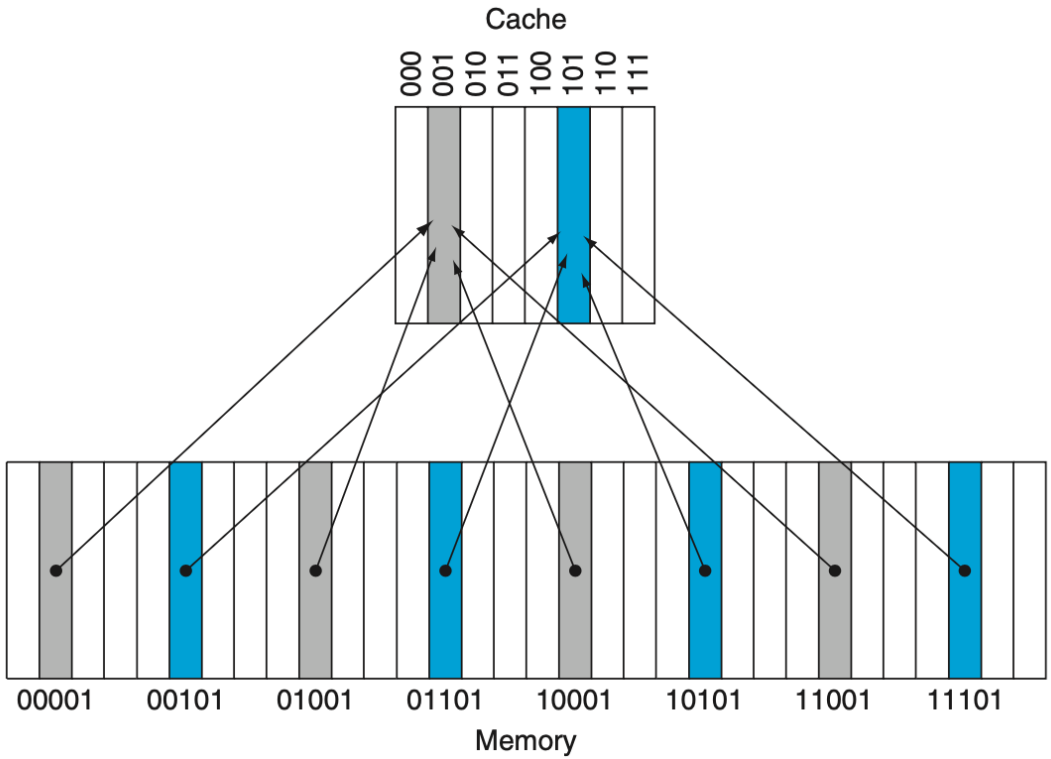
How do we know if a data item is in the cache? If it is, how do we find it?

We need a mapping scheme to answer these questions.

Basics of Cache

Direct mapping: Each (main) memory location is mapped to exactly one location in the cache.

Cache location = (Block address) modulo (Number of blocks in the cache)



Basics of Cache

As multiple addresses are mapped to a single cache location, the referenced address is divided into two fields:

- *Tag:* contains the address information required to identify associated block
- *Cache index:* used to select the block

The index of a cache block, together with the tag, uniquely specifies the corresponding memory address.

Basics of Cache

As multiple addresses are mapped to a single cache location, the referenced address is divided into two fields:

- *Tag:* contains the address information required to identify associated block
- *Cache index:* used to select the block

The index of a cache block, together with the tag, uniquely specifies the corresponding memory address.

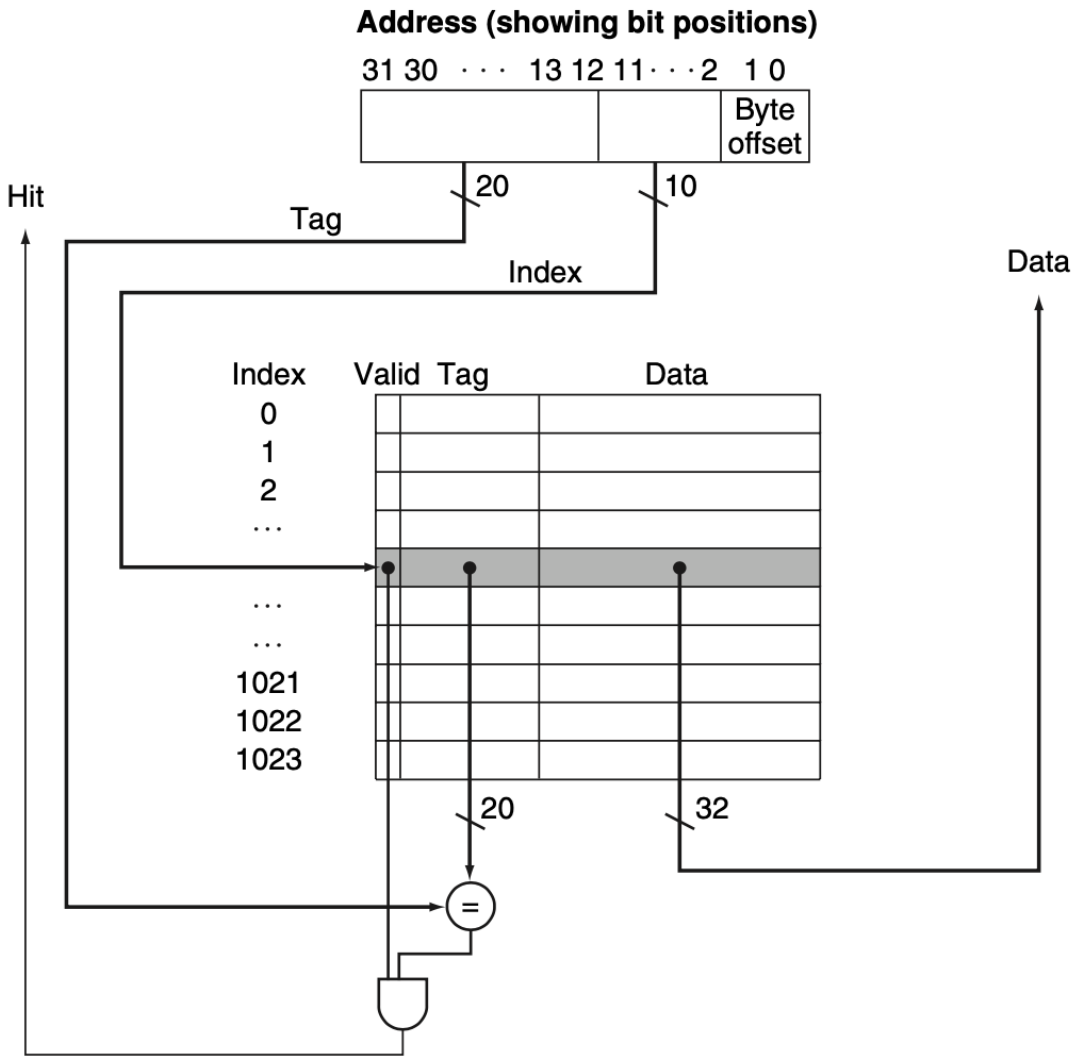
Valid bit: A field in the tables of a memory hierarchy that indicates that the associated block in the hierarchy contains valid data.

Example:

Binary address of reference	Hit or miss in cache	Assigned cache block (where found or placed)
10110_{two}	miss (5.6b)	$(10110_{two} \bmod 8) = 110_{two}$
11010_{two}	miss (5.6c)	$(11010_{two} \bmod 8) = 010_{two}$
10110_{two}	hit	$(10110_{two} \bmod 8) = 110_{two}$
11010_{two}	hit	$(11010_{two} \bmod 8) = 010_{two}$
10000_{two}	miss (5.6d)	$(10000_{two} \bmod 8) = 000_{two}$
00011_{two}	miss (5.6e)	$(00011_{two} \bmod 8) = 011_{two}$
10000_{two}	hit	$(10000_{two} \bmod 8) = 000_{two}$
10010_{two}	miss (5.6f)	$(10010_{two} \bmod 8) = 010_{two}$
10000_{two}	hit	$(10000_{two} \bmod 8) = 000_{two}$

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

Cache Misses



Assuming that the block size is one word and address is 32 bit long

Cache Misses

Example: How many total bits are required for a direct-mapped cache with 16 KB of data and 4-word blocks, assuming a 32-bit address?

Cache Misses

Example: How many total bits are required for a direct-mapped cache with 16 KB of data and 4-word blocks, assuming a 32-bit address?

Ans: 147 Kbits

Cache Misses

Steps to be taken on an instruction cache miss:

Cache Misses

Steps to be taken on an instruction cache miss:

1. Send the original PC value (current PC – 4) to the memory.

Cache Misses

Steps to be taken on an instruction cache miss:

1. Send the original PC value (current PC – 4) to the memory.
2. Instruct main memory to perform a read and wait (stall) for the memory to complete its access.

Cache Misses

Steps to be taken on an instruction cache miss:

1. Send the original PC value (current PC – 4) to the memory.
2. Instruct main memory to perform a read and wait (stall) for the memory to complete its access.
3. Write the cache entry

Cache Misses

Steps to be taken on an instruction cache miss:

1. Send the original PC value (current PC – 4) to the memory.
2. Instruct main memory to perform a read and wait (stall) for the memory to complete its access.
3. Write the cache entry
   - putting the data from memory in the data portion of the entry,
   - writing the upper bits of the address (from the ALU) into the tag field,
   - turn the valid bit on.

Cache Misses

Steps to be taken on an instruction cache miss:

1. Send the original PC value (current PC – 4) to the memory.
2. Instruct main memory to perform a read and wait (stall) for the memory to complete its access.
3. Write the cache entry
   - putting the data from memory in the data portion of the entry,
   - writing the upper bits of the address (from the ALU) into the tag field,
   - turn the valid bit on.
4. Restart the instruction execution at the first step.

Cache Misses

The control of the cache on a data access is essentially identical: on a miss, we simply stall the processor until the memory responds with the data.

Cache Misses

Handling the write miss:

- *Write-through:* A scheme in which writes always update both the cache and the next lower level of the memory hierarchy.

Cache Misses

Handling the write miss:

- *Write-through:* A scheme in which writes always update both the cache and the next lower level of the memory hierarchy.
- *Write buffer:* A queue that holds data while the data is waiting to be written to memory.

Cache Misses

Handling the write miss:

- *Write-through:* A scheme in which writes always update both the cache and the next lower level of the memory hierarchy.
- *Write buffer:* A queue that holds data while the data is waiting to be written to memory.
- *Write-back:* Updating values only to the block in the cache, then writing the modified block to the lower level of the hierarchy when the block is replaced.

Cache Performance Analysis

How to evaluate cache performance

Cache Performance Analysis

How to evaluate cache performance

$$\text{CPU time} = (\text{CPU execution clock cycles} + \text{Memory-stall clock cycles}) \times \text{Clock cycle time}$$

Cache Performance Analysis

How to evaluate cache performance

$$\text{CPU time} = (\text{CPU execution clock cycles} + \text{Memory-stall clock cycles}) \times \text{Clock cycle time}$$

$$\text{Memory-stall clock cycles} = \text{Memory accesses per program} \times \text{Miss rate} \times \text{Miss penalty}$$

Cache Performance Analysis

How to evaluate cache performance

$$\text{CPU time} = (\text{CPU execution clock cycles} + \text{Memory-stall clock cycles}) \times \text{Clock cycle time}$$

$$\text{Memory-stall clock cycles} = \text{Memory accesses per program} \times \text{Miss rate} \times \text{Miss penalty}$$

$$\text{Memory-stall clock cycles} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Misses}}{\text{Instructions}} \times \text{Miss penalty}$$

Cache Performance Analysis

Example: Assume the miss rate of an instruction cache is 2% and the miss rate of the data cache is 4%. If a processor has a CPI of 2 without any memory stalls and the miss penalty is 100 cycles for all misses, determine how much faster a processor would run with a perfect cache that never missed. Assume the frequency of all loads and stores is 36%.

Cache Performance Analysis

Example: Assume the miss rate of an instruction cache is 2% and the miss rate of the data cache is 4%. If a processor has a CPI of 2 without any memory stalls and the miss penalty is 100 cycles for all misses, determine how much faster a processor would run with a perfect cache that never missed. Assume the frequency of all loads and stores is 36%.

Ans: 2.72 times

Reducing Cache Misses

Flexible Mapping of Blocks

- *Fully Associative:* A cache structure in which a block can be placed in any location in the cache.

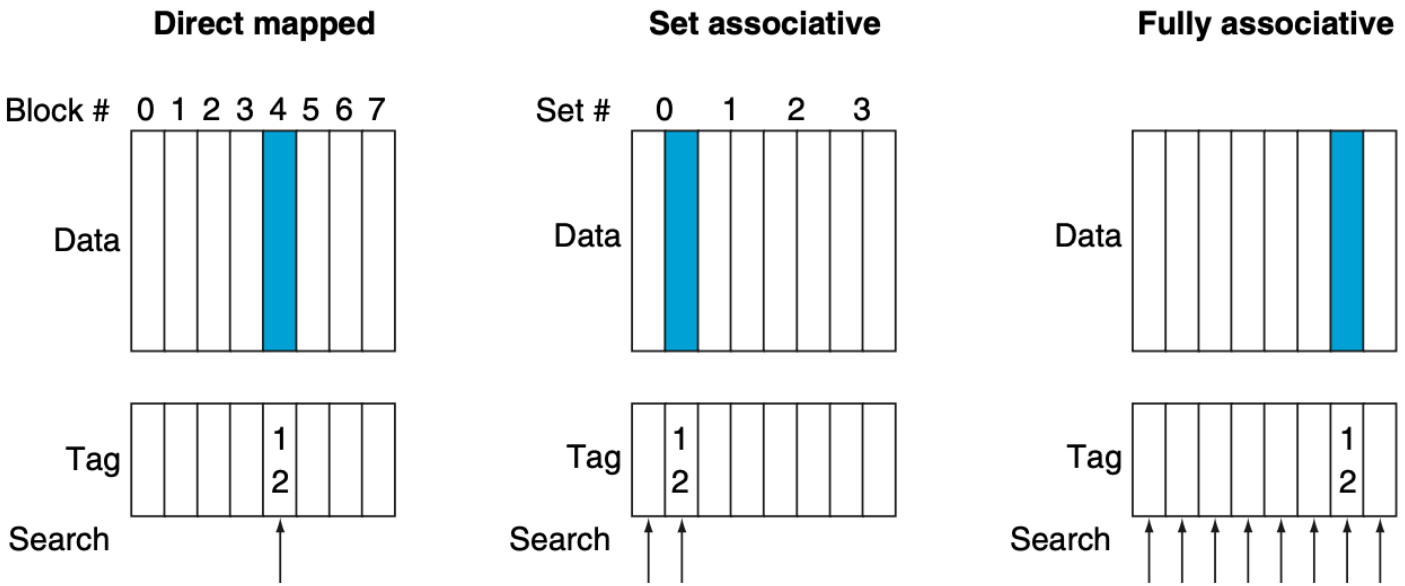
Reducing Cache Misses

Flexible Mapping of Blocks

- *Fully Associative:* A cache structure in which a block can be placed in any location in the cache.
- *Set-associative:* A cache that has a fixed number of locations (at least two) where each block can be placed. A set-associative cache with n locations for a block is called an n-way set-associative cache.

Flexible Mapping of Blocks

- *Fully Associative:* A cache structure in which a block can be placed in any location in the cache.
- *Set-associative:* A cache that has a fixed number of locations (at least two) where each block can be placed. A set-associative cache with n locations for a block is called an n-way set-associative cache.



Reducing Cache Misses

Example: Assume there is a small cache consisting of eight one-word blocks. Given the following sequence of block addresses: 0, 8, 0, 5, 12, 9, 1, 5, and 8, find the number of misses for 2-way, 4-way, and 8-way set-associative mapping with LRU replacement scheme.

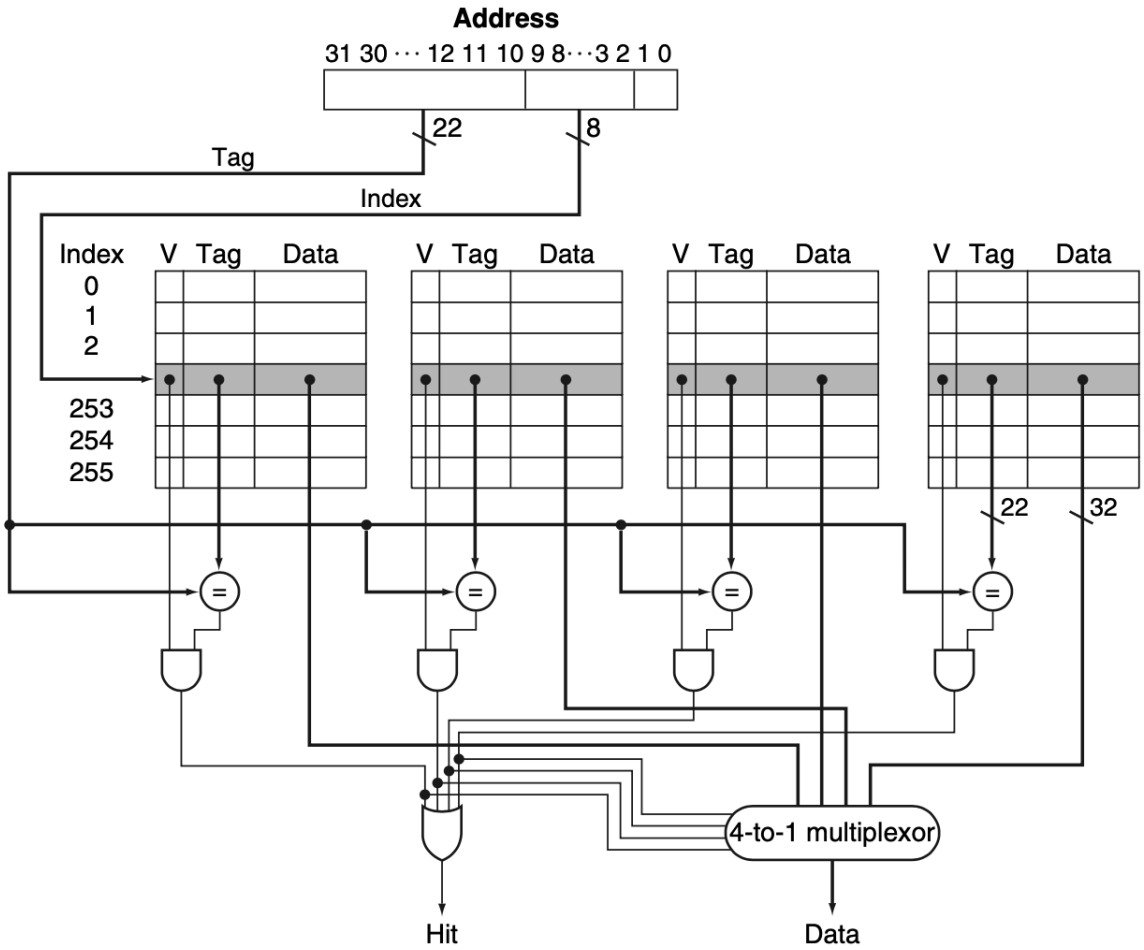
Reducing Cache Misses

Example: Assume there is a small cache consisting of eight one-word blocks. Given the following sequence of block addresses: 0, 8, 0, 5, 12, 9, 1, 5, and 8, find the number of misses for 2-way, 4-way, and 8-way set-associative mapping with LRU replacement scheme.

Ans: 8, 6, 6

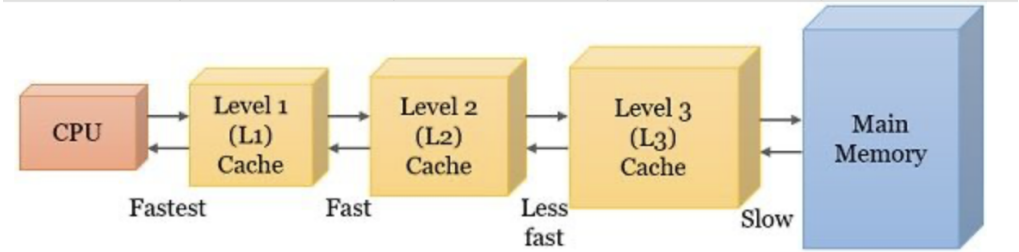
Reducing Cache Misses

The tag of every cache block within the appropriate set is checked to see if it matches the block address from the processor.

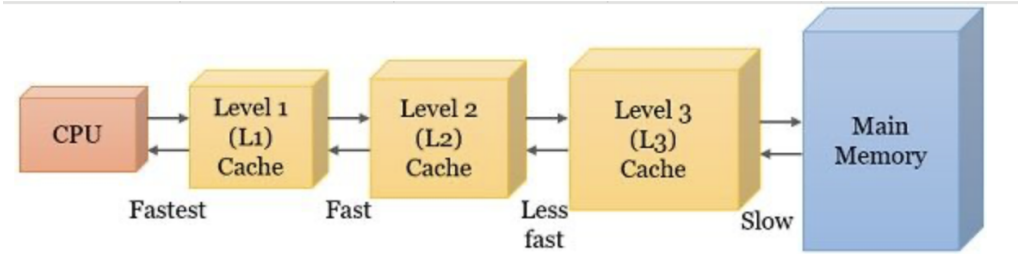


Implementation of a 4-way set-associative cache requires four comparators and a 4-to-1 multiplexor.

Reducing Cache Misses - Multilevel Cache

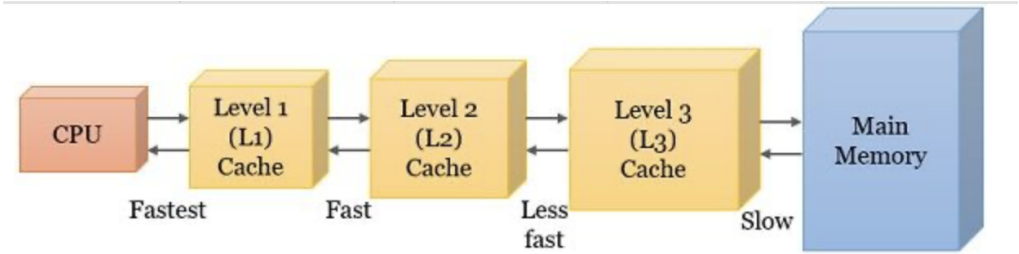


Reducing Cache Misses - Multilevel Cache



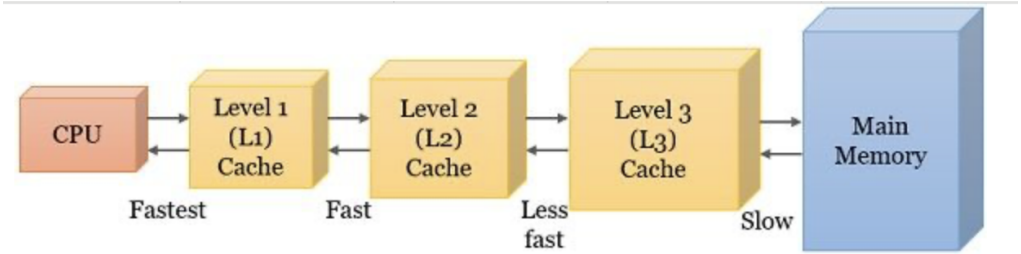
- Multilevel cache extends the advantages of memory hierarchy.

Reducing Cache Misses - Multilevel Cache



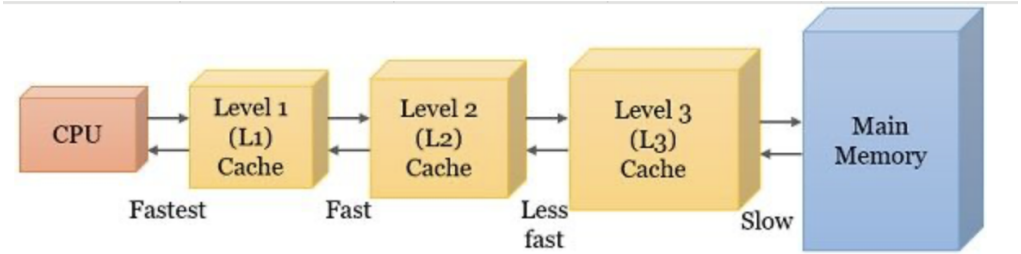
- Multilevel cache extends the advantages of memory hierarchy.
- If the lower-level cache contains the desired data, the miss penalty for the first-level cache will be essentially the access time of the second(lower)-level cache.

Reducing Cache Misses - Multilevel Cache



- Multilevel cache extends the advantages of memory hierarchy.
- If the lower-level cache contains the desired data, the miss penalty for the first-level cache will be essentially the access time of the second(lower)-level cache.
  - This will be much smaller than the access time of main memory.

Reducing Cache Misses - Multilevel Cache



- Multilevel cache extends the advantages of memory hierarchy.
- If the lower-level cache contains the desired data, the miss penalty for the first-level cache will be essentially the access time of the second(lower)-level cache.
  - This will be much smaller than the access time of main memory.
- If none of the caches contains the data, a main memory access is required, and a larger miss penalty is incurred.

Example: Suppose we have a processor with a base CPI of 1.0 and a clock rate of 4 GHz. Assume a main memory access time of 100 ns, including all the miss handling. Suppose the miss rate per instruction at the primary cache is 2%. How much faster will the processor be if we add a secondary cache that has a 5 ns access time for either a hit or a miss and is large enough to reduce the miss rate to main memory to 0.5%?

Example: Suppose we have a processor with a base CPI of 1.0 and a clock rate of 4 GHz. Assume a main memory access time of 100 ns, including all the miss handling. Suppose the miss rate per instruction at the primary cache is 2%. How much faster will the processor be if we add a secondary cache that has a 5 ns access time for either a hit or a miss and is large enough to reduce the miss rate to main memory to 0.5%?

Ans: 2.6

Virtual Memory

- So far we have considered individual programs where we have assumed that

Virtual Memory

- So far we have considered individual programs where we have assumed that
  - a program perceives the entire memory system to be its own

Virtual Memory

- So far we have considered individual programs where we have assumed that
  - a program perceives the entire memory system to be its own
  - every program owns 4 GB of memory space

Virtual Memory

- So far we have considered individual programs where we have assumed that
  - a program perceives the entire memory system to be its own
  - every program owns 4 GB of memory space
- In reality multiple programs need to run simultaneously

Virtual Memory

- So far we have considered individual programs where we have assumed that
  - a program perceives the entire memory system to be its own
  - every program owns 4 GB of memory space
- In reality multiple programs need to run simultaneously
  - CPU switches from one program to another

Virtual Memory

- So far we have considered individual programs where we have assumed that
  - a program perceives the entire memory system to be its own
  - every program owns 4 GB of memory space
- In reality multiple programs need to run simultaneously
  - CPU switches from one program to another
  - we need to ensure that the programs do not corrupt the data of each other

Virtual Memory

- So far we have considered individual programs where we have assumed that
  - a program perceives the entire memory system to be its own
  - every program owns 4 GB of memory space
- In reality multiple programs need to run simultaneously
  - CPU switches from one program to another
  - we need to ensure that the programs do not corrupt the data of each other
  - we also need to design memory systems that have less than 4 GB of main memory

Virtual Memory

- So far we have considered individual programs where we have assumed that
  - a program perceives the entire memory system to be its own
  - every program owns 4 GB of memory space
- In reality multiple programs need to run simultaneously
  - CPU switches from one program to another
  - we need to ensure that the programs do not corrupt the data of each other
  - we also need to design memory systems that have less than 4 GB of main memory
- In order to achieve the aforementioned we use virtual memory

Virtual Memory

- Virtual memory is the memory space which is assumed by a program

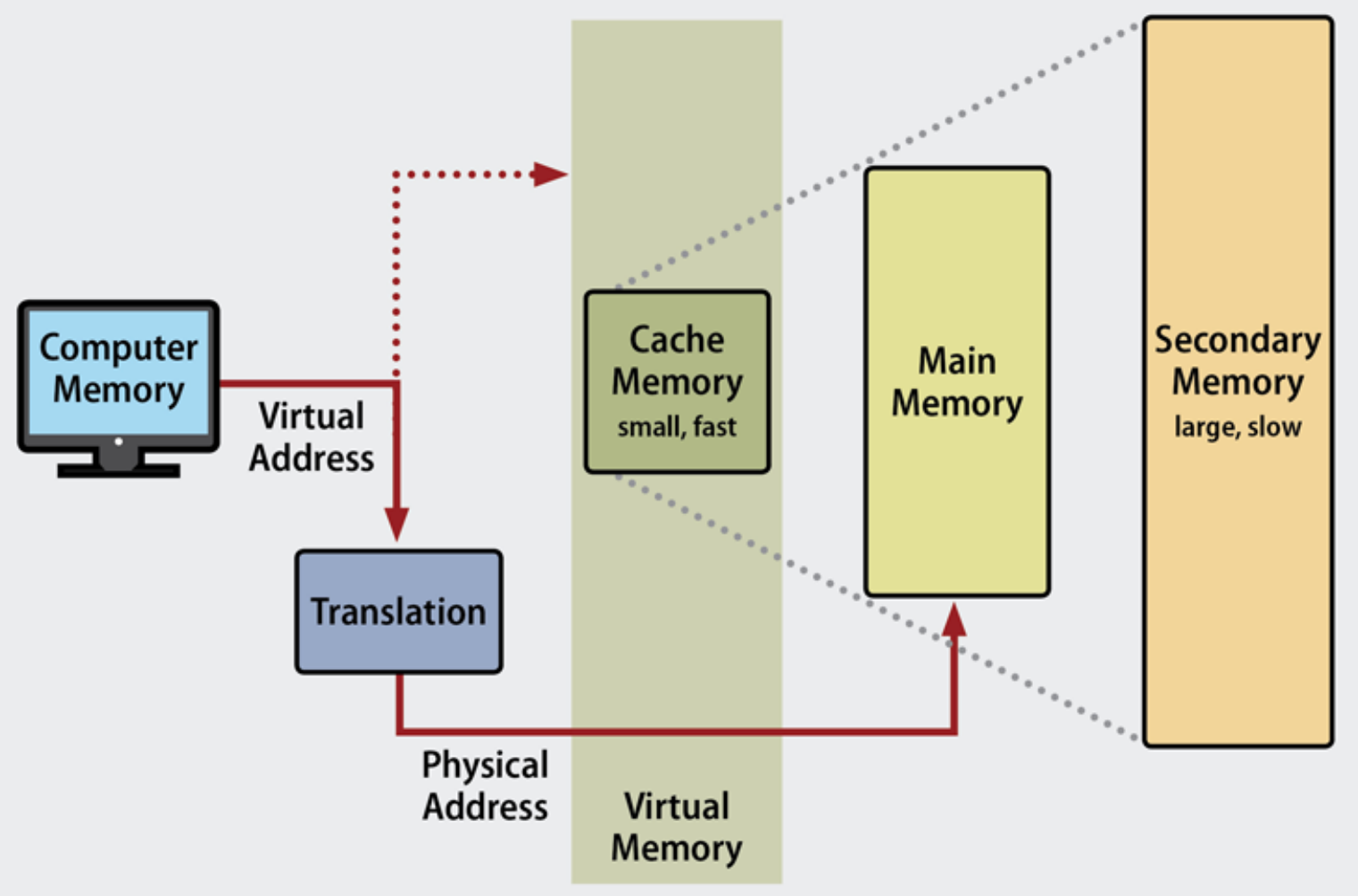
Virtual Memory

- Virtual memory is the memory space which is assumed by a program
- It is a contiguous space which is mapped to the noncontiguous physical memory space (main memory) using address translation system.

Virtual Memory

- Virtual memory is the memory space which is assumed by a program
- It is a contiguous space which is mapped to the noncontiguous physical memory space (main memory) using address translation system.
- Address translation system converts a virtual address to a physical address.

Virtual Memory



source: ded9.com

Virtual Memory - Pages

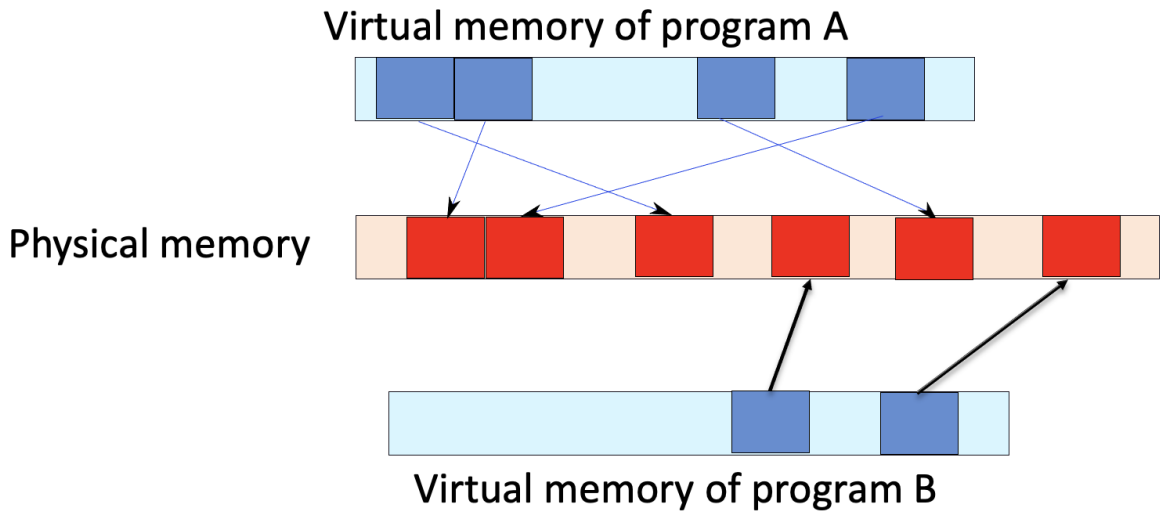
- Divide the virtual address space into atomic chunks of 4 KB called as (virtual) **page**

Virtual Memory - Pages

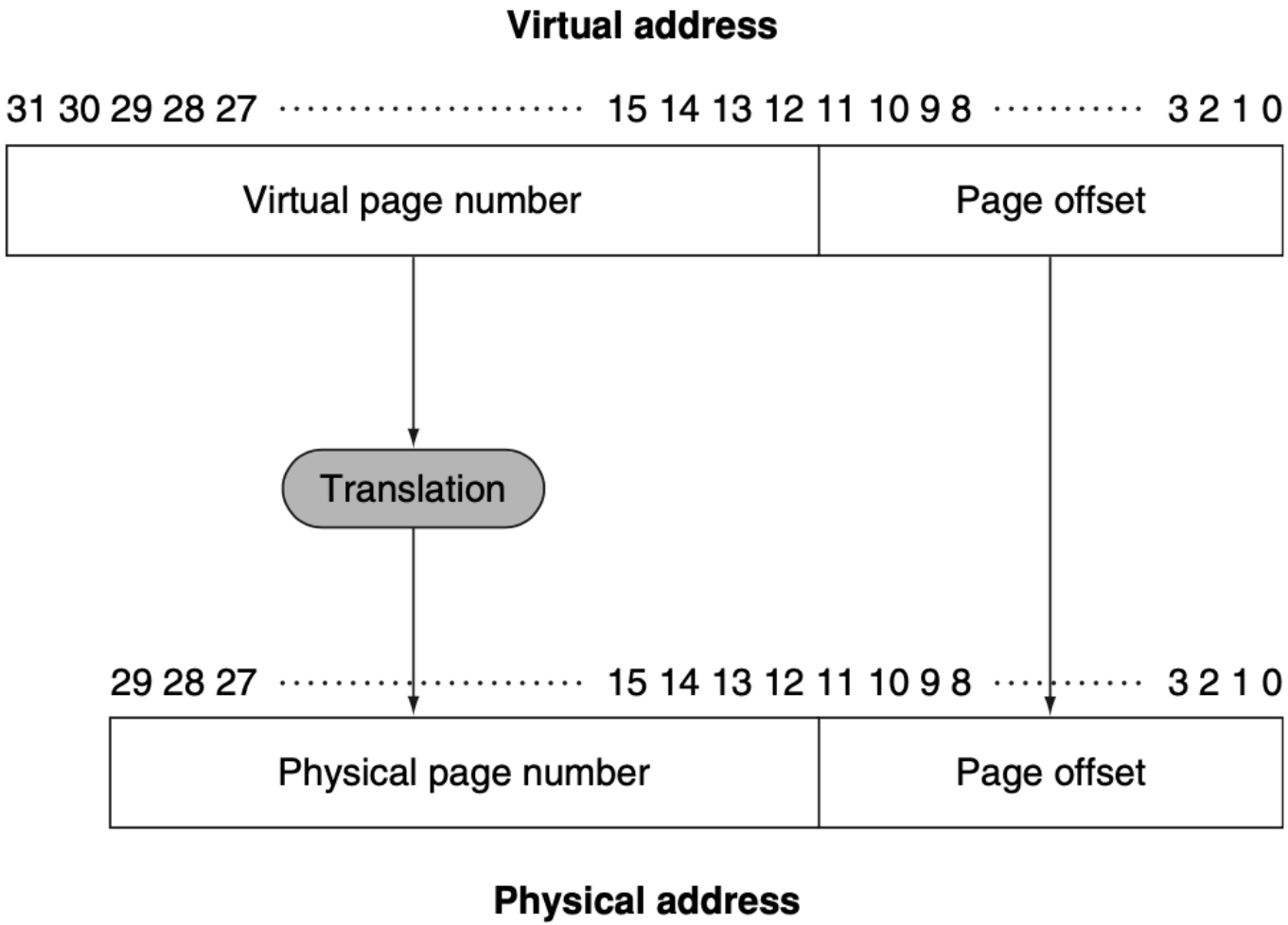
- Divide the virtual address space into atomic chunks of 4 KB called as (virtual) **page**
- Divide the physical address space into chunks of 4 KB called as physical page or frame

Virtual Memory - Pages

- Divide the virtual address space into atomic chunks of 4 KB called as (virtual) **page**
- Divide the physical address space into chunks of 4 KB called as physical page or frame



Virtual Memory - Address Translation



Virtual Memory - Page Table

Page table is the table which contains virtual to physical address mapping.

Virtual Memory - Page Table

Page table is the table which contains virtual to physical address mapping.

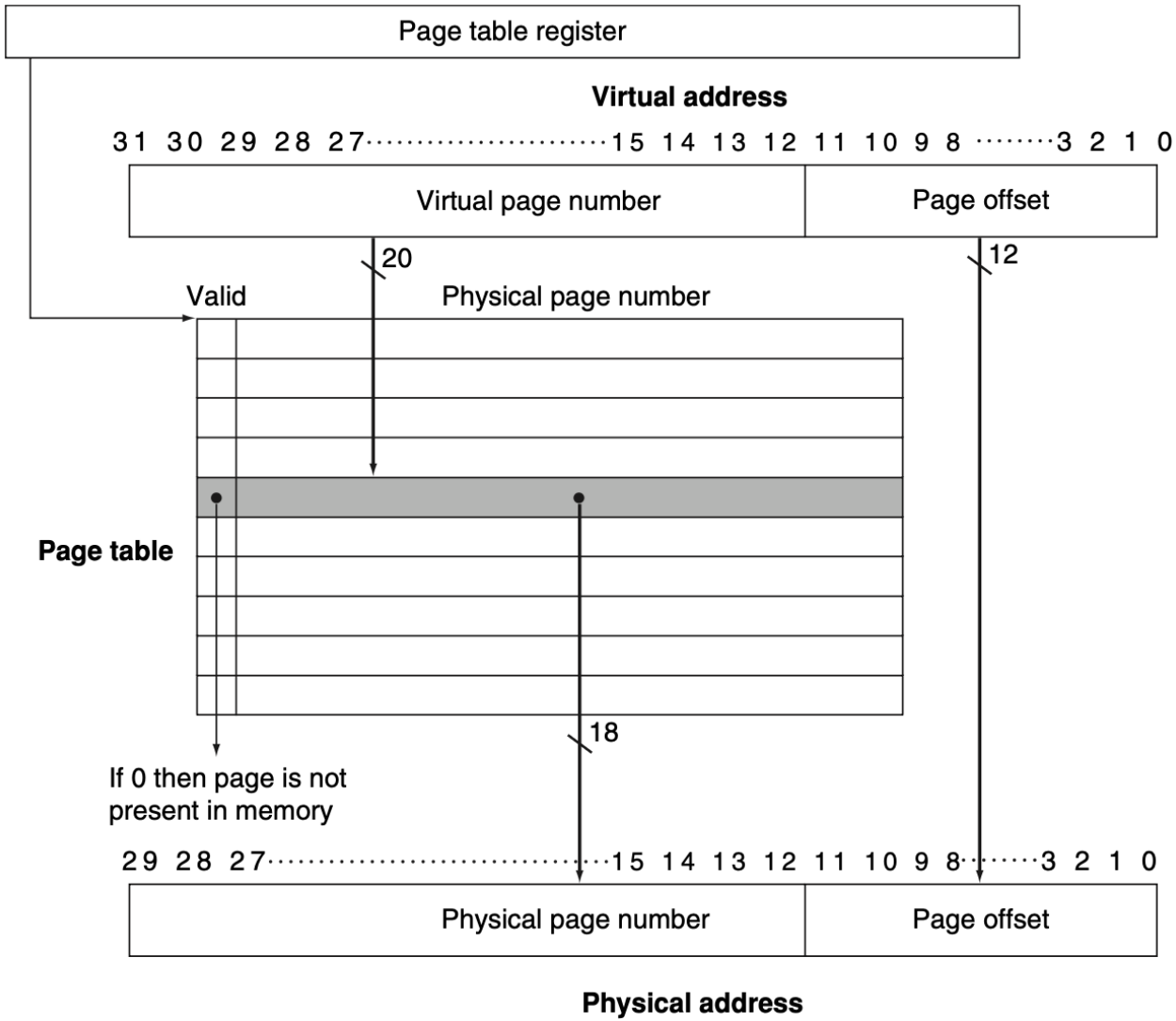
- It is typically indexed by the virtual page number

Virtual Memory - Page Table

Page table is the table which contains virtual to physical address mapping.

- It is typically indexed by the virtual page number
- Each entry in the table contains the corresponding physical page number

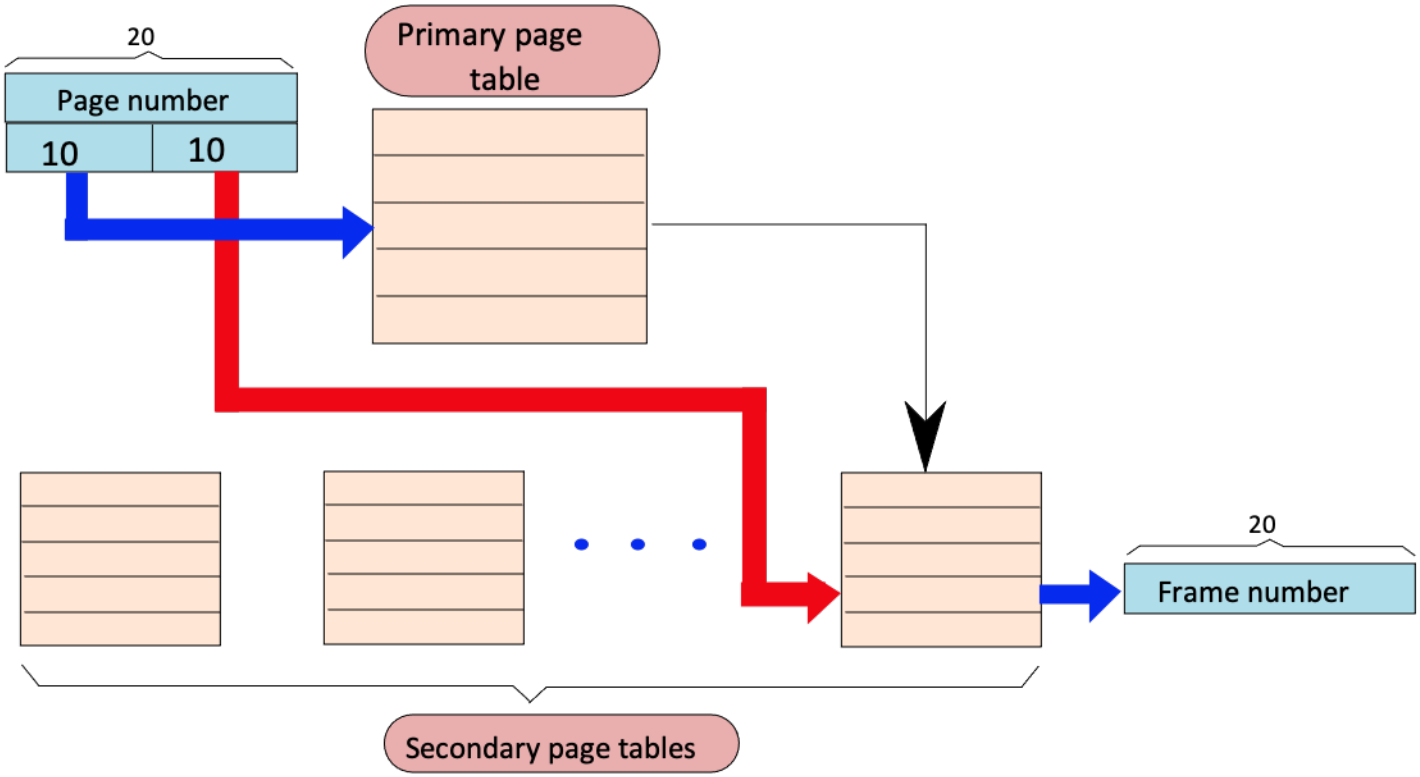
Virtual Memory - Page Table



Valid bit tells whether the page is present in main memory or not.

Virtual Memory - Two Level Page Table

We allocate only those many secondary page tables as required



Source: Basic Computer Architecture by SR Sarangi

Virtual Memory - Page Faults

- If the valid bit for a virtual page is off, a page fault occurs.

Virtual Memory - Page Faults

- If the valid bit for a virtual page is off, a page fault occurs.
- OS finds the page in next level (disk) and decides where to place the requested page in main memory.

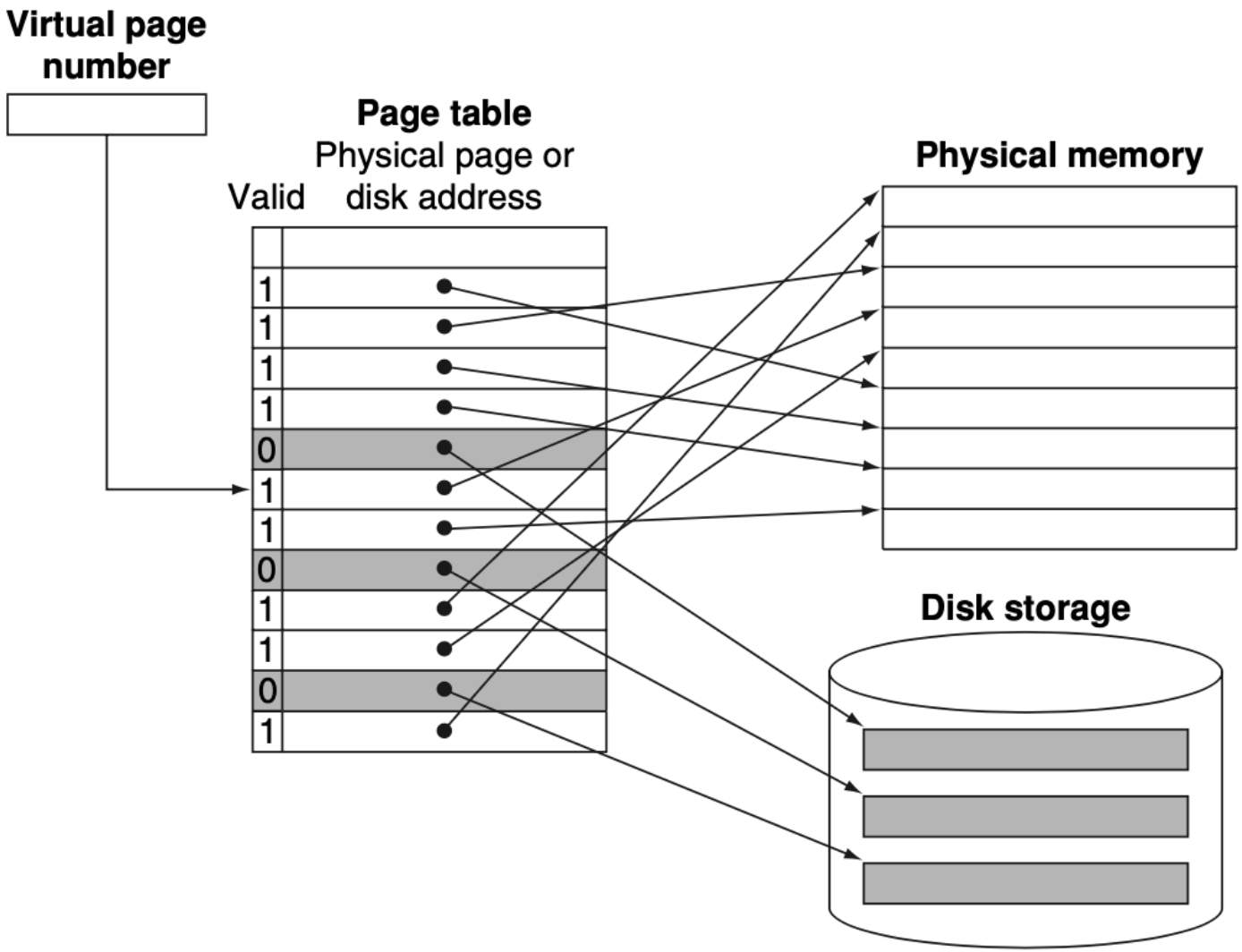
Virtual Memory - Page Faults

- If the valid bit for a virtual page is off, a page fault occurs.
- OS finds the page in next level (disk) and decides where to place the requested page in main memory.
- Virtual address alone is not enough.
- OS usually creates space on disk, called as *swap space*, for all the pages of a process.

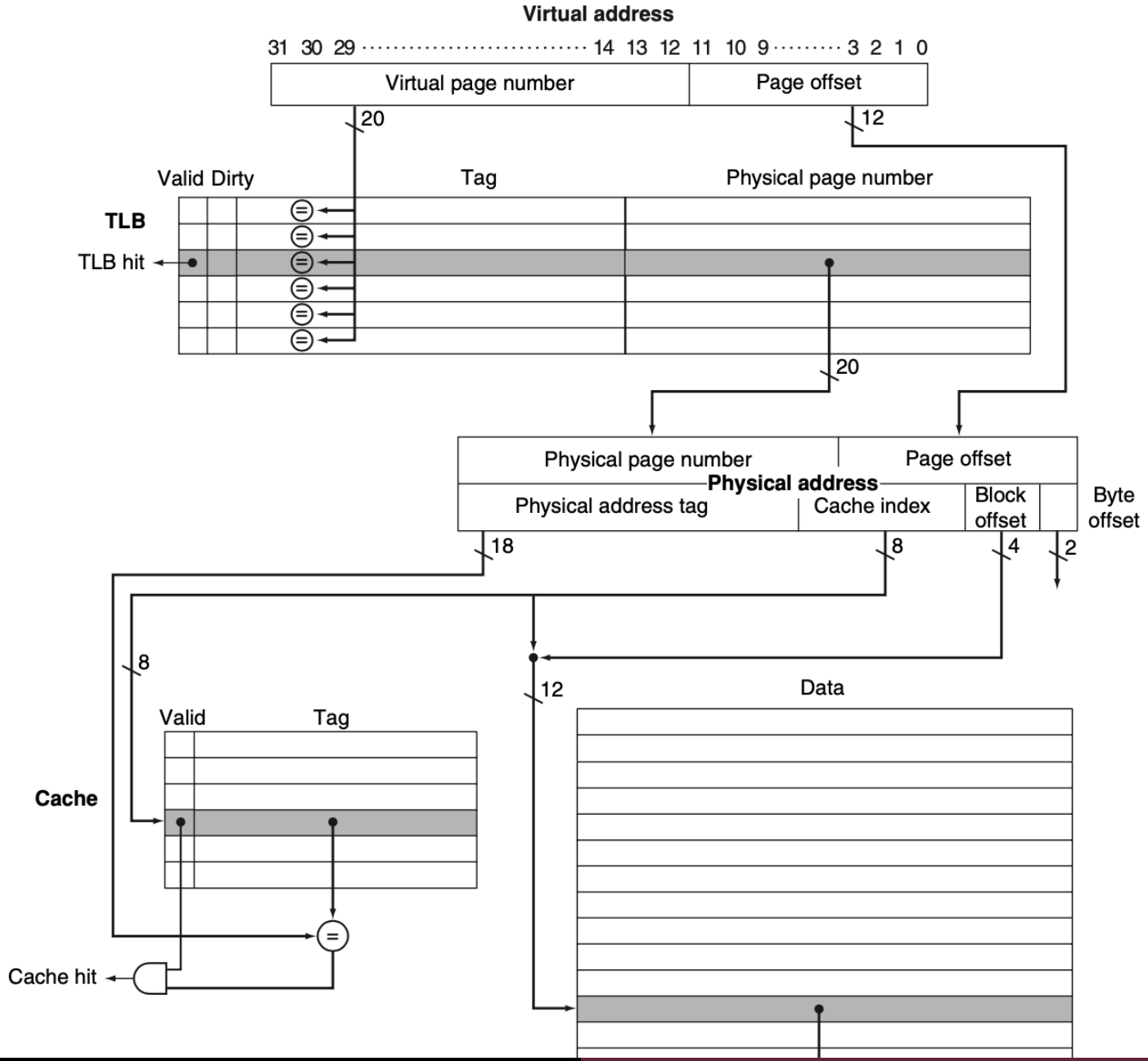
Virtual Memory - Page Faults

- If the valid bit for a virtual page is off, a page fault occurs.
- OS finds the page in next level (disk) and decides where to place the requested page in main memory.
- Virtual address alone is not enough.
- OS usually creates space on disk, called as *swap space*, for all the pages of a process.
- If main memory is full, OS follows the (approximate) LRU replacement scheme.

Virtual Memory - Swap Space



Virtual Memory - TLB



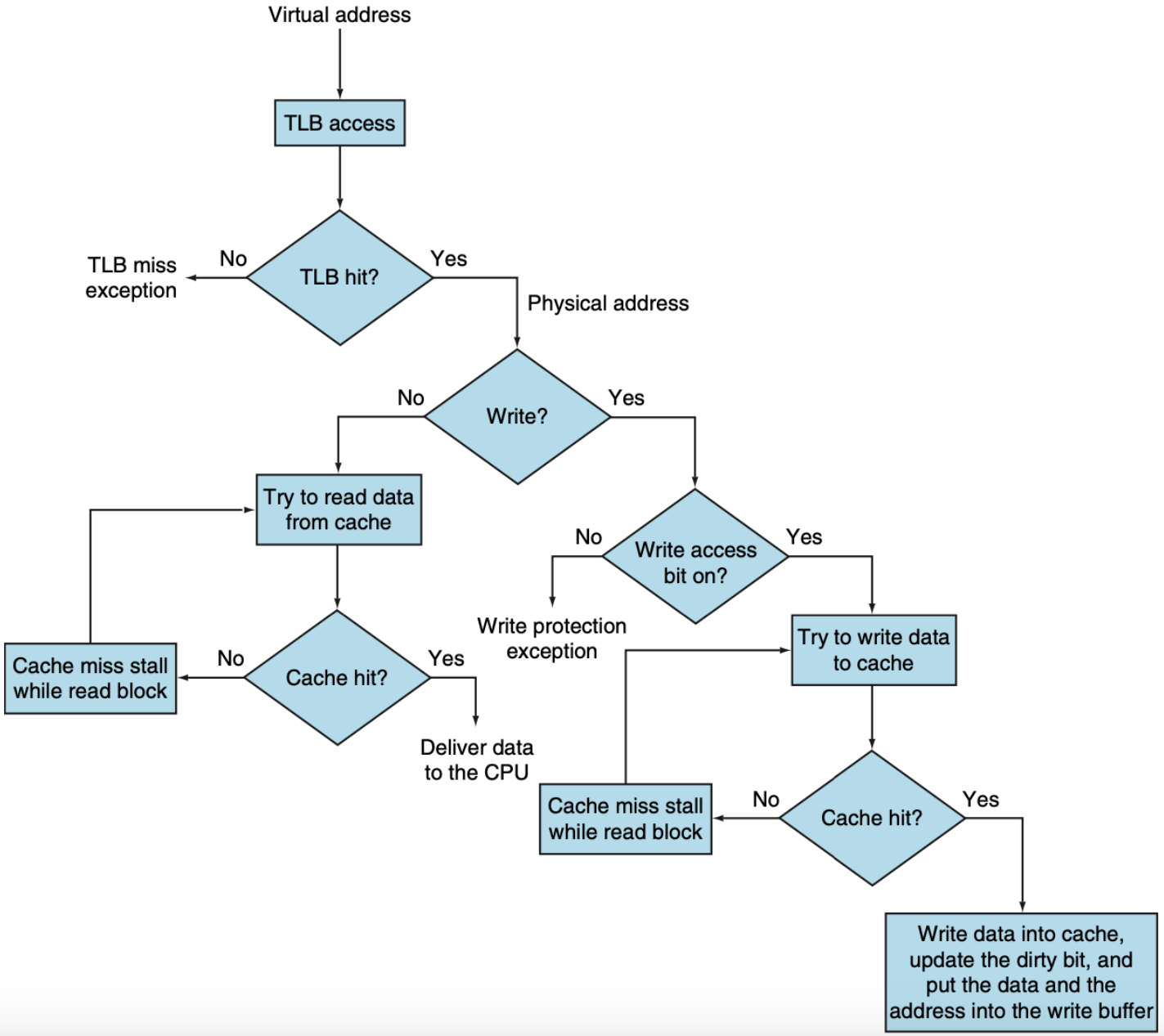
Virtual Memory - TLB

Example: Consider a system with 1 KB of main memory, 8-block cache where each block is of size 1 word, and TLB of size 4 entries. The system supports 1 KB of virtual memory which is divided into pages of size 64 B. If CPU references address 980_{ten}, find the corresponding data from cache. Shown below are the current state of TLB (left) and cache (right).

1	1000	0011
1	1111	1010
1	1010	0111
1	0101	1011

1	00011	010001110....00
1	10001	100000000....11
1	11110	000010001....00
1	10111	1111110001....01
1	10101	0101011111....00
1	10100	101010100....11
1	10000	0000011111....01
1	00101	1100111000....11

Virtual Memory - TLB & ache



Memory - Summary

Caches, TLBs, and virtual memory may initially look very different, but they rely on the same two principles of locality, and they can be understood by their answers to four questions:

Question 1: Where can a block be placed?
Answer: One place (direct mapped), a few places (set associative), or any place (fully associative).

Question 2: How is a block found?
Answer: There are four methods: indexing (as in a direct-mapped cache), limited search (as in a set-associative cache), full search (as in a fully associative cache), and a separate lookup table (as in a page table).

Question 3: What block is replaced on a miss?
Answer: Typically, either the least recently used or a random block.

Question 4: How are writes handled?
Answer: Each level in the hierarchy can use either write-through or write-back.

Memory - Three Cs Model

All misses are classified into one of three categories (the three Cs).

Memory - Three Cs Model

All misses are classified into one of three categories (the three Cs).

- *Compulsory misses:* Misses caused by the first access to a block in the cache. Also called cold-start misses.

Memory - Three Cs Model

All misses are classified into one of three categories (the three Cs).

- *Compulsory misses:* Misses caused by the first access to a block in the cache. Also called cold-start misses.
- *Capacity misses:* Misses caused when the cache cannot contain all the blocks needed during execution.

Memory - Three Cs Model

All misses are classified into one of three categories (the three Cs).

- *Compulsory misses:* Misses caused by the first access to a block in the cache. Also called cold-start misses.
- *Capacity misses:* Misses caused when the cache cannot contain all the blocks needed during execution.
- *Conflict misses:* Misses that occur when multiple blocks compete for the same location. Also called as conflict misses.

Memory - Three Cs Model

All misses are classified into one of three categories (the three Cs).

- *Compulsory misses:* Misses caused by the first access to a block in the cache. Also called cold-start misses.
- *Capacity misses:* Misses caused when the cache cannot contain all the blocks needed during execution.
- *Conflict misses:* Misses that occur when multiple blocks compete for the same location. Also called as conflict misses.

Design change	Effect on miss rate	Possible negative performance effect
Increase cache size	Decreases capacity misses	May increase access time
Increase associativity	Decreases miss rate due to conflict misses	May increase access time
Increase block size	Decreases miss rate for a wide range of block sizes due to spatial locality	Increases miss penalty. Very large block could increase miss rate

Memory - Cache Coherence

- In a multiprocessor design, all the processors generally share main memory and subsequent hierarchy but have their own caches.

Memory - Cache Coherence

- In a multiprocessor design, all the processors generally share main memory and subsequent hierarchy but have their own caches.
- Cache coherence refers to the consistency of the data shared among multiple caches.

Memory - Cache Coherence

- In a multiprocessor design, all the processors generally share main memory and subsequent hierarchy but have their own caches.
- Cache coherence refers to the consistency of the data shared among multiple caches.

Time step	Event	Cache contents for CPU A	Cache contents for CPU B	Memory contents for 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

Memory - Cache Coherence

A memory system is coherent if

- Reads to a location X by multiple processor are consistent.
- Writes to the same location are serialized.

Enforcing Coherence – Snooping

- The protocols to maintain coherence for multiple processors are called cache coherence protocols.

Enforcing Coherence – Snooping

- The protocols to maintain coherence for multiple processors are called cache coherence protocols.
- The most popular cache coherence protocol is snooping.

Enforcing Coherence – Snooping

- The protocols to maintain coherence for multiple processors are called cache coherence protocols.
- The most popular cache coherence protocol is snooping.
  - Snooping logic in the processor broadcasts a message over the bus each time a word in its cache has been modified.

Enforcing Coherence – Snooping

- The protocols to maintain coherence for multiple processors are called cache coherence protocols.
- The most popular cache coherence protocol is snooping.
  - Snooping logic in the processor broadcasts a message over the bus each time a word in its cache has been modified.
  - All the processors snoop on the bus looking for such messages.

Enforcing Coherence – Snooping

- The protocols to maintain coherence for multiple processors are called cache coherence protocols.
- The most popular cache coherence protocol is snooping.
  - Snooping logic in the processor broadcasts a message over the bus each time a word in its cache has been modified.
  - All the processors snoop on the bus looking for such messages.
  - When a processor detects that another processor has changed a value at an address existing in its own cache, the snooping logic invalidates that entry in its cache.

Enforcing Coherence – Snooping

- The protocols to maintain coherence for multiple processors are called cache coherence protocols.
- The most popular cache coherence protocol is snooping.
  - Snooping logic in the processor broadcasts a message over the bus each time a word in its cache has been modified.
  - All the processors snoop on the bus looking for such messages.
  - When a processor detects that another processor has changed a value at an address existing in its own cache, the snooping logic invalidates that entry in its cache.
  - This style of protocol is called a write invalidate protocol.

Memory - Cache Coherence

Processor activity	Bus activity	Contents of CPU A's cache	Contents of CPU B's cache	Contents of memory location X
				0
CPU A reads X	Cache miss for X	0		0
CPU B reads X	Cache miss for X	0	0	0
CPU A writes a 1 to X	Invalidation for X	1		0
CPU B reads X	Cache miss for X	1	1	1

During the write A takes exclusive access, and the copy held by the reading processor (B) is invalidated.