Cache Memories

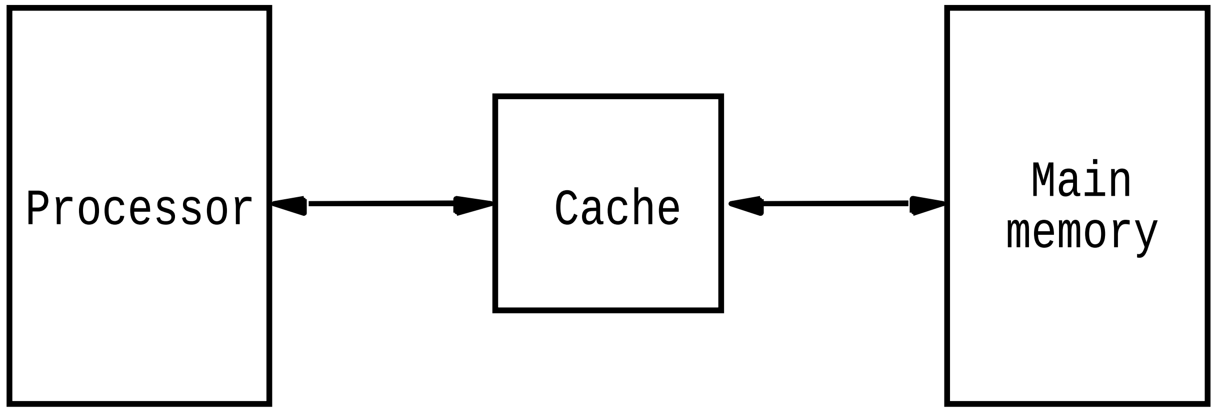
Cache memories

- Processor is much faster than the main memory
  - time waiting
  - performance

  - Speed of the main memory cannot be increased beyond a certain point.
  - Cache memory makes the main memory appear faster to the processor than it really is

- Cache memory - “locality of reference”
    - Temporal locality of reference:
    - Spatial locality of reference:

Cache memories



- *Miss - Block transferred*
- *Subsequent references will be hit*
- *Size of cache smaller than MM*
- *“mapping function” : decide the blocks to be cache*
- *“replacement algorithm “ cache is full*

Cache hit

- ***Read hit:***
  - *The data is obtained from the cache.*

- ***Write hit:***
  - Cache has a replica of the contents of the main memory.
  - **write-through protocol** - updated simultaneously
  - **write-back or copy-back protocol** - Update only cache set - **dirty bit** or modified bit
    - MM updated during block replacement

Cache miss

- **Read miss:**
  - Transferred the block from the memory
    - After the block is transferred, the desired word is forwarded to the processor.
    - **load-through or early-restart -** desired word may also be forwarded to the processor as soon as it is transferred without waiting for the entire block to be transferred
- **Write-miss:**
  - Write-through protocol - contents of the main memory are updated directly
  - Write-back protocol - word is first brought into the cache, overwritten with new information

Cache Coherence Problem

- ***“valid bit”***
  - *for each block*
  - *set to 1 – block is valid else it is 0.*
  - ***Valid bit :***
    - ***0 – Power on , invalid data, disk changes the block in main memory and copy resides in cache***
    - ***1- new block loaded, valid data***

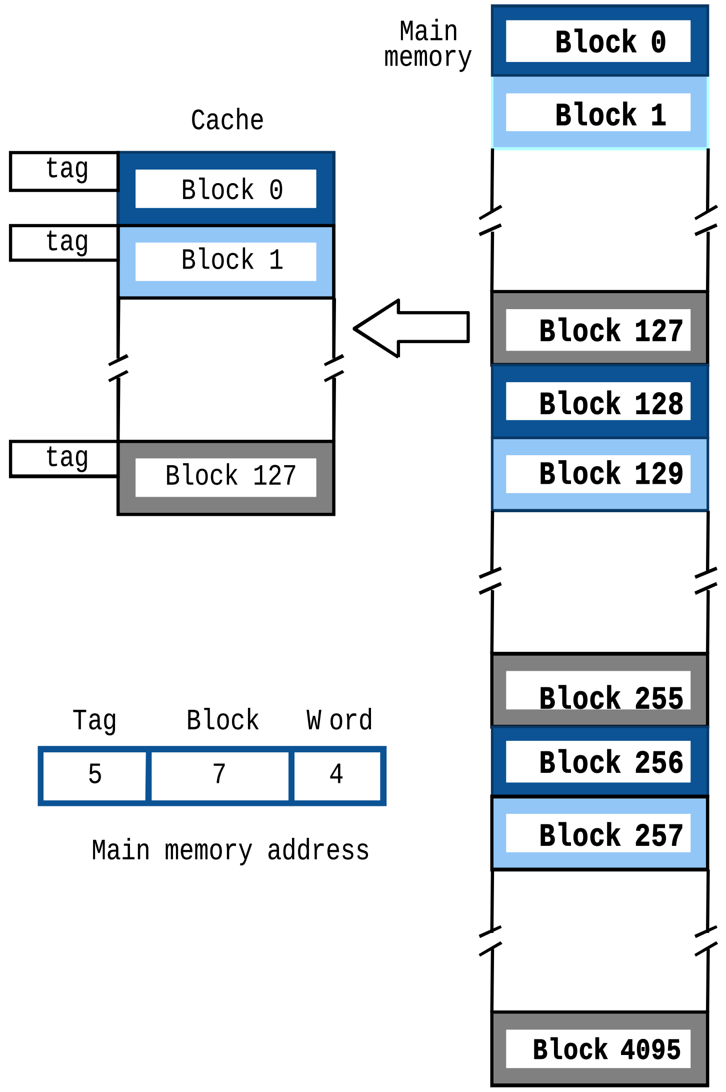
- ***cache coherence problem***
  - *The copies of the data in the cache, and the main memory are different.*
  - *When we copy data from main memory into disk and the data in the cache may also have changed and is indicated by the dirty bit.*
  - *One option is to force a* ***write-back before*** *the main memory update disk.*

Mapping functions

- Mapping functions determine how memory blocks are placed in the cache.
- Three mapping functions:
  - Direct mapping
  - Associative mapping
  - Set-associative mapping.

Direct mapping

$2^1 = 2$	$2^9 = 512$
$2^2 = 4$	$2^{10} = 1024$
$2^3 = 8$	$2^{11} = 2048$
$2^4 = 16$	$2^{12} = 4096$
$2^5 = 32$	$2^{13} = 8192$
$2^6 = 64$	$2^{14} = 16384$
$2^7 = 128$	$2^{15} = 32768$
$2^8 = 256$	$2^{16} = 65536$



- j modulo 128
  - 0 maps to 0
  - 129 maps to 1

- More than one memory block is mapped onto the same position in the cache.
- May lead to **contention** for cache blocks even if the cache is not full.

- Simple to implement but not very flexible.

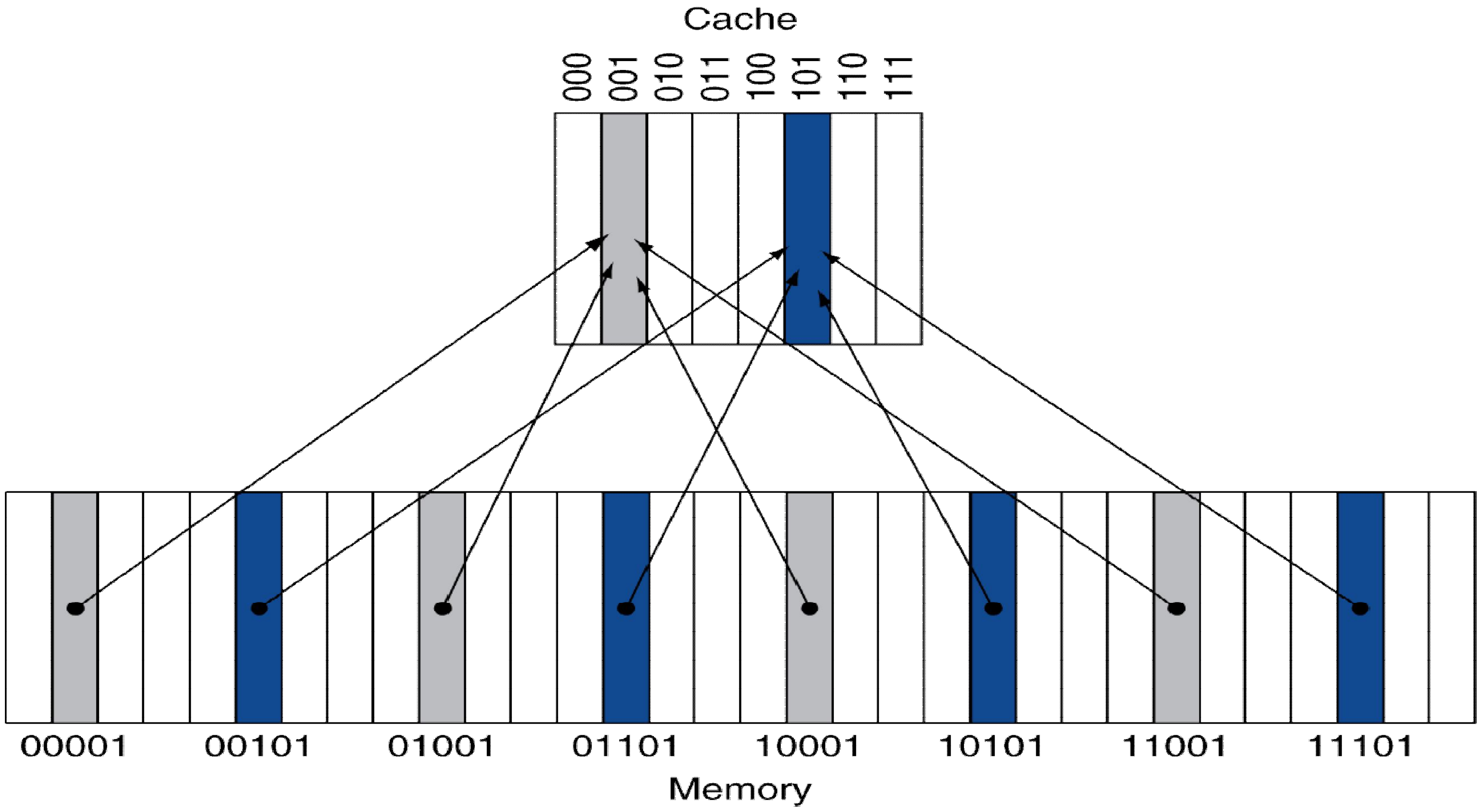
  - Total size of cache is 2048 (2K) words
    - uses 11 bit address line
    - 128 blocks of 16 words each
    - 7 bits for blocks and 4 bits for words

  - *Main memory has 64K words*
    - *uses 16 bit address line*
    - *4K blocks of 16 words each.,*
    - *12 bits for blocks and 4 bit for words*
    - *5 bits tag 7 bits block and 4 bits word*

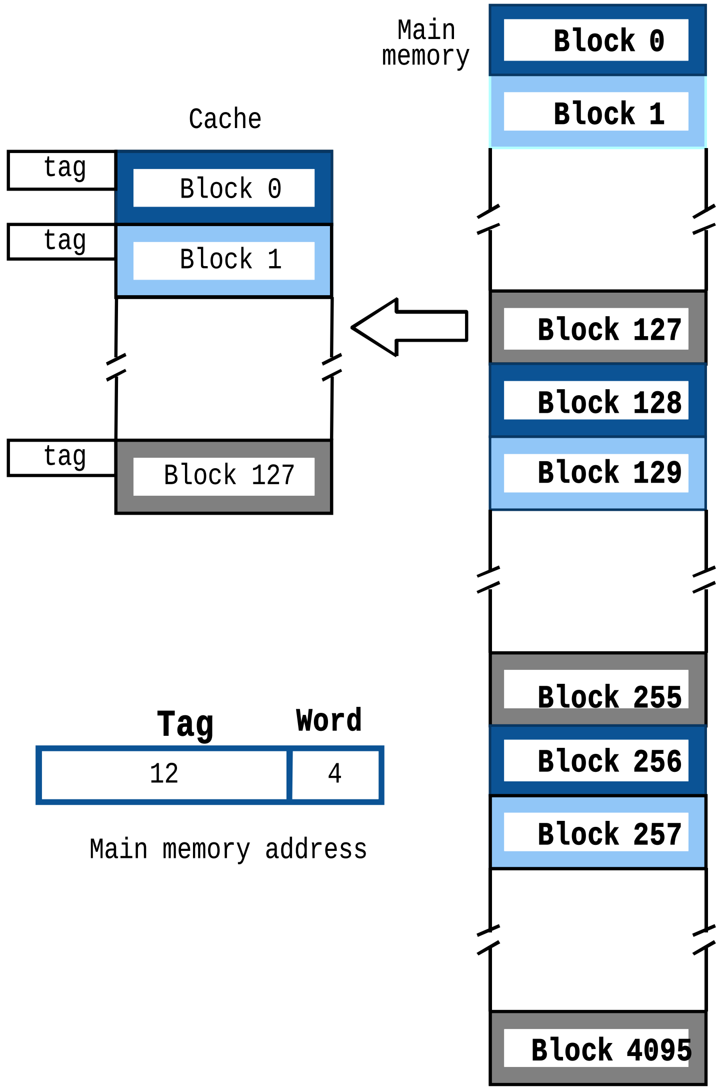
Direct mapping

- Location determined by address
- Direct mapped: only one choice
  - (Block address) modulo (#Blocks in cache)

- #Blocks is a power of 2
- Use low-order address bits

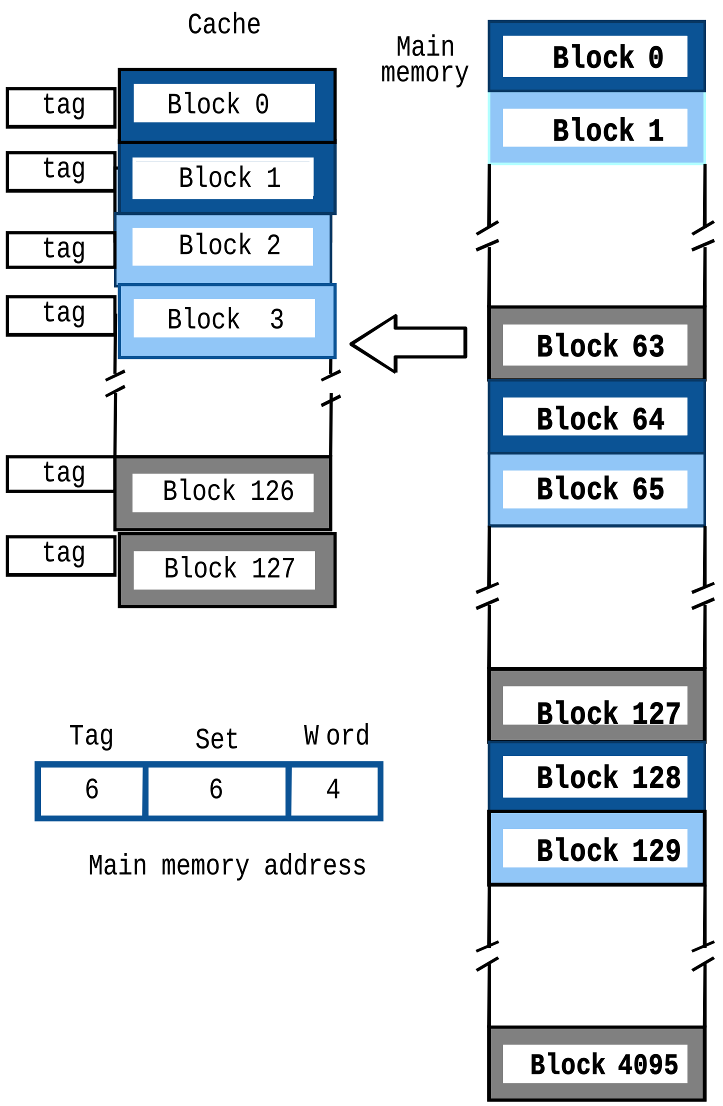


Associative mapping



- *Main memory block can be placed into any cache position.*
- *Memory address is divided into two fields:*
- *Flexible, and uses cache space efficiently.*
- *Replacement algorithms can be used to replace an existing block in the cache when the cache is full.*
- *Cost is higher than direct-mapped cache because of the need to search all 128 patterns to determine whether a given block is in the cache*

Set-Associative mapping



•Blocks of cache are grouped into sets.

•Mapping function allows a block of the main memory to reside in any block of a specific set.

•Divide the cache into 64 sets, with two blocks per set.
•Memory block 0, 64, 128 etc. map to set 0, and they can occupy either of the two positions.
•Memory address is divided into three fields:
- 4 bits for word
- 6 bit field determines the set number.
- High order 6 bit fields are compared to the tag fields of the two blocks in a set.
•Set-associative mapping combination of direct and associative mapping.

•Number of blocks per set is a design parameter.
- One extreme is to have all the blocks in one set, requiring no set bits (fully associative mapping).
- Other extreme is to have one block per set, is the same as direct mapping.

Cache Example

- 8-blocks, 1 word/block, direct mapped
- Initial state

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

Direct Mapping Cache Example

Word addr	Binary addr	Hit/miss	Cache block
22	10 110	Miss	110
26	11 010	Miss	010

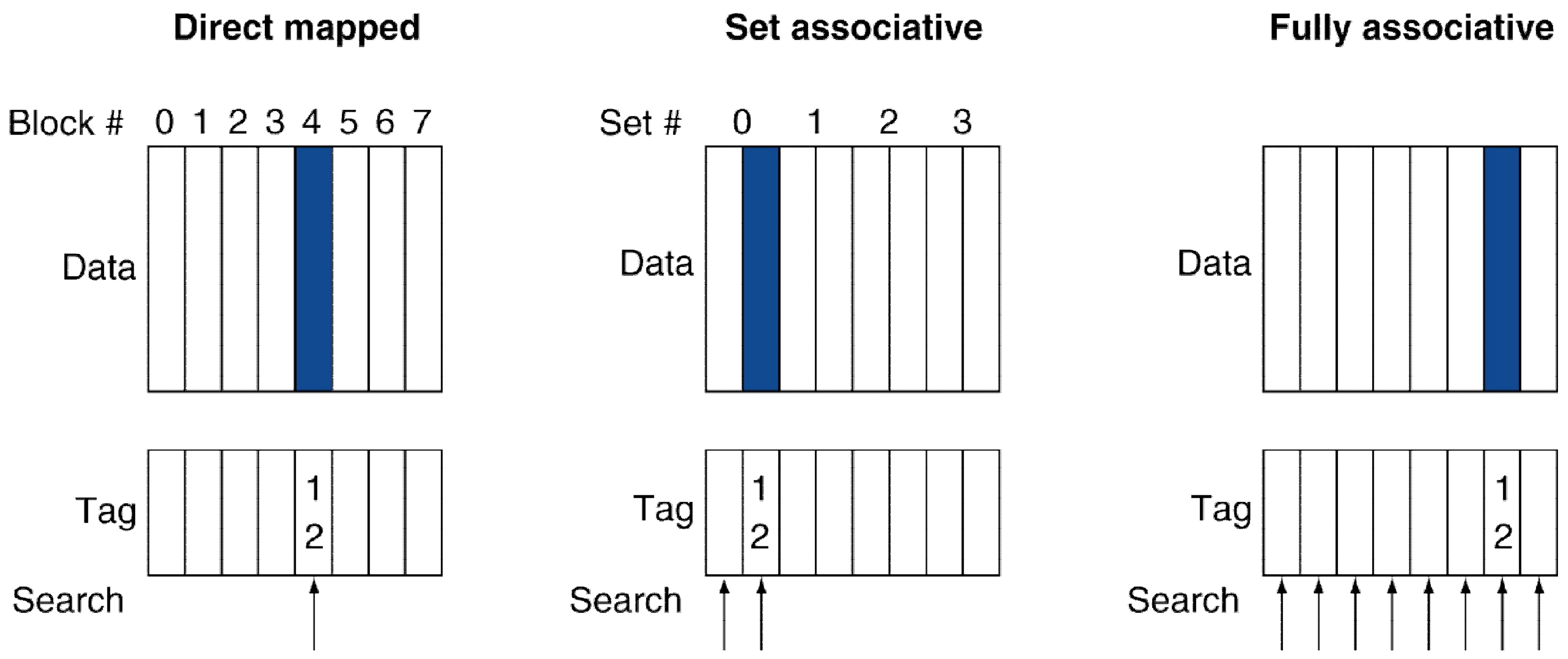
Index	V	Tag	Data
000	N		
001	N		
010	**Y**	**11**	**Mem[11010]**
011	N		
100	N		
101	N		
110	**Y**	**10**	**Mem[10110]**
111	N		

Cache Example

Word addr	Binary addr	Hit/miss	Cache block
22	10 110	Hit	110
26	11 010	Hit	010
16	10 000	Miss	000

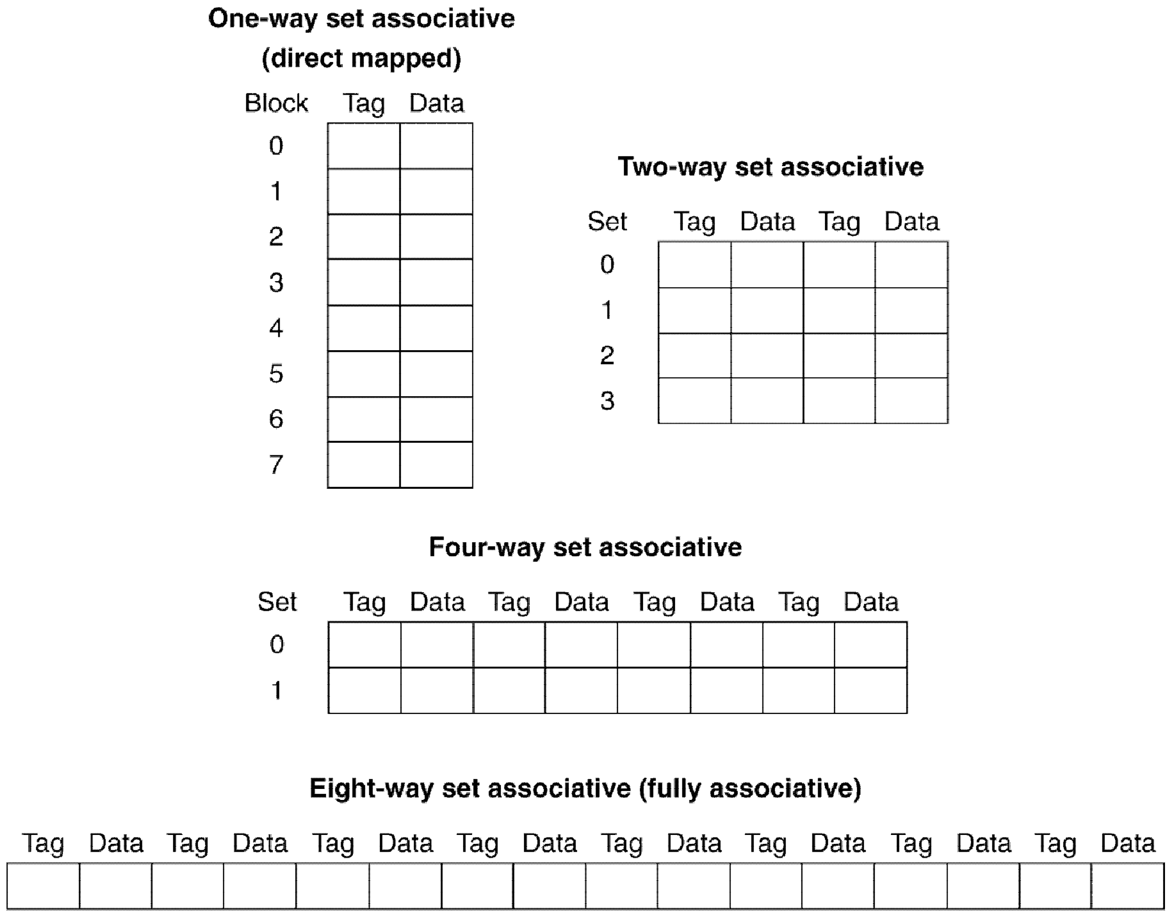
Index	V	Tag	Data
000	**Y**	**10**	**Mem[10000]**
001	N		
010	Y	11	Mem[11010]
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Associative Cache Example



Spectrum of Associativity

- For a cache with 8 entries


One-way set associative
(direct mapped)
Block Tag Data
0 1 2 3 4 5 6 7
Two-way set associative
Set Tag Data Tag Data
0 1 2 3
Four-way set associative
Set Tag Data Tag Data Tag Data Tag Data
0 1
Eight-way set associative (fully associative)
Tag Data Tag Data Tag Data Tag Data Tag Data Tag Data Tag Data Tag Data

Associativity Example

- Compare 4-block caches
  - Direct mapped, 2-way set associative, fully associative
  - Block access sequence: 0, 8, 0, 6, 8
- Direct mapped

Block address	Cache index	Hit/miss	Cache content after access			
			0	1	2	3
0	0	miss	**Mem[0]**			
8	0	miss	**Mem[8]**			
0	0	miss	**Mem[0]**			
6	2	miss	Mem[0]		**Mem[6]**	
8	0	miss	**Mem[8]**		Mem[6]	

Associativity Example

- 2-way set associative

Block address	Cache index	Hit/miss	Cache content after access			
			Set 0		Set 1	
0	0	miss	**Mem[0]**			
8	0	miss	Mem[0]	**Mem[8]**		
0	0	hit	**Mem[0]**	Mem[8]		
6	0	miss	Mem[0]	**Mem[6]**		
8	0	miss	**Mem[8]**	Mem[6]		

- Fully associative

Block address		Hit/miss	Cache content after access			
0		miss	**Mem[0]**			
8		miss	Mem[0]	**Mem[8]**		
0		hit	**Mem[0]**	Mem[8]		
6		miss	Mem[0]	Mem[8]	**Mem[6]**	
8		hit	Mem[0]	**Mem[8]**	Mem[6]	

Page replacement policies

- Random: Choose a page at random to evict from the cache.
  - Takes no advantage of any temporal or spatial localities
- First-in, First-out(FIFO): Evict the page that has been in the cache the longest
  - Queue
  - Try to take advantage of temporal locality
- Least recently used (LRU): Evict the page whose last request occurred furthest in the past. (property of locality of reference)
  - priority queue Q

Page replacement policies

2	0
1	2
1	2
3	3

- Least recently used (LRU)
- Eg:
  - 4 way set-associative cache - use 2-bit counter for each block
  - Hit
    - Set the block counter =0
    - All blocks having counter value less than the hit block is incremented
    - all others remain unchanged

2	3
	0
1	2

  - Miss
    - *set is not full load the new block set its counter 0*
    - values of all other counters are increased by one

    - set is full, block with the counter value 3 is removed
    - new block is put in its place, and its counter is set to 0.
    - other three block counters are incremented by one

2	3
1	2
1	2
3	0

Cache Performance considerations

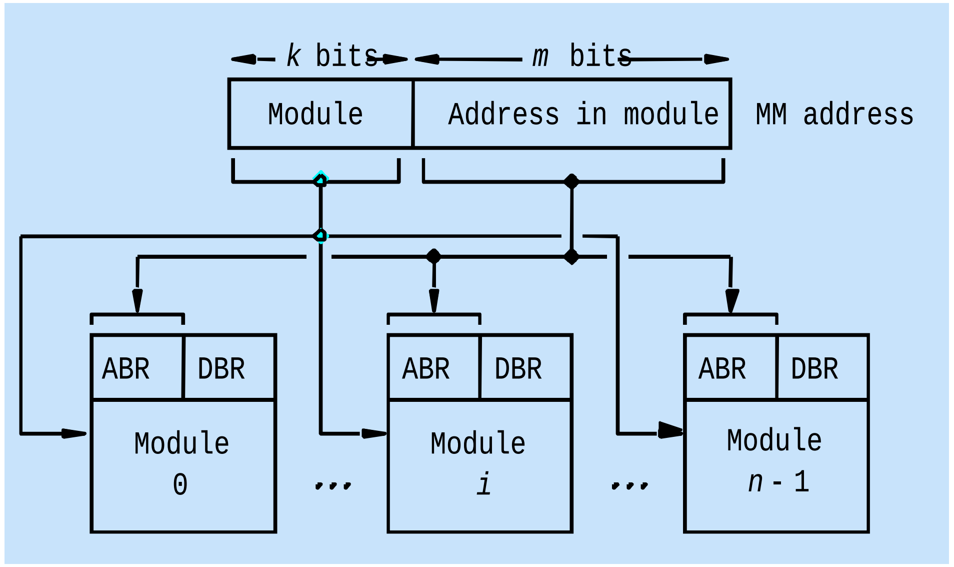
Performance considerations

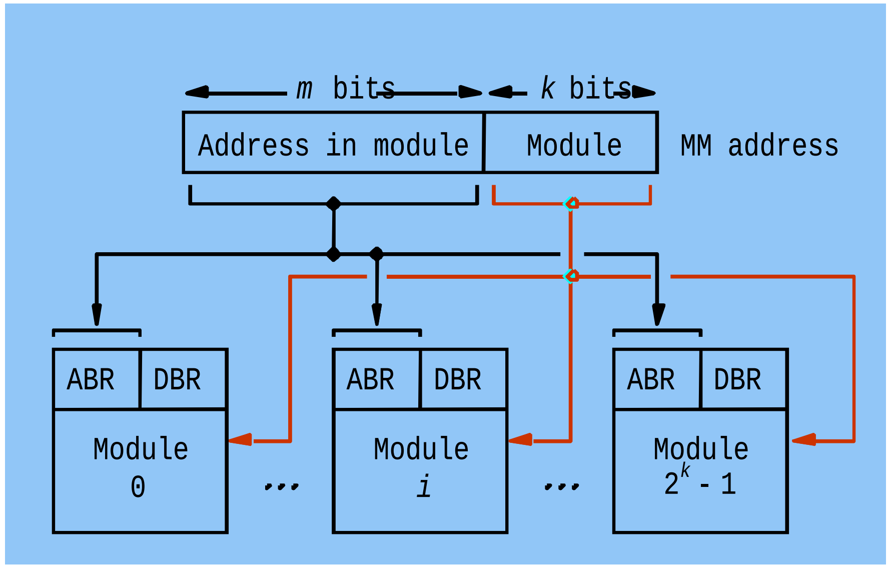
- – Price/performance ratio
- Performance of a processor depends on:
  - How fast machine instructions can be brought into the processor for execution.
  - How fast the instructions can be executed.

Interleaving

- Arranges addressing so that successive words in the address space are placed in different modules.

Methods of address layouts





- ***Consecutive words are placed in a module.***
- ***only one module is busy at a time.***

•Consecutive words are located in consecutive modules.
•While transferring a block of data, several memory modules can be kept busy at the same time.

Hit Rate and Miss Penalty

- Hit rate can be improved by increasing block size, while keeping cache size constant
- **Block sizes that are neither very small nor very large give best results**.
- Miss penalty can be reduced if load-through approach is used when loading new blocks into cache.

Caches on the processor chip

- 1 levels of cache
  - $T_{avg} = hC + (1 - h)M$
    - *h - hit rate*
    - *M – Miss penalty*
    - *C- time to access the cache*
- In high performance processors 2 levels of caches are normally used.

  $T_{avg} = h1c1+(1-h1)h2c2+(1-h1)(1-h2)M$

Other Performance Enhancements

Write buffer

- *Write-through:*
- *Processor places each write request into the buffer and continues execution.*

- *Write-back:*
- *Fast write buffer can hold the block to be written, and the new block can be read first.*

Other Performance Enhancements (Contd.,)

Prefetching

- *Prefetch the data into the cache before they are actually needed, or a before a Read miss occurs.*
- *Prefetching can be accomplished through hardware or software*

Other Performance Enhancements (Contd.,)

Lockup-Free Cache

- *Cache structure which supports multiple outstanding misses is called a lockup free cache.*
- *Since only one miss can be serviced at a time, a lockup free cache must include circuits that keep track of all the outstanding misses.*

Virtual Memory

The Memory System

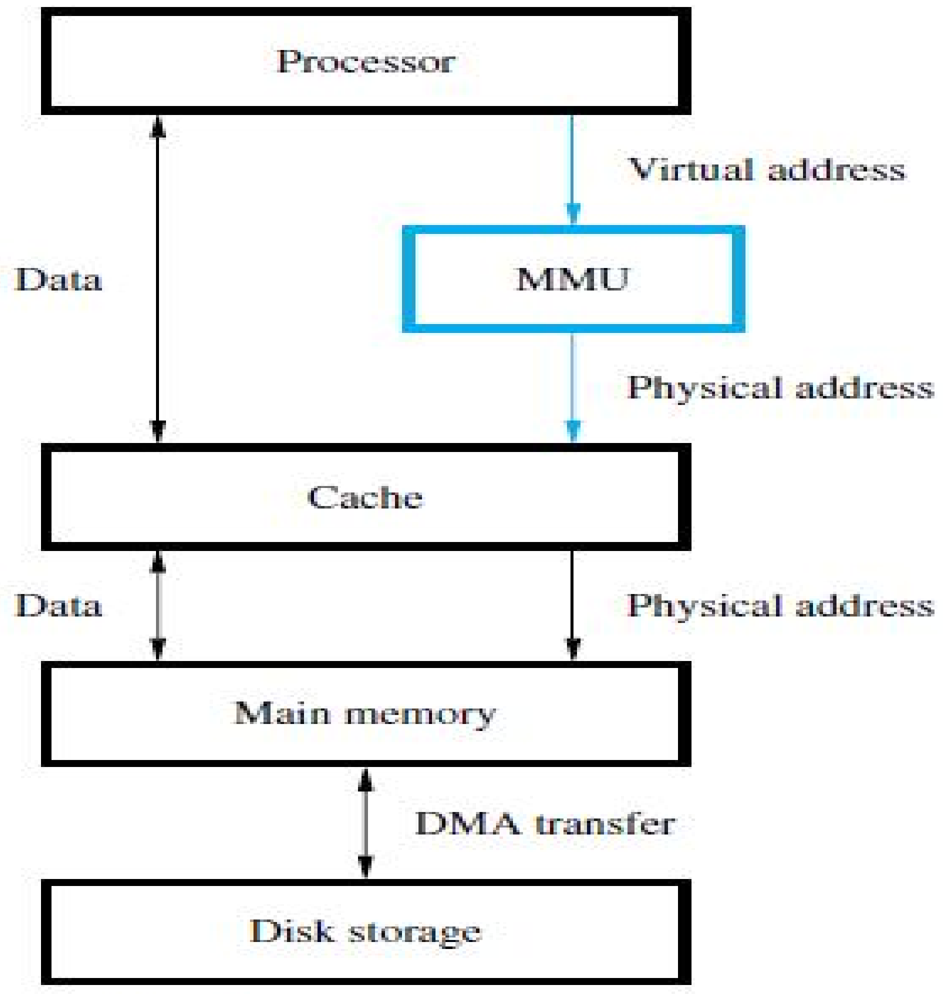
Virtual Memory

- modern computer systems, the physical main memory is not as large as the address space of the processor
- Virtual Memory
  - Managed by CPU hardware and OS
  - Programs share main memory
  - Programs gets a private virtual address space holding its frequently used code and data
  - CPU and OS translate virtual addresses to physical addresses

Virtual Memory

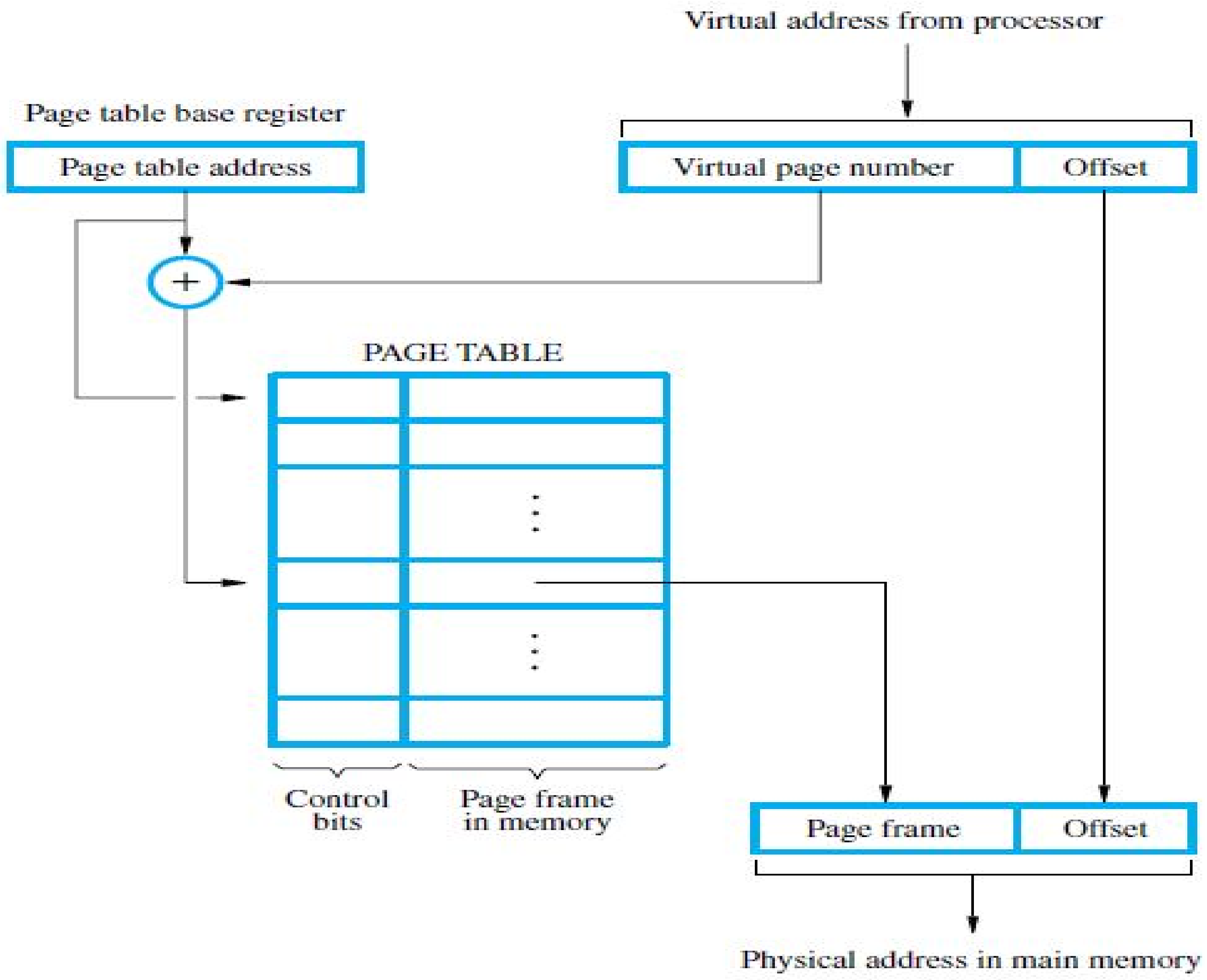
- Binary addresses that the processor issues for either instructions or data are called ***virtual or logical addresses***
- *These addresses are translated into* ***physical*** **addresses** by a combination of hardware and software actions
  - *Memory Management Unit (MMU),*
    - *keeps track of which parts of* the virtual address space are in the physical memory.
    - Hit - MMU translates the virtual address into the corresponding physical address
    - Miss - the MMU causes the operating system to transfer the data from the disk to the memory - using the DMA scheme

Virtual Memory



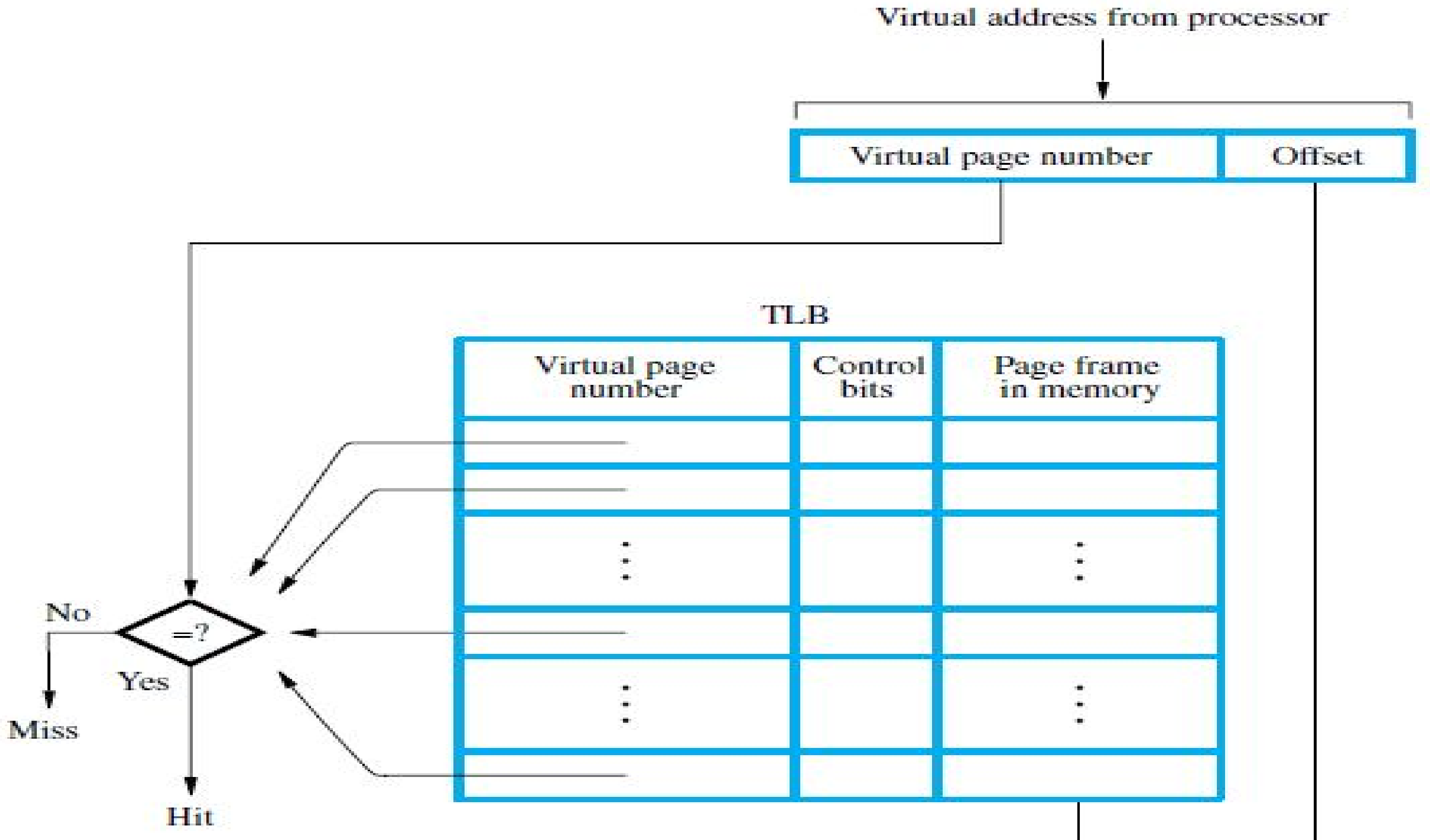
Virtual memory organization.

Address Translation



Virtual-memory address translation.

Fast Translation Using a TLB



Use of an associative-mapped TLB.

Fast Translation Using a TLB

- Portion of page table maintained within the MMU – *Translation Lookaside Buffer (TLB)*
- *TLB* - consists of the entries corresponding to the most recently accessed pages.
- When the operating system changes the contents of a page table, it must
simultaneously invalidate the corresponding entries in the TLB. One of the
control bits in the TLB is provided for this purpose. When an entry is invalidated,

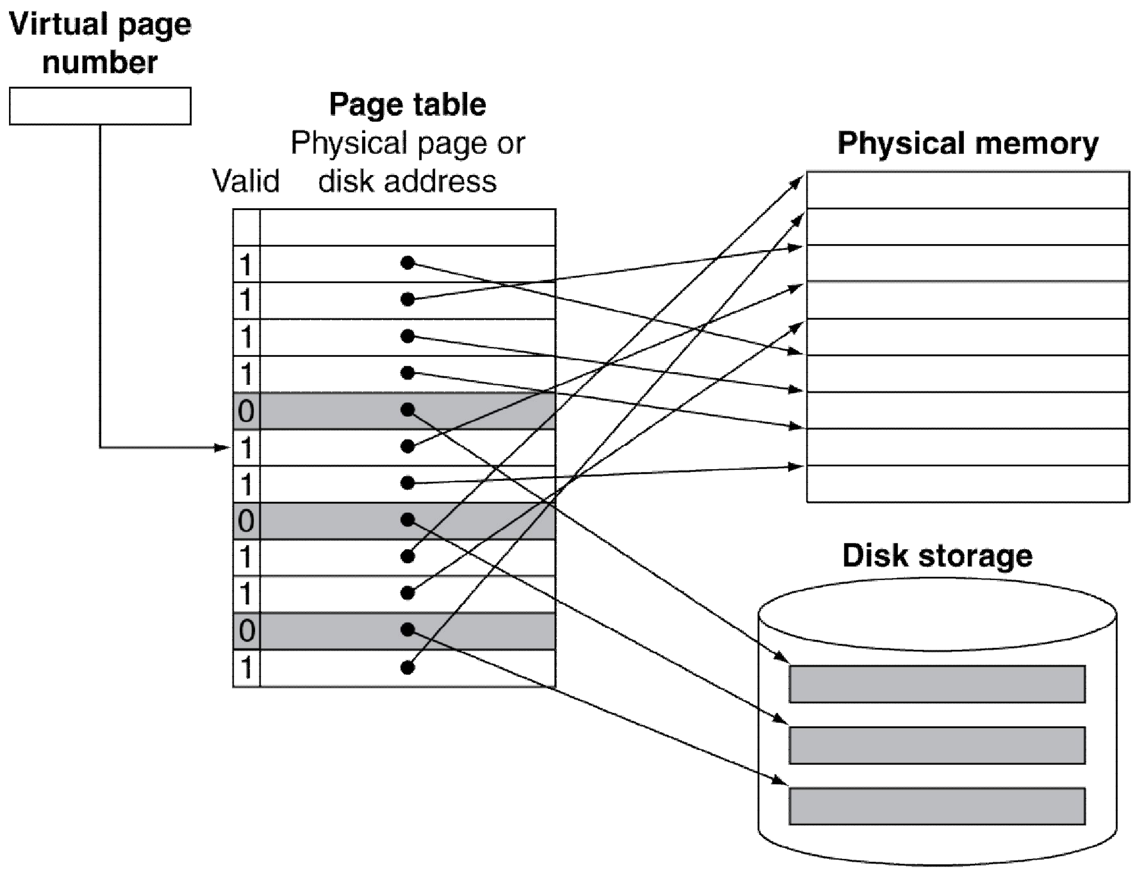
Page Fault Penalty

- On page fault, the page must be fetched from disk
  - Takes millions of clock cycles
  - Handled by OS code
- Try to minimize page fault rate
  - Fully associative placement
  - Smart replacement algorithms

Page Tables

- Stores placement information
  - Array of page table entries, indexed by virtual page number
  - Page table register in CPU points to page table in physical memory
- If page is present in memory
  - PTE stores the physical page number
  - Plus other status bits (referenced, dirty, …)
- If page is not present
  - PTE can refer to location in swap space on disk

Mapping Pages to Storage



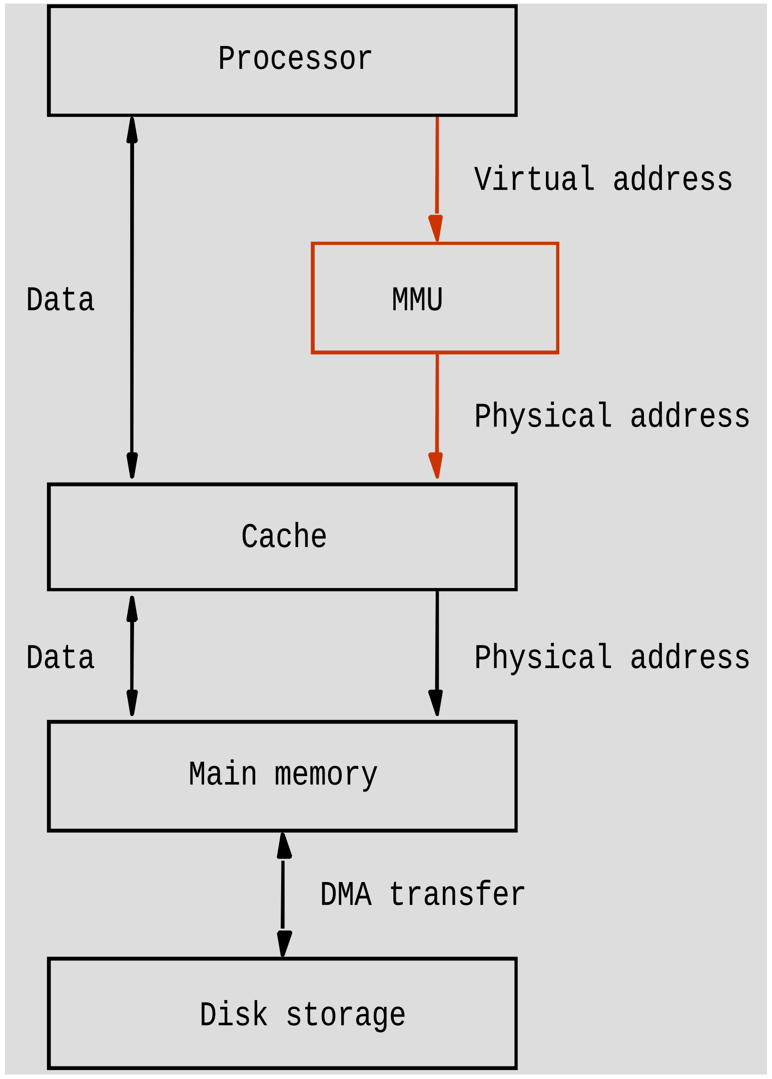
TLB Misses

- If page is in memory
  - Load the PTE from memory and retry
  - Could be handled in hardware
    - Can get complex for more complicated page table structures
  - Or in software
    - Raise a special exception, with optimized handler
- If page is not in memory (page fault)
  - OS handles fetching the page and updating the page table
  - Then restart the faulting instruction

TLB Miss Handler

- TLB miss indicates
  - Page present, but PTE not in TLB
  - Page not preset
- Must recognize TLB miss before destination register overwritten
  - Raise exception
- Handler copies PTE from memory to TLB
  - Then restarts instruction
  - If page not present, page fault will occur

Virtual memory organization



- *Memory management unit (MMU) translates virtual addresses into physical addresses.*
- *If the desired data or instructions are in the main memory they are fetched as described previously.*
- *If the desired data or instructions are not in the main memory, they must be transferred from secondary storage to the main memory.*
- *MMU causes the operating system to bring the data from the secondary storage into the main memory.*