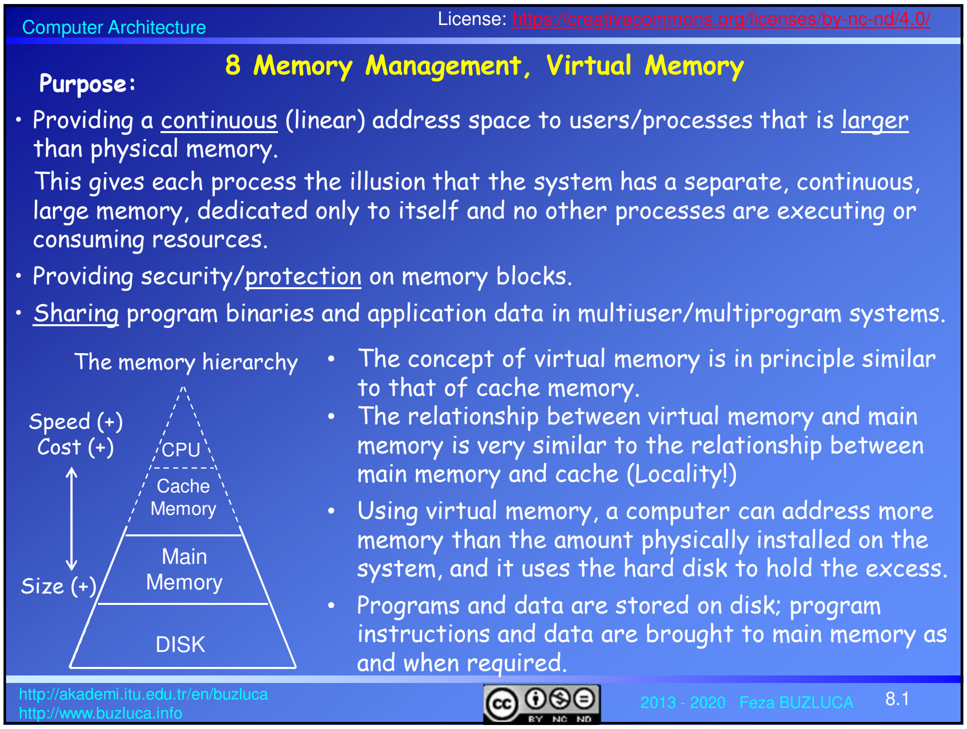


8 Memory Management, Virtual Memory

Purpose:

- Providing a continuous (linear) address space to users/processes that is larger than physical memory.
  This gives each process the illusion that the system has a separate, continuous, large memory, dedicated only to itself and no other processes are executing or consuming resources.
- Providing security/protection on memory blocks.
- Sharing program binaries and application data in multiuser/multiprogram systems.

The memory hierarchy

Speed (+)
Cost (+)

Size (+)

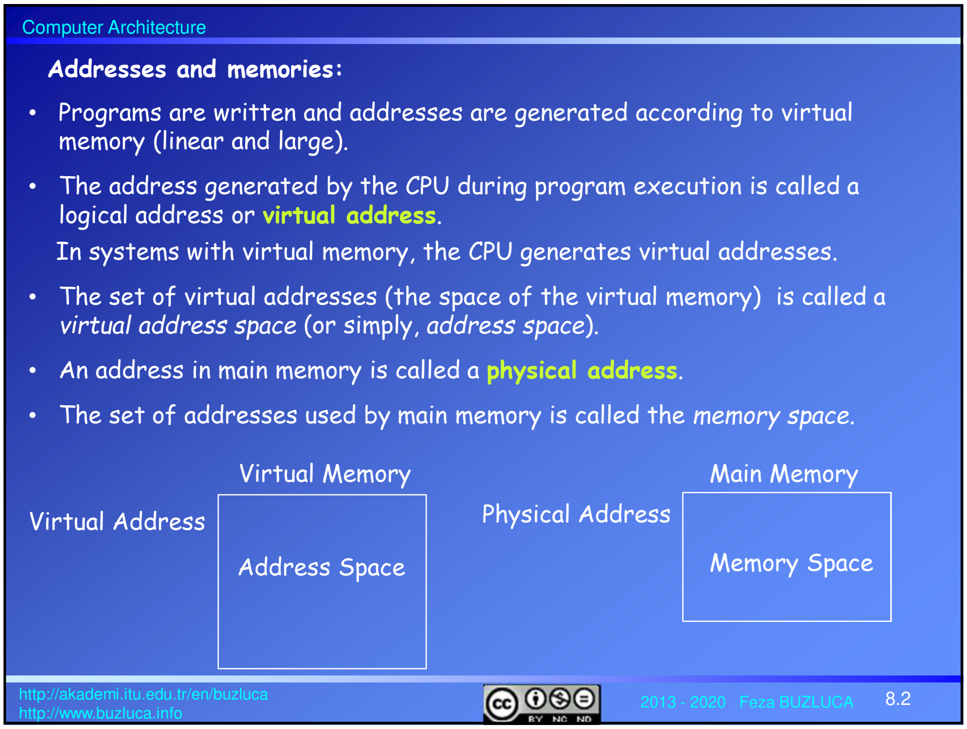
CPU

Cache Memory

Main Memory

DISK

- The concept of virtual memory is in principle similar to that of cache memory.
- The relationship between virtual memory and main memory is very similar to the relationship between main memory and cache (Locality!)
- Using virtual memory, a computer can address more memory than the amount physically installed on the system, and it uses the hard disk to hold the excess.
- Programs and data are stored on disk; program instructions and data are brought to main memory as and when required.



Addresses and memories:

- Programs are written and addresses are generated according to virtual memory (linear and large).
- The address generated by the CPU during program execution is called a logical address or **virtual address**.
  In systems with virtual memory, the CPU generates virtual addresses.
- The set of virtual addresses (the space of the virtual memory) is called a *virtual address space* (or simply, *address space*).
- An address in main memory is called a **physical address**.
- The set of addresses used by main memory is called the *memory space*.

Virtual Memory

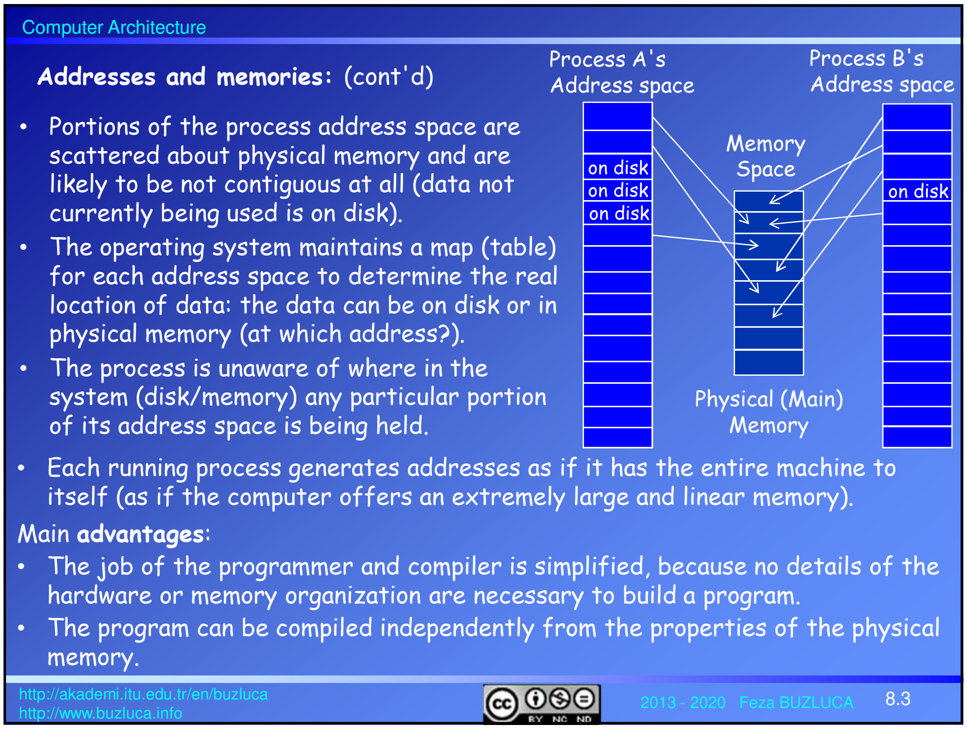
Virtual Address

Address Space

Main Memory

Physical Address

Memory Space



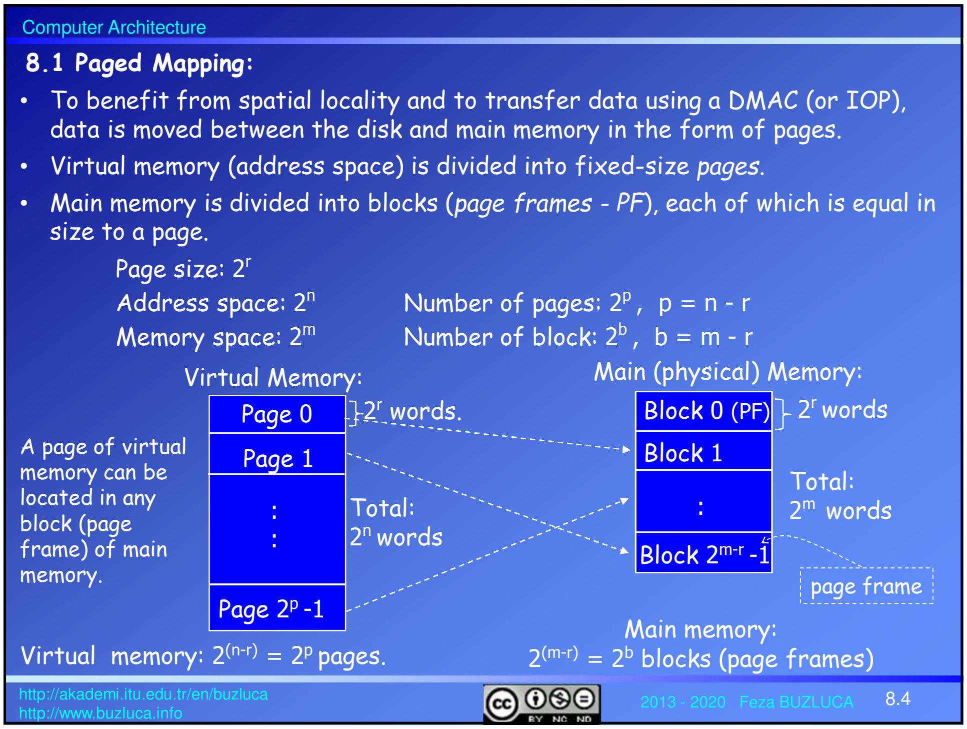
Addresses and memories: (cont'd)

- Portions of the process address space are scattered about physical memory and are likely to be not contiguous at all (data not currently being used is on disk).
- The operating system maintains a map (table) for each address space to determine the real location of data: the data can be on disk or in physical memory (at which address?).
- The process is unaware of where in the system (disk/memory) any particular portion of its address space is being held.

- Each running process generates addresses as if it has the entire machine to itself (as if the computer offers an extremely large and linear memory).

Main **advantages**:

- The job of the programmer and compiler is simplified, because no details of the hardware or memory organization are necessary to build a program.
- The program can be compiled independently from the properties of the physical memory.



8.1 Paged Mapping:

- To benefit from spatial locality and to transfer data using a DMAC (or IOP), data is moved between the disk and main memory in the form of pages.
- Virtual memory (address space) is divided into fixed-size *pages*.
- Main memory is divided into blocks (*page frames - PF*), each of which is equal in size to a page.

Page size: 2^r

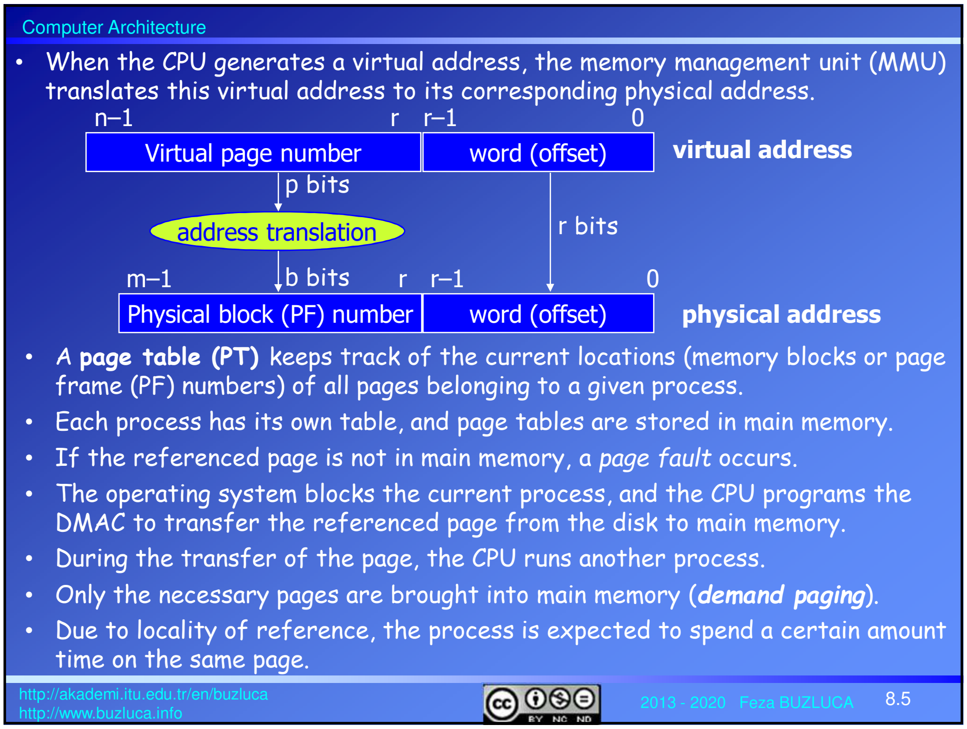
Address space: 2^n Number of pages: 2^p , $p = n - r$

Memory space: 2^m Number of block: 2^b , $b = m - r$

A page of virtual memory can be located in any block (page frame) of main memory.

Virtual memory: $2^{(n-r)} = 2^p$ pages.

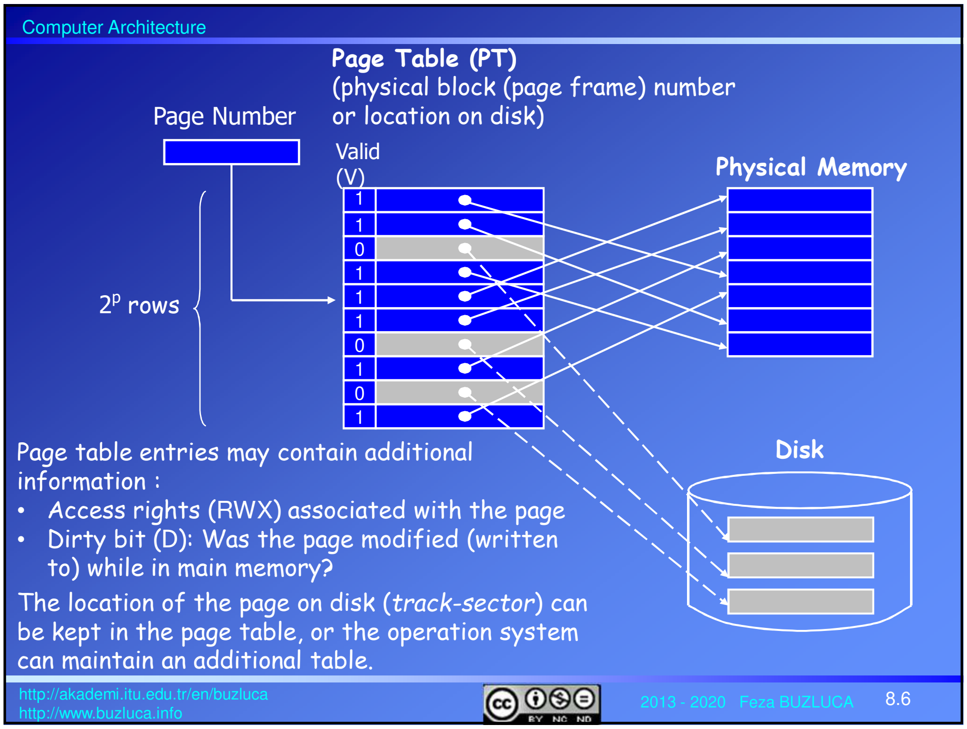
Main memory: $2^{(m-r)} = 2^b$ blocks (page frames)



- When the CPU generates a virtual address, the memory management unit (MMU) translates this virtual address to its corresponding physical address.

n–1
r
r–1
0
Virtual page number
word (offset)
virtual address
p bits
address translation
r bits
m–1
b bits
r
r–1
0
Physical block (PF) number
word (offset)
physical address

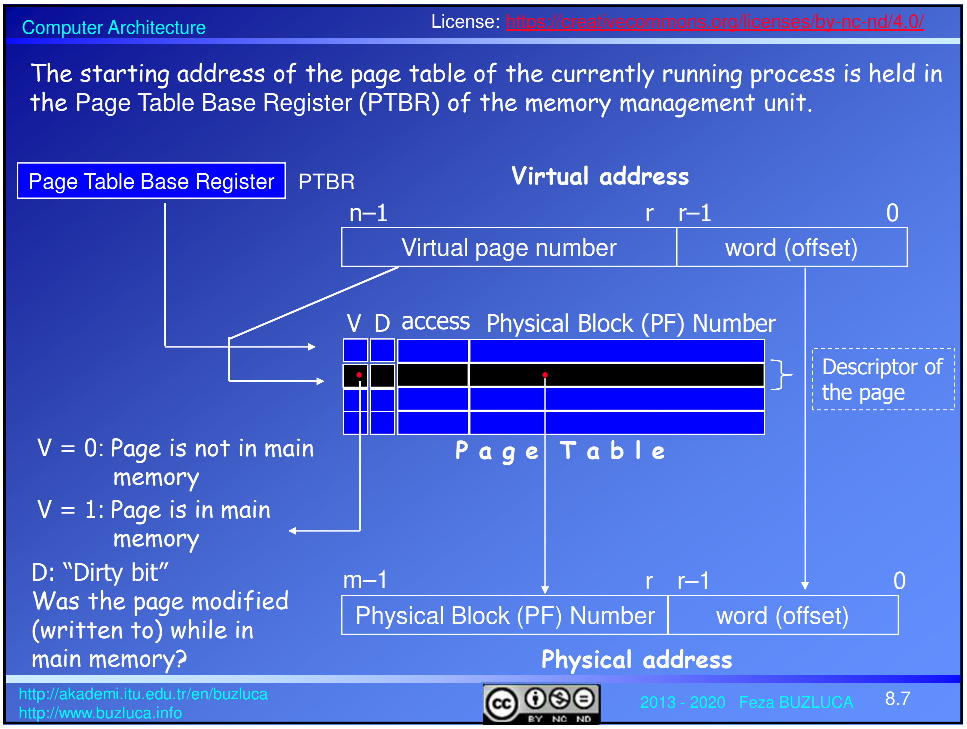
- A **page table (PT)** keeps track of the current locations (memory blocks or page frame (PF) numbers) of all pages belonging to a given process.
- Each process has its own table, and page tables are stored in main memory.
- If the referenced page is not in main memory, a *page fault* occurs.
- The operating system blocks the current process, and the CPU programs the DMAC to transfer the referenced page from the disk to main memory.
- During the transfer of the page, the CPU runs another process.
- Only the necessary pages are brought into main memory (***demand paging***).
- Due to locality of reference, the process is expected to spend a certain amount time on the same page.



Page table entries may contain additional information :

- Access rights (RWX) associated with the page
- Dirty bit (D): Was the page modified (written to) while in main memory?

The location of the page on disk (*track-sector*) can be kept in the page table, or the operation system can maintain an additional table.



The starting address of the page table of the currently running process is held in the Page Table Base Register (PTBR) of the memory management unit.

Page Table Base Register PTBR

Virtual address

n–1 r r–1 0

Virtual page number | word (offset)

V D access Physical Block (PF) Number

Descriptor of the page

P a g e T a b l e

V = 0: Page is not in main memory

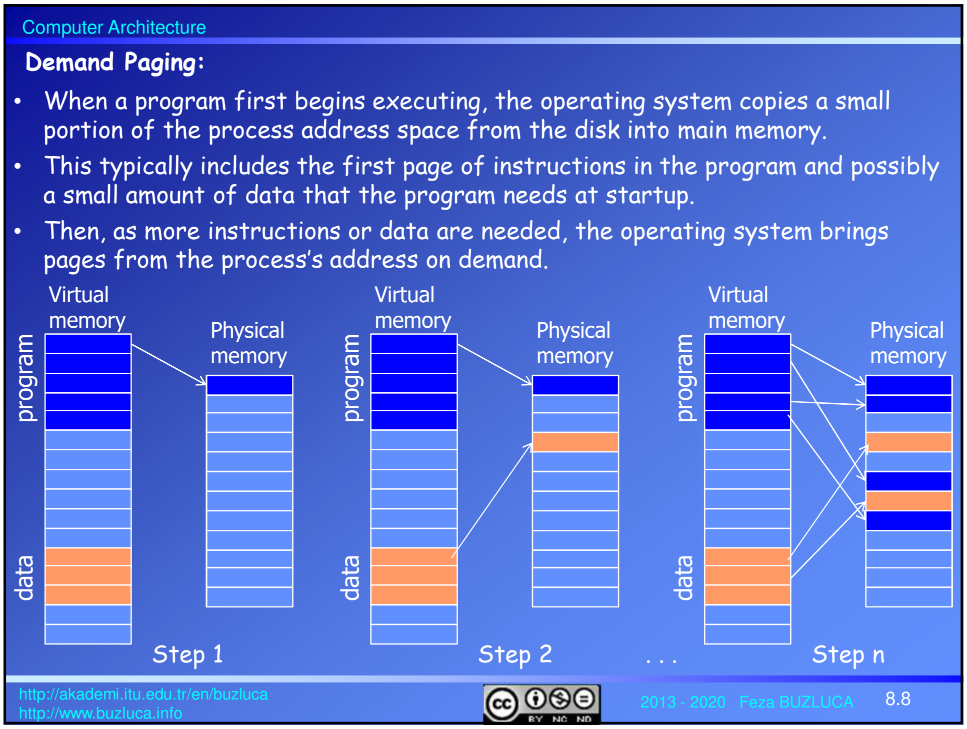
V = 1: Page is in main memory

D: "Dirty bit"
Was the page modified (written to) while in main memory?

m–1 r r–1 0

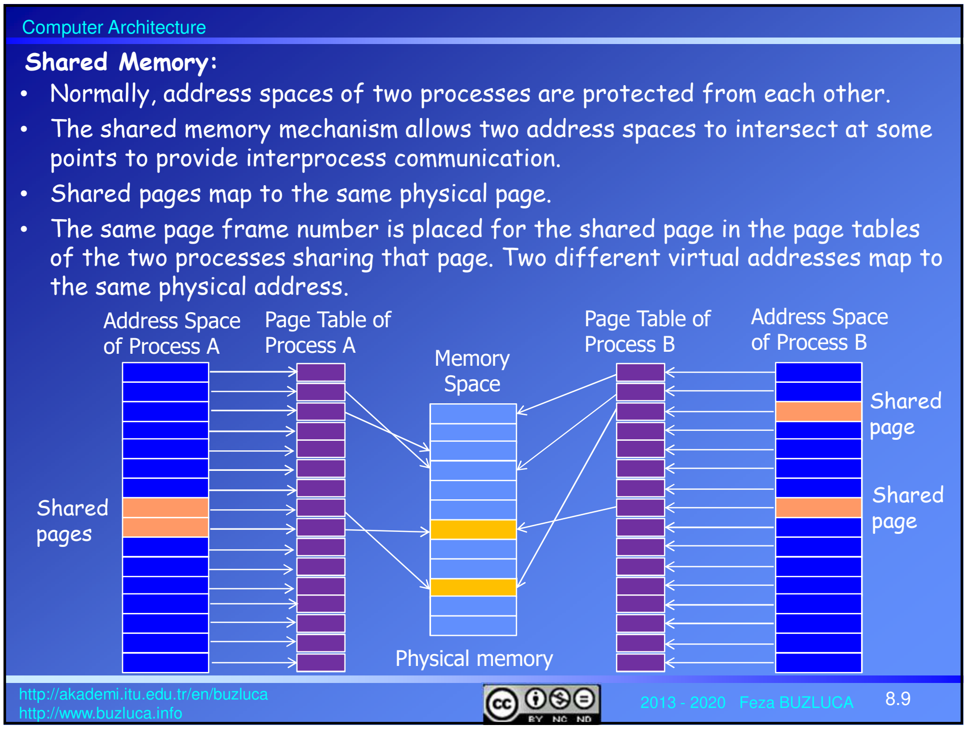
Physical Block (PF) Number | word (offset)

Physical address



Demand Paging:

- When a program first begins executing, the operating system copies a small portion of the process address space from the disk into main memory.
- This typically includes the first page of instructions in the program and possibly a small amount of data that the program needs at startup.
- Then, as more instructions or data are needed, the operating system brings pages from the process's address on demand.



Shared Memory:

- Normally, address spaces of two processes are protected from each other.
- The shared memory mechanism allows two address spaces to intersect at some points to provide interprocess communication.
- Shared pages map to the same physical page.
- The same page frame number is placed for the shared page in the page tables of the two processes sharing that page. Two different virtual addresses map to the same physical address.

Address Space of Process A

Page Table of Process A

Memory Space

Page Table of Process B

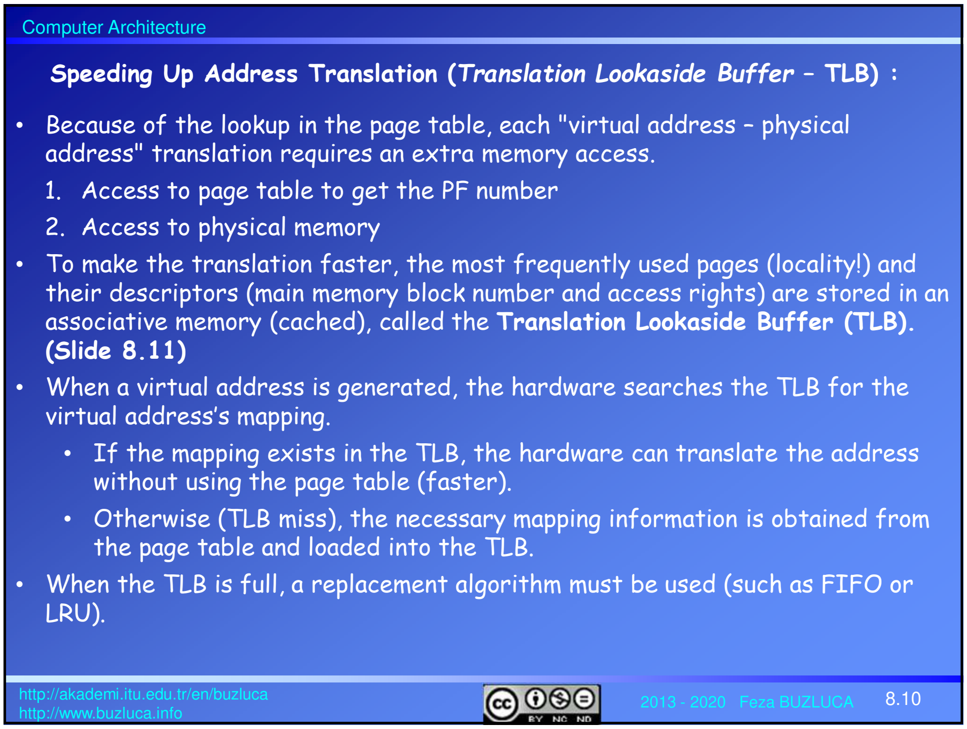
Address Space of Process B

Shared pages

Shared page

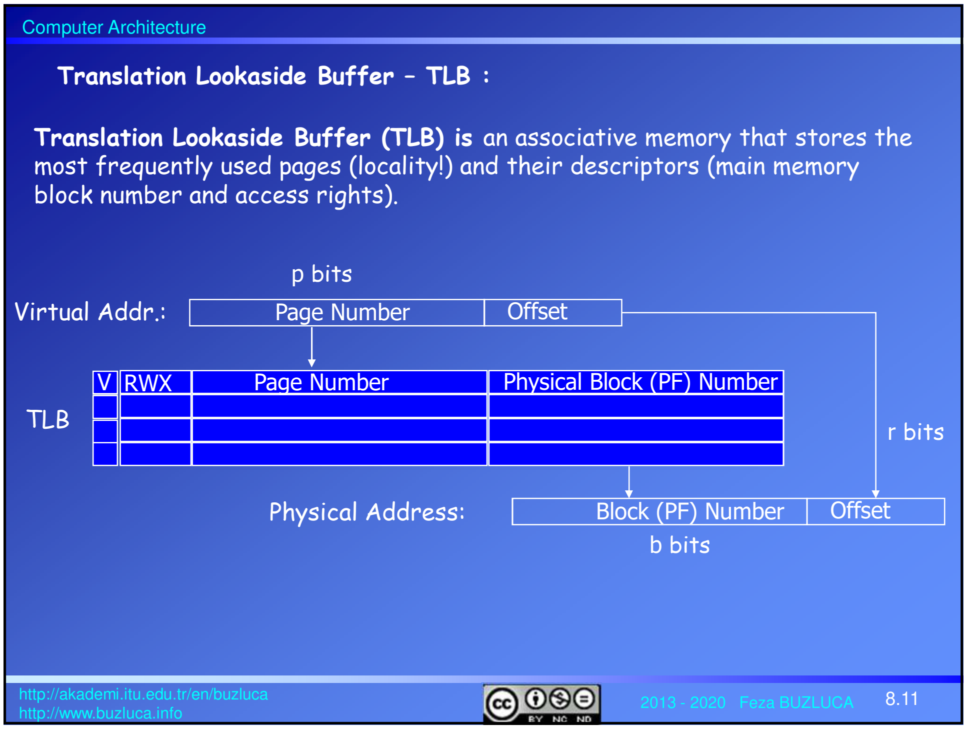
Shared page

Physical memory



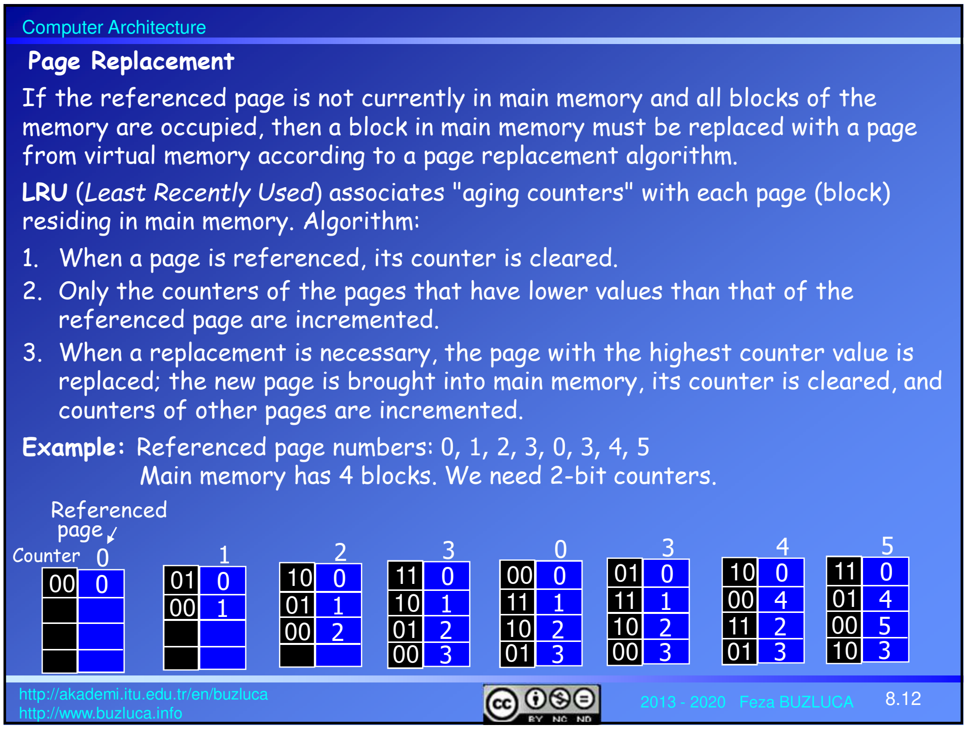
Speeding Up Address Translation (*Translation Lookaside Buffer* – TLB) :

- Because of the lookup in the page table, each "virtual address – physical address" translation requires an extra memory access.
  1. Access to page table to get the PF number
  2. Access to physical memory
- To make the translation faster, the most frequently used pages (locality!) and their descriptors (main memory block number and access rights) are stored in an associative memory (cached), called the **Translation Lookaside Buffer (TLB). (Slide 8.11)**
- When a virtual address is generated, the hardware searches the TLB for the virtual address's mapping.
  - If the mapping exists in the TLB, the hardware can translate the address without using the page table (faster).
  - Otherwise (TLB miss), the necessary mapping information is obtained from the page table and loaded into the TLB.
- When the TLB is full, a replacement algorithm must be used (such as FIFO or LRU).



Translation Lookaside Buffer – TLB :

Translation Lookaside Buffer (TLB) is an associative memory that stores the most frequently used pages (locality!) and their descriptors (main memory block number and access rights).



Page Replacement

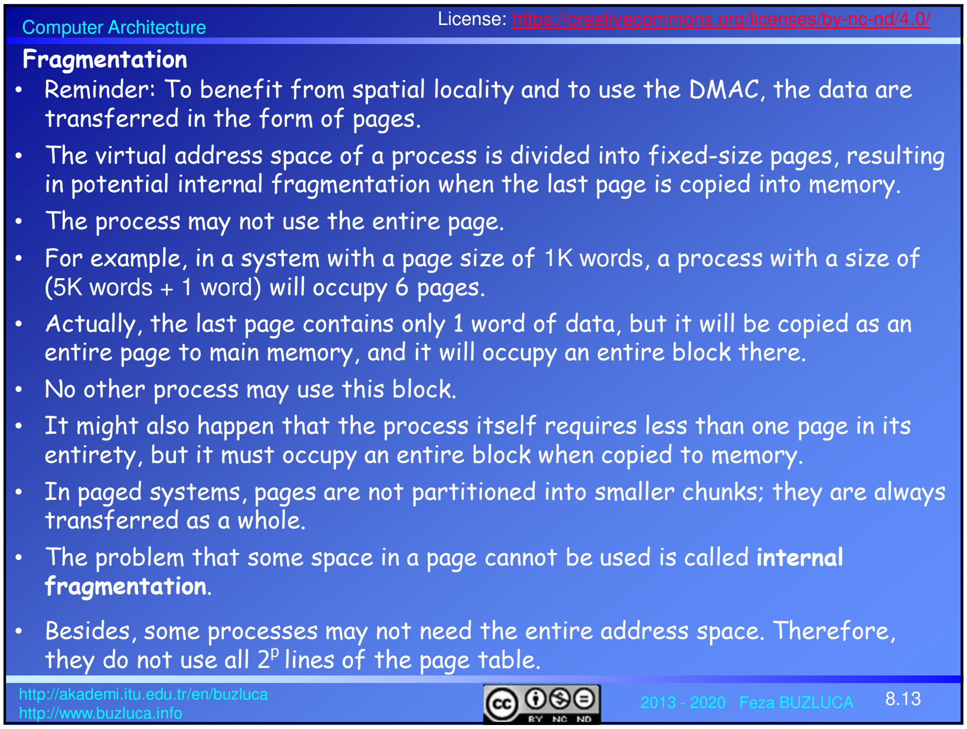
If the referenced page is not currently in main memory and all blocks of the memory are occupied, then a block in main memory must be replaced with a page from virtual memory according to a page replacement algorithm.

LRU (*Least Recently Used*) associates "aging counters" with each page (block) residing in main memory. Algorithm:

1. When a page is referenced, its counter is cleared.
2. Only the counters of the pages that have lower values than that of the referenced page are incremented.
3. When a replacement is necessary, the page with the highest counter value is replaced; the new page is brought into main memory, its counter is cleared, and counters of other pages are incremented.

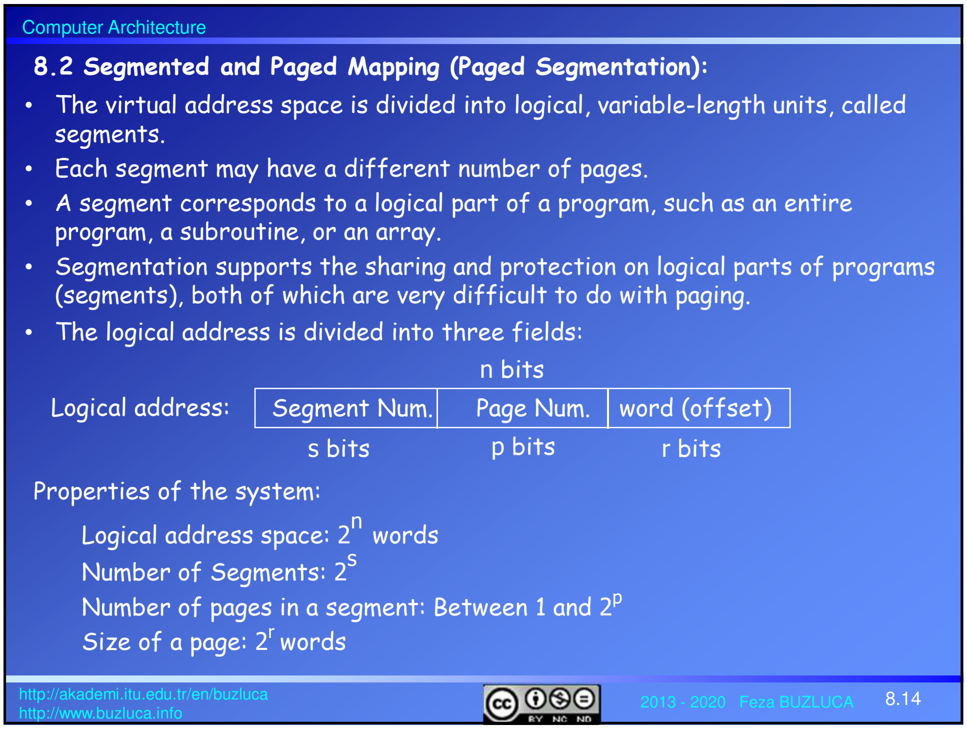
Example: Referenced page numbers: 0, 1, 2, 3, 0, 3, 4, 5
Main memory has 4 blocks. We need 2-bit counters.

Referenced page	0	1	2	3	0	3	4	5
Counter	00 0	01 0	10 0	11 0	00 0	01 0	10 0	11 0
		00 1	01 1	10 1	11 1	11 1	00 4	01 4
			00 2	01 2	10 2	10 2	11 2	00 5
				00 3	01 3	00 3	01 3	10 3



Fragmentation

- Reminder: To benefit from spatial locality and to use the DMAC, the data are transferred in the form of pages.
- The virtual address space of a process is divided into fixed-size pages, resulting in potential internal fragmentation when the last page is copied into memory.
- The process may not use the entire page.
- For example, in a system with a page size of 1K words, a process with a size of (5K words + 1 word) will occupy 6 pages.
- Actually, the last page contains only 1 word of data, but it will be copied as an entire page to main memory, and it will occupy an entire block there.
- No other process may use this block.
- It might also happen that the process itself requires less than one page in its entirety, but it must occupy an entire block when copied to memory.
- In paged systems, pages are not partitioned into smaller chunks; they are always transferred as a whole.
- The problem that some space in a page cannot be used is called **internal fragmentation**.
- Besides, some processes may not need the entire address space. Therefore, they do not use all 2^p lines of the page table.



8.2 Segmented and Paged Mapping (Paged Segmentation):

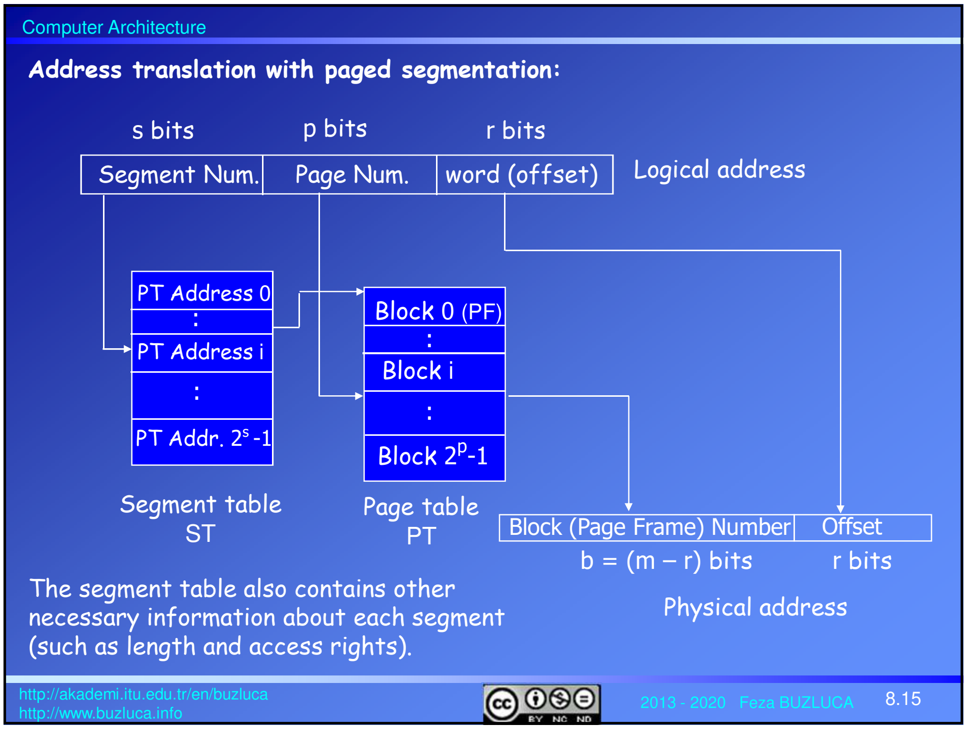
- The virtual address space is divided into logical, variable-length units, called segments.
- Each segment may have a different number of pages.
- A segment corresponds to a logical part of a program, such as an entire program, a subroutine, or an array.
- Segmentation supports the sharing and protection on logical parts of programs (segments), both of which are very difficult to do with paging.
- The logical address is divided into three fields:

n bits

Logical address:	Segment Num.	Page Num.	word (offset)
	s bits	p bits	r bits

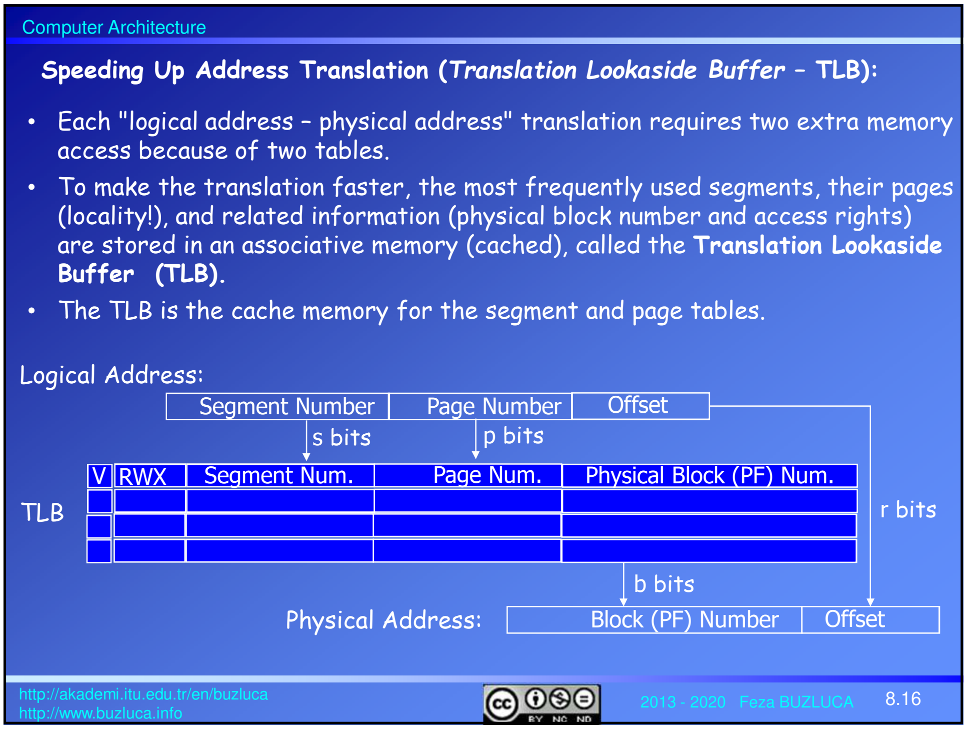
Properties of the system:

Logical address space: 2^n words
Number of Segments: 2^s
Number of pages in a segment: Between 1 and 2^p
Size of a page: 2^r words



Address translation with paged segmentation:

The segment table also contains other necessary information about each segment (such as length and access rights).



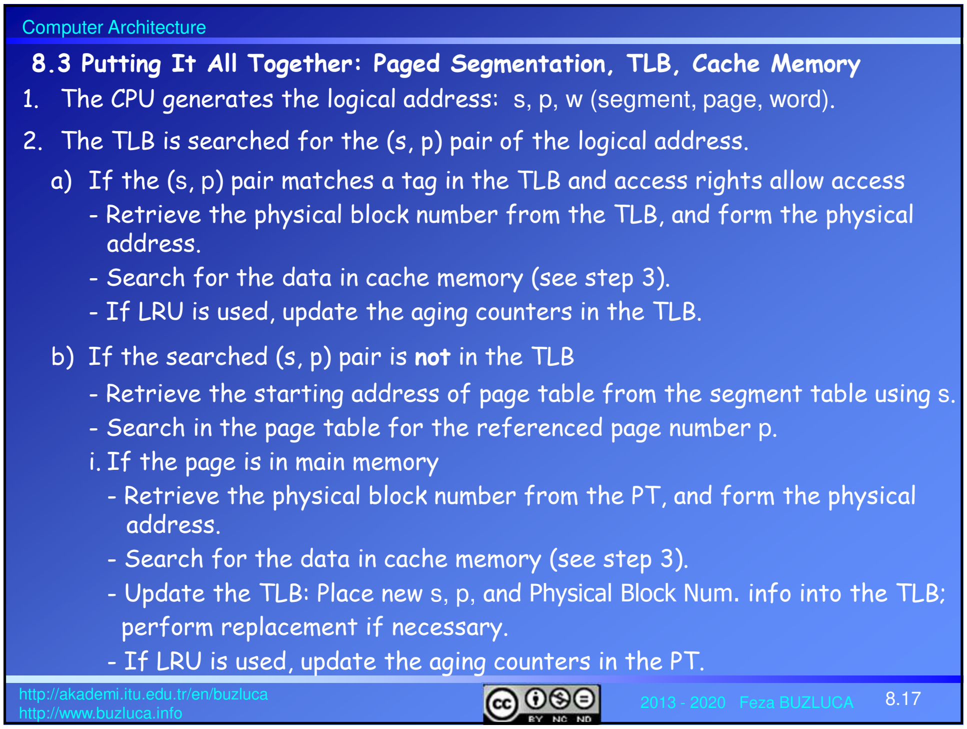
Speeding Up Address Translation (*Translation Lookaside Buffer* – TLB):

- Each "logical address – physical address" translation requires two extra memory access because of two tables.
- To make the translation faster, the most frequently used segments, their pages (locality!), and related information (physical block number and access rights) are stored in an associative memory (cached), called the **Translation Lookaside Buffer (TLB)**.
- The TLB is the cache memory for the segment and page tables.

Logical Address:

V	RWX	Segment Num.	Page Num.	Physical Block (PF) Num.

Physical Address:

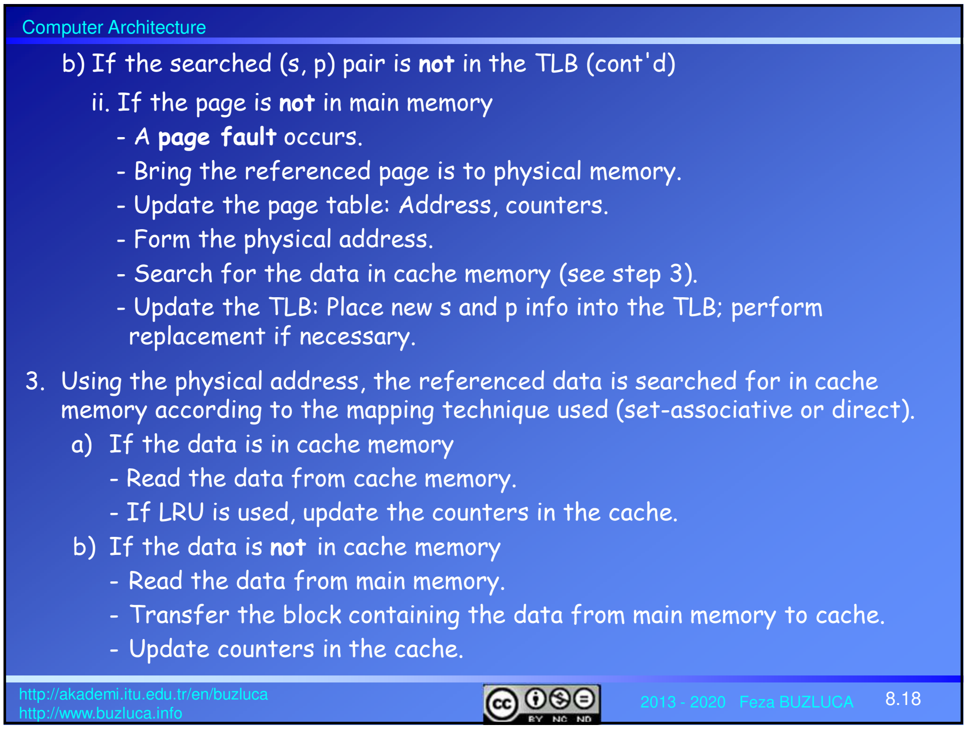


8.3 Putting It All Together: Paged Segmentation, TLB, Cache Memory

1. The CPU generates the logical address: s, p, w (segment, page, word).
2. The TLB is searched for the (s, p) pair of the logical address.
   a) If the (s, p) pair matches a tag in the TLB and access rights allow access
      - Retrieve the physical block number from the TLB, and form the physical address.
      - Search for the data in cache memory (see step 3).
      - If LRU is used, update the aging counters in the TLB.

   b) If the searched (s, p) pair is **not** in the TLB
      - Retrieve the starting address of page table from the segment table using s.
      - Search in the page table for the referenced page number p.

      i. If the page is in main memory
         - Retrieve the physical block number from the PT, and form the physical address.
         - Search for the data in cache memory (see step 3).
         - Update the TLB: Place new s, p, and Physical Block Num. info into the TLB; perform replacement if necessary.
         - If LRU is used, update the aging counters in the PT.



b) If the searched (s, p) pair is **not** in the TLB (cont'd)

ii. If the page is **not** in main memory
   - A **page fault** occurs.
   - Bring the referenced page is to physical memory.
   - Update the page table: Address, counters.
   - Form the physical address.
   - Search for the data in cache memory (see step 3).
   - Update the TLB: Place new s and p info into the TLB; perform replacement if necessary.

3. Using the physical address, the referenced data is searched for in cache memory according to the mapping technique used (set-associative or direct).
   a) If the data is in cache memory
      - Read the data from cache memory.
      - If LRU is used, update the counters in the cache.

   b) If the data is **not** in cache memory
      - Read the data from main memory.
      - Transfer the block containing the data from main memory to cache.
      - Update counters in the cache.