Fundamentos de los Sistemas Operativos (FSO)

Departamento de Informática de Sistemas y Computadoras (DISCA)
Universitat Politècnica de València

Part 3: Memory management

Unit 9

Virtual memory (I)

fSO

DISCA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Goals and bibliography

- **Goals**
  - To know the **advantages** of virtual memory and its effects on **system performance**
  - To understand **demand paging concept**
  - To know the **page replacement techniques**

Bibliography

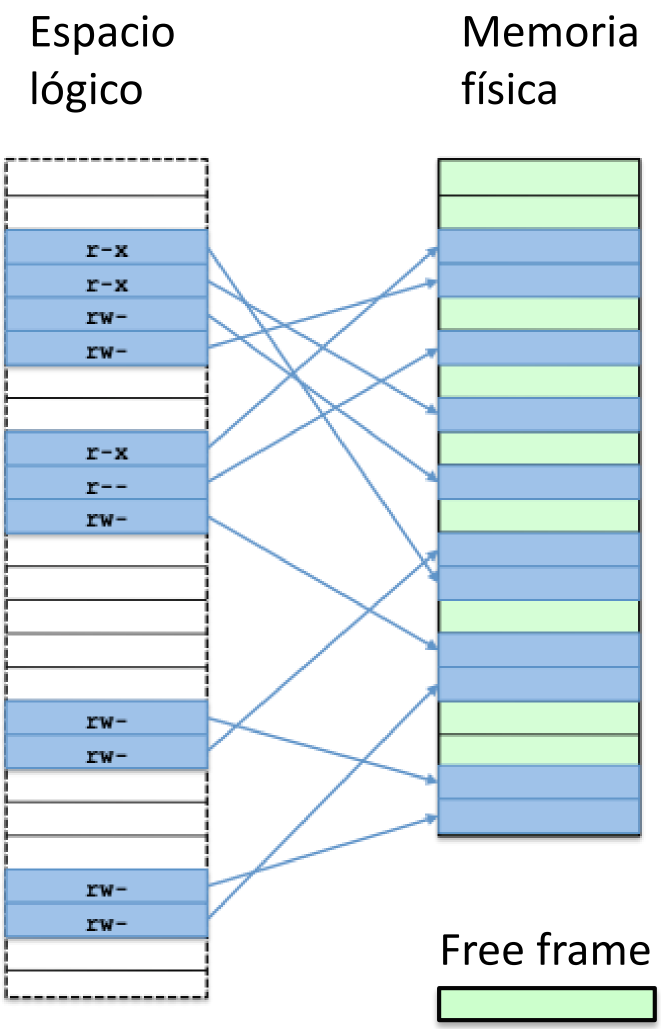
- Silberschatz, chapter 9

Content

- **Virtual memory concept**
- Virtual memory support
- Demand paging
- Virtual Memory and process management
- FIFO replacement algorithm
- Optimal replacement algorithm
- LRU replacement algorithm

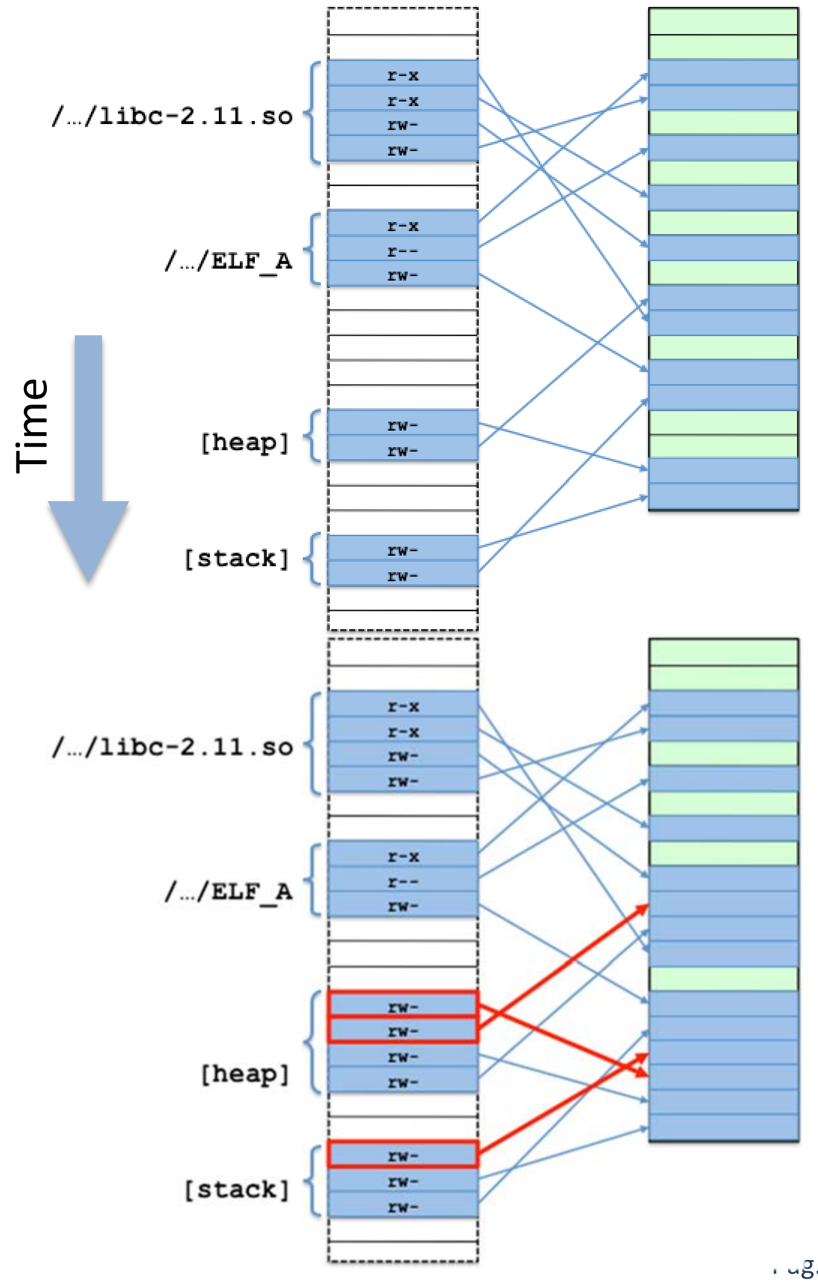
Virtual memory concept

- Paging without virtual memory
  - The OS reserves **all the physical memory** required by a process to start it
  - Al memory accesses take the same amount of time
  - Whe a process end the OS releases all the frames used by the process

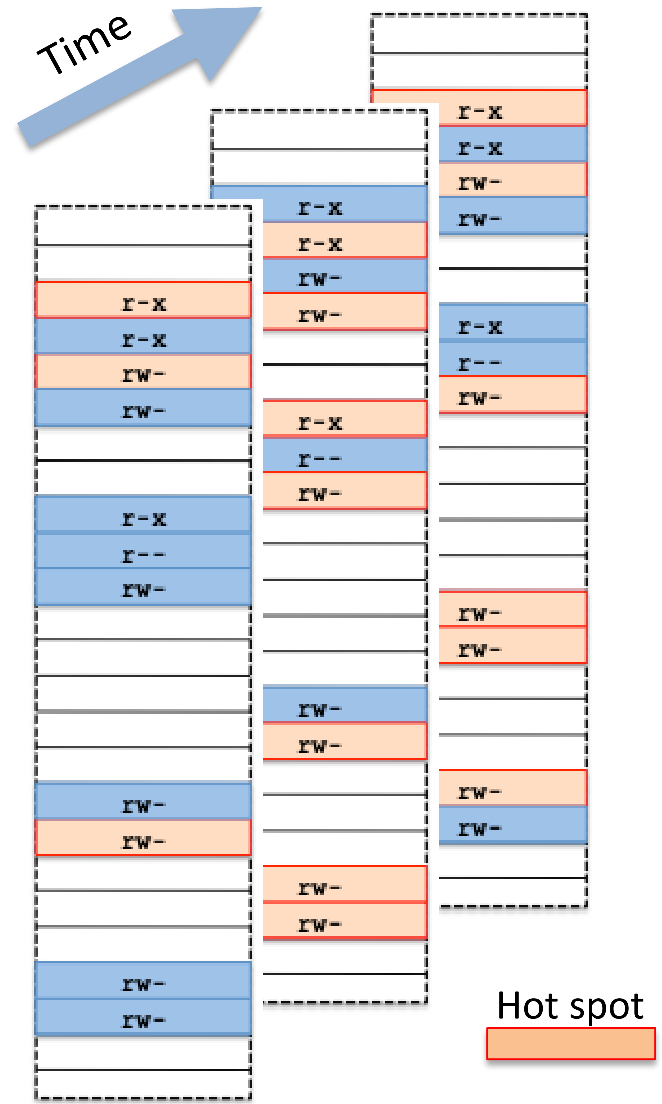


Virtual memory concept

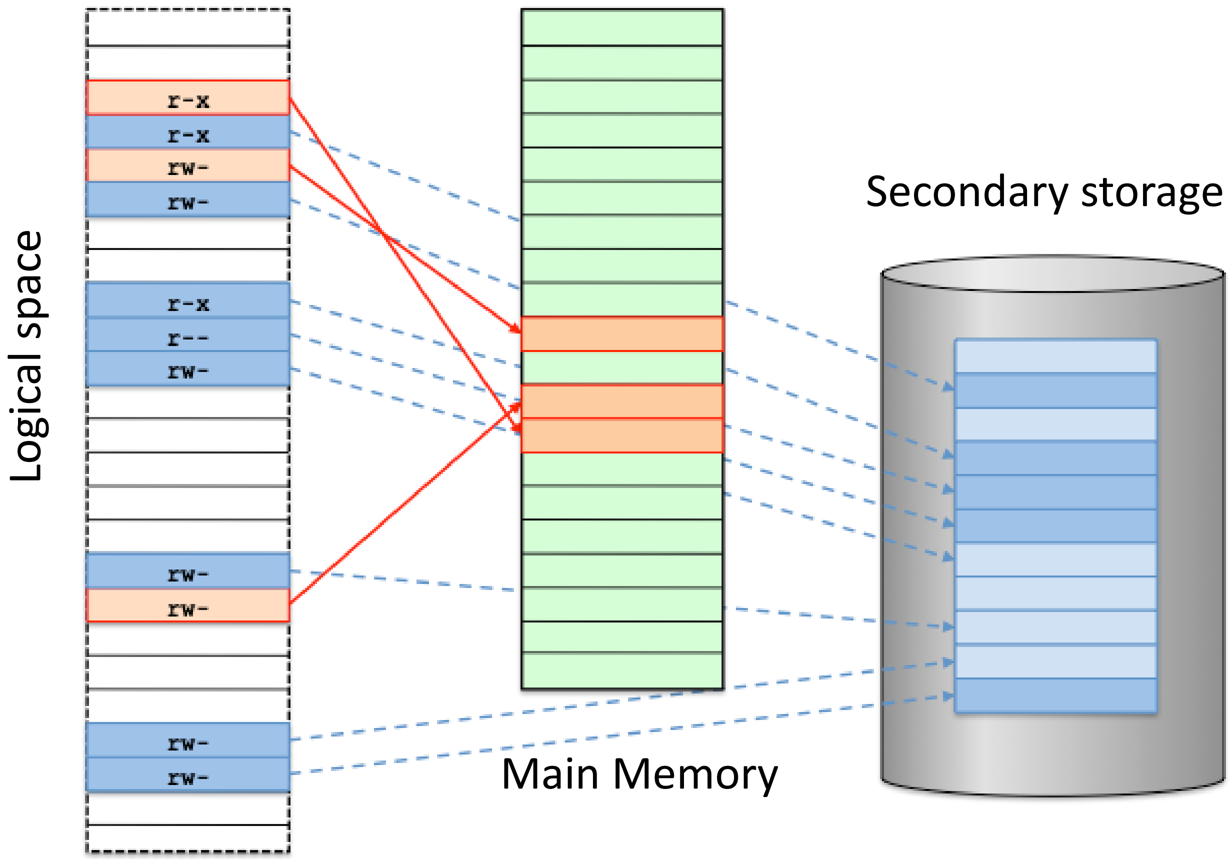
- Memory requirements change due to dynamic regions
  - The stack grows due to function calls
  - Dynamic memory allocation creates the *heap* region that can grow and shrink due to *malloc* and *free* calls
  - Creating new threads produce new thread stack regions



- **Reference locality principle**
  - Independently of the logical size of a process its memory accesses are localted into **hot spots**
    - Given a time interval a small set of instructions is accessed particularly inner loops, something similar happens with data access
    - Along the process lifetime hot spots of code and data move



- **Memory management based on virtual memory**
  - The OS manages memory allocation to processes in such a way that **only their hot spots are allocated on physical memory**
  - The **remaining logical space** content is allocated on **secondary storage** (swaping area)



Virtual memory concept

- **Virtual memory base tecnologies**
  - It combines physical memory (RAM) with secondary storage (hard disk or SSD)
  - Main memory is made of words (i.e. 32-bit, 64-bit) addresable by the CPU, access time is a couple of nanoseconds
    - Every instruction cycle performs one or more accesses to main memory
  - Secondary storage is made of blocks (i.e. 512-byte, 4096-byte), access time is a couple of miliseconds ($1\ ms = 10^6\ ns$)
    - A page transfer is made in one sigle I/O operation that requires the execution of several instructions by the CPU

Virtual memory concept

- **Virtual memory scheme**
  – The OS manages memory following a sparse allocation approach, typicaly paging
  – A swap area is available on disk, as a partition or as a system file
  – For every page in use by a process it can be in two states:
    - **Valid**: page allocated into a main memory frame
    - **Invalid**: page not allocated in main memory but on the swap area
  – For every memory access it can happen a:
    - **Hit** (most common): reference to a valid page
    - **Fault**: reference to an invalid page -> the page table has to be updated and one or more pages are transfered between main memory and the swap area

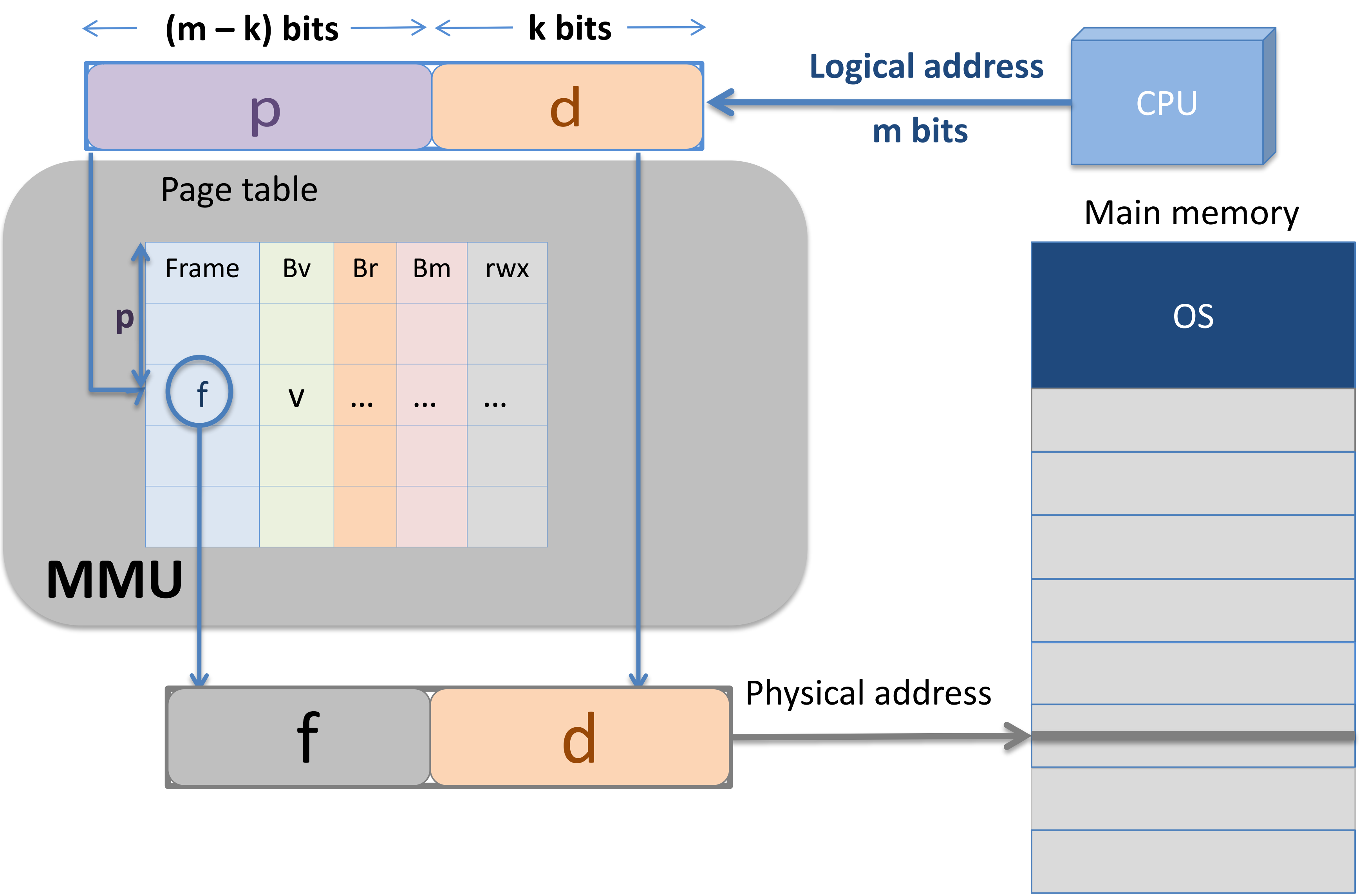
Virtual memory concept

- Benefits:
  - The same as paging: **page sharing and protection**
  - **It saves memory** and increases the multiprogramming level
  - It **allows bigger executable process size**
  - It **easies dynamic memory** management
- Penalties:
  - **Turnaround time** can increase due to page faults
  - **Workload of secondary storage** increases
  - Greater OS design **complexity**

Content

- Virtual memory concept
- **Virtual memory support**
- Demand paging
- Virtual Memory and process management
- FIFO replacement algorithm
- Optimal replacement algorithm
- LRU replacement algorithm

- Address translation as paging (UT08)



Virtual memory support

- It is every page table entry
- Every page descriptor has the following fields:
  - **Frame number** where the page is allocated in physical memory
  - **Valid bit**: it indicates if a page is mapped in memory, it supports demand paging
  - **Reference bit**: it indicates page access done. It is required for a second opportunity algorithm
  - **Modified bit**: it indicates page write access done. Trouble with shared pages.
  - **Access mode bits:** read only, read-write, execution, ...

Page descriptor

Frame number	Valid bit	Reference bit	Modified bit	Access mode bits rwx

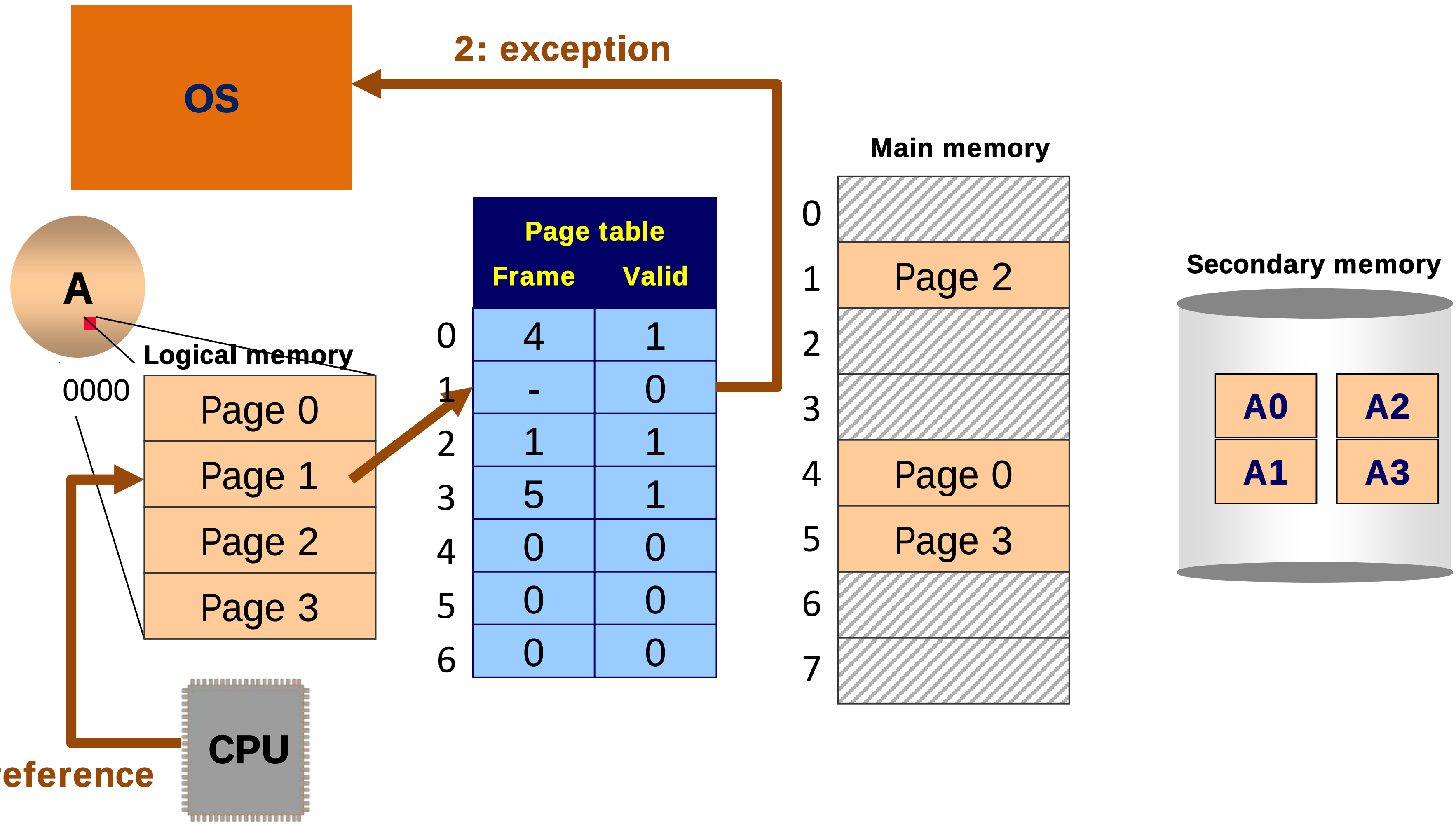
- **Page fault -> MMU exception**
  - It happens when **a non memory allocated page is referenced**, that is **with valid bit=0**
  - Page fault cases:
    - **Page on disk**: The reference page belongs to the process → it is allocated into a memory frame
    - **Access error**: The reference page doesn't belong to the process and cannot be assigned to it → the process is aborted
    - **Process growing**: The process asks for new pages, if the OS permits it, a new page is assigned to the process with its valid bit set and the new page is allocated into a memory frame

Content

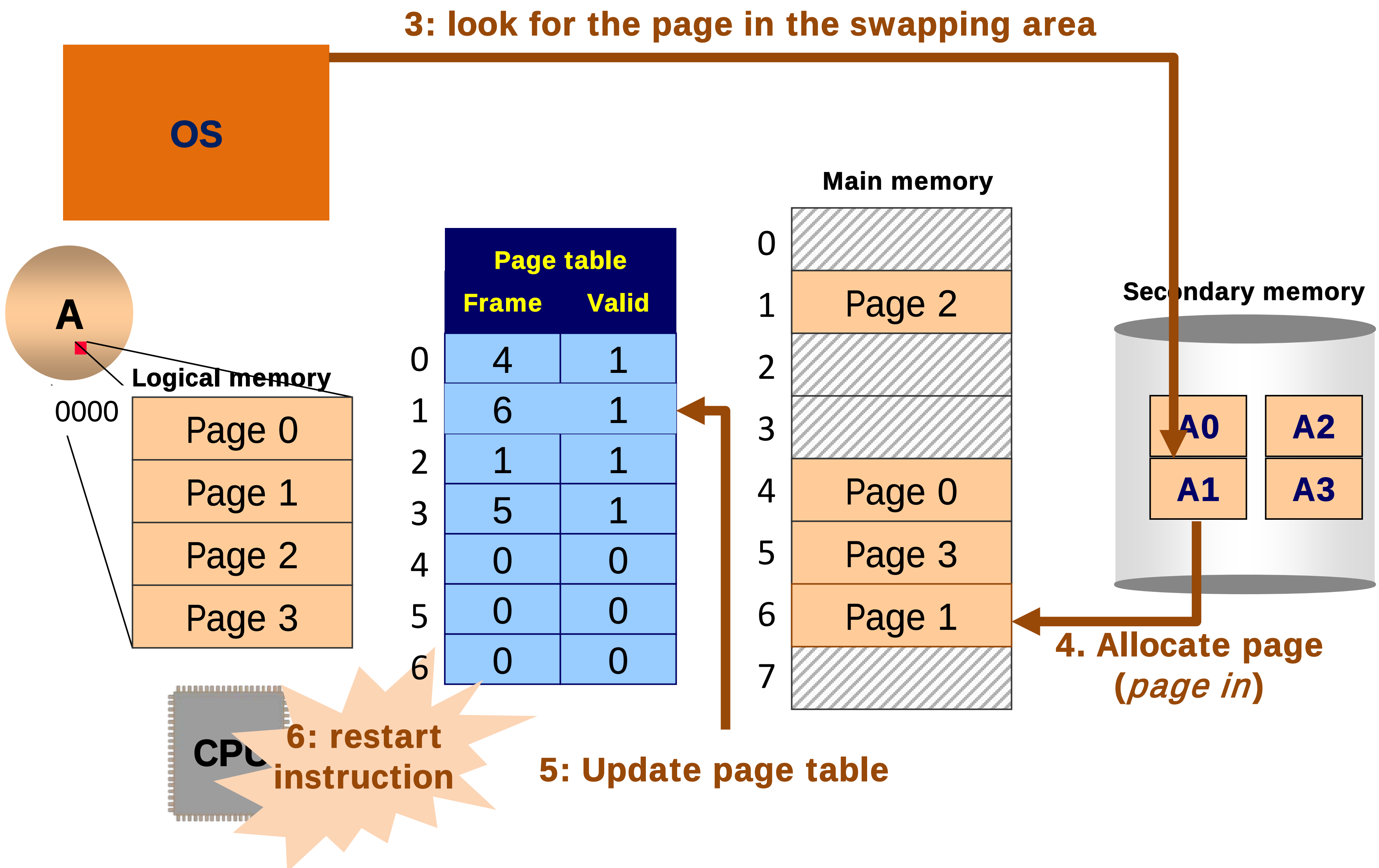
- Virtual memory concept
- Virtual memory support
- **Demand paging**
- Virtual Memory and process management
- FIFO replacement algorithm
- Optimal replacement algorithm
- LRU replacement algorithm

Demand paging

- Page fault: Page on disk case



- Page fault: Page on disk case



- **Page fault algorithm: page on disk case**
  - Find demanded page on disk
  - Find a free frame:
    - If there is a free frame, then use it
    - If there are no free frames apply **page replacement algorithm** (next slide)
  - **Read demanded page on disk** (page in) and allocate it into the free frame
  - **Update page table** of the process that generates the page fault
  - Update free frame table
  - Transfer control to user process
    - Restart the instruction that has produced the page fault

Demand paging

- **Page replacement:** It is required when main memory is full and there is a page fault
  - **Select an allocated page in main memory**, named **victim**, to **leave its frame**
    - There are several victim selection algorithms
  - If the victim page has its modified bit equal to 1 then **save the victim page to disk (page out)**
  - The victim page entry on the page table is updated with valid bit = 0
  - Update the free frame table

Demand paging

- **Reference string:** Sequence of accessed pages along a certain time period
  - From every logical address sent by the processor its page number is obtained
  - Repetitions are removed: several consecutive references to the same page are replaced by a single reference
- Example:
  - Page size = 1000 words
  - Referenced logical addresses from process P

    2500, 5100, 5234, 1800, 1432, 4388, 2124, 8216, 8498
  - Referenced pages

    2, 5, 5, 1, 1, 4 , 2 , 8, 8
  - The corresponding reference string is:

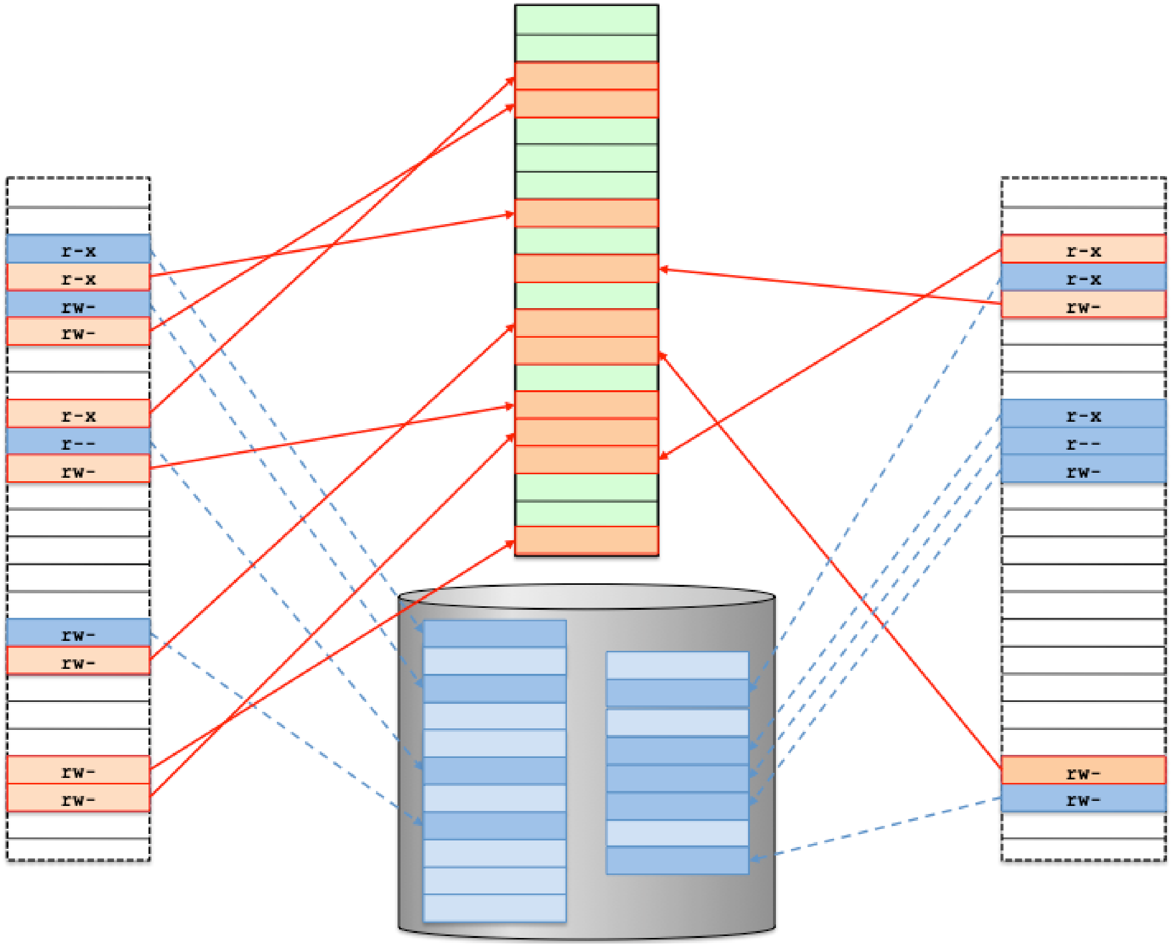
    2, 5, 1, 4, 2, 8

Content

- Virtual memory concept
- Virtual memory support
- Demand paging
- **Virtual Memory and process management**
- FIFO replacement algorithm
- Optimal replacement algorithm
- LRU replacement algorithm

- Virtual memory **favors concurrency**
  - It keeps in memory only processes hot spot areas, so memory capacity is optimized.

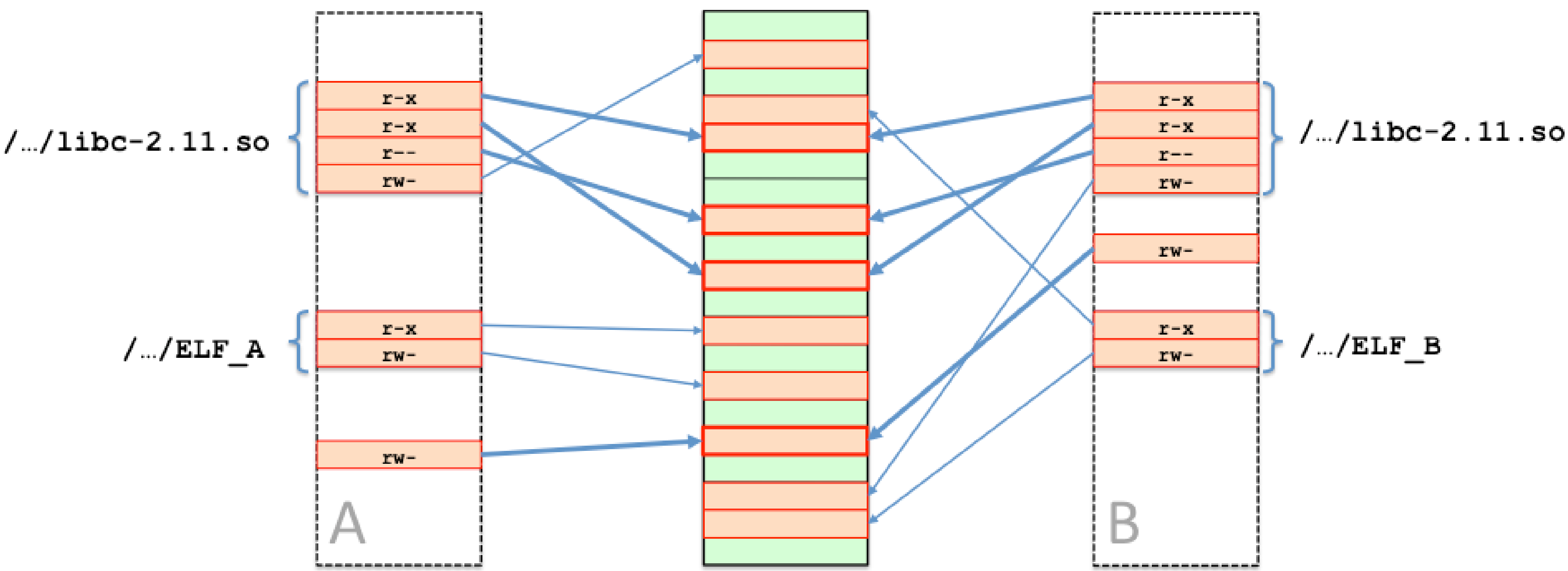
Example. Process A has 11 pages and B has 8 pages, but actually they only use 10 frames.



- Virtual memory allows **frame sharing**
  – Executable code pages from common dynamic libreries are shared between processes, as well as shared variables and mapped files

Example. Processes A and B, created from executable files ELF_A and ELF_B, share four allocated pages:

  – Three pages from library libc-2.11.so
  – Un page without support with shared variables



- **fork** and **exec** calls
  - An *exec* call invalidates all process pages
    - Coming page faults will update page descriptors with the new executable code
  - A *fork* call clones the parent's page descriptor table into the new process
    - Parent and child access read only pages (executable code and constants) on shared frames
    - Pages with writing permissions (i.e. variables, stack and heap) will be allocated on different frames, following *copy-on-write* policy: the new frame allocation is done when the first writing operation happens

Content

- Virtual memory concept
- Virtual memory support
- Demand paging
- Virtual Memory and process management
- **FIFO replacement algorithm**
- Optimal replacement algorithm
- LRU replacement algorithm

FIFO replacement algorithm

- Victim page: the one that is longer loaded in memory
- It suffers from Belady's anomaly

Frame		1	2	3	4	1	2	5	1	2	3	4	5
0		1	1	1	4	4	4	5	5	5	5	5	5
1			2	2	2	1	1	1	1	1	3	3	3
2				3	3	3	2	2	2	2	2	4	4

With 3 frames: 9 faults (6 faults with replacement)

marco		1	2	3	4	1	2	5	1	2	3	4	5
0		1	1	1	1	1	1	5	5	5	5	4	4
1			2	2	2	2	2	2	1	1	1	1	5
2				3	3	3	3	3	3	2	2	2	2
3					4	4	4	4	4	4	3	3	3

With 4 frames: 10 faults (6 faults with replacement)

FIFO replacement algorithm

- Victim page: the one that is longer loaded in memory
- It suffers from Belady's anomaly

Belady's anomaly

Frame		1	2	3	4	1	2	5	1	2	3	4	5
0		1	1	1	4	4	4	5	5	5	5	5	5
1			2	2	2	1	1	1	1	1	3	3	3
2				3	3	3	2	2	2	2	2	4	4

With 3 frames: 9 faults (6 faults with replacement)

Frame		1	2	3	4	1	2	5	1	2	3	4	5
0		1	1	1	1	1	1	5	5	5	5	4	4
1			2	2	2	2	2	2	1	1	1	1	5
2				3	3	3	3	3	3	2	2	2	2
3					4	4	4	4	4	4	3	3	3

With 4 frames: 10 faults (6 faults with replacement)

Optimal replacement algorithm

- **Stack algorithms**
  - A page set kept into N frames is **always** a subset of the one kept into N+1
  - Do not suffer from Belady’s anomaly

Content

- Virtual memory concept
- Virtual memory support
- Demand paging
- Virtual Memory and process management
- FIFO replacement algorithm
- **Optimal replacement algorithm**
- LRU replacement algorithm

Optimal replacement algorithm

- Victim page: the one which take longer to be referenced
- Minimum number of faults -> impossible to implement (future is unknown)

Frame		1	2	3	4	1	2	5	1	2	3	4	5
0		1	1	1	1	1	1	1	1	1	3	3	3
1			2	2	2	2	2	2	2	2	2	4	4
2				3	4	4	4	5	5	5	5	5	5

With 3 frames: 7 faults (4 faults with replacement)

Frame		1	2	3	4	1	2	5	1	2	3	4	5
0		1	1	1	1	1	1	1	1	1	1	4	4
1			2	2	2	2	2	2	2	2	2	2	2
2				3	3	3	3	3	3	3	3	3	3
3					4	4	4	5	5	5	5	5	5

With 4 frames: 6 faults (2 faults with replacement)

Content

- Virtual memory concept
- Virtual memory support
- Demand paging
- FIFO replacement algorithm
- Optimal replacement algorithm
- **LRU replacement algorithm**

Content

- Virtual memory concept
- Virtual memory support
- Demand paging
- Virtual Memory and process management
- FIFO replacement algorithm
- Optimal replacement algorithm
- **LRU replacement algorithm**

LRU replacement algorithm

- Victim page: the one that lasted more without being referenced or **least recently used**
- It is a stack algorithm

Frame		1	2	3	4	1	2	5	1	2	3	4	5
0		1	1	1	4	4	4	5	5	5	3	3	3
1			2	2	2	1	1	1	1	1	1	4	4
2				3	3	3	2	2	2	2	2	2	5

With 3 frames: 10 faults (7 faults with replacement)

Frame		1	2	3	4	1	2	5	1	2	3	4	5
0		1	1	1	1	1	1	1	1	1	1	1	5
1			2	2	2	2	2	2	2	2	2	2	2
2				3	3	3	3	5	5	5	5	4	4
3					4	4	4	4	4	4	3	3	3

With 4 frames: 8 faults (4 faults with replacement)

LRU replacement algorithm

- LRU implementations
  - Using counters:
    - Every page has an associated counter
  - Using an ordered list of referenced pages
    - When a page is referenced it is put at the end
    - The victim page is the first one
- Performance analysis
  - Advantages
    - Good approach to the optimal algorithm
  - Disadvantages
    - Costly implementation –> requires **hardware support**
  - Solution
    - LRU approximation algorithms -> based on **reference bit**