

Fundamentos de los Sistemas Operativos (FSO)

Departamento de Informática de Sistemas y Computadoras (DISCA)

Universitat Politècnica de València

Part 4: Memory management

Unit 10

Sparse memory allocation

fSO

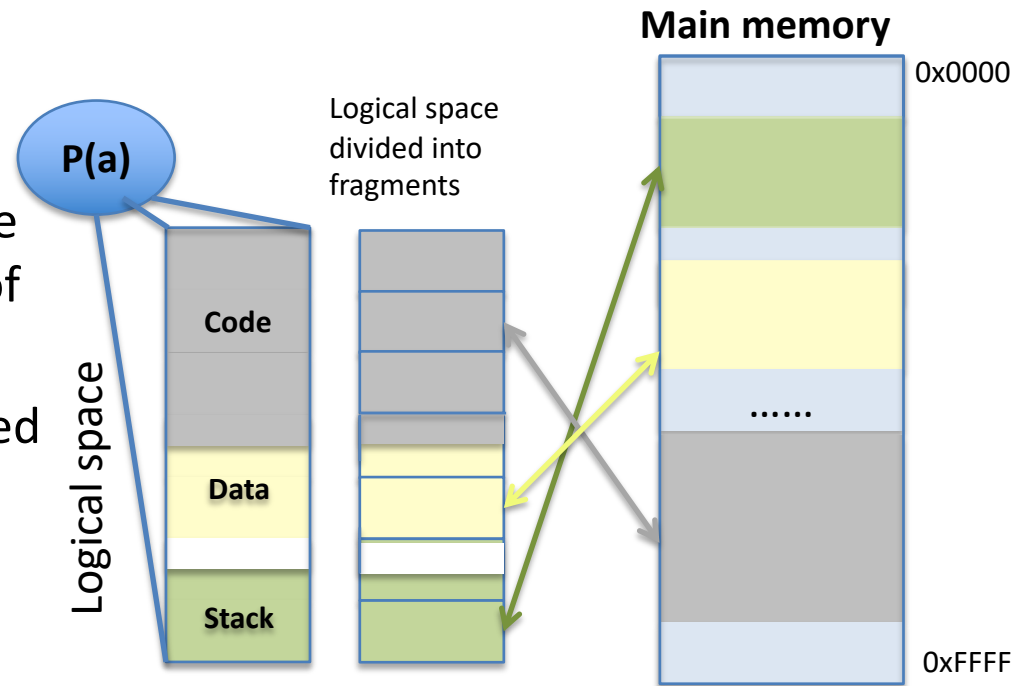
DISCA

- Goals
 - To understand sparse memory allocation approach
 - To know the basic sparse memory allocation techniques:
 - **Paging**
 - **Segmentation**
 - To know the combined techniques:
 - **Multilevel paging**
- Bibliography
 - Silbershatz, chapter 8

- **Sparse memory allocation concept**
- Paging
- Segmentation
- Multilevel paging

- **Sparse allocation**

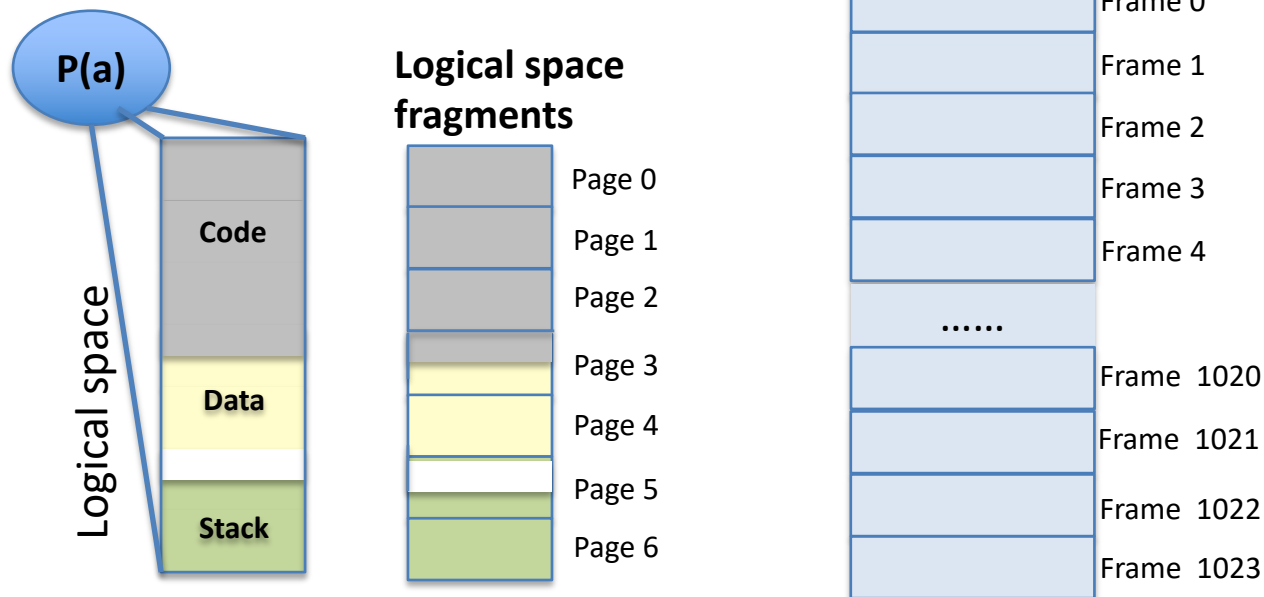
- The logical memory space of a process is made up of fragments
- Every fragment is allocated into physical memory independently



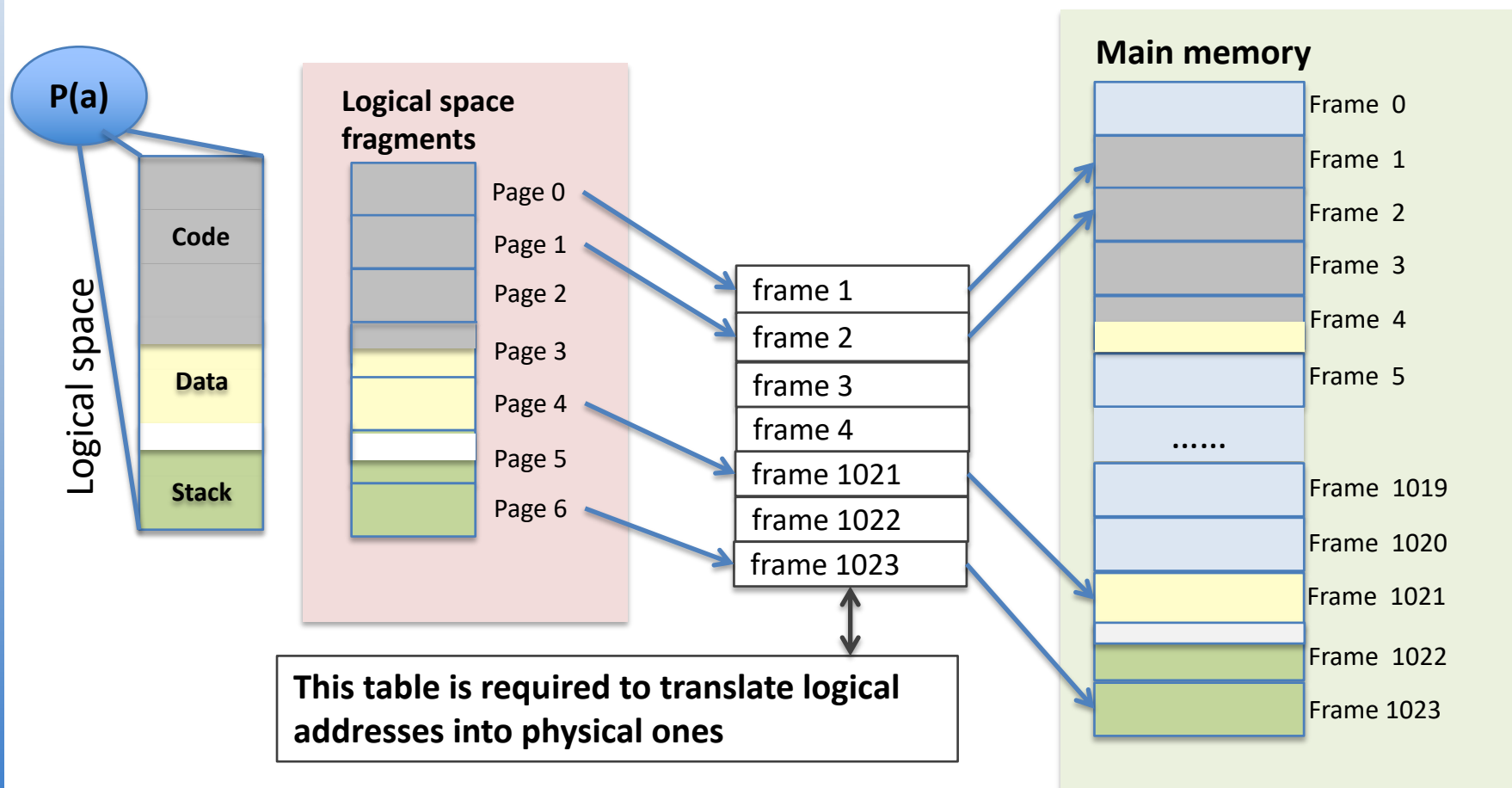
- The mapped physical memory space of a process doesn't need to be contiguous
 - **Paging**: fragments of fixed size
 - **Segmentation**: fragments of variable size
- The MMU needs to know for every segment its location and size
 - **Page table**
 - **Segment table**

- Sparse memory allocation concept
- **Paging**
- Segmentation
- Multilevel paging

- Allows non-contiguous allocation of process logical memory space
- It is based on considering both logical and physical spaces divided into **fixed size fragments**
 - **Pages** in case of **logical space**
 - **Frames** in case of **physical memory**



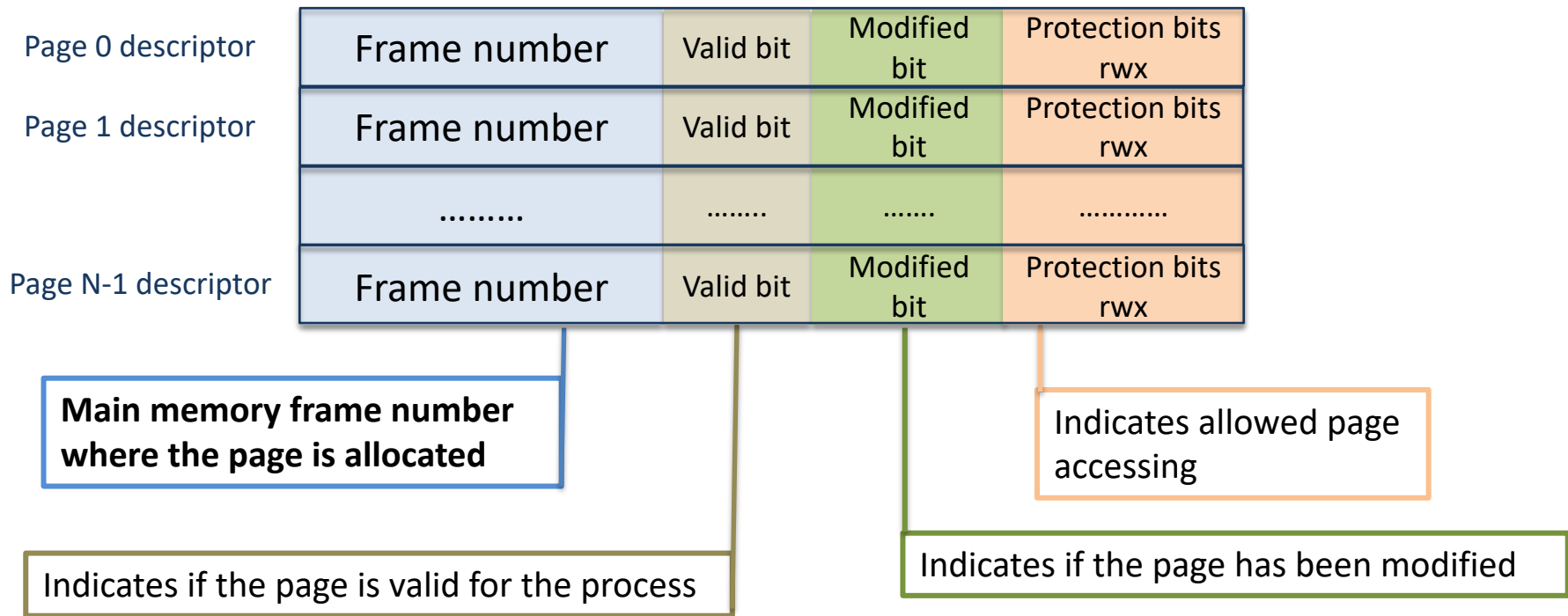
- When a process enters the system the OS loads all its pages in physical memory frames
 - Every page fits a frame
 - A table is built to store the frame number where every process page is allocated
 - Every process has its page table



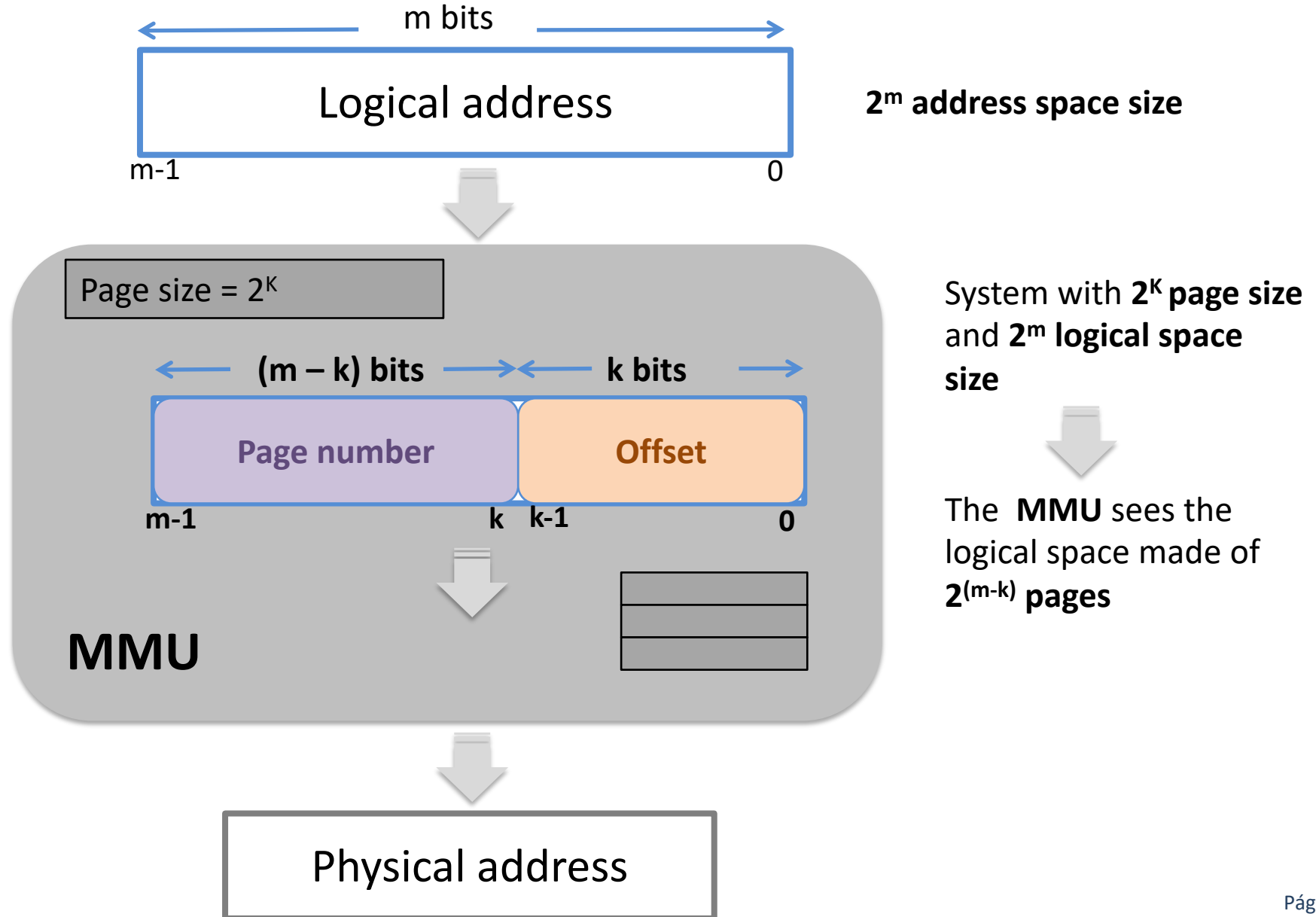
- **Page table**

- Every table entry is called **page descriptor** and contains:

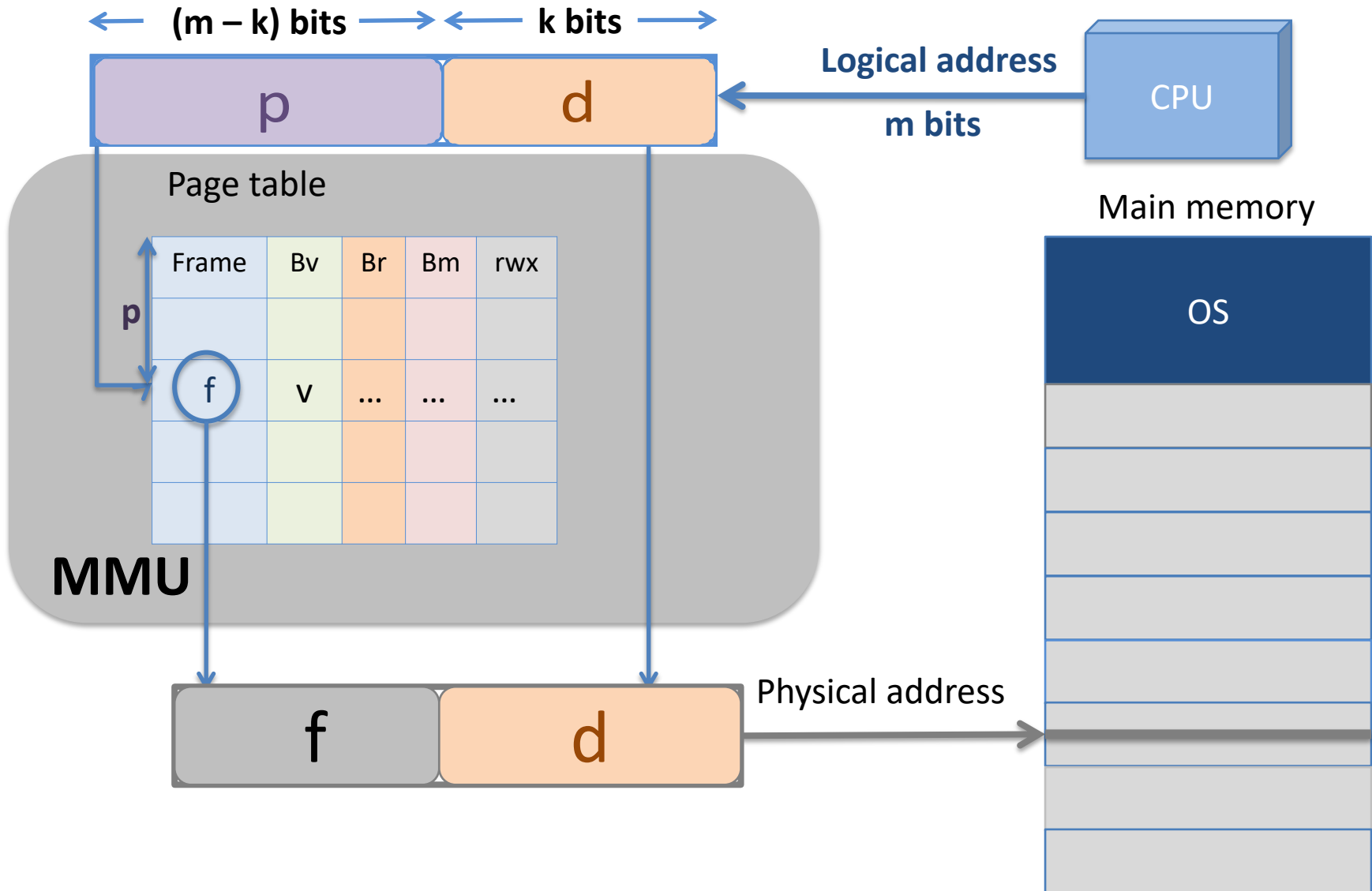
- The page allocated **frame number**
- A set of control bits: **valid bit**, **modified bit** and **protection bits**



- Logical address structure

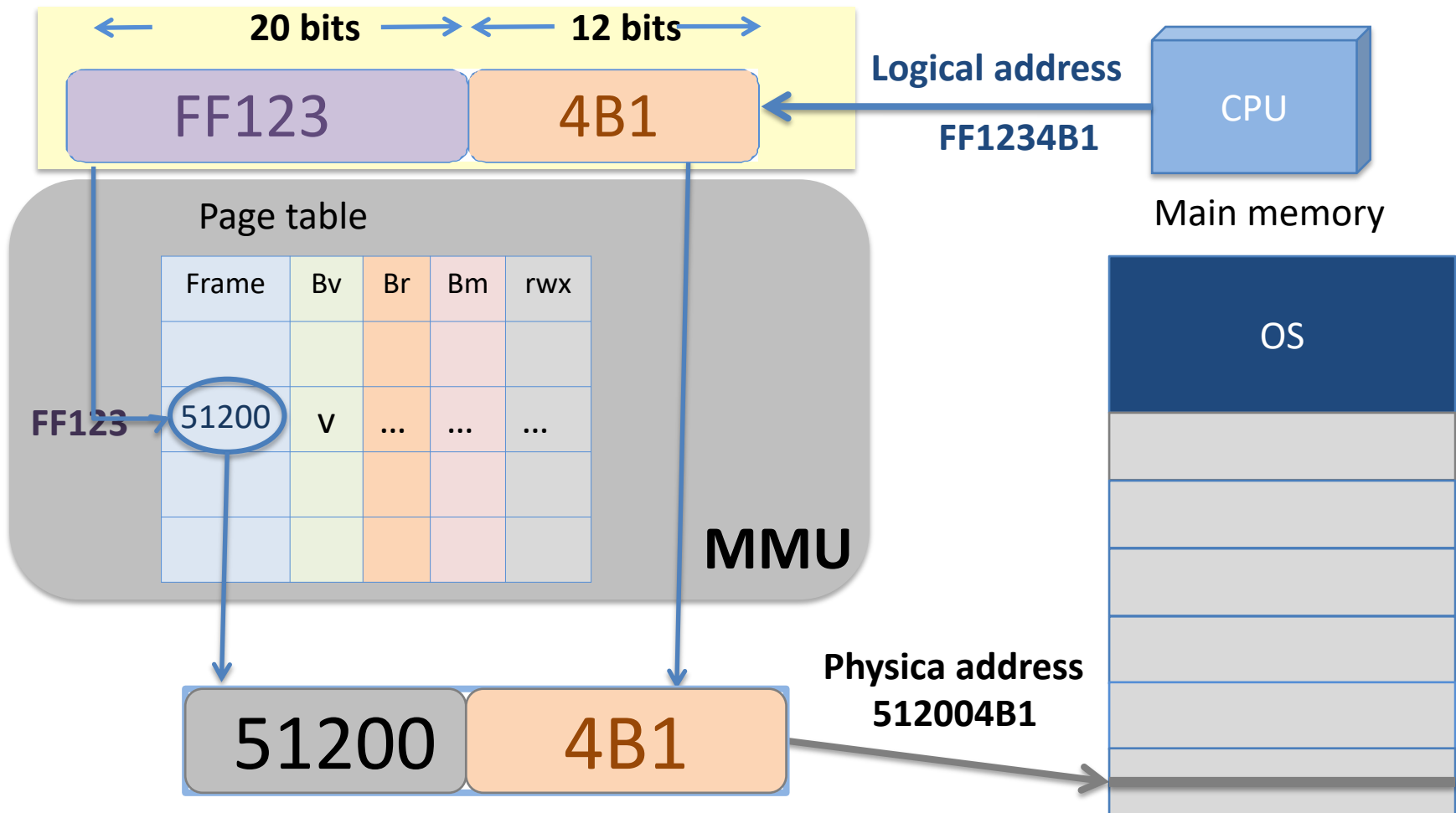


- Address translation



- **Example: paging on a 32 bit microprocessor**

- Page size 4K: $k=12$ bits ($2^{12}=4096$)
- Number of pages = $2^{20} = 1\text{M} = 1048576$ ($m-k = 32-12=20$ bits)
- Logical address = **FF1234B1**



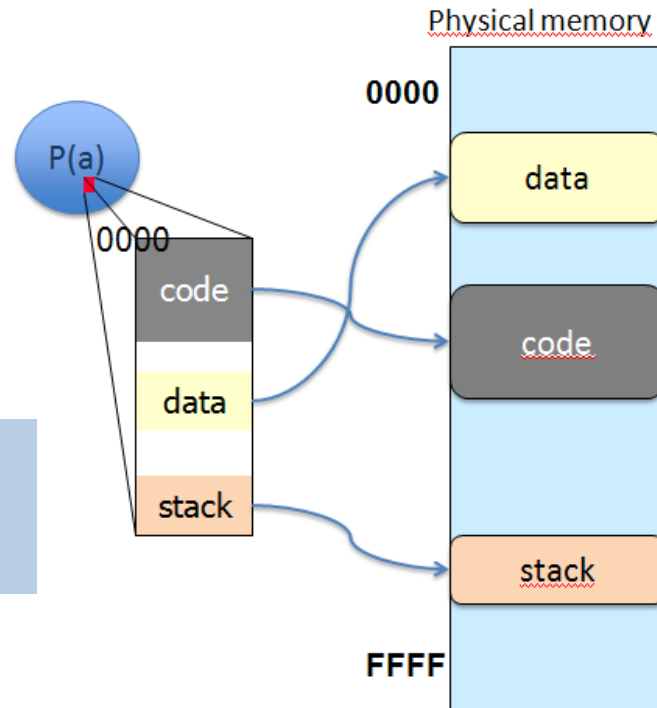
- Advantages
 - There is no external fragmentation
 - It eases reallocation
 - It provides protection
 - Code pages are shared between processes
- Disadvantages
 - **Internal fragmentation:** page size should be an integer power of 2 (actual sizes 4K, 8K, etc) to ease getting page number and offset
 - **Big pages:** a lot of internal fragmentation
 - **Small pages:** very big page tables

- **Page table implementations**
 - MMU registers
 - Only feasible for small logical spaces (few pages)
 - Memory
 - The page table base register (PTBR) is kept with the page table starting physical address
 - An additional memory access is required (page table access) to translate a logical address
 - TLB (*translation look-aside buffer*)
 - The TLB contains only a small subset of page table entries (the ones recently used)
 - Much faster than memory implementation
 - High hit rate with few entries

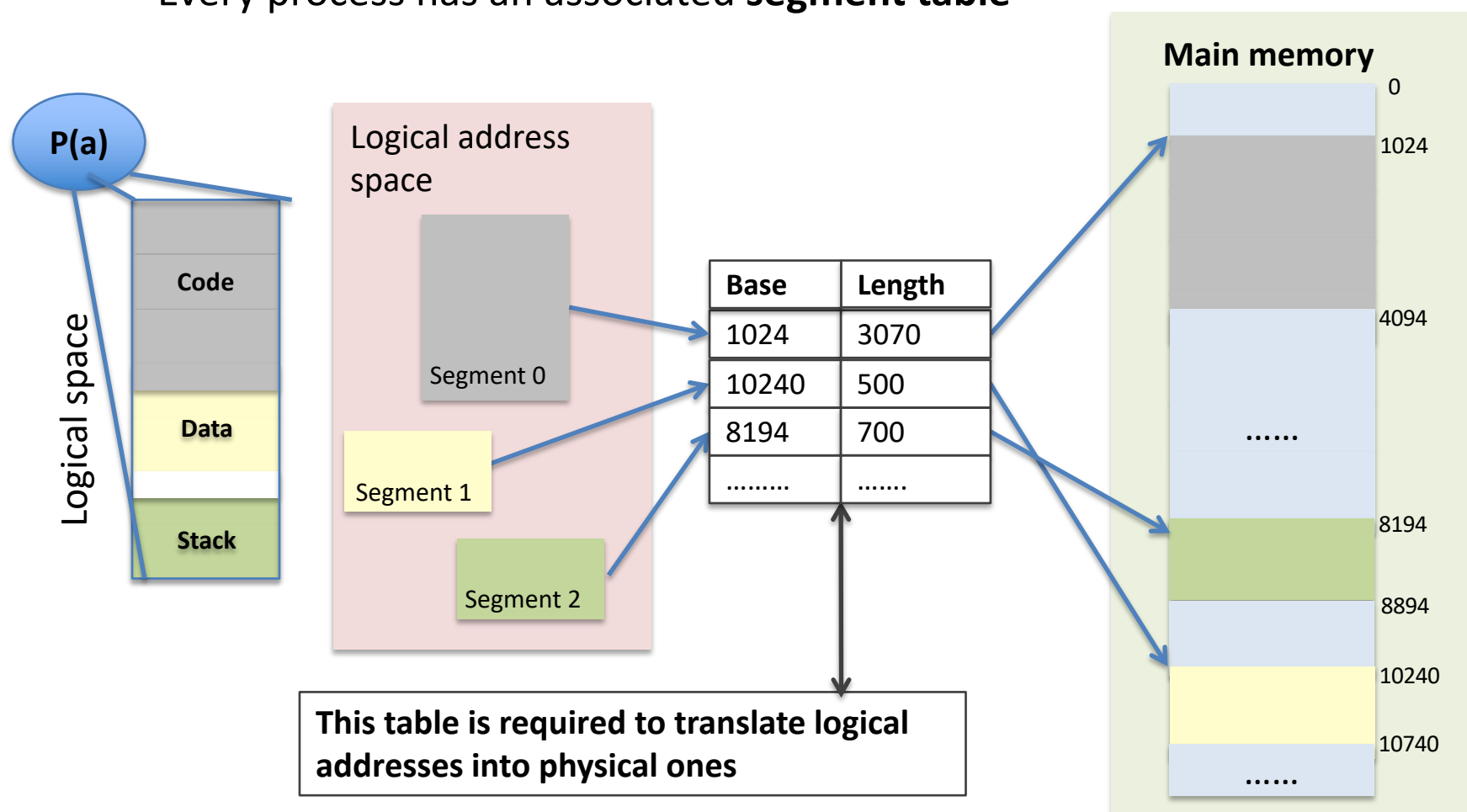
- Sparse memory allocation concept
- Paging
- **Segmentation**
- Multilevel paging

- Process logical space is divided into **variable length fragments**
- Logical space is a set of **segments**
- Every segment has its **name** and **size**
- Segments are defined by the compiler
 - **Code segment**
 - **Data segment**
 - **Stack segment**

A segment is always contiguously allocated in physical memory



- When a process enters the system the OS loads all its segments in main memory
- A **table** is built to store the **starting physical address** and **length** of every segment
 - Every process has an associated **segment table**



- **Segment table**

- Every segment has an entry in the segment table called **segment descriptor** that contains:

- Starting (base) segment physical address
- Segment size
- Set of control bits: **protection bits, valid bit and modified bit**

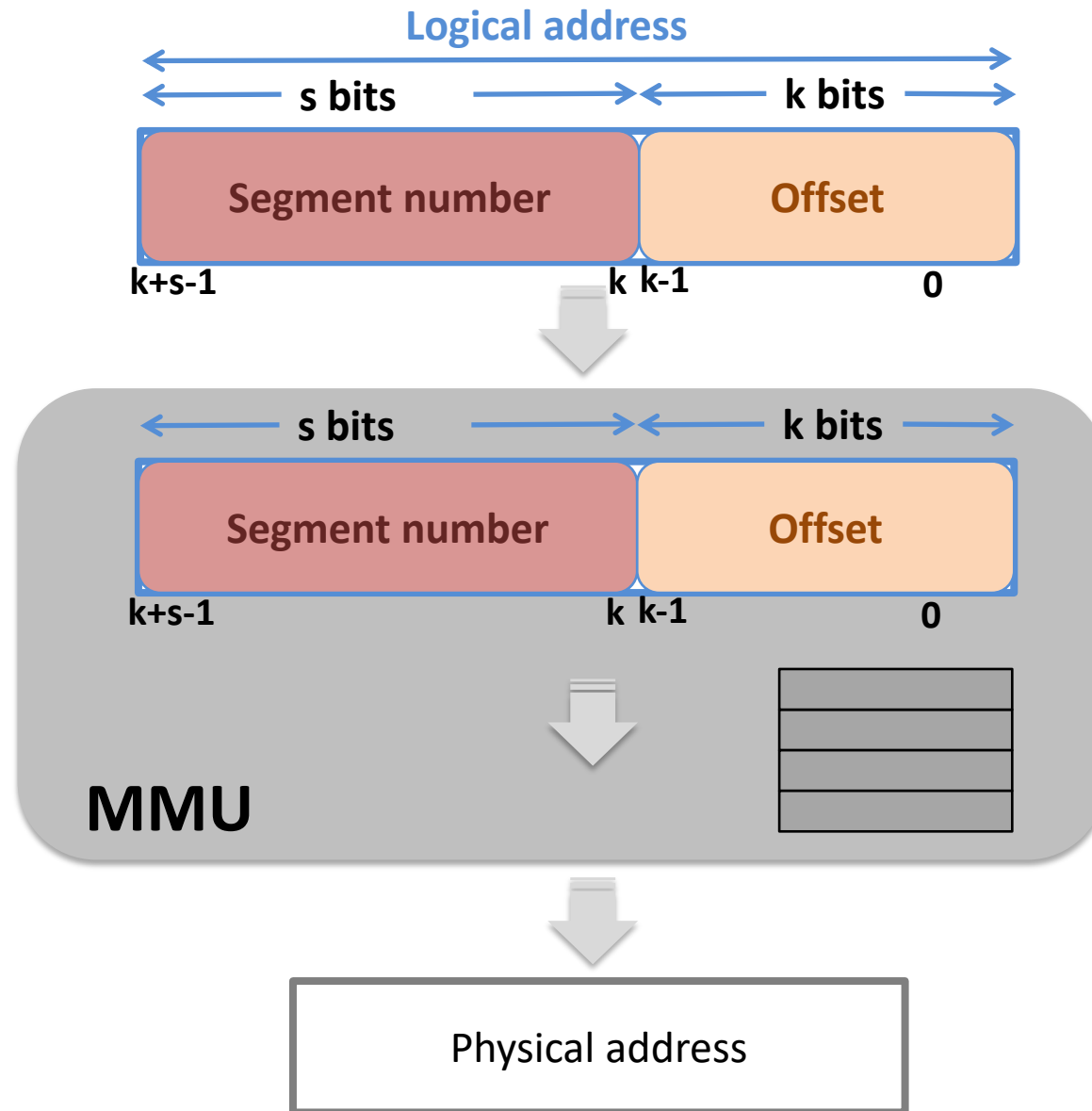
Segment 0 descriptor	Base address	Limit	Valid bit	Modified bit	Protection bits rwx
Segment 1 descriptor	Base address	Limit	Valid bit	Modified bit	Protection bits rwx

Segment n-1 descriptor	Base address	Limit	Valid bit	Modified bit	Protection bits rwx

It contains the segment starting address in main memory

It contains the segment length

- Logical address structure

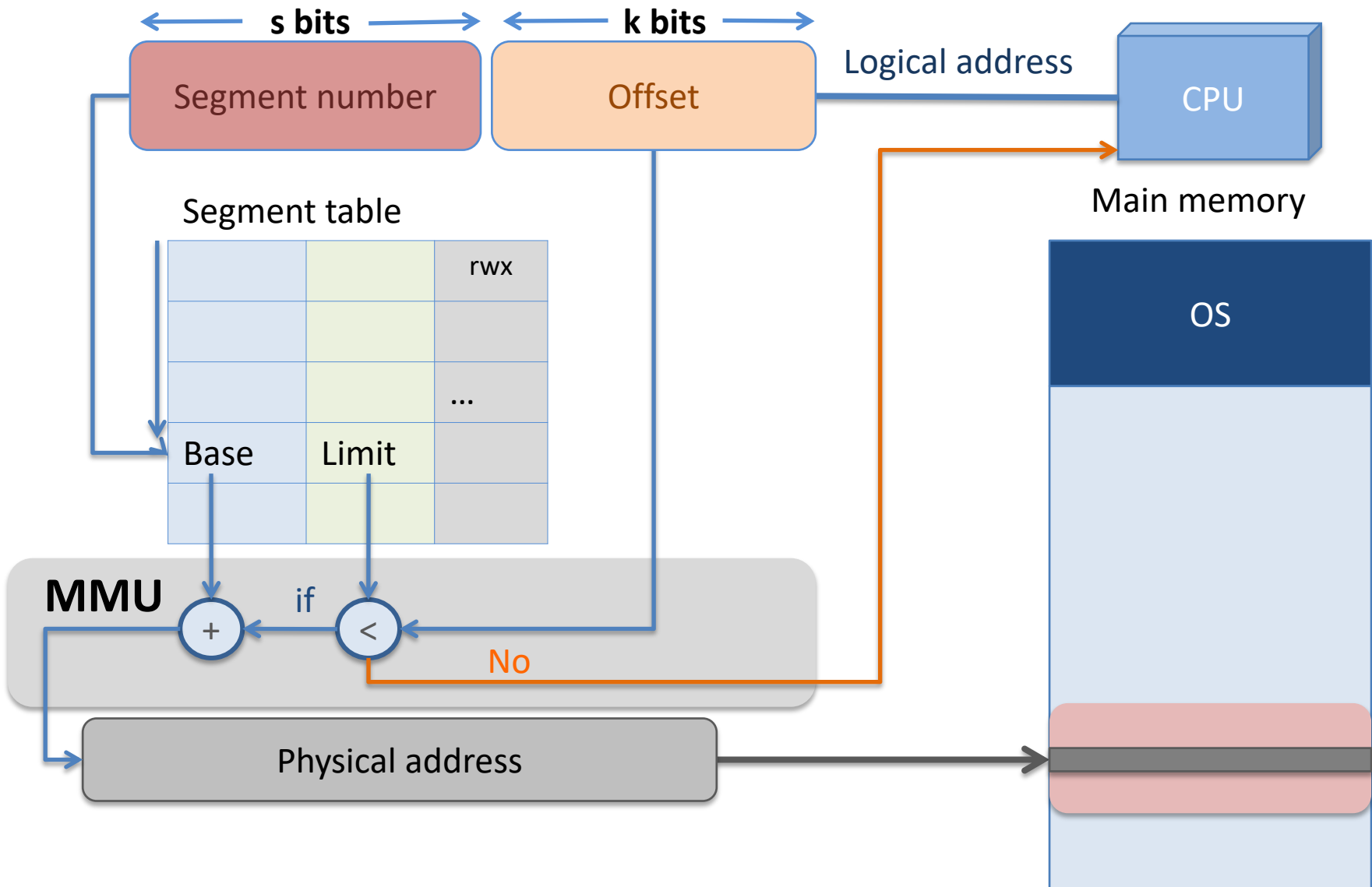


Logical space is made up of 2^s segments. Every segment size is 2^k

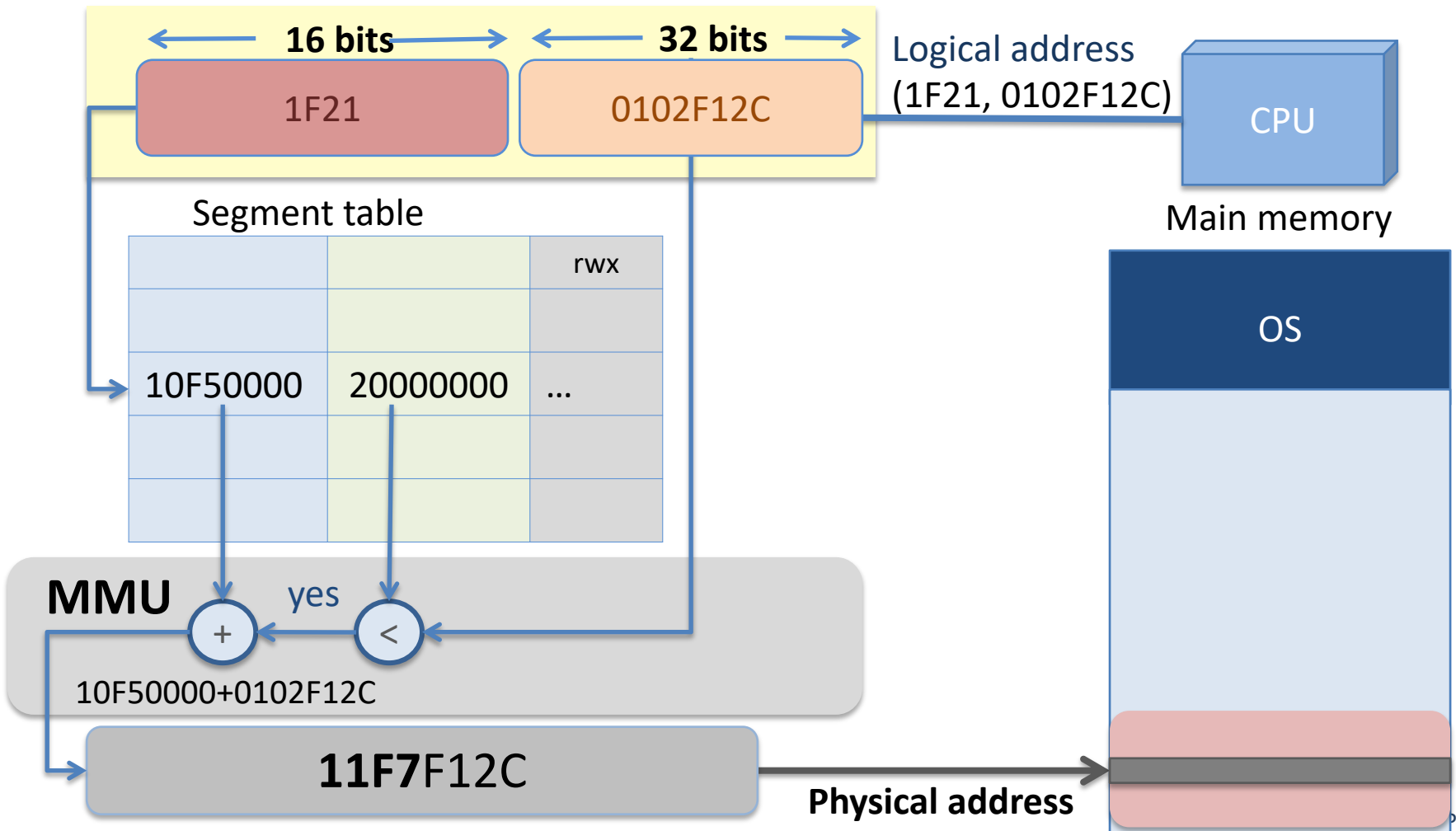
The **MMU** sees the process logical space as 2^s segments of 2^k size

Segments are identified by their number and size

- Address translation



- **Example: segmentation on a 32 bits processor (Intel x86 architecture)**
 - Offset $k = 32$ bits
 - Segmentation $s = 16$ bits
 - Logical address (segment, offset) : (1F21, 0102F12C)



- Advantages
 - There is no internal fragmentation
 - It eases reallocation
 - It provides protection and segment code sharing
- Disadvantages
 - External fragmentation
- Segment size
 - Big → big external fragmentation (same as variable partitions)
 - Small → little external fragmentation but very big segment tables
 - Fixed size → same as paging

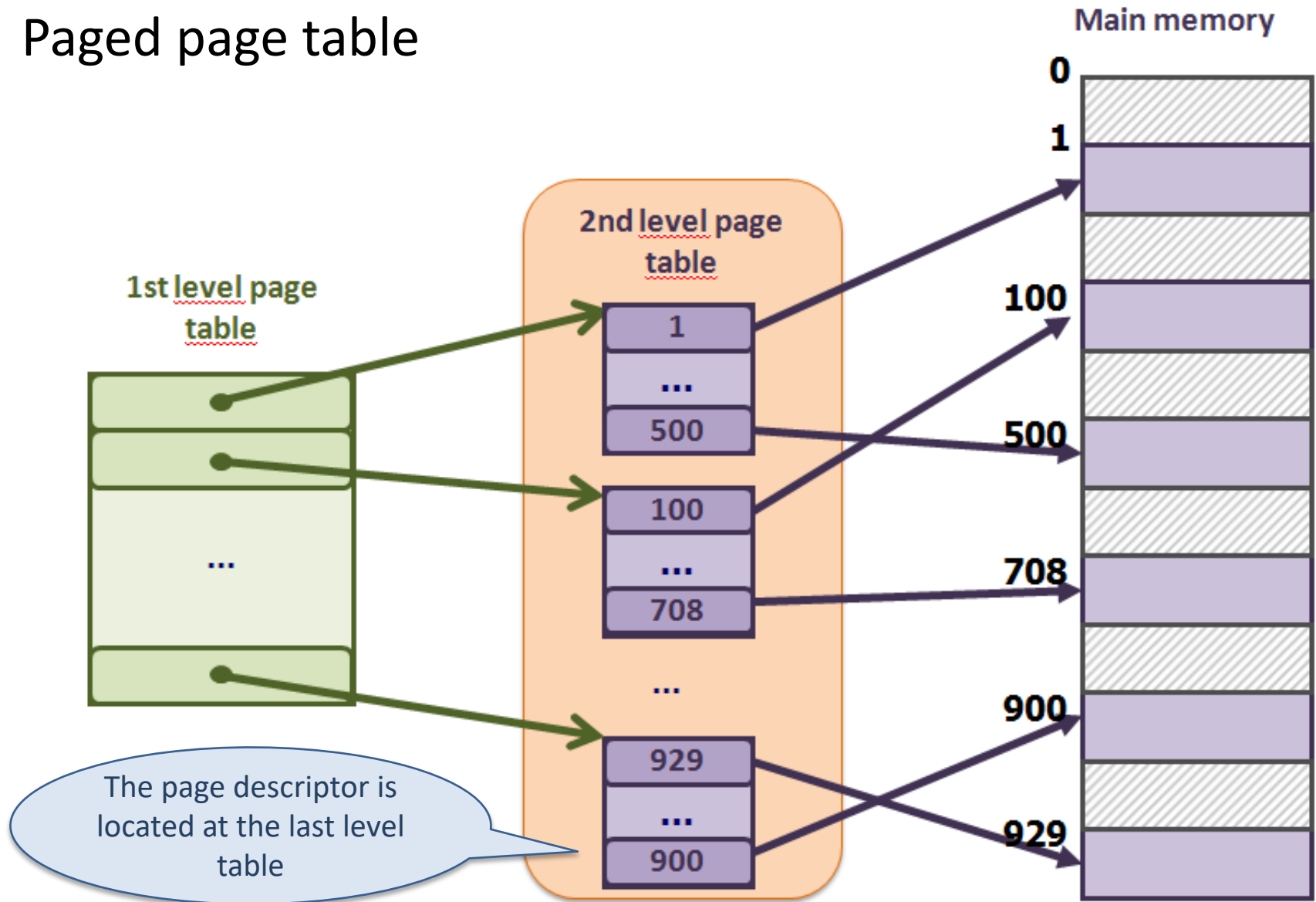
- Sparse memory allocation concept
- Paging
- Segmentation
- **Multilevel paging**

- Motivation
 - When a process has a very big logical address space its page table becomes also very big -> to allocate it contiguously in memory is inefficient
- Solution:
 - To page the page table itself, that means to break it in such a way that every page table fragment will fit a memory frame

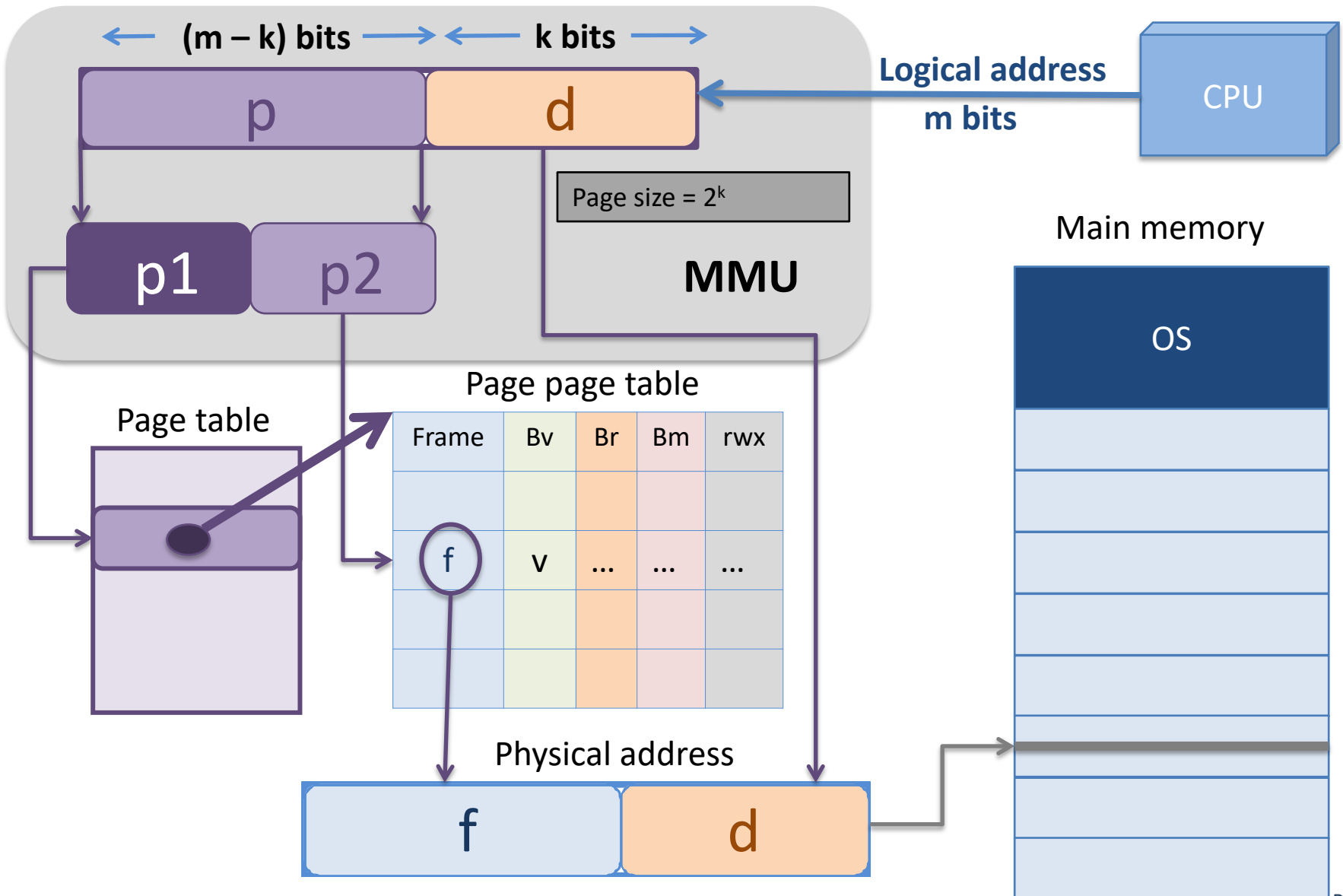


Multilevel paging

Paged page table



- Address translation



- **Example: multilevel paging on a 32 bit microprocessor**
 - Page size 4K: $k=12$ bits ($2^{12}=4096$)
 - Number of pages (8 bit for first level / 12 bits for second level) $8+12=20=m-k$
 - Logical address = **FF1234B1**

