Chapter 3: Memory management

Operating system

@ptit.edu.vn

Posts and Telecommunications
Institute of Technology
Faculty of Information Technology 1

IT1 Faculty
PTIT

August 15, 2023

Table of contents

Content of chapter 3

Memory chaptering

- To execute the process, the OS needs to allocate the necessary memory space to the process.
- Memory allocation and management is an important function of the operating system
- The simplest allocation technique is for each process to be allocated a contiguous area of memory
- The operating system divides memory into continuous parts called partitions, each process will be provided with a chapter to contain its instructions and data.
- The process of dividing memory into chapters is called *memory chaptering*.
- Depending on the choice of chapter location and size, fixed and dynamic chaptering can be distinguished

Memory chaptering (cont.)

Memory management strategies:

- Fixed chapter division
- Dynamic chapter division
- Pagination
- Segment
- Combination of segmentation - paging

Memory chaptering

Fixed chapter division

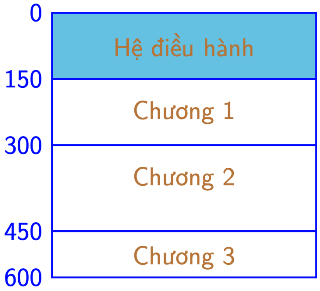
Rule

- Memory is divided into n parts
  - Each part is called a chapter(partition)
  - Each chapter is in a fixed location
  - Chapter is used as an independent memory area
    - Each chapter contains exactly one process
    - When loaded, the process is allocated a chapter. After the process terminates, the OS releases the chapter and the chapter can be allocated to a new process.
  - Chapters can be of equal or different sizes

Memory chaptering (cont.)

Fixed chapter division

Example: Consider the system below where the memory is organized according to fixed chapters, calculate the time it takes the OS to complete the following processes:



Process	Size	time
P_1	120	20
P_2	80	15
P_3	70	5
P_4	50	5
P_5	140	12
Hàng đợi		

Memory chaptering (cont.)

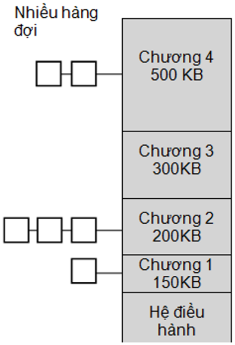
Fixed chapter division

- The size of the chapters is equal:
  - Advantages: Simple
  - Disadvantage: Program size > chapter size leads to unable to allocate
  - Causes internal fragmentation
- Different chapter sizes:
  - There are two ways to choose a memory chapter to give to a waiting process by choosing the chapter with the smallest size
    - Each chapter has its own queue
    - One common queue for all chapters

Memory chaptering (cont.)

Fixed chapter division

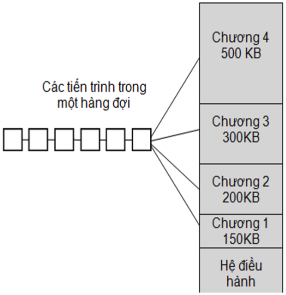
Each chapter has its own queue: *processes of the appropriate size for a chapter will be in that chapter's queue*



- Advantages: Saves memory, reduces internal fragmentation
- Disadvantages: The system is not optimal, at times the large chapter queue is empty, the smaller chapter queue contains many processes.

Memory chaptering (cont.)
Fixed chapter division

One common queue for all chapters: *Whenever there is an empty chapter, the process closest to the top of the queue and with the size most suitable for the chapter will be loaded and executed.*



- Advantages: Saves memory, reduces internal fragmentation and optimizes the system

Memory chaptering (cont.)
Fixed chapter division

Comment on fixed subchapters

- Advantage
  - Simple and less processing
  - Reduce search time
- Disadvantage
  - The parallelism factor cannot exceed the number of memory chapters
  - Fragmented memory
    - The program size is larger than the largest chapter size
    - The total free memory is still large, but is not used to load other programs

Memory chaptering
Dynamic chapter division

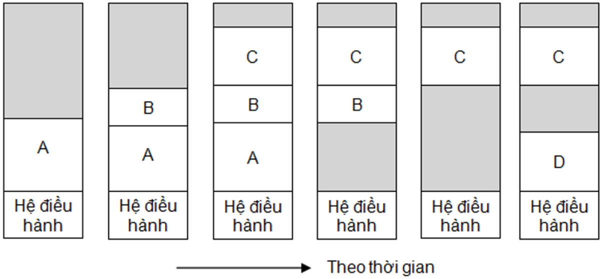
Rule

- Chapter size, number and location are all subject to change
- Free areas of memory can also be linked into a linked list
- When a process requests memory
  - The operating system grants the process a chapter the exact size that the process needs
  - If a corresponding memory area is found
    - The empty area is divided into 2 parts
    - Partial available upon request
    - Partial returns a list of free empty regions
  - If not found
    - Have to wait until there is a satisfactory empty area
    - Allow other processes in the queue to execute (if priority warrants)

Memory chaptering (cont.)

Dynamic chapter division

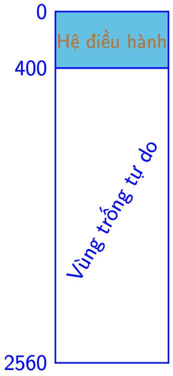
- When the process ends, a free area is created in Memory and the free areas next to each other are merged into a larger area



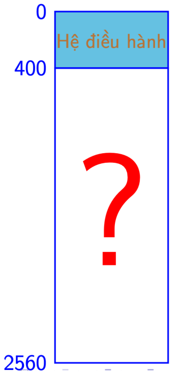
Memory chaptering (cont.)

Dynamic chapter division

Example: Consider the system below where the memory is organized according to dynamic programming, calculate the time it takes the OS to complete the following processes:



Process	Size	time
P_1	600	10
P_2	1000	5
P_3	300	20
P_4	700	8
P_5	500	15
File đợi		



Memory chaptering (cont.)

Dynamic chapter division

- Dynamic chaptering avoids internal fragmentation
- Causes external fragmentation: compresses small free areas into large ones (compression)
- Chapter-level strategies for choosing areas to combat:
  - **First Fit**: Select the appropriate region first
  - **Best Fit**: Most suitable area
  - **Worst Fit**: The most suitable empty area - the largest size area

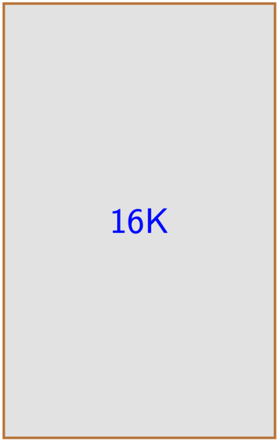
Memory chaptering
Dynamic chapter division

Adjacent method - Buddy system
Rule: *Continuously divide the free space in half until the smallest satisfactory space is obtained*

- Chapters and empty blocks have a size of $2^k (L \leq k \leq H)$
- 2^L: smallest chapter size
- 2^H : size of the entire memory space
- Request S memory allocation
  - $2^{H-1} < S \leq 2^H$: allot 2^H
  - $S \leq 2^{H-1}$ Divide the found empty area into 2 equal blocks (called buddies) 2^{H-1}
    - If $2^{H-2} < S \leq 2^{H-1}$: grant both 2^{H-1}
    - Continue dividing in half until you find a region that satisfies $2^{k-1} < S \leq 2^k$

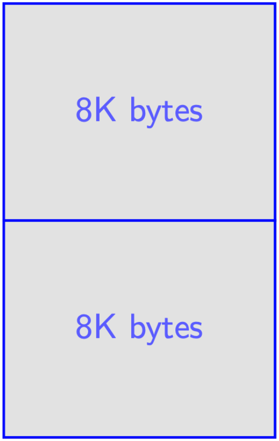
Memory chaptering (cont.)
Dynamic chapter division

For example: 16K Bytes free space, request to allocate a chapter of size 735 Bytes



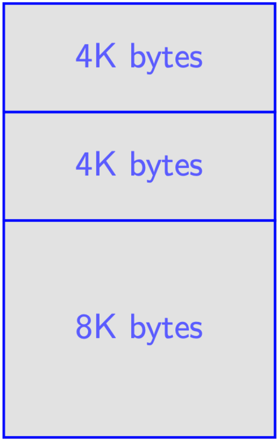
Memory chaptering (cont.)
Dynamic chapter division

For example: 16K Bytes free space, request to allocate a chapter of size 735 Bytes



Memory chaptering (cont.)
Dynamic chapter division

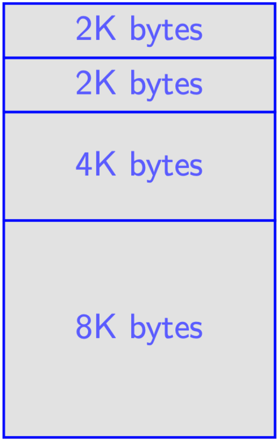
For example: 16K Bytes free space, request to allocate a chapter of size 735 Bytes



Memory chaptering (cont.)

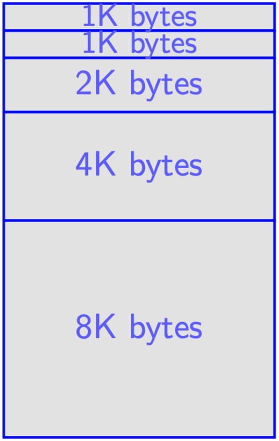
Dynamic chapter division

For example: 16K Bytes free space, request to allocate a chapter of size 735 Bytes



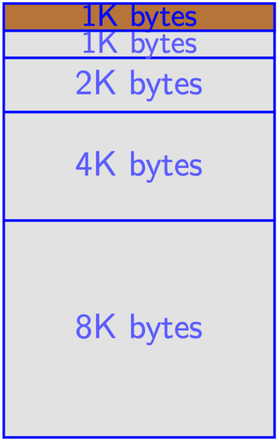
Memory chaptering (cont.)
Dynamic chapter division

For example: 16K Bytes free space, request to allocate a chapter of size 735 Bytes



Memory chaptering (cont.)
Dynamic chapter division

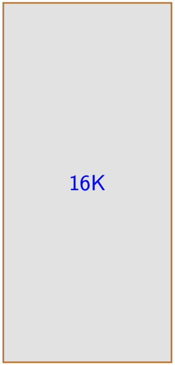
For example: 16K Bytes free space, request to allocate a chapter of size 735 Bytes



Memory chaptering (cont.)
Dynamic chapter division

The adjacency method provides fast memory: with a memory of size n, it is necessary to traverse $log_2 n$ of the list
For example: 16K Bytes free space, allocated one chapter of size

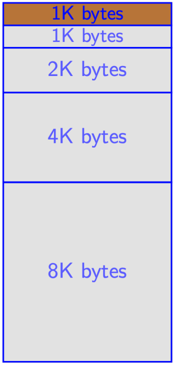
- 835 Bytes



Memory chaptering (cont.)
Dynamic chapter division

The adjacency method provides fast memory: with a memory of size n, it is necessary to traverse $log_2 n$ of the list
For example: 16K Bytes free space, allocated one chapter of size

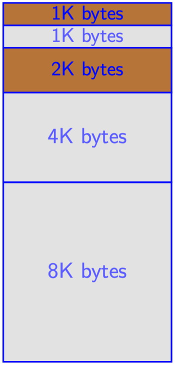
- 835 Bytes



Memory chaptering (cont.)
Dynamic chapter division

The adjacency method provides fast memory: with a memory of size n, it is necessary to traverse $log_2 n$ of the list
For example: 16K Bytes free space, allocated one chapter of size

- 1101 Bytes



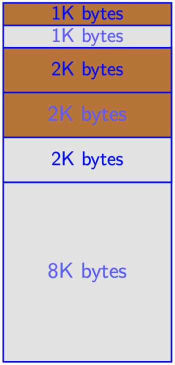
Memory chaptering (cont.)

Dynamic chapter division

The adjacency method provides fast memory: with a memory of size n, it is necessary to traverse $log_2 n$ of the list

For example: 16K Bytes free space, allocated one chapter of size

- 2022 Bytes



Memory chaptering (cont.)

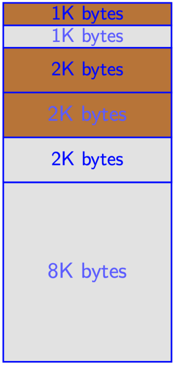
Dynamic chapter division

Recall memory area:
It is possible to combine two adjacent areas of the same size, continuing to combine continuously until creating the largest possible empty area.
For example:

- Giải phóng vùng nhớ thứ nhất (1K)

Memory chaptering (cont.)

Dynamic chapter division

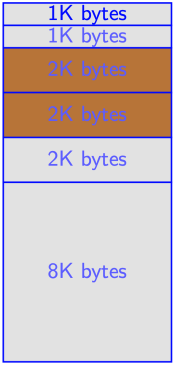


Memory chaptering (cont.)
Dynamic chapter division

Recall memory area:
It is possible to combine two adjacent areas of the same size, continuing to combine continuously until creating the largest possible empty area.
For example:

- Free the first memory area (1K)

Memory chaptering (cont.)
Dynamic chapter division



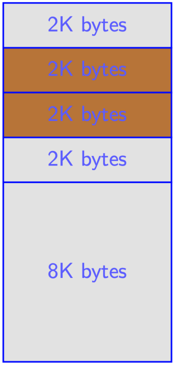
Memory chaptering (cont.)
Dynamic chapter division

Recall memory area:
It is possible to combine two adjacent areas of the same size, continuing to combine continuously until creating the largest possible empty area.
For example: Free the first memory area (1K)

- Combine two 1K regions into a 2K region

Memory chaptering (cont.)

Dynamic chapter division

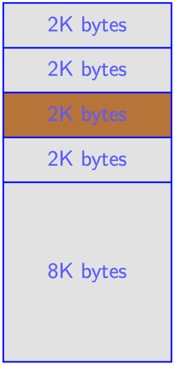


Memory chaptering (cont.)
Dynamic chapter division

Recall memory area:
It is possible to combine two adjacent areas of the same size, continuing to combine continuously until creating the largest possible empty area.
For example:

- Frees the second memory area (2K)

Memory chaptering (cont.)
Dynamic chapter division

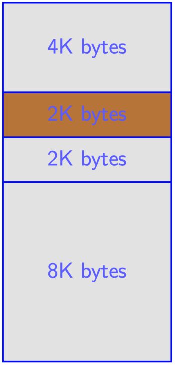


Memory chaptering (cont.)
Dynamic chapter division

Recall memory area:
It is possible to combine two adjacent areas of the same size, continuing to combine continuously until creating the largest possible empty area.
For example: Frees the second memory area (2K)

- Combine two 2K regions into a 4K region

Memory chaptering (cont.)
Dynamic chapter division

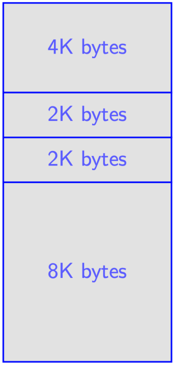


Memory chaptering (cont.)
Dynamic chapter division

Recall memory area:
It is possible to combine two adjacent areas of the same size, continuing to combine continuously until creating the largest possible empty area.
For example:

- Free the third memory area (2K)

Memory chaptering (cont.)
Dynamic chapter division

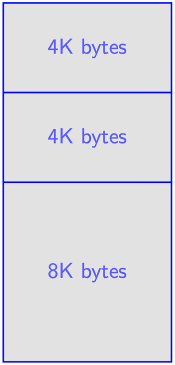


Memory chaptering (cont.)

Dynamic chapter division

Recall memory area:
Can combine 2 adjacent areas of the same size, continue to combine continuously until creating the largest possible empty area.
Ví dụ: Free the third memory area (2K)

- Combine two 2K regions into a 4K region



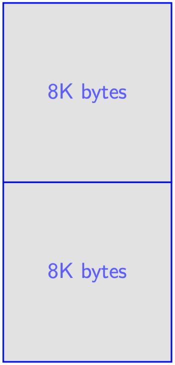
Memory chaptering (cont.)
Dynamic chapter division

Recall memory area:
It is possible to combine two adjacent areas of the same size, continuing to combine continuously until creating the largest possible empty area.
Ví dụ: Free the third memory area(2K)

- Combine two 4K regions into an 8K region

Memory chaptering (cont.)

Dynamic chapter division

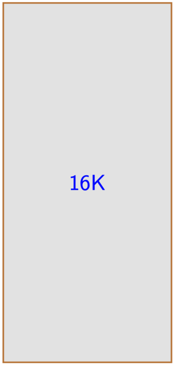


Memory chaptering (cont.)
Dynamic chapter division

Recall memory area:
It is possible to combine two adjacent areas of the same size, continuing to combine continuously until creating the largest possible empty area.
Ví dụ: Free the third memory area(2K)

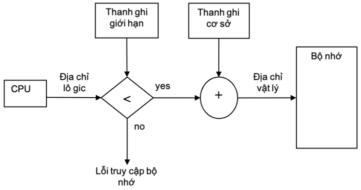
- Combine two 8K regions into a 16K region

Memory chaptering (cont.)
Dynamic chapter division



Memory chaptering
Address mapping and preventing unauthorized access

- When the process is loaded into memory, the CPU reserves 2 registers:
  - Base register: contains the starting address of the process
  - Limit register: contains chapter length



- The logical address is compared with the contents of the limit register
  - If greater: access error
  - Smaller: fed to the adder with the base register as a physical address

Memory chaptering (cont.)

Address mapping and preventing unauthorized access

- If the chapter is moved, the contents of the base register are changed to contain the new address

Memory chaptering
Exchange between memory and disk (swapping)

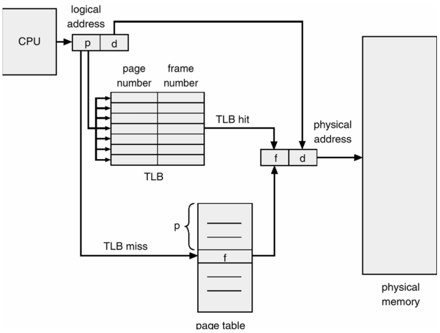
Swap method (*swapping*):

- Ongoing processes can be temporarily loaded to disk to make room for loading other processes
- Then it is loaded again (if not finished) for further execution
  - A process has run out of its CPU time
  - Make room for another process with higher priority
- Load time depends on disk access speed, memory access speed, and process size
- When reloaded, the process can be contained in the chapter in the same location or given a completely new address for the chapter.
- Exchanged processes must be in a resting state, especially not performing I/O operations

Memory paging

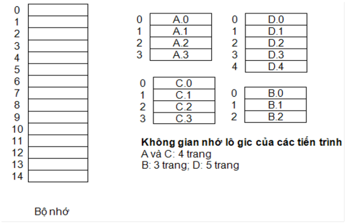
In the chapter segmentation technique, each process occupies a chapter, i.e. a continuous area in memory, leading to fragmentation and not fully utilizing the memory.

Paging technique (*paging*) to limit the disadvantages of chaptering technique:



Memory paging
Pagination concept

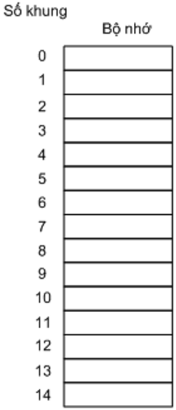
- Physical memory is divided into equal sized blocks: *physical page frame (page frame)*:
  - Physical page frames are numbered 0,1,2,... : physical address of the page frame
  - The page frame is used as the memory distribution unit
- The logical address space of a process or logical memory (program) is divided into blocks called *page (page)* is the same size as the page frame.



Memory paging (cont.)
Pagination concept

- The process is given frames to contain its pages.
- The process is given frames to contain its pages.

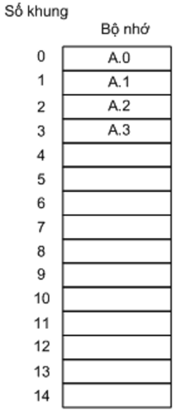
For example: Memory space for processes A, B, C and D:



Memory paging (cont.)
Pagination concept

- The process is given frames to contain its pages.
- Pages can be contained in frames scattered in memory

For example: Memory space for processes A, B, C and D:

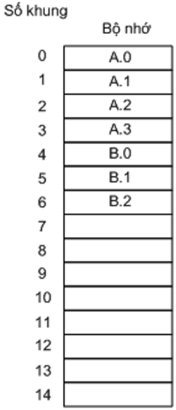


Memory paging (cont.)

Pagination concept

- The process is given frames to contain its pages.
- Pages can be contained in frames scattered in memory

For example: Memory space for processes A, B, C and D:

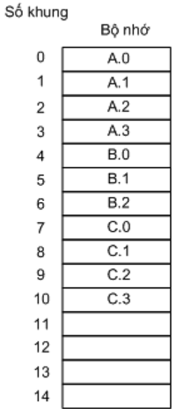


Memory paging (cont.)

Pagination concept

- The process is given frames to contain its pages.
- Pages can be contained in frames scattered in memory

For example: Memory space for processes A, B, C and D:

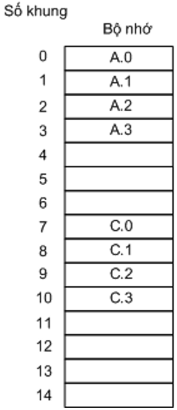


Memory paging (cont.)

Pagination concept

- The process is given frames to contain its pages.
- Pages can be contained in frames scattered in memory

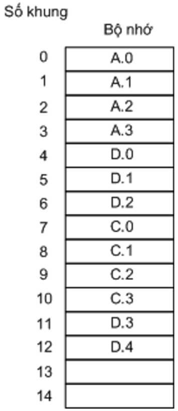
For example: Memory space for processes A, B, C and D:



Memory paging (cont.)
Pagination concept

- The process is given frames to contain its pages.
- Pages can be contained in frames scattered in memory

For example: Memory space for processes A, B, C and D:



Memory paging (cont.)
Pagination concept

Page table: The OS manages frame allocation for each process using the page table: each cell corresponds to 1 page and contains the number of frames allocated to that page.
Each process has its own page table

0	0
1	1
2	2

Bảng trang tiến trình A

0	5
1	6
2	7

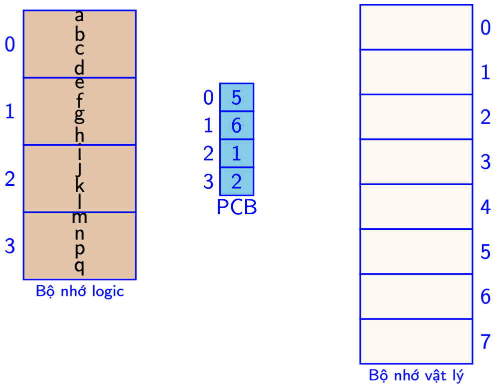
Bảng trang tiến trình C

0	3
1	4
2	8
3	9

Bảng trang tiến trình D

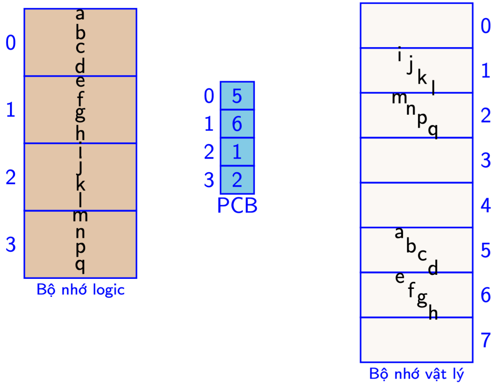
Memory paging

Address mapping logical address when paging to physical address.
Example: Address of g in page 1 (Page number (p): m = 1; Offset in page (o): n = 2)



Memory paging (cont.)
Address mapping

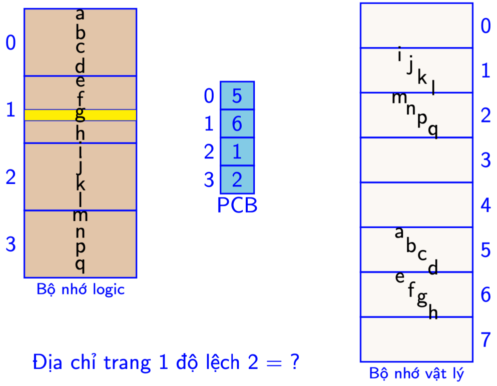
Address mapping maps the logical address in paging to the physical address.
Example: Address of g in page 1 (Page number (p): m = 1; Offset in page (o): n = 2)



Memory paging (cont.)

Address mapping

Address mapping logical address when paging to physical address.
Example: Address of g in page 1 (Page number (p): m = 1; Offset in page (o): n = 2)



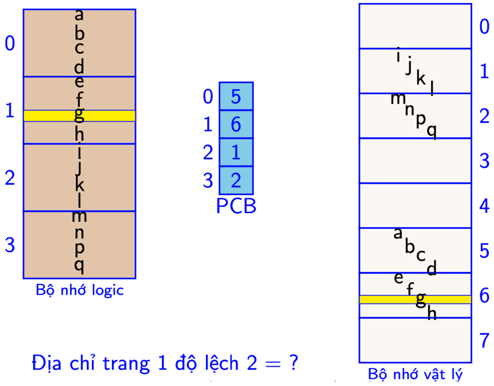
Địa chỉ trang 1 độ lệch 2 = ?

Memory paging (cont.)

Address mapping

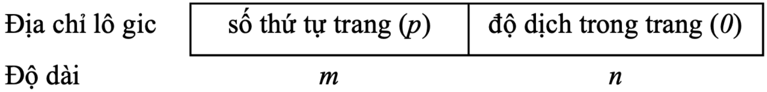
Address mapping logical address when paging to physical address.

Example: Address of g in page 1 (Page number (p): m = 1; Offset in page (o): n = 2) = 6*4 + 2 = 26



Memory paging (cont.)
Address mapping

- **Address mapping** consists of 2 parts:
  - Page number (p)
  - Offset (address offset) of the address compared to the top of the page (o)



Địa chỉ lô gic	số thứ tự trang (*p*)	độ dịch trong trang (*0*)
Độ dài	*m*	*n*

- If the page size is 2^n. Represent a logical address as an address of length (m + n) bits
  - m high bit: represents page number
  - n low bits: represents the offset in the memory page

Memory paging (cont.)
Address mapping

For example: For a memory page of size 1024Byte, the logical address length is 16 bits. Calculate logical address 1500: Calculate page number and page offset?

Memory paging (cont.)
Address mapping

For example: For a memory page of size 1024Byte, the logical address length is 16 bits. Calculate the logical address of physical address 1500? Calculate the number of pages and page translation?

Explain:

method 1:

m+n = 16 because the logical memory space is 16 bits

The number of frames in a page is $1024 Bytes = 2^{10} Bytes$, so 10 bits are needed to represent a frame in a page.

=> n = 10 ; m = 6

$1500 = 000001\mathit{0111011100}$

Vậy ta có $p = 000001_2 = 1; o = 0111011100_2 = 476$

Method 2:

Phần $p = 1500/1024 = 1$

Phần $o = 1500 mod 1024 = 476$

Vậy ta có $p = 1; o = 476$

Memory paging (cont.)
Address mapping

- Convert logical address to physical address:
  - Take the m high bits of the address => get the page number
  - Based on the page table, find the physical frame number (k)
  - The starting physical address of the frame is $k * 2^n$
  - **Physical address of referenced byte** $=$ **start address of frame** $+$ **Offset address (offset)**
- **Comment** : Just add the frame number before the bit sequence representing the offset

Memory paging (cont.)
Address mapping

For example: Given the page table as follows, with a page size of 1KB. Let's convert the following logical addresses to physical addresses:

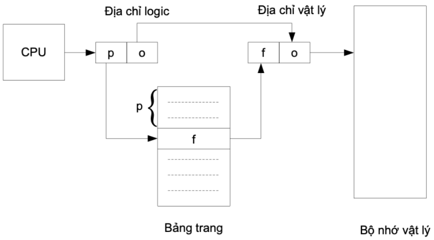
- 1240
- 3580
- 1502

Page	Frame
0	3
1	2
2	6
3	4

- Instruct:
  - B1: Get (Logical address) div (page size) = page => Determine Frame
  - B2: Frame * (page size) + (logical address) mod (page size) = physical address

Memory paging (cont.)
Address mapping

- The conversion process from logical address to physical address is performed by hardware
- Page size is a power of 2, ranging from 512B to 16MB
- Separating the p and o parts of a logical address is easily done by bit shifting



Memory paging (cont.)
Address mapping

Advantages of pagination

- External fragmentation does not exist: Fragmentation during pagination has an average value of half a page
- High parallelism and allows page sharing
- The address mapping mechanism is completely transparent to the program

Disadvantages of pagination

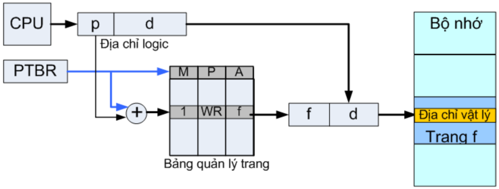
- The phenomenon of internal segmentation exists
  - Always appears on the last page
  - Reducing page size allows saving MEM?
    - Small page size => large number, large page table, difficult to manage
    - It is inconvenient to exchange with the disk because it is done in large blocks

Memory paging (cont.)
Address mapping

- Requires hardware support for large paging strategy
- When the program is large, the table manages pages with many elements

Memory paging
Organize page tables

- Each memory access operation requires access to the page table => organize the page table so that access speed is highest



- Use a set of registers as a page table:
  - Very high access speed
  - Limited number of registers => not applicable
- Keep page tables in MEM:
  - The location of each table is pointed to by the PTBR (Page Table Base Register)
  - Time to access table => use high speed cache

Memory paging (cont.)

Organize page tables

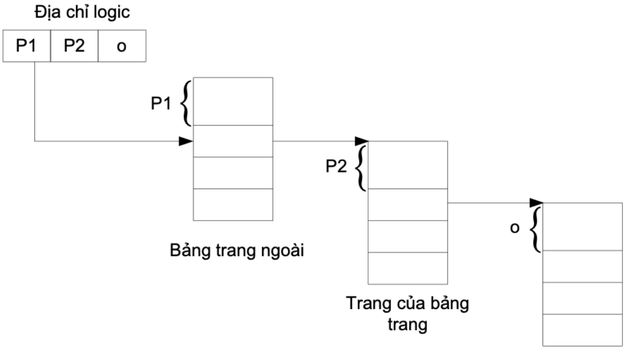
Multi-level page table:

- Modern computing systems allow the use of large logical address spaces ($2^{32} -> 2^{64}$) $=>$ The number of pages that need to be managed increases, leading to an increase in page table size
- **Rule**: *Paginated site management panel*
  - Divide the page table into smaller parts
  - Multi-level page table organization: Upper-level table entries point to other page tables
- **Example of a 2-level table**:
  - The computer has a logical address space of 4GB ($2^{32}B$), memory page size is 4KB ($2^{12}B$):
    - Page number $2^{32}/2^{12} = 2^{20} => m = 20bit$
    - The amount of translation in the page $2^{12} => n = 12bit$
- The page table is paginated. The page number is divided into 2 parts:

Memory paging (cont.)
Organize page tables

- P1: 10 bit Allows locating items in the above level table
- P2: 10 bit Locate the item in the lower level table
- O: 12 bit, contains the translation within the page

▶ The access address has the form: P_1, P_2, o



summary

Chapter 3 Memory management

- Memory chaptering
- Memory paging

Chapter 3 Memory management

- Memory segmentation
- Virtual memory

Questions and exercises

1. **Question 1**: Given a memory size of 1MB. Use the adjacency method to allocate processes in turn with the following sizes: A : 120KB, B : 210KB, C : 150KB, D : 40KB.
2. **Question 2**: Logical memory space consists of 512 pages, each page has size 1024B. Physical memory consists of 64 frames. Indicates how many bits long the logical address is, how many bits the page number and page offset are respectively. How many bits long is the physical address?
3. **Question 3**: Paging system, page size 1024B. The current paging table is as shown, calculate the physical address for the logical addresses:

3	0
2	4
1	
0	1

   3.1 1020

Questions and exercises (cont.)

3.2 2060