



HCMUS - FIT

Virtual Memory

Lecturer: VU THI MY HANG

OPERATING SYSTEM

Plan

- Fundamentals
- Segmentation
- Paging

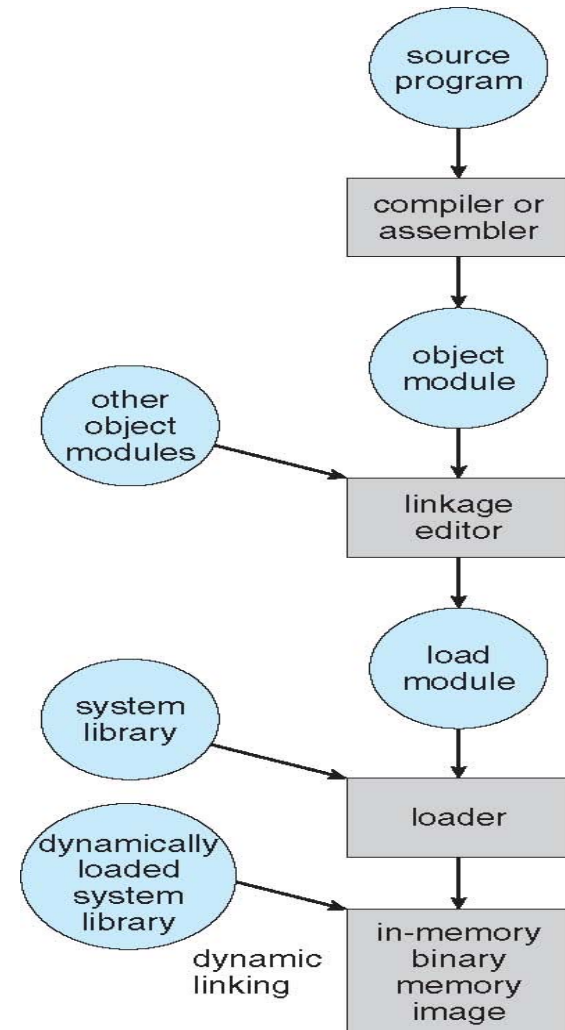
Plan

- **Fundamentals**
- Segmentation
- Paging

Fundamentals

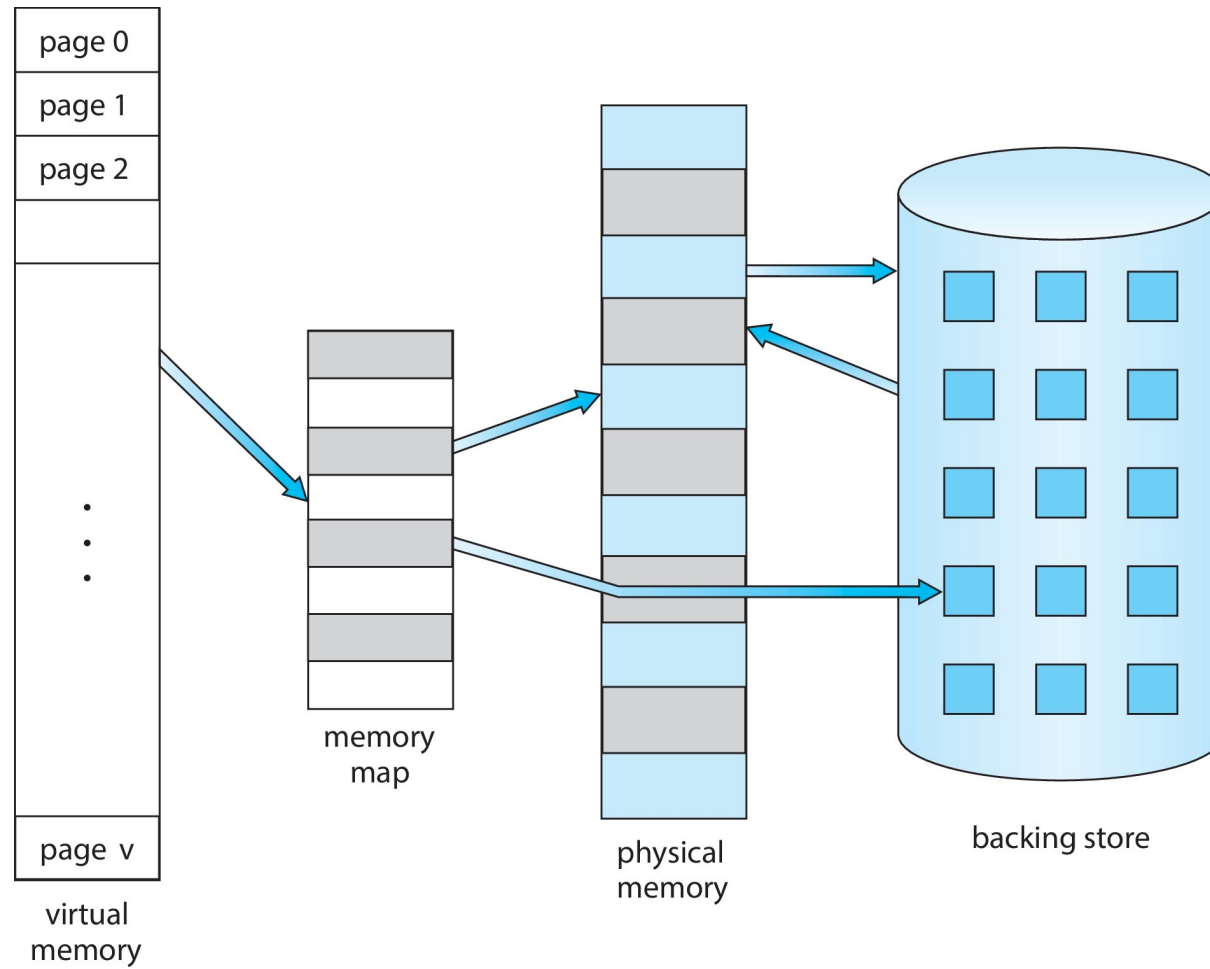
Basic definitions

- Dynamic linking
 - ✓ Linking is delayed until execution time
- Dynamic loading
 - ✓ Routine is not loaded until it is called
- Overlays
 - ✓ Only some parts of a process that are currently executing will be loaded into memory



Fundamentals

Virtual Memory





Non-contiguous Memory Allocation

- Process is divided into different parts and loaded into various memory partitions, which can be **non-contiguous**.
- Two fundamental techniques: **Segmentation** and **Paging**.



Memory

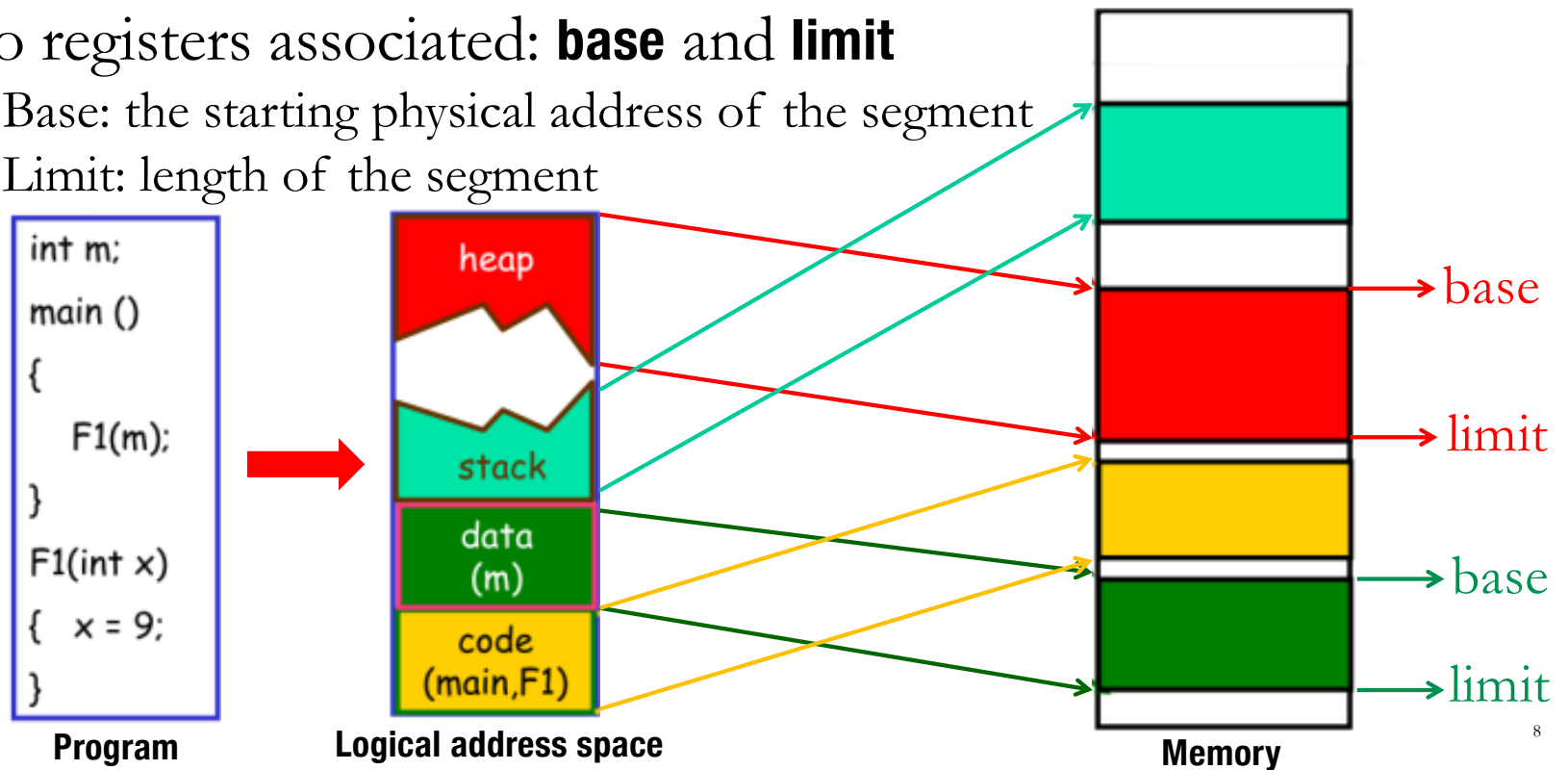
Plan

- Fundamentals
- **Segmentation**
- Paging

Segmentation

Principle

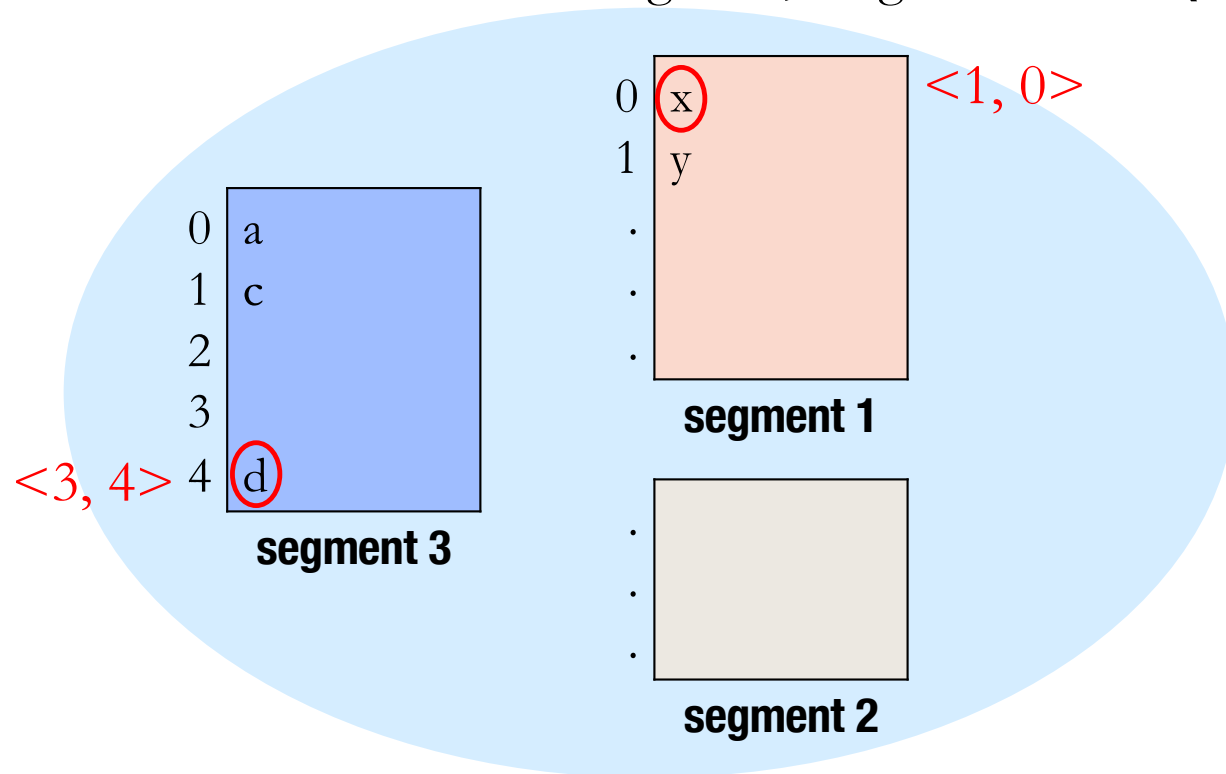
- A process is divided into segments (varying in size) and loaded in non-contiguous partitions in the memory.
- Two registers associated: **base** and **limit**
 - Base: the starting physical address of the segment
 - Limit: length of the segment



Segmentation

Address Binding and Protection

- Logical address contains two parts: $\langle s: \text{segment number}, d: \text{offset} \rangle$
 - Offset: relative address within a segment, ranges from **0** to **(limit – 1)**

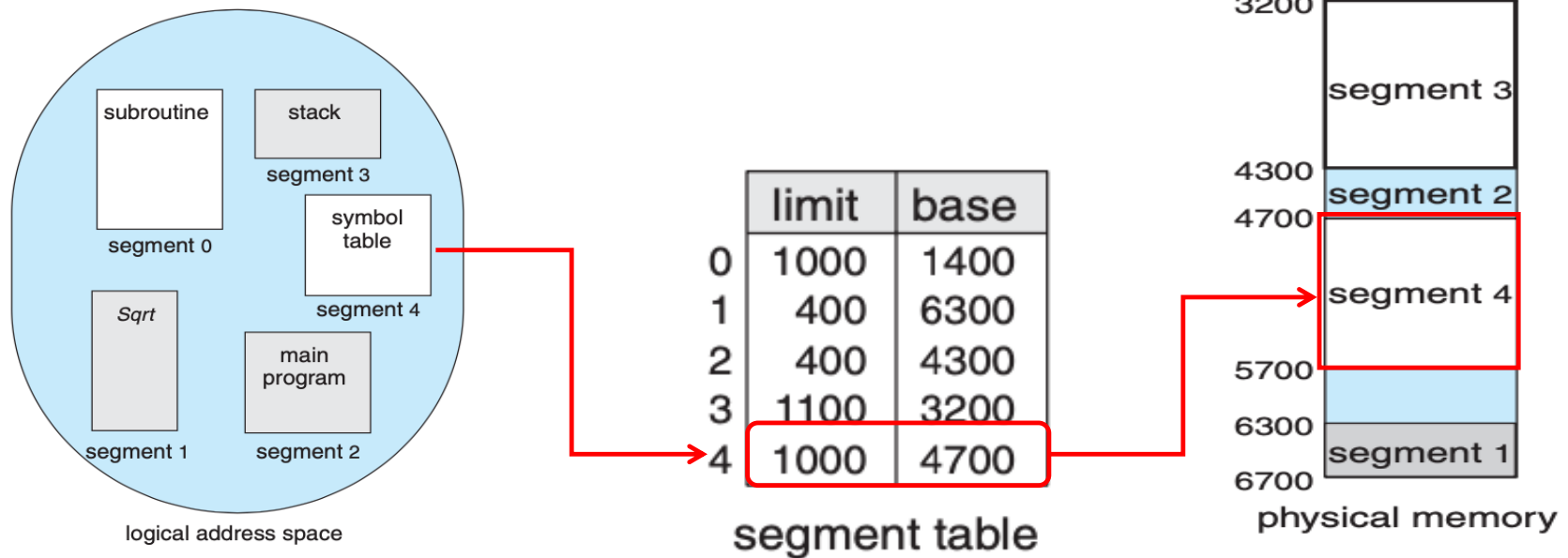


Segmentation

Address Binding and Protection

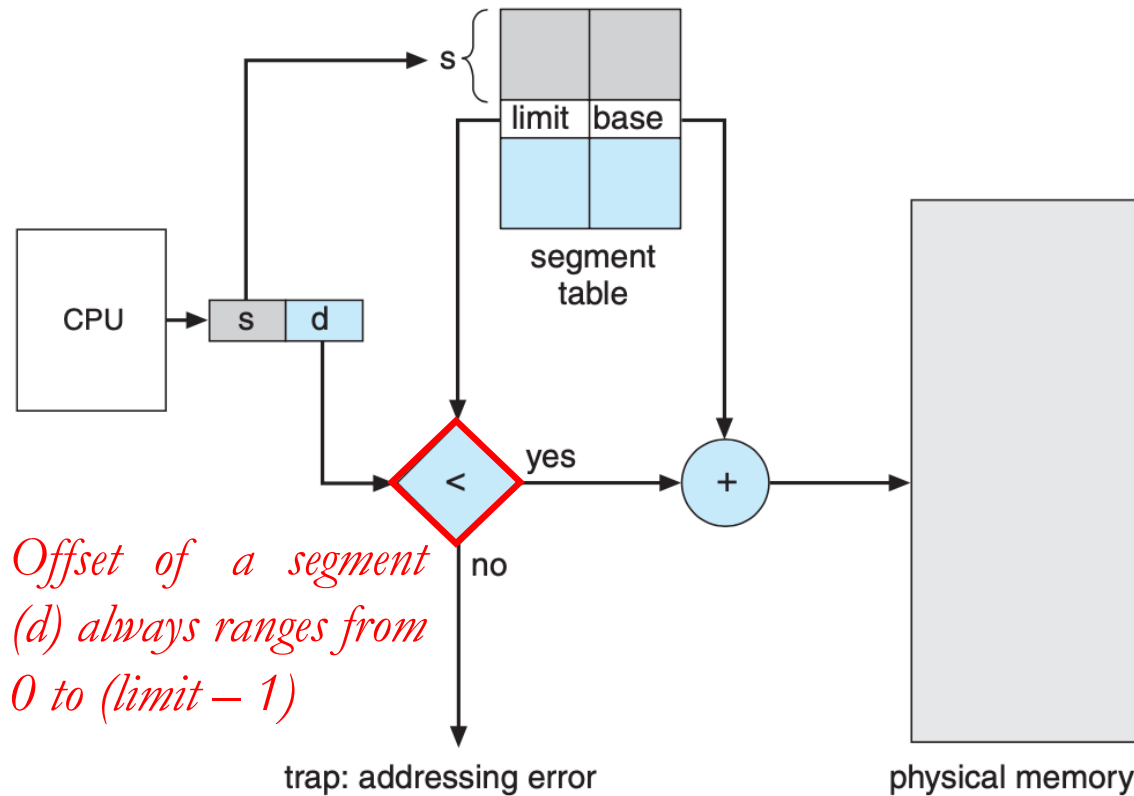
Segment Table

- Used for segment – memory mapping
- Each table entry contains base and limit values of a segment



Segmentation

Address Binding and Protection



	limit	base
0	1000	1400
1	400	6300
2	400	4300
3	1100	3200
4	1000	4700

segment table

Logical address	Physical address
<4, 1200>	invalid
<2, 1000>	invalid
<0, 200>	1600

Plan

- Fundamentals
- Segmentation
- **Paging**
 - ❑ *Principles*
 - ❑ *Page Replacement Algorithms*
 - ❑ *Paging with Large Address Space*

Plan

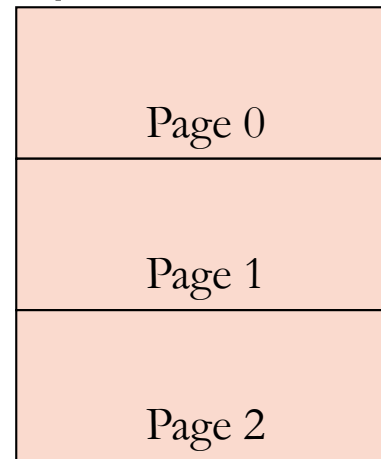
- Fundamentals
- Segmentation
- **Paging**
 - ❑ *Principles*
 - ❑ *Page Replacement Algorithms*
 - ❑ *Paging with Large Address Space*

Memory Allocation with Paging

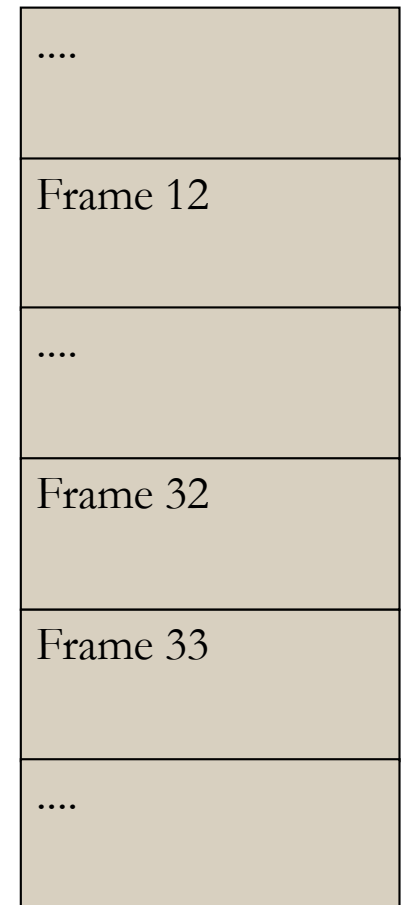
- Process divided into fixed-size pages
- Memory divided into fixed-size frames
- A page will be loaded into one frame

Page size = Frame size

(This size is defined by computer systems)



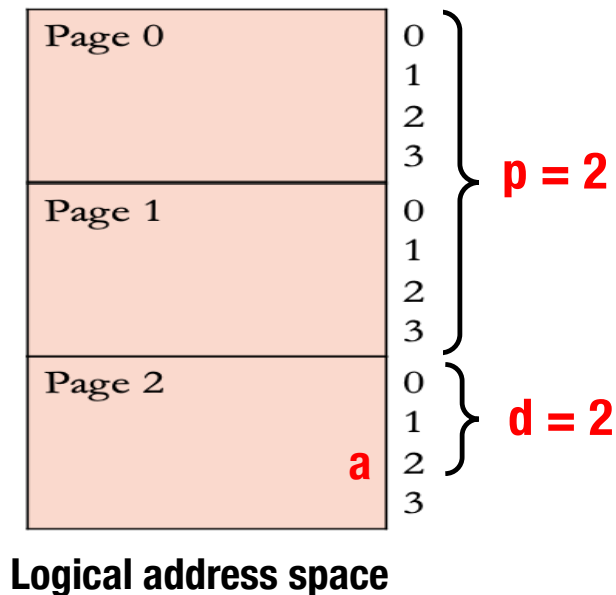
Logical address space



Memory

Logical and Physical Address

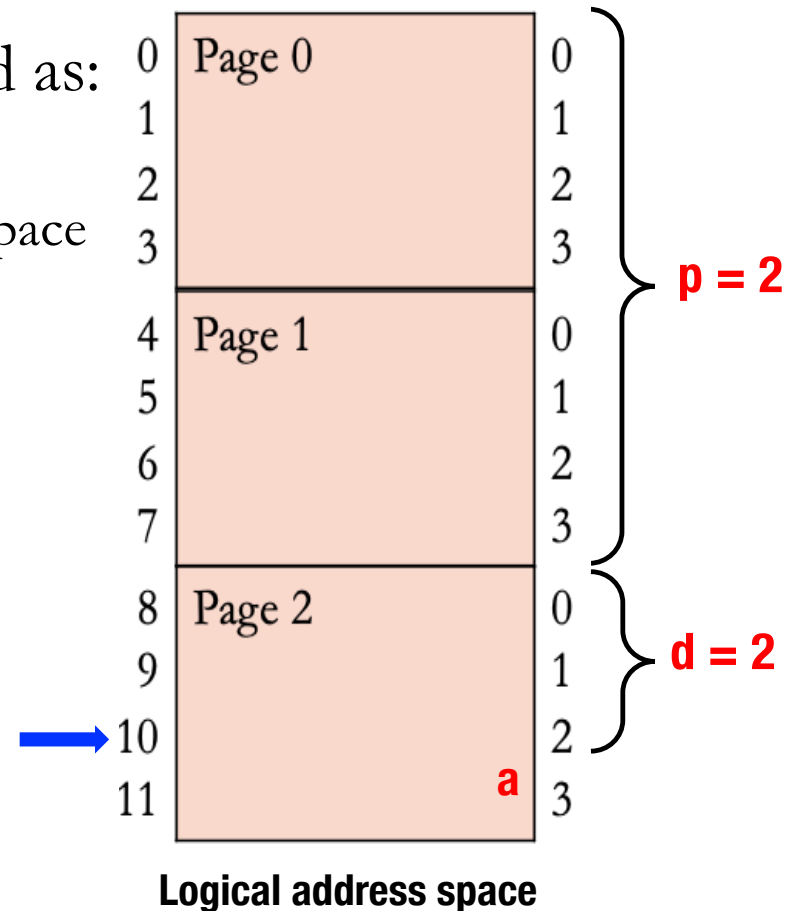
- Logical address contains two parts: $\langle p: \text{page number}, d: \text{offset} \rangle$
 - Offset: relative address within a page, starting at 0
- Physical address contains two parts: $\langle f: \text{frame number}, d: \text{offset} \rangle$
 - Offset: relative address within a frame, starting at 0



Address representation

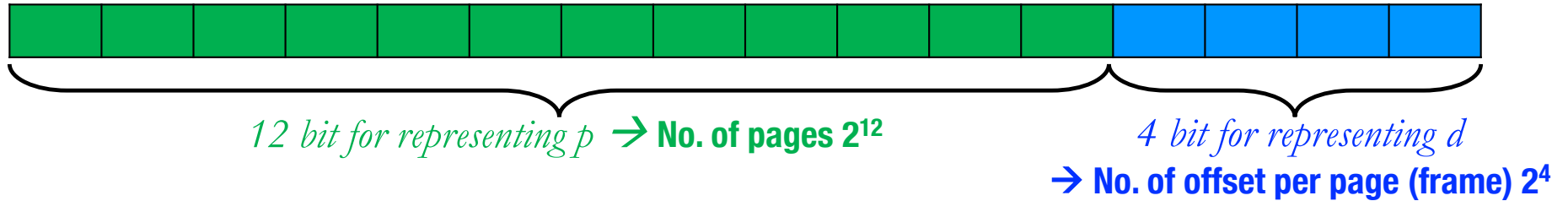
- A logical address can be represented as:
 - $\langle p, d \rangle$
 - Absolute value within logical address space

Absolute value of an address =
 $p * (\text{No. of offset per page}) + d$

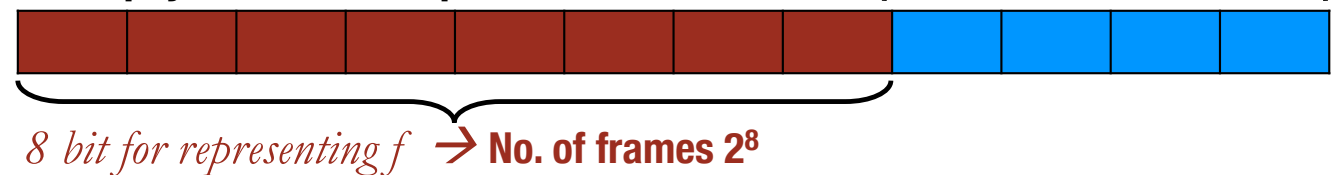


Logical and Physical Address with Paging

16-bit logical address space



12-bit physical address space



Each address points to a byte-addressable in the memory

(Note: If memory is word-addressable, each address, generally, points to a 4-byte word)

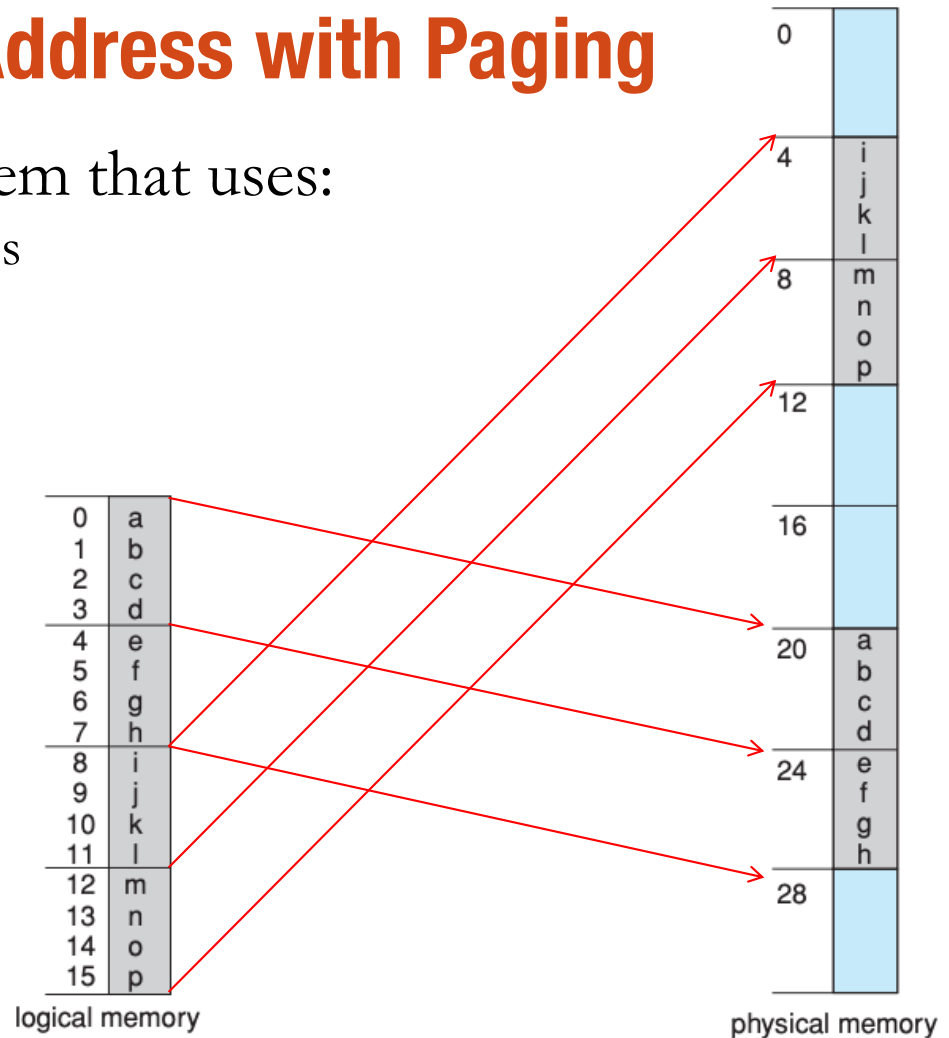
→ Size of a page (and frame): 2^4 bytes = 16 bytes

→ Size of logical address space: 2^{16} bytes = 64KB

→ Size of physical memory: 2^{12} bytes = 4KB

Logical and Physical Address with Paging

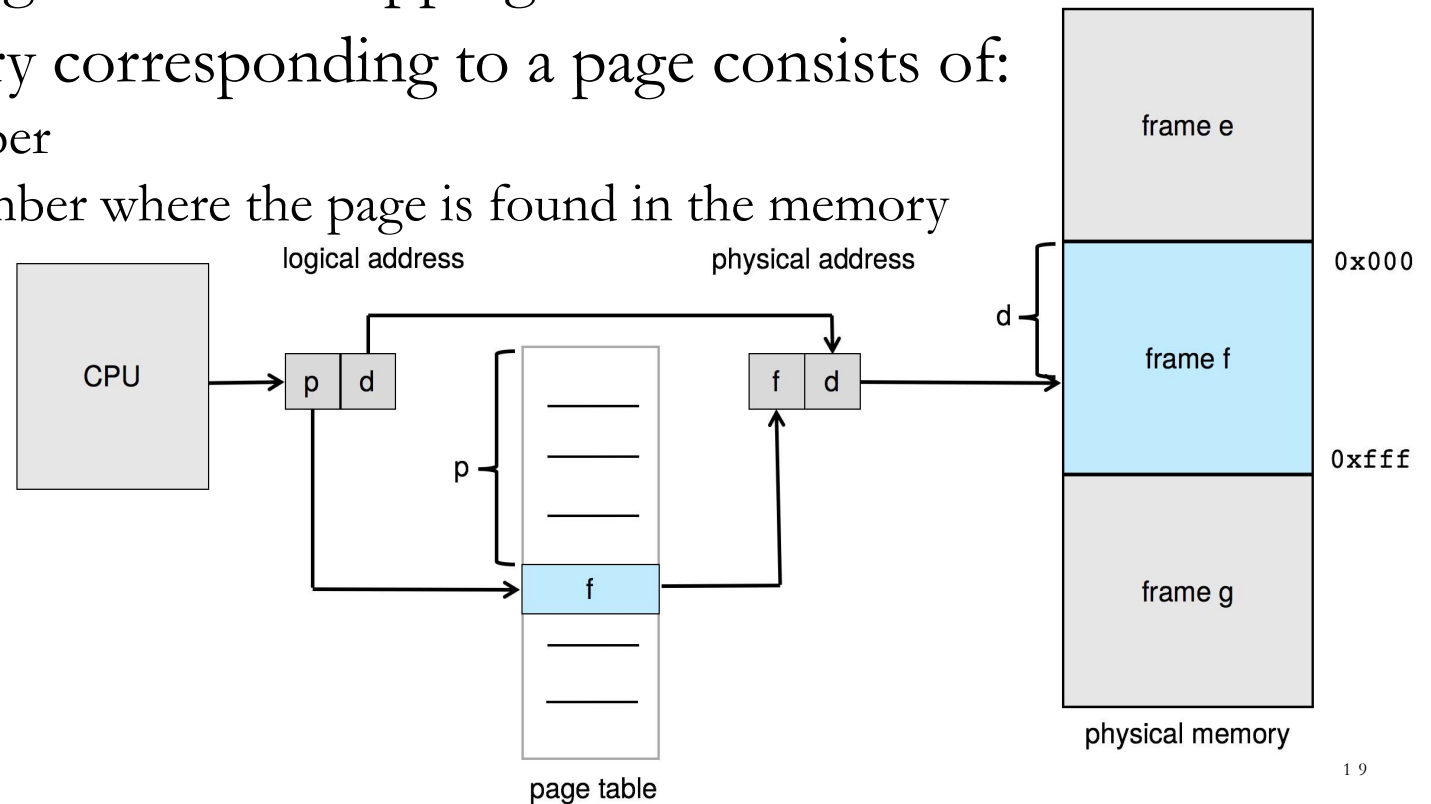
- Paging example for a system that uses:
 - Physical memory of 32 bytes
 - Page size of 4 bytes
 - 4-bit logical address space



Address Binding and Protection

Page Table

- Used for page – frame mapping
- A table entry corresponding to a page consists of:
 - Page number
 - Frame number where the page is found in the memory



Address Binding and Protection

An example of Page Table Entry Structure

Frame Number	Valid/Invalid	Protection	Caching	Referenced	Modified
---------------------	---------------	------------	---------	------------	----------

- Contain page information
- Depend on the operating system

Address Binding and Protection

An example of Page Table Entry Structure

Frame Number	Valid/Invalid	Protection	Caching	Referenced	Modified
-------------------------	---------------	------------	---------	------------	----------



Most important information

Indicate where to find the page in the memory

Address Binding and Protection

An example of Page Table Entry Structure

Frame Number	Valid/Invalid	Protection	Caching	Referenced	Modified
--------------	---------------	------------	---------	------------	----------



(also called as Present/Absent bit)

1: Page is present in the memory

0: Page is not present in the memory

Address Binding and Protection

An example of Page Table Entry Structure

Frame Number	Valid/Invalid	Protection	Caching	Referenced	Modified
--------------	---------------	------------	---------	------------	----------



(also called as Read/Write bit)

1: Read-only

0: Read/Write

Address Binding and Protection

An example of Page Table Entry Structure

Frame Number	Valid/Invalid	Protection	Caching	Referenced	Modified
-------------------------	---------------	------------	---------	------------	----------



1: Disable page caching
0: Enable page caching

Address Binding and Protection

An example of Page Table Entry Structure

Frame Number	Valid/Invalid	Protection	Caching	Referenced	Modified
-------------------------	---------------	------------	---------	------------	----------



1: Page has been referred recently
0: Page has not been referred recently

Address Binding and Protection

An example of Page Table Entry Structure

Frame Number	Valid/Invalid	Protection	Caching	Referenced	Modified
---------------------	---------------	------------	---------	------------	----------

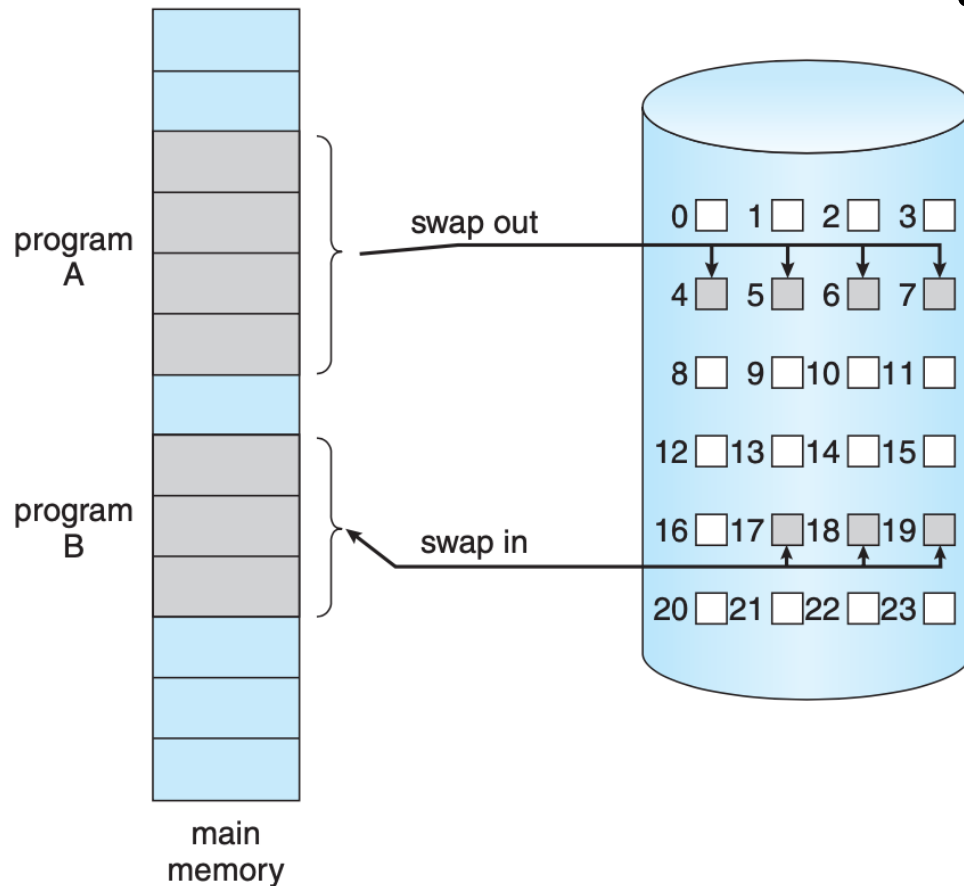


(also called as Dirty bit)

1: Page has been modified and must be written back in the secondary memory

0: Page has not been modified

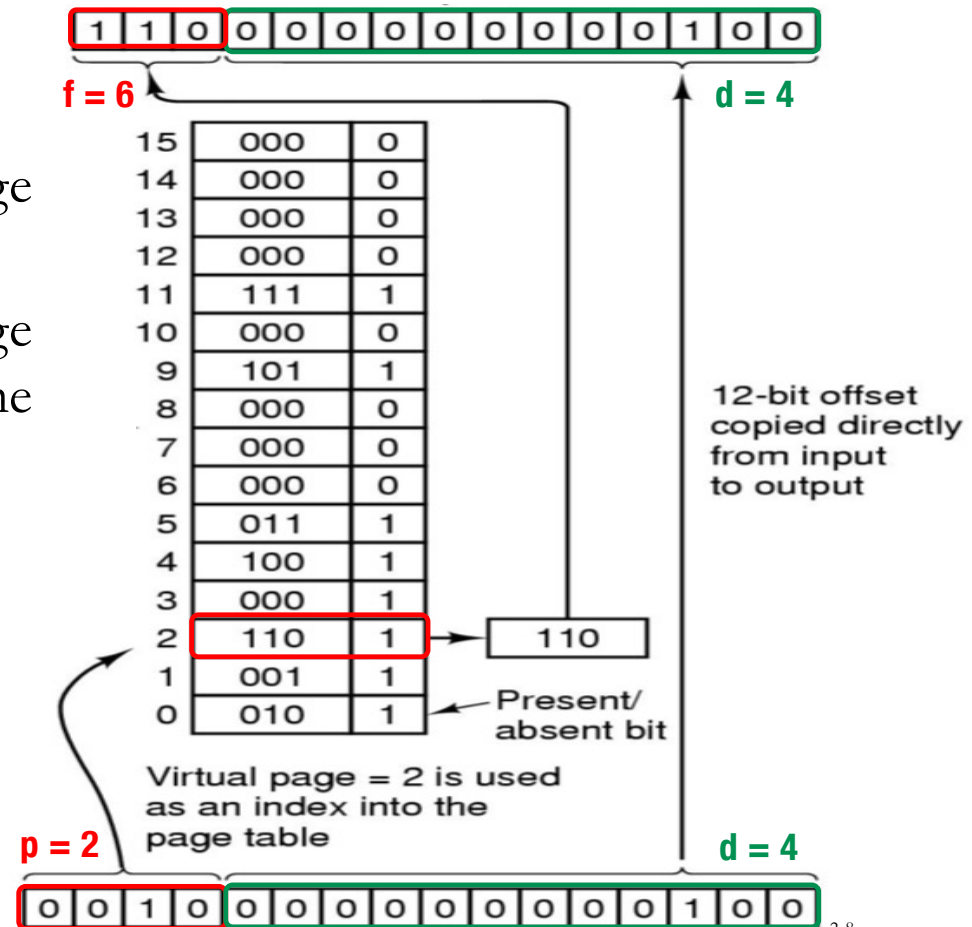
Demand Paging



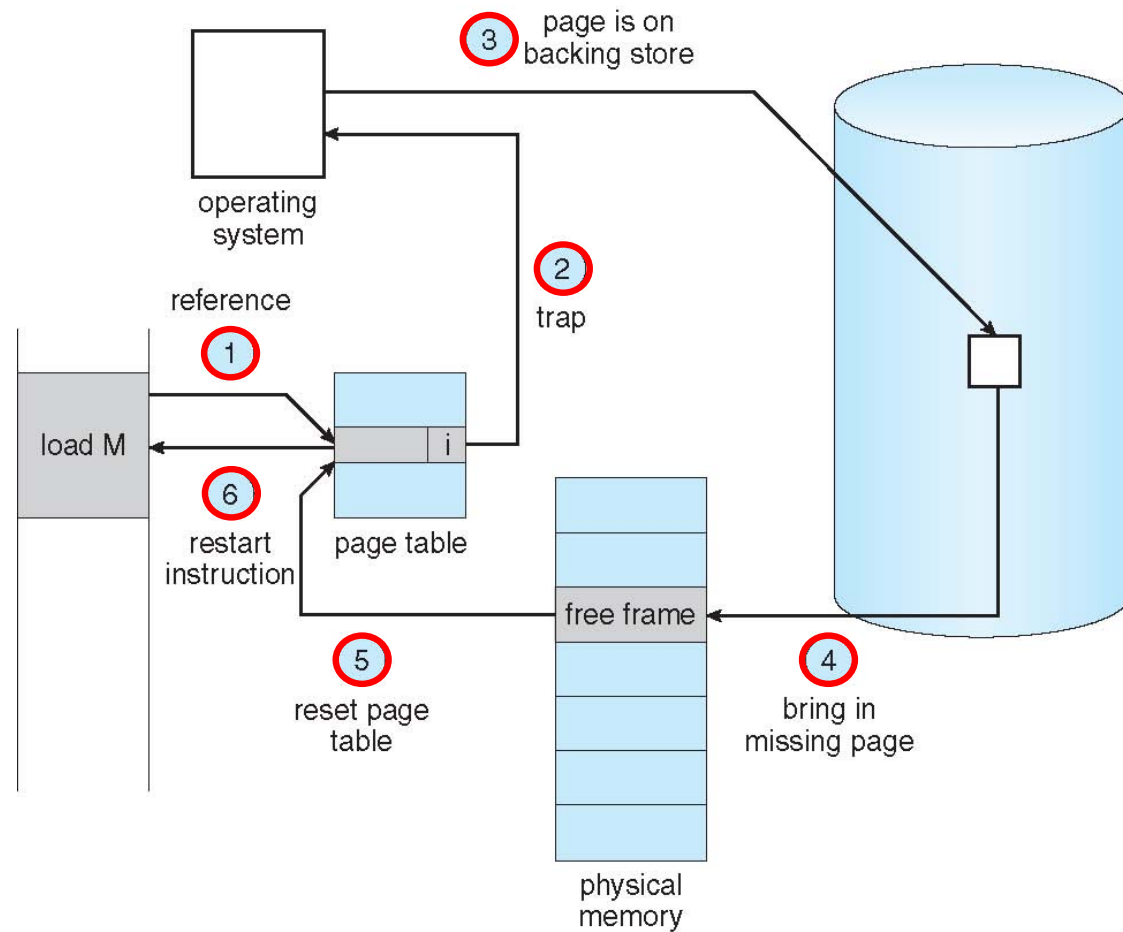
- Loading only the pages which are currently executing by the CPU into memory
 - Swapping zone (backup zone): a portion of hard disk used for containing all pages of active processes in the system

How to know if a page is presented or absent?

- Verify present/absent bit
 - 1 (or V): valid page (i.e., the page has been loaded into the memory)
 - 0 (or I): invalid page (i.e., the page is still placed in the swapping zone (fast disk))



Page Fault Handling



Effective Memory-Access Time (EAT or EMAT)

$$\text{EAT} = (1 - p) * t_m + p * t_p$$

p: page fault ratio (probability of occurring a page fault)

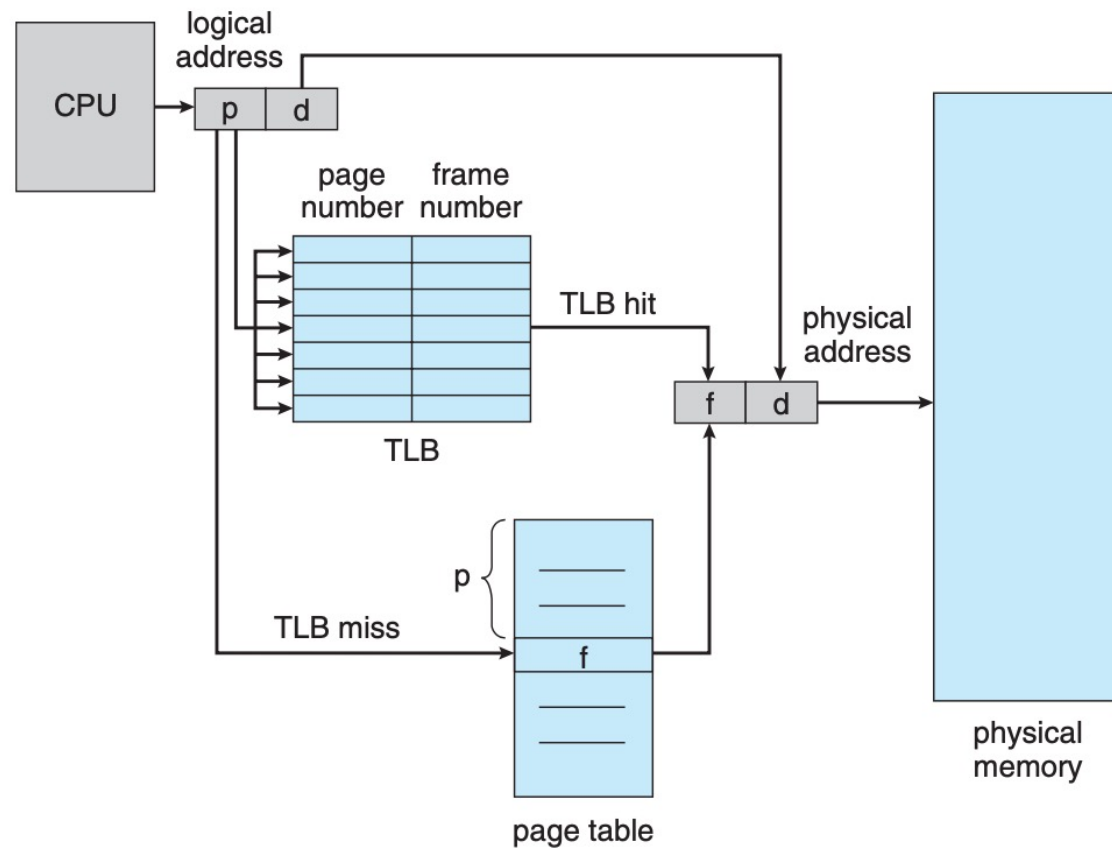
t_m: memory-access time

t_p: page-fault service time (swap-in, swap-out, restart instruction, ...)

Translation Lookaside Buffers (TLBs)

- **High-speed cache** memory for containing a **few of page table entries**, which have been most **recently or frequently used**
 - ✓ Number of entries generally ranges from 64 to 1024 entries
- Page lookup with TLBs
 - ✓ If page number is presented in TLBs (**TLB hit**)
 - No need to access the Page Table for address binding
 - ✓ If page number is not presented in TLBs (**TLB miss**)
 - Access Page Table for address binding
- Used to reduce Effective memory-access time (EAT)

Translation Lookaside Buffers (TLBs)



EAT (or EMAT) with TLBs support

$$EAT = h * (t_c + t_m) + (1 - h) * (t_c + 2 * t_m)$$

h: TLB hit ratio (probability of finding a desired page in TLB)

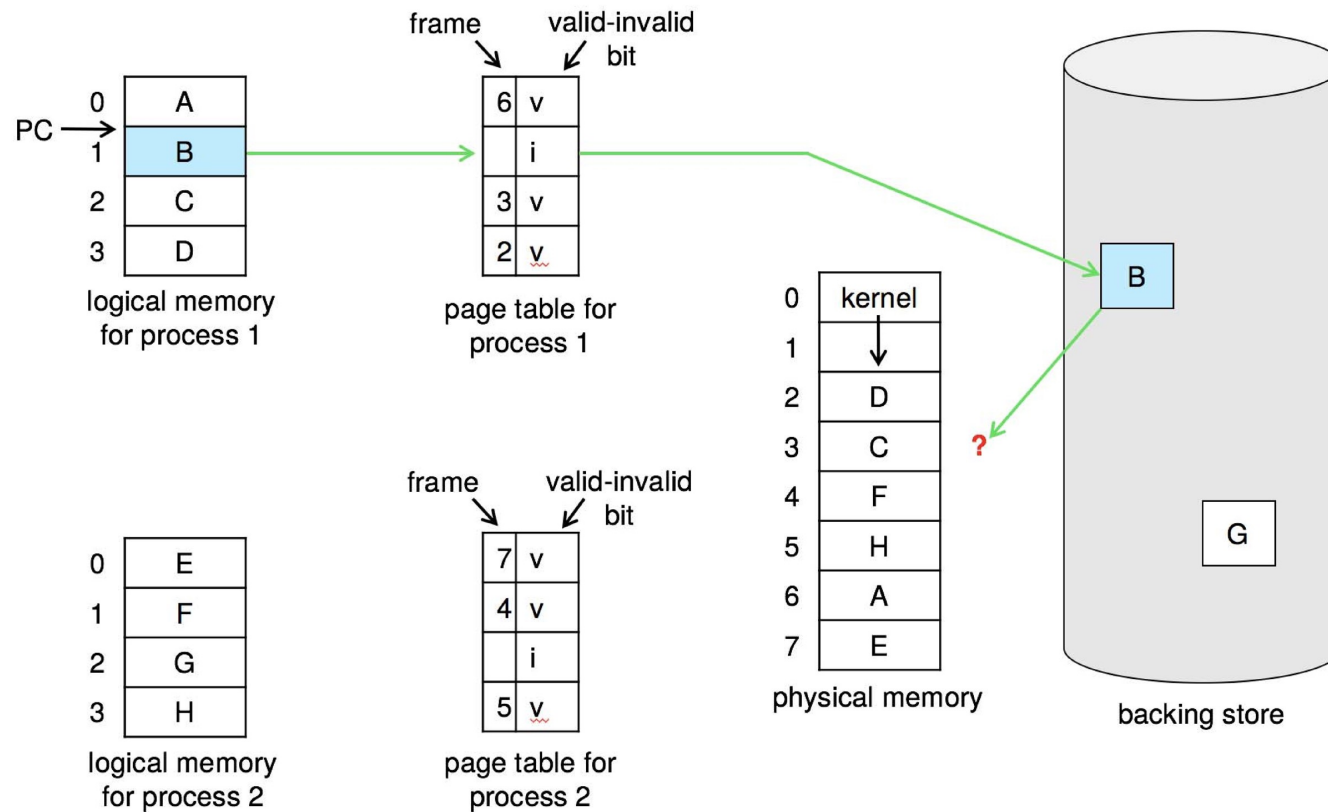
t_m: memory-access time

t_c: TLB lookup time

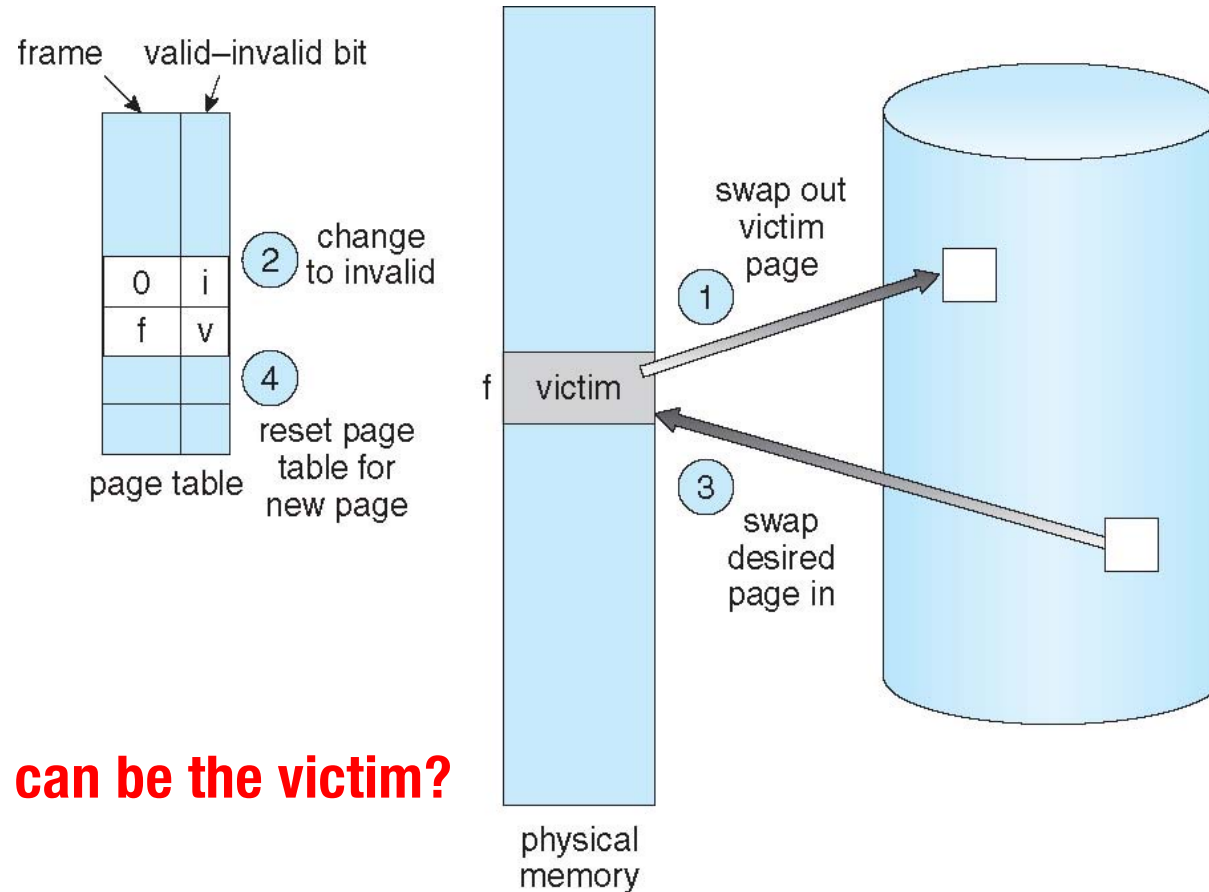
1 for Page Table access
1 for memory access



What if no free frames?



Page Replacement



Which page can be the victim?

Plan

- Fundamentals
- Segmentation
- **Paging**
 - *Principles*
 - ***Page Replacement Algorithms***
 - *Paging with Large Address Space*

FIFO

- Based on the principle of “First In First Out”
- ☺ Simplicity
- ☹ What if frequently referred pages or system pages moved out?
- ☹ Belady’s anomaly



reference string

15 page faults

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2	2	2	2	4	4	4	0	0	0	0	0	0	0	7	7	7
	0	0	0	0	3	3	3	2	2	2	2	2	1	1	1	1	1	0	0
		1	1	1	1	0	0	0	3	3	3	3	3	2	2	2	2	2	1
*	*	*	*		*	*	*	*	*	*			*	*			*	*	*

** : a page fault*

Paging | Page Replacement Algorithms

FIFO

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	4	4	4	5	5	5	5	5	5
	2	2	2	1	1	1	1	1	3	3	3
		3	3	3	2	2	2	2	2	4	4
*	*	*	*	*	*	*			*	*	

3 frames

→ 9 page faults

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	1	1	1	5	5	5	5	4	4
	2	2	2	2	2	2	1	1	1	1	5
		3	3	3	3	3	3	2	2	2	2
			4	4	4	4	4	4	3	3	3
*	*	*	*			*	*	*	*	*	*

4 frames

→ 10 page faults

Belady's anomaly

Frames  *Page faults* 

Optimal

- Look at the future, replace the page that will not be referred to for the longest time.

☺ Best performance

☹ How to know the future?

9 page faults

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2	2	2	2	2	2	2	2	2	2	2	2	2	2	7	7	7
	0	0	0	0	0	0	4	4	4	0	0	0	0	0	0	0	0	0	0
		1	1	1	3	3	3	3	3	3	3	3	1	1	1	1	1	1	1
*	*	*	*		*		*			*			*				*		

Least Recently Used (LRU)

- Look at the past, replace the page that has not been referred to for the longest time.

☺ Possible implementation

☹ Need hardware support

12 page faults

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2	2	2	2	4	4	4	0	0	0	1	1	1	1	1	1	1
	0	0	0	0	0	0	0	0	3	3	3	3	3	3	0	0	0	0	0
		1	1	1	3	3	3	2	2	2	2	2	2	2	2	2	7	7	7
*	*	*	*		*		*	*	*	*			*		*		*		

Second Chance

- Enhanced version of FIFO
 - A page will be given a second chance if its referenced bit is set to 1

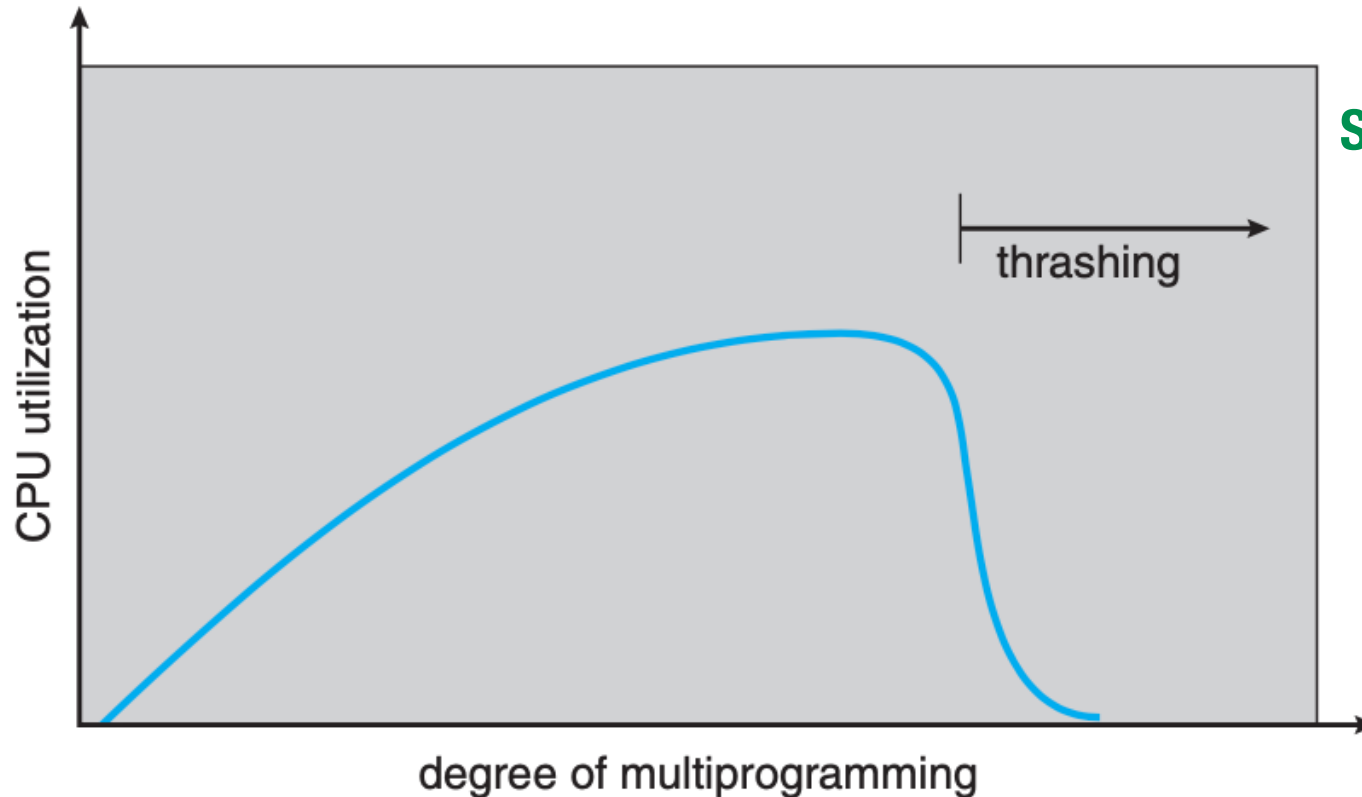
	7	0	1	2	0	3	0	4	2	3	0
Frame 1	7	7	7	2	2	2	2	4	4	4	4
Frame 2		0	0	0	0	0	0	0	2	2	2
Frame 3			1	1	1	3	3	3	3	3	0
Fifo	7	70	701	012	012	203	203	034	342	342	420
Referenced bit	7	70	701	2	20	23	230	4	42	423	0
	*	*	*	*		*		*	*		*

And other ...

- Not Recently Used (NRU)
- Not Frequently Used (NFU)
- WSClock
- ...

Issue: Thrashing

More time paging than executing because page faults occur at a very high rate.



Solution: Working Set

Plan

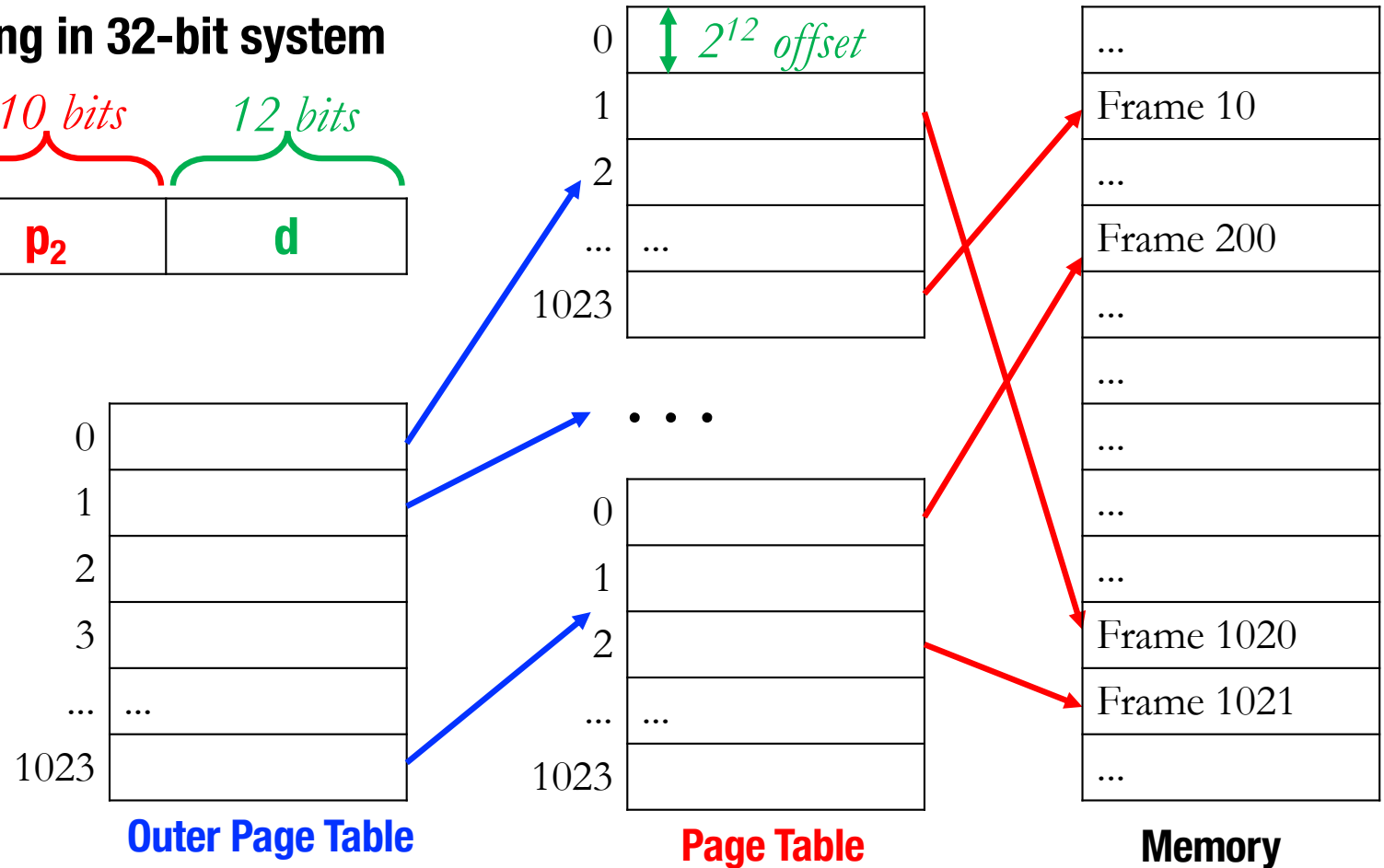
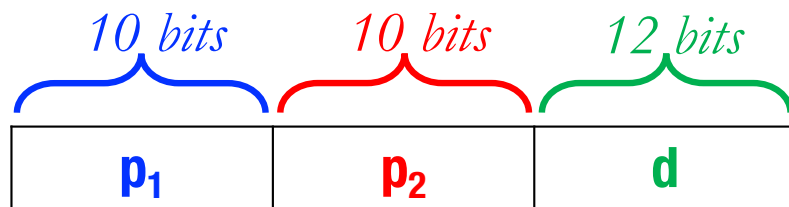
- Fundamentals
- Segmentation
- **Paging**
 - ❑ *Principles*
 - ❑ *Page Replacement Algorithms*
 - ❑ *Paging with Large Address Space*

Issue

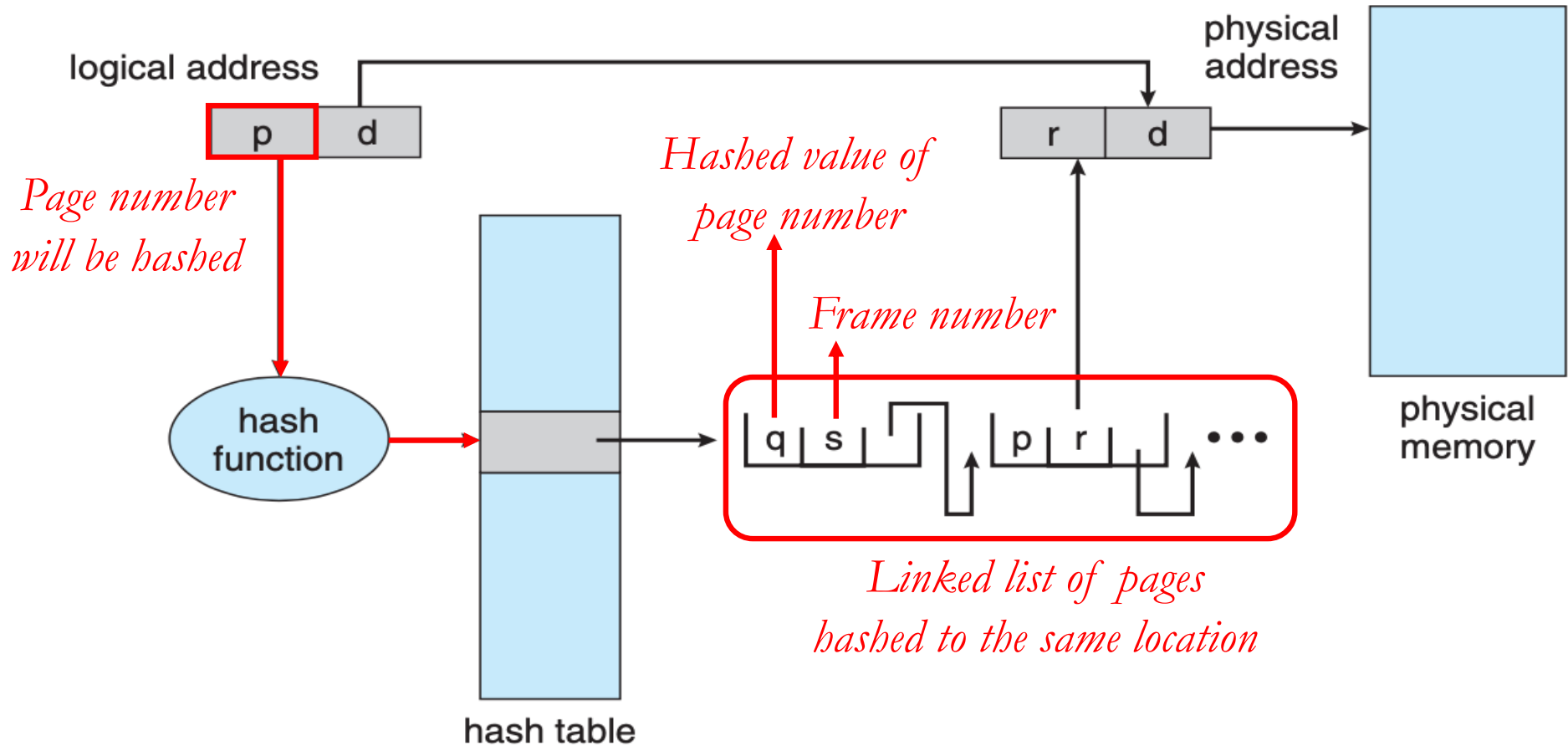
- Let's consider:
 - A system with a 64-bit logical address space
 - Page size: 4KB (2^{12})
 - No. of page table entries may be $2^{64-12} = 2^{52}$ entries
 - Assume page table entry size of 4 bytes
 - Page table size can be 2^{54} bytes \sim 16 million GB
 - Even in a 32-bit system, page table can have 1 million entries
- ☹ Loading huge page tables into memory?
- ☹ Lookup in a page table consisting of 1 million entries?

Hierarchical (or Multilevel) Paging

Two-level paging in 32-bit system



Hashed Page Table



Inverted Page Table

