Operating System

Nguyen Tri Thanh
ntthanh@vnu.edu.vn

Review

How many conditions for a dead lock to happen are there?

A. 2

B. 3

C. 4

D. 5

Review

Which are the conditions for a dead lock to happen?

A. circular wait, no-preemption, hold and wait, mutual exclusion
B. circular wait, preemption, hold and wait, mutual exclusion
C. circular wait, no-preemption, hold, mutual exclusion
D. circular wait, no-preemption, hold and wait, mutual wait

Review

Which is the most correct about deadlock prevention?

A. ensures the system will never enter a deadlock
B. ensures at least one of the four deadlock conditions will never occur
C. allows the system enter a deadlock and then recovers
D. detects the deadlock state and recovers

Review

Which is NOT a deadlock handling method?

A. deadlock avoidance
B. deadlock prevention
C. deadlock prediction
D. deadlock ignorance

Review

Which is the most correct about deadlock avoidance?

A. ensures the system will never enter a deadlock
B. ensures at least one of the four deadlock conditions will never occur
C. allows the system enter a deadlock and then recovers
D. ensures the circular wait condition will never occur

Review

Which is the most correct about safe state?

A. the state of a process
B. the state of the system
C. the running sequence (order) of processes that ensures the system does not enter a deadlock
D. the state that ensures a process can safely run

Review

Which is correct about deadlock?

A. a deadlock will surely occur if the system is in unsafe state
B. a deadlock may occur even when the system is in safe state
C. there is only one method for handling deadlock
D. deadlock handling is available in all OSes

Review

Which is the correct method for recovering from a deadlock?

A. restart the system
B. abort each process involved in the deadlock until the deadlock disappears
C. provide more resources for the system
D. ignore the deadlock

Review

Which is the criteria for selecting a process to abort when a deadlock occurs?

A. the number of processes in the system
B. the resources a process needs to complete
C. the available resources the system has
D. the available RAM

Memory management

Objectives

- Introduce what swap is
- Introduce two contiguous memory allocation
- Introduce paging method
- Introduce segmentation method
- Implement memory allocation methods

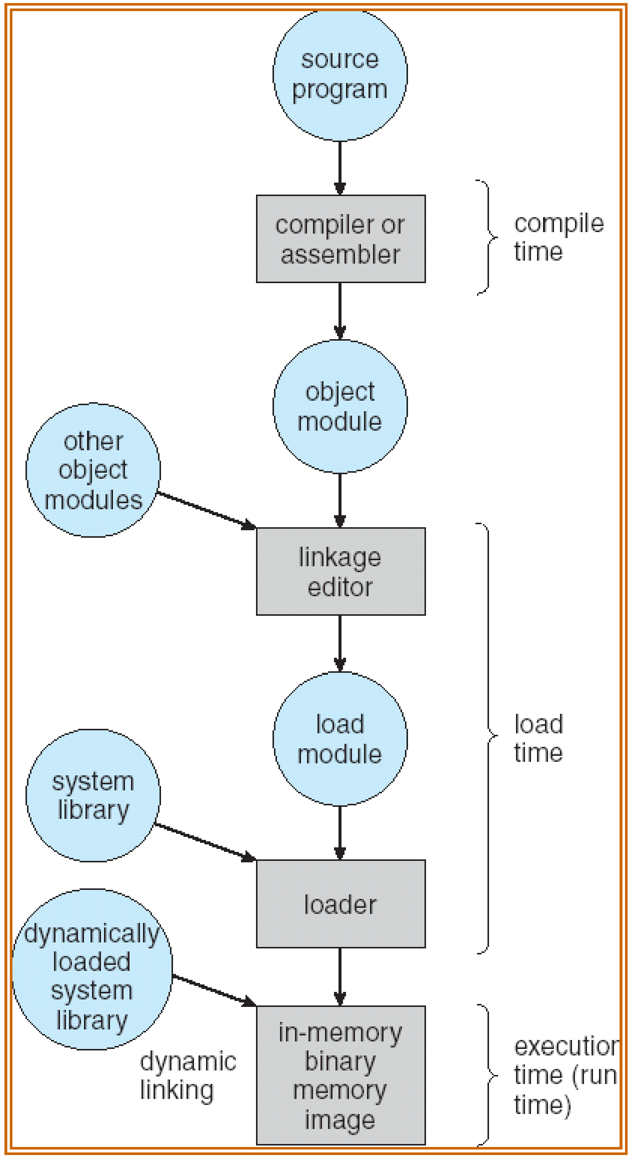
Reference

- Chapter 8 of Operating System Concepts

Introduction

- *Input queue* – a queue of programs on disks waiting to be run
- A program is loaded into memory (by OS) and then is executed
- A program has to be through some steps before being executed

Multistep processing of a user program



Contiguous allocation

MFT

MVT

Contiguous allocation

- Main memory usually into two partitions
  - Resident operating system, usually held in low memory with interrupt vector
  - User processes then held in high memory
- Relocation registers used to protect user processes from each other, and from changing operating-system code and data
  - Base register contains value of smallest physical address
  - Limit register contains range of logical addresses – each logical address must be less than the limit register
  - MMU maps logical address *dynamically*

Multiprogramming with a Fixed number of Tasks (MFT)

- Multiple-partition allocation
  - MEM is statically split into a number of partitions
  - When a process arrives, it is allocated memory from a hole large enough to accommodate it
    - process is limited by the size of partition to run
  - Hole – block of available memory; holes of various size are scattered throughout memory
  - Operating system maintains information about: a) allocated partitions b) free partitions
- System uses MFT: IBM/360

IBM/360

IBM 360 Computer

IBM System 360, Model 30, Memory size: up to 64K bytes, 1965
(3 times faster processor and memory access speed than the 1050)

Question

Suppose a system

- uses MFT with n MEM partitions
- has m programs waiting ($m>n$)

Which of the following is incorrect?

A. maximum n processes can run at the same time

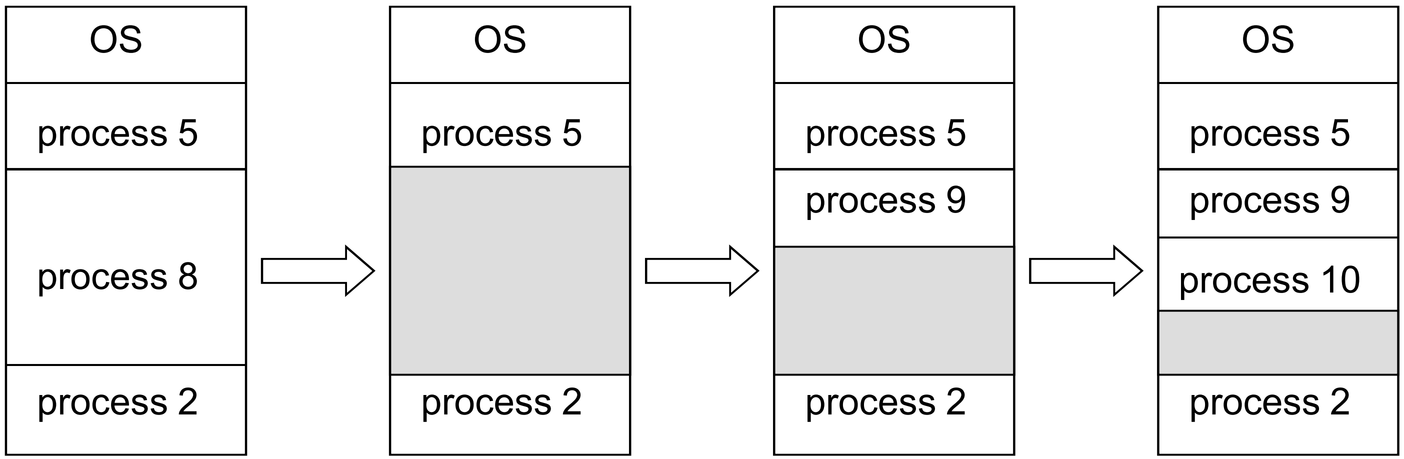
B. the whole process is in one partition

C. a process can reside in one or more consecutive slots

D. the number of running processes is less than n even when there are some free slots in some cases

Multiprogramming with a Variable number of Tasks (MVT)

- Mem is not statically split into partitions
  - A process is allocated a partition large enough to run
  - System manages the list of free/allocated partitions
  - *Hole* – small free partition; scattered in MEM



Question

- Which of the following is incorrect about MVT?
  A. it supports multiprogramming systems
  B. the whole process is in a contiguous range
  C. the process can be run providing that its size is less than the MEM size
  D. MEM is previously split into partitions

Dynamic MEM Allocation

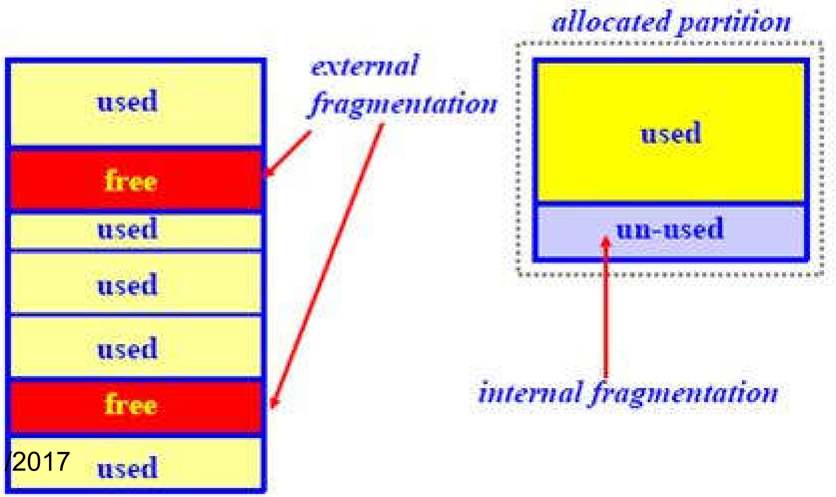
- How to satisfy a request of size *n* from a list of free holes
  - **First-fit**: Allocate the *first* hole that is big enough
  - **Best-fit**: Allocate the *smallest,* big enough hole
    - must search entire list, unless ordered by size
    - Produces the smallest leftover hole
  - **Worst-fit**: Allocate the *largest* hole;
    - must also search entire list
- First-fit and best-fit are better than worst-fit

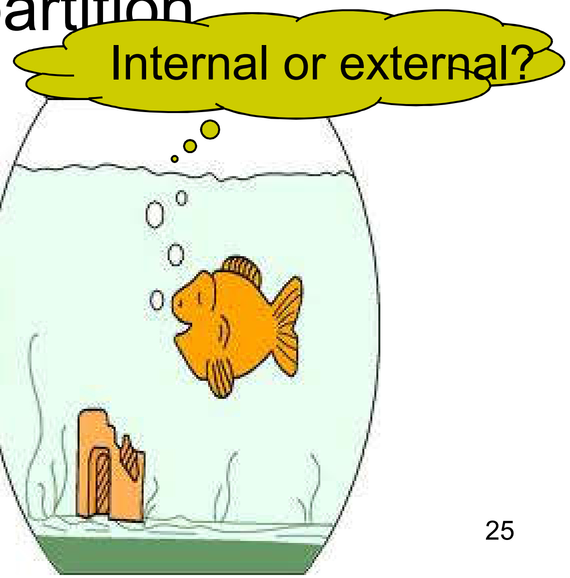
Question

- Which of the following is incorrect about dynamic allocation algorithms?
  A. these algorithms are only used in MFT
  B. the algorithm is called whenever a process requests to run
  C. the purpose of the algorithm is to find a suitable partition to load the process into
  D. some algorithms don’t have to search the whole list

Fragmentation

- **External Fragmentation**
  - total memory space exists to satisfy a request, but it is not contiguous
- **Internal Fragmentation**
  - allocated memory is larger than requested memory
  - the unused memory is internal to a partition





Fragmentation (cont'd)

- Reduce external fragmentation by **compaction**
  - Shuffle memory contents to place all free memory together in one large block
  - Compaction is possible *only* if relocation is dynamic, and is done at execution time
  - I/O problem
    - Latch job in memory while it is involved in I/O
    - Do I/O only into OS buffers

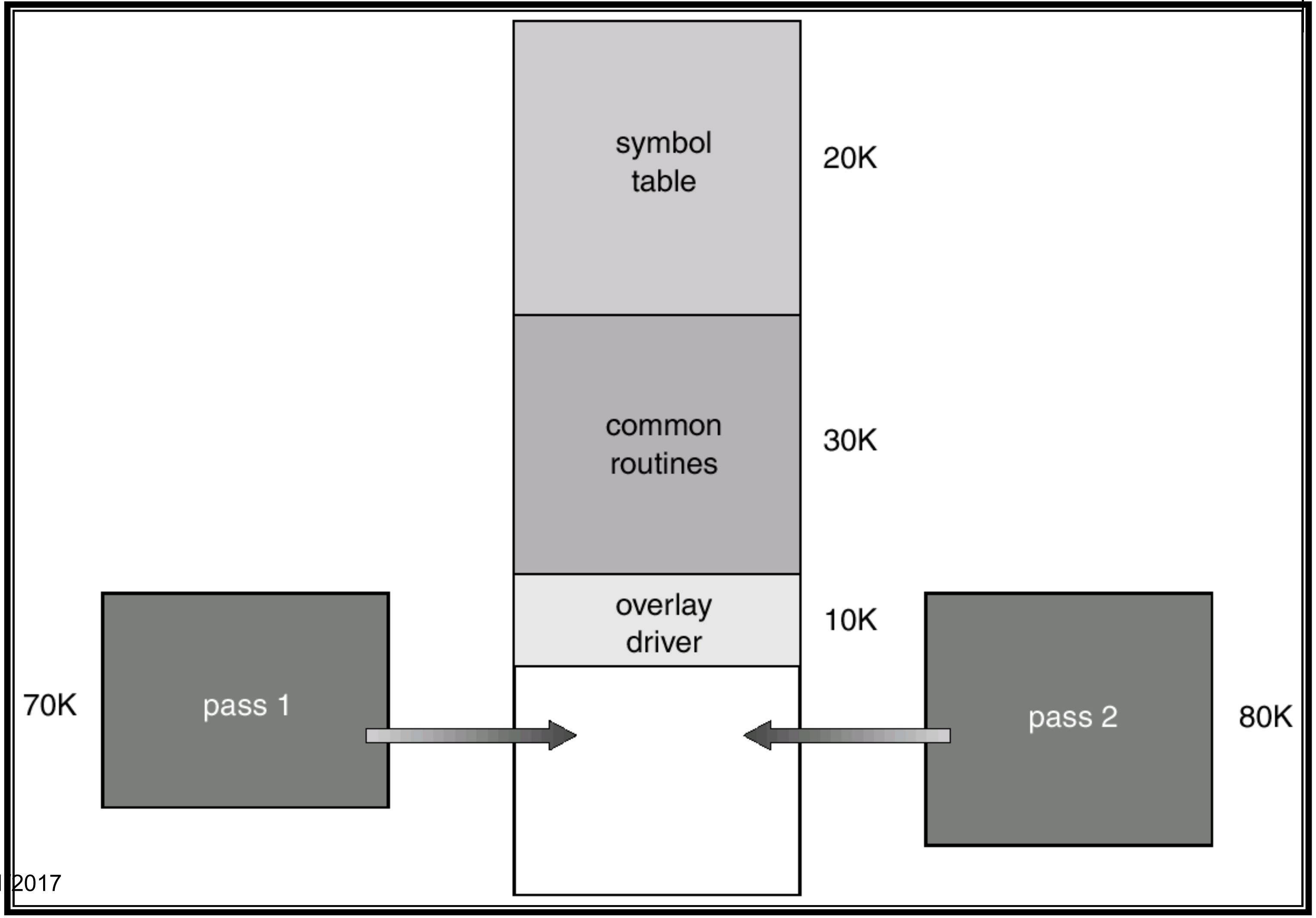
Question

- Which of the following is incorrect about fragmentation?
  A. it may lead to a situation where a process cannot run, even though the total free MEM is larger than the process size
  B. it only happens in MEM
  C. there are two types of fragmentation
  D. fragmentation results in ineffective use of MEM

Overlays

- Only keep in MEM needed data and instructions during execution
- Need support from the programming language and programmers
  - old method supported in PASCAL
- Used when a process requests memory larger than allocated size

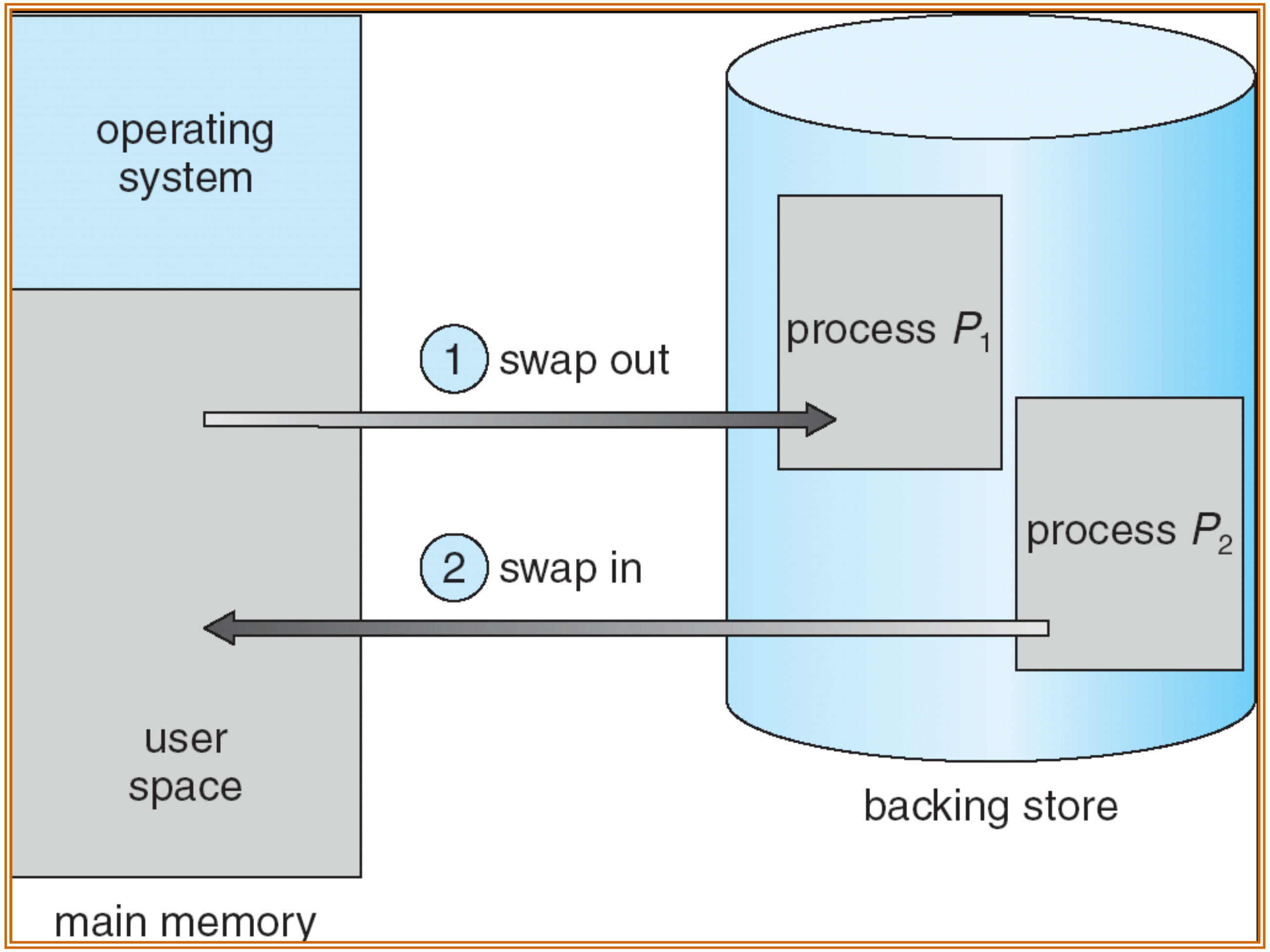
Overlays example



Swapping

- *Swapping*
  - temporarily save a process into *backing store*, and restore it when possible
  - *backing store* – a space on disk, large enough to store many user programs (Swap in Linux, pagefile.sys in Windows)
- *Roll out, roll in*
  - swap method for priority scheduling
  - low priority process is *rolled out*, higher one is *rolled in* to be executed
- swap time is proportional to the size of processes
- UNIX, Linux, and Windows use swapping for pages

Schematic View of Swapping



Question

What is incorrect about overlays?

A. overlays allows a large program to run in a smaller MEM
B. Overlays only loads codes on demand (when they are used)
C. Programmers need to split the program into modules
D. Overlays is supported in all high level programming languages

Question

What is incorrect about swapping?

A. swapping is the same as overlays
B. swapping uses hard disk as the *backing store*
C. swapping allows many processes whose total size is larger than MEM to run
D. a lower priority process is rolled out for a higher priority one to run (when needed)

Question

What is incorrect about overlays and swapping?

A. both are the solution to utilize memory more effectively
B. both roll out the same object
C. swapping is a special case of overlays where the object is rolled out is the whole process
D. both use hard disk for temporary storage

Logical vs. Physical Address Space

- There are two address space
  - *Logical address* – generated by CPU;
    - Also called *virtual address space*
  - *Physical address* – generated by memory management unit (MMU)
    - also called *real physical address space*
- Logical and physical addresses are
  - the same in “compile-time” and “load-time”
  - different in “execution-time”

Logical vs Physical address

Virtual address space *Physical address space*

ĐẠI HỌC QUỐC GIA HÀ NỘI CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

DANH SÁCH SIN

Môn học: **Nguyên lý hệ** Mã lớp môn học:

Thứ - Tiết: **4, 6 - 8**

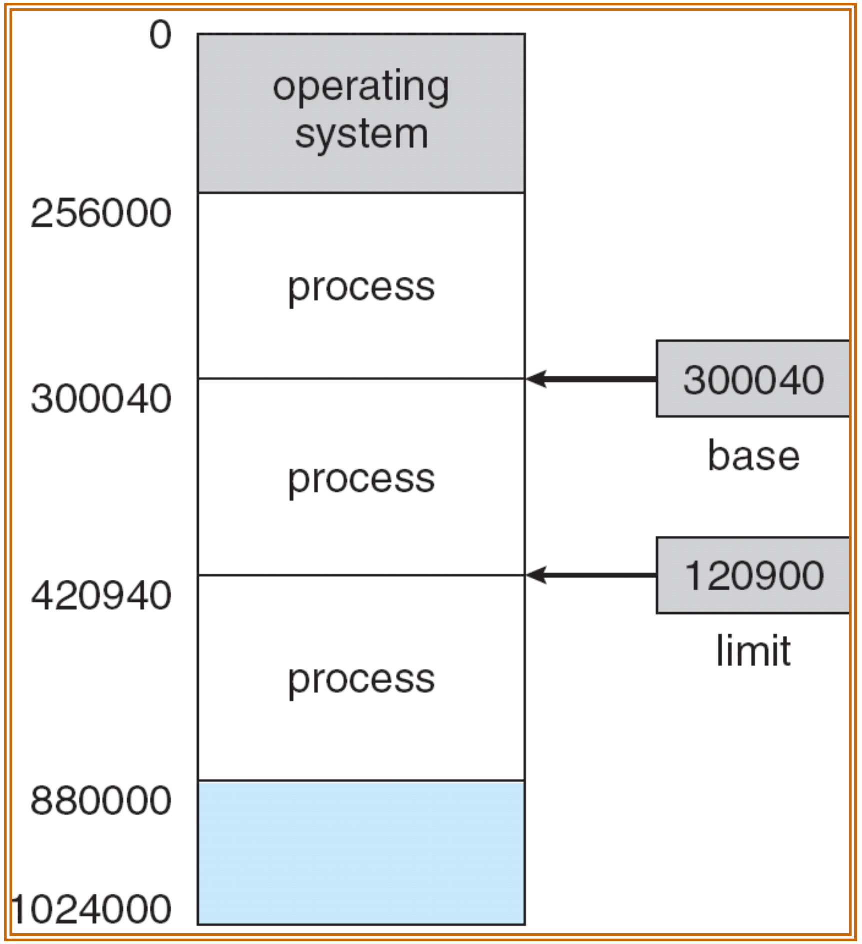
STT	Mã SV	Họ và tên			
1	10020007	Nguyễn Công Anh			
2	10020010	Nguyễn Thế Anh			
3	10020446	Nguyễn Thị Ánh			
4	10020020	Phạm Đức Bình			
5	10020449	Phan Văn Chương			
6	9020076	Nguyễn Lưu Cường			
7	10020041	Nguyễn Văn Cường			
8	10020042	Nguyễn Văn Cường			
9	10020044	Phạm Văn Cường			
10	10020047	Trần Minh Diện			
11	10020052	Nguyễn Thị Thuỳ Du			
12	10020067	Đỗ Hoàng Dương			
13	10020073	Phan Văn Đại	21/10/1992	QH-2010-I/CQ-C-C	
14	10020081	Nguyễn Tiến Đạt	23/07/1992	QH-2010-I/CQ-C-C	

page not belonging to process

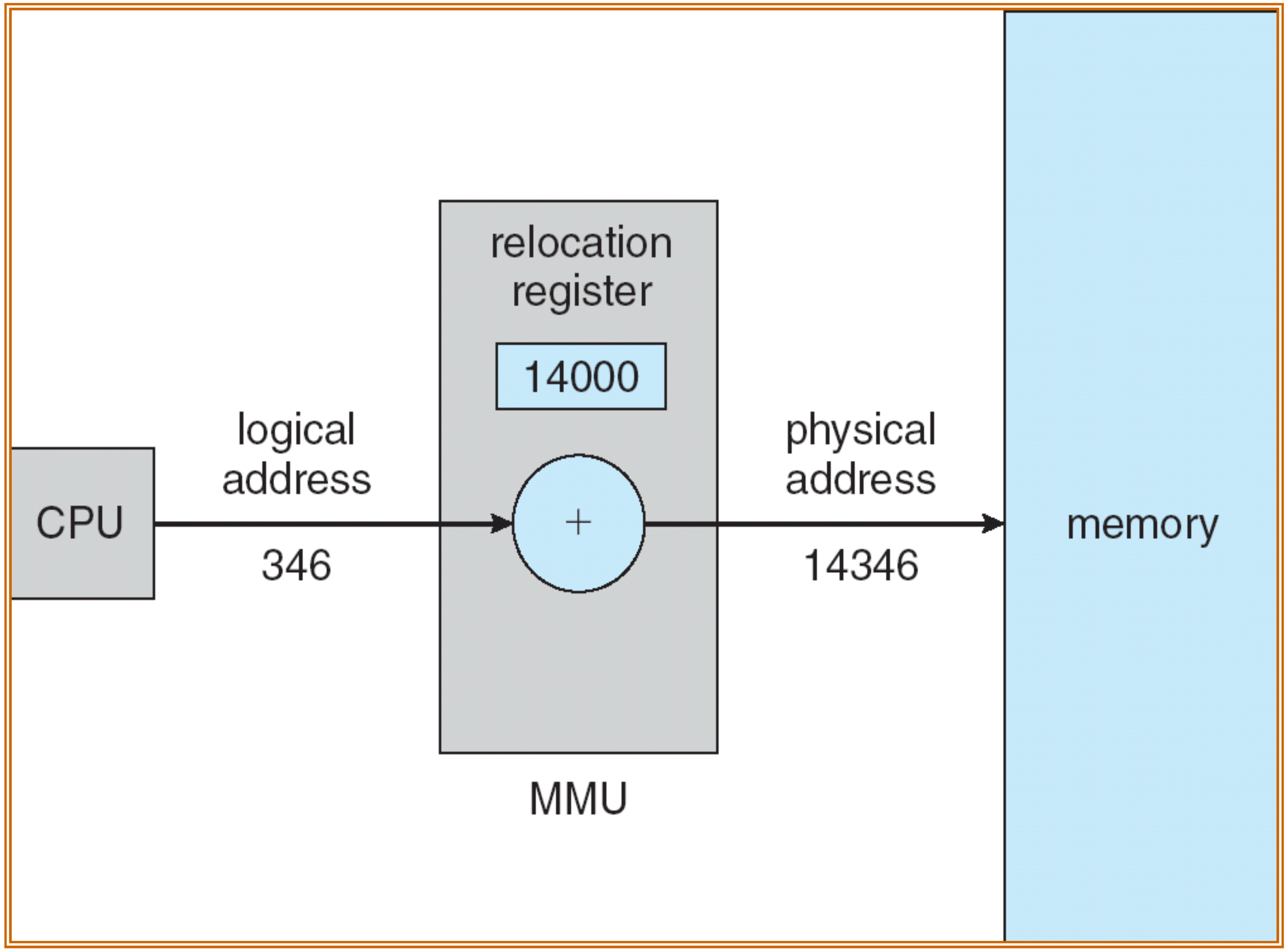
Memory Management Unit (MMU)

- Hardware device that maps virtual to physical address
- In MMU
  - the value in the relocation register is added to every address generated by a user process at the time it is sent to memory
- The user program deals with *logical* addresses
  - it never sees the *real* physical addresses

Address translation



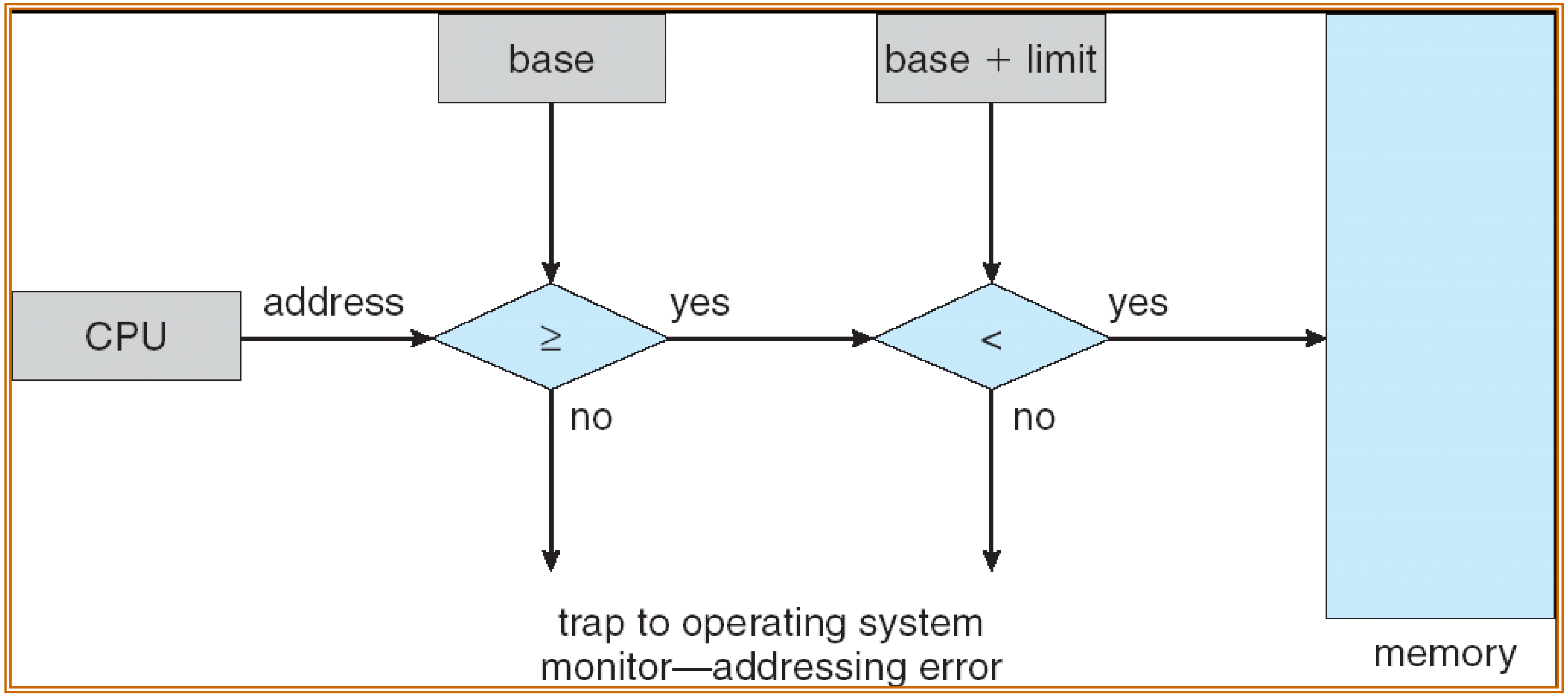
Dynamic relocation using a relocation register



Question

- What is the mission of MMU?
  - A. Allocate memory for a process
  - B. Load a process into memory
  - C. Map a logical address into a physical address
  - D. Map a physical address into a logical address

HW address protection with base and limit registers



Question

Suppose

- the base address is 13400
- the limit register is 1200
- the reference is 246;

Which of the following is the correct physical address of the reference ?

A. 13646
B. 1446
C. 13154
D. 954

Question

- Which of the following is incorrect about address protection hardware ?
  A. it checks the validity of an address
  B. it raises an error if the address is invalid
  C. there are two tests of a given address
  D. it returns 0 if the address is invalid

Question

Suppose

- the base address is 13400
- the limit register is 1200
- the reference is 1246;

Which of the following is the correct physical address of the reference ?

A. 14646

B. 2446

C. 11154

D. invalid reference

Non-contiguous allocation

Paging

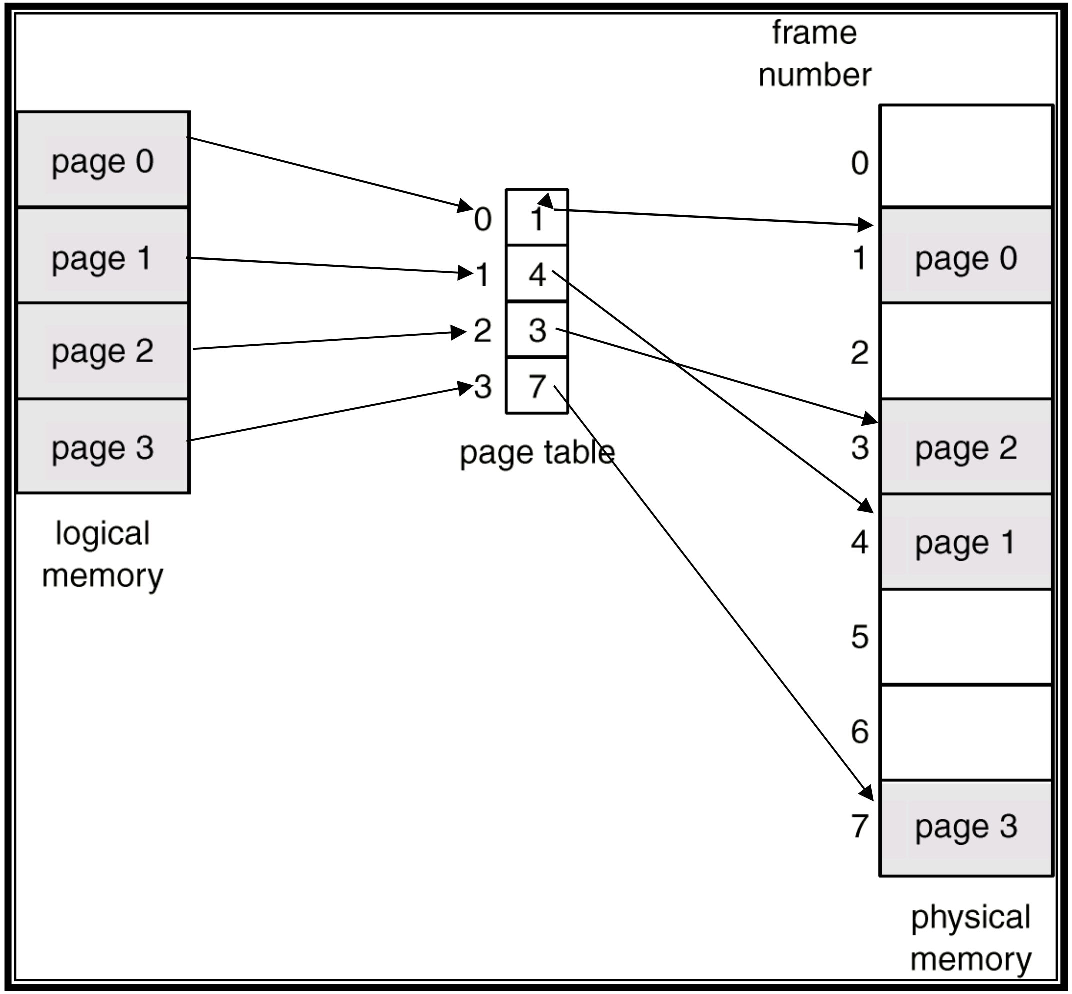
Introduction

- Divide physical memory into fixed-sized blocks called **frames**
  - size is power of 2, e.g., 512, 1024, or 8,192 bytes
- Divide logical memory into blocks of same size called **pages**
- Physical address space of a process can be noncontiguous

Introduction (cont'd)

- Keep track of all free frames
- To run a program of size *n* pages, need to find *n* free frames and load program
- Set up a page table to translate logical to physical addresses
- Have Internal fragmentation
- Have a mapping from pages → frames called *page table*
  - each process has a copy of page table

Paging example



Address Translation Scheme

- Address generated by CPU is divided into
  - *Page number - p*
    - used as an index into a *page table* which contains base address of each page in physical memory
  - *Page offset - d*
    - combined with base address to define the physical memory address that is sent to the memory unit
- Address register of *m* bits
  - *m-n* high bits are used for *page number*
  - *n* lower bits are offset

Address Translation Scheme (cont'd)

- Address register

page number	page offset
p	*d*
m - n	*n*

- No external fragmentation but internal fragmentation
- What is the effect of page size (large/small)?
  - Decrease page size → internal fragmentation is reduced → performance is decreased
  - Increase page size → performance is increased → internal fragmentation is increased

Question

Which of the following is incorrect about paging?

A. it is a contiguous memory allocation method
B. a process' virtual address is divided into pages
C. pages of a virtual address are of the same size
D. the page size is the same as the frame size

Question

A system uses paging

- the frame size of 4KB;
- the address register is 32 bits

Which of the following is correct about register segmentation?

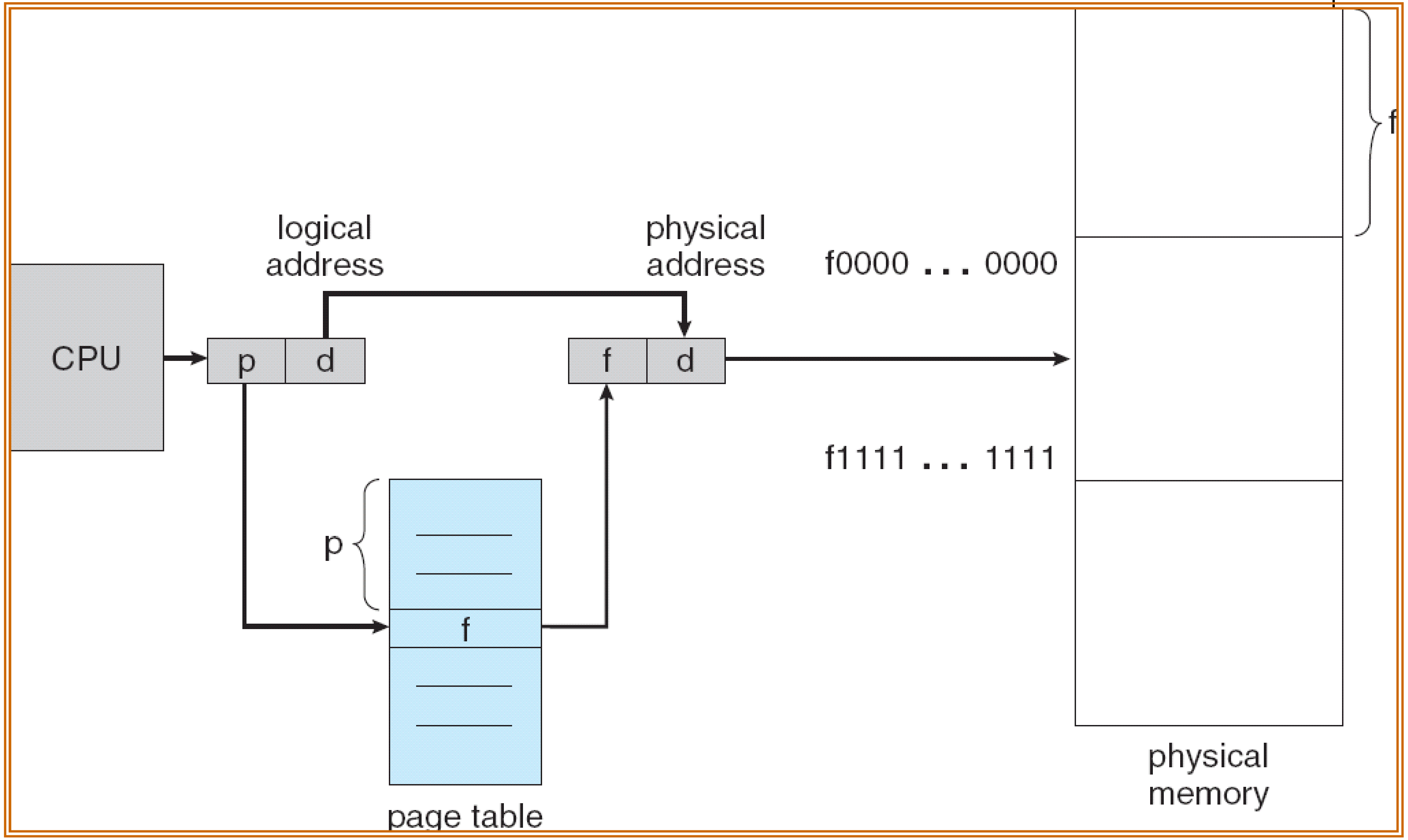
A. (page:offset) = (19:13)

B. (page:offset) = (21:11)

C. (page:offset) = (22:10)

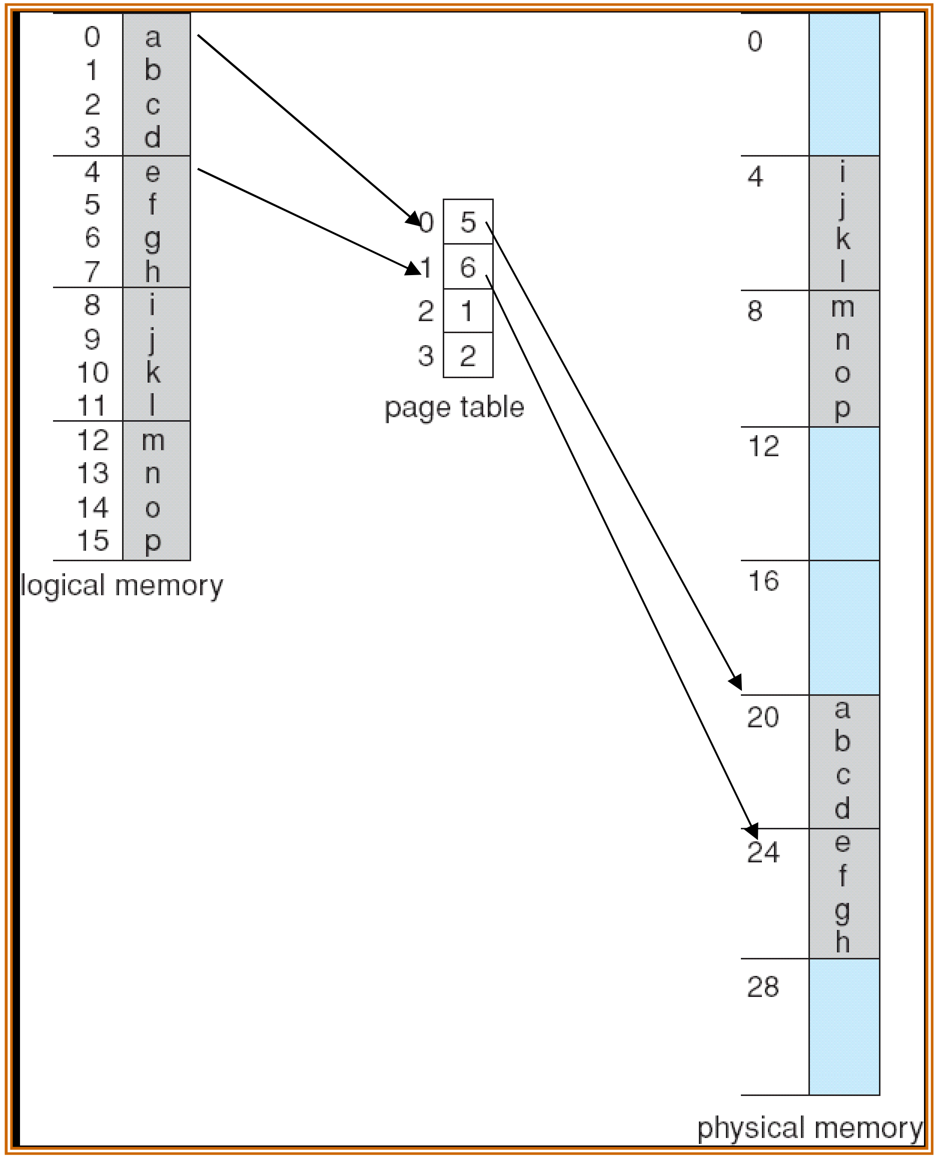
D. (page:offset) = (20:12)

Address translation



Paging example

page size is 4 bytes



Question

Frame
56
120
3

A system uses paging

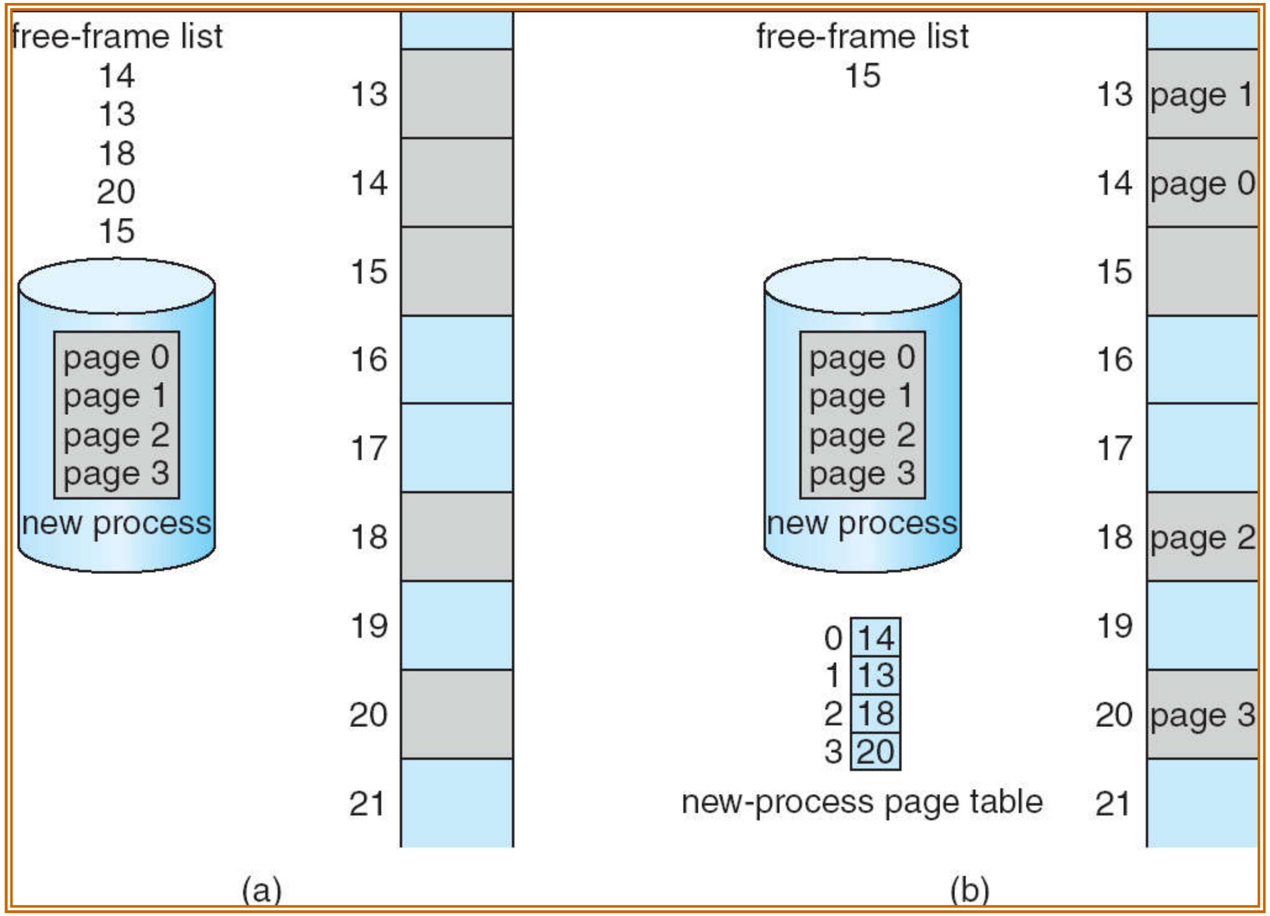
- the frame size of 4KB;
- the address register is 32 bits

- Which of the following is the correct physical address of the reference (1,1296)?
  A. 560*4096+1296
  B. 120*4096+1296
  C. 3*4096+1296
  D. 120*1024+1296

Question

- Which of the following is incorrect about address translation in paging?
    A. an reference has the form of (p,d)
    B. a page table is needed for address translation
    C. the physical address is $f*2^n+d$, where f is the corresponding frame of p, 2^n is the frame size
    D. the physical address is $p*2^n+d$, where 2^n is the frame size

Free frame list



Before allcation

After allocation

Implementation of Page Table

- Page table is kept in main memory
- **Page-table base register (PTBR)** points to the page table
- **Page-table length register (PRLR)** indicates the size of the page table
- In this scheme every data/instruction access requires two memory accesses.
  - one for the page table and
  - one for the data/instruction.

Implementation of Page Table (cont'd)

- The two memory access problem can be solved by
  - use of a special fast-lookup hardware cache called **associative memory**
  - or **translation look-aside buffers (TLBs)**
- Some TLBs store **address-space identifiers (ASIDs)** in each TLB entry
  - uniquely identifies each process
  - to provide address-space protection for process

Implementation of Page Table (cont'd)

- Each TLB entry includes
  - key: identifies a page number of a process
  - value: the corresponding frame of the page
- The number of entries in TLB
  - from 64 to1024

Associative Memory

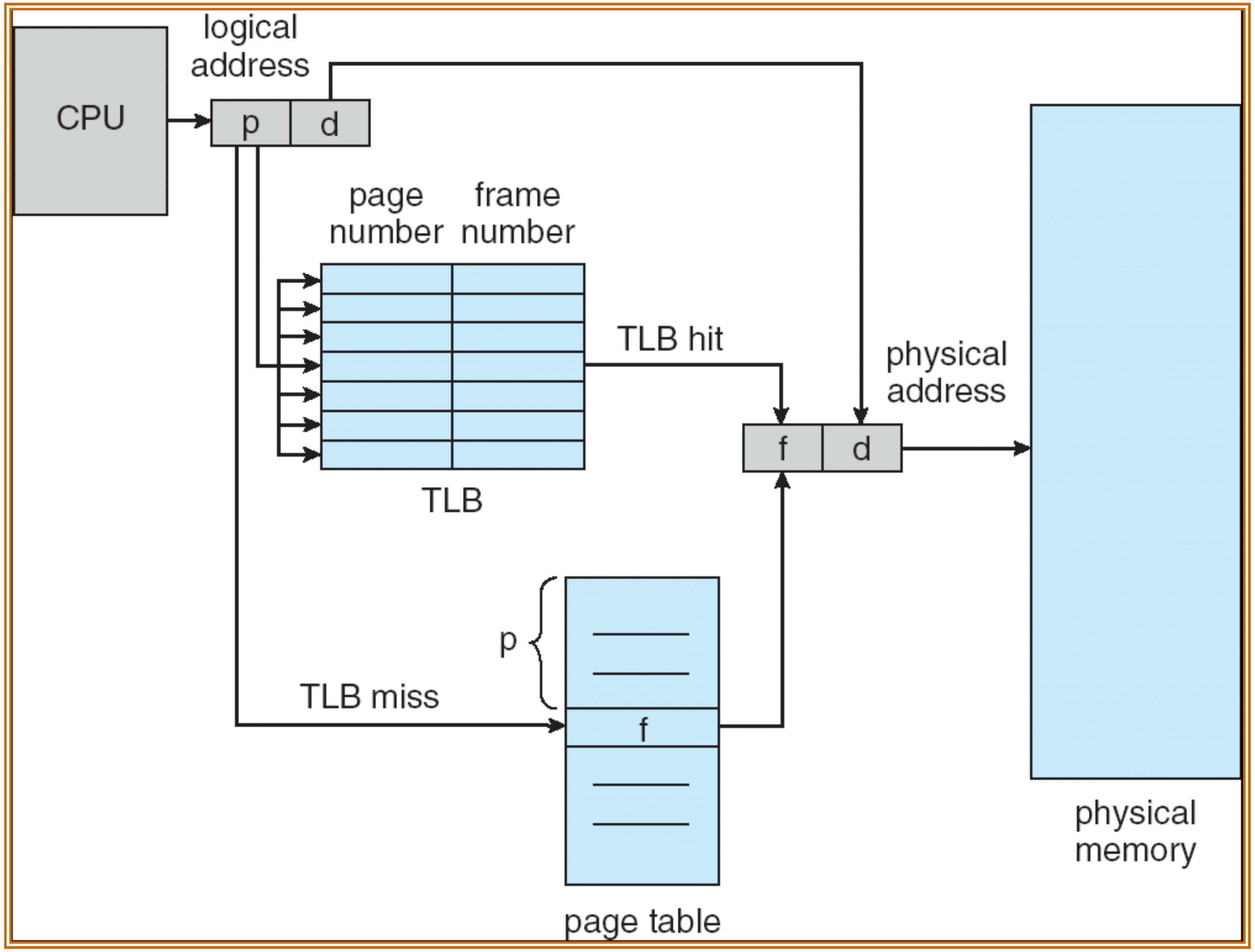
- Associative memory – parallel search

Page #	Frame #

Address translation (*p*, *d*)

- If p is in associative register, get frame # out
- Otherwise get frame # from page table in memory

Paging Hardware With TLB



Effective Access Time

- Associative Lookup = ε time unit
- Assume memory access time is β
- Hit ratio = α
  - percentage of times that a page number is found in the associative registers;
  - ratio related to number of associative registers
- **Effective Access Time** (EAT)

$$EAT = (\beta + \varepsilon)\,\alpha + (2\,\beta + \varepsilon)(1 - \alpha) = 2\beta + \varepsilon - \alpha\beta$$

Question

- Which of the following is incorrect about paging with a TBL?
  A. a reference has the form of (p,d)
  B. a memory reference takes at least two steps to access the physical address
  C. the physical address is $f*2^n+d$, where f is the corresponding frame of p
  D. a physical access always takes 3 steps

Question

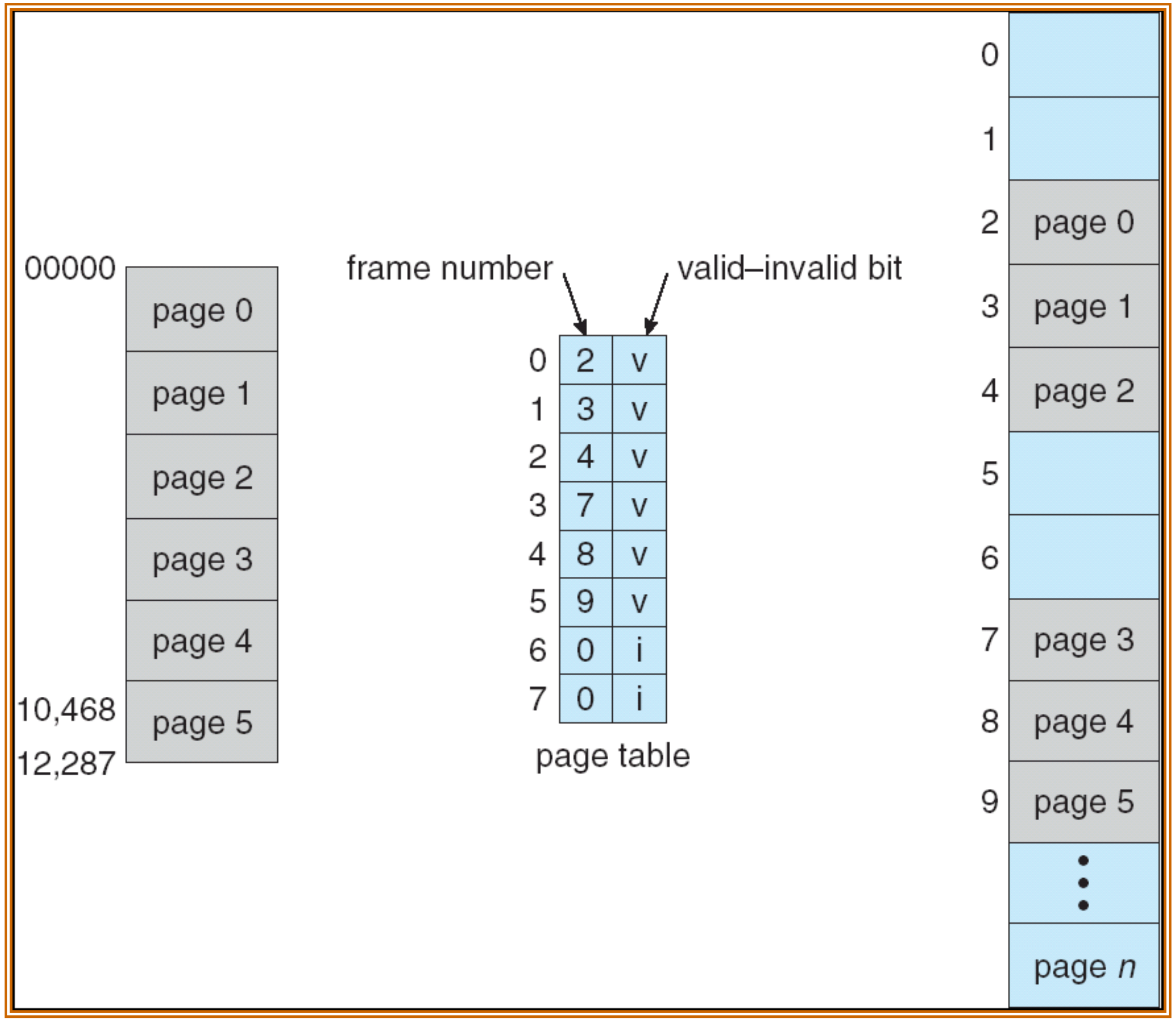
- Suppose a system uses paging with a TLB
  - memory access time is 200 nanoseconds
  - TBL access time is 10 nanoseconds
  - hit rate is 80%
- What is the EAT of the system?

A. 250 nanoseconds
B. 260 nanoseconds
C. 280 nanoseconds
D. 220 nanoseconds

Memory Protection

- Memory protection implemented by associating protection bit with each entry
  - **Valid-invalid** bit attached to each entry in the page table:
  - "valid" indicates that the associated page is in the process' logical address space, and is thus a legal page
  - "invalid" indicates that the page is not in the process' logical address space

Valid (v) or Invalid (i) Bit In A Page Table



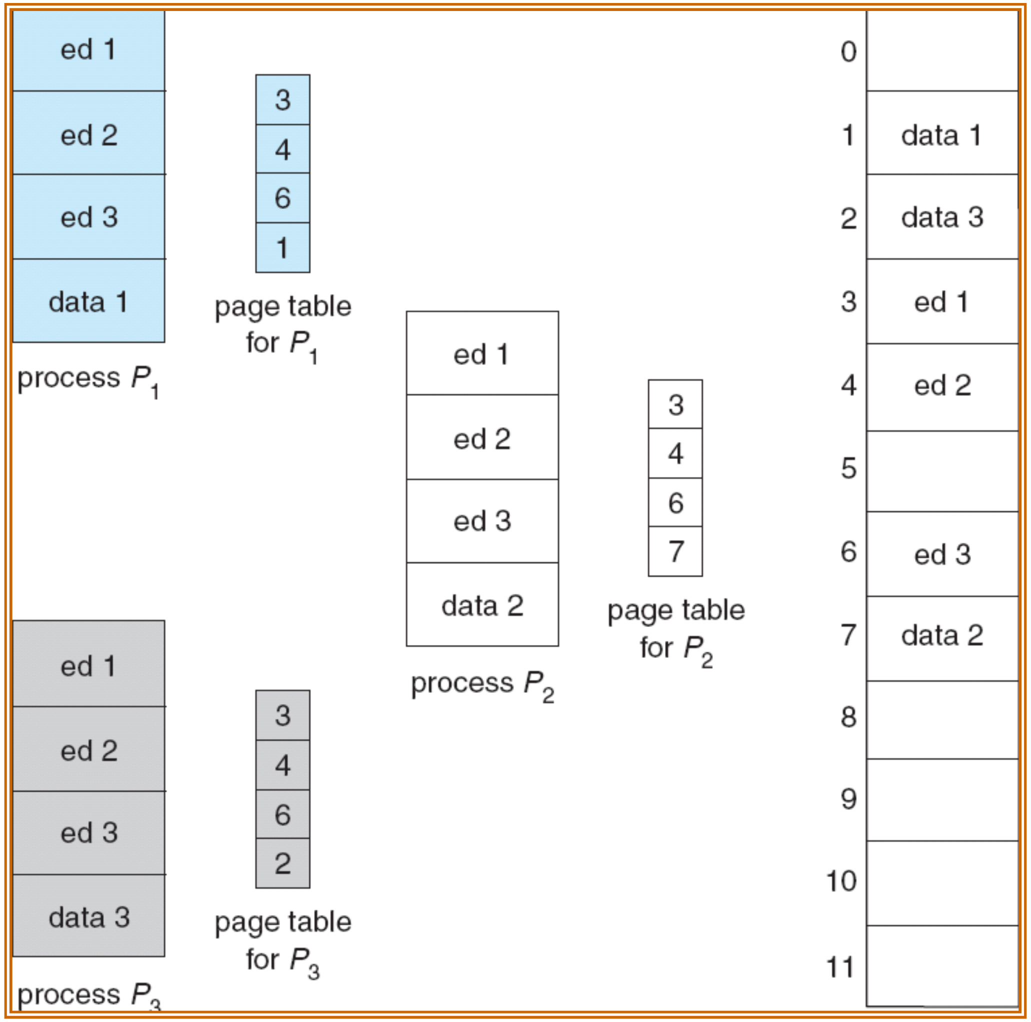
Question

- Which of the following is incorrect about memory protection in a page table?
  A. to make sure the process refers to a position within MEM
  B. to prevent a process from referring to an invalid physical address
  C. to protect a process from referring to an position out of its address space
  D. to make sure the process refers to a position within its address space

Shared Pages

- **Shared code**
  - One copy of read-only (reentrant) code shared among processes (i.e., text editors, compilers, window systems).
  - Shared code must appear in same location in the logical address space of all processes
  - refer to Sect. 3, Chapter 7 of “Lập trình C/C++ trên Linux”
- **Private code and data**
  - Each process keeps a separate copy of the code and data
  - The pages for the private code and data can appear anywhere in the logical address space

Shared Pages Example



Question

- Which of the following is incorrect about shared pages
  A. they are supported by all operating systems
  B. they help to utilize MEM more effectively
  C. they are normally libraries
  D. many processes use the same pages

The Structure of Page Table

Hierarchical Paging

Hashed Page Tables

Inverted Page Tables

Question

- Suppose a system has
  - 4GB (2^{32} bytes) RAM
  - frame size is 1KB
  - What is the data type of the frame column in page table?
    A. int (32 bits)
    B. long (64 bits)
    C. float (32 bits)
    D. double (64 bits)

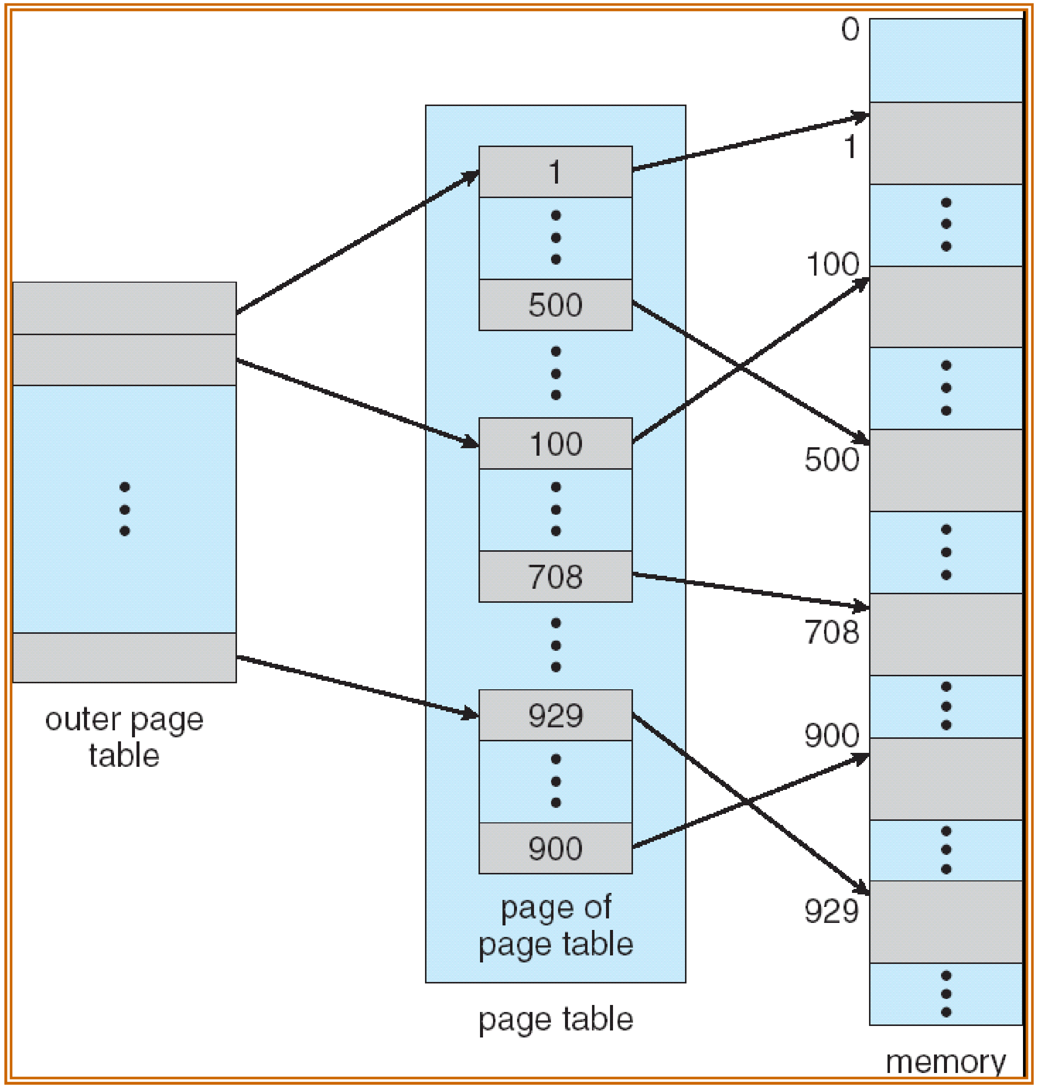
Question

- Suppose a system has
  - 4GB (2^{32} bytes) RAM
  - frame size is 1KB
  - What is the maximum size of a page table?
    A. 16 MB
    B. 2MB
    C. 4 MB
    D. 8MB

Hierarchical Page Tables

- One level page table
  - Big page table
  - Can not fit a page
- Break up the logical address space into multiple page tables
- A simple technique is a two-level page table

Two-Level Page-Table Scheme



Two-Level Paging Example

- A logical address (on 32-bit machine with 4K frame size) is divided into:
  - a page number consisting of 20 bits
  - a page offset consisting of 12 bits
- Since the page table is paged, the page number is further divided into:
  - a 10-bit outter page number
  - a 10-bit inner page offset

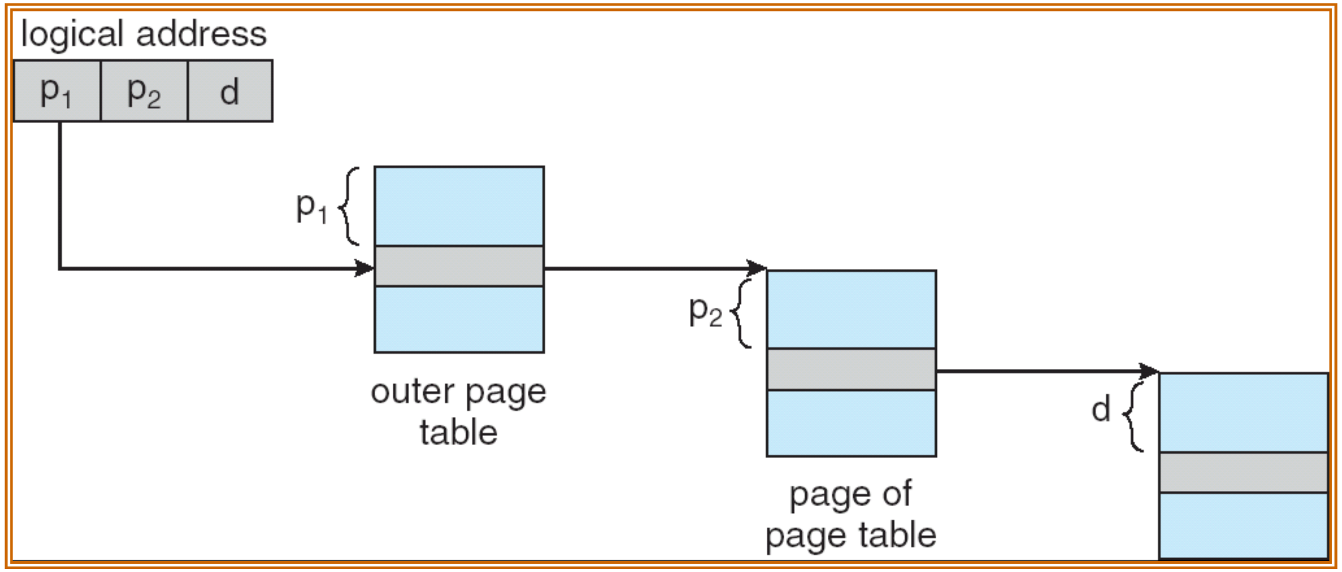
Two-Level Paging Example (cont'd)

- A logical address is as follows

page number		page offset
p_1	p_2	d
10	10	12

- p_1 is an index into the outer page table
- p_2 is the displacement within the page of the outer page table

Two-Level Paging Example (cont'd)



Multi-level Paging Scheme

outer page	inner page	offset
p_1	p_2	d
42	10	12

2nd outer page	outer page	inner page	offset
p_1	p_2	p_3	d
32	10	10	12

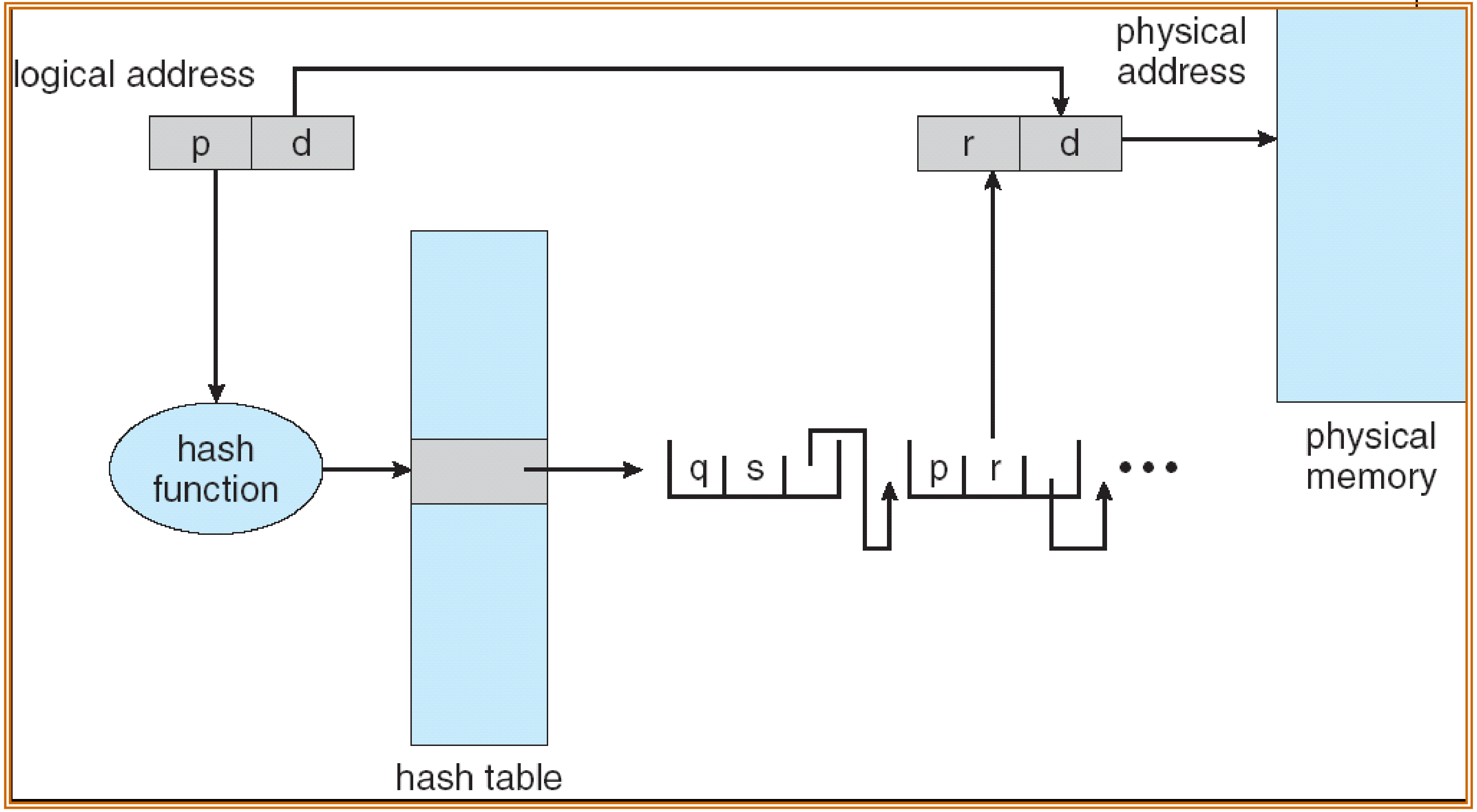
Question

- A system use two level paging; the address register uses
  - *m* bits for outer page
  - *n* bits for inner page
  - *k* bits for offset
- Which of the following is incorrect?
  A. the resolution of a physical takes 3 steps
  B. a memory reference has the form ($p1$,$p2$,d)
  C. the physical address is $p_1*2^m+p_2*2^n+d$, where p_1, p_2 are corresponding values from page tables
  D. d must be less than 2^k

Hashed Page Tables

- Common in address spaces > 32 bits
- The virtual page number is hashed into a page table
  - This page table contains a chain of elements hashing to the same location.
- Virtual page numbers are compared in this chain searching for a match
  - If a match is found, the corresponding physical frame is extracted.

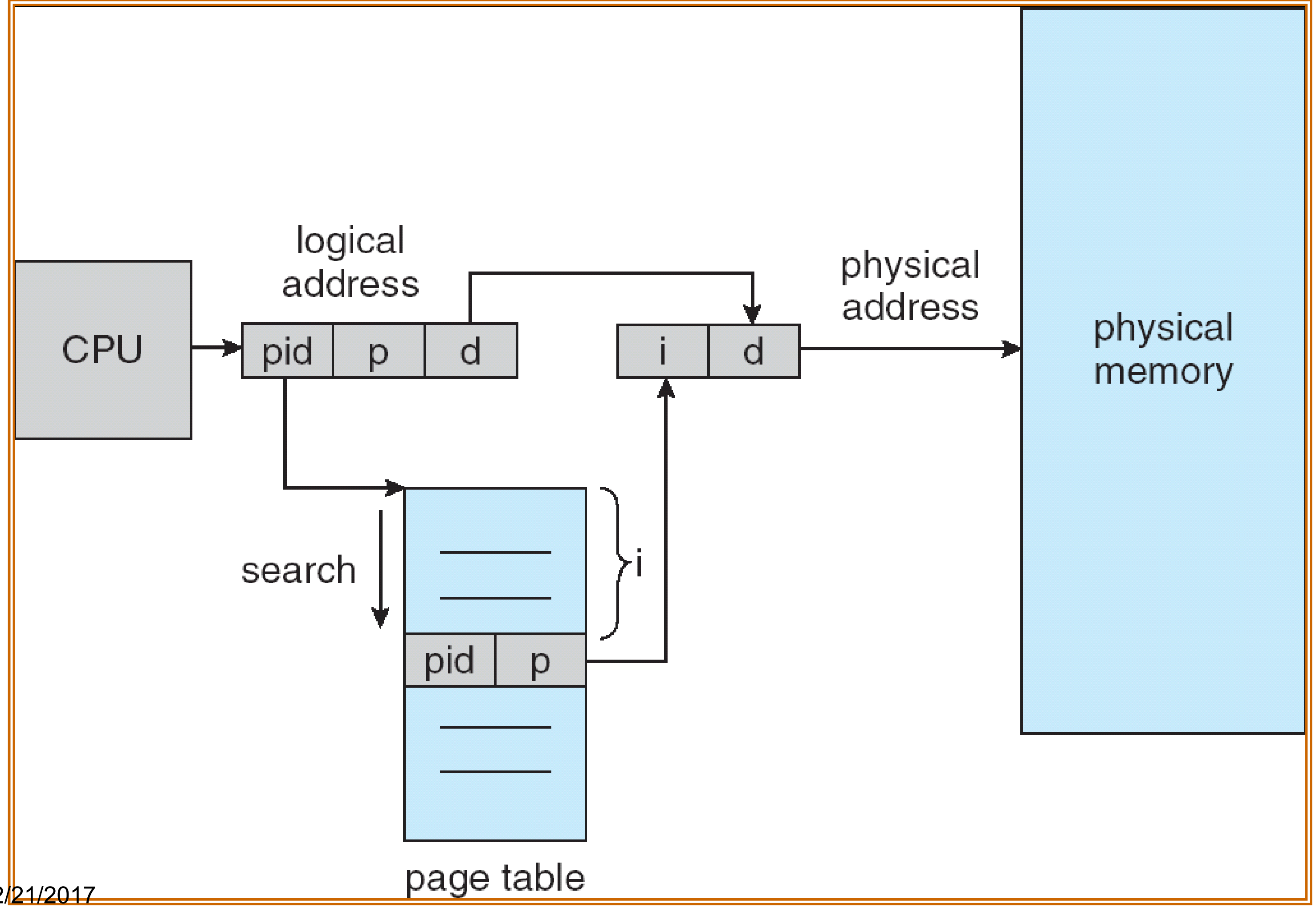
Hashed Page Table



Inverted Page Table

- One entry for each real page of memory
- Entry consists of
  - the virtual address of the page stored in that real memory location,
  - information about the process that owns that page
- Decreases memory needed to store each page table, but increases time needed to search the table when a page reference occurs
- Limit the search to one — or at most a few — page-table entries

Inverted Page Table Architecture



Segmentation

Segmentation

- Memory-management scheme that supports user view of memory
- A program is a collection of segments
- A segment is a logical unit

  main program,

  procedure,

  function,

  method,

  object,

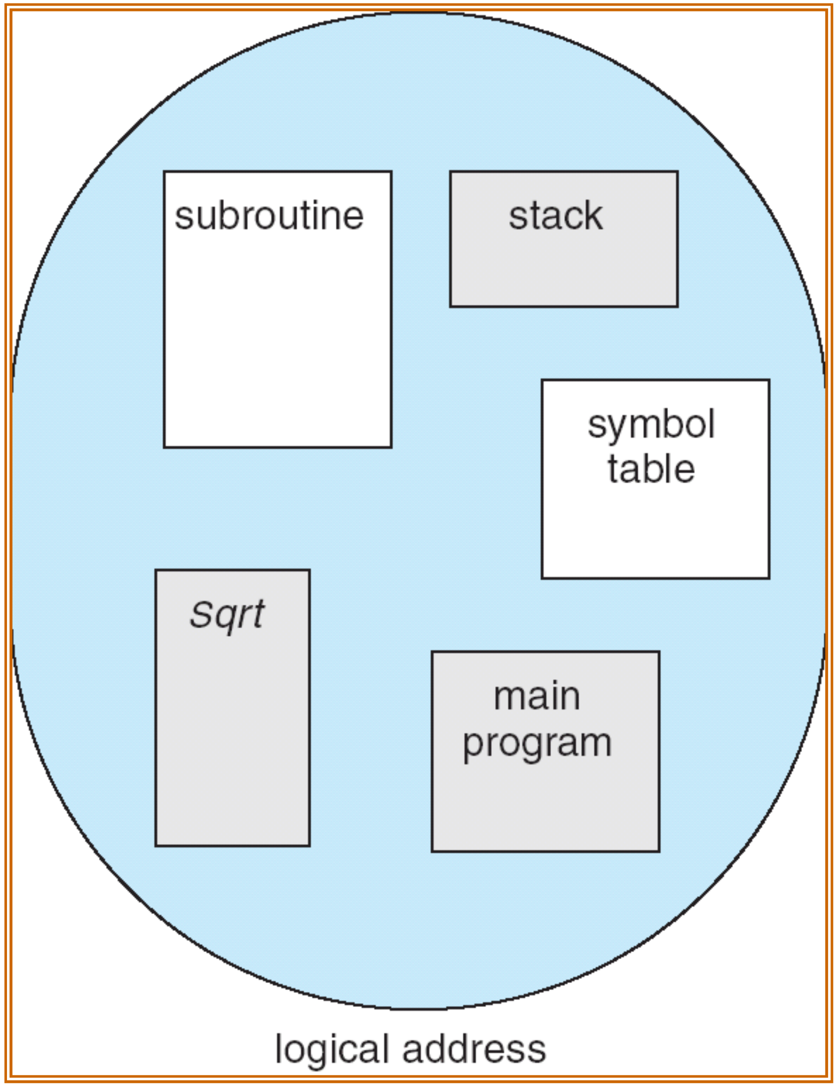
  local variables, global variables,

  common block,

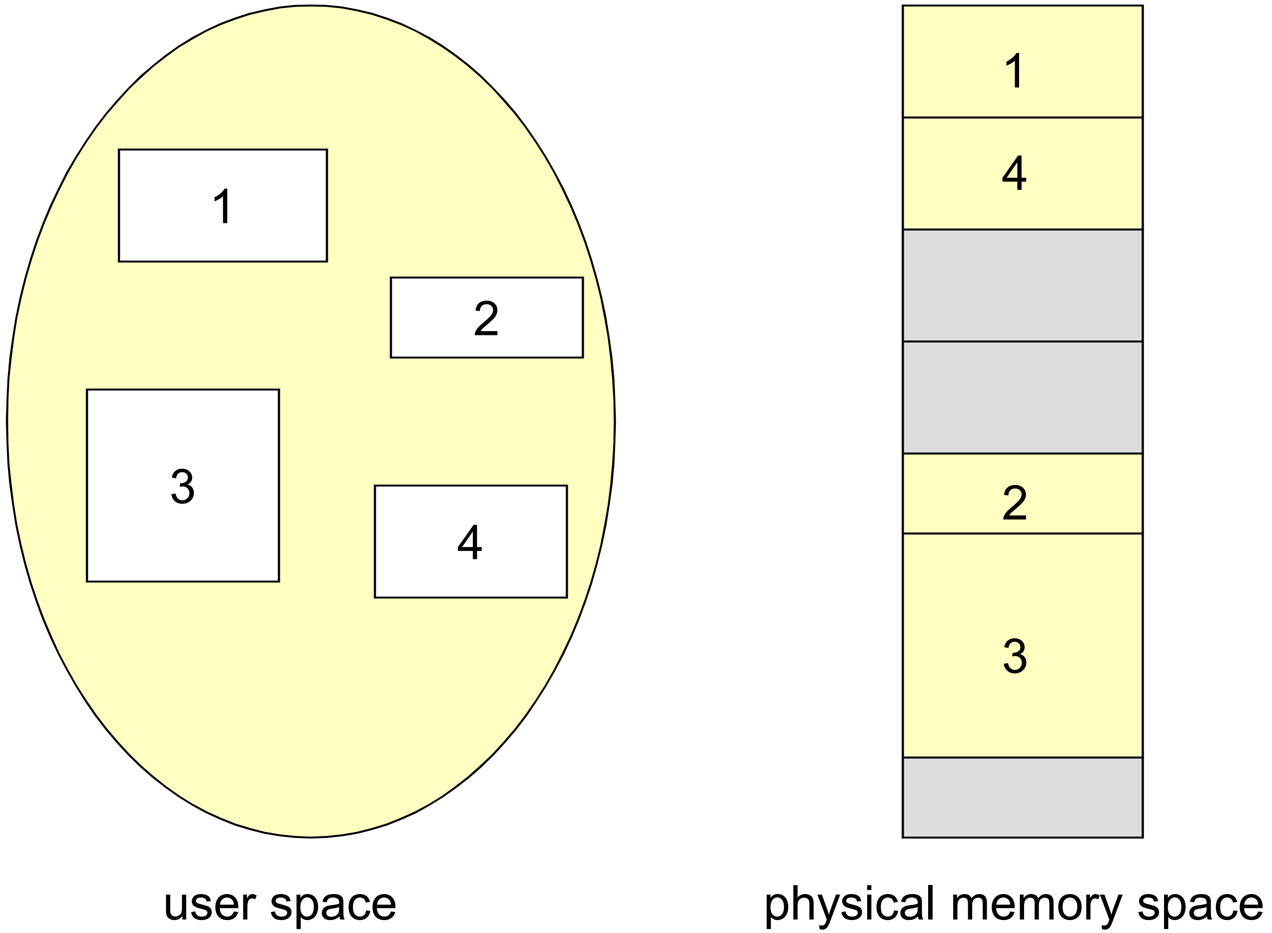
  stack,

  symbol table, arrays

User's View of a Program



Logical View of Segmentation



Segmentation Architecture

- Logical address consists of a two tuple

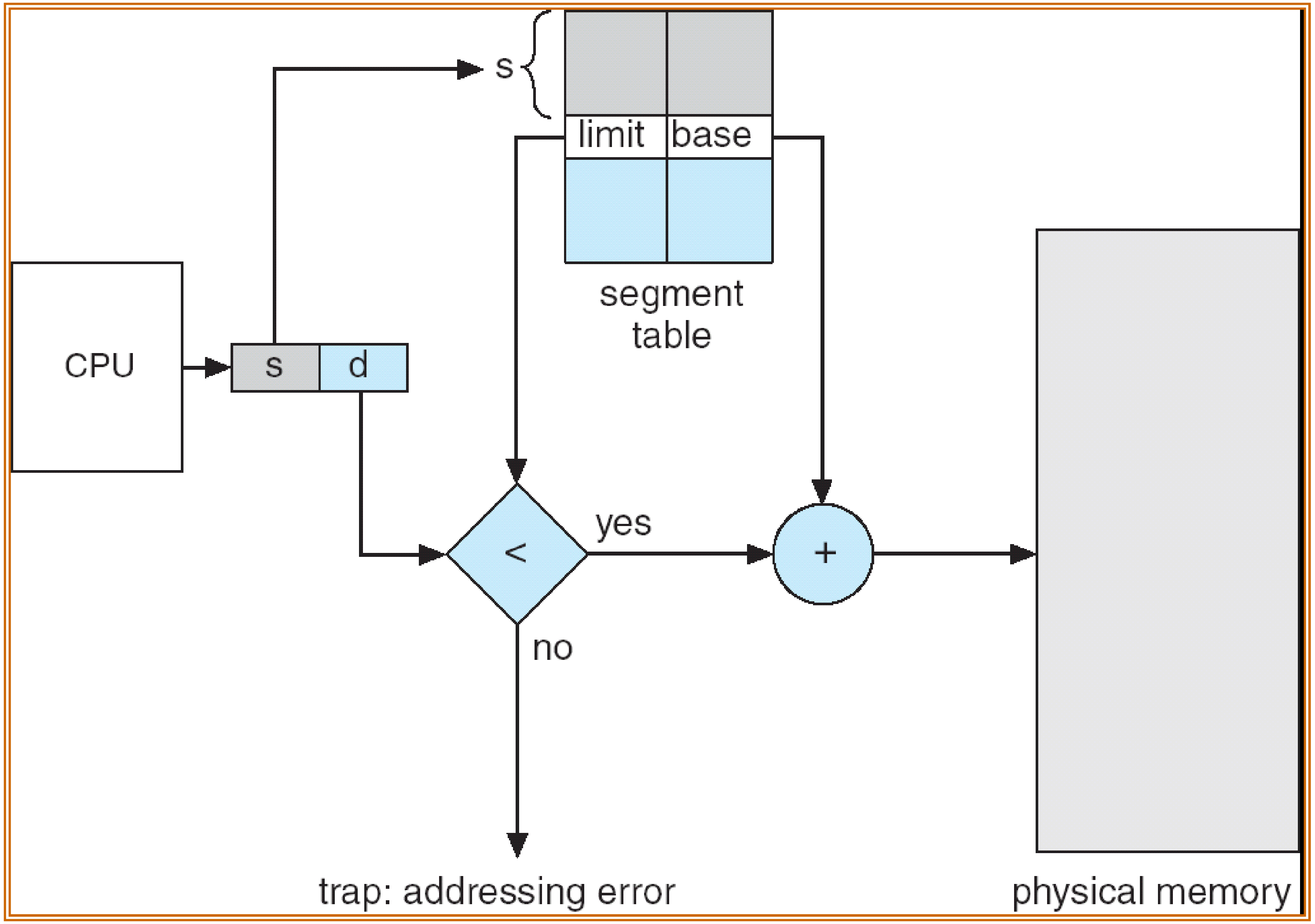
  <segment-number, offset>,
- **Segment table** – maps two-dimensional physical addresses; each table entry has:
  - **base** – contains the starting physical address where the segments reside in memory
  - **limit** – specifies the length of the segment
- **Segment-table base register (STBR)** points to the segment table's location in memory
- **Segment-table length register (STLR)** indicates number of segments used by a program;

  segment number s is legal if $s < STLR$

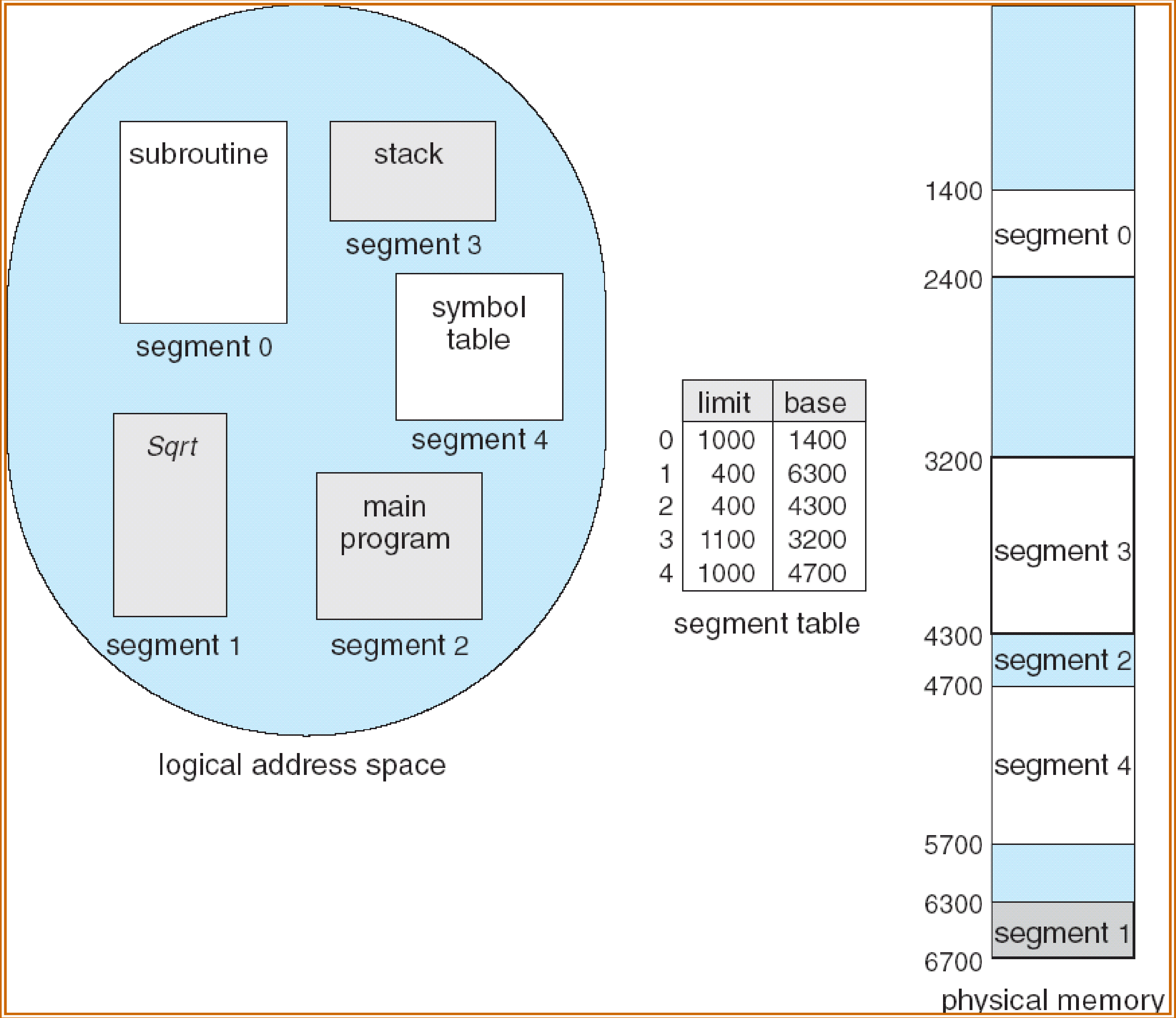
Segmentation Architecture (Cont.)

- Protection
  - Each entry in segment table includes:
    - validation bit = 0 $\Rightarrow$ illegal segment
    - read/write/execute privileges
- Protection bits associated with segments; code sharing occurs at segment level
- Segments vary in length
  - $\Rightarrow$memory allocation is a dynamic storage-allocation problem

Segmentation Hardware



Example of Segmentation



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

segment table

Question

- Which of the following is incorrect about segmentation allocation?
  A. it originates from user's view of a program
  B. it is also of noncontiguous memory allocation method
  C. segment table has the same structure as page table
  D. segments can have different sizes

Question

- Consider the allocation of the above program, calculate the physical address of the reference (3, 208)

A. 3408

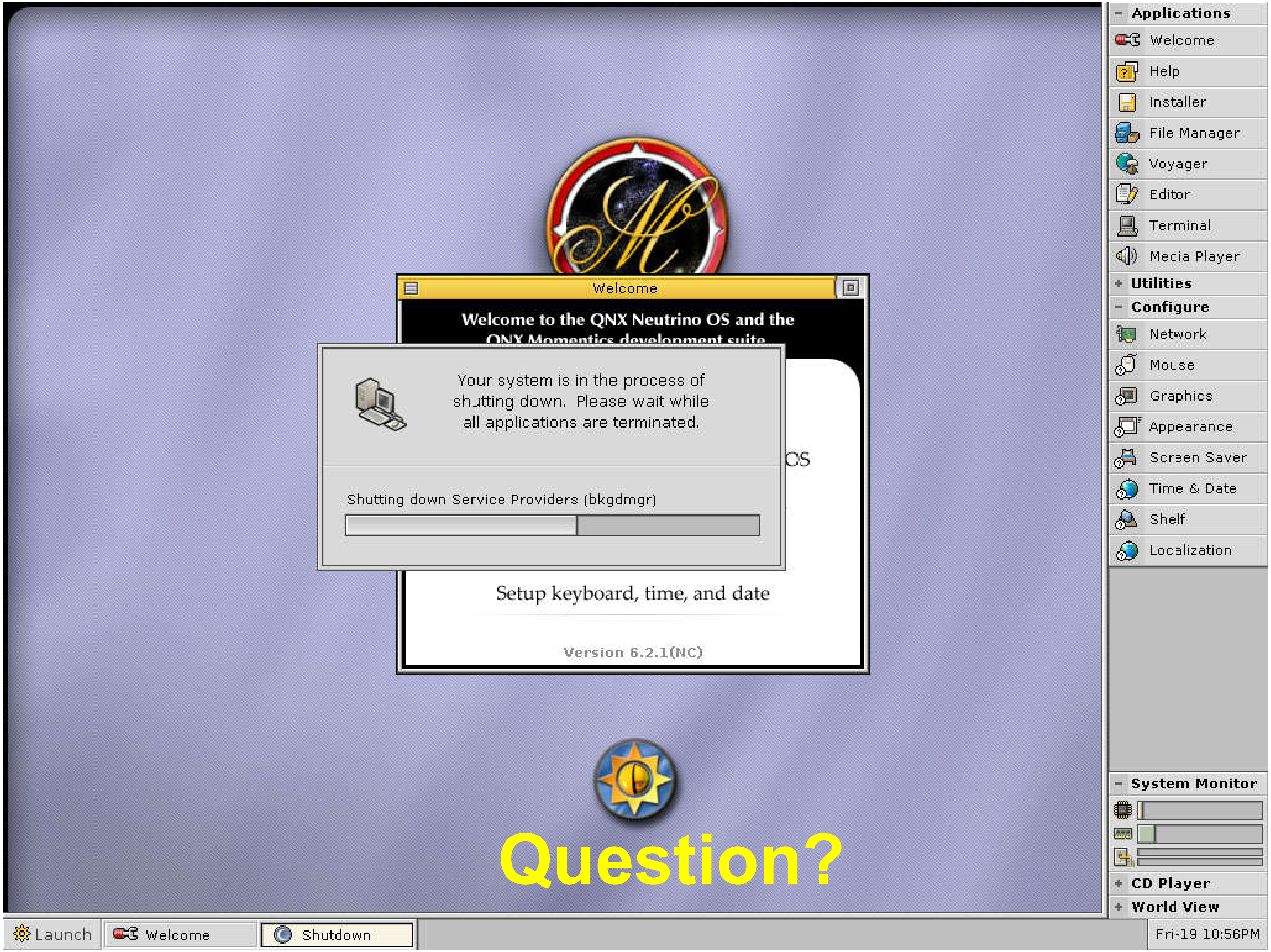
B. 3208

C. 4408

D. 2008

Question

- Consider the allocation of the above program, calculate the physical address of the reference (2, 450)
  A. 3408
  B. 3208
  C. 4408
  D. 2008


Applications
Welcome
Help
Installer
File Manager
Voyager
Editor
Terminal
Media Player
Utilities
Configure
Network
Mouse
Graphics
Appearance
Screen Saver
Time & Date
Shelf
Localization
System Monitor
CD Player
World View
Welcome
Welcome to the QNX Neutrino OS and the
Your system is in the process of
shutting down. Please wait while
all applications are terminated.
Shutting down Service Providers (bkgdmgr)
Setup keyboard, time, and date
Version 6.2.1(NC)
Launch
Welcome
Shutdown
Fri-19 10:56PM

Question?