Comp 204: Computer Systems and Their Implementation

Lecture 15: Paging

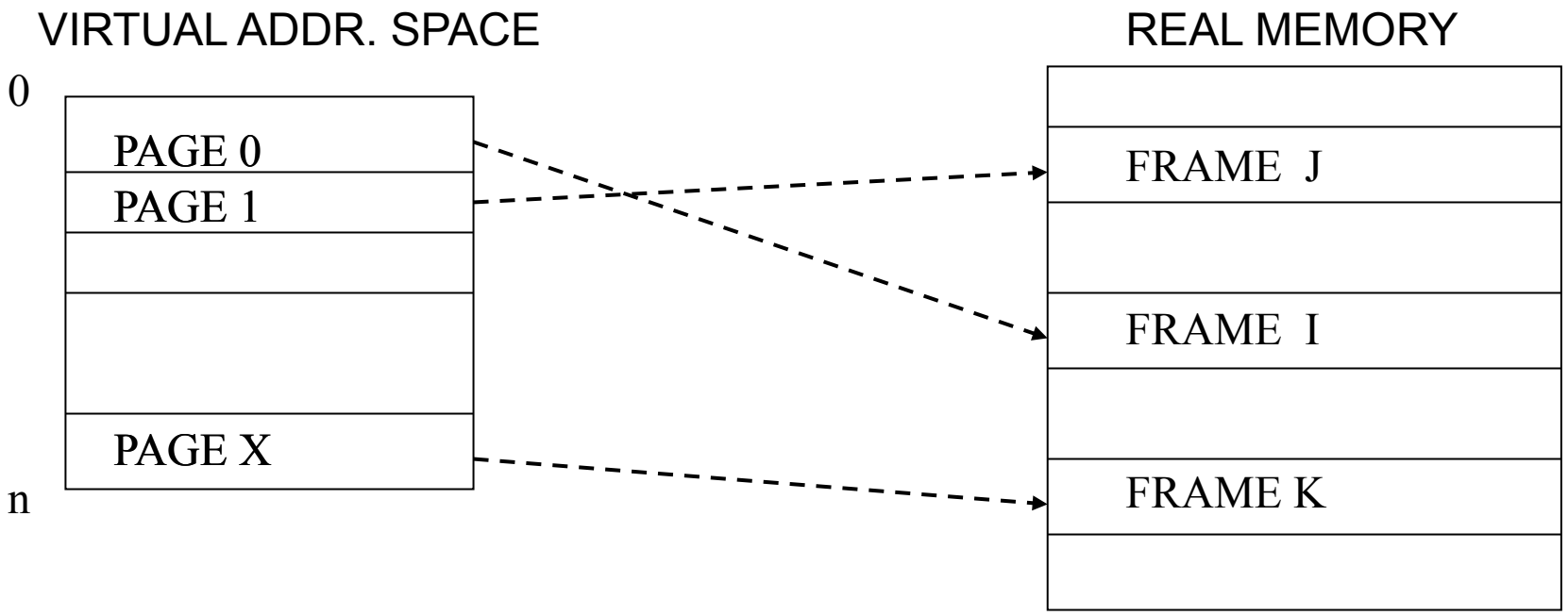
Today

- Paging
  - Paged memory
  - Virtual addressing
- Page replacement
  - Principle of Locality

Paging

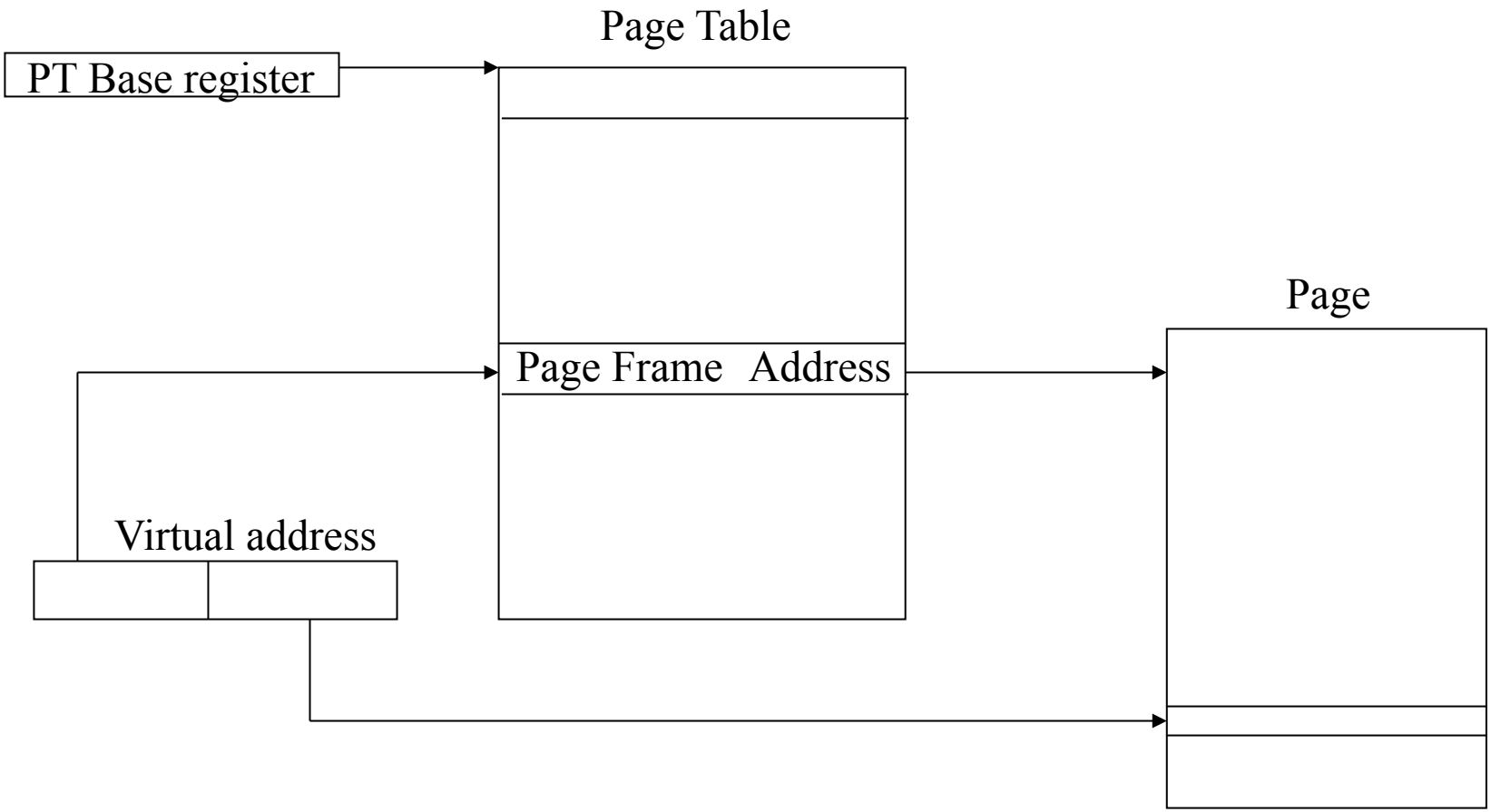
- Paging is the physical division of a program's (or segment's) address space into equal-sized blocks called pages
- Each page resides within a page frame in the real memory
- Pages which are consecutive in the address space (virtual memory) need not occupy consecutive page frames in real memory

Page Mapping



- Translation of virtual addresses (used by program) into real addresses performed by hardware via a page table per process

Paged Memory



Virtual Addressing

Virtual Address

Virtual Page Number	In-page address

Storage Mapping function

1. PTBR addresses Page Table in memory.

2. Virtual page number indexes page table to produce real page address

3. In-page address indexes real page.

Question

- In a paged memory system, why are page sizes invariably a power of 2?

a) Because computer memory is usually a multiple of 1K, which is a power of 2.
b) Because pages have to begin at address boundaries that are even.
c) Because virtual address spaces are usually a power of 2 in size
d) Because it simplifies indexing of the page table and the calculation of the page offset.
e) Because most data types occupy even numbers of bytes.

Answer: d
Because it simplifies indexing of the page table and the calculation of the page offset.

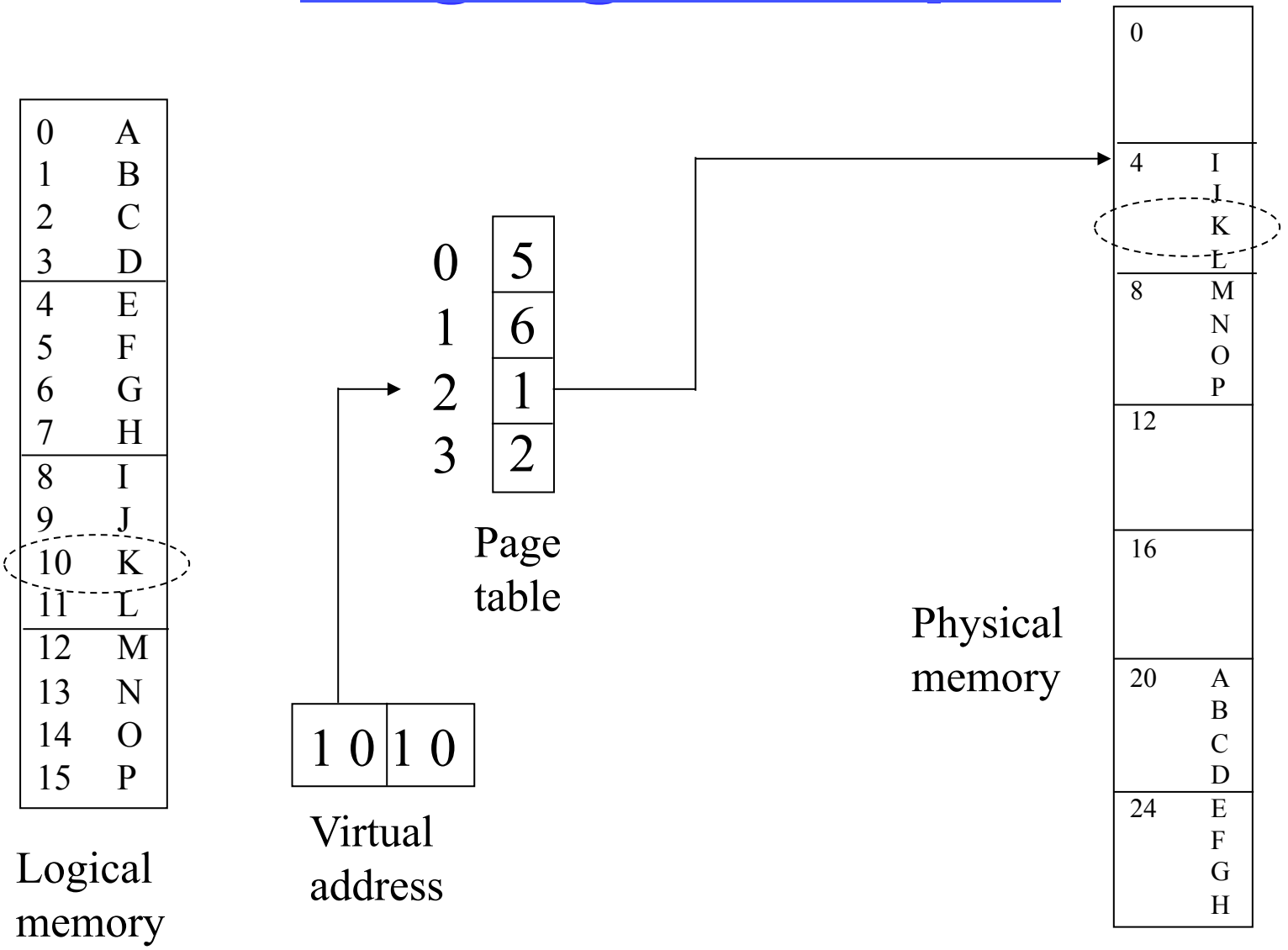
Question

- A computer uses 16-bit addressing. Its page size is 512 bytes. What is the maximum number of entries that the page table must be capable of holding?

  a) 16
  b) 64
  c) 128
  d) 256
  e) 512

Answer: c
128; 9 bits are used for addressing the 512 bytes in each page, so the remaining 7 bits are used to address the pages

Paging Example



Question

- The page table shown below is for a job in a paged virtual storage system with a page size of 1K:

segment	datum
0	4
1	2
2	0
3	1

- A virtual address of [1, 352] would map on to what actual address?

a) 354
b) 1376
c) 2352
d) 2400
e) 4448

Answer: d
2048 (from page 1) + 352 (offset)

Segmentation & Paging

- Segmentation:
  - Logical division of address space
  - varying sized units
  - units ‘visible’ to program

- Paging:
  - Physical division of address space
  - fixed-size units
  - units bear no relation to program structure

Segmentation & Paging

- Either may be used as a basis for a swapping system
- Store may be both segmented and paged
  - more complex mapping function using 2 tables

- Advantages of paging:
  - fixed-size units make space allocation simpler
  - normal fragmentation eliminated, but still some internal fragmentation, i.e. space wasted within frames

Example: The Intel Pentium

- Supports segmentation with paging
- CPU generates logical address, which is passed to segmentation unit
- Segmentation unit produces a linear address, which is passed to paging unit
- Paging unit generates physical address in main memory

Virtual Memory

- The maximum logical address space per process may be smaller than physical memory
- Conversely, it may be larger!
  - May want to run a 100MB process on a machine with 64MB memory
- Possible with paging
  - Not all pages need be in memory
  - Unneeded pages can reside on disk
  - Memory becomes virtual, i.e. not restricted by physical memory

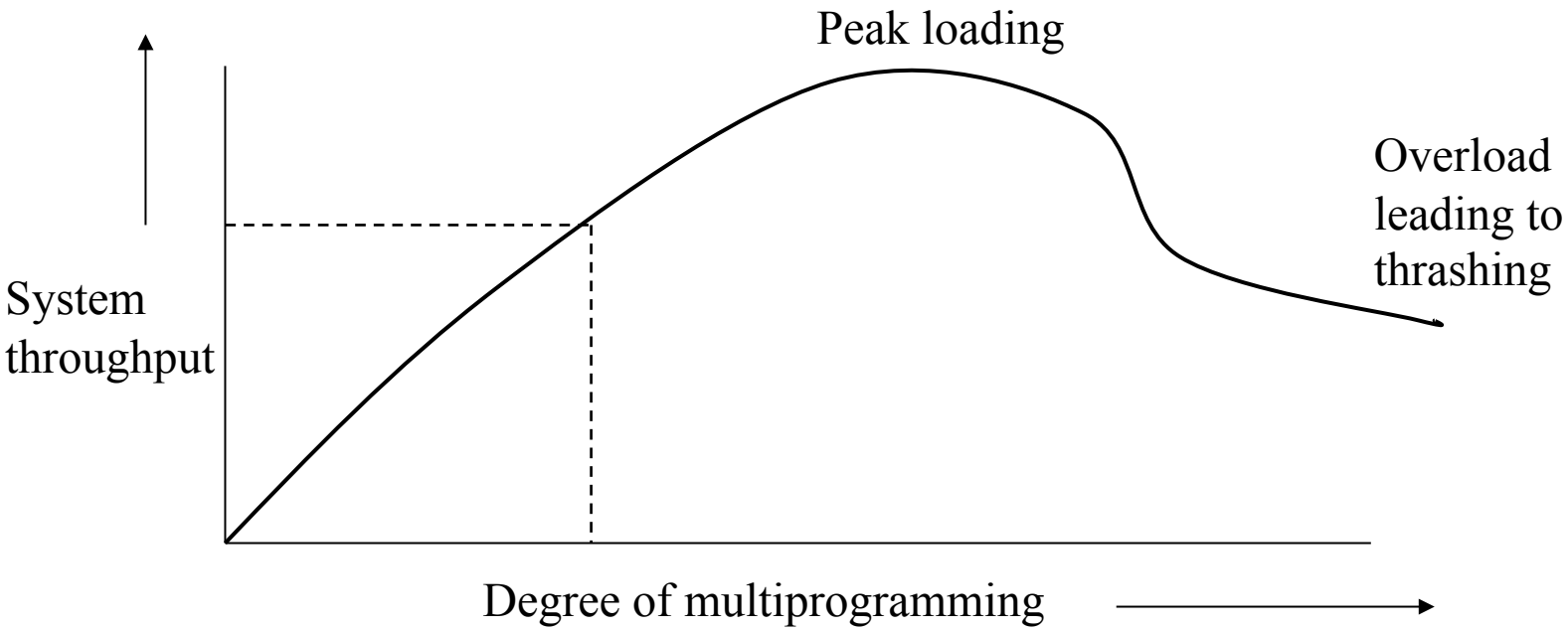
Problem 1

- What happens if a process references a page that is not in main store?
  - A page fault ensues
- Page fault generates an interrupt because address references cannot be satisfied until page swapped in
- O.S. response is normally to fetch page required (demand paging)

Problem 2

- How do we make room for fetched page? (page replacement problem)
- Would like to swap out pages not immediately needed
  - have to guess!
- A poor guess will quickly lead to a page fault
- Many poor guesses will lead to persistent swapping (thrashing)

Thrashing



Thrashing comes about as result of system overload

– can be delayed by good page replacement policy

Page Replacement

- Optimal policy: swap out page that will not be needed for longest time in the future
- To estimate this, use...

PRINCIPLE OF LOCALITY

Over any short period of time, a program's memory references tend to be spatially localised

Consequence is that program's memory needs in next time period are likely to be close to those during last time period

Question

- Which of the following programming constructs tend to contribute to the phenomenon expressed in the Principle of Locality?

I. Iteration (e.g. FOR and WHILE loops)
II. Selection (e.g. IF-statements)
III. Recursion

a) I only
b) III only
c) I and III only
d) II and III only
e) I and II only

Answer: c
I and III only