Operating System

Dr. GuoJun LIU

Harbin Institute of Technology

http://guojunos.hit.edu.cn

Chapter 08

Virtual Memory

虚拟内存

Learning Objectives

- **Define virtual memory**
- **Describe the hardware and control structures that support virtual memory**

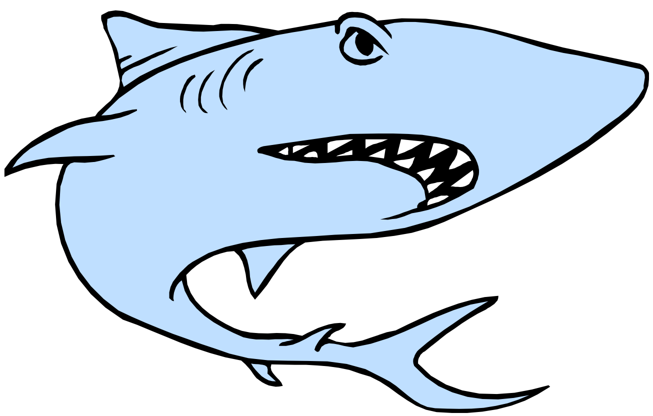
Outline

- **Locality and Virtual Memory**
- **Paging**
  - Page table structure
  - Translation lookaside buffer
  - Page size
- **Segmentation**
- **Combined Paging and Segmentation**
- **Protection and Sharing**

You're gonna need a **bigger** boat.

-- ***Steven Spielberg* ,**

JAWS, 1975



Terminology

- **Virtual memory**
  - **secondary memory** can be **addressed**
  - The **addresses a program** may use to reference memory are **distinguished from** the **addresses the memory system** uses to identify physical storage sites
  - The size of virtual storage is limited
    - by the addressing scheme of the **computer system**
    - by the **amount of secondary memory** available
    - **not by** the actual number of main storage locations
- **Virtual address**
  - The address assigned to a location in **virtual memory** to allow that location to be accessed as though it were part of main memory
- **Virtual address space**
  - **virtual storage** assigned to a process
- **Address space**
  - The **range** of **memory addresses** available to a process
- **Real address**
  - The address of a storage location in **main memory**

Hardware and Control Structures

- **Two characteristics fundamental to memory management:**
  - all memory references are **logical addresses** that are **dynamically translated** into **physical addresses** at run time
  - a process may be broken up into **a number of pieces** that **don't** need to be **contiguously located** in main memory during execution
- **If these two characteristics are present**
  - it is not necessary that **all of the pages or segments** of a process be in **main memory during execution**

Execution of a Process 1/2

- **OS brings into main memory a few pieces of the program**
  - the term **piece** to refer to either **page** or **segment**
- **Resident set**
  - **portion of process** that is in main memory
- **An interrupt is generated when an address is needed that is not in main memory**
- **OS places the process in a blocking state**

Execution of a Process 2/2

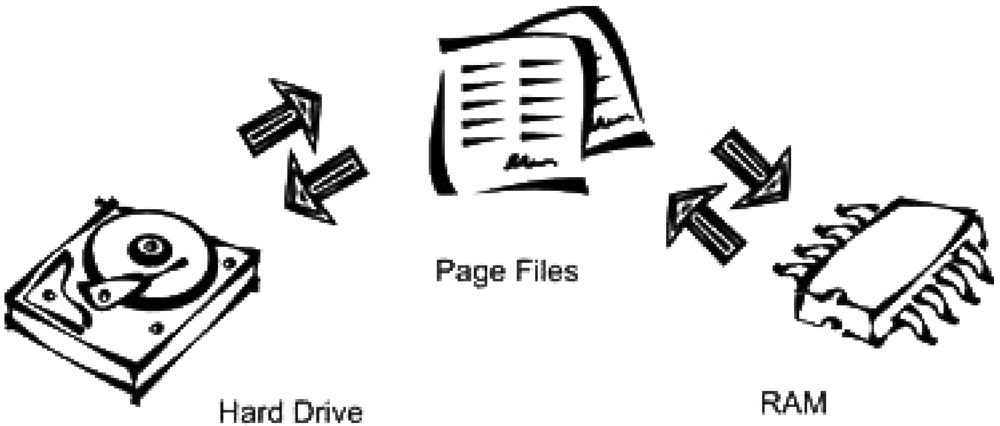
- **Piece of process that contains the logical address is brought into main memory**
  - OS **issues** a disk I/O Read **request**
  - **another process is dispatched to run** while the disk I/O takes place
  - **an interrupt** is issued when disk I/O is **complete**, which causes the OS to place the affected process in the **Ready state**

Two Implications

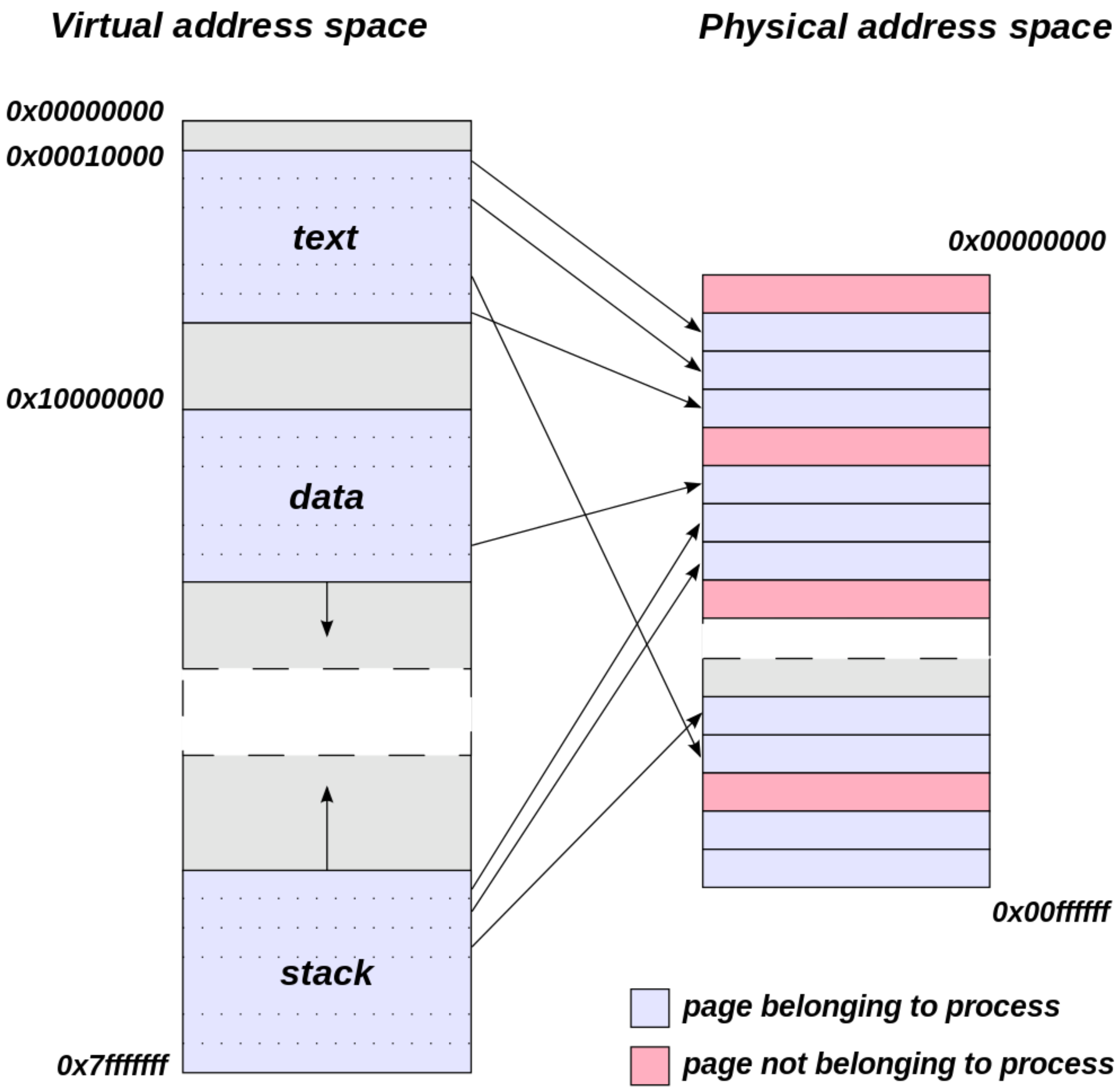
- **More processes may be maintained in main memory**
  - only load in some of the pieces of each process
  - with so many processes in main memory, it is very likely a process will be in the **Ready state** at any particular time
- **A process may be larger than all of main memory**
  - **Without** the scheme **a programmer** must be **acutely aware** of **how much memory is available**
  - If the program being written is **too large**, the programmer must devise ways to **structure the program into pieces** that can be loaded separately in some sort of overlay strategy
  - OS **automatically loads** pieces of a process into **main memory** as required

Real and Virtual Memory

- **Real memory**
  - **main** memory
  - the **actual** RAM
- **Virtual memory**
  - memory on **disk**
  - allows for effective multiprogramming
  - relieves the user of tight constraints of main memory



Simple Paging	Virtual Memory Paging	Simple Segmentation	Virtual Memory Segmentation
Main memory partitioned into small fixed-size chunks called frames	Main memory partitioned into small fixed-size chunks called frames	Main memory not partitioned	Main memory not partitioned
Program broken into pages by the compiler or memory management system	Program broken into pages by the compiler or memory management system	Program segments specified by the programmer to the compiler (i.e., the decision is made by the programmer)	Program segments specified by the programmer to the compiler (i.e., the decision is made by the programmer)
Internal fragmentation within frames	Internal fragmentation within frames	No internal fragmentation	No internal fragmentation
No external fragmentation	No external fragmentation	External fragmentation	External fragmentation
Operating system must maintain a page table for each process showing which frame each page occupies	Operating system must maintain a page table for each process showing which frame each page occupies	Operating system must maintain a segment table for each process showing the load address and length of each segment	Operating system must maintain a segment table for each process showing the load address and length of each segment
Operating system must maintain a free frame list	Operating system must maintain a free frame list	Operating system must maintain a list of free holes in main memory	Operating system must maintain a list of free holes in main memory
Processor uses page number, offset to calculate absolute address	Processor uses page number, offset to calculate absolute address	Processor uses segment number, offset to calculate absolute address	Processor uses segment number, offset to calculate absolute address
All the pages of a process must be in main memory for process to run, unless overlays are used	**Not all pages of a process need be in main memory frames for the process to run. Pages may be read in as needed**	**All the segments of a process must be in main memory for process to run, unless overlays are used**	**Not all segments of a process need be in main memory for the process to run. Segments may be read in as needed**
	**Reading a page into main memory may require writing a page out to disk**		**Reading a segment into main memory may require writing one or more segments out to disk**



Common method of using paging to create a virtual address space

Thrashing

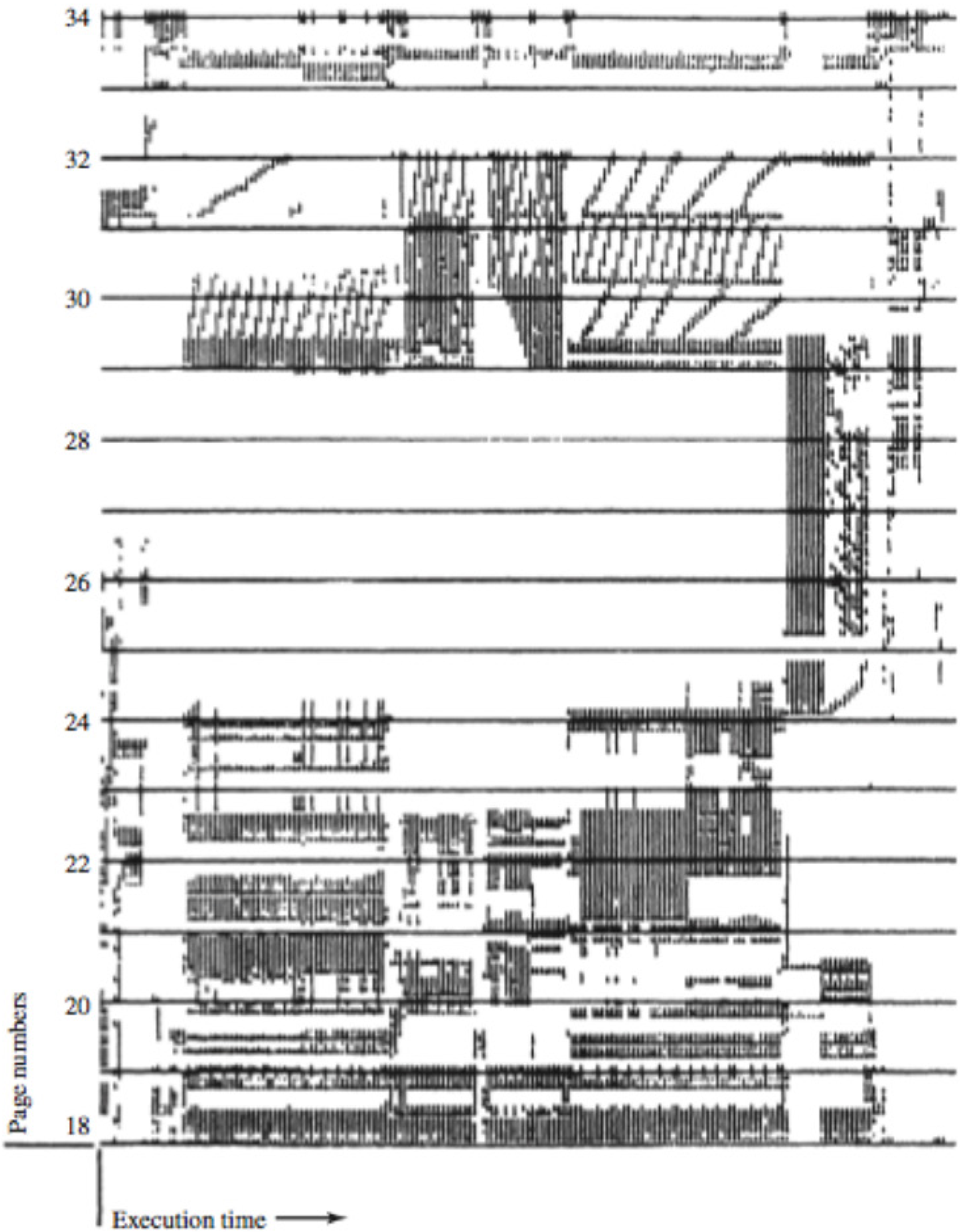
A state in which the system spends most of its time swapping process pieces rather than executing instructions

To avoid this, the operating system tries to guess, based on recent history, which pieces are least likely to be used in the near future

Principle of Locality

- Program and data references within a process tend to cluster
- Only a few pieces of a process will be needed over a short period of time
- Therefore it is possible to make intelligent guesses about which pieces will be needed in the future
- Avoids thrashing

During the lifetime of the process, references are confined to a subset of pages



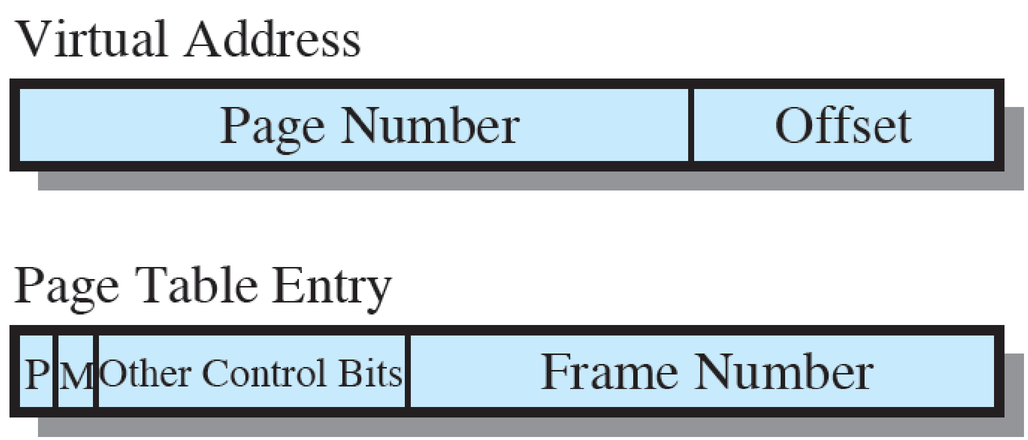
Paging Behavior

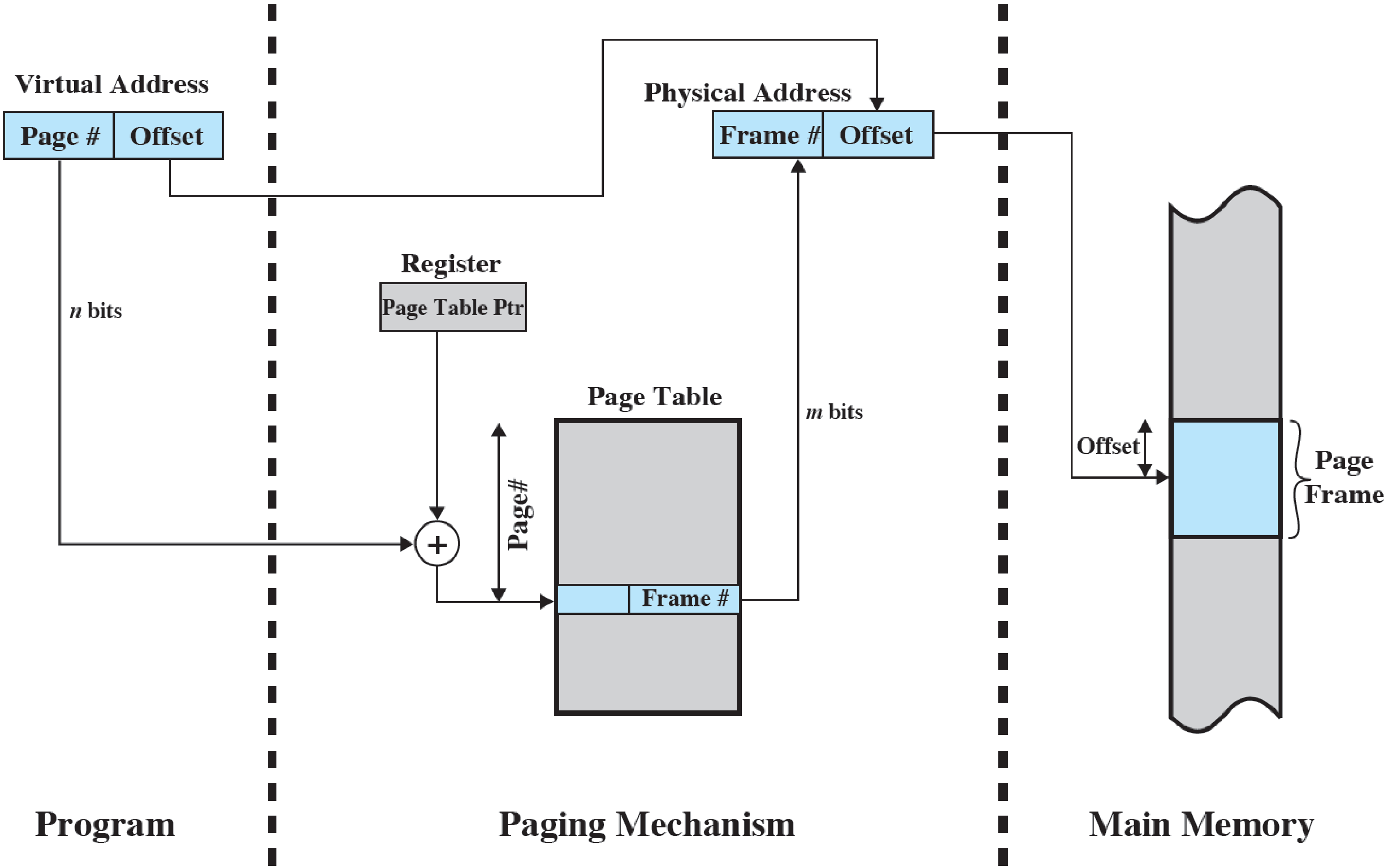
Support Needed for Virtual Memory

- **For virtual memory to be practical and effective**
  - hardware must support paging and segmentation
  - operating system must include software for managing the movement of pages and/or segments between secondary memory and main memory

Paging

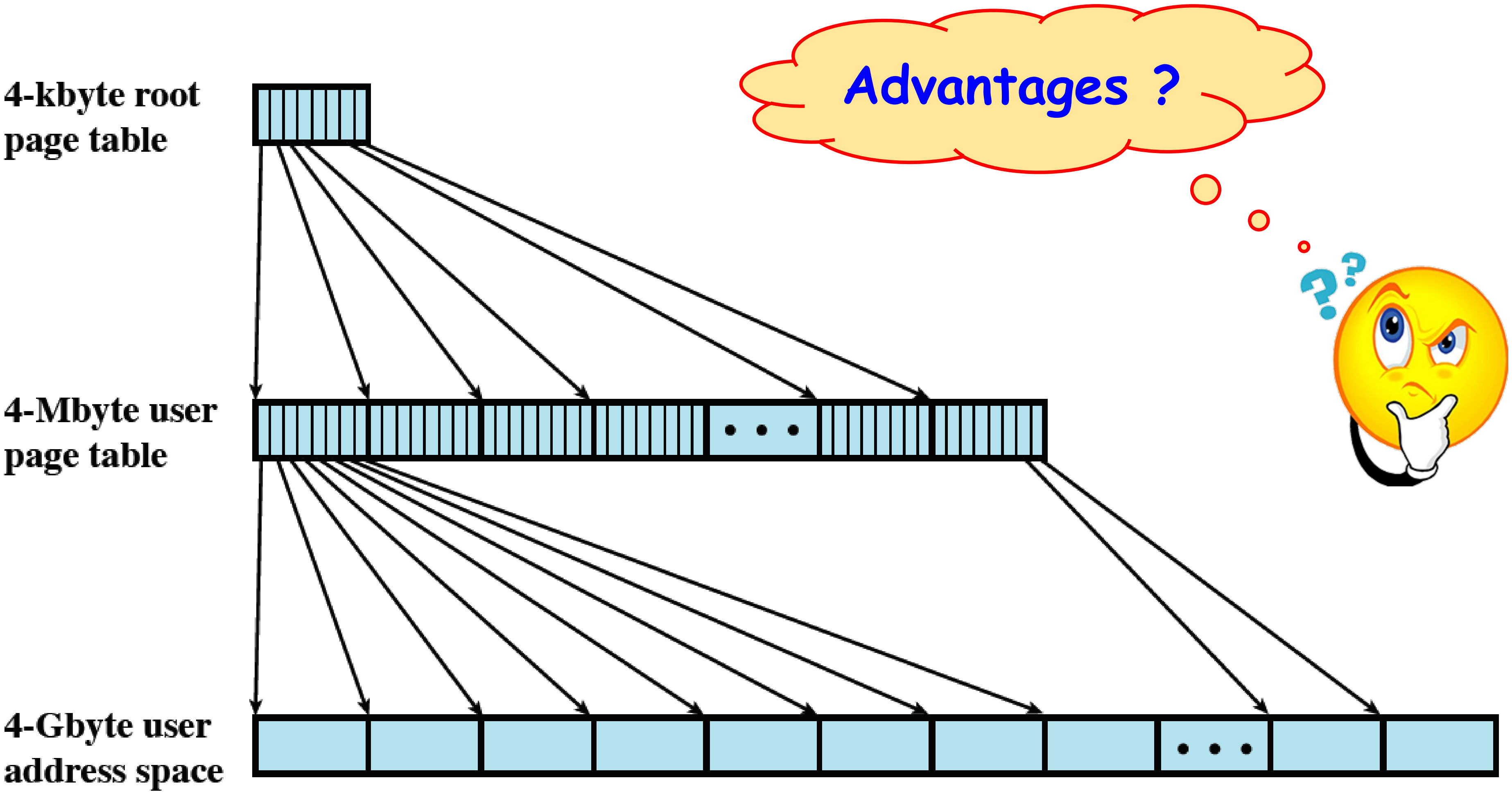
- The term virtual memory is usually associated with systems that employ paging
- Use of paging to achieve virtual memory was first reported for the Atlas computer
- Each process has its own page table

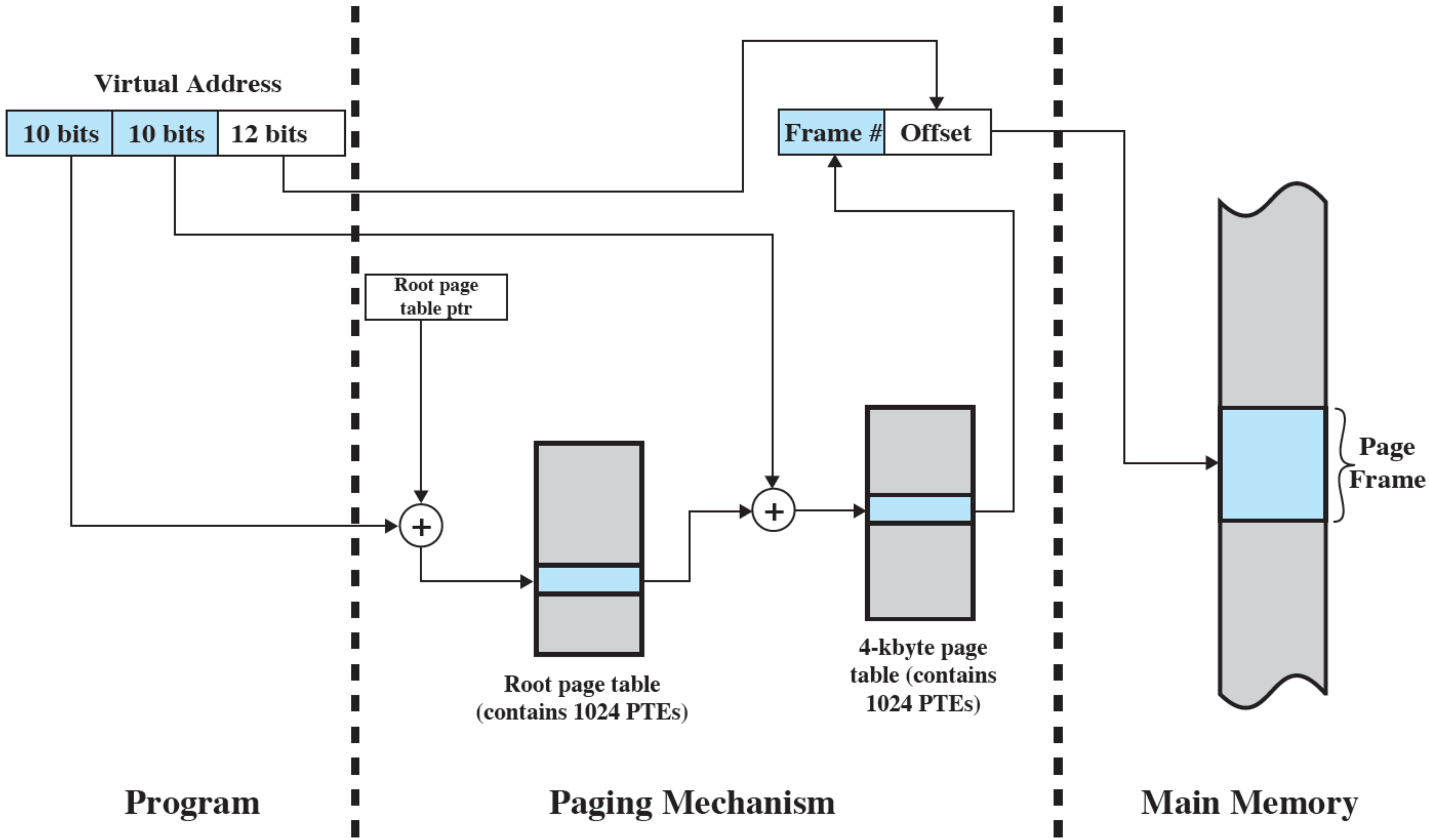




Address Translation in a Paging System

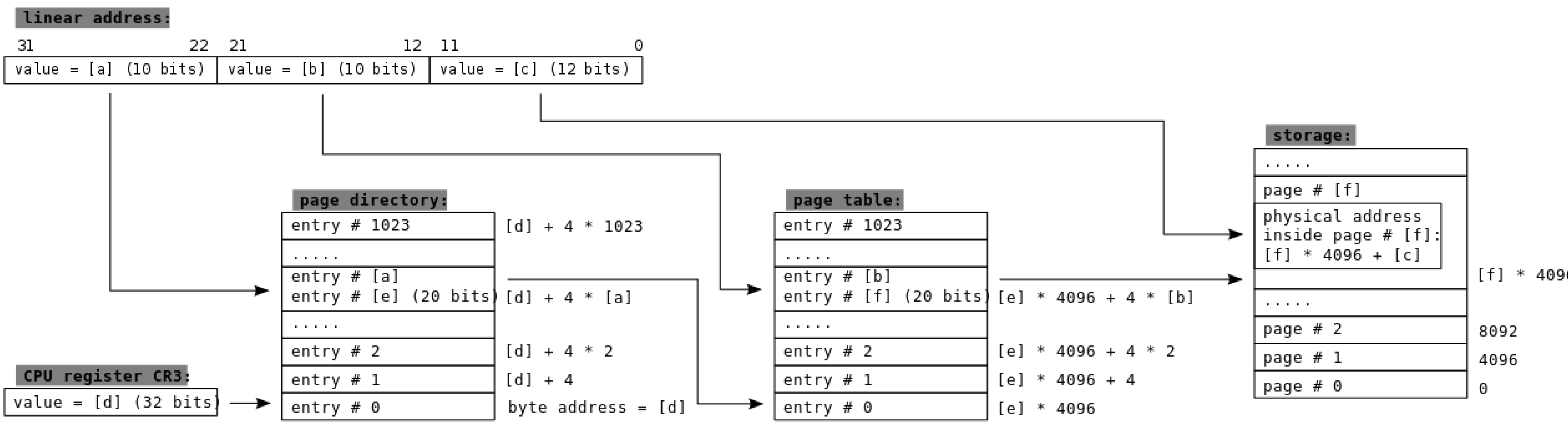
A Two-Level Hierarchical Page Table





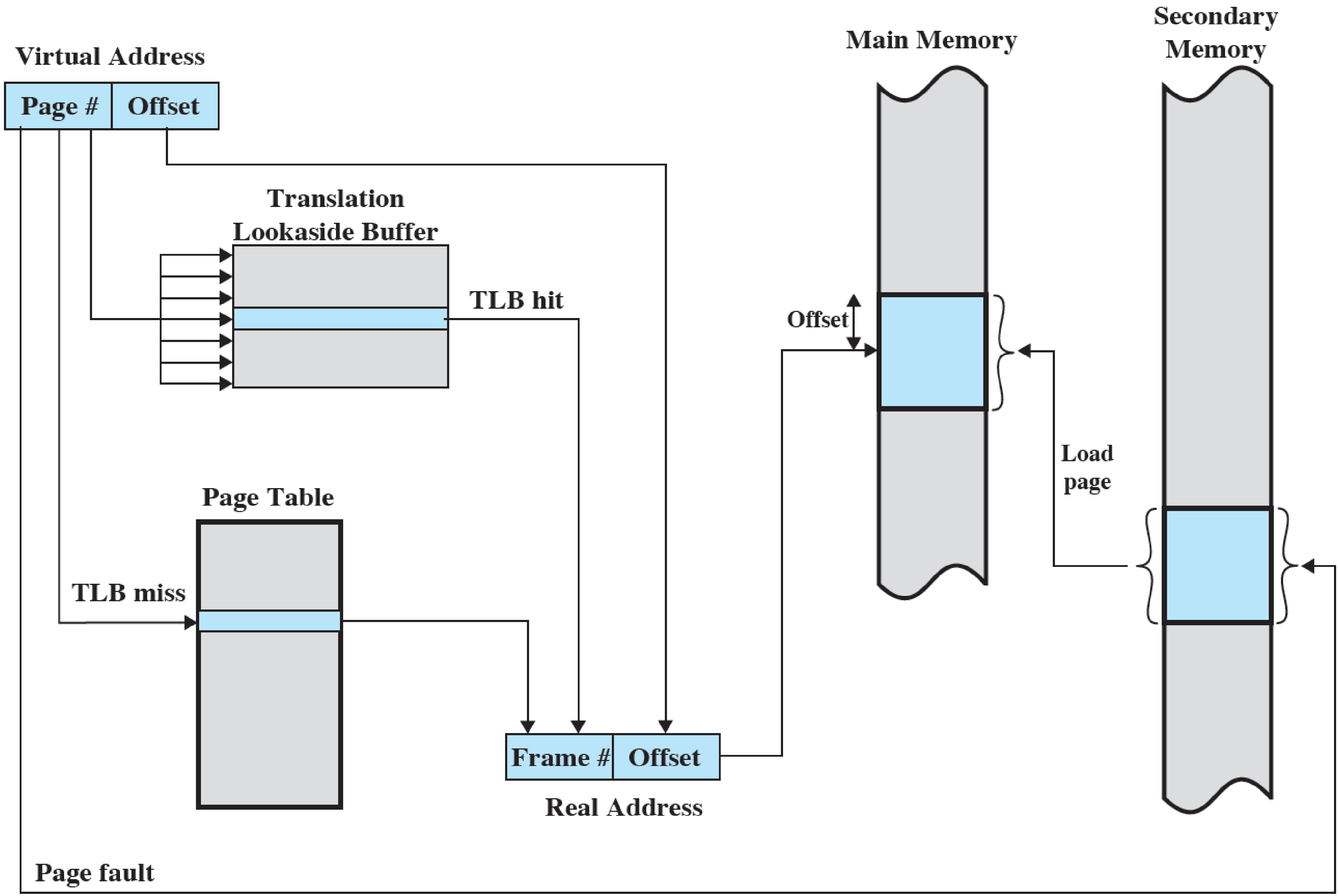
Address Translation in a Two-Level Paging System

Paging with page size of 4K -- Intel 80386



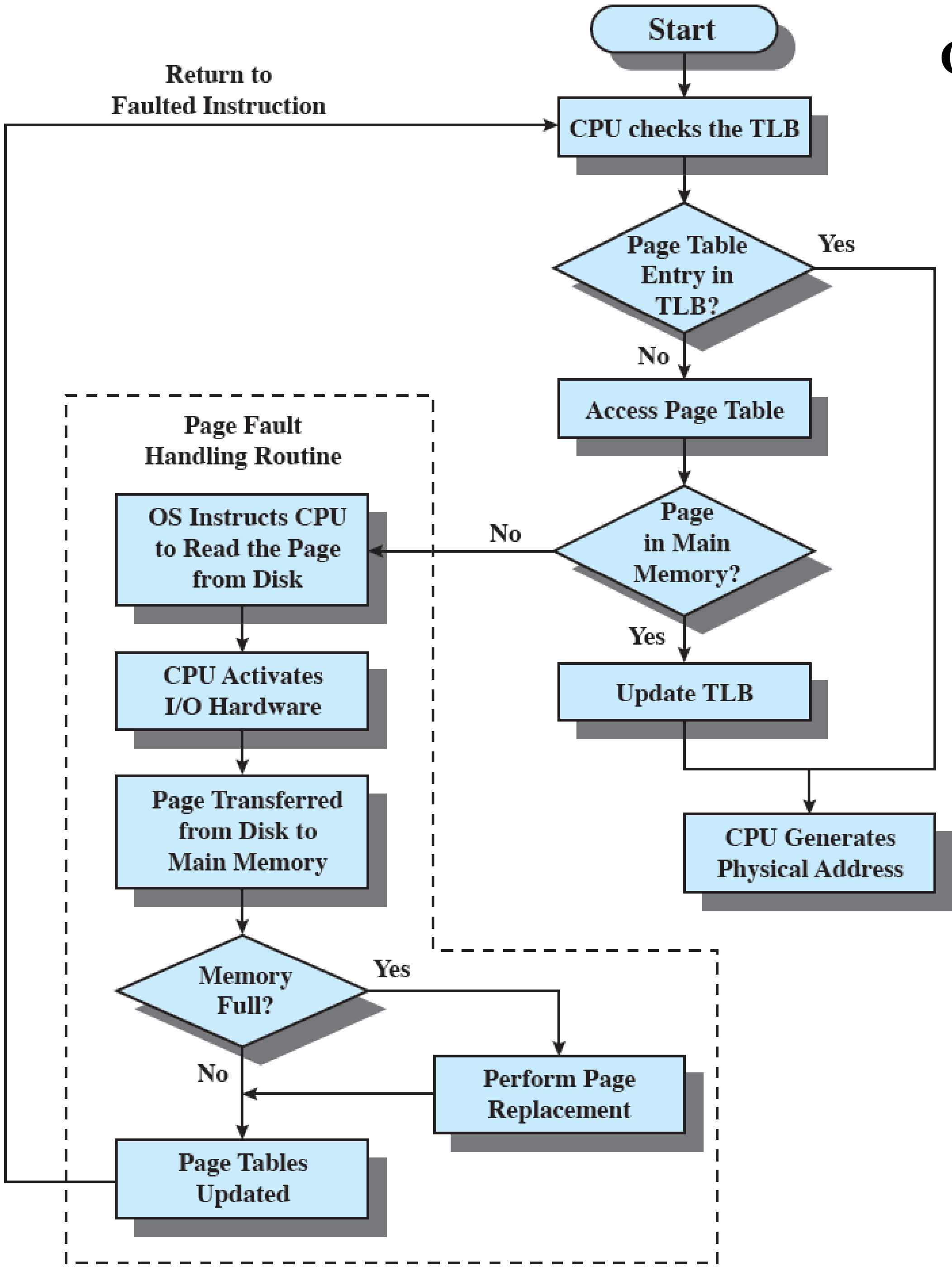
Translation Lookaside Buffer (TLB)

- **Each virtual memory reference can cause two physical memory accesses:**
  - one to fetch the page table entry
  - one to fetch the data
- **To overcome the effect of doubling the memory access time, most virtual memory schemes make use of a special high-speed cache called a translation lookaside buffer**



Use of a Translation Lookaside Buffer

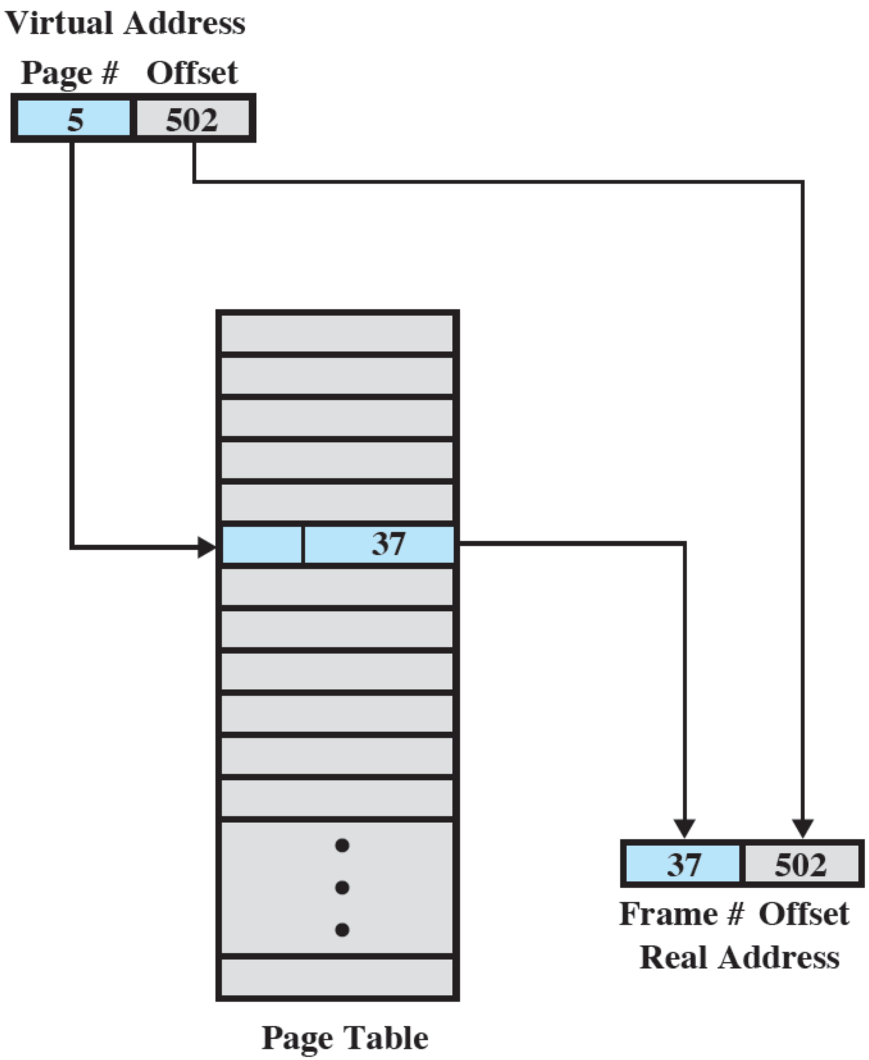
Operation of Paging and TLB



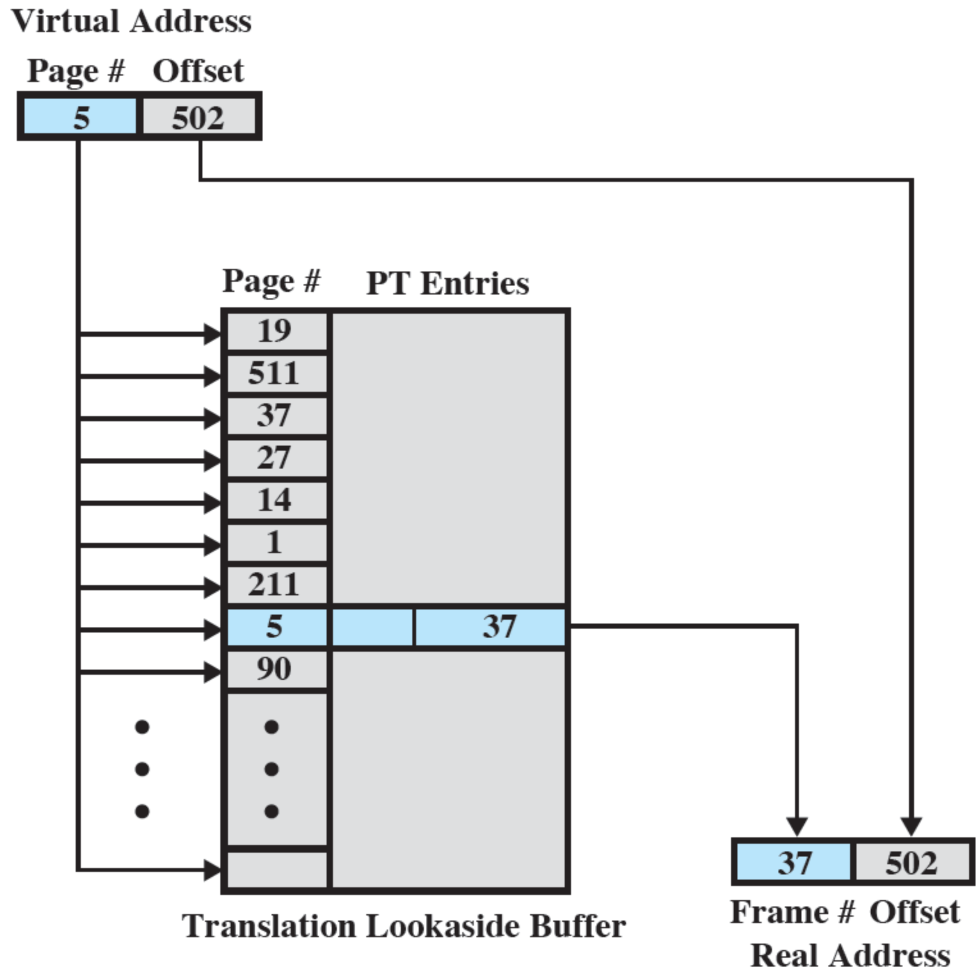
Associative Mapping

- **The TLB only contains some of the page table entries so we cannot simply index into the TLB based on page number**
  - each TLB entry must include the page number as well as the complete page table entry
- **The processor is equipped with hardware that allows it to interrogate simultaneously a number of TLB entries to determine if there is a match on page number**

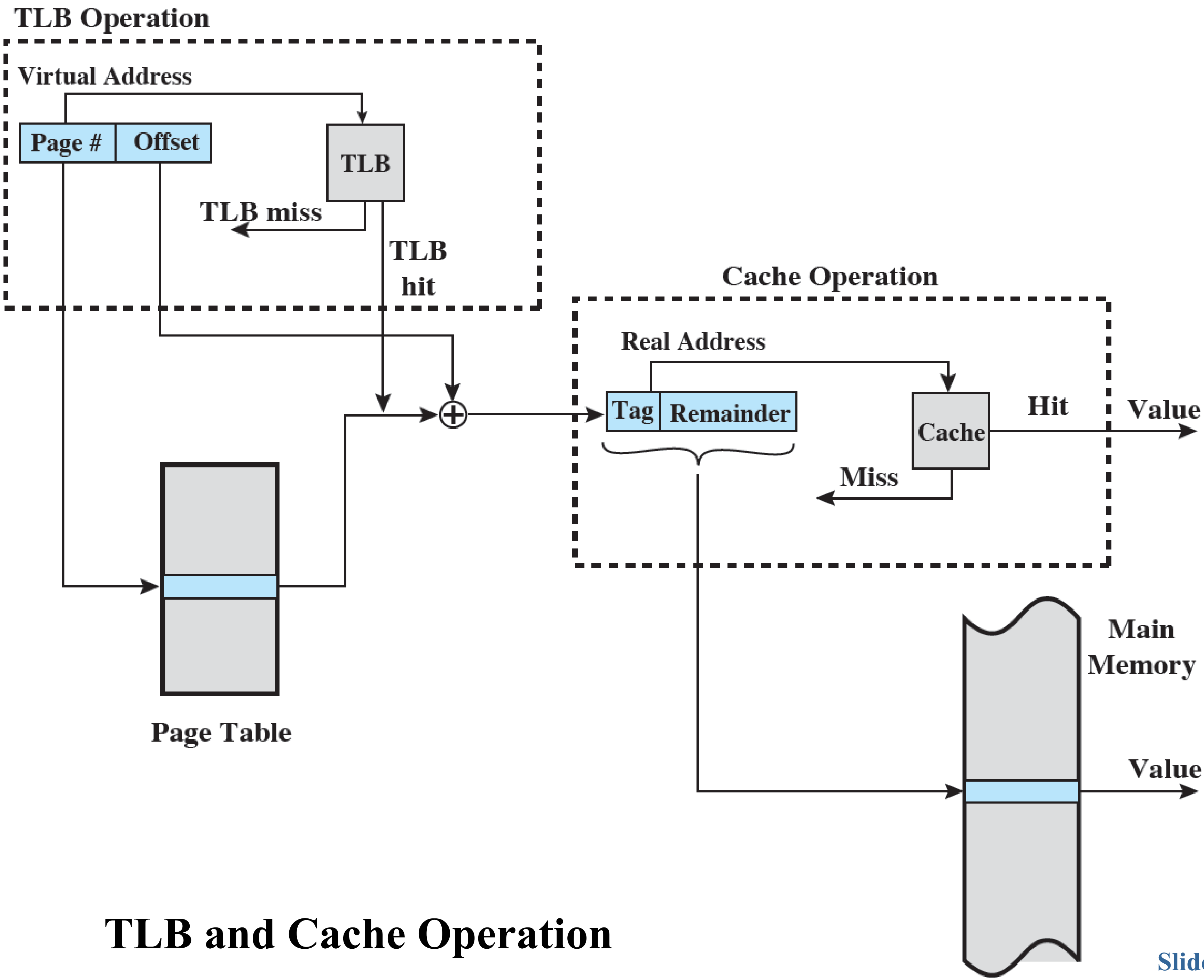
Direct Versus Associative Lookup for Page Table Entries



Direct mapping



Associative mapping

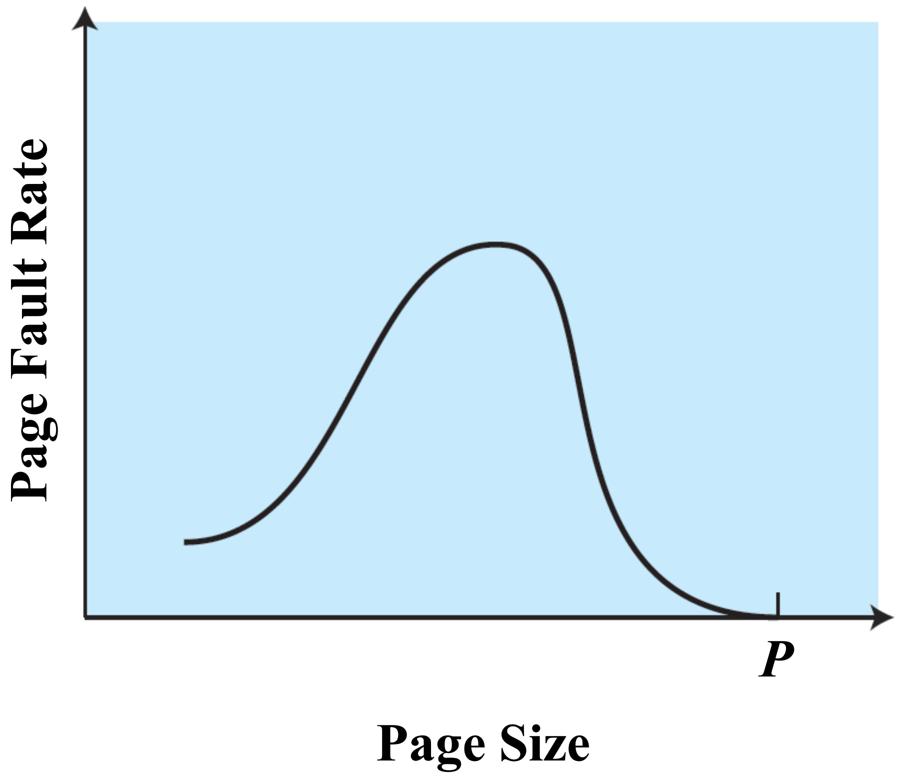


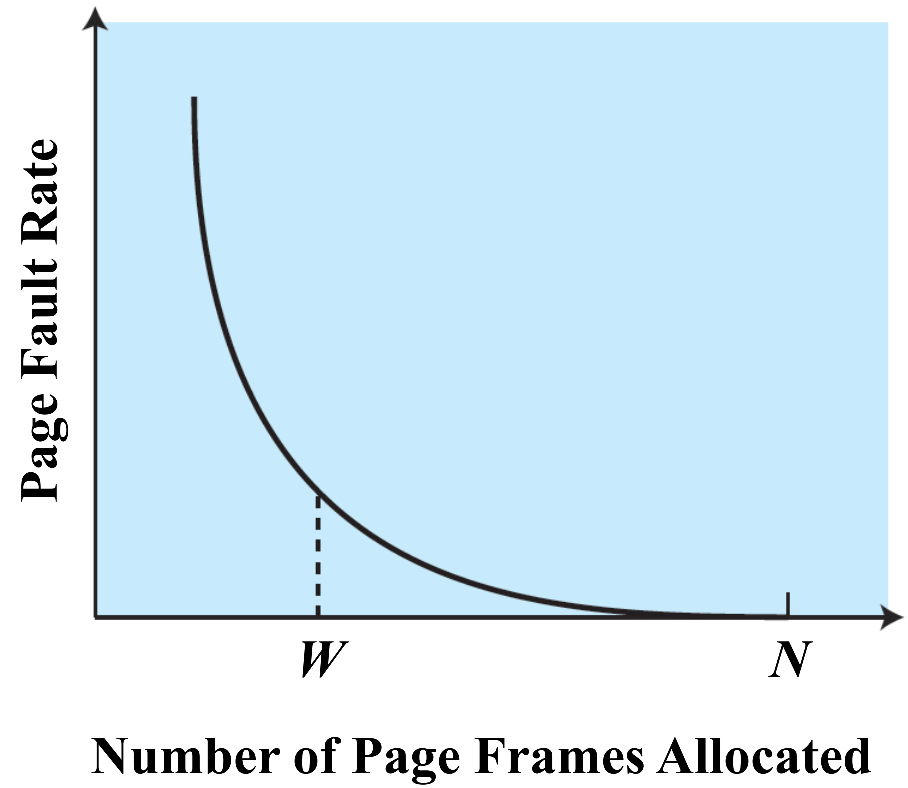
TLB and Cache Operation

Page Size

- **The smaller the page size, the lesser the amount of internal fragmentation**
  - however, more pages are required per process
  - more pages per process means larger page tables
  - for large programs in a heavily multiprogrammed environment some portion of the page tables of active processes must be in virtual memory instead of main memory
  - the physical characteristics of most secondary-memory devices favor a larger page size for more efficient block transfer of data

Paging Behavior of a Program





P = size of entire process
W = working set size
N = total number of pages in process

Example: Page Sizes

Computer	Page Size
Atlas	512 48-bit words
Honeywell-Multics	1024 36-bit words
IBM 370/XA and 370/ESA	4 Kbytes
VAX family	512 bytes
IBM AS/400	512 bytes
DEC Alpha	8 Kbytes
MIPS	4 Kbytes to 16 Mbytes
UltraSPARC	8 Kbytes to 4 Mbytes
Pentium	4 Kbytes or 4 Mbytes
IBM POWER	4 Kbytes
Itanium	4 Kbytes to 256 Mbytes

Page Size

The design issue of page size is related to the size of physical main memory and program size

main memory is getting larger and address space used by applications is also growing

most obvious on personal computers where applications are becoming increasingly complex

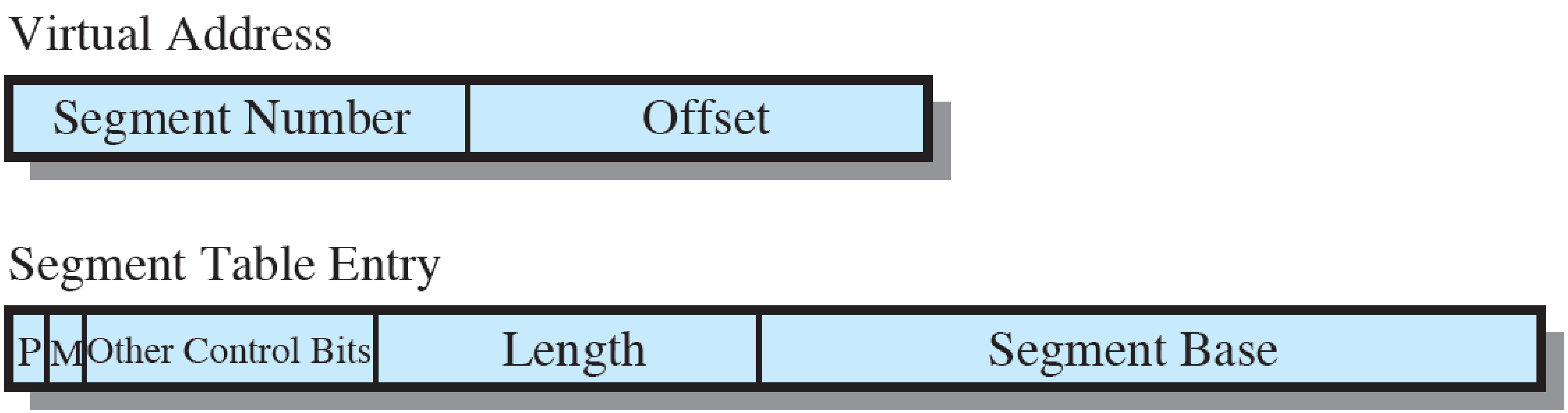
Contemporary programming techniques used in large programs tend to decrease the locality of references within a process

Segmentation

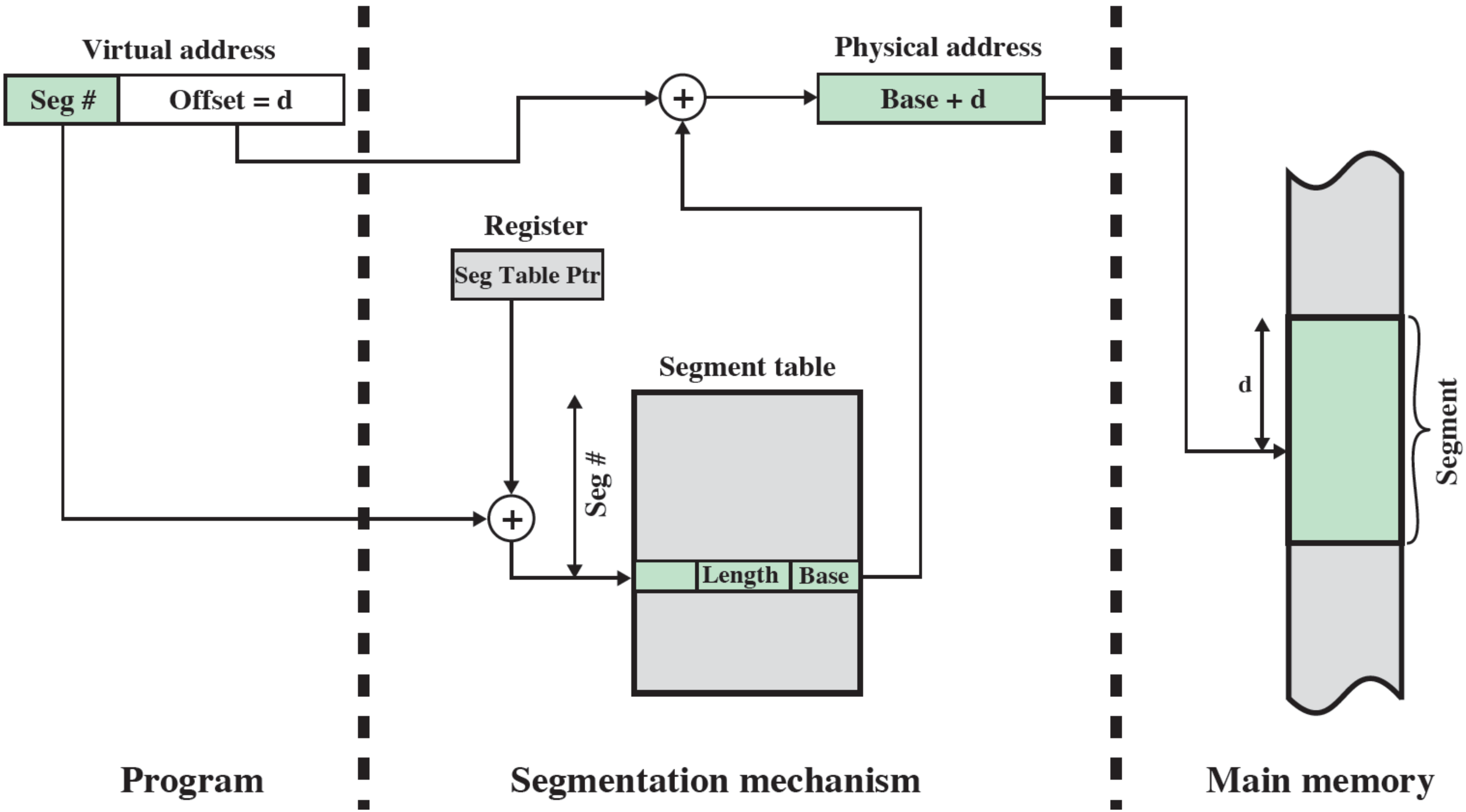
- **Segmentation**
  - allows the **programmer** to view memory as consisting of **multiple address** spaces or segments
  - Segments may be of **unequal**, indeed dynamic, **size**
- **Advantages**
  - simplifies handling of **growing** data structures
  - allows programs to be altered and **recompiled independently**
  - lends itself to sharing data among processes
  - lends itself to protection

Segment Organization

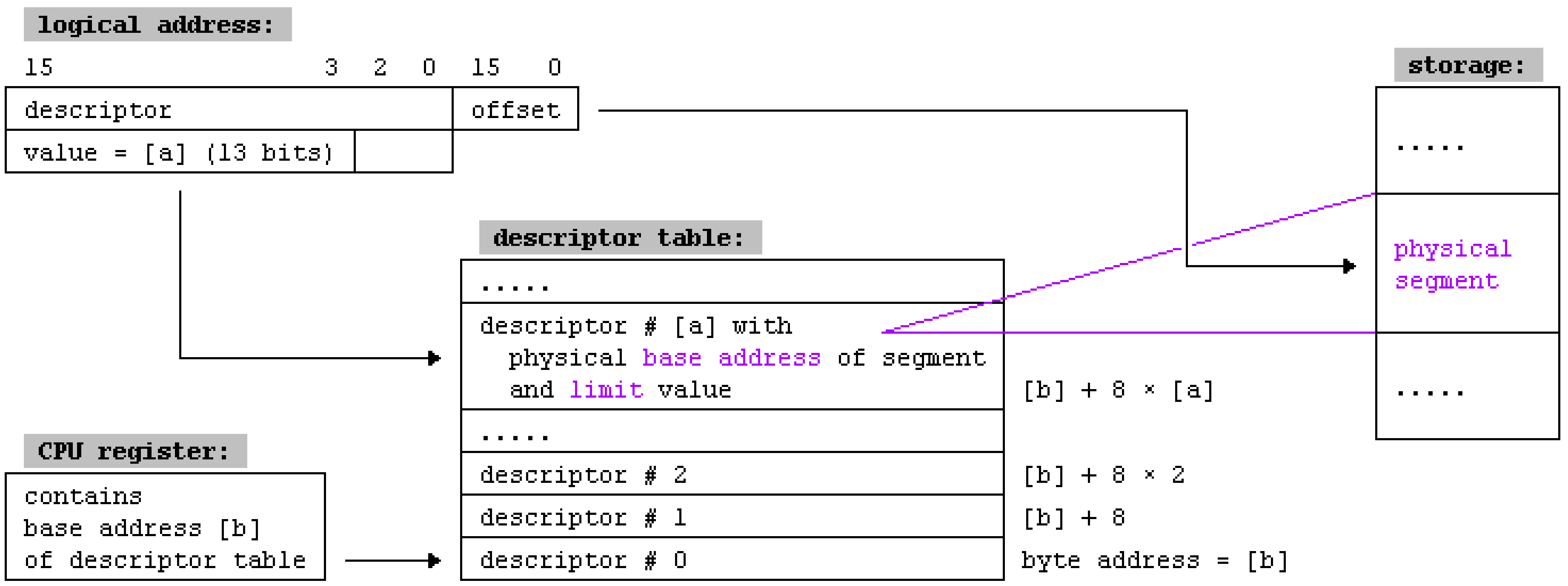
- **Each segment table entry contains**
  - the **starting address** of the corresponding segment in main memory
  - the **length** of the segment



Address Translation in a Segmentation System



Virtual segments of 80286



Combined Paging and Segmentation

In a combined paging/segmentation system a user's address space is broken up into a number of segments.

Each segment is broken up into a number of fixed-sized pages which are equal in length to a main memory frame

Segmentation is visible to the programmer

Paging is transparent to the programmer

Combined Segmentation and Paging

Virtual Address

Segment Number	Page Number	Offset

Segment Table Entry

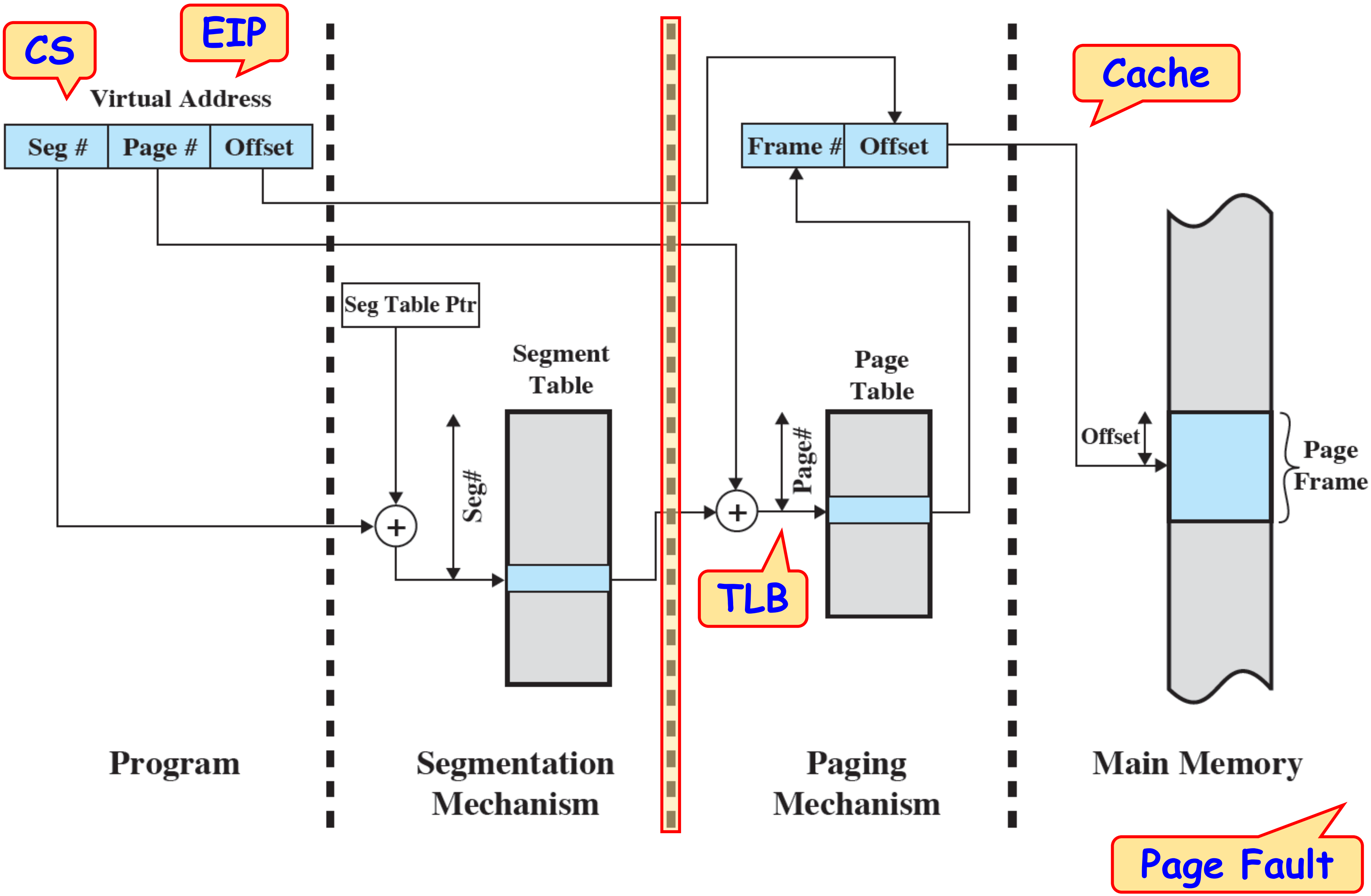
Control Bits	Length	Segment Base

Page Table Entry

P	M	Other Control Bits	Frame Number

P = Present bit

M = Modified bit



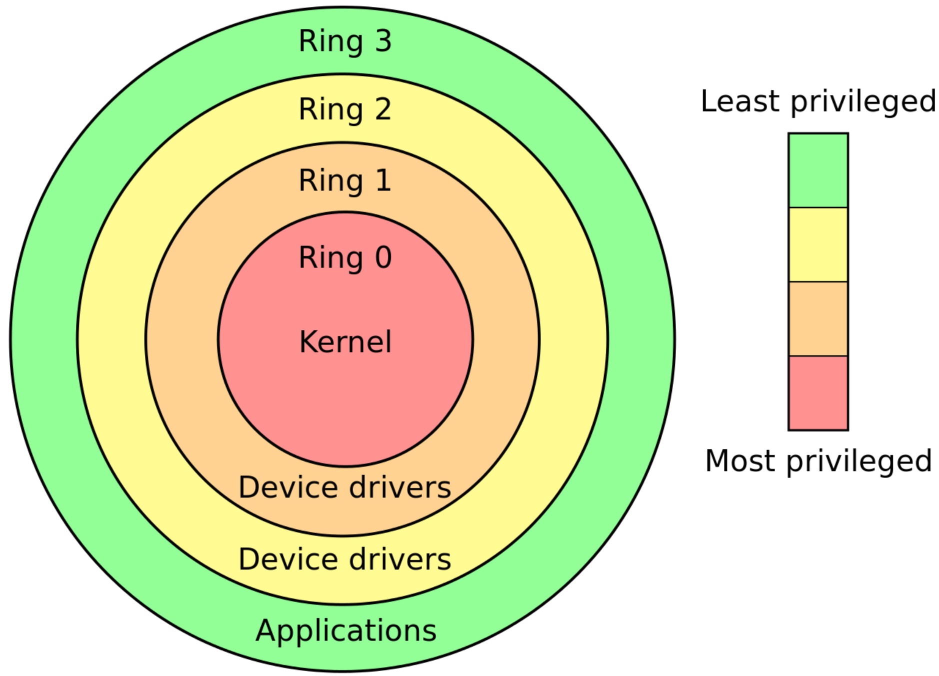
Address Translation in a Segmentation/Paging System

Protection and Sharing

- Segmentation lends itself to the implementation of protection and sharing policies
- Each entry has a base address and length so inadvertent memory access can be controlled
- Sharing can be achieved by segments referencing multiple processes

Use a ring-protection structure

- **Basic principles of the ring system are**
  - A program may **access only data** that reside on the **same** ring or a **less** privileged ring
  - A program may **call services** residing on the **same** or a **more** privileged ring



Privilege rings for the x86 available in protected mode

Summary

- **Desirable to:**
  - maintain as many processes in main memory as possible
  - free programmers from size restrictions in program development
- **With virtual memory:**
  - all address references are logical references that are translated at run time to real addresses
  - a process can be broken up into pieces
  - two approaches are paging and segmentation
  - management scheme requires both hardware and software support