Virtual Memory

control structures
and
hardware support

Hardware and Control Structures

- Memory references are dynamically translated into physical addresses at run time

  – A process may be swapped in and out of main memory such that it occupies different regions

- A process may be broken up into pieces that do not need to located contiguously in main memory

- Not necessarily all pieces of a process need to be loaded in main memory during execution

Execution of a Program

- Operating system brings into main memory **a few** pieces of the program
- **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 (**page fault**)
- Operating system places the process in a blocking state
  - the process is waiting its page from disk
  - this is equivalent to a blocking I/O request

Execution of a Program

- Piece of process that contains the logical address is brought into main memory
  - Operating system 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 complete which causes the operating system to place the affected process in the Ready state

Advantages of Breaking up a Process

- 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

Thrashing

- Swapping out a piece of a process just before that piece is needed
  - if this happens frequently a lot of I/O is needed

  - at the extreme point all processes are waiting for their pages from the disk

- The processor spends most of its time swapping pieces rather than executing user instructions

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
- Possible to make intelligent guesses about which pieces will be needed in the future
- This suggests that virtual memory may work efficiently

principle of locality

numeri di pagina

tempo di esecuzione
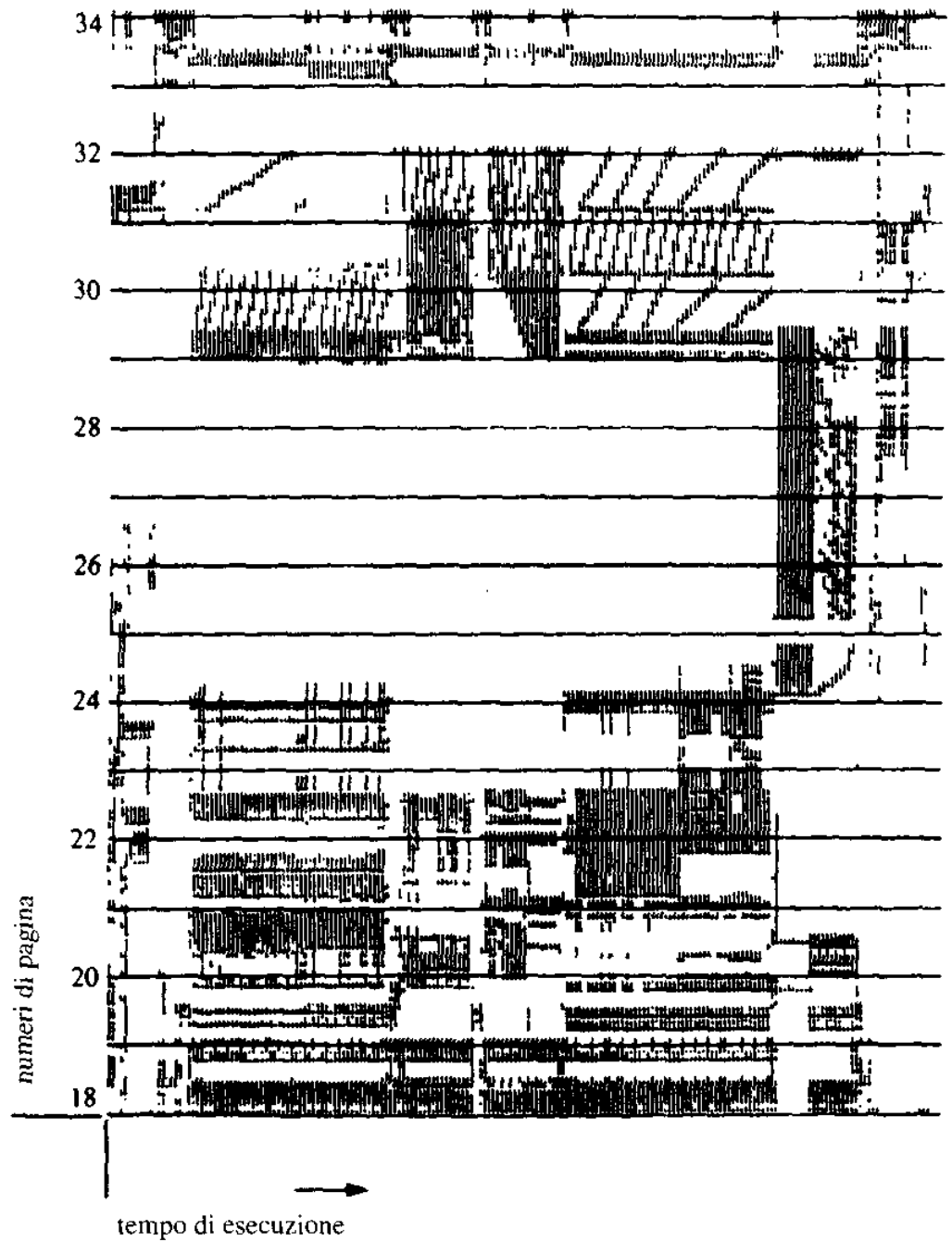
Support Needed for Virtual Memory

- Hardware must support paging and segmentation

- Operating system must be able to manage the movement of pages and/or segments between secondary memory and main memory

paging and virtual memory

- Each process has its own page table
- Each page table entry contains the frame number of the corresponding page in main memory
- An additional bit is needed to indicate whether the page is in main memory or not
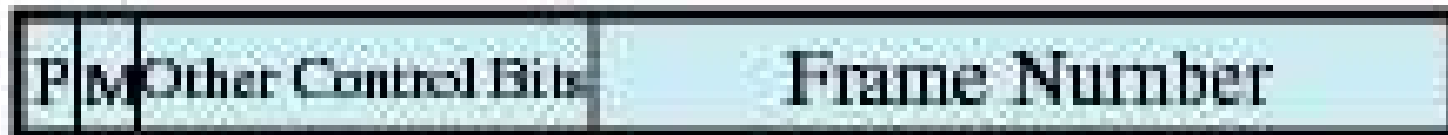
Modify Bit in
Page Table

- Modify bit is needed to indicate if the page has been altered since it was last loaded into main memory
  - If no change has been made, the page does not have to be written to the disk when it needs to be swapped out
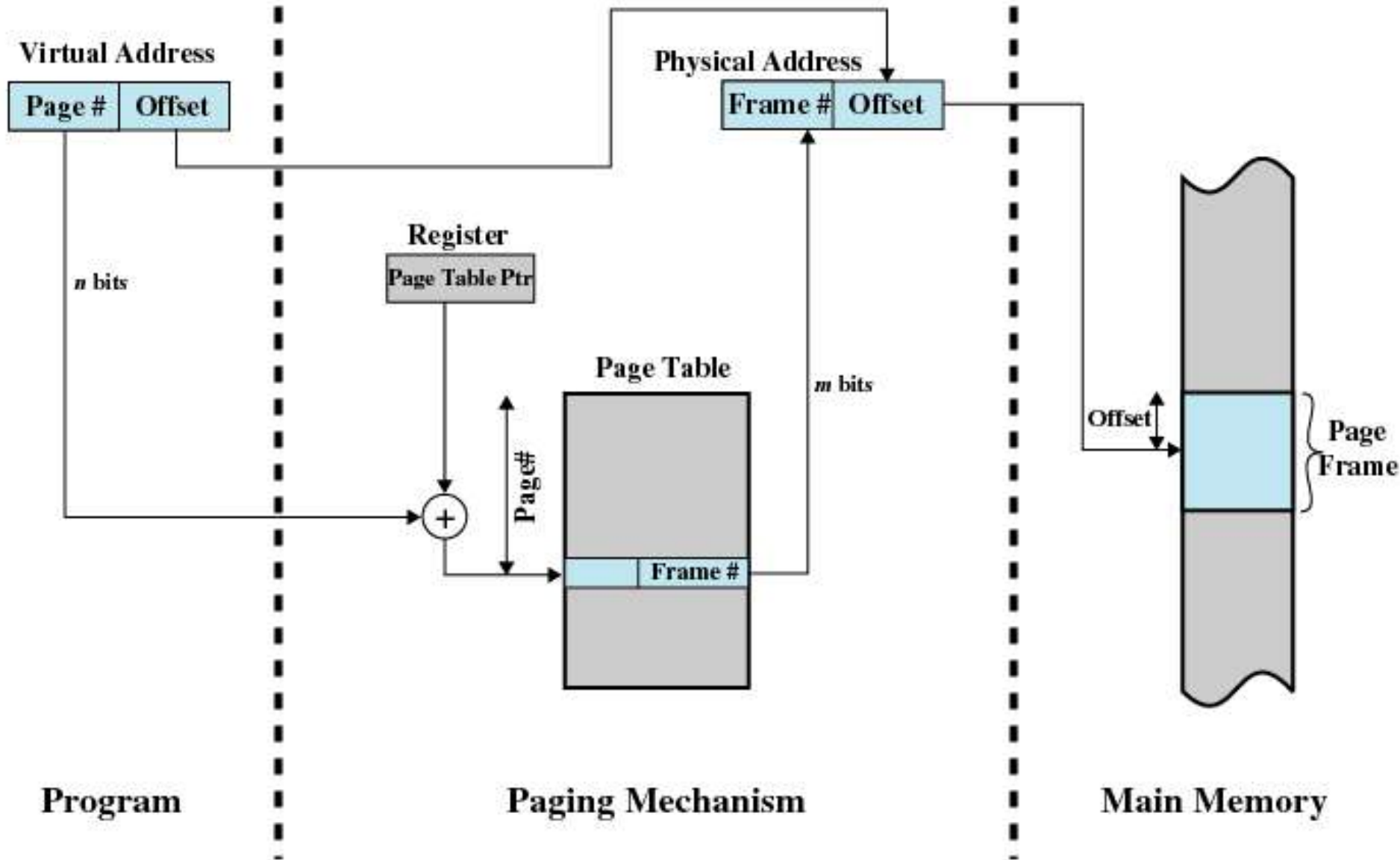
Paging

Virtual Address

Page Number	Offset

Page Table Entry

P	M	Other Control Bits	Frame Number

address translation for paging

very big page tables

- what if a process use a limited number of small parts of the page table?

  – other parts may be not used at the moment or not used at all

  – a lot of memory wasted for unused page table entries

- page tables should treated largely as part of the process image

- hierarchical page tables, inverted page tables
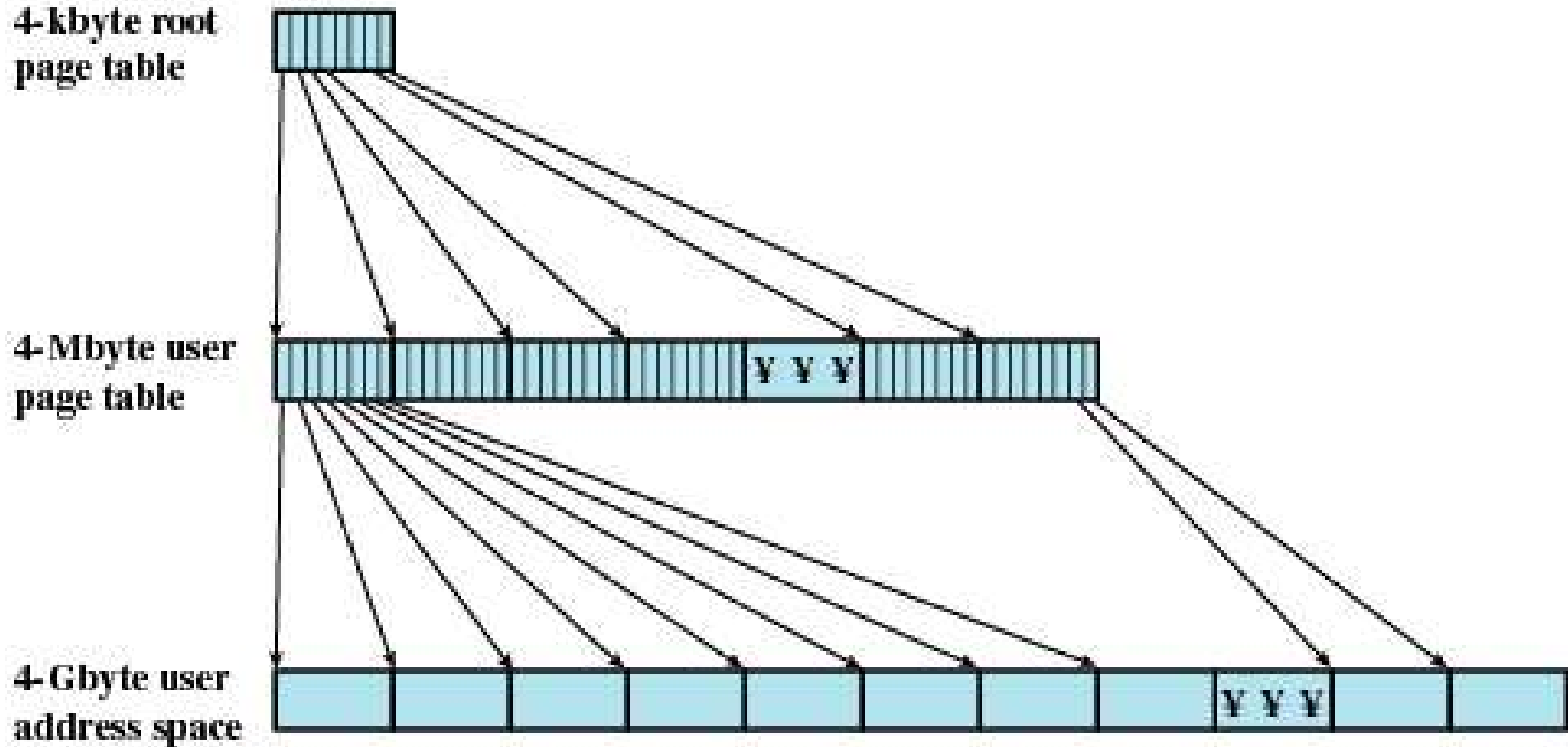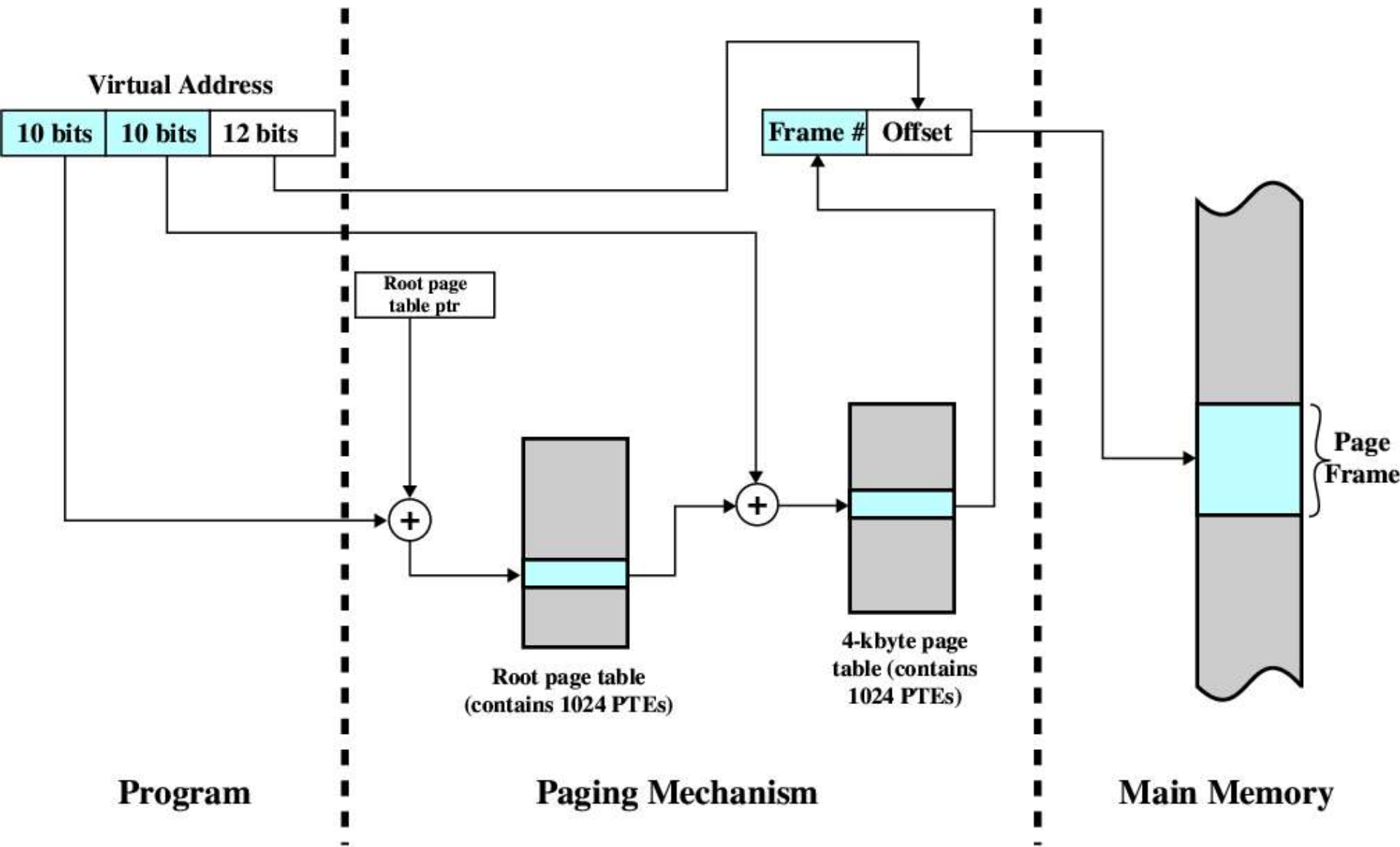
Two-Level Scheme for 32-bit Address (pentium like)

Figure 8.4 A Two-Level Hierarchical Page Table

address translation in a two-level schema

Virtual Address

10 bits	10 bits	12 bits

Frame #	Offset

Root page table ptr

Root page table
(contains 1024 PTEs)

4-kbyte page
table (contains
1024 PTEs)

Page Frame

Program

Paging Mechanism

Main Memory

Inverted Page Table

- one table and one entry for each frame regardless of the number of processes
- Page number portion of a virtual address is mapped into a hash value
- Hash value points to inverted page table
- collisions are solved by chaining
- Used on PowerPC, UltraSPARC, and IA-64 architecture

inverted page table

Virtual Address

n bits

| Page # | Offset |

n bits

hash function

m bits

Control bits

Page #	Process ID	Chain	
			0
			i
			j
			$2^m - 1$

Inverted Page Table
(one entry for each physical memory frame)

| Frame # | Offset |

m bits

Real Address

18
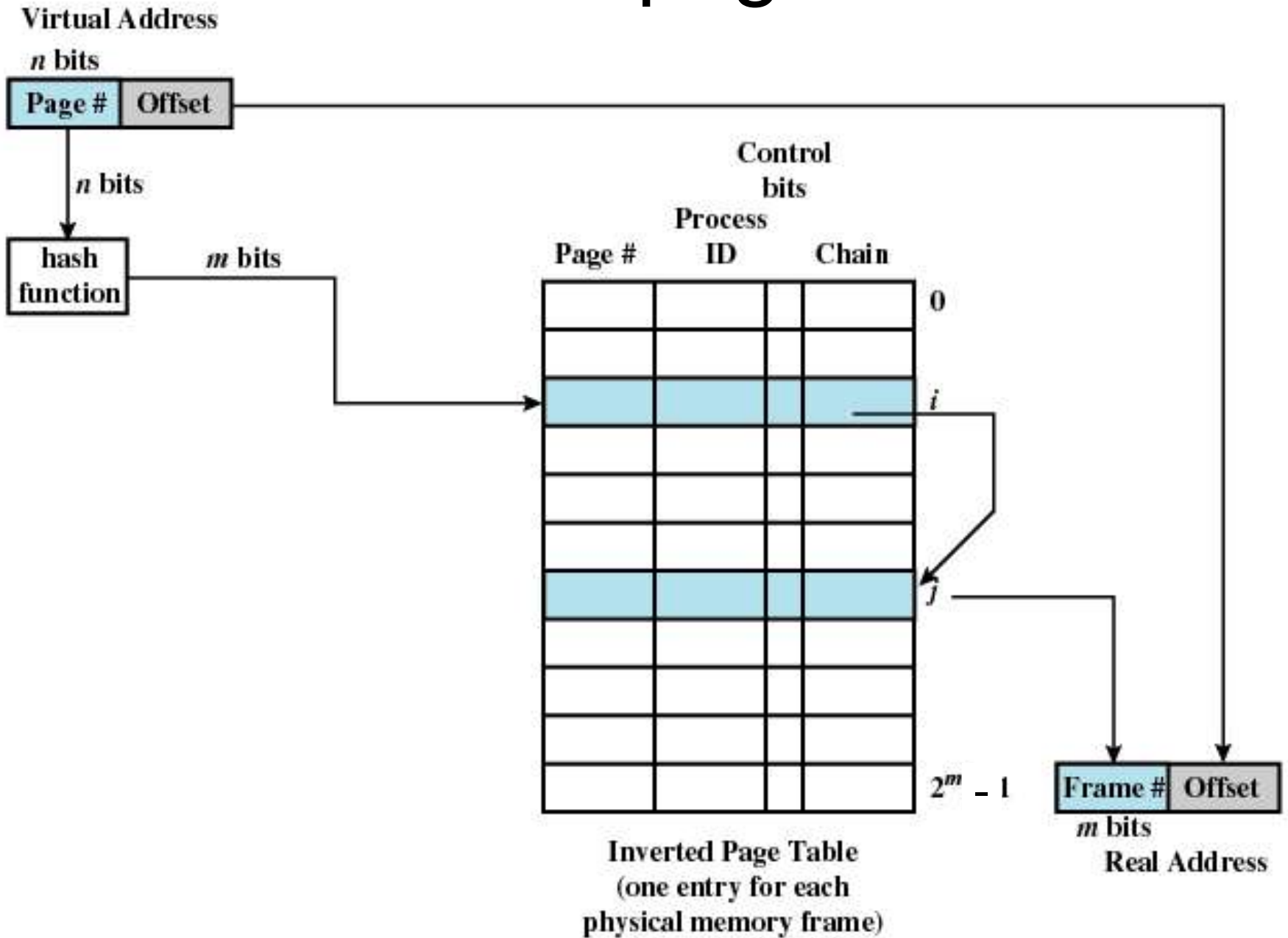
Translation Lookaside Buffer

- Each virtual memory reference can cause two physical memory accesses
  - One to fetch the page table entry
  - One to read/write the data

- To overcome this problem a **high-speed cache** is set up for page table entries
  - Called a Translation Lookaside Buffer (TLB)

Translation Lookaside Buffer

- Contains page table entries that have been most recently used
- it performs an **associative mapping** between page numbers and page table entries
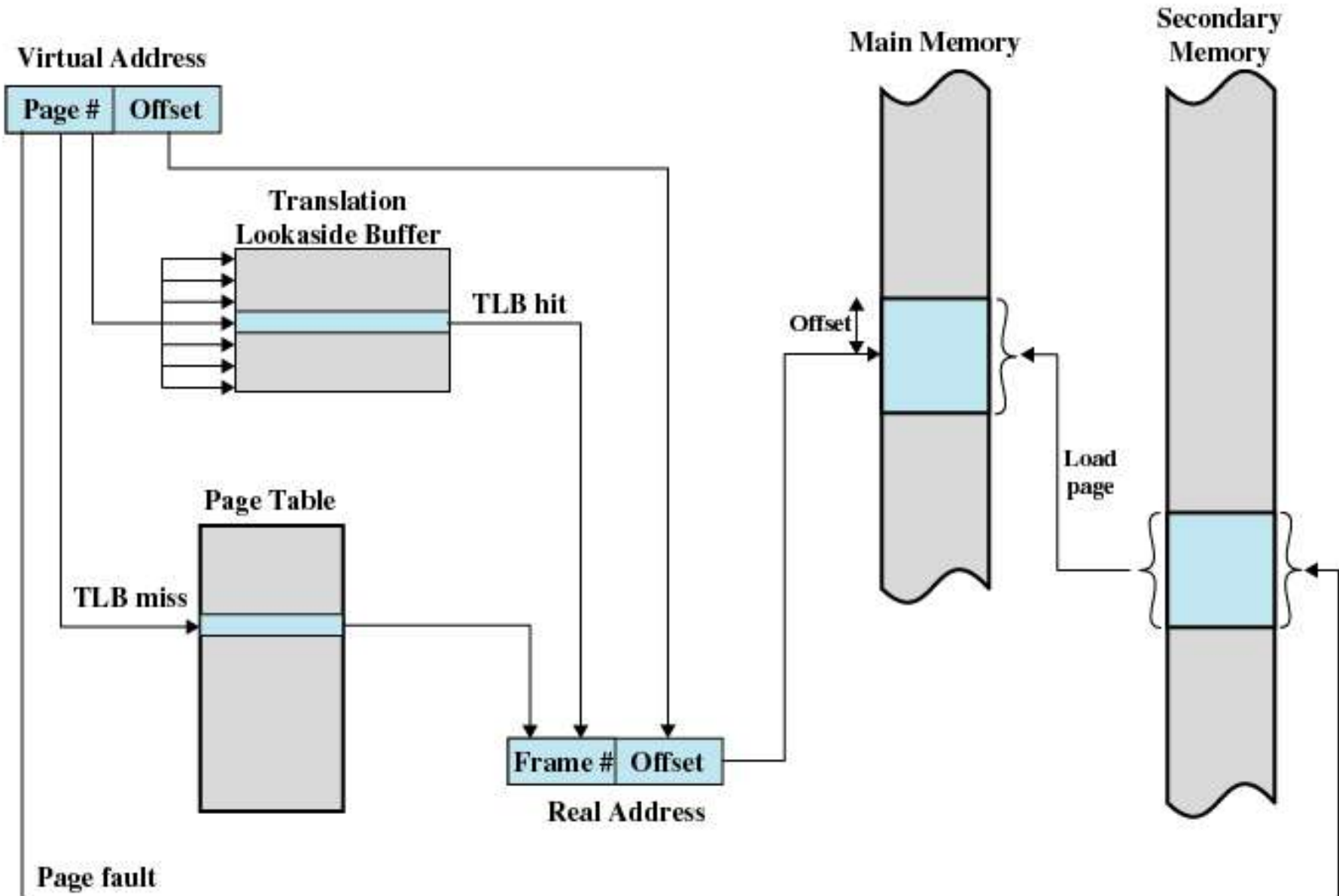
Translation Lookaside Buffer

- Given a virtual address, processor examines the TLB

- If page table entry is present (TLB hit), the frame number is retrieved and the real address is formed

- If page table entry is not found in the TLB (TLB miss), the page number is used to index the process page table
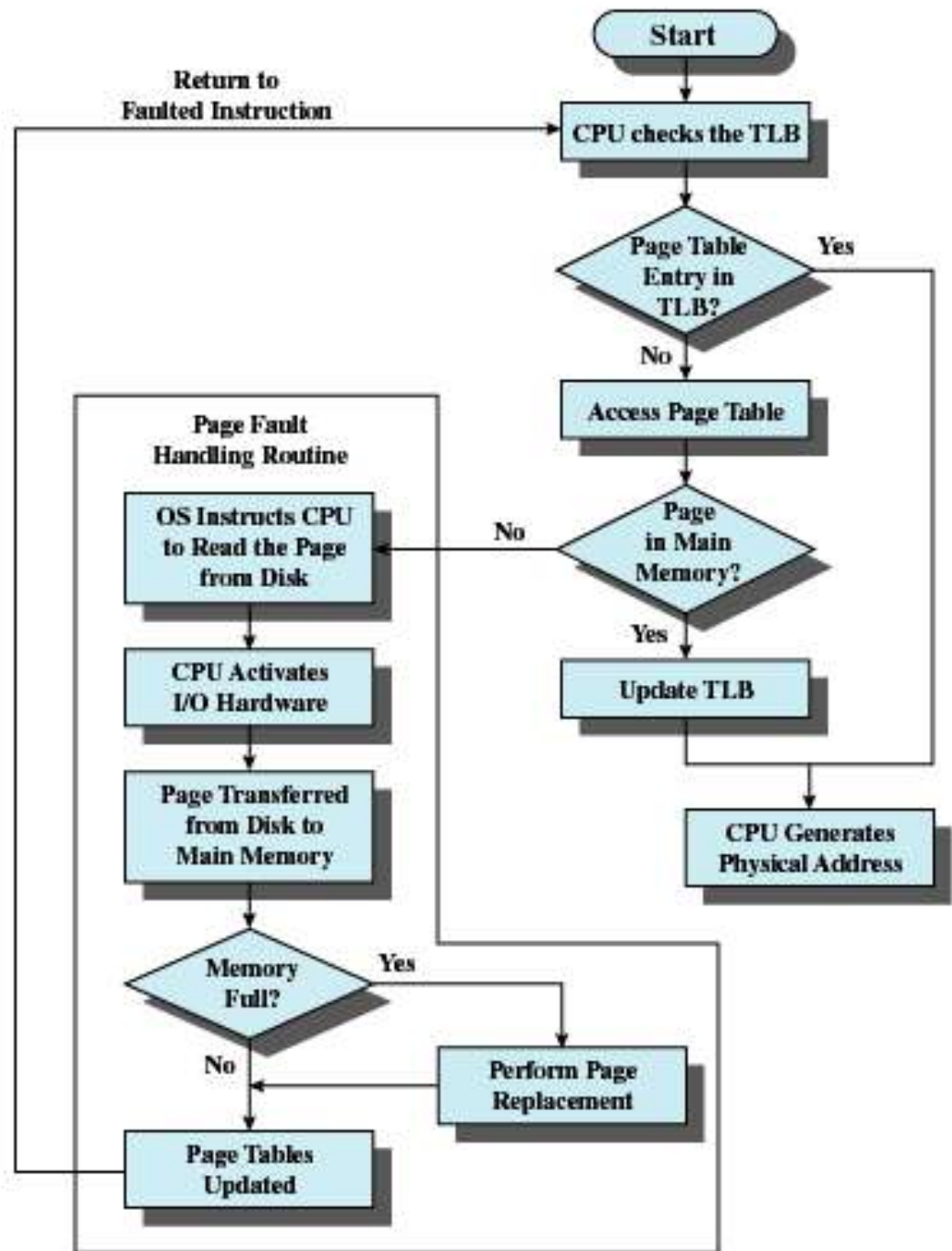
Translation Lookaside Buffer

- if page is already in main memory the TLB is updated to include the new page entry
  - If not in main memory a page fault is issued
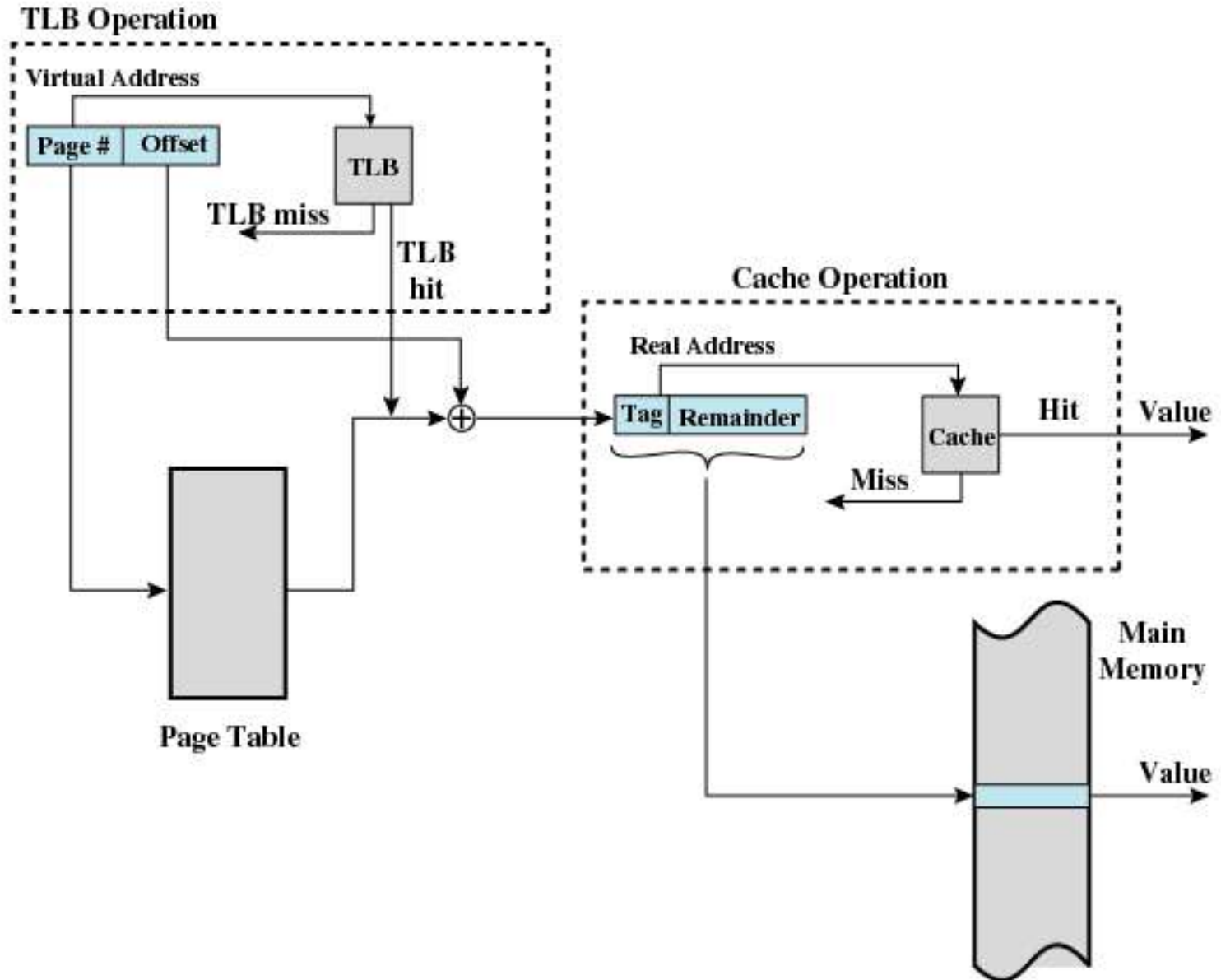 - OS is called

address translation with TLB

Virtual Address

Page #	Offset

Translation Lookaside Buffer

TLB hit

Page Table

TLB miss

Main Memory

Offset

Load page

Secondary Memory

Frame #	Offset

Real Address

Page fault

23

lookup algorithm for virutal memory paging with TLB

direct vs. associative mapping

(a) Direct mapping

(b) Associative mapping

TLB and regular cache

© 2005 maurizio pizzonia - sistemi operativi a.a. 2004-2005

Page Size

- Smaller page size, less amount of internal fragmentation
- Smaller page size, more pages required per process
- More pages per process means larger page tables
- Larger page tables means large portion of page tables in virtual memory
- Secondary memory is designed to efficiently transfer large blocks of data so a large page size is better

Page Size

- Small page size, large number of pages will be found in main memory

- As time goes on during execution, the pages in memory will all contain portions of the process near recent references. Page faults low.

- Increased page size causes pages to contain locations further from any recent reference. Page faults rise.

typical paging behavior

anomalous

typical

(a) Page Size

(b) Number of Page Frames Allocated
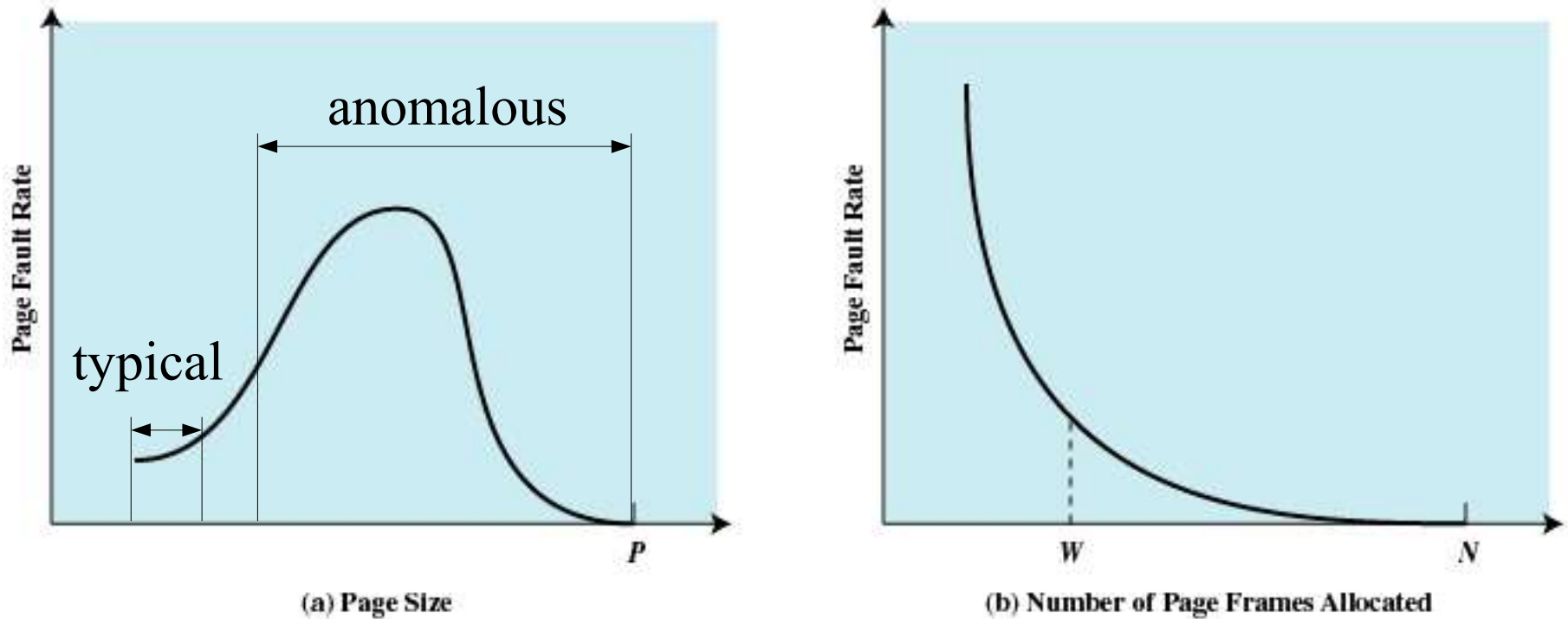
P = size of entire process
W = working set size
N = total number of pages in process

Figure 8.11 Typical Paging Behavior of a Program

Example Page Sizes

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

30

Segmentation

- Segments may have be unequal size
- segment size may dynamically increase
  - may simplify handling of growing data structures

- Allows modules of programs to be altered and recompiled independently
- makes easy to share data among processes
- implements protection mechanisms

Segment Tables

- one entry for each segment of the process
- each entry contains
  - base address for the segment in main memory
  - the length of the segment
- A bit is needed to determine if segment is already in main memory
- Another bit is needed to determine if the segment has been modified since it was loaded in main memory
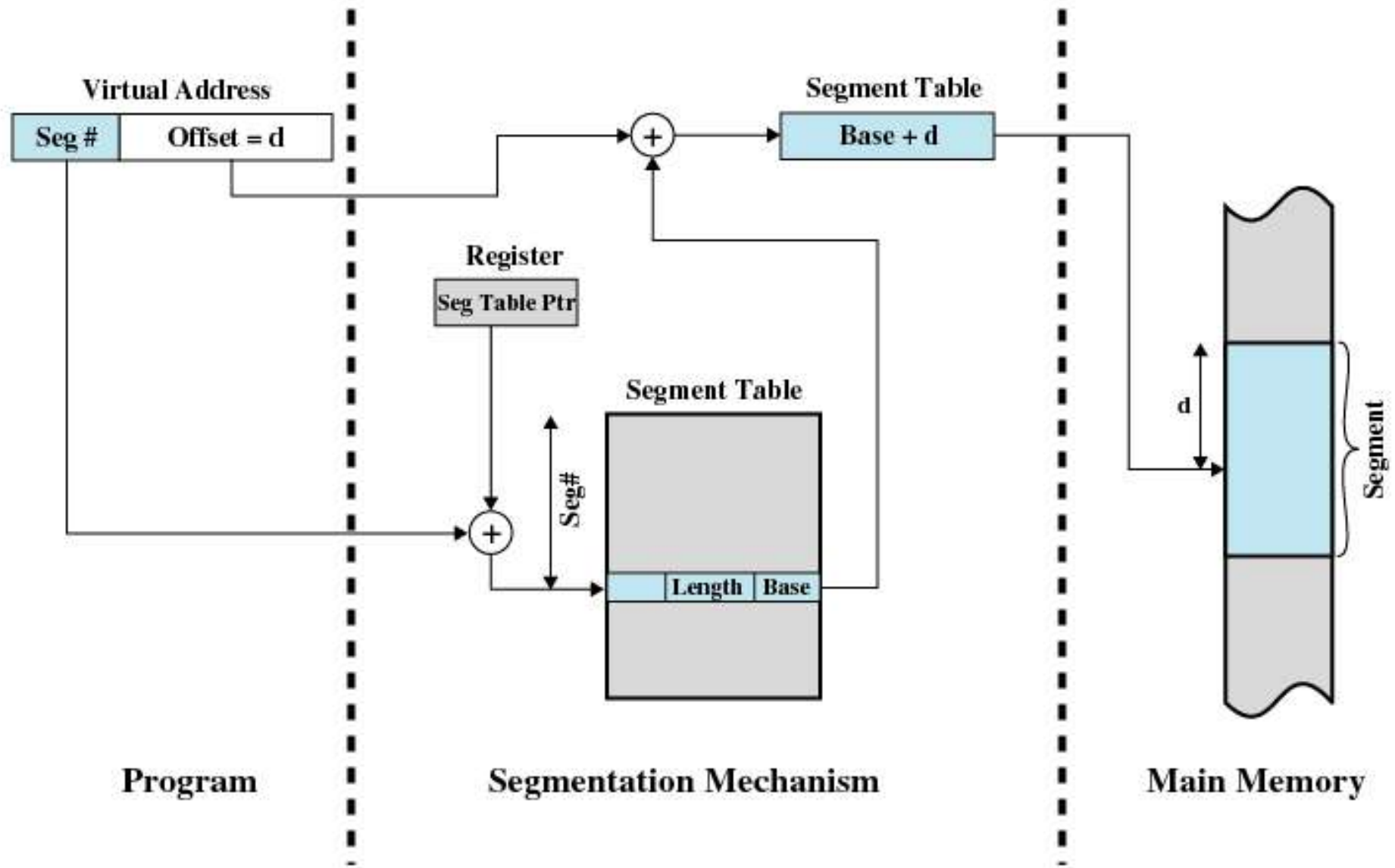
Segment Table Entries

Virtual Address

Segment Number	Offset

Segment Table Entry

P	M	Other Control Bits	Length	Segment Base
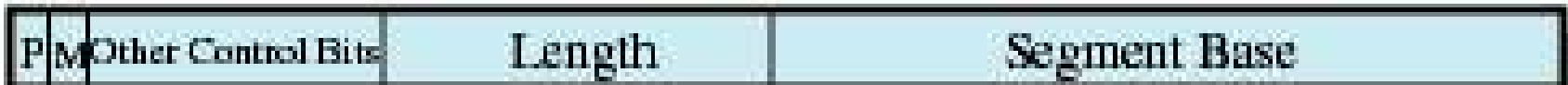
(b) Segmentation only

address translation for segmentation
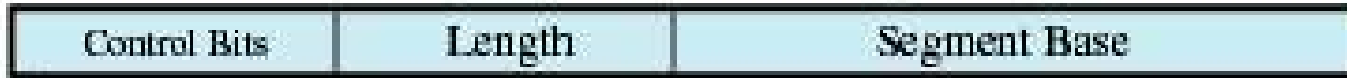
34

Combined Paging and Segmentation

- Paging is transparent to the programmer

- Segmentation is visible to the programmer

- Each segment is broken into fixed-size pages
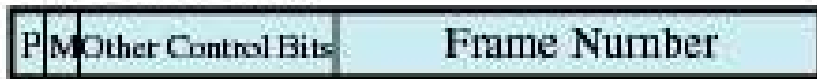
Combined Segmentation and Paging

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

(c) Combined segmentation and paging

address translation for segmentation/paging systems

Virtual Address

| Seg # | Page # | Offset |

Frame # | Offset

Seg Table Ptr

Segment Table

Page Table

Seg#

Page#

Offset

Page Frame

Program

Segmentation Mechanism

Paging Mechanism

Main Memory