Griffith UNIVERSITY

2803ICT

Memory Management

based on
William Stallings
Chapter 7 and 8

TOPICS

1) Introduction and overview
2) Advanced C programming
3) Memory management
4) File system and device IO
5) Socket programming
6) Processes
7) Interprocess communication (IPC)
8) Multithreading
9) Synchronisation (*2 weeks*)
10) Distributed software

Memory Management

- Subdividing memory to **accommodate multiple processes**
- Memory needs to be allocated to ensure a reasonable supply of ready processes to **utilise available processor time**

Related Terms

- **Frame**

  A **fixed-length** block of **main** memory.

- **Page**

  A **fixed-length** block of **data** that resides in **secondary** memory (such as disk). A page of data may temporarily be copied into a frame of main memory.

- **Segment**

  A **variable-length** block of **data** that resides in **secondary** memory. An entire segment may temporarily be copied into an available region of main memory (**segmentation**) or the segment may be divided into pages which can be individually copied into main memory (combined segmentation and paging)

Requirements

- Memory management is intended to satisfy the following requirements:
  - Relocation
  - Protection
  - Sharing
  - Logical organisation
  - Physical organisation

Requirements

- Memory management is intended to satisfy the following requirements:
  - Relocation
  - Protection
  - Sharing
  - Logical organisation
  - Physical organisation

Relocation

- **The need to relocate processes to a different area of memory**
- Programmers typically do not know in advance which other programs will be resident in main memory at the time of execution of their program
- Active processes need to be able to be swapped in and out of main memory in order to maximise processor utilisation

- Specifying that a process must be placed in the same memory region when it is swapped back in would be limiting

Addressing Requirements

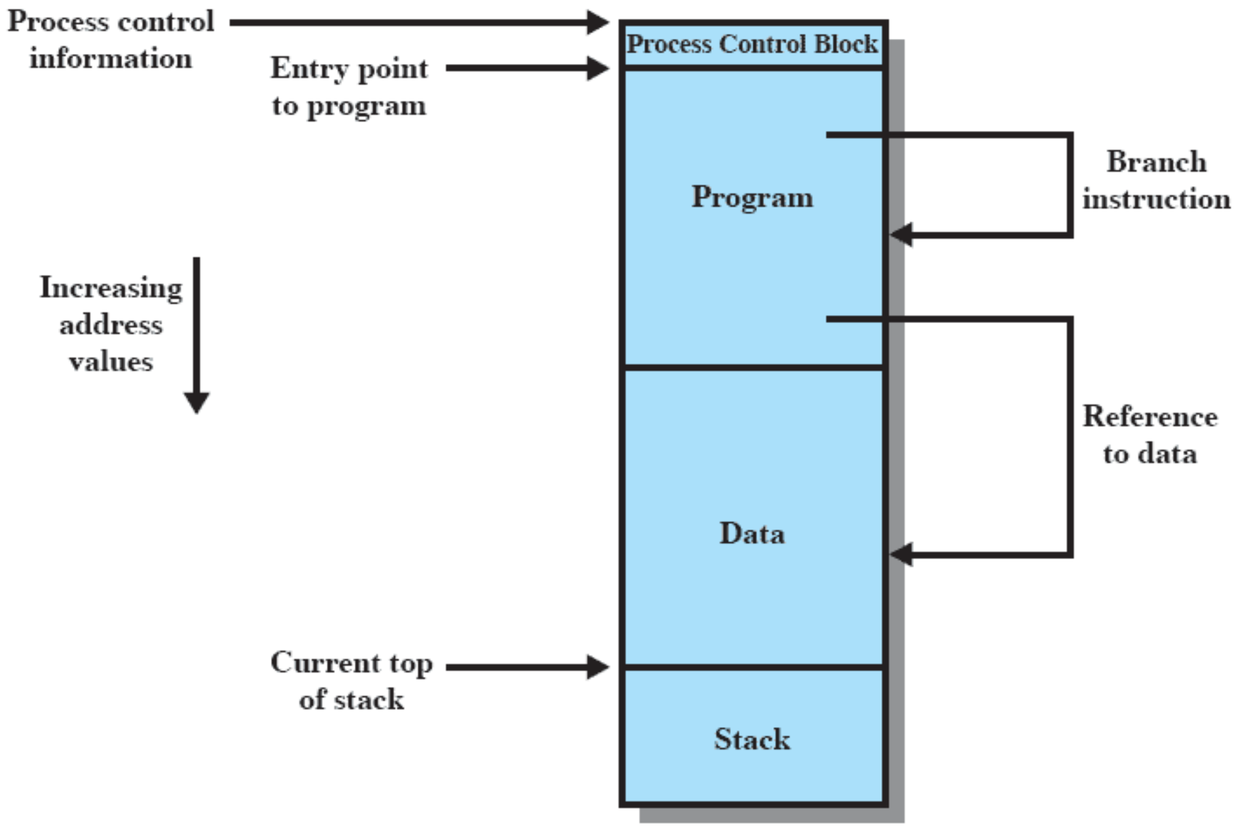


Figure 7.1 Addressing Requirements for a Process

Requirements

- Memory management is intended to satisfy the following requirements:
  - Relocation
  - Protection
  - Sharing
  - Logical organisation
  - Physical organisation

Protection

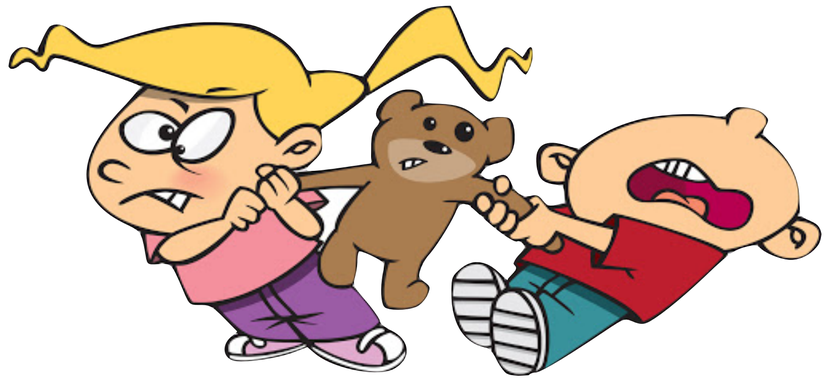
- Each process should be protected against unwanted interference by other processes
- Thus, processes need to acquire **permission** to reference memory locations for reading or writing purposes
- Location of a program in main memory is unpredictable
- Memory references generated by a process must be **checked at run time**
- Mechanisms that support relocation also support protection

Requirements

- Memory management is intended to satisfy the following requirements:
  - Relocation
  - Protection
  - Sharing
  - Logical organisation
  - Physical organisation

Sharing

- Advantageous to allow **each process access to the same copy** of the program rather than have their own separate copy
- Memory management must allow **controlled access** to shared areas of memory **without compromising protection**
- Mechanisms used to support relocation support sharing capabilities



Requirements

- Memory management is intended to satisfy the following requirements:
  - Relocation
  - Protection
  - Sharing
  - Logical organisation
  - Physical organisation

Logical Organisation

- Memory is organised as **linear**
- However, programs are written in modules

  modules can be written and compiled independently

  different degrees of protection given to modules (read-only, execute-only).

  sharing on a module level corresponds to the user's way of viewing the problem.

- **Segmentation** is the tool that most readily satisfies requirements

Requirements

- Memory management is intended to satisfy the following requirements:
  - Relocation
  - Protection
  - Sharing
  - Logical organisation
  - Physical organisation

Physical Organisation

- At least 2 levels of memory - **main** and **secondary**
- The flow of information between main and secondary memory is a major **system** concern
- Cannot leave the programmer with the responsibility to manage memory
  - Memory available for a program plus its data may be insufficient
  - Programmer does not know how much space will be available

Griffith UNIVERSITY

PARTITIONING

MEMORY PARTITIONING TECHNIQUES

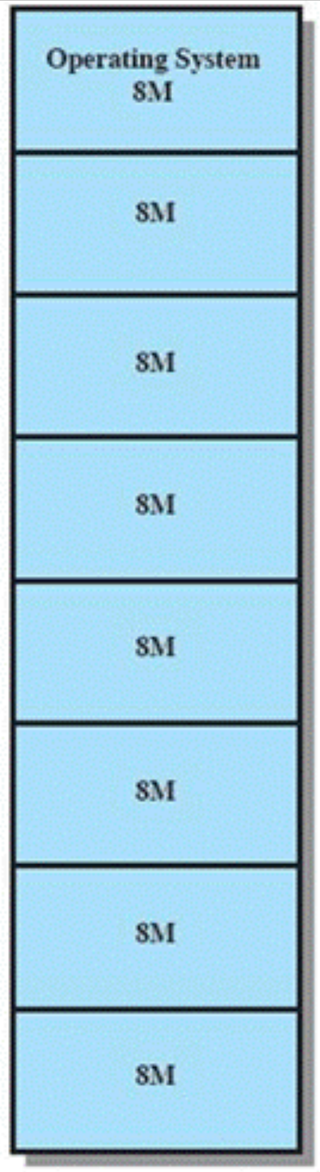
Memory Management Techniques

Griffith UNIVERSITY

- Partitioning
  - fixed vs. dynamic
  - equal vs. unequal size partitions
- Paging
  - simple
  - virtual memory
- Segmentation
  - simple
  - virtual memory

Fixed Partitioning

- **Equal-size partitions**
  - any process whose size is less than or equal to the partition size can be loaded into an available partition
- The operating system can swap out a process if all partitions are full and no process is in the Ready or Running state



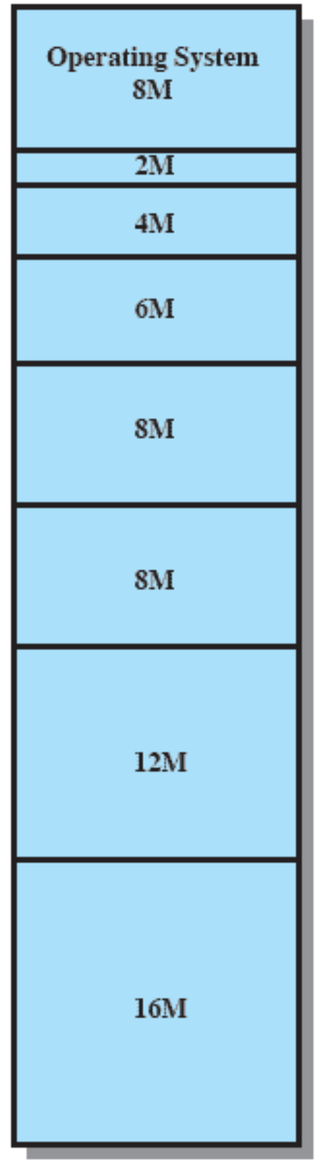
(a) Equal-size partitions

Fixed Partitioning - disadvantages

- A program may be **too big** to fit in a partition
  - program needs to be designed with the use of overlays
- **Main memory utilisation is inefficient**
  - any program, regardless of size, occupies an entire partition (**internal fragmentation**)
  - **wasted space** due to the block of data loaded being smaller than the partition

Unequal Size Partitions

- Using unequal size partitions helps lessen the problems
- programs up to16M can be accommodated without overlays
- partitions smaller than 8M allow smaller programs to be accommodated with less internal fragmentation



(b) Unequal-size partitions

Unequal size Partitioning - disadvantages

- The number of partitions specified at system generation time limits the number of active processes in the system
- Small jobs will not utilise partition space efficiently

Dynamic Partitions

- Partitions are of **variable length** and **number**
- Process is allocated **exactly** as much memory as it requires
- This technique was used by IBM’s mainframe operating system, OS/MVT

Effect of Dynamic Partitioning

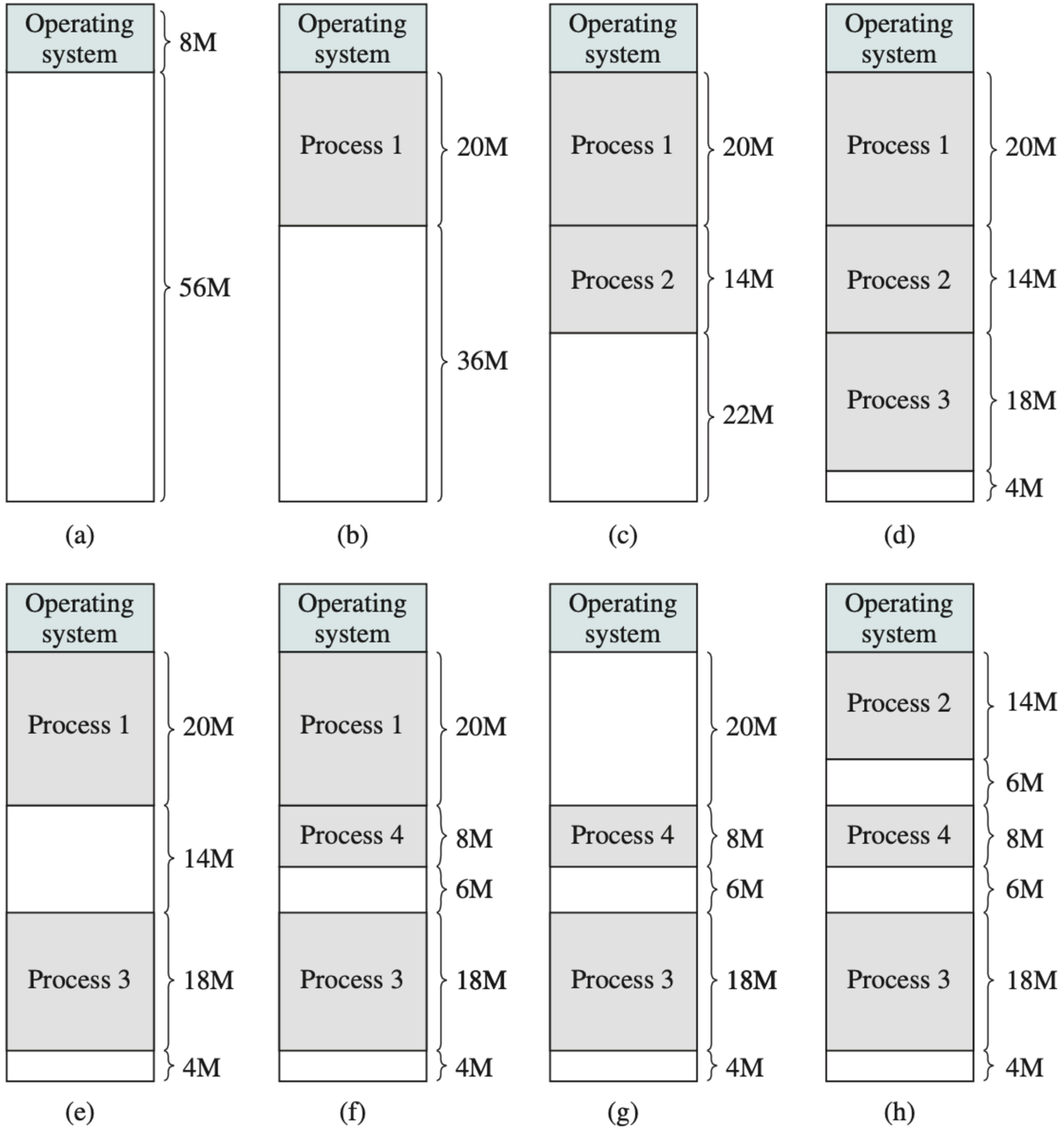


Figure 7.4 The Effect of Dynamic Partitioning

Dynamic Partitions

- **External Fragmentation**
  - memory becomes more and more fragmented
  - memory utilisation declines

- **Compaction**
  - technique for overcoming external fragmentation
  - OS shifts processes so that they are contiguous
  - free memory is together in one block
  - time consuming and wastes CPU time

Address Types

- **Logical**
    - reference to a memory location independent of the current assignment of data to memory
- **Relative**
    - address is expressed as a location relative to some known point
- **Physical or Absolute**
    - actual location in main memory

Relocation

- When a program is loaded into memory the **actual (absolute) memory locations are determined**
- A process may occupy **different partitions** which means different absolute memory locations **during execution** (from swapping)
- Compaction will also cause a program to occupy a different partition

Relocation with Hardware Support

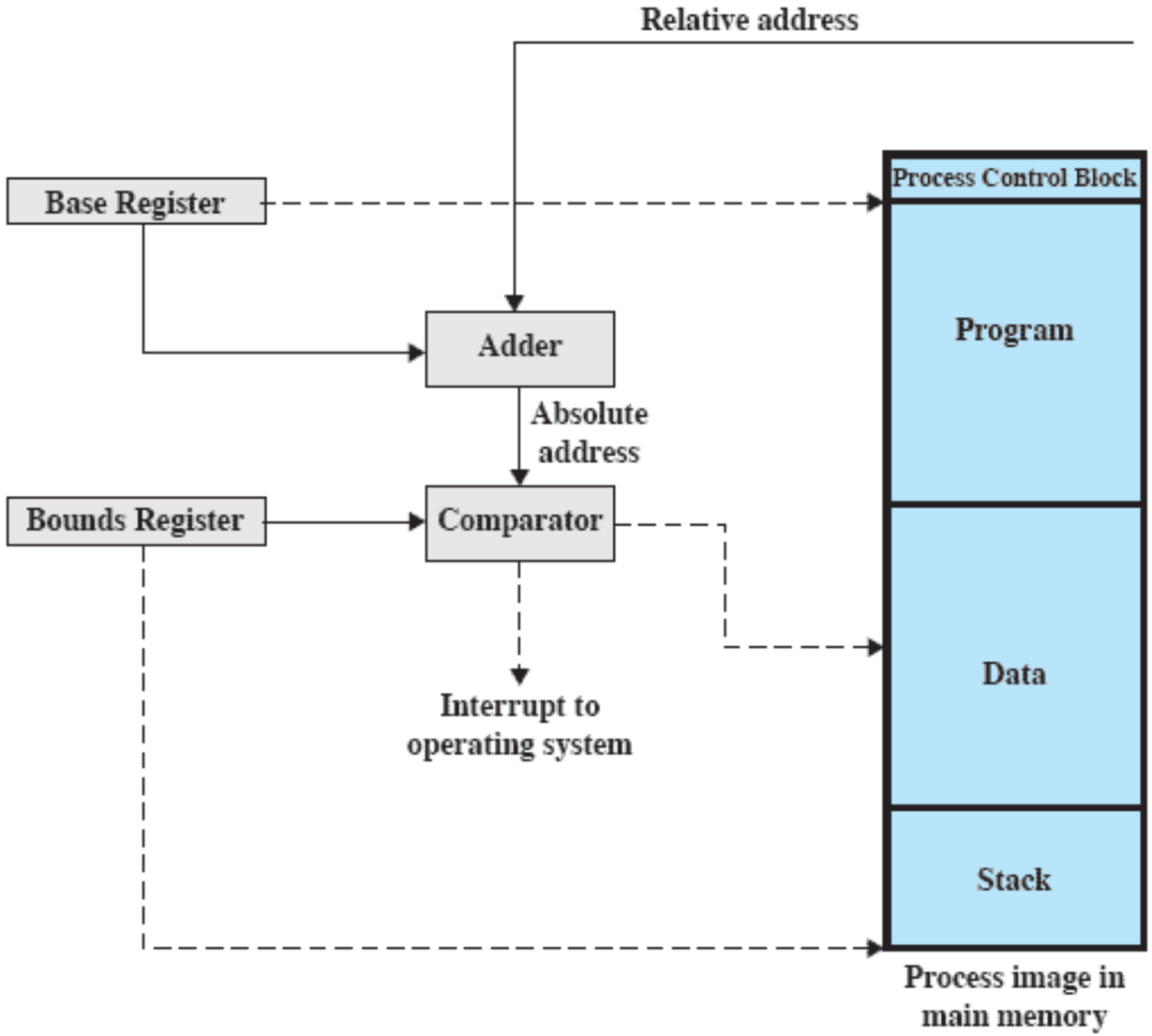


Figure 7.8 Hardware Support for Relocation

Memory Management Techniques

- Partitioning
  - fixed vs. dynamic
  - equal vs. unequal size partitions
- Paging
  - simple
  - virtual memory
- Segmentation
  - simple
  - virtual memory

Paging

- Partition memory into **equal fixed-size chunks** that are relatively **small**
- Process is also divided into small fixed-size chunks of the same size

- The chunks of a process, known as **pages**, could be assigned to available chunks of memory, known as **frames**, or page frames.

Assignment of Process to Free Frames

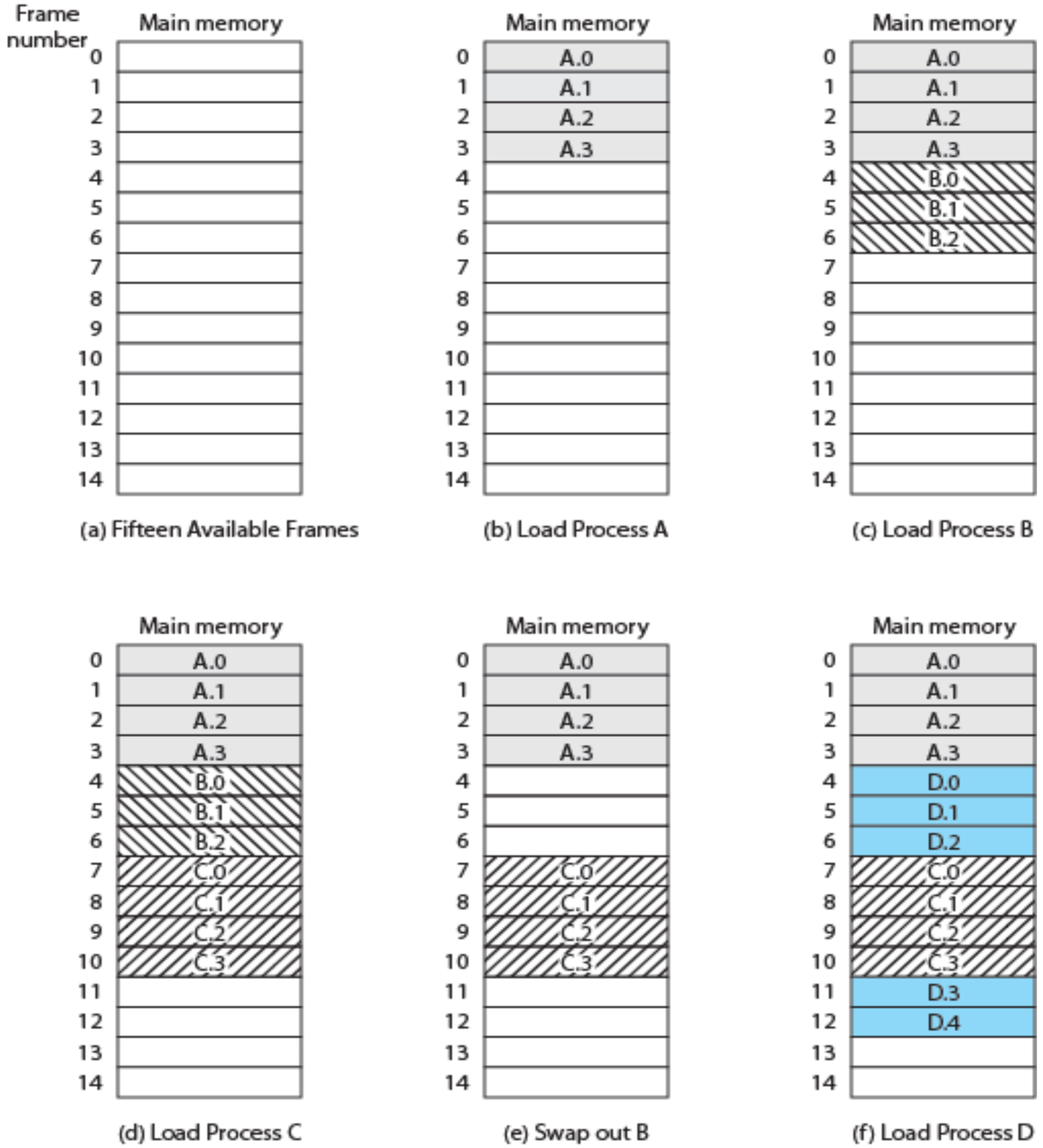


Figure 7.9 Assignment of Process Pages to Free Frames

Page Table

- The OS maintains a **page table** for each process
- Contains the frame location for each page in the process
- Processor must know how to access for the current process
- Used by processor to produce a physical address

Logical address

(page number, offset)

Physical address

(frame number, offset)

Logical Address

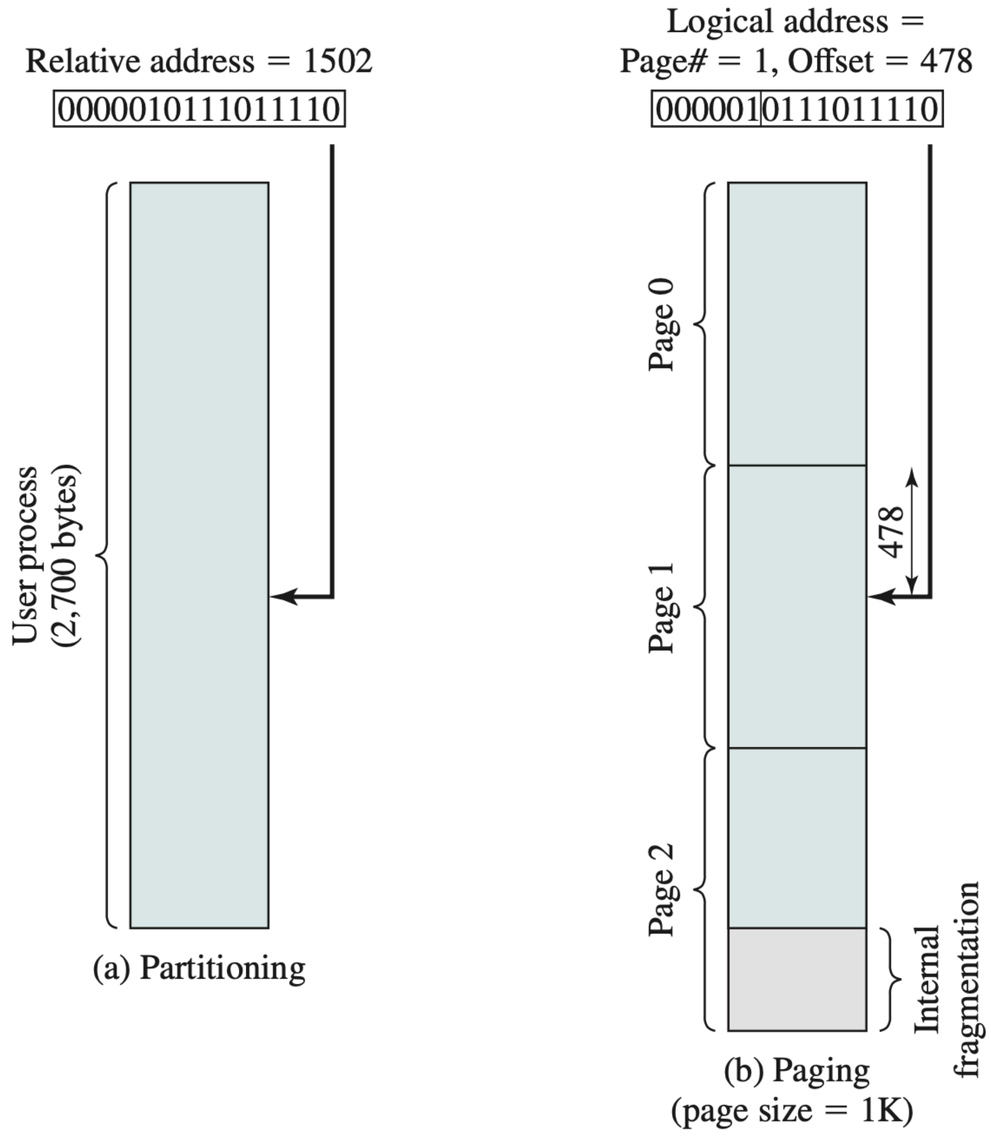
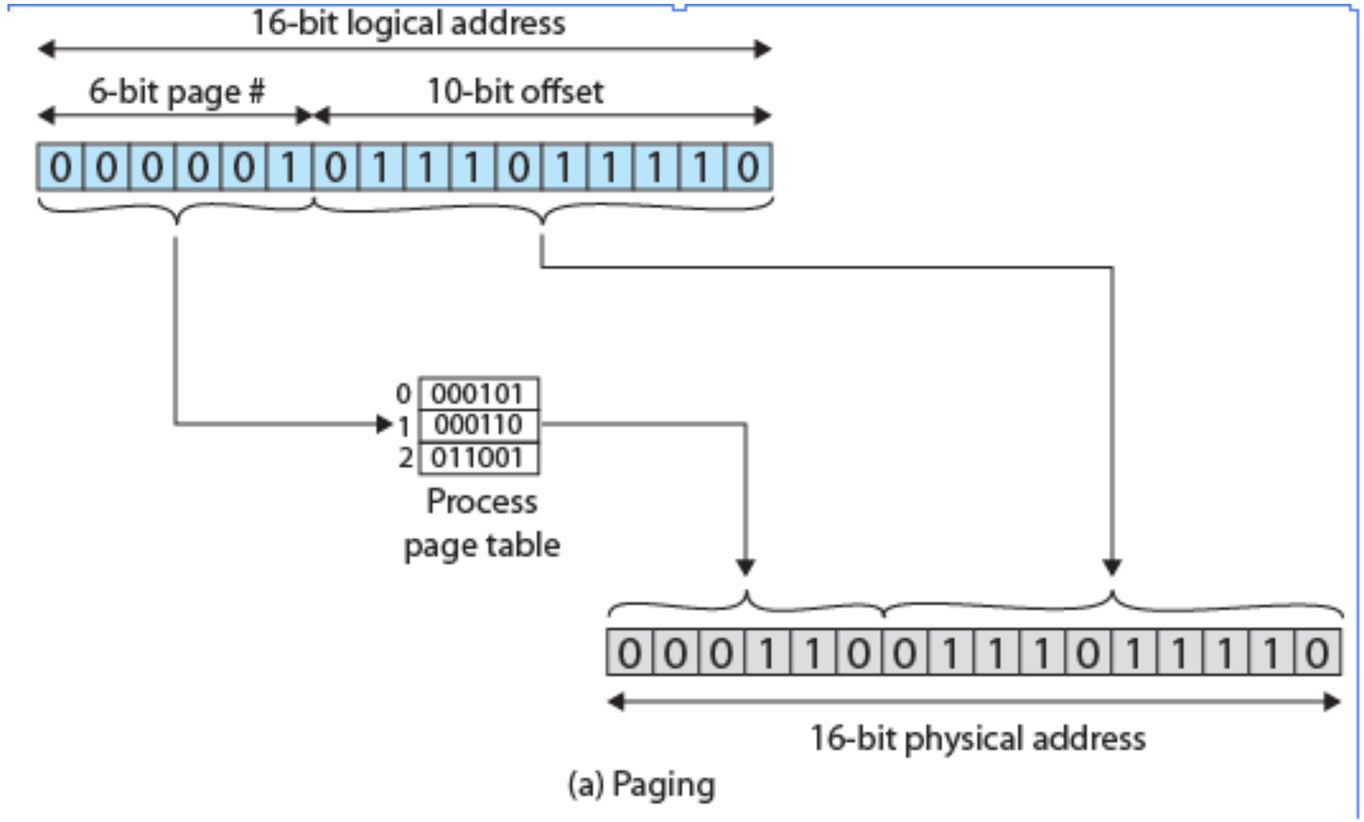


Figure 7.11 **Logical Addresses**

Logical-to-Physical Address Translation - Paging



Memory Management Techniques

- Partitioning
  - fixed vs. dynamic
  - equal vs. unequal size partitions
- Paging
  - simple
  - virtual memory
- Segmentation
  - simple
  - virtual memory

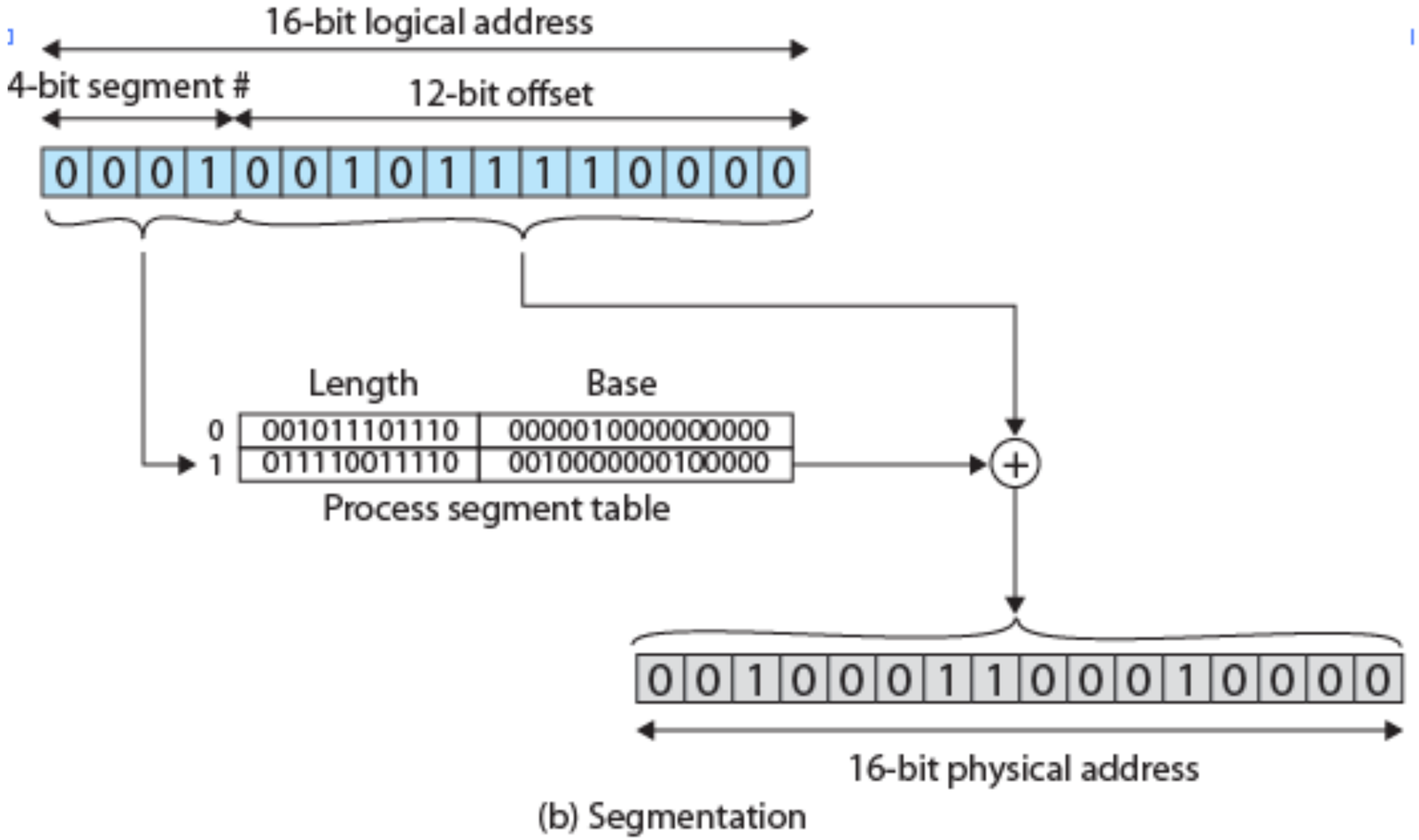
Segmentation

- A program can be subdivided into segments
  - may vary in length, with a maximum length

- **Addressing** consists of two parts:
  - **Segment number**
  - **An offset**

- Similar to dynamic partitioning
- Eliminates internal fragmentation

Logical-to-Physical Address Translation - Paging



(b) Segmentation

Memory Management Techniques

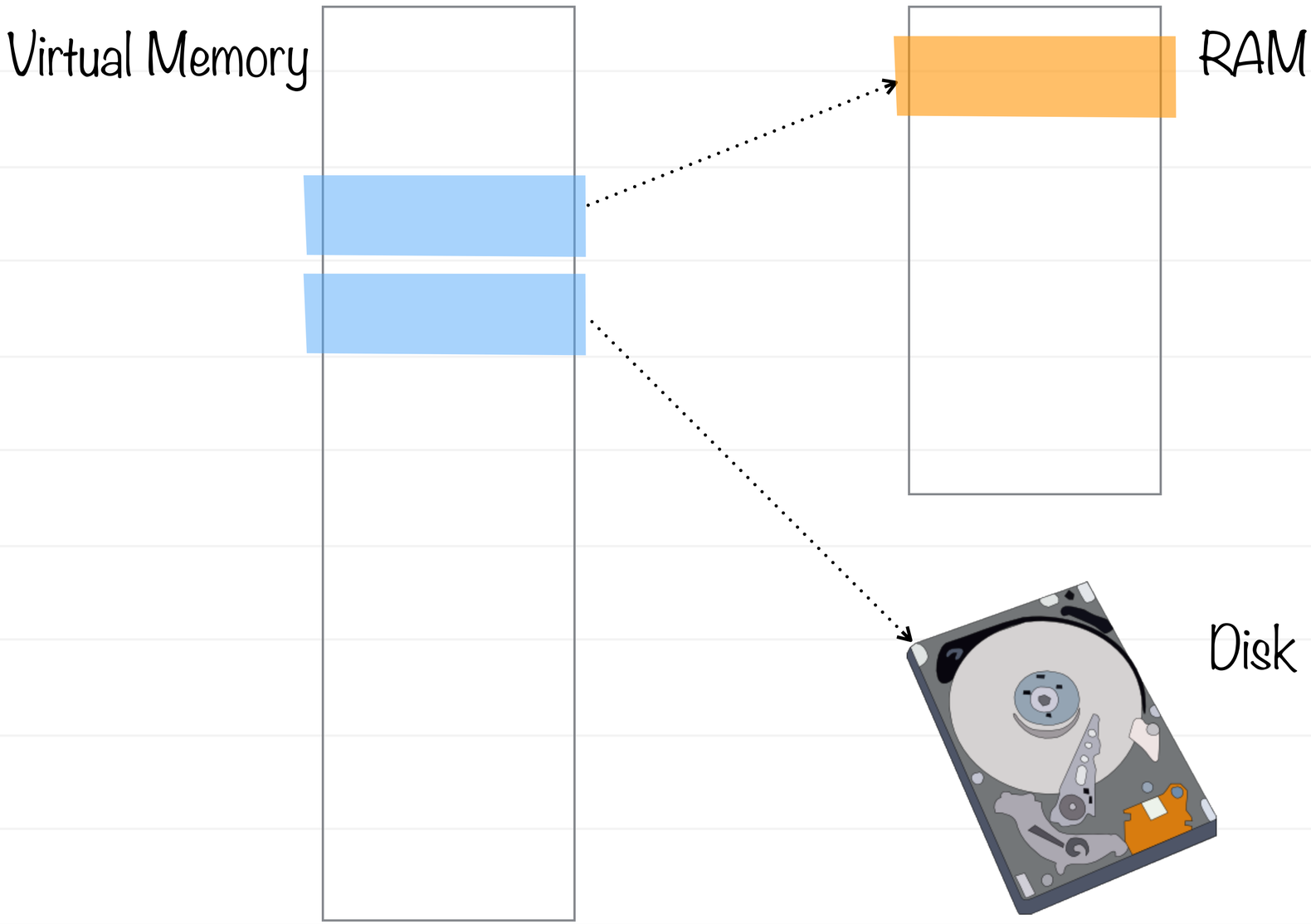
- Partitioning
  - fixed vs. dynamic
  - equal vs. unequal size partitions
- Paging
  - simple
  - virtual memory
- Segmentation
  - simple
  - virtual memory

Summary

- Memory management is one of the **most important and complex tasks** of an operating system
- **Main memory** needs to be treated as a **resource** to be allocated to and shared among a number of active processes
- Desirable to maintain **as many processes in main memory as possible**
- Desirable to **free programmers from size restriction** in program development
- **Basic tools** are **paging** and **segmentation** (possible to combine)
  - **paging** – small fixed-sized pages
  - **segmentation** – pieces of varying size

VIRTUAL MEMORY

HARDWARE


Virtual Memory
RAM
Disk

Hardware and Control Structures

- Two characteristics fundamental to memory management:

1. all memory references are **logical addresses** that are **dynamically** translated into physical addresses at **run time**

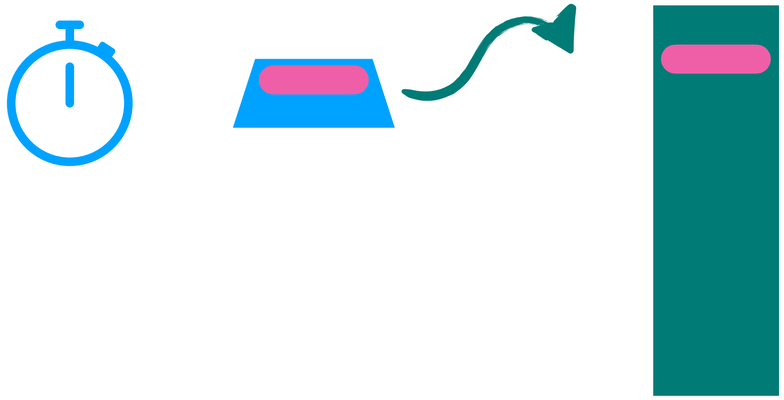
2. a **process** may be broken up into a number of **pieces** that don't need to be contiguously located in main memory during execution

From the combination of these: it is not necessary that all of the pages or segments of a process be in main memory during execution

Terminology

Virtual memory	A storage allocation scheme in which secondary memory can be addressed as though it were part of main memory. The addresses a program may use to reference memory are distinguished from the addresses the memory system uses to identify physical storage sites, and program-generated addresses are translated automatically to the corresponding machine addresses. The size of virtual storage is limited by the addressing scheme of the computer system, and by the amount of secondary memory available and 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	The 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.

Execution of a Process



- Operating system brings into main memory a few pieces of the program
- ***Resident set*** - portion of process that is in main memory

- As long as all memory references are to locations in the resident set the execution proceeds
- **If not** - the processor generates an **interrupt** indicating the needed address is not in the memory
- The OS places the process in a blocking state and takes control

Execution of a Process

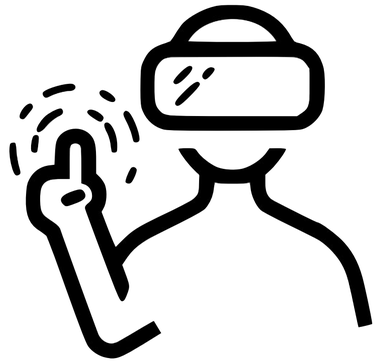
- Piece of process that contains the logical address is brought into main memory
  - operating system issues a disk I/O Read request
  - another process is dispatched to run while the disk I/O takes place
  - an interrupt is issued when disk I/O is complete, which causes the operating system to place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

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**

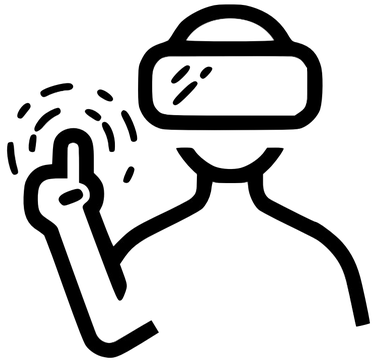
Real and Virtual Memory

- **Real memory** - main memory, the actual RAM



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

Thrashing



- When the system spends most of its time swapping pieces rather than executing instructions.
- To avoid this, the OS 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

Support Needed for Virtual Memory

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

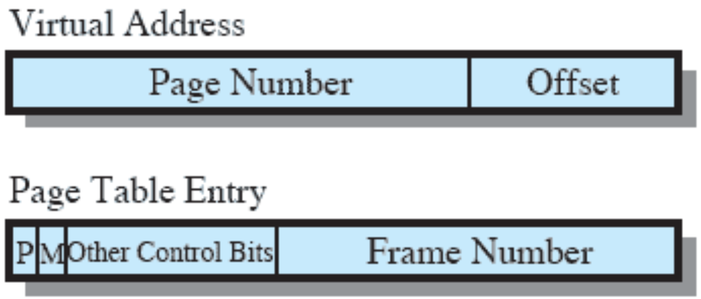
Memory Management Techniques

- Partitioning
  - fixed vs. dynamic
  - equal vs. unequal size partitions
- Paging
  - simple
  - virtual memory
- Segmentation
  - simple
  - virtual memory

Paging

- The term **virtual memory** is usually associated with systems that employ **paging**
- Each process has its own **page table**
- Each Page Table Entry (PTE) contains the frame number of the corresponding page in main memory

Memory Management Format



(a) Paging only

Address Translation

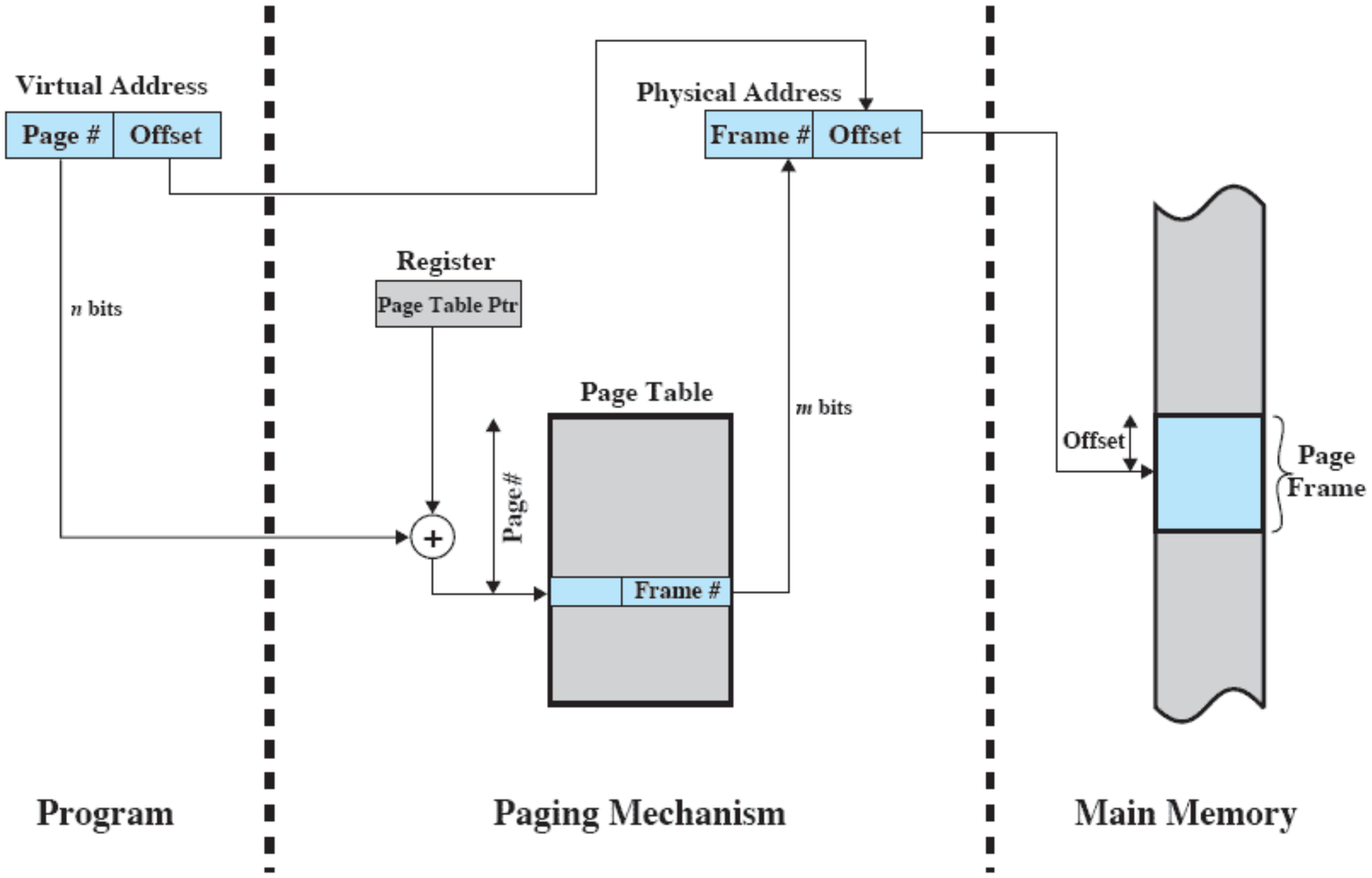


Figure 8.3 Address Translation in a Paging System

Two-Level Hierarchical Page Table

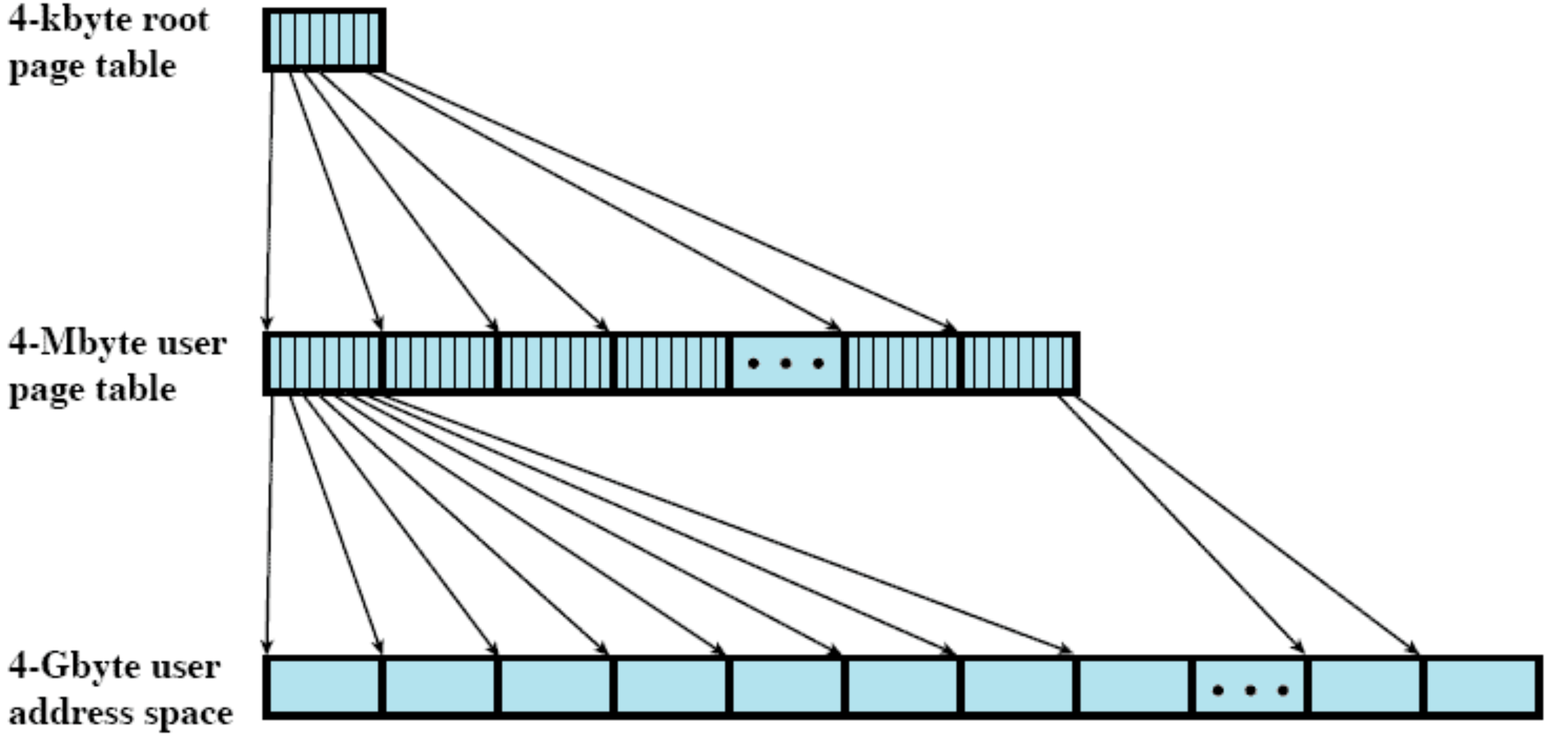


Figure 8.4 A Two-Level Hierarchical Page Table

Address Translation

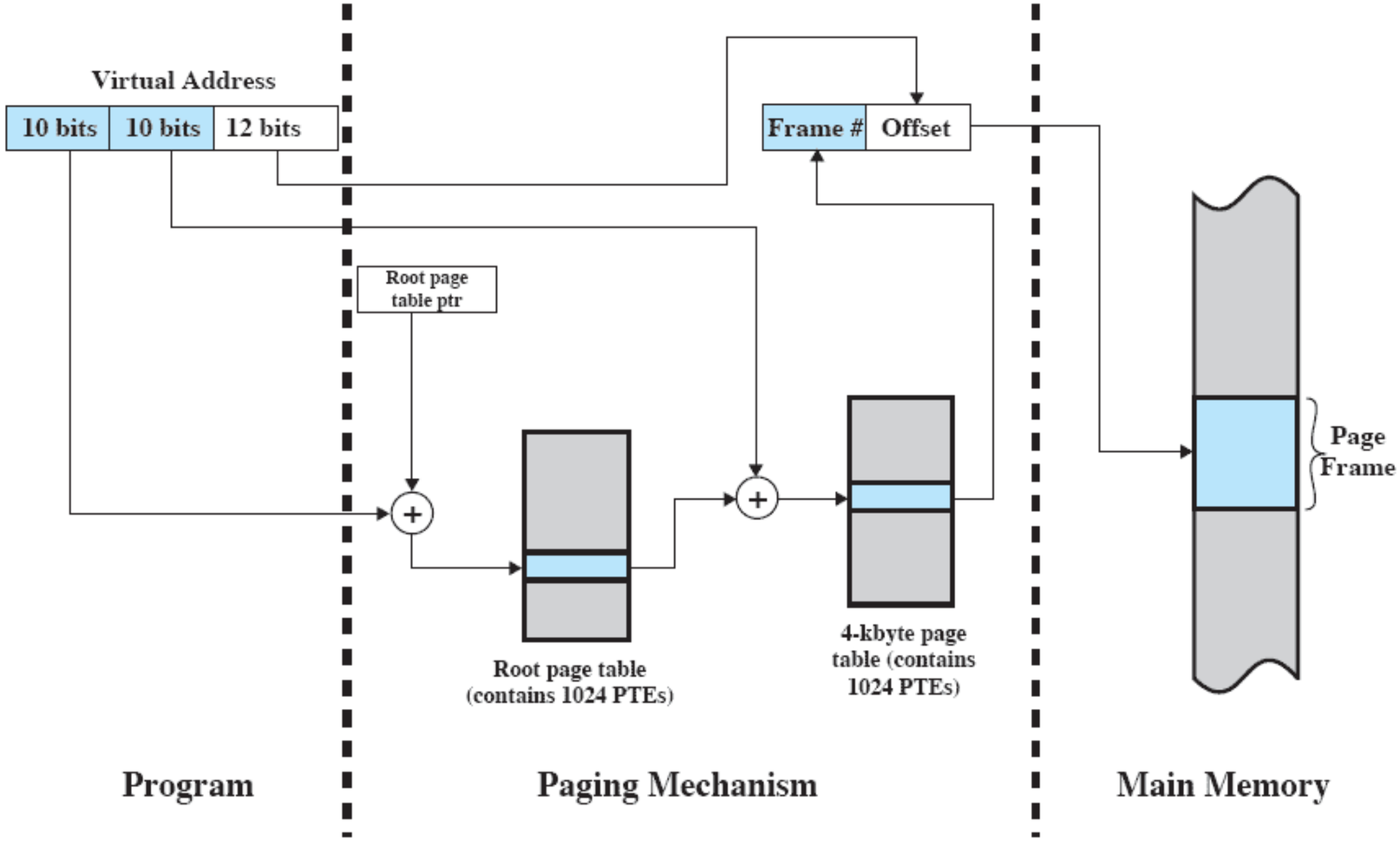


Figure 8.5 Address Translation in a Two-Level Paging System

Inverted Page Table

- Page number portion of a virtual address is mapped into a **hash value**
- Hash value points to **inverted page table**
- Fixed proportion of real memory is required for the tables regardless of the number of processes or virtual pages supported
- Structure is called inverted because it indexes page table entries by frame number rather than by virtual page number

Inverted Page Table

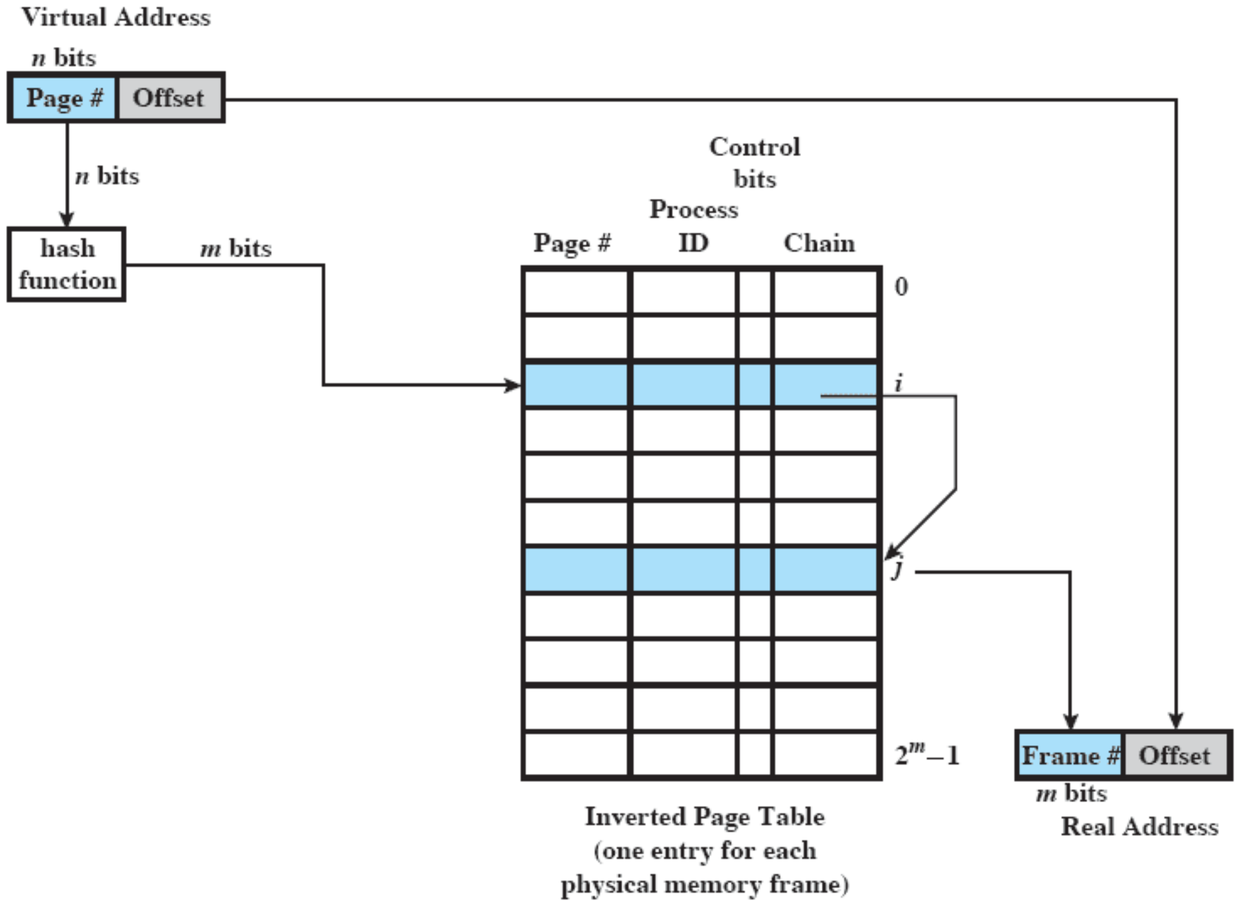


Figure 8.6 Inverted Page Table Structure

Inverted Page Table

- Each entry in the page table includes:

Page number	Process identifier	Control bits	Chain pointer
	The process that owns the page	includes flags and protection and locking information	the index value of the next entry in the chain

Translation Lookaside Buffer (TLB)

Griffith UNIVERSITY

- 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 TLB

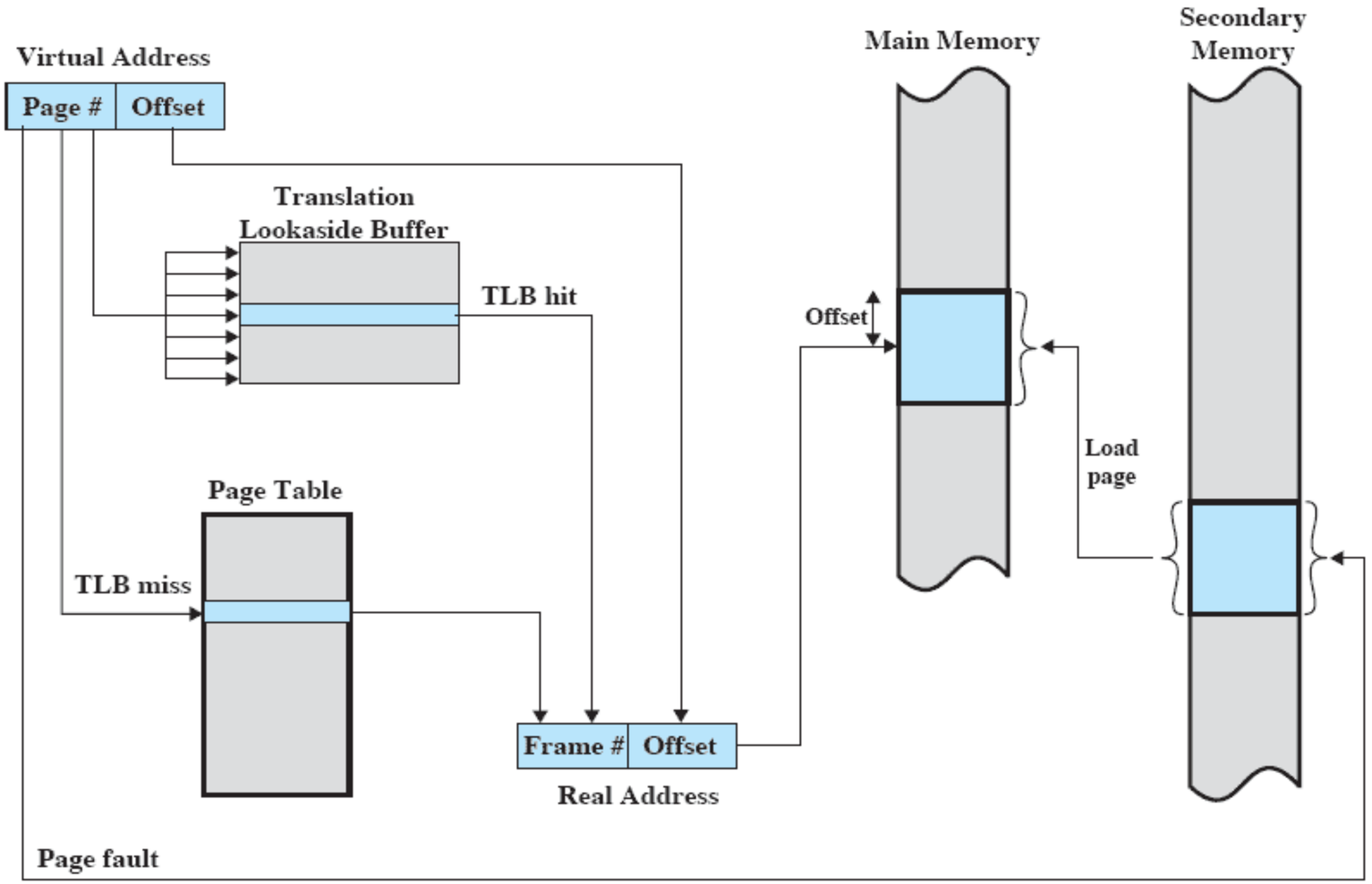


Figure 8.7 Use of a Translation Lookaside Buffer

TLB Operation

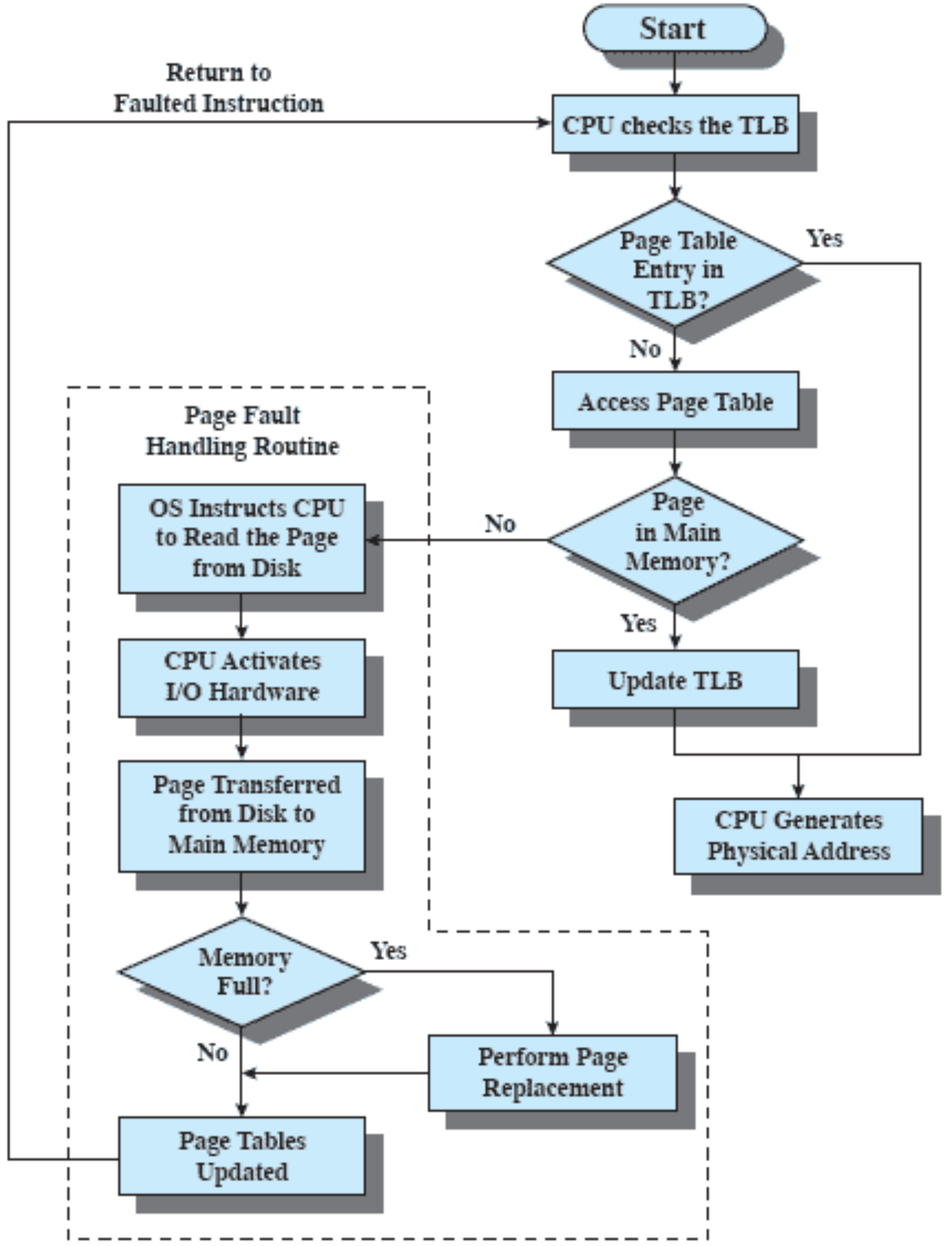


Figure 8.8 Operation of Paging and Translation Lookaside Buffer (TLB) [FURH87]

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 vs. Associative Lookup

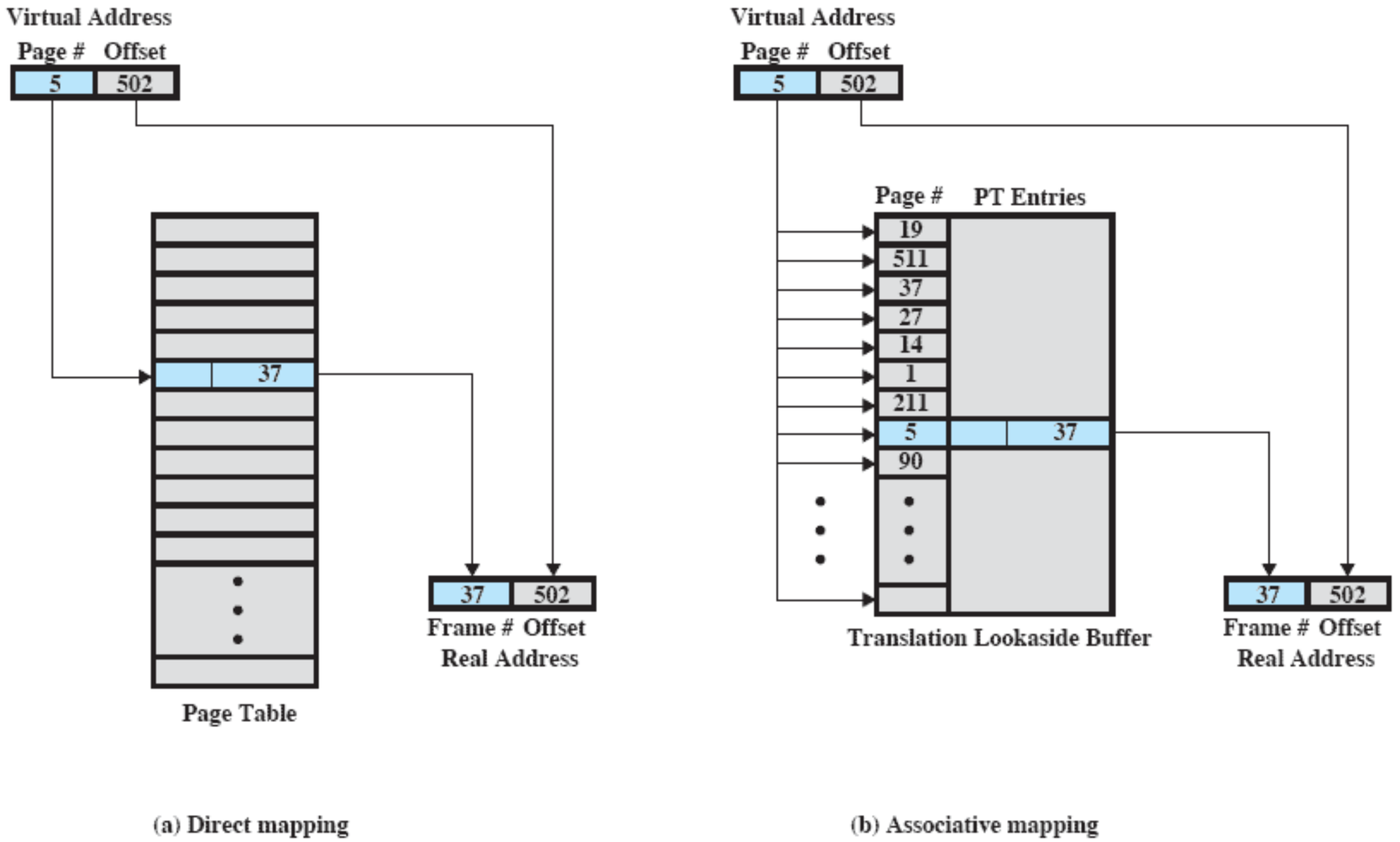
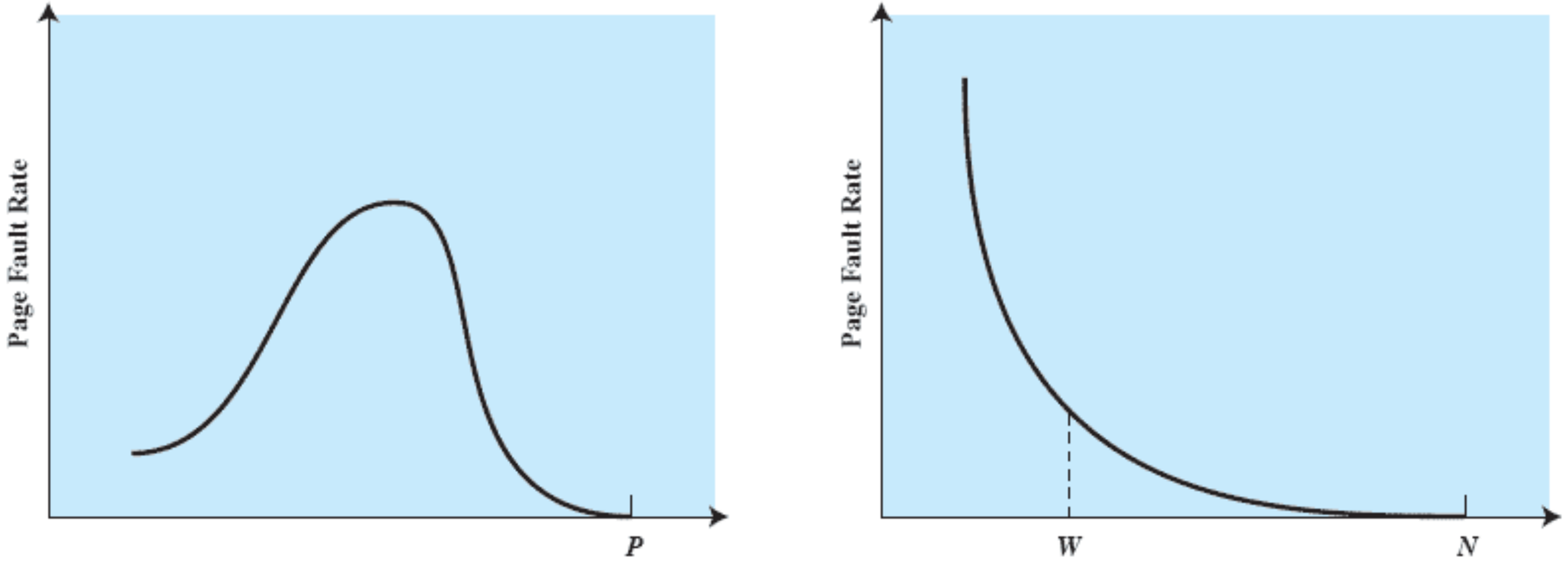


Figure 8.9 Direct Versus Associative Lookup for Page Table Entries

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 => **double page fault**
- **the physical characteristics** of most secondary-memory devices favour a **larger page size for more efficient block transfer** of data

Paging Behaviour of a Program



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

Table 8.3 Example of Page Sizes

Computer	Page Size
Atlas	512 48-bit words
Honeywell-Multics	1,024 36-bit words
IBM 370/XA and 370/ESA	4 kB
VAX family	512 bytes
IBM AS/400	512 bytes
DEC Alpha	8 kB
MIPS	4 kB to 16 MB
UltraSPARC	8 kB to 4 MB
Pentium	4 kB or 4 MB
Intel Itanium	4 kB to 256 MB
Intel core i7	4 kB to 1 GB

Memory Management Techniques

- Partitioning
  - fixed vs. dynamic
  - equal vs. unequal size partitions
- Paging
  - simple
  - virtual memory
- Segmentation
  - simple
  - virtual memory

Segmentation

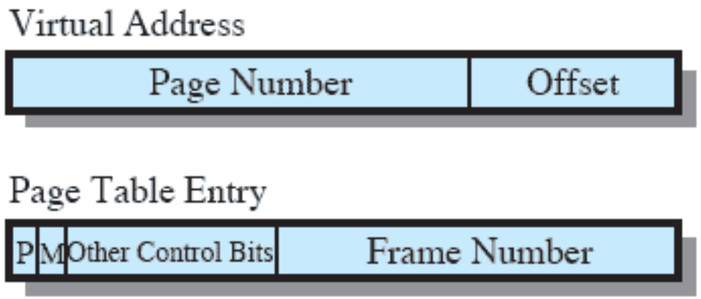
- Segmentation allows the programmer to view memory as consisting of multiple address spaces or segments

- **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

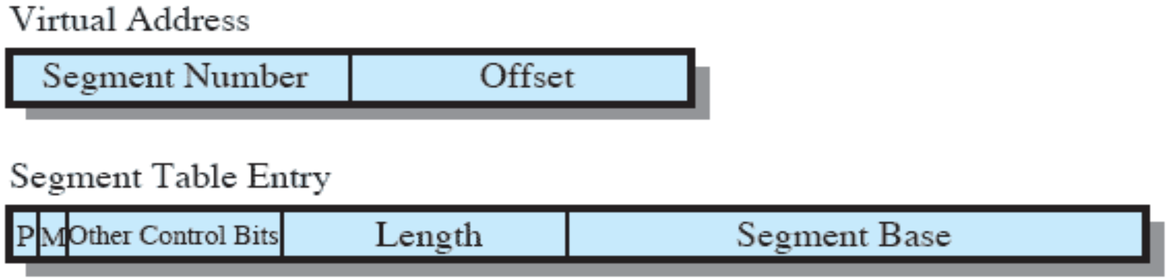
Segmentation Organisation

- Each segment table entry contains the **starting address** of the corresponding segment in main memory and the **length** of the segment
- A **bit (P)** is needed to determine if the segment is already in main memory
- Another **bit (M)** is needed to determine if the segment has been modified since it was loaded in main memory

Memory Management Format



(a) Paging only



(b) Segmentation only

Address Translation

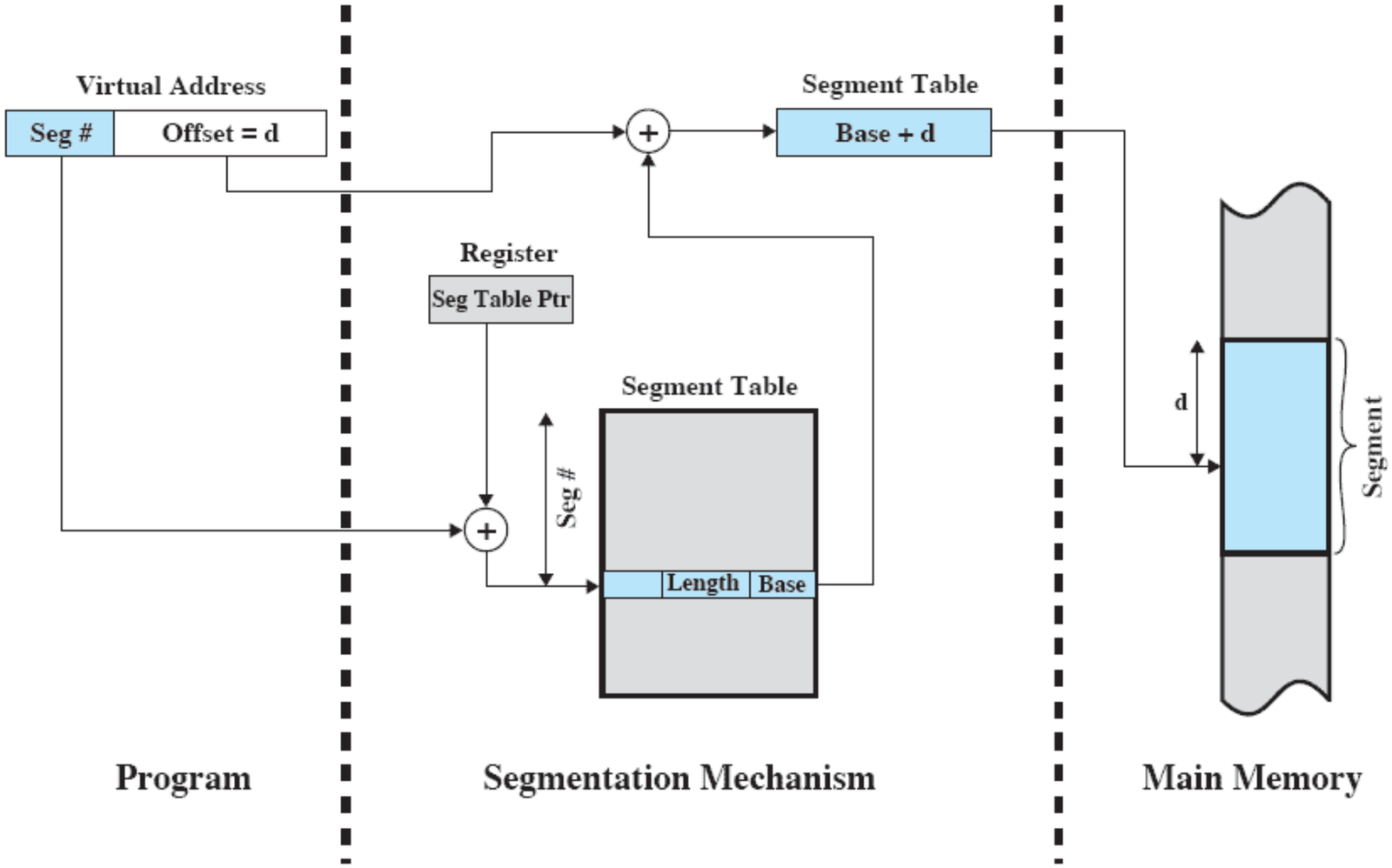
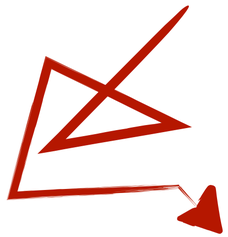


Figure 8.12 Address Translation in a Segmentation System

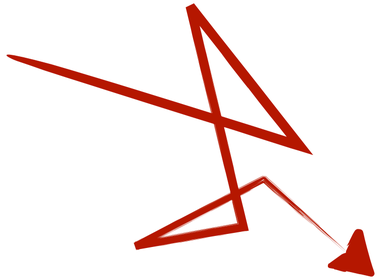
Combined Paging and Segmentation

- 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

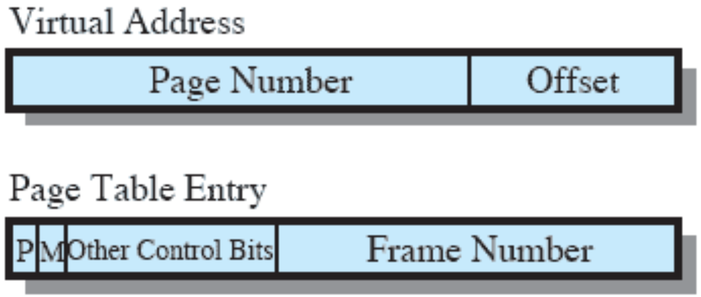


Segmentation is **visible** to the programmer

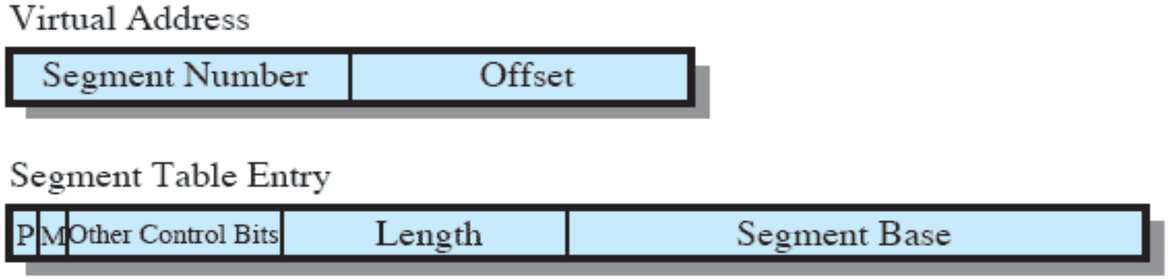


Paging is **transparent** to the programmer

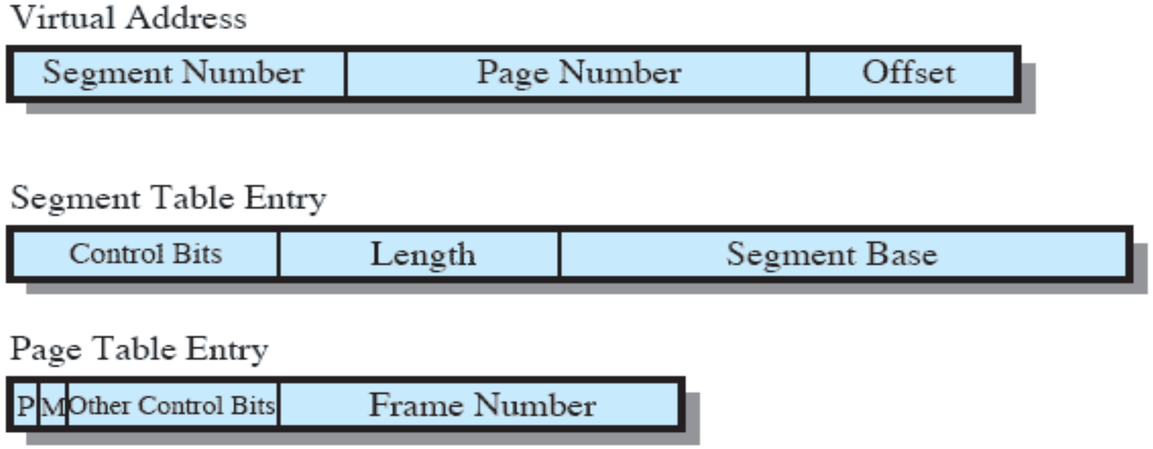
Memory Management Format



(a) Paging only



(b) Segmentation only



(c) Combined segmentation and paging

P= present bit
M = Modified bit

Address Translation

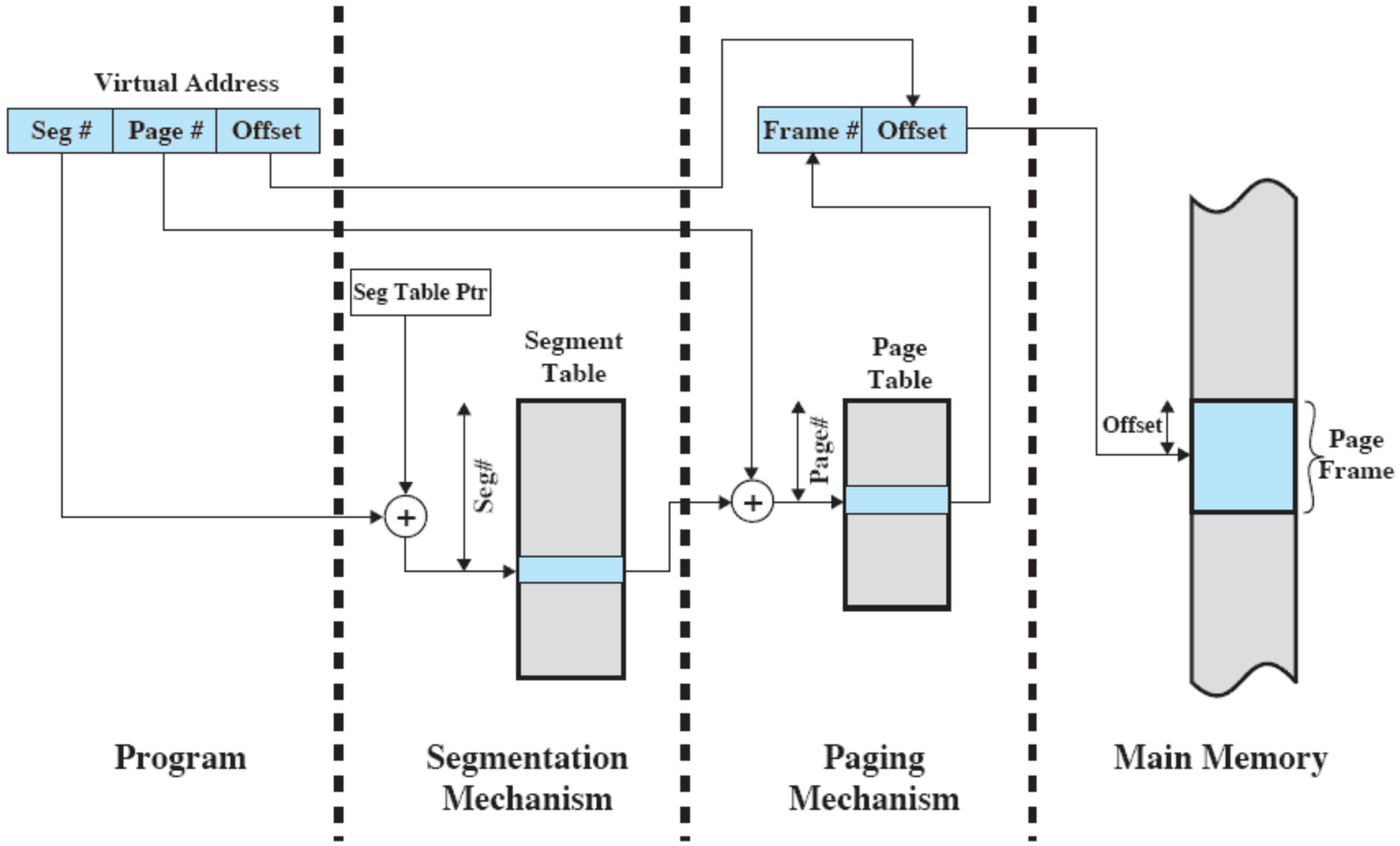


Figure 8.13 Address Translation in a Segmentation/Paging System

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

Protection Relationship

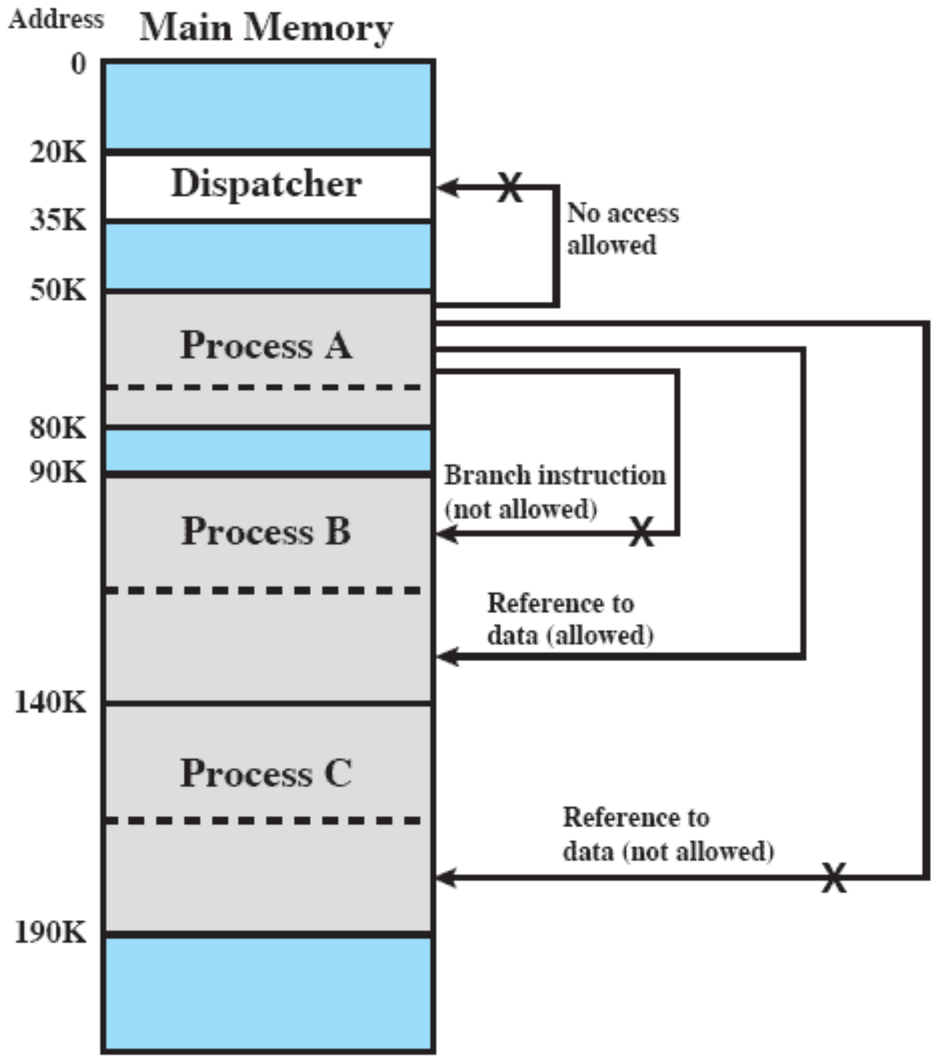


Figure 8.14 Protection Relationships Between Segments

Griffith UNIVERSITY

VIRTUAL MEMORY

OPERATING SYSTEM SOFTWARE

Operating System Software

- The design of the memory management portion of an operating system depends on three fundamental areas of choice:
  - whether or not to use **virtual memory** techniques
  - the use of **paging** or **segmentation** or **both**
  - the **algorithms** employed for various aspects of memory management

Resident Set Management

- The OS must decide how many pages to bring into main memory
  - the smaller the amount of memory allocated to each process, the more processes can reside in memory
  - small number of pages loaded increases page faults
  - beyond a certain size, further allocations of pages will not effect the page fault rate

Resident Set Size

- **Fixed-allocation**
  - gives a process a **fixed number of frames** in main memory within which to execute
  - when a page fault occurs, one of the pages of that process must be replaced
- **Variable-allocation**
  - allows the number of page frames allocated to a process to be **varied over the lifetime** of the process

Replacement Scope

- The scope of a replacement strategy can be categorised as **global** or **local**
  - both types are activated by a page fault when there are no free page frames

- **Local**
  - chooses only among the resident pages of the process that generated the page fault

- **Global**
  - considers all unlocked pages in main memory

Page Fault Frequency (PFF)

- Requires a **use bit** to be associated with each page in memory
- Bit is set to 1 when that page is accessed
- When a page fault occurs, the OS notes the virtual time since the last page fault for that process
- Does not perform well during the transient periods when there is a shift to a new locality

Cleaning Policy

- Concerned with determining when a modified page should be written out to secondary memory

- **Demand clean**
  - a page is written out to secondary memory only when it has been selected for replacement

- **Precleaning**
  - allows the writing of pages in batches

Load control

- Determines the number of processes that will be resident in main memory
  - **multiprogramming** level
- Critical in effective memory management
- Too few processes, many occasions when all processes will be blocked and much time will be spent in swapping
- Too many processes will lead to thrashing

Multiprogramming

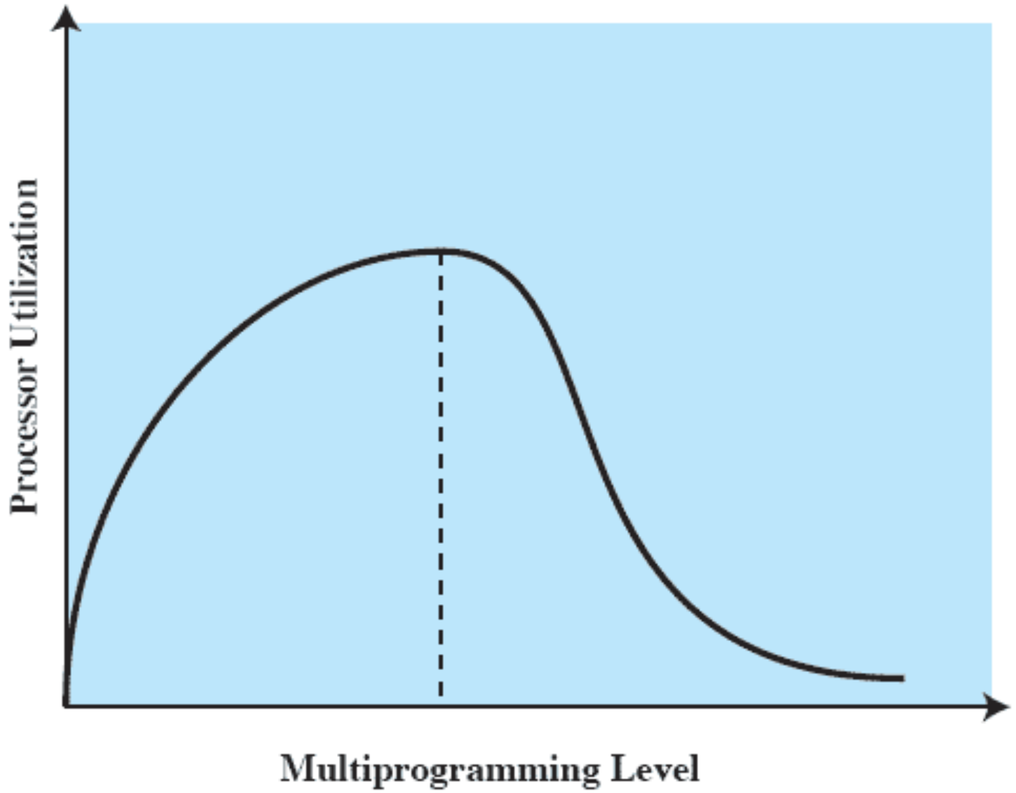


Figure 8.21 Multiprogramming Effects

Process Suspension

- If the degree of multiprogramming is to be reduced, one or more of the currently resident processes must be swapped out

- Six possibilities exist:
    - Lowest-priority process
    - Faulting process
    - Last process activated
    - Process with the smallest resident set
    - Largest process
    - Process with the largest remaining execution window

Page Replacement

- The page frame data table is used for page replacement
- Pointers are used to create lists within the table
  - all available frames are linked together in a list of free frames available for bringing in pages
  - when the number of available frames drops below a certain threshold, the kernel will steal a number of frames to compensate

Kernel Memory Allocator

- The kernel generates and destroys small tables and buffers frequently during the course of execution, each of which requires dynamic memory allocation.
- Most of these blocks are significantly smaller than typical pages (therefore paging would be inefficient)
- Allocations and free operations must be made as fast as possible

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