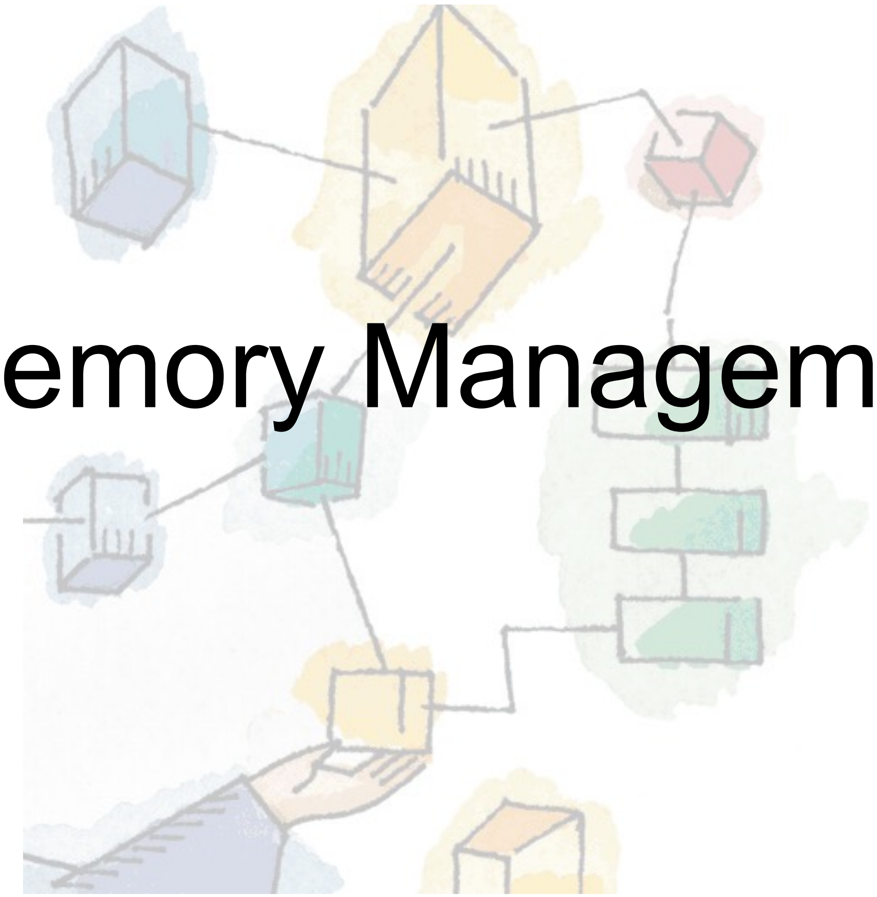
Operating Systems: Internals and Design Principles, 6/E
William Stallings



Memory Management

Patricia Roy
Manatee Community College, Venice, FL

Memory Management

Memory needs to be allocated to ensure a reasonable supply of ready processes to consume available processor time

Memory Management Terms

Table 7.1 Memory Management Terms

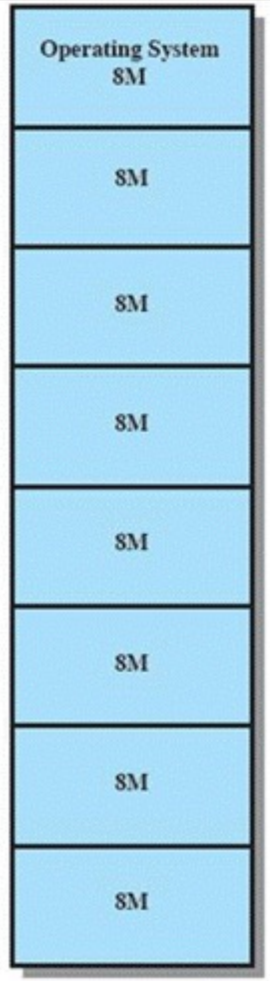
Term	Description
Frame	***Fixed***-length block of main memory.
Page	***Fixed***-length block of data in secondary memory (e.g. on disk).
Segment	***Variable-length*** block of data that resides in secondary memory.

Types of Partitioning

- Fixed Partitioning
- Dynamic Partitioning
- Simple Paging
- Simple Segmentation
- Virtual Memory Paging
- Virtual Memory Segmentation

Fixed Partitioning

- Equal-size partitions (see fig 7.3a)
  - 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 a process out of a partition
  - If none are in a ready or running state



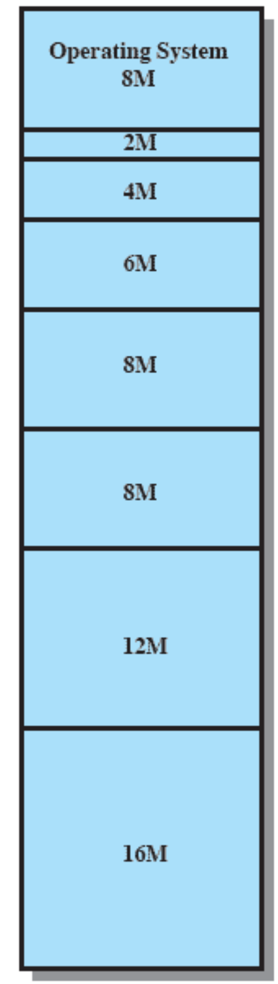
(a) Equal-size partitions

Fixed Partitioning Problems

- A program may not fit in a partition.
  - The programmer must design the program with overlays
- Main memory use is inefficient.
  - Any program, no matter how small, occupies an entire partition.
  - This is results in ***internal fragmentation.***

Solution – Unequal Size Partitions

- Lessens both problems
  - but doesn't solve completely
- In Fig 7.3b,
  - Programs up to 16M can be accommodated without overlay
  - Smaller programs can be placed in smaller partitions, reducing internal fragmentation



(b) Unequal-size partitions

Placement Algorithm

- Equal-size
  - Placement is trivial (no options)
- Unequal-size
  - Can assign each process to the smallest partition within which it will fit
  - Queue for each partition
  - Processes are assigned in such a way as to minimize wasted memory within a partition

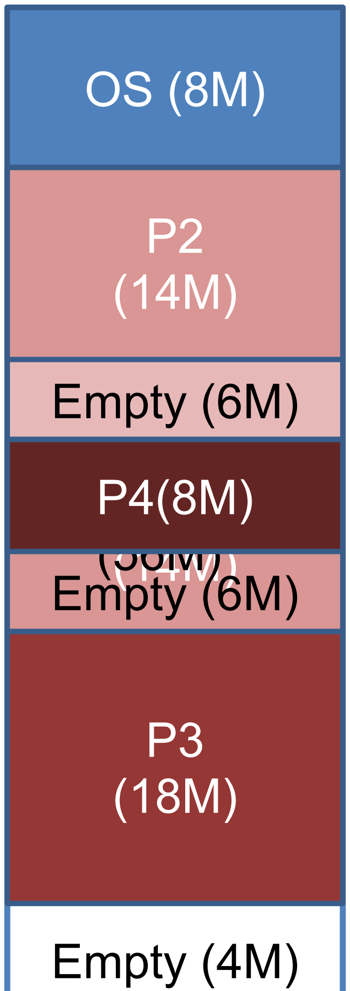
Remaining Problems with Fixed Partitions

- The number of active processes is limited by the system
  - I.E limited by the pre-determined number of partitions
- A large number of very small process will not use the space efficiently
  - In either fixed or variable length partition methods

Dynamic Partitioning

- Partitions are of variable length and number
- Process is allocated exactly as much memory as required

Dynamic Partitioning Example



Refer to Figure 7.4

- ***External Fragmentation***
- Memory external to all processes is fragmented
- Can resolve using ***compaction***
  - OS moves processes so that they are contiguous
  - Time consuming and wastes CPU time

Dynamic Partitioning

- Operating system must decide which free block to allocate to a process
- Best-fit algorithm
  - Chooses the block that is closest in size to the request
  - Worst performer overall
  - Since smallest block is found for process, the smallest amount of fragmentation is left
  - Memory compaction must be done more often

Dynamic Partitioning

- First-fit algorithm
  - Scans memory form the beginning and chooses the first available block that is large enough
  - Fastest
  - May have many process loaded in the front end of memory that must be searched over when trying to find a free block

Dynamic Partitioning

- Next-fit
  - Scans memory from the location of the last placement
  - More often allocate a block of memory at the end of memory where the largest block is found
  - The largest block of memory is broken up into smaller blocks
  - Compaction is required to obtain a large block at the end of memory

Allocation

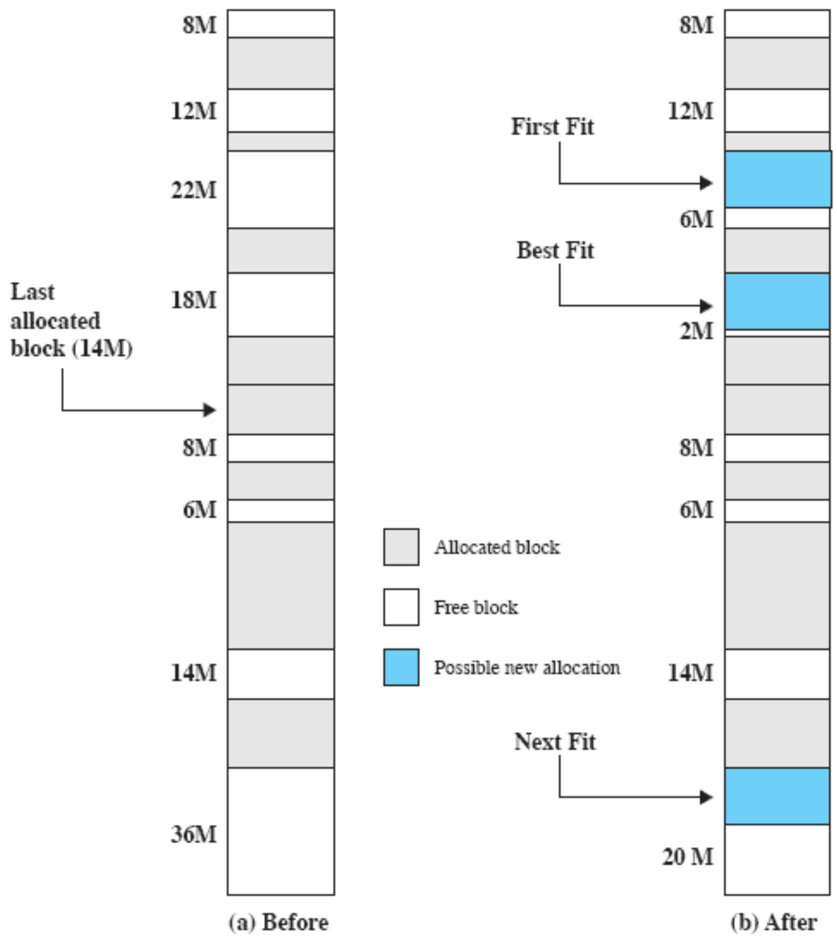


Figure 7.5 Example Memory Configuration before and after Allocation of 16-Mbyte Block

Relocation

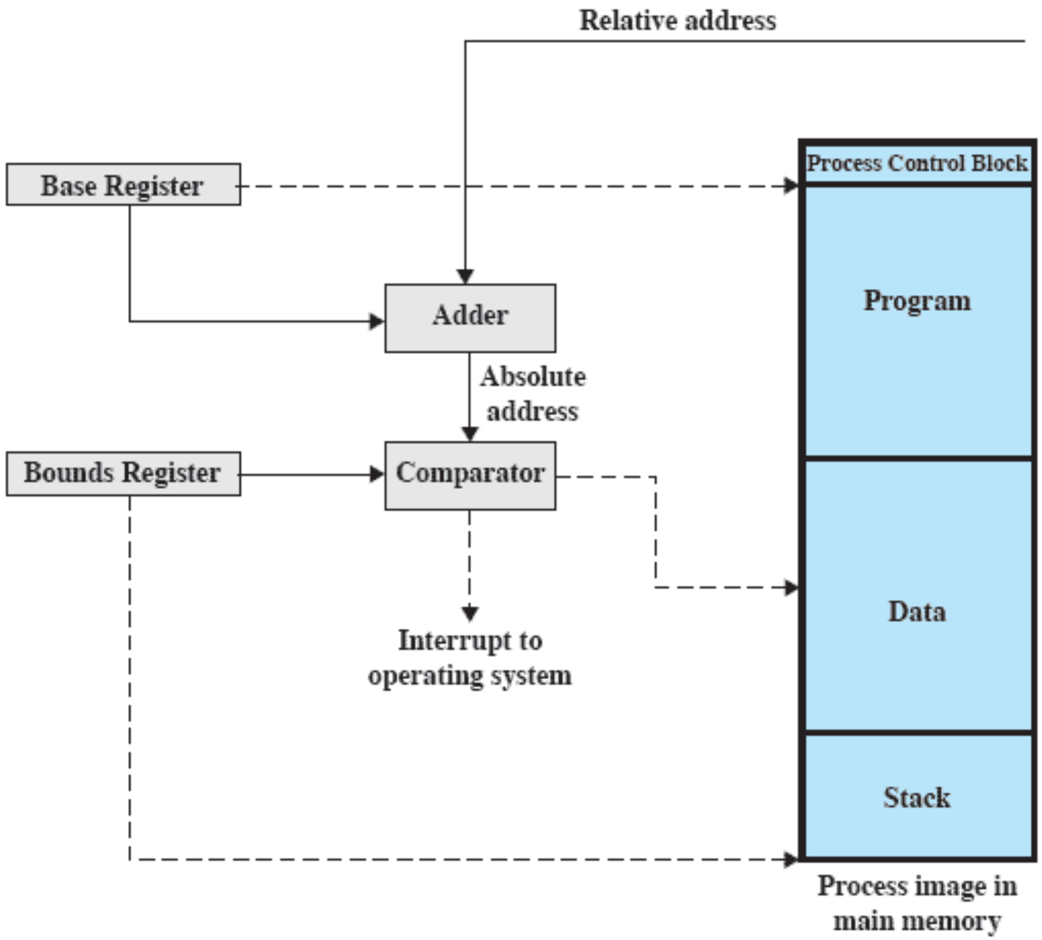


Figure 7.8 Hardware Support for Relocation

Registers Used during Execution

- Base register
  - Starting address for the process
- Bounds register
  - Ending location of the process
- These values are set when the process is loaded or when the process is swapped in

Registers Used during Execution

- The value of the base register is added to a relative address to produce an absolute address
- The resulting address is compared with the value in the bounds register
- If the address is not within bounds, an interrupt is generated to the operating system

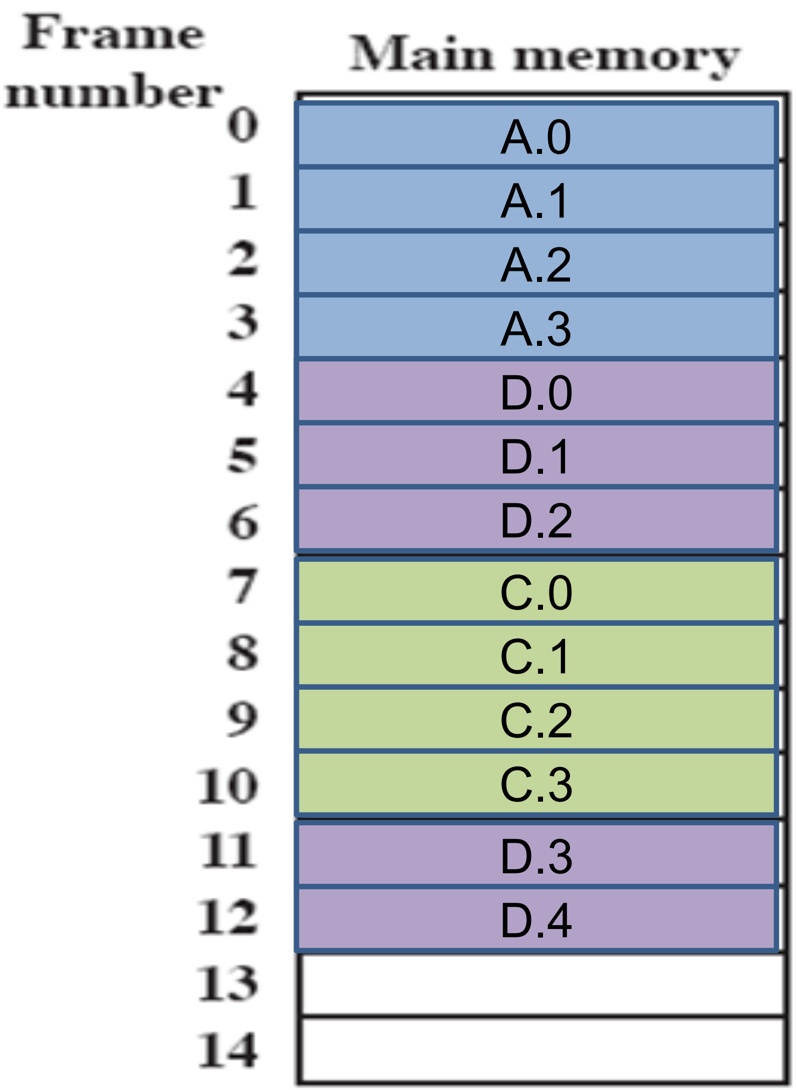
Paging

- Partition memory into small equal fixed-size chunks and divide each process into the same size chunks
- The chunks of a process are called ***pages***
- The chunks of memory are called ***frames***

Paging

- Operating system maintains a page table for each process
  - Contains the frame location for each page in the process
  - Memory address consist of a page number and offset within the page

Processes and Frames



Page Table

Process A page table

Page	Frame
0	0
1	1
2	2
3	3

Process B page table

Page	Frame
0	—
1	—
2	—

Process C page table

Page	Frame
0	7
1	8
2	9
3	10

Process D page table

Page	Frame
0	4
1	5
2	6
3	11
4	12

Free frame list

13
14

Figure 7.10 Data Structures for the Example of Figure 7.9 at Time Epoch (f)

Segmentation

- A program can be subdivided into segments
  - Segments may vary in length
  - There is a maximum segment length
- Addressing consist of two parts
  - a segment number and
  - an offset
- Segmentation is similar to dynamic partitioning

Logical Addresses

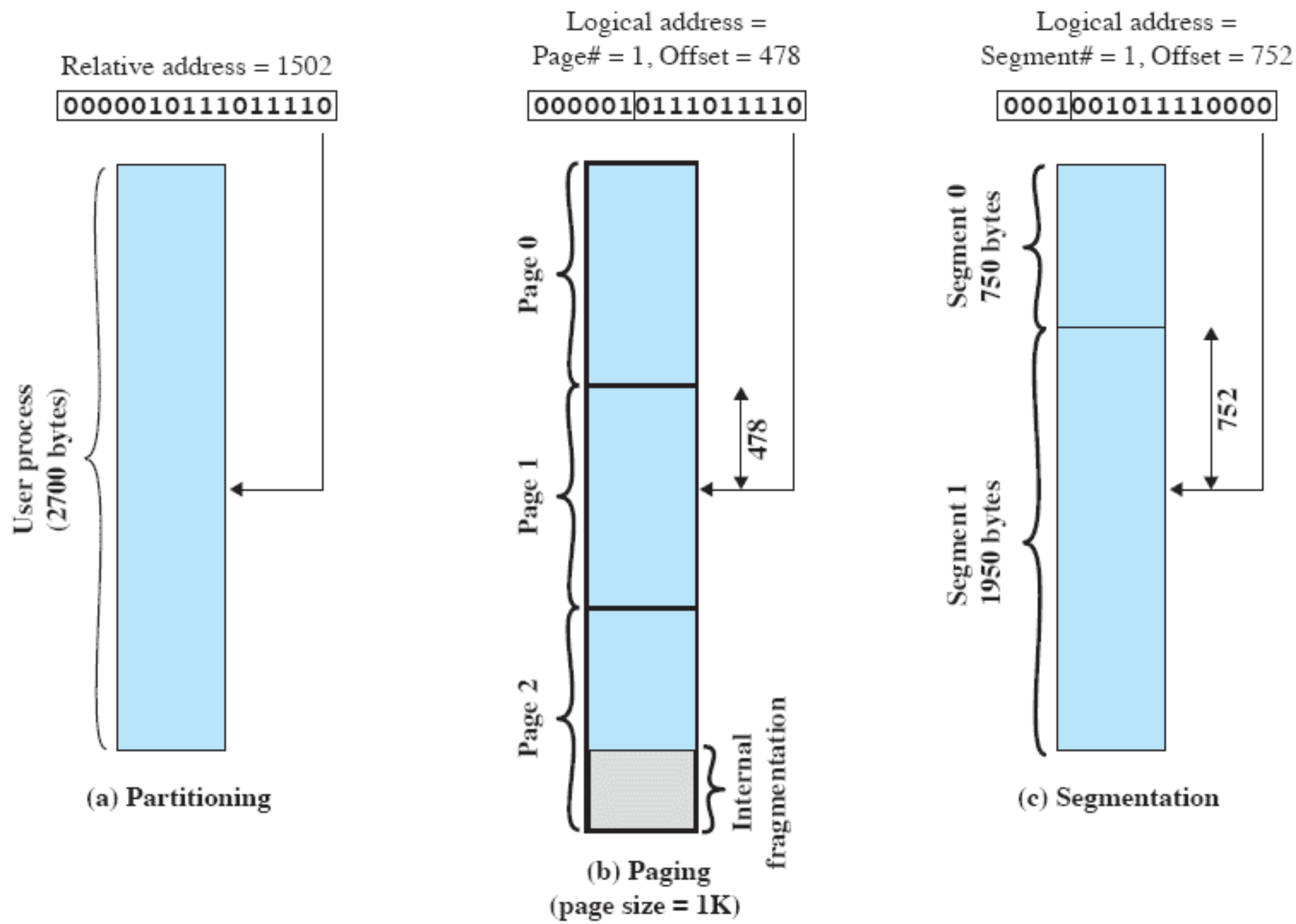
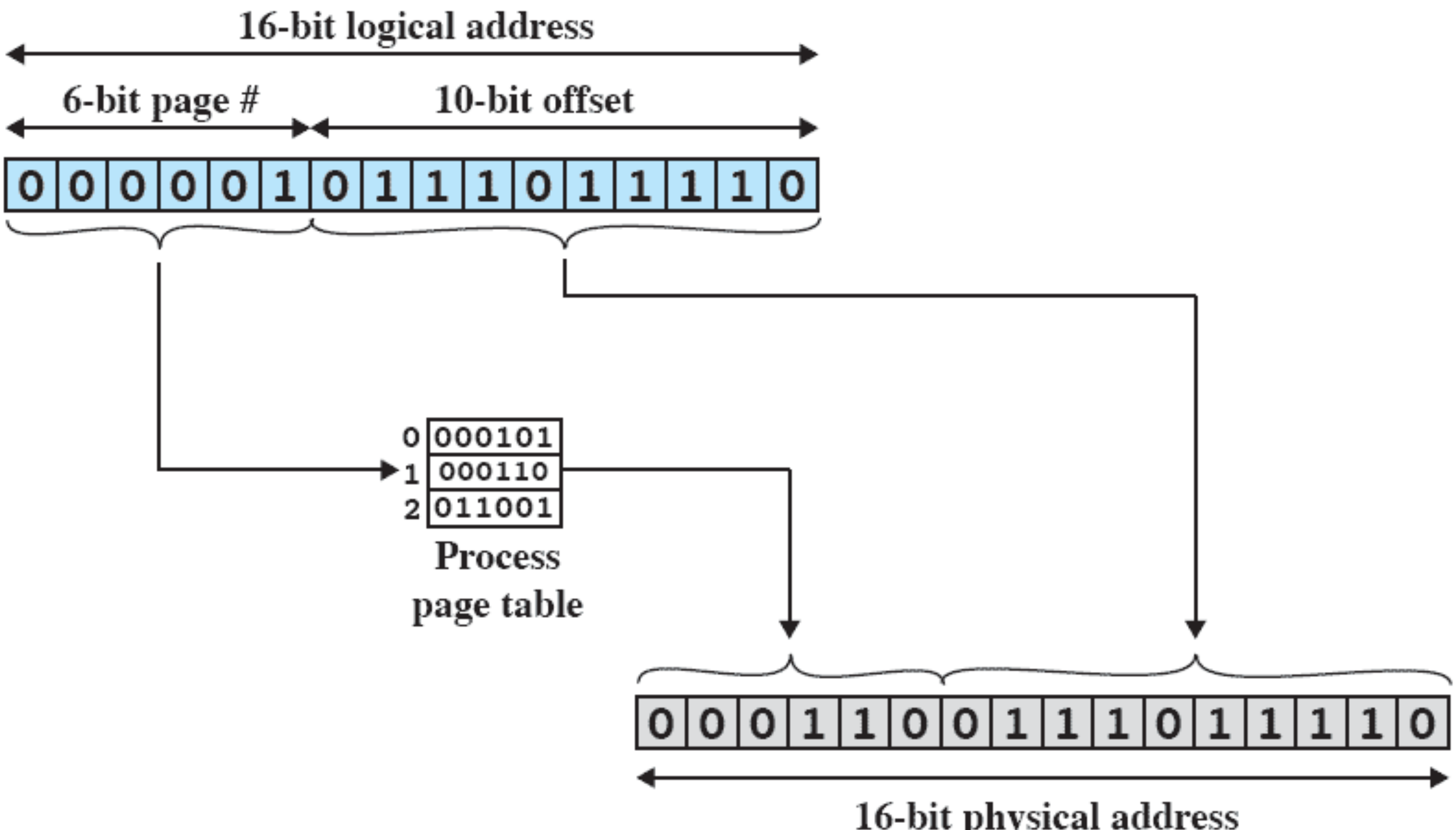


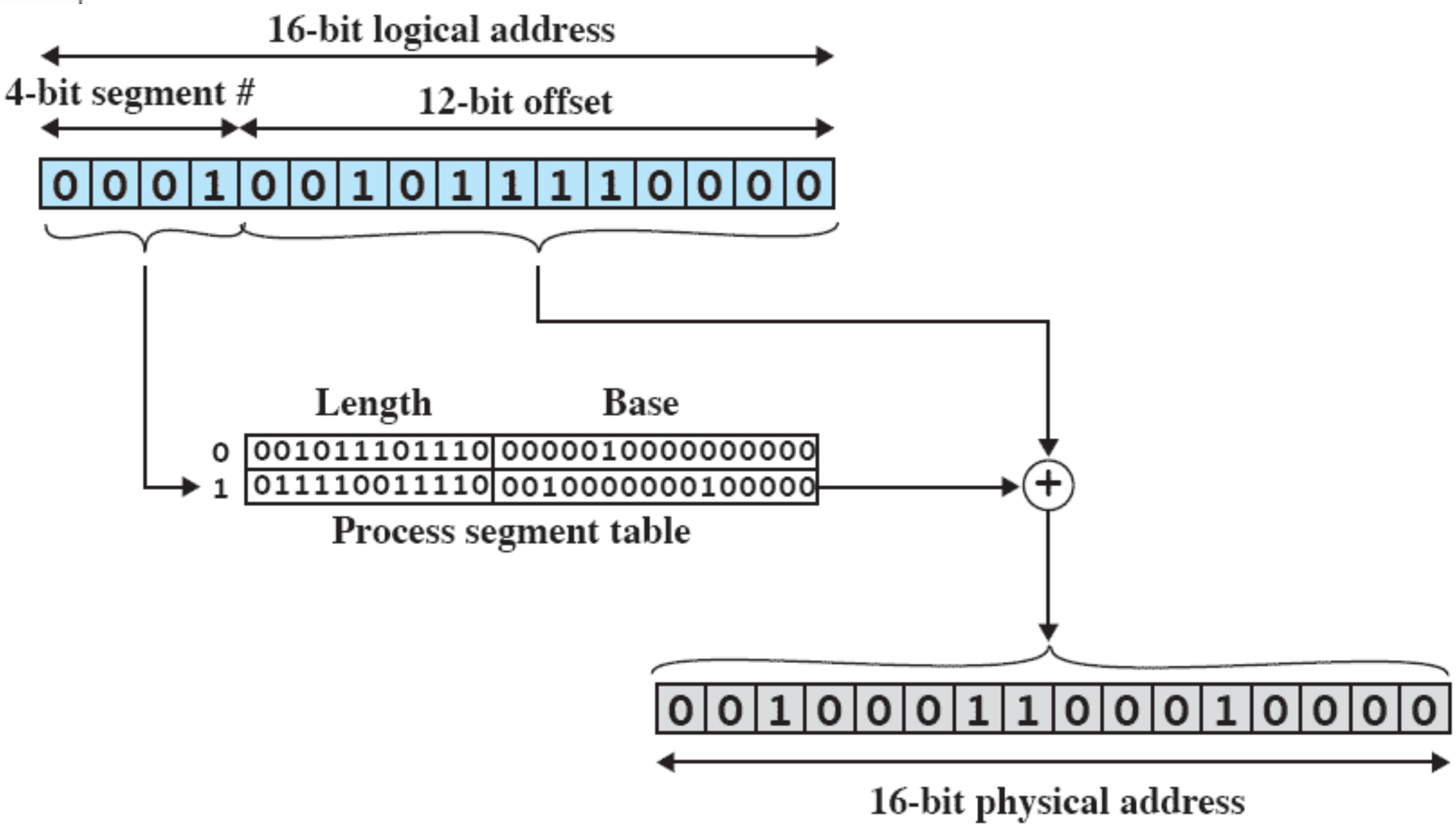
Figure 7.11 Logical Addresses

Paging



(a) Paging

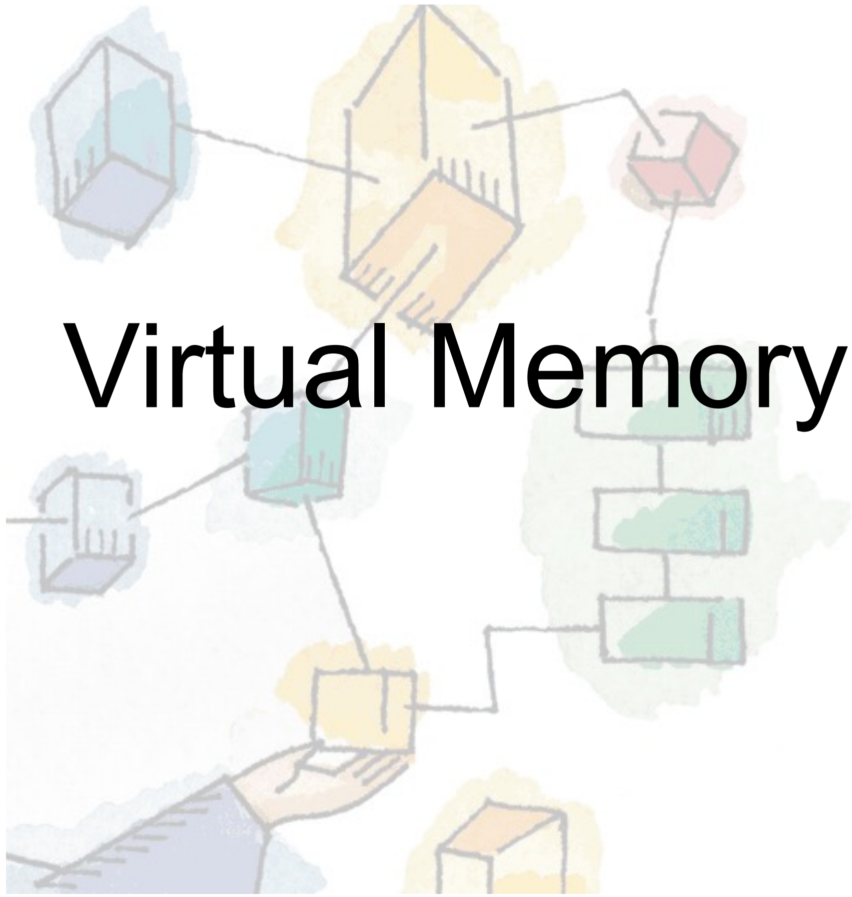
Segmentation



(b) Segmentation

Figure 7.12 Examples of Logical-to-Physical Address Translation

Operating Systems:
Internals and Design Principles, 6/E
William Stallings



Virtual Memory

Dave Bremer
Otago Polytechnic, N.Z.

Terminology

Table 8.1 Virtual Memory 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.

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

- A state in which the system spends most of its time swapping pieces rather than executing instructions.
- To avoid this, the operating system tries to guess which pieces are least likely to be used in the near future.
  - The guess is based on recent history

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
- This suggests that virtual memory may work efficiently

Support Needed for Virtual Memory

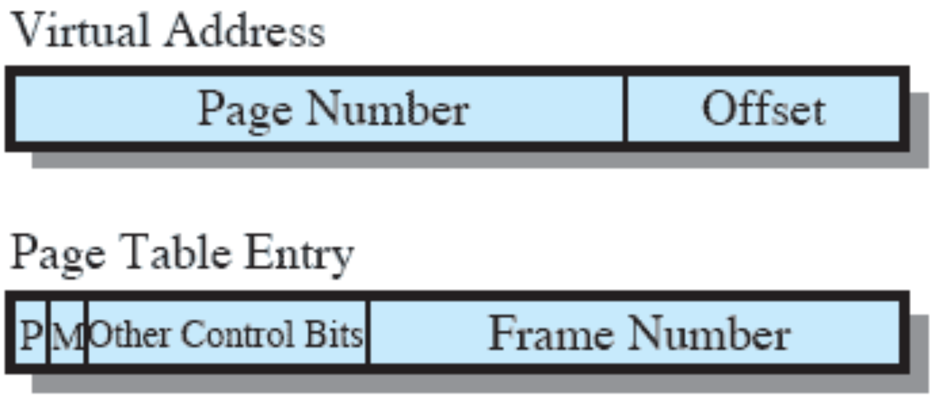
- Hardware must support paging and segmentation
- Operating system must be able to manage the movement of pages and/or segments between secondary memory and main memory

Paging

- Each process has its own page table
- Each page table entry contains the frame number of the corresponding page in main memory
- Two extra bits are needed to indicate:
  - whether the page is in main memory or not
  - Whether the contents of the page has been altered since it was last loaded

(see next slide)

Paging Table



(a) Paging only

Address Translation

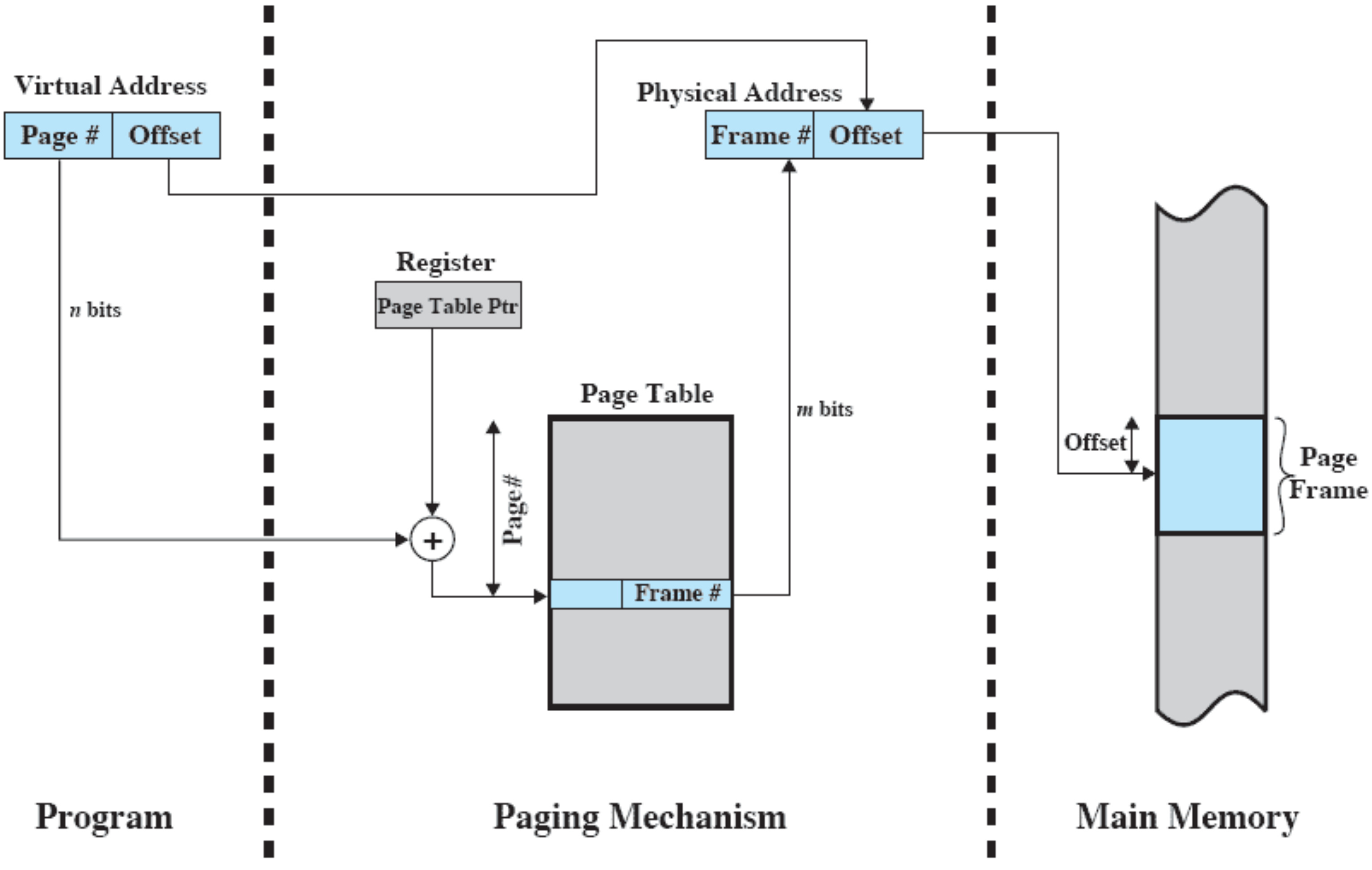


Figure 8.3 Address Translation in a Paging System

Page Tables

- Page tables are also stored in virtual memory
- When a process is running, part of its page table is in main memory

Translation Lookaside Buffer

- Each virtual memory reference can cause two physical memory accesses
  - One to fetch the page table
  - One to fetch the data
- To overcome this problem a high-speed cache is set up for page table entries
  - Called a Translation Lookaside Buffer (TLB)
  - Contains page table entries that have been most recently used

TLB Operation

- 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

Looking into the Process Page Table

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

Translation Lookaside Buffer

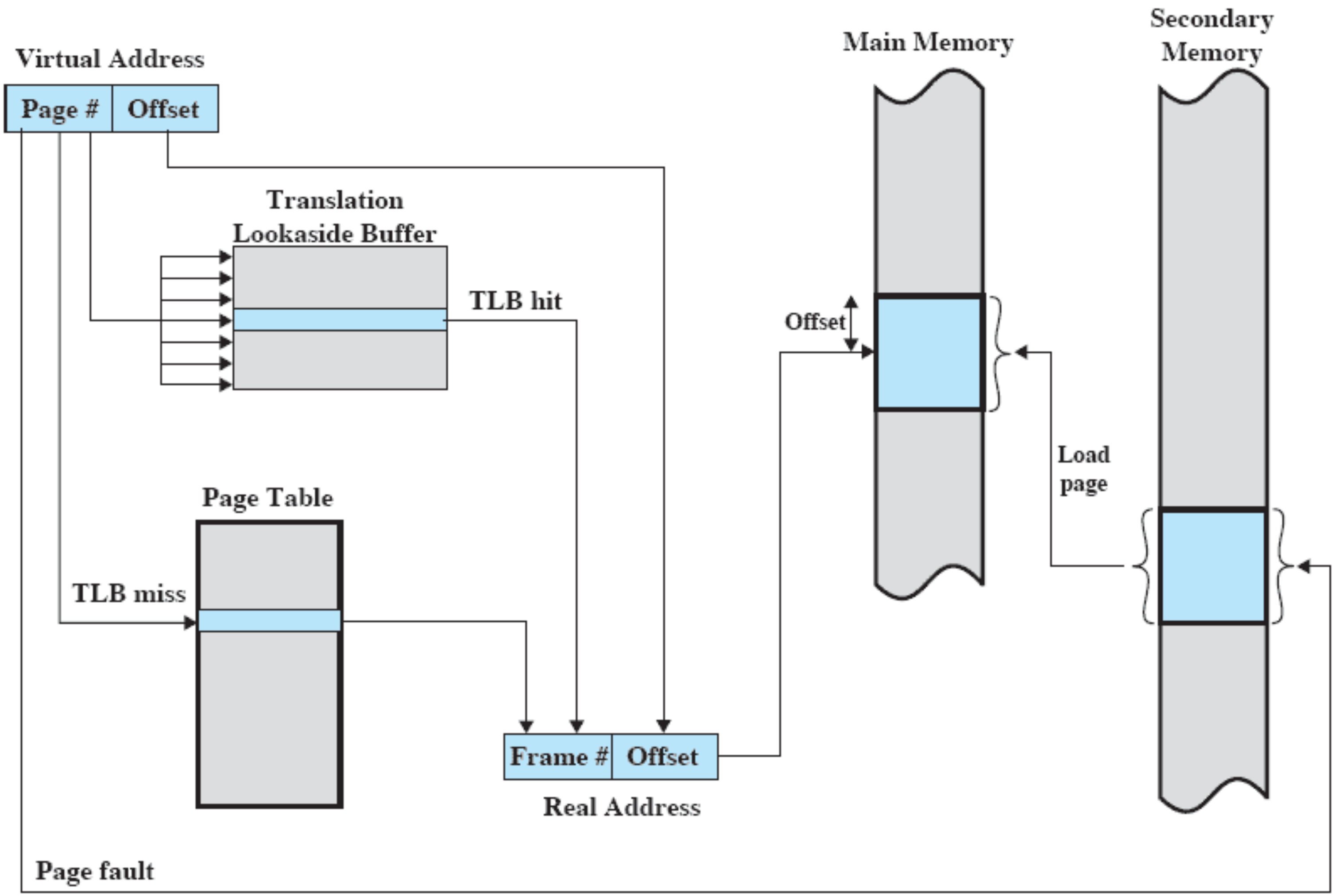


Figure 8.7 Use of a Translation Lookaside Buffer

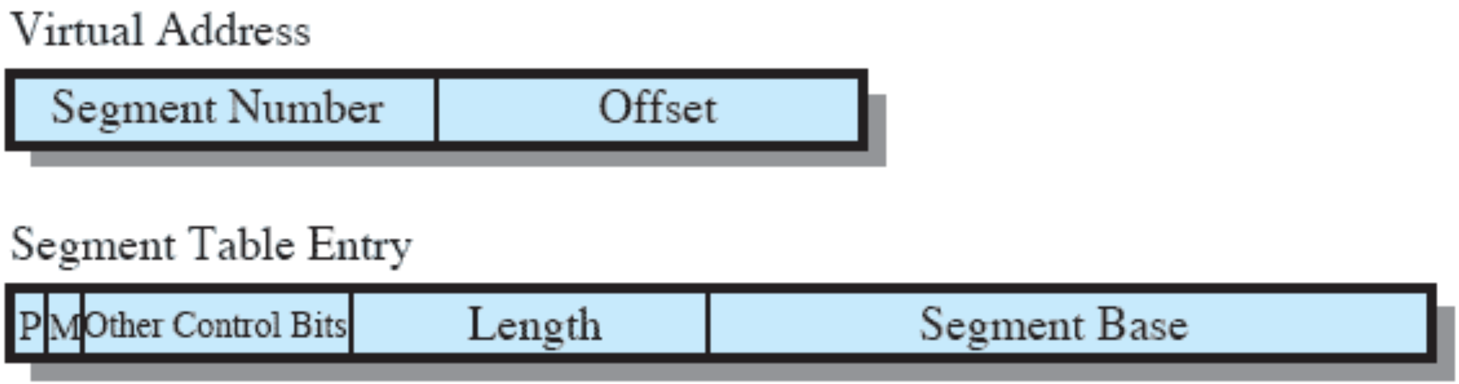
Segmentation

- Segmentation allows the programmer to view memory as consisting of multiple address spaces or segments.
  - May be unequal, dynamic size
  - 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

- Starting address corresponding segment in main memory
- Each entry contains 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



(b) Segmentation only

Address Translation in Segmentation

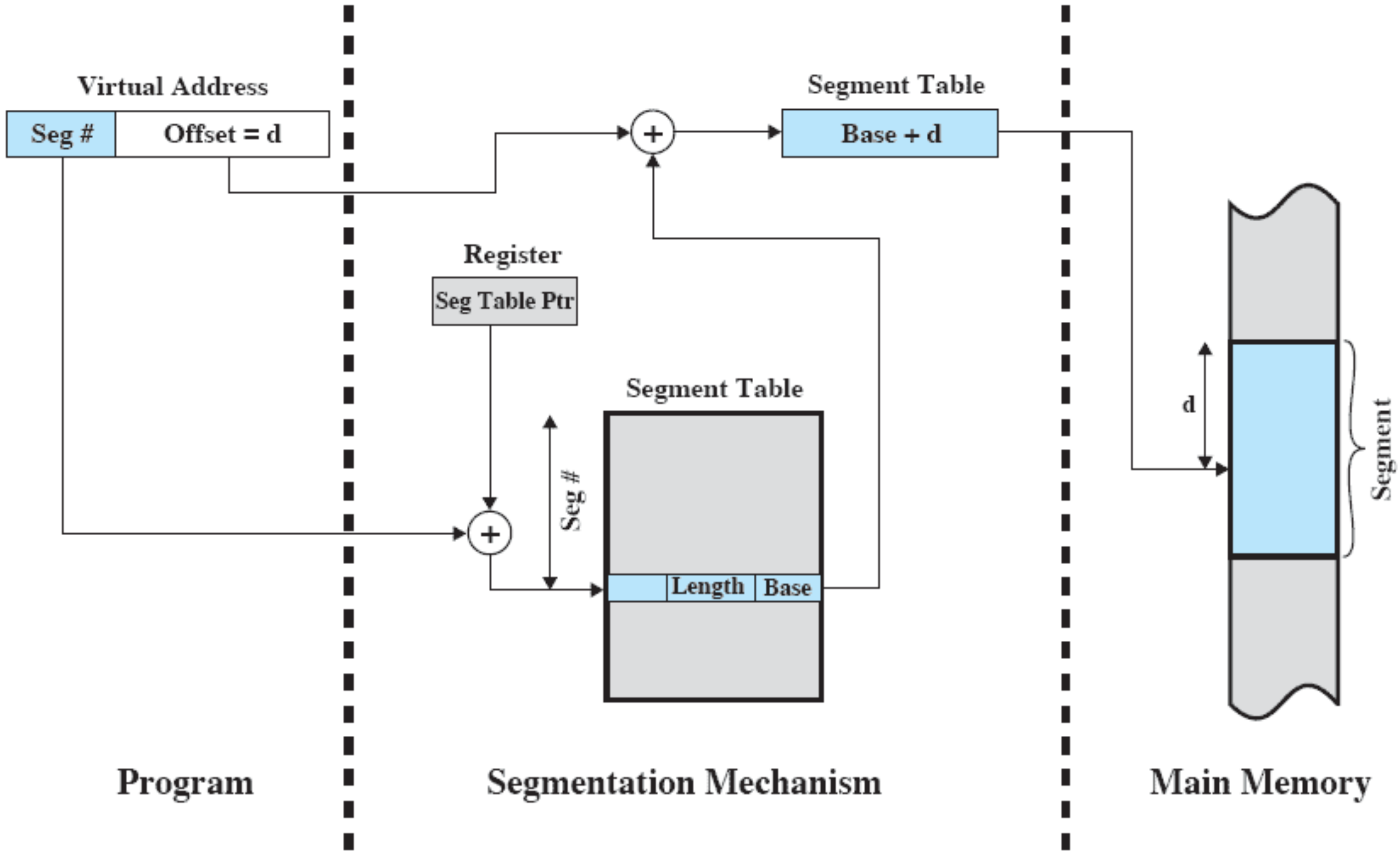
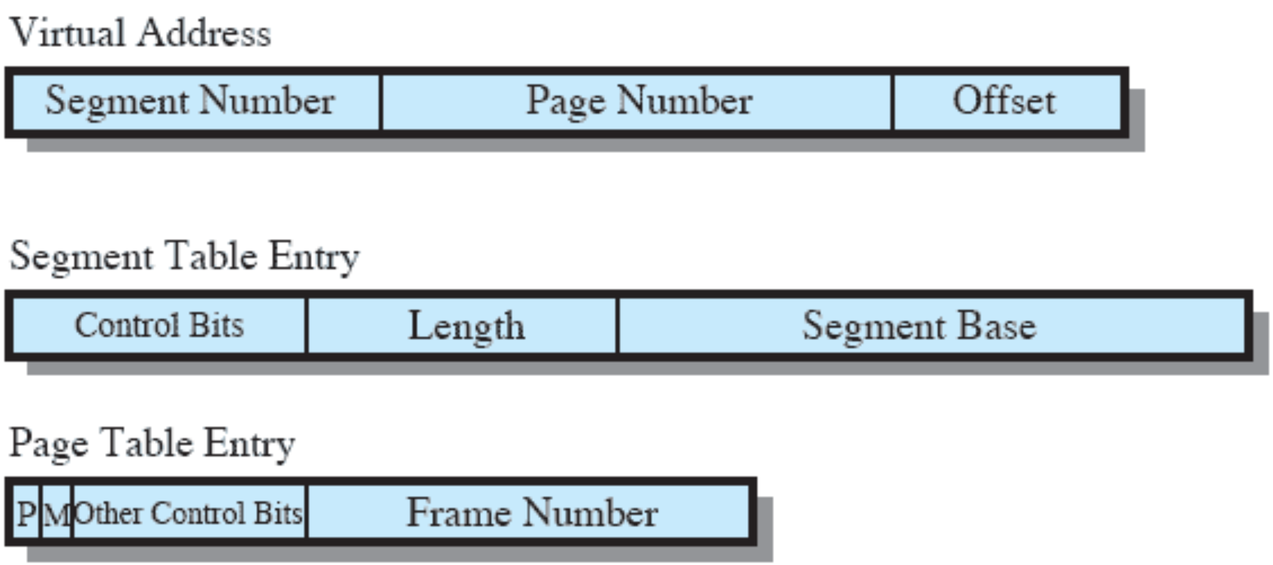


Figure 8.12 Address Translation in a Segmentation System

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 Paging and Segmentation



P= present bit
M = Modified bit

(c) Combined segmentation and paging

Address Translation

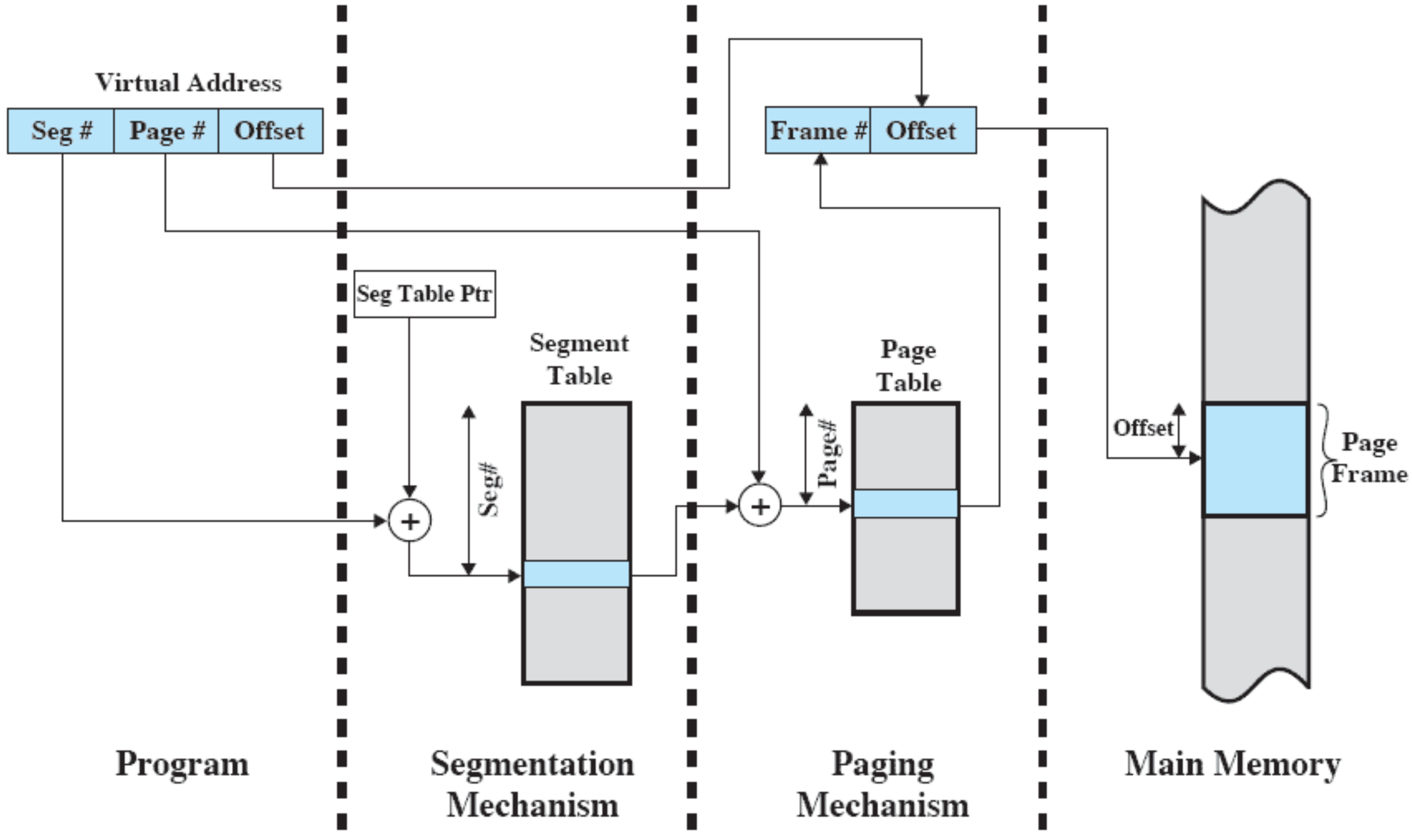
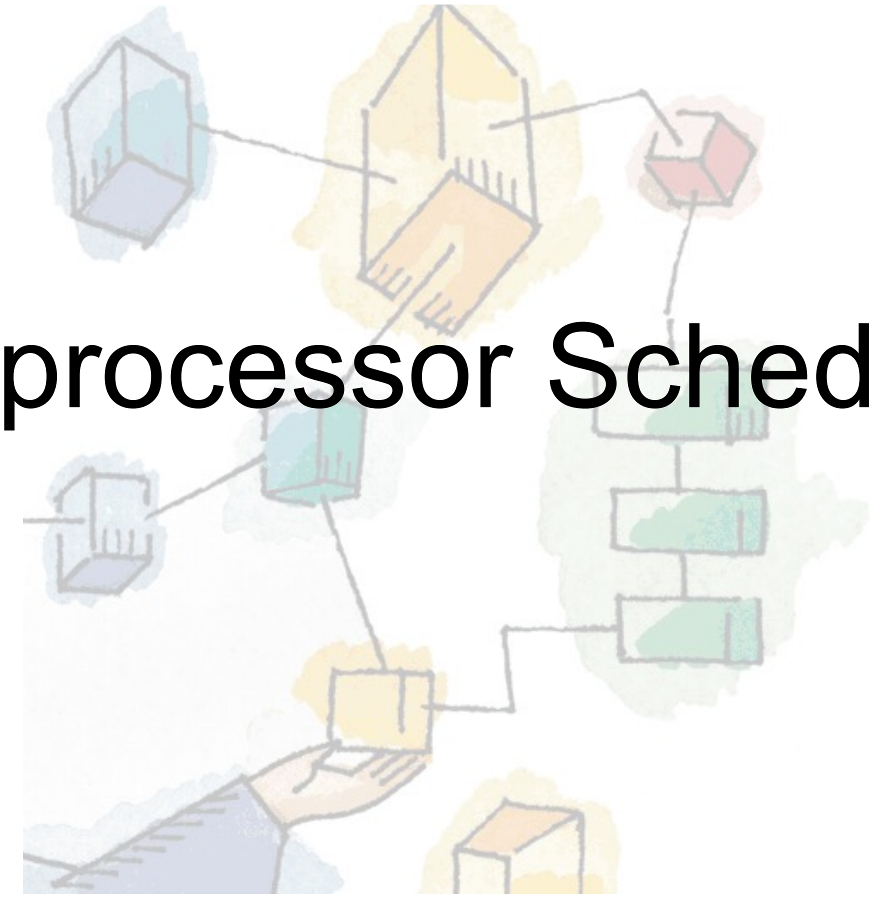


Figure 8.13 Address Translation in a Segmentation/Paging System

Operating Systems: Internals and Design Principles, 6/E
William Stallings



Uniprocessor Scheduling

Dave Bremer
Otago Polytechnic, N.Z.

Scheduling

- An OS must allocate resources amongst competing processes.
- The resource provided by a processor is execution time
  - The resource is allocated by means of a schedule

Types of Scheduling

Table 9.1 Types of Scheduling

Long-term scheduling	The decision to add to the pool of processes to be executed
Medium-term scheduling	The decision to add to the number of processes that are partially or fully in main memory
Short-term scheduling	The decision as to which available process will be executed by the processor
I/O scheduling	The decision as to which process's pending I/O request shall be handled by an available I/O device

Long-Term Scheduling

- Determines which programs are admitted to the system for processing
  - May be first-come-first-served
  - Or according to criteria such as priority, I/O requirements or expected execution time
- Controls the degree of multiprogramming
- More processes, smaller percentage of time each process is executed

Medium-Term Scheduling

- Part of the swapping function
- Swapping-in decisions are based on the need to manage the degree of multiprogramming

Short-Term Scheduling

- Known as the dispatcher
- Executes most frequently
- Invoked when an event occurs
  - Clock interrupts
  - I/O interrupts
  - Operating system calls
  - Signals

Decision Mode

- Specifies the instants in time at which the selection function is exercised.
- Two categories:
  - Nonpreemptive
  - Preemptive

Nonpreemptive vs Premeptive

- Non-preemptive
  - Once a process is in the running state, it will continue until it terminates or blocks itself for I/O
- Preemptive
  - Currently running process may be interrupted and moved to ready state by the OS
  - Preemption may occur when new process arrives, on an interrupt, or periodically.

Process Scheduling Example

- Example set of processes, consider each a batch job

Table 9.4 Process Scheduling Example

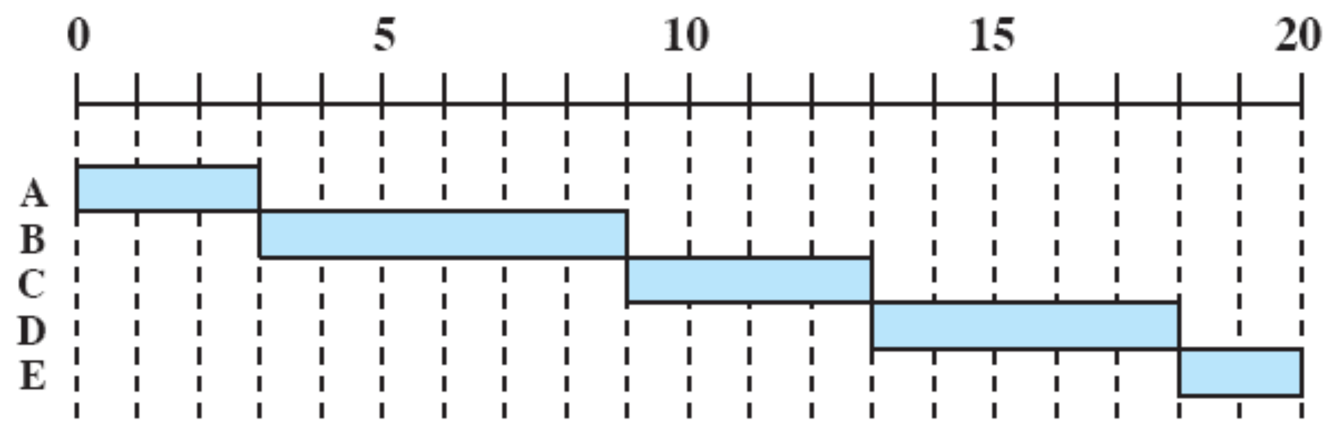
Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

  – Service time represents total execution time

First-Come-First-Served

- Each process joins the Ready queue
- When the current process ceases to execute, the longest process in the Ready queue is selected

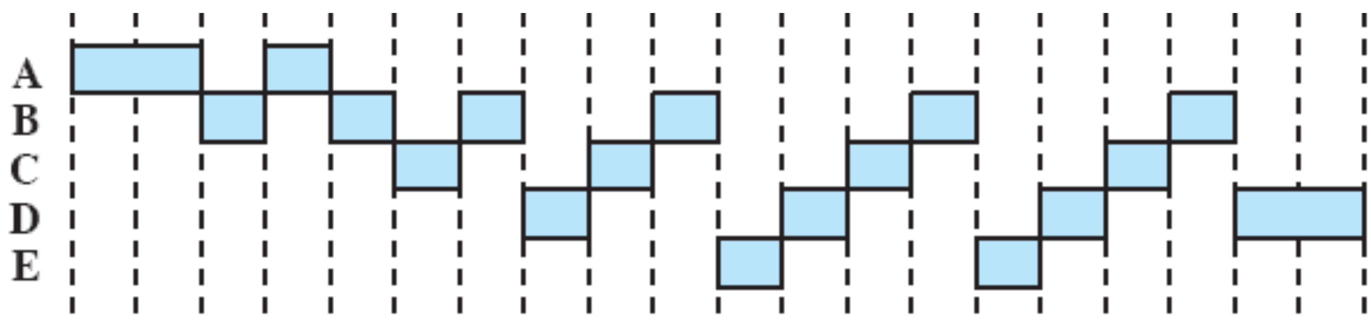
First-Come-First Served (FCFS)



Round Robin

- Uses preemption based on a clock
  - also known as time slicing, because each process is given a slice of time before being preempted.

Round-Robin (RR), $q = 1$



‘Virtual Round Robin’

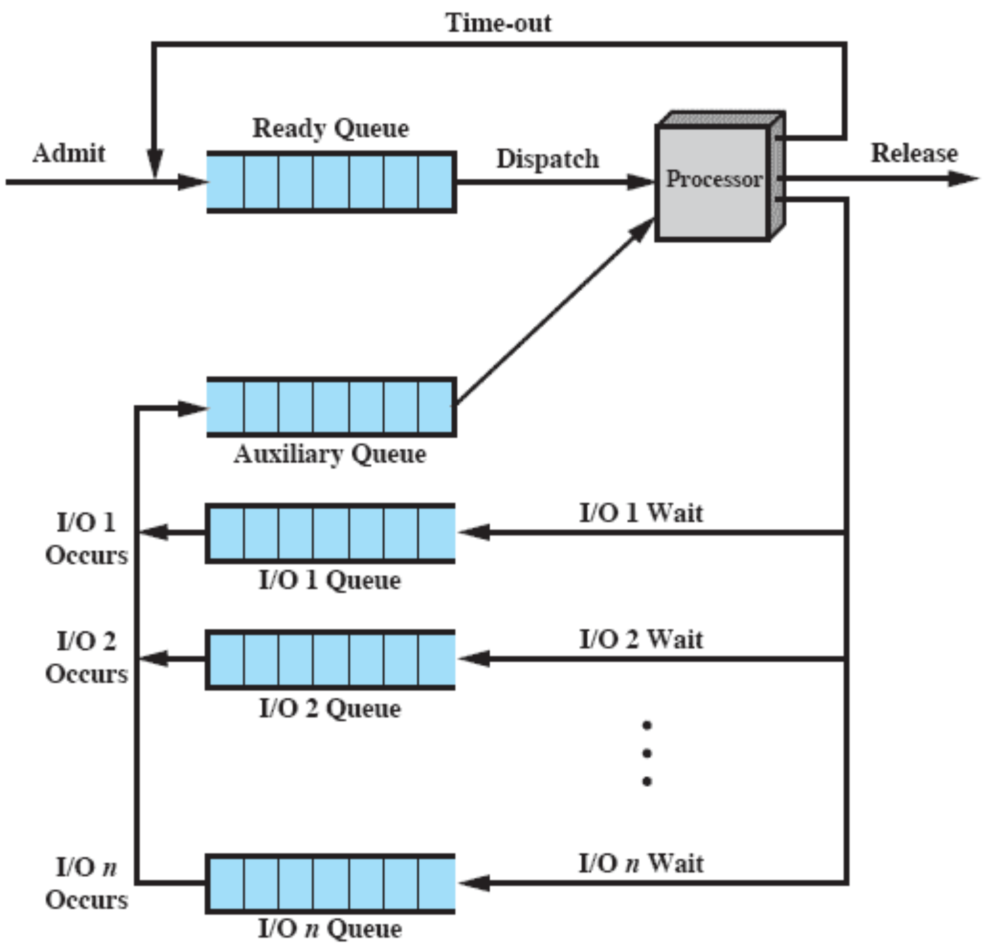
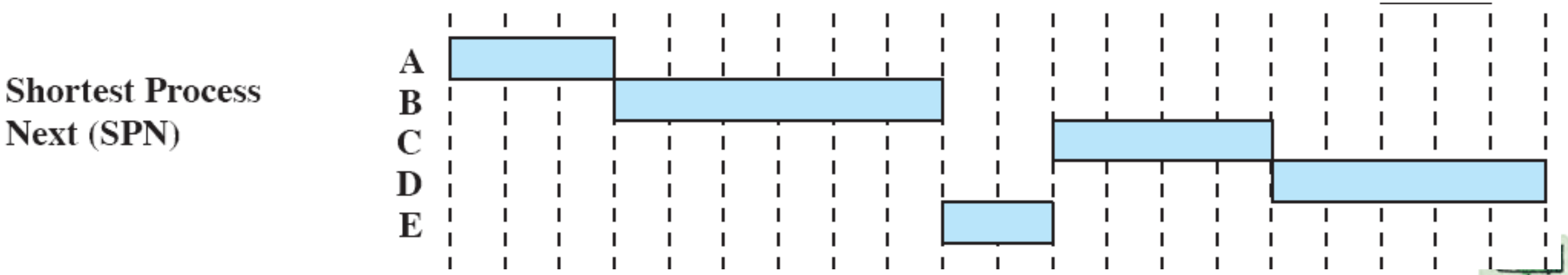


Figure 9.7 Queuing Diagram for Virtual Round-Robin Scheduler

Shortest Process Next

- Nonpreemptive policy
- Process with shortest expected processing time is selected next
- Short process jumps ahead of longer processes

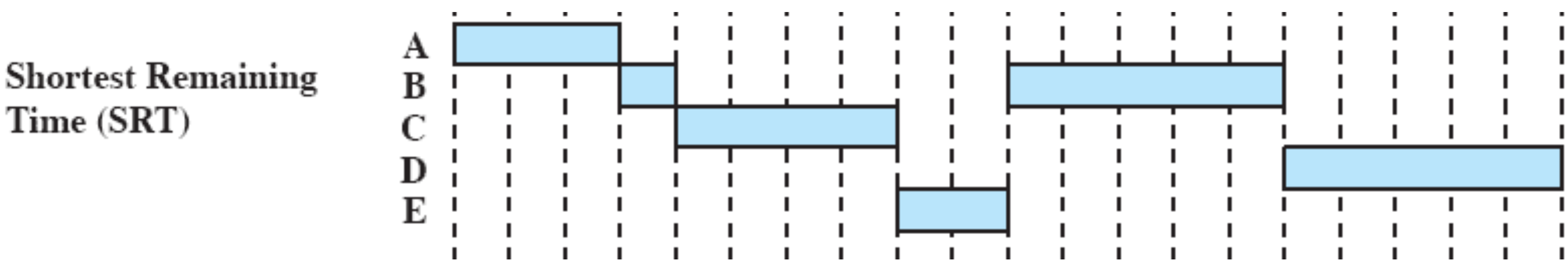


Shortest Process Next

- Predictability of longer processes is reduced
- If estimated time for process not correct, the operating system may abort it
- Possibility of starvation for longer processes

Shortest Remaining Time

- Preemptive version of shortest process next policy
- Must estimate processing time and choose the shortest

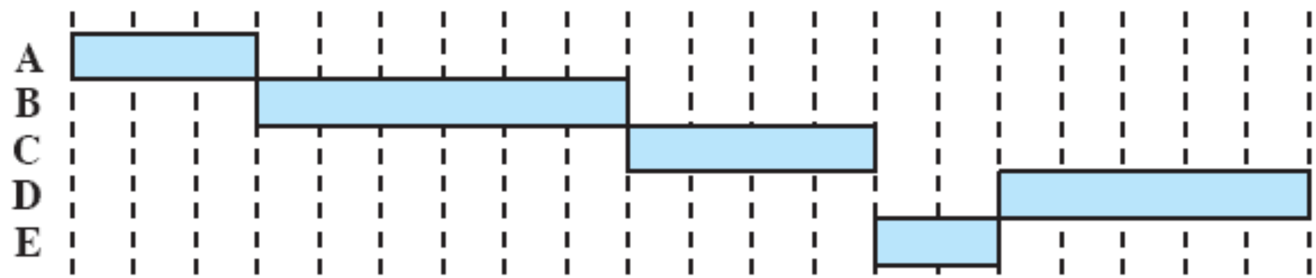


Highest Response Ratio Next

- Choose next process with the greatest ratio

$$Ratio = \frac{time\ spent\ waiting + expected\ service\ time}{expected\ service\ time}$$

Highest Response Ratio Next (HRRN)



Operating Systems: Internals and Design Principles, 6/E
William Stallings

Real-Time Scheduling

Dave Bremer
Otago Polytechnic, N.Z.

Real-Time Systems

- A real-time system requires that results be produced within a specified deadline period
- An embedded system is a computing device that is part of a larger system (I.e. automobile, airliner)
- A safety-critical system is a real-time system with catastrophic results in case of

Terms and Definitions

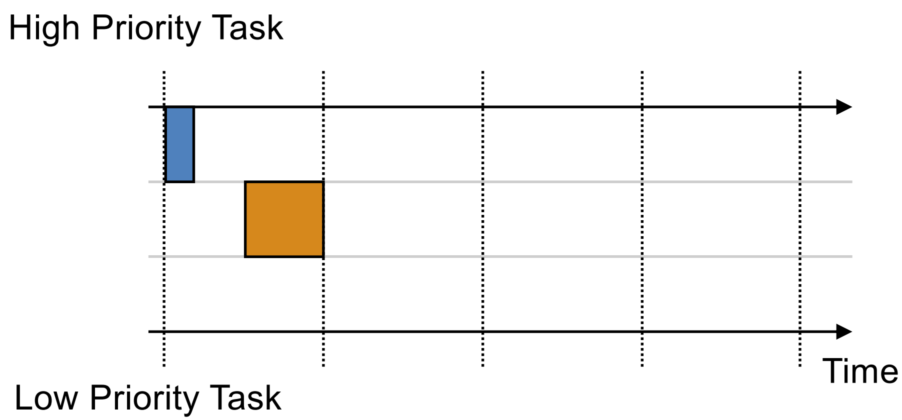
- Tardiness: Specifies the amount of time by which a task misses its deadline. Its is equal to the difference between completion time and deadline

- Laxity: Is defined as deadline minus remaining computation time. The laxity of task is the maximum amount of time it can wait and still meets its deadline

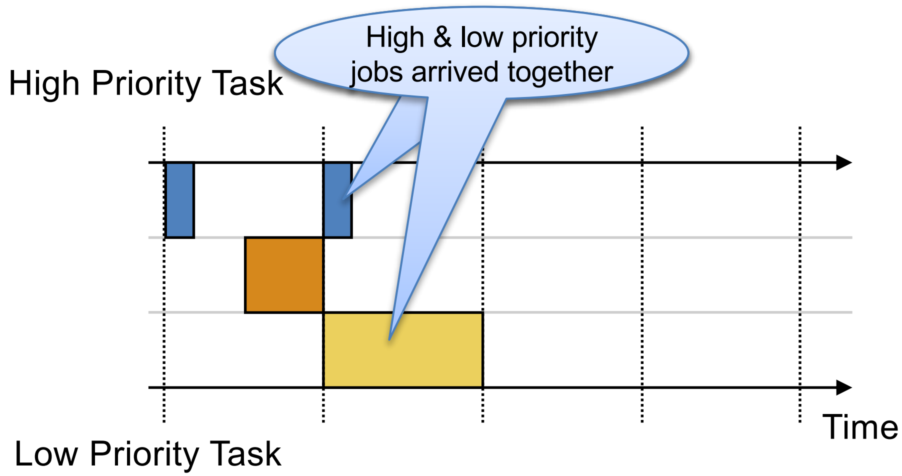
Types of Real Time Tasks

- Hard Real-time task
  - Air traffic control
  - Vehicle subsystems control
  - Nuclear power plant control
- Soft Real-time Task
  - Multimedia transmission and reception
  - Networking, telecom (cellular) networks
  - Web sites and services
  - Computer games.

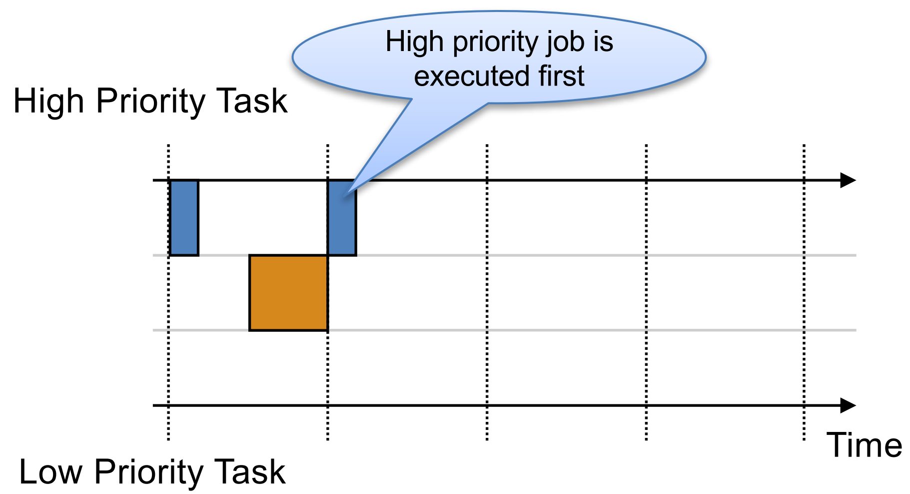
Preemptive Fixed Priority Scheduling



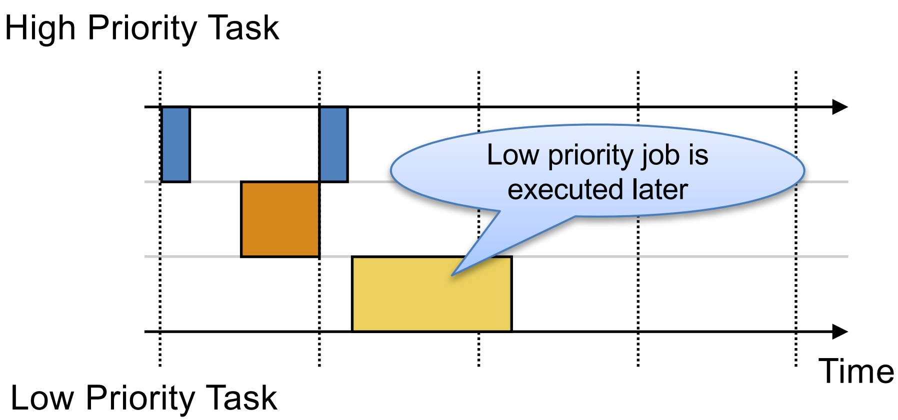
Preemptive Fixed Priority Scheduling



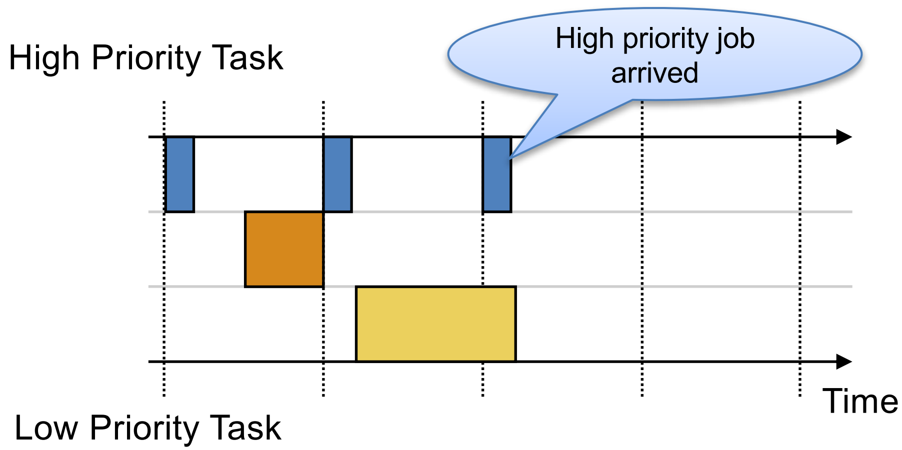
Preemptive Fixed Priority Scheduling



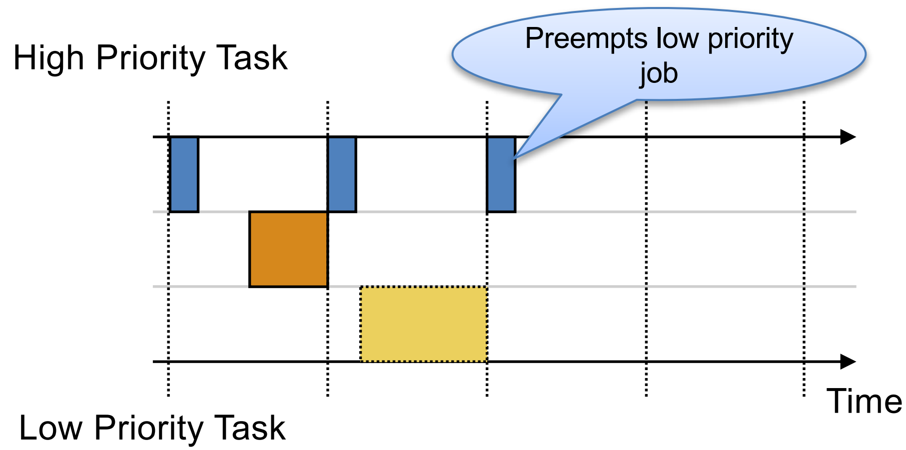
Preemptive Fixed Priority Scheduling



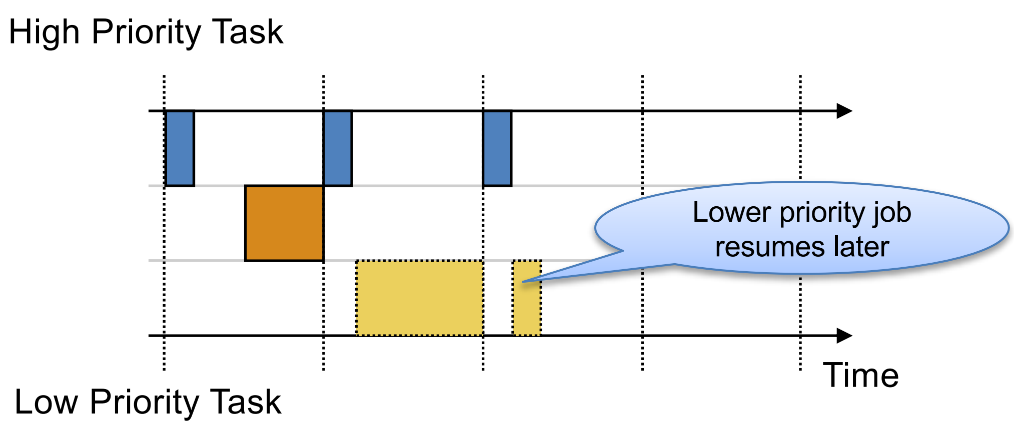
Preemptive Fixed Priority Scheduling



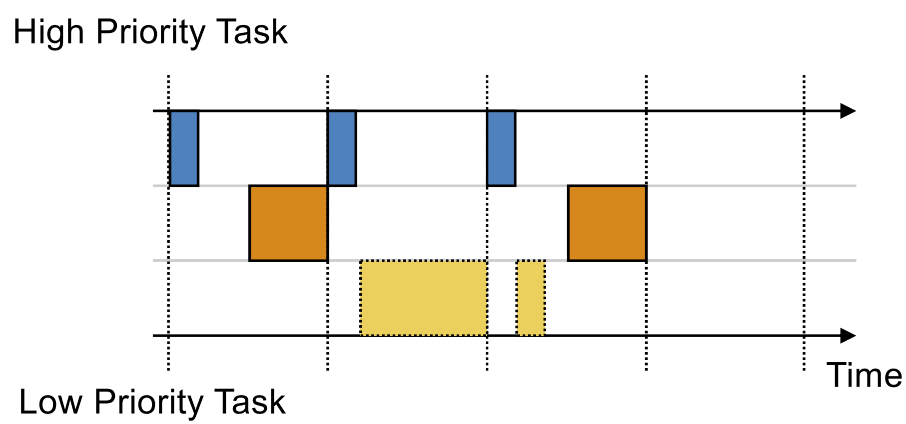
Preemptive Fixed Priority Scheduling



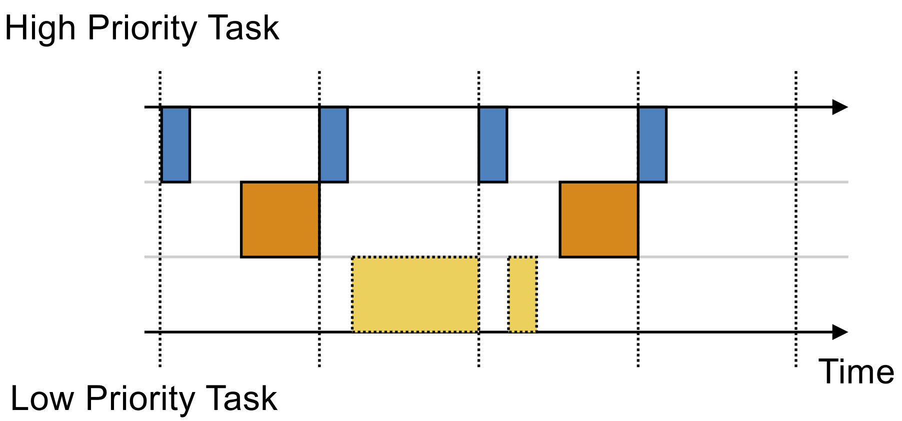
Preemptive Fixed Priority Scheduling



Preemptive Fixed Priority Scheduling

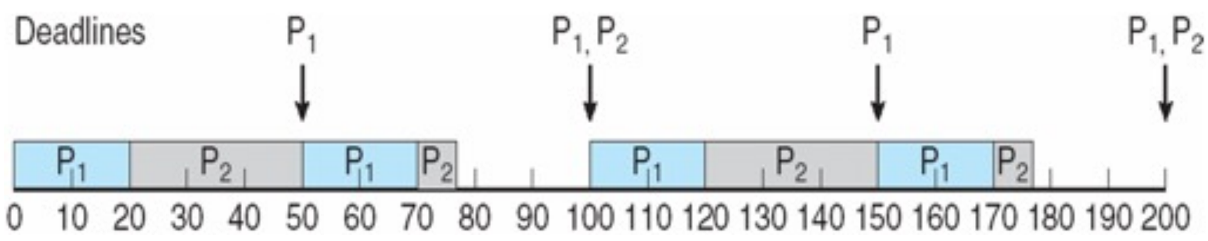


Preemptive Fixed Priority Scheduling



Rate Montonic Scheduling

- A priority is assigned based on the inverse of its period
- Shorter periods = higher priority
- Longer periods = lower priority
- P_1 is assigned a higher priority than P_2.



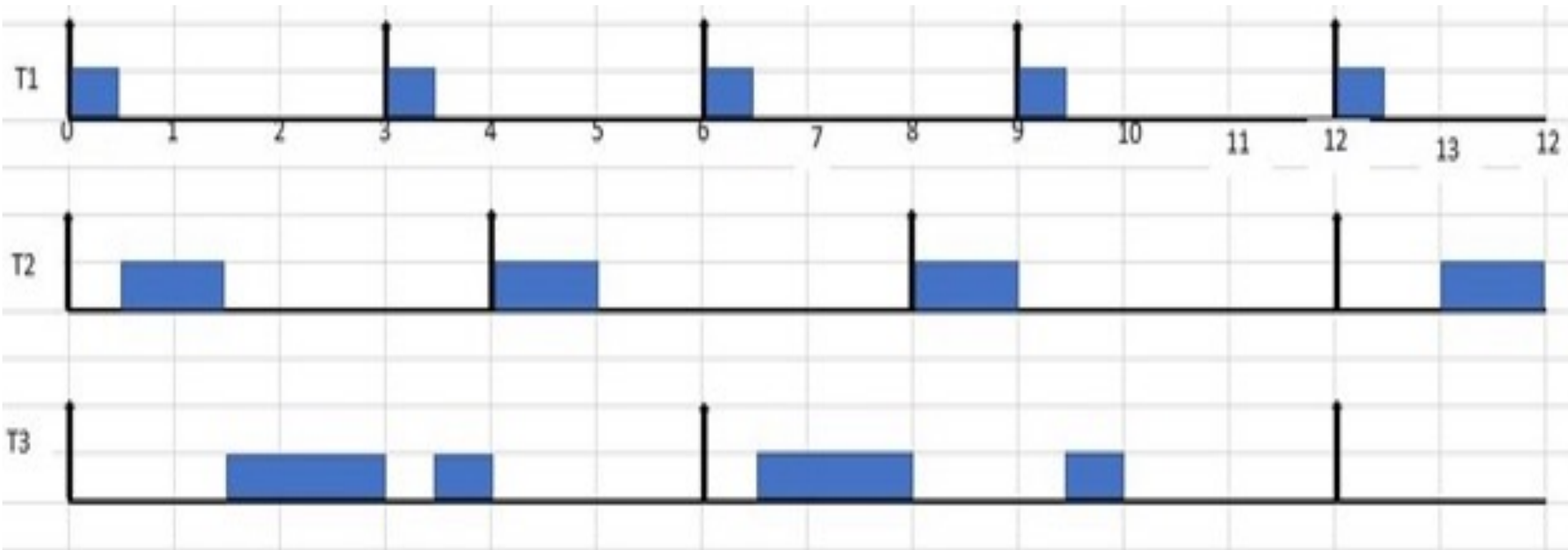
Example

Tasks	Release time(ri)	Execution time(Ci)	Deadline (Di)	Time period(Ti)
T1	0	0.5	3	3
T2	0	1	4	4
T3	0	2	6	6

U= 0.5/3 +1/4 +2/6 = 0.167+ 0.25 + 0.333 = 0.75

As processor utilization is less than 1 or 100% so task set is schedulable and it also satisfies the rate monotonic scheduling algorithm critieria

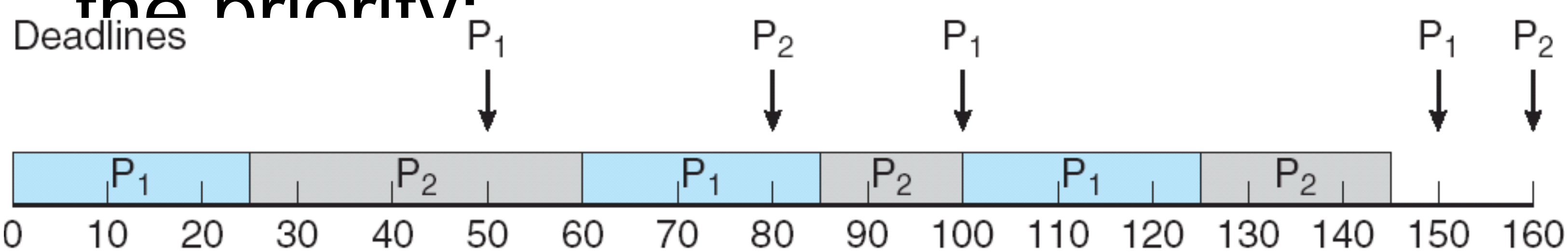
Solution



Earliest Deadline First Scheduling

- Priorities are assigned according to deadlines:

  the earlier the deadline, the higher the priority;



EDF Scheduling

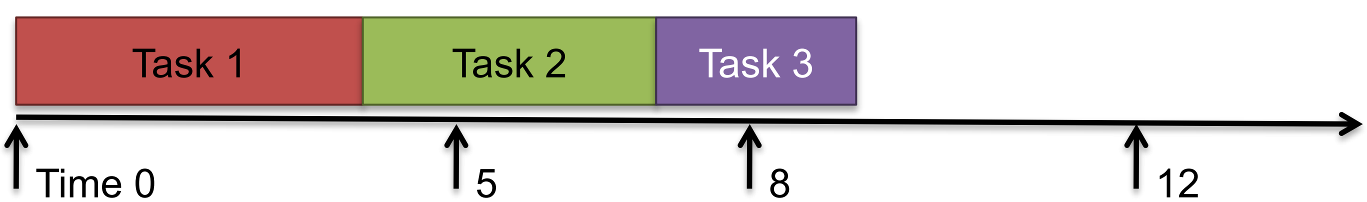
Earliest Deadline First (EDF) scheduling

- Simple, yet effective

- Whichever task has next deadline gets to run

Task 1	Deadline: 5 Exec time: 4	Task 2	Deadline: 8 Exec time: 3	Task 3	Deadline: 12 Exec time: 2

- Theory exists to analyze such systems



Example

Task	Release time(ri)	Execution Time(Ci)	Deadline (Di)	Time Period(Ti)
T1	0	1	4	4
T2	0	2	6	6
T3	0	3	8	8

U= 1/4 +2/6 +3/8 = 0.25 + 0.333 +0.375 = 0.95 = 95%

Solution