Today's class

- Memory management
- Virtual memory

UPPSALA UNIVERSITET

Informationsteknologi

Memory Management

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

Five Requirements of Memory Management

- Relocation
- Protection
- Sharing
- Logical organization
- Physical organization

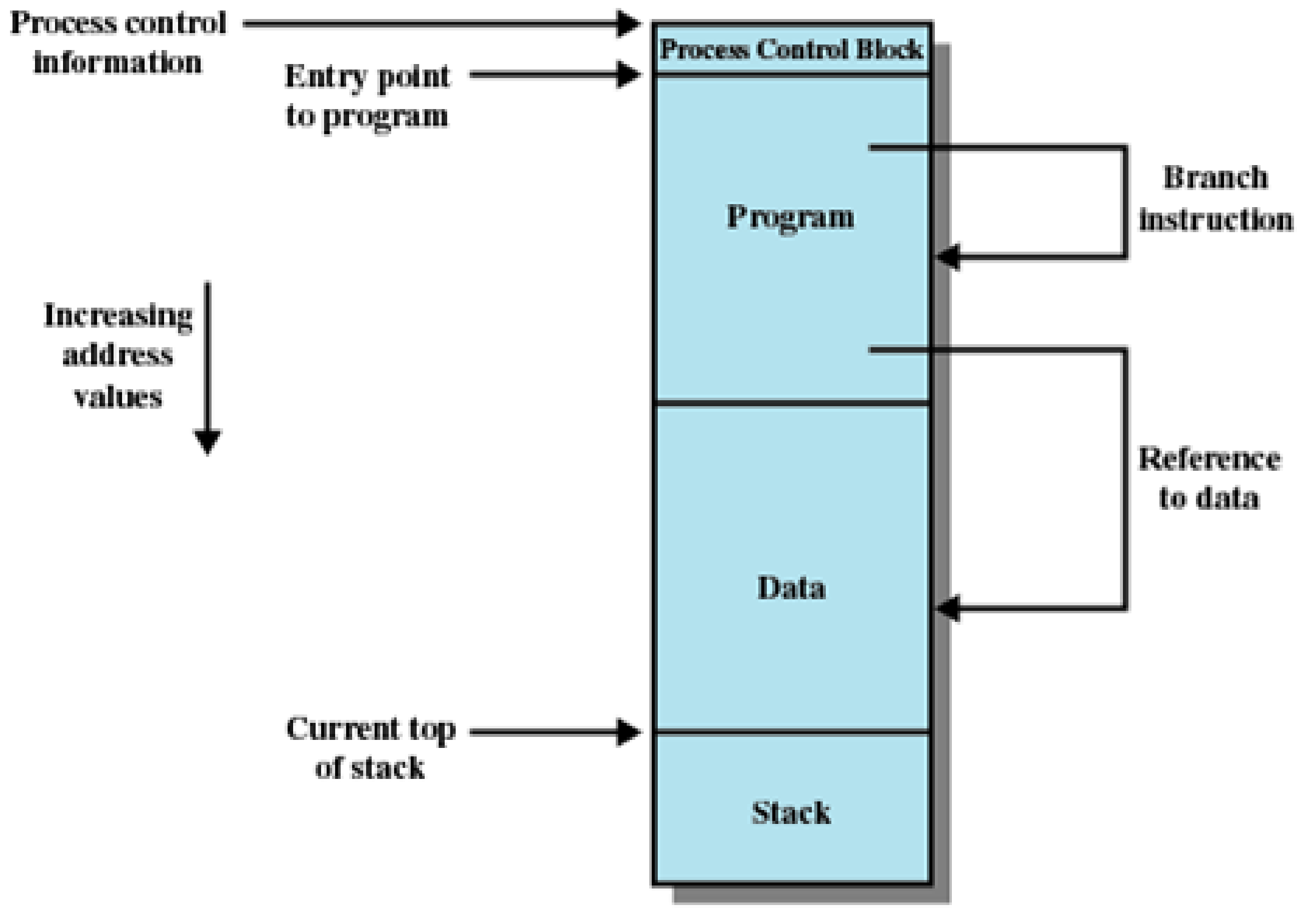
UPPSALA
UNIVERSITET

Informationsteknologi

Memory Management Requirements

- Relocation
  - Programmer does not know where the program will be placed in memory when it is executed
  - While the program is executing, it may be swapped to disk and returned to main memory at a different location (relocated)
  - Memory references must be translated in the code to actual physical memory address

Addressing Requirements for a Process



UPPSALA UNIVERSITET

Informationsteknologi

Memory Management Requirements

- Protection
  - Processes should not be able to reference memory locations in another process without permission
  - Impossible to check absolute addresses at compile time; must be checked at run time
  - Memory protection requirement must be satisfied by the processor (hardware) rather than the operating system (software)
    - Operating system cannot anticipate all of the memory references a program will make

UPPSALA UNIVERSITET

Informationsteknologi

Memory Management Requirements

- Sharing
  - Allow several processes to access the same portion of memory
  - Better to allow each process access to the same copy of the program rather than have their own separate copy

UPPSALA UNIVERSITET

Informationsteknologi

Memory Management Requirements

- Logical Organization
  - Programs are written in modules
  - Modules can be written and compiled independently
  - Different degrees of protection given to modules (read-only, execute-only)
  - Share modules among processes

Memory Management Requirements

- Physical Organization
  - Memory available for a program plus its data may be insufficient
    - Overlaying allows various modules to be assigned the same region of memory
  - Programmer does not know how much space will be available

UPPSALA UNIVERSITET

Informationsteknologi

Fixed Partitioning

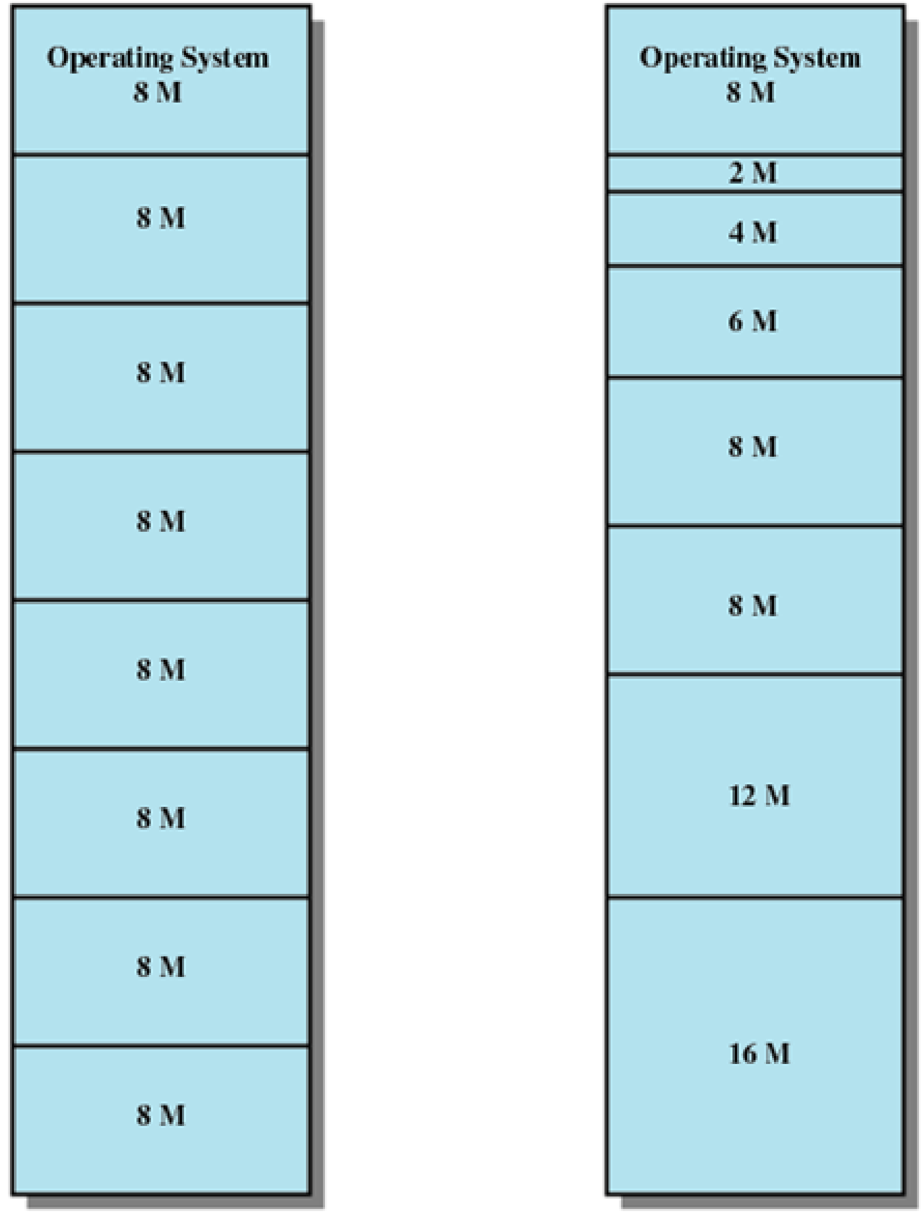
- Equal-size partitions
  - Any process whose size is less than or equal to the partition size can be loaded into an available partition
  - If all partitions are full, the operating system can swap a process out of a partition
  - A program may not fit in a partition. The programmer must design the program with overlays

UPPSALA UNIVERSITET

Informationsteknologi

Fixed Partitioning

- Main memory use is inefficient. Any program, no matter how small, occupies an entire partition. This is called ***internal fragmentation***.
- This problem can be lessened by using unequal sized partitions.



(a) Equal-size partitions (b) Unequal-size partitions

Placement Algorithm with Partitions

- Equal-size partitions
  - Because all partitions are of equal size, it does not matter which partition is used
- Unequal-size partitions
  - 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

UPPSALA UNIVERSITET

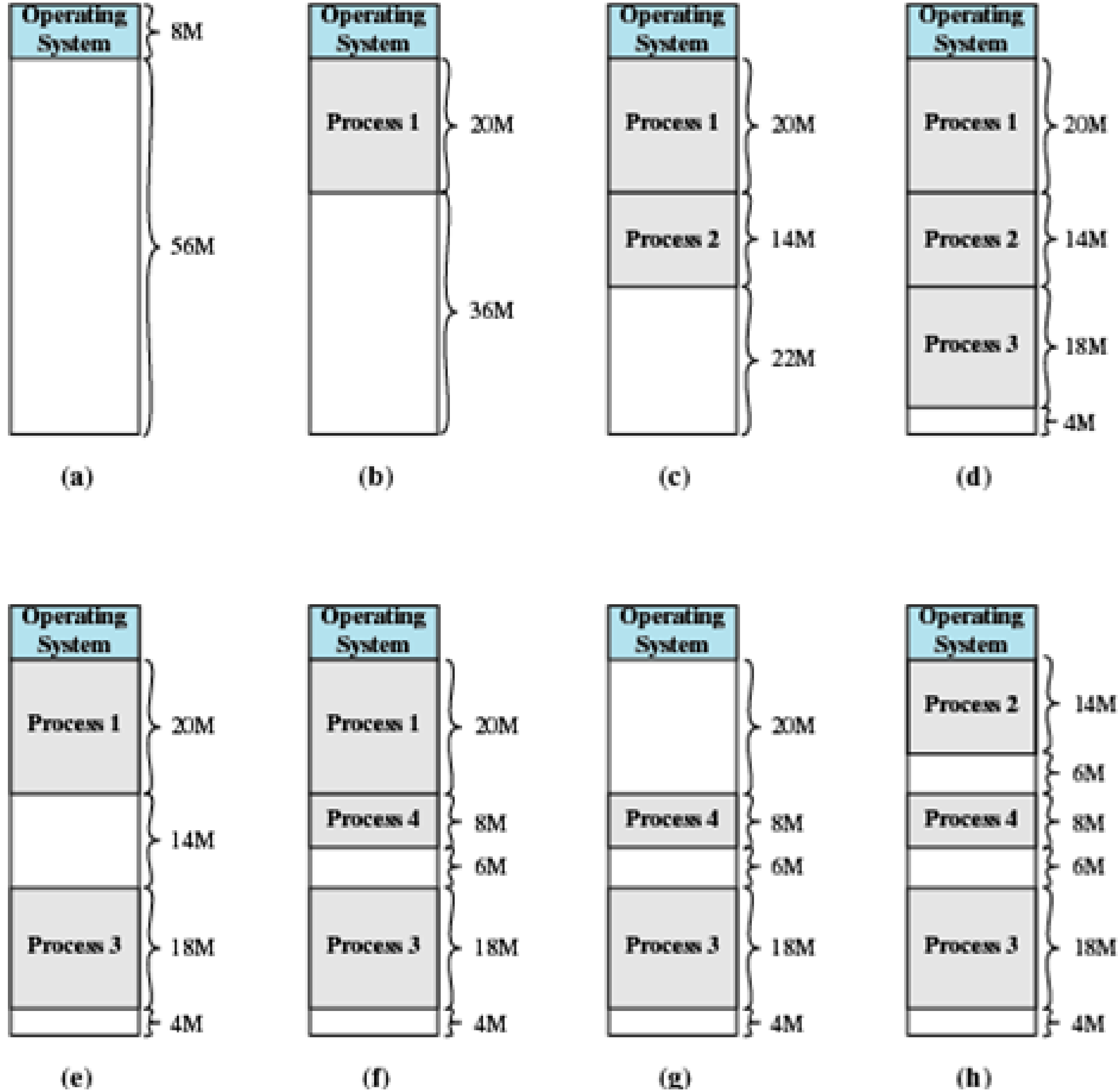
Informationsteknologi

Dynamic Partitioning

- Partitions are of variable length and number
- Process is allocated exactly as much memory as required
- Eventually get holes in the memory. This is called ***external fragmentation***
- Must use compaction to shift processes so they are contiguous and all free memory is in one block

UPPSALA UNIVERSITET

Informationsteknologi



Dynamic Partitioning Placement Algorithm

- 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 Placement Algorithm

- 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

UPPSALA UNIVERSITET

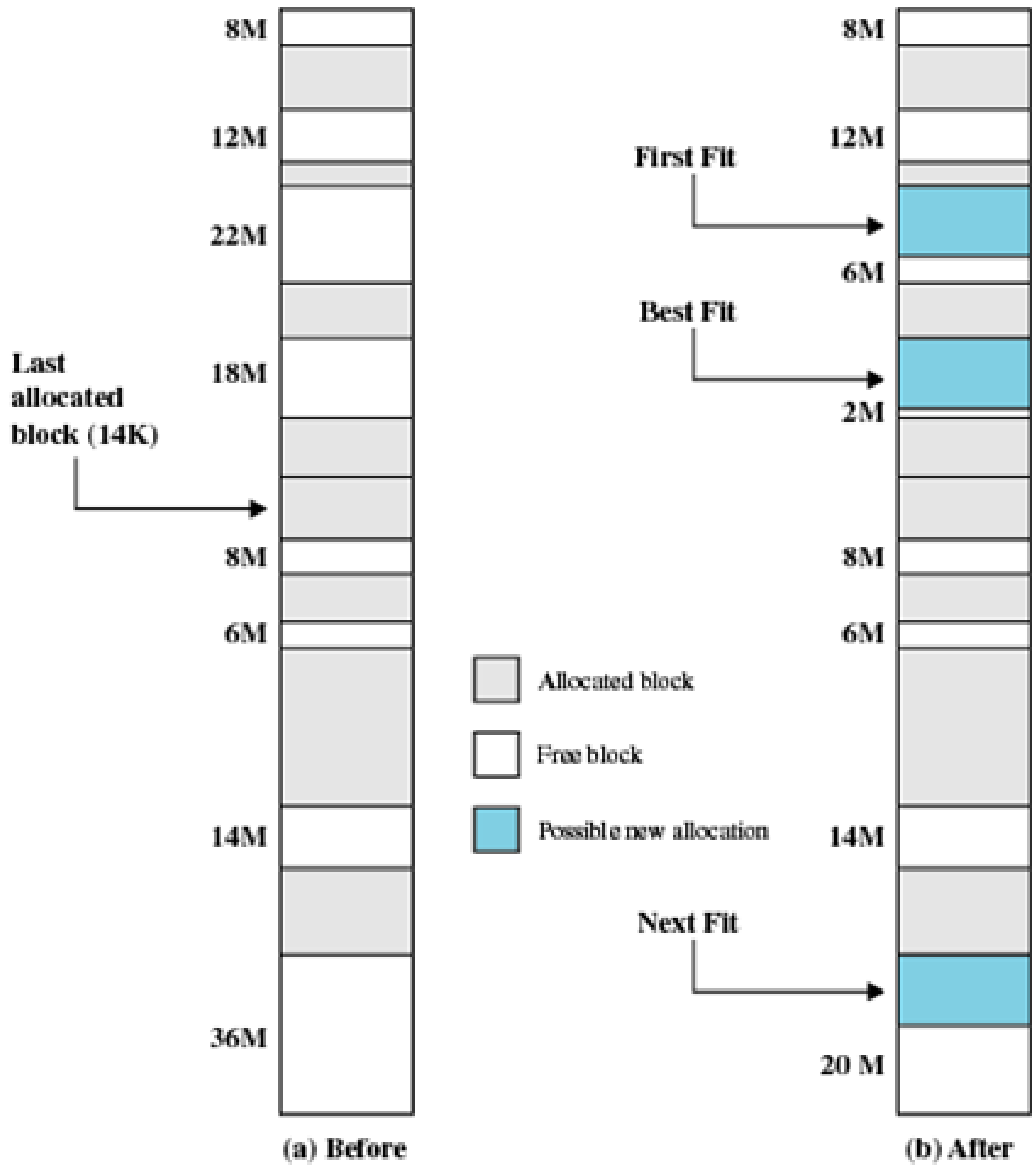
Informationsteknologi

Dynamic Partitioning Placement Algorithm

- 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

UPPSALA UNIVERSITET

Informationsteknologi



UPPSALA UNIVERSITET

Informationsteknologi

Buddy System

- Entire space available is treated as a single block of 2^U
- If a request of size s such that $2^{U-1} < s <= 2^U$, entire block is allocated
  - Otherwise block is split into two equal buddies
  - Process continues until smallest block greater than or equal to s is generated

UPPSALA UNIVERSITET

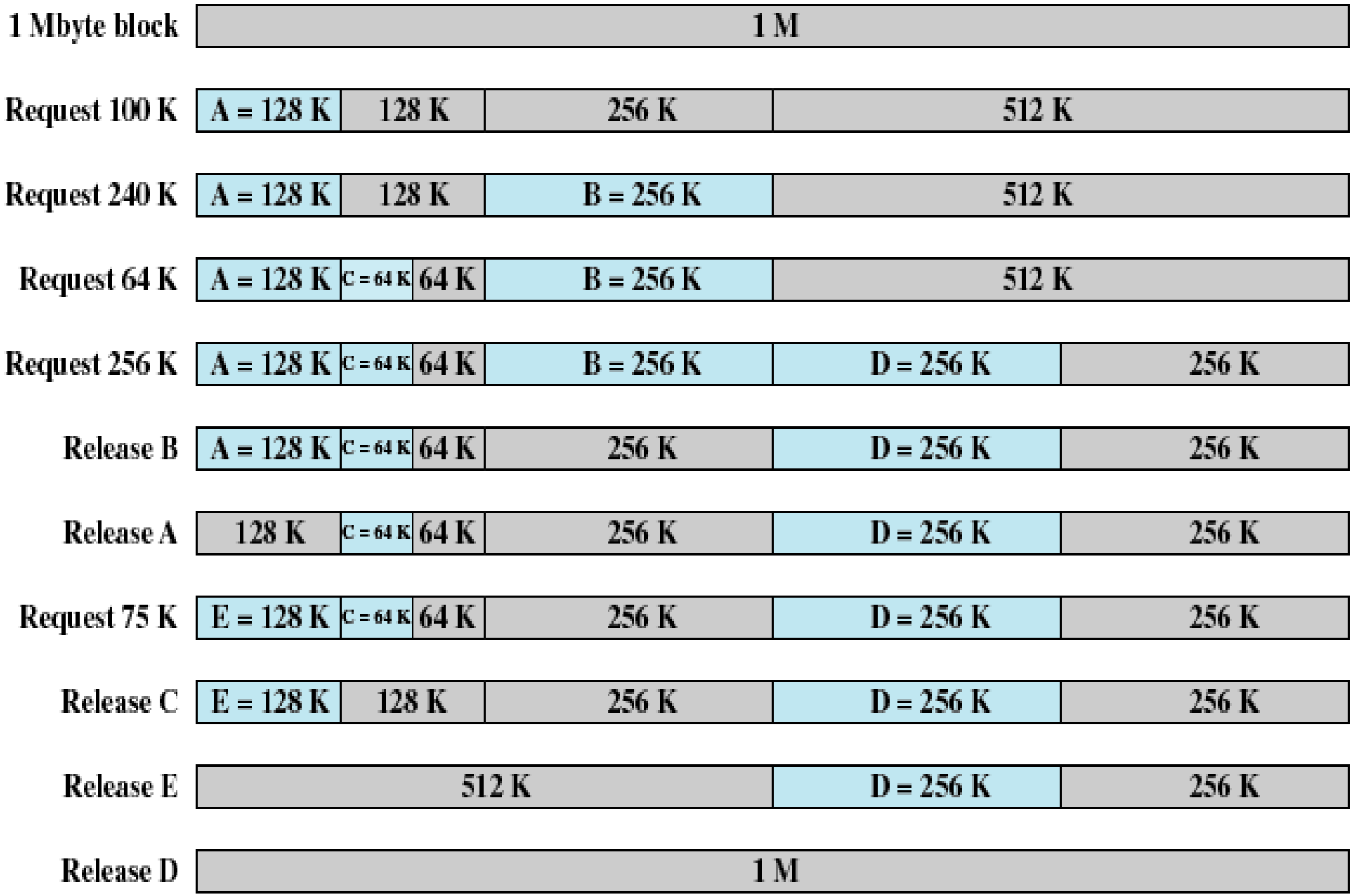
Informationsteknologi

Buddy System

- Maintains a list of hole (unallocated blocks) of each size 2^i
- A hole may be removed from the (i + 1) list by splitting it in half to create two buddies of size 2^i in the i list
- Whenever a pair of buddies on the i list both become unallocated, they are removed from that list and joined into a single block on the (i + 1) list

UPPSALA UNIVERSITET

Informationsteknologi



UPPSALA UNIVERSITET

Informationsteknologi

Relocation

- When program 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 which means different absolute memory locations

UPPSALA UNIVERSITET

Informationsteknologi

Addresses

- Logical
  - Reference to a memory location independent of the current assignment of data to memory
  - Translation must be made to the physical address
- Relative
  - Address expressed as a location relative to some known point
- Physical
  - The absolute address or actual location in main memory

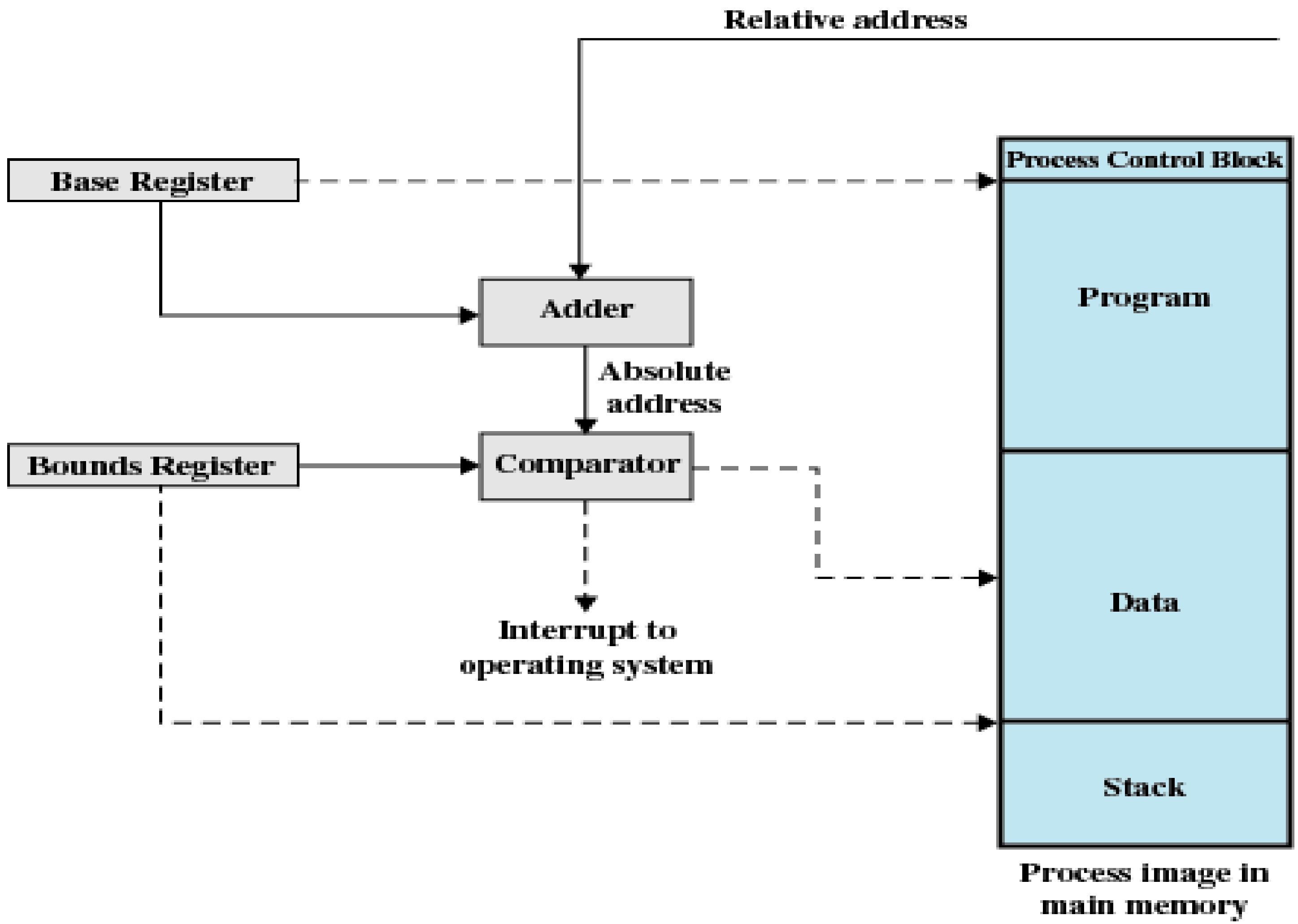


Figure 7.8 Hardware Support for Relocation

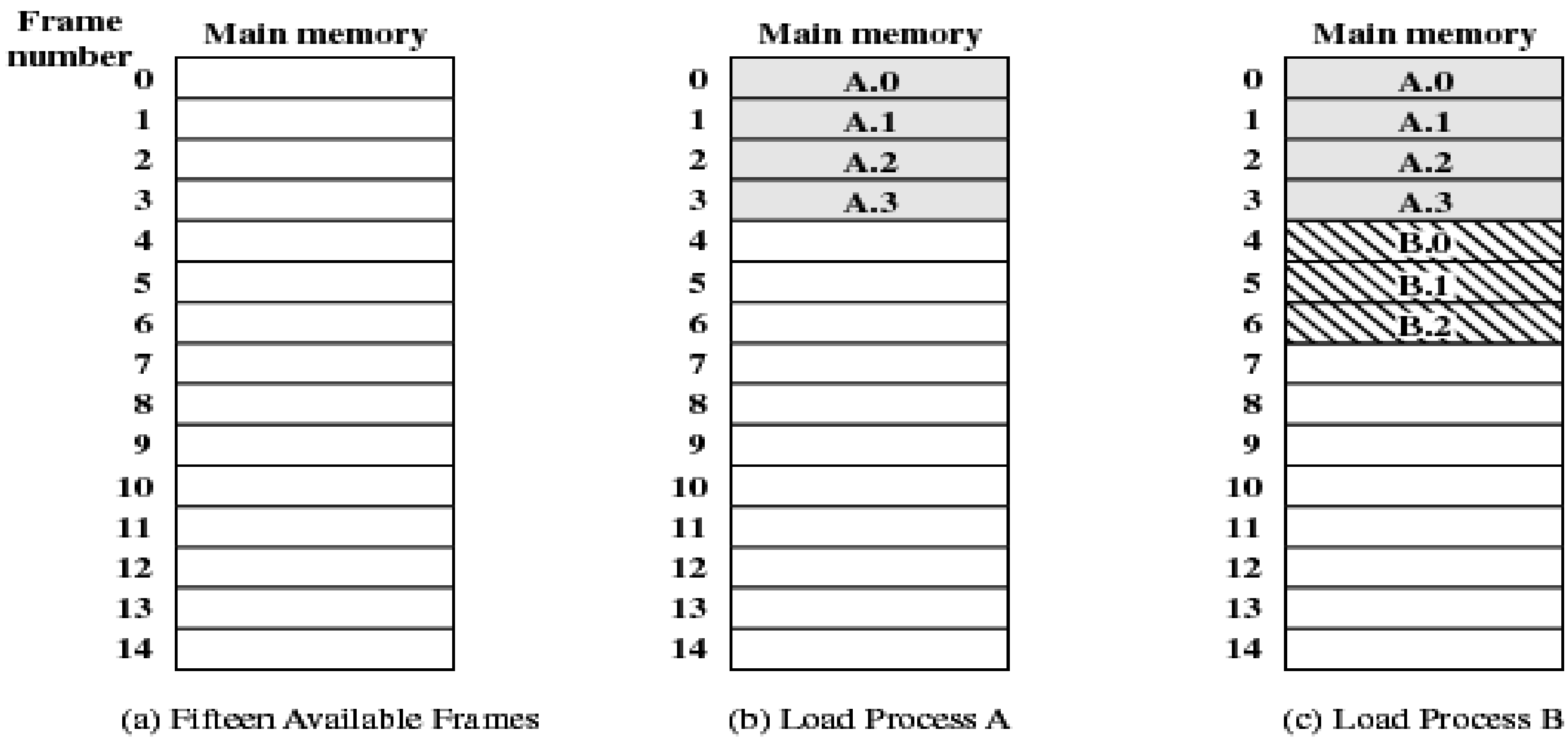
UPPSALA UNIVERSITET

Informationsteknologi

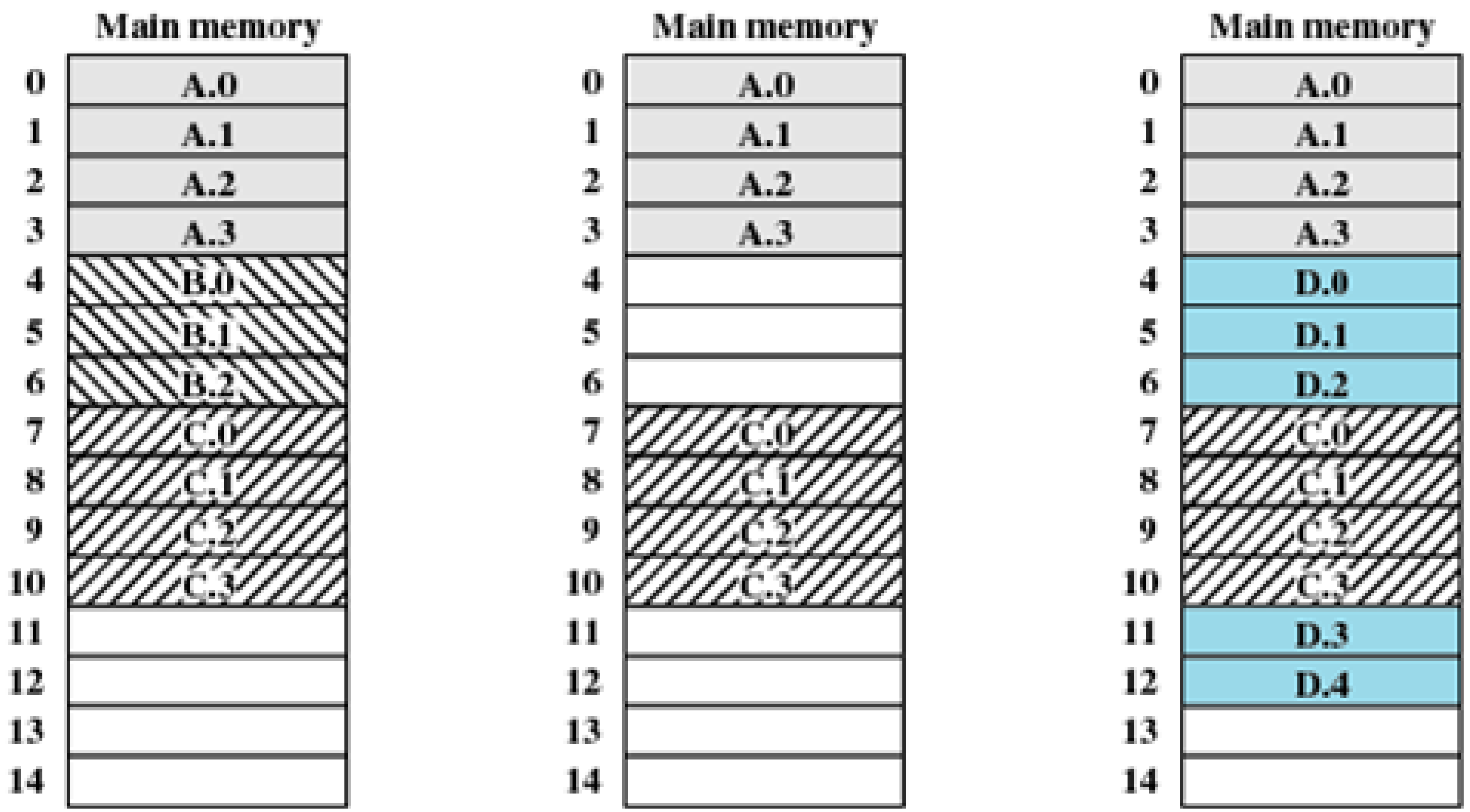
Paging

- 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 and chunks of memory are called frames
- 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

Assignment of Process Pages to Free Frames



Assignment of Process Pages to Free Frames



(d) Load Process C (e) Swap out B (f) Load Process D

UPPSALA UNIVERSITET

Informationsteknologi

Page Tables for the Previous Example

0	0
1	1
2	2
3	3

Process A page table

0	Ñ
1	Ñ
2	Ñ

Process B page table

0	7
1	8
2	9
3	10

Process C page table

0	4
1	5
2	6
3	11
4	12

Process D page table

13
14

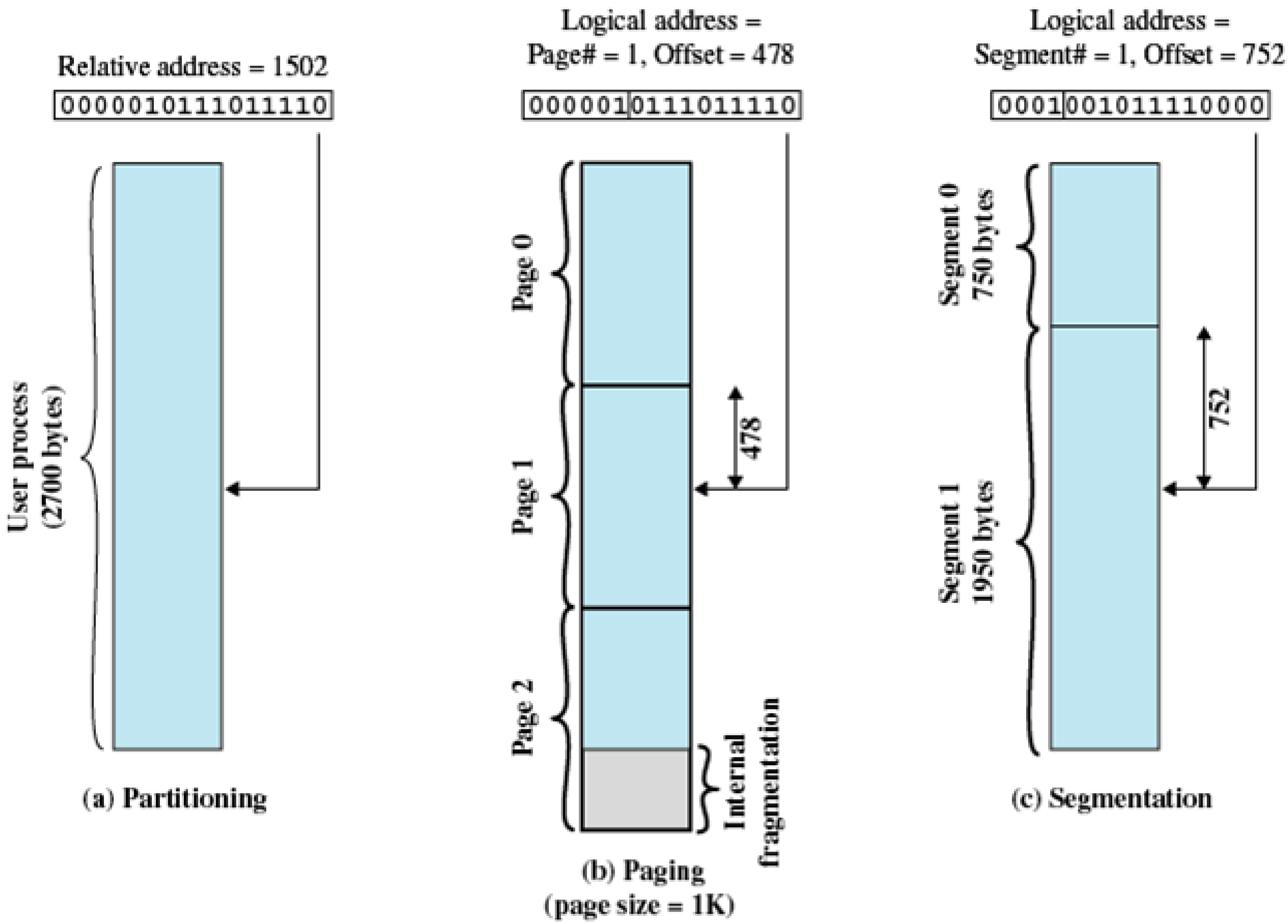
Free frame list

UPPSALA UNIVERSITET

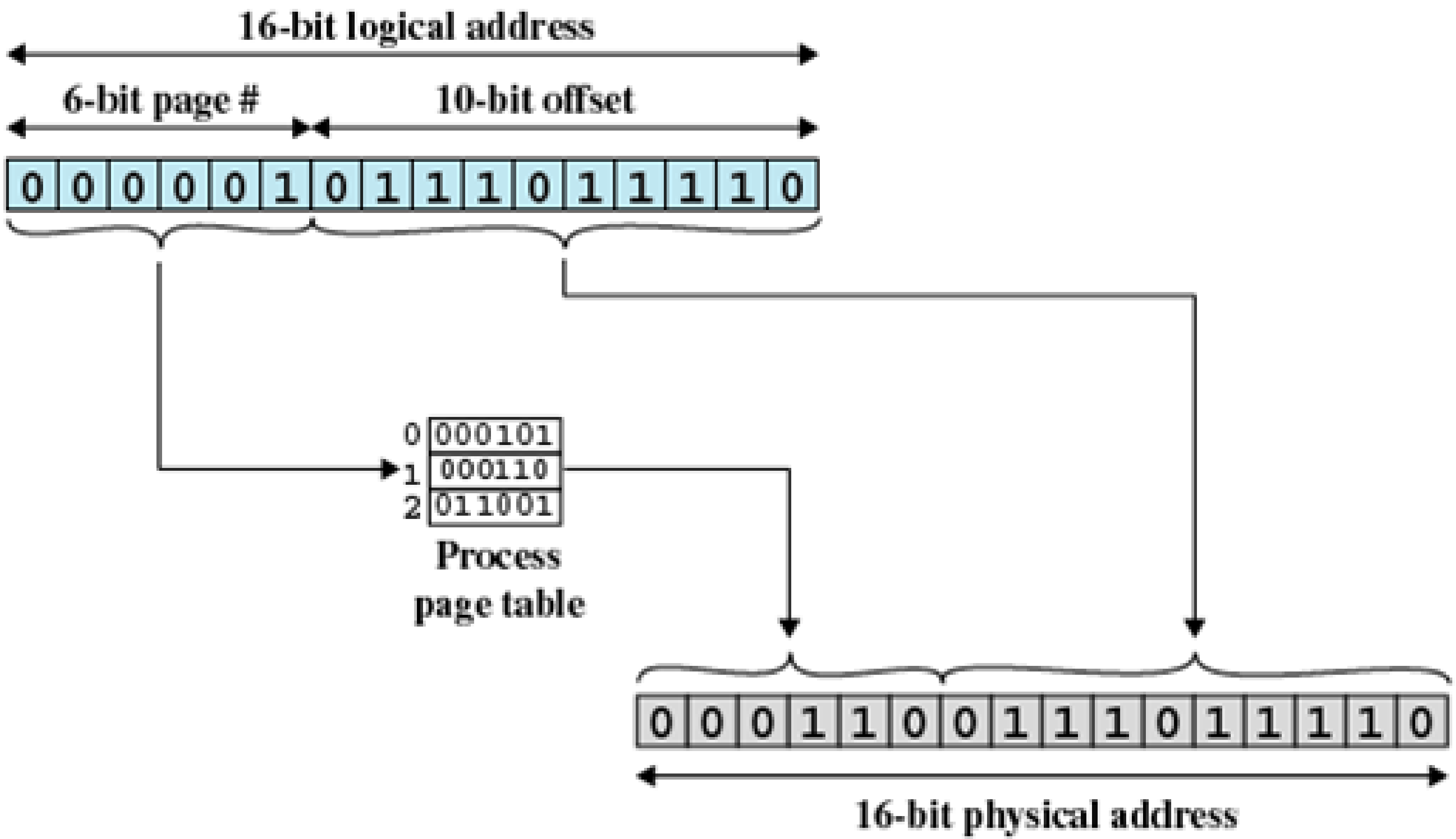
Informationsteknologi

Segmentation

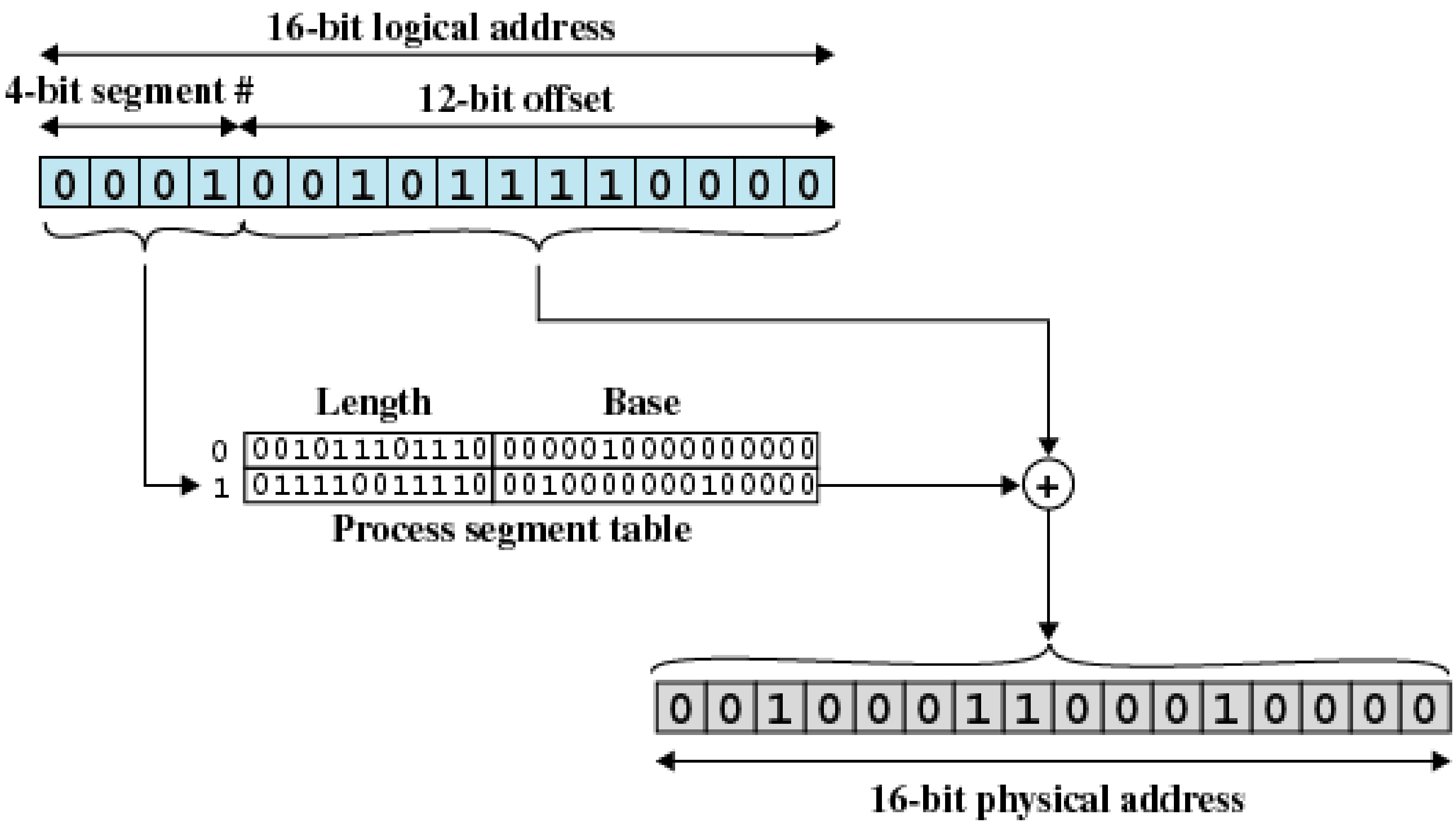
- All segments of all programs do not have to be of the same length
- There is a maximum segment length
- Addressing consist of two parts - a segment number and an offset
- Since segments are not equal, segmentation is similar to dynamic partitioning


Relative address = 1502
0000010111011110
User process (2700 bytes)
(a) Partitioning
Logical address =
Page# = 1, Offset = 478
000001 0111011110
Page 0
Page 1
Page 2
478
Internal fragmentation
(b) Paging
(page size = 1K)
Logical address =
Segment# = 1, Offset = 752
0001 001011110000
Segment 0 750 bytes
Segment 1 1950 bytes
752
(c) Segmentation

Logical-to-Physical Address Translation - Paging



Logical-to-Physical Address Translation - Segmentation



UPPSALA UNIVERSITET

Informationsteknologi

Virtual Memory

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 be located contiguously in main memory
- All pieces of a process do not need to be loaded in main memory during execution

UPPSALA UNIVERSITET

Informationsteknologi

Execution of a Program

- Operating system brings into main memory a few pieces of the program
- Resident set – the portion of the process that is in main memory
- An interrupt is generated when an address is needed that is not in main memory
- Operating system places the process in a blocking state

UPPSALA UNIVERSITET

Informationsteknologi

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

UPPSALA UNIVERSITET

Informationsteknologi

Thrashing

- Operating system must be clever about how it manages this process
- It may swap out a piece of a process just before that piece is needed
- If it does this a lot, the processor spends most of its time swapping pieces rather than executing user instructions
- This is known as ***thrashing***

UPPSALA UNIVERSITET

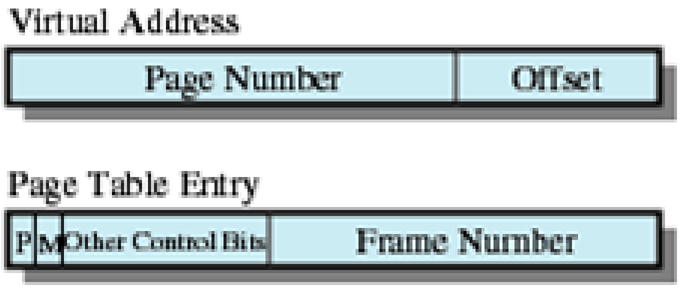
Informationsteknologi

Principle of Locality

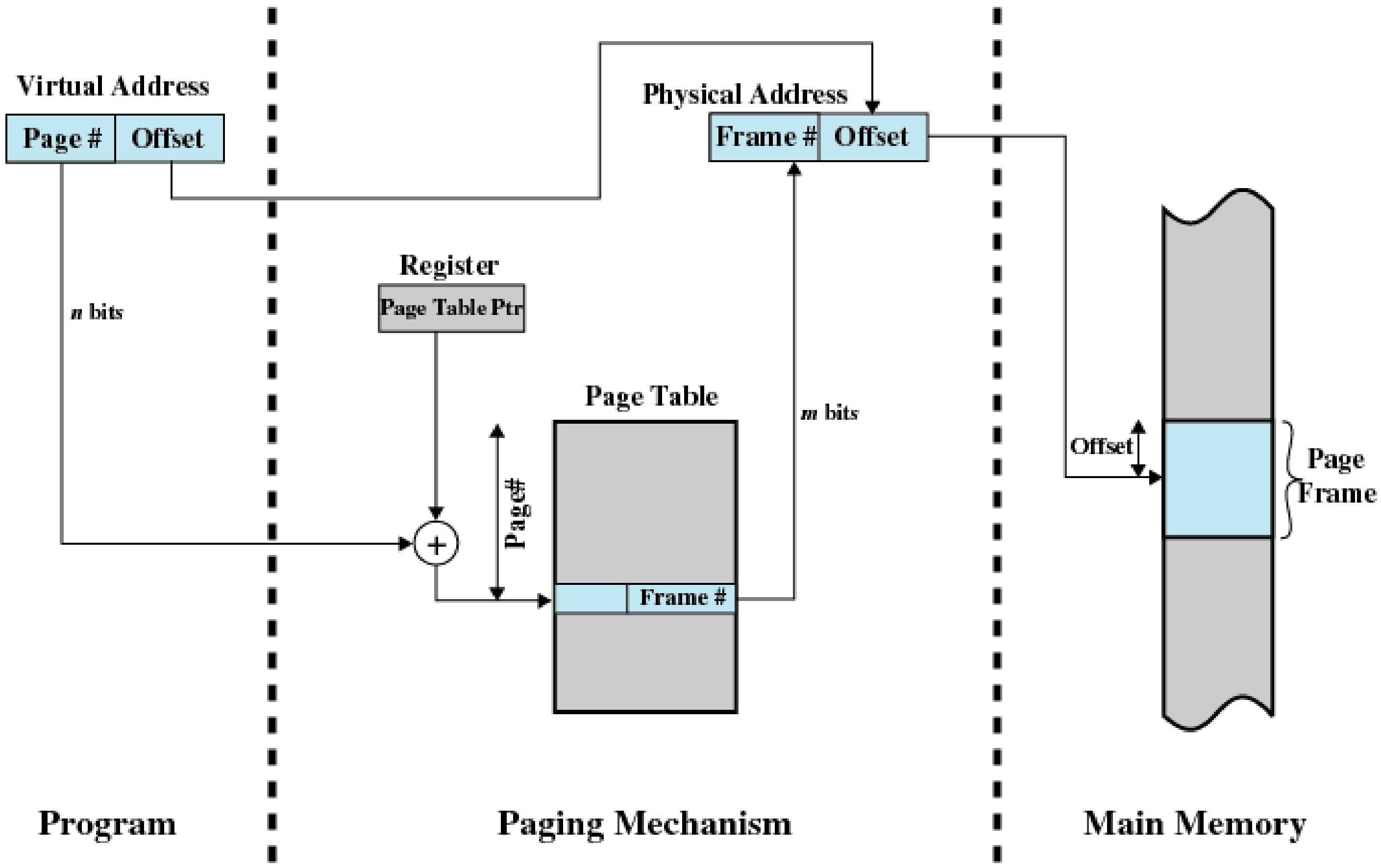
- Operating system tries to guess, based on recent history, which pieces in memory are least likely to be used in the near future
- Program and data references within a process tend to cluster
- Only a few pieces of a process will be needed over a short period of time
- This suggests that virtual memory may work efficiently

Paging

- Each process has its own page table
- Each page table entry contains the frame number of the corresponding page in main memory
- A bit is needed to indicate whether the page is in main memory or not



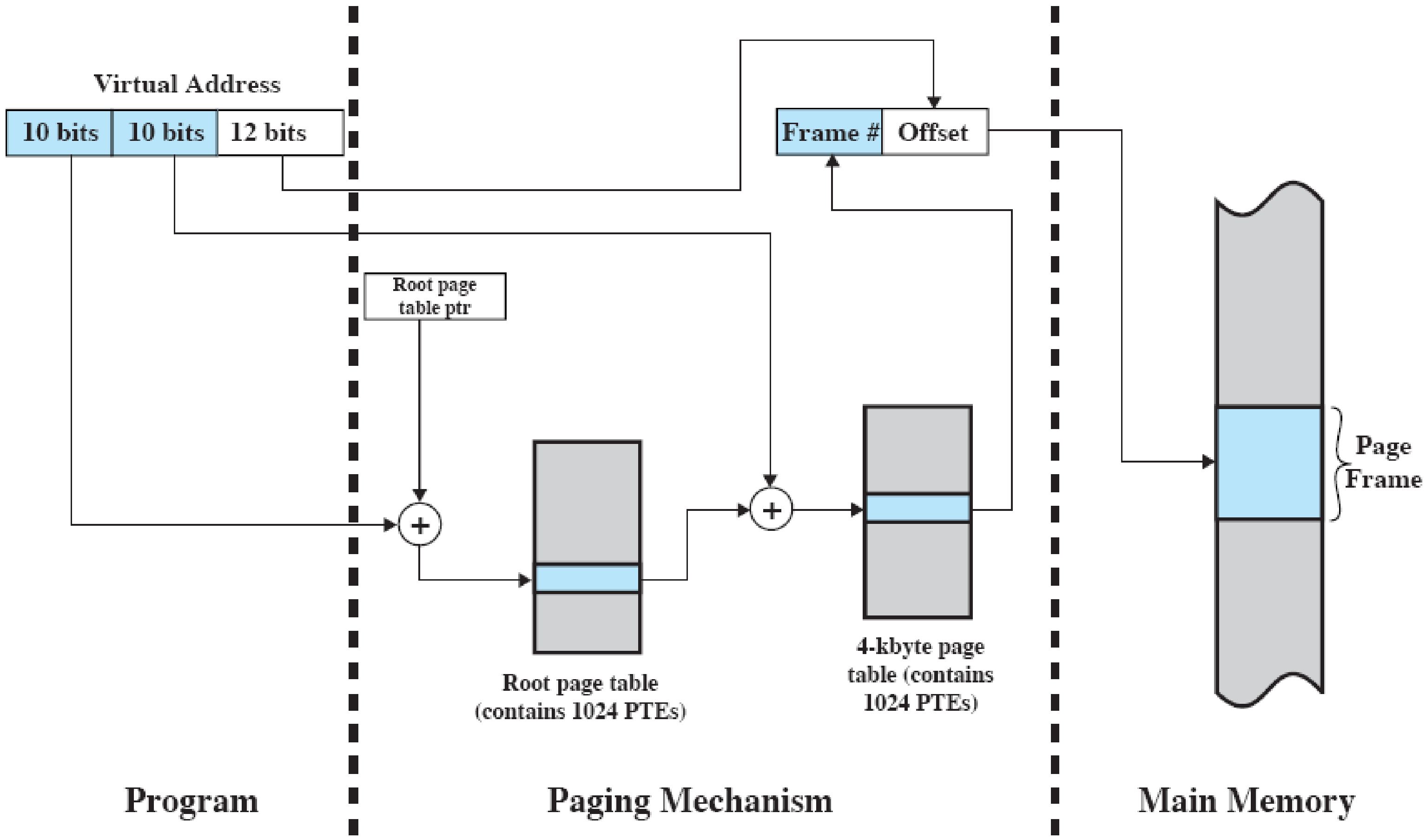
Address Translation in a Paging System



UPPSALA UNIVERSITET

Informationsteknologi

Address Translation in Two-Level Paging System



UPPSALA UNIVERSITET

Informationsteknologi

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

UPPSALA UNIVERSITET

Informationsteknologi

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
- 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

UPPSALA UNIVERSITET

Informationsteknologi

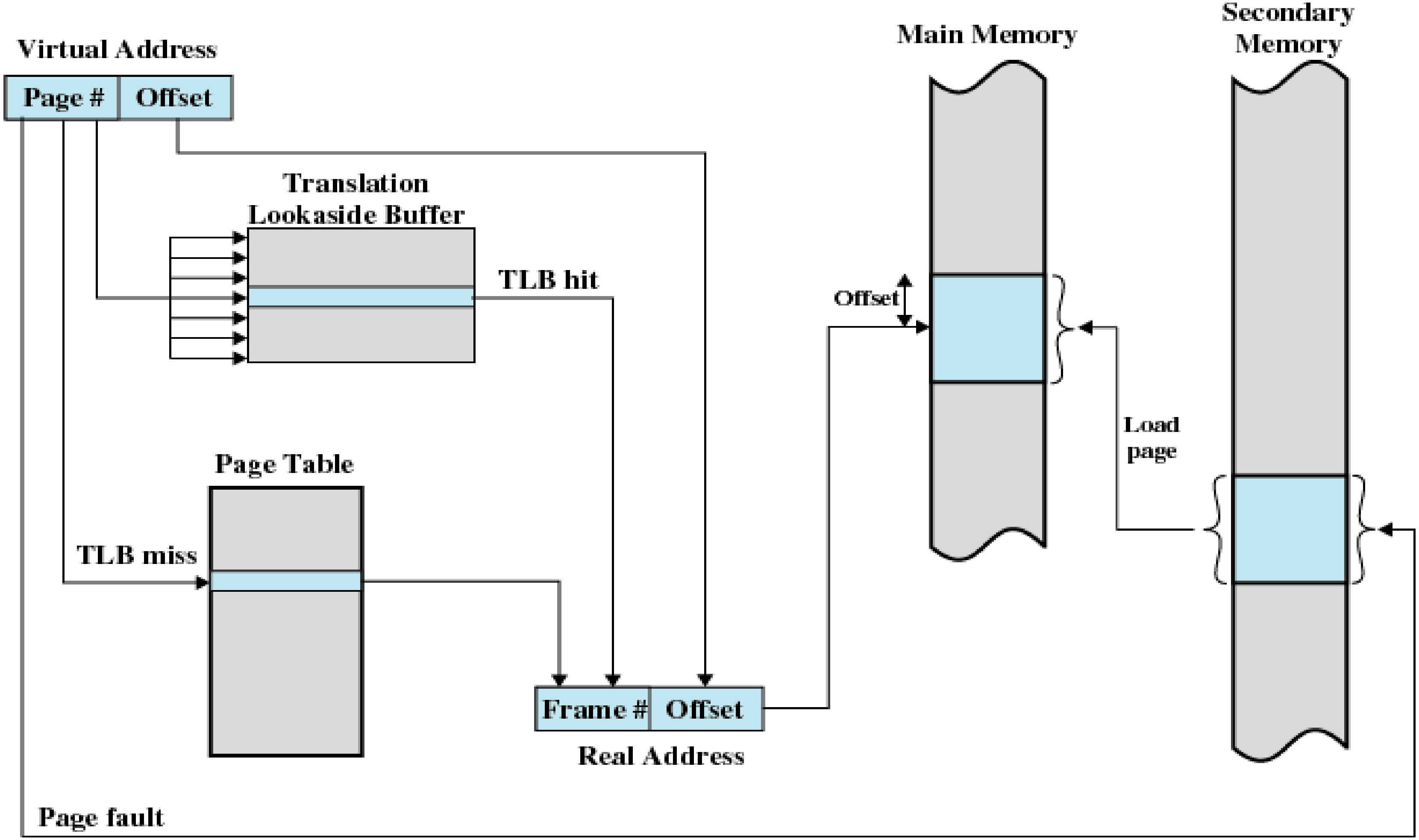


Figure 8.7 Use of a Translation Lookaside Buffer

UPPSALA UNIVERSITET

Informationsteknologi

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

UPPSALA UNIVERSITET

Informationsteknologi

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.

UPPSALA UNIVERSITET

Informationsteknologi

Segmentation

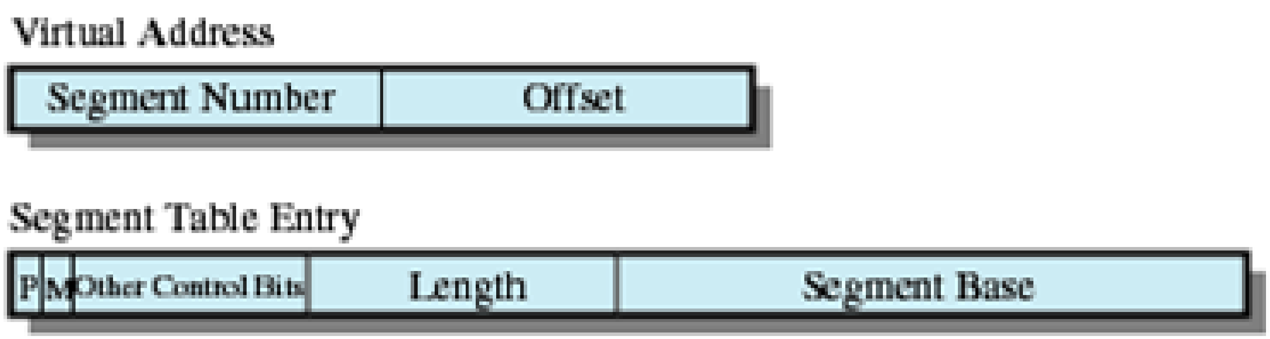
- May be unequal, dynamic size
- Simplifies handling of growing data structures
- Allows programs to be altered and recompiled independently
- Lends itself to sharing data among processes
- Lends itself to protection

Informationsteknologi

UPPSALA UNIVERSITET

Segment Tables

- 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



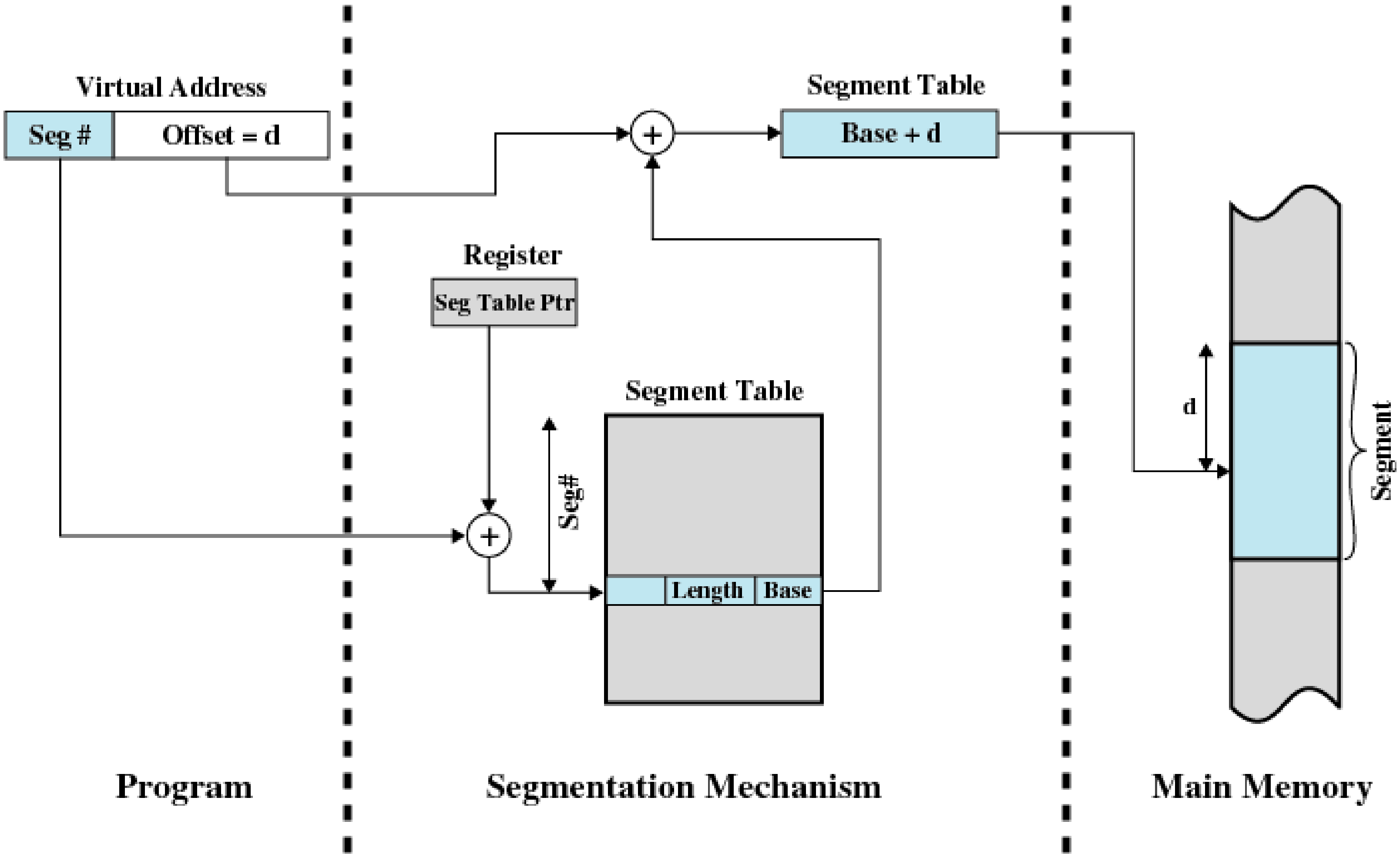
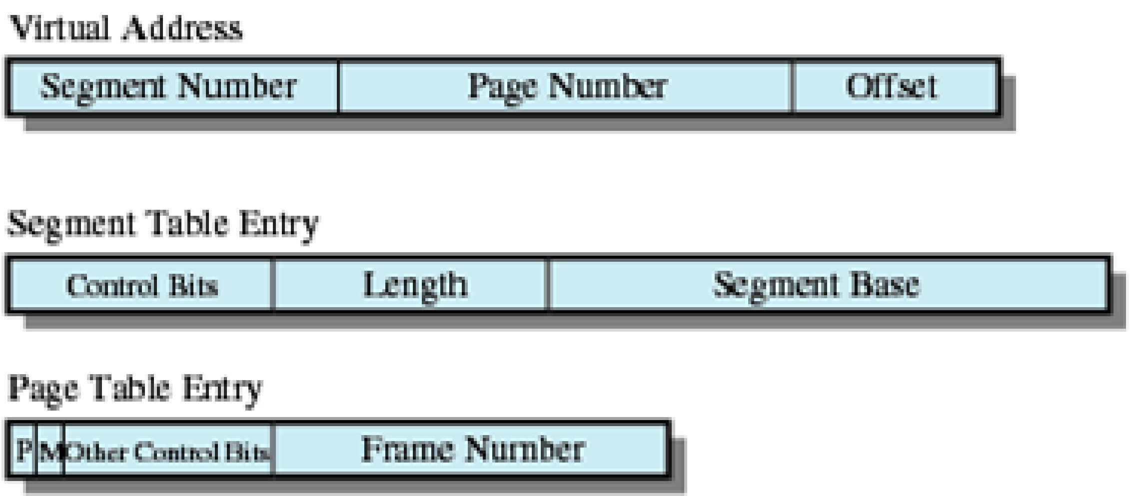


Figure 8.12 Address Translation in a Segmentation System

Combined Paging and Segmentation

- Paging is transparent to the programmer
- Segmentation is visible to the programmer
- Each segment is broken into fixed-size pages



UPPSALA UNIVERSITET

Informationsteknologi

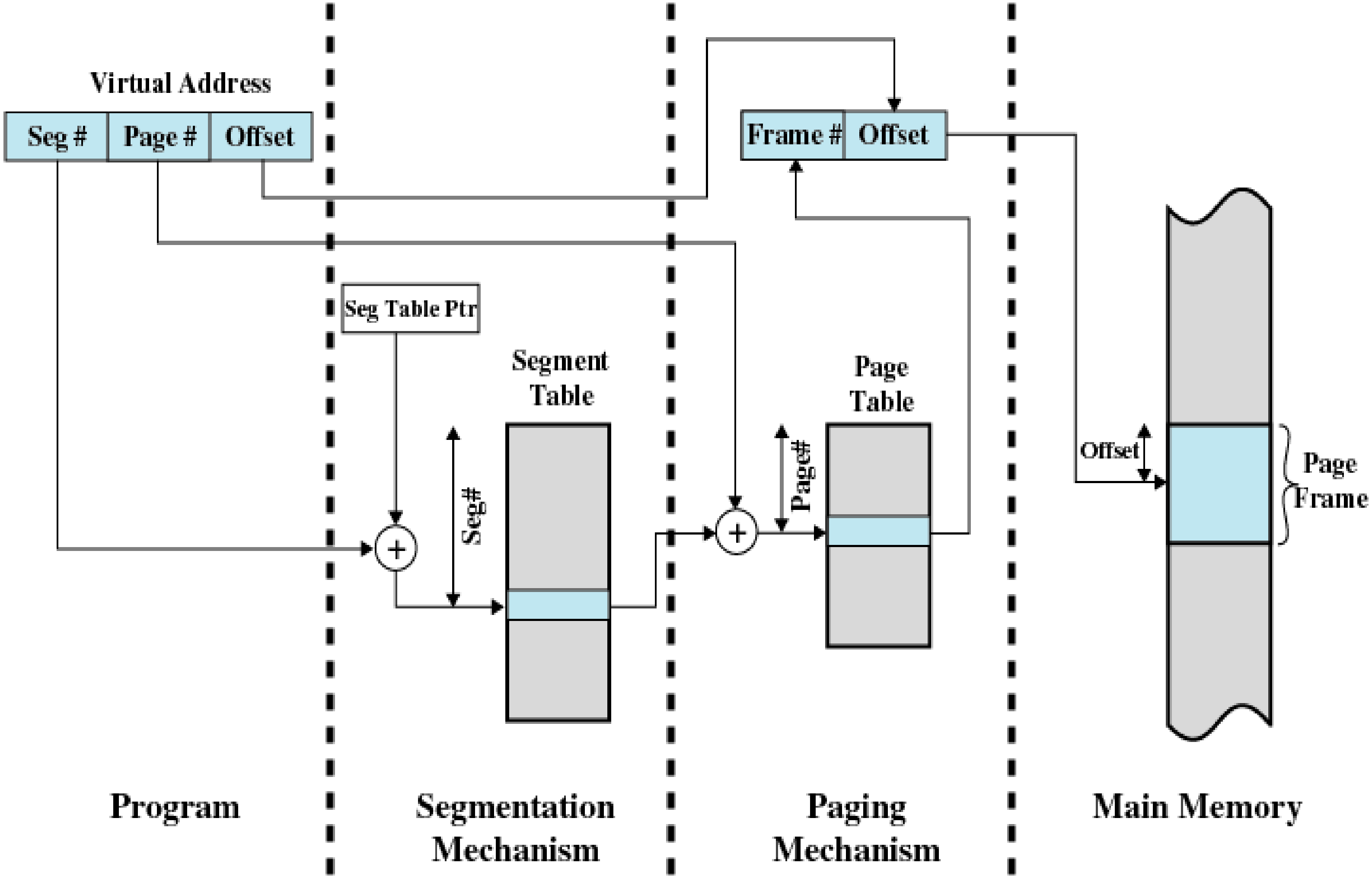


Figure 8.13 Address Translation in a Segmentation/Paging System