Operating Systems

Chapter 7 Memory Management(内存管理)

Agenda

- 7.1 Memory Management Requirements
- 7.2 Memory Partitioning
- 7.3 Paging
- 7.4 Segmentation
- 7.5 Summary

Memory Management

- Subdividing memory to accommodate multiple processes(为支持多道程序将内存进行划分)
- Memory needs to be allocated to ensure a reasonable supply of ready processes to consume available processor time(内存管理应确保有适当数目的就绪进程使用处理器时间)

7.1 Memory Management Requirements

- 7.1.1 Relocation
- 7.1.2 Protection
- 7.1.3 Sharing
- 7.1.4 Logical Organization
- 7.1.5 Physical Organization

Memory Management Requirements

- Relocation(重定位)
  - 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(交换) to disk and returned to main memory at a different location (relocated)
  - Memory references(访问) must be translated in the code to actual physical memory address(物理内存地址)

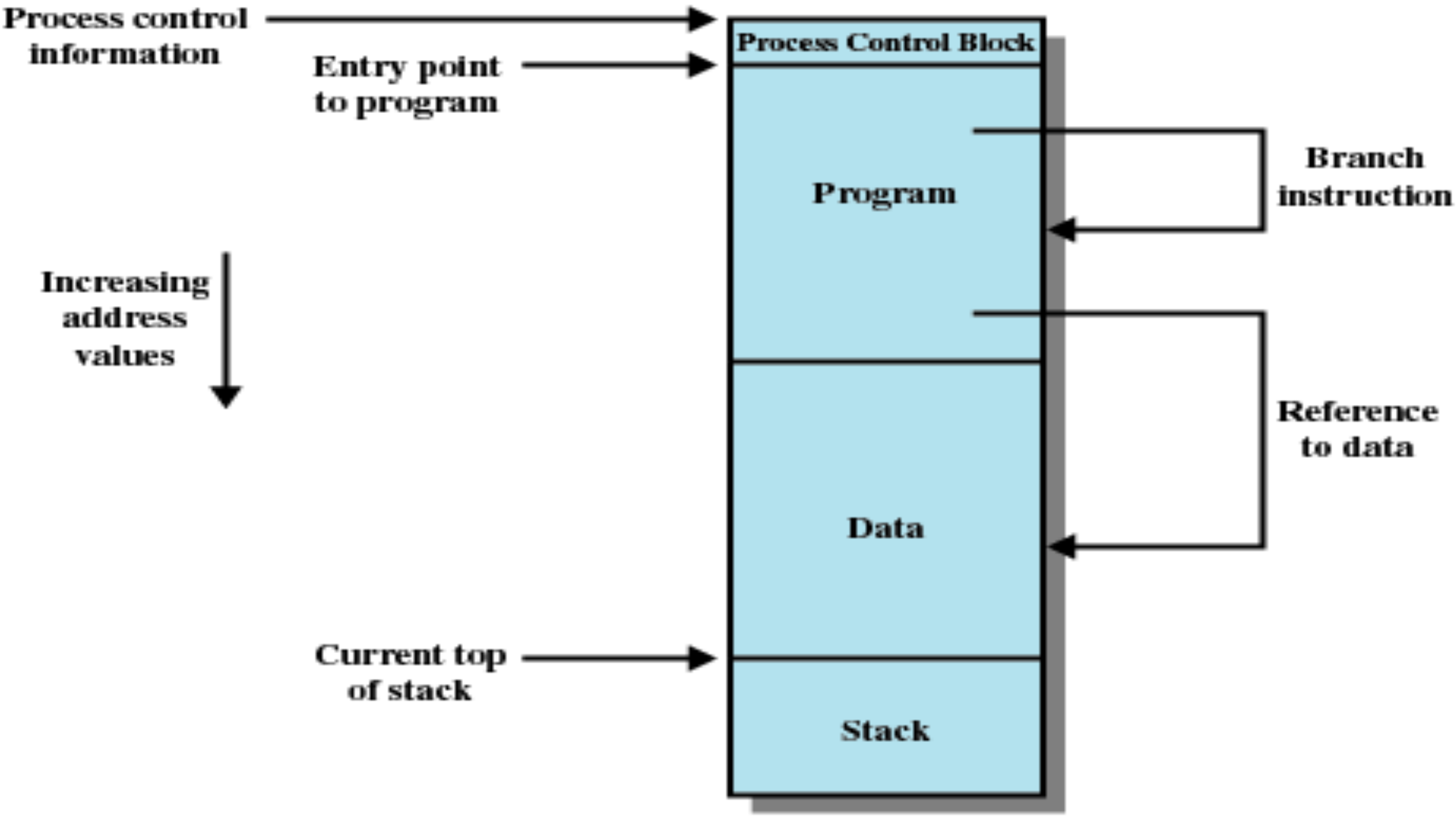


Figure 7.1 Addressing Requirements for a Process

7.1 Memory Management Requirements

- 7.1.1 Relocation
- 7.1.2 Protection
- 7.1.3 Sharing
- 7.1.4 Logical Organization
- 7.1.5 Physical Organization

Memory Management Requirements

- Protection(保护)
  - Processes should not be able to reference memory locations in another process without permission or jump to instructions area of another process (进程不能在未授权的情况下访问其他进程的数据，不能跳转到其他进程的代码区域执行指令)
  - Normally, processes cannot access any portion of the OS, neither program nor data
  - Impossible to check absolute addresses at compile time, instead, absolute addresses must be checked ***at rum time.***
  - Memory protection requirement must be satisfied by the processor (***hardware***) rather than the operating system

7.1 Memory Management Requirements

- 7.1.1 Relocation
- 7.1.2 Protection
- 7.1.3 Sharing
- 7.1.4 Logical Organization
- 7.1.5 Physical Organization

Memory Management Requirements

- Sharing(共享)
  - Allow several processes to access the same portion of memory
    - Share same copy of the program
    - Share data structure to cooperate on some task

7.1 Memory Management Requirements

- 7.1.1 Relocation
- 7.1.2 Protection
- 7.1.3 Sharing
- <u>7.1.4 Logical Organization</u>
- 7.1.5 Physical Organization

Memory Management Requirements

- Logical Organization(逻辑组织)
  - Main memory is organized in a linear address space, consisting of a sequence of bytes or words
  - 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
  - Segmentation satisfies these requirements

7.1 Memory Management Requirements

- 7.1.1 Relocation
- 7.1.2 Protection
- 7.1.3 Sharing
- 7.1.4 Logical Organization
- 7.1.5 Physical Organization

Memory Management Requirements

- Physical Organization(物理组织)
  - Memory is organized into at least two levels, referred to as main memory and secondary memory
  - Memory available for a program plus its data may be insufficient(内存对程序和其数据来说可能不足)
    - Overlaying(覆盖) allows various modules to be assigned the same region of memory
  - Programmer does not know how much space will be available and where his/her program will be loaded in memory

Agenda

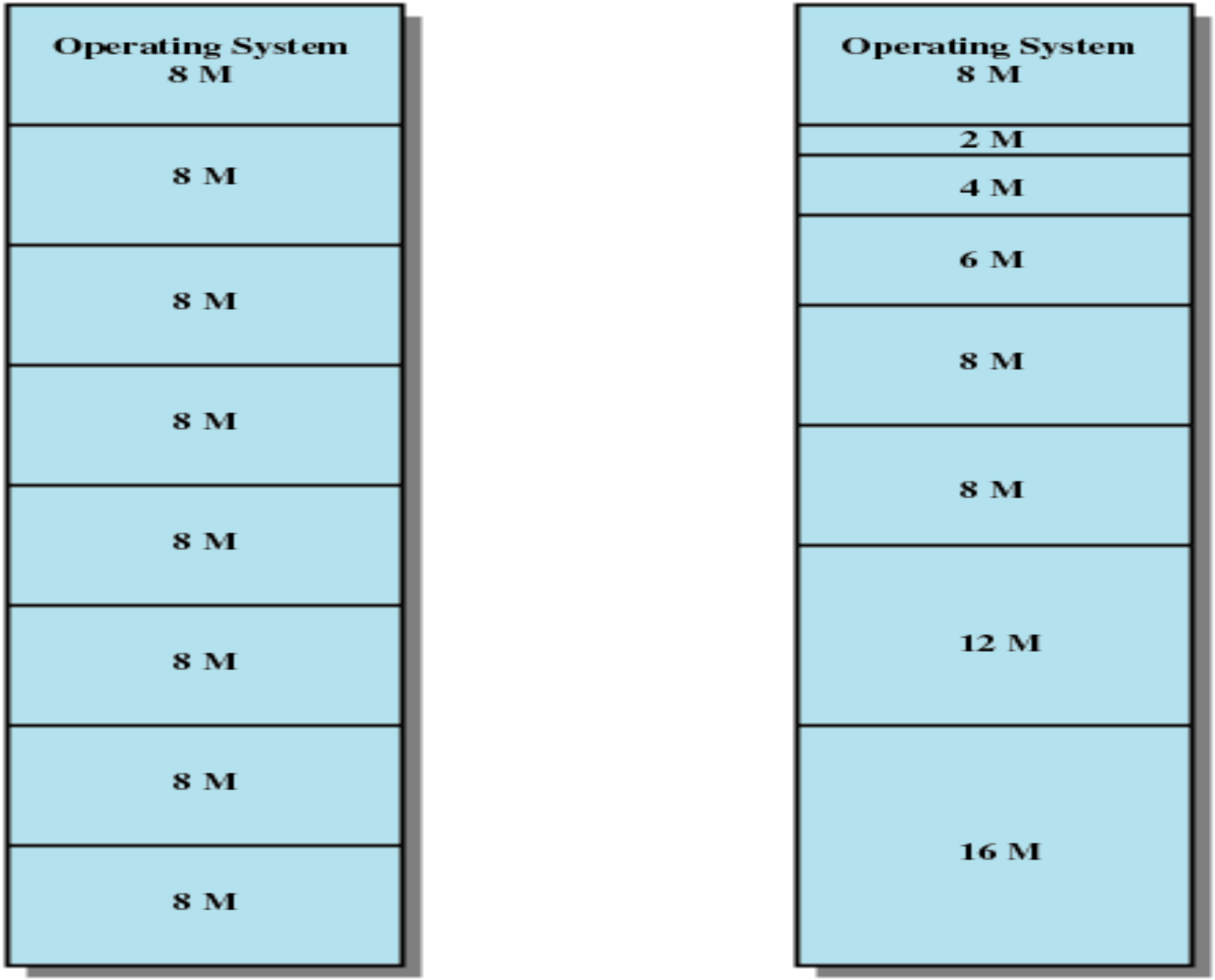
- 7.1 Memory Management Requirements
- <u>7.2 Memory Partitioning</u>
- 7.3 Paging
- 7.4 Segmentation
- 7.5 Summary

7.2 Memory Partitioning(内存分区)

- 7.2.1 Fixed Partitioning
- 7.2.2 Dynamic Partitioning
- 7.2.3 Buddy System
- 7.2.4 Relocation

Fixed Partitioning(固定分区)

- Alternatives for Fixed Partitioning
  - Equal-size partitions(大小相等的分区)
  - Unequal-size partitions(大小不等的分区)



(a) Equal-size partitions

(b) Unequal-size partitions

Figure 7.2 Example of Fixed Partitioning of a 64-Mbyte Memory

Fixed Partitioning

- Equal-size partitions: features
  - 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

Fixed Partitioning

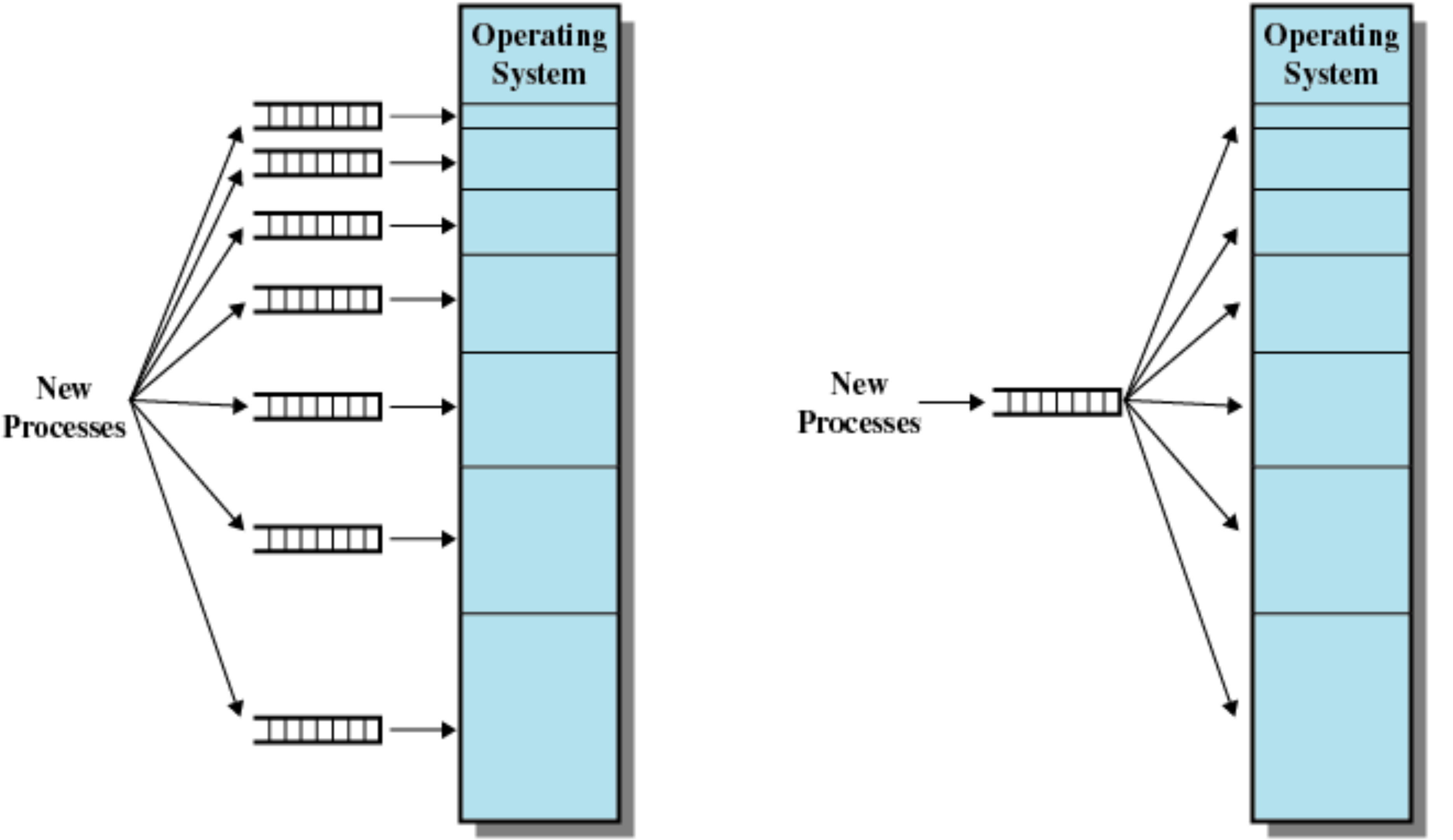
- Equal-size partitions: difficulties
  - A program may not fit in a partition. The programmer must design the program with **overlays**
  - Main memory use is inefficient. Any program, no matter how small, occupies an entire partition.
    - **internal fragmentation(**内部碎片/内零头**)**.

Fixed Partitioning

- Unequal-size partitions(大小不等的分区)
  - Both of these problems of equal-size partitions can be lessened, though not solved, by using unequal-size partitions.

Placement Algorithm(放置算法) with Fixed 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



(a) One process queue per partition

(b) Single queue

Figure 7.3 Memory Assignment for Fixed Partitioning

Disadvantages of Fixed Partitions

- The number of active processes in the system is limited by the number of partitions
- Small jobs will not utilize partition efficiently

7.2 Memory Partitioning

- 7.2.1 Fixed Partitioning
- 7.2.2 Dynamic Partitioning
- 7.2.3 Buddy System
- 7.2.4 Relocation

Dynamic Partitioning(动态分区)

- 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(外部碎片/外零头)**
- Must use compaction(压缩) to shift(移动) processes so they are contiguous and all free memory is in one block

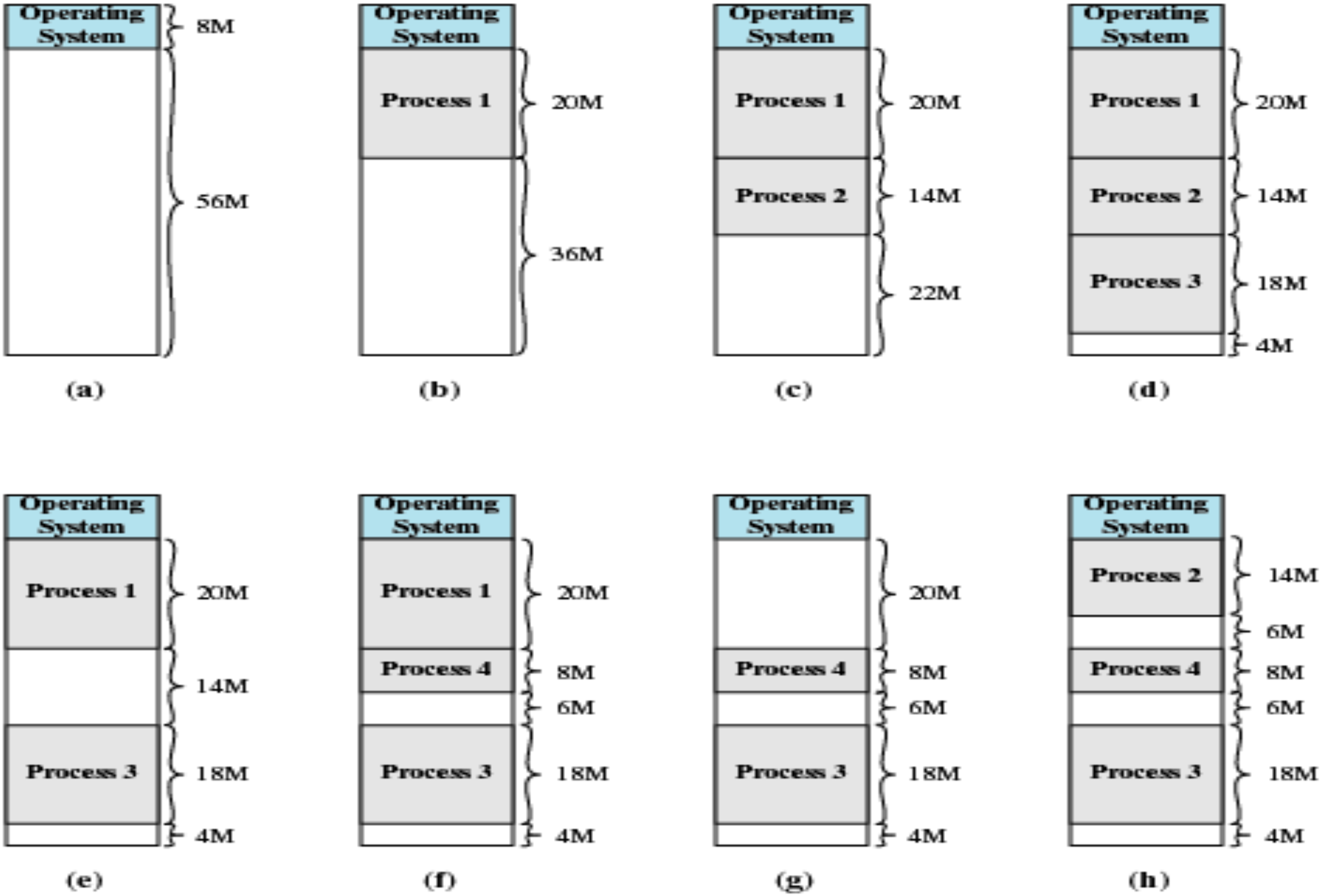


Figure 7.4 The Effect of Dynamic Partitioning

Dynamic Partitioning Placement Algorithm

- Three placement algorithms：

1. Best-fit algorithm(最佳适配)
   - 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

2. First-fit algorithm(首次适配)
   - Scans memory form the beginning and chooses the first available block that is large enough
   - Simplest and usually fastest and best
   - 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 Placement Algorithm

- Next-fit(邻近适配)
  - Scans memory from the location of the last placement and chooses the next available block that is large enough
  - 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

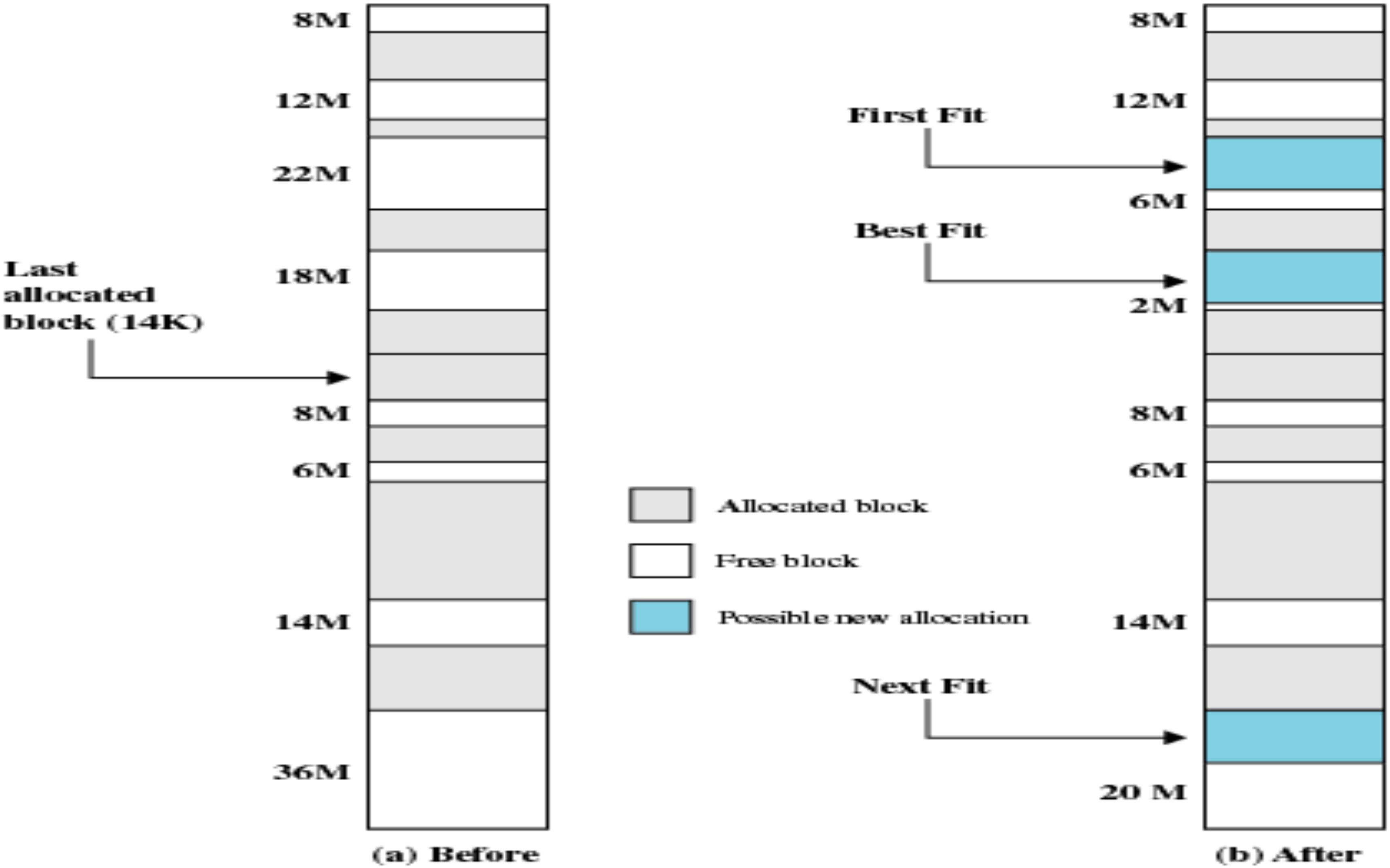


Figure 7.5 Example Memory Configuration Before and After Allocation of 16 Mbyte Block

7.2 Memory Partitioning

- 7.2.1 Fixed Partitioning
- 7.2.2 Dynamic Partitioning
- 7.2.3 Buddy(伙伴) System
- 7.2.4 Relocation

Buddy System

- Entire space available is treated as a single block of 2^U (e.g. 1M = 2^{20})
- 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

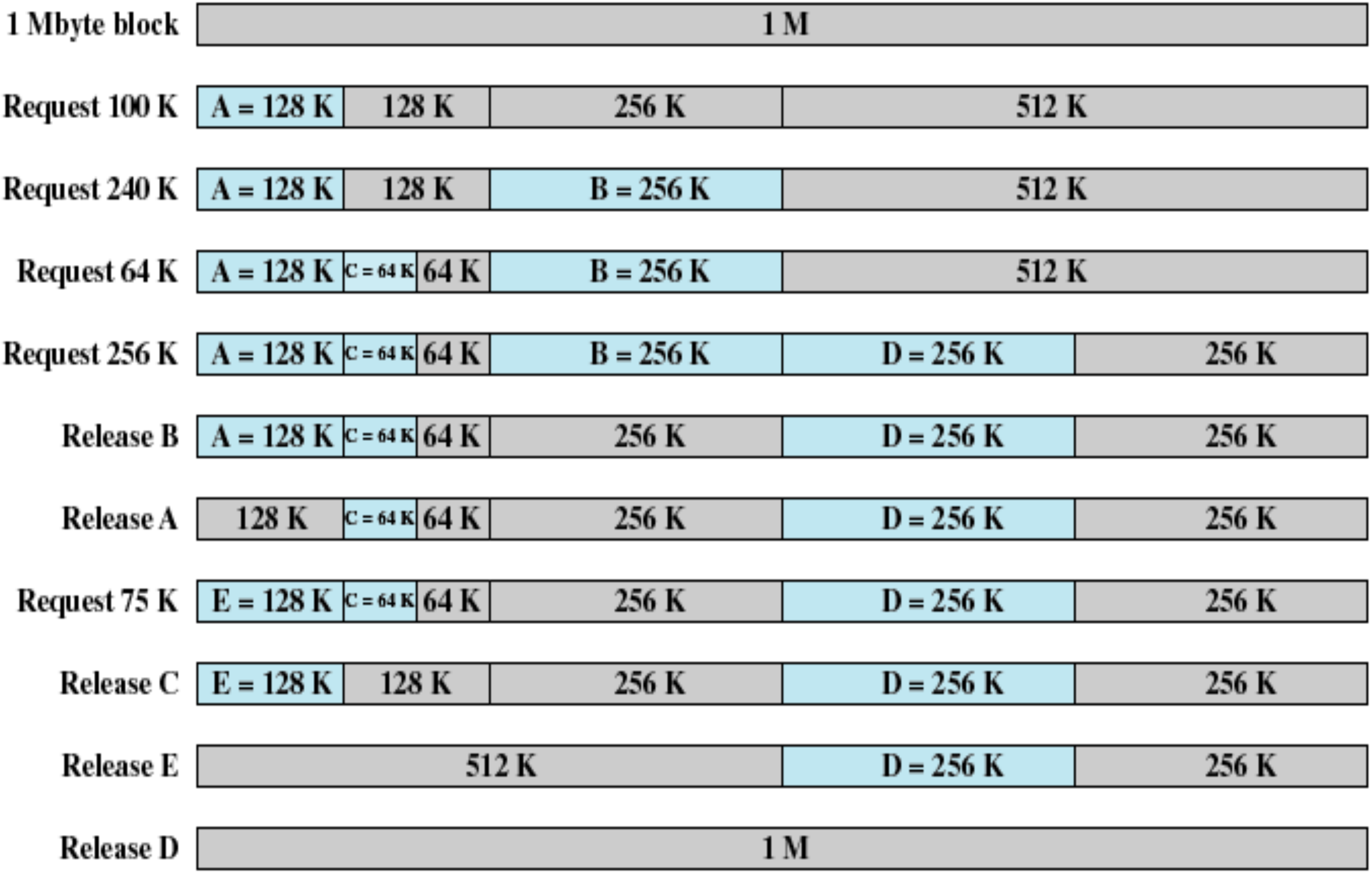


Figure 7.6 Example of Buddy System

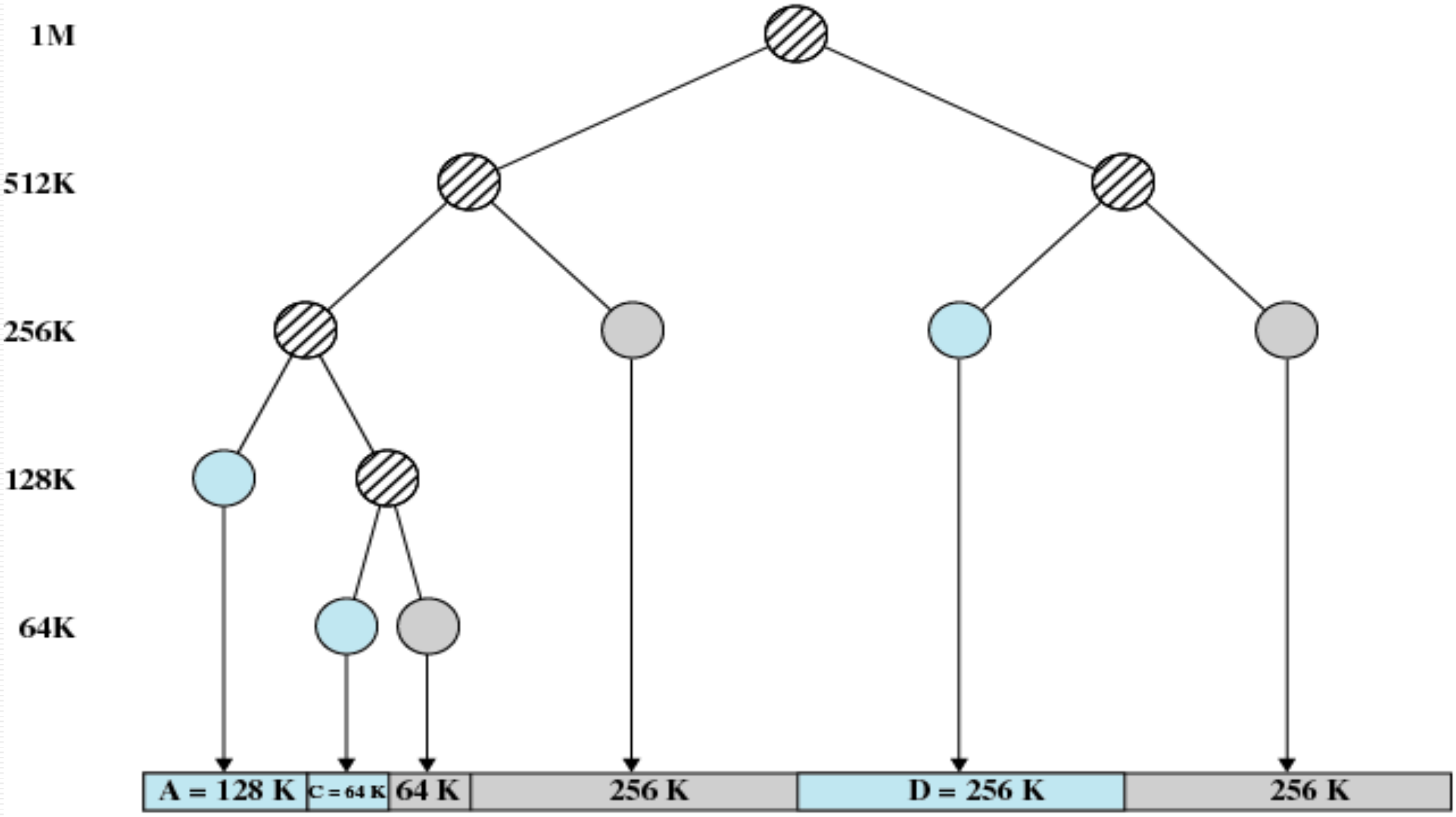


Figure 7.7 Tree Representation of Buddy System

7.2 Memory Partitioning

- 7.2.1 Fixed Partitioning
- 7.2.2 Dynamic Partitioning
- 7.2.3 Buddy System
- 7.2.4 Relocation

Relocation(重定位)

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

Addresses

- Logical Address(逻辑地址)
  - Reference to a memory location independent of the current assignment of data to memory(与当前数据在内存中的分配无关的访问地址)
  - Translation must be made to the physical address
- Relative Address(相对地址)
  - Address expressed as a location relative to some known point
- Physical Address(物理地址)
  - The absolute address(绝对地址) or actual location in main memory

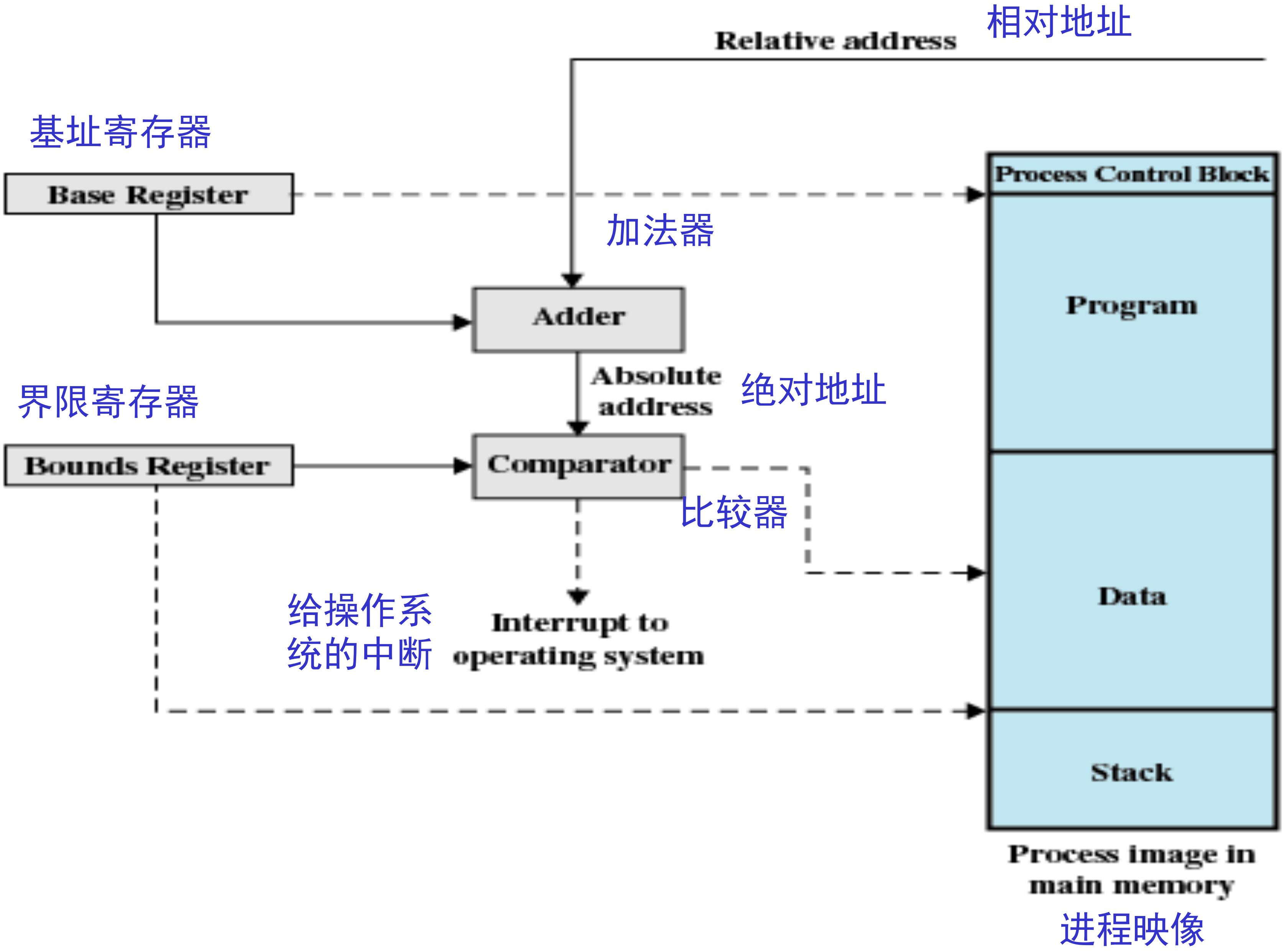


Figure 7.8 Hardware Support for Relocation

Registers Used during Execution

- Base register(基址寄存器)
  - Starting address for the process
- Bounds register(界限寄存器)
  - Ending location of the process
- These values are set when the process is loaded(加载) or when the process is swapped in(换入)

Registers Used during Execution

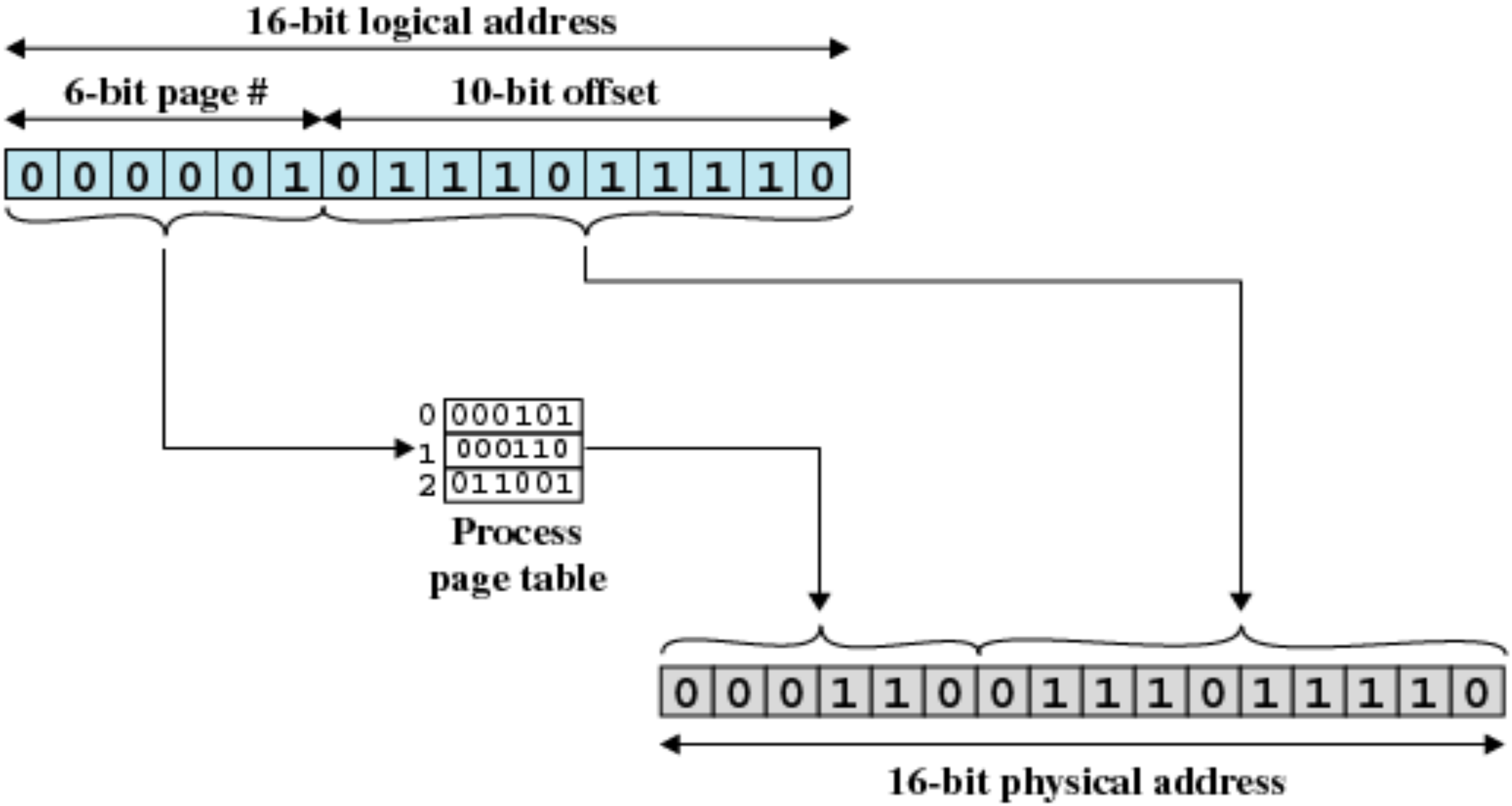
- The value of the base register is added to a relative address to produce an absolute address
- The resulting address is compared with the value in the bounds register
- If the address is not within bounds, an interrupt is generated to the operating system

Agenda

- 7.1 Memory Management Requirements
- 7.2 Memory Partitioning
- <u>7.3 Paging</u>
- 7.4 Segmentation
- 7.5 Summary

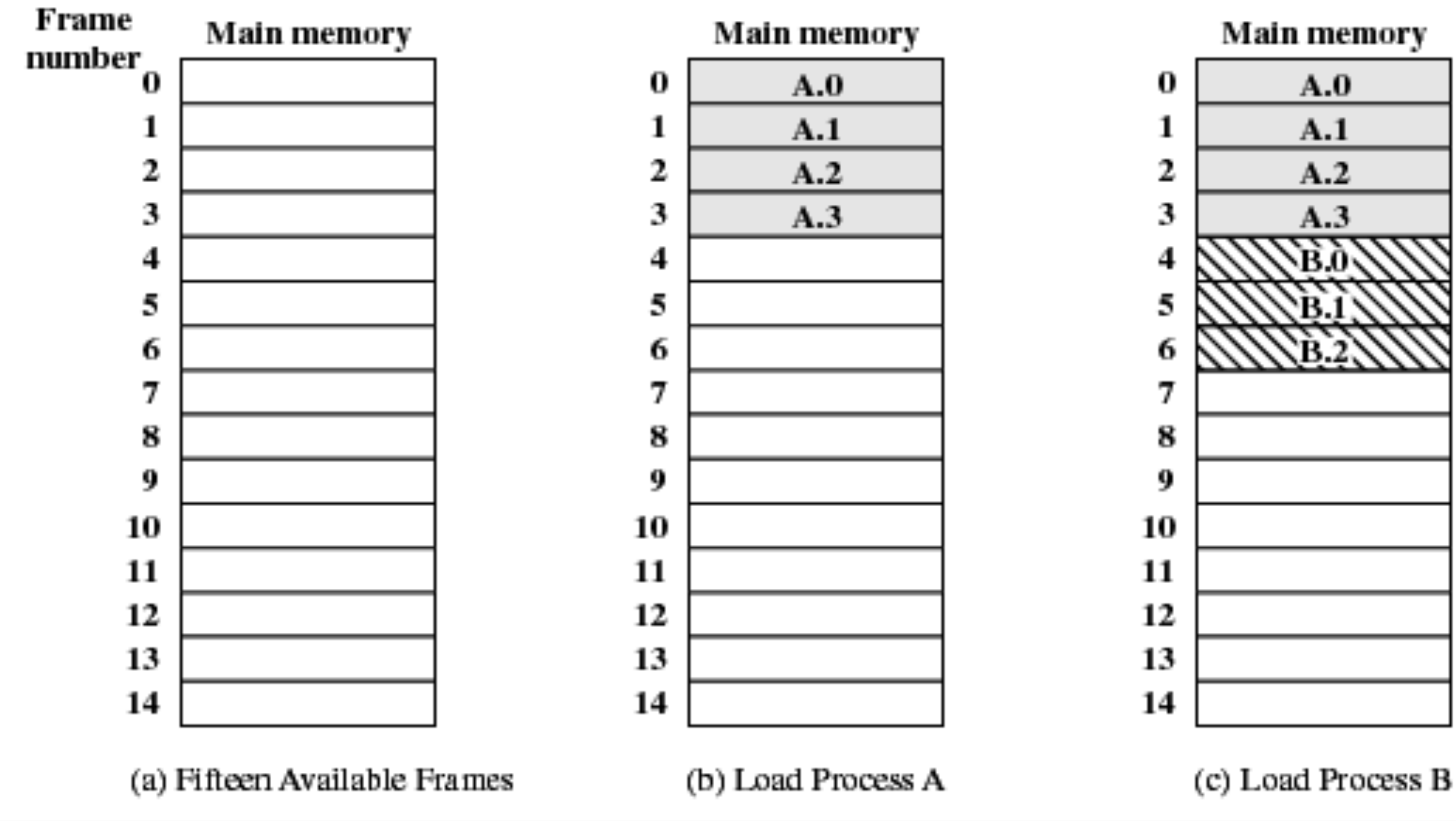
Paging(分页)

- Partition memory into small equal fixed-size chunks(块) which are called frames(帧)
- Divide each process into small equal fixed-size chunks which are called pages(页). The size of pages are equal to the size of frames
- Operating system maintains a page table(页表) for each process
  - Contains the frame location(帧位置) for each page in the process
  - Memory address consist of a page number(页号) and offset(偏移量) within the page


16-bit logical address
6-bit page #
10-bit offset
0 0 0 0 0 1 0 1 1 1 0 1 1 1 1 0
0 000101
1 000110
2 011001
Process
page table
0 0 0 1 1 0 0 1 1 1 0 1 1 1 1 0
16-bit physical address

(a) Paging

Assignment of Process Pages to Free Frames

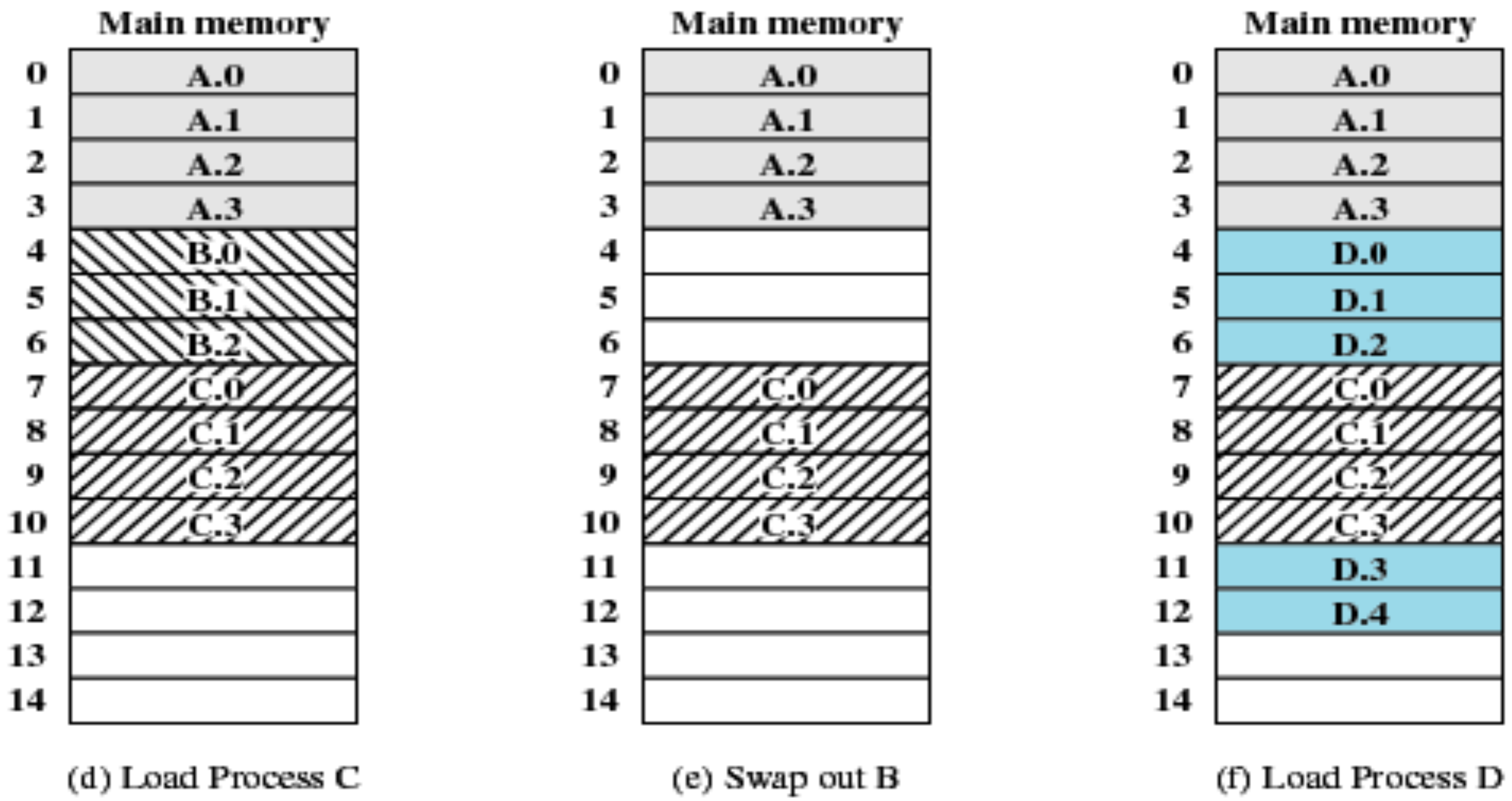


(a) Fifteen Available Frames

(b) Load Process A

(c) Load Process B

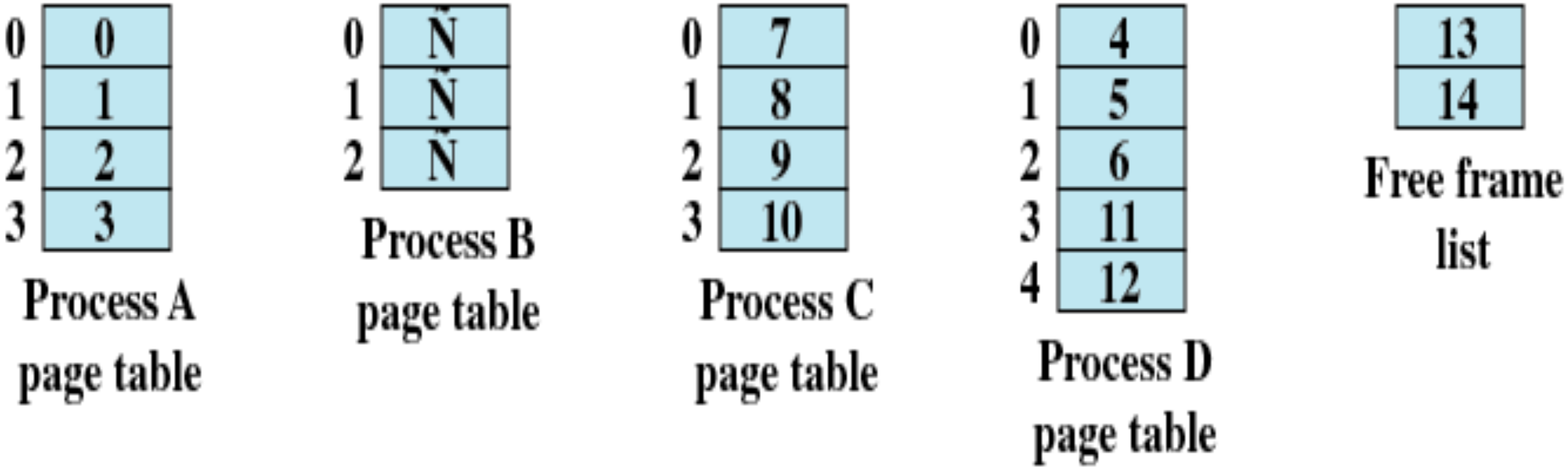
Assignment of Process Pages to Free Frames



	Main memory	Main memory	Main memory
0	A.0	A.0	A.0
1	A.1	A.1	A.1
2	A.2	A.2	A.2
3	A.3	A.3	A.3
4	B.0		D.0
5	B.1		D.1
6	B.2		D.2
7	C.0	C.0	C.0
8	C.1	C.1	C.1
9	C.2	C.2	C.2
10	C.3	C.3	C.3
11			D.3
12			D.4
13			
14			
	(d) Load Process C	(e) Swap out B	(f) Load Process D

Figure 7.9 Assignment of Process Pages to Free Frames

Page Tables for 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

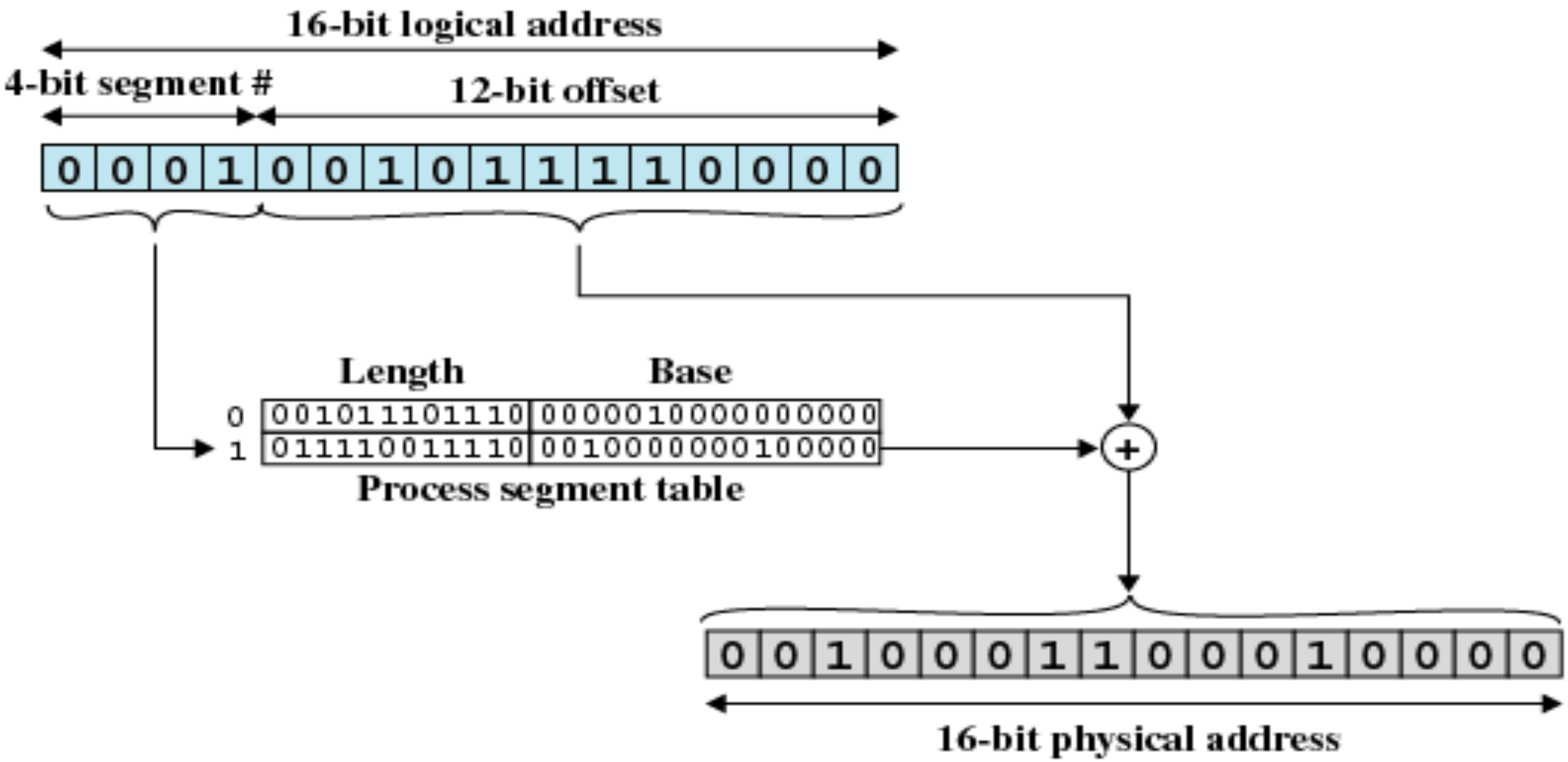
Figure 7.10 Data Structures for the Example of Figure 7.9 at Time Epoch (f)

Agenda

- 7.1 Memory Management Requirements
- 7.2 Memory Partitioning
- 7.3 Paging
- <u>7.4 Segmentation</u>
- 7.5 Summary

Segmentation(分段)

- Program and its data can be divided into a number of ***segments***, 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
  - segment number(段号)
  - offset(偏移量)
- Since segments are not equal, segmentation is similar to dynamic partitioning(动态分区)



16-bit logical address

4-bit segment #

12-bit offset

0 0 0 1 0 0 1 0 1 1 1 1 0 0 0 0

	Length	Base
0	001011101110	0000010000000000
1	011110011110	0010000000100000

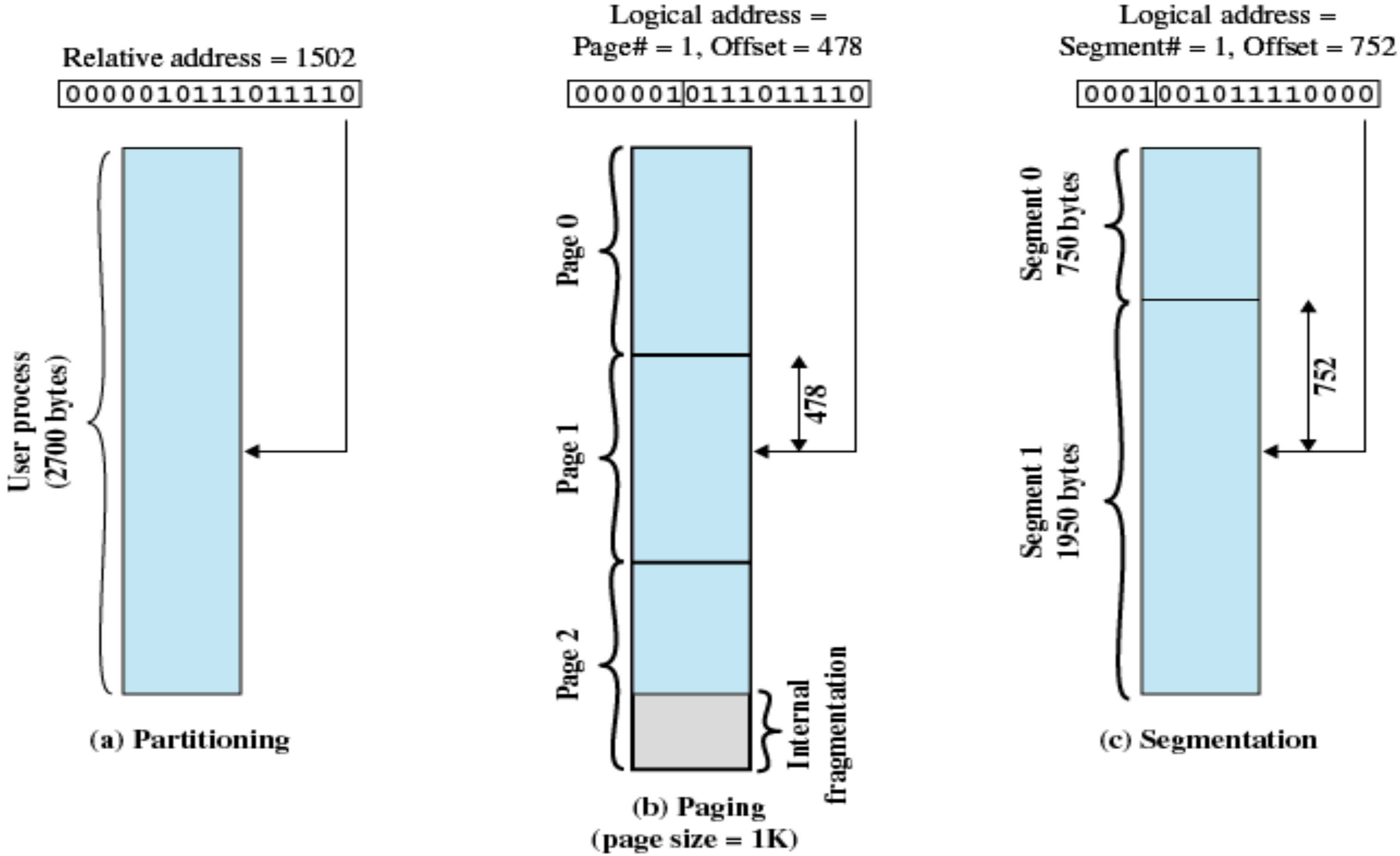
Process segment table

0 0 1 0 0 0 1 1 0 0 0 1 0 0 0 0

16-bit physical address

(b) Segmentation

Figure 7.12 Examples of Logical-to-Physical Address Translation



(a) Partitioning

(b) Paging (page size = 1K)

(c) Segmentation

Figure 7.11 Logical Addresses

Agenda

- 7.1 Memory Management Requirements
- 7.2 Memory Partitioning
- 7.3 Paging
- 7.4 Segmentation
- 7.5 Summary