第28讲 存储划分技术：分区技术

Memory Management Techniques

- Fixed Partitioning(固定分区)
- Dynamic Partitioning(动态分区)
- Simple Paging(简单分页)
- Simple Segmentation(简单分段)
- Virtual-Memory Paging(虚拟存储分页)
- Virtual-Memory Segmentation(虚拟存储分段)

§3.2 Simple Memory Management Techniques

Partitioning Technique

- Fixed Partitioning Technique(固定分区技术)
- Dynamic Partitioning Technique(动态分区技术)
- Buddy System(伙伴系统)

Fixed Partitioning

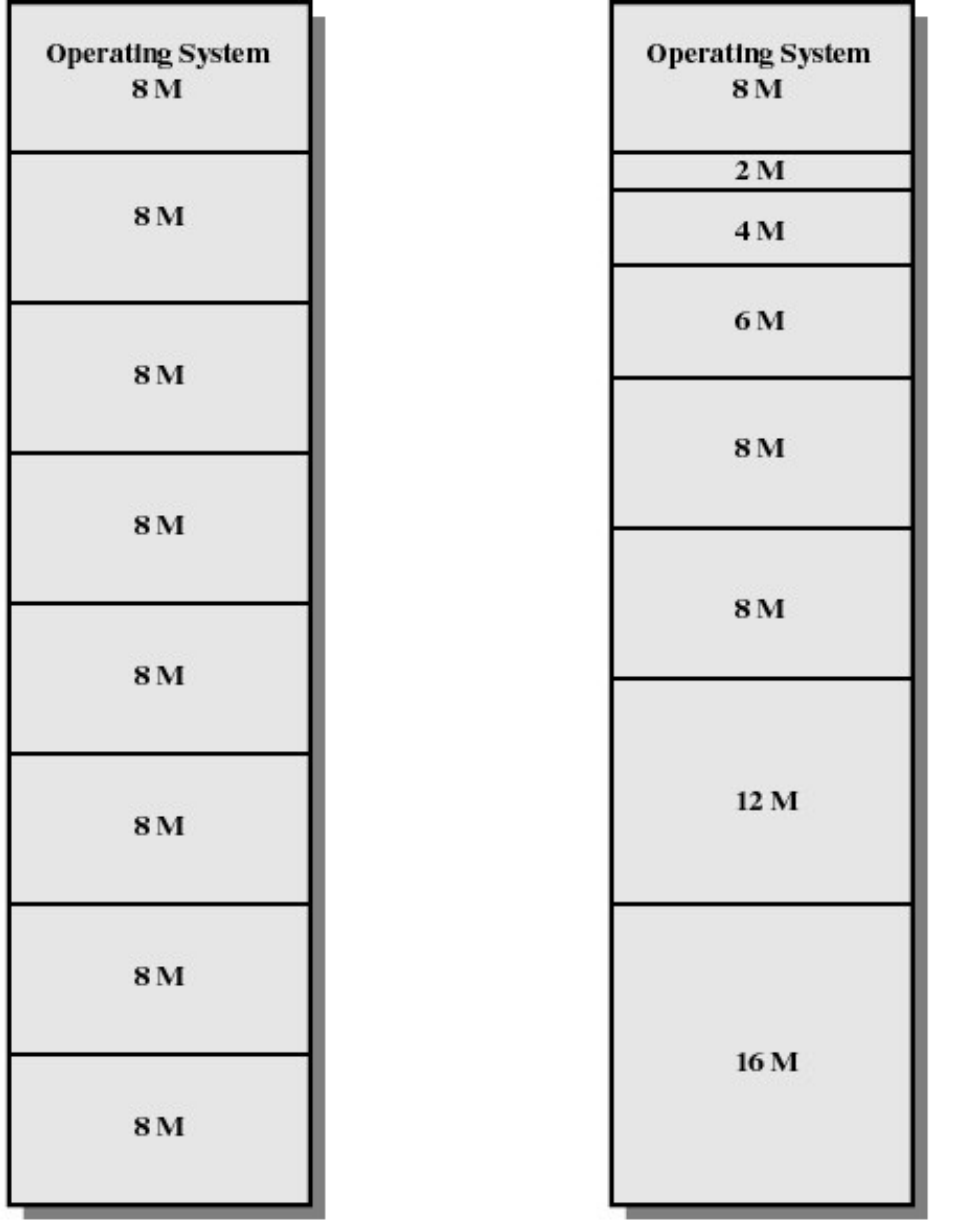
- 系统初始启动时将内存划分为数目固定、尺寸固定的多个分区。
- 这些分区的尺寸可以相等也可以不等。

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.

Fixed Partitioning

- A program may be too big to fit into 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 called **internal fragmentation (内零头).**
- Both of these problems can be lessened, though not solved, by using unequal-size partitions.



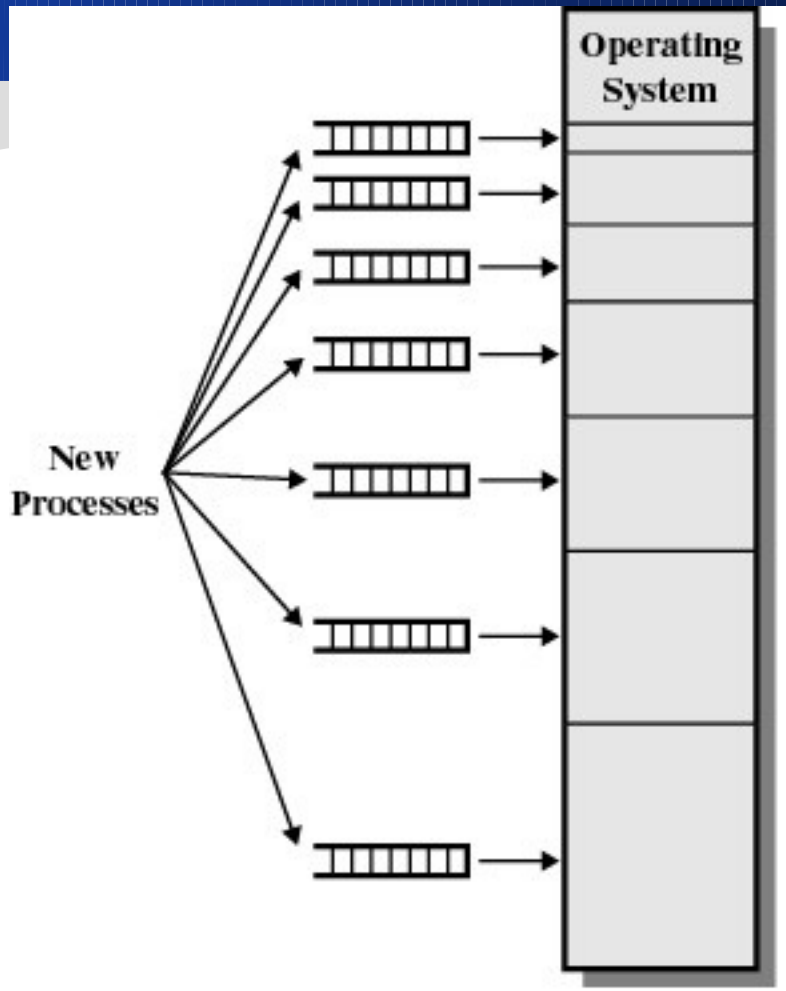
(a) Equal-size partitions

(b) Unequal-size partitions

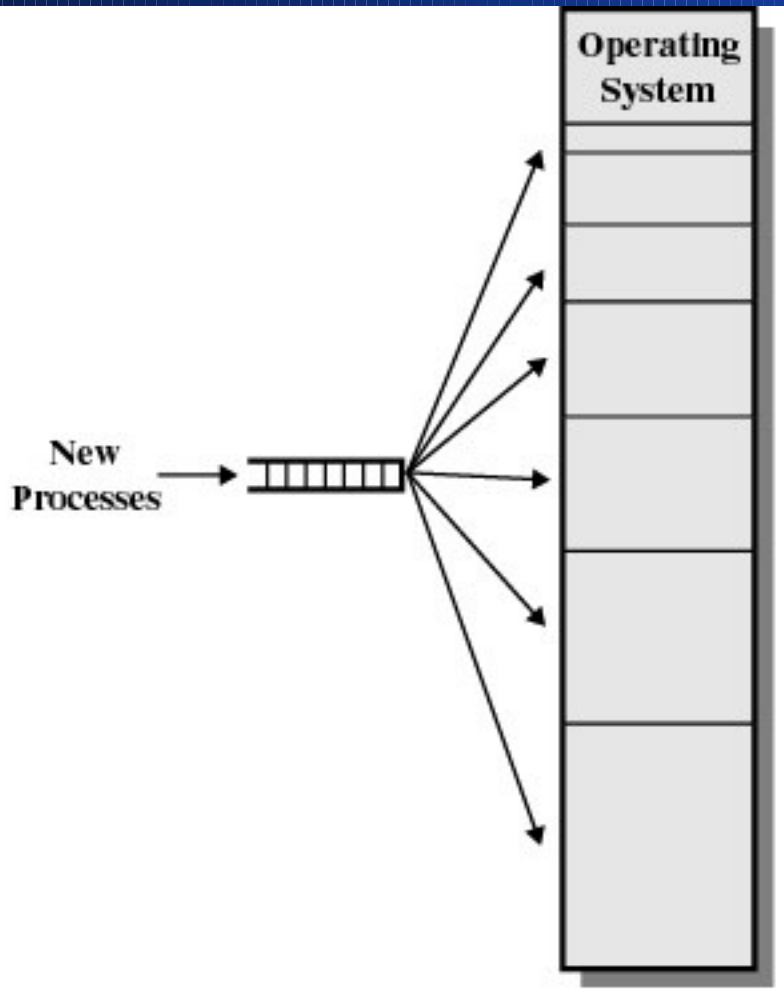
Figure 7.2 Example of Fixed Partitioning of a 64-Mbyte Memory

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.



(a) One process queue per partition



(b) Single process queue

Figure 7.3 Memory Assignment for Fixed Partitioning

电子科技大学
University of Electronic Science and Technology of China

Fixed Partitioning（长处和不足）

- ***Advantages***
  - 实现简单；
  - 系统开销小。

- ***Disadvantages***
  - 存在 ***Internal Fragment***，存储利用率不高；
  - 分区尺寸固定，系统无法运行大程序；
  - 分区数目固定，使活动进程的数目受限

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(*外零头*).**
- Must use **compaction(*紧凑*)** to shift processes so they are contiguous and all free memory is in one block.

Dynamic Partitioning（外零头和紧凑技术）

- 动态分区为进程分配大小合适的分区，消除了分区 *Internal Fragment* 。但是，却产生分区 *External Fragment*

- *Compaction* : 为了使外零头得到充分利用，可以将把内存中的所有空闲分区拼接成一个较大的空闲分区。即系统可以把内存中的所有进程移到内存的某一端；相应地，内存中的所有空闲分区将被移到内存的另一端。

- *Compaction* 技术要求系统具有动态重定位的能力

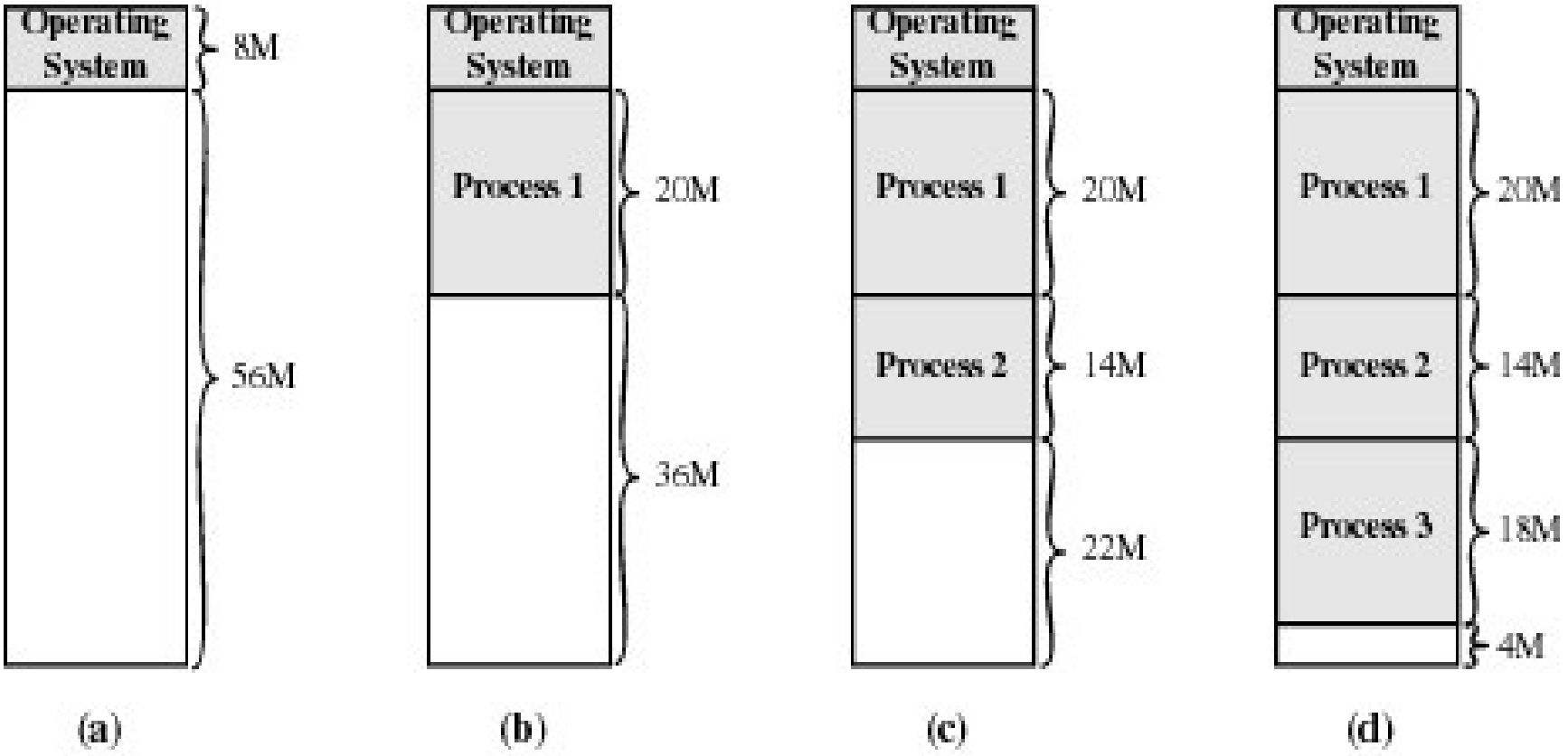


Figure 7.4 The Effect of Dynamic Partitioning

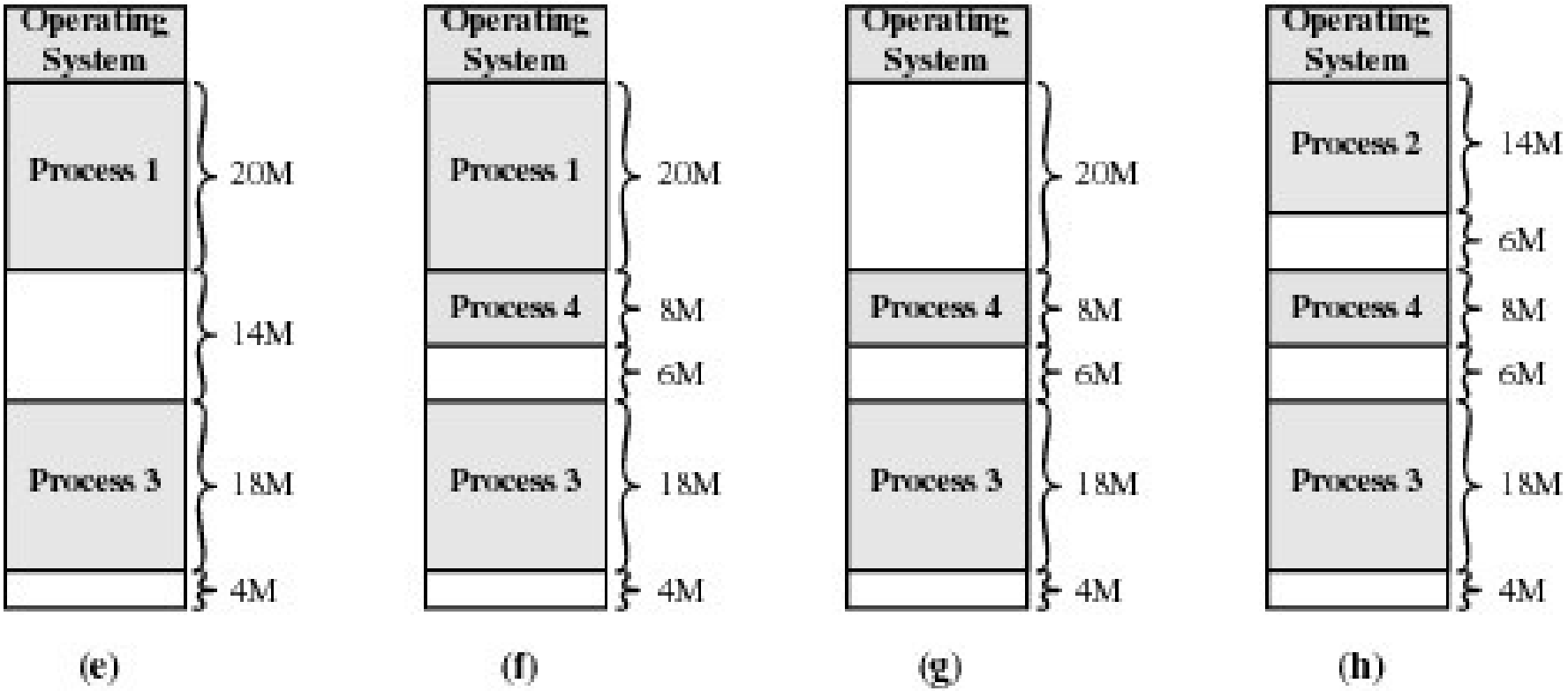


Figure 7.4 The Effect of Dynamic Partitioning

电子科技大学
University of Electronic Science and Technology of China

Dynamic Partitioning Placement Algorithm

- Operating system must decide which free block to allocate to a process.
- Best-fit algorithm(*最佳适应算法*)
  - 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(*首次适应算法*)**
  - **Scan memory from the beginning and choose the first available block that is large enough.**
  - **Simplest, Best and 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 Placement Algorithm

- **Next-fit(*下次适应算法*)**
  - **Scan memory from the location of the last placement and choose 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 more frequently to obtain a large block at the end of memory.**

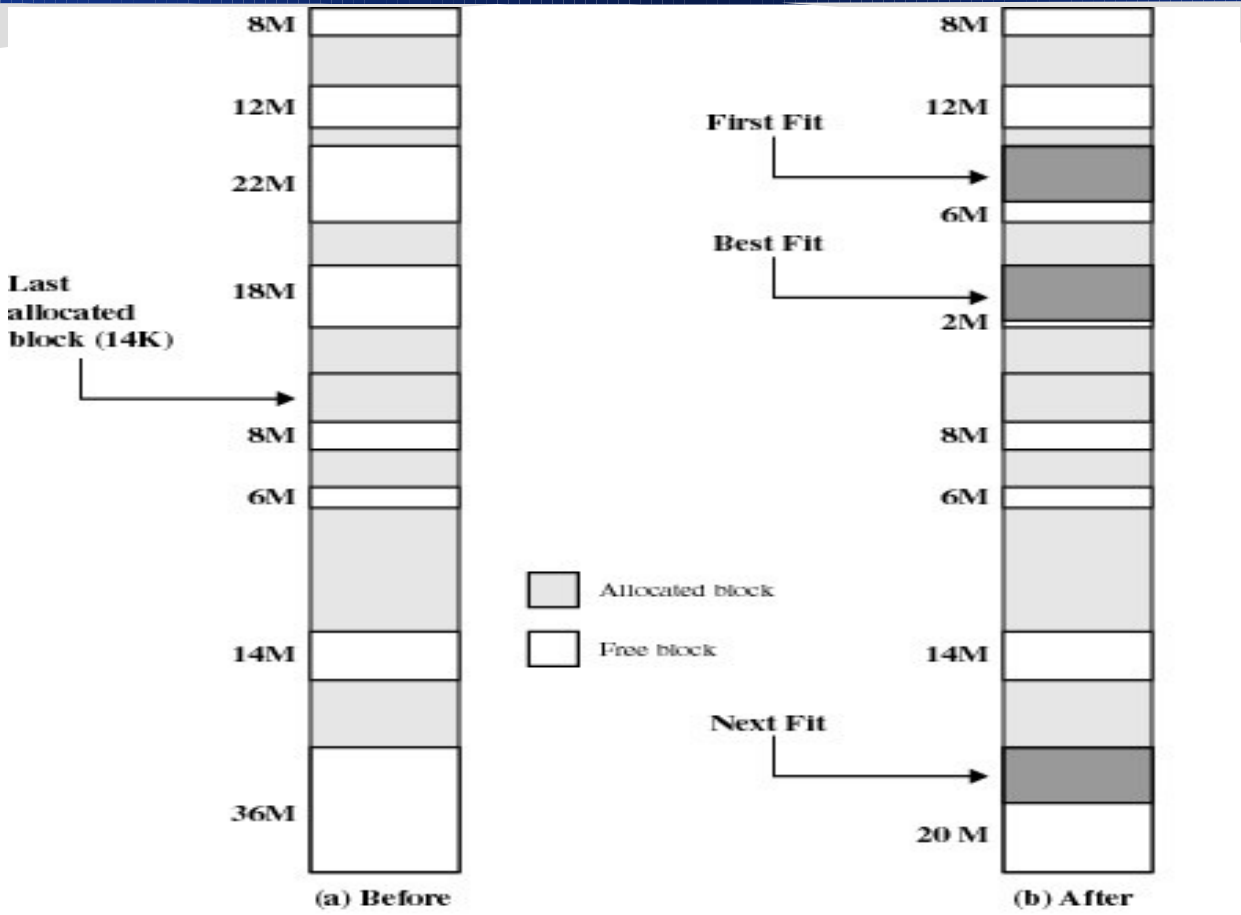


Figure 7.5 Example Memory Configuration Before and After Allocation of 16 Mbyte Block

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 and 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.

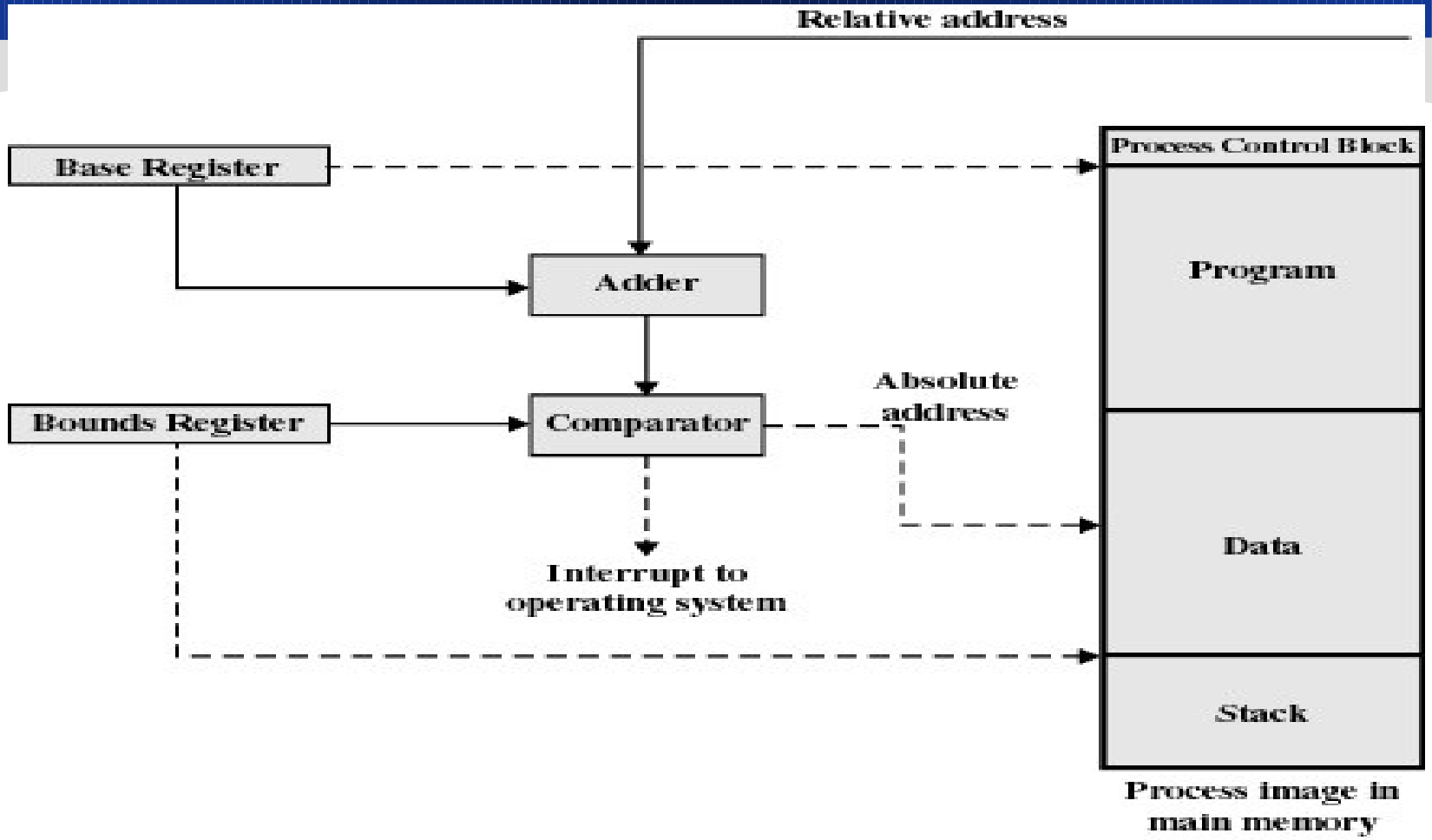


Figure 7.8 Hardware Support for Relocation

电子科技大学
University of Electronic Science and Technology of China