IT2164/IT2561 Operating Systems

# Tutorial 8

**Memory Management**

Attempt the following questions before you attend tutorial.

1. What are the 3 ways to assign addresses to the instructions in a computer program? Suppose we have a computer system in which we require to shift the program in memory whenever other programs get terminated or new programs come in. Suggest which method of address binding we can use and why.
2. Compile-time binding
3. Load-time binding
4. Run-time binding

Requires the use of run-time binding

This is because by calculating the actual address when the instruction is executed, programs are able to be moved anywhere in memory during execution.

The other 2 methods do not allow this as they assign fixed addresses during compile-time and loading-time.

1. Under what circumstances do external fragmentation and internal fragmentation occur? Can they occur together?

Internal fragmentation will occur when memory is divided into fixed-size blocks (fixed partition) and allocated in blocks (note that the sizes of each block may or may not be the same).

External fragmentation occurs when there is dynamic allocation of memory in blocks of different sizes (variable partition).

Over time, many small separate blocks will be created due to processes being added and terminated.

1. What is compaction? What problem does it solve? Which address-binding method do you think a feature like compaction requires?

### **What is Compaction?**

Compaction is a memory management technique used in operating systems to reduce fragmentation by rearranging memory contents. It consolidates free memory spaces into a single contiguous block, making it easier to allocate memory for new processes.

### **What Problem Does It Solve?**

Compaction primarily addresses **external fragmentation**, which occurs when free memory is divided into small, non-contiguous blocks, preventing large processes from being allocated despite sufficient total free space. By shifting processes and merging free memory, compaction ensures better utilization of available memory and allows larger processes to fit.

### **Which Address-Binding Method Does Compaction Require?**

Compaction requires **dynamic address binding**, where logical addresses are translated to physical addresses at runtime. Since compaction involves moving processes in memory, the address references need to be updated dynamically to reflect new locations. **Static and load-time binding would not work**, as they assign physical addresses that cannot change after allocation.

1. The following diagram shows a snapshot of the memory at time T0.

Program 1

–

100

K

Program 2

–

250

K

Program 3

–

K

150

Empty

–

160

K

Empty

–

K

140

Empty

–

200

K

1000

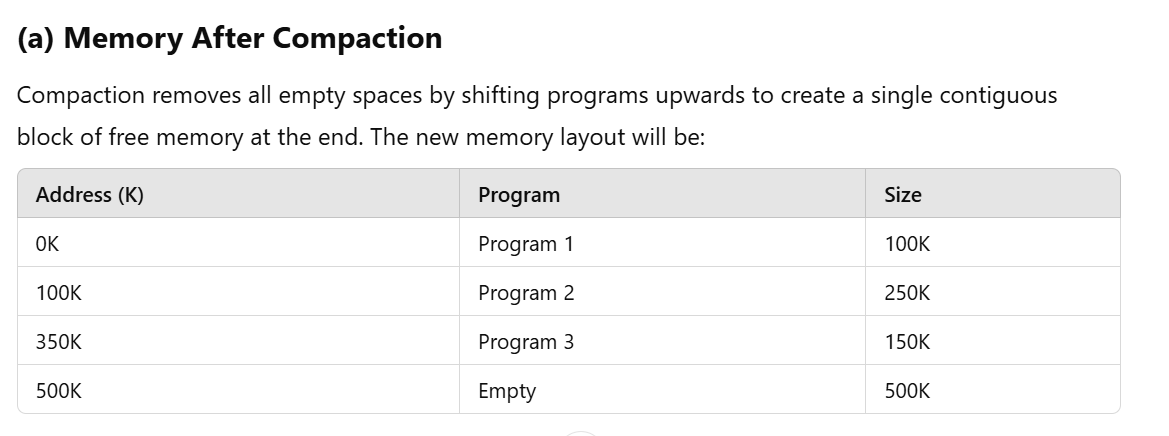
K

K

0

Load =7, R1

* 1. At time T1, the system undergoes compaction. With the aid of a diagram, show the result after compaction.



* 1. What is the value of the relocation register for Program 2 after it is moved during compaction?

### **(b) New Relocation Register for Program 2**

Before compaction, **Program 2** was located at **address 260K**.  
After compaction, it moves up to **address 100K**.  
Thus, the **relocation register for Program 2 is updated to 100K**.

* 1. Before compaction, the memory address of the instruction ‘Load =7, R1’ is 368640. What is the new memory address after compaction?

### **(c) New Memory Address for Instruction**

The instruction **‘Load =7, R1’** was originally at **368640 (decimal)**, which is in **Program 2**.

1. The **original base address** of Program 2 was **260K** (or 266240 in decimal).
2. The **new base address** after compaction is **100K** (or 102400 in decimal).
3. The instruction’s **offset** in Program 2 is: 368640−266240=102400
4. After compaction, the new memory address becomes: 102400+102400=204800

Thus, the new memory address for **‘Load =7, R1’** is **204800**.

Tutorial 9 Page 1 of 1