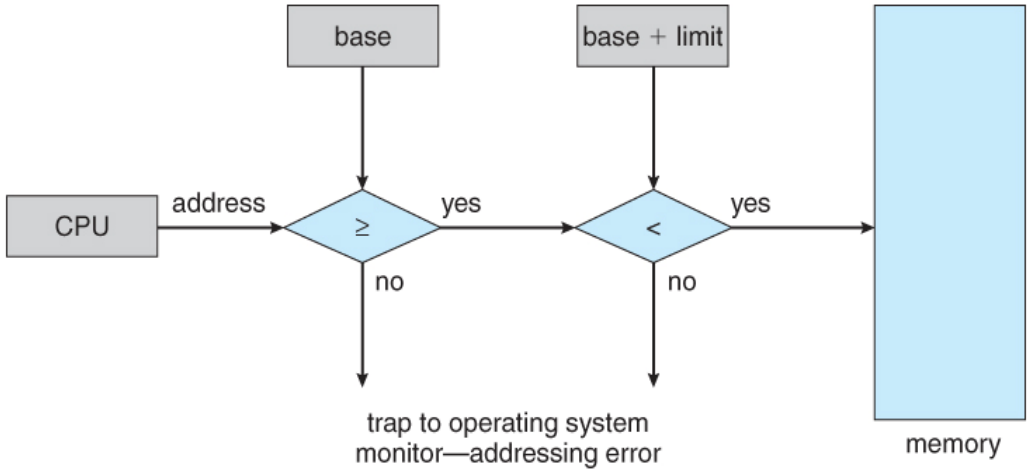
COMP3432 - PS 9

Operating Systems

Question 1: How is the memory protected from illegal accesses in contiguous memory allocation?

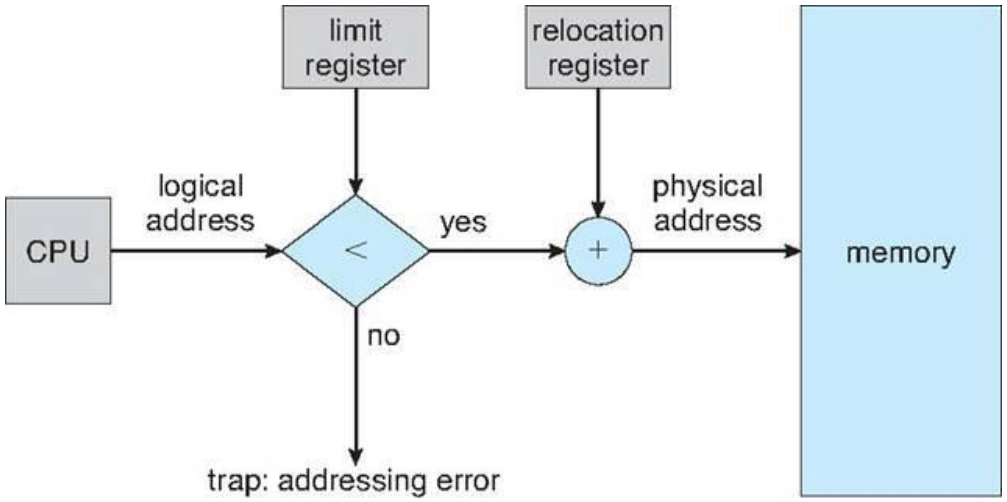
Question 1: How is the memory protected from illegal accesses in contiguous memory allocation?

- In contiguous memory allocation, a full segment of a memory is allocated to a process.
- When a process produces a memory address for instruction fetch, data read or write, the address is compared with the base and limit registers for the process.
- If the address is not in the range **[base, base+limit)**, the request is trapped to the OS.
- This strategy is for **compile time addressing**. Note that the addresses produced by the process are **absolute addresses** to the memory unit.



Question 1: How is the memory protected from illegal accesses in contiguous memory allocation?

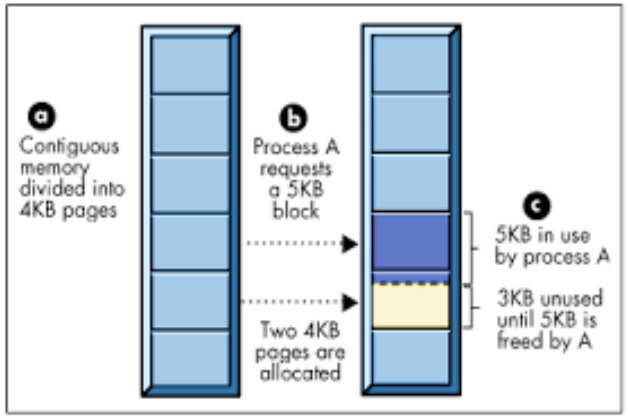
- If the process can be relocated (for example, after a **swap** operation), the addresses produced by the process will be **logical addresses** and it will **have to be converted** to **physical addresses**.
- In this case, a **relocation register** will translate a logical address to corresponding physical address.



Question 2: Explain internal and external fragmentation in memory allocation.

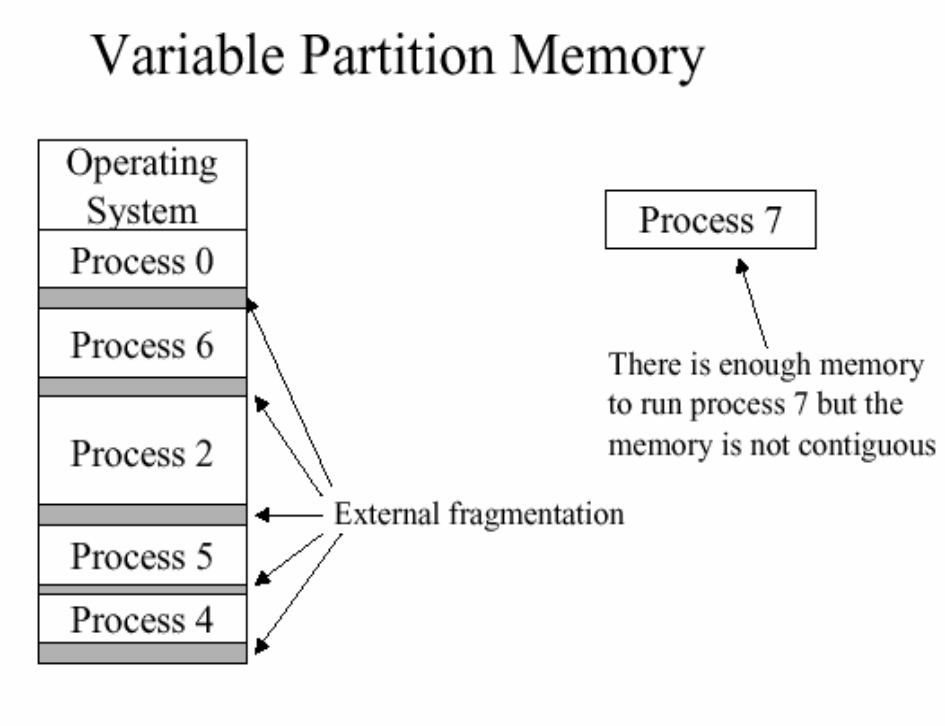
Question 2: Explain internal and external fragmentation in memory allocation.

- In **contiguous memory allocation**, the memory can be divided into fixed or varying partitions.
- When **fixed partition** strategy is used, each process is allocated exactly the same amount of memory.
- In this case, not all processes will use the entire space allocated to them, which will create **internal fragmentation**.



Question 2: Explain internal and external fragmentation in memory allocation.

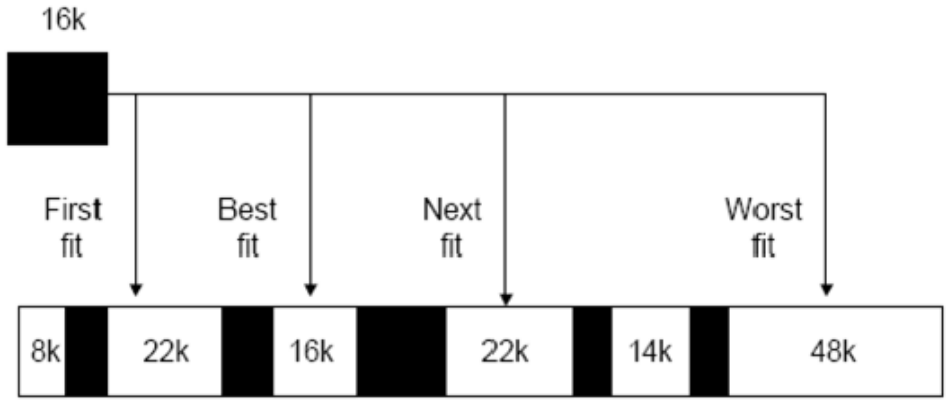
- On the other hand, when processes are allocated segments that vary in size, OS may not be able to fill the partitioned spaces when a process is terminated or swapped.
- In this case there is **unallocated space between partitions**, which is called **external fragmentation**.



Question 3: Describe the first fit, best fit and worst fit strategies in memory allocation.

Question 3: Describe the first fit, best fit and worst fit strategies in memory allocation.

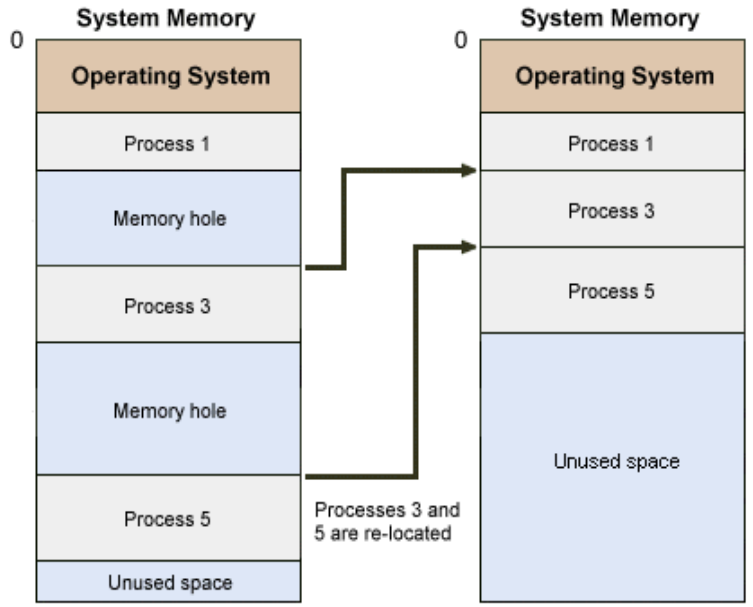
- When we use **varying partitions** in memory allocation, we experience **external fragmentation**.
- In this case, a new process (or a process swapped back from the disk) can be located in different sized partitions.
- If we just scan the free memory from the lowest memory address to highest, and **allocate the first big enough partition to the process**, this is **first fit** strategy.
- In **best fit**, we find the **smallest partition that is greater than or equal** to the memory requirement of the process.
- **Worst fit** is **allocating the largest partition to the process**, regardless of its memory requirement.



Question 4: What is compaction in memory management?

Question 4: What is compaction in memory management?

- **Compaction** is **relocation** of all processes in the memory to **eliminate external fragmentation**.
- This solves the **fragmentation** problem but it is **computationally costly**.



Question 5: What is paging? How is it implemented? Why does it double the memory accesses for each logical address produced by the CPU? How is the lookup made more efficient?

Question 5: What is paging? How is it implemented? Why does it double the memory accesses for each logical address produced by the CPU? How is the lookup made more efficient?

- Paging is **partitioning** the memory into **fixed** and **small segments**.
- These segments are named as **pages in the logical address space**, and **frames in the physical address space**.
- A **page table** is kept for each process, which **translates logical addresses to physical addresses**.
- This strategy **eliminates external fragmentation** and **minimizes internal fragmentation**.
- Since the page table is also kept in the main memory, each logical address produced by the CPU requires **two memory accesses**,
  - one of the page table
  - one for the physical address.

Question 5: What is paging? How is it implemented? Why does it double the memory accesses for each logical address produced by the CPU? How is the lookup made more efficient?

- Doubled memory accesses **reduces** the system performance.
- For this reason, **Translation Lookaside Buffer** is implemented in **hardware**.
- This is a **cache** that keeps **recent logical address requests** and **corresponding physical addresses**.
- TLB facilitates simultaneous query on each row of the cache, so the lookup is **very fast**.