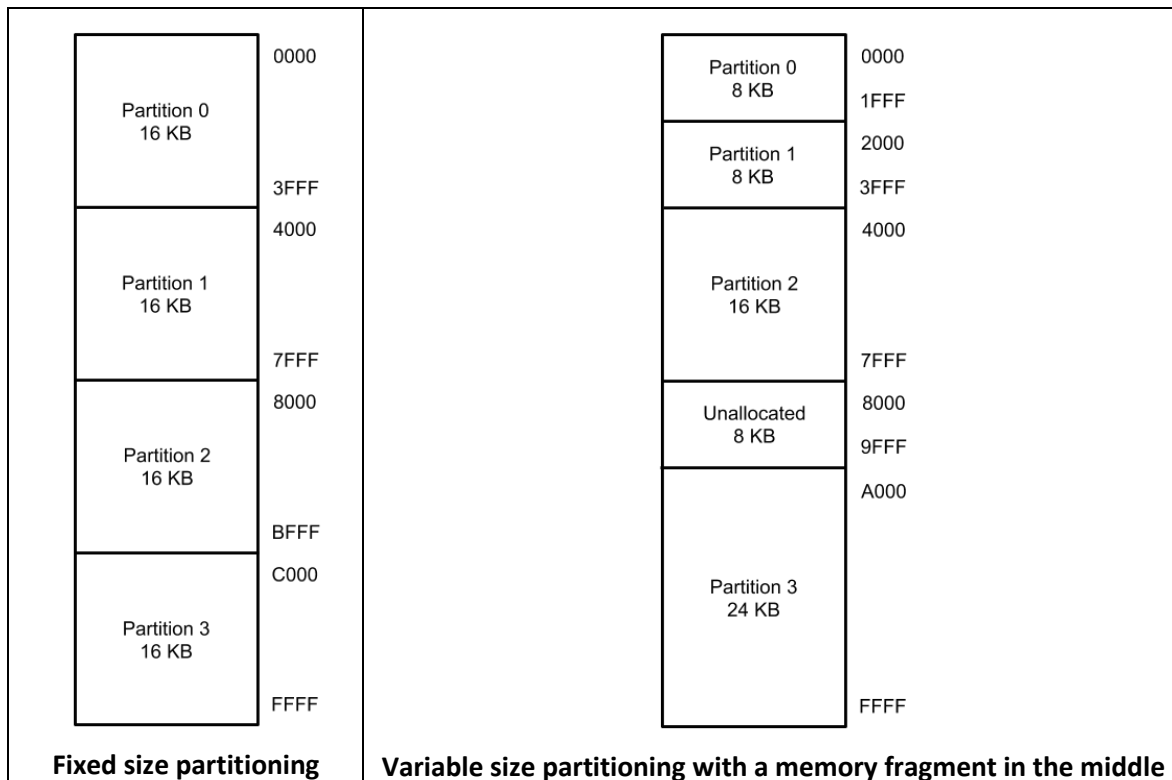


1. Memory management

- a. In a multiprogramming machine, memory dynamically allocated to OS and processes
 - i. Want more processes in RAM so that we have a bigger chance something is ready to execute
 - ii. Could expand RAM, but this is expensive
- b. Swapping – move non-ready processes out of RAM onto disk, into an intermediate queue
 - i. *Swap* another process in from the long-term queue which should be read to go
 - ii. Disk I/O is slow, and so it's possible that doing this can make the problem worse
 - iii. In general, though, swapping is good, but there are better alternatives
- c. Partitioning – divide RAM into blocks, each process gets a block
- d. Fixed size partitioning – a given partition is always the same size
 - i. Process gets placed into smallest available partition that holds it
 - ii. Somewhat wasteful – extremely small processes may be allocated far more than it needs
 1. Larger processes may not get enough memory
- e. Variable size (dynamic) partitioning – process allocated exactly as much memory as it needs
 - i. Memory fragments – small pockets of memory in non-contiguous chunks that are useless
 1. Too small to utilize or combine
 - ii. Eventually memory becomes fragmented due to these small holes
 1. OS must spend CPU time *compacting* processes
 2. Do so by moving their partitions next to each other



2. Virtual versus physical addresses
 - a. Discussed this with caches, now go into further depth
 - b. Logical/virtual address – location relative to the start of the program
 - i. You can imagine each process has an addressing space that starts at 0
 1. Exact start differs from OS to OS, though
 - ii. Instructions only contain logical addresses
 - iii. Programs start from 0 (in our example)
 1. Memory accesses are made based on that assumption
 - c. Physical addresses – actual addresses in RAM
 - d. Programs are always written/compiled to use virtual addresses!
 - i. What would happen if we wrote a program in terms of physical addresses?
 1. Imagine we move the process through compaction
 2. Move from partition 1 to 0 in the variable example above
 - ii. References in the instructions would reference wrong data now!
 1. 0x2015 is the physical address when the process is in partition 1
 2. 0x0015 is the physical address when the process is in partition 0
 - e. Base address – where a process starts in RAM
 - i. Virtual address + base address = physical address
 - ii. Base address can change if we change where the process resides in memory
 1. May change because of compaction, for example
 - f. MMU – memory management unit
 - i. MMU converts every virtual address access to the corresponding physical address
 - ii. Must do this for every single virtual address
 - iii. This is the cost of having a physically addressed cache versus a virtual one