1. Memory management
    a. In a multiprogramming machine, memory dynamically allocated to OS and processes
        i. Multiprogramming

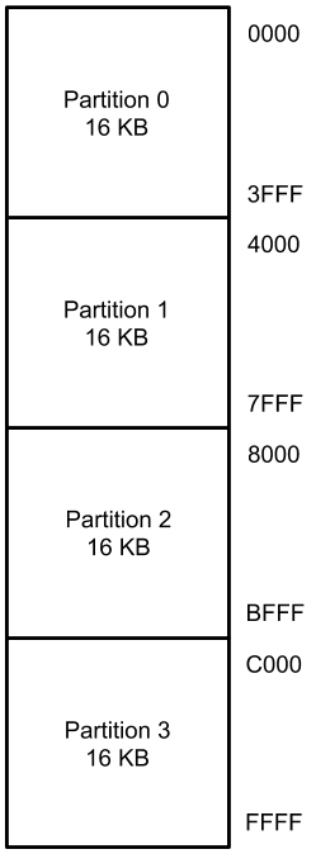
    b. Swapping

    c. Partitioning

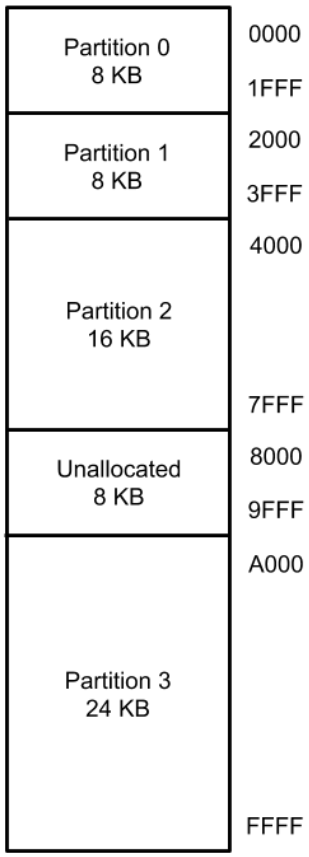
d. Fixed size partitioning

e. Variable size (dynamic) partitioning
   i. Memory fragments

   ii. Eventually memory becomes fragmented due to these small holes



Fixed size partitioning



Variable size partitioning with a memory fragment in the middle

2. Virtual versus physical addresses
    a. Discussed this with caches, now go into further depth
    b. Logical/virtual address

    c. Physical addresses

    d. Programs are always written/compiled to use virtual addresses!
        i. What would happen if we wrote a program in terms of physical addresses?

    e. Base address

    f. MMU

3. Virtual memory overview
    a. Fixed and variable-size partitioning is inefficient

    b. Instead, let's divide RAM into small, fixed size chunks called *frames*

    c. Processes' address space can be divided into *pages* that are the same size as RAM frames

        i. Imagine a picture frame

    d. Given process can have one or more of its pages in RAM at a time

    e. Virtual memory reduces the memory fragmentation of partitioning

4. Virtual memory and ties to caching
    a. Virtual memory only keeps active pages of each process in memory

    b. When a process references data that is in a page not in RAM, have a *page fault*

    c. When a page fault occurs, must overwrite existing page in RAM with the new page

5. Ways to fetch pages
    a. Demand paging

        i. *Thrashing*

    b. Alternative based on locality (both spatial and temporal)

    c. Idea: could prefetch pages before they're needed