How is time-sharing achieved?

- By switch among processes on a cpu and create illusion that system is dedicated to a process
- Question: How can the physical memory (RAM) be split among processes?
- Solution: Chunk memory into some partitions and give each process a partition.
- Question #2: What if there are many processes?
- Solution: Swap (moving the memory into disk) out memory of waiting process and give that to a process that is selected for running. This allows N processes to run using M partitions of memory (N > M). This is similar to time-sharing of CPU.
- Thus we end up in memory virtualization: Allow memory to be shared among multiple programs in a system

Issues that arise

- Addressing: The program might not be get the same memory location in RAM every time when swapped in.
- Protection: Program A can access program B's data since B's data also resides in memory

What is an address space?

- Another abstraction provided by the OS like process abstraction that tries to solve the above problems
- Every process has its own address space and programs have no idea if the physical memory is shared with other programs
- Address space is an imaginary contiguous memory region (like a stub) starting with address zero. So, every program's address space will have its own address zero.
- Addresses in the address space is referred to as the virtual address and all addresses within a program (code address or data address) are virtual addresses.
- Maximum size of the address space is $2^{\# \text{ of address bits}}$ (32 in 32-bit systems and 64 in 64-bit systems). Address space could be larger than physical memory (swapping comes into picture here)

Goals

- Transparency: Address space could be mapped anywhere on the physical memory. OS should translate virtual address to physical address without the programs knowing it.
- Protection: A program should access only its memory and should not be able to access other program's memory.
- Efficiency: Better utilization of memory

Assumption

- Contiguous
- Address space is smaller than physical memory
- each address space is same size
Virtual Address cannot be used to access memory

- Addresses used and generated by the program is of virtual address type (Virtual address remains the same even if program is relocated)
- But virtual address cannot be used to access the physical memory and so, we need a way for this conversion to happen

Hardware support for address translation

- Similar to the hardware support (trap and timer interrupt) provided for CPU virtualization
- Two registers are available in the CPU: Base and bounds registers (Also referred as memory management unit - MMU)
- Base register contains the physical address where the address space of the program is mapped by OS
- Bound register contains the size of the address space
- When a process is scheduled, OS is responsible to populate these CPU registers
- When a process is context switched out, OS is responsible to save these registers (in process Control Block) similar to how it saves register context

How translation happens?

- When a virtual address is generated by the program, it is added to the base register to get the actual physical address
- A check is made to ensure that the virtual address is smaller than the bounds register (Protection: This ensures that a program cannot access memory other than its own)

Duties of hardware and software

- Hardware (MMU within CPU)
  - Responsible for VA to PA translation
- Software (OS)
  - Maintain free list
  - Collect memory assigned to address space when process exits
  - Save base and bound register on context switch to PCB
  - Populate base and bound register in CPU from the new process's PCB

Drawbacks

- Internal fragmentation: Memory unused within process is wasted (low memory utilization)

Questions

1. What is the problem if \( N \gg M \) i.e. number of processes is very high compared to memory partitions (Think along the lines of too much swapping)
2. What is the impact of swapping when the amount of RAM installed in the system is increased?
3. Why do we need to do relocate address space in physical memory?
4. What happens if (a) changing base register is not a privilege instruction? (b) changing bounds register is not a privilege instruction?
5. How would software based address translation work? And is it possible achieve protection in software based address translation?

6. Memory virtualization introduces a layer of indirection (e.g. program generates a virtual address which then has to be converted to physical address). Can you identify similar examples in OS for layer indirection?