Memory Management

- Memory Management Basics
- Virtual Memory

Memory Access

- Memory Hierarchy:
  - Registers
  - Cache
  - Main Memory (RAM)
  - Secondary Storage: E.g.: Disk
- A process can directly address only registers or RAM
  - Must load instructions from RAM
  - Must load and store data (global, local, dynamic data) from/to RAM

Multiprogramming and Protection

- Multiple processes executing concurrently
  - These need to reside in the memory
  - OS needs to multiplex the available memory among multiple processes
- OS must provide each process with a separate memory space
  - A process cannot access another process’s memory
  - Kernel memory must not be accessed in user mode
  - OS must restrict set of addresses accessed by a process
Simple Addressing Scheme

- Two registers:
  - Base: Starting address for a process
  - Limit: Size of address space
- When a process accesses an address, check:
  - Address > Base
  - Address < Base+Limit
- Only OS can load the Base and Limit registers

How do addresses get generated?

- Given a program, how do we generate the addresses for instructions, variables, etc.?
- A program goes through multiple steps:
  - Compilation/Linking: Translate source code to binary executable code
    - Symbolic addresses (e.g.: offset in a function, offset from beginning of program binary)
  - Loading: Load program binary to be executed
    - Absolute "logical" addresses
  - Execution: Actual running of a process
    - Absolute "physical" addresses

Address Binding

- When do we bind the final physical addresses?
  - Impacts where in physical memory can the process be placed and executed
- Compile time:
  - Absolute physical addresses are generated
- Load time
  - Address binding done when program loaded into memory
- Execution time
  - Address binding done while process is running

Virtual Memory

- The process is given a "logical" view of the memory
  - Contiguous
  - Starting from low address
  - Ending at high address (based on the architecture)
- The process references "virtual" memory addresses
  - Addresses generated by loader
  - Process has no knowledge of where an address would actually lie in physical memory
Virtual Memory Mapping

- Virtual addresses are mapped to physical addresses at runtime
  - Transparent to the process
- Memory Management Unit (MMU):
  - Hardware unit that does this runtime mapping
  - The mapping is controlled by the OS and the hardware
- Multiple processes can have same virtual addresses mapped to different physical addresses

Virtual Memory: Benefits

- Physical memory layout transparent to process
  - Virtual addresses can be generated at link/load time
- Multiple processes can use the same (virtual) contiguous address space
- Can share physical memory among processes
  - E.g.: shared libraries
- Memory Protection
  - A process cannot address another process’s memory
- Process memory not constrained by available RAM
  - Virtual address space can be of any size (modulo architectural constraints)