Basic Memory Management
Readings

- Silberschatz et al: chapters 8.1-8.2
Outline

- Memory Hierarchy
- Memory management requirements
Introduction

- Our machines today have 10,000 times more memory than the IBM 7094 – leading edge machine of the 1960’s
- Cost of memory has dropped dramatically
- Bill Gates (former chair of Microsoft) once said “640K should be enough”
- There is a high-performance machine at Western (for bioinformatics research) with 1 terabyte of memory
Introduction

- Software and data sets expand to fill the memory available
  - The 1 terabyte of memory - the researchers already want more

- Technology does not allow for each program to have infinitely large and fast memory

- Operating systems must manage memory
Introduction

- Memory management requires
  - Keep track of what parts of memory are in use.
  - Allocate memory to processes when needed
  - Deallocate memory when processes are done
Instruction-Execution Cycle

- You can think of memory as a large array of bytes
  - Each byte has its own address
- Fetch an instruction from memory
- Instruction is decoded
  - May cause operands to be fetched from memory
- After instruction execution
  - Results may be stored back in memory
- Each of these operations require memory addresses
Registers built into the CPU are generally accessible within one cycle of the CPU clock.

Completing a memory access may take many cycles of the CPU clock.
Memory Hierarchy

- A processor waiting for data from main memory is not desired
- Remedy: Add fast memory between the CPU and main memory called a cache
Memory Management

- Memory is to be shared by multiple processes
- Processes should not be able to reference another process’s memory without permission
  - This requires an ability to determine the range of legal addresses that the process may access
Address Binding

- Program execution requires that a program be brought into memory from the disk.
- The process can reside in any part of the physical memory.
- Address space starts at 00000 but the first address of a user need not be 00000.
Address Binding

- User program goes through several steps before it is ready to be executed
- The binding of instructions/data to memory addresses can be done at any step along the way
Address Binding

- Addresses in the source program are generally symbolic e.g., count
- Compiler will bind these symbolic addresses to relocatable addresses
  - Example: “14 bytes from the beginning of this module”
The loader will in turn bind the relocatable addresses to absolute addresses.
Address Binding

- **Execution time:**
  - Binding to actual physical memory addresses must be delayed until run-time.
Address Binding

- Typically the programmer does not know where the program will be placed in memory when it is executed.

- While the program is executing, it may be placed in disk and returned to main memory at a different location (swapping).
Address Binding

- Memory references in the code (virtual or logical) must be translated to actual physical memory addresses.

- Run-time Mapping from virtual to physical addresses is done by a hardware device called the memory-management unit (MMU).

- Run-time mappings depend on how memory is allocated to processes.