Chapter 8
Main Memory

Images from Silberschatz
How does the OS manage memory?

- Allocation
- Swapping
- Hardware support
- Pentium + Linux

- Assume the entire process must be in memory!
  - Virtual Memory – chapter 9
  - Does not make this assumption
Memory Access Basics

- Register
- Cache
  - Stall
- Main Memory
- Disk
- Protection
(Basic) Mapping + Protection

- **Software**
  - Thinks it can access address zero to limit

- **Hardware**
  - Two registers
    - Base
    - Limit
  - Privileged instructions
    - Kernel mode!
  - On error
    - Trap!

![Diagram showing mapping and protection with registers and memory segments]

- Base: 30004
- Limit: 12090
- Operating system
- Processes
CPU → address

≥

no

≥

no

trap to operating system
monitor—addressing error

memory
Address Bind Time

- When are addresses in the executable set?
  - Compile time
    - Must always be in the same location
  - Load time
    - Can be loaded anywhere
  - Execution time
    - Can be *moved* during execution!

Diagram:
- Source program
- Compiler or assembler
- Object module
- Linkage editor
- Load module
- Loader
- System library
- Dynamically loaded system library
- In-memory binary memory image
- Dynamic linking
- Other object modules
Logical vs Physical Addresses

- **Logical Address (Virtual Address)**
  - Software only ever sees this!

- **Physical Address**

- **Memory Management Unit**
  - Generalization of the base/limit register method
  - Relocation register
Dynamic Linking

- Linking at execution time
- Static linking
- stub
- Shared libraries
  - .dll or .so
Swapping

• Not all processes fit in physical memory
  – Chapter 9: not all of a *single process* will fit into physical memory

• Physical memory $\leftrightarrow$ Backing store

• Swap back into memory
  – Same location
  – Different location

• Context Switch Time
  – Size * Transfer rate
  – How does this affect time slices?
Contiguous Memory Allocation

- Two Partitions
  - OS
  - User Processes
Allocation of Memory

- Allocate part of User Space partition to each process
- Hole (technical term)

- First Fit
  - OS
    - process 5
    - process 8
    - process 2

- Best Fit
  - OS
    - process 5
    - process 9

- Worst Fit
  - OS
    - process 5
    - process 2

- Best Fit/First Fit found (experimentally) to be better than Worst Fit in terms of time and memory utilization

- What happens if 5 & 2 terminate?
Fragmentation

- External
- Internal
- Compaction
- 50% Rule
Paging!

- Noncontiguous memory allocation
- Frame
  - Physical memory
- Page
  - Logical memory
  - Allocate an entire page at a time
- Page table
- Internal Fragmentation
Page and Frame size are determined by the hardware.
Address Translation

- Logical Address to Page Number + Offset

  \[
  \begin{array}{cc}
  \text{page number} & \text{page offset} \\
  p & d \\
  \end{array}
  \]

  \[m - n \quad n\]

- Logical address space \(2^m\) and page size \(2^n\)
- 32 byte memory
- 4 byte pages
- No guarantee of ordering
- What happens

```c
char *pChar = 0x7;
pChar ++;
printf(pChar);
```
Page Table

• Pages are not always reloaded to the same frame
  - ??

• Contains base address of each page in physical memory
  - Per process (usually)
  - Which frame is it in
  - In main memory

• Hardware (not per process)
  - Page table base register (PTBR)
  - Page table length register (PRLR)
  - Translation look-aside buffers (TLBs)
    • Address space identifiers (ASIDs)
    • protection
CPU

logical address

p d

page number frame number

TLB

TLB hit

f d

physical address

physical memory

TLB miss

p

f

page table
Logical -> Physical Address

• What do we need to do to get a physical address?
  – How long will it take?
Protection

- Add valid/invalid bit to each page table entry

- ASIDs in TLBs denote which process owns each frame
Shared Pages

- `.dll / .so`
  - Share read only code pages

- Shm
  - Shared read/write data pages
Problems with page tables

- What do you think?
Multilevel Page Tables

- Page the page table
- Forward mapped page table
Hashed Page Tables

- Address spaced > 32 bits
- Use Virtual address to hash into the table
Inverted Page Table

- One entry per **frame** in physical memory
- One page table for the entire system
- Track pid in the table
- Problem?

- Solution?