

# **CSE 120**

## **Principles of Operating Systems**

**Fall 2001**

**Lecture 8: Memory Management**

Geoffrey M. Voelker

## **Memory Management**

---

Next few lectures are going to cover memory management

- Goals of memory management
  - ◆ To provide a convenient abstraction for programming
  - ◆ To allocate scarce memory resources among competing processes to maximize performance with minimal overhead
- Mechanisms
  - ◆ Physical and virtual addressing (1)
  - ◆ Techniques: Partitioning, paging, segmentation (1)
  - ◆ Page table management, TLBs, VM tricks (2)
- Policies
  - ◆ Page replacement algorithms (3)

## Lecture Overview

---

- Virtual memory warm-and-fuzzy
- Survey techniques for implementing virtual memory
  - ♦ Fixed and variable partitioning
  - ♦ Paging
  - ♦ Segmentation
- Focus on hardware support and lookup procedure
  - ♦ Next lecture we'll go into sharing, protection, efficient implementations, and other VM tricks and features

## Virtual Memory

---

- The abstraction that the OS will provide for managing memory is virtual memory (VM)
  - ♦ Virtual memory enables a program to execute with less than its complete data in physical memory
    - » A program can run on a machine with less memory than it “needs”
    - » Can also run on a machine with “too much” physical memory
  - ♦ Many programs do not need all of their code and data at once (or ever) – no need to allocate memory for it
  - ♦ OS will adjust amount of memory allocated to a process based upon its behavior
  - ♦ VM requires hardware support and OS management algorithms to pull it off
- Let's go back to the beginning...

## In the beginning...

---

- Rewind to the days of batch programming
  - ♦ Programs use **physical addresses** directly
  - ♦ OS loads job, runs it, unloads it
- Multiprogramming changes all of this
  - ♦ Want multiple processes in memory at once
    - » Overlap I/O and CPU of multiple jobs
  - ♦ Can do it a number of ways
    - » Fixed and variable partitioning, paging, segmentation
  - ♦ Requirements
    - » Need protection – restrict which addresses jobs can use
    - » Fast translation – lookups need to be fast
    - » Fast change – updating memory hardware on context switch

October 19, 2001

CSE 120 – Lecture 8 – Memory Management

5

## Virtual Addresses

---

- To make it easier to manage the memory of processes running in the system, we're going to make them use **virtual addresses** (logical addresses)
  - ♦ Virtual addresses are independent of the actual physical location of the data referenced
  - ♦ OS determines location of data in physical memory
  - ♦ Instructions executed by the CPU issue virtual addresses
  - ♦ Virtual addresses are translated by hardware into physical addresses (with help from OS)
  - ♦ The set of virtual addresses that can be used by a process comprises its **virtual address space**
- Many ways to do this translation...
  - ♦ Start with old, simple ways, progress to current techniques

October 19, 2001

CSE 120 – Lecture 8 – Memory Management

6

# Fixed Partitions

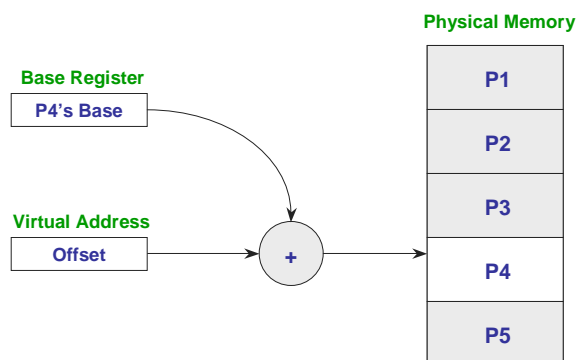
- Physical memory is broken up into fixed partitions
  - Hardware requirements: **base register**
  - Physical address = virtual address + base register
  - Base register loaded by OS when it switches to a process
  - Size of each partition is the same and fixed
  - How do we provide protection?
- Advantages
  - **Easy to implement, fast context switch**
- Problems
  - **Internal fragmentation**: memory in a partition not used by a process is not available to other processes
  - **Partition size**: one size does not fit all (very large processes?)

October 19, 2001

CSE 120 – Lecture 8 – Memory Management

7

# Fixed Partitions



October 19, 2001

CSE 120 – Lecture 8 – Memory Management

8

# Variable Partitions

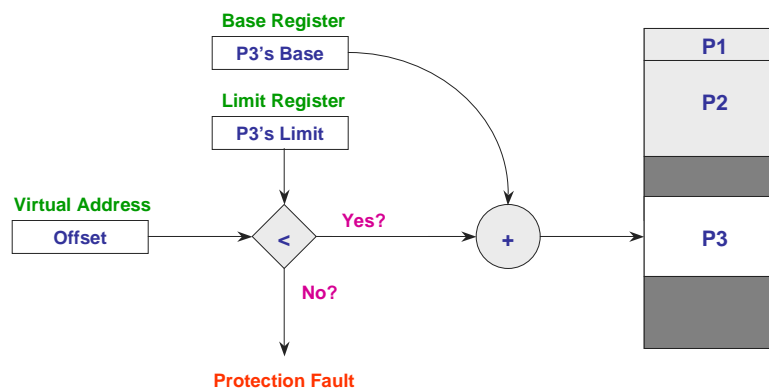
- Natural extension -- physical memory is broken up into variable sized partitions
  - Hardware requirements: **base register** and **limit register**
  - Physical address = virtual address + base register
  - Why do we need the limit register? Protection
  - If (physical address > base + limit) then exception fault
- Advantages
  - **No internal fragmentation**: allocate just enough for process
- Problems
  - **External fragmentation**: job loading and unloading produces empty holes scattered throughout memory

October 19, 2001

CSE 120 – Lecture 8 – Memory Management

9

# Variable Partitions



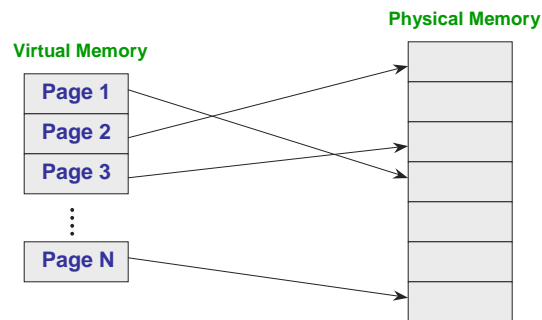
October 19, 2001

CSE 120 – Lecture 8 – Memory Management

10

# Paging

- Paging solves the external fragmentation problem by using fixed sized units in both physical and virtual memory



October 19, 2001

CSE 120 – Lecture 8 – Memory Management

11

# User/Process Perspective

- Users (and processes) view memory as one contiguous address space from 0 through N
  - Virtual address space (VAS)
- In reality, pages are scattered throughout physical storage
- The mapping is invisible to the program
- Protection is provided because a program cannot reference memory outside of its VAS
  - The address "0x1000" maps to different physical addresses in different processes

October 19, 2001

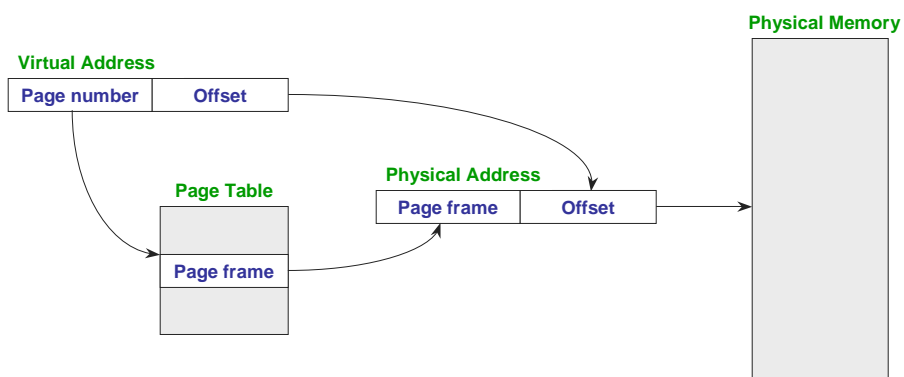
CSE 120 – Lecture 8 – Memory Management

12

# Paging

- Translating addresses
  - Virtual address has two parts: virtual page number and offset
  - Virtual page number (VPN) is an index into a page table
  - Page table determines page frame number (PFN)
  - Physical address is PFN::offset
- Page tables
  - Map virtual page number (VPN) to page frame number (PFN)
    - » VPN is the index into the table that determines PFN
  - One page table entry (PTE) per page in virtual address space
    - » Or, one PTE per VPN

# Page Lookups



## Paging Example

- Pages are 4K
  - VPN is 20 bits ( $2^{20}$  VPNs), offset is 12 bits
- Virtual address is 0x7468
  - Virtual page is 0x7, offset is 0x468
- Page table entry 0x7 contains 0x2000
  - Page frame number is 0x2000
  - Seventh virtual page is at address 0x2000 (second physical page)
- Physical address =  $0x2000 + 0x468 = 0x2468$

## Page Table Entries (PTEs)



- Page table entries control mapping
  - The Modify bit says whether or not the page has been written
    - » It is set when a write to the page occurs
  - The Reference bit says whether the page has been accessed
    - » It is set when a read or write to the page occurs
  - The Valid bit says whether or not the PTE can be used
    - » It is checked each time the virtual address is used
  - The Protection bits say what operations are allowed on page
    - » Read, write, execute
  - The page frame number (PFN) determines physical page

## Paging Advantages

---

- Easy to allocate memory
  - Memory comes from a free list of fixed size chunks
  - Allocating a page is just removing it from the list
  - External fragmentation not a problem
- Easy to swap out chunks of a program
  - All chunks are the same size
  - Use valid bit to detect references to swapped pages
  - Pages are a convenient multiple of the disk block size

## Paging Limitations

---

- Can still have internal fragmentation
  - Process may not use memory in multiples of a page
- Memory reference overhead
  - 2 references per address lookup (page table, then memory)
  - Solution – use a hardware cache of lookups (more later)
- Memory required to hold page table can be significant
  - Need one PTE per page
  - 32 bit address space w/ 4KB pages =  $2^{20}$  PTEs
  - 4 bytes/PTE = 4MB/page table
  - 25 processes = 100MB just for page tables!
  - Solution – page the page tables (more later)

# Segmentation

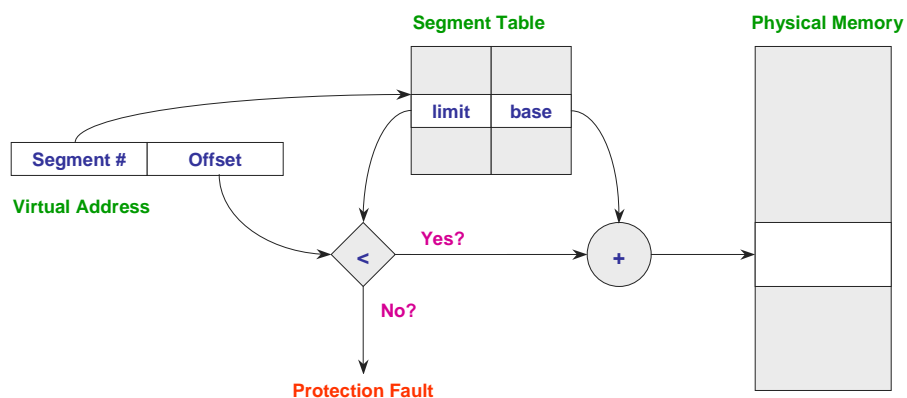
- Segmentation is a technique that partitions memory into logically related data units
  - Module, procedure, stack, data, file, etc.
  - Virtual addresses become <segment #, offset>
  - **Units of memory from user's perspective**
- Natural extension of variable-sized partitions
  - Variable-sized partitions = 1 segment/process
  - Segmentation = many segments/process
- Hardware support
  - Multiple base/limit pairs, one per segment (segment table)
  - Segments named by #, used to index into table

October 19, 2001

CSE 120 – Lecture 8 – Memory Management

19

# Segment Lookups



October 19, 2001

CSE 120 – Lecture 8 – Memory Management

20

## Segment Table

---

- Extensions
  - ♦ Can have one segment table per process
    - » Segment #s are then process-relative (why do this?)
  - ♦ Can easily share memory
    - » Put same translation into base/limit pair
    - » Can share with different protections (same base/limit, diff prot)
- Problems
  - ♦ Cross-segment addresses
    - » Segments need to have same #s for pointers to them to be shared among processes
  - ♦ Large segment tables
    - » Keep in main memory, use hardware cache for speed

## Segmentation and Paging

---

- Can combine segmentation and paging
  - ♦ The x86 supports segments and paging
- Use segments to manage logically related units
  - ♦ Module, procedure, stack, file, data, etc.
  - ♦ Segments vary in size, but usually large (multiple pages)
- Use pages to partition segments into fixed size chunks
  - ♦ Makes segments easier to manage within physical memory
    - » Segments become “pageable” – rather than moving segments into and out of memory, just move page portions of segment
  - ♦ Need to allocate page table entries only for those pieces of the segments that have themselves been allocated
- Tends to be complex...

## Summary

---

- Virtual memory
  - ♦ Processes use virtual addresses
  - ♦ OS + hardware translates virtual address into physical addresses
- Various techniques
  - ♦ Fixed partitions – easy to use, but internal fragmentation
  - ♦ Variable partitions – more efficient, but external fragmentation
  - ♦ Paging – use small, fixed size chunks, efficient for OS
  - ♦ Segmentation – manage in chunks from user's perspective
  - ♦ Combine paging and segmentation to get benefits of both

## Next time...

---

- Chapter 4.3