Virtual Memory

Steve Goddard
goddard@cse.unl.edu

http://www.cse.unl.edu/~goddard/Courses/CSCE451
Virtual Memory
Concept

- Hide all physical aspects of memory from users
  - Memory is a logically unbounded virtual address space of $2^n$ bytes
  - Only portions of VAS are in physical memory at any one time

- Issues
  - Placement strategies
  - Replacement strategies
  - Load control strategies

Realizing Virtual Memory
Paging

- Physical memory partitioned into equal sized page frames

Memory address is a pair $(f, o)$
- $f$ — frame number ($2^{f_{MAX}}$ frames)
- $o$ — frame offset ($2^{o_{MAX}}$ bytes/frames)
- Physical address = $2^{o_{MAX}}f + o$

Program

VAS

PA: $(f, o) = (3, 6)$

$(f_{MAX}, o_{MAX})$ (Program)

Physical Memory

$(0, 0)$

$(0, 0)$
Realizing Virtual Memory

- A process’s virtual address space is partitioned into equal sized pages
  - page \( = \text{page frame} \)

- A virtual address is a pair \( (p, o) \)
  - \( p \) — page number \( (2^{\log_2 p_{\text{MAX}}}) \)
  - \( o \) — page offset \( (2^{\log_2 o_{\text{MAX}}} \text{ bytes/pages}) \)
  - Virtual address \( = 2^{\log_2 o_{\text{MAX}}} p + o \)

VA: ____________________________
\( p \quad o \quad \log_2 p_{\text{MAX}} \quad \log_2 o_{\text{MAX}} \)
\( (p, o) \quad (0, 0) \)

Paging

Mapping virtual addresses to physical addresses

- Pages map to frames
  - Pages are contiguous in a VAS but are arbitrarily located in physical memory
  - Not all pages mapped at all times
Paging
Virtual address translation

CPU

Program

PO

Virtual Addresses

Physical Addresses

Page Table

Physical Addresses

f

Virtual Addresses

PTBR

CPU

Page Table

Paging
Page table structure

- 1 table per process
  - part of process’s state
- Contents:
  - flags — dirty bit, resident bit, clock bit
  - frame number
**Paging Example**

- A system with 16-bit addresses
- 32 KB of physical memory
- 1024 byte pages

**Physical Memory**

- CPU

**Virtual Addresses**

- (4, 0)
- (3, 1023)

**Page Table**

- (4, 1023)
- (0, 0)

**Virtual Memory Performance**

**Address translation caching**

- Problem — VM reference requires 2 memory references!
- Solution — Cache page-to-frame translations
  - Translation lookaside buffer

**CPU**

**Virtual Addresses**

- page 0
- page 1

**Physical Addresses**

- page 0
- page 1

**Page Table**

- Key Value

**TLB**

- Key

- Value
Page Fault Handling

- References to non-mapped pages generate a page fault

- Page fault handling
  » Service the fault
  » Read in the unmapped page
  » Restart the faulting process

Virtual Memory Performance
Page fault analysis

- How can VM possible work?!
  » Memory access time: 20 ns
  » Disk access time: 25 ms
  » Effective access time (EAT)
    - Let $p$ = the probability of a page fault
    - $EAT = 20(1-p) + 25,000,000p$
    - For an $EAT$ within 5% of minimum, $p \leq 0.000,000,04$
      (less than one fault every 25,000,000 references)

- Moral: OS had better do a good job of page replacement!