Contents

- Virtual Memory
- Paging
- Page Table
- Summary

Review

- Memory Manager
  - Monitor used and free memory
  - Allocate memory to processes
  - Reclaim (De-allocate) memory
  - Swapping between main memory and disk
- Mono-programming memory management
  - Overlay
- Multi-programming memory management
  - Fixed-sized partition
  - Variable-sized partition
  - Relocation and protection
- Swapping
Problems

- Programs are too big to fit in the available memory

Solutions

- Overlay
  - OS does swapping
  - Programmers split programs into overlays
- Virtual Memory

Virtual Memory

- Provide user with virtual memory that is as big as user needs
- Store virtual memory on disk
- Cache parts of virtual memory being used in real memory
- Load and store cached virtual memory without user program intervention
Benefits of Virtual Memory

- Use secondary storage
  - Extend RAM with reasonable performance
- Protection
  - Programs do not step over each other
- Convenience
  - Flat address space
  - Programs have the same view of the world
  - Load and store cached virtual memory without user program intervention
- Reduce fragmentation:
  - Make cacheable units all the same size (page)
- Remove memory deadlock possibilities:
  - Permit pre-emption of real memory

Paging

Request Page 3

Real Memory

Page Table

VM Frame

Virtual Memory Stored on Disk

Real Memory

Disk
Paging

Request Page 1

Memory

Page Table
VM Frame

Virtual Memory Stored on Disk

Disk

Real Memory

Paging

Request Page 6

Memory

Page Table
VM Frame

Virtual Memory Stored on Disk

Disk

Real Memory
Paging

Load Page 8 to Memory

Virtual Memory Stored on Disk

Real Memory

Page Mapping Hardware

Physical Memory

Virtual Memory
**Page Mapping Hardware**

- Virtual Address (004006)
- Page Table
  - 0
  - 1
  - 4→5
  - 1
  - 1
  - 0
  - 1

- Physical Address (F,D)
- Virtual Memory
  - 004
  - 006
- Physical Memory
  - 005
  - 006

- Page size 1000
- Number of Possible Virtual Pages 1000
- Number of Page Frames 8

---

**Paging Issues**

- **Page size is** $2^n$
  - usually 512 bytes, 1 KB, 2 KB, 4 KB, or 8 KB
  - E.g. 32 bit VM address may have $2^{20}$ (1 MB) pages with 4k ($2^{12}$) bytes per page

- **Page table**:
  - $2^{20}$ page entries take $2^{22}$ bytes (4 MB)
  - page frames must map into real memory
  - Page Table base register must be changed for context switch

- **No external fragmentation; internal fragmentation on last page only**
Multilevel Page Tables

- Since the page table can be very large, one solution is to page the page table.
- Divide the page number into:
  - An index into a page table of second level page tables
  - A page within a second level page table
- Advantage:
  - No need to keeping all the page tables in memory all the time
  - Only recently accessed memory’s mapping need to be kept in memory, the rest can be fetched on demand

What does this buy us? Sparse address spaces and easier paging
Example of 2-Level Page Table

- A logical address (on 32-bit x86 with 4k page size) is divided into
  - A page number consisting of 20 bits (> 1 page)
  - A page offset consisting of 12 bits

- Divide the page number into
  - A 10-bit page table page number ($p1$) (4byte/PTE)
  - A 10-bit page table offset ($p2$)

<table>
<thead>
<tr>
<th>page number</th>
<th>page offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p1$</td>
<td>$p2$</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

Example of 2-Level Page Table

- Logical address
  - $p1$, $p2$, $d$

  - $p1$ points to Directory
    - Directory contains pointers to
      - Page tables
        - Page tables store page offsets
  - $p2$ points to
    - Page tables

- $d$ points to page tables

CSE660: Introduction to Operating Systems
Multilevel Paging and Performance

Since each level is stored as a separate table in memory, memory reference with a three-level page table at least take four memory accesses. Why?

A Typical Page Table Entry

![Diagram of a page table entry with fields labeled: Caching disabled, Modified, Present/absent, Referenced, Protection, and Page frame number.]
Page Fault

- Access a virtual page that is not mapped into any physical page
  - A fault is triggered by hardware
- Page fault handler (in OS's VM subsystem)
  - Find if there is any free physical page available
    - If no, evict some resident page to disk (swapping space)
  - Allocate a free physical page
  - Load the faulted virtual page to the prepared physical page
  - Modify the page table

Summary

- Virtual Memory
- Paging
- Page Table

- Next lecture: Virtual Memory (II)