Roadmap

• Virtual Memory
  - Demand Paging
  - Page Faults
  - Page Replacement
  - Page Replacement Algorithms
Virtual Memory

- separation of user logical memory from physical memory.
  - Only part of the program needs to be in memory for execution.
  - Logical address space can therefore be much larger than physical address space.
  - Allows address spaces to be shared by several processes.
  - Allows for more efficient process creation.
Goals

- Make programmers job easier
  - Can write code without knowing how much DRAM is there
  - Only need to know general memory architecture
    - (e.g., 32-bit address space)

- Enable Multiprogramming
  - Keep several programs running concurrently
    - Together, these programs may need more DRAM than we have.
    - Keep just the actively used pages in DRAM.
  - Share when possible
    - When one program does I/O switch CPU to another.
How it works?
Implementation

• Virtual memory can be implemented via:
  - Demand paging
  - Demand segmentation
Demand Paging

• Bring a page into memory only when it is needed
  - Less I/O needed
  - Less memory needed
  - Faster response
  - More users

• Page is needed ⇒ reference to it
  - invalid reference ⇒ abort
  - not-in-memory ⇒ bring to memory
Valid-Invalid Bit

- With each page table entry a valid-invalid bit is associated
  \(1 \Rightarrow\) in-memory and legal, \(0 \Rightarrow\) not-in-memory or invalid
- Initially valid-invalid bit is set to 0 on all entries
- Example of a page table snapshot:

```
Frame # | valid-invalid bit
--- | ---
1 | 1
1 | 1
1 | 1
1 | 1
0 | 0
... | ...
0 | 0
```

- During address translation, if valid-invalid bit in page table entry is 0 \(\Rightarrow\) page fault
Page Table When Some Pages Are Not in Main Memory
Transfer of a Paged Memory to Contiguous Disk Space
Page Fault

- If there is ever a reference to a page not in memory, first reference will trap to OS ⇒ page fault
- OS looks at another table (in PCB) to decide:
  - Invalid reference ⇒ abort.
  - Just not in memory. ==> page-in
- Get an empty frame.
- Swap (read) page into the new frame.
- Set validation bit = 1.
- Restart instruction
Steps in Handling a Page Fault

1. Trap
2. Page is on backing store
3. Operating system
4. Bring in missing page
5. Reset page table
6. Restart instruction

Load M

Physical memory

Page table

Free frame

Reference
CPU tries to load a word

Page Table

Iw $20, 100($10)

Valid
Page 0
Page 1
Page N-1

Disk Address

Word not In DRAM

Operating System

O.S. initiates Disk I/O

Frame 0
Frame 1
Frame 2
Frame 3

Find an empty frame & Put the page there

Trap to O.S.

Restart the instruction

Update the Page Table

Update the Page Table
What happens if there is no free frame?

- Page replacement - find some page in memory, but not really in use, swap it out
  - Algorithms (FIFO, LRU ..)
  - performance - want an algorithm which will result in minimum number of page faults

- Same page may be brought into memory several times
Page Replacement

• Prevent over-allocation of memory by modifying page-fault service routine to include page replacement

• Use modify (dirty) bit to reduce overhead of page transfers - only modified pages are written to disk

• Page replacement completes separation between logical memory and physical memory - large virtual memory can be provided on a smaller physical memory
Basic Page Replacement

1. Find the location of the desired page on disk

2. Find a free frame:
   - If there is a free frame, use it
   - If there is no free frame, use a page replacement algorithm to select a victim frame

3. Read the desired page into the (newly) free frame. Update the page and frame tables.

4. Restart the process
Page Replacement

1. Swap out victim page
2. Change to invalid
3. Swap desired page in
4. Reset page table for new page
Page Replacement Algorithms

- Want lowest page-fault rate
- Evaluate algorithm by running it on a particular string of memory references (reference string) and computing the number of page faults on that string
- In all our examples, the reference string is 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
Graph of Page Faults Versus The Number of Frames
First-In-First-Out (FIFO) Algorithm

- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- 3 frames (3 pages can be in memory at a time per process)
First-In-First-Out (FIFO) Algorithm

- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- 3 frames (3 pages can be in memory at a time per process)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

9 page faults
First-In-First-Out (FIFO) Algorithm

- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- 3 frames (3 pages can be in memory at a time per process)

```
1  4  5
2  1  3  9 page faults
3  2  4
```

- 4 frames
First-In-First-Out (FIFO) Algorithm

- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- 3 frames (3 pages can be in memory at a time per process)

```
1 1 4 5
2 2 1 3
3 3 2 4
```

9 page faults

- 4 frames

```
1 1 5 4
2 2 1 5
3 3 2
4 4 3
```

10 page faults

- FIFO Replacement - Belady’s Anomaly
  - more frames ⇒ more page faults
FIFO Illustrating Belady’s Anomaly
Performance of Demand Paging

- Page Fault Rate $0 \leq p \leq 1.0$
  - if $p = 0$ no page faults
  - if $p = 1$, every reference is a fault

- Effective Access Time (EAT)
  \[
  EAT = (1 - p) \times \text{memory access} \\
  + p \times (\text{page fault overhead} \\
  + \text{[swap page out]} \\
  + \text{swap page in} \\
  + \text{restart overhead})
  \]
Demand Paging Example

- Memory access time = 1 microsecond
- 50% of the time the page that is being replaced has been modified and therefore needs to be swapped out
- Swap Page Time = 10 msec = 10,000 microsec
- EAT = ?
Demand Paging Example

- Memory access time = 1 microsecond

- 50% of the time the page that is being replaced has been modified and therefore needs to be swapped out

- Swap Page Time = 10 msec = 10,000 microsec

- EAT = \((1 - p) \times 1 + p \times (10,000 + \frac{1}{2} \times 10,000)\)
  = \(1 + 14,999 \times p\) (in microsec)

- What if 1 out of 1000 memory accesses cause a page fault?

- What if we only want 30% performance degradation?
Summary

• Virtual Memory
  - Demand Paging
  - Page Faults
  - Page Replacement
  - Page Replacement Algorithms
    - FIFO

• Next Lecture: Virtual Memory - II

• Reading Assignment: Chapter 9 from Silberschatz.
Acknowledgements


- “Modern Operating Systems” book and supplementary material by A. Tanenbaum

- R. Doursat and M. Yuksel from UNR