Virtual Memory
Virtual Memory Mechanisms

If page fault (i.e., present bit is cleared)
• Trap into OS (not handled by hardware. Why?)
• OS selects victim page in memory to replace
• Write victim page out to disk if modified. Add modified ("dirty") bit to PTE
• OS reads referenced page from disk into memory
• Page table is updated, present bit is set
• Process continues execution

What should scheduler do?
Mechanism for Continuing a Process

Continuing a process after a page fault is tricky

- Want page fault to be transparent to user
- Page fault may have occurred in middle of instruction
  - When instruction is being fetched
  - When data is being loaded or stored
- Requires hardware support
  - precise interrupts: stop CPU pipeline such that instructions before faulting instruction have completed, and those after can be restarted

Complexity depends upon instruction set

- Can faulting instruction be restarted from beginning?
  - Example: \texttt{move +(SP), R2}
  - Must track side effects so hardware can roll them back if needed
Virtual Memory Policies

Goal: Minimize number of page faults

- Page faults require milliseconds to handle (reading from disk)
- Implication: Plenty of time for OS to make good decision

OS has two decisions

- Page selection
  - **When** should a page (or pages) on disk be **brought into** memory?
- Page replacement
  - **Which** resident page (or pages) in memory should be **thrown out** to disk?
Average Memory Access Time (AMAT)

Hit% = portion of accesses that go straight to RAM
Miss% = portion of accesses that go to disk first
Tm = time for memory access
Td = time for disk access

AMAT = (Tm) + (Miss% * Td)
Page Selection

When should a page be brought from disk into memory?

**Demand paging:** Load page only when page fault occurs

- Intuition: Wait until page must absolutely be in memory
- When process starts: No pages are loaded in memory
- Problems: Pay the cost of a page fault for every newly accessed page
Page Selection

When should a page be brought from disk into memory?

Pre-paging (anticipatory, prefetching): Load page before referenced

- OS predicts future accesses (oracle) and brings pages into memory early
- Works well for some access patterns (e.g., sequential)
- Problems?
Page Selection

When should a page be brought from disk into memory?

Hints: Combine above with user-supplied hints about page references

- User specifies: may need page in future, don’t need this page anymore, or sequential access pattern, ...
- Example: `madvise()` in Unix
Page Replacement

Which page in main memory should selected as victim?
  • Write out victim page to disk if modified (“dirty” bit set)
  • If victim page is not modified (clean), just discard

OPT: Replace page not used for longest time in future
  • Advantages: Guaranteed to minimize number of page faults
  • Disadvantages: Requires that OS predict the future;
    Not practical, but good for comparison
OPT Replacement Example

Page reference string: 1, 2, 3, 1, 2, 4, 1, 4, 2, 3, 2

Miss: 1, 2, 3

Three pages of physical memory

Metric: Miss count
OPT Replacement Example

Page reference string: 1,2,3,1,2,4,1,4,2,3, 2

Miss: 1,2,3

Metric:
Miss count : 3

Three pages of physical memory
OPT Replacement Example

Page reference string: 1,2,3,1,2,4,1,4,2,3, 2

Miss: 1,2,3

OPT

Hit 1

Miss count: 3

Hit 2

Three pages of physical memory

Compulsory misses
OPT Replacement Example

Page reference string: 1,2,3,1,2,4,1,4,2,3, 2

Miss: 1,2,3
Hit 1: 1 2 3
Hit 2: 1 2 3
Miss: 4, Replace: 3
Hit 1: 1 2 4

Hit 1: 1 2 3

Three pages of physical memory

Metric:
Miss count: 4

capacity miss
OPT Replacement Example

Page reference string: 1,2,3,1,2,4,1\textbf{4,2},3, 2

<table>
<thead>
<tr>
<th>Miss: 1,2,3</th>
<th>OPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3</td>
<td></td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Miss:4, Replace: 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 4</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Hit: 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hit: 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 4</td>
</tr>
</tbody>
</table>

Metric:
Miss count: 4

Three pages of physical memory
## OPT Replacement Example

### Page reference string: 1,2,3,1,2,4,1,4,2,3,2

<table>
<thead>
<tr>
<th>Miss: 1,2,3</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hit 1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Hit 2</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Miss: 4, Replace: 3</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Hit 1</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Hit: 4</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Hit: 2</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Miss: 3, Replace: 1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Hit: 2</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

**Metric:**
- AMAT?
- Miss count: 5

5 misses, 4 compulsory misses

### AMAT = (Tm) + (Miss% * Td)

Assume Tm = 100ns
Assume Td = 1000000 ns (1millisecond)

AMAT = ?

### Three pages of physical memory
FIFO

FIFO: Replace page that has been in memory the longest

- Intuition: First referenced long time ago, done with it now
- Advantages: Fair: All pages receive equal residency; Easy to implement (circular buffer)
- Disadvantage: Some pages may always be needed
FIFO Example

Page reference string: {1,2,3,1,2,4,1,4,2,3,2}

OPT

Miss: 1,2,3

Metric:
Miss count: 3

Three pages of physical memory
**FIFO Example**

Page reference string: 1,2,3,1,2,4,1,4,2,3,2

**OPT**

Miss: 1,2,3

Hit: 1

Hit: 2

Miss: 4, Replace: 1

Metric:
Miss count: 4

Three pages of physical memory
FIFO Example

Page reference string: 1,2,3,1,2,4,1,4,2,3,2

Three pages of physical memory

Metric:
Miss count: 5
FIFO Example

Page reference string: 1,2,3,1,2,4,1,4,2,3,2

Three pages of physical memory

OPT

Miss: 1,2,3

Hit: 1

Hit: 2

Miss:4, Replace:1

Miss:1, Replace:2

Hit: 4

Miss:2, Replace:3

Miss:3, Replace:4

Metric:
Miss count : 7
FIFO Example

Page reference string: 1,2,3,1,2,4,1,4,2,3,2

Three pages of physical memory

<table>
<thead>
<tr>
<th>Miss: 1,2,3</th>
<th>OPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3</td>
<td></td>
</tr>
<tr>
<td>Hit: 1</td>
<td>1</td>
</tr>
<tr>
<td>1 2 3</td>
<td></td>
</tr>
<tr>
<td>Hit: 2</td>
<td>1</td>
</tr>
<tr>
<td>1 2 3</td>
<td></td>
</tr>
<tr>
<td>Miss: 4, Replace: 1</td>
<td>2 3 4</td>
</tr>
<tr>
<td>Hit: 4</td>
<td>3</td>
</tr>
<tr>
<td>3 4 1</td>
<td></td>
</tr>
<tr>
<td>Miss: 2, Replace: 3</td>
<td>4 1 2</td>
</tr>
<tr>
<td>Hit: 2</td>
<td>1</td>
</tr>
<tr>
<td>1 2 3</td>
<td></td>
</tr>
</tbody>
</table>

Metric:
Miss count : 7
FIFO Example

Page reference string: 1,2,3,1,2,4,1,4,2,3,2

OPT

Miss: 1,2,3

Hit: 1

Hit: 2

Miss: 4, Replace: 1

Miss: 1, Replace: 2

Hit: 4

Miss: 2, Replace: 3

Miss: 3, Replace: 4

Hit: 2

Three pages of physical memory

7 total misses, 4 compulsory misses

AMAT = (Tm) + (Miss% * Td)

Assume Tm = 100ns
Assume Td = 1000000 ns (1millisecond)

AMAT = ?
LRU Example – Replace

Least Recently Used

Page reference string: 1,2,3,1,2,4,1,4,2,3,2

Three pages of physical memory

OPT

Miss: 1,2,3

Hit: 1

Hit: 2

Miss: 4, Replace: 3

Hit: 1

Hit: 4

Hit: 2

Miss: 3, Replace: 1

Hit: 2

Metric: 5 total misses

Miss count: 4 compulsory misses

In this example, same as OPT!