Page replacement

- Not all virtual pages can be in physical memory
  - Swap in page \( \rightarrow \) Need to evict another page
- Common page replacement policy: LRU
  - Works well because of temporal locality
- LRU is hard to implement precisely
  - How to simplify LRU by approximating it?

Clock replacement algorithm

- Most MMUs maintain a “referenced” bit for each resident page
  - Set by MMU when page is read or written
  - Can be cleared by OS
- How to use reference bit to identify old pages?

Clock replacement algorithm

- Arrange resident pages around a clock
  - Arrangement: F, E, D, B, A
- Algorithm to select page for eviction:
  - Consider page pointed to by clock hand
    - If not referenced, page not accessed since last sweep \( \rightarrow \) Evict
    - If referenced, page has been accessed since last sweep
      - What to do?
- What if all pages have been referenced since last sweep?
- What about new pages?

Page eviction

- Where to evict page to?
- When do you NOT need to write page to disk?
  - Rely on hardware/MMU to maintain dirty bit in PTE
- Why not write to disk on every store?
- When do you save by not writing on every store?
- How can you optimize eviction?

Page table contents

```
Written by OS, Read by MMU  Written by OS/MMU
Resident  Protection  Dirty  Referenced
```

Physical page \#  Resident  Protection  Dirty  Referenced

if (virtual page is non-resident or protected) {
  trap to OS fault handler
  retry access
} else {
  physical page \# = pageTable[virtual page \#].physPageNum
  pageTable[virtual page \#].referenced = true
  if (access is write) {
    pageTable[virtual page \#].dirty = true
  }
  access physical memory
}
**Page table contents**

<table>
<thead>
<tr>
<th>Physical page #</th>
<th>Resident</th>
<th>Protection</th>
<th>Dirty</th>
<th>Referenced</th>
</tr>
</thead>
<tbody>
<tr>
<td>● Why no valid bit in PTE?</td>
<td>All invalid virtual pages are non-resident</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>● For valid non-resident pages, does PTE contain disk block?</td>
<td>OS must maintain this, MMU simply traps to OS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>● How can we make do without resident bit?</td>
<td>Use protection bits</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Project 3

- Covered all material you need to know to do the project
- We’ll discuss testing strategies next week

**Page table contents**

<table>
<thead>
<tr>
<th>Physical page #</th>
<th>Protection</th>
<th>Dirty</th>
<th>Referenced</th>
</tr>
</thead>
<tbody>
<tr>
<td>● Can we make do without dirty bit?</td>
<td>Use protection bits</td>
<td></td>
<td></td>
</tr>
<tr>
<td>● Can we make do without referenced bit?</td>
<td>Application too may want to control protection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>● Application too may want to control protection</td>
<td>Not in project 3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>