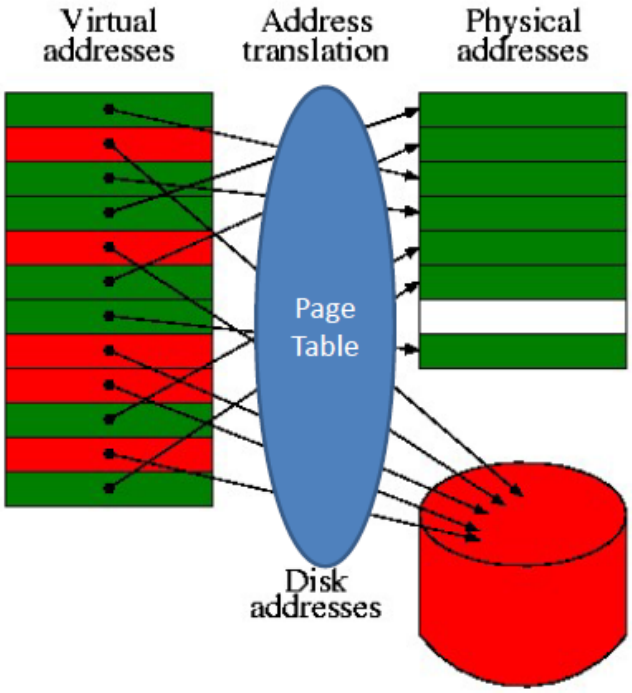
Operating Systems W10L2 - Memory Management (ctd)

Class	Operating Systems
Created	@Nov 4, 2020 7:10 PM
Materials	
Reviewed	☐
Type	Lecture

Page Faults

▼ Virtual to physical translation occurs in page table



- If memory is full, what do we remove? This is what leads to a page fault

Replacement Policies

- These show up everywhere in all different contexts
- We want to consider...
  1. Measure of success
  2. Cost
- An optimal algorithm entails knowing the future, and mathematically it's perfect, but we do not know the future
- Not Recently Used (NRU) Algorithm
  - Each page has two status bits: referenced/used (R) and written (W) — Bits maintained/set by the OS
  - Bits are *updated* by hardware and *reset* only by OS
  - R bit is periodically cleared → Not the M bit because we need to know when a page is written (thus if we can/cannot overwrite it)

    - Removes page at random from the lowest numbered non-empty class
- FIFO Replacement
    - Easy to implement, worse performance though
    - Not at all close to the optimal prediction → First in page might not be the one not used in the future
- Second-Chance Replacement
    - Modified FIFO
    - Inspect R-bit of oldest page → If unused (R=0) then replace, otherwise (R=1) page gets added to end of list
    - If all pages have R=1, then we just degenerate to FIFO
- Least Recently Used (LRU) Algorithm
    - Good approximation to optimal algorithm
    - At page fault, replace page that has remained unused the longest
    - Implementing it is not particularly straightforward → We want to have a proper data structure

LRU Implementation

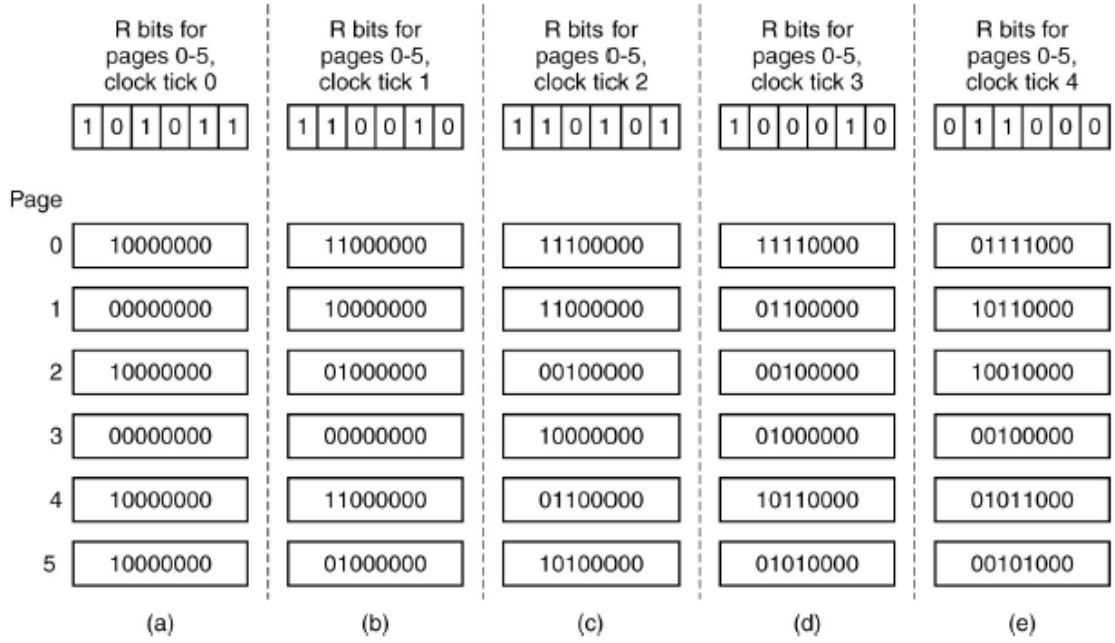
- Hardware Approach 1
    - Uses 64-bit counter incrementing after each instruction
    - Way too expensive and way too slow → Consequence of searching
- Hardware Approach 2
    - Given `n` page frames, a matrix of `n x n` is created and initialized to all zeros
    - When page frame `k` is referenced, set bits of row `k` to 1 and column `k` to 0 → Row with lowest value is least recently used
- Simulating in Software
    - Based on the Not Frequently Used (NFU) Algorithm
    - There is a software counter associated each page (starting at 0)

- At each clock interrupt, OS adds R bit of each page to the counter of the page
- At page fault, the page with the *lowest* counter is replaced

NRU Algorithm

- Enhancing NRU
  - **High Inertia:** Never forgets anything
  - Shifting counter right 1 bit before adding R to leftmost
  - The above modification is called *agin* → Lowest counter replaced at page fault

▼ Age Algorithm Diagram



Working Set Model

▼ What is the **working set**?

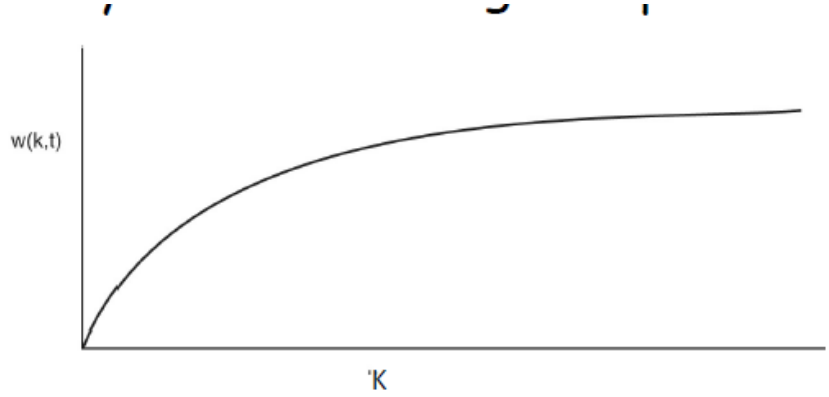
The set of pages that a process is currently using. Constantly changes.

▼ What is **thrashing**?

A program causing page faults every few instructions.

- Important question lies in knowing what pages to bring back

▼ `w(k, t)`



w(k,t): the set of pages accessed in the last k references at instant t

- OS keeps track of pages in the working set → The replacement algorithm evicts pages not in this workng set
- Again, idea clear but how to implement... not so much
  1. Possible Implementation: Working set is the set of pages accessed in the last `k` memory references
  2. Approximations: Working set is all the pages used in the last 100 msec