# Operating Systems Design (CS 423)

Elsa L Gunter
2112 SC, UIUC
http://www.cs.illinois.edu/class/cs423/

Based on slides by Roy Campbell, Sam King, and Andrew S Tanenbaum

## Entry Layouts form Intel Manual

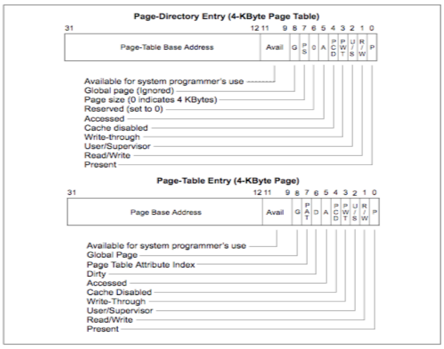
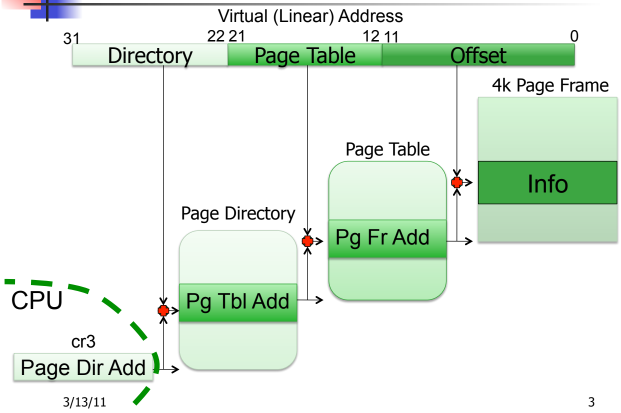
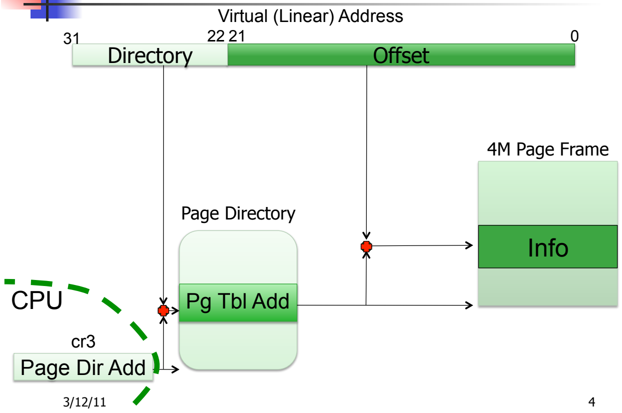


**Figure 3-14. Format of Page-Directory and Page-Table Entries for 4-KByte Pages and 32-Bit Physical Addresses**

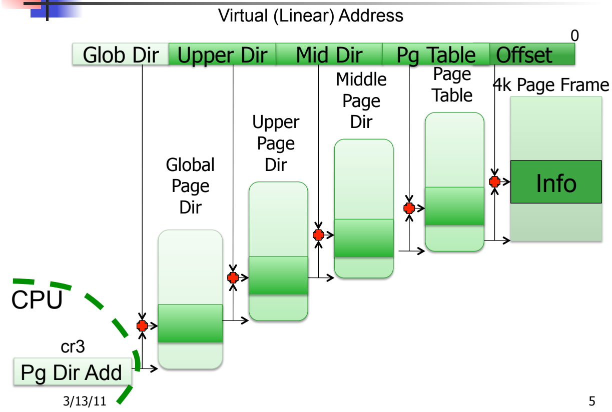
## Translating 32 bit Virtual Address



## Translating 32 bit Virtual Address



## Translating 64 bit Virtual Address



## Translation Lookaside Buffer

- Problem: Accessing a memory location assoc with an instruction requires at least 3 memory accesses
  - Page directory entry
  - Page table entry
  - Info in memory location
- All instruction need at least 1, but up to three memory accesses
- Translates into 9 accesses IF all is currently in memory
- Answer: small cache of page table entries - TLB

## Page Dealloaction

- Allocating memory fine so long as free memory exists
- Deallocating pages when process exits straightforward
- Problem: What to do when memory full of pages for active processes and more memory is needed?
- Answer: you have to pick something resident to kick out, but which one?

## Page Replacement Agolrithms

- Optimal page replacement algorithm (fiction)
- Not recently used
- First In, First Out
- Second Chance
- Clock
- Least Recently Used
- Working Set
- Working Set Clock

## Second Chance Algorithm

- Start by storing pages FIFO

| Load time | 0 | 2 | 5 | 6 | 9 | 11 | 15 | 17 | |
|---|---|---|---|---|---|---|---|---|---|
| Oldest page | A | B | C | D | E | F | G | H | Newest page |

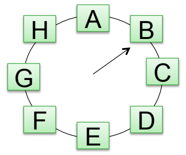
- When need to dealloacte, if A has access bit set, reset bit and move A to new of FIFO

| Load time | 2 | 5 | 6 | 9 | 11 | 15 | 17 | 20 | |
|---|---|---|---|---|---|---|---|---|---|
| Oldest page | B | C | D | E | F | G | H | A | Newest page |

- Keep cycling; take first with access bit reset
  - May be A

## Clock Replacement Algorithm

- Modification of Second Chance
- Keep circular list, point to current candidate
- When page frame needed, inspect page pointed to
- If access bit reset, replace it
- If access bit set, reset and advance pointer



## Working Set Algorithm

- Observation: Locally processes only use small subset of their pages
- Set currently being used called working set
- Objective: reduce page fault rate by keeping working set in memory
- When process swapped out, pages replaced
  - Record pages present and recently referenced before replacing (working set)
- When process swapped in, prepage - replace old working set

## Working Set

- Definition: Virtual time $t$ of process $p$ is the amount of execution time (CPU time) of $p$
- Definition: Working set of process $p$ and $k$ memory accesses at virtual time $t$ is the number of $p$-pages referenced in the last $k$ memory accesses before virtual time $t$
  - Written $w_p(k,t)$
  - $w_p(k,t) \leq w_p(k+n,t)$
- Hard to compute; replace mem access by virt time

## Working Set Algorithm

- Choose medium-sized *k*; exact size not important
- At clock interrupt, all pages have reference bit reset
- At page fault, scan all (resident) pages
  - If Accessed bit set, record current virtual time (*CVT)* as *time of last use (TLU)*
  - If Accessed bit reset, and *TLU – CVT > k,* evict
  - If Accessed bit reset, and *TLU– CVT ≤ k,* keep looking
  - If no page with Accessed bit reset has *TLU – CVT > k,* evict oldest
  - If all pages have Accessed bit set, evict one randomly

## Working Set Clock

- Circular list of page info with *TLU, Accessed, Modifed*
- At page fault, if page pointed to is *Accessed*, reset *Accessed* and move pointer ahead
- If not *Accessed* and age *> k*,
  - Not *Modifed*, then evict
  - *Modifed*, then schedule write-to-disk, advance pointer
- If return to start
  - If write scheduled, keep going until one is clean
  - No write scheduled, evict randon page