



COS 318: Operating Systems

Virtual Memory Paging



Today's Topics

- ◆ Paging mechanism
- ◆ Page replacement algorithms



Virtual Memory Paging

◆ Simple world

- Load entire process into memory. Run it. Exit.

◆ Problems

- Slow (especially with big processes)
- Wasteful of space (doesn't use all of its memory all the time)
- Reduces number of processes ready to run at a time

◆ Solution

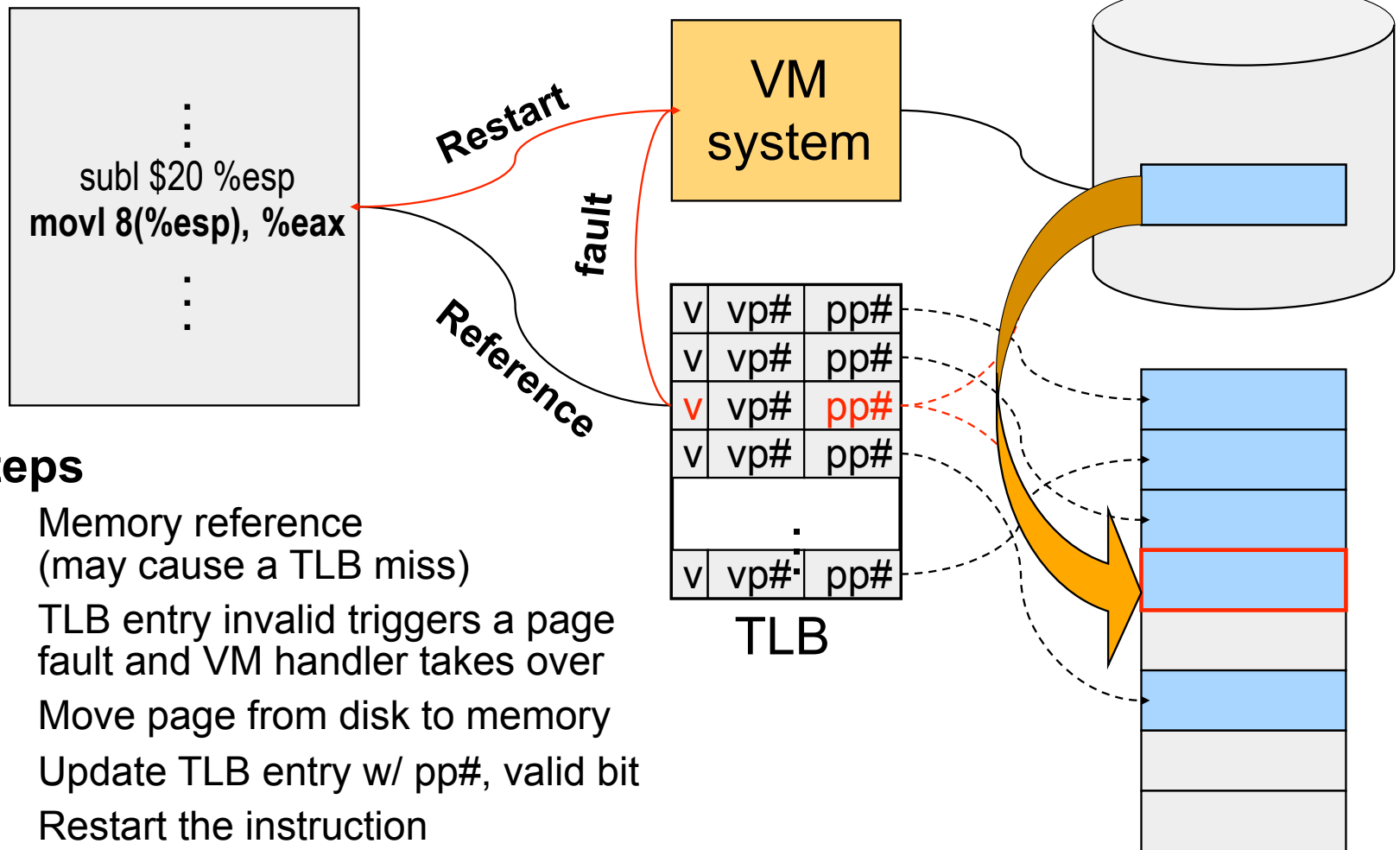
- Demand paging: only bring in pages actually used
- Paging: only keep frequently used pages in memory

◆ Mechanism:

- Programs refer to addresses in virtual address space
- Virtual memory maps some to physical pages in memory, some to disk



VM Paging Steps



Steps

- ◆ Memory reference (may cause a TLB miss)
- ◆ TLB entry invalid triggers a page fault and VM handler takes over
- ◆ Move page from disk to memory
- ◆ Update TLB entry w/ pp#, valid bit
- ◆ Restart the instruction
- ◆ Memory reference again

VP# = virtual page no.
PP# = physical page no.

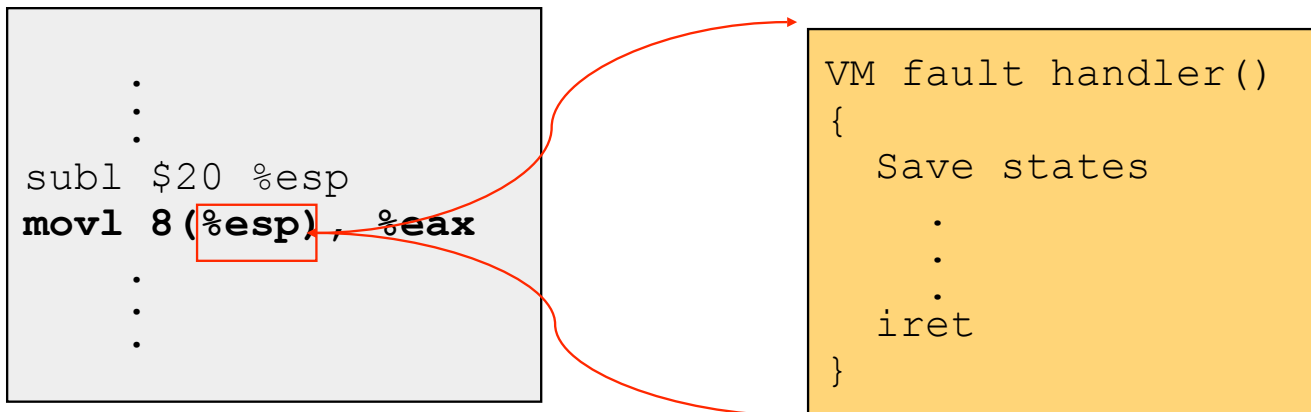


Virtual Memory Issues

- ◆ How to switch a process after a fault?
 - Need to save state and resume
 - Is it the same as an interrupt?
- ◆ What to page in?
 - Just the faulting page or more?
 - Want to know the future...
- ◆ What to replace?
 - Memory is a software-managed cache on disk
 - Caches always too small, which page to replace?
 - Want to know the future...



How Does Page Fault Work?



- ◆ User program should not be aware of the page fault
- ◆ Fault may have happened in the middle of the instruction!
- ◆ Can we skip the faulting instruction?
- ◆ Is a faulting instruction always restartable?

What to Page In?

- ◆ Page in the faulting page
 - Simplest, but each “page in” has substantial overhead
- ◆ Page in more pages each time
 - May reduce page faults if the additional pages are used
 - Waste space and time if they are not used
 - Real systems do some kind of prefetching
- ◆ Applications control what to page in
 - Some systems support for user-controlled prefetching
 - But, many applications do not always know



VM Page Replacement

- ◆ Things are not always available when you want them
 - It is possible that no unused page frame is available
 - VM needs to do page replacement
- ◆ On a page fault
 - If there is an unused frame, get it
 - **If no unused page frame available,**
 - **Find a used page frame**
 - **If it has been modified, write it to disk**
 - **Invalidate its current PTE and TLB entry**
 - Load the new page from disk
 - Update the faulting PTE and remove its TLB entry
 - Restart the faulting instruction
- ◆ General data structures
 - A list of unused page frames
 - A table to map page frames to PID and virtual pages, why?

**Page
Replacement**



Which “Used” Page Frame To Replace?

- ◆ Random
- ◆ Optimal or MIN algorithm
- ◆ NRU (Not Recently Used)
- ◆ FIFO (First-In-First-Out)
- ◆ FIFO with second chance
- ◆ Clock
- ◆ LRU (Least Recently Used)
- ◆ NFU (Not Frequently Used)
- ◆ Aging (approximate LRU)
- ◆ Working Set
- ◆ WSClock



Optimal or MIN

◆ Algorithm:

- Replace the page that won't be used for the longest time
(Know all references in the future)

◆ Example

- Reference string: 1 2 3 4 1 2 5 1 2 3 4 5
- 4 page frames
- 6 faults

◆ Pros

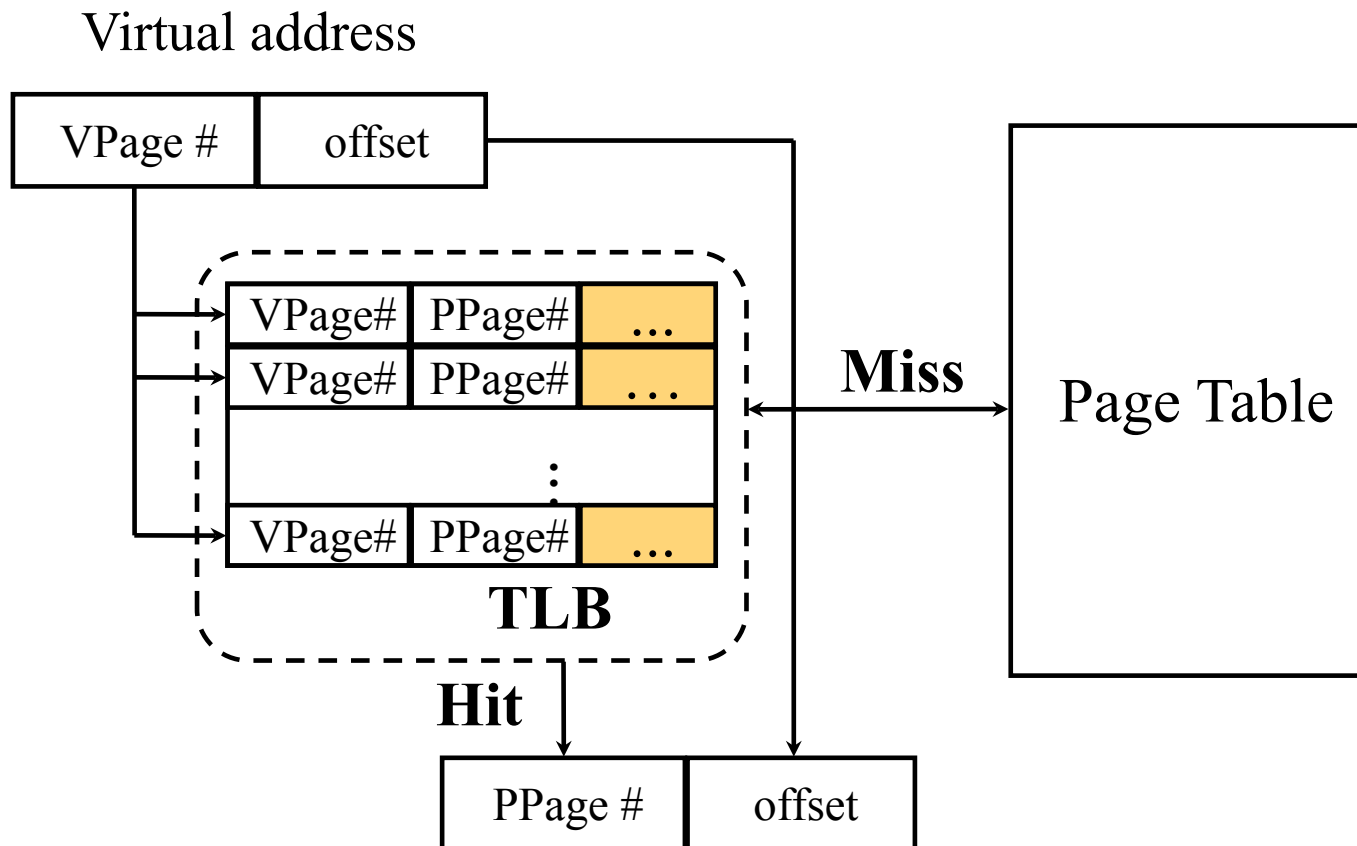
- Optimal solution and can be used as an off-line analysis method

◆ Cons

- No on-line implementation



Revisit TLB and Page Table



◆ Important bits for paging

- **Reference:** Set when referencing a location in the page
- **Modify:** Set when writing to a location in the page



Not Recently Used (NRU)

◆ Algorithm

- Randomly pick a page from lowest-numbered non-empty class below
 - Not referenced and not modified
 - Not referenced and modified (huh?)
 - Referenced and not modified
 - Referenced and modified
- Clear reference bits periodically (e.g. at clock interrupts)

◆ Example

- 4 page frames
- Reference string
- 8 page faults

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

◆ Pros

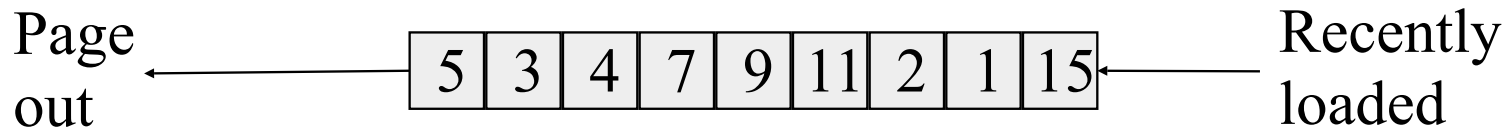
- Implementable

◆ Cons

- Require scanning through reference bits and modified bits



First-In-First-Out (FIFO)



◆ Algorithm

- Throw out the oldest page

◆ Example

- 4 page frames
- Reference string
- 10 page faults

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

◆ Pros

- Low-overhead implementation

◆ Cons

- May replace the heavily used pages



More Frames → Fewer Page Faults?

- ◆ Consider the following with 4 page frames

- Algorithm: FIFO replacement
- Reference string:
- 10 page faults

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

- ◆ Same string with 3 page frames

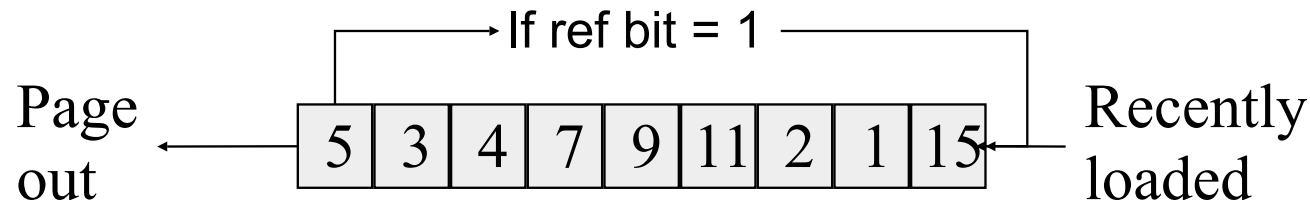
- Algorithm: FIFO replacement
- Reference string:
- **9 page faults!**

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

- ◆ This is so called “Belady’s anomaly” (Belady, Nelson, Shedler 1969)



FIFO with 2nd Chance



◆ Algorithm

- Check the reference-bit of the oldest page
- If it is 0, then replace it
- If it is 1, clear the referent-bit, put it to the end of the list (as if it had just been loaded), and continue searching

◆ Example

- 4 page frames
- Reference string:
- 8 page faults

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

◆ Pros

- Simple to implement

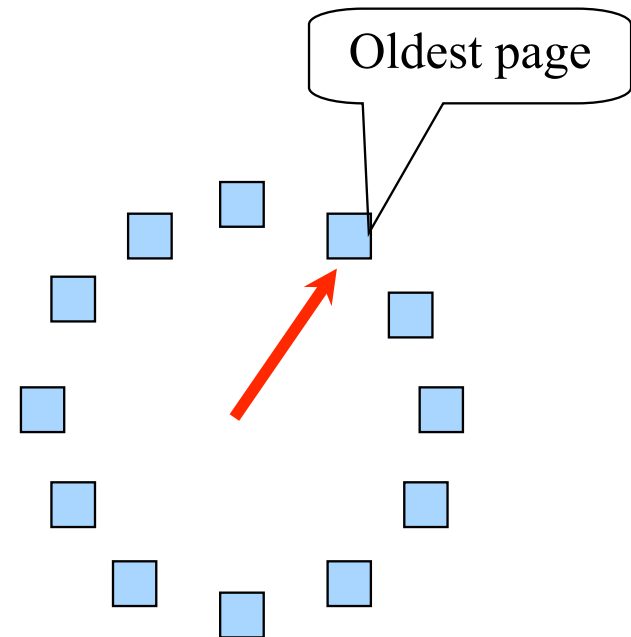
◆ Cons

- The worst case may take a long time (moving pages around on the list)

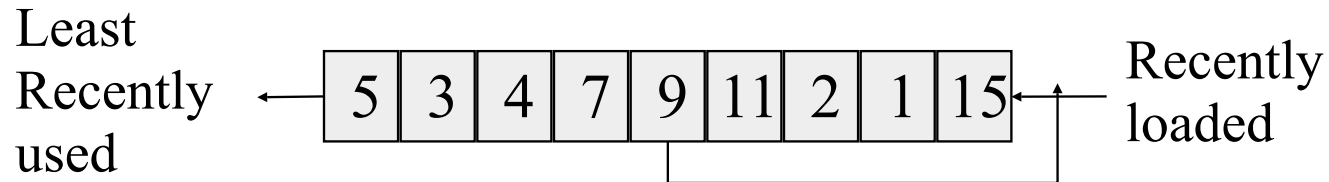


Clock

- ◆ FIFO clock algorithm
 - Hand points to the oldest page
 - On a page fault, follow the hand to inspect pages
- ◆ Second chance
 - If the reference bit is 0, use it for replacement, new page replaces it, advance the hand
 - If the reference bit is 1, set it to 0 and advance the hand



Least Recently Used



◆ Algorithm

- Replace page that hasn't been used for the longest time
 - Order the pages by time of reference
 - Timestamp for each referenced page

◆ Example

- 4 page frames
- Reference string:
- 8 page faults

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

◆ Pros

- Good to approximate MIN

◆ Cons

- Expensive: maintain list of pages by reference, update on every reference



Aging: Not Frequently Used (NFU)

◆ Algorithm

- Shift reference bits into counters at every clock interrupt
- Pick the page with the smallest counter to replace

00000000	00000000	10000000	01000000	10100000
00000000	10000000	01000000	10100000	01010000
10000000	11000000	11100000	01110000	00111000
00000000	00000000	00000000	10000000	01000000

◆ Old example

- 4 page frames
- Reference string:
- 8 page faults

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

◆ Main difference between NFU and LRU?

- NFU can't distinguish LRU within a clock interrupt period
- NFU has a short history (counter length)

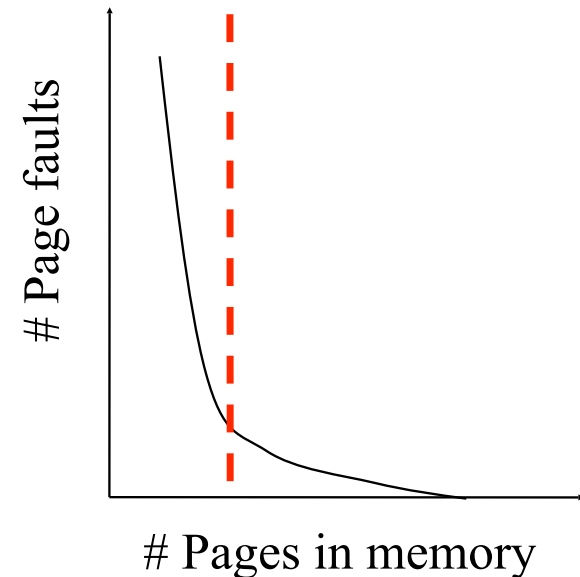
◆ How many bits are enough?

- In practice 8 bits are quite good



Program Behavior (Denning 1968)

- ◆ 80/20 rule
 - > 80% memory references are within <20% of memory space
 - > 80% memory references are made by < 20% of code
- ◆ Spatial locality
 - Neighbors are likely to be accessed
- ◆ Temporal locality
 - The same page is likely to be accessed again in the near future



Working Set

- ◆ Main idea (Denning 1968, 1970)
 - Define a working set as the set of pages in the most recent K page references
 - Keep the working set in memory will reduce page faults significantly
- ◆ Approximate working set
 - The set of pages of a process used in the last T seconds
- ◆ An algorithm
 - On a page fault, scan through all pages of the process
 - If the reference bit is 1, record the current time for the page
 - If the reference bit is 0, check the “time of last use,”
 - If the page has not been used within T , replace the page
 - Otherwise, go to the next
 - Add the faulting page to the working set



WSClock

- ◆ Follow the clock hand
- ◆ If the reference bit is 1
 - Set reference bit to 0
 - Set the current time for the page
 - Advance the clock hand
- ◆ If the reference bit is 0, check “time of last use”
 - If the page has been used within δ , go to the next
 - If the page has not been used within δ and modify bit is 1
 - Schedule the page for page out and go to the next
 - If the page has not been used within δ and modify bit is 0
 - Replace this page



Replacement Algorithms

◆ The algorithms

- Random
- Optimal or MIN algorithm
- NRU (Not Recently Used)
- FIFO (First-In-First-Out)
- FIFO with second chance
- Clock
- LRU (Least Recently Used)
- NFU (Not Frequently Used)
- Aging (approximate LRU)
- Working Set
- WSClock

◆ Which are your top two?



Summary

- ◆ VM paging
 - Page fault handler
 - What to page in
 - What to page out
- ◆ LRU is good but difficult to implement
- ◆ Clock (FIFO with 2nd hand) is considered a good practical solution
- ◆ Working set concept is important
- ◆ Aging and WSClock do quite well and are implementable

