Memory

Virtual Memory: Paging

At last we can talk about paging

Memory

Virtual Memory: Paging

At last we can talk about paging

Paging is copying pages to and from disk

Memory

Virtual Memory: Paging

At last we can talk about paging

Paging is copying pages to and from disk

Suppose there is a memory access

Memory

Virtual Memory: Paging

At last we can talk about paging

Paging is copying pages to and from disk

Suppose there is a memory access

If there is a TLB hit, the memory access continues at full speed

Memory

Virtual Memory: Paging

At last we can talk about paging

Paging is copying pages to and from disk

Suppose there is a memory access

If there is a TLB hit, the memory access continues at full speed

On a TLB soft miss the usual case is that the page is still resident in physical memory (not been swapped out), so it's just a matter of updating the TLB to refer to it by copying the page table entry into the TLB

Memory

Virtual Memory: Paging

At last we can talk about paging

Paging is copying pages to and from disk

Suppose there is a memory access

If there is a TLB hit, the memory access continues at full speed

On a TLB soft miss the usual case is that the page is still resident in physical memory (not been swapped out), so it's just a matter of updating the TLB to refer to it by copying the page table entry into the TLB

And then the access can continue as for a hit

Memory

Virtual Memory: Paging

But if the page has been swapped out (“paged out”), then its contents need to be read back from disk: thus a large cost in this case

Memory

Virtual Memory: Paging

If there is a TLB miss when the TLB table is full, the OS must choose which mapping to remove from the TLB to make space for the new one

Memory

Virtual Memory: Paging

If there is a TLB miss when the TLB table is full, the OS must choose which mapping to remove from the TLB to make space for the new one

Usually a *least recently used* (LRU) strategy is used as pages that haven't been touched recently often are not needed in the near future: this is called *temporal locality*

Memory

Virtual Memory: Paging

If there is a TLB miss when the TLB table is full, the OS must choose which mapping to remove from the TLB to make space for the new one

Usually a *least recently used* (LRU) strategy is used as pages that haven't been touched recently often are not needed in the near future: this is called *temporal locality*

Thus the TLB hardware must also keep track of *when* pages are used, e.g., using a timestamp

Memory

Virtual Memory: Paging

If there is a TLB miss when the TLB table is full, the OS must choose which mapping to remove from the TLB to make space for the new one

Usually a *least recently used* (LRU) strategy is used as pages that haven't been touched recently often are not needed in the near future: this is called *temporal locality*

Thus the TLB hardware must also keep track of *when* pages are used, e.g., using a timestamp

Again, temporal locality is a feature of many programs, but it is easy to write programs that confound the LRU strategy

Memory

Virtual Memory: Paging

The OS may use the opportunity of a TLB miss to choose some pages to swap out

Memory

Virtual Memory: Paging

The OS may use the opportunity of a TLB miss to choose some pages to swap out

Note there are two separate issues here:

Memory

Virtual Memory: Paging

The OS may use the opportunity of a TLB miss to choose some pages to swap out

Note there are two separate issues here:

- which entry in the TLB to remove when the TLB table is full

Memory

Virtual Memory: Paging

The OS may use the opportunity of a TLB miss to choose some pages to swap out

Note there are two separate issues here:

- which entry in the TLB to remove when the TLB table is full
- which page in physical memory to swap out when physical memory is full

Memory

Virtual Memory: Paging

The OS may use the opportunity of a TLB miss to choose some pages to swap out

Note there are two separate issues here:

- which entry in the TLB to remove when the TLB table is full
- which page in physical memory to swap out when physical memory is full

They are related in that an infrequently used TLB entry implies an infrequently used page; but conversely an infrequently used page is probably not in the TLB table at all

Paging Strategies

Many strategies exist for choosing pages to evict (swap out)

Paging Strategies

Many strategies exist for choosing pages to evict (swap out)

- Random. Pick a random page. Simple and better than you think

Paging Strategies

Many strategies exist for choosing pages to evict (swap out)

- Random. Pick a random page. Simple and better than you think
- FIFO. First in first out. Poor, as pages that have been around for a long time tend to be the ones that are needed

Paging Strategies

Many strategies exist for choosing pages to evict (swap out)

- Random. Pick a random page. Simple and better than you think
- FIFO. First in first out. Poor, as pages that have been around for a long time tend to be the ones that are needed
- LRU. Least Recently Used. Good, but needs to keep track on a when a page was last used (different from the TLB page timestamp)

Paging Strategies

Many strategies exist for choosing pages to evict (swap out)

- Random. Pick a random page. Simple and better than you think
- FIFO. First in first out. Poor, as pages that have been around for a long time tend to be the ones that are needed
- LRU. Least Recently Used. Good, but needs to keep track on a when a page was last used (different from the TLB page timestamp)
- LFU. Least frequently used. Increment a counter on the page on each access; remove pages with low counts. Not so good as pages just brought in because you need them tend to have low counts

Paging Strategies

Many strategies exist for choosing pages to evict (swap out)

- Random. Pick a random page. Simple and better than you think
- FIFO. First in first out. Poor, as pages that have been around for a long time tend to be the ones that are needed
- LRU. Least Recently Used. Good, but needs to keep track on a when a page was last used (different from the TLB page timestamp)
- LFU. Least frequently used. Increment a counter on the page on each access; remove pages with low counts. Not so good as pages just brought in because you need them tend to have low counts
- And so on.

Memory

Virtual Memory

The first time a (virtual) page is touched by a process it will cause a (major) page fault

Memory

Virtual Memory

The first time a (virtual) page is touched by a process it will cause a (major) page fault

The OS needs to allocate a new physical page for this process

Memory

Virtual Memory

The first time a (virtual) page is touched by a process it will cause a (major) page fault

The OS needs to allocate a new physical page for this process

Allocation of new pages is facilitated by the fixed page size: just find *any* unallocated page in physical memory and set it as the physical page mapping from the requested virtual page

Memory

Virtual Memory

The first time a (virtual) page is touched by a process it will cause a (major) page fault

The OS needs to allocate a new physical page for this process

Allocation of new pages is facilitated by the fixed page size: just find *any* unallocated page in physical memory and set it as the physical page mapping from the requested virtual page

This simplification over earlier allocation methods is the big benefit of using pages

Memory

Virtual Memory

The first time a (virtual) page is touched by a process it will cause a (major) page fault

The OS needs to allocate a new physical page for this process

Allocation of new pages is facilitated by the fixed page size: just find *any* unallocated page in physical memory and set it as the physical page mapping from the requested virtual page

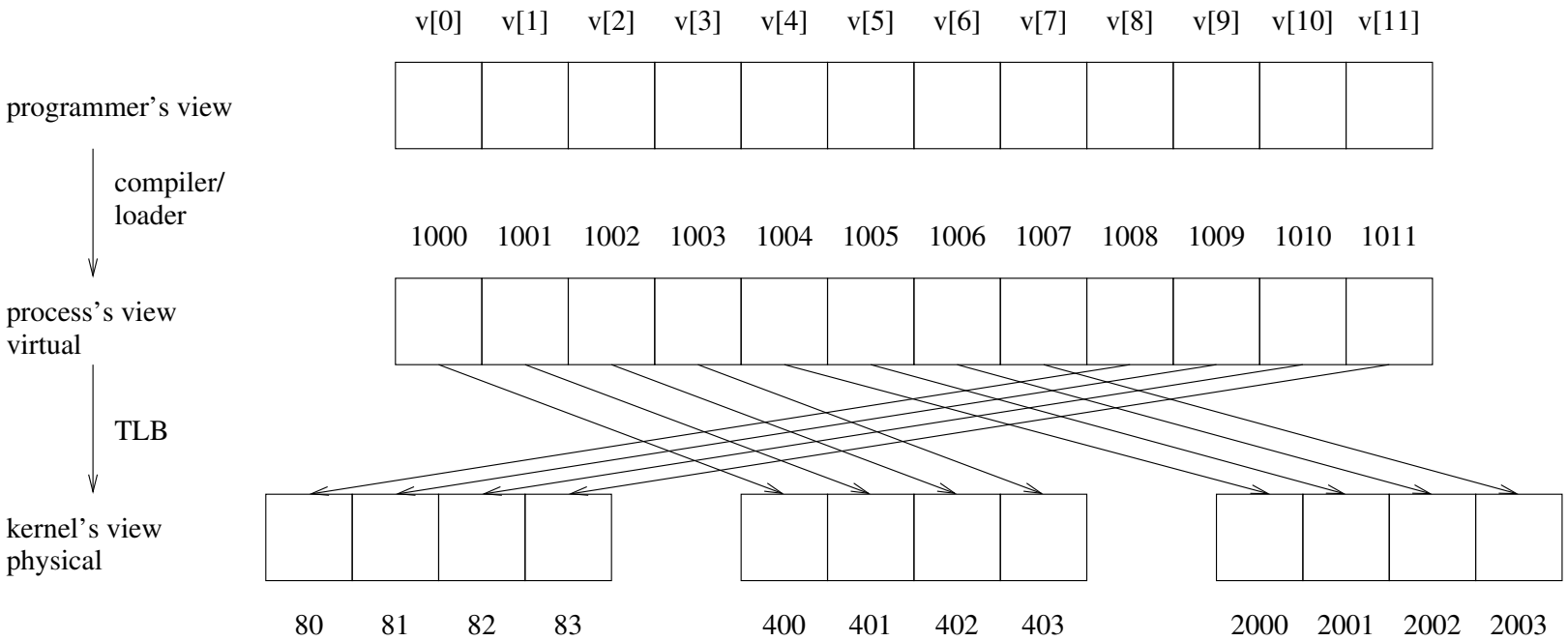
This simplification over earlier allocation methods is the big benefit of using pages

The first page on the freelist of pages is always suitable. No need to search, no size fit issues, no fragmentation issues

Memory

Virtual Memory

A single large datastructure (e.g., a vector, which you normally think of as a contiguous region of memory) in your process might actually be spread, in chunks, all over the place in physical memory



Memory

Virtual Memory

Similarly for code: a chunk of code spanning multiple pages may well be distributed all over physical memory

□

Memory

Virtual Memory

Similarly for code: a chunk of code spanning multiple pages may well be distributed all over physical memory

Code or data might be contiguous in the virtual address space, but definitely not contiguous in the physical address space