Today

- Virtual Memory
  - Page selection
  - Prefetching
  - Allocation
  - Other issues

Review: Memory Management

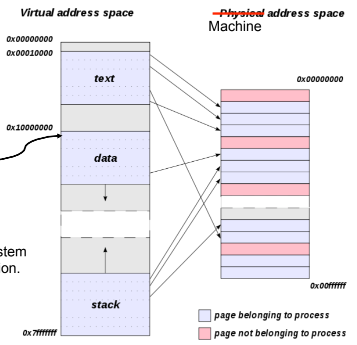
- In general, what is the memory abstraction that the OS provides users?
- How does the OS allow multiprogramming?
- How does the OS implement virtual memory?
- What are some optimizations that the OS provides with respect to virtual memory?

VM Page Maps

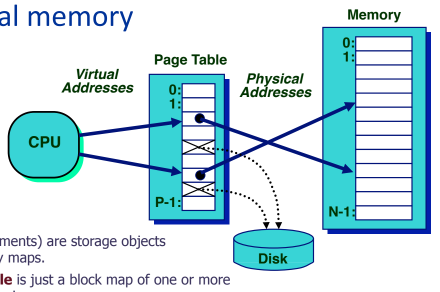
This picture is an **example** of a virtual memory on a 32-bit machine. Details vary.

Global data and dynamic ("**heap**") memory.

A key role of the operating system is to manage the VM abstraction.



Virtual memory



VMs (or segments) are storage objects described by maps.

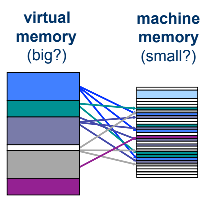
A **page table** is just a block map of one or more VM segments in memory.

The hardware hides the indirection from user programs.

Virtual addressing

Code running on a core addresses memory through **virtual addresses**.

The machine translates virtual addresses via an in-memory **page table**.



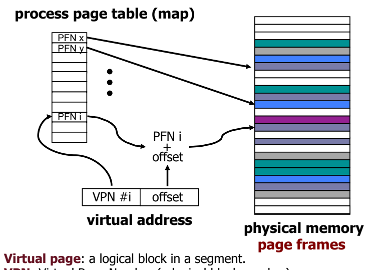
The machine allows a user process to access memory only by a valid **translation** in the page table.

The OS controls the contents of the page table.

The page table represents a functional mapping of virtual pages (VPNs) to page frames (PFNs) for **resident** pages. If a page is not resident in memory, then its page table entry is marked as invalid.

The specific mechanisms for virtual address translation are **machine-dependent**.

Cartoon view of a page table



This is an **example**. Any PFN may be used for any VPN.

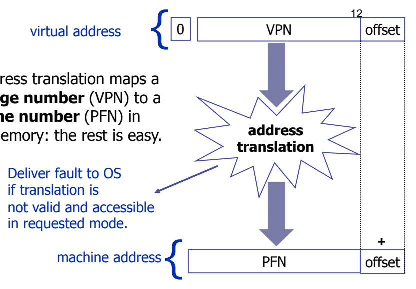
The map itself is just another data structure stored in memory.

A protected CPU register holds the machine address of the current map.

Virtual page: a logical block in a segment.
VPN: Virtual Page Number (a logical block number).
Page frame: a physical block in machine memory.
PFN: Page Frame Number (a block pointer).
PTE: Page Table Entry (an entry in the block map).

Virtual Address Translation

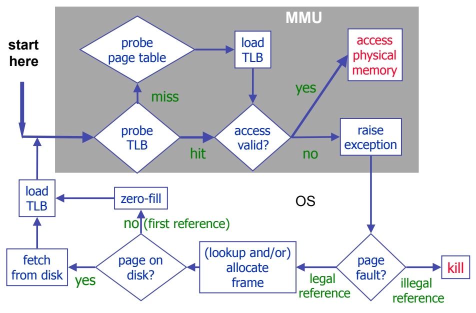
Example only: a typical 32-bit architecture with 4KB pages.


virtual address
0
VPN
12
offset
address translation
Deliver fault to OS if translation is not valid and accessible in requested mode.
machine address
+
PFN
offset

Virtual address translation maps a **virtual page number** (VPN) to a **page frame number** (PFN) in machine memory: the rest is easy.

Addresses Phil's Question

Virtual Addressing: Under the Hood



Partitioning Summary

- Variable partitioning is a pain
  - But, we need it for heaps
- For files, we can break objects down into "pieces".
  - When access to files is through an API, we can add some code behind that API to represent the file contents with a dynamic linked data structure (a map).
  - If the pieces are fixed-size (called pages or logical blocks), we can use fixed partitioning to allocate the underlying storage, which is efficient and trivial.
  - With that solution, internal fragmentation is an issue, but only for small objects. (Why?)
- That approach can work for VM segments too
  - have VM hardware to support it since the 1970s

PAGE CACHING POLICY

Page Caching Policy

- Each thread/process/job makes a stream of page/block references.
  - reference string: e.g., abcabcdabce..
- OS tries to minimize number of fetches/faults
  - Try to arrange for the resident set of blocks to match the set of blocks that will be needed in the near future

Page Replacement

- Replacement policy is the key
  - On each access, select a victim block to evict from memory
    - Read the new block into victim's frame
  - I/O caches are fully associative: a given block can be anywhere in the cache, so any block is a potential eviction candidate.
- How do we know which page should be the "victim"?
  - Removed from virtual memory
  - Replaced with another page
- Provably optimal: replace the page whose next reference is furthest in the future (OPT/MIN)

Policy for selecting a victim

- The oldest block? (FIFO policy)
- The coldest block? (Least Recently Used)
- The hottest block? (Most Recently Used)?
- The least popular block? (Least Frequently Used)
- A random block?
- A block that has not been used recently?

FIFO in Action

Reference	A	B	C	D	E	A	B	C	D	E	A	B	C	D	E
1	A	A	A	A	E	E	E	E	D				C		
2		B	B	B	B	A	A	A	A	E				D	
3			C	C	C	C	B	B	B		A				E
4				D	D	D	D	C	C			B			

Worst case for FIFO is if program strides through memory that is larger than the cache

How would random's performance compare?

FIFO in Action

All cache misses →

Reference	A	B	C	D	E	A	B	C	D	E	A	B	C	D	E
1	A	A	A	A	E	E	E	E	E				C		
2		B	B	B	B	A	A	A	D	E				D	
3			C	C	C	C	B	B	B		A				E
4				D	D	D	D	C	C			B			

Worst case for FIFO is if program strides through memory that is larger than the cache

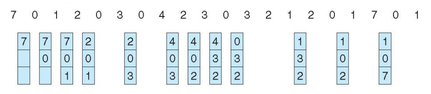
How would random's performance compare?

Random could result in better performance

Least Recently Used (LRU) Algorithm

- Use past knowledge rather than future
- Replace page that has not been used in the most amount of time
- Associate time of last use with each page

String of page requests:



Memory over time →

LRU Algorithm Implementations

- Counter implementation
  - Every page entry has a counter
  - Every time page is referenced through this entry, copy the clock into the counter
  - When a page needs to be changed, look at the counters to find smallest value
- Stack implementation
  - Keep a stack of page numbers in a double link form
  - Page referenced:
    - move it to the top

Implementation Tradeoffs?

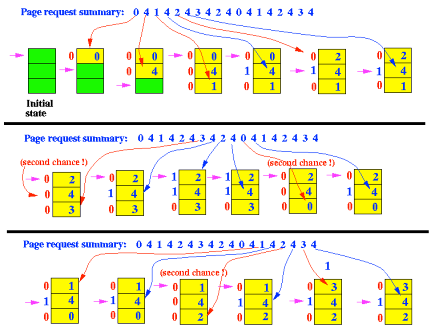
LRU Algorithm Implementations

- Counter implementation
  - Every page entry has a counter
  - Every time page is referenced through this entry, copy the clock into the counter
  - When a page needs to be changed, look at the counters to find smallest value
  - ◆ Requires searching through table
- Stack implementation
  - Keep a stack of page numbers in a double link form
  - Page referenced:
    - move it to the top
    - requires 6 pointers to be changed
  - ◆ No search for replacement, but updates are more expensive

LRU Approximation Algorithms

- LRU needs special hardware and still slow
- Reference bit
  - Associate each page with a bit, initially 0
  - When page is referenced, bit set to 1
  - Replace any page with reference bit 0 (if one exists)
    - We do not know the order replaced
- Second-chance, aka clock, algorithm
  - Generally FIFO, plus hardware-provided reference bit
  - If page to be replaced has
    - Reference bit = 0 → replace it
    - reference bit = 1 →
      - set reference bit 0, leave page in memory
      - replace next page, subject to same rules

Second Chance Page Replacement Algorithm



http://www.mathcs.emory.edu/~cheung/Courses/355/Syllabus/9-virtual-mem/SC-replace.html

Enhanced Second-Chance Algorithm

- Improve by using both reference bit and modify bit
- Consider ordered pair (reference, modify)
  - (0, 0) not recently used, not modified – best page to replace
  - (0, 1) not recently used but modified – not quite as good, must write out before replacement
  - (1, 0) recently used but clean – probably will be used again soon
  - (1, 1) recently used and modified – probably will be used again soon and need to write out before replacement
- Algorithm: use the clock scheme but use the four classes to replace page in lowest non-empty class
  - Might need to search circular queue several times

Review: Page-Buffering Algorithms

- Keep a pool of free frames, always
  - Frame available when needed, not found at fault time
  - Read page into free frame and select victim to evict and add to free pool
  - When convenient, evict victim
- Possibly, keep list of modified pages
  - When backing store otherwise idle, write pages there and set to non-dirty
- Possibly, keep free frame contents intact and note what is in them
  - If referenced again before reused, no need to load contents again from disk
  - Generally useful to reduce penalty if wrong victim frame selected

PREFETCHING

Cache Misses

Miss Type	Description	Hotel Analogy
Compulsory or Cold	The first reference to a block of memory, starting with an empty cache.	The hotel is empty and the first guest has not yet arrived.
Capacity	The cache is not big enough to hold every block you want to use.	The hotel has no vacancies.
Conflict	Two blocks are mapped to the same location and there is not enough room to hold both.	A particular floor of the hotel which a guest has to stay on has all rooms occupied.

Prefetching

- Idea: Fetch the data before it is needed (i.e. pre-fetch) by the program
- Why?
  - Memory latency is high. If we can prefetch accurately and early enough we can reduce/eliminate that latency.
  - Can eliminate compulsory cache misses
  - Can eliminate all cache misses? Capacity, conflict?
- Involves predicting which address will be needed in the future
  - Works if programs have predictable miss address patterns

Prefetching and Correctness

- Does a misprediction in prefetching affect correctness?
  - No, prefetched data at a "mispredicted" address is simply not used
- There is no need for state recovery
- In contrast to branch misprediction or value misprediction

Basics

- In modern systems, prefetching is usually done in cache block granularity
- Prefetching is a technique that can reduce both
  - Miss rate
  - Miss latency
- Prefetching can be done by
  - hardware
  - compiler
  - programmer

Prefetching Challenges?

Prefetching: The Four Questions

- What?
  - What addresses to prefetch
- When?
  - When to initiate a prefetch request
- Where?
  - Where to place the prefetched data
- How?
  - Software, hardware, execution-based, cooperative

Challenges in Prefetching: What

- Prefetching useless data wastes resources
  - Memory bandwidth
  - Cache or prefetch buffer space
  - Energy consumption
  - These could all be utilized by demand requests or more accurate prefetch requests
- Accurate prediction of addresses to prefetch is important
  - Prefetch accuracy = used prefetches / sent prefetches
- How do we know what to prefetch
  - Predict based on past access patterns
  - Use the compiler's knowledge of data structures
- Prefetching algorithm determines what to prefetch

Challenges in Prefetching: What

- One option: choose based on locality
  - Temporal locality (history)
  - Spatial locality (“nearby” data)

Challenges in Prefetching: When

- Prefetching too early
  - Prefetched data might not be used before it is evicted from storage
- Prefetching too late
  - Might not hide the whole memory latency
- When a data item is prefetched affects the timeliness of the prefetcher
- Prefetcher can be made more timely by
  - Making it more aggressive: try to stay far ahead of the processor’s access stream (hardware)
  - Moving the prefetch instructions earlier in the code (software)

Challenges in Prefetching: Where

- In cache
  - + Simple design, no need for separate buffers
  - -- Can evict useful demand data → cache pollution
- In a separate prefetch buffer
  - + Demand data protected from prefetches → no cache pollution
  - -- More complex memory system design
    - Where to place the prefetch buffer
    - When to access the prefetch buffer (parallel vs. serial with cache)
    - When to move the data from the prefetch buffer to cache
    - How to size the prefetch buffer
    - Keeping the prefetch buffer coherent

Challenges in Prefetching: Where (II)

- Which level of cache to prefetch into?
  - Memory to L2, memory to L1. Advantages/disadvantages?
  - L2 to L1? (a separate prefetcher between levels)
- Where to place the prefetched data in the cache?
  - Do we treat prefetched blocks the same as demand-fetched blocks?
  - Prefetched blocks are not known to be needed
- Do we skew the replacement policy such that it favors the demand-fetched blocks?
  - E.g., place all prefetches into the LRU position in a way?
- Where to place the hardware prefetcher in the memory hierarchy?
- …

Challenges in Prefetching: How

- Software prefetching
  - ISA provides prefetch instructions
  - Programmer or compiler inserts prefetch instructions (effort)
  - Usually works well only for “regular access patterns”
- Hardware prefetching
  - Hardware monitors processor accesses
  - Memorizes or finds patterns/strides
  - Generates prefetch addresses automatically
- Execution-based prefetchers
  - A “thread” is executed to prefetch data for the main program
  - Can be generated by either software/programmer or hardware

Looking Ahead

- Project 5 – due Friday
- Exam envelopes - Friday