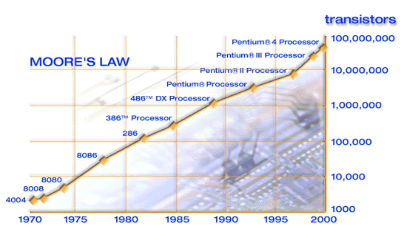
Memory Performance of Algorithms

CSE 326
Data Structures
Lecture 4

Algorithm Performance Factors

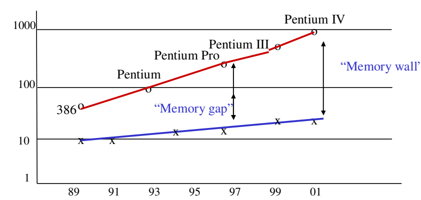
- Algorithm choices (asymptotic running time)
  - $O(n^2)$ or $O(n \log n)$ …
- Data structure choices
  - List or Arrays
- Language and Compiler
  - C, C++, Java, Fortran
- Memory performance
  - How near is the data to the processor

Moore's Law

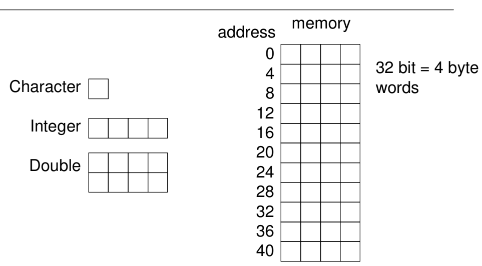


Processor-Memory Performance Gap

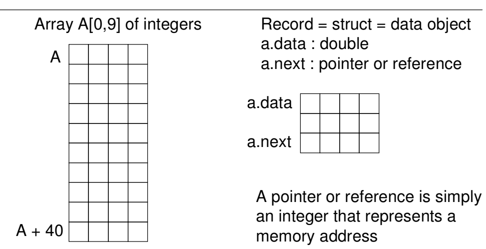
- x86 CPU speed (100x over 10 years)



Program Model of Memory I



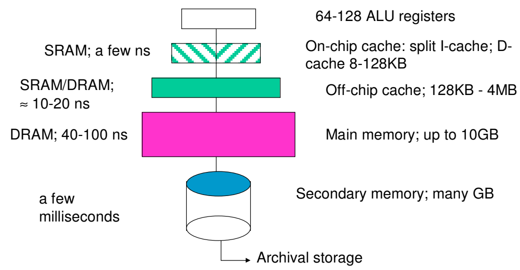
Program Model of Memory II



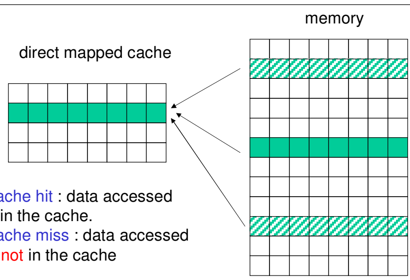
Memory Model vs. Reality

- The program memory model is very simple and elegant
- The reality is not.
- Physical memory is organized in a hierarchy.
  - › Accessing memory close to the processor is fast
  - › Accessing memory far from the processor is slower
- Caching allows for accessed data to be moved closer to the processor.
  - › There is a win if that data is accessed again

Levels in the Memory Hierarchy

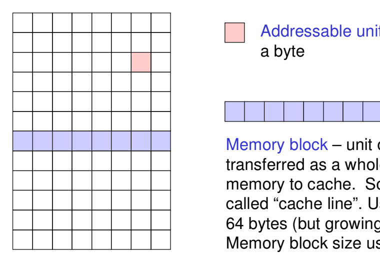


The Cache



Cache hit : data accessed is in the cache.
Cache miss : data accessed Is not in the cache

Memory Blocks



Addressable unit, usually a byte

Memory block – unit of memory transferred as a whole from memory to cache. Sometimes called "cache line". Usually, 32 64 bytes (but growing in size). Memory block size usually greater than word size

Why Memory Blocks

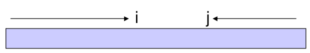
- Time to transfer x bytes is given by
  T(x) = a + bx. (a is latency, b ~ 1/bandwidth)
- Because a is large relative to b, it pays to transfer more than one byte at a time.
  - › The hope is that bytes near the accessed byte will be accessed soon – good spatial locality.

Locality

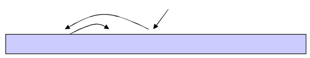
- Spatial locality : addresses near a recently accessed byte are accessed also.
- Temporal locality : the same address that was accessed recently is accessed again.

Examples of Locality

- Good spatial locality
  - Quicksort – the array is scanned

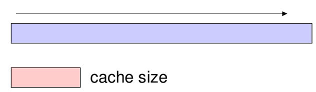


- Poor spatial locality
  - Binary search – jump around the array



Examples of locality

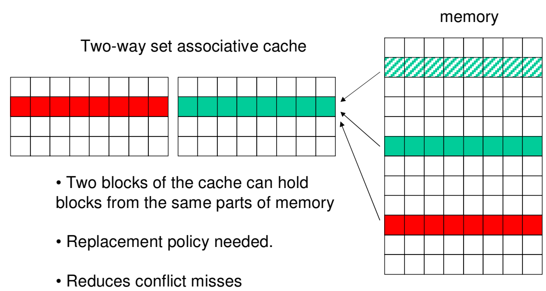
- Good temporal locality
  - For loop index i in a tight loop.
    for i = 1 to n do { …}
- Poor temporal locality
  - Repeated long scans that exceeds the cache size, like in iterative merge sort.



Classifying Cache Misses

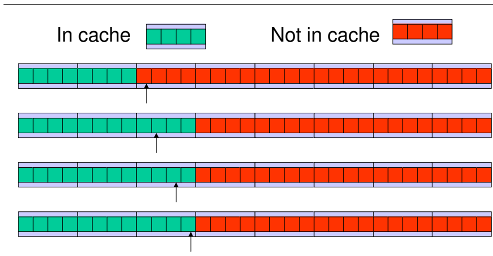
- Compulsory misses – first time a block is accessed
  - Can never be avoided
- Capacity misses – data structure does not fit in cache
  - Can be avoided by algorithmic design.
- Conflict misses – several accessed blocks map to the same location in cache
  - Conflict misses are not much of a problem because modern caches are set associative.

Set Associative Cache



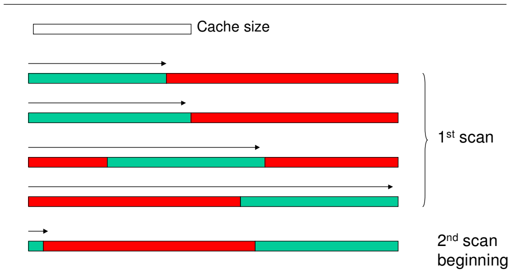
- Two blocks of the cache can hold blocks from the same parts of memory
- Replacement policy needed.
- Reduces conflict misses

Cache Misses for Scans



1/B misses per access where B is number of access per line

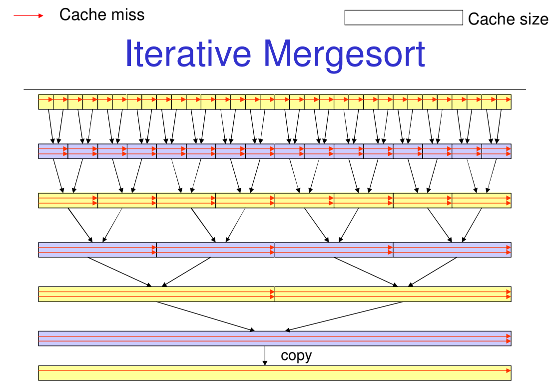
Repeated Long Scans



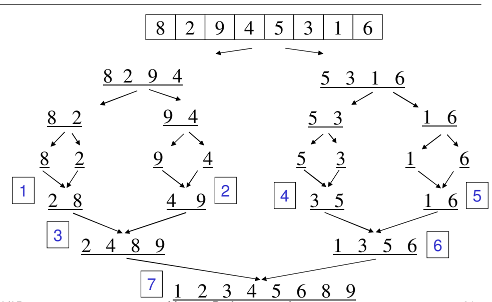
Repeated Long Scans

- Have good spatial locality
- Poor temporal locality
- If there are B accesses per memory block then 1/B of the accesses are cache misses.

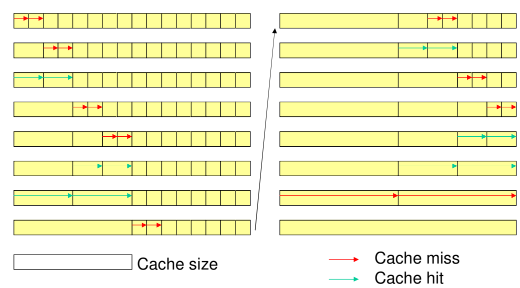
Iterative Mergesort



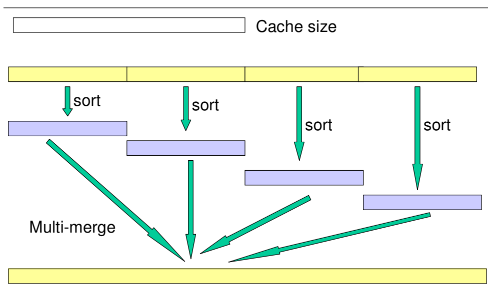
Recursive Mergesort



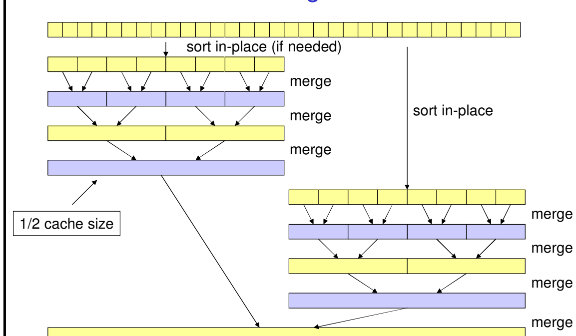
Recursive Mergesort



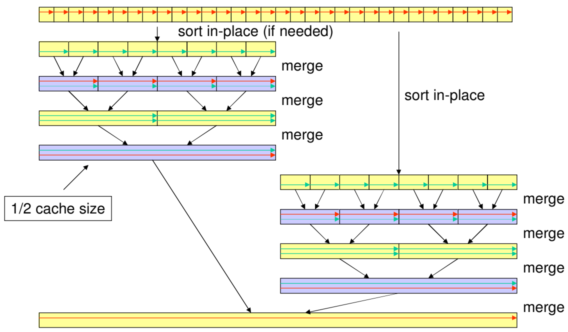
Multi-mergesort

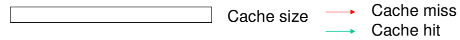


Multi-Mergesort

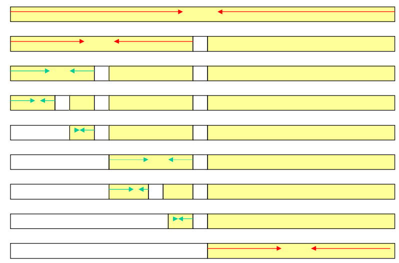


Multi-Mergesort Cache Behavior





Quicksort



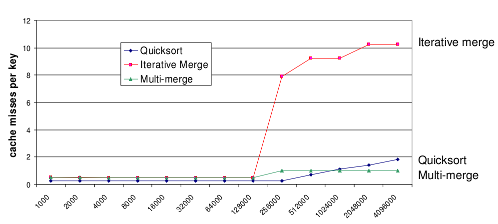
Sorting Study from 1996

- Compared sorting algorithms
  - Cache misses
  - Instruction count
  - Execution time
- The study is still valid today, because the gap between processor speed and memory speed is even larger.

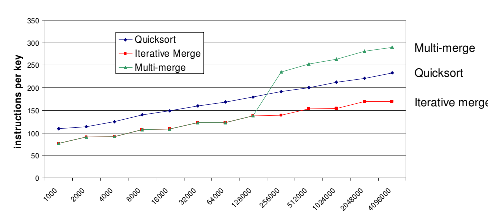
Algorithms

- Iterative mergesort
- Multi-mergesort
- Quicksort

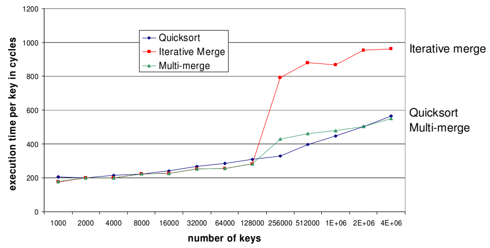
Cache Misses



Instructions



Execution Time



Notes on Memory Performance

- Memory performance may matter.
- Tips
  - Sacrifice instructions to get better cache performance.
  - Smaller memory footprint is good.
  - Divide and conquer is good.
  - Processing data into cache sized pieces is good.
  - Fully utilize memory blocks if possible
    - Short scans are good.
    - Multiway trees are good.

External Memory Sort

- Memory bottleneck even worse for disk
- If input too big to fit in main memory, regular sorting algorithms are too slow
- Whole subject of external sorting

Disks

- In-memory sorting uses random access model of memory. Disks are sequential.
- A movable head over a rotating platter
- Reading sequentially fast
- Seeking to new location slow
- Sort time dominated by number of seeks

One external sort model

- With only 1 sequential access memory, sorting takes Omega(N^2)
- We'll use a model with 4 disks.
- Each can be read concurrently
- Call disks A1, A2, B1, B2
- Say main memory can hold M elements

A simple algorithm

- Data initially on A1
- Sort block of size M in memory, writing first half to B1, second half to B2
- Now merge half of B1 and B2 onto A1, and the other half to A2
- Blocks are now of size 2M
- Repeat for log(N/M) steps.