Computer Systems Lecture 15

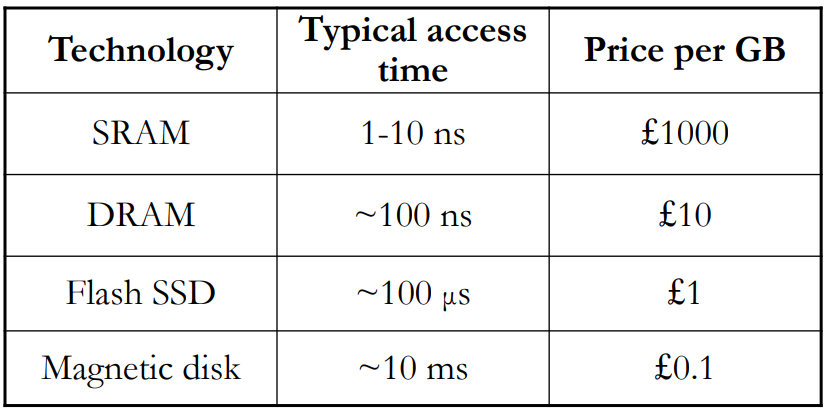
Memory Requirements

Programmers want memory to be: large, fast and random access

This is not achievable with one kind of memory, there are issues with cost and technical feasibility.

The idea of a memory hierarchy is used to approximate the ideal large and fast memory through a combination of different kinds of memories.

Memory Examples

(main memory is make of DRAM)

As you can see, as the memory gets faster, it gets much MUCH more expensive

Memory Hierarchy Overview

Use a combination of the types of memory:

* Smaller amounts of expensive but fast memory closer to the processor
* Larger amounts of cheaper but slower memory farther from the processor

Why is Memory Hierarchy Effective

Temporal Locality:

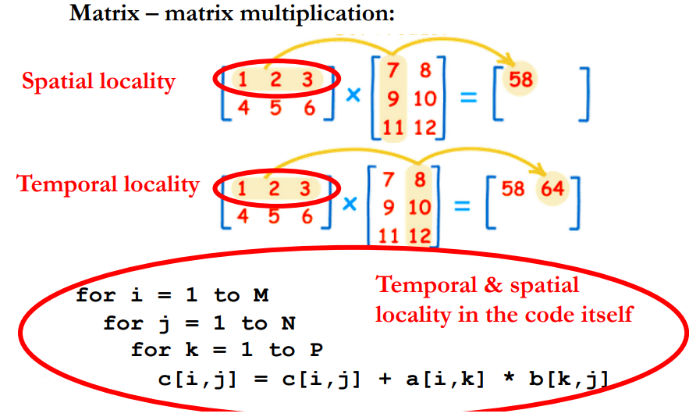
* A recently accessed memory location (instruction or data) is likely to be accessed again in the near future

Spatial Locality:

* Memory locations (instructions or data) close to a recently accessed location are likely to be accessed in the near future

Why does locality exist in programs?

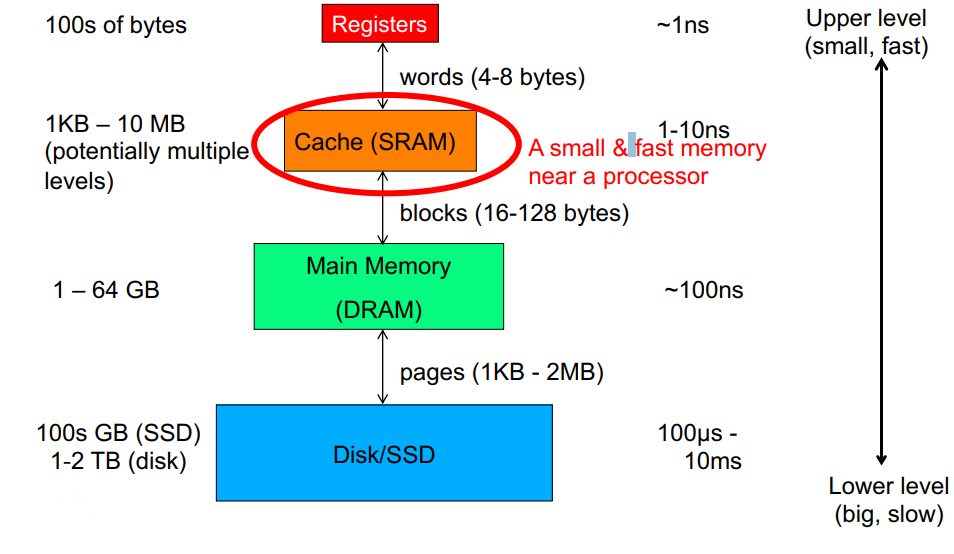
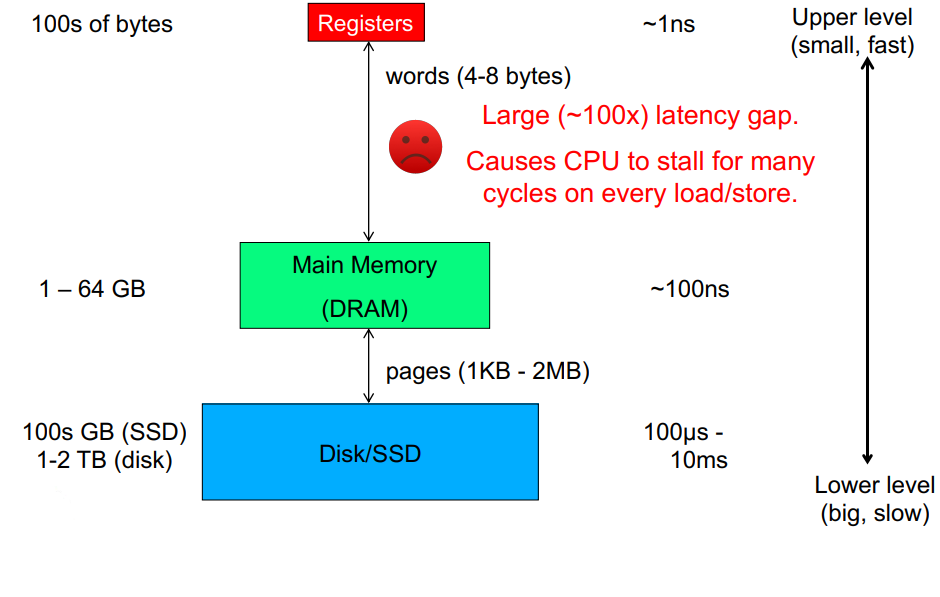
* Instruction reuse: loops, functions
* Data working sets: arrays, temporary variables, objects

Example of Temporal and Spatial Locality

the first is an example of special locality because the row is all stored together

The second is an example of temporal locality because we’ve already used the row to calculate the first element of the result, then we use it again for the second element.

Levels of the Memory Hierarchy



Approximately 20% of all instructions are loads and stores, If the cycle time of the processor is much faster than the access time for the memory its trying to load/store, many cycles will be wasted waiting for the operation to complete.

To fix this, we can exploit temporal and special locality with cache. Caches are much faster than main memory but also much smaller (in many cases, cache also has an internal hierarchy, of a similar structure).

Memory Hierarchy in a Modern Processor

Small, fast cache next to a processor backed up by larger and slower cache(s) and main memory give the impression of a single large, fast memory.

Cache takes advantage of temporal locality:

* If we access data from slower memory, move it to faster memory
* If the data in faster memory is unused recently, move it to the slower memory

Cache also takes advantage of spatial locality:

* If wee need to move a word from slower to faster memory, move adjacent words at the same time
* This gives rise to blocks and pages: units of storage within the memory hierarchy composed of multiple contiguous words.

Control of Data Transfers in Hierarchy

Should the software or hardware be responsible for moving data between levels of the memory hierarchy?

It depends, there is a trade-off between ease of programming, hardware complexity and performance:

* Software (compiler): Is responsible for moving data between registers and cache/main memory
* Hardware: Is responsible for moving data between caches and main memory (Software is usually unaware of caches)
* Software (Operating System): Is responsible for moving data between main memory and disk

Hardware-Managed Transfers Between Levels

This occurs between cache memory and main memory levels. The programmer and processor are both oblivious to where the data resides, they just issue loads and stores to ‘memory’

Cache hardware manages transfers between levels

* Data moved or copied between levels automatically in response to the program’s memory accesses
* Memory always has a copy of cached data, but data in the cache may be more recent. This can create interesting problems (not discussed in this course).

Cache Terminology

* Block (or line): the unit of data stored in the cache
  + Typically in the range of 32-128 bytes
  + Block size is larger than a word, this helps exploit spatial locality
* Hit: data is found (this is what we want to happen
  + This means the memory access completes quickly
* Miss: data not found
  + We must continue the search at the next level of the memory hierarchy (could be another cache or main memory)
  + After data is eventually located, it is copied to the memory level where the miss happened.
* Hit rate (hit ratio): the fraction of accesses that are hits at a given level of the memory hierarchy
* Miss rate (miss ratio): the fraction of accesses that are misses at a given level (= 1-hit rate)
* Allocation: the placement of a new block into the cache which typically results in an eviction of another block
* Eviction: displacement of a block from the cache, which commonly happens when a new block is allocated in its place