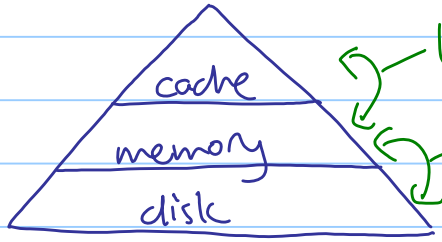
Virtual memory

Note Title

Recall memory hierarchy includes cache, memory and disk (among other things):

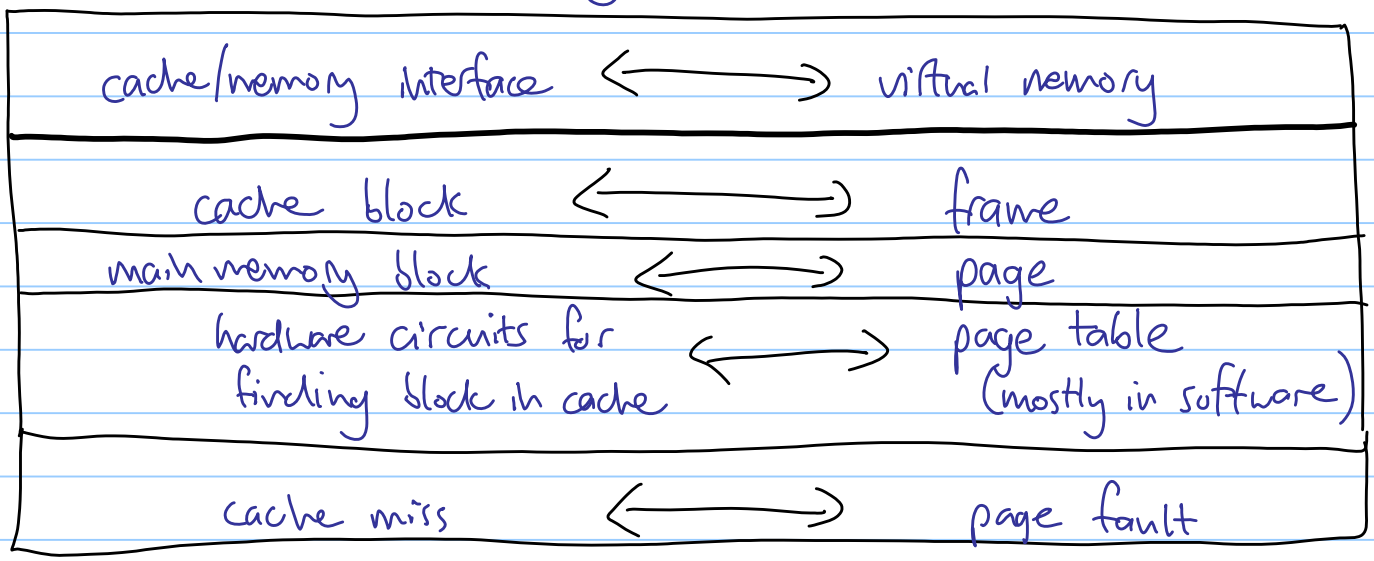


Main idea of virtual memory:

- each process (i.e. program) has its own logical (as opposed to physical) address space numbered from 0 to, for example, 2^{32} (on 32-bit machine) or 2^{64} (on 64-bit machine)
- the program executing in a process uses logical addresses at all times.
  e.g. LOAD 007 means "load logical address 7". But this data could be stored anywhere on the computer. For example, it could be at the real, physical address 14AB6203
- thus, the computer must maintain a mapping from the 'virtual' memory of a process (i.e. its logical address space) to the physical addresses in the computer. A separate map must be maintained for each process

- the (logical) memory used by a process is not necessarily all stored in the physical memory. Parts of it are stored on the disk instead, and are copied into the memory when needed. The caching and eviction strategies are similar to those for the memory/cache strategies studied previously.

Ideas are similar, but terminology is different:



See worksheet for details.
- we study the version "without TLB" first, then "with TLB"

① VM without TLB

Exercise: compute page table contents for page 1 of worksheet

Problem: Even when we get a hit (i.e. no page fault), 2 memory accesses are required:
- one to access the page table (which is itself stored in memory)
- one to access the desired data.

So, memory accesses take twice as long as they should (assuming we can design the system better).

A better design is...

(2) VM with TLB

The TLB (or translation lookaside buffer) is a special piece of hardware designed to cache the contents of the page table.
Whenever we get a TLB hit, we avoid the first memory access mentioned above. But on a TLB miss, 2 accesses are required as before.

Exercise: do page 2 of the worksheet

Demo and/or mimlab: bigarray.c