—The Memory Hierarchy

Random-Access memory (RAM)

(I) key features

(i) RAM is traditionally packaged as a chip.

(ii) Basic storage unit is normally a cell (one bit per cell)

(iii) Multiple RAM chips form a memory.

(II) RAM comes in two varieties.

(i) SRAM (static RAM)

(ii) DRAM (Dynamic RAM)

	Trans. per bit	Access time	Needs refresh?	Needs EDC?	Cost	Applications
SRAM	4 or 6	1X	NO	Maybe	100X	cache memories
DRAM	1	10X	YES	Yes	1X	Main memories frame buffers

Nonvolatile Memories

(I) DRAM and SRAM are volatile memories
(i) Lose information if powered off.

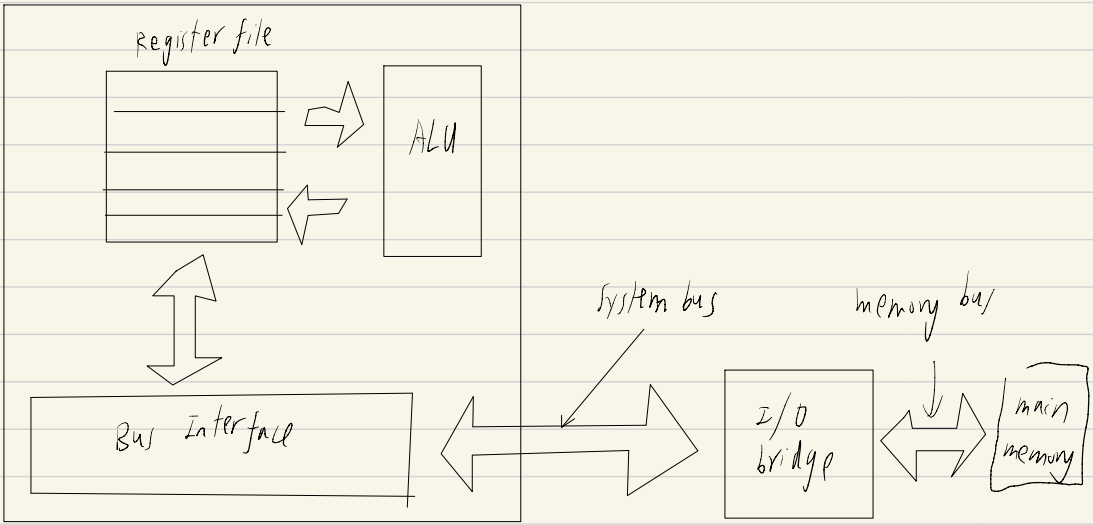
(II) Nonvolatile memories retain value even if powered off:
(i). Read-only memory (ROM): programmed during production.
(ii) Programmable ROM (PROM): can be programmed once.
(iii) Erasable PROM (EPROM): can be bulk erased (UV, X-Ray)
(iv). Electrically erasable PROM (EEPROM): electronic erase capability.
(v). Flash memory: EEPROMs with partial (block-level) erase capability.
· Wears out after about 100,000 erasing.

(III) Uses for Nonvolatile Memories.
(i) Firmware programs stored in a ROM (BIOS, controllers for disks network cards, graphics accelerators, security subsystems...)
(ii) Solid state disks (replace rotating disks in thumb drives, smart phones, mp3 players, tablets, laptops...)
(iii) Disk caches.

Traditional Bus Structure connecting CPU and memory

A bus is a collection of parallel wires that carry address, data and control signals.

Buses are typically shared by multiple devices.



Main memory reads A from the memory bus, retrieves word x, and places it on the bus.

CPU read word x from the bus and copies it into register %rax.

Platter rotating

slower than DRAM

SRAM.

Disk Geometry

(I) Disks consist of platters, each with two surfaces.

(II) Each surface consists of concentric rings called tracks.

(III) Each track consists of sectors separated by gaps.

Disk capacity:

(I) Capacity: maximum number of bits that can be stored.

Vendors express capacity in units of gigabytes (GB), where $1\,GB = 10^9$ Bytes.

(II) Capacity is determined by these technology factors:

(i) Recording density (bits/in): number of bits that can be squeezed into a 1 inch segment of track.

(ii) Track density (tracks/in): number of tracks that can be squeezed into a 1 inch radial segment.

(iii) Areal density (bits/in^2): product of recording and track density.

(III) Modern disks partition tracks ..... into disjoint subsets called recording zones

Disk Access Time

(I) Average time to access some target sector approximated by:

(i) $T_{access} = T_{avg\ seek} + T_{avg\ transfer}$

(II) Seek Time ($T_{avg\ seek}$)

Time to positions heads over cylinder containing target sector.

Typical $T_{avg\ seek}$ is 3-9 ms.

(III) Rotational Latency ($T_{avg\ rotation}$) (read/write)

(i) Time waiting for first bit of target sector to pass under r/w head.

(ii) $T_{avg\ rotation} = 1/2 \times 1/RPMs \times 60 sec/1 min.$

(iii) Typical $T_{avg\ rotation} = 7200\ RPMs.$

(IV) Transfer time ($T_{avg\ transfer}$) ignorable compared to preceding two.

(i) Time to read the bits into the target sector.

(ii) $T_{avg\ transfer} = 1/RPM \times 1/(avg\ \#sectors/track) \times 60 secs/1 min.$

Logical Disk Blocks:

(4) Modern disks present a simpler abstract view of the complex sector geometry:

(i) The set of available sectors is modelled as a sequence of b-sized logical blocks (0, 1, 2, ...)

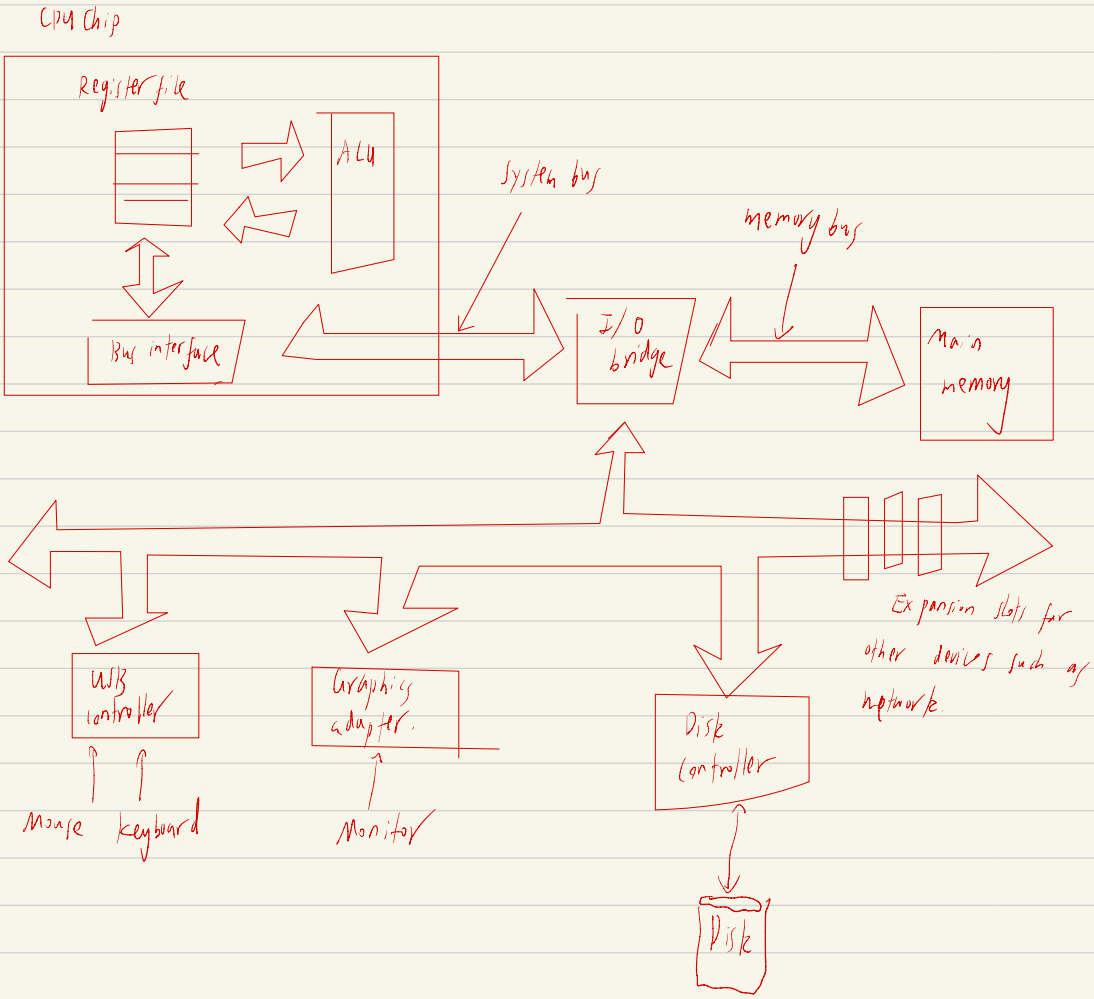
(ii) Mapping between logical blocks and actual (physical) sectors.

(i) Maintained by hardware/firmware device called disk controller.

(ii) Converts requests for logical blocks into (surface, track, sector) triple.

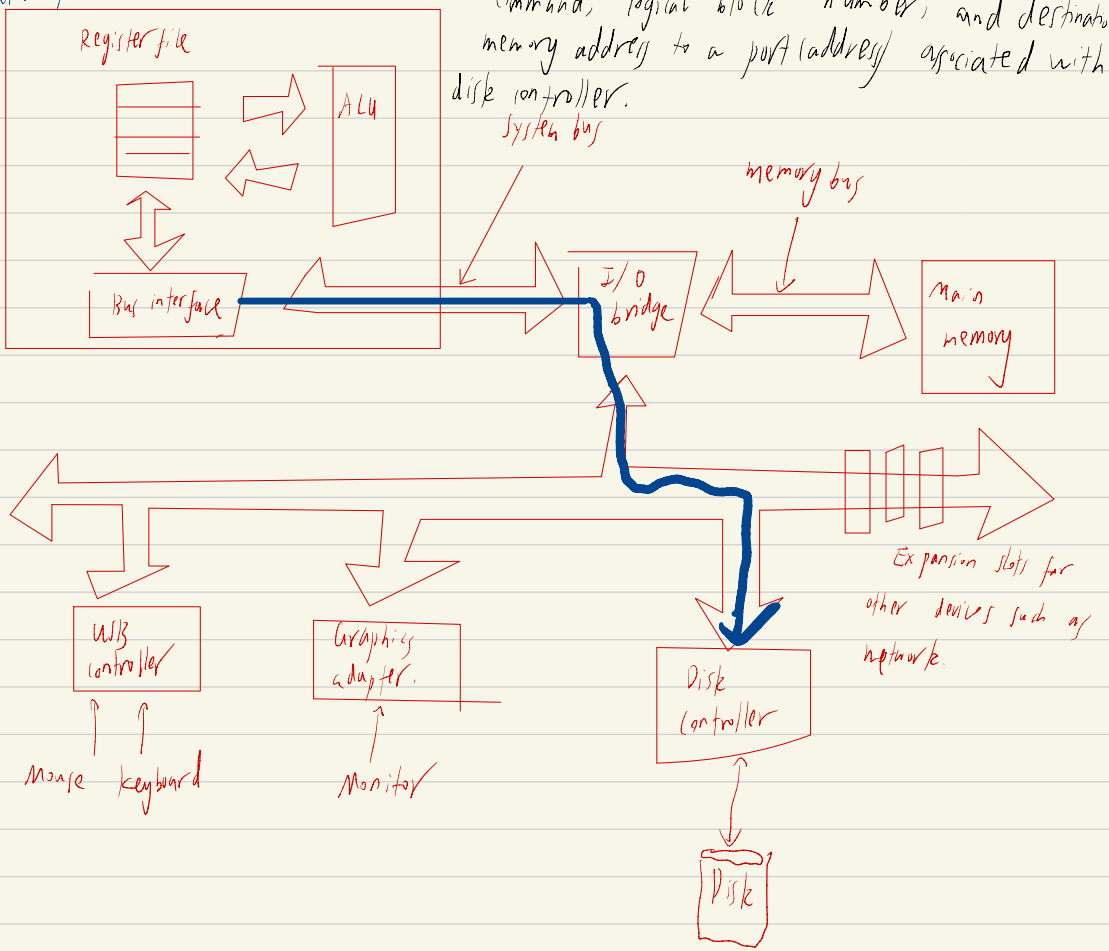
(iii) Allows controller to set aside spare cylinders for each zone. Accounts for the difference in "formatted capacity" and "maximum capacity".

I/O Bus


CPU Chip
Register file
ALU
Bus interface
System bus
I/O bridge
memory bus
Main memory
USB controller
Graphics adapter
Disk controller
Expansion slots for other devices such as network
Mouse
Keyboard
Monitor
Disk

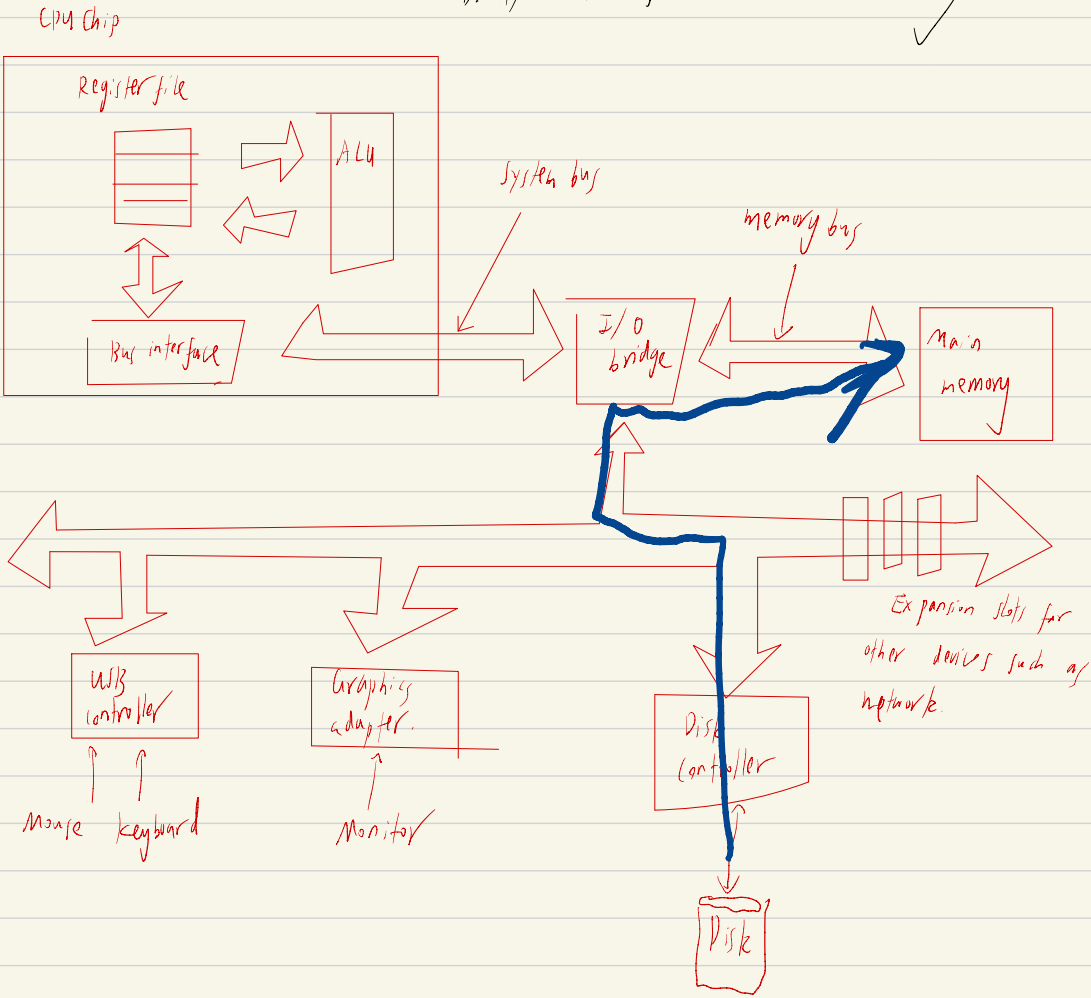
Reading a Disk Sector (1)

CPU initiates a disk read by writing a command, logical block number, and destination memory address to a port (address) associated with disk controller.



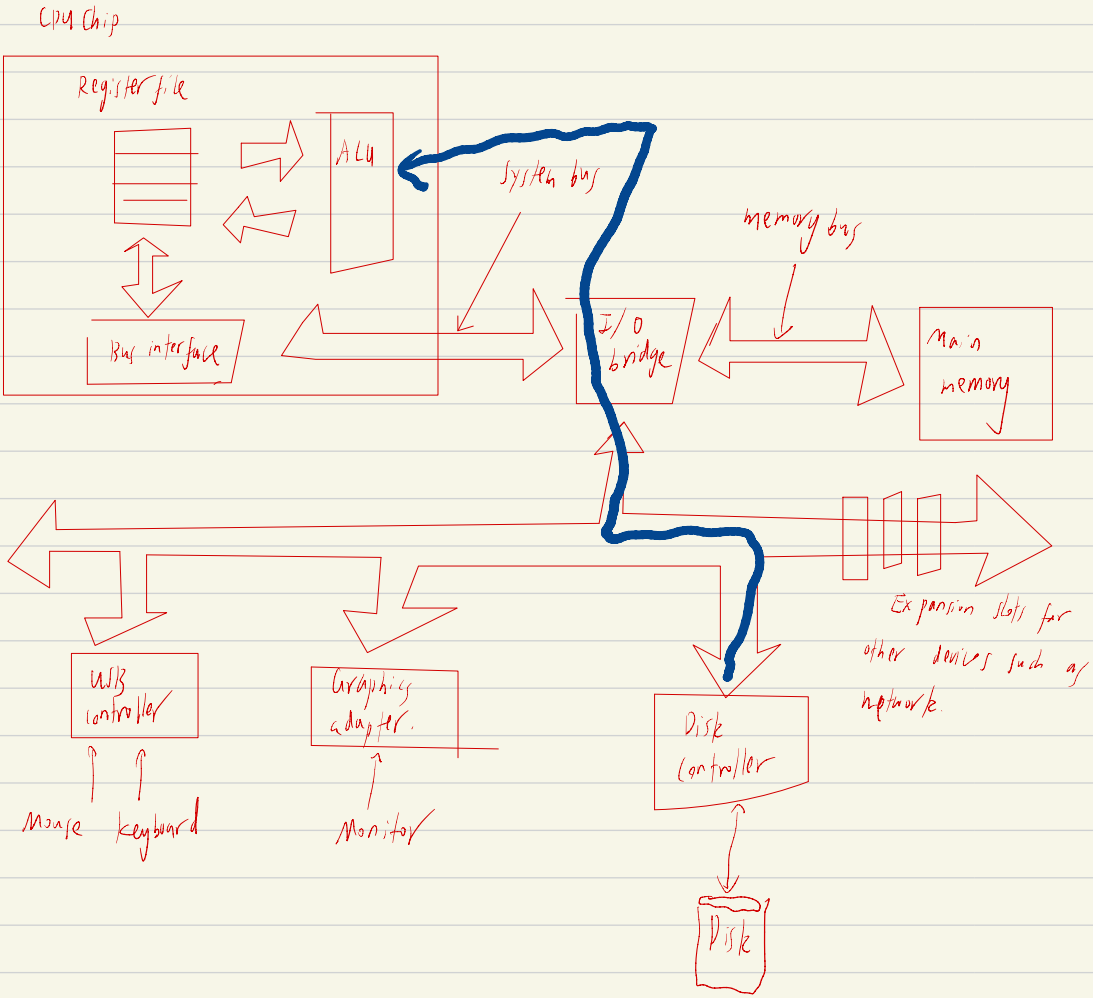
Reading a Disk Sector (1)

Disk controller reads the sector and performs a direct memory access (DMA) transfer into memory



Reading a Disk Sector (5)

When the DMA transfer completes, the disk controller notifies the CPU with an interrupt (i.e. asserts a special "interrupt" pin on the CPU)



SSD Performance Characteristics

Sequential read tput	550 MB/s	Sequential write tput	470 MB/s
Random read tput	365 MB/s	Random write tput	303 MB/s
Avg seq read time	50 µs	Avg seq write time	60 µs

(1) Sequential access faster than random access
(i) Common theme in the memory hierarchy.

(2) Random writes are somewhat slower.
(i) Erasing a block takes a long time (~1ms)
(ii) Modifying a block page requires all other pages to be copied to new block.
(iii) In earlier SSDs, the read/write gap was much larger.

solid-state drive

SSD trade offs v.s. rotating Disks

(I) Advantages

(i) No moving parts → faster, less power, more rugged

(II) Disadvantages

(i) Have the potential to wear out

(ii) In 2011, about 30 times more expensive per byte

(III) Applications

(i) MP3 players, smart phones, laptops

(ii) Beginning to appear in desktops and servers

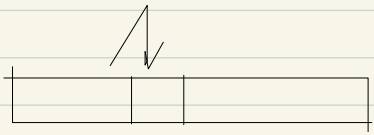
Locality to the Rescue!

The key to bridging this CPU-Memory gap is a fundamental property of computer program known as locality.

Principle of Locality: Programs tend to use data and instructions with addresses near or equal to those they have used recently

Temporal locality:

- recently referenced items are likely to be referenced again in the near future.



Spatial locality:

items with nearby addresses tend to be referenced close together in time

Locality Example

```
sum = 0;
for (i=0; i<n; i++)
sum += a[i];
return sum;
```

spatial locality

temporal locality.

(I) Data references
  (i) Reference array elements in succession
    (stride-1 reference pattern)
  (ii) Reference variable sum each iteration.

(II) Instruction references
  (i) Reference instructions in sequence
  (ii) Cycle through loop repeatedly

Qualitative estimates of locality:

Claim: Being able to look at code and get a qualitative sense of its locality is a key skill for a professional programmer.

Question: Does the fcn have good locality with respect to array a?

```
int sum_array_rows(int a[M][N])
{
    int i, j, sum = 0;
    for (i = 0; i < M; i++)
        for (j = 0; j < N; j++)
            sum += a[i][j];
    return sum;
}
```

temporal and spatial quality both good.

if i & j exchanged awful (:

skipping in memory

Memory Hierarchies

(I) Some fundamental and enduring properties of hardware and software:

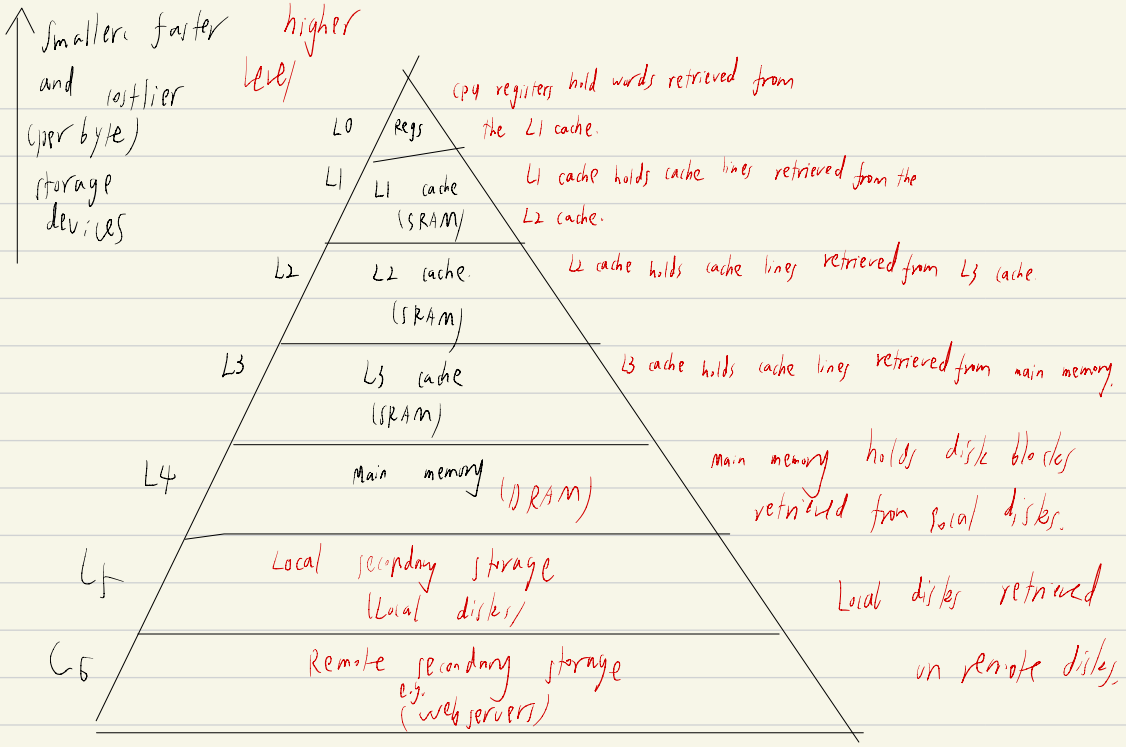
(i) fast storage technologies cost more per byte, have less capacity, and require more power (heat!).

(ii) The gap between CPU and main memory speed is widening.

(iii) Well-written programs tend to exhibit good locality.

(II) These fundamental properties complement each other beautifully.

(III) They suggest an approach for organizing memory and storage systems known as a memory hierarchy.


Smaller, faster, and costlier (per byte) storage devices
higher level
L0
Regs
CPU registers hold words retrieved from the L1 cache.
L1
L1 cache (SRAM)
L1 cache holds cache lines retrieved from the L2 cache.
L2
L2 cache (SRAM)
L2 cache holds cache lines retrieved from L3 cache.
L3
L3 cache (SRAM)
L3 cache holds cache lines retrieved from main memory.
L4
Main memory (DRAM)
Main memory holds disk blocks retrieved from local disks.
L5
Local secondary storage (local disks)
Local disks retrieved from remote disks.
L6
Remote secondary storage e.g. (web servers)
Larger, slower, and cheaper (per byte) storage devices
lower level

Caches.

(comparison: your backpack in school)

(I) Cache: A smaller, faster storage device that acts as a staging area for a subset of the data in a larger, slower device.

(II) Fundamental idea of a memory hierarchy:

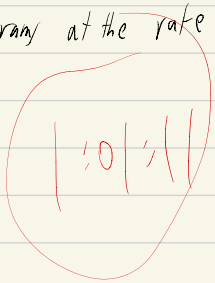
(i) For each k, the faster, smaller device at level k serves as a cache for the larger, slower device at level k+1.

(III) Why do memory hierarchies work?

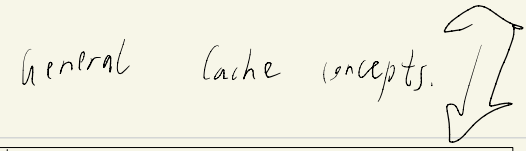
(i) Because of locality, programs tend to access the data at level k more often than they access the data at level k+1.

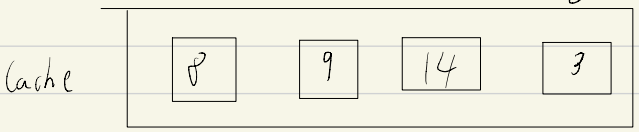
(ii) Thus, the storage at level k+1 can be slower, and thus larger and cheaper per unit.

Big idea: the memory hierarchy creates a large pool of storage that costs as much as the cheap storage near the bottom, but that serves data to programs at the rate of level k. fast storage near the top.

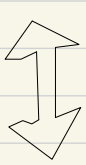


General Cache concepts.

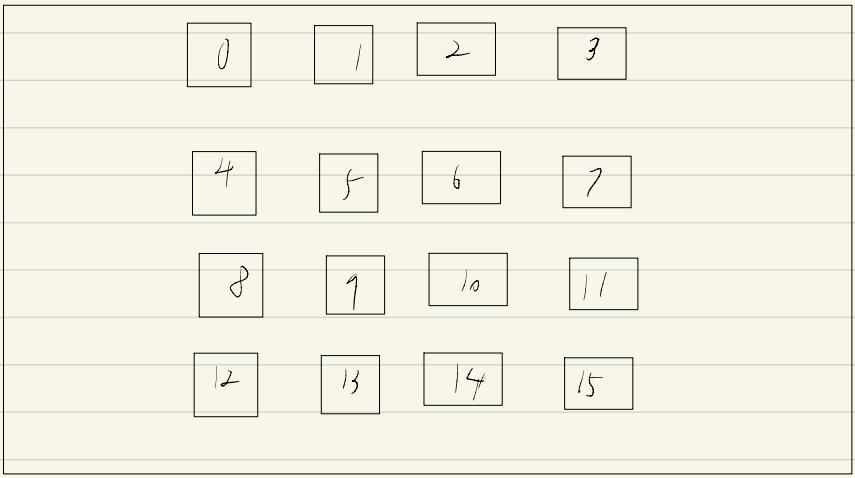




Smaller, faster, more expensive memory caches a subset of blocks.



Data is copied in block-sized transfer unit.



Larger, slower, cheaper memory viewed as "partitioned" into blocks

General caching concepts:

Types of cache misses

(I) Cold (compulsory) miss:

(i) Cold misses occur because the cache is empty.

(II) Conflict miss:

(i) Most caches limit blocks at level k+1 to a small subset (sometimes a singleton) of the block positions at level k.

e.g. block i at level k+1 must be placed in block (i mod 4) at level k.

(ii) Conflict misses occur when the level k cache is large enough, but multiple data objects all map to the same level k block.

e.g. Referencing blocks 0, 8, 0, 8, 0, 8, ... would miss every time.

(III) Capacity miss

(i) Occurs when the set of active cache blocks (working set) is larger than the cache

Summary:

(i) The speed gap between CPU, memory and mass storage continues to widen.

(ii) Well-written programs exhibit a property called locality.

(iii) Memory hierarchies based on caching close the gap by exploiting locality.