Ch 14

	Capacity	Latency	Cost/GB
Register	1000s of bits	20 ps	$$$$
SRAM	~10 KB-10 MB	1-10 ns	~$1000
DRAM	~10 GB	80 ns	~$10
Flash*	~100 GB	100 us	~$1
Hard disk*	~1 TB	10 ms	~$0.10

Processor Datapath

Memory Hierarchy

I/O subsystem

* non-volatile (retains contents when powered off)

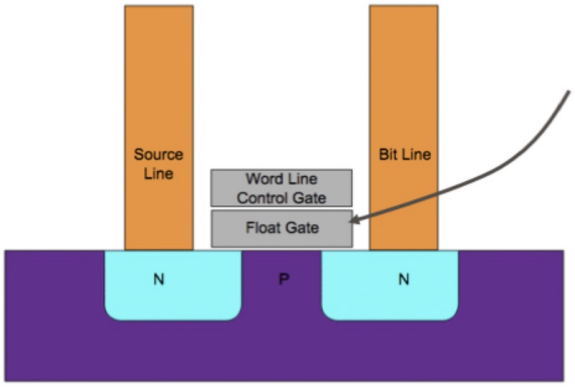
Static RAM

- array of $k * b$ cells (k words, b cells per word)
- cell is a bistable element + access transistors
  - ↳ analog circuit with carefully sized transistors to allow reads and writes
- Read: precharge bitlines, activate wordline, overpower cells
- Write: drive bitlines, activate wordline, overpower cells

Dynamic RAMs

- 1T DRAM cell: transistor + capacitor
- smaller than SRAM cell, but destructive reads and capacitor leaks charges
- DRAM arrays include circuitry to:
  - write word again after every read (to avoid losing data)
  - refresh (read + write) every word periodically
- DRAM vs SRAM
  - ~20x denser than SRAM
  - ~2-10x slower than SRAM

Non-Volatile Storage: Flash



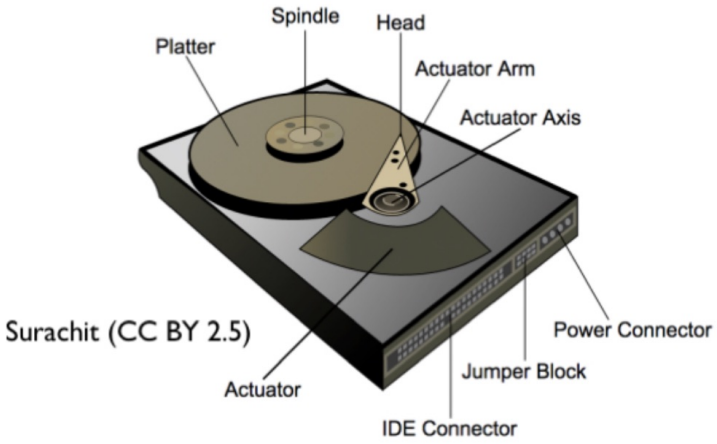
Electrons here diminish strength of field from control gate ⇒ no inversion ⇒ NFET stays off even when word line is high.

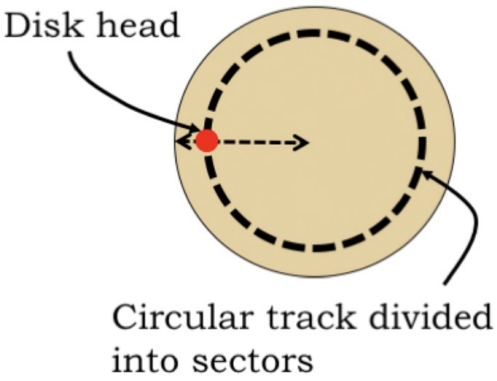
Cyferz (CC BY 2.5)

Flash Memory: Use "floating gate" transistors to store charge

- Very dense: Multiple bits/transistor, read and written in blocks
- Slow (especially on writes), 10-100 us
- Limited number of writes: charging/discharging the floating gate (writes) requires large voltages that damage transistor

Non-Volatile Storage: Hard Disk





Hard Disk: Rotating magnetic platters + read/write head

- Extremely slow (~10ms): Mechanically move head to position, wait for data to pass underneath head
- ~100MB/s for sequential read/writes
- ~100KB/s for random read/writes
- Cheap

Locality Principle

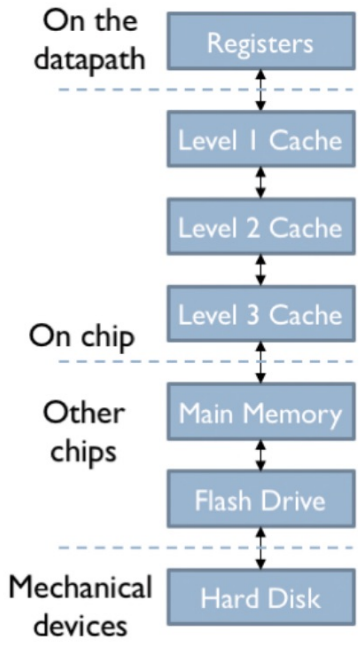
- Access to address X at time t implies that access to address $X + \Delta X$ at time $t + \Delta t$ becomes more probable as $\Delta X, \Delta t \rightarrow 0$.

Cache

- interim storage (transparent)
  - recently accessed location
  - exploits locality principle

A Typical Memory Hierarchy

- Everything is a cache for something else…



Access time	Capacity	Managed By
1 cycle	1 KB	Software/Compiler
2-4 cycles	32 KB	Hardware
10 cycles	256 KB	Hardware
40 cycles	10 MB	Hardware
200 cycles	10 GB	Software/OS
10-100us	100 GB	Software/OS
10ms	1 TB	Software/OS

average memory access time

Summary: Cache Tradeoffs

data size

$$AMAT = HitTime + MissRatio \times MissPenalty$$

- Larger cache size: Lower miss rate, higher hit time
- Larger block size: Trade off spatial for temporal locality, higher miss penalty
- More associativity (ways): Lower miss rate, higher hit time
- More intelligent replacement: Lower miss rate, higher cost
- Write policy: Lower bandwidth, more complexity
- How to navigate all these dimensions? Simulate different cache organizations on real programs

addresses can be in several locations

↳ replacing specific cache lines

how to manage memory writes in the cache