Memory

- all data, info and code is stored here

Semi-conductor based

- solid state drive
- CF, SD, mini SD
- memory IC chips
- solid state drive (SSD)
- thumb drive

magnetic-based

- magnetic HD
- HDD

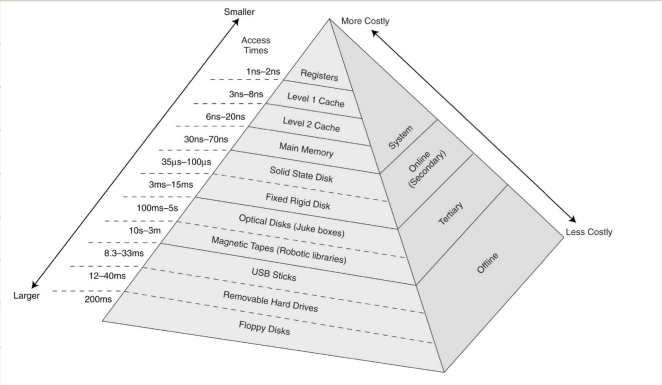
- HDD is being replaced by SSD in laptops these days but is used as the main storage element in Data centers used for cloud storage

- support XIP (execute in place) ← run code directly from these memory

processor	system memory	storage memory
CPU Reg Cache Internal memory	NOR flash memory DRAM SRAM	HDD SSD Storage ICs
↔	info transfered during runtime ←	
	contains code/data used during run-time Subset of active program running	contains all code/Data of all program and data in comp system

- function v/s cost

fastest access time → due to higher cost, smaller



Volatile v/s Non-Volatile

data is lost when power is removed	data is retained even when power is lost
temporary storage	permanent storage
system memory usually	typically main storage
SRAM, DRAM	FLASH, magnetic Hard disk

Semiconductor Memories

- based on semiconductor integrated circuits (IC)

- used in → processor internal memories
  → system memory

- types of memory
  → SRAM
  → DRAM
  → Flash memory
  → SSD → based on

  SRAM, DRAM, FLASH based

  - registers
  - buffers
  - cache
  - int system memory
  - storage memory

bulk storage memory
(in an array)
consisting of FLASH memory ICs

Volatile Memory

① RAM : Random Access Memory

- **static RAM**
  - → data is stored as long as power supply is applied
  - → large 4-6 transistors
  - → fast access time
  - → low power consumption (both active and standby)

  wrt DRAM

- dynamic RAM
  - → periodic refresh required
  - → small (1 capacitor + 1 transistor)
  - → slower
  - → high power consumption due to periodic refresh to maintain data integrity.

to use :	write	read
1. address		
2. chip select (CS)	· address	· address
3. write enable (WE)	· CS	· CS
4. output enable (OE)	· WE	· OE
5. databus		

- memory takes up the maximum space in most systems / microprocessors.

- SRAM has an array of memory cells and each cell contains 1-bit

  ↳ every row of cells makes 1 word (8 cells) = 1 byte

  ↓

  enabled by word line which is decoded from address

- out of the 6 transistors / per cell:

→ 4 form the invertor logic which basically stores the logic state of said cell.

→ 2 control the data in & out of the cell.

- **DRAM**

- 1 transistor
  ↳ controls flow of charges in & out
  1 capacitor
  ↳ holds charge which is used to determine the logic value.

- write : by enabling transistor (both cases)
  1 ⇒ charge capacitor
  0 ⇒ discharge capacitor

- read : enable transistor, read charge of capacitor w/ sense amplifier
  ↓
  destroys the data
  ↳ og data must be rewritten after every read.

- capacitor charges leak over time → ∴ periodic refreshes are required

- we use Synchronous Dynamic Random Access Memory (SDRAM) now
  ↳ has a clock to sync data transfer
  ↳ employs pipeline architecture

Non-Volatile Memory

- **EPROM** : erasable programmable ROM ← UV to erase
  EERROM : electrically erasable programmable ROM ← electrically possible

- **Floating gate transistor** (FGT) in MOSFET

2 states of FGT:
- Erased State : electrically neutral
- Programmed state : excess electrons in FGT
  ↳ stay even when power is removed.

- to program, a large positive voltage is applied.

- let $\overset{V_1}{\cancel{x}}$ V be the threshold in the erased state to turn the FGT on.
  once that voltage is applied and FGT is in the programmed state, the threshold increases to V_2

- erased state · logic 1 : no charges : transistor ON
  prog. state · logic 0 · contains charges : transistor OFF

- checking of state : inject a voltage V : $V_1 < V < V_2$

  if the FGT is ON ⇒ erased state
  OFF ⇒ prog state

- "programming" ⇒ modifying cell content from 1 to 0
  "erasing" ⇒ 0 → 1

- **FLASH** : also based on FGT → NAND / NOR

comp. to EEPROM
- can be erased in larger block size
- faster access speed
- costs less

1. **NOR FLASH**

- cells are connected w/ logic similar to NOR
  any cell has voltage > V_T ⇒ erased state
  bit line O/P = 0

- supports execute-in-place or XIP
  ∴ used to store program code.
  which can be run directly from flash w/o loading into RAM

- allows random reading by address only
- writing is complicated
- erasure has to be done at block level

- can be used as system memory (store code) or general storage memory

- has separate address and data lines

2. NAND FLASH

- memory cells are connected by nand logic :
  all cells are > V_T ⇒ bit line o/p = 0

- does not allow X/P
- all memory is accessed at page level
  ↓
  (pages form a block) multiple data bytes

- less connecting wires, more compact and hence used in mass memory storage devices → USB, mobile, SSD

- uses one combined databus for I/O
  I : command, address, data
  O : data.

EEPROM	FLASH
more expensive	faster erasure · larger capacity

System v/s Storage Memory

1. System Memory

- stores run-time program and code is executed directly
- can be :
  internal : SRAM / DRAM / NOR Flash
  external : NOR Flash
- DRAM doesn't inherently support XIP but interface help the controller make it work

2. Storage Memory

- stores all code and programs of the computer system
- code cannot be run from here
- any and all kinds of memory can be used.

end of semiconductor memory

main mass storage → magnetic HDD
→ semiconductor SDD

Magnetic HDD

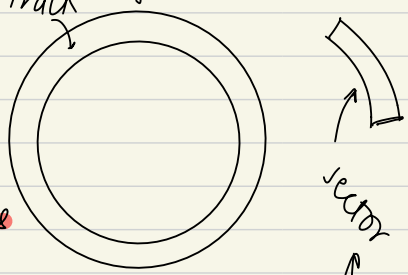
- records data by changing orientation of tiny magnetic elements

- longitudinal recording : cells are on the surface
  perpendicular recording : cells are ⊥ to disk surface

- disk ⇒ platter
  ↳ data is stored on both surfaces

sector : header, data, trailer
header → sync info
trailer → error correction code

track = cylinder (numbers)



sector → minimum block size

no of head = surfaces
head * cylinder = total no of tracks

- HDD transfer rate

a) seek time (T_S) → time to track
b) Rotational Delay (T_R) → time to sector
c) Access Time (T_A) → time from request to when pointer is at head.
d) Transfer Time (T_T) → time req to transfer data after head is in position

$T_A = (T_S + T_R)$
total time : $T_A + T_T$

- T_R depends on RPM (convert to RPS)

$$T_R = \frac{0.5}{RPS}$$ (avg T_R)

- T_T = time to traverse target sector

$D_t * D_s$ = bytes per track
N = no of bytes to read.
$N / D_t * D_s$ = data to read / data on track = $\frac{\text{time taken to read data}}{\text{time of one rotation}}$

$$T_T = \frac{N}{RPS * D_T * D_s}$$

D_T : track density (no of sectors / track)
D_s : no of bytes / sector

- **disk defragmentation** : ensuring data that must be accessed sequentially is stored in consecutive sectors / tracks

- **Physical layout** : instead of having same no of sectors / track,
size of sectors is now kept constant
∴ more sectors on the outer tracks

zone bit recording format
- better use of recording media
- more complex

- addressing schemes : i) CHS : cylinder, head, sector
  ↑ redundant now

  ii) LBA : logical block addressing
  ↓ 48 bits ↓ allows for 128 PByte addressing (2^{50} bytes)
  ↳ linear addressing
  First block : LBA = 0
  second : LBA = 1
  and so on

- physical vs logical layout : logical is more standardised : physical is dependent on type of HDD and not standard across devices.

- cache & buffers are used for speed
- limitations
  → actuator ARM is fixed, access has to be sequential
  → vulnerable to motion
  → track to track takes time
  → reading from random sectors is hard
  → block access is better + easier

SSD : Solid State Drive

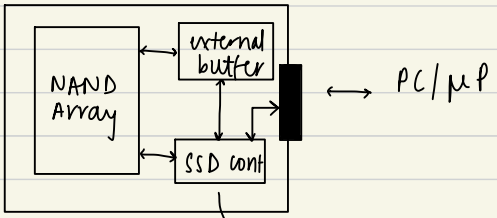
- based on NAND / NOR Flash
- more expensive than HDD
- has limited program - erase cycles
- techniques used to increase longitivity

1) wear levelling : dist data uniformly
2) use external RAM as buffer to minimise number of read and write
3) error correction code

↑

w/ all this mean time to failure benchmark · MTFB

- block diag :



microprocessor that controls SSD

- address translating from logical to physical layout
- caching + buffering
- wear levelling algorithm

HDD	v/s SSD
lower cost per bit	· pricey
· ∞ erasure cycles	· limited erasure cycles
· mechanical parts : motion sensitive	· robust to movement
· heavy, large size	· small + light
· slow transfer rate	· faster transfer rate

— x —

· EEPROM > FLASH
NOR > NAND
· SRAM > DRAM

· flash for large value : NAND (cheap)
· Non-vol for system memory of processor : NOR (XIP)
· memory for smartphone : DRAM (cheap + lots of Read write remove Flash)
· cache of µP : SRAM (small and fast, volatile)

· volatiles support a lot of read & write f^n
- user config : NAND (not volatile, must stay after power off)