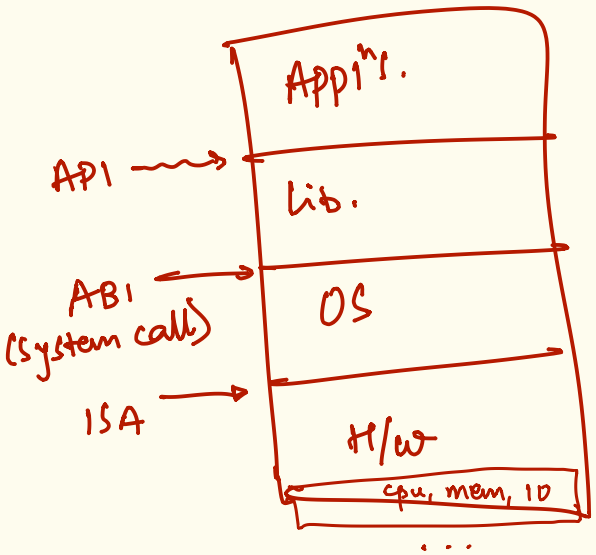
CS744 Lecture #3

* OS inside-out -

abstractions, layering



ISA — instruction set architecture

(i) instruction set of the compute (CPU) system

(ii) registers (width) — general purpose regs. ax, bx, cx ...
- control regs. cr0, cr1, cr2, cr3 ... esp ...

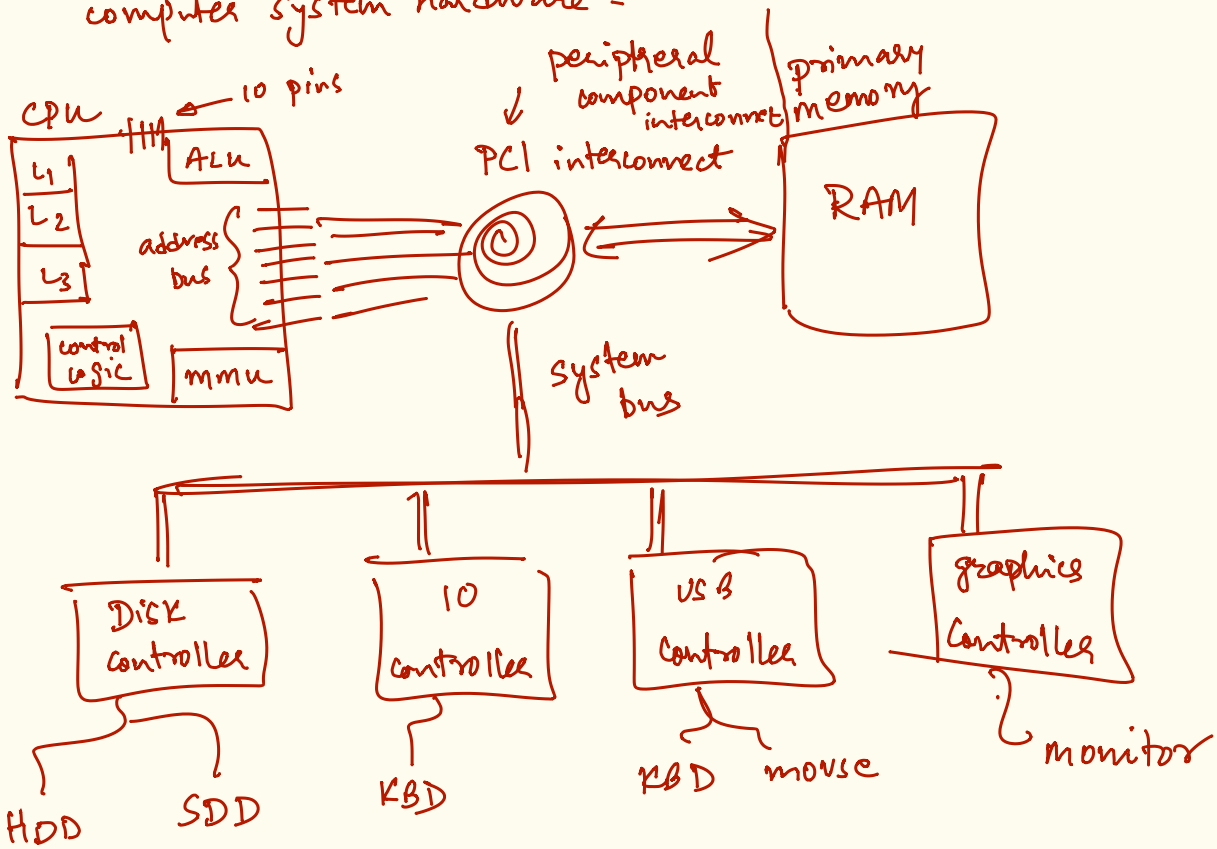
(iii) address bus width — 32 bit vs. 64-bit vs 16-bit

(iv) functionality. capabilities

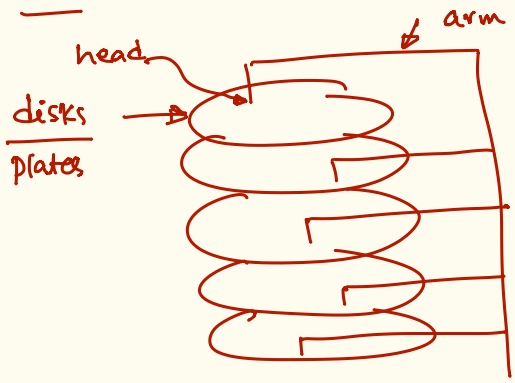
e.g.s: x86, mips, arm, powerpc ——

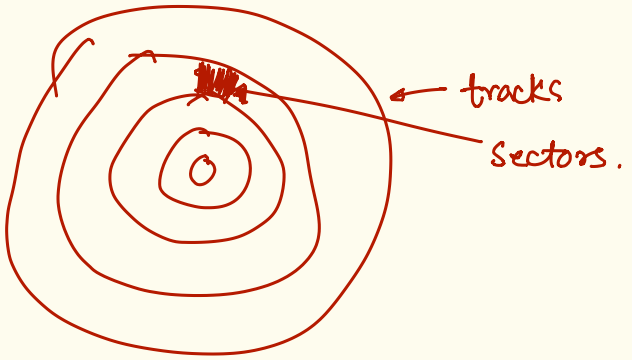
approx.

* logical view of computer system hardware -



(#) disk controller - HDD





disk controller

(i) provides a simpler abstraction ~ linearly ordered blocks.

disk#, track, sector, r/w, data info

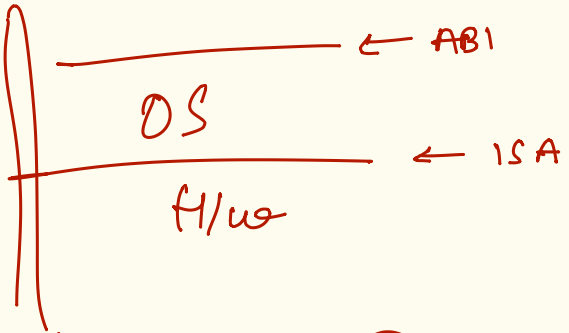
block #

(#) stored-program + data arch. — von Neumann Arch.

- decouple program execution from storage of program & data.
- fetch. decode. execute ← basic CPU seq.
- PC / regs. — program counter, instruction pointer / IP

(#) OS inside-out

(i) Why OS? / What is an OS?



~ provide useful abstractions for general purpose computing and enforce efficient usage of h/w resources. (?)

~ while managing h/w resources.

- s/w to manage all programs / tasks.
- s/w to execute programs
- s/w to manage / access to h/w
- s/w to provide safe execution environment.

--

what are examples of bad things/situations w/o an OS during program execution?

(i) - one process peeking into memory of another process. / access of data - for execution

(ii) - contention of resources — deadlock / lack of. (e.g.) — Synchronization

e.g: more than one programs update the PC regs.

(iii) - monopolizing resources

e.g CPU — PC cannot be changed by other programs.

(iv) ~ all pkts. delivered to the same appln.

(v) if a program goes down, it takes every body down.

↳ lack of isolation

(vi) ISA code lengths large = errors = f(prog. size)

#

(some) CPU

time interval regs. || stores the time of execution on the CPU | current slot.

who writes to this register ?