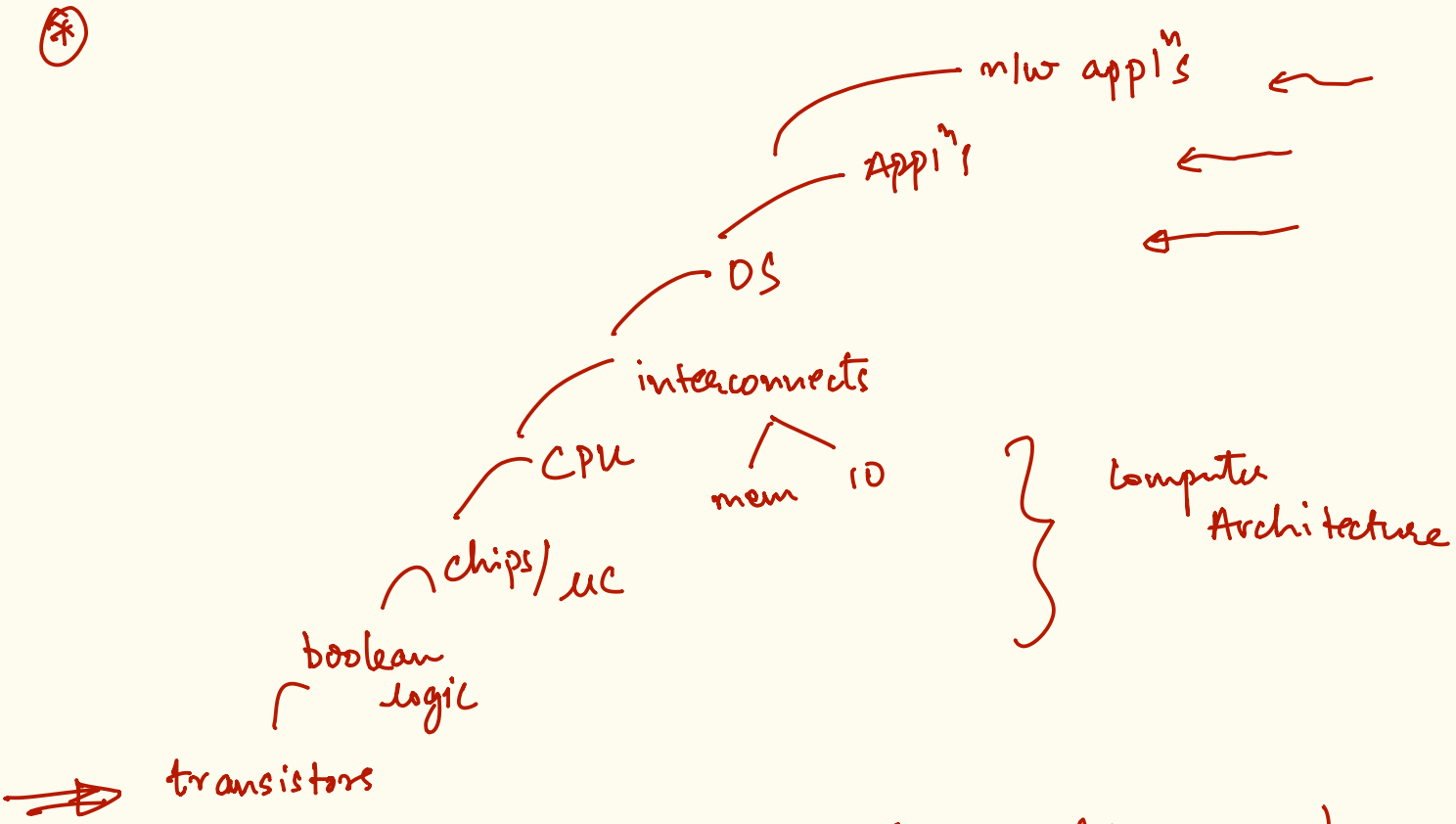
CS 744 Lecture #2.

Lab #1 due next week.

computing systems
- provide functionality / service to users (?)
  - design. build. test. repeat



* abstractions are key to world peace!
  - encapsulation of functionality
  - specification of invoking functionality
    - interface
  - implementation of functionality is opaque to users of functionality.

design principles (incomplete list).

(i) **abstractions**
- gui, API, n/w protocols, ide, device driver, system calls, logic gates, files, process

(ii) **modularity** ~ breakup into smaller tasks & interconnect. & reuse.

(iii) **layering.** ~ s/w + h/w stack
- n/w processing stack

L7	Appln
L4	TCP
L3	IP
L2	Ethernet

(iv) **caching** ~ local (near) store and & remote (far) store / location.
- CPUs, file system, web caches (CDNs)...

(v) **parallelism** ~ multiple things simultaneously. multi-core CPU executing #processes

(vi) **concurrency** ~ ~~simultaneous~~ progress of work via time sharing / context switching

(vii) **virtualization** / **multiplexing** : sharing resources across diff. entities.
- CPU & IO scheduling, cache eviction policies..

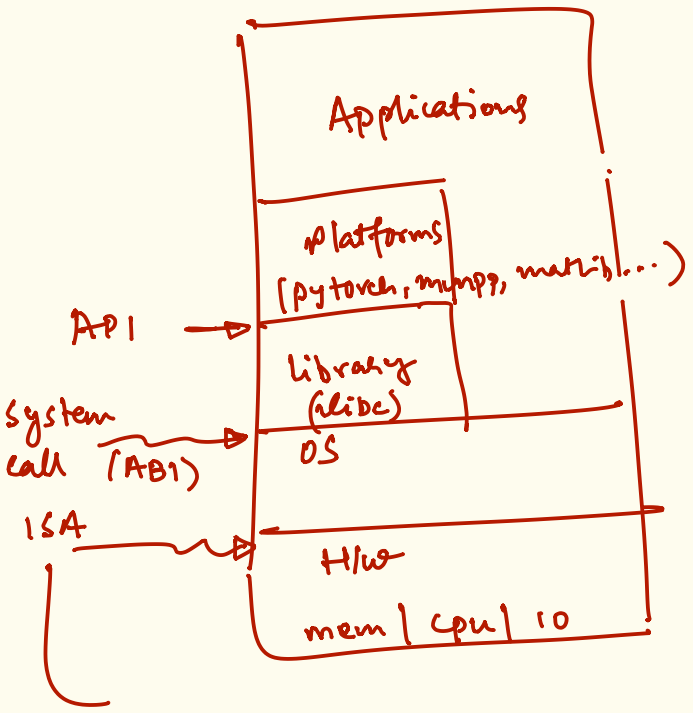
(viii) **provisioning** ~ performance $\approx f$(resources)
- e.g.: VA, ~~IP~~ URL, file offset

(viii) **indirection** - **indexing** } reference to entity name has to lookup / translated for actual identification.

- logging, pre-fetching, stateless vs stateful, offloading, cross-layer design, isolation, ...

---

② operating systems inside out.



API – appln. programming interface

ABI ?

application binary interface

} Machine Hardware

instruction set architecture
- specification of the CPU
  - instruction set
    - registers
    - IO interface

HW

API vs ABI