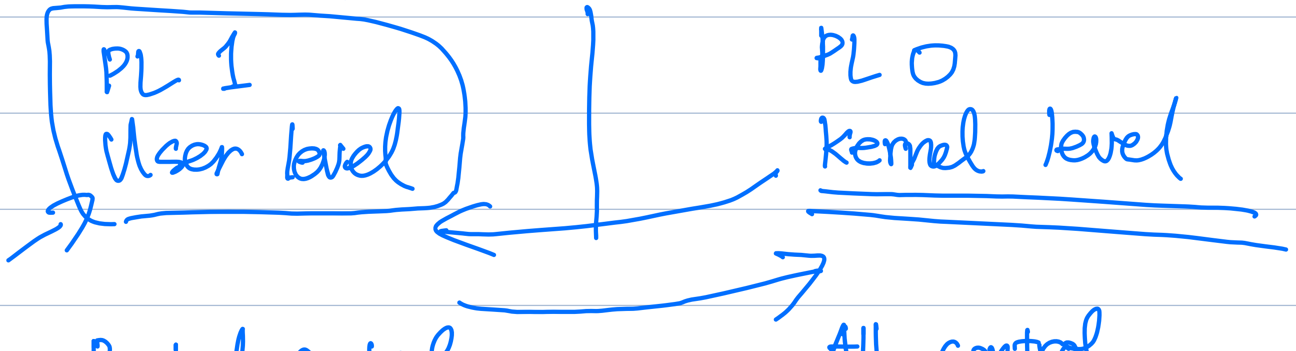
* Privilege levels.



PL 1
User level

PL 0
Kernel level

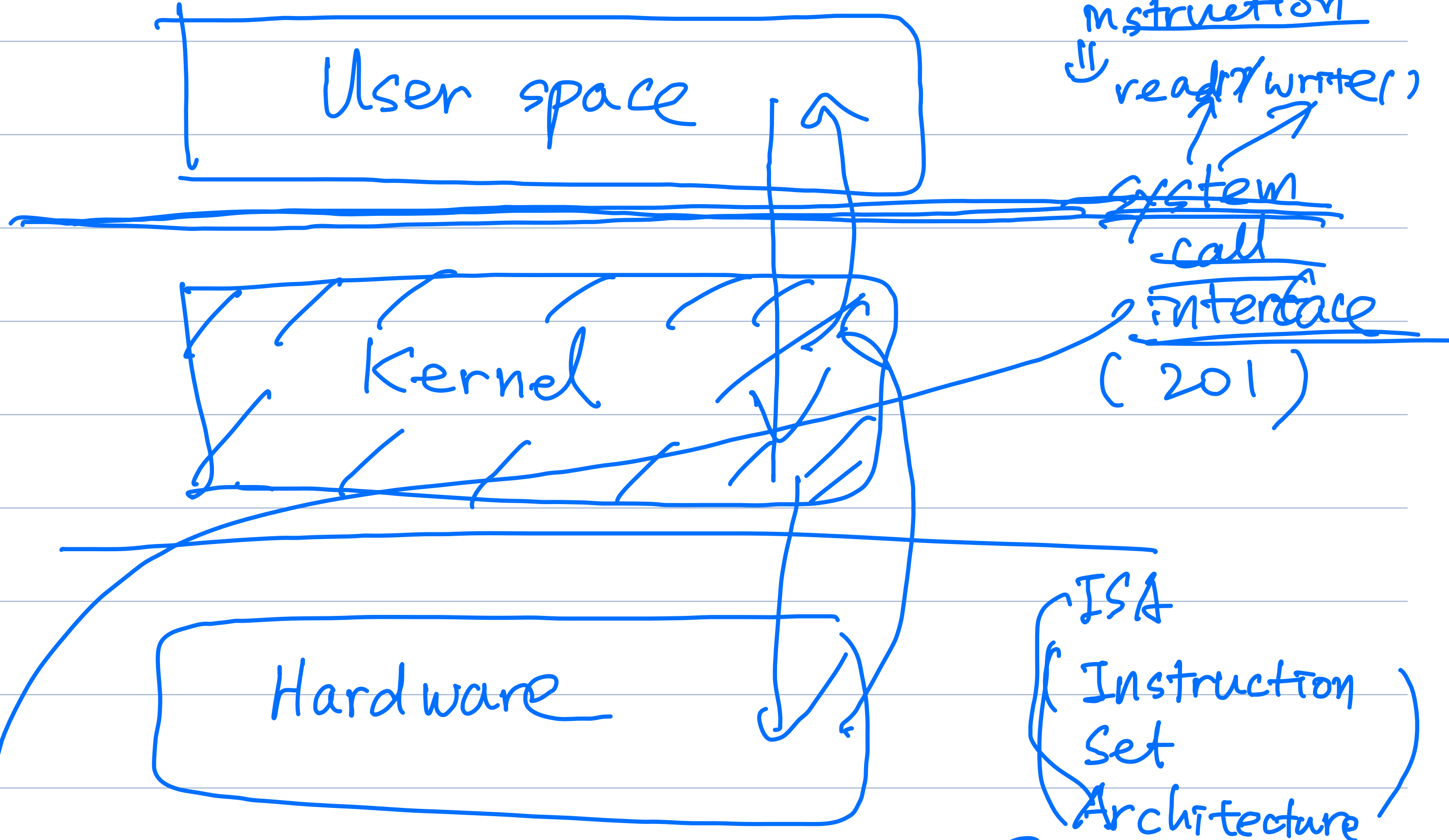
Partial control (some portion of memory; no direct access to devices, etc.)

All control (all memory, all devices (direct access))

Your OS : Kernel + user-level services.

- Hardware control.
- Program control

User programs : user-level


* hardware access instruction
⇓ read() / write()
User space
system call interface (201)
Kernel
Hardware
ISA
Instruction Set Architecture

Privileged instructions

Unprivileged instructions

*Program control : function calls

*Hardware control : function calls

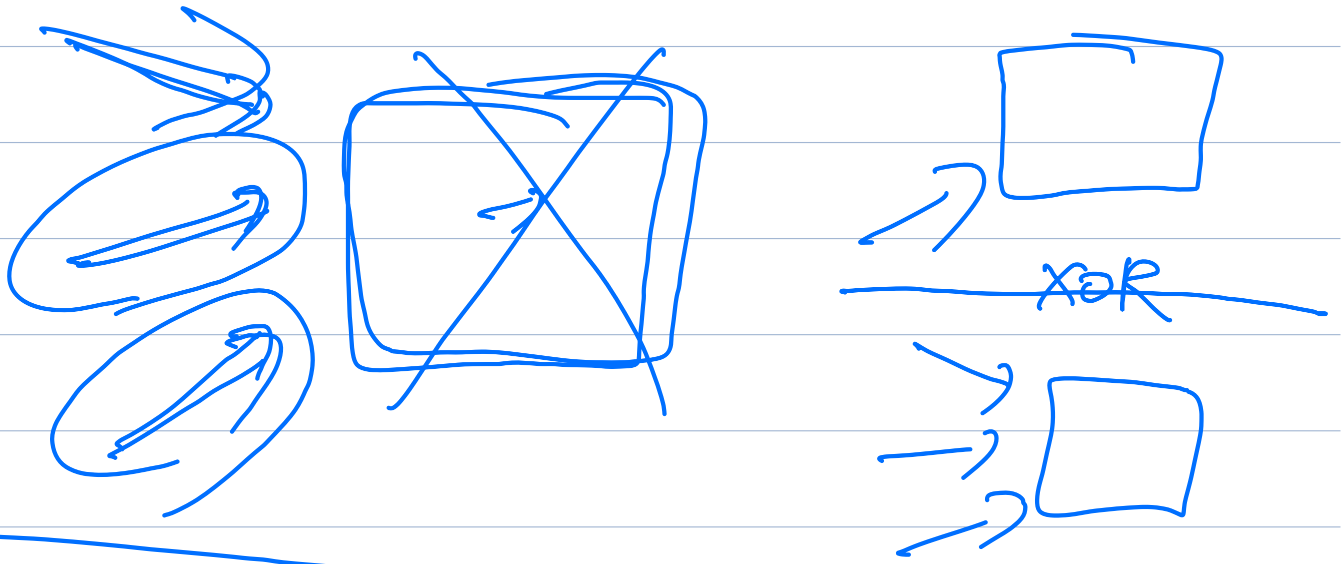
* mutability XOR aliasing

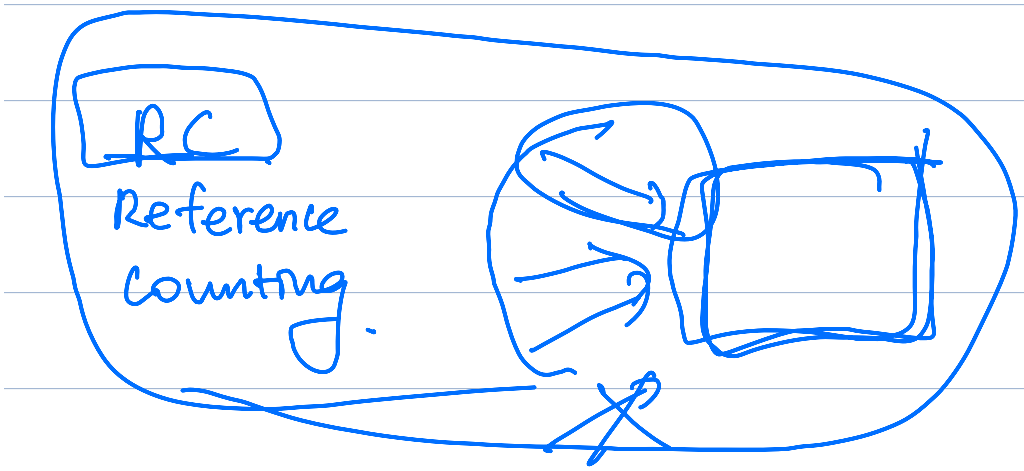
mutable

exclusive or
either X or Y
but never both

multiple pointers to

memory locations} both } a memory location





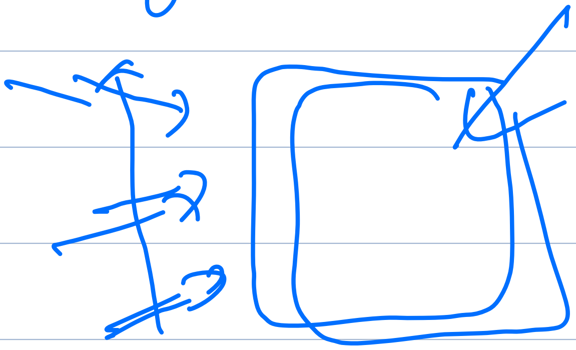
{

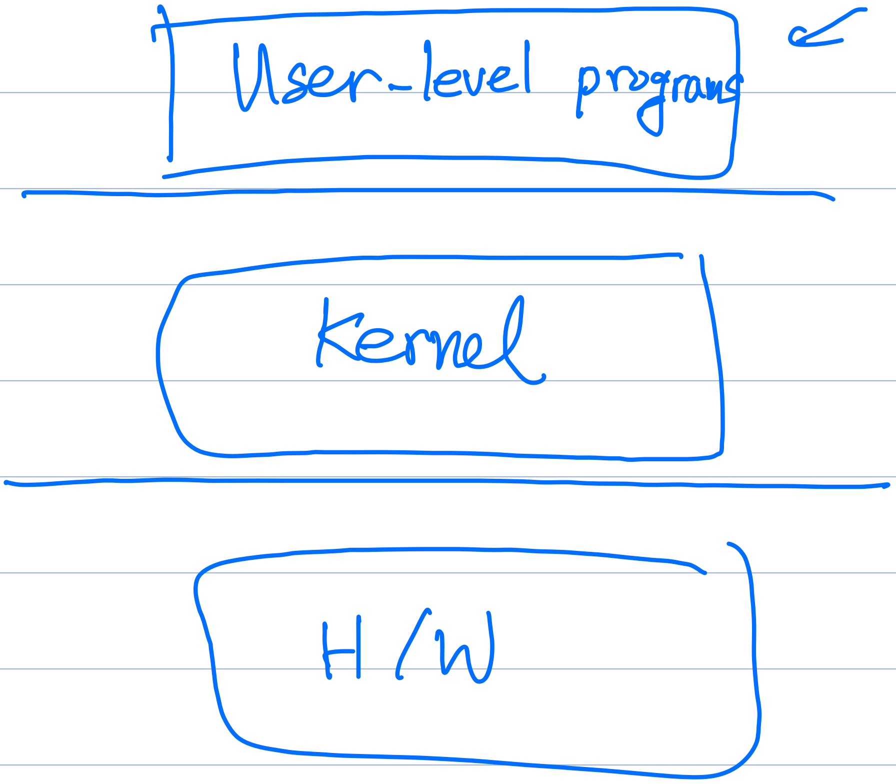
b

}
drop

mem::drop(b);

0





system calls

read()

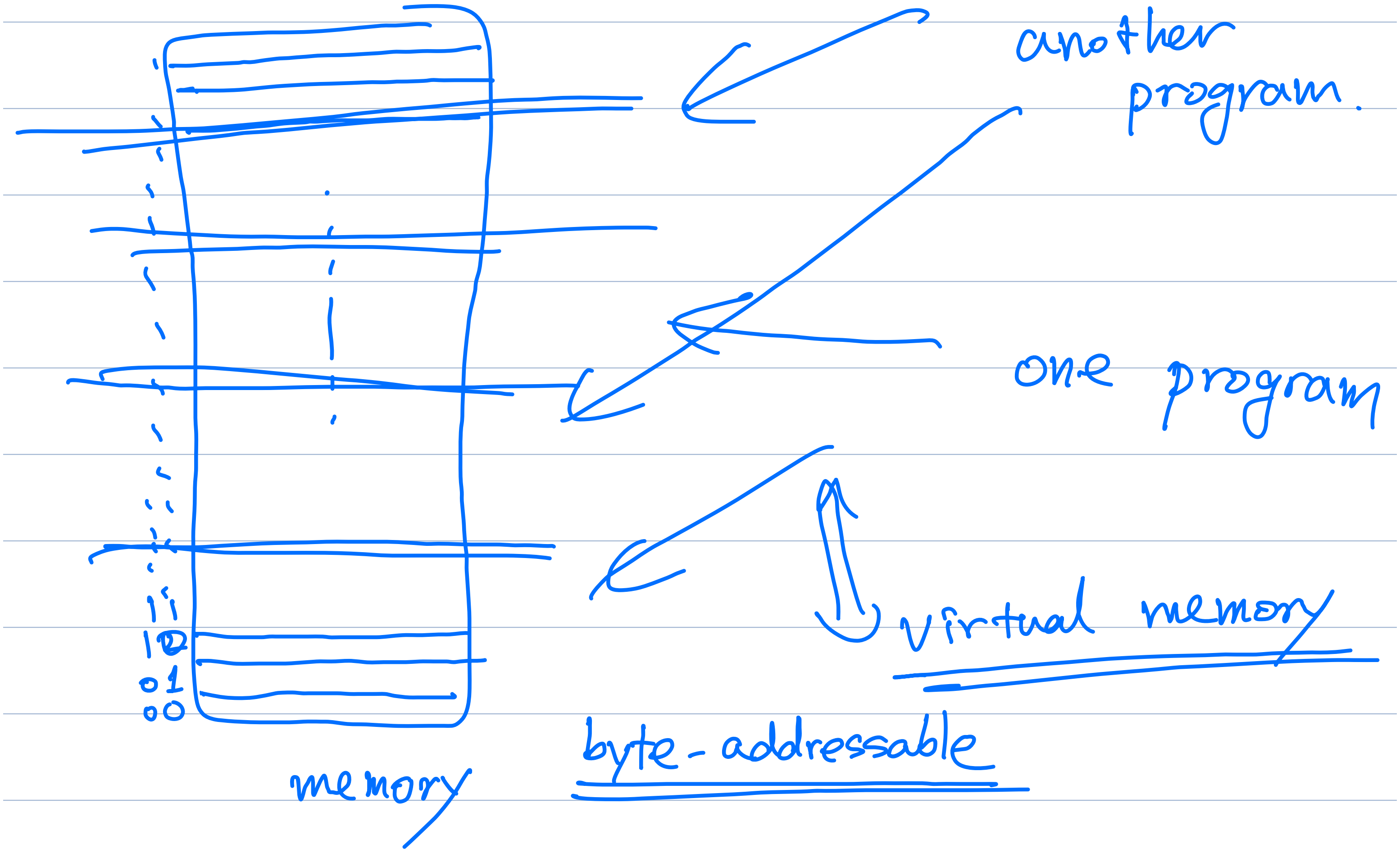
ISA = machine instructions.

Interface
- Communication : request & reply
- Contract : read(f. mode. buf. ..)
  - How to use the interface.
  - what you need to provide
  - what you're getting back
  - what it does

ISA
- Privileged instructions : kernel mode only
- Unprivileged instructions : both user mode & kernel mode.

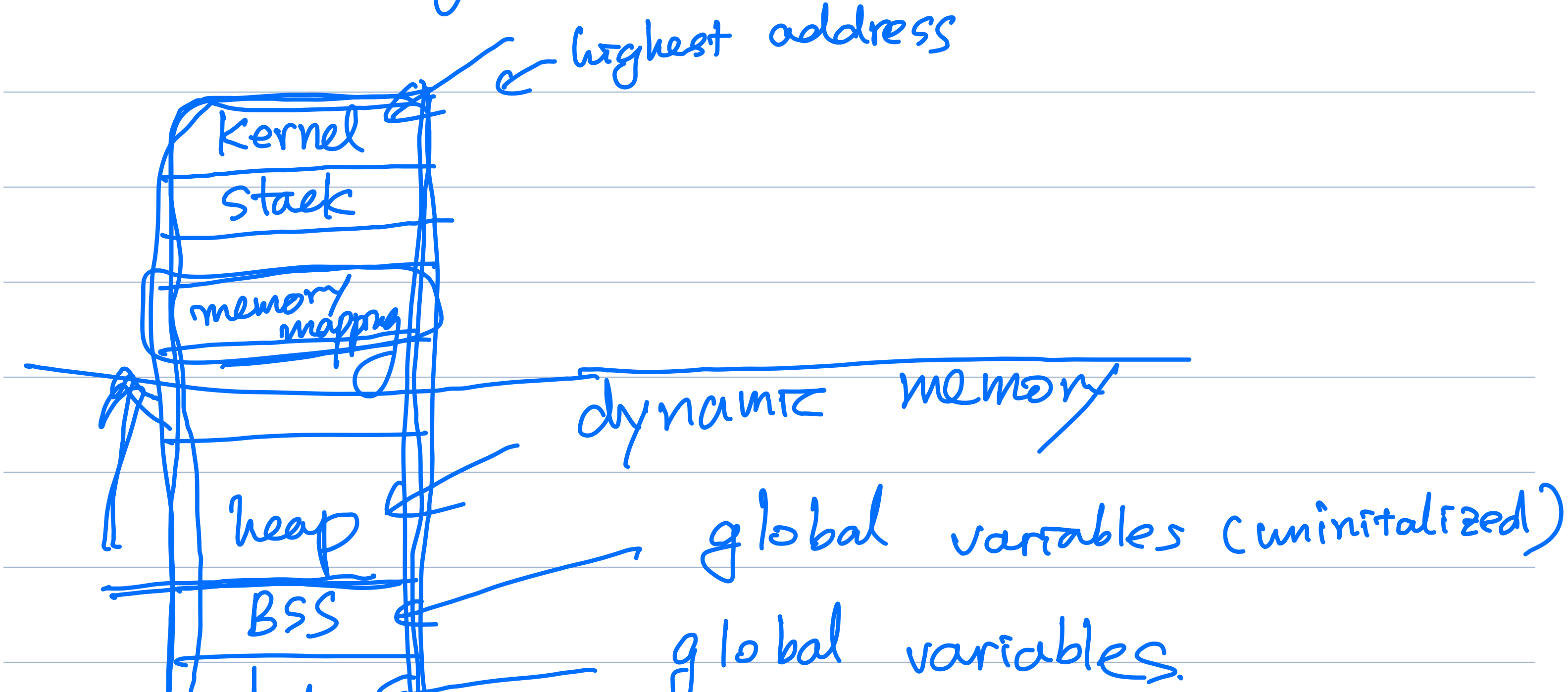
* Process

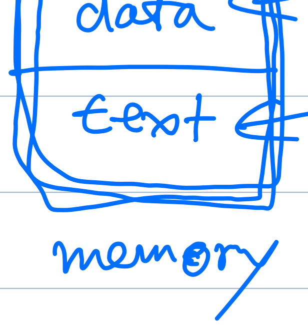
- A running instance of a program.



* Virtual memory

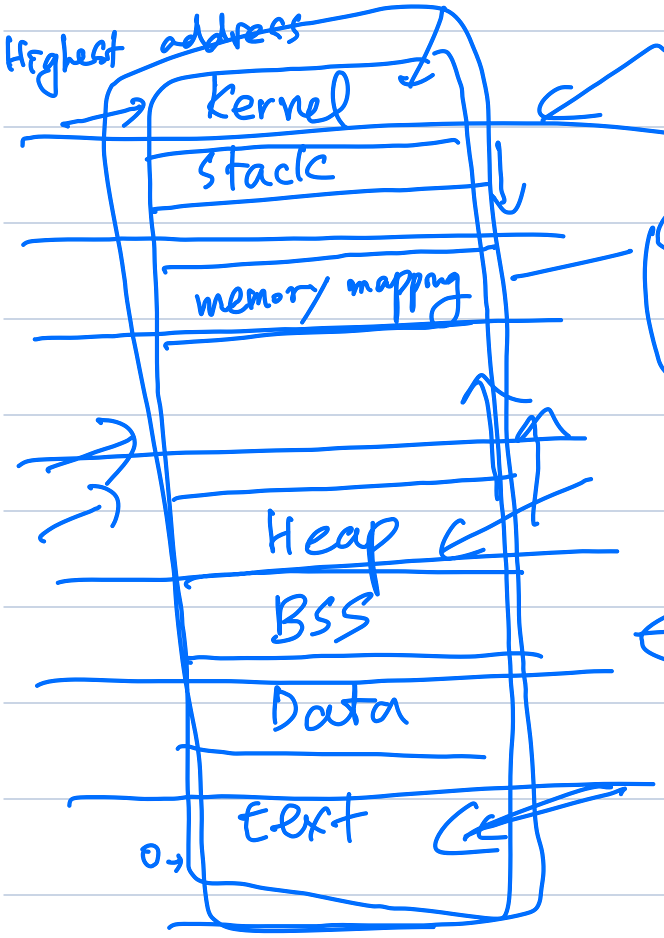
  * Illusion to each program that the whole memory is available to the program.





program code

reserved for the kernel

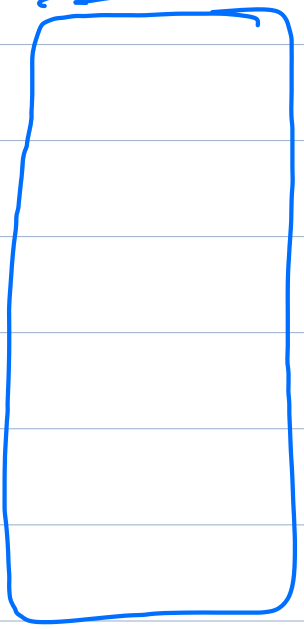


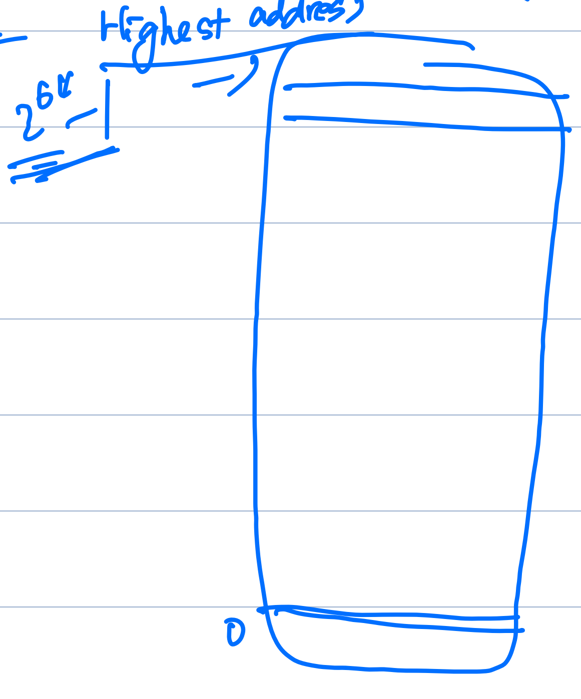
single process's address space

① File mapping
② Memory allocation (= heap)
③ Shared memory with other processes

Virtual memory space.

physical mem (8GB)

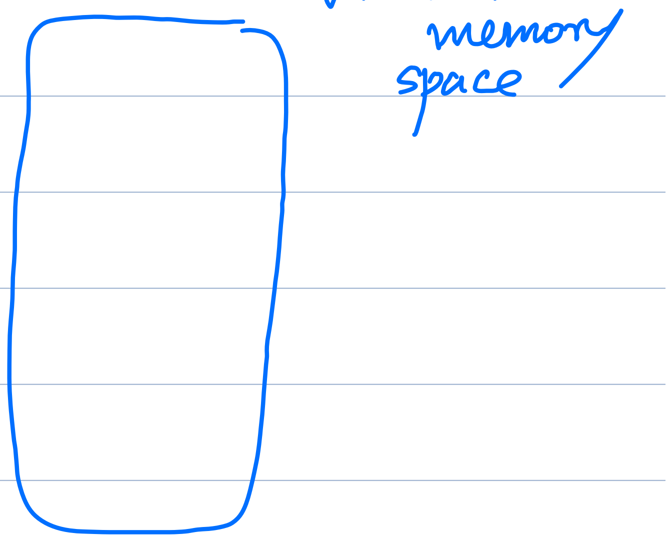




Virtual memory space (per process)

16 EB

Virtual



pointer size : 64 bits = 64-bit addresses

2^{10} : 1 KB

2^{20} : 1 MB

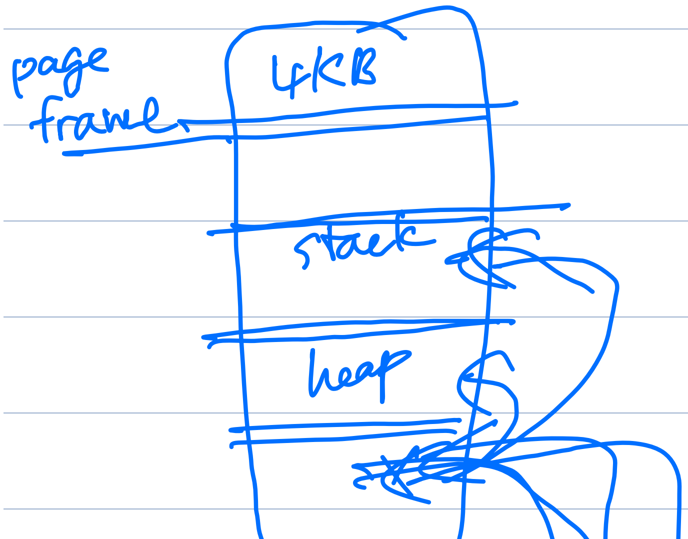
2^{30} : 1 GB

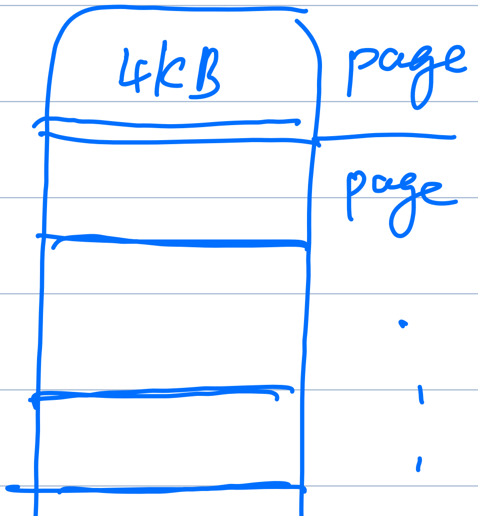
2^{40} : 1 TB

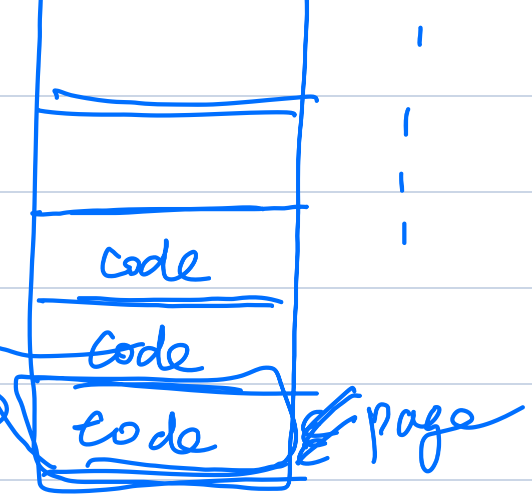
2^{50} : 1 PB

2^{60} : 1 EB

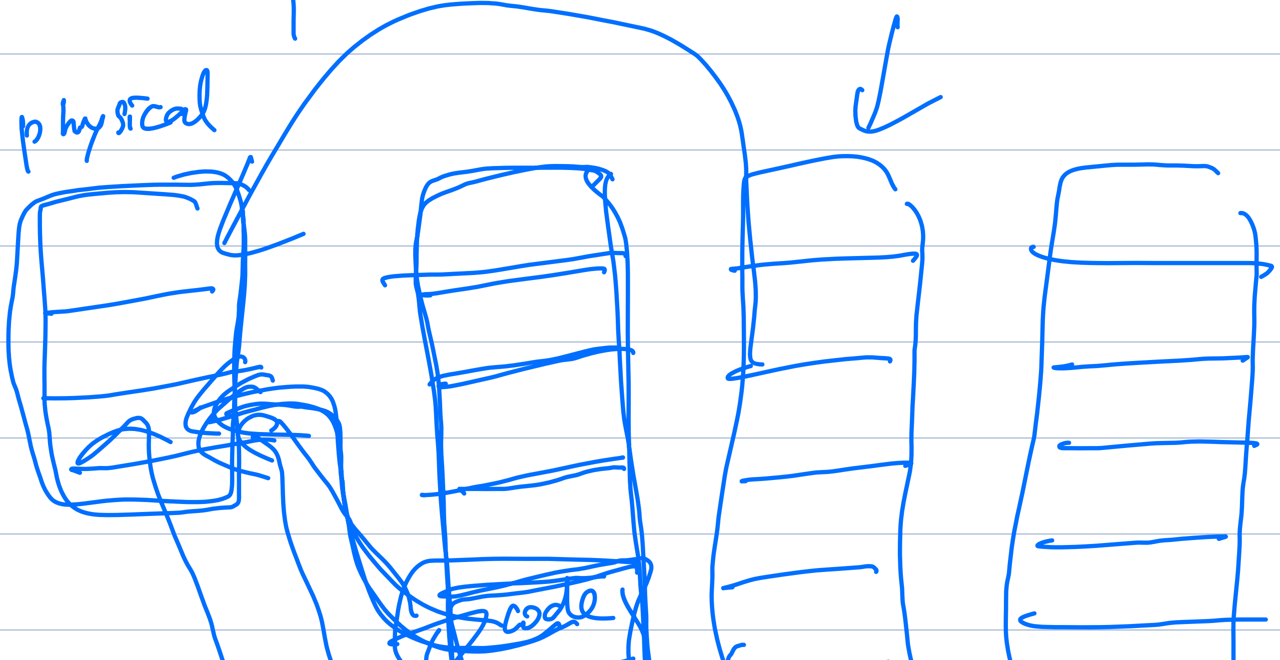
$2^{4} * 2^{60}$: 16 EB

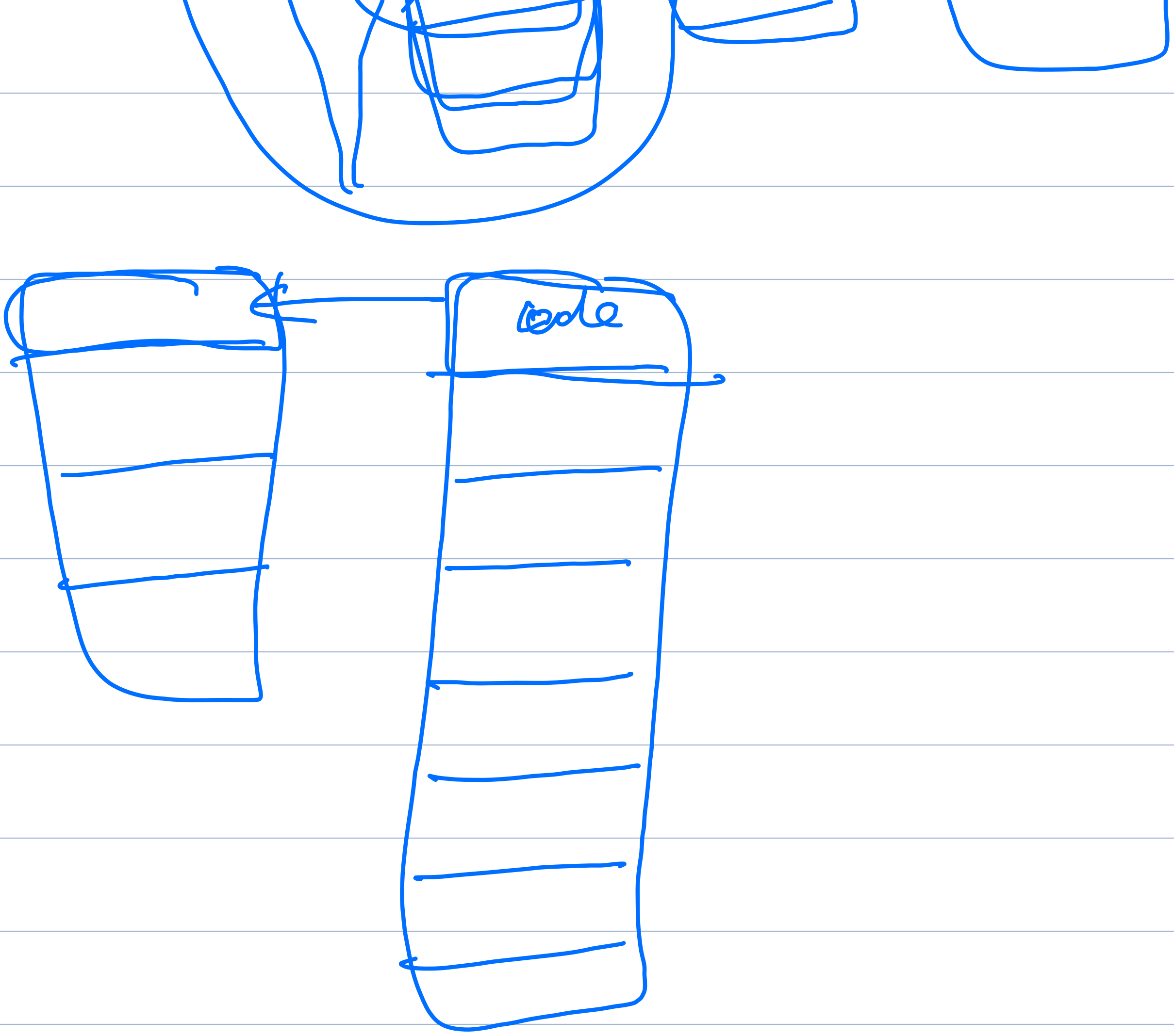






```
int arr[10] = {0};
for (i=0; i<10; i++) {
        arr[i] = 1;
    }
```




code