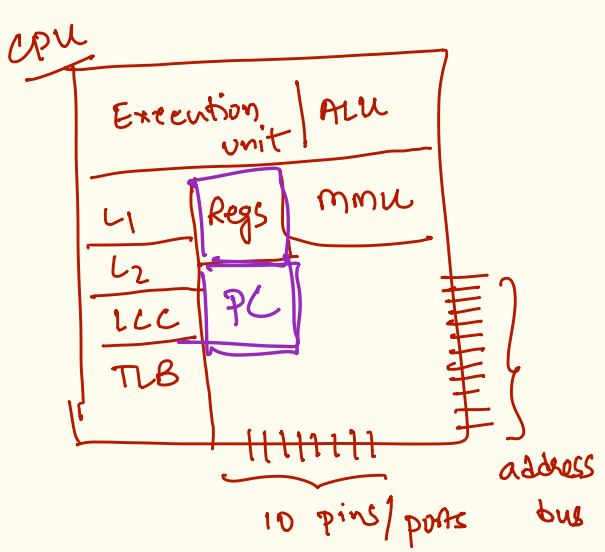
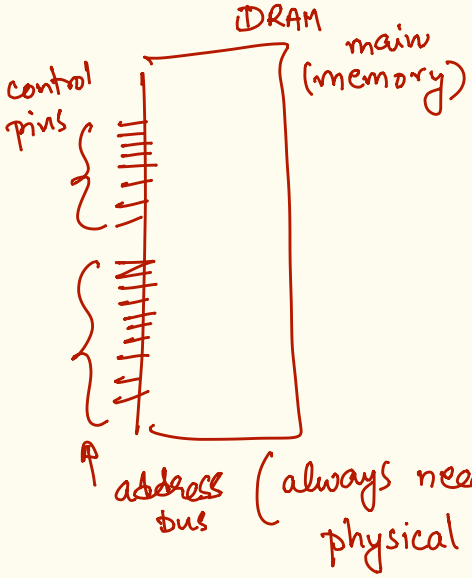
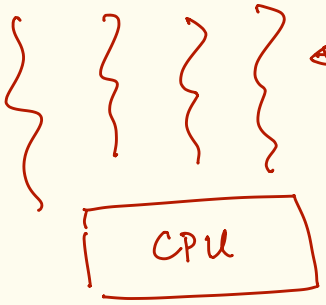
CS 744 Lecture 19

Midsem: 11th Oct 2pm



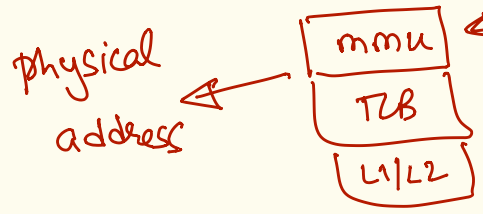




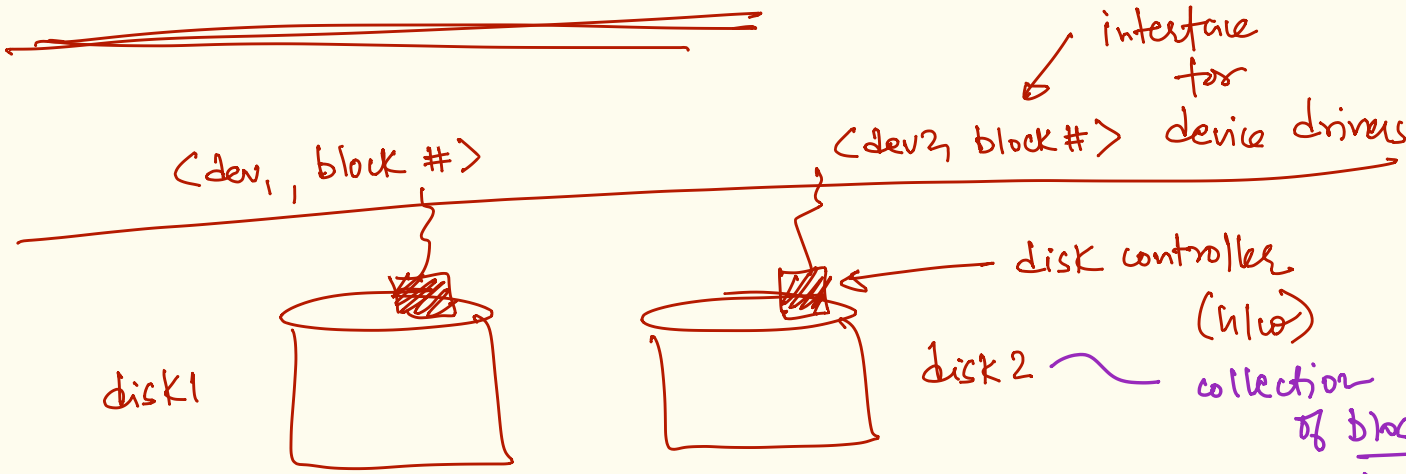
instruction sequences of
- user process
- Kernel code
  - Kernel threads
  - device drivers
  - interrupt handlers

ISA instructions
LOAD, STORE, JMP, ...

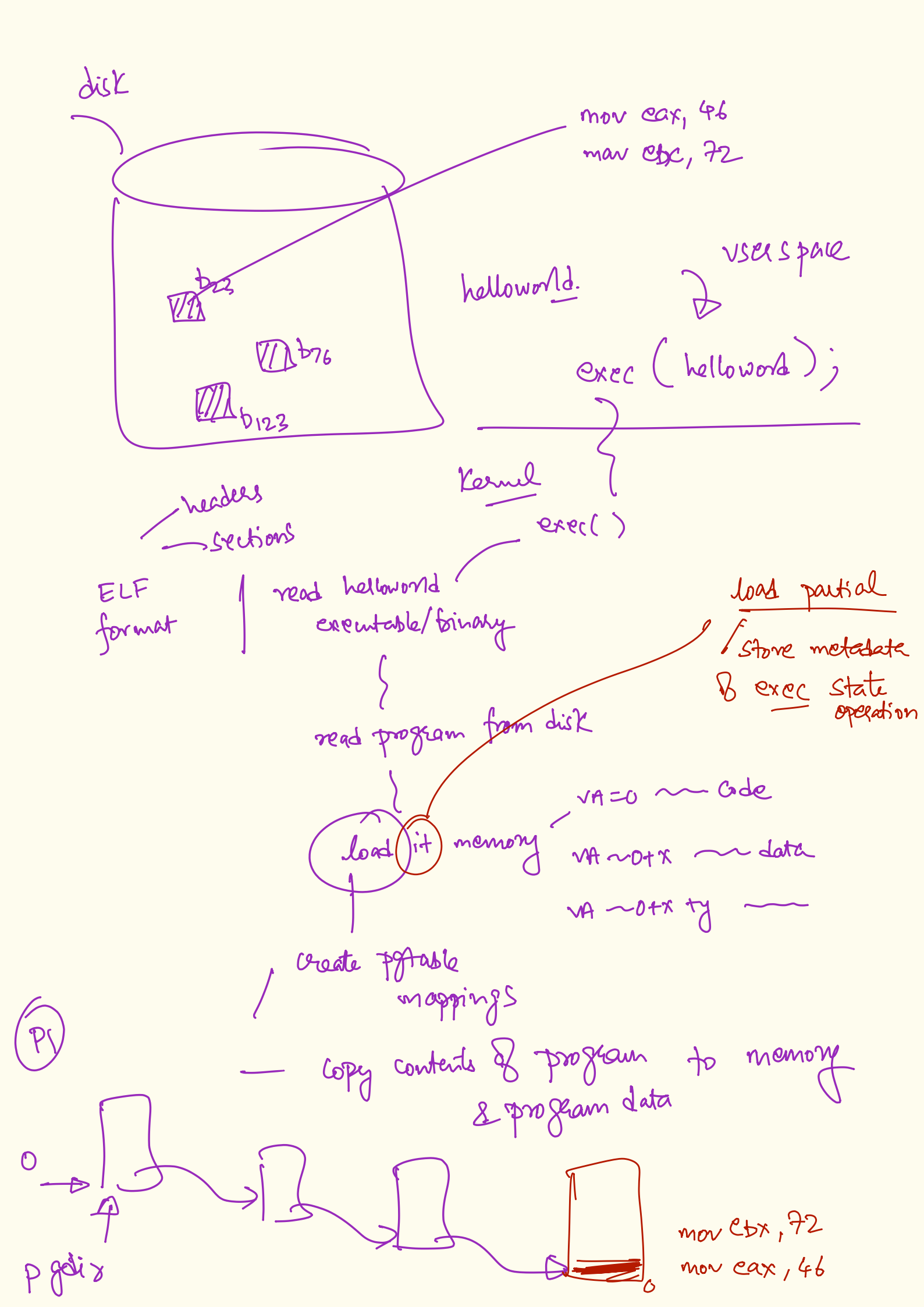
CPU (with paging) — while executing / fetching instructions issues virtual addresses

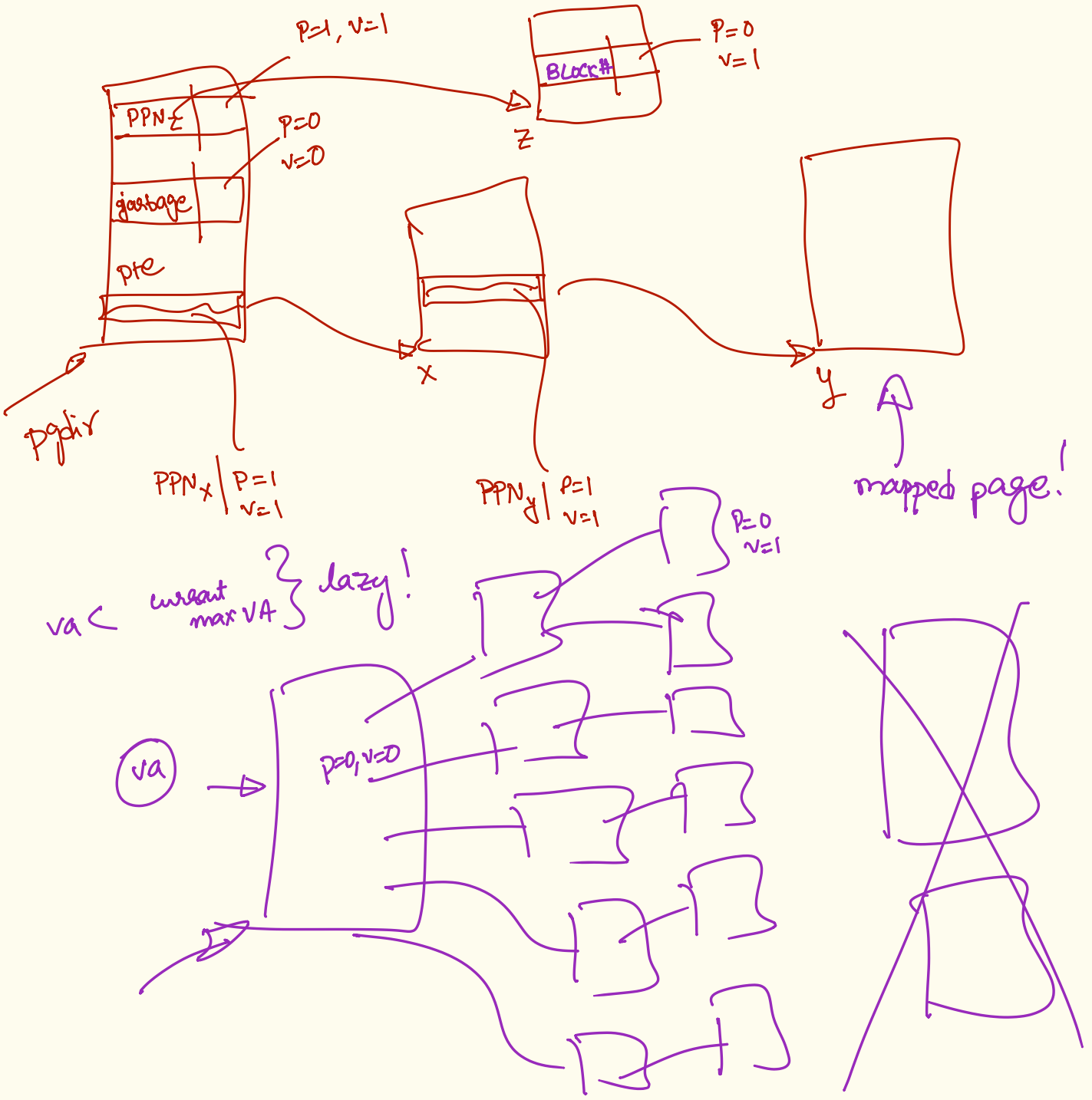


physical address → value at (virtual) address



(*) NO virtual address interface for disk.


disk
mov eax, 46
mov ebx, 72
b22
b76
b123
helloworld.
vuserspace
exec (helloworld);
Kernel
exec( )
headers
sections
ELF format
read helloworld executable/binary
read program from disk
load partial
store metadata & exec state operation
load it
memory
VA=0 ~~ code
VA ~0+x ~~ data
VA ~0+x ty ~~
create pgtable mappings
P1
— copy contents of program to memory & program data
0
pgdir
mov ebx, 72
mov eax, 46
0

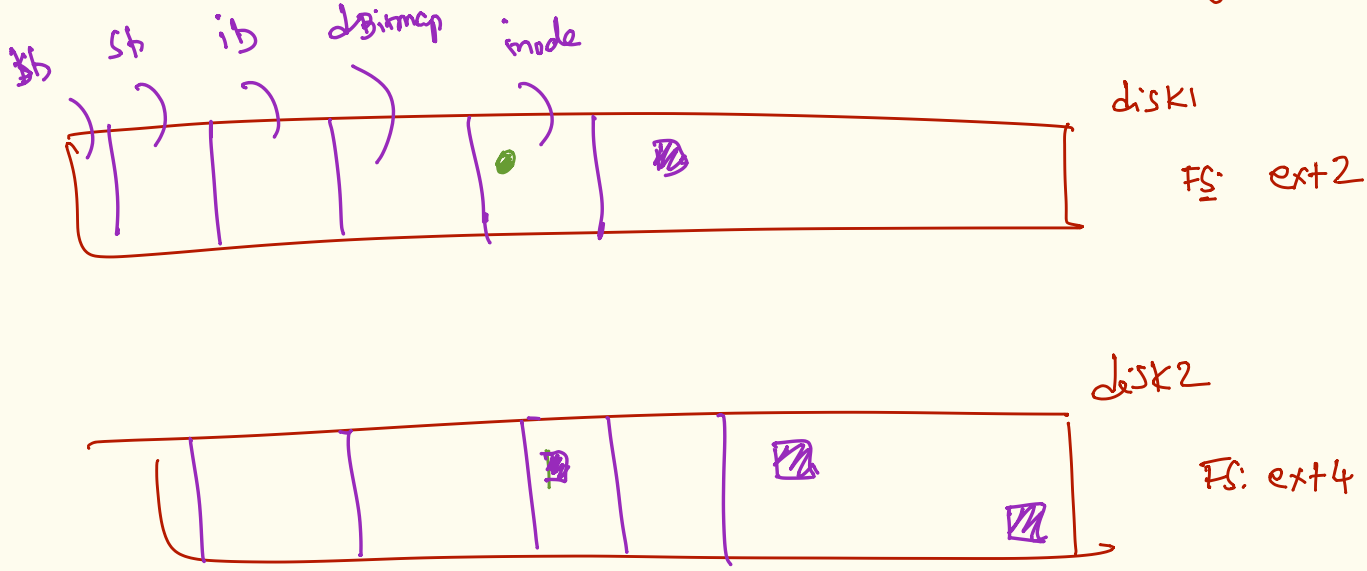

P=1, V=1
PPN z
P=0
V=0
garbage
pte
pgdir
PPN x | P=1 V=1
z
Block#
P=0
V=1
x
y
PPN y | P=1 V=1
mapped page!

va < curent max VA
lazy!
va
P=0, V=0
P=0
V=1

the file system! & the file abstraction.

```
fd = open ( filename path, options);
n= read (fd, buf, len);
n= write (fd, buf, len);
```

this interface is consistent across different file systems






```
$ cp /tmp/data/foo.txt /data/bkup/foo.txt
```

Linux : VFS ~ virtual file system specification.

- inodes
- f-ops — file system operations. — mount

(#) read —— system call

(

fd → file object

└ f_ops — structure / list of fn. ptrs.

---

the VFS design/spec. also extends to the network! (to network IO).

sd = socket (—, —, —);

domain type protocol

bind

listen

newfd = accept

read (newfd, buf, len);

descriptor in the fd table

file object

f_ops

do n/w read