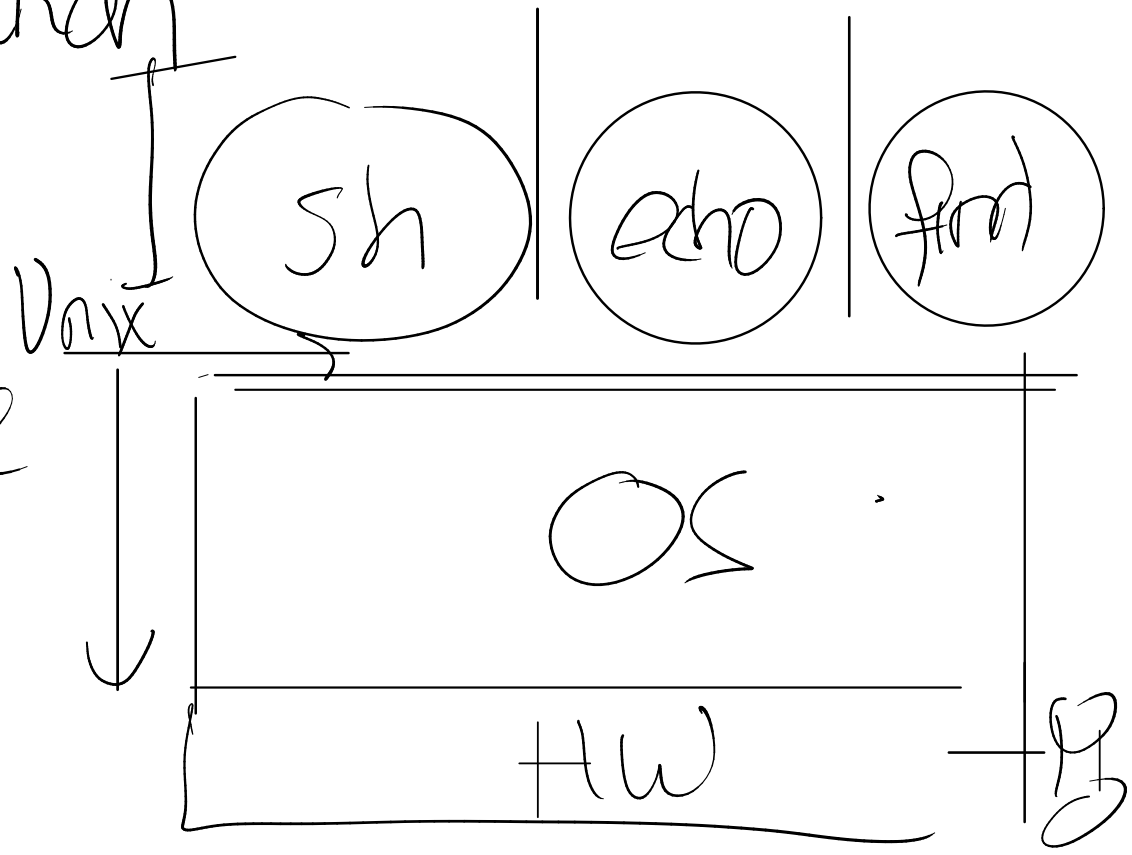
6.5081: OS organization

topic:

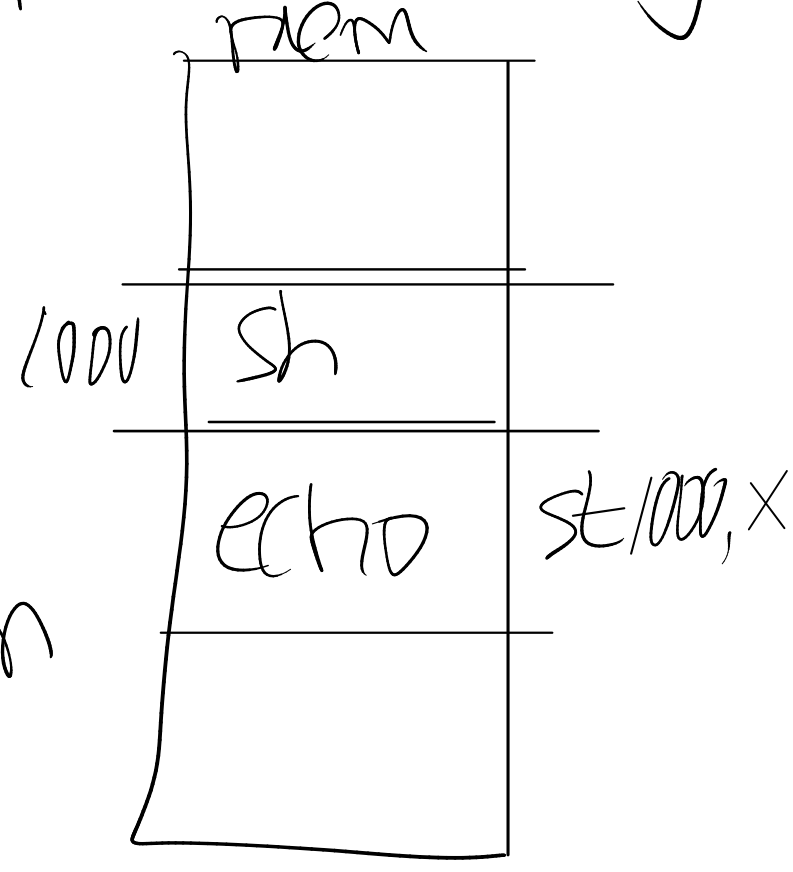
- isolation
- kernel/user mode
- system call
- xv6



Strawman design: no OS

No strong isolation

enforced multiplexing



Unix interface

abstract the HW resources

process: instead CPU

exec: instead of memory

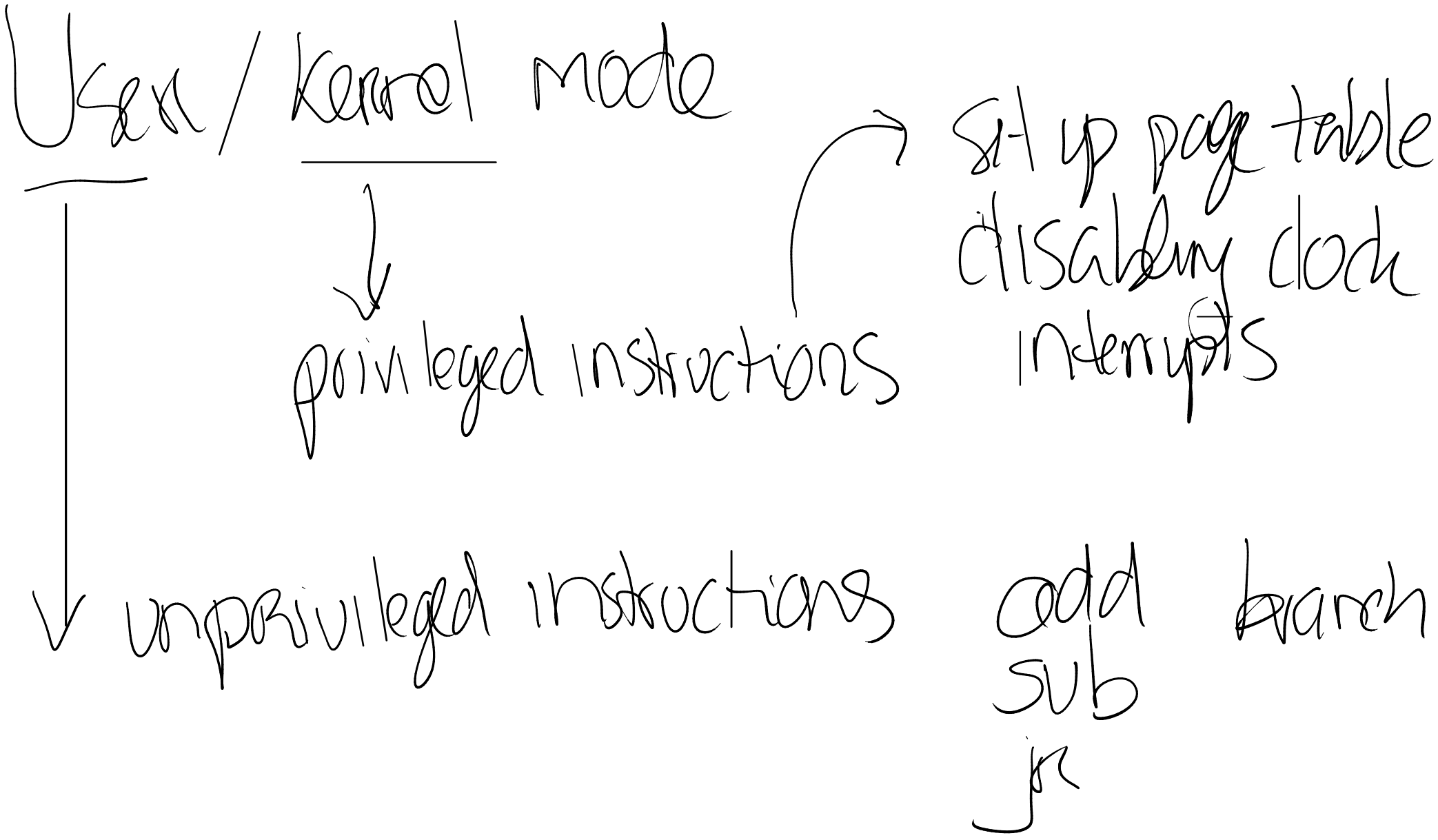
files: instead of disk block

OS should be defensive

- app cannot crash the OS
- app cannot break out of its isolation

⇒ Strong isolation between apps + OS

typical: HW support < ① user/kernel mode
② virtual memory

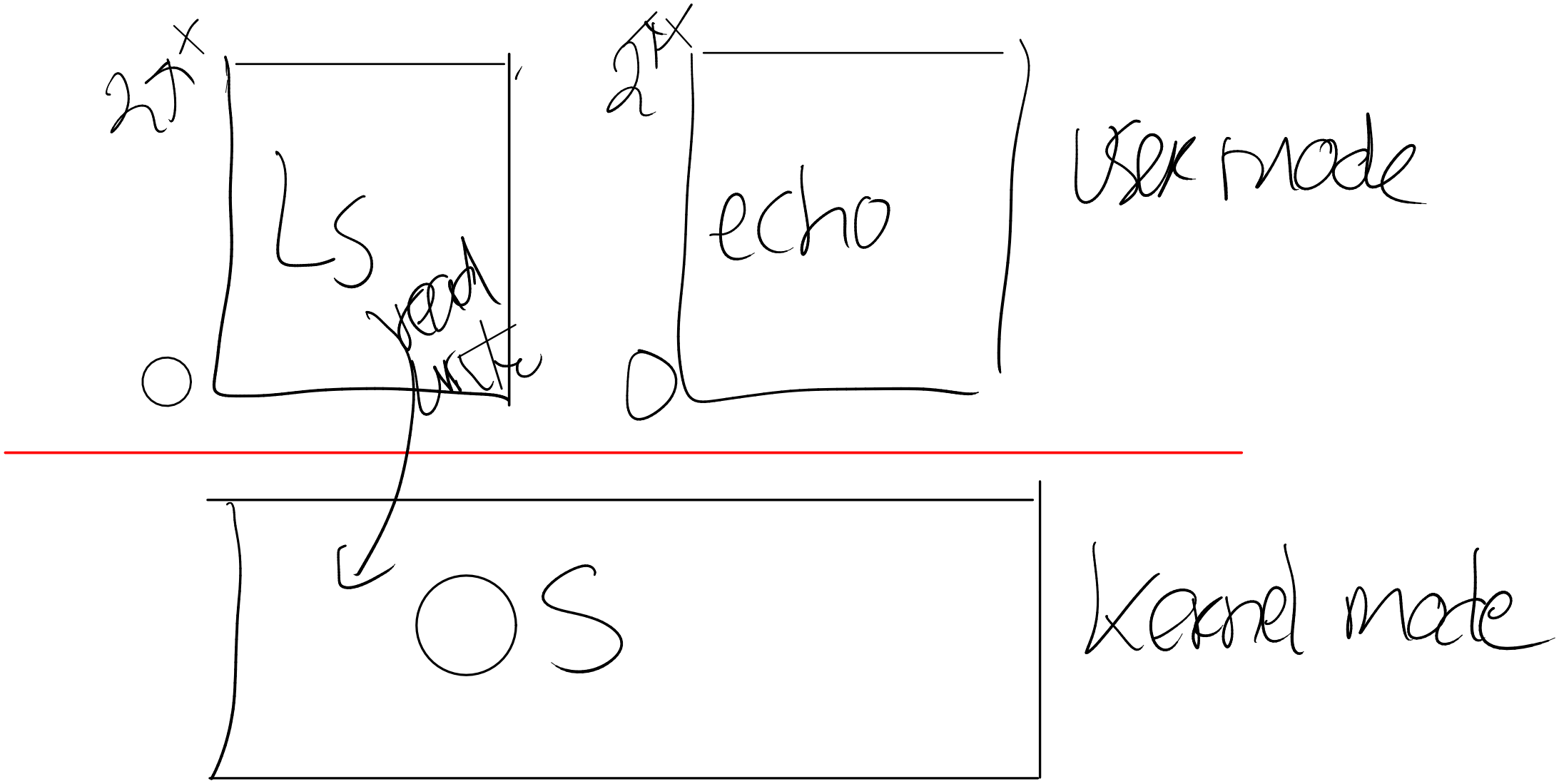

User / kernel mode
privileged instructions
set up page table
disabling clock interrupts
unprivileged instructions
add
sub
jr
branch

CPUs provide virtual memory

page table: virtual addr → physical.

process its own page table

memory isolation


2^x
LS
read
write
2^x
echo
User mode
OS
Kernel mode

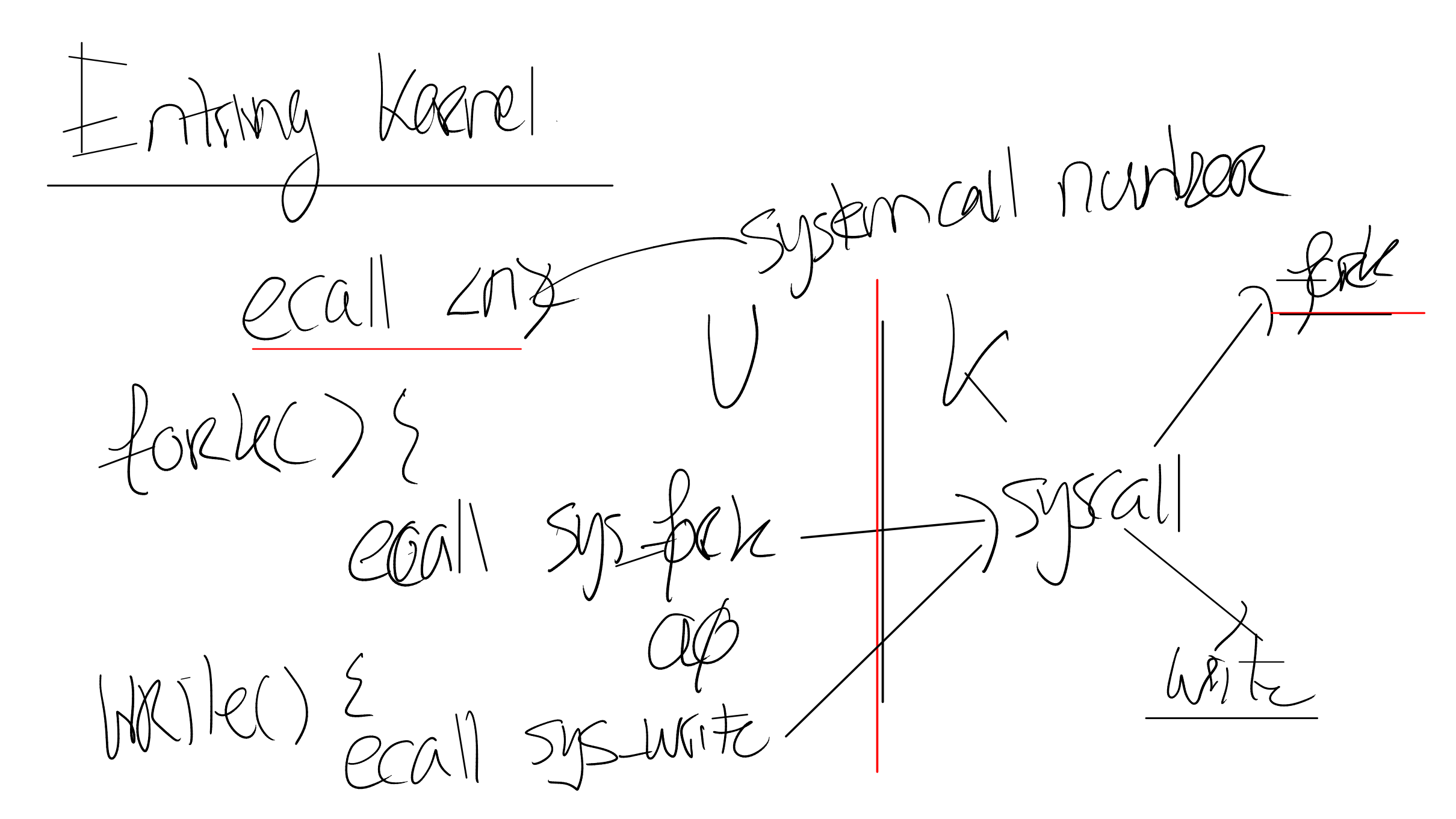
Entering Kernel

ecall <nr>

system call number

fork() {
ecall sys_fork
}

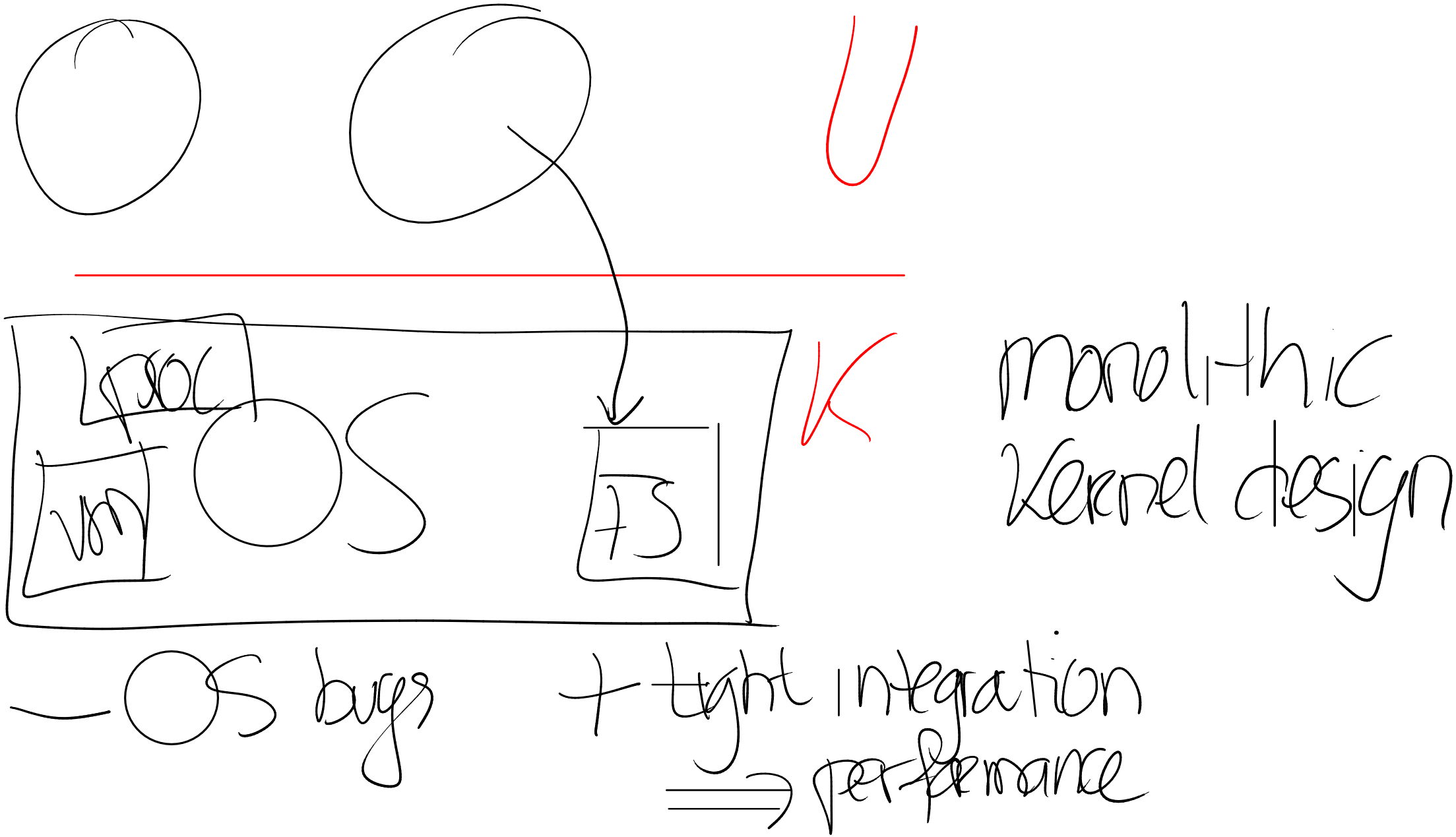
write() {
ecall sys_write
}



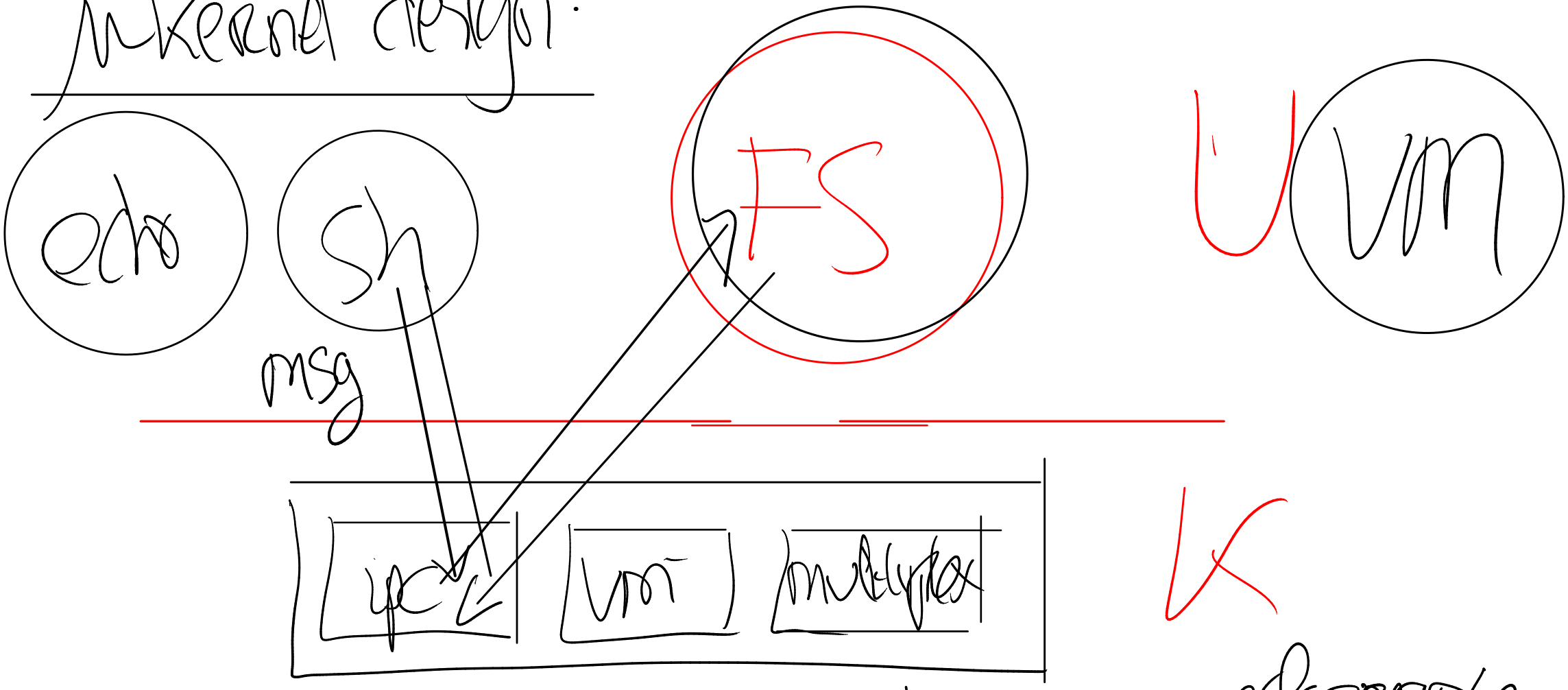
Kernel = trusted computing base (TCB)

- Kernel must have no bugs
- Kernel must treat processes as malicious

$\Rightarrow$ security

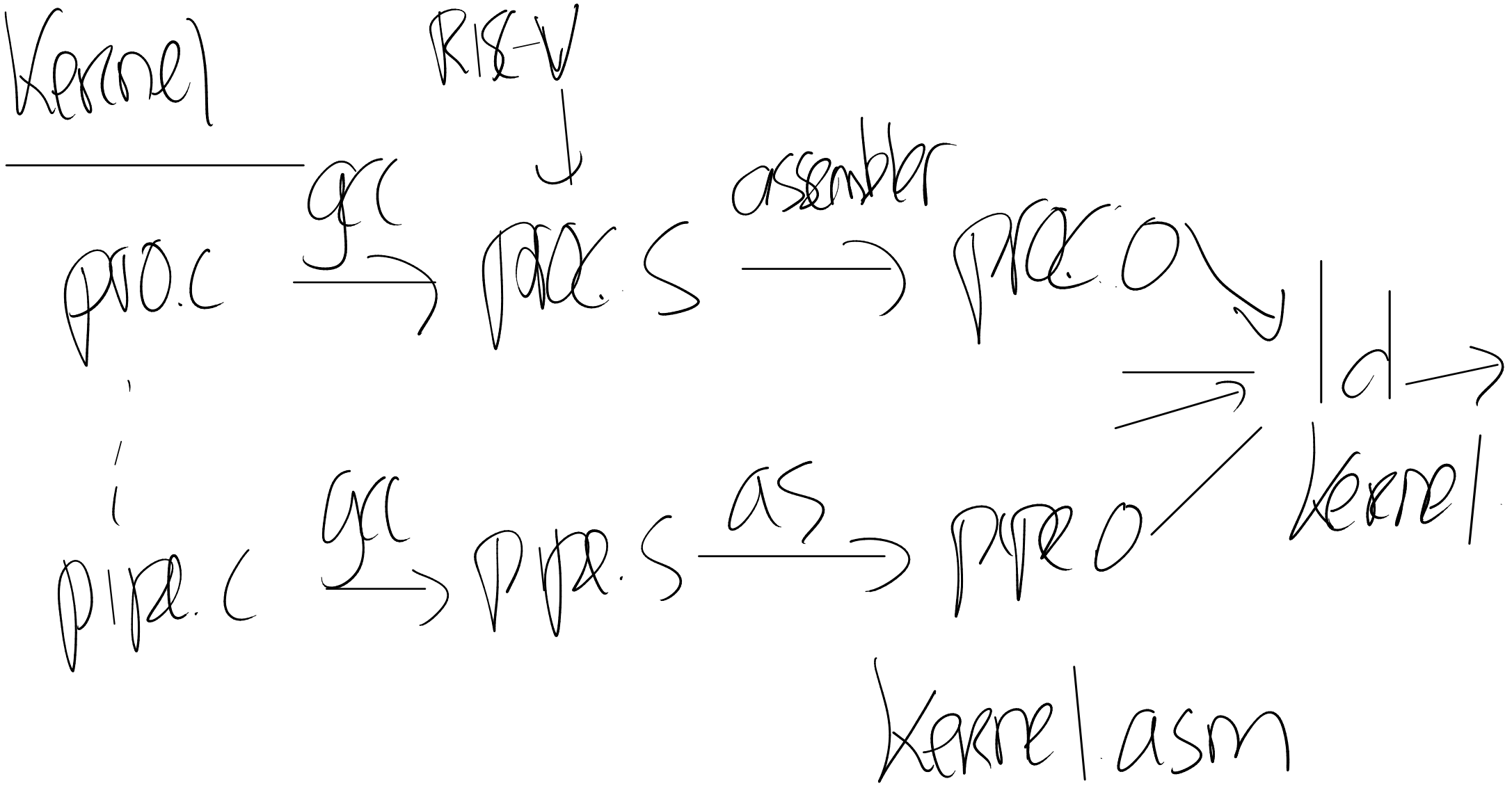

U
K
monolithic
kernel design
OS
VM
FS
- OS bugs
+ tight integration
⟹ performance

microkernel design:



+ kernel is small → fewer bugs

- performance


Kernel
RISC-V
proc
gcc
proc.s
assembler
proc.o
ld
kernel
proc
gcc
proc.s
as
proc.o
kernel.asm

SiFive
HiFive
Unleashed

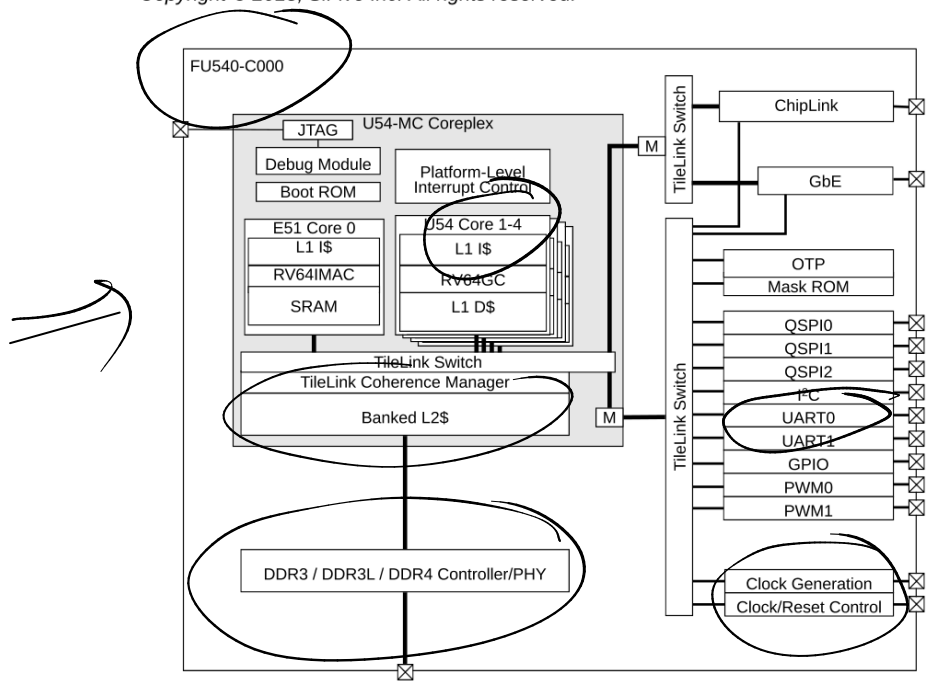


Figure 1: FU540-C000 top-level block diagram.

Qemulate emulate RISC-V.

```
for (;;) {
    read instruction
    decode instruction → add
                       → sub
    execute instruction  add a0, 7, 1
}
```