

the system call mechanism

- API vs ABI
- activation record.
- jump to an address in a pre-compiled & in-memory binary.

Q

how to setup the syscall mechanism?

- how to invoke? "
- how to pass arguments? & return values?
- how to switch modes? "
- save & restore context -?
- mechanism to identify the system call

⊕ OS builds this with ISA support.

(i) Lab 3

- Office hours
today - 2:30 pm
4:30 pm
- ~~boo-boo~~
will follow up today
- due date
- ~~24th Aug.~~
- 28th Aug

(ii) Lab Session

24th Aug. Mon.
9am to
10:30am

(iii) Quiz 1

31st Aug. Mon.
8am

(iv) Lab Quiz 1

2nd Sept. Wed.
5pm

(v) Midsem

???

(vi) Extra class

24th Aug
8am

~ all system calls are interrupts

explicit software ~ program instructions.

x86: int 0x80h risc-v ecall ISA supported ISA inst^r

int/ecall

- switch to privileged mode
- switch to kernel stack
- copy user context to kernel stack / only on x86
all regs.
- jmp to an address - interrupt handler
- ... } interrupt handling
- ~ restore context
- iret / sret return from interrupt handler to user space.
(x86) (riscv inst^r)

➔
 (#) where is the interrupt handler?

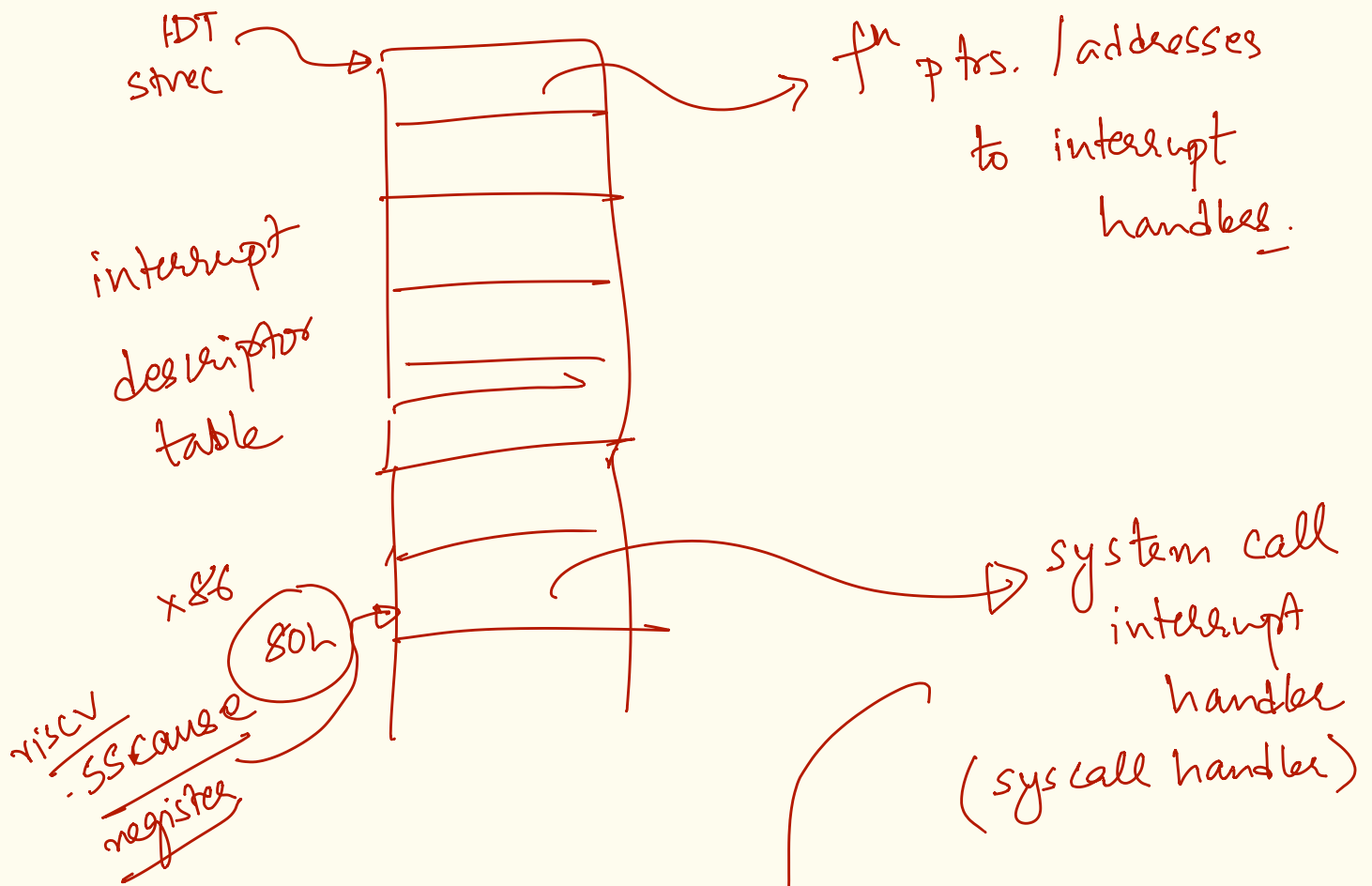
x86 riscv

IDTR stvec

(IDT register)

trap vector base register

➔ registers to store the interrupt desc^r. table address.



[Q] which system call?

x86
+ eax

riscv
+ a7

← register to store syscall number

syscall handler () {

num = value in a7; (or in eax) ← which syscall

fn ptr = syscall-table[num];

call fn ptr;

table of syscall fn pointers

→ prepare for return;

ret;

}

(*) how to pass arguments?

passed via cpu registers
Stored & accessed on the kernel stack
by the system call handler.

[Q] where is ecall in userspace?

embedded in userspace fn.
wrappers for syscalls.

[Q] which is the most frequent
interrupt?