21-1

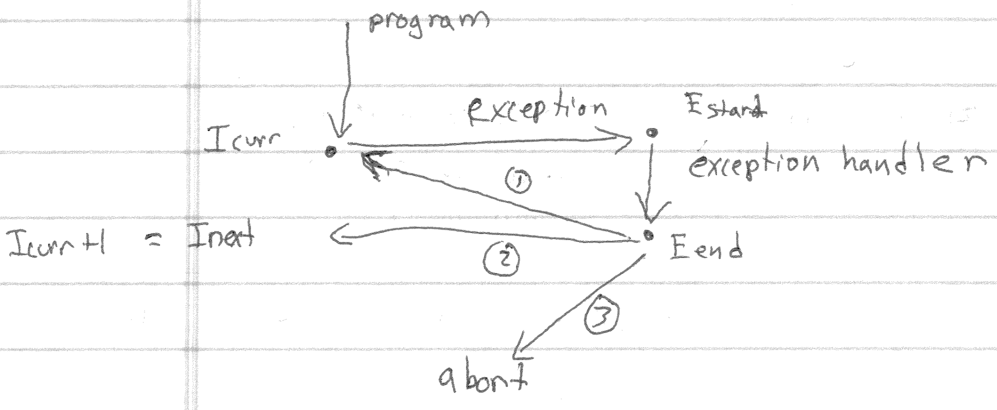
```
8000  movq  %rax, $13
8001  movq  %rbx, $22
8002  addq  %rax, %rbx
8003  jnz   8000
8004  ret
8005  divq  %rax, $0
```

$$\begin{tikzcd}
\text{PC} \arrow[r, Rightarrow] & 8000 \arrow[r] & 8001 \arrow[r] & 8002 \arrow[r] & 8003 \arrow[dl] \arrow[dr] & \\
 & & & 8000 & & 8004
\end{tikzcd}$$

"program counter"

PC data is the control-flow only influenced by the program text

Exceptions — when the PC takes on values not in the program text



① Estart is a known place and the exception is data

jump to addr 9000   %rax = what exception it was

② Estart is a customizable place and the exn is data

jump to * %exn

protected by CPU exection "modes"

two modes: safe and unsafe

safe	unsafe
can't change %exn	can change %exn
stay safe	become safe
	start
	exn handler

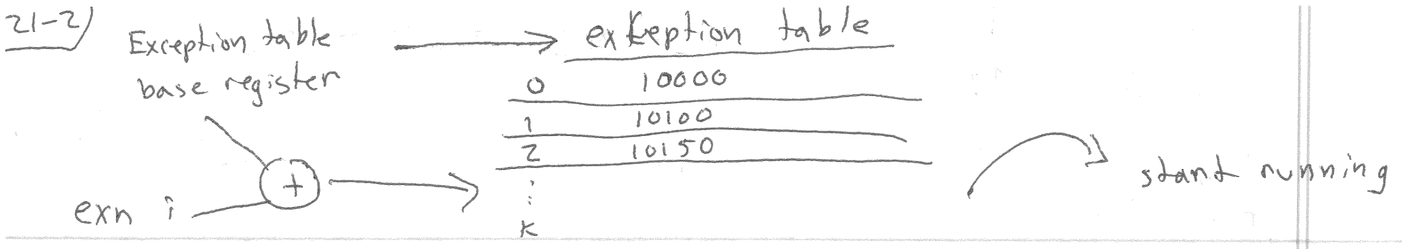
③ Estart = customizable place (%exn) + exn no.

%exn = 9000   exn = 17   exn = 18

jump 9017   jump 9018

9017: jump 10000   9018: jump 10100

21-2)

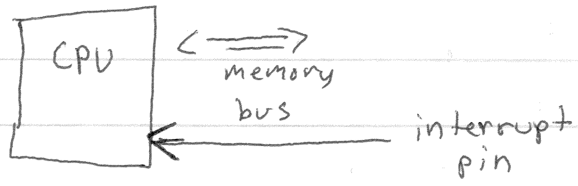


CPU on exn i : push PC ; switches to unsafe mode ;
jump ETBR + i

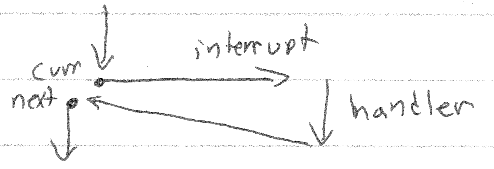
Inside exn handler : save the programs registers — skip ③
does work → stop on ③
①+② restoreregisters
go safe
④ return ① pop PC to ~~[illegible]~~ reg
figure out prev inst
jump to it

Interrupts —	Device Signals —	return to Inext —	Async
Trap —	Intentional exception —	Inext —	Sync
Fault —	Error that could be fixed —	may Icurr —	Sync
Abort —	Irrecoverable error —	aborts —	Sync

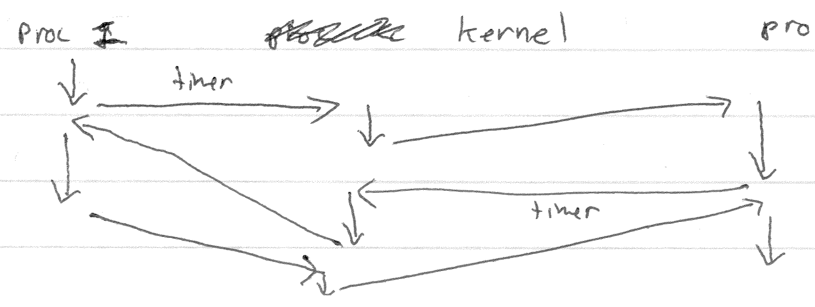
Interrupt



```
while (1) {
    inst = mem[pc]
    decode inst
    pc++
    do the inst
    if (interrupt) {
        do exn }
}
```



examples are disks, timers, network, etc



21-3/

Traps — intentional exn thrown by program

a user program traps to run the OS/kernel

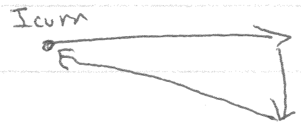
a "system call"

communicate (ie save in a known place) the OS request

strace pid

returns to Inext

Faults



return to Icurr

page fault (movq %rax, 10000) and 10000 isn't in memory

exn handler puts 10000 in memory (±4k)

then run instruction again

---

aborts — div by ∅, tried to read memory and corrupted

---

exn 0	—	divide by zero	abort
13	—	general protection fault (seg fault)	fault
14	-	page fault	fault
18	—	"check machine" fault	abort
32–127	—	used by hardware/OS	
128 (0x80)	—	system call	
129–255	—	used by OS	

/usr/include/sys/syscall.h

→ C : exit(0)

```
asm:  movq  $1, %rax    ← rax = syscall number
      movq  $0, %rbx    ← rbx = exit's argument
      int   $x80        ← cause interrupt
                          (syscall)
```

21-4/

```
write(1, "hello world\n", 13);
      ↑        ↑           ↑
     port     data       length of data
```

```
.section .data
string:   .ascii "hello world\n"
num0:     .int 13
.section .text
main:   mov $4, rax          // 4 is "write"
        mov $1, rbx
        mov $string, cx
        mov num0, rdx
        int $x80
```