



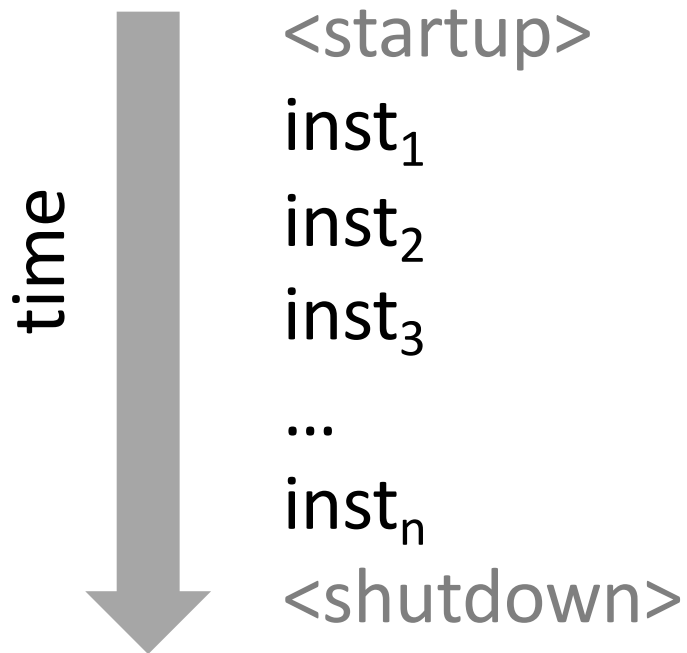
Exceptional Control Flow

Hardware support for reacting to the rest
of the world.

Control Flow

Processor: read instruction, execute it, go to next instruction, repeat

Physical control flow



Explicit changes:

Exceptional changes:

Exceptions

Synchronous: caused by instruction

Traps: system calls

Faults: unintentional, maybe recoverable

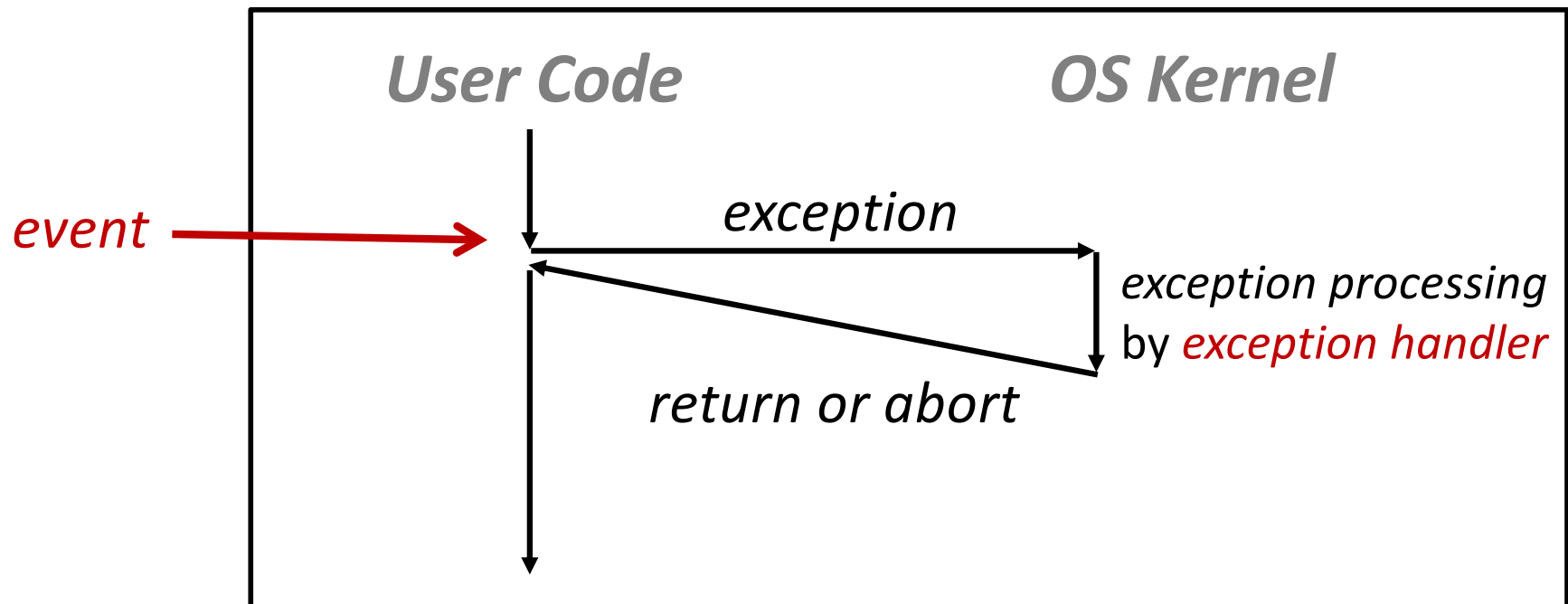
Aborts: unintentional, unrecoverable

Asynchronous (Interrupts): caused by external events
incoming I/O activity, reset button, timers, signals

Exceptions: hardware support for OS

transfer control to OS in response to *event*

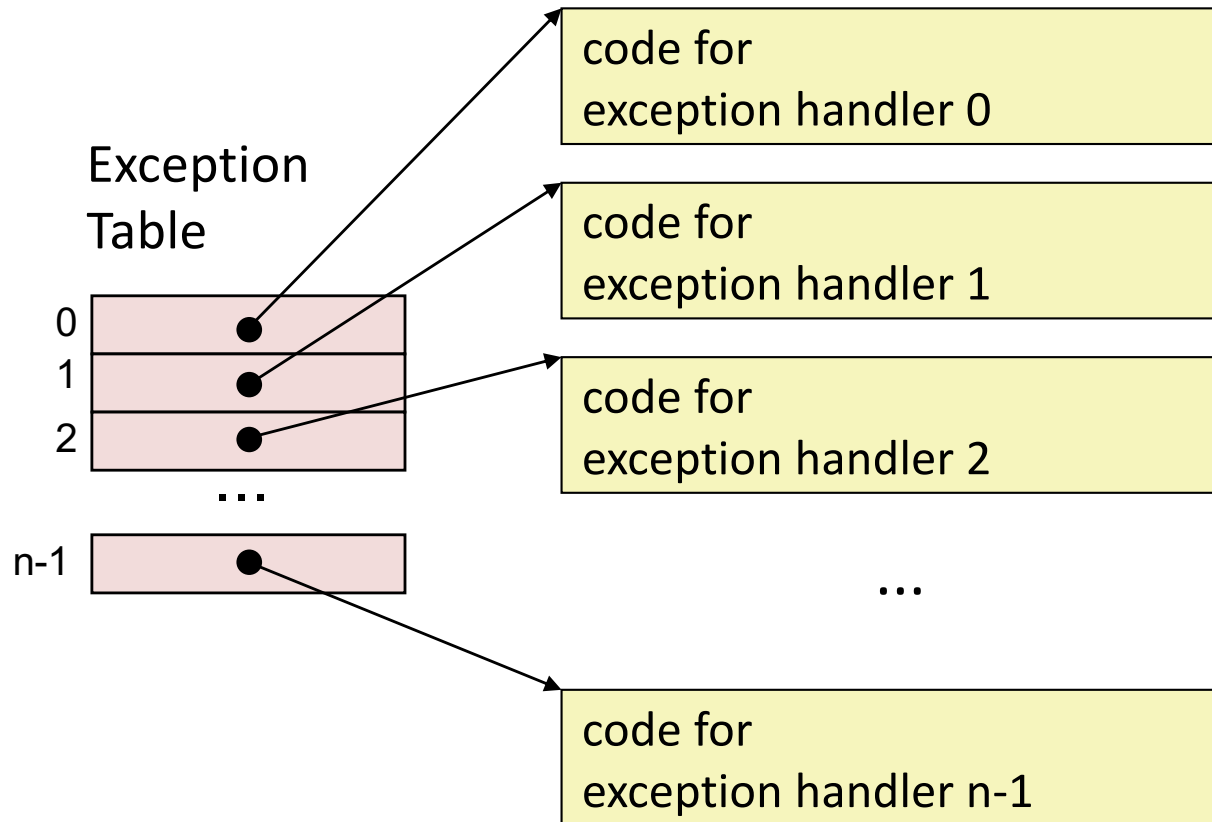
What code should the OS run?



Interrupt Vector

in memory

special register holds base address



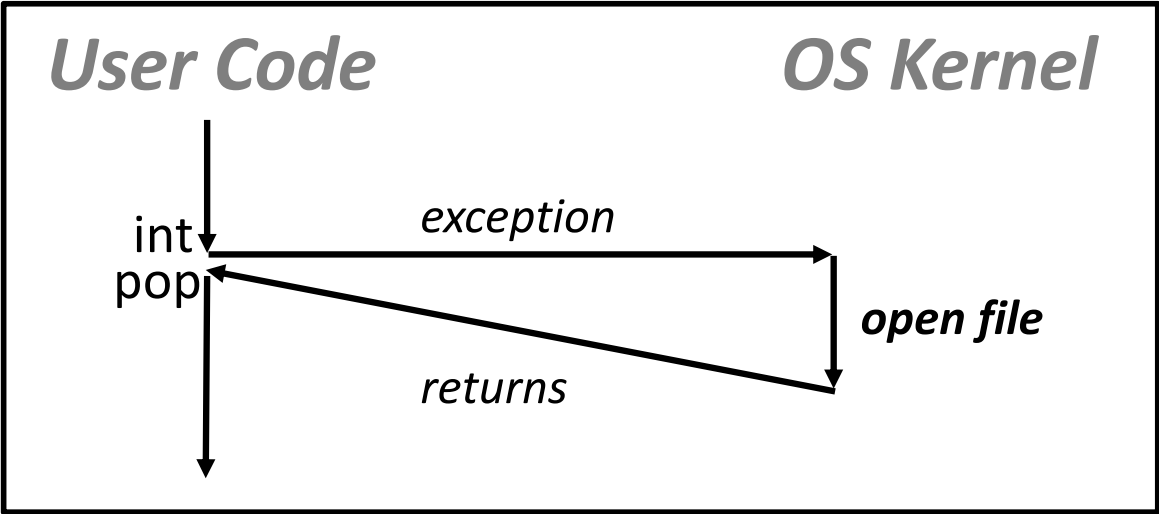
a jump table for exceptions...

Open a file (trap/system call)

User process calls: `open(filename, options)`

`open` executes system call instruction `int`

```
0804d070 <__libc_open>:  
. . .  
804d082:  cd 80          int    $0x80  
804d084:  5b            pop    %ebx  
. . .
```

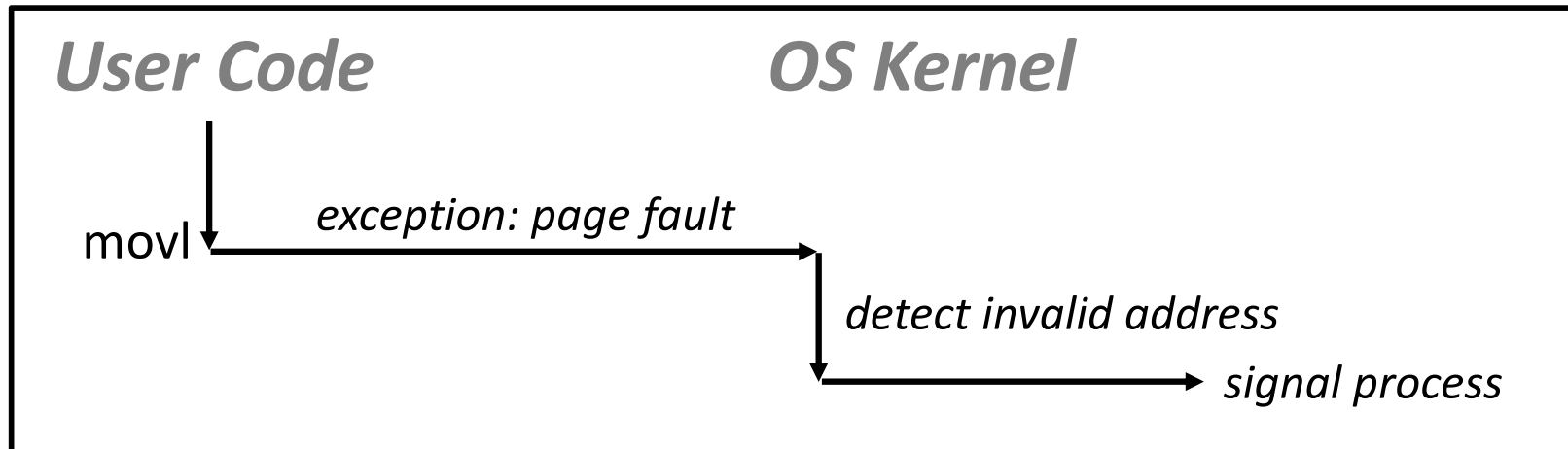


Segmentation Fault

```
int a[1000];  
void bad () {  
    a[5000] = 13;  
}
```

Write to invalid memory location.

```
80483b7: c7 05 60 e3 04 08 0d movl $0xd,0x804e360
```



aborts process with SIGSEGV signal

Page Fault

Write to valid memory location

... but contents currently on disk instead

(more later: virtual memory)

```
int a[1000];
main () {
    a[500] = 13;
}
```

```
80483b7: c7 05 10 9d 04 08 0d movl $0xd,0x8049d10
```

