CS 240
Foundations of Computer Systems
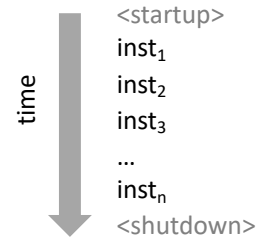
WELLESLEY

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

time

<startup>
inst$_1$
inst$_2$
inst$_3$
...
inst$_n$
<shutdown>

Explicit changes:

Exceptional changes:

Exceptions

Synchronous: caused by instruction

Traps: system calls
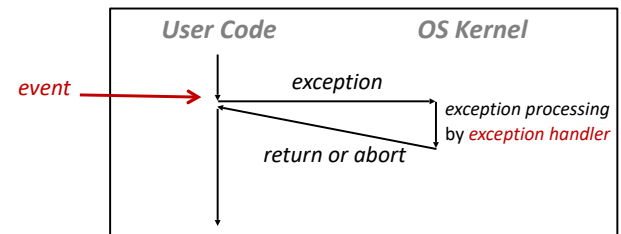
Faults: unintentional, maybe recoverable

Aborts: unintentional, unrecoverable

Asynchronous (Interrupts): caused by external events
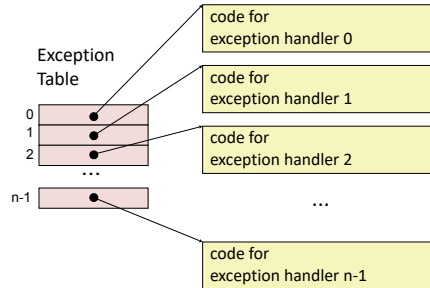

Exceptions: hardware support for OS

transfer control to OS in response to *event*

What code should the OS run?

User Code   *OS Kernel*

event

exception

exception processing
by *exception handler*

return or abort

Interrupt Vector

in memory
special register holds base address

Exception
Table

```
0
1
2
...
n-1
```

code for
exception handler 0

code for
exception handler 1

code for
exception handler 2

...

code for
exception handler n-1
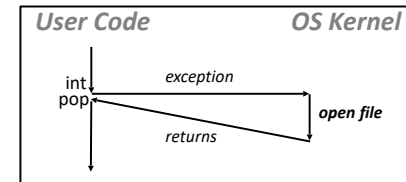
a jump table for exceptions...

Open a file (trap/system call)

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

User Code   *OS Kernel*

int   *exception*
pop   *open file*
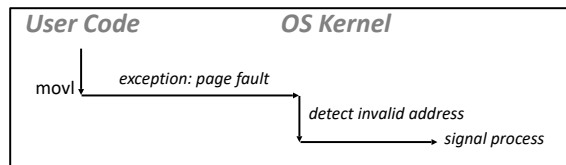   *returns*

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
```

User Code   *OS Kernel*

movl   *exception: page fault*

                    detect invalid address
                              signal process

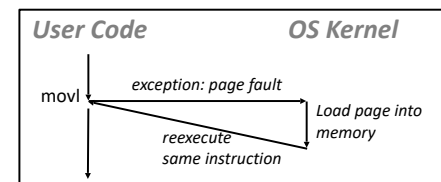aborts process with SIGSEGV signal

Page Fault

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

Write to valid memory location

... but contents currently on disk instead

(more later: virtual memory)

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

User Code   *OS Kernel*

movl   *exception: page fault*

                              Load page into
         reexecute   *memory*
         same instruction