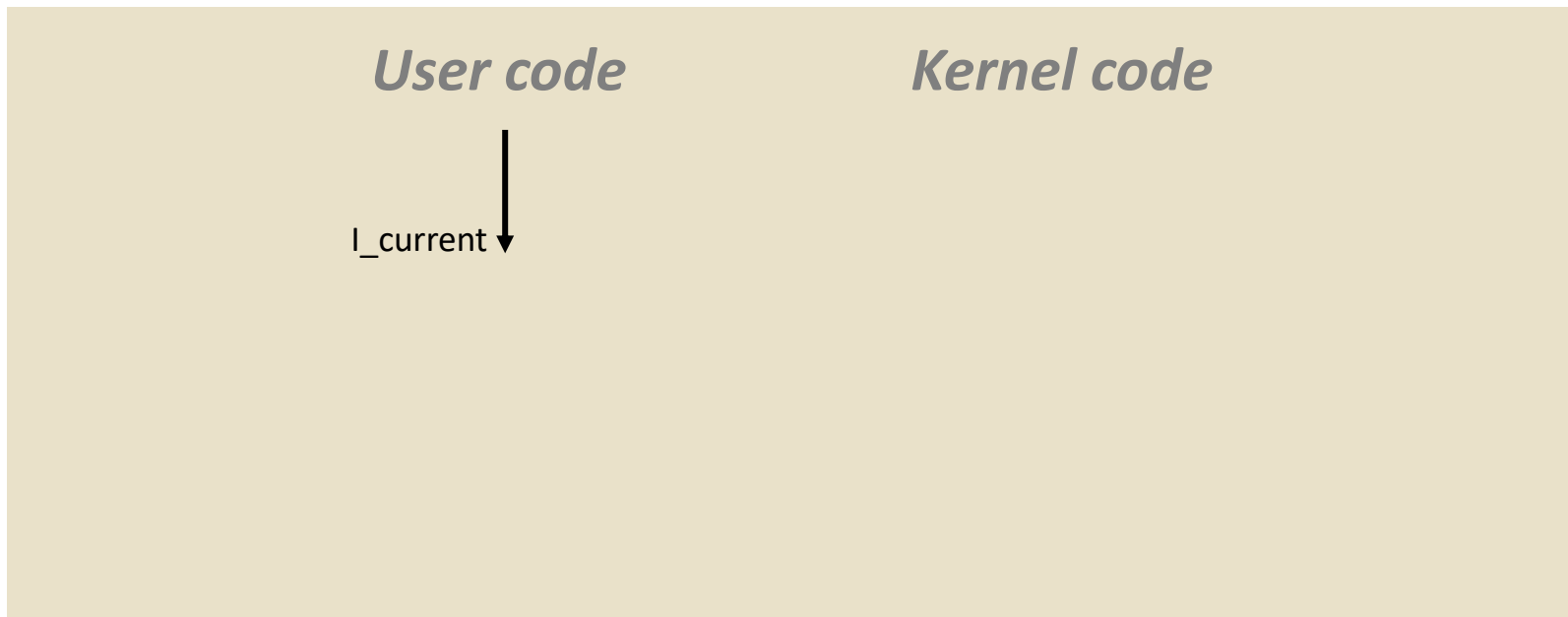


Exceptional Control Flow:

Exceptions

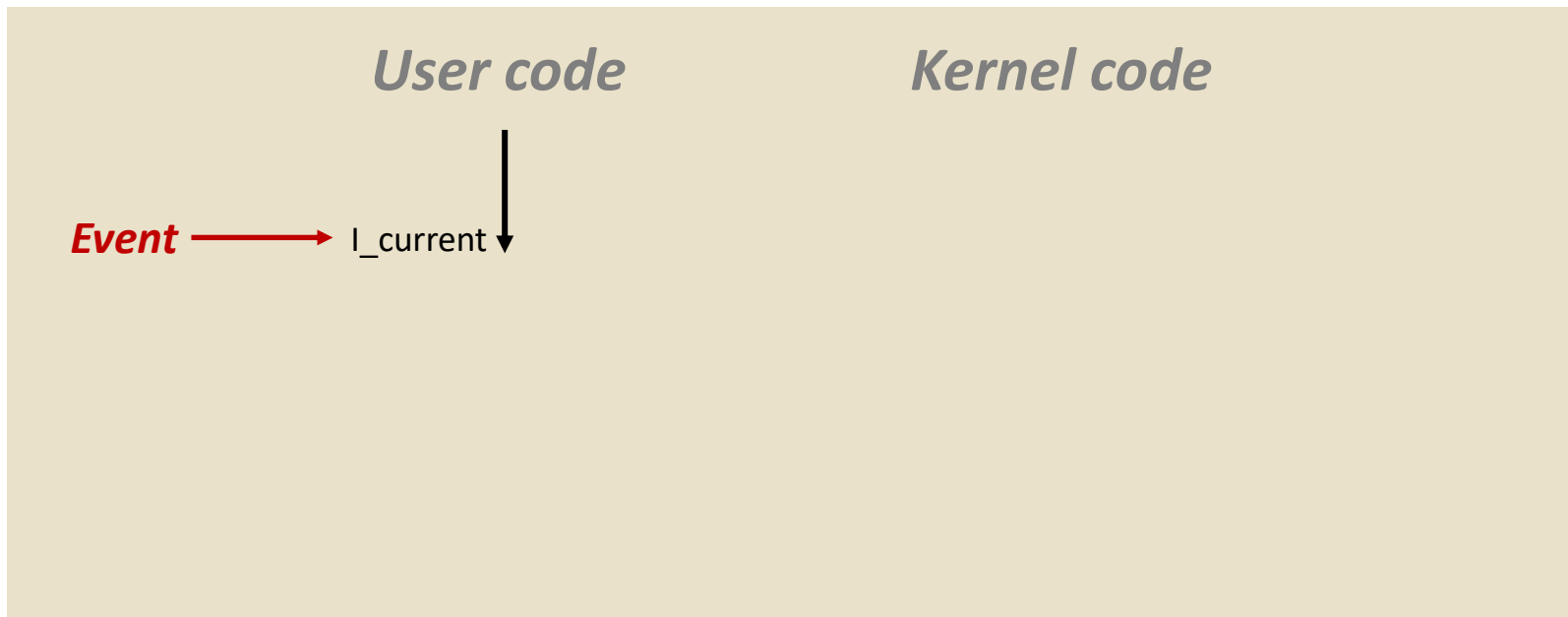
Exceptions

- An **exception** is a transfer of control to the OS *kernel* in response to some *event* (i.e., change in processor state)
 - Kernel is the memory-resident part of the OS
 - Examples of events: Divide by 0, arithmetic overflow, page fault, I/O request completes, typing Ctrl-C



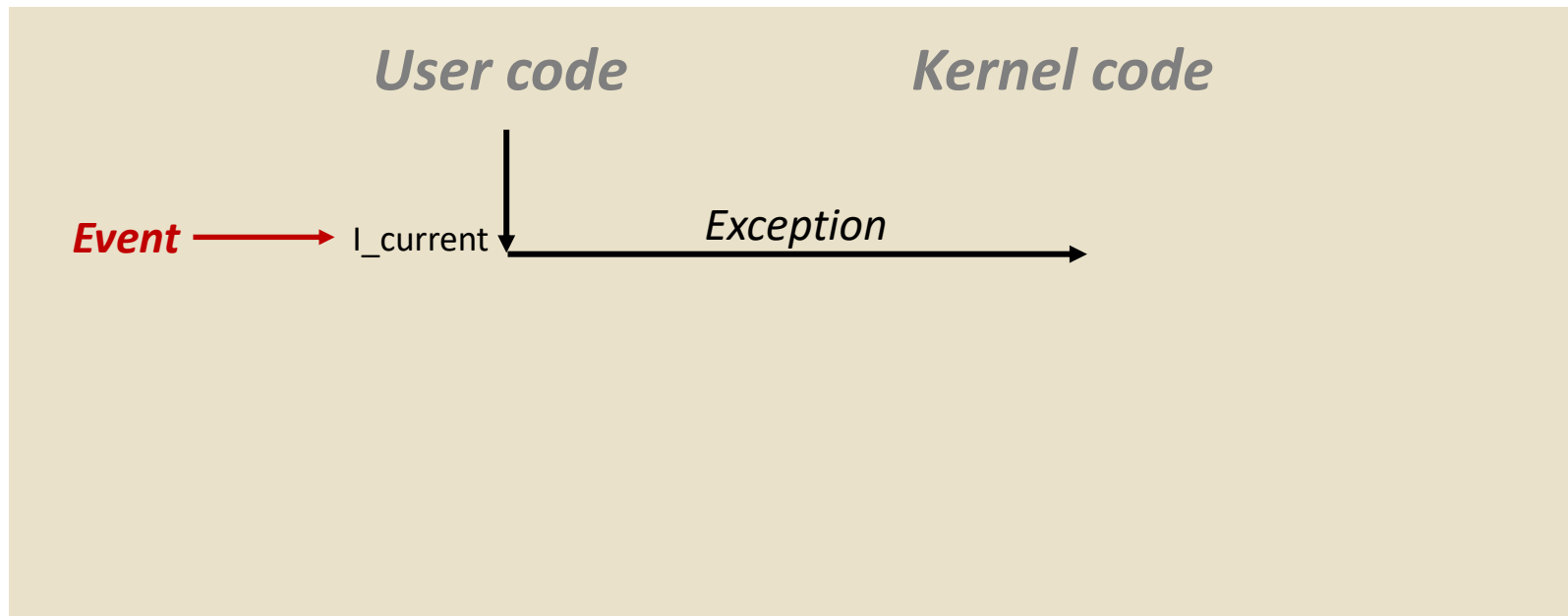
Exceptions

- An **exception** is a transfer of control to the OS *kernel* in response to some *event* (i.e., change in processor state)
 - Kernel is the memory-resident part of the OS
 - Examples of events: Divide by 0, arithmetic overflow, page fault, I/O request completes, typing Ctrl-C



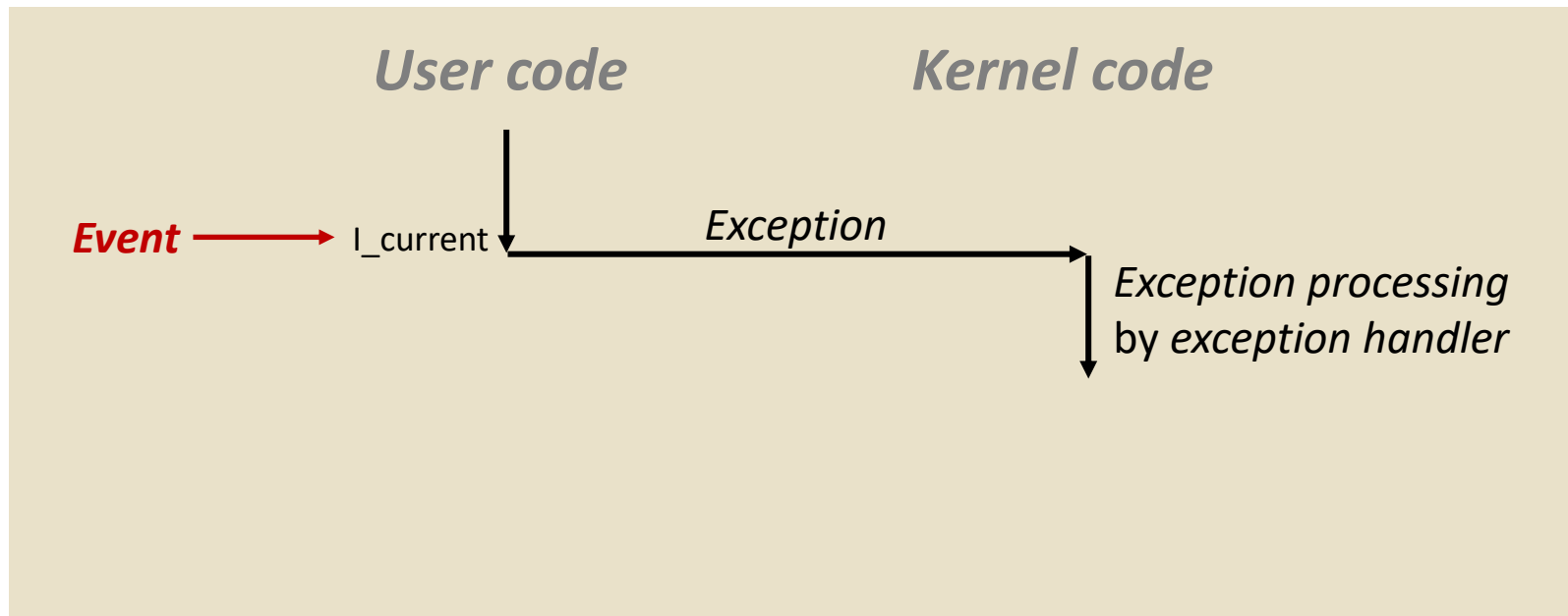
Exceptions

- An **exception** is a transfer of control to the OS *kernel* in response to some *event* (i.e., change in processor state)
 - Kernel is the memory-resident part of the OS
 - Examples of events: Divide by 0, arithmetic overflow, page fault, I/O request completes, typing Ctrl-C



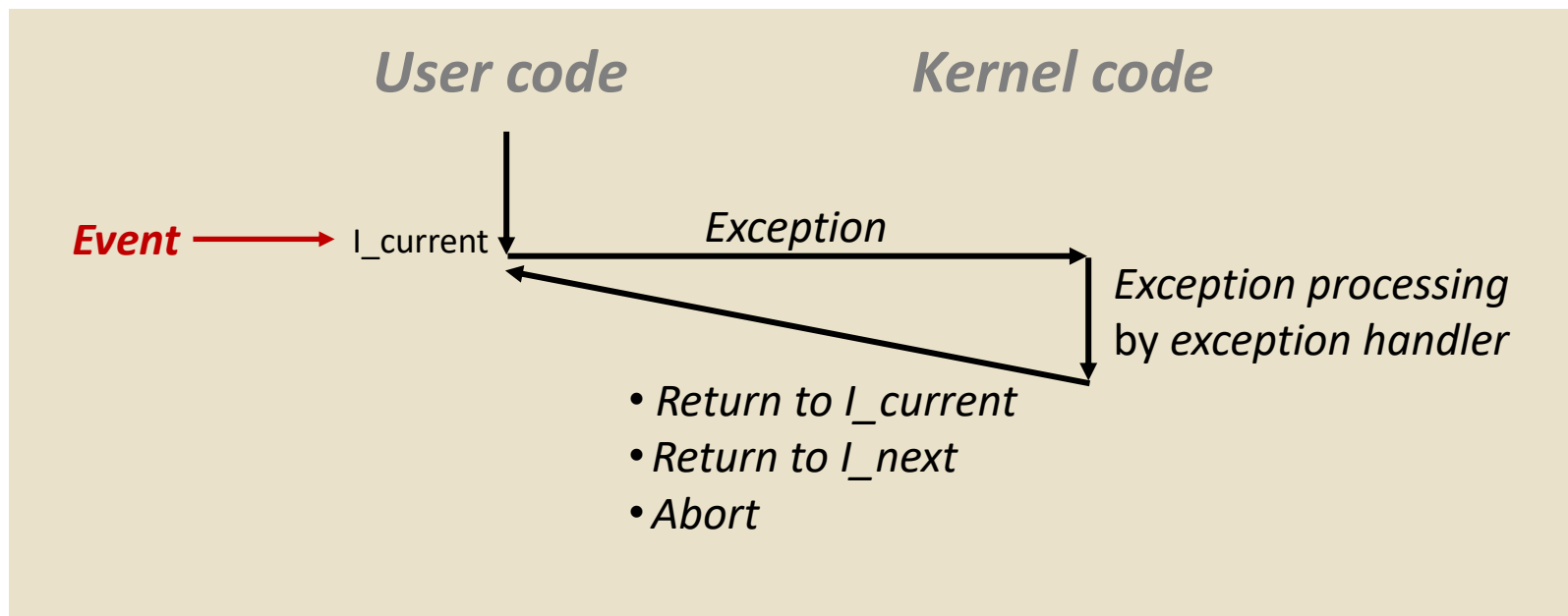
Exceptions

- An **exception** is a transfer of control to the OS *kernel* in response to some *event* (i.e., change in processor state)
 - Kernel is the memory-resident part of the OS
 - Examples of events: Divide by 0, arithmetic overflow, page fault, I/O request completes, typing Ctrl-C



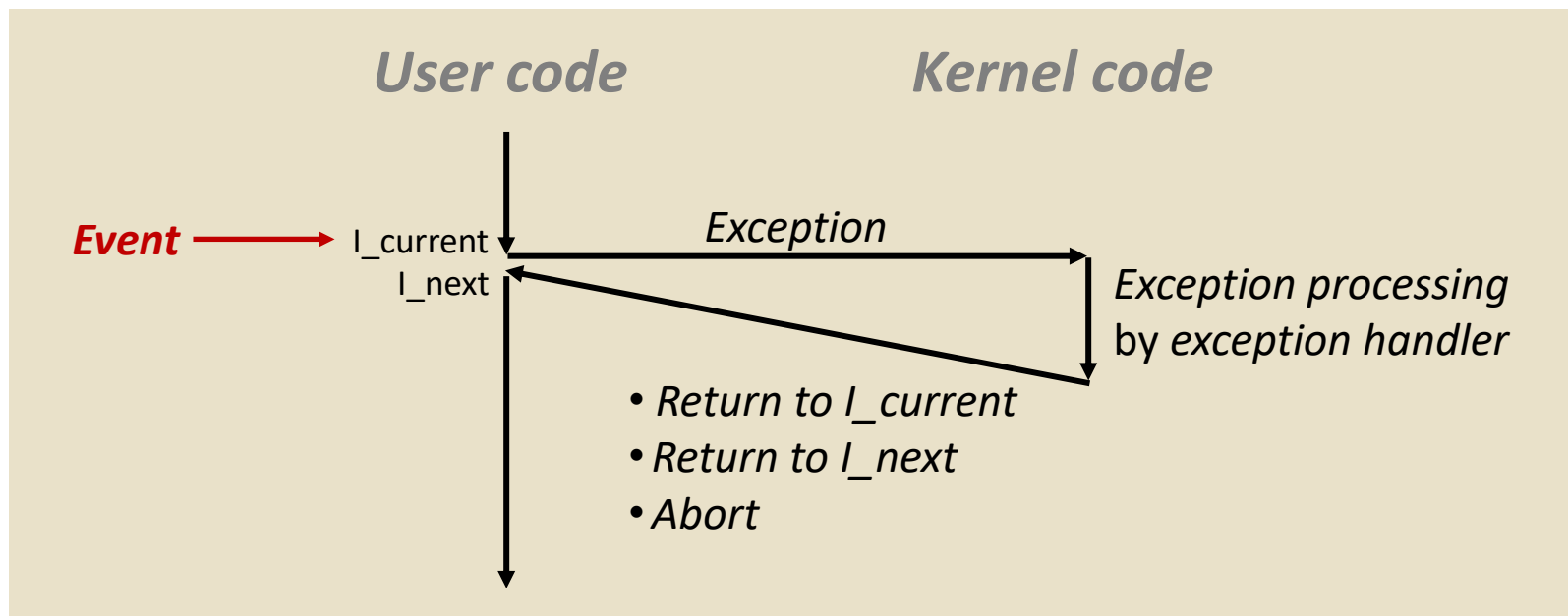
Exceptions

- An **exception** is a transfer of control to the OS *kernel* in response to some *event* (i.e., change in processor state)
 - Kernel is the memory-resident part of the OS
 - Examples of events: Divide by 0, arithmetic overflow, page fault, I/O request completes, typing Ctrl-C

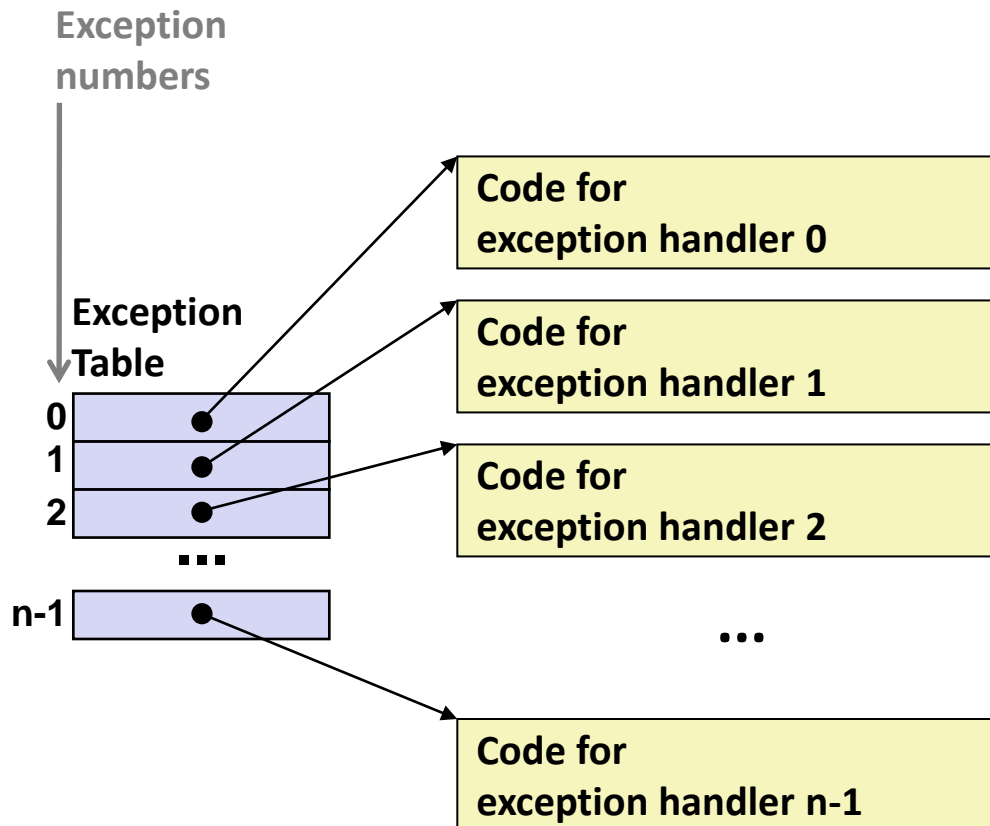


Exceptions

- An **exception** is a transfer of control to the OS *kernel* in response to some *event* (i.e., change in processor state)
 - Kernel is the memory-resident part of the OS
 - Examples of events: Divide by 0, arithmetic overflow, page fault, I/O request completes, typing Ctrl-C



Exception Tables



- Each type of event has a unique exception number k
- k = index into exception table (a.k.a. interrupt vector)
- Handler k is called each time exception k occurs

Asynchronous Exceptions (Interrupts)

- **Caused by events external to the processor**
 - Indicated by setting the processor's *interrupt pin*
 - Handler returns to “next” instruction

Asynchronous Exceptions (Interrupts)

- **Caused by events external to the processor**
 - Indicated by setting the processor's *interrupt pin*
 - Handler returns to “next” instruction

- **Examples:**
 - Timer interrupt
 - Every few ms, an external timer chip triggers an interrupt
 - Used by the kernel to take back control from user programs

Asynchronous Exceptions (Interrupts)

■ Caused by events external to the processor

- Indicated by setting the processor's *interrupt pin*
- Handler returns to “next” instruction

■ Examples:

- Timer interrupt
 - Every few ms, an external timer chip triggers an interrupt
 - Used by the kernel to take back control from user programs
- I/O interrupt from external device
 - Hitting Ctrl-C at the keyboard
 - Arrival of a packet from a network
 - Arrival of data from a disk

Synchronous Exceptions

- **Caused by events that occur as a result of executing an instruction:**

Synchronous Exceptions

- Caused by events that occur as a result of executing an instruction:
 - *Traps*
 - Intentional
 - Examples: *system calls*, breakpoint traps, special instructions
 - Returns control to “next” instruction

Synchronous Exceptions

- Caused by events that occur as a result of executing an instruction:
 - **Traps**
 - Intentional
 - Examples: **system calls**, breakpoint traps, special instructions
 - Returns control to “next” instruction
 - **Faults**
 - Unintentional but possibly recoverable
 - Examples: page faults (recoverable), protection faults (unrecoverable), floating point exceptions
 - Either re-executes faulting (“current”) instruction or aborts

Synchronous Exceptions

- Caused by events that occur as a result of executing an instruction:
 - **Traps**
 - Intentional
 - Examples: **system calls**, breakpoint traps, special instructions
 - Returns control to “next” instruction
 - **Faults**
 - Unintentional but possibly recoverable
 - Examples: page faults (recoverable), protection faults (unrecoverable), floating point exceptions
 - Either re-executes faulting (“current”) instruction or aborts
 - **Aborts**
 - Unintentional and unrecoverable
 - Examples: illegal instruction, parity error, machine check
 - Aborts current program

System Calls

- Each x86-64 system call has a unique ID number
- Examples:

<i>Number</i>	<i>Name</i>	<i>Description</i>
0	<code>read</code>	Read file
1	<code>write</code>	Write file
2	<code>open</code>	Open file
3	<code>close</code>	Close file
4	<code>stat</code>	Get info about file
57	<code>fork</code>	Create process
59	<code>execve</code>	Execute a program
60	<code>_exit</code>	Terminate process
62	<code>kill</code>	Send signal to process

System Call Example: Opening File

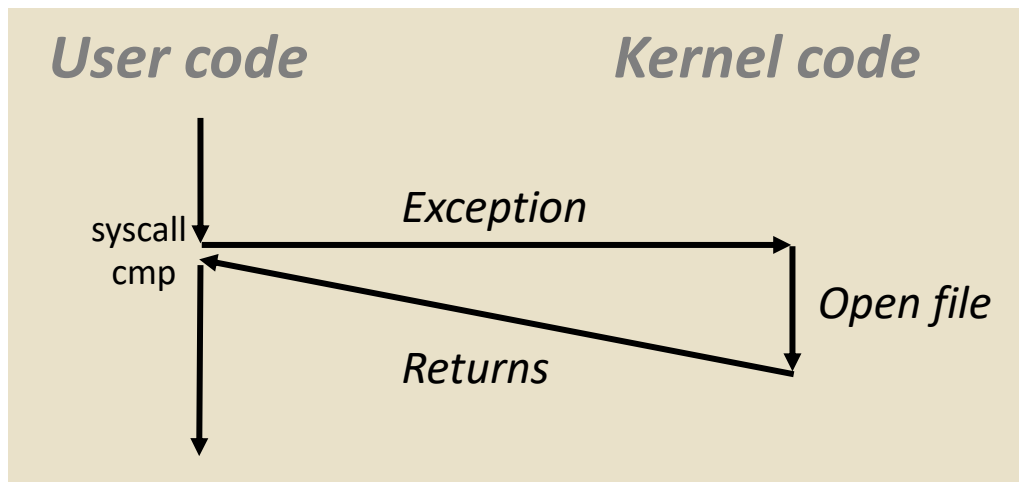
- User calls: `open(filename, options)`
- Calls `__open` function, which invokes system call instruction `syscall`

```
0000000000e5d70 <__open>:
```

```

...
e5d79:  b8 02 00 00 00      mov  $0x2,%eax  # open is syscall #2
e5d7e:  0f 05               syscall         # Return value in %rax
e5d80:  48 3d 01 f0 ff ff   cmp  $0xffffffffffffffff001,%rax
...
e5dfa:  c3                  retq

```



- `%rax` contains syscall number
- Other arguments in `%rdi`, `%rsi`, `%rdx`, `%r10`, `%r8`, `%r9`
- Return value in `%rax`
- Negative value is an error corresponding to negative `errno`

Fault Example: Page Fault

- User writes to memory location
- That portion (page) of user's memory is currently on disk

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

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

Fault Example: Page Fault

- User writes to memory location
- That portion (page) of user's memory is currently on disk

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

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

User code

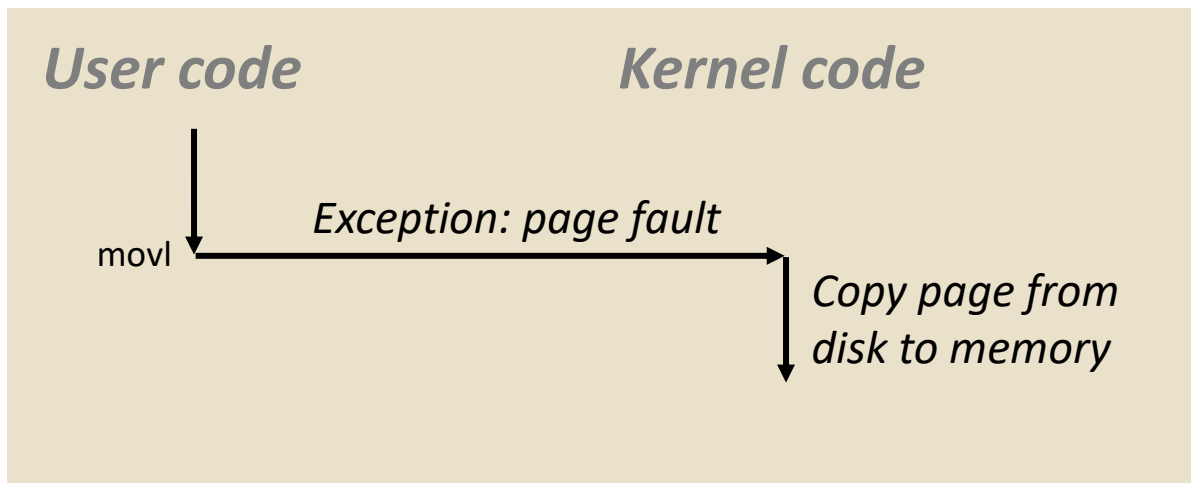
↓
movl

Fault Example: Page Fault

- User writes to memory location
- That portion (page) of user's memory is currently on disk

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

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

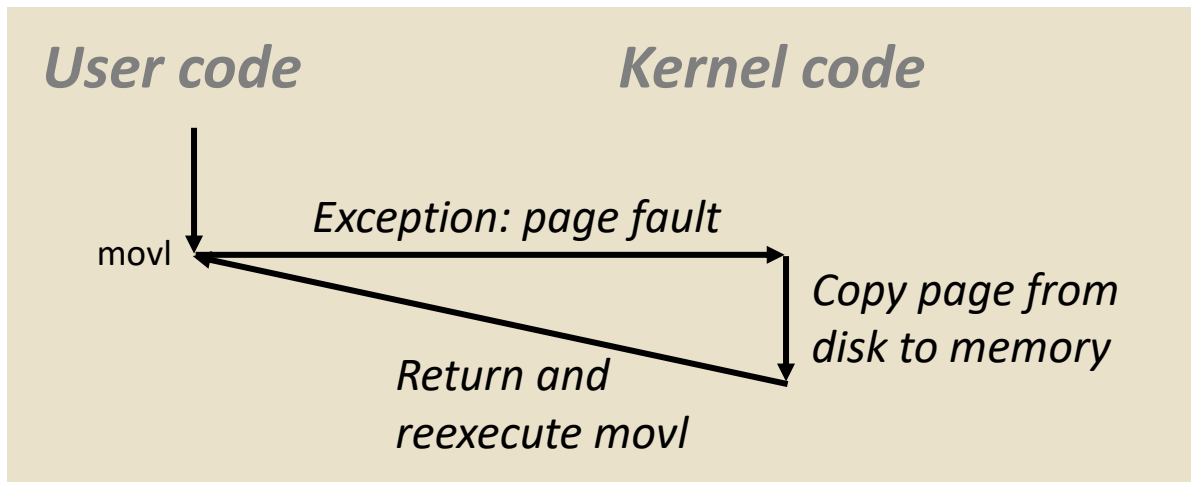


Fault Example: Page Fault

- User writes to memory location
- That portion (page) of user's memory is currently on disk

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

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

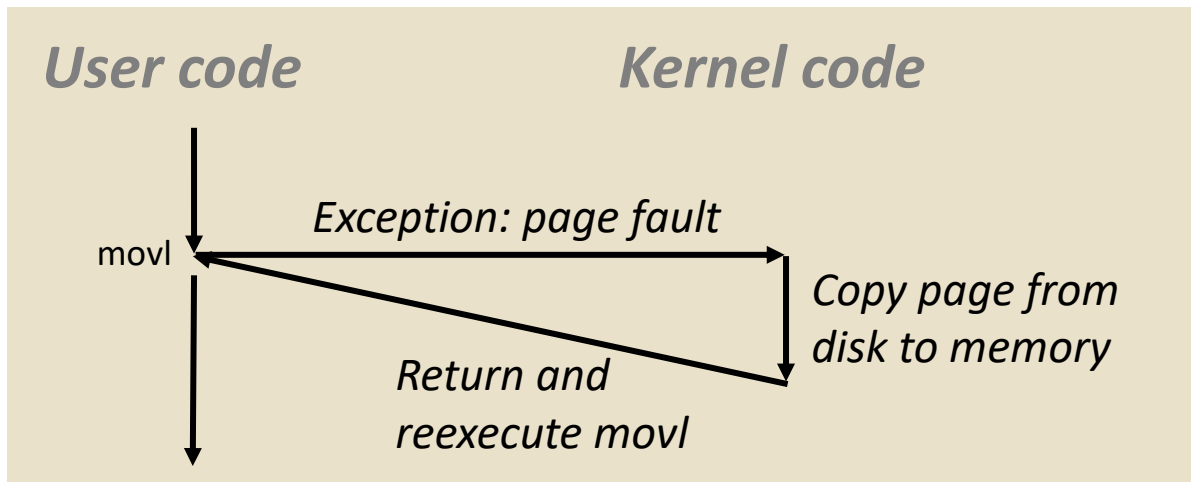


Fault Example: Page Fault

- User writes to memory location
- That portion (page) of user's memory is currently on disk

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

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



Fault Example: Invalid Memory Reference

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

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

Fault Example: Invalid Memory Reference

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

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

User code

↓
movl

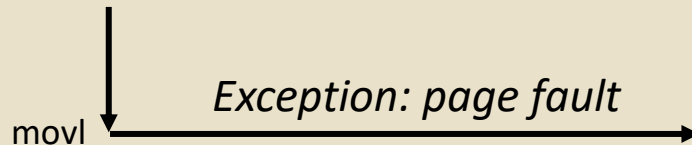
Fault Example: Invalid Memory Reference

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

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

User code

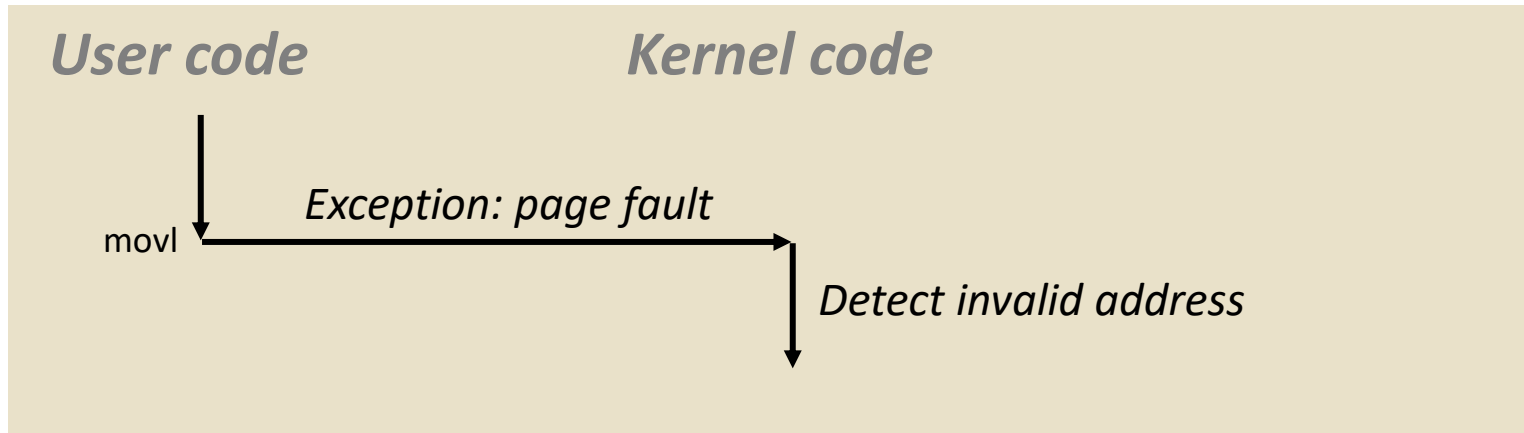
Kernel code



Fault Example: Invalid Memory Reference

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

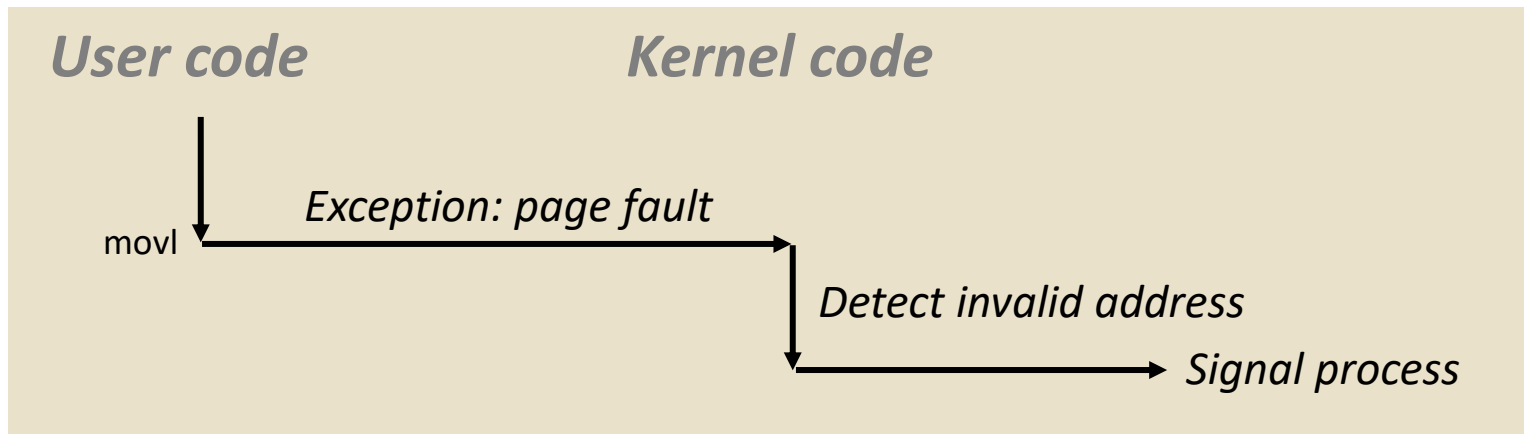
80483b7: c7 05 60 e3 04 08 0d movl \$0xd,0x804e360



Fault Example: Invalid Memory Reference

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

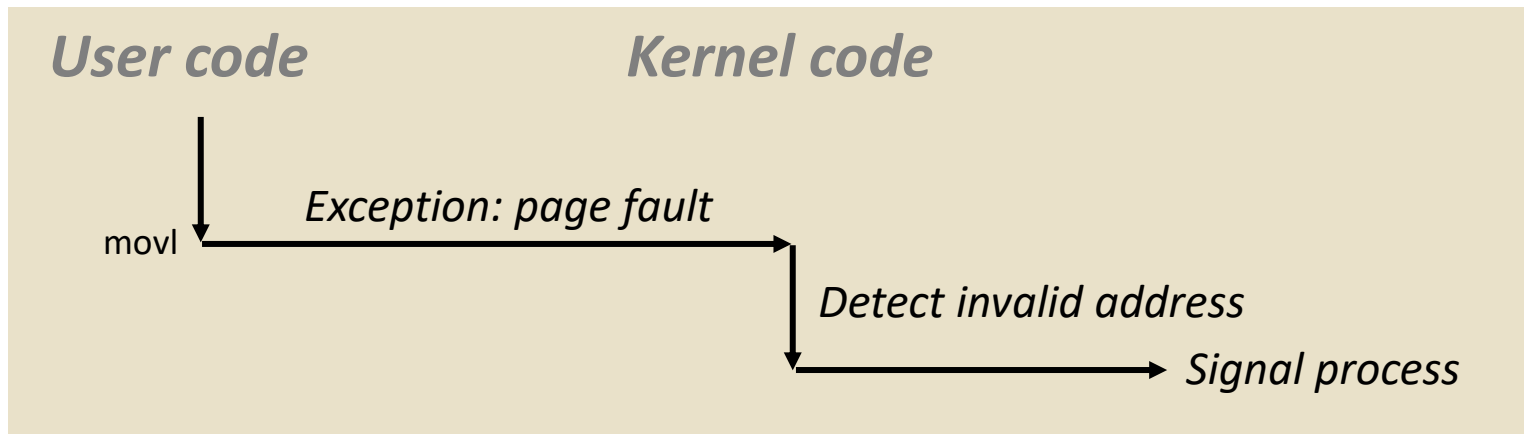
80483b7: c7 05 60 e3 04 08 0d movl \$0xd,0x804e360



Fault Example: Invalid Memory Reference

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

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



- Sends **SIGSEGV** signal to user process
- User process exits with “segmentation fault”