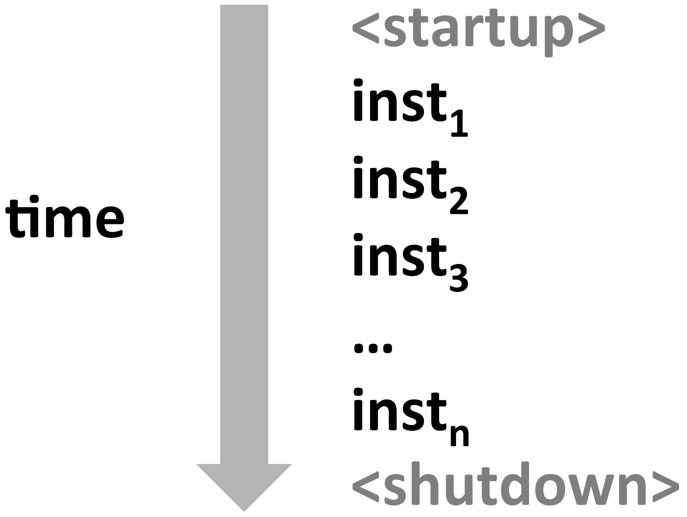
# Control Flow

- **Processors do only one thing:**
  - From startup to shutdown, a CPU simply reads and executes (interprets) a sequence of instructions, one at a time
  - This sequence is the CPU's *control flow* (or *flow of control*)

***Physical control flow***



time

<startup>

$\text{inst}_1$

$\text{inst}_2$

$\text{inst}_3$

…

$\text{inst}_n$

<shutdown>

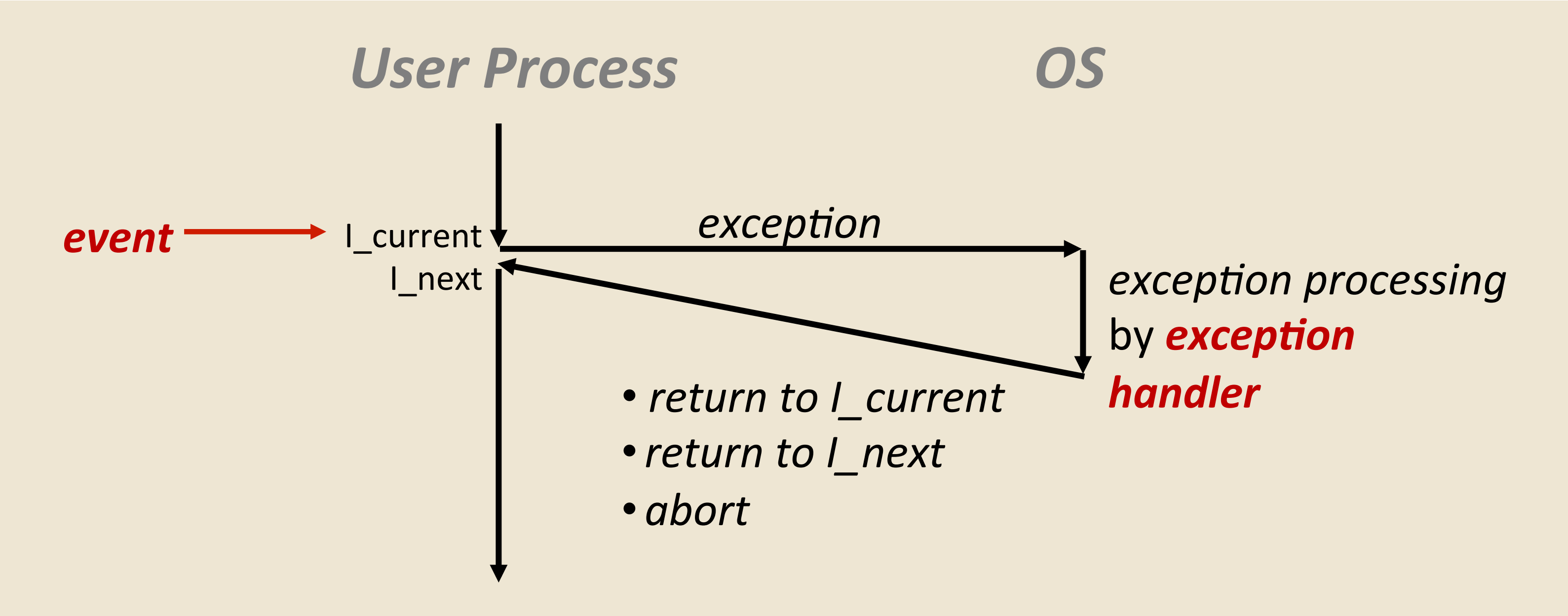
# Altering the Control Flow

- **Up to now: two ways to change control flow:**
  - Jumps (conditional and unconditional)
  - Call and return

  Both react to changes in ***program state***
- **Processor also needs to react to changes in *system state***
  - user hits “Ctrl-C” at the keyboard
  - user clicks on a different application’s window on the screen
  - data arrives from a disk or a network adapter
  - instruction divides by zero
  - system timer expires
- **Can jumps and procedure calls achieve this?**
  - Jumps and calls are not sufficient – the system needs mechanisms for “*exceptional*” control flow!

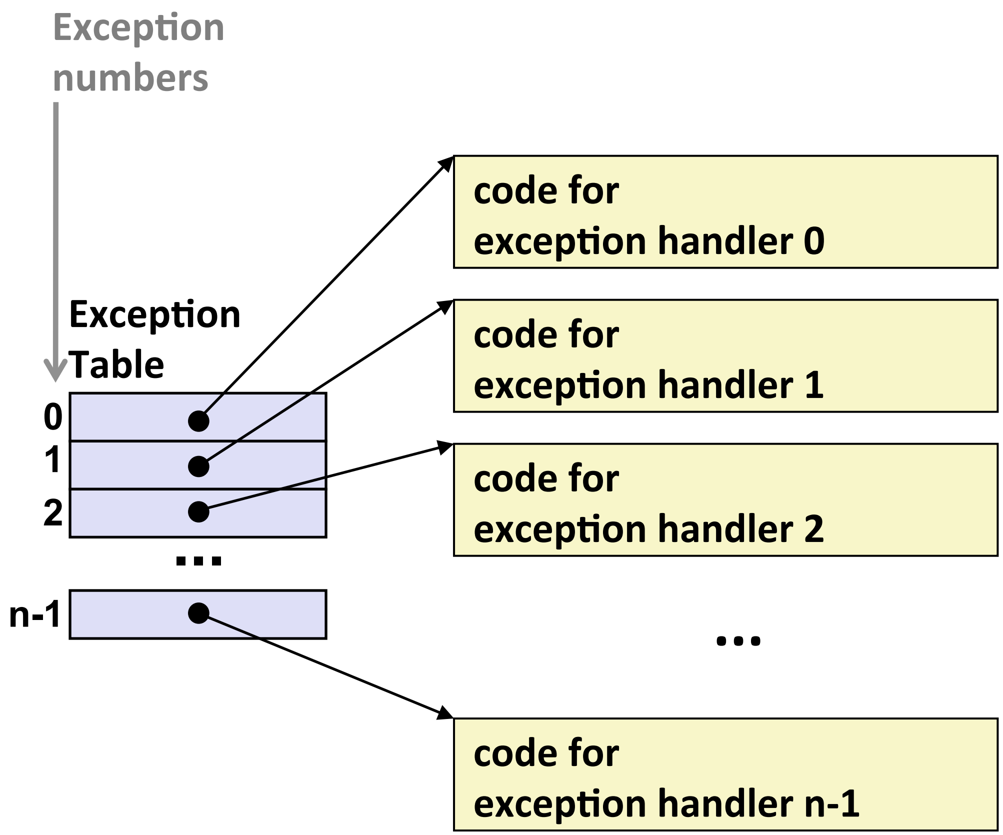
# Exceptions

- An ***exception*** **is transfer of control to the operating system (OS) in response to some** ***event*** **(i.e., change in processor state)**



- **Examples:**
  div by 0, page fault, I/O request completes, Ctrl-C
- *How does the system know where to jump to in the OS?*

# Interrupt Vectors


Exception numbers
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

- 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(s)
  - Handler returns to "next" instruction
- **Examples:**
  - I/O interrupts
    - hitting Ctrl-C on the keyboard
    - clicking a mouse button or tapping a touchscreen
    - arrival of a packet from a network
    - arrival of data from a disk
  - Hard reset interrupt
    - hitting the reset button on front panel
  - Soft reset interrupt
    - hitting Ctrl-Alt-Delete on a PC

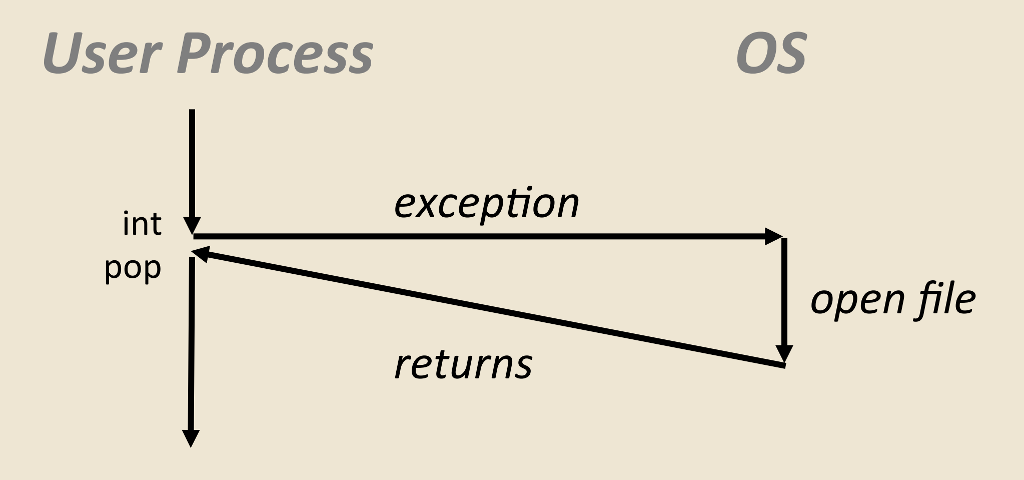
# Synchronous Exceptions

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

# Trap Example: Opening File

- User calls: **`open(filename, options)`**
- Function **`open`** executes system call instruction **`int`**

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



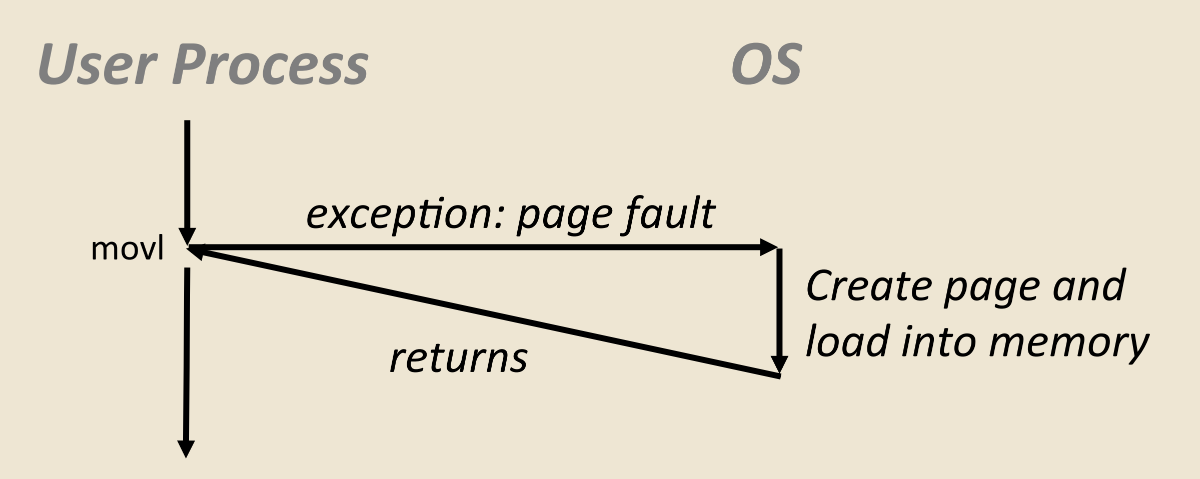
- OS must find or create file, get it ready for reading or writing
- Returns integer file descriptor

# 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
```

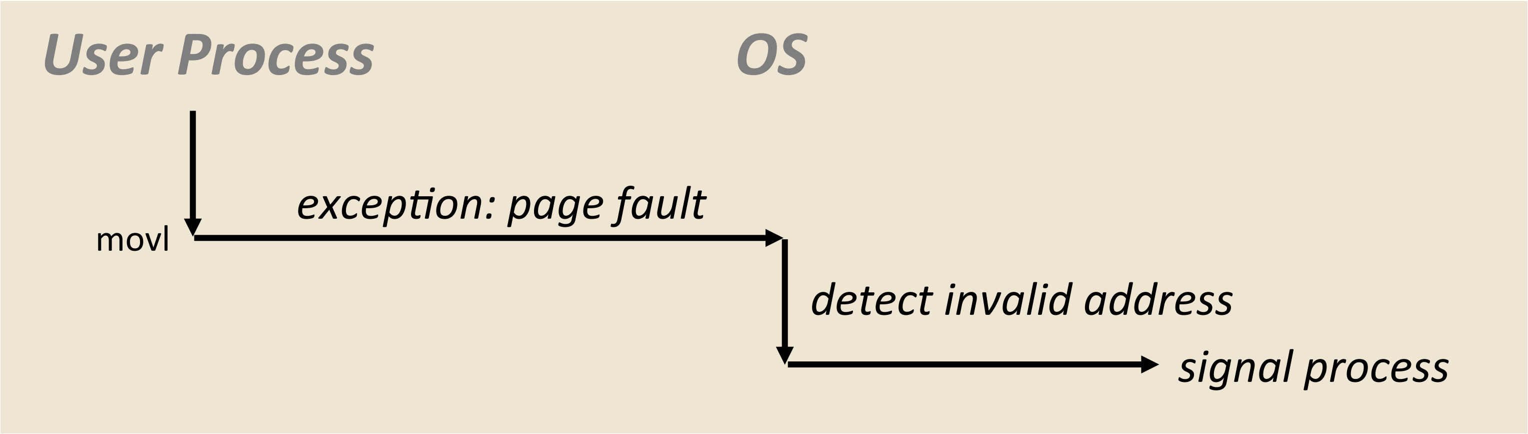


- Page handler must load page into physical memory
- Returns to faulting instruction: **`mov`** is executed again!
- Successful on second try

# Fault Example: Invalid Memory Reference

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

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



- Page handler detects invalid address
- Sends **`SIGSEGV`** signal to user process
- User process exits with “segmentation fault”

# Summary

- **Exceptions**
  - Events that require non-standard control flow
  - Generated externally (interrupts) or internally (traps and faults)
  - After an exception is handled, one of three things may happen:
    - Re-execute the current instruction
    - Resume execution with the next instruction
    - Abort the process that caused the exception