Exceptions

Program Counter (%eip)

\[ \rightarrow 0x8052 \textrm{ mov} \ldots \]
\[ \rightarrow 0x8058 \textrm{ add} \ldots \]
\[ 0x805d \textrm{ mull} \ldots \]

Transfer of Control / Control Transfer

\[ \rightarrow \textrm{ moving to an other instruction} \]
\[ \quad \textrm{sequentially} \]
\[ \quad \textrm{smooth control flow} \]
Control flow $\rightarrow$ abrupt!

- Jump
- Return
- Call

Jump is within the process.

What if it's not?

These kinds of jumps / transfer of control $\rightarrow$ exceptions (reason behind these jumps).

Exception $\rightarrow$ an abrupt change in control flow

Why?

due to some exceptional / anomalous conditions / events that need special processing.
1. Exceptional event
2. Looks up a table called exception table to figure out what to do...
3. Which exception handler should I call?
4. 3 things can happen:
   i) return control to I cur
   ii) return control to I next
   iii) Abort
Note: Exceptions can happen both from hardware/softwore contexts.

How is an exception handled?

Each type of exception has a unique non-negative integer exception number.

On system boot-up, the OS allocates an exception table.

< House - Landlord - Services - Analysis >
Exception Base Table Register

Each entry contains the address of the handler

EBTR + Exception \#k
Before the exception handler takes over, what happens?

Save all the details of the current process (context)

- General purpose registers + control flags
- Stack
- Heap

<You move out of your apartment (along with your stuff) when the pest control is working>
Exceptions

Ganesh Kumar  .  April 22, 2016
Classes of Exceptions
GRRM releases The Winds of Winter!

You pre-order it for first day delivery.

What will you be doing on that day?
OPTION A

Wait at your doorstep for the delivery?
OPTION B

Maybe keep preparing for that midterm the day after while doing your laundry?
OPTION A!!
Option A is highly inefficient. Are we getting any work done?

This event is an asynchronous event.

It happens irrespective of our current activity or status. No need to wait. No control.

These classes of exceptions => **Interrupts**.

Typically triggered by Disk Devices, Network Adapters and Timer Chips.

How?
How?

They trigger a pin on the processor.

And

Place the exception's # that identifies the interrupt handler on the system bus.

Return?

Returns control to the next instruction.
II Trap / System Call

These are intentional exceptions triggered by an instruction.

E.g., `read()`, `exit()`, `write()`, `open()`.

System call → ask the OS to get something done.
Point6 ( )

Calls the write( ) system

call

Trap handler

print6

App

print6

I next

print stuff out

Faults

results from errors that an handler might be able to fix.
① Page fault
\[\downarrow\] Fixed

Returned to the same instruction

\[\downarrow\]

that caused the fault.

② Segmentation Fault
\[\downarrow\] Not fixable

Abort!

IV Abort

\[\Rightarrow\] triggered by unrecoverable fatal errors.

Control is not returned.

Eg. Corrupted memory.
IA32 \rightarrow 256 \text{ Exception types.}

0 - 31 \rightarrow \text{defined by the processor.}

32 - 255 \rightarrow \text{defined by OS.}

0 \rightarrow \text{Divide by zero}

13 \rightarrow \text{segmentation fault}

14 \rightarrow \text{Page Fault}

128 \rightarrow \text{System Calls}.

\text{read} \quad \text{write}