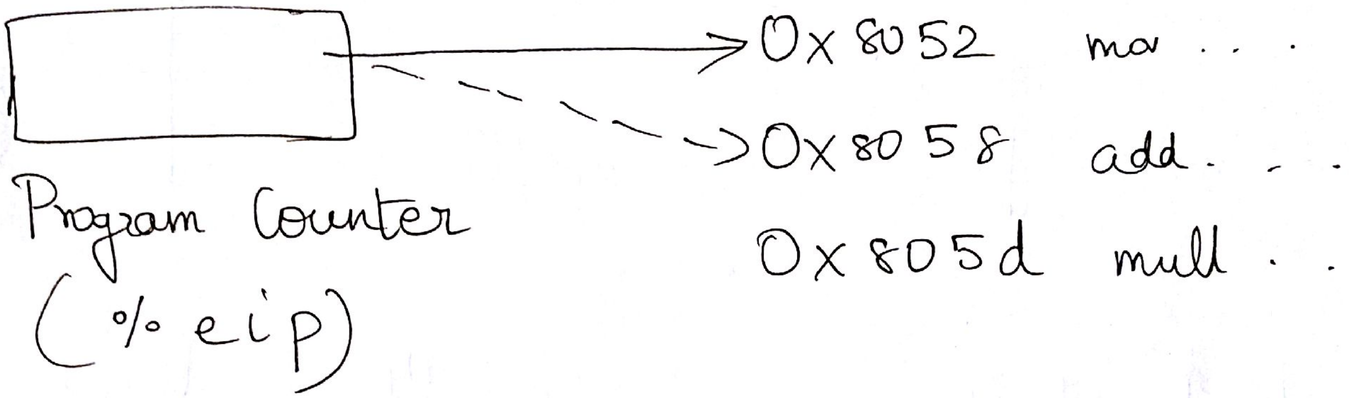
4/22.

Exceptions



Transfer of Control / Control Transfer → moving to an other instruction sequentially.

↓

Smooth control flow.

Control flow → abrupt!

Jump return call

Jump is within the process.

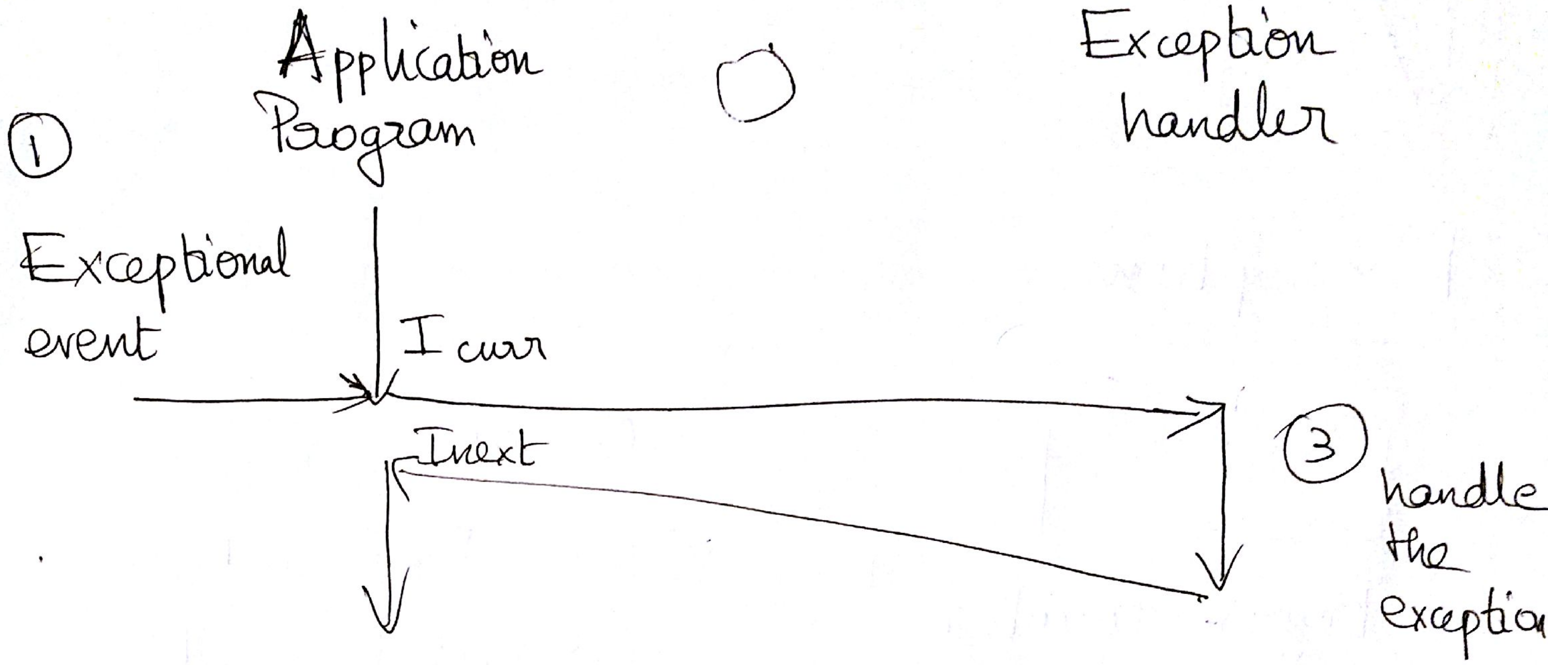
What if its not?

These kinds of jumps / transfer of control → exceptions (reason behind these jumps).

Exception → an abrupt change in control flow

Why?

due to some exceptional / anomalous conditions / events that need special processing.



② Looks up a table called exception table to figure out what to do . . .

Which exception handler should I call ?

④ 3 things can happen.

i) return control to I curr

ii) return control to I next

iii) Abort //

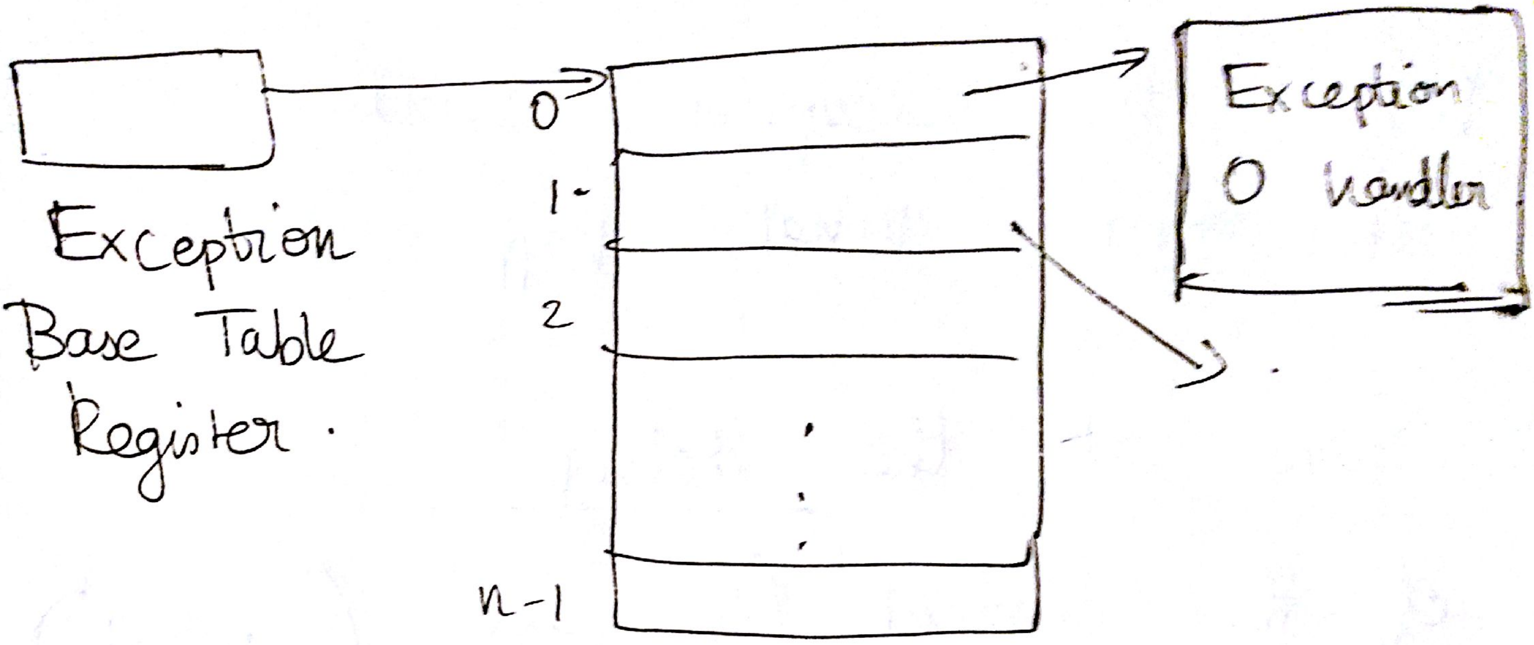
Note : Exceptions can happen both from hardware / software contexts.

How is an exception handled ?

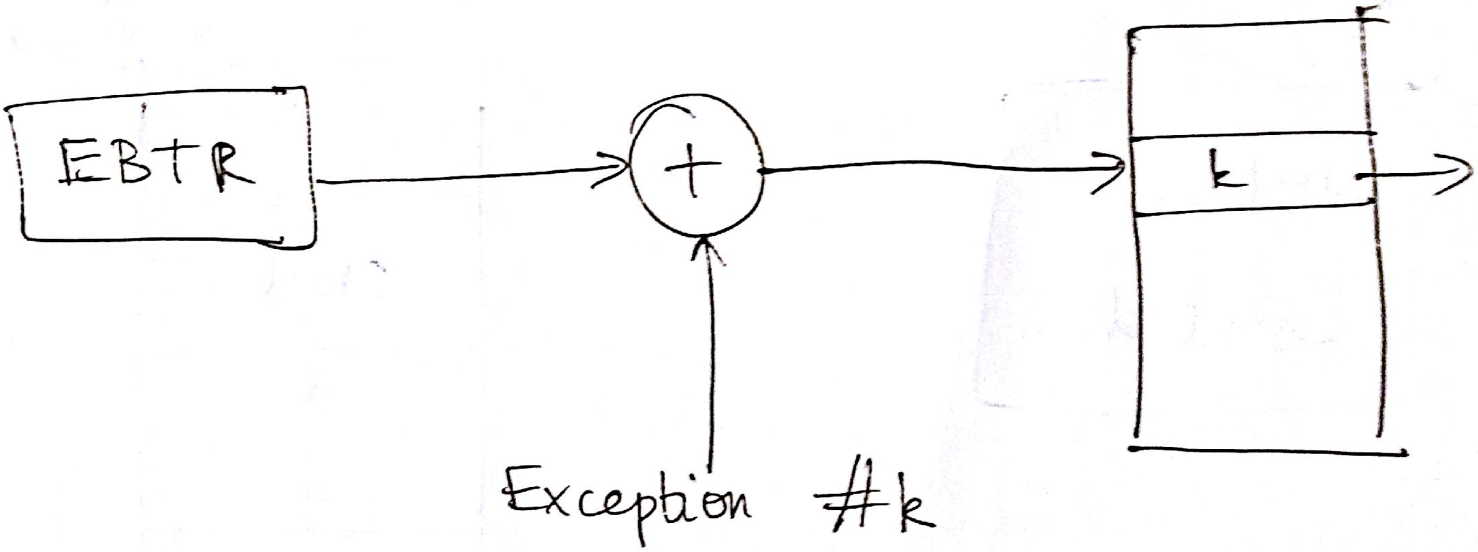
Each type of exception $\xrightarrow{\text{has}}$ unique non-negative integer exception #

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

< House - Landlord - Services - Analogy >



Each entry contains ~~[illegible]~~. address of the handler



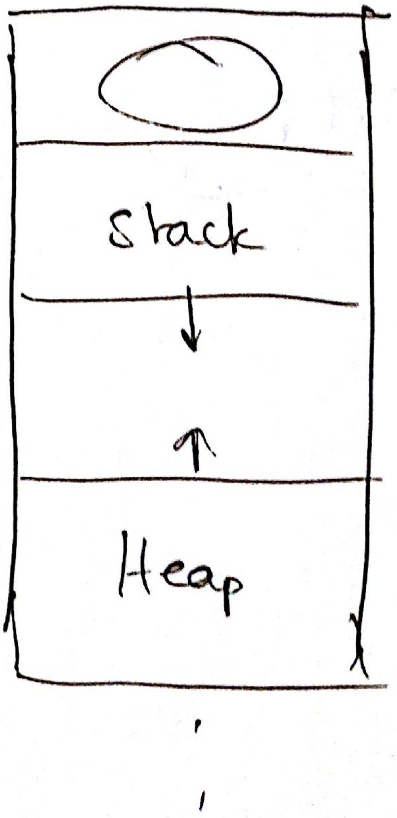
Before the exception handler takes over, what happens ?

Save all the details of the current process (context)

↓

general purpose registers + control flags //

Context Switch



<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 the 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 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 ? I Interrupts

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

↳ are intentional ~~expen~~ exceptions.

↓

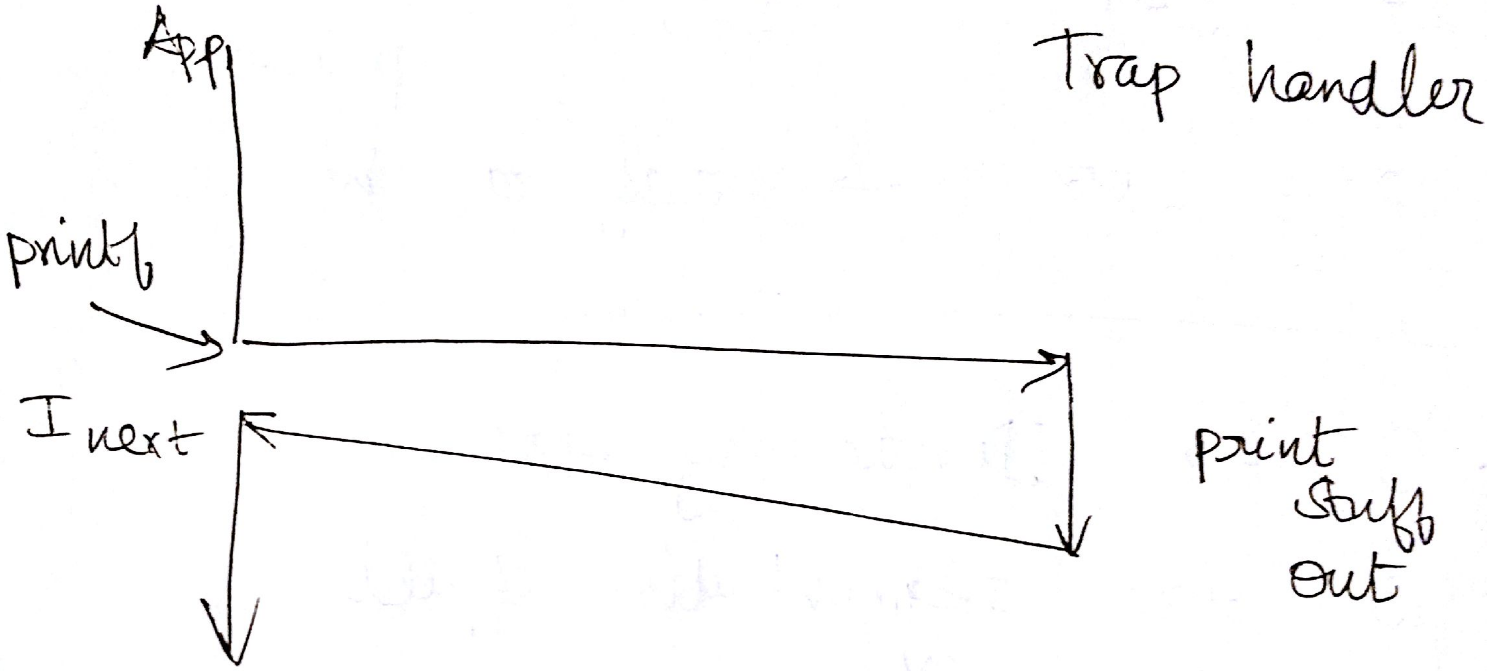
triggered by an instruction.

Eg. read ( ), exit ( ),
write ( ), open ( ).

System call → ask the OS to get something done.

Printf ( )

Calls the write ( ) system call.



III Faults

↳ results from errors that an handler <u>might</u> be able to fix.

① Page fault

↓ Fixed

Returned to the same instruction

↓

that caused the fault.

② Segmentation Fault.

↓ Not fixable

Abort!

IV Abort

→ triggered by unrecoverable fatal errors.

Control is not returned.

Eg. Corrupted memory.

IA32 → 256 Exception types.

0 - 31 → defined by the processor.

32 - 255 → defined by OS.

---

0 → Divide by zero

13 → segmentation fault.

14 → Page Fault

128 → System Calls. ✓

read write