

Exceptional Control Flow



Exceptional Control Flow (Chapter 8)

I Context

CPU \leftrightarrow Infinite loop of

Fetch
Decode
Execute

Most of the time Next Instruction
in valP

Rest (that we know about)

jump
call
ret

This non-linear changes
in control is

React to changes in program state

Q: why is this not sufficient for a system

A system is a set of interactive / interdependent
component parts forming a complex / intricate
whole

In computer systems,

Processor

= Disk - File system

I/O Devices

= Network

= GPU / Mouse / Keyboard

OS / kernel

Processes

Distributed -- change the boundaries
Networking

look at
Process
list

A: Need to "react" to changes in system state

Data arrives from

Disk
Network

Keyboard

Graphics Card

+ Divide by 0

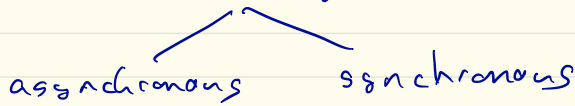
Access illegal mem

CtrlC of keyboard

System timer expires \leftrightarrow Run a different process

Exceptional Control Flow

-- change instruction sequence to respond to changes in system state



Exist at all levels of a computer system

HW/OS

1. Exceptions / Interrupts

OS

2. Process context switch

OS/Process

3. Signals

Process

4. Non local jumps

direct: setjump / longjump

HLL: C++ / Java Exception catch