Tests & Quizzes Status

Test	# Submissions	x <= 50%	50% < x <= 80%	80% < x < 100%	100%
1 Introduction & Motivation	37	12	15	6	4
C1 Hello World & Compiling	54	3	5	20	26

FREIE UNIVERSITÄT BERLIN

Operating Systems & Computer Networks
2. Interrupts & System Calls

Dr. Larissa Groth
Computer Systems & Telematics (CST)

Roadmap

1. Introduction and Motivation
2. **Interrupts and System Calls**
3. Processes
4. Scheduling
5. Memory
6. I/O and File System
7. Booting, Services, and Security

Lernziele

Sie nennen:

- die wesentlichen Schritte des Interrupt Handlings, aufgeteilt in HW und SW
- Beispiele für typische HW Interrupts und typische SW Interrupts

Sie beschreiben:

- den Unterschied zwischen Interrupt Handling und Polling
- die Rolle der Interrupt Service Routine
- die Funktionsweise der Interrupt Vector Table
- den Unterschied zwischen Mode Switches und Process Switches

Sie stellen dar:

- die Erweiterung des Instruction Cycle um die Interrupt Detection und das Interrupt Handling
- die vier möglichen Implementierungen von System Calls

Sie wenden an:

- Sequential und Nested Interrupt Processing auf eine gegebene Folge von Interrupts (mit Ankunftszeiten, Ausführungszeiten und Prioritäten)

Sie wägen die Vor- und Nachteile ab:

- zwischen Interrupts und Polling

Interrupts

Motivation

A lot of devices are connected to a computer, e.g., keyboard, hard disk, network interface

These devices occasionally need CPU service:

- Keyboard: a key is pressed
- Hard disk: a load/store-task is completed
- Network interface: a packet has arrived

BUT: it is **not predictable** when these devices need to be serviced

How does the CPU find out that a device needs attention?
Two options: **Interrupts** and **Polling**

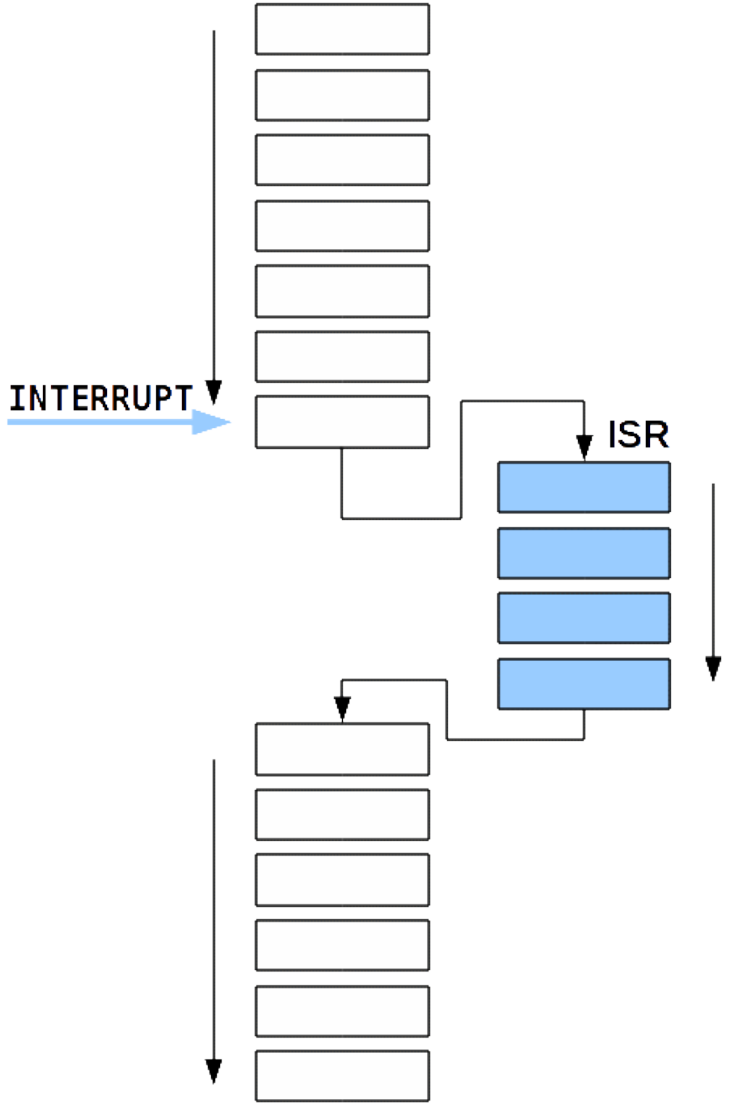
Interrupts vs. Polling

Interrupts	**Polling**
Give each device a wire that it can use to signal the CPU.	Ask the devices periodically if an event has occured.
Like a phone that rings when a call comes in.	Like a phone without a bell: You have to pick it up every few seconds to see if you have a call.
No overhead when no requests pending, efficient use of CPU time.	Takes CPU time even when no requests pending.
Devices are serviced as soon as possible - low latency.	Response time depends on polling rate.

Interrupt Service Routine (ISR)

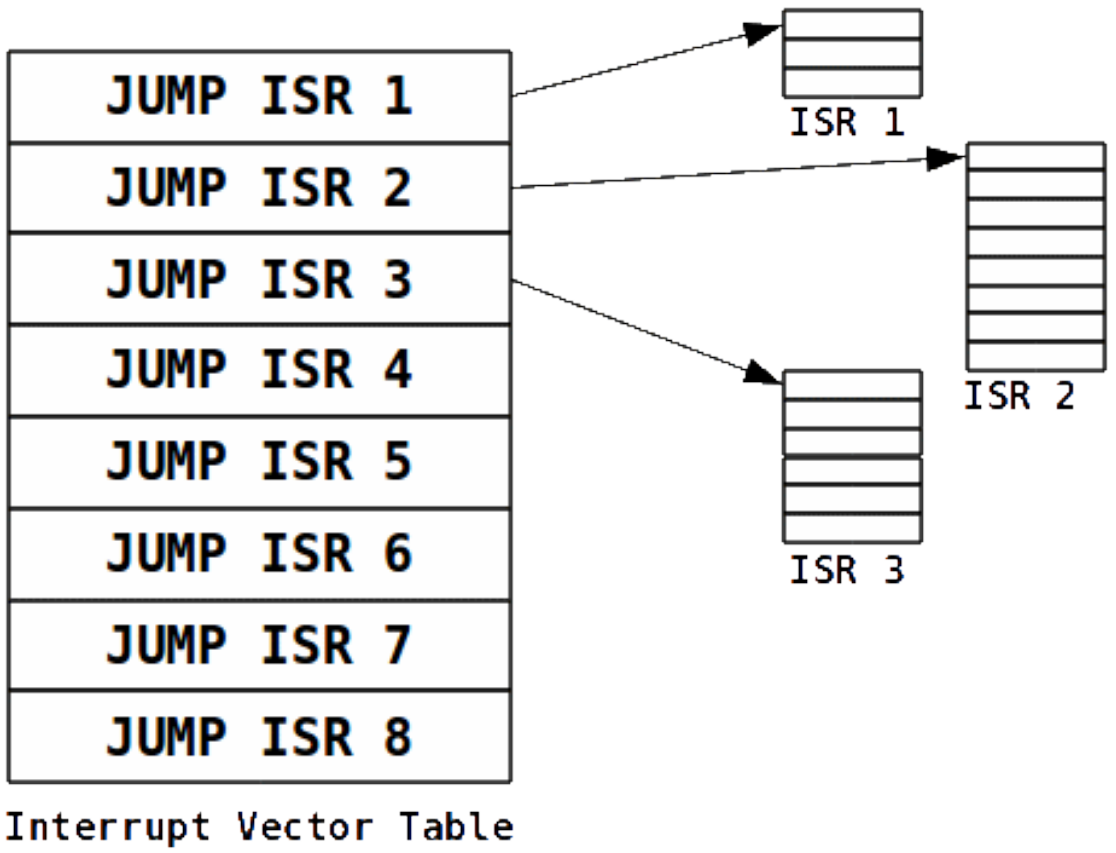
Interrupt handling is performed by the operating system (device drivers) in **interrupt service routines** (ISRs).

Interrupts temporarily discontinue the currently executing application.



Interrupt Vector Table (IVT)

- **Interrupt vector table** maps interrupts to service routines that handle them
- Table has one entry for each interrupt
- Each entry contains the address of the ISR
  (interrupt vector)
- Table resides in main memory at a constant address (interrupt base address)
- Interrupt number provides index into the table



Detecting Interrupts

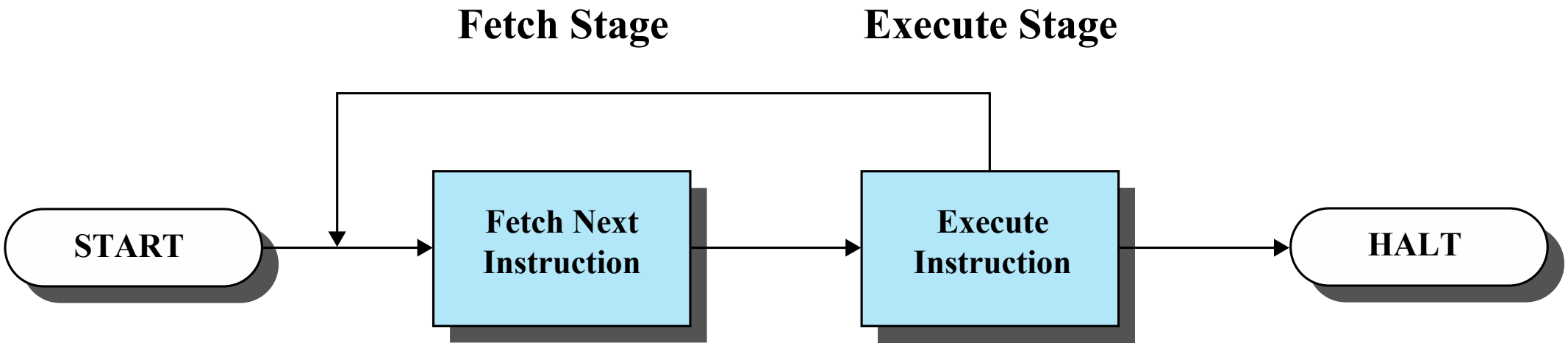


Figure 1.2 Basic Instruction Cycle

Detecting Interrupts

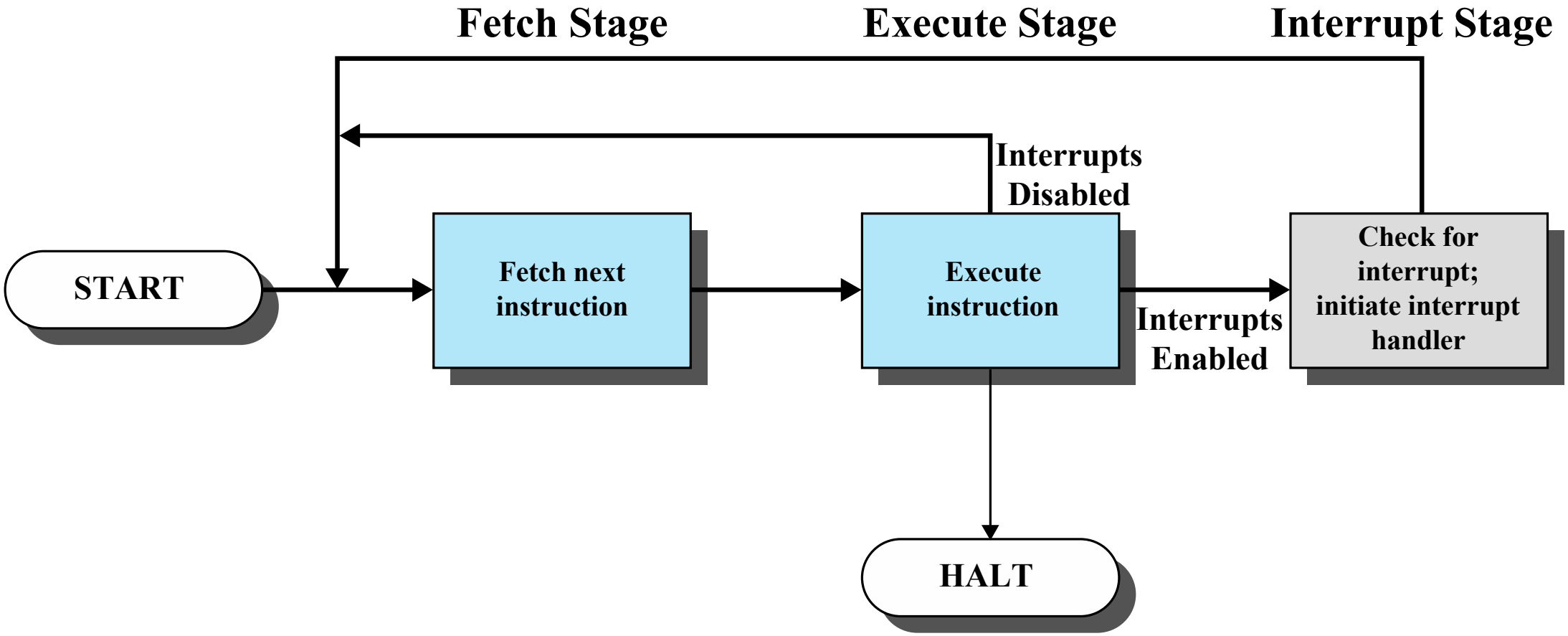
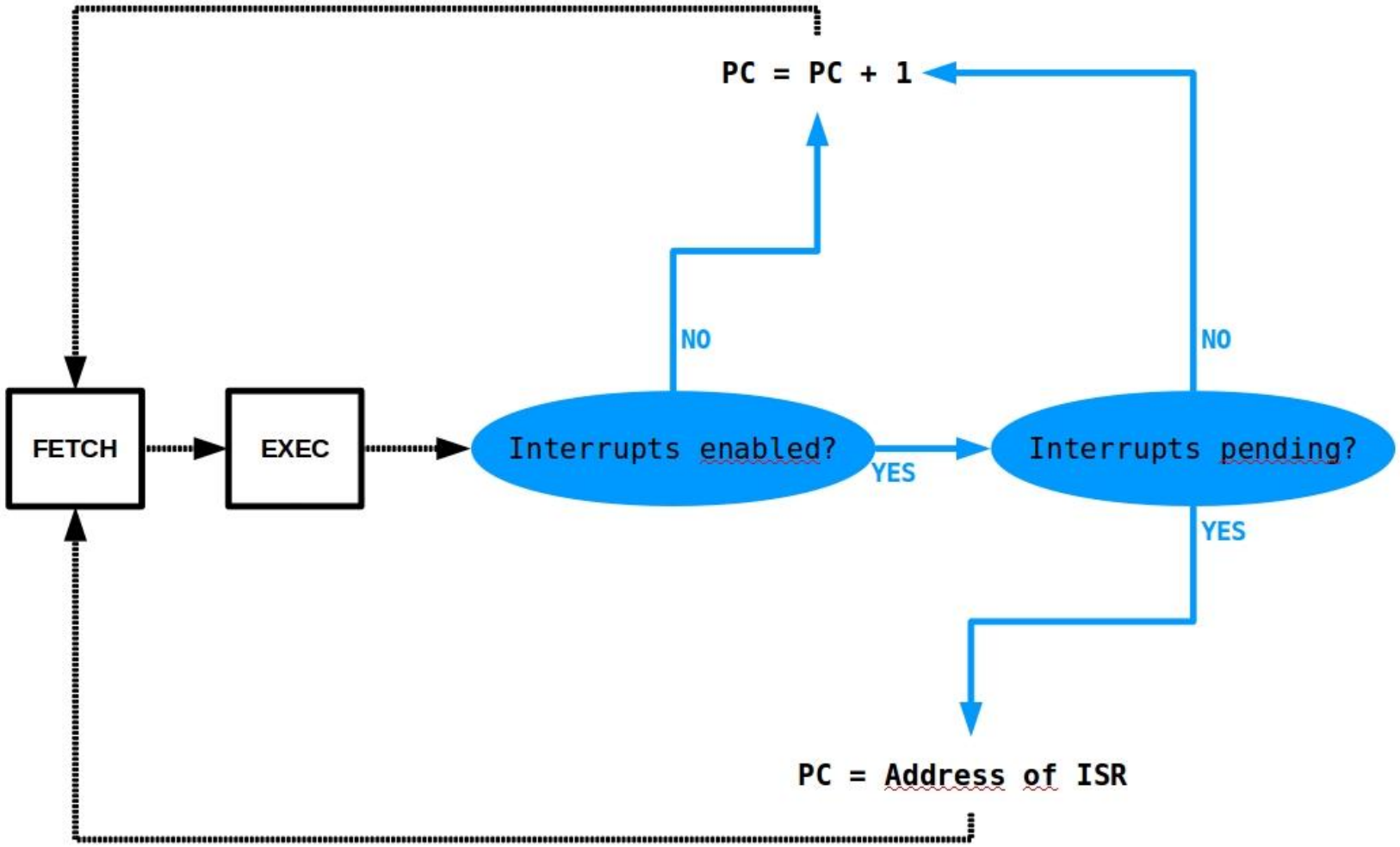


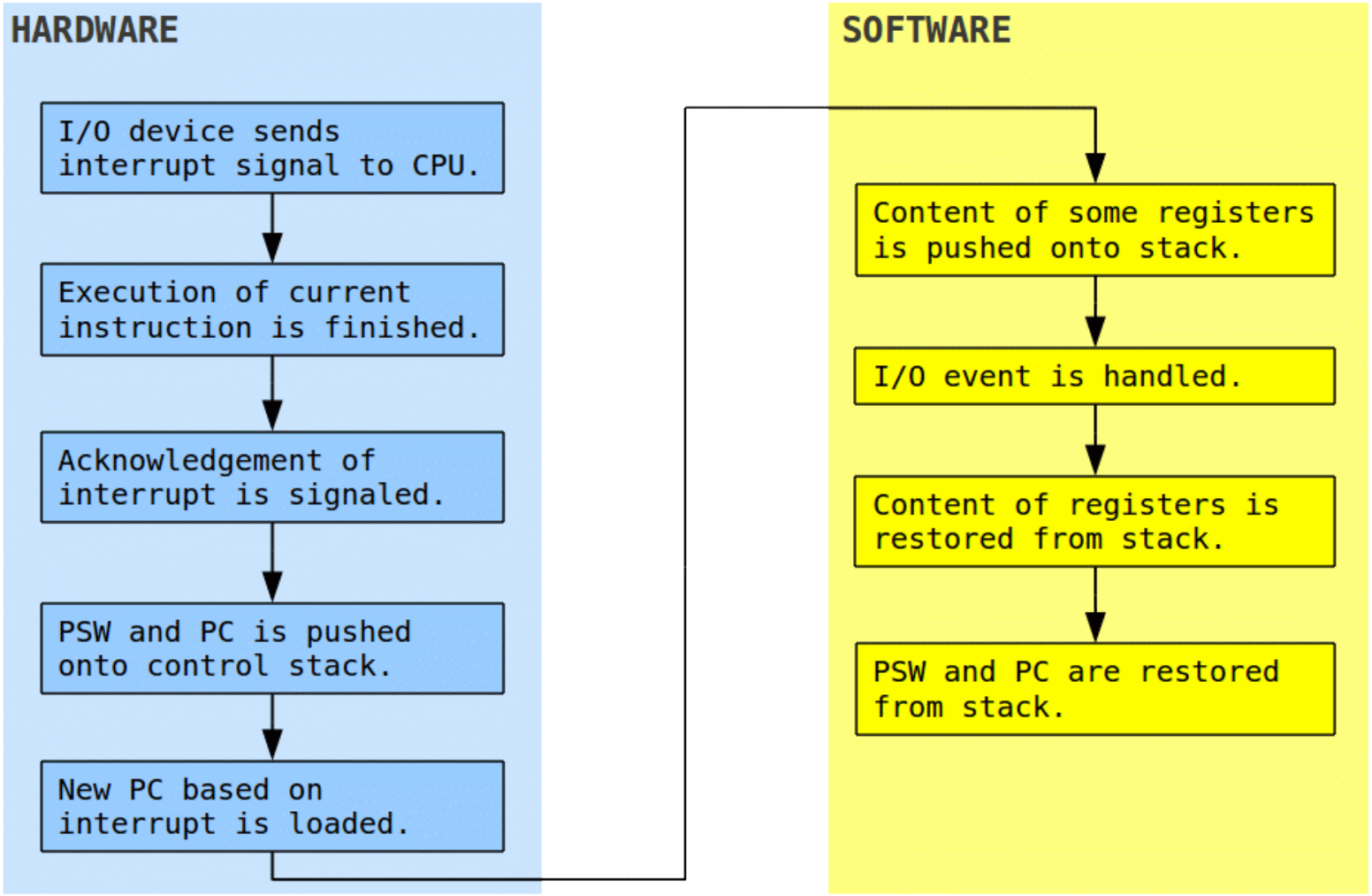
Figure 1.7 Instruction Cycle with Interrupts

Detecting Interrupts



Steps of Interrupt Handling

Example: Handling of an I/O event, assuming that execution of interrupted process resumes afterwards



Types of Interrupts

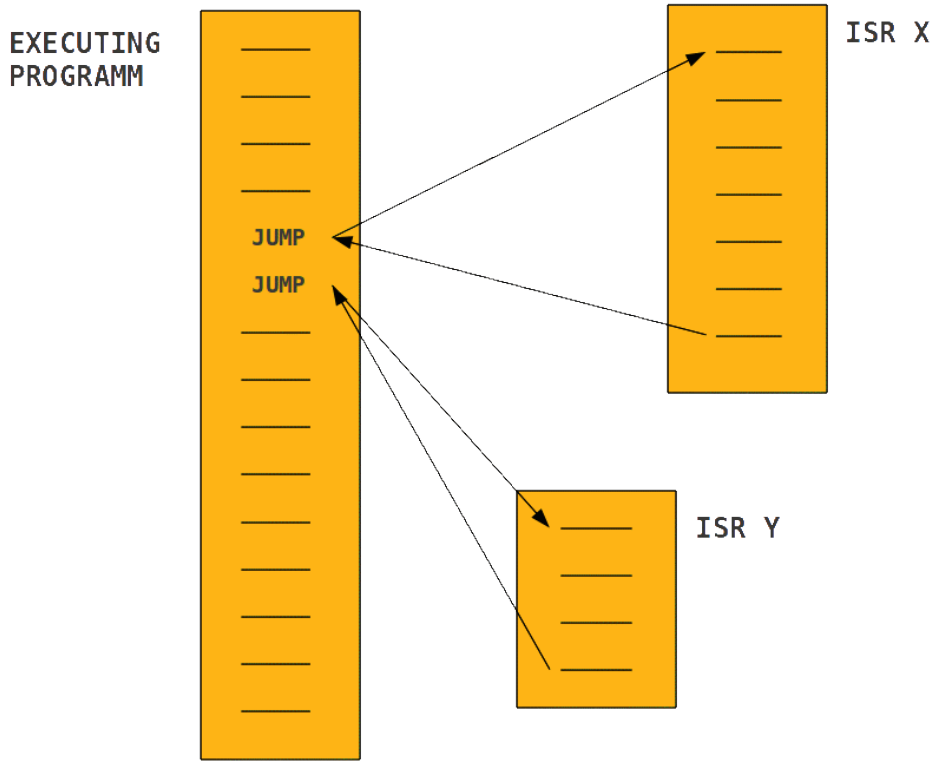
Hardware interrupts (asynchronous)
- Triggered by hardware devices, e.g.,
  - Timer
  - I/O device
  - Printer

Software interrupts (synchronous)
- Triggered within a processor by executing an instruction (INT)
- Often used to implement system calls

Exceptions, e.g.,
- Arithmetic overflow, division by zero
- Illegal instruction
- Illegal memory access

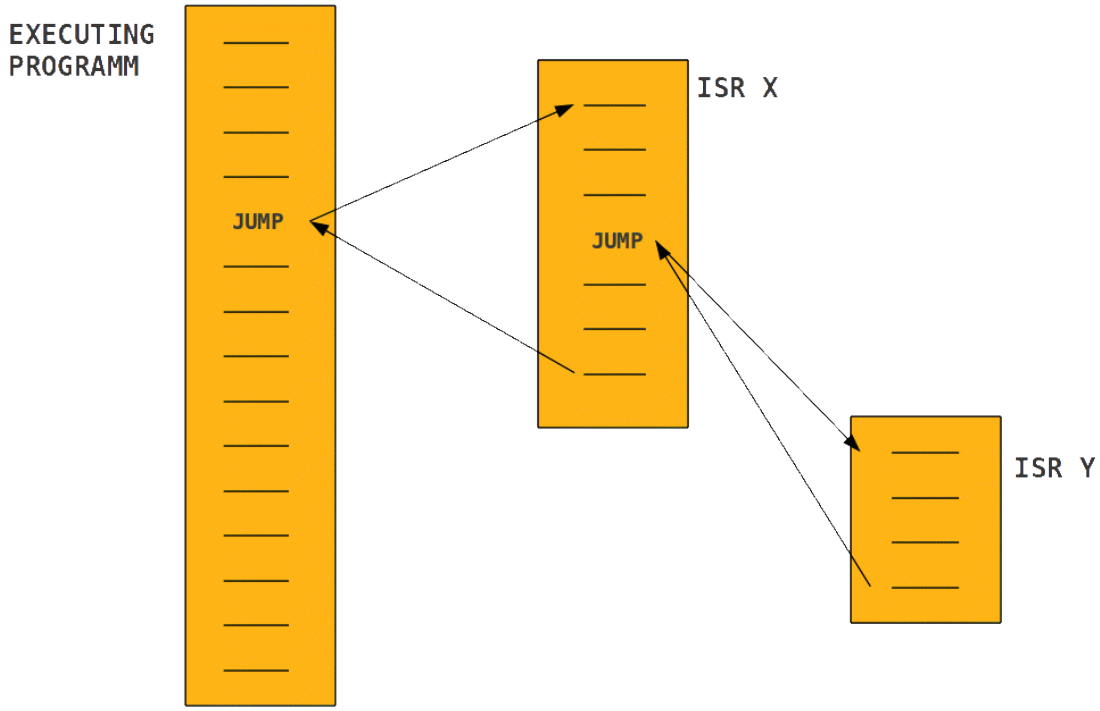
Multiple Interrupts

Sequential interrupt processing:



Delay of interrupt handling **unpredictable** under load

Nested interrupt processing:



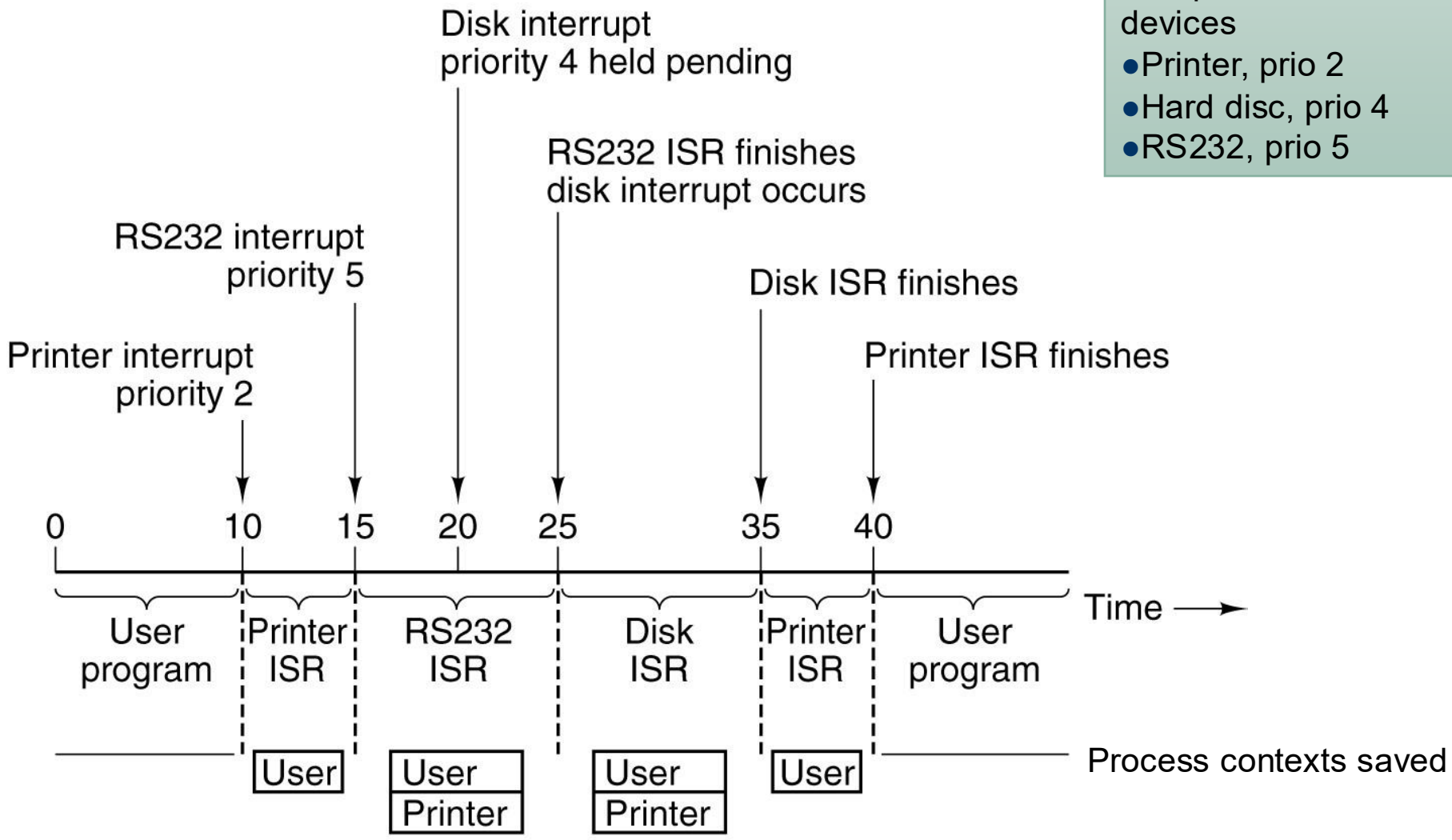
Delay depends on interrupt priority level

Highest priority guarantees **constant** delay

Required for real-time applications

Time sequence of multiple interrupts

Computer with 3 I/O devices
- Printer, prio 2
- Hard disc, prio 4
- RS232, prio 5



Roadmap

1. Introduction and Motivation
2. **Interrupts and System Calls**
3. Processes
4. Scheduling
5. Memory
6. I/O and File System
7. Booting, Services, and Security