THE UNIVERSITY OF NEWCASTLE AUSTRALIA

FACULTY OF ENGINEERING AND BUILT ENVIRONMENT

www.newcastle.edu.au

OPERATING SYSTEMS
Week 1

Much of the material on these slides comes from the recommended textbook by William Stallings

Detailed content

Weekly program

- ❑ **Week 1 – Operating System Overview**
- ❑ Week 2 – Processes and Threads
- ❑ Week 3 – Scheduling
- ❑ Week 4 – Real-time System Scheduling and Multiprocessor Scheduling
- ❑ Week 5 – Concurrency: Mutual Exclusion and Synchornisation
- ❑ Week 6 – Concurrency: Deadlock and Starvation
- ❑ Week 7 – Memory Management I
- ❑ Week 8 – Memory Management II
- ❑ Week 9 – Disk and I/O Scheduling
- ❑ Week 10 – File Management
- ❑ Week 11 – Security and Protection
- ❑ Week 12 – Revision of the course
- ❑ Week 13 – Extra revision (if needed)

Announcement

- Workshop 0 on week 1

  11:00 -13:00 Thursday 02/08/2018 (EA101)
  9:00 -11:00 Thursday 02/08/2018 (ES238)
- No workshop on week 2
- Regular workshop from week 3-12.

Week 01 Lecture Outline

Operating System Overview

- ❑ What is an OS?
- ❑ OS role/objectives
- ❑ Hardware Review
- ❑ Instruction Execution
- ❑ Interrupts

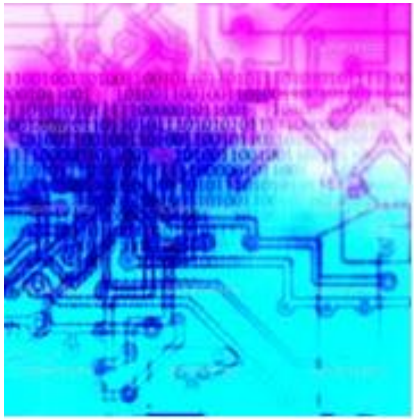
What is an Operating System?

- Easier to give an example

- May be easier to talk about the features of OS
  - Ubuntu 18.04, will support middle click of your mouse
  - Windows 10 has its Start Menu back
  - Android Oreo features smart text selection

- But these features do not define what OS is
    - OS's are usually invisible to the user (lurking somewhere behind the GUI....)

Operating System as an Abstract Machine

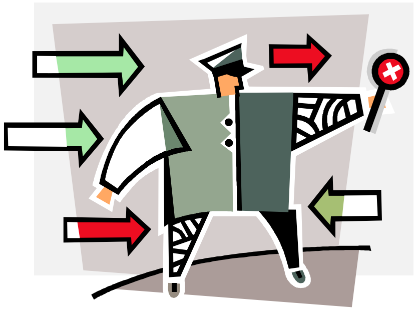


- Extends the basic hardware with added functionality
- Provides high-level abstractions
  - More programmer friendly
  - Common core for all applications
- It hides the details of the hardware
  - Makes application code portable

Operating System as a Service Provider

- Program development
- Program execution
- Access I/O devices
- Controlled access to files
- System access
- Error detection and response
- Accounting

Operating System as a Resource Manager



- Responsible for allocating resources to users and processes

- Must ensure
  - No starvation and ongoing progress
  - Allocation is according to some desired policy

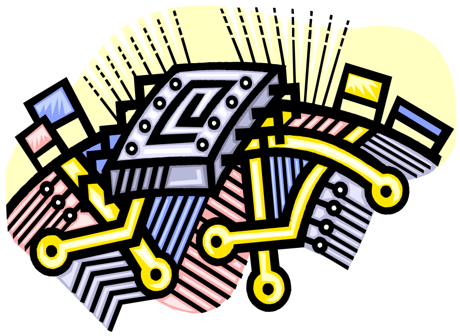
- Applications should not be able to interfere or bypass the operating system
  - OS can enforce the “extended machine”
  - OS can enforce resource allocation policies

- First-come, first-served; Fair share; Weighted fair share; limits (quotas), etc…
  - Overall, that the system is efficiently used

Still OS is a Software

- Functions in the same way as ordinary computer software
- Program, or suite of programs, executed by the processor
- Frequently relinquishes control and must depend on the processor to allow it to regain control

- Key difference is
  - In the intend of the program
  - OS directs the processor in the use of system resources



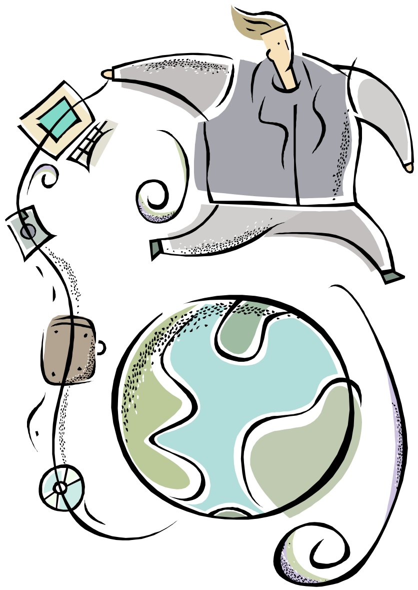
Operating System Objectives

- **Convenience**
  – Makes a computer more convenient to use
- **Efficiency**
  – Allows the computer resources to be used in an efficient manner
- **Ability to evolve**
  – Should permit the effective development, testing and introduction of new system functions without interfering with service.

Evolution of Operating Systems

- A major OS will evolve over time for a number of reasons:

- hardware upgrades
- new types of hardware
- new services
- Fixes



What the OS does …

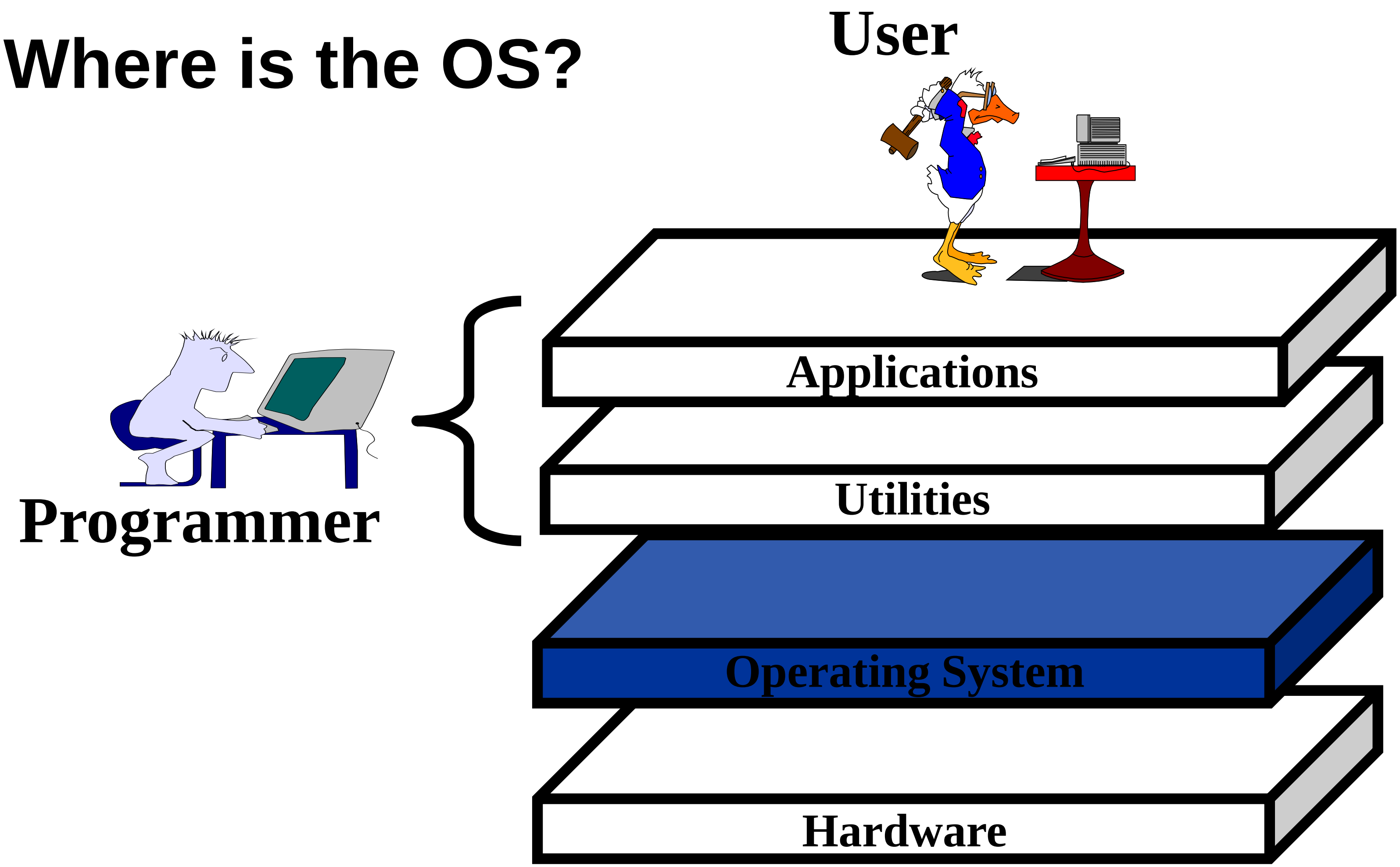
- OS separates applications from the hardware they access
  - Provides services that allow each application to execute safely and effectively.

- OS is a “black box” between the applications and the hardware they run on that ensures the proper result, given appropriate input.

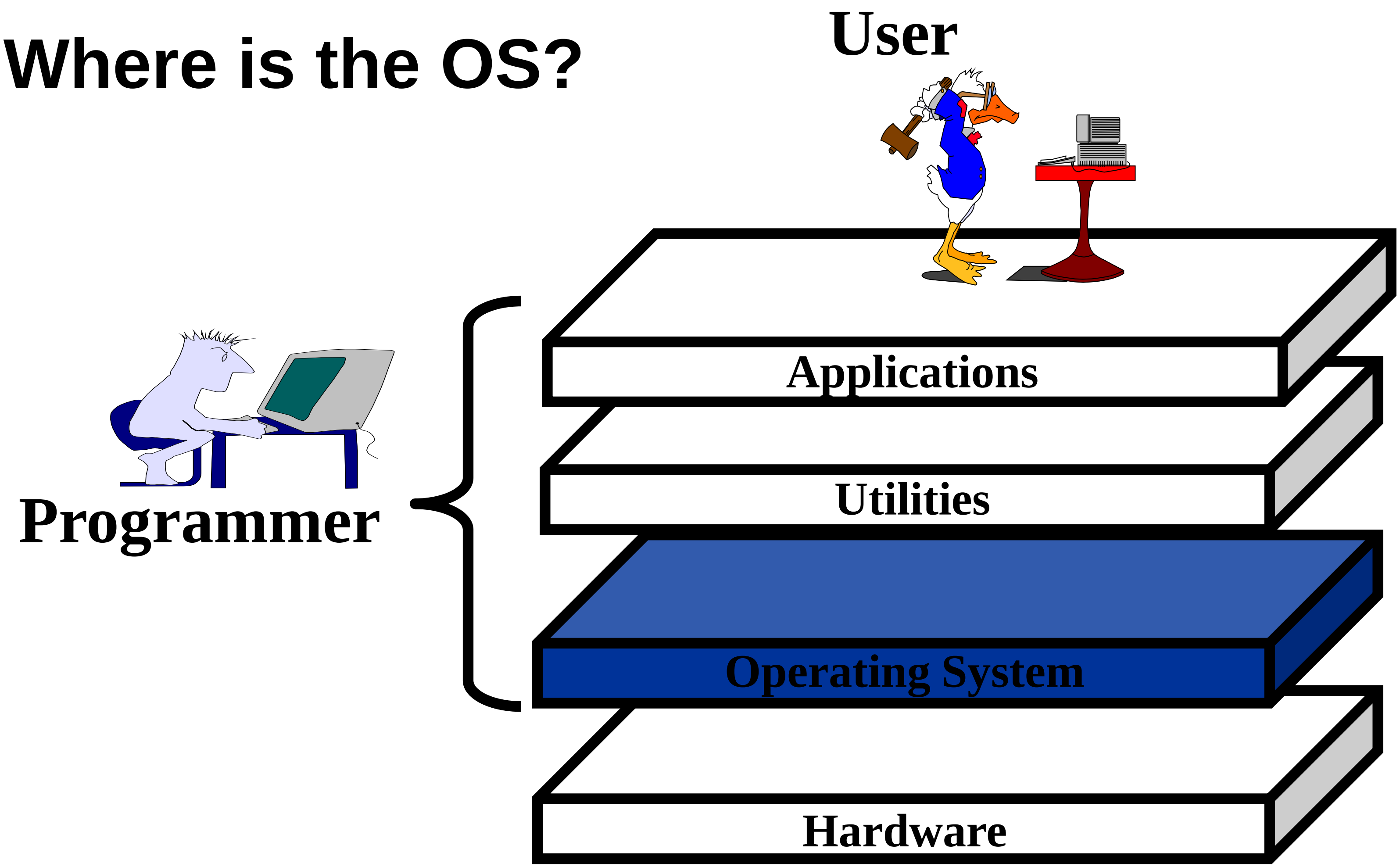
- Operating systems are primarily resource managers
  - Manage hardware, including processors, memory, input/output device and communication devices.

- Operating systems must also manage application and other software abstractions

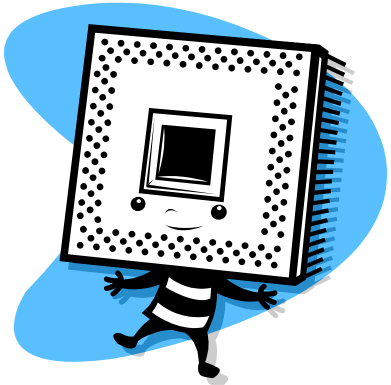
Where is the OS?



Where is the OS?



Hardware Review

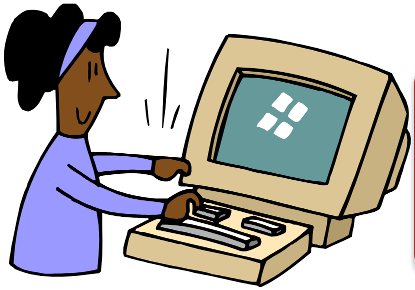


- Operating systems
  - Exploit the hardware available
  - Provide a set of high-level services that represent or are implemented by the hardware.
  - Manages the hardware reliably and efficiently
- *Understanding operating systems requires a basic understanding of the underlying hardware*

Basic Elements

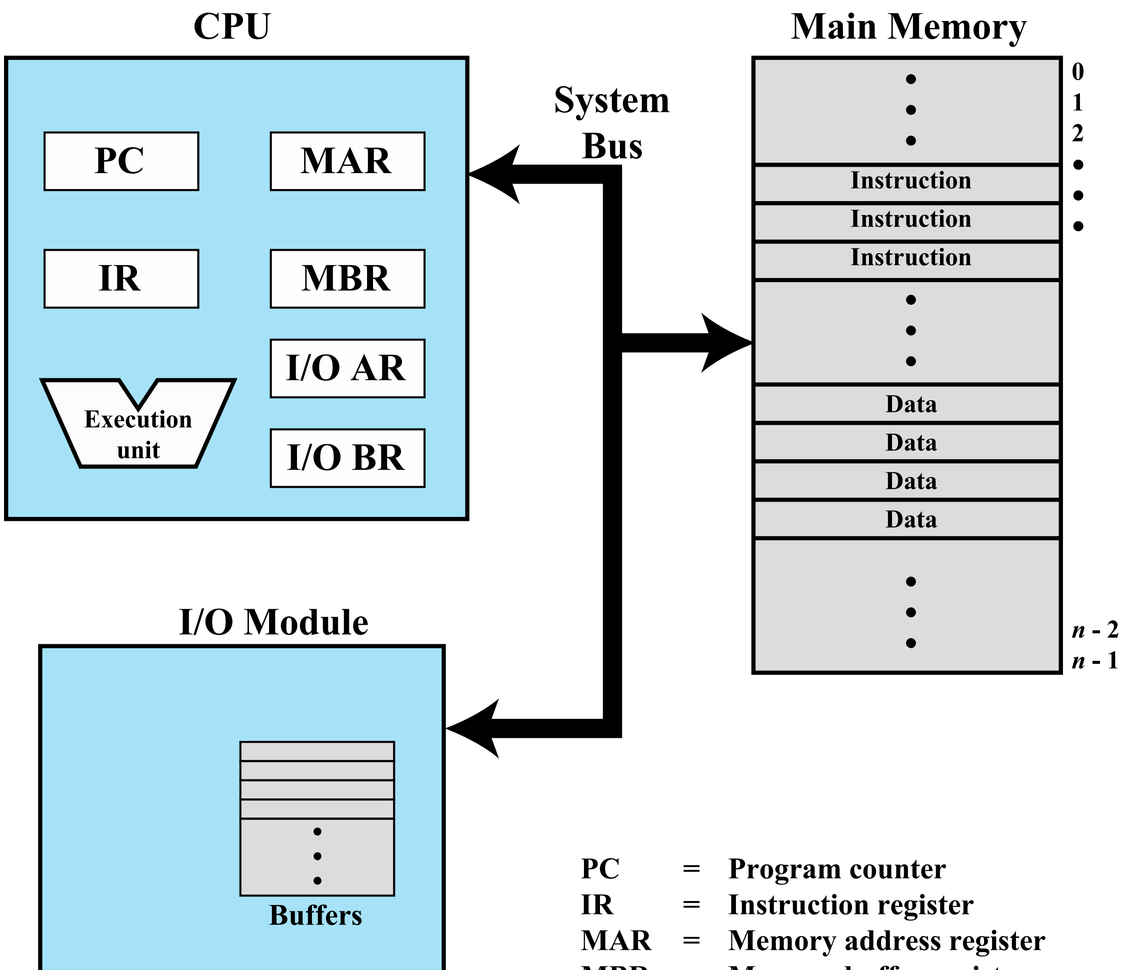
Processor

I/O Modules



Main Memory

System Bus

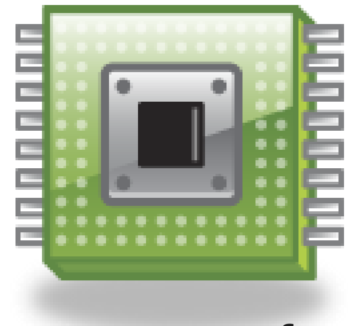


PC	=	Program counter
IR	=	Instruction register
MAR	=	Memory address register
MBR	=	Memory buffer register
I/O AR	=	Input/output address register
I/O BR	=	Input/output buffer register

Microprocessor

- Invention that brought about desktop and handheld computing
- Processor on a single chip
- Fastest general purpose processor
- Multiprocessors
- Each chip (socket) contains multiple processors (cores)

Other Processing Units



- GPUs provide efficient computation on arrays of data using Single-Instruction Multiple Data (SIMD) techniques
- DSPs deal with streaming signals such as audio or video
- To satisfy the requirements of handheld devices, the microprocessor is giving way to the SoCs (System on a Chip) which have components such as DSPs, GPUs, codecs and main memory, in addition to the CPUs and caches on the same chip

Instruction Execution

- A program consists of a set of instructions stored in memory

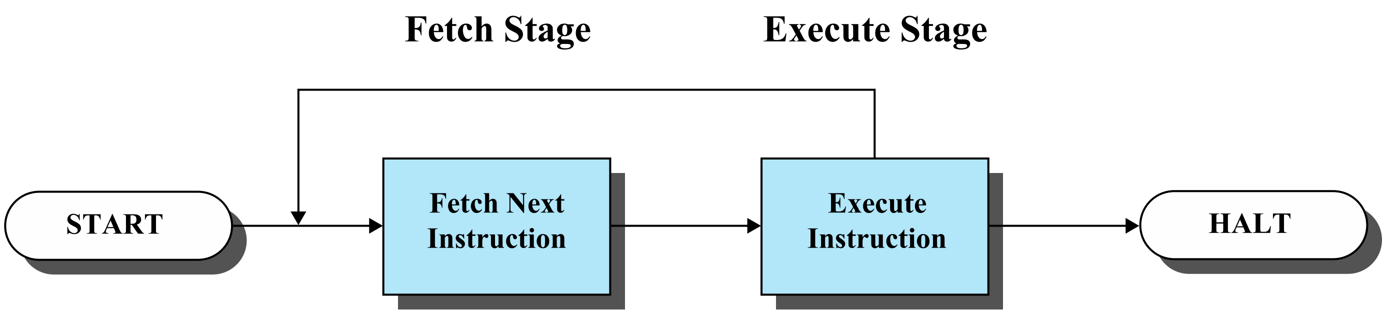
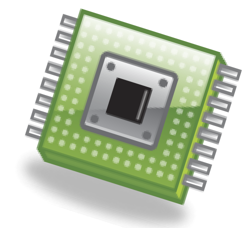


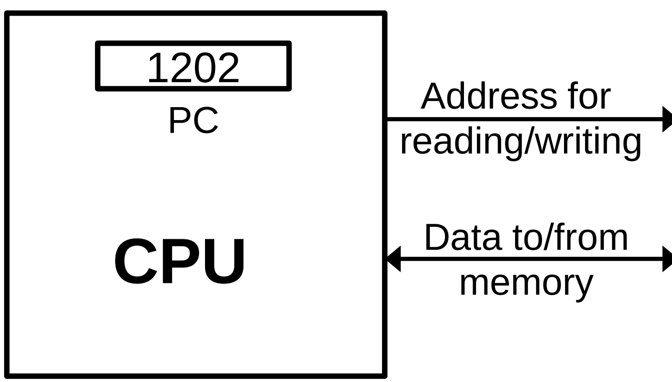
Figure 1.2 Basic Instruction Cycle

…	….
1201	LDA 1510
1202	LDB 1511
1203	ADD A, B
1204	ST A, 1512
1205	HLT
..	
1510	10
1511	20
1512	
….	

Instruction Fetch and Execute

- The processor fetches the instruction from memory
- Program counter (PC) holds address of the instruction to be fetched next
  - PC is incremented after each fetch





…	….
1201	LDA 1510
1202	LDB 1511
1203	ADD A, B
1204	ST A, 1512
1205	HLT
..	
1510	10
1511	20
1512	
….	

Instruction Register (IR)

Fetched instruction is loaded into Instruction Register (IR)

- Processor interprets the instruction and performs required action:
  - Processor-memory
  - Processor-I/O
  - Data processing
  - Control

0–3	4–15
Opcode	Address

(a) Instruction format

0	1–15
S	Magnitude

(b) Integer format

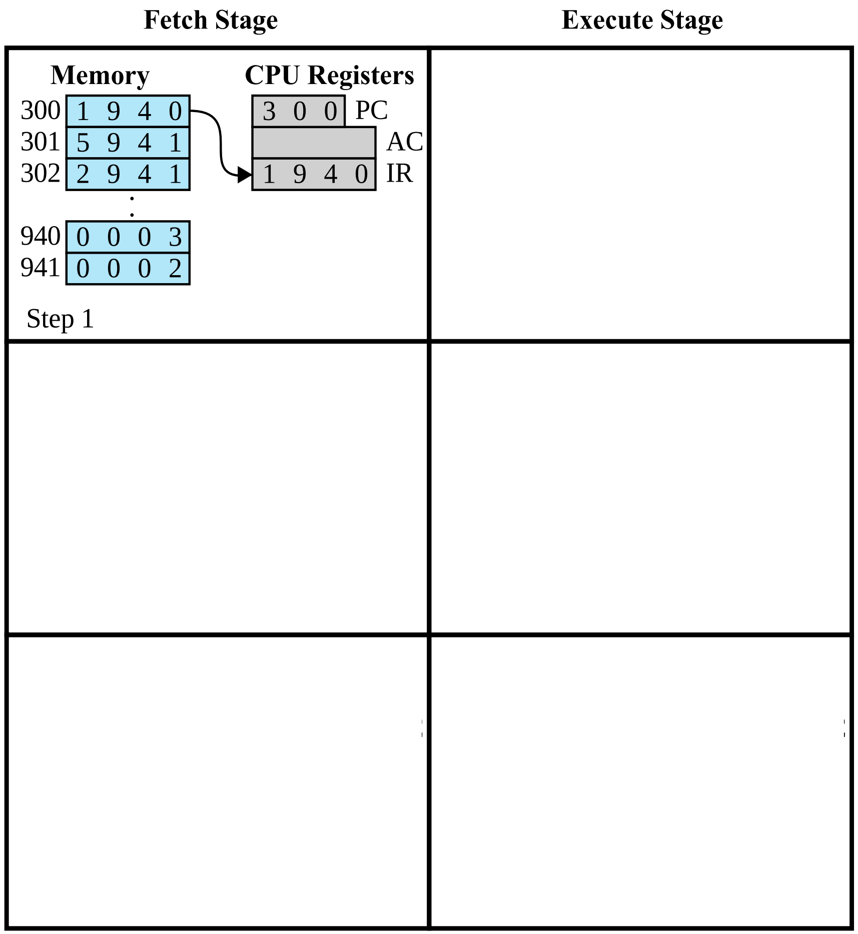
Program counter (PC) = Address of instruction
Instruction register (IR) = Instruction being executed
Accumulator (AC) = Temporary storage

(c) Internal CPU registers

0001 = Load AC from memory
0010 = Store AC to memory
0101 = Add to AC from memory

(d) Partial list of opcodes

Figure 1.3 Characteristics of a Hypothetical Machine



0001 = Load AC from memory
0010 = Store AC to memory
0101 = Add to AC from memory

Figure 1.4 Example of Program Execution (contents of memory and registers in hexadecimal)

0001 = Load AC from memory
0010 = Store AC to memory
0101 = Add to AC from memory

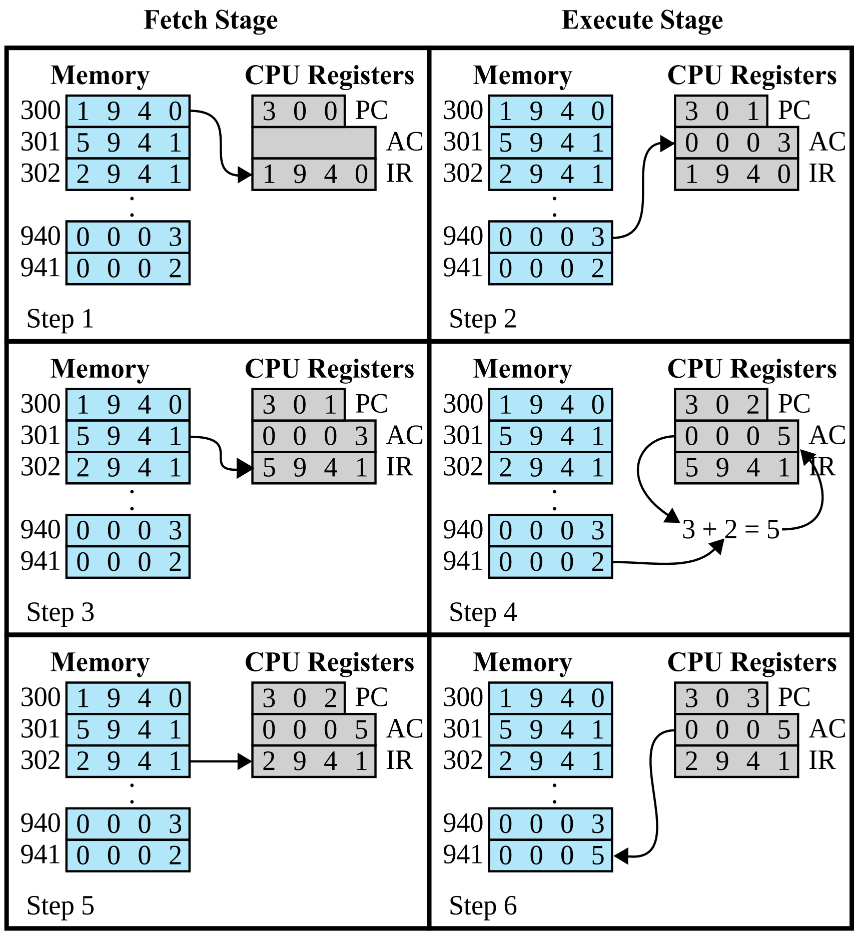


Figure 1.4 Example of Program Execution (contents of memory and registers in hexadecimal)

Interrupts

- Interrupt the normal sequencing of the processor
- Provided to improve processor utilization
    - most I/O devices are slower than the processor
    - processor must pause to wait for device
    - wasteful use of the processor

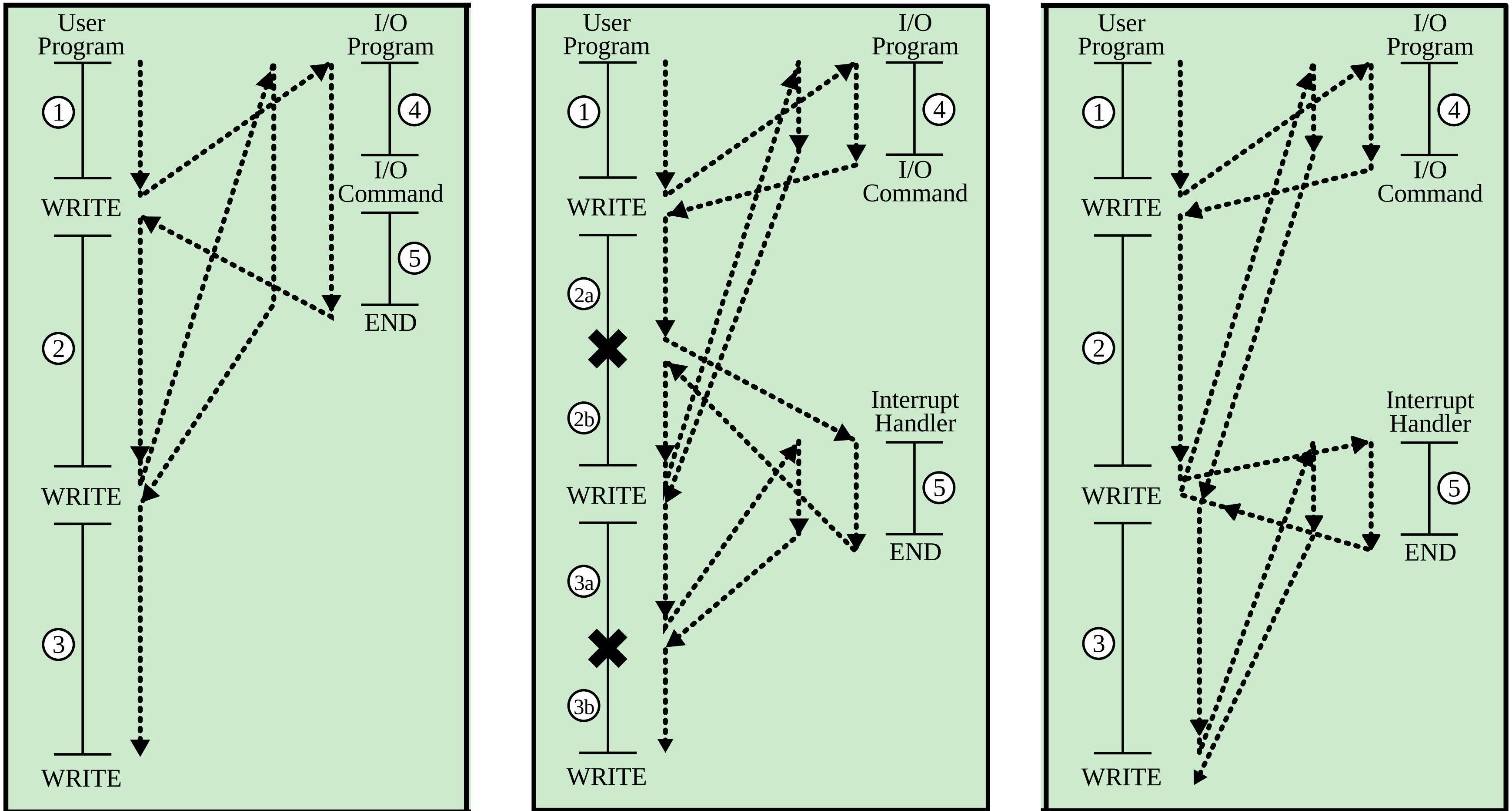
Classes of Interrupts

- **Program:** Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, and reference outside a user's allowed memory space.

- **Timer:** Generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis.

- **I/O:** Generated by an I/O controller, to signal normal completion of an operation or to signal a variety of error conditions.

- **Hardware failure:** Generated by a failure, such as power failure or memory parity error.



(a) No interrupts

(b) Interrupts; short I/O wait

(c) Interrupts; long I/O wait

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

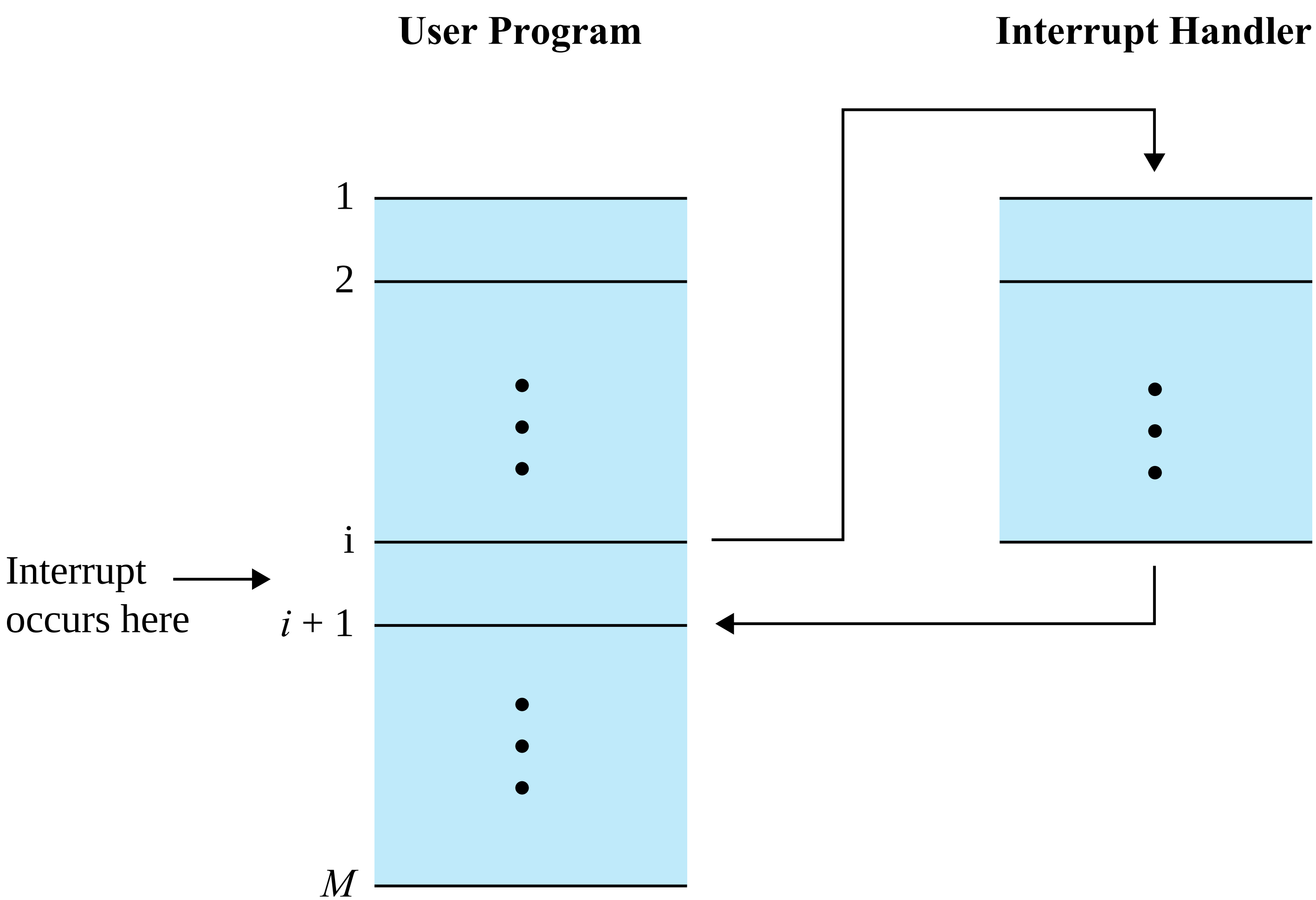


Figure 1.6 Transfer of Control via Interrupts

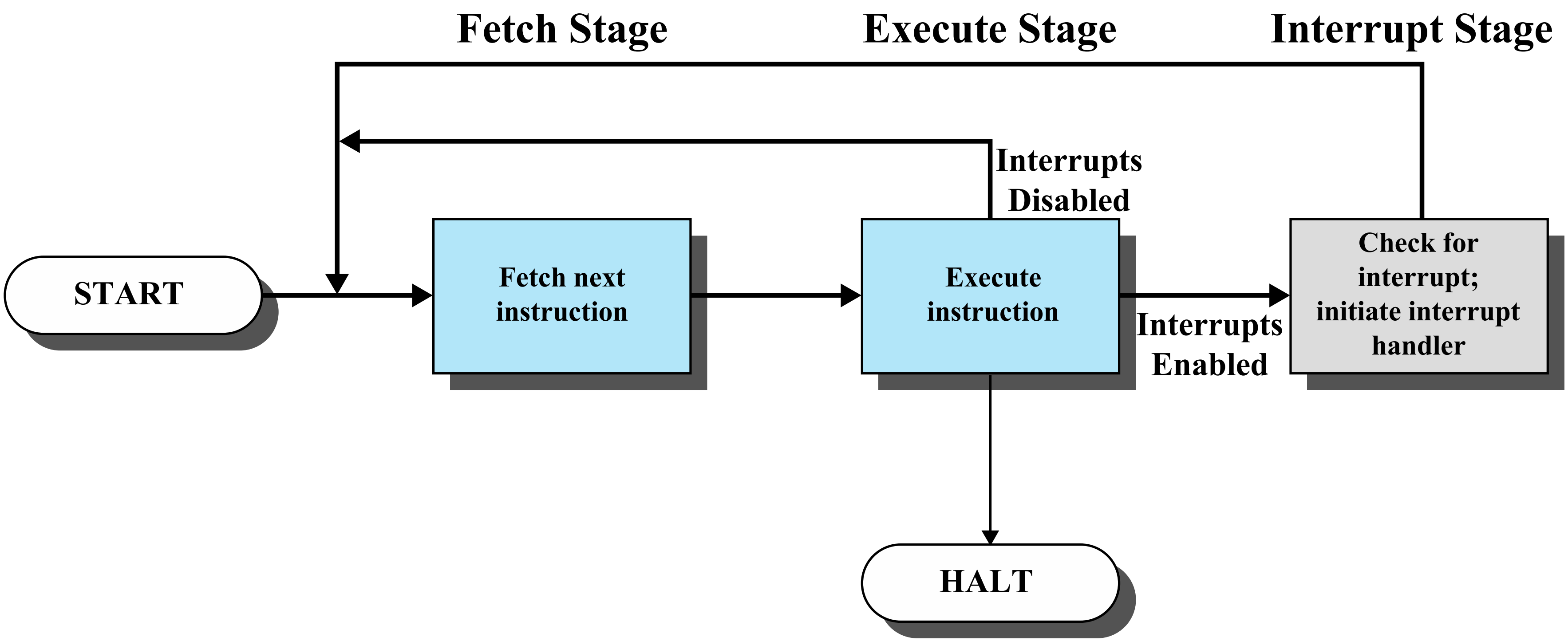


Figure 1.7 Instruction Cycle with Interrupts

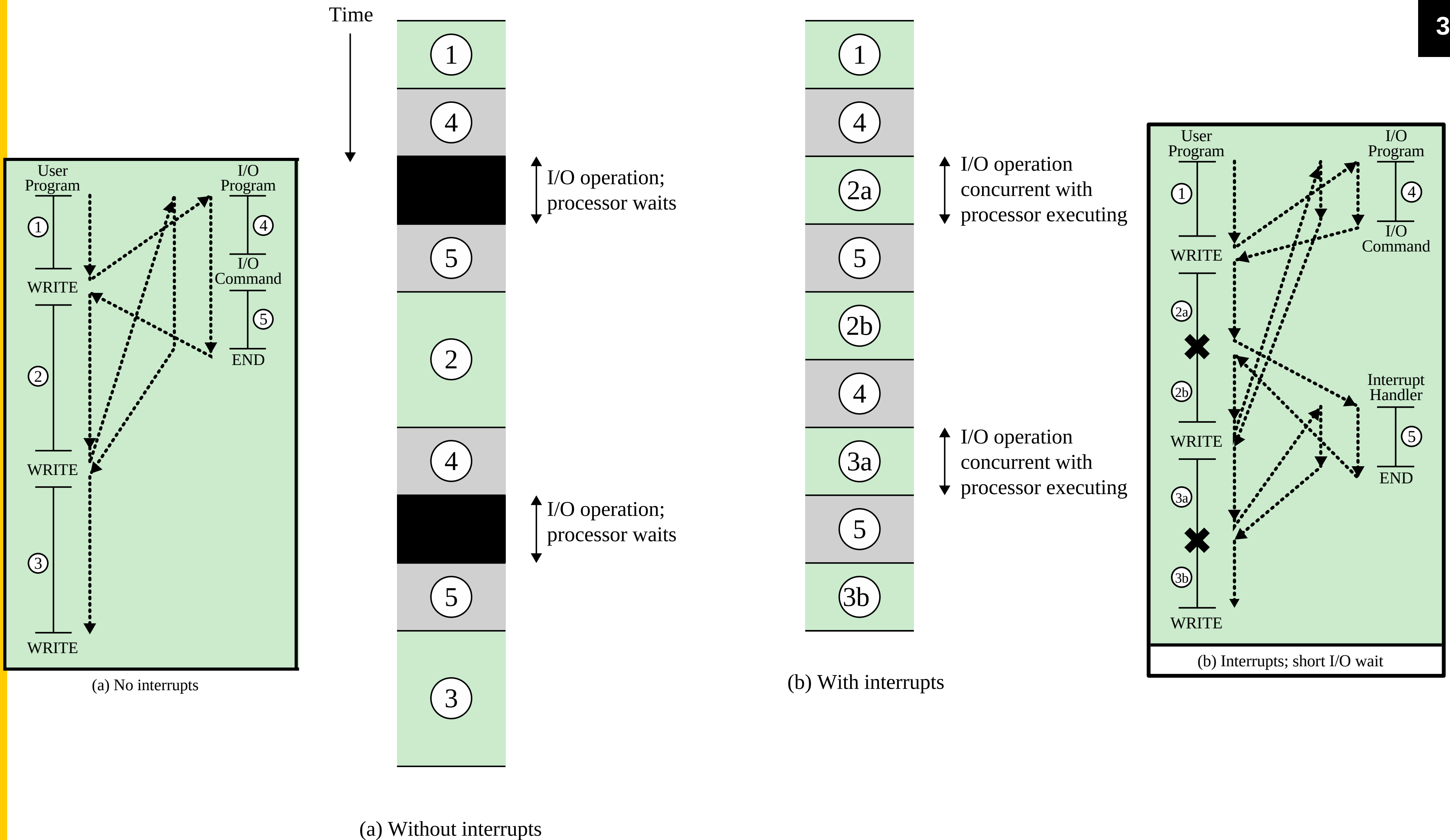


Figure 1.8 Program Timing: Short I/O Wait

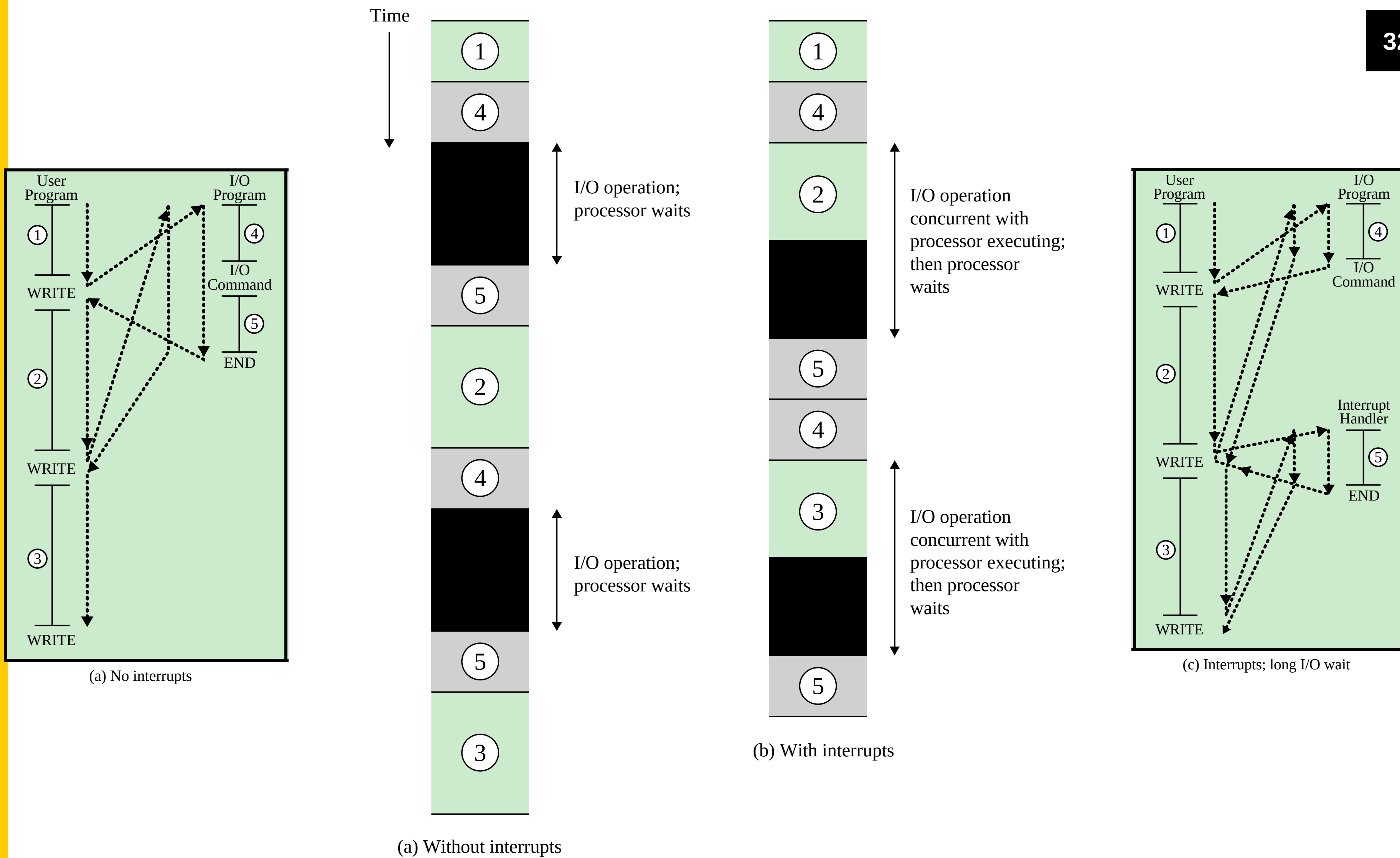


Figure 1.9 Program Timing: Long I/O Wait

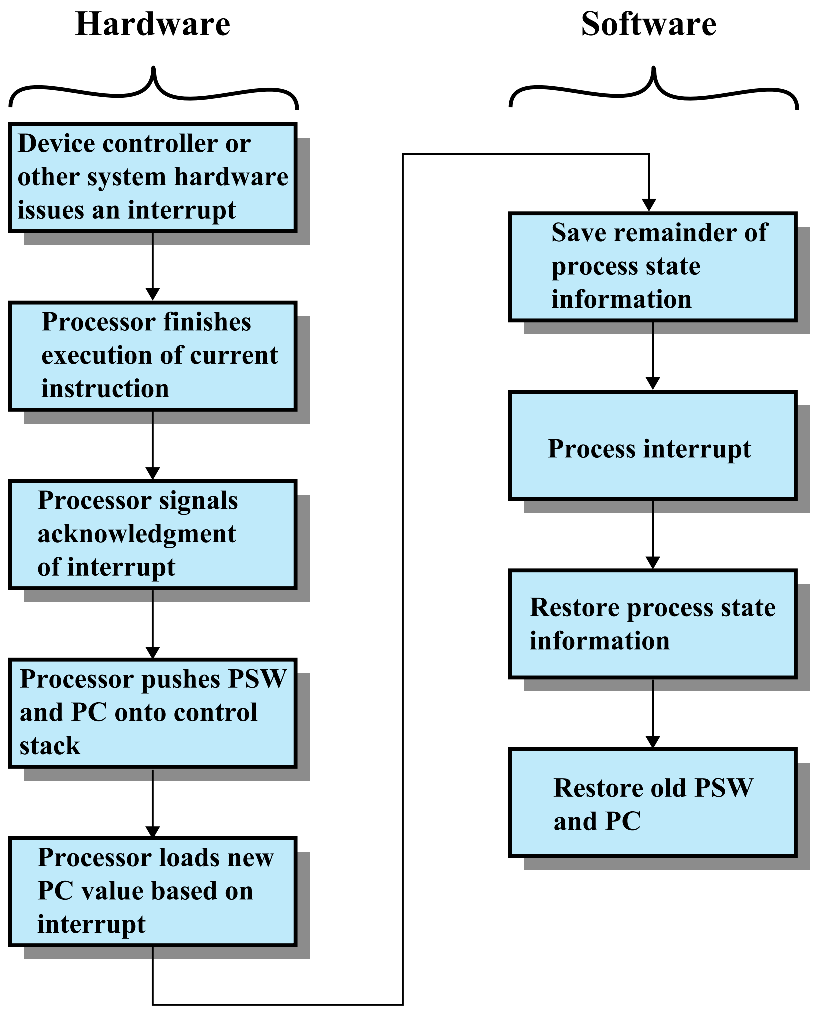


Figure 1.10 Simple Interrupt Processing

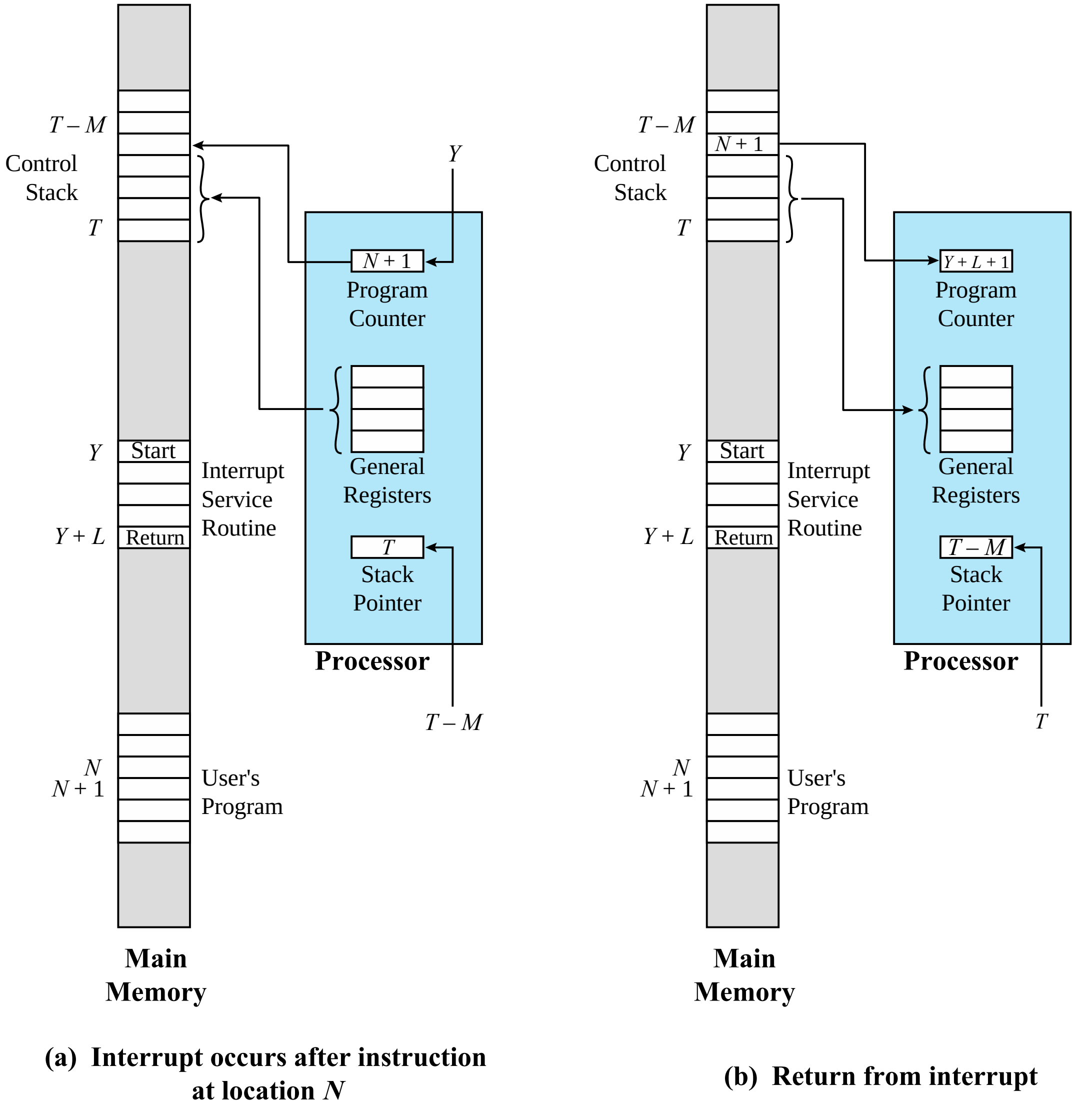


Figure 1.11 Changes in Memory and Registers for an Interrupt

Multiple Interrupts

- An interrupt occurs while another interrupt is being processed
  - e.g. receiving data from a communications line and printing results at the same time

- Two Approaches:
  - disable interrupts while an interrupt is being processed
  - use a priority scheme

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

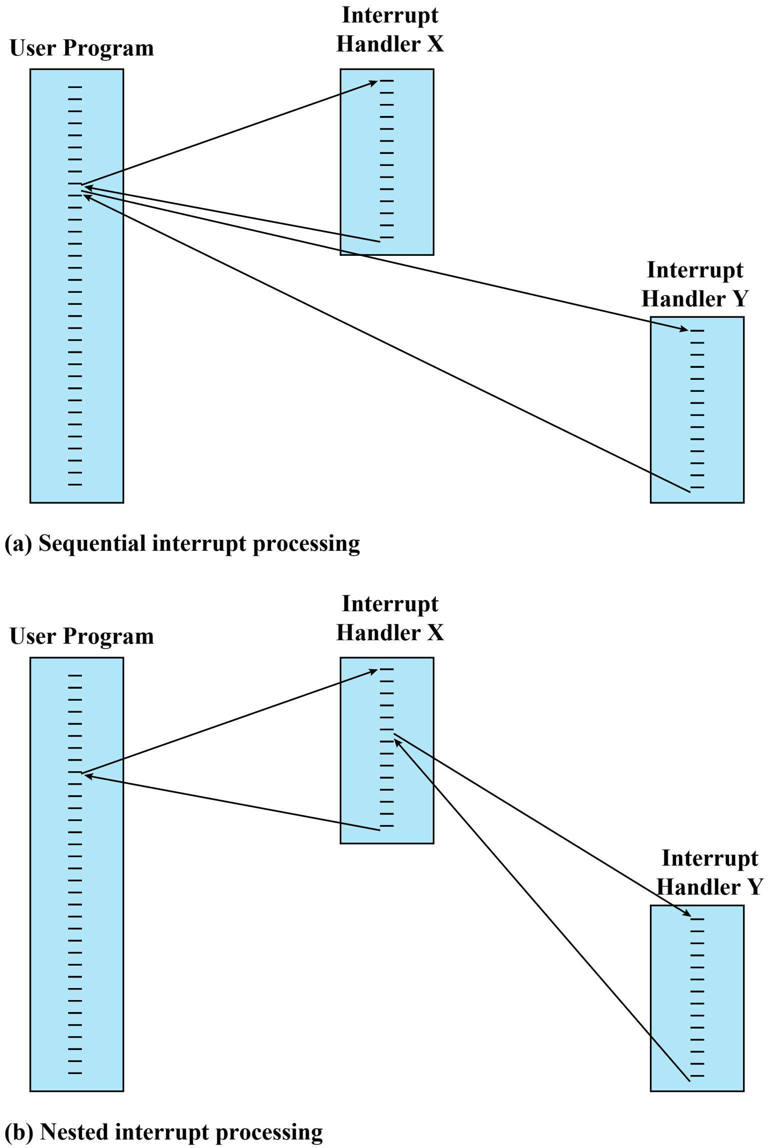


Figure 1.12 Transfer of Control with Multiple Interrupts

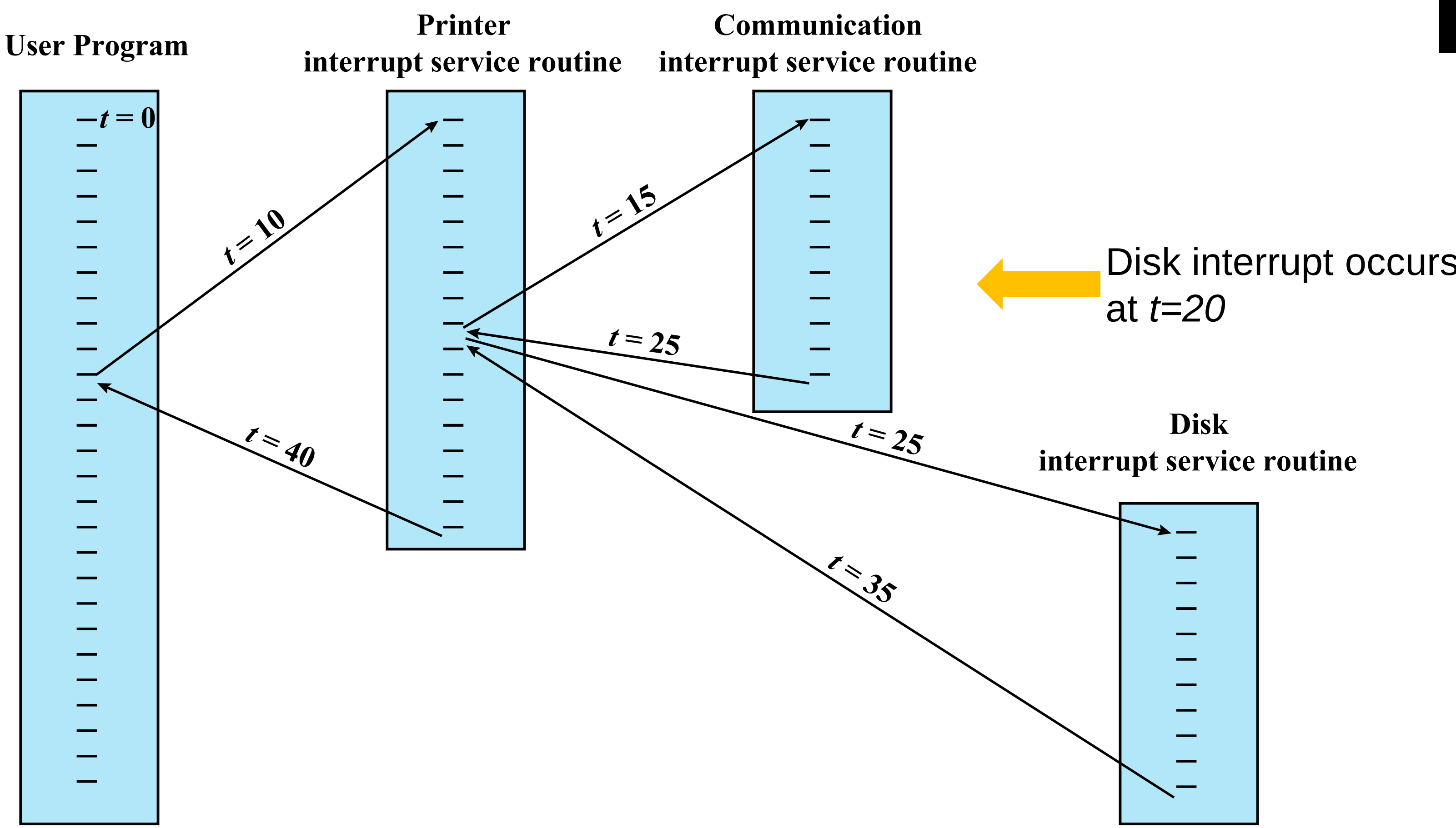


Figure 1.13 Example Time Sequence of Multiple Interrupts

Summary

- OS is a program that controls the execution of application programs and acts as an interface between application program and hardware.
- Instruction is executed in two phases: fetch and execute
- Interrupt is mechanism primarily to improve the processor utilization

References

- **Operating Systems – Internal and Design Principles**
  - By William Stallings
- Chapter 1