




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 Synchronisation
 - ☐ 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

- ❑ No workshop on week 1
- ❑ Regular workshop from week 2-12.

19:00 – 21:00 Monday (RW203)

16:00 – 18:00 Tuesday (HPE202)

11:00 -13:00 Thursday (HA142)

12:00 -14:00 Thursday (HE28)

Week 01 Lecture Outline

Operating System Overview

- ☐ What is an OS?
- ☐ OS role/objectives
- ☐ Hardware Review
- ☐ Instruction Execution
- ☐ Interrupts
- ☐ Memory Hierarchy
- ☐ Cache Memory
- ☐ I/O Techniques
- ☐ Multiprocessors and Multicores

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

6

- 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

7

- 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

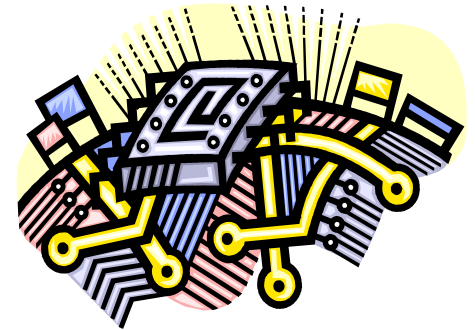
8



- 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

11

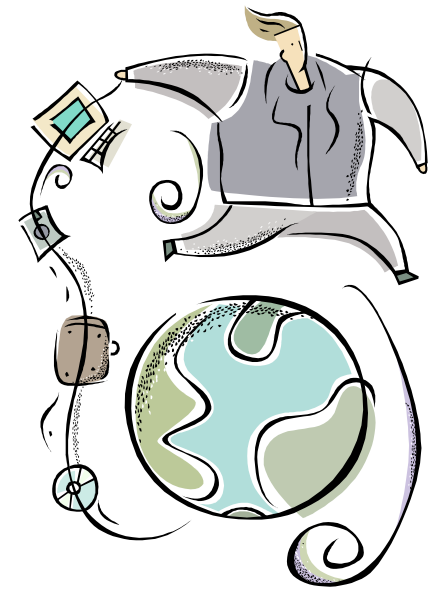
- A major OS will evolve over time for a number of reasons:

hardware upgrades

new types of hardware

new services

Fixes



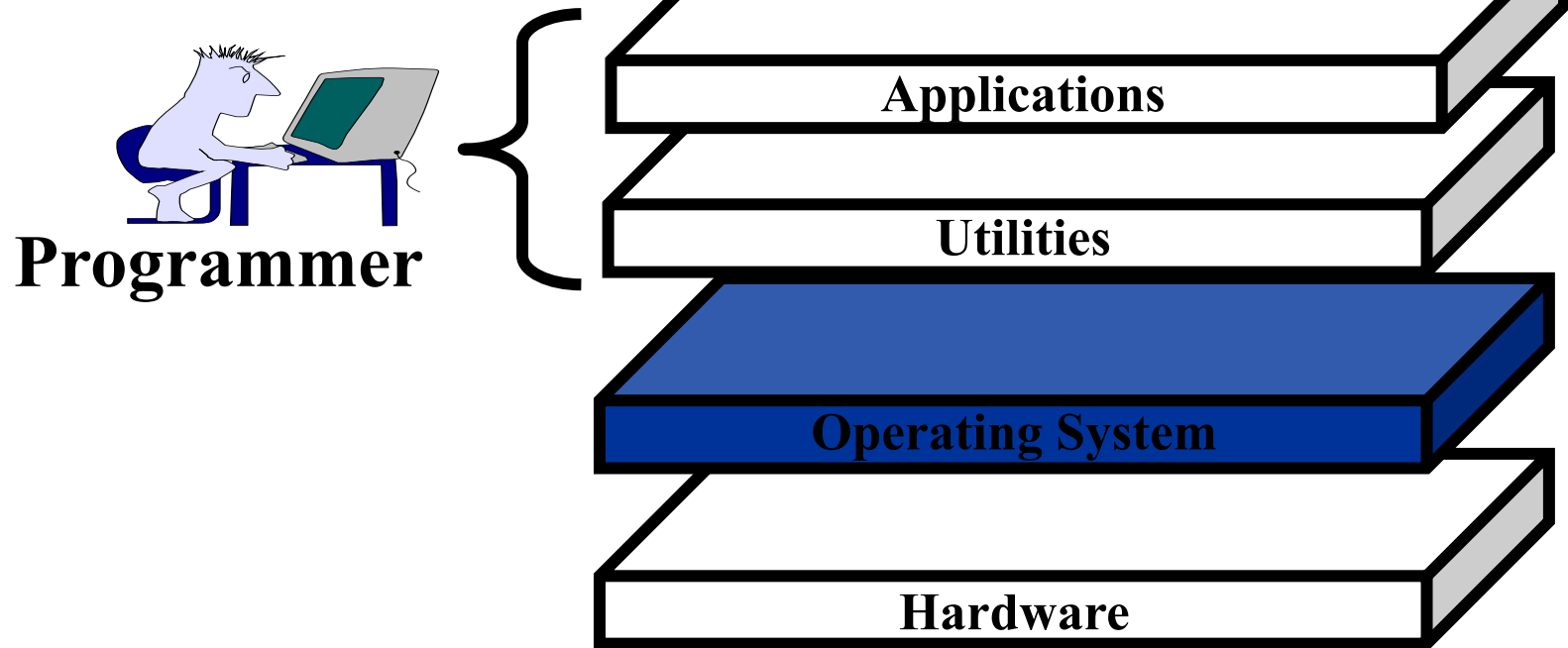
What the OS does ...

- 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?

User

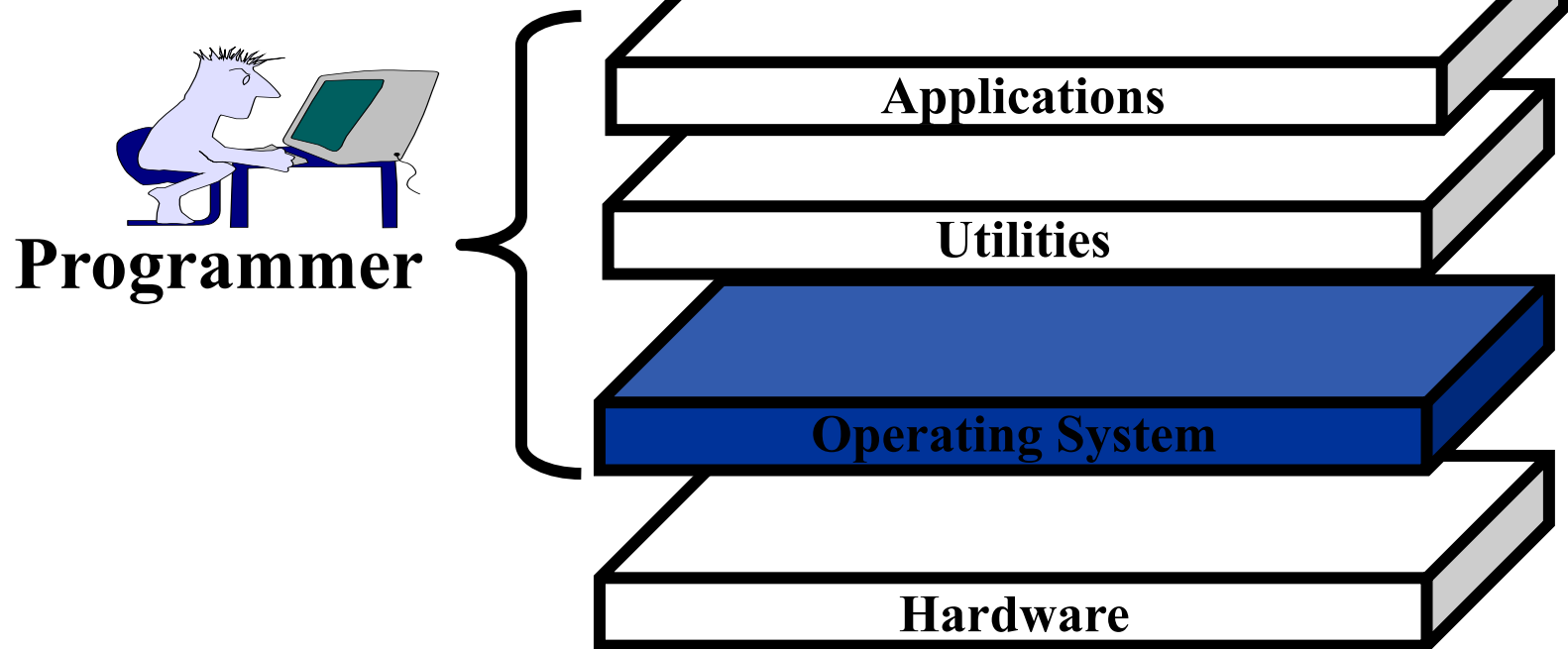
13



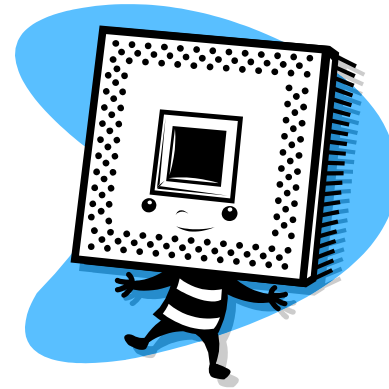
Where is the OS?

User

14



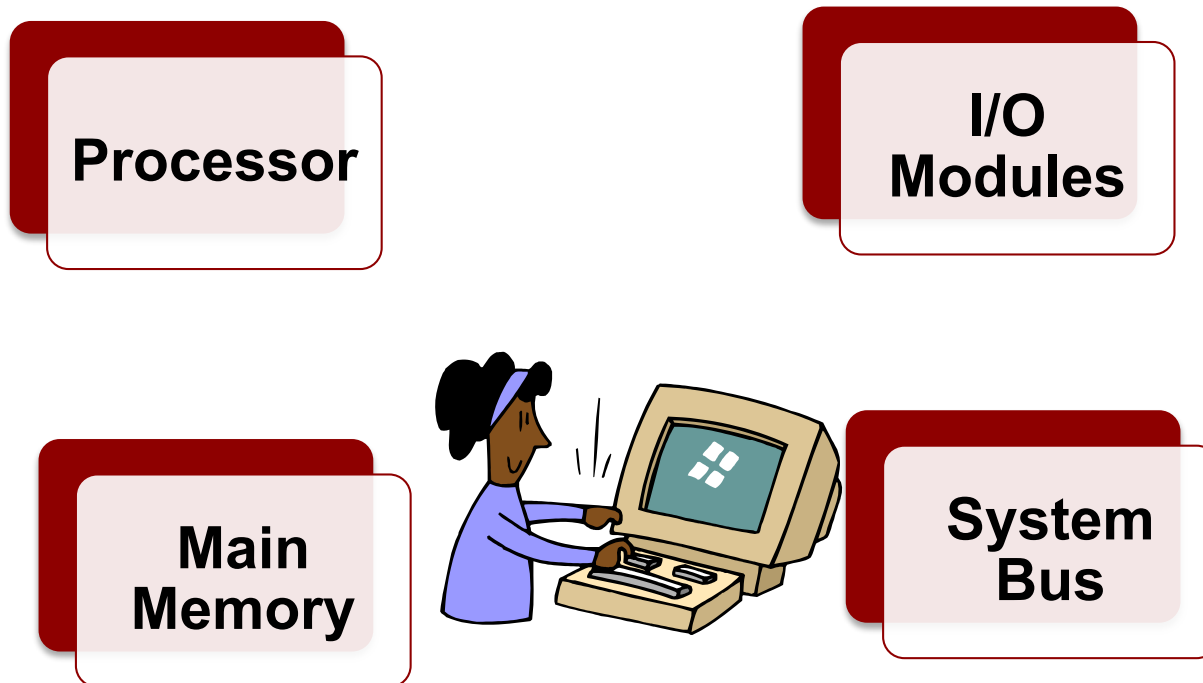
Hardware Review

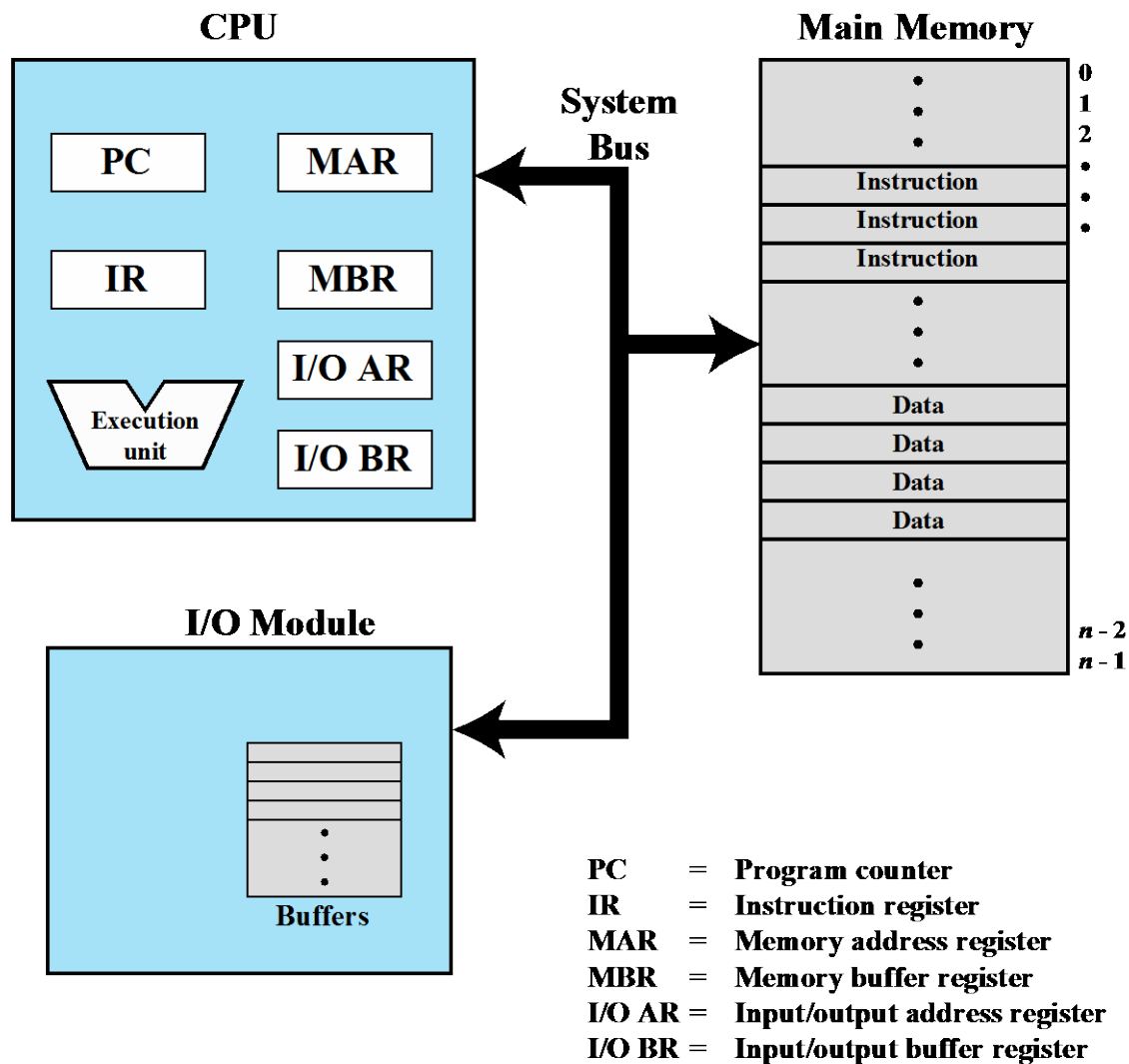


- Operating systems
 - Exploit the hardware available
 - Provide a set of high-level services that represent or are implemented by the hardware.
 - Manages the hardware reliably and efficiently
- *Understanding operating systems requires a basic understanding of the underlying hardware*

Basic Elements

16





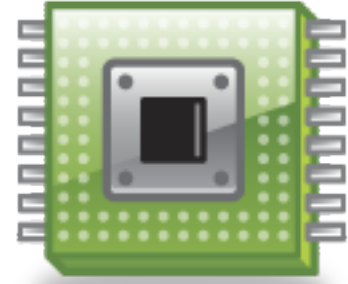
Microprocessor



18

- 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



19

- 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

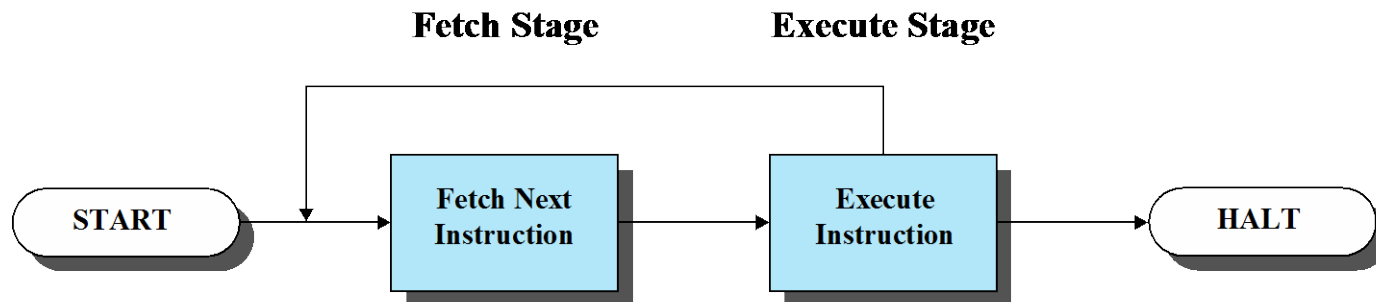
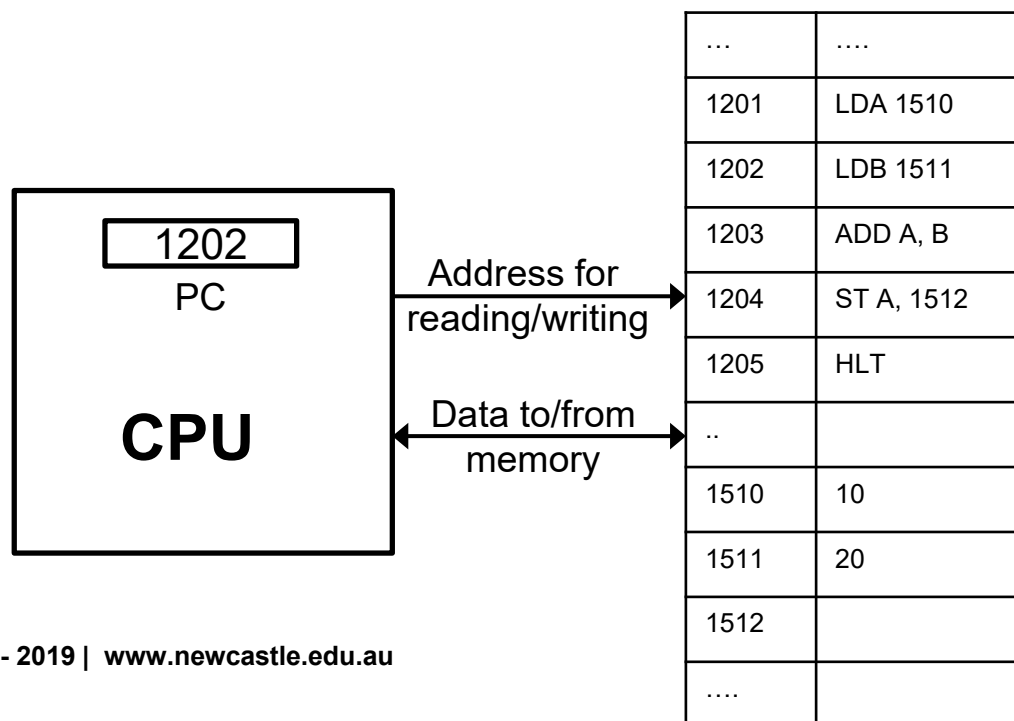
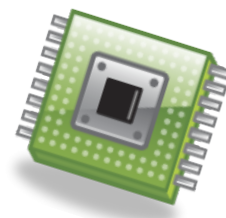


Figure 1.2 Basic Instruction Cycle

...
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



Instruction Register (IR)

Fetches instruction is loaded into Instruction Register (IR)

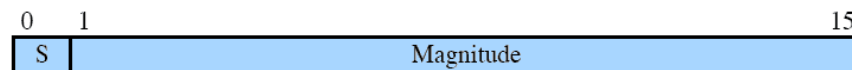


- Processor interprets the instruction and performs required action:
 - Processor-memory
 - Processor-I/O
 - Data processing
 - Control





(a) Instruction format



(b) Integer format

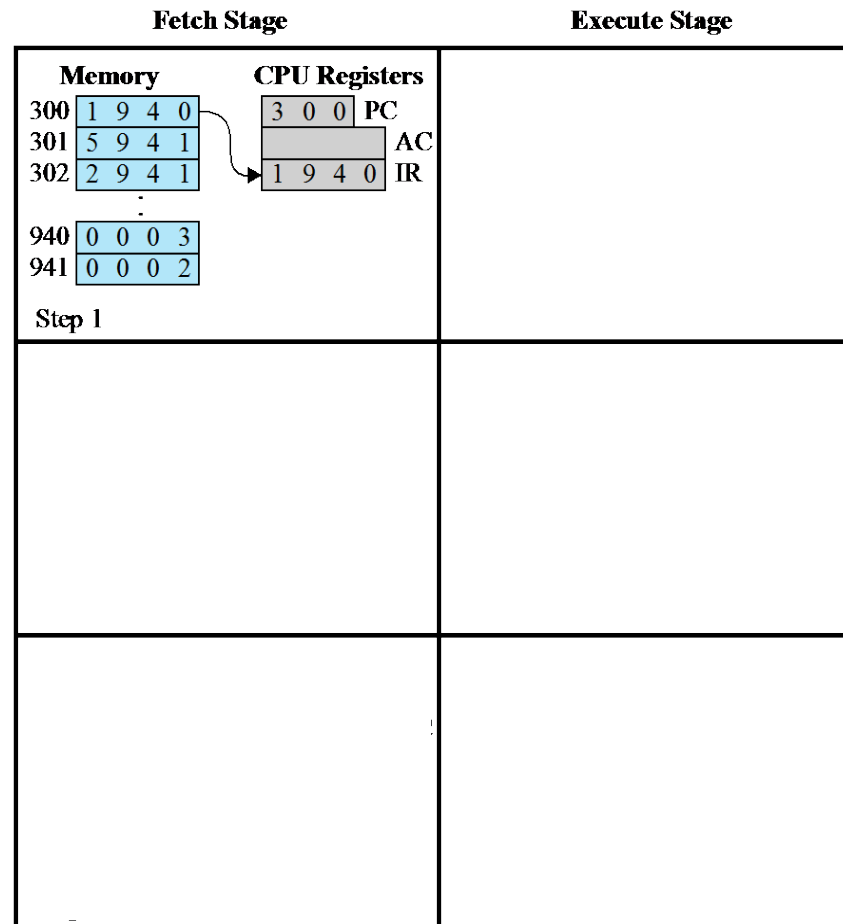
Program counter (PC) = Address of instruction
 Instruction register (IR) = Instruction being executed
 Accumulator (AC) = Temporary storage

(c) Internal CPU registers

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

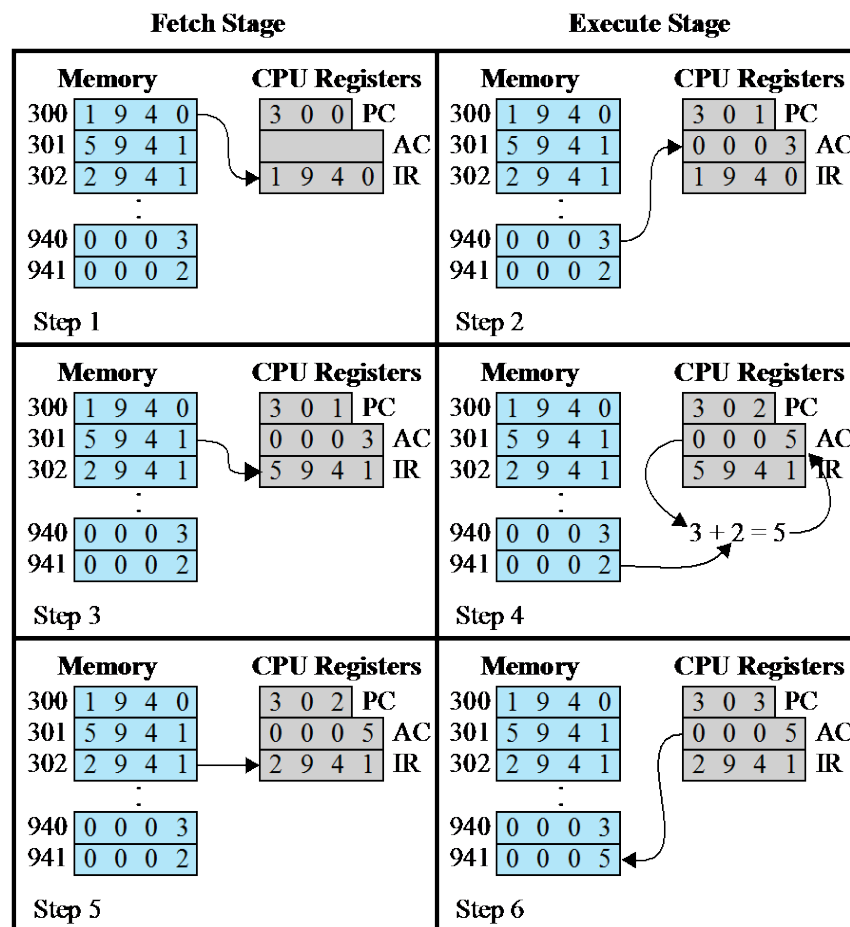
(d) Partial list of opcodes

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)

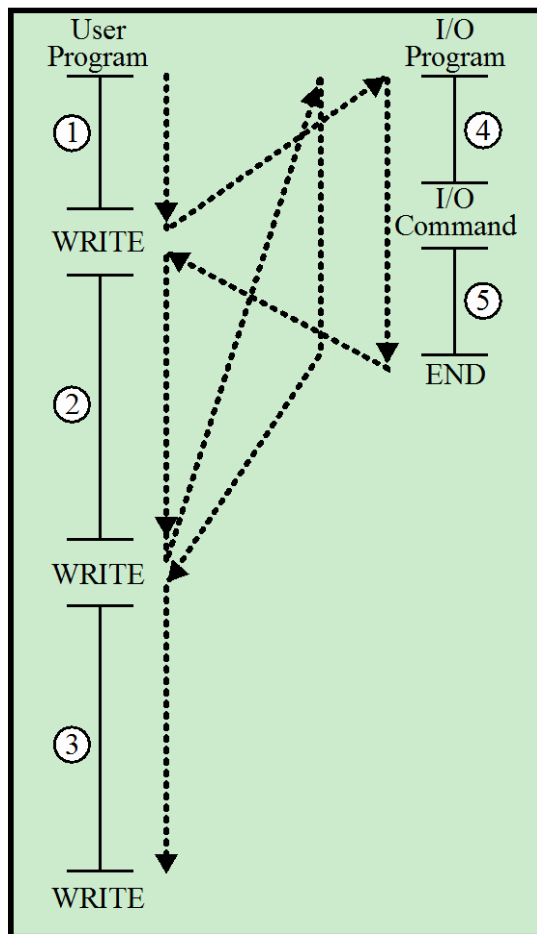
Interrupts

- 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

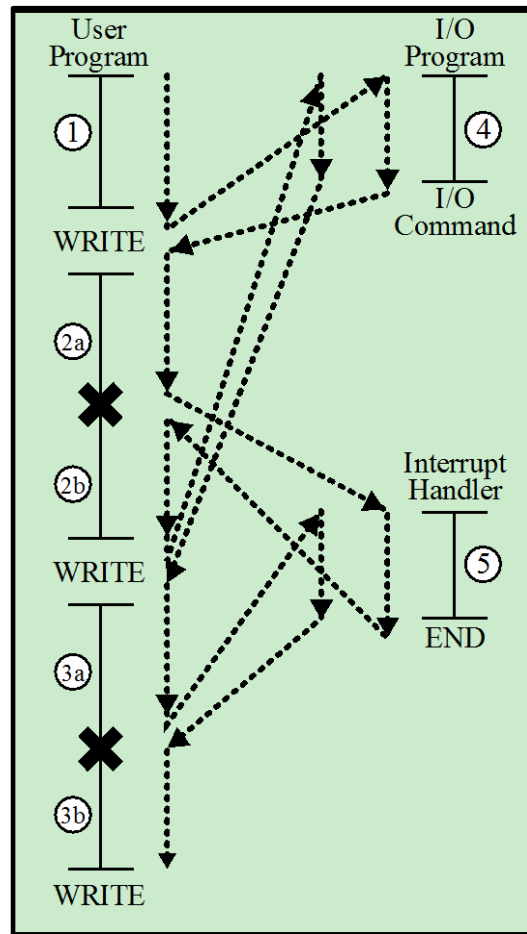


Classes of Interrupts

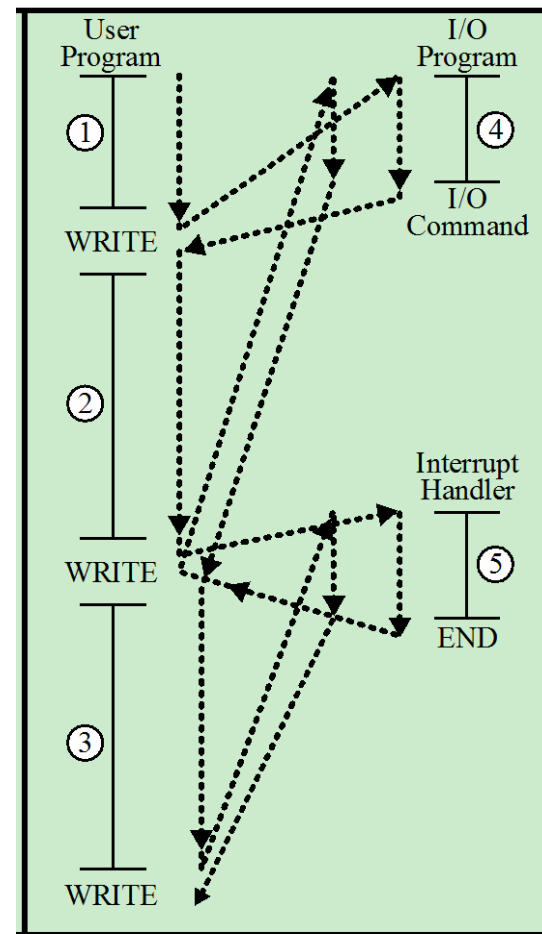
- **Program:** Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, and reference outside a user's allowed memory space.
- **Timer:** Generated by a timer within the processor.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



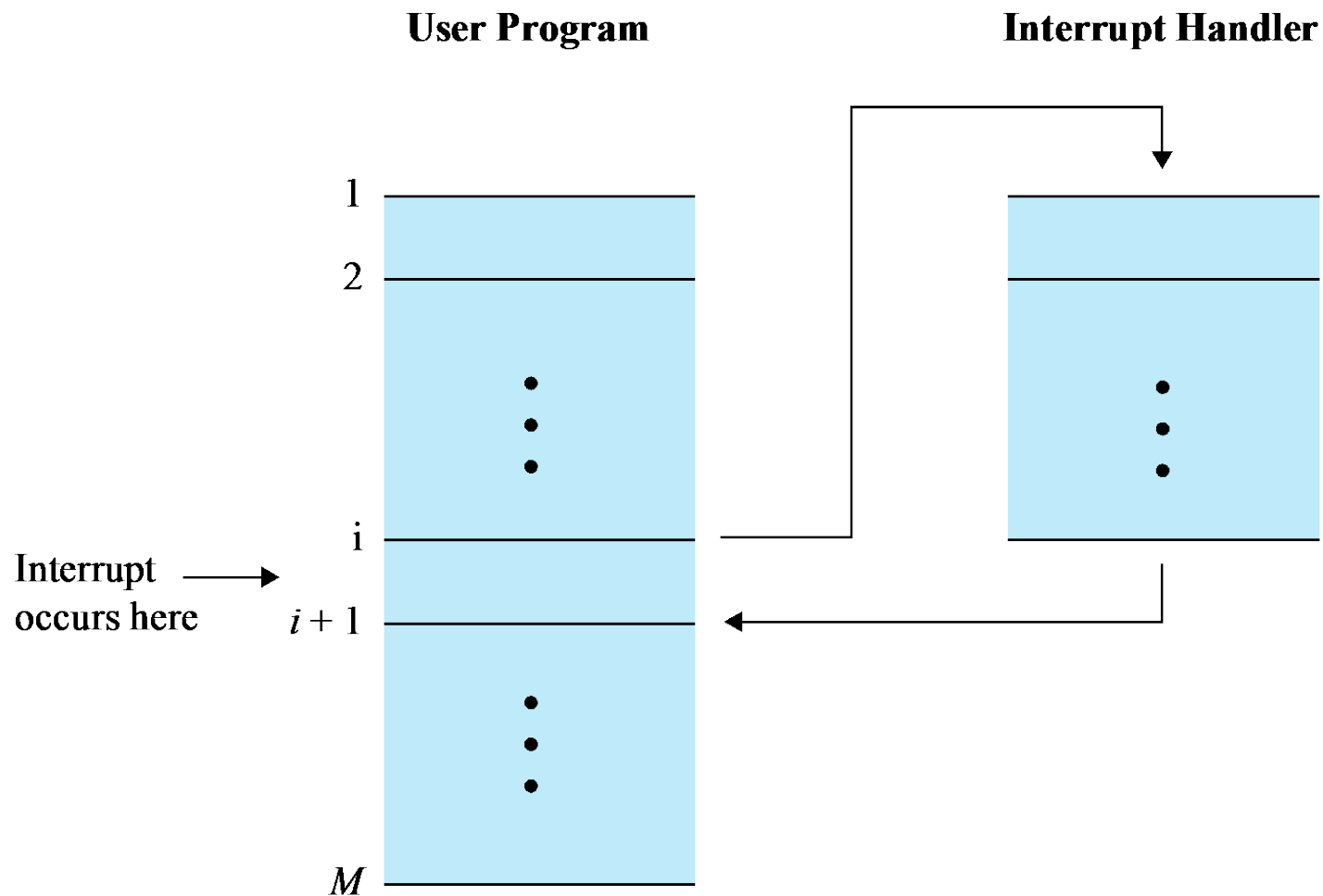
(a) No interrupts



(b) Interrupts; short I/O wait



(c) Interrupts; long I/O wait



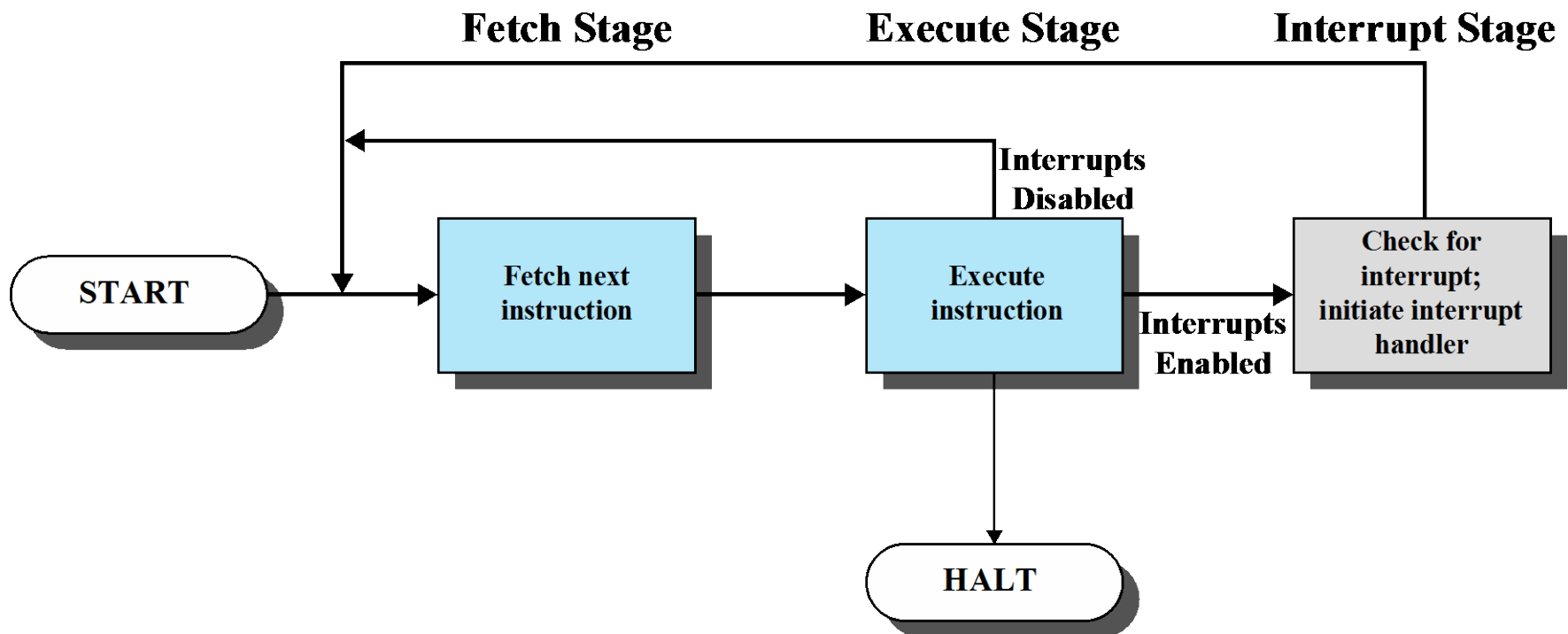
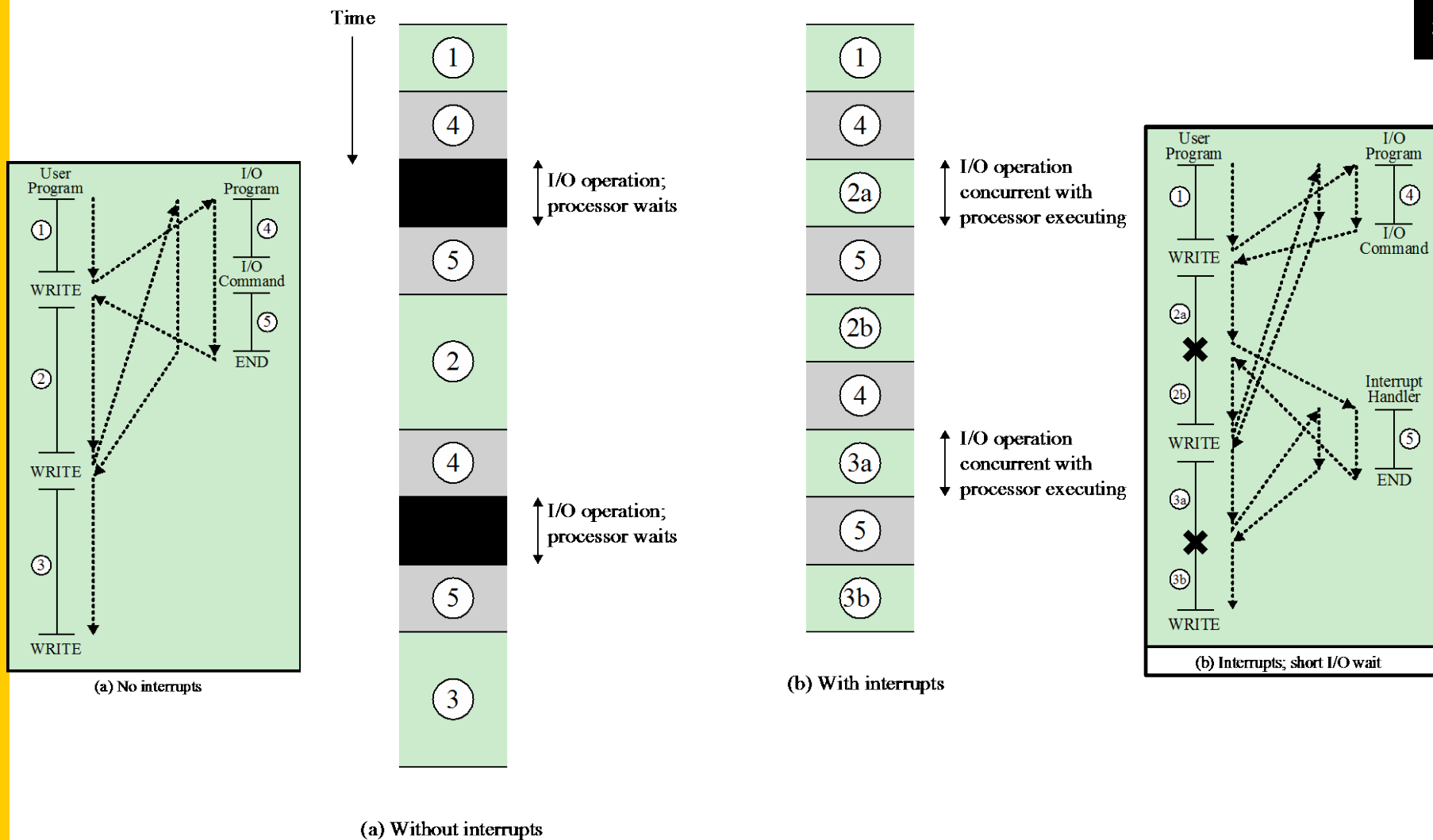
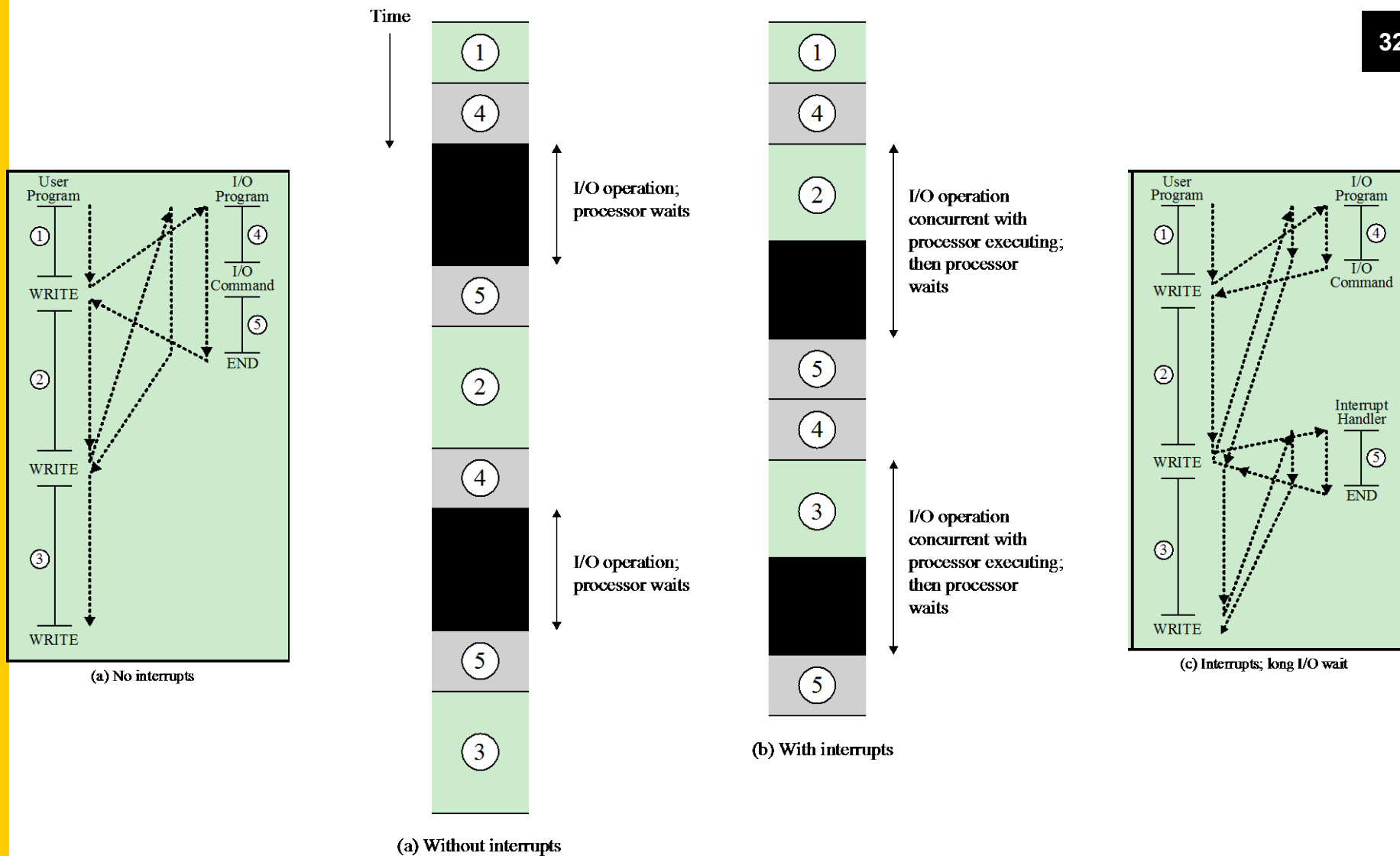


Figure 1.7 Instruction Cycle with Interrupts





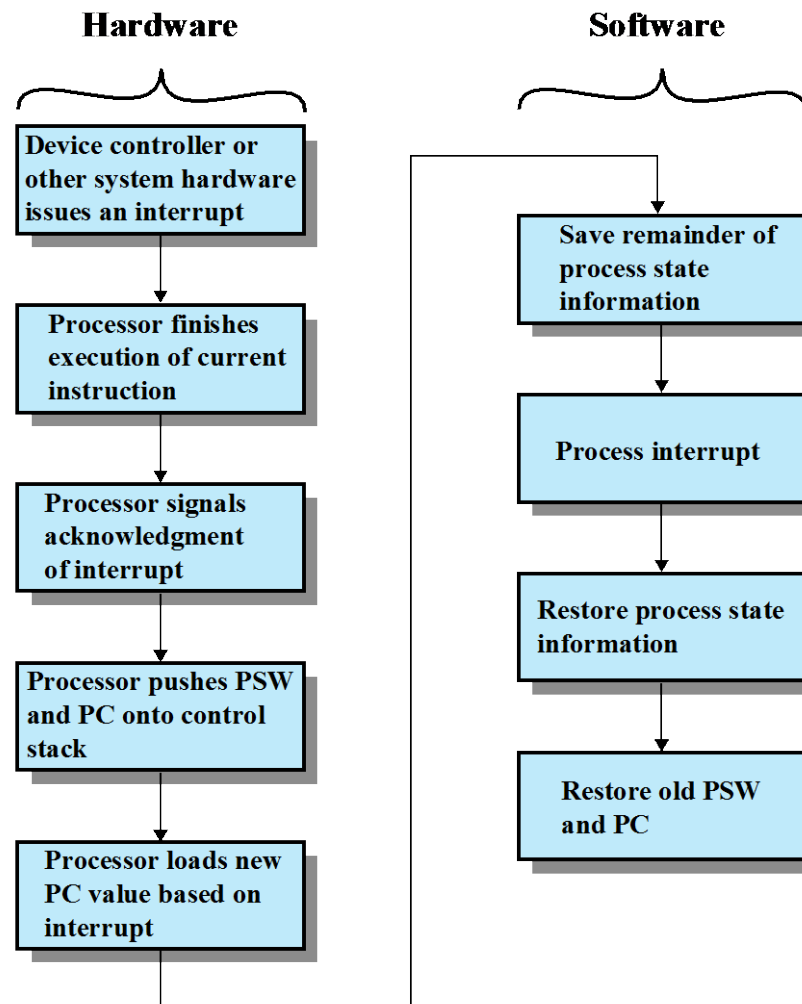
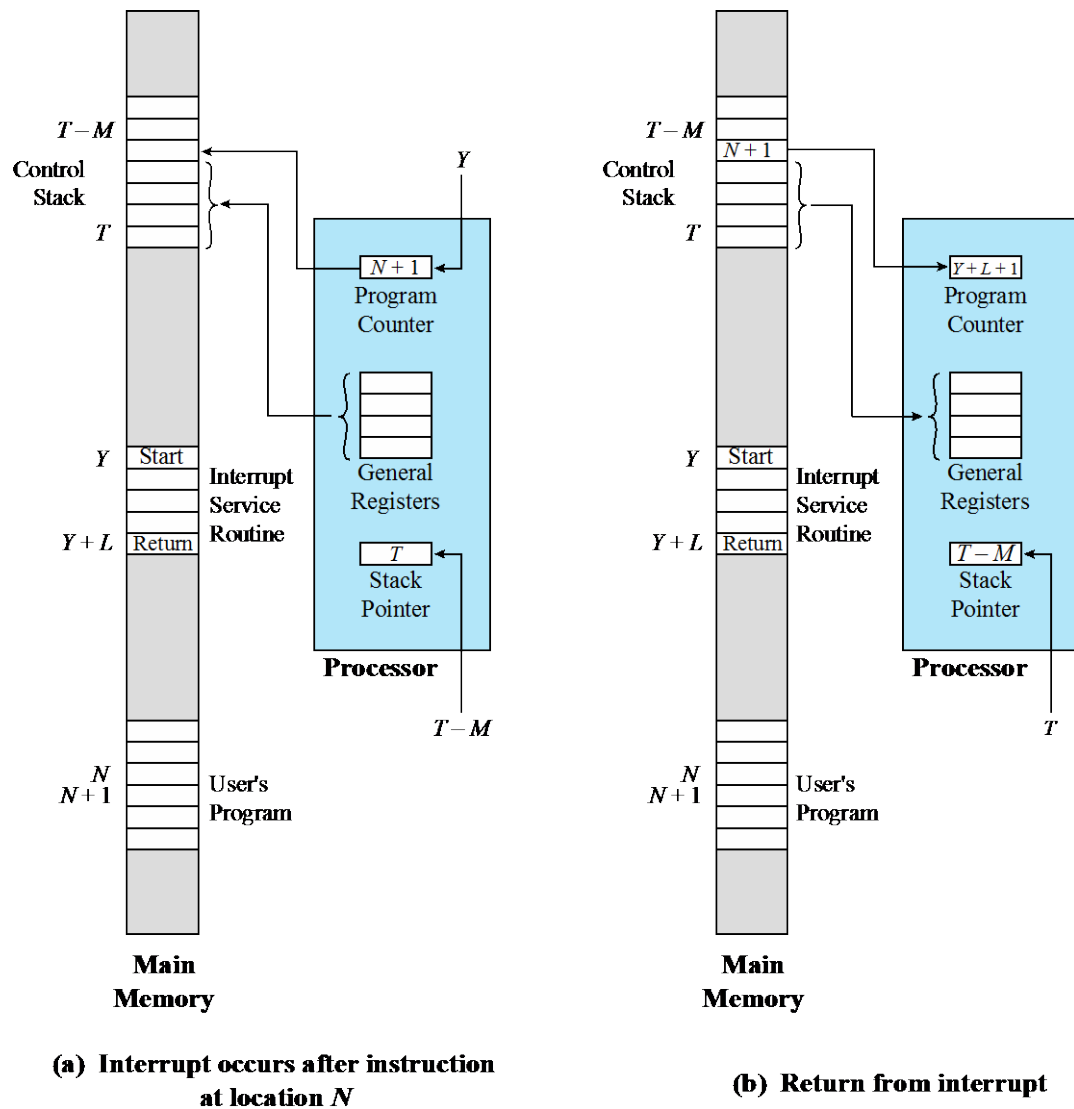
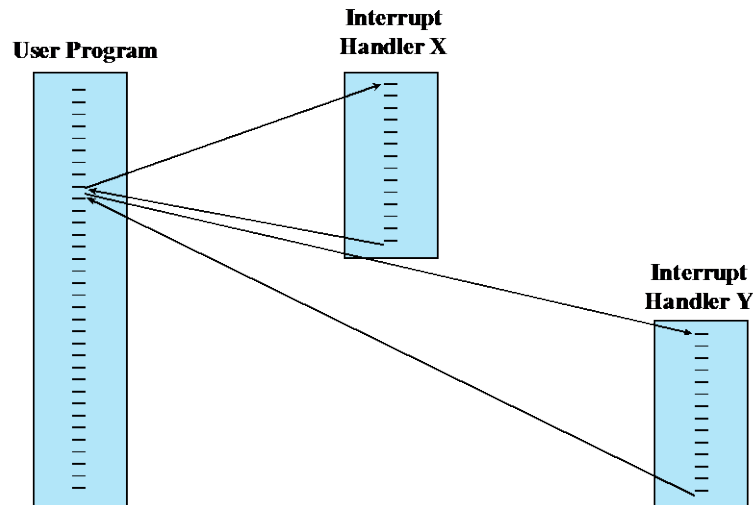


Figure 1.10 Simple Interrupt Processing

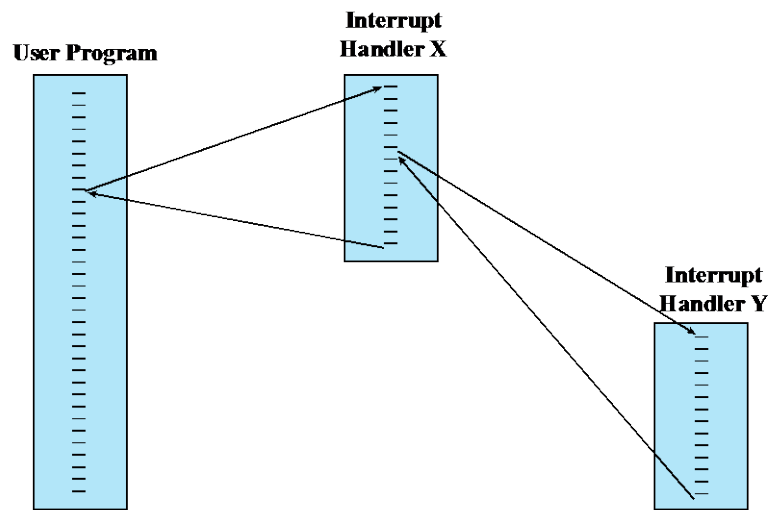


Multiple Interrupts

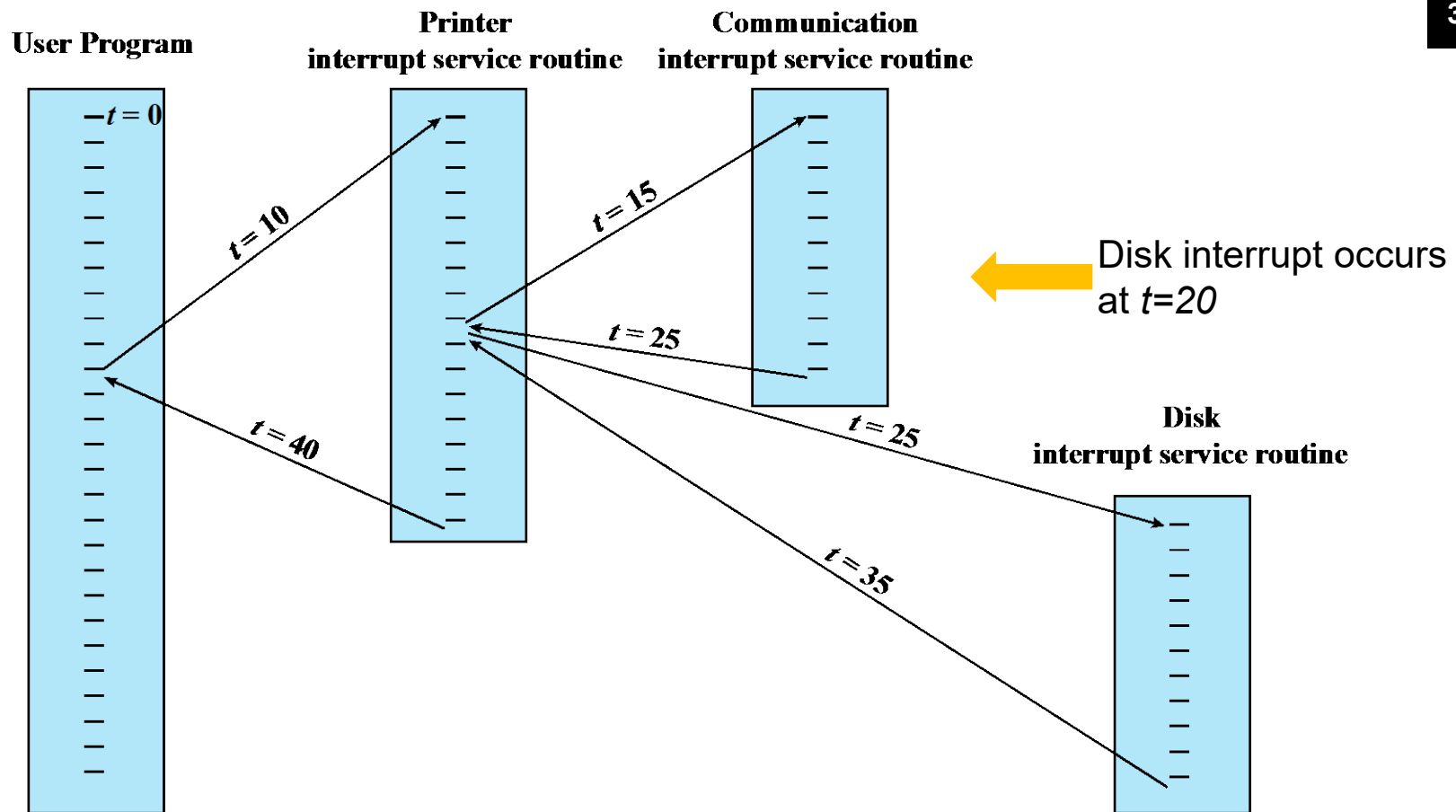
- An interrupt occurs while another interrupt is being processed
 - e.g. receiving data from a communications line and printing results at the same time
- Two Approaches:
 - disable interrupts while an interrupt is being processed
 - use a priority scheme



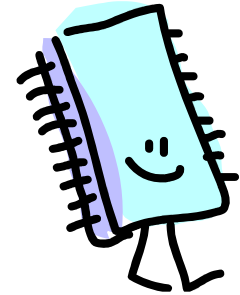
(a) Sequential interrupt processing



(b) Nested interrupt processing



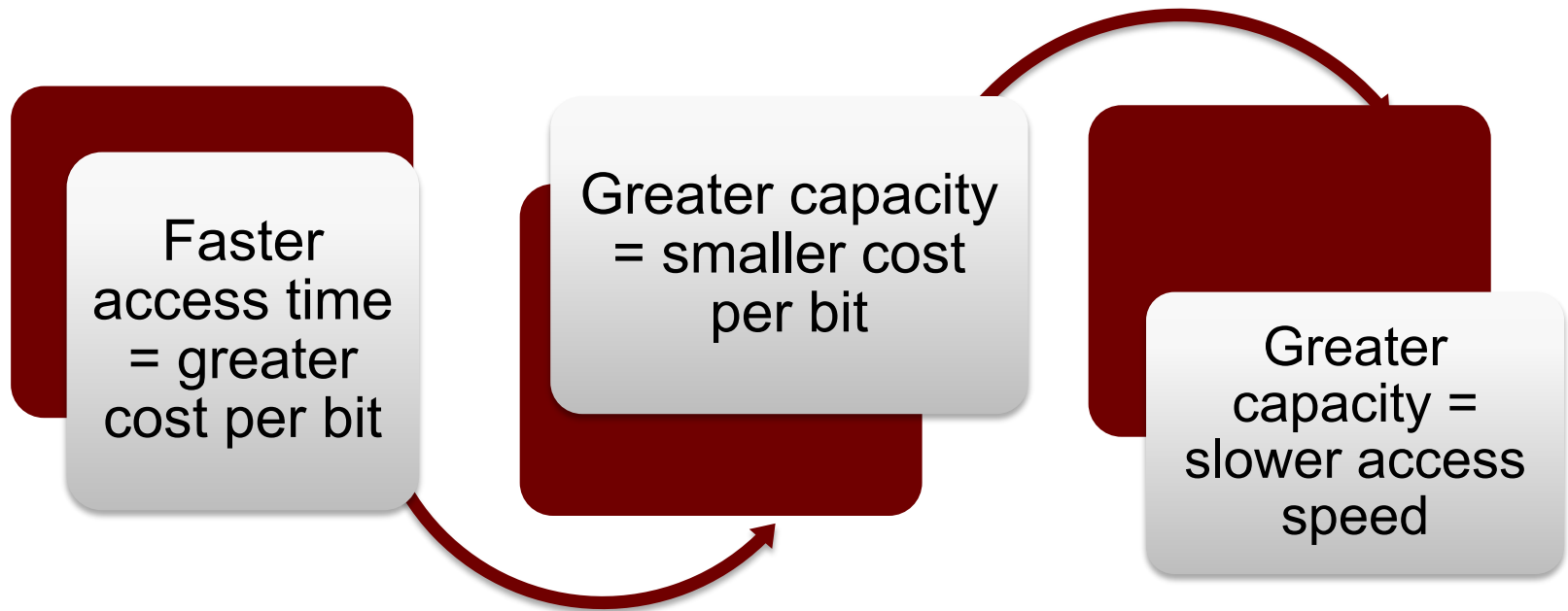
Memory Hierarchy



- Major constraints in memory
 - amount
 - speed
 - expense
- Memory must be able to keep up with the processor
- Cost of memory must be reasonable in relationship to the other components

Memory Relationships

39



The Memory Hierarchy

- Going down the hierarchy:
 - decreasing cost per bit
 - increasing capacity
 - increasing access time
 - decreasing frequency of access to the memory by the processor

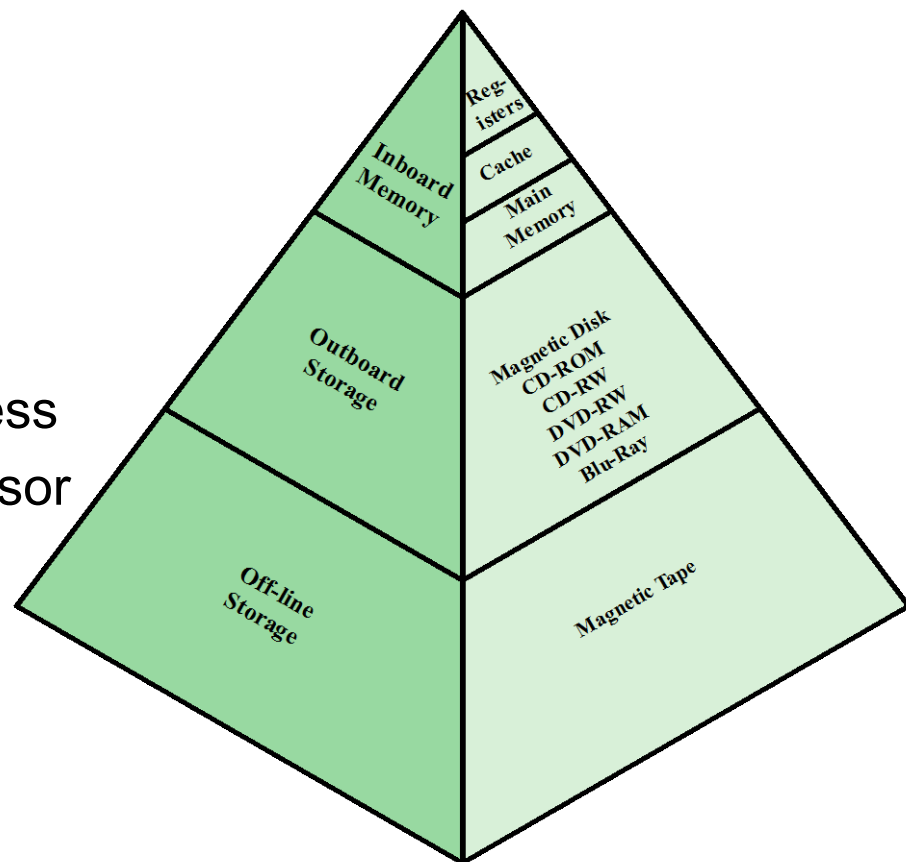


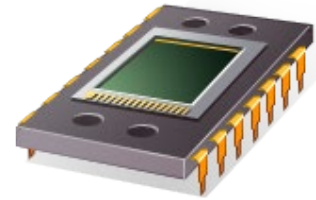
Figure 1.14 The Memory Hierarchy

Secondary Memory

- Also referred to as auxiliary memory
 - external
 - nonvolatile
 - used to store program and data files



Cache Memory



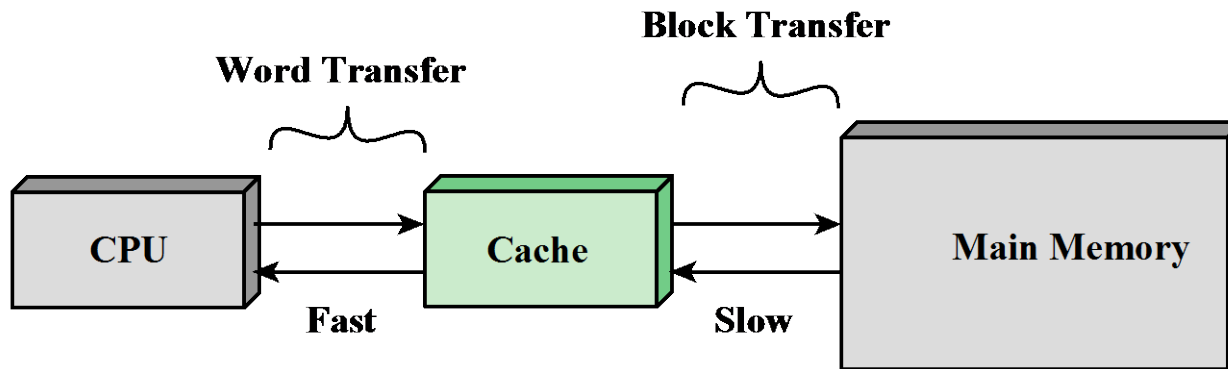
42

- Invisible to the OS
- Interacts with other memory management hardware
- Processor must access memory at least once per instruction cycle
- Processor execution is limited by memory cycle time
- Exploit the principle of locality with a small, fast memory

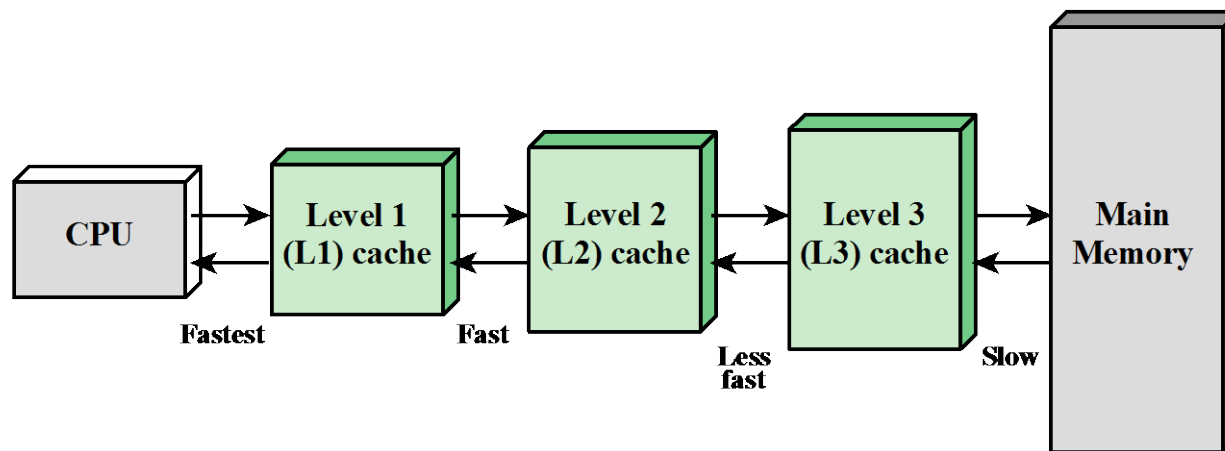
Principle of Locality

43

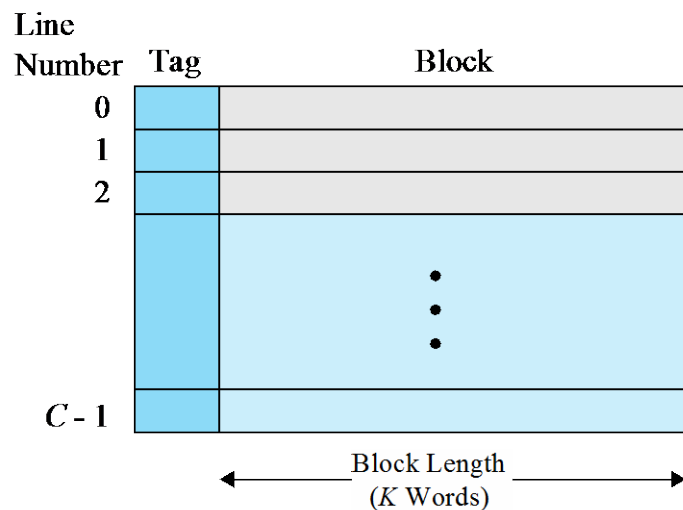
- Memory references by the processor tend to cluster
- Data is organized so that the percentage of accesses to each successively lower level is substantially less than that of the level above
- Can be applied across more than two levels of memory



(a) Single cache



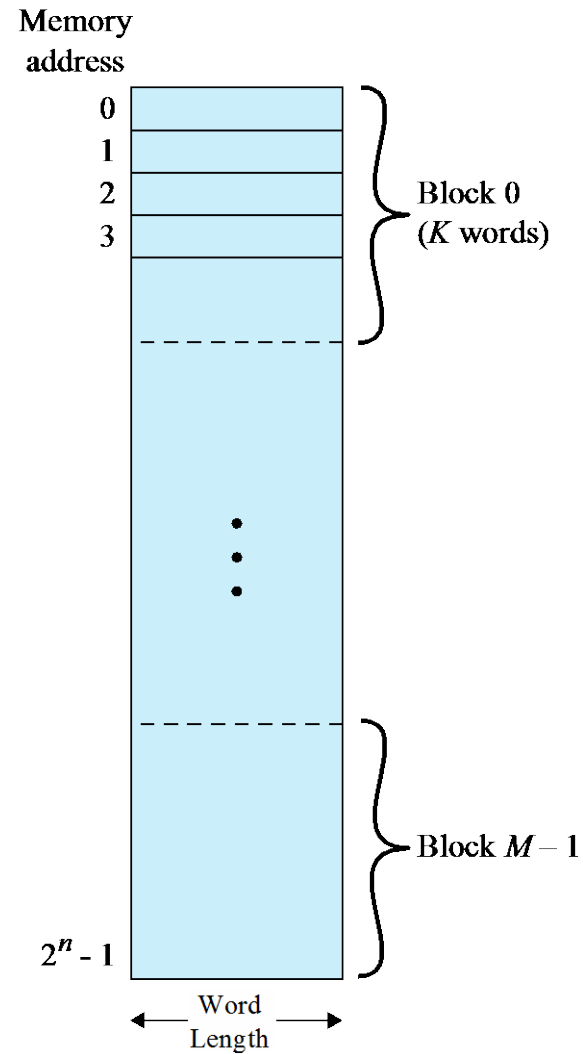
(b) Three-level cache organization



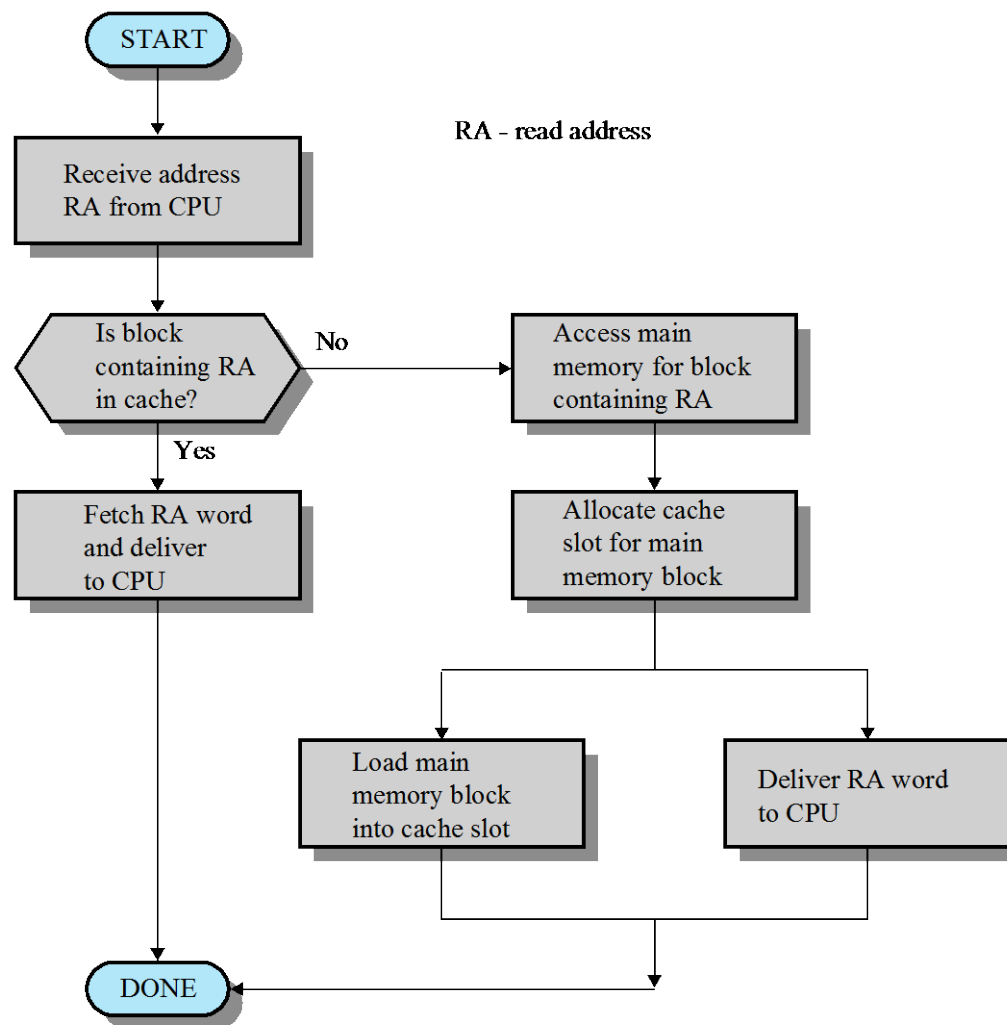
(a) Cache

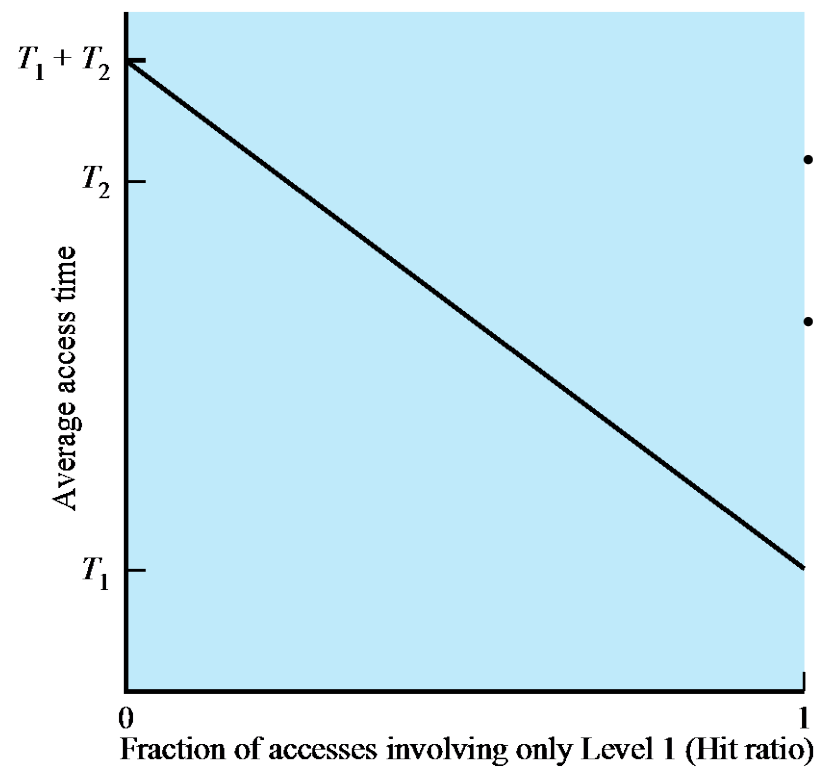
- C: Number of slots
- M: Number of blocks

$$C \ll M$$



(b) Main memory





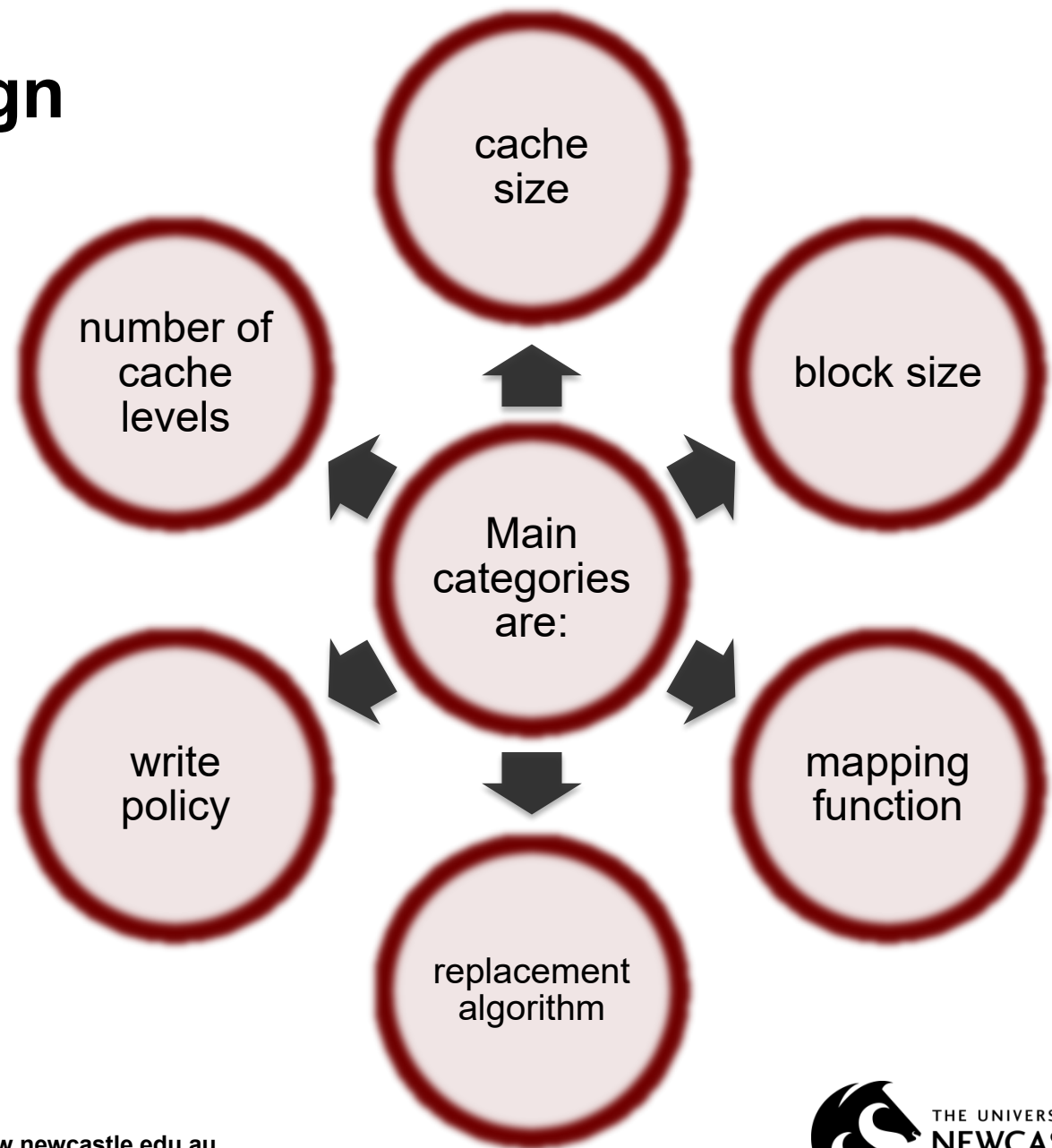
- Level 1
 - 1,000 bytes
 - Access time of $T_1 = 0.1 \mu\text{s}$
- Level 2
 - 100,000 bytes
 - Access time of $T_2 = 1 \mu\text{s}$

If $H = 0.95$, the average time to access a byte is

$$(0.95)(0.1 \mu\text{s}) + (0.05)(0.1 \mu\text{s} + 1 \mu\text{s}) = 0.095 + 0.055 = 0.15 \mu\text{s}$$

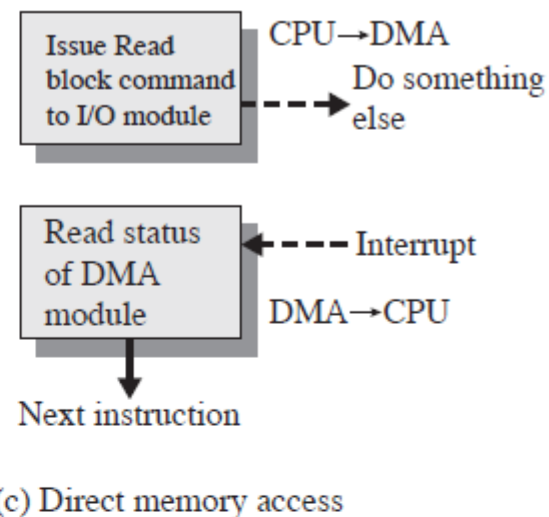
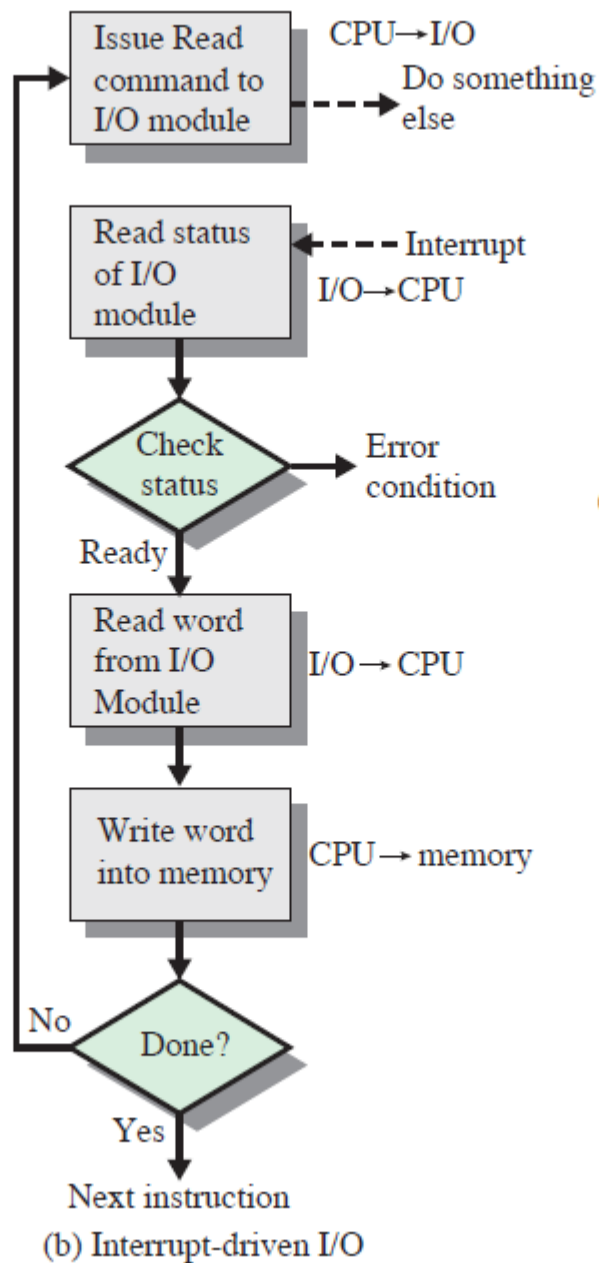
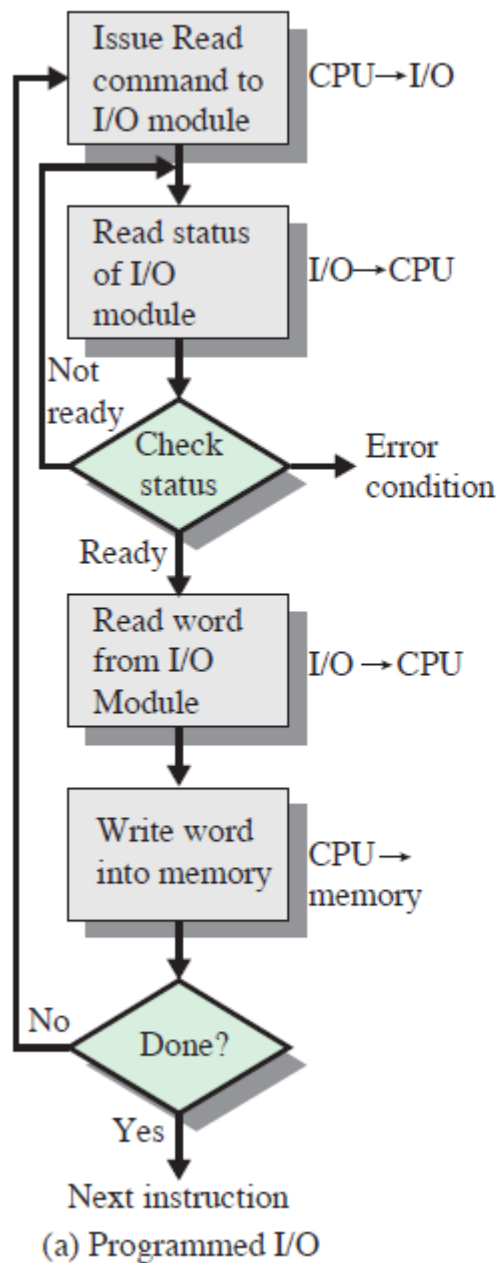
Cache Design

48



I/O Techniques

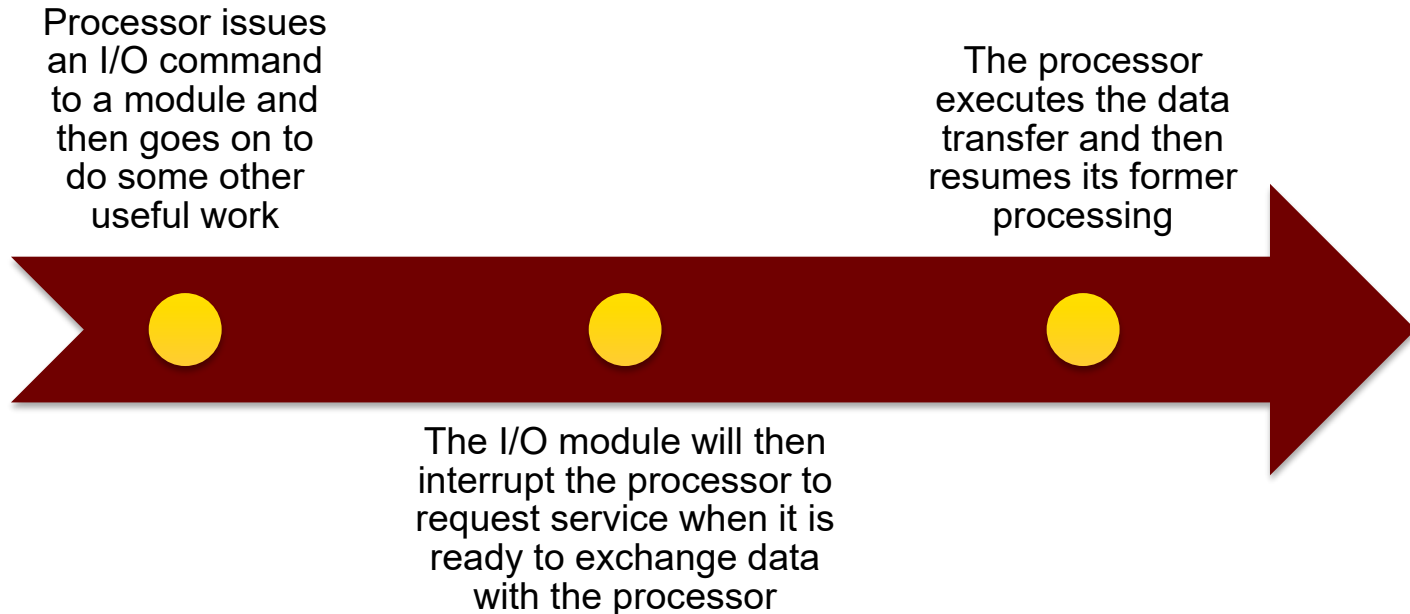
- When the processor encounters an instruction relating to I/O, it executes that instruction by issuing a command to the appropriate I/O module
- Three techniques are possible for I/O operations:
 - Programmed I/O
 - Interrupt-Driven I/O
 - Direct Memory Access (DMA)



Programmed I/O

- The I/O module performs the requested action then sets the appropriate bits in the I/O status register
- The processor periodically checks the status of the I/O module until it determines the instruction is complete
- With programmed I/O the performance level of the entire system is severely degraded

Interrupt-Driven I/O



More efficient than Programmed I/O but still requires active intervention of the processor to transfer data between memory and an I/O module

Drawbacks



53

- Transfer rate is limited by the speed with which the processor can test and service a device
- The processor is tied up in managing an I/O transfer
 - a number of instructions must be executed for each I/O transfer

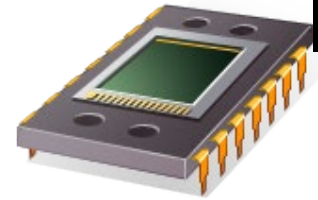
Direct Memory Access (DMA)

- Performed by a separate module on the system bus or incorporated into an I/O module
- When the processor wishes to read or write data it issues a command to the DMA module containing:
 - whether a read or write is requested
 - the address of the I/O device involved
 - the starting location in memory to read/write
 - the number of words to be read/written

Direct Memory Access

- Transfers the entire block of data directly to and from memory without going through the processor
 - processor is involved only at the beginning and end of the transfer
 - processor executes more slowly during a transfer when processor access to the bus is required
- More efficient than interrupt-driven or programmed I/O

Symmetric Multiprocessors (SMP)

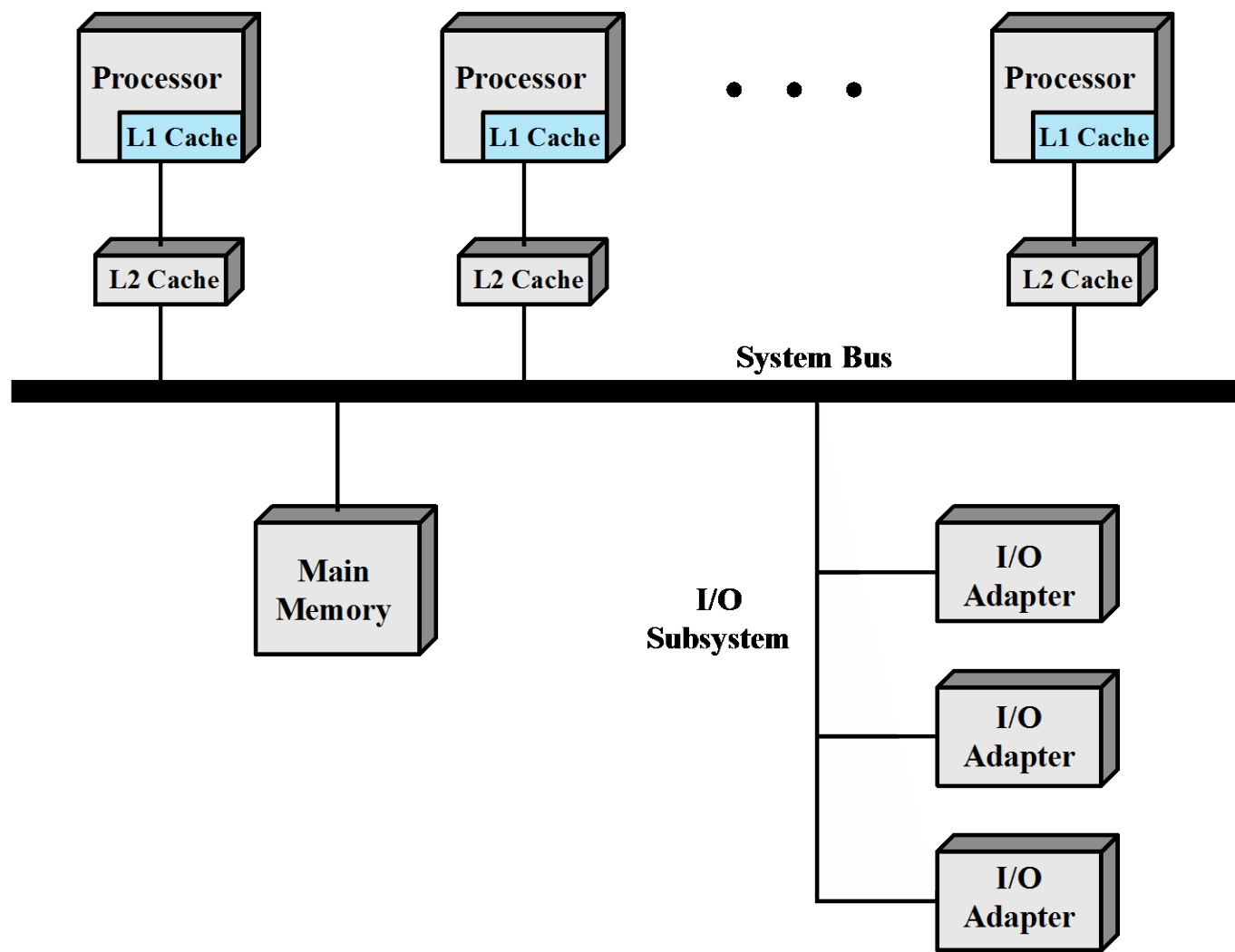


56

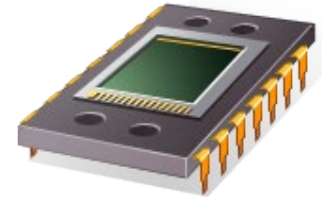
- A stand-alone computer system with the following characteristics:
 - two or more similar processors of comparable capability
 - processors share the same main memory and are interconnected by a bus or other internal connection scheme
 - processors share access to I/O devices
 - all processors can perform the same functions
 - the system is controlled by an integrated operating system that provides interaction between processors and their programs at the job, task, file, and data element levels

SMP Advantages

- Performance
 - a system with multiple processors will yield greater performance if work can be done in parallel
- Availability
 - the failure of a single processor does not halt the machine
- Incremental Growth
 - an additional processor can be added to enhance performance
- Scaling
 - vendors can offer a range of products with different price and performance characteristics

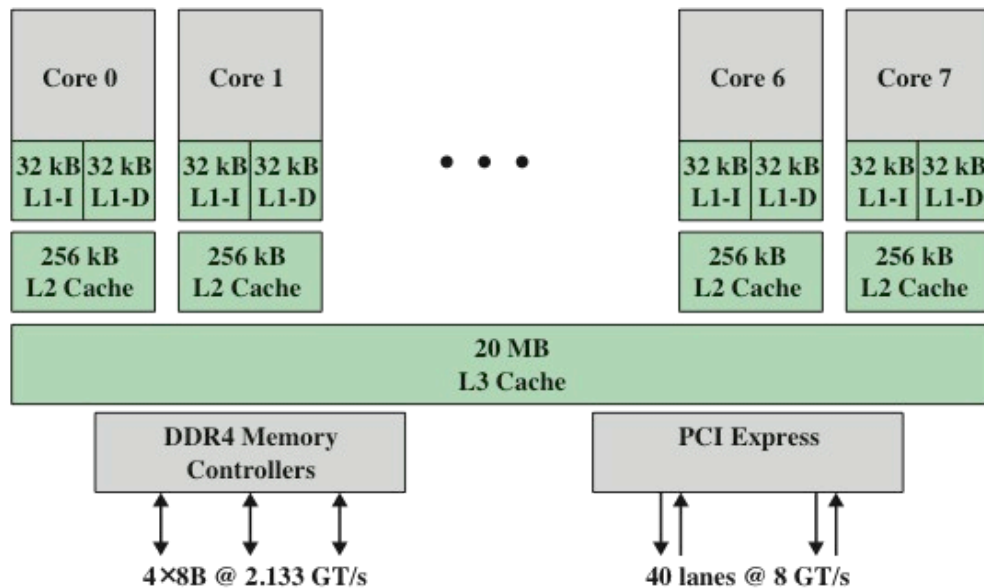


Multicore Computer

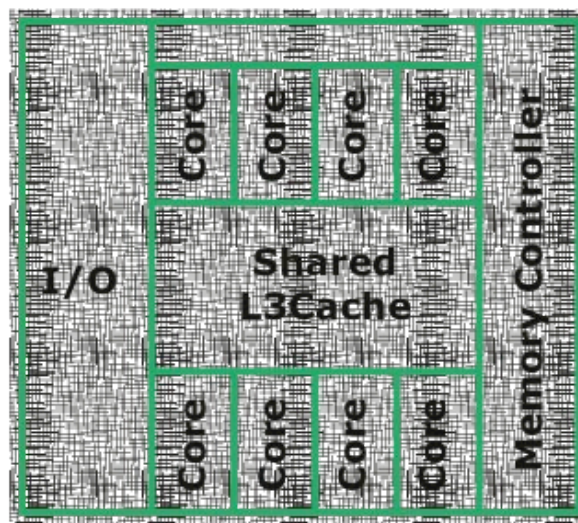


59

- Also known as a chip multiprocessor
- Combines two or more processors (cores) on a single piece of silicon (die)
 - each core consists of all of the components of an independent processor
- In addition, multicore chips also include L2 cache and in some cases L3 cache



(a) Block diagram



(b) Physical layout on chip

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
- Memory hierarchy is the solution to the conflicting trade-off among cost, capacity, access time
- Cache is invisible to OS but principles of cache is applied in virtual memory
- DMA frees processor from the burden of I/O

References

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