Operating Systems:
Internals and Design Principles,
8/E
William Stallings

Chapter 1
Computer System Overview

Patricia Roy
Manatee Community College, Venice, FL

TIMETABLE FOR EKT333

Tuesday : 8:00 AM to 9:00AM (BPU4)
Friday : 5:00 PM to 7:00PM (BPU5)

Course Outcomes

CO1: Ability to interpret the major concepts of components which build up an operating system

CO2: Ability to utilize GNU/Linux operating system for coding, compile, execute and test C programming in simulating processes in operating system

CO3: Ability to analyze and apply processes, thread, file management, processor scheduler and memory management in operating system

CO4: Ability to design and develop the applications of operating system

Course Assessment

1) Examinations (80%)
2) Coursework (20%)

Course Evaluation

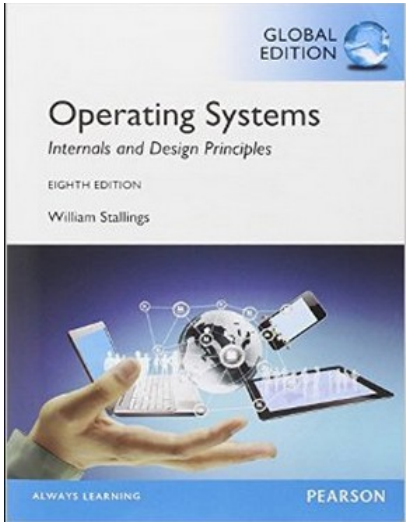
1) Final Examination = 60%
2) Examination I = 10%
3) Examination II = 10%
4) Coursework (Quizzes/Assignments) = 20
   - Quizes = 10%
   - Assignments = 10%

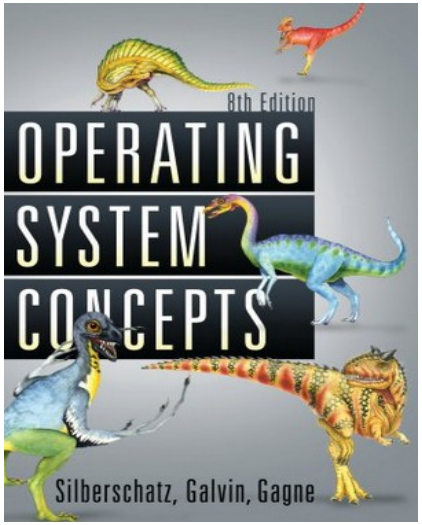
REFERENCES

1. W. Stallings, Operating Systems : Internals and Design Principles, 8th Edition, Pearson Prentice Hall, 2015.

2. A. Silberchatz, Galvin & Gagne, Operating System Principles, 8th Edition. John Wiley, 2008

3. A.S. Tanenbaum, A.S. Woodhull, Operating Systems Design and Implementation, 3rd Edition. Prentice Hall., 2006.





Course Outcomes Matrix

Course Outcome (CO) Domain	Level of complexity	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	Possible Assessment
CO1: Ability to **interpret** the major concepts of components which build up an operating system	**C2 A2 P3**		√	√								√		Assignment, Quiz, Tests, Examination
CO2: Ability to **utilize** GNU/Linux operating system for coding, compile, execute and test C programming in simulating processes in operating system	**C4 A3 P4**		√	√								√		Assignment, Quiz, Tests, Examination
CO3: Ability to **analyze** and **apply** processes, thread, file management, processor scheduler and memory management in operating system	**C4 A3 P4 CTP S**		√	√								√		Assignment, Quiz, Tests, Examination
CO4: Ability to **design** and **develop** the applications of operating system	**C6 A3 P4 CTP S**		√	√								√		Assignment, Quiz, Tests, Examination

Method of Assessment and Course Outcomes

Study Week	Course Content	Delivery Mode	Level Of Complexity	Possible Asssessment
Week 1	Computer System & Operating System Overview - Introduction to Operating System - Processor, Instruction and Memory - Modern Operating System	Lecture; Problem Solving	C4	Examination, Test 1 Quiz 1
Week 2	Process Description and Control - Process: States, Description and Control - Process Management	Lecture; Problem Solving	C6	Examination, Test 1 Quiz 1
Week 3- 4	Threads, SMP and Microkernels - Processes an Threads - Microkernels - Processes and Threads Managements	Lecture; Problem Solving	C6	Examination, Test 1 Quiz 2 Assignment 1
Week 5-6	Mutual Exclusion and Synchronization - Concurrency - Mutual Exclusion - Semaphores - Monitors - Message Passing	Lecture; Problem Solving	C6	Examination, Test 1 Quiz 2 Assignment 2

Method of Assessment and Course Outcomes

Study Week	Course Content	Delivery Mode	Level Of Complexity	Possible Assessment
Week 7-8	Deadlock and Starvation - Deadlock - Deadlock: Prevention, Avoidance and Detection - Deadlock Strategy - Dining Philosophers Problem - Concurrency Mechanism	Lecture; Problem Solving	C6	Examination, Test 1 Quiz 2 Assignment 2
Week 9	Memory Management - Memory Management Requirements - Memory Partitioning - Paging - Segmentation	Lecture; Problem Solving	C6	Examination, Test 2 Quiz 3 Assignment 3
Week 10	Virtual Memory - Hardware and Control Structures - Operating System Software Memory Managements	Lecture; Problem Solving	C6	Examination, Test 2 Quiz 3 Assignment 3
Week 11 - 12	Uniprocessor Scheduling, I/O Management and Disk Scheduling - Types of Processor Scheduling - Scheduling Algorithms - Traditional UNIX Scheduling - Operating System Design Issues - I/O Buffering, Disk Scheduling, RAID, Disk Cache	Lecture; Problem Solving	C6	Examination, Test 2 Quiz 4 Assignment 4

Method of Assessment and Course Outcomes

Study Week	Course Content	Delivery Mode	Level Of Complexity	Possible Assessment
Week 13	File Management - File Organization and Access - File Directories and Sharing - Record Blocking - Secondary Storage Management	Lecture; Problem Solving	C6	Examination, Test 2 Quiz 4 Assignment 4
Week 15	MINGGU ULANGKAJI / REVISION WEEK			
Week 16	PEPERIKSAAN AKHIR SEMESTER / FINAL EXAMINATION			

Outline:

1.1 Basic Elements

1.2 Processor Registers

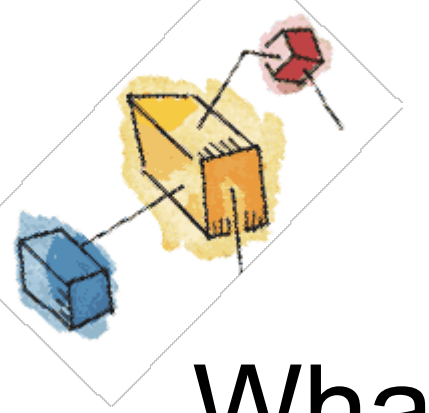
1.3 Instruction Execution

1.4 Interrupts

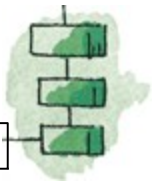
1.5 Memory Hierarchy

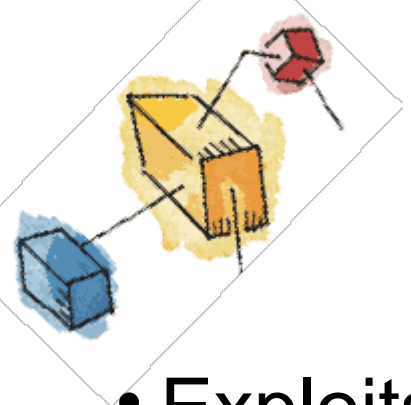
1.6 Cache Memory

1.7 I/O Communication Technique



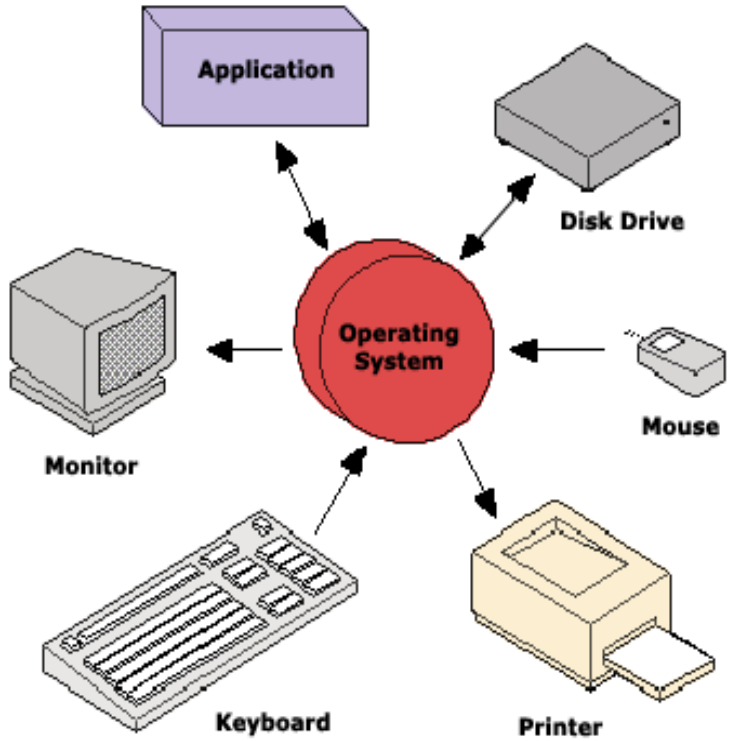
What is an Operating System? Example?

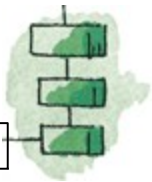




Operating System

- Exploits the hardware resources of one or more processors
- Provides a set of services to system users
- Manages secondary memory and I/O devices





Basic Elements

Processor

Main Memory

I/O Modules

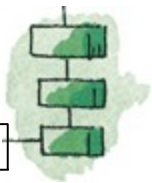
System Bus

Execute Programs!

Processor

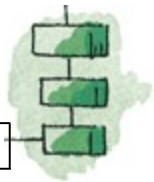
Controls the operation of the computer

- Central Processing Unit (CPU)
- Function:
    - Exchange data with memory
    - Two internal registers - (Address and Buffer)
        - MAR – Specifies address in memory for the next read or write
        - MBR – Contains data to be written into memory / receives data from memory
        - I/OAR – Specifies a particular I/O address
        - I/OBR - Exchange data between I/O module and processor



Main Memory

- Sets of locations to store data & programs
- Addresses (instructions / data)
- Volatile
  - Contents of the memory is lost when the computer is shut down
  - Referred to as real memory or primary memory

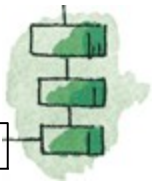


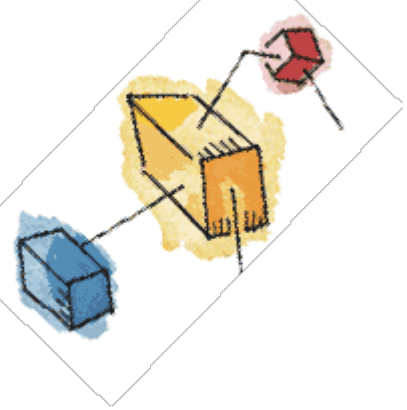
I/O Modules

Moves data between computer and external devices such as:

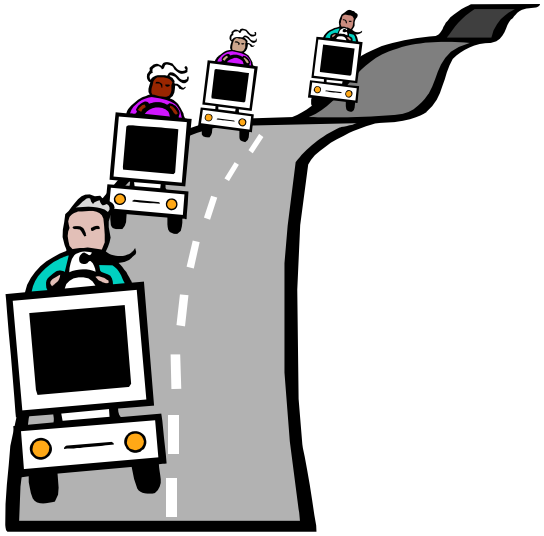
- Storage: (e.g. Hard Drives)
- Communications Equipments (NIC)
- Terminals

* Contains Memory to hold data

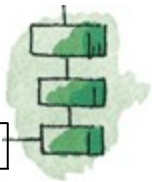




System Bus



Provides communication among processors, main memory, and I/O modules



Computer Components: Top-Level View

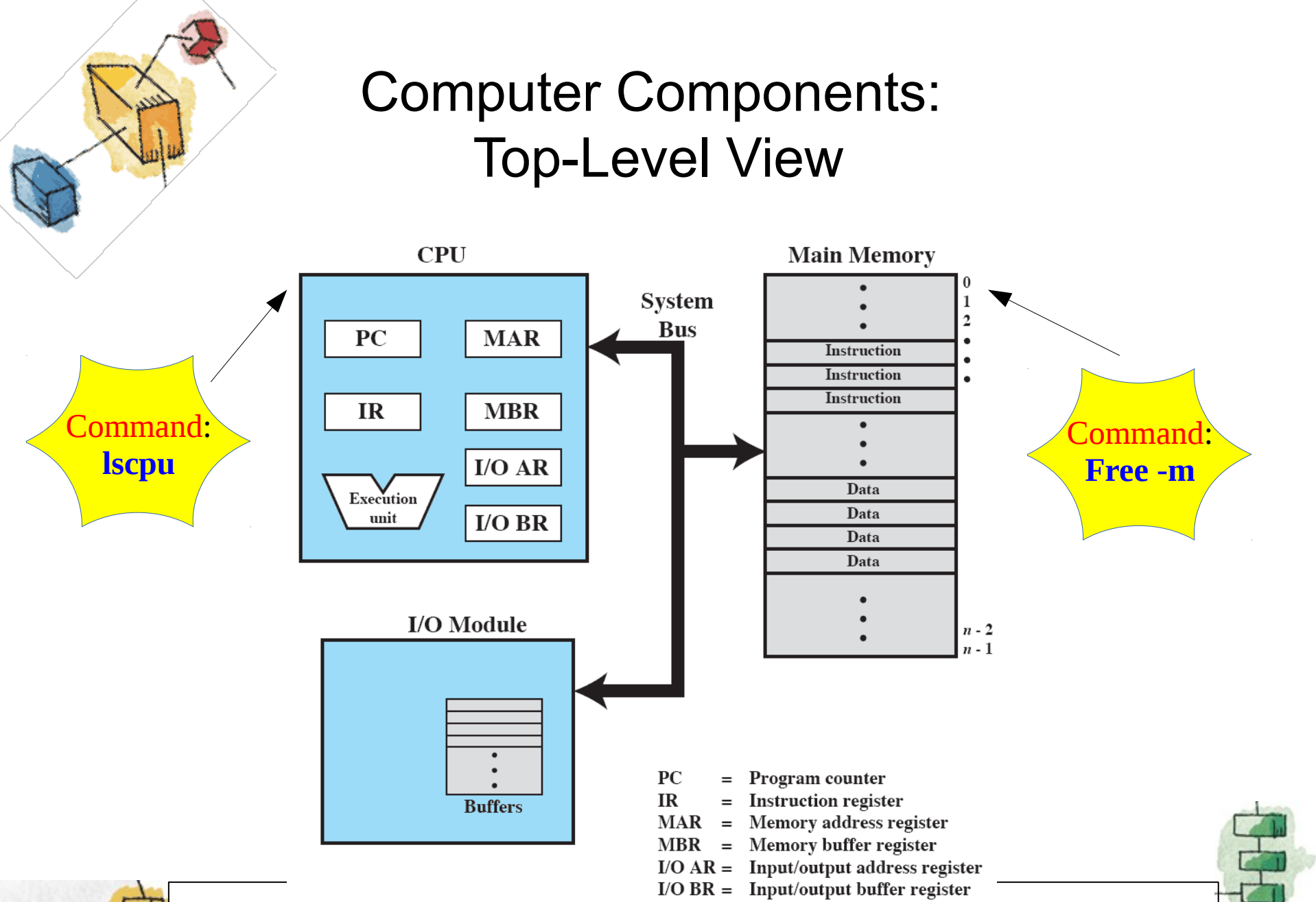
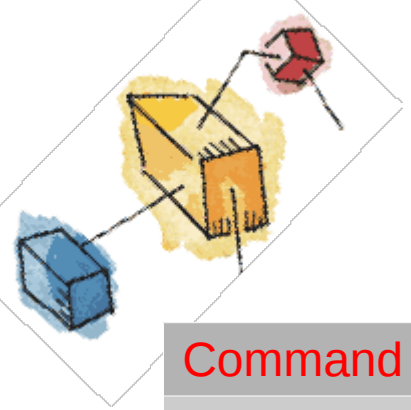
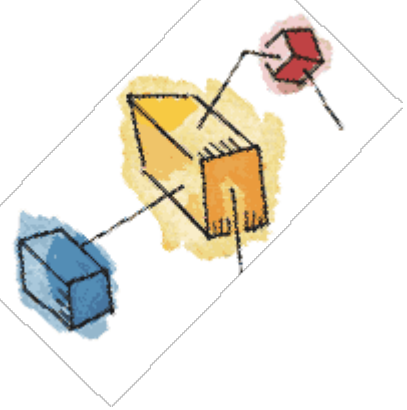


Figure 1.1 Computer Components: Top-Level View



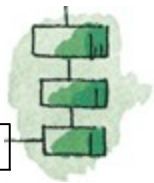
Related Command Lines

Command	Meaning
lscpu	To display information about the CPU architecture
free -m	To check memory usage on linux
top	To check memory and CPU usage per process. However it also reports total memory usage and can be used to monitor the total RAM usage.
htop	Shows memory usage along with various other details (e.g. RAM, swap).
sudo dmidecode -t 17	Find out hardware information about the installed RAM, use the demidecode command. It reports lots of information about the installed RAM memory



Processor Registers

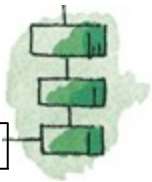
- Small amount of storage! - can be accessed quickly
- A processor consists of a set of registers that provide a type of memory that is faster and smaller than the main memory
- Functions:
  - User-visible registers
  - Control and status registers



User-Visible Registers

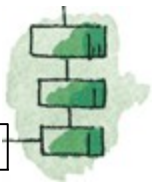
- May be referenced by machine language
- Minimize references to main memory
- Available to all programs – application programs and system programs

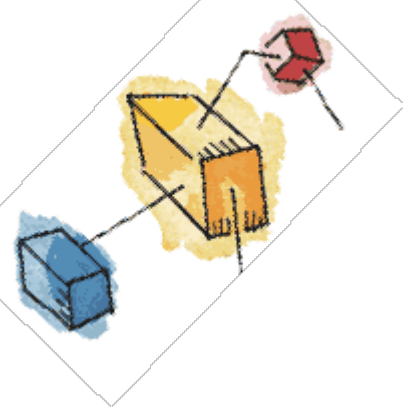
- Types:
  - Data Registers
  - Address Registers



User-Visible Registers (Address Registers)

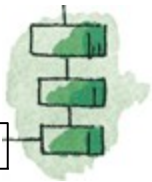
- Contains main memory addresses of data and instructions
- Index registers
  - Adding index to a base value
- Segment pointer
  - When memory is divided into segments, memory is referenced by a segment and an offset
- Stack pointer
  - Dedicated register that points to top of stack

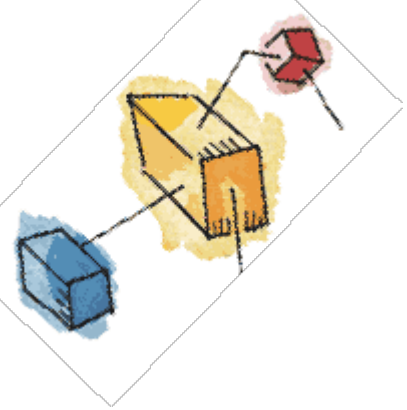




Control and Status Registers

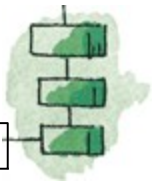
- Program counter (PC)
  - Contains the address of the next instruction to be fetched
- Instruction register (IR)
  - Contains the instruction most recently fetched
- Program status word (PSW)
  - Contains condition codes + status information
  - E.g. - Interrupt enable/disable bit

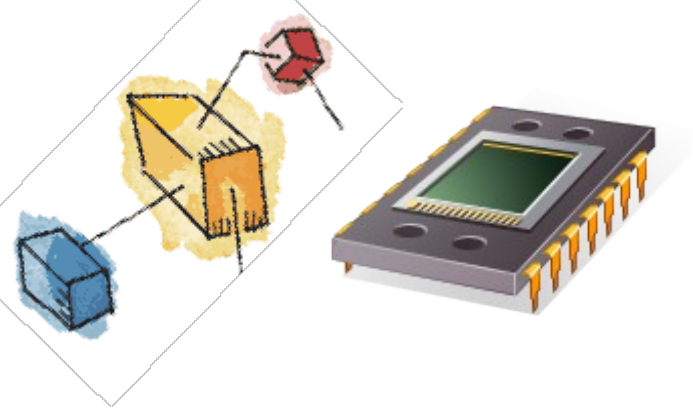




Control and Status Registers

- Condition codes or flags
  - Bits set by processor hardware as a result of operations
  - Example
    - Arithmetic operation
    - Positive, negative, zero, or overflow result
    - Set following the execution of the arithmetic instruction





Microprocessor

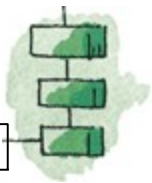
- Invention that brought about desktop and hand held computing

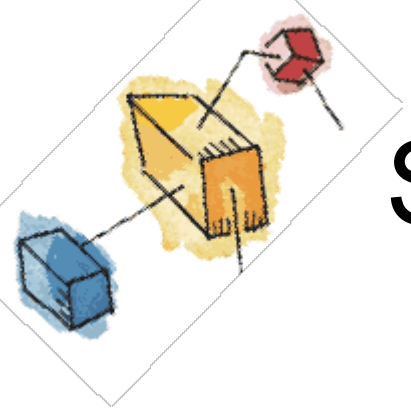
- Incorporates processor on a single chip (IC)

- Fastest general purpose processor

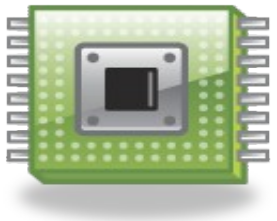
- Multipurpose, programmable, digital data as input

- Each chip contains multiple processors (cores)



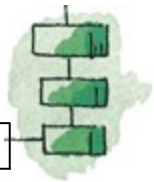


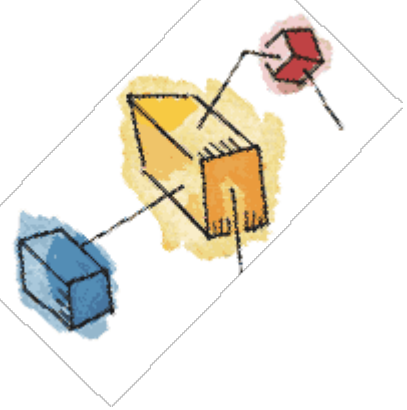
System-on-Chip (SoC)



- To satisfy the requirements of handheld devices, the microprocessor is giving way to the SoC

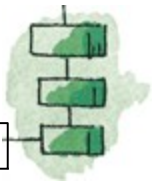
- IC that Integrates all components of computers into a single chip:
  - DSPs, GPUs, codecs and main memory, in addition to the CPUs and caches, are on the same chip

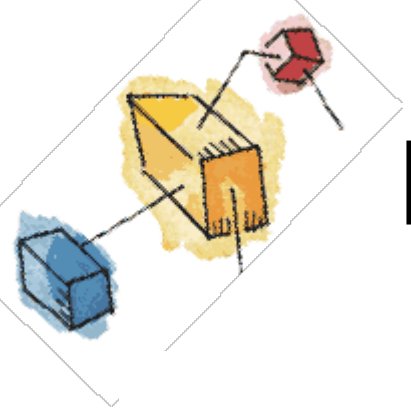




Instruction Execution

- Two steps
  - Processor reads (fetches) instructions from memory
  - Processor executes each instruction





Basic Instruction Cycle

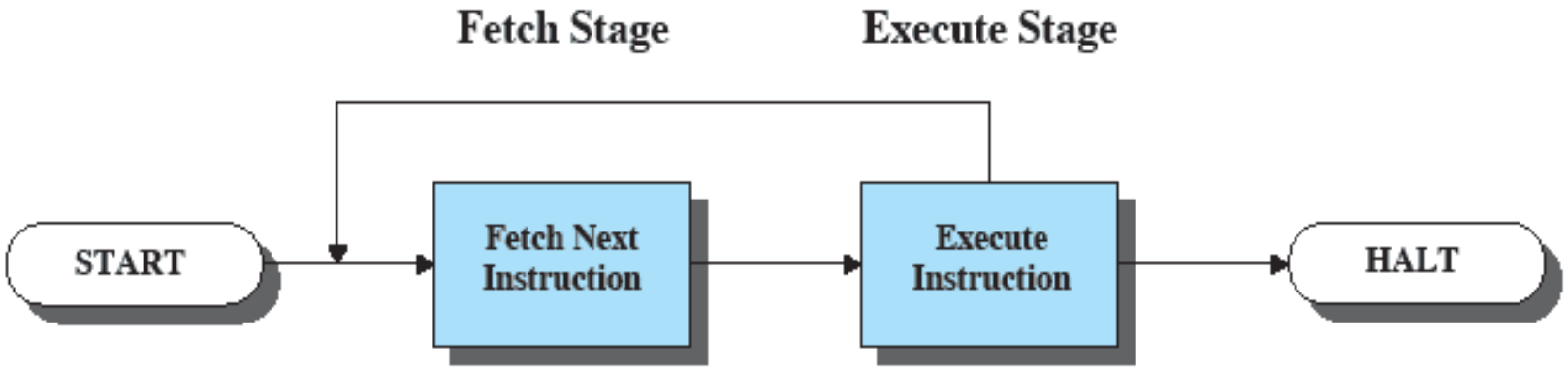
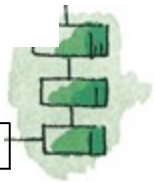
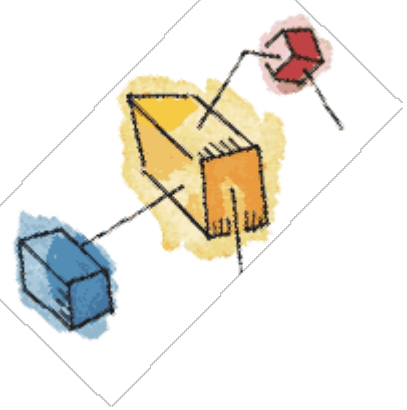


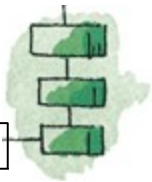
Figure 1.2 Basic Instruction Cycle

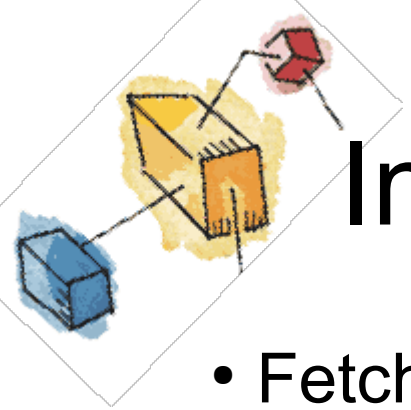




Instruction Fetch and Execute

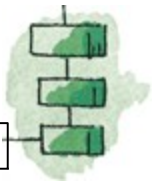
- The processor fetches the instruction from memory
- Program counter (PC) holds address of the instruction to be fetched next
- PC is incremented after each fetch

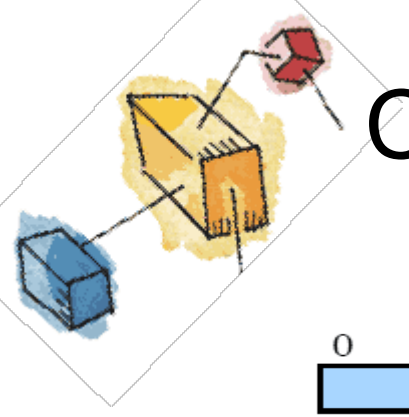




Instruction Register (IR)

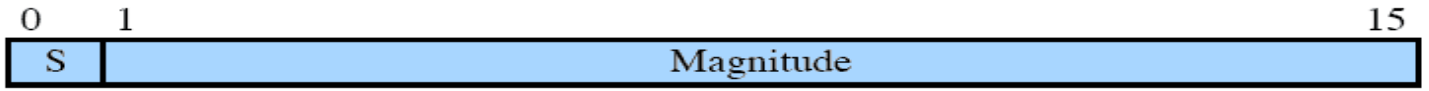
- Fetched instruction loaded into instruction register
- Contains bits that specify the action.
- **Processor interprets the instruction and performs some required actions.**
- Categories of actions:
  - Processor-memory
  - Processor-I/O
  - Data processing – arithmetic / logic
  - Control – alter the sequence of execution
    - E.g. Current address is 149, fetch 182 rather than 150





Characteristics of a Hypothetical Machine

(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

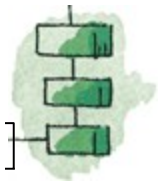
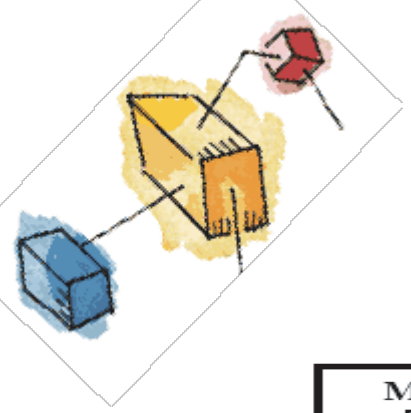


Figure 1.3 Characteristics of a Hypothetical Machine



Example of Program Execution

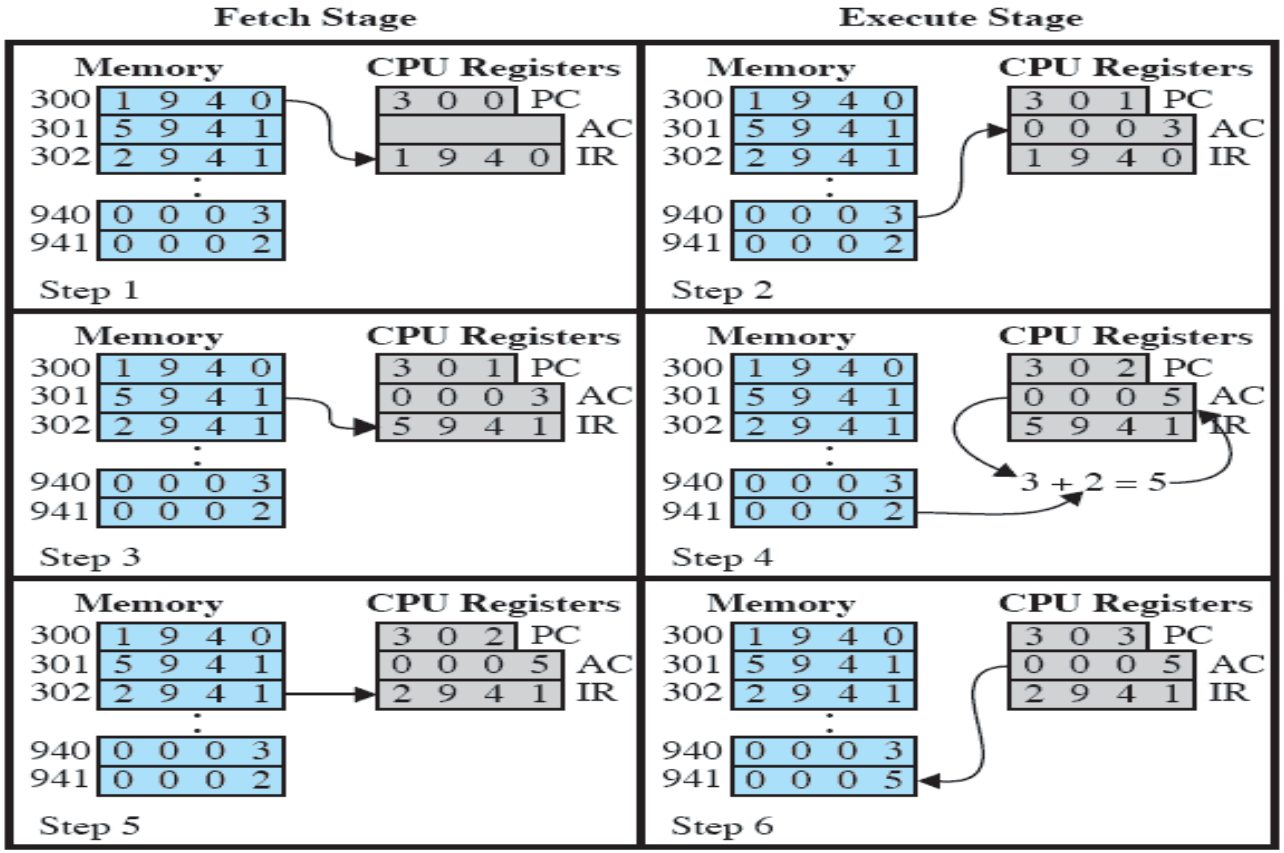
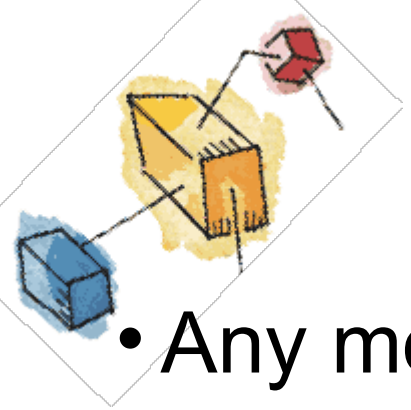
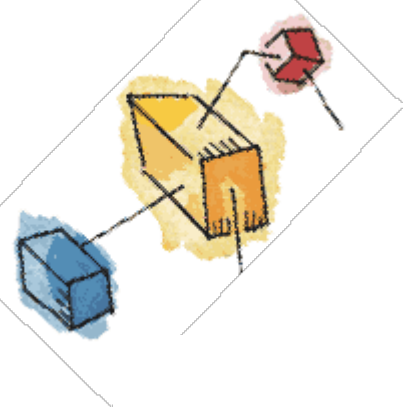


Figure 1.4 Example of Program Execution (contents of memory and registers in hexadecimal)



Interrupts

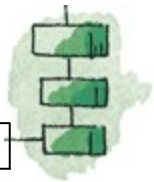
- Any modules may interrupt the normal sequencing of the processor
- Reason: To improve resource utilization (i.e. processor)
- Most I/O devices are slower than the processor
  - Processor must pause to wait for device (e.g. printer)
  - Processor remains idle
    - Millions instruction cycles – waste of resources



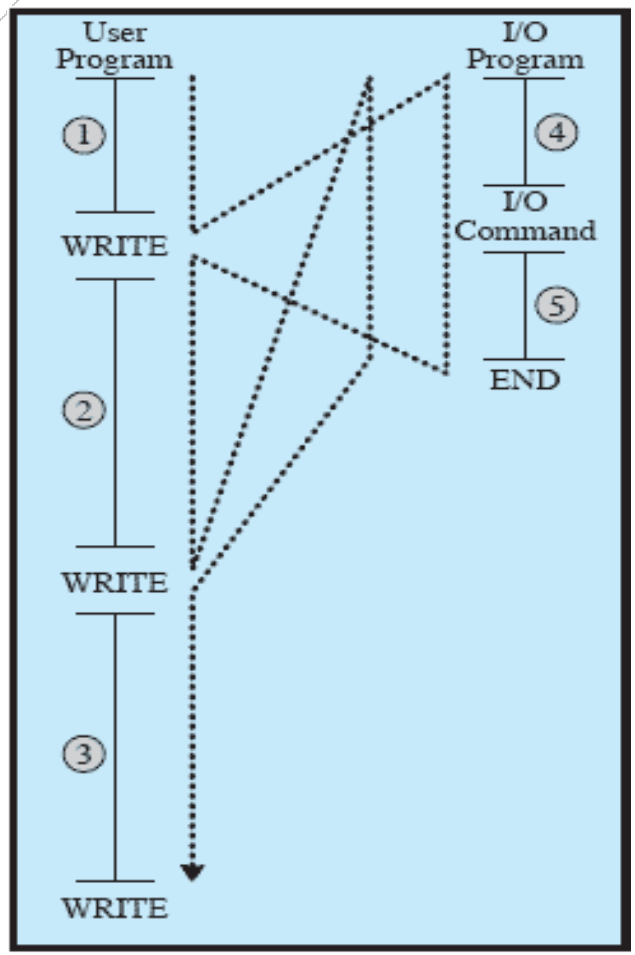
Classes of Interrupts

Table 1.1 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.



Program Flow of Control

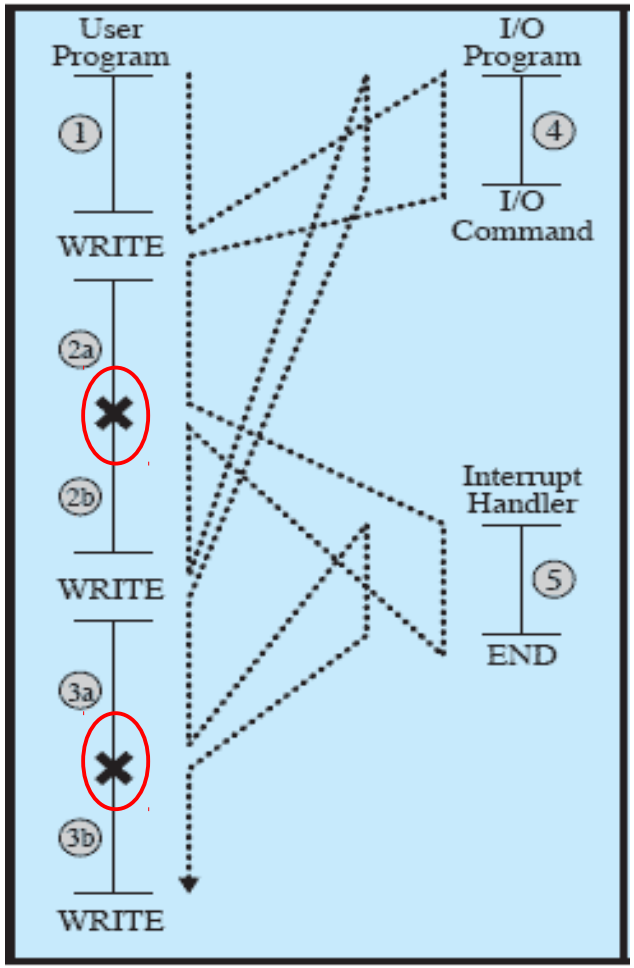


(a) No interrupts

- The I/O operation may take some time to complete. User program is stopped at the WRITE point for some considerable period of time

INTERRUPT!

Program Flow of Control



(b) Interrupts; short I/O wait

- The processor can be engaged in executing other instructions while waiting for an I/O operation to be completed
- Can occur at any point.
  - E.g when I/O device is ready to accept more data from processor
- No special code needed by user program
- I/O module sends interrupt request signal to processor
  - processor suspends current operation
  - sending off job to Interrupt handler
  - resume original process after the device is serviced

Transfer of Control via Interrupts

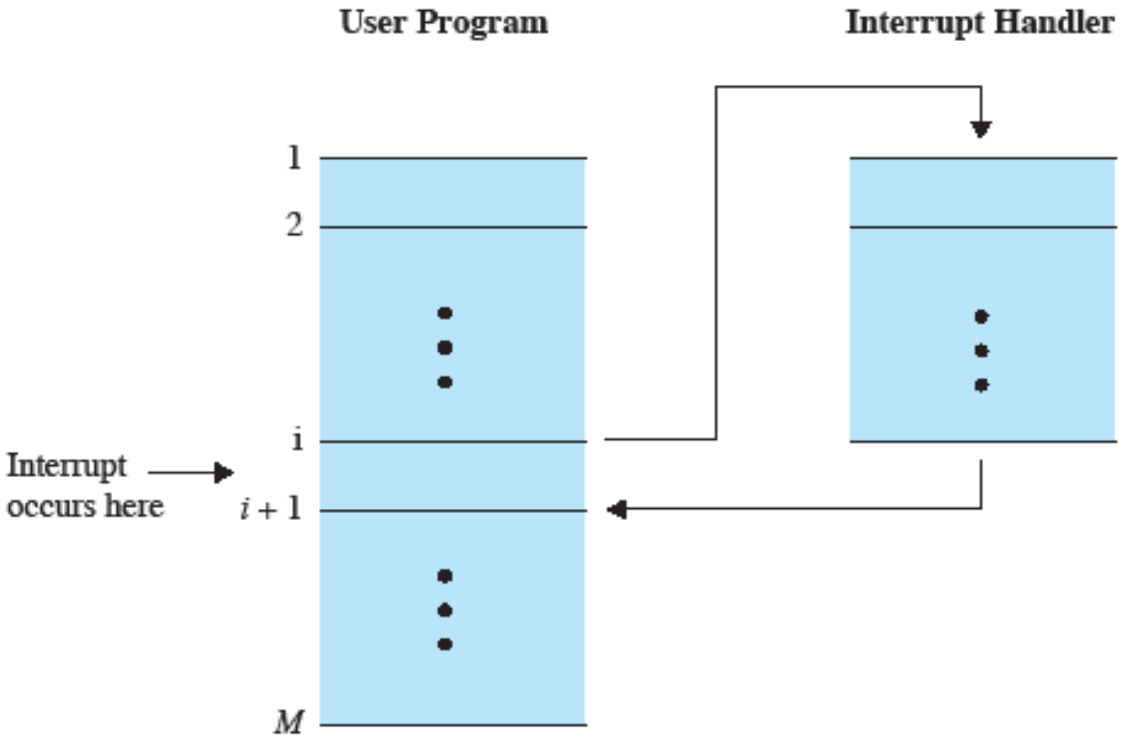
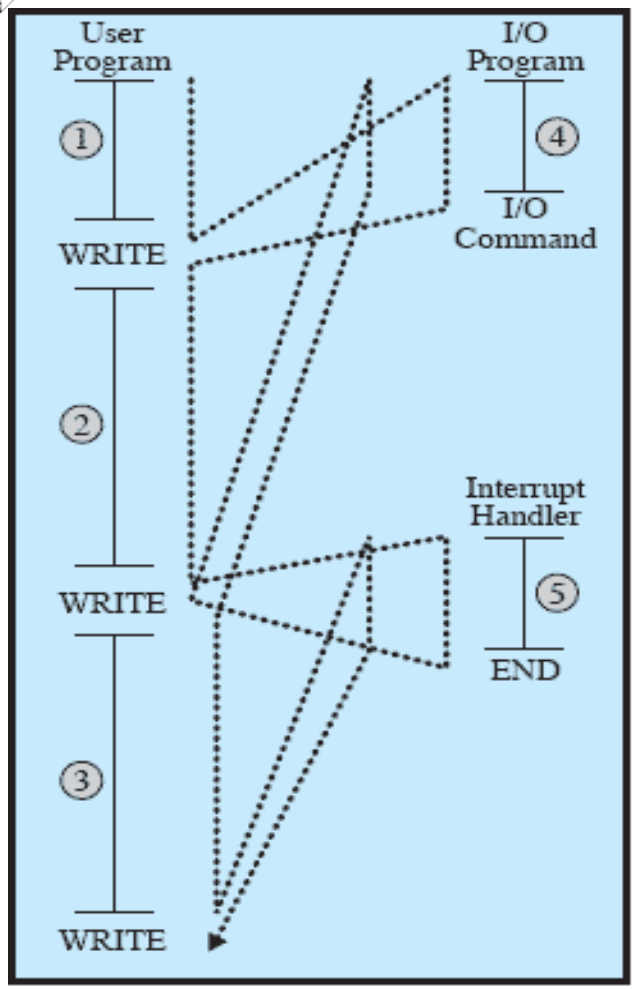


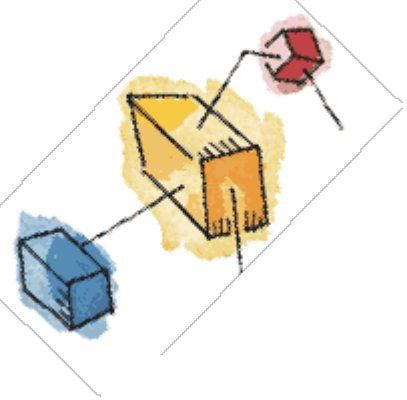
Figure 1.6 Transfer of Control via Interrupts

Program Flow of Control



(c) Interrupts; long I/O wait

- When the I/O device (e.g Printer) takes so much time compared to the sequence of user instructions
- Interrupt can only be invoked when the I/O device is ready
- Thus cause the I/O long wait



Instruction Cycle with Interrupts

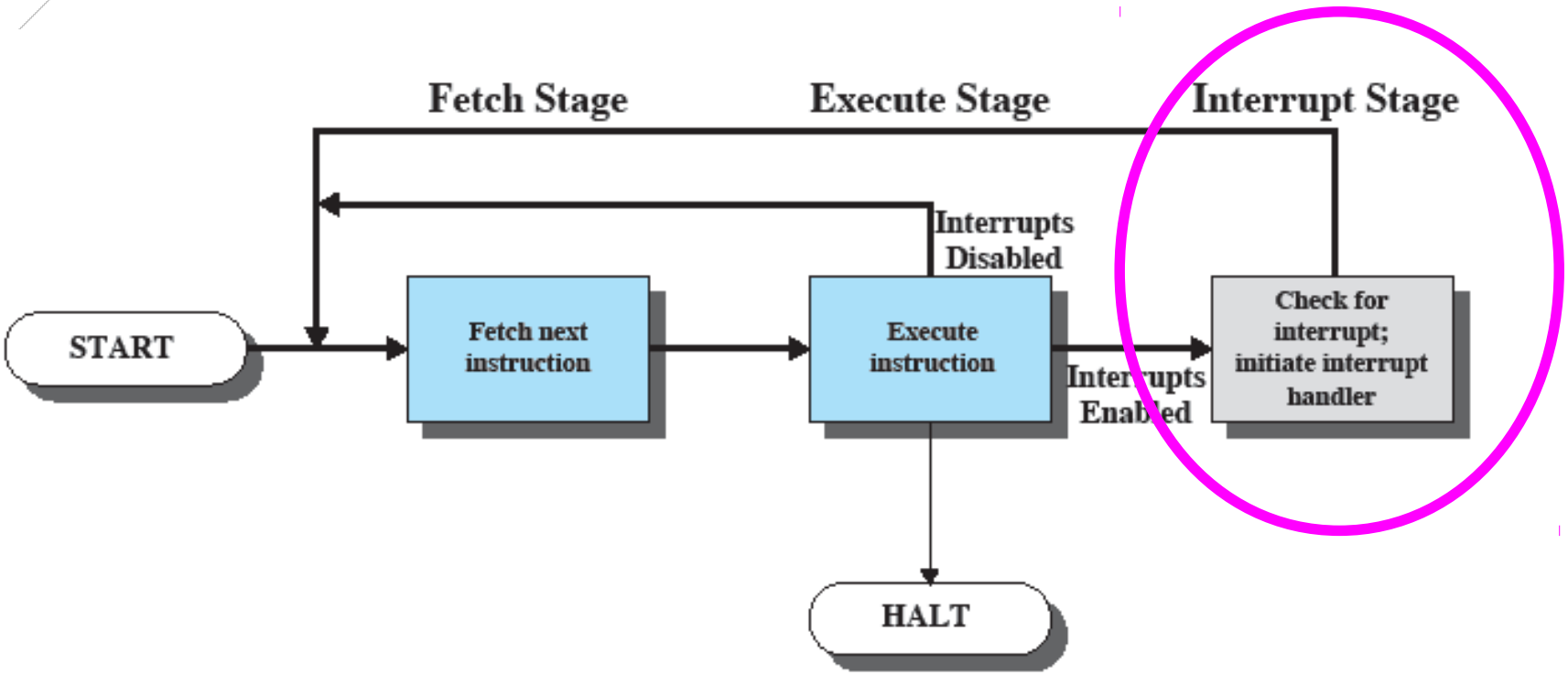
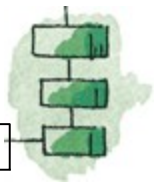
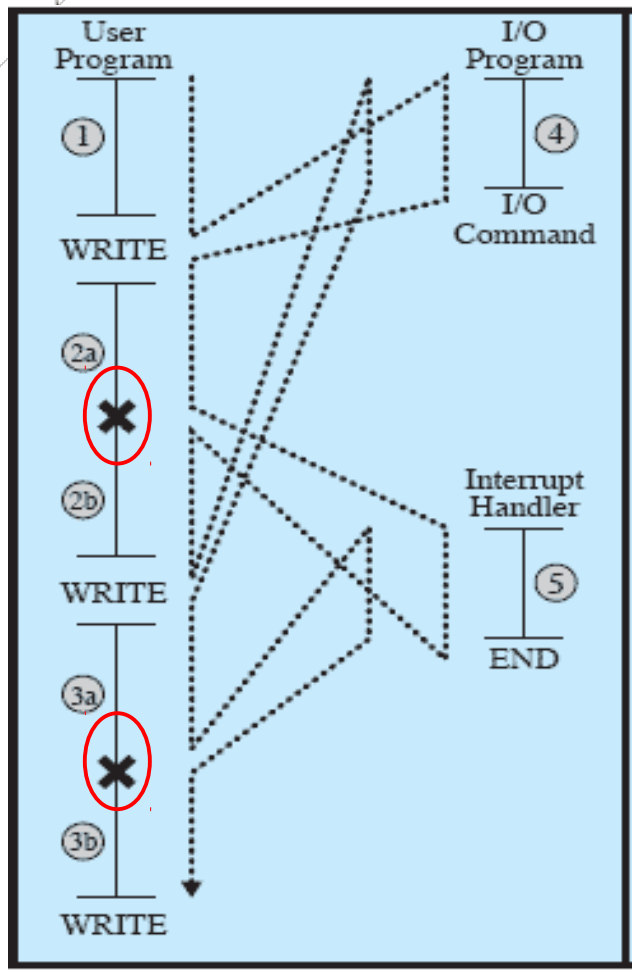


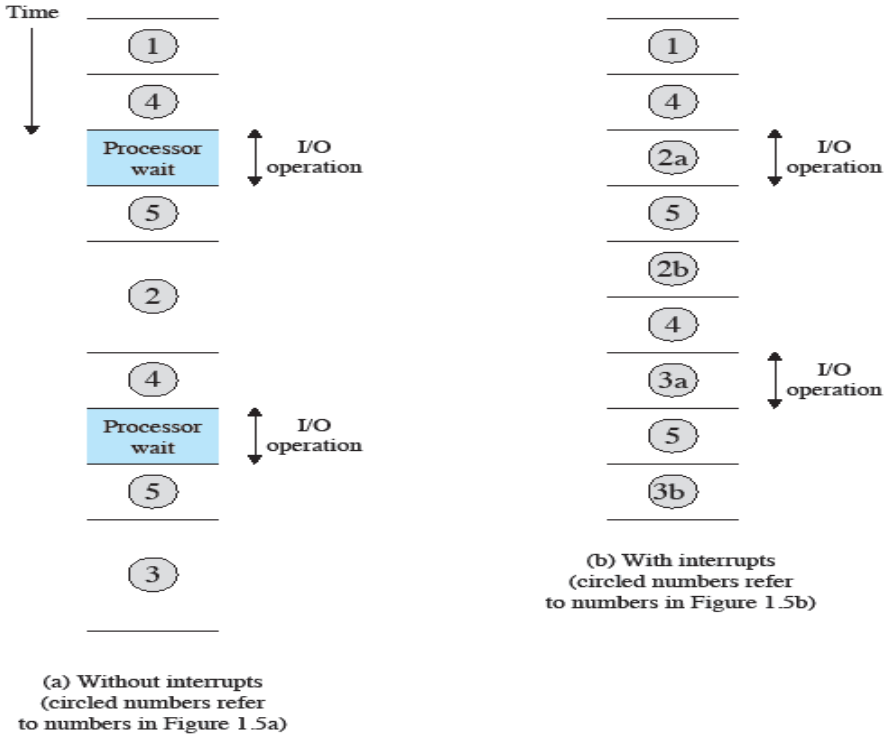
Figure 1.7 Instruction Cycle with Interrupts



Program Timing: Short I/O Wait



(b) Interrupts; short I/O wait

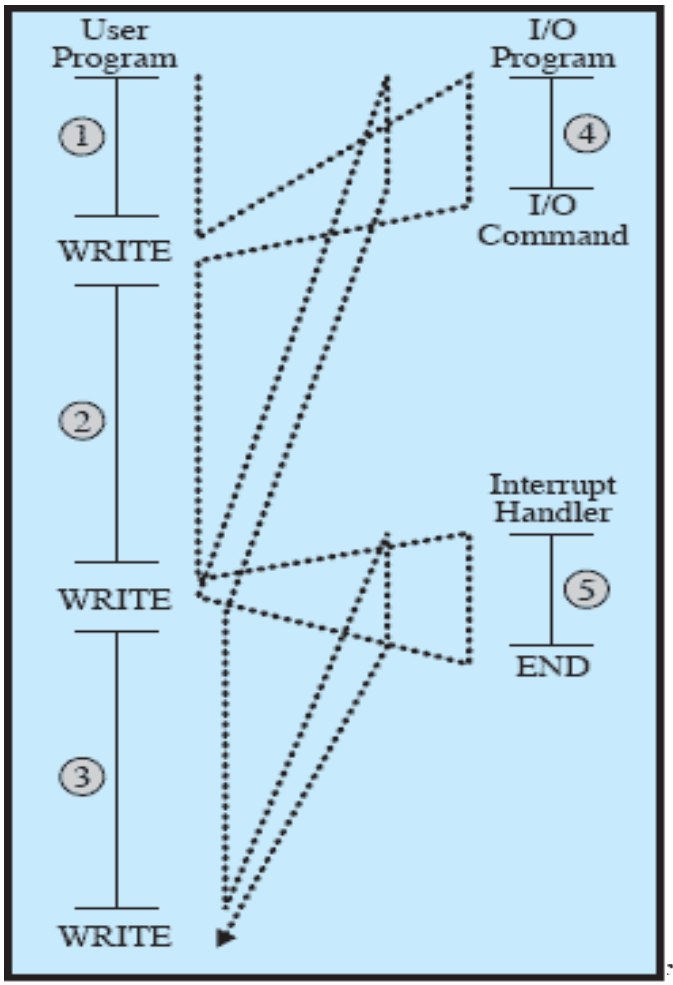


(a) Without interrupts (circled numbers refer to numbers in Figure 1.5a)

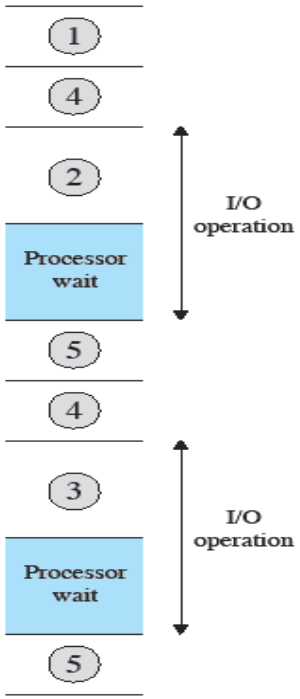
(b) With interrupts (circled numbers refer to numbers in Figure 1.5b)

Figure 1.8 Program Timing: Short I/O Wait

Program Timing: Long I/O Wait

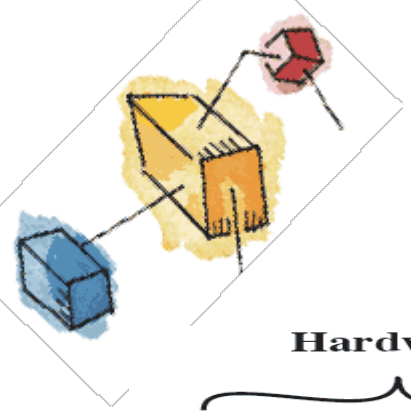


(c) Interrupts; long I/O wait

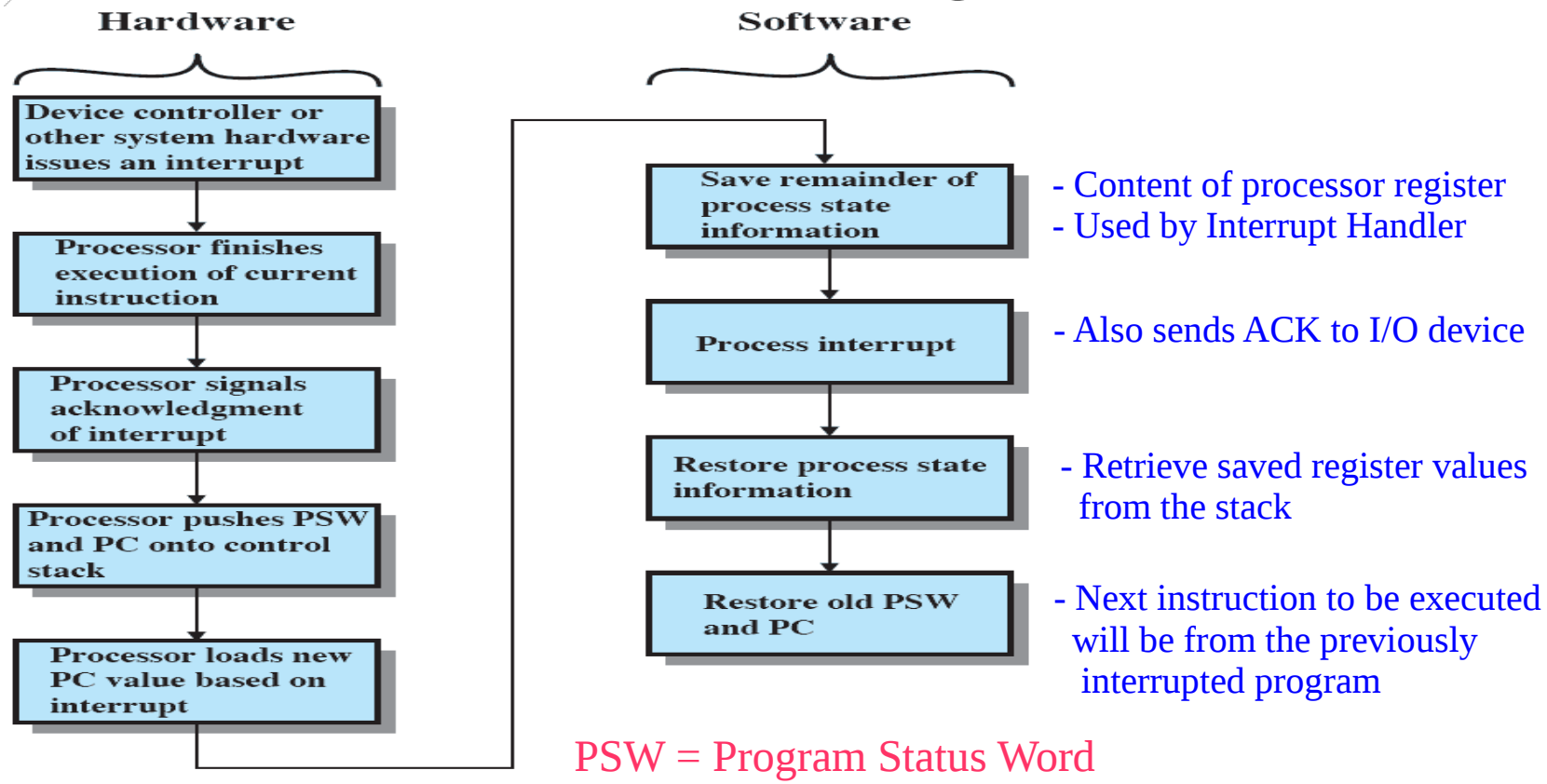


(b) With interrupts
(circled numbers refer
to numbers in Figure 1.5c)

rogram Timing: Long I/O Wait

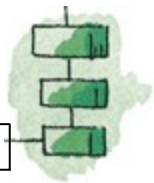


Simple Interrupt Processing

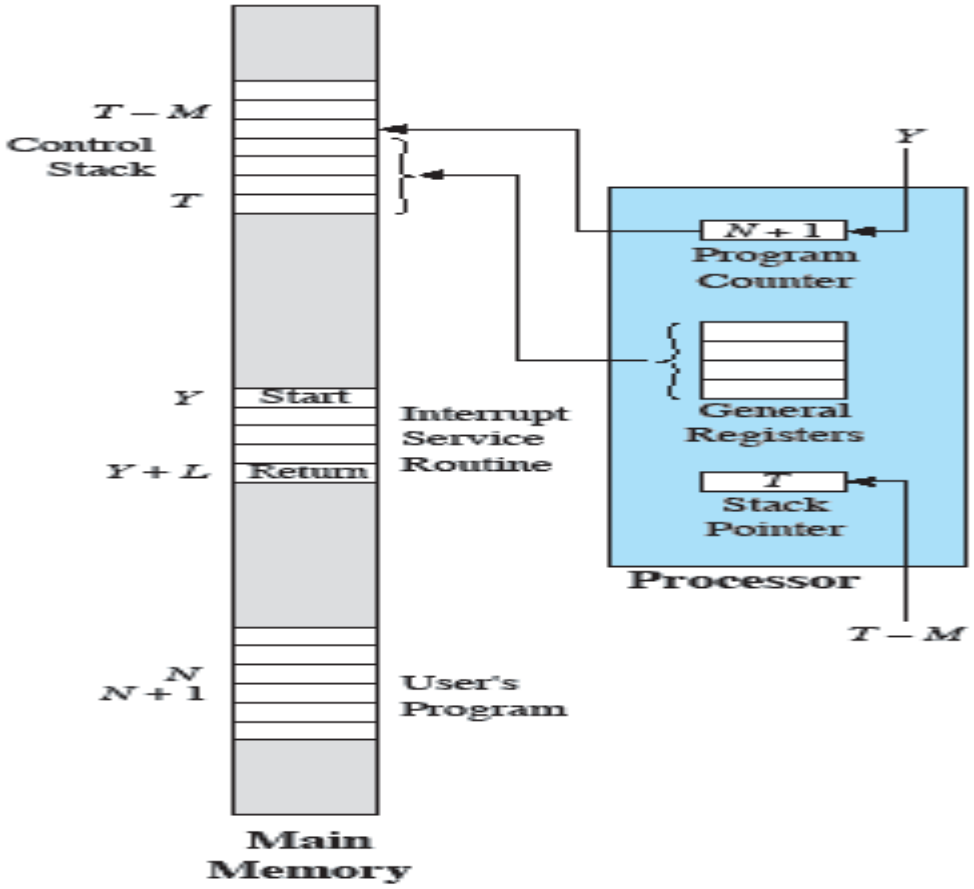


PSW = Program Status Word
PC = Program Counter

Figure 1.10 Simple Interrupt Processing

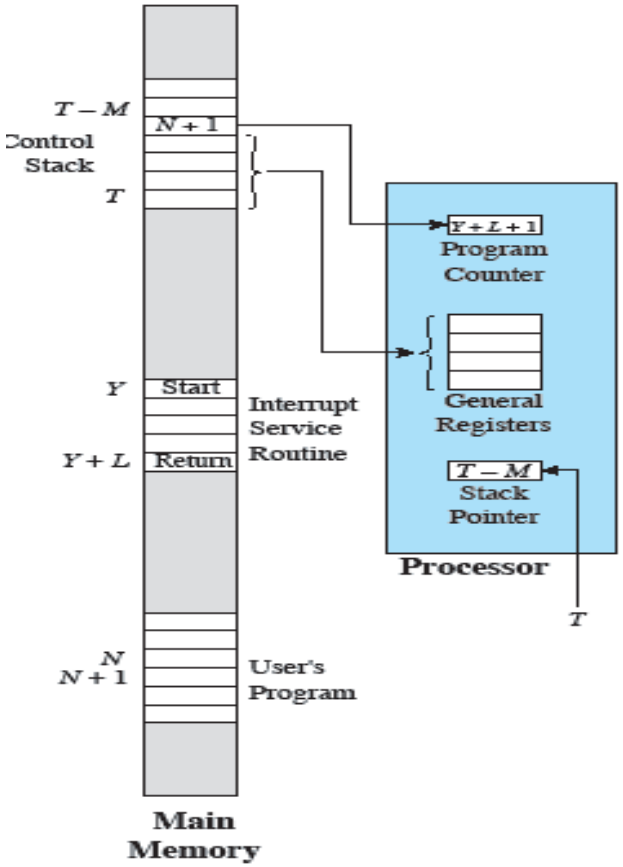


Changes in Memory and Registers for an Interrupt

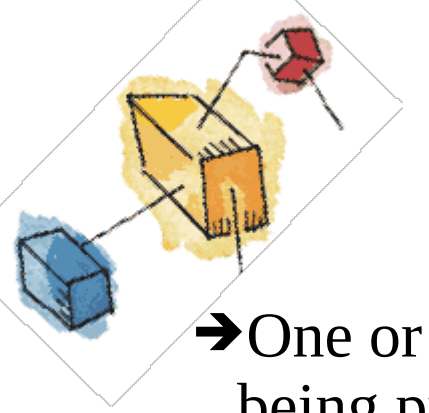


(a) Interrupt occurs after instruction at location N

Changes in Memory and Registers for an Interrupt



(b) Return from interrupt



Multiple Interrupt

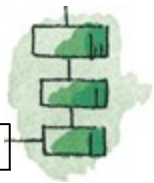
➔One or more interrupts may occur while an interrupt is being processed.

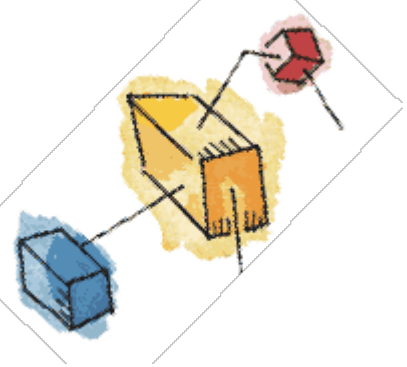
➔E.g: A program receiving data from communication line and printing results at the same time.

- ➔A printer generates an interrupt everytime it completes a print operation
- ➔A communication line controller generates an interrupt everytime a unit of data arrives.

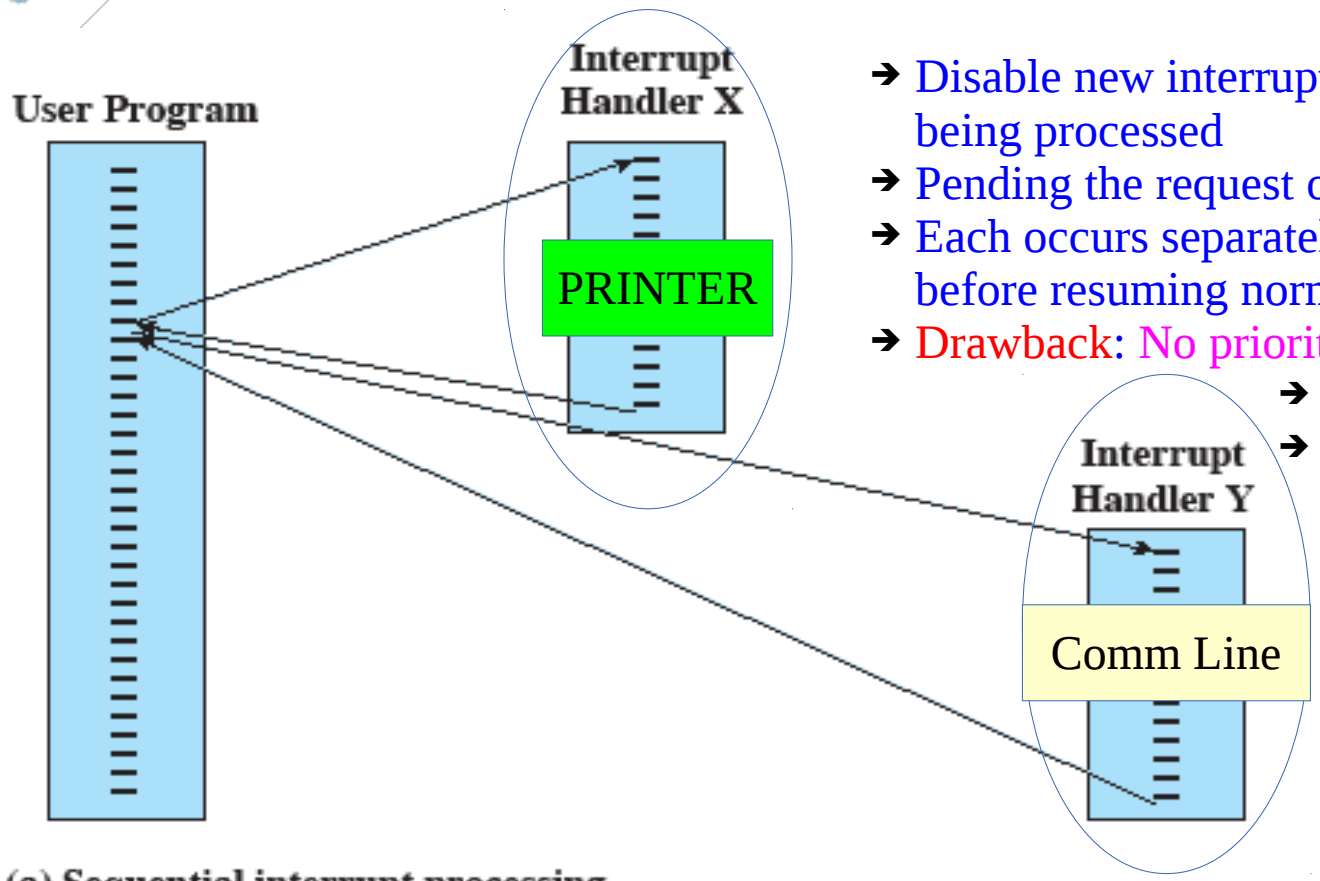
Two Types:

- Sequential
- Nested



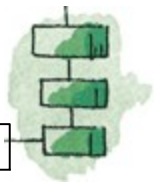


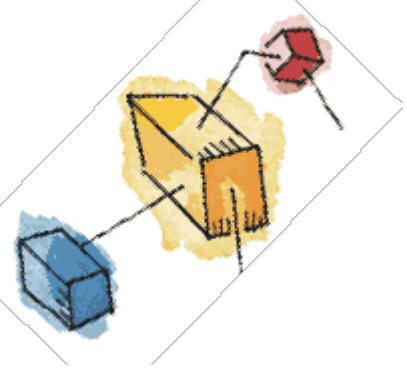
Multiple Interrupt: Sequential



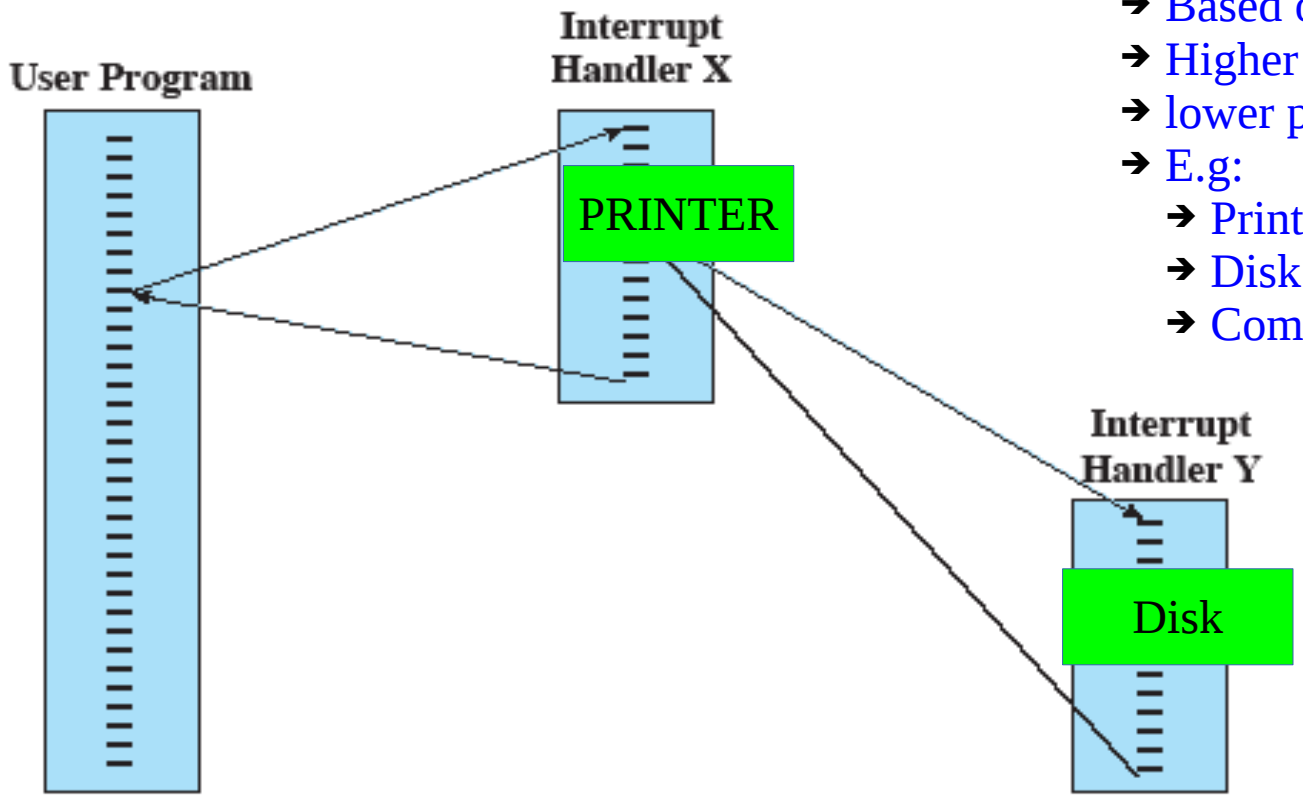
(a) Sequential interrupt processing

- Disable new interrupt while an interrupt is being processed
- Pending the request of any new interrupt signal
- Each occurs separately, one after another before resuming normal operation
- Drawback: No priority for time-critical needs
  - Buffer overflow
  - Data Loss



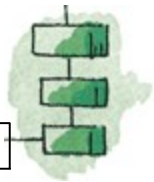


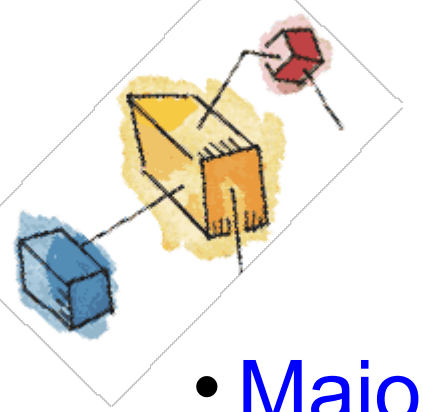
Multiple Interrupt: Nested



(b) Nested interrupt processing

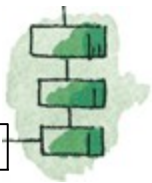
- Based on priorities
- Higher priority may interrupt the
- lower priority
- E.g:
  - Printer (2)
  - Disk (4)
  - Comm Line (5)

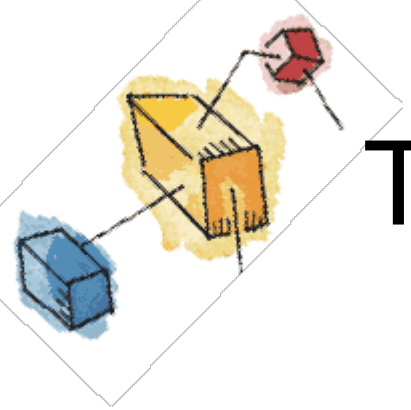




Memory Hierarchy

- Major problem of memory:
  - How much? Amount → Capacity
  - How fast? Speed → Access Time
  - How expensive? Cost
- Trade-off
  - Faster access time, greater cost per bit
  - Greater capacity, smaller cost per bit
  - Greater capacity, slower access speed
  - Solution – Memory hierarchy





The Memory Hierarchy

a) Decreasing cost per bit

b) Increasing capacity

c) Increasing access time

d) Decreasing frequency of access to the memory by processor

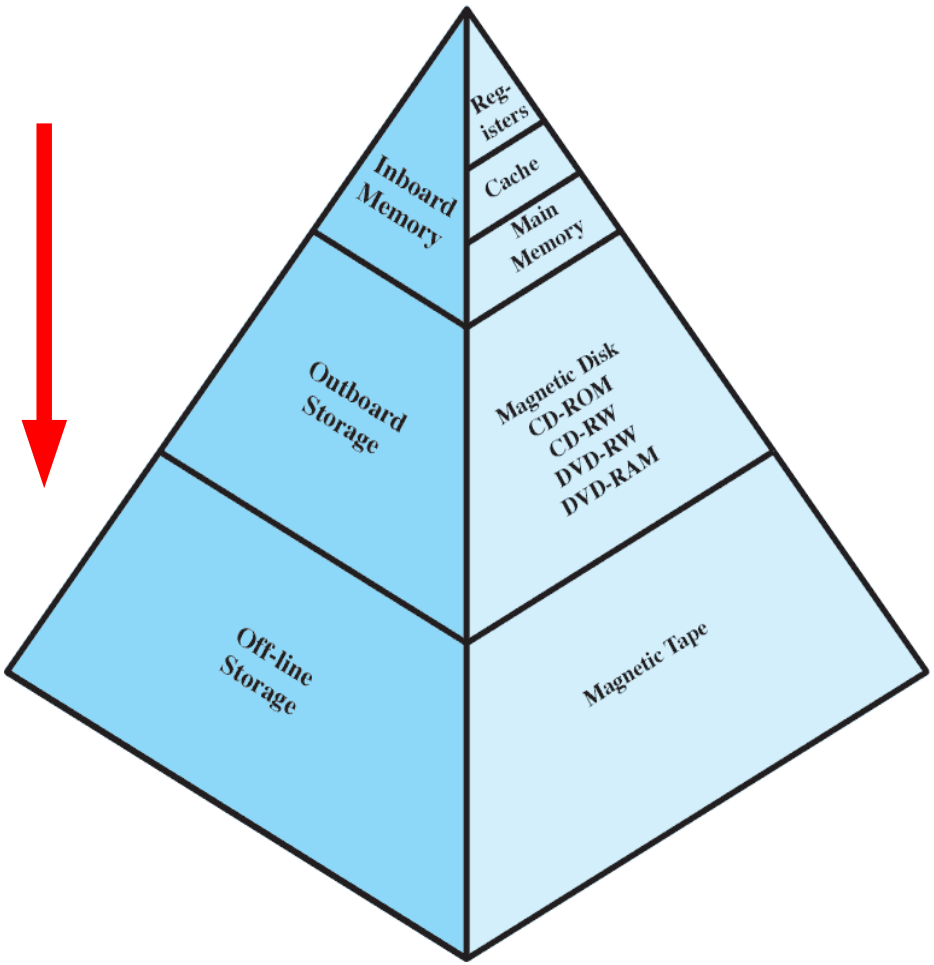


Figure 1.14 The Memory Hierarchy

Performance of a Simple Two-Level Memory

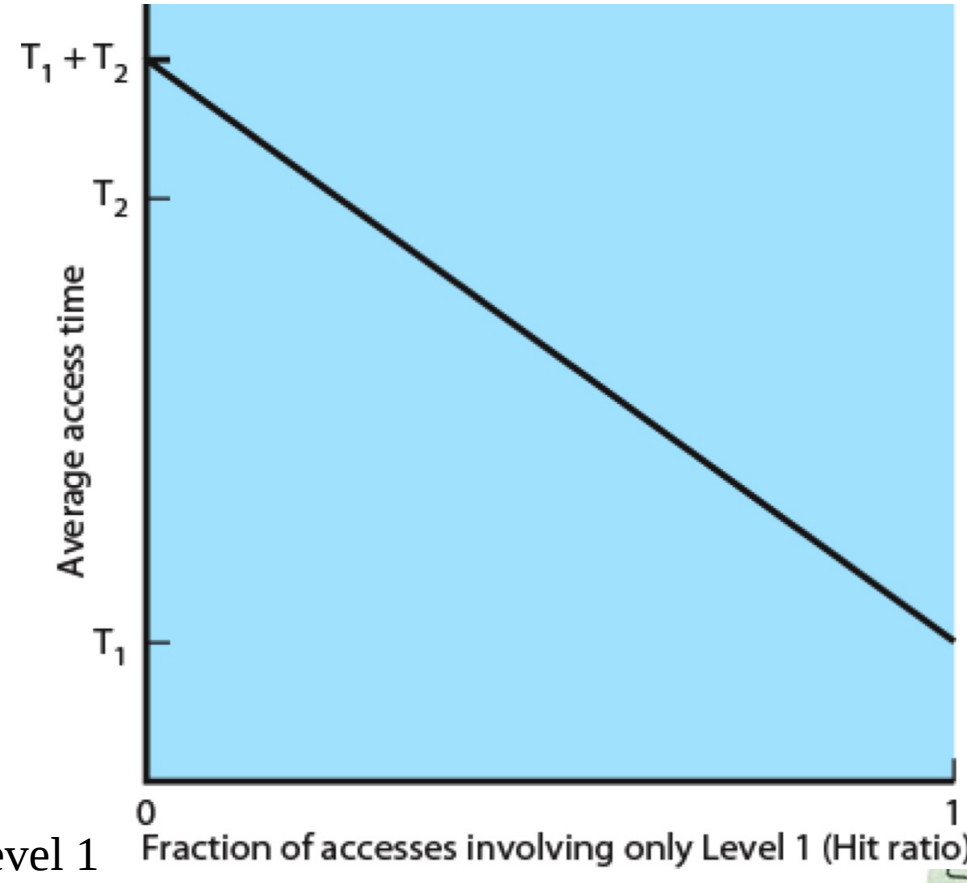
Level 1: 1000 Bytes, 0.1 µS
Level 2: 100,000 Bytes, 1 µS
** The Smaller, The Faster!

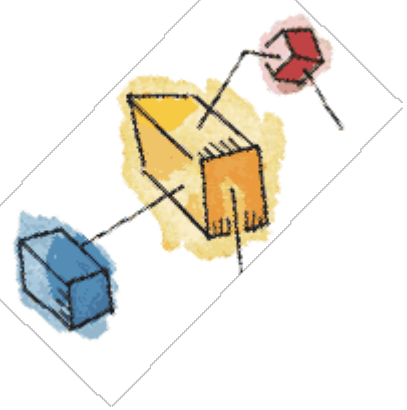
Level 1: Access directly
Level 2: Transfer to Level 1 first

Hit Ratio: If the accessed word is **found** in faster memory

Miss: The accessed word is **not found** in faster memory

For higher % of Level 1 access,
the avg access time is closer to that of Level 1





Proof: Calculation of the Avg Access Time

- Suppose 95% of memory access are found in the cache (Hit Ratio):

- Avg Access Time:

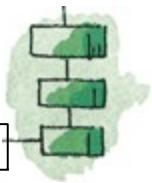
$= T1 + (T1+T2)$

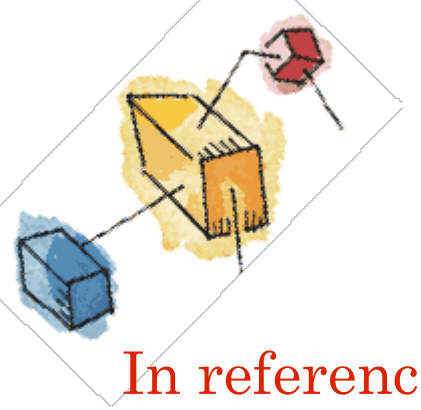
$= (0.95)(0.1\ \mu S) + (0.05)(0.1\ \mu S + 1\ \mu S)$

$= 0.095 + 0.055$

$= 0.15\ \mu S$

** Closer to the access time of the faster memory (0.1µS)



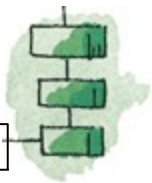


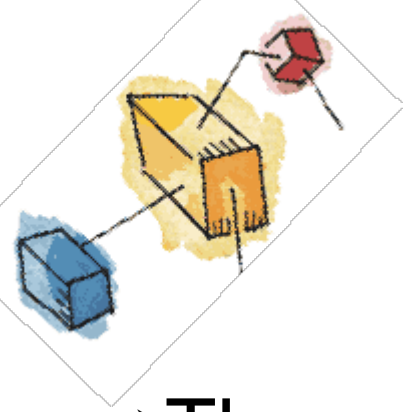
Principle of Locality

In reference to (d) Decreasing frequency of access to the memory by processor

- During program execution, memory references by the processor tend to cluster
  - Iterative Loops / subroutines
  - Repeated references to a small set of instructions

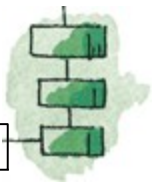
- Over long period → cluster may change,
- Over short period → fixed clusters of memory references
- (Internal registers of processor)
- Can be applied across more than two levels of memory





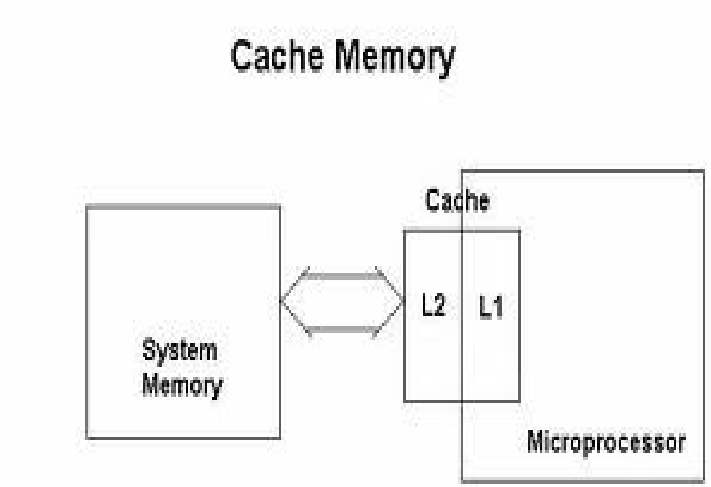
Secondary Memory

- The memory just described is volatile
- Data stored permanently in external storage
  - Hard disk
  - Removable media
- Auxiliary memory / secondary memory
- Nonvolatile
- Used to store program and data files

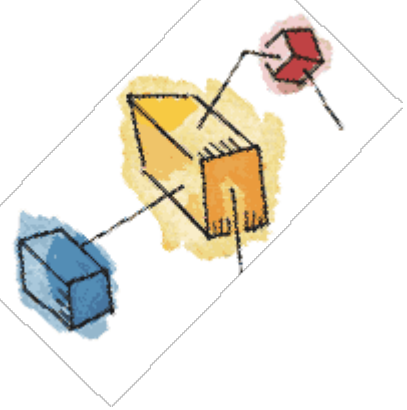


Cache Memory

- Special high-speed storage
- A portion of main memory use as a buffer



- Main memory is slow! (GHz vs MHz)
- To cope with processor speed
  - Persistent mismatch between processor & main memory
- Exploit the principle of locality with a small fast memory, clustered



Cache and Main Memory

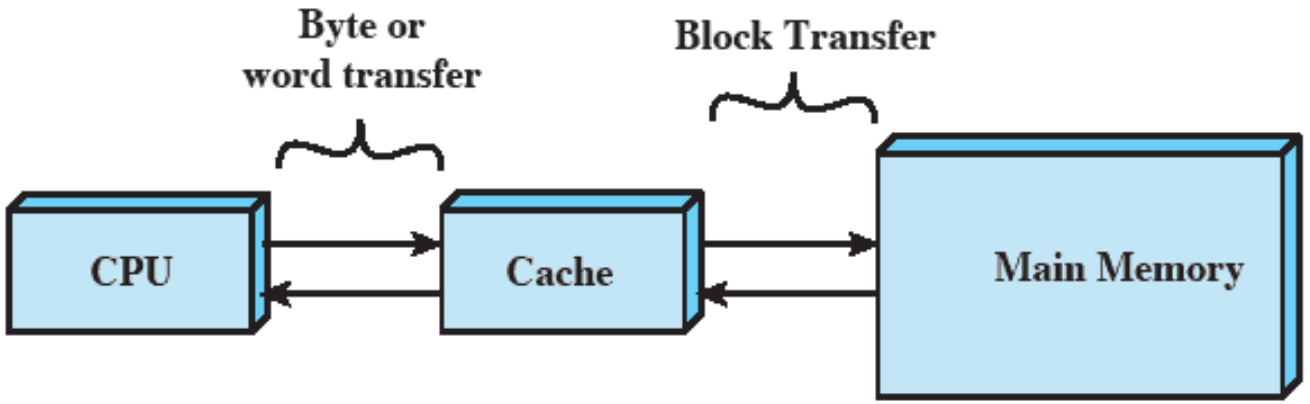
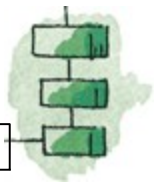
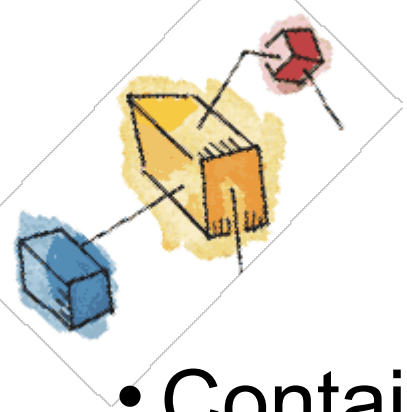


Figure 1.16 Cache and Main Memory



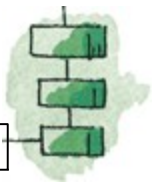


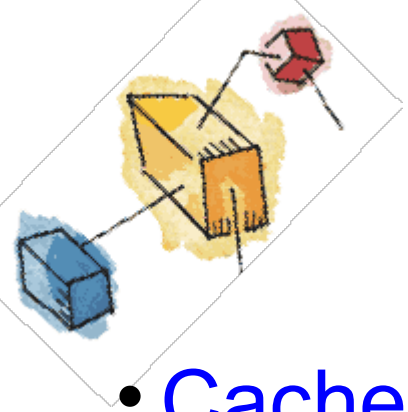
Cache Principles

- Contains copy of a portion of main memory
- Processor first checks the cache

- If the byte is not found, block of memory read into cache → transfer to processor

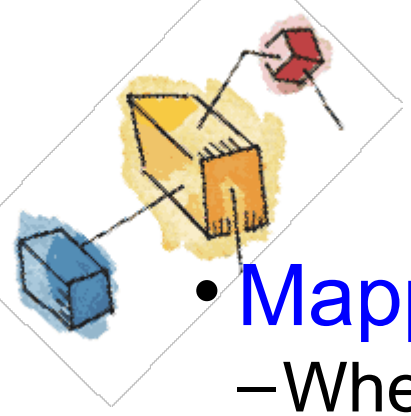
- Because of locality of reference, likely future memory references are in that block





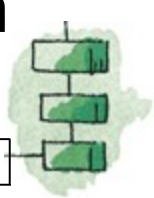
Cache Design

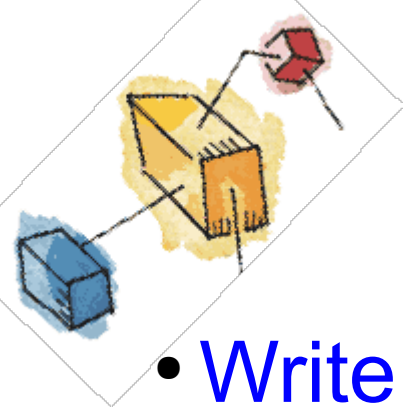
- Cache size
  - Small caches have significant impact on performance
- Block size
  - The unit of data exchanged between cache and main memory
  - Larger block size → more hits until probability of using newly fetched data becomes less than the probability of reusing data that have to be moved out of cache to make room for new block



Cache Design

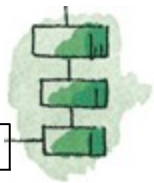
- Mapping function
  - When a new block of data is read
  - Determines which cache location the block will occupy → which block to be replaced
- Replacement algorithm
  - When one block is read, another may have to be replaced
  - Chooses which block to replace
  - Minimize probability of replacing the block that is needed in the near future
  - Least-recently-used (LRU) algorithm – longest in cache, not referenced

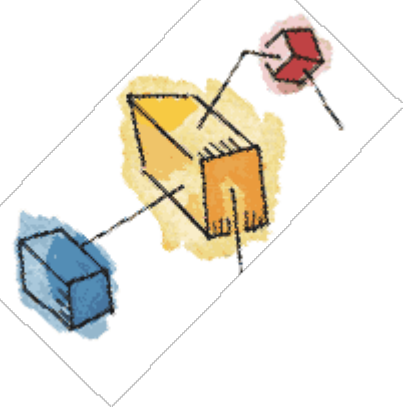




Cache Design

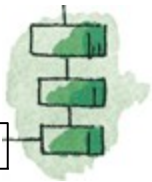
- Write policy
  - If the content of block in cache is altered, write to memory
  - Dictates when the memory write operation takes place
  - Can occur every time the block is updated
  - Can occur when the block is replaced, but make memory obsolete
  - Minimize write operations
  - Leave main memory in an obsolete state
- Number of cache Levels
  - Level 1 (closest to processor), Level 2, Level 3





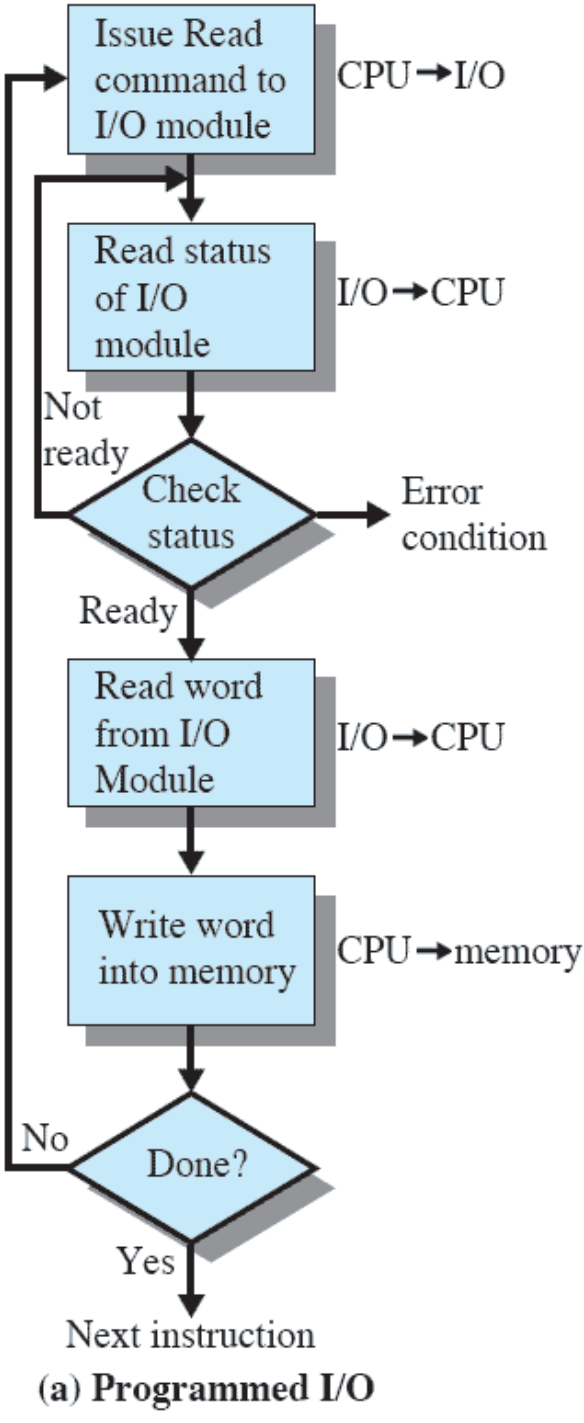
I/O Communication Techniques

- Three techniques possible for I/O operations
  - Programmed I/O
  - Interrupt-driven I/O
  - Direct Memory Access (DMA)

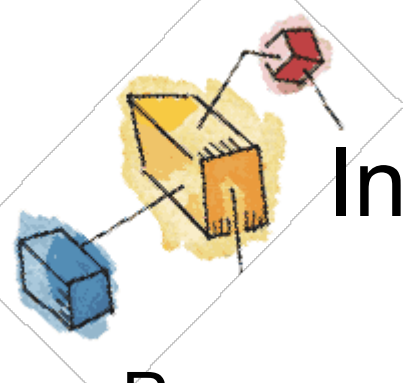


Programmed I/O

- When processor encounters an instruction relating to I/O, it execute the instruction:
  - I/O module performs the requested action
  - Sets the appropriate bits in the I/O status register
    - But take no action to alert the processor (No interrupts)
  - Thus, processor periodically checks status until operation is complete
    - Processor busy, degrade performance

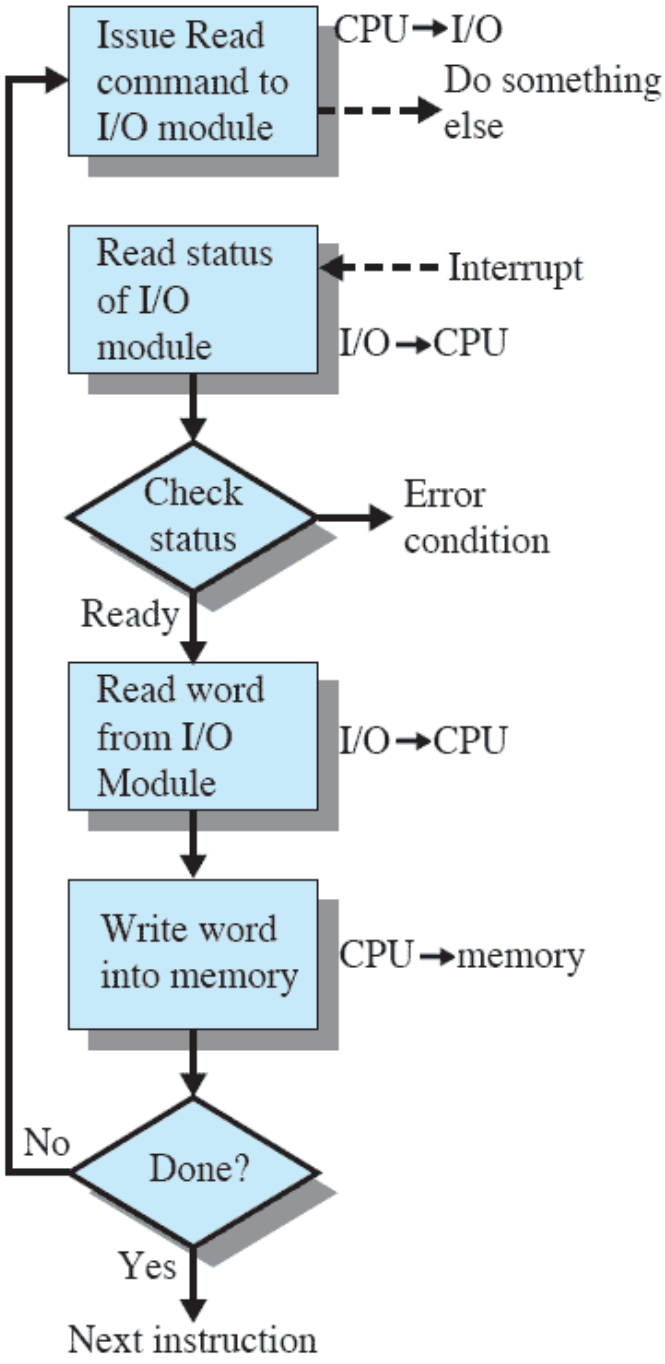


(a) Programmed I/O

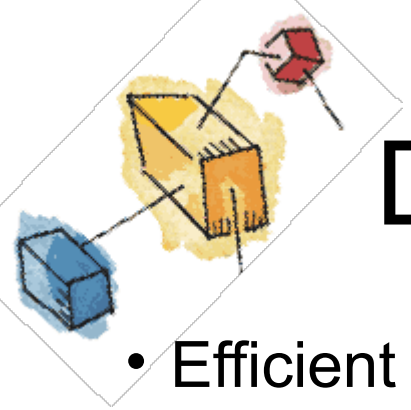


Interrupt-Driven I/O

- Processor is interrupted when I/O module ready to exchange data
- Processor saves context of program executing and begins executing interrupt-handler
- No needless waiting
- Still consumes a lot of processor time because every word read or written passes through the processor

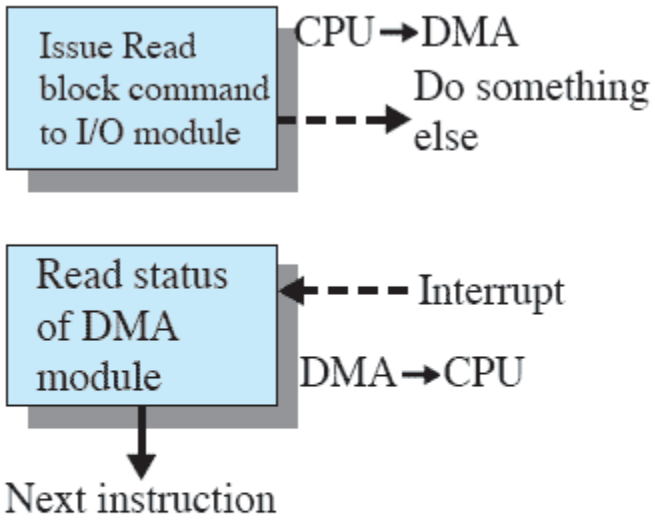


(b) Interrupt-driven I/O

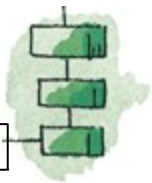


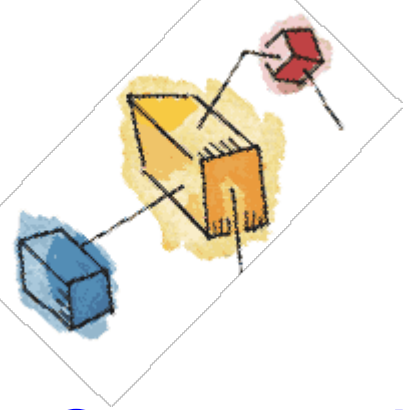
Direct Memory Access

- Efficient when large transfer of data are to be moved
- Processor issue direct command to DMA module, DMA will take care
- Processor then continue with other work
- Transfers a block of data directly to or from memory
- An interrupt is sent when the transfer is complete
- Processor involves only beginning & end



(c) Direct memory access

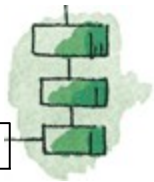


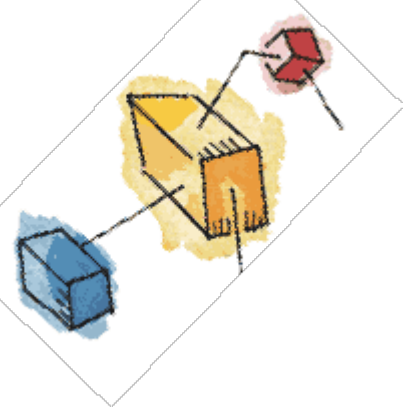


MultiProcessor & Multicore Organization:

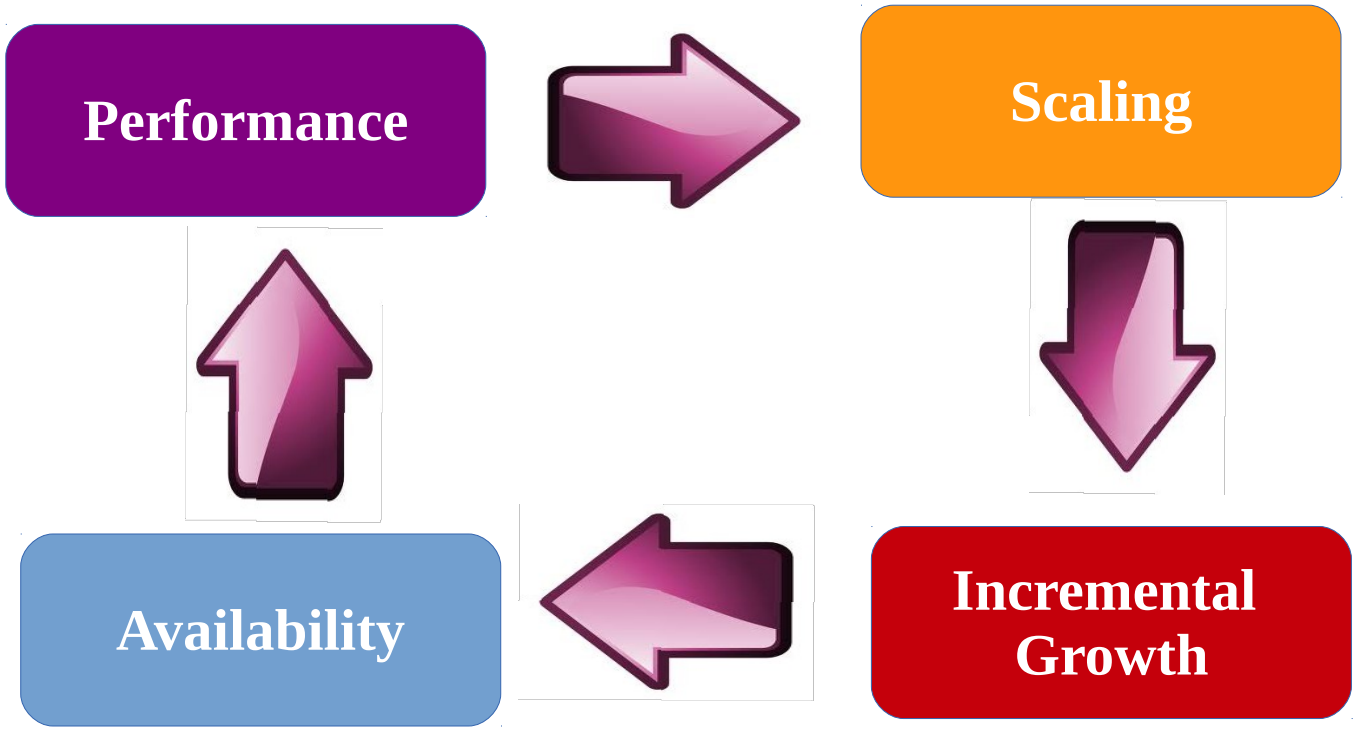
Symmetric Multiprocessors (SMP)

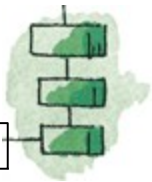
- A stand-alone computer system with the following characteristics:
  - two or more similar processors of comparable capability
  - processors share the same main memory and are interconnected by a bus or other internal connection scheme
  - processors share access to I/O devices
  - all processors can 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

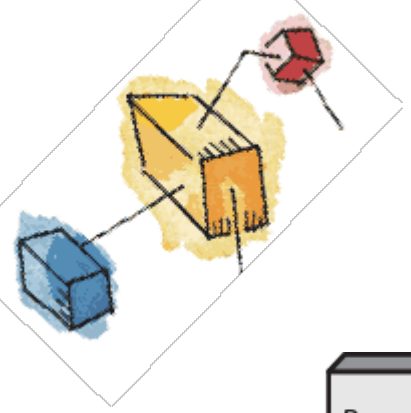




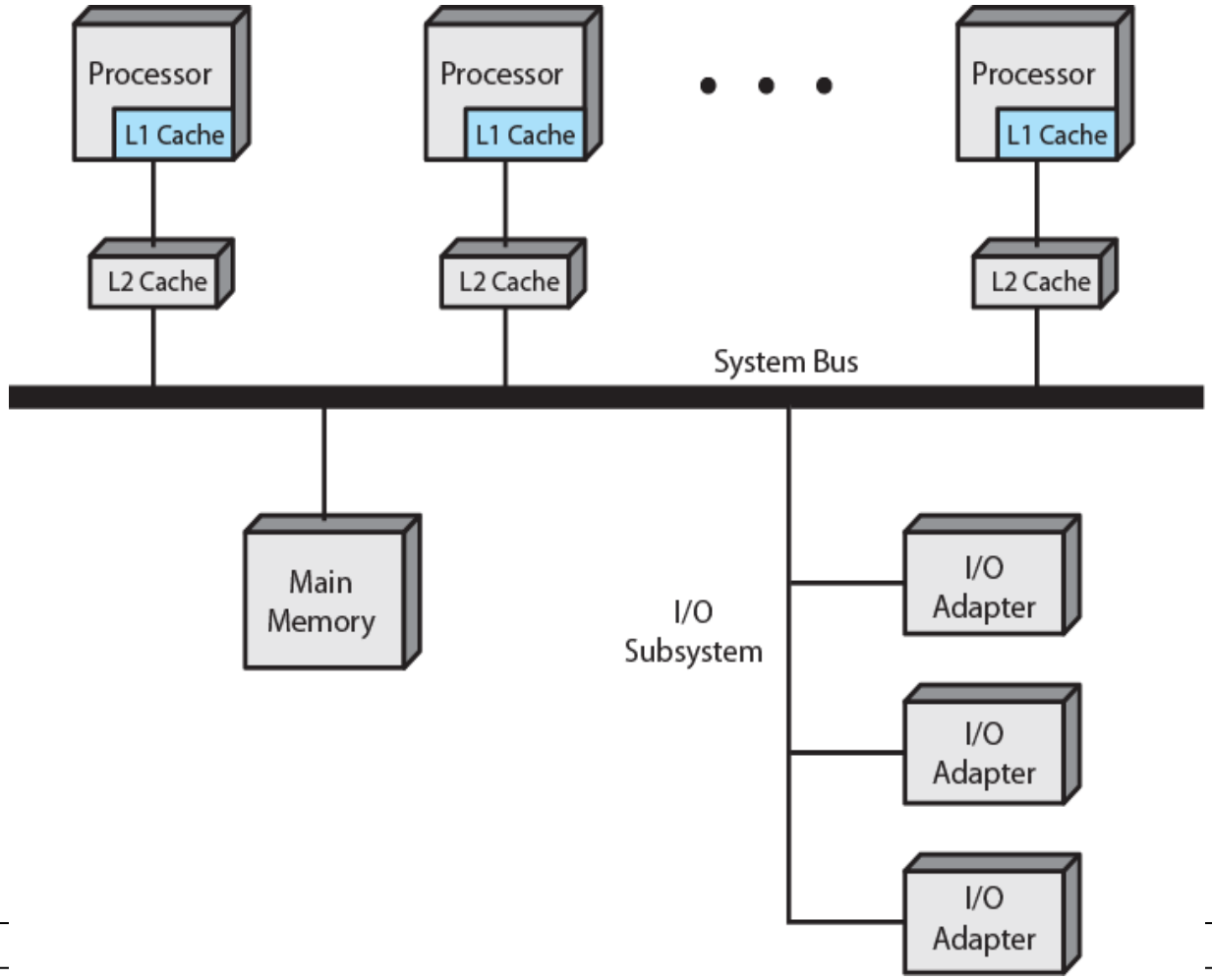
SMP Advantages

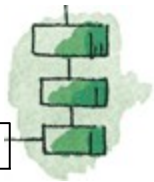


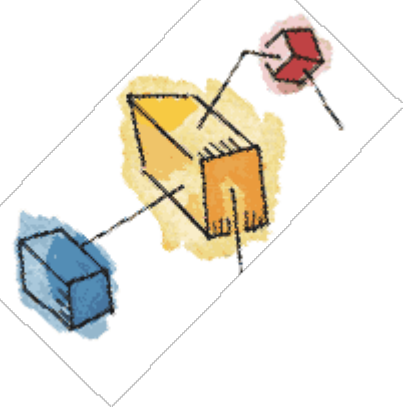




SMP Organization

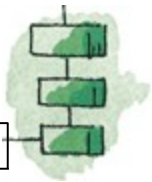


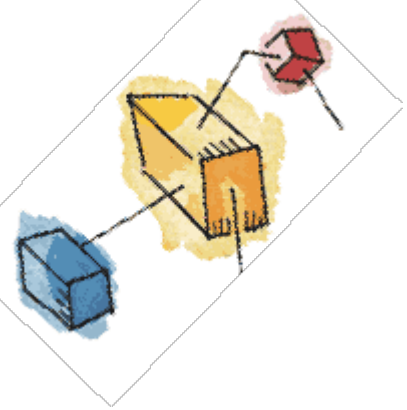




MultiCore Computer

- 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





Intel Core i7

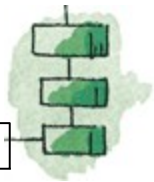
Supports two forms of external communications to other chips:

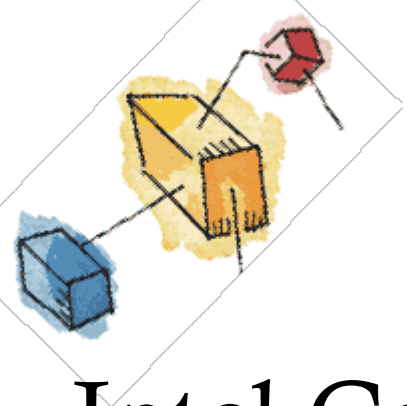
- DDR3 Memory Controller

Brings the memory controller for the DDR (double data rate) main memory onto the chip with the memory controller on the chip the Front Side Bus is eliminated

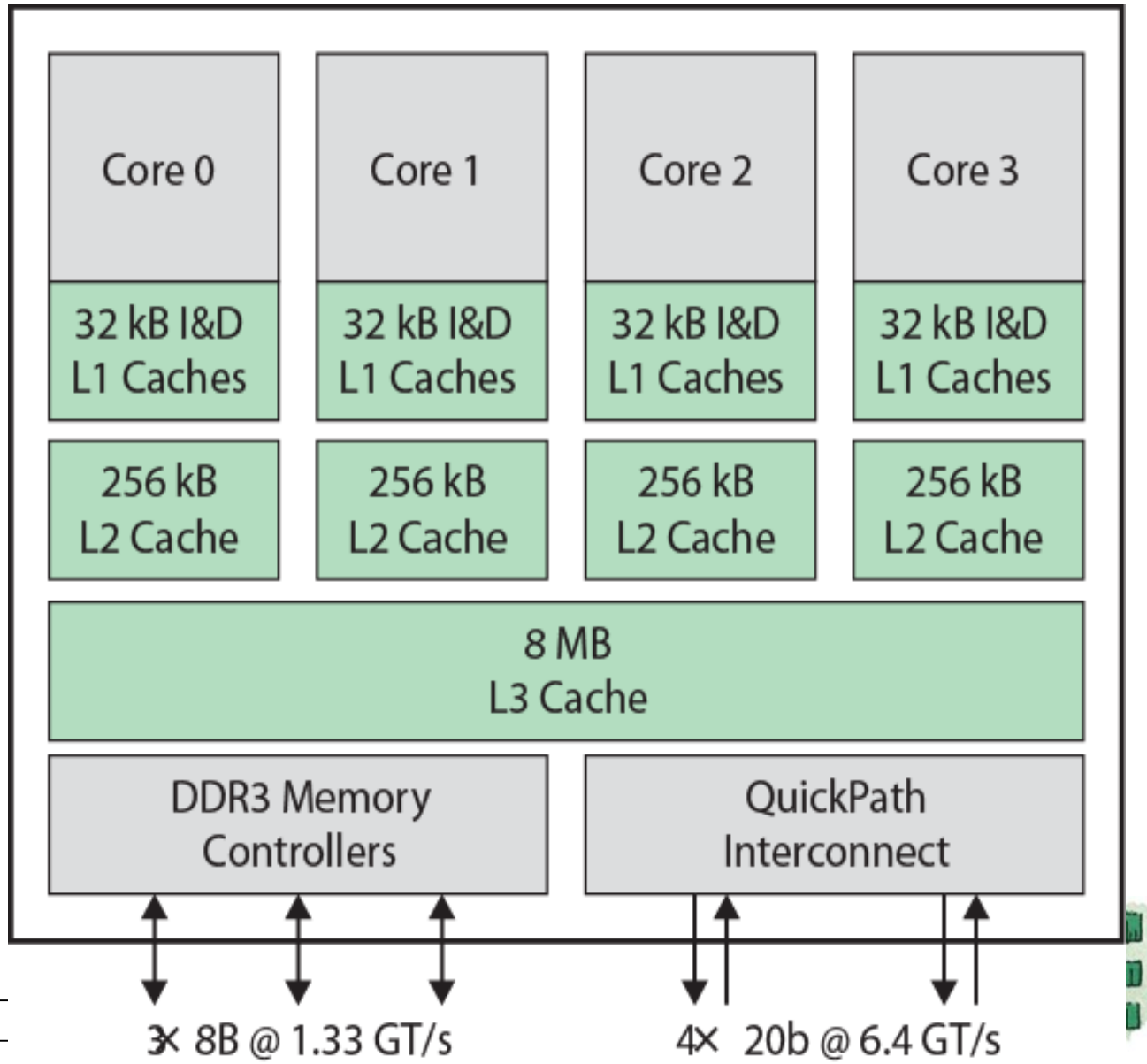
- QuickPath Interconnect (QPI)

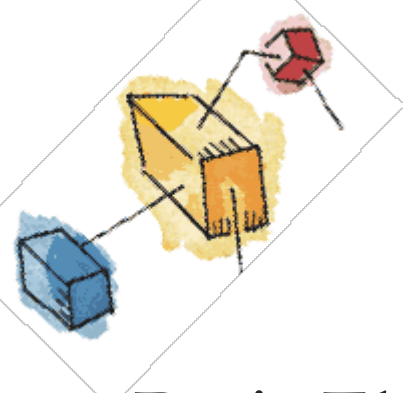
Enables high-speed communications among connected processor chips





Intel Core i7





Summary

- Basic Elements
  - processor, main memory, I/O modules, system bus
  - GPUs, SIMD, DSPs, SoC
  - Instruction execution
    - processor-memory, processor-I/O, data processing, control
  - Interrupt/Interrupt Processing
  - Memory Hierarchy
  - Cache/cache principles and designs
  - Multiprocessor/multicore
  - Command Lines