Lecture 02 Computer Hardware Review

Dr. Tushar, Mosaddek Hossain Kamal
Professor

Computer Science and Engineering, University of Dhaka,

BSc Third Year, Semester 2 (July – Dec), Academic Year: 2022

CSE3201: Operating Systems

July 21, 2022

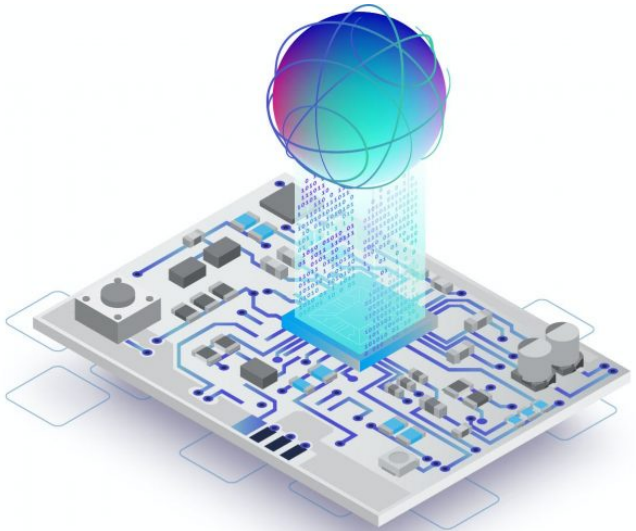
Outline

Learning Outcome

- Understand the basic components of computer hardware
  - CPU, buses, memory, devices controllers, DMA, Interrupts, hard disks
- Understand the concepts of memory hierarchy and caching, and how they affect performance.

System Software or Program

- 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 Computer Elements

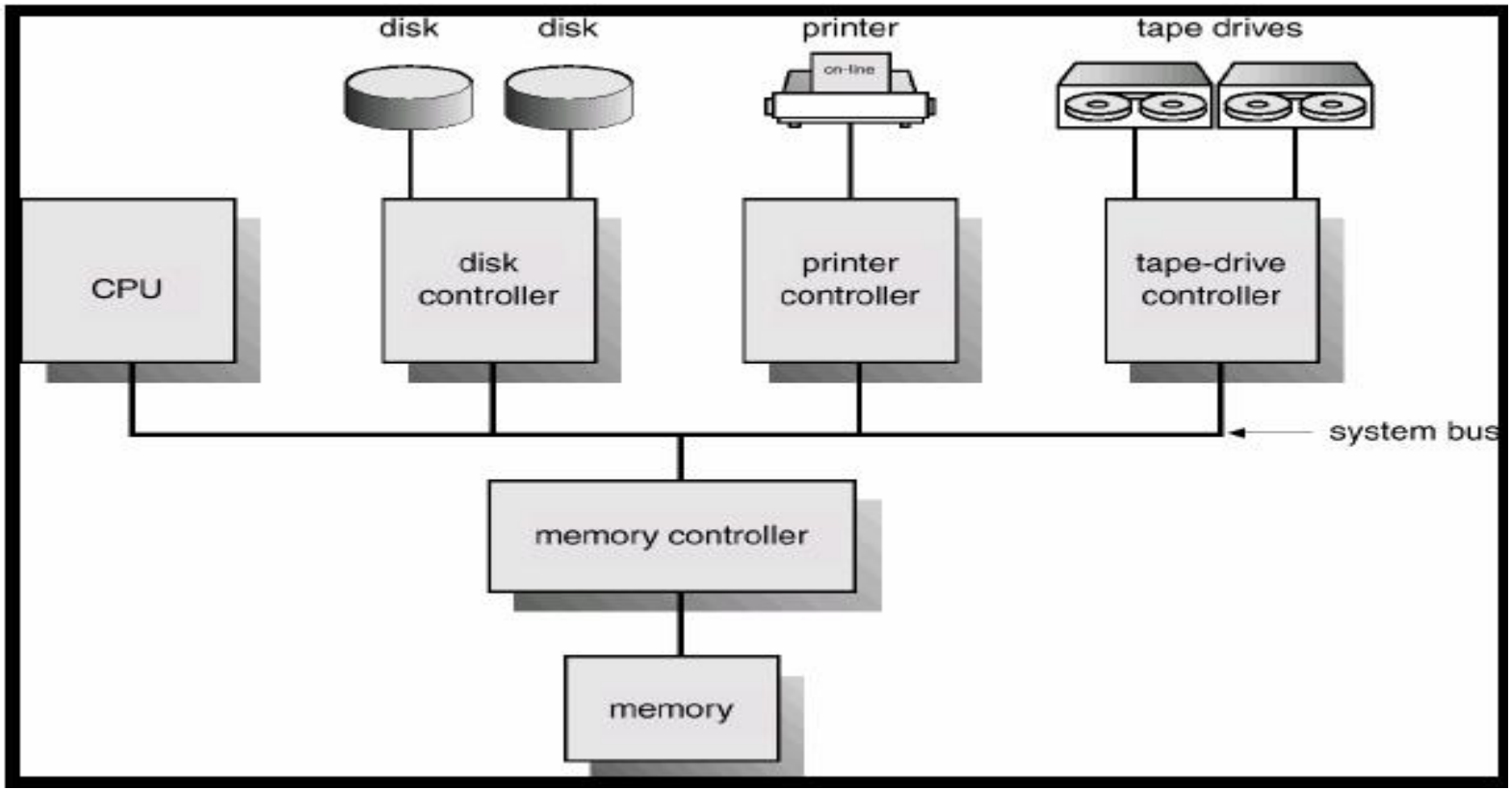


Figure 1: Basic Computer Elements

Basic Computer Elements

CPU

- Performs computations
- Load data to/from memory via system bus

Device controllers

- Control operation of their particular device
- Operate in parallel with CPU
- Can also load/store to memory (Direct Memory Access, DMA)
- Control register appear as memory locations to CPU
  - Or I/O ports
- Signal the CPU with "interrupts"

Memory Controller

- Responsible for refreshing dynamic RAM
- Arbitrating access between different devices and CPU

Basic Computer Elements

The real world is logically similar, but a little more complex

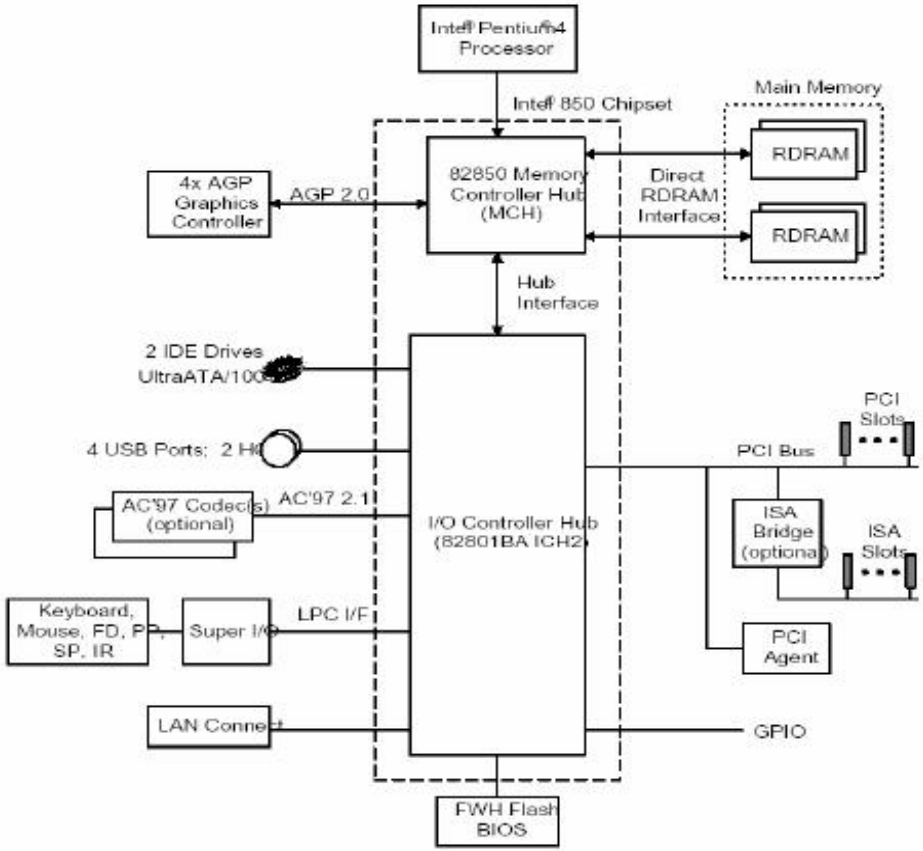


Figure 2: Computer Elements – real world view

Basic Computer Elements

The real world is logically similar, but a little more complex

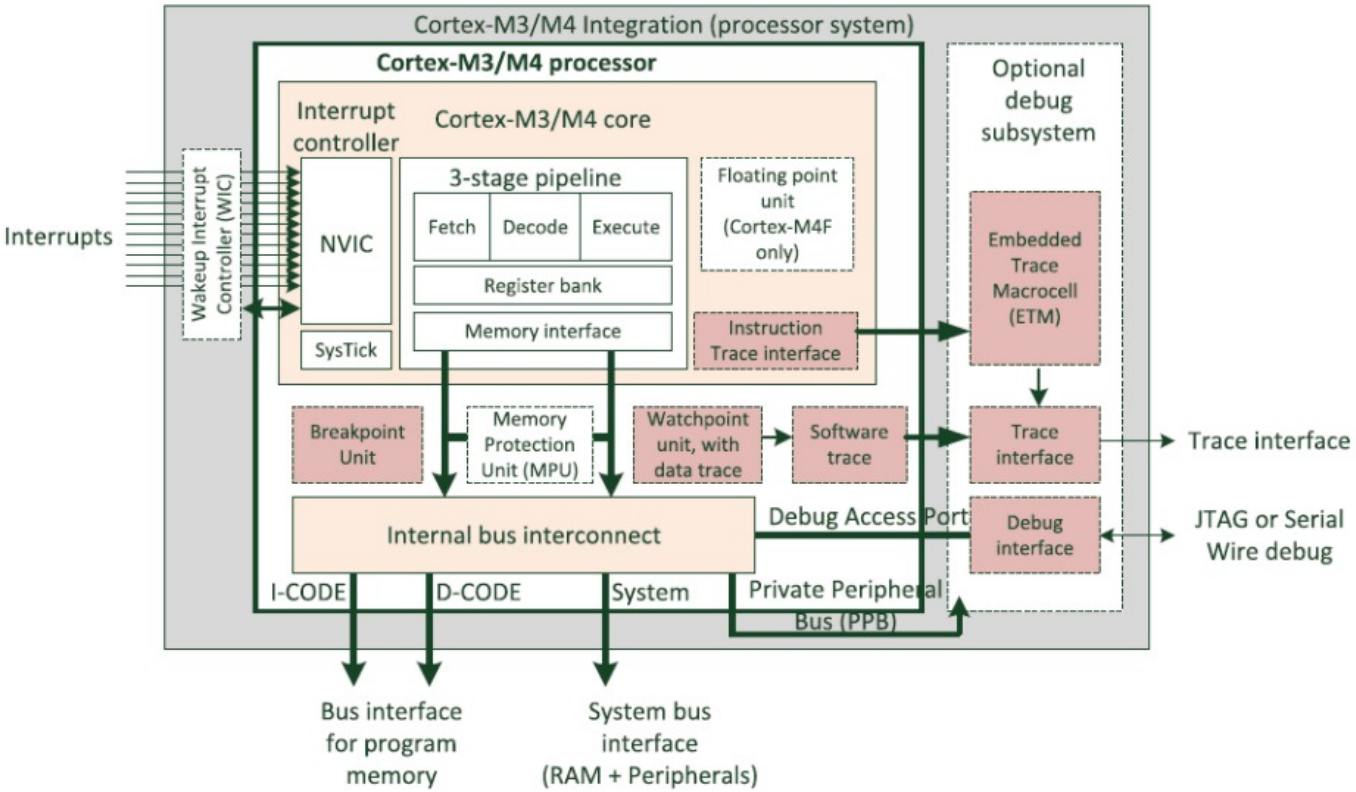


Figure 3: STM32 – real world view

A Simple Model of CPU Computation

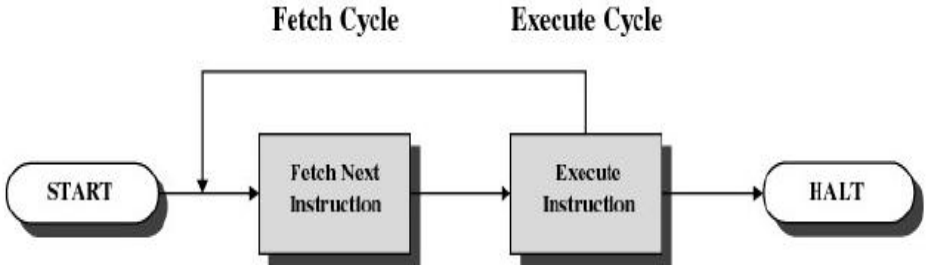


Figure 4: Simple Fetch Cycle

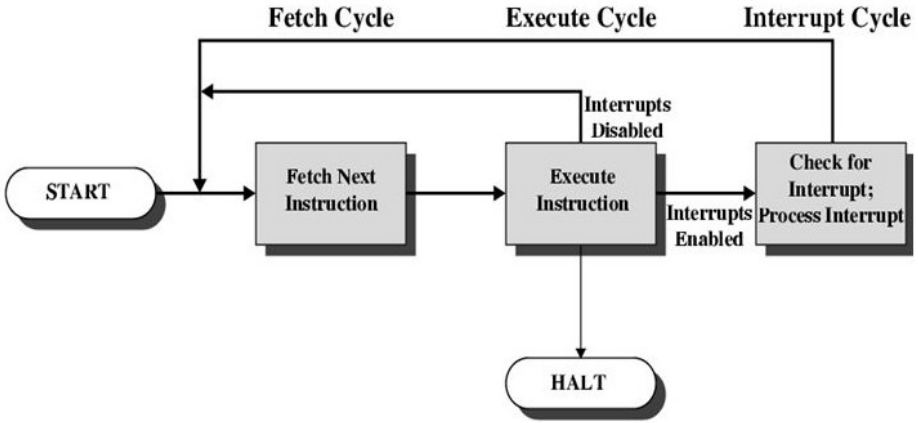


Figure 5: Simple Fetch Cycle with Interrupt

A Simple Model of CPU Computation

- Program Counter (PC)
- Stack Pointer
- Status Register
  - Condition codes
    - Positive result
    - Zero result
    - Negative result
- General Purpose Registers
  - Holds operands of most instructions
  - Enables programmers to minimize memory references.

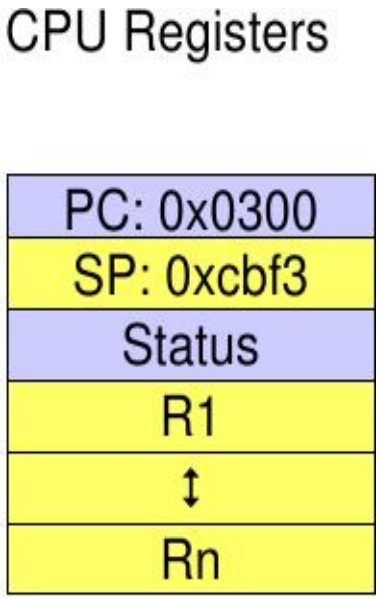


Figure 6: CPU registers

A Simple Model of CPU Computation

The fetch-execute cycle

- Load memory contents from address in program counter(PC)
  - The instruction
- Execute the instruction
  - Decode instruction
  - Encode machine cycle
  - Generate control signal
- Increment PC
- Repeat

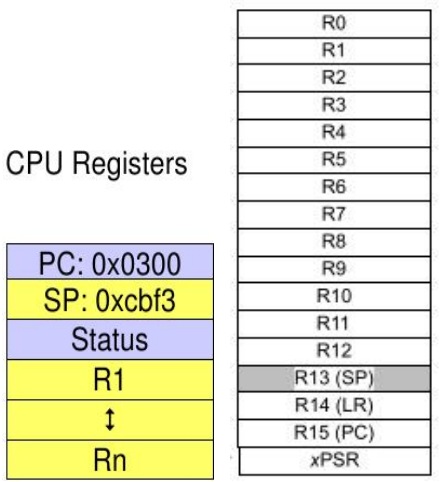


Figure 7: CPU registers

Privileged-mode Operation

To protect operating system execution, two or more CPU modes of operation exist

- Privileged mode (system-kernel-mode)
  - All instructions and registers are available
- User-mode
  - Uses 'safe' subset of the instruction set
    - E.g. no disable interrupts instruction
      Only 'safe' registers are accessible

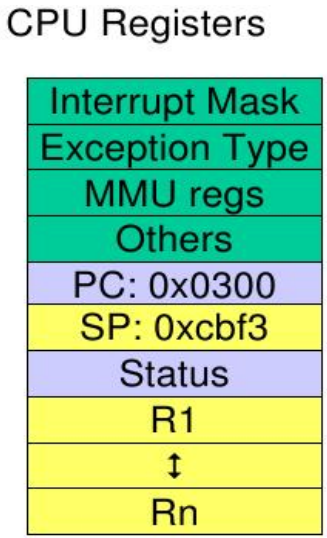


Figure 8: CPU registers

‘Safe’ registers and instructions

- Registers and instructions are safe if
  - Only affect the state of the application itself
  - They cannot be used to uncontrollably interfere with
    - The operating system
    - Other applications
  - They cannot be used to violate a correctly implemented operating system policy.

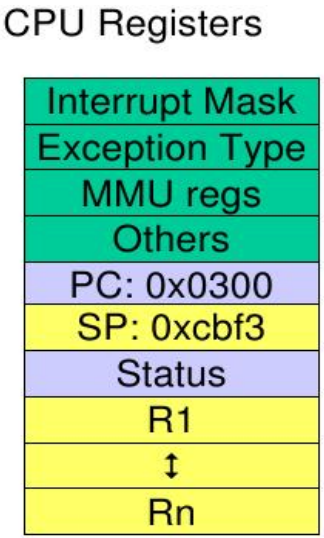


Figure 9: CPU registers

Privileged-mode Operation

The accessibility of addresses within an address space changes depending on operating mode

- To protect kernel code and data

Mode	Mode Identifier
User	usr
*Fast Interrupt	fiq
*Supervisor	svc
*Abort	abt
*System	sys
*undefined	und

Table 1: Arm Processor Operating Mode

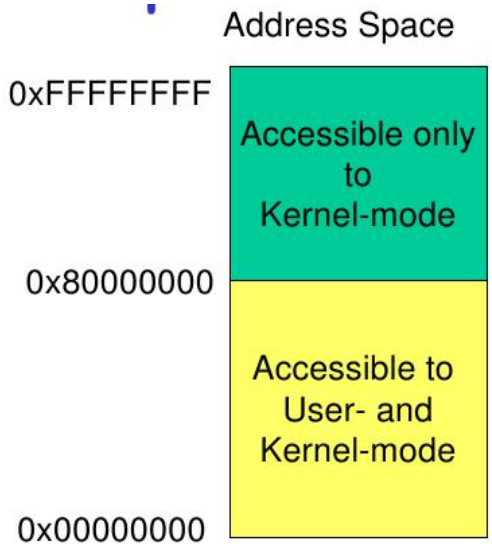


Figure 10: Microprocessor address space access

*Similar to kernel mode or handled by the kernel

ARM – Entering System mode from another privileged mode by modifying the mode bit of the Current Program Status Register (CPSR).

I/O and Interrupt

I/O and Interrupts

- I/O events (keyboard, mouse, incoming network packets) happen at unpredictable times
- How does the CPU know when to service an I/O event?

Interrupt

- An interruption of the normal sequence of execution
- A suspension of processing caused by an event external to that processing, and performed in such a way that the processing can be resumed.
- Improves processing efficiency
  - Allows the processor to execute other instructions while an I/O operation is in progress
  - Avoids unnecessary completion checking (polling)

Interrupt Cycle

interrupt cycle

- Processor checks for interrupts
- If no interrupts, fetch the next instruction
- If an interrupt is pending, divert to the interrupt handler

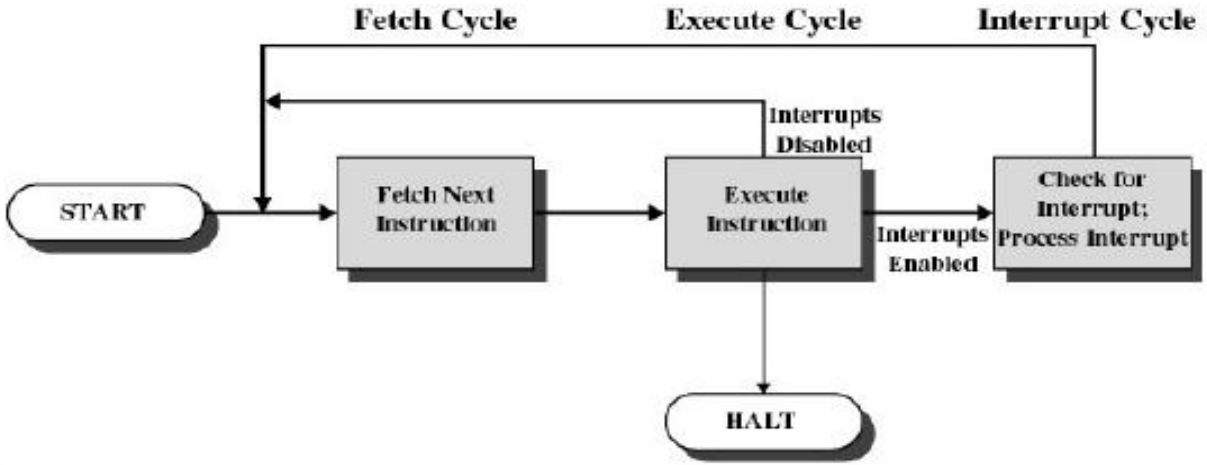


Figure 11: Interrupt Life Cycle

Classes of Interrupts

Program exceptions (also called synchronous interrupts)

- Arithmetic overflow
- Division by zero
- Executing an illegal/privileged instruction
- Reference outside user's memory space.

Asynchronous (external) events

- Timer
- Input and output – I/O
- Hardware or power failure

Interrupt Handler

Interrupt Routine

- A program that determines the nature of the interrupt and performs whatever actions are needed.
- Control is transferred to the handler by hardware.
  - Save a few registers values – PC, SP, LR and $\cdots$
- The handler is generally part of the operating system.
- register values restored before on return

Single or Simple Interrupt

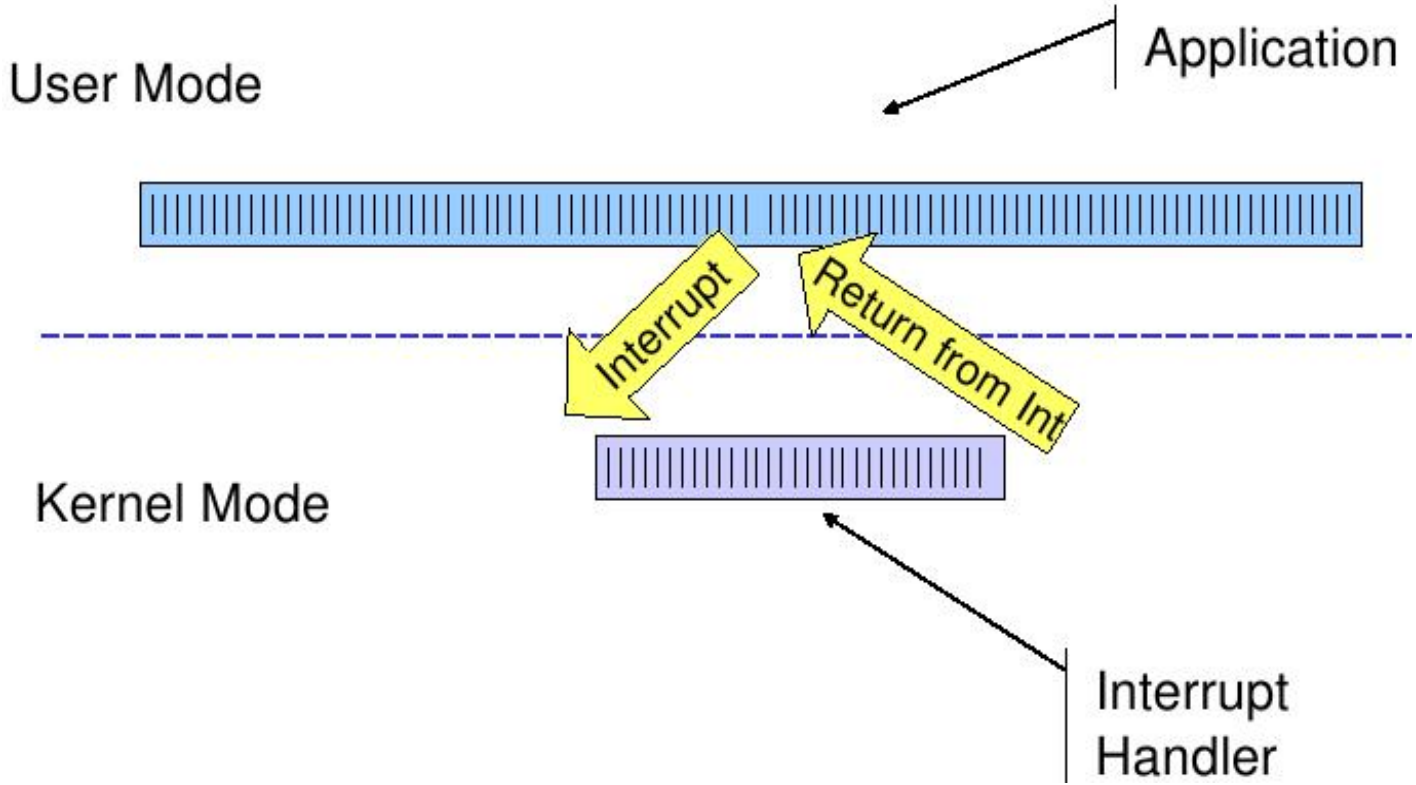


Figure 12: Simple interrupt handling

Simple Interrupt Processing

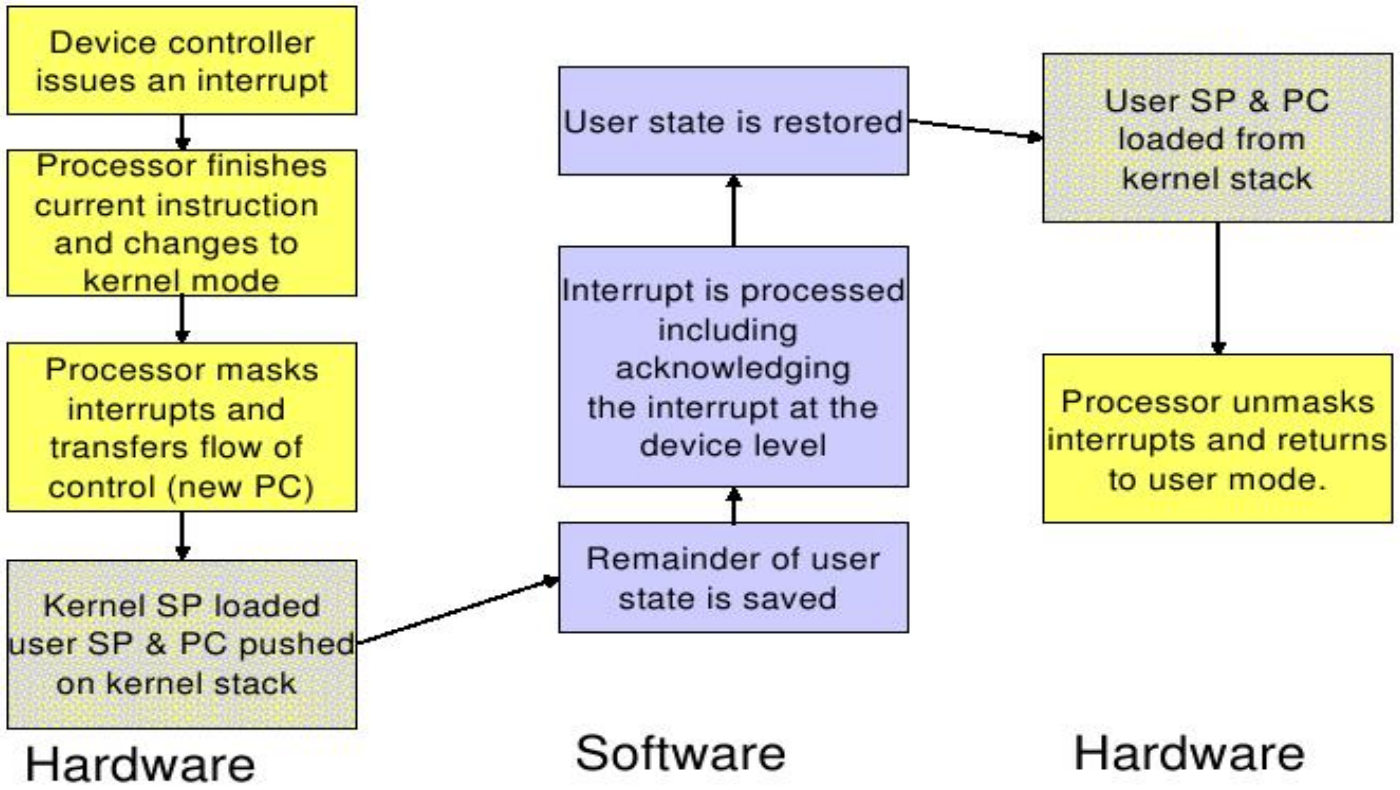


Figure 13: Interrupt Processing

Multiple Interrupts

Sequential interrupts

- Processor ignores any new interrupt signals
- Interrupts remain pending until current interrupt completes
- Upon completion, processor checks for additional interrupts

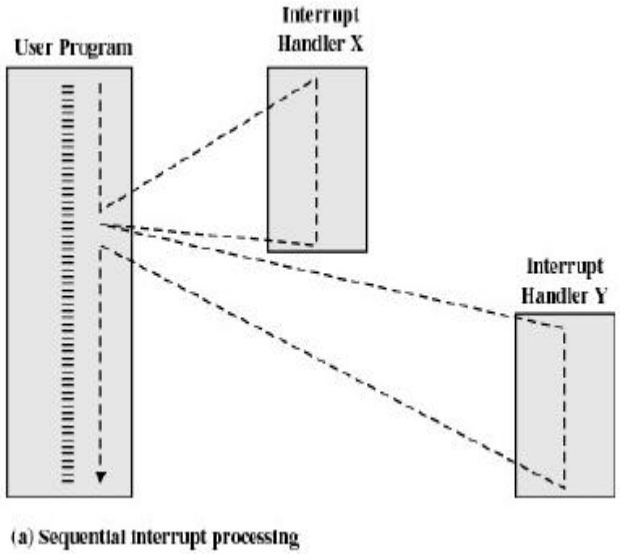


Figure 14: Sequencial Interrupt

Multiple Interrupts

Prioritized (nested) interrupts

- Processor ignores any new lower-priority interrupt signals
- New higher-priority interrupts interrupt the current interrupt handler
- Example: when input arrive from a communication line, it needs to be absorbed quickly to make room for more input

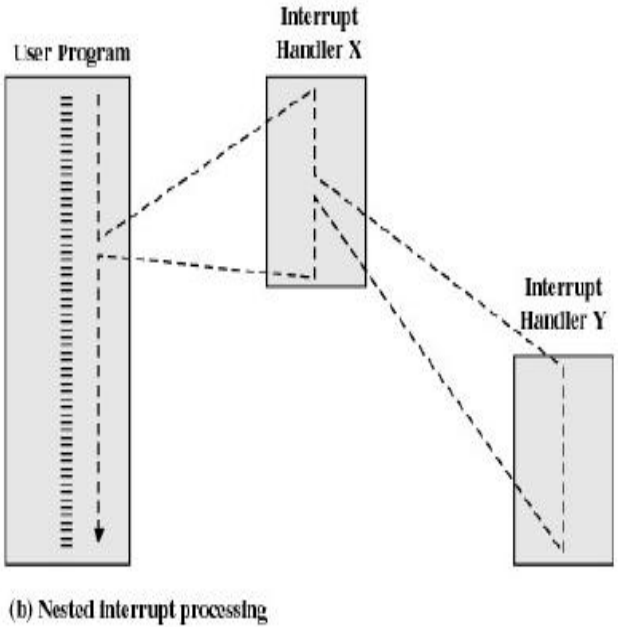
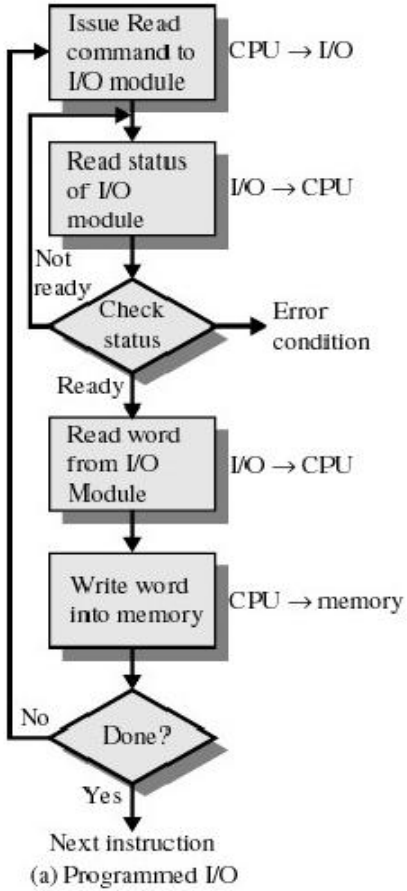


Figure 15: Nested Interrupt

Programmed I/O

- Also called polling, or busy waiting
- I/O module (controller)performs the action, not the processor
- Sets appropriate bits in the I/O status register
- No interrupts occur
- Processor checks status until operation is complete
  - Wastes CPU cycles

Programmed I/O



(a) Programmed I/O

Interrupt-Driven I/O

- Processor is interrupted when I/O module (controller) ready to exchange data
- Processor is free to do other work
- No needless waiting
- Consumes a lot of processor time because every word read or written passes through the processor

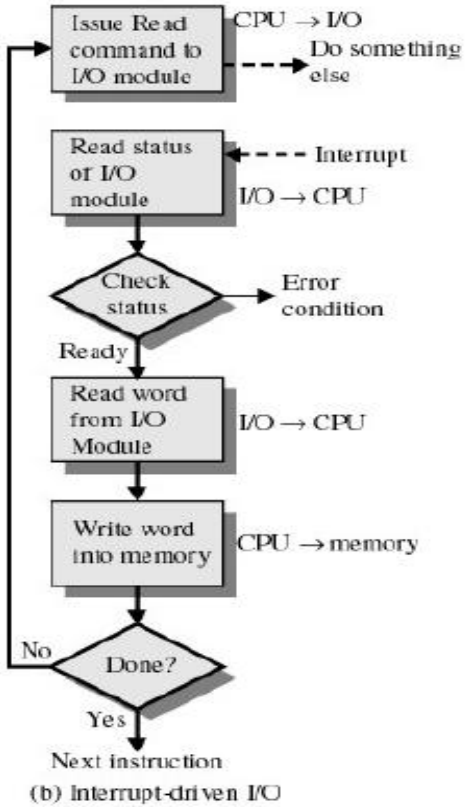


Figure 16: Interrupt driven I/O

Direct memory access (DMA)

- I/O exchanges occur directly with memory
- Processor directs I/O controller to read/write to memory
- Relieves the processor of the responsibility for data transfer
- Processor free to do other things
- An interrupt is sent when the task is complete

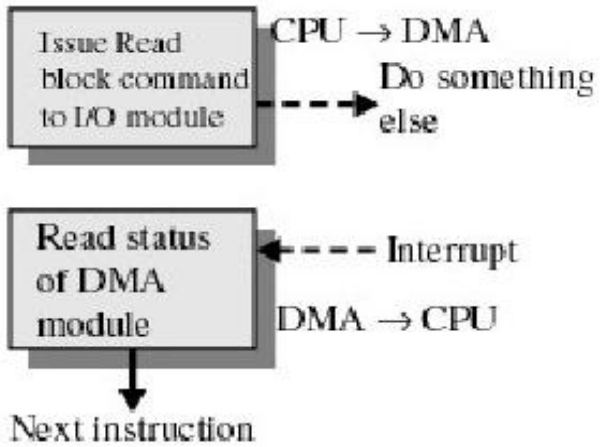


Figure 17: Drect Memory Access

Multiprogramming (Multitasking)

Multi-Tasking

- Processor has more that one program to execute.
  - Some tasks waiting for I/O to complete
  - Some tasks ready to run, but not running
- Interrupt handler can switch to other tasks when they become runnable
- Regular timer interrupts can be used for time sharing
- Example: nature – rain, sun , earth rotation, breezing .. at the same time !!!

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
  - Hopefully
  - Principle of locality!!!!!

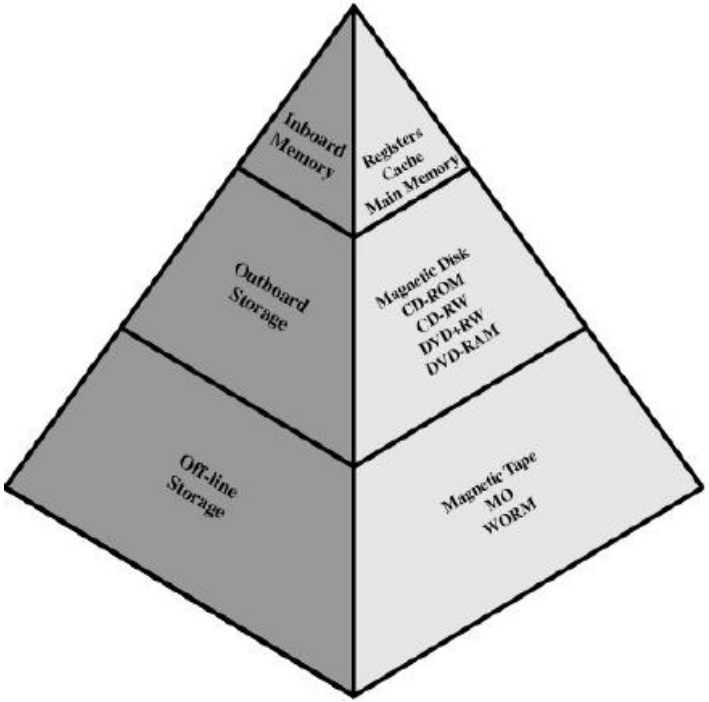
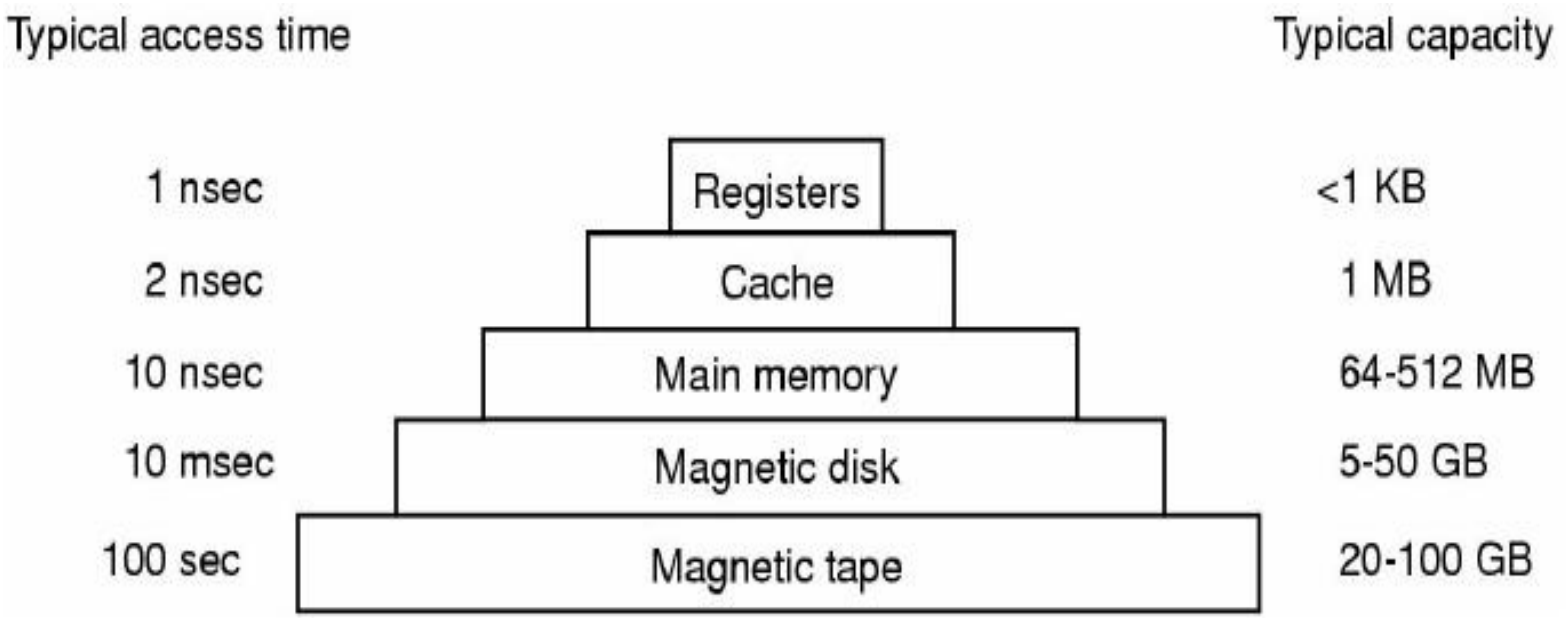


Figure 18: Memory hierarchy

Rough approximation of memory hierarchy



Cache Memory

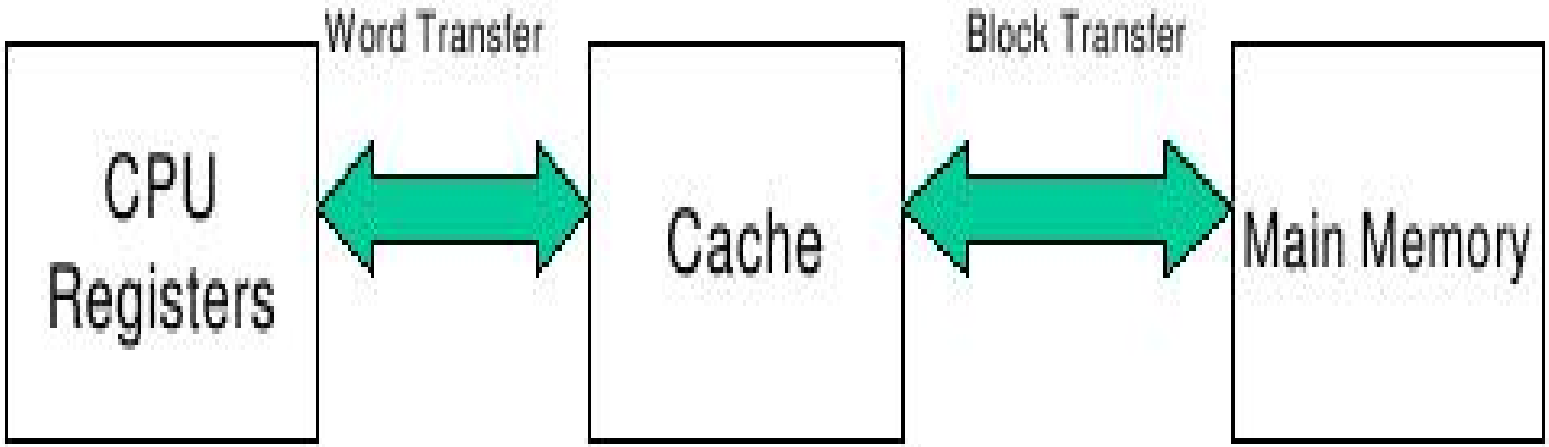


Figure 19: Cache Memory

Cache Memory

- Cache is fast memory placed between the CPU and main memory
  - 1 to a few cycles access time compared to RAM access time of tens – hundreds of cycles
- Holds recently used data or instructions to save memory accesses.
- Matches slow RAM access time to CPU speed if high hit rate
- Is hardware maintained and (mostly) transparent to software
- Sizes range from few kB to several MB.
- Usually a hierarchy of caches (2–5 levels), on-and off-chip.
- Block transfers can achieve higher transfer bandwidth than single words.
  - Also assumes probability of using newly fetch data is higher than the probability of reuse ejected data.

Processor-DRAM Gap (latency)

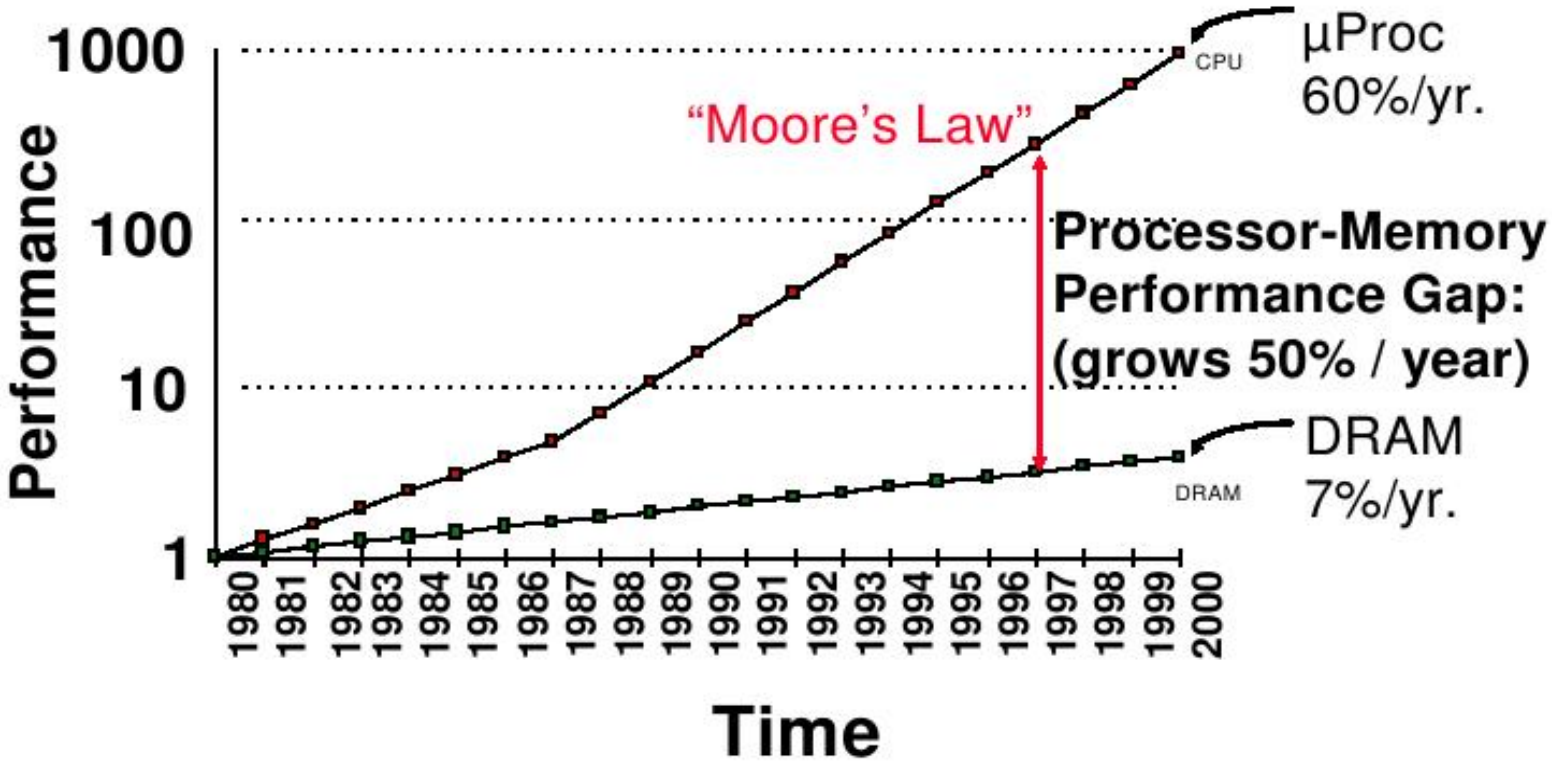


Figure 20: Processor-DRAM Gap

Cache size affect on performance

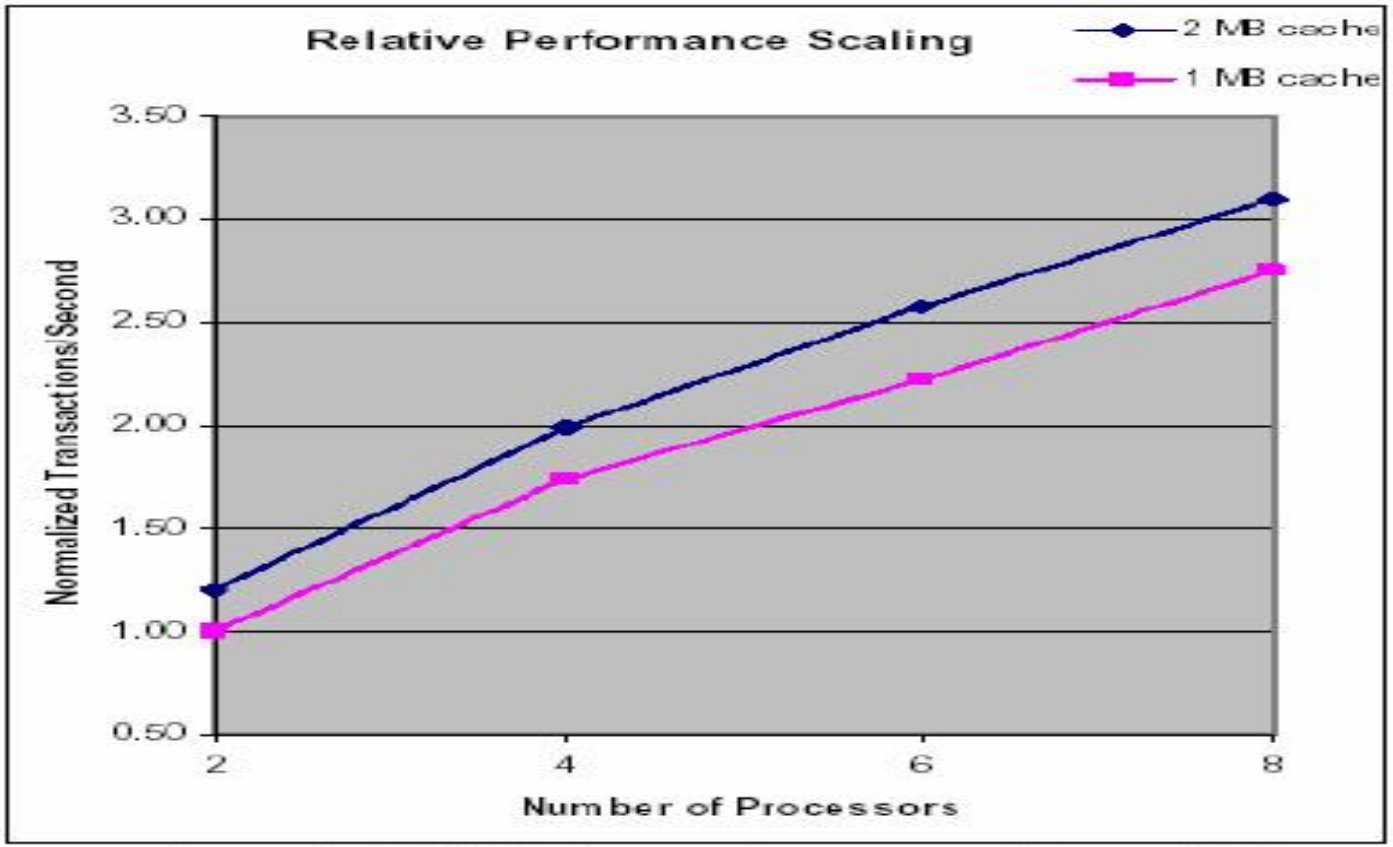


Figure 1 - OLTP Performance Improvement Between 1-MB and 2-MB Caches in a ProLiant 8500 Server

Figure 21: System performance on cache size

Moving-Head Disk Mechanism

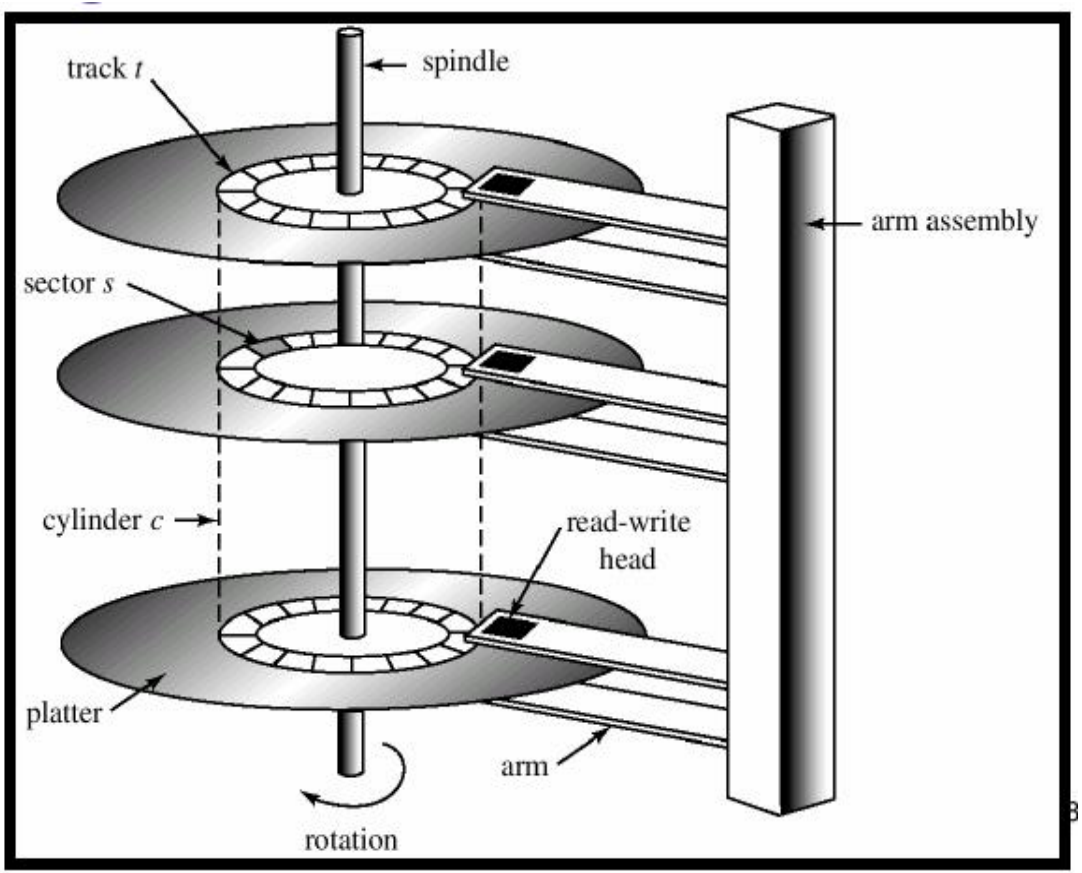


Figure 22: HDD head movement mechanism

Example Disk Access Times

Example

- Disk can read/write data relatively fast
  - 15,000 rpm drive - 80 MB/sec
  - 1 KB block is read in 12 microseconds
- Access time dominated by time to locate the head over data
  - Rotational latency
    - Half one rotation is 2 milliseconds
  - Seek time
    - Full inside to outside is 8 milliseconds
    - Track to track .5 milliseconds
- 2 milliseconds is 164KB in "lost bandwidth"

A Strategy: Avoid Waiting for Disk Access

- Keep a subset of the disk's data in memory
- Main memory acts as a cache of disk contents

Two-level Memories and Hit Rates

Hit Ratio

- Given a two-level memory,
  - cache memory and main memory (RAM)
  - main memory and disk

what is the effective access time?
Answer: It depends on the hit rate in the first level.

Effective Access Time

$$T_{eff} = H \times T_1 + (1 - H) \times (T_1 + T_2) \quad (1)$$

where,
T_1 = Access Time of memory1
T_2 = Access Time of memory2
H = hit ratio of memory1
T_{eff} = effective access time of the system

Example

Example

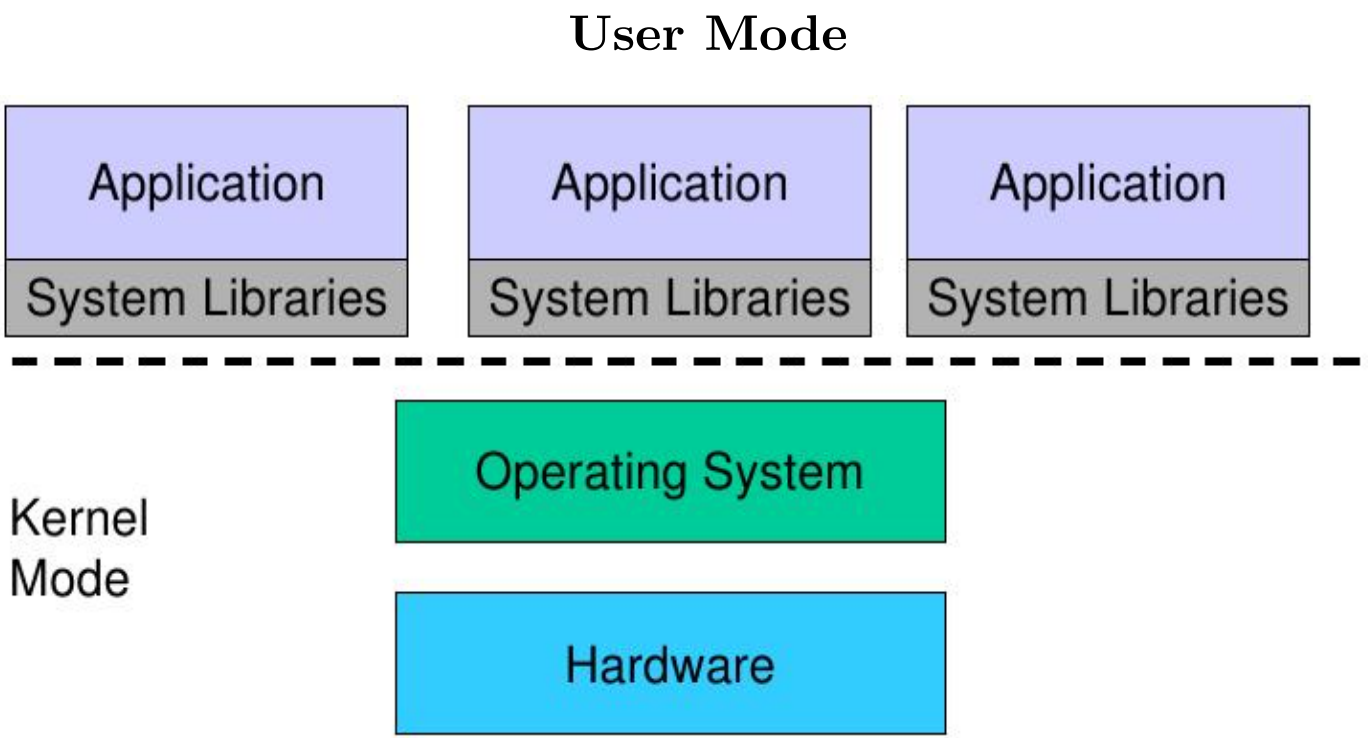
- Cache memory access time 1ns
- Main memory access time 10ns
- Hit rate of 95%

$$
\begin{aligned}
T_{eff} &= 0.95 \times 1 \times 10^{-9} + 0.05 \\
&\times (1 \times 10^{-9} + 10 \times 10^{-9} \\
&= 1.5 \times 10^{-9}
\end{aligned} \tag{2}
$$

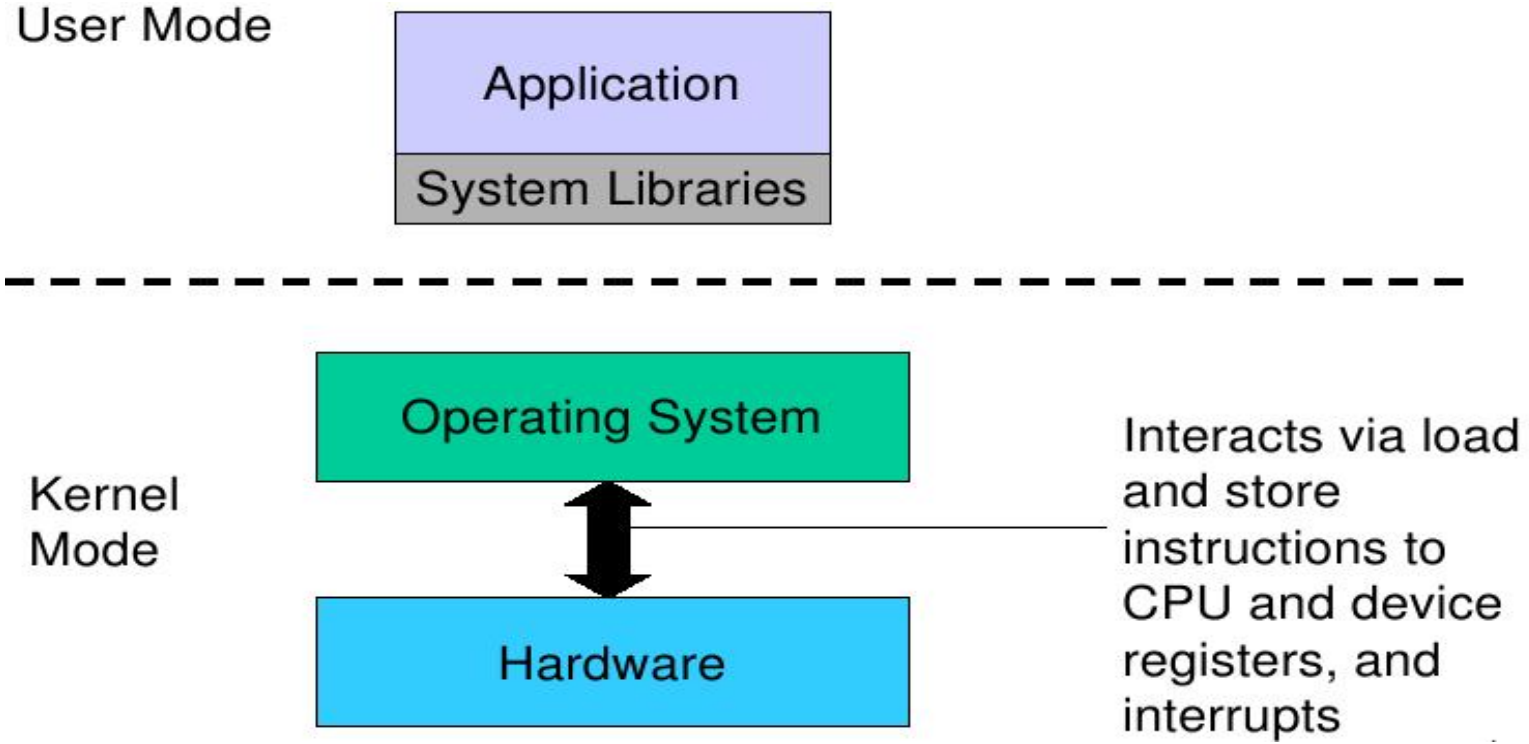
Operating System

- program that controls execution of applications
  - The resource manager
- An interface between applications and hardware
- The extended machine

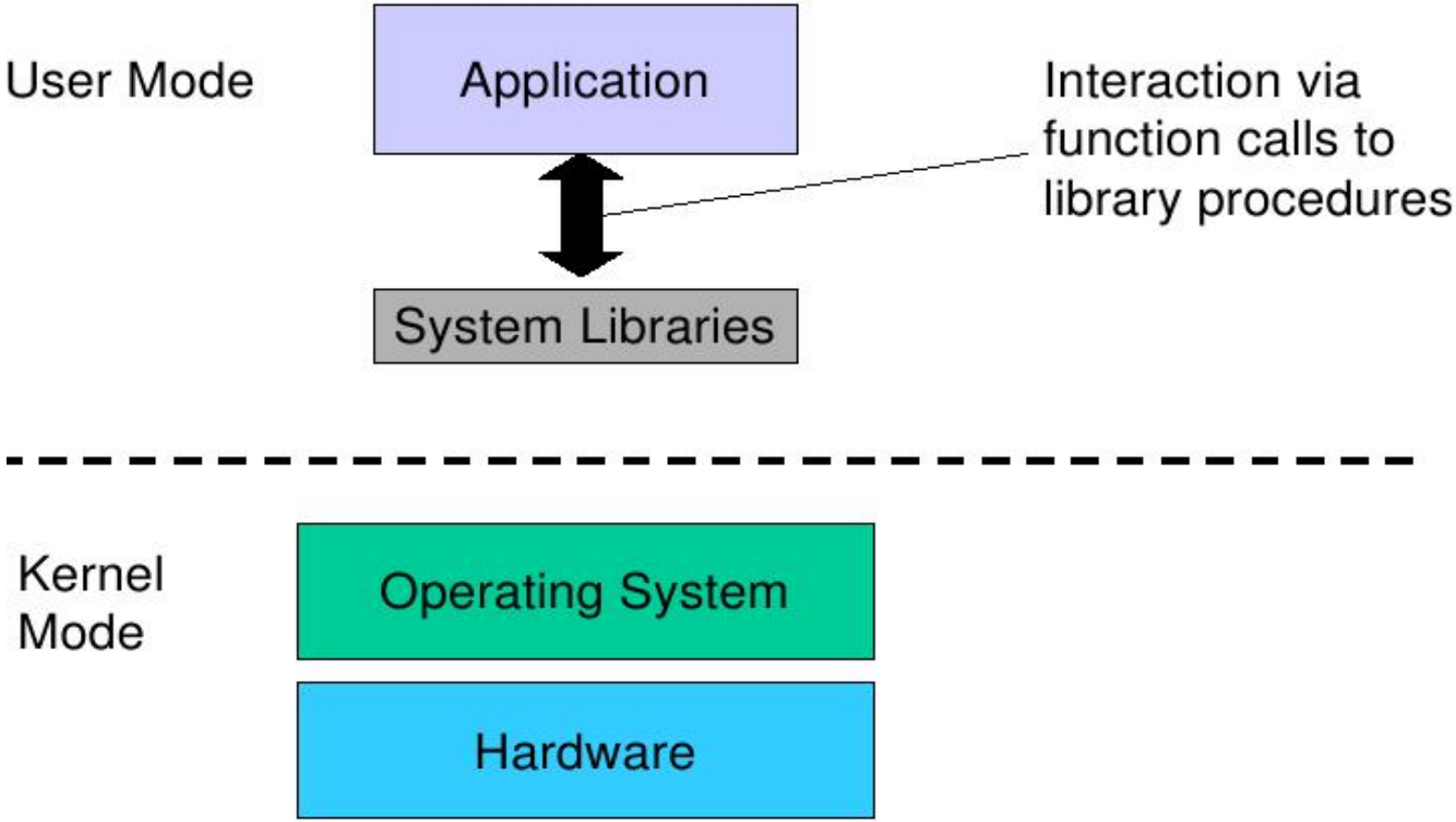
Structure of a Computer System



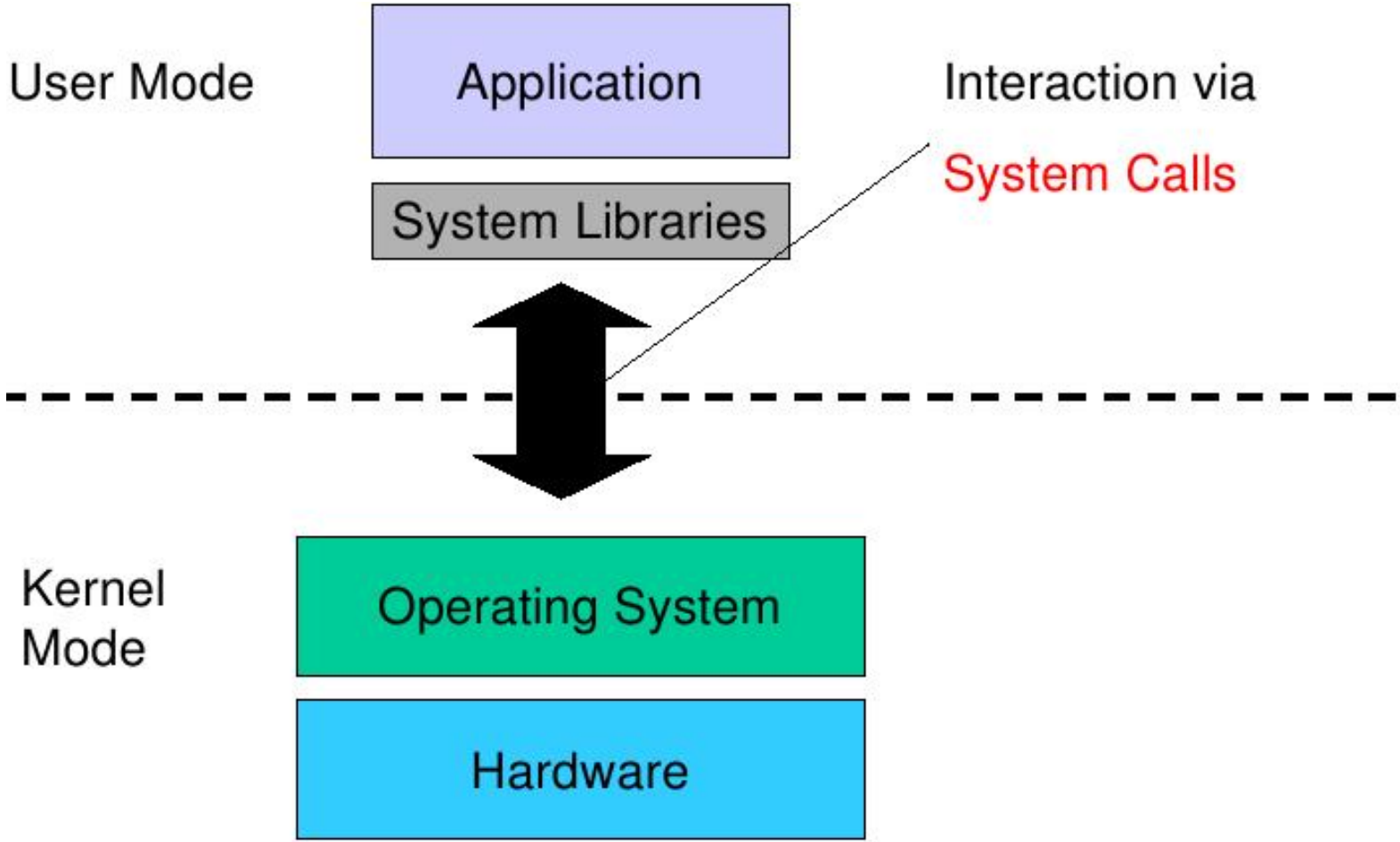
Structure of a Computer System



Structure of a Computer System



Structure of a Computer System



A note on System Libraries

System Libraries

- System libraries are just that, libraries of support functions (procedures, subroutines)
  - Only a subset of library functions are actually systems calls
    - strcmp(), memcpy(), are pure library functions open(), close(), read(), write() are system calls
  - System call functions are in the library for convenience

Operating System Objectives

- Convenience
  - Make the computer more convenient to use
- Abstraction
  - Hardware-independent programming model
- Efficiency
  - Allows the computer system to be used in an efficient manner
- Ability to evolve
  - Permit effective development, testing, and introduction of new system functions without interfering with existing services
- Protection

Services Provided by the Operating System

- Program development
  - Editors, compilers, debuggers
    - Not so much these days
- Program execution
  - Load a program and its data
- Access to I/O devices
- Controlled access to files
  - Access protection
- System access
  - User authentication
- Error detection and response
  - internal and external hardware errors
    - memory error
    - device failure
  - software errors
    - arithmetic overflow
    - access forbidden memory locations
  - operating system cannot grant request of application