IN1011

Operating Systems

Lecture 01 (part 1): Operating Systems – what are they and why are they Useful?

Questions

- What is an Operating System (OS)?
- What services do OSes provide?
- What is the relationship between an OS and the brain of a computer – a processor (e.g. CPU**)?

***central processing unit*

What is an Operating System?

- A program (or collection of programs) that controls the execution of application programs
- An interface between applications and hardware

Main objectives of an OS:

- User Convenience
- System Efficiency, Reliability, and Security
- Ability to add new system functions without interfering with services provided by the system

Is there a formal OS Definition?

- No universally accepted definition
- People typically mean the **Kernel**
  - a collection of software modules, each responsible for a specific low-level function
  - Kernel code runs when needed e.g. at system startup or every time the CPU is interrupted
  - Everything else is either a *system* program (ships with the operating system) or an *application* program
    - A system program is used in creating, running, manipulating or analysing applications – e.g. file explorers, runtime libraries, debuggers and process monitors.

Image courtesy of medium.com

The Kernel

- The kernel is a **resource allocator**
  - manages all resources
  - decides between conflicting requests, for efficient and fair resource use
- The kernel is a **control program**
  - it prevents and contain errors
  - it prevents the improper use of the computer

Image courtesy of Gizbot.com

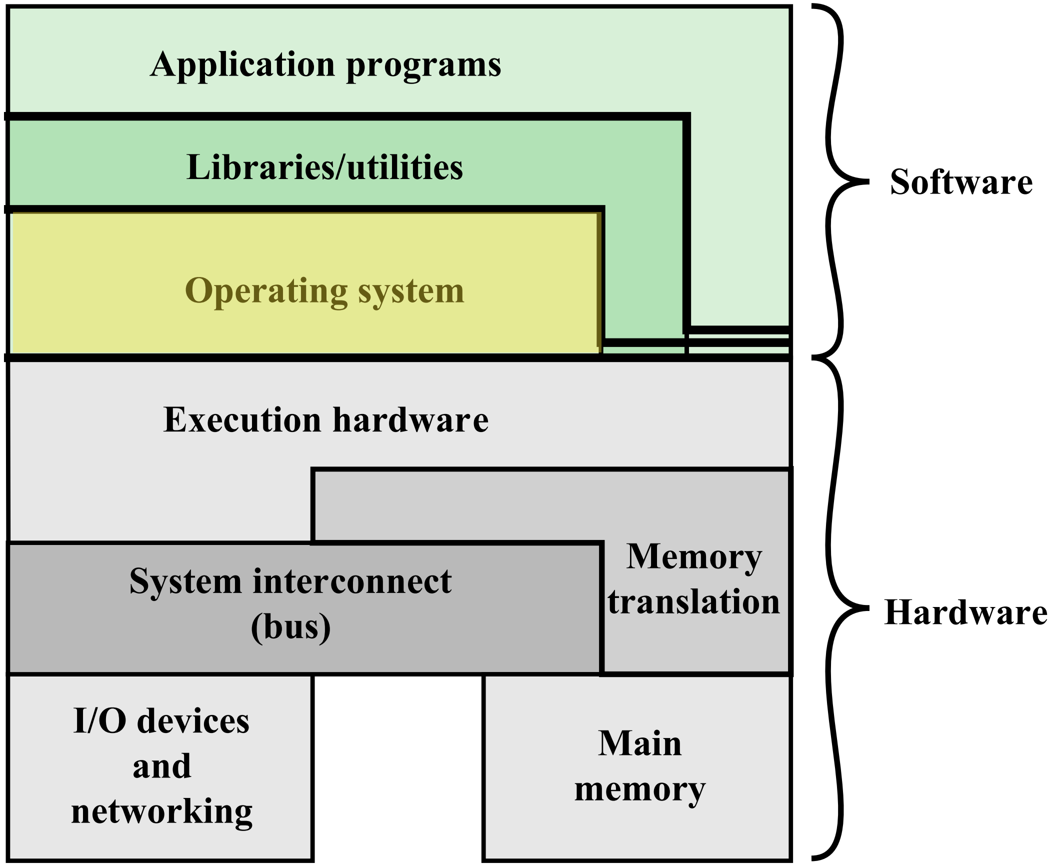


Figure 2.1 Computer Hardware and Software Structure

What Services does an OS Provide?

- Program development – editors, debuggers, utilities
- Manage program execution
- Access I/O devices via uniform interface
- Controlled access to files
- System access
- Error detection and response
- Usage and Performance Statistics

Do we really need an OS?

- We want to run programs
- A program is just a sequence of machine instructions
- The brain of a computer is a processor, e.g. *central processing unit* (CPU), which executes machine instructions
- So, to run a program, all we need to do is point the CPU at the program and let it "do its thing" – i.e. let it execute the instructions.
- ….So what? Big deal!

An OS “*is*” a big deal!

- A CPU can only do one thing at a time, but we want to run lots of programs at the same time
  - **<u>without interfering with each other!</u>**
- We need to get the CPU to work *with a host of devices*!
  - keyboard, disk drive, display, wifi, …
- The CPU can only execute instructions which are in RAM, so we need some way to transfer a program from disk into RAM!

Image courtesy of medium.com

OS and CPU <u>must</u> work together

- If the OS controls the use of a computer's resources…
- …and the OS is not always in explicit control of CPU…
- …it must relinquish control of the CPU to user programs, and must depend on the CPU to allow it to regain control when needed!
- To assist the OS, the CPU supports two modes of operation

Image courtesy of medium.com

Modes Of CPU Operation

User Mode

- CPU executes in user mode for user programs
- Certain areas of memory are protected from user access
- Certain instructions may not be executed

Kernel Mode

- CPU executes in kernel mode for OS
- Protected areas of memory may be accessed
- Privileged instructions may be executed

IN1011

Operating Systems

Lecture 01 (part 2): Computer Systems Review

Question

- What basic hardware elements are needed for modern OSes (and software) to run?

OS Manages Hardware Activity

- It exploits the hardware resources of one or more processors
- It provides a set of services to system users
- It manages secondary memory and I/O devices
- … But, to understand these, we must understand how hardware executes software

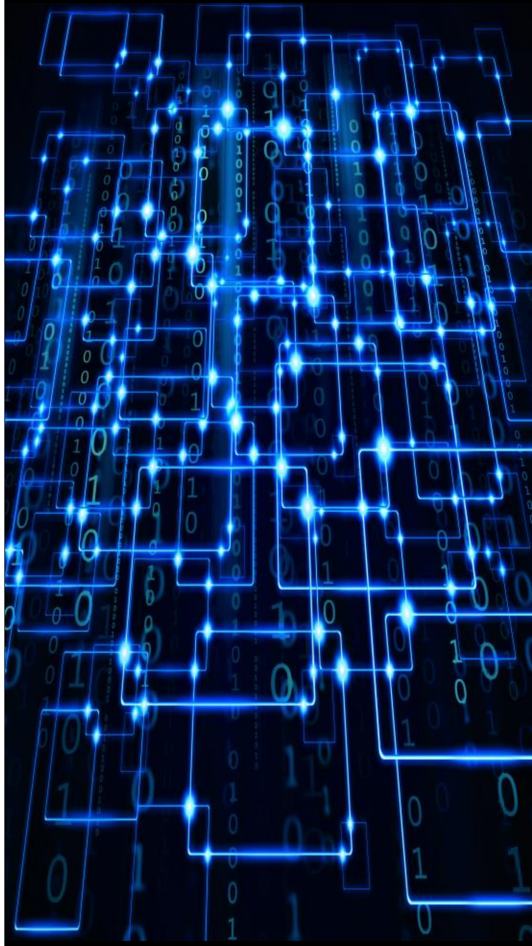
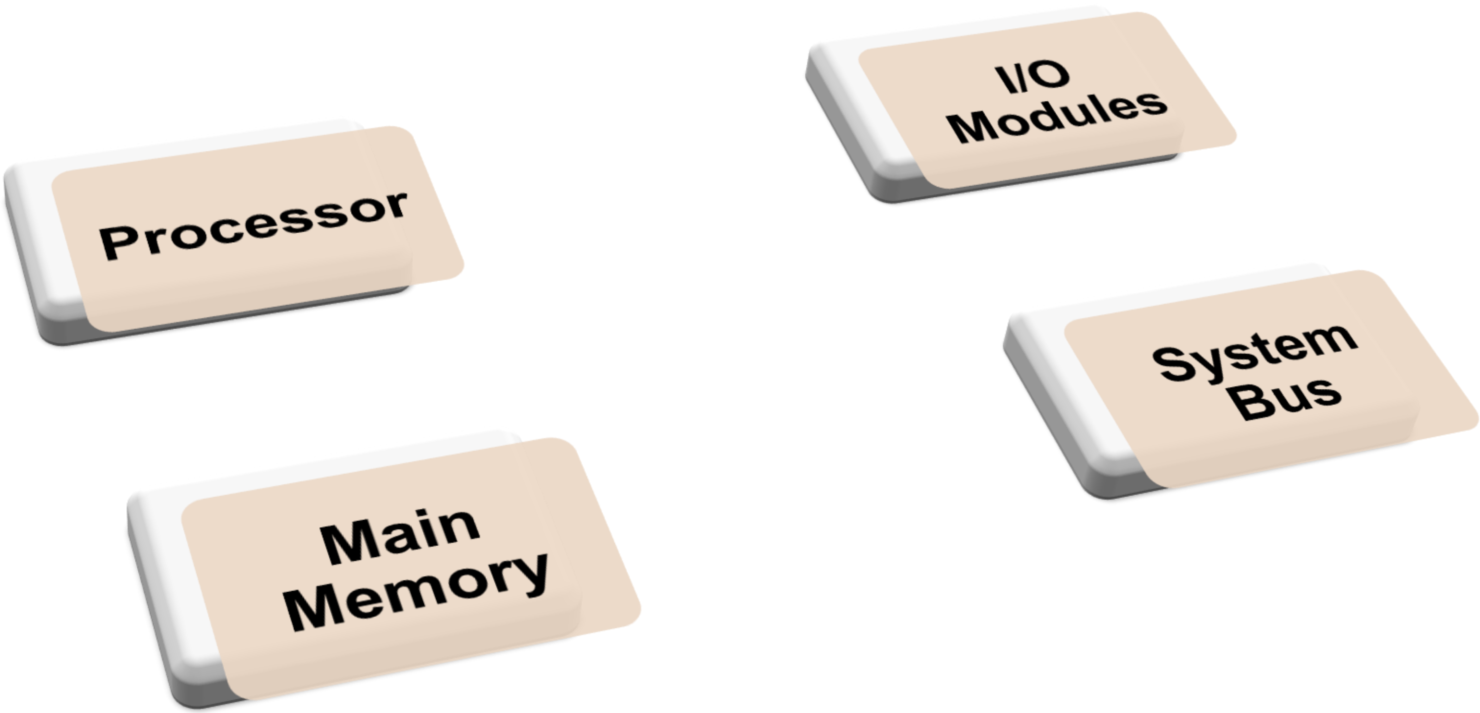


Image courtesy of Wix.com

Computer Systems: 4 Basic Elements



The Processor (CPU)

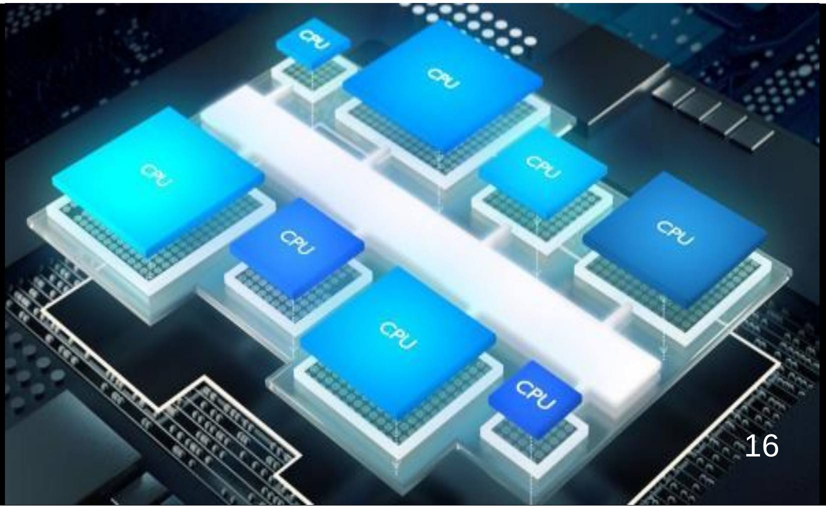
Controls the operation of the computer

Performs most of the data processing functions

Referred to as the ***Central Processing Unit*** (CPU) in modern computers

Comprised of a **control unit**, memory modules called ***registers***, an **arithmetic logic unit**, and **memory caches**

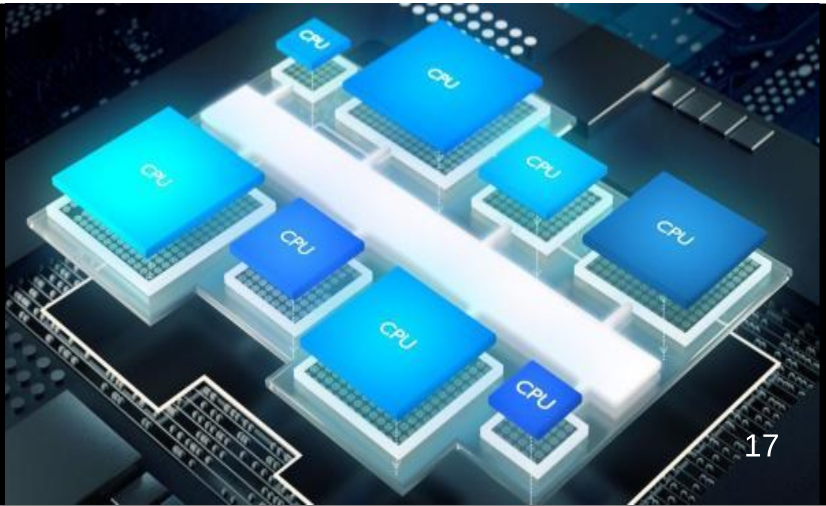
Image courtesy of eeNewsAutomotive.com



Multiple Processors (CPUs)

- Multi-CPU (multi-core) processors are becoming the norm
- IN1011 will primarily consider just a **single CPU**
- Most key OS structures and concepts remain the same as for single CPU machines
  - but also significant additional complexity

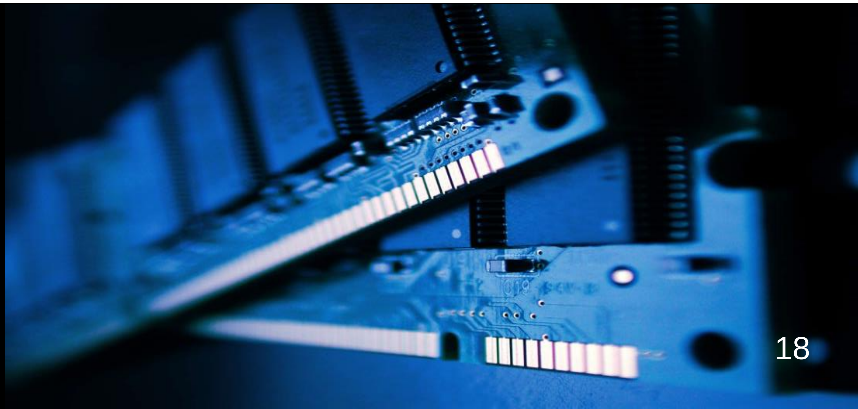
Image courtesy of eeNewsAutomotive.com



Main Memory

- Stores data and programs
- Typically volatile
  - Contents of the memory is lost when the computer is shut down
- Also referred to as *real/primary memory*, *or* ***random access memory*** *(RAM)*

Image courtesy of lifewire.com, nazarethman / Getty Images



I/O Modules

Facilitates moving data between the computer and its external environment (e.g. device controller, I/O channels, directly with I/O device)

- Secondary memory devices (e.g. disks)
- Communications equipment
- Terminals

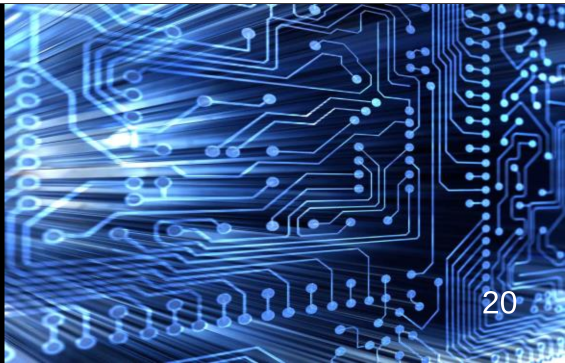
System Bus

– a group of parallel wires/conductors on the main circuit board over which data and signals flow

– Facilitates communication among processors, main memory, and I/O modules

– Architectures vary, from each wire sending a single bit, to each wire being a “lane” along which a series of bits are sent as *messages*

Image courtesy of smallwinsinnovation.com



Computer Systems (simplified)

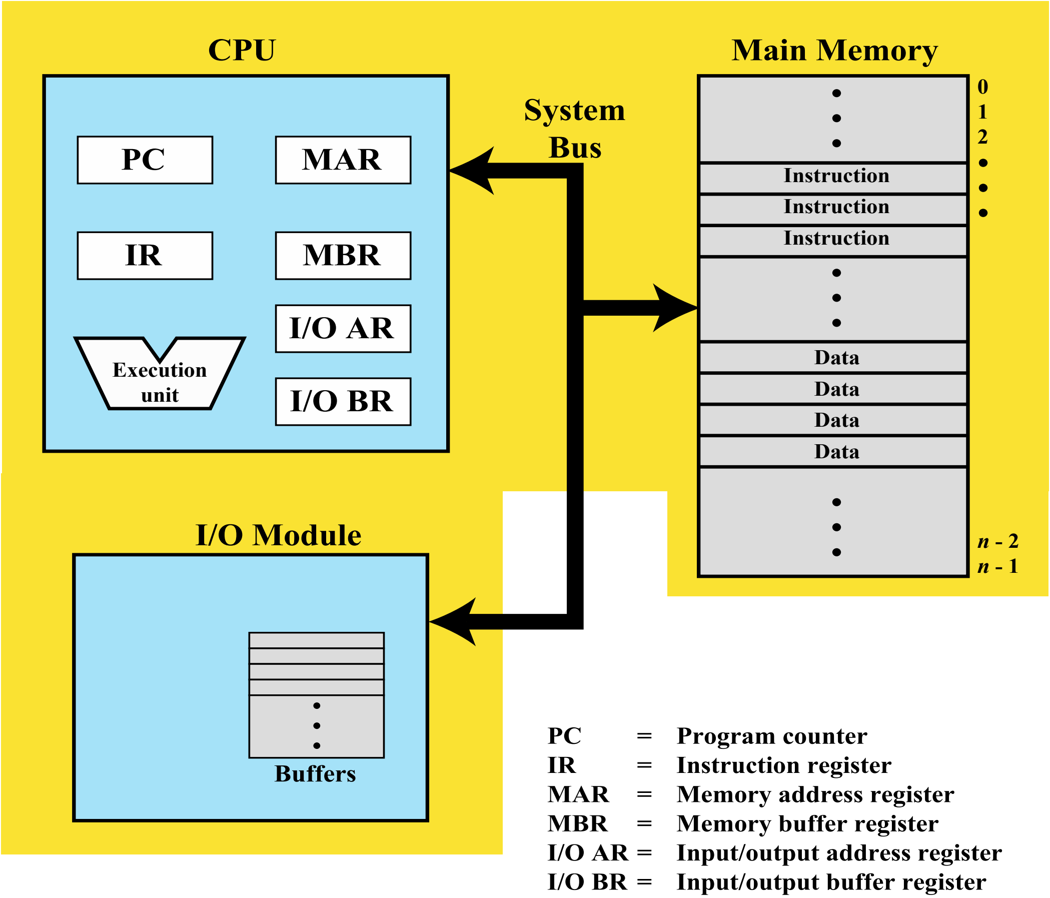


Figure 1.1 Computer Components: Top-Level View

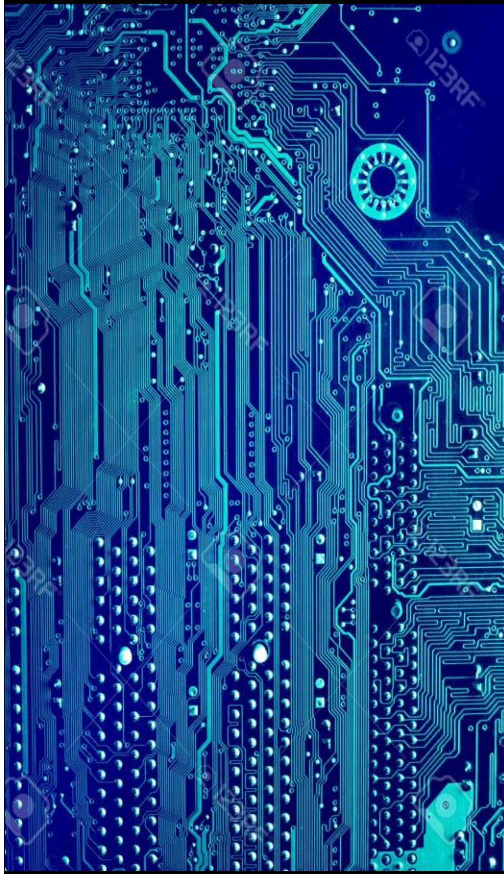


Image courtesy of 123rf.com

How Do these 4 Elements Work Together?

OS design relies on 3 important hardware concepts:

- Basic CPU instruction cycle
  - Executing a program
- Interrupts (CPU and I/O module interaction)
  - More efficient use of CPU
- The Memory Hierarchy
  - Balancing speed of memory, data persistence and cost of memory

IN1011

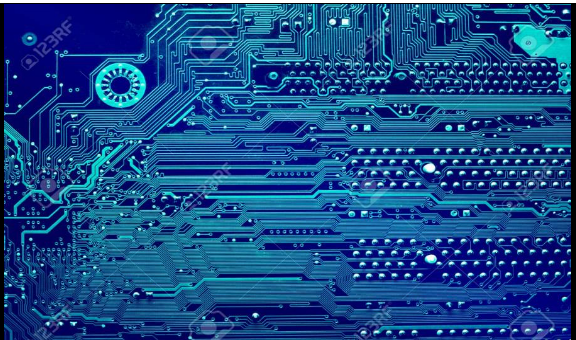
Operating Systems

Lecture 01 (part 3): CPU Instruction Cycle

3 important hardware concepts for OSes

- Basic CPU instruction cycle
  - Executing a program
- Interrupts (CPU and I/O module interaction)
  - More efficient use of CPU
- The Memory Hierarchy
  - Balancing speed of memory, data persistence and cost of memory

Image courtesy of 123rf.com



Questions

- But, what is the CPU instruction cycle?
- How do programs run on a CPU?

Basic CPU Instruction Cycle

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

Three steps

Processor reads (fetches) instructions from memory

Processor decodes and executes each instruction

Basic CPU Instruction Cycle: Fetch, Decode and Execute

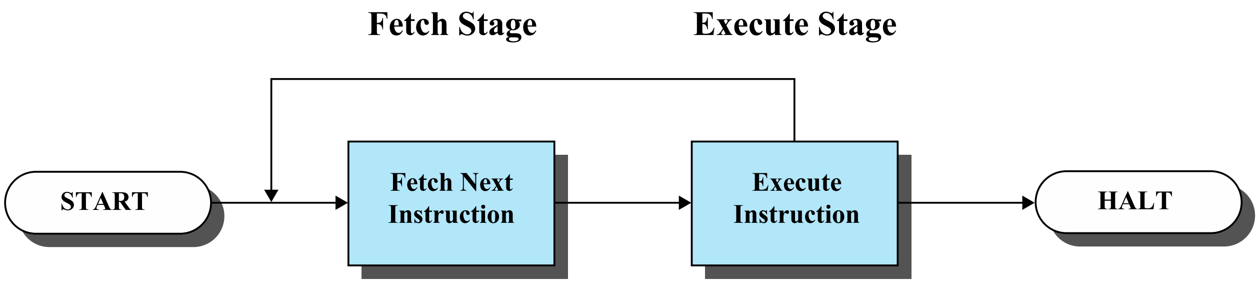
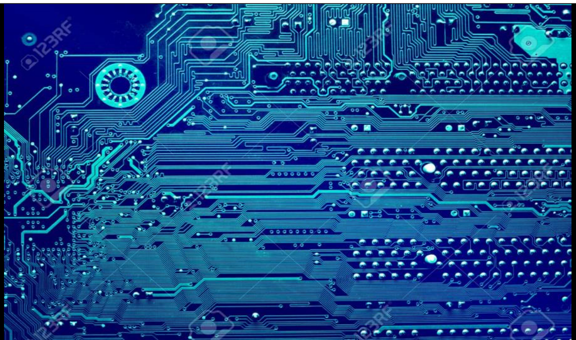


Figure 1.2 Basic Instruction Cycle

Image courtesy of 123rf.com



The Fetch Stage

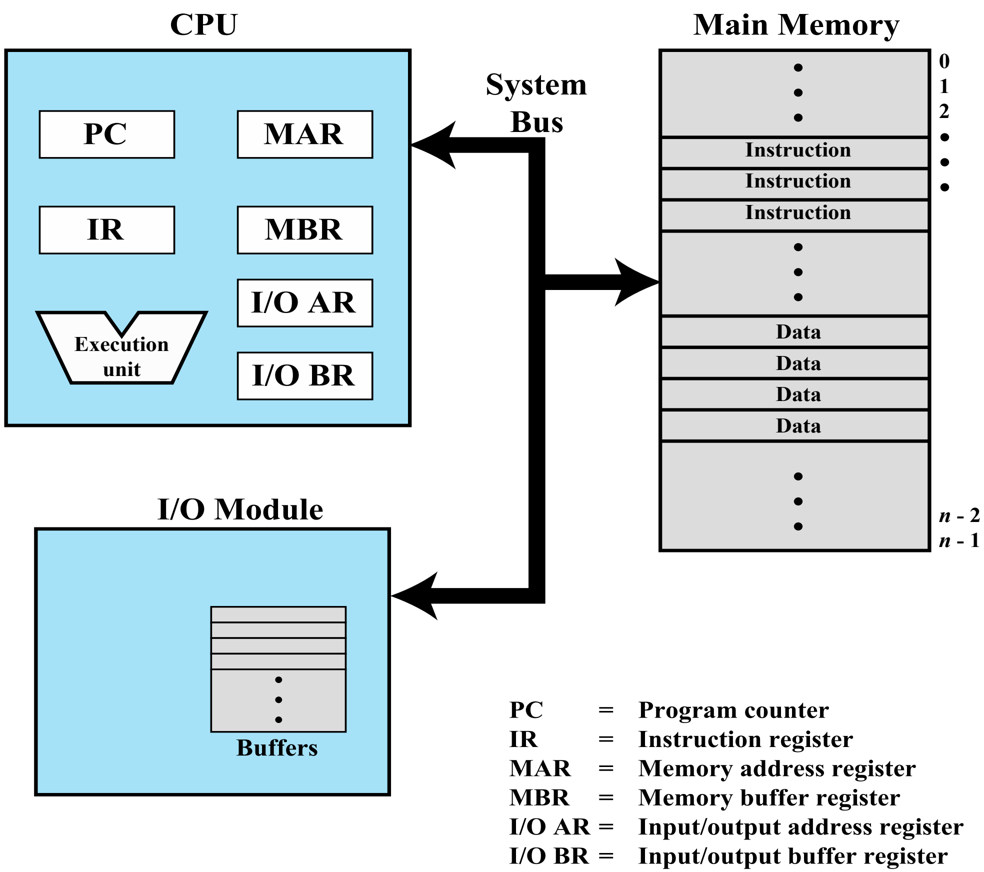


Figure 1.1 Computer Components: Top-Level View

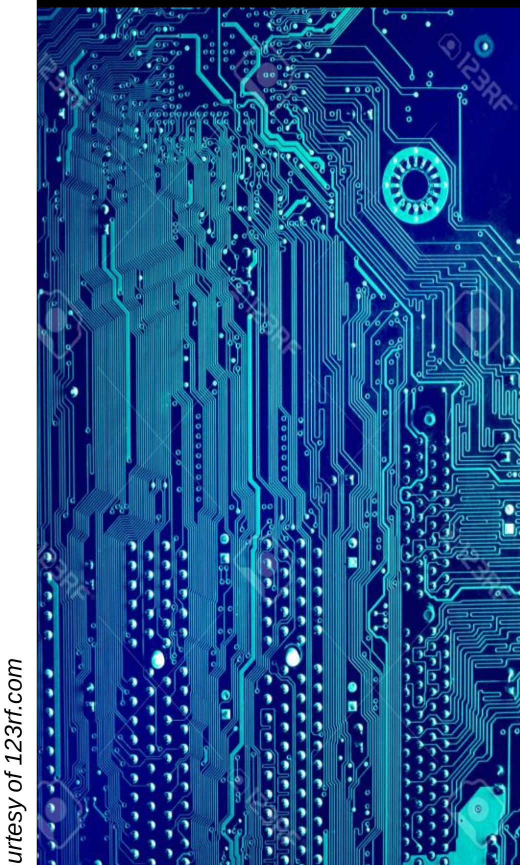
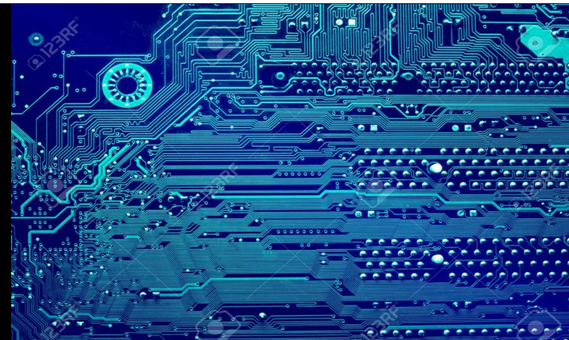


Image courtesy of 123rf.com

Basic CPU Instruction Cycle: Fetch Stage

- The processor fetches an instruction from memory
  - Typically the ***program counter*** (PC) holds the location in memory of the next instruction to be fetched;
  - PC is copied to ***memory address register*** (MAR);
  - Contents of memory at location indicated by MAR are fetched and stored in the ***memory buffer register*** (MBR), then swiftly transferred to the ***instruction register*** (IR) for decoding;
  - PC incremented to store memory location of potential next instruction

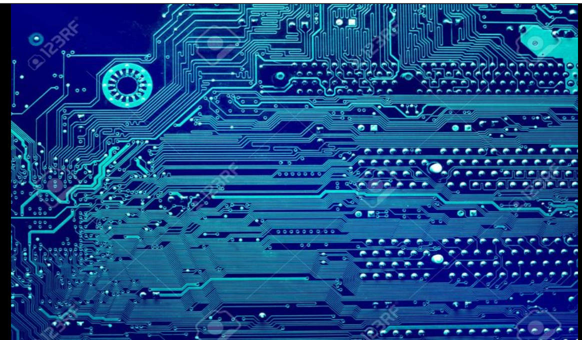
Image courtesy of 123rf.com



Basic CPU Instruction Cycle: Decode and Execute Stages

- CPU control unit interprets/decodes the instruction and initiates the required action:
    - Processor-memory transfer
    - Processor-I/O transfer
    - Data processing
    - Control (e.g. jumps, or returns from procedure calls)

Image courtesy of 123rf.com

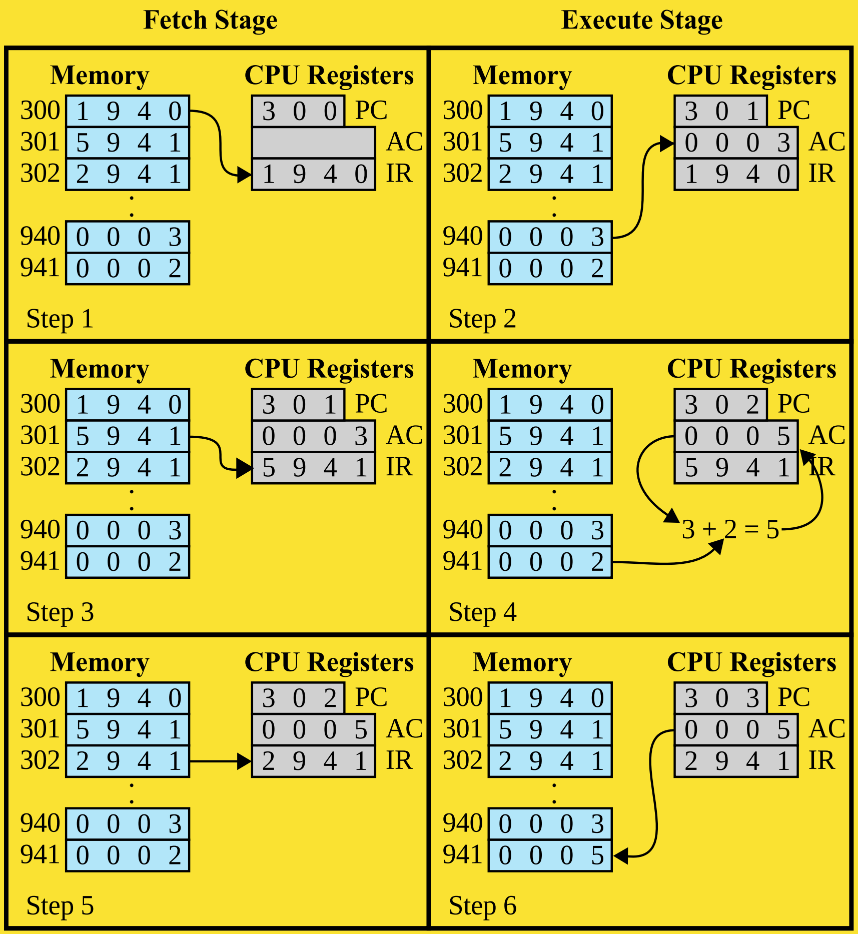


Decode and Execute Stages (contd.)

(a) Instruction format

Image courtesy of 123rf.com

Basic CPU "FDE" Instruction Cycle



opcodes for this example:

1 – copy from location to accumulator register AC

2 – copy AC to location

5 – add data in location to data in AC

Suppose instruction "1940" is in **hexadecimal** and the opcode is stored in the first 4 bits (from the left) of an instruction. Which digits in "1940" represent the opcode?

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

IN1011

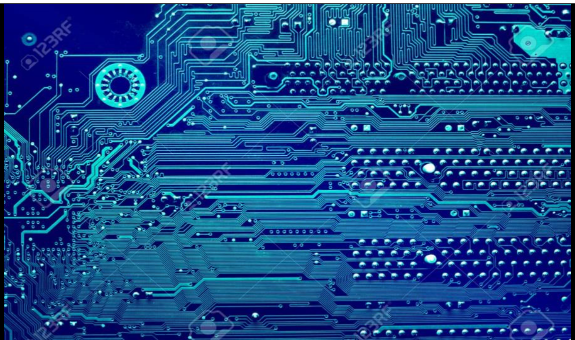
Operating Systems

Lecture 01 (part 4): Interrupts

3 important hardware concepts for OSes

- Basic CPU instruction cycle
  – Executing a program
- Interrupts (CPU and I/O module interaction)
  – More efficient use of CPU
- The Memory Hierarchy
  – Balancing speed of memory, data persistence and cost of memory

Image courtesy of 123rf.com



Questions

- What are interrupts and why are they useful?
- What happens when an interrupt occurs when a program is executing on the CPU?

Basic CPU Instruction Cycle: Fetch, Decode and Execute

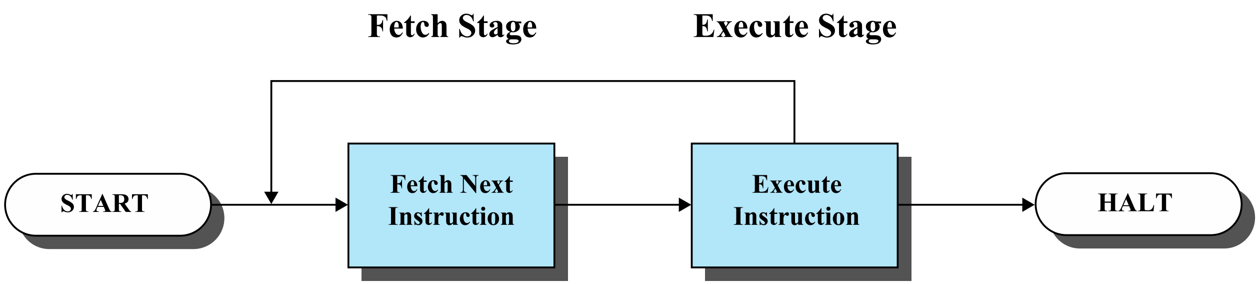
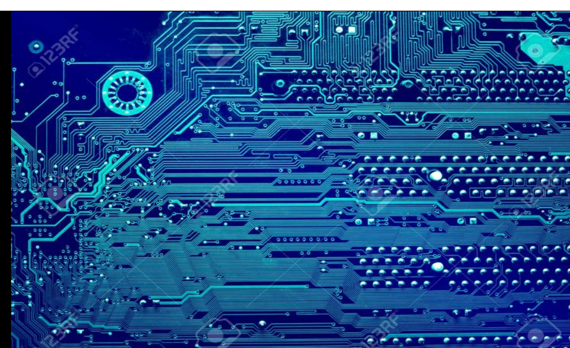
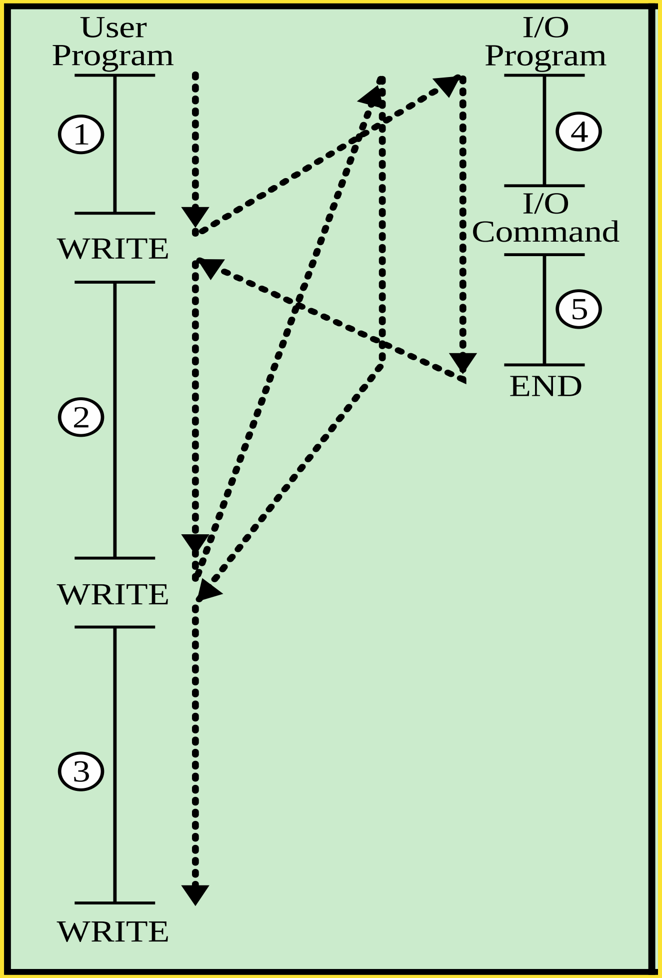


Figure 1.2 Basic Instruction Cycle

Image courtesy of 123rf.com



… But Basic CPU Instruction Cycle is Wasteful



(a) No interrupts

Figure 1.5a

Interrupts!

- A mechanism by which other modules (e.g. I/O, memory) may interrupt the normal sequencing of a CPU
- *Interrupts* improve processor utilization
  - – Most I/O devices are slower than the processor
  - – Processor must pause to wait for device
  - – Wasteful use of the processor

Image courtesy of 123rf.com

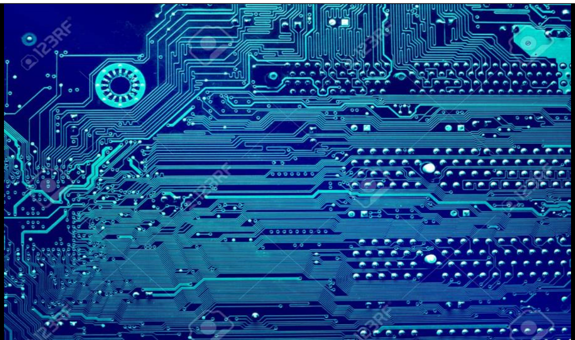
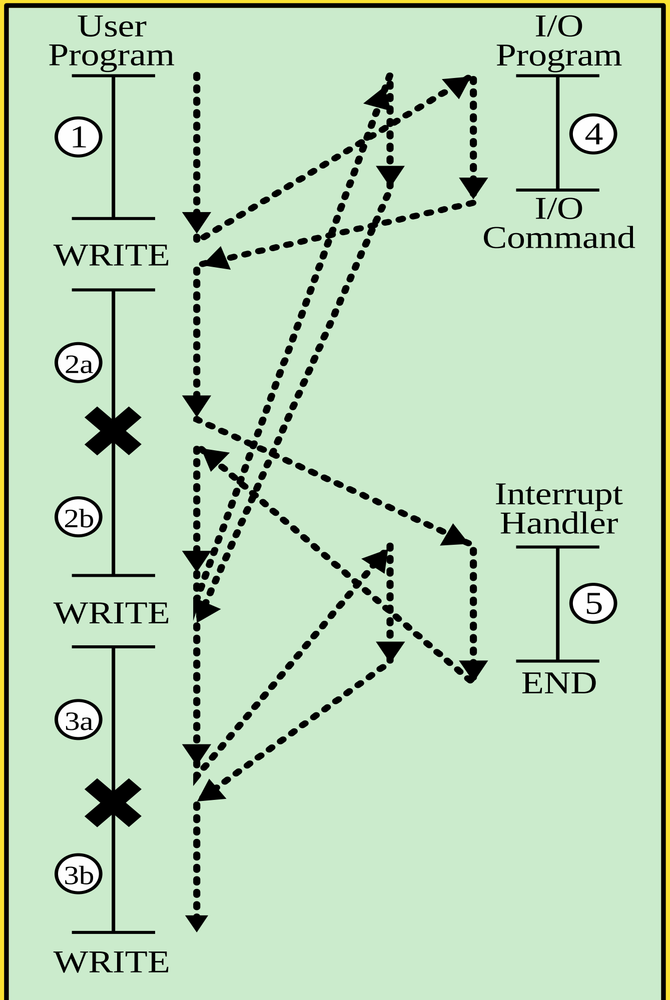


Table 1.1 Classes of Interrupts

Program/ Software	Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempting to execute an illegal machine instruction, or referencing 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.

Interrupts with Short I/O Activity

X = interrupt occurs during course of execution of user program



(b) Interrupts; short I/O wait

Figure 1.5b

Interrupts with Short I/O Activity

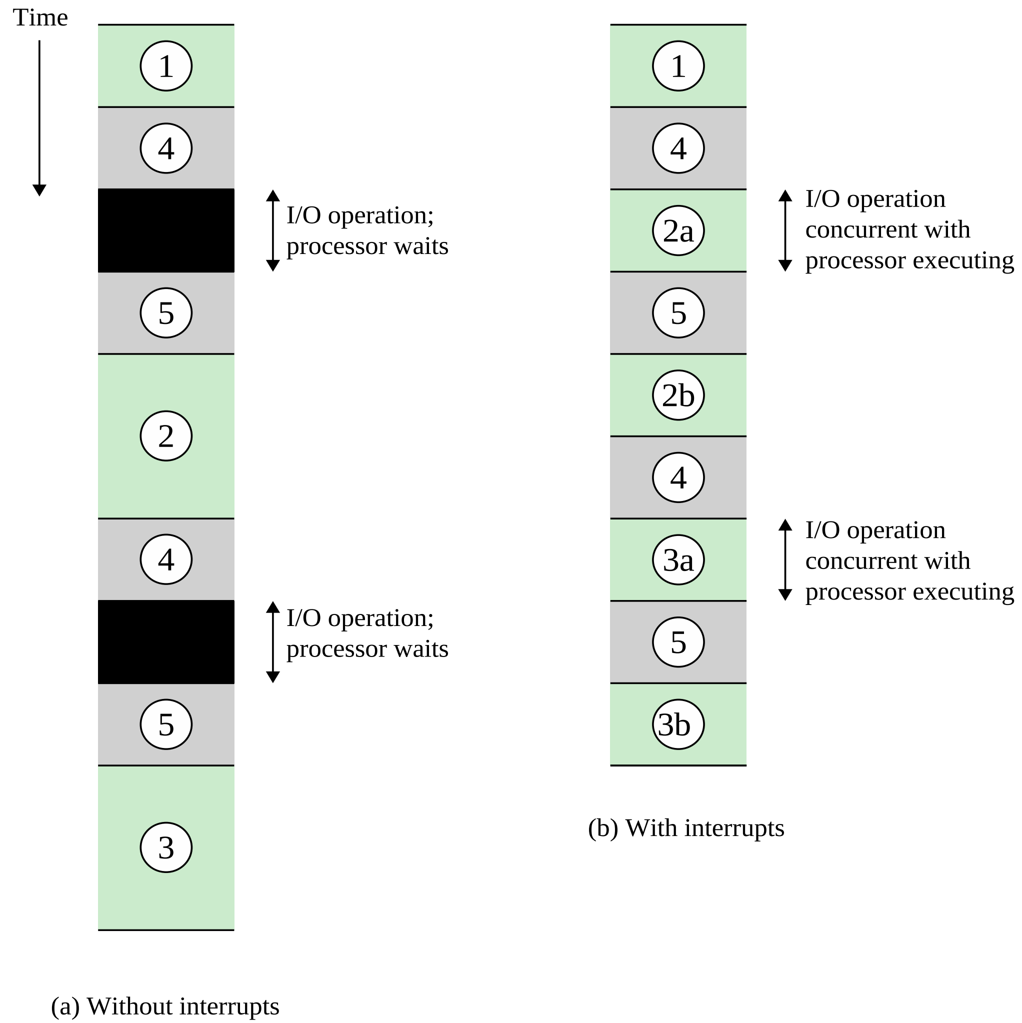
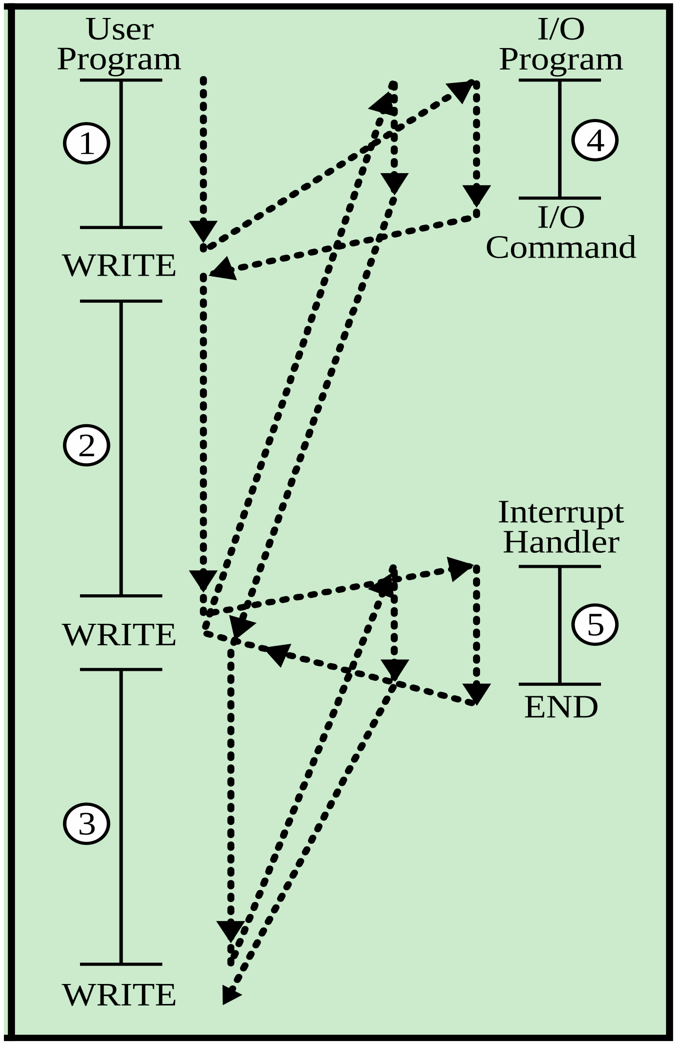


Figure 1.8 Program Timing: Short I/O Wait

Interrupts with Long I/O Activity



(c) Interrupts; long I/O wait

Figure 1.5c

Interrupts with Long I/O Activity

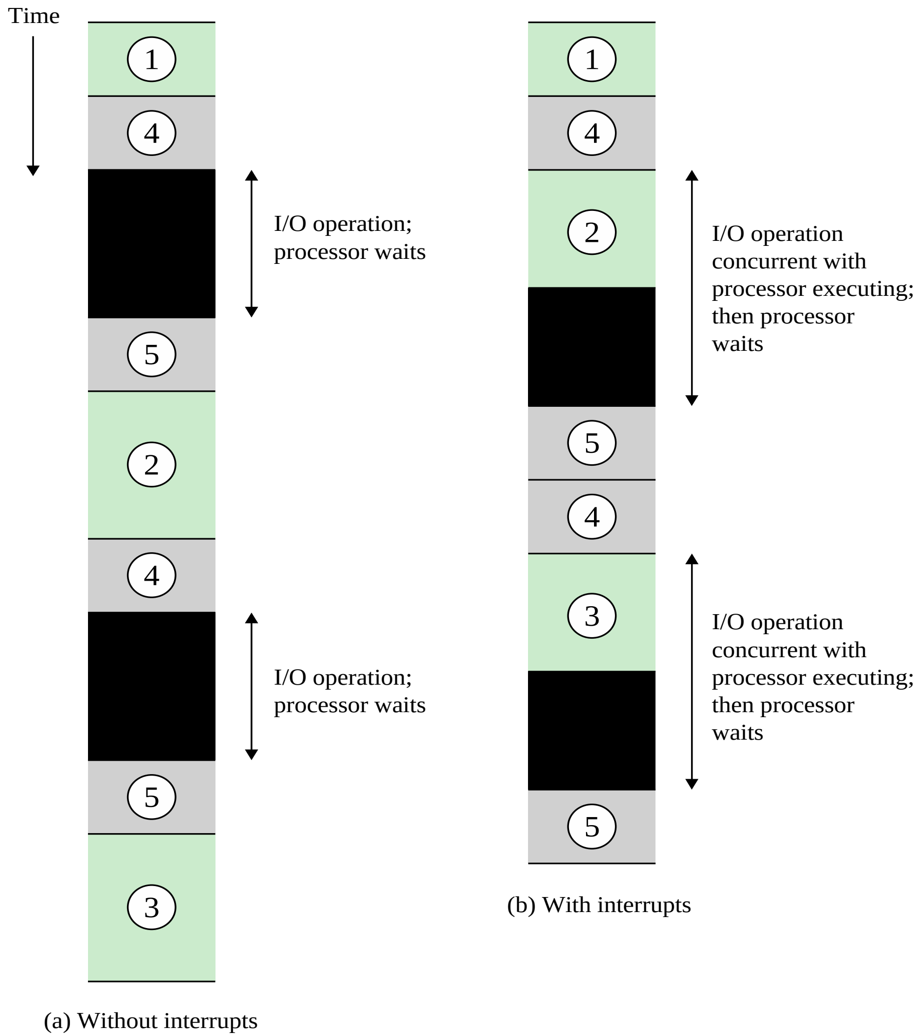
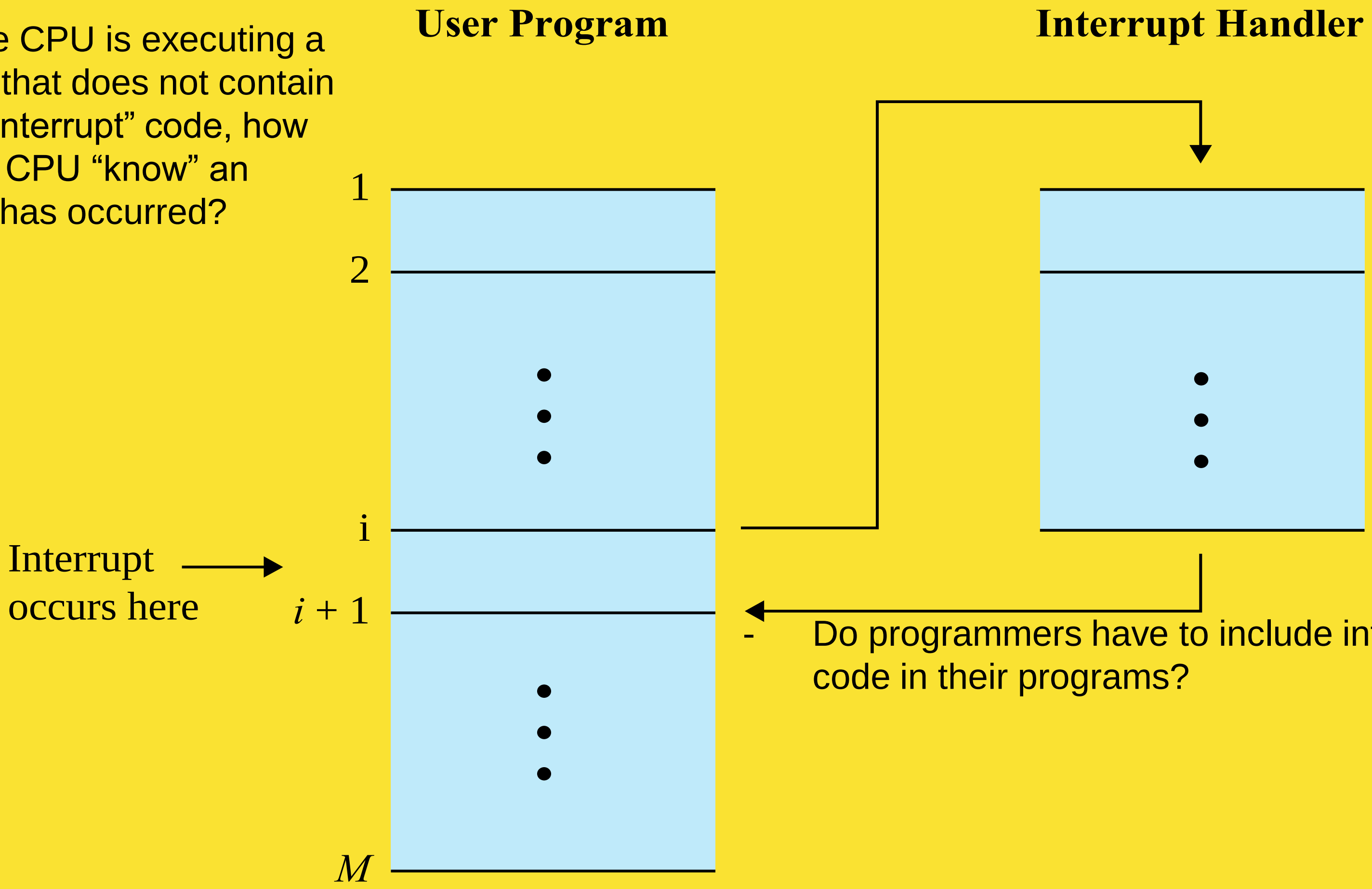


Figure 1.9 Program Timing: Long I/O Wait

How are Interrupts “handled”?

But, if the CPU is executing a program that does not contain special “interrupt” code, how does the CPU “know” an interrupt has occurred?



- Do programmers have to include interrupt handling code in their programs?

Figure 1.6

CPU Instruction Cycle … with Interrupts!

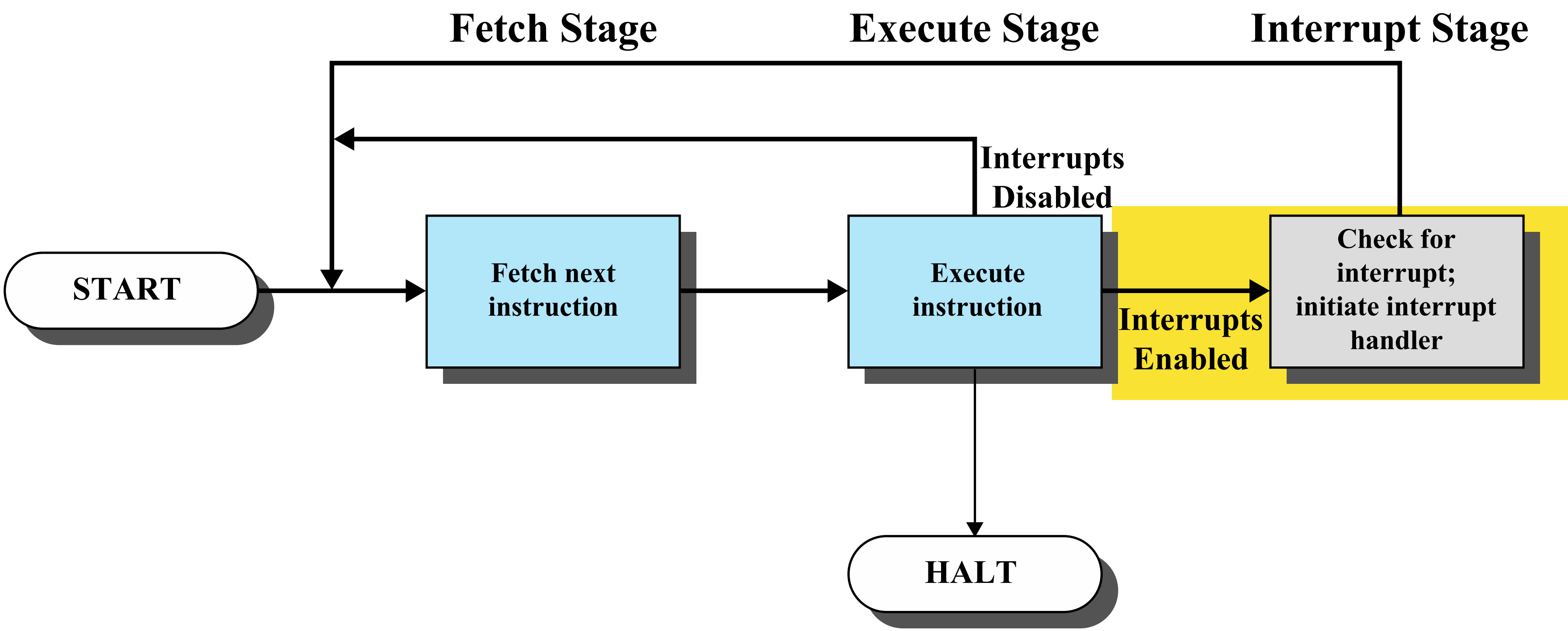


Figure 1.7 Instruction Cycle with Interrupts

Interrupt Handling Process

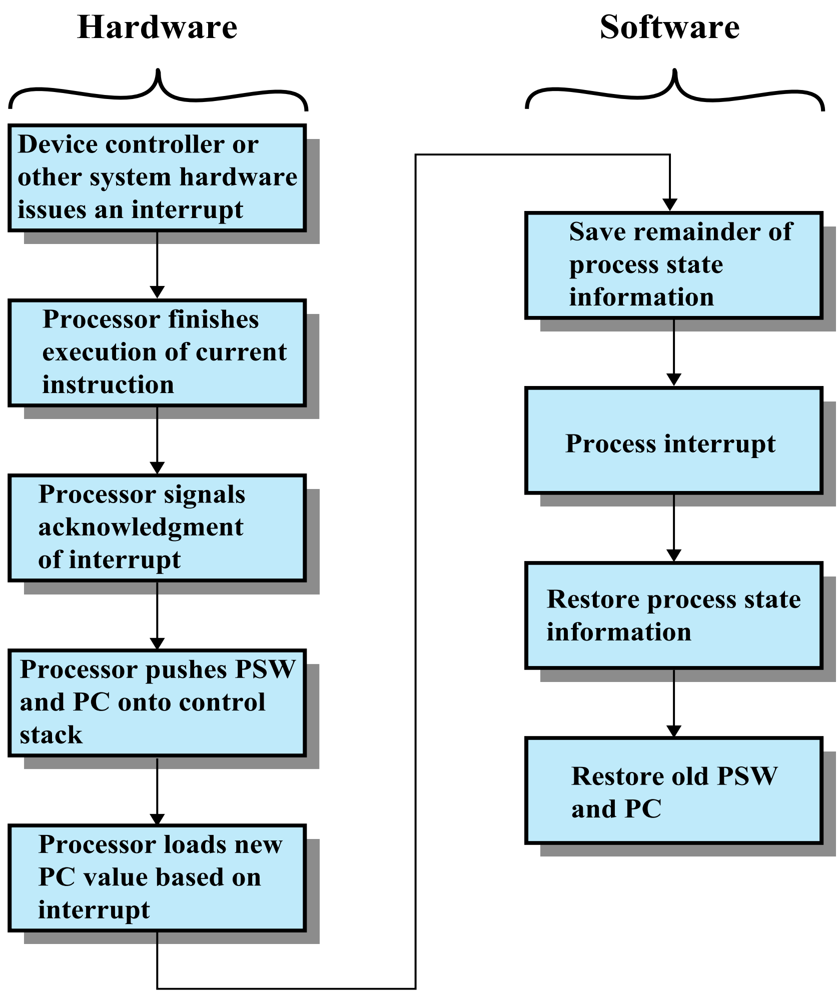


Figure 1.10 Simple Interrupt Processing

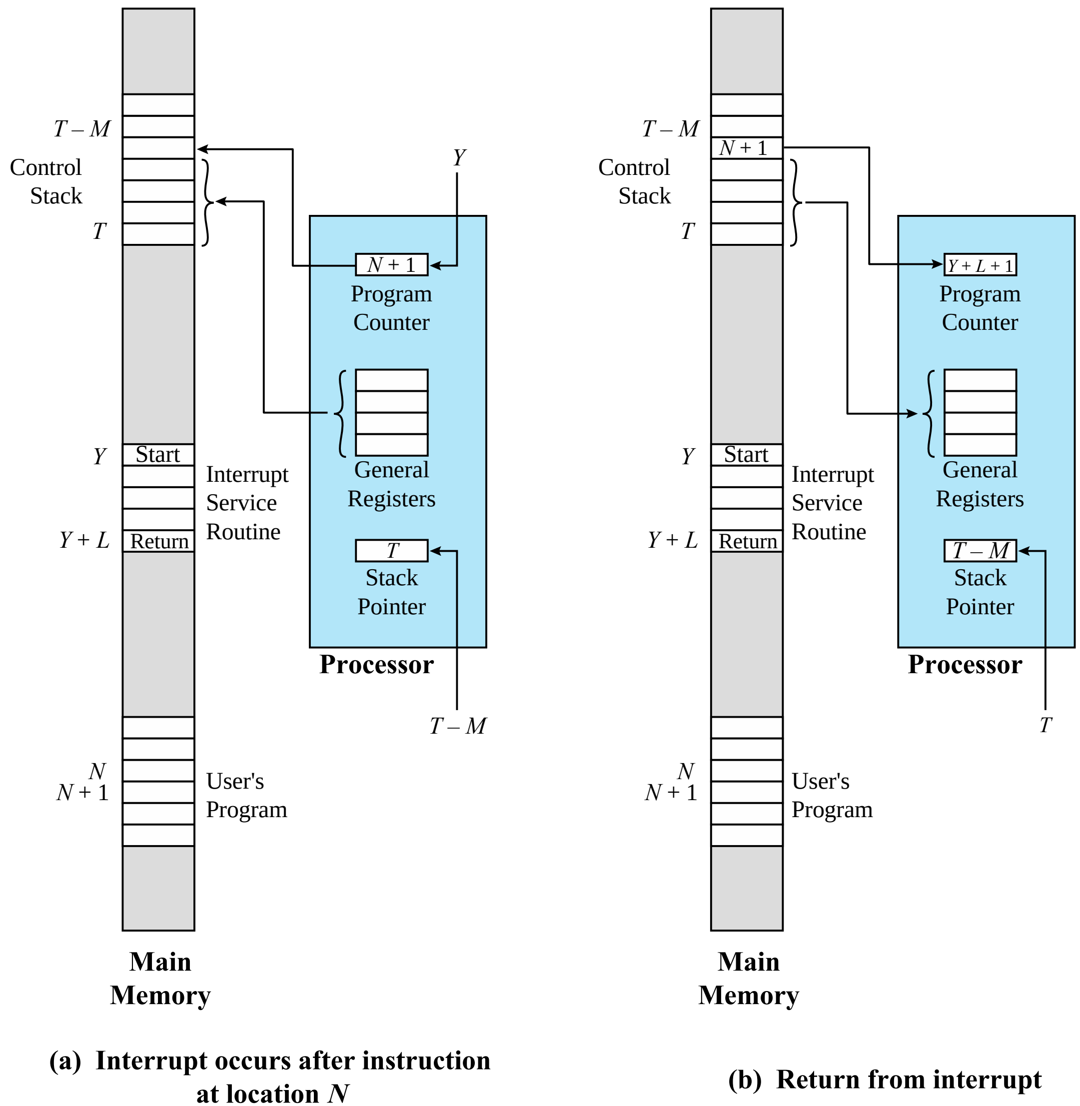


Figure 1.11 Changes in Memory and Registers for an Interrupt

Multiple Interrupts are Possible

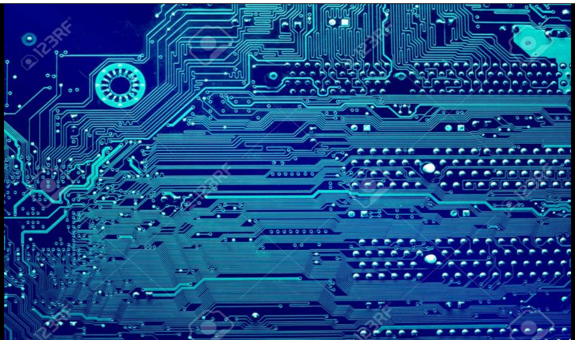
An interrupt occurs while another interrupt is being processed

- 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

Image courtesy of 123rf.com



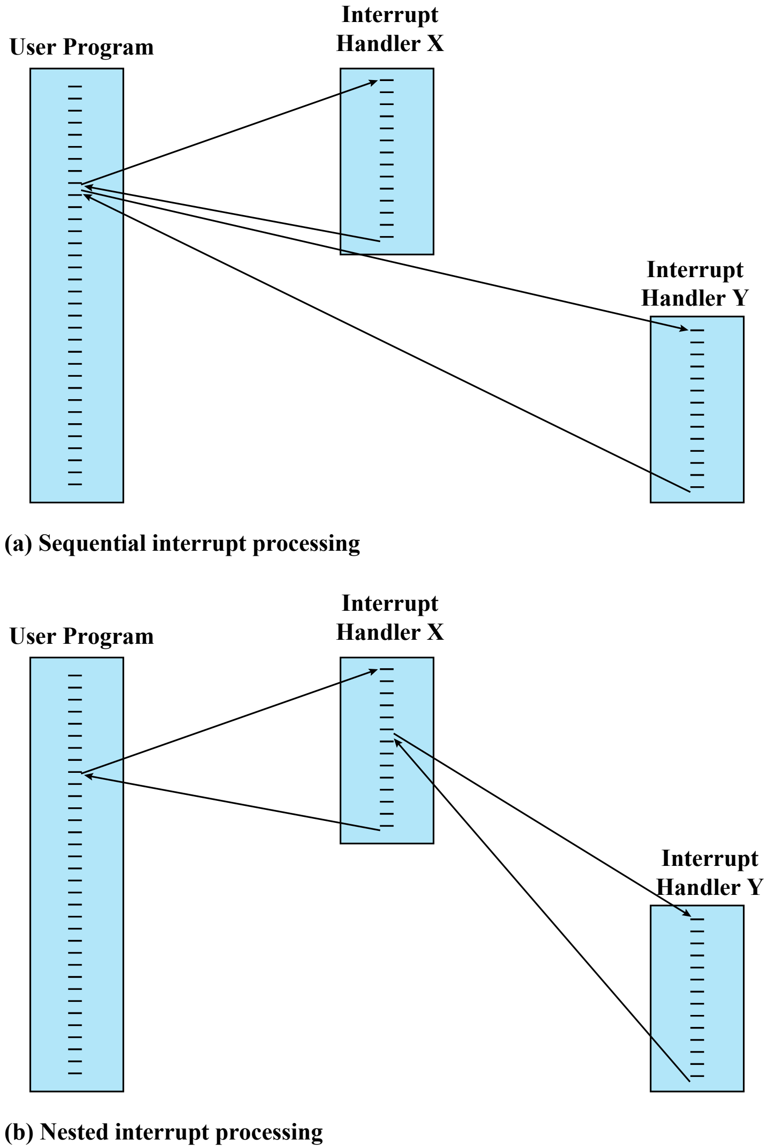


Figure 1.12 Transfer of Control with Multiple Interrupts

IN1011

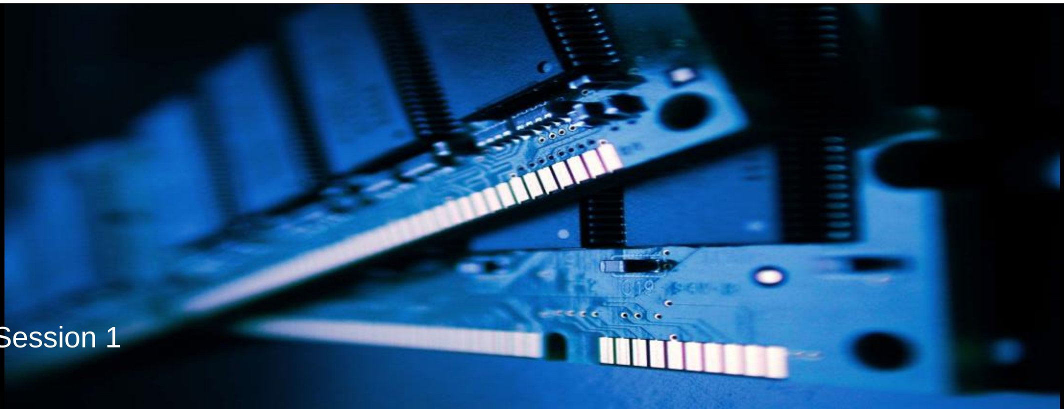
Operating Systems

Lecture 01 (part 5): The Memory Hierarchy

3 important hardware concepts for OSes

- Basic CPU instruction cycle
  - Executing a program
- Interrupts (CPU and I/O module interaction)
  - More efficient use of CPU
- The Memory Hierarchy
  - Balancing speed of memory, data persistence and cost of memory

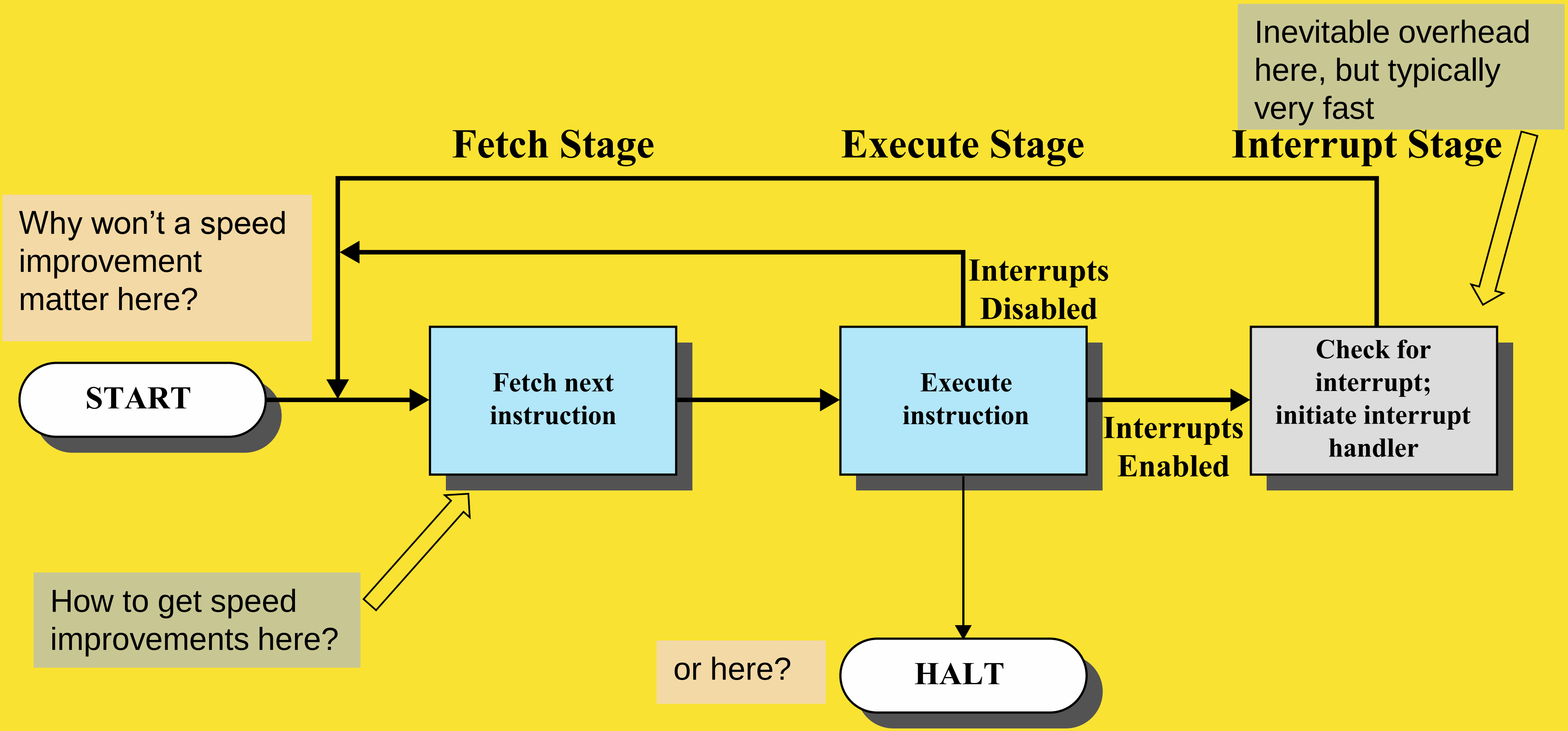
Image courtesy of lifewire.com, nazarethman / Getty Images



Questions

- Why do modern computers have different levels of memory?
- How are these different levels of memory used when a program is running?

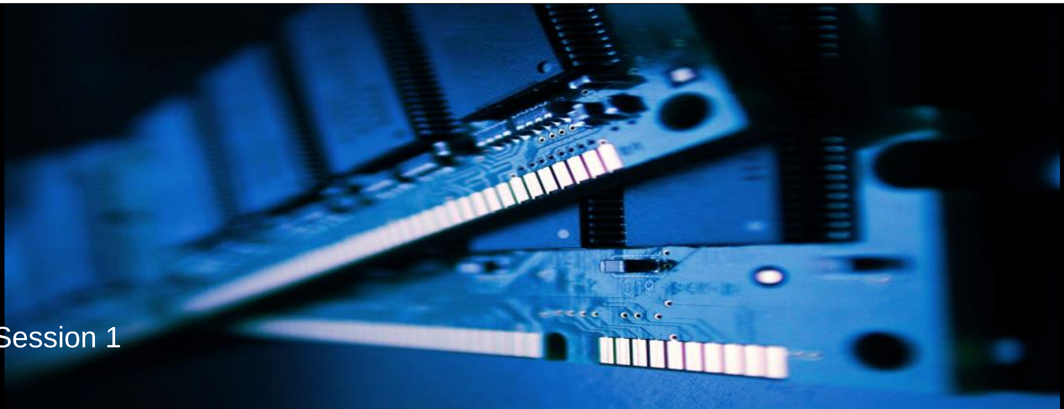
Instruction Cycle Needs to be Fast!



Faster Memory = Faster CPU cycle!

- Design constraints on a computer's memory
  - How much?
  - How fast?
  - How expensive?
- *How much*? If the capacity is there, applications will likely be developed to use it
- *How fast*? Memory must be able to keep up with the processor
- *How expensive*? Cost of memory must be reasonable in relationship to the other components

Image courtesy of lifewire.com, nazarethman / Getty Images



Memory Relationships

Faster access time = greater cost per bit

Greater capacity = slower access speed

Greater capacity = smaller cost per bit

There is a Memory Hierarchy

- Going down the hierarchy:
  a. Decreasing cost per bit
  b. Increasing capacity
  c. Decreases in speed
  d. Decreasing frequency of access to the memory by the processor

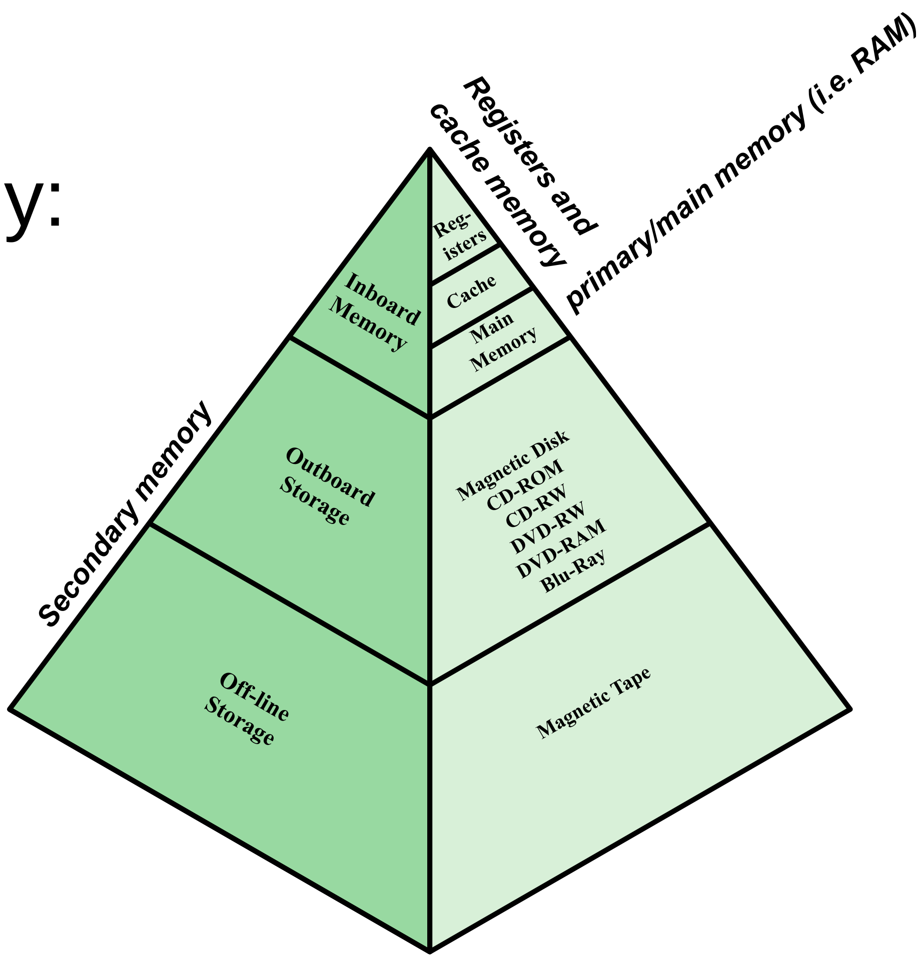
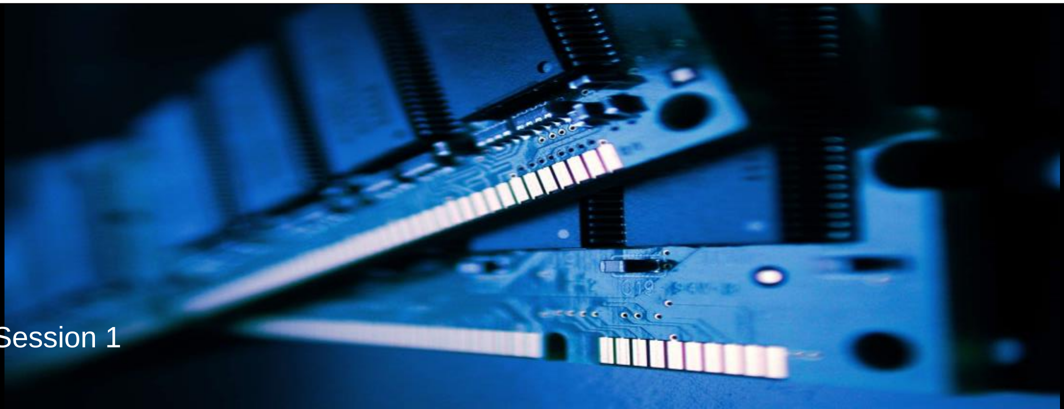


Figure 1.14 The Memory Hierarchy

Principle of “Locality of Reference”

- The processor is likely to reference locations in memory “close to” locations that have recently been referenced;
- That is, CPUs references to both instructions and data in memory tend to cluster;
- Think about program loops or processing arrays containing data;
- Data is typically organized so that the percentage of accesses to each successively lower level in the hierarchy is substantially less than that of the level above

Image courtesy of lifewire.com, nazarethman / Getty Images



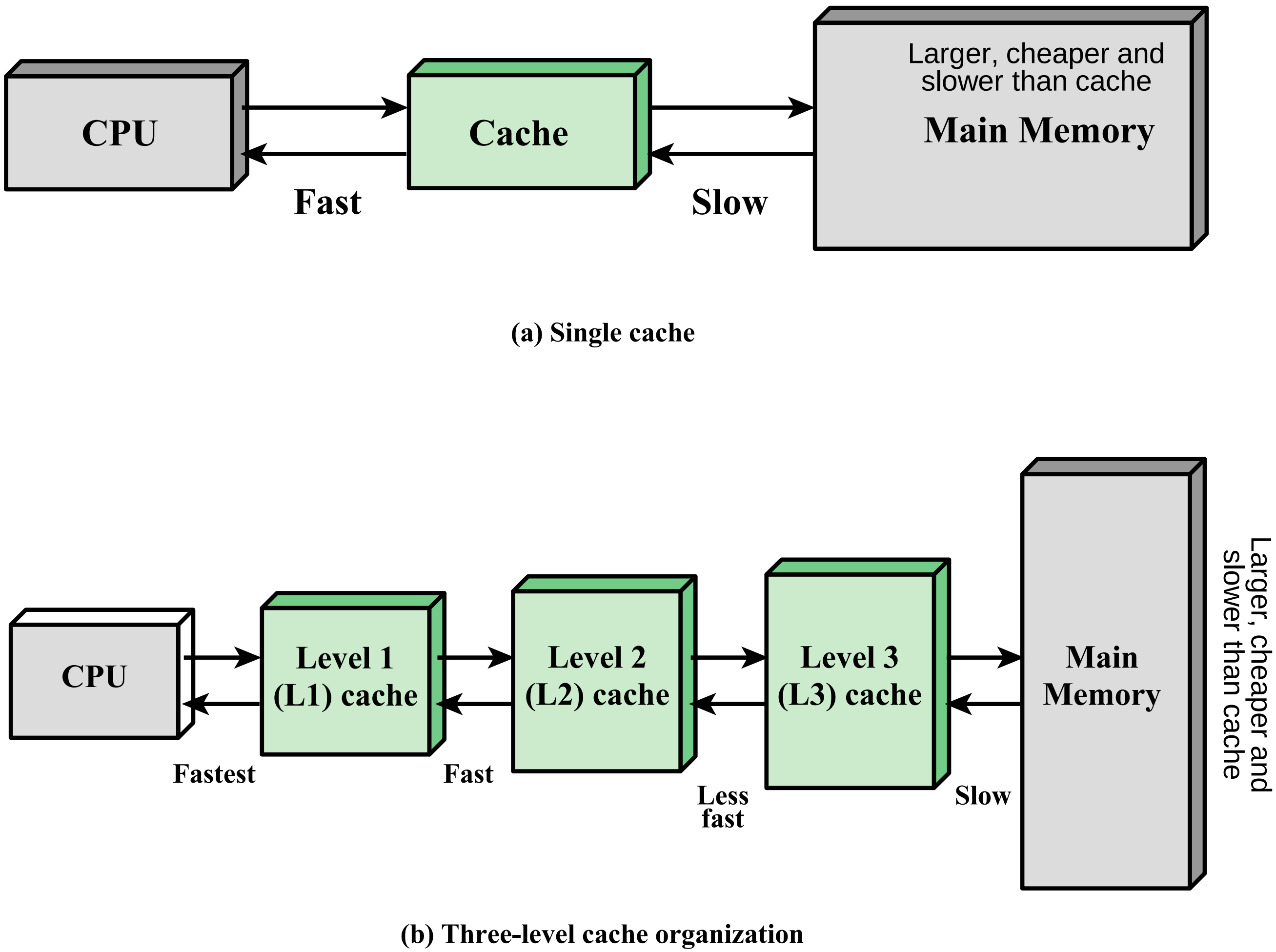


Figure 1.16 Cache and Main Memory

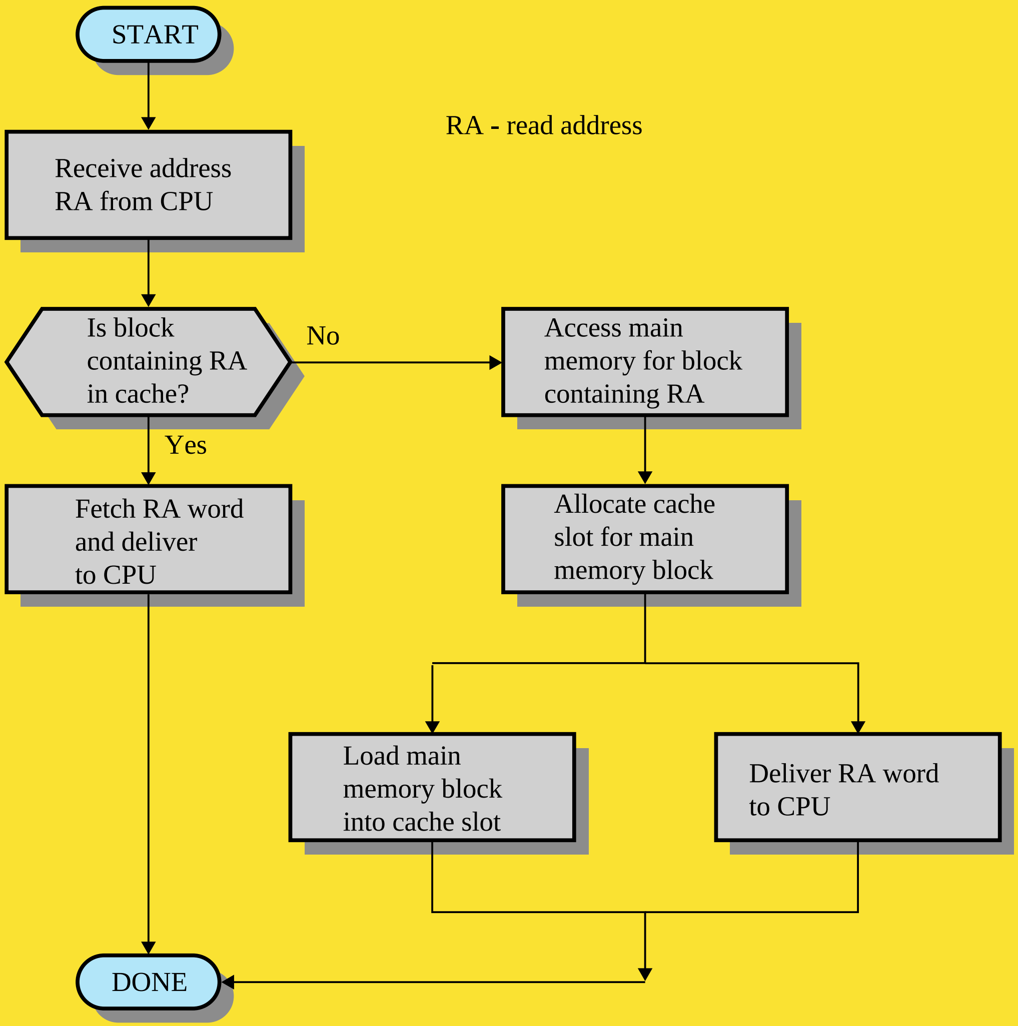


Figure 1.18 Cache Read Operation