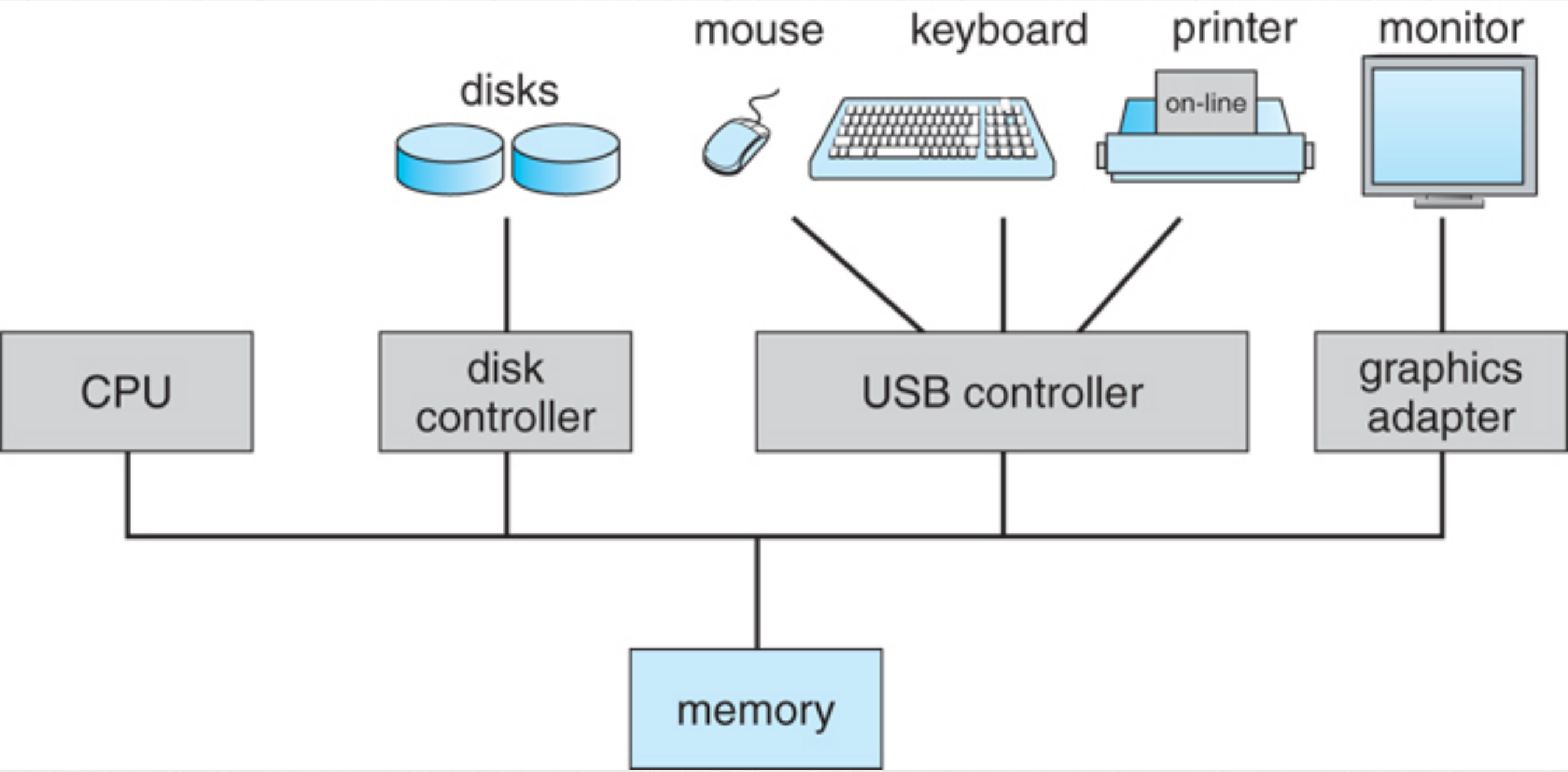
Computer System Overview

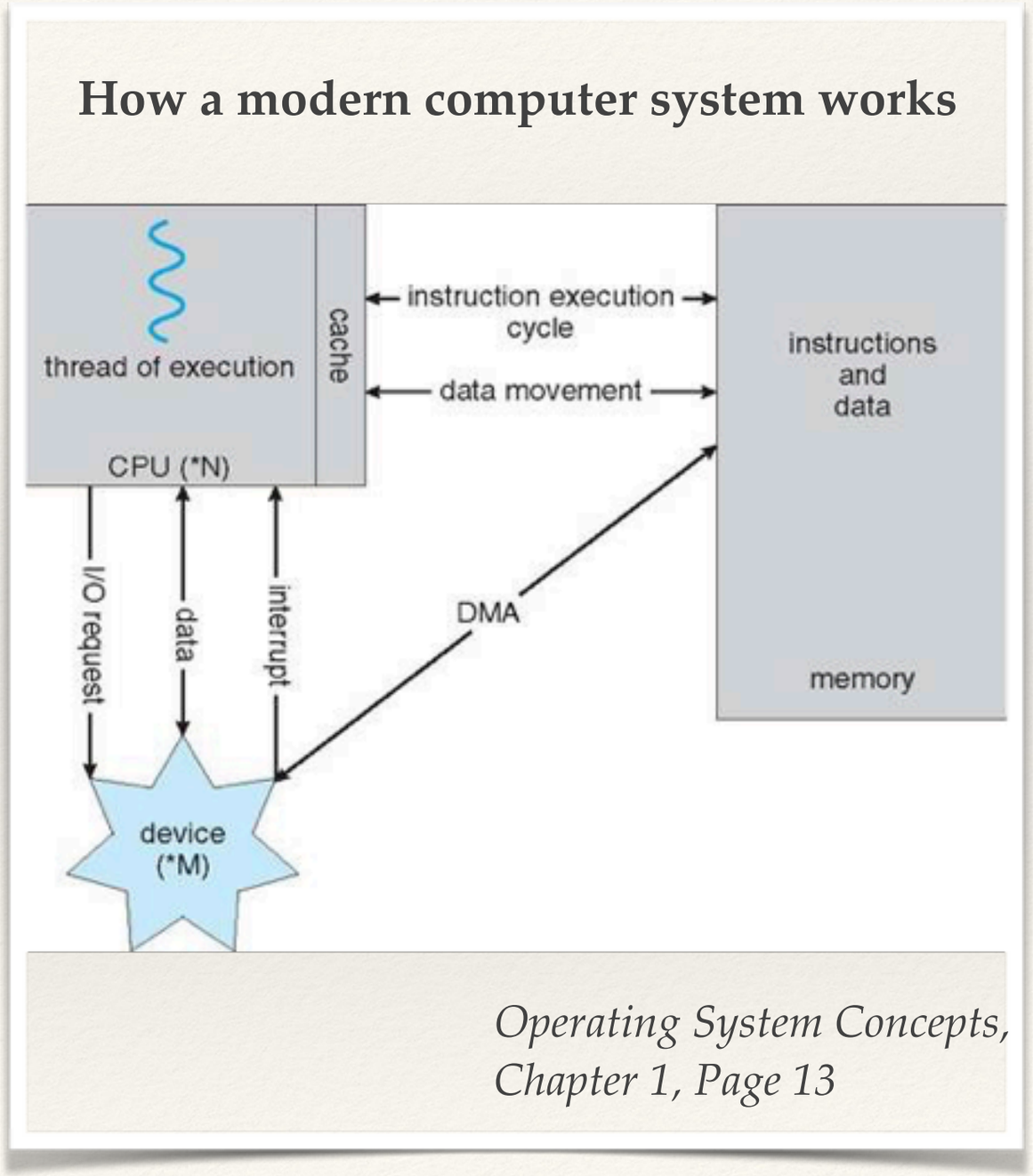
Review of the Fundamentals of Computer System Organization

A Modern Computer System



Instruction Execution

- The CPU executes instructions in a program.
- An instruction is fetched from memory and stored in an instruction register.
- Instruction is decoded (operands may be fetched from memory and stored in internal registers).
- After execution, results may be stored back in memory.



Registers

- **User-Visible Registers**
  - *May be referenced by machine language*
  - Available to all programs - application and system programs
  - Data Registers – can be changed by user
  - Address Registers – could be separate from data register
  - Stack Registers – user / supervisor stacks
  - Condition Codes – results of operations

Registers

- **Control and Status Registers**
  - *May or may not be visible*
    - Program Counter (PC) – address of next instruction
    - Instruction Register (IR) – most recently fetched instruction
    - MAR/MBR – memory reference registers
    - Program Status Word (PSW) – condition codes, interrupts, mode

Interrupts

- **Definition**: A suspension of a *process* caused by an event external to that *process* and performed in such a way that the *process* can be resumed.
- It improves processing *efficiency* by allowing the processor to execute other instructions while an I/O operation is in progress.

Examples of Interrupts

- Mouse moved / clicked
- Disk drive operation completed
- Keyboard key pressed
- Printer out of paper
- Video card wants memory access
- Modem sending or receiving
- USB scanner has data

Interrupt Processing

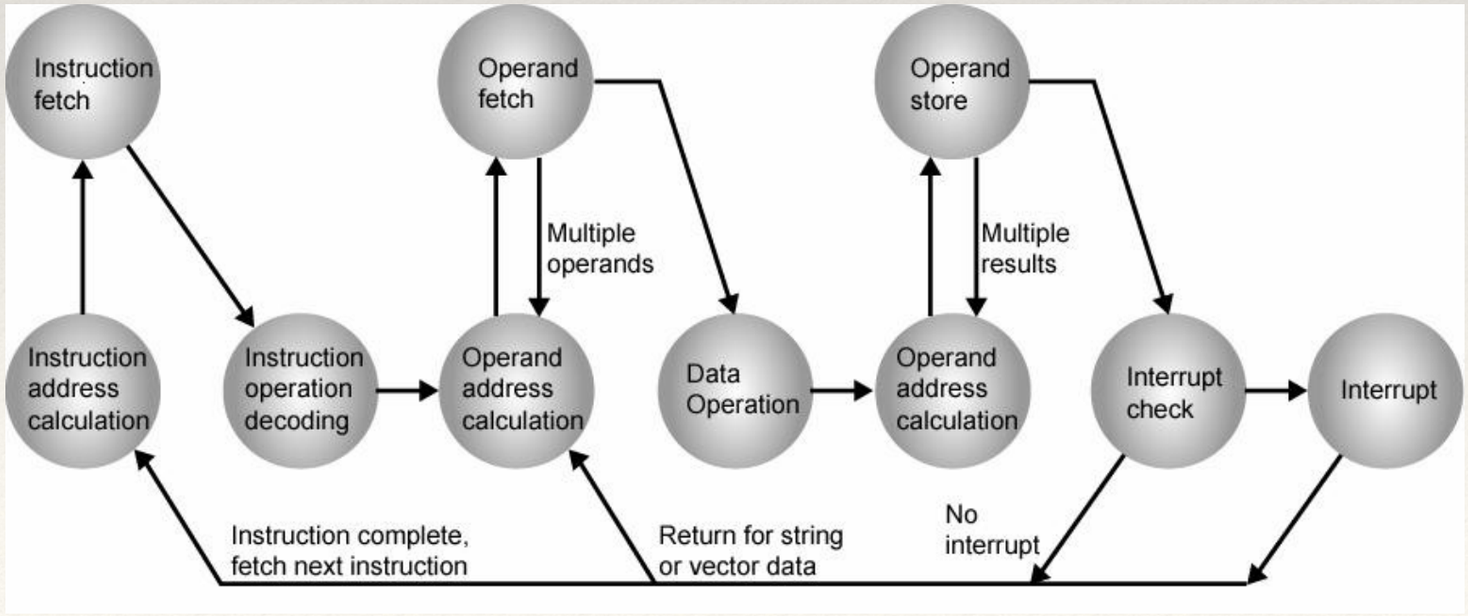
- **Classes of Interrupts**
  - *Program*
    - arithmetic overflow, division by zero, execute illegal instruction, reference outside user's memory space
  - *Timer*
  - *I/O*
  - *Hardware failure*

Interrupt Processing

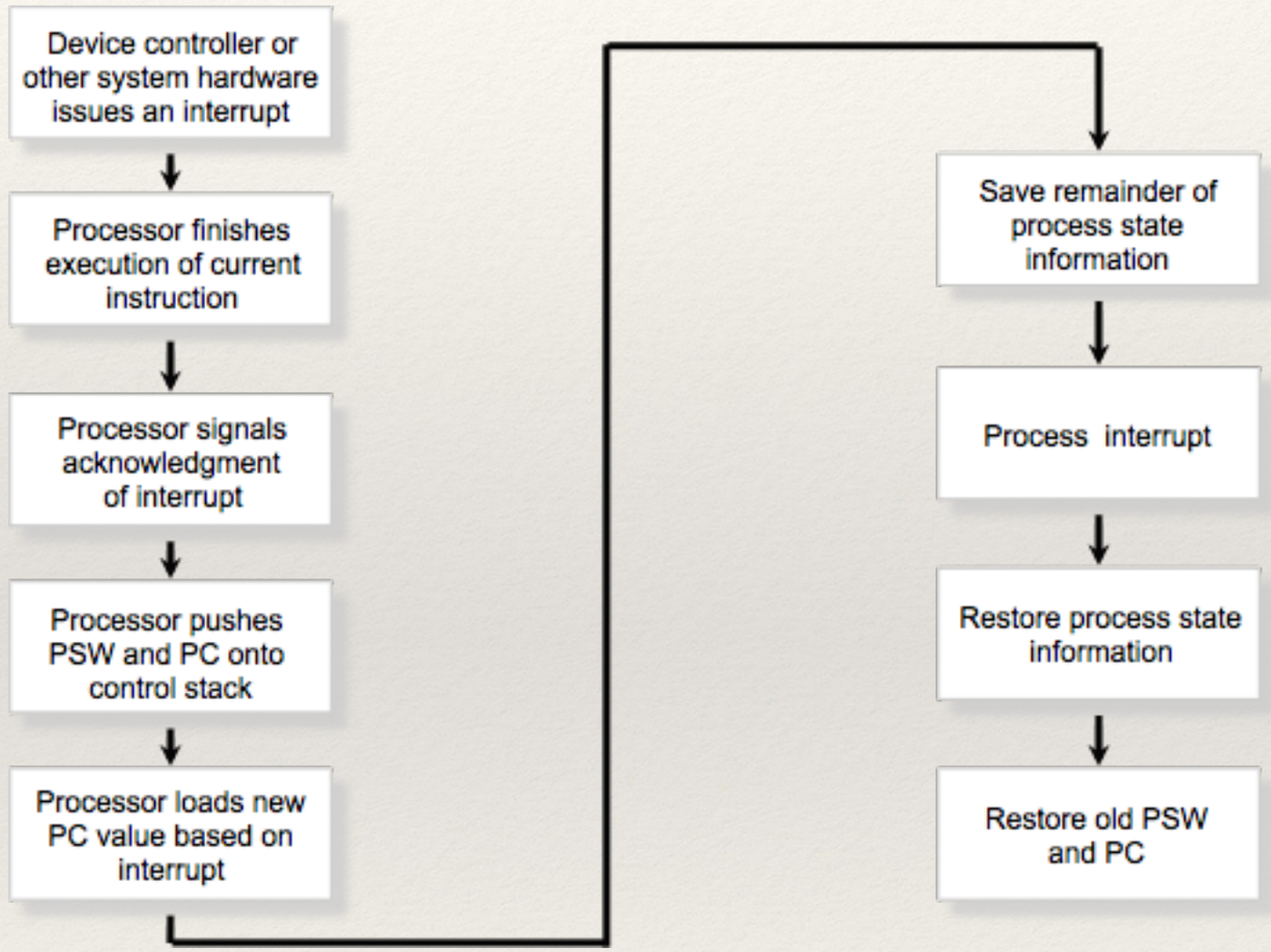
- An *interrupt handler* determines the nature of the interrupt and performs whatever actions are needed.
  - Control is transferred to this program
  - Generally part of the operating system

Interrupt Cycle

- The processor checks for interrupts.
- If there are no interrupts, it fetches the next instruction for the current program.
- If an interrupt is pending, it suspends execution of the current program and executes the interrupt handler.



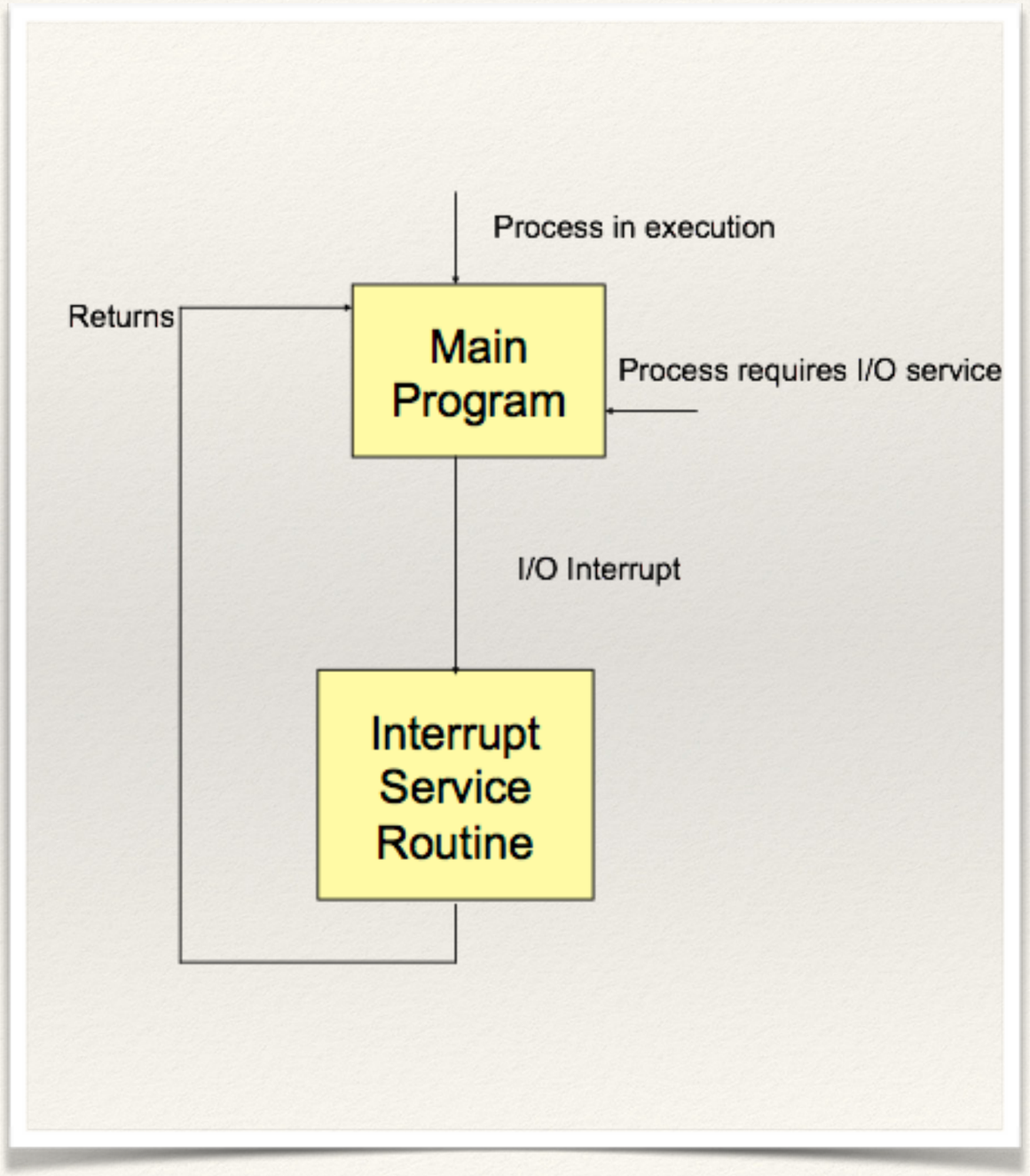
Simple Interrupt Processing



PSW: program status word (status register and program counter)
PC: program counter

General Flow of Interrupts

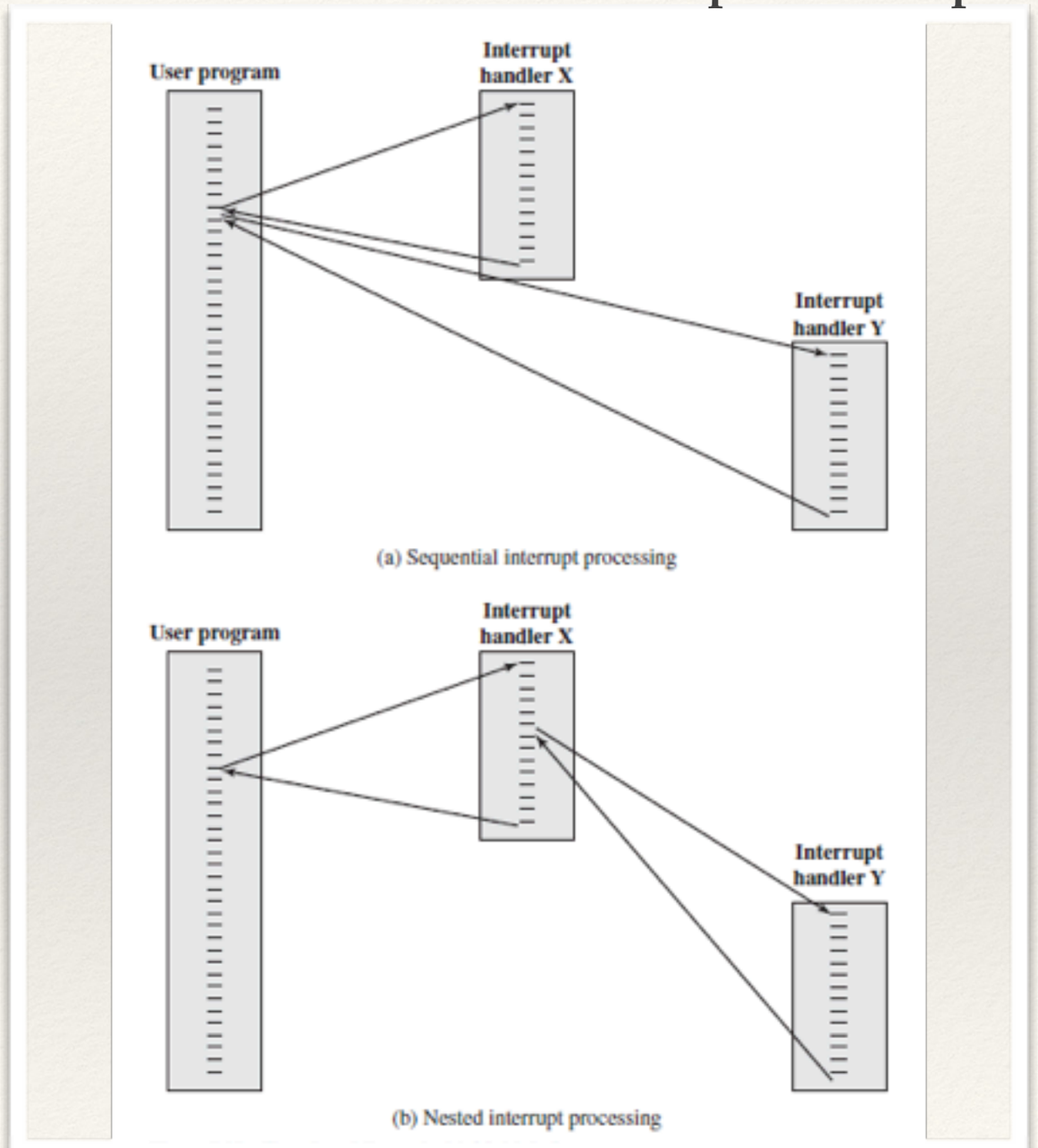
- Disable further interrupts.
- Store current state of program.
- Execute appropriate interrupt handling routine.
- Restore state of program.
- Enable interrupts.
- Resume program execution.



Multiple Interrupts

- **Two Choices**
  - *Disable Interrupts*
    - Disable upon entering an interrupt handler
    - Enable upon exiting
  - *Allow Interrupts*
    - Allow an interrupt handler to be interrupted
    - Priorities?

Transfer of Control with Multiple Interrupts



http://aeridoc.blogspot.ca/

Multiple Interrupts: Sequential Order

- Disable interrupts so processor can complete task.
- Interrupts remain pending until the processor enables interrupts.
- After interrupt handler routine completes, the processor checks for additional interrupts.
- *What happens to the interrupts that occur when disabled?*

Multiple Interrupts: Priorities

- Higher priority interrupts cause lower-priority interrupts to wait.
- This also causes a lower-priority interrupt handler to be interrupted.
- Example: when input arrives from a communication line, it needs to be absorbed quickly to make room for more input.
- *How does this change interrupt handlers?*

Interrupt Handlers

- Interrupt handlers are the routines that are called when an interrupt is detected.
- Interrupt handlers are usually short sections of code that are designed to handle the immediate needs of the device and return control to the operating system or user program.

Interrupt Priority

- When multiple I/O devices are present in an interrupt system, two difficulties must be resolved:
  - How to handle interrupt requests from more than one device at a time
  - Identification of the selected device
- Assigning priority levels to each device means that highest level priorities can be served before lower levels.

Interrupt Summary

- An interrupt is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention.
- An interrupt can occur at any time.
- Hardware and software are needed to support interrupt handling. The hardware must choose the appropriate time in which to interrupt the executing program and transfers control to an interrupt handler.
- The current state of the interrupted program must be saved.

Fleeting or forever…

http://techiesofcjhss.blogspot.ca/p/1.html

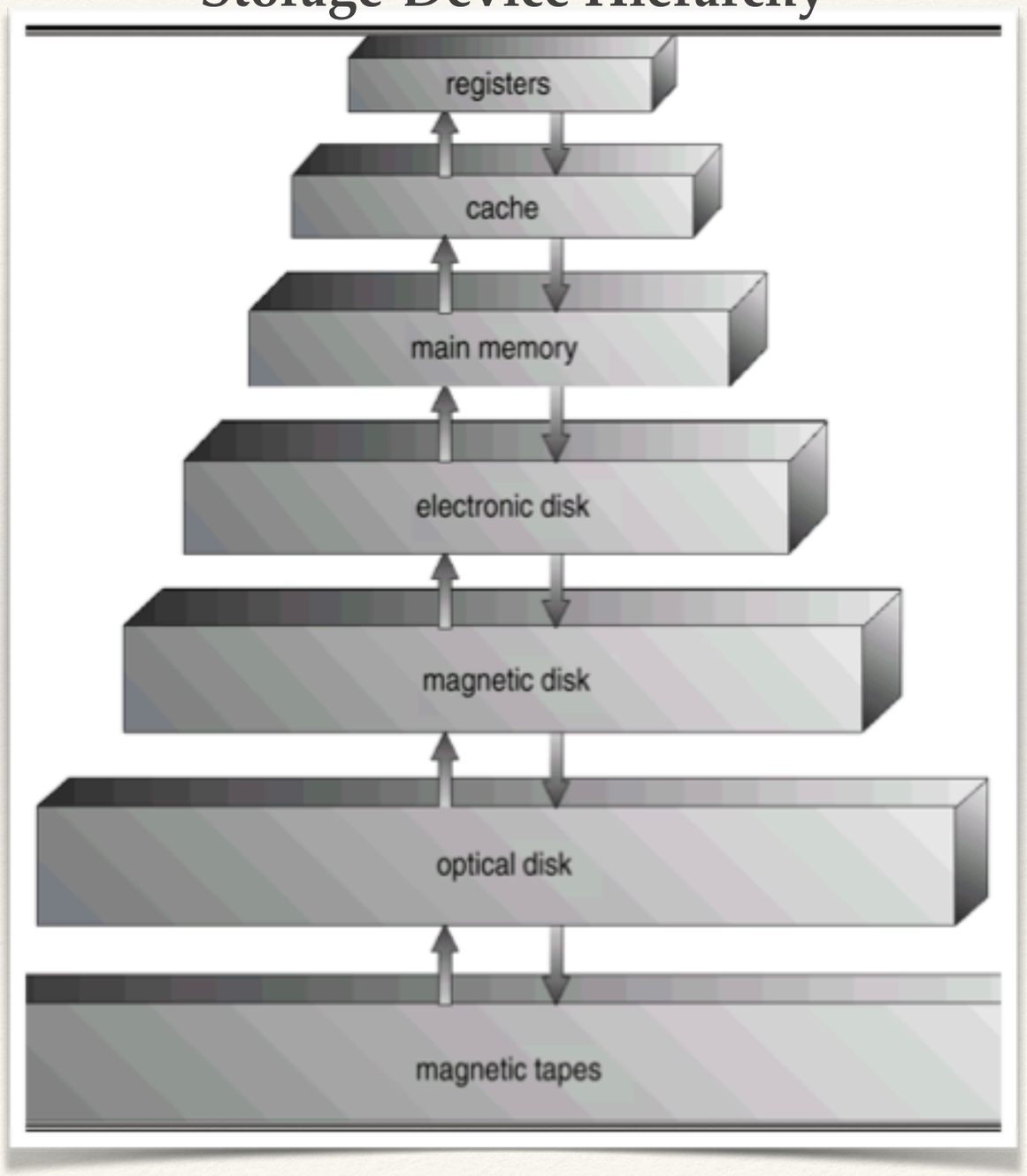
Memory and Storage

Caching

Storage: Comparative Access Timings

- Processor registers: 5 nanoseconds
- SRAM memory: 15 nanoseconds
- DRAM memory: 60 nanoseconds
- Magnetic disk: 10 milliseconds
- Optical disk: (even slower)

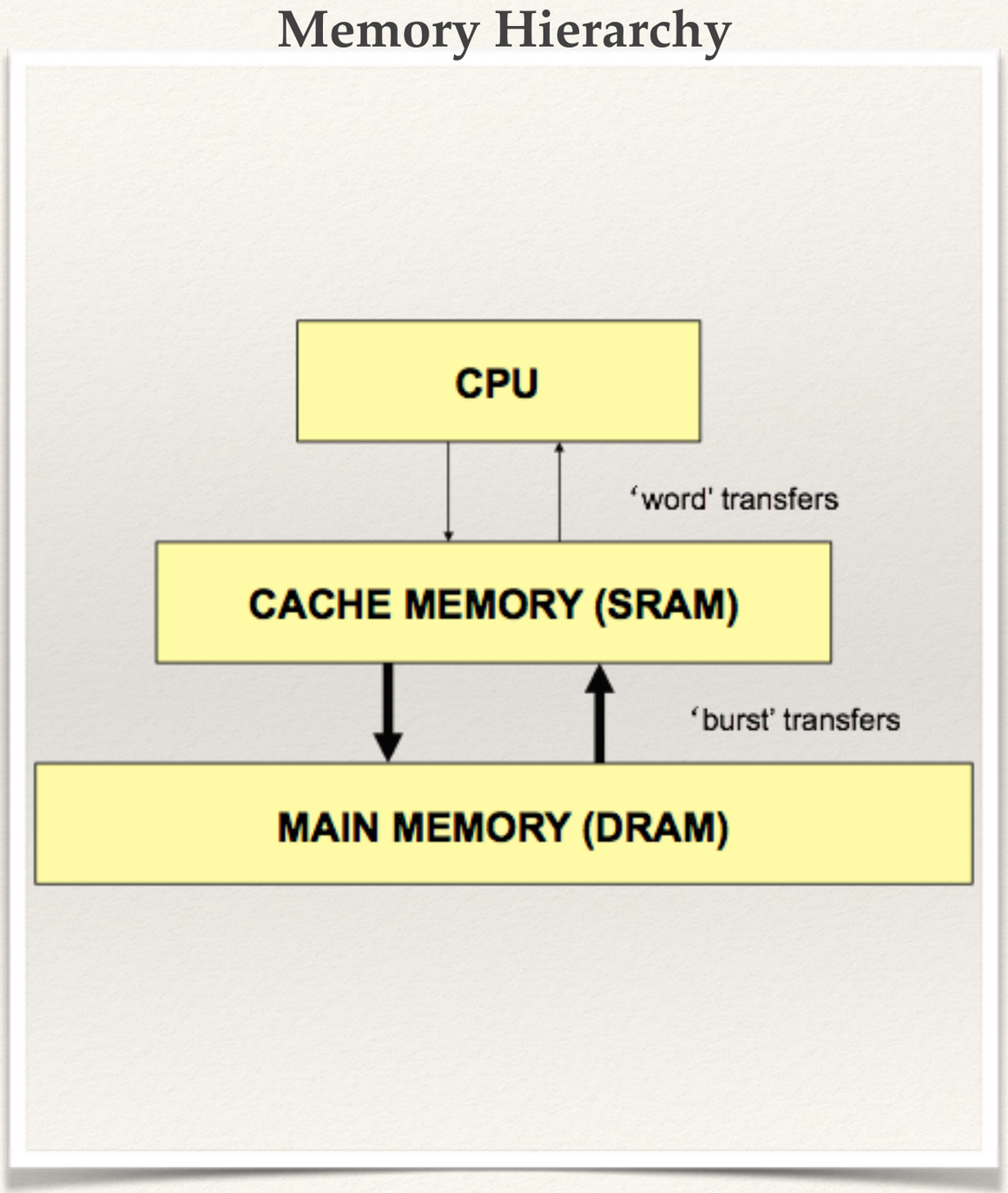
Storage-Device Hierarchy



Operating System Concepts, Chapter 1, Page 11

System Design Decisions

- How much memory?
- How fast?
- How expensive?
- Tradeoffs and compromises
- Modern systems employ a ‘hybrid’ design in which small, fast, expensive SRAM is supplemented by larger, slower, cheaper DRAM



Memory Cache

- **Given** that
  - Processor speed is faster than memory speed
  - Execution/data localizes
- **Cache**
  - Contains a portion of main memory
  - Invisible to operating system
  - Increases the speed of memory
- The CPU first checks the cache and if not found there, the block of memory containing the needed information is moved to the cache.

Cache Design

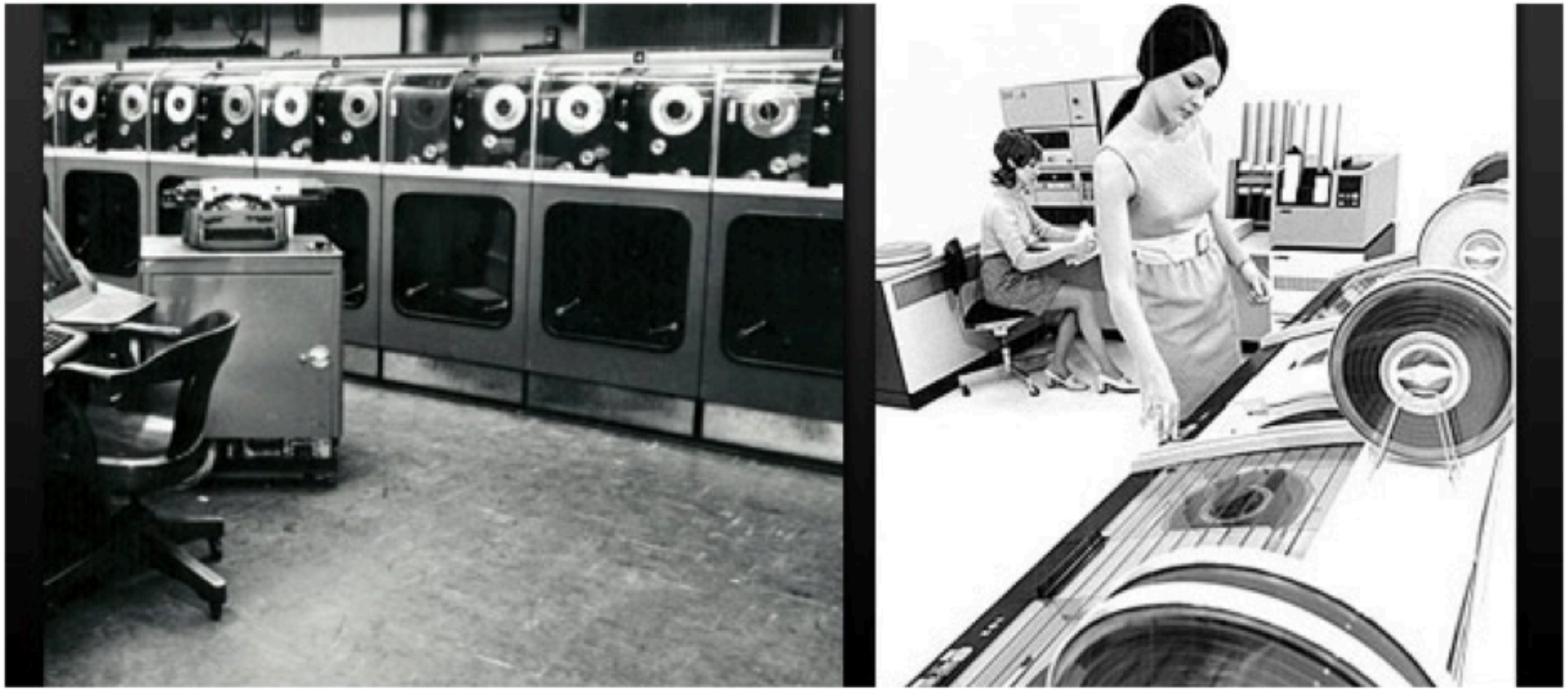
- **Cache size**
  - Small caches have a significant impact on performance.
- **Block size**
  - The unit of data exchanged between cache and main memory
- **Hit** means the information was found in the cache.

Cache Design

- **Mapping Function**
  - Determines which cache location the block will occupy
- **Replacement Algorithm**
  - Determines which block to replace
  - Least-Recently-Used (LRU) algorithm
- **Write Policy**
  - Write a (*dirty*) block of cache back to main memory

Disk Cache

- A portion of main memory is used as a buffer to temporarily to hold data for the disk.
- Disk writes can then be clustered.
- Some data written out may be referenced again. The data are retrieved rapidly from the software cache instead of slowly from disk.



Garbage in…Garbage out…

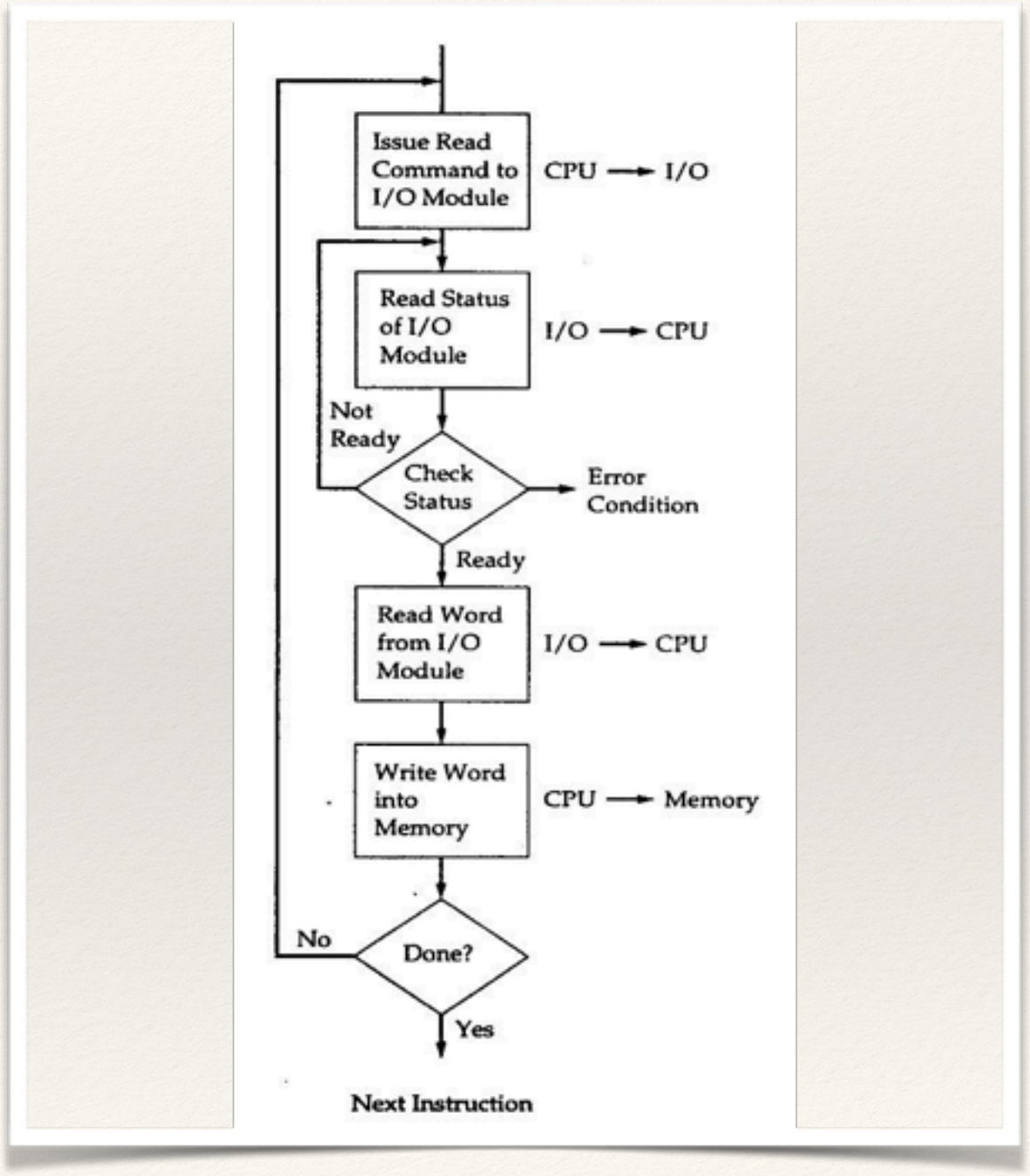
http://www.democraticunderground.com/1014993481

Input/Output

Programmed I/O
Interrupt-Driven I/O
Direct Memory Access
Hard Disks

Programmed I/O

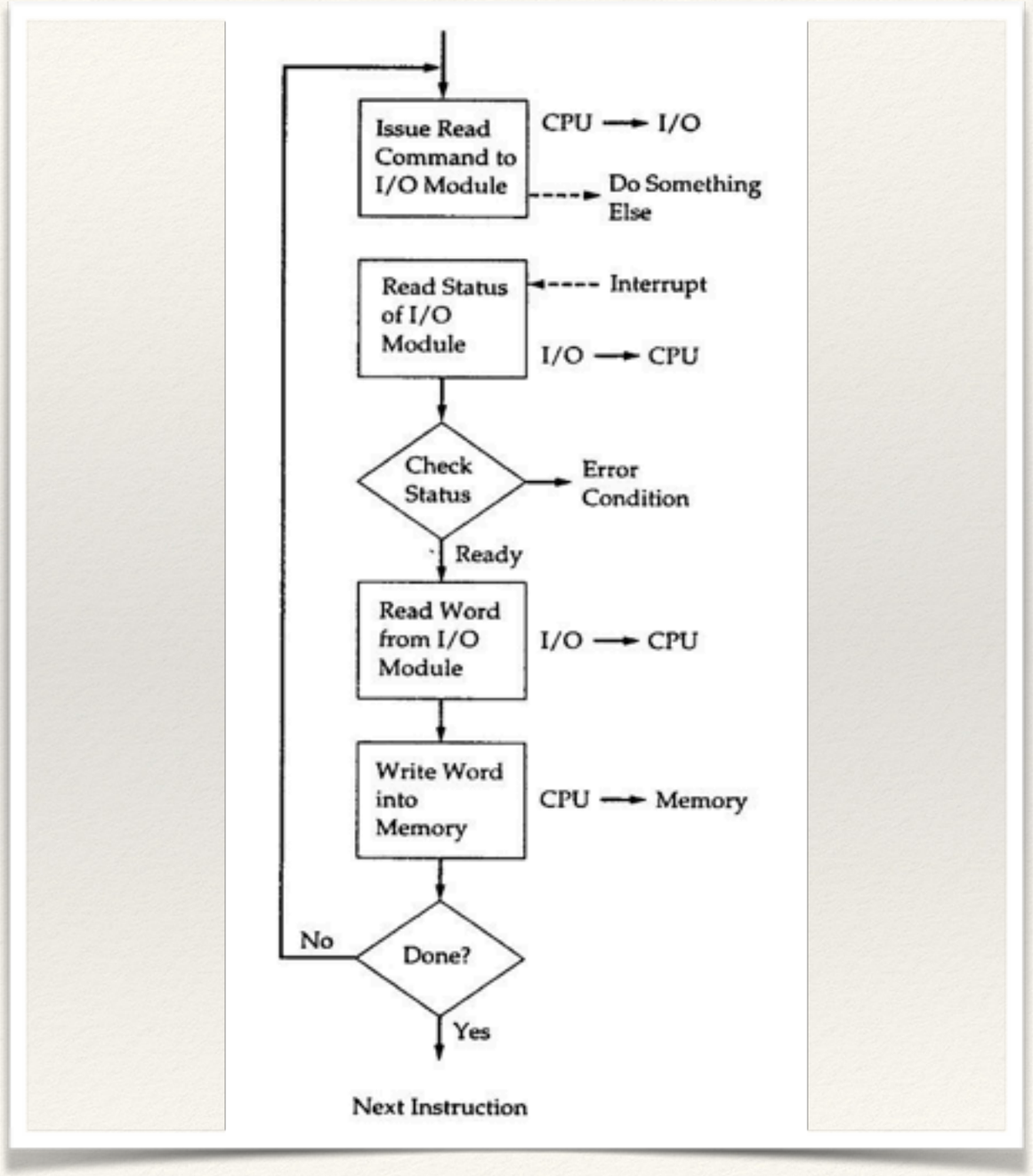
- CPU encounters an I/O instruction while executing a program and issues I/O command to I/O module.
- I/O module performs the action, not the processor.
- Sets appropriate bits in the I/O status register.
- No interrupts occur.
- But CPU must check the status registers periodically to see if I/O operation is complete.



http://cnx.org/contents/d6bcdabc-b579-4633-afa1-4228a7277710@1/Module-13-InputOutput

Interrupt-Driven I/O

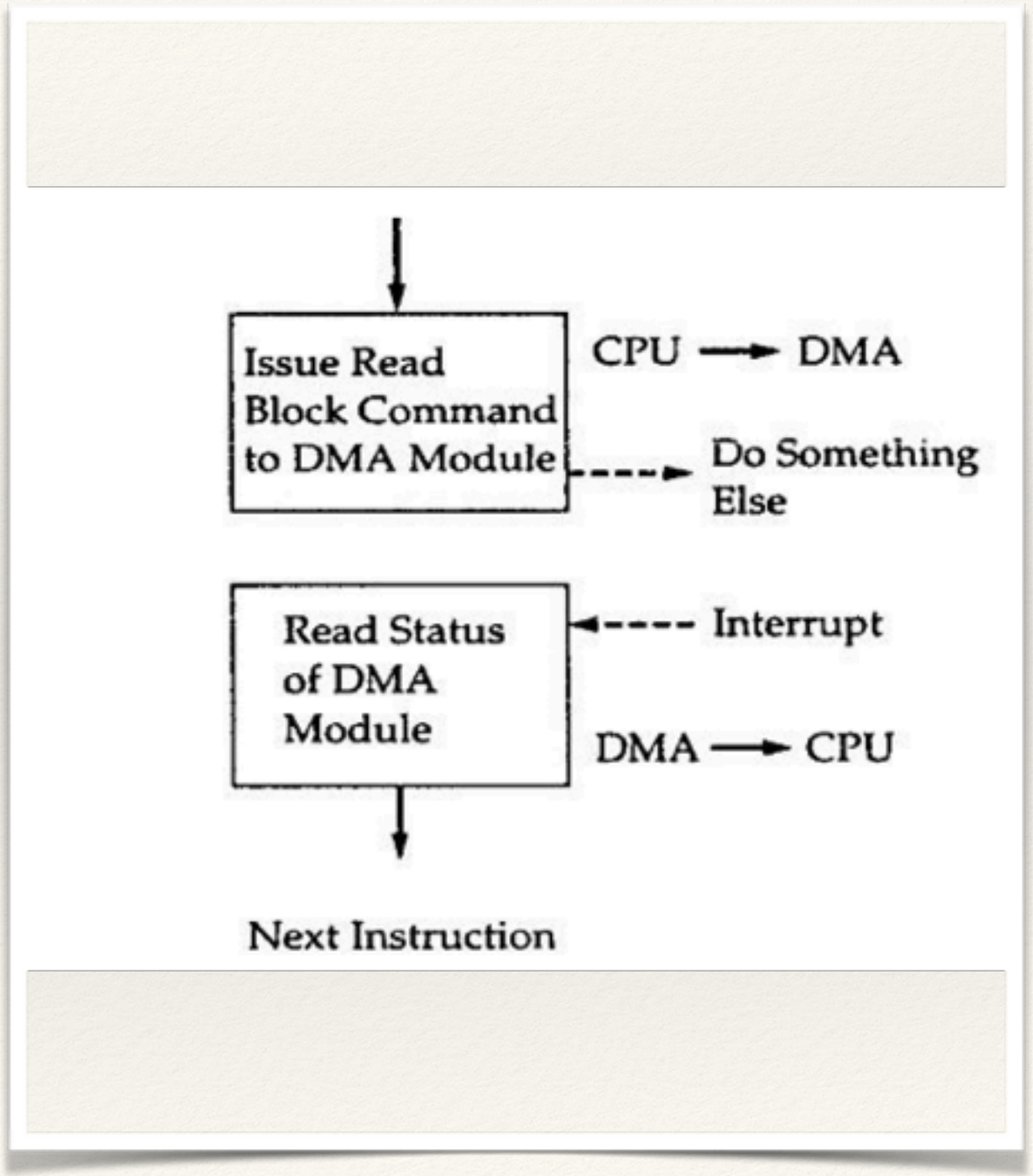
- Processor is interrupted when I/O module is ready to exchange data.
- Processor is free to do other work.
- No needless waiting.
- Consumes a lot of processor time because every word read or written passes through the processor.



http://cnx.org/contents/d6bcdabc-b579-4633-afa1-4228a7277710@1/Module-13-InputOutput

Direct Memory Access

- Transfers a block of data directly to or from memory.
- An interrupt is sent when the task is complete.
- The processor is only involved at the beginning and end of the transfer.



http://cnx.org/contents/d6bcdabc-b579-4633-afa1-4228a7277710@1/Module-13-InputOutput

Hard Drives

- HDD Interfaces
  - SCSI - Small Computer System Interface
  - Serial Attached SCSI (SAS)
  - Fibre Channel (FC)
  - SATA - Serial Advanced Technology Attachment (Serial ATA)
  - IDE - Integrated Drive Electronics (Parallel ATA)

http://www.dpbestflow.org/data-storage-hardware/hard-drive-101

Hard Disks

- There exist about 40 different commands (*e.g.* read, write, seek, format, sleep, *etc*.)
- Some are *mandatory*, others optional.
- Example: the '*Identify Drive*' command
  - It provides information on the disk's geometry and other operational characteristics.
  - It identifies the disk's manufacturer and it provides a unique disk serial-number.

Disk Command Protocol

- Disk Commands typically have 3 phases:
  - *Command Phase*: CPU issues a command
  - *Data Phase*: data moves to/from buffer
  - *Result Phase*: CPU reads status/errors

To serve and protect…

Protection

CPU
Memory
I/O

http://es.slideshare.net/luisbremo/significado-vida-privada

Hardware Protection

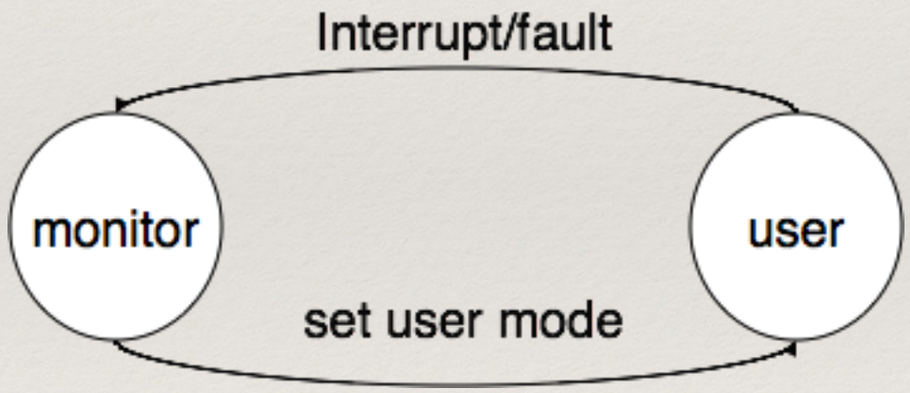
- **Why** protect hardware and from **what**?
  - Shared hardware resources – memory, disk, …
  - Errant programs

Hardware Protection

- **How?**
  - CPU provides 2 modes of operation
    - *User Mode* (non-privileged)
    - *Supervisor/Monitor/OS mode* (privileged)
- Privileged instructions can only be executed in Monitor mode
  - All I/O instructions are privileged

Dual Mode Operation

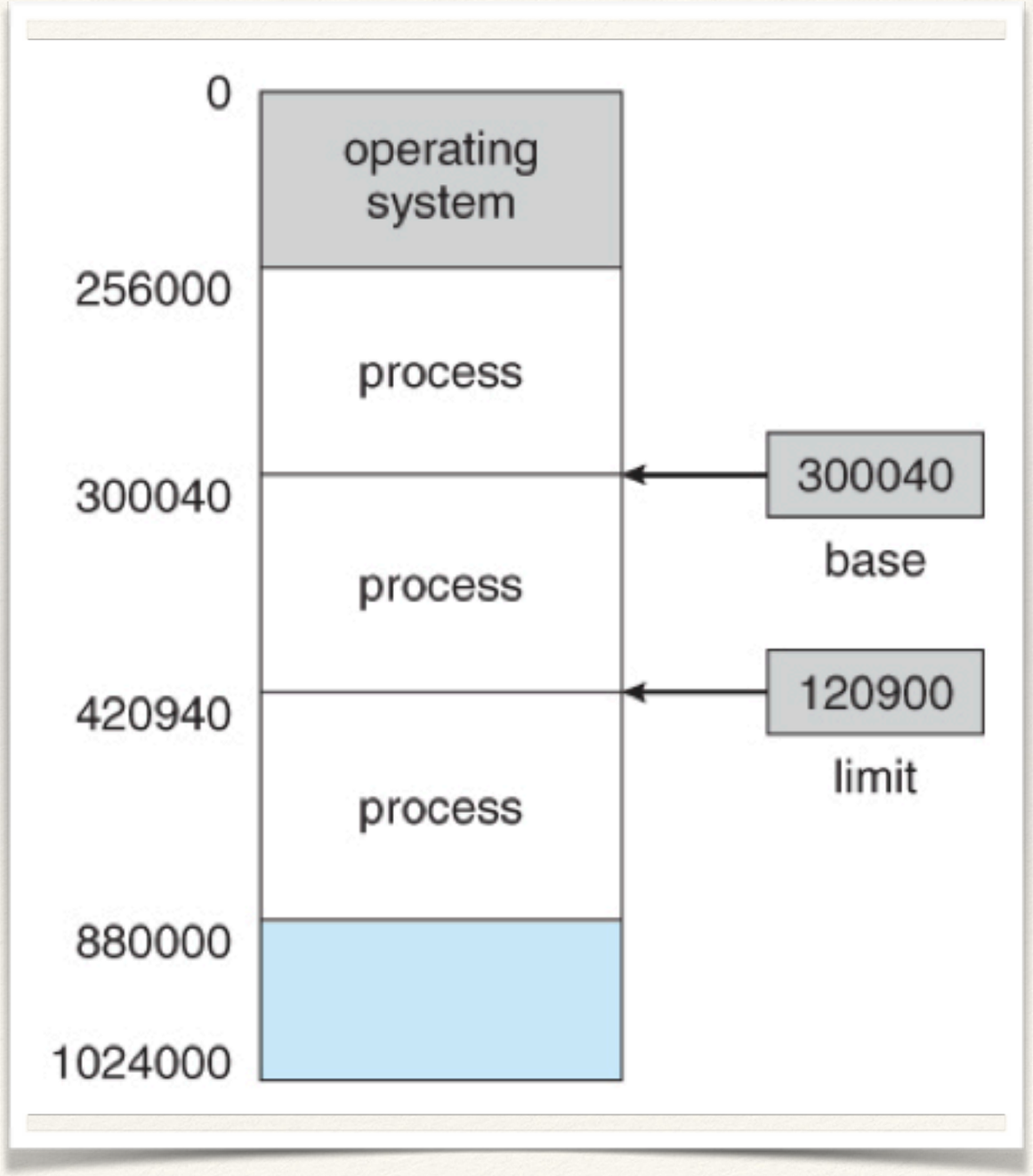
- Mode bit added to computer hardware to indicate the current mode: Monitor (0) or User (1).
- When an interrupt or fault occurs hardware switches to Monitor mode.



Privileged instructions can be issued **only** in *Monitor* mode.

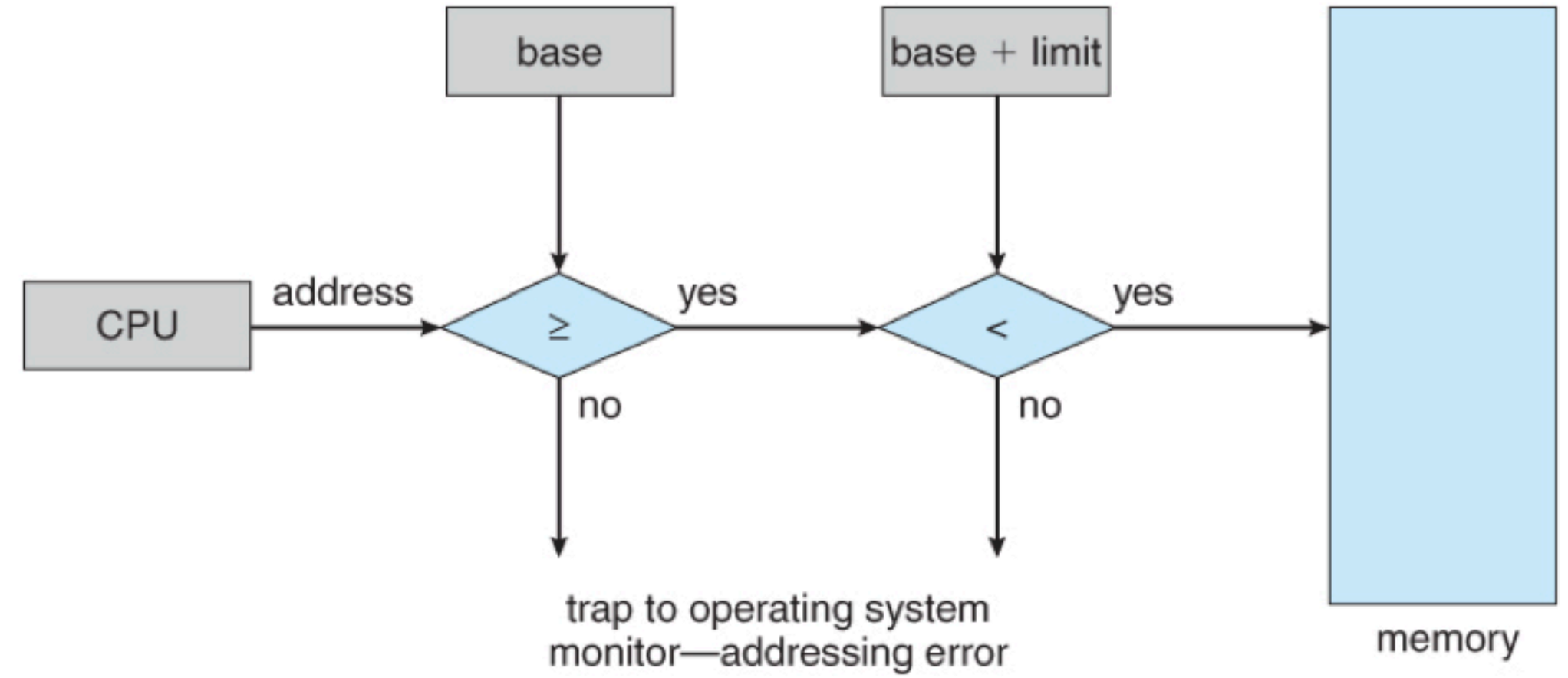
Memory Protection

- We must not allow a program to access memory outside of its space.
- **How**?
  - Add two registers
    - **Base**
    - **Limit**
- Use the registers to check every reference.



https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/8_MainMemory.html

Hardware Address Protection with Base and Limit Registers



https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/8_MainMemory.html

I/O Protection

- All I/O instructions are *privileged* instructions.
- Must ensure that a user program could never gain control of the computer in *Monitor* mode.
- If all I/O instructions are protected, how do you perform input or output?

I/O Protection

- **System Calls**
  - Often the call is a **trap** (soft interrupt) to a **ISR** (Interrupt Service Routine)
  - Control is passed to the ISR which sets the mode bit to *Monitor* mode
  - ISR verifies that parameters are correct
  - ISR executes request, resets mode
  - Control is returned to user program