CS 416
Operating Systems

Prof Badri Nath

http://www.cs.rutgers.edu/~badri/416.html

Dept. of Computer Science

Rutgers University

Computer System Overview

Chapter 1

Why are operating systems interesting?

- An Operating System makes the computing power available to users by controlling the hardware resources
- Services used by application programmers
- Provides an easy way to program a machine
- Reality-soul of any computer
  - Linux, palm-OS, WIN/NT, UNIX flavors

Why are they complex

- Code needed to control variety of devices
- Calls for detailed knowledge of hardware
- Provides various services for applications
  - Examples? –
- Components designed for performance

Course Goals

- Understand “what is an OS?”
  - Understand key OS components; management of resources
  - An OS as a control program, resource allocator and an environment
- Present interesting algorithms used in OS
- To outline some important features of modern OS
  - Linux, WIN/NT, Unix flavors, palm OS

Historical perspective

- Oth generation
  - Single users, toggle switches loaders
- 1st generation
  - 1950's batch operating systems
- 2nd generation
  - Early 60's multiprogramming
  - Efficient use of resources (memory + CPU)
- 3rd generation
  - mid 60s to mid 70s
  - Timesharing, interactive response
- 4th generation
  - Mid 70s to mid 80s
  - Smaller compact systems minis

Historical perspective

- 5th generation
  - Mid 80s to mid 90s
  - PCs, WINDOWS, LINUX
  - Networking, client server systems
- Current generation
  - Small compact OS. Cluster computing, NOW (network of workstations). Emphasis on availability
  - OS for special devices
  - Focus on communication and entertainment
  - OS for webTV, Replay TV, Cellular phones, palm
  - OS in your car

Current trends in OS

- Flexibility VS performance
  - Plug N play
- Exploit architecture features on which OS executes
- OS for specialty devices
- Consider application requirements

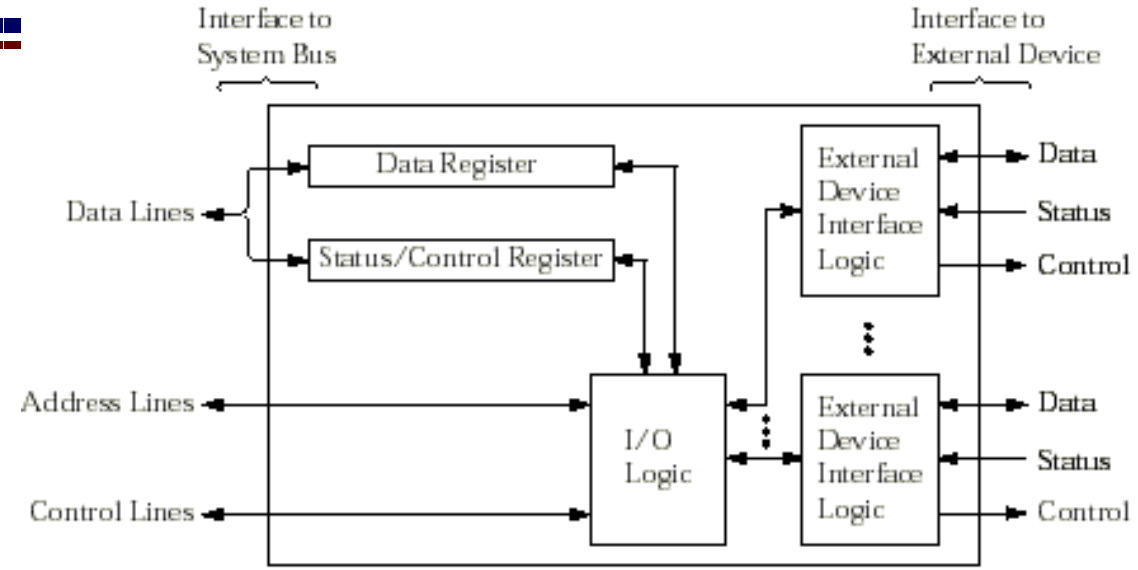
Basic Components

- Processor (CPU)
- Main Memory (aka physical memory, aka primary memory)
  - holds data in code
- I/O modules (I/O controllers, I/O channels, I/O processors...)
  - hardware (with registers called I/O ports) that moves data between cpu and peripherals like:
    - secondary memory devices (eg: hard disks)
    - keyboard, display...
    - communications equipment
- System interconnection (ie: Buses)
  - communication among processors, memory, and I/O modules

What's happening with CPU

- 8088 .5 MIPS
- 286 1 MIPS
- 386 5 to 10 MIPS
- 486 20 to 50 MIPS
- Pentium, power PC alpha 100 MIPS
- P6 200 to 300 MIPS
- Pentium III or AMD athlon 600 to 700 MIPS
  - 1 GHZ clock (10^{-9} sec)
  - http://www.intel.com/mobile/performance/pentiumiii/index.htm

I/O Module Structure



- Data to/from system bus are buffered in data register(s)
- Status/Control register(s) holds
  - current status information of the I/O operation
  - current control information from CPU
- I/O logic interact with CPU via control bus
- Contains logic specific to the interface of each device

CPU Registers (fast memory on cpu)

- Control & Status Registers
  - Generally not available to user programs
  - some used by CPU to control its operation
  - some used by OS to control program execution
- User-visible Registers
  - available to system (OS) and user programs
  - holds data, addresses, and some condition codes

Examples of Control & 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)
  - A register or group of registers containing:
    - condition codes and status info bits
    - Interrupt enable/disable bit
    - Supervisor(OS)/user mode bit

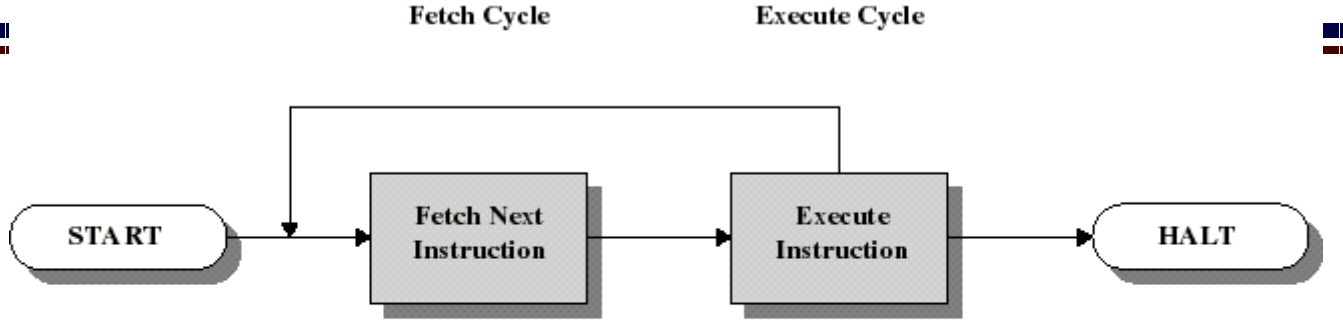
User-Visible Registers

- Data Registers
  - can be assigned by the user program to perform operations on data
- Address Registers
  - contain memory address of data and instructions
  - may contain a portion of an address that is used to calculate the complete address

User-Visible Registers

- Condition Codes or Flags
  - Bits set by the processor hardware as a result of operations
  - Can be accessed by a program but not changed directly
  - Examples
    - sign flag
    - zero flag
    - overflow flag

The Basic Instruction Cycle

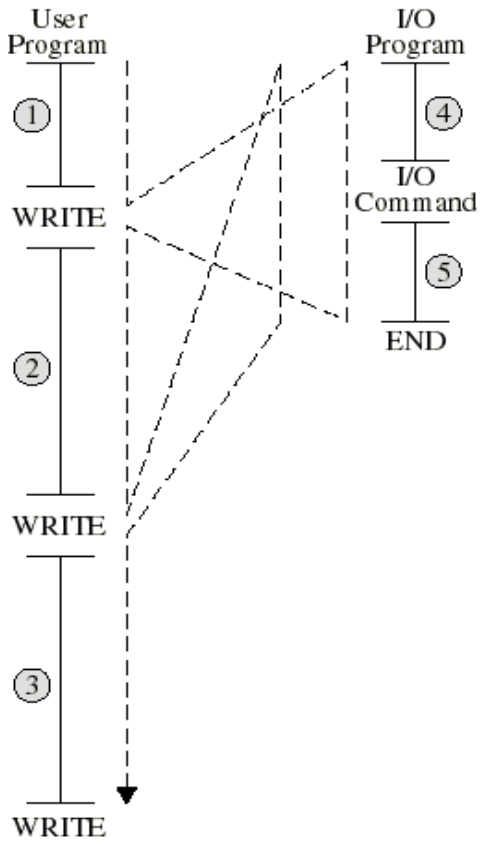


Basic Instruction Cycle

- The CPU fetches the next instruction (with operands) from memory.
- Then the CPU executes the instruction
- Program counter (PC) holds address of the instruction to be fetched next
- Program counter is automatically incremented after each fetch

Then CPU must wait for I/O to complete!

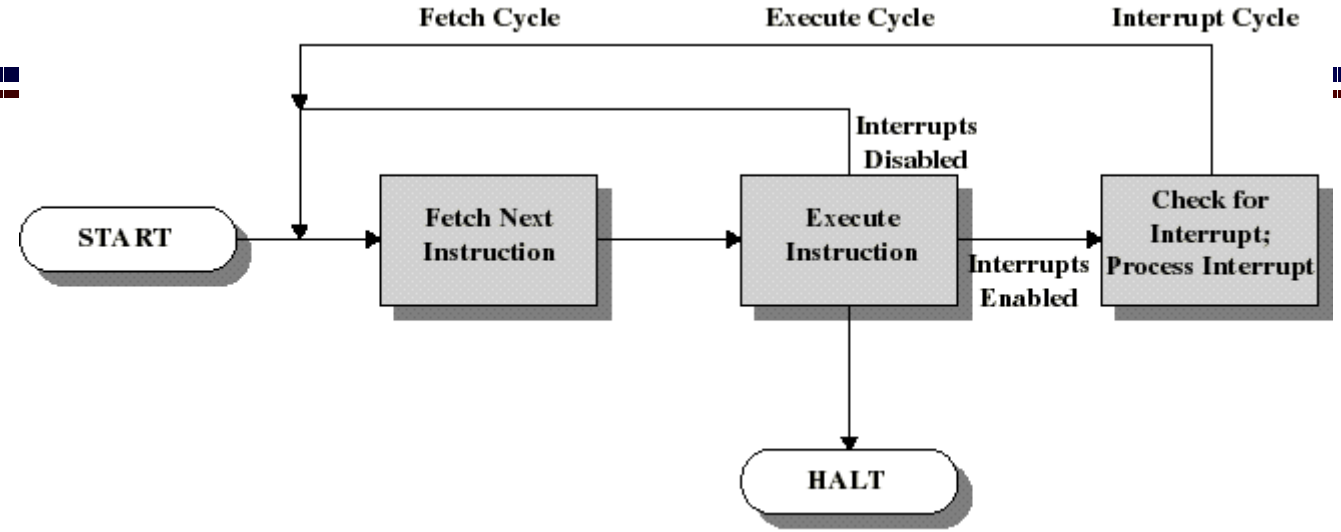
- WRITE transfer control to the printer driver (I/O pgm)
- I/O pgm prepare I/O module for printing (4)
- CPU has to WAIT for I/O command to complete
- Long wait for a printer
- I/O pgm finishes in (5) and report status of operation



Interrupts

- Computers now permit I/O modules to INTERRUPT the CPU.
- For this the I/O module just assert an interrupt request line on the control bus
- Then CPU transfer control to an Interrupt Handler Routine (normally part of the OS)

Instruction Cycle with Interrupts!

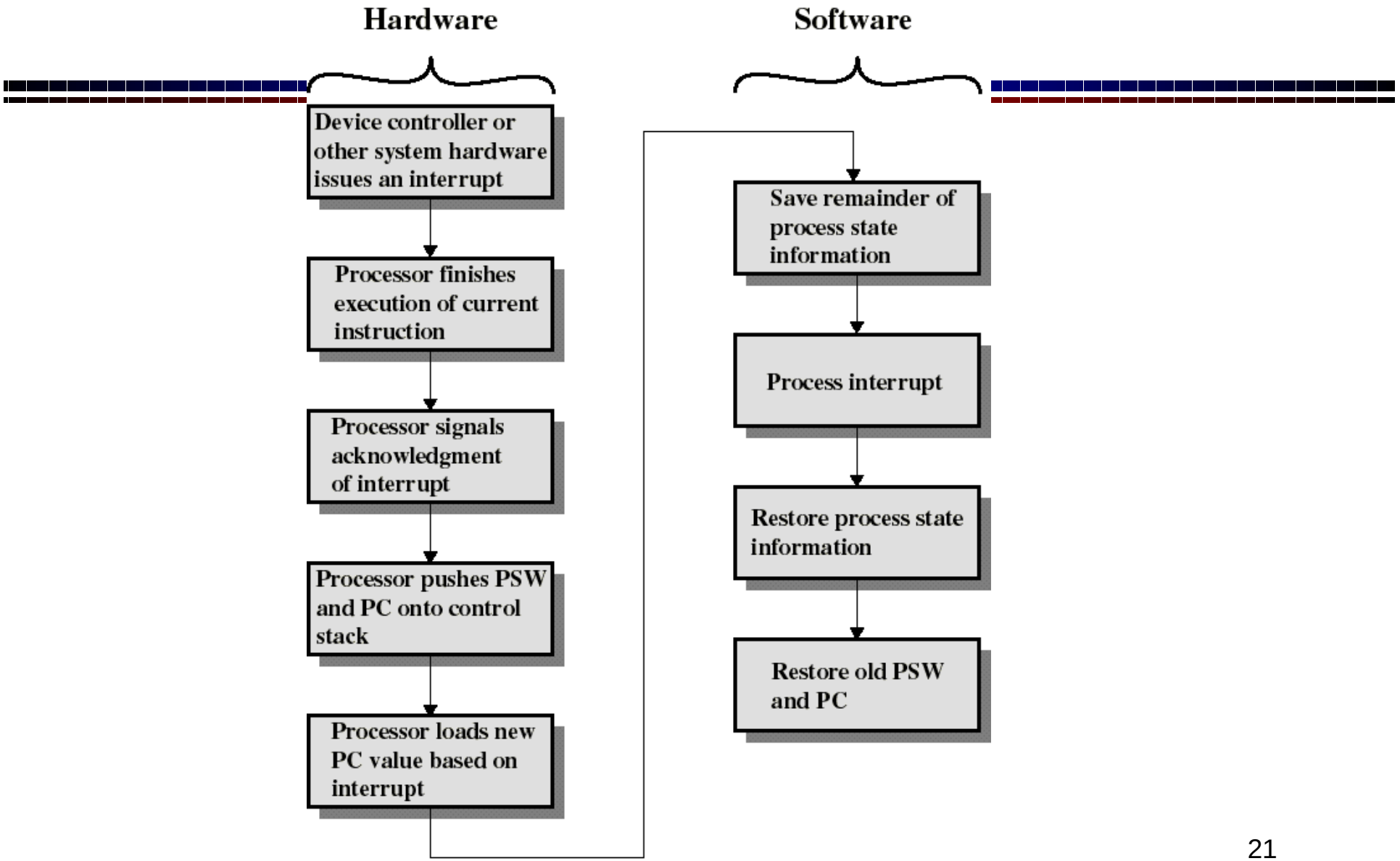


- CPU checks for interrupts after each instruction
- If no interrupts, then fetch the next instruction for the current program
- If an interrupt is pending, then suspend execution of the current program, and execute the interrupt handler

Interrupt Handler

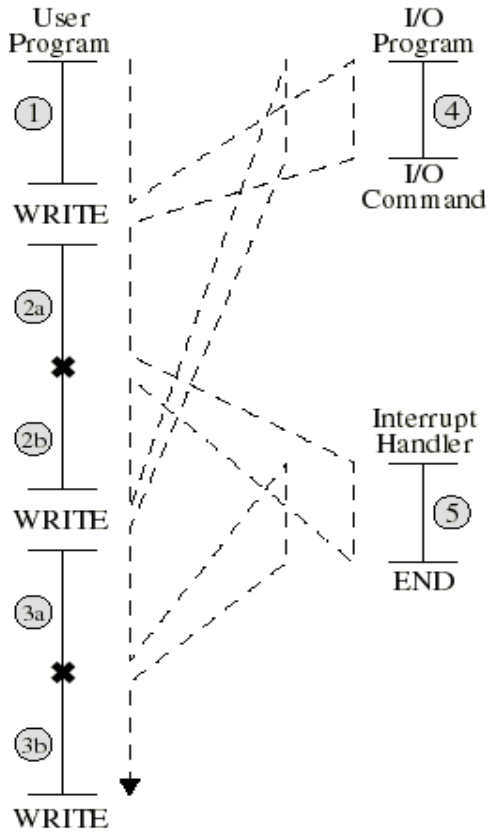
- Is a program that determines nature of the interrupt and performs whatever actions are needed
- Control is transferred to this program
- Control must be transferred back to the interrupted program so that it can be resumed from the point of interruption
- This point of interruption can occur anywhere in the program
- Thus: must save the state of the program (content of PC + PSW + registers + ...)

Simple Interrupt Processing



Interrupts improve CPU usage

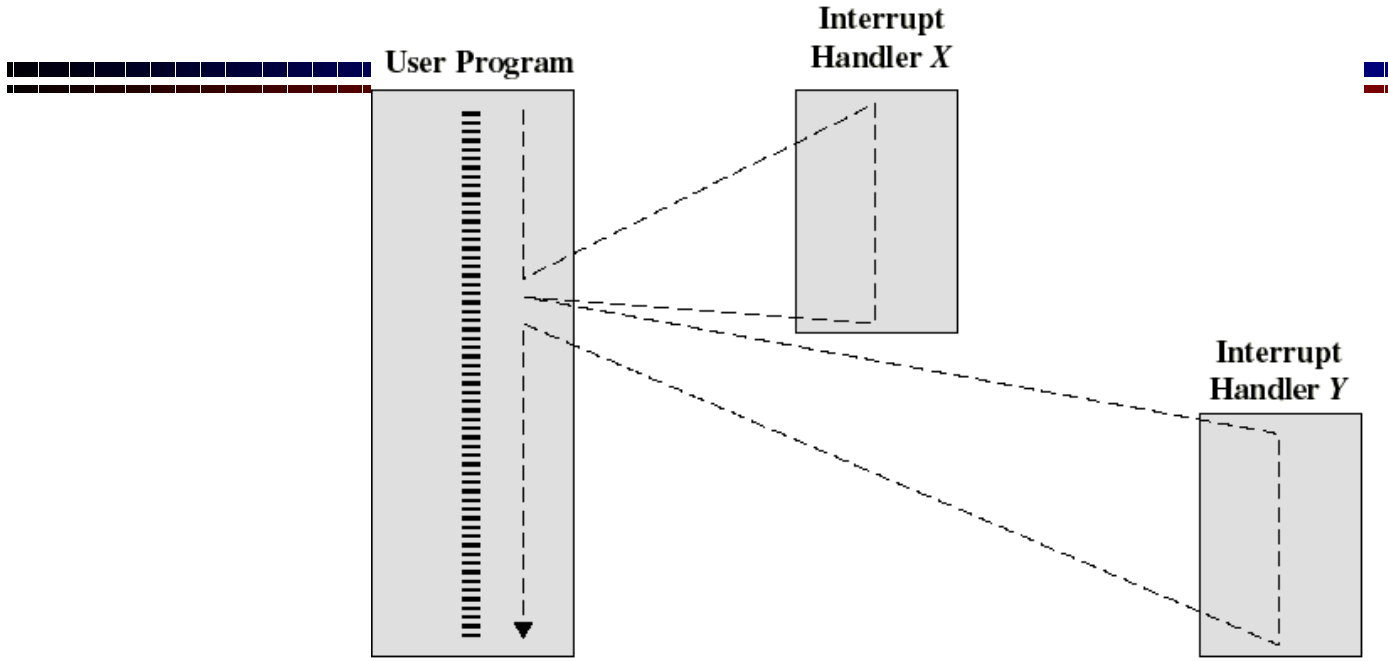
- I/O pgm prepares the I/O module and issues the I/O command (eg: to printer)
- I/O pgm branches to user pgm
- User code gets executed during I/O operation (eg: printing): no waiting
- User pgm gets interrupted (x) when I/O operation is done and branches to interrupt handler to examine status of I/O module
- Execution of user code resumes



Classes of Interrupts

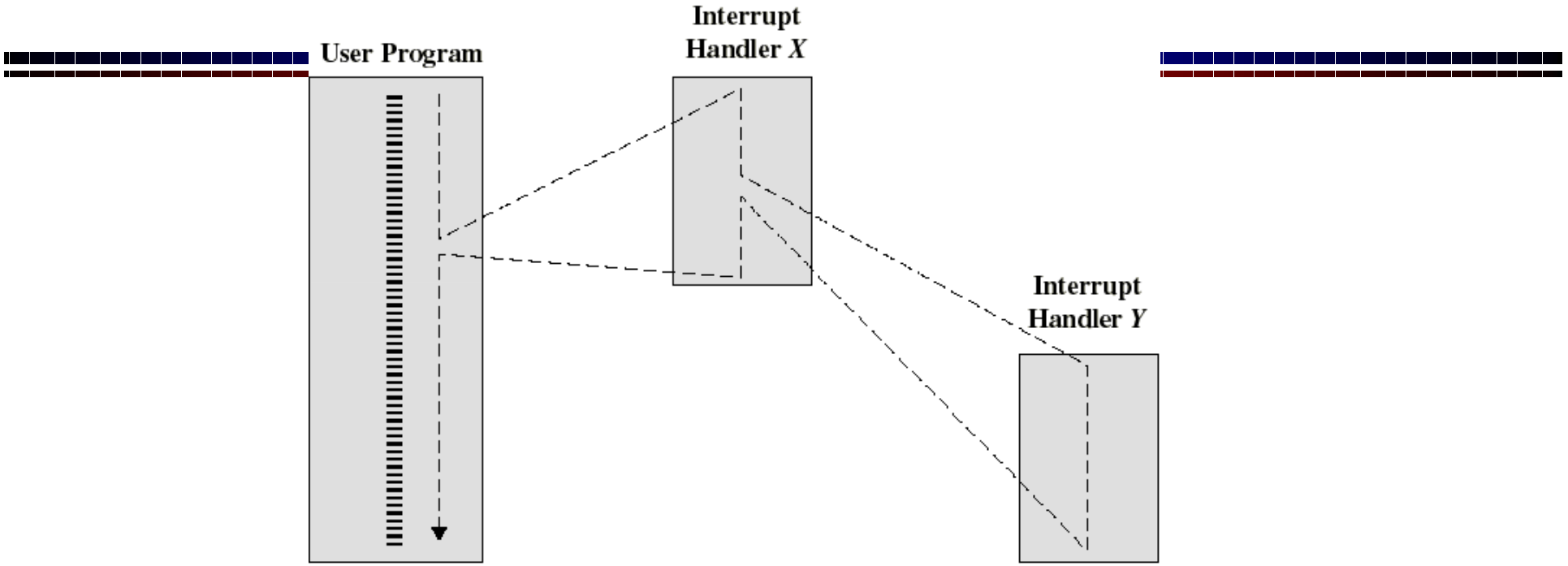
- I/O or Hardware Interrupts
  - signals normal completion of operation or error
- Program Exception
- An unusual condition resulting from the execution of an instruction – caused by the running process (need attention of OS)
  - Traps – occurs at end of instruction (special instructions or overflows)
- Other types of exceptions
- try to execute illegal instruction (beginning of instruction)
  - Fault Vs abort (can occur in the middle of executing an instruction)
  - Fault: exception that can be cleared (Page fault)
  - Abort: exception that cannot be cleared (reference outside user's memory space)
- Timer
  - preempts a programm to perform another task
- Hardware failure (eg: memory parity error)
- Software interrupts –signals (independent of running process)

Multiple interrupts: sequential order



- Disable interrupts during an interrupt
- Interrupts remain pending until the processor enables interrupts
- After interrupt handler routine completes, the processor checks for additional interrupts

Multiple Interrupts: priorities



- Higher priority interrupts cause lower-priority interrupts to wait
- Causes a lower-priority interrupt handler to be interrupted
- Example: when input arrives from communication line, it needs to be absorbed quickly to make room for more input

Multiprogramming

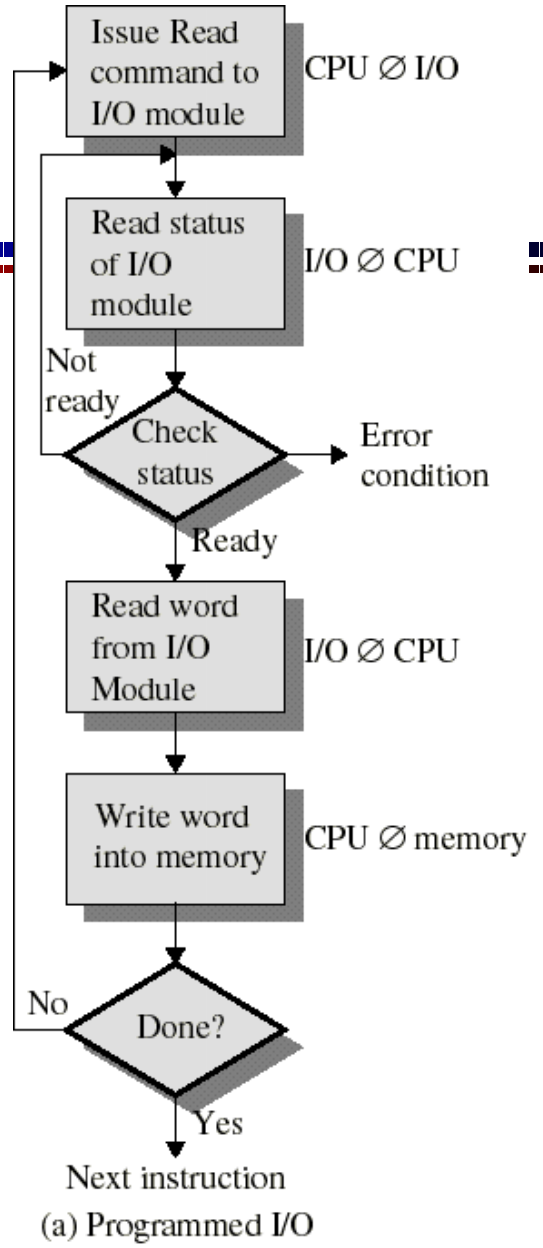
- When a program reads a value on a I/O device it will need to wait for the I/O operation to complete
- Interrupts are mostly effective when a single CPU is shared among several concurrently active processes.
- The CPU can then switch to execute another program when a program waits for the result of the read operation.

I/O communication techniques

- 3 techniques are possible for I/O operation
  - Programmed I/O
    - Does not use interrupts: CPU has to wait for completion of each I/O operation
  - Interrupt-driven I/O
    - CPU can execute code during I/O operation: it gets interrupted when I/O operation is done.
  - Direct Memory Access
    - A block of data is transferred directly from/to memory without going through CPU

Programmed I/O

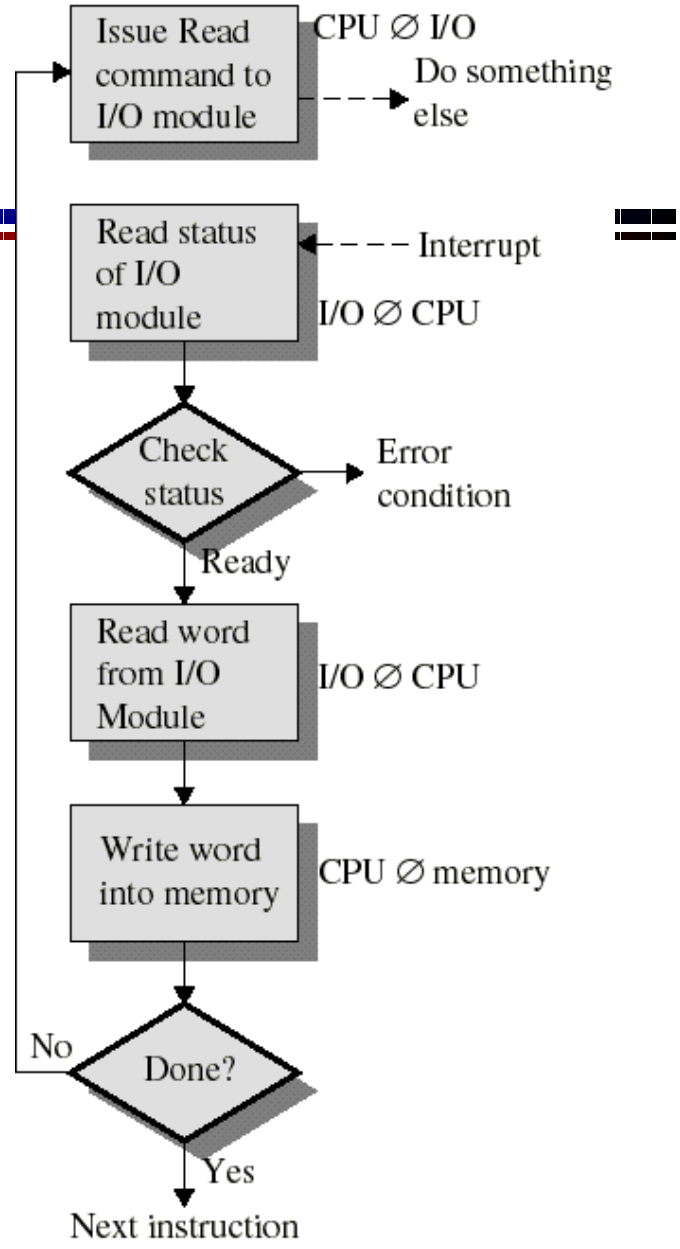
- I/O module performs the action, on behalf of the processor
- But the I/O module does not interrupt the CPU when I/O is done
- Processor is kept busy checking status of I/O module



(a) Programmed I/O

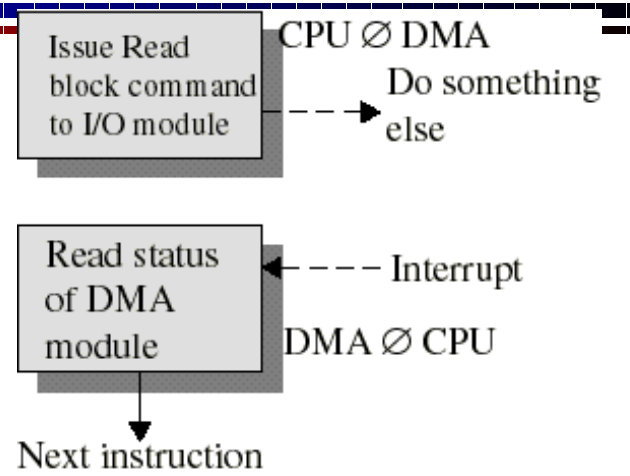
Interrupt-Driven I/O

- Processor is interrupted when I/O module 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 and requires an interrupt

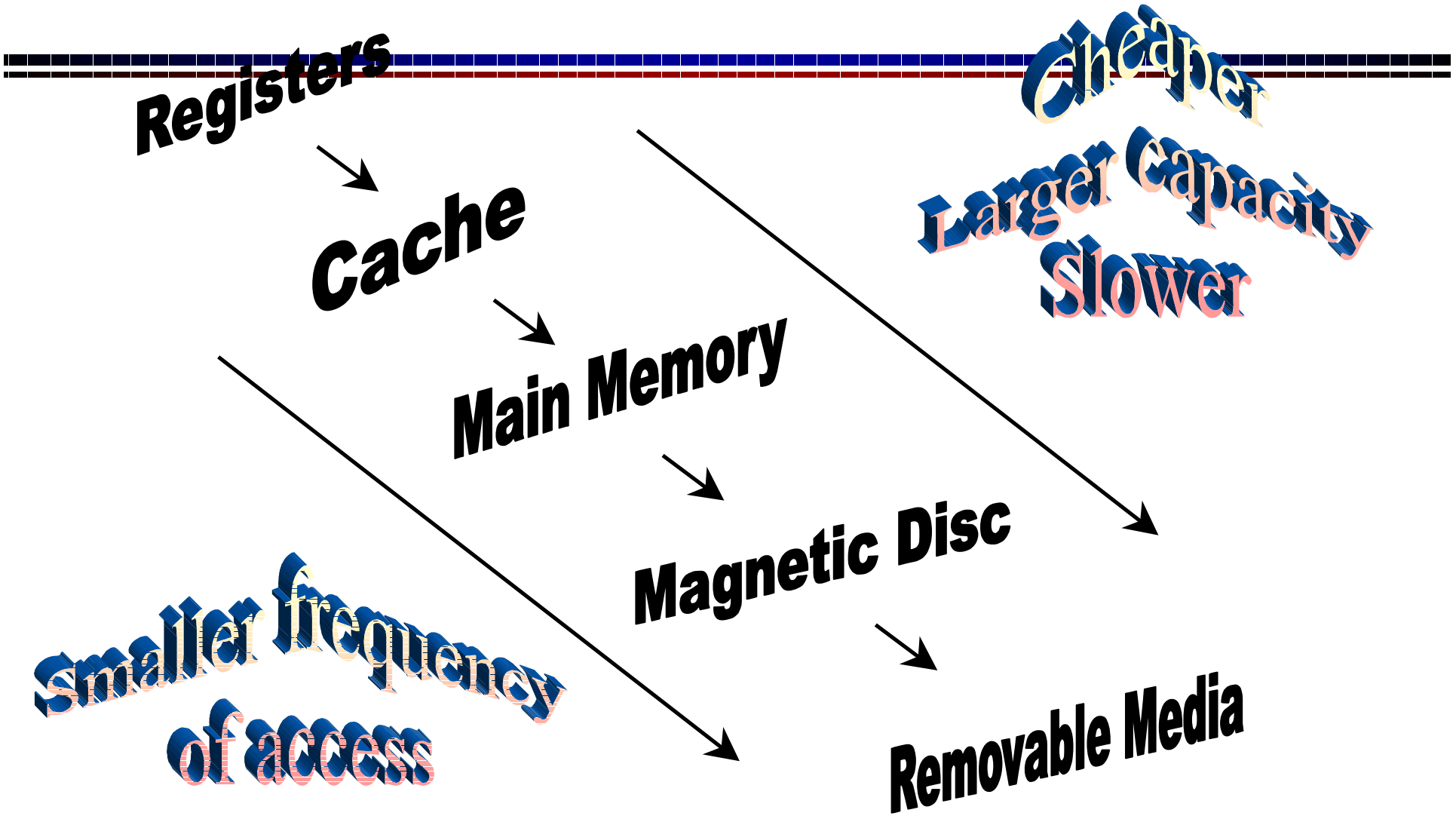


Direct Memory Access

- CPU issues request to a DMA module (separate module or incorporated into I/O module)
- DMA module transfers a block of data directly to or from memory (without going through CPU)
- An interrupt is sent when the task is complete
- The CPU is only involved at the beginning and end of the transfer
- The CPU is free to perform other tasks during data transfer



Memory Hierarchy

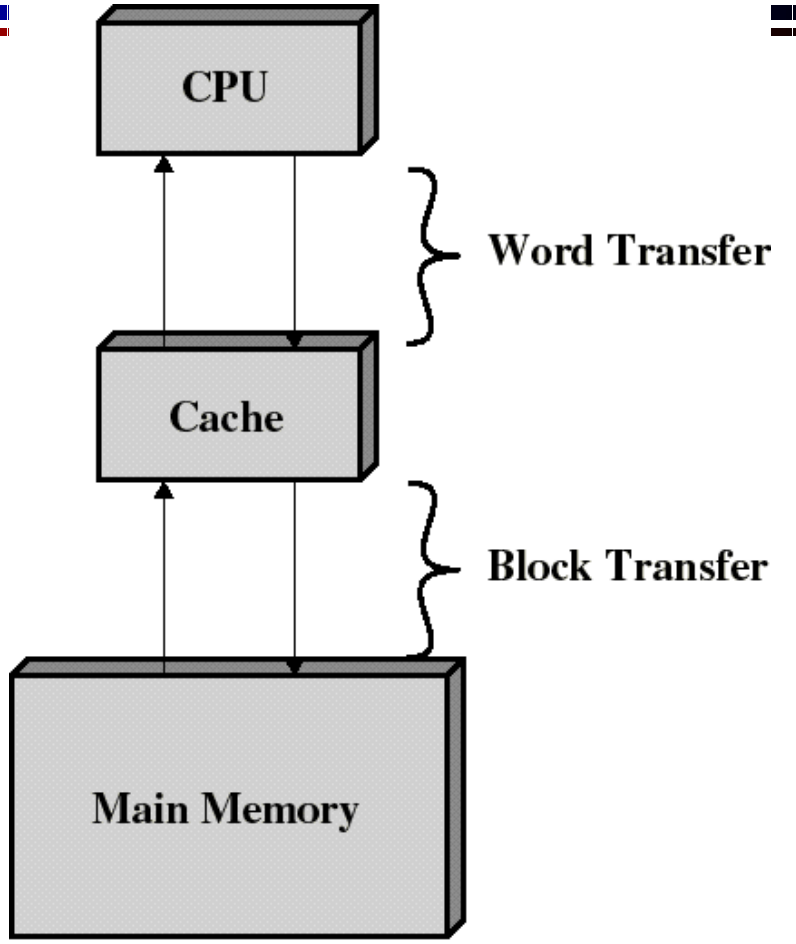


What's happening with memory

- SRAM few nsecs, 10$ to 40$/Mbyte
- DRAM 100 nsecs, 1 to 2$/Mbyte
- Disk few mses access time
  - 20G for 300$ (SCSI) or 60G for 300$ (IDE)
- Flash RAM (storage safe when power is off)
  - 32 M costs 145$
- CD-ROM few msecs, 700 MB, 1$
  - ZIP disks 10$ per 100 MB

Cache Memory

- Small cache of expensive but very fast memory interacting with slower but much larger memory
- Invisible to OS and user programs but interact with other memory management hardware
- Processor first checks if word referenced to is in cache
- If not found in cache, a block of memory containing the word is moved to the cache



Locality of reference

- Memory reference for both instruction and data tend to cluster over a long period of time.
- Example: once a loop is entered, there is frequent access to a small set of instructions.

- Hence: once a word gets referenced, it is likely that nearby words will get referenced often in the near future.
- Thus, the hit ratio will be close to 1 even for a small cache.

What's happening with network

- Ethernet 10 to 100 Mbps
- Fast ethernet 70 to 100 Mbps
- Switched ATM or gigabit ethernet 100 to 1000 Mbps
- Wireless
  - Cellular 9.2 Kbps
  - Wireless LAN 2 to 10 Mbps