

---

# Input/Output: Polling and Interrupts

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Outline

---

- **I/O Background**

- **Polling**

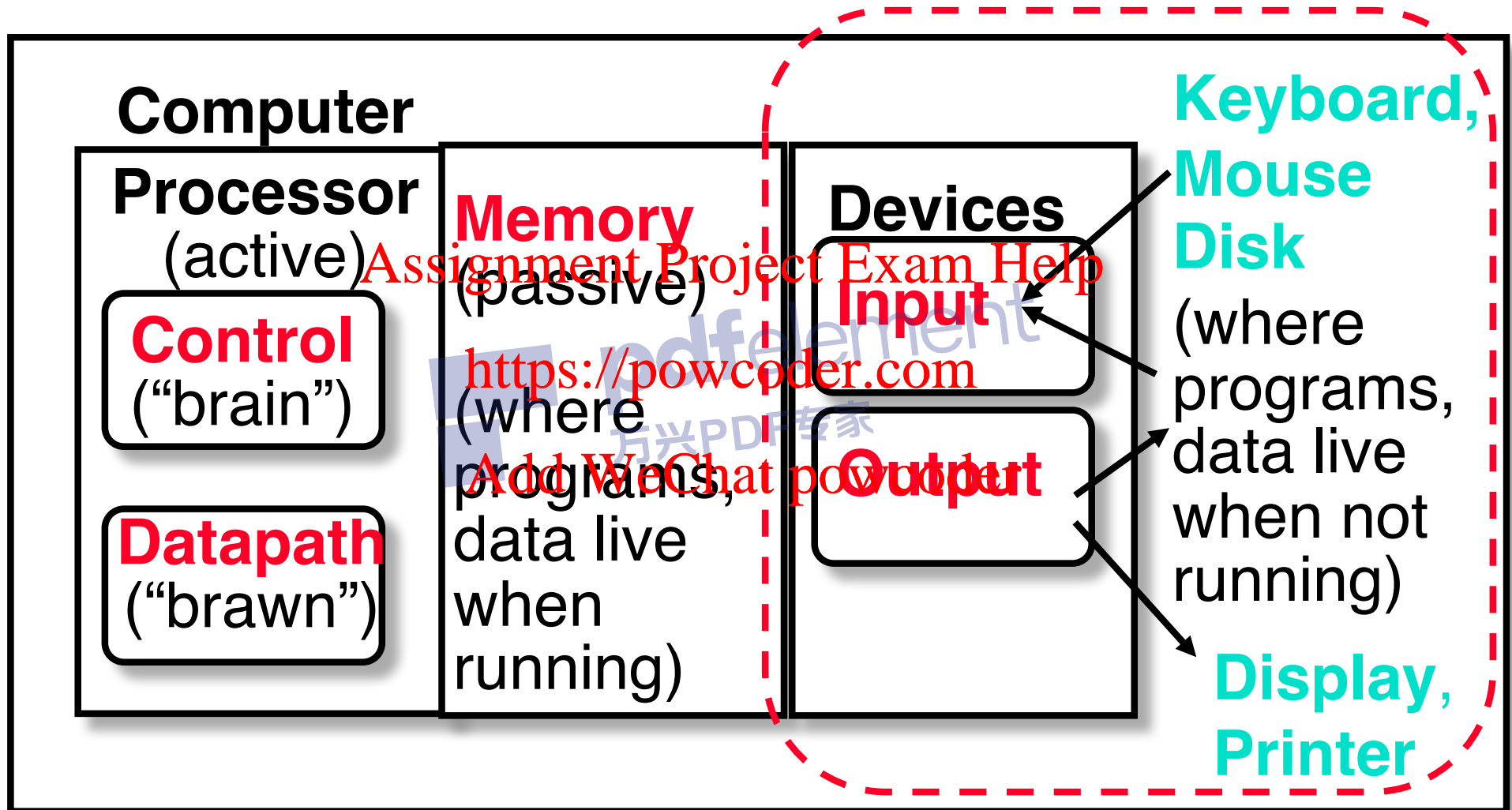
- **Interrupts**

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Anatomy: 5 components of any Computer



# Motivation for Input/Output

---

- I/O is how humans interact with computers
- I/O gives computers long-term memory.

Assignment Project Exam Help

- I/O lets computers do amazing things:



- Read pressure of synthetic hand and control synthetic arm and hand of fireman
- Display complex medical data
- Computer without I/O like a car without wheels; great technology, but won't get you anywhere

<https://powcoder.com>  
Add WeChat powcoder

# I/O Device Examples and Speeds

° **I/O Speed: bytes transferred per second**

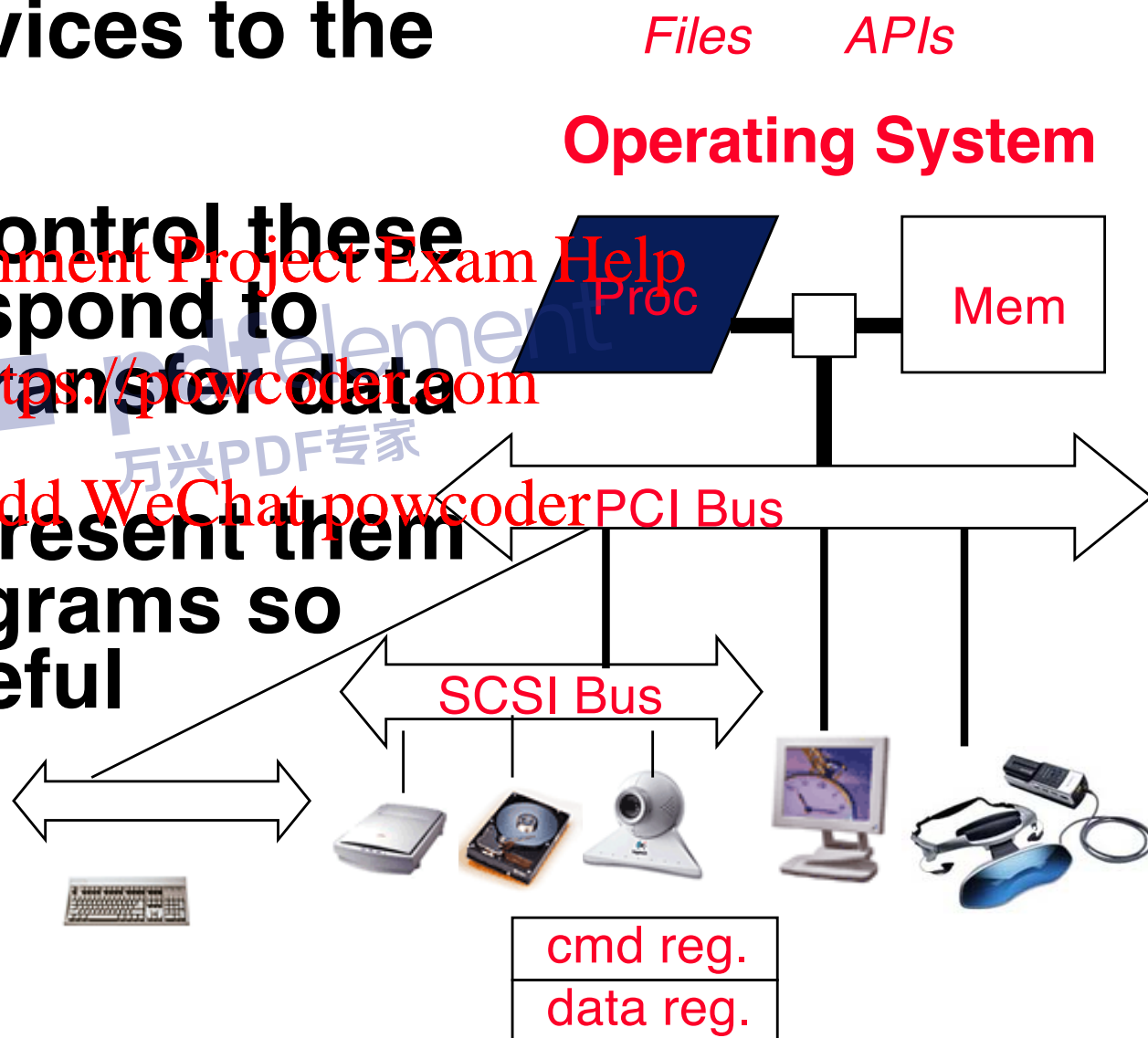
<b>Device</b>	<b>Behavior</b>	<b>Partner</b>	<b>Data Rate (KBytes/s)</b>
<b>Keyboard</b>	<b>Input</b>	<b>Human</b>	<b>0.01</b>
<b>Mouse</b>	<b>Input</b>	<b>Human</b>	<b>0.02</b>
<b>Voice output</b>	<b>Output</b>	<b>Human</b>	<b>5.00</b>
<b>Floppy disk</b>	<b>Storage</b>	<b>Machine</b>	<b>50.00</b>
<b>Laser Printer</b>	<b>Output</b>	<b>Human</b>	<b>100.00</b>
<b>Magnetic Disk</b>	<b>Storage</b>	<b>Machine</b>	<b>10,000.00</b>
<b>Network-LAN</b>	<b>I or O</b>	<b>Machine</b>	<b>10,000.00</b>
<b>Graphics Display</b>	<b>Output</b>	<b>Human</b>	<b>30,000.00</b>

# What do we need to make I/O work?

- A way to connect many types of devices to the Proc-Mem

- A way to control these devices, respond to them, and transfer data

- A way to present them to user programs so they are useful



# Current Bus Speeds

---

- ° **PCI (Peripheral Component Interconnect) bus:** PCI bus was created by Intel in 1993. PCI bus can transfer 32 or 64 bits at one time. PCI bus ran originally at 33 Mhz, with a data transfer of 250 Mbyte/s. PCI Express is used with modern graphics processor cards at 1 GByte/s (or more), also network cards.
- ° **SCSI (Small Computer System Interface) bus:** It is a high performance 16-bit bus which was used for fast disks, scanners, and for devices which require high bandwidth. It has a data rate of 640 MByte/s.  
<https://powcoder.com>
- ° **USB (Universal Serial Bus), a single bit bus:** It is used for connecting keyboard and mouse and printer and other USB devices such as wireless network adapters to the computer. A USB bus has a connector with four wires. Two wires are used for supplying electrical power to the USB devices. USB 1.0 had a data rate of 1½ MByte/s and USB 2.0 has a data rate of 60 MByte/s. There is now a USB 3.0, that can travel at 640 MByte/s.
- ° **IEEE 1394 or FireWire:** IEEE 1394 is used for high speed data transfer. It was built by Apple, **though Apple have moved away from it.** It can transfer data at a rate of up to 400 MByte/s. It is a single bit serial bus which is used for connecting cameras, and other multimedia devices.

# Instruction Set Architecture for I/O

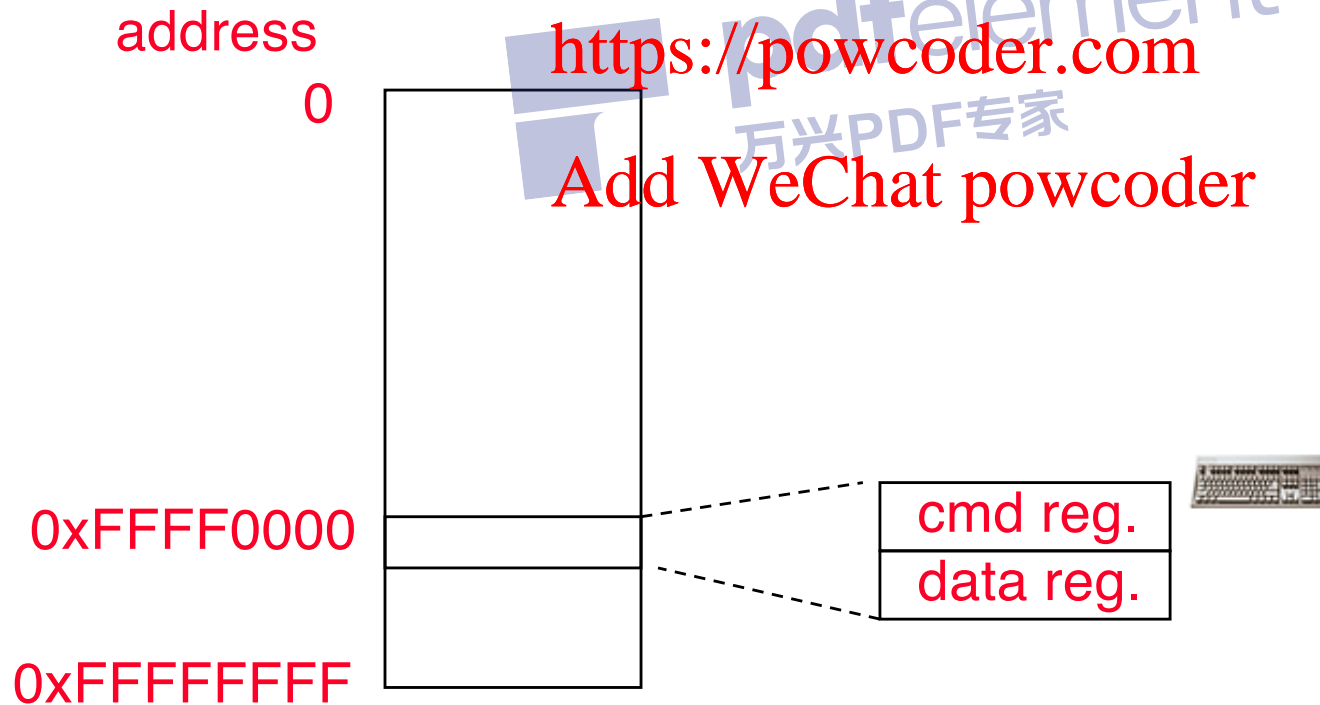
---

- What must the processor do for I/O?
  - Input: reads a sequence of bytes
  - Output: writes a sequence of bytes
- Some processors have special input and output instructions
- Alternative model (used by MIPS):
  - Use loads for input, stores for output
  - Called “Memory Mapped Input/Output”
  - A portion of the address space dedicated to communication paths to Input or Output devices (no memory there)



# Memory Mapped I/O (Polling)

- ° Certain addresses are not regular memory
- ° Instead, they correspond to registers in I/O devices



# Processor-I/O Speed Mismatch

---

- **1GHz microprocessor can execute 1 billion load or store instructions per second, or 4,000,000 KB/s data rate**
  - **I/O devices data rates range from 0.01 KB/s to 30,000 KB/s**
- **Input: device may not be ready to send data as fast as the processor loads it**
  - **Also, might be waiting for human to act**
- **Output: device may not be ready to accept data as fast as processor stores it**
- **What to do?**

# Processor Checks Status before Acting

- Path to device generally has 2 registers:
  - Control Register, says it's OK to read/write (I/O ready)
  - Data Register, contains data
- Processor reads from Control Register in loop, waiting for device to set *Ready* bit in Control reg (0  $\Rightarrow$  1) to say its OK
- Processor then loads from (input) or writes to (output) data register
  - Load from or Store into Data Register resets Ready bit (1  $\Rightarrow$  0) of Control Register

# MARS I/O Simulation

- MARS simulates 1 I/O device: memory-mapped terminal (keyboard + display)
  - Read from keyboard (**receiver**); 2 device regs
  - Writes to terminal (**transmitter**); 2 device regs

Assignment Project Exam Help

Receiver Control

0xffffffff0000

Receiver Data

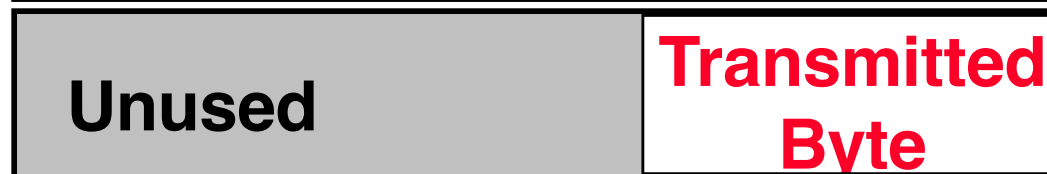
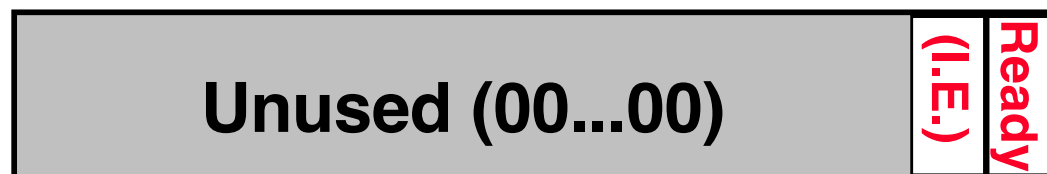
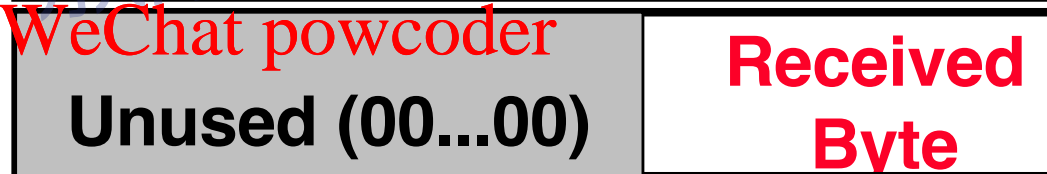
0xffffffff0004

Transmitter Control

0xffffffff0008

Transmitter Data

0xffffffff000c



# SPIM I/O

- **Control register rightmost bit (0): Ready**
  - **Receiver: Ready==1 means character in Data Register not yet been read;**  
**1  $\Rightarrow$  0 when data is read from Data Reg**
  - **Transmitter: Ready==1 means transmitter is ready to accept a new character;**  
**0  $\Rightarrow$  Transmitter still busy writing last char**
    - I.E. bit discussed later
- **Data register rightmost byte has data**
  - **Receiver: last char from keyboard; rest = 0**
  - **Transmitter: when write rightmost byte, writes char to display**

## I/O Example

- Input: Read from keyboard into \$v0

```

                                lui    $t0, 0xffff #ffff0000
Waitloop:                     lw     $t1, 0($t0) #control
                                andi  $t1, $t1, 0x0001
                                beq   $t1, $zero, Waitloop
                                lw     $v0, 4($t0) #data

```

- Output: Write to display from \$a0

```

                                lui    $t0, 0xffff #ffff0000
Waitloop:                     lw     $t1, 8($t0) #control
                                andi  $t1, $t1, 0x0001
                                beq   $t1, $zero, Waitloop
                                sw   $a0, 12($t0) #data

```

- Processor waiting for I/O called “Polling”

# Cost of Polling?

---

- The following example used dated estimates, but it conveys the general idea. Assume for a processor with a 1GHz clock it takes 400 clock cycles for a polling operation (call polling routine, accessing the device, and returning). Determine % of processor time used for polling
  - Mouse: polled 30 times/sec so as not to miss user movement  
<https://powcoder.com>
  - Floppy disk: transfers data in 2-Byte units and has a data rate of 50 KB/second. No data transfer can be missed.
  - Hard disk: transfers data in 16-Byte chunks and can transfer at 16 MB/second. Again, no transfer can be missed.

# % Processor time to poll mouse, floppy

- Mouse Polling, Clocks/sec  
 $= 30 * 400 = 12000 \text{ clocks/sec}$
- % Processor for polling:  
 $12 * 10^3 / 1 * 10^9 = 0.0012\%$   
 $\Rightarrow$  Polling mouse little impact on processor
- Frequency of Polling Floppy  
 $= 50 \text{ KB/s} / 25 \text{K} = 2 \text{ polls/sec}$
- Floppy Polling, Clocks/sec  
 $= 25\text{K} * 400 = 10,000,000 \text{ clocks/sec}$
- % Processor for polling:  
 $10 * 10^6 / 1 * 10^9 = 1\%$   
 $\Rightarrow$  OK if not too many I/O devices



## % Processor time to hard disk

---

- Frequency of Polling Disk  
 $= 16 \text{ MB/s} / 16 \text{B} = 1 \text{M polls/sec}$
- Disk Polling, Clocks/sec  
 $= 1 \text{M} * 400 = 400,000,000 \text{ clocks/sec}$
- % Processor for polling:  
 $400 * 10^6 / 1 * 10^9 = 40\%$   
 $\Rightarrow$  Unacceptable

# What is the alternative to polling?

---

- Wasteful to have processor spend most of its time “spin-waiting” for I/O to be ready
- Would like an unplanned procedure call that would be invoked only when I/O device is ready
- Solution: use *exception mechanism* to help I/O. *Interrupt* program when I/O ready, return when done with data transfer

# I/O Interrupt

---

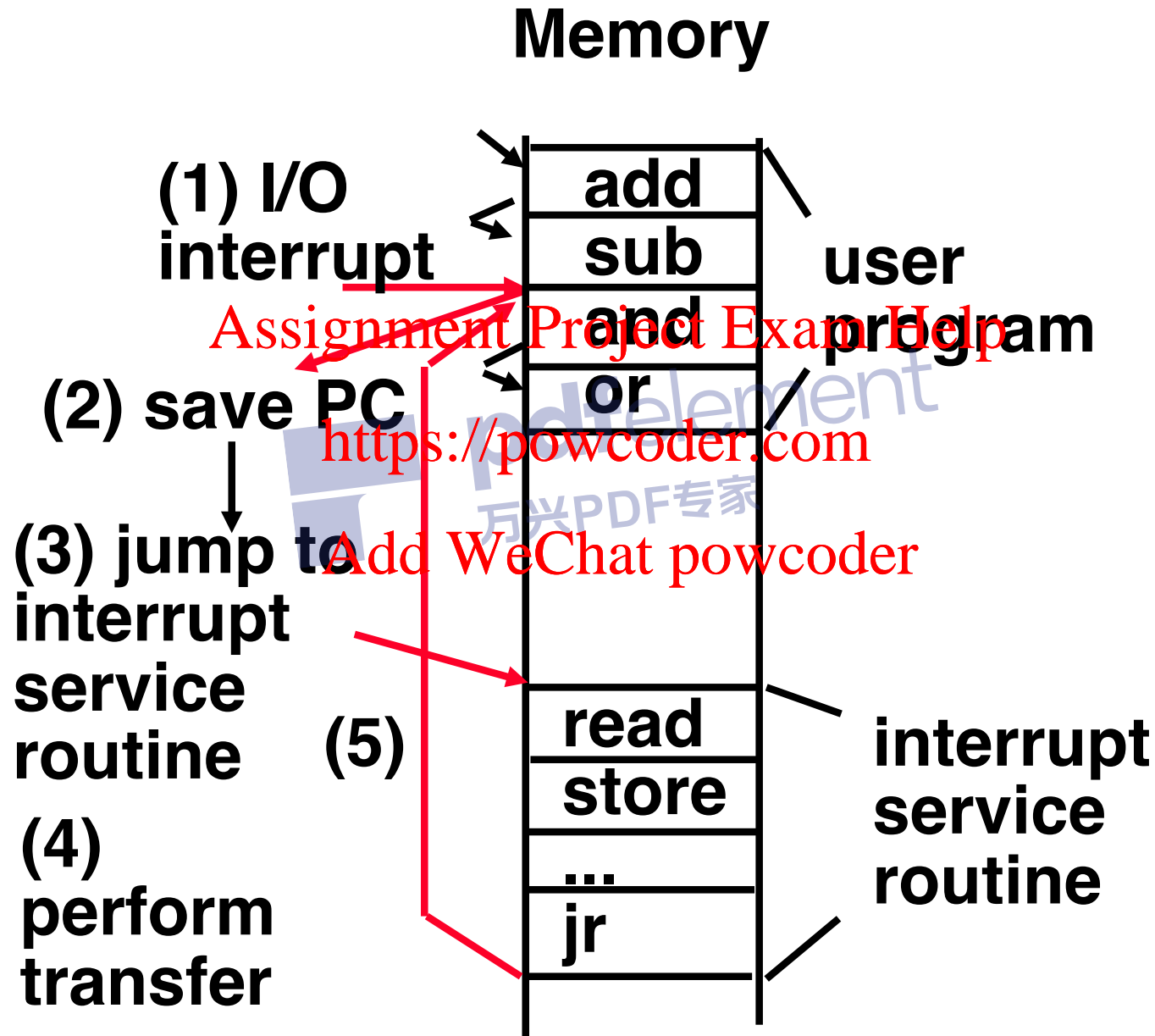
- **An I/O interrupt is like overflow exceptions except:**
  - An I/O interrupt is “asynchronous”
  - More information needs to be conveyed
- **An I/O interrupt is asynchronous with respect to instruction execution:**
  - I/O interrupt is not associated with any instruction, but it can happen in the middle of any given instruction
  - I/O interrupt does not prevent any instruction from completion

# Definitions for Clarification

---

- **Exception**: signal marking that something “out of the ordinary” has happened and needs to be handled
  - **Interrupt**: asynchronous exception
  - **Trap**: synchronous exception
- <https://powcoder.com>  
Add WeChat powcoder

# Interrupt Driven Data Transfer



# Instruction Set Support for I/O Interrupt

---

- **Save the PC for return**
  - But where?
- **Where to go when interrupt occurs?**
  - MIPS defines location: 0x80000080
- **Determine cause of interrupt?**
  - MIPS has Cause Register, 4-bit field (bits 5 to 2) gives cause of exception

# Instruction Set Support for I/O Interrupt

- Portion of MIPS architecture for interrupts called “coprocessor 0”
- Coprocessor 0 Instructions
  - Data transfer: lwcc0, swcc0
  - Move: mfc0, mtc0
- Coprocessor 0 Registers:

<i>name</i>	<i>number</i>	<i>usage</i>
Status	\$12	Interrupt enable
Cause	\$13	Exception type
EPC	\$14	Return address

(details in appendix A.7)

# SPIM I/O Simulation: Interrupt Driven I/O

- I.E. stands for Interrupt Enable
- Set Interrupt Enable bit to 1 have interrupt occur whenever Ready bit is set

Assignment Project Exam Help

Receiver Control

0xffff0000

Unused (00...00)

(I.E.)

Ready

Receiver Data

0xffff0004

Unused (00...00)

Received  
Byte

Transmitter Control

0xffff0008

Unused (00...00)

(I.E.)

Ready

Transmitter Data

0xffff000c

Unused

Transmitted  
Byte



# Benefit of Interrupt-Driven I/O

- Find the % of processor consumed if the hard disk is only active 5% of the time. Assuming 500 clock cycle overhead for each transfer, including interrupt:

Assignment Project Exam Help

- Disk Interrupts/sec =  $16 \text{ MB/s} / 16 \text{ B}$   
= 1M interrupts/sec

<https://powcoder.com>

- Disk Interrupts, Clocks/sec =  $1 \text{ M} * 500$   
= 500,000,000 clocks/sec

Add WeChat powcoder

- % Processor for during transfer:  
 $500 * 10^6 / 1 * 10^9 = 50\%$

- Disk active 5%  $\Rightarrow 5\% * 50\% \Rightarrow 2.5\%$   
busy

# Questions Raised about Interrupts

---

- **Which I/O device caused exception?**
  - Needs to convey the identity of the device generating the interrupt
- **Can avoid interrupts during the interrupt routine?**
  - What if more important interrupt occurs while servicing this interrupt?
  - Allow interrupt routine to be entered again?
- **Who keeps track of status of all the devices, handle errors, know where to put/supply the I/O data?**

## 4 Responsibilities leading to OS

---

- The I/O system is shared by multiple programs using the processor
- Low-level control of I/O device is complex because requires managing a set of concurrent events and because requirements for correct device control are often very detailed
- I/O systems often use interrupts to communicate information about I/O operations
- Would like I/O services for all user programs under safe control

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

## 4 Functions OS must provide

---

- **OS:** guarantees that user's program accesses only the portions of I/O device to which user has rights (e.g., file access)
- provides abstractions for accessing devices by supplying routines that handle low-level device operations  
<https://powcoder.com>
- handles the exceptions generated by I/O devices (and arithmetic exceptions generated by a program)  
Add WeChat powcoder
- tries to provide equitable access to the shared I/O resources, as well as schedule accesses to enhance system performance

# Things to Remember

---

- I/O gives computers their 5 senses
- I/O speed range is a few million to one
- Processor speed means must synchronize with I/O devices before use
- Polling works, but expensive
  - processor repeatedly queries devices
- Interrupts works, more complex
  - devices causes an exception, causing OS to run and deal with the device
- I/O control leads to Operating Systems