Topic 1

Computer System Overview

Topic Objectives

- review basic computer hardware components: processor, memory, I/O modules and system bus
- understand the roles of different CPU registers (pc, psw etc)
- understand how data flow between CPU, memory and various I/O devices via system buses
- understand *interrupt mechanism* and *interrupt handling*
- understand memory hierarchy - cost versus speed, decreasing frequency of access to lower speed memory, and cache memory
- understand the differences between programmed I/O, interrupt-driven I/O and DMA

Main Points

- How to balance the speed differences between components (such as CPU vs hard disk) so that:
  - fast and expensive components are not dragged down needlessly by slower and cheap components
  - make the overall systems fast at acceptable costs

Main Points - Cont.

- Use interrupts
  - to keep CPU waiting time down
  - to make the system more responsive
- Use cache technologies to achieve
  - large memory space
  - overall fast speed
  - reasonable cost
- Interrupts play key role in pre-emptive multi-tasking

Readings

- Must read:
  - Stalling Chapter 1
- Other readings:
  - Stalling Appendix 1A: on locality

Basic Elements

- Processor
- Main Memory
  - volatile
  - referred to as real memory or primary memory
- I/O modules
  - secondary memory devices
  - communications equipment
  - terminals
- System bus
  - communication among processors, memory, and I/O modules

Processor

- Two internal registers
  - Memory address register (MAR)
    - Specifies the address for the next read or write
  - Memory buffer register (MBR)
    - Contains data written into memory or receives data read from memory
  - I/O address register
  - I/O buffer register

Top-Level Components

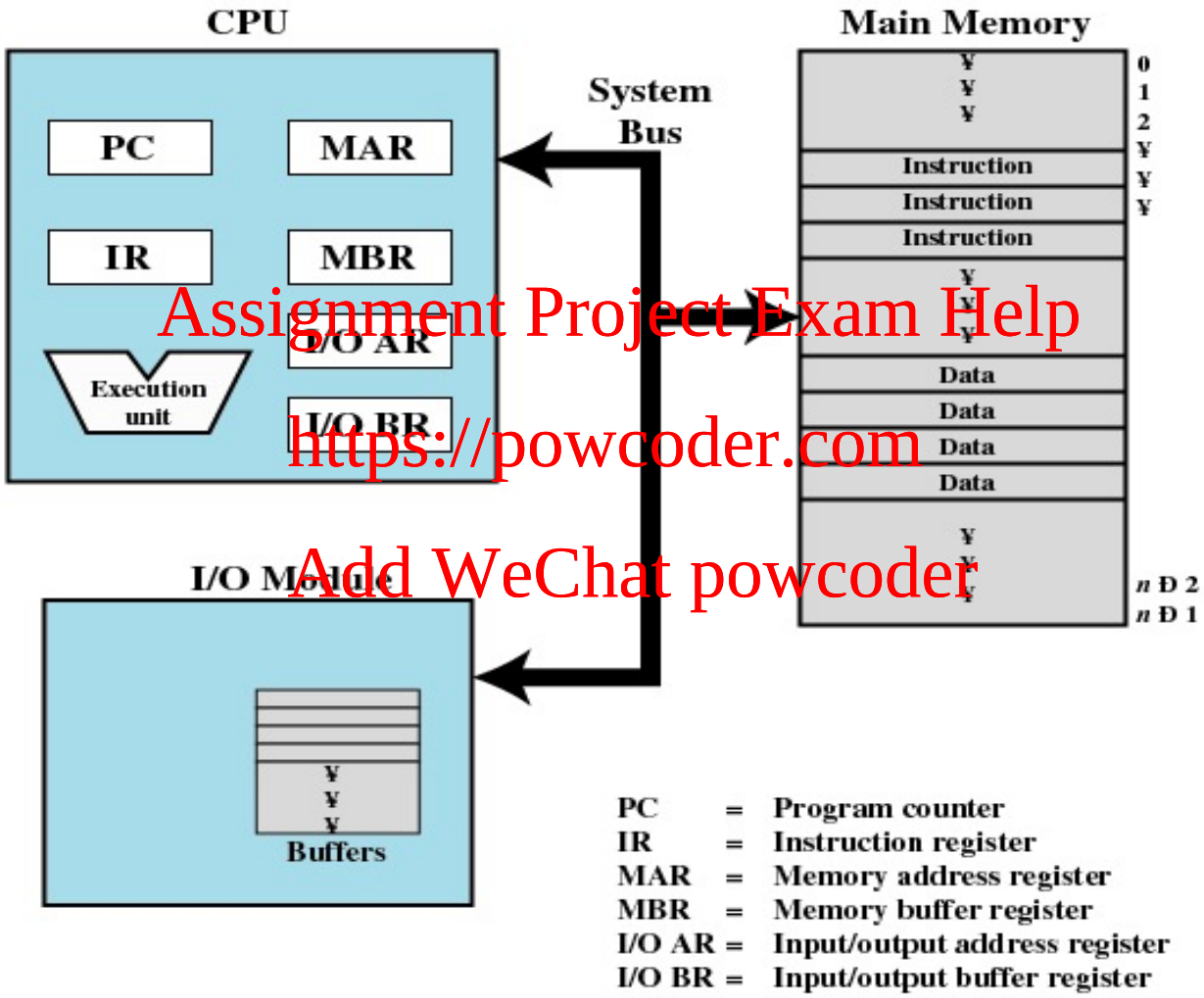


Figure 1.1 Computer Components: Top-Level View

Processor Registers

- User-visible registers
  - Enable programmer to minimize main-memory references by optimizing register use
- Control and status registers
  - Used by processor to control the operation of the processor
  - Used by privileged operating-system routines to control the execution of programs

User-Visible Registers

- May be referenced by machine language
- Available to all programs - application programs and system programs
- Types of registers
  - Data
  - Address
    - Index
    - Segment pointer
    - Stack pointer

User-Visible Registers

- Address Registers
  - Index
    - Involves adding an index to a base value to get an address
  - Segment pointer
    - When memory is divided into segments, memory is referenced by a segment and an offset
  - Stack pointer
    - Points to the top of stack

Control and Status Registers

- Program Counter (PC)
  - Contains the address of an instruction to be fetched
- Instruction Register (IR)
  - Contains the instruction most recently fetched
- Program Status Word (PSW)
  - Condition codes
  - Interrupt enable/disable
  - Supervisor/user mode

Control and Status Registers

- Condition Codes or Flags
  - Bits set by the processor hardware as a result of operations
  - Examples
    - Positive result
    - Negative result
    - Zero
    - Overflow

Instruction Execution

- Two steps
  - Processor reads instructions from memory
    - Fetches
  - Processor executes each instruction

Instruction Cycle

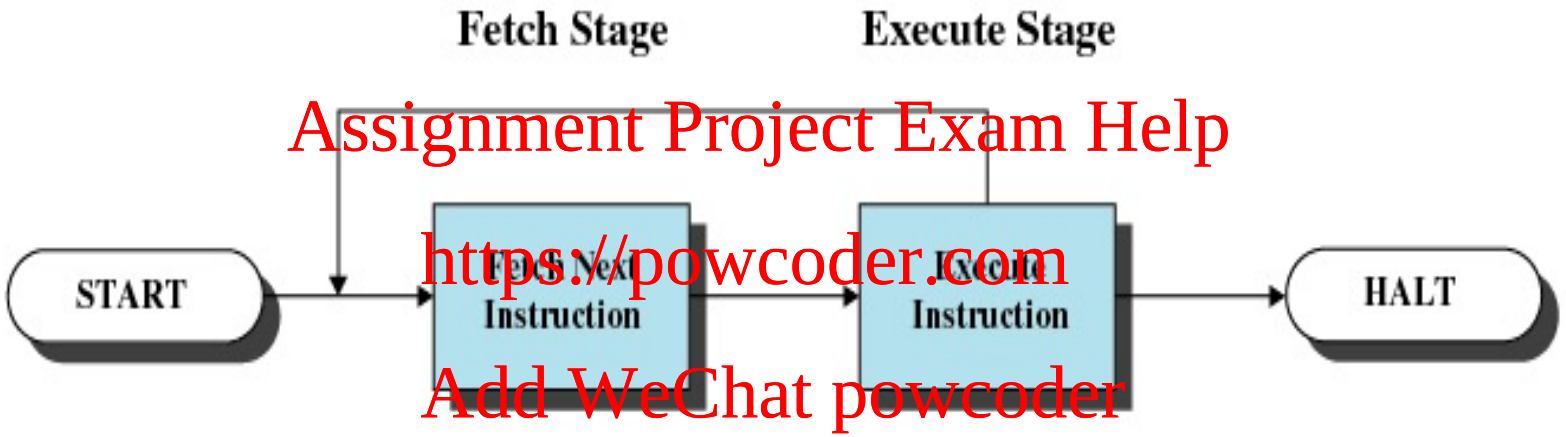


Figure 1.2 Basic Instruction Cycle

Instruction Fetch and Execute

- The processor fetches the instruction from memory
- Program counter (PC) holds address of the instruction to be fetched next
- Program counter is incremented after each fetch

Instruction Register

- Fetched instruction is placed in the instruction register
- Categories
  - Processor-memory
    - Transfer data between processor and memory
  - Processor-I/O
    - Transfer data to or from a peripheral device
  - Data processing
    - Arithmetic or logic operation on data
  - Control
    - Alter sequence of execution

Interrupts

- Interrupt the normal sequencing of the processor
- Most I/O devices are slower than the processor
  - Processor must pause to wait for device

Classes of Interrupts

Program:

generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, and reference 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

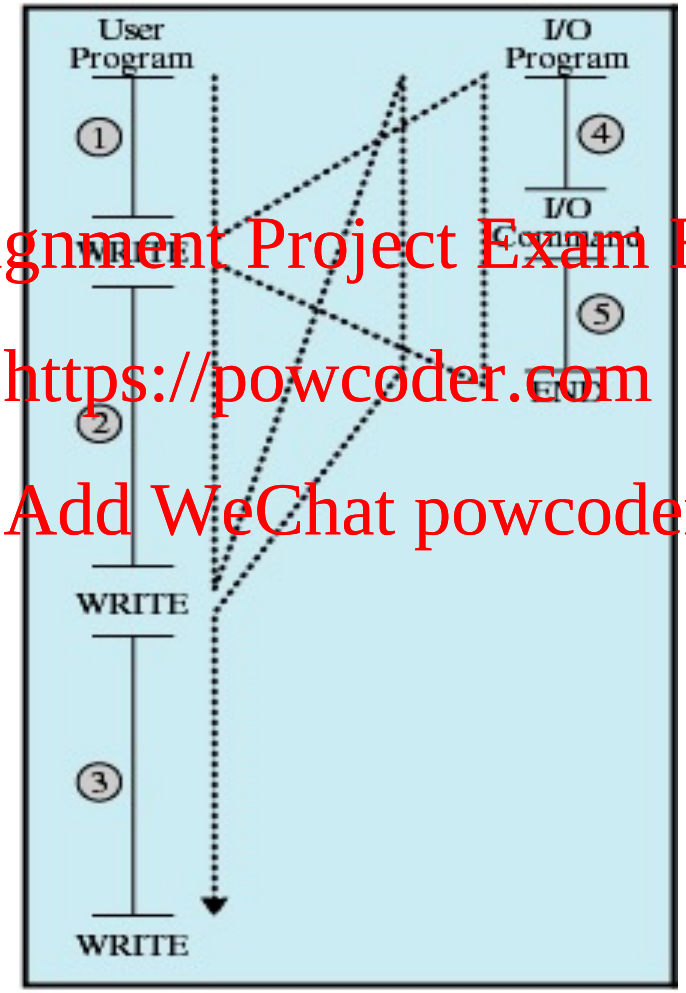
I/O

generated by an I/O controller, to signal normal completion of an operation or to signal a variety of error conditions

Hardware failure:

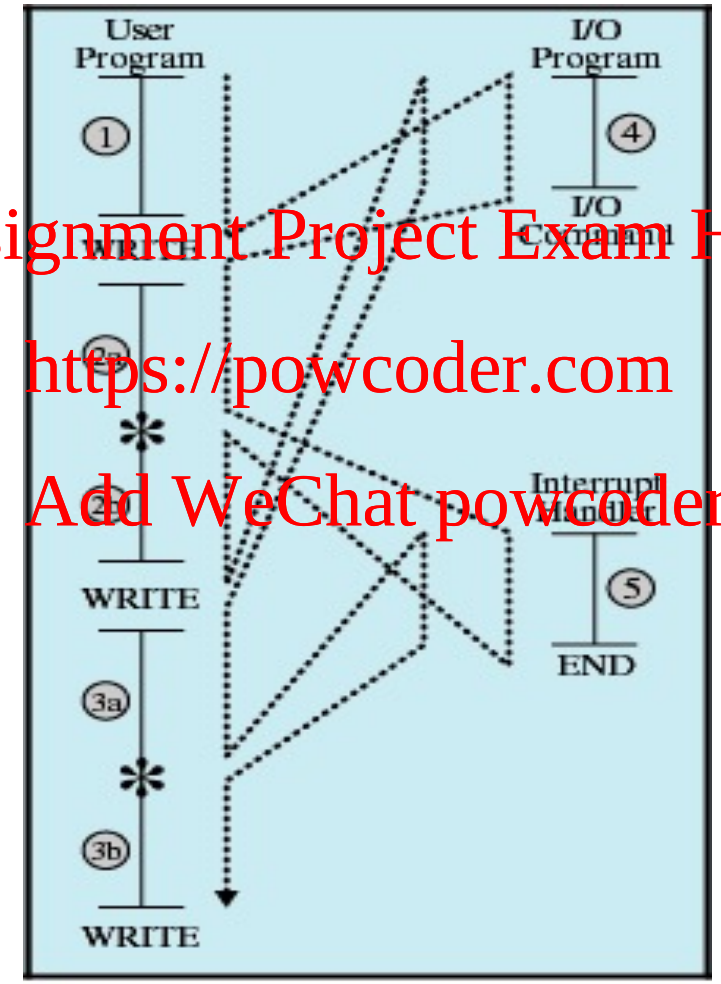
generated by a failure, such as power failure or memory parity error

Program Flow of Control Without Interrupts



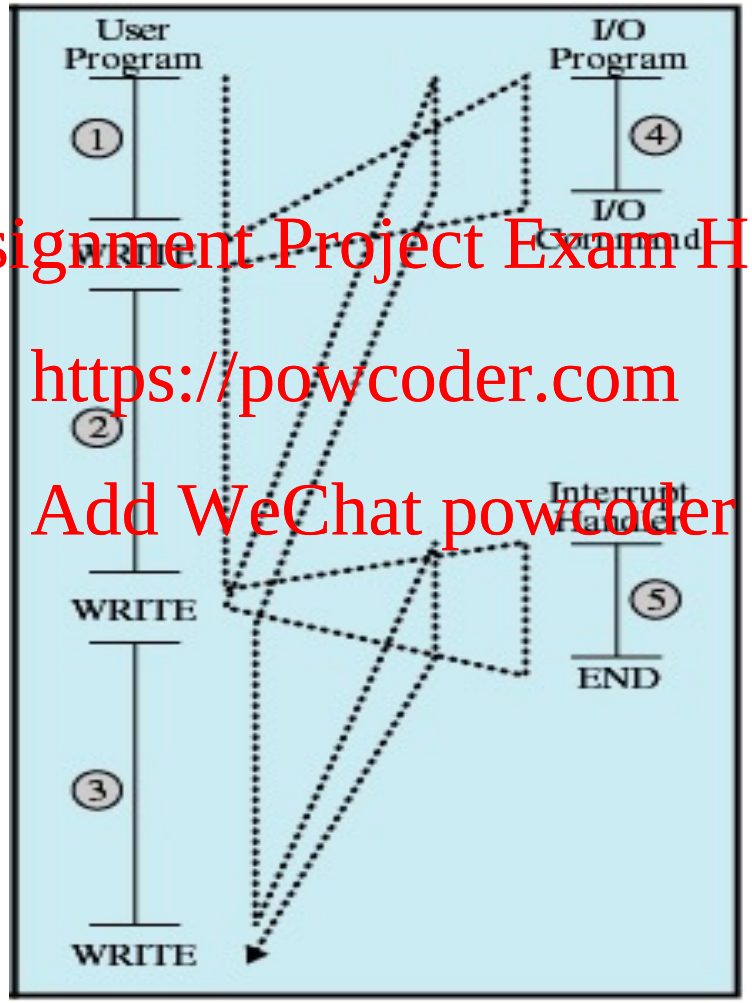
(a) No interrupts

Program Flow of Control With Interrupts, Short I/O Wait



(b) Interrupts; short I/O wait

Program Flow of Control With Interrupts; Long I/O Wait



(c) Interrupts; long I/O wait

Interrupt Handler

- Program to service a particular I/O device
- Generally part of the operating system

Interrupts

- Suspends the normal sequence of execution

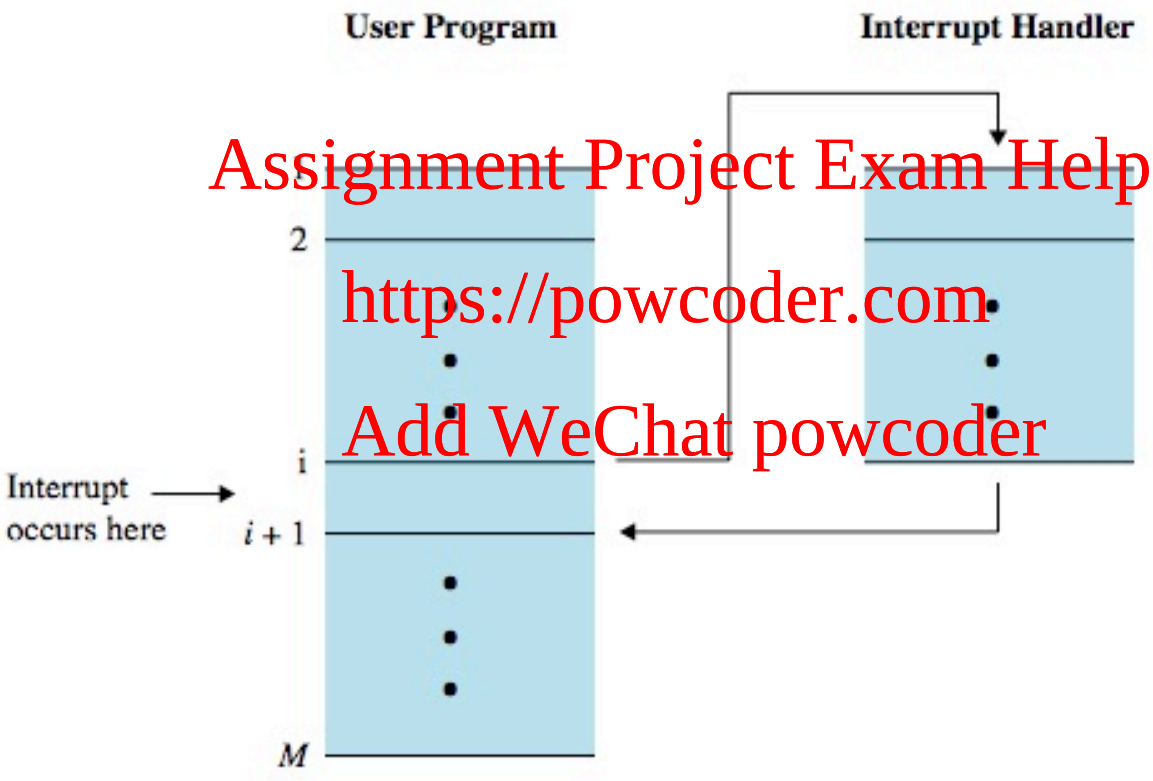


Figure 1.6 Transfer of Control via Interrupts

Interrupt Cycle

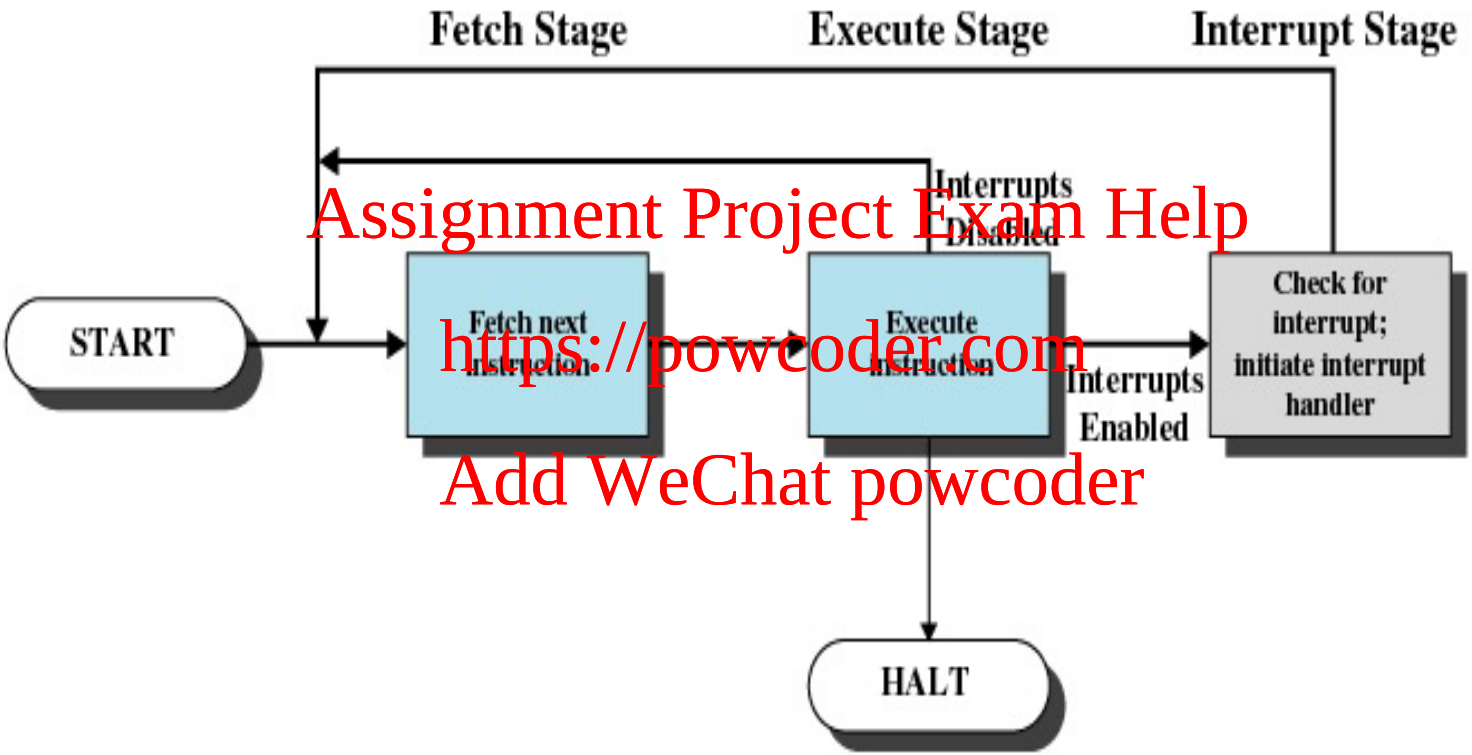


Figure 1.7 Instruction Cycle with Interrupts

Interrupt Cycle

- Processor checks for interrupts
- If no interrupts, fetch the next instruction for the current program
- If an interrupt is pending, suspend execution of the current program, and execute the interrupt-handler routine

Timing Diagram Based on Short I/O Wait

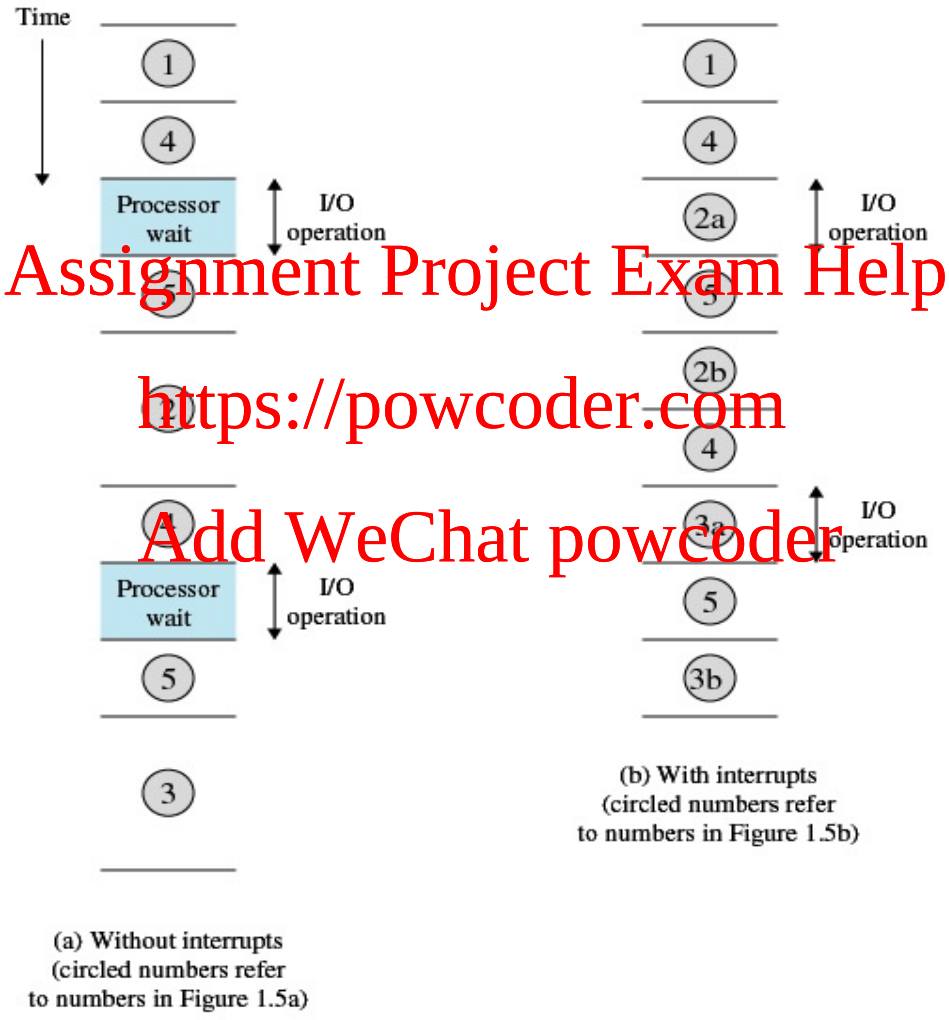


Figure 1.8 Program Timing: Short I/O Wait

Timing Diagram Based on Long I/O Wait

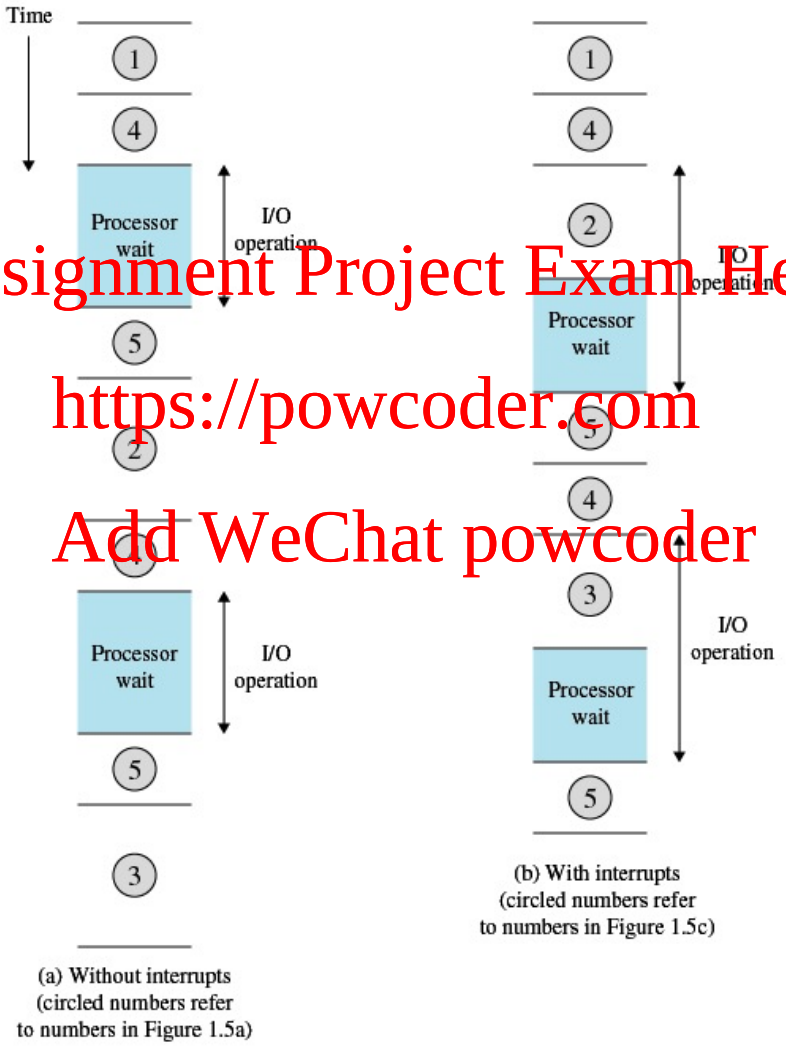


Figure 1.9 Program Timing: Long I/O Wait

Simple Interrupt Processing

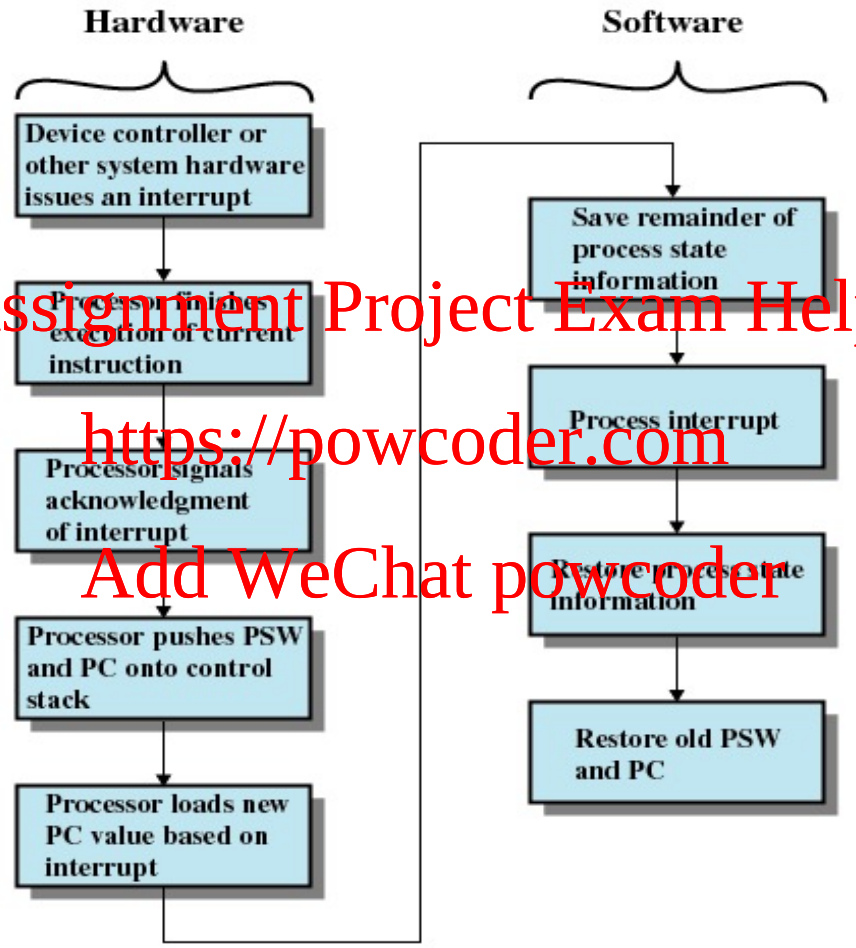


Figure 1.10 Simple Interrupt Processing

Changes in Memory and Registers for an Interrupt

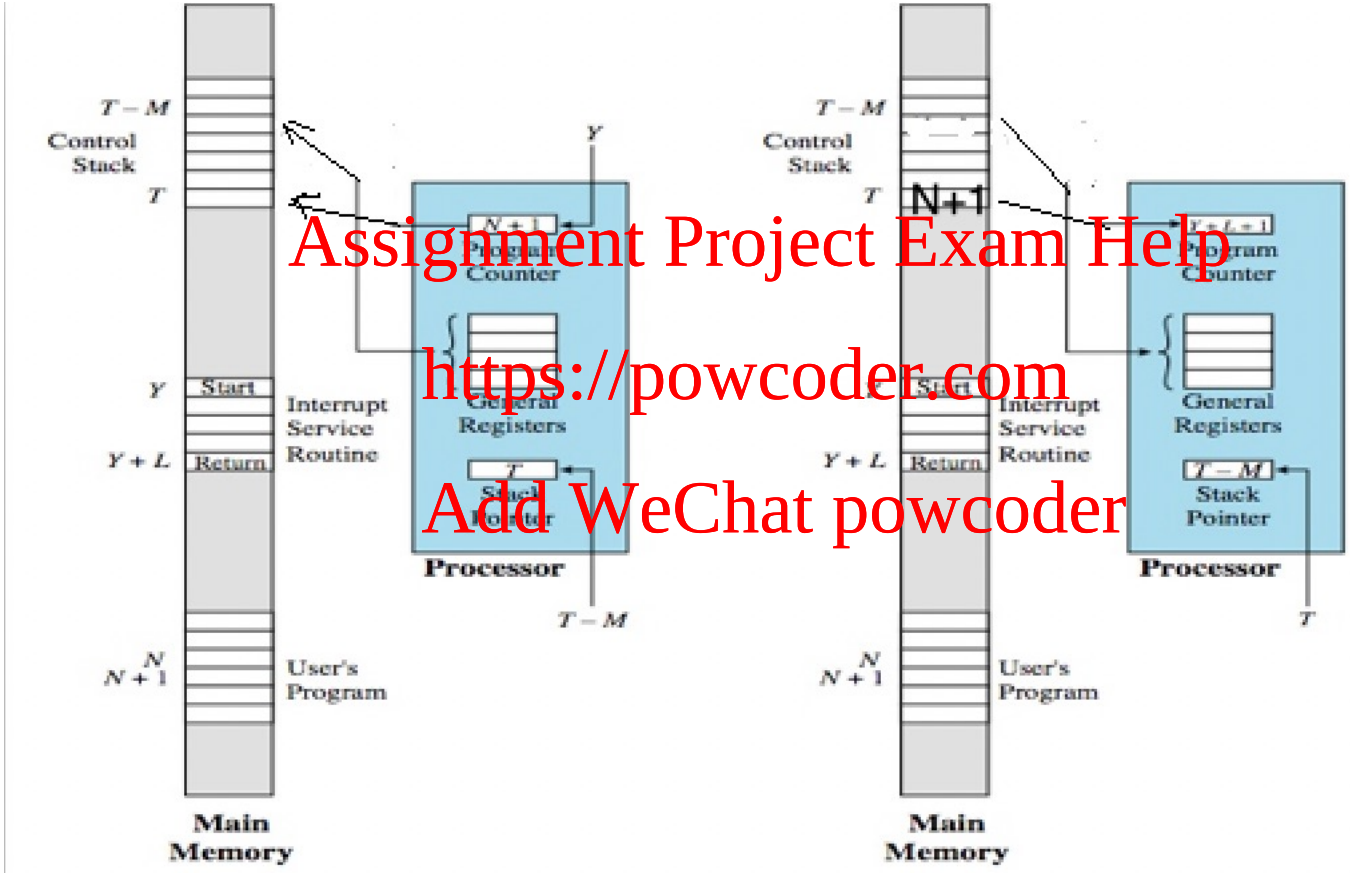
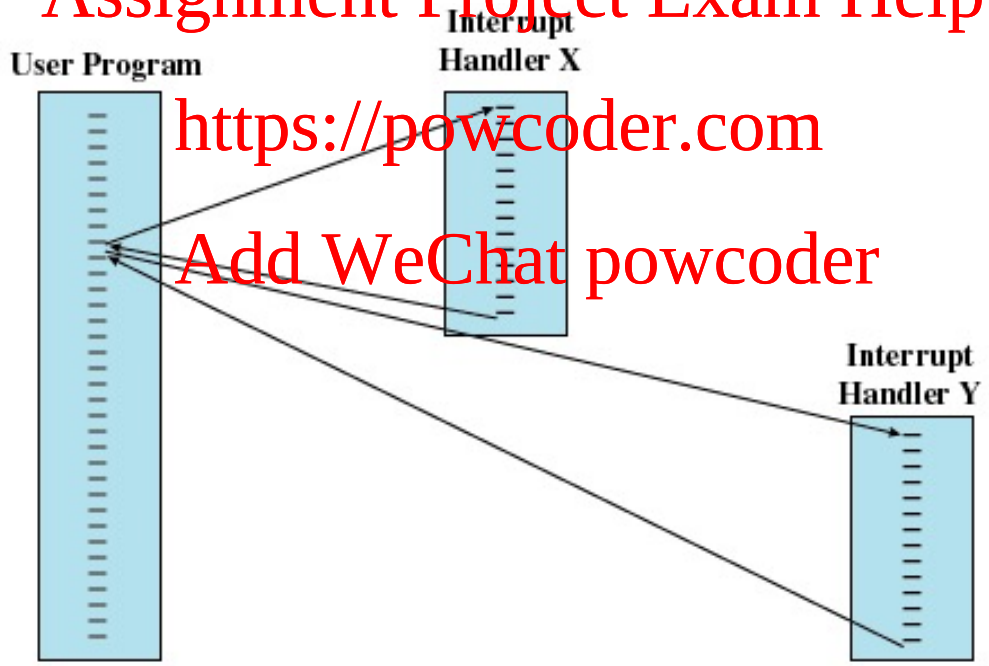


Figure 1.11 Changes in Memory and Registers for an Interrupt

Multiple Interrupts

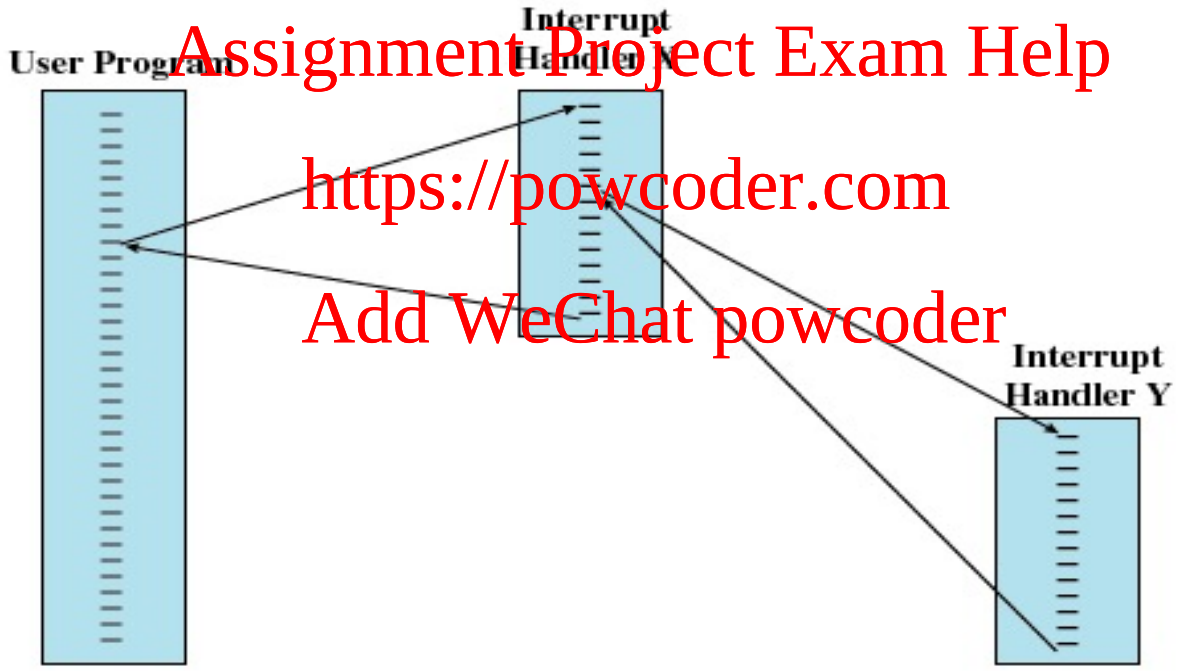
- Disable interrupts while an interrupt is being processed



(a) Sequential interrupt processing

Multiple Interrupts

- Define priorities for interrupts



(b) Nested interrupt processing

Multiple Interrupts

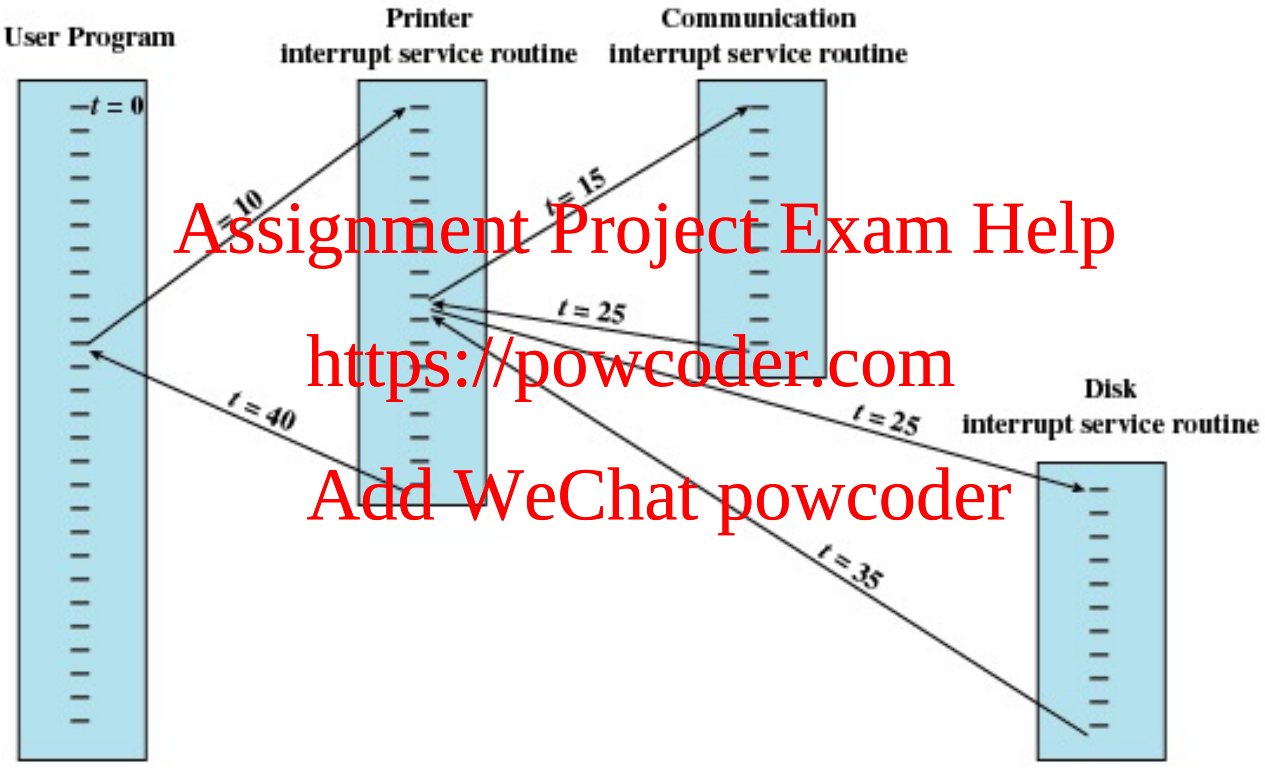


Figure 1.13 Example Time Sequence of Multiple Interrupts

Multiprogramming

- Processor has more than one program to execute
- The sequence the programs are executed depend on their relative priority and whether they are waiting for I/O
- After an interrupt handler completes, control may not return to the program that was executing at the time of the interrupt

Memory Hierarchy

- Faster access time, greater cost per bit
- Greater capacity, smaller cost per bit
- Greater capacity, slower access speed

Memory Hierarchy

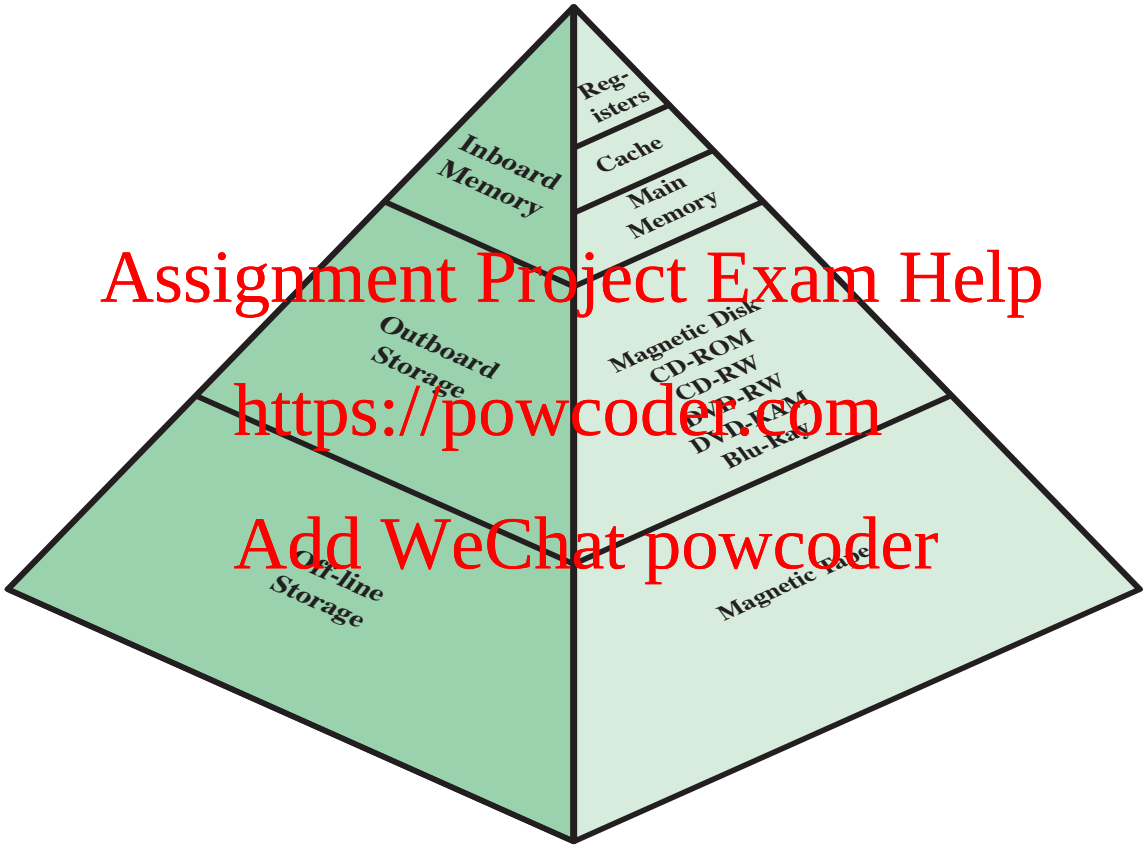


Figure 1.14 The Memory Hierarchy

Going Down the Hierarchy

- Decreasing cost per bit
- Increasing capacity
- Increasing access time
- Decreasing frequency of access of the memory by the processor
  - Locality of reference

Secondary Memory

- Nonvolatile
- Auxiliary memory
- Used to store program and data files

Disk Cache

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

Cache Memory

- Invisible to operating system
- Increase the speed of memory
- Processor speed is faster than memory speed
- Exploit the principle of locality

Cache Memory

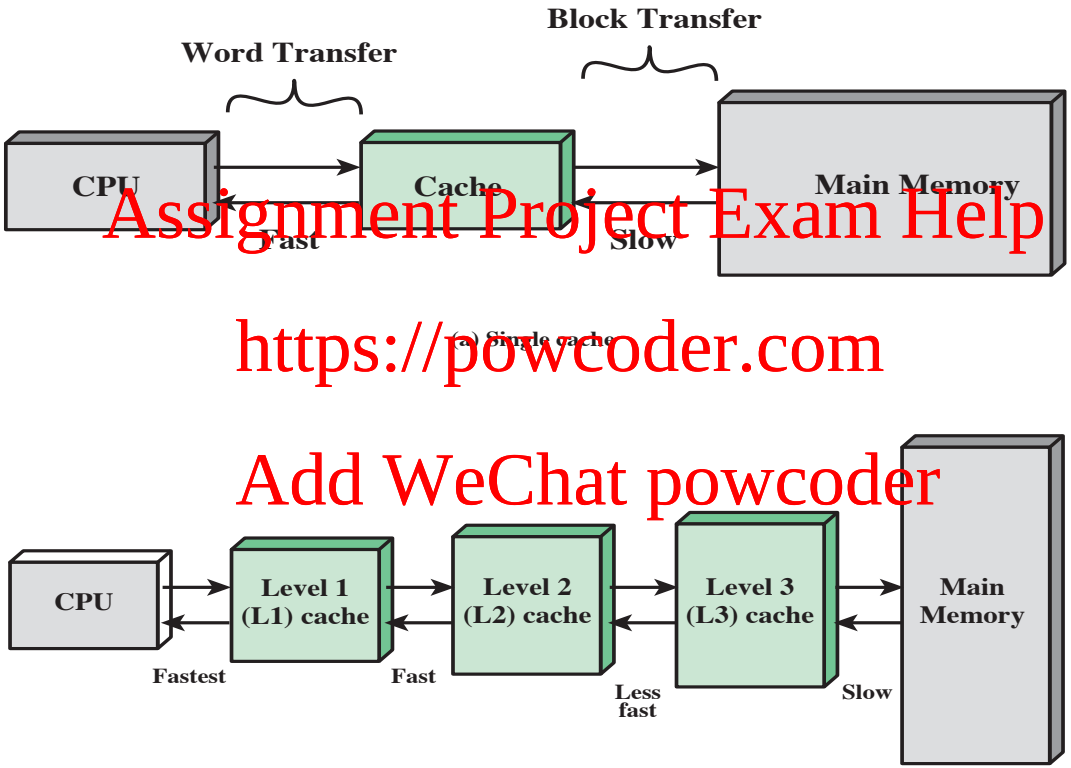


Figure 1.16 Cache and Main Memory

Cache Memory

- Contains a copy of a portion of main memory
- Processor first checks cache
- If not found in cache, the block of memory containing the needed information is moved to the cache and delivered to the processor

Cache/Main Memory System

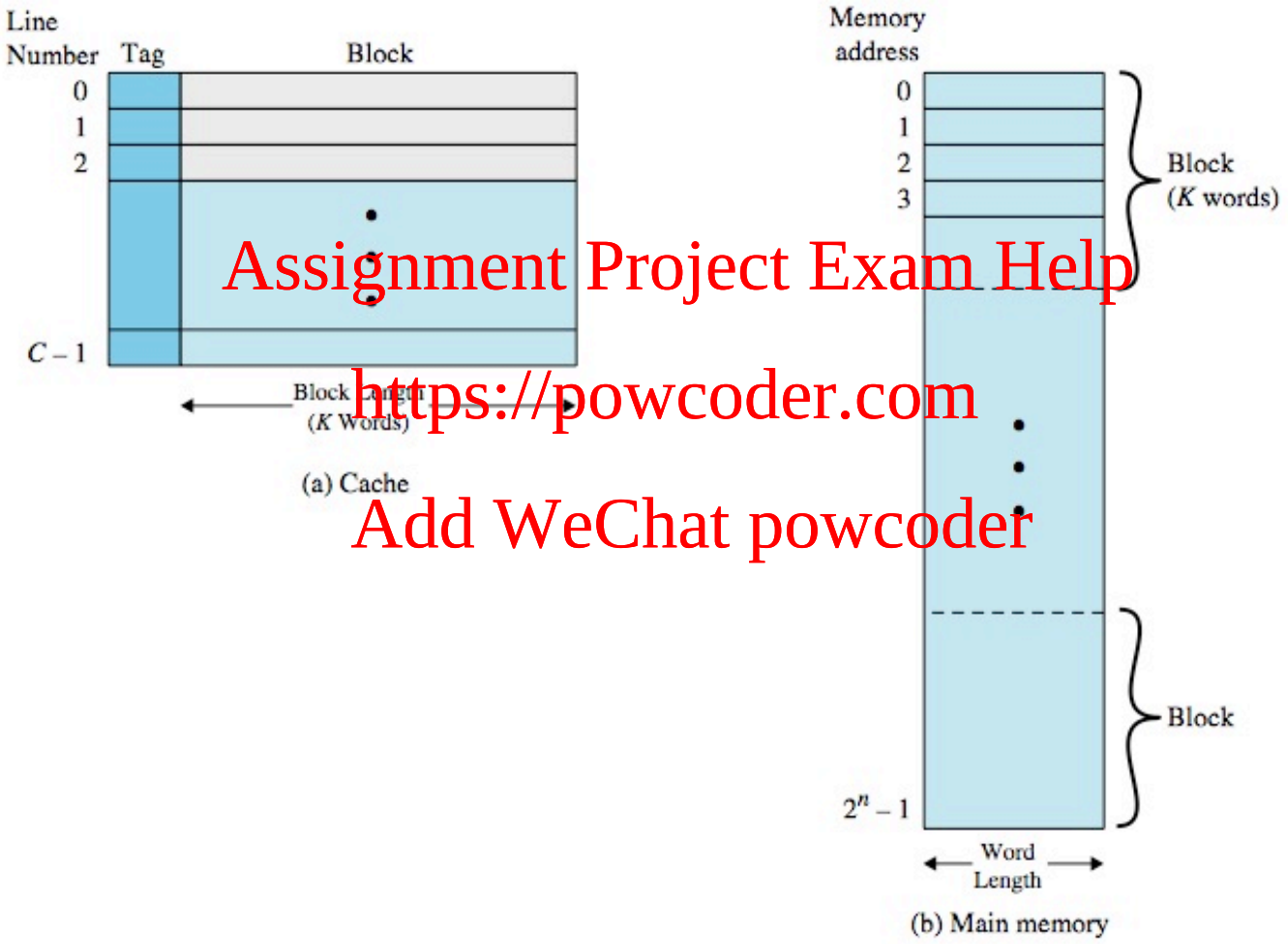


Figure 1.17 Cache/Main-Memory Structure

Cache Read Operation

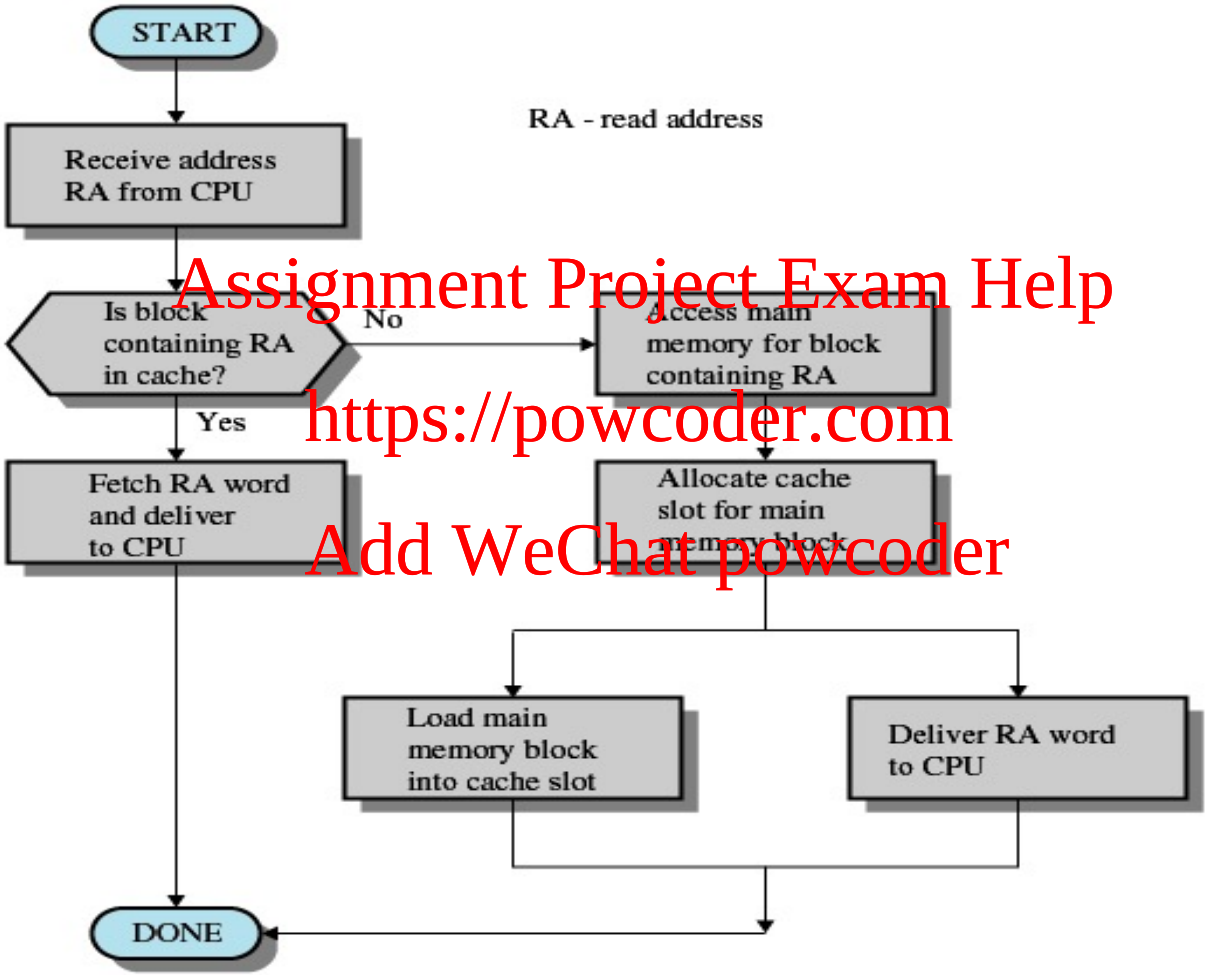
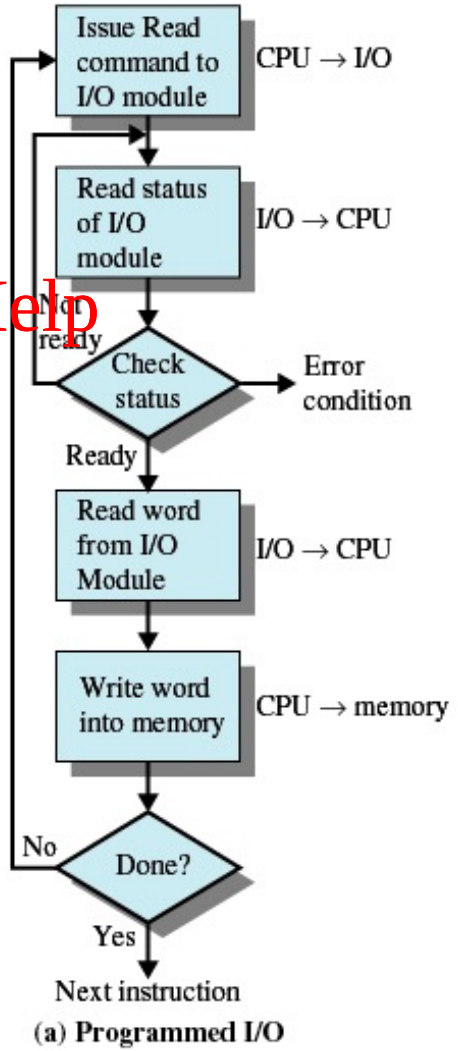


Figure 1.18 Cache Read Operation

Programmed I/O

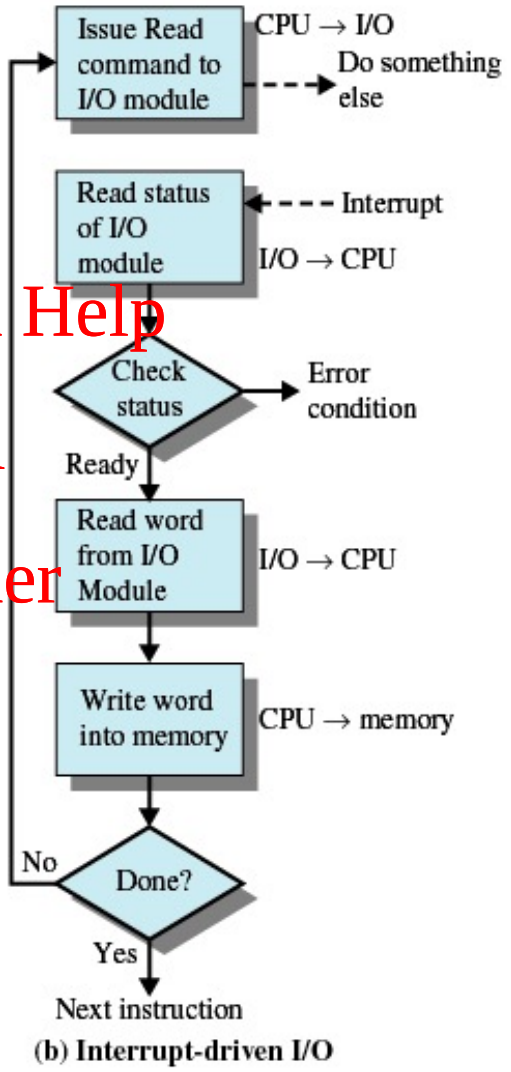
- I/O module performs the action, not the processor
- Sets appropriate bits in the I/O status register
- No interrupts occur
- Processor checks status until operation is complete



(a) Programmed I/O

Interrupt-Driven I/O

- Processor is interrupted when I/O module ready to exchange data
- Processor saves context of program executing and begins executing interrupt-handler
- No needless waiting
- Consumes a lot of processor time because every word read or written passes through the processor



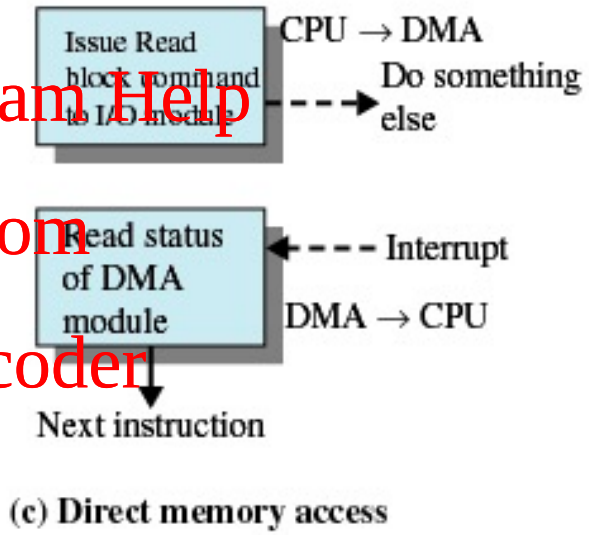
(b) Interrupt-driven I/O

Direct Memory Access (DMA)

- I/O exchanges occur directly with memory
- Processor grants I/O module authority to read from or write to memory
- Relieves the processor responsibility for the exchange

Direct Memory Access

- Transfers a block of data directly to or from memory
- An interrupt is sent when the transfer is complete
- Processor continues with other work



(c) Direct memory access