



**University of Engineering and Technology (UET),
Peshawar, Pakistan**

Lecture 2

CSE-304: Computer Organization and Architecture

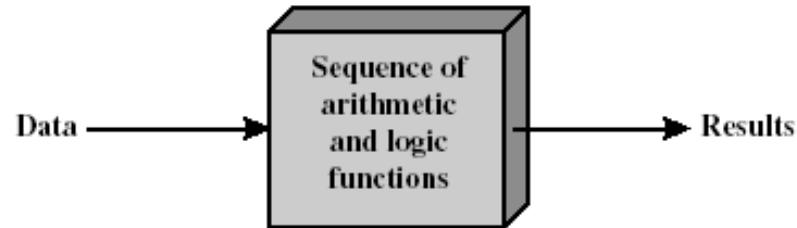
BY:

Dr. Muhammad Athar Javed Sethi

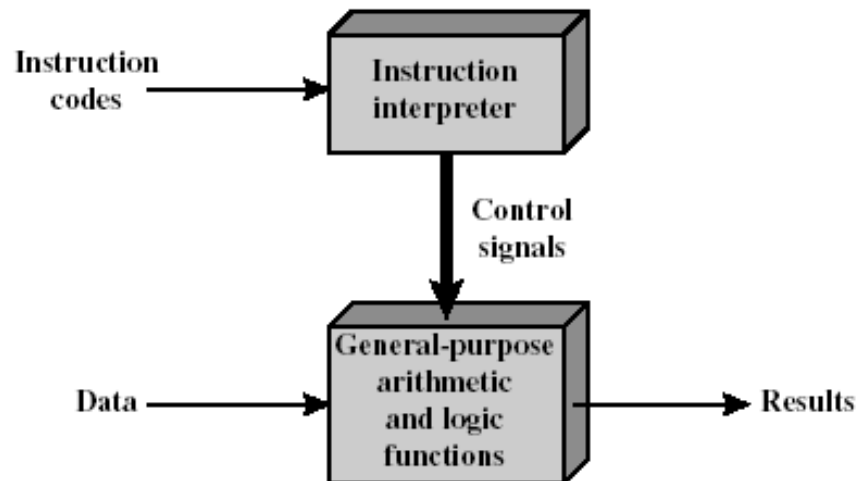
Program Concept

- Hardwired systems are inflexible
- General purpose hardware can do different tasks, given correct control signals
- Instead of re-wiring, supply a new set of control signals

Hardware vs. HW + SW



(a) Programming in hardware



(b) Programming in software

What is a program?

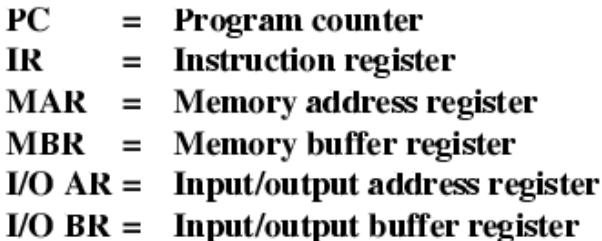
- A sequence of steps
- For each step, an arithmetic or logical operation is done
- For each operation, a different set of control signals is needed

Function of Control Unit

- For each operation a unique code (opcode) is provided
 - e.g. ADD, MOVE
- A hardware segment accepts the code and issues the control signals
- We have a computer!

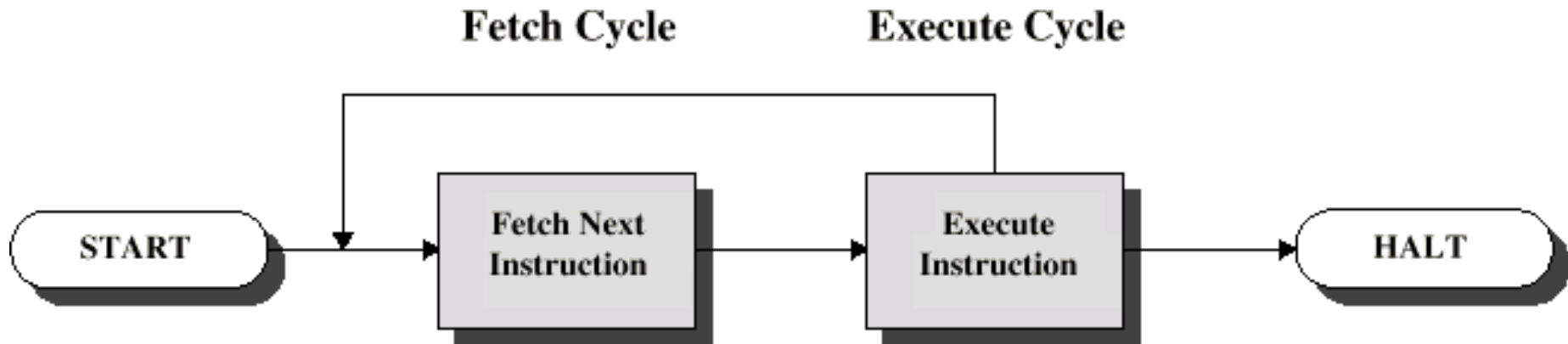
Components

- The Control Unit (CU) and the Arithmetic and Logic Unit (ALU) constitute the Central Processing Unit (CPU)
- Data and instructions need to get into the system and results need to get out
 - Input/output (I/O module)
- Temporary storage of code and results is needed
 - Main memory (RAM)



Instruction Cycle

- Two steps:
 - Fetch
 - Execute



Fetch Cycle

- Program Counter (PC) holds address of next instruction to fetch
- Processor fetches instruction from memory location pointed to by PC
- Increment PC
 - Unless told otherwise
- Instruction loaded into Instruction Register (IR)

Execute Cycle

- Processor interprets instruction and performs required actions, such as:
 - Processor - memory
 - data transfer between CPU and main memory
 - Processor - I/O
 - Data transfer between CPU and I/O module
 - Data processing
 - Some arithmetic or logical operation on data
 - Control
 - Alteration of sequence of operations
 - e.g. jump
 - Combination of above

Example of Program Execution

- Opcode (4 bit):**

0001: Load AC from memory

0010: Store AC to memory

0101: Add to AC from memory

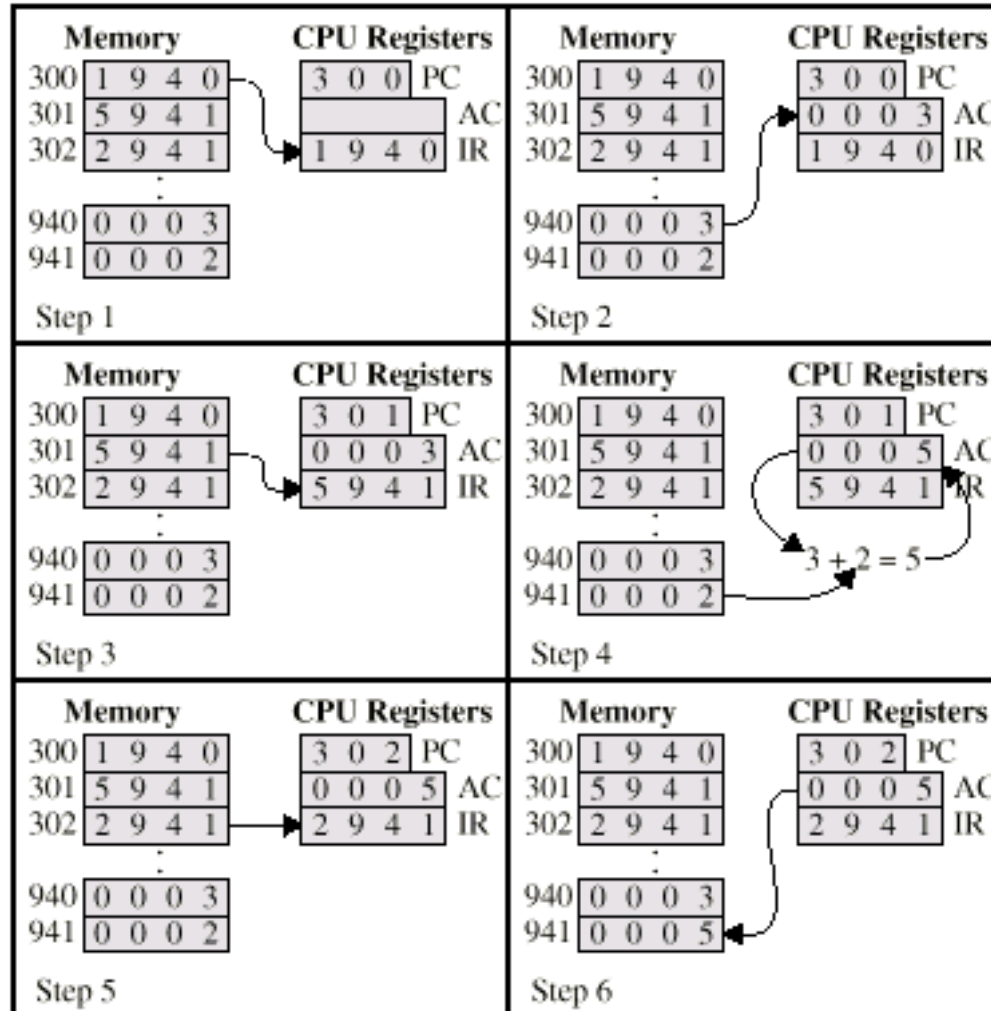
- Operand (12 bit):**

Address

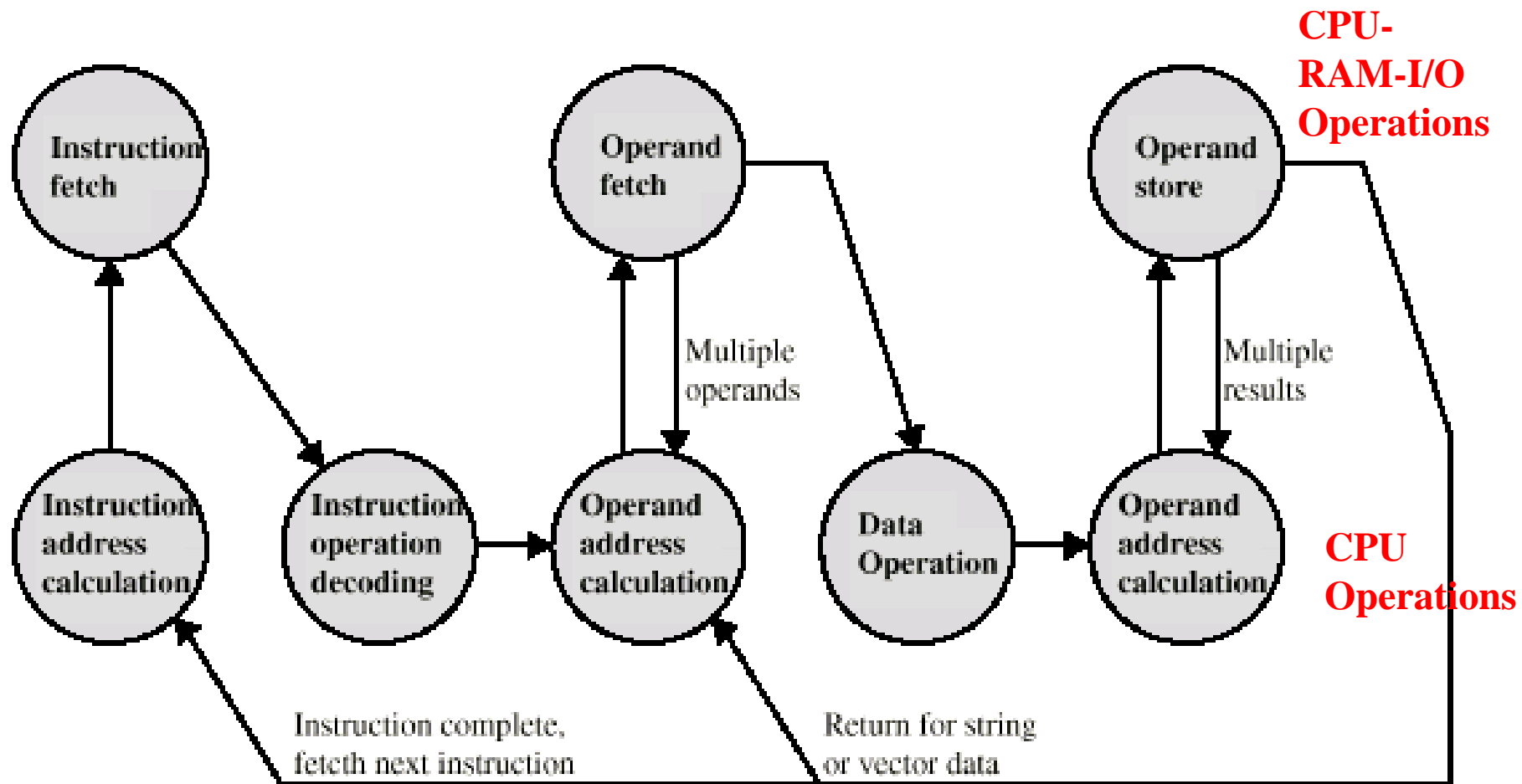
- Both instruction and data are 16 bits long.

Fetch

Execution



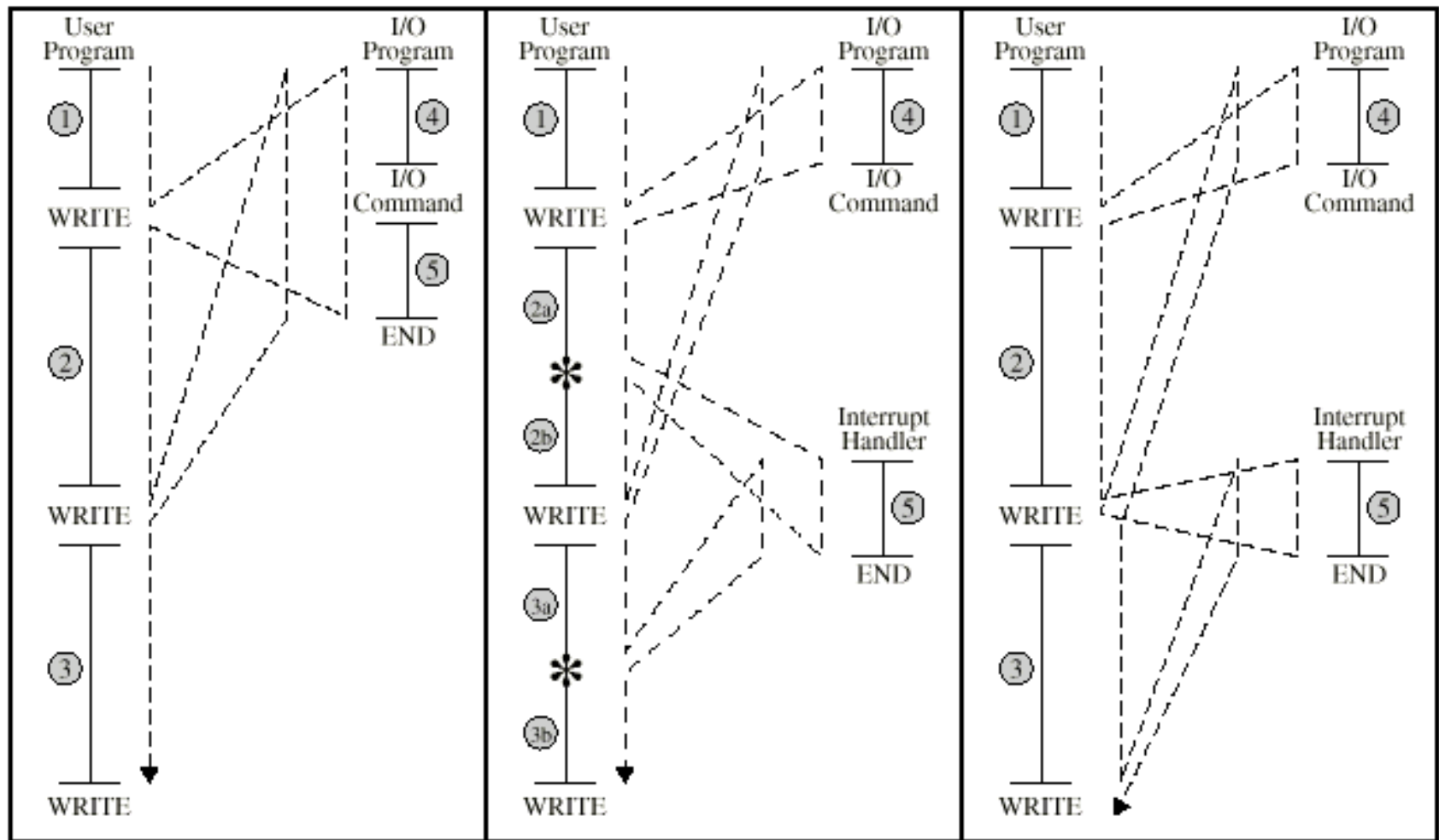
Instruction Cycle - State Diagram



Interrupts

- Mechanism by which other modules (e.g. I/O) may interrupt normal sequence of processing
- Program
 - e.g. overflow, division by zero
- Timer
 - Generated by internal processor timer
 - Used in pre-emptive multi-tasking
- I/O
 - from I/O controller
- Hardware failure
 - e.g. power failure, memory parity error

Program Flow Control



(a) No interrupts

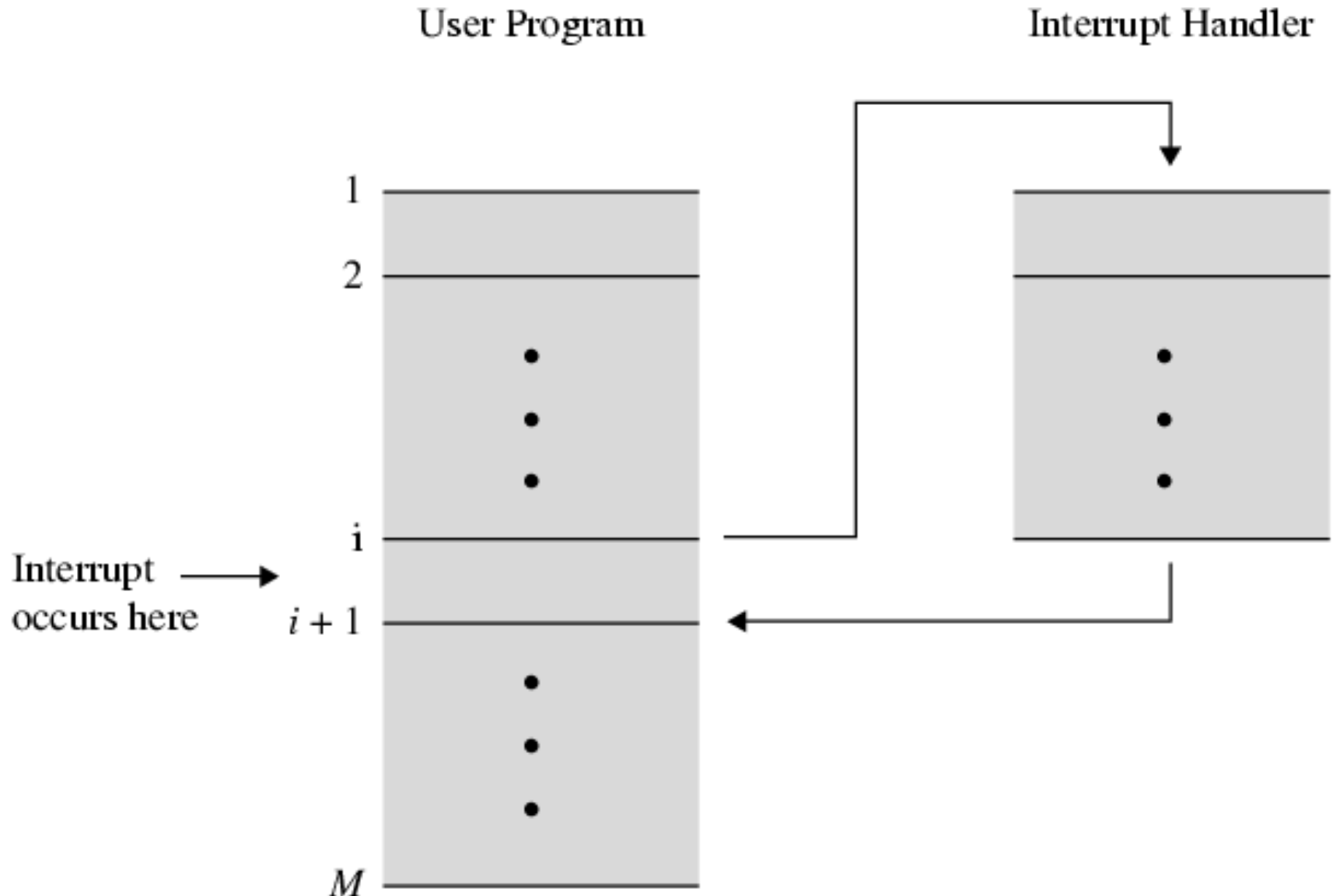
(b) Interrupts; short I/O wait

(c) Interrupts; long I/O wait

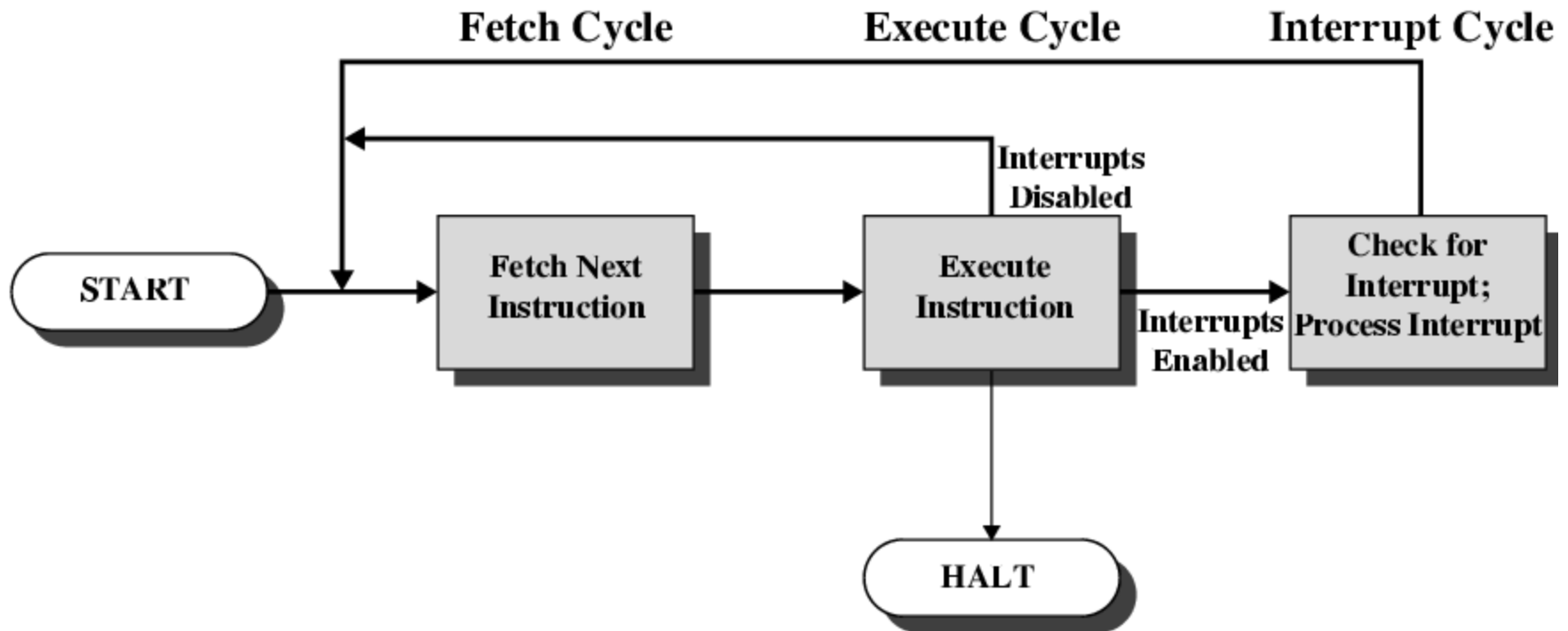
Interrupt Cycle

- Added to instruction cycle
- Processor checks for interrupt
 - Indicated by an interrupt signal
- If no interrupt, fetch next instruction
- If interrupt pending:
 - Suspend execution of current program
 - Save context
 - Set PC to start address of interrupt handler routine
 - Process interrupt
 - Restore context and continue interrupted program

Transfer of Control via Interrupts

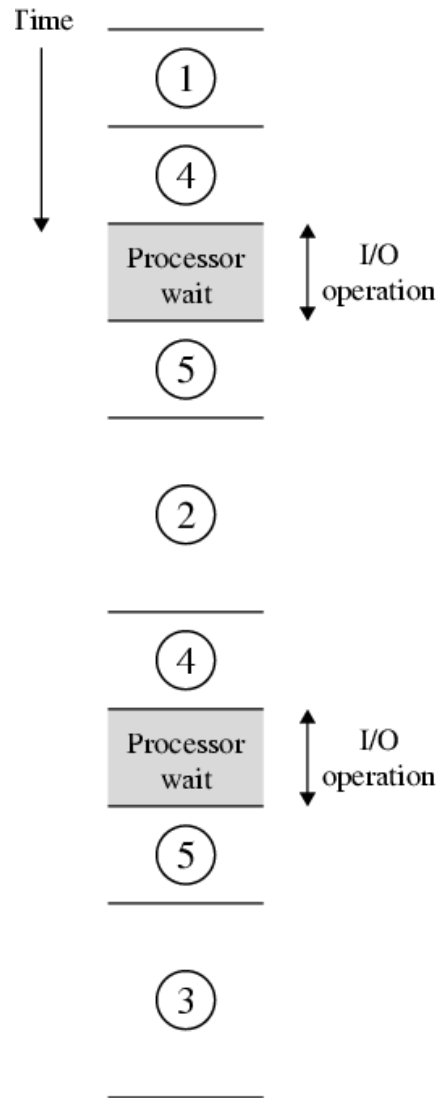


Instruction Cycle with Interrupts

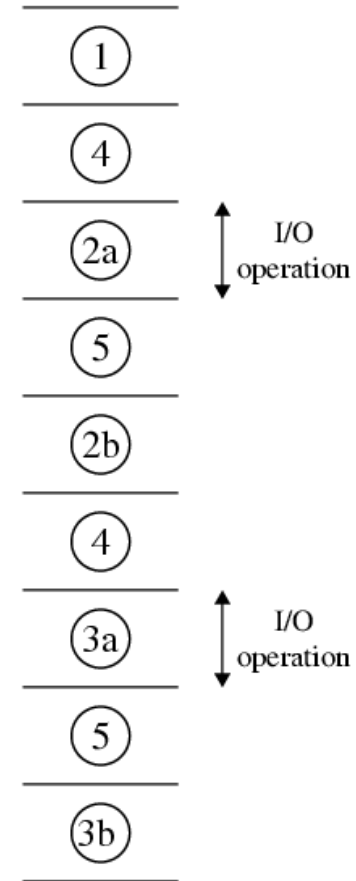


Program Timing

Short I/O Wait



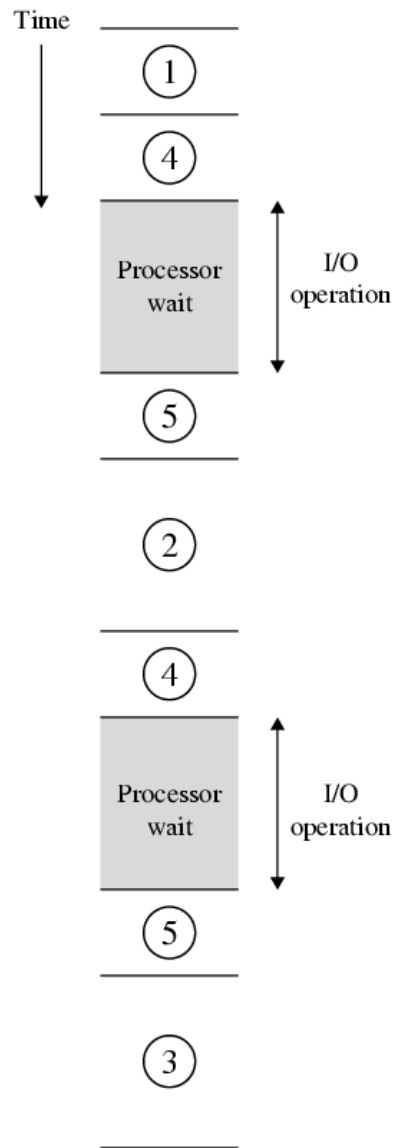
(a) Without interrupts



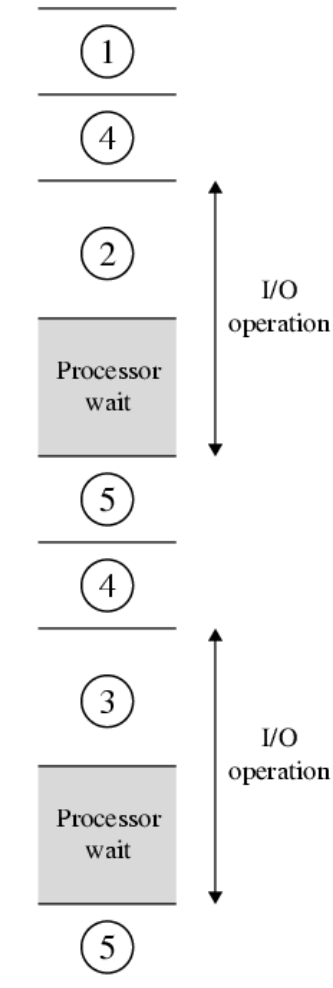
(b) With interrupts

Program Timing

Long I/O Wait

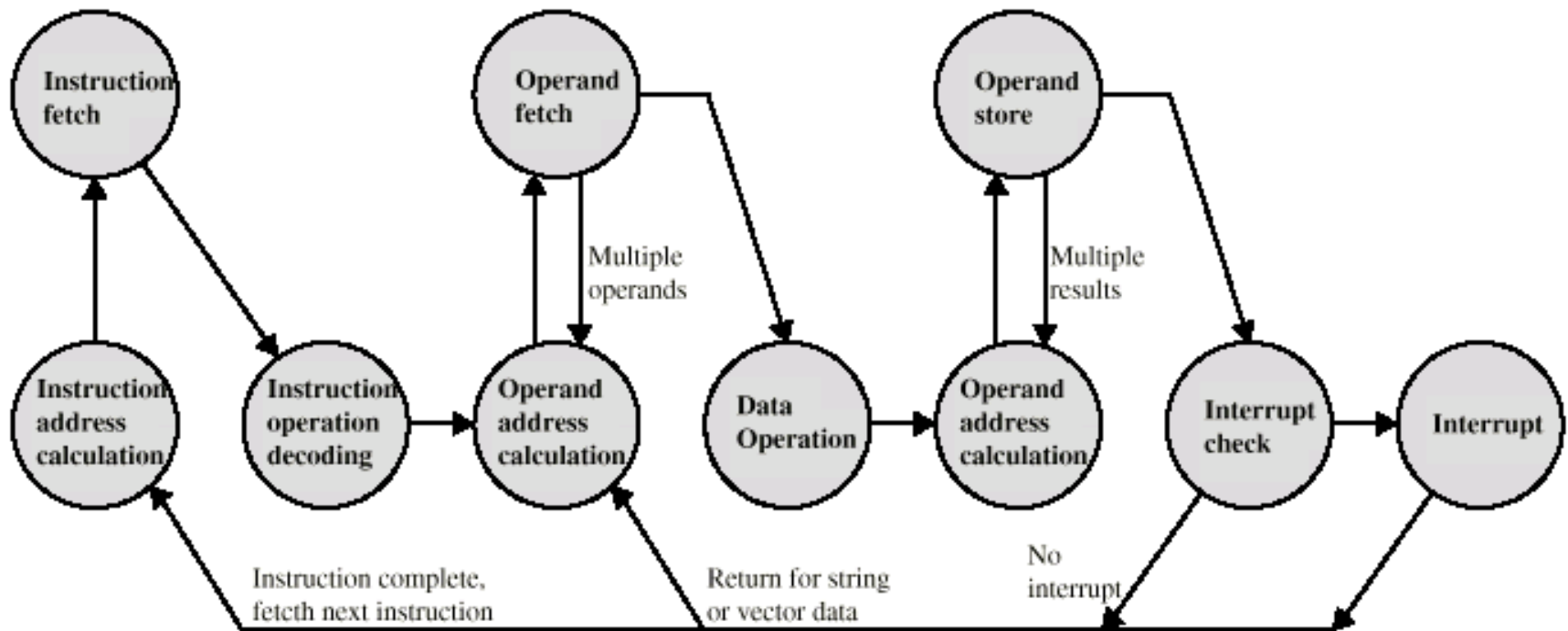


(a) Without interrupts



(b) With interrupts

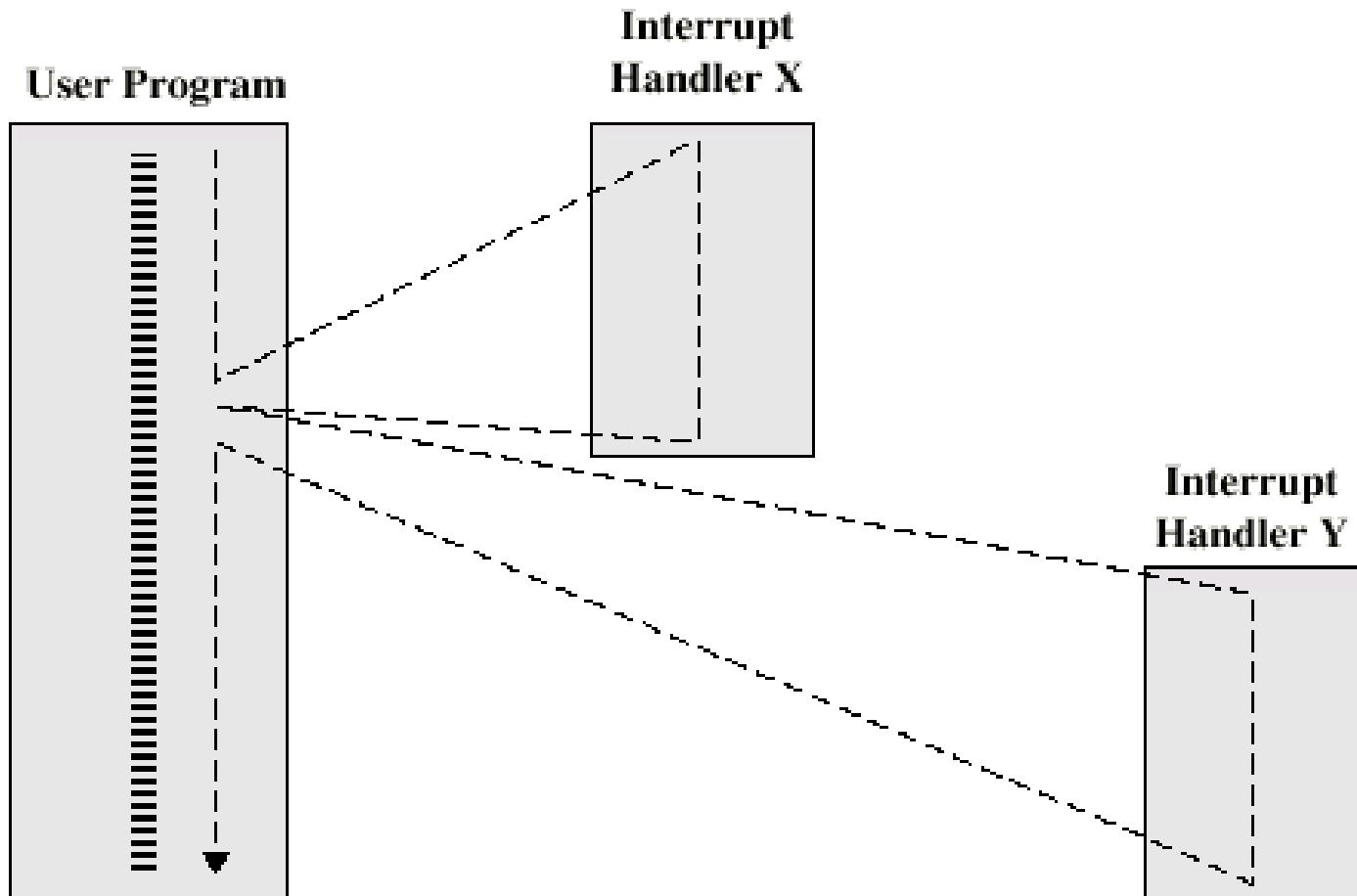
Instruction Cycle (with Interrupts) - State Diagram



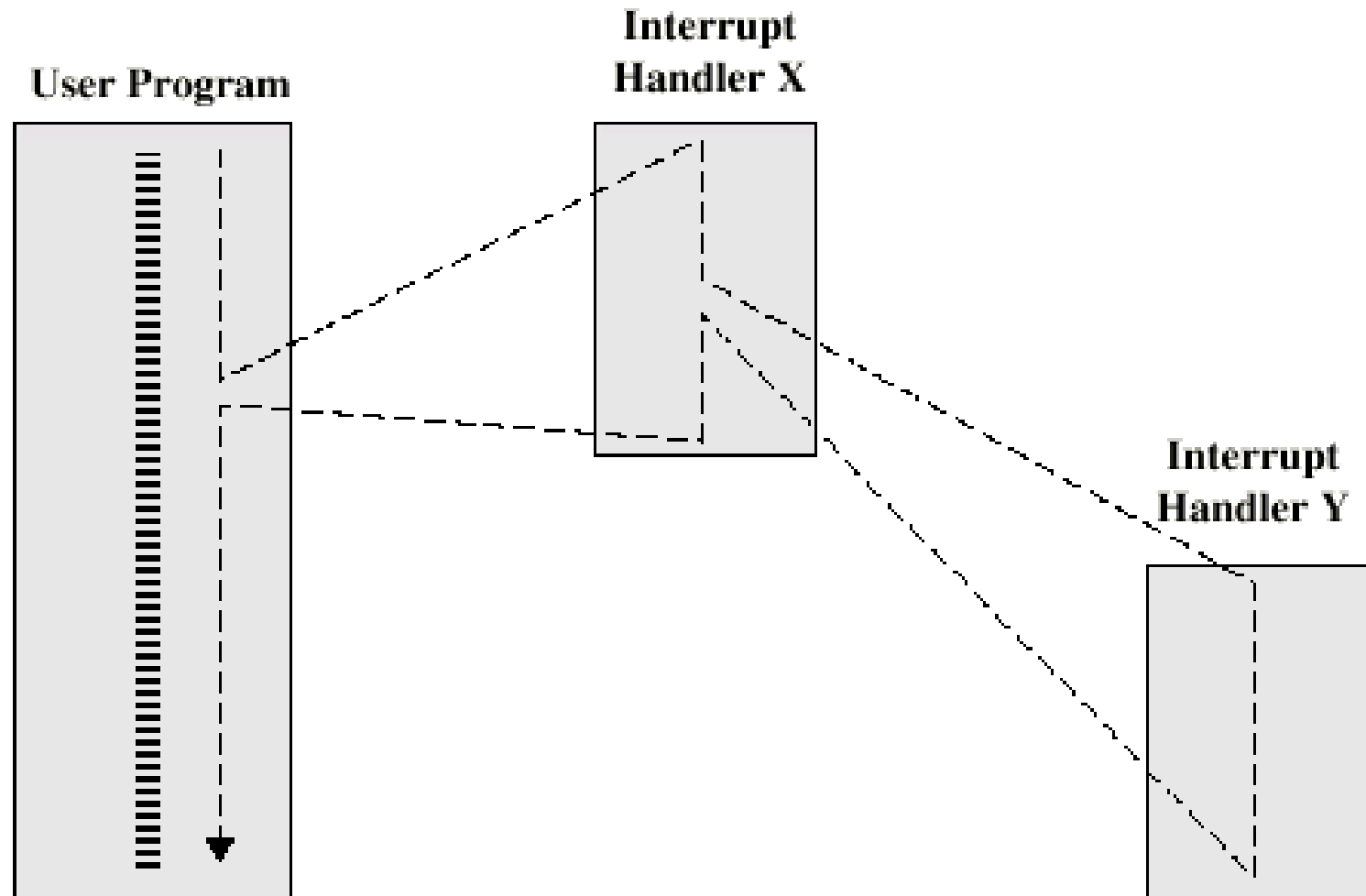
Multiple Interrupts

- Disable interrupts (approach #1)
 - Processor will ignore further interrupts whilst processing one interrupt
 - Interrupts remain pending and are checked after first interrupt has been processed
 - Interrupts handled in sequence as they occur
- Define priorities (approach #2)
 - Low priority interrupts can be interrupted by higher priority interrupts
 - When higher priority interrupt has been processed, processor returns to previous interrupt

Multiple Interrupts - Sequential



Multiple Interrupts – Nested



Time Sequence of Multiple Interrupts

