# PROCESSOR ORGANAZATION

INSTRUCTION FORMAT

# Instruction Format

- Computer perform task on the basis of instruction provided.

# What are Instruction formats?

- when the assembler processes an Instruction, it converts the instruction from its mnemonics form to standard machine language format called the "**Instruction format**".

- In the process of conversion the assembler must determine the type of instruction, convert symbolic labels and explicit notation to a base/displacement format, determine the lengths of certain operands and parse any literal and constants.

# What are Instruction formats?

- An instruction format defines layout of bits of an instruction, in terms of its constituent parts.
- An instruction format must include an opcode and implicitly or explicitly, zero or more operands.
- Each explicit operand is referenced using one of addressing modes.
- Format must, implicitly or explicitly, indicate addressing mode for each operand.

# Introduction

- An instruction is of various length depending upon the number of addresses it contain. Generally CPU organization are of three types on the basis of number of address fields

# General Instruction Format

Opcode-Field Address-Field

Op-field: specifies the operation to be performed;

Address-field: provides operands or the CPU register/MM addresses of the operands.

# Interrupt Cycle

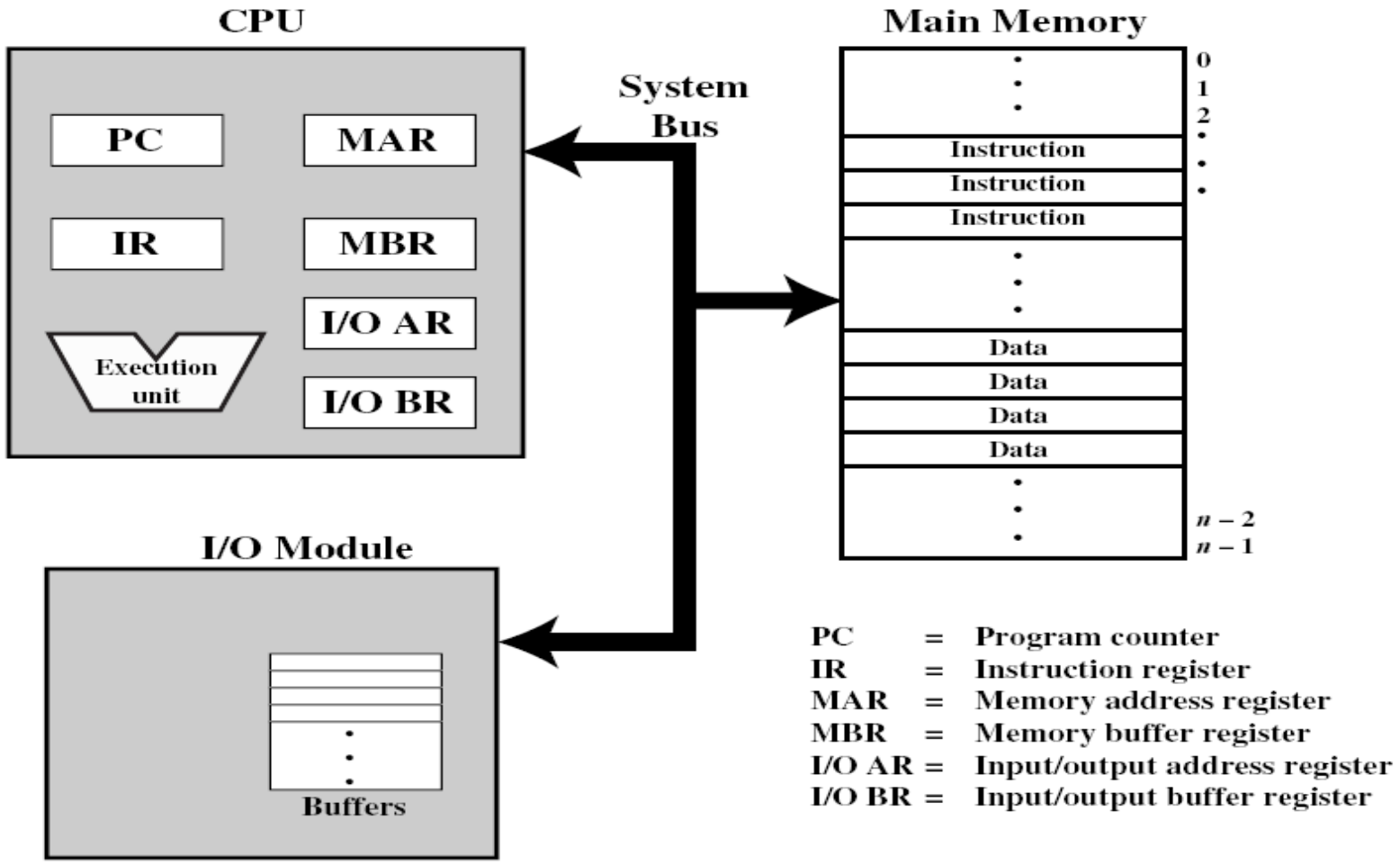
# Key Concept

- Data and instructions require single read-write memory
- Memory contents are addressable by location regardless of whether content is data or instruction
- Execution of code is sequential from one instruction to the next unless a jump is encountered

# Key Concept

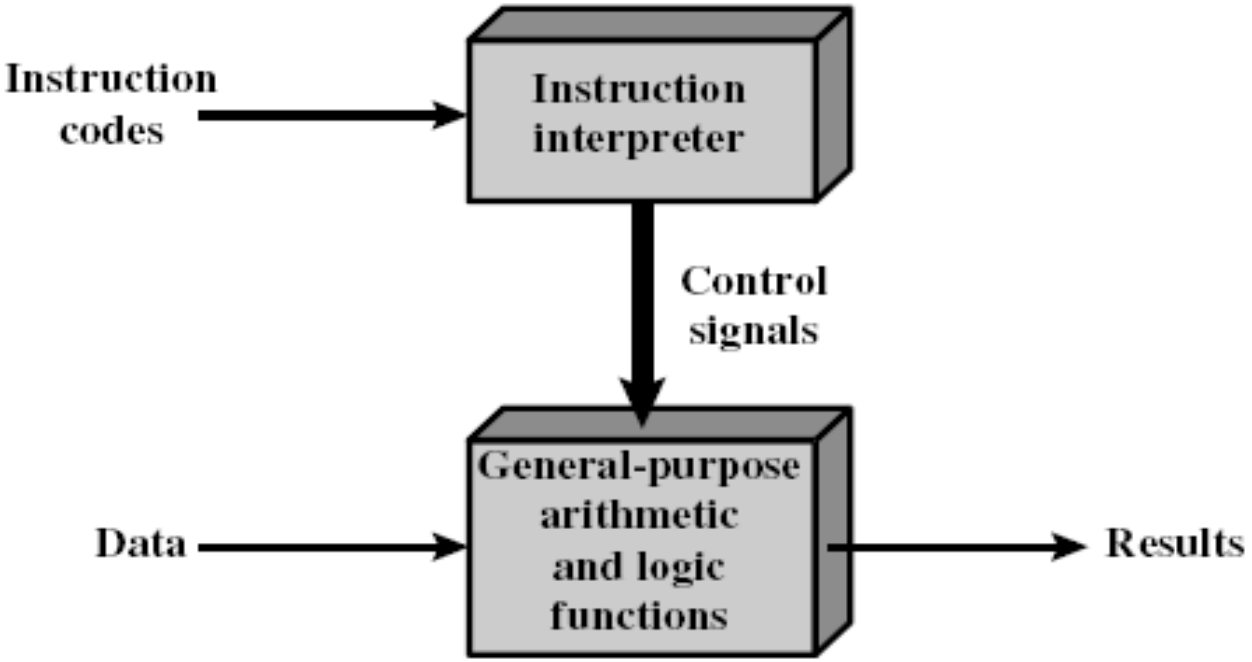
- Each step activates a set of control signals to control general purpose logic
- Each step is an arithmetic or logical operation
- For each operation, a different set of control signals is needed
- Program is  the sequence of steps
- Programming is not rewiring

# Computer Components: Top-Level View

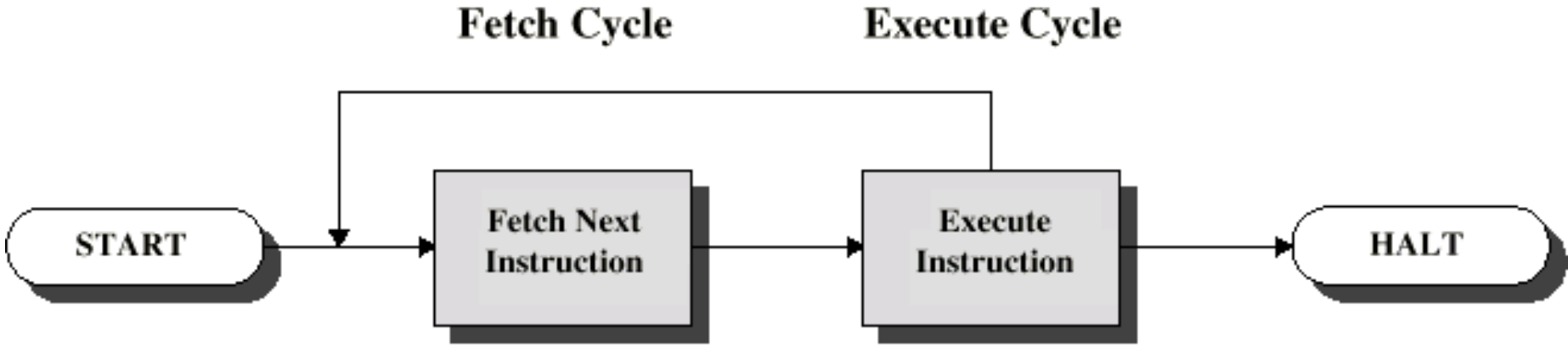


# Instruction Interpreter

- we need a device to accept instruction codes and turn them into control signals for the arithmetic and logic hardware.



- Instruction cycle is not the same thing as a clock cycle
- Two steps:
  - Fetch
  - Execute



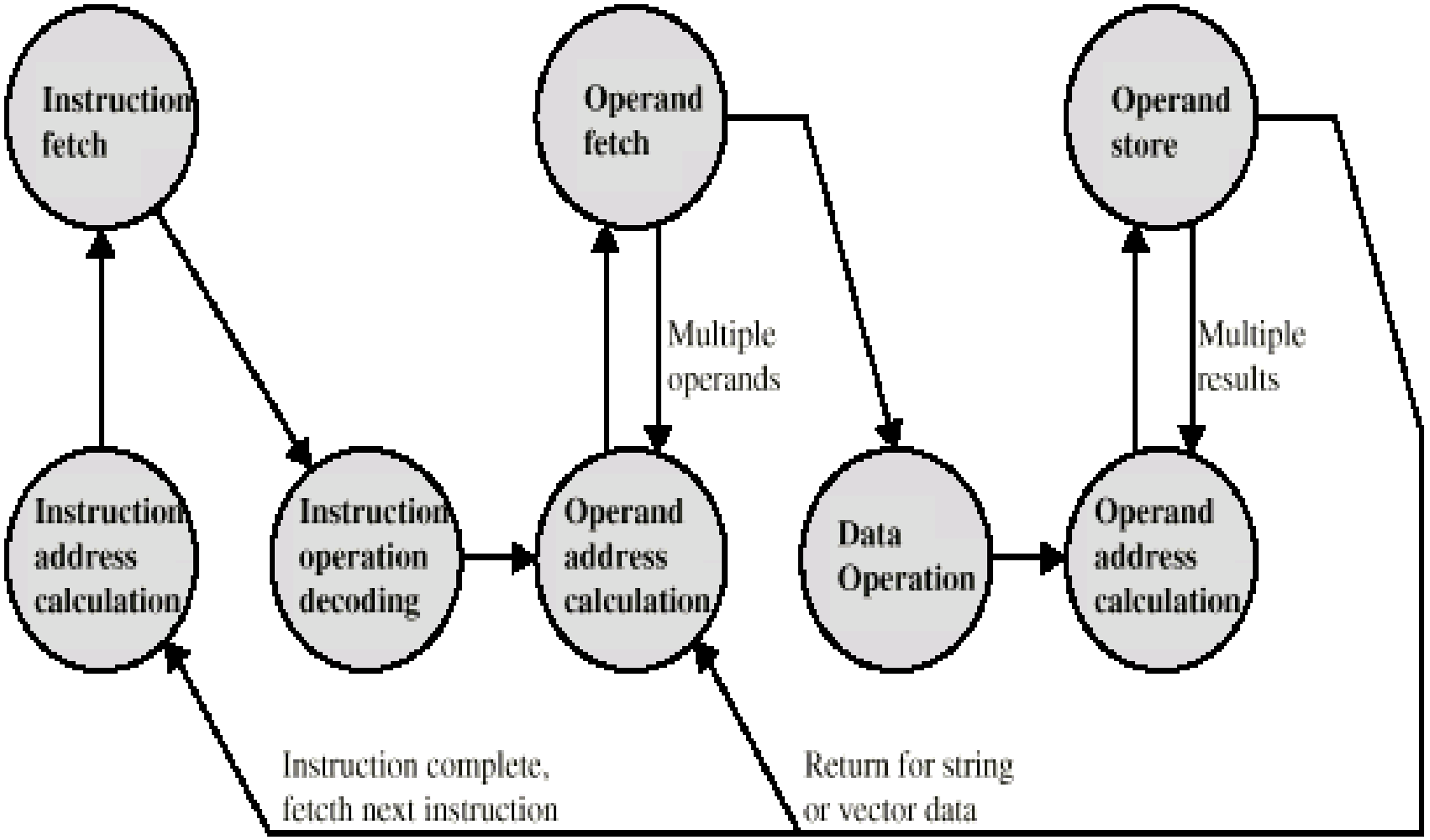
# Fetch Cycle

- Program Counter (PC) holds address of next instruction to fetch
- Processor fetches instruction from memory location pointed to by PC
- Increment PC unless instructed otherwise
- Instruction loaded into Instruction Register (IR)
- Processor interprets instruction and performs required actions
- What problems can you predict happening in this cycle?

# Execute Cycle

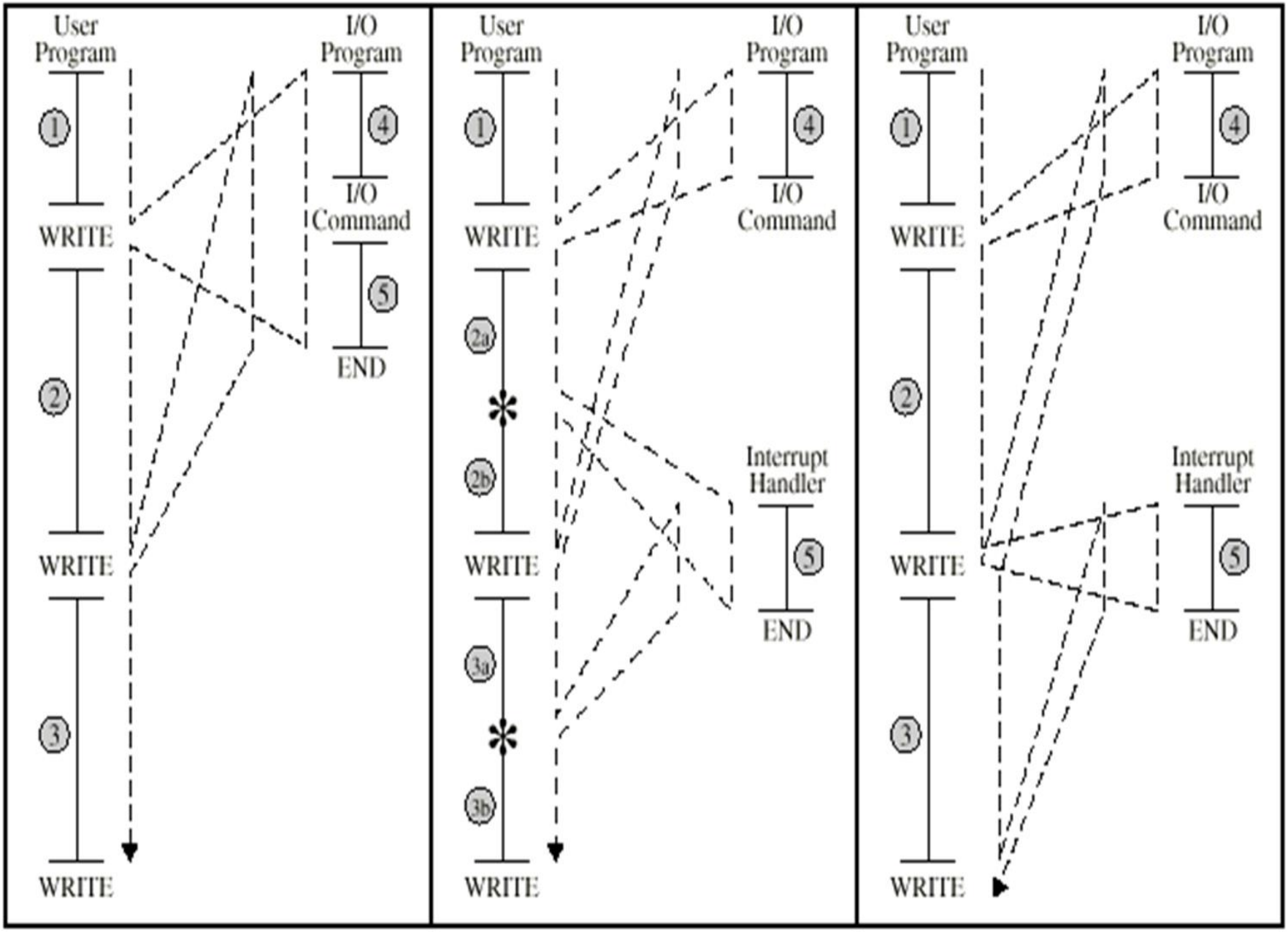
- Processor-memory
  - data transfer between CPU and main memory
- Processor I/O
  - Data transfer between CPU and I/O module
- Data processing
  - Some arithmetic or logical operation on data
- Control
  - Alteration of sequence of operations, e.g. jump
- Combination of above

# Instruction Cycle State Diagram



# Interrupts

- No special code is needed in main code
- Interrupt Service Routines (ISR) handle condition
- Interrupts may be disabled; pending interrupts serviced as soon as interrupts are enabled again
  - Global enabling – affects all maskable interrupts
  - Local enabling – affects individual interrupts

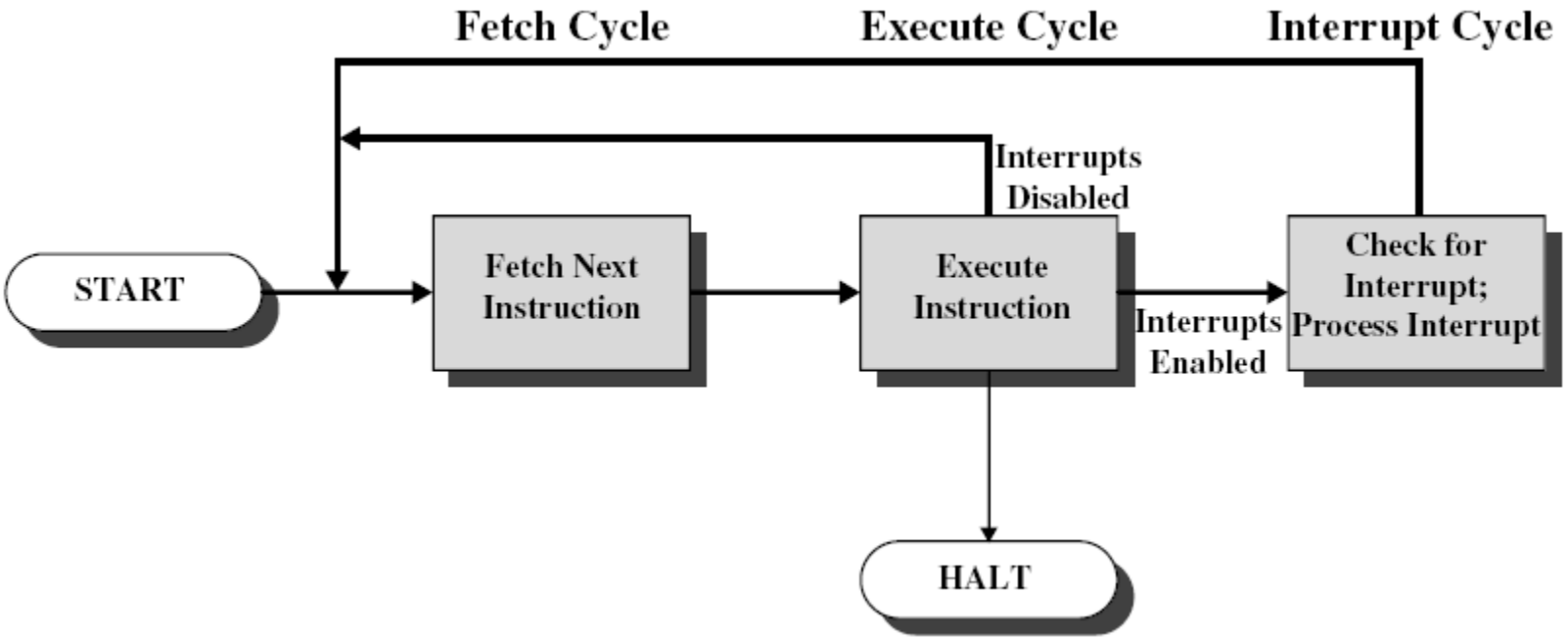



(a) No interrupts

(b) Interrupts; short I/O wait

(c) Interrupts; long I/O wait

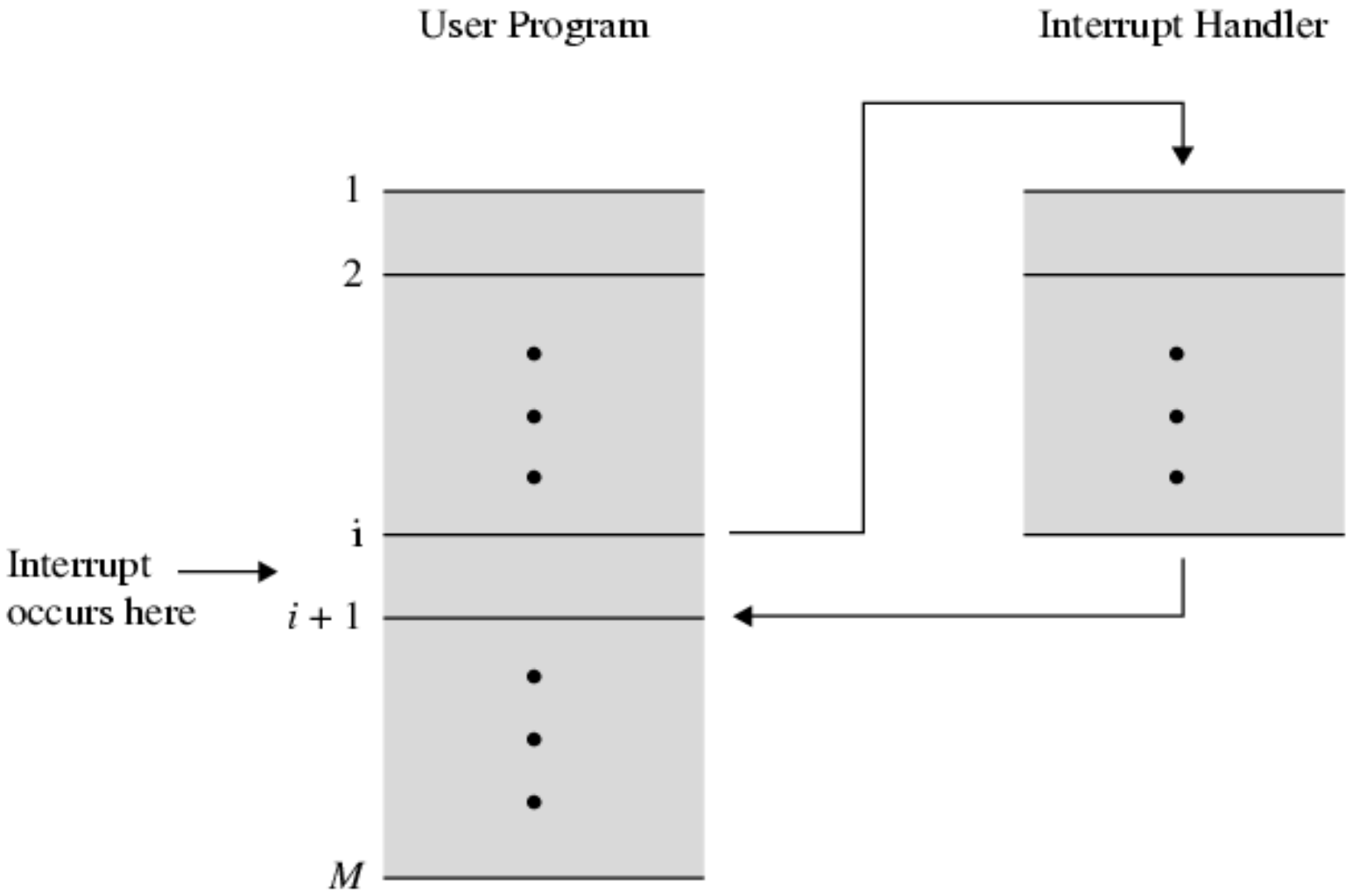
# Interrupt Cycle



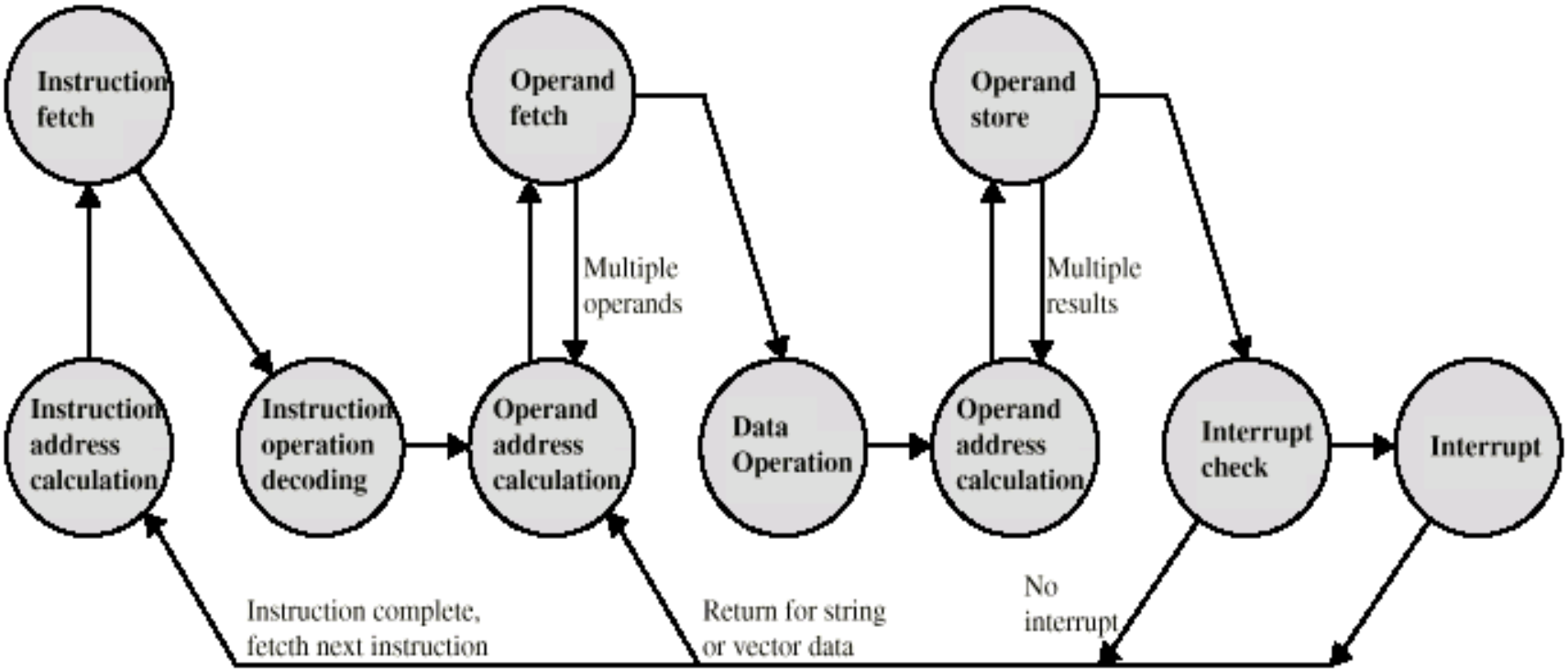
# Interrupt Cycle (continued)

- Added to instruction cycle
- Processor checks for interrupt
- If no interrupt, fetch next instruction
- If interrupt pending:
  - Suspend execution of current program
  - Save context on stack (typically registers, PC, flags, etc.)
  - Set PC to start address of interrupt handler routine
  - Process interrupt
  - Restore context and continue interrupted program

# Transfer of Control via Interrupts



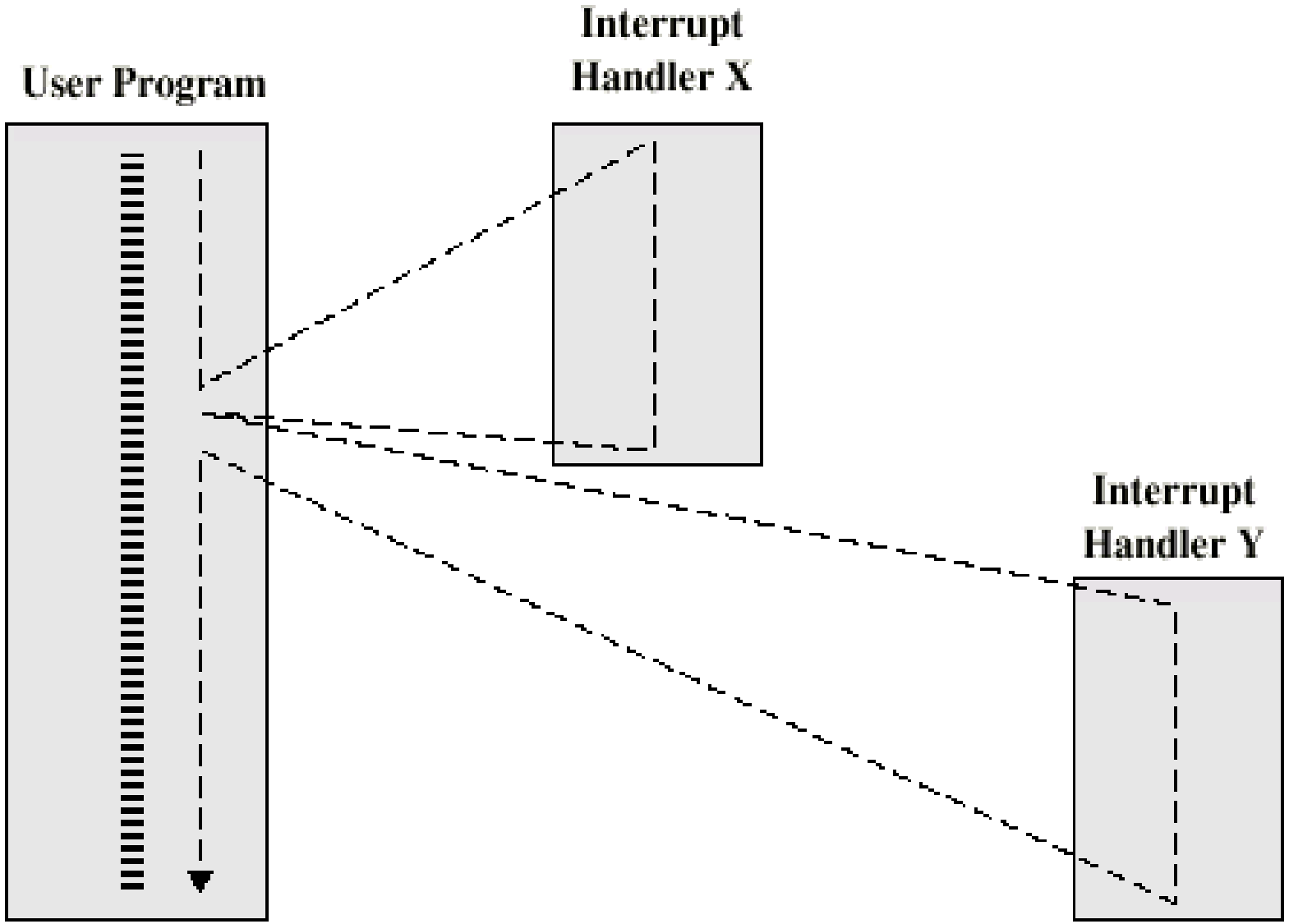
# Instruction Cycle (with Interrupts) - State Diagram



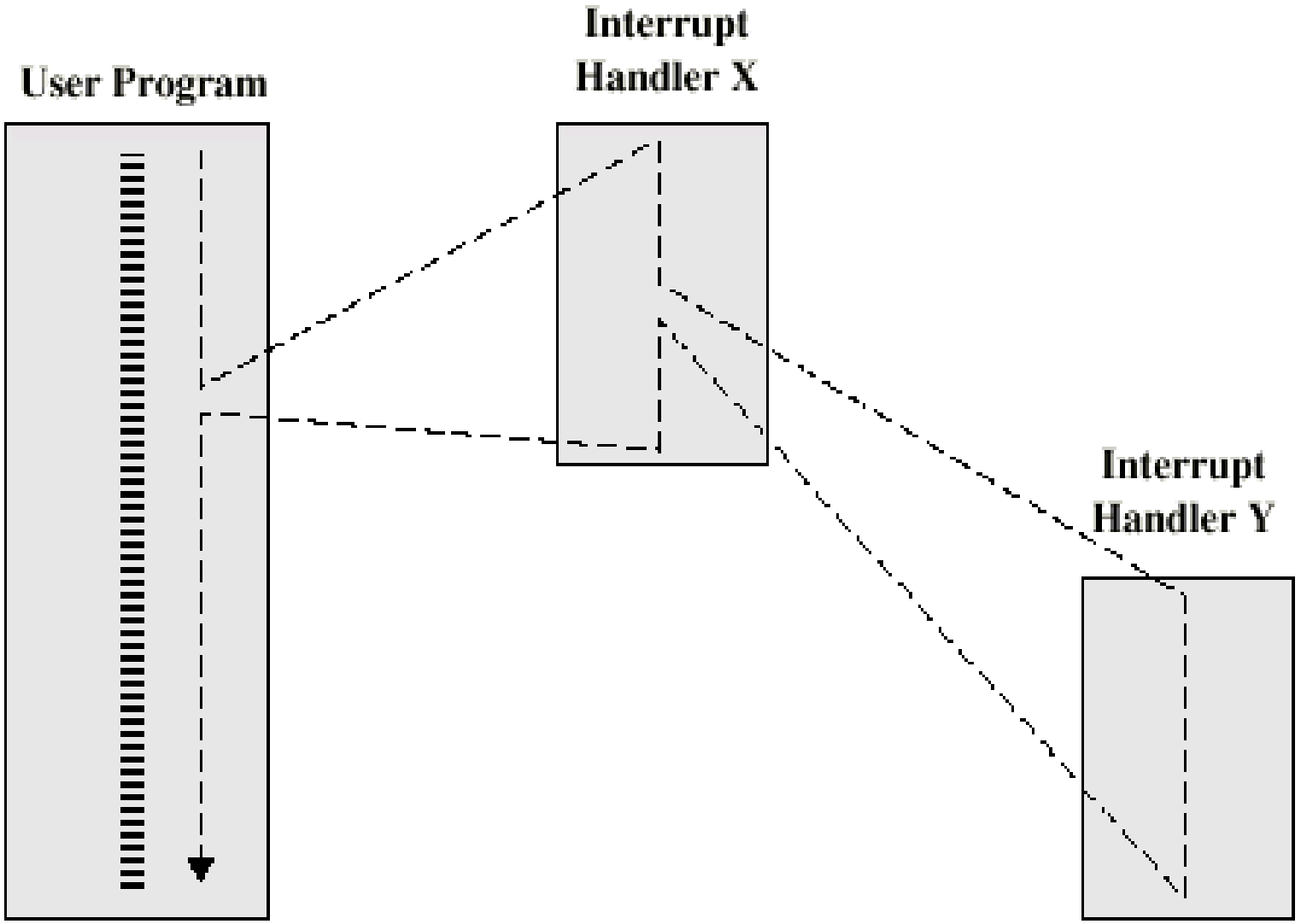
# Multiple Interrupts

- Disable interrupts
  - Processor can ignore further interrupts whilst processing one interrupt or interrupts may be nested
  - Ignored interrupts remain pending and are checked after first interrupt has been processed
- Define priorities
  - Low priority interrupts can be interrupted by higher priority interrupts
  - When higher priority interrupt has been processed, processor returns to previous interrupt

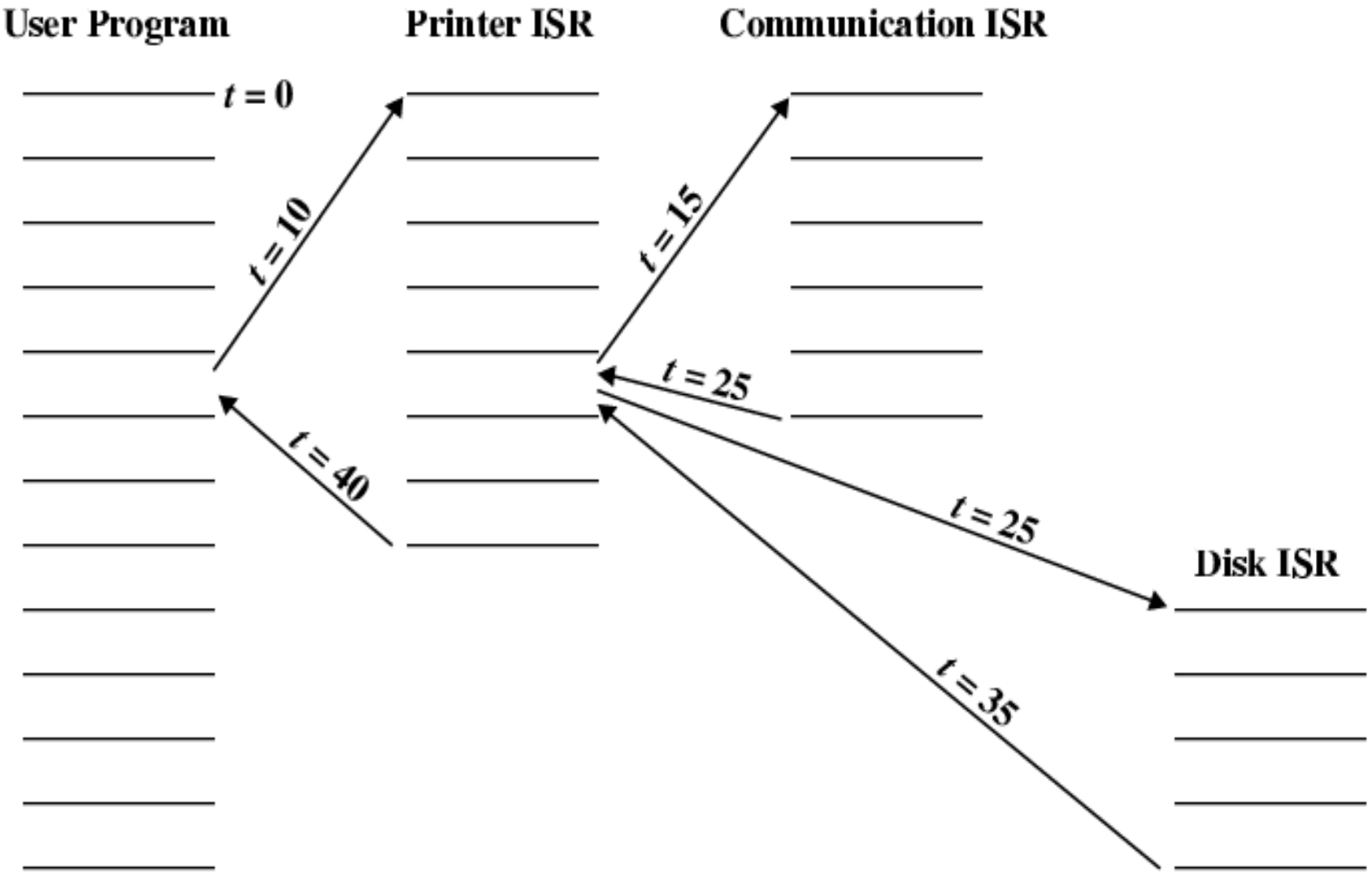
# Multiple Interrupts - Sequential



# Multiple Interrupts – Nested



# Time Sequence of Multiple Interrupts



# I/O module

- I/O modules occasionally require attention, usually in the form of a data transfer
- Processor can simply transfer data back and forth with the device as if it were memory
- Alternatively, processor can grant I/O module permission to write directly to memory – Direct Memory Access (DMA) – Interrupt occurs when DMA is complete

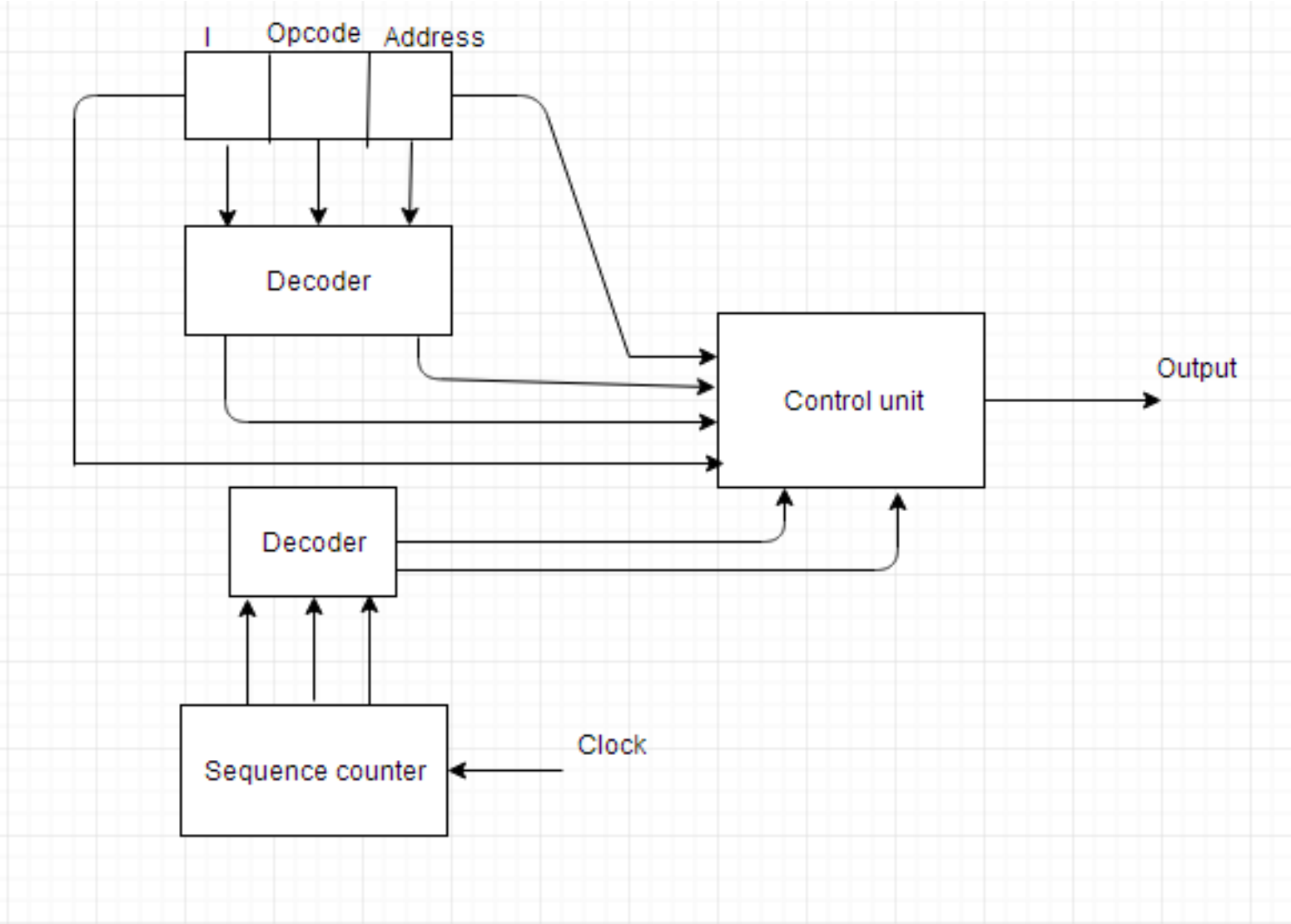
# DESIGN OF CONTROL UNIT

# Control unit

- Control unit generates timing and control signals for the operations of the computer.
- The control unit communicates with ALU and main memory.
- It also controls the transmission between processor, memory and the various peripherals.
- It also instructs the ALU which operation has to be performed on data.
- Control unit can be designed by two methods

# Hardwired Control Unit

- it is implemented with the help of gates, flip flops, decoders etc. in the hardware.
- The inputs to control unit are the instruction register, flags, timing signals etc.
- This organization can be very complicated if we have to make the control unit large.
- If the design has to be modified or changed, all the combinational circuits have to be modified which is a very difficult task.


I
Opcode
Address
Decoder
Control unit
Output
Decoder
Sequence counter
Clock

# Microprogrammed Control Unit

- It is implemented by using programming approach.
- A sequence of micro operations is carried out by executing a program consisting of micro-instructions.
- In this organization any modifications or changes can be done by updating the micro program in the control memory by the programmer

# Difference between Hardwired Control and Microprogrammed Control

| Hardwired Control | Microprogrammed Control |
|---|---|
| Technology is circuit based. | Technology is software based. |
| It is implemented through flip-flops, gates, decoders etc. | Microinstructions generate signals to control the execution of instructions. |
| Fixed instruction format. | Variable instruction format (16-64 bits per instruction). |
| Instructions are register based. | Instructions are not register based. |
| ROM is not used. | ROM is used. |
| It is used in RISC. | It is used in CISC. |
| Faster decoding. | Slower decoding. |
| Difficult to modify. | Easily modified. |
| Chip area is less. | Chip area is large. |