COMPUTER ARCHITECTURE

Intensive Computation

Annalisa Massini

Lecture 1

2022-2023

OVERVIEW OF COMPUTER ARCHITECTURE AND ORGANIZATION

- On these slides you will find a summary of the conventional computer architecture and organization: **Von Neumann architecture**

- http://WilliamStallings.com/COA/COA7e.html

Architecture & Organization

In describing computers, a distinction is often made:

- **Architecture -** attributes visible to the programmer
  - Instruction set, number of bits used for data representation, I/O mechanisms, addressing techniques
  - e.g. Is there a multiply **instruction**?

- **Organization -** operational units and their interconnections that realize the architectural specifications
  - Control signals, interfaces, memory technology
  - e.g. Is there a **hardware** multiply unit or is it done by repeated addition?

Structure and Function

- A **computer** is a complex system containing millions of elementary electronic components

- The key to describe a computer is to recognize its hierarchical nature, as for most complex systems:
  - At each level, the system consists of a **set of components**
  - The **interrelationships** between components
  - The **behavior at each level** depends only on a simplified, abstracted characterization of the system at the next lower level

Structure and Function

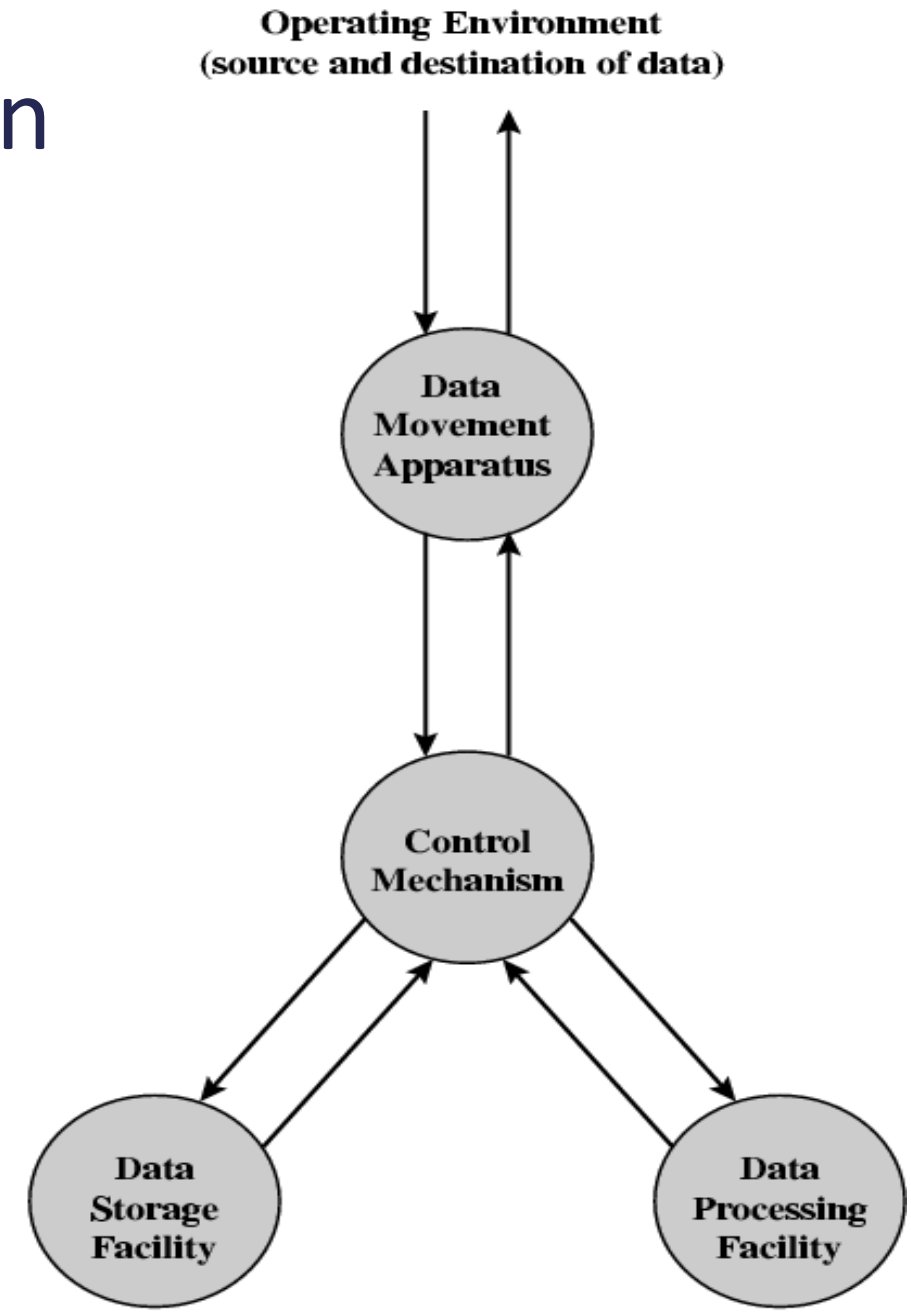
- At each level, we are concerned with:
  - **Structure** - the way in which components relate to each other
  - **Function** - the operation of individual components as part of the structure

- The basic functions that a computer can perform are:
  - Data processing
  - Data storage
  - Data movement
  - Control

Structure and Function

The computer must be able:

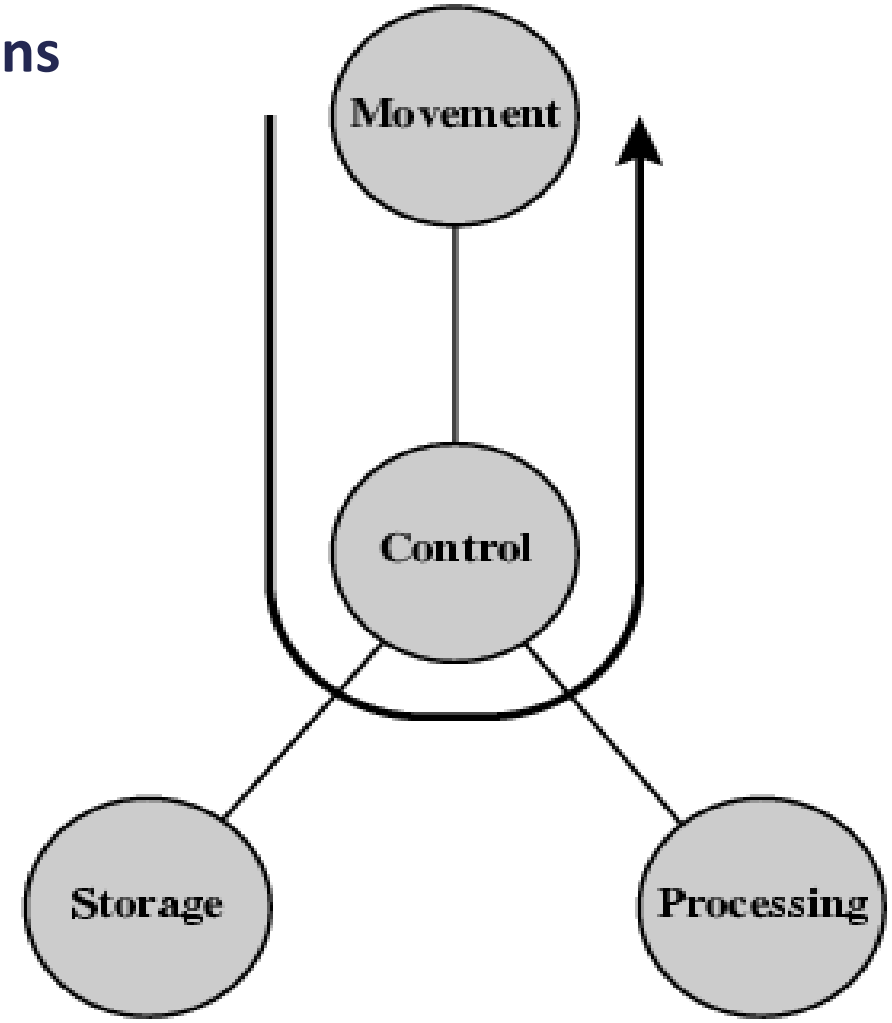
- To **process data**, that may take a wide variety of forms
- To **store data** - temporarily store at least those pieces of data that are being worked on at any given moment
- To **move data** between itself and the outside world
- To **control** these three functions



Operations - Data movement

Four possible types of operations

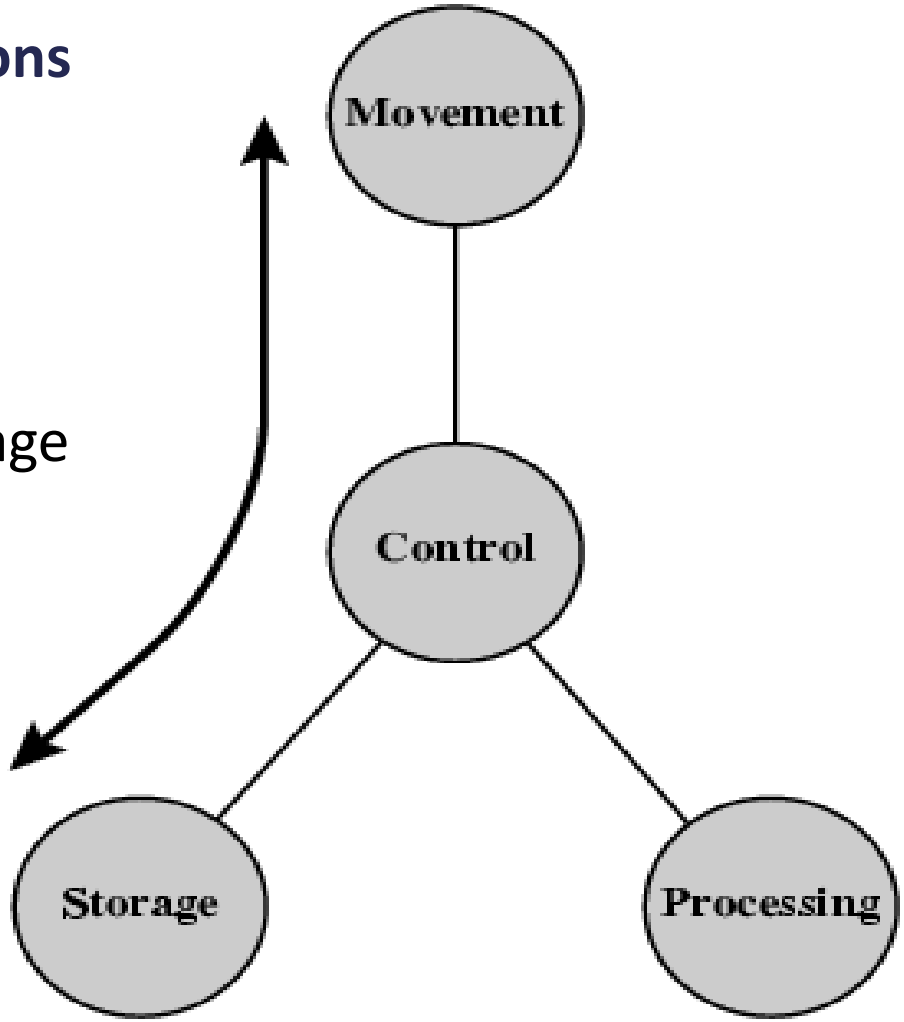
The *computer* can function as a data movement device, simply transferring data from one peripheral or communications line to another



Operations - Storage

Four possible types of operations

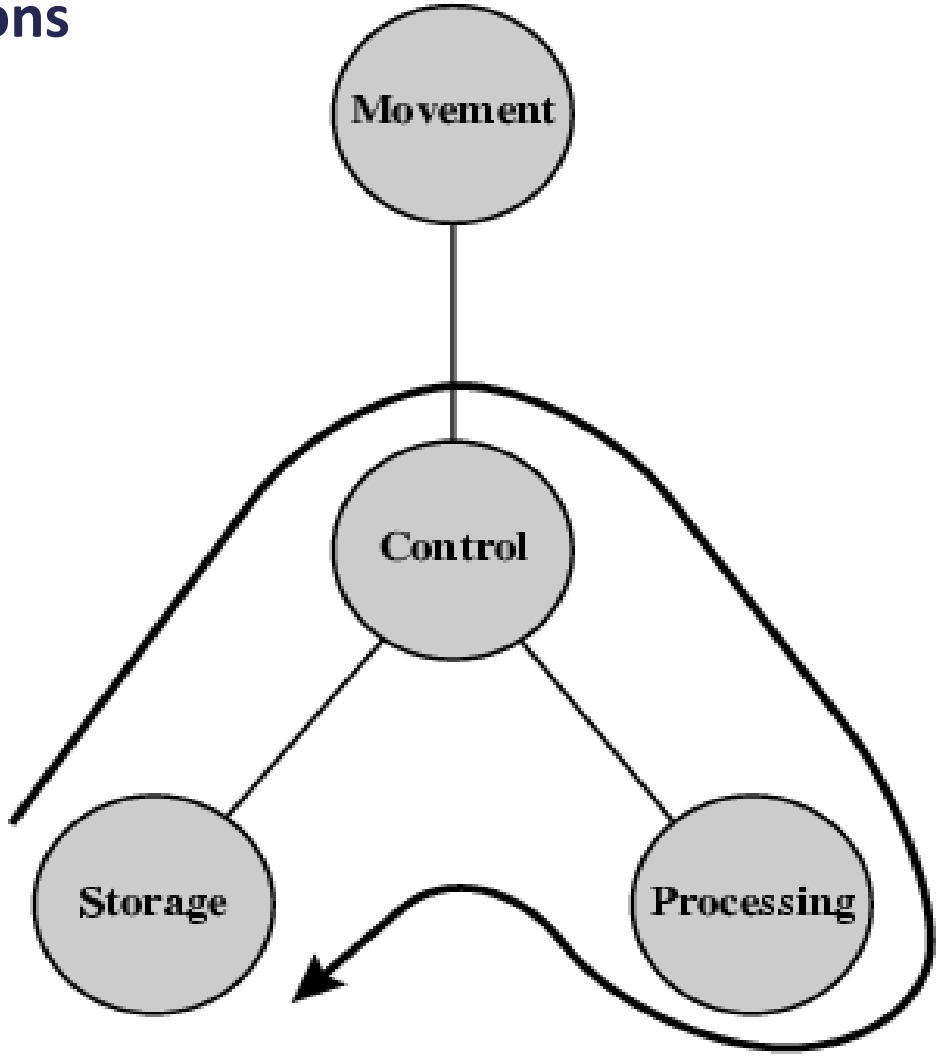
The *computer* can function as a data storage device, with data transferred from the external environment to computer storage (*read*) and vice versa (*write*)



Operation - Processing from/to storage

Four possible types of operations

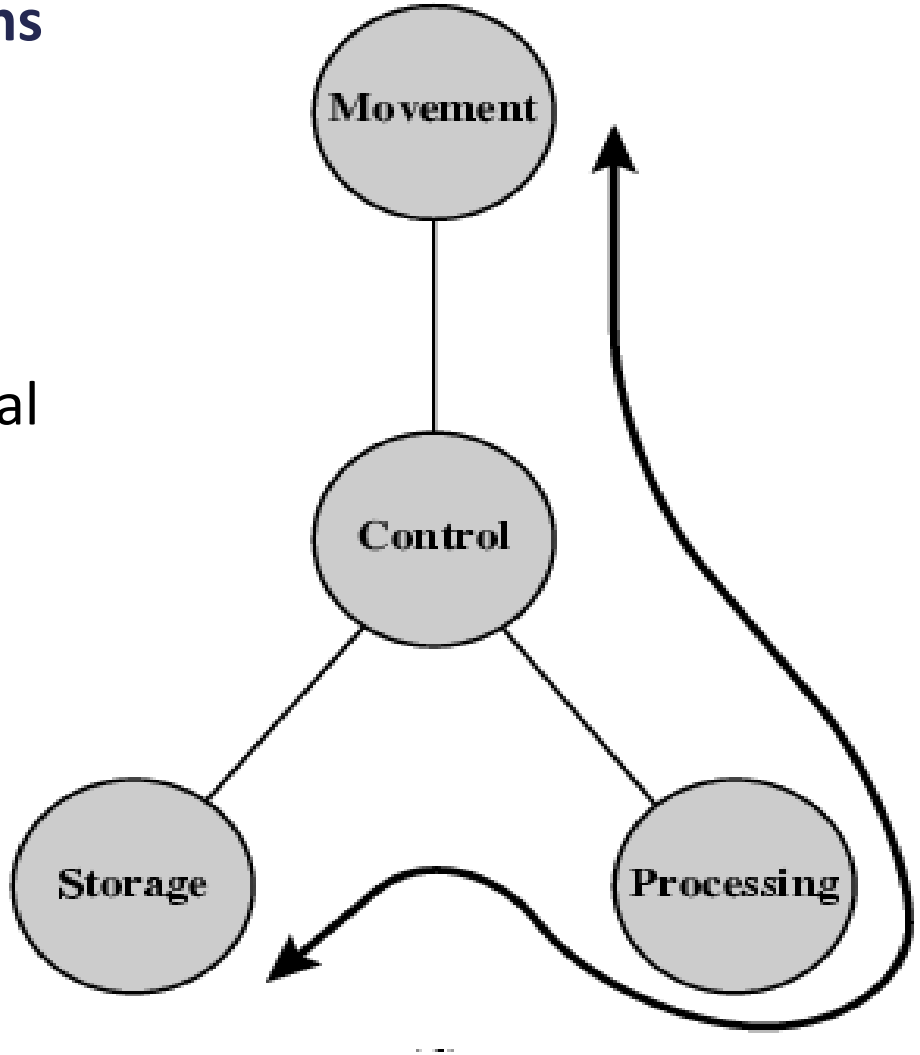
The *computer* can execute operations involving data processing, on *data in storage*



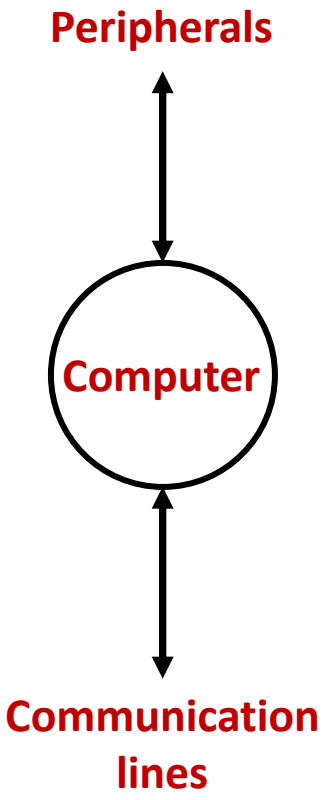
Operation - Processing from storage to I/O

Four possible types of operations

The *computer* can execute operations involving data processing, on *data en route* between storage and the external environment



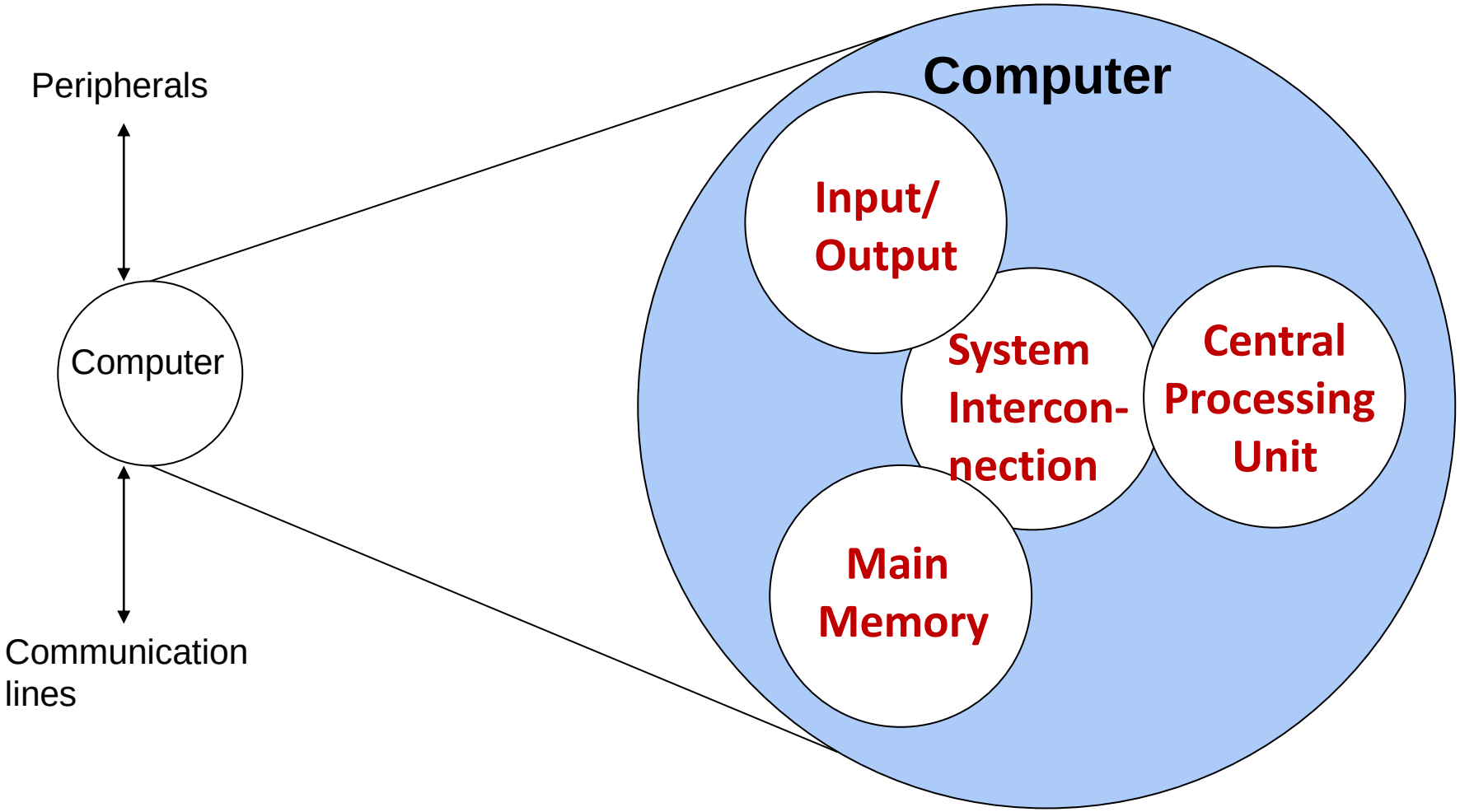
Structure - Top Level



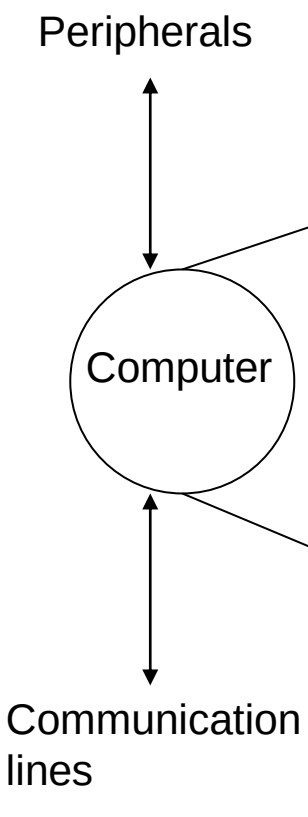
The internal structure of the computer consists of four main structural components:

- **Central processing unit** (CPU): Controls the operation of the computer and performs its data processing functions (processor)
- **Main memory**: Stores data
- **I/O devices**: Moves data between the computer and its external environment
- **System interconnection**: Some mechanism that provides for communication among CPU, main memory, and I/O (for example a system bus)

Structure - Top Level

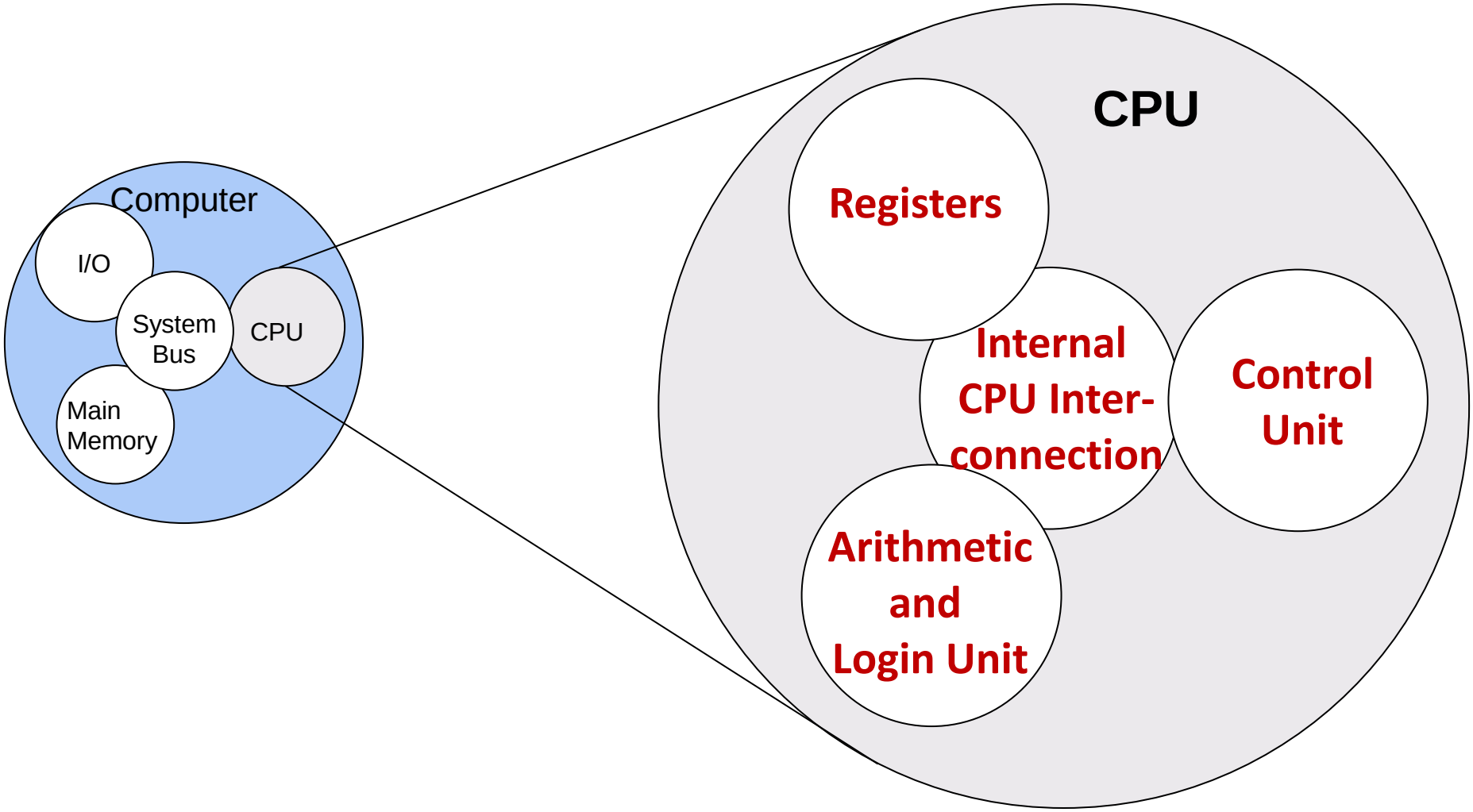


Structure - Top Level



- The **Central Processing Unit** is constituted by:
  - Control Unit
  - Arithmetic and Logic Unit
- Data and instructions get into the system and results out
  - **Input/output**
- Temporary storage of code and results is needed
  - **Main memory**

Structure - The CPU

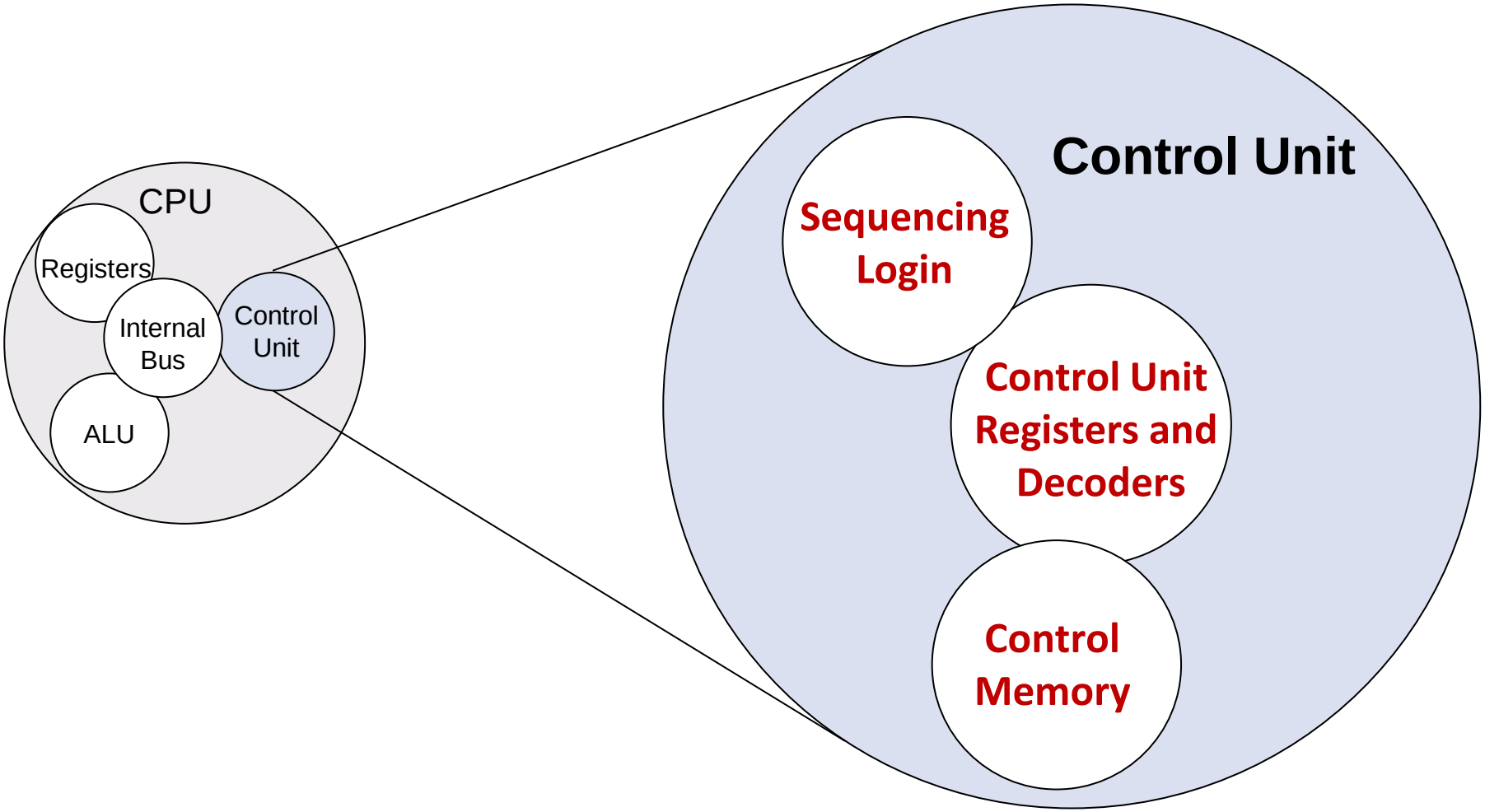


Components

Basic element of a Central Processing Unit (processor)

- Control Unit
- ALU Arithmetic and Logic Unit
- Registers
- Internal data paths
- External data paths

Structure - The Control Unit



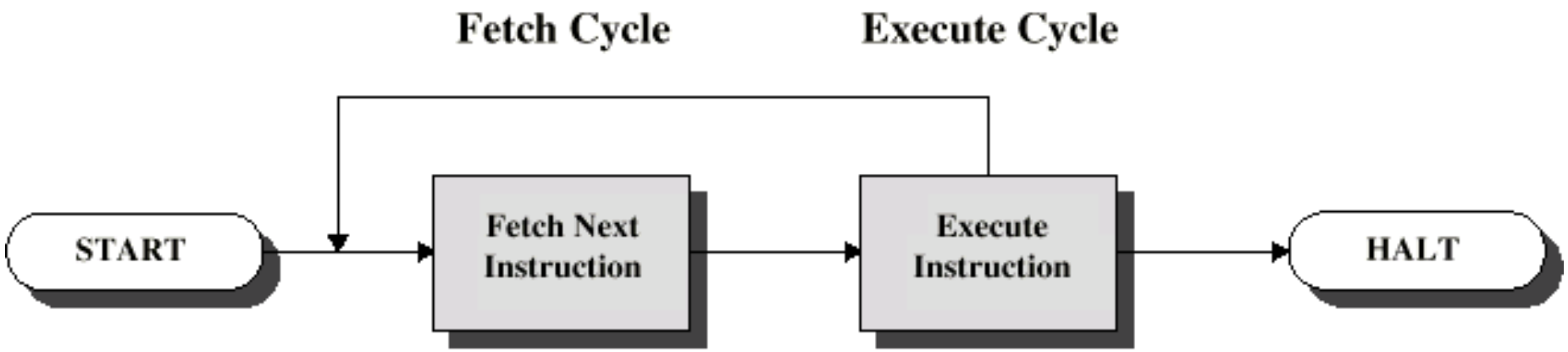
Observations

- Traditionally, the computer has been viewed as a **sequential** machine
- Most **computer programming languages** require the programmer to specify algorithms as ***sequences*** of instructions
- **Processors** execute **programs** by executing machine instructions in a ***sequence*** and one at a time
- Each **instruction** is executed in a ***sequence*** of operations (fetch instruction, fetch operands, perform operation, store results)
- *This view of the computer has never been entirely true*

INSTRUCTION EXECUTION

Instruction Cycle

- The processing required for a single instruction is called an ***instruction cycle***
- The instruction cycle can be illustrated using a simplified two-step description
- The two steps are referred to as the ***fetch cycle*** and the ***execute cycle***



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)
- Processor interprets instruction and performs required actions

Execute Cycle

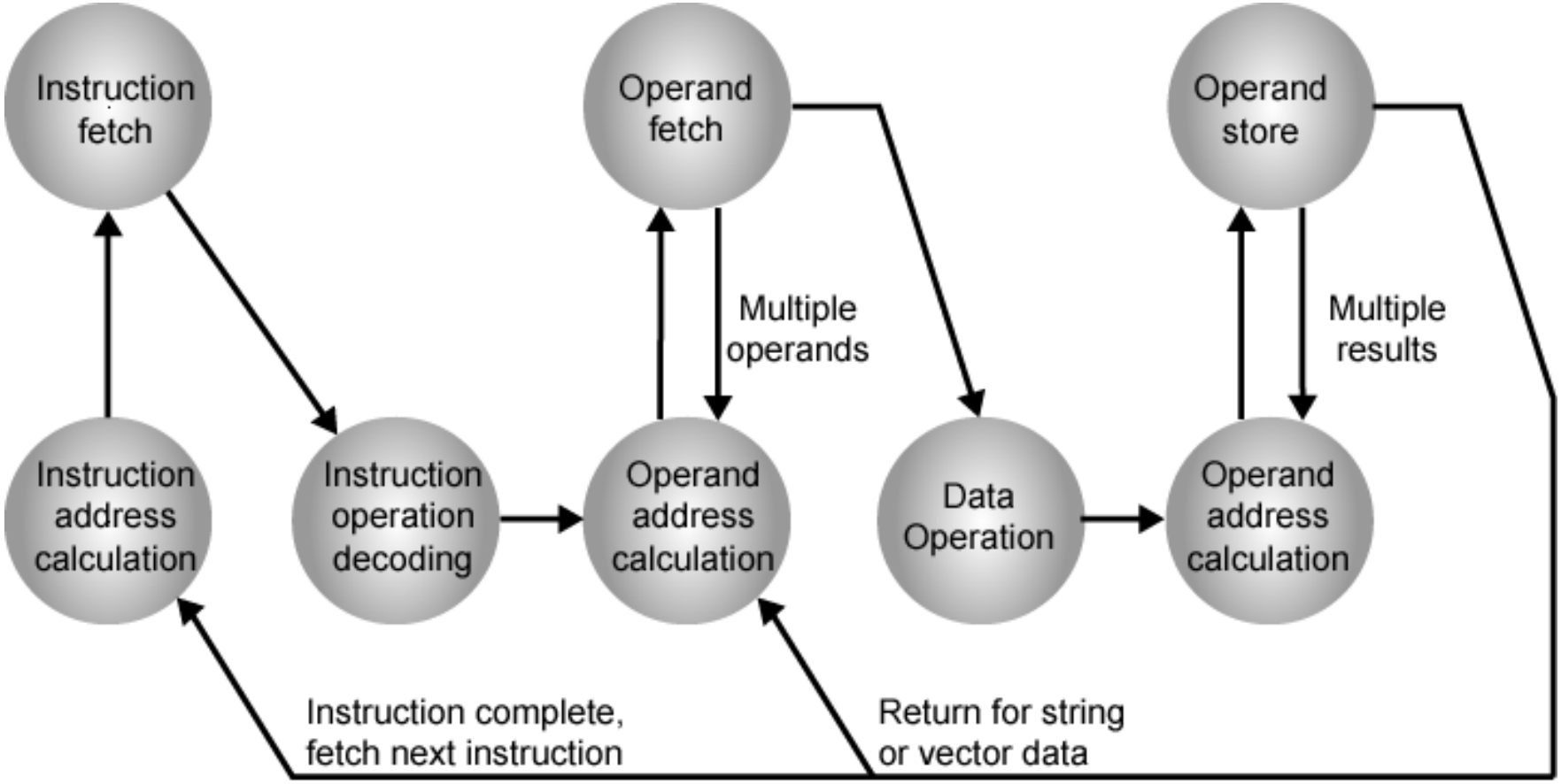
- Processor-memory
  - data transfer between CPU and main memory
- Processor I/O
  - Data transfer between CPU and I/O module
- Data processing
  - Some arithmetic or logical operation on data
- Control
  - Alteration of sequence of operations
  - e.g. jump
- Combination of above

Instruction Execution

The requirements placed on the processor (that is the things that it must do):

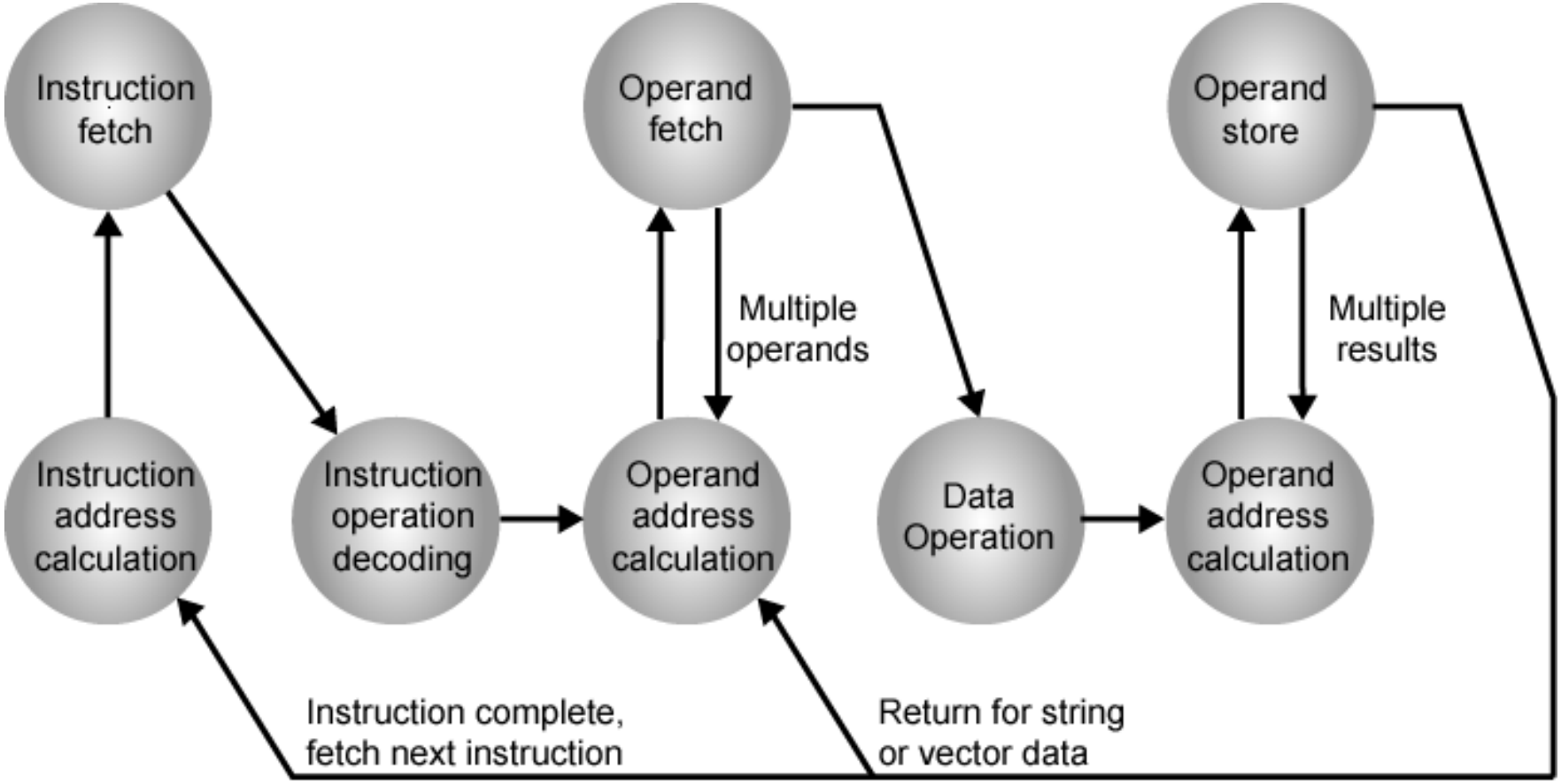
- **Fetch instruction**: The processor reads an instruction from memory (register, cache, main memory)
- **Interpret instruction**: The instruction is decoded to determine what action is required
- **Fetch data**: The execution of an instruction may require reading data from memory or an I/O module
- **Process data**: The execution of an instruction may require performing some arithmetic or logical operation on data.
- **Write data**: The results of an execution may require writing data to memory or an I/O module

Instruction Cycle State Diagram



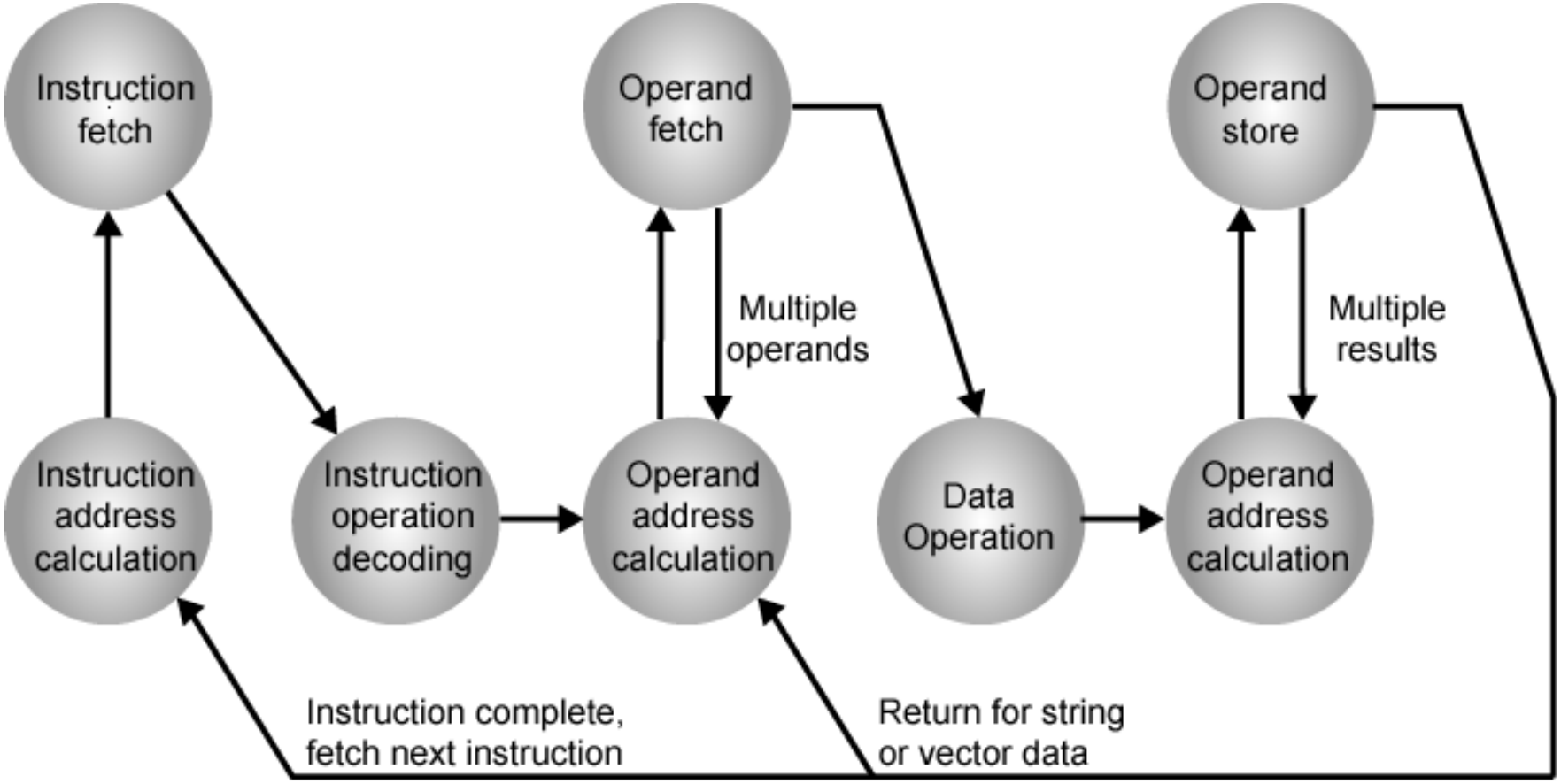
Instruction address calculation (iac): Determine the address of the next instruction to be executed (usually, adding a fixed number to the address of the previous instr.)

Instruction Cycle State Diagram



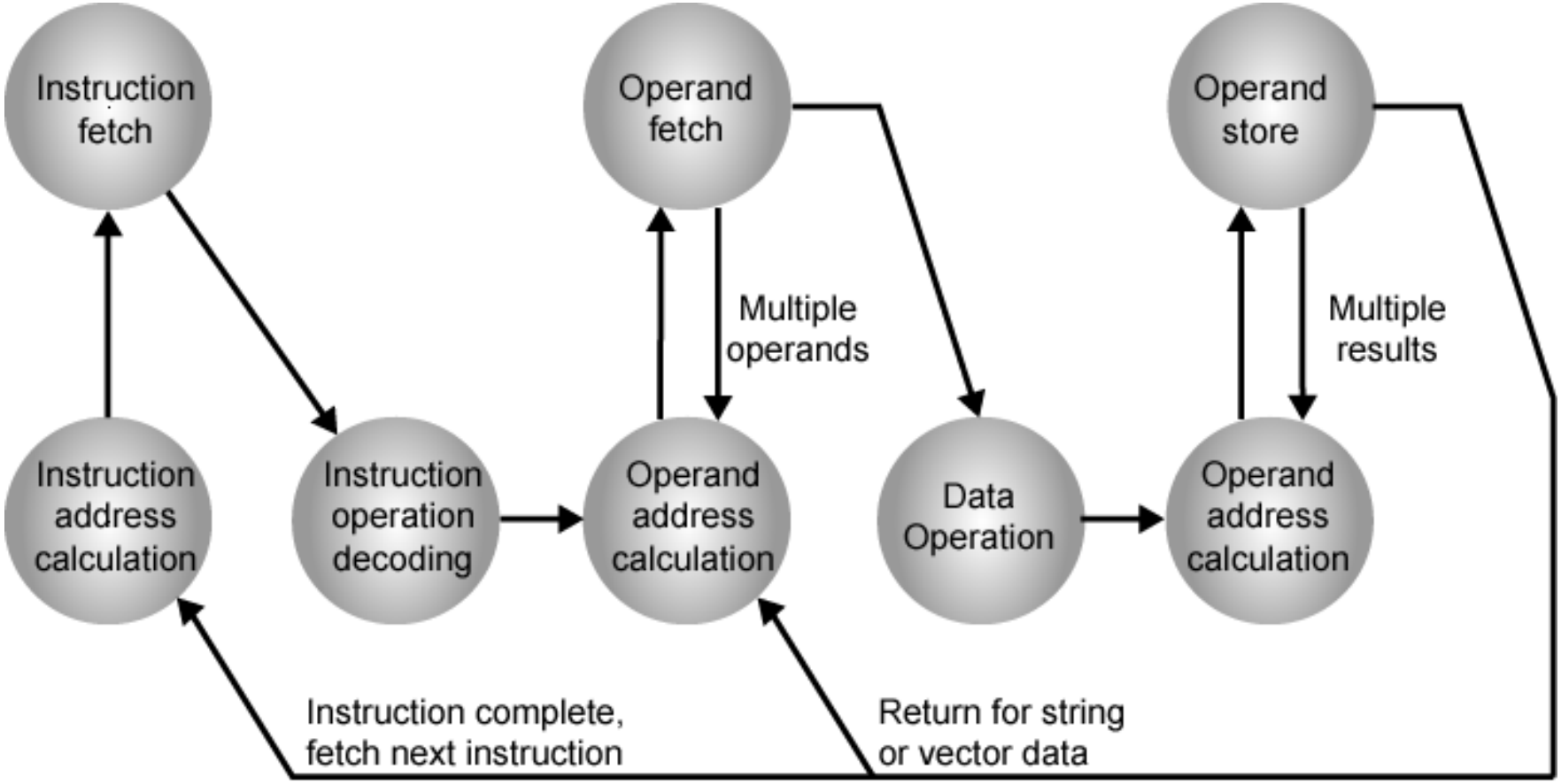
Instruction fetch (if): Read instruction from its memory location into the processor

Instruction Cycle State Diagram



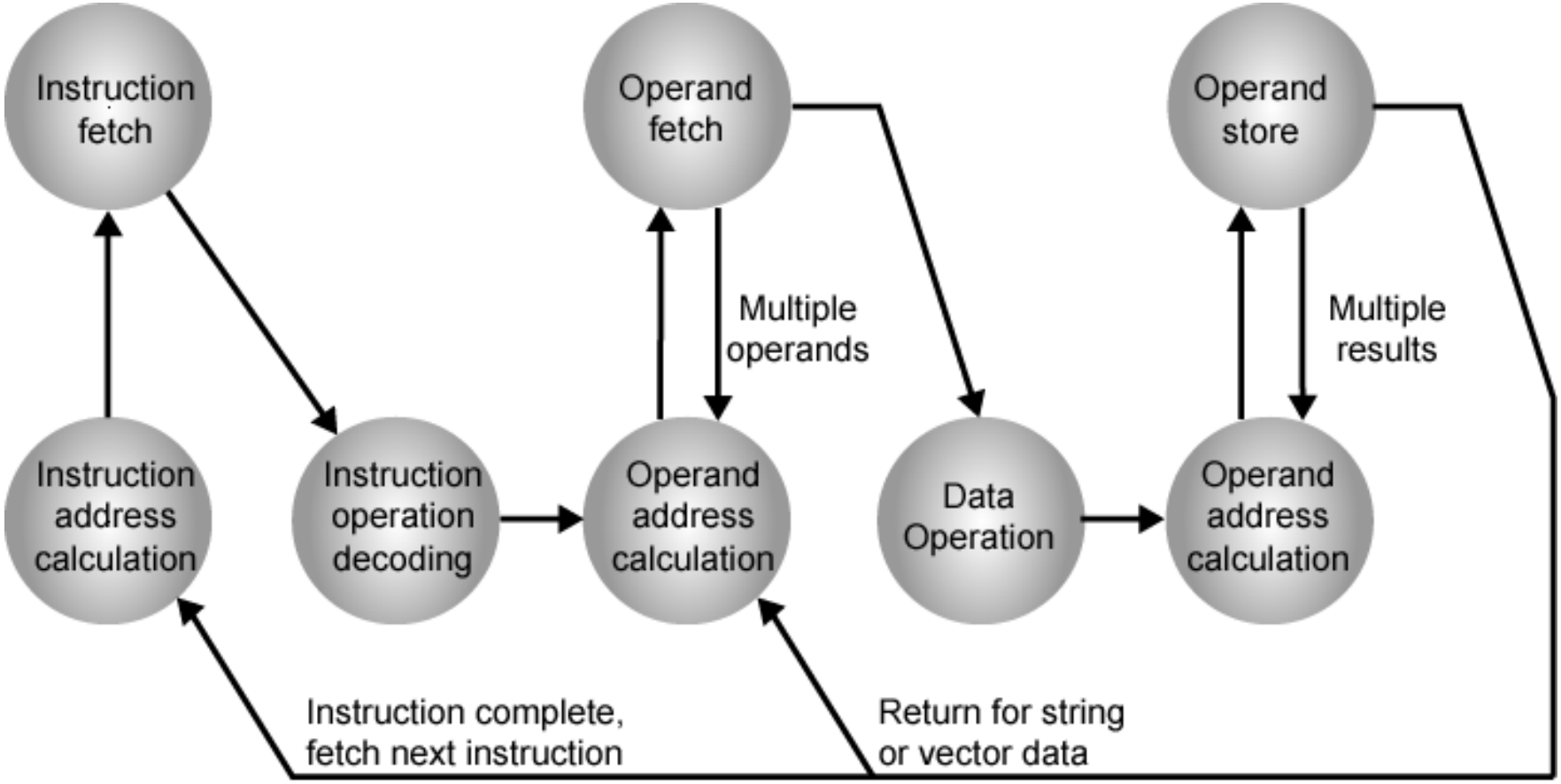
Instruction operation decoding (iod): Analyze instruction to determine type of operation to be performed and operand(s) to be used

Instruction Cycle State Diagram



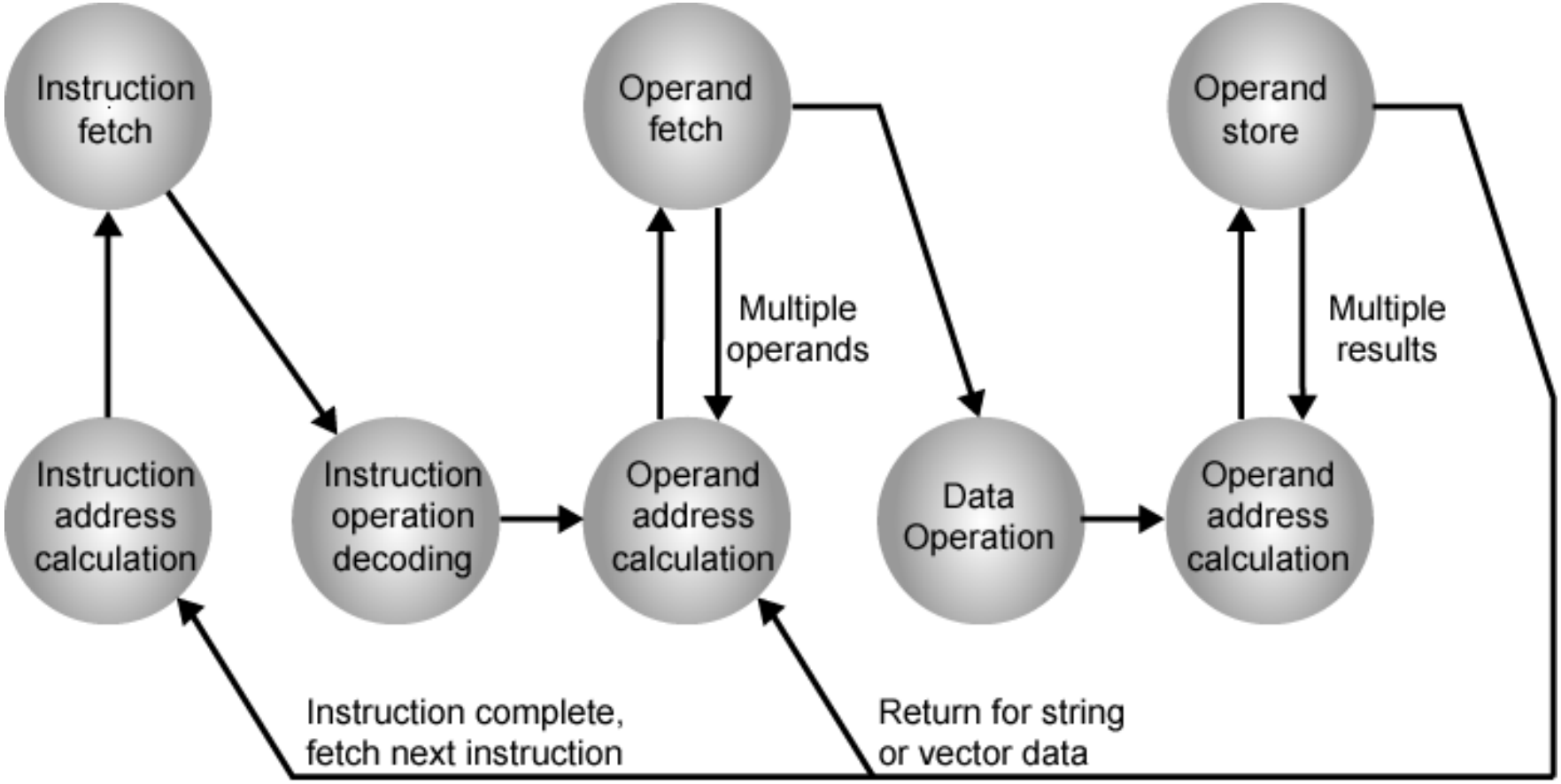
Operand address calculation (oac): If the operation involves reference to an operand in memory or available via I/O, then determine the address of the operand

Instruction Cycle State Diagram



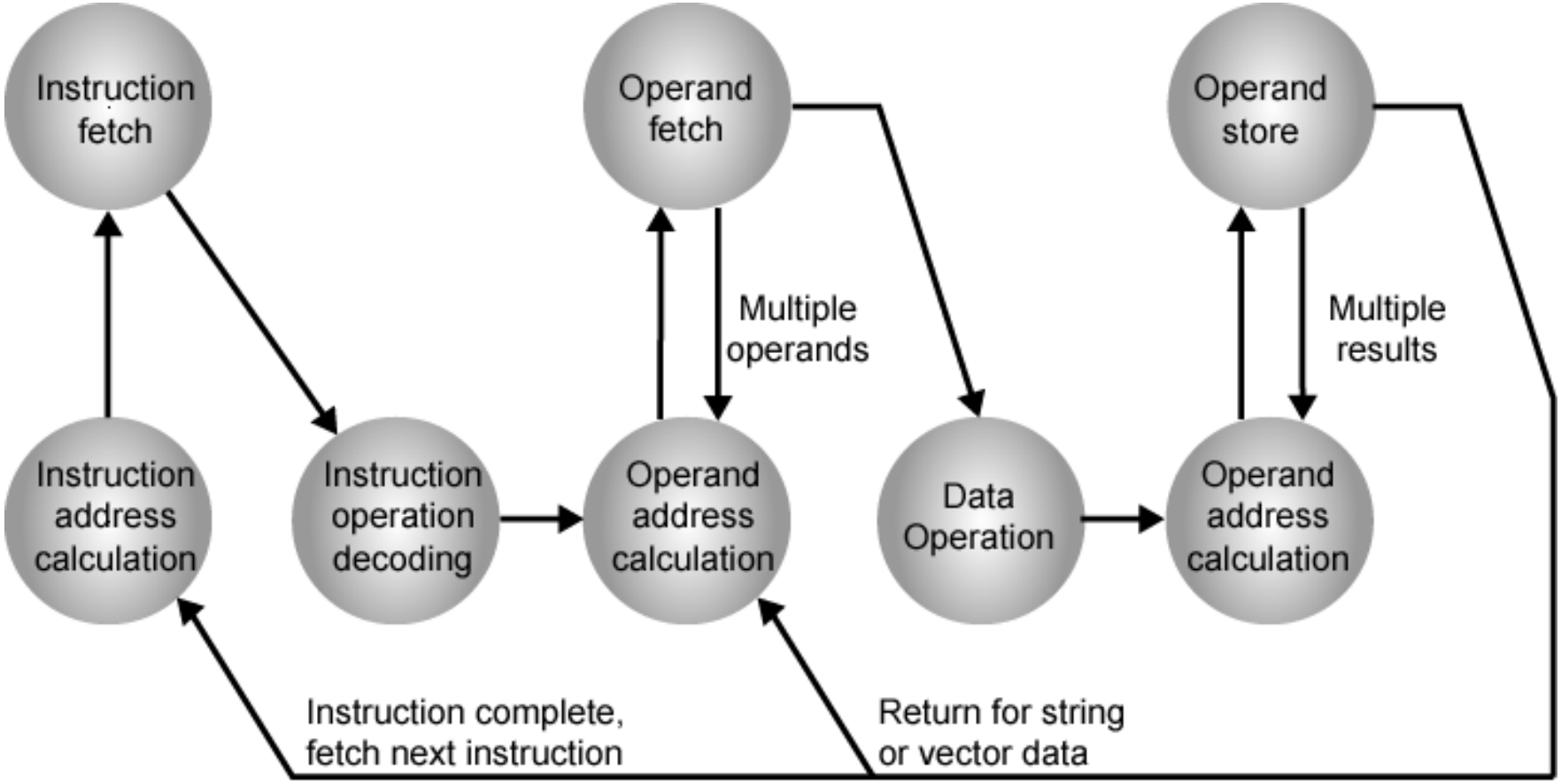
Operand fetch (of): Fetch the operand from memory or read it in from I/O

Instruction Cycle State Diagram



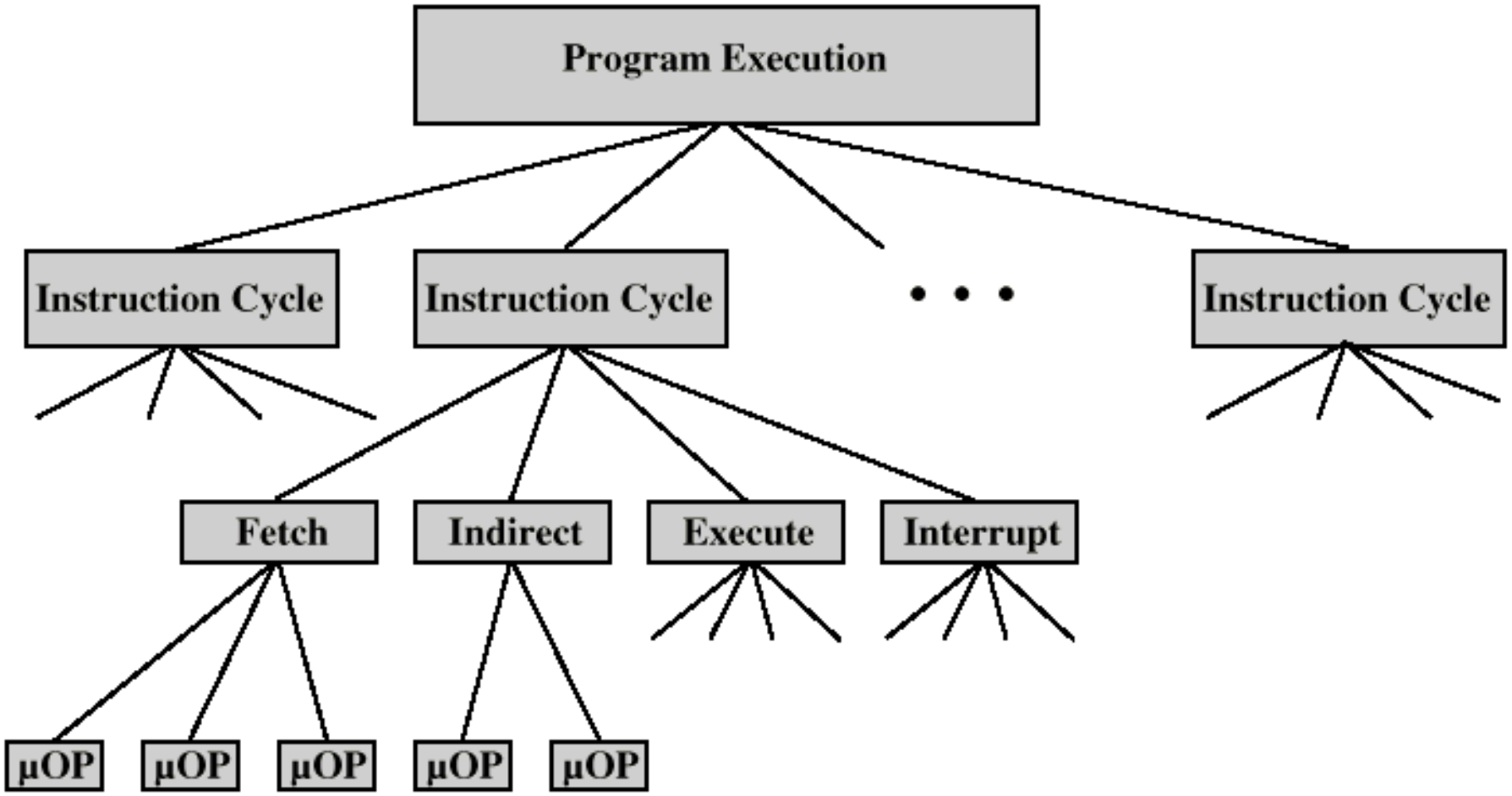
Data operation (do): Perform the operation indicated in the instruction

Instruction Cycle State Diagram



Operand store (os): Write the result into memory or out to I/O

Constituent Elements of Program Execution



The instruction cycle is decomposed into sequence of elementary *micro-operations*

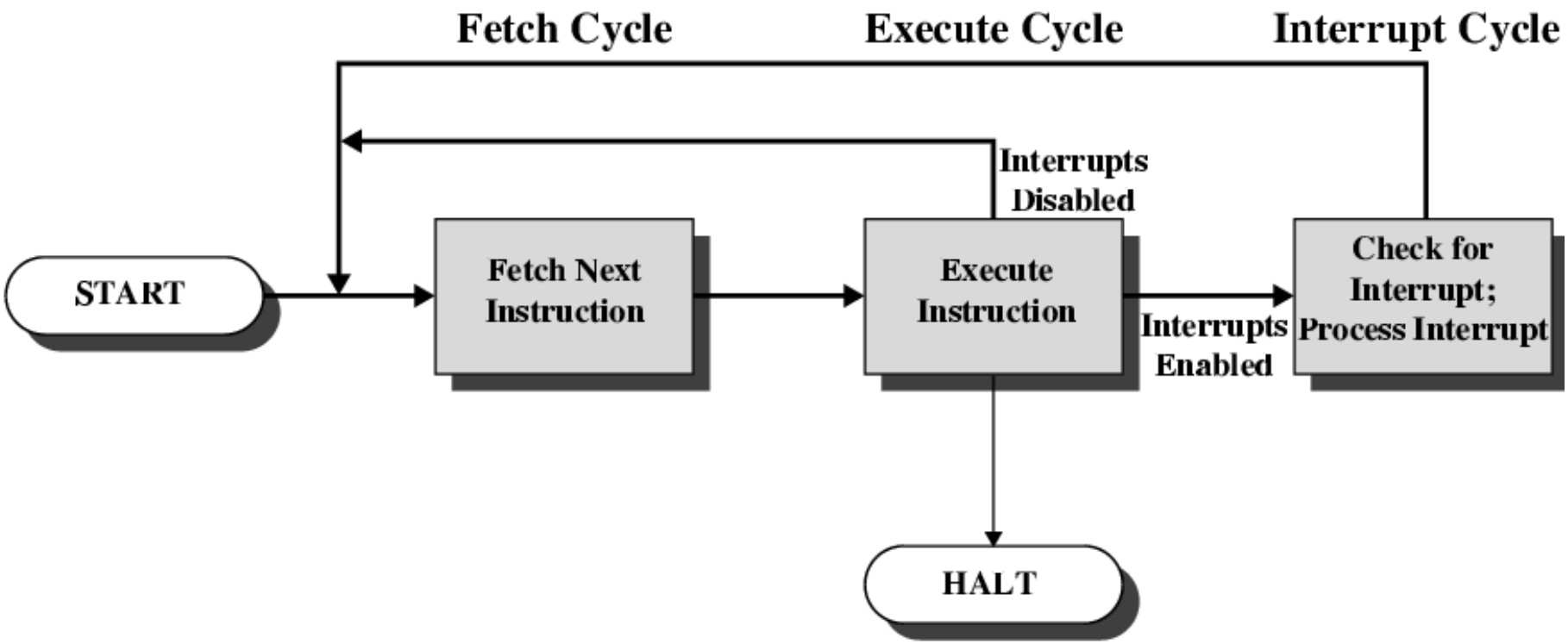
Interrupts

- **Interrupt** is the mechanism by which ***other modules 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. memory parity error

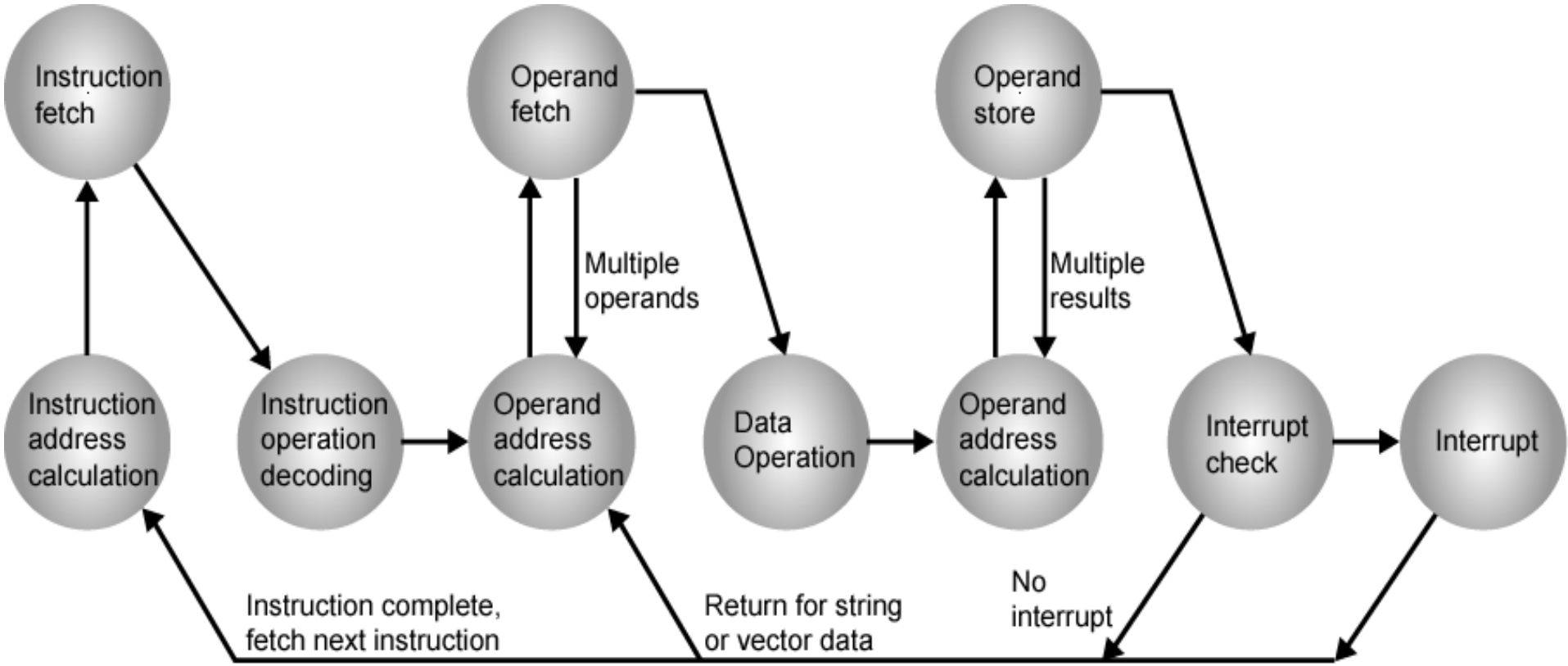
Interrupt Cycle

- **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

Instruction Cycle with Interrupts



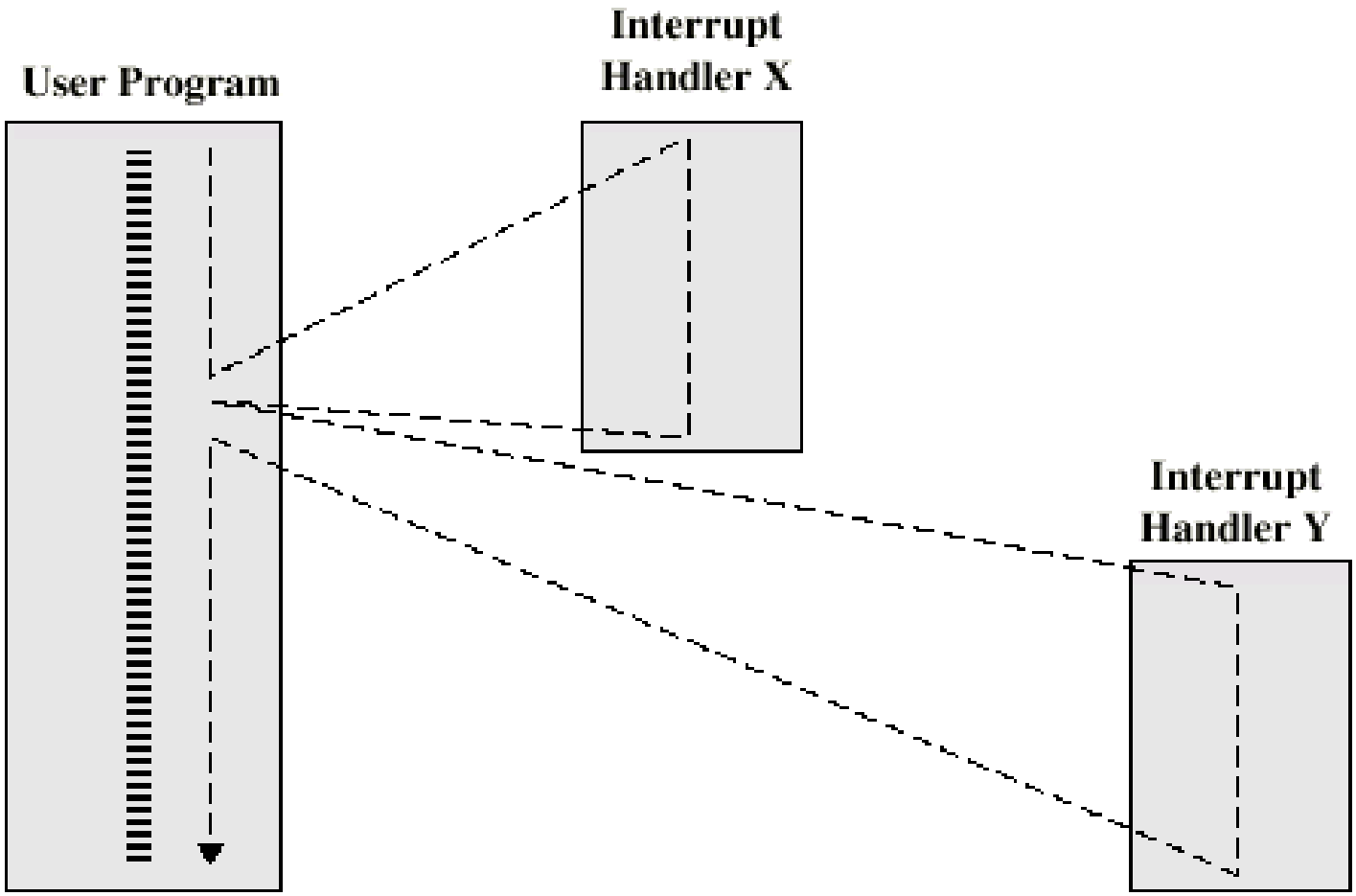
Instruction Cycle with Interrupts - State Diagram



Multiple Interrupts

- Disable interrupts – sequential interrupts
  - 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 – nested interrupts
  - 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