Microprocessor and Assembly Language CSC-321

Sheeza Zaheer

Lecturer

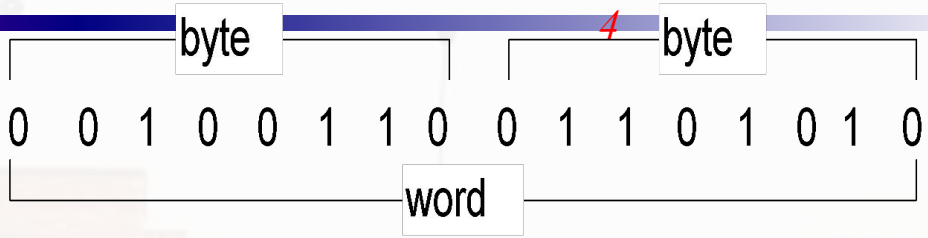
COMSATS UNIVERSITY ISLAMABAD

LAHORE CAMPUS

Memory & Processor Basics

MEMORY

Bits, Bytes and Double words



- *Each 1 or 0 is called a bit.*
- *Group of 4 bits = **N**ibble*
- *Group of 8 bits = **B**yte*
- *Group of 16 bits = **W**ord*
- *Group of 32 bits = **D**ouble words*

Common Prefixes

Prefix	Abbr.	Metric Meaning		CS Approximation	
Kilo	K	10^3	1,000	2^{10}	1,024
Mega	M	10^6	1,000,000	2^{20}	1,048,576
Giga	G	10^9	1,000,000,000	2^{30}	1,073,741,,824
Tera	T	10^{12}		2^{40}	
Peta	P	10^{15}		2^{50}	
Milli	m	10^{-3}	0.001	2^{-10}	0.0009765625
Micro	μ	10^{-6}	0.000001	2^{-20}	0.0000000954
Nano	n	10^{-9}	0.000000001	2^{-30}	
Pico	p	10^{-12}		2^{-40}	
Femto	f	10^{-15}		2^{-50}	

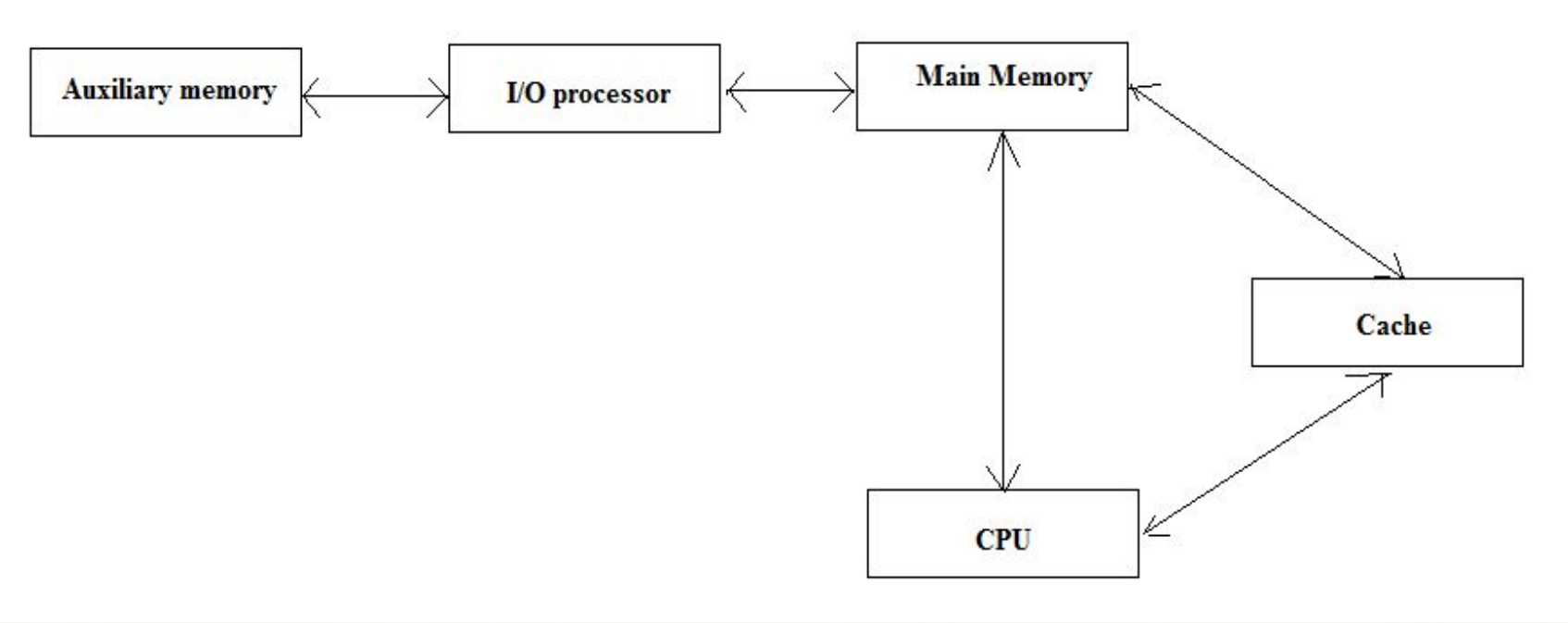
Memory

- Information processed by the computer is stored in its memory.
  - Program
  - Data
- Not all accumulated information is needed by the CPU at the same time
  - Therefore, it is more economical to use low-cost storage devices to serve as a backup for storing the information that is not currently used by CPU
- Memory Operations:
  - Read (Fetch contents of a location)
  - Write (Store data at a location)

Contd..

- The memory unit that directly communicate with CPU is called the *main memory*
- Devices that provide backup storage are called *auxiliary memory*
- The main memory occupies a central position by being able to communicate directly with the CPU and with auxiliary memory devices through an I/O processor
- A special very-high-speed memory called **cache** is used to increase the speed of processing by making current programs and data available to the CPU at a rapid rate

Contd..



Main Memory

- Most of the main memory in a general purpose computer is made up of RAM integrated circuits chips, but a portion of the memory may be constructed with ROM chips
- Memory Circuits:
  - RAM
    - Program Data and Instructions
    - Read and Write
  - ROM
    - ROM is used by the manufacturers to store system programs. Used for storing an initial program called *bootstrap loader,* which is required to start the computer software operating when power is turned off.
    - Only Read

Cache

- A special very-high-speed memory called **cache** is used to increase the speed of processing by making current programs and data available to the CPU at a rapid rate.

Memory Organization

- Memory is organized into a collection of bytes.
- Each byte is identified by a number – **Address**
  - Number of bits in an address depends on the processor
  - Example:- Intel 8086: 20-bit address, Intel 80286: 24-bit address
- Data stored in a memory byte – **Contents**

Contd..

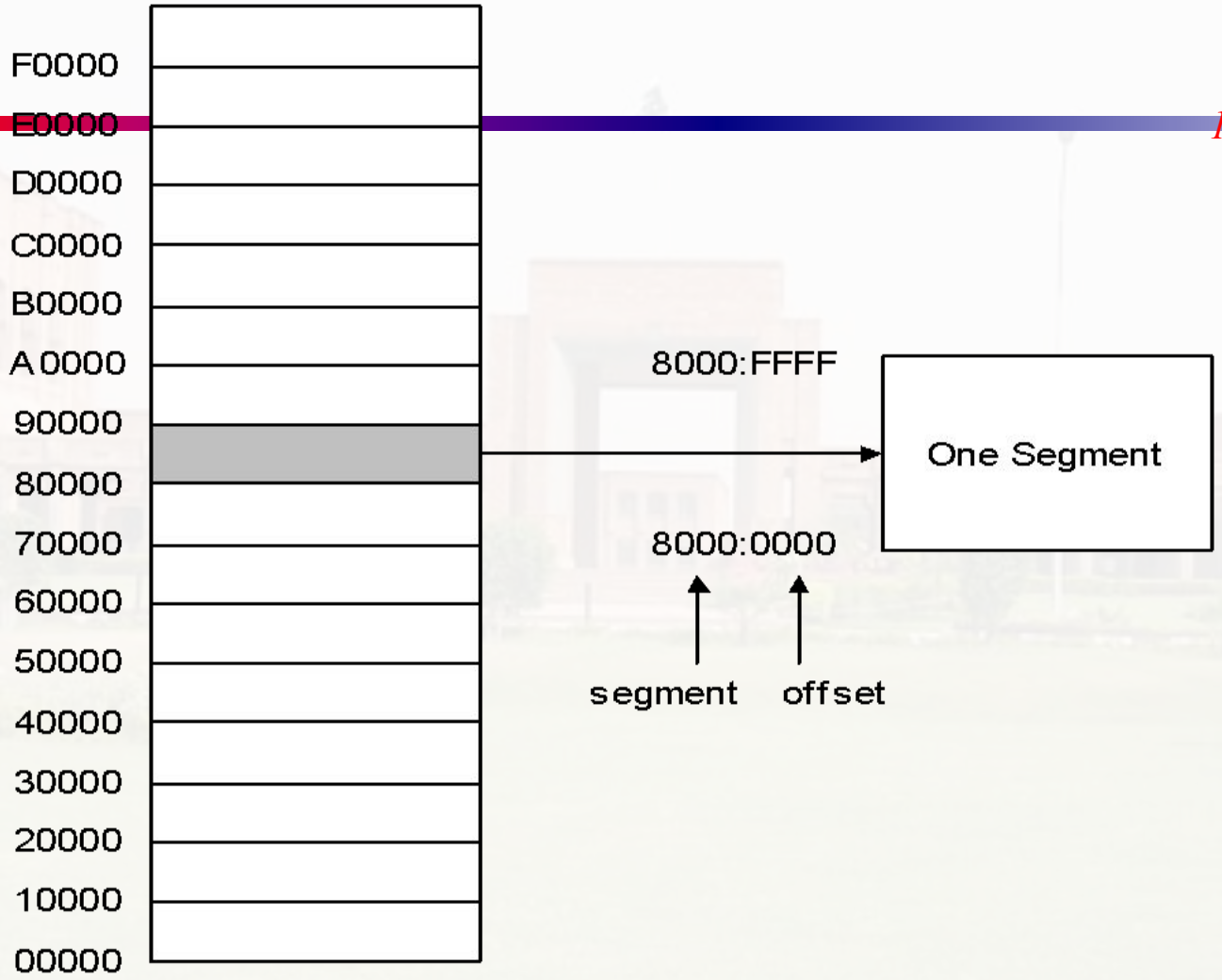
- Number of bits used in the address determines the number of bytes that can be accessed by the processor.

- **Example**: If processor uses 20-bit address, it can access 2^{20} = 1048576 bytes = 1 MB of memory

- **Question**: If processor uses 24-bit address, how many bytes of memory can be accessed?

Memory Segments

- A memory segment is a block of 2^{16} (or 64 K) consecutive memory bytes.
- Each segment has a number.
- Within a segment, memory location is specified by an offset. This is the number of bytes from the beginning of the segment.


F0000
E0000
D0000
C0000
B0000
A0000
90000
80000
70000
60000
50000
40000
30000
20000
10000
00000
8000:FFFF
One Segment
8000:0000
segment
offset

Segment : Offset Address

- **Logical Address = Segment : Offset**
  - 16-bit segment, 16-bit offset
- **Physical Address = Segment * 10h + Offset**
  - 20-bit address
- Example:

  Logical Address = A4FB:4872

  Physical Address = A4FB0h + 4872h = A9822h

Exercise

A memory location has a physical address 4A37Bh. Compute:

a. The offset address if the segment number is 40FFh.

b. The segment number if the offset address is 123Bh.

Program Segments

- A typical machine language program is loaded into following different memory segments:
  - Code Segment
  - Data Segment
  - Stack Segment
- Stack is a data structure used by processor to implement procedure calls.

BASIC OPERATIONAL CONCEPTS

Machine Instruction Elements

- Each instruction must have elements that contain the information required by the CPU for execution.
- These elements can be:
  - **Operation code**: Specifies the operation to be performed (e.g.. ADD, I/O). The operation is specified by a binary code, known as the operation code, or opcode.
  - **Source operand reference**: The operation may involve one or more source operands, that is, operands that are inputs for the operation.
  - **Result operand reference**: The operation may produce a result. Also called destination operand.
  - **Next instruction reference**: This tells the CPU where to fetch the next instruction.

***Reference**: http://cnx.org/content/m29425/latest/*

Instruction Representation

- Within the computer, each instruction is represented by a sequence of bits.
- 16 bits instruction
  - 4 bit opcode, 6 bit operand 1, 6 bit operand 2
  - 4 bit opcode, 12 bit operand
- 32 bits instruction
- 64 bits instruction

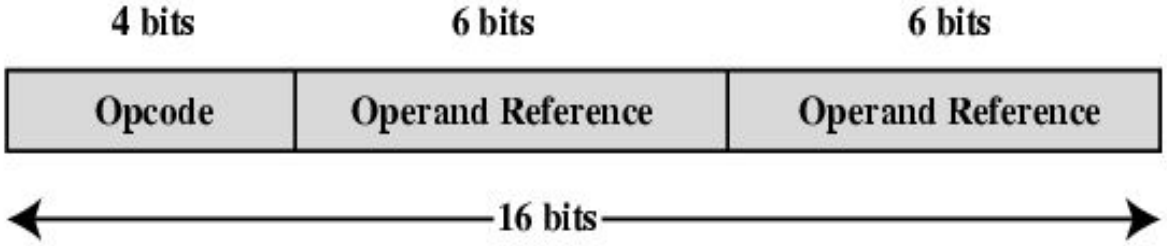
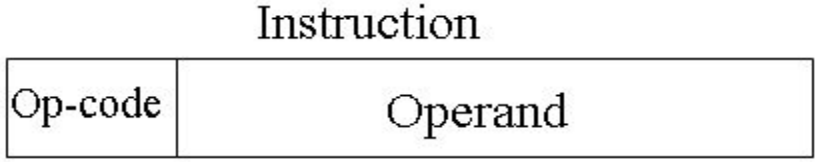


Figure: A Simple Instruction Format



Contd..

- Binary representations of machine instructions is difficult to remember.
- Use a symbolic representation of machine instructions.
- Opcodes are represented by abbreviations, called mnemonics, that indicate the operation. Common examples include:

ADD	Add
SUB	Subtract
MPY	Multiply
DIV	Divide
LOAD	Load data from memory
STOR	Store data to memory

Instruction Types

- **Data processing**: Arithmetic and logic instructions
- **Data storage**: Memory instructions
- **Data movement**: I/O instructions
- **Transfer of Control**: branch instructions

No. of Addresses in an Instruction

- Three addresses
  - Operand 1, operand 2, result
- Two addresses
  - Source
  - Destination
- One addresses
  - Source or Destination
- Zero address
  - Zero-address instructions are applicable to a special memory organization, called a Stack. A stack is a last-in-first-out set of locations.

Types of Operands

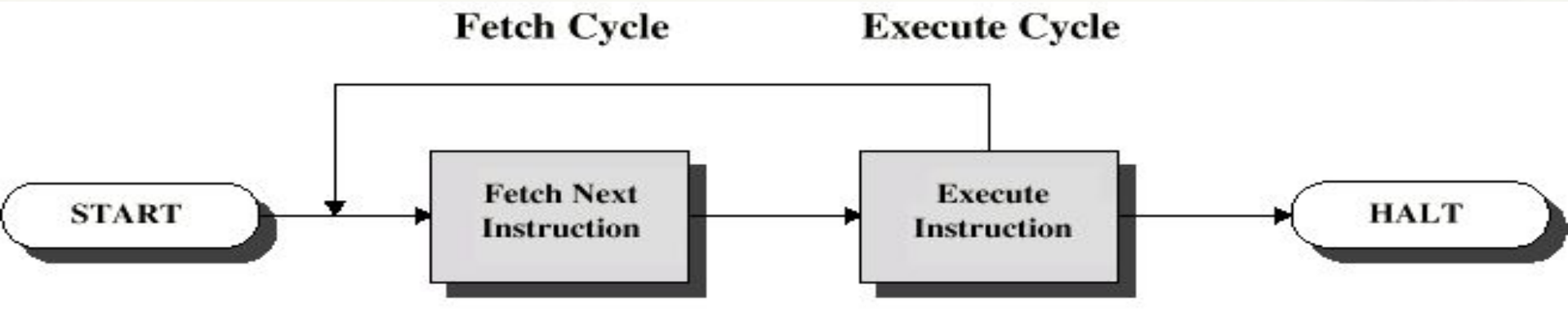
- Machine instructions operate on data.
- The most important general categories of data are:
  - Addresses
  - Numbers
  - Characters

Basic Operations – Processor

- Execute the software by fetching instruction from memory
- Look for any external signal and react accordingly
  - Input signals from keyboard or mouse etc.

Basic Instruction Cycle

- Fetch □ Decode □ Execute
- Fetch
  1. Fetch an instruction from memory
  2. Decode the instruction to determine the operation
  3. Fetch data from memory if necessary
- Execute
  4. Perform the operation on the data
  5. Store the result in memory if needed



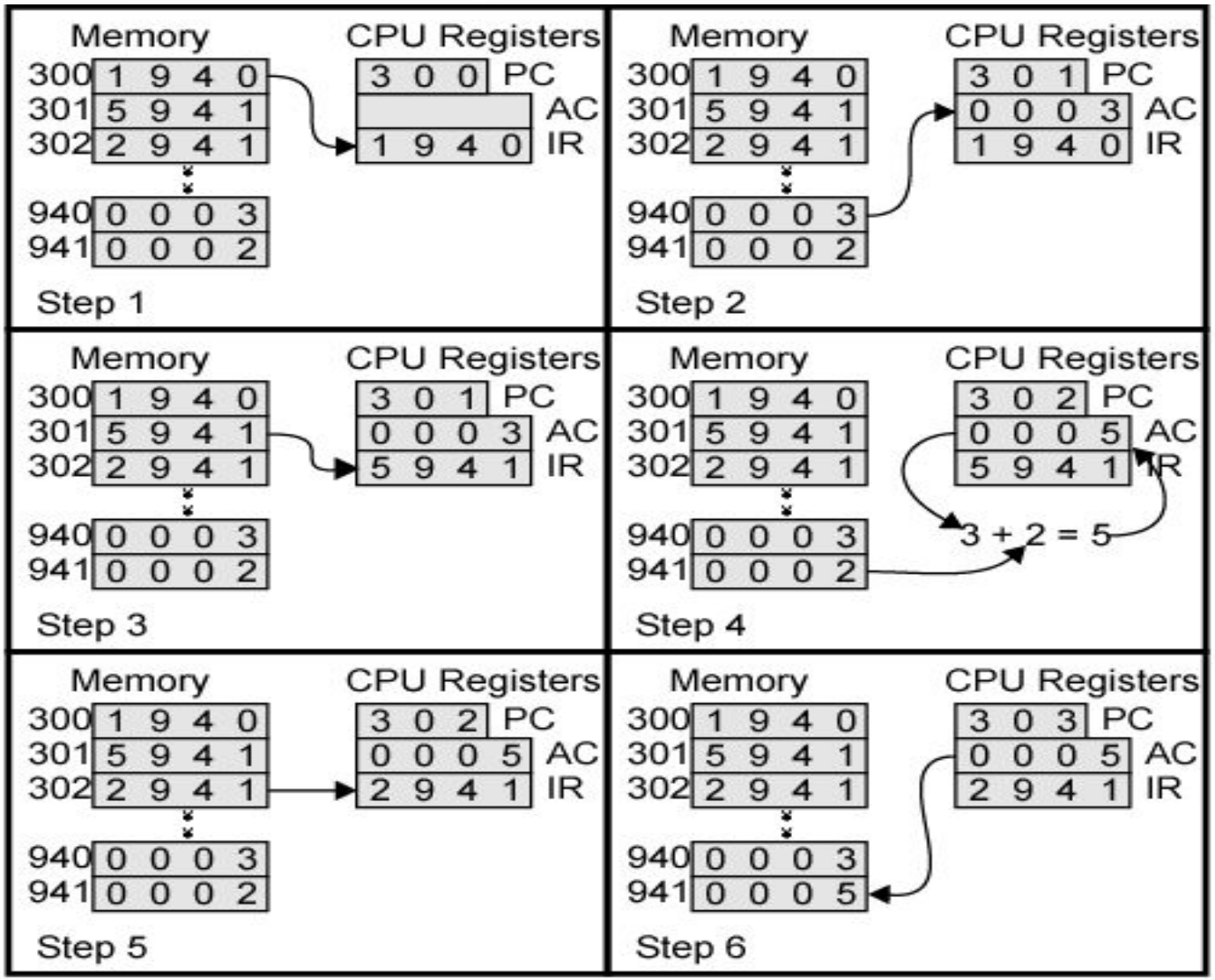
Contd..

- Internal CPU Registers used in instruction cycle:
  - Program Counter (PC) = Address of instruction
  - Instruction Register (IR) = Instruction being executed
  - Accumulator (AC) = Temporary Storage

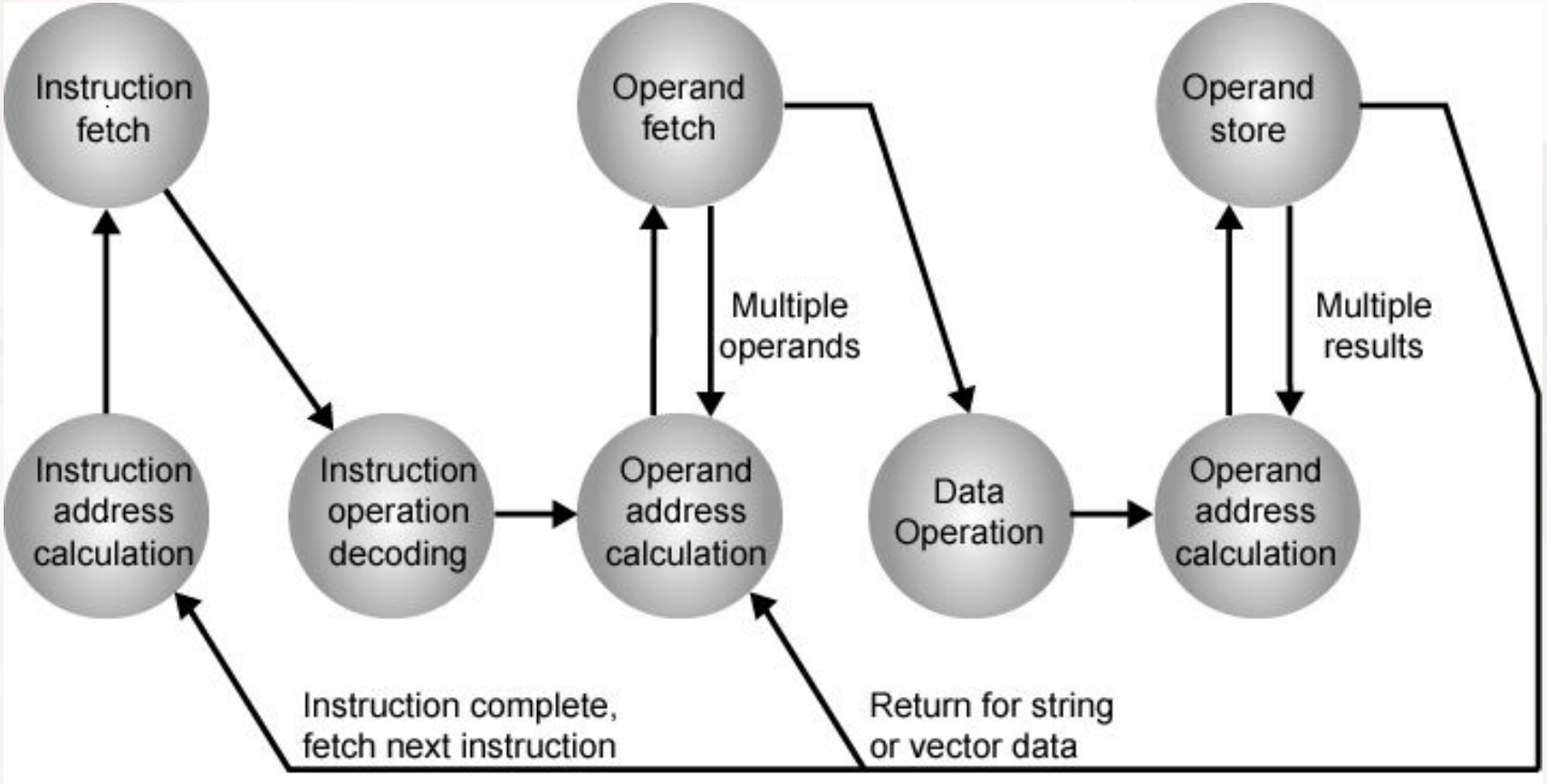
Detailed Steps

- Address in the Program Counter register
  - Program Counter (PC) holds address of next instruction to fetch
- Fetch the instruction from the memory
- Increment the Program Counter
  - Unless told otherwise
- Instruction loaded into Instruction Register (IR)
- Decode the type of instruction
- Fetch the operands
- Execute the instruction
- Store the results

Example Program Execution



Instruction Cycle State Diagram



Contd..

- Instruction Fetch
  - Read instruction from memory into processor
- Instruction Operation Decoding
  - Determine the type of operation to be performed and operand(s) to be used.
- Operand Address Calculation
  - If operation involves reference to an operand in memory or I/O, then determine the address of operand.
- Operand Fetch
  - Fetch from memory or read from I/O
- Data Operation
  - Perform the operation
- Operand Store
  - Write into memory or out to I/O if required