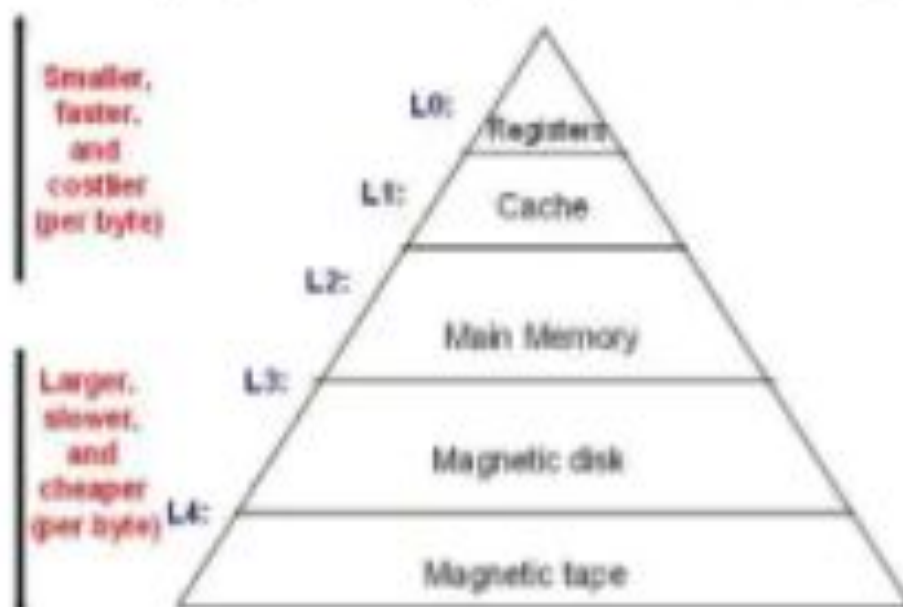


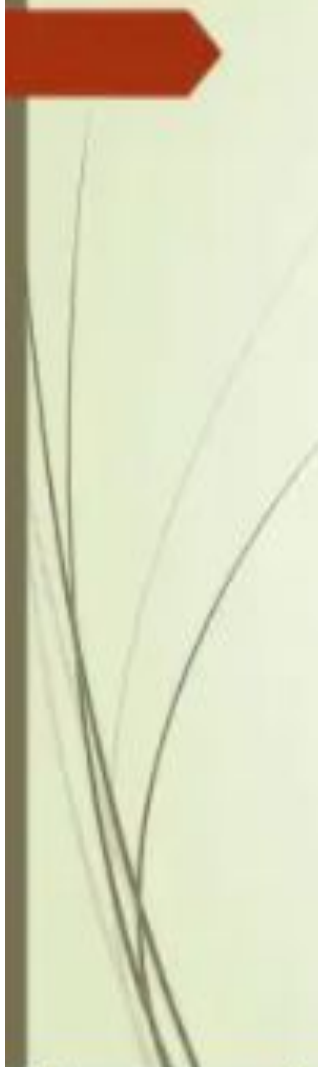



# Register Organization

## REGISTER ORGANIZATION

- A register is a very small amount of very fast memory that is built into the CPU (central processing unit) in order to speed up its operations by providing quick access to commonly used values.
- Registers are normally measured by the number of bits they can hold, for example, an 8-bit register or a 32-bit register.



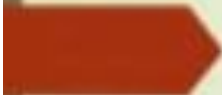
- 
- The register in the processor perform two roles:
    1. **User-visible register:** Enable the machine- or assembly language programmer to minimize main memory references by optimizing use of registers.
    2. **Control and status registers:** Used by the control unit to control the operation of the processor and by privileged, operating system programs to control the execution of programs.



# User-visible Registers

## CATEGORIES:-

- General Purpose
- Data
- Address
- Condition Codes



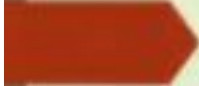
# General Purpose

- **General Purpose Registers** can be assigned to a variety of functions by the programmer
- Mostly these registers contain the operand for any opcode.
- In some cases these are used for addressing purpose.




# Data Registers

- **Data Register** to hold data and cannot be employed in the calculation of an operand address
- Eg. Accumulator.



# Address Registers

- **Address Register** they may be devoted to a particular addressing mode
  - **Segment pointers** :a segment register holds the address of the base of the segment
  - **Index registers** :are used for indexed addressing and may be autoindexed.
  - **Stack Pointer**: If there is user-visible stack addressing, then typically there is a dedicated register that points to the top of the stack.



## Condition Codes

- **Condition codes** are bits set by the processor hardware as the result of operations.
- Condition codes are bits set by the processor hardware as the result of operation.





# CONTROL AND STATUS REGISTERS

## Four Essential Registers:

- **Program counter (PC):** Contains the address of an instruction to be fetched.
- **Instruction register (IR):** Contains the instruction most recently fetched.
- **Memory address register (MAR):** Contains the address of a location in memory.
- **Memory buffer register (MBR):** Contains a word of data to be written to memory or the word most recently read.

# Program Status Word

- **Program status word (PSW)** contain status information.
- The PSW typically contains condition codes plus other status information.
  - **Sign:** Contains the sign bit of the result of the last arithmetic operation.
  - **Zero:** Set when the result is 0.
  - **Carry:** Set if an operation resulted in a carry (addition) into or borrow (subtraction) out of a high-order bit. Used for multiword arithmetic operations.
  - **Equal:** Set if a logical compare result is equality.
  - **Overflow:** Used to indicate arithmetic overflow.
  - **Interrupt Enable/Disable:** Used to enable or disable interrupts.
  - **Supervisor:** Indicates whether the processor is executing in supervisor or user mode. Certain privileged instructions can be executed only in supervisor mode, and certain areas of memory can be accessed only in supervisor mode.

# Instruction Format

## Introduction

- ❖ An instruction format or instruction code is a group of bits used to perform a particular operation on the data stored in computer
- ❖ Processor fetches an instruction from memory and decodes the bits to execute the instruction.
- ❖ Different computer may have their own instruction set .
- ❖ The operation of the processor is determined by the instructions it executes, referred to as machine instructions or computer instructions
- ❖ The collection of different instructions that the processor can execute is referred to as the processor's instruction set
- ❖ Each instruction must contain the information required by the processor for execution

# Elements of a Machine Instruction

## ❖ Operation code (opcode)

- ❖ Specifies the operation to be performed. The operation is specified by a binary code, known as the operation code, opcode do ADD, SUB, DIV, LOAD, ...

## ❖ 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

## ❖ Next instruction reference

- ❖ This tells the processor where to fetch the next instruction after the execution of this instruction is complete



# Instruction Cycle State Diagram

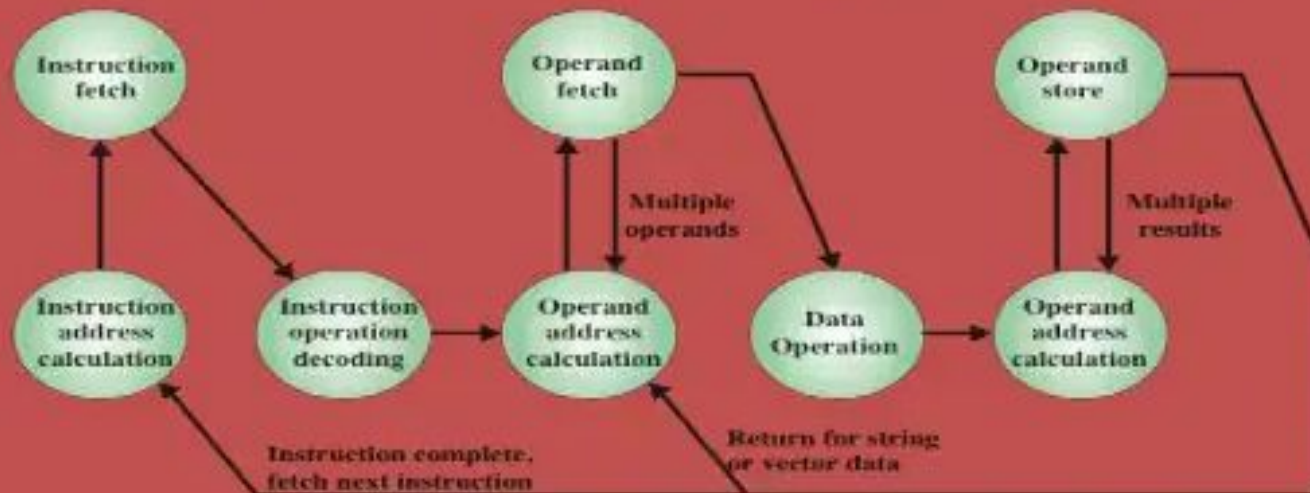


Figure 12.1 Instruction Cycle State Diagram

# Source and result operands can be in one of four areas:

## 1) Main memory or virtual memory

- ❖ As with next instruction references, the main or virtual memory address must be supplied.

## 2) I/O devices

- ❖ The instruction must specify the I/O module and device for the operation. If memory-mapped I/O is used, this is just another main or virtual memory address.

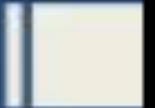
## 3) Processor register

- ❖ A processor contains one or more registers that may be referenced by machine instructions.
- ❖ If more than one register exists each register is assigned a unique name or number and the instruction must contain the number of the desired register

## 4) Immediate

- ❖ The value of the operand is contained in a field in the instruction being executed

# Types of Operand



- ❖ Addresses: immediate, direct, indirect, stack
- ❖ Numbers: integer or fixed point (binary, two's complement), floating point (sign, exponent), (packed) decimal  
(246 = 0000 0010 0100 0110)
- ❖ Characters: ASCII (128 printable and control characters + bit for error detection)
- ❖ Logical Data: bits or flags, e.g., Boolean 0 and 1

# Instruction Representation

- ❖ Within the computer each instruction is represented by a sequence of bits
- ❖ The instruction is divided into fields, corresponding to the constituent elements of the instruction

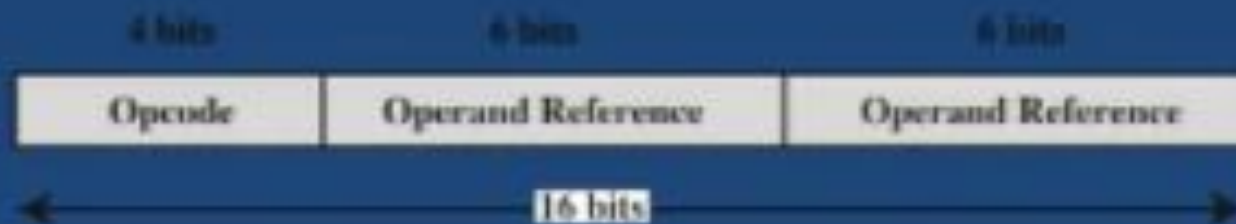


Figure 10.2 A Simple Instruction Format

Figure 12.2 A Simple Instruction Format



# Instruction Types

- A computer should have a set of instructions that allows the user to formulate any data processing task.. Any program written in a high-level language must be translated into machine language to be executed, so we can categorize instruction types as follows:
- ❖ Data processing: Arithmetic instructions provide computational capabilities for processing numeric data
- ❖ Data storage: Movement of data into or out of register and or memory locations
- ❖ Data movement :I /O instructions are needed to transfer programs and data into memory and the results of computations back out to the user
- ❖ Control: test instruction test the value of a data word or the status of a computation
- ❖ Branch instruction used to branch to a different set of instructions depending on the decision made

# Instruction Formats



- ❖ Layout of bits in an instruction
- ❖ Includes opcode
- ❖ Includes (implicit or explicit) operand(s)
- ❖ Usually more than one instruction format in an instruction set

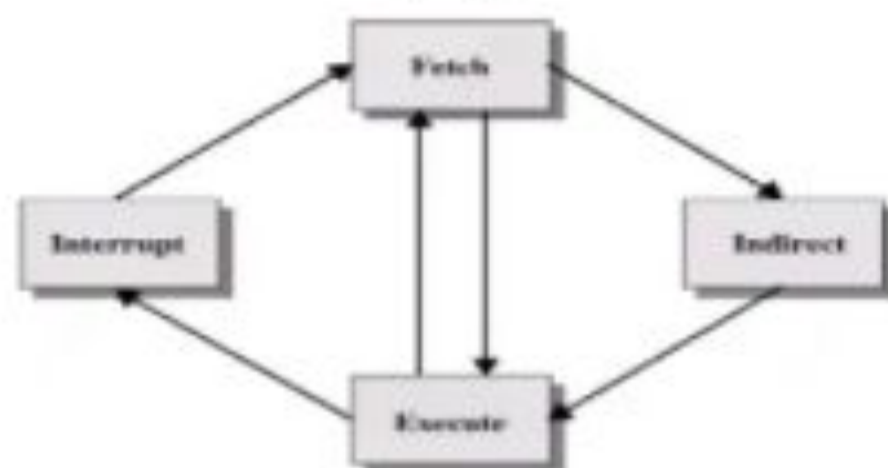
## Instruction Length

Affected by and affects:

- ❖ Memory size
- ❖ Memory organization - addressing
- ❖ Bus structure, e.g., width
- ❖ CPU complexity
- ❖ CPU speed
- ❖ Trade off between powerful instruction repertoire and saving space

# INSTRUCTION CYCLE:

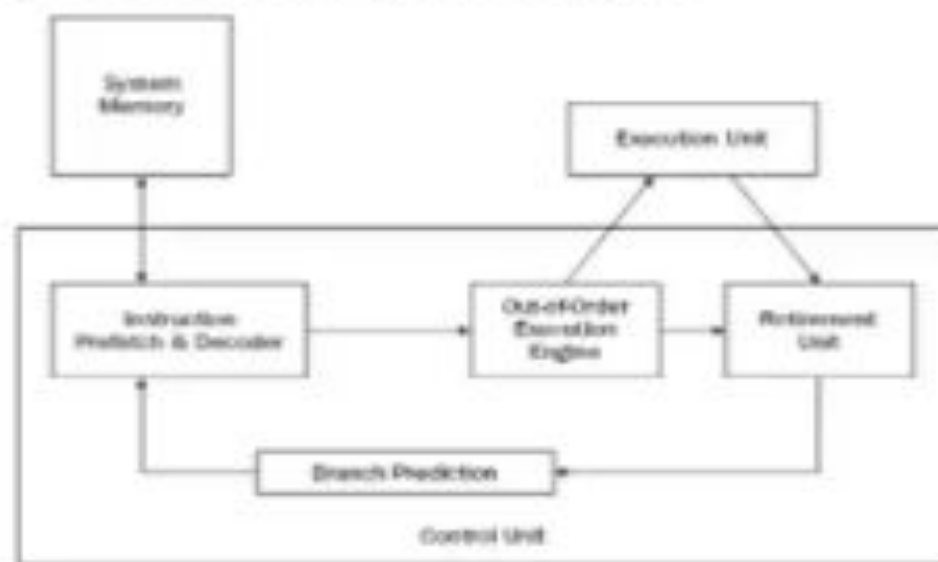
- An instruction cycle' (also called fetch-and-execute cycle, fetch-decode-execute cycle, and FDX) is the time period during which a computer processes a machine language instruction from its memory or the sequence of actions that the central processing unit (CPU) performs to execute each machine code instruction in a program.



- The name fetch-and-execute cycle is commonly used. The instruction must be fetched from main memory, and then executed by the CPU. This is fundamentally how a computer operates, with its CPU reading and executing a series of instructions written in its machine language. From this arise all functions of a computer familiar from the user's end.
- There are typically four stages of an instruction cycle that the CPU carries out:
  - 1) Fetching the Instruction.
  - 2) Decode the Instruction.
  - 3) "Read the effective address".
  - 4) Execute the Instruction.



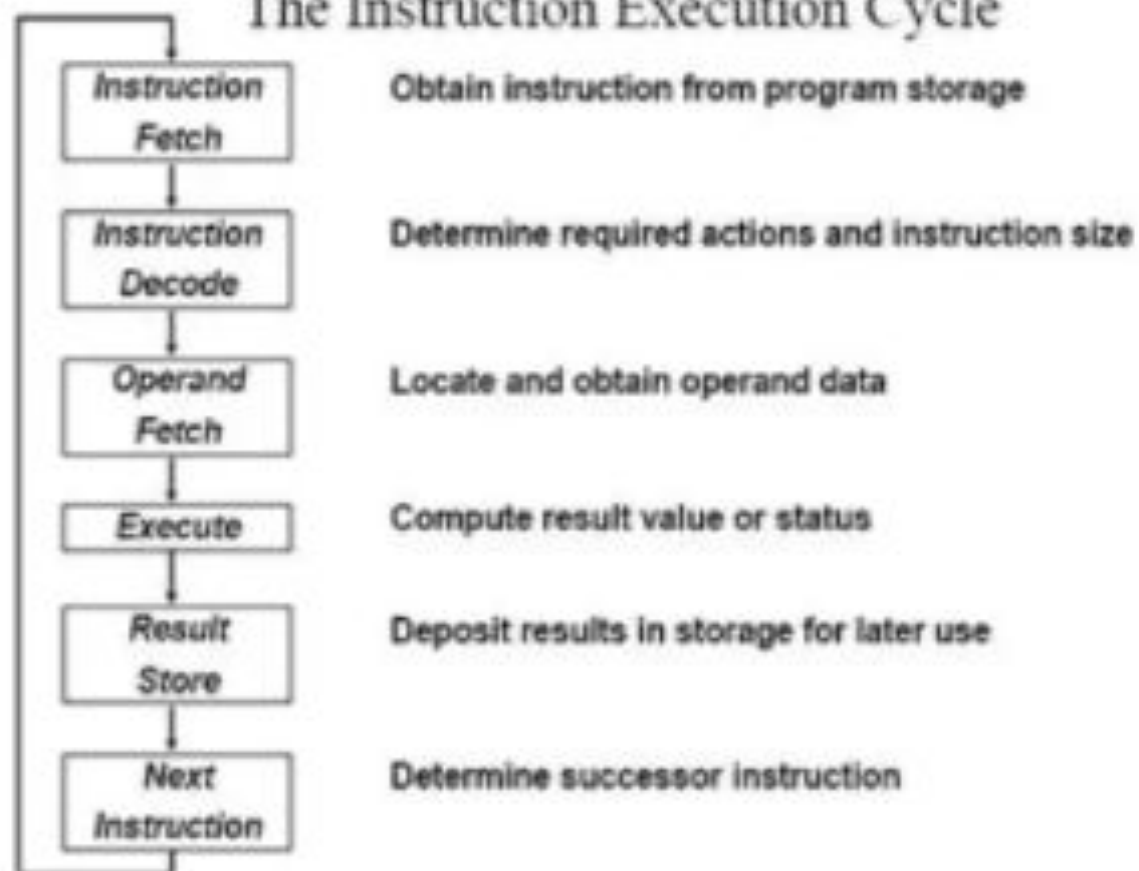
- The Cycle is Then repeated Again



- The MAR (Memory Address Register) holds the address of the location to or from which data are to be transferred. As can be seen from the figure above, the connection of the MAR to the main memory is one-way or unidirectional.
- The MDR (Memory Data Register) contains the data to be written or read out of the addressed location.



## The Instruction Execution Cycle



## OPERATING STEPS:

- 1) PC is set to point to the first instruction of the program (the operating system loads the memory address of the first instruction).
- 2) The contents of the PC are transferred to the MAR (which is automatically transmitted to the MM) and a Read signal is sent to the MM (Main Memory).
- 3) The addressed word is read out of MM and loaded into the MDR.
- 4) The contents of MDR are transferred to the IR. The instruction is ready to be decoded and executed.
- 5) During execution, the contents of the PC are incremented or updated to point to the next instruction.

Example:

Enumerate the different steps needed to execute the machine instruction

