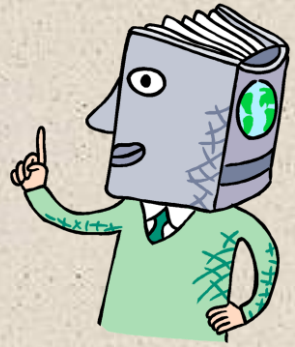
William Stallings Computer Organization and Architecture 10th Edition

Chapter 12

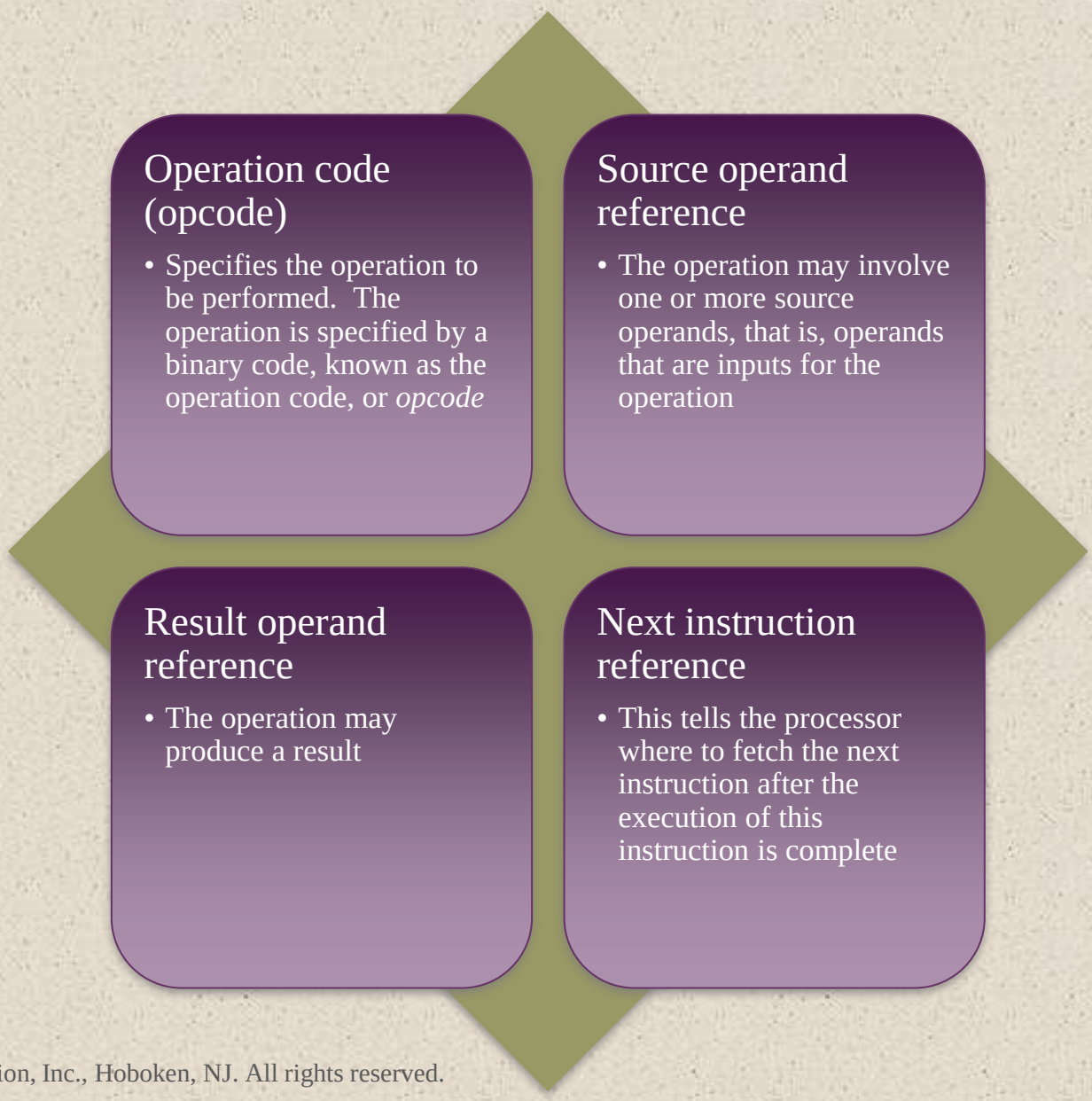
Instruction Sets: Characteristics and Functions

Machine Instruction Characteristics

- 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



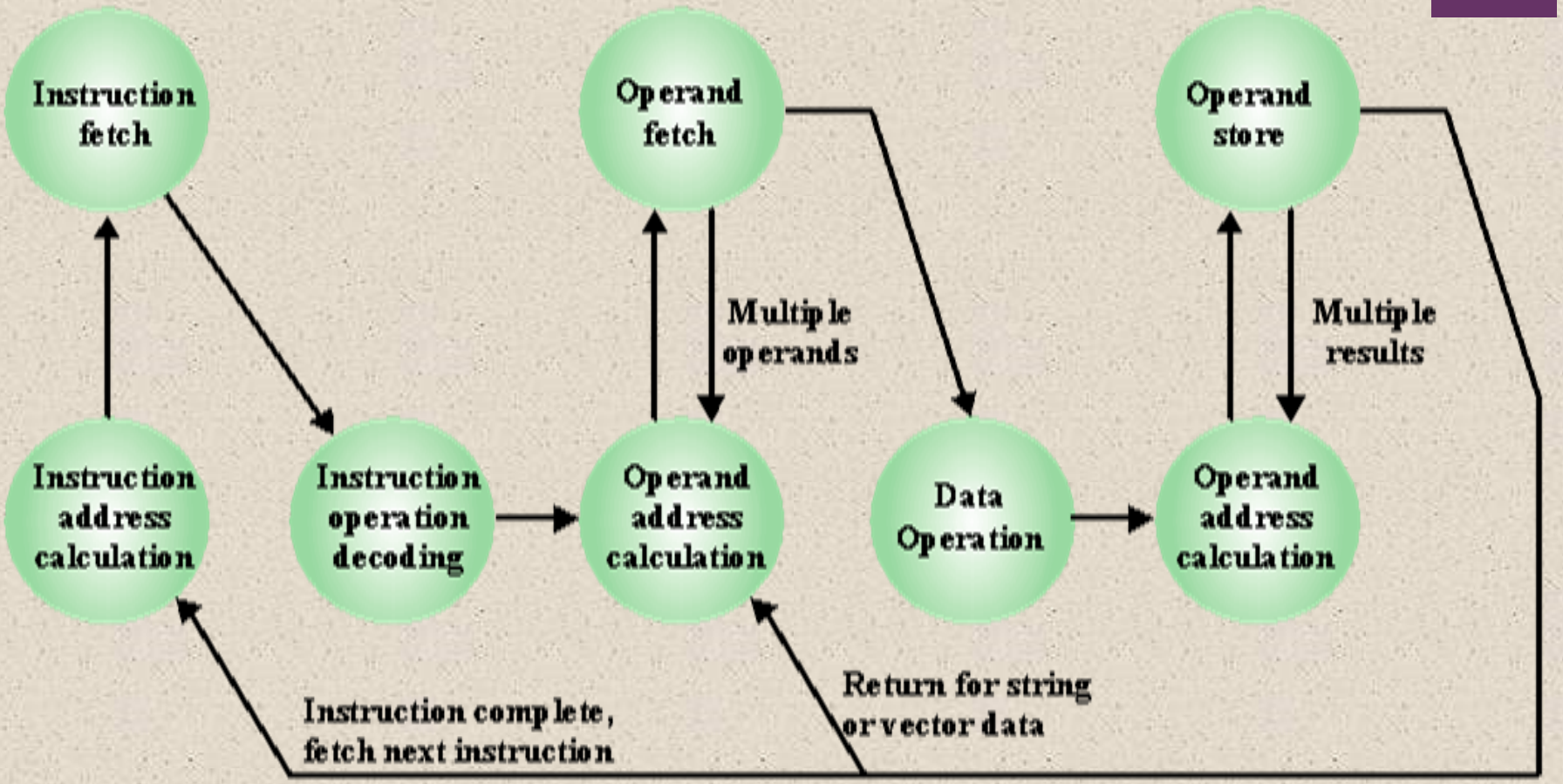


Figure 12.1 Instruction Cycle State Diagram

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

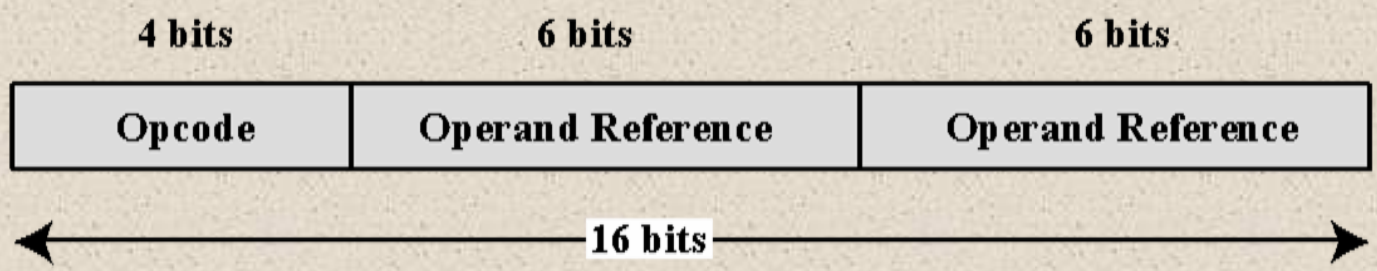


Figure 12.2 A Simple Instruction Format

Instruction Representation

- Opcodes are represented by abbreviations called *mnemonics*
- Examples include:
  - ADD Add
  - SUB Subtract
  - MUL Multiply
  - DIV Divide
  - LOAD Load data from memory
  - STOR Store data to memory
- Operands are also represented symbolically
- Each symbolic opcode has a fixed binary representation
  - The programmer specifies the location of each symbolic operand

Instruction		Comment
SUB	Y, A, B	$Y \leftarrow A - B$
MPY	T, D, E	$T \leftarrow D \times E$
ADD	T, T, C	$T \leftarrow T + C$
DIV	Y, Y, T	$Y \leftarrow Y \div T$

(a) Three-address instructions

Instruction		Comment
MOVE	Y, A	$Y \leftarrow A$
SUB	Y, B	$Y \leftarrow Y - B$
MOVE	T, D	$T \leftarrow D$
MPY	T, E	$T \leftarrow T \times E$
ADD	T, C	$T \leftarrow T + C$
DIV	Y, T	$Y \leftarrow Y \div T$

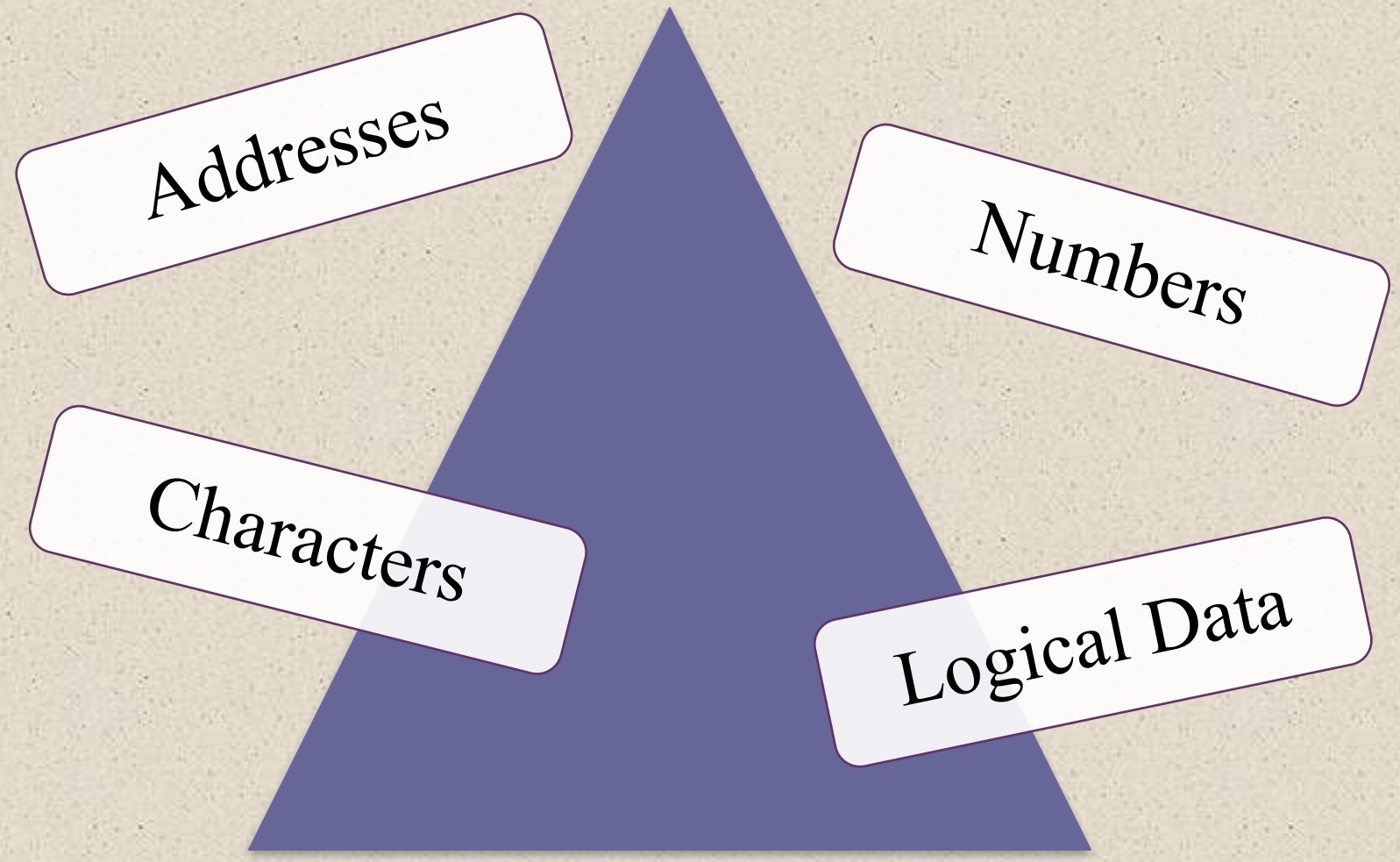
(b) Two-address instructions

Instruction		Comment
LOAD	D	$AC \leftarrow D$
MPY	E	$AC \leftarrow AC \times E$
ADD	C	$AC \leftarrow AC + C$
STOR	Y	$Y \leftarrow AC$
LOAD	A	$AC \leftarrow A$
SUB	B	$AC \leftarrow AC - B$
DIV	Y	$AC \leftarrow AC \div Y$
STOR	Y	$Y \leftarrow AC$

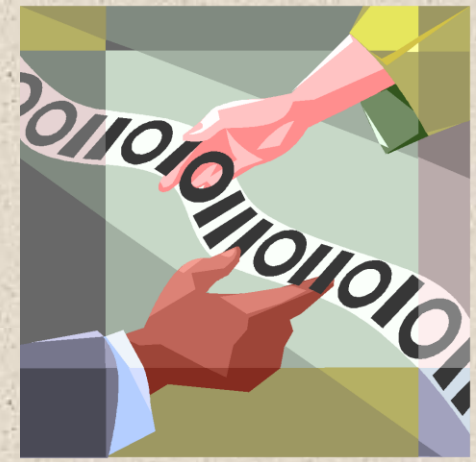
(c) One-address instructions

Figure 12.3 Programs to Execute $Y = \dfrac{A - B}{C + (D \times E)}$

Types of Operands



Numbers



- All machine languages include numeric data types
- Numbers stored in a computer are limited:
  - Limit to the magnitude of numbers representable on a machine
  - In the case of floating-point numbers, a limit to their precision
- Three types of numerical data are common in computers:
  - Binary integer or binary fixed point
  - Binary floating point
  - Decimal
- Packed decimal
  - Each decimal digit is represented by a 4-bit code with two digits stored per byte

Characters

- A common form of data is text or character strings
- Textual data in character form cannot be easily stored or transmitted by data processing and communications systems because they are designed for binary data
- Most commonly used character code is the International Reference Alphabet (IRA)
  - Referred to in the United States as the American Standard Code for Information Interchange (ASCII)
- Another code used to encode characters is the Extended Binary Coded Decimal Interchange Code (EBCDIC)
  - EBCDIC is used on IBM mainframes

Logical Data

- An *n*-bit unit consisting of *n* 1-bit items of data, each item having the value 0 or 1
- Two advantages to bit-oriented view:
  - Memory can be used most efficiently for storing an array of Boolean or binary data items in which each item can take on only the values 1 (true) and 0 (false)
  - To manipulate the bits of a data item
    - If floating-point operations are implemented in software, we need to be able to shift significant bits in some operations

Single-Instruction-Multiple-Data (SIMD) Data Types

- Introduced to the x86 architecture as part of the extensions of the instruction set to optimize performance of multimedia applications
- These extensions include MMX (multimedia extensions) and SSE (streaming SIMD extensions)
- SIMD (Single Instruction, Multiple Data) is a parallel processing technique used in CPUs to perform the same operation on multiple data points simultaneously.
- Without SIMD (normal addition, one by one):

  a1 = b1 + c1

  a2 = b2 + c2

  a3 = b3 + c3

  a4 = b4 + c4

- With SIMD (single instruction for all four):SIMD_ADD(a1-a4, b1-b4, c1-c4)

Type	Operation Name	Description
Data Transfer	Move (transfer)	Transfer word or block from source to destination
	Store	Transfer word from processor to memory
	Load (fetch)	Transfer word from memory to processor
	Exchange	Swap contents of source and destination
	Clear (reset)	Transfer word of 0s to destination
	Set	Transfer word of 1s to destination
	Push	Transfer word from source to top of stack
	Pop	Transfer word from top of stack to destination
Arithmetic	Add	Compute sum of two operands
	Subtract	Compute difference of two operands
	Multiply	Compute product of two operands
	Divide	Compute quotient of two operands
	Absolute	Replace operand by its absolute value
	Negate	Change sign of operand
	Increment	Add 1 to operand
	Decrement	Subtract 1 from operand
Logical	AND	Perform logical AND
	OR	Perform logical OR
	NOT (complement)	Perform logical NOT
	Exclusive-OR	Perform logical XOR
	Test	Test specified condition; set flag(s) based on outcome
	Compare	Make logical or arithmetic comparison of two or more operands; set flag(s) based on outcome
	Set Control Variables	Class of instructions to set controls for protection purposes, interrupt handling, timer control, etc.
	Shift	Left (right) shift operand, introducing constants at end
	Rotate	Left (right) shift operand, with wraparound end

Table 12.3

Common Instruction Set Operations (page 1 of 2)

(Table can be found on page 426 in textbook.)

Type	Operation Name	Description
Transfer of Control	Jump (branch)	Unconditional transfer; load PC with specified address
	Jump Conditional	Test specified condition; either load PC with specified address or do nothing, based on condition
	Jump to Subroutine	Place current program control information in known location; jump to specified address
	Return	Replace contents of PC and other register from known location
	Execute	Fetch operand from specified location and execute as instruction; do not modify PC
	Skip	Increment PC to skip next instruction
	Skip Conditional	Test specified condition; either skip or do nothing based on condition
	Halt	Stop program execution
	Wait (hold)	Stop program execution; test specified condition repeatedly; resume execution when condition is satisfied
	No operation	No operation is performed, but program execution is continued
Input/Output	Input (read)	Transfer data from specified I/O port or device to destination (e.g., main memory or processor register)
	Output (write)	Transfer data from specified source to I/O port or device
	Start I/O	Transfer instructions to I/O processor to initiate I/O operation
	Test I/O	Transfer status information from I/O system to specified destination
Conversion	Translate	Translate values in a section of memory based on a table of correspondences
	Convert	Convert the contents of a word from one form to another (e.g., packed decimal to binary)

Table 12.3

Common Instruction Set Operations (page 2 of 2)

(Table can be found on page 426 in textbook.)

Table 12.4
Processor Actions for Various Types of Operations

Data Transfer	Transfer data from one location to another
	If memory is involved: Determine memory address Perform virtual-to-actual-memory address transformation Check cache Initiate memory read/write
Arithmetic	May involve data transfer, before and/or after
	Perform function in ALU
	Set condition codes and flags
Logical	Same as arithmetic
Conversion	Similar to arithmetic and logical. May involve special logic to perform conversion
Transfer of Control	Update program counter. For subroutine call/return, manage parameter passing and linkage
I/O	Issue command to I/O module
	If memory-mapped I/O, determine memory-mapped address

(Table can be found on page 427 in textbook.)

Instructions Set

Instruction Set:

A set of all instructions that a CPU can perform is called instruction set.

- Different types of CPU can execute different types of instruction sets.
- Modern CPU can execute 80 to 120 instructions.

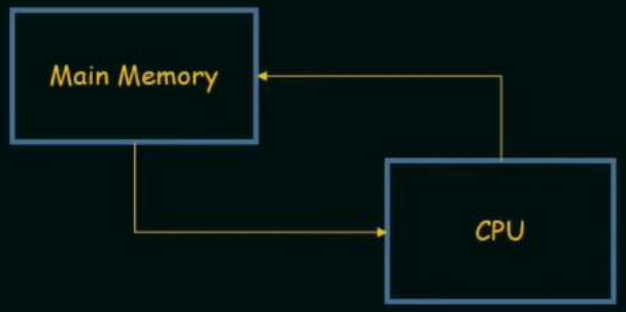
Types of Instructions:

- Data Transfer Instructions
- I/O Instructions
- Arithmetic & Logical Instructions
- Control Transfer Instructions

Data Transfer Instructions:

The instructions used to transfer data from one component to another component during program execution are called data transfer instructions.

Programmer can use these instructions to move data from CPU to main memory.



Instructions Set

Arithmetic and logical Instructions:

The instructions used to perform arithmetic operations are called arithmetic instruction.

Different arithmetic operations are:

Addition + Subtraction - Multiplication * Division /

The instructions used to perform logical operations are called logical instruction.

Different logical operations are:

Less Than < Greater Than > Equal to =

I/O Instructions:

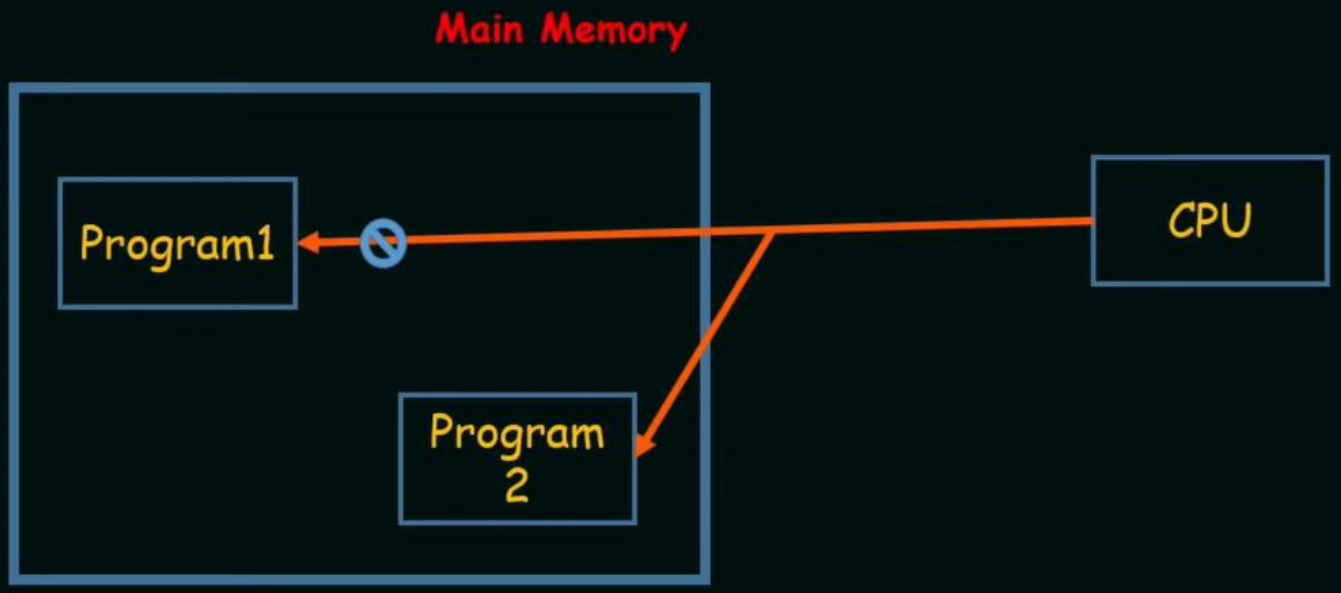
Every CPU provides the operations of reading data from peripheral devices and writing data to peripheral devices.

Peripheral devices means input and output devices such as Keyboard , mouse and monitor.

Instructions Set

Control Transfer Instructions:

The instructions used to change the sequence of instructions of a program are called control transfer instructions.



Instructions Format

Instruction:

An instruction is a statement that tells the computer to do something.

Instruction Format:

They way in which instruction is given is called instruction format.

- A computer has a variety of instruction format.

There are two common parts of instruction format that are as follows:

- Operand Code
- Address of Operand

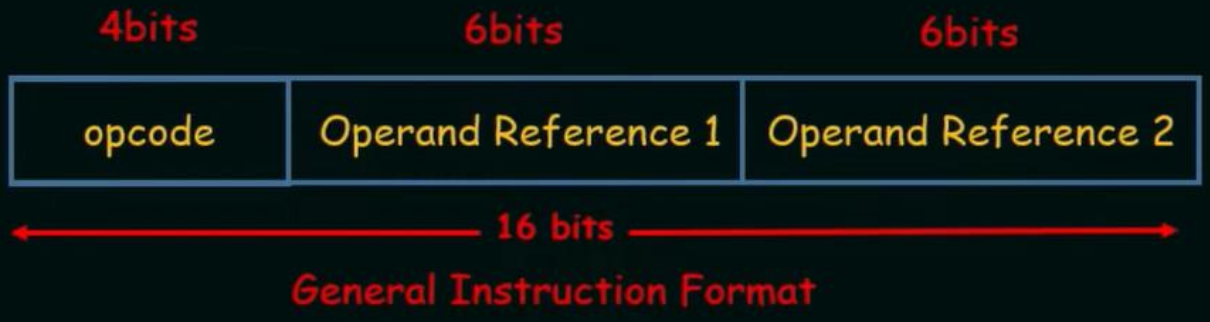
Operand Code:

The operand code specifies the operation to be performed by the computer such as ADD, SUB and MOV etc.

Address of Operand:

The address of operand refers to a location in main memory where the value is stored.

Instructions Format



General Instruction Format

Types of instruction Format:

1. Zero-Address Instruction Format
2. One-Address Instruction Format
3. Two-Address Instruction Format
4. Three-Address Instruction Format

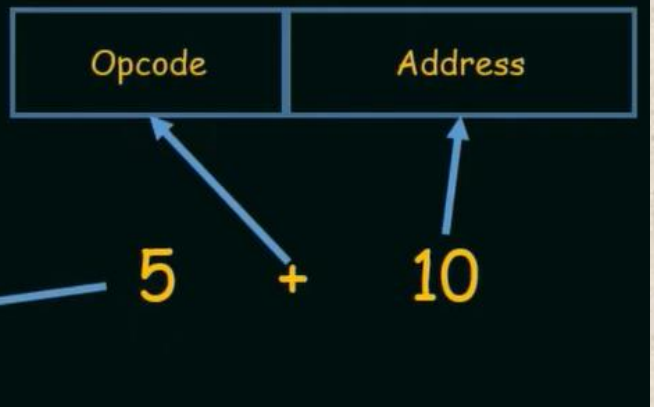
Instructions Format

Zero-Address Instruction Format:

- In zero address instruction format, an address field is absent in the instruction.
- Only opcode used in the instruction.
- Stack is used to store the data.

One-Address Instruction Format:

- This format use one address field with opcode.
- The other operand is stored on Accumulator register.



Instructions Format

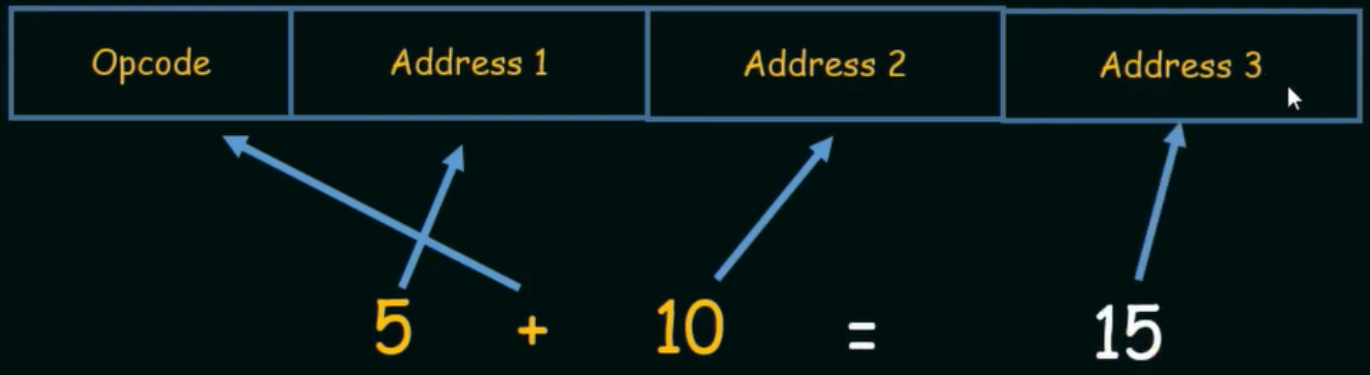
Two-Address Instruction Format:

- This format uses two address field.
- Each address field represent a memory address.

Instructions Format

Three-Address Instruction Format:

- This format uses Three address field.



Fetch-Decode-Execute Cycle

Fetch-Decode-Execute-Cycle:

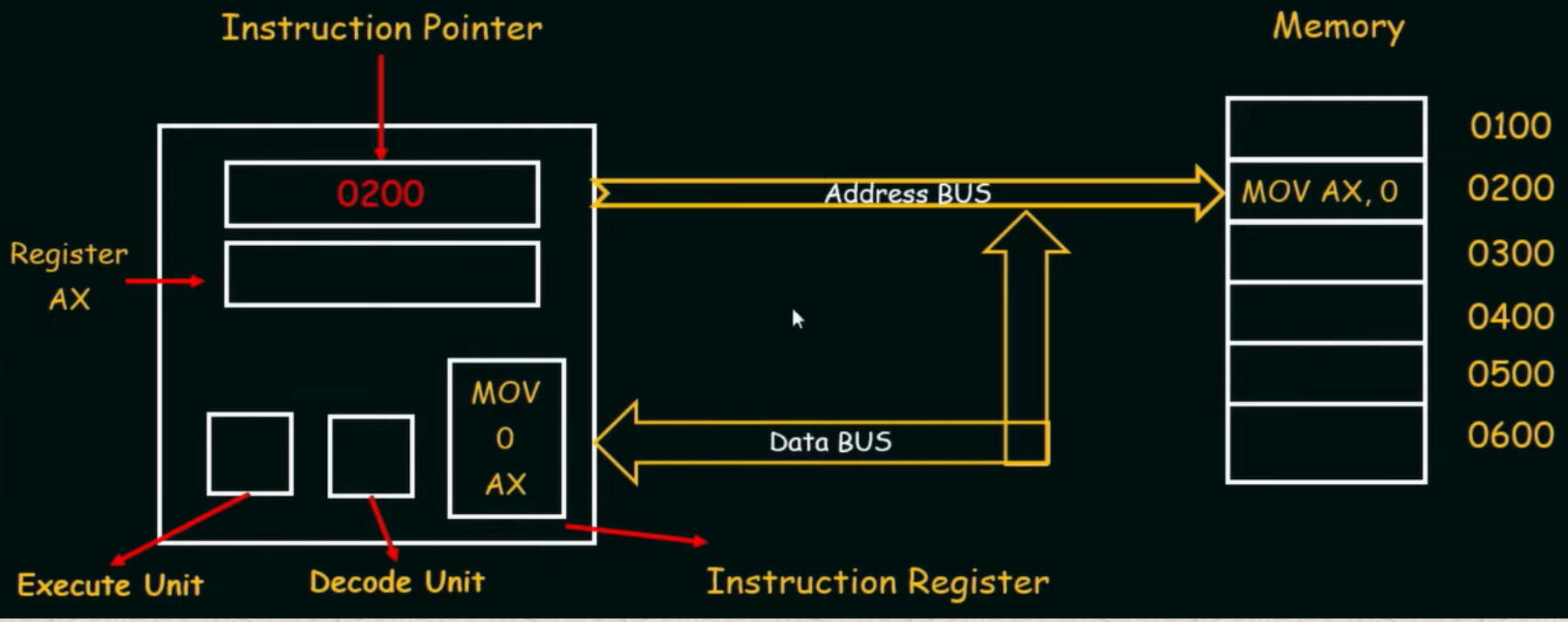
- Most modern processors work on fetch-decode-execute principle.
- When a set of instructions is to be executed, the instructions and data are loaded in main memory.

- The execution of an instruction by a processor is divided in three parts.
- These parts are fetch, decode and execute.
- This architecture is also called von Neumann Architecture.

Fetch-Decode-Execute Cycle

Fetch Instruction:

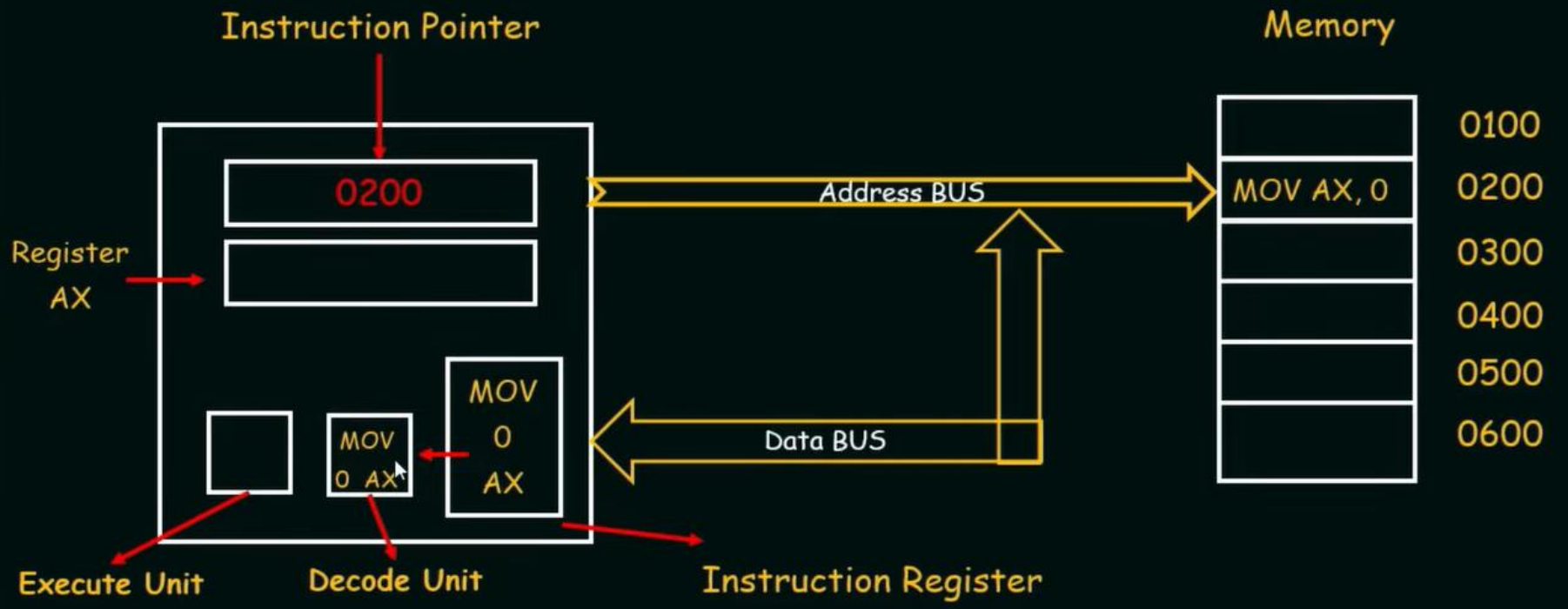
In this first step, the processor fetches the instruction from the memory. The instructions is transferred from memory to instruction register.



Fetch-Decode-Execute Cycle

Decode Instruction:

In this step, the instruction is decoded by the processor.



Fetch-Decode-Execute Cycle

Execute Instruction:

In this last phase. The processor executes the instruction. It stores 0 in register AX.