Computer Architecture

Lecture 03

Instruction Execution

Reference Books

- Computer Organization and Architecture: Designing for Performance-William Stallings
  (8th Edition)
  - Any later edition is fine

Overview

- At a top level, a computer consists of CPU (central processing unit), memory, and I/O components, with one or more modules of each type
- These components are interconnected in some fashion to achieve the basic function of the computer
- At a top level, we can describe a computer system by

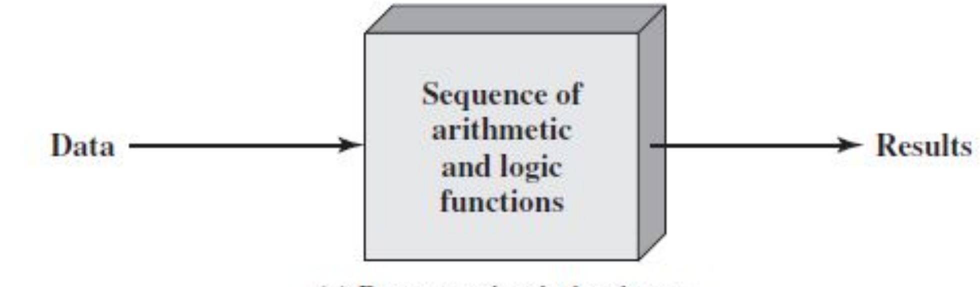
1. Describing the external behavior of each component— that is, the data and control signals that it exchanges with other components; and
2. Describing the interconnection and controls required structure manage the use the to interconnection of the structure

Overview

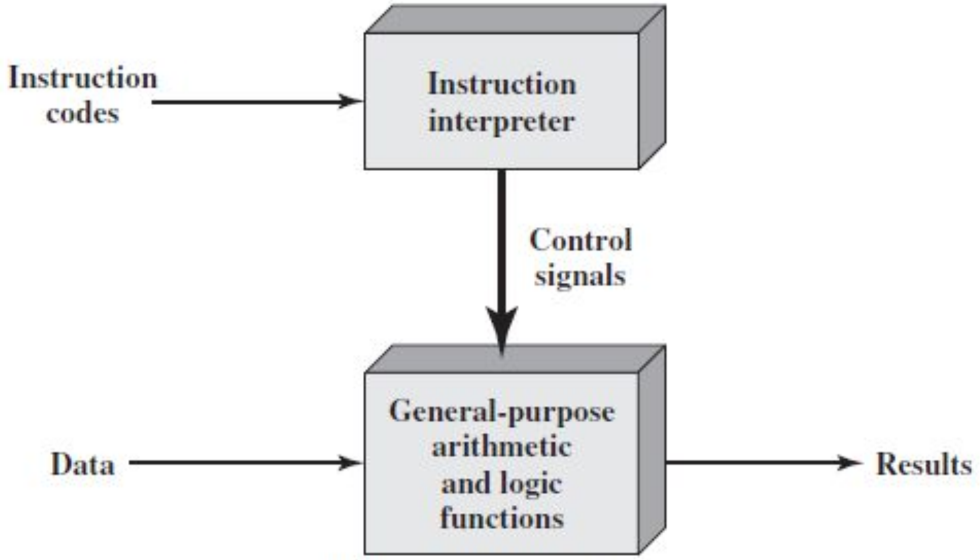
- Virtually all contemporary computer designs are based on concepts developed by John von Neumann at the Institute for Advanced Studies, Princeton
- Such a design is referred to as the *von Neumann architecture* and is based on three key concepts:
  - Data and instructions are stored in a single read–write memory
  - The contents of this memory are addressable by location, without regard to the type of data contained there
  - Execution occurs in a sequential fashion (unless explicitly modified) from one instruction to the next

Types of Programming

- Two types:
  - Hardwired Programming
  - Software Programming



(a) Programming in hardware

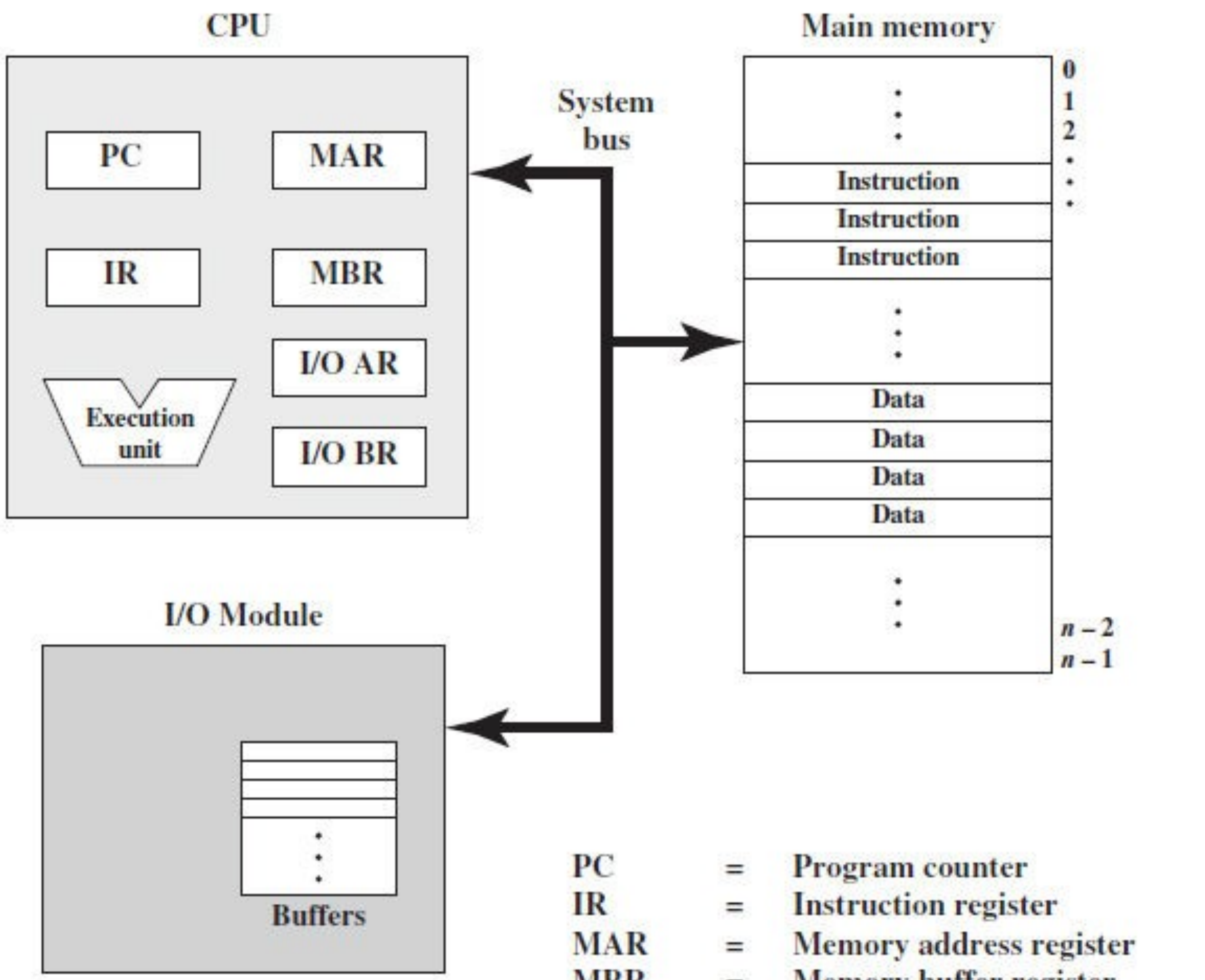


(b) Programming in software

GP Processor

- Figure b indicates two major components of the system: an instruction interpreter and a module of general-purpose arithmetic and logic functions
  - These two constitute the CPU
- Several other components areneeded to yield a functioning computer
- Data and instructions must be put into the system and results must be shown in realizable forms
  - We need I/O module for that
- A place is also needed for storing the data and instructions
- We need Memory for that

GP Processor



PC	=	Program counter
IR	=	Instruction register
MAR	=	Memory address register
MBR	=	Memory buffer register
I/O AR	=	Input/output address register
I/O BR	=	Input/output buffer register

GP Processor

- Figure illustrates these top-level components
- CPU exchanges data with memory
- For this purpose, it typically makes use of two internal (to the CPU) registers: a memory address register (MAR), which specifies the address in memory for the next read or write, and a memory buffer register (MBR), which contains the data to be written into memory or receives the data read from memory
- Similarly, an I/Oaddress register (I/OAR) specifies

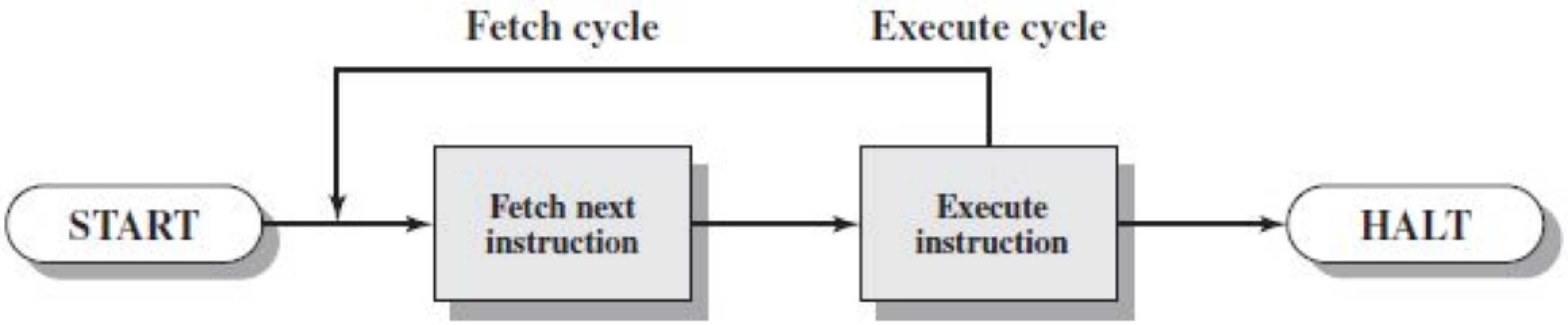
  a

  particular I/O device
- An I/O buffer (I/OBR) register is used for the exchange of data between an I/O module and the CPU

Function of GPP

- The basic function performed by a computer is execution of a program, which consists of a set of instructions stored in memory
- In its simplest form, instruction processing consists of two steps:
- The processor reads (*fetches*) instructions from memory one at a time and executes each instruction
- Program execution consists of repeating the process of instruction fetch and instruction execution
- The processing required for a single instruction is called an *instruction cycle*

Instruction Fetch and Execute



- In a typical processor, a register called the program counter (PC) holds the address of the instruction to be fetched next
- Unless told otherwise, the processor always increments the PC after each instruction fetch so that it will fetch the next instruction in sequence

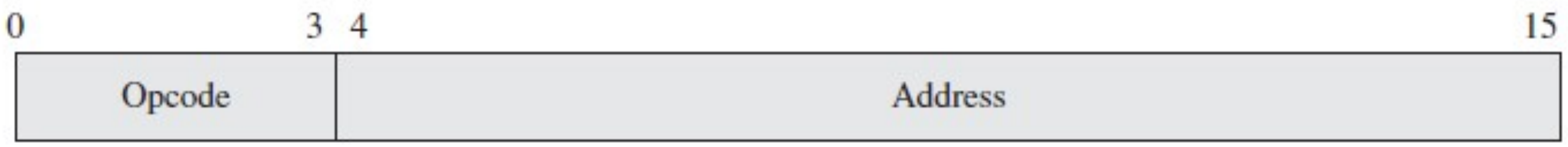
Instruction Fetch and Execute

- The fetched instruction is loaded into a register in the processor known as the instruction register (IR)
- The instruction contains bits that specify the action the processor is to take
- In general, these actions fall into four categories:
  - **Processor-memory**: Data may be transferred from processor to memory or from memory to processor
  - **Processor-I/O**: Data may be transferred to or from a peripheral device by transferring between the processor and an I/O module
  - **Data processing**: The processor may perform some arithmetic or logic operation on data
  - **Control**: An instruction may specify that the sequence of execution be altered

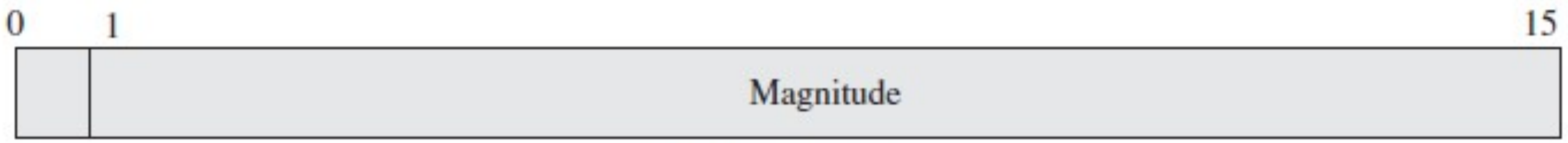
Instruction Fetch and Execute

- For example, the processor may fetch an instruction from location 149, which specifies that the next instruction be from location 182. The processor will remember this fact by setting the program counter to 182
- Thus, on the next fetch cycle, the instruction will be fetched from location 182 rather than 150

A Hypothetical Processor



(a) Instruction format



(b) Integer format

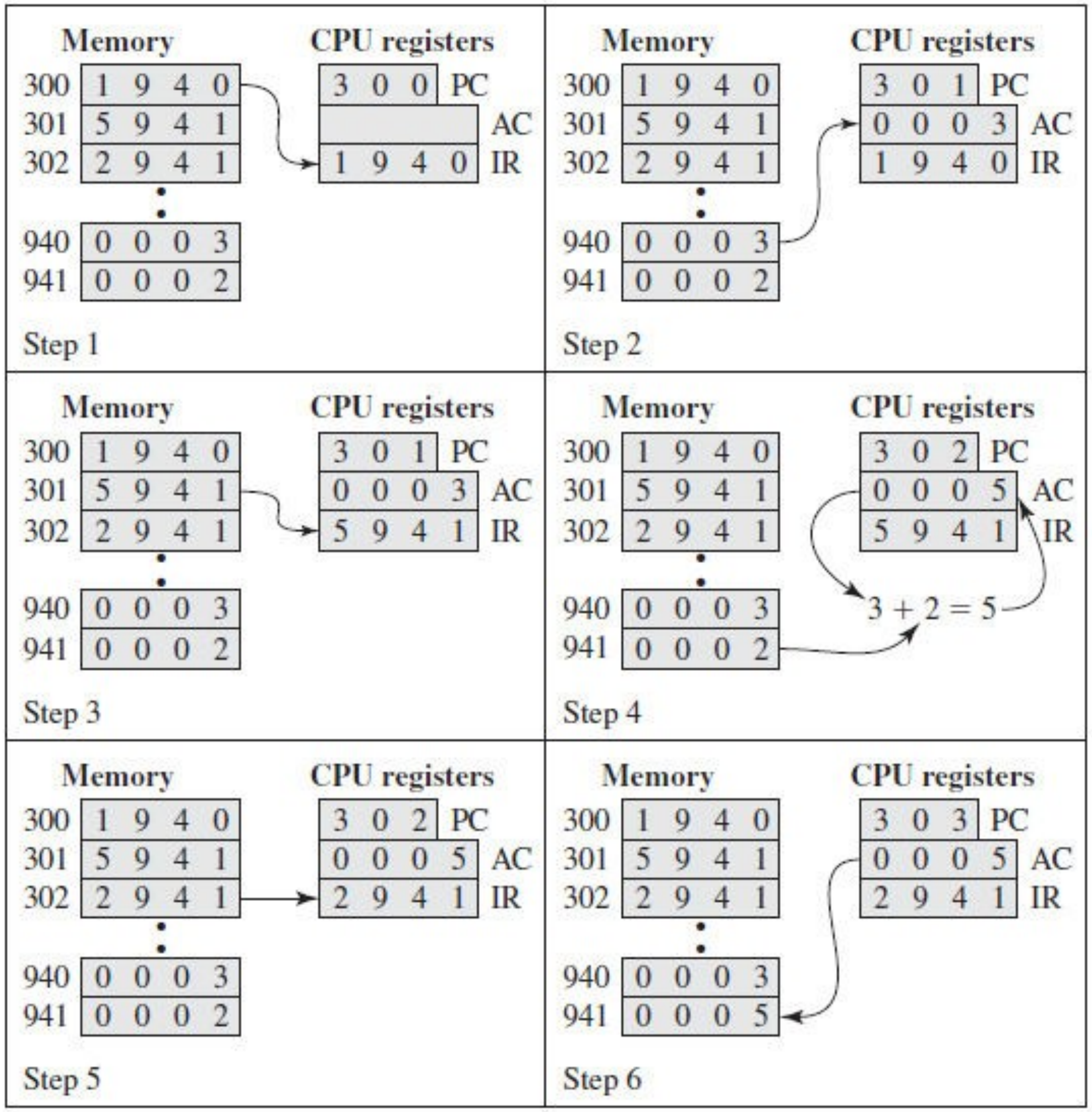
Program counter (PC) = Address of instruction
Instruction register (IR) = Instruction being executed
Accumulator (AC) = Temporary storage

(c) Internal CPU registers

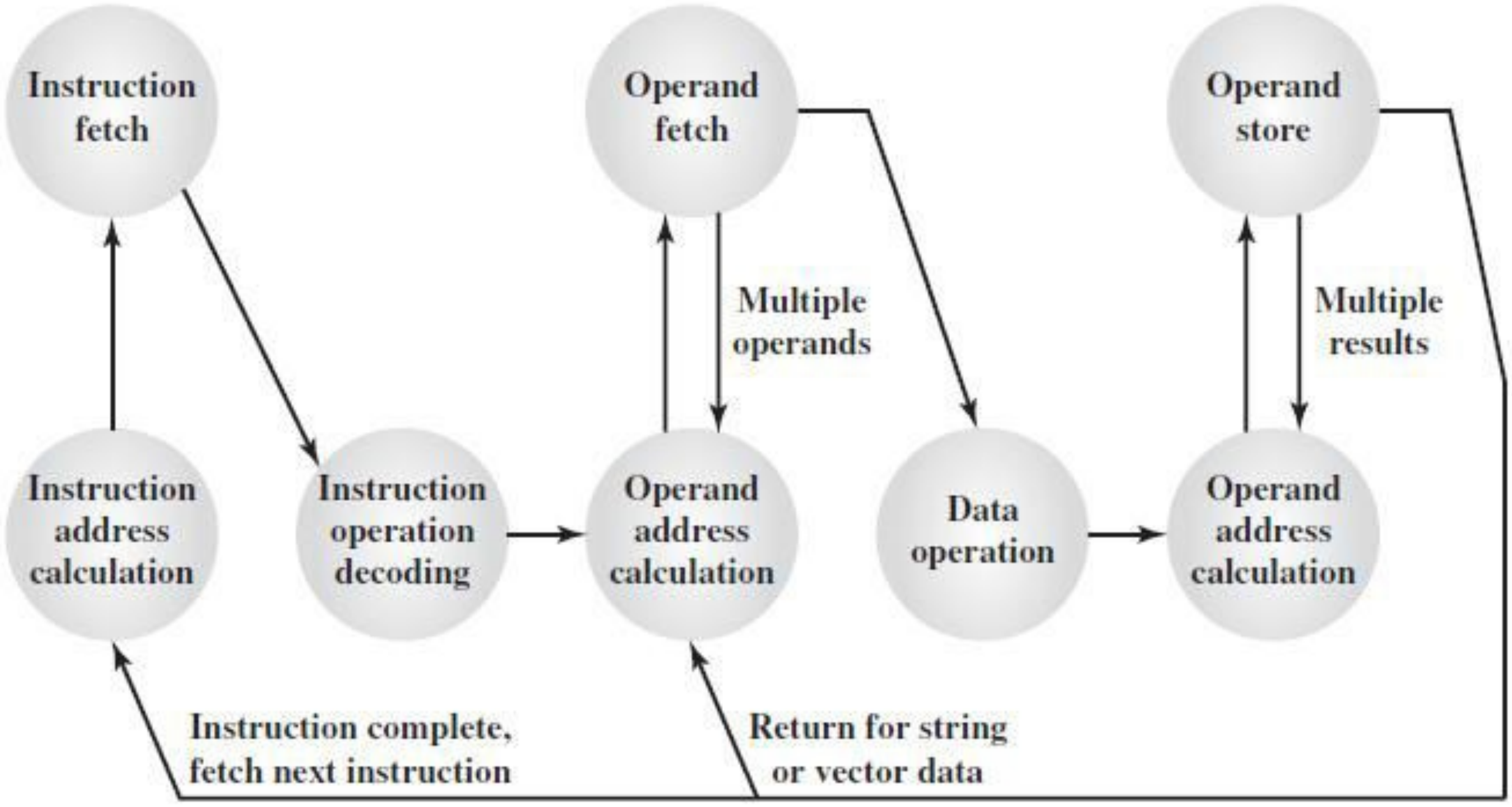
0001	= Load AC from memory
0010	= Store AC to memory
0101	= Add to AC from memory

(d) Partial list of opcodes

A Hypothetical Processor



Instruction Cycle State Diagram


Instruction fetch
Operand fetch
Operand store
Multiple operands
Multiple results
Instruction address calculation
Instruction operation decoding
Operand address calculation
Data operation
Operand address calculation
Instruction complete, fetch next instruction
Return for string or vector data

Instruction Cycle States

- Instruction address calculation (iac)
- Instruction fetch (if)
- Instruction operation decoding (iod)
- Operand address calculation (oac)
- Operand fetch (of)
- Data operation (do)
- Operand store (os)

Interrupts

- Virtually all computers provide a mechanism by which other modules (I/O, memory) may interrupt the normal processing of the processor
- Interrupts are provided primarily as a way to improve processing efficiency
- For details, refer to textbook

Assignment 1

- Write a short article on Interrupt mechanism found in Processors
- Make sure that your article covers the following topics:
  - Concept of Interrupt
  - Instruction Cycle State Diagram with interrupt
  - Transfer of control with interrupt
  - Applications