Chapter No. 03 – A Top-Level View of Computer Function and Interconnection

Week – 04

Date: 26-Feb to 2-Mar

Topics to Cover

- 3.1 Computer Components
- 3.2 Computer Function
- Instruction Fetch and Execute
- Interrupts
- I/O Function
- 3.3 Interconnection Structures
- 3.4 Bus Interconnection
- Bus Structure - Multiple Buses – Elements of Bus Design

Quiz-1 Today

Introduction

Skip

- At a top level, a computer consists of

1) CPU(Central Processing Unit) 2) memory
3) I/O components. 4) buses to interconnect

- These components are interconnected in some fashion to achieve the basic function of the computer, which is to **execute programs.**
- At a top-level view, a computer system is characterized by:

1. The external behaviour of each components, the data and control signals that it exchanges with other components.
2. The interconnection structure and the control signals required to manage the use of interconnection structure.

3.1 Computer Components

Skip

- All computer designs are based on the Von-Neumann architecture and is based on three key components:

1. Data and instructions are stored in a single read-write memory.
2. The contents of this memory are addressable by location, without regard to the type of data contained there.
3. Execution occurs in a sequential fashion (unless explicitly modified) from one instruction to the next.

- The basic logic components (transistors) can be combined in various ways to store binary data and perform ALU operations on the data.

What is a ‘Program’?

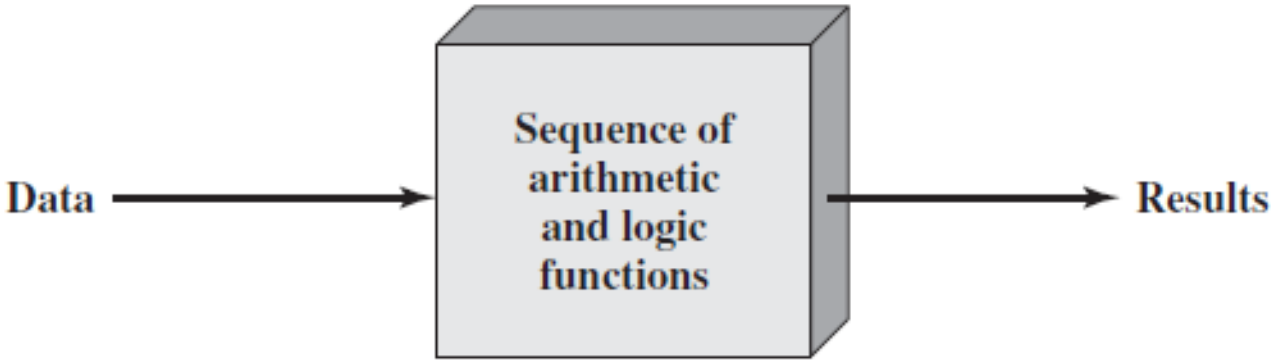
Skip

- A sequence of steps/instructions (written in logical order to perform a specific task).
- For each step, an arithmetic/logical operations are performed on data.
- For each operation, a different set of control signals is needed.
- Its executed in the CPU, which interprets each line of code, and generates appropriate control signals to get the task done.
- There are two approaches to writing a program:

1. Hardwired
2. Software

1. Hardwired Program

- The process of connecting the various components in the desired configuration as a form of programming e.g. in microcontrollers.
- The resulting 'program' is in the form of hardware and is termed as a **hardwired program.** (e.g. program is burnt/loaded into the IC).
- Hardwired systems are fast but inflexible (main disadvantage).

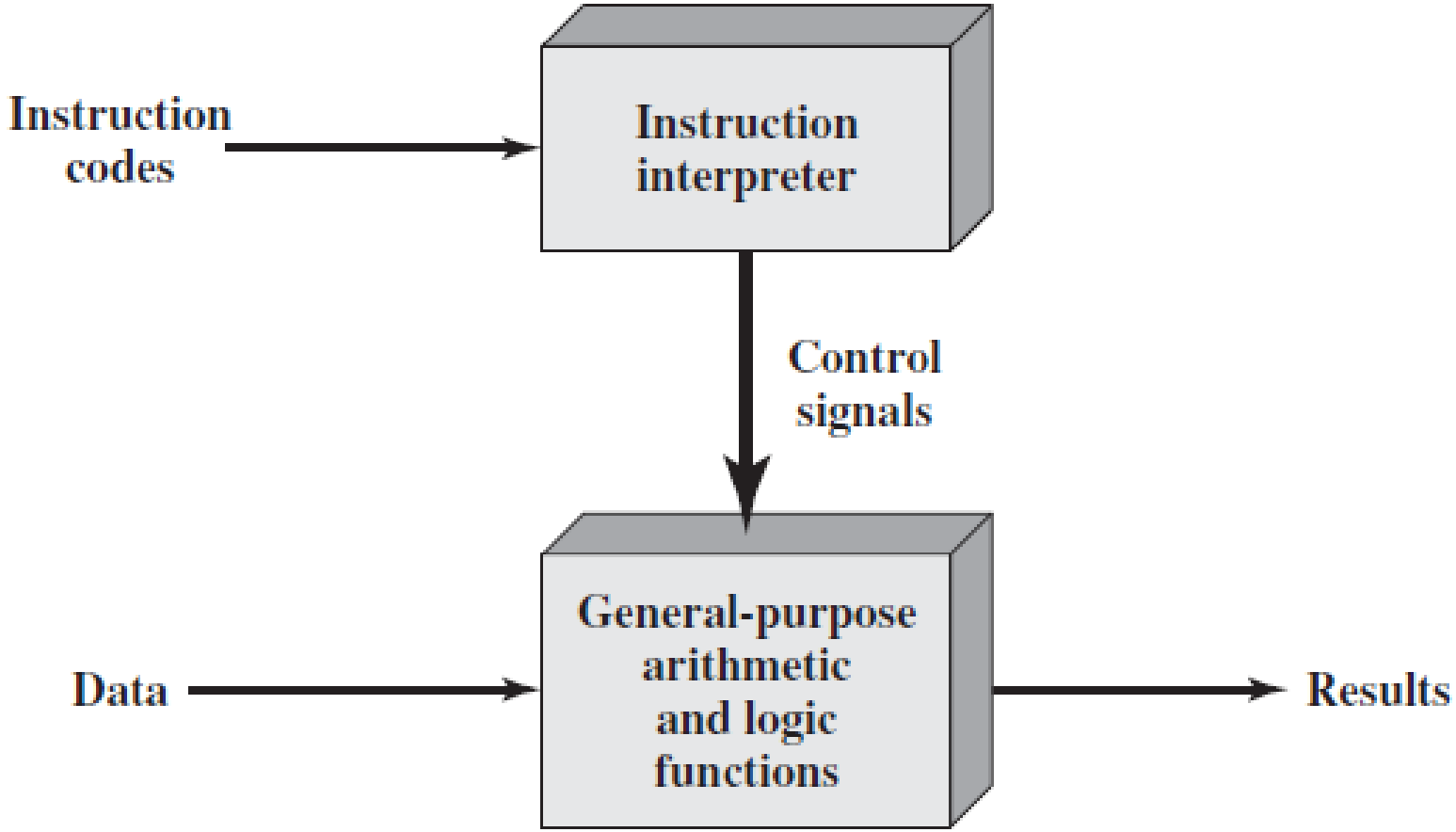


(a) Programming in hardware

2. Software Program (figure)

- With general-purpose hardware, the system accepts data and control signals and produces results. (can perform multipurpose tasks)
- Thus, instead of rewiring the hardware for each new program, the programmer merely needs to supply a new set of control signals.
- To supply these control signals, the general-purpose hardware needs a segment that can accept a code and generate control signals.
- Programming is now much easier, all we need to do is provide a new sequence of codes/instructions. Hardware interprets them and generates a unique set of control signals.
- This programming using a sequence of codes/instruction is **Software.**

Fig. 3.1 Programming in Software



(b) Programming in software

Components of the CPU

Skip

- Two major components of the system:

1. An instruction interpreter (Control Unit).
2. A module of general-purpose arithmetic and logic functions (ALU)

- These two constitute the **CPU** (Central processing unit).
- **Function of Control Unit**
- For each operation a unique code is provided.
  - e.g. ADD, MOVE
- A hardware segment accepts the code and issues the control signals.

I/O Components

Skip

- A **module** is a Computer circuit consisting of an assembly of electronic components (as of computer hardware)
- Data and instructions must be put into the system. For this we need some sort of **Input module.**
- This module contains basic components of accepting instructions and data in some form and converting them into an internal form of signals usable by the system.
- A means of reporting results is needed, and this is in the form of an **output module.**
- Taken together, these are referred to as the **I/O components.**

Main Memory (RAM)

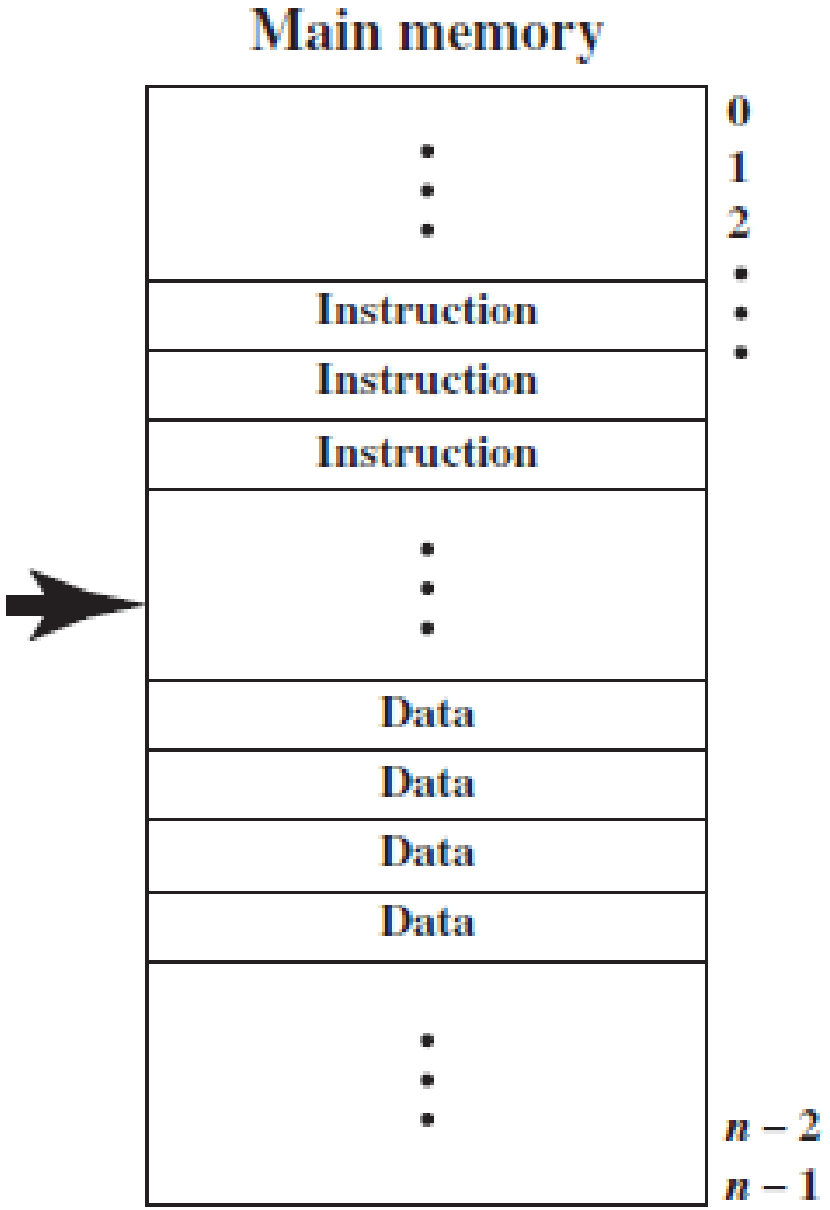
Skip

- An input device will bring instructions and data in sequentially.
- But a program is not invariably executed sequentially; it may **jump** around (e.g. using program flow control instructions).
- Similarly, operations on data may require access to more than just one element (operand) at a time in a predetermined sequence.
- Thus temporary storage of both code/inst. and results/data is needed.
- This module is called **memory,** or **main memory** called **(RAM).**
- RAM (Random access memory) is directly accessed by the CPU.
- This memory could be used for both instructions and data fetches and for storing results.

Memory Module

Skip

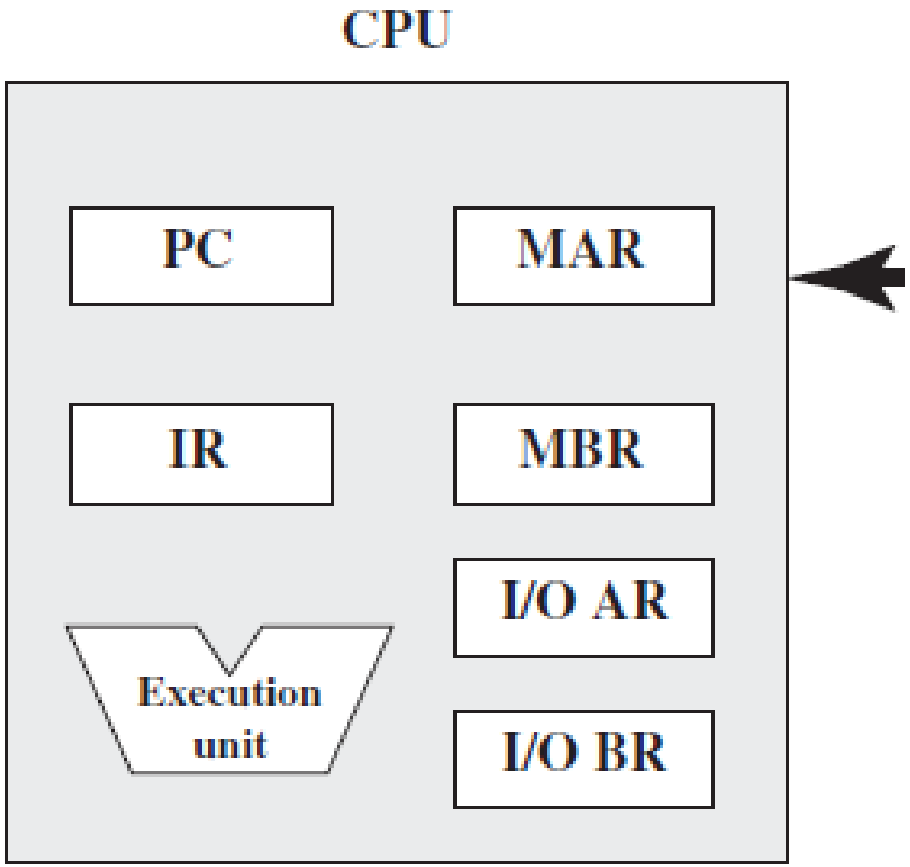
- A memory module consists of a set of locations, defined by sequentially numbered addresses.
- Each location contains a binary number that can be interpreted as either an instruction or data.
- E.g. for 32 bits system for our memory addresses. That works out to 2^32 unique combinations of addresses. Each address points to 1 byte of data. Therefore, we can access up to a total 2^32 bytes of data = 4GB of RAM.



CPU Communications with Memory

Skip

- The 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.
- A **memory buffer register (MBR),** which contains the data to be written into memory or receives the data read from memory.



CPU Communications with I/O

Skip

- An **I/O address register (IO/AR)** specifies a particular I/O device.
- An I/O module transfers data from external devices to CPU and memory, and vice versa.
- An **I/O buffer register (IO/BR)** is used for the exchange of data between an I/O module and the CPU.
- Modules contain internal buffers for temporarily holding these data until they can be sent on.

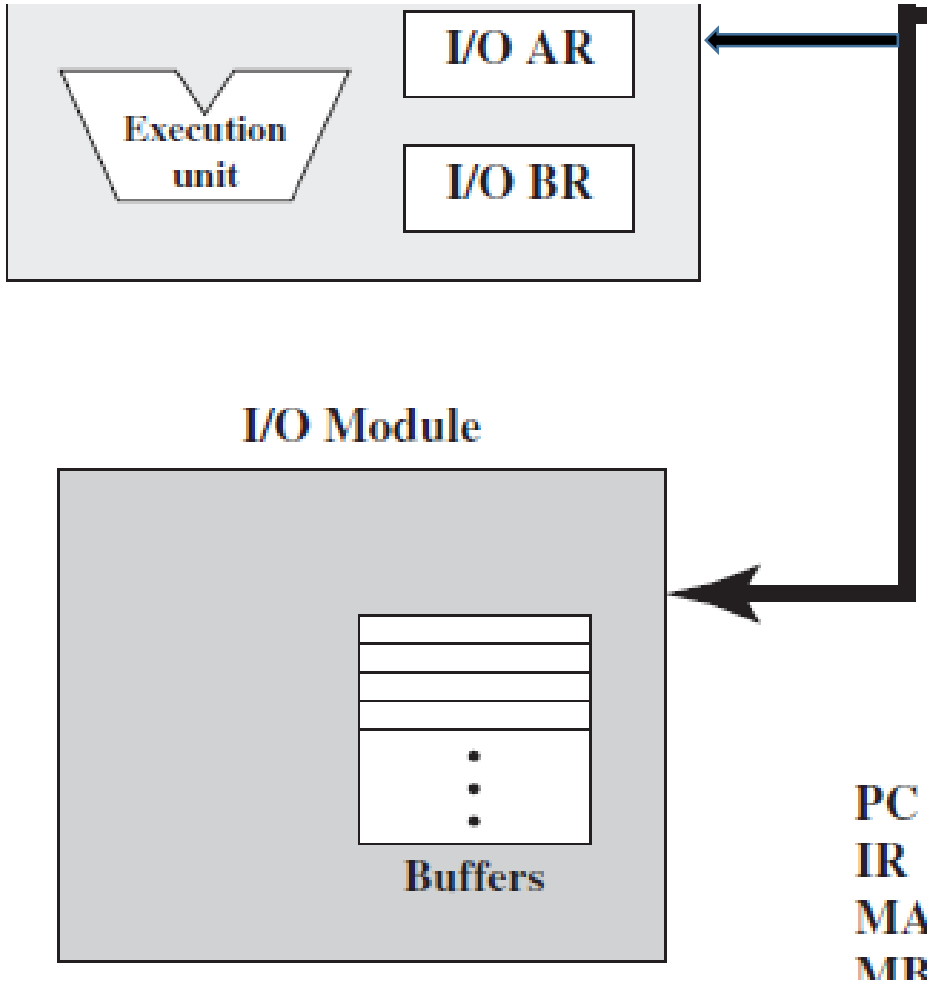
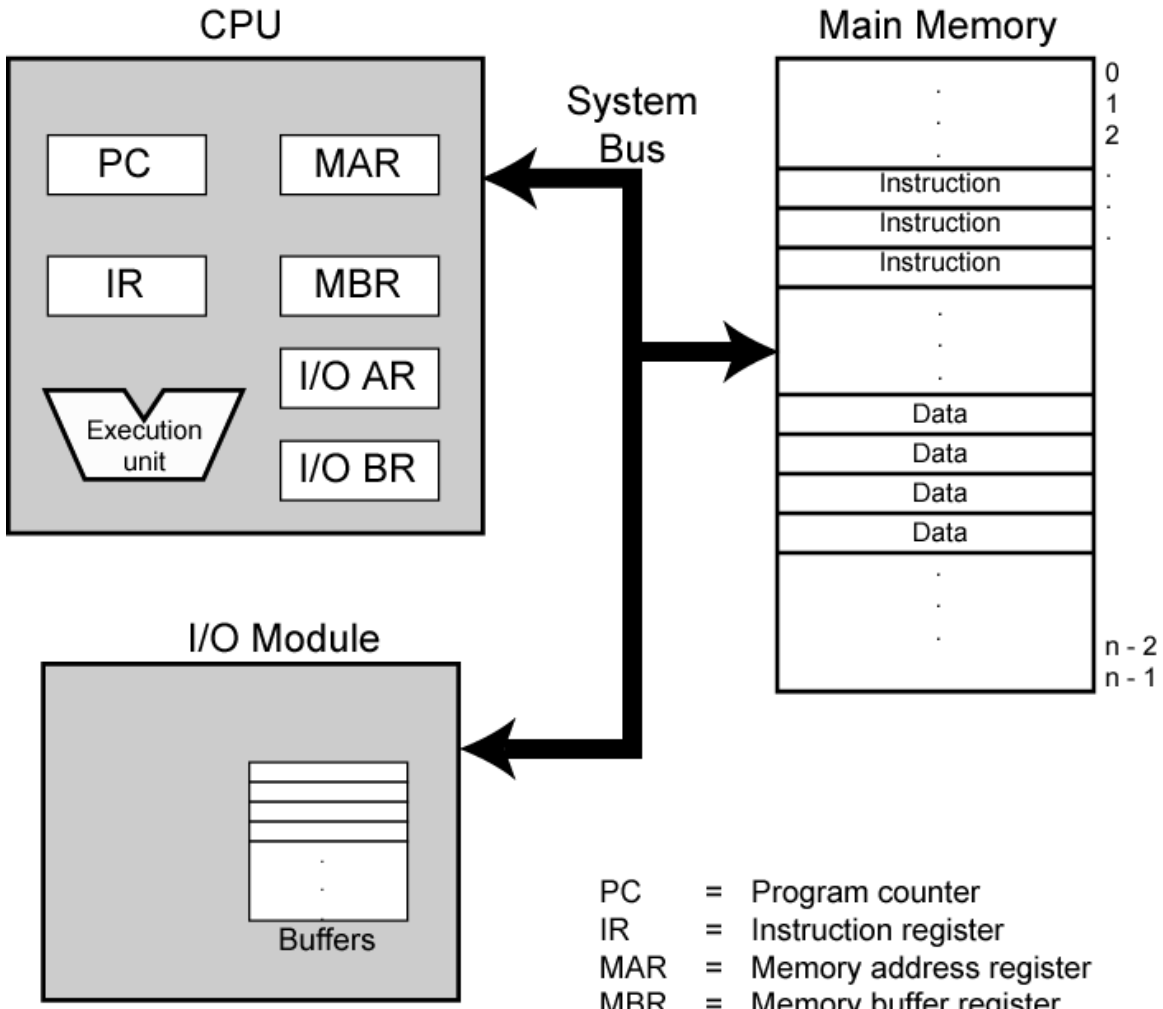


Fig. 3.2 Computer Components Top-Level View



Skip

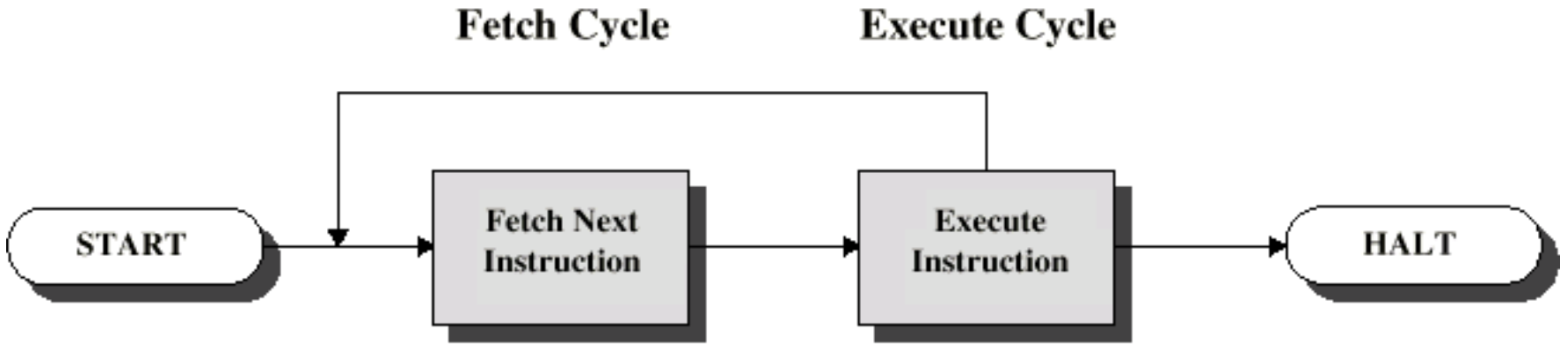
(Quiz-1)

3.2 Computer Function

- The basic function performed by computer is execution of a program.
- A program consists of a set of instructions stored in memory.
- The processor does the actual work by executing instructions specified in the program.
- Instruction processing consists of two steps:
- The processor reads **(fetches)** instructions from memory one at a time and **executes** each instruction. (and repeats the fetch cycle)
- Program execution consists of repeating the steps of instruction fetch and instruction execution. (until the last line of the code)
- The instruction execution may involve several operations and depends on the nature of the instruction.

Instruction Cycle

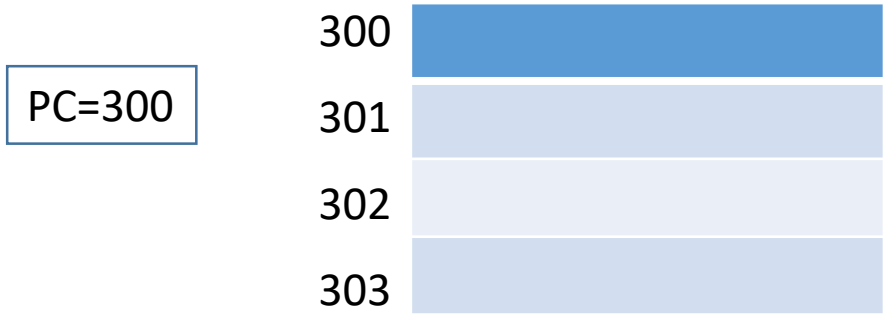
- The processing required for a single instruction is called an **instruction cycle.**
- Instruction cycle two steps are referred to as the:
- 1) Fetch cycle 2) Execute cycle
- **Fetch cycle** is the process by which a computer retrieves a program instruction from its memory.
- Then it determines what actions the instruction dictates, and carries out those actions called **execute cycle.**



1. Fetch Cycle

- At the beginning of each instruction cycle, the processor fetches an instruction from memory.
- A register called the **Program Counter (PC)** holds the address of the instruction to be fetched next.
- Processor fetches instruction from memory location pointed to by PC, unless program flow control is altered by a jump instruction.
- The processor always increments the PC after each instruction fetch, so that it will fetch the next instruction in sequence.
- The Next instruction is the instruction located at the next higher memory address.

Fetch Example



- For example, consider a computer in which each instruction occupies one 16-bit word of memory.
- Assume that the program counter (PC) is set to memory location 300, where the location address (300) refers to a 16-bit word.
- The processor will next fetch the instruction at location 300.
- On succeeding instruction cycles, it will fetch instructions from locations 301, 302, 303, and so on.
- This sequence may be altered, as explained in next slides.

2. Execute Cycle

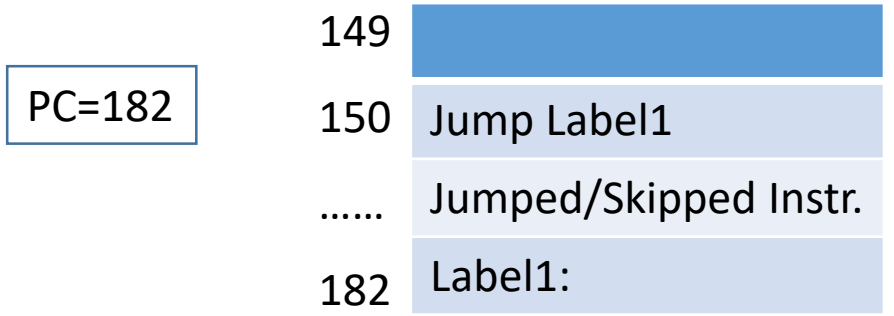
- In ‘Instruction Fetch’ we load the ‘PC (program counter)’ memory location value into the ‘IR (instruction register)’ and increments the PC.
- The fetched instruction is loaded into a register in the processor known as the **Instruction Register (IR).**
- The instruction contains the bits (opcode) that specify the action the processor is to take.
- The processor interprets the instruction (matches the opcode from its instruction set) and performs the required action.

General Categories of Functions Specified by Computer Instructions

- An instruction execution may involve a combination of these actions:

1. **Processor-memory:** Data may be transferred from processor to memory or from memory to processor. E.g. move instruction.
2. **Processor-I/O:** Data may be transferred to or from a peripheral device by transferring between the processor and an I/O module.
3. **Data processing:** The processor may perform some arithmetic or logic operation on data. E.g. Add, Sub, Mul, Div etc.
4. **Control:** An instruction may specify that the sequence of execution be altered. E.g. Jump instruction.

Execute Example



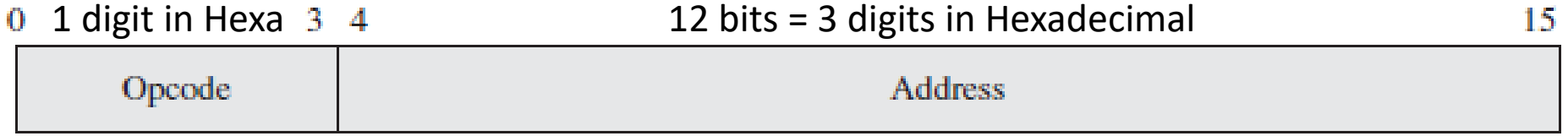
- For example, the processor may fetch an instruction from memory location 150.
- The location 150 specifies the next instruction to be from location 182. (Jump instruction).
- The processor will remember this fact by setting the program counter to 182.
- Thus, on the next fetch cycle, the instruction will be fetched from location 182 rather than 151.

Example of a Hypothetical Machine

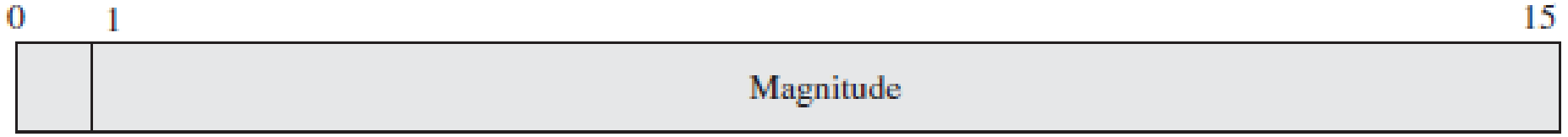
- This machine includes the characteristics listed in Figure next slide.
- The processor contains a single data register, called an accumulator (AC). Both instruction and data are 16 bits long.
- Thus, the memory is organized using 16-bit words.
- The instruction format provides 4 bits for the opcode, so that there can be as many as $2^4 = 16$ different opcodes.
- And up to $2^{12} = 4096(4K)$ words of memory can be directly addressed.

(See Next Slide)

Fig. 3.4 Characteristics of a Hypothetical Machine



(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

01h = 0001 = Load AC from memory
02h = 0010 = Store AC to memory
05h = 0101 = Add to AC from memory

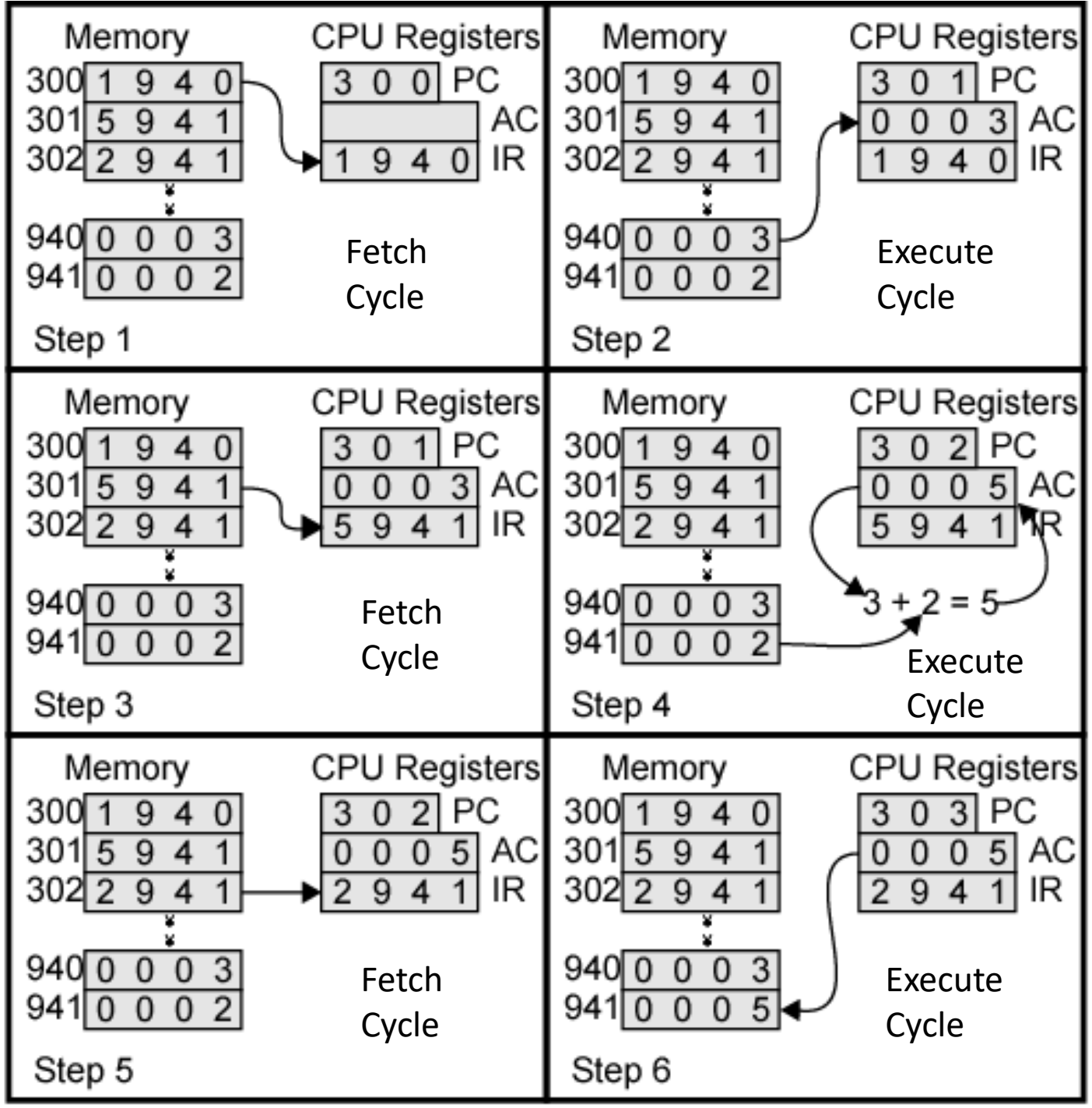
Opcodes used in Next Example

(d) Partial list of opcodes

Example of Program Execution

(Note: Hexadecimal Notation is used)

- Figure illustrates a partial program execution, showing the relevant portions of memory and processor registers.
- The program fragment shown adds the contents of the memory word at address 940 to the contents of the memory word at the address 941 and stores the result in location 941.
- Three instructions, which can be described as three fetch and three execute cycles are required.



Steps of Instruction Cycle for ADD Instruction (See Figure last slide)

1. The PC contains 300, the address of the first instruction. This instruction (the value 1940 in hexadecimal) is loaded into the instruction register IR, and the PC is incremented.
2. The first 4 bits (first hexadecimal digit) in the IR (opcode=1h) indicate that the AC is to be loaded. The remaining 12 bits (three hexadecimal digits) specify the address (940) from which data are to be loaded in the AC.
3. The next instruction (5941) is fetched from location 301, and the PC is incremented again.

Steps of Instruction Cycle for ADD Instruction (See Figure last slide)

4. The old contents of the AC and the contents of location 941 are added (opcode=05h), and the result is stored in the AC.
5. The next instruction (2941) is fetched from location 302, and the PC is again incremented.
6. Finally, the contents of the AC are stored (opcode=2) in location 941.

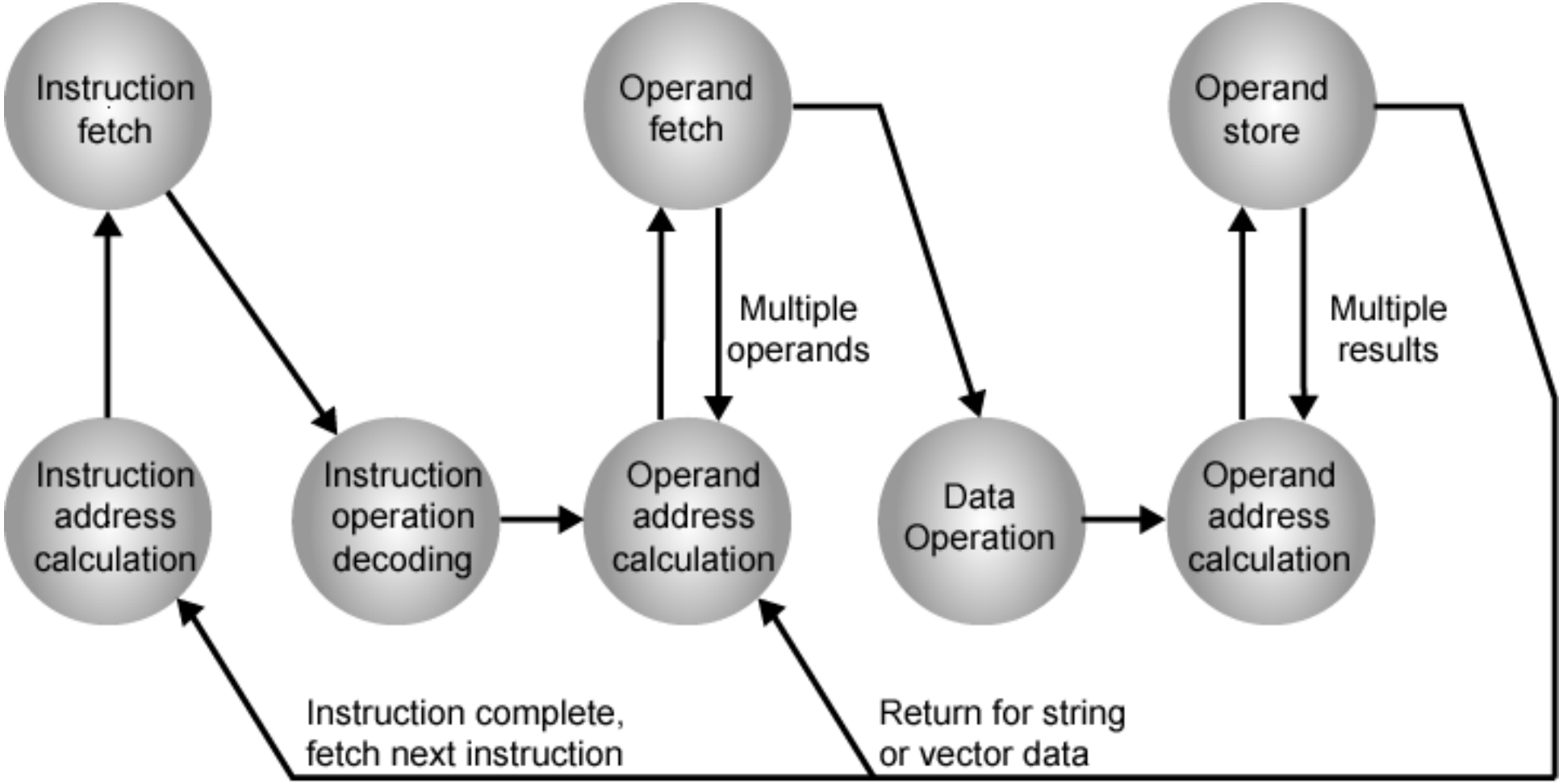
- In this example, three instruction cycles, each consisting of a fetch cycle and an execute cycle, are needed to add the contents of location 940 to the contents of 941.

Example Summarized in 3 Steps

- Instruction cycle-1: 940 value is loaded into the AC.
- Instruction cycle-2: Add AC in memory location 941.
- Instruction cycle-3: Store AC to location 941

- With a more complex set of instructions, fewer cycles would be needed.
- Also, instead of memory references, an instruction may specify an I/O operation. (opcode=read/write to I/O, address= of I/O device).See Slide-33

Fig. 3.6 Instruction Cycle State Diagram



Clock and Instruction Execution Cycle (Read)

The duration of a clock cycle is the reciprocal of the clock's speed, measured in oscillations per second. A clock that oscillates 1 billion times per second (1 GHz), for example, produces a clock cycle with a duration of one billionth of a second (1 nanosecond).

A machine instruction requires at least one clock cycle to execute, and a few require in excess of 50 clocks (the multiply instruction on the 8088 processor, for example). Instructions requiring memory access often have empty clock cycles called *wait states* because of the differences between the speed of the CPU, the system bus, and memory circuits. (Recent research suggests that in the near future we may abandon the synchronized computing model in favor of a type of asynchronous operation that would not require a system clock.)

2.1.2 Instruction Execution Cycle

The execution of a single machine instruction can be divided into a sequence of individual operations called the *instruction execution cycle*. When the CPU executes an instruction using a memory operand, it must calculate the address of the operand, place the address on the address bus, wait for memory to get the operand, and so on.

Instruction Execution Cycle (Read for Info)

Before it executes, a program must be loaded into memory. In Figure 2-2, the *program counter* is a register that contains the address of the next instruction about to be executed. The *instruction queue* is a holding area inside the microprocessor into which one or more instructions are copied just before they execute. When the CPU executes a single machine instruction, three primary operations are always necessary: *fetch*, *decode*, and *execute*. Two more steps are required when the instruction uses a memory operand: *fetch operand*, and *store output operand*. In other words, as many as five operations may be required by instructions that access memory.

- ***Fetch:*** The control unit fetches the instruction, copying it from memory into the CPU and increments the program counter (PC).
- ***Decode:*** The control unit determines the type of instruction to be executed. It passes zero or more operands to the arithmetic logic unit (ALU) and sends signals to the ALU that indicate the type of operation to be performed.
- ***Fetch operands:*** If a memory operand is used, the control unit initiates a read operation to retrieve the input operand from memory.
- ***Execute:*** The arithmetic logic unit executes the instruction, sends its data to the output operand, and updates status flags providing information about the output.
- ***Store output operand:*** If the output operand is in memory, the control unit initiates a write operation to store the data.

Preparatory Questions

Q1. What is an ‘instruction cycle’? Write down the two sub-cycles involved in it.

Q2. What happens to PC register when a processor executes a ‘branching’ statement?

Q3. What ‘instruction cycle’ steps are involved to add the contents of memory location 940 to 941, and placing the results at location 941. Explain.

Problems

3.1 The hypothetical machine of Figure 3.4 also has two I/O instructions:

03h = 0011 = Load AC from I/O
07h = 0111 = Store AC to I/O

In these cases, the 12-bit address identifies a particular I/O device. Show the program execution (using the format of Figure 3.5) for the following program: Refer to Slide-25

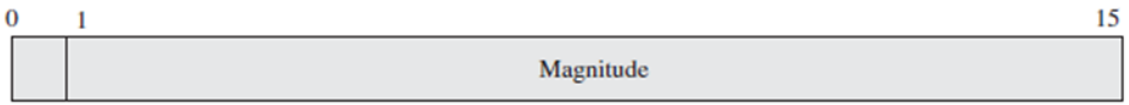
1. Load AC from device 5.
2. Add contents of memory location 940. = 05h
3. Store AC to device 6.

Assume that the next value retrieved from device 5 is 3 and that location 940 contains a value of 2.

Solution: **3.1** Memory (contents in hex): 300: 3005; 301: 5940; 302: 7006

Step 1: 3005 → IR; **Step 2:** 3 → AC
Step 3: 5940 → IR; **Step 4:** 3 + 2 = 5 → AC
Step 5: 7006 → IR; **Step 6:** AC → Device 6

(a) Instruction format



(b) Integer format

5-1. A computer uses a memory unit with 256K words of 32 bits each. A binary instruction code is stored in one word of memory. The instruction has four parts: an indirect bit, an operation code, a register code part to specify one of 64 registers, and an address part.

a. How many bits are there in the operation code, the register code part, and the address part?

b. Draw the instruction word format and indicate the number of bits in each part.

c. How many bits are there in the data and address inputs of the memory?

5.1 $256\ K = 2^8 \times 2^{10} = 2^{18}$

$64 = 2^6$

(a) Address: 18 bits

Register code: 6 bits

Indirect bit: 1 bit

Sum = 25 32 – 25 = 7 bits for opcode.

(b) 1 7 6 18 = 32 bits

I	opcode	Register	Address

(c) Data; 32 bits; address: 18 bits.