Computer Organization and Architecture

The distinction between computer architecture and organization is difficult to define precisely. However, the general areas covered by each are as follows:

Computer Organization

1. The term 'Computer Organization' refers to the operational units of the computer and their interconnections.

2. The main hardware units are the CPU, memory, I/O units, etc. The hardware details such as various computational units, control signals, interfaces between computer peripherals, and the memory technology used are included in the organization of the computer.

3. Therefore, it can be said that the hardware units and their connection details cover the organization of the computer system.

Computer Architecture

1. By the term 'Computer architecture', it is generally meant those qualities of a system that are visible to a programmer, such as the number of bits used to represent various data types, the instruction set of the computer, techniques for addressing memory, methods used for input-output, etc.

2. These qualities are known as architectural qualities and have an effect on the execution of a program.

3. In short, the quality associated with the functioning of software is linked to computer architecture whereas those associated to hardware aspects of the computer are related to computer organization.

4. For example, if a computer is running multiple instructions, it is an architectural design issue. On the other hand, it is an organizational issue whether that instruction will be implemented by a special multiple unit or by a unit that makes repeated use of the add instruction. Many computer manufactures offer a family of computer models, all with the same architecture but with differences in organization.

REGISTERS AND THEIR TYPES

Register are used to quickly accept, store, and transfer data and instructions that are being used immediately by the CPU, there are various types of Registers those are used for various purpose. Among of the some Mostly used Registers named as AC or **Accumulator**, Data Register or DR, the AR or **Address Register**, **program counter** (PC), **Memory Data Register**(MDR) ,**Index register**,***Memory Buffer Register***.

These Registers are used for performing the various Operations. While we are working on the System then these Registers are used by the **CPU for Performing the Operations**. When We Gives Some

Input to the System then the I**nput will be Stored into the Registers** and When the System will gives us the Results after Processing then the Result will also be from the Registers. So that they are used by the **CPU for Processing the Data** which is given by the User. Registers Perform:-+

1) **Fetch**: The Fetch Operation is used for taking the instructions those are given by the user and the Instructions those are stored into the Main Memory will be fetch by using Registers.

2) **Decode**: The Decode Operation is used for interpreting the Instructions means the Instructions are decoded means the CPU will find out which Operation is to be performed on the Instructions.

3) **Execute**: The Execute Operation is performed by the CPU. And Results those are produced by the CPU are then Stored into the Memory and after that they are displayed on the user Screen.

Types of Registers are as Followings

MAR stand for Memory Address Register

This register holds the memory addresses of data and instructions. This register is used to access data and instructions from memory during the execution phase of an instruction. **Suppose CPU wants to store some data in the memory or to read the data from the memory. It places the address of the-required memory location in the MAR**.

Program Counter

The **program counter (PC)**, commonly called the **instruction pointer** (IP) in Intel x86 microprocessors, and sometimes called the **instruction address register**, or just part of the instruction sequencer in some computers, is a processor register

It is a 16 bit special function register in the 8085 microprocessor. It keeps track of the the **next memory address** of the instruction that is to be executed once the execution of the current instruction is completed. **In other words, it holds the address of the memory location of the next instruction when the current instruction is executed by the microprocessor.**

Accumulator Register

This Register is used for storing the Results those are produced by the System. When the CPU will generate Some Results after the Processing then all the Results will be Stored into the **AC Register**.

Memory Data Register (MDR)

MDR is the register of a computer's control unit that contains the **data to be stored in the computer storage** (e.g. RAM), or the **data after a fetch from the computer storage**. It acts **like a buffer** and holds anything that is copied from the memory ready for the processor to use it. **MDR hold the information before it goes to the decoder.**

MDR which contains the data to be written into or readout of the addressed location. For example, to retrieve the contents of cell 123, we would load the value 123 (in binary, of course) into the MAR and perform a fetch operation. When the operation is done, a copy of the contents of cell 123 would be in the MDR. To store the value 98 into cell 4, we load a 4 into the MAR and a 98 into the MDR and perform a store. When the operation is completed the contents of cell 4 will have been set to 98, by discarding whatever was there previously.

The MDR is a two-way register. When data is fetched from memory and placed into the MDR, it is written to in one direction. When there is a write instruction, the data to be written is placed into the MDR from another CPU register, which then puts the data into memory.

The Memory Data Register is half of a minimal interface between a micro program and computer storage, the other half is a memory address register.

Index Register

A hardware element which holds a number that can be added to (or, in some cases, subtracted from) the address portion of a computer instruction to form an effective address. Also known as **base register**. An index register in a computer's CPU is a processor register used for modifying operand addresses during the run of a program.

Memory Buffer Register

MBR stand for ***Memory Buffer Register***. This register holds the contents of data or instruction read from, or written in memory. It means that this register is used to store data/instruction coming from the memory or going to the memory.

Data Register

A register used in microcomputers to temporarily store data being transmitted to or from a peripheral device.

Computer Instructions

Computer instructions are a set of machine language instructions that a particular processor understands and executes. A computer performs tasks on the basis of the instruction provided.

An instruction comprises of groups called fields. These fields include:

- The Operation code (Opcode) field which specifies the operation to be performed.
- The Address field which contains the location of the operand, i.e., register or memory location.
- The Mode field which specifies how the operand will be located.

A basic computer has three instruction code formats which are:

1. Memory - reference instruction
2. Register - reference instruction
3. Input-Output instruction

Memory - reference instruction

In Memory-reference instruction, 12 bits of memory is used to specify an address and one bit to specify the addressing mode 'I'.

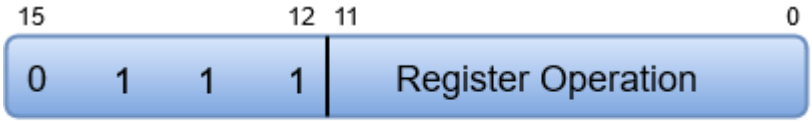
(Opcode = 000 through 110)

Register - reference instruction

The Register-reference instructions are represented by the Opcode 111 with a 0 in the leftmost bit (bit 15) of the instruction.

Note: The Operation code (Opcode) of an instruction refers to a group of bits that define arithmetic and logic operations such as add, subtract, multiply, shift, and compliment.

A Register-reference instruction specifies an operation on or a test of the AC (Accumulator) register.



(Opcode = 111, I = 0)

Input-Output instruction

Just like the Register-reference instruction, an Input-Output instruction does not need a reference to memory and is recognized by the operation code 111 with a 1 in the leftmost bit of the instruction. The remaining 12 bits are used to specify the type of the input-output operation or test performed.

(Opcode = 111, I = 1)

Instruction Cycle

An instruction cycle, also known as fetch-decode-execute cycle is the basic operational process of a computer. This process is repeated continuously by CPU from boot up to shut down of computer.

Following are the steps that occur during an instruction cycle:

1. Fetch the Instruction

The instruction is fetched from memory address that is stored in PC(Program Counter) and stored in the instruction register IR. At the end of the fetch operation, PC is incremented by 1 and it then points to the next instruction to be executed.

2. Decode the Instruction

The instruction in the IR is executed by the decoder.

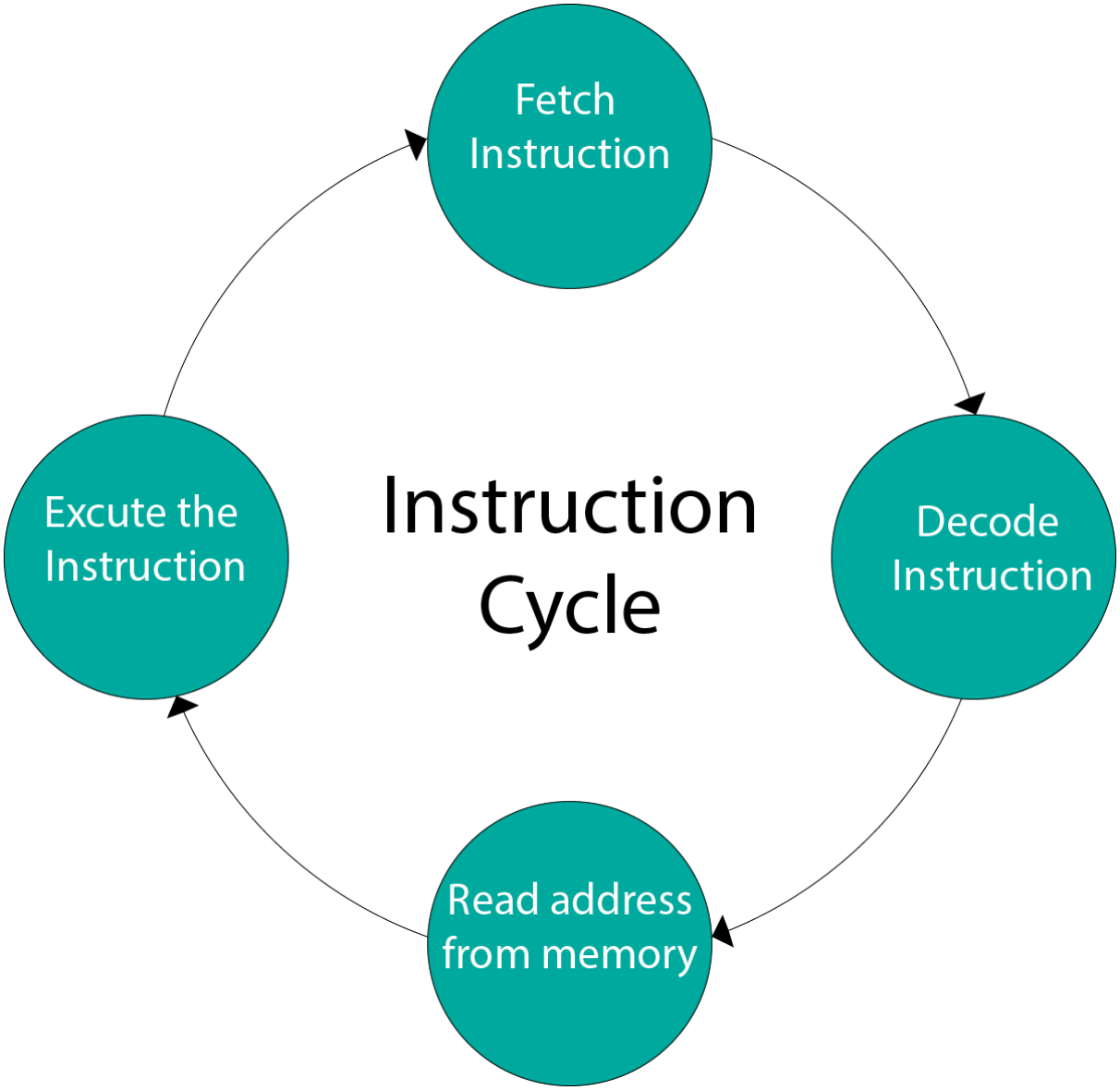
3. Read the Effective Address

If the instruction has an indirect address, the effective address is read from the memory. Otherwise operands are directly read in case of immediate operand instruction.

4. Execute the Instruction

The Control Unit passes the information in the form of control signals to the functional unit of CPU. The result generated is stored in main memory or sent to an output device.

The cycle is then repeated by fetching the next instruction. Thus in this way the instruction cycle is repeated continuously.



Memory-Reference Instructions

In order to specify the microoperations needed for the execution of each instruction, it is necessary that the function that they are intended to perform be defined precisely. Some instructions have an ambiguous description. This is because the explanation of an instruction in words is usually lengthy, and not enough space is available in the table for such a lengthy explanation.

There are 7 memory reference instructions as mentioned below.

- LDA
- STA
- AND

· ADD

· BUN

· BSA

· ISZ

LDA : Load to AC

This instruction transfers the memory word specified by the effective address to AC . The microoperations needed to execute this instruction are

DR← M [AR]
AC← DR , ← 0

STA: Store AC

This instruction stores the content of AC into the memory word specified by the effective address. Since the output of AC is applied to the bus and the data input of memory is connected to the bus, we can execute this instruction with one microoperation:

M [AR] <-- AC, SC <-- 0

AND to AC

This is an instruction that perform the AND logic operation on pairs of bits in AC and the memory word specified by the effective address. The result of the operation is transferred to AC . The microoperations that execute this instruction are:

DR <- M [AR]
AC <- AC /\ DR, SC <--- 0

ADD to AC

This instruction adds the content of the memory word specified by the effective address to the value of AC . The sum is transferred into AC and the output carry C_{out} is transferred to the E (extended accumulator) flip-flop. The rnicrooperations needed to execute this instruction are
DR← M[AR]

AC← AC + DR, E← C_{out} , SC ← 0

BUN: Branch Unconditionally

This instruction transfers the program to the instruction specified by the effective address. The instruction is executed with one microoperation:
PC ← AR, SC ← 0

BSA: Branch and Save Return Address

This instruction is useful for branching to a portion of the program called a subroutine or procedure. When executed, the BSA instruction stores the address of the next instruction in sequence (which is available in PC) into a memory location specified by the effective address. The effective address plus one is then transferred to PC to serve as the address of the first instruction in the subroutine.

M[AR] <-- PC, PC <-- AR + I

ISZ: Increment and Skip if Zero

This instruction increments the word specified by the effective address, and if the incremented value is equal to 0, PC is incremented by 1. The programmer usually stores a negative number (in 2's complement) in the memory word. As this negative number is repeatedly incremented by one, it eventually reaches the value of zero. At that time PC is incremented by one in order to skip the next instruction in the program.

This is done with the following sequence of microoperations:
D6T4: DR <-- M [AR]
D6T5: DR <-- DR + 1
D,T,: M [AR] <-- DR,
if (DR = 0) then (PC ← PC + 1), SC ← 0

Instruction Set Completeness

A set of instructions is said to be complete if the computer includes a sufficient number of instructions in each of the following categories:

- Arithmetic, logical and shift instructions
- A set of instructions for moving information to and from memory and processor registers.
- Instructions which controls the program together with instructions that check status conditions.
- Input and Output instructions

Arithmetic, logic and shift instructions provide computational capabilities for processing the type of data the user may wish to employ.

A huge amount of binary information is stored in the memory unit, but all computations are done in processor registers. Therefore, one must possess the capability of moving information between these two units.

Program control instructions such as branch instructions are used change the sequence in which the program is executed.

Input and Output instructions act as an interface between the computer and the user. Programs and data must be transferred into memory, and the results of computations must be transferred back to the user.

Timing and control

Control unit generates **timing and control** signals for the operations of the **computer**.
The **control** unit communicates with ALU and main memory. It also **controls** the transmission between processor, memory and the various peripherals. It also instructs the ALU which operation has to be performed on data.

It tells the computer's **memory**, arithmetic and logic unit and input and **output** devices how to respond to the instructions that have been sent to the **processor**. It directs the operation of the other units by providing timing and control signals.

Design of Control Unit

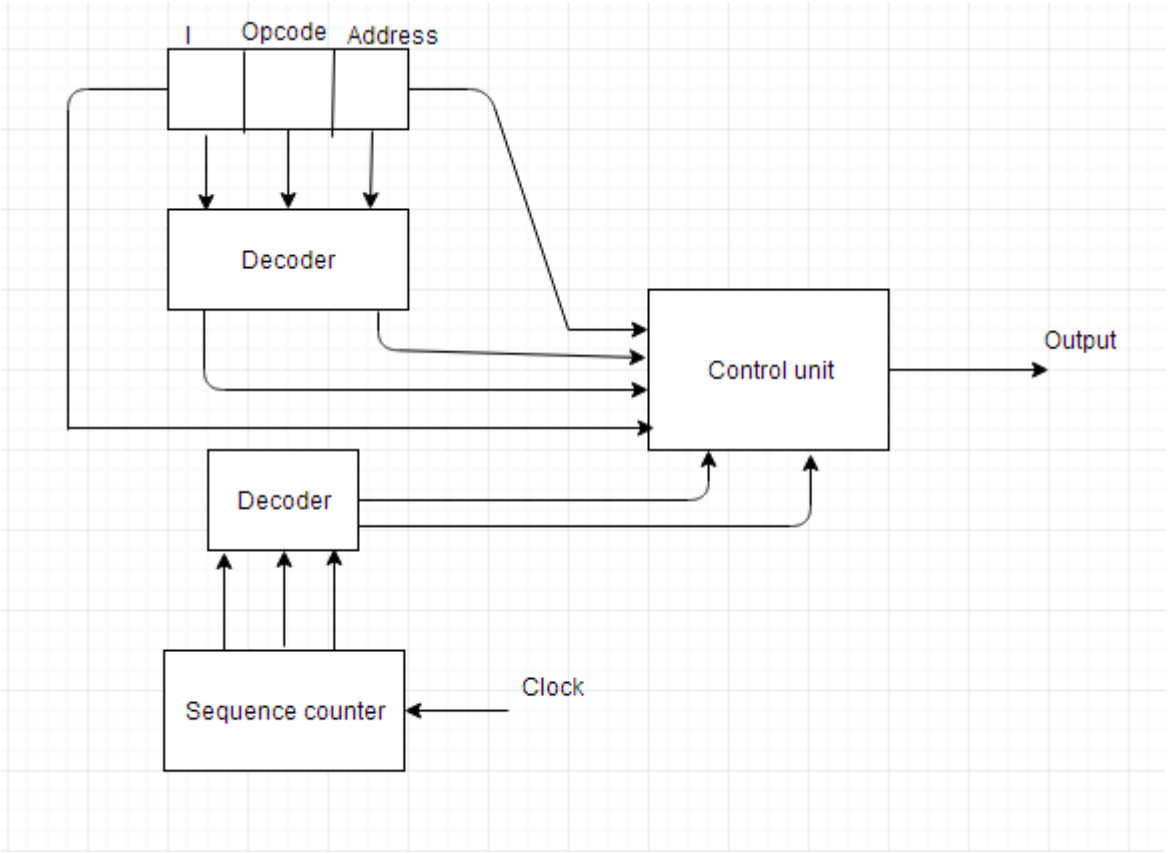
Control unit generates timing and control signals for the operations of the computer. The control unit communicates with ALU and main memory. It also controls the transmission between processor, memory and the various peripherals. It also instructs the ALU which operation has to be performed on data.

Control unit can be designed by two methods which are given below:

Hardwired Control Unit

It is implemented with the help of gates, flip flops, decoders etc. in the hardware. The inputs to control unit are the instruction register, flags, timing signals etc. This organization can be very complicated if we have to make the control unit large.

If the design has to be modified or changed, all the combinational circuits have to be modified which is a very difficult task.



Microprogrammed Control Unit

It is implemented by using programming approach. A sequence of micro operations is carried out by executing a program consisting of micro-instructions. In this organization any modifications or changes can be done by updating the micro program in the control memory by the programmer.

Difference between Hardwired Control and Microprogrammed Control

Hardwired Control	**Microprogrammed Control**
Technology is circuit based.	Technology is software based.
It is implemented through flip-flops, gates, decoders etc.	Microinstructions generate signals to control the execution of instructions.
Fixed instruction format.	Variable instruction format (16-64 bits per instruction).
Instructions are register based.	Instructions are not register based.
ROM is not used.	ROM is used.
It is used in RISC.	It is used in CISC.
Faster decoding.	Slower decoding.
Difficult to modify.	Easily modified.
Chip area is less.	Chip area is large.

Computer Architecture: Interrupts

When a Process is executed by the CPU and when a user Request for another Process then this will create disturbance for the Running Process. This is also called as the **Interrupt**.

Interrupts can be generated by User, Some Error Conditions and also by Software's and the hardware's. But CPU will handle all the Interrupts very carefully because when Interrupts are generated then the CPU must handle all the Interrupts Very carefully means the CPU will also Provides Response to the Various Interrupts those are generated. So that When an interrupt has Occurred then the CPU will handle by using the Fetch, decode and Execute Operations.

Data transfer between the CPU and the peripherals is initiated by the CPU. But the CPU cannot start the transfer unless the peripheral is ready to communicate with the CPU. When a device is ready to communicate with the CPU, it generates an interrupt signal. A number of input-output devices are attached to the computer and each device is able to generate an interrupt request.

The main job of the interrupt system is to identify the source of the interrupt. There is also a possibility that several devices will request simultaneously for CPU communication. Then, the interrupt system has to decide which device is to be serviced first.

Priority Interrupt

A priority interrupt is a system which decides the priority at which various devices, which generates the interrupt signal at the same time, will be serviced by the CPU. The system has authority to decide which conditions are allowed to interrupt the CPU, while some other interrupt is being serviced. Generally, devices with high speed transfer such as *magnetic disks* are given high priority and slow devices such as *keyboards* are given low priority.

When two or more devices interrupt the computer simultaneously, the computer services the device with the higher priority first.

Types of Interrupts:

Following are some different types of interrupts:

Hardware Interrupts

When the signal for the processor is from an external device or hardware then this interrupts is known as **hardware interrupt**.

Let us consider an example: when we press any key on our keyboard to do some action, then this pressing of the key will generate an interrupt signal for the processor to perform certain action. Such an interrupt can be of two types:

- **Maskable Interrupt**

The hardware interrupts which can be delayed when a much high priority interrupt has occurred at the same time.

- **Non Maskable Interrupt**

The hardware interrupts which cannot be delayed and should be processed by the processor immediately.

Software Interrupts

The interrupt that is caused by any internal system of the computer system is known as a **software interrupt**. It can also be of two types:

- **Normal Interrupt**

The interrupts that are caused by software instructions are called **normal software interrupts**.

- **Exception**

Unplanned interrupts which are produced during the execution of some program are called **exceptions**, such as division by zero.