Instruction Execution and Micro-Operations :-

A simple instruction may require:

* Instruction fetch : fetching instruction from memory
* Instruction decode : decode the instruction.
* Operand address calculation : find out the effective address of the operands
* ~~Execution : Execute the instruction~~
* Interrupt acknowledge : Perform an interrupt acknowledge cycle if an interrupt request is pending.
* Execution :- Execute the instruction.

Let us explain how these steps of instruction execution can be broken down to micro-operation.

The Fetch cycle :-

Fetch cycle, which occurs at the beginning of each instruction cycle and causes an instruction to be fetched from memory. Four registers are involved,

Memory Address Register (MAR) : It is connected to the address line of the system bus. It specifies the address in memory for read or write operation.

Memory Buffer Register (MBR) :- It is connected to the data lines of system bus. It contains the value to be stored in memory or the last value read from memory

Program Counter (PC) : Holds the address of the next instruction to be fetched

Instruction register (IR) : Holds the last instruction fetched.

Let us Look at the sequence of events for the fetch cycle.

* The first step is to move the address of next instruction to be fetched to the memory address register (MAR)
* The second step is to bring in the instruction. The desired address (in MAR) is placed on the address bus, the control unit issues a READ command on the control bus, and the result appears on the data bus and is copied into memory buffer register (MBR). We also need to increment the PC to get ready for next instruction.
* The third step is to move the contents of the MBR to the instruction register (IR).

Thus, the simple fetch cycle actually consists of three steps of four micro-operations. Symbolically, we can write this sequence of events as follows:

t_1: MAR $\leftarrow$ PC

t_2: MBR $\leftarrow$ Memory

PC $\leftarrow$ PC + 1

t_3: IR $\leftarrow$ (MBR)

Instruction Decode: This phase is performed under the control of the control unit of the computer.

Operand address calculation :-

Once an instruction is fetched, the next step is to fetch some operands. If the instruction specifies an indirect address then indirect cycle must precede the execute cycle and includes the following micro-operation.

t_1: MAR ← (IR(address))

t_2: MBR ← Memory

t_3: IR(address) ← (MBR(address))

The address field of the instruction is transferred to the MAR. This is then used to fetch the address of operand. Finally, the address field of the IR is updated from the MBR, so that it now contains a direct rather than an indirect address.

The Interrupt cycle :- The sequence of events in this cycle are

t_1: MBR ← PC

t_2: MAR ← Save_address

PC ← Routine_address

t_3: Memory ← (MBR)

In the first step, the contents of PC are transferred to the MBR, so that they can be saved for return from the interrupt. Then the MAR is loaded with the address at which contents of the PC are to be saved, and the PC is loaded with the address of the start of interrupt-processing routine. The final step is to store the MBR, which contains the old value of PC, into memory.

The Execute cycle :- The fetch, indirect, and interrupt cycles are simple and predictable. Each involves fixed sequence of micro-operation. This is not true for execute cycle. Because of the variety of opcodes, there are a number of different sequences of micro-operation that can occur.

For example, consider an add instruction

ADD R_1, X

which adds the contents of the location X to register R_1. The following sequence of micro-operations occurs:

t_1: MAR ← (IR(address))

t_2: MBR ← Memory

t_3: R_1 ← (R_1) + (MBR)

In the first step, address portion of IR is loaded into the MAR, then referenced memory location is read. Finally, the contents of R_1 and MBR are added by the ALU.

Different instructions have different sequences of micro-operations.

Instruction Pipelining :-

An instruction pipeline is a technique used in the design of computer to increase their instruction throughput (the number of instructions that can be executed in a unit of time). The basic instruction cycle is broken up into a series called a pipeline. Rather than processing each instruction sequentially, (one at a time, finishing one instruction before starting the next), each instruction is split-up into a sequence of steps so different steps can be executed concurrently (at the same time) and in parallel.

Pipelining increases instruction throughput by performing multiple operations at the same time (concurrently), but does not reduce instruction latency (the time to complete a single instruction from start to finish) as it still must go through all steps.