CONTROL UNIT OPERATIONS

MICRO- OPERATIONS

- The operation of a computer, in executing a program, consists of a sequence of instruction cycles, with one machine instruction per cycle.
- This sequence of instruction cycles is not necessarily the same as the *written sequence* of instructions that make up the program, because of the existence of branching instructions.
- What we are referring to here is the execution *time sequence* of instructions.
- We have further seen that each instruction cycle is made up of a number of smaller units. One subdivision that we found convenient is fetch, indirect, execute, and interrupt, with only fetch and execute cycles always occurring.
- Each of the smaller cycles involves a series of steps, each of which involves the processor registers. We will refer to these steps as **micro-operations**.

MICRO- OPERATIONS

- The execution of a program consists of the sequential execution of instructions.
- Each instruction is executed during an instruction cycle made up of shorter subcycles (e.g., fetch, indirect, execute, interrupt).
- The execution of each subcycle involves one or more shorter operations, that is, micro-operations.
- Micro-operationsare the functional, or atomic, operations of a processor.

The Fetch Cycle

- We begin by looking at the 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): Is connected to the address lines of the system bus. It specifies the address in memory for a read or write operation.

  Memory buffer register (MBR): Is connected to the data lines of the 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.
- We can write this sequence of events as follows:

  t1: MAR <- (PC)

  t2: MBR <- Memory

  PC <- (PC) + *I*

  t3: IR <- (MBR)

The Indirect Cycle

- Once an instruction is fetched, the next step is to fetch source operands.
- If the instruction specifies an indirect address, then an indirect cycle must precede the execute cycle.
- The following micro - operations:

  t1: MAR <- (IR(Address))

  t2: MBR <- Memory

  t3: IR(Address) <- (MBR(Address))

The Interrupt Cycle

- At the completion of the execute cycle, a test is made to determine whether any enabled interrupts have occurred. If so, the interrupt cycle occurs. The nature of this cycle varies greatly from one machine to another.

  t1: MBR <- (PC)

  t2: MAR <- Save_Address

  PC <- Routine_Address

  t3: Memory <- (MBR)

The Execute Cycle

- Because of the variety of opcodes, there are a number of different sequences of micro- operations that can occur.
- The control unit examines the opcode and generates a sequence of micro-operations based on the value of the opcode. This is referred to as instruction decoding.
- Consider an add instruction:

  ADD R1, X

  which adds the contents of the location X to register R1.
- The following sequence of micro-operations might occur:

  t1: MAR <- (IR(address))

  t2: MBR <- Memory

  t3: R1 <- (R1) + (MBR)

Control Unit Implementation

- Two approaches are used:
  - Hardwired implementation
  - Microprogrammed implementation