- **Delay Calculation for AVR:**

In Assembly Language instructions, to create a time delay one must consider two important factors.

1. **The crystal frequency –**
   The frequency of the crystal oscillator connected to XTAL1 and XTAL2 is one factor for calculating the time delay. The duration of the clock period for the instruction cycle is a function of this crystal frequency.
2. **The AVR design –**
   AVR microprocessors are able to execute an instruction in one cycle. There are three ways of doing this.
   3.
      1. Use Harvard architecture to get the maximum amount of code and data into the CPU.
      2. Use RISC architecture features such as fixed-size instructions.
      3. Use pipelining to overlap fetching and execution of instructions.

Pipelining :

In the early microprocessors, the CPU could either fetch or execute at a given time. In other words, the CPU had to fetch an instruction from the memory, then execute it then again fetch the next instruction, execute it, and so on. Pipelining allows the CPU to fetch and execute the given instruction at the same time.

We can use the pipeline to speed up execution of instructions. In pipelines, the process of execution is split up into smaller steps that are all executed in parallel. In the execution of instructions, we must make sure that the sequence of instructions is kept intact and that there is no different execution.

Instruction cycle time for the AVR :

It takes a certain amount of time for the CPU to execute an instruction. This time is referred to as machine cycles. All the instructions in the AVR are either 2-byte or 4-byte and hence most of the instructions take no more than 2 machine cycles to execute (some instructions may take up 3 to 4 machine cycles to execute). In the AVR family, the duration of the machine cycle depends upon the frequency of the oscillator connected to the AVR system. In the AVR, one machine cycle consists of one oscillator period, which means that with each oscillator clock, one machine cycle passes. Therefore, to calculate the machine cycle for the AVR, we take the inverse of the crystal frequency.

Example-1 :

For the given crystal frequencies, calculate the period of the instruction cycles.

```
a) 8 MHz    b) 16 MHz
```

```
Solution : a) instruction cycle = 1/ 8 MHz = 0.125 us (microsecond)
           b) instruction cycle = 1/ 16 MHz = 0.0625 us
```

Instruction cycles required by different instructions (considering 1 MHz as the crystal frequency) :

Instruction	Instruction cycles	Time to execute
LDI	1	1 us
DEC	1	1 us
OUT	1	1 us
ADD	1	1 us
NOP	1	1 us
JMP	3	3 us
CALL	4	4 us
BRNE	2/1	2 us if taken, 1 us if it fails

Example-2 :

Find the delay in us of the code snippet below if the crystal frequency is 10 MHz.

```
                                          Instruction Cycles
DELAY :   LDI        COUNT, 0XFF                 0
Again :   NOP                                    1
          NOP                                    1
          NOP                                    1
          DEC        COUNT                       1
          BRNE       AGAIN                      2/1
          RET                                    4


Solution :  Time Delay = [1 +(( 1+ 1+ 1+ 1 + 2 ) x 255) + 4 ] x 0.1
us = 153.5 us
```

Instruction pipelining in a pic:

Pipelining is a technique where multiple instructions are overlapped during execution. Pipeline is divided into stages and these stages are connected with one another to form a pipe like structure. Instructions enter from one end and exit from another end. Pipelining increases the overall instruction throughput.

In a pipelined computer, instructions flow through the central processing unit (CPU) in stages. For example, it might have one stage for each step of the von Neumann cycle: Fetch the instruction, fetch the operands, do the instruction, write the results. A pipelined computer usually has "pipeline registers" after each stage. These store information from the instruction and calculations so that the logic gates of the next stage can do the next step.

This arrangement lets the CPU complete an instruction on each clock cycle. It is common for even-numbered stages to operate on one edge of the square-wave clock, while odd-numbered stages operate on the other edge. This allows more CPU throughput than a multicycle computer at a given clock rate, but may increase latency due to the added overhead of the pipelining process itself. Also, even though the electronic logic has a fixed maximum speed, a pipelined computer can be made faster or slower by varying the number of stages in the pipeline. With more stages, each stage does less work, and so the stage has fewer delays from the logic gates and could run at a higher clock rate.

A pipelined model of computer is often the most economical, when cost is measured as logic gates per instruction per second. At each instant, an instruction is in only one pipeline stage, and on average, a pipeline stage is less costly than a multicycle computer. Also, when made well, most of the pipelined computer's logic is in use most of the time. In contrast, out of order computers usually have large amounts of idle logic at any given instant. Similar calculations usually show that a pipelined computer uses less energy per instruction.