Chapter 8
Counters & Time Delays

Counters & Time Delays

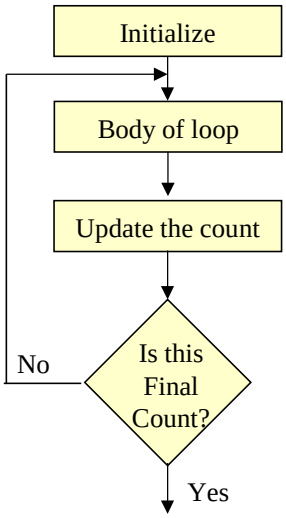
- Counters are used for keep track of events like in For loop
- Delays are used in setting up reasonably accurate timing between two events
- Counters and time delays can be implemented either on
  - Software
  - Hardware
- Both are implemented with Loop construct
- What is appropriate for counter and Time Delay
  - Counts Time duration
  - Counts Number of iteration

Counters

- A loop counter is set up by loading a register with a certain value
- Then using the DCR (to decrement) and INR (to increment) the contents of the register are updated.
- A loop is set up with a conditional jump instruction that loops back or not depending on whether the count has reached the termination count.

Counters

- The operation of a loop counter can be described using the following flowchart.



Delays

- It was shown in Chapter 2 that each instruction passes through different combinations of Fetch, Memory Read, and Memory Write cycles.
- Knowing the combinations of cycles, one can calculate how long such an instruction would require to complete.
- The table in Appendix F of the book contains a column with the title B/M/T.
  - B for Number of Bytes
  - M for Number of Machine Cycles
  - T for Number of T-State.

Delays

- Knowing how many T-States an instruction requires, and keeping in mind that a T-State is one clock cycle long, we can calculate the time using the following formula:

$$\text{Delay} = \text{No. of T-States} / \text{Frequency}$$

- For example a “MVI” instruction uses 7 T-States. Therefore, if the Microprocessor is running at 2 MHz, the instruction would require 3.5 µSeconds to complete.

Delay loops

- We can use a loop to produce a certain amount of time delay in a program.

- The following is an example of a delay loop:

```
      MVI C, FFH          7 T-States
LOOP  DCR C               4 T-States
      JNZ LOOP           10 T-States
```

- The first instruction initializes the loop counter and is executed only once requiring only 7 T-States.
- The following two instructions form a loop that requires 14 T-States to execute and is repeated 255 times until C becomes 0.

Delay Loops (Contd.)

- We need to keep in mind though that in the last iteration of the loop, the JNZ instruction will fail and require only 7 T-States rather than the 10.
- Therefore, we must deduct 3 T-States from the total delay to get an accurate delay calculation.
- To calculate the delay, we use the following formula:

$$T_{delay} = T_0 + T_L$$

  - T_{delay} = total delay
  - T_0 = delay outside the loop
  - T_L = delay of the loop

- T_0 is the sum of all delays outside the loop.

- T_L is calculated using the formula

$$T_L = T \text{ X Loop T-States X } N_{10}$$

Delay Loops (Contd.)

- Using these formulas, we can calculate the time delay for the previous example:

- T_0 = 7 T-States
  - Delay of the MVI instruction

- T_L = (14 X 255) - 3 = 3567 T-States
  - 14 T-States for the 2 instructions repeated 255 times ($FF_{16} = 255_{10}$) reduced by the 3 T-States for the final JNZ.

- T_{Delay} = (7 + 3567) X 0.5 µSec = 1.787 mSec
  - Assuming f = 2 MHz

Using a Register Pair as a Loop Counter

- Using a single register, one can repeat a loop for a maximum count of 255 times.

- It is possible to increase this count by using a register pair for the loop counter instead of the single register.
  - A minor problem arises in how to test for the final count since DCX and INX do not modify the flags.
  - However, if the loop is looking for when the count becomes zero, we can use a small trick by ORing the two registers in the pair and then checking the zero flag.

Using a Register Pair as a Loop Counter

- The following is an example of a delay loop set up with a register pair as the loop counter.

```
        LXI B, 1000H     10 T-States
LOOP    DCX B             6 T-States
        MOV A, C          4 T-States
        ORA B             4 T-States
        JNZ LOOP         10 T-States
```

Using a Register Pair as a Loop Counter

- Using the same formula from before, we can calculate:

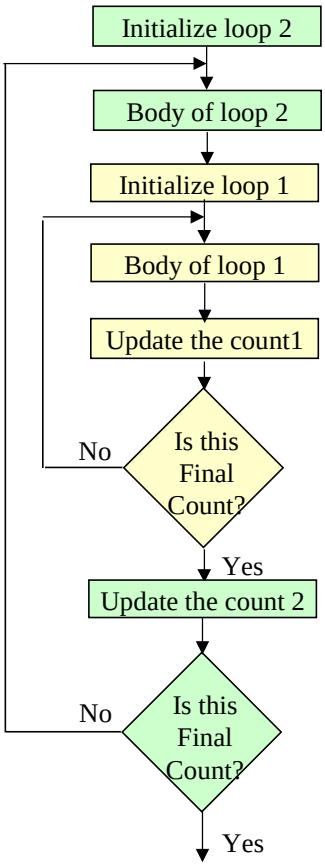
- T_0 = 10 T-States
  - The delay for the LXI instruction

- T_L = (24 X 4096) - 3 = 98301 T- States
  - 24 T-States for the 4 instructions in the loop repeated 4096 times ($1000_{16} = 4096_{10}$) reduced by the 3 T-States for the JNZ in the last iteration.

- T_{Delay} = (10 + 98301) X 0.5 mSec = 49.155 mSec

Nested Loops

- Nested loops can be easily setup in Assembly language by using two registers for the two loop counters and updating the right register in the right loop.
  - In the figure, the body of loop2 can be before or after loop1.



Nested Loops for Delay

- Instead (or in conjunction with) Register Pairs, a nested loop structure can be used to increase the total delay produced.

```
        MVI B, 10H          7 T-States
LOOP2   MVI C, FFH          7 T-States
LOOP1   DCR C               4 T-States
        JNZ LOOP1          10 T-States
        DCR B               4 T-States
        JNZ LOOP2          10 T-States
```

Delay Calculation of Nested Loops

- The calculation remains the same except that it the formula must be applied recursively to each loop.
  - Start with the inner loop, then plug that delay in the calculation of the outer loop.

- Delay of inner loop
  - T_{01} = 7 T-States
    - MVI C, FFH instruction
  - T_{L1} = (255 X 14) - 3 = 3567 T-States
    - 14 T-States for the DCR C and JNZ instructions repeated 255 times ($FF_{16} = 255_{10}$) minus 3 for the final JNZ.
  - T_{LOOP1} = 7 + 3567 = 3574 T-States

Delay Calculation of Nested Loops

- Delay of outer loop
  - T_{O2} = 7 T-States
    - MVI B, 10H instruction
  - T_{L1} = (16 X (14 + 3574)) - 3 = 57405 T-States
    - 14 T-States for the DCR B and JNZ instructions and 3574 T-States for loop1 repeated 16 times ($10_{16} = 16_{10}$) minus 3 for the final JNZ.
  - T_{Delay} = 7 + 57405 = 57412 T-States

- Total Delay
  - T_{Delay} = 57412 X 0.5 µSec = 28.706 mSec

Increasing the delay

- The delay can be further increased by using register pairs for each of the loop counters in the nested loops setup.
- It can also be increased by adding dummy instructions (like NOP) in the body of the loop.