Experiment #8

Flight86 Application I – Traffic Lights

8.0 Objectives:

The objective of this experiment is to simulate a traffic lights system.

In this experiment, you will do the following:

- Create software time delays
- Write programs to simulate a traffic lights system
- Assemble, download, and test your program on the trainer board

8.1 Equipment and Software

- Flight86 Trainer and Application Boards
- PC with Flight86 Monitor program
- Assembler and conversion utilities (exe2bin, bin2hex)

8.2 Introduction:

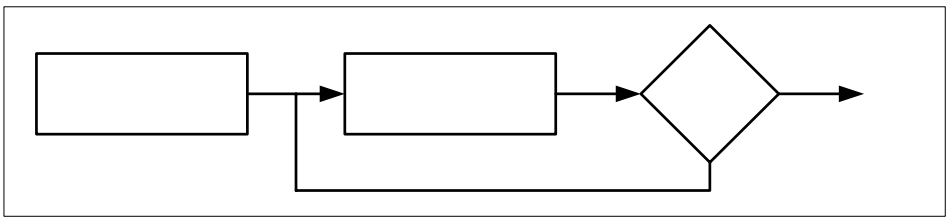
It is often necessary to control how long certain actions last, this can be achieved using software delays, or more accurately by the use of a timer.

In this experiment we will simulate a traffic lights system that requires use of software time delay.

8.2.1 Creating Software Delays

In the various states of the traffic lights sequence, lights have to be ON or OFF for a clearly defined time in seconds, so our program must contain a means of measuring one second. The easiest way, which does not need any further hardware devices, is a software delay.

If we create a program that loops around itself, and does this for a fixed number of times, for a given processor, running at a given clock rate, this process will always take the same time. All we have to do is write such a multiple loop so that it takes one second to complete. This process is illustrated in the flowchart below:



Now the question is: how do we calculate the 'large number' to be loaded in the register for the loop?

To calculate a specific time delay we need to calculate the number of times the program will loop around itself. To do this, we need to know how many clock cycles are required to carry out a particular instruction(s), and the processor clock rate which ultimately decides how long an instruction takes to execute.

Let's examine the code below. This code can be used to produce a certain delay value. We will try to find the value of N such that this code produces a delay of approximately 100ms.

```
DELAY:  MOV CX, N       ; Load CX with a fixed value
DEL1:   DEC CX          ; decrement CX
        JNZ DEL1        ; and loop if not zero
        RET             ; when CX=0, then exit
```

We can see in the code above that instructions which get repeatedly executed (inside the loop) are **DEC CX** and **JNZ DEL1**. The number of clock cycles required to execute these two instructions once, are:

DEC CX	2 clock cycles
JNZ DEL1	16 clock cycles
Total :	18 clock cycles

Note: If we also consider the time required to execute the instructions outside the loop, i.e., the first and the last instructions above, then we can get a more accurate time delay. But we will ignore this, since these instructions are executed only once.

Number of times this loop is executed is: **N**
CPU clock rate for the Flight86 Trainer system is: **CLK = 4.9152 MHz**
Time period for one clock cycle, $\mathbf{T_{CLK}} = 1/(4.9152 \times 10^6) =$ **203.5ns**

(Total clock cycles) x (Number of times loop is executed) x T_{CLK} = 100 ms

18 x N x 203.5ns = 100 ms; ⟶ Solving for N, we get **N = 27300** = 6AA4H

Therefore, if the above loop is executed N = 27300 times we get a delay of 100ms approximately. Now, this loop can be used to produce a delay of **1 second** if it is executed 10 times. That means, there will be two loops – one that produces 100 ms delay, and the other that executes this loop 10 times to produce 10 x 100ms = 1 second.

8.3 Exercise: Simulating a Traffic lights system

The LED's on the application board are arranged in two groups of 4 - Red, Amber, Green, and a Yellow. Using these two sets of four lights we can easily simulate the traffic lights at a busy cross road, one set representing the main road, the other set the side road.

Red1	Amber1	Green1	Yellow1	Red2	Amber2	Green2	Yellow2
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

The traffic lights system must be simulated according to the following sequence which must be **repeated** continuously (the Yellow LED's are not used):

Main Road		Side Road								
Signal	Duration	Signal	Duration	Bit value							
Red 1	15 sec	Green 2	15 sec	1	0	0	0	0	0	1	0
Red 1	03 sec	Amber 2	03 sec								
Red & Amber 1	03 sec	Red 2	03 sec								
Green 1	25 sec	Red 2	25 sec								
Amber 1	03 sec	Red 2	03 sec								
Red 1	03 sec	Red & Amber 2	03 sec								

Table 1: Sequence of lights

The bit value represents the output pattern to turn on the required LED's.

An easy way to implement the above sequence of lights repetitively is to set all the data up in a table, and advance through the data step by step, until the end of the table is reached, when it can be repeated. The table can contain both the required LED pattern, and the time that pattern is to be manipulated. For the above case, the table would be:

Data, time
82h,150

LED pattern 82h (10000010), maintained for 150 x 100ms (= 15 sec).

The delay is implemented in multiples of 100 ms.

Table 2: Data for traffic lights sequence

8.4 Review

1. Review the hardware specifications of the Flight86 system described in the last experiment.
2. Read about instruction clock cycles from your text book.

8.5 Pre-lab:

1. Complete the bit-value output pattern in Table 1.
2. Complete Table 2 for the corresponding bit-value pattern in Table 1.

   Table 2 can be implemented in assembly language using the "DB" (define byte) assembler directive, as shown below:

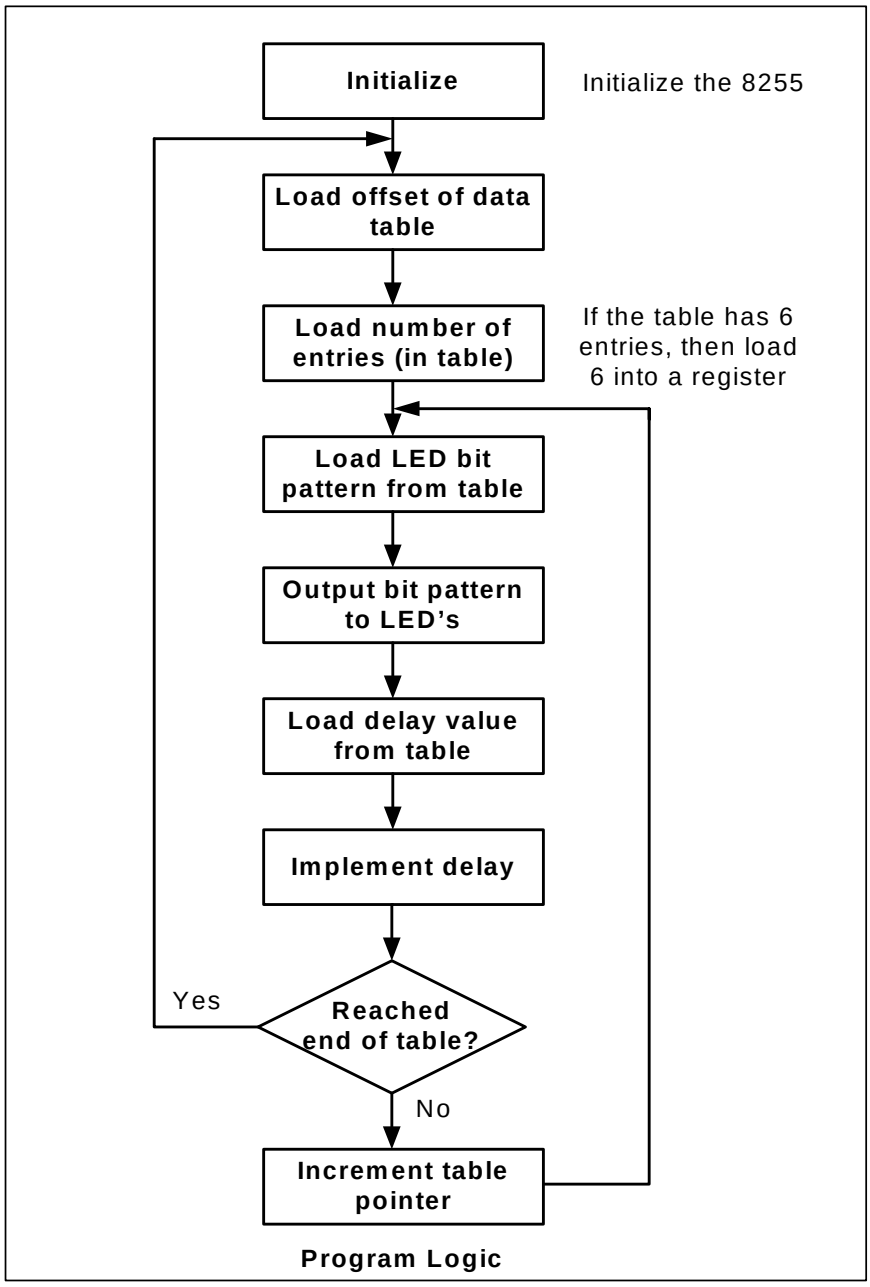
```
TABLE   DB 82h, 150     ; RED1  GREEN2          15 sec
        DB      ,       ; RED1  AMBER2          03 sec
        DB      ,       ; RED/AMBER1 RED2       03 sec
        DB      ,       ; GREEN1 RED2           25 sec
        DB      ,       ; AMBER1 RED2           03 sec
        DB      ,       ; RED1 RED/AMBER2       03 sec
```

3. Complete the remaining entries of this table and place it after the last instruction of your program that you will write for the Lab Work.
4. Write a program to produce a delay of 5 seconds using the code shown below which produces a delay of 100ms.

```
DELAY:  MOV CX, 27300   ; Load CX with a fixed value
DEL1:   DEC CX          ; decrement CX
        JNZ DEL1        ; and loop if not zero
        RET             ; when CX=0, then exit
```

8.6 Lab Work:

1. Write your program according to the flow chart shown below.



2. Include the following initialization code at the beginning of your program. (See previous experiment).

```
INIT:
        MOV AL, 99          ; set up Port A & Port C IN, Port B OUT
        OUT 07, AL          ; and output this word  to control Port
        MOV AL,0            ; data zero
        OUT 03,AL           ; output to Port B to ensure all LED's off
```

3. Use an editor to write to your program. Name your file as ***traffic.asm***
4. Assemble and link your program using the TASM assembler to produce ***traffic.exe***.
5. Convert the ***traffic.exe*** file to binary format using the exe2bin.exe program by typing ***exe2bin traffic.exe*** at the DOS prompt.
6. Convert the ***traffic.bin*** file ***to traffic.hex*** using the bin2hex.exe program by typing ***bin2hex traffic.bin*** at the DOS prompt.
7. Start the Flight86 monitor program.
8. Download ***traffic.hex*** to the Flight86 controller board by typing at the '-' prompt

   : C:\EE390\FLIGHT86\traffic.hex

9. Before you run your program make sure the mode switches are in the correct position.

Switch SW2A – Switch position
All other switches - OFF

10. Now enter **G 0050:0100** at the '-' prompt to run the program.
11. Check the sequence of lights and the time duration generated by your program and make sure it is working correctly.