1. Why “stopTime” is defined as “DWORD32”?
Because the stopTime is being used to assign and compare with Now_Plus() and Millisecond() function, which both data type are DWORD32 (unsigned long int). Therefore, stopTime is defined as DWORD32 is reasonable to prevent wasted of the system resources.

2. Why “finalCount” is defined as “QWORD64”?
The same reason with stopTime, finalCount is being used to assign from CPU_Clock_Cycles (), which data type is QWORD64 (unsigned long long int).

3. What does “Now_Plus()” do?
Calculates and returns the 32 bit unsigned count that will be in the system timer some number of seconds in the future, specified by “seconds”. The timer runs at 1000 ticks per second.

4. What does “CPU_Clock_Cycles()” do?
Returns a 64 bits count of CPU cycles since the processor was reseted.

5. What does “x” (or similarly “xM”) represent?
It represent elapsed time of the counter

6. What does “Milliseconds()” do?
Return the 32 bits unsigned count of milliseconds that have elapsed since program execution began.

7. Compare numbers printed by section 2 and sections 3 of the code

7.1. Are they different? -- Yes

7.2. Should they be similar? -- No

7.3. How do you explain that these two sections may report different values?

Milliseconds () show the value from the execution time of the program.

Now_Plus () show the value when the function is being call.

7.4. Which approach (section 2 or 3) is more appropriate? Why?

Section 3 is more appropriate since the Milliseconds take the time at the execution of the program.

8. Section 4 prints two different numbers. What causes their difference?

Because the second line print the value of Milliseconds() + 2345. Then the values are different. Furthermore, while running section 4, some code lines (instructions) will cause time to execute, then it lead to the 1 milliseconds more.

9. Consider an operation (e.g., multiplication, addition, ...) and approximate its execution time. A technique is to perform multiple of the selected operation in a loop and divide the overall time by the number of executions it had in the loop.

Based on the section 1 and 2, the CPU frequency is 2.7 GHz. And we using the CPU_Clock_Cycles () function to measure how many cycles it take to done the operation (addition).

At first, we calculate the program without addition operation

```
//+++++ Question 9. +++++
initialCount = CPU_Clock_Cycles();

// for (int i = 0; i <= 1000; i++){
//   a = a + b;
// }

finalCount = CPU_Clock_Cycles();
x = finalCount - initialCount;
SetCursorPosition(5, 1);
PutString("Time elapsed for addition: ");
PutUnsigned(x, 10, 8);
PutString("\n");

return 0;
}
```

The number of cycles is about 12500 cycles

Secondly, we put the addition operation back

```
//+++++ Question 9. +++++
initialCount = CPU_Clock_Cycles();

for (int i = 0; i <= 1000; i++){
  a = a + b;
}

finalCount = CPU_Clock_Cycles();
x = finalCount - initialCount;
SetCursorPosition(5, 1);
PutString("Time elapsed for addition: ");
PutUnsigned(x, 10, 8);
PutString("\n");

return 0;
```

The number of cycles of 1000 Iteration about 83580 cycles

Therefore, it takes around (83580 - 12500) / 1000 = 71 cycles. Then 71/ 2.7 x 10^9 = 26.3 nanoseconds.

As a result, it takes 26.3 nanoseconds for a addition operation.