Homework 3

Amin Parchami

After visiting the wonderland, this time Alice is going back to school for the first time in the new year. It’s her first homework assignment and she gives each of you a task to help her.

Your task is to compute how many times a two digit binary code (00, 01, 10, or 11, depending on your student number, look at the table at the end of this document) occurs in each element of a sequence of numbers, and perform consequent computations as detailed below.

As input, your code receives a sequence of (at most 100) positive integers followed by a zero indicating end of the input.

You must count the number of occurrences of your binary code in the binary representation of each number and store them in the memory.

For example, the number of occurrences for each “00” and “01” in number 9 is equal to 4 and 2, respectively:

```
9 = 00001001
00: 00001001, 00001001, 00001001, 00001001
01: 00001001, 00001001
```

You must store the number of occurrences of each number in an array in the memory (.data or .bss segments) in the same order as the input. Then you need to multiply each entry of the array to its opposite entry from the end of the array and sum up the resulting numbers. In other words, assuming you have stored n numbers with indices $0, 1, \dots, n-1$, you have to calculate the sum of multiplying each number at index i with the one at index $n-i-1$.

For example, if the number of the occurrences are: 3,1,0,5, then the answer will be 3 * 5 + 1 * 0 = 15.

Input:

The input contains a sequence of at most 100 positive integers followed by a 0 indicating the end of input. All input numbers fit into 1 byte. The ending zero is not counted as input.

Output:

At the first line, your code must print the exact phrase: “**`The answer is:`**“ without using the **print_string** function.
At the second line, you must print your answer.
Your code **must** comply with the following rules:

- You can only use the EAX, EBX, ECX, EDX, ESI and EDI registers.
- You must use the **read_int**, **print_int** and also **print_char** functions (from the textbook) for I/O. You are **not allowed** to use the **print_string** function**.**
- You can only use the commands you have learned so far in the class.
- You must not print extra outputs.

Please upload only the “.asm” file on courses.kntu.ac.ir**.**

Your code will be checked for similarity. In the case of cheating, the student will receive a **negative** point. It is **your responsibility** to protect your code.
Your code **must** be able to run using the following **driver.c** file.

```
void asm_main();
int main() {

    asm_main();
    return 0;
}
```

Example 1: (looking for “11”)

Input:

7 11 14 1 3 0

Output:

The answer is:

6

Solution:

7: 00000111 → 00000111 → **2**

11: 00001011 → 00001011 → **1**

14: 00001110 → 00001110 → **2**

1: 00000001 → 00000001 → **0**

3: 00000011 → 00000011 → **1**

2 * 1 + 1 * 0 + 2 * 2 = 6

Notice that the 0 at the end of the input is not counted as an input number.

Example 2: (Looking for “01”)

Input:

4 6 5 21 0

Output:

The answer is:

5

Solution:

4: 00000100 → 00000100 → **1**

6: 00000110 → 00000110 → **1**

5: 00000101 → 00000101 → **2**

21: 00010101 → 00010101 → **3**

1 * 3 + 1 * 2 = 5

The two digit binary code for each student ID.

Student ID	Binary Code	Student ID	Binary Code
9213773	11	9628973	10
9322943	11	9629063	10
9426293	01	9629183	01
9427773	01	9629433	11
9523003	01	9629463	10
9524623	11	9629613	11
9526923	01	9629853	10
9526983	10	9630003	01
9624443	01	9630293	01
9624533	11	9630393	11
9624803	11	9630513	01
9625113	01	9630623	11
9625123	01	9630683	01
9625543	11	9631063	01
9625903	10	9631383	11
9626143	11	9631723	01
9626743	11	9631773	11
9626873	01	9631793	01
9627123	11	9631913	01
9627133	10	9632073	11
9627173	01	9642633	10
9627193	01	9625773	10
9627343	10	9626853	01
9627353	10	9629273	01
9628053	10	9630463	11
9628063	11	9631663	01
9628143	11	9631993	01
9628433	01	9525173	01
9628473	01	9427773	10
9628733	01		