Notes:

1. Read **ALL** of the given notes.
2. Save the program in a file named **zad.S**. This is the only file that will be reviewed.
3. Write your **name**, **last name** and **your index number (student ID)** as a comment at the beginning of the file. If you do not write this information, your solution will not be graded.
4. Solutions which fail to compile will be graded with **0** points.
5. Make sure to write comments.
6. Use the practicum from the directory *ispitni_materijali* (**do not** make a copy in your home directory).
7. You are not allowed to take a brake during the test. Leaving the classroom means you have finished the test.
8. During the test, teaching assistants can only provide information about the text of the assignment and help with the usage of programming tools.
9. The zad.S file should contain only the code of the subroutine, **without the data section**. If more variables are needed, use local variables.
10. **The classroom is under video surveillance.**
11. The test lasts 2 hours and 15 minutes.

Assignment:

Write an assembly subroutine that inverts the bits of the elements of an array of unsigned 32-bit integers. Each element should be analyzed starting from the least-significant bit (bit at position 0). If the currently checked bit is set to 1, invert the first bit to its left (the first bit at a higher position). Otherwise, continue to the next bit and check its value. For example, if the bit at position 0 is set to 1, the bit at position 1 should be inverted. If the bit in position 0 is set to 0, the bit in position 1 should not be inverted. The analogy applies to the remaining bits of each value in the string. The subroutine declaration is:

int BitSwitch(unsigned int* vrednosti, int n)

The first parameter of the subroutine is the address of an array of unsigned 32-bit values, and the second is the number of elements in the array. The return value of the routine is the total number of inverted bits across all the elements of the array. The array can have a maximum of 10 elements, and it can also be empty.

Example:

For an array of values of 3 elements {0x0, 0x1, 0x555} (the first row of the table) whose binary representations are shown in the second row of the table, the result is shown in the third row.

Hexadecimal	`0x0`	`0x1`	`0x555`
Binary	`0000000000000000` (the higher 16 bits are 0)	`0000000000000001` (the higher 16 bits are 0)	`0000010101010101` (the higher 16 bits are 0)
Result (the lower 16 bits are shown)	`0000000000000000` (the higher 16 bits are 0)	`1111111111111111` (the higher 16 bits are 0)	`0000001100110011` (the higher 16 bits are 0)

For more examples, run the provided test examples. Apart from the given test examples, the program will be tested with additional examples during grading, so the program should be tested with various inputs.

The maximum number of points on this test is 20.