```
1: # lab 3 exercise 2
2:     .data
3: prompt: .asciiz "Enter an integer:\n"
4: output_message: .asciiz "Binary representation: "
5: newline: .asciiz "\n"
6: bit_string: .space 33  # 32 bits + null
7:
8:
9: .text
10:
11:     # Prompt user for input
12:     li $v0, 4
13:     la $a0, prompt
14:     syscall
15:
16:     # Read integer input and store in $t0
17:     li $v0, 5
18:     syscall
19:     move $t0, $v0
20:
21:
22:     la $t1, bit_string  # Pointer to bit string
23:     li $t2, 0x80000000  # Mask
24:     li $t3, 32  # loop counter
25:
26: bit_loop:
27:     # Isolate  MSB,  AND operation
28:     and $t4, $t0, $t2  # t4 = t0 & mask
29:
30:     # Store 1 or 0 in bit string
31:     li $t5, 1
32:     beqz $t4, store_zero  # if t4 = 0, store 0
33:     sb $t5, 0($t1)        # Store 1
34:     j continue_loop
35:
36: store_zero:
37:     li $t5, 0
38:     sb $t5, 0($t1) # Store 0
39:
40: continue_loop:
41:     addi $t1, $t1, 1  # move to next character in string
42:     srl $t2, $t2, 1   # Shift mask to right
43:     subi $t3, $t3, 1  # dec counter
44:     bnez $t3, bit_loop  # Repeat until all 32 bits processed
45:
46:     # Null-terminate bit string
47:     li $t5, 0
48:     sb $t5, 0($t1)
```

```
49:
50:     #print
51:     li $v0, 4
52:     la $a0, output_message
53:     syscall
54:
55:
56:     li $v0, 4
57:     la $a0, bit_string
58:     syscall
59:
60:
61:     li $v0, 4
62:     la $a0, newline
63:     syscall
64:
65:     # quit
66:     li $v0, 10
67:     syscall
```