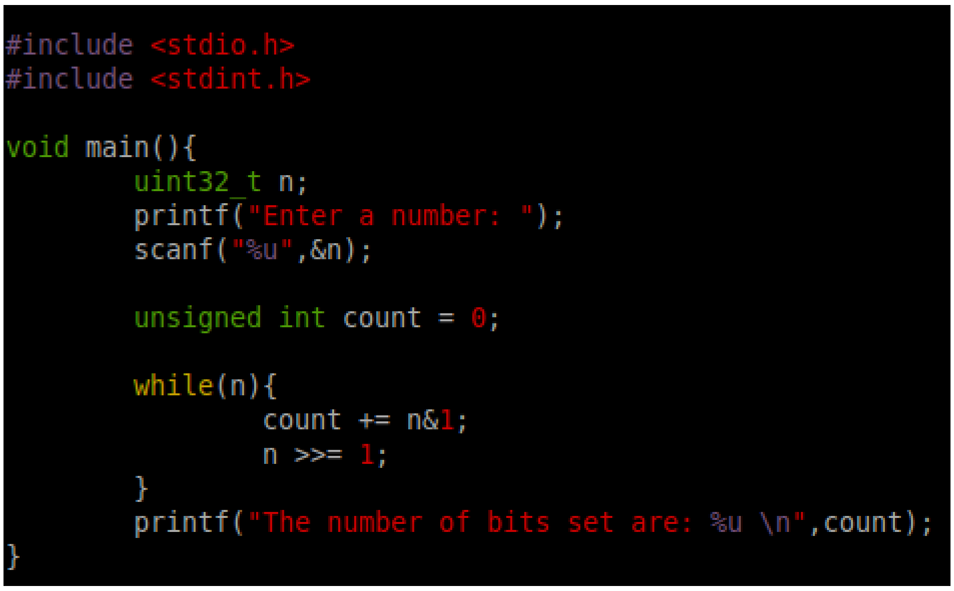
CSN – 221 Coding Project 2

GroupID: 08
ProblemNo: 57

Members:
Rohith A.S.R.K. (15116044)
Shashwat Kumar (15116053)

The **code** required are available in the following files,
1. '**code.c**' – the c code of the problem
2. '**code.s**' – the output of the following command 'gcc -S code.c', i.e., the assembly generated by gcc.
3. '**code.asm**' – the assembly language program written in MIPS 32-bit ISA for execution in QtSPIM.

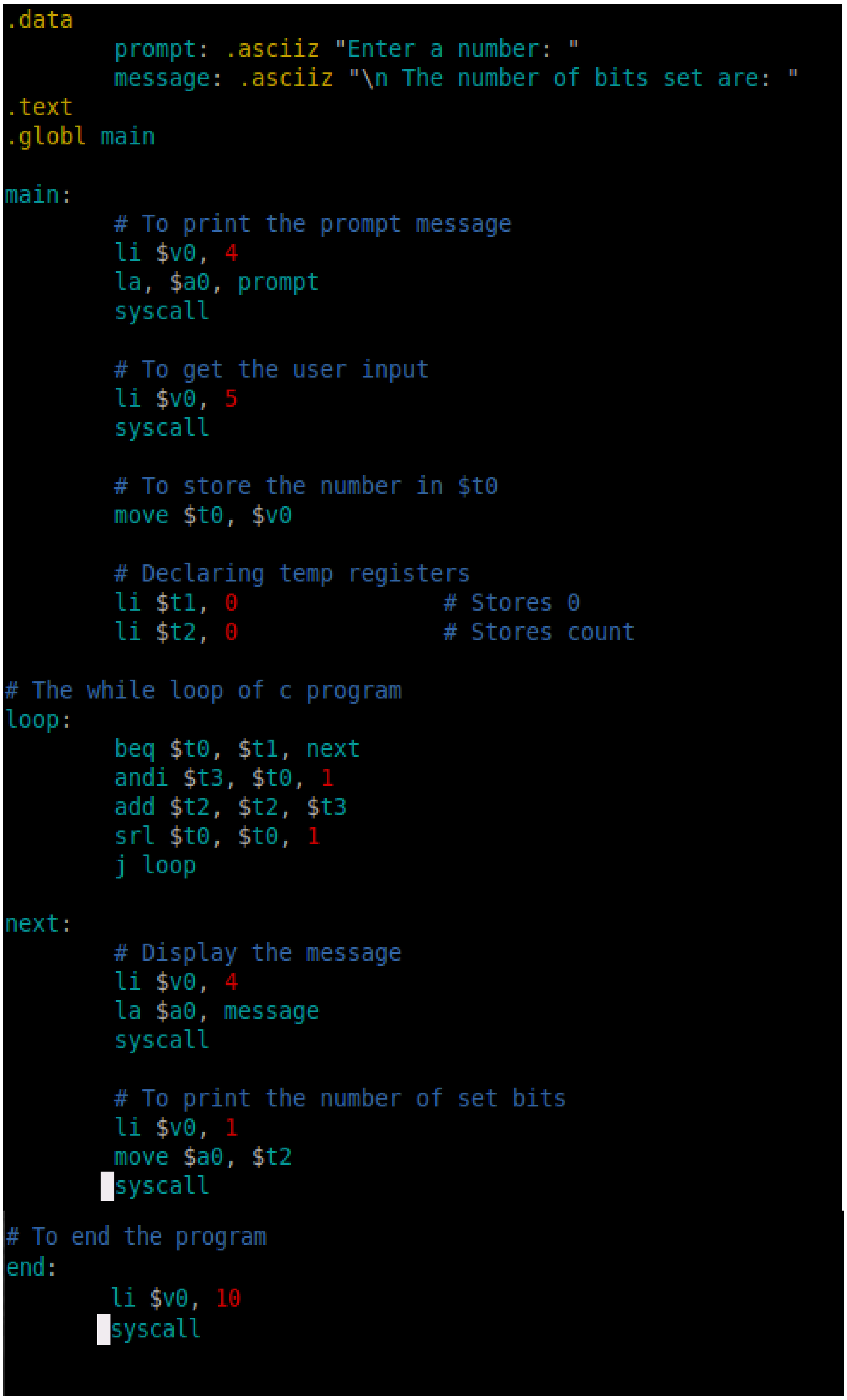
The code snipped is as shown in the image. The image has been inserted because of low readability of plain text snippets.



```c
#include <stdio.h>
#include <stdint.h>

void main(){
	uint32_t n;
	printf("Enter a number: ");
	scanf("%u",&n);

	unsigned int count = 0;

	while(n){
		count += n&1;
		n >>= 1;
	}
	printf("The number of bits set are: %u \n",count);
}
```

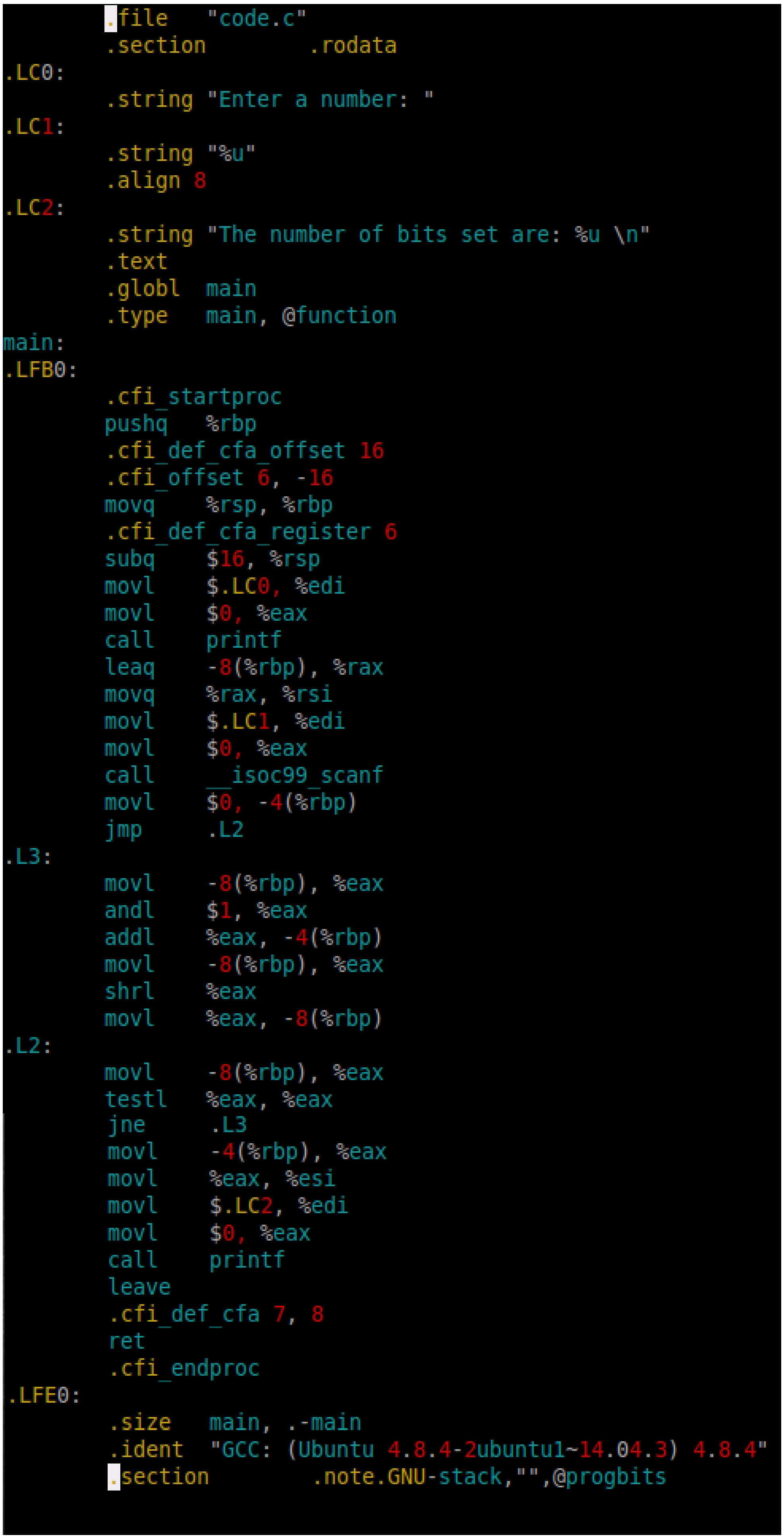
(P.T.O)

The assembly language code snippet written in MIPS 32-bit ISA is as follows,



```
.data
        prompt: .asciiz "Enter a number: "
        message: .asciiz "\n The number of bits set are: "
.text
.globl main

main:
        # To print the prompt message
        li $v0, 4
        la, $a0, prompt
        syscall

        # To get the user input
        li $v0, 5
        syscall

        # To store the number in $t0
        move $t0, $v0

        # Declaring temp registers
        li $t1, 0                # Stores 0
        li $t2, 0                # Stores count

# The while loop of c program
loop:
        beq $t0, $t1, next
        andi $t3, $t0, 1
        add $t2, $t2, $t3
        srl $t0, $t0, 1
        j loop

next:
        # Display the message
        li $v0, 4
        la $a0, message
        syscall

        # To print the number of set bits
        li $v0, 1
        move $a0, $t2
        syscall

# To end the program
end:
        li $v0, 10
        syscall
```

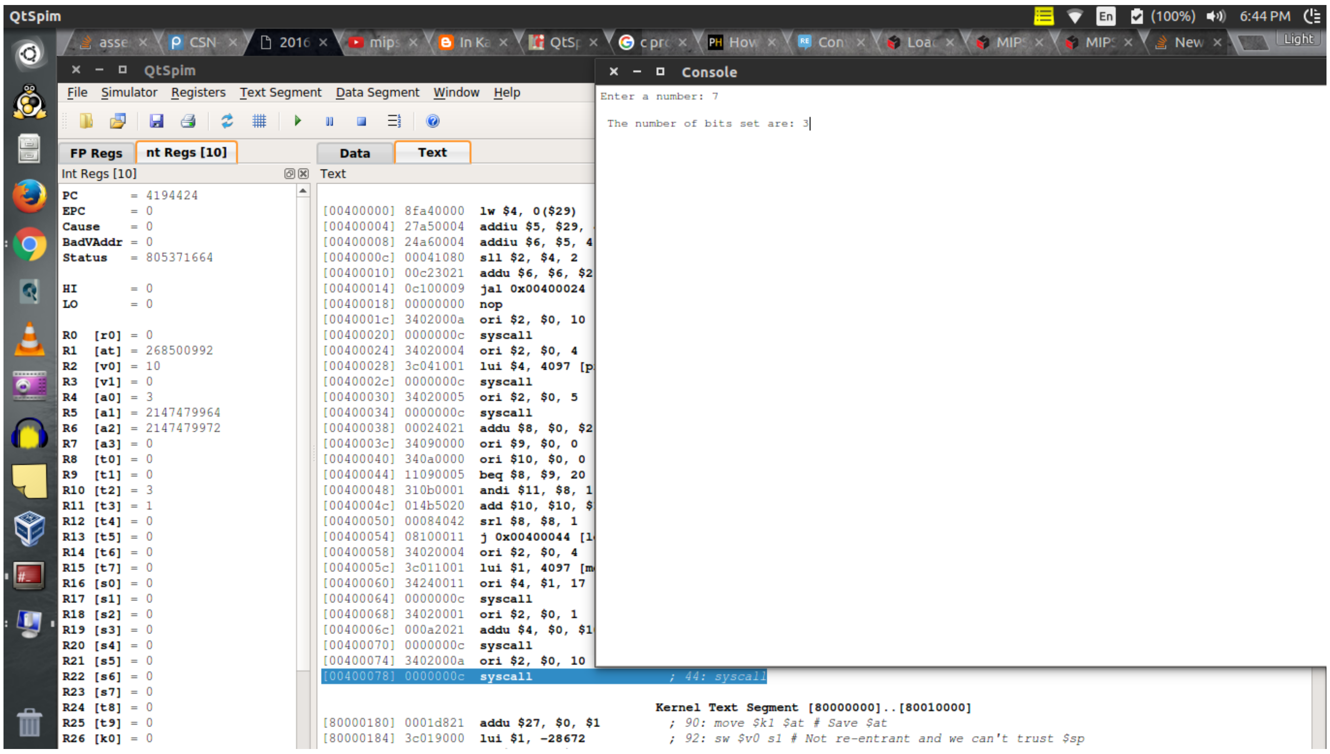
The output assembly of gcc is as follows,



```
        .file   "code.c"
        .section        .rodata
.LC0:
        .string "Enter a number: "
.LC1:
        .string "%u"
        .align 8
.LC2:
        .string "The number of bits set are: %u \n"
        .text
        .globl  main
        .type   main, @function
main:
.LFB0:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $16, %rsp
        movl    $.LC0, %edi
        movl    $0, %eax
        call    printf
        leaq    -8(%rbp), %rax
        movq    %rax, %rsi
        movl    $.LC1, %edi
        movl    $0, %eax
        call    __isoc99_scanf
        movl    $0, -4(%rbp)
        jmp     .L2
.L3:
        movl    -8(%rbp), %eax
        andl    $1, %eax
        addl    %eax, -4(%rbp)
        movl    -8(%rbp), %eax
        shrl    %eax
        movl    %eax, -8(%rbp)
.L2:
        movl    -8(%rbp), %eax
        testl   %eax, %eax
        jne     .L3
        movl    -4(%rbp), %eax
        movl    %eax, %esi
        movl    $.LC2, %edi
        movl    $0, %eax
        call    printf
        leave
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE0:
        .size   main, .-main
        .ident  "GCC: (Ubuntu 4.8.4-2ubuntu1~14.04.3) 4.8.4"
        .section        .note.GNU-stack,"",@progbits
```

The screenshots below are the ones for a couple of test cases,

1. The number first entered was 7 which is 111 in binary. It has 3 set bits, which is the output.

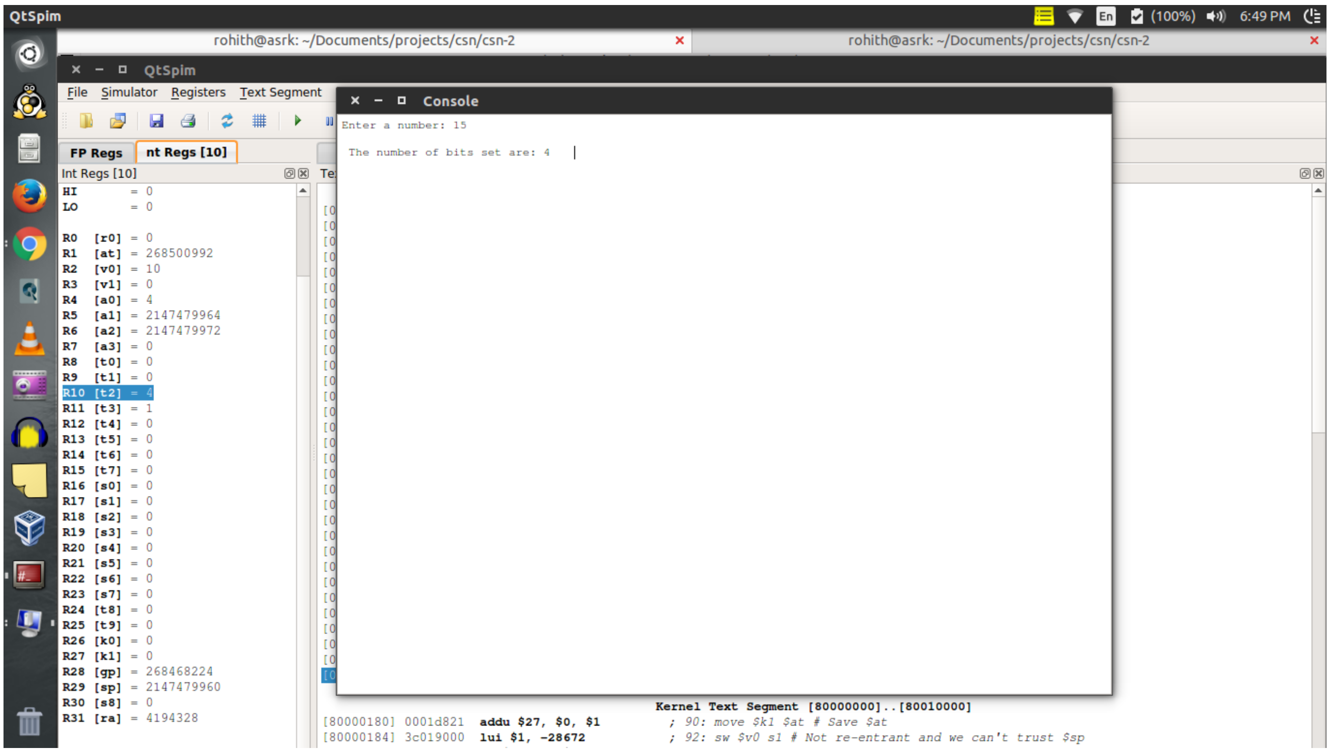


```
Int Regs [10]
PC       = 4194424
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 3
R5  [a1] = 2147479964
R6  [a2] = 2147479972
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 0
R10 [t2] = 3
R11 [t3] = 1
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
```

```
Text
[00400000] 8fa40000  lw $4, 0($29)
[00400004] 27a50004  addiu $5, $29,
[00400008] 24a60004  addiu $6, $5, 4
[0040000c] 00041080  sll $2, $4, 2
[00400010] 00c23021  addu $6, $6, $2
[00400014] 0c100009  jal 0x00400024
[00400018] 00000000  nop
[0040001c] 3402000a  ori $2, $0, 10
[00400020] 0000000c  syscall
[00400024] 34020004  ori $2, $0, 4
[00400028] 3c041001  lui $4, 4097 [p
[0040002c] 0000000c  syscall
[00400030] 34020005  ori $2, $0, 5
[00400034] 0000000c  syscall
[00400038] 00024021  addu $8, $0, $2
[0040003c] 34090000  ori $9, $0, 0
[00400040] 340a0000  ori $10, $0, 0
[00400044] 11090005  beq $8, $9, 20
[00400048] 310b0001  andi $11, $8, 1
[0040004c] 014b5020  add $10, $10, $
[00400050] 00084042  srl $8, $8, 1
[00400054] 08100011  j 0x00400044 [l
[00400058] 34020004  ori $2, $0, 4
[0040005c] 3c011001  lui $1, 4097 [m
[00400060] 34240011  ori $4, $1, 17
[00400064] 0000000c  syscall
[00400068] 34020001  ori $2, $0, 1
[0040006c] 000a2021  addu $4, $0, $1
[00400070] 0000000c  syscall
[00400074] 3402000a  ori $2, $0, 10
[00400078] 0000000c  syscall                      ; 44: syscall

                                    Kernel Text Segment [80000000]..[80010000]
[80000180] 0001d821  addu $27, $0, $1        ; 90: move $k1 $at # Save $at
[80000184] 3c019000  lui $1, -28672          ; 92: sw $v0 s1 # Not re-entrant and we can't trust $sp
```

```
Console
Enter a number: 7

 The number of bits set are: 3
```

2. Then 15 was taken as a test case. Which is 1111 in binary and has 4 bits set, which again was the output. (As it can be seen from the figure.)



```
Int Regs [10]
HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 4
R5  [a1] = 2147479964
R6  [a2] = 2147479972
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 0
R10 [t2] = 4
R11 [t3] = 1
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 268468224
R29 [sp] = 2147479960
R30 [s8] = 0
R31 [ra] = 4194328
```

```
Console
Enter a number: 15

 The number of bits set are: 4
```

```
                                    Kernel Text Segment [80000000]..[80010000]
[80000180] 0001d821  addu $27, $0, $1        ; 90: move $k1 $at # Save $at
[80000184] 3c019000  lui $1, -28672          ; 92: sw $v0 s1 # Not re-entrant and we can't trust $sp
```

Thank you