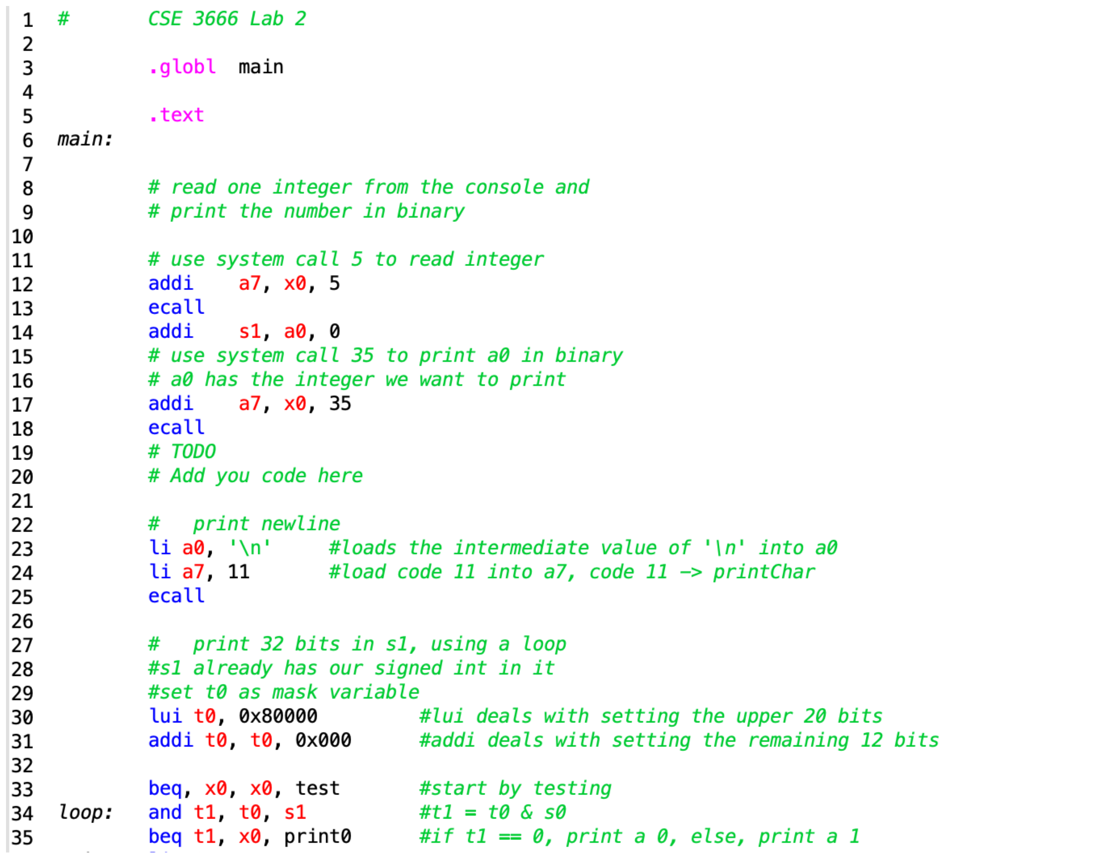
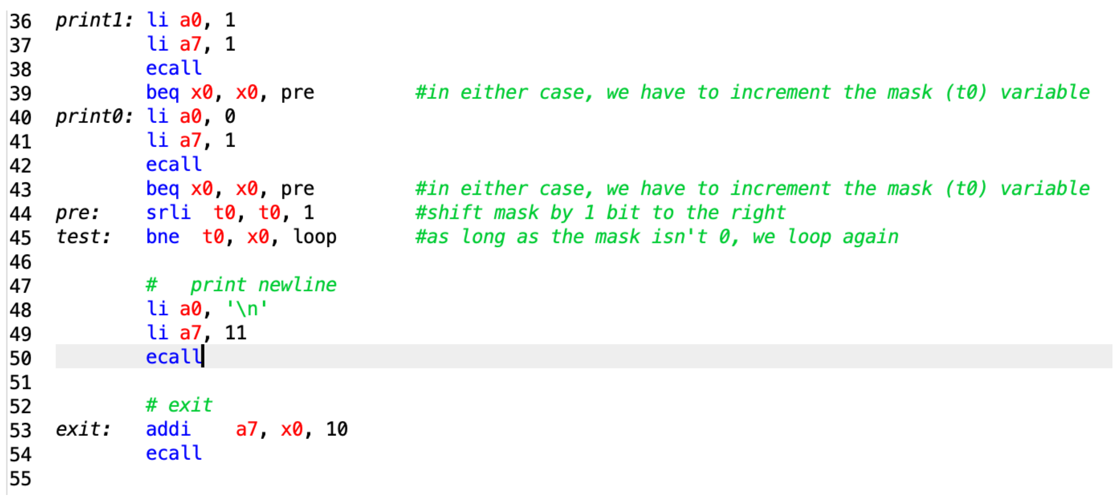
CSE 3666 - Lab 2

Jonathan Ameri



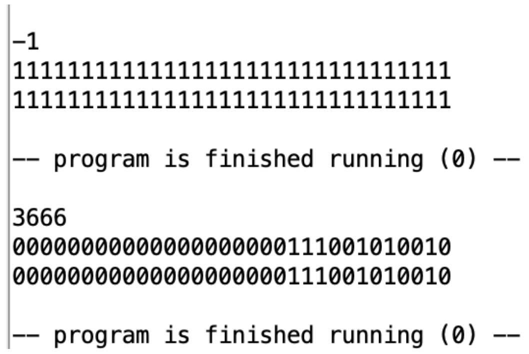
```
#       CSE 3666 Lab 2

        .globl  main

        .text
main:

        # read one integer from the console and
        # print the number in binary

        # use system call 5 to read integer
        addi    a7, x0, 5
        ecall
        addi    s1, a0, 0
        # use system call 35 to print a0 in binary
        # a0 has the integer we want to print
        addi    a7, x0, 35
        ecall
        # TODO
        # Add you code here

        #   print newline
        li a0, '\n'     #loads the intermediate value of '\n' into a0
        li a7, 11       #load code 11 into a7, code 11 -> printChar
        ecall

        #   print 32 bits in s1, using a loop
        #s1 already has our signed int in it
        #set t0 as mask variable
        lui t0, 0x80000         #lui deals with setting the upper 20 bits
        addi t0, t0, 0x000      #addi deals with setting the remaining 12 bits

        beq, x0, x0, test       #start by testing
loop:   and t1, t0, s1          #t1 = t0 & s0
        beq t1, x0, print0      #if t1 == 0, print a 0, else, print a 1
```



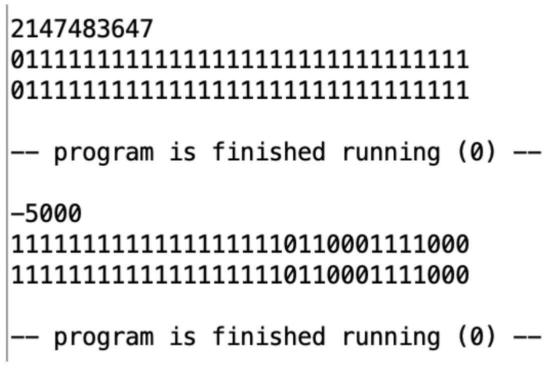
```
print1: li a0, 1
        li a7, 1
        ecall
        beq x0, x0, pre         #in either case, we have to increment the mask (t0) variable
print0: li a0, 0
        li a7, 1
        ecall
        beq x0, x0, pre         #in either case, we have to increment the mask (t0) variable
pre:    srli  t0, t0, 1         #shift mask by 1 bit to the right
test:   bne  t0, x0, loop       #as long as the mask isn't 0, we loop again

        #   print newline
        li a0, '\n'
        li a7, 11
        ecall

        # exit
exit:   addi    a7, x0, 10
        ecall
```

Describe the results of the code. Does the code work or only work for some inputs? Include a screenshot of "Run I/O" tab showing a few runs of the code.

The code works for all decimal numbers that can be represented with 32 bits. The largest integer that you could use would be 2,147,483,649. The code works with both positive and negative numbers.



```
-1
11111111111111111111111111111111
11111111111111111111111111111111

-- program is finished running (0) --

3666
00000000000000000000111001010010
00000000000000000000111001010010

-- program is finished running (0) --
```



```
2147483647
01111111111111111111111111111111
01111111111111111111111111111111

-- program is finished running (0) --

-5000
11111111111111111110110001111000
11111111111111111110110001111000

-- program is finished running (0) --
```