```
CS 2734, Org II, Spring 2002
First Exam, Answers
========================================================================
1.  (a) 46 (base 10) = 101110 (base 2)
       -46 (base 10 = ~101110 + 1 = 1111 1111 1101 0001 + 1 =
                                    1111 1111 1101 0010
    (b) 1011 1111 1101 1100  (48 more 0's)  (binary) =

        0 01111111101  1100  (48 more 0's)  (broken into fields) =

        (-1)^Sign x (1 + Significand) x 2^(Exponent - Bias) =
        (-1)^1    x (1 + 0.750000000) x 2^(1021     - 1023) =
        (-1)      x 1.75              x 2^(-2)              =
           -(7/4)x(1/4) = -7/16 = -0.4375
========================================================================
2.
# Answer to Exam 1, Problem 2
        .globl  main
main:   add     $s7, $0, $ra    # save return address

        .data
  A:    .word  4, 9, 25, 49, 121, 169, 289  # squares of first 7 primes
        .text
################# Answer to Problem 2 ##########################
        la      $s1, A          # start address of A
        addi    $s2, $0, 0      # running sum
        addi    $s3, $0, 0      # array index of A
        addi    $s4, $0, 7     # constant 10

 Loop:  lw      $t1, 0($s1)     # $t1 = A[$s3]
        add     $s2, $s2, $t1   # $s2 = sum of A[] so far
        addi    $s1, $s1, 4     # $s1 += 4
        addi    $s3, $s3, 1     # $s3 += 1
        bne     $s3, $s4, Loop  # brach back to Loop until $s4 == 10

        addi    $v0, $0, 1      # print the sum
        add     $a0, $0, $s2
        syscall
################# End of Answer to Problem 2 ###################
        addi    $v0, $0, 4      # print a newline
        la      $a0, Newln
        syscall

        add     $ra, $0, $s7    # restore return address
        jr      $ra
        .data
Newln:  .asciiz "\n"
################# Output #######################################
# ten42% spim -file exam1_2.s
# SPIM Version 6.0 of July 21, 1997
# Copyright 1990-1997 by James R. Larus (larus@cs.wisc.edu).
# All Rights Reserved.
# See the file README for a full copyright notice.
# Loaded: /usr/local/lib/trap.handler
# 666
################# End of output ################################

Another anwser:
################# Second Answer to Problem 2 ###################
        la      $s1, A          # start address of A
        addi    $s2, $0, 0      # running sum
        addi    $s3, $0, 0      # array index of A
        addi    $s4, $0, 7     # constant 10

 Loop:  mul     $t0, $s3, 4     # $t0 = array index * 4
        add     $t2, $t0, $s1   # $t2 = start of A + offset
        lw      $t1, 0($t2)     # $t1 = contents at start of A + offset
        add     $s2, $s2, $t1   # $s2 = sum of A[] so far
        addi    $s3, $s3, 1     # $s3 += 1
        bne     $s3, $s4, Loop  # brach back to Loop until $s4 == 10

        addi    $v0, $0, 1      # print the sum
        add     $a0, $0, $s2
        syscall
################# End of Second Answer to Problem 2 ############

========================================================================
3.
# MIPS program giving answer to problem 3
        .globl main
main:   addu    $s7, $zero, $ra
########## CALL function F #################
        li      $v0, 4
        la      $a0, Prob3
        syscall
        jal     F
        move    $a0, $v0
        li      $v0, 1
        syscall
        jal     Newl
########## Finish main#####################
        li      $v0, 4
        la      $a0, Thatsall
        syscall
        addu    $ra, $zero, $s7
        jr      $ra
########## PROB 3, function F ##############
F:      addi    $sp, $sp, -12   # Prob 3(a)
        sw      $a0, 0($sp)     # Prob 3(a)
        sw      $a1, 4($sp)     # Prob 3(a)
        sw      $ra, 8($sp)     # Prob 3(a)
        addi    $a0, $zero, 14  # Prob 3(b)
        addi    $a1, $zero, 47  # Prob 3(b)
        jal     G               # Prob 3(b)
        addi    $v0, $v0, 1     # Prob 3(d)
        lw      $a0, 0($sp)     # Prob 3(e)
        lw      $a1, 4($sp)     # Prob 3(e)
        lw      $ra, 8($sp)     # Prob 3(e)
        addi    $sp, $sp, 12    # Prob 3(e)
        jr      $ra             # Prob 3(f)
########## PROB 3, function G ##############
G:      add     $v0, $a0, $a1   # Prob 7(c)
        jr      $ra             # Prob 7(c)
########## write newline ################
Newl:   li      $v0, 4
        la      $a0, Newline
        syscall
        jr      $ra
########## write blank ##################
Blan:   li      $v0, 4
        la      $a0, Blank
        syscall
        jr      $ra
########## DATA ########################
        .data
Blank:  .asciiz  " "
Newline:   .asciiz  "\n"
Prob3:  .asciiz "Problem 3:\n"
Thatsall:  .asciiz "Th-th-th-thats all folks!\n"
##############################################################
# Output from the program:
# Problem 7:
```

```
# 62
# Th-th-th-thats all folks!
##############################################################

==========================================================================
4. Use
    (a)  add    $s1, $s2,   $0    or  addi $s1, $s2, 0
    (b)  addi   $s3, $0,   200
    (c)  beq    $0,  $0,  Loop

==========================================================================
5. With the given inputs, the upper two transistors are open (don’t conduct
  current), while the bottom two are closed.  Thus C is connected to
  the ground at the bottom and not to any power, so C’s output is 0.

   If either input is 0, C is 1, while both inputs 1 makes C a 0.
   Thus C is a NAND gate.
==========================================================================
```