1. (a) 92 (base 10) = 1011100 (base 2)

(b) 1011 1111 1110 0110 (48 more 0's) (binary) =

\[-92 (base 10) = ^{1111 1111 1010 0011} + 1 = ^{1111 1111 1010 0100}\]

(b) 1011 1111 1110 0110 (48 more 0's) (binary) =

\[-92 (base 10) = ^{1111 1111 1010 0011} + 1 = ^{1111 1111 1010 0100}\]

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, 0  # 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 7
Loop: lw      $t1, 0($s1)     # $t1 = A[$s3]
        add     $s2, $s2, $t1   # $s2 = sum of A[ ] so far
        addi    $s1, $s1, 1     # $s3 += 1
        bne     $s3, $s4, Loop  # branch back to Loop until $s4 == 7
addi    $v0, $0, 1      # print the sum
add     $a0, $0, $s2
syscall
# Answer to Problem 2 #################################################################

3. # Answer to Exam 1, Problem 3 #################################################################
.globl main
main:
    add     $s7, $0, $ra # save return address

# First part of answer, call to Addup #################################################################
add $a0, $0, 7
addi $a1, $0, 19
jal Addup
# End of call to Addup #################################################################
add $t0, $0, $v0    # save result in $t0
add     $a0, $0, $t0
syscall
# Second part of answer, Code for Addup #################################################################
Addup:
    addi $sp, $sp, -4
    sw $ra, 0($sp)
    add $v0, $a0, $a1
    lw $ra  0($sp)
    addi $sp, $sp, 4
    jr $ra
# End of code for Addup #################################################################

4. (a) One can branch 2^17 bytes or 2^15 words in either direction
(forward or backward), that is 128K bytes or 32K words in either direction.
(b) Change the given instruction to

    bne   $t0, $t1, Nextj
Label

Next: ...

5. (a) Assume A is 1 and B is 0. So upper switch connected to A is open
(doesn't conduct), while the upper switch connected to B is closed.
Since these are connected in series and one is open, no voltage goes
to C from the source. In the lower switches, A grounds the right switch, while B does not ground the left switch, but the grounds are connected in parallel, so C is grounded. Thus the value at C is 0.
If C is 0, this makes the upper switch conduct, giving voltage to D, while the lower switch does not conduct, so D is 1.
(b) This is a NOR gate (the output at C is 1 unless both A and B are 0) connected to an inverter, so the two form an OR gate.