1. c02e0000 00000000 (hexadecimal) = 
   1100 0000 0010 1110 (48 more 0’s) (binary) = 
   1 10000000010 1110 (48 more 0’s) (broken into fields) = 

\((-1)^{Sign} \times 2^{(Exponent - Bias)} \times (1 + Significand) = \\
\((-1)^1 \times 2^{(1026 - 1023)} \times (1 + 0.875000000) = \\
-1 \times 2^3 \times 1.875 = -8 \times 1.875 = -15.0
\)

2. 

```mips
.globl main
main:        addu $s7, $zero, $ra
            .globl main
            .data
            A:  .space 40
            .text
            la $s0, A          # address of A
            addi $t0, $0, 0      # loop counter, start at 0
            addi $t1, $zero, 10  # to terminate loop
            add $t2, $0, $s0    # address of item in A
            Loop:   sw $t0, 0($t2)     # store current $t0 into A
                     addi $t0, $t0, 1     # increment loop counter
                     addi $t2, $t2, 4     # increment pointer into A
                     bne $t0, $t1, Loop  # branch back to form loop
            .globl main
            .data
            A:  .space 40
            .text
            la $s0, A          # address of A
            addi $t0, $0, 0      # loop counter, start at 0
            addi $t1, $zero, 10  # to terminate loop
            add $t2, $0, $s0    # address of item in A
            Loop:   sw $t0, 0($t2)     # store current $t0 into A
                     addi $t0, $t0, 1     # increment loop counter
                     addi $t2, $t2, 4     # increment pointer into A
                     bne $t0, $t1, Loop  # branch back to form loop
            .globl main
            .data
            A:  .space 40
            .text
            la $s0, A          # address of A
            addi $t0, $0, 0      # loop counter, start at 0
            addi $t1, $zero, 10  # to terminate loop
            add $t2, $0, $s0    # address of item in A
            Loop:   sw $t0, 0($t2)     # store current $t0 into A
                     addi $t0, $t0, 1     # increment loop counter
                     addi $t2, $t2, 4     # increment pointer into A
                     bne $t0, $t1, Loop  # branch back to form loop
```

## Print the array

```
la $a0, A
li $a1, 10
jal Write_array
jal Newl
```

## Finish main

```
addu $ra, $zero, $s7
jr $ra
```

## write an array

```
Write_array:
            addi $sp, $sp, -4  # room for $ra on stack
            sw $ra, 0($sp)  # save $ra because not leaf
            ## initialization for loop
            move $s0, $a0  # $s0 = $a0 = start of A
            move $s1, $a1  # $s1 = $a1 = N
            move $t1, $zero  # start $t1 = 0, the index
            LoopA: beq $s1, $t1, EndA  # if (N == index) goto EndA
            ## write value for A[i]
            addu $t2, $t1, $t1
            addu $t2, $t2, $t2
            addu $t2, $s0, $t2  # $t2 = index*4 + start of A
            li $v0, 1
            lw $a0, 0($t2)  # integer to print
```
syscall
### write a blank
jal Blan
addi $t1, $t1, 1
j LoopA
EndA: lw $ra, 0($sp)
addi $sp, $sp, 4
jr $ra

### 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
Blank: .asciiz " 
Newline: .asciiz "\n"

Alternatively, the following code uses the mul pseudo-instr:

### Start of answer to Question 2 #######
.data
A: .space 40
.text
la $s0, A          # address of A
addi $t0, $0, 0      # loop counter, start at 0
addi $t1, $zero, 10  # to terminate loop
Loop: mul $t2, $t0, 4     # pseudo-instr, mult $t0 by 4
add $t3, $t2, $s0   # add to start addr of A
sw $t0, 0($t3)     # store current $t0 into A
addi $t0, $t0, 1   # increment loop counter
bne $t0, $t1, Loop  # branch back to form loop

### End of answer to Question 2 #######

3.
# Answer to Exam1, Problem 3 #######
.globl main
main: add $s7, $0, $ra # save return address

### First part of answer, call to Addup #######
addi $a0, $0, 7
addi $a1, $0, 19
jal Addup

### End of call to Addup #######
add $t0, $0, $v0 # save result in $t0
addi $v0, $0, 1      # print the returned value
add $a0, $0, $t0
syscall

addi $v0, $0, 4      # print a newline
la $a0, Newln
syscall

add $ra, $zero, $s7 # restore return address
jr $ra

####### 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 ########################

.data
Newln:    .asciiz "\n"

################# Output ####################################
four06% spim -file exam1_3.s
26

==========================================================================

4.

<table>
<thead>
<tr>
<th>Instruction</th>
<th>op</th>
<th>rs</th>
<th>rt</th>
<th>rd</th>
<th>sh</th>
<th>ft</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x010A4820</td>
<td>000000 01000 01010 01001 00000 100000</td>
<td>0 8 10 9 32</td>
<td>add $t1, $t0, $t2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x20080064</td>
<td>001000 00000 01000 00000 00001 100100</td>
<td>8 0 8 100</td>
<td>addi $t0, $0, 100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x08100200</td>
<td>000010 00000 10000 00000 01000 00000</td>
<td>2</td>
<td>0x0010200  j</td>
<td>0x00400800</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

==========================================================================

5.(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, Next
j   Label
Next: ...

==========================================================================

6. 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.

==========================================================================