CS 2734, Computer Organization II
Spring Semester, 2001
First Examination

1. This question is about the representation of a **`double`**: **(15)**

   Consider the floating point number (a `double`) with hexadecimal and binary representations:

```
c02e0000 00000000                    (hexademcimal)
1100 0000 0010 1110  (48 more 0's)   (binary)
```

   What is the number? (Show your work. Remember that the bias for a `double` is 1023, and that an exponent of 1 is represented by `100 0000 0000`.)

2. Consider the following MIPS code fragment: **(20)**

```
        .data
A:      .space 40
        .text
# insert MIPS instructions here.
```

   For insertion at the comment, write MIPS instructions that will do the following:

   (a) Create a loop of 10 iterations that will let the register `$t0` take on values 0, 1, ..., 9.

   (b) Store the values of `$t0` into successive words of the array `A`, so that if we printed the 10 locations of `A`, we would print out the numbers 0, 1, ... 9.
   (You should not include code to do this printing. Your MIPS code should do what is asked for above and *nothing more*.)

3. Write a MIPS function `Addup` so that **(20)**

   (a) `Addup` saves register `$ra` *on the stack*.

   (b) `Addup` adds its two input parameters and returns the sum.

   (c) `Addup` restores the register `$ra` saved above and returns.

   (d) Separately show a call to `Addup` with input parameters 7 and 19.
   Note: You should just give code for the call to `Addup` and for the definition of the function `Addup` that do the above items and *nothing more*. You should follow MIPS parameter convensions.

4. Give the machine code for each of the given instructions below. You can simply give the decimal value of the bits in each of the fields of the instruction, so you do not need to convert to binary or hexadecimal. (Note that register `$t0` is 8 and register `$t1` is 9.) **(15)**

   (a) `add  $t1, $t0, $t2.`

   (b) `addi  $t1, $t0, 100.`

   (c) `j  Loop,`
   where `Loop` is at the address $0040\ 0800_{hex}$.

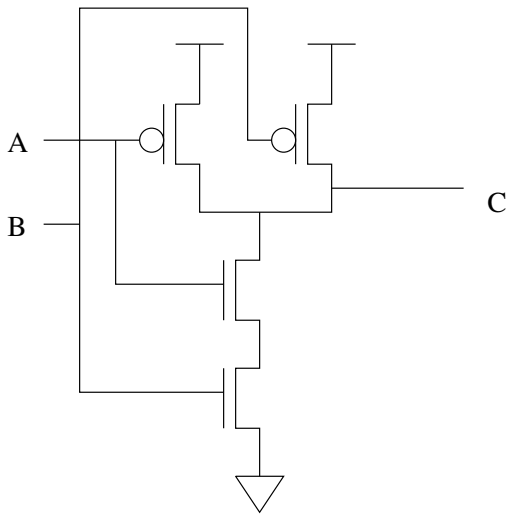
5. Consider the following assembler instruction: **(15)**

```
        beq  $t0, $t1, Label
          ...  # a large number of instructions
Label:
```

   (a) This instruction will not work in case **`Label`** is "too far away." Say precisely how far "too far away" is. (Be sure to say whether your answer is in bytes or words.)

   (b) In case **`Label`** is too far away as in (b) above, show how the **`beq`** instruction could be changed to two instructions that would always work.

6. Consider the following logic gate constructed out of CMOS transistors. **(15)**

   (a) In case A is a 1 (voltage high) and B is also a 1, what will be the output at C? Explain your answer in terms of the diagram and the properties of the transistors. (Show which switches are open (don't conduct current), which are closed (conduct current), and explain why the output at C is what it is.)

   (b) Say what kind of logic gate this represents.