CS 2733/2731, Computer Organization II
Fall Semester, 2003
First Examination

1. Below are questions about number representations and conversions: **(20)**

   (a) Convert the (decimal) number -92 to 16-bit two's complement binary. (The binary representation for 92 is 1011100.)

   (b) Consider the floating point number (a **`double`**) with representations:

```
1011 1111 1110 0110  (48 more 0's)  (binary)
b    f    e    6     (12 more 0's)  (hex)
```

   i. What is the sign of this number?

   ii. What is its exponent (power of 2)? (Remember that the bias for a **`double`** is 1023, and that an exponent of 1 is represented by **`100 0000 0000`**.)

   iii. What is the significant part?

   iv. Put i, ii, and iii together to get the number.

---

2. Consider the following MIPS code fragment: **(25)**

```
      .data
# stored in A are squares of first 7 primes, zero at end
A:    .word  4, 9, 25, 49, 121, 169, 289, 0
      .text
# insert MIPS instructions here.
```

For insertion at the comment, write a *single* MIPS program that will do all of the following (not item-by-item, but all at once):

   (a) Put the starting address of **`A`** into register **`$s1`**.

   (b) Inside a loop, access each element of **`A`** and add these values, leaving the result in register **`$s2`**. [You must use a loop for this.]

   (c) Print the resulting sum, using **`syscall`**. [Recall that **`syscall`** requires **`$v0`** equal to 1 to print the value in **`$a0`**.]

   Your MIPS code should do what is asked for above and *nothing more*.

---

3. Write a *single* MIPS function **`Addup`** that does a., b., and c. below. **(25)**

   (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.

(10)

4. Consider the following assembler instruction:

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

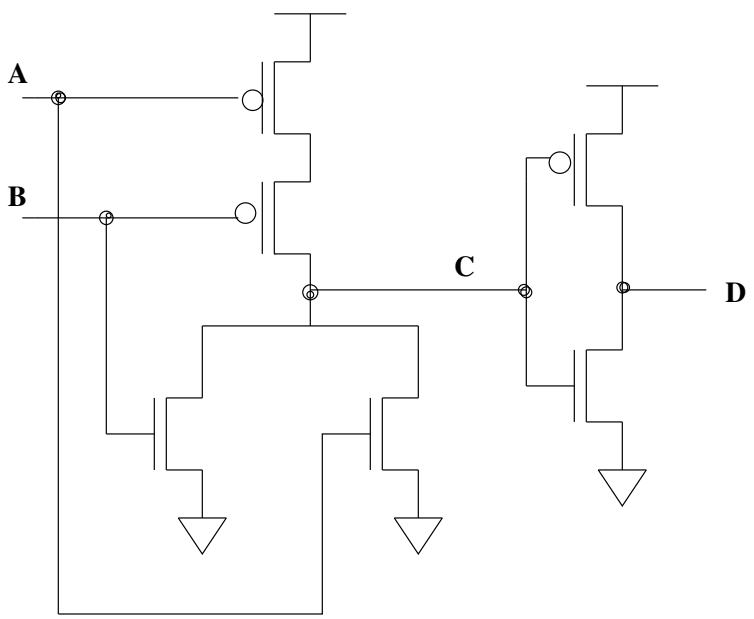
   (a) This instruction will not work in case **`Label`** is "too far away." Say precisely how far "too far away" is. (Be sure to say whether your answer is in bytes or words.)

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

---

(20)

5. Consider the following logic gate constructed out of CMOS transistors.

   (a) In case A is a 1 (voltage high) and B is a 0, what will be the value at C and the output at D? Explain your answer in terms of the diagram and the properties of the transistors. (Show which switches are open, which are closed.)

   (b) What kind of gate does this diagram represent? (Explain.)