CS 2734, Computer Organization II
Fall Semester, 2002
First Examination

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

   (a) Convert the (decimal) number -82 to 16-bit two's complement binary. (The binary representation for 82 is 1010010.)

   (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
A:      .word  4, 9, 25, 49, 121, 169, 289
        .text
# insert MIPS instructions here.
```

   For insertion at the comment, write a *single* MIPS program that will do the following:

   (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. Consider the same MIPS code fragment in the previous question, that defines an array `A` of 7 integers. **(25)**

   Write a *single* MIPS function `Avals` so that

   (a) `Avals` is passed the starting address of `A` as its first parameter and the number 7 as its second parameter. (You should follow MIPS parameter passing conventions.)

   (b) `Avalues` saves the register `$ra` on the stack.

(c) `Avals` adds the 0th and 1st array elements, and returns this number. (`Avals` should return `A[0] + A[1]`. You should follow MIPS conventions for returning a value.)

(d) Before returning, `Avals` restores the register `$ra` saved above and should restore the stack.

(e) Separately show a call to `Avals` with first input parameter the starting address of `A` and second input parameter the number 7.

Note: You should just give code for the call to `Avals` and for the definition of the function `Avals` that do the above items and *nothing more*. You should follow MIPS parameter convensions.

---

4. The MIPS assembler can use actual machine instruction to create other *pseudo-instructions*. In each case below, show how the given instruction could be rendered using an actual instruction (examples of actual instructions include `add`, `addi`, `slt`, `beq`, and `bne`): **(10)**

```
(a)   move $s1, $s2          # $s1 = $s2
(b)   li   $s3, 200          # $s3 = 200
(c)   b    Loop              # unconditional branch
```

---

5. Consider the following logic gate constructed out of CMOS transistors. **(20)**

(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.)