CS 2733/2731, Computer Organization II
Fall 2004, *First Examination*

1. (a) Convert the (decimal) number -92 to 16-bit two's complement binary. (The binary representation for 92 is 1011100.) **(15)**

   (b) Extend your answer in part (a) to a 32-bit two's complement binary number.

---

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

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

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

(a) Put the address of **A** into **$s0**.

(b) *Use a loop* to put the numbers

2, 4, 6, 8, 10, 12, 14, 16, 18, 20

into succesive locations of the array **A**. (You must use a loop for credit. You should not print anything. Your MIPS code should do what is asked for above and *nothing more*.)

---

3. Consider the following MIPS code segment: **(30)**

```
        .data
# stored in A are squares of the first 7 primes
A:      .word   4, 9, 25, 49, 121, 169, 289
        .text
main:   add     $s7, $0, $ra    # save return address
        la      $a0, A
        li      $a1, 7
        jal     P
        move    $a0, $v0
        li      $v0, 1
        syscall
        add     $ra, $0, $s7    # restore return address
        jr      $ra

# insert MIPS code here for the function P
```

For insertion at the comment, write a *single* MIPS function **P** that will fit with the code above and will do each of the following:

(a) The first parameter of **P** should be an array.

(b) The second parameter of **P** should be an integer.
(c) You should follow MIPS parameter passing conventions.
(d) The function **P** should save the return address *on the stack* at the beginning and restore it at the end.
(e) The function should add numbers of the array which is the first parameter.
(f) The number of elements of the array to add should be given by the second parameter.
(g) The value returned by the function should be this sum.
(You must use a loop inside the function for credit. You should not print anything. Your MIPS code should do what is asked for and should work correctly with the additional code given above.)

---

4. Consider the *add immediate* (**addi**) instruction in MIPS. **(20)**

(a) Give the MIPS machine language for the following instruction (where **$s0** is **16** and **$t0** is **8**):

```
addi $s0, $t0, 100
```

For each field in the machine language version, give the name, the number of bits, and the contents as a decimal number.

(b) Show how to use an **addi** instruction to implement the following pseudo-instructions:

i.
```
li      $v0,   4    # load immediate
```
ii.
```
move    $v0, $t0    # move
```

---

5. Consider the following diagram of a circuit component. (On the left is the abstract diagram, and on the right is the implementation using gates.) **(15)**

(a) What is the name of this circuit component?
(b) Suppose the input A is 1, the input B is 1, and the input S is 1. Trace the values through the circuit, showing the inputs to each gate and the output from each gate (as a 0 or a 1). In particular show the output at C in this case.
(c) Say in simple words what is happening when S is 1.