

# Computer Organization

## Fall 2001

### Test #1

There are 6 pages - make sure you have all of them.  
 Answer all questions - pay attention to the # of points for each question.  
 Don't leave anything blank - partial credit is always possible!

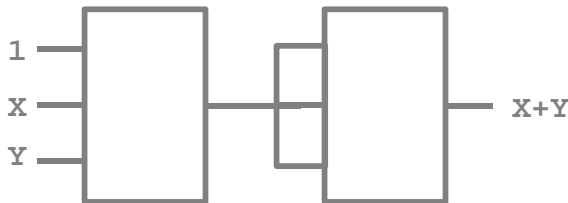
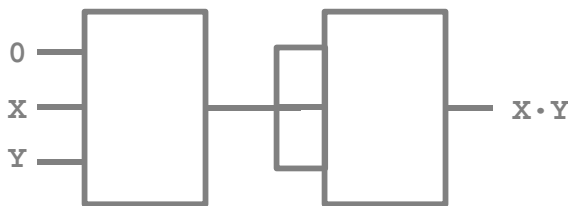
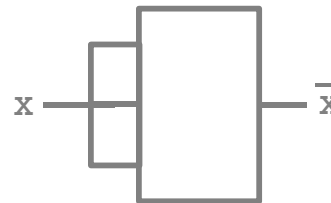
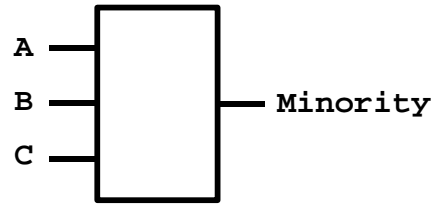
---

**Question 1 (15 pts):** Prove that *minority function* over 3 variables is a universal operator (*functionally complete*). The value of the minority function is defined as:

- 0: when there are more inputs that have the value 1 than inputs that have the value 0.
- 1: when there are more inputs that have the value 0 than inputs that have the value 1.

**Truth Table**

| A | B | C | Minority |
|---|---|---|----------|
| 0 | 0 | 0 | 1        |
| 0 | 0 | 1 | 1        |
| 0 | 1 | 0 | 1        |
| 0 | 1 | 1 | 0        |
| 1 | 0 | 0 | 1        |
| 1 | 0 | 1 | 0        |
| 1 | 1 | 0 | 0        |
| 1 | 1 | 1 | 0        |



**Question 2 (20 pts):** Provide a sequence of MIPS assembly language instructions that reverses a string in place (changes the original string). Here is one possible C code segment to accomplish this – if you base your assembly code on different C code - please show us your C code as well! Comments can help us understand your assembly code!

```
/*
C code segment to reverse the string s
n,left and right are int, tmp is a char
n is the length of the string
*/

left=0; right=n-1;

while (left<right) {
    tmp = s[left];
    s[left] = s[right];
    s[right] = tmp;
    left++; right--;
}
```

```
# $S0 is left, $S1 is right
# $S2 is s $S3 is tmp

add $s0,$zero,$zero      # left = 0
add $s1,$s3,-1           # right=n-1
loop:
    slt $t0,$s0,$s1      # left < right?
    beq $t0,$zero,endlp  # no - jump

    # calc address of s[l] ($t0)
    add $t0,$s2,$s0
    # calc address of s[r] ($t1)
    add $t1,$s2,$s1

    lb $s3,0($t0)         # tmp = s[l]
    lb $t3,0($t1)         # get s[r]
    sb $t3,0($t0)         # s[l] = s[r]
    sb $s3,0($t1)         # s[r] = tmp

    add $s0,$s0,1         # l++;
    add $s1,$s1,-1        # r--;
    j loop
endlp:
```

**Question 3 (20 pts): True/False - 2 pts each.** No explanation needed (or graded)!

The MIPS (*million instructions per second*) rating of a fixed processor can vary depending on the program used to make measurements.

TRUE

FALSE

A *benchmark* is a computer program.

TRUE

FALSE

When making performance comparisons of two computers, it's not enough to know just execution time, you also need to know the IC, CPI and clock rate for both computers.

TRUE

FALSE

Assuming register  $\$s1$  contains the value  $1234_{10}$ , the MIPS instruction: `SW $zero, 2($s1)` will put a 0 in memory location  $1242_{10}$ .

TRUE

FALSE

The implementation of a single Boolean function using (non-minimized) SOP form always requires exactly the same number of gates as the implementation using POS form.

TRUE

FALSE

A 32 input Multiplexor will require 5 selection inputs.

TRUE

FALSE

IC (Instruction Count) depends on the clock rate.

TRUE

FALSE

It is possible to have two different truth tables that describe the same Boolean function.

TRUE

FALSE

In Boolean Algebra, the following is always true (regardless of the values of A and B):

TRUE

FALSE

$$\overline{\overline{A} + B} = B \bullet \overline{A}$$

This circuit becomes unstable if A is a 1 and then becomes a 0.

TRUE

FALSE



**Question 4 (15 pts):** We have two MIPS assembly language program segments, each accomplishes the same task: clear an array of integers. The MIPS assembly code for both segments is shown below, the corresponding C code is also shown.

```

PROGRAM SEGMENT A

#   for (i=0;i<n;i++)
#       x[i] = 0;
#
# s0 is i, s1 is n, s2 is x

        add $s0,$zero,$zero
loop:   slt $t0,$s0,$s1
        beq $t0,$zero,end
        add $t1,$s0,$s0
        add $t1,$t1,$t1
        add $t1,$t1,$s2
        sw $zero,0($t1)
        addi $s0,$s0,1
        j loop
end:

```

```

PROGRAM SEGMENT B

#   for (i=0;i<n;i++) {
#       *x=0; x++;
#   }
#
# s0 is i, s1 is n, s2 is x

        add $s0,$zero,$zero
loop:   slt $t0,$s0,$s1
        beq $t0,$zero,end
        sw $zero,0($s2)
        addi $s0,$s0,1
        addi $s2,$s2,4
        j loop
end:

```

The number of cycles it takes for each of the MIPS instruction types is known:

|                                       |                          |
|---------------------------------------|--------------------------|
| Arithmetic: add, sub, addi, slt, slti | 1 cycle per instruction  |
| Memory: sw, lw, sb, lb                | 5 cycles per instruction |
| Control: beq, bne, j                  | 2 cycles per instruction |

Our computer runs at 10 MHz. Answer the following questions (fill in the 3 blanks). You must show your work! (use the back of the page if necessary):

Program segment A will take 1.4 seconds when n=1,000,000.

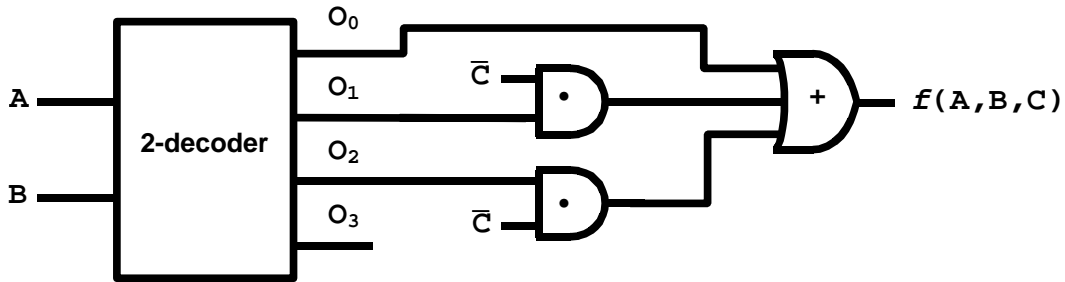
Program segment B will take 1.2 seconds when n=1,000,000.

Program segment B is 1.4/1.2 times faster than A when n=1,000,000.

**Segment A is:  $1+1+2+n*(1+1+1+5+1+2+1+2) = 4+n*14$  cycles**

**Segment B is:  $1+1+2+n*(5+1+1+2+1+2) = 4+n*12$  cycles**

**Question 5 (15 pts):** The following diagram shows an implementation of a Boolean function  $f(A,B,C)$ .



**Part 1 (8 pts):** Fill in the truth table for  $f$ .

| A | B | C | $f$ |
|---|---|---|-----|
| 0 | 0 | 0 | 1   |
| 0 | 0 | 1 | 1   |
| 0 | 1 | 0 | 1   |
| 0 | 1 | 1 | 0   |
| 1 | 0 | 0 | 1   |
| 1 | 0 | 1 | 0   |
| 1 | 1 | 0 | 0   |
| 1 | 1 | 1 | 0   |

**Part 2 (7 pts):** Describe  $f$  with a Boolean algebra expression in Sum of Products form.

$$f(A,B,C) = \bar{A} \cdot \bar{B} \cdot \bar{C} + \bar{A} \cdot \bar{B} \cdot C + \bar{A} \cdot B \cdot \bar{C} + A \cdot \bar{B} \cdot \bar{C}$$

*Minority Function!*

**Question 6 (15 pts):** You need to write a C program named `doboth` that will accept two command line arguments; each argument is the complete path to a program. `doboth` should run the first program and wait for it to complete, then run the second program. For example, we could tell `doboth` to run the `date` command followed by the `ls` command like this: `doboth /usr/bin/date /usr/bin/ls`. We might also have it run those programs in the other order as well. Here is a sample session showing both orders:

```
> doboth /usr/bin/date /usr/bin/ls
Mon Oct 8 11:23:02 EDT 2001
blah doboth.c doboth

> doboth /usr/bin/ls /usr/bin/date
blah doboth.c doboth
Mon Oct 8 11:23:58 EDT 2001
```

**NOTE:** neither of the programs that is run by `doboth` will be passed any command line arguments!

Do your best to provide the actual C code, but feel free to describe what is needed in place of actual calls to library functions or system calls if you can't remember the exact prototypes.

```
#include <stdio.h>

int main(int argc, char **argv) {
    int stat;

    if (argc<3) {
        printf("Invalid usage\n");
        exit(1);
    }

    if (fork()) {
        // parent - wait for child
        wait(&stat);
        // no exec the second one
        execl(argv[2],argv[2],NULL);
    } else {
        // child - exec the first one
        execl(argv[1],argv[1],NULL);
    }
    return(0);
}
```