1. (30%) A majority function is generated in a combinational circuit when the output is equal to 1 if the input variables have more 1’s than 0’s. The output is 0 otherwise.

   a. (10%) Please write the truth table for a 4-input majority function \( F \).
   b. (10%) Please use the Karnaugh map to find the minimum sum of products form for \( F \) and the minimum sum of products form for \( F' \).
   c. (10%) Please draw the logic schematic for \( F \) by using AND, OR, and INVERT gates.

[Answer]

   a. 
   
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

   b. 
   
<table>
<thead>
<tr>
<th></th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

   \( F = ABD + BCD + ABC + ACD \)
\[ F' = A'B' + B'D' + B'C' + C'D' + A'D' + A'C' \]

c.
No answer provided.
2. (10%) Prove the following simplification theorems using the first eight laws of Boolean algebra:

   a. (5%) \((X + Y)(\overline{X} + Z) = XZ + XY\)
   b. (5%) \(XY + YZ + \overline{X}Z = XY + \overline{X}Z\)

[Answer]

a.
\((X+Y)(X'+Z)\)

\begin{align*}
&= XX'+XZ+X'Y+YZ \\
&= XZ+X'Y+YZ \\
&= XZ+X'Y+YZ(X+X') \\
&= XZ+X'Y+XYZ+X'YZ \\
&= XZ(1+Y)+X'Y(1+Z) \\
&= XZ+X'Y
\end{align*}

b.
\((XY+YZ+X'Z)\)

\begin{align*}
&= XY+YZ(X+X')+X'Z \\
&= XY+X'Z+XYZ+X'YZ \\
&= XY(1+Z)+X'Z(1+Y) \\
&= XY+X'Z
\end{align*}
3. (10%) The table below shows the number of floating-point operations executed in two different programs and the runtime for those programs on three different machines:

<table>
<thead>
<tr>
<th>Program</th>
<th>Floating-point operations</th>
<th>Execution time in seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Computer A</td>
</tr>
<tr>
<td>Program 1</td>
<td>10,000,000</td>
<td>1</td>
</tr>
<tr>
<td>Program 2</td>
<td>100,000,000</td>
<td>1000</td>
</tr>
</tbody>
</table>

a. (5%) Which machine is fastest according to total execution time?
b. (5%) How much faster is it than the other two machines?

[Answer]
a. Total execution time of A is 1001 seconds, of B is 110 seconds, and of C is 40 seconds. Computer C is fastest.
b. C is 25 (1001 / 40) times faster than A and is 2.75 (110 / 40) times faster than B.
4. (10%) You are going to enhance a machine, and there are two possible improvements: either make multiply instructions run four times faster than before, or make memory access instructions run two times faster than before. You repeatedly run a program that takes 100 seconds to execute. Of this time, 20% is used for multiplication, 50% for memory access instructions, and 30% for other tasks.

   a. (3%) What will the speedup be if you improve only multiplication?
   b. (3%) What will the speedup be if you improve only memory access?
   c. (4%) What will the speedup be if both improvements are made?

[Answer]
Using Amdahl’s law (or just common sense) we can determine the following:
   a. $\frac{100}{30 + 50 + \frac{20}{4}} = \frac{100}{85} = 1.18$
   b. $\frac{100}{30 + \frac{50}{2} + 20} = \frac{100}{75} = 1.33$
   c. $\frac{100}{30 + \frac{50}{2} + \frac{20}{4}} = \frac{100}{60} = 1.67$
5. (10%) The following code fragment processes two arrays. Assume that the base address of the first array is stored in $a0 and the base address of the second array is stored in $a1.

sub $sp, $sp, 4
sw $s0, 0($sp)
add $s0, $zero, $zero

John: add $t1, $a1, $s0
lb $t2, 0($t1)
add $t3, $a0, $s0
sb $t2, 0($t3)
beq $t2, $zero, Mary
add $s0, $s0, 1
j John

Mary: lw $s0, 0($sp)
add $sp, $sp, 4
jr $ra

a. (5%) Please write the comment for each code fragment line and describe what this code does.
b. (5%) Assume that the size of the two arrays are both 60,000 and the code is run on a machine with 60 MHz clock that requires the following number of cycles for each instruction. In the worst case, how many seconds will it take to execute this code?

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>add, sub</td>
<td>2</td>
</tr>
<tr>
<td>lw, sw, lb, sb, beq, j, jr</td>
<td>3</td>
</tr>
</tbody>
</table>

[Answer]

b. 60000 x (2+3+2+3+3+2+3) / (60 x 10^6) = 0.018 seconds.
6. (10%) Consider the following fragment of C code:

```c
for (i = 0; i <= 200; i = i + 1) { a[i] = b[i] + c[i]; }
```

Assume that a and b are arrays of words and the base address of a is in $a0$, the base address of b is in $a1$, and the base address of c is in $a2$. Register $t0$ is associated with the variable i.

a. (6%) Please write the code for MIPS and write the comment for each MIPS statement.
b. (2%) How many instructions are executed during the running of this code?
c. (2%) How many memory data references will be made during the execution?

[Answer]

a. 
```
  add $t0, $zero, $zero # Temp reg $t0 = 0
  lw $t1, AddressConstant4($zero) # Temp reg $t1 = 4
  lw $t2, AddressConstant801($zero) # Temp reg $t2 = 801
  Loop: add $t3, $a1, $t0 # Temp reg $t3 = address of b[i]
         lw $t4, 0($t3) # Temp reg $t4 = b[i]
         add $t5, $a2, $t0 # Temp reg $t5 = address of c[i]
         lw $t6, 0($t5) # Temp reg $t6 = c[i]
         add $t7, $t4, $t6 # Temp reg $t7 = b[i] + c[i]
         add $t8, $a0, $t0 # Temp reg $t8 = address of a[i]
         sw $t7, 0($t8) # a[i] = b[i] + c
         add $t0, $t0, $t1 # i = i + 4
         slt $t9, $t0, $t2 # $t9 = 1 if $t0 < 801 (i ≤ 200)
         bne $t9, $zero, Loop # go to Loop if i ≤ 200
```

b. $3 + 201 \times 10 = 2013$

c. $2 + 201 \times 3 = 605$
7. (10%) IEEE 754 is a floating-point standard.

   a. (5%) Please show the IEEE 754 binary representation for the floating-point number 
      \(-0.4375_{\text{ten}}\) in single precision.
   b. (5%) Please convert the following IEEE 754 binary representation in single 
      precision to the decimal number: \(1011\ 1111\ 1100\ 0000\ 0000\ 0000\ 0000\ 0000\).

[Answer]
   a. \(1\ 01111101\ 1100000000000000000000000\)
   b. \(-1.5_{\text{ten}}\)
8. (10%) The ALU supported set on less than (slt) using just the sign bit of the adder. Let’s try a set on less than operation using the values -7\text{ten} and 6\text{ten}. To make it simpler to follow the example, let’s limit the binary representations to 4 bits: 1001\text{two} and 0110\text{two}.

\[ 1001\text{two} - 0110\text{two} = 1001\text{two} + 1010\text{two} = 0011\text{two} \]

This result would suggest that -7\text{ten} > 6\text{ten}, which is clearly wrong. Hence we must factor in overflow in the decision. Modify the 1-bit ALU in the following figures to handle slt correctly. (You can just write on the next page and deliver it with your answer papers.)

![Figure 1: A 1-bit ALU that performs AND, OR, and addition on a and b or b'].

![Figure 2: A 1-bit ALU for the most significant bit.]

[Answer]

<table>
<thead>
<tr>
<th>Overflow</th>
<th>Result31</th>
<th>LessThan</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

LessThan = Overflow XOR Result31