1. Number Systems ___________________  (15 points)

2. Binary Addition/Condition Code Bits/Overflow Detection ___________________  (12 points)

3. Branching ___________________  (13 points)

4. Bit Operations ___________________  (10 points)

5. Parameter Passing and Return Values ___________________  (12 points)

6. Local Variables, The Stack, and Return Values ___________________  (18 points)

SubTotal ___________________  (80 points)

Extra Credit ___________________  (4 points)

Total ___________________
1. Number Systems

Convert $\text{DAED}_{16}$ (2’s complement, 16-bit word) to the following. (6 points)

- decimal: 
- octal: 0
- binary: 

Convert $-197_{10}$ to the following (assume 16-bit word). Express answers in hexadecimal. (6 points)

- sign-magnitude: 0x
- 1’s complement: 0x
- 2’s complement: 0x

Convert $+174_{10}$ to the following (assume 16-bit word). Express answers in hexadecimal. (3 points)

- sign-magnitude: 0x
- 1’s complement: 0x
- 2’s complement: 0x
2. Binary Addition/Condition Code Bits/Overflow Detection

Indicate what the condition code bits are when adding the following 8-bit 2’s complement numbers. (12 points)

\[
\begin{align*}
01011101 & \quad +00101011 & \quad 01010100 & \quad +10111011 & \quad 10111001 & \quad +10010110 \\
\hline
--------- & \quad --------- & \quad --------- & \quad --------- & \quad --------- & \quad --------- \\
\end{align*}
\]

\[
\begin{array}{cccccccc}
N & Z & V & C & \quad N & Z & V & C & \quad N & Z & V & C \\
\hline
| & | & | & \quad | & | & | & \quad | & | & | \\
\hline
\end{array}
\]

3. Branching

Fill in the SPARC assembly instructions to perform the following statements. Do not optimize. (9 points)

\[
\begin{array}{cccccccc}
C & \quad \text{SPARC assembly} & \quad \text{! i mapped to } \%10 \\
\hline
\end{array}
\]

\[
\begin{array}{c}
\text{if ( i >= 15 )} \\
\text{statement1; } \\
\text{statement2; } \\
\text{else} \\
\text{statement3; } \\
\text{statement4; } \\
\end{array}
\]

For the following instruction sequence, mark with an X the conditional branch instruction which would transfer control to loop if used in place of ba. (0-4 points: +1 for each correct; -1 for each incorrect; 0 if all marked)

<table>
<thead>
<tr>
<th>Instruction sequence</th>
<th>bvs</th>
<th>bne</th>
<th>bl</th>
<th>bcs</th>
<th>ble</th>
<th>bge</th>
<th>bpos</th>
</tr>
</thead>
<tbody>
<tr>
<td>mov -2, %10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>addcc %10, 5, %10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ba loop</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>loop</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4. Bit Operations

What is the value of `%l0` after each statement is executed? Express your answers in hexadecimal.

set 0xA6127834, %l0
or %l0, 0x3C8, %l0
Value in %l0 is 0x______________________________ (2 points)

set 0xA6127834, %l0
sll %l0, 10, %l0
Value in %l0 is 0x______________________________ (2 points)

set 0xA6127834, %l0
sra %l0, 7, %l0
Value in %l0 is 0x______________________________ (2 points)

set 0xA6127834, %l0
set 0x6943B9C5, %l1
xor %l0, %l1, %l0
Value in %l0 is 0x______________________________ (2 points)

set 0xA6127834, %l0
set 0x6943B9C5, %l1
btog %l1, %l0
Value in %l0 is 0x______________________________ (2 points)
5. Parameter Passing and Return Values

Write the equivalent unoptimized SPARC assembly language instructions to perform the following C code fragment. (10 points)

```c
x = foo( 420, -5432 );                    /* x is mapped to %l0 */
```

Now optimize your answer to eliminate any delay slots. (2 points)
6. Local Variables, The Stack, and Return Values

Here is a C function that doesn’t do much but allocate local variables, perform some assignments, and returns a value:

```c
int foo( int i, int j ) {
    int    local_stack_var1;
    short  local_stack_var2[3];
    local_stack_var1 = 6061;
    local_stack_var2[2] = 10;
    j = local_stack_var2[1];
    return ( i - 5 );
}
```

Now write the equivalent unoptimized SPARC assembly language instructions to perform the equivalent. You must allocate all local variables on the stack. Perform each instruction literally. No short-cuts. (18 points)

```
SPARC assembly
.global foo
.section "".text"

foo: /* Your unoptimized code goes below this point */
```
Extra Credit

Optimize the following SPARC assembly language code fragment. Some optimizations may be better than others. More points for the better optimizations. Use only the code shown. (4 points)

<table>
<thead>
<tr>
<th>Unoptimized Version</th>
<th>Optimized Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>/* some code above here */</td>
<td>/* some code above here */</td>
</tr>
<tr>
<td>loop1:</td>
<td></td>
</tr>
<tr>
<td>add  %l0, %l1, %l2</td>
<td></td>
</tr>
<tr>
<td>mov  %l2, %o0</td>
<td></td>
</tr>
<tr>
<td>mov  2, %o1</td>
<td></td>
</tr>
<tr>
<td>call .rem</td>
<td></td>
</tr>
<tr>
<td>nop</td>
<td></td>
</tr>
<tr>
<td>cmp  %o0, %l3</td>
<td></td>
</tr>
<tr>
<td>ble  loop1</td>
<td></td>
</tr>
<tr>
<td>nop</td>
<td></td>
</tr>
<tr>
<td>/* other code */</td>
<td>/* other code */</td>
</tr>
</tbody>
</table>
Scratch Paper