CSE 30
Spring 2003
Midterm Exam

1. Number Systems ___________________ (15 points)
2. Binary Addition/Condition Code Bits/Overflow Detection ___________________ (12 points)
3. Branching ___________________ (20 points)
4. Bit Operations / C Runtime Environment ___________________ (17 points)
5. Parameter Passing and Return Values (Structures) ___________________ (12 points)
6. Local Variables, The Stack and Return Values ___________________ (15 points)
7. Load/Store/Memory ___________________ (9 points)

SubTotal ___________________ (100 points)
Extra Credit ___________________ (5 points)

Total ___________________
1. Number Systems

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

binary  

octal  0

decimal  

Convert $282_{10}$ to the following (assume 16-bit word). **Express answers in hexadecimal.** (3 points)

sign-magnitude 0x

1’s complement 0x

2’s complement 0x

Convert $-489_{10}$ to the following (assume 16-bit word). **Express answers in hexadecimal.** (6 points)

sign-magnitude 0x

1’s complement 0x

2’s complement 0x
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)

\[
\begin{array}{c}
10010111 & 01010110 & 10111011 \\
+10101001 & +10111001 & +00010100 \\
\hline
\end{array}
\]

<table>
<thead>
<tr>
<th>N</th>
<th>Z</th>
<th>V</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3. Branching
Write the SPARC assembly instructions to complete the following. **Do not optimize nops.** (20 points)

```c
int findMin( int b[], int n ) {
    int j;
    int min = b[0];
    for ( j = 0; j < n; ++j ) {
        if ( *b < min )
            min = *b;
        ++b;
    }
    return min;
}
```

```sparc
.global findMin
.section ".text"
findMin:
    save %sp, -96, %sp
    /* Complete the rest of this function starting here */
    ! j mapped to %11
    ! min mapped to %13
```
4. Bit Operations / C Runtime Environment

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

set 0xE95CD4BA, %l0
sra %l0, 9, %l0

Value in %l0 is 0x__________________________________________ (2 points)

set 0xE95CD4BA, %l0
sll %l0, 13, %l0

Value in %l0 is 0x__________________________________________ (2 points)

set 0xE95CD4BA, %l0
set __________, %l1
xor %l0, %l1, %l0 ! Value in %l0 is now OxCAFEBABE

Value set in %l1 must be this bit pattern 0x__________________________________________ (3 points)

Fill in the names of the 5 areas of the C Runtime Environment as laid out by the SPARC architecture. Then state what parts of a C program are in each area. (10 points)

low memory

__________________________________________________________
__________________________________________________________
__________________________________________________________
__________________________________________________________
__________________________________________________________

high memory
Write the equivalent **unoptimized** SPARC assembly language instructions to perform the following C code fragment. You can assume just this one local variable. (12 points)

```c
/* Function Prototype */
char foo( int, short, char );

/* ... Other code ... */

/* Assume this local variable
   is declared appropriately
   and is the only local var. */

struct fubar {
    short a;
    char  b;
    long  c;
    char  d;
    short e;
    int   f;
} fb;   /* Local variable fb */

/* ... Other code ... */

/*
Write the code for just this
function call saving the
return value appropriately
*/

fb.b = foo( fb.f, fb.a, fb.d );
```
6. Local Variables, The Stack, and Return Values
Here is a C function that doesn’t do much but allocate local variables, perform statements, and returns a value:

```c
int fubar( int a, int b ) {
    long  local_stack_var1[6];
    long *local_stack_var2;
    local_stack_var2 = local_stack_var1 + 4;  /* statement 1 */
    *local_stack_var2 = 420420;               /* statement 2 */
    --local_stack_var2;                       /* statement 3 */
    b = local_stack_var1[4];                  /* statement 4 */
    return ( a + local_stack_var1[1] );       /* statement 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.** Draw a line between groups of instructions to indicate which instructions are associated with each C statement. (15 points)

```assembly
.global fubar
.section " .text"
fubar: /* Your unoptimized code goes below this point */
```
7. Load/Store/Memory
What gets printed in the following program? (9 points)

.global main

.sect "".data"
fmt:    .asciz "0x%x\n" ! prints value as hex 0xXXXXXXXX

.c:     .byte 0xBB

.s:     .half 0x87AC

.il:    .word 0x12345678
.i2:    .word 0x12345678
.i3:    .word 0x12345678
.x:     .word 0

.sect "".text"
main:
    save  %sp, -96, %sp
    set   i1, %l0
    set   s, %l1
    ldsb  [%l1+1], %l1
    sth  %l1, [%l0+2]
    set   fmt, %o0
    ld    [%l0], %o1
    call  printf _________________________________
    nop
    set   i2, %l0
    set   c, %l1
    ldub  [%l1], %l1
    stb  %l1, [%l0+1]
    set   fmt, %o0
    ld    [%l0], %o1
    call  printf _________________________________
    nop
    set   i3, %l0
    set   x, %l1
    ldub  [%l0+3], %l2
    sth  %l2, [%l1]
    ldsh  [%l0], %l2
    sth  %l2, [%l0+2]
    mov  %l1, %l0
    set   fmt, %o0
    ld    [%l0], %o1
    call  printf _________________________________
    nop
    ret
    restore
Extra Credit (5 points)
Consider the following recursive function.

```c
void enigma( int n )
{
    if ( n >= 2 )
        enigma( n / 3 );
    printf( "%d ", n ); /* Output the value of n followed by a space */
}
```

What gets printed if the function is called as `enigma( 37 )`? (2 pts)

Now translate this C function into SPARC assembly. Optimize as best you can. Depending on how you code this, you may not be able to eliminate all nops. Best optimizations will result in fewest number of instructions. (3 pts)