CSE 30
Winter 2005
Final Exam

1. Number Systems ___________________  (25 points)
2. Binary Addition/Condition Code Bits/Overflow Detection ___________________  (12 points)
3. Branching ___________________  (22 points)
4. Bit Operations ___________________  (13 points)
5. Recursion/SPARC Assembly ___________________  (10 points)
6. Local Variables, The Stack, Return Values ___________________  (24 points)
7. More Recursive Subroutines ___________________  (16 points)
8. Floating Point ___________________  (12 points)
9. Machine Instructions ___________________  (20 points)
10. Linkage, Scope, Lifetime, Data ___________________  (32 points)
11. Load/Store/Memory ___________________  (9 points)
12. Miscellaneous ___________________  (29 points)

SubTotal ___________________  (224 points)
Extra Credit ___________________  (11 points)
Total ___________________
1. Number Systems

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

\begin{itemize}
  \item \text{binary} \hspace{1cm} \\
  \item \text{octal} \hspace{1cm} \\
  \item \text{decimal} \hspace{1cm}
\end{itemize}

Convert \text{297}_{10} to the following (assume 16-bit word). \textbf{Express answers in hexadecimal.} (3 points)

\begin{itemize}
  \item \text{sign-magnitude} \\
  \item \text{1’s complement} \\
  \item \text{2’s complement}
\end{itemize}

Convert \text{-574}_{10} to the following (assume 16-bit word). \textbf{Express answers in hexadecimal.} (6 points)

\begin{itemize}
  \item \text{sign-magnitude} \\
  \item \text{1’s complement} \\
  \item \text{2’s complement}
\end{itemize}

Testing

In PA3 & PA4 you wrote compare functions to compare specific values. You passed a pointer to one of these compare functions to qsort() so qsort() could use this compare function to perform the type of comparison you defined. For some comparisons (like comparing integers) a simple subtraction of the values to be compared is not sufficient ( \text{return val1 – val2}; ). Using integer comparison as a specific example, under what circumstances would subtraction not result in the correct return value? (2 points)

Write an \text{intCompare()} function that can be passed to qsort() to properly sort an array of integers keeping in mind you cannot use simple subtraction because of the above. qsort() function prototype: (8 points)

\begin{verbatim}
void qsort(void *base, size_t nel, size_t width, int (*compar)(const void *, const void *));
\end{verbatim}
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)

\[
\begin{array}{c c c c c}
10101010 & + & 11010110 & = & \text{N Z V C} \\
01110110 & + & 11001101 & = & \text{N Z V C} \\
11111111 & + & 00000001 & = & \text{N Z V C}
\end{array}
\]

3. Branching

Write the SPARC Assembly leaf subroutine to perform the following C function. Use only standard looping/branching format as detailed in class and class notes. Comment! Do not optimize. (22 points)

C

```c
int checkIfPowerOf2( unsigned int value )
{
    int i;
    int cnt = 0;
    unsigned int mask = 0x80000000;

    for ( i = 0; i < 32; ++i )
    {
        if ( (value & mask) != 0 )
            ++cnt;

        if ( cnt > 1 )
            return 0;

        mask = mask >> 1;
    }

    return cnt;
}
```

SPARC ASSEMBLY

```assembly
.global checkIfPowerOf2
.section "text"
checkIfPowerOf2:
    ! i mapped to _____
    ! cnt mapped to _____
    ! mask mapped to _____
```
4. Bit Operations

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

```
set 0xCA4273BD, %l0
set 0x9035768A, %l1
and %l0, %l1, %l0
Value in %l0 is _______________________________________ (2 points)
```

```
set 0xCA4273BD, %l0
sra %l0, 13, %l0
Value in %l0 is _______________________________________ (2 points)
```

```
set 0xCA4273BD, %l0
sll %l0, 9, %l0
Value in %l0 is _______________________________________ (2 points)
```

```
set 0xCA4273BD, %l0
set 0x?????????, %l1
btog %l1, %l0 ! Value in %l0 is now 0xFEEDBEEF
Value set in %l1 must be this bit pattern _______________________________________ (3 points)
```

```
set 0xCA4273BD, %l0
set 0x9035768A, %l1
or %l0, %l1, %l0
Value in %l0 is _______________________________________ (2 points)
```

```
set 0xCA4273BD, %l0
srl %l0, 7, %l0
Value in %l0 is _______________________________________ (2 points)
```
5. Recursion/SPARC Assembly
Given `main.s` and `recurse.s`, what gets printed when executed? (10 points)

```
.global main
/* main.s */
.section ".text"
main:
    save  %sp, -96, %sp
    set   14289, %o0
    call  recurse
    nop
    ret
    restore

.global recurse
/* recurse.s */
.section ".rodata"
fmt:   .asciz  "\%d\n"
.section ".text"
recurse:
    save  %sp, -(92 + 4) & -8, %sp
    mov   %i0, %o0
    mov   10, %o1
    call  .rem
    nop
    st    %o0, [%fp - 4]
    mov   %i0, %o0
    mov   10, %o1
    call  .div
    nop
    mov   %o0, %i0
    set   fmt, %o0
    ld    [%fp - 4], %o1
    call  printf
    nop
    cmp   %i0, %g0
    be    base
    nop
    mov   %i0, %o0
    call  recurse
    nop
base:
    set   fmt, %o0
    mov   %i0, %o1
    call  printf
    nop
    ret
    restore
```
6. Local Variables, The Stack, and Return Values
Here is a C function that allocates a few local variables, performs some assignments and returns a value. Don’t worry about any local variables not being initialized before being used. Just do a direct translation. **Draw lines.**

```c
int fubar( short x, long y ) {
    char   *local_stack_var1;
    short  *local_stack_var2;
    struct foo {
        short s1;
        char s2[3];
        short s3[3];
        int s4;
    }       local_stack_var3;
    local_stack_var1    = &local_stack_var3.s2[2]; /* 1 */
    local_stack_var2    = local_stack_var3.s3 + y; /* 2 */
    local_stack_var3.s1 = y + *++local_stack_var2; /* 3 */
    return ( local_stack_var3.s4 + x ); /* 4 */
}
```

Write the equivalent full **unoptimized** SPARC assembly language module to perform the equivalent. **You must allocate all local variables on the stack.** No short cuts. Treat each statement independently. (24 points)
What is the output of the following program? (16 pts)

```c
#include <stdio.h>

int mystery1( int x )
{
    int result;

    printf( "x = %d\n", x );

    if ( x <= 0 )
        return 0;
    else {
        result = mystery2( x - 1 );
        printf( "result = %d\n", result );
        return result;
    }
}

int mystery2( int x )
{
    int result;

    printf( "x = %d\n", x );

    if ( x <= 0 )
        return 0;
    else {
        result = (mystery1( x - 1 ) + x);
        printf( "result = %d\n", result );
        return result;
    }
}

int main( int argc, char *argv[] )
{
    printf( "%d\n", mystery1( 7 ) );
    return 0;
}
```
8. Floating Point

Convert -119.125\textsubscript{10} (decimal fixed-point) to binary fixed-point (binary) and single-precision IEEE floating-point (hexadecimal) representations.

binary fixed-point __________________________________   (2 points)
IEEE floating-point __________________________________   (4 points)

Convert 0x4348C000 (single-precision IEEE floating-point representation) to fixed-point decimal.

fixed-point decimal __________________________________   (6 points)

9. Machine Instructions

Translate the following instructions into SPARC machine code. Use hexadecimal values for your answers. If an instruction is a branch, specify the number of instructions away for the target (vs. a Label).

\texttt{and} %i2, %l3, %o4 ___________________________________  (5 points)
\texttt{std} %o2, [%fp - 8] ___________________________________  (5 points)

Translate the following SPARC machine code instructions into SPARC assembly instructions.

0x3CBFFFF8 ___________________________________  (5 points)
0xEC0B4014 ___________________________________  (5 points)
10. Linkage, Scope, Lifetime, Data

For the following program fragment, specify what C runtime area/segment will be used for each variable definition or statement: (32 points — 1 point each)

```c
static int a = 411;               ______________
int b;                           ______________
int c = 404;                      ______________
static int d;                     ______________

int foo( int e ) {               ____________ (foo) ______________ (e)
    double f = 4.20;              ______________
    static int g = 8675309;       ______________
    static int *h;               ______________
    h = (int *) malloc( b );     ______________ (where h is pointing)
    int (*i)(int) = foo;         ____________ (i) ______________ (where i is pointing)

    ...                         }
```

Fill in the letter corresponding to the correct scoping/visibility for each of the variables:

- A) Global across all modules/functions linked with this source file.
- B) Global just to this source file.
- C) Local to function foo().

<table>
<thead>
<tr>
<th>Variable</th>
<th>Scoping/Visibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>c</td>
<td>d</td>
</tr>
<tr>
<td>d</td>
<td>e</td>
</tr>
<tr>
<td>e</td>
<td>f</td>
</tr>
<tr>
<td>f</td>
<td>g</td>
</tr>
<tr>
<td>g</td>
<td>h</td>
</tr>
<tr>
<td>h</td>
<td>i</td>
</tr>
<tr>
<td>i</td>
<td>foo</td>
</tr>
</tbody>
</table>

Fill in the letter corresponding to the correct lifetime for each of the variables:

- A) Exists from the time the program is loaded to the point when the program terminates.
- B) Exists from the time function foo() is called to the point when foo() returns.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Lifetime</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>c</td>
<td>d</td>
</tr>
<tr>
<td>d</td>
<td>e</td>
</tr>
<tr>
<td>e</td>
<td>f</td>
</tr>
<tr>
<td>f</td>
<td>g</td>
</tr>
<tr>
<td>g</td>
<td>h</td>
</tr>
<tr>
<td>h</td>
<td>i</td>
</tr>
<tr>
<td>i</td>
<td>foo</td>
</tr>
</tbody>
</table>
11. Load/Store/Memory
What gets printed in the following program? (9 points)

.global main

.section ".data"
fmt: .asciz "0x%08X\n"         ! prints value as hex  0xXXXXXXXX

c: .byte 0x77
.s: .half 0xBEEF
.i1: .word 0x89ABCD12
.i2: .word 0x89ABCD12
.i3: .word 0x89ABCD12
.x: .word 0x00005555

.section ".text"
main:
save %sp, -96, %sp

set   i1, 10
set   s, 11
ldsh  [%l1], 11
stb   %l1, [%10]
sth   %l1, [%10+2]

set   fmt, 00
ld    [%10], 1
call  printf ________________________
nop

set   i2, 10
set   c, 11
ldsb  [%l1], 11
sth   %l1, [%10]
stb   %l1, [%10+2]

set   fmt, 00
ld    [%10], 1
call  printf ________________________
nop

set   x, 11
set   i3, 10
ldsb  [%l1+3], 12
sth   %l2, [%10+2]
stb   %l2, [%10]

set   fmt, 00
ld    [%10], 1
call  printf ________________________
nop

ret
restore
12. Miscellaneous

Given the following type definitions:

```c
struct foo1
{
    int a;
    short b;
    double c;
    char d;
    long e;
};

union foo2
{
    int a;
    short b;
    double c;
    char d;
    long e;
};
```

What is the sizeof( struct foo1 )? _______  What is the sizeof( union foo2 )? _______  

(2 points)  (2 points)

What gets printed with the statements below?

```c
unsigned short x = 0xF024;
putchar( (x & 0xF) + '0' );
putchar( (x << 8 >> 12) + '0' );
putchar( (x & 0xF00) + '0' );
putchar( (x >> 12) + '0' );
```

______________  (4 points)

Where is the location of the Martini Bar having a computer trivia night this weekend? (1 pt)

Given the following program, order the printf() lines so that the values that are printed when run on a Sun SPARC Unix system are displayed from smallest value to largest value. (2 points each)

```c
void foo( int, int ); /* Function Prototype */
static int a = 311;

int main( int argc, char *argv[] ) {
    static int b;
    int c = 69;
    foo( argc, b );

    / 1 */ (void) printf( "argc --> %p\n", &argc );
    / 2 */ (void) printf( "foo --> %p\n", foo );
    / 3 */ (void) printf( "a --> %p\n", &a );
    / 4 */ (void) printf( "malloc --> %p\n", malloc(50) );
    / 5 */ (void) printf( "c --> %p\n", &c );
    / 6 */ (void) printf( "b --> %p\n", &b );
}

void foo( int d, int e ) {
    int f = e;
    int g = f;

    / 7 */ (void) printf( "e --> %p\n", &e );
    / 8 */ (void) printf( "g --> %p\n", &g );
    / 9 */ (void) printf( "d --> %p\n", &d );
    / 10 */ (void) printf( "f --> %p\n", &f );
}
```
Extra Credit (11 points)

Optimized Version

```
.global foo
.section "."text

foo:
  save %sp, -96, %sp
  mov %g0, %l0

  cmp %i0, %i1
  be L3
  nop

  bl L2
  nop

L1:
  inc %i1
  inc %l0
  cmp %i0, %i1
  bne L1
  nop
  mov %l0, %i1
  ret
  restore

L2:
  inc %i0
  inc %l0
  cmp %i0, %i1
  bne L2
  nop
  mov %l0, %i1
  neg %i0
  ret
  restore

L3:
  mov %g0, %i0
  ret
  restore
```

Output if called as `foo( 123456, 123459)` ________

Output if called as `foo( 9, 4)` ________

Output if called as `foo( 123456, 123456)` ________

Now optimize the code (in the box above to the right) to eliminate the nops.
| 00 NUL | 01 SOH | 02 STX | 03 ETX | 04 EOT | 05 ENQ | 06 ACK | 07 BEL |
| 08 BS  | 09 HT  | 0A NL  | 0B VT  | 0C NP  | 0D CR  | 0E SO  | 0F SI  |
| 10 DLE | 11 DC1 | 12 DC2 | 13 DC3 | 14 DC4 | 15 NAK | 16 SYN | 17 ETB |
| 18 CAN | 19 EM  | 1A SUB | 1B ESC | 1C FS  | 1D GS  | 1E RS  | 1F US  |
| 20 SP  | 21 !   | 22 "   | 23 #   | 24 $   | 25 %   | 26 &   | 27 ’   |
| 28 (   | 29 )   | 2A *   | 2B +   | 2C ,   | 2D -   | 2E .   | 2F /   |
| 30 0   | 31 1   | 32 2   | 33 3   | 34 4   | 35 5   | 36 6   | 37 7   |
| 38 8   | 39 9   | 3A :   | 3B ;   | 3C <   | 3D =   | 3E >   | 3F ?   |
| 40 @   | 41 A   | 42 B   | 43 C   | 44 D   | 45 E   | 46 F   | 47 G   |
| 48 H   | 49 I   | 4A J   | 4B K   | 4C L   | 4D M   | 4E N   | 4F O   |
| 50 P   | 51 Q   | 52 R   | 53 S   | 54 T   | 55 U   | 56 V   | 57 W   |
| 58 X   | 59 Y   | 5A Z   | 5B [   | 5C \   | 5D ]   | 5E ^   | 5F _   |
| 60 `   | 61 a   | 62 b   | 63 c   | 64 d   | 65 e   | 66 f   | 67 g   |
| 68 h   | 69 i   | 6A j   | 6B k   | 6C l   | 6D m   | 6E n   | 6F o   |
| 70 p   | 71 q   | 72 r   | 73 s   | 74 t   | 75 u   | 76 v   | 77 w   |
| 78 x   | 79 y   | 7A z   | 7B {   | 7C | 7D }   | 7E ~   | 7F DEL |
Scratch Paper