CSE 30
Spring 2004
Final Exam

1. Number Systems ___________________ (15 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, and Return Values ___________________ (24 points)
7. SPARC Leaf Subroutines ___________________ (18 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 ___________________ (33 points)

SubTotal ___________________ (220 points)
Extra Credit ___________________ (10 points)
Total ___________________
1. Number Systems

Convert $\text{FB6}_{16}$ (2’s complement, 16-bit word) to the following. (6 points)

binary_______________________________________
octal  _______________________________________
decimal _______________________________________

Convert $-594_{10}$ to the following (assume 16-bit word). **Express answers in hexadecimal.** (6 points)

sign-magnitude________________________________________
1’s complement________________________________________
2’s complement________________________________________

Convert $375_{10}$ to the following (assume 16-bit word). **Express answers in hexadecimal.** (3 points)

sign-magnitude________________________________________
1’s complement________________________________________
2’s complement________________________________________
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)

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>11010110</td>
<td>+10111011</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10111001</td>
<td>+01010110</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01011101</td>
<td>+01101001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<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>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3. Branching

Write the C statements to perform the following SPARC Assembly statements. No gotos. Use only standard looping and conditional statements (for/while/if). Do not optimize. (22 points)

```c
int x;
int y;

! map x to %l2
! map y to %l6

mov 25, %l2
mov 15, %l6
cmp %l6, 75
bg L1
nop

L3:
mov %l6, %o0
call bar
nop
mov %o0, %l2
cmp %l2, %l6
bg L2
nop
mov %l6, %o0
mov 5, %o1
call .rem
nop
mov %o0, %l2
L2:
add %l6, 3, %l6
cmp %l6, 75
gle L3
nop
L1:
```
4. Bit Operations

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

set 0xFA420DED, %l0
set 0x8675309A, %l1
or %l0, %l1, %l0

Value in %l0 is _______________________________________  (2 points)

set 0xFA420DED, %l0
sll %l0, 9, %l0

Value in %l0 is _______________________________________  (2 points)

set 0xFA420DED, %l0
srl %l0, 13, %l0

Value in %l0 is _______________________________________  (2 points)

set 0xFA420DED, %l0
set 0x?????????, %l1
btog %l1, %l0 ! Value in %l0 is now OxFEEDBabe

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

set 0xFA420DED, %l0
set 0x8675309A, %l1
and %l0, %l1, %l0

Value in %l0 is _______________________________________  (2 points)

set 0xFA420DED, %l0
sra %l0, 7, %l0

Value in %l0 is _______________________________________  (2 points)
Given `main.s` and `recurse.s`, what gets printed when executed? (10 points)

```assembly
.global main
/* main.s */
.section ".text"
main:
save   %sp, -96, %sp
mov    4, %o0
call   recurse
nop
ret
restore

.global recurse
/* recurse.s */
.section ".text"
.recurse:
save   %sp, -(92 + 4) & -8, %sp ! One int local var
cmp    %i0, %g0
bge    else
nop
mov    1, %l0
st     %l0, [%fp-4]
ba     endif
nop
else:
sub    %i0, 1, %o0
call   recurse
nop
sll    %o0, 1, %o0
st     %o0, [%fp-4]
endif:
set    fmt, %o0
mov    %i0, %o1
ld     [%fp-4], %o2
call   printf
nop
ld     [%fp-4], %i0
ret
restore
```

5. Recursion/SPARC Assembly
6. Local Variables, The Stack, and Return Values

Here is a C function that allocates a couple 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
long fubar( short x, long y ) {
    short   local_stack_var1;
    int    *local_stack_var2;
    struct foo {
        int    s1[4];
        char   s2;
        short  s3;
        long   s4;
    }       local_stack_var3;

    local_stack_var3.s4 = y + *local_stack_var2++; /* 1 */
    local_stack_var2 = &local_stack_var3.s1[2];    /* 2 */
    local_stack_var1 = local_stack_var3.s2 + x;     /* 3 */
    return ( x + local_stack_var3.s3 );            /* 4 */
}
```

Now 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)
7. SPARC Leaf Subroutines

Write a full unoptimized leaf SPARC assembly function translation of the following C function to determine if a positive value is evenly divisible by some other positive value or not. Return 1 to indicate true; return 0 to indicate false. **Be sure to state which registers you are using.** (18 pts)

For example, \(\text{isEvenlyDivisible}( 18, 4 )\) will return 0
\(\text{isEvenlyDivisible}( 125, 5 )\) will return 1

\[
\begin{align*}
\text{C} \\
\text{int isEvenlyDivisible( unsigned int value, unsigned int base )} \\
\{ \text{do} \{ \\
\quad \text{value} = \text{value} - \text{base}; \\
\text{while} ( \text{value} > 0 ); \\
\} \text{if ( value } == \text{0 )} \text{return} \text{1}; \text{else} \text{return} \text{0;} \\
\}
\end{align*}
\]
8. Floating Point

Convert \(-124.625_{10}\) (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 0x43446000 (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).

xorcc %g6, -14, %o3 __________________________ (5 points)
sth %o3, [%l5 + %i4] __________________________ (5 points)

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

0xF6130017 __________________________ (5 points)
0x36BFFFFC __________________________ (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
int a;

static int b = 911;
static int c = 404;
int d = 0;

int foo( int e ) {     (foo) (e)
    int f = 420;
    static double g = 4.20;
    int (*h)(int) = foo; (h) (where h is pointing)
    static int *i;
    i = (int *) malloc( b ); (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>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
<th>h</th>
<th>i</th>
<th>foo</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
<th>h</th>
<th>i</th>
<th>foo</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

11. Load/Store/Memory

What gets printed in the following program? (9 points)

```assembly
.global main

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

.c: .byte 0x99
    .align 2
s: .half 0xBABE
    .align 4
i1: .word 0xAB88CDEF
i2: .word 0xAB88CDEF
i3: .word 0xAB88CDEF
x: .word 0x00004444

.section ".text"
main:
    save %sp, -96, %sp
    set i1, %l0
    set s, %l1
    ld uh [%l1], %l1
    st %l1, [%l0]
    st b %l1, [%l0+1]
    set fmt, %o0
    ld [%l0], %o1
    call printf ________________________
    nop

    set i2, %l0
    set c, %l1
    lds b [%l1], %l1
    st h %l1, [%l0+2]
    st b %l2, [%l0]
    set fmt, %o0
    ld [%l0], %o1
    call printf ________________________
    nop

    set x, %l0
    set i3, %l1
    lds b [%l1+1], %l2
    st h %l2, [%l0]
    st b %l2, [%l0+2]
    set fmt, %o0
    ld [%l0], %o1
    call printf ________________________
    nop

    ret
    restore
```
12. Miscellaneous

At the end of the last lecture, the class chose a not-so-secret bonus point password. What is it? (1 pt)

List two events that will cause a full context switch. (4 points)

1)

2)

Put the following in the correct order/sequence using the numbers to the left of each word: (8 points)

1. loader 5. compiler
2. executable (.exe/a.out) 6. assembler
3. program execution 7. preprocessor
4. source code 8. linker

——— —— —— —— —— —— —— —— —— —— —— —— —— —— —— ——

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 ); /* Function Prototype */

int a = 911;

int main( int argc, char *argv[] ) {
    int b;
    int c = 405;

    foo( c);

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

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

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

prints smallest value

prints

largest value
Extra Credit
Optimize the following code fragment. Some optimizations are better than others. Go for the fewest machine cycles assuming memory accesses are several more cycles than other non-memory access instructions. You cannot change the overall algorithm. (10 points)

set 5525, %12
mov 15, %16
cmp %16, 75
bg L1
nop

L3:
ldub [%l3+5], %14
mov %16, %o0
call bar
nop
mov %o0, %12
cmp %12, %16
bg L2
nop
mov %16, %o0
mov 128, %o1
call .mul
nop
mov %o0, %12

L2:
add %16, 3, %16
cmp %16, 75
ble L3
nop

L1:
mov %12, %o0
ret
restore
| 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 |