CSE 30
Winter 2012
Midterm Exam

1. Number Systems ___________________ (15 points)
2. Binary Addition/Condition Code Bits/Overflow Detection ___________________ (12 points)
3. Branching ___________________ (24 points)
4. Bit Operations / C Runtime Environment ___________________ (18 points)
5. Parameter Passing and Return Values (Stack Variables) ___________________ (12 points)
6. Local Variables, The Stack and Return Values ___________________ (16 points)
7. Load/Store/Memory ___________________ (11 points)

SubTotal ___________________ (108 points)

Extra Credit ___________________ (7 points)

Total ___________________
1. Number Systems

Convert \(0xFBF1\) (2’s complement, 16-bit word) to the following. (6 points)

- **binary**: 
- **octal**: 0 
- **decimal**: 

Convert -437 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 +318 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)

\[
\begin{array}{c}
11010111 \\
+10101001 \\
\hline
\end{array} \quad \begin{array}{c}
01101101 \\
+00111001 \\
\hline
\end{array} \quad \begin{array}{c}
01001001 \\
+10110111 \\
\hline
\end{array}
\]

\[
\begin{array}{cccccc}
N & Z & V & C \\
| & | & | & | \\
\hline
| & | & | & | \\
\hline
| & | & | & | \\
\hline
\end{array} \quad \begin{array}{cccccc}
N & Z & V & C \\
| & | & | & | \\
\hline
| & | & | & | \\
\hline
| & | & | & | \\
\hline
\end{array} \quad \begin{array}{cccccc}
N & Z & V & C \\
| & | & | & | \\
\hline
| & | & | & | \\
\hline
| & | & | & | \\
\hline
\end{array}
\]
3. **Branching** (24 points)
Translate the C code below into the equivalent unoptimized SPARC Assembly code using the control flow specified in class. Just perform a direct translation – no optimizations. Use the local register mappings for the variables in assembly as specified.

```c
/* Assume variables x and y have been properly defined as ints. */

if ( x < 42 )
{
    x = y % x;
}
else
{
    /* x += 4 same as x = x + 4 */
    for ( x = (-8888); x <= y; x += 4 )
    {
        y = x + 42;
    }
    x = y;
}
```

```sparc
! x is mapped to %l2
! y is mapped to %l6
```
4. Bit Operations / C Runtime Environment

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

set 0xDEADC0DE, %l0
sra %l0, 9, %l0

Value in %l0 is 0x

set 0xDEADC0DE, %l0
sll %l0, 11, %l0

Value in %l0 is 0x

set 0xDEADC0DE, %l0
set 0x????????, %l1
xor %l0, %l1, %l0 ! Value in %l0 is now 0xCAFEBABE

Value set in %l1 must be this bit pattern 0x

What character gets printed with each putchar() line? An ASCII char chart is on page 8.

unsigned int x = 0xA613;
putchar( (((x << 8) & 0xFFFF) >> 12) + '0' ); ______
putchar( (x & 0xF) + '0' ); ______
putchar( (x ^ x) + '1' ); ______
putchar( '0' - ((x & 0xF00)>> 8) ); ______

In PA2, which bit operation did you use to correctly implement the toggle operation? ______

In PA2, if the first command specified is

> set 0x1234ABCD 012345676543

followed by

> rotate 5

what hex values will be in the two light banks? (4 points)

lightBank[0] lightBank[1]

Using the Rt-Lt Rule, write the C variable definition for the variable named foo that is an array of 7 elements where each element is a pointer to a function that takes a pointer to char and returns a pointer to struct fubar.
5. Parameter Passing and Return Values (Local Stack Variables)

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 */
short foo( char, int, unsigned short );

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

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

struct fubar {
    short    a;
    char     b[3];
    int      c[3];
    unsigned short d[4];
} fb; /* Local variable fb */

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

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

fb.a = foo( fb.b[1], fb.c[0], fb.d[3] );
```

---

**Put your SPARC Assembly code in the box below.**
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 x, int y ) {
    int local_stack_var1[6];
    int *local_stack_var2;

    x = local_stack_var1[3];                   /* statement 1 */
    local_stack_var2 = &local_stack_var1[0];   /* statement 2 */
    local_stack_var1[2] = *local_stack_var2;   /* statement 3 */
    *local_stack_var2++ = 24680;               /* statement 4 */
    return ( local_stack_var1[5] + y );        /* 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. (16 points)

```sparc
.global fubar
.section "text"
fubar: /* Your unoptimized code goes below this point */
```

```sparc
"Your unoptimized code goes below this point"
7. Load/Store/Memory Specify the full 32 bit hex values after each line has been fully executed. (11 points)

```
.global main

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

.c: .byte 0xAA
.s: .half 0x9753

.i1: .word 0xABCD1234
.i2: .word 0xABCD1234
.i3: .word 0xABCD1234
.x: .word 0x22220000

.section ".text"
main: save %sp, -96, %sp
.set i1, %l0
.set s, %l1
.lds %l0, %l2
.sth %l2, [%l0+2]
.sll %l2, 8, %l2
.sth %l2, [%l0]

.set fmt, %o0
.l [%l0], %o1
call printf
.nop

.set x, %l0
.set c, %l1
.lds %l1, %l2
.sth %l2, [%l0+2]
.stb %l2, [%l0+1]

.set fmt, %o0
.l [%l0], %o1
call printf
.nop

.set i2, %l0
.set i3, %l1
.lduh %l1, %l2
.stb %l2, [%l0+3]
.sll %l2, 12, %l2
.sth %l2, [%l0]

.set fmt, %o0
.l [%l0], %o1
call printf
.nop

ret
restore
```
Extra Credit (7 points)

What gets printed at each printf() statement given the following C program?

```c
#include <stdio.h>

int main()
{
    char a[] = "DATA/BSS";
    char *p = a;

    printf( "\%c", *p++ );
    printf( "\%c", *(p+5) = *p);  
    printf( "\%c", +++p );
    printf( "\%c", *++p );
    printf( "\%d", ++p - a );

    printf( "\n%s\n", a );
    return 0;
}
```

A portion of the C Operator Precedence Table

<table>
<thead>
<tr>
<th>Operator</th>
<th>Associativity</th>
</tr>
</thead>
<tbody>
<tr>
<td>++ postfix increment</td>
<td>L to R</td>
</tr>
<tr>
<td>-- postfix decrement</td>
<td>R to L</td>
</tr>
<tr>
<td>[] array element</td>
<td></td>
</tr>
<tr>
<td>() function call</td>
<td></td>
</tr>
<tr>
<td>* indirection</td>
<td>R to L</td>
</tr>
<tr>
<td>++ prefix increment</td>
<td>L to R</td>
</tr>
<tr>
<td>-- prefix decrement</td>
<td></td>
</tr>
<tr>
<td>&amp; address-of</td>
<td></td>
</tr>
<tr>
<td>sizeof size of type/object</td>
<td></td>
</tr>
<tr>
<td>(type) type cast</td>
<td></td>
</tr>
<tr>
<td>/ division</td>
<td></td>
</tr>
<tr>
<td>% modulus</td>
<td></td>
</tr>
<tr>
<td>+ addition</td>
<td>L to R</td>
</tr>
<tr>
<td>- subtraction</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td></td>
</tr>
<tr>
<td>- assignment</td>
<td>R to L</td>
</tr>
</tbody>
</table>

Hexadecimal - Character

|   | 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 NF | 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 0 | 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