CSE 30
Fall 2000
Final Exam

1. Number Systems ___________________  (15 points)
2. Binary Addition/Condition Code Bits/Overflow Detection ___________________  (12 points)
3. Branching ___________________  (20 points)
4. Bit Operations ___________________  (10 points)
5. SPARC Assembly, Parameter Passing, and Return Values ___________________  (20 points)
6. Local Variables, The Stack, and Return Values ___________________  (20 points)
7. SPARC Subroutines and Calling Convention ___________________  (24 points)
8. Floating Point ___________________  (14 points)
9. Linkage, Scope, Lifetime, Data ___________________  (26 points)
10. Machine Instructions ___________________  (20 points)
11. I/O & Virtual Memory ___________________  (12 points)
12. ALU & Control Unit ___________________  (14 points)
13. Miscellaneous ___________________  (14 points)
14. Programming Errors ___________________  (9 points)

SubTotal ___________________  (230 points)
Extra Credit ___________________  (10 points)
Total ___________________
1. Number Systems

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

- octal ________________________________
- binary ______________________________
- decimal ______________________________

Convert \(-153_{10}\) to the following (assume 16-bit word). **Express answers in hexadecimal.** (6 points)

- sign-magnitude_______________________________________________
- 1’s complement_______________________________________________
- 2’s complement_______________________________________________

Convert \(+136_{10}\) 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)

\[
\begin{array}{c}
11010110 + 10111011 = --------- \\
1011001 + 01010110 = --------- \\
01011101 + 01101001 = --------- \\
\end{array}
\]

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>
N  | Z  | V  | C  |
---|----|----|----|
|   |   |   |   |

3. Branching

Fill in the SPARC assembly instructions to perform the following statements. Do not optimize. (16 points)

```c
int x = 37;                   ! x mapped to %l0
while ( x >= 21 ) {
    statement1;
    --x;
    statement2;
}
```

For the following instruction sequence, mark with an X the conditional branch instructions which would transfer control to loop if used in place of ba. (4 points)

<table>
<thead>
<tr>
<th>Instruction sequence</th>
<th>be</th>
<th>bne</th>
<th>bcs</th>
<th>bleu</th>
<th>ble</th>
<th>bge</th>
<th>bpos</th>
</tr>
</thead>
<tbody>
<tr>
<td>mov</td>
<td>5, %10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>addcc</td>
<td>%10, -2, %10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ba</td>
<td>loop</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.**

```plaintext
set 0x87651234, %l0
set 0x43218765, %l1
and %l0, %l1, %l0

Value in %l0 is _______________________________________ (2 points)

set 0x87651234, %l0
sll %l0, 7, %l0

Value in %l0 is _______________________________________ (2 points)

set 0x87651234, %l0
sra %l0, 11, %l0

Value in %l0 is _______________________________________ (2 points)

set 0x56781234, %l0
set 0xF1413121, %l1
xor %l1, %l0, %l0

Value in %l0 is _______________________________________ (2 points)

set 0x56781234, %l0
set 0x8A423121, %l1
bset %l1, %l0

Value in %l0 is _______________________________________ (2 points)
```
5. SPARC Assembly, Parameter Passing, and Return Values

Write a full unoptimized SPARC assembly function to determine whether every even numbered bit is set or not in the single parameter passed to it. Return 1 to indicate true/yes; return 0 to indicate false/no. (20 points)

Function prototype for this function: \( \text{int checkBits( long value );} \)
6. Local Variables, The Stack, and Return Values

Here is a C function that allocates a local struct, performs some assignments and returns one of the params:

```c
int fubar( int a, int b ) {

    struct foo {
        int s1[5];
        char s2;
        short s3;
    } local_stack_var1;
    long local_stack_var2 = 5;

    local_stack_var1.s1[3] = 5432;
    local_stack_var1.s2 = '?'; /* Use the ASCII value, not '?' */

    b = local_stack_var1.s3; /* Yes, s3 is uninitialized - bad coding */

    return ( a + local_stack_var2 );
}
```

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. (20 points)
7. SPARC Subroutines and Calling Convention

The _______ registers are mapped to the _______ registers as part of the save instruction. (4 points)

The _______ instruction saves the current value of %pc in %o7. (2 points)

The retl instruction adds _______ to the value in _______ and stores the result in _______. (6 points)

This instruction is almost always placed in the delay slot of the ret instruction. _______ (2 points)

If we have a save instruction, we will always have a _______ instruction. (2 points)

If we have a call instruction, we will always have a _______ or _______ instruction. (4 points)

When passed more than 6 arguments to a function, the SPARC architecture calling convention dictates the called routine will access the 7th argument at ______ offset relative to the ______ pointer. (4 points) [Specify the absolute +/- value]

8. Floating Point

Convert -48.875 (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 0xC319A000 (single-precision IEEE floating-point representation) to fixed-point decimal.

fixed-point decimal ____________________________ (6 points)

Why are the exponents of IEEE floating-point values stored in biased/excess notation? (2 point)
9. Linkage, Scope, Lifetime, Data

For the following program fragment, specify what C runtime area/segment will be used for each variable definition or statement: (26 points)

```c
int x = 4; ______________
static int a = 4; ______________
int y; ______________
static int b; ______________

int
main( int argc, char *argv[] ) { ______________ (argc & argv)
    float *c; ______________
    int d = 9; ______________
static int e; ______________
    static int f = 8; ______________

    c = (float *) calloc( 3, sizeof(float) ); ______________ (where c 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 main().

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td></td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td></td>
<td>A</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 main() is called to the point when main() returns.

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td></td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td></td>
<td>A</td>
</tr>
</tbody>
</table>
10. 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).

\[
\text{sub \ %l2, 44, \%i1} \quad \text{______________________________} \quad (5 \text{ points})
\]

\[
\text{ldsh \ [%o4], \%i2} \quad \text{______________________________} \quad (5 \text{ points})
\]

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

\[
\text{0x3CBFFF5A} \quad \text{______________________________} \quad (5 \text{ points})
\]

\[
\text{0x929D0008} \quad \text{______________________________} \quad (5 \text{ points})
\]

11. I/O & Virtual Memory

Order by general access speed the following data storage types. [ 1 is fastest, 6 is slowest ] (6 points)

SASD ______
L1 cache______
RAM ______
DASD ______
L2 cache_____ 
registers ______

List 3 ways/events which would cause a context switch to occur. (6 points)

1)

2)

3)
12. ALU and Control Unit

The two common types of control units are ________________ and _________________. (2 points)

RISC processors try to execute _____ instruction(s) every machine cycle. (2 points)

A device which can electronically connect and disconnect itself from the bus is called ________________ (2 points)

Fill in the letter corresponding to the correct answer for each of the following:

______ sequential logic/circuit (2 points)
______ asynchronous (2 points)
______ combinational logic/circuit (2 points)
______ synchronous (2 points)

A. depends on timing by clock pulses or control signals
B. output is only a function of input
C. output is a function of input and some previous saved state
D. depends only on input changes and not timing by clock pulses or control signals

13. Miscellaneous

A ______-Endian machine stores the MSB (most significant byte) of a word at a lower memory location than the LSB (least significant byte). (2 points)

A Latch or Flip-Flop is an example of __________________________ logic circuit. (2 points)

When promoting a smaller data type to a larger data type (smaller number of bits to larger number of bits), we use ________________ data types to ensure zero extension. (2 points)

Increasing the number of _________ bits in the IEEE floating-point representation allows us to represent larger values. (2 points)

What two major program entities are stored in the Stack Frame? (4 points)

1)

2)

RISC architecture binaries are usually ____________ in size than equivalent CISC architecture binaries. (2 points)
14. Programming Errors

There are a few things wrong with this program such that it won’t pass lint as it is. Put on your compiler and lint hat and list the problems/errors/things that are wrong which prevents this from passing lint without warnings. Assume the general logic and basic syntax is correct. Assume this is the only module to compile, therefore it is a fully self-contained program. (9 points)

```c
#include <stdio.h>

int main( void ) {
    char buf[BUFSIZ] = {"CSE 30 Rocks\n"};
    int i;

    while ( buf[i] != '\0' )
        printf( "%c", buf[i++] );
}
```
Extra Credit (10 points)

The following program initializes some data and prints it out. The printing section is commented out. Optimize this program without changing what the program does so that after your optimizations, I could uncomment the printing section and the optimized program will print exactly the same output as the unoptimized program. Some optimizations are better than others and are worth more points. Change only code following the save instruction.

```assembly
.global main
A_ELEMENTS = 50 ! array a has 50 elements
A_ELEMENT_SIZE = 4 ! sizeof( int ) = 4 bytes
A_OFFSET = A_ELEMENTS * A_ELEMENT_SIZE ! array a’s offset

.section ".rodata"
fmt: .asciz   "%d\n"

.section ".text"
main:
    save %sp, -(92 + (A_OFFSET)) & -8, %sp  ! local variable int a[50];
    clr %l0
    ba test1
    nop
loop1:
    mov %l0, %o0
    mov A_ELEMENT_SIZE, %o1
    call .mul
    nop
    mov %o0, %l1
    add %l1, -A_OFFSET, %l1
    st %l0, [%fp + %l1]

print section
******
.

inc %l0

test1:
    cmp %l0, A_ELEMENTS
    bl loop1
    nop
    mov 0, %l0
ret

restore
```
/* Print the section of extra credit */

mov %l0, %o0
mov A_ELEMENT_SIZE, %o1
call .mul
nop

mov %o0, %l2
add %l2, -A_OFFSET, %l2
set fmt, %o0
ld [%fp + %l2], %o1
call printf
nop

******

Hexadecimal - Character

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