CSE 30
Winter 2007
Final Exam

1. Number Systems ___________________  (15 points)

2. Binary Addition/Condition Code Bits/Overflow Detection ___________________  (12 points)

3. Branching ___________________  (17 points)

4. Bit Operations ___________________  (13 points)

5. Recursion/SPARC Assembly ___________________  (10 points)

6. Local Variables, The Stack, ld/st, and Return Values ___________________  (28 points)

7. Bit Slinging ___________________  (12 points)

8. Floating Point ___________________  (12 points)

9. Machine Instructions ___________________  (20 points)

10. Linkage, Scope, Lifetime, Data ___________________  (34 points)

11. Load/Store/Memory ___________________  (9 points)

12. Miscellaneous ___________________  (35 points)

SubTotal ___________________  (217 points)

Extra Credit ___________________  (10 points)

Total ___________________  
1. Number Systems
Convert 0xFBA9 (2’s complement, 16-bit word) to the following. (6 points)

- binary ____________________________ (straight bit pattern translation)
- octal ____________________________ (straight bit pattern translation)
- decimal __________________________ (pos/neg decimal value from 2’s complement encoding)

Convert 309₁₀ to the following (assume 16-bit word). **Express answers in hexadecimal.** (3 points)

- sign-magnitude ____________________________
- 1’s complement ____________________________
- 2’s complement ____________________________

Convert -428₁₀ to the following (assume 16-bit word). **Express answers in hexadecimal.** (6 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)

\[
\begin{array}{cccc}
00011011 & +01101011 & = & 10110100 \\
\hline
N & Z & V & C \\
\hline
| & | & | & |
\end{array}
\]

\[
\begin{array}{cccc}
01010101 & +11011001 & = & 10110110 \\
\hline
N & Z & V & C \\
\hline
| & | & | & |
\end{array}
\]

\[
\begin{array}{cccc}
10110110 & +01001010 & = & 11111100 \\
\hline
N & Z & V & C \\
\hline
| & | & | & |
\end{array}
\]
3. Branching

Write the SPARC assembly statements to perform the following C statements. **Do not optimize nops.**
Be sure to properly `ld/st` local vars. Remember logical OR short circuiting. (17 points)

C

```c
int a;
int b;

/* other stmts here giving a and b values */

if ( a != 15 || b < 35 )
{
    a = foo( b % 17 );
}
else
{
    b = a - b;
}

/* other stmts here */
```

SPARC assembly

```assembly
save    %sp, -(92 + 8) & -8, %sp

! local var a mapped to %fp - 4
! local var b mapped to %fp - 8

```

4. Bit Operations

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

```plaintext
set 0xCAFE2007, %l0
set 0x867B5309, %l1
or  %l0, %l1, %l0

Value in %l0 is ________________________________  (2 points)

set 0xCAFE2007, %l0
sll  %l0, 9, %l0

Value in %l0 is ________________________________  (2 points)

set 0xCAFE2007, %l0
srl  %l0, 9, %l0

Value in %l0 is ________________________________  (2 points)

set 0xCAFE2007, %l0
set 0x?????????, %l1
xor  %l0, %l1, %l0  ! Value in %l0 is now 0xFEEDFACE

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

set 0xCAFE2007, %l0
set 0x867B5309, %l1
and  %l0, %l1, %l0

Value in %l0 is ________________________________  (2 points)

set 0xCAFE2007, %l0
sra  %l0, 11, %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
.section ".text"
main:
    save  %sp, -96, %sp
    mov   6, %o0
    call  fubar
    nop
    ret
    restore

.global fubar
.section ".text"
fubar:
    save  %sp, -96, %sp
    cmp   %i0, 1
    ble   L1
    nop
    sub   %i0, 1, %o0
    call  fubar
    nop
    mov   %i0, %o1
    call  .mul
    nop
    mov   %o0, %i0
    set   fmt, %o0
    mov   %i0, %o1
    call  printf
    nop
    ba    L2
    L1:
    mov   1, %i0
    L2:
    ret
    restore
```

Output ____________________________
6. Local Variables, The Stack, and Return Values
Here is a C function that allocates a couple local variables, performs some assignments, 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 ( int x, short *y ) {  
    int *local_stack_var1;
    struct foo {
        char s1[3];
        short s2;
        short s3;
        short *s4;
    } local_stack_var2;
    char local_stack_var3;

    local_stack_var3 = local_stack_var2.s1[0] + 0x20; /* 1 */
    local_stack_var2.s4 = &local_stack_var2.s3; /* 2 */
    local_stack_var2.s2 = *y; /* 3 */

    return ( x + ++*local_stack_var1 ); /* 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. (28 points)
7. Bit Slinging

Write a C function named hi_lo_byteSwap() that takes a word (unsigned int) and returns the word with the high and low bytes swapped. (12 pts)

For example, hi_lo_byteSwap( 0x2A4567C9 ) will return the value 0xC945672A
- high byte 2A is swapped with low byte C9

hi_lo_byteSwap( 0xE812343B ) will return the value 0x3B1234E8
- high byte E8 is swapped with low byte 3B

```c
unsigned int
hi_lo_byteSwap( unsigned int word )
```
8. Floating Point

Convert 144.125_{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 0xC35FE000 (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).

std %i2, [%o2 + %l3] __________________________________ (5 points)
addcc %i3, -19, %l6 __________________________________ (5 points)

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

0xAB32A009 __________________________________ (5 points)
0xC8000010 __________________________________ (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: (34 points — 1 point each)

```c
int a;
static int b = 42;
static int c;
int d = 420;

int foo( int e ) {
    double f = -3.33;
    static int *g;
    static int h;
    int (*i)(int) = foo;
    g = (int *) malloc( d );
    ...
}
```

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>A</td>
</tr>
<tr>
<td>b</td>
<td>B</td>
</tr>
<tr>
<td>c</td>
<td>C</td>
</tr>
<tr>
<td>d</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td></td>
</tr>
<tr>
<td>f</td>
<td></td>
</tr>
<tr>
<td>g</td>
<td></td>
</tr>
<tr>
<td>h</td>
<td></td>
</tr>
<tr>
<td>i</td>
<td></td>
</tr>
<tr>
<td>foo</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>Variable</th>
<th>Lifetime</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>A</td>
</tr>
<tr>
<td>b</td>
<td>B</td>
</tr>
<tr>
<td>c</td>
<td>C</td>
</tr>
<tr>
<td>d</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td></td>
</tr>
<tr>
<td>f</td>
<td></td>
</tr>
<tr>
<td>g</td>
<td></td>
</tr>
<tr>
<td>h</td>
<td></td>
</tr>
<tr>
<td>i</td>
<td></td>
</tr>
<tr>
<td>foo</td>
<td></td>
</tr>
</tbody>
</table>

If function foo() is called 7 times, how many times does the variable `f` get initialized to -3.33? _______

In function foo(), what is the initial value of `h`? _______
11. Load/Store/Memory

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

```
.globa l main

.se ction ".data"
fmt:    .asci z "0x%08X\n"         ! prints value as hex  0xFFFFFFFF

c:      .byte   0x9A

.s ection ".text"
main:
  save  %sp, -96, %sp
  set   i1, %10
  set   s, %11
  ldsh  [%11], %11
  st   %11, [%10]
  stb  %11, [%10+1]
  set   fmt, %o0
  ld   [%10], %o1
  call  printf

  set   i2, %10
  set   c, %11
  ldub  [%11], %12
  stb  %12, [%10]
  sth  %12, [%10+2]
  set   fmt, %o0
  ld   [%10], %o1
  call  printf

  set   x, %10
  set   i3, %11
  ldsb  [%11+1], %12
  sth  %12, [%10]
  stb  %12, [%10+3]
  set   fmt, %o0
  ld   [%10], %o1
  call  printf

ret
```
12. Miscellaneous

Put the following in the correct order/sequence using the letters to the left of each stage. (11 pts)

A. program execution  B. compiler  C. executable file (.exe/a.out)
D. C preprocessor  E. assembler  F. C source code
G. Core Dump (Segmentation Fault)  H. linker/linkage editor  I. loader
J. resulting .s file  K. resulting .o file


Give the equivalent C expression for each of the following without using array access square brackets [ ]. (4 pts)

\(a[2]\)  _____________
\&a[0]  _____________
\&a[2]  _____________
a[0]  _____________

When the following program is run on a Sun SPARC Unix system and sorted by the address printed with the \%p format specifier, specify the order of the lines printed from smallest value to largest value. (2 points each)

```c
void foo( int, int ); /* Function Prototype */
int x;

int main( int argc, char *argv[] ) {
    int y = 7;
    static int z = 11;
    foo( argc, z );

    /* 1 */ (void) printf( "1: y --> %p\n", &y );
    /* 2 */ (void) printf( "2: malloc --> %p\n", malloc(20);
    /* 3 */ (void) printf( "3: z --> %p\n", &z );
    /* 4 */ (void) printf( "4: argc --> %p\n", &argc );
    /* 5 */ (void) printf( "5: foo --> %p\n", foo );
}

void foo( int a, int b ) {
    int c;
    int d;

    /* 6 */ (void) printf( "6: c --> %p\n", &c );
    /* 7 */ (void) printf( "7: x --> %p\n", &x );
    /* 8 */ (void) printf( "8: b --> %p\n", &b );
    /* 9 */ (void) printf( "9: d --> %p\n", &d );
```

Smallest value
/ * 10 */ (void) printf("10: a --> %p\n", &a);

Extra Credit
Optimize the following assembly code fragment. Some optimizations are worth more than others. (up to 9 pts)

/* other code here */

```assembly
/* other code here */

l1:  
  ld    [%fp – 16], %l0
  cmp   %l0, 42
  bne   L1
  nop

  ld    [%fp – 32], %l1
  cmp   %l1, -42
  bge   L2
  nop

L1:
  mov   %l0, %o0
  mov   128, %o1
  call  .mul
  nop

  st    %o0, [%fp – 40]
  ba    L3
  nop

L2:
  add   %l0, %l1, %l0

L3:
  st    %l0, [%fp – 48]

/* other code here */
```

Tell me something you learned in this class that is extremely valuable to you and that you think you will be able to use for the rest of your programming/computer science career. (1 point if serious; you can add non-serious comments also)
<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