**NAME:**

Login name:

---

# Computer Science 217 Second Midterm Test April 19, 2002 2PM-4PM

---

This test is 9 questions. Put your name on every page, and write out and sign the Honor Code pledge before turning in the test.

``I pledge my honor that I have not violated the Honor Code during this examination."

| Question | Score |
|---|---|
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| 8 | |
| 9 | |
| **Total** | |

## ***QUESTION 1*** *(6 POINTS)*

a) Convert the following decimal numbers to Binary, Octal, and Hexadecimal.
Use only as many digits as necessary to represent each number.

| | Binary | Octal | Hexadecimal |
|---|---|---|---|
| 82 = | | | |
| 31 = | | | |

b) Convert the following decimal numbers to 2's Complement Binary, 1's Complement Binary, and Sign Magnitide Binary. Write 8 bit results.

| | Signed Magnitude | 1's Complement | 2's Complement |
|---|---|---|---|
| 22 = | | | |
| -22 = | | | |
| 64 = | | | |
| -64 = | | | |

c) What are the advantages and disadvantages of ...

i) Sign Magnitude Representation

ii) 1's Complement Representation

iii) 2's Complement Representation

## ***QUESTION 2*** *(6 POINTS)*

For each of the following (possibly hypothetical) synthetic instructions and pseudo-ops, please write Sparc assembly language code that implements the described functionality. Please do not use other synthetic instructions (e.g., set).

a) `double reg`
Replace the integer in `reg` with twice the value

b) `neg reg`
Replace the integer in `reg` with its negative, assuming a two's complement representation

c) `setall reg`
Load the value 0xFFFFFFFF into `reg`

d) `callfunc reg`
Call the function whose address is in `reg`

f) `.zero`
Reserve space for `1` byte in the current section and initialize it to zero

g) `.intarray N`
Allocates space for an array of N 4-byte integers in the current section.

## *QUESTION 3* *(12 POINTS)*

The following instructions are two ways to return from a subroutine:

(i) `jmpl %o7+8, %g0`

(ii) `jmpl %i7+8, %g0`

a) When is each of these instructions used (one or two sentences)?

b) For the first of the two return instructions (labeled (i)), describe the sequence of operations executed during the subroutine invocation that are typically used to provide a value in `%o7`.

c) For the second of the two return instructions (labeled (ii)), describe the sequence of operations executed during the subroutine invocation that are typically used to provide a value in `%i7`.

d) Why is there a "`+8`" in each of these instructions?

e) Why is register `%g0` used?

f) Why can't we use branch instructions to return from subroutines, in general?

## ***QUESTION 4*** *(12 POINTS)*

Write a subroutine in SPARC assembly language to compute N factorial.
First convert the “orignal” C source code (provided below) into “expanded” C source code that contains no program control structures (except `goto`s). Then, convert your expanded C source code into Sparc assembly code. Your Sparc assembly code should expect its argument (N) in register `%o0`, and it should leave its return value in `%o0`. You can use the leaf subroutine `.mul` that takes two arguments and returns their product. You can assume N is small enough that the result will fit into a single register. You will get full credit even if your code is not optimized.

**Original C Source Code:**

```
int factorial(int N)
{
     int i;
     int fact = 1;
     for (i=1; i<N; i++)
           fact *= i;
     return fact;
}
```

**Expanded C Source Code:** **Sparc Assembly Code:**

## ***QUESTION 5*** *(12 POINTS)*

a) What is the "annul bit"? How is it used? (two sentences):

b) Why shouldn't a leaf subroutine modify register `%o7`? (one sentence):

c) Describe the main function of an assembler (one sentence):

d) Describe the two main functions of a linker (two sentences):

e) Describe how a user level process performs functions that
execute privileged instructions (one sentence).

f) What is the main reason modern operating systems can execute processes in short time,
whereas only batch processing was practical a few decades ago? (one sentence):

## ***QUESTION 6*** *(12 POINTS)*

a) Write a sequence of Sparc assembly language instructions that require a load delay slot (indicate the delay slot with a nop).

b) Explain why a load delay slot is needed.

c) Write an optimized version of the following assembly code, assuming the following macro definitions:

```
#define sum %g1
#define array %g2
#define N %g3
#define i %l0
#define offset %l1
#define value %l2
```

**<u>Original Sparc Assembly Code:</u>** **<u>Optimized Sparc Assembly Code:</u>**

```
      clr i
      clr sum
test: cmp i, N
      bge done
      nop
      sll i, 2, offset
      ld [array+offset], value
      add sum, value, sum
      inc i
      ba test
      nop
done: ...
```

## ***QUESTION 7*** *(8 POINTS)*

For each of the following fields stored in the Processor State Register (PSR), provide a short description of when/how they are modified and how they are used.

a) ICC - Integer Condition Codes (4 bits)

b) S - Supervisor Mode Enabled (S) (1 bit)

c) ET - Traps Enabled (1 bit)

d) CWP - Current Window Pointer (5 bits)

## ***QUESTION 8*** (16 POINTS)

a) Convert the following C code into Sparc assembly language. Include terse comments to help with grading. Also, on the next page, draw a diagram of the stack and label how it is allocated at the time that the program reaches the line of code commented *"... show stack here ..."*. For this exercise, make sure you allocate and access all local variables on the stack. You will get full credit even if your code is not optimized. *(hint: you may want to draw the stack on the next page before writing assembly code).*

**Original C Source Code:**

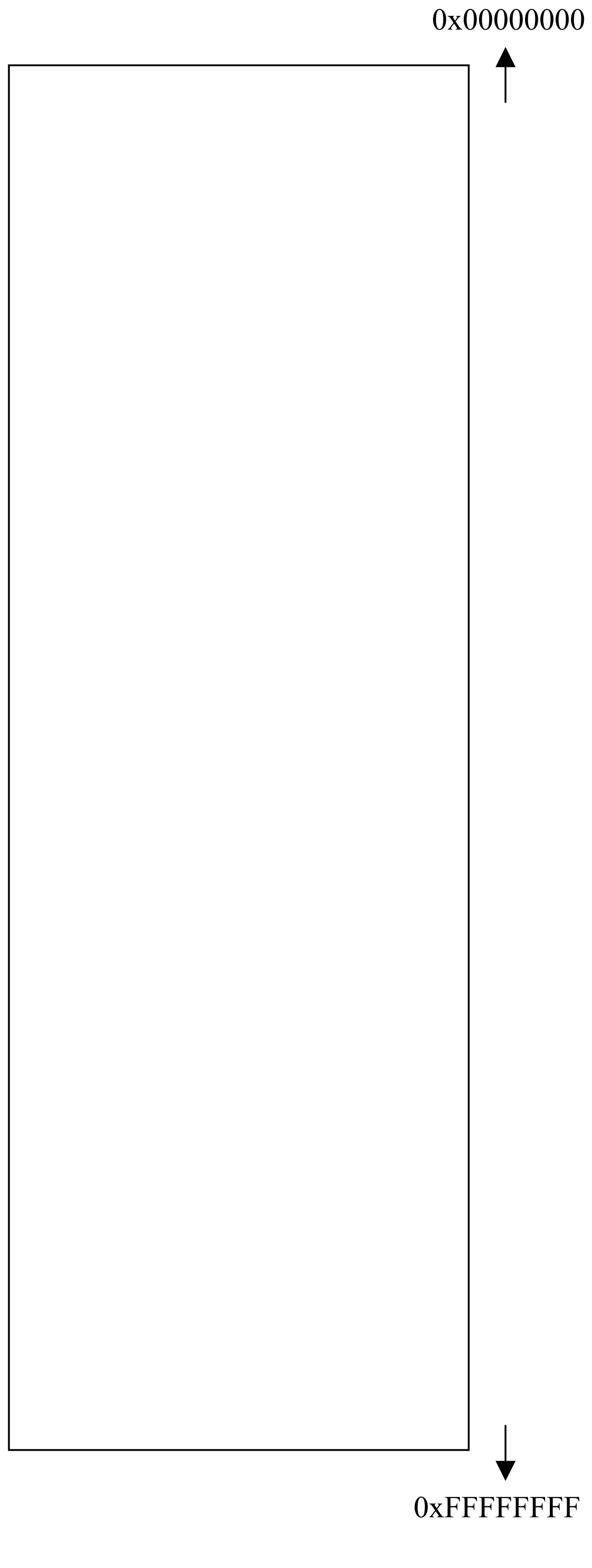
```
int g(int a1, int a2, int a3, int a4, int a5, int a6, int a7, int a8, int a9)
{
     int b1 = 1;
     int b2 = 2;
     /* ... show stack here ... */
     return b1 + b2 + a1 + a9;
}

int f(void)
{
     return g(0, 1, 2, 3, 4, 5, 6, 7, 8);
}
```

**Sparc Assembly Code for "f":**

**Sparc Assembly Code for "g":**

## *STACK FOR QUESTION 8*

b) Provide a diagram of the call stack when the line of code commented *"... show stack here ..."* (from previous page) is reached. Label the locations of `%fp` and `%sp`, and show how regions within the stack frames for subroutines `"f"` and `"g"` are allocated.



## ***QUESTION 9*** *(16 POINTS)*

Fill in the boxes (on the following two pages) showing the text and data sections that the `as` assembler will produce when given this Sparc assembly language program. Fill any bit that cannot be determined by the assembler with a question mark (?). As an example, an appropriate answer for the first instruction of each section has been provided in the first line of each table. Please annotate your answers with comments as shown in the examples.

```
          .section ".data"
          .ascii "COS"
          .align 2
          .byte 1
          .skip 7
var2:     .asciz "IS"
          .ascii "GR"
          .word 8

          .section ".text"
          add %r8, %r9, %r10
          add %r10, 22, %r10
          call addthree
          bg label1
          sethi %hi(var2), %r8
          or %r8, %lo(var2), %r8
          call printf
label1:   ld [%r5], %r7
          ld [%r5 + %r6], %r8
          ld [%r5 + 3], %r9
          ret

addthree: add %r8, %r9, %r8
          retl
          add %r8, %r10, %r8
```

## *DATA SECTION FOR QUESTION 9*

| Offset | Machine Code in Hexadecimal | Comment |
|---|---|---|
| 0 | 0x43 | C .ascii "COS" |
| 1 | 0x4F | O |
| 2 | 0x53 | S |
| 3 | | |
| 4 | | |
| 5 | | |
| 6 | | |
| 7 | | |
| 8 | | |
| 9 | | |
| 10 | | |
| 11 | | |
| 12 | | |
| 13 | | |
| 14 | | |
| 15 | | |
| 16 | | |
| 17 | | |
| 18 | | |
| 19 | | |
| 20 | | |
| 21 | | |
| 22 | | |
| 23 | | |
| 24 | | |
| 25 | | |
| 26 | | |
| 27 | | |
| 28 | | |
| 29 | | |
| 30 | | |

## *TEXT SECTION FOR QUESTION 9*

| **Offset** | **Machine Instruction in Binary**<br>**(comment for each opcode and operand)** |
|---|---|
| 0 | 10 01010 000000 01000 0 00000000 01001<br>(format=3b) (rd=10) (op3=add) (rs1=8) (rs2=9) |
| 4 | |
| 8 | |
| 12 | |
| 16 | |
| 20 | |
| 24 | |
| 28 | |
| 32 | |
| 36 | |
| 40 | |
| 44 | |
| 48 | |
| 52 | |
| 56 | |
| 60 | |