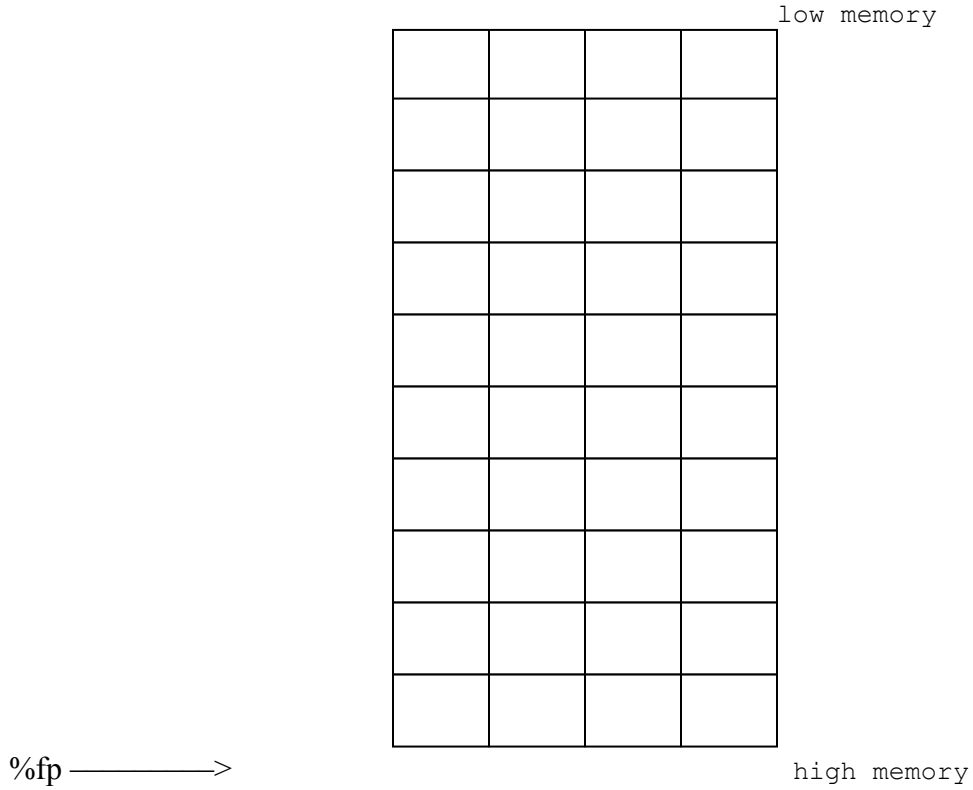
Login name _____ **Quiz 5** **Name** _____

CSE 131B

Signature _____ **Winter 2003** **Student ID** _____

1. Show the memory layout of the following local variables allocated on the runtime Stack taking into consideration the SPARC data type memory alignment restrictions discussed in class. Fill bytes in memory with the appropriate local variable name. For example, if local variable name `p` takes 4 bytes, you will have 4 `p`'s in the appropriate memory locations. If the local variable is an array, use the name followed by the index number. For example, some number of `p0`'s, `p1`'s, `p2`'s, etc. Place an `X` in any bytes of padding.

```
char   a;
short  b;
int    c;
short  d[3];
double e;
```



Write the SPARC assembly instructions that would be generated by the following instruction using the above memory layout of the local variables. Make no assumptions about values already loaded into any registers. All local variable accesses must access the local variable space allocated on the Stack. Just give a straightforward suboptimal code generation to perform this operation. Hint: This should take 4 instructions.

c = 420420 + c;

_____

_____

_____

_____

2. Given the following code for foo(), write an equivalent more highly optimized version in SPARC assembly.
Assume: a is mapped to local register %l0
b is mapped to local register %l1
c is mapped to local register %l2
x is a global variable allocated in the Data segment and NOT mapped to a register

Oberon

```
VAR x : INTEGER;

PROCEDURE foo( i : INTEGER );
  VAR a, b, c : INTEGER;

BEGIN
  a := 15;
  b := a + 10;
  c := i + b;

  x := c;
  a := x;
  b := x;
  x := c;
(* Other code may access a,b,c,x *)
END foo;

BEGIN
  foo( 5 );
END.
```

SPARC Assembly

```
        .global  main, foo

        .section ".data"
        .align 4
x:      .word 0

        .section ".text"
foo:                                ! mapping local vars
        save    %sp, -96, %sp       ! to local regs

        mov     15, %l0             ! r0 = 15
        add     %l0, 10, %l1        ! r1 = r0 + 10
        add     %i0, %l1, %l2       ! r2 = param1 + r1

        set     x, %l3              ! x = r2
        st      %l2, [%l3]
        set     x, %l3              ! r0 = x
        ld      [%l3], %l0
        set     x, %l3              ! r1 = x
        ld      [%l3], %l1
        set     x, %l3              ! x = r2
        st      %l2, [%l3]

        /* other code may access a,b,c,x */
        ret
        restore

main:
        /* Code for main() not important */
```

Rewrite only code that is in the bounds of the rectangle.

What question would you most like to see on the Final?