Login name ________________ **Name** ______________________

Quiz 4
CSE 131
Winter 2008

Signature ___________________ **Student ID** __________________

1. Project II – Phase I.1:
Write the SPARC Assembly code that would be generated for the following Reduced-C statement:

```
cout << 17 << "World";
```

You can assume the following is available for you to use:

```
        .section  ".data"
intFmt: .asciz    "%d"
strFmt: .asciz    "%s"
```

Assume `x` is defined as the first local variable on the stack for some function as:

```
float x;
```

Write the SPARC Assembly code that would be generated for the following Reduced-C statement:

```
cout << x;
```

2. Pick of one the following letters to answer the questions below.

1) Prologue
2) Epilogue
3) Pre-Call
4) Post-Return

_____ Where local variable space is allocated

_____ Where parameter space is allocated

_____ Store return value in %i0 in SPARC subroutine

_____ Saves the return address

_____ Performs initialization of local variables

_____ Where local variable space is deallocated

_____ Where parameter space is deallocated

_____ Retrieves saved return address

_____ Restores caller-save registers

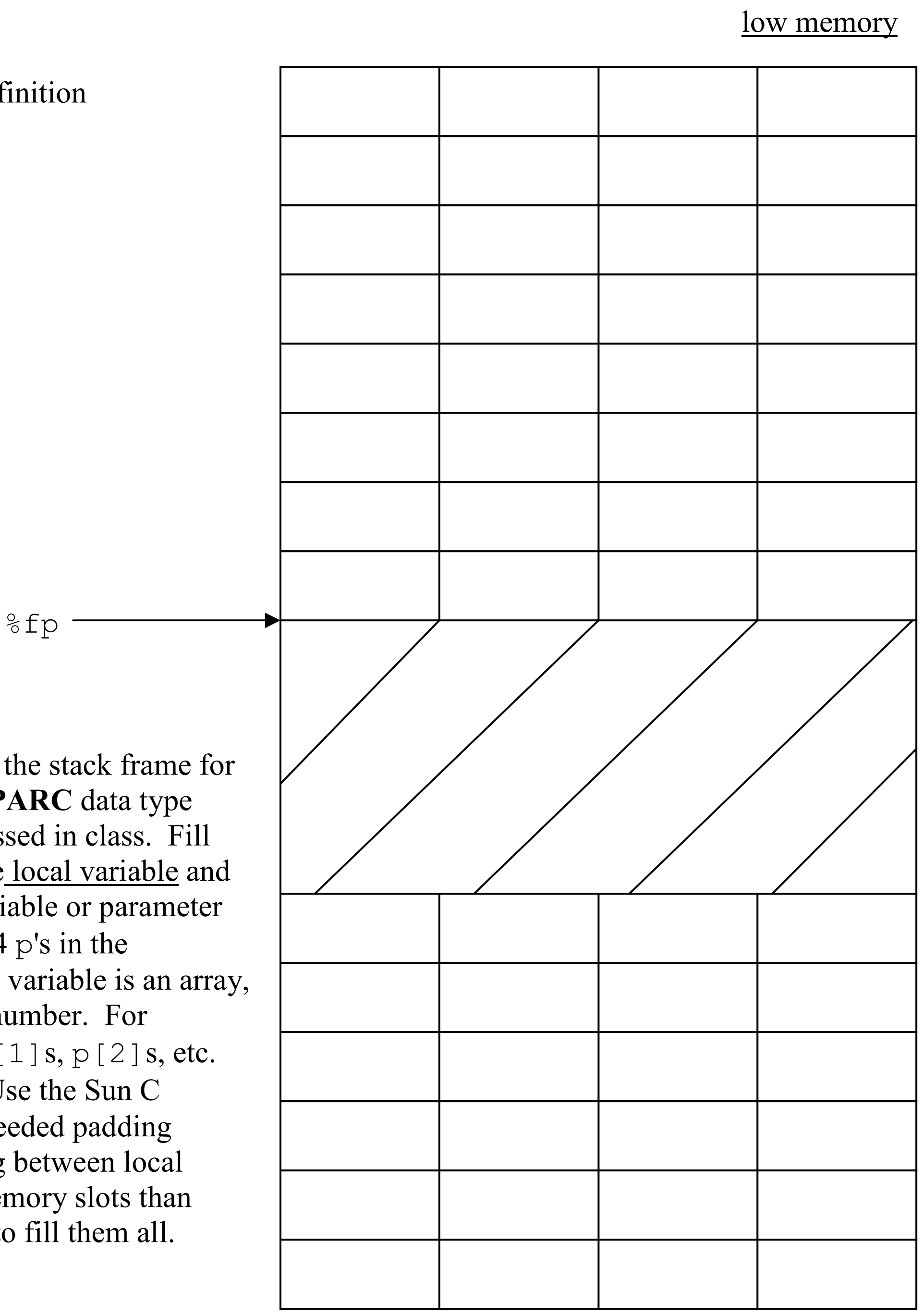
_____ Saves callee-save registers

_____ Retrieve return value from %o0 in SPARC subroutine

3. Given the following C function definition

```
void foo( int a, int b )
{
    char   c[3];
    short  d;
    int    e;
    double f;
    int    g;

    /* function body */
}
```



Show the **SPARC** memory layout of the stack frame for foo() taking into consideration the **SPARC** data type memory alignment restrictions discussed in class. Fill bytes in memory with the appropriate local variable and parameter name. For example, if variable or parameter name `p` takes 4 bytes, you will have 4 `p`'s in the appropriate memory locations. If the variable is an array, use the name followed by the index number. For example, some number of `p[0]`s, `p[1]`s, `p[2]`s, etc. Place an `X` in any bytes of padding. Use the Sun C compiler model. Do not allocate unneeded padding similar to how gcc puts extra padding between local variables. There is probably more memory slots than needed, so do not feel like you have to fill them all.