Login name _____ **Quiz 4** **Name** _____

CSE 131B

Signature _____ **Spring 2004** **Student ID** _____

1. In the box at the lower right, state the order the C compiler (and your Oberon compiler) might perform operations to access a random array element **a[i][j]** for an arbitrary two-dimensional array (you may use each number more than once; there may be more than one correct sequence, but only state one sequence you want us to grade; the groupings of operations are merely to help group similar operations together and does not imply one or more operations should be selected from each grouping). Suggestion: write down the storage map equation first.

1) Calculate the address of the 1st element of the array a and store the result in Reg0

2) Multiply i times the number of rows in a and store the result in Reg1
3) Multiply i times the numbers of cols/row in a and store the result in Reg1
4) Multiply i times the size of each array element and store the result in Reg1
5) Multiply i times j and store the result in Reg1
6) Multiply i times Reg0 and store the result in Reg1

7) Multiply Reg1 times the number of rows in a and store the result in Reg1
8) Multiply Reg1 times the numbers of cols/row in a and store the result in Reg1
9) Multiply Reg1 times the size of each array element and store the result in Reg1
10) Multiply Reg1 times j and store the result in Reg1
11) Multiply Reg1 times Reg0 and store the result in Reg1

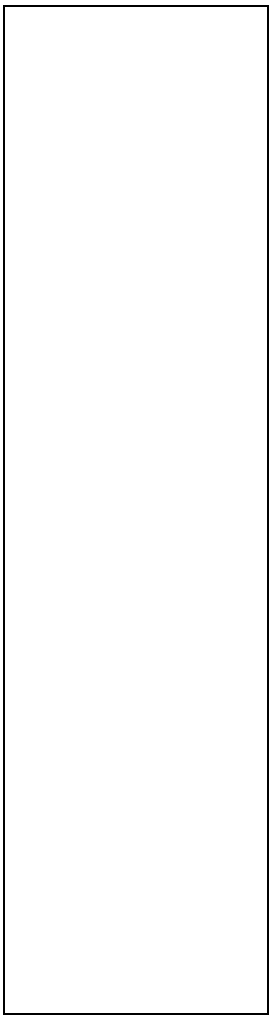
12) Multiply j times the number of cols in a and store the result in Reg2
13) Multiply j times the numbers of cols/row in a and store the result in Reg2
14) Multiply j times the size of each array element and store the result in Reg2
15) Multiply i times j and store the result in Reg2
16) Multiply j times Reg0 and store the result in Reg2
17) Multiply j times Reg1 and store the result in Reg2

18) Multiply Reg2 times the number of cols in a and store the result in Reg2
19) Multiply Reg2 times the numbers of cols/row in a and store the result in Reg2
20) Multiply Reg2 times the size of each array element and store the result in Reg2
21) Multiply Reg2 times i and store the result in Reg2
22) Multiply Reg2 times Reg0 and store the result in Reg2
23) Multiply Reg2 times Reg1 and store the result in Reg2

24) Add Reg1 to Reg2 and store the result in Reg0
25) Add Reg1 to Reg2 and store the result in Reg1

26) Add Reg0 to Reg1 and store the result in Reg0
27) Add Reg0 to Reg2 and store the result in Reg0

28) Dereference Reg0 and store the result in Reg0
29) Dereference Reg1 and store the result in Reg0
30) Dereference Reg2 and store the result in Reg0



In general, when making a function call the compiler must generate code to perform various duties on the way in and on the way out of the function call. These duties are usually divided up between the **caller** (the code making the function call) and the **called** (the code in the function being called) functions. Generally, anything the caller does to build part of the stack frame the caller needs to undo. Likewise for the called code.

List two operations that the caller performs either directly or indirectly to build the part of the stack frame/activation record it is generally responsible for.

1)

2)

List two operations that the called function performs either directly or indirectly to build the part of the stack frame/activation record it is generally responsible for.

1)

2)

In the SPARC architecture, where does the caller retrieve the return value that is being returned by the function it just called?

What would a bare bones oberon.s file look like after compiling an Oberon program that has nothing except a BEGIN END.