CS 164, Spring 2009 **CS 164: Homework #6** **P. N. Hilfinger**

Due: Friday, 1 May 2008

1. I produced the following program using `gcc -S foo.c` (with an older version of gcc):

```
.globl f
    .type   f, @function
f:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    movl    $0, -8(%ebp)
    jmp     .L2
.L3:
    movl    -8(%ebp), %eax
    sall    $2, %eax
    addl    8(%ebp), %eax
    movl    (%eax), %eax
    addl    %eax, -4(%ebp)
    incl    -8(%ebp)
.L2:
    movl    -8(%ebp), %eax
    cmpl    12(%ebp), %eax
    jl      .L3
    movl    -4(%ebp), %eax
    leave
    ret
```

Produce a plausible definition (in C) of function `f`, one that might have produced this output. The function does return a value.

2. In lecture, we talked about *array descriptors,* which are data structures containing all the information one needs to access (get the address of) an array element `A[i,j]` in an implementation that allocates all elements of a new array contiguously. In C, multidimensional arrays are composed of rows of rows, so that `A[i,j]` (or `A[i][j]` in C) is located at address$(A_{0,0}) + M \cdot S \cdot i + S \cdot j$, where the array in `A` is $M \times N$ and each element has size S. Thus, the three constants data address$(A_{0,0})$ (the virtual origin), $M \cdot S$ (the row stride), and S (the column stride) can be precomputed into an *array descriptor,* which the program can use to generate array accesses and can pass as a parameter to functions that expect to receive the array as a by-reference parameter. Show the IL code that you'd use to access array element `A[i][j]`, assuming that the d, t_i, and t_j are IL registers containing the address of the array descriptor for `A`, the value of `i`, and the value of `j`.

3. These exercises involve operations on array descriptors to give different view of an array. Just describe the calculations; we don't need actual IL code.

a. Suppose that a certain array descriptor contains the information (VO, S_1, S_2) for accessing two-dimensional array `B`. Show how to create a new array descriptor that accesses column number j of `B`. This will be a one-dimensional array descriptor (having only one stride).

b. Show how to create a new array descriptor that accesses the transpose of `B`.

c. Show how to create a new array descriptor (for array view `B'`) that accesses the rows and columns of `B` in reverse, so that `B'[0,0]` is the same as the last column of the last row of `B`.