EECS665

Compiler Construction

Drew Davidson
Ruturaj Vaidya

Lecture: LEEP2 G415
MWF 3:00-3:50

Lab: Eaton 1005B

ANOUNCEMENTS | LAB | SCHEDULE | MATERIALS | ASSIGNMENTS

Homework 10

Due on December 5th @ 3:00 PM (in class, to Drew, or at Engineering front office)

Not accepted late

ALL homework must be done individually

Question 1

Assume we have a language similar to that features arrays and that parameters in this language can be passed by value or passed by reference. Consider the following program:

```
int A[4];
int k;

void f( int x, int y ) {
    x = x + 1;
    k = k + 1;
```

```
    A[k] = A[k] * 2;
    y = y + 1;
    cout << x;
    cout << y;
    cout << k;
    cout << A[k];
}

void main() {
    k = 0;
    while (k < 4) {
        A[k] = k;
        k = k + 1;
    }
    k = 0;
    f(k, A[k]);
    cout << k;
    cout << A[0];
    cout << A[1];
    cout << A[2];
    cout << A[3];
}
```

What is the output of the program is the language is pass-by-value? What if it is pass-by-reference?

Question 2

Recall that *access links* provide a mechanism whereby each function points to "next-closest" scope. In statically-scoped languages that do not allow nested function declarations, access links are not required. Why not?

Question 3

Access links (recalled in Question 2) can be traversed to find variables in "next-closest" scopes. In some types of languages, the number of links that need to be traversed to reach a variable definition is known at compile time. In other types of languages, the number of access links to be traversed is unknown until runtime. What are these two types of languages? Why does the distinction change how access links can be traversed?