EECS665

Compiler Construction

Drew Davidson
Ruturaj Vaidya

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

Lab: Eaton 1005B

ANOUNCEMENTS | LAB | SCHEDULE | MATERIALS | ASSIGNMENTS

Homework 8

Due on November 7th @ 3:00 PM (in class, to Drew, or at Engineering front office)

Not accepted late

ALL homework must be done individually

Question 1

Briefly explain why compilers with static allocation cannot implement languages with recursion.

Question 2

Consider the following Lil' C program:

```
int addAll(int a, int b, int c){          # line 1
  int local;                              # line 2
```

```
  local = a + b + c;                    # line 3
  return local;                         # line 4
}                                       # line 5
int main(){                             # line 6
  int mainLocal = 4;                    # line 7
  addAll(mainLocal, 3, mainLocal);      # line 8
  return 0;                             # line 9
}                                       # line 1
```

At what point in the execution of the above program is the stack at it's deepest? Explain why

Question 3

In class, we have used the two registers `$fp` and `$sp` to track the base of the stack and the end of the stack, respectively. A mysterious cloaked figure claims to have implemented a Lil' C compiler that only needs to use `$fp`, but it comes at an extra cost to memory use. Is this possible? If so, why might there be an extra cost to memory use?

Question 4

Assume that your nemesis has invented a language with the following aspects:

- Users may only define 10 functions
- Dynamic memory is allowed, but is always placed on the heap
- Functions may not call themselves

Your nemesis claims that you will never run out of memory on a function call, and has therefore designed a runtime environment in which programs only have 100 memory slots. Describe at least 2 scenarios in which a function call will cause you to run out of memory. You may write pseudocode if you wish.

Instructor KU EECS