Lab 8: Runtime Stack, Heap, and Program Execution

Due Date: Thursday 4/6/2017 11:59PM

This lab covers material on the the runtime stack and dynamic memory allocation (lectures 18 and 19) . There are 100 points total.

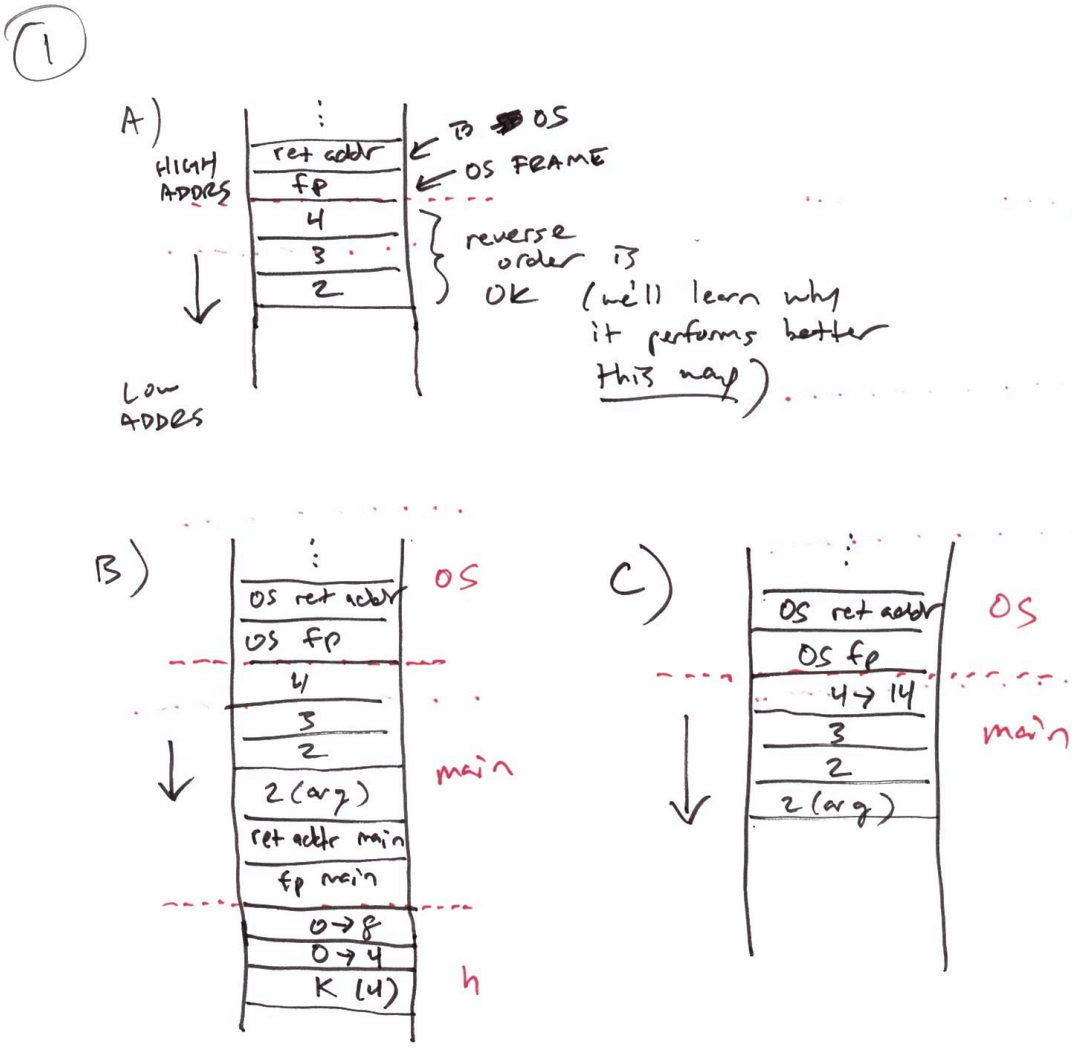
Written Problems (*100 points*)

1. Study the following main program and subroutine `h`. For the program below, show what the runtime stack looks like whenever execution is at locations `(A)`, `(B)`, and `(C)`. (For the arrays, show each element as a separate entry.) You should end up with 3 snapshots showing. Don't forget to include the frame pointers and return addresses!

```
int h (int n);        // Prototype

int main() {                  // return address = (somewhere in OS)
      int b[3] = {2,3,4}     // (A)
      h(b[0])
      b[2] = result of h
      RV = b[2]           //   (C)
      return
  }

  int h (int n) {
    int b[2] = {0,0}
    intk = n * n
    b[0] = k
    b[1] = 2 * k
    RV = n + b[0] + b[1] // (B)
    return
  }
```

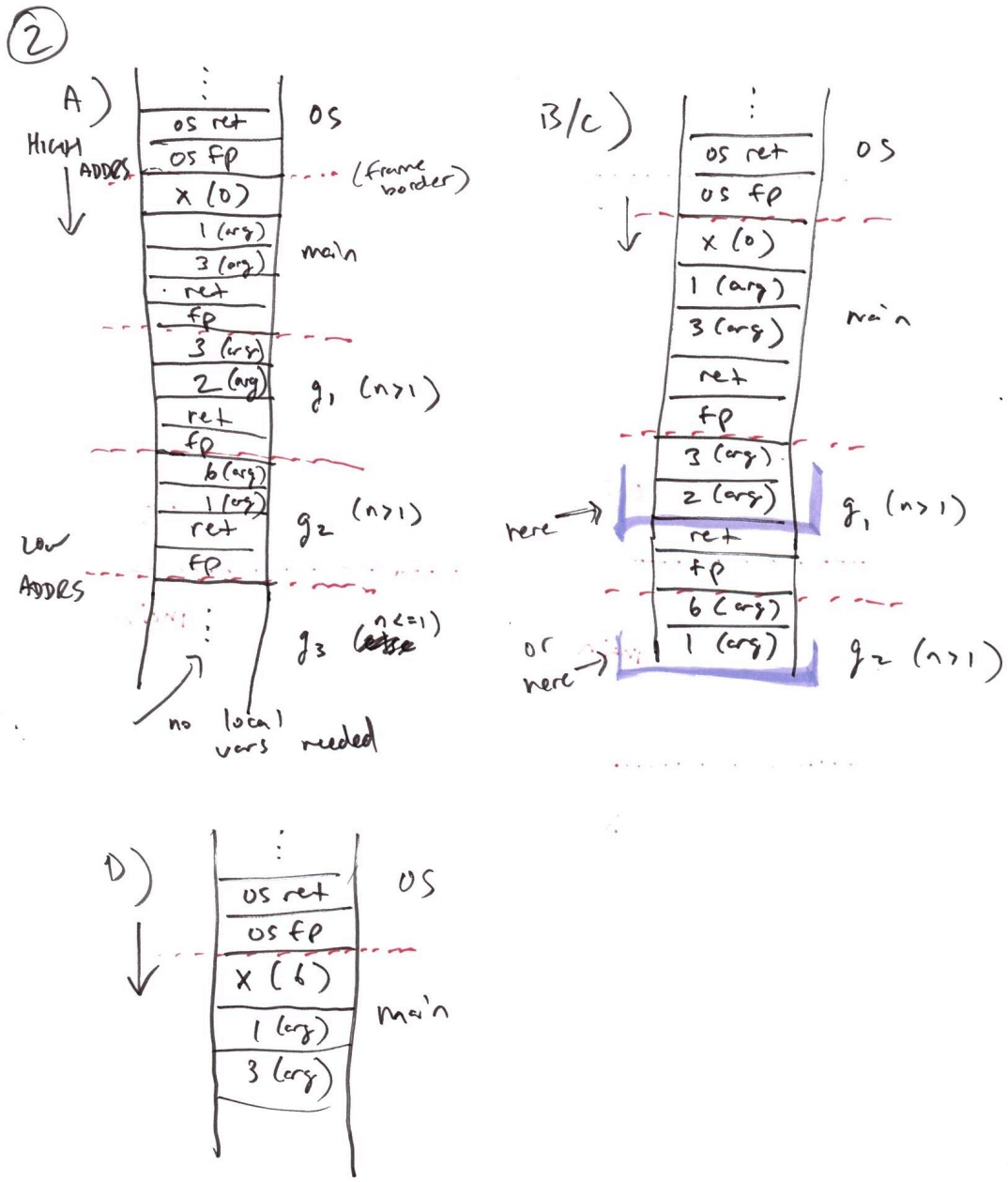


2. Repeat Problem 1 with the following program; show the stack layout at locations `(A)`, `(B)`, `(C)`, and `(D)`. You should end up with 4 snapshots of the stack.

```
int g (int n, int r); // Prototype

int g (int n, int r) {
    if (n <= 1)
        RV = r      // (A) "RV" = "returned value"
    else {
        g(n-1, r*n) // (B)
        RV = result of g // (C)
    }
    return
}

int main () {          // Return address = (somewhere in OS)
    int x = 0
    g(3, 1)
    x = result of g
    RV = 0             // (D)
    return
}
```



3. For the following code, indicate where the lettered variables are stored in memory. Will they be in the stack, the heap, `.bss`, or global data?

```
static int x = 1000; // (A)
float g = 25.3;  // (B)
int g[100] = 0; // (C)

int main () {

    int z;    // (D)
    int k[2] = {5,200]; // (E)
    int *b;             // (F)
```

```
    b = malloc(sizeof(int)*100);

    b[20] = 0xdeadbeef;  // (G)
}
```

A. global data
B. global data
C. bss
D. stack
E. stack
F. stack
G. heap

4. You've been hired as a computer security consultant at a major company. They *suspect* that a foreign state actor has infiltrated their system and is siphoning customer credit card data and sending it over the network to a hacker controlled machine. The company believes that their database process (which is always running) has been modified with malicious code. Recall that in class I talked about using `strace` to see the system calls a process (a running program) invokes. We can also use `strace` to attach to processes that are already running using the `-p` flag and a process ID. How might we use this to detect that the company's database has indeed been hijacked?

   strace can give you an idea of what a process is doing in its "normal" activity. Ideally we'd have a long running trace of system calls from the company's database process, and we could correlate these with time. We might even use a sophisticated machine learning algorithm to derive a model of behavior for the process. When we suspect someone has tampered with the process, we could "validate" the current activity with the model of historical behavior. If it doesn't match, we have some evidence that there might be a break-in. This type of technique is called "anomaly detection." There are, of course, ways for attackers to mitigate this type of detection. Think about how you might do it.

Hand-in Instructions

Make sure to put your name on your submission. Submissions without names will be given zero points! For code, this means put a comment at the top of your code file(s) with your name on it.

Physical : If you're submitting a written copy, hand it to one of the TAs or to the instructor. You can also leave it in the instructor's mailbox in the CS department office, but make sure to get it time stamped when you do (see the "Submitting Work" section of the syllabus).

Digital : If you would like to submit an electronic copy, note that I will only accept PDF files (no Word docs please). Again, see the "Submitting Work" section of the syllabus. Please do not take a poorly lit picture of your assignment. Your grade will suffer commensurately with our inability to read your work. Once you have a PDF, you should submit it on `fourier`. You should name your file `yourid-lab8.pdf` where `yourid` is the thing in front of the @hawk.iit.edu in your e-mail address.

You can first get your PDF (for example, for me it might be called `kh123-lab8.pdf`) onto `fourier` like so:

```
[me@mylocalmachine]$ scp kh123-lab8.pdf kh123@fourier.cs.iit.edu:
```

Then you can login to `fourier` via ssh and submit it:

```
[kh123@fourier]$ cp kh123-lab8.pdf /home/khale/HANDIN/lab8
```

Late handins

If you're turning in your assignment late digitally, you'll need to e-mail me your PDF file directly.