Runtime Stack

CS 350: Computer Organization & Assembler Language Programming
Lab 9, due Wed Mar 30

A. Why?

- Information for a procedure call is stored in an activation frame. At runtime, the activation frames form a stack (in C and C++) or heap (in Java).

B. Outcomes

After this lab, you should be able to describe

- The contents of the runtime stack as routines are called and returned from.

C. Problems [50 points total]

1. [12 points] Study the following main program and subroutine `g`. For the program below, show what the runtime stack looks like whenever execution is at locations `(A)`, `(B)`, or `(C)`. (For the arrays, show each element as a separate entry.) You should end up with 2 snapshots showing a total of 3 activation records. [Grading breakdown: 3 points per record plus 3 for the overall stack structures.]

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

int main() {               // return address = (OS)
    int b[3] = {2,3,4};
    h(b[0]);               (A)
    b[2] = result of h
    RV = b[2];             (C)
    return
}
int h(int n) {
    int b[2] = {0,0};
    int k = n*n;
    b[0] = k;
    b[1] = 2*k;
    RV = n+b[0]+b[1];      (B)
    return
}
```

2. [38 points] Repeat Problem 1 with the following program; show the state at locations `(A)`, `(B)`, `(C)`, or `(E)`. You should end up with 4 snapshots showing a total of 10 activation records. [Grading breakdown: 3 points per record plus 2 points per snapshot.]

```
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 = (OS)
    int x = 0;
    g(3,1);              (D)
    x = result of g
    RV = 0               (E)
    return
}
```