ECE 468 & 573

Problem Set 3: Function calls and Common Subexpression Elimination.

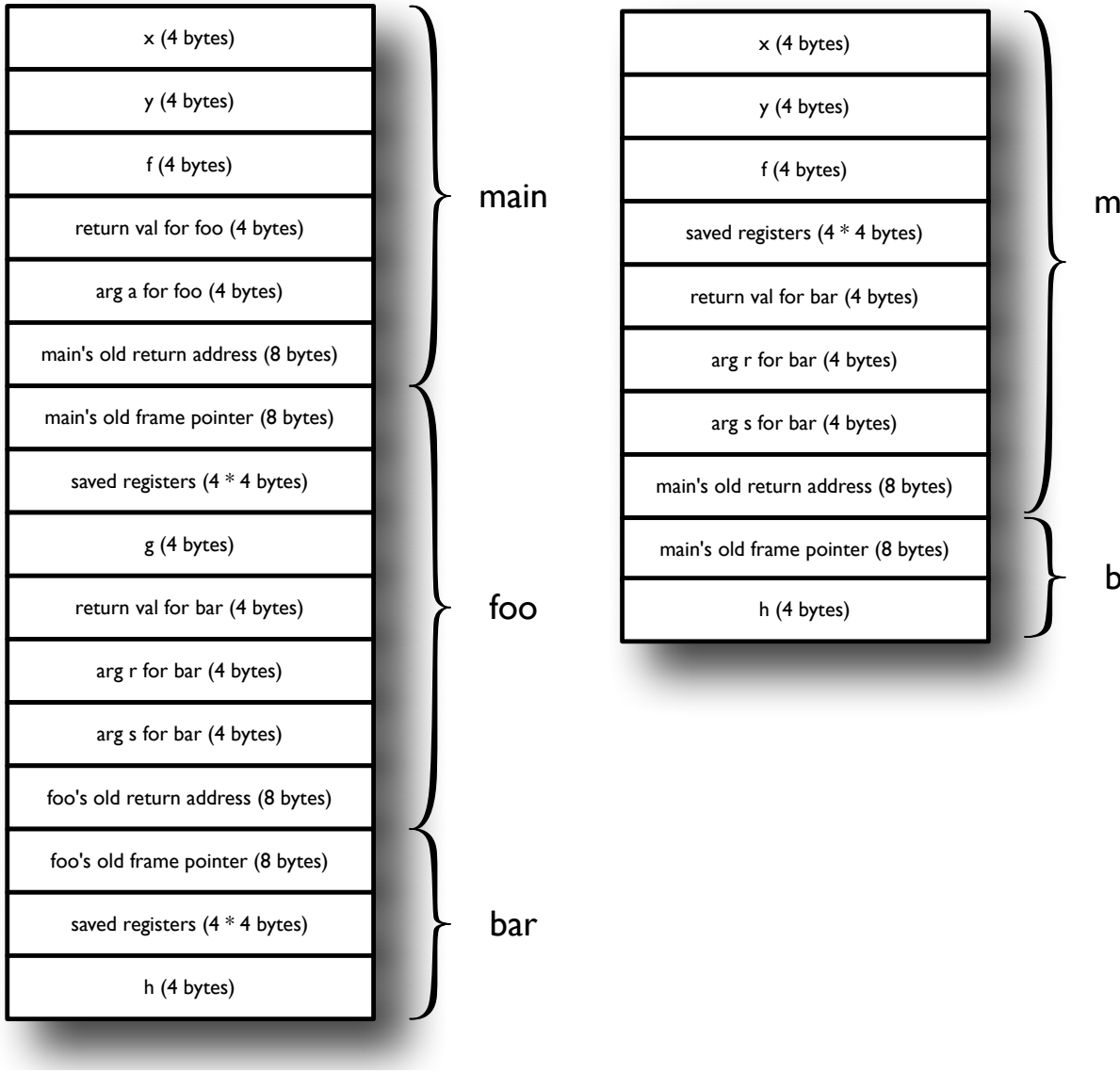
For the following problems, consider the following program:

```
void main() {
   int x;
   int y;
   float f;
   ... //some computations
   f = foo(x + y);
   f += bar(x, y);
   ... //some computations
}

float foo(int a) {
   float g;
   g = bar(a, a);
   return g;
}

float bar(int r, int s) {
   float h;
   h = 1.0 * (r + s);
   return h;
}
```

1. Assume your program is running on a machine with 4 registers, using callee saves. Assume addresses (i.e., pointers) are 8 bytes, floats are 4 bytes and ints are 4 bytes. Draw the *complete stack* (i.e., the stack including all active activation records) for the program *right after foo has called bar, and before bar returns.* For each slot in the stack, indicate what is stored there, and how much space that slot takes up.

2. Now assume that your program uses caller saves instead. Draw the complete stack for the program *right after main calls bar, and before bar returns.*

Answers:



For the following problems, consider the following piece of three-address code:

```
1.  READ(A)
2.  READ(B)
3.  C = A + B
4.  A = A + B
5.  B = C * D
6.  T1 = A + C
7.  T2 = T1 + C
```

```
8.  F = A + C
9.  C = F + B
10. G = A + C
11. T3 = F + B
12. WRITE(T3)
```

1. Show the result of performing Common Subexpression Elimination (CSE) on the above code. Rather than generating assembly, just give new 3AC. If you're reusing the results of an expression, just generate code that looks like X = Y, where Y is the variable/temporary that held the result of the original calculation.

   ```
   1.  READ(A)
   2.  READ(B)
   3.  C = A + B
   4.  A = C //A + B
   5.  B = C * D
   6.  T1 = A + C
   7.  T2 = T1 + C
   8.  F = T1 //A + C
   9.  C = F + B
   10. G = A + C
   11. T3 = C //F + B
   12. WRITE(T3)
   ```

2. Suppose F and A were aliased. How would that change the results of CSE?

   No change. Although if there were other aliasing, things could change more.