**Daniyal Khan  
  
3765942  
  
CS-2263  
  
Lab 2**

**Exercise 1:**

#*include* <stdio.h>

#*include* <stdlib.h>

int *g1*(int a, int b)

{

int c = (a + b) \* b;

*putchar*('\n');

*printf*("Address of a in g1: %p\n", &a);

*printf*("Address of b in g1: %p\n", &b);

*printf*("Address of c in g1: %p\n", &c);

*printf*("g1: %d %d %d\n", a, b, c);

*return* c;

}

int *g2*(int a, int b)

{

int c = *g1*(a + 3, b - 11);

*putchar*('\n');

*printf*("Address of a in g2: %p\n", &a);

*printf*("Address of b in g2: %p\n", &b);

*printf*("Address of c in g2: %p\n", &c);

*printf*("g2: %d %d %d\n", a, b, c);

*return* c - b;

}

int *main*(int argc, char \*\*argv)

{

int a = 5;

int b = 17;

int c = *g2*(a - 1, b \* 2);

*putchar*('\n');

*printf*("Address of a in main: %p\n", &a);

*printf*("Address of b in main: %p\n", &b);

*printf*("Address of c in main: %p\n", &c);

*printf*("main: %d %d %d\n", a, b, c);

*return* *EXIT\_SUCCESS*;

}

Output of the program:

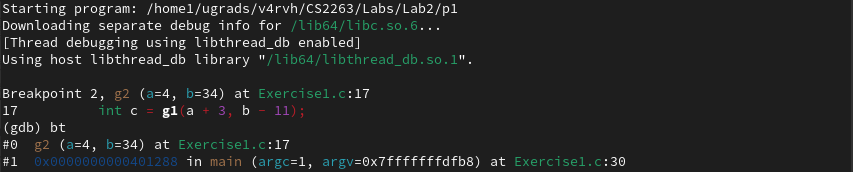


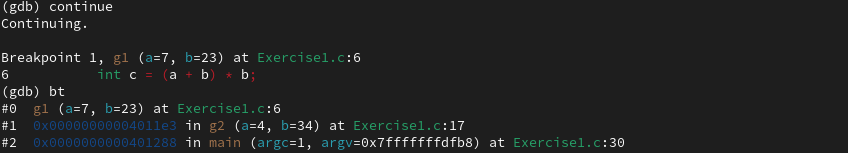
The values of the variables printed from my program is the same as my colleague because we are using the same values for a, b and c. And they don’t change.

The addresses of the variables printed from my program are different than my colleagues because every machine/computer stores them at a different place in memory.

The address printed for the variables in the function g1 is smaller than g2 because g1 is at the top of stack. In C language the memory address is allocated in descending order.

**Exercise 2:**

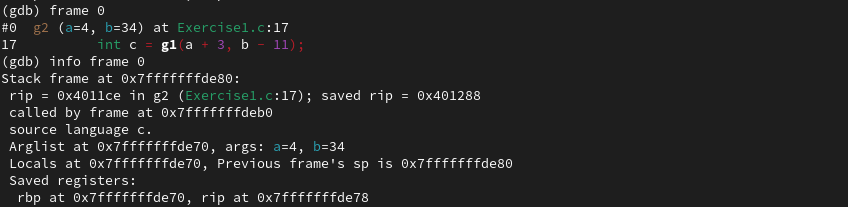
Backtrace after reaching the breakpoint g2:   


There are 2 frames shown in the trace.   
  
  
Backtrace after reaching the breakpoint g1:  


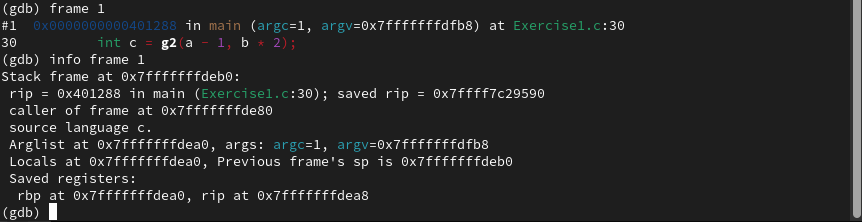
There are 3 frames shown in the trace.

At breakpoint g2:

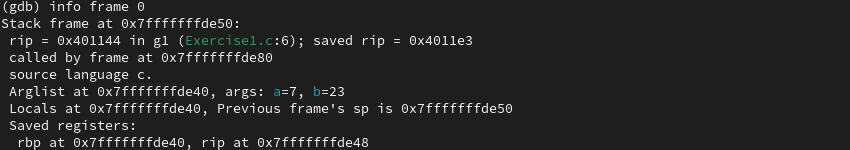
Frame 0:

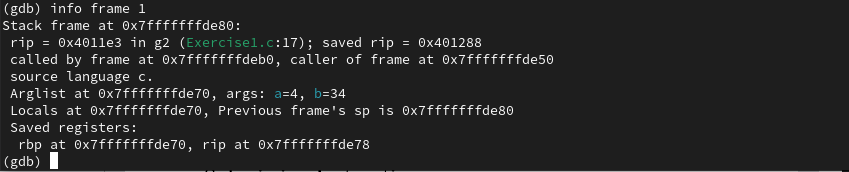


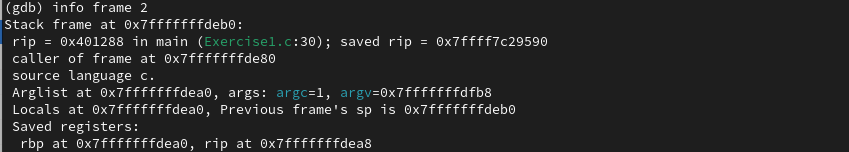
Frame 1:

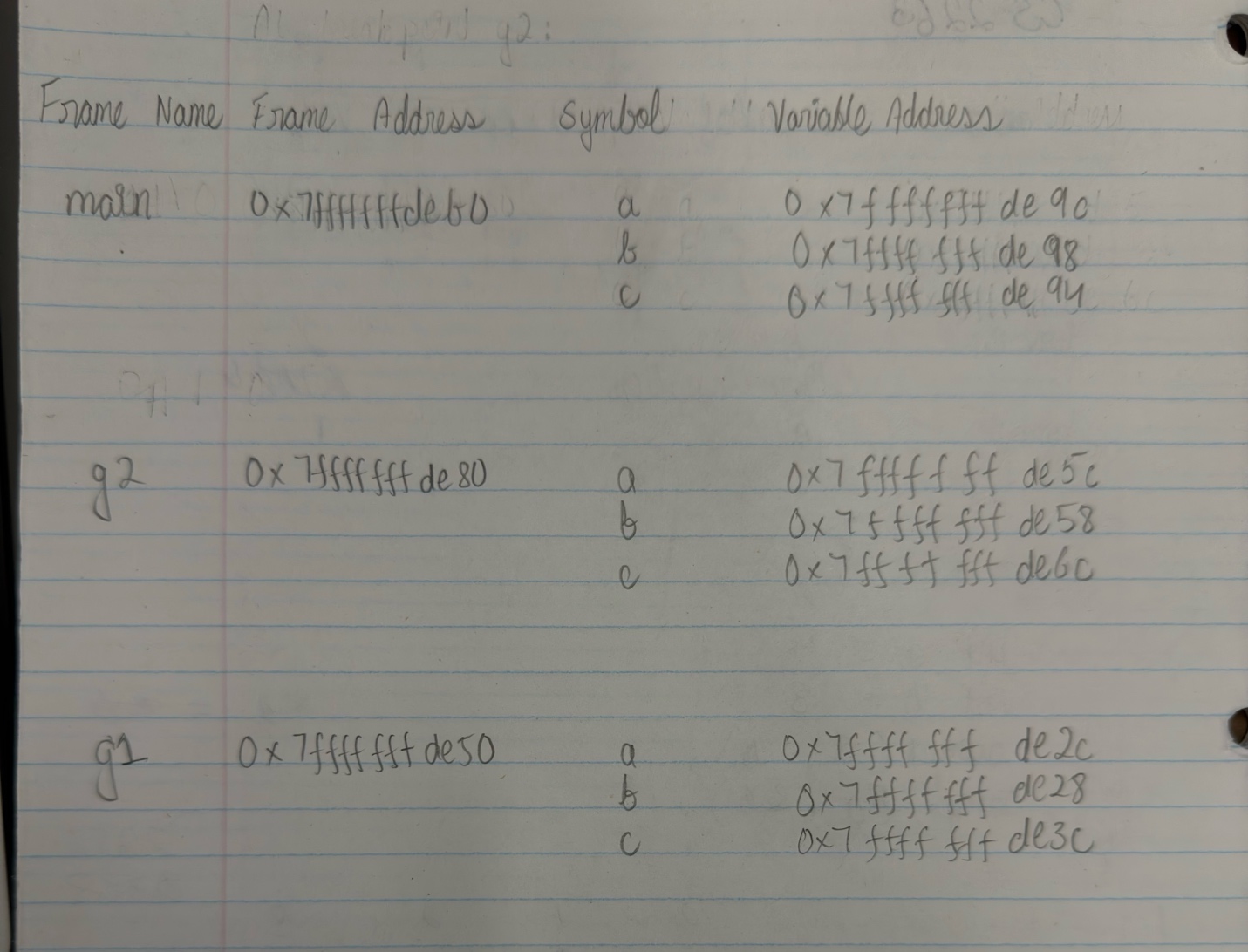


At breakpoint g1:

Frame 0:  


Frame 1:  


Frame 2:  




NOTE: My memory map has main on the top because as I was going through the program I just wrote main first because it gets called first but I do understand the fact that main would be at the bottom of the stack.

Stack addresses are related to the addresses of the variables as we can see that for the g2 stack the address is 0x7fffffffde80 and the variables lie between that and the stack address of the next stack which is g1 is this case 0x7fffffffde50.  
The address of a b and c lie between g2’s address and g1’s address.

**Exercise 3:**

#*include* <stdio.h>

#*include* <stdlib.h>

void *calcFib*(int n);

int *main*() {

*calcFib*(10);

*return* *EXIT\_SUCCESS*;

}

void *calcFib*(int n) {

int a = 0, b = 1, temp;

*if* (n <= 0) {

*return*;

}

*printf*("%d ", a);

*for* (int i = 1; i < n; i++) {

*printf*("%d ", b);

temp = a + b;

a = b;

b = temp;

}

}

Debugger screenshot showing the backtrace after reaching breakpoint at calcFib:

