CS2263

Lab 2

February 1st, 2024

UNB Fredericton

Class Code: Cs2263

Document: Lab 2

Student Name: Will Ross #3734692

Due Date: February 1st, 2024

Exercise 1

Source Code:

```
/* p1.c */
#include <stdio.h>
#include <stdlib.h>
int g1(int a, int b)
{
   int c = (a + b) * b;
   printf("g1:\na = %d\nb = %d\nc = %d\n", a, b, c);
   printf("g1:\naddress of a = %p\naddress of b = %p\naddress of c = %p\n", &a, &b, &c);
   return c;
}

int g2(int a, int b)
{
   int c = g1(a + 3, b - 11);
   printf("g2:\na = %d\nb = %d\nc = %d\n", a, b, c);
   printf("g2:\naddress of a = %p\naddress of b = %p\naddress of c = %p\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);
   printf("main:\na = %d\nb = %d\nc = %d\n", a, b, c);
   printf("main:\naddress of a = %p\naddress of b = %p\naddress of c = %p\n", &a,&b,&c);
   return EXIT_SUCCESS;
```

}

Output:

```
main: address of a = 0x7ffd0659d33c, address of b = 0x7ffd0659d338, address o
f c = 0x7ffd0659d334
[q3d5k@gc112m37 Lab2]$ cd "/home1/ugrads/q3d5k/Cs2263/Labs/Lab2/" && gcc p1.c
 -o p1 && "/home1/ugrads/q3d5k/Cs2263/Labs/Lab2/"p1
g1:
a = 7
b = 23
c = 690
g1:
address of a = 0x7fff05ada89c
address of b = 0x7fff05ada898
address of c = 0x7fff05ada8ac
g2:
a = 4
b = 34
c = 690
g2:
address of a = 0x7fff05ada8cc
address of b = 0x7fff05ada8c8
address of c = 0x7fff05ada8dc
main:
a = 5
b = 17
c = 656
main:
address of a = 0x7fff05ada90c
address of b = 0x7fff05ada908
address of c = 0x7fff05ada904
[q3d5k@gc112m37 Lab2]$
```

Questions:

Are the values of the variables printed from your program the same as obtained by your colleagues? Why?

Yes, the values are the same due to the values being gathered by our program and not hardware dependent.

- Are the addresses printed from your program the same as obtained by your colleagues? Why?

They are different from my colleagues due to c being a hardware dependent programming language and will set different variables to different memory locations.

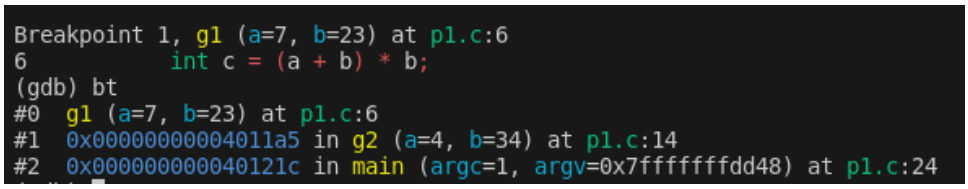
- Are the addresses printed for the variables used in the function g1 higher or lower than the addresses printed from the function g2? Why?

In G1 the addresses printed are lower in G1 compared to G2 due to the program when assigning variables to memory addresses, G1 has smaller memory address than G2 because G1 will be higher in the program.

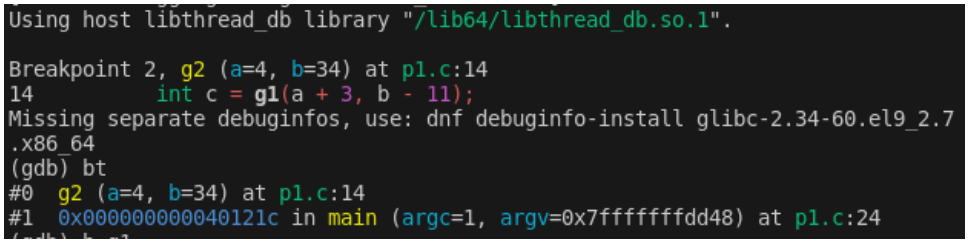
Exercise 2

Output 2.1:

G1 bt



G2 bt



Output 2.1:

Info frame 0,1,2

```
(gdb) info frame 0
Stack frame at 0x7fffffffdbe0:
 rip = 0x401134 in g1 (p1.c:6); saved rip = 0x4011a5
 called by frame at 0x7fffffffdc10
 source language c.
 Arglist at 0x7fffffffdbd0, args: a=7, b=23
 Locals at 0x7fffffffdbd0, Previous frame's sp is 0x7fffffffdbe0
 Saved registers:
  rbp at 0x7fffffffdbd0, rip at 0x7fffffffdbd8
(gdb) info frame 1
Stack frame at 0x7fffffffdc10:
 rip = 0x4011a5 in g2 (p1.c:14); saved rip = 0x40121c
 called by frame at 0x7fffffffdc40, caller of frame at 0x7fffffffdbe0
 source language c.
 Arglist at 0x7fffffffdc00, args: a=4, b=34
 Locals at 0x7fffffffdc00, Previous frame's sp is 0x7fffffffdc10
 Saved registers:
  rbp at 0x7fffffffdc00, rip at 0x7fffffffdc08
(gdb) info frame 2
Stack frame at 0x7fffffffdc40:
 rip = 0x40121c in main (p1.c:24); saved rip = 0x7ffff7c3feb0
 caller of frame at 0x7fffffffdc10
 source language c.
 Arglist at 0x7fffffffdc30, args: argc=1, argv=0x7fffffffdd48
 Locals at 0x7fffffffdc30, Previous frame's sp is 0x7fffffffdc40
 Saved registers:
  rbp at 0x7fffffffdc30, rip at 0x7fffffffdc38
```

Info frame

```
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./p1...
(gdb) b g1
Breakpoint 1 at 0x401134: file p1.c, line 6.
(gdb) b g2
Breakpoint 2 at 0x401190: file p1.c, line 14.
(gdb) cont
The program is not being run.
(gdb) run
Starting program: /home1/ugrads/q3d5k/Cs2263/Labs/Lab2/p1
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".

Breakpoint 2, g2 (a=4, b=34) at p1.c:14
14          int c = g1(a + 3, b - 11);
Missing separate debuginfos, use: dnf debuginfo-install glibc-2.34-60.el9_2.7
.x86_64
(gdb) info frame
Stack level 0, frame at 0x7fffffffdc10:
 rip = 0x401190 in g2 (p1.c:14); saved rip = 0x40121c
 called by frame at 0x7fffffffdc40
 source language c.
 Arglist at 0x7fffffffdc00, args: a=4, b=34
 Locals at 0x7fffffffdc00, Previous frame's sp is 0x7fffffffdc10
 Saved registers:
  rbp at 0x7fffffffdc00, rip at 0x7fffffffdc08
(gdb)
```

Are the stack addresses of each frame related to the addresses of the variables a, b and c printed from functions main, g1 and g2?

No, they are not the same as the functions from g1 and g2

Show a memory map indicating the boundaries of each frame and the storage locations used by each variable

Memory Map

Frame	Symbol	Address	Value
g1	a	0x7fffffffdbbc	7
g1	b	0x7fffffffdbb8	23
g1	c	0x7fffffffdbcc	690
g1	return	0x7fffffffdbd8	0x4011a5
g2	a	0x7fffffffdbbc	4
g2	b	0x7fffffffdbe8	34
g2	c	0x7fffffffdbfc	690
g2	return	0x7fffffffdc08	0x40121c
main	a	0x7fffffffdc2c	5
main	b	0x7fffffffdc28	17
main	c	0x7fffffffdc24	656
main	return	0x7fffffffdc38	0x7ffff7c3feb0

Info frame G1

```
g1:
a = 7
b = 23
c = 690
g1:
address of a = 0x7fffffffdbbc
address of b = 0x7fffffffdbb8
address of c = 0x7fffffffdbcc

Breakpoint 2, g2 (a=4, b=34) at p1.c:15
15          printf("g2:\na = %d\nb = %d\nc = %d\n", a, b, c);
(gdb) cont
```

Info frame G2

```
(gdb) cont
Continuing.
g2:
a = 4
b = 34
c = 690
g2:
address of a = 0x7fffffffdbec
address of b = 0x7fffffffdbe8
address of c = 0x7fffffffdbfc

Breakpoint 3, main (argc=1, argv=0x7fffffffdd48) at p1.c:25
25          printf("main:\na = %d\nb = %d\nc = %d\n", a, b, c);
```

Info frame main

```
Breakpoint 3, main (argc=1, argv=0x7fffffffdd48) at p1.c:25
25          printf("main:\na = %d\nb = %d\nc = %d\n", a, b, c);
(gdb) cont
Continuing.
main:
a = 5
b = 17
c = 656
main:
address of a = 0x7fffffffdc2c
address of b = 0x7fffffffdc28
address of c = 0x7fffffffdc24
[Inferior 1 (process 450299) exited normally]
```

Exercise 3

Source Code:

```
#include <stdio.h>


int calcTrib(int n) {
   printf("%d\n", n);
   return n;
}



int main(void){
   int numOfValues = 20, num1 = 0, num2 = 0, num3 = 1, num4 = 0;
   for(int i = 0; i < numOfValues; i++){
```

```
        if(i % 2 == 0){
            //printf("%d\n", num1);
            calcTrib(num1);
        }
        num4 = num1 + num2 + num3;
        num1 = num2;
        num2 = num3;
        num3 = num4;
    }
}
```

Output

Test results

```
cd "/home1/ugrads/q3d5k/Cs2263/Labs/Lab2/" && gcc [q3d5k@gc112m37 Lab2]$ cd "
/home1/ugrads/q3d5k/Cs2263/Labs/Lab2/" && gcc tempCodeRunnerFile.c -o tempCod
eRunnerFile && "/home1/ugrads/q3d5k/Cs2263/Labs/Lab2/"tempCodeRunnerFile
0
1
2
7
24
81
274
927
3136
10609
[q3d5k@gc112m37 Lab2]$
```

```
PS C:\Users\willr\Documents\GitHub\Cs2263\Labs\Lab2> cd "c:\Users\willr\Documents\GitHub\Cs2263\Labs\Lab2\" ; if ($?) { gcc calcTrib.c -o calc
Trib } ; if ($?) { .\calcTrib }
0 address: 00000066bd9ff7c0
1 address: 00000066bd9ff7c0
2 address: 00000066bd9ff7c0
7 address: 00000066bd9ff7c0
24 address: 00000066bd9ff7c0
81 address: 00000066bd9ff7c0
274 address: 00000066bd9ff7c0
927 address: 00000066bd9ff7c0
3136 address: 00000066bd9ff7c0
10609 address: 00000066bd9ff7c0
PS C:\Users\willr\Documents\GitHub\Cs2263\Labs\Lab2>
```

Note I complied this later to show that it is the same memory address ^

breakpoints

```
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./calcTrib...
(gdb) b 16
Breakpoint 1 at 0x401192: file calcTrib.c, line 17.
(gdb) run
Starting program: /home1/ugrads/q3d5k/Cs2263/Labs/Lab2/calcTrib
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
0

Breakpoint 1, main () at calcTrib.c:17
17              num4 = num1 + num2 + num3;
Missing separate debuginfos, use: dnf debuginfo-install glibc-2.34-60.el9_2.7
.x86_64
(gdb) bt
#0  main () at calcTrib.c:17
(gdb) info frame 0
Stack frame at 0x7fffffffdc30:
 rip = 0x401192 in main (calcTrib.c:17); saved rip = 0x7ffff7c3feb0
 source language c.
 Arglist at 0x7fffffffdc20, args:
 Locals at 0x7fffffffdc20, Previous frame's sp is 0x7fffffffdc30
 Saved registers:
  rbp at 0x7fffffffdc20, rip at 0x7fffffffdc28
```