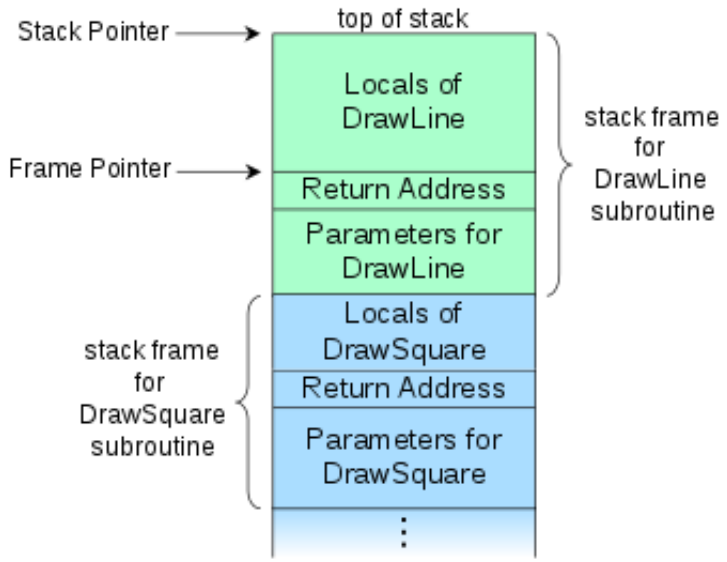
Lab1: Stack and Stack Frame (5 Points)



Objectives and Targets

The **stack** is a segment of memory where data like your local variables and function calls get added and/or removed in a last-in-first-out (LIFO) manner. When you compile a program, the compiler enters through the main function and a **stack frame** is created on the stack. A frame, also known as an activation record is the collection of all data on the stack associated with one subprogram call. The main function and all the local variables are stored in an initial frame.

In this lab, you'll use gdb to debug a program and answer questions related to stack and stack frame

Step 1: First, you need to connect to our badger CTF system, once connected, you need to go to folder "lab1" by typing

```
cd lab1
```

```c
#include <stdio.h>
int add_plus1(int a, int b)
{
    int x = a;
    int y = b;
    return x+y+1;
}

int main(int argc, char** argv) {
    int a=5, b=6;
    int d = add_plus1(a,b);
    printf("%d\n", d);
    return 0;
}
```

source code for lab1

Step 2: Use `gdb` to analysis the binary program of lab1.

```
gdb lab1
```

P.S. If you're not familar with `gdb`, now it's the best time to check this tutorial Link.

Step 3: Disassemble the `main` function by typing the following command and answer the following question(s):

```
disas main
```

```
Dump of assembler code for function main:
   0x0804919f <+0>:     push   ebp
   0x080491a0 <+1>:     mov    ebp,esp
   0x080491a2 <+3>:     push   ebx
   0x080491a3 <+4>:     sub    esp,0xc
   0x080491a6 <+7>:     call   0x80490b0 <__x86.get_pc_thunk.bx>
   0x080491ab <+12>:    add    ebx,0x2e55
   0x080491b1 <+18>:    mov    DWORD PTR [ebp-0x10],0x5
   0x080491b8 <+25>:    mov    DWORD PTR [ebp-0xc],0x6
   0x080491bf <+32>:    push   DWORD PTR [ebp-0xc]
   0x080491c2 <+35>:    push   DWORD PTR [ebp-0x10]
   0x080491c5 <+38>:    call   0x8049176 <add_plus1>
   0x080491ca <+43>:    add    esp,0x8
   0x080491cd <+46>:    mov    DWORD PTR [ebp-0x8],eax
   0x080491d0 <+49>:    push   DWORD PTR [ebp-0x8]
   0x080491d3 <+52>:    lea    eax,[ebx-0x1ff8]
   0x080491d9 <+58>:    push   eax
   0x080491da <+59>:    call   0x8049040 <printf@plt>
   0x080491df <+64>:    add    esp,0x8
   0x080491e2 <+67>:    mov    eax,0x0
   0x080491e7 <+72>:    mov    ebx,DWORD PTR [ebp-0x4]
   0x080491ea <+75>:    leave
   0x080491eb <+76>:    ret
End of assembler dump.
```

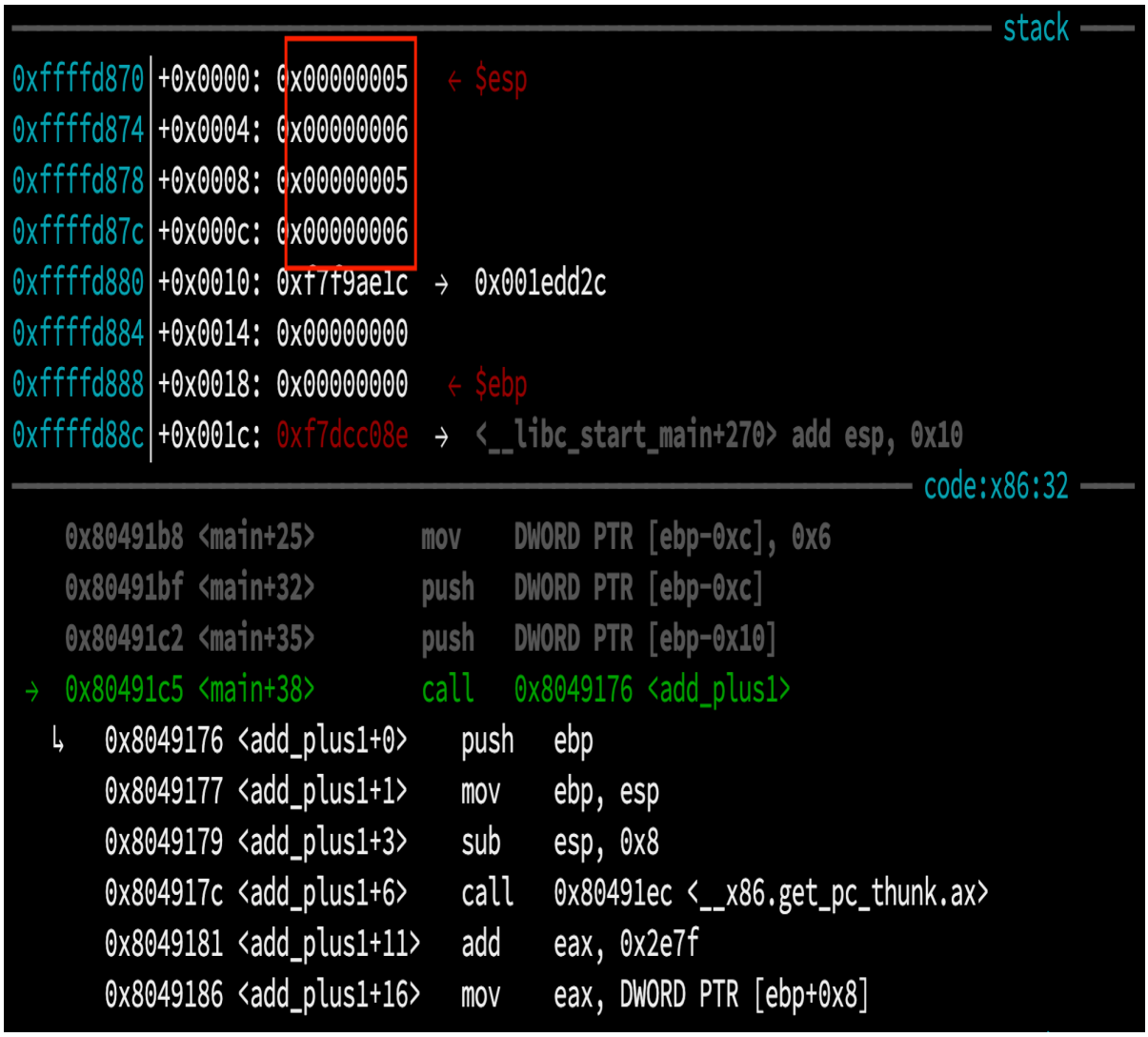
Assembly code for main function

Q1: Pointing out the assembler code (including their memory address) which are used for creating the stack frame of the main() function (1 point):

Q2: What's the meaning of these two lines (1 point):

```
0x080491b1 <+18>:    mov    DWORD PTR [ebp-0x10],0x5
0x080491b8 <+25>:    mov    DWORD PTR [ebp-0xc],0x6
```

Q3: Before calling add_plus1() function, the stack contains 5,6,5,6 (see the picture below), why there are two sets of "5,6" instead of just one?(1 point):



Q3

Step 4: Disassemble the `add_plus1` function by typing the following command and answer the following question(s):

```
disas add_plus1
```

```
Dump of assembler code for function add_plus1:
   0x08049176 <+0>:     push   ebp
   0x08049177 <+1>:     mov    ebp,esp
   0x08049179 <+3>:     sub    esp,0x8
   0x0804917c <+6>:     call   0x80491ec <__x86.get_pc_thunk.ax>
   0x08049181 <+11>:    add    eax,0x2e7f
   0x08049186 <+16>:    mov    eax,DWORD PTR [ebp+0x8]
   0x08049189 <+19>:    mov    DWORD PTR [ebp-0x8],eax
   0x0804918c <+22>:    mov    eax,DWORD PTR [ebp+0xc]
   0x0804918f <+25>:    mov    DWORD PTR [ebp-0x4],eax
   0x08049192 <+28>:    mov    edx,DWORD PTR [ebp-0x8]
   0x08049195 <+31>:    mov    eax,DWORD PTR [ebp-0x4]
   0x08049198 <+34>:    add    eax,edx
   0x0804919a <+36>:    add    eax,0x1
   0x0804919d <+39>:    leave
   0x0804919e <+40>:    ret
End of assembler dump.
```

Assembly code for add_plus1 function

Q4: What's the meaning of the first two lines (1 point):

```
0x08049176 <+0>:     push   ebp
0x08049177 <+1>:     mov    ebp,esp
```

Q5: Which register is being used to store the final summation result? (1 point):

Deliverables:

- A detailed project report in **PDF format** to answer the above questions, you can draw some pictures/diagrams and include code snippets if needed.

Submission

- Check lab due date on the course website. Late submission will not be accepted.

- The assignment should be submitted to D2L directly.
- No copy or cheating is tolerated. If your work is based on others', please give clear attribution. Otherwise, you **WILL FAIL** this course.