

CS240
Laboratory 9 Assignment
X86 Stack

Inspect the following C program:

```
#include <stdio.h>

long getAndSumValues() {
    long x;
    printf("Enter an integer: ");
    scanf(" %ld",&x);
    if (x == 0) {
        return 0;
    } else {
        return x + getAndSumValues();
    }
}

int main() {
    long result;
    result = getAndSumValues();
    printf("The result = %ld\n",result);
    return 0;
}
```

It produces the following x86 instructions when compiled:

Dump of assembler code for function main:

```
//setup
    0x0000000000400577 <+0>:      sub   $0x8,%rsp
//call getAndSumValues
    0x000000000040057b <+4>:      mov   $0x0,%eax
    0x0000000000400580 <+9>:      callq 0x400534 <getAndSumValues>

//print the result
    0x0000000000400585 <+14>:     mov   %rax,%rsi
    0x0000000000400588 <+17>:     mov   $0x4006bd,%edi
    0x000000000040058d <+22>:     mov   $0x0,%eax
    0x0000000000400592 <+27>:     callq 0x400418 <printf@plt>
    0x0000000000400597 <+32>:     mov   $0x0,%eax
    0x000000000040059c <+37>:     add   $0x8,%rsp
    0x00000000004005a0 <+41>:     retq
End of assembler dump.
```

Dump of assembler code for function getAndSumValues

//setup

```
0x000000000400534 <+0>:    sub   $0x18,%rsp
```

// prompt user to enter a value

```
0x000000000400538 <+4>:    mov   $0x4006a8,%edi
0x00000000040053d <+9>:    mov   $0x0,%eax
0x000000000400542 <+14>:   callq 0x400418 <printf@plt>
```

//accept the input from the user

```
0x000000000400547 <+19>:   lea  0x8(%rsp),%rsi
0x00000000040054c <+24>:   mov   $0x4006b8,%edi
0x000000000400551 <+29>:   mov   $0x0,%eax
0x000000000400556 <+34>:   callq 0x400438 <__isoc99_scanf@plt>
```

//explain this section

```
0x00000000040055b <+39>:   mov   $0x0,%eax
0x000000000400560 <+44>:   cmpq  $0x0,0x8(%rsp)
0x000000000400566 <+50>:   je    0x400572 <getAndSumValues+62>
0x000000000400568 <+52>:   callq 0x400534 <getAndSumValues>
0x00000000040056d <+57>:   add  0x8(%rsp),%rax
```

//finish

```
0x000000000400572 <+62>:   add  $0x18,%rsp
0x000000000400576 <+66>:   retq
```

1. What type of program is this? Hint: notice that there is a call to *getAndSumValues* within *getAndSumValues*.
2. How many parameters does *getAndSumValues()* have?
3. Where do the values input by the user get stored in memory?
4. Explain in some detail the highlighted code.
5. What register is used to accumulate the sum?