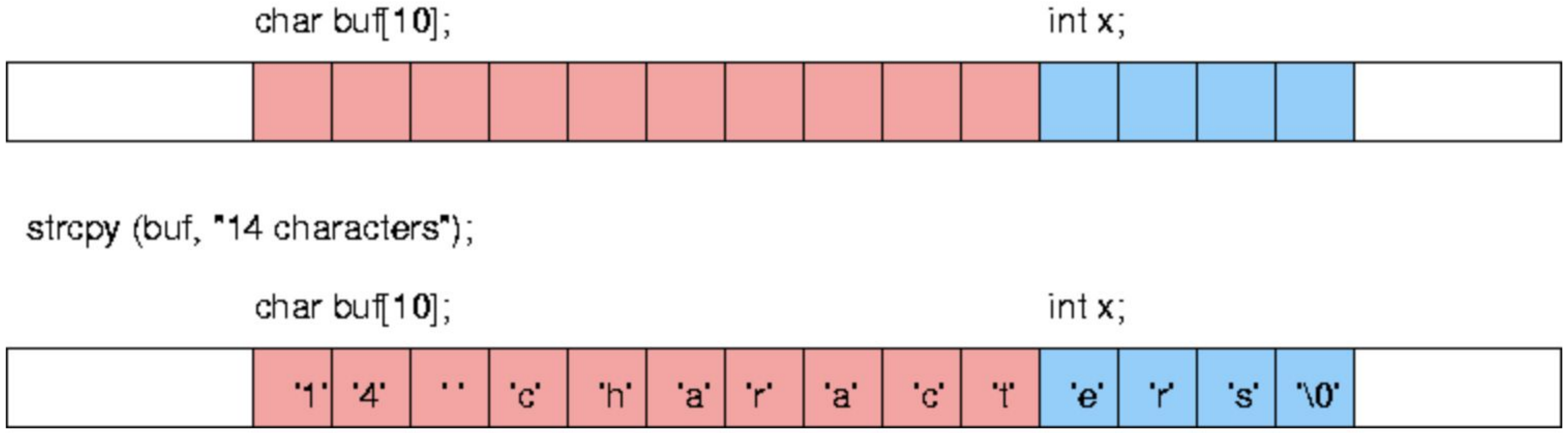
CS 33: Introduction to Computer Organization

TA: Aalisha Dalal
LA: Jonathan Myong
Office Hours: Friday, 9:30-11:30AM

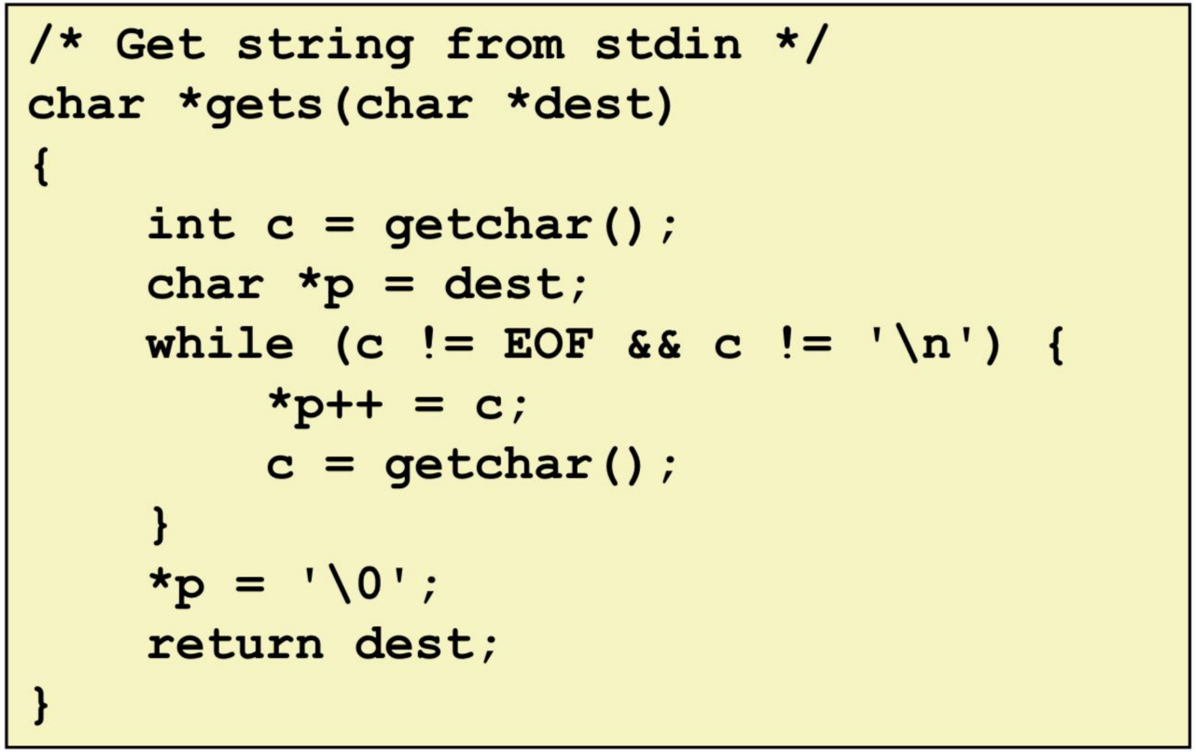
Outline

- Buffer Overflow/ Attacks
- Practice Problems for Midterm
- Worksheet problem

Buffer Overflow - Example



Buffer Overflow



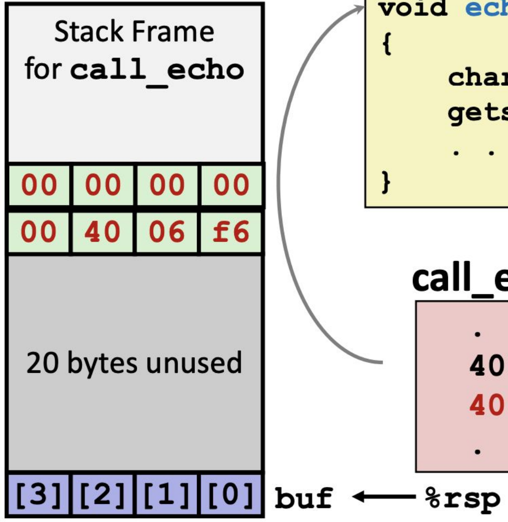
```
/* Get string from stdin */
char *gets(char *dest)
{
    int c = getchar();
    char *p = dest;
    while (c != EOF && c != '\n') {
        *p++ = c;
        c = getchar();
    }
    *p = '\0';
    return dest;
}
```

Is there any issue that you see with this function? How can it be exploited?

Security Breaches - Buffer Overflow (I)

Buffer Overflow Stack Example

Before call to gets



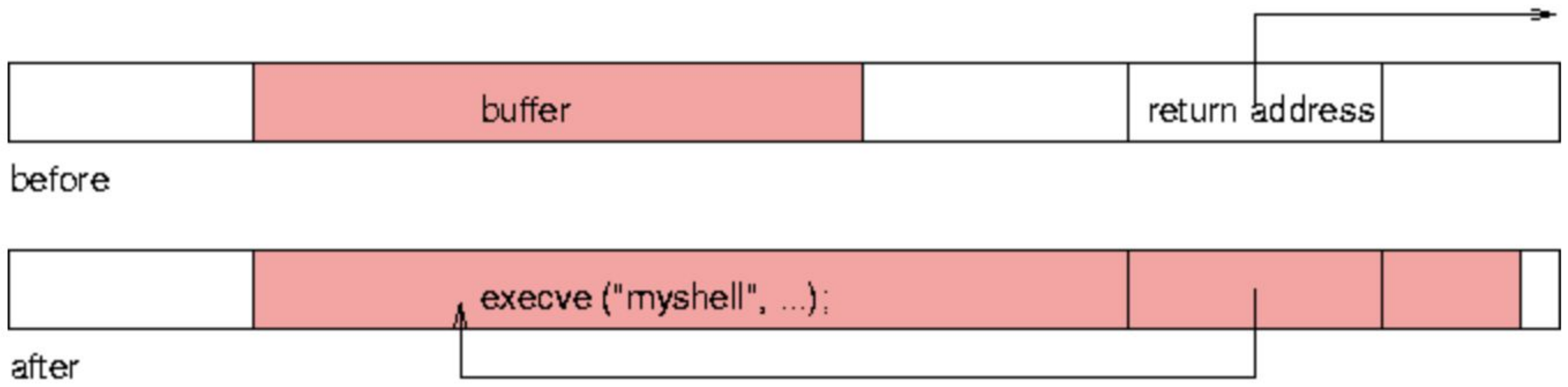
```
void echo()
{
    char buf[4];
    gets(buf);
    . . .
}
```

```
echo:
  subq  $24, %rsp
  movq  %rsp, %rdi
  call  gets
  . . .
```

call_echo:

```
  . . .
  4006f1:  callq  4006cf <echo>
  4006f6:  add    $0x8,%rsp
  . . .
```

Security Breaches - Buffer Overflow (II)



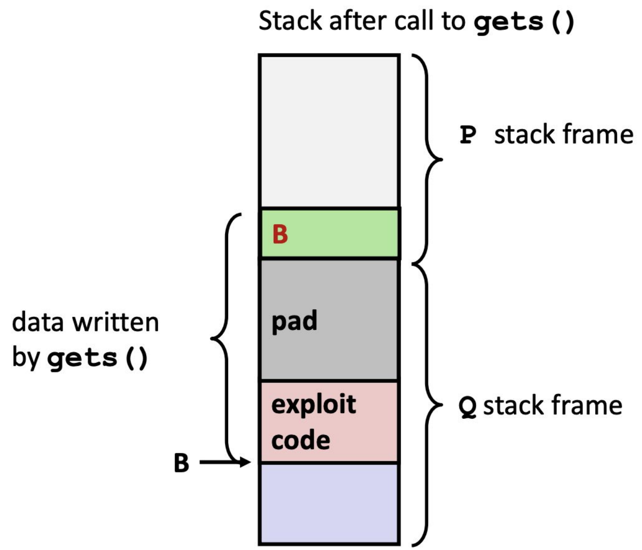
Security Breaches - Buffer Overflow (II)

Code Injection Attacks

```
void P(){
  Q();
  ...
}
```

return
address
A

```
int Q() {
  char buf[64];
  gets(buf);
  ...
  return ...;
}
```



- Input string contains byte representation of executable code
- Overwrite return address A with address of buffer B
- When Q executes `ret`, will jump to exploit code

How can we prevent such attacks?

- Library routines that ***limit string lengths*** - fgets, strncpy functions
- ***Randomized offset*** for function stack allocation
- Mark the stack as ***non-executable code segment***
- ***Stack canaries*** - placed after the buffer

Practice Problems - 0

```
int array[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
```

Suppose that the compiler has placed the variable `array` in the `%ecx` register. How do you move the value at `array[3]` into the `%eax` register? Assume that `%ebx` is 3.

(a) `leal 12(%ecx),%eax`

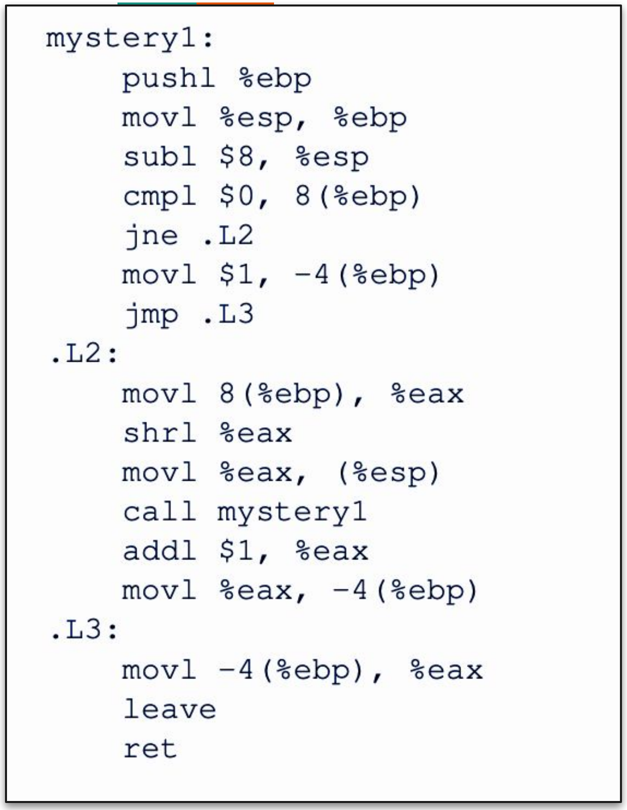
(b) `leal (%ecx,%ebx,4),%eax`

[✓ (c)] `movl (%ecx,%ebx,4),%eax`

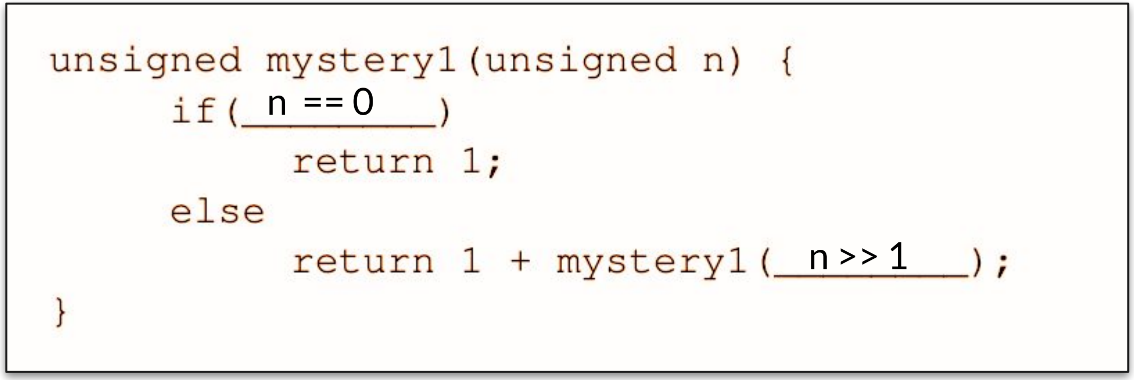
(d) `movl 8(%ecx,%ebx,2),%eax`

(e) `leal 4(%ecx,%ebx,1),%eax`

Practice Problems - 1

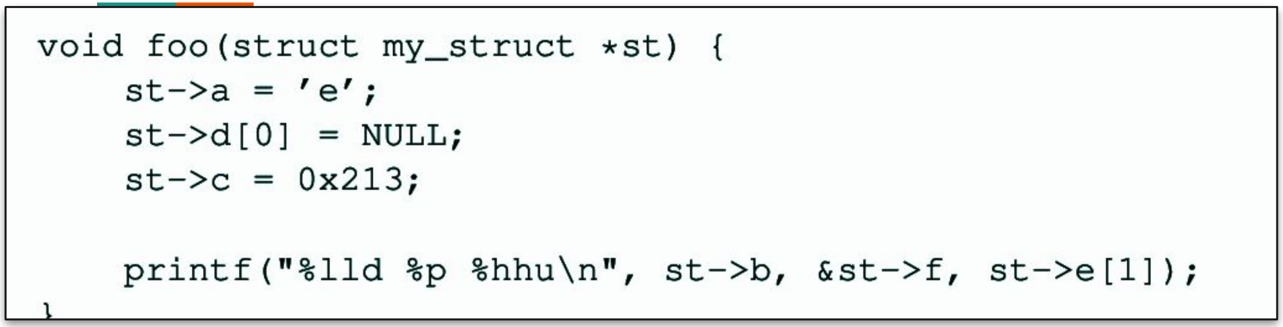


```
mystery1:
    pushl %ebp
    movl %esp, %ebp
    subl $8, %esp
    cmpl $0, 8(%ebp)
    jne .L2
    movl $1, -4(%ebp)
    jmp .L3
.L2:
    movl 8(%ebp), %eax
    shrl %eax
    movl %eax, (%esp)
    call mystery1
    addl $1, %eax
    movl %eax, -4(%ebp)
.L3:
    movl -4(%ebp), %eax
    leave
    ret
```



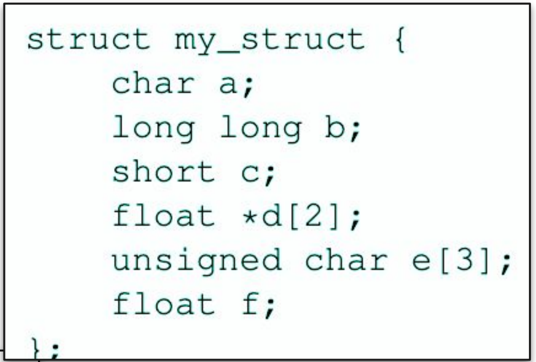
```
unsigned mystery1(unsigned n) {
      if(_n == 0______)
            return 1;
      else
            return 1 + mystery1(_n >> 1_____);
}
```

Practice Problems - 2

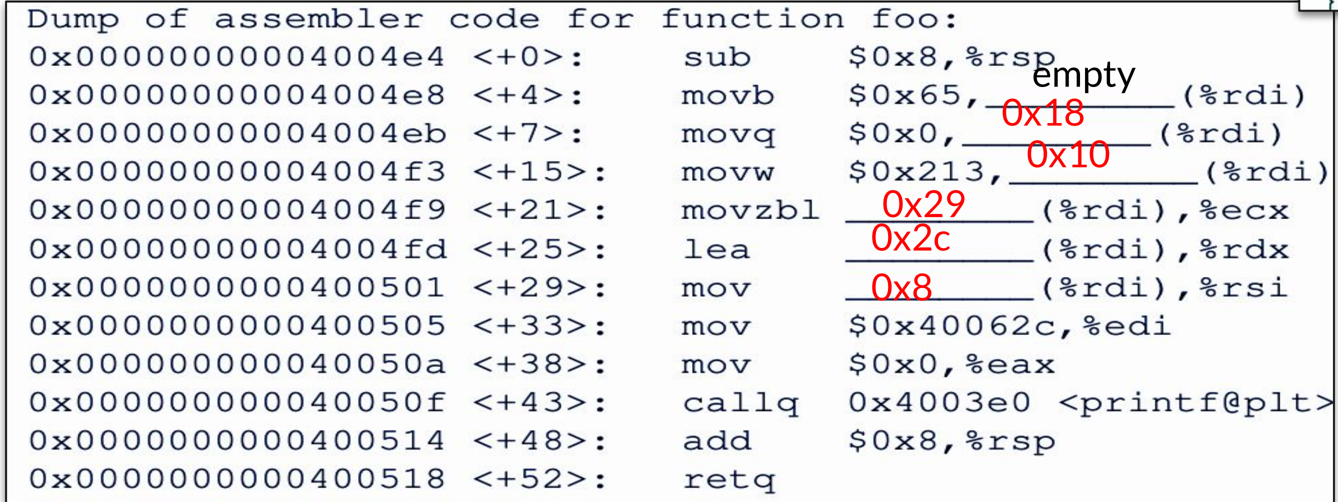


```
void foo(struct my_struct *st) {
    st->a = 'e';
    st->d[0] = NULL;
    st->c = 0x213;

    printf("%lld %p %hhu\n", st->b, &st->f, st->e[1]);
}
```



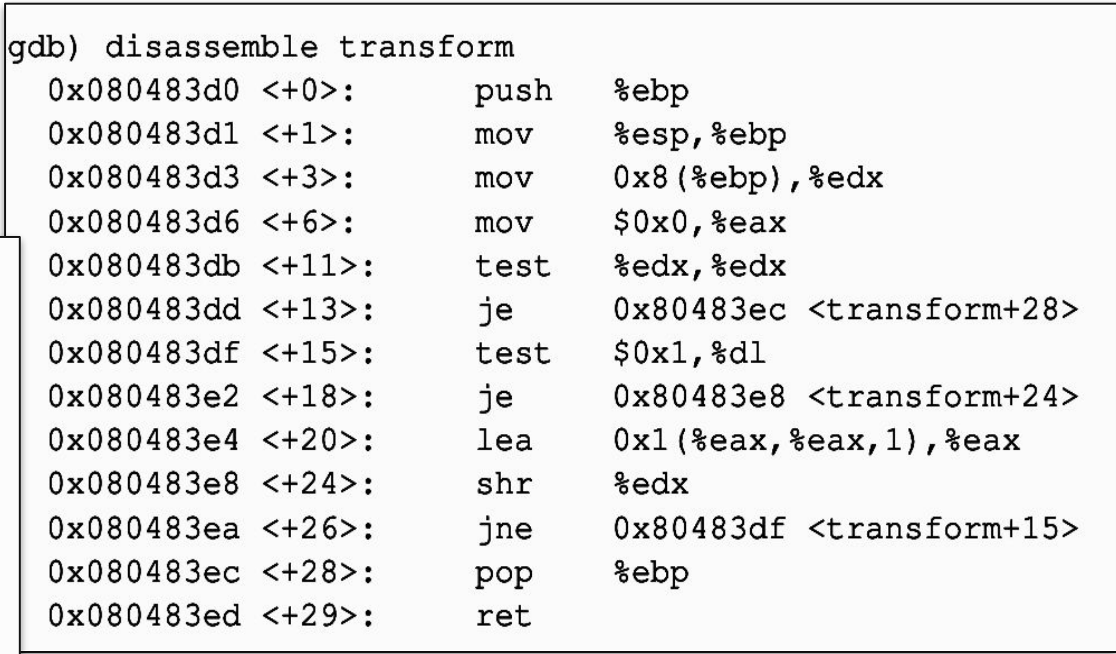
```
struct my_struct {
    char a;
    long long b;
    short c;
    float *d[2];
    unsigned char e[3];
    float f;
};
```



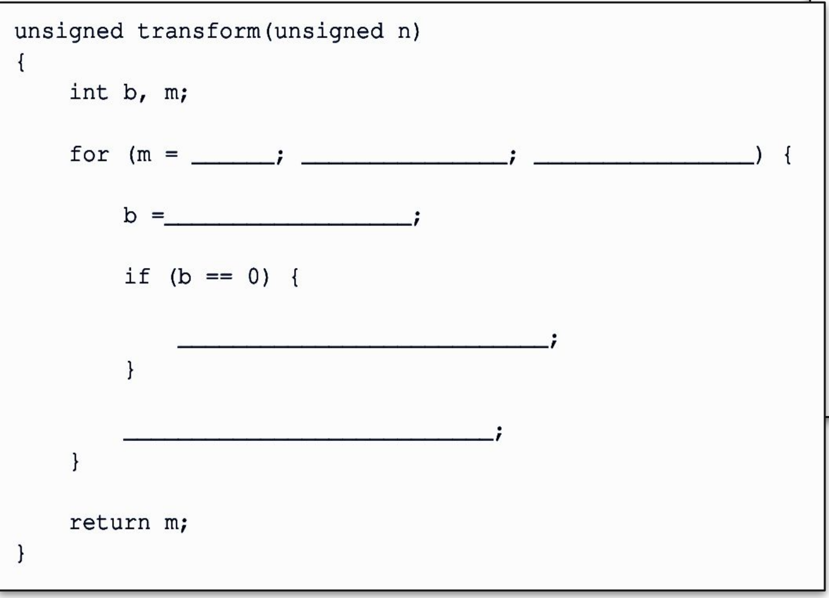
```
Dump of assembler code for function foo:
0x00000000004004e4 <+0>:     sub     $0x8,%rsp
0x00000000004004e8 <+4>:     movb    $0x65,__empty__(%rdi)
0x00000000004004eb <+7>:     movq    $0x0,__0x18__(%rdi)
0x00000000004004f3 <+15>:    movw    $0x213,__0x10__(%rdi)
0x00000000004004f9 <+21>:    movzbl  __0x29__(%rdi),%ecx
0x00000000004004fd <+25>:    lea     __0x2c__(%rdi),%rdx
0x0000000000400501 <+29>:    mov     __0x8__(%rdi),%rsi
0x0000000000400505 <+33>:    mov     $0x40062c,%edi
0x000000000040050a <+38>:    mov     $0x0,%eax
0x000000000040050f <+43>:    callq   0x4003e0 <printf@plt>
0x0000000000400514 <+48>:    add     $0x8,%rsp
0x0000000000400518 <+52>:    retq
```

How would the layout look like?

Practice Problems - 3



```
gdb) disassemble transform
   0x080483d0 <+0>:      push   %ebp
   0x080483d1 <+1>:      mov    %esp,%ebp
   0x080483d3 <+3>:      mov    0x8(%ebp),%edx
   0x080483d6 <+6>:      mov    $0x0,%eax
   0x080483db <+11>:     test   %edx,%edx
   0x080483dd <+13>:     je     0x80483ec <transform+28>
   0x080483df <+15>:     test   $0x1,%dl
   0x080483e2 <+18>:     je     0x80483e8 <transform+24>
   0x080483e4 <+20>:     lea    0x1(%eax,%eax,1),%eax
   0x080483e8 <+24>:     shr    %edx
   0x080483ea <+26>:     jne    0x80483df <transform+15>
   0x080483ec <+28>:     pop    %ebp
   0x080483ed <+29>:     ret
```



```
unsigned transform(unsigned n)
{
    int b, m;

    for (m = ______; ______________; _______________) {

        b =_________________;

        if (b == 0) {

           ___________________________;
        }

        ___________________________;
    }

    return m;
}
```

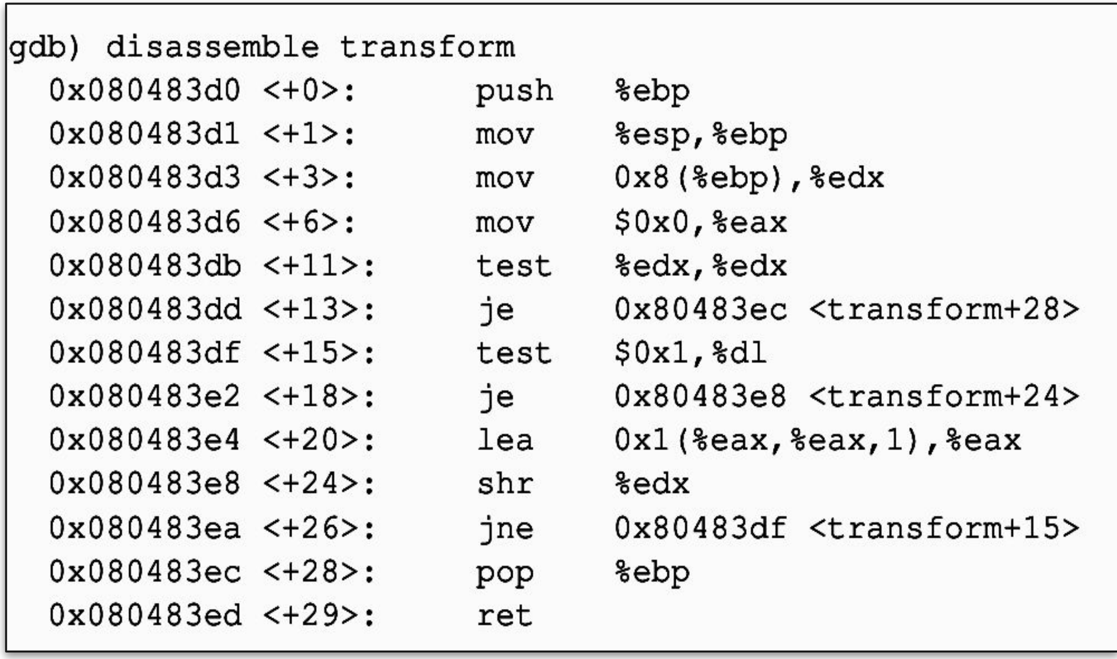
Solution to this question on the next slide

Practice Problems - 3

```
unsigned transform(unsigned n)
{
    int b, m;

    for(m = 0; n != 0; n >>= 1) { // (or)
for(m = 0; n > 0; n = n/2)

     b = n & 1; // (or) b = n % 2;
      if(b == 0) {
          continue;
      }
       m = 2*m + 1; // (or) m = m + m + 1;
(or) m = m<<1 + 1;
}
   return m;
}
```



```
gdb) disassemble transform
   0x080483d0 <+0>:      push   %ebp
   0x080483d1 <+1>:      mov    %esp,%ebp
   0x080483d3 <+3>:      mov    0x8(%ebp),%edx
   0x080483d6 <+6>:      mov    $0x0,%eax
   0x080483db <+11>:     test   %edx,%edx
   0x080483dd <+13>:     je     0x80483ec <transform+28>
   0x080483df <+15>:     test   $0x1,%dl
   0x080483e2 <+18>:     je     0x80483e8 <transform+24>
   0x080483e4 <+20>:     lea    0x1(%eax,%eax,1),%eax
   0x080483e8 <+24>:     shr    %edx
   0x080483ea <+26>:     jne    0x80483df <transform+15>
   0x080483ec <+28>:     pop    %ebp
   0x080483ed <+29>:     ret
```

Worksheet

https://tinyurl.com/cs33-endgame