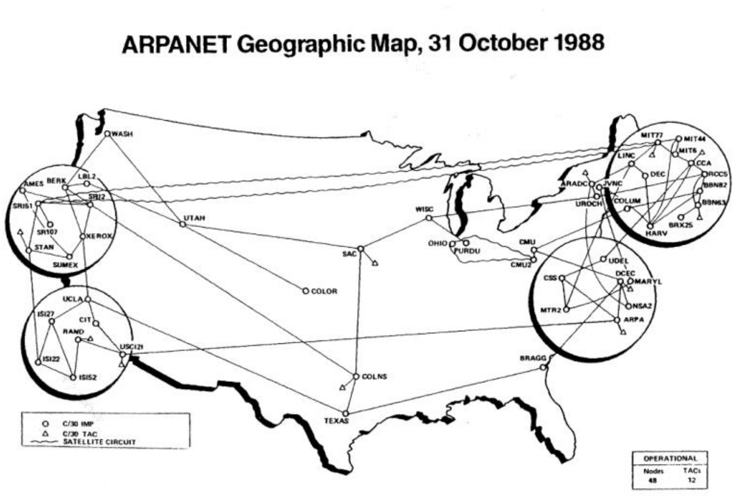
# Lecture 8: Buffer Overflows

CS 105 Fall 2024

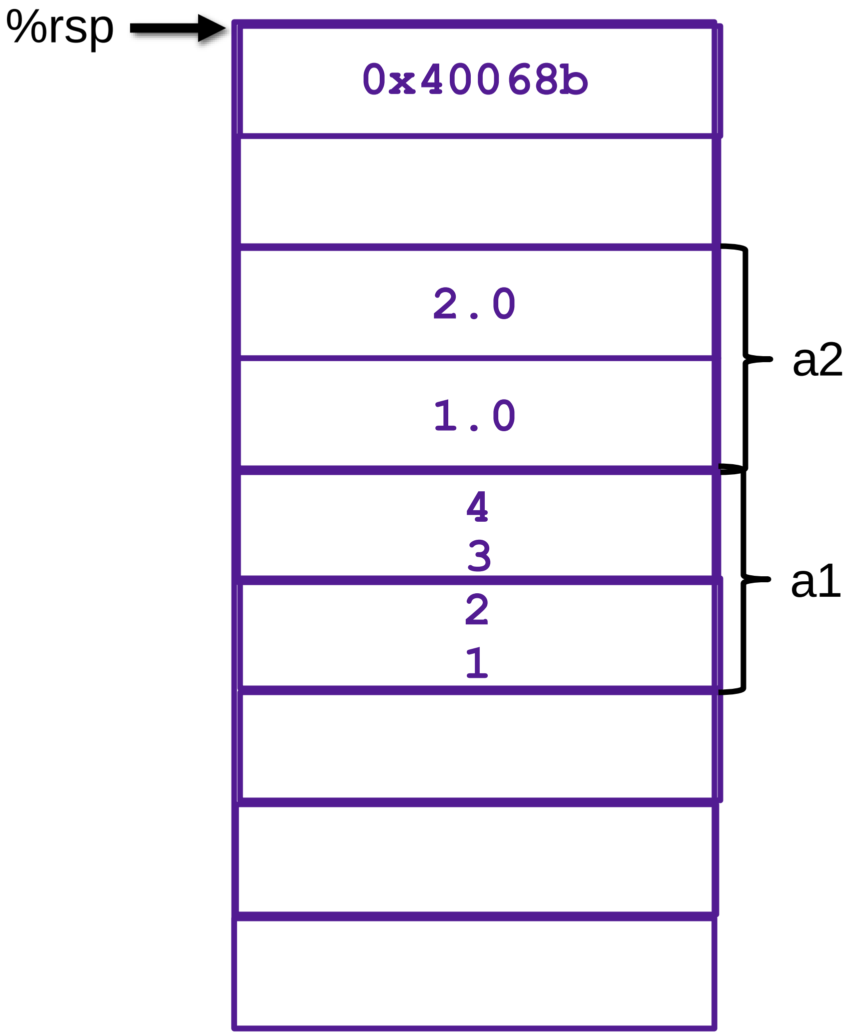
# Buffer Overflow Examples



# Review: Function Calls in Assembly

```
void f1(){
   double a2[2] = {1.0,2.0};
   int a1[4] = {1,2,3,4};
```

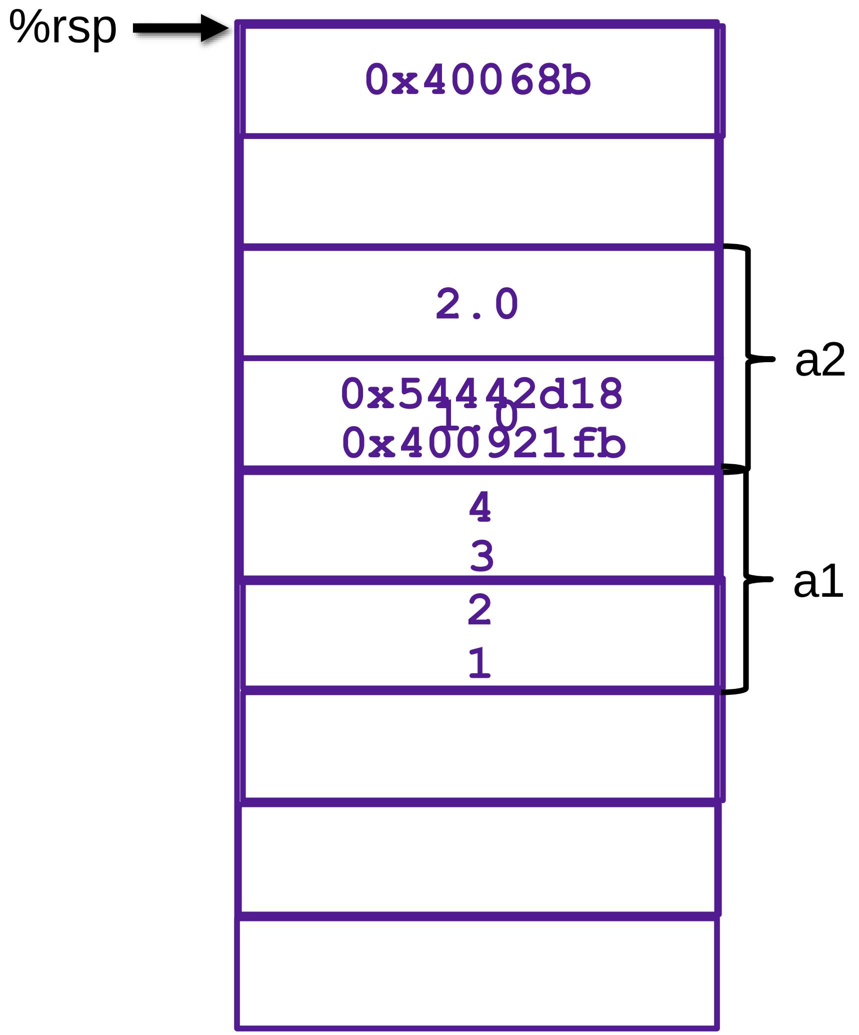
```
f1:
    sub     $0x28,%rsp
    movsd   0x216(%rip),%xmm0
    movsd   %xmm0,0x10(%rsp)
    movsd   0x210(%rip),%xmm0
    movsd   %xmm0,0x18(%rsp)
    movl    $0x1,(%rsp)
    movl    $0x2,0x4(%rsp)
    movl    $0x3,0x8(%rsp)
    movl    $0x4,0xc(%rsp)
    add     $0x28,%rsp
    retq
main:
    call f1
    retq
```



# Memory Referencing Bug Example

```
void f1(){
  double a2[2] = {1.0,2.0};
  int a1[4] = {1,2,3,4};
  a1[4] = 1413754136;
  a1[5] = 1074340347;
```

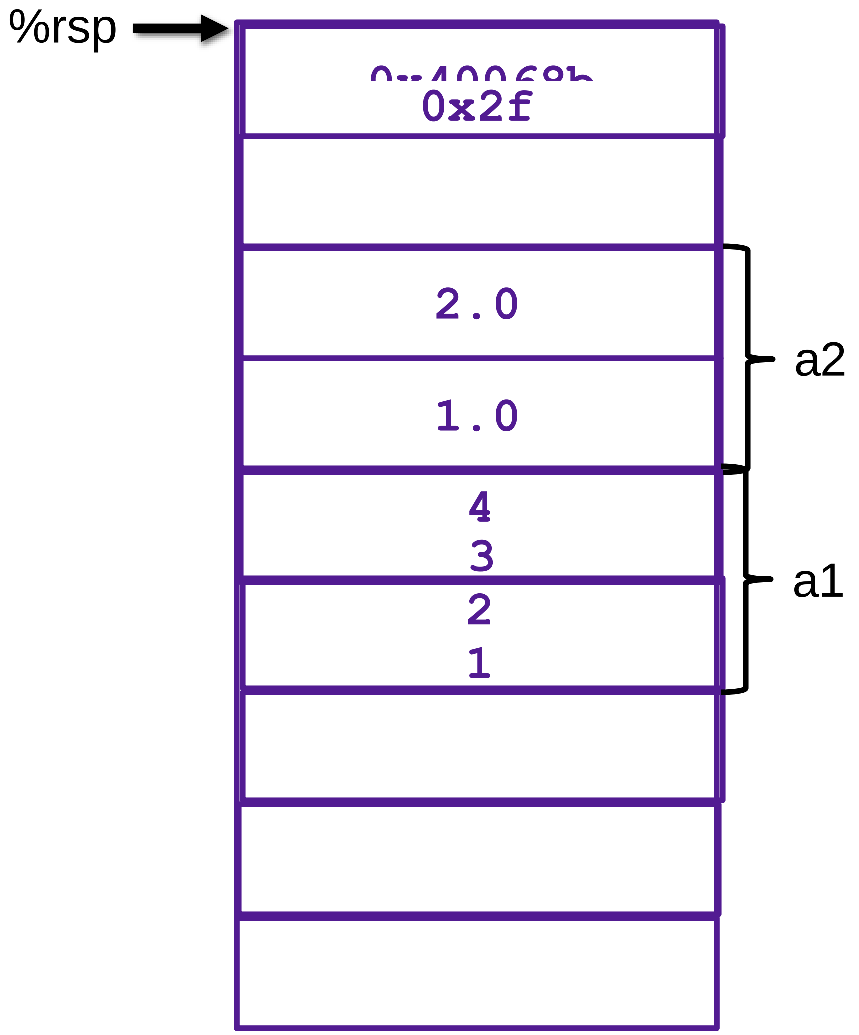
```
f1:
   sub     $0x28,%rsp
   movsd   0x216(%rip),%xmm0
   movsd   %xmm0,0x10(%rsp)
   movsd   0x210(%rip),%xmm0
   movsd   %xmm0,0x18(%rsp)
   movl    $0x1,(%rsp)
   movl    $0x2,0x4(%rsp)
   movl    $0x3,0x8(%rsp)
   movl    $0x4,0xc(%rsp)
   movl    $0x54442d18,0x10(%rsp)
   movl    $0x400921fb,0x14(%rsp)
   add     $0x28,%rsp
   retq
```



# Memory Referencing Bug Example

```
void f1(){
  double a2[2] = {1.0,2.0};
  int a1[4] = {1,2,3,4};
  a1[10] = 47;
}
```

```
f1:
   sub      $0x28,%rsp
   movsd    0x216(%rip),%xmm0
   movsd    %xmm0,0x10(%rsp)
   movsd    0x210(%rip),%xmm0
   movsd    %xmm0,0x18(%rsp)
   movl     $0x1,(%rsp)
   movl     $0x2,0x4(%rsp)
   movl     $0x3,0x8(%rsp)
   movl     $0x4,0xc(%rsp)
   movl     $0x2f,0x28(%rsp)
   add      $0x28,%rsp
   retq
```

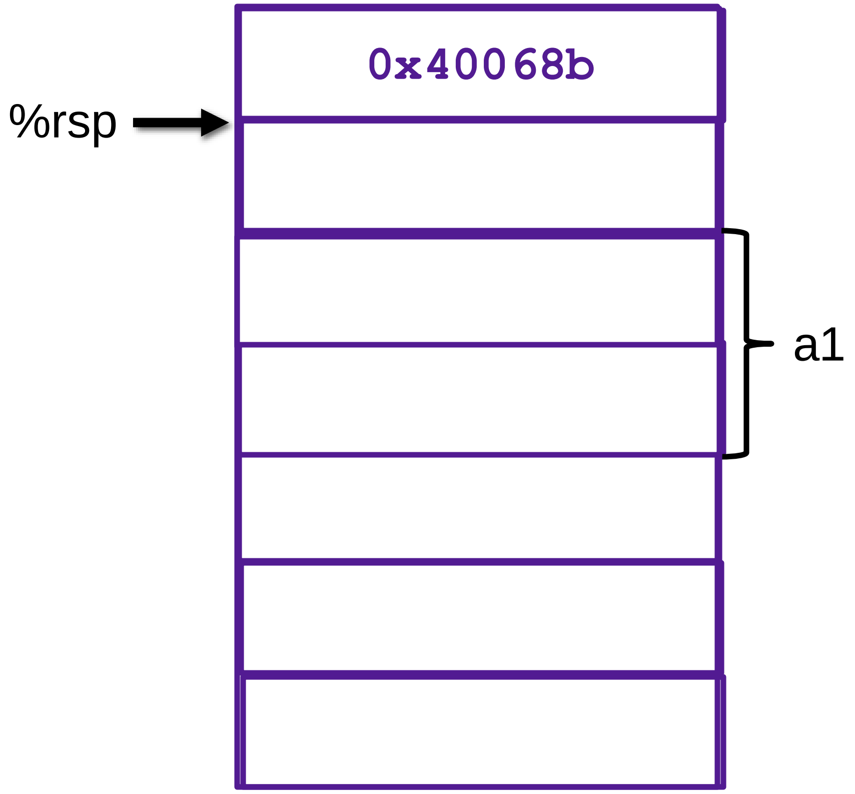


# Exercise 1: Memory Bugs

- What is the state of the stack immediately before the program returns from f2?
- What will happen immediately after f2 returns?

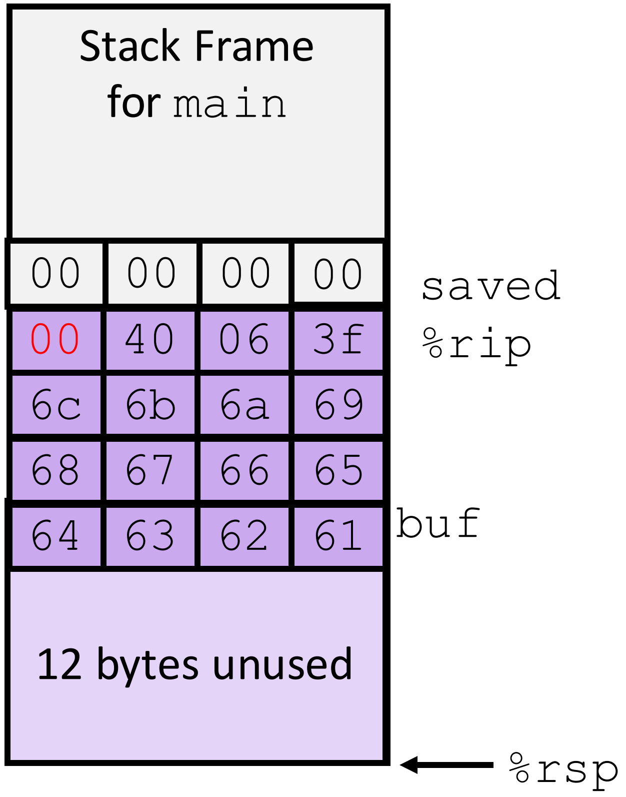
```
int f2(){
  int a1[4] = {1,2,3,4};
  a1[6] = 0x400667;
}
```

```
f2:
   sub    $0x18,%rsp
   movl   $0x1,(%rsp)
   movl   $0x2,0x4(%rsp)
   movl   $0x3,0x8(%rsp)
   movl   $0x4,0xc(%rsp)
   movl   $0x400667,0x18(%rsp)
   add    $0x18,%rsp
   retq
```



# Buffer Overflows

- Most common form of memory reference bug
  - Unchecked lengths on string inputs
  - Particularly for bounded character arrays on the stack



```
/* Echo Line */
void echo(){
    char buf[4];
    gets(buf);
    puts(buf);
}
```

```
echo:
  subq   $0x18, %rsp
  lea    0xc(%rsp),%rdi
  call   gets
  lea    0xc(%rsp),%rdi
  call   puts
  addq   $0x18, %rsp
  ret
```

# Exercise 2: Buffer Overflow

Construct an exploit string that will successfully cause the program to print "You are now logged in" without knowing the correct password

1. How many bytes of padding are in this exploit string?
2. What value will you overwrite the return address with?

```
int authenticate(char *password){
  char buf[4];
  gets(buf);
  int correct = !strcmp(password, buf);
  return correct;
}

int main(int argc, char ** argv){
  char * pw = "123456";
  printf("Enter your password: ");
  while(!authenticate(pw)){
    printf("Incorrect. Try again: ");
  }
  printf("You are now logged in\n");
  return 0;
}
```

# Exercise 2: Buffer C

Construct an exploit string that
program to print "You are now l
the correct password

1. How many bytes of padding
2. What value will you overwrit

```
int authenticate(char *password){
  char buf[4];
  gets(buf);
  int correct = !strcmp(password, buf);
  return correct;
}

int main(int argc, char ** argv){
  char * pw = "123456";
  printf("Enter your password: ");
  while(!authenticate(pw)){
    printf("Incorrect. Try again: ");
  }
  printf("You are now logged in\n");
  return 0;
}
```

```
authenticate:
   0x400666 <+0>:   sub     $0x28,%rsp
   0x40066a <+4>:   mov     %rdi,0x8(%rsp)
   0x40066f <+9>:   lea     0x18(%rsp),%rax
   0x400674 <+14>:  mov     %rax,%rdi
   0x400677 <+17>:  mov     $0x0,%eax
   0x40067c <+22>:  callq   0x400570 <gets@plt>
   0x400681 <+27>:  lea     0x18(%rsp),%rdx
   0x400686 <+32>:  mov     0x8(%rsp),%rax
   0x40068b <+37>:  mov     %rdx,%rsi
   0x40068e <+40>:  mov     %rax,%rdi
   0x400691 <+43>:  callq   0x400560 <strcmp@plt>
   0x400696 <+48>:  test    %eax,%eax
   0x400698 <+50>:  sete    %al
   0x40069b <+53>:  movzbl  %al,%eax
   0x40069e <+56>:  mov     %eax,0x1c(%rsp)
   0x4006a2 <+60>:  mov     0x1c(%rsp),%eax
   0x4006a6 <+64>:  add     $0x28,%rsp
   0x4006aa <+68>:  retq
main:
   0x4006ab <+0>:  sub     $0x28,%rsp
   0x4006af <+4>:  mov     %edi,0xc(%rsp)
   0x4006b3 <+8>:  mov     %rsi,(%rsp)
   0x4006b7 <+12>:  movq    $0x4007a8,0x18(%rsp)
   0x4006c0 <+21>:  mov     $0x4007af,%edi
   0x4006c5 <+26>:  mov     $0x0,%eax
   0x4006ca <+31>:  callq   0x400550 <printf@plt>
   0x4006cf <+36>:  jmp     0x4006e0 <main+53>
   0x4006d1 <+38>:  mov     $0x4007c8,%edi
   0x4006d6 <+43>:  mov     $0x0,%eax
   0x4006db <+48>:  callq   0x400550 <printf@plt>
   0x4006e0 <+53>:  mov     0x18(%rsp),%rax
   0x4006e5 <+58>:  mov     %rax,%rdi
   0x4006e8 <+61>:  callq   0x400666 <authenticate>
   0x4006ed <+66>:  test    %eax,%eax
   0x4006ef <+68>:  je      0x4006d1 <main+38>
   0x4006f1 <+70>:  mov     $0x4007e8,%edi
   0x4006f6 <+75>:  callq   0x400540 <puts@plt>
   0x4006fb <+80>:  mov     $0x0,%eax
   0x400700 <+85>:  add     $0x28,%rsp
   0x400704 <+89>:  retq
```

## Exercise 2: Buffer C

```
authenticate:
   0x400666 <+0>:   sub    $0x28,%rsp
   0x40066a <+4>:   mov    %rdi,0x8(%rsp)
   0x40066f <+9>:   lea    0x18(%rsp),%rax
   0x400674 <+14>:  mov    %rax,%rdi
   0x400677 <+17>:  mov    $0x0,%eax
   0x40067c <+22>:  callq  0x400570 <gets@plt>
   0x400681 <+27>:  lea    0x18(%rsp),%rdx
   0x400686 <+32>:  mov    0x8(%rsp),%rax
   0x40068b <+37>:  mov    %rdx,%rsi
   0x40068e <+40>:  mov    %rax,%rdi
   0x400691 <+43>:  callq  0x400560 <strcmp@plt>
   0x400696 <+48>:  test   %eax,%eax
   0x400698 <+50>:  sete   %al
   0x40069b <+53>:  movzbl %al,%eax
   0x40069e <+56>:  mov    %eax,0x1c(%rsp)
   0x4006a2 <+60>:  mov    0x1c(%rsp),%eax
   0x4006a6 <+64>:  add    $0x28,%rsp
   0x4006aa <+68>:  retq
main:
   0x4006ab <+0>:  sub    $0x28,%rsp
   0x4006af <+4>:  mov    %edi,0xc(%rsp)
   0x4006b3 <+8>:  mov    %rsi,(%rsp)
   0x4006b7 <+12>:  movq   $0x4007a8,0x18(%rsp)
   0x4006c0 <+21>:  mov    $0x4007af,%edi
   0x4006c5 <+26>:  mov    $0x0,%eax
   0x4006ca <+31>:  callq  0x400550 <printf@plt>
   0x4006cf <+36>:  jmp    0x4006e0 <main+53>
   0x4006d1 <+38>:  mov    $0x4007c8,%edi
   0x4006d6 <+43>:  mov    $0x0,%eax
   0x4006db <+48>:  callq  0x400550 <printf@plt>
   0x4006e0 <+53>:  mov    0x18(%rsp),%rax
   0x4006e5 <+58>:  mov    %rax,%rdi
   0x4006e8 <+61>:  callq  0x400666 <authenticate>
   0x4006ed <+66>:  test   %eax,%eax
   0x4006ef <+68>:  je     0x4006d1 <main+38>
   0x4006f1 <+70>:  mov    $0x4007e8,%edi
   0x4006f6 <+75>:  callq  0x400540 <puts@plt>
   0x4006fb <+80>:  mov    $0x0,%eax
   0x400700 <+85>:  add    $0x28,%rsp
   0x400704 <+89>:  retq
```