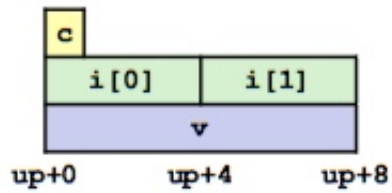
Machine-Level Programming: Advanced

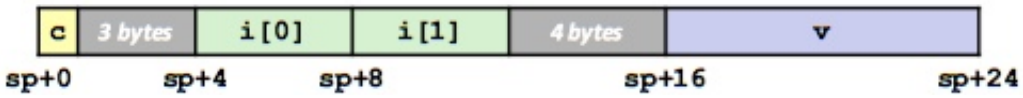
Union Allocation

- allocate according to largest element
- can only use on field at a time

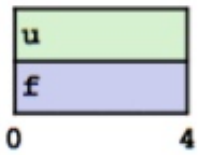
```
union U1 {
    char c;
    int i[2];
    double v;
} *up;
```

```
struct S1 {
    char c;
    int i[2];
    double v;
} *sp
```





Using Union to Access Bit Patterns



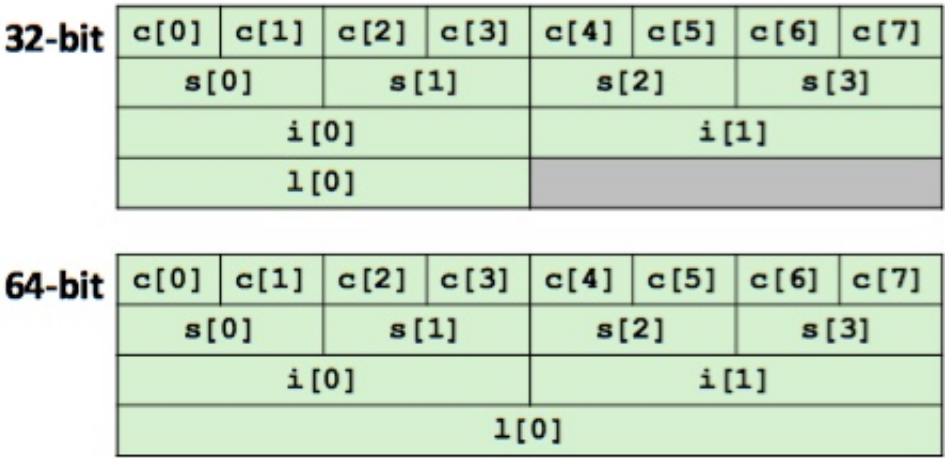

```
typedef union {
    float f;
    unsigned u;
} bit_float_t;
```

```
float bit2float(unsigned u) {
    bit_float_t arg;
    arg.u = u;
    return arg.f;
}
```

```c
unsigned float2bit(float f) {
    bit_float_t arg;
    arg.f = f;
    return arg.u;
}
```

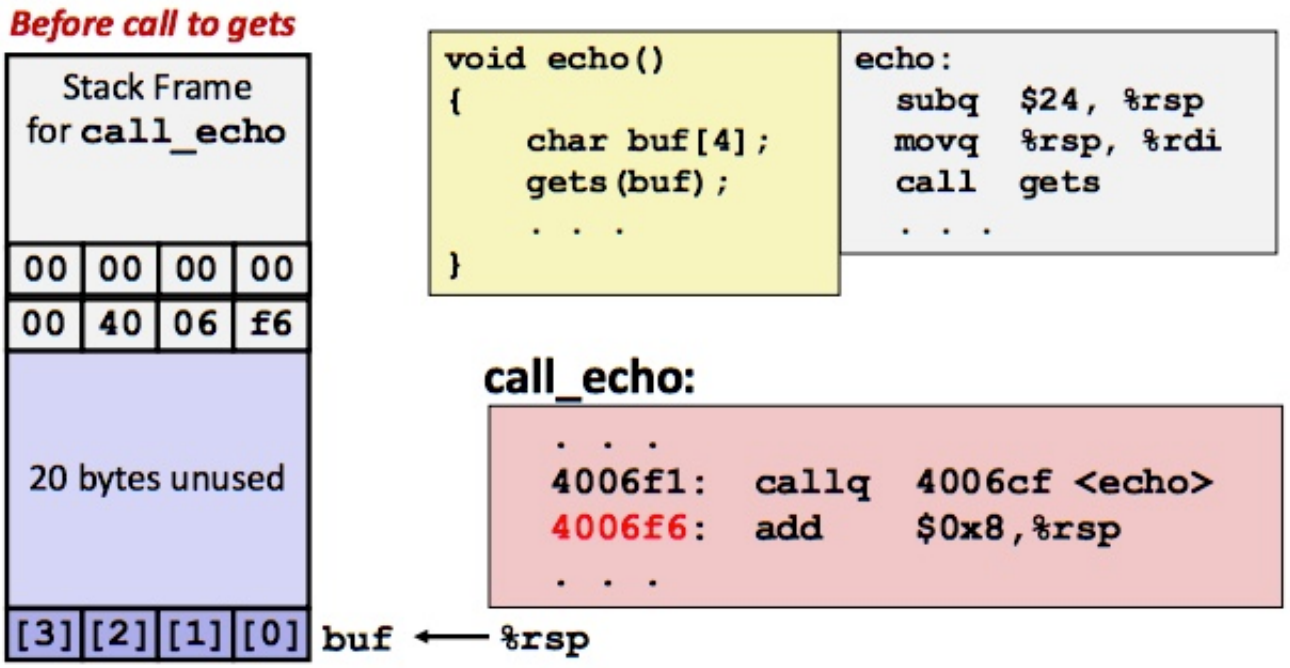
Byte Ordering Example

```c
union {
    unsigned char c[8];
    unsigned short s[4];
    unsigned int i[2];
    unsigned long l[1];
} dw;
```

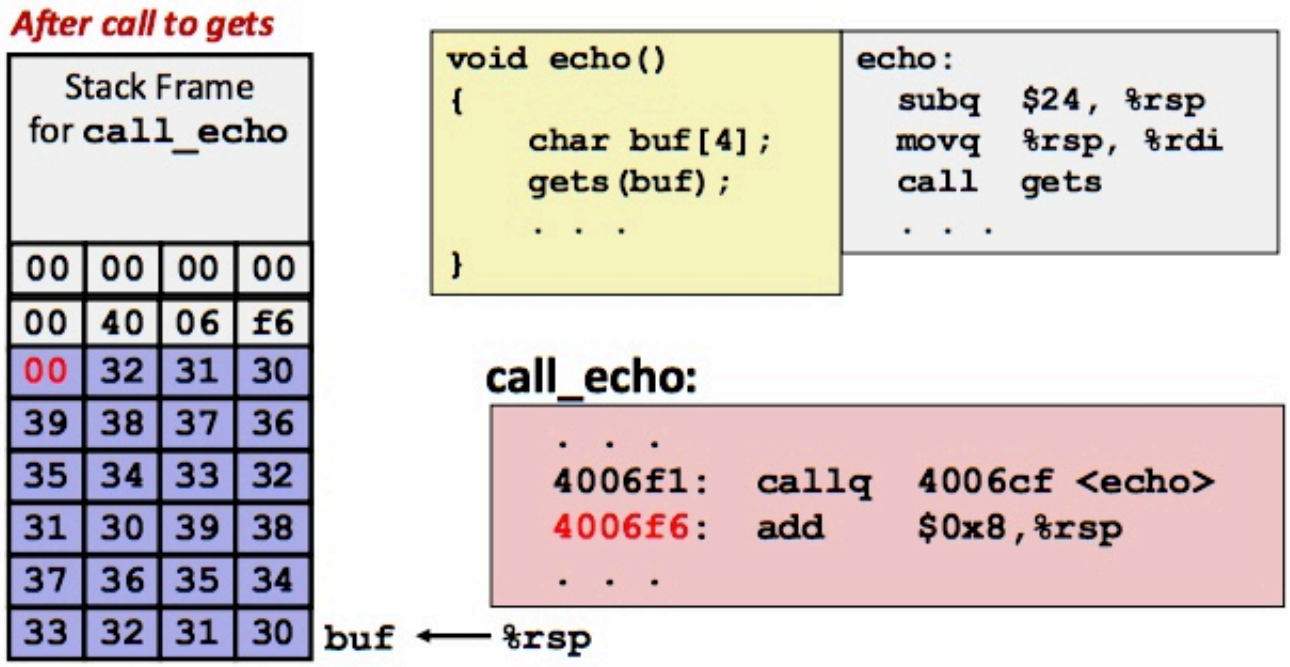


Buffer Overflow

- string library code for functions like `gets`, `sscanf`, etc. would not check for buffer overflow
- hackers take advantage of this to precisely change the return address in the stack
    - changes program flow to run what the hacker wants

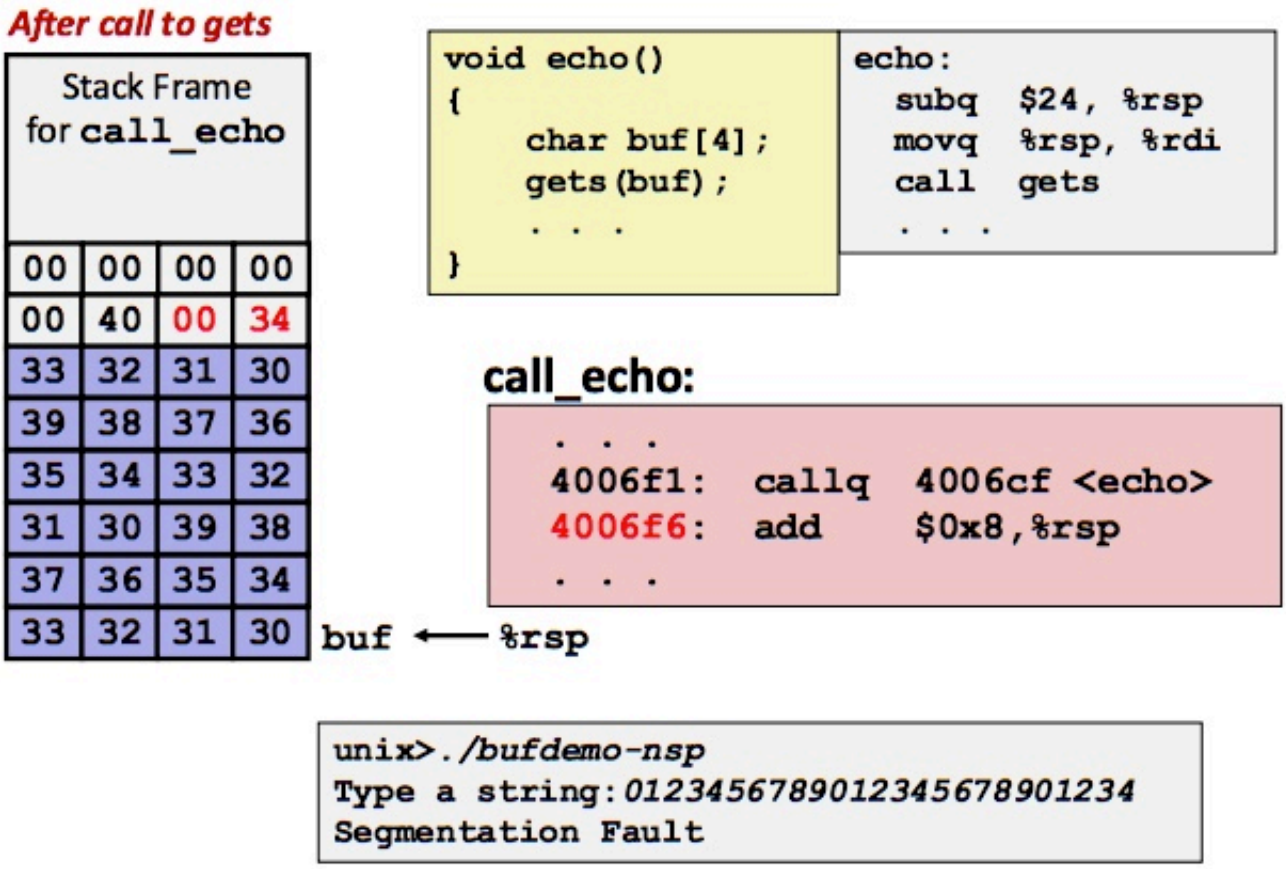


Doesn't cause a problem




```
unix> ./bufdemo-nsp
Type a string: 0123456789012345678901 2
0123456789012345678901 2
```

Changes return address



Avoiding Overflow Vulnerabilities

- use `fgets` instead of `gets`
- `strncpy` instead of `strcpy`
- don't use `scanf` with `%s` conversion specification
  - use `fgets` to read the string
  - or use `%ns` where n is a suitable integer