



C Introduction

CS 2130: Computer Systems and Organization 1
October 27, 2025

Announcements

- Homework 6 **due tonight at 11:59pm**

C

C is a thin wrapper around assembly

- This is by design!
- Invented to write an operating system
 - Can write inline assembly in C
- Many other languages decided to look like C

Helpful Resources

- Wikipedia
- Our Reference and Summary

`sizeof()` - returns size in bytes

- `sizeof(int)` returns 4

Data Types in C

Integer data types

unsigned int y = 32;

	Data type	Size	<u>sizeof</u>
<i>signed</i>	char	8 bits	1
	short	16 bits	2
	int	32 bits	4
	long	64 bits	8
	long long	64 bits	8

Each has 2 versions: *signed* and *unsigned*

Data Types in C

Floating point

- float
- double

Data Types in C

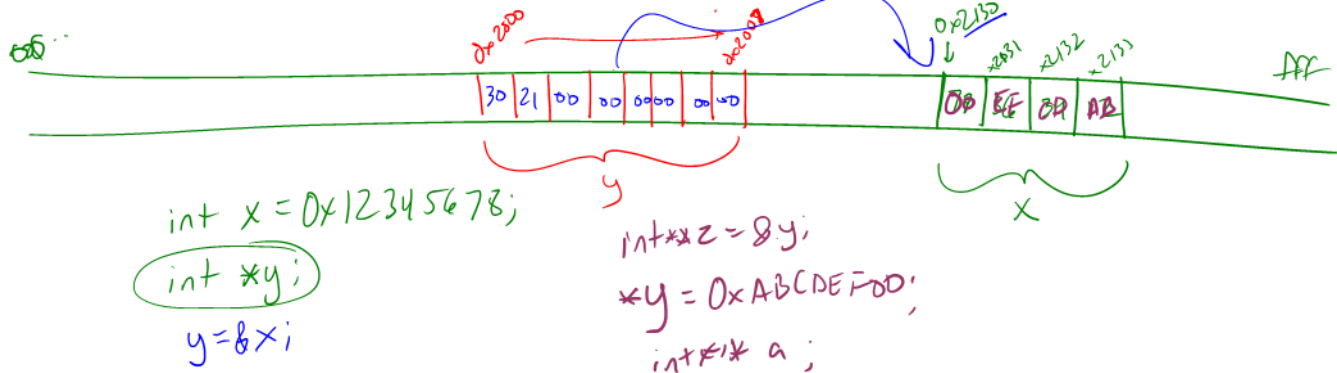
Data Types in C

Pointers - how C uses addresses!

Data Types in C

Pointers - how C uses addresses!

- Hold the address of a position in memory
- Need to know the kind of information stored at that location



Example

```
int main() {  
    int x = 3;  
    long y = 4;  
    int *a = &x;  
    long *b = &y;  
    long z = *a;  
    int w = *b;  
    return 0;  
}
```

Example

```
int main() {  
    int x = 3;  
    long y = 4;  
    int *a = &x;  
    long *b = &y;  
    long z = *a;  
    int w = *b;  
    return 0;  
}
```

0000000000000000 <main>:

```
0: 55  
1: 48 89 e5  
4: 31 c0  
6: c7 45 fc 00 00 00 00  
d: c7 45 f8 03 00 00 00  
14: 48 c7 45 f0 04 00 00  
1b: 00  
1c: 48 8d 4d f8  
20: 48 89 4d e8  
24: 48 8d 4d f0  
28: 48 89 4d e0  
2c: 48 8b 4d e8  
30: 48 63 09  
33: 48 89 4d d8  
37: 48 8b 4d e0  
3b: 48 8b 09  
3e: 89 4d d4  
41: 5d  
42: c3
```

```
push    %rbp  
mov     %rsp,%rbp  
xor     %eax,%eax  
movl    $0x0,-0x4(%rbp)  
movl    $0x3,-0x8(%rbp)  
movq    $0x4,-0x10(%rbp)  
  
lea     -0x8(%rbp),%rcx  
mov     %rcx,-0x18(%rbp)  
lea     -0x10(%rbp),%rcx  
mov     %rcx,-0x20(%rbp)  
mov     -0x18(%rbp),%rcx  
movslq  (%rcx),%rcx  
mov     %rcx,-0x28(%rbp)  
mov     -0x20(%rbp),%rcx  
mov     (%rcx),%rcx  
mov     %ecx,-0x2c(%rbp)  
pop     %rbp  
retq
```

Arrays

Array: 0 or more values of same type stored contiguously in memory

- Declare as you would use: `int myarr[100];`
- `sizeof(myarr) = 400` — 100 4-byte integers
- `myarr` treated as pointer to first element
- Can declare array literals:
`int y[5] = {1, 1, 2, 3, 5}`

Pointers and Arrays

`*x` and `x[0]` are equivalent

- Pointer to single value and pointer to first value in array
- Treat array as pointer to the first value (lowest address)
- Indexing into array: `x[n]` and `*(x+n)`
 - If `x` is an `int *`, then `x+1` points to **next int** in memory
 - Adding 1 to pointer adds `sizeof()` the type we're pointing to