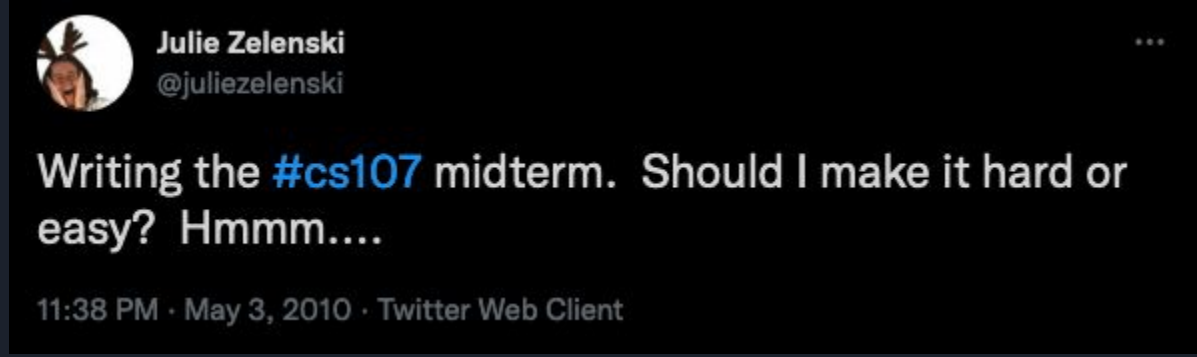


Section 11
Intro x86 / Review

CS 107A, Autumn 2021
Andrew Benson (adbenson@)

Don’t forget to start recording

Unix Tip Spotlight

- gdb conditional breakpoints
  - Breakpoints, but only if some condition about the code is true
  - break myfile.c:46 if i == 35

Announcements

- Midterm is tonight, in like 3 hours, in Bishop / STLC 111, 7:15-9:15pm
- My recommendations:
  - Sit near the aisle
  - Skim over the midterm first to get a feel for what's being tested
  - Watch time very carefully from the start
  - You'll do awesome!
- Midquarter Feedback Survey
  - https://forms.gle/B6zwdF77aWa7LyubA
  - Also on course homepage

Bird’s Eye View

Day	Week 6 Monday	Tuesday	Wednesday	Thursday	Friday	Week 7 Monday	Tuesday	Wednesday
CS 107A		Section: Intro x86 / Review		Section: x86 Basics			No Class - Democracy Day	
CS 107	Lecture: x86-64 ALU		Lab 5: Assembly		Lecture: x86-64 Control Flow	Lecture: x86 Runtime Stack		Lab: Runtime Stack
CS 107 assignments		Midterm						assign4 due, assign5 released

Agenda

- C Compilation Process
- Comparing C and Assembly
- Midterm Last-Minute Review

C Compilation Process

C Compilation

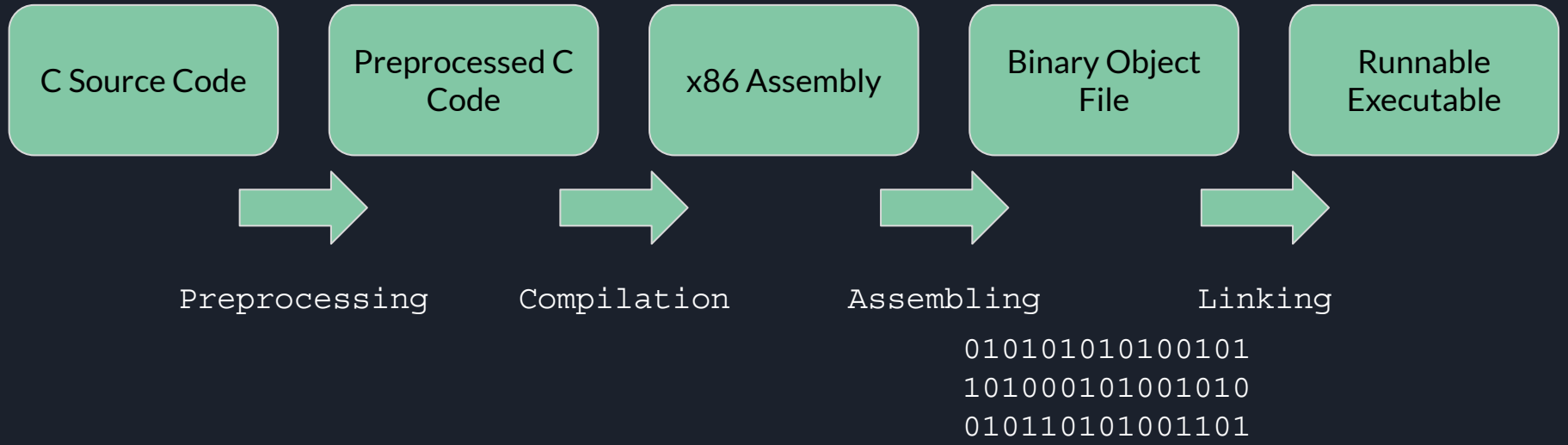
```
#define MAGIC 7

int x = 5;
x += MAGIC;
```

```
int x = 5;
x += 7;
```

```
movl $5, %rax
addl $7, %rax
```

```
0101010101001010
1010001010010100
0101101010011010
```

```
./myuniq
```



Assembly

- Lots of different kinds, just like there's lot of programming languages
- Tied to hardware
- x86 (Intel, AMD): Most laptops, most desktops, most servers, some game consoles
    - Specifically x86-64, the 64-bit version
- ARM (Qualcomm, Apple, Samsung, NVIDIA): iPhones, most Androids, the new MacBooks, Internet of Things, Nintendo Switch
- RISC-V (SiFive): the new hotness, maybe?

Comparing C and Assembly

Indentation

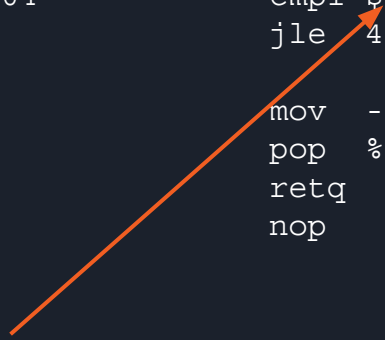
```
int main()
{
  int n = 0;
  for (int i = 0; i < 5; i++) {
    n += i;
  }
  return n;
}
```

```
00000000004004d6 <main>:
  4004d6:    55                      push %rbp
  4004d7:    48 89 e5                mov  %rsp,%rbp
  4004da:    c7 45 f8 00 00 00 00    movl $0x0,-0x8(%rbp)
  4004e1:    c7 45 fc 00 00 00 00    movl $0x0,-0x4(%rbp)
  4004e8:    eb 0a                   jmp  4004f4
<main+0x1e>
  4004ea:    8b 45 fc                mov  -0x4(%rbp),%eax
  4004ed:    01 45 f8                add  %eax,-0x8(%rbp)
  4004f0:    83 45 fc 01             addl $0x1,-0x4(%rbp)
  4004f4:    83 7d fc 04             cmpl $0x4,-0x4(%rbp)
  4004f8:    7e f0                   jle  4004ea
<main+0x14>
  4004fa:    8b 45 f8                mov  -0x8(%rbp),%eax
  4004fd:    5d                      pop  %rbp
  4004fe:    c3                      retq
  4004ff:    90                      nop
```

Constants (“Immediates”)

```
int main()
{
  int n = 0;
  for (int i = 0; i < 5; i++) {
    n += i;
  }
  return n;
}
```

```
00000000004004d6 <main>:
  4004d6:    55                       push %rbp
  4004d7:    48 89 e5                 mov  %rsp,%rbp
  4004da:    c7 45 f8 00 00 00 00     movl $0x0,-0x8(%rbp)
  4004e1:    c7 45 fc 00 00 00 00     movl $0x0,-0x4(%rbp)
  4004e8:    eb 0a                    jmp  4004f4
<main+0x1e>
  4004ea:    8b 45 fc                 mov  -0x4(%rbp),%eax
  4004ed:    01 45 f8                 add  %eax,-0x8(%rbp)
  4004f0:    83 45 fc 01              addl $0x1,-0x4(%rbp)
  4004f4:    83 7d fc 04              cmpl $0x4,-0x4(%rbp)
  4004f8:    7e f0                    jle  4004ea
<main+0x14>
  4004fa:    8b 45 f8                 mov  -0x8(%rbp),%eax
  4004fd:    5d                       pop  %rbp
  4004fe:    c3                       retq
  4004ff:    90                       nop
```



Variables

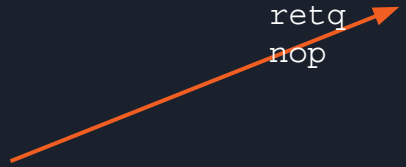
```
int main()
{
  int n = 0;
  for (int i = 0; i < 5; i++) {
    n += i;
  }
  return n;
}
```

```
0000000004004d6 <main>:
  4004d6:    55                      push %rbp
  4004d7:    48 89 e5                mov  %rsp,%rbp
  4004da:    c7 45 f8 00 00 00 00    movl $0x0,-0x8(%rbp)
  4004e1:    c7 45 fc 00 00 00 00    movl $0x0,-0x4(%rbp)
  4004e8:    eb 0a                   jmp  4004f4
<main+0x1e>
  4004ea:    8b 45 fc                mov  -0x4(%rbp),%eax
  4004ed:    01 45 f8                add  %eax,-0x8(%rbp)
  4004f0:    83 45 fc 01             addl $0x1,-0x4(%rbp)
  4004f4:    83 7d fc 04             cmpl $0x4,-0x4(%rbp)
  4004f8:    7e f0                   jle  4004ea
<main+0x14>
  4004fa:    8b 45 f8                mov  -0x8(%rbp),%eax
  4004fd:    5d                      pop  %rbp
  4004fe:    c3                      retq
  4004ff:    90                      nop
```

Registers (x86 only)

```
int main()
{
  int n = 0;
  for (int i = 0; i < 5; i++) {
    n += i;
  }
  return n;
}
```

```
00000000004004d6 <main>:
  4004d6:    55                       push %rbp
  4004d7:    48 89 e5                 mov  %rsp,%rbp
  4004da:    c7 45 f8 00 00 00 00     movl $0x0,-0x8(%rbp)
  4004e1:    c7 45 fc 00 00 00 00     movl $0x0,-0x4(%rbp)
  4004e8:    eb 0a                    jmp  4004f4
<main+0x1e>
  4004ea:    8b 45 fc                 mov  -0x4(%rbp),%eax
  4004ed:    01 45 f8                 add  %eax,-0x8(%rbp)
  4004f0:    83 45 fc 01              addl $0x1,-0x4(%rbp)
  4004f4:    83 7d fc 04              cmpl $0x4,-0x4(%rbp)
  4004f8:    7e f0                    jle  4004ea
<main+0x14>
  4004fa:    8b 45 f8                 mov  -0x8(%rbp),%eax
  4004fd:    5d                       pop  %rbp
  4004fe:    c3                       retq
  4004ff:    90                       nop
```



General-Purpose Registers

64-bit	RAX	RBX	RCX	RDX	RSI	RDI	RBP	RSP	R8	R9	R10	R11	R12	R13	R14	R15
32-bit	EAX	EBX	ECX	EDX	ESI	EDI	EBP	ESP	R8D	R9D	R10D	R11	R12D	R13D	R14D	R15D
16-bit	AX	BX	CX	DX	SI	DI	BP	SP	R8W	R9W	R10W	R11W	R12W	R13W	R14W	R15W
8-bit	AL	ZBL	CL	DL	SIL	DIL	BPL	SPL	R8B	R9B	R10B	R11B	R12B	R13B	R14B	R15B

Midterm
Last-Minute Review

[bits] What does this function do?

```
char mystery(char c) {
    int x = 0;
    for (int y = 0; y < 4; y++) {
        if ((c >> 2*y) & 0x3 == 0x3) {
            x++;
        }
    }
    return x;
}
```

[bits] Find 0b1101

- Write a function that takes in a `char` and returns true if and only if the input's binary representation has the bit sequence "1101" anywhere in it.

[strings] Implement `strcat`

- Write a function that implements `strcat`.

[generics] Alphanumeric sum

- Write a generic comparison function for strings such that a string is considered more than another string if and only if the sum of the ASCII values of its letters is more than the other's.