CS107, Lecture 15 Extras
Introduction to Assembly

Reading: B&O 3.1-3.4
Ed Discussion: https://edstem.org/us/courses/46162/discussion/3715585
Extra Practice
1. Extra Practice

Fill in the blank to complete the C code that

```c
int x = ...
int *ptr = malloc(...);
...
___???___ = _???_;
```

mov %ecx,(%rax)

1. mystery line compiles to this assembly
2. registers hold these values

 вал of x> %ecx 
вал of ptr> %rax

(Pedantic: You should sub in <x> and <ptr> with actual values, like 4 and 0x7fff80)
Fill in the blank to complete the C code that

```c
int x = ...;
int *ptr = malloc(...);
...
___???___ = _???_;  // Missing value
*ptr = x;
```

mov %ecx,(%rax)
Fill in the blank to complete the C code that
1. generates this assembly
2. results in this register layout

long arr[5];
...
long num = ____???____;

mov (%rdi, %rcx, 8),%rax

<val of num> %rax
3 %rcx
<val of arr> %rdi
2. Extra Practice

Fill in the blank to complete the C code that
generates this assembly
results in this register layout

long arr[5];
...
long num = ____???____;

long num = arr[3];
long num = *(arr + 3);
long num = *(arr + y);

(assume long y = 3; declared earlier)

mov (%rdi, %rcx, 8),%rax

<val of num> 3 <val of arr>
Fill in the blank to complete the C code that 1. generates this assembly 2. has this register layout

char str[5];
...
___???___ = 'c';

mov $0x63,(%rcx,%rdx,1)

[val of str] [2]
3. Extra Practice

Fill in the blank to complete the C code that 1. generates this assembly 2. has this register layout

```c
char str[5];

... 
___???___ = 'c';

str[2] = 'c';
*(str + 2) = 'c';
```

```asm
mov $0x63,(%rcx,%rdx,1)
```

<val of str>  2

%rcx  %rdx
The below code is the objdump of a C function, foo.

- foo keeps its 1st and 2nd parameters are in registers %rdi and %rsi, respectively.

```
0x4005b6 <foo>    mov   (%rdi),%rax
0x4005b9 <foo+3>  mov   (%rsi),%rdx
0x4005bc <foo+6>  mov   %rdx,(%rdi)
0x4005bf <foo+9>  mov   %rax,(%rsi)
```

1. What does this function do?
2. What C code could have generated this assembly?

(Hints: make up C variable names as needed, assume all regs 64-bit)
The below code is the objdump of a C function, `foo`.

- `foo` keeps its 1st and 2nd parameters are in registers `%rdi` and `%rsi`, respectively.

```asm
0x4005b6 <foo>    mov   (%rdi),%rax
0x4005b9 <foo+3>   mov   (%rsi),%rdx
0x4005bc <foo+6>   mov   %rdx,(%rdi)
0x4005bf <foo+9>   mov   %rax,(%rsi)
```

---

**Bonus: Sneak peek into next week**

- The below code is the objdump of a C function, `foo`.
- `foo` keeps its 1st and 2nd parameters are in registers `%rdi` and `%rsi`, respectively.