CS107, Lecture 15 Extras
Introduction to Assembly

Reading: B&O 3.1-3.4
Ed Discussion: https://edstem.org/us/courses/28214/discussion/2047914
Extra Practice
Fill in the blank to complete the C code that

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

mov %ecx,(%rax)

(val of x) %ecx
(val of ptr) %rax

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

(Pedantic: You should sub in <x> and <ptr> with actual values, like 4 and 0x7ffe80)
Fill in the blank to complete the C code that

```c
int x = ...;
int *ptr = malloc(...);
...
___???___ = _???_;    *ptr = x;
mov %ecx,(%rax)
```

<val of x> <val of ptr>

%ecx  %rax
Fill in the blank to complete the C code that

1. generates this assembly
2. results in this register layout

```c
long arr[5];
...
long num = ___???___;

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

<val of num> 3 <val of arr>

%rax %rcx %rdi
Fill in the blank to complete the C code that generates this assembly results in this register layout

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

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

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

(assume long y = 3; declared earlier)
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] %rcx
2 %rdx
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';
```

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

<table>
<thead>
<tr>
<th>val of str</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>%rcx</td>
<td>%rdx</td>
</tr>
</tbody>
</table>
• 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)
Bonus: Sneak peek into next week

- The below code is the objdump of a C function, `foo`.
  - `foo` keeps its 1\textsuperscript{st} and 2\textsuperscript{nd} 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)
```

```
<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>42</td>
<td>0x7ffe870</td>
</tr>
<tr>
<td>1000</td>
<td>0x7ffe868</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x7ffe868</td>
<td>%rdi</td>
</tr>
<tr>
<td>0x7ffe870</td>
<td>%rsi</td>
</tr>
<tr>
<td></td>
<td>%rax</td>
</tr>
<tr>
<td></td>
<td>%rdx</td>
</tr>
</tbody>
</table>
```

```