CS107, Lecture 15

Introduction to Assembly, Take II

Reading: B&O 3.1-3.4

Ed Discussion

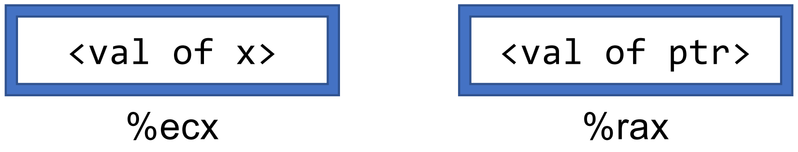
Extra Practice 1

Fill in the blank to complete the C code that

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

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

```
mov %ecx,(%rax)
```



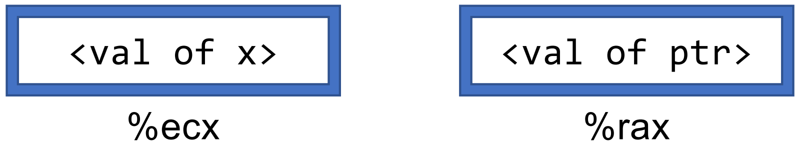
Try subbing in <x> and <ptr> with actual values, like 4 and 0x7fff80

Extra Practice 1

Fill in the blank to complete the C code that

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

```
mov %ecx,(%rax)
```



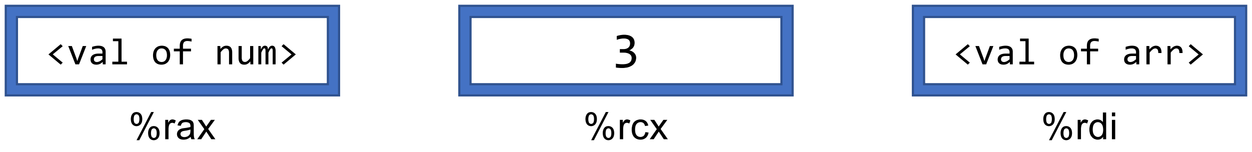
Extra Practice 2

Fill in the blank to complete the C code that

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

```
long *arr = malloc(…);
...
long num = ____???___;
```

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



Extra Practice 2

Fill in the blank to complete the C code that

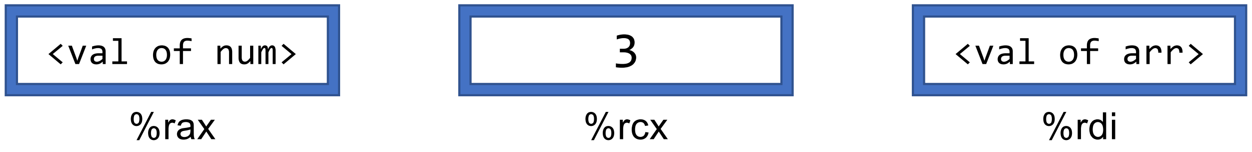
1. generates this assembly
2. **results in** this register layout

```
long *arr = malloc(…);
...
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
```



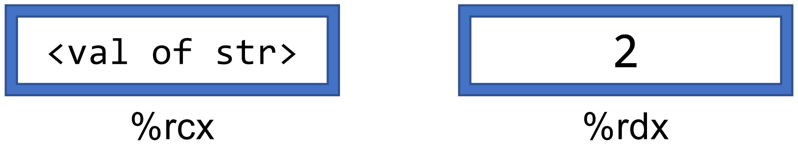
Extra Practice 3

Fill in the blank to complete the C code that

1. generates this assembly
2. has this register layout

```
char *str = malloc(…);
long i = 2;
___???___ = 'c';
```

```
movb $0x63,(%rcx,%rdx,1)
```



Extra Practice 3

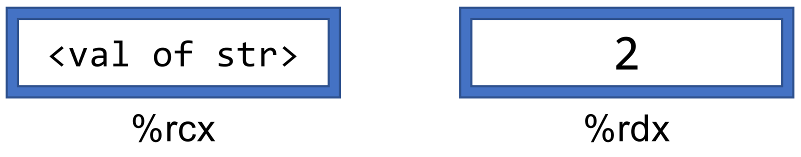
Fill in the blank to complete the C code that

1. generates this assembly
2. has this register layout

```
char *str = malloc(…);
long i = 2;
___???___ = 'c';
```

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

```
movb $0x63,(%rcx,%rdx,1)
```



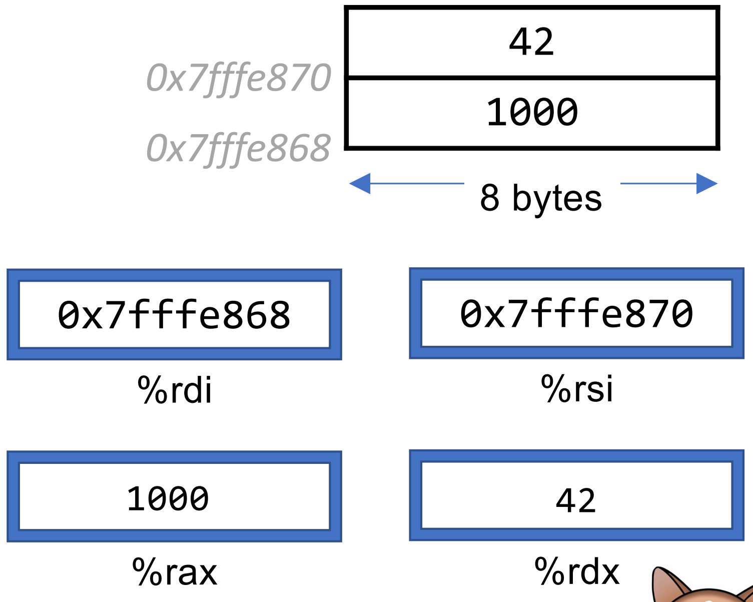
Bonus: Sneak peek into next week

- 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.

```
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)

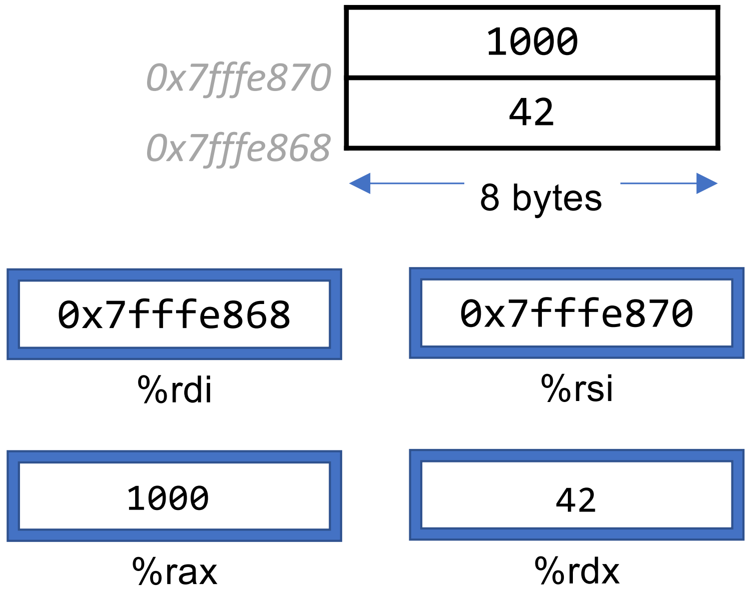


Bonus: Sneak peek into next week

- The below code is the **`objdump`** of a C function, **`foo`**.
  - `foo` keeps its 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)
```

```
void foo(long *xp, long *yp) {
  long a = *xp;
  long b = *yp;
  *yp = a;
  *xp = b;
  ...
```



1000
42
0x7fffe870
0x7fffe868
8 bytes
0x7fffe868
%rdi
0x7fffe870
%rsi
1000
%rax
42
%rdx