CS 240 SI Worksheet #15
Valerie Zhao
4/4/17

Call Stack

1. On the third page (in the given table), simulate the state of the call stack when `main()` calls **`treat(7, &x)`**, right up to (not including) when line **`0x400638 (add $0x18, %rsp)`** is executed for any of the recursive calls.

`&x = 0x7ff…ffb00, *(&x) = 5`
`%rsp` starts at `0x7fffffffffffad0` (the top row of the table).
Make sure to keep track of `%rsp` in addition to the other register contents.

```
long int treat(long int a, long int* b) {
  if (a <= 0) {
    return *b;
  } else {
    return treat(a-*b, b);
  }
}
```

```
4005fc <treat>:
4005fc: sub $0x18,%rsp
400600: mov %rdi,0x8(%rsp)
400605: mov %rsi,(%rsp)
400609: cmpq $0x0,0x8(%rsp)
40060f: jg 0x40061a <treat+30>
400611: mov (%rsp),%rax
400615: mov (%rax),%rax
400618: jmp 0x400638 <treat+60>
40061a: mov (%rsp),%rax
40061e: mov (%rax),%rax
400621: mov 0x8(%rsp),%rdx
400626: sub %rax,%rdx
400629: mov (%rsp),%rax
40062d: mov %rax,%rsi
400630: mov %rdx,%rdi
400633: callq 0x4005fc <treat>
400638: add $0x18,%rsp
40063c: retq
```

(Adapted from CSE351 Autumn 2014 – Midterm Exam (29 October 2014), Problem 4A: https://courses.cs.washington.edu/courses/cse351/16au/past-exams/cse351-14au-midterm.pdf)

What happens when the execution finishes and `treat(7, &x)` returns to `main()`?
(In other words, what is different between the registers and stack you completed in the next page, vs. the final contents of the stack and the registers after the function returns to `main()`?)

%rip = 0x400827
%rsp = `0x7fffffffffffad0 + 8 = 0x7fffffffffffad8`
Nothing else is different, including the state of the memory that was once the stack.

2. How do callee-saved registers work? What do functions do with them?

A function `f` preserves the value of a callee-saved register by pushing it onto the stack; afterwards, the function is free to change the register value. However, before the function terminates, it must pop the stack value back into that register (which requires that this pop instruction is executed when `%rsp` is where the original register value is on the stack). The register's value is identical right before `f` was called and after `f` returns.

%rdi	%rsi	%rdx	%rax
-3	0x7ff…ffb00	-3	0x7ff…ffb00

Memory address on stack	**Name/description of item**	**Value**
0x7fffffffffffad0	Return address back to main	0x400827
0x7fffffffffffac8		
0x7fffffffffffac0	(0x400600) %rdi	7
0x7fffffffffffab8 ~~%rsp (0x4005fc)~~	(0x400605) %rsi	0x7ff…ffb00
0x7fffffffffffab0 ~~%rsp (0x400633)~~	return address	0x400638
0x7fffffffffffaa8		
0x7fffffffffffaa0	(0x400600) %rdi	2
0x7fffffffffffa98 ~~%rsp (0x4005fc)~~	(0x400605) %rsi	0x7ff…ffb00
0x7fffffffffffa90 ~~%rsp (0x400633)~~	return address	0x400638
0x7fffffffffffa88		
0x7fffffffffffa80	(0x400600) %rdi	-3
0x7fffffffffffa78 %rsp (0x400633)	(0x400605) %rsi	0x7ff…ffb00
0x7fffffffffffa70		
0x7fffffffffffa68		