15213 Lecture 7: Procedures

1 Getting Started

To obtain a copy of today's activity, log into a shark machine and do the following:

```
1.$ wget http://www.cs.cmu.edu/~213/activities/lec7.tar
2.$ tar xf lec7.tar
3.$ cd lec7
```

First run `$ ./act6` and follow the instructions on your screen. You may refer to the sheet from the first GDB activity as a reference.

2 Discussion Questions: act6

Use GDB's `c` command to progress through the activities. These questions accompany the program; as it poses each one, discuss with your partner and write your answer here.

Contents of the stack:

`0x 15213`	← `$rsp = 0x 7fffffffd620`
`0x 555555555335`	
…	

1. What was the meaning of the second number on the stack?

   The second number on the stack is the functon's return address.

2. What are the semantics of the `ret` instruction?

   `ret` pops from the top of the stack to `%rip` (incrementing `%rsp` by 8 bytes).

3. Given your knowledge of the `ret` instruction, what must be the semantics of `call`?

   `call` pushes the value of `%rip` to the stack (decrementing `%rsp` by 8 bytes), then unconditionally branches (jumps) to the call address described by the operand.

4. Why does this optimization work? Can it be used on every call?

   The optimization of `returnOneOpt` (replacing a `call` followed by a `ret` with a `jmp` to the address of the function to be called) works because the `ret` of the called function `abs` will function identically to the `ret` of the calling function `returnOneOpt` by popping the address of the caller of `returnOneOpt` into `%rip` and unconditionally branching. This optimization cannot be used on every call — consider the case where a function is called in the body of another function, instead of before a return.

5. Given your knowledge of the `printf()` function, what is the first argument used for, and what is its type?

   The first argument of `printf()` should be the format string, with type const char *

6. Where did the compiler place arguments 7 and 8? Why do you think this happened?

   Arguments 7 and 8 were pushed onto the stack in reverse order. This happened because the compiler ran out of integer argument registers.

3 Discussion Questions: act7

7. Where does the `getV()` function allocate its array? How does it pass this location to `getValue()`?

   `getV()` allocates its array on the stack. It passes this location to `getV()` by using a normal pointer stored in a standard argument register

8. What is this function doing?

   The function `mrec` is computing the factorial of its integer argument.

4 Optional Endianness Preview

Rerun `act6` with the `m` argument and continue to the point where you printed the stack before.

1. What do you expect the first two *bytes* of the stack to contain?

   We expect the first two bytes of the stack to be `0x00` and `0x00`.

2. Check your hypothesis by running `x/2xb $rsp`. In what order are each integer's bytes stored?

   We see the bytes `0x13` and `0x52` — each integer's bytes are stored <u>least</u> significant to <u>most</u> significant.