Handout 10

COMP 264: Introduction to Computer Systems (Section 001) Fall 2015
R. I. Greenberg, Comp. Sci. Dept., Loyola U., 820 N. Michigan Ave., Chicago, IL

Assignment #5

Issued 10/21
Due (at class time) 10/28

Homework is due at the time of day that class starts. Two options to submit your .c file:

- Easiest if you are working on a Loyola GNU/Linux machine: Copy your file to the directory `~rig/c264hw5sub` with a filename in the form `Email-X.c`, where `Email` is your email address, and `X` is a "random" string of at least 8 alphanumeric characters. The Unix command for this would look *similar* to:

  ```
  cp switch_prob.c ~rig/c264hw5sub/YOUREMAILADDRESS-RANDOM.c
  ```

  where you must put your own things for "YOUREMAILADDRESS" and "RANDOM". (Don't cut and paste from the PDF, or your tilde might not come out right.) *Remember that if you submit this way the file must be readable by everybody, though you will want to have used* `chmod` *to protect the directory containing the file. Protections show with the* `ls -l` *command illustrated below.* You can verify successful submision by using the "ls" command with the same file name you just copied to, specifically you can use a command *similar* to:

  ```
  ls -l ~rig/c264hw5sub/YOUREMAILADDRESS-RANDOM.c
  ```

- Or if you prefer: Submit the file through the online submission mechanism on my course web page. Submit it as `switch_prob.c` or `5.c`.

HW5-1 (55 points)

This problem involves reverse engineering of a `switch` statement from assembly code. (It could also be done from disassembled object code with a bit more work, including using the GDB debugger to inspect the content of the jump table.) In the following procedure, the body of the `switch` statement has been removed:

```
long switch_prob(long x, long n)
{
  long result = x;
  switch(n) {
    /* Fill in code here */
  }
  return result;
}
```

Following is the assembly code for the procedure (compiled with `-O1`). Remember that parameters `x` and `n` will be passed in registers `%rdi`and `%rsi`, respectively.

```
.file "switch_prob2-soln.c"
.text
.globl switch_prob
.type switch_prob, @function
switch_prob:
.LFB0:
.cfi_startproc
subq $60, %rsi
cmpq $5, %rsi
ja .L2
jmp *.L7(,%rsi,8)
.section .rodata
.align 8
.align 4
.L7:
.quad .L3
.quad .L2
.quad .L3
.quad .L4
.quad .L5
.quad .L6
.text
.L3:
leaq 0(,%rdi,8), %rax
ret
.L4:
movq %rdi, %rax
sarq $3, %rax
ret
.L5:
movq %rdi, %rax
salq $4, %rax
subq %rdi, %rax
movq %rax, %rdi
.L6:
imulq %rdi, %rdi
.L2:
leaq 75(%rdi), %rax
ret
.cfi_endproc
.LFE0:
.size switch_prob, .-switch_prob
.ident "GCC: (Ubuntu/Linaro 4.6.3-1ubuntu5) 4.6.3"
.section .note.GNU-stack,"",@progbits
```

Fill in the body of the switch statement with C code that will have the same behavior as the assembly code.