Homework 4

JUMP INSTRUCTIONS

In the following excerpts from a disassembled binary, some of the information has been replaced by `X`s. Answer the following questions about these instructions.

A. What is the target of the `je` instruction below? (You don't need to know anything about the `callq` instruction here.)

```
4003fa: 74 02                   je    XXXXXX
4003fc: ff d0                   callq *%rax
```

B. What is the target of the `je` instruction below?

```
40042f: 74 f4                   je    XXXXXX
400431: 5d                      pop   %rbp
```

C. What is the address of the `ja` and `pop` instructions?

```
XXXXXX: 77 02                   ja    400547
XXXXXX: 5d                      pop   %rbp
```

D. In the code that follows, the jump target is encoded in PC-relative form as a 4-byte, two's-complement number. The bytes are listed from least significant to most, reflecting the little-endian byte ordering of x86-64. What is the address of the jump target?

```
4005e8: e9 73 ff ff ff          jmp XXXXXX
4005ed: 90                      nop
```

CONDITIONAL MOVES

In the following C function, we have left the definition of operation `OP` incomplete:

```
#define OP ____ /* Unknown operator */

long arith(long x) {
    return x OP 8;
}
```

When compiled, GCC generates the following assembly code:

long arith(long x)

x in %rdi

```
arith:
  leaq      7(%rdi), %rax
  testq     %rdi, %rdi
  cmovns    %rdi, %rax
  sarq      $3, %rax
  ret
```

What operation is OP (only one operation) and explain how it works.

LOOPS

Executing a `continue` statement in C causes the program to jump to the end of the current loop iteration. The stated rule for translating a `for` loop into a `while` loop needs some refinement when dealing with `continue` statements. For example, consider the following code:

```
/* Example of for loop using a continue statement */
/* Sum even numbers between 0 and 9 */
long sum = 0;
long i;
for (i = 0; i < 10; i++) {
    if (i & 1)
        continue;
    sum += i;
}
```

A. What would we get if we naively applied our rule for translating the `for` loop into a `while` loop? What would be wrong with this code?
B. How could you replace the `continue` statement with a `goto` statement to ensure that the `while` loop correctly duplicates the behavior of the `for` loop?