CS 240
Laboratory 9 Assignment
X86 Disassembly and Reverse Engineering

Examine the C code on the left and the corresponding X86 code on the right for the function **test_prime**

Answer the questions below assuming that ***test_prime*** is called with **num = 7**

C code for function **test_prime**

```
long int test_prime(long int num) {

  for (long int  i =2; i <= num/2;++i)  {
    if (num % i == 0) {
        return  0;
    }
  }
  return  1;
}
```

Disassembly of X86 code produced by **gdb** for function **test_prime**

```
0x0000000000400478 :     mov    %rdi,-0x18(%rsp)
0x000000000040047c :     movq   $0x2,-0x8(%rsp)
0x0000000000400484 :     jmp    0x4004a9

0x0000000000400485:      mov    -0x18(%rsp),%rax
0x0000000000400486:      movq   $0x0,%rdx
0x0000000000400491:      idivq  -0x8(%rsp)
0x0000000000400495:      mov    %rdx,%rax
0x0000000000400498:      test   %rax,%rax
0x000000000040049b:      jne    0x4004a4
0x000000000040049d:      mov    $0x0,%rax
0x00000000004004a2:      jmp    0x4004c7

0x00000000004004a4:      addq   $0x1,-0x8(%rsp)
0x00000000004004a9:      mov    -0x18(%rsp),%rax
0x00000000004004b8:      sar    %rax
0x00000000004004bb:      cmp    -0x8(%rsp),%rax
0x00000000004004bf:      jge    0x400485

0x00000000004004c1:      mov    $0x1,%rax
0x00000000004004c7:      retq
```

1. What is the starting address of **test_prime** in memory?

2. What register is the argument stored in when the assembler code begins execution?

3. Circle and label the statements (there are two) that set the return value for the function.

4. Circle and label the X86 statements that test the condition in the **for** loop. Describe how num/2 is calculated in this code:

5. Circle and label the X86 statements that implement testing the conditional for the *if* statement in the body of the loop. Look up the *idivq* X86 instruction, and explain how the *num%2* is accomplished with the given code: