CS 240
Laboratory 8 Assignment
Disassembly and Reverse Engineering

Analyze the X86 code for the C function ***test_prime***, and answer the questions below. Assume that the function has been invoked with the argument **num**= 5.

C function to test if a number is prime

```
int test_prime(long num) {

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

Dump of assembler code produce by **gdb** for function **test_prime**

NOTE: the <+xx> on each line represents an offset from the starting address of the function.

```
0x0000000000400478 <+0>:     mov    %rdi,-0x18(%rsp)
0x000000000040047c <+4>:     movq   $0x2,-0x8(%rsp)
0x0000000000400484 <+12>:    jmp    0x4004a9 <test_prime+49>
0x0000000000400486 <+14>:    mov    -0x18(%rsp),%rax
0x000000000040048a <+18>:    mov    %rax,%rdx
0x000000000040048d <+21>:    sar    $0x3f,%rdx
0x0000000000400491 <+25>:    idivq  -0x8(%rsp)
0x0000000000400495 <+29>:    mov    %rdx,%rax
0x0000000000400498 <+32>:    test   %rax,%rax
0x000000000040049b <+35>:    jne    0x4004a4 <test_prime+44>
0x000000000040049d <+37>:    mov    $0x0,%eax
0x00000000004004a2 <+42>:    jmp    0x4004c6 <test_prime+78>
0x00000000004004a4 <+44>:    addq   $0x1,-0x8(%rsp)
0x00000000004004a9 <+49>:    mov    -0x18(%rsp),%rax
0x00000000004004ad <+53>:    mov    %rax,%rdx
0x00000000004004b0 <+56>:    shr    $0x3f,%rdx
0x00000000004004b4 <+60>:    lea    (%rdx,%rax,1),%rax
0x00000000004004b8 <+64>:    sar    %rax
0x00000000004004bb <+67>:    cmp    -0x8(%rsp),%rax
0x00000000004004bf <+71>:    jge    0x400486 <test_prime+14>
0x00000000004004c1 <+73>:    mov    $0x1,%eax
0x00000000004004c6 <+78>:    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 line where the argument gets copied to the stack.

4. Where does the initial value of 2 used in the **for** loop get stored?

5. Circle and label the X86 statements that tests the condition in the **for** loop. Does this code come at the beginning of the assembler code, as it does in the C program?

6. Circle and label the X86 statements that divides ***num*** by ***i*** and checks that the remainder is 0.

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