Assembly Practice

Address	Value	Register	Value
0x100	0x34	rax	0x108
0x104	0xA1	rcx	0x5
0x108	0x6	rdx	0x3
0x10C	0xFF	rbx	0x4

Part 1: For each instruction write the value stored in register rdi after execution.

1. mov edi, 0x110

2. mov edi, rax

3. mov edi, [0x104]

4. mov edi, [rax]

5. mov edi, [rax+4]

6. lea edi, [rax]

7. lea rdi, [rax + rcx]

8. mov edi, [rax + rdx - 11]

9. lea edi, [rbx*8 + 256]

Part 2: Does the following instructions result in the jump being executed?

i. `lea rdi, [rax + rbx]`
ii. `cmp rdi, 0x108`
iii. `jg .JUMP`

Part 3: Look at the following couple instructions and explain the following.

```
0x4012e9        lea rax, [rip+0xda4]
0x4012f0        mov rdi, rax
0x4012f3        call 0x401040 <puts@plt>
```

i. Based on your observations, what do you think its trying to load into the rax register?

ii. Why is it storing the value of rax into rdi?

Bonus:

What's the special instruction used to perform a system call? (Switching from user to kernel)