COMP311 In Class Practice 11/13/25 Name: _____

Question 1: RISC-V and Memory Tracing Consider the following recursive implementation of GCD:

```
1  reset:
2    lui   sp, 0xc0000     # initialize stack pointer
3    addi  sp, sp, 0xff0     # adjust stack
4    jal   ra, main       # call main
5  halt:
6    j    halt          # infinite halt loop
7
8  ########################################
9
10 gcd:
11   addi  sp, sp, -4      # int gcd(int x, int y)
12   sw   ra, (sp)       # save return address
13   beq   a0, a1, return    # if (x == y) return x;
14   blt   a0, a1, else     # if (x > y) ...
15   sub   a0, a0, a1      # x = x - y
16   jal   ra, gcd        # call gcd(x-y, y)
17   beq   x0, x0, return    # unconditional branch
18
19 else:
20   sub   a1, a1, a0      # y = y - x
21   jal   ra, gcd        # call gcd(x, y-x)
22
23 return:
24   lw   ra, (sp)       # restore ra
25   addi  sp, sp, 4       # pop stack
26   ret              # return
27
28 y: .word  0           # int y
29
30 main:
31   addi  sp, sp, -4      # int main() {
32   sw   ra, (sp)       # save return addr
33   li   a0, 300        # x = 300
34   li   a1, 465        # y = 465
35   jal   x1, gcd        # call gcd
36   sw   a0, y         # y = gcd(300,465)
37   lw   ra, (sp)       # restore ra
38   addi  sp, sp, 4       # pop stack
39   ret              # }
```

After the initial call `gcd(300, 465)` what are the subsequent calls?

When `gcd()` is called the 4th time, the stack has the following conntents before `addi, sp, sp-4`

0xBFFFFFF0	0x00000000	0
0xBFFFFFEC	0x0000000C	12
0xBFFFFFE8	0x00000058	88
0xBFFFFFE4	0x00000034	52
sp->0xBFFFFFE0	0x00000028	40

1.1. At what address is `return`?

- ◯ 0
- ◯ 12
- ◯ 88
- ◯ 52
- ◯ 40

1.2. At what address is `else`?

- ◯ 40
- ◯ 44
- ◯ 48
- ◯ 52
- ◯ 56

1.3. At what address is `main`?

- ◯ 68
- ◯ 72
- ◯ 80
- ◯ 84
- ◯ 88