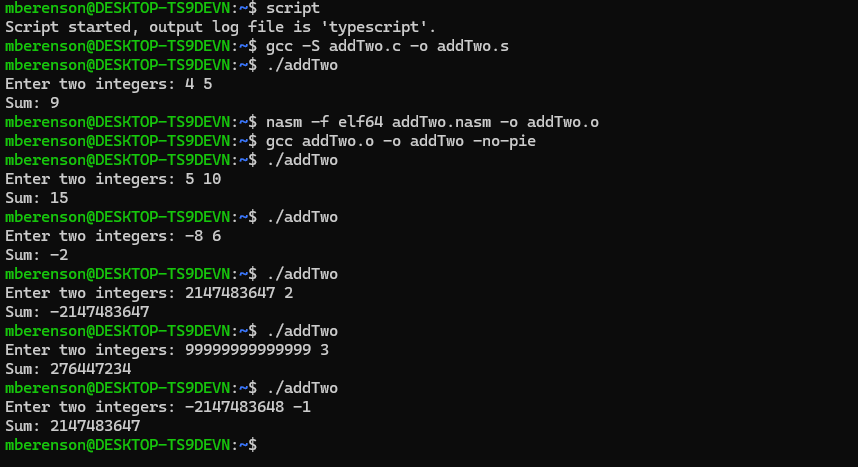
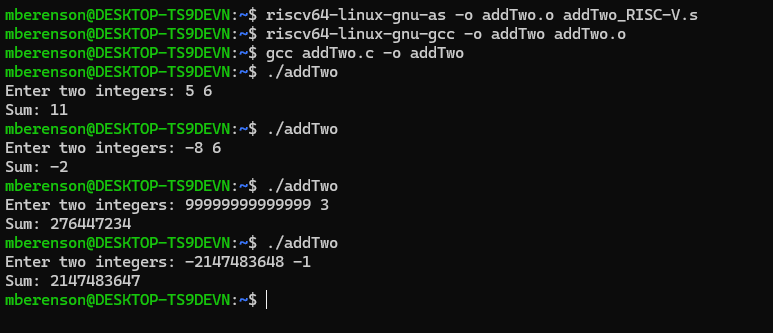
Task 1:





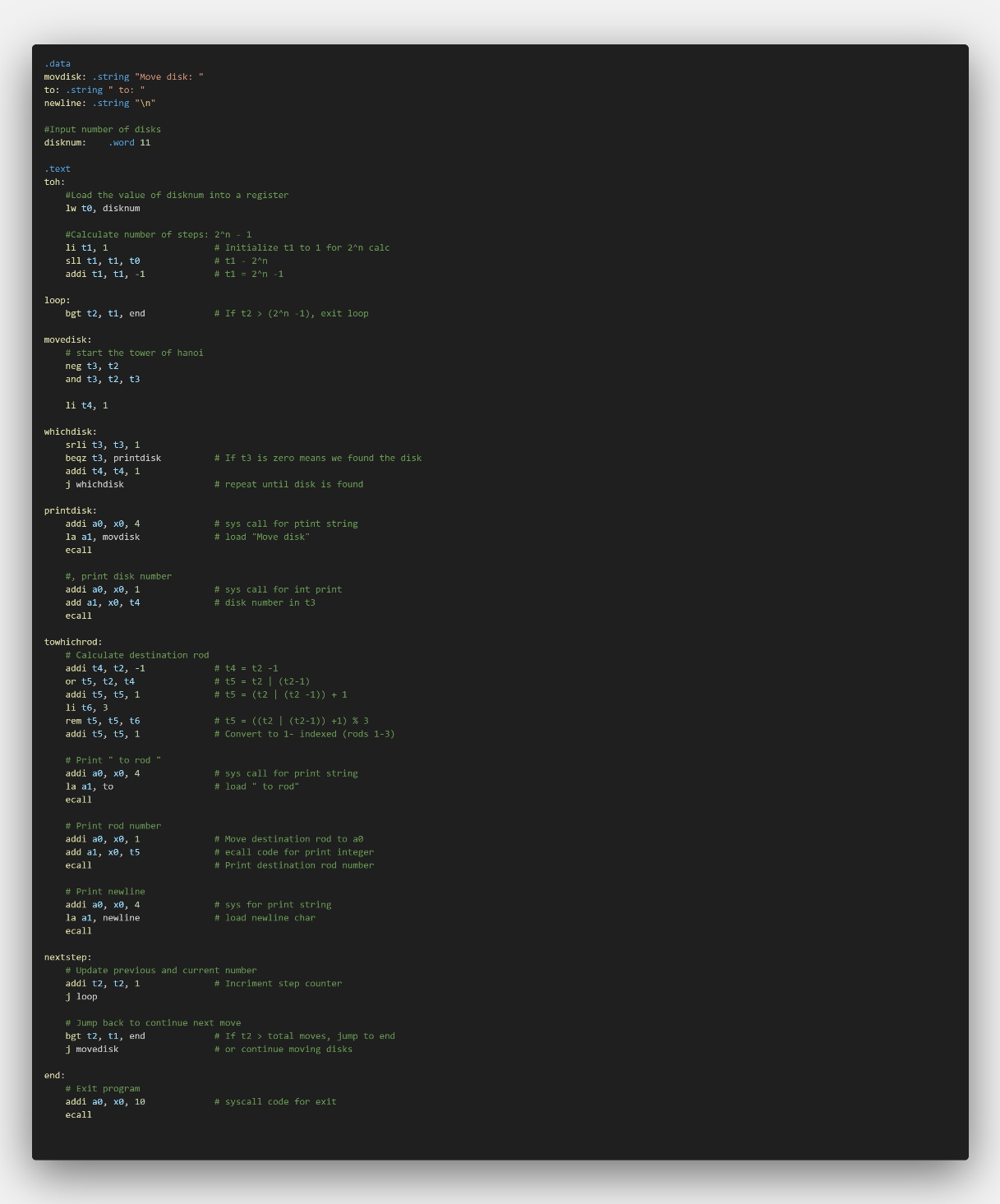
C: Reason: The input exceeds the range of a 32-bit integer. When the value is read, it is truncated to fit into a 32-bit variable, leading to incorrect additions.

Task 2:

**A.** The tower of Hanoi can be implemented in assembly by taking advantage of binary numbers to figure out which disk to move. Here's how I would go about it:

1. **Initialize registers**:
   * We first need to load the number of disks into a register (say t0), and another register (t1) will hold the total number of steps, which is 2n−12^n - 12n−1. This tells us how many moves we need to make.
2. **Create a loop for the moves**:
   * We then set up a loop that runs from 1 to 2n−12^n - 12n−1. The idea is to calculate which disk needs to be moved in each iteration. The disk to move can be found by looking at the binary representation of the move counter (t2). The smallest bit that flips tells us which disk to move.
3. **Calculate which disk to move**:
   * We find which disk to move by performing a bitwise AND operation. Specifically, disk = t2 & (-t2) will give us the bit that flipped, indicating the disk we need to move.
4. **Figure out the source and destination rods**:
   * The source rod is determined by (t2 & (t2 - 1)) % 3, and the destination rod is determined by ((t2 | (t2 - 1)) + 1) % 3. These calculations tell us where the disk came from and where it needs to go.
5. **Move the disk**:
   * After figuring out which disk to move and where to move it, we just print that out. We keep going until all moves are done.

**B. Tower of Hanoi implementation**



**C.** The largest number of disks I can solve with my current Tower of Hanoi code on Venus is around 15-16 disks. For 15 disks, the total number of moves is 215−1=32,7672^{15} - 1 = 32,767215−1=32,767, and for 16 disks, it jumps to 216−1=65,5352^{16} - 1 = 65,535216−1=65,535. Based on testing, 15-16 disks is the maximum number my code can solve efficiently on Venus before performance becomes an issue.