CSO Assignment 2
-Arushi Mittal - 2019101120

Question 4:

Purpose:

The code is the solution to the Towers of Hanoi function. It takes the number of disks as input and shows the individual steps to move all disks from one spindle to another while ensuring that a larger disk is never above a smaller disk, and only one disk is moved at a time. Each step in the process is separated by a blank line. In each step, the first line represents A, the starting tower, the second line represents B, the auxiliary tower, and the third line represents C, the final spindle.

Bugs:

- Fixed bug in line 113, changed jg to jl.

-Added line 131: mov %rsp, %rbp

Code functioning:

The number of disks is stored in %edi.

- F1: This function creates a tower. It is called thrice to create three towers: tower A, the starting tower, tower B, the auxiliary tower, and tower C, the final tower.

- F2: This function checks the topmost value stored in the tower. The tower is stored in %rdi, and the top value is checked and stored in %rax. If the tower is empty, the code moves to the procedure .peek_empty, where a large value is stored in %eax.

- F3: In addition to performing the function of F2, this also removes the topmost value stored in the tower.

- F4: This function takes the value stored in %rsi, and adds it to the top of the tower stored in %rdi.

- F5: This function moves a disk from one tower to another. It stores the current value of %rdi in %r9, and the current value of %rax in %r10, then checks the top of the tower stored in %rdi, and the top of the tower stored in %rsi, and compares them. If the tower top in %rdi is less than the one in %rsi, the code moves to the proceduer .less_branch, otherwise it moves to .greater_branch.

.greater_branch: This function swaps the values of %rdi and %rsi so .less_branch can proceed with the same registers but different values.

.less_branch: This function removes the top of the tower in %rdi, and adds it to the top of the tower in %rsi.

- Solve: The number of disks is pushed onto the stack. The three towers are created.

.init_s: This function initializes the first tower by adding the required number of disks onto the first tower. This only happens once.

- F7: This is the main loop. The top of towers A, B, and C are compared to each other and F5 is called to move the required disks to the required towers. If the task has been completed, the loop exits into F8.

- F8: This function increments the stack to move onto the next element and find the optimum way to transfer it from tower A to tower C in the required order. It then returns to F7.

- _start: This is the start function where the code begins to execute. In this case, we have three disks, so $3 is stored in %edi. If the value stored in the stack pointer is less than 1, move to .solve which calls solve. Otherwise, update %rax and %edi and call solve.

- .solve: This function simply calls the solve function as and when required.

The last three lines are standard procedure for exiting the program.