THE CODE:TOWER OF HANOI

EXPLANATION:
It consists of three rods and a number of disks of different sizes, which can slide onto any rod. The puzzle starts with the disks in a neat stack in ascending order of size on one rod, the smallest at the top.

The objective of the puzzle is to move the entire stack to another rod, obeying the following rules:

    1.Only one disk can be moved at a time.
    2.Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack or on an empty rod.
    3.No larger disk may be placed on top of a smaller disk.

BUGS IN THE CODE:

 1.in solve the frame pointer is not initialized i.e mov %rsp,%rbp(LINE 131)
 2.in LINE 113 : instead of jl given jg is given

CODE EXPLANATION:

```
.globl _start

  .data
space:   .ascii " "
newline:   .ascii "\n"

  .text

f1:#this is initialization
  xor %rax, %rax
  movl $64, %ecx
  rep stosb
  ret


f2:#peek the top of tower
  movl (%rdi),%ecx
  cmpl $0, %ecx
  jz .peek_empty
  dec %ecx
  movq 4(%rdi, %rcx, 4), %rax
  ret
.peek_empty:
  movl $100000, %eax
  ret

f3:#removes value
  movl (%rdi),%ebx
  dec %ebx
  #copy the value to %eax
  movl 4(%rdi, %rbx, 4), %eax
  #make it 0
  movl $0, 4(%rdi, %rbx, 4)
  #decrease count
  mov %ebx, (%rdi)
  ret

f4:#tower is in %rdi and the value in %rsi
  movl (%rdi),%ecx
  mov %rsi, 4(%rdi, %rcx, 4)
  inc %ecx
  mov %ecx, (%rdi)
  ret

pt:
  mov %rdi, %rsi
  movl $1, %eax
  movl %eax, %edi
  movl %eax, %edx
  syscall
  ret

pt2:
  push %rcx
  call pt
  mov $space, %rdi
  call pt
  pop %rcx
  ret

pt3:
  mov $newline, %rdi
  call pt
  ret
pt4:
  movl (%rdi), %ecx
  cmpl $0, %ecx
  jz pt6
  add $4, %rdi
  sub $8, %rsp
pt5:
  movl (%rdi), %eax
  addl $'0', %eax
  mov %rax, (%rsp)
  mov %rdi, %rbx
  mov %rsp, %rdi
  call pt2
  add $4, %rbx
  mov %rbx, %rdi
  loop pt5
  add $8, %rsp
pt6:
  call pt3
  ret

pt7:
  mov (%rbp), %rax
  andl $1, %eax
  jz pt8
  lea -64(%rbp), %rdi
  call pt4
  lea -128(%rbp), %rdi
  call pt4
  lea -192(%rbp), %rdi
  call pt4
  call pt3
  ret
pt8:
  lea -64(%rbp), %rdi
  call pt4
  lea -196(%rbp), %rdi
  call pt4
  lea -128(%rbp), %rdi
  call pt4
  call pt3
  ret

f5:#it moves a ring from the tower with smaller ring to the tower with the
larger ring
  #rdi and rsi are towers to move between
  #compare the top rings in the two towers
  mov %rdi, %r9
  call f2
  mov %rax, %r10
  mov %rsi, %rdi
  call f2
  mov %r9, %rdi
  cmp %rax, %r10
  jl .less_branch
.greater_branch:#swap rdi and rsi
  mov %rdi, %rax
  mov %rsi, %rdi
  mov %rax, %rsi
.less_branch:#sorce is rdi ,dest is rsi
  call f3
  push %rdi
  push %rsi
  mov %rsi, %rdi
  mov %rax, %rsi
  call f4
  pop %rsi
  pop %rdi
  jmp pt7

solve:
  #no of rings is in rdi
  push %rdi
  mov %rsp,%rbp
  #%(rbp) will be the ring count
  #TOWER:1
  sub $64, %rsp
  mov %rsp, %rdi
  call f1
  #-64(%rbp) is the first tower
  #TOWER:2
  sub $64, %rsp
  mov %rsp, %rdi
  call f1
  #-128(%rbp) is the second tower
  #TOWER:3
  sub $64, %rsp
  mov %rsp, %rdi
  call f1
  #-192(%rbp) is the third tower.
 #initialize the rings in TOWER 1
  lea -64(%rbp), %rax
  mov (%rbp),%rcx
  #setting the size of the tower
  mov %rcx, (%rax)
  add $4, %rax

.init_s:
  mov %rcx, (%rax)
  add $4, %rax
  loop .init_s
#This copies the ring count into the first 4 bytes of the first tower, and then
for each 4 byte integer after that in descending order it stores the ring count
 #The leading value is the number of rings, and then each value to the left is
the width of the next ring.
```

```
  call pt7
#loop for all possible moves.r15 is the loop variable
#total no of rings in r14
#copying n to %cl
  mov (%rbp), %cl
#shift operand
  mov $1, %r14
#1<<n
  shl %cl, %r14
# decrease by 1
  dec %r14
#set loop variable to 0
  xor %r15,%r15

f7:
  lea -64(%rbp), %rdi     #TOWER :2
  lea -192(%rbp), %rsi    #TOWER: 3
  call f5                 #to move tower

  inc %r15                #loop++
  cmp %r14, %r15          #Compare to end loop
  jge f8                  #leave if done
#do similarly below for each
  lea -64(%rbp), %rdi     #TOWER :2
  lea -128(%rbp), %rsi    #TOWER :1
  call f5

  inc %r15
  cmp %r14, %r15
  jge f8

  lea -192(%rbp), %rdi    #TOWER :3
  lea -128(%rbp), %rsi    #TOWER :1
  call f5

  inc %r15
  cmp %r14, %r15
  jge f8
  jmp f7

f8:       #to leave when done
  lea 8(%rbp), %rsp
  ret

_start:#this is the starting of the code
  #the no of rings in %rdi
  movl $3, %edi
  cmpl $1, (%rsp)
  jle .solve
  mov 16(%rsp), %rax
  movsbl (%rax), %edi
  subl $'0', %edi
.solve:
  call solve

  mov $60, %rax
  xor %rdi, %rdi
  syscall
```