TOWER OF HANOI

In this problem there are n discks on the source pole by using auxillary pole
we should move all n disks to destination pole
mathematically by using recursion we can prove that steps taken to move n steps
take 2^n -1 steps

here in this solution the programmer used arrays and many functions to print each steps
using no of disks in each pole.this is an iterative approach.

he used source array,auxillary array and destination array

In these all arrays first index is used to store no dusks currently on the pole
and other indexs are in the order of disks on the pole

Here in code 2^n -1 value is stored in %r14.we keep 0 in r15 it is iterator upto r14

1)f1:

```
        xor %rax, %rax
        movl $64, %ecx
        rep stosb
        ret
```

f1 is used to make the arrays every element to 0. ie for all i ,array[i]=0
rcx is used to counter purpose
rep inst is used to repeat the operation as no of times in rcx
‘stosb’ will do movq %rax,(%rdx)

2) f2:

```
          movl (%rdi),%ecx
          cmpl $0, %ecx
          jz .peek_empty
          dec %ecx
          movq 4(%rdi, %rcx, 4), %rax
          ret
          .peek_empty:
          movl $100000, %eax
          ret
```

f2 returns the top disk number if pole is empty then a large number is returned
because in acc to rules of prob only small number is able to keep on a large numberd
disc so if we keep empty pole as large number it cannot be moved any where and it acts
as ground for all other discs

3)f3:

```
          movl (%rdi),%ebx
          dec %ebx
          movl 4(%rdi, %rbx, 4), %eax
          movl $0, 4(%rdi, %rbx, 4)
          mov %ebx, (%rdi)
          ret
```

it moves the returned value from r2 ie top disk number to rax register ie moving top disc of pole
it make top as 0 and decreses first element ie size by 1 as 1 disk is removed

4)f4:

```
            movl (%rdi),%ecx
            mov %rsi, 4(%rdi, %rcx, 4)
            inc %ecx
            mov %ecx, (%rdi)
            ret
```

rdi decides to which next pole that the disk should be moved.it take two arguments one is next pole base pointer and another as disk number we compare here results from f2

5).greater branch:

```
             mov %rdi, %rax
             mov %rsi, %rdi
             mov %rax, %rsi
```

the argument that should be paseed to f4. we will compaere if %rsi has large value then swap here

6)f5:

```
          mov %rdi, %r9
          call f2
          mov %rax, %r10
          mov %rsi, %rdi
          call f2
          mov %r9, %rdi
          cmp %rax, %r10
          jg .less_branch
```

In this function the discks are shifted from one pole to another pole.we create a temp variable for rdi it is r9 as we are going to modify this while calling f2 we store retunr value in r10 and store rsi to rdi because rdi is first arguments so store it in rdi and again call f2 and after returning the value of rdi is restored by r9 and comapre te returned value and r10 if less than r10 then swap rdi ,rsi for shifting otherwise continues and calls f3 after restoring rdi and rsi and calling f4 after this agin call f4 and restor value of rsi and rdi

7)Init_s:

```
           mov %rcx, (%rax)
           add $4, %rax
           loop .init_s
           call pt7

           mov (%rbp), %cl
           mov $1, %r14
           shl %cl, %r14
           dec %r14
           xor %r15,%r15
```

In this the first is initialized ie if no of discks are 3 array as {1,2,3} here we know hat rcx is counter as we are storing it as 1,2,3 ... n-1,n so we decrement rcx by 1 this is done by loop instruction.here index is in r15 and value 2^n-1 is in r14 this is like for loop from i=0;i<2^n-1

```
8)f7:
            lea -64(%rbp), %rdi
            lea -192(%rbp), %rsi
            call f5

            inc %r15
            cmp %r14, %r15
            jge f8

            lea -64(%rbp), %rdi
            lea -128(%rbp), %rsi
            call f5

            inc %r15
            cmp %r14, %r15
            jge f8

            lea -192(%rbp), %rdi
            lea -128(%rbp), %rsi
            call f5

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

this is loop function that discussed in before label it is main function ie all functions are sourced from here .we consider here sourece and destination arrays,source and auxillary and auxilarry and destination arrays respectively.first shifting happens first sift happen between first and third arrays they are storing base adress in rdi and rsi as it pases argument for f5 function after returning we increase r15 by 1 by the same way we do shift from 2nd and 3rd and between 1st and 3rd .if reached 2^n -1 we excute that cycle and stop

```
9)f8:
          lea 8(%rbp),%rsp
          ret
```

this is called when for loop is completed
this function frees memory we used .rsp is moved to rbp + 8 ie all values below are erased after it returns where solve is called and exit

MISTAKES IN CODE

line 113 and 131 have errors
in 113 we goto less_branch when leser condition is met but in code it is wriiten in such a way thst it goes when higher condition is met
in 131 we shiuld store rsp value to rbp as this can act as frame pointer for that block if not we cannot store where function starts