**INHA UNIVERSITY TASHKENT**

**DEPARTMENT OF CSE & ICE**

**SPRING SEMESTER 2019**

**SOC 2040 - SYSTEMS PROGRAMMING**

**LAB ASSIGNMENT 3**

**Submitted by**

**Student Name Student ID**

**Group : Sophomore**



**INSTRUCTIONS :**

**- ALL LAB ASSIGNMENTS ARE TO BE COMPLETED IN GROUPS OF MAXIMUM 6 STUDENTS.**

**- LAB ASSIGNMENT REPORT SHOULD BE PREPARED USING THE LAB ASSIGNMENT 3 REPORT TEMPLATE PROVIDED.**

**- ONE HARD COPY OF THE LAB ASSIGNMENT REPORT OF EACH GROUP SHOULD BE HANDED IN AT THE OFFICE BY THE GROUP LEADER.**

**- EVERY MEMBER OF THE TEAM MUST UPLOAD THE SOFTCOPY OF THE REPORT AT THE E- CLASS PORTAL.**

**- FOR ALL PART-A ASSEMBLY LANGUAGE PROGRAMMING PRACTICE QUESTIONS, YOU NEED TO PROVIDE PROGRAM AND RESULTS SCREENSHOTS.**

**- FOR ALL PART-B ASSEMBLY LANGUAGE PROGRAMMING QUESTIONS, YOU NEED TO PROVIDE ALGORITHM, PROGRAM AND RESULTS & PROGRAM SCREENSHOTS.**

**- FOR ALL PART-C QUESTIONS, YOU NEED TO PROVIDE DDD DEBUGGER SCREENSHOTS AFTER EXECUTION OF EACH INSTRUCTION IN TRACE MODE BY SHOWING THE INVOLVED REGISTER CONTENTS, MEMORY CONTENTS AND STACK CONTENTS IN CASE OF STACK OPERATIONS.**

**- ALL FILES CONTAINING ASSEMBLY LANGUAGE PROGRAMS WRITTEN AND EXECUTED FOR ALL THE PART-A, PART-B AND PART-C QUESTIONS SHOULD BE UPLOADED AT THE E-CLASS PORTAL ALONG WITH THE LAB ASSIGNMENT REPORT.**

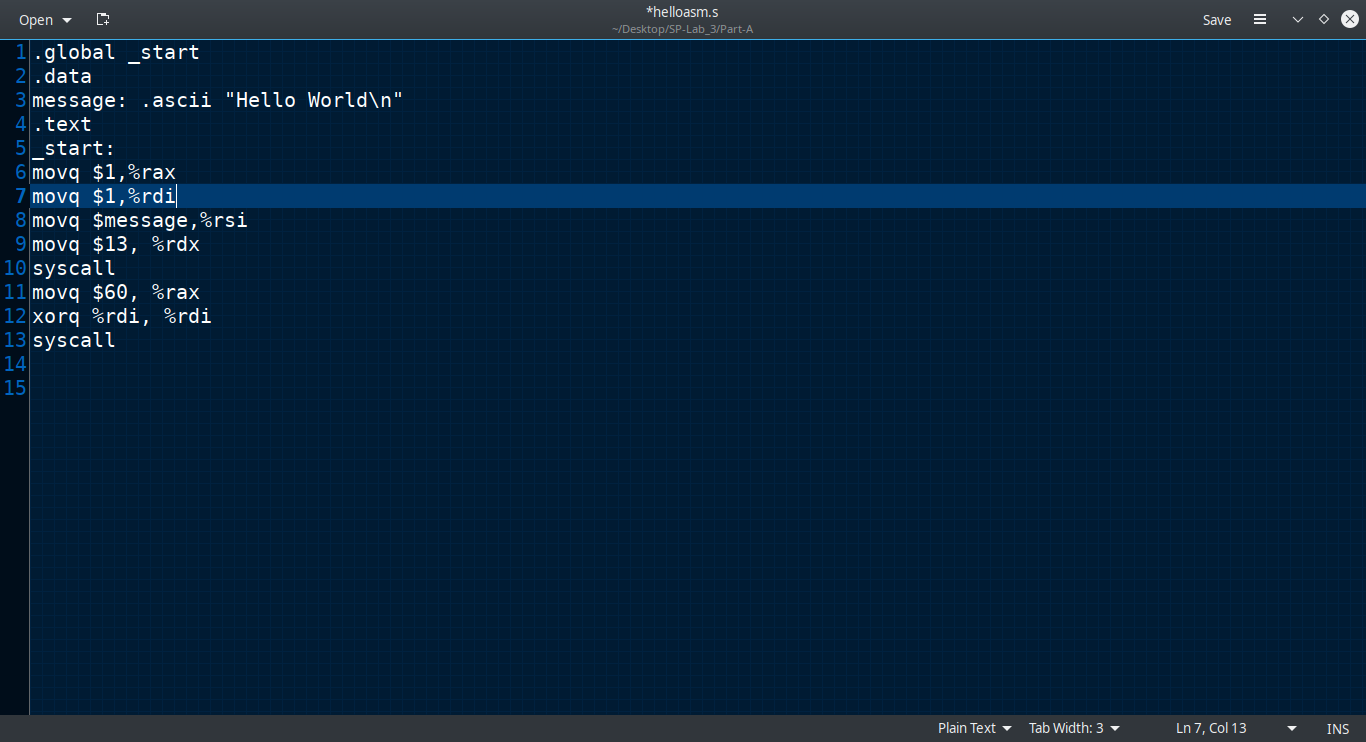
**- LAST DATE FOR SUBMISSION OF THE LAB ASSIGNMENT REPORT IS 15th MARCH 2019**

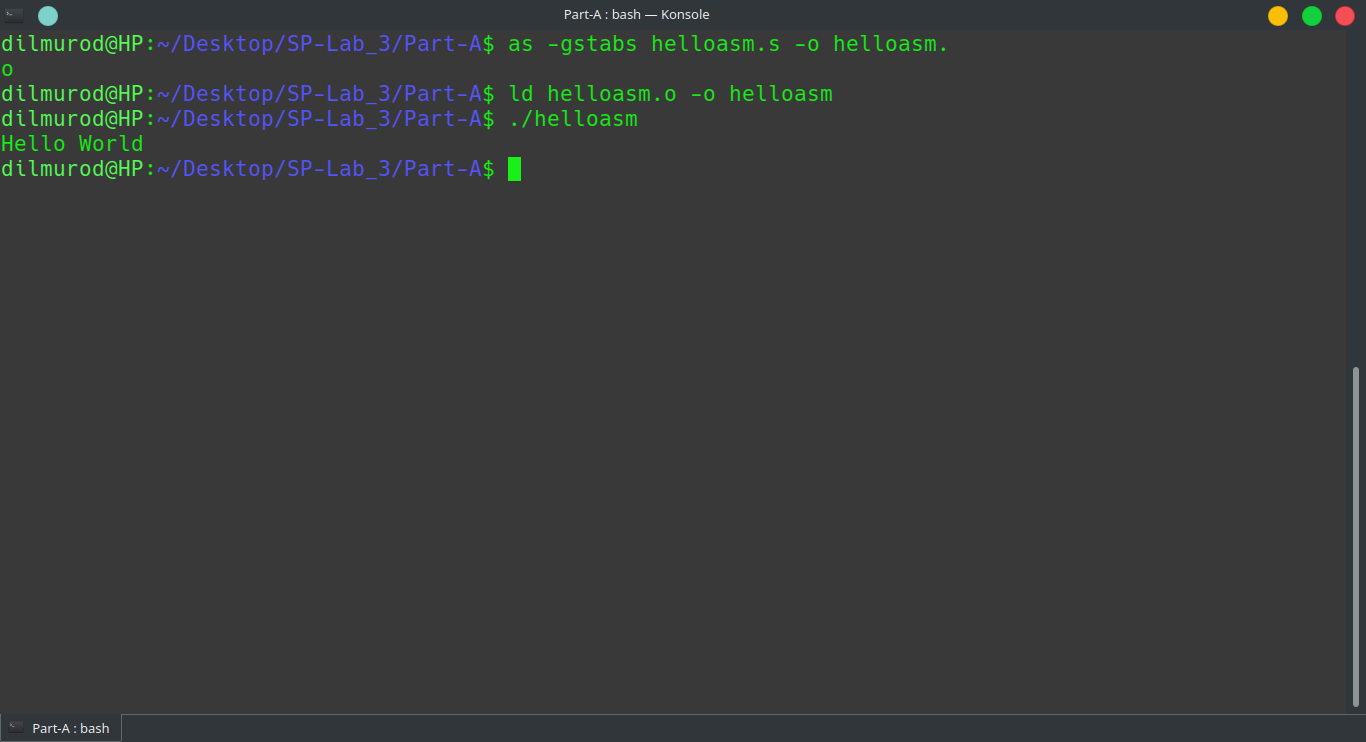
**- LATE SUBMISSIONS ARE NOT ENTERTAINED, ADHERE TO THE DEADLINE STRICTLY**

**- READ THE QUESTIONS CORRECTLY & CAREFULLY**

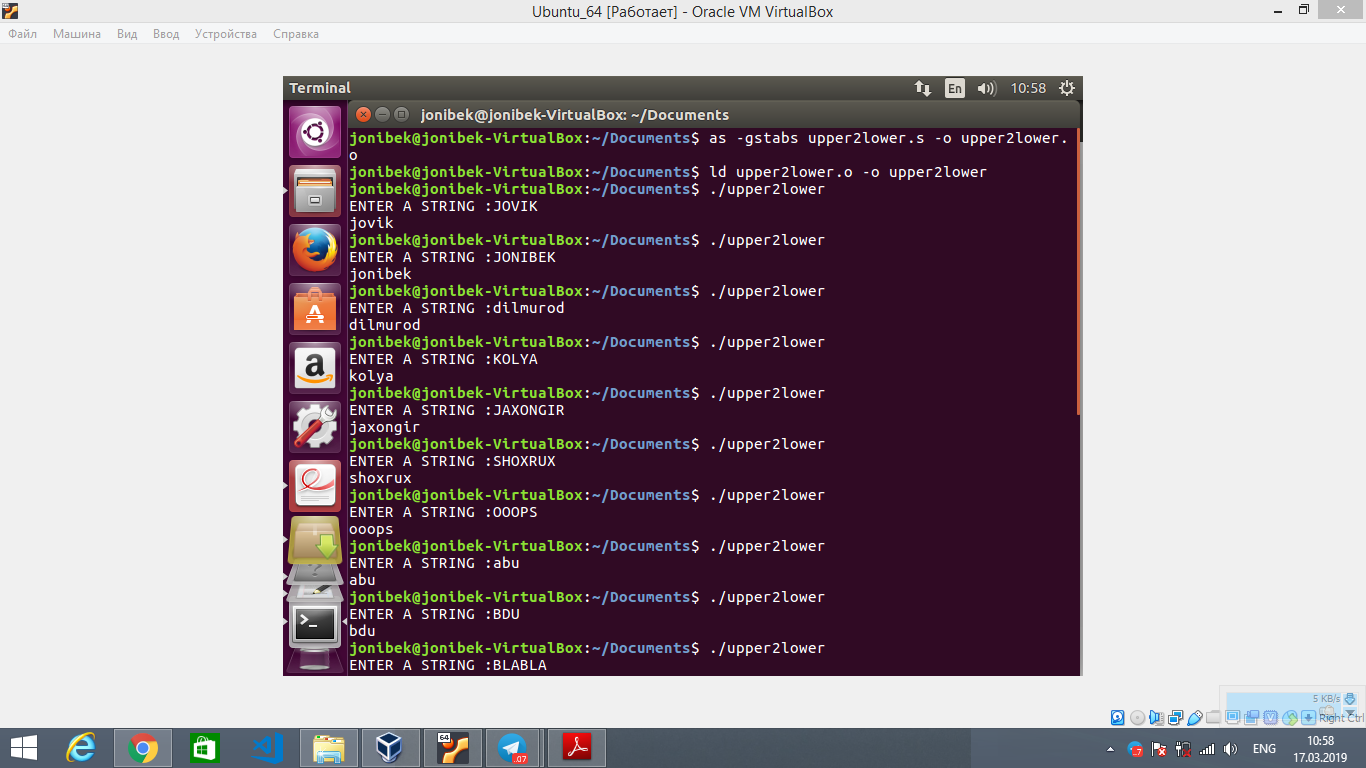
Part-A

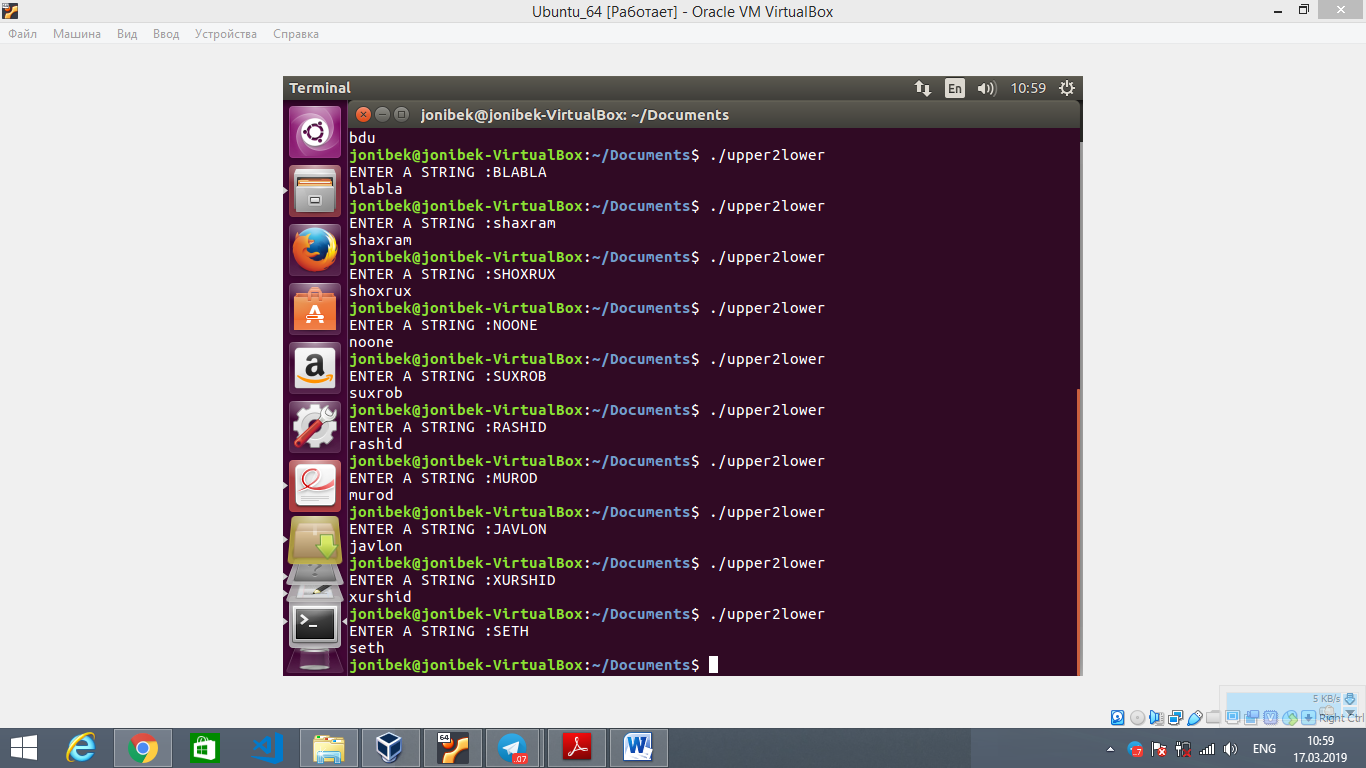
1.



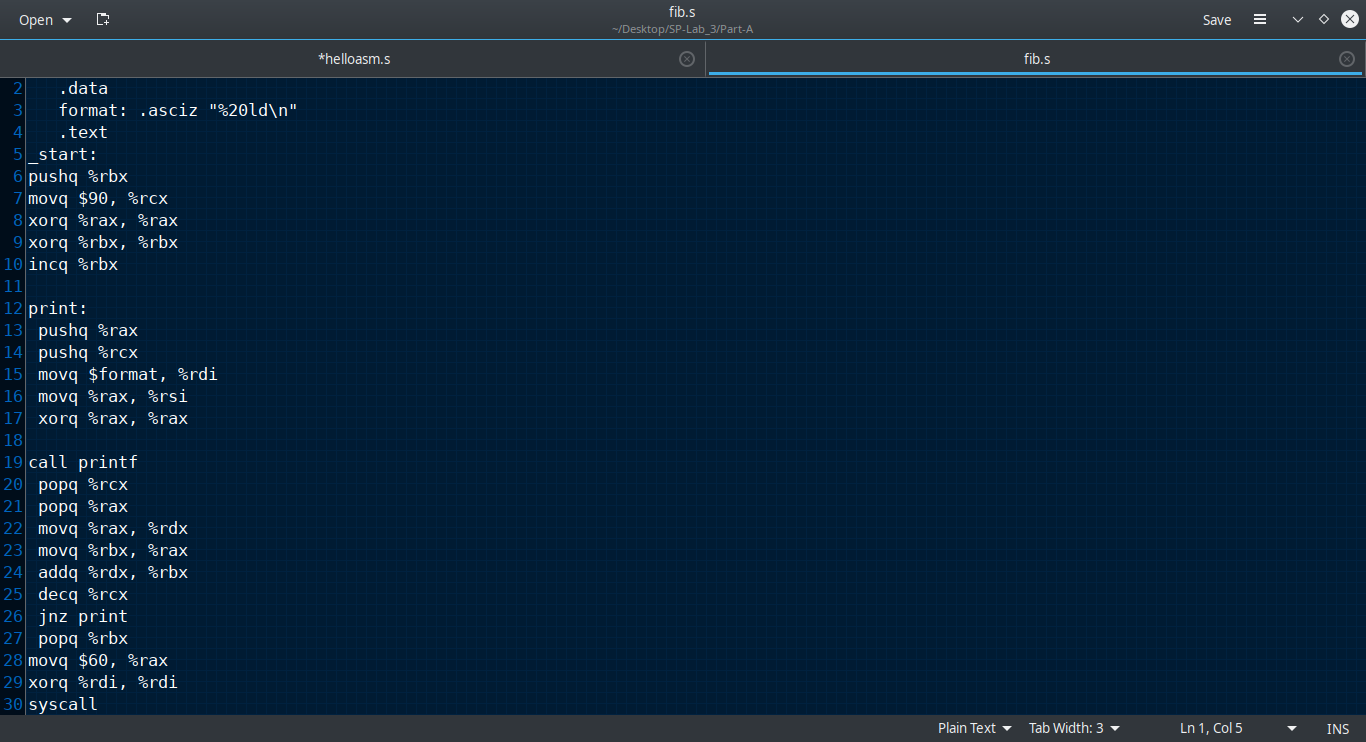


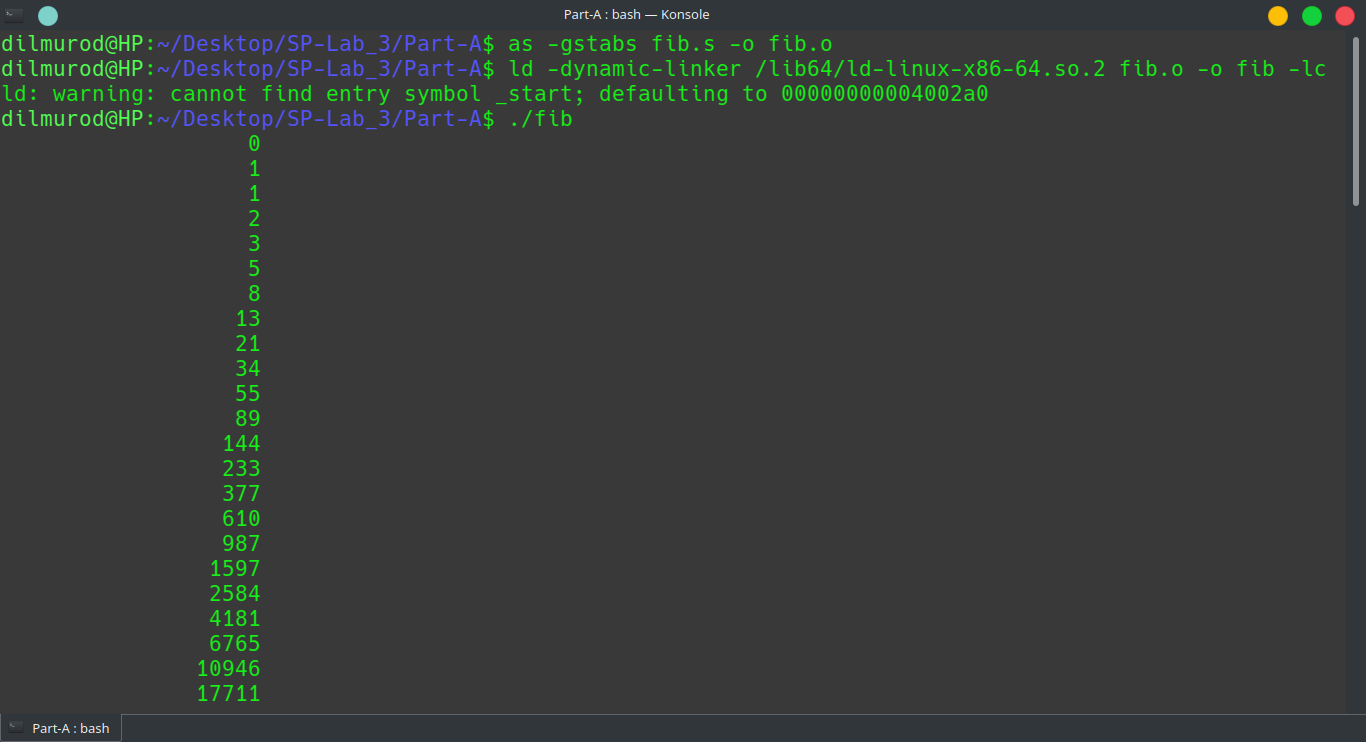
5.

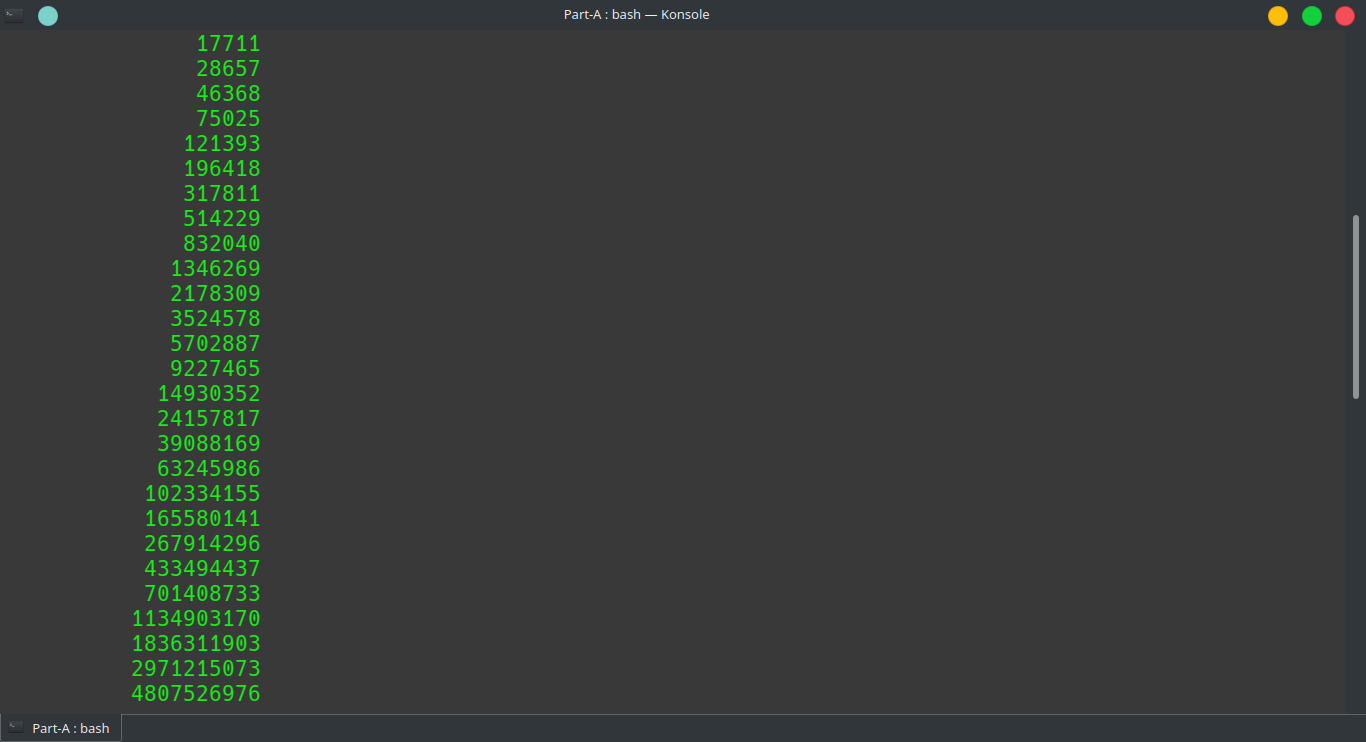


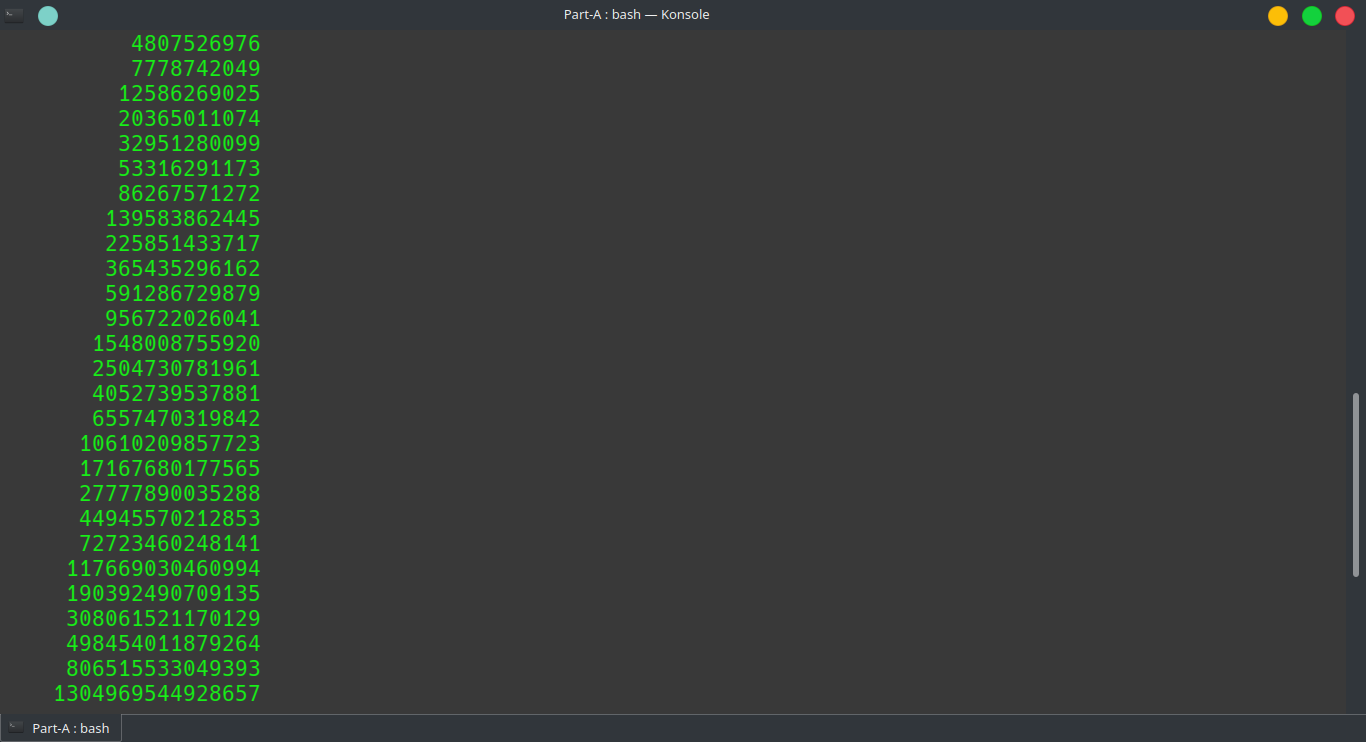


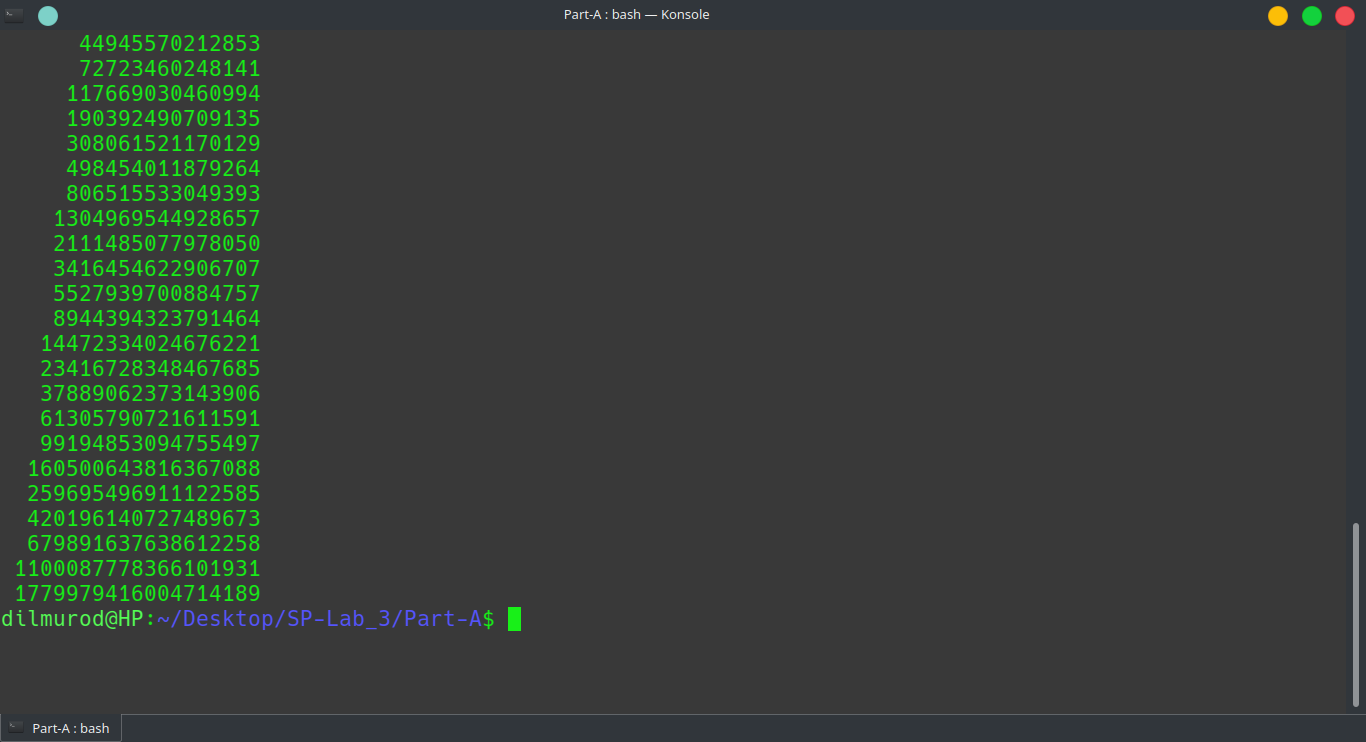
6.











Task13

Write an X86-64 assembly language program to read encrypted

message generated in Q12) from the keyboard and perform the following:

DECRYPT the message by subtracting 9 from every character in the

message and store this decrypted message in a separate memory area.

.global \_start

.data

buf: .skip 1024

message: .asciz "Enter a string: "

givenstr: .skip 100

nxtchars: .space 100

.text

\_start:

# To output message

movq $1,%rax

movq $1, %rdi

movq $message, %rsi

movq $16, %rdx

syscall

# To output message

movq $0, %rax

movq $0, %rdi

movq $givenstr, %rsi

movq $100, %rdx

syscall

movq %rax, %r8

movq %r8, %rcx

decq %rcx

movq $givenstr, %rsi

movq $nxtchars, %rdi

up:

movb (%rsi), %al

subb $9, %al

down:

movb %al, (%rdi)

incq %rsi

incq %rdi

decq %rcx

jnz up

movb (%rsi), %al

movb %al, (%rdi)

movq $1,%rax

movq $1, %rdi

movq $nxtchars, %rsi

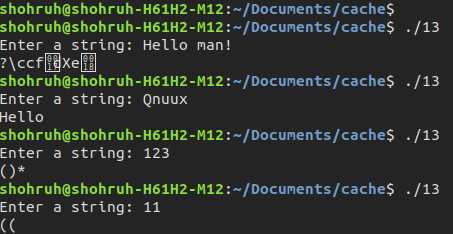
movq $100, %rdx

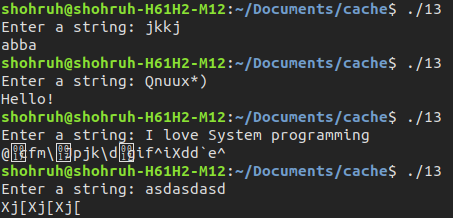
syscall

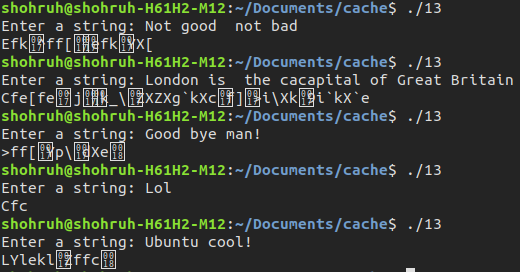
movq $60, %rax

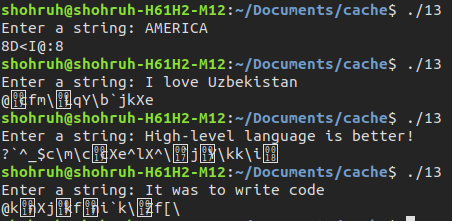
xorq %rdi, %rdi

syscall









Task15

Write an X86-64 assembly language program to read a string of characters of any length (maximum 1024 characters) from the keyboard and count the number of alphabets, number of numerals, number of special characters (all characters including space - other than alphabets and numerals), total number of characters and total number of words in the string.

.global \_start

.data

str: .skip 1024

alphabet: .int 0

special: .int 0

numbers: .int 0

spaces: .int 0

all: .int 0

message1: .string "Enter String: "

message2: .string "\nNumber of alphabets: %d"

message3: .string "\nNumber of numerals: %d"

message4: .string "\nNumber of special characters: %d"

message5: .string "\nTotal number of characters in string: %d"

message6: .string "\nTotal number of words in string: %d\n"

.text

\_start:

movq $1, %rdi

movq $1, %rax

movq $message1, %rsi

movq $14, %rdx

syscall

push %rbx

#getting input from user

movq $0, %rax

movq $0, %rdi

movq $str, %rsi

movq $1024, %rdx

syscall

movq %rax, all

movq $0, %r8 #indexer

add $str, %r8 #pointing to base

while:

movb (%r8), %cl

cmp $32, %cl

je spc

#if it is >= 48 then

cmp $48, %cl

jge first

#else it is special character

incq special

done:

incq %r8

movb (%r8), %bl

cmp $10, %bl

jne while

jmp while\_end

movq $1, %rax

movq $str, %rsi

syscall

first:

cmp $65, %cl #if it is >= 65 then

jge step\_2

cmp $57, %cl #else if it is <= 57 this is number

jle number

#else it is special character

incq special

jmp done

step\_2:

cmp $97, %cl #if it is >= 97

jge nextt

#else if it is <= 90 upppercase aplphabet

cmpb $90, %cl

jle alpha

#else it is special character

incq special

jmp done

nextt:

cmp $122, %cl #if it is <= 122 it is alphabet

jle alpha

#else it is special character

incq special

jmp done

spc:

incq spaces

incq special

jmp done

number:

incq numbers

jmp done

alpha:

incq alphabet

jmp done

while\_end:

movq $message2, %rdi

movq alphabet, %rsi

xorq %rax, %rax

call printf

movq $message3, %rdi

movq numbers, %rsi

xorq %rax, %rax

call printf

movq $message4, %rdi

movq special, %rsi

xorq %rax, %rax

call printf

decq all

movq $message5, %rdi

movq all, %rsi

xorq %rax, %rax

call printf

incq spaces

movq $message6, %rdi

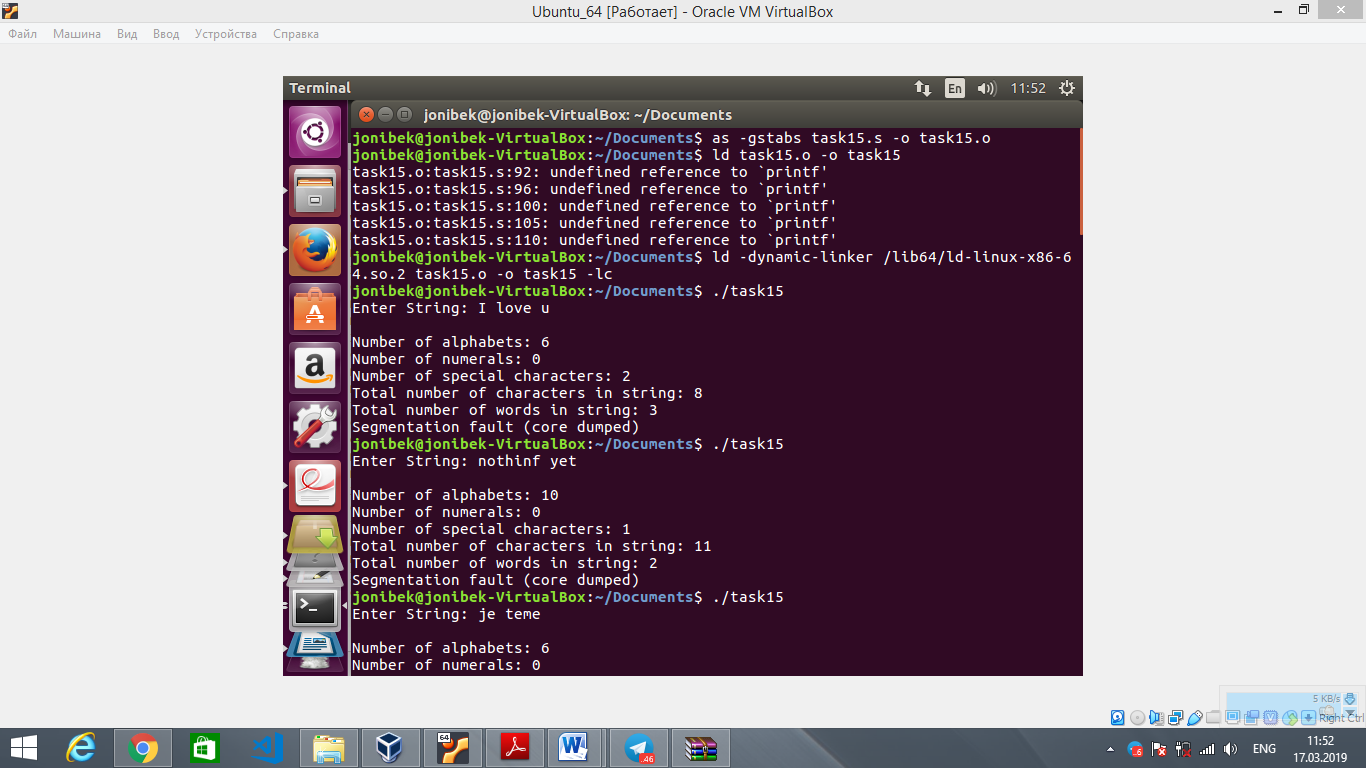
movq spaces, %rsi

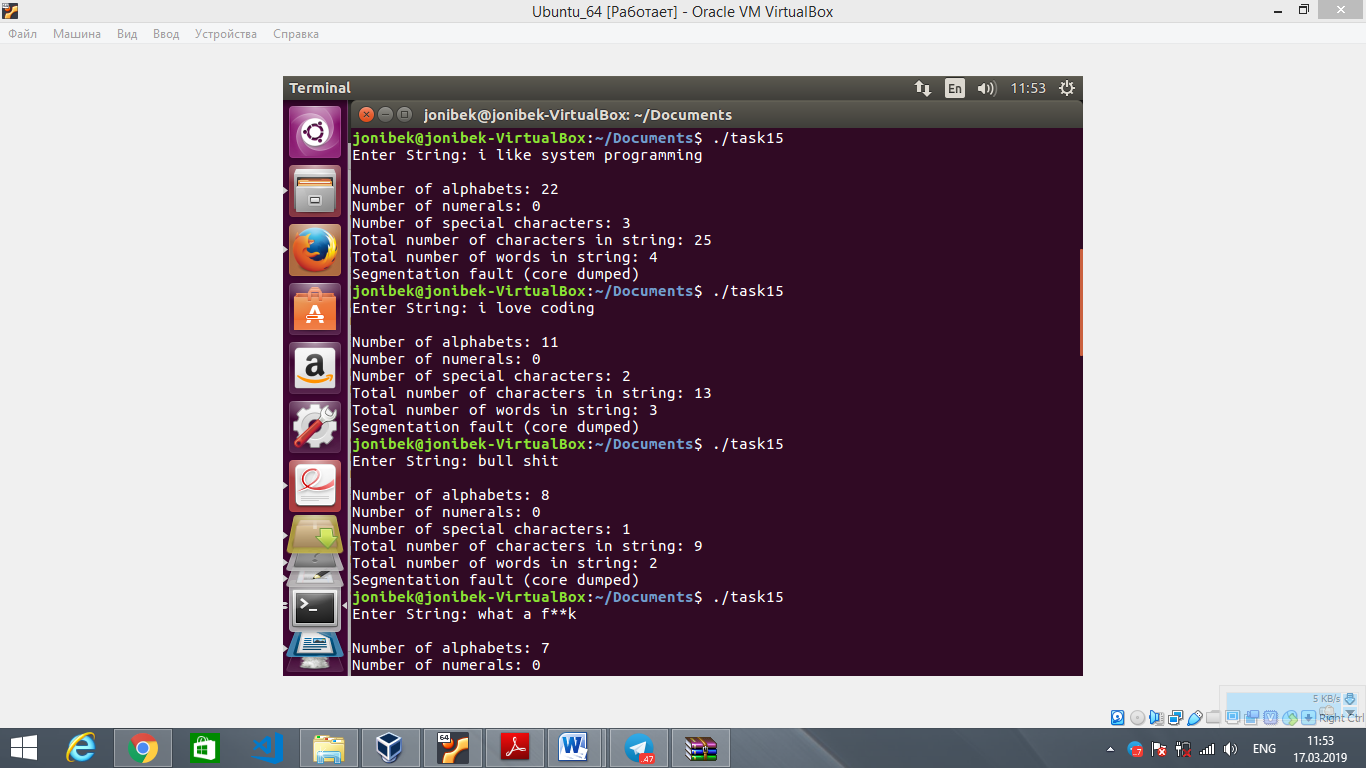
xorq %rax, %rax

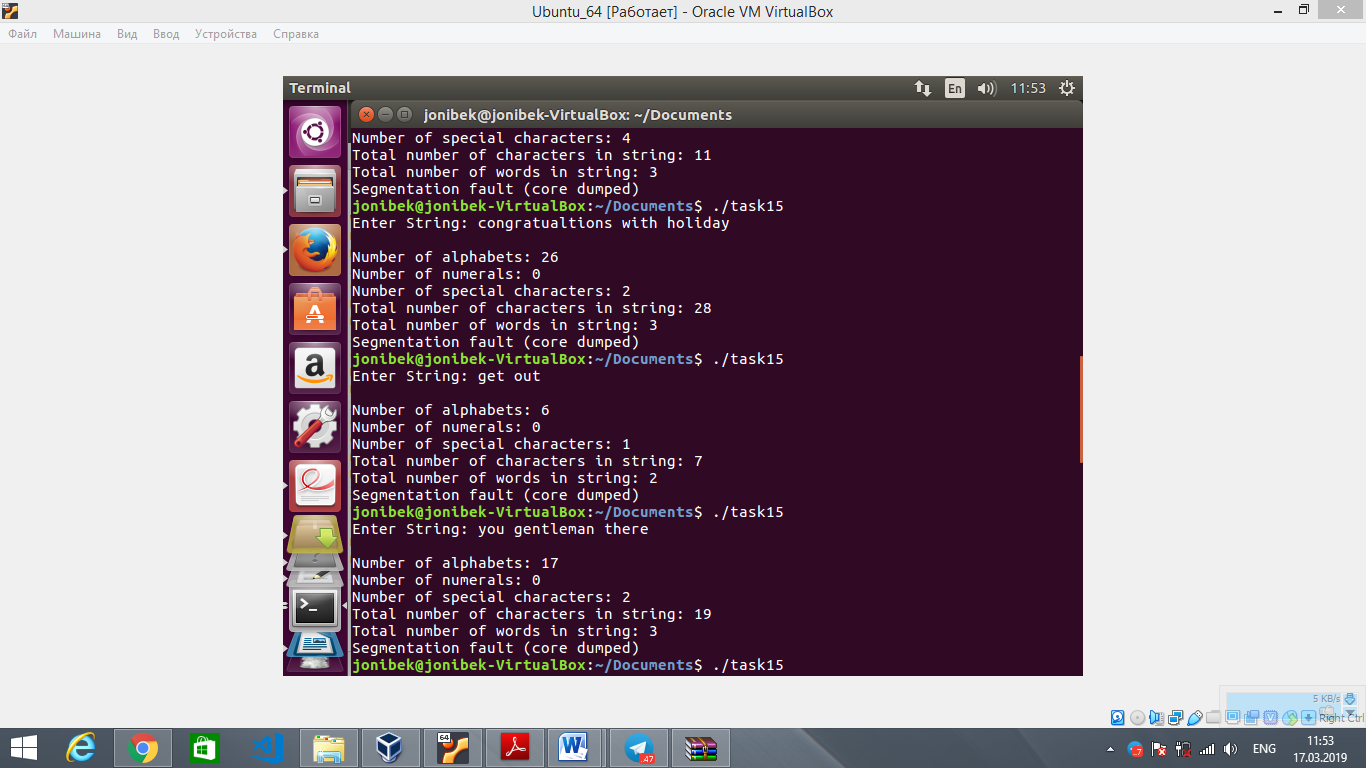
call printf

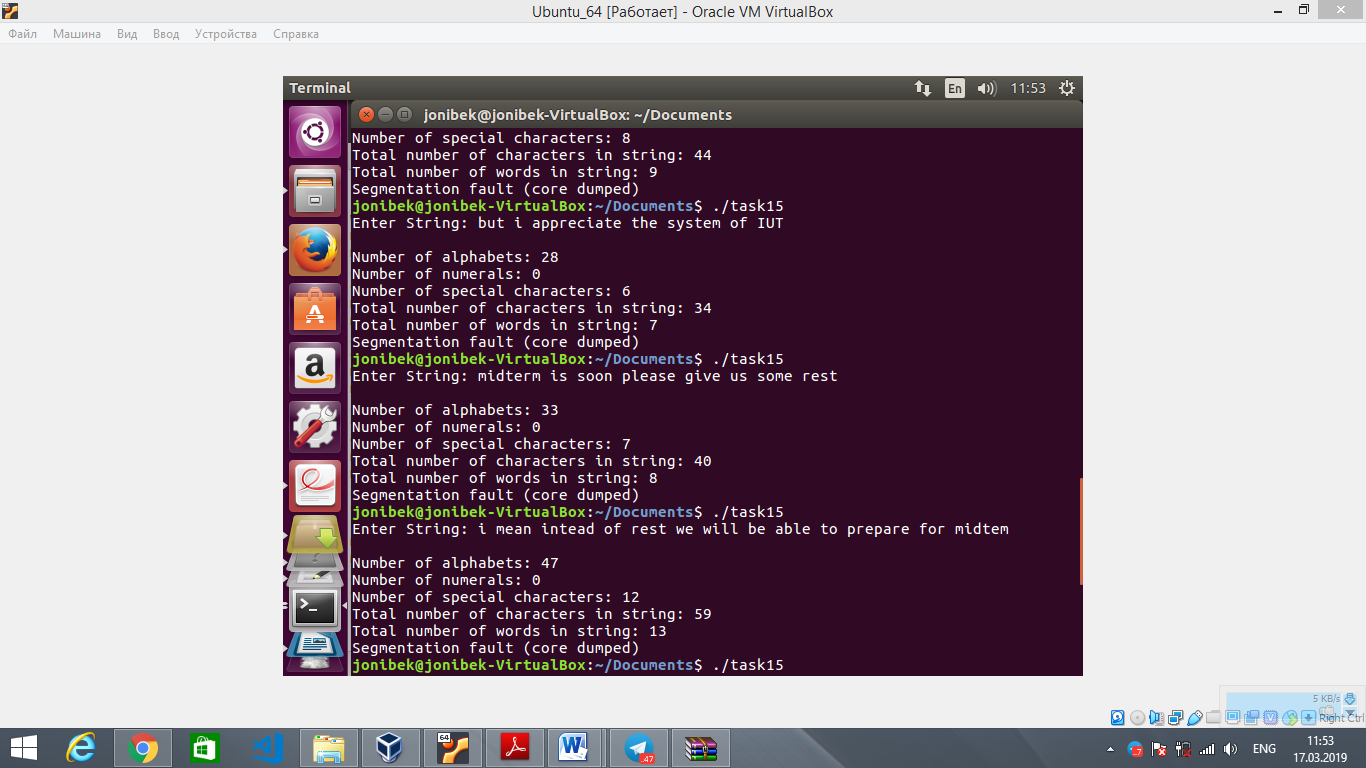
pop %rbx

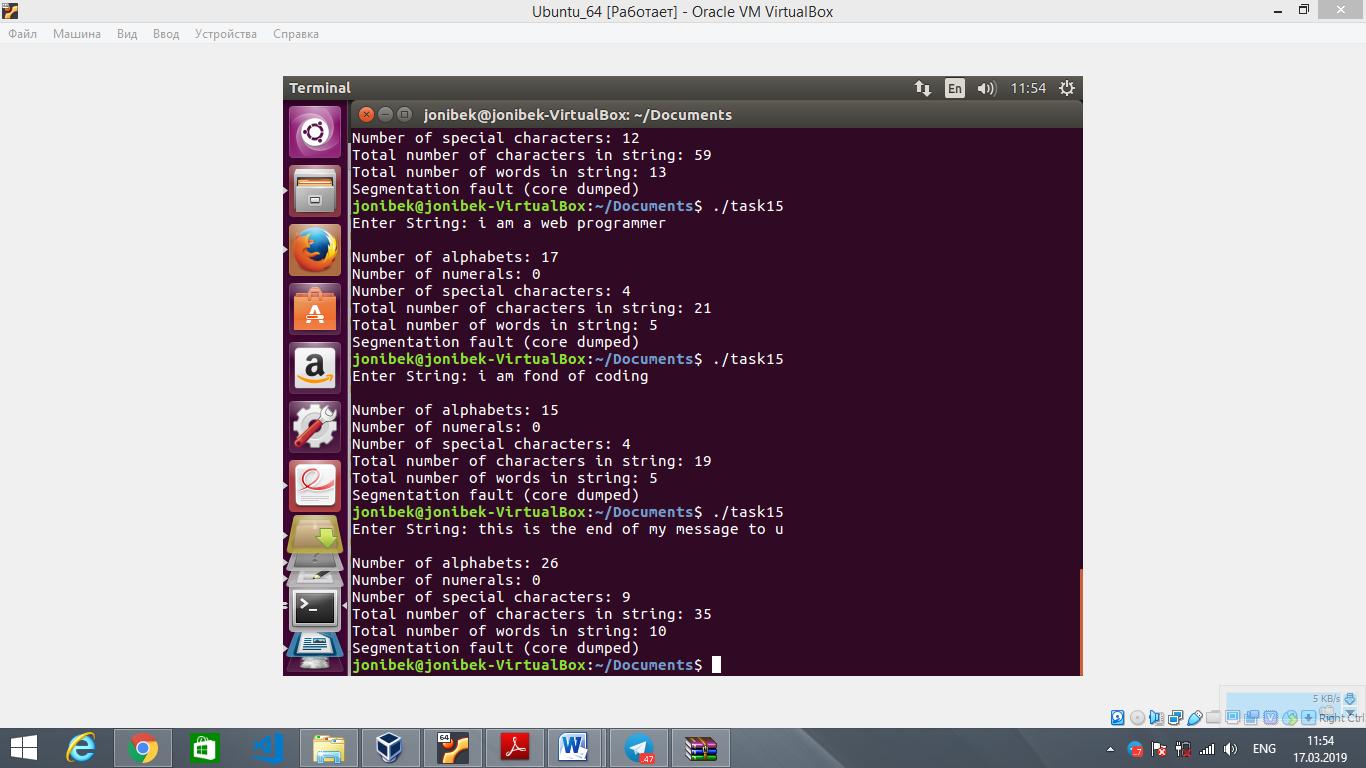
ret











Task18

Write an X86-64 assembly language program to generate all prime num-

bers between 1 and n. The input ‘n’ should be read from the keyboard and

the output should be printed on the screen.

.global \_start

.data

format: .asciz ", %d"

x: .quad 0

message: .asciz "Enter value for n: "

message2: .asciz "Prime numbers between 1 and %d are : 1"

message3: .asciz "\n"

f: .asciz "%d"

.text

isprime: # declaring isprime function

movq %rdi, %r8 #passing the first argument to r8

movq $1, %r13

nextstage:

incq %r13

cmpq %r8, %r13

je prime

movq %rdi, %rax

cqto

idivq %r13 #diving each number until the given to check it is prime or not

movq %rdx, %rcx #move the remainder to rcx

jrcxz notprime #if the remainder is zero, then the number is not prime

jmp nextstage

notprime:

movq $1, %rax

ret

prime:

movq $0, %rax

ret

\_start:

movq $message, %rdi

call puts #displaying a message

scan:

movq $0, %rax

movq $f, %rdi

movq $x, %rsi

call scanf #getting an input

movq x, %r12

cmpq $1, %r12

je end

movq $1, %r8 #checking each number up until the given

pushq %r8

pushq %r12

movq $message2, %rdi

movq %r12, %rsi

xorq %rax, %rax

call printf

xorq %rdi, %rdi

call fflush

popq %r12

popq %r8

l0:

incq %r8 #incrementing the changing number that we check

cmpq %r8, %r12

je end

movq %r8, %rdi

call isprime #check whether it is prime or not

movq %rax, %rcx

jrcxz l1

jmp l0

l1:

jrcxz print

jmp l0

print: #print out the prime number

pushq %rax

pushq %r8

movq $format, %rdi

movq %r8, %rsi

xorq %rax, %rax

call printf

xorq %rdi, %rdi

call fflush

popq %r8

popq %rax

movq $1, %rcx

jmp l0

end:

movq $message3, %rdi

call puts

movq $60, %rax

xorq %rdi, %rdi

syscall #exit the program

