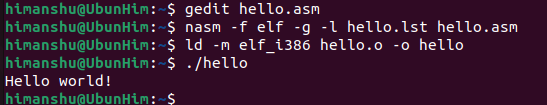
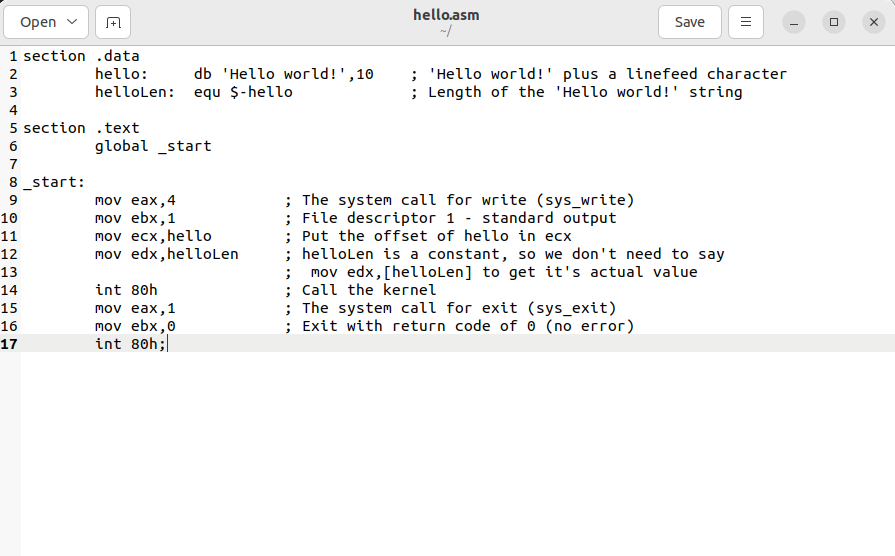
1. To Install NASM:



2) To print Hello World





1. Take input from user as name, roll number and address:

section .bss

name resb 50 ; buffer for name (50 bytes)

roll resb 10 ; buffer for roll number (10 bytes)

address resb 100 ; buffer for address (100 bytes)

section .data

prompt\_name db "Enter your name: ",0

len\_name equ $-prompt\_name

prompt\_roll db "Enter your roll no: ",0

len\_roll equ $-prompt\_roll

prompt\_addr db "Enter your address: ",0

len\_addr equ $-prompt\_addr

newline db 0xA,0

section .text

global \_start

\_start:

; --- Name input ---

mov rax, 1 ; sys\_write

mov rdi, 1 ; stdout

mov rsi, prompt\_name

mov rdx, len\_name

syscall

mov rax, 0 ; sys\_read

mov rdi, 0 ; stdin

mov rsi, name

mov rdx, 50 ; max bytes

syscall

; --- Roll number input ---

mov rax, 1

mov rdi, 1

mov rsi, prompt\_roll

mov rdx, len\_roll

syscall

mov rax, 0

mov rdi, 0

mov rsi, roll

mov rdx, 10

syscall

; --- Address input ---

mov rax, 1

mov rdi, 1

mov rsi, prompt\_addr

mov rdx, len\_addr

syscall

mov rax, 0

mov rdi, 0

mov rsi, address

mov rdx, 100

syscall

; --- Print newline ---

mov rax, 1

mov rdi, 1

mov rsi, newline

mov rdx, 1

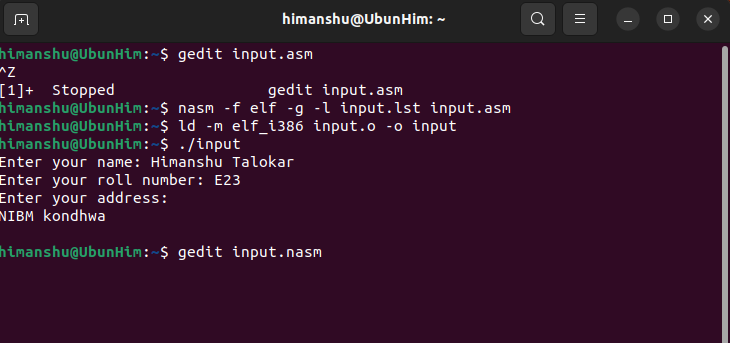
syscall

; --- Exit ---

mov rax, 60 ; sys\_exit

xor rdi, rdi

syscall



1. Use of macro:

section .data

msg db "The result is: ",0

msg\_len equ $-msg

newline db 0xA,0

section .text

global \_start

; --- Define macro to add two numbers ---

%macro ADD\_NUMS 2

mov rax, %1

add rax, %2

%endmacro

\_start:

; Use macro to add 5 + 7

ADD\_NUMS 5, 7 ; result stored in RAX

; Convert result (12) to ASCII '12'

mov rbx, 10 ; divisor for decimal conversion

xor rcx, rcx ; counter/index

mov rdx, 0

div rbx ; divide RAX by 10

add dl, '0' ; remainder = second digit

add al, '0' ; quotient = first digit

mov [num+0], al

mov [num+1], dl

mov byte [num+2], 0

; Print message

mov rax, 1 ; sys\_write

mov rdi, 1 ; stdout

mov rsi, msg

mov rdx, msg\_len

syscall

; Print number

mov rax, 1

mov rdi, 1

mov rsi, num

mov rdx, 2

syscall

; Print newline

mov rax, 1

mov rdi, 1

mov rsi, newline

mov rdx, 1

syscall

; Exit

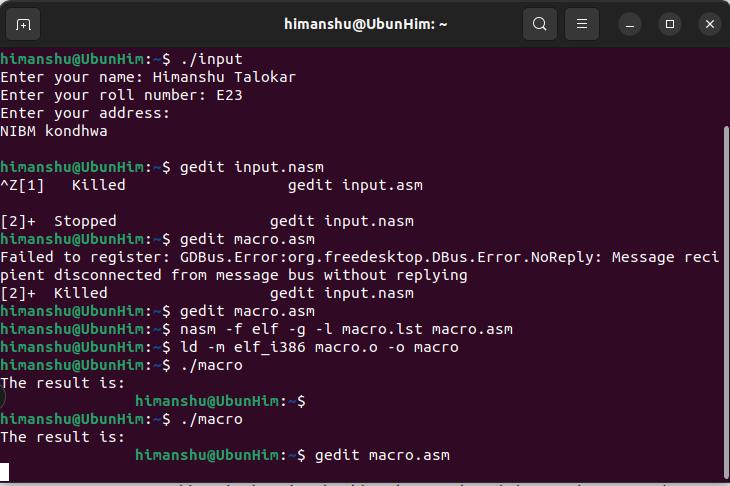
mov rax, 60 ; sys\_exit

xor rdi, rdi

syscall

section .bss

num resb 3 ; buffer to store two digits + null



1. Operations (addition, subtraction, multiplication, division) no input taken:

section .data

newline db 0xA

section .bss

res resb 3

section .text

global \_start

\_start:

mov al, 12

mov bl, 5

add al, bl

call print

mov al, 12

mov bl, 5

sub al, bl

call print

mov al, 12

mov bl, 5

mul bl

call print

mov al, 12

mov bl, 5

xor ah, ah

div bl

call print

mov eax, 1

xor ebx, ebx

int 0x80

print:

add al, '0'

mov [res], al

mov eax, 4

mov ebx, 1

mov ecx, res

mov edx, 1

int 0x80

mov eax, 4

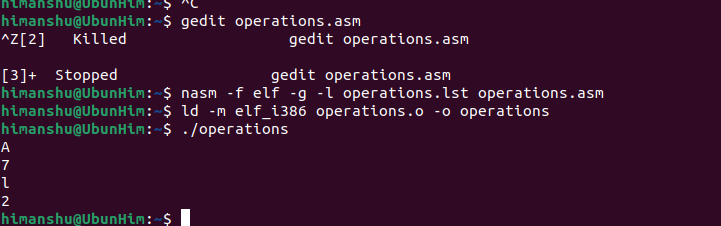
mov ebx, 1

mov ecx, newline

mov edx, 1

int 0x80

ret



1. Operations while taking input

section .data

msg1 db "Enter first digit: ", 0xA

len1 equ $-msg1

msg2 db "Enter second digit: ", 0xA

len2 equ $-msg2

msg\_sum db "The sum is: ", 0xA

len\_sum equ $-msg\_sum

msg\_sub db "The difference is: ", 0xA

len\_sub equ $-msg\_sub

msg\_mul db "The product is: ", 0xA

len\_mul equ $-msg\_mul

msg\_div db "The quotient is: ", 0xA

len\_div equ $-msg\_div

section .bss

num1 resb 2

num2 resb 2

res resb 2

section .text

global \_start

\_start:

mov eax, 4

mov ebx, 1

mov ecx, msg1

mov edx, len1

int 0x80

mov eax, 3

mov ebx, 0

mov ecx, num1

mov edx, 2

int 0x80

mov eax, 4

mov ebx, 1

mov ecx, msg2

mov edx, len2

int 0x80

mov eax, 3

mov ebx, 0

mov ecx, num2

mov edx, 2

int 0x80

mov al, [num1]

sub al, '0'

mov bl, [num2]

sub bl, '0'

mov ah, 0

add al, bl

add al, '0'

mov [res], al

mov eax, 4

mov ebx, 1

mov ecx, msg\_sum

mov edx, len\_sum

int 0x80

mov eax, 4

mov ebx, 1

mov ecx, res

mov edx, 1

int 0x80

mov al, [num1]

sub al, '0'

mov bl, [num2]

sub bl, '0'

sub al, bl

add al, '0'

mov [res], al

mov eax, 4

mov ebx, 1

mov ecx, msg\_sub

mov edx, len\_sub

int 0x80

mov eax, 4

mov ebx, 1

mov ecx, res

mov edx, 1

int 0x80

mov al, [num1]

sub al, '0'

mov bl, [num2]

sub bl, '0'

mul bl ; AL \* BL → AX

add al, '0' ; convert result to ASCII

mov [res], al

mov eax, 4

mov ebx, 1

mov ecx, msg\_mul

mov edx, len\_mul

int 0x80

mov eax, 4

mov ebx, 1

mov ecx, res

mov edx, 1

int 0x80

mov al, [num1]

sub al, '0'

mov bl, [num2]

sub bl, '0'

xor ah, ah

div bl

add al, '0'

mov [res], al

mov eax, 4

mov ebx, 1

mov ecx, msg\_div

mov edx, len\_div

int 0x80

mov eax, 4

mov ebx, 1

mov ecx, res

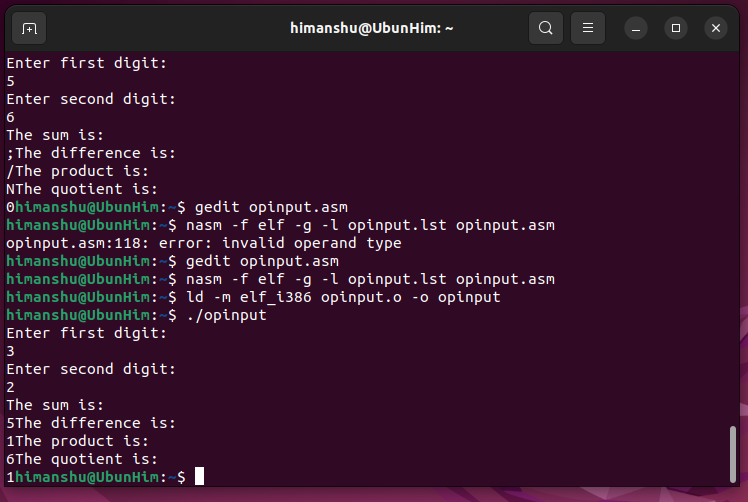
mov edx, 1

int 0x80

mov eax, 1

mov ebx, 0

int 0x80



GDB(GNU debugger):

