PAI Assignment

Name : Sahil Ghule   
Enrollment No: ADT24SOCB0974   
Roll No: 44  
Class : SY-06   
  
Assignment No-01

A. Hello world

B. GDB execution

#Code-

global \_start

section .data

hello db "Hello, World", 10

length equ $ - hello

section .text

\_start:

mov eax, 4 ; write to file

mov ebx, 1 ; STDOUT handle

mov ecx, hello ; our message

mov edx, length ; size of our message

int 0x80 ; execute the syscall

xor ebx, ebx ; send 0 as 'exit code'

mov eax,1 ; terminate process

int 0x80 ; execute the syscall

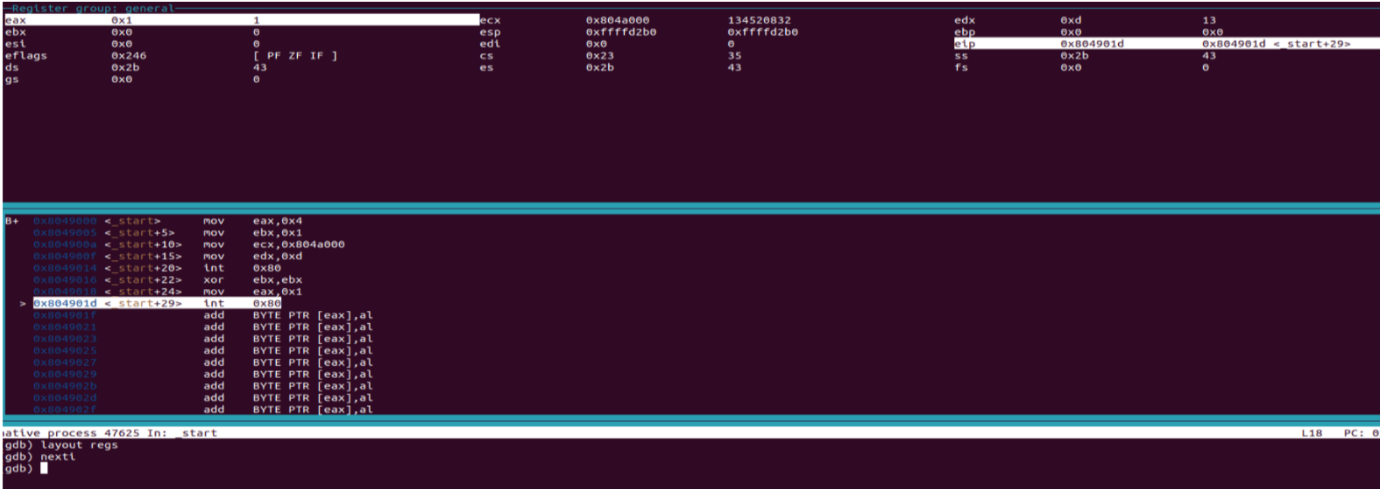
#Output-

A screenshot of a computer program

AI-generated content may be incorrect.

# Debugging-

(GDB) Hello World



Assignment No-02

A. Name And Surname Input

B. GDB execution

#Code-

section .data

name db "Sahil ",10

len\_name equ $ - name

surname db "Ghule",10

len\_surname equ $ - surname

section .text

global \_start

\_start:

mov eax, 4

mov ebx, 1

mov ecx, name

mov edx, len\_name

int 0x80

mov eax, 4

mov ebx, 1

mov ecx, surname

mov edx, len\_surname

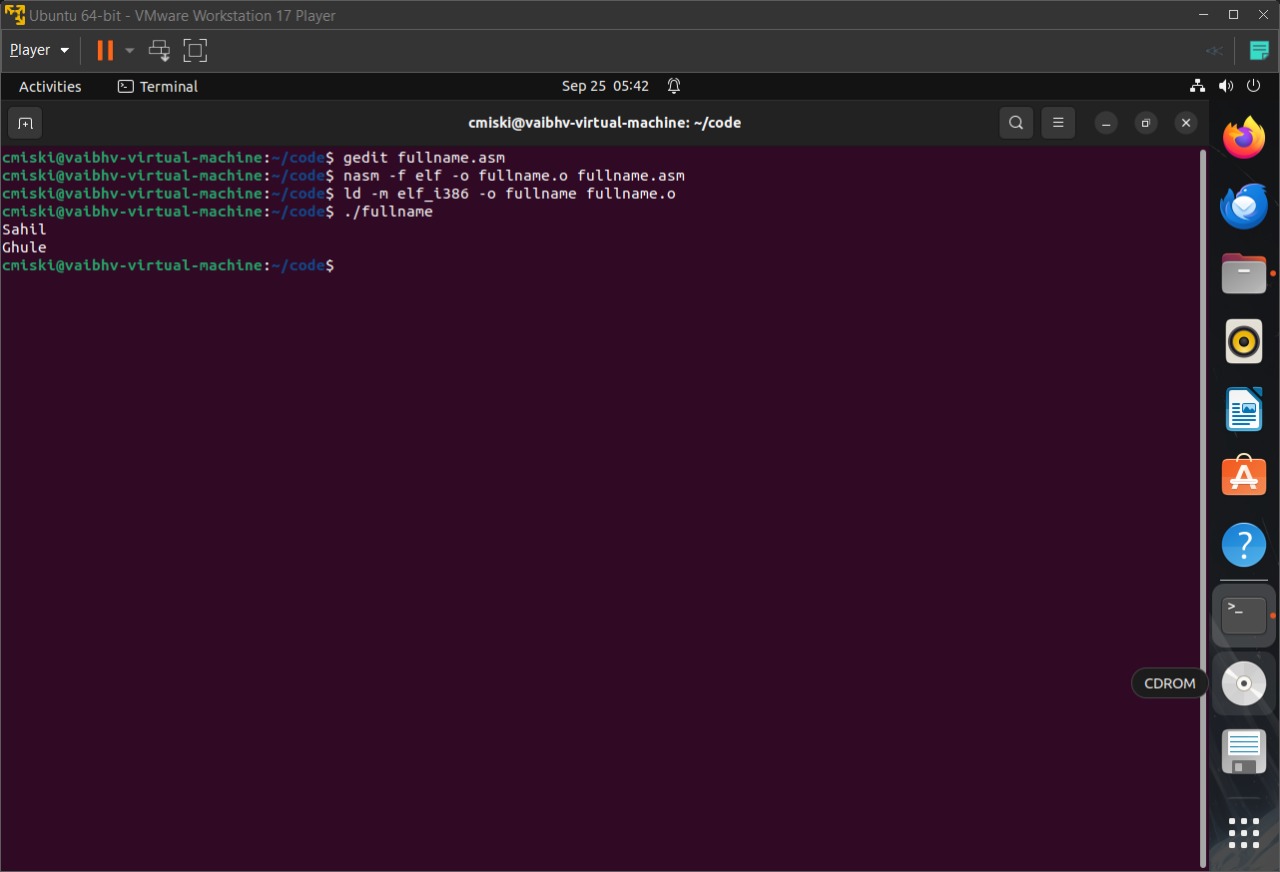
int 0x80

mov eax, 1

xor ebx, ebx

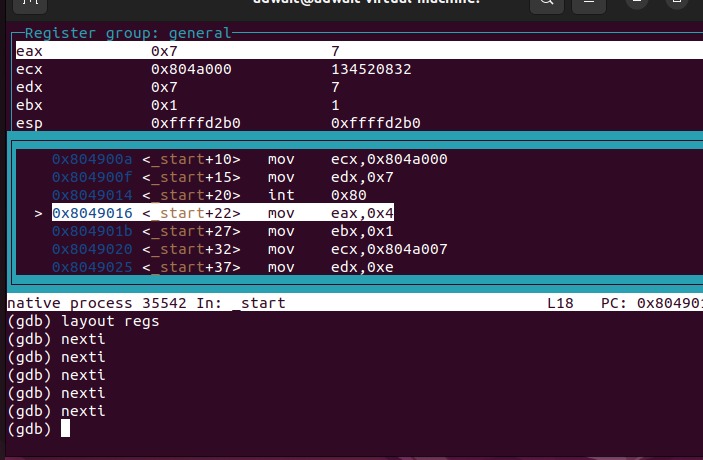
int 0x80

#Output



# Debugging-

(GDB) Name And Surname



Assignment No-03

1. Addition, Subtraction, Multiplication, Division (Terminal Output)
2. Addition, Subtraction, Multiplication, Division

(GDB execution)

A) Terminal Output

1. Addition Code-

section .data

msg1 db "Enter first number: ",0

len1 equ $-msg1

msg2 db "Enter second number: ",0

len2 equ $-msg2

resultMsg db "Result = ",0

lenRes equ $-resultMsg

newline db 10

section .bss

num1 resb 10

num2 resb 10

res resb 10

section .text

global \_start

\_start:

; Ask for first number

mov eax, 4

mov ebx, 1

mov ecx, msg1

mov edx, len1

int 0x80

; Read input (up to 10 chars)

mov eax, 3

mov ebx, 0

mov ecx, num1

mov edx, 10

int 0x80

; Convert num1 -> integer

mov esi, num1

call atoi

mov ebx, eax ; store first number in ebx

; Ask for second number

mov eax, 4

mov ebx, 1

mov ecx, msg2

mov edx, len2

int 0x80

; Read input

mov eax, 3

mov ebx, 0

mov ecx, num2

mov edx, 10

int 0x80

; Convert num2 -> integer

mov esi, num2

call atoi

; Add

add eax, ebx ; eax = num1 + num2

; Convert result back to string

mov edi, res+10 ; point to end of buffer

call itoa ; eax -> string in res

; Print "Result = "

mov eax, 4

mov ebx, 1

mov ecx, resultMsg

mov edx, lenRes

int 0x80

; Print result

mov eax, 4

mov ebx, 1

mov ecx, edi

mov edx, 10

int 0x80

; Print newline

mov eax, 4

mov ebx, 1

mov ecx, newline

mov edx, 1

int 0x80

; Exit

mov eax, 1

xor ebx, ebx

int 0x80

; -------- Functions --------

; atoi: ASCII string -> integer

; esi = string address, eax = result

atoi:

xor eax, eax

.next:

mov bl, [esi]

cmp bl, 10 ; newline?

je .done

sub bl, '0'

imul eax, eax, 10

add eax, ebx

inc esi

jmp .next

.done:

ret

; itoa: integer (eax) -> string, edi = buffer end

; returns edi pointing to start of string

itoa:

mov ecx, 10

mov edx, 0

.loop:

xor edx, edx

div ecx

add dl, '0'

dec edi

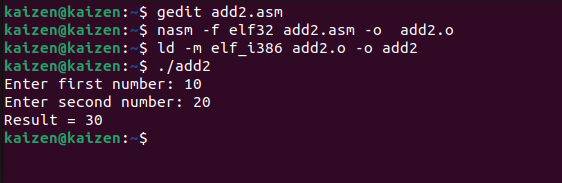
mov [edi], dl

test eax, eax

jnz .loop

ret

#Output-



1. Subtraction Code-

section .data

msg1 db "Enter first number: ",0

l1 equ $-msg1

msg2 db "Enter second number: ",0

l2 equ $-msg2

msg3 db "Subtraction = ",0

l3 equ $-msg3

minus db "-",0

nl db 10

section .bss

num1 resb 16

num2 resb 16

outbuf resb 16

section .text

global \_start

atoi: ; ecx=buf → eax=int

xor eax,eax

.next: mov bl,[ecx]

cmp bl,'0' ; below '0'

jb .done

cmp bl,'9' ; above '9'

ja .done

sub bl,'0'

imul eax,eax,10

add eax,ebx

inc ecx

jmp .next

.done: ret

itoa: ; eax=num, edi=buf

mov ecx,0

.loop: xor edx,edx

mov ebx,10

div ebx

add dl,'0'

push edx

inc ecx

test eax,eax

jnz .loop

.w: pop edx

mov [edi],dl

inc edi

loop .w

mov edx,edi

sub edx,outbuf

ret

\_start:

; ---- prompt 1 ----

mov eax,4; write

mov ebx,1

mov ecx,msg1

mov edx,l1

int 0x80

; read first

mov eax,3

mov ebx,0

mov ecx,num1

mov edx,16

int 0x80

mov ecx,num1

call atoi

push eax ; save first

; ---- prompt 2 ----

mov eax,4

mov ebx,1

mov ecx,msg2

mov edx,l2

int 0x80

; read second

mov eax,3

mov ebx,0

mov ecx,num2

mov edx,16

int 0x80

mov ecx,num2

call atoi

mov ebx,eax ; second

pop eax ; first

sub eax,ebx

mov ecx,eax

; ---- print result label ----

mov eax,4

mov ebx,1

mov ecx,msg3

mov edx,l3

int 0x80

; ---- handle sign ----

cmp ecx,0

jge .pos

neg ecx

mov eax,4

mov ebx,1

mov ecx,minus

mov edx,1

int 0x80

.pos:

; convert & print number

mov eax,ecx

mov edi,outbuf

call itoa

mov eax,4

mov ebx,1

mov ecx,outbuf

int 0x80

; newline

mov eax,4

mov ebx,1

mov ecx,nl

mov edx,1

int 0x80

; exit

mov eax,1

xor ebx,ebx

int 0x80

#Output-

A computer screen with white text

AI-generated content may be incorrect.

1. Multiplication Code-

section .data

msg1 db "Enter first number: ",0

len1 equ $-msg1

msg2 db "Enter second number: ",0

len2 equ $-msg2

resmsg db "Multiplication = ",0

lenres equ $-resmsg

newline db 10

section .bss

num1 resb 16

num2 resb 16

outbuf resb 16

section .text

global \_start

; --- atoi: ECX=buffer, returns EAX=int ---

atoi:

xor eax, eax

.next:

mov bl, [ecx]

cmp bl, '0'

jb .done

cmp bl, '9'

ja .done

sub bl, '0'

imul eax, eax, 10

add eax, ebx

inc ecx

jmp .next

.done:

ret

; --- itoa: EAX=number, EDI=buffer, returns length in EAX ---

itoa:

mov ebx, 10

xor ecx, ecx ; digit count

.loop:

xor edx, edx

div ebx ; EAX/10, remainder in EDX

add dl, '0'

push edx

inc ecx

test eax, eax

jnz .loop

mov eax, ecx ; length

mov esi, edi

.poploop:

pop edx

mov [esi], dl

inc esi

loop .poploop

mov byte [esi], 0

ret

; --- main ---

\_start:

; prompt 1

mov eax,4;write

mov ebx,1

mov ecx,msg1

mov edx,len1

int 0x80

; read num1

mov eax,3

mov ebx,0

mov ecx,num1

mov edx,16

int 0x80

; prompt 2

mov eax,4

mov ebx,1

mov ecx,msg2

mov edx,len2

int 0x80

; read num2

mov eax,3

mov ebx,0

mov ecx,num2

mov edx,16

int 0x80

; convert numbers

mov ecx,num1

call atoi

push eax

mov ecx,num2

call atoi

mov ebx,eax

pop eax

imul eax, ebx ; multiply

; print "Multiplication = "

push eax

mov eax,4

mov ebx,1

mov ecx,resmsg

mov edx,lenres

int 0x80

pop eax

; convert result to string

mov edi,outbuf

call itoa

mov edx,eax ; length

; print result

mov eax,4

mov ebx,1

mov ecx,outbuf

int 0x80

; print newline

mov eax,4

mov ebx,1

mov ecx,newline

mov edx,1

int 0x80

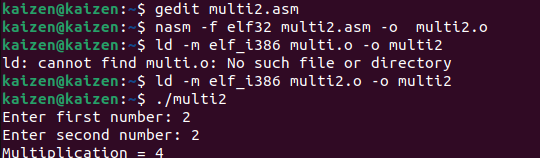
; exit

mov eax,1

xor ebx,ebx

int 0x80

#Output-



1. Division Code-

section .data

msg1 db "Enter dividend: ",0

len1 equ $-msg1

msg2 db "Enter divisor: ",0

len2 equ $-msg2

resmsg db "Quotient = ",0

lenres equ $-resmsg

newline db 10

section .bss

num1 resb 16

num2 resb 16

outbuf resb 16

section .text

global \_start

\_start:

; prompt dividend

mov eax,4

mov ebx,1

mov ecx,msg1

mov edx,len1

int 0x80

; read dividend

mov eax,3

mov ebx,0

mov ecx,num1

mov edx,16

int 0x80

; atoi inline for dividend

xor eax,eax

mov esi,num1

.d1: mov bl,[esi]

cmp bl,'0'

jb .d1done

cmp bl,'9'

ja .d1done

sub bl,'0'

imul eax,eax,10

add eax,ebx

inc esi

jmp .d1

.d1done:

push eax ; save dividend

; prompt divisor

mov eax,4

mov ebx,1

mov ecx,msg2

mov edx,len2

int 0x80

; read divisor

mov eax,3

mov ebx,0

mov ecx,num2

mov edx,16

int 0x80

; atoi inline for divisor

xor ebx,ebx

mov esi,num2

.d2: mov dl,[esi]

cmp dl,'0'

jb .d2done

cmp dl,'9'

ja .d2done

sub dl,'0'

imul ebx,ebx,10

add ebx,edx

inc esi

jmp .d2

.d2done:

; divide

pop eax ; dividend

xor edx,edx

div ebx ; quotient in eax

; print message

push eax

mov eax,4

mov ebx,1

mov ecx,resmsg

mov edx,lenres

int 0x80

pop eax

; itoa inline

mov edi,outbuf

mov ecx,0

.conv: xor edx,edx

mov ebx,10

div ebx

add dl,'0'

push edx

inc ecx

test eax,eax

jnz .conv

mov eax,ecx

mov esi,edi

.writedigs:

pop edx

mov [esi],dl

inc esi

loop .writedigs

mov byte [esi],0

mov edx,eax

; print result

mov eax,4

mov ebx,1

mov ecx,outbuf

int 0x80

; newline

mov eax,4

mov ebx,1

mov ecx,newline

mov edx,1

int 0x80

; exit

mov eax,1

xor ebx,ebx

int 0x80

#Output-

A computer screen shot of a number

AI-generated content may be incorrect.

B) GDB Execution

1. Addition Code-

section .data

result db 0 ; to store addition result

section .text

global \_start

\_start:

mov al, 50 ; load first number into AL

add al, 30 ; add second number to AL

mov [result], al ; store the result (80) in memory

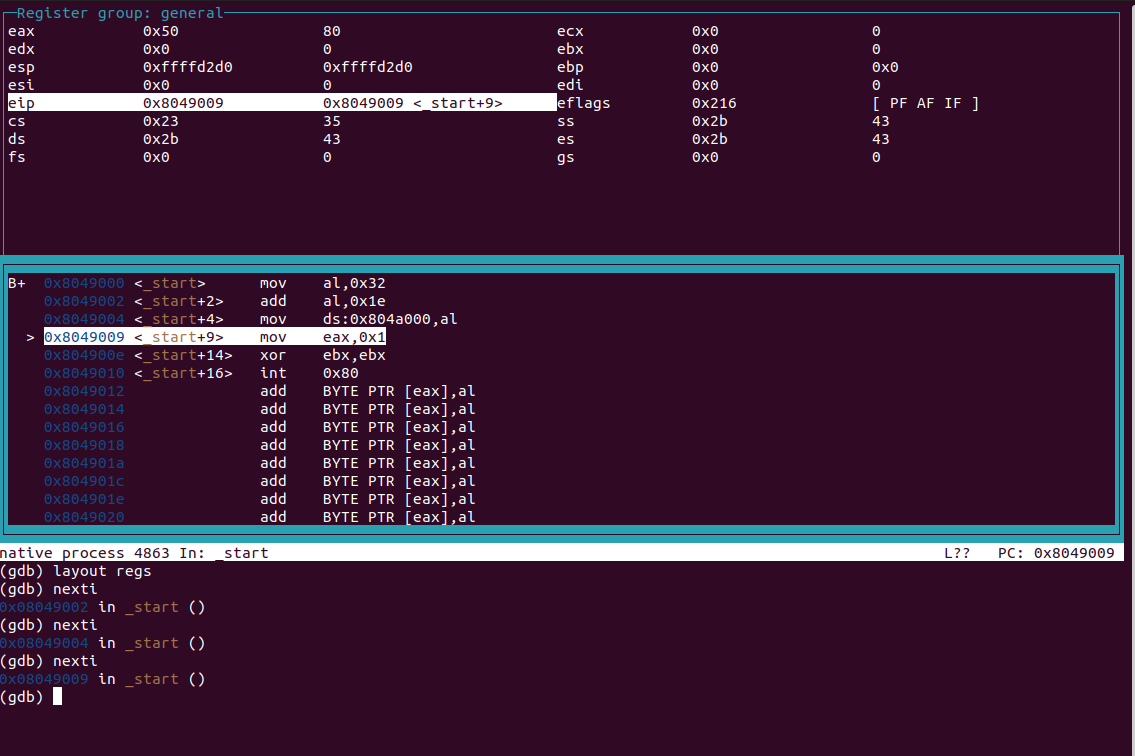
mov eax, 1 ; sys\_exit system call

xor ebx, ebx ; return 0

int 0x80

#Output-

Addition Of Number 50+30= 80



1. Subtraction Code-

section .data

result db 0 ; store result here

section .bss

section .text

global \_start

\_start:

; Put values into registers

mov eax, 50 ; EAX = 50

mov ebx, 30 ; EBX = 30

; Subtraction

sub eax, ebx ; EAX = EAX - EBX (50 - 30 = 20)

; Store only the lowest byte of result (20 fits in 1 byte)

mov [result], al

; Exit syscall

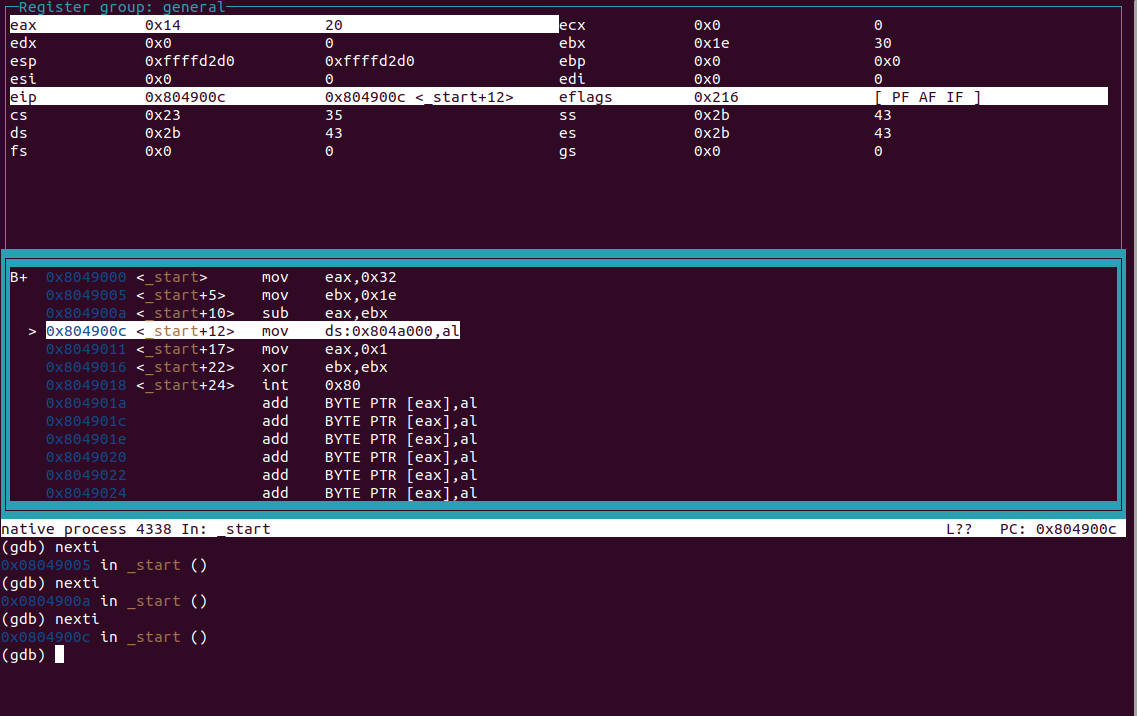
mov eax, 1

xor ebx, ebx

int 0x80

#Output-

Subtraction Of Number 50-30= 20



3) Multiplication Code-

section .data

result dw 0 ; 16-bit result storage

section .text

global \_start

\_start:

mov al, 12 ; AL = 12

mov bl, 15 ; BL = 15

mul bl ; AX = AL \* BL → 180

mov [result], ax ; store result (180)

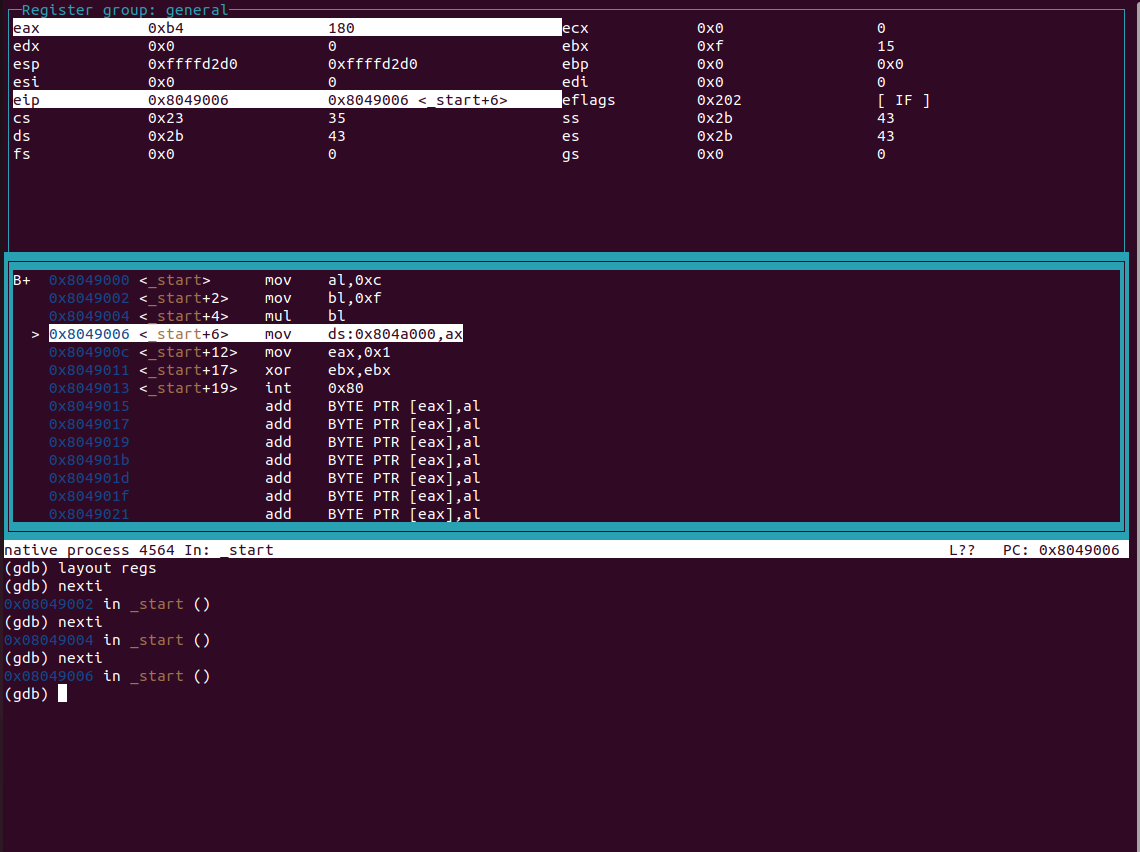
mov eax, 1 ; sys\_exit

xor ebx, ebx

int 0x80

#Output-

Multiplication Of Number 12\*15= 180



1. Division Code

section .data

quotient db 0 ; to store quotient

remainder db 0 ; to store remainder

section .text

global \_start

\_start:

mov ax, 144 ; AX = 144

mov bl, 12 ; BL = 12

div bl ; AL = 144 ÷ 12 = 12, AH = 0

mov [quotient], al

mov [remainder], ah

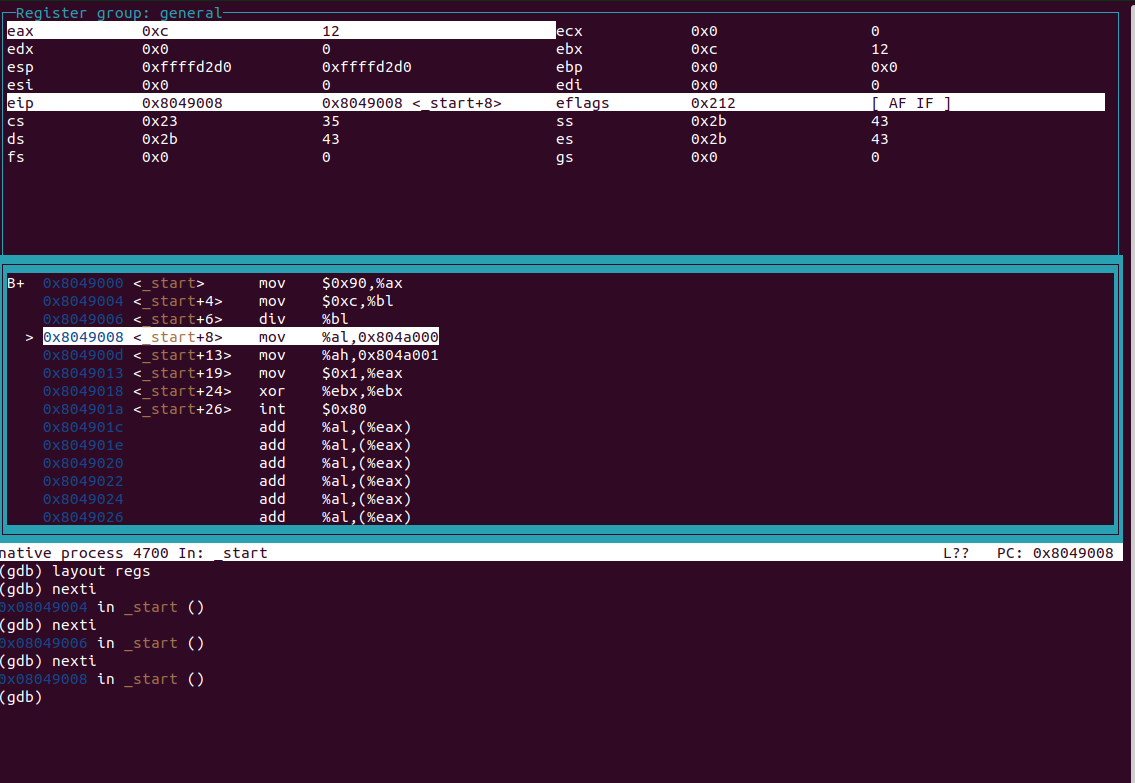
mov eax, 1 ; sys\_exit

xor ebx, ebx

int 0x80

#Output-

Division Of Number 12/144= 12



Assignment No-04

A. Array Addition (result less than 10 and 2nd by using AAM)

# Array Addition Code-

section .text

global \_start

\_start:

mov eax, x ; pointer to numbers

mov ebx, 0 ; EBX will store the sum

mov ecx, 5 ; number of elements

top:

add bl, [eax] ; add current element to sum

inc eax ; move pointer to next element

loop top ; repeat until ECX = 0

done:

add bl, '0' ; convert to ASCII

mov [sum], bl ; store result in "sum"

display:

mov edx, 1 ; message length

mov ecx, sum ; message to write

mov ebx, 1 ; file descriptor (stdout)

mov eax, 4 ; system call number (sys\_write)

int 0x80 ; call kernel

mov eax, 1 ; system call number (sys\_exit)

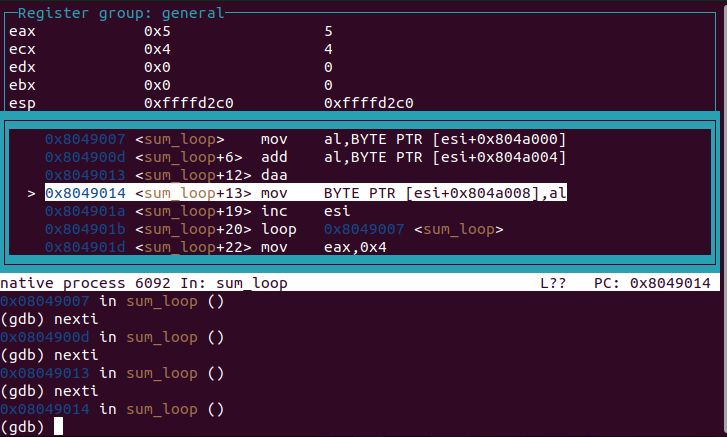
int 0x80 ; call kernel

section .data

x:

times 5 db 0 ; reserve 5 numbers (user can modify later)

sum:

db 0, 0xa ; result + newline 

B. String Operation.

global \_start

section .text

\_start:

cld ; clear direction flag (process forward)

mov ecx, len ; counter = string length

mov esi, s1 ; source string

mov edi, s2 ; destination string

loop\_here:

lodsb ; load byte from [esi] into AL

or al, 20h ; convert uppercase to lowercase (ASCII trick)

stosb ; store AL into [edi]

loop loop\_here ; repeat ECX times

; print result

mov edx, len ; message length (not hardcoded 20)

mov ecx, s2 ; message to write

mov ebx, 1 ; file descriptor (stdout)

mov eax, 4 ; system call (sys\_write)

int 0x80

; exit

mov eax, 1 ; sys\_exit

xor ebx, ebx ; return 0

int 0x80

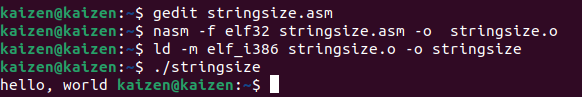
section .data

s1 db 'HELLO, WORLD', 0xa ; source string with newline

len equ $-s1 ; string length

section .bss

s2 resb len ; reserve same size as s1



v