PAI Assignment Lab 3 & 4

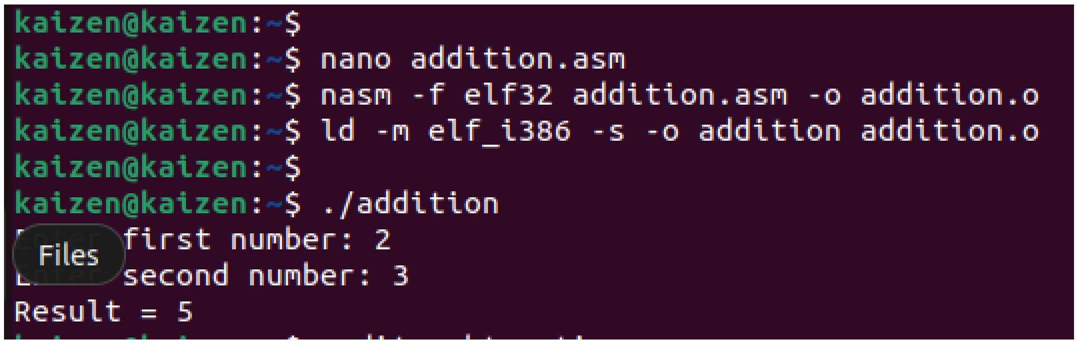
Name= SAMARTH VALSANGE
Roll no= 46
Enrollment No= ADT24SOCB0996
SY-06 Batch(B)

Asm codes of Addtion , Subtraction , Multiplication & Divison

Addition.asm

```
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 10,0

section .bss
   num1 resb 2
   num2 resb 2
   res  resb 2

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
mov eax, 3
mov ebx, 0
mov ecx, num1
mov edx, 2
int 0x80

; 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, 2
int 0x80

; Convert ASCII to number
mov al, [num1]
sub al, '0'
mov bl, [num2]
sub bl, '0'

; Add
add al, bl

; Convert back to ASCII
add al, '0'
mov [res], al

; 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, res
mov edx, 1
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
```



```
kaizen@kaizen:~$
kaizen@kaizen:~$ nano addition.asm
kaizen@kaizen:~$ nasm -f elf32 addition.asm -o addition.o
kaizen@kaizen:~$ ld -m elf_i386 -s -o addition addition.o
kaizen@kaizen:~$
kaizen@kaizen:~$ ./addition
[illegible] first number: 2
[illegible] second number: 3
Result = 5
```

Subtraction

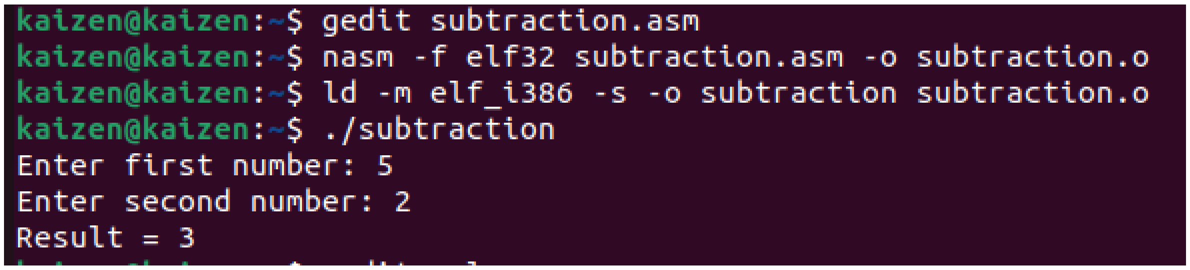
```
section .data
   msg1 db "Enter first number: ",0
   len1 equ $-msg1
   msg2 db "Enter second number: ",0
```

```
    len2 equ $-msg2

    resultMsg db "Result = ",0

    lenRes equ $-resultMsg

    newline db 10,0


section .bss

    num1 resb 2

    num2 resb 2

    res  resb 2


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

    mov eax, 3

    mov ebx, 0

    mov ecx, num1

    mov edx, 2
```

```
int 0x80

; 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, 2
int 0x80

; Convert ASCII to number
mov al, [num1]
sub al, '0'
mov bl, [num2]
sub bl, '0'

; Subtract
sub al, bl

; Convert back to ASCII
```

```
add al, '0'

mov [res], al


; 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, res

mov edx, 1

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
```



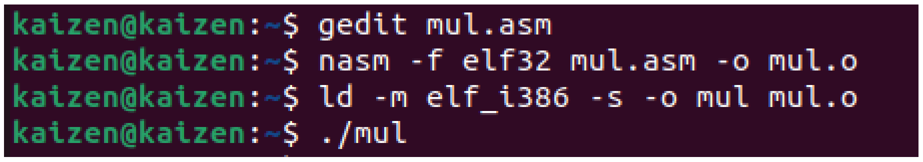
```
kaizen@kaizen:~$ gedit subtraction.asm
kaizen@kaizen:~$ nasm -f elf32 subtraction.asm -o subtraction.o
kaizen@kaizen:~$ ld -m elf_i386 -s -o subtraction subtraction.o
kaizen@kaizen:~$ ./subtraction
Enter first number: 5
Enter second number: 2
Result = 3
```

Multiplication.asm

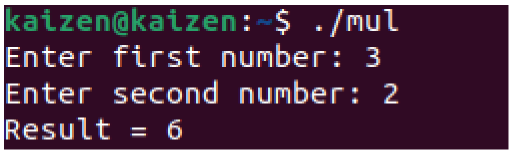
```
section .data

   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 10,0


section .bss

   num1 resb 2

   num2 resb 2

   res  resb 2


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

    mov eax, 3

    mov ebx, 0

    mov ecx, num1

    mov edx, 2

    int 0x80


    ; 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, 2
    int 0x80

    ; Convert ASCII to number
    mov al, [num1]
    sub al, '0'
    mov bl, [num2]
    sub bl, '0'

    ; Multiply
    mul bl      ; AX = AL * BL
    add al, '0'  ; Convert back to ASCII
    mov [res], al

    ; 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, res
    mov edx, 1
```

```
    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
```



```
kaizen@kaizen:~$ gedit mul.asm
kaizen@kaizen:~$ nasm -f elf32 mul.asm -o mul.o
kaizen@kaizen:~$ ld -m elf_i386 -s -o mul mul.o
kaizen@kaizen:~$ ./mul
```



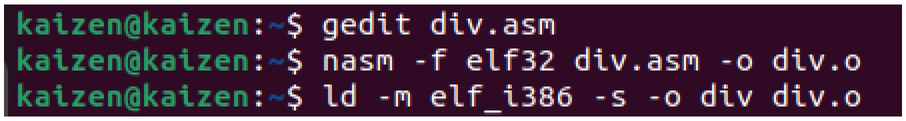
```
kaizen@kaizen:~$ ./mul
Enter first number: 3
Enter second number: 2
Result = 6
```

Division.asm

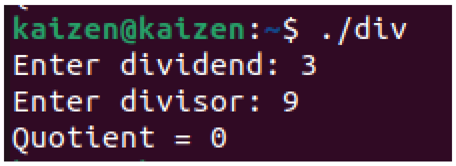
```
section .data
    msg1 db "Enter dividend: ",0
    len1 equ $-msg1
    msg2 db "Enter divisor: ",0
    len2 equ $-msg2
    resultMsg db "Quotient = ",0
    lenRes equ $-resultMsg
    newline db 10,0

section .bss
    num1 resb 2
    num2 resb 2
    res  resb 2

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
    mov eax, 3
    mov ebx, 0
    mov ecx, num1
    mov edx, 2
    int 0x80
```

```
; 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, 2

int 0x80


; Convert ASCII to number

mov al, [num1]

sub al, '0'

mov bl, [num2]

sub bl, '0'


; Divide

xor ah, ah    ; clear upper byte

div bl        ; AL = AL / BL, AH = remainder

add al, '0'

mov [res], al
```

```
; Print "Quotient = "
mov eax, 4
mov ebx, 1
mov ecx, resultMsg
mov edx, lenRes
int 0x80

; Print result
mov eax, 4
mov ebx, 1
mov ecx, res
mov edx, 1
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
```



```
kaizen@kaizen:~$ gedit div.asm
kaizen@kaizen:~$ nasm -f elf32 div.asm -o div.o
kaizen@kaizen:~$ ld -m elf_i386 -s -o div div.o
```



```
kaizen@kaizen:~$ ./div
Enter dividend: 3
Enter divisor: 9
Quotient = 0
```

Assignment No-04

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

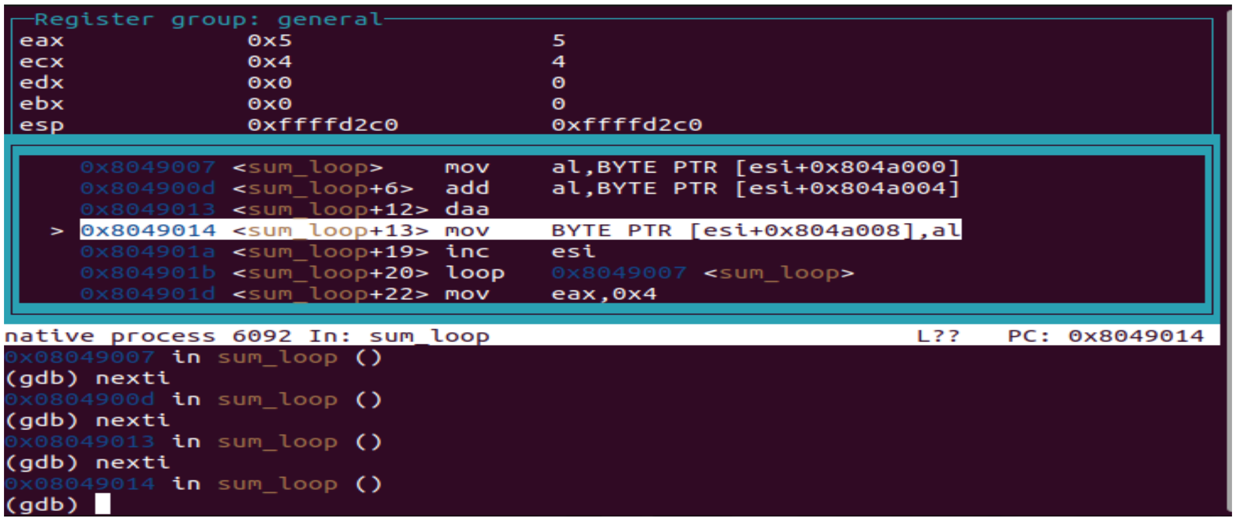
B. String Operation

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
```

Output:-



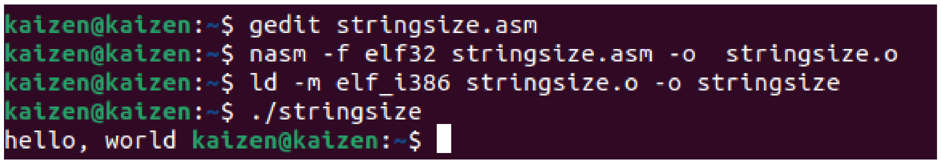
```
Register group: general
eax            0x5                 5
ecx            0x4                 4
edx            0x0                 0
ebx            0x0                 0
esp            0xffffd2c0          0xffffd2c0

     0x8049007 <sum_loop>     mov    al,BYTE PTR [esi+0x804a000]
     0x804900d <sum_loop+6>   add    al,BYTE PTR [esi+0x804a004]
     0x8049013 <sum_loop+12>  daa
   > 0x8049014 <sum_loop+13>  mov    BYTE PTR [esi+0x804a008],al
     0x804901a <sum_loop+19>  inc    esi
     0x804901b <sum_loop+20>  loop   0x8049007 <sum_loop>
     0x804901d <sum_loop+22>  mov    eax,0x4

native process 6092 In: sum_loop                            L??   PC: 0x8049014
0x08049007 in sum_loop ()
(gdb) nexti
0x0804900d in sum_loop ()
(gdb) nexti
0x08049013 in sum_loop ()
(gdb) nexti
0x08049014 in sum_loop ()
(gdb)
```

B. String Operation Code-

```
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
```

Output:-



```
kaizen@kaizen:~$ gedit stringsize.asm
kaizen@kaizen:~$ nasm -f elf32 stringsize.asm -o  stringsize.o
kaizen@kaizen:~$ ld -m elf_i386 stringsize.o -o stringsize
kaizen@kaizen:~$ ./stringsize
hello, world kaizen@kaizen:~$
```