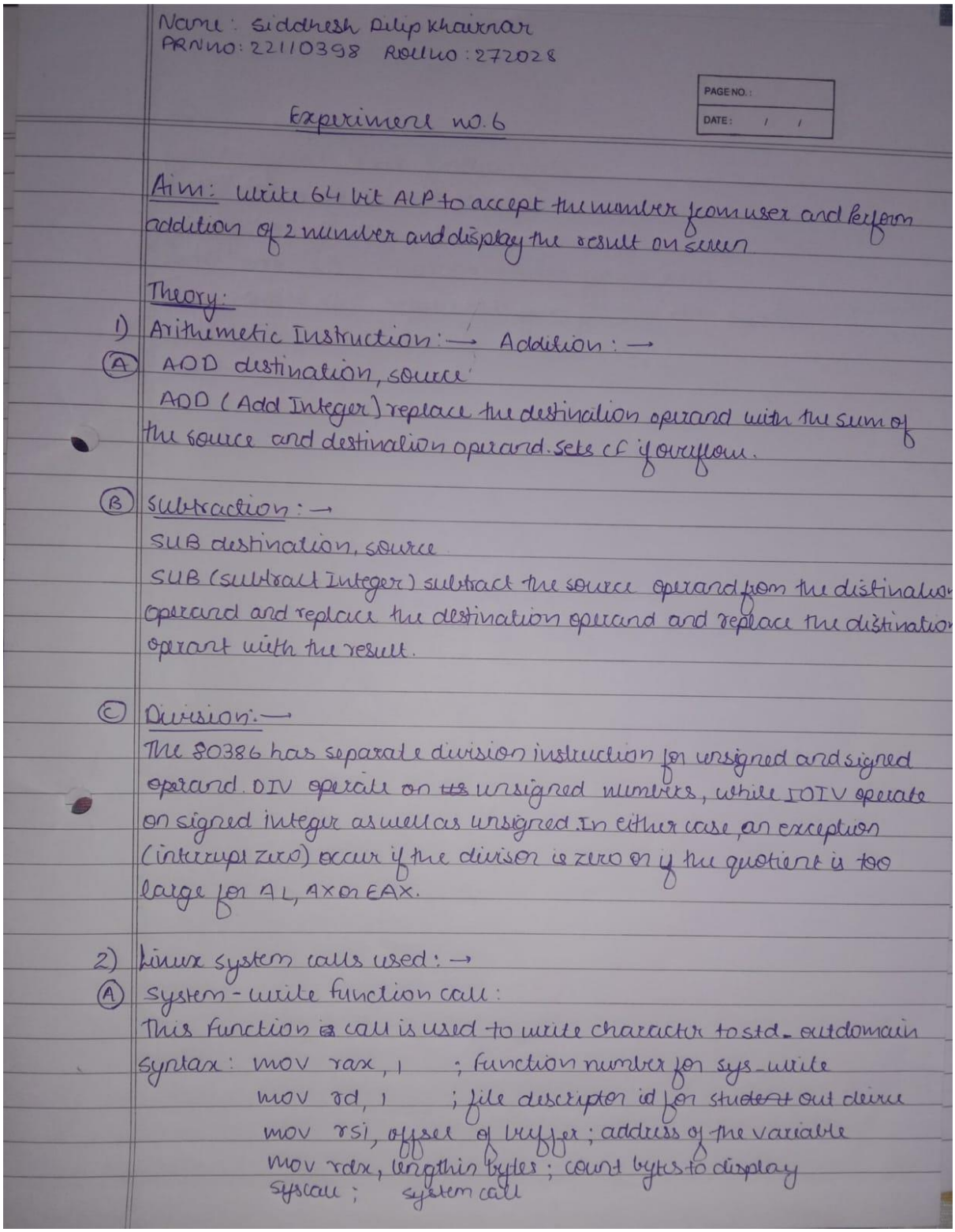
MP Practical- 6

Name- Siddhesh Dilip Khairnar

Roll No.- 272028

Batch- B2



Name: Siddhesh Dilip Khairnar
PRN no: 22110398 Roll no: 272028

PAGE NO.:
DATE: / /

Experiment no. 6

Aim: Write 64 bit ALP to accept the number from user and perform addition of 2 number and display the result on screen

Theory:

1) Arithmetic Instruction:→ Addition:→

(A) ADD destination, source

ADD (Add Integer) replace the destination operand with the sum of the source and destination operand. sets CF if overflow.

(B) Subtraction:→

SUB destination, source

SUB (subtract Integer) subtract the source operand from the distination operand and replace the destination operand and replace the distination oprant with the result.

(C) Division:→

The 80386 has separate division instruction for unsigned and signed operand. DIV operate on ~~its~~ unsigned numbers, while IDIV operate on signed integer as well as unsigned. In either case, an exception (interrupt zero) occur if the divisor is zero or if the quotient is too large for AL, AX or EAX.

2) Linux system calls used:→

(A) System - write function call:

This function call is used to write character to std. outdomain

Syntax: mov rax, 1 ; function number for sys_write
mov rd, 1 ; file descriptor id for ~~student~~ out deive
mov rsi, offset of buffer ; address of the variable
mov rdx, length in bytes ; count bytes to display
syscall ; system call

PAGE NO.:
DATE: / /

(b) ~~System~~ system_exit function call:
This function call is used to exist.
syntax:
mov rax, 60 ; function number for sys_exist
mov rdi, 0 ; return code ~~from~~ for zero error syscall.

Code:

```
%macro scall 4
        mov rax,%1
        mov rdi,%2
        mov rsi,%3
        mov rdx,%4
        syscall
%endmacro

section .data

        m1 db "Enter first 2digit number=",10d,13d
        l1 equ $-m1

        m2 db "The 1 is =",10d,13d
        l2 equ $-m2

        m3 db "",10d,13d
        l3 equ $-m3

        m4 db "Enter second 2digit number=",10d,13d
        l4 equ $-m4

        m5 db "The 2 is =",10d,13d
        l5 equ $-m5

        m6 db "Sum is =",10d,13d
```

```
        l6 equ $-m6

section .bss
        num resb 20
        array resb 200
        char_ans resb 16

section .text
global _start

_start:

;--------------------first number-------------------------
        scall 1,1,m1,l1
        scall 0,0,num,3

        call accept_proc

        mov rbp,array

        mov [rbp],bx

        scall 1,1,m2,l2

        mov ax,bx

        call display_proc
```

```
        inc rbp

;----------------------second number----------------------------
        scall 1,1,m4,l4
        scall 0,0,num,3

        call accept_proc


        mov [rbp],bx


        scall 1,1,m5,l5

        mov ax,bx

        call display_proc
;----------------------sum---------------------

   mov rbp,array

        mov dx,[rbp]
        add bx,dx

        inc rbp

        mov cx,[rbp]
        add bx,cx
```

```
        scall 1,1,m6,l6

        mov ax,bx

        call display_proc

;----------------------exit---------------------
        mov rax,60
        mov rdi,0
        syscall


;----------------------accept procedure--------------------
accept_proc:
        mov rsi,num
        mov rbx,0
        mov rax,0
        mov rcx,2
back:
        rol rbx,04
        mov al,[rsi]

        cmp al,39h
        jbe next
        sub al,07h
next:
```

```
        sub al,30h
        add bx,ax

        inc rsi
        dec rcx
        jnz back
ret

;----------------dispaly procedure----------------------------

display_proc:

        mov rbp,char_ans
        mov rcx,2

up3:
        rol al,04
        mov dl,al

        and dl,0Fh

        cmp dl,09h
        jbe next1

        add dl,07h
next1:
        add dl,30h

        mov [rbp],dl
```

```
        inc rbp
        dec rcx

        jnz up3

        scall 1,1,char_ans,3

        scall 1,1,m3,l3
ret
```

Output: