;Name: Uday Pratap Singh

;Roll No: 7259

;Date: 17 March, 2025

%macro io 4

mov rax, %1 ; System call number (1 for write, 0 for read)

mov rdi, %2 ; File descriptor (1 for stdout, 0 for stdin)

mov rsi, %3 ; Buffer address

mov rdx, %4 ; Buffer size

syscall ; Invoke system call

%endmacro

%macro exit 0

mov rax, 60 ; System call number for exit

mov rdi, 0 ; Exit status code (0 for success)

syscall ; Invoke system call

%endmacro

section .data

msg1 db "Write an x86/64 ALP to accept 5 hexadecimal numbers from user and",\

" store them in an array and display the count of positive and negative numbers", 10

msg1len equ $-msg1

msg2 db "Enter 5 64-bit hexadecimal numbers (0-9, A-F only): ", 10

msg2len equ $-msg2

msg3 db "The count of positive numbers is: ", 10

msg3len equ $-msg3

msg4 db "The count of negative numbers is: ", 10

msg4len equ $-msg4

newline db 10

section .bss

asciinum resb 17 ; Buffer for user input (16 characters + 1 for null terminator)

hexnum resq 5 ; Array to store 5 64-bit hexadecimal numbers

pcount resb 1 ; Positive count

ncount resb 1 ; Negative count

section .text

global \_start

\_start:

; Display initial message

io 1, 1, msg1, msg1len

io 1, 1, msg2, msg2len

; Input 5 hexadecimal numbers

mov rcx, 5 ; Loop counter for 5 inputs

mov rsi, hexnum ; Address to store the converted numbers

xor byte [pcount], 0 ; Initialize positive count

xor byte [ncount], 0 ; Initialize negative count

next\_input:

push rsi ; Save registers

push rcx

io 0, 0, asciinum, 17 ; Read input from user (up to 16 characters)

call ascii\_hex64 ; Convert ASCII to hexadecimal

; Store the converted number

pop rcx

pop rsi

mov [rsi], rbx

add rsi, 8 ; Move to the next storage slot

loop next\_input ; Repeat for 5 numbers

; Count positive and negative numbers

mov rcx, 5

mov rsi, hexnum

check\_numbers:

mov rax, [rsi] ; Load the number

bt rax, 63 ; Test bit 63 (sign bit)

jnc is\_positive ; Jump if no carry (positive number)

inc byte [ncount] ; Increment negative count

jmp skip\_check

is\_positive:

inc byte [pcount] ; Increment positive count

skip\_check:

add rsi, 8 ; Move to the next number

loop check\_numbers

; Display positive count

io 1, 1, msg3, msg3len

mov bl, [pcount]

call hex\_ascii8

; Display negative count

io 1, 1, msg4, msg4len

mov bl, [ncount]

call hex\_ascii8

; Exit program

exit

; Function to convert a single byte to ASCII and print

hex\_ascii8:

mov rsi, asciinum ; Address of output buffer

mov rcx, 2 ; Loop for 2 characters to convert to hexadecimal

next\_digit:

rol bl, 4

mov al, bl ; Isolate the nibble

and al, 0Fh ; Mask the lower 4 bits

cmp al, 9

jbe add\_0 ; Convert to '0'-'9'

add al, 7h ; Convert to 'A'-'F'

add\_0:

add al, 30h ; Convert to ASCII

mov [rsi], al ; Store in output buffer

inc rsi ; Move to next character

loop next\_digit

io 1, 1, asciinum, 2 ; Print the converted number

io 1, 1, newline, 1 ; Print newline

ret

; Function to convert ASCII to 64-bit hexadecimal

ascii\_hex64:

mov rsi, asciinum ; Address of input buffer

xor rbx, rbx ; Clear rbx to store the number

mov rcx, 16 ; Loop for 16 characters

next\_char:

rol rbx, 4 ; Make space for the next nibble

mov al, [rsi] ; Load a character

cmp al, '9'

jbe convert\_digit ; Convert '0'-'9'

sub al, 7h ; Adjust 'A'-'F'

convert\_digit:

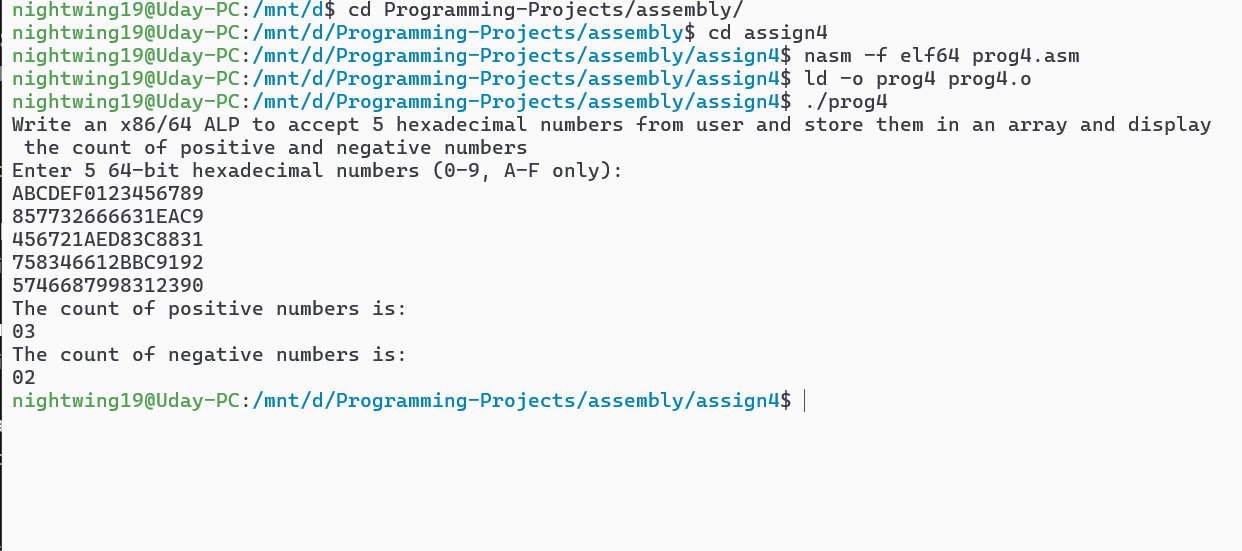
sub al, 30h ; Convert ASCII to numeric value

add bl, al ; Add to rbx

inc rsi ; Move to next character

loop next\_char

ret



; Write X86/64 ALP to convert 4-digit Hex number into its equivalent BCD number and

; 5- digit BCD number into its equivalent HEX number. Make your program user friendly

; to accept the choice from user for: (a) HEX to BCD b) BCD to HEX (c) EXIT.

; Display proper strings to prompt the user while accepting the input and displaying the result.

;Name: Uday Pratap Singh

;Roll No: 7259

;Date: 17 March, 2025

; Macro for write

%macro write 2

mov rax, 1

mov rdi, 1

mov rsi, %1

mov rdx, %2

syscall

%endmacro

; Macro for read

%macro read 2

mov rax, 0

mov rdi, 0

mov rsi, %1

mov rdx, %2

syscall

%endmacro

section .data

msgname db "Write X86/64 ALP to convert 4-digit Hex number into \

its equivalent BCD number and 5- digit BCD number into its equivalent \

HEX number. Make your program user friendly to accept the choice from \

user for: (a) HEX to BCD b) BCD to HEX (c) EXIT.", 10

msgnamelen equ $-msgname

msg db "------------------MENU------------------", 10

db " 1. Hex to BCD ", 10

db " 2. BCD to Hex ", 10

db " 3. Exit ", 10

db "Enter your choice : "

msglen equ $-msg

msg1 db "Number : "

len equ $-msg1

endl db 10

section .bss

num resd 1

choice resb 2

a\_hex resb 5

a\_bcd resb 6

buffer resb 5

temp resb 4

section .text

global \_start

\_start :

write msgname,msgnamelen

main\_menu :

write msg, msglen

read choice, 2

cmp byte[choice], "1"

je case1

cmp byte[choice], "2"

je case2

cmp byte[choice], "3"

je case3

jmp main\_menu

case1 :

write msg1, len

read a\_hex, 5

call hex\_bcd

jmp main\_menu

case2 :

write msg1, len

read a\_bcd, 6

call bcd\_hex

jmp main\_menu

case3 :

mov rax, 60

mov rdi, 0

syscall

; Procedure to convert a hexadecimal number to bcd number

hex\_bcd:

call ascii\_hex ; first convert the number to hex form

mov ax, bx ; move the hex number into accumualtor

mov bx, 0AH ; move 0AH ie (10 in deccimal)

mov rcx, 5

convert :

mov rdx, 0 ; initialize rdx with 0

div bx ; div ax with bx

push dx ; store the remainder in the stack, while the quotient remains in the accumualator

loop convert

mov rcx, 5

mov rsi, buffer

print\_bcd :

pop ax ; pop the remainder(reverse order)

add ax, 30h ; add 30h to it

mov [rsi], ax ; move that value in the buffer

inc rsi ; increment the pointer

loop print\_bcd

write buffer, 5 ; print the answer (BCD number)

write endl, 1

ret

; Procedure to convert a bcd number to a hexadecimal number

bcd\_hex :

mov rsi, a\_bcd

mov rcx, 5

mov bx, 0AH

mov rax, 0 ; initialize rax with 0

b\_convert :

mul bx ; multiply the rax with 0Ah(10 in decimal)

mov dl, [rsi] ; move the digit stored in buffer in dl

sub dl, 30h ; subtract 30h from the digit

add al, dl ; add the number obtained to al

inc rsi ; increment the pointer

loop b\_convert

mov bx, ax

call display

ret

; Procedure to convert ascii to heaxadecimal number

ascii\_hex:

mov rbx, 0

mov rsi, a\_hex

mov rcx, 4

next :

rol bx, 4

mov al, [rsi]

cmp al, '9'

jbe sub30h

sub al, 7h

sub30h :

sub al, 30h

add bl, al

inc rsi

loop next

ret

; Procedure to convert hexdecimal number to ascii

display :

mov rsi, temp

mov rcx, 4

next1 :

rol bx, 4

mov al, bl

and al, 0Fh

cmp al, 9

jbe add30h

add al, 7h

add30h :

add al, 30h

mov [rsi], al

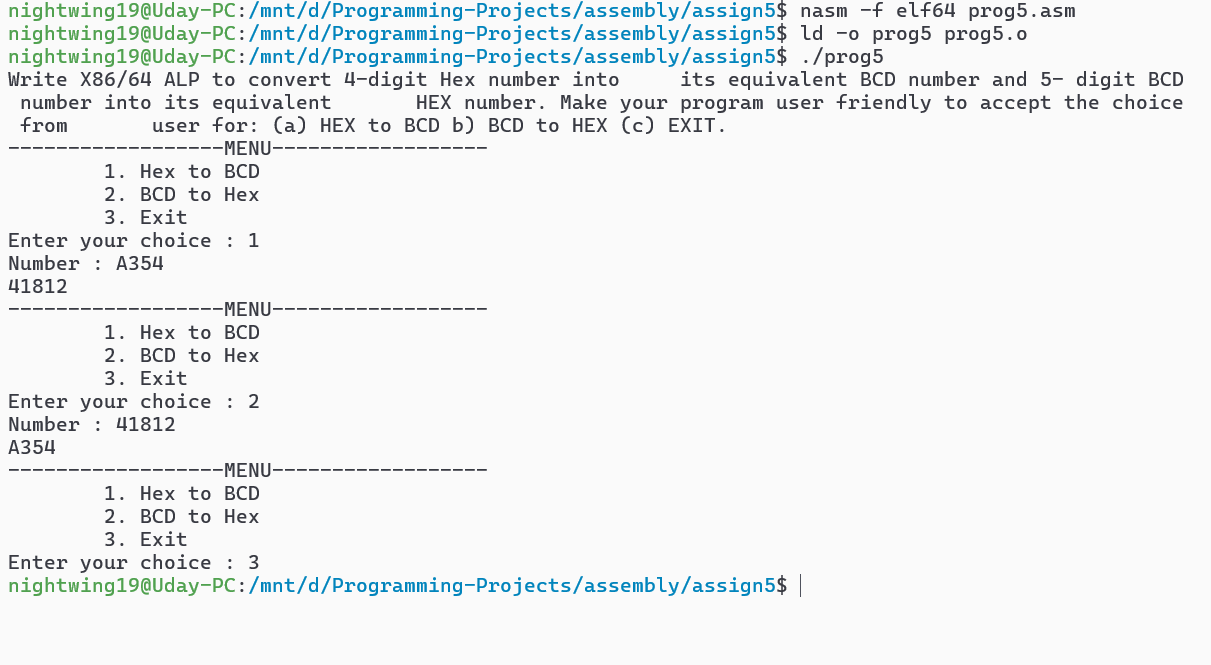
inc rsi

loop next1

write temp, 4

write endl, 1

ret



;Program for non overlapped block transfer

;Name: Uday Pratap Singh

;Roll No: 7259

;Date: 24 March, 2025

%macro io 4

mov rax,%1 ; System call number (1 for write, 0 for read)

mov rdi,%2 ; File descriptor (1 for stdout, 0 for stdin)

mov rsi,%3 ; Buffer address

mov rdx,%4 ; Buffer size

syscall ; Invoke system call

%endmacro

%macro exit 0

mov rax,60 ; System call number for exit

mov rdi,0 ; Exit status code (0 for success)

syscall ; Invoke system call

%endmacro

section .data

source dq 0x123456789ABCDEF0, \

0x0FEDCBA987654321, \

0xA1B2C3D4E5F60718, \

0xFFFFFFFF00000000, \

0x7F8E9DA1BC2D3E4F

dest dq 0,0,0,0,0

len db 5

msg1 db "Source: ",10

msg1len equ $-msg1

msg2 db "Destination: ",10

msg2len equ $-msg2

menu db "0. Exit",10,"1. Non-overlapped block transfer w/o string instructions"\

,10, "2. Non-overlapped block transfer with string instructions",10

menulen equ $-menu

newline db 10

arrow db " ---> "

arrowlen equ $-arrow

section .bss

ascii64 resb 16

choice resb 2

section .text

global \_start

\_start:

io 1,1,menu,menulen

io 0,0,choice,2

cmp byte[choice], "1"

je opt1

cmp byte[choice], "2"

je opt2

exit ; default option exit

opt1:

call print\_src

mov rsi,source

mov rdi,dest

mov rcx,5

lp1:

mov rbx,[rsi]

mov [rdi],rbx

add rsi, 8

add rdi, 8

loop lp1

call print\_dest

exit

opt2:

call print\_src

mov rsi,source

mov rdi,dest

mov rcx,5

rep movsq

call print\_dest

exit

print\_src:

io 1,1,msg1,msg1len

mov rsi,source

mov rcx, 5

next:

mov rbx,rsi

push rcx

push rsi

call hex\_ascii64

io 1,1,arrow,arrowlen

pop rsi

mov rbx,[rsi]

push rsi

call hex\_ascii64

io 1,1,newline,1

pop rsi

pop rcx

add rsi, 8

loop next

ret

print\_dest:

io 1,1,msg2,msg2len

mov rdi,dest

mov rcx,5

next2:

mov rbx,rdi

push rcx

push rdi

call hex\_ascii64

io 1,1,arrow,arrowlen

pop rdi

mov rbx,[rdi]

push rdi

call hex\_ascii64

io 1,1,newline,1

pop rdi

pop rcx

add rdi, 8

loop next2

ret

hex\_ascii64:

mov rsi, ascii64 ; Address of output buffer

mov rcx, 16 ; Loop for 16 characters

next3:

rol rbx, 4 ; Get the most significant nibble

mov al, bl ; Isolate the nibble

and al, 0Fh ; Mask the lower 4 bits

cmp al, 9

jbe add30h ; Convert to '0'-'9'

add al, 7h ; Convert to 'A'-'F'

add30h:

add al, 30h ; Convert to ASCII

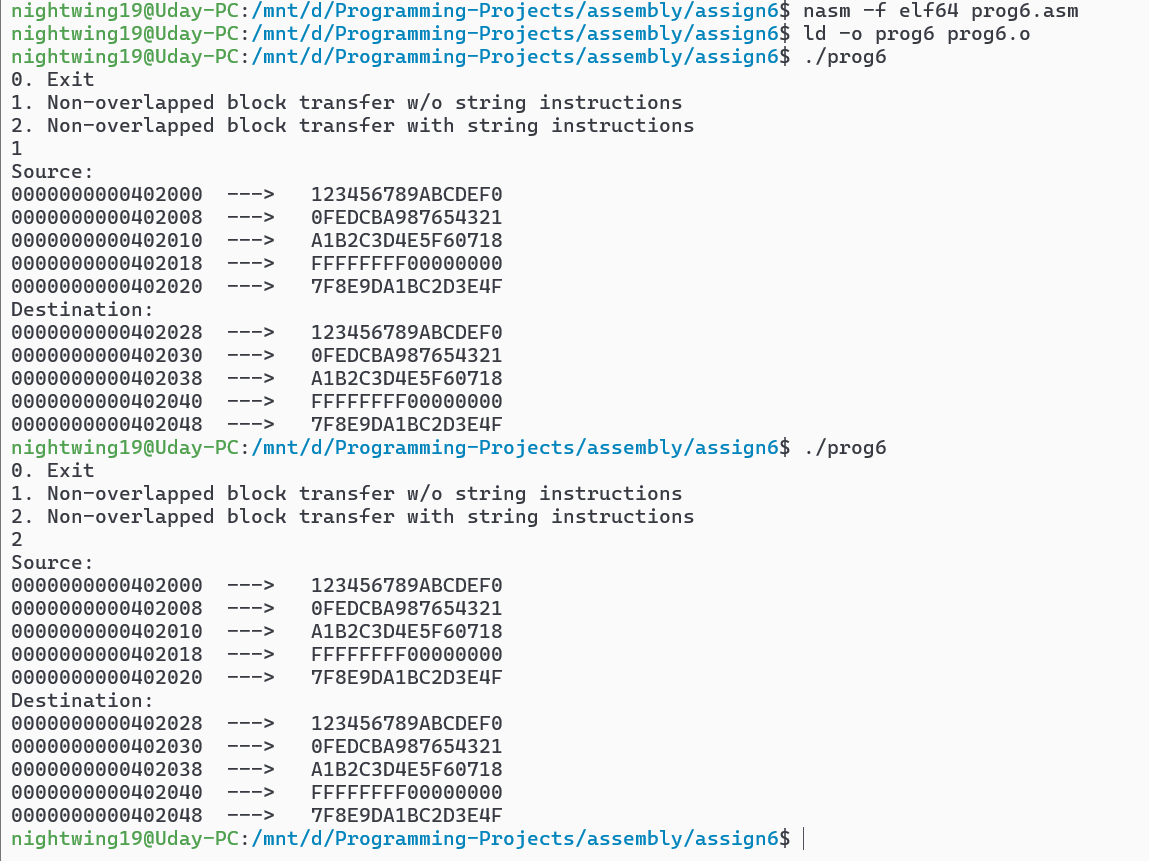
mov [rsi], al ; Store in output buffer

inc rsi ; Move to next character

loop next3

io 1,1, ascii64, 16 ; io 1,1, the converted number

ret



;Program for overlapped block transfer

;Name: Uday Pratap Singh

;Roll No: 7259

;Date: 24 March, 2025

%macro io 4

mov rax,%1 ; System call number (1 for write, 0 for read)

mov rdi,%2 ; File descriptor (1 for stdout, 0 for stdin)

mov rsi,%3 ; Buffer address

mov rdx,%4 ; Buffer size

syscall ; Invoke system call

%endmacro

%macro exit 0

mov rax,60 ; System call number for exit

mov rdi,0 ; Exit status code (0 for success)

syscall ; Invoke system call

%endmacro

section .data

source dq 0x123456789ABCDEF0, \

0x0FEDCBA987654321, \

0xA1B2C3D4E5F60718, \

0xFFFFFFFF00000000, \

0x7F8E9DA1BC2D3E4F

msg1 db "Source: ",10

msg1len equ $-msg1

msg2 db "Destination: ",10

msg2len equ $-msg2

menu db "0. Exit",10,"1. Overlapped block transfer w/o string instructions"\

,10, "2. Overlapped block transfer with string instructions",10

menulen equ $-menu

newline db 10

arrow db " ---> "

arrowlen equ $-arrow

section .bss

ascii64 resb 16

choice resb 2

section .text

global \_start

\_start:

io 1,1,menu,menulen

io 0,0,choice,2

cmp byte[choice], "1"

je opt1

cmp byte[choice], "2"

je opt2

exit ; default option exit

opt1:

call print\_src

io 1,1,msg2,msg2len

mov rsi,source

add rsi,20h

mov rdi,source

add rdi,30h

mov rcx,5

lp1:

mov rbx,[rsi]

mov [rdi],rbx

sub rsi, 8

sub rdi, 8

loop lp1

call print\_dest

exit

opt2:

call print\_src

io 1,1,msg2,msg2len

mov rsi,source

add rsi,20h

mov rdi,source

add rdi,30h

std

mov rcx,5

rep movsq

call print\_dest

exit

print\_src:

io 1,1,msg1,msg1len

mov rsi,source

add rsi,20h

mov rcx, 5

next:

mov rbx,rsi

push rcx

push rsi

call hex\_ascii64

io 1,1,arrow,arrowlen

pop rsi

mov rbx,[rsi]

push rsi

call hex\_ascii64

io 1,1,newline,1

pop rsi

pop rcx

sub rsi, 8

loop next

ret

print\_dest:

mov rdi,source

add rdi,30h

mov rcx,5

next2:

mov rbx,rdi

push rcx

push rdi

call hex\_ascii64

io 1,1,arrow,arrowlen

pop rdi

mov rbx,[rdi]

push rdi

call hex\_ascii64

io 1,1,newline,1

pop rdi

pop rcx

sub rdi, 8

loop next2

ret

hex\_ascii64:

mov rsi, ascii64 ; Address of output buffer

mov rcx, 16 ; Loop for 16 characters

next3:

rol rbx, 4 ; Get the most significant nibble

mov al, bl ; Isolate the nibble

and al, 0Fh ; Mask the lower 4 bits

cmp al, 9

jbe add30h ; Convert to '0'-'9'

add al, 7h ; Convert to 'A'-'F'

add30h:

add al, 30h ; Convert to ASCII

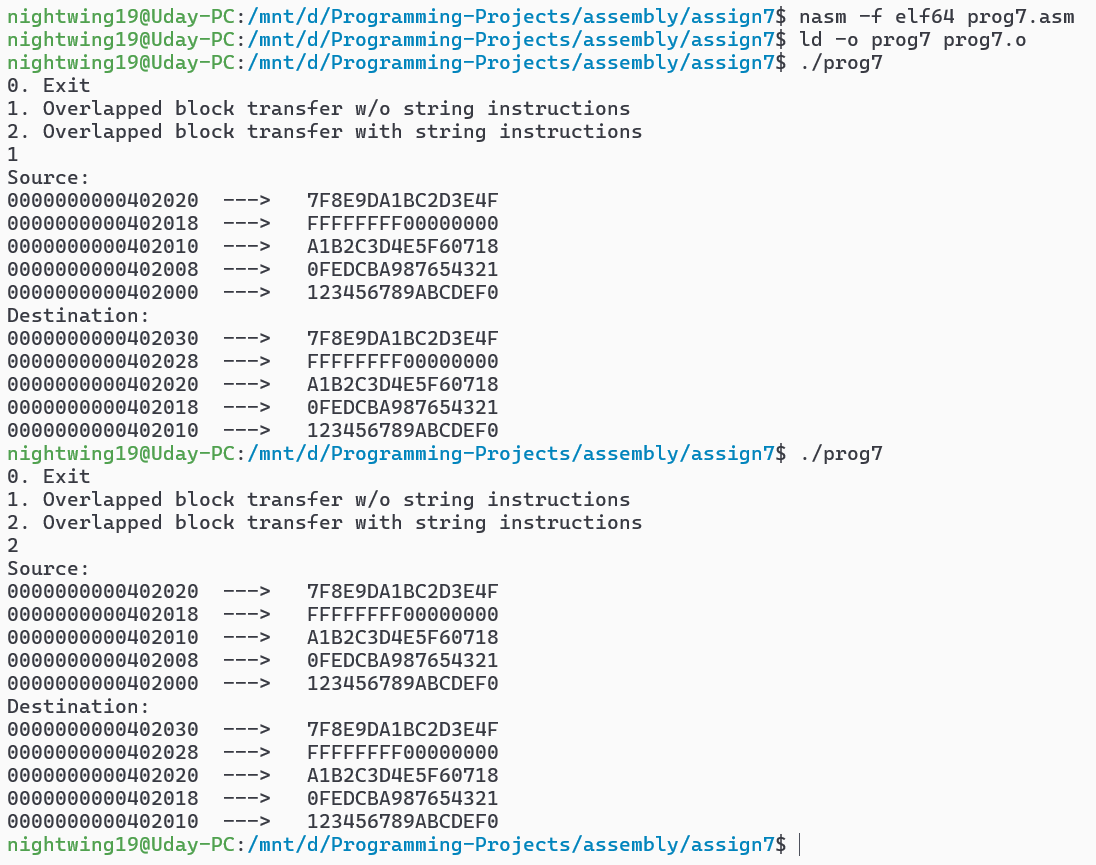
mov [rsi], al ; Store in output buffer

inc rsi ; Move to next character

loop next3

io 1,1, ascii64, 16 ; io 1,1, the converted number

ret



; Name: Uday Pratap Singh

; Roll No: 7259

; Date: 3 April, 2025

; Program to multiply two 8-bit hex numbers using Successive Addition and

; Shift-Add methods

; Uses NASM syntax for Linux environment

section .data

msg1 db "Enter first 8-bit hex number (2 digits): ", 10

msg1len equ $ - msg1

msg2 db "Enter second 8-bit hex number (2 digits): ", 10

msg2len equ $ - msg2

msg3 db "Result (Successive Addition): ", 10

msg3len equ $ - msg3

msg4 db "Result (Shift and Add): ", 10

msg4len equ $ - msg4

dispbuff db 4 dup(0) ; Buffer for displaying 4-digit hex result

newline db 10 ; Newline character

section .bss

ascii\_num resb 3 ; Buffer for 2-digit hex input + newline

num1 resb 1 ; First hex number

num2 resb 1 ; Second hex number

; Macro to print messages using sys\_write - Jatin Yadav 7226

%macro PRINT 2

mov rax, 1 ; System call number for sys\_write

mov rdi, 1 ; File descriptor (1 = stdout)

mov rsi, %1 ; Buffer address

mov rdx, %2 ; Length of buffer

syscall ; Make system call

%endmacro

; Macro to accept input using sys\_read - Jatin Yadav 7226

%macro ACCEPT 2

mov rax, 0 ; System call number for sys\_read

mov rdi, 0 ; File descriptor (0 = stdin)

mov rsi, %1 ; Buffer address

mov rdx, %2 ; Length of buffer

syscall ; Make system call

%endmacro

section .text

global \_start

\_start:

; Display message for first number - Jatin Yadav 7226

PRINT msg1, msg1len

; Accept first number

ACCEPT ascii\_num, 3 ; 2 digits + newline

call Ascii\_to\_Hex ; Convert to hex

mov [num1], bl ; Store first number

; Display message for second number - Jatin Yadav 7226

PRINT msg2, msg2len

; Accept second number

ACCEPT ascii\_num, 3 ; 2 digits + newline

call Ascii\_to\_Hex ; Convert to hex

mov [num2], bl ; Store second number

; Perform multiplication using Successive Addition - Jatin Yadav 7226

call Succ\_Add

PRINT msg3, msg3len ; Display result header

PRINT dispbuff, 4 ; Display result

PRINT newline, 1 ; Newline for formatting

; Perform multiplication using Shift and Add - Jatin Yadav 7226

call Shift\_Add

PRINT msg4, msg4len ; Display result header

PRINT dispbuff, 4 ; Display result

PRINT newline, 1 ; Newline for formatting

; Exit program - Jatin Yadav 7226

mov rax, 60 ; sys\_exit

mov rdi, 0 ; Return code 0

syscall

Succ\_Add: ; Successive Addition method - Jatin Yadav 7226

xor rax, rax ; Clear RAX

xor rbx, rbx ; Clear RBX (result)

xor rcx, rcx ; Clear RCX (counter)

mov al, [num1] ; Load first number into AL

mov cl, [num2] ; Load second number into CL (counter)

add\_loop:

test rcx, rcx ; Check if counter is zero

jz done\_succ ; If zero, exit loop

add bx, ax ; Add AX to BX (accumulate result)

dec rcx ; Decrease counter

jmp add\_loop ; Repeat

done\_succ:

call Hex\_to\_Ascii ; Convert result to ASCII

ret ; Return

Shift\_Add: ; Shift and Add method - Jatin Yadav 7226

xor rcx, rcx ; Clear RCX (result)

xor rax, rax ; Clear RAX

xor rbx, rbx ; Clear RBX

mov dx, 8 ; Counter for 8 bits

mov al, [num1] ; Load first number into AL

mov bl, [num2] ; Load second number into BL

shift\_loop:

test dx, dx ; Check if counter is zero

jz done\_shift ; If zero, exit loop

shr bl, 1 ; Shift BL right by 1 bit

jnc no\_add ; If no carry, skip addition

add cx, ax ; Add AX to CX (accumulate result)

no\_add:

shl ax, 1 ; Shift AX left by 1 (next partial product)

dec dx ; Decrease counter

jmp shift\_loop ; Repeat

done\_shift:

mov bx, cx ; Move result to BX for display

call Hex\_to\_Ascii ; Convert result to ASCII

ret ; Return

Hex\_to\_Ascii: ; Convert hex result to ASCII - Jatin Yadav 7226

mov rsi, dispbuff ; Load display buffer address

mov rcx, 4 ; Counter for 4 digits

convert\_hex:

rol bx, 4 ; Rotate BX left by 4 bits

mov al, bl ; Move lower byte to AL

and al, 0Fh ; Mask lower nibble

cmp al, 9 ; Compare with 9

jbe add\_30\_hex ; If <= 9, add 30h

add al, 7 ; If > 9, add 37h for A-F

add\_30\_hex:

add al, 30h ; Convert to ASCII

mov [rsi], al ; Store in buffer

inc rsi ; Move to next position

dec rcx ; Decrease counter

jnz convert\_hex ; Repeat until done

ret ; Return

Ascii\_to\_Hex: ; Convert ASCII input to hex - Jatin Yadav 7226

mov rsi, ascii\_num ; Load input buffer address

mov rcx, 2 ; Counter for 2 digits

xor bl, bl ; Clear BL (result)

convert\_ascii:

rol bl, 4 ; Shift BL left by 4 bits

mov al, [rsi] ; Load ASCII digit

cmp al, '9' ; Compare with '9' (39h)

jbe sub\_30\_ascii ; If <= '9', subtract 30h

sub al, 37h ; If > '9', subtract 37h (for A-F)

jmp combine

sub\_30\_ascii:

sub al, 30h ; Subtract 30h (for 0-9)

combine:

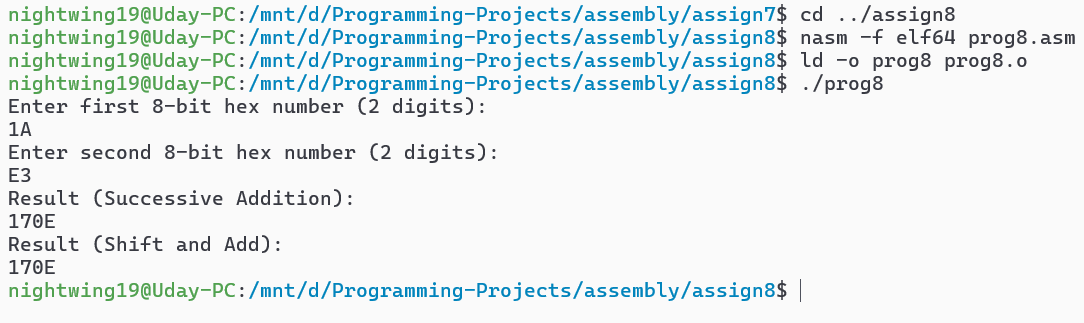
add bl, al ; Add to result

inc rsi ; Move to next digit

dec rcx ; Decrease counter

jnz convert\_ascii ; Repeat until done

ret ; Return



Macro file

; macro.asm - Macros for Linux syscalls

%macro Print 2

mov rax, 1 ; syscall: write

mov rdi, 1 ; stdout

mov rsi, %1 ; buffer

mov rdx, %2 ; length

syscall

%endmacro

%macro Accept 2

mov rax, 0 ; syscall: read

mov rdi, 0 ; stdin

mov rsi, %1

mov rdx, %2

syscall

%endmacro

%macro fopen 1

mov rax, 2 ; syscall: open

mov rdi, %1 ; filename pointer

mov rsi, 0 ; read-only

syscall

%endmacro

%macro fread 3

mov rax, 0 ; syscall: read

mov rdi, %1 ; file descriptor

mov rsi, %2 ; buffer

mov rdx, %3 ; size

syscall

%endmacro

%macro fwrite 3

mov rax, 1 ; syscall: write

mov rdi, %1 ; file descriptor

mov rsi, %2 ; buffer

mov rdx, %3 ; size

syscall

%endmacro

%macro fclose 1

mov rax, 3 ; syscall: close

mov rdi, %1

syscall

%endmacro

%macro fcreate 1

mov rax, 2 ; syscall: open

mov rdi, %1 ; filename

mov rsi, 577o ; O\_WRONLY | O\_CREAT | O\_TRUNC

mov rdx, 0644o ; permissions

syscall

%endmacro

%macro fdelete 1

mov rax, 87 ; syscall: unlink

mov rdi, %1

syscall

%endmacro

Assignment 9

;ass9.asm - Menu-driven TYPE, COPY, DELETE

;Name - Uday Pratap Singh

;Roll No: 7259

;Date: 17/04/2025

%include "macro.asm"

section .data

intro\_msg db 10,"Write X86/64 ALP to implement TYPE, COPY, DELETE \

using file operations", 10, \

intro\_len equ $-intro\_msg

msg db "------------------MENU------------------", 10

db "1. TYPE ", 10

db "2. COPY ", 10

db "3. DELETE ", 10

db "4. Exit ", 10

db "Enter your choice : "

msglen equ $-msg

endl db 10

m db "DONE!", 10

section .bss

choice resb 2

fname1 resb 50

fname2 resb 50

filehandle1 resq 1

filehandle2 resq 1

buffer resb 100

bufferlen resq 1 ; Changed to resq for 64-bit length

section .text

global \_start

\_start:

; Command-line arguments

pop rbx ; argc

pop rsi ; skip program name

; Show intro

Print intro\_msg, intro\_len

; Read first argument into fname1

mov rdi, fname1

.mark:

pop rsi

mov rdx, 0

.next:

mov al, byte [rsi + rdx]

mov [rdi + rdx], al

cmp al, 0

je .next1

inc rdx

jmp .next

.next1:

cmp rdi, fname2

je main\_menu

mov rdi, fname2

jmp .mark

main\_menu:

Print msg, msglen

Accept choice, 2

cmp byte [choice], '1'

je case1

cmp byte [choice], '2'

je case2

cmp byte [choice], '3'

je case3

cmp byte [choice], '4'

je case4

jmp main\_menu

case1:

call type

jmp main\_menu

case2:

call copy

jmp main\_menu

case3:

call delete

jmp main\_menu

case4:

mov rax, 60 ; syscall: exit

xor rdi, rdi

syscall

; TYPE implementation

type:

fopen fname1

cmp rax, -1

je case4

mov [filehandle1], rax

fread [filehandle1], buffer, 100

mov [bufferlen], rax

Print endl, 1

Print buffer, [bufferlen]

fclose [filehandle1]

ret

; COPY implementation (fixed with loop)

copy:

fopen fname1

cmp rax, -1

je case4

mov [filehandle1], rax

fcreate fname2

cmp rax, -1

je case4

mov [filehandle2], rax

.copy\_loop:

fread [filehandle1], buffer, 100

cmp rax, 0 ; EOF

je .copy\_done

mov rdi, [filehandle2]

mov rsi, buffer

mov rdx, rax

mov rax, 1 ; sys\_write

syscall

jmp .copy\_loop

.copy\_done:

fclose [filehandle1]

fclose [filehandle2]

Print m, 6

ret

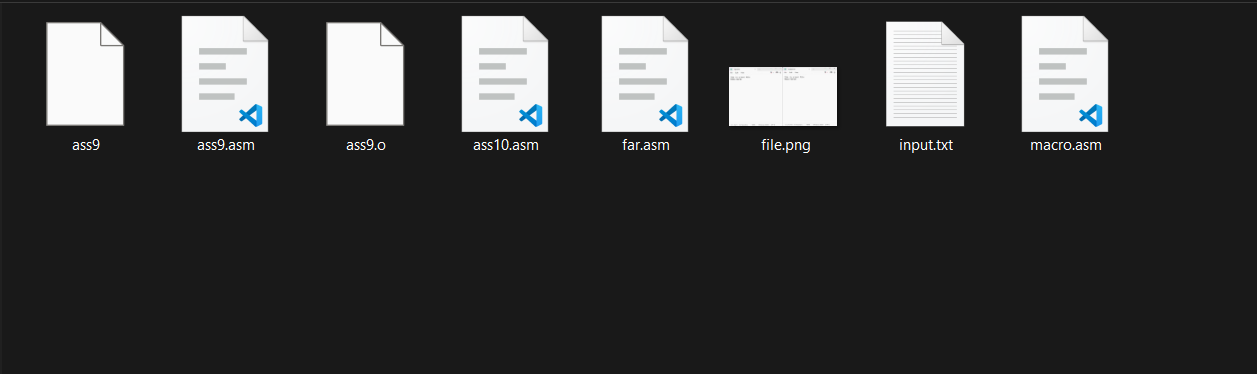
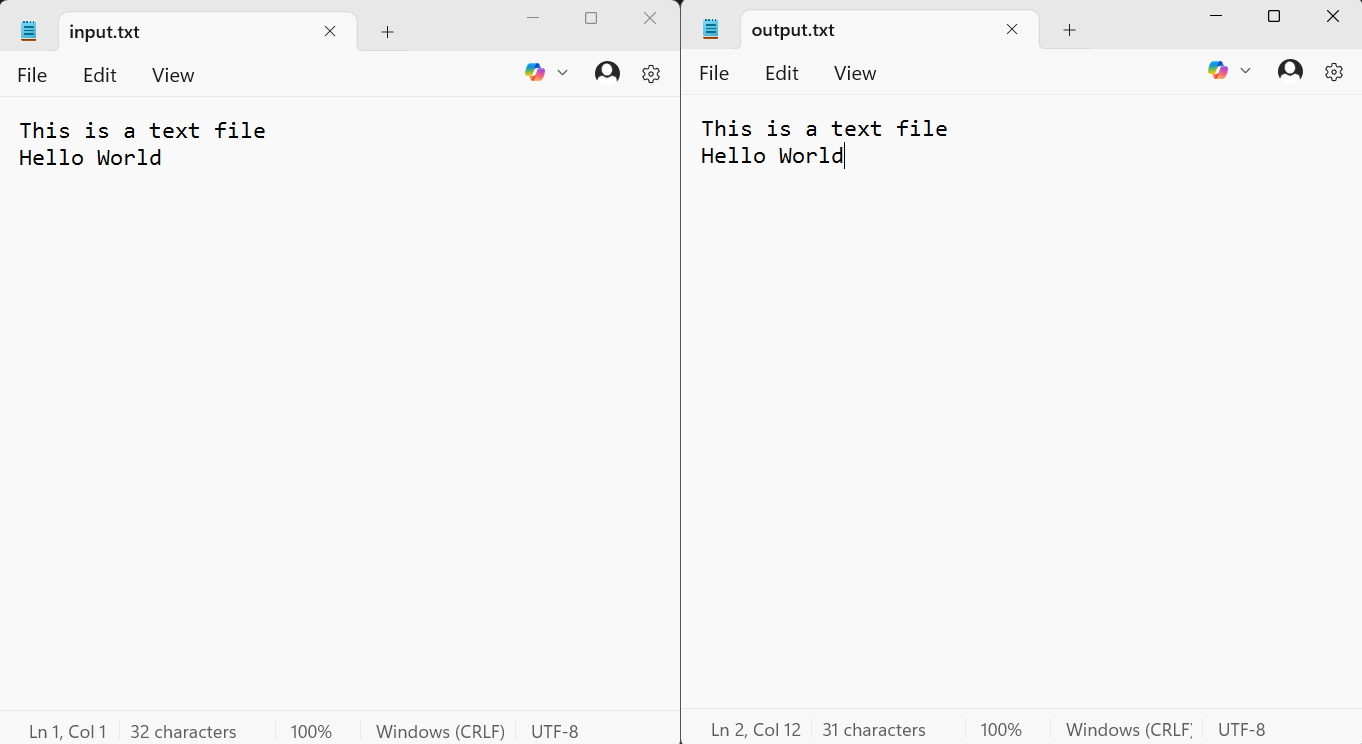
; DELETE implementation

delete:

fdelete fname2

Print m, 6

ret



Far Procedure

;far.asm

%include "macro.asm"

section .data

msg\_space db "Number of spaces: ", 0

msg\_space\_len equ $-msg\_space

msg\_line db "Number of lines: ", 0

msg\_line\_len equ $-msg\_line

msg\_char db "Number of occurrences of character: ", 0

msg\_char\_len equ $-msg\_char

dispbuff db 0, 0

nl db 10

section .bss

scount resb 1

ncount resb 1

ccount resb 1

section .text

global far\_procedure

extern buffer, buf\_len, character

far\_procedure:

xor rcx, rcx

xor rbx, rbx

xor rdx, rdx

mov rsi, buffer

mov rcx, [buf\_len]

mov bl, byte [character]

.count\_loop:

cmp rcx, 0

je display\_results

mov al, [rsi]

cmp al, 0x20

jne .check\_line

inc byte [scount]

.check\_line:

cmp al, 0x0A

jne .check\_char

inc byte [ncount]

.check\_char:

cmp al, bl

jne .next

inc byte [ccount]

.next:

inc rsi

dec rcx

jmp .count\_loop

display\_results:

; Display space count

Print msg\_space, msg\_space\_len

mov bl, [scount]

call display8num

; Display line count

Print msg\_line, msg\_line\_len

mov bl, [ncount]

call display8num

; Display character count

Print msg\_char, msg\_char\_len

mov bl, [ccount]

call display8num

ret

display8num:

mov rsi, dispbuff

mov rcx, 2

.next\_digit:

rol bl, 4

mov al, bl

and al, 0x0F

cmp al, 9

jbe .add30

add al, 0x37

jmp .store

.add30:

add al, 0x30

.store:

mov [rsi], al

inc rsi

loop .next\_digit

Print dispbuff, 2

Print nl, 1

ret

Assignment 10

;assignment 10

;Name: Uday Pratap Singh

;Roll No: 7259

;Date4: 17/04/2025

%include "macro.asm"

section .data

intro\_msg db "Write X86 ALP to find,", 10, \

"a) Number of Blank spaces", 10, \

"b) Number of lines", 10, \

"c) Occurrence of a particular character.", 10

intro\_len equ $-intro\_msg

msg1 db "Enter file name: ", 0

msg1len equ $-msg1

msg2 db "Enter character to search: ", 0

msg2len equ $-msg2

error\_msg db "Error in Opening File", 10

error\_len equ $-error\_msg

section .bss

global buffer

global buf\_len

global character

filename resb 100

character resb 2

buffer resb 1024

buf\_len resq 1

filehandle resq 1

section .text

global \_start

extern far\_procedure

\_start:

; Show assignment intro

Print intro\_msg, intro\_len

; Prompt for file name

Print msg1, msg1len

Accept filename, 100

; Replace newline with null terminator

mov rsi, filename

.find\_newline:

mov al, [rsi]

cmp al, 10

je .null\_terminate

cmp al, 0

je .after\_filename

inc rsi

jmp .find\_newline

.null\_terminate:

mov byte [rsi], 0

.after\_filename:

; Prompt for character

Print msg2, msg2len

Accept character, 2

mov byte [character+1], 0 ; Ensure null termination

; Open file

mov rax, 2 ; syscall: open

mov rdi, filename ; file name

mov rsi, 0 ; 0-READDONLY

syscall

cmp rax, -1

je open\_error

mov [filehandle], rax

; Read file into buffer

mov rdi, [filehandle]

mov rax, 0

mov rsi, buffer

mov rdx, 1024

syscall

mov [buf\_len], rax

; Close file

mov rax, 3

mov rdi, [filehandle]

syscall

; Call FAR procedure

call far\_procedure

; Exit

mov rax, 60

xor rdi, rdi

syscall

open\_error:

Print error\_msg, error\_len

mov rax, 60

mov rdi, 1

syscall

