====================================================

3) Arithmetic operations

%macro write 2

mov rax, 1;

mov rdi, 1;

mov rsi, %1;

mov rdx, %2;

syscall

%endmacro

%macro read 2

mov rax, 0;

mov rdi, 0;

mov rsi, %1;

mov rdx, %2;

syscall

%endmacro

%macro exit 0

mov rax, 60;

mov rdi, 0; syscall;

%endmacro

section .data

intro db "NAME : Ajay Kailas INgle", 10

db "prn: 202302040021", 10

introLen equ $-intro;

menu db 10,"\*MENU\*",10;

db "1. Addtion", 10;

db "2. Substraction", 10;

db "3. Multiplication", 10;

db "4. Exit", 10;

menuLen equ $-menu;

msg1 db "Enter First Number : "

msg1Len equ $-msg1;

msg2 db "Enter Second Number : "

msg2Len equ $-msg2;

msg3 db "Addition : ";

msg3Len equ $-msg3;

section .bss

num1 resb 8;

num2 resb 8;

result resb 10;

choice resb 2;

section .text

global \_start

\_start: write intro, introLen;

menuLoop:

write menu, menuLen;

read choice, 2;

cmp byte[choice], 31h;

je addition;

cmp byte[choice], 32h;

je substraction;

cmp byte[choice], 33h;

je multiplication;

cmp byte[choice], 34h;

je end;

addition:

write msg1, msg1Len;

read num1, 8;

dec rax;

mov rsi, num1;

mov rcx, rax;

mov rbx, 0;

call stringToNumber;

push rbx;

write msg2, msg2Len;

read num2, 8;

dec rax;

mov rsi, num2;

mov rcx, rax;

mov rbx, 0;

call stringToNumber;

pop rax;

add rax, rbx;

mov rbx, 10;

mov rsi, result+9;

mov rcx, 10;

call numberToString;

write msg3, msg3Len;

write result, 10;

jmp menuLoop;

substraction:

write msg1, msg1Len;

read num1, 8;

dec rax;

mov rsi, num1;

mov rcx, rax;

mov rbx, 0;

call stringToNumber;

push rbx;

write msg2, msg2Len;

read num2, 8;

dec rax;

mov rsi, num2;

mov rcx, rax;

mov rbx, 0;

call stringToNumber;

pop rax;

sub rax, rbx;

mov rbx, 10;

mov rsi, result+9;

mov rcx, 10;

call numberToString;

write msg3, msg3Len;

write result, 10;

jmp menuLoop;

multiplication:

write msg1, msg1Len;

read num1, 8;

dec rax;

mov rsi, num1;

mov rcx, rax;

mov rbx, 0;

call stringToNumber;

push rbx;

write msg2, msg2Len;

read num2, 8;

dec rax;

mov rsi, num2;

mov rcx, rax;

mov rbx, 0;

call stringToNumber;

pop rax;

mul rbx;

mov rbx, 10;

mov rsi, result+9;

mov rcx, 10;

call numberToString;

write msg3, msg3Len;

write result, 10;

jmp menuLoop;

end:

exit;

numberToString: numberLoop:

mov rdx, 0;

div rbx;

add dl, 30h;

mov [rsi], dl;

dec rsi;

dec rcx;

jnz numberLoop;

ret;

stringToNumber: stringLoop:

mov rax, 10;

mul rbx;

mov rbx, rax;

mov rdx, 0;

mov dl, byte[rsi];

sub dl, 30h;

add rbx, rdx;

inc rsi;

dec rcx;

jnz stringLoop ret;

===================================================

4) BCD to hex and hex to bcd

Code: -

%macro WRITE 02

mov rax ,1

mov rdi ,1

mov rsi ,%1

mov rdx ,%2

syscall

%endmacro

%macro READ 02

mov rax ,0

mov rdi ,0

mov rsi ,%1

mov rdx ,%2

syscall

%endmacro

section .data

msg1 db "Enter the BCD no. : ",10

len1 equ $-msg1

msg2 db "Hex equavalent is : ",10

len2 equ $-msg2

msg3 db "Enter the HEX no. : ",10

len3 equ $-msg3

msg4 db "BCD equavalent is : ",10

len4 equ $-msg4

msg5 db "Wrong choice",10

len5 equ $-msg5

menu db 10,"Ujjwal Pramod Nimbokar",10

db "PRN: 202302040007",10

db 10,"\*\*\* MENU \*\*\*",10

db"1.BCD to HEX",10

db"2.HEX to BCD",10

db"Enter your choice",10

menulen equ $-menu

section .bss

char\_buff resb 17

ans resq 1

cnt resq 01

char resb 01

choice resb 02

section .text

global \_start

\_start:

printmenu : WRITE menu,menulen

READ choice,02

cmp byte[choice],31H

je BCDtoHEX

cmp byte[choice],32H

je HEXtoBCD

cmp byte[choice],33H

je exit

WRITE msg5,len5

jmp printmenu

mov rax,60

mov rdx,00

syscall

BCDtoHEX:

WRITE msg1,len1

READ char\_buff,17

dec rax

mov rcx,rax

mov rsi,char\_buff

mov rbx,00H

up: mov rax,0AH

mul rbx

mov rbx,rax

mov rdx,00H

mov dl,byte[rsi]

sub dl,30H

add rbx,rdx

inc rsi

dec rcx

jnz up

mov[ans],rbx

WRITE msg2,len2

mov rbx,[ans]

call display

jmp \_start

HEXtoBCD:

WRITE msg3,len3

READ char\_buff,17

call accept

mov byte[cnt],00H

mov rax,rbx

up1:mov rdx,00H

mov rbx,0AH

div rbx

push rdx

inc byte[cnt]

cmp rax,00H

jne up1

WRITE msg4,len4

up2:pop rdx

add dl,30H

mov byte[char],dl

WRITE char,01

dec byte[cnt]

jnz up2

jmp \_start

exit : mov rax,60

mov rdi,00

syscall

ret

accept: dec rax

mov rcx,rax

mov rsi,char\_buff

mov rbx,00H

up4:shl rbx,04H

mov rdx,00H

mov dl,byte[rsi]

cmp dl,39H

jbe l1

sub dl,07H

l1:sub dl,30H

add rbx,rdx

inc rsi

dec rcx

jnz up4

ret

display: mov rcx,16

mov rsi,char\_buff

up3:rol rbx,04H

mov dl,bl

and dl,0FH

cmp dl,09H

jbe l2

add dl,07H

l2:add dl,30H

mov byte[rsi],dl

inc rsi

dec rcx

jnz up3

WRITE char\_buff,16

Ret

====================================================

5) Multiplication on hexadecimal on succesve right shift the practical .

%macro write 2

mov rax,1

mov rdi,1

mov rsi,%1

mov rdx,%2

syscall

%endmacro

%macro read 2

mov rax,0

mov rdi,0

mov rsi,%1

mov rdx,%2

syscall

%endmacro

section .data

msg1 db "Enter the mutiplicant",10

msg1\_len equ $-msg1

msg2 db "Enter the mutiplier",10

msg2\_len equ $-msg2

msg3 db "Multiplication Result/product =",10

msg3\_len equ $-msg3

imp\_msg db "By NABIL ANSARI",10

imp\_msg\_len equ $-imp\_msg

msg db " ",10

msg\_len equ $-msg

section .bss

num resb 17

buff resb 17

ccnt resq 1

no1 resq 1

no2 resq 1

section .text

global \_start

\_start:

write imp\_msg,imp\_msg\_len

write msg1,msg1\_len

read num,17

dec rax

mov qword[ccnt],rax

call accept ; to accept multiplicand

mov qword[no1],rbx

write msg2,msg2\_len

read num,17

dec rax

mov qword[ccnt],rax

call accept ; to accept multiplier

mov qword[no2],rbx

mov rbx,00

l1:

add rbx,qword[no1]

dec qword[no2] ; decrement multiplier by 1 e.g. decrement 4

cmp qword[no2],0 ;till 00

jne l1

write msg3,msg3\_len

call disp

write msg,msg\_len

accept:

mov rbx,0

mov rsi,num

mov rdx,00h

up1:

shl rbx,04h

mov dl,byte[rsi]

cmp dl,39h

jbe sub\_30

sub dl,07h

sub\_30:sub dl,30h

add rbx,rdx

inc rsi

dec qword[ccnt]

jnz up1

ret

disp:

mov rsi,buff

mov rcx,16

mov rdx,00

up2:

rol rbx,04

mov dl,bl

and dl,0fh

cmp dl,09

jbe mc

add dl,07h

mc:

add dl,30h

mov [rsi],dl

inc rsi

dec rcx

jnz up2

write buff,16

ret

====================================================

6) String operation

%macro WRITE 2

mov rax, 01

mov rdi, 01

mov rsi, %1

mov rdx, %2

syscall

%endmacro

%macro READ 2

mov rax, 00

mov rdi, 00

mov rsi, %1

mov rdx, %2

syscall

%endmacro

%macro EXIT 00

mov rax, 60

mov rdi, 60

syscall

%endmacro

section .data

menu db 10,"1.Length of string", 10

db "2.copy", 10

db "3.Concat", 10

db "4.Exit", 10

db "Enter Choice: ", 10

menulen equ $-menu

msg1 db "Enter 1st string: ", 10

len1 equ $-msg1

msg2 db "Enter 2nd string: ", 10

len2 equ $-msg2

msg3 db "length of string is: ", 10

len3 equ $-msg3

msg4 db "Copied string is: ", 10

len4 equ $-msg4

msg5 db "Concanated string is: ", 10

len5 equ $-msg5

msg13 db "Wrong Choice: ", 10

len13 equ $-msg13

section .bss

str1 resb 30

str2 resb 30

str3 resb 60

choice resb 02

l1 resq 1

l2 resq 1

l3 resq 1

char\_buff resb 17

actl resq 1

section .text

global \_start

\_start:

WRITE msg1, len1

READ str1, 30

dec rax

mov [l1], rax

prtmenu:WRITE menu, menulen

READ choice, 02

cmp byte[choice], 31H

je strlen

cmp byte[choice], 32H

je strcpy

cmp byte[choice], 33H

je strcat

cmp byte[choice], 34H

je exit

WRITE msg13, len13

jmp prtmenu

strlen: WRITE msg3, len3

mov rbx, [l1]

call display

jmp prtmenu

strcpy: mov rsi, str1

mov rdi, str3

mov rcx, [l1]

cld

rep movsb

WRITE msg4, len4

WRITE str3, [l1]

jmp prtmenu

strcat: WRITE msg2, len2

READ str2, 30

dec rax

mov [l2], rax

mov rsi, str1

mov rdi, str3

mov rcx, [l1]

cld

rep movsb

mov rsi, str2

mov rcx, [l2]

cld

rep movsb

mov rax, [l1]

add rax, [l2]

mov [l3], rax

WRITE msg5, len5

WRITE str3, [l3]

jmp prtmenu

exit: EXIT

display:

mov rsi,char\_buff

mov rcx,16

above:rol rbx,04H

mov dl,bl

and dl,0FH

cmp dl,09H

jbe add30

add dl,07H

add30:add dl,30H

mov byte[rsi],dl

inc rsi

dec rcx

jnz above

WRITE char\_buff,16

ret