MICROPROCESSOR ASSIGNMENTS

NAME : SHRIRANG. R. MHALGI

CLASS : S.E.

DIVISION : B

ROLL NO 222006

PROBLEM STATEMENT :

Write X86/64 ALP to perform non-overlapped and overlapped block transfer (with and without string specific instructions). Block containing data can be defined in the data segment.

CODE :

OVERLAPPING :

%macro scall 4

mov rax,%1

mov rdi,%2

mov rsi,%3

mov rdx,%4

syscall

%endmacro

;------------ data--------------------

section .data

no db "Enter number of array elements : "

no\_l equ $-no

msg db 10,"Enter the array elements ",10

msg\_l equ $-msg

menu db 10,"1: without string "

db 10,"2: with string "

db 10,"Enter your choice : "

menu\_l equ $-menu

res db 10,"Array elements after overlapping are",10

res\_l equ $-res

c db 10,"Enter the position to overlap : "

c\_l equ $-c

endl db 10," "

endl\_l equ $-endl

r db 10,"OK",10

rl equ $-r

;-------------bss-------------------------------

section .bss

count resb 4

choice resb 2

cnt1 resb 4

cnt2 resb 4

cnt3 resb 4

pos resb 4

total resb 4

arr1 resb 50

arr2 resb 50

num resb 4

char\_ans resb 4

;---------------------txt-------------------------

section .txt

global \_start

\_start:

scall 1,1,no,no\_l

scall 0,0,count,5

mov rsi,count

call accept

mov [count],dx

mov [cnt1],dx

mov [cnt2],dx

mov [cnt3],dx

; accept array elements

scall 1,1,msg,msg\_l

mov rbx,arr1

back : scall 0,0,num,5

mov rsi,num

call accept

mov [rbx],dx

add rbx,2

dec byte[count]

jnz back

scall 1,1,c,c\_l ;accept position

scall 0,0,pos,5

mov rsi,pos

call accept

mov [pos],dx

mov ax,word[cnt3]

add ax,dx

mov word[total],ax

;-------------menu------------------------

scall 1,1,menu,menu\_l

scall 0,0,choice,3

mov dx,[choice]

cmp dx,31h

je p1

cmp dx,32h

je p2

p1 : call ws

call disp\_arr

call exit

p2 : call s

call disp\_arr

call exit

;-------------------ws-------------------------------

ws: mov rsi,arr1

mov rdi,arr1

dec byte[cnt3] ;index of elements starts from 0

l1 : add rsi,2

add rdi,2

dec byte[cnt3]

jnz l1

l2 : add rdi,2

dec byte[pos]

jnz l2

l3 : mov ax,[rsi]

mov [rdi],ax

sub rsi,2

sub rdi,2

dec byte[cnt2]

jnz l3

ret

;----------------------with string-----------------------

s: mov rsi,arr1

mov rdi,arr1

dec byte[cnt3] ;index of elements starts from 0

l6 : add rsi,2

add rdi,2

dec byte[cnt3]

jnz l6

l5 : add rdi,2

dec byte[pos]

jnz l5

STD ;set direction flag as we move bottom to top

l4 : movsw

dec byte[cnt3]

jnz l4

ret

;----------------- display array-----------------------

disp\_arr:

scall 1,1,res,res\_l

mov rbx,arr1

loop : mov rax,[rbx]

call display

add rbx,2

dec byte[total]

jnz loop

ret

;------------------- accept -------------------------

accept:

mov rax,0

mov rdx,0

mov rcx,4

bk : rol dx,4

mov al,[rsi]

cmp al,39h

jle nxt

sub al,07h

nxt : sub al,30h

add dx,ax

inc rsi

dec rcx

jnz bk

ret

;------------------- display ---------------------------

display:

mov rcx,4

mov rsi,char\_ans

bk1: rol ax,4

mov dl,al

and dl,0Fh

cmp dl,09h

jle nxt1

add dl,07h

nxt1 : add dl,30h

mov [rsi],dl

inc rsi

dec rcx

jnz bk1

scall 1,1,char\_ans,4

scall 1,1,endl,endl\_l

ret

;------------------- exit ---------------------------------

exit:

mov rax,60

mov rdi,0

syscall

NONOVERLAPPING :

%macro scall 4

mov rax,%1

mov rdi,%2

mov rsi,%3

mov rdx,%4

syscall

%endmacro

;------------ data--------------------

section .data

no db "Enter number of array elements : "

no\_l equ $-no

msg db 10,"Enter the array elements ",10

msg\_l equ $-msg

menu db 10,"1: without string "

db 10,"2: with string "

db 10,"3: exit"

db 10,"Enter your choice : "

menu\_l equ $-menu

res db 10,"Destination array elements after transfer are",10

res\_l equ $-res

c db 10,"Do you want to continue ? : "

c\_l equ $-c

endl db 10," ",

endl\_l equ $-endl

;-------------bss-------------------------------

section .bss

count resb 4

choice resb 2

cnt1 resb 4

cnt2 resb 4

arr1 resb 50

arr2 resb 50

num resb 4

char\_ans resb 4

;---------------------txt-------------------------

section .txt

global \_start

\_start:

scall 1,1,no,no\_l

scall 0,0,count,5

mov rsi,count

call accept

mov [count],dx

mov [cnt1],dx

mov [cnt2],dx

; accept array elements

scall 1,1,msg,msg\_l

mov rbx,arr1

back : scall 0,0,num,5

mov rsi,num

call accept

mov [rbx],dx

add rbx,2

dec byte[count]

jnz back

;call disp\_arr

menul :scall 1,1,menu,menu\_l

scall 0,0,choice,2

cmp byte[choice],31h

je p1

cmp byte[choice],32h

je p2

cmp byte[choice],33h

je p3

p1 : call ws

call disp\_arr

call exit

p2 : call s

call disp\_arr

call exit

p3 : call exit

;--------------without string-------------------------

ws:

mov rsi,arr1

mov rdi,arr2

up : mov ax,[rsi]

mov [rdi],ax

add rsi,2

add rdi,2

dec byte[cnt2]

jnz up

ret

;--------------with string-----------------------------

s:

mov rsi,arr1

mov rdi,arr2

CLD ;clear direction flag

up1 : movsw

dec byte[cnt2]

jnz up1

ret

;----------------- display array-----------------------

disp\_arr:

scall 1,1,res,res\_l

mov rbx,arr2

loop : mov ax,[rbx]

call display

add rbx,2

dec byte[cnt1]

jnz loop

;------------------- accept -------------------------

accept:

mov rax,0

mov rdx,0

mov rcx,4

bk : rol dx,4

mov al,[rsi]

cmp al,39h

jle nxt

sub al,07h

nxt : sub al,30h

add dx,ax

inc rsi

dec rcx

jnz bk

ret

;------------------- display ---------------------------

display:

mov rcx,4

mov rsi,char\_ans

bk1: rol ax,4

mov dl,al

and dl,0Fh

cmp dl,09h

jle nxt1

add dl,07h

nxt1 : add dl,30h

mov [rsi],dl

inc rsi

dec rcx

jnz bk1

scall 1,1,char\_ans,4

scall 1,1,endl,endl\_l

ret

;------------------- exit ---------------------------------

exit:

mov rax,60

mov rdi,0

syscall

OUTPUT :

