Assembly Language Project Code

Table of Contents:


```
nclude Irvine32.inc

Include macros.inc

BUFFER_SIZE = 1000
```

Data Segment

```
-------------------------------------------------

str1 BYTE "Enter Word:",0

input BYTE 10 DUP(?)

score BYTE 0

Lives BYTE 3

check BYTE 1

word_list BYTE "FAST","APPLE","SPOT","TOUCH","SHOUT",0

word_list1 BYTE "VALUE","EMPLOYEE","SUCCESS","LAW","VIRUS",0

word_list2 BYTE "FINANCE","MONEY","REWARD","WALLET","WARE",0

arr_L1 BYTE 5 DUP(1)

arr_L2 BYTE 5 DUP(1)

arr_L3 BYTE 5 DUP(1)
```

```
file_L1 BYTE "level1.txt",0

file_L2 BYTE "level2.txt",0

file_L3 BYTE "level3.txt",0

file_L4 BYTE "instruction.txt",0

char BYTE 4 Dup("0")



;read file

buffer BYTE BUFFER_SIZE DUP(0)

fileHandle HANDLE ?



;write high score to file

filename BYTE "high_score.txt",0

stringLength DWORD ?
```

Code Segment

```
-------------------------------------------------

main proc


Again:

call clrscr

mWrite<" 1-Quick Play",0dh,0ah," 2-Instruction",0dh,0ah," 3-Setting",0dh,0ah>

mWrite<" 4-High Score",0dh,0ah, " 5-Quit",0dh,0dh,0ah,0ah>

mWrite<"Enter Choice:",0>

mov eax,0
```

```
call readdec


cmp al,1
jne next
call Quick_play
jmp quit


next:
cmp al,2
jne next1
call instruction
jmp quit


next1:
cmp al,3
jne next2
call setting
jmp quit


next2:
cmp al,4
jne next3
      mWrite<"High Score:",0>
      mov edx,offset filename
 call read_file
 call crlf
jmp quit
```

```
next3:
cmp al,5
jne next4
mov check,0
jmp Quit1

next4:
mWrite <"You Enter Invalid Number",0dh,0ah>
mov eax,500
call delay
jmp Again
quit:

call readdec
cmp check,0
jne Again

Quit1:
exit
main endp

Quick_play PROC
call clrscr
call level1
call clrscr
call level2
```

```
call clrscr

call level3

ret

Quick_play endp


setting PROC


mWrite<" 1-change Color",0dh,0ah>

Again:

mWrite<"Enter Choice:",0>


mov eax,0

call readdec


cmp al,1

jne next

call changecolor

jmp next1

next:

mWrite<"You enter Invalid number",0dh,0ah>

mov eax,500

call delay

jmp Again


next1:

ret

setting endp
```

```
changecolor PROC

call clrscr
mWrite<" 1-Blue",0dh,0ah," 2-White",0dh,0ah," 3-Green",0dh,0ah>
mWrite<" 4-Red",0dh,0ah," 5-Magenta",0dh,0ah," 6-Yellow",0dh,0ah>
mWrite<" 7-Cyan",0dh,0ah," 8-Brown",0dh,0ah>
mWrite<"Select Color:",0>
mov eax,0
call readdec

cmp al,1
jne next
mov eax,blue
call settextcolor
jmp quit

next:
cmp al,2
jne next1
mov eax,white
call settextcolor
jmp quit

next1:
cmp al,3
jne next2
```

```
mov eax,green
call settextcolor
jmp quit

next2:
cmp al,4
jne next3
mov eax,red
call settextcolor
jmp quit

next3:
cmp al,5
jne next4
mov eax,magenta
call settextcolor
jmp quit

next4:
cmp al,6
jne next5
mov eax,yellow
call settextcolor
jmp quit

next5:
cmp al,7
```

```
jne next6
mov eax,cyan
call settextcolor
jmp quit

next6:
cmp al,8
jne next7
mov eax,brown
call settextcolor
jmp quit
next7:
mWrite <"You Enter Invalid Number",0dh,0ah>
quit:

ret
changecolor endp

level1 PROC

whileloop:
      cmp lives,0
 je quit
      mWrite<"Lives:",0>
      movzx eax,lives
      call writedec
      mWrite<"     Score:",0>
```

```
     movzx  eax,score
     call WriteDec
call crlf
call crlf

mov edx,offset file_L1
call read_file
call crlf

mov edx,offset str1
call writestring

mov  edx,OFFSET input
     mov  ecx,9
     call ReadString

mov al,arr_L1[0]
cmp al,1
jne else1

cld
mov esi,offset input
mov edi,offset word_list[0]
mov ecx,4
repe cmpsb
jnZ else1
mWrite <"your enter word found",0dh,0ah>
```

```
 inc score
 mov arr_L1[0],0
 jmp next
else1:
 mov al,arr_L1[1]
 cmp al,1
 jne else2
 cld
 mov esi,offset input
 mov edi,offset word_list[4]
 mov ecx,5
 repe cmpsb
 jnz else2
 mWrite <"your enter word found",0dh,0ah>
 inc score
 mov arr_L1[1],0
 jmp next
else2:
      mov al,arr_L1[2]
 cmp al,1
 jne else3
 cld
 mov esi,offset input
 mov edi,offset word_list[9]
 mov ecx,4
 repe cmpsb
 jnz else3
```

```
 mWrite <"your enter word found",0dh,0ah>
 inc score
 mov arr_L1[2],0
 jmp next
else3:
      mov al,arr_L1[3]
 cmp al,1
 jne else4
 cld
 mov esi,offset input
 mov edi,offset word_list[13]
 mov ecx,5
 repe cmpsb
 jnz else4
 mWrite <"your enter word found",0dh,0ah>
 inc score
 mov arr_L1[3],0
 jmp next
else4:
      mov al,arr_L1[4]
 cmp al,1
 jne else5
 cld
 mov esi,offset input
 mov edi,offset word_list[18]
 mov ecx,5
 repe cmpsb
```

```
 jnz else5
 mWrite <"your enter word found",0dh,0ah>
 inc score
 mov arr_L1[4],0
 jmp next
else5:
      mWrite<"You enter word not found!",0dh,0ah>
 dec lives
next:
MOV EAX,500
CALL delay
call clrscr
MOV AL,score
cmp al,5
jl whileloop

quit:

ret
level1 endp

level2 Proc

whileloop:
      cmp lives,0
 je quit
      mWrite<"Lives:",0>
```

```
     movzx eax,lives
     call writedec
     mWrite<"     Score:",0>
     movzx  eax,score
     call WriteDec
call crlf
call crlf
mov edx,offset file_L2
call read_file
call crlf

mov edx,offset str1
call writestring

mov  edx,OFFSET input
     mov  ecx,9
     call ReadString



mov al,arr_L2[0]
cmp al,1
jne else1
cld
mov esi,offset input
mov edi,offset word_list1[0]
mov ecx,5
repe cmpsb
```

```
 jnZ else1
 mWrite <"your enter word found",0dh,0ah>
 inc score
 mov arr_L2[0],0
;call clrscr
 jmp next
else1:
 mov al,arr_L2[1]
 cmp al,1
 jne else2
 cld
 mov esi,offset input
 mov edi,offset word_list1[5]
 mov ecx,8
 repe cmpsb
 jnz else2
 mWrite <"your enter word found",0dh,0ah>
 inc score
 mov arr_L2[1],0
 jmp next
else2:
      mov al,arr_L2[2]
 cmp al,1
 jne else3
 cld
 mov esi,offset input
 mov edi,offset word_list1[13]
```

```
 mov ecx,7
 repe cmpsb
 jnz else3
 mWrite <"your enter word found",0dh,0ah>
 inc score
 mov arr_L2[2],0
 jmp next
else3:
      mov al,arr_L2[3]
 cmp al,1
 jne else4
 cld
 mov esi,offset input
 mov edi,offset word_list1[20]
 mov ecx,3
 repe cmpsb
 jnz else4
 mWrite <"your enter word found",0dh,0ah>
 inc score
 mov arr_L2[3],0
 jmp next
else4:
      mov al,arr_L2[4]
 cmp al,1
 jne else5
 cld
 mov esi,offset input
```

```
 mov edi,offset word_list1[23]
 mov ecx,5
 repe cmpsb
 jnz else5
 mWrite <"your enter word found",0dh,0ah>
 inc score
 mov arr_L2[4],0
 jmp next
else5:
     mWrite<"You enter word not found!",0dh,0ah>
dec lives
next:
MOV EAX,500
CALL delay
call clrscr
MOV AL,score
cmp al,10
jl whileloop

quit:
ret
level2 endp

level3 Proc

whileloop:
      cmp lives,0
```

```
je quit
     mWrite<"  Lives:",0>
     movzx eax,lives
     call writedec
     mWrite<"     Score:",0>
     movzx  eax,score
     call WriteDec
call crlf
call crlf
mov edx,offset file_L3
call read_file
call crlf

mov edx,offset str1
call writestring

mov  edx,OFFSET input
     mov  ecx,9
     call ReadString



mov al,arr_L3[0]
cmp al,1
jne else1
cld
mov esi,offset input
mov edi,offset word_list2[0]
```

```
 mov ecx,7
 repe cmpsb
 jnZ else1
 mWrite <"your enter word found",0dh,0ah>
 inc score
 mov arr_L3[0],0
;call clrscr
 jmp next
else1:
 mov al,arr_L3[1]
 cmp al,1
 jne else2
 cld
 mov esi,offset input
 mov edi,offset word_list2[7]
 mov ecx,5
 repe cmpsb
 jnz else2
 mWrite <"your enter word found",0dh,0ah>
 inc score
 mov arr_L3[1],0
 jmp next
else2:
      mov al,arr_L3[2]
 cmp al,1
 jne else3
 cld
```

```
 mov esi,offset input
 mov edi,offset word_list2[12]
 mov ecx,6
 repe cmpsb
 jnz else3
 mWrite <"your enter word found",0dh,0ah>
 inc score
 mov arr_L3[2],0
 jmp next
else3:
      mov al,arr_L3[3]
 cmp al,1
 jne else4
 cld
 mov esi,offset input
 mov edi,offset word_list2[18]
 mov ecx,6
 repe cmpsb
 jnz else4
 mWrite <"your enter word found",0dh,0ah>
 inc score
 mov arr_L3[3],0
 jmp next
else4:
      mov al,arr_L3[4]
 cmp al,1
 jne else5
```

```
 cld
 mov esi,offset input
 mov edi,offset word_list2[24]
 mov ecx,4
 repe cmpsb
 jnz else5
 mWrite <"your enter word found",0dh,0ah>
 inc score
 mov arr_L3[4],0
 jmp next
else5:
      mWrite<"You enter word not found!",0dh,0ah>
 dec lives
next:
MOV EAX,500
CALL delay
call clrscr
MOV AL,score
cmp al,15
jl whileloop
quit:
ret
level3 endp


read_file proc


call OpenInputFile
```

```
mov fileHandle,eax
; Check for errors.
cmp eax,INVALID_HANDLE_VALUE ; error opening file?
jne file_ok ; no: skip
mWrite <"Cannot open file",0dh,0ah>
jmp quit ; and quit
file_ok:
; Read the file into a buffer.
mov edx,OFFSET buffer
mov ecx,BUFFER_SIZE
call ReadFromFile
jnc check_buffer_size ; error reading?
mWrite "Error reading file. " ; yes: show error message
call WriteWindowsMsg
jmp close_file
check_buffer_size:
cmp eax,BUFFER_SIZE ; buffer large enough?
jb buf_size_ok ; yes
mWrite <"Error: Buffer too small for the file",0dh,0ah>
jmp quit ; and quit
buf_size_ok:
mov buffer[eax],0 ; insert null terminator
;mWrite "File size: "
;call WriteDec ; display file size
;call Crlf
; Display the buffer.
;mWrite <"Buffer:",0dh,0ah,0dh,0ah>
```

```
mov edx,OFFSET buffer ; display the buffer
call WriteString
call Crlf
close_file:
mov eax,fileHandle
call CloseFile
quit:
ret
read_file endp

instruction PROC
      mov edx,offset file_L4
 call read_file
 call crlf
ret
instruction endp

write_file PROC

; Create a new text file.
mov edx,OFFSET filename
call CreateOutputFile
mov fileHandle,eax

; Check for errors.
cmp eax, INVALID_HANDLE_VALUE ; error found?
jne file_ok ; no: skip
```

```
mWrite<"Cannot create file",0dh,0ah,0> ; display error
jmp quit
file_ok:

mov eax,0
cld
mov al,score
mov edi,offset char
stosd

; Write the buffer to the output file.
     mov eax,fileHandle
     mov edx,offset char
     mov ecx,4
     call WriteToFile
     call CloseFile

    quit:

ret
write_file endp

end main
```