Coal Project

PROGRAMMABLE KEYS

INSTRUCTOR:

Sir Kashan Hussain

GROUP MEMBERS:

SUFIYAAN USMANI (21K-3195)

QASIM HASAN (21K-3210)

YOUSUF AHMED (21K-4594)

TOOLS AND LIBRARIES USED IN PROJECT

INTRODUCTION

We are the first to create this assembly language programme from the ground up using all of the libraries that are part of Irvine32, including masm32 and kernel32.

Currently another Programmable Keys programme that can provide direct access to a variety of assembly language-designed keyboard functionality is not available.

We overcame the difficulty and developed an app for universal programmable keys on Assembly.

OBJECTIVE

Our "Programmable keys" main objective of developing is to deal with an important problem.

We are tackling the problem that our software can be used with any computer, regardless of its specifications, and that it is independent of the keyboard because it can be used with both regular desktop keyboards and keyboards that are supplied particularly to have software keys on them.

FEATURES

1. Users can assign applications and specific programmes to key ranges from A to Z using the GUI, providing them the option to arrange them anyway they feel comfortable.
2. The user of our software can access numerous default Windows settings, including volume and scroll. The list is provided in the document below.
3. Pressing (ALT+KEY) while the programme is running will bring you to the application we assigned that key to.

REFERENCES

1. As we have used main libraries in our program we are using all of them at the same time for that purpose we have to delete some similar built in code from one library so that the

call does not overlap. References were taken from online sources.

2. We have taken reference from documents online we have research about how to call command prompt will our program is running and run the path of an application/Functionality using that code.
3. To access Key ASCII value we have made a function to generate ASCII VALUES by pressing the key we will have the ASCII values this was only made to get the values so that we can hardcode it on to our program as user input it it will redirect towards that ascii value function in which the functionality has been coded by the programmer.
4. GUI for project has been taken reference from online. It is similar to a widows built in application page example task manager it has been built on the same pattern.

RESULTS USER WILL OBTAIN:

EXAMPLE:

ALT+A when pressed while the program is running will open/Redirect us to DEV C++

As with the same ALT key and second key which ranges From A to Z will be our input on which our result will be performance of certain functionality that ranges from A to Z.

CODE DESIGN

APPLICATION BEING ACCESS FROM A TO Z

	APPLICATION AVAILABLE IN THE PROGRAM RANGE (A TO Z)
A	DEV C++
B	SOUND CHECK
C	CALCULATOR
D	CHANGE VOLUME
E	MS EDGE
F	WINDOWS EXPLORER
G	CHROME
H	MUTE
I	INFO BOX
J	CLIPBOARD
K	OSK
L	TASK MANAGER
M	MUTE
N	UNMUTE
O	STORAGE DRIVE
P	PUTTY
Q	POWERPOINT
R	ORACLE
S	SNIPPING TOOL
T	GIT BASH
U	CHANGE SYSTEM VOLUME
V	VS CODE
W	WORD
X	INCREASE BRIGHTNESS
Y	ZOOM
Z	DECREASE BRIGHTNESS

LIBRARIES IN THE PROGRAM

	LIBRARIES
1	Kernel32.lib
2	User32.lib
3	Irvine32.lib
4	Macros.inc

Variables Used In Code

```
row BYTE 0
col BYTE 0
ascii BYTE ?
ebxValue DWORD ?
validFlag BYTE 0
newLine BYTE 0ah
keyA db 150 DUP(?)
```

TO

```
keyZ db 150 DUP(?)
;Here we are initializing
;variables from A to Z for
;the keys
buffer BYTE ?
bufSize DWORD ($-buffer)
keyBufSize DWORD 0
errMsg BYTE "Cannot open
file",0dh,0ah,0
logFileName  BYTE
"log.txt",0
keysFileName BYTE
"keys.txt",0
fileHandle   HANDLE ?
;handle to output file
bytesWritten DWORD ?
buffer1 BYTE BUFFER_SIZE
DUP(?)
buffSize DWORD ?
lineToWrite BYTE ?
```

METHOD WITH CODE

1. Main FUNCTION:

```
main PROC
call storeKeysToVar
;mov buffer, 0ah
;call createLog
call clearAll
call crlf
L::
mov validFlag, 0
call getKey
call isValid
movzx eax, validFlag
cmp eax, 1
jnz L
call openApp
jmp L
Invoke ExitProcess,0
main ENDP
```

Description:

This function main is from where every Function in the code will be called firstly we will call the Function storeKeystoVar after that infinite jmp while be running until we want to use the program inside we will be asking the user for the key which he want to execute to call function openApp we will also check as the key we are entering is valid or not here we will be calling isValid Function.

NOTE: All functions in main are explained in further detail down below.

2. clearAll FUNCTION:

```
clearAll PROC
mov eax, 0
mov ebx, 0
mov ecx, 0
mov edx, 0      ; we can also clear esi and edi
ret
clearAll ENDP
```

Description:

This function clears all the register it return and receive void.

3. getKey FUNCTION:

```
getKey PROC
L1:
mov  eax, 50
call Delay
call ReadKey
jz   L1
mov ascii, al
mov ebxValue, ebx
ret
getKey ENDP
```

Description:

This function reads the key input by the user. This function return the ebxValue that is the key ascii value.

4. jump FUNCTION:

```
jump PROC
mov dl, col
mov dh, row
call gotoxy
ret
jump ENDP
```

Description:

This function allow to move to specific coordinates in the console.
It revceives parameters col and row and return void.

5. *isValid FUNCTION:*

```
isValid PROC
PUSH eax
movzx eax, ascii
cmp eax, 60h
jng invalid
movzx eax, ascii
cmp eax, 7Bh
jnl invalid
mov eax, ebxValue
cmp eax, 22h
jnz invalid
mov validFlag, 1
invalid:
POP eax
ret
isValid ENDP
```

Description:

This function receives the ascii,ebxValue from the getkey function and check by comparing if value is in the correct range that is A to Z. After checking the function return 0 and 1 as in Variable validFlag.

6. ***openApp FUNCTION:***

```
openApp PROC
mov eax, 0
A:
mov al, ascii
cmp al, 61h
jnz B
mWriteLn "A",0
push OFFSET keyA
call WinExec
mov buffer, 61h
call createLog
mov buffer, 0ah
call createLog
jmp L
ret
openApp ENDP
```

Description:

This function have a switch case from A to Z we are comparing the value of the key one by one passing From A to B and B to C until the key matches the ascii value.From here we will be calling our most important functions createlog and WinExec These function will be further explored in detail bellow.

7. *createlog FUNCTION:*

```
createLog PROC
INVOKE CreateFile,
ADDR logFileName, GENERIC_WRITE, DO_NOT_SHARE, NULL,
OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0
mov fileHandle,eax                    ; save file handle
.IF eax == INVALID_HANDLE_VALUE
mov  edx,OFFSET errMsg                ; Display error message
call WriteString
jmp  QuitNow
.ENDIF
; Move the file pointer to the end of the file
INVOKE SetFilePointer,
fileHandle,0,0,FILE_END
; Append text to the file
INVOKE WriteFile,
```

```
fileHandle, ADDR buffer, bufSize,
ADDR bytesWritten, 0
INVOKE CloseHandle, fileHandle
QuitNow:
ret
createLog ENDP
```

Description:

This function create a list of all the keys we are pressing when the program is running all of it will be saved and can be viewed after the program is done running.

Here we have use built in Irvine32 Filing functions to directly open and create a file and write the value.

8. saveKeys FUNCTION:

```
saveKeys PROC
INVOKE CreateFile,
ADDR keysFileName, GENERIC_WRITE, DO_NOT_SHARE, NULL,
CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, 0
mov fileHandle,eax                    ; save file handle
.IF eax == INVALID_HANDLE_VALUE
mov  edx,OFFSET errMsg                ; Display error message
call WriteString
jmp  QuitNow
```

```
.ENDIF
mov keyBufSize, SIZEOF keyA-1
INVOKE WriteFile,              ; write text to file
fileHandle,                    ; file handle
ADDR keyA,                     ; buffer pointer
keyBufSize,                    ; number of bytes to write
ADDR bytesWritten,             ; number of bytes written
0                              ; overlapped execution flag
mov keyBufSize, SIZEOF newLine
INVOKE WriteFile,              ; write text to file
fileHandle,                    ; file handle
ADDR newLine,                  ; buffer pointer
keyBufSize,                    ; number of bytes to write
ADDR bytesWritten,             ; number of bytes written
        0                      ; overlapped execution flag
INVOKE CloseHandle, fileHandle
QuitNow:
ret
saveKeys ENDP
```

Description:

This function is written for all values A to Z the main purpose is to save keys values that the user is provided into the file which is used by Irvine built in filing commands.

LICENSE: