**National University of Computer & Emerging Sciences**

**Karachi Campus**



**Project Report**

**COAL**

**Sadist Gaming Hub**

**Sarim Uzzaman Khan; 21K-4946**

**Abdullah Javad; 21K-3154**

**BCS-3G**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Acknowledgement:**

After spending some time looking up project ideas for x86-based Intel CPUs, we finally hit upon the concept of creating a dual game combo of Tic Tac Toe and Hang Man. The majority of our project was developed as a result of learning from various internet resources, the textbook, and our lab activities. In the early days of November 2022, we set out to do this mentally demanding assignment, and we enjoyed it.

**Introduction:**

We decided to make a gaming pair and then set out to make it using the book "Assembly Language for Intel Based Computers" and online resources. The traditional games Hang Man and Tic Tac Toe can be found in our Sadist Gaming Hub.

We made the decision to design and display a real stick figure for hangman so that consumers may calculate the number of remaining attempts in order to enhance the user interface.

We designed a display for our Tic Tac Toe game that will indicate the spots where either player choose to place their moves.

**Applications**

* Teachers frequently utilise hangman to learn spelling, vocabulary, and just for fun.
* Good sportsmanship and the area of artificial intelligence that deals with the search of game trees can both be taught using tic tac toe as a teaching method.

**Code:**

INCLUDE Irvine32.inc

INCLUDE Macros.inc

INCLUDE VirtualKeys.inc

.data

Welcome BYTE "Welcome to the Sadist Gaming Hub! A masterpiece by Abdullah Javad and Sarim U. Khan",0ah

head1 BYTE " ------------------------ WELCOME TO THE SADIST GAMING HUB ------------------------",0

headerr BYTE " ",0ah

BYTE " ",0ah

BYTE " ",0ah

BYTE "-----------------------------------------------------------------------------------------------------",0ah

BYTE "| SADIST GAMING |",0ah

BYTE "| |",0ah

BYTE "| |",0ah

BYTE "| GROUP MEMBERS |",0ah

BYTE "| |",0ah

BYTE "| Abdullah Javad - 21K-3154 |",0ah

BYTE "| Sarim Uzzaman Khan - 21K-4946 |",0ah

BYTE "| |",0ah

BYTE "| |",0ah

BYTE "| PRESS 1 TO CONTINUE |",0ah

BYTE "-----------------------------------------------------------------------------------------------------",0

head2 BYTE " Choose an option to proceed",0ah

BYTE " Press 1 to play HANGMAN",0ah

BYTE " Press 2 to play TIC TAC TOE",0ah

BYTE " Press 4 to exit",0

head3 BYTE " Option chosen is: ",0

head4 BYTE " Please input a valid choice! ",0

choice BYTE ?

captionhang byte "|||||||||||||||||||||||||||Welcome To HangMan|||||||||||||||||||||||||||",0

messageofwinning BYTE "------------------------",0ah

BYTE " YOU WINNNNNNN!",0ah

BYTE " YOU GUESSED ALL THE LETTERS CORRECTLY!",0ah

BYTE"------------------------",0

messageoflosing BYTE "------------------------",0ah

BYTE " YOU DIEDDDD!",0ah

BYTE " YOU GUESSED ALL THE LETTERS WRONG!",0ah

BYTE " YOU LOST!",0ah

BYTE"------------------------",0

HANGMAN\_LIVES\_06 BYTE " ",0ah

BYTE "+------+ ",0ah

BYTE "| | ",0ah

BYTE "| ",0ah

BYTE "| ",0ah

BYTE "| ",0ah

BYTE "| ",0ah

BYTE "+------------+ ",0ah

BYTE "| | ",0ah

BYTE "+------------+ ",0

HANGMAN\_LIVES\_05 BYTE " ",0ah

BYTE "+------+ ",0ah

BYTE "| | ",0ah

BYTE "| O ",0ah

BYTE "| ",0ah

BYTE "| ",0ah

BYTE "| ",0ah

BYTE "+------------+ ",0ah

BYTE "| | ",0ah

BYTE "+------------+ ",0

HANGMAN\_LIVES\_04 BYTE " ",0ah

BYTE "+------+ ",0ah

BYTE "| | ",0ah

BYTE "| O ",0ah

BYTE "| | ",0ah

BYTE "| ",0ah

BYTE "| ",0ah

BYTE "+------------+ ",0ah

BYTE "| | ",0ah

BYTE "+------------+ ",0

HANGMAN\_LIVES\_03 BYTE " ",0ah

BYTE "+------+ ",0ah

BYTE "| | ",0ah

BYTE "| O ",0ah

BYTE "| /| ",0ah

BYTE "| ",0ah

BYTE "| ",0ah

BYTE "+------------+ ",0ah

BYTE "| | ",0ah

BYTE "+------------+ ",0

HANGMAN\_LIVES\_02 BYTE " ",0ah

BYTE "+------+ ",0ah

BYTE "| | ",0ah

BYTE "| O ",0ah

BYTE "| /|\ ",0ah

BYTE "| ",0ah

BYTE "| ",0ah

BYTE "+------------+ ",0ah

BYTE "| | ",0ah

BYTE "+------------+ ",0

HANGMAN\_LIVES\_01 BYTE " ",0ah

BYTE "+------+ ",0ah

BYTE "| | ",0ah

BYTE "| O ",0ah

BYTE "| /|\ ",0ah

BYTE "| / ",0ah

BYTE "| ",0ah

BYTE "+------------+ ",0ah

BYTE "| | ",0ah

BYTE "+------------+ ",0

HANGMAN\_LIVES\_00 BYTE " ",0ah

BYTE "+------+ ",0ah

BYTE "| | ",0ah

BYTE "| O ",0ah

BYTE "| /|\ ",0ah

BYTE "| / \ ",0ah

BYTE "| ",0ah

BYTE "+------------+ ",0ah

BYTE "| | ",0ah

BYTE "+------------+ ",0

ranNum DWORD ?

messagehang BYTE " ",0ah

BYTE " ",0ah

BYTE "OUR HANGMAN WORD GUESSER CONSISTS OF ANIMALS ",0ah

BYTE "GUESS ONLY ANIMALS",0

manyWords BYTE "DOG", 0

BYTE "FISH", 0

BYTE "FOX", 0

BYTE "LION", 0

BYTE "TIGER", 0

BYTE "SNAKE", 0

BYTE "ELEPHANT", 0

BYTE "GIRAFFE", 0

BYTE "MONKEY", 0

BYTE "ZEBRA", 0

BYTE 0 ; End of list

len equ $ - manyWords

statusGameLive DWORD ?

;Words what we select by rundom code

selectedWords BYTE " ", 0

;Use as variable in functions for length of Array

lengthArray DWORD ?

;Letter what we guess, input from keyboard

guessLetter BYTE ?

;World what we print with -------,0

guessWords BYTE 50 DUP (?)

;Array of guess Letter

guessLetterArray BYTE 50 DUP (?)

;Letter what are unknows, change with -

letterDash BYTE '-'

;tictactoe

arr dword 1,2,3

dword 4,5,6

dword 7,8,9

arr1 byte '\_','\_','\_'

byte '\_','\_','\_'

byte '\_','\_','\_',0

caption byte "|||||||||||||||||||||||||||Welcome To Tic Tac Toe|||||||||||||||||||||||||||",0

sp1 byte " ",0

st1 byte "PLAYER 1 - PLAY 'O' :",0

st2 byte "PLAYER 2 - PLAY 'X' :",0

st3 byte "Player 1 wins",0

st4 byte "Player 2 wins",0

st5 byte "INVALID INPUT",0

st6 byte "Position you entered is already occupied please select appropiate position",0

st8 byte "Match Draw !!!",0

count1 dword ?

alpha byte "abcdefghijklmnopqrstuvwyzABCDEFGHIJKLMNPQRSTUVWYZ!@#$%^&\*()0 ",0

.code

main PROC

call clrscr

mov ecx,lengthof Welcome

mov dh, 10

mov dl, 43

call gotoxy

mov dh, 12

mov dl, 36

mov esi,0

l1:

call gotoxy

mov eax, 30

call delay

mov eax, 219

mov al,Welcome[esi]

call WriteChar

inc dl

inc esi

loop l1

MOV eax, 2000

NOP

call clrscr

mov edx, offset headerr

call writestring

call readchar

cmp eax, 1

je choices

choices:

call clrscr

mov edx, OFFSET head1

call writestring

call crlf

call crlf

mov edx, OFFSET head2

call writestring

call crlf

mov edx, OFFSET head3

call writestring

call readint

cmp al, 1

JE HANGMAN

cmp al, 2

JE TICTACTOE

cmp al, 4

JE EXITING\_CODE

mov edx, OFFSET head4

call writestring

jmp choices

HANGMAN:

call clrscr

call HANGMANFUNC

TICTACTOE:

call clrscr

call TICTACTOEFUNC

EXITING\_CODE:

exit

main ENDP

HANGMANFUNC PROC

jump\_game\_start\_again:

call Crlf

mov edx, offset captionhang

call writestring

call Crlf

mov edx, offset messagehang

call writestring

mov eax,10

call Randomize

call RandomRange

mov ranNum,eax

call Crlf

mov edx, ranNum

call find\_str ;finds a word from the list of animals

INVOKE Str\_copy,

ADDR [edi],

ADDR selectedWords

call make\_array\_dash

mov statusGameLive, 6

again\_input\_world:

call print\_hangman\_live

cmp statusGameLive, 0

je loop\_game\_over

mWrite <"Guess a letter: ">

call readChar

call clrscr

and al, 0DFH ;Convert lowercase input to uppercase.

;If uppercase, it remains uppercase

push eax

sub al, 'A' ;checks if it is a letter

cmp al, 'Z'-'A'

jbe uppercase

jmp again\_input\_world

uppercase:

pop eax

mov guessLetter, al

call WriteChar

call Crlf

call Crlf

mov ecx, LENGTHOF guessLetterArray

mov edi, offset guessLetterArray

mov al, guessLetter

repne scasb

je loop\_guess\_letter\_exists

call make\_array\_guess\_letter

mov ecx, LENGTHOF selectedWords

mov edi, offset selectedWords

mov al, guessLetter

repne scasb

jne loop\_take\_live

mov esi, offset selectedWords

mov edi, offset guessWords

mov ecx, LENGTHOF selectedWords

mov al, guessLetter

xor ebx, ebx

rhl:

cmp [esi+ebx], al

jne @F

mov [edi+ebx], al

@@:

inc ebx

dec ecx

jne rhl

mov ecx, LENGTHOF guessWords

mov edi, offset guessWords

mov al, letterDash

repne scasb

jne loop\_game\_win

jmp again\_input\_world

loop\_guess\_letter\_exists:

mWrite <"Sorry, you alredy guessed the letter, ">

mov al, guessLetter

call WriteChar

call Crlf

mWrite <"I repeat you one more time the letter what you guessed. ">

call Crlf

mWrite <"Guessed letter are: ">

mov edx, offset guessLetterArray

call WriteString

call Crlf

call Crlf

jmp again\_input\_world

loop\_take\_live:

dec statusGameLive

jmp again\_input\_world

loop\_game\_win:

mov edx, offset messageofwinning

call writestring

jmp eee

loop\_game\_over:

mov edx, offset messageoflosing

call writestring

eee:

exit

HANGMANFUNC ENDP

TICTACTOEFUNC PROC

mov edx,offset caption

call writestring

call crlf

call crlf

mov ebx,0

mov ecx,4

jmp L

error:

mov edx,offset st5

call writestring

call crlf

jmp L

errorchecking1:

mov edx,offset st5

call writestring

call crlf

jmp jojo

errorchecking2:

mov edx,offset st5

call writestring

call crlf

jmp jojo1

ps:

mov edx,offset st6

call writestring

call crlf

jmp dk

pschk:

cmp arr1[esi-1],'O'

je ps

cmp arr1[esi-1],'X'

je ps

jmp jeje

ps1:

mov edx,offset st6

call writestring

call crlf

jmp jojo

pschk1:

cmp arr1[esi-1],'O'

je ps1

cmp arr1[esi-1],'X'

je ps1

jmp jeje1

ps2:

mov edx,offset st6

call writestring

call crlf

jmp jojo1

pschk2:

cmp arr1[esi-1],'O'

je ps2

cmp arr1[esi-1],'X'

je ps2

jmp jeje2

string5:

mov edx,offset st5

call writestring

call crlf

jmp L

echeck:

cmp eax,9

jg string5

cmp eax,1

jl string5

jmp rsume

string51:

mov edx,offset st5

call writestring

call crlf

jmp jojo

echeck1:

cmp eax,9

jg string51

cmp eax,1

jl string51

jmp rsume1

string52:

mov edx,offset st5

call writestring

call crlf

jmp jojo1

echeck2:

cmp eax,9

jg string52

cmp eax,1

jl string52

jmp rsume2

L:

push ecx

dk:

call Display1

call readdec

call clrscr

jmp echeck

rsume:

mov ecx,lengthof alpha

chk:

cmp al,[alpha+ebx]

je error

add ebx,TYPE alpha

loop chk

call crlf

mov esi,eax

jmp pschk

jeje:

mov arr1[esi-1],'O'

call DisplayTicTacToe

call check1

cmp count1,3

je L1

jojo:

call Display2

call readdec

call clrscr

jmp echeck1

rsume1:

mov ecx,lengthof alpha

chk1:

cmp al,[alpha+ebx]

je errorchecking1

add ebx,TYPE alpha

loop chk1

call crlf

mov esi,eax

jmp pschk1

jeje1:

mov arr1[esi-1],'X'

call DisplayTicTacToe

call check2

cmp count1,3

je L2

pop ecx

loop tmp1

jmp tmp2

tmp1:

jmp L

tmp2:

jojo1:

call crlf

call Display1

call readdec

jmp echeck2

rsume2:

mov ecx,lengthof alpha

chk2:

cmp al,[alpha+ebx]

je errorchecking2

add ebx,TYPE alpha

loop chk2

call crlf

mov esi,eax

jeje2:

mov arr1[esi-1],'O'

call DisplayTicTacToe

cmp count1,3

je L1

jmp L4

L1:

mov edx,offset st3

call writestring

call crlf

call crlf

jmp L3

L2:

mov edx,offset st4

call writestring

call crlf

call crlf

jmp L3

L4:

mov edx,offset st8

call writestring

call crlf

call crlf

L3:

exit

TICTACTOEFUNC ENDP

find\_str PROC

lea edi, manyWords ; Address of string list

mov ecx, len ; Maximal number of bytes to scan

xor al, al ; Scan for 0

@@:

sub edx, 1

jc done ; No index left to scan = string found

repne scasb ; Scan for AL

jmp @B ; Next string

done:

ret

find\_str ENDP ; RESULT: EDI pointer to string[edx]

make\_array\_dash PROC

mov edx,OFFSET selectedWords

call StrLength ; Length of a null-terminated string pointed to by EDX

mov lengthArray,eax

mov al, '-' ; Default charcter for guessWords

mov ecx, lengthArray ; REP counter

mov edi, offset guessWords ; Destination

rep stosb ; Build guessWords

mov BYTE PTR [edi], 0 ; Store the null termination

ret

make\_array\_dash ENDP

make\_array\_guess\_letter PROC

mov edx, OFFSET guessLetterArray

call StrLength ; Length of a null-terminated string pointed to by EDX

mov lengthArray, eax

mov edi, offset guessLetterArray ; Destination

add edi, lengthArray

mov al, guessLetter

mov BYTE PTR [edi], al ; Store guessLetter

inc edi

mov BYTE PTR [edi], ',' ; Store the null termination

ret

make\_array\_guess\_letter ENDP

print\_hangman\_live PROC

mov eax, statusGameLive

cmp eax, 6

je live\_6

cmp eax, 5

je live\_5

cmp eax, 4

je live\_4

cmp eax, 3

je live\_3

cmp eax, 2

je live\_2

cmp eax, 1

je live\_1

cmp eax, 0

je live\_0

live\_6:

mov edx, offset HANGMAN\_LIVES\_06

call writestring

call Crlf

call Crlf

mov edx, offset guessWords

call WriteString

call Crlf

call Crlf

mWrite <"Guessed letter are: ">

mov edx, offset guessLetterArray

call WriteString

call Crlf

call Crlf

ret

live\_5:

mov edx, offset HANGMAN\_LIVES\_05

call writestring

call Crlf

call Crlf

mov edx, offset guessWords

call WriteString

call Crlf

call Crlf

mWrite <"Guessed letter are: ">

mov edx, offset guessLetterArray

call WriteString

call Crlf

call Crlf

ret

live\_4:

mov edx, offset HANGMAN\_LIVES\_04

call writestring

call Crlf

call Crlf

mov edx, offset guessWords

call WriteString

call Crlf

call Crlf

mWrite <"Guessed letter are: ">

mov edx, offset guessLetterArray

call WriteString

call Crlf

call Crlf

ret

live\_3:

mov edx, offset HANGMAN\_LIVES\_03

call writestring

call Crlf

call Crlf

mov edx, offset guessWords

call WriteString

call Crlf

call Crlf

mWrite <"Guessed letter are: ">

mov edx, offset guessLetterArray

call WriteString

call Crlf

call Crlf

ret

live\_2:

mov edx, offset HANGMAN\_LIVES\_02

call writestring

call Crlf

call Crlf

mov edx, offset guessWords

call WriteString

call Crlf

call Crlf

mWrite <"Guessed letter are: ">

mov edx, offset guessLetterArray

call WriteString

call Crlf

call Crlf

ret

live\_1:

mov edx, offset HANGMAN\_LIVES\_01

call writestring

call Crlf

call Crlf

mov edx, offset guessWords

call WriteString

call Crlf

call Crlf

mWrite <"Guessed letter are: ">

mov edx, offset guessLetterArray

call WriteString

call Crlf

call Crlf

ret

live\_0:

mov edx, offset HANGMAN\_LIVES\_00

call writestring

call Crlf

call Crlf

mov edx, offset guessWords

call WriteString

call Crlf

call Crlf

mWrite <"Guessed letter are: ">

mov edx, offset guessLetterArray

call WriteString

call Crlf

call Crlf

ret

print\_hangman\_live ENDP

DisplayTicTacToe PROC

mov edx,offset sp1

mov ecx,3

mov esi,0

L:

mov al,arr1[esi]

call writechar

call writestring

mov al,arr1[esi+1]

call writechar

call writestring

mov al,arr1[esi+2]

call writechar

call writestring

call crlf

add esi,3

loop L

call crlf

ret

DisplayTicTacToe ENDP

Display1 PROC

mov edx,offset st1

call writestring

call crlf

call crlf

mov edx,offset sp1

mov ecx,3

mov esi,0

L:

mov eax,arr[esi]

call writedec

call writestring

mov eax,arr[esi+4]

call writedec

call writestring

mov eax,arr[esi+8]

call writedec

call writestring

call crlf

add esi,12

loop L

call crlf

ret

Display1 ENDP

Display2 PROC

mov edx,offset st2

call writestring

call crlf

call crlf

mov edx,offset sp1

mov ecx,3

mov esi,0

L:

mov eax,arr[esi]

call writedec

call writestring

mov eax,arr[esi+4]

call writedec

call writestring

mov eax,arr[esi+8]

call writedec

call writestring

call crlf

add esi,12

loop L

call crlf

ret

Display2 ENDP

Check1 PROC

mov count1,0

mov esi,0

mov ecx,3

L:

cmp arr1[esi],'O'

jne L1

inc count1

L1:

inc esi

loop L

cmp count1,3

je exit1

mov count1,0

mov esi,3

mov ecx,3

L2:

cmp arr1[esi],'O'

jne L3

inc count1

L3:

inc esi

loop L2

cmp count1,3

je exit1

mov count1,0

mov esi,6

mov ecx,3

L4:

cmp arr1[esi],'O'

jne L5

inc count1

L5:

inc esi

loop L4

cmp count1,3

je exit1

mov count1,0

mov esi,0

mov ecx,3

L6:

cmp arr1[esi],'O'

jne L7

inc count1

L7:

add esi,3

loop L6

cmp count1,3

je exit1

mov count1,0

mov esi,1

mov ecx,3

L8:

cmp arr1[esi],'O'

jne L9

inc count1

L9:

add esi,3

loop L8

cmp count1,3

je exit1

mov count1,0

mov esi,2

mov ecx,3

L10:

cmp arr1[esi],'O'

jne L11

inc count1

L11:

add esi,3

loop L10

cmp count1,3

je exit1

mov count1,0

mov esi,2

mov ecx,3

L12:

cmp arr1[esi],'O'

jne L13

inc count1

L13:

add esi,2

loop L12

cmp count1,3

je exit1

mov count1,0

mov esi,0

mov ecx,3

L14:

cmp arr1[esi],'O'

jne L15

inc count1

L15:

add esi,4

loop L14

exit1:

ret

Check1 ENDP

Check2 PROC

mov count1,0

mov esi,0

mov ecx,3

L:

cmp arr1[esi],'X'

jne L1

inc count1

L1:

inc esi

loop L

cmp count1,3

je exit1

mov count1,0

mov esi,3

mov ecx,3

L2:

cmp arr1[esi],'X'

jne L3

inc count1

L3:

inc esi

loop L2

cmp count1,3

je exit1

mov count1,0

mov esi,6

mov ecx,3

L4:

cmp arr1[esi],'X'

jne L5

inc count1

L5:

inc esi

loop L4

cmp count1,3

je exit1

mov count1,0

mov esi,0

mov ecx,3

L6:

cmp arr1[esi],'X'

jne L7

inc count1

L7:

add esi,3

loop L6

cmp count1,3

je exit1

mov count1,0

mov esi,1

mov ecx,3

L8:

cmp arr1[esi],'X'

jne L9

inc count1

L9:

add esi,3

loop L8

cmp count1,3

je exit1

mov count1,0

mov esi,2

mov ecx,3

L10:

cmp arr1[esi],'X'

jne L11

inc count1

L11:

add esi,3

loop L10

cmp count1,3

je exit1

mov count1,0

mov esi,2

mov ecx,3

L12:

cmp arr1[esi],'X'

jne L13

inc count1

L13:

add esi,2

loop L12

cmp count1,3

je exit1

mov count1,0

mov esi,0

mov ecx,3

L14:

cmp arr1[esi],'X'

jne L15

inc count1

L15:

add esi,4

loop L14

exit1:

ret

Check2 ENDP

END main

**Conclusion:**

Using the ideas from COAL and an Intel x86 processor, we were able to effectively design and implement both games. Users will be entertained by our games and their love of traditional games will be rekindled.

**References:**

[**https://github.com/SakaSerbia/Hangman---Visual-Studio-2015-with-Irvine/blob/master/AddTwo.asm**](https://github.com/SakaSerbia/Hangman---Visual-Studio-2015-with-Irvine/blob/master/AddTwo.asm)

**Assembly Language for Intel Based Processors (7th edition)**