Logo

Description automatically generated with low confidence

**EL -2003 Project Report**

**Number guessing game**

**21K-4523 | 21K-4899**

# **NUMBER GUESSING GAME**

# **Team Members**

21K-4523 Nabiha Rajani

21K-4899 Asad Tariq

# **Overview**

It is a number guessing game where the program generates a random number within the given range.

The user must guess a number within the range in a limited number of trials.

# **Description**

If the user fails to guess the number and runs out of tries, the program restarts and prompts the user for a decision.

The user may choose to exit the game, change settings or play another game, via a main menu function.

The user may select the difficulty level and the program will adjust the number of tries the user gets before the game automatically ends.

Out of range entries are detected and the user is prompted to re-enter a valid value.

# **Conclusion**

Tools used:

* Microsoft Visual Studio 2019
* Assembly language 8086

Implementation of concepts:  
Jumps and labels, stack, comparisons, loops, shifts, procedures, functions (randomize, settextcolor, etc.)

# **Code**

TITLE EL2001 Computer Architecture and Assembly Language Lab Project

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;                                              ;

;    K21-4523 Nabiha Rajani                    ;

;    K21-4899 Asad Tariq                       ;

;                                              ;

;    Number Guessing Game                      ;

;                                              ;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

include Irvine32.inc

.data

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Game Prompts                  ;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

msg byte "Chances to guess the number: ", 0

difficultyMsg byte "Select difficulty level (1/2/3) : ", 0

mainMsg byte "Enter a number between 0 and 500: ", 0

lessMsg byte "Your guess is too high!", 0

equalMsg byte "Correct - shabash!", 0

greaterMsg byte "Your guess is too low!", 0

replayMsg byte "Play again? (y/n) : ", 0

goodbyeMsg byte "Thank you for playing!", 0

errorMsg byte "Out of range  - ", 0

livesMsg byte "Lives left: ", 0

revealMsg byte "Hah! The correct answer was: ", 0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Main Menu                     ;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

start byte "1. Start Game", 0

set byte "2. Settings", 0

inst byte"3. Instructions", 0

quit byte "4. Quit Game", 0

Invalid byte "Invalid Option, you will be returned to the Main Menu", 0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Settings                      ;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Color byte "Change theme to:",  0

color1 byte "1. Springtime",  0

color2 byte "2. Bumblebee",  0

color3 byte "3. Red Velvet",  0

color4 byte "4. Light Cyan",  0

color5 byte "5. Notepad",  0

color6 byte "6. Rapunzel",  0

def byte "7. Default",  0

Diff byte "Change difficulty",  0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Welcome Screen            ;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

wMsg byte "The Number Guessing Game",  0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Instructions          ;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

ins1 byte "1. Select difficulty level as per your intelligence", 0

ins2 byte "2. Enter number between 0 and 500 ", 0

ins3 byte "3. Try to guess the number within a limited number of trials", 0

ins4 byte "4. Don't be disappointed if you fail to guess the number :(", 0

ins5 byte "5. Good Luck :) ", 0

back byte "Press any key to return to the main menu", 0

random dword ?

life dword 0

difficulty dword 3

.code

main proc

call Welcome

call MainGame

main endp

Welcome PROC

pushfd

mov dh, 5

mov dl, 39

call gotoxy

Mov eax, green + (black\*16)

call SetTextColor

mov edx, OFFSET wMsg

call writeString

mov eax, 1000

call delay

mov ecx, 5

dot:

mov al, '.'

call writeChar

mov eax, 1000

call delay

loop dot

call gotoxy

mov eax, 0

popfd

ret

Welcome endp

MainMenu PROC

pushfd

call crlf

mov edx, OFFSET start

call writeString

call crlf

mov edx, OFFSET set

call writeString

call crlf

mov edx, OFFSET inst

call writeString

call crlf

mov edx, OFFSET quit

call writeString

call crlf

popfd

ret

MainMenu ENDP

MainGame PROC

Mov eax, green + (black\*16)

call SetTextColor

mov eax, 0

Line:   ; yeh bas ek label hai

call clrscr

call MainMenu

call ReadInt

cmp eax, 1

je startGame

cmp eax, 2

je Settings

cmp eax, 3

je Instructions

cmp eax, 4

je byebye

mov edx, OFFSET Invalid

call writeString

jmp Line

Instructions:

call clrscr

mov edx, OFFSET ins1

call writeString

call crlf

mov edx, OFFSET ins2

call writeString

call crlf

mov edx, OFFSET ins3

call writeString

call crlf

mov edx, OFFSET ins4

call writeString

call crlf

mov edx, OFFSET ins5

call writeString

call crlf

call crlf

mov edx, OFFSET back

call writeString

call readChar

jmp line

Settings:

call clrscr

Mov edx, OFFSET Color

call writeString

call crlf

Mov edx, OFFSET color1

call writeString

call crlf

Mov edx, OFFSET color2

call writeString

call crlf

Mov edx, OFFSET color3

call writeString

call crlf

Mov edx, OFFSET color4

call writeString

call crlf

Mov edx, OFFSET color5

call writeString

call crlf

Mov edx, OFFSET color6

call writeString

call crlf

Mov edx, OFFSET def

call writeString

call crlf

Mov eax, 0

call ReadInt

cmp eax, 1

je gre

cmp eax, 2

je yel

cmp eax, 3

je re

cmp eax, 4

je lc

cmp eax, 5

je whi

cmp eax, 6

je lm

cmp eax, 7

je default

jmp line

gre:

mov eax, green + (yellow\*16)

call SetTextColor

jmp line

yel:

mov eax, yellow + (black\*16)

call SetTextColor

jmp line

re:

mov eax, white + (red\*16)

call SetTextColor

jmp line

lc:

mov eax, lightCyan + (black\*16)

call SetTextColor

jmp line

whi:

mov eax, black  + ( white\*16)

call SetTextColor

jmp line

lm:

mov eax, lightMagenta + (gray\*16)

call SetTextColor

jmp line

default:

mov eax, green + (black\*16)

call SetTextColor

jmp line

startGame:

call init

call crlf

;;call writedec ;;; reveal answer for testing purposes

call crlf

call getlevel

call crlf

call crlf

mov eax, difficulty

mov life, eax

L1:

mov eax, life

mov edx, offset livesMsg

call writestring

call writedec

call crlf

dec life

askMain:

mov edx, offset mainMsg

call writestring

call readint

call crlf

cmp eax, -1

jle error

cmp eax, 500

jg error

jmp continue

error:

mov edx, offset errorMsg

call writestring

jmp askMain

continue:

cmp eax, random

jl less

je equal

jg greater

Loop L1

less:

mov edx, offset greaterMsg

call writestring

call crlf

call crlf

call crlf

cmp life, 0

je quitGame

jmp L1

equal:

mov edx, offset equalMsg

call writestring

call crlf

call crlf

jmp quitGame

greater:

mov edx, offset lessMsg

call writestring

call crlf

call crlf

call crlf

cmp life, 0

je quitGame

jmp L1

quitGame:

cmp eax, random

je replay

mov edx, offset revealMsg

call writestring

mov eax, random

call writedec

call crlf

replay:

mov edx, offset replayMsg

call writestring

call readchar

cmp al, 'y'

je startGame

cmp al, 'Y'

je startGame

jmp Line

byebye:

call crlf

mov edx, offset goodbyeMsg

call writestring

call crlf

exit

MainGame endp

init PROC

pushfd

mov difficulty, 3

mov life, 0

call Randomize

mov eax, 500

call RandomRange

mov random, eax

popfd

ret

init ENDP

getlevel PROC

pushfd

askdifficulty:

mov edx, offset difficultyMsg

call writestring

call readint

cmp eax, 0

jle askdifficulty

cmp eax, 4

jge askdifficulty

cmp eax, 1

je lvl1

cmp eax, 2

je lvl2

jmp game

lvl1: shl difficulty, 1

lvl2: shl difficulty, 1

game:

mov edx, offset msg

call writestring

mov eax, difficulty

call writedec

popfd

ret

getlevel ENDP

end main