|  |  |  |  |
| --- | --- | --- | --- |
| |  |  | | --- | --- | |  | **Number Guessing Game**  **Made by:**  Muhammad Abdul Nafay (P19-0117)  Muhammad Usman (P19-0116) |   [Document title]  [Document subtitle] |  |

**INTRODUCTION:**

This is a number guessing game that has a secret number stored in system that can be altered in the program (highlighted). The player has maximum of 10 attempts to guess the number by entering a three-digit number.

As soon as the player guesses the number correctly the game will end and you will win, but if the number is not the secret number then the system will display a message according to your guess of how close your guessed number is to the secret number.

One of the following Statements will displayed according to the Difference b/w your guessed number and Secret Number:

|  |  |  |
| --- | --- | --- |
| **Statement** | **Color** | **Difference (b/w your number and the secret number)** |
| Low | Yellow | 50 |
| Very Low | Light Red | 300 |
| Extremely Low | Red | 999 |
| Out of Bounds | Red | >9999 |

|  |  |  |
| --- | --- | --- |
| **Statement** | **Color** | **Difference (b/w your number and the secret number)** |
| High | Yellow | 50 |
| Very High | Light Red | 300 |
| Extremely High | Red | 999 |
| Out of Bounds | Red | >9999 |

**Variables:**

The following are the variables which have been used to display different messages in the code:

|  |  |  |
| --- | --- | --- |
| **Variables** | **Type** | **Display** |
| msg0 | db | 'WELCOME TO THE NUMBER GUESSING GAME !' |
| by | db | 'CREATED BY:' |
| name1 | db | 'Muhammad Abdul Nafay' |
| Rollno1 | db | '19P-0117' |
| name2 | db | 'Muhammad Usman’ |
| Rollno2 | db | '19P-0116' |
| I1 | db | ‘ ----INSTRUCTIONS----' |
| I2 | db | '-- Type numbers in series and avoid spaces ' |
| I3 | db | '-- Range is from 0 to 999 ' |
| I4 | db | "-- 1 ---> TYPE ---> '001' " |
| I5 | db | "-- 50 ---> TYPE ---> '050' " |
| I6 | db | "-- 150 ---> TYPE ---> '150' " |
| I7 | db | '-- You have only 10 attempts to guess.' |
| msg1 | db | 'The Secret Number was: ' |
| msg2 | db | 'Your Number: ' |
| msg3 | db | 'High' |
| msg4 | db | 'Very High' |
| msg5 | db | 'Extremely High' |
| msg6 | db | 'Low' |
| msg7 | db | 'Very Low' |
| msg8 | db | 'Extremely Low' |
| msg9 | db | 'CONGRATULATION ! You got it !' |
| msg10 | db | 'Attempts Taken: ' |
| msg11 | db | 'Now your turn to guess...' |
| msg12 | db | 'YOU LOST !' |
| msg13 | db | "Press 'ENTER' to continue. " |
| msg14 | db | 'OUT OF BOUNDS' |

The following the variables that are used to store same values. The following are the variables and why they are used:

|  |  |  |
| --- | --- | --- |
| **Variables** | **Type** | **Purpose** |
| I0 | dw | Stores the length of msg0 (37) |
| b0 | dw | Stores the length of by (11) |
| n1 | dw | Stores the length of name1 (20) |
| n2 | dw | Stores the length of name2 (14) |
| r | dw | Stores the length of Rollno1 and Rollno2 (8) |
| IL1 | dw | Stores the length of I1 (20) |
| IL2 | dw | Stores the length of I2 (43) |
| IL3 | dw | Stores the length of I3 (25) |
| IL456 | dw | Stores the length of I4, I5 and I6 (33) |
| IL7 | dw | Stores the length of I7 (38) |
| I1 | dw | Stores the length of msg1 (23) |
| I2 | dw | Stores the length of msg2 (13) |
| I3 | dw | Stores the length of msg3 (4) |
| I4 | dw | Stores the length of msg4 (9) |
| I5 | dw | Stores the length of msg5 (14) |
| I6 | dw | Stores the length of msg6 (3) |
| I7 | dw | Stores the length of msg7 (8) |
| I8 | dw | Stores the length of msg8 (13) |
| I9 | dw | Stores the length of msg9 (29) |
| I10 | dw | Stores the length of msg10 (16) |
| I11 | dw | Stores the length of msg11 (25) |
| I12 | dw | Stores the length of msg12 (10) |
| I13 | dw | Stores the length of msg13 (27) |
| I14 | dw | Stores the length of msg14 (13) |
| guess | dw | used to take input from the user |
| limit | dw | total number of chances |
| attempts | dw | total number of attempts taken to guess the secretN0 |
| loc | dw | stores the location a every string and changes with different strings |
| temp | dw | temp variable used to store a three-digit number (digit-by-digit) |
| secretNO | dw | used to store the secret number |

**CODE EXPLANATION:**

This code uses the following FUNCTIONS: -

**CLRSCREEN:**

The code starts with the PUSH of all registers, ES storing 0xb800 through AX register, and a loop which assigns 0x0720 (which is black background and space character) to [es:di] till di reaches 4000. Finally, POP of all register followed by ret.

**PRINT:**

This function is designed to take the following parameters in the following order:

Color of the message at [bp+12]

Message at [bp+10]

Length of message at [bp+8]

Location where the message is to be printed [bp+6]

Message is moved to al, Color to ah, location to di, length to cx. Using this a loop is used which assigns [si] to al, where si is incremented, and the method of stows is used along with that.

**PRINTNUM:**

This function is designed to take the following parameters in the following order:

Number to print [bp+6]

Location [bp+4]

First the number is separated using div (by 10) and the remainder (after converted to ASCII) is PUSHED to stack and then printing it by POPING them in a loop by the same method of [es:di] of assigning to dl and color/style to dh and moving dx to [es:di] in a loop.

**CHECK:**

As usual function starts with the PUSH of all registers, dx is assigned the value 0 for being incremented later. This register is used to calculate attempts, and the register is incremented each time when the user gets the guesses wrong. Next, the secretNO being stored and the guess are being compared whether they are equal or not (by using JE [JUMP IF EQUAL], JA [JUMP ABOVE] or JB [JUMP BELOW]),

If equal then it will jump to

**FOUND:**

Which CALLS clrscreen, moves 1 to the address found and then prints msg9 and msg10 along with the required location, colors & length of message. After printing msg9 & msg10 and updating location, it ends with attempts being printed by calling the function printnum followed by POP of all registers.

If above then it will jump to

**GREATER:**

Which first subtract guess & secretNO and then the difference is compared with 50, 300 & 900.

* If the difference is smaller than 50 then a jump to G1 will be made which updates attempts (by incrementing it with 1) and printing msg3 along with its length, Color & location, after printing msg3 and updating location, it ends with POP of all registers followed by ret.
* Similarly, If the difference is smaller than 300 then it prints msg4 along with its length, Color & location, after printing updating location, it ends with POP of all registers followed by ret.
* Finally, the same thing is done to if the difference is smaller than 999 (which is every number in our case) then msg5 is printed.
* If neither jumps are used then it means that the difference is larger than 999, if that’s the case, then neither jumps are taken then the code swiftly moves to GE which prints msg14 (“OUT OF BOUNDS”).

Finally, if below then it will jump to

**SMALLER:**

Which first subtract secretNO & guess and then the difference is compared with 50, 300 & 900.

* If the difference is smaller than 50 then a jump to S1 will be made which updates attempts (by incrementing it with 1) and printing msg6 along with its length, Color & location, after printing msg6 and updating location, it ends with POP of all registers followed by ret.
* Similarly, If the difference is smaller than 300 then it prints msg7 along with its length, Color & location, after printing updating location, it ends with POP of all registers followed by ret.
* Finally, the same thing is done, if the difference is smaller than 999 (which is every number in our case) then msg8 is printed.
* If neither jumps are used then it means that the difference is larger than 999, if that’s the case, then neither jumps are taken then the code swiftly moves to SE which prints msg14 (“OUT OF BOUNDS”) as well.

A jump instruction is called to START which immediately CALLS clrscreen. After clearing all registers, the following strings are printed with its color, location & length.

Msg0, by, name1, Rollno1, name2, Rollno2, I1, I2, I7, I3, I4, I5, I6, & msg13. After putting an interrupt and calling clrscreen, the registers are cleared again to avoid any problems and msg11 is printed. After Clearing di & si registers, di is assigned [limit] and dx is assigned [attempts]. Now **IN A LOOP**, registers are cleared and the input process begins. We are taking three-digit input by:

To take the first digit, we first moved 1 to ah, then calling an interrupt for input (int 21h) which stores our input value at al, we moved that value to bl and subtracted 48 to convert it to ASCII value and multiplying it to 100. We saved that value to an address named [temp], and then we took the second value by the same process, after converting to ASCII, we multiplied this with 10 and added to [temp]. Finally, after taking 3rd digit input, converting it to ASCII and adding it to [temp] we got our 3-digit input. Moving that value to cx and pushing it along with the location, we printed it by calling printnum and also moved it to [guess]. After Clearing the registers once more, we called the function check with the input of the user [guess]. Now IF the check function returns with 1 in [found] it will directly jump to EXIT which stops the program (by int 0x21), else it moves on ahead by comparing decrementing di (limit) and comparing di to 0, IF YES then loop stops, else it doesn’t.

After the loop finishes, clrscreen is called, along with that, msg12, msg1, secretNO, msg10 & attempts are printed using print and printnum respectively.

The program ends after clearing all registers and calling interrupt (int 21h & int 0x21).