CSE-341

PROJECT REPORT

TOPIC- GUESSING GAME

SECTION-05

GROUP NO:07

GROUP NAME: IMPERIAL

GROUP MEMBERS:

- AFSANA AFRIN BORNA-17101031
- ASHFAK AHMED ANI -17101460
- SHAHRIAR SALEH FAHIM -17101455

SUBMITTED TO:

NAZMUS SAKEEF

Lecturer

Department of Computer Science And Engineering

BRAC UNIVERSITY

SUBMISSION DATE: 30th March 2020

INTRODUCTION:

One of the simplest two-player games is “Guess the number”. The first player thinks of a secret number in some known range while the second player attempts to guess the number. After each guess, the first player answers either “Higher”, “Lower” or “Correct!” depending on whether the secret number is higher, lower or equal to the guess. In this project, you will build a simple interactive program in assembly language where the computer will take the role of the first player while you play as the second player. You will interact with your program using an input field and several buttons. For this project, we will ignore the canvas and print the computer's responses in the console.

Nokia has installed the “Guessing Game” on many of its phones. The game is also available on several website, android and IOS app store as well.

METHODOLOGY:

This assembly language program is designed to create a very simple game where user is allowed to guess a hard coded number between 1 and 255. This hard coded number can be replaced by a randomly generated number using a “random number generator” for EMU8086, which is bit complex because EMU8086 do not contain an instruction to do this implicitly. Program will output if guess is higher or lower than the input number.

HISTORY OF GUESSING GAME:

First, Webster says guessing game is “a game in which the participants compete individually or in teams in the identification of something indicated obscurely. Allen points out that “A guessing game is a game in which the object is to guess some kind of information, such as a word, a phrase, a title, a number or the location of an object”. By 1997, guessing game found its way into people’s pocket, onto their Nokia phones and created the craze of mobile gaming among teenagers. The Nokia 6200 was Nokia’s first phone with guessing game and they continued to manufacture new models with the new game installed through the next decades.

Technology:

This code is written for EMU8086

Flow of the program:

System will print a message saying “Please enter a valid number:” Enter a number and *** Press ENTER *** key to continue.

The approach used to code this program made validating so easy. Because of that such features are implemented in this program. This system does NOT require user to enter a three-digit numbers.

Ex: Inputs like 32,1 are valid. They will be converted to numeric representation.

34 = 034 (Implicitly)
1 = 001 (Implicitly)

Also is users enter a value out of rage such as -1, 256…. system will print:

“Error – Number out of range!”

Finally, it asks if user wants to retry, when a correct guess is made.

DEMO RUN:

*** means ‘Enter key’ is pressed ****

```
Please enter a valid number: 0
Value is More

Please enter a valid number: 128
Value is More

Please enter a valid number: 170
Value if Less

Please enter a valid number: 255
Value if Less

Please enter a valid number: 256
Error – Number out of range!

Please enter a valid number: -1
Error – Number out of range!
```

```
Please enter a valid number: 3333
Error – Number out of range!

Please enter a valid number: 169
You have made fine Guess!

Retry [y/n] ?
```

Program Code:

```
 .model small
.stack 100h
.data

    number      db  169d   ;variable 'number' stores the random value

    ;declarations used to add LineBreak to strings
    CR          equ 13d
    LF          equ 10d

    ;String messages used through the application
    prompt      db  CR, LF,'Please enter a valid number : $'
    lessMsg     db  CR, LF,'Value if Less ','$'
    moreMsg     db  CR, LF,'Value is More ', '$'
    equalMsg    db  CR, LF,'You have made fine Guess!', '$'
    overflowMsg db  CR, LF,'Error - Number out of range!', '$'
    retry       db  CR, LF,'Retry [y/n] ? ' ,'$'

    guess       db  0d      ;variable user to store value user entered
    errorChk    db  0d      ;variable user to check if entered value is in range

    param       label Byte

.code

start:

    ; --- BEGIN resting all registers and variables to 0h
    MOV ax, 0h
    MOV bx, 0h
    MOV cx, 0h
    MOV dx, 0h

    MOV BX, OFFSET guess    ; get address of 'guess' variable in BX.
    MOV BYTE PTR [BX], 0d  ; set 'guess' to 0 (decimal)

    MOV BX, OFFSET errorChk ; get address of 'errorChk' variable in BX.
    MOV BYTE PTR [BX], 0d  ; set 'errorChk' to 0 (decimal)
    ; --- END resting

    MOV ax, @data         ; get address of data to AX
    MOV ds, ax            ; set 'data segment' to value of AX which is 'address of data'
    MOV dx, offset prompt   ; load address of 'prompt' message to DX

    MOV ah, 9h             ; Write string to STDOUT (for DOS interrupt)
    INT 21h                ; DOS INT 21h (DOS interrupt)

    MOV cl, 0h             ; set CL to 0  (Counter)
    MOV dx, 0h             ; set DX to 0  (Data register used to store user input)

; -- BEGIN reading user input
while:

    CMP     cl, 5d        ; compare CL with 10d (5 is the maximum number of digits allowed)
    JG      endwhile       ; IF CL > 5 then JUMP to 'endwhile' label

    MOV     ah, 1h         ; Read character from STDIN into AL (for DOS interrupt)
    INT     21h            ; DOS INT 21h (DOS interrupt)

    CMP     al, 0Dh        ; compare read value with 0Dh which is ASCII code for ENTER key
    JE      endwhile       ; IF AL = 0Dh, Enter key pressed, JUMP to 'endwhile'

    SUB     al, 30h        ; Substract 30h from input ASCII value to get actual number. (Because
ASCII 30h = number '0')
    MOV     dl, al         ; Move input value to DL
    PUSH    dx             ; Push DL into stack, to get it read to read next input
    INC     cl            ; Increment CL (Counter)

    JMP while              ; JUMP back to label 'while' if reached
```

```
endwhile:
; -- END reading user input

   DEC cl              ; decrement CL by one to reduce increament made in last iteration

   CMP cl, 02h             ; compare CL with 02, because only 3 numbers can be accepted as IN RANGE
   JG  overflow            ; IF CL (number of input characters) is greater than 3 JUMP to 'overflow' label

   MOV BX, OFFSET errorChk ; get address of 'errorChk' variable in BX.
   MOV BYTE PTR [BX], cl  ; set 'errorChk' to value of CL

   MOV cl, 0h              ; set CL to 0, because counter is used in next section again

; -- BEGIN processing user input

; -- Create actual NUMERIC representation of
;--   number read from user as three characters
while2:

   CMP cl,errorChk
   JG endwhile2

   POP dx              ; POP DX value stored in stack, (from least-significant-digit to most-significant-digit)

   MOV ch, 0h              ; clear CH which is used in inner loop as counter
   MOV al, 1d             ; initially set AL to 1   (decimal)
   MOV dh, 10d            ; set DH to 10  (decimal)

 ; -- BEGIN loop to create power of 10 for related possition of digit
 ; --  IF CL is 2
 ; --   1st loop will produce  10^0
 ; --   2nd loop will produce  10^1
 ; --   3rd loop will produce  10^2
 while3:

   CMP ch, cl           ; compare CH with CL
   JGE endwhile3          ; IF CH >= CL, JUMP to 'endwhile3

   MUL dh                 ; AX = AL * DH whis is = to (AL * 10)

   INC ch                 ; increment CH
   JMP while3

 endwhile3:
 ; -- END power calculation loop

   ; now AL contains 10^0, 10^1 or 10^2 depending on the value of CL

   MUL dl                 ; AX = AL * DL, which is actual positional value of number

   JO  overflow           ; If there is an overflow JUMP to 'overflow'label (for values above 300)

   MOV dl, al             ; move restlt of multiplication to DL
   ADD dl, guess          ; add result (actual positional value of number) to value in 'guess'
variable

   JC  overflow           ; If there is an overflow JUMP to 'overflow'label (for values above 255 to
300)

   MOV BX, OFFSET guess    ; get address of 'guess' variable in BX.
   MOV BYTE PTR [BX], dl  ; set 'errorChk' to value of DL

   INC cl                 ; increment CL counter

   JMP while2             ; JUMP back to label 'while2'

endwhile2:
; -- END processing user input

   MOV ax, @data          ; get address of data to AX
   MOV ds, ax             ; set 'data segment' to value of AX which is 'address of data'

   MOV dl, number         ; load original 'number' to DL
   MOV dh, guess          ; load guessed 'number' to DH

   CMP dh, dl             ; compare DH and DL (DH - DL)
   JC greater             ; if DH (GUESS) > DL (NUMBER) cmparision will cause a Carry. Becaus
of that if carry has been occured print that 'number is more'
   JE equal               ; IF DH (GUESS) = DL (NUMBER) print that guess is correct
   JG lower               ; IF DH (GUESS) < DL (NUMBER) print that number is less

equal:

   MOV dx, offset equalMsg ; load address of 'equalMsg' message to DX
   MOV ah, 9h              ; Write string to STDOUT (for DOS interrupt)
   INT 21h                ; DOS INT 21h (DOS interrupt)
   JMP exit               ; JUMP to end of the program

greater:

   MOV dx, offset moreMsg  ; load address of 'moreMsg' message to DX
   MOV ah, 9h              ; Write string to STDOUT (for DOS interrupt)
   INT 21h                ; DOS INT 21h (DOS interrupt)
   JMP start              ; JUMP to beginning of the program

lower:

   MOV dx, offset lessMsg  ; load address of 'lessMsg' message to DX
   MOV ah, 9h              ; Write string to STDOUT (for DOS interrupt)
   INT 21h                ; DOS INT 21h (DOS interrupt)
   JMP start              ; JUMP to beginning of the program

overflow:

   MOV dx, offset overflowMsg ; load address of 'overflowMsg' message to DX
   MOV ah, 9h              ; Write string to STDOUT (for DOS interrupt)
   INT 21h                ; DOS INT 21h (DOS interrupt)
   JMP start              ; JUMP to beginning of the program

exit:

; -- Ask user if he needs to try again if guess was successful
retry_while:

   MOV dx, offset retry    ; load address of 'prompt' message to DX
   MOV ah, 9h             ; Write string to STDOUT (for DOS interrupt)
   INT 21h               ; DOS INT 21h (DOS interrupt)

   MOV ah, 1h             ; Read character from STDIN into AL (for DOS interrupt)
   INT 21h               ; DOS INT 21h (DOS interrupt)

   CMP al, 6Eh            ; check if input is 'n'
   JE return_to_DOS        ; call 'return_to_DOS' label is input is 'n'

   CMP al, 79h            ; check if input is 'y'
   JE restart             ; call 'restart' label is input is 'y' ..
                       ;   "JE start" is not used because it is translated as NOP by emu8086

   JMP retry_while         ; if input is neither 'y' nor 'n' re-ask the same question

retry_endwhile:

restart:
   JMP start              ; JUMP to begining of program
return_to_DOS:
   MOV ax, 4c00h           ; Return to ms-dos
   INT 21h               ; DOS INT 21h (DOS interrupt)
   end start

RET
```

Real Life Use:

- Get Idea about real Life
- Post-traumatic stress therapy
- Exercise of brain and improvement to solving problems
- Help us to deal with the world rationally
- Help us to remove anxiety
- Idea about multitasking

- Rewire brain for happiness and positivity and so on.

Future Scope of Development:

Our guessing game worked perfectly as it followed most of the objectives. It will people if their guess is too high or too low or their guess is right. I think we can still make some improvements on our work. We can make it more complex by getting the computer to count the guesses until the person guesses the right number or a person can only guess a limited number of times and if the person fail to make right guess the program will print “YOU FAILED GAME OVER” .

Conclusion:

In conclusion, guessing game can sometimes be hectic for people participating in it if they can not guess the right number. These types of games are highly motivating because the games are amusing and entertaining.

References:

1. http://iqrometro.co.id/the-concept-of-guessing-game.html
2. https://theingots.org/community/node/18583

3. https://www.ayomaonline.com/academic/guessing-game-in-assembly/
4. https://www.slideshare.net/azharniaz3/report-on-snake-game
5. https://www.academia.edu/6393229/The_use_of_guessing_game_to_improve_students_speaking_skill