Department of IT & Computer Science

Pak-Austria Fachhochschule: Institute of Applied Sciences and Technology, Haripur, Pakistan

COMP-261 Computer Organization & Assembly Language

GUESSING GAME PROJECT

Group Members:

Azeem Mohamed Husain	Mohammed Askee	Abdhur Rahman
B22F0759CS142	B22F0758CS143	B22F0760CS141

Submitted To:

Mr.Shoaib Khan

Submitted On:

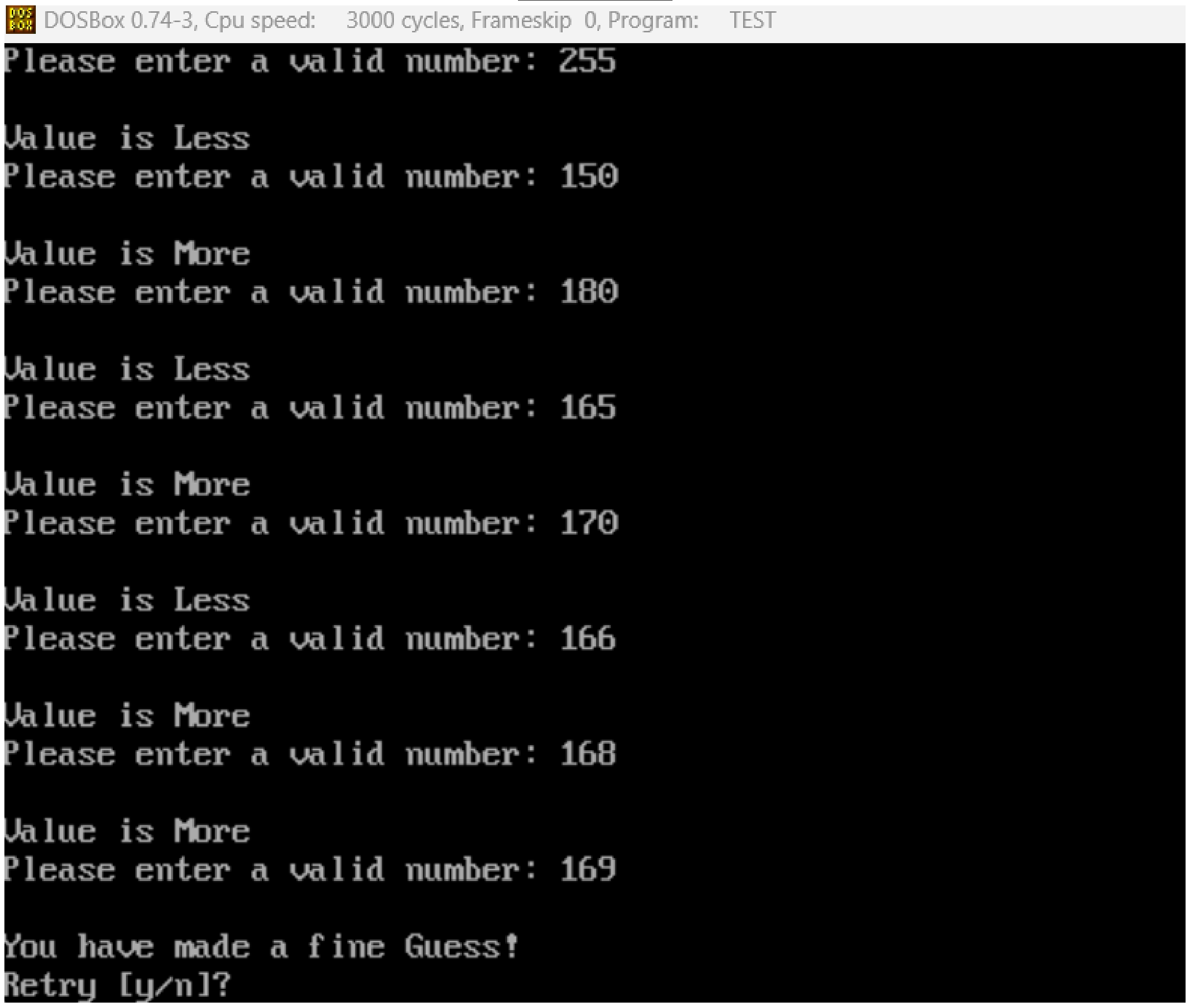
21st of May 2024

Code

```
; Author: Ali Hassan Soomro
; Date: 12/24/2017
; ****** This code is written for MASM v6.11 in DOSBox ******

.MODEL SMALL
.STACK 100h

.DATA
    number      DB  169     ; variable 'number' stores the random value

    ; Declarations used to add LineBreak to strings
    CR          EQU 13
    LF          EQU 10

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

    guess       DB  0       ; variable used to store the value entered by the
user
    errorChk    DB  0       ; variable used to check if the entered value is
in range

.CODE
start:
    ; Initialize data segment
    MOV AX, @DATA
    MOV DS, AX
    MOV ES, AX

    ; Reset all registers and variables to 0
    MOV AX, 0
    MOV BX, 0
    MOV CX, 0
    MOV DX, 0

    MOV BX, OFFSET guess    ; get address of 'guess' variable in BX
    MOV BYTE PTR [BX], 0    ; set 'guess' to 0

    MOV BX, OFFSET errorChk ; get address of 'errorChk' variable in BX
    MOV BYTE PTR [BX], 0    ; set 'errorChk' to 0

    ; Display prompt message
    MOV DX, OFFSET prompt
    MOV AH, 09h
    INT 21h

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

; Read user input
while_input:
    CMP CL, 5               ; compare CL with 5 (maximum number of digits
allowed)
    JG endwhile_input       ; if CL > 5 then jump to 'endwhile_input'

    MOV AH, 01h             ; Read character from STDIN into AL
    INT 21h

    CMP AL, 0Dh             ; compare read value with 0Dh (ENTER key)
    JE endwhile_input       ; if AL = 0Dh, jump to 'endwhile_input'

    SUB AL, 30h             ; convert ASCII to numeric
    MOV DL, AL              ; move input value to DL
    PUSH DX                ; push DL into stack
    INC CL                  ; increment CL (Counter)

    JMP while_input         ; jump back to 'while_input'

endwhile_input:
    DEC CL                  ; decrement CL by one

    CMP CL, 2               ; compare CL with 2 (only 3 numbers accepted as in
range)
    JG overflow             ; if CL > 2 jump to 'overflow'

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

    MOV CL, 0               ; reset CL to 0

; Process user input
while_process:
    CMP CL, errorChk
    JG endwhile_process

    POP DX                  ; POP DX value from stack

    MOV CH, 0               ; clear CH (counter)
    MOV AL, 1               ; set AL to 1
    MOV DH, 10              ; set DH to 10

; Calculate power of 10 for the position
while_power:
    CMP CH, CL
    JGE endwhile_power

    MUL DH                  ; AL = AL * 10
    INC CH                  ; increment CH
    JMP while_power

endwhile_power:
    MUL DL                  ; AX = AL * DL (positional value of digit)

    JO overflow             ; if overflow occurs, jump to 'overflow'

    MOV DL, AL              ; move result to DL
    ADD DL, guess           ; add to 'guess' variable

    JC overflow             ; if overflow occurs, jump to 'overflow'

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

    INC CL                  ; increment CL counter
    JMP while_process       ; jump back to 'while_process'

endwhile_process:
    ; Compare guessed number with stored number
    MOV DL, number          ; load original 'number' to DL
    MOV DH, guess           ; load guessed 'number' to DH

    CMP DH, DL              ; compare DH and DL
    JC greater              ; if DH < DL, jump to 'greater'
    JE equal                ; if DH = DL, jump to 'equal'
    JMP lower               ; otherwise, jump to 'lower'

equal:
    MOV DX, OFFSET equalMsg ; display equal message
    MOV AH, 09h
    INT 21h
    JMP retry_prompt        ; prompt for retry

greater:
    MOV DX, OFFSET moreMsg  ; display more message
    MOV AH, 09h
    INT 21h
    JMP start               ; restart the program
```

```
lower:
    MOV DX, OFFSET lessMsg  ; display less message
    MOV AH, 09h
    INT 21h
    JMP start               ; restart the program

overflow:
    MOV DX, OFFSET overflowMsg ; display overflow message
    MOV AH, 09h
    INT 21h
    JMP start               ; restart the program

retry_prompt:
    MOV DX, OFFSET retry    ; display retry message
    MOV AH, 09h
    INT 21h

retry_while:
    MOV AH, 01h             ; read user input
    INT 21h

    CMP AL, 'n'             ; check if input is 'n'
    JE return_to_DOS        ; if 'n', return to DOS

    CMP AL, 'y'             ; check if input is 'y'
    JE start                ; if 'y', restart the program

    JMP retry_while         ; otherwise, prompt again

return_to_DOS:
    MOV AX, 4C00h           ; return to DOS
    INT 21h

END start
```

Output


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TEST

```
Please enter a valid number: 255

Value is Less
Please enter a valid number: 150

Value is More
Please enter a valid number: 180

Value is Less
Please enter a valid number: 165

Value is More
Please enter a valid number: 170

Value is Less
Please enter a valid number: 166

Value is More
Please enter a valid number: 168

Value is More
Please enter a valid number: 169

You have made a fine Guess!
Retry [y/n]?
```