

coal lab

FINAL SEMESTER PROJECT



December 7, 2022

Rai saqlain

F21-9255

RAI MUHAMMAD SAQLAIN

ROLL NO 21F-9255

BCS-3E

**GUESSING GAME PROJECT**

**CODE :**

**dosseg**

**.model small**

**.stack 100h**

**.data**

**number db 150d ;we have variable number stores the random value**

**CR equ 13d ;This is used to add LineBreak to the strings**

**LF equ 10d ;This is used to add LineBreak to the strings**

**;String messages which appera when Game run**

**prompt db CR, LF,'Please Enter a number : $'**

**lessMsg db CR, LF,'Value is Less ','$'**

**moreMsg db CR, LF,'Value is More ', '$'**

**equalMsg db CR, LF,'congratulations! you won the Game ', '$'**

**overflowMsg db CR, LF,'Invalid number! Input again ', '$'**

**Againplay db CR, LF,'will you play game again if yes then press (y) if not then press(n) ? ' ,'$'**

**guess db 0d ;to store the user enter value**

**check db 0d ;This is check for user enter number is in range**

**param label Byte**

**.code**

**start:**

**; intialize with all register with 0**

**MOV ax, 0h**

**MOV bx, 0h**

**MOV cx, 0h**

**MOV dx, 0h**

**MOV BX, OFFSET guess ; We get the adddress of guess**

**MOV BYTE PTR [BX], 0d ; we set guess to 0 in decimal number**

**MOV BX, OFFSET check ;we get the address of errorChk variable in BX.**

**MOV BYTE PTR [BX], 0d ;we set errorChk to 0 in decimal number**

**; END resting**

**MOV ax, @data ; we get address of data to AX**

**MOV ds, ax ; we 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 ; for string print**

**INT 21h ; DOS interrupt**

**MOV cl, 0h ; set Counter 0**

**MOV dx, 0h ; Data register used to store user input**

**;read All user input**

**LOOP:**

**CMP cl, 5d ; compare CL with 10d because 5 is the maximum number of digits allowed to write**

**JG ENDLOOP ; 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 ENDLOOP ; IF AL = 0Dh, Enter key pressed, JUMP to 'endwhile'**

**SUB al, 30h ; Substract 30h from input ASCII value to get actual number**

**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 counter**

**JMP LOOP ; JUMP back to label 'while' if reached**

**ENDLOOP:**

**;END read user input**

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

**CMP cl, 02h ; compare CL with 02 for range check 3 numbers allowed**

**JG overflow ; IF CL is greater than 3 JUMP to overflow label**

**MOV BX, OFFSET check ; 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**

**LOOP2:**

**CMP cl,check**

**JG ENDLOOP2**

**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)**

**LOOP3:**

**CMP ch, cl ; compare CH with CL**

**JGE ENDLOOP3 ; IF CH >= CL, JUMP to 'endwhile3**

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

**INC ch ; increment CH**

**JMP LOOP3**

**ENDLOOP3:**

**;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 LOOP2 ; JUMP back to label 'while2'**

**ENDLOOP2:**

**;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 End ; JUMP to end of the program**

**greater:**

**MOV dx, offset lessMsg ; 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 moreMsg ; 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**

**END:**

**;Ask user if he needs to try again if guess was successful**

**Againplay\_LOOP:**

**MOV dx, offset Againplay ; 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 Again ; call 'restart' label is input is 'y' ..**

**; "JE start" is not used because it is translated as NOP by emu8086**

**JMP Againplay\_LOOP ; if input is neither 'y' nor 'n' re-ask the same question**

**Againplay\_ENDLOOP:**

**Again:**

**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**

**OUTPUT:**

