****

Name: Muhammad Jawwad Shah

ID: CSC-18F-140

Course: Computer Organization and Assembly language

Submit To: Miss Saher Jumani

**ACKNOWLEDGEMENT**

The success and final outcome of this project required a lot of guidance and assistance from my group members and teachers and I am extremely privileged to have got this all along the completion of my project. All that I have done is only due to such supervision and assistance and I would not forget to thank them.

I respect and thank ( Miss Saher ) for providing me an opportunity to do the project work and giving us all support and guidance which made me complete the project duly.

**Assembly Language Based Calculator**

**Introduction:**

**Assembly language:**

Each personal computer has a microprocessor that manages the computer's arithmetical, logical, and control activities.

Each family of processors has its own set of instructions for handling various operations such as getting input from keyboard, displaying information on screen and performing various other jobs. These set of instructions are called 'machine language instructions'.

A processor understands only machine language instructions, which are strings of 1's and 0's. However, machine language is too obscure and complex for using in software development. So, the low-level assembly language is designed for a specific family of processors that represents various instructions in symbolic code and a more understandable form.

**Working Concept :**

* The Assembly language program ( Word hunt ) can be used to perform task related to puzzle to find the correct word
  + Playing as a game
  + Hunt the new word
  + Correct word having color
* Playing games with assembly language is Interest for the user
* Provide the color for the correct word

**Code Of Program** :

.model small

.stack 100h

.data

; add your data here!

seed dw 2332h

rndnum dw 0

line db 0

column db 0

page\_number db 0

word\_1 db "ROBERTO$"

word\_2 db "HOUSE$"

word\_3 db "SCHOOL$"

word\_4 db "COMPUTER$"

word\_5 db "FRIEND$"

word\_pointer dw " $"

.code

start:

; set segment registers:

mov ax, @data

mov ds, ax

mov es, ax

main\_loop:

call set\_word\_pointer

call print\_board

mov cx, 5

mov si, 0 ; index for word\_pointer

mov di, 0 ; word direction

word\_print\_loop:

; column

mov dh, 30

mov al, 00011111b

call sort\_number

; sort only 30 first columns and multuply by 2, the result is only even columns

mov al, dl

mov bl, 2

mul bl

mov dl, al

mov column, dl

; line

mov dh, 14

mov al, 00011111b

call sort\_number

mov line, dl

; print

call print\_word

xor di, 00000001b

add si, 2

loop word\_print\_loop

call end\_check

call clear\_screen

jmp main\_loop

fim:

mov ax, 4c00h ; exit to operating system.

int 21h

sort\_number: ; sort a number between 0 and DH; XOR with AL; store in dl

call random

mov dl, byte ptr rndnum

and dl, al

cmp dl, dh

jg sort\_number

ret

print\_word: ; print word stored in SI; DI:0 = vertical, DI:1 = horizontal

pushf

push ax

push dx

push si

push di

call set\_cursor

mov si, word\_pointer[si]

do\_print\_word:

mov ax, ds:[si]

cmp al, "$"

je end\_print\_word

call print\_custon\_character

inc si

cmp di, 0

je inc\_line

cmp di, 1

je inc\_column

return\_check\_position:

call set\_cursor

; if is horizontal, print blanck space

cmp di, 1

je print\_blanck\_space

return\_print\_blanck\_space:

jmp do\_print\_word

end\_print\_word:

pop di

pop si

pop dx

pop ax

popf

ret

inc\_line:

inc line

jmp return\_check\_position

inc\_column:

inc column

jmp return\_check\_position

print\_blanck\_space:

mov al, " "

call print\_custon\_character

inc column

call set\_cursor

jmp return\_print\_blanck\_space

end\_check:

pushf

push ax

mov ah, 1

int 21h

cmp al, "E"

je fim

cmp al, "e"

je fim

pop ax

popf

ret

clear\_screen: ; get and set video mode

pushf

push ax

mov ah, 0fh

int 10h

mov ah, 0

int 10h

pop ax

popf

ret

set\_word\_pointer:

mov word\_pointer[0], offset word\_1

mov word\_pointer[2], offset word\_2

mov word\_pointer[4], offset word\_3

mov word\_pointer[6], offset word\_4

mov word\_pointer[8], offset word\_5

ret

print\_custon\_character: ; print content stored in AL

pushf

push ax

push bx

push cx

mov ah, 09h

mov bh, page\_number

mov bl, 10 ; color

mov cx, 1 ;number of times to print

int 10h

pop cx

pop bx

pop ax

popf

ret

print\_board:

pushf

push cx

push dx

mov cx, 24 \* 40

print\_board\_loop:

; get new letter

call random

mov dl, byte ptr rndnum

and dl, 00011111b

add dl, "A"

; check if is letter

cmp dl, "A"

jl print\_board\_loop

cmp dl, "Z"

jg print\_board\_loop

; print letter

call print\_character

mov dl, " "

call print\_character

loop print\_board\_loop

pop dx

pop cx

popf

ret

print\_character: ; print character stored in DL

pushf

push ax

mov ah, 2

int 21h

pop ax

popf

ret

random:

pushf

push ax

push cx

push dx

mov ah, 2ch

int 21h

; segundos em dh

; usa tambem cx

mov ax, seed ; ax = seed

add al, dh

mov dx, 8405h ; dx = 8405h

mul dx ; mul (8405h \* seed) into dword dx:ax

cmp ax, seed

jnz gotseed ; if new seed = old seed, alter seed

mov ah, dl

inc ax

gotseed:

mov seed, ax ; we have a new seed, so store it

mov ax, dx ; al = random number

mov rndnum, ax

pop dx

pop cx

pop ax

popf

ret

set\_cursor:

pushf

push ax

push bx

push dx

mov ah, 2

mov bh, page\_number

mov dh, line

mov dl, column

int 10h

pop dx

pop bx

pop ax

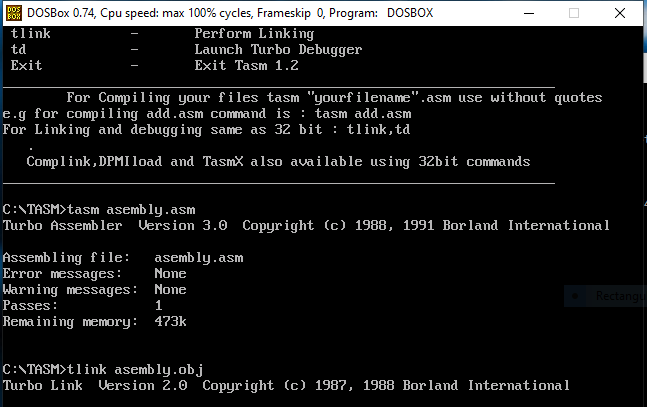
popf

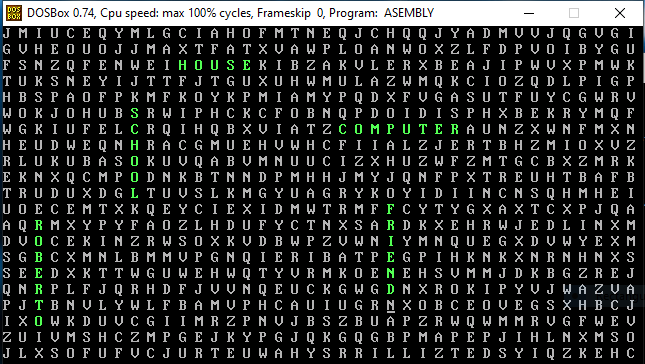
ret

ends

end start

**Output:**

****

****