National University of Computer and Emerging Sciences



Coal Semester Project

Coal

|  |  |
| --- | --- |
| Names | Danish Ahmad (19F0170)  Muhammad Zain (19F0228)  Muhammad Umar Rathore (19F0328) |
| INSTRUCTOR | Mr. Abuzar Ghafari |
| Semester | Fall 2020 |

2048 Game

# Source Code:

Include Irvine32.inc

.data

;=========================================================================================

;Title

Line1 DB " \_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_ \_\_\_ \_\_\_\_\_\_ ",0

Line2 DB " / \_\_ \ / \_\_ \ | | | | / \_\_ \ ",0

Line3 DB " |\_| | | | | | | | |\_\_\_| | | | | | ",0

Line4 DB " \_\_\_\_| | | | | | |\_\_\_\_\_ | | |\_\_| | ",0

Line5 DB " / \_\_\_\_/ | | | | | | | | | | ",0

Line6 DB " | |\_\_\_\_ | |\_\_| | | | | |\_\_| | ",0

Line7 DB " |\_\_\_\_\_\_\_| \\_\_\_\_\_\_/ |\_| \\_\_\_\_\_\_/ ",0

;===========================================================================================

;Rules

Rule db "Rules",0

Rule1 db "There are 16 tiles in a 4 × 4 grid.",0

Rule2 db "Merge the similar tiles by moving them in any of the four directions to make bigger tiles.",0

Rule3 db "After each move, a new tile appears at random empty position with a value of either 2 or 4.",0

Rule4 db "Game terminates when all the boxes are filled and there are no moves that can merge the tiles,",0

Rule5 db "or you create a tile with a value of 2048.",0

;===========================================================================================

;How to play

HowPlay db "How to Play",0

HowPlay1 db " 1) 2048 is an addictive puzzle game based on simple addition.",0

HowPlay2 db " 2) The objective of the game is to reach 2048 by merging adjacent similar number tiles on a 4x4 Grid.",0

HowPlay3 db " 3) The game starts with two tiles of 2 random positions on the Grid ",0

HowPlay4 db " and continues to add new tiles of 2 on random tiles after every move.",0

HowPlay5 db " 4) Game ends when Grid is completely filled with numbers and you donot have any move left or when 2048 appears ",0

;==========================================================================================

;Credits

Credits\_1 db "Programmed by: ",0

Credits\_2 db "Danish Ahmad (19F-0170)",0

Credits\_3 db "Muhammad Umar (19F-0328) ",0

Credits\_4 db "Muhammad Zain (19F-0228) ",0

;====================================================================================

;Menu

;implementation starts on line # 305 in MENU PROC

Menu\_M0 db " \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_",0

Menu\_M1 db "| Menu : |",0

Menu\_M2 db "| 1. Play Game |",0

Menu\_M3 db "| 2. Credits |",0

Menu\_M4 db "| 3. How to play |",0

Menu\_M5 db "| 4. Game Rules |",0

Menu\_M6 db "| 5. To Exit |",0

Menu\_M7 db "|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|",0

counter db 0

;Gameplay instructions

Msg1 db " ^ Moves : ",0

Msg2 db " < > Score : ",0

Msg3 db " v ",0

Msg4 db " 'Esc' to Endgame ",0

Msg5 db " Use Arrow Keys" ,0

Win\_Msg db "HURRAYY!! 'YOU WON!!' :-D",0

Lose\_Msg db "ALAS!! 'YOU LOST!!' :-(",0

RowNo dd 0

ColNo dd 0

Menu\_input db ?

;===================================================================================

;MAIN Board

Grid word 16 dup(0)

Backup\_Grid word 16 dup(0)

boardSize = $-Backup\_Grid

bytesWritten dword ?

temp word 4 dup(0) ; used in movement and removing zeroes

temp\_EAX dword 0

ranNum byte 0

Moves\_Counter dd 0

Underline db "===========================================",0

Space db " ",0

Escc db "011Bh",0

Score dd ?

;========================================================================================

.code

main proc

call Run\_Game

exit

main ENDP

;====================================================

Run\_Game Proc

;Recieves: Nothing

;Description:Runs game And Call the Menu Function

;Returns: Nothing

call Title\_2048

call Loading

mov esi,offset Grid

mov edi,offset Backup\_Grid

mov ecx,16

Copy\_Loop:

lodsw

stosw

LOOP Copy\_Loop

call Menu

ret

RUN\_Game ENDP

;====================================================

Title\_2048 proc

;Recieves: Nothing

;Description: Display The Logo " 2048 "

;Returns: Nothing

mov eax , 0h

mov AL , 0Bh

call setTextColor ;Set Black background and Blue text (Title)

mov edx ,offset Line1

call writestring

call crlf

mov edx ,offset Line2

call writestring

call crlf

mov edx ,offset Line3

call writestring

call crlf

mov edx ,offset Line4

call writestring

call crlf

mov edx ,offset Line5

call writestring

call crlf

mov edx ,offset Line6

call writestring

call crlf

mov edx ,offset Line7

call writestring

call crlf

ret

Title\_2048 endp

;====================================================

Loading PROC

;Recieves: Nothing

;Description:Print Loading Screen

;Returns: Nothing

call crlf

call crlf

mov eax, white + (lightred \* 16)

call SetTextColor

mov dh,10

mov dl,40

mov ecx,12

mov counter,0

L1:

call Gotoxy

.if(counter>=0&&counter<=12)

mov eax," "

mov eax," "

call writechar

.endif

mov eax,500

call Delay

add dl,2

add counter,1

loop L1

call crlf

mov ecx ,0

ret

Loading endp

;====================================================

Menu PROC

;Recieves: Nothing

;Description: Print The Menu And get input and then call the respective Function

;Returns: Option For Selection In AL

call clrscr

M:

mov eax , 0h

mov AL , 0Bh

call setTextColor

mov edx , offset Space

call writestring

mov edx , offset Menu\_M0

call writestring

call crlf

call crlf

mov edx , offset Space

call writestring

mov edx , offset Menu\_M1

call writestring

call crlf

mov edx , offset Space

call writestring

mov edx , offset Menu\_M2

call writestring

call crlf

mov edx , offset Space

call writestring

mov edx , offset Menu\_M3

call writestring

call crlf

mov edx , offset Space

call writestring

mov edx , offset Menu\_M4

call writestring

call crlf

mov edx , offset Space

call writestring

mov edx , offset Menu\_M5

call writestring

call crlf

mov edx , offset Space

call writestring

mov edx , offset Menu\_M6

call writestring

call crlf

call crlf

mov edx , offset Space

call writestring

mov edx , offset Menu\_M7

call writestring

call crlf

mov edx, offset Space

call writestring

call readchar

mov Menu\_input,al

.if(AL == '1')

call Play

JMP M

.elseif(AL == '2')

call credits

JMP M

.elseif(AL == '3')

call HowtoPlay

JMP M

.elseif(AL == '4')

call Rules

JMP M

.elseif(AL == '5')

Exit

.else

JMP M

.endif

ret

Menu endp

;===================================================

Credits PROC

;Recieves: Nothing

;Description:Print Credits

;Returns: Nothing

mov eax, white + (cyan \* 16)

call setTextColor

mov edx , offset Credits\_1

call writestring

call crlf

mov edx , offset Credits\_2

call writestring

call crlf

mov edx , offset Credits\_3

call writestring

call crlf

mov edx , offset Credits\_4

call writestring

call waitmsg

ret

Credits endp

;===================================================

HowtoPlay PROC

;Recieves: Nothing

;Description:Print Instruction To How to Play game

;Returns: Nothing

mov eax , 0h

mov AL , 0Bh

call setTextColor

mov edx , offset Space

call writestring

mov edx , offset HowPlay

call writestring

call crlf

mov edx , offset Space

call writestring

mov edx , offset HowPlay1

call writestring

call crlf

mov edx , offset Space

call writestring

mov edx , offset HowPlay2

call writestring

call crlf

mov edx , offset Space

call writestring

mov edx , offset HowPlay3

call writestring

call crlf

mov edx , offset Space

call writestring

mov edx , offset HowPlay4

call writestring

call crlf

mov edx , offset Space

call writestring

mov edx , offset HowPlay5

call writestring

call crlf

call waitmsg

ret

HowtoPlay endp

;==================================================

Rules PROC

;Recieves: Nothing

;Description:Print Rules

;Returns: Nothing

mov eax , 0h

mov AL , 0Bh

call setTextColor

mov edx , offset Space

call writestring

mov edx , offset Rule

call writestring

call crlf

mov edx , offset Space

call writestring

mov edx , offset Rule1

call writestring

call crlf

mov edx , offset Space

call writestring

mov edx , offset Rule2

call writestring

call crlf

mov edx , offset Space

call writestring

mov edx , offset Rule3

call writestring

call crlf

call crlf

call waitmsg

ret

Rules endp

;==================================================

Play PROC

;Recieves: Nothing

;Description:Start the Game By Calling Proc Game\_Start and Perform Operation By calling Proc PerformOper

;Returns: Nothing

call clrscr

call Game\_Start ;initializes board and moves with new game values

call performOper ; main function that reads char and performs operations like left right etc

ret

Play endp

;===================================================

Game\_Start proc

;Recieves: Nothing

;Description:Initializes the board and 2 placing at Two Randomly Generate Indexes

;Returns: Board With 2 placing at Two Randomly Generate Indexes

mov eax,0

mov Moves\_Counter,0

call Two\_Insert

call Two\_Insert

mov esi, offset Grid

mov edi,offset Backup\_Grid

mov ecx,16

l1:

lodsw

stosw

loop l1

call printboard

ret

Game\_Start endp

;=====================================================================

; gives random position where 2 and 4 should be inserted gives index as even number

Two\_or\_Four proc

;Recieves: Nothing

;Description:Generate Random Index And then Check Whether the Value On That Index is 0 Or Not If It is Zero Then Call Random\_Generator And

;Description Continue: ;Place The Random Number On The Index ,Other wise Generate Random Index Again

;Returns: Place 2 Or 4 On the board

again:

mov eax, 32 ; max range of random position

call randomize

call randomRange ; 0 to 31

mov ebx,eax

clc

shr ebx,1 ;check if even

jnc next

inc eax

cmp eax,32

jb next

mov eax,0

next:

cmp Grid[eax],0

je next1

add eax,2

cmp eax,32

jb next2

mov eax,0

next2: jmp next

next1:

mov temp\_EAX,eax

call RandomGenerator

mov bl,al

mov eax,temp\_EAX

mov Grid[eax],bx

ret

Two\_or\_Four endp

;==============================================

Two\_Insert proc

;Recieves: Nothing

;Description:Generate Random Index And then Check Whether the Value On That Index is 0 Or Not If It is Zero Then Call Random\_Generator And

;Description Continue: ;Place The Random Number On The Index ,Other wise Generate Random Index Again

;Returns: Place 2 On the board

againn:

mov eax, 32 ; max range of random position

call randomize

call randomRange ; 0 to 31

mov ebx,eax

clc

shr ebx,1 ;check if even

jnc nextt

inc eax

cmp eax,32

jb nextt

mov eax,0

nextt:

cmp Grid[eax],0

je next11

add eax,2

cmp eax,32

jb next22

mov eax,0

next22: jmp nextt

next11:

mov Grid[eax],2d

ret

Two\_Insert endp

;==============================================

RandomGenerator Proc

;Recieves: Nothing

;Description:Generate Random Number Whether 2 or 4

;Returns: Returns The Random Number In Al

J44:

call Randomize

mov eax,5 ;get random 0 to 4

call RandomRange ;

.if (eax != 2 && eax != 4)

JMP J44

.else

mov ranNum,AL ;save random number

.endif

ret

RandomGenerator ENDP

;==============================================

printBoard proc

;Recieves: Nothing

;Description:Prints The Main Game Board On Screen With Colors

;Returns:Nothing Just Display The Game Board

mov eax, white + (cyan \* 16)

call SetTextColor

call clrscr

mov esi, offset Grid

mov ecx, 4

mL:

call S\_space

call crlf

push ecx

mov ecx,4

call writespace

inL:

MOV EAX,0

mov ax, [esi]

call writeDec

add esi, type Grid

call writespace

CMP CL,1

je SKP

call writespace

SKP:

loop inL

call crlf

pop ecx

loop mL

call S\_space

call crlf

call GameControls

ret

printBoard endp

;====================================================

S\_space proc

;Recieves: Nothing

;Description:Just To Write Or Display a Single Space On Screen

;Returns:Nothing,Just Write A Single Space On Screen

mov al, ' '

call writechar

ret

S\_space endp

;====================================================

writespace proc

;Recieves: Nothing

;Description:Just To Write Or Display a Tab Space On Screen

;Returns:Nothing,Just Write A Tab Space On Screen

mov al, TAB

call writechar

ret

writespace endp

;=====================================

performOper proc

;Recieves: Nothing

;Description:Take Input For Movements And then Calling The Respective Function and Then Checking For Game Over and Game Win And If Both Not Then Generating a new Random Number

;Returns:Nothing,Just Updating The Board at the End

again:

call printBoard

mov eax,0

call readChar

cmp ax,4800h ;up arrow

je arrow\_up

cmp ax,5000h ;down arrow

je arrow\_Down

cmp ax,4D00h ;right arrow

je arrow\_Right

cmp ax,4B00h ;left arrow

je arrow\_left

cmp ax,011Bh ;escape key

je outt

jmp endd ; invalid key Menu\_input

arrow\_up:

inc Moves\_Counter

call Up\_Mov ; all operations related to mov up

jmp endd

arrow\_Down:

inc Moves\_Counter

call Down\_mov

jmp endd

arrow\_left:

inc Moves\_Counter

call Left\_mov

jmp endd

arrow\_Right:

inc Moves\_Counter

call Right\_mov

endd:

call checkWin ; check for win then out of perform operation

jz outt

call isEmpty ; check for isEmpty if not empty means lost

jnz outt

call Generate\_Values ;after every movement inserts2 at random position iff tiles have shifted there position

jmp again ;again repeat process

outt:

ret

performOper endp

;=====================================================

GameControls PROC

;Recieves: Nothing

;Description:This Function Displays Instructions For Movement and Display Score And Total Moves

;Returns:Nothing,Just Displays Move and Total moves

mov edx ,offset Msg1

call writestring

mov eax ,Moves\_Counter

call writeDec

call crlf

mov edx ,offset Msg2

call writestring

mov eax, Score

call writeDec

call crlf

mov edx ,offset Msg3

call writestring

call crlf

mov edx ,offset Msg4

call writestring

call crlf

ret

GameControls endp

;=====================================================

Up\_Mov proc

;Recieves: Nothing

;Description: Calling The Necessary Functions For The Upward Movement but It will remove 0's and add The Same Values and Place 0 at that position and again remove 0's

;Returns:Nothing

mov colNo,0

mov ecx, 4

moveL:

call Up\_zero\_Rmoving

call Up\_Add

call Up\_zero\_Rmoving

add colNo,2

loop moveL

ret

Up\_Mov endp

;=====================================================

Up\_zero\_Rmoving proc uses ecx

;Recieves: Ecx and colNO

;Description:Replacing Zero's Coloumn Wise Upward

;Returns:Nothing,Just Updating The Grid

mov esi,colNo

mov ecx, 4

mov edi, 0

mL:

cmp Grid[esi],0

je next

mov bx, Grid[esi]

mov temp[edi],bx

add edi, type Grid

next:

add esi,8

loop mL

mov esi,colNo

mov edi, 0

mov ecx,4

copyagain:

mov bx, temp[edi]

mov Grid[esi],bx

mov temp[edi],0

add esi, 8

add edi, type Grid

loop copyagain

ret

Up\_zero\_Rmoving endp

;============================================================

Up\_Add proc uses ecx

;Recieves: ECX,colNo

;Description:This Function Adds The Similar Adjacent Values Upward and Update the Score and Place 0 at merged Position

;Returns:Nothing,Just Update The Grid

mov esi, colNo

mov ecx,3

l1:

mov bx, Grid[esi]

cmp bx,Grid[esi+8]

jne next

shl bx,1

mov Grid[esi],bx

add Score ,ebx

mov Grid[esi+8],0

next:

add esi,8

loop l1

ret

Up\_Add endp

;=============================================================

Down\_mov proc

;Recieves: Nothing

;Description: Calling The Necessary Functions For The Downward Movement but It will remove 0's and add The Same Values and Place 0 at that position and again remove 0's

;Returns:Nothing

mov colNo,0

mov ecx, 4

moveD:

call Down\_zero\_Removing

call Down\_Add

call Down\_zero\_Removing

add colNo,2

loop moveD

ret

Down\_mov endp

;==============================================================

Down\_zero\_Removing proc uses ecx

;Recieves: Ecx and colNO

;Description:Replacing Zero's Coloumn Wise Downward

;Returns:Nothing,Just Updating The Grid

mov esi,colNo

add esi,24

mov ecx, 4

mov edi, 6

mL:

cmp Grid[esi],0

je next

mov bx, Grid[esi]

mov temp[edi],bx

sub edi, type Grid

next:

sub esi,8

loop mL

mov esi,colNo

mov edi, 0

mov ecx,4

copyagain:

mov bx, temp[edi]

mov Grid[esi],bx

mov temp[edi],0 ;reinitializing temp to zero for further moves

add esi, 8

add edi, type Grid

loop copyagain

ret

Down\_zero\_Removing endp

;==============================================================

Down\_Add proc uses ecx

;Recieves: ECX,colNo

;Description:This Function Adds The Similar Adjacent Values DownWard and Update the Score and Place 0 at merged Position

;Returns:Nothing,Just Update The Grid

mov esi, colNo

add esi,24

mov ecx,3

l1:

mov bx, Grid[esi]

cmp bx,Grid[esi-8]

jne next

shl bx,1

mov Grid[esi],bx

add Score ,ebx

mov Grid[esi-8],0

next:

sub esi, 8

loop l1

ret

Down\_Add endp

;==============================================================

Left\_mov proc

;Recieves: Nothing

;Description: Calling The Necessary Functions For The LeftWard Movement but It will remove 0's and add The Same Values and Place 0 at that position and again remove 0's

;Returns:Nothing

mov rowNo,0

mov ecx, 4

moveL:

call Left\_zero\_Removing

call Left\_Add

call Left\_zero\_Removing

add rowNo,8

loop moveL

ret

Left\_mov endp

;===============================================================

Left\_zero\_Removing proc uses ecx

;Recieves: Ecx and colNO

;Description:Replacing Zero's Row Wise Leftward

;Returns:Nothing,Just Updating The Grid

mov esi,rowNo

mov ecx, 4

mov edi, 0

mL:

cmp Grid[esi],0

je next

mov bx, Grid[esi]

mov temp[edi],bx

add edi, type Grid

next:

add esi, type Grid

loop mL

mov esi,rowNo

mov edi, 0

mov ecx,4

copyagain:

mov bx, temp[edi]

mov Grid[esi],bx

mov temp[edi],0

add esi, type Grid

add edi, type Grid

loop copyagain

ret

Left\_zero\_Removing endp

;==================================================================

Left\_Add proc uses ecx

;Recieves: ECX,colNo

;Description:This Function Adds The Similar Adjacent Values Leftward and Update the Score and Place 0 at merged Position

;Returns:Nothing,Just Update The Grid

mov esi, rowNo

mov ecx,3

l1:

mov bx, Grid[esi]

cmp bx,Grid[esi+2]

jne next

shl bx,1

mov Grid[esi],bx

add Score ,ebx

mov Grid[esi+2],0

next:

add esi,type word

loop l1

ret

Left\_Add endp

;==================================================================

Right\_mov proc

;Recieves: Nothing

;Description: Calling The Necessary Functions For The RightWard Movement but It will remove 0's and add The Same Values and Place 0 at that position and again remove 0's

;Returns:Nothing

mov rowNo,0

mov ecx, 4

moveR:

call Right\_zero\_Removing

call Right\_Add

call Right\_zero\_Removing

add rowNo,8

loop moveR

ret

Right\_mov endp

;===================================================================

Right\_zero\_Removing proc uses ecx

;Recieves: Ecx and colNO

;Description:Replacing Zero's Row Wise Rightward

;Returns:Nothing,Just Updating The Grid

mov esi,rowNo

add esi,6

mov ecx, 4

mov edi, 6

mL:

cmp Grid[esi],0

je next

mov bx, Grid[esi]

mov temp[edi],bx

sub edi, type Grid

next:

sub esi, type Grid

loop mL

mov esi,rowNo

mov edi, 0

mov ecx,4

copyagain:

mov bx, temp[edi]

mov Grid[esi],bx

mov temp[edi],0

add esi, type Grid

add edi, type Grid

loop copyagain

ret

Right\_zero\_Removing endp

;=====================================================================

Right\_Add proc uses ecx

;Recieves: ECX,colNo

;Description:This Function Adds The Similar Adjacent Values Rightward and Update the Score and Place 0 at merged Position

;Returns:Nothing,Just Update The Grid

mov esi, rowNo

add esi,6

mov ecx,3

l1:

mov bx, Grid[esi]

cmp bx,Grid[esi-2]

jne next

shl bx,1

mov Grid[esi],bx

add Score ,ebx

mov Grid[esi-2],0

next:

sub esi,type word

loop l1

ret

Right\_Add endp

;======================================================================

checkWin proc uses ecx

;Recieves: ECX

;Description:This Function Checks If Any Index has value 2048 Then Declares The Winner

;Returns:Nothing,Just Declare Winner

mov ecx,16

mov ax,2048

mov edi, offset Grid

repne scasw

jnz endd

call printBoard

call crlf

call crlf

MOV EAX,0

MOV AL,04h

Call setTextColor

mov edx, offset Win\_Msg

Call WriteString

Call ReadChar

endd:

ret

checkWin endp

;========================================================================

isEmpty proc uses ecx;;Game over

;Recieves: ECX

;Description:This Function Checks If The Blocks Are Complete And No Movement is possible

;Returns:Nothing,Just Declare the Loser

mov ecx,16

mov ax,0

mov edi, offset Grid

repne scasw

jz endd

call printBoard

call crlf

call crlf

MOV EAX,0

MOV AL,04h

Call setTextColor

mov edx, offset Lose\_Msg

Call WriteString

Call ReadChar

endd:

ret

isEmpty endp

;=========================================================================

Generate\_Values proc

;Recieves:Nothing

;Description:This Function Checks If Grid Is Updated Then This Function Is Called For Generating the values

;Returns: 2 or 4 in Al

call isChanged

jz endd

call Two\_or\_Four

endd:

ret

Generate\_Values endp

;=========================================================================

isChanged Proc

;Recieves:

;Description:This Function compares the backup grid and the Real grid if it is changed or not

;Returns:Nothing

mov ecx,16

mov esi,offset Grid

mov edi , offset Backup\_Grid

repe cmpsw

ret

isChanged endp

;=========================================================================

End main



















