National University of Computer and Emerging Sciences



Laboratory Final Project

Computer Organization and Assembly Language

|  |  |
| --- | --- |
| Lab Instructor(s) | Uzair Tahir |
| Semester | Fall 2019 |

**Department of Computer Science**

Introduction:

A Sudoku puzzle consists of 81 cells which are divided into nine columns, rows and regions. The task is now to place the numbers from 1 to 9 into the empty cells in such a way that in every row, column and 3×3 region each number appears only once.

Features:

* 9x9 with same difficulty level
* Customizable colors
* used to represent values (numbers)
* Choices of input
* Input value modes which accepts only valid values
* Print solved board
* Option for prompt exit
* Check whether the number is editable or not
* User friendly menu

**CODE:**

include irvine32.inc

.data

solved\_board DB 7, 3, 5, 6, 1, 4, 8, 9, 2

board\_r\_size = ($-unsolved\_board)

DB 8, 4, 2, 9, 7, 3, 5, 6, 1

DB 9, 6, 1, 2, 8, 5, 3, 7, 4

DB 2, 8, 6, 3, 4, 9, 1, 5, 7

DB 4, 1, 3, 8, 5, 7, 9, 2, 6

DB 5, 7, 9, 1, 2, 6, 4, 3, 8

DB 1, 5, 7, 4, 9, 2, 6, 8, 3

DB 6, 9, 4, 7, 3, 8, 2, 1, 5

DB 3, 2, 8, 5, 6, 1, 7, 4, 9

unsolved\_board DB 7, 3, 0, 0, 1, 4, 0, 9, 0

board\_r\_size = ($-unsolved\_board)

DB 0, 0, 2, 9, 7, 0, 0, 0, 1

DB 0, 6, 0, 0, 8, 0, 3, 7, 0

DB 2, 0, 6, 0, 4, 9, 0, 0, 0

DB 4, 0, 0, 8, 0, 7, 9, 0, 6

DB 0, 0, 9, 1, 2, 0, 0, 0, 8

DB 1, 5, 7, 0, 0, 2, 0, 8, 0

DB 6, 0, 0, 7, 3, 0, 2, 0, 5

DB 3, 2, 8, 0, 0, 1, 0, 4, 0

count\_helper DD 0

board\_helper DB 1

r\_ind DB ?

c\_ind DB ?

xCor Byte ?

yCor Byte ?

number Byte ?

number1 DD 0

comparison\_positive DB "You made the right guess!",0h

comparison\_negative DB "You made the wrong guess!",0h

xcord DB "Enter x cordinate",0h

ycord DB "Enter y cordinate",0h

num DB "Enter number",0h

intro DB "Welcome to Sudoku Game built with Assembly",0h

exit\_string DB "Thank You For Playing!", 0h

A DB "Press A to Add a new cell",0h

S DB "Press S to print the solved board",0h

E DB "Press E to exit current board",0h

.code

;----------------------compare---------------------------

;Checks if the answer in the given index is correct |

;Param x\_cor: row number |

;Param y\_cor: column number |

;Param input\_val: value entered by player |

;Returns: 1 in EAX if true, and 0 otherwise. |

;------------------------------------------------------------

compare PROC, x\_cor:BYTE, y\_cor:BYTE, input\_val:BYTE

PUSH EAX

;Decrementing and setting the global variables

MOV AL, x\_cor

DEC AL

MOV r\_ind, AL

MOV AL, y\_cor

DEC AL

MOV c\_ind, AL

POP EAX

;Computing the BASE address

MOV EAX, 0

MOV AL, r\_ind

MOV BL, board\_r\_size

MUL BL

;Getting the real answer value in AL

MOV EBX, OFFSET solved\_board

ADD EBX, EAX

MOVZX ESI, c\_ind

MOV AL, [EBX + ESI]

MOVZX EDX, AL

MOV BL, input\_val

CMP AL, BL

JE Right

JMP Wrong

Right:

MOV EAX, 2 ;set green color

CALL SetTextColor

PutInUnsolvedArray:

MOV EAX, 0

MOV AL, r\_ind

MOV BL, board\_r\_size

MUL BL

MOV EBX, OFFSET unsolved\_board

ADD EBX, EAX

MOVZX ESI, c\_ind

ADD ESI, EBX

MOV [ESI], DL

MOV EDX, OFFSET comparison\_positive

call WriteString

MOV EBX, OFFSET unsolved\_board

CALL crlf

CALL crlf

MOV EAX, 15 ;set color back to black

CALL SetTextColor

MOV EAX, 1

RET

Wrong:

MOV EAX, 4 ;set red color

CALL SetTextColor

PutInUnsolvedArray1:

MOV EAX, 0

MOV AL, r\_ind

MOV BL, board\_r\_size

MUL BL

MOV EBX, OFFSET unsolved\_board

ADD EBX, EAX

MOVZX ESI, c\_ind

ADD ESI, EBX

MOV [ESI], DL

MOV EDX, OFFSET comparison\_negative

call WriteString

CALL crlf

CALL crlf

MOV EAX, 15 ;set color back to black

CALL SetTextColor

MOV EAX, 0

RET

compare ENDP

;----------------------displayBoard----------------------------

;Prints the game board to the console screen. |

;------------------------------------------------------------

displayBoard proc

MOV EAX, 1

MOV ECX, board\_r\_size

PUSH EAX

MOV AL, ' '

CALL WriteChar

CALL WriteChar

CALL WriteChar

POP EAX

TopNumbers:

CALL writedec

PUSH EAX

MOV AL,' '

CALL writechar

CALL writechar

POP EAX

INC EAX

loop TopNumbers

CALL crlf

MOV ECX, board\_r\_size

PUSH EAX

MOV AL, ' '

CALL WriteChar

CALL WriteChar

CALL WriteChar

MOV AL, '\_'

TopDashes:

CALL WriteChar

CALL WriteChar

CALL WriteChar

loop TopDashes

POP EAX

CALL crlf

;Computing the BASE address

mov r\_ind, 0

mov c\_ind, 0

MOV EAX, 0

MOV AL, r\_ind

MOV BL, board\_r\_size

MUL BL

;Getting the real answer value in DL

MOV EBX, OFFSET unsolved\_board

ADD EBX, EAX

MOVZX ESI, c\_ind

MOV AL, [EBX + ESI]

MOV ECX, board\_r\_size

MOV count\_helper, ECX

INC count\_helper

PrintColumn:

MOV ECX, board\_r\_size

PUSH EAX

MOV AL, board\_helper

CALL WriteDec

MOV AL, ' '

CALL WriteChar

MOV AL, '|'

CALL WriteChar

POP EAX

PrintRow1:

CALL writedec

PUSH EAX

MOV AL,' '

CALL writechar

CALL writechar

POP EAX

INC ESI

MOV AL, [EBX + ESI]

loop PrintRow1

CALL crlf

DEC count\_helper

INC board\_helper

MOV ECX, count\_helper

loop PrintColumn

RET

displayBoard endp

;----------------------displaySolvedBoard-------------------------------

;Prints the solved game board to the console screen. |

;-----------------------------------------------------------------------

displaySolvedBoard proc

MOV EAX, 1

MOV ECX, board\_r\_size

PUSH EAX

MOV AL, ' '

CALL WriteChar

CALL WriteChar

CALL WriteChar

POP EAX

TopNumbers:

CALL writedec

PUSH EAX

MOV AL,' '

CALL writechar

CALL writechar

POP EAX

INC EAX

loop TopNumbers

CALL crlf

MOV ECX, board\_r\_size

PUSH EAX

MOV AL, ' '

CALL WriteChar

CALL WriteChar

CALL WriteChar

MOV AL, '\_'

TopDashes:

CALL WriteChar

CALL WriteChar

CALL WriteChar

loop TopDashes

POP EAX

CALL crlf

;Computing the BASE address

mov r\_ind, 0

mov c\_ind, 0

MOV EAX, 0

MOV AL, r\_ind

MOV BL, unsolved\_board

MUL BL

;Getting the real answer value in DL

MOV EBX, OFFSET solved\_board

ADD EBX, EAX

MOVZX ESI, c\_ind

MOV AL, [EBX + ESI]

MOV ECX, board\_r\_size

MOV count\_helper, ECX

INC count\_helper

PrintColumn:

MOV ECX, board\_r\_size

PUSH EAX

MOV AL, board\_helper

CALL WriteDec

MOV AL, ' '

CALL WriteChar

MOV AL, '|'

CALL WriteChar

POP EAX

PrintRow1:

CALL writedec

PUSH EAX

MOV AL,' '

CALL writechar

CALL writechar

POP EAX

INC ESI

MOV AL, [EBX + ESI]

loop PrintRow1

CALL crlf

DEC count\_helper

INC board\_helper

MOV ECX, count\_helper

loop PrintColumn

RET

displaySolvedBoard endp

;----------------------displayMenu------------------------------

;Prints the game menu to the console screen. |

;---------------------------------------------------------------

displayMenu proc

mov edx,offset A

call writestring

CALL crlf

mov edx,offset S

call writestring

CALL crlf

mov edx,offset E

call writestring

CALL crlf

RET

displayMenu endp

;----------------------CheckIndex--------------------------------

;Checks whether the input is correct. |

;---------------------------------------------------------------

CheckIndex PROC, val1:Byte, val2:Byte, val3:Byte

PUSH EAX

MOV AL, val1

MOV xCor, AL

MOV AL, val2

MOV yCor, AL

MOV AL, val3

MOV number, AL

POP EAX

;Checking xCor lies between 1 and 9

CMP xCor,9

ja WRONG

CMP xCor,1

jb WRONG

;Checking yCor lies between 1 and 9

CMP YCor,9

ja WRONG

CMP YCor,1

jb WRONG

;Checking num lies between 1 and 9

CMP number,9

ja WRONG

CMP number,1

jb WRONG

JMP RIGHT

WRONG:

MOV EAX,0

ret

RIGHT:

MOV EAX,1

ret

CheckIndex ENDP

;----------------------TakeInput--------------------------------

;Takes input from the player. |

;---------------------------------------------------------------

TakeInput proc

again:

mov edx,offset xcord

call writestring

CALL ReadDec

MOV xCor,AL

mov edx,offset ycord

call writestring

CALL ReadDec

MOV yCor,AL

mov edx,offset num

call writestring

CALL ReadDec

MOV number,AL

INVOKE checkindex, xCor, yCor, number

CMP EAX ,1

JE done

CALL crlf

JMP again

;if done then

done:

INVOKE compare, xCor, yCor, number

RET

TakeInput endp

;----------------------MAIN-------------------------------------

;Handles everything else in the game. |

;---------------------------------------------------------------

main proc

MOV EDX, OFFSET intro

CALL WriteString

CALL crlf

CALL crlf

MainAgain:

CALL displayBoard

CALL displayMenu

CALL ReadChar

GetChoice:

CMP AL,'A'

JNE Skip

CALL TakeInput

JMP MainAgain

Skip:

CMP AL,'S'

JE displaySolvedBoard

CMP AL,'E'

JE Ex

Ex:

MOV EDX, OFFSET exit\_string

CALL WriteString

exit

main endp

end main

