University of the Philippines Cebu
Department of Computer Science

CMSC 131
Computer Organization & Machine-level Programming
Final Game Project

MEMBERS:

Cobo, Mark Louis F.
BS Computer Science III

Lapura, Tehillah Leigh O.
BS Computer Science III

December 2017

TABLE OF CONTENTS

PROJECT SUMMARY

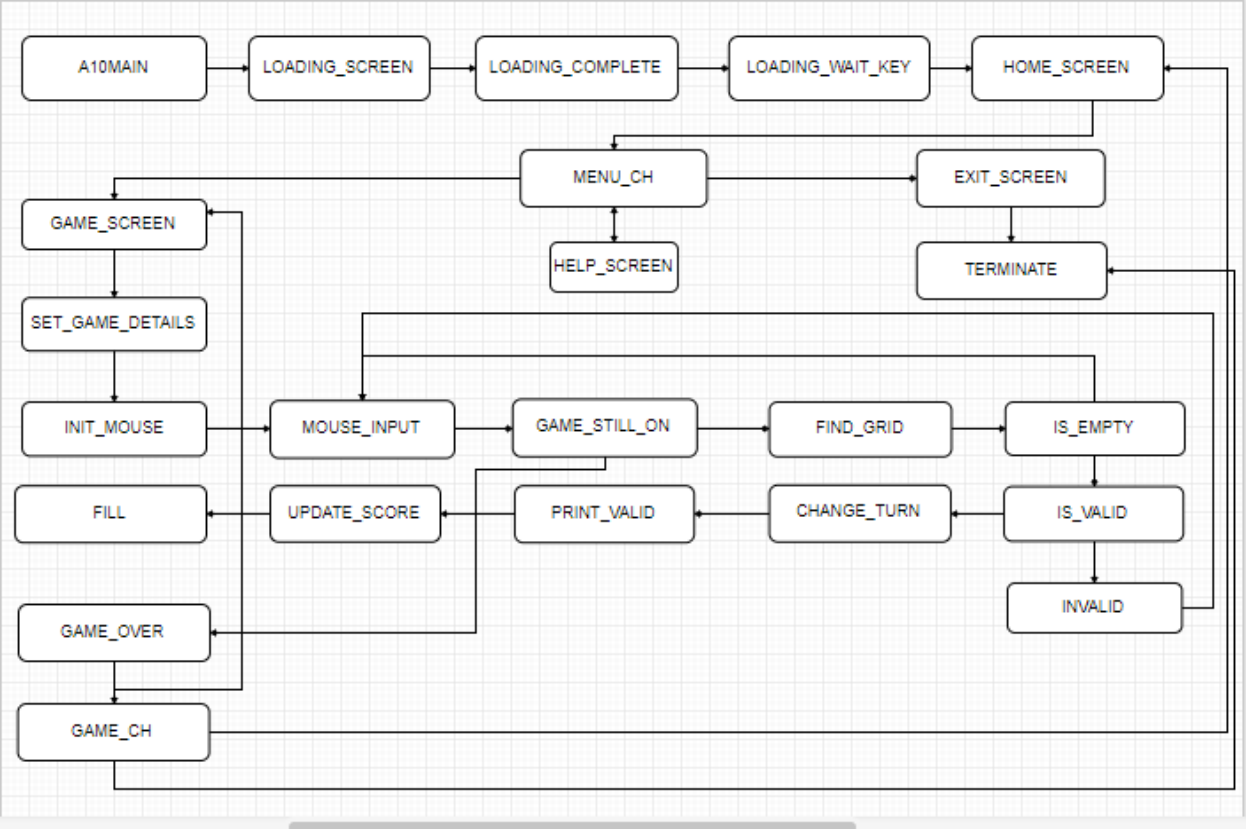
Othello is a strategy board game for two players, played on a 4×4 un-checkered board. There are 16 identical game pieces called disks (often spelled "discs"), which are light on one side and dark on the other. Players take turns placing disks on the board with their assigned color facing up. During a play, any disks of the opponent's color that are in a straight line and bounded by the disk just placed and another disk of the current player's color are turned over to the current player's color.

The object of the game is to have the majority of disks turned to display your color when the last playable empty square is filled.

GITHUB LINK: **https://github.com/tolapura/CMSC-131-Assembly-Programming**

LIST OF PROCEDURES

PROCEDURES FLOWCHART



FLOWCHART: THIS REPRESENTS THE FLOW OF THE PROGRAM

DEFINITION OF PROCEDURES

1. FILL – FILLS SELECTED CELL
2. 10MAIN – MAIN PROCEDURE
3. LOADING_SCREEN - DISPLAYS LOADING SCREEN
4. FILE_READ - READS FILE AND DISPLAYS THE CONTENT
5. CLEARFB - CLEARS FILE BUFFER
6. DISPLAY - PRINTS CONTENT
7. LOADING_COMPLETE - DISPLAYS COMPLETE STORE
8. LOADING_WAIT_KEY - WAITS FOR USER TO PRESS ANY KEY TO PROCEED TO HOME SCREEN
9. TERMINATE - TERMINATES PROGRAM
10. CLEAR_SCREEN - CLEARS SCREEN
11. SET_CURSOR - SETS CURSOR
12. DELAY - PROVIDES DELAY EFFECT
13. DELAYMORE - PROVIDES MORE DELAY EFFECT
14. HOME_SCREEN - DISPLAYS HOME SCREENN
15. MENU_CH - MENU FOR HOME SCREEN
16. GAME_SCREEN - SETS GAME SCREEN (SCORE, STATUS, TURN)
17. LOAD_WINNER - LOAD FILE THEN DISPLAYS PREVIOUS WINNERS
18. INIT_MOUSE - INITIALIZES MOUSE
19. SET_MOUSE - SETS MOUSE
20. DISP_MOUSE - DISPLAYS MOUSE
21. HIDE_MOUSE - HIDES MOUSE
22. SET_MINMAX_VER_MOUSE - SETS MIN AND MAX ROW OF MOUSE
23. SET_MINMAX_HOR_MOUSE - SETS MIN AND MAX COL OF MOUSE
24. UPDATE_SCORE - UPDATES SCORES AND GRID
25. FIND_GRID - FINDS THE MOUSE INPUT IN THE GRID
26. INVALID - PRINTS INVALID STRING
27. IS_EMPTY - CHECKS IF CELL IS EMPTY
28. IS_VALID - CHECKS IF MOVE IS VALID, FLIPS DISCS
29. MOUSE_INPUT - CHECKS IF PLAYER PRESSED LEFT MOUSE BUTTON
30. CHANGE_TURN - UPDATES TURN
31. CHECK_POSSIBLE - CHECKS IF CELL IS VALID FOR MOVEMENT
32. GAME_STILL_ON - CHECKS IF GAME IS STILL ON
33. GAME_OVER - GAME OVER SCREEN, DISPLAYS WINNER
34. WRITE_WINNER - WRITE WINNER NAME TO FILE
35. GAME_CH - MENU FOR GAME OVER
36. HELP_SCREEN - DISPLAYS HELP SCREEN
37. EXIT_SCREEN - TERMINATES PROGRAM, DOUBLE JUMP
38. BEEP_1 - PRODUCES BEEP SOUND
39. MUSIC - PRODUCES MUSIC
40. COMBI1 - PRODUCES MUSIC COMBINATON

SOURCE CODE OF PROCEDURES

```
;-------------------------------------------------------
;MAIN PROCEDURE
;-------------------------------------------------------
A10MAIN         PROC    FAR
            MOV     AX, @DATA
            MOV     DS, AX
            CALL    HIDE_MOUSE
            CALL    LOADING_SCREEN
A10MAIN         ENDP
;-------------------------------------------------------
;DISPLAYS LOADING SCREEN
;-------------------------------------------------------
LOADING_SCREEN      PROC    NEAR
            CALL    CLEAR_SCREEN
            LEA     DX,LOADING
            CALL    FILE_READ
            MOV     DL, 20H
            MOV     DH, 11
            CALL    SET_CURSOR
            LEA     DX, LOAD_STR
            CALL    DISPLAY
            MOV     CH, 32
            MOV     AH, 1
            INT     10H
            MOV     TEMP, 00
        ITERATE:
            MOV     DL, TEMP
            MOV     DH, 12
            CALL    SET_CURSOR
            MOV     AL, 0DBH
            MOV     AH, 02H

            MOV     DL, AL
            INT     21H
            CALL    DELAY
            INC     TEMP
            CMP     TEMP, 79
            JE      LOADING_COMPLETE
            JMP     ITERATE
LOADING_SCREEN      ENDP
;-------------------------------------------------------
;READS FILE AND DISPLAYS THE CONTENT
;-------------------------------------------------------
FILE_READ       PROC    NEAR
            MOV     AX, 3D02H
            INT     21H
            JC      FILE_ERROR
            MOV     FILE_HANDLE, AX
            CALL    CLEARFB
            MOV     AH, 3FH
            MOV     BX, FILE_HANDLE
            MOV     CX, 2000
            LEA     DX, FILE_BUFFER
            INT     21H
            JC      FILE_ERROR
            MOV     DX, 0000H
            CALL    SET_CURSOR
            LEA     DX, FILE_BUFFER
            CALL    DISPLAY
            MOV     AH, 3EH
            MOV     BX, FILE_HANDLE
            INT     21H
            JC      FILE_ERROR
            RET
        FILE_ERROR:
            LEA     DX, ERROR_STR
            CALL    DISPLAY
            RET
FILE_READ       ENDP
;-------------------------------------------------------
;CLEARS FILE BUFFER
;-------------------------------------------------------
CLEARFB         PROC    NEAR
            PUSH    CX
            LEA     SI, FILE_BUFFER
            MOV     CX, 2000
        L1:
            MOV     BYTE PTR[SI], '$'
            INC     SI
            LOOP    L1
            POP     CX
            RET
CLEARFB         ENDP
;-------------------------------------------------------
;PRINTS CONTENT
;-------------------------------------------------------
DISPLAY         PROC    NEAR
            MOV     AH, 09H
            INT     21H
            RET
DISPLAY         ENDP
;-------------------------------------------------------
;DISPLAYS COMPLT_STR
;-------------------------------------------------------
LOADING_COMPLETE    PROC    NEAR
            MOV     DL, 20H
            MOV     DH, 11
            CALL    SET_CURSOR
            LEA     DX, COMPLT_STR
            CALL    DISPLAY
            CALL    LOADING_WAIT_KEY
LOADING_COMPLETE    ENDP
;-------------------------------------------------------
;WAITS FOR USER TO PRESS ANY KEY TO
PROCEED TO HOME_SCREEN
;-------------------------------------------------------
LOADING_WAIT_KEY    PROC    NEAR
            MOV     AH, 00H
            INT     16H
            CALL    HOME_SCREEN
LOADING_WAIT_KEY    ENDP
;-------------------------------------------------------
;TERMINATES PROGRAM
;-------------------------------------------------------
TERMINATE       PROC    NEAR
            MOV     DL, 00
            MOV     DH, 13
            CALL    SET_CURSOR
            MOV     AX, 4C00H
            INT     21H
TERMINATE       ENDP
;-------------------------------------------------------
;CLEARS SCREEN
;-------------------------------------------------------
CLEAR_SCREEN        PROC    NEAR
            MOV     AX, 0600H
            MOV     BH, 02H
            MOV     CX, 0000H
            MOV     DX, 184FH
            INT     10H
            RET
```

```
CLEAR_SCREEN      ENDP
;-------------------------------------------------------
;SETS THE CURSOR'S POSITION
;-------------------------------------------------------
SET_CURSOR        PROC    NEAR
                  MOV     AH, 02H
                  MOV     BH, 00
                  INT     10H
                  RET
SET_CURSOR        ENDP
;-------------------------------------------------------
;PROVIDES A DELAY EFFECT
;-------------------------------------------------------
DELAY             PROC    NEAR
                  MOV     BP, 2
                  MOV     SI, 2
            DELAY2:
                  DEC     BP
                  NOP
                  JNZ     DELAY2
                  DEC     SI
                  CMP     SI, 0
                  JNZ     DELAY2
                  RET
DELAY             ENDP
;-------------------------------------------------------
;DELAYS THE PROGRAM LONGER THAN DELAY (PROC)
;-------------------------------------------------------
DELAYMORE         PROC    NEAR
                  MOV     BP, 5
                  MOV     SI, 5
            DELAY1:
                  DEC     BP
                  NOP
                  JNZ     DELAY1
                  DEC     SI
                  CMP     SI, 0
                  JNZ     DELAY1
                  RET
DELAYMORE         ENDP
;-------------------------------------------------------
;DISPLAYS HOME SCREEN
;-------------------------------------------------------
HOME_SCREEN       PROC    NEAR
                  CALL    CLEAR_SCREEN
                  MOV     DH, 0
                  MOV     DL, 0
                  CALL    SET_CURSOR
                  LEA     DX, MENU
                  CALL    FILE_READ
                  CALL    MENU_CH
HOME_SCREEN       ENDP
;-------------------------------------------------------
;MENU FOR HOME SCREEN
;-------------------------------------------------------
MENU_CH           PROC    NEAR
                  MOV     ROW, 11
                  MOV     COL, 15
                  MOV     DH, ROW
                  MOV     DL, COL
                  CALL    SET_CURSOR
                  LEA     DX, ARROW
                  CALL    DISPLAY
            CHOOSE:
                  MOV     AH, 10H
                  INT     16H
                  CMP     AL, 0DH
                  JE      CHOICE
                  CMP     AH, 4BH
                  JE      _LEFT
                  CMP     AH, 4DH
                  JE      _RIGHT
                  JMP     CHOOSE
            _RIGHT:
                  CALL    BEEP_1
                  CMP     COL, 49
                  JE      CHOOSE
                  MOV     DH, ROW
                  MOV     DL, COL
                  CALL    SET_CURSOR
                  LEA     DX, SPACE
                  CALL    DISPLAY
                  ADD     COL, 17
                  CALL    DISP_ARR
            _LEFT:
                  CALL    BEEP_1
                  CMP     COL, 15
                  JE      CHOOSE
                  MOV     DH, ROW
                  MOV     DL, COL
                  CALL    SET_CURSOR
                  LEA     DX, SPACE
                  CALL    DISPLAY
                  SUB     COL, 17
                  CALL    DISP_ARR
            CHOICE:
                  CMP     COL, 15
                  JE      GO_GAME
                  CMP     COL, 32
                  JE      GO_HELP
                  CMP     COL, 49
                  JE      GO_EXIT
            DISP_ARR:
                  MOV     DH, ROW
                  MOV     DL, COL
                  CALL    SET_CURSOR
                  LEA     DX, ARROW
                  CALL    DISPLAY
                  JMP     CHOOSE
            GO_GAME:
                  CALL    GAME_SCREEN
            GO_HELP:
                  CALL    HELP_SCREEN
            GO_EXIT:
                  CALL    EXIT_SCREEN
MENU_CH           ENDP
;-------------------------------------------------------
;MAIN LOGIC FOR GAMEPLAY
;-------------------------------------------------------
GAME_SCREEN       PROC    NEAR
                  CALL    CLEAR_SCREEN
                  CALL    SET_GAME_DETAILS
                  CALL    INIT_MOUSE
                  CALL    MOUSE_INPUT
GAME_SCREEN       ENDP
;-------------------------------------------------------
;SETS GAME SCREEN DETAILS (SCORE,STATUS,TURN)
;-------------------------------------------------------
SET_GAME_DETAILS  PROC    NEAR
                  MOV     BLACK_SCORE,0
                  MOV     WHITE_SCORE,0
                  MOV     SI, 0
            INIT_GRID:
                  MOV     GRID_ARRAY[SI], 0
```

```
        INC     SI
        CMP     SI,16
        JE      SET_BG
        JMP     INIT_GRID
SET_BG:
        MOV     SI,9
        MOV     GRID_ARRAY[SI],2
        MOV     SI,6
        MOV     GRID_ARRAY[SI],2
        MOV     SI,5
        MOV     GRID_ARRAY[SI],1
        MOV     SI,10
        MOV     GRID_ARRAY[SI],1
        MOV     TURN_BOL, 1
        MOV     AX, 0600H
        MOV     BH, 20H
        MOV     CX, 020FH
        MOV     DX, 163FH
        INT     10H
        MOV     DL, 15
        MOV     DH, 02
        CALL    SET_CURSOR
        LEA     DX, BOARD
        CALL    DISPLAY
        MOV     AX, 0600H
        MOV     BH, 0FH
        MOV     CX, 0000H
        MOV     DX, 014FH
        INT     10H
        MOV     DH, 01
        MOV     DL, 15
        CALL    SET_CURSOR
        LEA     DX, STATUS_STR
        CALL    DISPLAY
        MOV     DH, 01
        MOV     DL, 53
        CALL    SET_CURSOR
        LEA     DX, TURN_STR
        CALL    DISPLAY
        MOV     AX, 0600H
        MOV     BH, 0FH
        MOV     CX, 0800H
        MOV     DX, 0E09H
        INT     10H
        MOV     DL, 2
        MOV     DH, 8
        CALL    SET_CURSOR
        LEA     DX, BLACK_STR
        CALL    DISPLAY
        MOV     DL, 2
        MOV     DH, 13
        CALL    SET_CURSOR
        LEA     DX, WHITE_STR
        CALL    DISPLAY
        MOV     AX, 0600H
        MOV     BH, 0FH
        MOV     CX, 1700H
        MOV     DX, 174FH
        INT     10H
        MOV     DH,23
        MOV     DL,15
        CALL    SET_CURSOR
        LEA     DX,PREV_WINNER
        CALL    DISPLAY
        CALL    LOAD_WINNER
        CALL    UPDATE_SCORE
        RET
SET_GAME_DETAILS    ENDP
;-------------------------------------------------------
;LOAD FILE THEN DISPLAY PREVIOUS WINNER
;-------------------------------------------------------
LOAD_WINNER         PROC    NEAR
                    MOV     AH, 3DH
                    MOV     AL, 00
                    LEA     DX, WINNER_FILE
                    INT     21H
                    JC      DIS_READERR1
                    MOV     RFILEHANDLE, AX
                    MOV     AH, 3FH
                    MOV     BX, RFILEHANDLE
                    MOV     CX, 10
                    LEA     DX, RECORD_STR
                    INT     21H
                    JC      DIS_READERR2
                    CMP     AX, 00
                    JE      DIS_READERR3
                    LEA     SI, RECORD_STR
                    MOV     DH, 23
                    MOV     DL, 33
                    CALL    SET_CURSOR

                LOAD_ITER:
                    MOV     DL, [SI]
                    INC     SI
                    MOV     AH,02
                    INT     21H
                    MOV     DL,13
                    CMP     [SI], DL
                    JE      LEAVE_LOAD
                    MOV     DL,'$'
                    CMP     [SI], DL
                    JE      LEAVE_LOAD
                    JMP     LOAD_ITER
                LEAVE_LOAD:
                    MOV AH, 3EH
                    MOV BX, RFILEHANDLE
                    INT 21H
                    RET
                DIS_READERR1:
                    LEA DX, READ_ERR1
                    MOV AH, 09
                    INT 21H
                    RET
                DIS_READERR2:
                    LEA DX, READ_ERR2
                    MOV AH, 09
                    INT 21H
                    RET
                DIS_READERR3:
                    LEA DX, READ_ERR3
                    MOV AH, 09
                    INT 21H
                    RET
LOAD_WINNER         ENDP
;-------------------------------------------------------
;INITIALIZES MOUSE
;-------------------------------------------------------
INIT_MOUSE          PROC    NEAR
                    MOV     AX, 0000H
                    INT     33H
                    CALL    SET_MOUSE
                    CALL    SET_MINMAX_VER_MOUSE
                    CALL    SET_MINMAX_HOR_MOUSE
                    CALL    DISP_MOUSE
```

```
                RET
INIT_MOUSE      ENDP
;-------------------------------------------------------
;SETS MOUSE'S POSITION
;-------------------------------------------------------
SET_MOUSE       PROC    NEAR
                MOV     AX, 04H
                MOV     CX, 10H
                MOV     DX, 80H
                INT     33H
SET_MOUSE       ENDP
;-------------------------------------------------------
;SHOWS MOUSE CURSOR
;-------------------------------------------------------
DISP_MOUSE      PROC    NEAR
                MOV     BL, 00H
                MOV     AX, 01H
                INT     33H
                RET
DISP_MOUSE      ENDP
;-------------------------------------------------------
;HIDES MOUSE CURSOR
;-------------------------------------------------------
HIDE_MOUSE      PROC    NEAR
                MOV     AX, 02H
                INT     33H
                RET
HIDE_MOUSE      ENDP
;-------------------------------------------------------
;SETS MINIMUM AND MAXIMUM ROW OF MOUSE
;-------------------------------------------------------
SET_MINMAX_VER_MOUSE PROC   NEAR
                MOV     AX,08H
                MOV     CX,18H
                MOV     DX,0A8H
                INT     33H
SET_MINMAX_VER_MOUSE ENDP
;-------------------------------------------------------
;SETS MINIMUM AND MAXIMUM COLUMN OF MOUSE
;-------------------------------------------------------
SET_MINMAX_HOR_MOUSE PROC   NEAR
                MOV     AX,07H
                MOV     CX,80H
                MOV     DX,1F0H
                INT     33H
SET_MINMAX_HOR_MOUSE ENDP
;-------------------------------------------------------
;UPDATES SCORES AND GRID
;-------------------------------------------------------
UPDATE_SCORE    PROC    NEAR
                MOV     SI, 0
                MOV     BLACK_SCORE, 0
                MOV     WHITE_SCORE, 0
        ITERATE_GRID:
                CMP     GRID_ARRAY[SI],0
                JNE     CMP_1
                JMP     INCREMENT_SI
        CMP_1:
                CMP     GRID_ARRAY[SI],1
                JNE     CMP_2
                INC     BLACK_SCORE
                JMP     INCREMENT_SI
        CMP_2:
                INC     WHITE_SCORE
        INCREMENT_SI:
                INC     SI
                CMP     SI,16
                JE      UPDATE_EXIT
                JMP     ITERATE_GRID
UPDATE_EXIT:
                MOV     DH,9
                MOV     DL,2
                CALL    SET_CURSOR
                MOV     AX, BLACK_SCORE
                DIV     TEN
                ADD     AL, '0'
                MOV     DL, AL
                MOV     AH, 02
                INT     21H
                MOV     AX, BLACK_SCORE
                DIV     TEN
                ADD     AH, '0'
                MOV     DL, AH
                MOV     AH, 02
                INT     21H
                MOV     DH,14
                MOV     DL,2
                CALL    SET_CURSOR
                MOV     AX, WHITE_SCORE
                DIV     TEN
                ADD     AL, '0'
                MOV     DL, AL
                MOV     AH, 02
                INT     21H
                MOV     AX, WHITE_SCORE
                DIV     TEN
                ADD     AH, '0'
                MOV     DL, AH
                MOV     AH, 02
                INT     21H
                MOV     SI,0
                MOV     CURR_COL,0
                MOV     CURR_ROW,0
                CALL    HIDE_MOUSE
FILL_GRID:
                CMP     GRID_ARRAY[SI],0
                JNE     ARRAY_1
                JMP     INC_CURR
ARRAY_1:
                PUSH    SI
                CALL    DELAY
                MOV     BH,20H
                FILL    CURR_COL, CURR_ROW
                POP     SI
                CMP     GRID_ARRAY[SI],1
                JNE     ARRAY_2
                JMP     INC_CURR
ARRAY_2:
                MOV     BH,2FH
                PUSH    SI
                FILL    CURR_COL,CURR_ROW
                POP     SI
INC_CURR:
                INC     CURR_COL
                CMP     CURR_COL,4
                JNE     SIMPLIFY
                MOV     CURR_COL,0
                INC     CURR_ROW
SIMPLIFY:
                MOV     AX, CURR_ROW
                MUL     FOUR
                ADD     AX,CURR_COL
                MOV     SI,AX
                CMP     SI,16
```

```
            JE     __UPDATE_EXIT
            JMP    FILL_GRID
        __UPDATE_EXIT:
            CALL   DISP_MOUSE
            RET
UPDATE_SCORE      ENDP
;---------------------------------------------------------
;COMPUTES FOR GRID ROW AND COL, CONVERT UNITS TO 0-3
;---------------------------------------------------------
FIND_GRID         PROC   NEAR
            CALL   DELAYMORE
            ADD    MOUSEX,CX
            MOV    AX, MOUSEX
            DIV    EIGHT
            MOV    POSX, AL
            ADD    MOUSEY,DX
            MOV    AX, MOUSEY
            DIV    EIGHT
            MOV    POSY, AL
            MOV    DH, 00
            MOV    DL, 23
            CALL   SET_CURSOR
            CMP    POSX, 27
            JE     INVALID
            CMP    POSX, 39
            JE     INVALID
            CMP    POSX, 51
            JE     INVALID
            CMP    POSY, 7
            JE     INVALID
            CMP    POSY, 12
            JE     INVALID
            CMP    POSY, 17
            JE     INVALID
            CMP    POSX, 26
            JBE    FIND_GRID_ROW
            INC    GRID_COL
            CMP    POSX, 38
            JBE    FIND_GRID_ROW
            INC    GRID_COL
            CMP    POSX, 50
            JBE    FIND_GRID_ROW
            INC    GRID_COL
        FIND_GRID_ROW:
            CMP    POSY, 6
            JBE    IS_EMPTY
            INC    GRID_ROW
            CMP    POSY, 11
            JBE    IS_EMPTY
            INC    GRID_ROW
            CMP    POSY, 16
            JBE    IS_EMPTY
            INC    GRID_ROW
            CALL   IS_EMPTY
FIND_GRID         ENDP
;---------------------------------------------------------
;PRINTS INVALID_STR
;---------------------------------------------------------
INVALID           PROC   NEAR
            MOV    DH, 01
            MOV    DL, 23
            CALL   SET_CURSOR
            LEA    DX, INVALID_STR
            CALL   DISPLAY
            CALL   MOUSE_INPUT
INVALID           ENDP
;---------------------------------------------------------
;CHECKS IF CELL IS EMPTY
;---------------------------------------------------------
IS_EMPTY          PROC   NEAR
            MOV    AX, GRID_ROW
            MUL    FOUR
            ADD    AX, GRID_COL
            MOV    SI, AX
            MOV    DL, GRID_ARRAY[SI]
            CMP    DL, 0
            JNE    INVALID
            CALL   IS_VALID
IS_EMPTY          ENDP
;---------------------------------------------------------
;CHECKS IF MOVE IS VALID, FLIPS DISCS
;---------------------------------------------------------
IS_VALID          PROC   NEAR
            MOV    HAS_MOVE,0
            MOV    PCS_TO_FLIP,0
            MOV    AX, GRID_COL
            DEC    AX
            MOV    CURR_COL, AX
            MOV    AX,GRID_ROW
            MOV    CURR_ROW, AX
            CMP    GRID_COL, 0
            JNE    LC1
            JMP    START_RIGHT
        LC1:
            CMP    GRID_COL, 1
            JNE    LINT
            JMP    START_RIGHT
        LINT:
            MOV    CX, CURR_COL
            INC    CX
        LEFT:
            MOV    AX, CURR_ROW
            MUL    FOUR
            ADD    AX, CURR_COL
            MOV    SI, AX
            MOV    AH,GRID_ARRAY[SI]
            MOV    AL,TURN_BOL
            CMP    AL,AH
            JE     LEFT_FLAG
            CMP    GRID_ARRAY[SI],0
            JE     START_RIGHT
            INC    PCS_TO_FLIP
            DEC    CURR_COL
            LOOP   LEFT
            JMP    START_RIGHT
        LEFT_FLAG:
            CMP    PCS_TO_FLIP,0
            JE     START_RIGHT
            INC    HAS_MOVE
            MOV    AX, GRID_COL
            DEC    AX
            MOV    CURR_COL, AX
            MOV    AX,GRID_ROW
            MOV    CURR_ROW, AX

            MOV    CX, PCS_TO_FLIP
        LEFT_FLIP:
            MOV    AX, CURR_ROW
            MUL    FOUR
            ADD    AX, CURR_COL
            MOV    SI, AX
            MOV    AL, TURN_BOL
            MOV    GRID_ARRAY[SI],AL
            DEC    CURR_COL
```

```
        LOOP    LEFT_FLIP

    START_RIGHT:
        MOV     PCS_TO_FLIP,0
        MOV     AX, GRID_COL
        INC     AX
        MOV     CURR_COL, AX
        MOV     AX,GRID_ROW
        MOV     CURR_ROW, AX
        CMP     GRID_COL, 2
        JE      START_UP
        CMP     GRID_COL, 3
        JE      START_UP
    RIGHT:
        MOV     AX, CURR_ROW
        MUL     FOUR
        ADD     AX, CURR_COL
        MOV     SI, AX
        MOV     AL,TURN_BOL
        MOV     AH,GRID_ARRAY[SI]
        CMP     AL,AH
        JE      RIGHT_FLAG
        CMP     GRID_ARRAY[SI],0
        JE      START_UP
        INC     PCS_TO_FLIP
        INC     CURR_COL
        CMP     CURR_COL, 3
        JG      START_UP
        JMP     RIGHT
    RIGHT_FLAG:
        CMP     PCS_TO_FLIP,0
        JE      START_UP
        INC     HAS_MOVE
        MOV     AX, GRID_COL
        INC     AX
        MOV     CURR_COL, AX
        MOV     AX,GRID_ROW
        MOV     CURR_ROW, AX
        MOV     CX, PCS_TO_FLIP
    RIGHT_FLIP:
        MOV     AX, CURR_ROW
        MUL     FOUR
        ADD     AX, CURR_COL
        MOV     SI, AX
        MOV     AL, TURN_BOL
        MOV     GRID_ARRAY[SI],AL
        INC     CURR_COL
        LOOP    RIGHT_FLIP
    START_UP:
        MOV     PCS_TO_FLIP,0
        MOV     AX, GRID_COL
        MOV     CURR_COL, AX
        MOV     AX,GRID_ROW
        DEC     AX
        MOV     CURR_ROW, AX
        CMP     GRID_ROW, 0
        JE      START_DOWN
        CMP     GRID_ROW, 1
        JE      START_DOWN
    UP:
        MOV     AX, CURR_ROW
        MUL     FOUR
        ADD     AX, CURR_COL
        MOV     SI, AX
        MOV     AL,TURN_BOL
        MOV     AH,GRID_ARRAY[SI]
        CMP     AL,AH
        JE      UP_FLAG
        CMP     GRID_ARRAY[SI],0
        JE      START_DOWN
        INC     PCS_TO_FLIP
        DEC     CURR_ROW
        CMP     CURR_ROW, 0
        JL      START_DOWN
        JMP     UP
    UP_FLAG:
        CMP     PCS_TO_FLIP,0
        JE      START_DOWN
        INC     HAS_MOVE
        MOV     AX, GRID_COL
        MOV     CURR_COL, AX
        MOV     AX,GRID_ROW
        DEC     AX
        MOV     CURR_ROW, AX
        MOV     CX, PCS_TO_FLIP
    UP_FLIP:
        MOV     AX, CURR_ROW
        MUL     FOUR
        ADD     AX, CURR_COL
        MOV     SI, AX
        MOV     AL,TURN_BOL
        MOV     GRID_ARRAY[SI],AL
        DEC     CURR_ROW
        LOOP    UP_FLIP
    START_DOWN:
        MOV     PCS_TO_FLIP,0
        MOV     AX, GRID_COL
        MOV     CURR_COL, AX
        MOV     AX,GRID_ROW
        INC     AX
        MOV     CURR_ROW, AX
        CMP     GRID_ROW, 2
        JE      START_TL
        CMP     GRID_ROW, 3
        JE      START_TL
    DOWN:
        MOV     AX, CURR_ROW
        MUL     FOUR
        ADD     AX, CURR_COL
        MOV     SI, AX
        MOV     AL,TURN_BOL
        MOV     AH,GRID_ARRAY[SI]
        CMP     AL,AH
        JE      DOWN_FLAG
        CMP     GRID_ARRAY[SI],0
        JE      START_TL
        INC     PCS_TO_FLIP
        INC     CURR_ROW
        CMP     CURR_ROW, 3
        JG      START_TL
        JMP     DOWN
    DOWN_FLAG:
        CMP     PCS_TO_FLIP,0
        JE      START_TL
        INC     HAS_MOVE
        MOV     AX, GRID_COL
        MOV     CURR_COL, AX
        MOV     AX,GRID_ROW
        INC     AX
        MOV     CURR_ROW, AX
        MOV     CX, PCS_TO_FLIP
    DOWN_FLIP:
        MOV     AX, CURR_ROW
        MUL     FOUR
```

```
    ADD    AX, CURR_COL
    MOV    SI, AX
    MOV    AL,TURN_BOL
    MOV    GRID_ARRAY[SI],AL
    INC    CURR_ROW
    LOOP   DOWN_FLIP
START_TL:
    MOV    PCS_TO_FLIP,0
    MOV    AX, GRID_COL
    DEC    AX
    MOV    CURR_COL, AX
    MOV    AX,GRID_ROW
    DEC    AX
    MOV    CURR_ROW, AX
    CMP    GRID_ROW, 0
    JNE    TL_R1
    JMP    START_TR
TL_R1:
    CMP    GRID_ROW, 1
    JNE    TL_C0
    JMP    START_TR
TL_C0:
    CMP    GRID_COL, 0
    JNE    TL_C1
    JMP    START_TR
TL_C1:
    CMP    GRID_COL ,1
    JNE    TL
    JMP    START_TR
TL:
    MOV    AX, CURR_ROW
    MUL    FOUR
    ADD    AX, CURR_COL
    MOV    SI, AX
    MOV    AL,TURN_BOL
    MOV    AH,GRID_ARRAY[SI]
    CMP    AL,AH
    JE     TL_FLAG
    CMP    GRID_ARRAY[SI],0
    JE     START_TR
    INC    PCS_TO_FLIP
    DEC    CURR_ROW
    DEC    CURR_COL
    CMP    CURR_ROW, 0
    JL     START_TR
    CMP    CURR_COL, 0
    JL     START_TR
    JMP    TL
TL_FLAG:
    CMP    PCS_TO_FLIP,0
    JZ     START_TR
    INC    HAS_MOVE
    MOV    AX, GRID_COL
    DEC    AX
    MOV    CURR_COL, AX
    MOV    AX,GRID_ROW
    DEC    AX
    MOV    CURR_ROW, AX
    MOV    CX,PCS_TO_FLIP
TL_FLIP:
    MOV    AX, CURR_ROW
    MUL    FOUR
    ADD    AX, CURR_COL
    MOV    SI, AX
    MOV    AL,TURN_BOL
    MOV    GRID_ARRAY[SI],AL
    DEC    CURR_ROW
    DEC    CURR_COL
    LOOP   TL_FLIP
START_TR:
    MOV    PCS_TO_FLIP,0
    MOV    AX, GRID_COL
    INC    AX
    MOV    CURR_COL, AX
    MOV    AX,GRID_ROW
    DEC    AX
    MOV    CURR_ROW, AX
    CMP    GRID_ROW, 0
    JNE    TR_R3
    JMP    START_DL
TR_R3:
    CMP    GRID_ROW, 1
    JNE    TR_C2
    JMP    START_DL
TR_C2:
    CMP    GRID_COL, 2
    JNE    TR_C3
    JMP    START_DL
TR_C3:
    CMP    GRID_COL ,3
    JE     START_DL
TR:
    MOV    AX, CURR_ROW
    MUL    FOUR
    ADD    AX, CURR_COL
    MOV    SI, AX
    MOV    AL,TURN_BOL
    MOV    AH,GRID_ARRAY[SI]
    CMP    AL,AH
    JE     TR_FLAG
    CMP    GRID_ARRAY[SI],0
    JE     START_DL
    INC    PCS_TO_FLIP
    DEC    CURR_ROW
    INC    CURR_COL
    CMP    CURR_ROW, 0
    JL     START_DL
    CMP    CURR_COL, 3
    JG     START_DL
    JMP    TR
TR_FLAG:
    CMP    PCS_TO_FLIP,0
    JE     START_DL
    INC    HAS_MOVE
    MOV    AX, GRID_COL
    INC    AX
    MOV    CURR_COL, AX
    MOV    AX,GRID_ROW
    DEC    AX
    MOV    CURR_ROW, AX
    MOV    CX, PCS_TO_FLIP
TR_FLIP:
    MOV    AX, CURR_ROW
    MUL    FOUR
    ADD    AX, CURR_COL
    MOV    SI, AX
    MOV    AL,TURN_BOL
    MOV    GRID_ARRAY[SI],AL
    DEC    CURR_ROW
    INC    CURR_COL
    LOOP   TR_FLIP
START_DL:
    MOV    PCS_TO_FLIP,0
    MOV    AX, GRID_COL
```

```
        DEC     AX
        MOV     CURR_COL, AX
        MOV     AX,GRID_ROW
        INC     AX
        MOV     CURR_ROW, AX
        CMP     GRID_ROW, 2
        JNE     DL_R1
        JMP     START_DR
    DL_R1:
        CMP     GRID_ROW, 3
        JNE     DL_C0
        JMP     START_DR
    DL_C0:
        CMP     GRID_COL, 0
        JNE     DL_C1
        JMP     START_DR
    DL_C1:
        CMP     GRID_COL ,1
        JE      START_DR
    _DL:
        MOV     AX, CURR_ROW
        MUL     FOUR
        ADD     AX, CURR_COL
        MOV     SI, AX
        MOV     AL,TURN_BOL
        MOV     AH,GRID_ARRAY[SI]
        CMP     AL,AH
        JE      DL_FLAG
        CMP     GRID_ARRAY[SI],0
        JE      START_DR
        INC     PCS_TO_FLIP
        INC     CURR_ROW
        DEC     CURR_COL
        CMP     CURR_ROW, 3
        JG      START_DR
        CMP     CURR_COL, 0
        JL      START_DR
        JMP     _DL
    DL_FLAG:
        CMP     PCS_TO_FLIP,0
        JE      START_DR
        INC     HAS_MOVE
        MOV     AX, GRID_COL
        DEC     AX
        MOV     CURR_COL, AX
        MOV     AX,GRID_ROW
        INC     AX
        MOV     CURR_ROW, AX
        MOV     CX, PCS_TO_FLIP
    DL_FLIP:
        MOV     AX, CURR_ROW
        MUL     FOUR
        ADD     AX, CURR_COL
        MOV     SI, AX
        MOV     AL,TURN_BOL
        MOV     GRID_ARRAY[SI],AL
        INC     CURR_ROW
        DEC     CURR_COL
        LOOP    DL_FLIP
    START_DR:
        MOV     PCS_TO_FLIP,0
        MOV     AX, GRID_COL
        INC     AX
        MOV     CURR_COL, AX
        MOV     AX,GRID_ROW
        INC     AX
        MOV     CURR_ROW, AX
        CMP     GRID_ROW, 2
        JNE     DR_R3
        JMP     VALID_END
    DR_R3:
        CMP     GRID_ROW, 3
        JNE     DR_C2
        JMP     VALID_END
    DR_C2:
        CMP     GRID_COL, 2
        JNE     DR_C3
        JMP     VALID_END
    DR_C3:
        CMP     GRID_COL ,3
        JNE     DR
        JMP     VALID_END

    DR:
        MOV     AX, CURR_ROW
        MUL     FOUR
        ADD     AX, CURR_COL
        MOV     SI, AX
        MOV     AL,TURN_BOL
        MOV     AL,TURN_BOL
        MOV     AH,GRID_ARRAY[SI]
        CMP     AL,AH
        JE      DR_FLAG
        CMP     GRID_ARRAY[SI],0
        JE      VALID_END
        INC     PCS_TO_FLIP
        INC     CURR_ROW
        INC     CURR_COL
        CMP     CURR_ROW, 3
        JG      VALID_END
        CMP     CURR_COL, 3
        JG      VALID_END
        JMP     DR
    DR_FLAG:
        CMP     PCS_TO_FLIP,0
        JE      VALID_END
        INC     HAS_MOVE
        MOV     AX, GRID_COL
        INC     AX
        MOV     CURR_COL, AX
        MOV     AX,GRID_ROW
        INC     AX
        MOV     CURR_ROW, AX
        MOV     CX, PCS_TO_FLIP
    DR_FLIP:
        MOV     AX, CURR_ROW
        MUL     FOUR
        ADD     AX, CURR_COL
        MOV     SI, AX
        MOV     AL,TURN_BOL
        MOV     GRID_ARRAY[SI],AL
        INC     CURR_ROW
        INC     CURR_COL
        LOOP    DR_FLIP
    VALID_END:
        CMP     HAS_MOVE,0
        JE      MOVE_INVALID
        CALL    CHANGE_TURN
    MOVE_INVALID:
        CALL    INVALID
IS_VALID        ENDP
;-------------------------------------------------------
;CHECKS IF PLAYER PRESSED LEFT MOUSE BUTTON
;-------------------------------------------------------
```

```
MOUSE_INPUT        PROC    NEAR
                ;CALL   GAME_OVER
                CALL    GAME_STILL_ON
        MOUSE_CLICKED:
                CALL    DISP_MOUSE
                MOV     GRID_ROW, 0
                MOV     GRID_COL, 0
                MOV     DH, 01
                MOV     DL, 59
                CALL    SET_CURSOR
                CMP     TURN_BOL, 1
                JNE     PRINT_P2
                LEA     DX, P1TURN_STR
                JMP     CHECK_BUTTON
        PRINT_P2:
                LEA     DX, P2TURN_STR
        CHECK_BUTTON:
                CALL    DISPLAY
                MOV     MOUSEX, 0
                MOV     MOUSEY, 0
                MOV     AX, 03
                INT     33H
                CMP     BX, 0001H
                JNE     MOUSE_CLICKED
                CALL    BEEP_1
                CALL    FIND_GRID
MOUSE_INPUT ENDP
;---------------------------------------------------------
;UPDATES TURN
;---------------------------------------------------------
CHANGE_TURN        PROC    NEAR
                MOV     AX, GRID_ROW
                MUL     FOUR
                ADD     AX, GRID_COL
                MOV     SI, AX
                CMP     TURN_BOL, 1
                JNE     DEC_TURN_BOL
                MOV     BH, 20H
                MOV     GRID_ARRAY[SI],1
                INC     TURN_BOL
                JMP     PRINT_VALID
        DEC_TURN_BOL:
                MOV     BH, 2FH
                MOV     GRID_ARRAY[SI],2
                DEC     TURN_BOL
        PRINT_VALID:
                CALL    DELAYMORE
                CALL    UPDATE_SCORE
                MOV     DH, 01
                MOV     DL, 23
                CALL    SET_CURSOR
                LEA     DX, VALID_STR
                CALL    DISPLAY
                CALL    MOUSE_INPUT
CHANGE_TURN        ENDP
;---------------------------------------------------------
;CHECKS IF GAME IS STILL ON
;---------------------------------------------------------
GAME_STILL_ON      PROC    NEAR
                MOV     GRID_COL,0
                MOV     GRID_ROW,0
                MOV     SI,0
        GAME_LOOP:
                CMP     GRID_ARRAY[SI], 0
                JNE     INCREMENT
                CALL    CHECK_POSSIBLE
                CMP     HAS_MOVE,1
                JE      ___RETURN
        INCREMENT:
                INC     GRID_COL
                CMP     GRID_COL,4
                JNE     COMPUTE_SI
                INC     GRID_ROW
                MOV     GRID_COL,0
        COMPUTE_SI:
                MOV     AX, GRID_ROW
                MUL     FOUR
                ADD     AX, GRID_COL
                MOV     SI, AX
                CMP     SI,16
                JE      __GAME_OVER
                JMP     GAME_LOOP
        ___RETURN:
                RET
        __GAME_OVER:
                CALL    GAME_OVER
GAME_STILL_ON      ENDP
;---------------------------------------------------------
;GAME OVER SCREEN, DISPLAYS WINNER
;---------------------------------------------------------
GAME_OVER          PROC    NEAR
                CALL    DELAYMORE
                CALL    HIDE_MOUSE
                CALL    CLEAR_SCREEN
                MOV     DH, 0
                MOV     DL, 0
                CALL    SET_CURSOR
                LEA     DX, GAMEOVER
                CALL    FILE_READ
                MOV     DH,7
                MOV     DL,33
                CALL    SET_CURSOR
                CALL    COMBI1
                MOV     AX, BLACK_SCORE
                DIV     TEN
                MOV     CL,AL
                MOV     AX, WHITE_SCORE
                DIV     TEN
                MOV     CH,AL
                CMP     CL,CH
                JE      COMPARE_ONES
                JL      WHITE_WINS
                JG      BLACK_WINS
        COMPARE_ONES:
                MOV     AX, BLACK_SCORE
                DIV     TEN
                MOV     CL,AH
                MOV     AX, WHITE_SCORE
                DIV     TEN
                MOV     CH,AH
                CMP     CL,CH
                JE      DRAW
                JL      WHITE_WINS
                JG      BLACK_WINS
        DRAW:
                LEA     DX, DRAW_STR
                CALL    DISPLAY
                JMP     ENTER_NAME
        BLACK_WINS:
                LEA     DX, BLACKW_STR
                CALL    DISPLAY
                JMP     ENTER_NAME
        WHITE_WINS:
                LEA     DX, WHITEW_STR
```

```
        CALL    DISPLAY
    ENTER_NAME:
        MOV     DH, 15
        MOV     DL, 35
        CALL    SET_CURSOR
        LEA     DX, WINNER_NAME
        MOV     AH, 3FH
        MOV     BX, 00
        MOV     CX, 20
        INT     21H

        CALL    WRITE_WINNER
        CALL    GAME_CH
GAME_OVER       ENDP
;-------------------------------------------------------
;WRITE WINNER NAME TO FILE
;-------------------------------------------------------
WRITE_WINNER    PROC    NEAR
        MOV     AH, 3CH
        MOV     CX, 00
        LEA     DX, WINNER_FILE
        INT     21H
        JC      DIS_WRITEERR1
        MOV     WFILEHANDLE, AX

        MOV     AH, 40H
        MOV     BX, WFILEHANDLE
        MOV     CX, 10
        LEA     DX, WINNER_NAME
        INT     21H
        JC      DIS_WRITEERR2
        CMP     AX, 10
        JNE     DIS_WRITEERR3
        JMP     CLOSE_FILE_HANDLE

    DIS_WRITEERR1:
        LEA     DX, WRITE_ERR1
        MOV     AH, 09
        INT     21H
        JMP     CLOSE_FILE_HANDLE

    DIS_WRITEERR2:
        LEA     DX, WRITE_ERR2
        MOV     AH, 09
        INT     21H
        JMP     CLOSE_FILE_HANDLE

    DIS_WRITEERR3:
        LEA     DX, WRITE_ERR3
        MOV     AH, 09
        INT     21H
        JMP     CLOSE_FILE_HANDLE

    CLOSE_FILE_HANDLE:
        MOV     AH, 3EH
        MOV     BX, WFILEHANDLE
        INT     21H
        RET

WRITE_WINNER    ENDP
;-------------------------------------------------------
;MENU FOR GAME OVER
;-------------------------------------------------------
GAME_CH         PROC    NEAR
        MOV     ROW, 11
        MOV     COL, 15
        MOV     DH, ROW
        MOV     DL, COL
        CALL    SET_CURSOR
        LEA     DX, ARROW
        CALL    DISPLAY
    GAME_CHOOSE:
        MOV     AH, 10H
        INT     16H
        CMP     AL, 0DH
        JE      GAME_CHOICE
        CMP     AH, 4BH
        JE      GAME_LEFT
        CMP     AH, 4DH
        JE      GAME_RIGHT
        JMP     GAME_CHOOSE
    GAME_RIGHT:
        CALL    BEEP_1
        CMP     COL, 49
        JE      GAME_CHOOSE
        MOV     DH, ROW
        MOV     DL, COL
        CALL    SET_CURSOR
        LEA     DX, SPACE
        CALL    DISPLAY
        ADD     COL, 17
        CALL    GAME_DISP_ARR
    GAME_LEFT:
        CALL    BEEP_1
        CMP     COL, 15
        JE      GAME_CHOOSE
        MOV     DH, ROW
        MOV     DL, COL
        CALL    SET_CURSOR
        LEA     DX, SPACE
        CALL    DISPLAY
        SUB     COL, 17
        CALL    GAME_DISP_ARR
    GAME_CHOICE:
        CMP     COL, 15
        JE      GAME_GO_PLAY
        CMP     COL, 32
        JE      GAME_GO_HOME
        CMP     COL, 49
        JE      GAME_GO_EXIT
    GAME_DISP_ARR:
        MOV     DH, ROW
        MOV     DL, COL
        CALL    SET_CURSOR
        LEA     DX, ARROW
        CALL    DISPLAY
        JMP     GAME_CHOOSE
    GAME_GO_PLAY:
        CALL    GAME_SCREEN
    GAME_GO_HOME:
        CALL    HOME_SCREEN
    GAME_GO_EXIT:
        CALL    EXIT_SCREEN
GAME_CH         ENDP
;-------------------------------------------------------
;DISPLAYS HELP SCREEN
;-------------------------------------------------------
HELP_SCREEN     PROC    NEAR
        MOV     DH, 0
        MOV     DL, 0
        CALL    SET_CURSOR
        CALL    CLEAR_SCREEN
        LEA     DX, HELP
        CALL    FILE_READ
```

```
            MOV    AH, 00H
            INT    16H
            JMP    HOME_SCREEN
HELP_SCREEN       ENDP
;-------------------------------------------------------
;PRODUCES BEEP SOUND
;-------------------------------------------------------
BEEP_1          PROC   NEAR
            MOV    AL, 182
            OUT    43H, AL
            MOV    AX, 4304
            OUT    42H, AL
            MOV    AL, AH
            OUT    42H, AL
            IN     AL, 61H
            OR     AL, 00000011B
            OUT    61H, AL
            MOV    BEEPBX, 25
        .PAUSEA:
            MOV    BEEPCX, 2900
        .PAUSEB:
            DEC    BEEPCX
            JNE    .PAUSEB
            DEC    BEEPBX
            JNE    .PAUSEA
            IN     AL, 61H
            AND    AL, 11111100B
            OUT    61H, AL
            RET
BEEP_1          ENDP
;-------------------------------------------------------
;PRODUCES MUSIC
;-------------------------------------------------------
MUSIC           PROC   NEAR
            MOV    AL, 182
            OUT    43H, AL

            OUT    42H, AL
            MOV    AL, AH
            OUT    42H, AL
            IN     AL, 61H

            OR     AL, 00000011B
            OUT    61H, AL
            MOV    BEEPBX, 25

        .PAUSE1:
        .PAUSE2:
            DEC    BEEPCX
            JNE    .PAUSE2
            DEC    BEEPBX
            JNE    .PAUSE1
            IN     AL, 61H

            AND    AL, 11111100B
            OUT    61H, AL
            RET
MUSIC           ENDP
;-------------------------------------------------------
;PRODUCES MUSIC COMBO
;-------------------------------------------------------
COMBI1          PROC   NEAR
            MOV    AX, 2152
            MOV    BEEPCX, 500
            CALL   MUSIC
            MOV    AX, 3403
            MOV    BEEPCX, 10
            CALL   MUSIC
            MOV    AX, 3224
            MOV    BEEPCX, 1
            CALL   MUSIC
            MOV    AX, 3416
            MOV    BEEPCX, 10
            CALL   MUSIC
            MOV    AX, 3834
            MOV    BEEPCX, 10
            CALL   MUSIC
            RET
COMBI1    ENDP
```

GAME SCREENS

I. Starting Screen

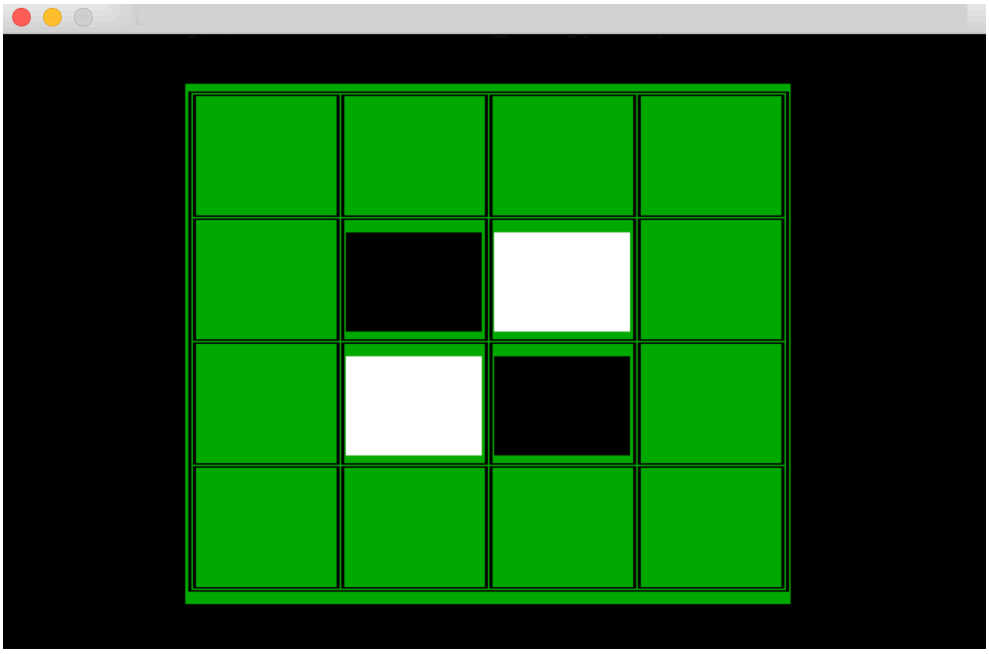
- before heading to the *game title screen*,
a screen with a loading prompt will first appear.

II. Game Title Screen

- the user can choose which screen to navigate:
  - **PLAY GAME** – to start the game,
  - **HOW TO PLAY** – contains instructions of the game,
  - **EXIT GAME** – to stop the program.

III. How to Play Screen

- contains the *instruction* of the game.

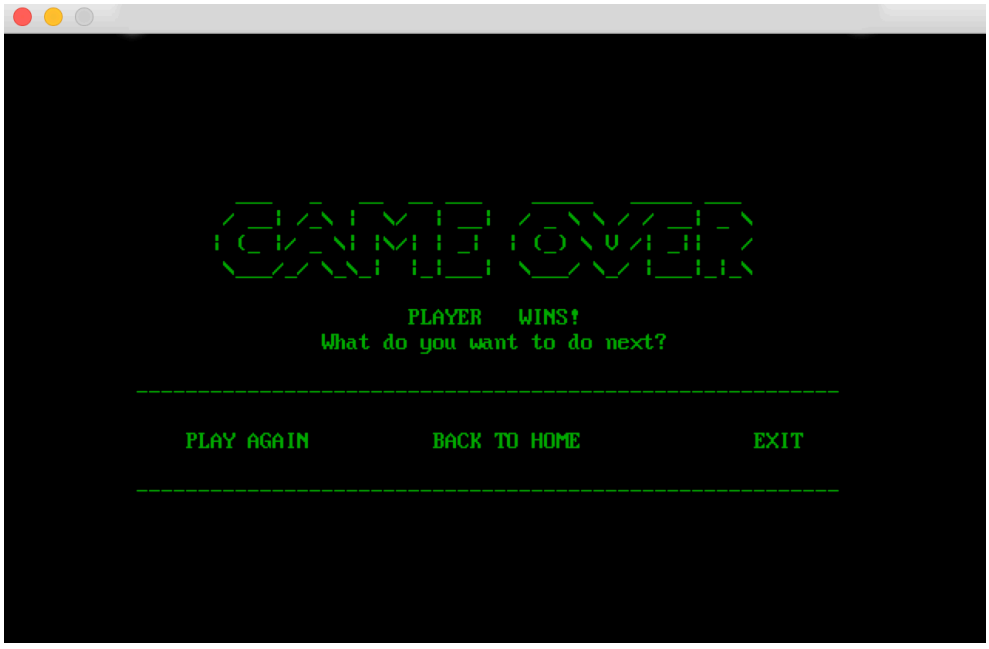


(we haven't included the texts yet because we still haven't finalized the UI)

IV. Actual Game Screen

- the following can be found in the actual game screen:

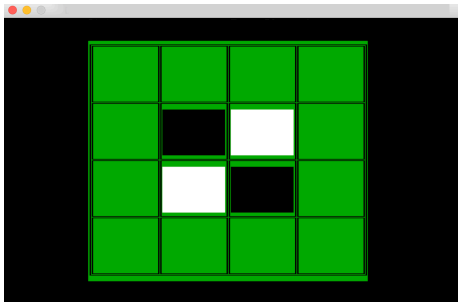
- **4 x 4 GRID** – where the game is played,
- **SCORES TEXT** – scores of both players,
- **CURRENT PLAYER** – current player to move,
- **STATUS** – prompts if the key entered is *valid* or *invalid*.



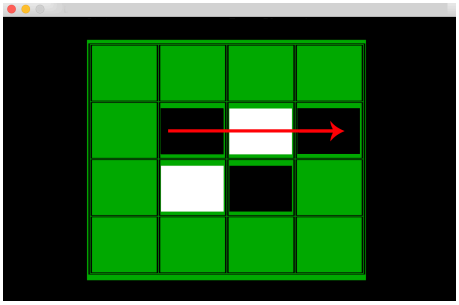
IV. Game Over Screen

- this screen appears once the game has ended.

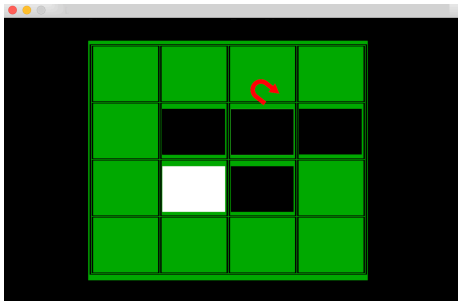
GAME RULES



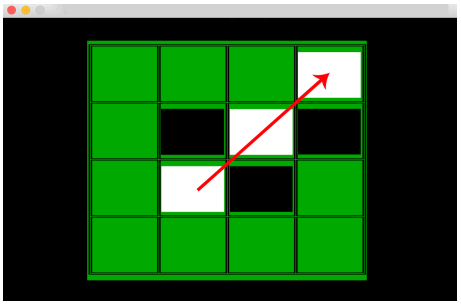
Game board and discs. Othello is played on an 4x4 non-checkered board with 16 discs which are either black or white in color. The player with black disks typically goes first; in other versions, the players decide who plays first.



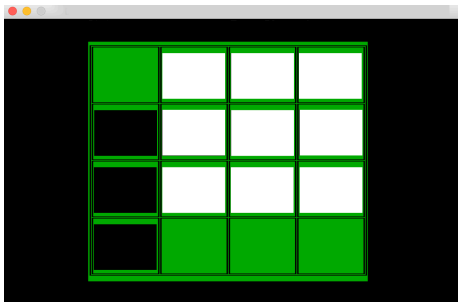
Place the first disc in a spot that outflanks an opponent's disc. To "outflank" a disc means to surround a row of your opponent's discs with two of your own discs. A "row" consists of one or more discs that form a line horizontally, vertically or diagonally.



Flip the outflanked disc to its opposite side. Once a disc is outflanked, it is flipped to the opposite color and becomes the other player's disc. Following the example in the previous step, the white disc that is outflanked is turned to black and belongs to the black player..



Pass the turn to your opponent. The opponent now places the second disc in a spot that outflanks at least one of the first player's discs. Assuming the second player plays the white discs, they would place one of their discs so a row of at least one black disc is framed by two white discs on each side, flipping the outflanked black disks to white. Remember that the row can be horizontal, diagonal or vertical.



Continue taking turns placing discs until a legal move isn't possible. For a move to be legal, a disc must always be placed in a position where it can outflank a row of the opponent's discs. If this isn't possible, you must forfeit your turn until you can perform a legal move. If neither player can perform a legal move, usually because all spaces are filled, play ends.