UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS
ARQUITECTURA DE COMPUTADORES Y ENSAMBLADORES 1
SEGUNDO SEMESTRE 2020
ING. OTTO ESCOBAR
TUTOR ACADÉMICO SECCIÓN A: HERBERTH ARGUETA

MANUAL TECNICO: PRACTICA 3

NOMBRE: Juan José Ramos Campos
CARNET: 201801262
FECHA:26/09/20

Contenido

Main.inc

Este es el archivo principal de la aplicación, cuenta con una serie de macros especiales que le permiten realizar diferentes cosas entre ellas:

print

```
print macro cadena ; 
MOV ah,09h
MOV dx,@data
MOV ds,dx
MOV dx, offset cadena
int 21h
endm
```

Esta macro sirve para hacer un print en pantalla de una cadena que recibe como parámetro

getText

```
getText macro buffer
LOCAL CONTINUE, FIN
PUSH SI
PUSH AX

xor si,si
CONTINUE:
getChar
cmp al,0dh
je FIN
mov buffer[si],al
inc si
jmp CONTINUE

FIN:
mov al, '$'
mov buffer[si],al

POP AX
POP SI
endm
```

Esta macro sirve para poder escribir desde el teclado, recibe como parámetro un arreglo de caracteres donde se almacenara cada carácter pulsado en el teclado

COMPARARSAVE, COMPARARSHOW, COMPARAREXIT

```
COMPARARSAVE macro buffer
LOCAL IGUAL, DIFERENTE

    xor CX, CX
    mov cx,4   ;Determinamos
    mov AX,DS  ;mueve el seg
    mov ES,AX  ;Mueve los da
    lea si, buffer
    lea di, savecommand

    repe cmpsb  ;compara las
    je IGUAL ;Si fueron igua
    jne DIFERENTE

    IGUAL:
        jmp CREATE

    DIFERENTE:
        MOV cx, 7
endm
```

Estas macros se encargan de realizar la operación de comprar el comando entrante con la palabra SAVE, SHOW o EXIT, respectivamente eso para ejecutar el comando especial que se encarga de salvar el tablero actual, mostrar el html o salir de la aplicación

Las tres macros tienen la misma estructura que es la siguiente, lo que cambia es el comando a evaluar, y el jumper a hacer

En primer lugar se limpia el registro CX, para pode inicarle la cantidad de bits a leer, se carga los datos de entrada al registro AX y de AX a EX

Se cargan el buffer al registro si y el comando a leer en el di, luego se procede a usar la función cmpsb o compare para verificar si ambas cadenas son iguales

Segmento de datos

En el segmento de dato se declararon todas las variables que se utilizaran en la aplicación,

```
;====================== SEGMENTO DE DATOS =================
;========== VARIABLES

comando     db 7 dup('$'), '$'
savecommand db 'SAVE', '$'
exitcommand db 'EXIT', '$'
showcommand db 'SHOW', '$'

encab       db 'UNIVERSIDAD DE SAN CARLOS DE GUATEMALA', 10,13,  'FACULTA
tplayerb    db 10,13,'Turno Jugador Blanco: ', '$'
tplayern    db 10,13,'Turno Jugador Negro: ', '$'
error       db 10,13,'Comando no soportado ',10,13, '$'
final       db 10,13,'Gracias pr jugar, vuelva pronto :) ',10,13, '$'
```

Como por ejemplo, algunos mensajes, la variable comando que servirá para escribir los comandos a realizar

Las variables de archivo sirven para poder trabajar con los archivos que se van a crear y leer

```
;========== VARIABLES ARCHIVO
filepath    db 100 dup('$') ; ruta archivo
buffread    db 200 dup('$') ; buffer lectura
buffwrite   db 200 dup('$') ; buffer escritura
handlefile  dw ?


msgcreatefile   db 0ah,0dh, 'Ingrese el nombre para guardar: ', 32, '$'
msgfilesave     db 0ah,0dh, 'Juego guardado con exito! ', 32, '$'


msgOpenFile db 0ah,0dh, 'Ingrese el nombre para abrir: ', 32, '$'
msgFileOpen db 0ah,0dh, 'Juego cargado con exito! ', 32, '$'

msmError1 db 0ah,0dh,'Error al abrir archivo','$'
msmError2 db 0ah,0dh,'Error al leer archivo','$'
msmError3 db 0ah,0dh,'Error al crear archivo','$'
msmError4 db 0ah,0dh,'Error al Escribir archivo','$'
```

El file path almacenara el path del archivo que se va a leer
bufferread almacenara el texto leído

Buffwrite almacena el texto que se va a escribir

Los mensajes son para que el usuario sepa que esta haciendo en todo momento

Las variabes de tablero: estas se encargan de simular el tablero en la pantalla, costa de 8 arreglos llenos de los caracteres que simularan las fichas, la explicación de como se imprime el tablero se hará mas adelante

```
;========== VARIABLES TABLERO


row8    db 101b,000b,101b,000b,101b,000b,101b,000b
row7    db 000b,101b,000b,101b,000b,101b,000b,101b
row6    db 101b,000b,101b,000b,101b,000b,101b,000b
row5    db 000b,000b,000b,000b,000b,000b,000b,000b
row4    db 000b,000b,000b,000b,000b,000b,000b,000b
row3    db 000b,100b,000b,100b,000b,100b,000b,100b
row2    db 100b,000b,100b,000b,100b,000b,100b,000b
row1    db 000b,100b,000b,100b,000b,100b,000b,100b



entere  db 0ah,0dh,'$'
endline db 0ah,0dh
eig     db '8   |','$'
sev     db '7   |','$'
six     db '6   |','$'
fiv     db '5   |','$'
fou     db '4   |','$'
thr     db '3   |','$'
two     db '2   |','$'
one     db '1   |','$'

fneg    db 'FNEG|','$'
fblc    db 'FBLA|','$'
rneg    db 'RNEG|','$'
rblc    db 'RBLA|','$'
void    db '    |','$'
```

Segmento código

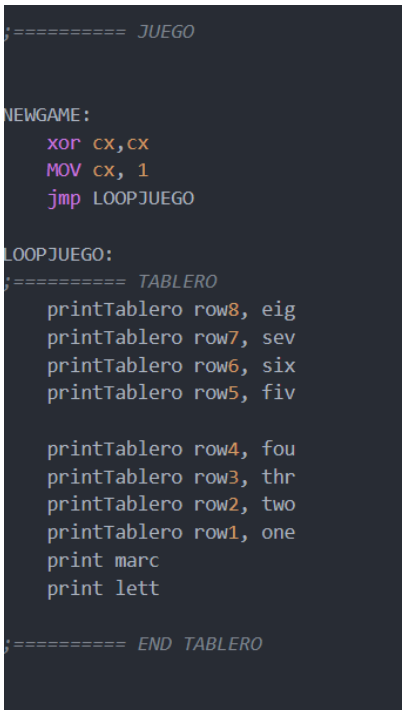
Aquí es donde empieza la funcionalidad del juego

```
main proc
    MENUPRINCIPAL:
    print encab
    getChar
    cmp al,'1'
    je NEWGAME
    cmp al,'2'
    je OPENGAME
    cmp al,'3'
    je ENDGAME
    jmp MENUPRINCIPAL
```

La primera ejecución es el menú principal, se le solicita a la persona que indique la acción a realizar y lo envia al apartado correspondiente

NEWGAME:

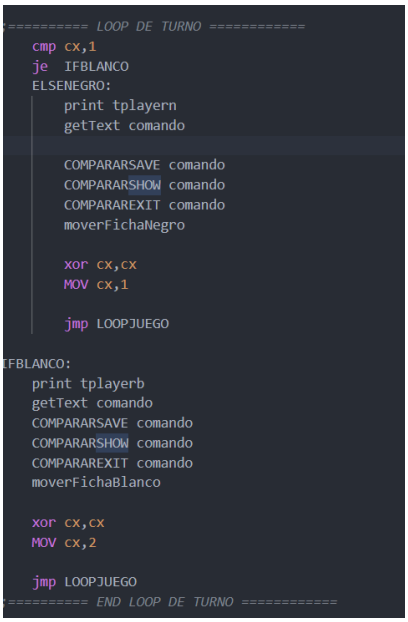
Esta es la etiqueta inicial del juego, se encarga de llamar otra etiqueta llamada LOOPJUEGO QUE se encarga de hacer un print al tablero y avanzar a los turnos



```
;========== JUEGO


NEWGAME:
    xor cx,cx
    MOV cx, 1
    jmp LOOPJUEGO

LOOPJUEGO:
;========== TABLERO
    printTablero row8, eig
    printTablero row7, sev
    printTablero row6, six
    printTablero row5, fiv

    printTablero row4, fou
    printTablero row3, thr
    printTablero row2, two
    printTablero row1, one
    print marc
    print lett

;========== END TABLERO
```

LOOPTURNO

Las etiquetas IFBLANCO y ELSENEGRO se encargan de los turnos, dependiendo del valor del registro CX entra a una etiqueta o a otra, cambiando el valor del registro CX para que pueda entrar al otro jugador en el siguiente turno



```
;========== LOOP DE TURNO ===========
    cmp cx,1
    je  IFBLANCO
    ELSENEGRO:
        print tplayern
        getText comando

        COMPARARSAVE comando
        COMPARARSHOW comando
        COMPARAREXIT comando
        moverFichaNegro

        xor cx,cx
        MOV cx,1

        jmp LOOPJUEGO

IFBLANCO:
    print tplayerb
    getText comando
    COMPARARSAVE comando
    COMPARARSHOW comando
    COMPARAREXIT comando
    moverFichaBlanco

    xor cx,cx
    MOV cx,2

    jmp LOOPJUEGO
;========== END LOOP DE TURNO ===========
```

OPENGAME

Esta etiqueta se encarga de abrir el archivo de un juego guardado y llenar el tablero con los nuevos valores, lo que hace es solicitar la ruta del archivo y abrirlo, leer el contenido y enviar al macro obtenerArregloTablero el texto leido y los arreglos que simulan las filas y columnas. La macro obtenerArregloTablero se explica mas adelante.

```
OPENGAME:
    print msgOpenFile
    getRuta filepath
    openFile filepath, handlefile

    readFile SIZEOF buffread,buffread,handlefile
    closeFile handlefile
    print buffread

    obtenerArregloTablero buffread, row8, row7, row6, row5, row4, row3, row2, row1
    getChar

    xor cx,cx
    MOV cx, 1
    jmp LOOPJUEGO
```

SHOW

Esta etiqueta se encarga de generar el archivo html del tablero actual, para ello solicita el nombre del archivo y envia a la macro Wriehtml los parámetros necesarios para imprimir el archivo. De igual manera, la macro writehtml se explica mas adelate

```
SHOW:

    obtenerFecha dia, mes, ano, hora, minuto

    print msgcreatefile
    getRuta filepath
    createFile filepath,handlefile
    openhtml filepath, handlefile

    ;abrir html
    whritehtml SIZEOF abrirhtml, abrirhtml, handlefile
    whritehtml SIZEOF abrirfecha,abrirfecha,handlefile

    whritehtml SIZEOF dia, dia, handlefile
    whritehtml SIZEOF slash, slash, handlefile

    whritehtml SIZEOF mes, mes, handlefile
    whritehtml SIZEOF slash, slash, handlefile

    whritehtml SIZEOF ano, ano, handlefile
    whritehtml SIZEOF slash, slash, handlefile


    whritehtml SIZEOF slash1, slash1, handlefile
    whritehtml SIZEOF hora, hora, handlefile
    whritehtml SIZEOF slash2, slash2, handlefile
    whritehtml SIZEOF minuto, minuto, handlefile
```

CREATE

Este etiqueta se encarga de generar el archivo .arq que contendrá el save del tablero actual, solicita el nombre y envia los parámetros necesarios a la macro correspondiente

```
CREATE:
    print msgcreatefile
    getRuta filepath
    createFile filepath,handlefile
    openFile filepath, handlefile

    txttablero row8, handlefile
    txttablero row7, handlefile
    txttablero row6, handlefile
    txttablero row5, handlefile
    txttablero row4, handlefile
    txttablero row3, handlefile
    txttablero row2, handlefile
    txttablero row1, handlefile

    closeFile handlefile
    print msgfilesave
    getChar
    jmp NEWGAME
```

Tablero.inc

El archivo tablero.inc es un archivo que contiene macros necesarias para hacer funcionar el tablero entre ellas

printTablero:

Se encarga de pintar el tablero en la pantalla, recibe como parámetros el arreglos a leer y dependiendo de lo que se encuentre dentro del arreglo, pinta una cosa u otra

```
printTablero macro buffer, row  ; Rcor
LOCAL CONT, FIN,FN, FB, RN, RB, S
PUSH SI
PUSH AX

xor si,si
print marc
print row ;imprime la columna, 8,7,6,.

CONT:
    cmp si,8
    je FIN

    cmp buffer[si], 101b
    je FN
    cmp buffer[si], 100b
    je FB
    cmp buffer[si], 111b
    je RN
    cmp buffer[si], 110b
    je RB
```

PrintFichas

Estas macros se encargan de pintar las letra en el tablero, en lugar de los 0 y 1, muestra como tal las letras FNEG o FBLC para las fichas negras y blancas respectivamente

```
PrintFichaNegra macro jumper
    print fneg
    inc si
    jmp jumper
endm
```

txttablero

Esta macro se encarga de ayudar a crear el archivo que guarda el tablero actual, recibe como parámetro, el arreglo de cada fila y escribe 0,1,2 si lo que contiene el arreglo es un espacio, una ficha negra o una ficha blanca respectivamente

```
txttablero macro buffer, handle
LOCAL CONT, FIN,FN, FB, RN, RB, S
PUSH SI
PUSH AX
xor si,si

CONT:
    cmp si,8
    je FIN

    cmp buffer[si], 101b
    je FN
    cmp buffer[si], 100b
    je FB
    cmp buffer[si], 111b
    je RN
    cmp buffer[si], 110b
    je RB
    jmp S
    FN:
        writeFile SIZEOF txtfneg, txtfneg, handle
        inc si
        jmp CONT
    FB:
        writeFile SIZEOF txtfblc, txtfblc, handle
        inc si
        jmp CONT
    S:
        writeFile SIZEOF txtvoid, txtvoid, handle
```

obtenerArregloTablero

esta macro realiza la operación inversa a exttablero, esta macro, se encarga de armar el tablero según el archivo de entrada, lo lee, verifica si viene un 0, 1 o 2 y coloca en el arreglo correspondiente el valor necesario para que pueda ser leído en la pantalla

```
obtenerArregloTablero macro texto, arreglo8, arreglo7, arreglo6, arreglo5, arreglo4, arreglo3, arreglo2, arreglo1
    LOCAL LOO, LOO71, LOO72, LLO61, LOO62, LOO51, LOO52, LOO41, LOO42, LOO31, LOO32, LOO21, LOO22, LOO11, LOO12,FIN
    xor si, si
    xor di, di
    LOO:
        cmp si,8
        je LOO71
        recrearTablero arreglo8, texto
        jmp LOO

    LOO71:
        xor SI, SI
        JMP LOO72

    LOO72:
        cmp si,8
        je LOO61
```

recrearTablero

Esta macro es complemento de la macro anterior, se encarga de agregar al arreglo el valor correspondiente para simular las fichas o espacios que haya. Recibe como parámetro el arreglo a llenar y el texto por leer, dependiendo de lo que haya en el texto, así se inserta en el arreglo

```
recrearTablero macro buffer, texto
    LOCAL RECREARNEGRA, RECREARBLANCA, RECREARSPACIO, FIN

    cmp texto[di], '1'
    je  RECREARNEGRA
    cmp texto[di], '2'
    je  RECREARBLANCA
    cmp texto[di], '0'
    je  RECREARSPACIO
    jmp FIN
    RECREARNEGRA:
        NEGRA buffer, SI
        inc si
        jmp FIN
    RECREARBLANCA:
        BLANCA buffer, SI
        inc si
        jmp FIN
    RECREARSPACIO:
        ESPACIO buffer, SI
        inc si
        jmp FIN
    FIN:
        inc di

endm
```

Mover.inc

Este archivo también es un archivo de macros que tienen como principal función la movilidad sobre el tablero, cuenta con las siguientes macros:

moverFichaBlanco y moverFichaNegro

Su principal función es esa, reciben el turno dependiendo de quien sea el turno y mueve ya sea las fichas negras o blancas.

```
moverFichaBlanco macro turno
    LOCAL ORIGEN,EXIT
        CMP comando[2],','
        JE ORIGEN
        JMP EXIT
        xor di,di
        xor si,si
        ORIGEN:
            moveOrigen
            moveBlanca
        EXIT:
endm

moverFichaNegro macro turno
    LOCAL ORIGEN,EXIT
        CMP comando[2],','
        JE ORIGEN
        JMP EXIT

        ORIGEN:
            moveOrigen
            moveNegro
        EXIT:
endm
```

moverOrigen y moverDestino

se encarga de determinar el origen del movimiento, es decir, la posición de donde se quiere mover y la otra se encaga de determinar la posición a la que se quiere llegar

```
moveOrigen macro
    LOCAL getRow1,getRow2,getRow3,getRow4,getRow5,getRow6,getRow7,getRow8,EXIT, ERROR
            CMP comando[1],'1'
            JE getRow1
            CMP comando[1],'2'
            JE getRow2
            CMP comando[1],'3'
            JE getRow3
            CMP comando[1],'4'
            JE getRow4
            CMP comando[1],'5'
            JE getRow5
            CMP comando[1],'6'
            JE getRow6
            CMP comando[1],'7'
```

writeBlanca y writeNegra

coloca los valores de ficha negra 100h o blanca 101h en el arreglo en la posición a al que se quería llegar.

```
writeBlanca macro row
    LOCAL dest1,dest2,dest3,dest4,dest5,dest6,dest7,dest8,EXIT
        CMP comando[3],'A'
        JE dest1
        CMP comando[3],'B'
        JE dest2
        CMP comando[3],'C'
        JE dest3
        CMP comando[3],'D'
        JE dest4
        CMP comando[3],'E'
        JE dest5
        CMP comando[3],'F'
        JE dest6
        CMP comando[3],'G'
        JE dest7
        CMP comando[3],'H'
        JE dest8
```

Delete

Se encarga de vaciar la posición de partida, agrega 000h al arreglo en la posición de origen

```
delete macro row
    LOCAL F1,F2,F3,F4,F5,F6,F7,F8,EXIT
        CMP comando[0],'A'
        JE F1
        CMP comando[0],'B'
        JE F2
        CMP comando[0],'C'
        JE F3
        CMP comando[0],'D'
        JE F4
        CMP comando[0],'E'
        JE F5
        CMP comando[0],'F'
        JE F6
        CMP comando[0],'G'
        JE F7
        CMP comando[0],'H'
        JE F8
```

Html.inc

El ultimo de los archivos macros es el siguiente, se encarga de generar el reporte html del tablero actual del juego, la macros con las que cuenta son las siguiente

Htmltablero:

Recibe el arreglo que representa cada fila, lo recorre y agrega al archivo el texto html que representara la ficha, ya sea negra, blanca o espacio

```
htmltablero macro buffer, handle
LOCAL CONT, FIN,FN, FB, RN, RB, S
PUSH SI
PUSH AX
whritehtml SIZEOF tropen, tropen, handle
xor si,si

CONT: ···
FIN:
whritehtml SIZEOF trclose, trclose, handle
POP AX
POP SI
xor si,si
endm
```

createFileHmtl

se encarga de generar el archivo, recibe el texto completo para poder generarlo.

```
createFileHtml macro buffer,handle
mov ah,3ch
mov cx,00h
lea dx,buffer
int 21h
mov handle,ax
jc ErrorCrear
endm
```

Openhtml

Abre el archivo html para poder escribir en el. Recibe como parámetro, la ruta o el nombre del archivo.

```
openhtml macro ruta,handle
mov ah,3dh
mov al,10b
lea dx,ruta
int 21h
mov handle,ax
jc ErrorAbrir
endm
```

Whritehtml

Sirve para escribir dentro del archivo html

```
whritehtml macro numbytes,buffer,handle
    mov ah, 40h
    mov bx,handle
    mov cx,numbytes
    lea dx,buffer
    int 21h
    jc ErrorEscribir
endm
```