MANUAL TÉCNICO

Macro TableroInicial

Este macro llena el tablero de damas inicial colocando los valores en ASCII de los caracteres dentro del arreglo del tablero y posicionándolos de con condicionales dentro de un ciclo para poder obtener las posiciones iniciales del juego.

```
;jmp FinMientras

PrintNegras: ; INICIA ALMACENAMIENTO DE FICHAS NEGRAS
cmp di, 0
je AlineacionNegra1
jmp AlineacionNegra2

    AlineacionNegra1:
        cmp ah,0
        je PrintEspacio ; je == 0
        jmp PrintFichaNegra

    AlineacionNegra2:
        cmp ah,0
        je PrintFichaNegra ; je == 0
        jmp PrintEspacio

    PrintFichaNegra:
        mov tablero[si], 1
        jmp Mientras

SegundoIf:
    cmp si,40
    jg PrintBlancas
    jmp PrintEspacio
```

```
PrintBlancas: ; INICIA ALMACENAMIENTO DE FICHAS BLANCAS

cmp di, 0
je AlineacionBlanca1
jmp AlineacionBlanca2

    AlineacionBlanca1:
        cmp ah,0
        je PrintEspacio ; je == 0
        jmp PrintFichaBlanca

    AlineacionBlanca2:
        cmp ah,0
        je PrintFichaBlanca ; je == 0
        jmp PrintEspacio

    PrintFichaBlanca:
        mov tablero[si], 2
        jmp Mientras

PrintEspacio:
    mov tablero[si], 95
```

Macro CrearTableroHtml

Este macro crea, abre, escribe y cierra el archivo de reporte del juego a través de validaciones para saber que jugador esta generado el reporte.

```
CrearTableroHtml macro arreglo
LOCAL Mientras, FinMientras, ImprimirSalto, EscribirBlanca, EscribirNegra, Seguir, EscribirEspacio,


; SE CREA EL ARCHIVO
crear nombreArchivo, handler
escribir handler, abrirHtml, SIZEOF abrirHtml
escribir handler, estilosCSS1, SIZEOF estilosCSS1
escribir handler, saltoRep, SIZEOF saltoRep
escribir handler, estilosCSS2, SIZEOF estilosCSS2
escribir handler, saltoRep, SIZEOF saltoRep
escribir handler, estilosCSS3, SIZEOF estilosCSS3
escribir handler, saltoRep, SIZEOF saltoRep
escribir handler, estilosCSS4, SIZEOF estilosCSS4
escribir handler, saltoRep, SIZEOF saltoRep
escribir handler, abrirTabla, SIZEOF abrirTabla
escribir handler, abrirFila, SIZEOF abrirFila

escribir handler, abrirParrafo, SIZEOF abrirParrafo
escribir handler, nombresHtml, SIZEOF nombresHtml

limpiarTexto nombre1,SIZEOF nombre1,32,36 ; LIMPIA LA COPIA DEL NOMBRE DEL JUGADOR
escribir handler, nombre1, SIZEOF nombre1
limpiarTexto separadorHtml,SIZEOF separadorHtml,32,36 ; LIMPIA LA COPIA DEL NOMBRE DEL JUGADOR
escribir handler, separadorHtml, SIZEOF separadorHtml
limpiarTexto nombre2,SIZEOF nombre2,32,36 ; LIMPIA LA COPIA DEL NOMBRE DEL JUGADOR
escribir handler, nombre2, SIZEOF nombre2

escribir handler, saltoHtml, SIZEOF saltoHtml
escribir handler, punteoHtml, SIZEOF punteoHtml
cmp jueganNegras,1
je PunteoBlancas
limpiarTexto punteoTexto1,SIZEOF punteoTexto1,32,36 ; LIMPIA LA COPIA DEL PUNTEO
escribir handler, punteoTexto1, SIZEOF punteoTexto1
jmp SeguirEscribiendo
PunteoBlancas:
limpiarTexto punteoTexto2,SIZEOF punteoTexto2,32,36 ; LIMPIA LA COPIA DEL PUNTEO
escribir handler, punteoTexto2, SIZEOF punteoTexto2
SeguirEscribiendo:
escribir handler, cerrarParrafo, SIZEOF cerrarParrafo

push si
push di

xor si,si
mov si,1
xor di,di
```

```
Mientras:
    cmp si,65;17
    je FinMientras              ; while(si<=17){}

        mov al, arreglo[si]
        mov aux, al             ; print(arreglo[si])

        cmp aux,95
        je EscribirEspacio
        cmp aux,2
        je EscribirBlanca
        cmp aux,1
        je EscribirNegra
        cmp aux,20
        je EscribirReinaNegra
        cmp aux,21
        je EscribirReinaBlanca

        EscribirEspacio:
        escribir handler, abrirColumna, SIZEOF abrirColumna
        escribir handler, espacioRep, SIZEOF espacioRep
        escribir handler, cerrarColumna, SIZEOF cerrarColumna
        jmp Seguir
        ;print aux

        EscribirBlanca:
        escribir handler, fichaBlancaHtml, SIZEOF fichaBlancaHtml
        jmp Seguir

        EscribirNegra:
        escribir handler, fichaNegraHtml, SIZEOF fichaNegraHtml
        jmp Seguir

        EscribirReinaNegra:
        escribir handler, reinaNegraHtml, SIZEOF reinaNegraHtml
        jmp Seguir

        EscribirReinaBlanca:
        escribir handler, reinaBlancaHtml, SIZEOF reinaBlancaHtml
        jmp Seguir

        Seguir:

        cmp di,7;3
        je ImprimirSalto     ; if(di == 3){ Imprimir salto}
```

Macro IntToString

Este macro se encarga de convertir las salidas de entero a texto para poder mostrar en pantalla los números según su número ASCII, recibe el parámetro num que es el numero entero a convertir y number que es donde se va a almacenar el valor ya convertido, la conversión la realiza al sumar 48 al número que recibe esto lo hace la misma cantidad de veces que el tamaño del número entero.

cros.asm

```
;[49][46][49]
;ax = entera
;bx = decimal
IntToString macro num, number ; ax 1111 1111 1111 1111 -> 65535
LOCAL Inicio,Final,Mientras,MientrasN,Cero,InicioN
push si
push di
limpiar number,SIZEOF number,24h
mov ax,num ; ax = numero entero a convertir 23
cmp ax,0
je Cero
xor di,di
xor si,si
jmp Inicio

;ax = 123

Inicio:

    cmp ax,0 ;ax = 0
    je Mientras
    mov dx,0
    mov cx,10
    div cx ; 1/10 = ax = 0 dx = 2
    mov bx,dx
    add bx,30h ; 1 + 48 = ascii
    push bx
    inc di  ; di = 3
    jmp Inicio

Mientras:
    ;si = 0 , di = 3
    cmp si,di
    je Final
    pop bx
    mov number[si],bl
    inc si
    ;si = 2 di = 3
    jmp Mientras

Cero:
 mov number[0],30h
 jmp Final

Final:
pop di
pop si
```

Macro AnalizarComando

Dentro del macro AnalizarComando se hacen todas las validaciones de movimientos de las piezas y se cambian los valores dentro de las posiciones del arreglo, todo se realiza mediante comparaciones y saltos a etiquetas, también se hacen algunas operaciones aritméticas para llevar control de puntajes, turnos y repeticiones.

```
AnalizarComando macro com ; A1:B2 arreglo1 = [A][1]; arreglo2 = [B][2]
LOCAL Ver, Seguir, Mover, Error, ValidarNegras, ValidarBlancas, SaltarFicha, SoloMover

mov al,com[0]
mov posicionInicial[0],al ; arreglo1[0] = comando[0]

mov al,com[1]
mov posicionInicial[1],al ; arreglo1[1] = comando[1]

mov al,com[3]
mov posicionFinal[0],al ;  arreglo2[0] = comando[3]

mov al,com[4]
mov posicionFinal[1],al ;  arreglo2[1] = comando[4]


cmp posicionFinal[0],72 ; VALIDA QUE NO SE PASE DEL LIMITE DE LAS COLUMNAS
jg Error
cmp posicionFinal[1],56 ; VALIDA QUE NO SE PASE DEL LIMITE DE LAS FILAS
jg Error


ConversionCoordenadas posicionInicial ;convierte la coordenada y la guarda en al


xor si,si ;si tiene el indicie inicial
mov si,ax

ConversionCoordenadas posicionFinal ;convierte la coordenada y la guarda en al

xor di,di ;di tiene el indice final
mov di,ax


;aqui van validaciones

mov dx,di ; GUARDA EL INIDICE FINAL EN dx
mov cx,si ; GUARDA EL INDICE INICIAL EN cx

sub cx,dx ; cx = cx - dx

cmp tablero[si],20
je ValidarMovReina
cmp tablero[si],21
je ValidarMovReina

cmp jueganNegras,1
je ValidarNegras
```

```
cmp saltaFicha,1
je SaltarFicha
jmp SoloMover

SaltarFicha:
mov bx,si ; bx = posicion inicial
sub bx,restador ; posicion inicial - restador

cmp jueganNegras,1 ; juegan negras == 1
jl validarComidaBlancas ; SI ES MENOR ENTONCES ES 0
cmp tablero[bx],1 ; COMPARA SI LA FICHA A COMER ES DEL COLOR CONTRARIO
je Error ;SI ES DEL MISMO COLOR LA FICHA QUE SALTA ENTONCES SALTA ERROR
jmp limpiarEspacio
validarComidaBlancas:
cmp tablero[bx],2
je Error
jmp limpiarEspacio

limpiarEspacio:
mov tablero[bx],95 ; [ficha] = _

cmp jueganNegras, 0 ; jueganNegras == 0
je IncrementoJugador1
inc punteoJugador2 ; AUMENTAR EL PUTNEO DEL JUGADOR 2
jmp SoloMover
IncrementoJugador1:
inc punteoJugador1 ; AUMENTAR EL PUNTEO DEL JUGADOR 1


SoloMover:
xor ax,ax
mov al, tablero[si] ;al = arreglo[si]
; ---VERIFICA QUE NO EXISTA UNA FICHA EN ESA POSICION---
cmp tablero[di],1
je Error
cmp tablero[di],2
je Error
; ---VERIFICA QUE NO EXISTA UNA FICHA EN ESA POSICION---
mov tablero[si],95 ;arreglo[si] = '_'
mov tablero[di],al ;arreglo[di] = arreglo[si]

cmp jueganNegras, 1
je ValidarReinaN
jmp ValidarReinaB
jmp Seguir

ValidarReinaN:
cmp di,58
```