

Manual Técnico

JUEGO DE DAMAS

Elaborado por: Mariana Sic 201504051

Encabezado

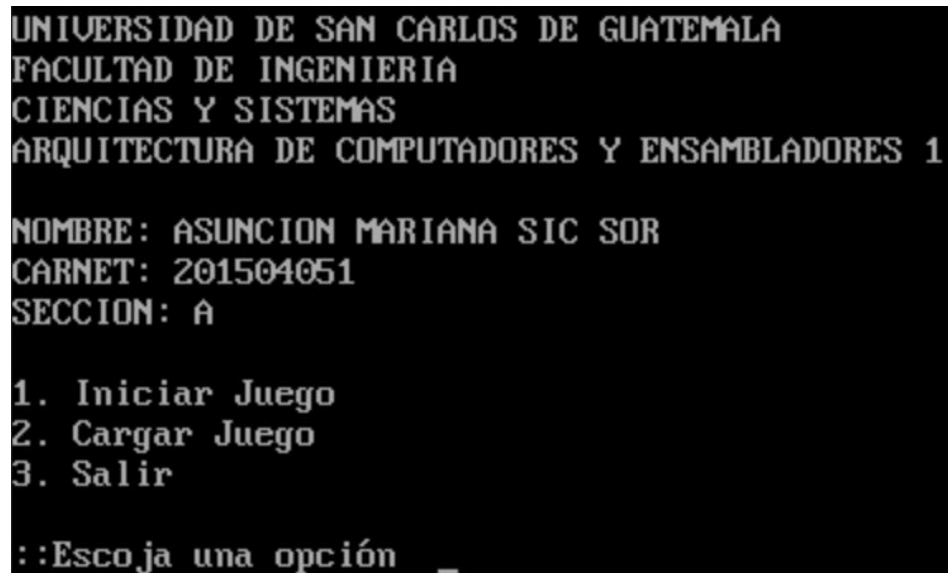
Para el encabezado se utilizaron tres variables: “encab”, “enc”, “encab2”. Las cuales tienen el contenido solicitado para el encabezado

```
encab db 10,10,13,'UNIVERSIDAD DE SAN CARLOS DE GUATEMALA',10,13,'FACULTAD  
DE INGENIERIA',10,13,'CIENCIAS Y SISTEMAS',10,13,'ARQUITECTURA DE  
COMPUTADORES Y ENSAMBLADORES 1','$'
```

```
enc db 10,10,13,'NOMBRE: ASUNCION MARIANA SIC SOR',10,13,'CARNET:  
201504051',10,13,'SECCION: A','$'
```

```
encab2 db 10,13,10,13,'1. Iniciar Juego',10,13,'2. Cargar Juego',10,13,'3.  
Salir',10,13,10,13,'::Escoja una opci',162,'n ','$'
```

Dando como resultado:



```
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA  
FACULTAD DE INGENIERIA  
CIENCIAS Y SISTEMAS  
ARQUITECTURA DE COMPUTADORES Y ENSAMBLADORES 1  
  
NOMBRE: ASUNCION MARIANA SIC SOR  
CARNET: 201504051  
SECCION: A  
  
1. Iniciar Juego  
2. Cargar Juego  
3. Salir  
  
::Escoja una opción _
```

Donde se debe de ingresar la opción deseada, estas opciones se detallan en las siguientes secciones:

1. Opción 1: Se inicia el juego. [Ver Detalle](#)
2. Opción 2: Cargar Juego. [Ver Detalle](#)
3. Opción 3: Salir. [Ver Detalle](#)

Jugabilidad

Para el control del movimiento de fichas, si el jugador movió ficha o no, si el turno es de las fichas blancas o negras y para los comandos especiales se utilizó un arreglo llamado “comando” de la siguiente manera:

POS	COMANDO
0	POSICIÓN EN COLUMNAS (COLUMNAS A -H)
1	POSICIÓN EN COLUMNAS (EN DADO CASO VENGA COMA, COLUMNAS A-H)
2	1 = TURNO DE FICHAS BLANCAS, 0 = TURNO DE FICHAS NEGRAS
3	1 = HUBO MOVIMIENTO DE FICHAS, 0 = NO HUBO MOVIMIENTO DE FICHAS
4	1 = VIENE COMA EN EL MOVIMIENTO (EJ. F6,H2), 0 = NO VIENE COMA
5	POSICIÓN EN FILAS (FILAS 1-8)
6	POSICIÓN EN FILAS (EN DADO CASO VENGA COMA, FILAS 1-8)
7	1111 = EXIT , 1110 = SAVE , 1101 = SHOW

El control de toda la matriz se llevó a través de ocho arreglos, el cual cada uno representa una fila en la matriz del tablero:

```
pos1 db 9 dup(32), '$'
pos2 db 9 dup(32), '$'
pos3 db 9 dup(32), '$'
pos4 db 9 dup(32), '$'
pos5 db 9 dup(32), '$'
pos6 db 9 dup(32), '$'
pos7 db 9 dup(32), '$'
pos8 db 9 dup(32), '$'
```

Al inicio, los arreglos se encuentran con el carácter de espacio en blanco, es decir, 32 en código ASCII

Conforme el juego avance, los arreglos se irán llenando conforme la siguiente tabla: (Ausencia representa el código 32 ASCII)

1	1	FB (FICHA BLANCA)
1	0	FN (FICHA NEGRA)
0	1	RB (REINA BLANCA)
0	0	RN (REINA NEGRA)
		AUSENCIA (ASCII 32)

Como al inicio los arreglos están con ausencia, se utilizó un macro "llenarInicial" el cual coloca las fichas en la posición inicial:

```
llenarInicial macro n1,n2,n3,b1,b2,b3
    mov b1[0],11b
    mov b1[2],11b
    mov b1[4],11b
    mov b1[6],11b
    mov b2[1],11b
    mov b2[3],11b
    mov b2[5],11b
    mov b2[7],11b
    mov b3[0],11b
    mov b3[2],11b
    mov b3[4],11b
    mov b3[6],11b
    mov n2[0],10b
    mov n2[2],10b
    mov n2[4],10b
    mov n2[6],10b
    mov n1[1],10b
    mov n1[3],10b
    mov n1[5],10b
    mov n1[7],10b
    mov n3[1],10b
    mov n3[3],10b
    mov n3[5],10b
    mov n3[7],10b
endm
```

Impresión de Tablero en Consola

Para la impresión del tablero, se creó un arreglo auxiliar de 29 posiciones de la siguiente manera:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
#	9																										10	13

Donde:

9 = tabulador en código ASCII

10 = salto de línea en código ASCII

13 = retorno de carro en código ASCII

= número de fila que corresponde (1-8)

| = símbolo para división entre columnas

espacio en blanco = lugares vacíos para colocar las fichas que corresponden

Finalmente, se utilizaron dos macros para ir imprimiendo, en el primero se llena el arreglo anteriormente descrito según el contenido de los ocho arreglos:

```
llenarArr macro arr,entrada
    local CONTINUE, FIN, FB, FN, RB, RN, SPACE
    mov arr[0],9
    mov arr[1],bl
    mov arr[2],32
    mov arr[3],32
    mov arr[4],179 ;|

    mov di,5
    mov si,0
    CONTINUE:
        cmp si,8
        je FIN
        mov al,entrada[si]
        cmp al,11b
        je FB
        cmp al,10b
        je FN
        cmp al,01b
        je RB
        cmp al,00b
        je RN
        cmp al,32
        je SPACE
```

```
FB:
    mov arr[di], 'F'
    inc di
    mov arr[di], 'B'
    inc di
    mov arr[di], 179
    inc di
    inc si
    jmp CONTINUE
```

```
FN:
    mov arr[di], 'F'
    inc di
    mov arr[di], 'N'
    inc di
    mov arr[di], 179
    inc di
    inc si
    jmp CONTINUE
```

```
RB:
    mov arr[di], 'R'
    inc di
    mov arr[di], 'B'
    inc di
    mov arr[di], 179
    inc di
    inc si
    jmp CONTINUE
```

```
RN:
    mov arr[di], 'R'
    inc di
    mov arr[di], 'N'
    inc di
    mov arr[di], 179
    inc di
    inc si
    jmp CONTINUE
```

```
SPACE:
    mov arr[di], 32
    inc di
    mov arr[di], 32
    inc di
    mov arr[di], 179
    inc di
    inc si
```

```

        jmp CONTINUE

FIN:
    mov al,10
    mov arr[di],al
    inc di
    mov al,13
    mov arr[di],al
    inc di
    mov al,'$'
    mov arr[di],al
endm

```

Y por último, en el segundo macro se limpia el arreglo de 28 posiciones para luego llenarlo ocho veces e ir imprimiendo cada vez, de la siguiente manera:

```

imprimirTablero macro p8,p7,p6,p5,p4,p3,p2,p1,arr,g
    cleanArr arr, sizeof arr
    mov bl,'8'
    llenarArr arr,p8
    print arr
    print g

    cleanArr arr, sizeof arr
    mov bl,'7'
    llenarArr arr,p7
    print arr
    print g

    cleanArr arr, sizeof arr
    mov bl,'6'
    llenarArr arr,p6
    print arr
    print g

    cleanArr arr, sizeof arr
    mov bl,'5'
    llenarArr arr,p5
    print arr
    print g

    cleanArr arr, sizeof arr
    mov bl,'4'
    llenarArr arr,p4
    print arr
    print g

    cleanArr arr, sizeof arr

```

```

mov bl,'3'
llenarArr arr,p3
print arr
print g

cleanArr arr, sizeof arr
mov bl,'2'
llenarArr arr,p2
print arr
print g

cleanArr arr, sizeof arr
mov bl,'1'
llenarArr arr,p1
print arr

cleanArr arr, sizeof arr
endm

```

El resultado final de esto es (en la posición inicial):

	A	B	C	D	E	F	G	H
8		FN		FN		FN		FN
7	FN		FN		FN		FN	
6		FN		FN		FN		FN
5								
4								
3	FB		FB		FB		FB	
2		FB		FB		FB		FB
1	FB		FB		FB		FB	
	A	B	C	D	E	F	G	H

Determinar siguiente turno

Primero verifica cual es el turno actual y también si el anterior turno movió la ficha que le correspondía. De lo contrario, seguirá en el turno actual hasta que el jugador mueva una ficha de su color.

```
sigTurno macro
    local SIGTURNO, TURNOBLANCA, TURNONEGRA, SWBN, SWNB, FIN

    SIGTURNO:
        cmp comando[2],1b
        je TURNOBLANCA
        cmp comando[2],0b
        je TURNONEGRA

    TURNOBLANCA:
        cmp comando[3],1b
        je SWBN
        print msjTBlancas
        jmp FIN

    SWBN:
        mov comando[2],0b
        mov comando[3],0b
        jmp SIGTURNO

    TURNONEGRA:
        cmp comando[3],1b
        je SWNB
        print msjTNegras
        jmp FIN

    SWNB:
        mov comando[2],1b
        mov comando[3],0b
        jmp SIGTURNO

    FIN:
        ObtenerTexto arregloAux

endm
```

Y para el movimiento se utilizan los siguientes macros:

```
movimiento macro accion, errMV
    local FIN, ERROR, HAYCOMA, ERRORTURNO, MOVERFICHA, REFRESH;, NOHAYCOMA
```



```

mov comando[7],00b
cmp accion[0],0
jl ERROR
cmp accion[0],8
jge ERROR
cmp accion[5],0
jle ERROR
cmp accion[5],8
jg ERROR

cmp accion[4],1b
je HAYCOMA
;cmp accion[4],0b
;je NOHAYCOMA

HAYCOMA:
    mov cx,1
    mov dx,6
    cmp accion[1],0
    jl ERROR
    cmp accion[1],8
    jge ERROR
    cmp accion[6],0
    jle ERROR
    cmp accion[6],8
    jg ERROR

    leerFichaActual
    cmp ah,100b
    je MOVERFICHA
    cmp ah,10b
    je MOVERFICHA
    cmp ah,00b
    je MOVERFICHA
    jne ERRORTURNO
    jmp FIN

MOVERFICHA:
    leerDestino
    cmp comando[7],1100b
    jne REFRESH
    jmp FIN

ERROR:
    print errMV
    jmp FIN

REFRESH:
    actualizarMovimiento

```

```

        jmp FIN

ERRORTURNO:
        print errorTurno

FIN:
        getChar
endm

leerFichaActual macro
        local ENPOS1, ENPOS2, ENPOS3, ENPOS4, ENPOS5, ENPOS6, ENPOS7, ENPOS8,
TURNOCORRECTO

        xor bx,bx
        mov bl,comando[0] ;LETRA -> posNUMERO[LETRA]
        mov si,bx
        mov bl,comando[5] ;NUMERO -> posNUMERO

        push bx ; numero de arreglo a modificar
        cmp bl,1
        je ENPOS1
        cmp bl,2
        je ENPOS2
        cmp bl,3
        je ENPOS3
        cmp bl,4
        je ENPOS4
        cmp bl,5
        je ENPOS5
        cmp bl,6
        je ENPOS6
        cmp bl,7
        je ENPOS7
        cmp bl,8
        je ENPOS8

ENPOS1:
        mov al,pos1[si]
        jmp TURNOCORRECTO

ENPOS2:
        mov al,pos2[si]
        jmp TURNOCORRECTO

ENPOS3:
        mov al,pos3[si]
        jmp TURNOCORRECTO

ENPOS4:

```

```

        mov al,pos4[si]
        jmp TURNOCORRECTO

ENPOS5:
        mov al,pos5[si]
        jmp TURNOCORRECTO

ENPOS6:
        mov al,pos6[si]
        jmp TURNOCORRECTO

ENPOS7:
        mov al,pos7[si]
        jmp TURNOCORRECTO

ENPOS8:
        mov al,pos8[si]
        jmp TURNOCORRECTO

TURNOCORRECTO:
        push si ;posicion en el arreglo
        xor ah,ah
        push ax
        mov ah,comando[2]
        add ah,al
endm

leerDestino macro
    local IMPAR, PAR, MOVALIDO, ENPOS1, ENPOS2, ENPOS3, ENPOS4, ENPOS5,
    ENPOS6, ENPOS7, ENPOS8, ERROR, FIN

    mov si,cx
    mov di,dx

    xor bx,bx
    mov bl,comando[si] ;LETRA -> posNUMERO[LETRA]
    mov si,bx
    mov bl,comando[di] ;NUMERO -> posNUMERO

    cmp bl,1
    je IMPAR
    cmp bl,2
    je PAR
    cmp bl,3
    je IMPAR
    cmp bl,4
    je PAR
    cmp bl,5
    je IMPAR

```

```

cmp bl,6
je PAR
cmp bl,7
je IMPAR
cmp bl,8
je PAR

IMPAR:
    cmp si,0
    je MOVALIDO
    cmp si,2
    je MOVALIDO
    cmp si,4
    je MOVALIDO
    cmp si,6
    je MOVALIDO

PAR:
    cmp si,1
    je MOVALIDO
    cmp si,3
    je MOVALIDO
    cmp si,5
    je MOVALIDO
    cmp si,7
    je MOVALIDO

MOVALIDO:
    cmp bl,1
    je ENPOS1
    cmp bl,2
    je ENPOS2
    cmp bl,3
    je ENPOS3
    cmp bl,4
    je ENPOS4
    cmp bl,5
    je ENPOS5
    cmp bl,6
    je ENPOS6
    cmp bl,7
    je ENPOS7
    cmp bl,8
    je ENPOS8

ENPOS1:
    mov dx,1
    mov al,pos1[si]
    jmp TURNOCORRECTO

```

```

ENPOS2:
    mov al,pos2[si]
    jmp TURNOCORRECTO

ENPOS3:
    mov al,pos3[si]
    jmp TURNOCORRECTO

ENPOS4:
    mov al,pos4[si]
    jmp TURNOCORRECTO

ENPOS5:
    mov al,pos5[si]
    jmp TURNOCORRECTO

ENPOS6:
    mov al,pos6[si]
    jmp TURNOCORRECTO

ENPOS7:
    mov al,pos7[si]
    jmp TURNOCORRECTO

ENPOS8:
    mov dx,8
    mov al,pos8[si]
    jmp TURNOCORRECTO

TURNOCORRECTO:
    cmp al,32
    jne ERROR
    je FIN

ERROR:
    print errorBlanco
    mov comando[7],1100b
    jmp FIN

FIN:

```

endm

actualizarMovimiento macro

local ENPOS1, ENPOS2, ENPOS3, ENPOS4, ENPOS5, ENPOS6, ENPOS7, ENPOS8,
TURNOCORRECTO, INICIO, BORRARANTERIOR, FIN, REINABLANCA, REINANEGRA

```

pop ax
xor di,di

INICIO:
    cmp dx,8
    je REINABLANCA
    cmp dx,1
    je REINANEGRA
    cmp bl,1
    je ENPOS1
    cmp bl,2
    je ENPOS2
    cmp bl,3
    je ENPOS3
    cmp bl,4
    je ENPOS4
    cmp bl,5
    je ENPOS5
    cmp bl,6
    je ENPOS6
    cmp bl,7
    je ENPOS7
    cmp bl,8
    je ENPOS8

REINABLANCA:
    mov pos8[si],01b
    jmp TURNOCORRECTO

REINANEGRA:
    mov pos1[si],00b
    jmp TURNOCORRECTO

ENPOS1:
    mov pos1[si],al
    jmp TURNOCORRECTO

ENPOS2:
    mov pos2[si],al
    jmp TURNOCORRECTO

ENPOS3:
    mov pos3[si],al
    jmp TURNOCORRECTO

ENPOS4:
    mov pos4[si],al
    jmp TURNOCORRECTO

```

```

ENPOS5:
    mov pos5[si],al
    jmp TURNOCORRECTO

ENPOS6:
    mov pos6[si],al
    jmp TURNOCORRECTO

ENPOS7:
    mov pos7[si],al
    jmp TURNOCORRECTO

ENPOS8:
    mov pos8[si],al
    jmp TURNOCORRECTO

TURNOCORRECTO:
    cmp di,0
    je BORRARANTERIOR
    cmp di,1
    je FIN

BORRARANTERIOR:
    xor dx,dx
    pop si
    pop bx
    mov al,32
    inc di
    jmp INICIO

FIN:
    mov comando[3],1b
    mov comando[7],0000b

endm

```

JUEGO INICIADO

	A	B	C	D	E	F	G	H
8		FN		FN		FN		FN
7	FN		FN		FN		FN	
6		FN		FN				FN
5					FN			
4				FB				
3	FB		FB				FB	
2		FB		FB		FB		FB
1	FB		FB		FB		FB	
	A	B	C	D	E	F	G	H

: Turno Blancas G3,H4

Y realiza el movimiento indicado

JUEGO INICIADO

	A	B	C	D	E	F	G	H
8		FN		FN		FN		FN
7	FN		FN		FN		FN	
6		FN		FN				FN
5					FN			
4				FB				FB
3	FB		FB					
2		FB		FB		FB		FB
1	FB		FB		FB		FB	
	A	B	C	D	E	F	G	H

: Turno Negras

Comandos Especiales

EXIT

Con este comando, se sale del juego sin guardar el estado actual y regresa al Menú Principal

EXIT:

```
inc di
mov al,paso[di]
cmp al,'x'
je EXIT
cmp al,'i'
je EXIT
cmp al,'t'
jne ERROR
je FINEXIT
jmp FIN
```

Se visualiza de la siguiente manera:

```
::Turno Blancas exit

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERIA
CIENCIAS Y SISTEMAS
ARQUITECTURA DE COMPUTADORES Y ENSAMBLADORES 1

NOMBRE: ASUNCION MARIANA SIC SOR
CARNET: 201504051
SECCION: A

1. Iniciar Juego
2. Cargar Juego
3. Salir

::Escoja una opción _
```

SAVE

Para este comando, se utilizó el macro de guardarArq, donde se utiliza un arreglo para la escritura de la siguiente manera:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	,		,		,		,		,		,		,		10	13

Donde los espacios en blanco representan al código ASCII 32 y el 10 y 13 representan el salto de línea y el retorno de carro.

Las variables aparte del arreglo anteriormente descrito, son el mensaje de haber guardado correctamente el juego actual

```
filaArch db ' , , , , , , , , ',10,13
saveSuccess db 10,13, ' ',173,173,' Juego guardado con ',130,'xito !!!','$'
```

Luego en el macro, solo se va llenando el arreglo de “filaArch” según los ocho arreglos los cuales representan una fila en el tablero, para eso se utilizó un loop para que también se vaya escribiendo en el fichero especificado.

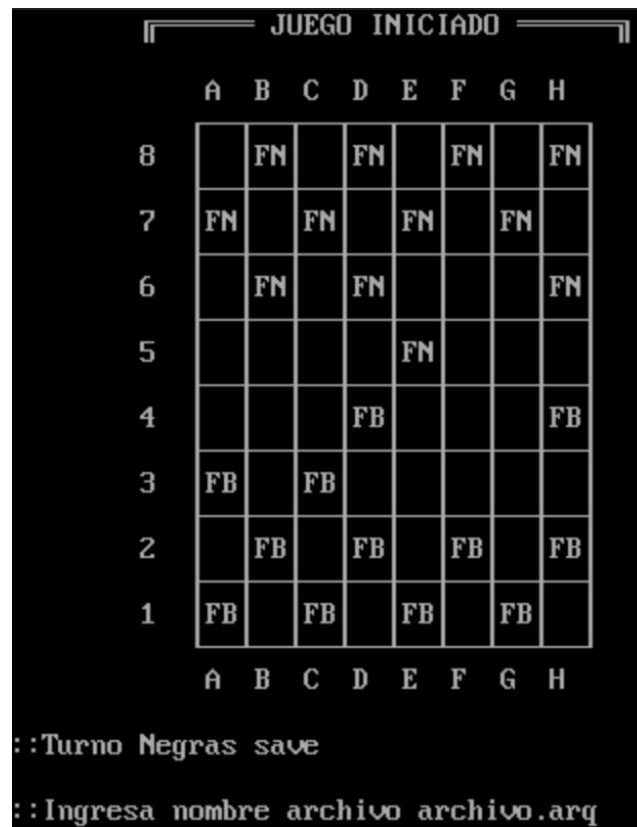
```
guardarArq macro posicion
    local FIRST, LAST
    xor si,si
    xor di,di
    mov cx,8
    FIRST:
        mov al,posicion[si]
        mov filaArch[di],al
        inc si
        inc di
        inc di
        loop FIRST

    escribirF handle2, sizeof filaArch, filaArch

    xor di,di
    mov cx,8
    LAST:
        mov filaArch[di],32
        inc di
        inc di
        loop LAST

endm
```

Entonces se pedirá que ingrese el nombre a guardar el archivo



Y en el archivo con extensión .arq se guardan los códigos ASCII de las fichas, según la tabla de las fichas, estarán separadas por comas.

```
ASM_arqui > P3 > ≡ ARCHIVO.ARQ
1  | ,□, ,□, , , ,□
2
3  | □, ,□, ,□, ,□,
4
5  | ,□, ,□, , , ,□
6
7  | , ,□, , , ,□,
8
9  | ,□, , , , , ,
10
11 | □, ,□, , , ,□,
12
13 | ,□, ,□, ,□, ,□
14
15 | □, ,□, ,□, ,□,
16
```

Page 10 of 10

Para la generación del estado actual del tablero se utilizaron las siguientes variables

[illegible]

Para la generacion de la fecha y hora, se utilizaron la interrupcion 21h con la funcion 2ah y 2ch y estos resultados se almacenaban en los arreglos de fechaHTML y horaHTML

escribirFecha macro handle, fecha, hora

```

; ; ; ; ; FECHA
MOV AH, 2AH ; To get System Date
INT 21H
MOV AL, DL ; Day is in DL
AAM
MOV BX, AX
CALL DISP
mov fecha[23], dl
mov fecha[24], al

MOV AH, 2AH ; To get System Date
INT 21H
MOV AL, DH ; Month is in DH
AAM
MOV BX, AX

```

```

CALL DISP
mov fecha[26],dl
mov fecha[27],al

MOV AH,2AH      ; To get System Date
INT 21H
ADD CX,0F830H   ; To negate the effects of 16bit value,
MOV AX,CX       ; since AAM is applicable only for AL (YYYY -> YY)
AAM
MOV BX,AX
CALL DISP
mov fecha[31],dl
mov fecha[32],al

escribirF handle, sizeof fecha, fecha

;;;;;;;;; HORA
MOV AH,2CH      ; To get System Time
INT 21H
MOV AL,CH       ; Hour is in CH
AAM
MOV BX,AX
CALL DISP
mov hora[26],dl
mov hora[27],al

MOV AH,2CH      ; To get System Time
INT 21H
MOV AL,CL       ; Minutes is in CL
AAM
MOV BX,AX
CALL DISP
mov hora[29],dl
mov hora[30],al

MOV AH,2CH      ; To get System Time
INT 21H
MOV AL,DH       ; Seconds is in DH
AAM
MOV BX,AX
CALL DISP
mov hora[32],dl
mov hora[33],al
endm

```

Y para graficar el tablero se utilizó una tabla con ocho filas y ocho columnas, se dividió la graficada por filas par y filas impares; esto quiere decir que las pares serían las filas 2,4,6 y 8 y las impares las filas 1,3,5 y 7. Para esto, los siguientes macros:

```
filaParHTML macro posicion
    local INICIO, FB, FN, RB, RN, ESCRIBIR, FIN, BLANK
    xor di,di
    xor si,si
    xor bx,bx

    INICIO:
        mov di,80
        mov cx,30
        inc si ;pos si: 1,3,5,7
        mov bx,si
        cmp bx,9
        je FIN
        mov al,posicion[si]
        push si
        xor si,si
        cmp al,11b
        je FB
        cmp al,10b
        je FN
        cmp al,01b
        je RB
        cmp al,00b
        je RN
        cmp al,32
        je BLANK
        pop si
        inc si
        jmp INICIO

    FB:
        mov dl,blancaHTML[si]
        mov filaParHTMLx4[di],dl
        inc si
        inc di
        loop FB
        jmp ESCRIBIR

    FN:
        mov dl,negraHTML[si]
        mov filaParHTMLx4[di],dl
        inc si
        inc di
        loop FN
        jmp ESCRIBIR
```

```

RB:
    mov dl,blancaReinaHTML[si]
    mov filaParHTMLx4[di],dl
    inc si
    inc di
    loop RB
    jmp ESCRIBIR

RN:
    mov dl,negraReinaHTML[si]
    mov filaParHTMLx4[di],dl
    inc si
    inc di
    loop RN
    jmp ESCRIBIR

BLANK:
    mov filaParHTMLx4[di],32
    inc si
    inc di
    loop BLANK
    jmp ESCRIBIR

ESCRIBIR:
    escribirF handle2, sizeof filaParHTMLx4, filaParHTMLx4
    pop si
    inc si
    jmp INICIO

FIN:
    xor si,si
    xor di,di
endm

filaImparHTML macro posicion
    local INICIO, FB, FN, RB, RN, ESCRIBIR, FIN, BLANK
    xor di,di
    xor si,si
    xor bx,bx

    INICIO:
        mov di,38
        mov cx,30
        mov bx,si
        cmp bx,8
        je FIN
        mov al,posicion[si]
        inc si ;pos si: 0,2,4,6

```

```
push si
xor si,si
cmp al,11b
je FB
cmp al,10b
je FN
cmp al,01b
je RB
cmp al,00b
je RN
cmp al,32
je BLANK
pop si
inc si
jmp INICIO
```

FB:

```
mov dl,blancaHTML[si]
mov filaImparHTMLx4[di],dl
inc si
inc di
loop FB
jmp ESCRIBIR
```

FN:

```
mov dl,negraHTML[si]
mov filaImparHTMLx4[di],dl
inc si
inc di
loop FN
jmp ESCRIBIR
```

RB:

```
mov dl,blancaReinaHTML[si]
mov filaImparHTMLx4[di],dl
inc si
inc di
loop RB
jmp ESCRIBIR
```

RN:

```
mov dl,negraReinaHTML[si]
mov filaImparHTMLx4[di],dl
inc si
inc di
loop RN
jmp ESCRIBIR
```

BLANK:


```

mov filaImparHTMLx4[di],32
inc si
inc di
loop BLANK
jmp ESCRIBIR

ESCRIBIR:
    escribirF handle2, sizeof filaImparHTMLx4, filaImparHTMLx4
pop si
inc si
jmp INICIO

FIN:
    xor si,si
    xor di,di
endm

```

Las imágenes que se utilizaron fueron:

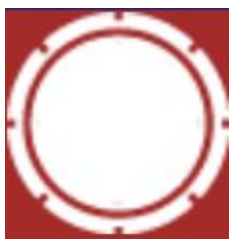
Para las fichas negras:



Para las reinas negras:



Para las fichas blancas:



Para las reinas blancas:



El reporte final del estado actual del juego, queda de la siguiente manera



Cargar Juego

Para la carga de juego, se utilizó la lógica inversa del comando especial [SAVE](#). Utilizando el siguiente macro:

```
cargaFichero macro posicion
    local LAST, FIRST

    xor di,di
    mov cx,8
    LAST:
        mov filaArch[di],32
        inc di
        inc di
        loop LAST

    leerF sizeof filaArch, filaArch, handle2

    xor si,si
    xor di,di
    mov cx,8
    FIRST:
        mov al,filaArch[di]
        mov posicion[si],al
```

```

inc si
inc di
inc di
loop FIRST

```

```
endm
```

Para que al final, se mande a llamar ocho veces con los ocho diferentes arreglos

CARGAJUEGO:

```

mov handle2,00h
print msjOpc2
print ingreseRuta
print aCargar
getRuta arregloAux
abrirF arregloAux, handle2
cargaFichero pos8
cargaFichero pos7
cargaFichero pos6
cargaFichero pos5
cargaFichero pos4
cargaFichero pos3
cargaFichero pos2
cargaFichero pos1
cerrarF handle2
mov handle2,00h
print loadSuccess
getChar
jmp INICIOJUEGO

```

Luego aparecera mensaje de éxito si se cargó correctamente el archivo al ingresar el archivo deseado

```

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERIA
CIENCIAS Y SISTEMAS
ARQUITECTURA DE COMPUTADORES Y ENSAMBLADORES 1

```

```

NOMBRE: ASUNCION MARIANA SIC SOR
CARNET: 201504051
SECCION: A

```

```

1. Iniciar Juego
2. Cargar Juego
3. Salir

```

```
::Escoja una opción 2
```

```
----- CARGAR JUEGO -----
```

```
::Ingresa nombre archivo a cargar archivo.arc
```

```
!! Juego cargado con éxito !!
```

Y se cargan las fichas en las posiciones guardadas

JUEGO INICIADO

	A	B	C	D	E	F	G	H
8		FN		FN				FN
7	FN		FN		FN		FN	
6		FN		FB				FN
5			FN				FN	
4		FN						
3	FB		FB				FB	
2		FB		FB		FB		FB
1	FB		FB		FB		FB	
	A	B	C	D	E	F	G	H

:Turno Negras

Salir

Para realizar la acción de salir del programa en general, se utilizó la función 4CH de la interrupción 21H junto con la etiqueta "SALIR" y un mensaje "msjOpc3" que se despide del programa

```
SALIR:
    print msjOpc3
    mov ah,4ch
    int 21h
```

Funciona de la siguiente manera:

NOMBRE: ASUNCION MARIANA SIC SOR
CARNET: 201504051
SECCION: A

1. Iniciar Juego
2. Cargar Juego
3. Salir

::Escoja una opción 3

===== ADIOS =====

C:\ASM_AR~1\P3>