Universidad Nacional de General Sarmiento

Organización del Computador I
Comisión Noche

Trabajo Práctico

Organización del Computador I
TP 1er Cuatrimestre 2018

Martha Semken
Mariano Vargas

APELLIDO Y NOMBRE	LEGAJO	EMAIL
Sánchez, Matías Alejandro	38.391.082/2015	mattisanchez94@gmail.com
Fernandez Medina, Mateo	36.829.764/2017	mateomef@gmail.com
Tula, Ignacio Mariano	35.226.620/2014	itula@campus.ungs.edu.ar

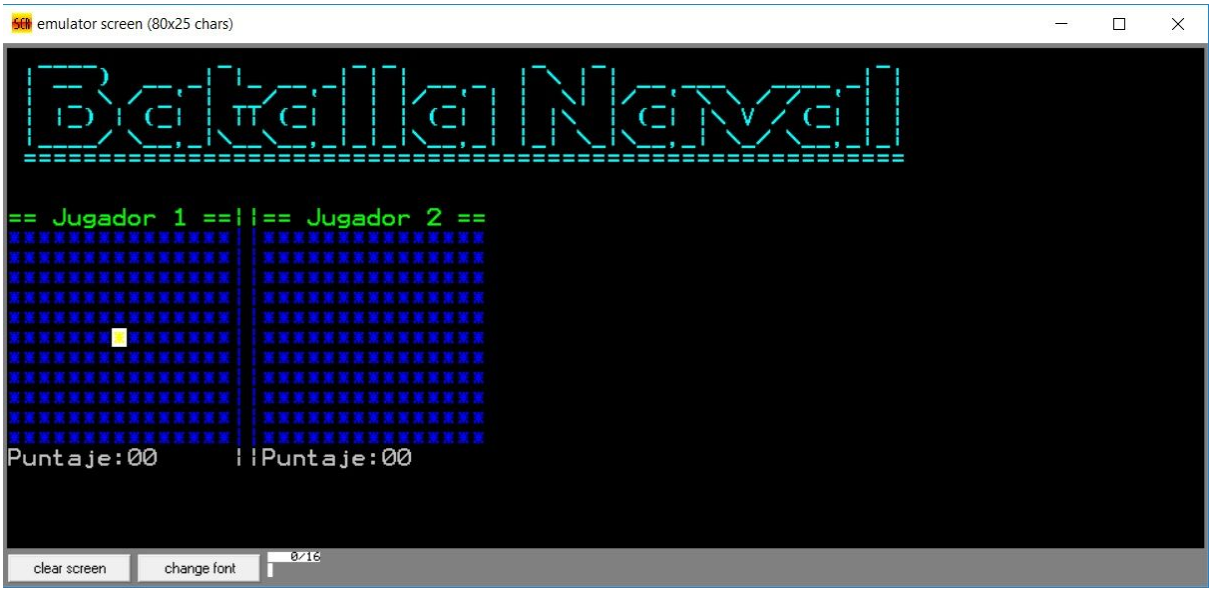
INTRODUCCIÓN

Se implementó sobre lenguaje ensamblador para Intel 8086/8088 un software o videojuego que emula el clásico juego de mesa “Batalla Naval”.

Esta implementación guarda en memoria dos matrices que representan el mar individual de cada jugador. Dichas matrices pueden contener espacios de mar vacíos y también porciones del mismo donde se encuentre el segmento de algún barco propio.

El objetivo del juego para cada jugador es poder encontrar primero los veinte valores de la matriz del contrincante donde se encuentran segmentos de barcos. Para lo cual van a turnarse para realizar disparos sobre el mar del oponente y develar si dicha posición era ocupada por un barco o si el disparo fue fallido.

Para esto el software realiza un uso de los periféricos de salida (la consola) y de entrada (el teclado) para mostrar información o recibir órdenes respectivamente.



PROBLEMAS HALLADOS Y RESOLUCIONES IMPLEMENTADAS

- La cuestión de realizar un mapeo sobre las posiciones de un cursor en pantalla con respecto a dos matrices cuyos formas de acceso a sus elementos no es posible mediante un par ordenado, sino mediante un índice único para cada elemento.

En primer lugar para resolver este problema se debe trabajar sobre la posición actual del cursor en pantalla, conocer sobre qué matriz se está moviendo y obtener luego el índice que ocupa en dicha matriz.

Conocer sobre qué matriz se está operando es sencillo conociendo el turno actual del jugador.

Por otro lado, el dibujo en pantalla sobre el que se mueve el cursor es estático o consistente en la cantidad de caracteres (posiciones) que posee.

Aquí dejamos la representación gráfica de la matriz tablero (que contiene ambos mares, que representa ambas matrices y además posee un apartado gráfico similar a una interfaz de usuario).

DH\DL	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	29	30	31	32	33
CARAC ->	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	|	|	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	CR	NL
9	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	29	30	31	32	33
10	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67
11	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101
12	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135
13	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169
14	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203
15	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237
16	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271
17	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303	304	305
18	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339

La primera columna de la ilustración anterior que numera del 9 al 18, representa las posiciones del cursor que ocupa el registro DH (fila actual de la consola).

La primera fila de la ilustración anterior que numera del 0 al 33, representa las posiciones del cursor que ocupa el registro DL (columna actual de la consola).

La segunda fila muestra cual es el caracter mostrado en pantalla en dicha posición.

El resto de la tabla indica la posición el índice que ocupa dicho caracter con respecto a la etiqueta “tablero”.

Entonces, a cualquier posición que queramos obtener su índice mediante el cursor, se le debe restar 9 al registro DH que indica la fila.

A dicho resultado, luego se debe multiplicar por 15, que son los elementos que pueden encontrarse en una fila completa de una matriz de mar individual.

Luego sumar a este resultado el valor del cursor en DL, que son los elementos que se encuentran en la fila incompleta de nuestra posición.

Por ejemplo, para ubicar el elemento identificado como 109 en la ilustración anterior pero que en realidad es la posición 52 de la matriz individual del mar del jugador 2.

Dicha ubicación es la fila 12 - 9 = 3. Luego 3 * 15 = 45. Y finalmente 45 + 7 = 52

Para convertir el cursor del disparo del jugador 2 en el índice de la matriz del jugador 1, el algoritmo es el mismo, con la excepción de que antes de sumar DL al resultado de la multiplicación, a este registro se le deben restar los 17 caracteres excedentes de la izquierda.

Se puede observar el comportamiento de cómo se resuelve este problema en los comentarios del código fuente entre las líneas 351 y 444.

- La cuestión del tablero de puntajes de dos dígitos, representados en caracteres ASCII y su consiguiente problema al realizar una suma.

Para conservar la estética del juego y aprovechando el diseño de las fuentes que es de ancho fijo para cada caracter, se decidió que el puntaje sea representado con dos dígidos, aún cuando el valor sea cero. Por lo tanto se tuvo que utilizar la representación del cero a la izquierda de su equivalente en ASCII, puesto que el número decimal 00 no existe.

Cómo el máximo valor obtenido es 20 porque luego se termina el juego, se tuvo que tener especial cuidado, cuando el puntaje actual es 9 o 19 y se tuviera que sumar uno.

Para cuando el valor sea 9 (es decir caracteres 30h y 39h en ASCII), no se debía realizar un incremento, sino que se debían cambiar los valores a 31h y 30h respectivamente. Similar para el caso de 19 puntos.

Esta cuestión se ve reflejada en las líneas del código fuente desde 601 hasta 717

- La cuestión de reflejar los cambios producidos por los disparos acertados o errados en el mar individual. Evitar que un doble disparo sobre una zona ya marcada como “S”, no vuelva a sumar puntaje falso.

Nuestra implementación, una vez que obtiene desde la posición del cursor en pantalla, el índice de la matriz de mar individual, debe compararse el valor resguardado en memoria.

Notamos que este valor, que inicialmente se encontraba estático, debía ser modificado en cada disparo, para preservar los disparos ya efectuados. El no realizar dichos cambios,

permitía doble disparos sobre posiciones del mar que ya se encontraban marcadas como “S” y que las subrutinas que calculan el puntaje sumaran un falso punto.

La implementación de la resolución a esta cuestión se encuentra en el código fuente entre las líneas 517 y 767.

Se decidió que sea un procedimiento llamado en las etapas del programa marcadas con las etiquetas “acierto” y “errar” para utilizar el valor del índice ya resguardado en el registro DI.

CONCLUSIÓN

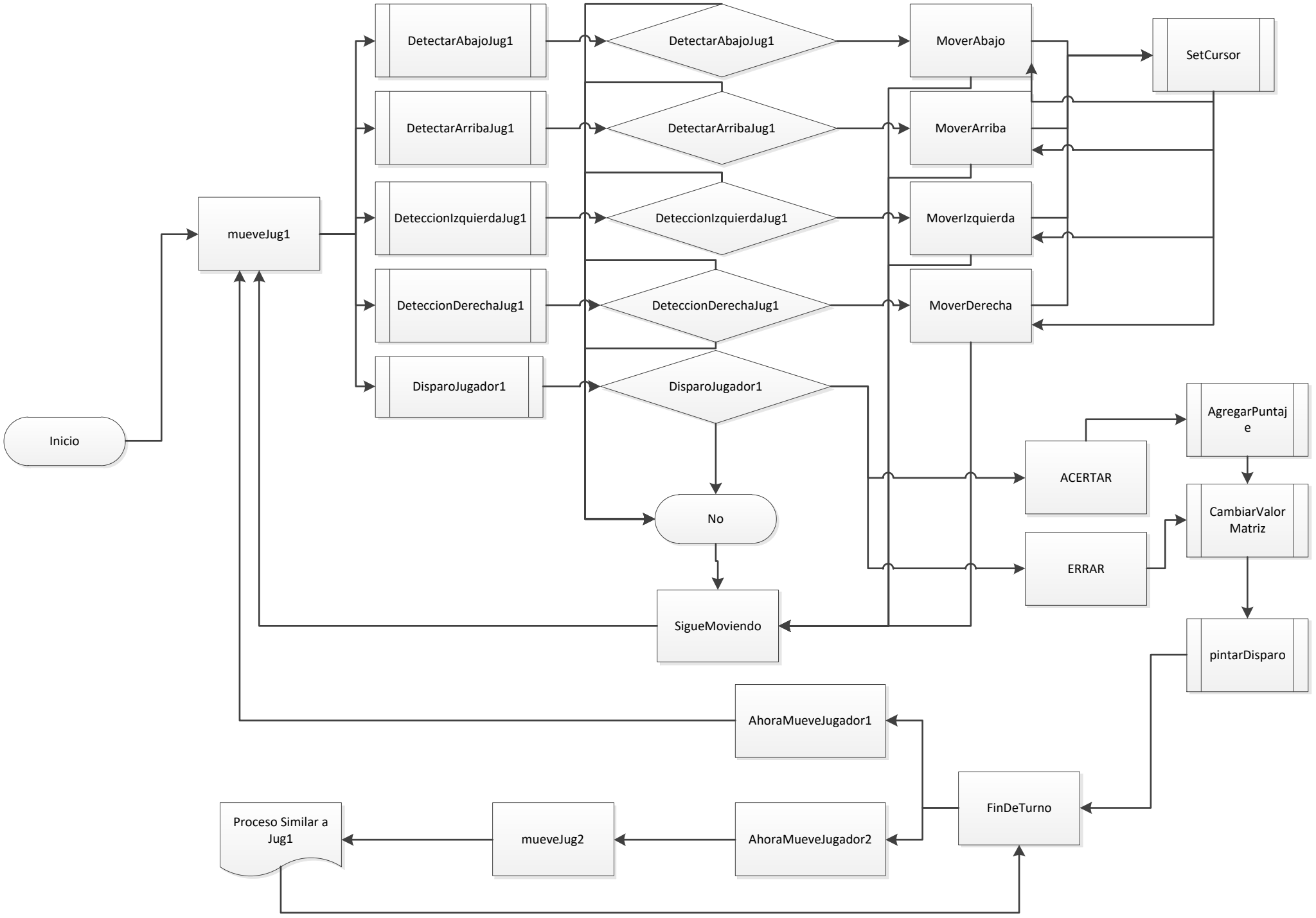
Quizás la conclusión más abreviada es que implementamos un software en lenguaje de bajo nivel. Pero dicha premisa esconde una serie de cuestiones de forma implícita.

Por ejemplo, se logró construir un videojuego utilizando las operaciones o instrucciones más básicas que un microprocesador puede realizar.

Si para realizar un programa, uno debe desarrollar una serie finita de pasos o instrucciones ordenadas, y que normalmente hacen uso de estructuras de control (bucles, condicionales) y principalmente se trabaja en un nivel de abstracción de la lógica.

En el caso del desarrollo del actual, el equipo de trabajo estuvo enfocado más en cuestiones del manejo de la memoria, de los registros limitados del procesador y en diseñar algoritmos más elementales que realizaran operaciones que quizás son más sencillas de realizar en un lenguaje de más alto nivel.

Como corolario, esto nos ayudó a comprender el funcionamiento de un computador en su nivel más esencial, a comprender que algunas estructuras que creíamos básicas, aún pueden ser deconstruidas en pequeños procesos más elementales aún y en llevar adelante su diseño e implementación.


DetectarAbajoJug1
DetectarArribaJug1
DeteccionIzquierdaJug1
DeteccionDerechaJug1
DisparoJugador1
DetectarAbajoJug1
DetectarArribaJug1
DeteccionIzquierdaJug1
DeteccionDerechaJug1
DisparoJugador1
MoverAbajo
MoverArriba
MoverIzquierda
MoverDerecha
SetCursor
mueveJug1
Inicio
No
ACERTAR
ERRAR
AgregarPuntaje
CambiarValorMatriz
pintarDisparo
SigueMoviendo
AhoraMueveJugador1
AhoraMueveJugador2
FinDeTurno
mueveJug2
Proceso Similar a Jug1

```
org 100h
jmp inicio


presentacionArte db "  ____        _        _ _          _   _                    _  ",  10,13
db                  " | __ )  __ _| |_ __ _| | | __ _  | \ | | __ ___   ____ _| | ",  10,13
db                  " |  _ \ / _` | __/ _` | | |/ _` | |  \| |/ _` \ \ / / _` | | ",  10,13
db                  " | |_) | (_| | || (_| | | | (_| | | |\  | (_| |\ V / (_| | | ",  10,13
db                  " |____/ \__,_|\__\__,_|_|_|\__,_| |_| \_|\__,_| \_/ \__,_|_| ",  10,13
db                  " ============================================================ ",  10,13 ,10,13, 10,13


  titulo db "== Jugador 1 ==||== Jugador 2 ==", 10,13
tablero  db "****************||****************", 10,13
         db "****************||****************", 10,13
         db "****************||****************", 10,13
         db "****************||****************", 10,13
         db "****************||****************", 10,13
         db "****************||****************", 10,13
         db "****************||****************", 10,13
         db "****************||****************", 10,13
         db "****************||****************", 10,13
         db "****************||****************", 10,13
         db "****************||****************", 10,13
         db "****************||****************", 10,13
         db "****************||****************", 10,13
puntaje  db "Puntaje:00     ||Puntaje:00    $", 10,13

exitoso   db 53h  ; Caracteres que representa el exito   "S"
fracasoso db 4Eh  ; Caracteres que representa el fracaso "N"
barquito db "#"   ; Caracteres que un fragmento de barco "#"

ComparaPosicion db 0


PuntajeJugador1 db "00$"
PuntajeJugador2 db "00$"

MensajeVictoriaCadena db "Ha ganado el jugador $"
Jugando db 1
MueveJugador db 0
```

```
botes_Jug1 db "**#####********"
           db "****************"
           db "*****##**********"
           db "****************"
           db "************#**"
           db "************#**"
           db "*#**********#**"
           db "*#**********#**"
           db "*#*****###*****"
           db "*#**************"
           db "*#**************"

botes_Jug2 db "**##************"
           db "****************"
           db "****************"
           db "*****#**********"
           db "*****#*****####*"
           db "#***#**********"
           db "#******#*******"
           db "#******#*******"
           db "#******#*******"
           db "#******#*******"
           db "#******#*******"
```

```
; ACA EL PROGRAMA
inicio:
; Pintar de cyan usando un carac NULL los espacios que ocuparia el encabezado ASCII
mov ah, 09h
mov al, 00h
mov bh, 0
mov bl, 1011b
mov cx, 480
int 10h

; mover cursor donde estaria el titulo del tablero
mov dh, 8
mov dl, 0
mov bh, 0
mov ah, 2
int 10h

; pintar de verde usando un carac NULL los espacios que ocuparia el titulo del tablero
mov ah, 09h
mov al, 00h
mov bh, 0
mov bl, 1010b
mov cx, 80
mov dh, 7
mov dl, 9
int 10h


; mover cursor donde estaria el mar
mov dh, 9
mov dl, 0
mov bh, 0
mov ah, 2
int 10h


; pintar de azul usando un carac NULL los espacios que ocuparia el mar
mov ah, 09h
mov al, 0
mov bh, 0
mov bl, 1001b
mov cx, 880
mov dh, 7
mov dl, 0
int 10h
```

```
151   ;mover cursor al cero para escribir caracteres que necesitamos que sean visibles
152   mov dh, 0
153   mov dl, 0
154   mov bh, 0
155   mov ah, 2
156   int 10h
157
158
159   ; Mostrar el arte ascii, el titulo y el tablero completo
160   mov dx, offset[presentacionArte]
161   mov bh, 0
162   mov bl, 03h
163   mov ah,9
164   int 21h
165
166
167   mov dh, 14
168   mov dl, 7
169   call setCursor
170
171   ;Hacer que el cursor parpadee en la posicion en la que se encuentre
172   mov ch, 0
173   mov cl, 7
174   mov bl, 1010b
175   mov ah, 1
176   int 10h
177
178
179
180   jmp mueveJug1
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
```

```
; ESTE LOOP ESCUCHA QUE TECLA PRESIONA EL JUGADOR 1
mueveJug1:
        mov ah, 00h
        int 16h

        cmp al, 78h     ;Compara que tecla se presiono, si es 'x'
        je DetectarAbajoJug1

        cmp al, 77h     ;Compara que tecla se presiono, si es 'w'
        je DetectarArribaJug1

        cmp al, 61h     ;Compara que tecla se presiono, si es 'a'
        je DeteccionIzquierdaJug1

        cmp al, 64h     ;Compara que tecla se presiono, si es 'd'
        je DeteccionDerechaJug1

        cmp al, 73h     ;Compara que tecla se presiono, si es 's'
        je DisparoJugador1



        jmp mueveJug1        ;Sino, no hace nada y vuelve a pedir una tecla



 ; ESTE LOOP ESCUCHA QUE TECLA PRESIONA EL JUGADOR 2

mueveJug2:
        mov ah, 00h
        int 16h

        cmp al, 78h     ;Compara que tecla se presiono, si es 'x'
        je DetectarAbajoJug2

        cmp al, 77h     ;Compara que tecla se presiono, si es 'w'
        je DetectarArribaJug2

        cmp al, 61h     ;Compara que tecla se presiono, si es 'a'
        je DeteccionIzquierdaJug2

        cmp al, 64h     ;Compara que tecla se presiono, si es 'd'
        je DeteccionDerechaJug2

        cmp al, 73h     ;Compara que tecla se presiono, si es 's'
        je DisparoJugador2


        jmp mueveJug2        ;Sino, no hace nada y vuelve a pedir una tecla

```

```
; CALCULADORA DE QUE MOVIMIENTOS SON POSIBLE PARA EL JUGADOR1
DeteccionDerechaJug1:
        mov ComparaPosicion, dl
        sub ComparaPosicion, 14 ; No puede ir a derecha de la columna 14
        JS MoverDerecha         ; si la diferencia es negativa, se puede mover a derecha
        jmp mueveJug1
        ret



DeteccionIzquierdaJug1:
        cmp dl, 0               ; No puede ir a izquierda si ya esta en columna 0
        je mueveJug1            ; si son iguales, volver al loop
        jmp MoverIzquierda      ; si no son iguales, permite el movimiento
        ret


DetectarArribaJug1:
        mov ComparaPosicion, dh    ; No puede ir hacia arriba si ya esta en fila 10 (0A)
        sub ComparaPosicion, 10    ; Si la resta es positiva permite subir
        JNS MoverArriba
        jmp mueveJug1
        ret


DetectarAbajoJug1:
        mov ComparaPosicion, dh    ; No puede ir hacia abajo si ya esta en fila 19 (0A)
        sub ComparaPosicion, 19    ; Si la resta es negativa permite bajar
        JS MoverAbajo
        jmp mueveJug1
        ret
```

```
; CALCULA QUE MOVIMIENTOS SON POSIBLE PARA EL JUGADOR2
DeteccionDerechaJug2:
        mov ComparaPosicion, dl
        sub ComparaPosicion, 31         ; No puede ir a derecha de la columna 31
        js  MoverDerecha                ; si la diferencia es negativa, se puede mover a derecha
        jmp mueveJug2
        ret



DeteccionIzquierdaJug2:
        mov ComparaPosicion, dl         ; No puede ir a izquierda si ya esta en columna 17
        sub ComparaPosicion, 17
        je  mueveJug2                    ; si son iguales, volver al loop
        jmp MoverIzquierda               ; si no son iguales, permite el movimiento
        ret


DetectarArribaJug2:
        mov ComparaPosicion, dh    ; No puede ir hacia arriba si ya esta en fila 10 (0A)
        sub ComparaPosicion, 10    ; Si la resta es positiva permite subir
        JNS MoverArriba
        jmp mueveJug2
        ret


DetectarAbajoJug2:
        mov ComparaPosicion, dh    ; No puede ir hacia abajo si ya esta en fila 19 (0A)
        sub ComparaPosicion, 19    ; Si la resta es negativa permite bajar
        JS MoverAbajo
        jmp mueveJug2
        ret
```

```
; DISPARO DEL JUGADOR 2 (ES UNA COMPARACION DE CARAC, QUE DETERMINA EL EXITO O EL FRACASO DEL DISPARO)
DisparoJugador1:



        mov ax, 0                   ; AX en cero (lo necesitamos para acumular o tener resultado de
                                    ; operaciones matematicas)

        mov al, 15                  ; En AL agregamos la cantidad de unidades encontradas en una sola fila
                                    ; de la matriz de mar individual

        sub dh,9                    ; como vamos a comparar la posicion del cursor en el momento actual con
                                    ; respecto a la posicion absoluta en la matriz, quitamos las 9 posiciones
                                    ; extras que tenemos por culpa del encabezado (el arte + el titulo)

        mul dh                      ; multiplicamos la cantidad de filas que nos desplazamos hacia abajo.
                                    ; Estamos contando filas usando la posicion. La fila cero es en realidad
                                    ; la fila actual donde estamos parados porque esta incompleta.
                                    ;
                                    ; De esta forma en AX ahora tenemos la cantidad de elementos de la matriz
                                    ; que estamos corridos con respecto a la fila actual.

        add dh,9                    ; Devolvemos el registro del cursor a su posicion original (sumando los 9
                                    ; requeridos por culpa del encabezado (el arte + el titulo).


        mov bl, dl                  ; Movemos el valor de la posicion de la columna actual del cursor hacia BL
        mov bh,0                    ; Ponemos BH en cero.
                                    ; Hicimos todo lo anterior porque necesitamos sumarle el valor de la posicion
                                    ; actual del cursor al registro AX. Pero BL, DL son registros de 8bits y AX
                                    ; es de 16 bits. Con las dos instruccciones anteriores, convertirmos la pos
                                    ; del cursor en un registro de 16bits


        add ax, bx                  ; Realizamos la suma y ahora si tenemos la cantidad de elementos de la matriz
                                    ; que estamos corridos con respecto a la fila y columna actual.

        mov bx,0                    ; No necesitamos mas a BX, asi que lo ponemos en cero. Nuestro valor esta en AX


        MOV di,ax                   ; Nuestro valor en AX lo pasamos al registro DI. Porque necesitamos realizar un
                                    ; direccionamiento indirecto


        mov al, botes_Jug2[di]      ; Realizamos nuestro direccionamiento indirecto y obtenemos el caracter que se
                                    ; en la matriz de mar individual.

        cmp al, exitoso
        je SigueMoviendo
```

```
401
402         cmp al, fracasoso
403         je SigueMoviendo
404
405         cmp al , barquito             ; Se compara el caracter obtenido con el que corresponde a un segmento de barco.
406         je  acertar                   ; Si coinciden salta a la subrutina acertar
407         jmp errar                     ; De lo contrario se dirige a la subrutina errar.
408
409
410
411 ; DISPARO DEL JUGADOR 1 (ES UNA COMPARACION DE CARAC, QUE DETERMINA EL EXITO O EL FRACASO DEL DISPARO)
412 DisparoJugador2:
413
414
415
416
417         mov ax, 0
418
419         mov al, 15
420         sub dh,9
421         mul dh
422         add dh,9
423
424
425         mov bl, dl
426         sub bl, 17
427         mov bh,0
428         add ax, bx
429
430         mov bx,0
431
432
433         MOV di,ax
434         mov al, botes_Jug1[di]
435
436         cmp al, exitoso
437         je SigueMoviendo
438
439         cmp al, fracasoso
440         je SigueMoviendo
441
442         cmp al , barquito
443         je  acertar
444         jmp errar
445
446
447
448
449
450
```

```
; APARTADO GRAFICO DEL DISPARO (CAMBIA EL CARAC EN PANTALLA Y SOLICITA EL FINAL DEL TURNO)
acertar:                ; pintar de amarillo el impacto en el marP
mov al, exitoso
mov bl, 1100b
call CambiarValorMatriz
call pintarDisparo
call AgregarPuntaje
jmp FinDeTurno


errar:                  ; pintar de rojo el fallido en el mar
mov al, fracasoso
mov bl, 0111b
call CambiarValorMatriz
call pintarDisparo
jmp FinDeTurno


pintarDisparo PROC near ; pintar el disparo en el mar
mov ah, 09h
mov bh, 0
mov cx, 1
int 10h
ret
pintarDisparo endp
```

```
; subrutinas  (o loops) que se encargan del absoluto movimiento del cursor

MoverDerecha:
        add dl, 1
        call SetCursor
        jmp SigueMoviendo
        ret


MoverIzquierda:
        sub dl, 1
        call SetCursor
        jmp SigueMoviendo
        ret

MoverArriba:
        sub dh, 1           ;para reposicionar el cursor fila
        call SetCursor      ;llamo al procedimiento para setear cursor
        jmp SigueMoviendo
        ret

MoverAbajo:
        add dh, 1           ;para reposicionar el cursor fila
        call SetCursor      ;llamo al procedimiento para setear cursor
        jmp SigueMoviendo
        ret

SetCursor proc
        mov ah, 02h
        mov bh, 00
        int 10h             ;Analizar esta int, que es la que posiciona el cursor
        ret

SetCursor endp
```

```
551  ; Subrutinas de turnos de jugadores
552
553  SigueMoviendo:          ; En el caso de que el movimiento finalizo
554      cmp MueveJugador,0  ; o es invalido pero continua el turno en el jugador actual
555      je  mueveJug1
556      jmp mueveJug2
557      ret
558
559
560  FinDeTurno:
561      call CalcularPuntaje
562
563      cmp Jugando,0
564      je terminarJuego
565
566      cmp MueveJugador,0     ; En el caso de que el movimiento finalizo
567      je  AhoraMueveJugador2 ; y con ello tambien el turno del jugador actual
568      jmp AhoraMueveJugador1
569
570      terminarJuego:
571      ret
572
573
574  AhoraMueveJugador1:         ;Realiza las acciones para el comienzo del turno del jugador 1
575      mov MueveJugador,0
576      mov dh, 14
577      mov dl, 7
578      call SetCursor
579      jmp mueveJug1
580      ret
581
582
583  AhoraMueveJugador2:         ;Realiza las acciones para el comienzo del turno del jugador 2
584      mov MueveJugador,1
585      mov dh, 14
586      mov dl, 24
587      call SetCursor
588      jmp mueveJug2
589      ret
590
591
592
593
594
595
596
597
598
599
600
```

```
AgregarPuntaje proc
    cmp MueveJugador,0
    je  sumarJug1
    jmp sumarJug2

    sumarJug1:
    mov ah,offset[PuntajeJugador1+0]
    mov al,offset[PuntajeJugador1+1]
    sub ah,30h
    sub al,30h

    mov dh,ah
    mov ah,0
    mov dl,al

    mov ax,10
    mul dh
    mov dh,0
    add ax,dx

    cmp ax,9
    je pasarAdiez1

    cmp ax,19
    je pasarAveinte1

    inc ax

    mov ah,offset[PuntajeJugador1+0]
        cmp al,10
    ja restarDiez1
    jmp continuarNoCambiaDecena1

    restarDIez1:
    sub al,10

    continuarNoCambiaDecena1:
    add al,30h
    jmp escribirPuntosEnMem1

    pasarAdiez1:
    mov ah,31h
    mov al,30h
    jmp escribirPuntosEnMem1

    pasarAveinte1:
    call MensajeVictoria
    mov ah,32h
    mov al,30h
    jmp escribirPuntosEnMem1
```

```
651
652   escribirPuntosEnMem1:
653       mov offset[PuntajeJugador1+0],ah
654       mov offset[PuntajeJugador1+1],al
655       jmp finalAgregarPuntaje
656
657
658
659       sumarJug2:
660       mov ah,offset[PuntajeJugador2+0]
661       mov al,offset[PuntajeJugador2+1]
662       sub ah,30h
663       sub al,30h
664
665       mov dh,ah
666       mov ah,0
667       mov dl,al
668
669       mov ax,10
670       mul dh
671       mov dh,0
672       add ax,dx
673
674       cmp ax,9
675       je pasarAdiez2
676
677       cmp ax,19
678       je pasarAveinte2
679
680       inc ax
681
682       mov ah,offset[PuntajeJugador2+0]
683
684       cmp al,10
685       ja restarDiez2
686       jmp continuarNoCambiaDecena2
687
688       restarDIez2:
689       sub al,10
690
691       continuarNoCambiaDecena2:
692       add al,30h
693       jmp escribirPuntosEnMem2
694
695       pasarAdiez2:
696       mov ah,31h
697       mov al,30h
698       jmp escribirPuntosEnMem2
699
700       pasarAveinte2:
```

```
701         call MensajeVictoria
702         mov ah,32h
703         mov al,30h
704         jmp escribirPuntosEnMem2
705
706
707
708         escribirPuntosEnMem2:
709         mov offset[PuntajeJugador2+0],ah
710         mov offset[PuntajeJugador2+1],al
711         jmp finalAgregarPuntaje
712
713
714
715
716         finalAgregarPuntaje:
717         ret
718
719     AgregarPuntaje endp
720
721
722
723     CalcularPuntaje proc
724
725     ; Mostrar en el puntaje en el tablero
726     mov dh, 20
727     mov dl, 8
728     call setCursor
729     mov dx, offset[PuntajeJugador1]
730     mov bh, 0
731     mov bl, 03h
732     mov ah,9
733     int 21h
734
735
736     ; Mostrar en el puntaje en el tablero
737     mov dh, 20
738     mov dl, 25
739     call setCursor
740     mov dx, offset[PuntajeJugador2]
741     mov bh, 0
742     mov bl, 03h
743     mov ah,9
744     int 21h
745
746     ret
747     CalcularPuntaje endp
748
749
750
```

```
751   ; APARTADO PARA CAMBIAR LOS VALORES DE MEMORIA DE LAS MATRICES DE MAR ORIGINALES
752   CambiarValorMatriz proc
753           cmp MueveJugador,0
754           je  cambiaValorJug2
755
756           mov botes_Jug1[di],al
757           jmp FinCambiarValorMatriz
758
759
760           cambiaValorJug2:
761           mov botes_Jug2[di],al
762           jmp FinCambiarValorMatriz
763
764           FinCambiarValorMatriz:
765           ret
766
767   CambiarValorMatriz endp
768
769   cmp Jugando,1
770   jne finalMensajeVictoria
771
772
773   MensajeVictoria proc
774   mov di,dx
775   mov dh, 22
776   mov dl, 0
777   call setCursor
778   mov dx, offset[MensajeVictoriaCadena]
779   mov bh, 0
780   mov bl, 03h
781   mov ah,9
782   int 21h
783
784   inc MueveJugador
785   add MueveJugador,30h
786   mov ah, 09h
787   mov al, MueveJugador
788   mov bh, 0
789   mov cx, 1
790   int 10h
791   mov dx,di
792   call setCursor
793   sub Jugando,1
794
795   finalMensajeVictoria:
796   ret
797
798
799   MensajeVictoria endp
```