



FACULTAD DE INGENIERÍA

ESTRUCTURA Y PROGRAMACIÓN DE COMPUTADORAS

GRUPO 2

Proyecto N°2

Martínez Baeza José Alfonso
Santana sanchez María Yvette

03 de junio de 2020

Índice

1. Introducción	2
1.1. Objetivo	2
1.2. Descripción del problema	2
1.3. Planteamiento del problema	2
2. Desarrollo	2
2.1. Interrupciones	3
2.1.1. int 21h	3
2.1.2. int 33h	3
2.1.3. int 10h	4
2.2. Código	4
2.2.1. Interfaz gráfica	6
2.2.2. Funcionamiento lógico	10
3. Diagramas de flujo	12
4. Pruebas de escritorio	14
5. Conclusiones	14

1. Introducción

1.1. Objetivo

Se requiere desarrollar un programa en lenguaje ensamblador para arquitectura Intel x86 que muestre una calculadora de manera gráfica en pantalla.

1.2. Descripción del problema

Para programar una calculadora con gráficos en lenguaje ensamblador es necesario utilizar algunas interrupciones para el modo en el que visualizamos los programas en Dosbox, también se hará uso de algunas macros y procedimientos del proyecto anterior para acelerar el desarrollo de esta calculadora, la dificultad de este proyecto radica en la división de la pantalla para saber en donde se ubica el cursor y donde efectúa el cliqueo, esto implica tener claro cuanto miden los botones utilizados y los espacios que tenemos dentro de la pantalla.

Utilizaremos un código base distinto al proporcionado por el profesor, el código fuente base puede encontrarse en el siguiente repositorio Github:

<https://github.com/Raufoon/Simple-Calculator-Assembly-Language-.git>

1.3. Planteamiento del problema

Desarrollar un programa en lenguaje ensamblador para arquitectura Intel x86 que muestre una calculadora de manera gráfica en pantalla.

- Números del 0 al 9
- Botón suma (+)
- Botón resta (-)
- Botón multiplicación (*)
- Botón cociente (/)
- Botón residuo (%)
- Botón resultado (=)
- Botón AC
- Botón C

Consideraciones:

- Los números deben estar en sistema decimal (dígitos de 0-9).
- Los números deberán ser enteros sin signo.
- Cada número introducido por el usuario puede ser de hasta 4 dígitos. El programa debe restringir que el usuario introduzca más números.

2. Desarrollo

Para la realización de este proyecto nos apoyaremos del uso de macros y procedimientos dentro del lenguaje ensamblador para poder reducir el número de líneas de código, explicaremos a grandes rasgos el funcionamiento de dos interrupciones.

2.1. Interrupciones

Las interrupciones son un método del que disponen los dispositivos e incluso los procesos para hacer notar a la CPU la aparición de alguna circunstancia que requiera su intervención, esto con el fin de que los dispositivos puedan provocar que la CPU deje por el momento o la tarea que estaba realizando y haga la interrupción que se requiere.

Las interrupciones que veremos son tres, aunque la gama de interrupciones es mucho más amplia.

2.1.1. int 21h

La interrupción 21h es una de las mas usadas e importantes para trabajar con el sistema operativo ya que llama o invoca todos los servicios de llamada función DOS, este compuesto por un grupo amplio de funciones, por ello solo mencionaremos la más usadas.

- 00h:** Terminación de Programa.
- 01h:** Entrada de caracteres con eco.
- 02h:** Salida de caracteres.
- 05h:** Salida de impresora.
- 06h:** E/S directa de consola.
- 08h:** Entrada de caracteres sin eco.
- 09h:** Salida de una cadena de caracteres.

No todas las funciones mencionadas se utilizarán en este proyecto, aunque es importante señalar que las funciones como 08h y 01h, es decir, lectura sin eco y con eco respectivamente, se utilizaron en el proyecto anterior.

2.1.2. int 33h

Esta es posiblemente una de las interrupciones que mas usaremos dentro de este proyecto, pues esta la que implica el manejo del ratón, dentro de esta misma interrupción tenemos las siguientes funciones de mayor uso.

- 00h:** Inicializa el ratón.
- 01h:** Muestra el apuntador del ratón.
- 02h:** Oculta el apuntador del ratón.
- 03h:** Obtiene el estado del botón y la posición del ratón.
- 04h:** Establece la posición del apuntador.
- 05h:** Obtiene información del botón presionado del ratón.
- 06h:** Obtiene la información acerca de la liberación del botón
- 07h:** Fija limites horizontales para el apuntador.
- 08h:** Fija limites verticales para el apuntador.
- 09h:** Establece el tipo de apuntador gráfico.
- 0Ah:** Establece el tipo de apuntador en texto.

Las funciones señaladas son las que deberemos tener en cuenta para el siguiente trabajo, como curiosidad esta interrupción trabaja directamente con el registro AX y no el Ah que se usan en las operaciones.

2.1.3. int 10h

Esta interrupción selecciona y activa el modo de video.

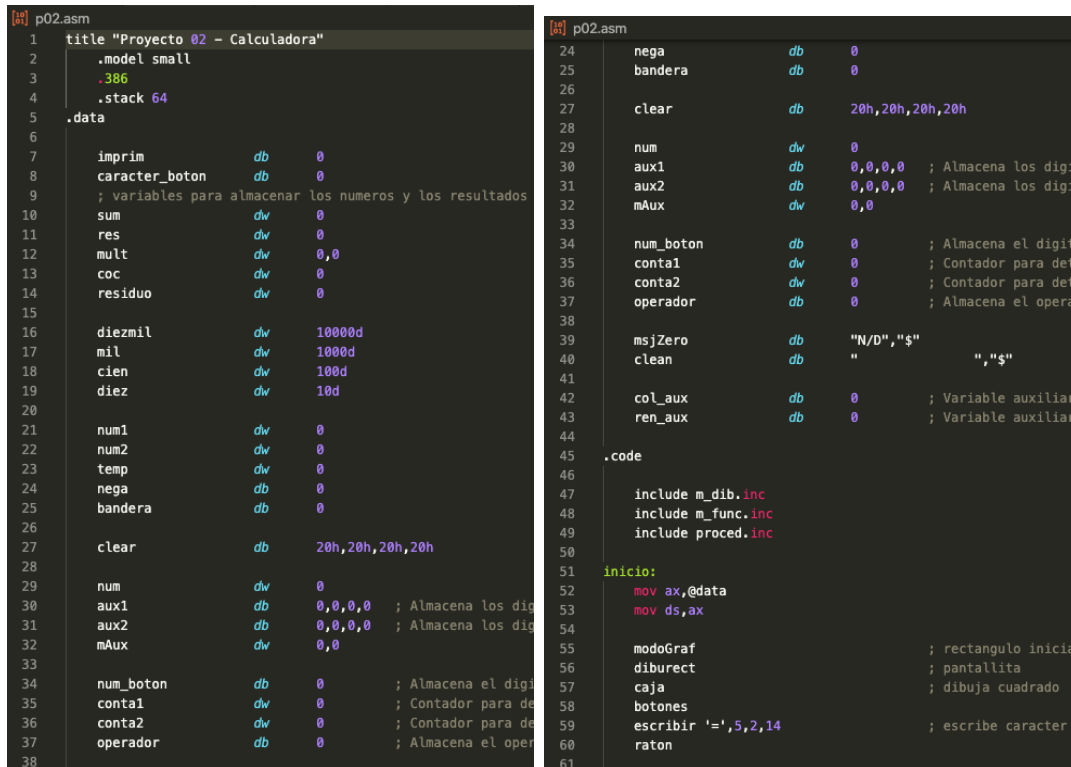
- 00h:** Asignar modo de video
- 01h:** Asignar tipo de cursor
- 02h:** Situar posición del cursor
- 03h:** Leer posición del cursor
- 08h:** Obtener atributo y carácter en el cursor
- 09h:** Escribir atributo y carácter en el cursor
- 0Ah:** Escribir únicamente carácter en el cursor
- 0Bh:** Asignar paleta de colores
- 0Ch:** Mostrar pixel grafico

En este caso se trabaja con el registro AH, como vemos ya tenemos señalado las funciones que se mostraran mas adelante en el proyecto.

2.2. Código



Como podemos notar en la imagen anterior tenemos el proyecto dividido en tres archivos: p02, m_dib, m_func y proced, esto con el fin de separar el código para una cómoda lectura no solo de un tercero sino de nosotros mismos para descubrir posibles errores de manera más rápida.



Tenemos las directivas conocidas en el proyecto anterior, en .data se ubican las variables que vamos a utilizar a lo largo del código, aux1 y aux2 será la variable que guarde los 4 dígitos

que se van a introducir por medio del ratón, en .code podemos ver como incluimos dentro del archivo p02.asm los otros componentes (o podemos verlos como bibliotecas locales semejante a C), donde m_dib y m_func contienen las respectivas macros utilizadas dentro del código para hacer el tema grafico mientras que proced contiene procedimientos que acumulan los cuatro dígitos en una variable para luego imprimirla (adaptación del proyecto anterior).

```
inicio:
    mov ax,@data
    mov ds,ax

    modoGraf                ; rectangulo inicial
    dibirect                ; pantallita
    caja                    ; dibuja cuadrado
    botones
    escribir '=',5,2,14     ; escribe caracter en pantalla
    raton
```

Dentro de la etiqueta inicio, que pertenece a la directiva .code, podemos ver como comenzamos el programa, con su respectiva reserva de segmento, posteriormente se hace una llamada a las macros en este caso modoGraf, dibirect, caja, botones y ratón.

A continuación, analizaremos dichas macros (iremos por ello al apartado m_dib.inc) al bloque de código perteneciente a modoGraf.

```
modoGraf macro
    mov ah,0
    mov al,13H
    int 10H
endm
```

Imaginemos que solo tenemos el código anterior ejecutado en el inicio, entonces al ejecutarlo en DOSBox encontraremos los siguiente:

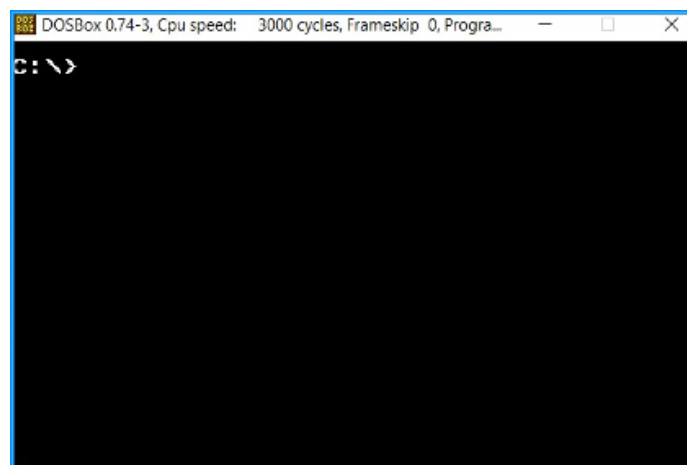


Figura 1: Entra en modo gráfico

Con este apartado de código iniciamos el modo grafico con la interrupción 13h.

En la siguiente imagen veremos las macros de diburect, caja.

2.2.1. Interfaz gráfica

```
; Dibujamos el rectangulo de afuera
diburect macro
    limpiar
    lineahor 4,254,4,1           ; Primero alarga la linea horizontal (bajo) corto (alto)
    limpiar
    lineahor 4,255,194,1
    limpiar
    lineaver 4,194,4,1
    limpiar
    lineaver 4,194,254,1
    limpiar
endm

; Dibuja caja
caja macro
    limpiar
    lineahor 9,249,9,7           ; El ultimo valor es el color
    limpiar
    lineahor 9,250,69,7          ; El primero es hacia arriba
    limpiar
    lineaver 9,69,9,7             ; El segundo hacia abajo
    limpiar
    lineaver 9,69,249,7           ; La tercera linea divisoria de altura
    limpiar
endm
```

En diburect, primero limpia la pantalla, posteriormente genera dos líneas horizontales y otras dos líneas verticales.

Nuevamente para ejemplificar mejor el desarrollo de este programa, mostraremos paso a paso como se desarrolla la transformación gráfica.

En la macro tenemos la siguiente línea de código:

- lineahor 4,254,4,1

Parámetro 1: La extensión de la línea horizontal hacia la izquierda.

Parámetro 2: La extensión de la línea horizontal hacia la derecha.

Parámetro 3: La posición de la línea horizontal más arriba o abajo.

Parámetro 4: Determina el color.

- lineaver 4,254,4,1

Parámetro 1: La extensión de la línea vertical hacia la izquierda.

Parámetro 2: La extensión de la línea vertical hacia la derecha.

Parámetro 3: La posición de la línea vertical más arriba o abajo.

Parámetro 4: Determina el color.

Estos códigos llaman a su vez a otra macro que determina el dibujo del rectángulo.

```

lineahor macro X,Y,Z,C
    LOCAL L1

    mov cx,X
    mov dx,Z

L1:
    mov ah,0CH
    mov al,C
    int 10H
    inc CX
    cmp CX,Y
    jnz L1
endm

```

```

lineaver macro X,Y,Z,C
    LOCAL L2

    mov cx,Z
    mov dx,X

L2:
    mov ah,0CH
    mov al,C
    int 10H
    inc DX
    cmp DX,Y
    jnz L2
endm

```



Figura 2: Dibuja marco de la calculadora

Al agregar la macro caja al código dibujaremos nuevamente otro rectángulo valiéndonos de las macros utilizadas anteriormente líneahor y lineaver.

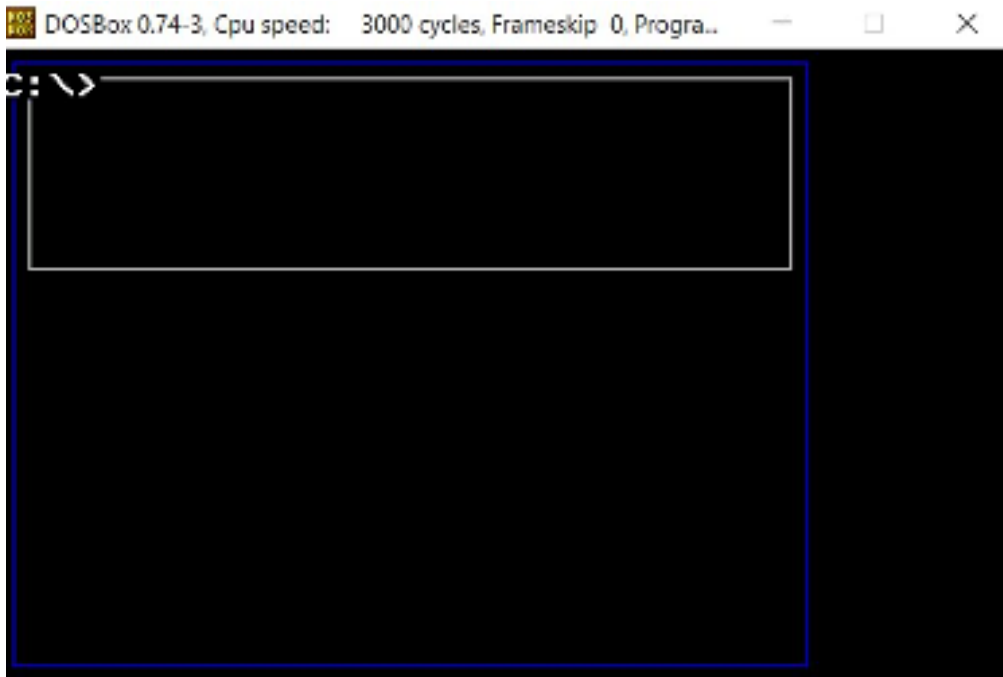


Figura 3: Dibuja caja donde imprimirá las operaciones

Un detalle importante al dibujar los rectángulos que deseamos debemos siempre tener en cuenta que ese código debe ir después de ejecutar el modo gráfico, sino el dibujo no se realizará.

```

botones macro
; botones izquierdos
dibujarboton 74,9,15
dibujarboton 104,9,15
dibujarboton 134,9,15
dibujarboton 164,9,40

; botones de en medio
dibujarboton 74,59,15
dibujarboton 104,59,15
dibujarboton 134,59,15
dibujarboton 164,59,15
dibujarboton 164,109,40

dibujarboton 74,109,15
dibujarboton 104,109,15
dibujarboton 134,109,15

dibujarboton 74,159,100
dibujarboton 104,159,100
dibujarboton 134,159,100
dibujarboton 164,159,100

; botones derechos
dibujarboton 74,209,100
botontipo 104,209,10

    escribir '0',22,9,15
    escribir '1',18,3,15
    escribir '2',18,9,31
    escribir '3',18,16,15

    escribir '4',14,3,31
    escribir '5',14,9,31

    escribir '6',14,16,31

    escribir '7',10,3,31
    escribir '8',10,9,31
    escribir '9',10,16,31

    escribir 'x',10,22,100
    escribir '/',14,22,100
    escribir '-',18,22,100
    escribir '+',22,22,100

    escribir '%',10,28,100

    escribir '=',18,28,10

    escribir 'A',22,3,40
    escribir 'C',22,4,40

    escribir 'D',22,15,40
    escribir 'E',22,16,40
    escribir 'L',22,17,40

    escribir 'X',0,30,40

endm
  
```

En esta línea de código se invocan dos macros:

- escribir '0',22,9,15

Parámetro 1: Se envía el carácter que deseamos imprimir dentro del modo gráfico.

Parámetro 2: Posiciona el carácter hacia arriba o abajo.

Parámetro 3: Posiciona el carácter hacia la derecha o izquierda

Parámetro 4: Determina el color del caracter.

- dibujarboton 74,9,15

Parámetro 1: Posiciona el botón hacia arriba o abajo.

Parámetro 2: Posiciona el botón hacia la derecha o hacia la izquierda.

Parámetro 3: Determina el color del botón.

La imagen siguiente muestra como se ve lo anteriormente mencionado para una mejor visualización de lo explicado.

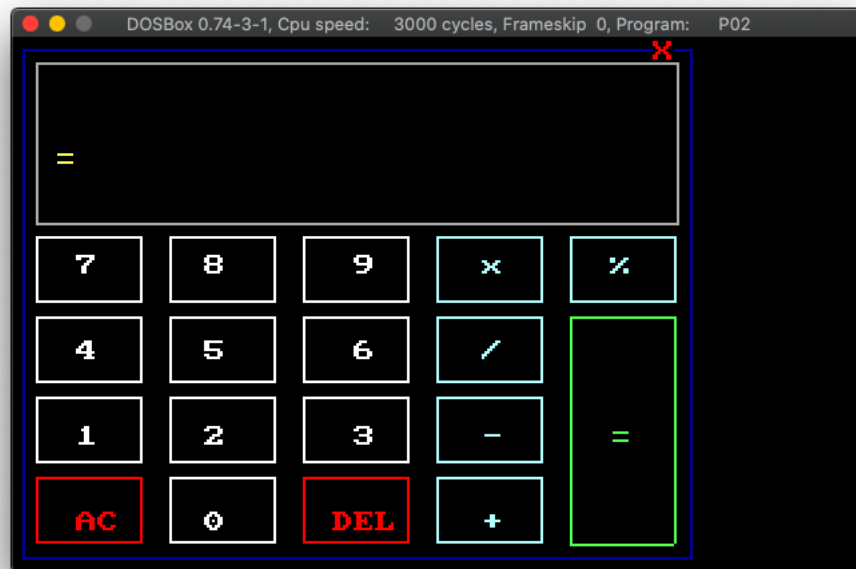


Figura 4: Dibuja botones

En la macro de botones debemos analizar la macro dibujarboton, botontipo y escribir, ya que esta dibuja las letras dentro de los botones.

```

botontipo macro X, Y, C
    limpiar
    lineahor Y,(Y+39),X,C
    limpiar
    lineahor Y,(Y+39),(X+35+25+25),C
    limpiar
    lineaver X,(X+35+25+25),Y,C
    limpiar
    lineaver X,(X+35+25+25),(Y+39),C
    limpiar
endm

dibujarboton macro X, Y, C
    limpiar
    lineahor Y,(Y+39),X,C
    limpiar
    lineahor Y,(Y+39),(X+24),C
    limpiar
    lineaver X,(X+24),Y,C
    limpiar
    lineaver X,(X+25),(Y+39),C
    limpiar
endm

escribir macro CHAR,R,C,COL
    mov ah,0Bh
    mov bh,0h
    mov bl,3
    int 10h
    mov ah,2
    mov bh,0
    mov dh,R
    mov dl,C
    int 10h
    mov ah,9
    mov al,CHAR
    mov bl,COL
    mov cx,1
    int 10h
endm

```

Estas macros anteriores son las encargadas de dibujar el botón y los caracteres que los representa para el usuario.

Otra de las macros importante es la ratón es la encargada de inicializar el mouse y mostrarlo.

```

raton macro
    mov ax,00h
    int 33h

    mov ax,01h
    int 33h
endm

```

2.2.2. Funcionamiento lógico

Una vez que se tiene la interfaz gráfica dibujada es necesario programar el código necesario para hacer funcionar cada uno de los botones. En este caso, a diferencia de otros lenguajes de alto nivel, es necesario ir delimitando la zona de cada botón en donde permitirá dar clic.

Al iniciar se verifica si se presiona el boton izquierdo del mouse, una vez que se comprueba esto se compara con el número de columna, si es menor o igual a 48 es probable que se haya presionado sobre los botones más a la izquierda (7,4,1 o AC); si es menor o igual a 98 es probable que se haya presionado sobre los números del centro (8,5,2 o 0); si es menor o igual a 148 es probable que se haya presionado sobre los números más a la derecha (9,6,3 o DEL); si es menor o igual a 198 es probable que se haya presionado sobre los botones de operaciones (X,/,− o +) y si es menor o igual a 248 es probable que se haya presionado sobre los botones más a la derecha (% o =)

```

mouse:
revisaClicIzquierdo
; Al final de la macro anterior, obtenemos las coordenadas del mouse
; DX es el numero del renglon, que puede ser desde 0 hasta 200
; CX es el numero de la columna, que puede ser desde 0 hasta 320

; Revisar si el mouse se presiono en X para cerrar
; ;En res 320x200, X inicia en (240,0) y termina en (247,8)
cmp dx,8
jle botonX

; Rango de columnas en donde esta la primera columna de botones (botones izquierdos)
; minimo 9, ya se hizo la comparacion por ese valor anteriormente
; maximo 48
cmp cx,48
jle botones_AC_1_4_7

cmp cx,98
jle botones_0_2_5_8

cmp cx,148
jle botones_C_3_6_9

cmp cx,198
jle botones_operaciones

cmp cx,248
jle botones_operaciones2

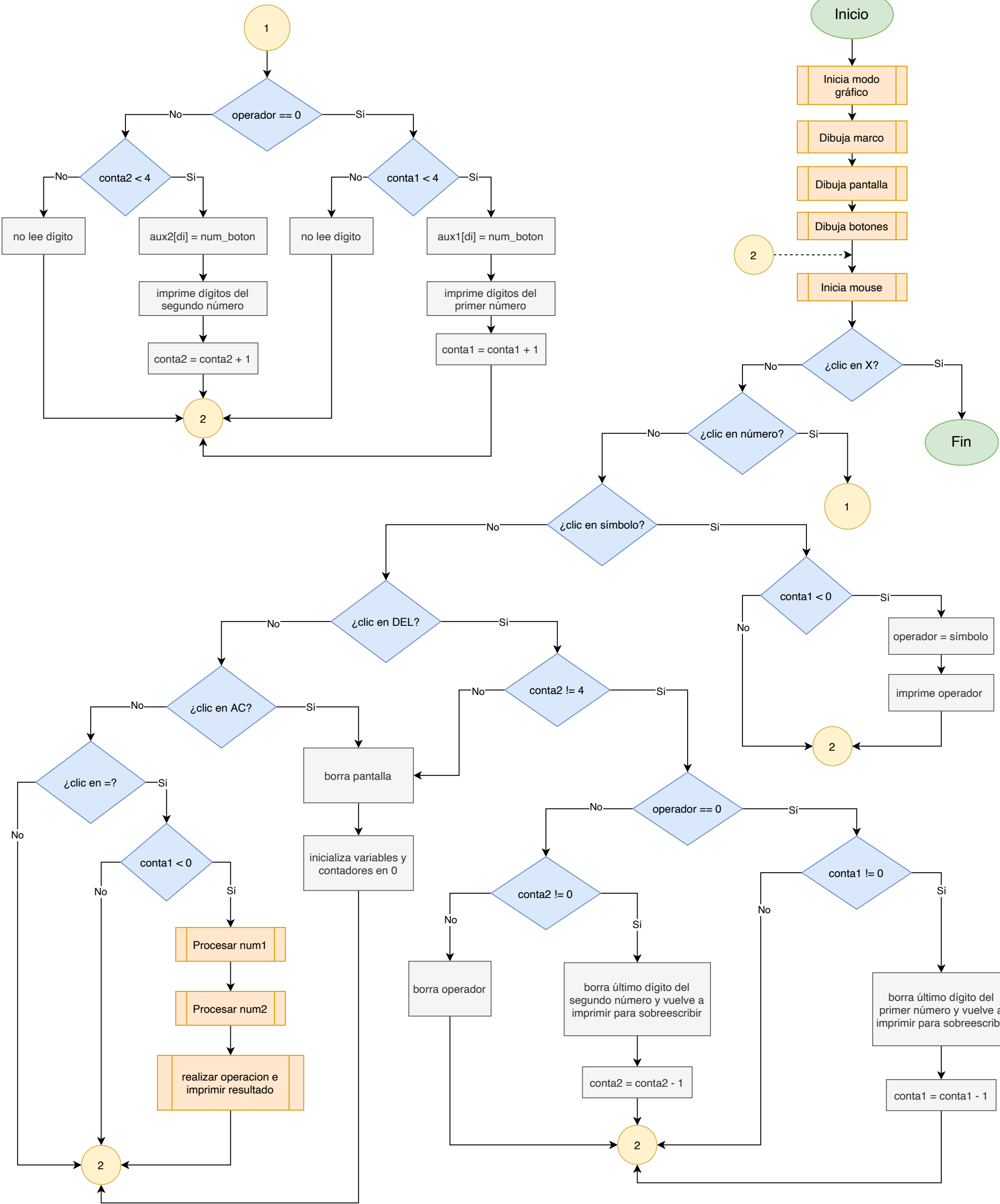
jmp mouse

```

El código base proporcionado por el profesor sugería una triple comprobación para verificar que estuviera dentro del rango de botones, dentro de ciertas columnas y dentro de ciertos renglones, parecía una solución viable, sin embargo al avanzar con el desarrollo del proyecto nos dimos cuenta de que el código comenzaba a extenderse demasiado, por lo que decidimos reducir estas comprobaciones con una macro llamada *isButton* la cual recibe 4 parámetros: *xmin*, *xmax*, *ymin* y *ymax*, estos hacen referencia a las cuatro esquinas que conforman un boton y a su vez su funcionamiento puede extenderse para verificar una zona completa.

```
isButton macro xmin,xmax,ymin,ymax
; Comparar que se haga clic dentro del boton
cmp dx,ymin
jl mouse ; Si DX < ymin, regresa a etiqueta 'mouse'
cmp dx,ymax
ja mouse ; Si DX > ymax, regresa a etiqueta 'mouse'
cmp cx,xmin
jl mouse ; Si CX < xmin, regresa a etiqueta 'mouse'
cmp cx,xmax
ja mouse ; Si CX > xmax, regresa a etiqueta 'mouse'
endm
```

3. Diagramas de flujo



4. Pruebas de escritorio

5. Conclusiones