Descripción del proyecto: “Desarrollo de un juego de TIC TAC TOE (GATO) en Lenguaje ensamblador”

Estudiantes:

- Sebastián Orozco Castillo C35719
- Melanny Hernández Rivera B83808

I. Descripción de la propuesta.

El proyecto consiste en programar el juego TIC TAC TOE, mejor conocido como “GATO”, utilizando instrucciones y código construido con lenguaje ensamblador. El IDE para realizar construir el código fue el programa MARS. En él, se configuraron además, algunas herramientas con el objetivo de proporcionar al usuario una mejor experiencia a la hora de ejecutar el código, las cuales se mostrarán más adelante.

a) Funcionalidades implementadas dentro del código.

- **Ciclo:** función para iniciar el juego, esta función "es la principal" para recibir el teclado y dibujar demás cosas del juego.
- **leerTeclado:** Espera un input del usuario para leer el teclado, recibe w a s d(funciones que dibujan una X o una O) y enter para poder jugar.
- **ponerFigura:** simplemente decide si se leyó una X o una O enviadas por leerTeclado y llama a la función de dibujar para colocar una X o una O en la posición enviada. Además de usar una función que se llama obtenerColor que esta lo que hace es controlar si se le envía un O o una X, y dependiendo de lo que sea decide qué color se va a dibujar, las X son Rojas y los O son azules.
- **dibujarTablero:** función para dibujar el tablero del juego, se complementa con las demás funciones de "dibujar" explicadas posteriormente.
- **dibujarVerticalIzquierda:** función que dibuja una línea vertical a la izquierda de la pantalla (primera línea vertical del tablero), y llama a la función
- **dibujarVerticalCentral:** dibuja la línea vertical central del tablero.
- **dibujarVerticalDerecha:** dibuja una línea vertical a la derecha de la pantalla (última línea vertical del tablero), y llama a la función dibujarPrimeraLinea.
- **dibujarPrimeraLinea:** reinicia los valores.
- **dibujarHorizontalSuperior:** función que dibuja la línea horizontal de arriba del tablero, y llama a la función dibujarHorizontalCentral.
- **dibujarHorizontalCentral:** función que dibuja la línea horizontal central del tablero.
- **dibujarHorizontalInferior:** función que dibuja la línea horizontal inferior del tablero, y llama a la función posicionarSelector.
- **posicionarSelector: f**unción que dibuja un pixel en el centro del tablero, el cual se mueve mediante las teclas a,s,d,e para elegir la posición del tablero en la que se quiere poner la figura correspondiente.
- **dibujarX:** función que dibuja píxel a píxel, una X dependiendo de donde se seleccione, usa las funciones dibujarPixel y posicionarSelector.
- **dibujarO:** función que dibuja píxel a píxel, un O dependiendo de donde se seleccione, usa las funciones dibujarPixel y posicionarSelector. Esta fue un poco

más compleja, no se hace un círculo perfecto ya que solo se pueden manejar pixeles en diagonal, entonces se intenta representar.
- **guardarPosicion:** Función que guarda donde se colocan los elementos en el tablero, se guarda el color y la ubicación de la figura (ROJO = x & AZUL = O).
- **comprobarGanador:** función que dibuja píxel a píxel una línea amarilla sobre las tres casillas ganadoras, usa las funciones dibujarVertical, dibujarHorizontal, dibujarDiagonalIzquierda y dibujarDiagonalDerecha.
- **dibujarPixel:** función encargada de dibujar un pixel, esta función es la más importante.
- **comprobarEmpate:** simplemente comprueba que no se jueguen más de 9 veces.

b) Funciones no implementadas

- **reiniciarTablero:** Se reiniciaría el tablero luego de una victoria, para volver a jugar.
- **verificarEspacioOcupado:** Permitiría que no se sobreescribiera en el espacio ya seleccionado.

Como se mencionó, estas dos funciones no se implementaron, pues no afectan al funcionamiento del programa. En resumen, no son necesarias, ya que no producen un error como tal. El programa cumple con los requerimientos mínimos para jugar una partida, por lo que el usuario puede compilar e iniciar el programa cada vez que desee jugar.

c) Errores conocidos dentro del programa:

1. Si no se inicia la herramienta bitmap y la de leer el teclado, el programa no hará nada (no es un error como tal, simplemente el programa está hecho solo para que funcione con bitmap).
2. Se puede sobreescribir en el tablero, es decir, se puede escribir una X encima de una O, sin embargo no afecta al funcionamiento.

II. Diagrama de flujo del programa principal

Debido al tamaño, se adjuntará el diagrama de flujo en las páginas siguientes.

III. Diagrama de flujo de las funciones utilizadas

Debido al tamaño, se adjuntarán los diagrama de flujo en las páginas siguientes.

IV. Resultados

Para evidenciar los resultados del proyecto elaborado, se mostrarán tres imágenes a continuación:

Imagen 1. Tablero

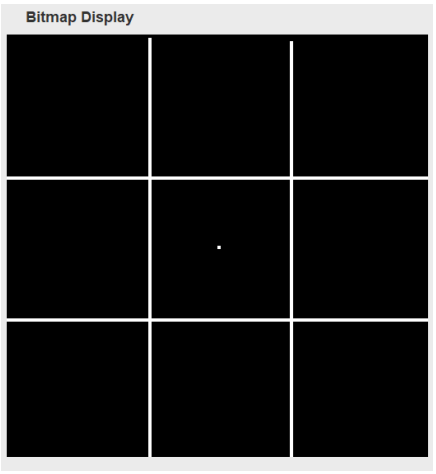


Imagen 2. Algunas casillas ocupadas

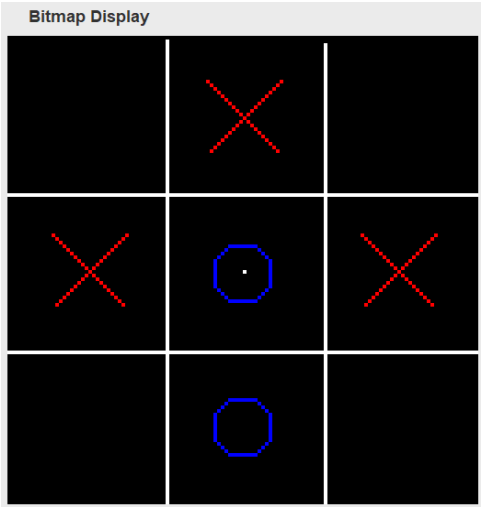
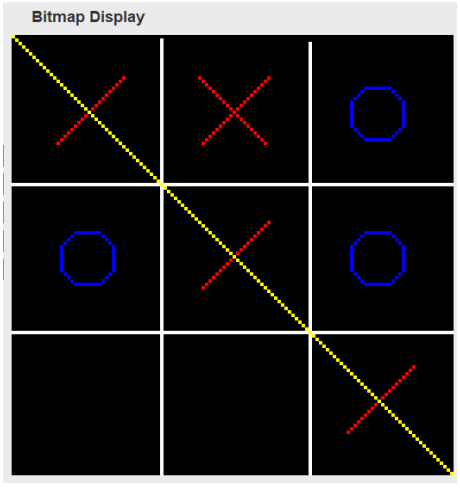


Imagen 3. Alguna partida ganada para ver las líneas verificadoras.

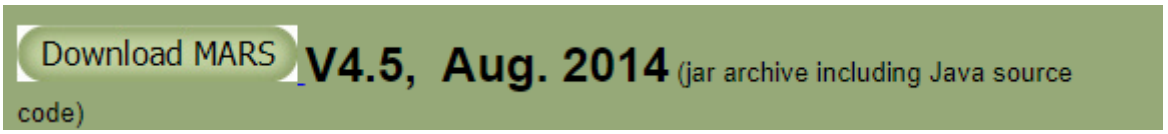


2. El código del programa en formato .asm con su respectiva documentación

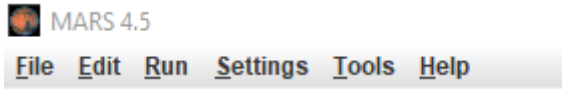
Se adjunta a la entrega, el documento juegoGato.asm, con su respectiva documentación.

3. Instrucciones detalladas para ejecutar el programa

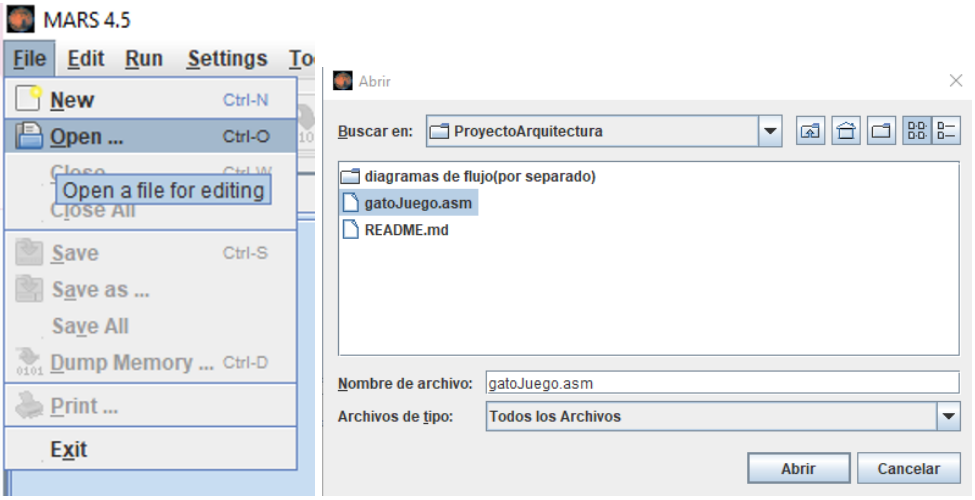
1. Instale el programa MARS. Puede hacerlo desde el siguiente enlace: https://courses.missouristate.edu/KenVollmar/MARS/ . Diríjase al botón “download” y siga las instrucciones.



2. Luego de instalar el programa, diríjase hacia el lado superior izquierdo, específicamente en el apartado de file.

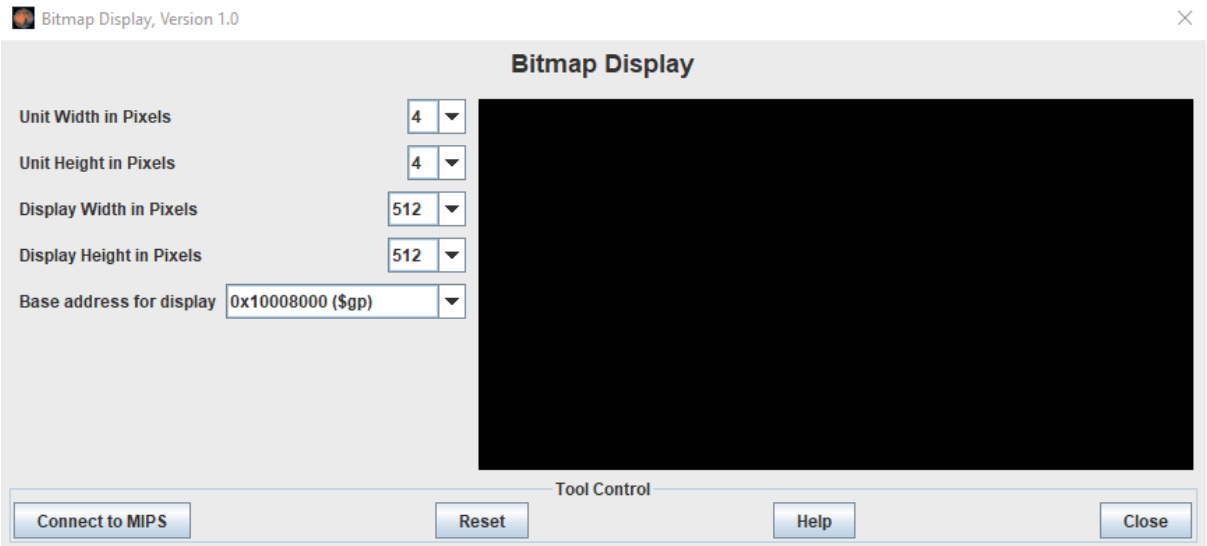


3. Seleccione la opción “open” y luego, busque el archivo juegoGato.asm de este repositorio para seguidamente abrirlo con MARS.

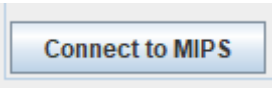


4. En las pestañas del programa, presione el apartado de "tools", seleccione bitmap display y configure la ventana con los siguientes parámetros:

- Unit Width in Pixels: 4
- Unit Height in Pixels: 4
- Display Width in Pixels: 512
- Display Height in Pixels: 512
- Base address for display: 0x10008000($gp): Registro "pila", las funciones del programa dependen de este registro.

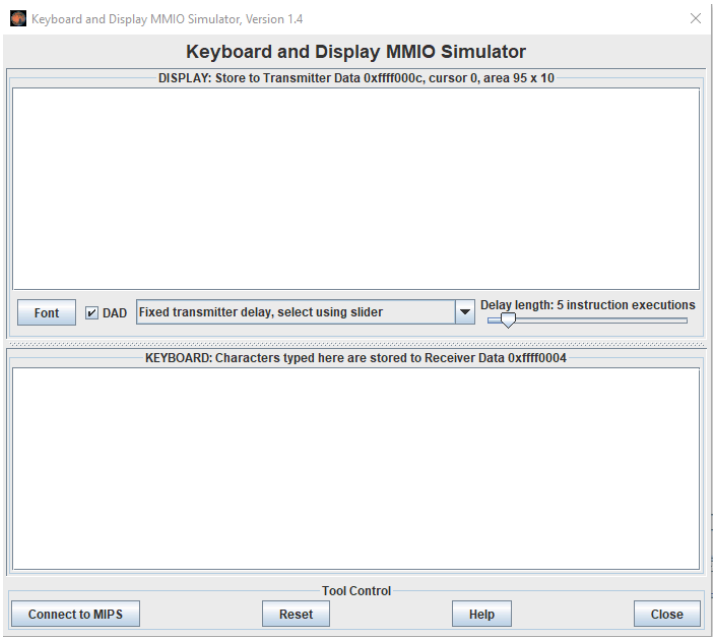


5. Presione el botón “Connect to MIPS.”

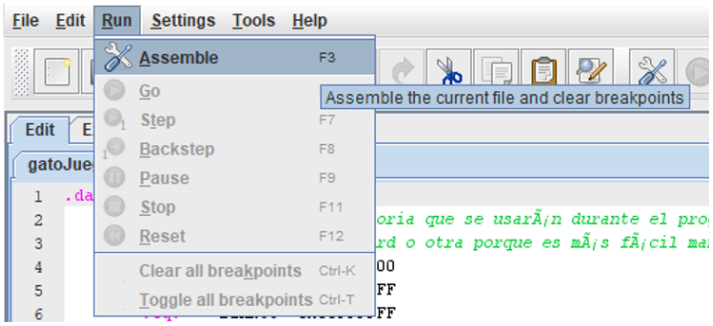


6. Busque en la pestaña “tools” la opción “Keyboard and Display MMIO Simulator” y seleccione la opción “Connect to MIPS” para que pueda recibir las teclas w,s,d,a y enter, que son las necesarias para moverse e insertar ficha.

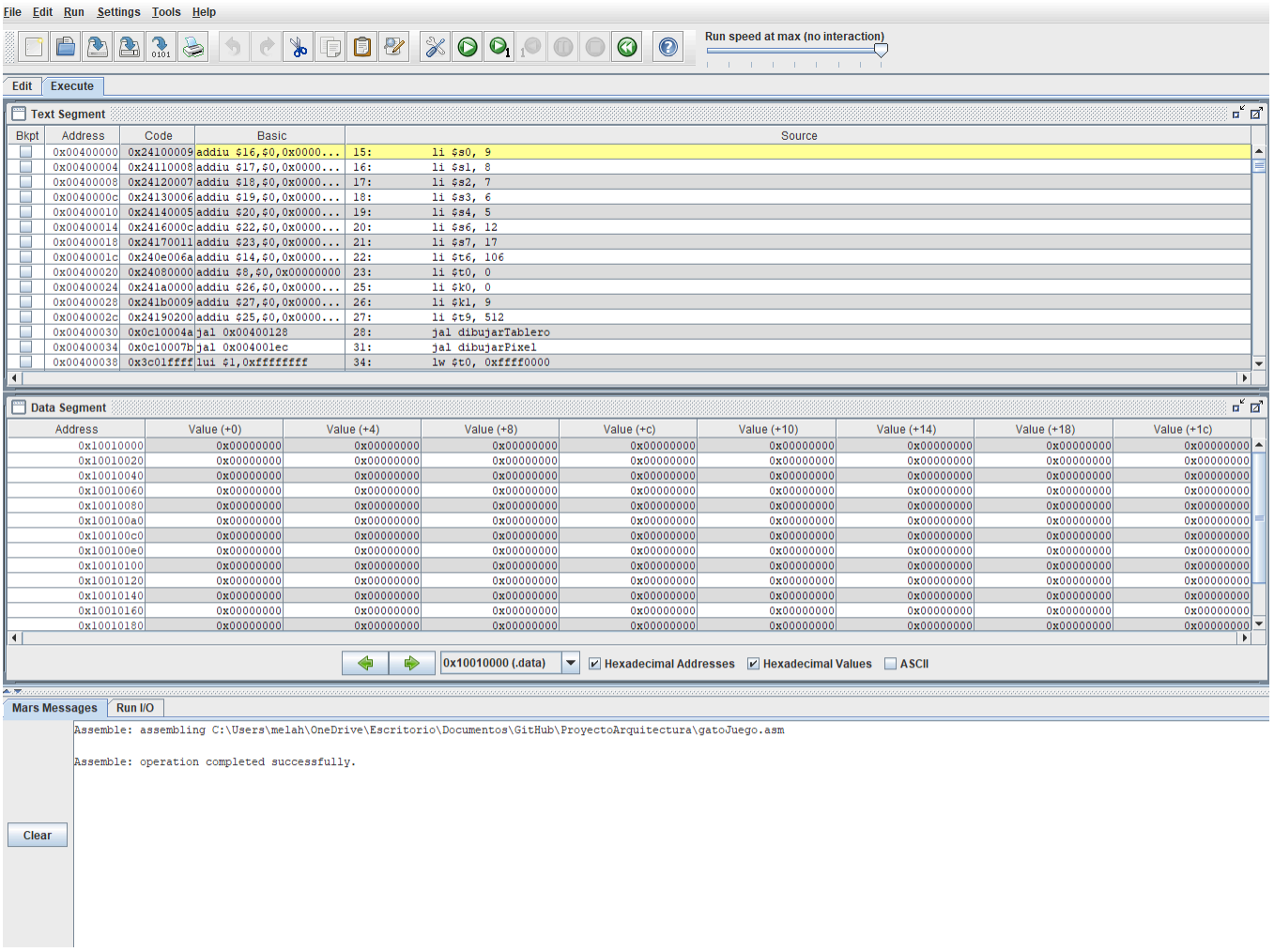
   Nota: con w(arriba), s(abajo), d(derecha), a(izquierda) se mueve en el tablero y con enter se selecciona el lugar donde pondrá la figura.



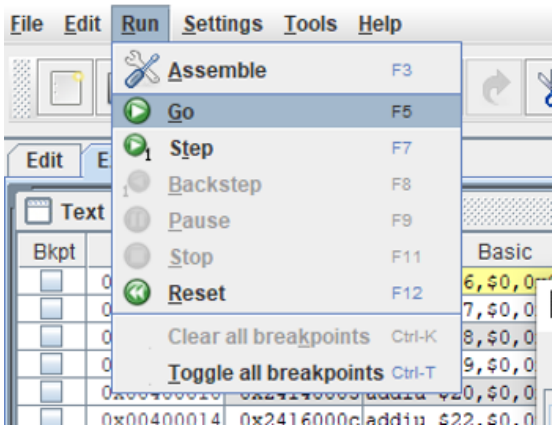
7. En la barra de las pestañas de MARS, busque la pestaña “run”. Seleccione la opción “Assemble para asegurarse que el programa compila.”



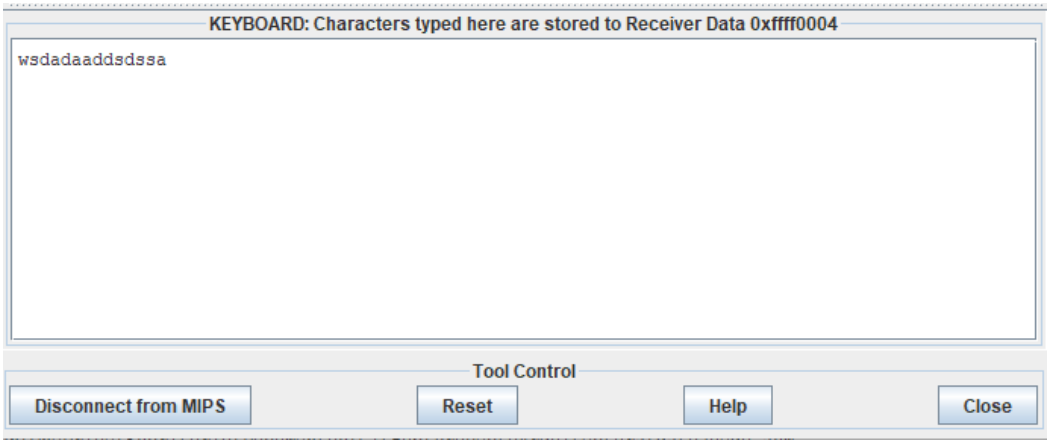
La pantalla debería verse de la siguiente manera:



8. En la pestaña “run” anteriormente ubicada, seleccione la opción “go”, para mostrar en BitMap Display el tablero de Gato.



9. Ubique la ventana “Keyboard and Display MMIO Simulator”. En la sección “KEYBOARD: Characters typed here are shorter to Receiver Data 0xffff0004”, proceda a escribir los comandos con las teclas w,s,d,a y enter, según el movimiento que son las necesarias para moverse e insertar ficha.



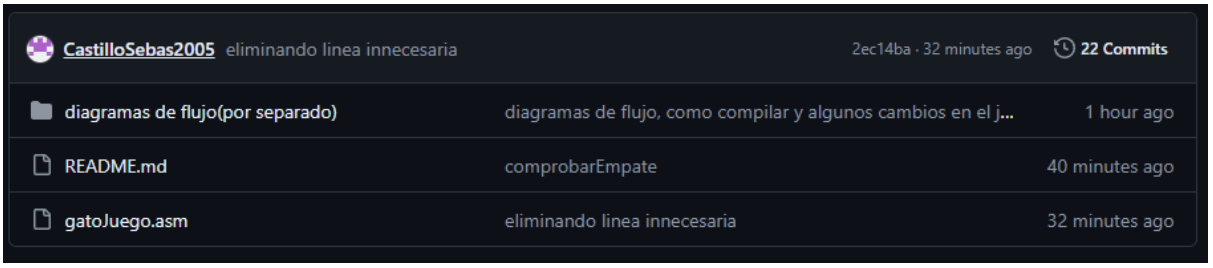
10. ¡Disfrute su partida!

Algunos aspectos a considerar:

- Si desea volver a jugar una partida, debe seleccionar el botón “reset” en ambas herramientas de MARS y proceder a compilar nuevamente, además de seleccionar “go”. Lo anterior para que se ejecute un nuevo programa.
- Si desea salir del programa, simplemente pulse la barra espaciadora en su teclado.

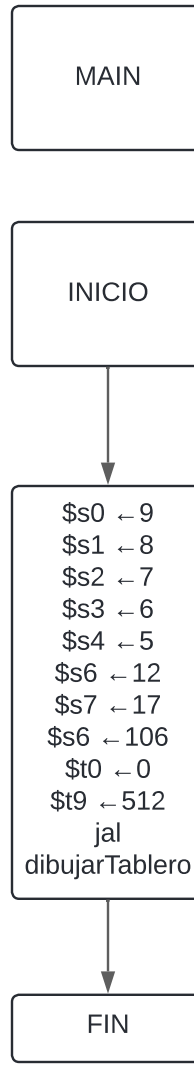
4. Contribuciones y progreso del proyecto documentado en un repositorio

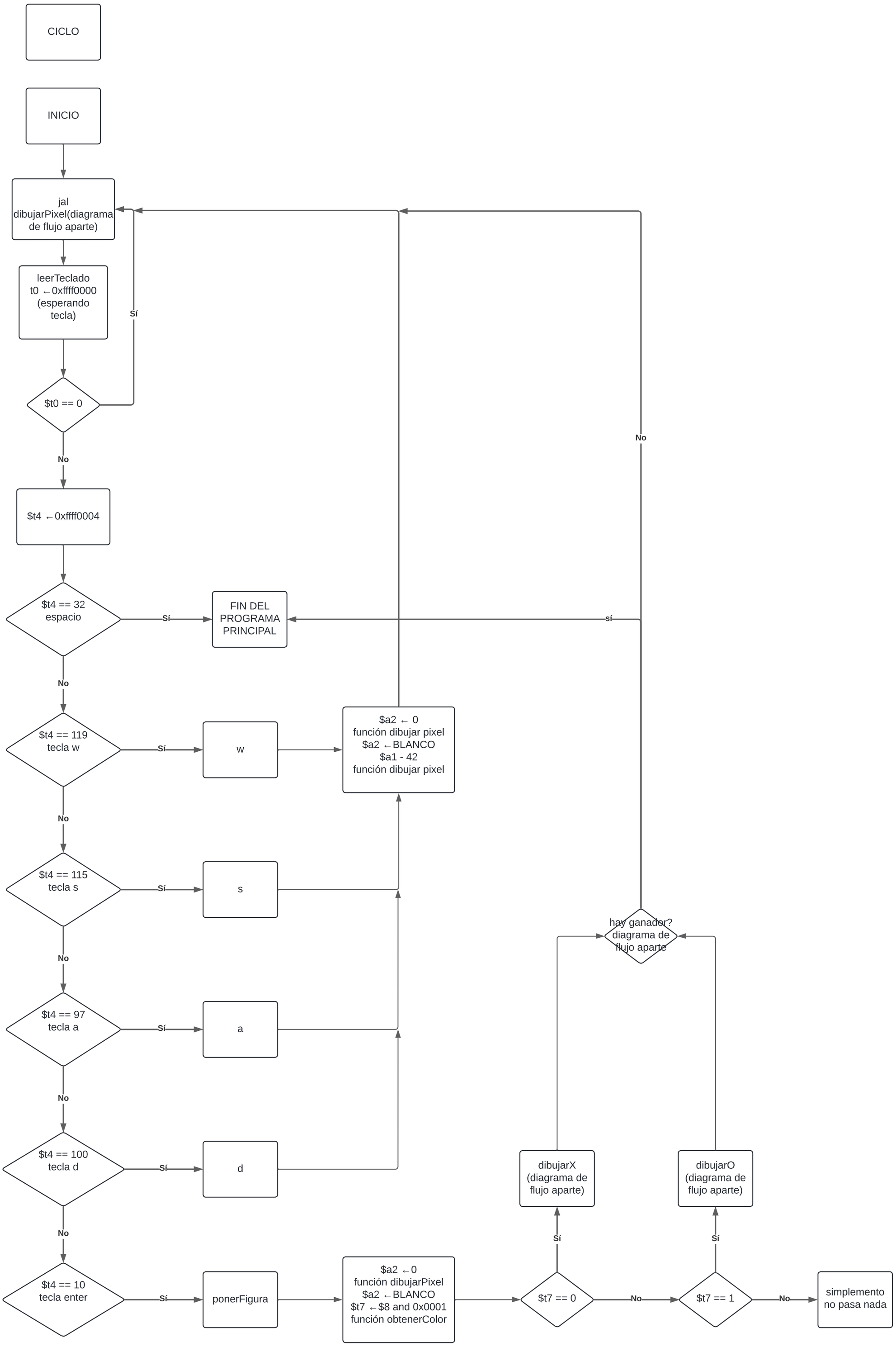
A continuación, se adjunta el enlace del repositorio en el que se trabajó el proyecto. Este contiene los siguientes archivos:



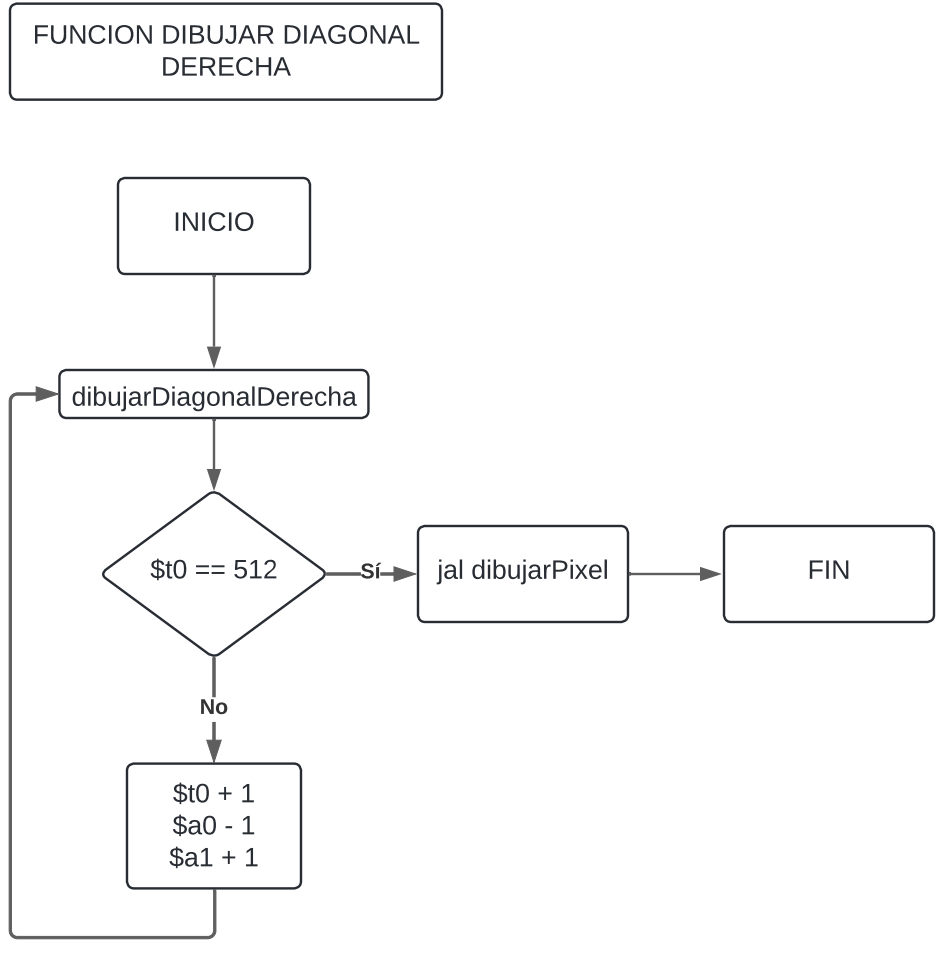
Nota: El repositorio se puede clonar.

https://github.com/CastilloSebas2005/ProyectoArquitectura.git

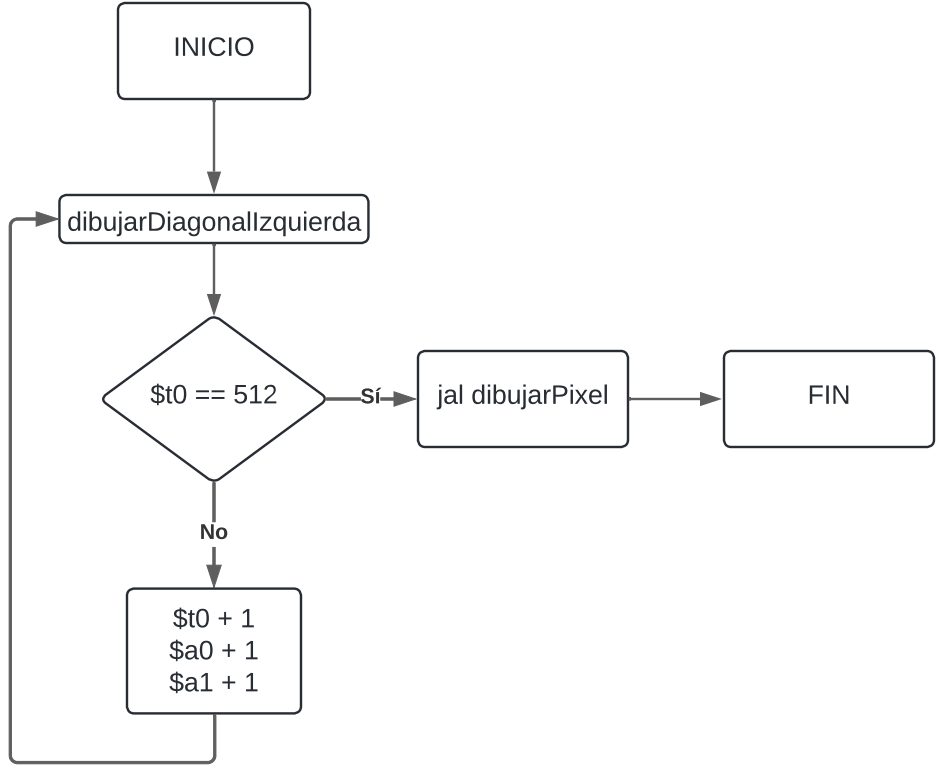

MAIN
INICIO
$s0 ← 9
$s1 ← 8
$s2 ← 7
$s3 ← 6
$s4 ← 5
$s6 ← 12
$s7 ← 17
$s6 ← 106
$t0 ← 0
$t9 ← 512
jal
dibujarTablero
FIN


CICLO
INICIO
jal dibujarPixel(diagrama de flujo aparte)
leerTeclado
t0 ← 0xffff0000
(esperando tecla)
$t0 == 0
Sí
No
$t4 ← 0xffff0004
$t4 == 32
espacio
Sí
FIN DEL PROGRAMA PRINCIPAL
No
$t4 == 119
tecla w
Sí
w
No
$t4 == 115
tecla s
Sí
s
No
$t4 == 97
tecla a
Sí
a
No
$t4 == 100
tecla d
Sí
d
No
$t4 == 10
tecla enter
Sí
ponerFigura
$a2 ← 0
función dibujarPixel
$a2 ← BLANCO
$t7 ← $8 and 0x0001
función obtenerColor
$t7 == 0
Sí
No
$t7 == 1
Sí
No
simplemento no pasa nada
$a2 ← 0
función dibujar pixel
$a2 ← BLANCO
$a1 - 42
función dibujar pixel
No
sí
hay ganador? diagrama de flujo aparte
dibujarX (diagrama de flujo aparte)
dibujarO (diagrama de flujo aparte)

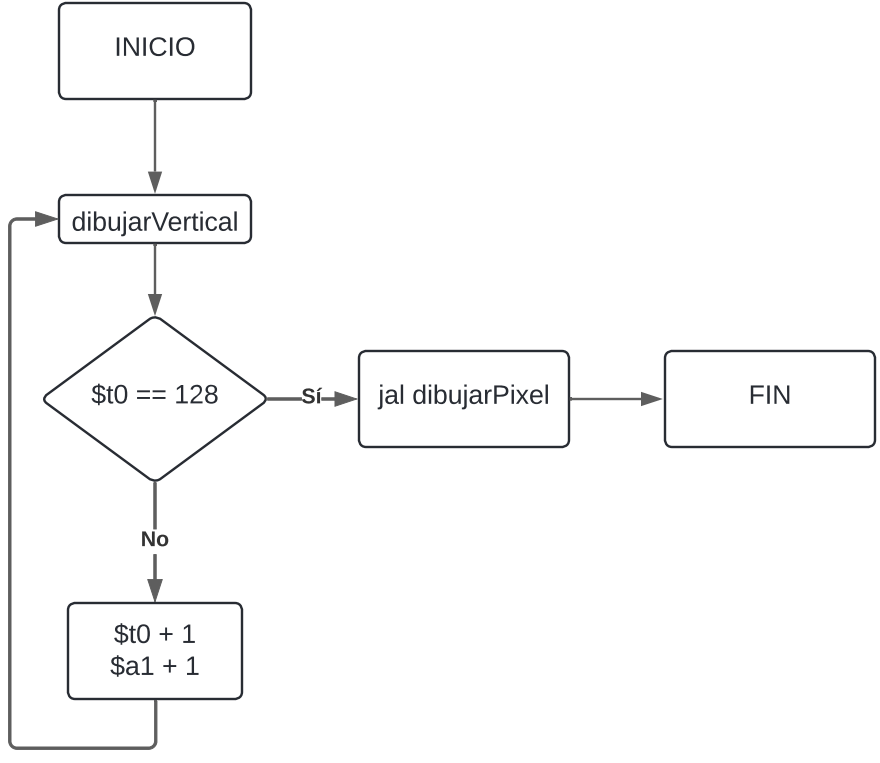

INICIO
$sp - 4
$ra ←(sp)
$t0 ←0
$a2 ←AMARILLO
Vertical1
$s0 == $s3
si
vertical1con2:
$a0 ←22
$a1 ←0
$s3 == $s6
si
no
vertical2
$s1 == $s4
si
vertical2con2:
$a0 ←64
$a1 ←0
$s4 == $s7
si
dibujarVertical
(tiene otro
diagrama)
no
vertical3
$s2 == $s5
si
vertical3con2:
$a0 ←107
$a1 ←0
$s5 == $s6
si
no
horizontal1
$s0 == $s1
si
horizontal1con2:
$a0 ←0
$a1 ←22
$s1==$s2
si
dibujarHorizontal
(otro diagrama)
no
horizontal2
$s3 == $s4
si
horizontal2con2:
$a0 ←0
$a1 ←64
$s1 == $s2
si
no
horizontal3
$s6 == $s7
si
horizontal2con2:
$a0 ←0
$a1 ←108
$s7 == $t6
si
no
dibujarDiagonalizquierda
(otro diagrama)
si
$4 == $s6
no
diagonalizquierdaCon2:
$a0 <- 0
$a1<-1
si
$s0 == $s4
no
backj
(otro diagrama)
(acá terminaría la función)
FIN
dibujarDiagonalDerecha
(otro diagrama)
si
$s4 == $s6
diagonalDerechaCon2:
$a0 <- 127
$a1 <- 0
$t0 <- 0
si
no
$s2 == $s4
no
diagonalizquierda
diagonalDerecha

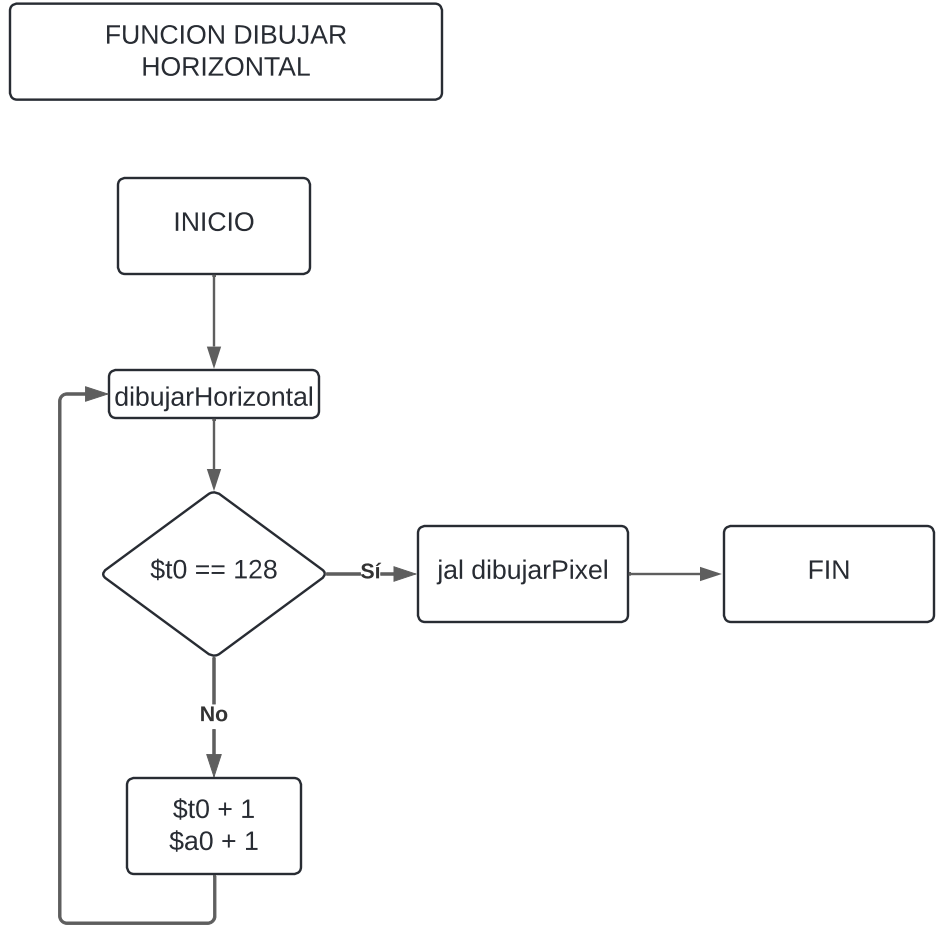

FUNCION DIBUJAR DIAGONAL DERECHA
INICIO
dibujarDiagonalDerecha
$t0 == 512
Sí
jal dibujarPixel
FIN
No
$t0 + 1
$a0 - 1
$a1 + 1

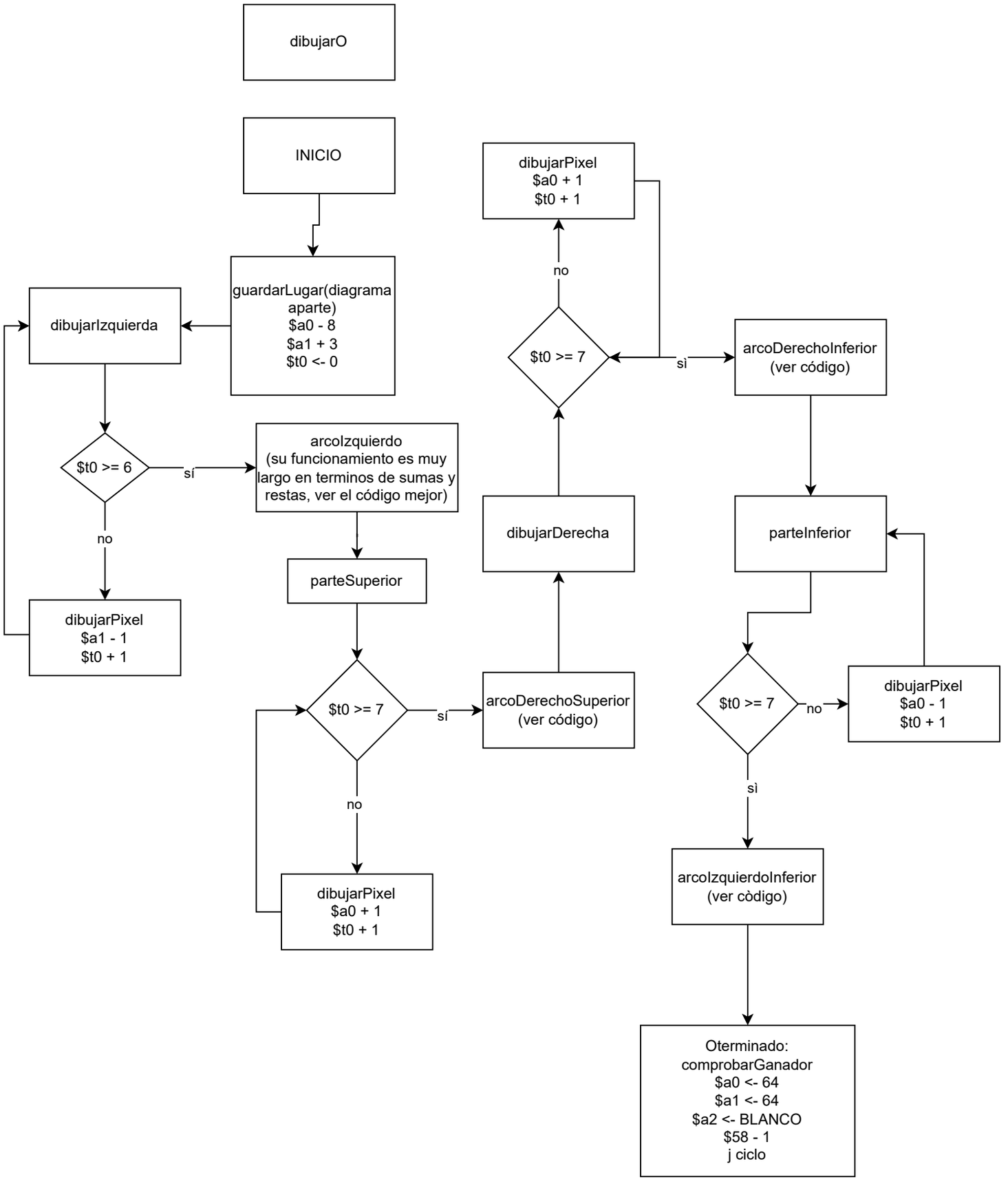
FUNCION DIBUJAR DIAGONAL IZQUIERDA

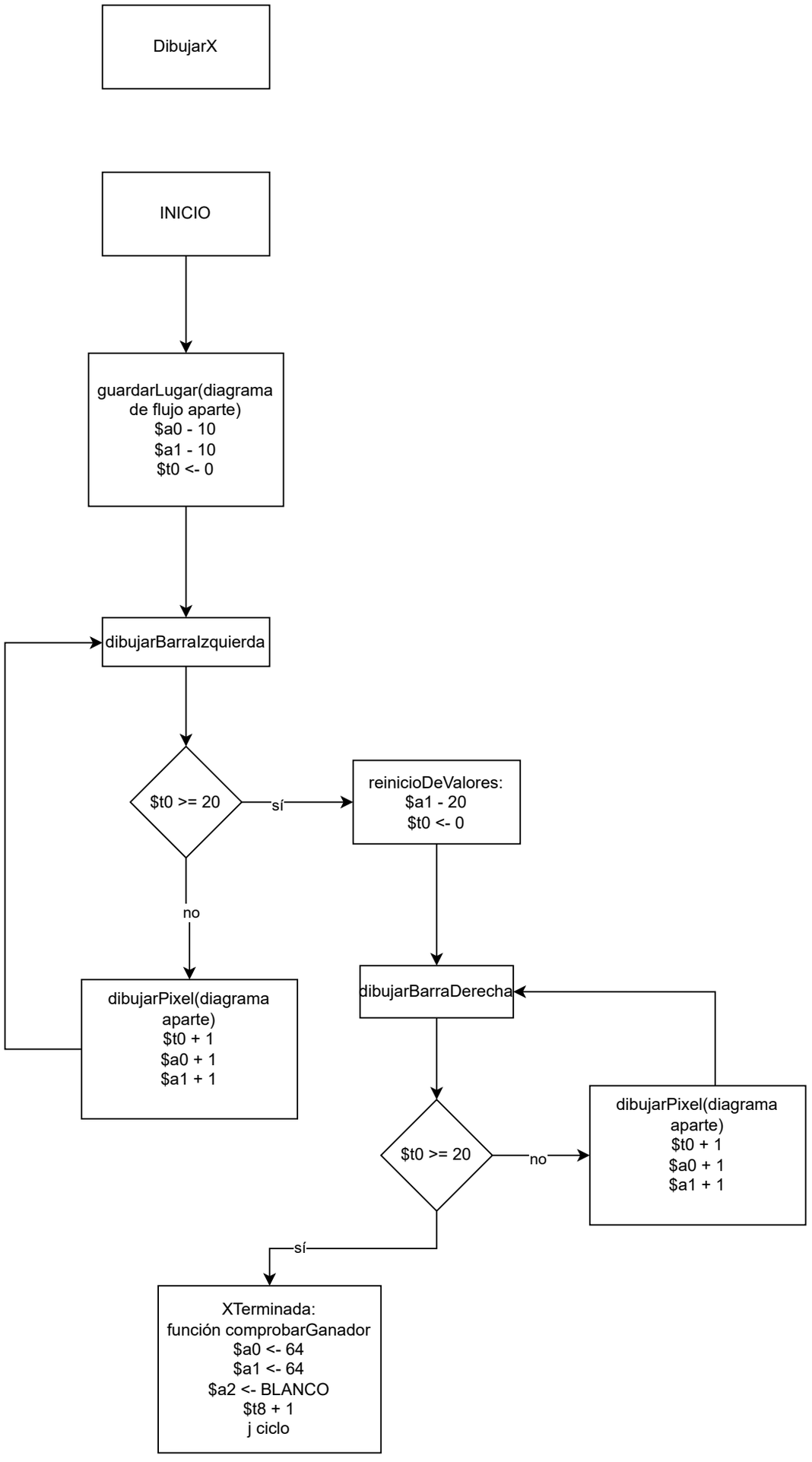


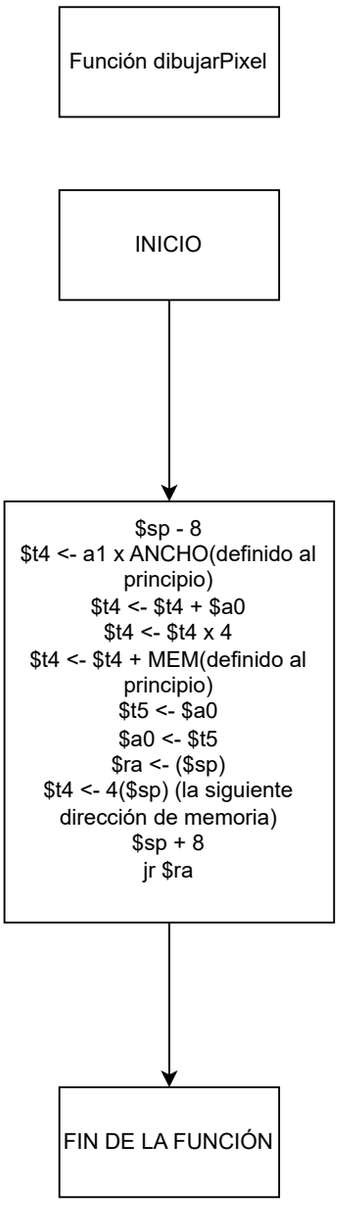
FUNCION DIBUJAR VERTICAL




FUNCION DIBUJAR HORIZONTAL
INICIO
dibujarHorizontal
$t0 == 128
Sí
jal dibujarPixel
FIN
No
$t0 + 1
$a0 + 1


dibujarO
INICIO
guardarLugar(diagrama aparte)
$a0 - 8
$a1 + 3
$t0 <- 0
dibujarIzquierda
$t0 >= 6
sí
no
arcoIzquierdo
(su funcionamiento es muy largo en terminos de sumas y restas, ver el código mejor)
dibujarPixel
$a1 - 1
$t0 + 1
parteSuperior
$t0 >= 7
sí
no
arcoDerechoSuperior
(ver código)
dibujarPixel
$a0 + 1
$t0 + 1
dibujarDerecha
$t0 >= 7
no
sì
dibujarPixel
$a0 + 1
$t0 + 1
arcoDerechoInferior
(ver código)
parteInferior
$t0 >= 7
no
sì
dibujarPixel
$a0 - 1
$t0 + 1
arcoIzquierdoInferior
(ver còdigo)
Oterminado:
comprobarGanador
$a0 <- 64
$a1 <- 64
$a2 <- BLANCO
$58 - 1
j ciclo


DibujarX
INICIO
guardarLugar(diagrama de flujo aparte)
$a0 - 10
$a1 - 10
$t0 <- 0
dibujarBarraIzquierda
$t0 >= 20
sí
reinicioDeValores:
$a1 - 20
$t0 <- 0
no
dibujarPixel(diagrama aparte)
$t0 + 1
$a0 + 1
$a1 + 1
dibujarBarraDerecha
$t0 >= 20
no
dibujarPixel(diagrama aparte)
$t0 + 1
$a0 + 1
$a1 + 1
sí
XTerminada:
función comprobarGanador
$a0 <- 64
$a1 <- 64
$a2 <- BLANCO
$t8 + 1
j ciclo


Función dibujarPixel
INICIO
$sp - 8
$t4 <- a1 x ANCHO(definido al principio)
$t4 <- $t4 + $a0
$t4 <- $t4 x 4
$t4 <- $t4 + MEM(definido al principio)
$t5 <- $a0
$a0 <- $t5
$ra <- ($sp)
$t4 <- 4($sp) (la siguiente dirección de memoria)
$sp + 8
jr $ra
FIN DE LA FUNCIÓN

FUNCIÓN DIBUJAR TABLERO

INICIO

$SP - 4(reservar espacio en el registro pila)
$ra ← (sp)
$a2 ← BLANCO
$a0 ← 171
$a1 ← 0

DibujarVerticalIzquierda

$t0 >= $t9

Sí

DibujarVerticalCentral

No

jal dibujarPixel
$t0 + 1
$a1 + 1

$t0 ← 0
$a0 ← 342
$a1 ← 0

DibujarVerticalDerecha

$t0 >= $t9

Sí

DibujarPrimeraLínea

No

jal dibujarPixel
$t0 + 1
$a1 + 1

$t0 ← 0
$a0 ← 0
$a1 ← 43
$t9 ← 128

DibujarHorizontalSuperior

$t0 >= $t9

Sí

dibujarHorizontalCentral

No

jal dibujarPixel
$t0 + 1
$a1 + 1

$t0 ← 0
$a0 ← 0
$a1 ← 86
$t9 ← 128

DibujarHorizontalInferior

$t0 >= $t9

Sí

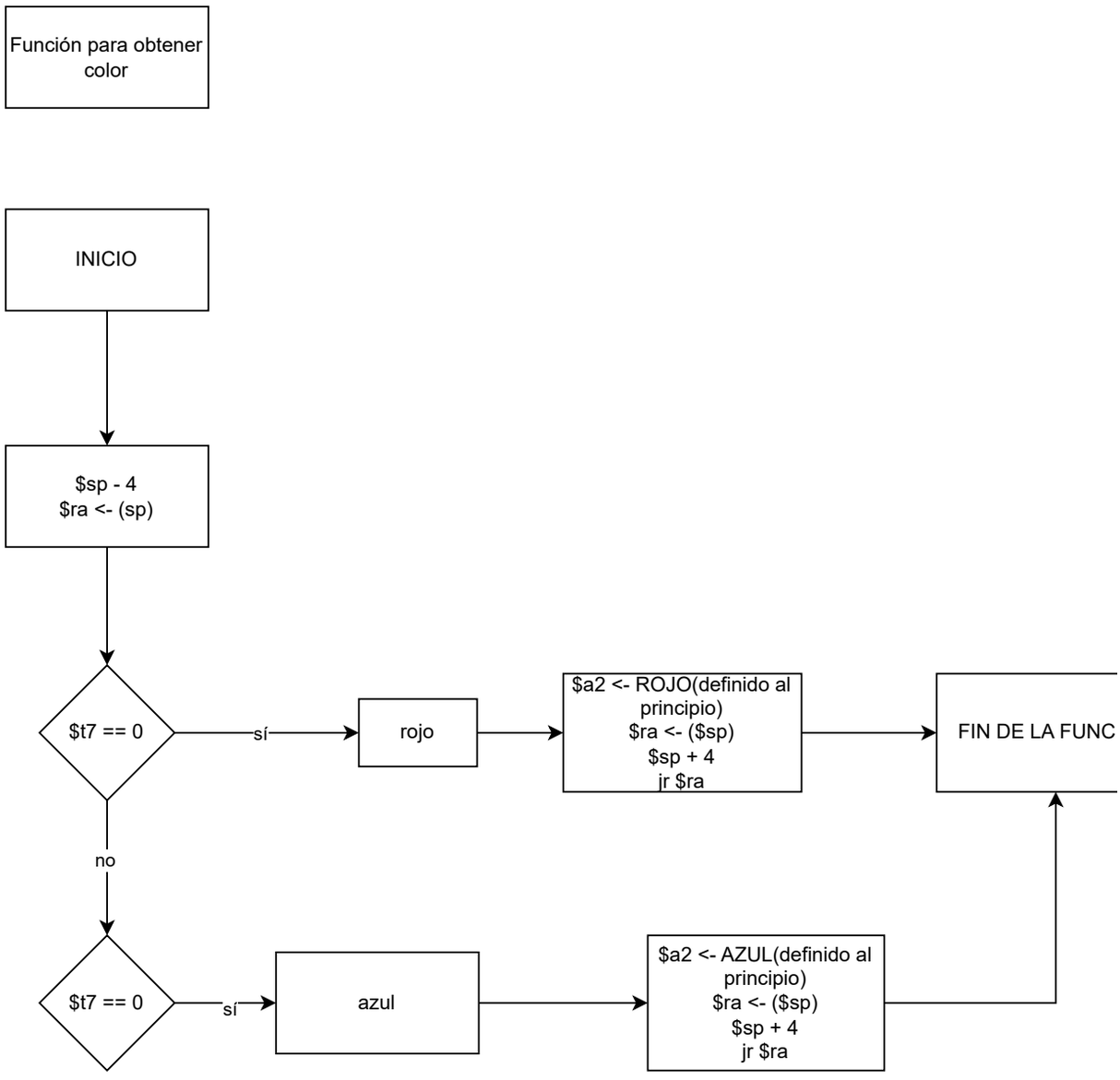
PosicionarSelector

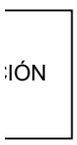
No

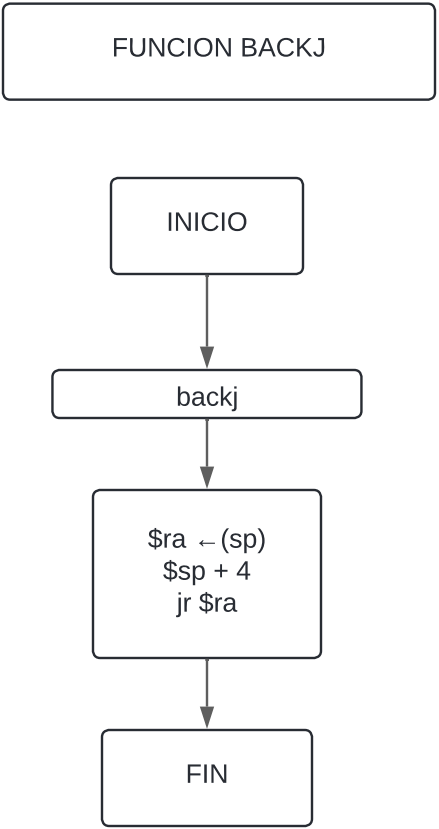
jal dibujarPixel
$t0 + 1
$a1 + 1

$a0 ← 64
$a1 ← 64
$ra ← (sp)
$sp +4

FIN


Función para obtener color
INICIO
$sp - 4
$ra <- (sp)
$t7 == 0
sí
rojo
$a2 <- ROJO(definido al principio)
$ra <- ($sp)
$sp + 4
jr $ra
FIN DE LA FUNC
no
$t7 == 0
sí
azul
$a2 <- AZUL(definido al principio)
$ra <- ($sp)
$sp + 4
jr $ra


CIÓN


FUNCION BACKJ
INICIO
backj
$ra ← (sp)
$sp + 4
jr $ra
FIN