# Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías



División de Tecnologías para la Integración CiberHumana

**Ingeniería en Computación**

**Programación de Bajo Nivel**

D02 - IL358 - 209850

## 7. Circunferencia de Un Círculo

**Profesor**: José Juan Meza Espinoza

**Alumno**: Alan Yahir Juárez Rubio

**Código**: 218517809

**Correo**: alan.juarez5178@alumnos.udg.mx



16 de noviembre de 2024

# Índice

# Índice de figuras

# Índice de códigos

# 7. Circunferencia de Un Círculo

## 1. Introducción

Para dibujar un círculo a través de la manipulación de píxeles, existen dos algoritmos muy conocidos el **algoritmo de punto medio para circunferencias** y el **algoritmo de Bresenham**, implementado en el código 1.

El **algoritmo de Bresenham** es un algoritmo eficiente y ampliamente utilizado en gráficos por computadora para dibujar círculos utilizando únicamente operaciones de suma, resta y desplazamientos. Este algoritmo se basa en la simetría del círculo, lo que permite reducir significativamente la cantidad de cálculos necesarios al trazar puntos en las diferentes octantes del círculo.

## 2. Implementación

```
; Macros
%macro assign 2
    MOV ax, [%2]
    MOV [%1], ax
%endmacro

%macro negate 1
    MOV ax, [%1]
    NEG ax
    MOV [%1], ax
%endmacro

%macro inc_var 1
    MOV ax, [%1]
    INC ax
    MOV [%1], ax
%endmacro

%macro dec_var 1
    MOV ax, [%1]
    DEC ax
    MOV [%1], ax
%endmacro

%macro compare_2_vars 2
    MOV cx, [%1]
    CMP cx, [%2]
%endmacro

%macro compare_var_and_num 2
    MOV cx, [%1]
    CMP cx, %2
%endmacro

%macro add_and_assign 3
    MOV ax, [%2]
    ADD ax, [%3]
    MOV [%1], ax
%endmacro

%macro sub_and_assign 3
    MOV ax, [%2]
    SUB ax, [%3]
    MOV [%1], ax
%endmacro

%macro add_3_nums_and_assign 4
    MOV ax, [%2]
    ADD ax, [%3]
    ADD ax, [%4]
    MOV [%1], ax
%endmacro

%macro sub_3_nums_and_assign 4
    MOV ax, [%2]
    SUB ax, [%3]
    SUB ax, [%4]
    MOV [%1], ax
%endmacro
```

```
60
61 ; Macro para dibujar un píxel
62 %macro draw_pixel 2
63     MOV cx, [%1]  ; columna
64     MOV dx, [%2]  ; fila
65
66     MOV al, GREEN  ; color verde
67     MOV ah, 0ch    ; función para poner píxel
68     INT 0x10       ; interrupción de video
69 %endmacro
70
71 ; Macro para dibujar un círculo
72 %macro draw_circle 3
73     assign y_off, %3
74     assign balance, %3
75     negate balance
76
77     draw_circle_loop:
78         add_and_assign x_plus_x, %1, x_off
79         sub_and_assign x_minus_x, %1, x_off
80         add_and_assign y_plus_y, %2, y_off
81         sub_and_assign y_minus_y, %2, y_off
82
83         add_and_assign x_plus_y, %1, y_off
84         sub_and_assign x_minus_y, %1, y_off
85         add_and_assign y_plus_x, %2, x_off
86         sub_and_assign y_minus_x, %2, x_off
87
88         draw_pixel x_plus_y, y_minus_x
89         draw_pixel x_plus_x, y_minus_y
90         draw_pixel x_minus_x, y_minus_y
91         draw_pixel x_minus_y, y_minus_x
92         draw_pixel x_minus_y, y_plus_x
93         draw_pixel x_minus_x, y_plus_y
94         draw_pixel x_plus_x, y_plus_y
95         draw_pixel x_plus_y, y_plus_x
96
97         add_3_nums_and_assign balance, balance, x_off, x_off
98
99         compare_var_and_num balance, 0
100        JL balance_negative
101
102        dec_var y_off
103        sub_3_nums_and_assign balance, balance, y_off, y_off
104
105        balance_negative:
106            inc_var x_off
107
108            compare_2_vars x_off, y_off
109            JG end_drawing
110            JMP draw_circle_loop
111
112        end_drawing:
113 %endmacro
114
115 section .data
116     ; Constantes
117     GREEN equ 0xA
118
119     ; Variables inicializadas
120     x DW 80 ; coordenada X del centro
```

```
    y DW 80 ; coordenada Y del centro
    r DW 20 ; radio del círculo

    balance DW 0
    x_off DW 0
    y_off DW 0

    x_plus_x DW 0
    x_minus_x DW 0
    y_plus_y DW 0
    y_minus_y DW 0

    x_plus_y DW 0
    x_minus_y DW 0
    y_plus_x DW 0
    y_minus_x DW 0

section .text
    global _start

_start:
    org 0x100

    MOV ah, 0    ; función para establecer modo de video
    MOV al, 0x13 ; modo 0x13 = 320x200 píxeles, 256 colores
    INT 0x10     ; configurar el modo de video

    draw_circle x, y, r

    ; Esperar pulsación de tecla
    MOV ah, 0
    INT 0x16

    ; Volver al modo texto
    MOV ah, 0    ; función para establecer modo de video
    MOV al, 3    ; modo texto estándar
    INT 0x10     ; configurar el modo
```

Código 1: Impresión de la círcunferencia de un círculo a través de píxeles

## 3. Ejecución del Programa

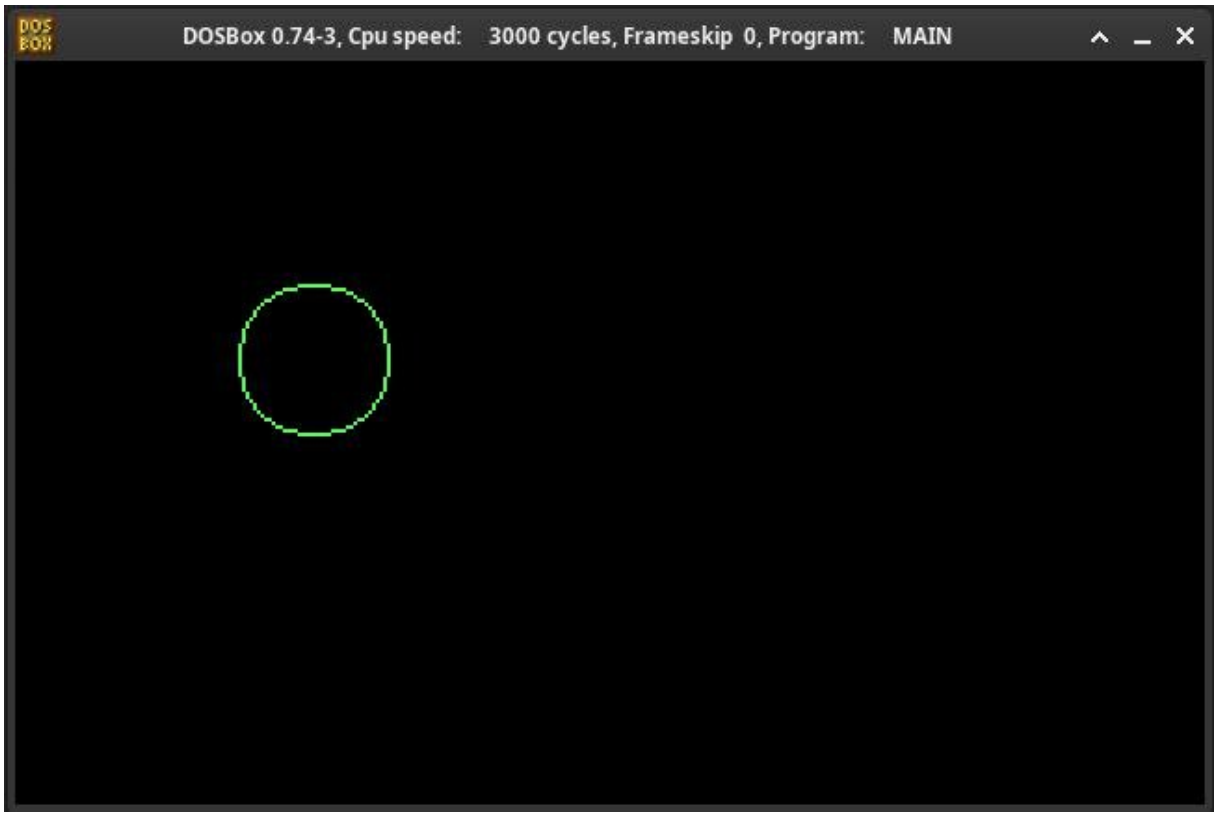


Fig. 1: Impresión de la círcunferencia de un círculo a través de píxeles

## 4. Conclusión

En retrospectiva, este código demuestra cómo implementar el **algoritmo de Bresenham** en ensamblador para el trazado de la circunferencia de un círculo mediante el uso de gráficos básico en DOS. Este algoritmo es muestra una solución eficiente y óptima para equipos de escasos recursos.

Para finalizar, es de vital importancia mencionar que para el código 1 fue implementado el uso de macros. Estas fueron de vital importancia debido a que ayudaron a hacer el código más modular, legible y menos repetitivo, aprovechando las ventajas que estas ofrecen. Cabe mencionar que estas son muy similares a las funciones, solo que las macros son más eficientes en términos de eficiencia computacional, mientras que las funciones son más eficientes en términos de memoria.

## Referencias

[1] Tutorials Point, "Assembly - macros." https://www.tutorialspoint.com/assembly_programming/assembly_macros.htm, s.f. Consultado el 16 de noviembre de 2024.

[2] Geeks for Geeks, "Bresenham's circle drawing algorithm." https://www.geeksforgeeks.org/bresenhams-circle-drawing-algorithm/, octubre 2024. Consultado el 16 de noviembre de 2024.

[3] Anónimo, "Win32 Programming in NASM – Part I." https://bitcodersblog.wordpress.com/2017/05/10/win32-in-nasm-part-1/, s.f. Consultado el 16 de noviembre de 2024.