UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

FACULTAD DE PRODUCCIÓN Y SERVICIOS

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Asignatura: Arquitectura de Computadoras

Grupo: Laboratorio – 'A'

Tema: Trabajo 1 – Barra de Progreso

Docente: Rodríguez González, Pedro Alex

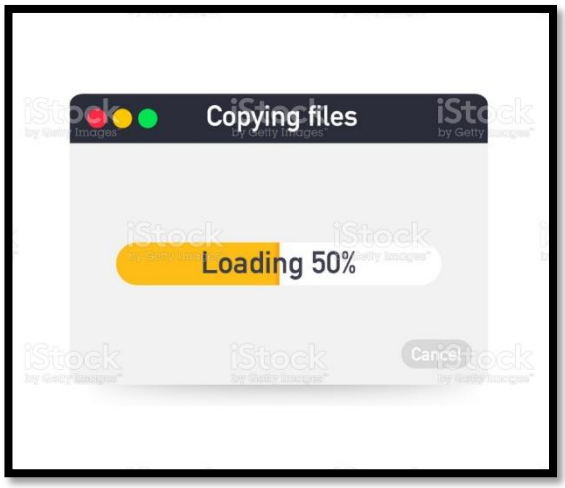
Alumno: Hincho Jove, Angel Eduardo

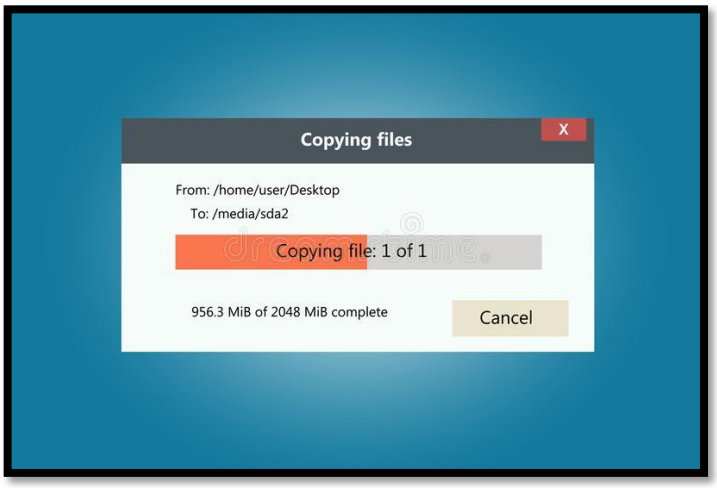
Arequipa - Perú

Octubre 2021

1. Descripción del Problema

- Para el presente Laboratorio y Guía 1, se nos pide implementar un programa usando el simulador EMU8086 que represente o simula una 'Barra de Progreso' como las que podemos ver al momento de subir archivos a Internet o cargar archivos. El ejemplo dado en clase fue las 'Barras de Progreso' que se ven normalmente al copiar o fotocopiar con una impresora algún documento.





2. Limitantes del Problema

- Algunas limitantes expuestas en Laboratorio: La 'Barra de Progreso' debe estar orientada verticalmente y debe estar centrada acorde a la pantalla de usuario.

3. Evidencias del Programa en EMU8086

- Captura del Código Parcial desarrollado para la 'Barra de Progreso'.

```
087
088 ; Cambiando el color de mi nombre a letras en blanco
089
090 mov cx, 26  ; Cantidad de pixeles que existen horizontalmente
091 mov di, 0fh ; Empezando en el primer bit de la segunda linea
092
093 nombreBlanco:
094
095     ; Recorrer desde la posicion donde empieza mi nombre y
096     ; asignar el bit de los datos una letra de color blanco
097
098     mov [di], 00001111b
099     add di, 2
100
101     loop nombreBlanco
102
103 ; Marco horitzontal de la pantalla
104
105 mov cx, 40  ; Cantidad de pixeles que existen horizontalmente
106 mov di, 51h ; Empezando en el primer bit de la segunda linea
107
108 marcoHorizontal:
109
110     ; Texto negro (0000) en fondo blanco (1111).Para la
111     ; parte superior de la pantalla como la inferior
112
113     mov [di], 11110000b
114     add di, 730h
115
116     mov [di], 11110000b
117     sub di, 730h
118
119     add di, 2
120
121     loop marcoHorizontal
```

- Captura de la vista final en la Pantalla de Usuario de la ‘Barra de Progreso’.

4. Link al Vídeo con ejecución del Programa

Vídeo en YouTube: https://youtu.be/iblbXO_rYJ8

5. Código del Programa Assembler en GitHub

Para una mayor disponibilidad se hace entrega del código del Programa de dos formas: Subido en un repositorio GitHub y ‘Copiado y Pegado’ al mismo Informe.

En este apartado se puede encontrar un repositorio con el archivo .asm que contiene el código realizado para la ‘Barra de Progreso’ solicitada.

Link del repositorio en GitHub: https://github.com/ahincho/LabAC_GrupoA.git

6. Código del Programa Assembler en EMU8086

```
name "progress-bar"

; Grupo: Laboratorio de AC - 'A'
; Autor: Hincho Jove, Angel Eduardo

org 100h

; Modo video
mov ax, 0     ; Texto en pantalla de 40x25
int 10h       ; Interrupcion

; Cancelar el blinking o brillo para los colores:
mov ax, 1003h
mov bx, 0
int 10h

; Segmento de registro
```

```
mov     ax, 0b800h
mov     ds, ax

; Dibujar barra de progreso en pantalla y el nombre del alumno

; Hincho Jove, Angel Eduardo -> Tiene 26 caracteres
; La pantalla permite mostrar en una linea 40 caracteres.

; Pixel de inicio -> 8 -> Para que el nombre este centrado

; Debemos multiplicar por 2 a 8 ya que el bit par almacena el
; caracter y el impar almacena metadatos como fore/back-ground

; 8 x 2 = 16 - 1 (Debido a que siempre empazamos en posicion = 0)
; Teniendo esto lo convertimos decimal a hexadecimal (15d = 0eh)

; Empezamos a 'escribir' desde la posicion 0eh

mov [0eh], 'H'

mov [10h], 'i'

mov [12h], 'n'

mov [14h], 'c'

mov [16h], 'h'

mov [18h], 'o'

mov [1ah], ' '

mov [1ch], 'J'

mov [1eh], 'o'

mov [20h], 'v'

mov [22h], 'e'

mov [24h], ','

mov [26h], ' '

mov [28h], 'A'

mov [2ah], 'n'

mov [2ch], 'g'

mov [2eh], 'e'

mov [30h], 'l'

mov [32h], ' '

mov [34h], 'E'

mov [36h], 'd'

mov [38h], 'u'
```

```
mov [3ah], 'a'

mov [3ch], 'r'

mov [3eh], 'd'

mov [40h], 'o'

; Cambiando el color de mi nombre a letras en blanco

mov cx, 26  ; Cantidad de pixeles que existen horizontalmente
mov di, 0fh ; Empezando en el primer bit de la segunda linea

nombreBlanco:

    ; Recorrer desde la posicion donde empieza mi nombre y
    ; asignar el bit de los datos una letra de color blanco

    mov [di], 00001111b
    add di, 2

    loop nombreBlanco

; Marco horitzontal de la pantalla

mov cx, 40  ; Cantidad de pixeles que existen horizontalmente
mov di, 51h ; Empezando en el primer bit de la segunda linea

marcoHorizontal:

    ; Texto negro (0000) en fondo blanco (1111).Para la
    ; parte superior de la pantalla como la inferior

    mov [di], 11110000b
    add di, 730h

    mov [di],  11110000b
    sub di, 730h

    add di, 2

    loop marcoHorizontal

; Marco vertical de la pantalla

mov cx, 22  ; Cantidad de pixeles que existen verticalmente
mov di, 0a1h ; Empezando en la tercera linea

marcoVertical:

    ; Texto negro (0000) en fondo blanco (1111). Tanto para la
    ; parte de la izquierda como la izquierda de la pantalla

    mov [di], 11110000b
    add di, 4eh

    mov [di], 11110000b
    add di, 2

    loop marcoVertical

; Ahora vamos a graficar nuestra barra de progreso y sus bordes
```

```
; Parte superior o 'punta' de la barra de progreso

mov [1b5h], 11110000b

mov [1b7h], 11110000b

mov [1b9h], 11110000b

mov [1bbh], 11110000b

; Borde horizontal de la barra de progreso

mov cx, 10  ; Ancho de barra de progreso en pixeles en pantalla
mov di, 1ffh ; Bit en el cual se empieza a 'colorear'

bordeHorizontal:

    ; Texto negro (0000) en fondo blanco (1111). Para poder hacer
    ; los bordes superiores e inferiores de la barra de progreso

    mov [di], 11110000b
    add di, 3c0h

    mov [di], 11110000b
    sub di, 3c0h

    add di, 2

    loop bordeHorizontal

; Borde vertical de la barra de progreso

mov cx, 11  ; Altura del cuerpo de barra de progreso en pixeles
mov di, 24fh ; Bit en el cual se empieza a 'colorear'

bordeVertical:

    ; Texto negro (0000) en fondo blanco (1111). Para poder hacer
    ; los bordes de derecha e izquierda de la barra de progreso

    mov [di], 11110000b
    add di, 4

    mov [di], 01111111b
    add di, 2

    mov [di], 01111111b
    add di, 2

    mov [di], 01111111b
    add di, 2

    mov [di], 01111111b
    add di, 2

    mov [di], 01111111b
    add di, 2

    mov [di], 01111111b
    add di, 4
```

```
    mov [di], 11110000b
    add di, 62

    loop bordeVertical

; Texto indicando que se esta 'copiando ...' -> 12 caracteres
; con texto en blanco (1111) y fondo negro (0000)

mov [65ch], 'C'

mov [65eh], 'O'

mov [660h], 'P'

mov [662h], 'I'

mov [664h], 'A'

mov [666h], 'N'

mov [668h], 'D'

mov [66ah], 'O'

mov [66ch], ' '

mov [66eh], '.'

mov [670h], '.'

mov [672h], '.'

; Cambiando el color del texto 'copiando ...' a letras en blanco

mov cx, 12  ; Cantidad de pixeles que existen horizontalmente
mov di, 65dh ; Empezando en el primer bit de datos 'C'

textoBlanco:

    mov [di], 00001111b
    add di, 2

    loop textoBlanco

; Pintar la subida en la 'copiando' de la barra de progreso

mov cx, 11  ; Ancho de la barra de progreso en pixeles
mov di, 57dh ; Bit en el cual se empieza a 'colorear'

colorearCarga:

    ; Colorea luego pasa al siguiente. Una vez coloreados
    ; toda la fila, se regresa o 'sube' a la anterior
    ; esto se logra usando la instruccion de resta 'sub'

    mov [di], 10100000b
    sub di, 2

    mov [di], 10100000b
    sub di, 2

    mov [di], 10100000b
```

```
    sub di, 2

    mov [di], 10100000b
    sub di, 2

    mov [di], 10100000b
    sub di, 2

    mov [di], 10100000b
    sub di, 70

    loop colorearCarga

; Texto indicando que el proceso ha 'finalizado' -> 10 caracteres
; con texto en blanco (1111) y fondo negro (0000)

mov [65ch], ' '

mov [65eh], 'F'

mov [660h], 'I'

mov [662h], 'N'

mov [664h], 'A'

mov [666h], 'L'

mov [668h], 'I'

mov [66ah], 'Z'

mov [66ch], 'A'

mov [66eh], 'D'

mov [670h], 'O'

mov [672h], ' '

; Cambiando el color del texto 'finalizado' a letras en blanco

mov cx, 10  ; Cantidad de pixeles que existen horizontalmente
mov di, 65fh ; Empezando en el primer bit de datos 'C'

finBlanco:

    mov [di], 00001111b
    add di, 2

    loop finBlanco

; Finalizacion del programa
mov ah, 0
int 16h

ret
```