Práctica 9

Instrucciones de control de flujo.

Luis Eduardo Galindo Amaya (1274895)

Asignatura	Organización de Computadoras (331)
Docente	Arturo Arreola Alvarez
Fecha	21-10-2022

Instrucciones de control de flujo.

Luis Eduardo Galindo Amaya (1274895)

21-10-2022

Objetivo

Seleccionar las instrucciones de control de flujo del programa adecuadas, para desarrollar aplicaciones de sistemas basados en microp

Desarrollo

Programe las siguientes rutinas usando las instrucciones de control de flujo del procesador 80386 y las rutinas en la biblioteca de funciones libpc_io.a. Escriba su código fuente en un archivo llamado P8.asm, ensamble el código con NASM y encadénelo con el comando ld.

a) gets

Almacena en memoria una cadena de caracteres ingresada por el usuario por medio del teclado. La captura termina cuando el usuario presiona la tecla ENTER. La cadena se almacena a partir de la dirección EDX y con un 0 (null) al final que indica el fin de cadena. Cuando el usuario teclea la cadena, se deben visualizar en pantalla los caracteres ingresados.

Una vez que tenga funcional la rutina, modifíquela de forma que acepte la tecla BACKSPACE. Cada que el usuario presiona esta tecla, el último carácter de la cadena es borrado de pantalla y de memoria. La rutina no debe usar gotoxy, haga uso de getch/getche y putchar.

Información: El código ASCII de la tecla ENTER es 10. El BACKSPACE es 8. Se hace uso de getch/getche para recibir cada carácter individual del teclado y se verifica si se recibió un ENTER ó BACKSPACE, para proceder a terminar la captura o hacer el borrado del último carácter ingresado.

b) getsAlpha

El procedimiento es similar a gets, a excepción de que sólo acepta caracteres del abecedario a – z y A – Z.

c) asteriscos

Colocar en CX la cantidad de renglones con asteriscos se desean imprimir. El procedimiento imprime las líneas con asteriscos. La primera línea cuenta con 1 asterisco, la segunda con 2, etc. Ejemplo: Si CX = 5 el procedimiento debe imprimir:

```
*
**
***
****
*****
```

Capturas

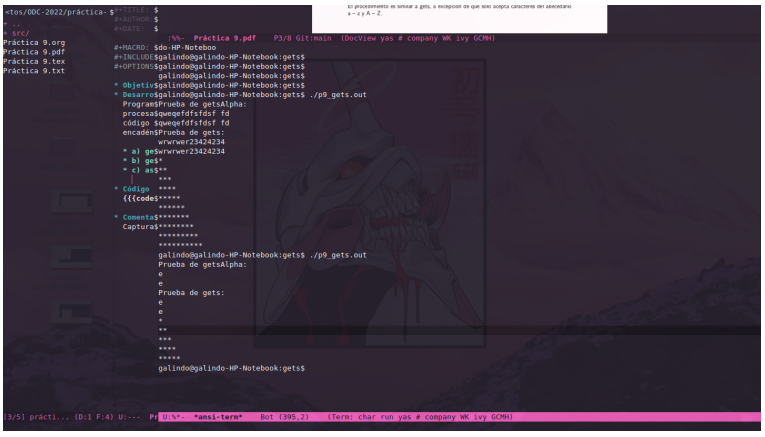

Figura 1: Programa funcionando

Código

```
1 ;;    AUTHOR: Luis Eduardo Galindo Amaya
2 ;;      DATE: 21-10-2022
3 ;; ASSEMBLE: nasm -f elf P9_gets.asm
4 ;;      LINK: ld -m elf_i386 P9_gets.o libpc_io.a -o p9_gets.out
5 ;;       RUN: ./p9_gets.out
6
```

```
%include "./lib/pc_io.inc"

section .data
    gets_message db 'Prueba de gets: ', 0xA, 0x0
    getsAlpha_message db 'Prueba de getsAlpha: ', 0xA, 0x0

section .bss
    string_captura_gets resb 255
    string_captura_getAlpha resb 255

section .text
global _start

_start:
    ;------------------------------------------------------ Test getsAlpha
    mov edx, getsAlpha_message
    call puts

    mov ebx, string_captura_getAlpha
    call getsAlpha

    mov edx, string_captura_getAlpha
    call puts                   ;mostrar el valor

    mov al, 10                  ;salto de linea
    call putchar

    ;------------------------------------------------------ Test gets
    mov edx, gets_message
    call puts

    mov ebx, string_captura_gets
    call gets

    mov edx, string_captura_gets
    call puts                   ;mostrar el valor capturado

    mov al, 10
    call putchar                ;salto de linea

    ;------------------------------------------------------ Test asterisks
    mov cx, 5
    call asterisks

    ; TERMINAR PROGRAMA
        mov eax, 1
        mov ebx, 0
        int 80h
```

```
55
56
57 ;; asterisks
58 ;; ========
59 ;; - CX maximo numero de astericos
60 ;;
61 ;; Imprime un triangulo de asteriscos hasta llegar a el
62 ;; numero que indica CX
63 asterisks:
64     mov dx, 0                   ;numero de lineas inicial
65     mov bx, 0                   ;numero de asteriscos inicial
66
67 .loopLines:                     ;---------------- inicio loopLines
68     mov dx, cx                  ;guardar el numero de lineas
69                                 ;en el iterador (cx)
70
71     inc bx                      ;incrementar el numero de
72                                 ;asteriscos
73
74     mov cx, bx                  ;guardar el numero de asteriscos
75                                 ;en el iterador
76
77 .loopAsterisks:                 ;---------- inicio loopAsterisks
78     mov al, '*'
79     call putchar                ;imprimir un asterisco
80     loop .loopAsterisks         ;---------------- fin loopAsterisks
81
82     mov cx, dx                  ;recargar el numero de lineas en
83                                 ;el iterador
84     mov al, 10
85     call putchar                ;imprimir un salto de linea
86     loop .loopLines             ;---------------- fin loopLines
87
88     ret
89
90
91 ;; getsAlpha
92 ;; ========
93 ;; - EBX direccion del string
94 ;;
95 ;; Esta funcion captura un string en una direccion de memoria
96 ;; almacenada en EBX. solamente toma caracteres en [A-Z] y [a-z]
97 ;; y el espacio
98 getsAlpha:
99     call getche                 ;capturar un caracter en al
100
101    cmp al, 8                   ;si el caracter es backspace
102    jz .clear_backspace_alpha
```

```
103
104     cmp al, 10                  ;si el caracter es return
105     jz .end_gets_alpha
106
107     cmp al, ' '                 ;si es espacio
108     jz .putchar_ebx_alpha
109
110 ;; alpha_numeric ------------------------------------------------------------
111     mov ah, al                  ;convertir el valor de AH a
112     or ah, 32                   ;minuscula
113
114     cmp ah, 'a'
115     jl .invalid_char
116
117     cmp ah, 'z'
118     jg .invalid_char
119 ;; end alpha_numeric ----------------------------------------------------------
120
121     jmp .putchar_ebx_alpha
122
123 .invalid_char:                  ;borrar el char si no es valido
124     mov al, 8                   ;caracter backspace
125     call putchar
126
127     mov al, ' '
128     call putchar                ;imprimir un espacio
129
130     mov al, 8                   ;caracter backspace
131     call putchar
132
133     jmp getsAlpha
134
135 .putchar_ebx_alpha:             ;poner AL caracter en ebx
136     mov byte[ebx], al
137     inc ebx
138
139     jmp getsAlpha
140
141 .clear_backspace_alpha:
142     call clean_backspace        ;limipiar el backspace
143     mov byte[ebx], 0            ;poner un null en el caracter
144
145     cmp ebx, 0                  ;revisar si hay mas caracteres
146     jz getsAlpha                ;si esta vacia volver a gets
147
148     dec ebx                     ;moverse un caracter hacia atras
149     jmp getsAlpha
150
```

```
151 .end_gets_alpha:
152     mov byte[ebx], 0            ;poner un null al final
153     mov al, 0                   ;limpiar el valor de al
154     ret
155
156 ;; gets
157 ;; =====
158 ;; - EBX direccion del string
159 ;;
160 ;; Esta funcion captura un string en una direccion de memoria
161 ;; almacenada en EBX.
162 gets:
163     call getche                 ;capturar un caracter en al
164
165     cmp al, 8                   ;si el caracter es backspace
166     jz .clear_backspace
167
168     cmp al, 10                  ;si el caracter es return
169     jz .end_gets
170
171     jmp .putchar_ebx
172
173 .putchar_ebx:                   ;poner AL caracter en ebx
174     mov byte[ebx], al
175     inc ebx
176     jmp gets
177
178 .clear_backspace:
179     call clean_backspace        ;limipiar el backspace
180     mov byte[ebx], 0            ;poner un null en el catacter
181
182     cmp ebx, 0                  ;revisar si hay mas caracteres
183     jz gets                     ;si esta vacia volver a gets
184
185     dec ebx                     ;moverse un caracter hacia atras
186     jmp gets
187
188 .end_gets:
189     mov byte[ebx], 0            ;poner un null
190     mov al, 0                   ;reiniciar el valor de al
191     ret
192
193 ;; clean_backspace
194 ;; ==============
195 ;;
196 ;; Esta función elimina el caracter '^H' al precionar CTRL+h
197 ;; en el teclado (backspace).
198 ;; - CHR_BACKSPACE = 8
```

```
199 ;; - CHR_SPACE = 32
200 clean_backspace:
201     mov cx, 3                   ;iterar 3 veces
202
203 .out:                           ;---------------- inicio bucle
204     mov al, ' '
205     call putchar                ;imprimir un espacio
206
207     mov al, 8                   ;caracter backspace
208     call putchar
209     call putchar                ;moverse a la derecha 2 veces
210
211     loop .out                   ;--------------- terminar bucle
212
213     mov al, ' '                 ;esto no es necesario pero
214     call putchar                ;hace que se vea fancy
215
216     mov al, 8
217     call putchar
218     ret
```

Comentarios

Capturar cosas con asembly no es facil, pero es un desafio interesante.