

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

3. Manipulación de Números

Profesor: José Juan Meza Espinoza

Alumno: Alan Yahir Juárez Rubio

Código: 218517809

Correo: alan.juarez5178@alumnos.udg.mx

Este documento contiene información sensible.
No debería ser impreso o compartido con terceras entidades.

02 de octubre de 2024



Índice

1. Introducción	4
2. Implementación	5
2.1. Código	5
2.2. Ejecución del Programa	8
3. Conclusión	9



Índice de figuras

1.	Suma de dos números	8
----	-------------------------------	---



Índice de códigos

1.	Entrada, salida y suma de números	5
----	---	---



3. Manipulación de Números

1. Introducción

La manipulación de números dentro de ensamblador es de suma importancia debido a que es uno de las bases fundamentales que practicamente cualquier programa necesita. Si bien es sencillo aplicar operaciones aritméticas, el hecho de representarlas al usuario tiene su complejidad a bajo nivel, debido a que es necesario la conversión de valores a ASCII y viceversa.



2. Implementación

2.1. Código

```
1 ; Constantes
2 SYS_EXIT    equ 1
3 SYS_READ    equ 3
4 SYS_WRITE   equ 4
5 STDIN       equ 0
6 STDOUT      equ 1
7
8 NUM_DIGITS  equ 4
9 NUM_CHARS   equ NUM_DIGITS + 1
10
11 ; Variables inicializadas
12 section .data
13     msg_1 DB "Digita el primer número: "
14     len_1 equ $- msg_1
15
16     msg_2 DB "Digita el segundo número: "
17     len_2 equ $- msg_2
18
19     msg_3 DB "El resultado de la suma es: "
20     len_3 equ $- msg_3
21
22 ; Variables sin inicializar
23 section .bss
24     num_1 RESB NUM_CHARS
25     num_2 RESB NUM_CHARS
26
27     res_aux RESB NUM_DIGITS
28     res RESB NUM_DIGITS
29
30 section .text
31     global _start
32
33 _start:
34     ; Impresión de msg_1
35     PUSH DWORD msg_1
36     PUSH DWORD len_1
37     CALL printf
38
39     ; Ingreso de num_1
40     PUSH DWORD num_1
41     PUSH DWORD NUM_CHARS
42     CALL scanf
43
44     ; Impresión de msg_2
45     PUSH DWORD msg_2
46     PUSH DWORD len_2
47     CALL printf
48
49     ; Ingreso de num_2
50     PUSH DWORD num_2
51     PUSH DWORD NUM_CHARS
52     CALL scanf
53
54     ; Suma: num_1 + num_2
55     PUSH DWORD [num_1]
56     PUSH DWORD [num_2]
57     CALL sum
```



```
58
59     ; Conversión de num_1 a ASCII
60     PUSH DWORD num_1
61     CALL ascii_to_num
62
63     ; Conversión de num_2 a ASCII
64     PUSH DWORD num_1
65     CALL ascii_to_num
66
67     ; Impresión de msg_3
68     PUSH DWORD msg_3
69     PUSH DWORD len_3
70     CALL printf
71
72     ; Convertir resultado en ascii
73     CALL num_to_ascii
74
75     ; Impresión de resultado
76     PUSH DWORD res
77     PUSH DWORD NUM_DIGITS
78     CALL printf
79
80 exit:
81     MOV eax, SYS_EXIT
82     MOV ebx, 0
83     INT 0x80
84
85 ; Funciones
86
87 printf:
88     PUSH ebp
89     MOV ebp, esp
90
91     MOV eax, SYS_WRITE
92     MOV ebx, STDOUT
93     MOV ecx, [ebp + 12]
94     MOV edx, [ebp + 8]
95     INT 0x80
96
97     POP ebp
98     RET
99
100 scanf:
101     PUSH ebp
102     MOV ebp, esp
103
104     MOV eax, SYS_READ
105     MOV ebx, STDIN
106     MOV ecx, [ebp + 12]
107     MOV edx, [ebp + 8]
108     INT 0x80
109
110     ; After scanf, remove the newline character
111     MOV BYTE [ebp + 12 + NUM_DIGITS], 0 ; Null-terminate the input string
112     MOV AL, [ebp + 12 + NUM_DIGITS - 1] ; Check the last character
113
114     ; If the newline was found, replace it with a null terminator
115     MOV BYTE [num_2 + NUM_DIGITS - 1], 0
116
117     POP ebp
118     RET
```



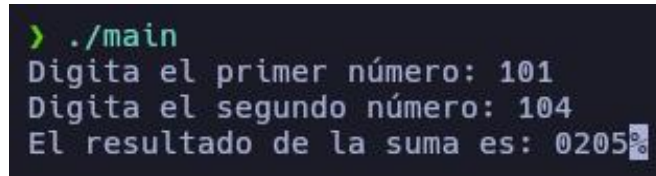
```
119
120 ascii_to_num:
121     PUSH ebp
122     MOV ebp, esp
123
124     MOV edi, NUM_DIGITS - 1
125
126     unidades_1:
127         MOV al, [ebp + 8 + edi]
128         SUB al, '0'
129         ADD bl, al
130
131     decenas_1:
132         DEC edi
133         MOV al, [ebp + 8 + edi]
134         SUB eax, '0'
135
136         MOV ecx, 10
137         MUL ecx
138         ADD bl, al
139
140     centenas_1:
141         DEC edi
142         MOV al, [ebp + 8 + edi]
143         SUB eax, '0'
144
145         MOV ecx, 100
146         MUL ecx
147         ADD bl, al
148
149     MOV edi, 0
150
151     POP ebp
152     RET
153
154 num_to_ascii:
155     MOV eax, [res_aux]
156
157     centenas_2:
158         MOV ebx, 100
159         MOV edx, 0
160         DIV ebx
161         ADD eax, '0'
162         MOV [res], eax
163
164     decenas_2:
165         MOV eax, edx
166         MOV ebx, 10
167         MOV edx, 0
168         DIV ebx
169         ADD eax, '0'
170         MOV [res + 1], eax
171
172     unidades_2:
173         ADD edx, '0'
174         MOV [res + 2], edx
175
176     RET
177
178 sum:
179     PUSH ebp
```



```
180     MOV ebp, esp
181
182     MOV eax, [ebp + 12]
183     MOV ebx, [ebp + 8]
184
185     SUB eax, '0'
186     SUB ebx, '0'
187
188     ADD ebx, eax
189     MOV [res_aux], bl
190
191     POP ebp
192     RET
```

Código 1: Entrada, salida y suma de números

2.2. Ejecución del Programa



```
> ./main
Digita el primer número: 101
Digita el segundo número: 104
El resultado de la suma es: 0205
```

Fig. 1: Suma de dos números



3. Conclusión

Para finalizar, me es importante mencionar que si bien esta actividad parece ser un tanto sencilla, la realidad es que el hecho de trabajar en bajo nivel es lo que lo convierte en laborioso. Cuestiones como conversión de valor a ASCII, almacenamiento de múltiples dígitos, entre otras, fueron un desafío que sin duda me ayudó a comprender muchas cosas, especialmente la manipulación e interpretación de datos numéricos a través de texto.