Práctica 7

Instrucciones de transferencia y aritmeticas.

Luis Eduardo Galindo Amaya (1274895)

Asignatura	Organización de Computadoras (331)
Docente	Arturo Arreola Alvarez
Fecha	07-10-2022

Instrucciones de transferencia y aritmeticas.

Luis Eduardo Galindo Amaya (1274895)

07-10-2022

Instrucciones

Objetivo

Seleccionar las instrucciones aritméticas y de transferencia de datos adecuadas para desarrollar aplicaciones de sistemas basados en microprocesador mediante la distinción de su funcionamiento, de forma lógica y ordenada.

Desarollo

Parte 1

Descargue el archivo print_hex.asm proporcionado en la plataforma. Abra Google Colab y ensamble el código con NASM por medio del comando:

```
nasm -f elf print_hex.asm
```

Encadene el archivo con la librería utilizando el siguiente comando:

```
ld -m -elf_i386 print_hex.o <PATH_de_la_libreria> -o print_hex
```

El cual generará el archivo ejecutable print_hex. Ejecute el archivo por medio del comando:

```
./print_hex
```

Pruebe el programa colocando diferentes valores en EAX para entender el funcionamiento del mismo.

Parte 2

Cree un programa llamado P7.asm que contenga la rutina print_hex, la cual recibe en EAX un valor que se quiere imprimir en formato hexadecimal. Agregue a P7.asm las instrucciones necesarias para hacer lo que se indica a continuación:

- a. Coloque en EBX el valor 0x5C4B2A60. Sume su matrícula como valor hexadecimal. Si su matrícula es 12345678 expresarla como 0x12345678. Almacene el resultado en EBX.
- b. Coloque los 16 bits menos significativos de EBX en la pila.
- c. Defina una variable N de 2 bytes de longitud. En ella, guarde el resultado de la multiplicación de BL por 8, sin considerar los signos.
- d. Incrementar en 1 el valor guardado en N.
- e. Divida el valor almacenado en BX entre 0xFF. Imprima tanto el cociente como el residuo de la operación.
- f. Realice la suma entre el valor almacenado en N y el residuo de la división anterior. Guarde el valor en N y decremente N. Realice las operaciones necesarias para imprimir el registro de banderas y explique que banderas están activas y porque.
- g. Saque un dato de 16 bits de la pila.

NOTA: Por cada inciso, despliegue en pantalla el nuevo valor del registro modificado utilizando la subrutina print_hex.

Desarollo de las actividades

parte 1

El programa se pudo compilar con los siguiente comandos:

```
nasm -f elf print_hex.asm
ld -m elf_i386 print_hex.o libpc_io.a -o a
./a

00BBCCDD
```

parte 2

código

```
;;
;;   AUTHOR: Luis Eduardo Galindo Amaya
;;     DATE: 07-10-2022
;; ASSEMBLE: nasm -f elf print_hex.asm
;;     LINK: ld -m elf_i386 P7.o print_hex.o libpc_io.a -o p7
;;      RUN: ./p7
;;

%include "./pc_io.inc"

section .data                   ;datos inicialiazdos
    salto_de_linea db 0xa, 0x0

section .bss                    ;datos no inicializados
    hex_string resb 255
    n resb 4
    n_foo resb 4
    dividendo resb 4
```

```
19      cociente resb 4
20      residuo resb 4
21
22  section .text
23      global _start:
24
25  _start:
26      ;A) COLOQUE EN EBX EL VALOR 0X5C4B2A60. SUME SU MATRÍCULA
27      ;   COMO VALOR HEXADECIMAL. SI SU MATRÍCULA ES 12345678
28      ;   EXPRESARLA COMO 0X12345678. ALMACENE EL RESULTADO EN EBX
29
30      mov ebx, 0X5C4B2A60
31      add ebx, 0x1274895
32
33      mov eax, ebx
34      mov esi, hex_string
35      call print_hex              ;Salida: 0x5D7272F5
36
37      ;B) COLOQUE LOS 16 BITS MENOS SIGNIFICATIVOS DE EBX EN LA
38      ;   PILA.
39
40      push bx
41      pop ebx
42
43      mov eax, ebx
44      mov esi, hex_string
45      call print_hex
46
47      ;C) DEFINA UNA VARIABLE N DE 2 BYTES DE LONGITUD. EN
48      ;   ELLA, GUARDE EL RESULTADO DE LA MULTIPLICACIÓN DE BL
49      ;   POR 8, SIN CONSIDERAR LOS SIGNOS.
50
51      mov al, 8
52      mul bl
53      mov [n], ax
54      mov ebx, [n]
55
56      mov eax, ebx
57      mov esi, hex_string
58      call print_hex              ;Salida: 0x000007A8
59
60      ;D) INCREMENTAR EN 1 EL VALOR GUARDADO EN N.
61
62      inc word[n]
63      mov eax, [n]
64
65      mov esi, hex_string
66      call print_hex              ;Salida: 0x000007A9
```

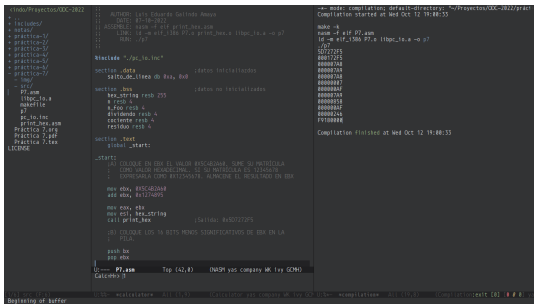
```
67
68      ;E) DIVIDA EL VALOR ALMACENADO EN BX ENTRE 0XFF. IMPRIMA
69      ;   TANTO EL COCIENTE COMO EL RESIDUO DE LA OPERACIÓN.
70
71      mov [dividendo], bx          ;el ultimo valor es n antes del
72                                   ;incremento del inciso D
73
74      mov eax, [dividendo]
75      mov esi, hex_string
76      call print_hex               ;Salida: 0x000007A8
77
78      mov ax, [dividendo]
79
80      mov bl, 0x00ff               ;division
81      div bl
82
83      mov [cociente], al
84      mov [residuo], ah
85
86      mov eax, [cociente]          ;       Salida
87      mov esi, hex_string
88      call print_hex               ;esperada: 0x00000007
89
90      mov eax, [residuo]
91      mov esi, hex_string
92      call print_hex               ;Salida: 0x000000AF
93
94      ;F) REALICE LA SUMA ENTRE EL VALOR ALMACENADO EN N Y EL
95      ;   RESIDUO DE LA DIVISIÓN ANTERIOR. GUARDE EL VALOR EN N Y
96      ;   DECREMENTE N.
97
98      mov ax, [n]                  ;valor previo de n antes de
99      mov [n_foo], ax              ;cualquier operacion
100
101     mov eax, [n_foo]
102     mov esi, hex_string
103     call print_hex               ;salida: 0x000007A9
104
105     mov bx, [residuo]            ;suma del resiudo anterior
106     add ax, bx                   ;el valor del residuo es: 0xAF
107     mov [n], ax                  ;0x000007A9 + 0xAF = 0x00000858
108
109     mov eax, [n]
110     mov esi, hex_string
111     call print_hex               ;obtenido: 0x00000858
112
113     sub ax, [n_foo]              ;0x00000858 - 0x000007A9 = 0xAF
114     mov [n], ax
```

```
115
116     mov eax, [n]
117     mov esi, hex_string
118     call print_hex              ;obtenido: 0x000000AF
119
120     ;REALICE LAS OPERACIONES NECESARIAS PARA IMPRIMIR EL
121     ;REGISTRO DE BANDERAS Y EXPLIQUE QUE BANDERAS ESTÁN ACTIVAS
122     ;Y PORQUE.
123
124     pushfd                      ;guarda las banderas en el stack
125     pop ebx                     ;saca las banderas del stack
126
127     mov eax, ebx
128     mov esi, hex_string
129     call print_hex              ;obtenido: 0x00000246
130
131     ;                          ODIT SZ   AP C
132     ;0000 0000 0000 0000 0000 0010 0100 0110
133     ;
134     ;C: No hay acarreo
135     ;P: AF es un numero impar
136     ;A: No hay acarreo auxiliar
137     ;Z: No es cero
138     ;S: Es positivo
139     ;T: ?
140     ;I: Las interrupciones estan habilitadas
141     ;D: No es una direccion
142     ;O: No hay sobreflujo
143
144     ;G) SAQUE UN DATO DE 16 BITS DE LA PILA.
145
146     pop eax                     ;guarda las banderas en el stack
147
148     mov esi, hex_string
149     call print_hex              ;obtenido: 0xF9180000
150
151
152     mov eax, 1                  ;TERMINAR PROGRAMA
153         mov ebx, 0
154         int 80h
155
156 ; ------------------------------------------------------------------
157
158 print_hex:
159     ;Imprime el valor hexadecimal en de eax
160
161     pushad                      ;Meter a la pila todos los
162                                 ;registros de proposito general
```

```
163
164     mov cl, 28                  ;CL sera el registro para
165                                 ;desplazar
166
167 .next:
168     mov ebx, eax                ;Utilizamos EBX para almacenar
169                                 ;el dato original
170
171     shr ebx, cl                 ;Desplazamos EBX a la derecha
172                                 ;para colocar en los 4 bits
173                                 ;mas a la derecha, los bits a
174                                 ;analizar
175
176     and ebx, 0xf                ;Utilizamos una AND y una
177                                 ;mascara para determinar el
178                                 ;estado de los 4 bits menos
179                                 ;significativos
180
181     cmp bl, 9                   ;Comparamos si el resultado es 9
182                                 ;o menos
183     jbe .menor
184     add bl, 7                   ;Si el resultado es 10 o mas
185                                 ;se representa con 'A-F'
186
187 .menor:
188     add bl, '0'                 ;Convertir al valor de los
189                                 ;caracteres ASCII
190     mov byte [esi], bl
191     inc esi                     ;Almacenar caracter en [esi] e
192                                 ;incrementar ESI
193
194     sub cl,4                    ;Actualizar numero de bits
195                                 ;a desplazar
196
197     cmp cl, 0                   ;Comprobar si se deben seguir
198                                 ;convirtiendo datos o no
199     jl .print
200     jmp .next
201
202 .print:
203     mov edx, hex_string         ;string con la representacion
204                                 ;hexadecimal
205     call puts
206
207     mov edx, salto_de_linea     ;salto de linea
208     call puts
209
210     popad                       ;recarga los datos en la pila en
```

```
211                                     ;los registros de uso general
212     ret
```

Foto



Conclusiones y comentarios

Fue muy interesante ver como hacer operaciones en ASM es bastante diferente de otros lenguajes, y entender por que la pila es tan útil para las operaciones matemáticas.

Dificultades en el desarrollo

Imprimir los valores del hex en la pantalla daba problemas, pude resolver parcialmente el error extendiendo el tamaño de string_hex a 255 y haciendo que todas mis variables fueran de 4 bytes.