Práctica 8

Instrucciones, lógicas y de manipulación de bits.

Luis Eduardo Galindo Amaya (1274895)

Asignatura	Organización de Computadoras (331)
Docente	Arturo Arreola Alvarez
Fecha	14-10-2022

Instrucciones, lógicas y de manipulación de bits.

Luis Eduardo Galindo Amaya (1274895)

14-10-2022

Objetivo

Seleccionar las instrucciones lógicas y de manipulación de bits adecuadas para desarrollar aplicaciones de sistemas basados en microprocesador mediante la distinción de su funcionamiento, de forma lógica y ordenada.

Desarrollo

Parte 1

Descargue el archivo print_hex.asm proporcionado en la plataforma. Abra un notebook en 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

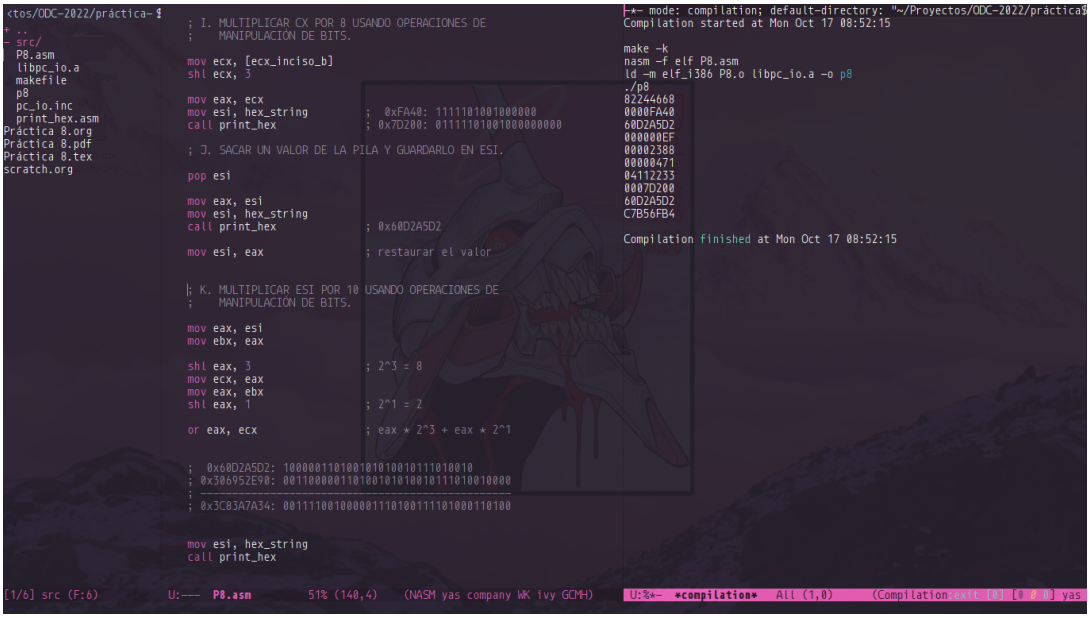
Instrucciones

Cree un programa llamado P8.asm que contenga la rutina print_hex, la cual recibe en EAX un valor que se quiere imprimir en formato hexadecimal. Agregue a P8.asm las instrucciones necesarias para hacer lo que se indica a continuación:

- a. Coloque en EAX el valor 0x22446688 y por medio de rotaciones obtener 0x82244668.
- b. Coloque en CX el valor 0x3F48 y por medio de corrimientos obtener 0xFA40.
- c. Colocar en el registro ESI el valor 0x20D685F3 y por medio de enmascaramiento invertir los bits 0, 5, 13, 18 y 30, sin modificar los demás.
- d. Guardar ESI en la pila
- e. Colocar en el registro CH el valor 0xA7 y por medio de enmascaramiento activar los bits 3 y
- 6, sin modificar los demás.
- f. Colocar en el registro BP el valor 0x67DA y por medio de enmascaramiento desactivar los bits 1, 4, 6, 10 y 14, sin modificar los demás
- g. Dividir BP entre 8 usando operaciones de manipulación de bits.
- h. Dividir EBX entre 32 usando operaciones de manipulación de bits.
- i. Multiplicar CX por 8 usando operaciones de manipulación de bits.
- j. Sacar un valor de la pila y guardarlo en ESI.
- k. Multiplicar ESI por 10 usando operaciones de manipulación de bits.

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

Capturas



Código

```
1  ;;
2  ;;    Author: Luis Eduardo Galindo Amaya
3  ;;      DATE: 14-10-2022
4  ;; ASSEMBLE:
5  ;;      LINK:
6  ;;       RUN:
7  ;;
8
9  %include "./pc_io.inc"
10
11 section .data                    ;datos inicialiazdos
12     salto_de_linea db 0xa, 0x0
13
14 section .bss                     ;datos no inicializados
15     hex_string resb 10
16     eax_inciso_a resb 4
17     ecx_inciso_b resb 4
18     esi_inciso_c resb 4
19
20 section .text
21 global _start
22
23 _start:
24     ; A. COLOQUE EN EAX EL VALOR 0X22446688 Y POR MEDIO DE
25     ;    ROTACIONES OBTENER 0X82244668.
26
```

```
27      mov eax, 0x22446688
28      ror eax, 4
29
30      mov [eax_inciso_a], eax     ;lo ocuparemos en el inciso H
31
32      mov esi, hex_string
33      call print_hex              ;salida: 82244668
34
35      ; B. COLOQUE EN CX EL VALOR 0X3F48 Y POR MEDIO DE
36      ;    CORRIMIENTOS OBTENER 0XFA40.
37
38      mov cx, 0x3F48
39      shl cx, 3
40
41      mov [ecx_inciso_b], ecx     ;lo ocuparemos en el inciso I
42
43      mov eax, 0
44      mov ax, cx
45      mov esi, hex_string         ; 0x3F48: 0011'1111'0100'1000
46      call print_hex              ; 0xFA40: 1111'1010'0100'0000
47
48      ; C. COLOCAR EN EL REGISTRO ESI EL VALOR 0X20D685F3 Y
49      ;    POR MEDIO DE ENMASCARAMIENTO INVERTIR LOS BITS
50      ;    0, 5, 13, 18 Y 30, SIN MODIFICAR LOS DEMÁS.
51
52      mov esi, 0x20D685F3
53      xor esi, 0x40042021
54
55      mov [esi_inciso_c], esi     ;lo ocuparemos en el inciso D
56
57      ; 0x20D685F3: 00100000110101101000010111110011b
58      ; 0x40042021: 01000000000001000010000000100001b
59      ; ------------------------------------------------
60      ; 0x60D2A5D2: 01100000110100101010010111010010b
61
62      mov eax, [esi_inciso_c]
63      mov esi, hex_string
64      call print_hex
65
66      ; D. GUARDAR ESI EN LA PILA
67
68      mov esi, [esi_inciso_c]
69      push esi
70
71      ; E. COLOCAR EN EL REGISTRO CH EL VALOR 0XA7 Y POR MEDIO
72      ;    DE ENMASCARAMIENTO ACTIVAR LOS BITS 3 Y 6, SIN
73      ;    MODIFICAR LOS DEMÁS.
74
```

```
75      mov ch, 0xA7
76      or ch, 0x48
77
78      mov eax, 0                      ; 0xA7: 10100111
79      mov al, ch                      ; 0x48: 01001000
80      mov esi, hex_string             ; ----------------
81      call print_hex                  ; 0xEF: 11101111
82
83      ; F. COLOCAR EN EL REGISTRO BP EL VALOR 0X67DA Y POR
84      ;    MEDIO DE ENMASCARAMIENTO DESACTIVAR LOS BITS
85      ;    1, 4, 6, 10 Y 14, SIN MODIFICAR LOS DEMÁS
86
87      mov bp, 0x67DA
88      and bp, 0xBBAD
89
90      mov eax, 0                      ; 0x67DA: 0110011111011010
91      mov ax, bp                      ; 0xBBAD: 1011101110101101
92      mov esi, hex_string             ; ------------------------
93      call print_hex                  ; 0x2388: 0010001110001000
94
95      ; G. DIVIDIR BP ENTRE 8 USANDO OPERACIONES DE
96      ;    MANIPULACIÓN DE BITS.
97
98      shr bp, 3
99
100     mov eax, 0                      ; 0x2388 / 0x8:    0x471
101     mov ax, bp                      ; ------------------------
102     mov esi, hex_string             ; 0x2388: 10001110001000
103     call print_hex                  ;  0x471:    10001110001
104
105     ; H. DIVIDIR EAX ENTRE 32 USANDO OPERACIONES DE
106     ;    MANIPULACIÓN DE BITS.
107
108     mov eax, [eax_inciso_a]
109     shr eax, 5
110
111     ; 0x82244668: 10000010001001000100011001101000
112     ;       0x20: 100000
113     ; ----------------------------------------------
114     ; 0x4112233: 10000010001001000100011001101000
115
116     mov esi, hex_string
117     call print_hex
118
119     ; I. MULTIPLICAR CX POR 8 USANDO OPERACIONES DE
120     ;    MANIPULACIÓN DE BITS.
121
122     mov ecx, [ecx_inciso_b]
```

```
123     shl ecx, 3
124
125     mov eax, ecx
126     mov esi, hex_string          ;  0xFA40: 1111101001000000
127     call print_hex               ; 0x7D200: 01111101001000000000
128
129     ; J. SACAR UN VALOR DE LA PILA Y GUARDARLO EN ESI.
130
131     pop esi
132
133     mov eax, esi
134     mov esi, hex_string
135     call print_hex               ; 0x60D2A5D2
136
137     mov esi, eax                 ; restaurar el valor
138
139
140     ; K. MULTIPLICAR ESI POR 10 USANDO OPERACIONES DE
141     ;    MANIPULACIÓN DE BITS.
142
143     mov eax, esi
144     mov ebx, eax
145
146     shl eax, 3                   ; 2^3 = 8
147     mov ecx, eax
148     mov eax, ebx
149     shl eax, 1                   ; 2^1 = 2
150
151     or eax, ecx                  ; eax * 2^3 + eax * 2^1
152
153
154     ;  0x60D2A5D2: 10000011010010101010010111010010
155     ; 0x306952E90: 00110000011010010101001011101001000
156     ; ------------------------------------------------
157     ; 0x3C83A7A34: 00111100100000111010011110100011010
158
159
160     mov esi, hex_string
161     call print_hex
162
163
164     ; TERMINAR PROGRAMA
165     mov eax, 1
166     mov ebx, 0
167     int 80h
168
169 ; ------------------------------------------------------------------
170
```

```
171 print_hex:
172     ;Imprime el valor hexadecimal en de eax
173 
174     pushad                      ;Meter a la pila todos los
175                                 ;registros de proposito general
176 
177     mov cl, 28                  ;CL sera el registro para
178                                 ;desplazar
179 
180 .next:
181     mov ebx, eax                ;Utilizamos EBX para almacenar
182                                 ;el dato original
183 
184     shr ebx, cl                 ;Desplazamos EBX a la derecha
185                                 ;para colocar en los 4 bits
186                                 ;mas a la derecha, los bits a
187                                 ;analizar
188 
189     and ebx, 0xf                ;Utilizamos una AND y una
190                                 ;mascara para determinar el
191                                 ;estado de los 4 bits menos
192                                 ;significativos
193 
194     cmp bl, 9                   ;Comparamos si el resultado es 9
195                                 ;o menos
196     jbe .menor
197     add bl, 7                   ;Si el resultado es 10 o mas
198                                 ;se representa con 'A-F'
199 
200 .menor:
201     add bl, '0'                 ;Convertir al valor de los
202                                 ;caracteres ASCII
203     mov byte [esi], bl
204     inc esi                     ;Almacenar caracter en [esi] e
205                                 ;incrementar ESI
206 
207     sub cl,4                    ;Actualizar numero de bits
208                                 ;a desplazar
209 
210     cmp cl, 0                   ;Comprobar si se deben seguir
211                                 ;convirtiendo datos o no
212     jl .print
213     jmp .next
214 
215 .print:
216     mov edx, hex_string         ;string con la representacion
217                                 ;hexadecimal
218     call puts
```

```
219
220     mov edx, salto_de_linea      ;salto de linea
221     call puts
222
223     popad                        ;recarga los datos en la pila en
224                                  ;los registros de uso general
225     ret
```

Conclusiones y comentarios

Las operaciones con bits son más complicadas para nosotros porque estamos acostumbrados a trbajar con los valores decimales, sin embargo la compurtadora trabaja mas facilmente en binario por lo que para ganar mas velocidad de ejecucion podemos utilizar estas operaciones, quiza unos pocos cientos de veces no es justificacion para poder justificar usar ASM pero millones haga alguna diferencia significativa.

Referencias

Bitshifting to multiply an integer by 10 `https://stackoverflow.com/a/10758005`