Práctica

8



**Objetivo**

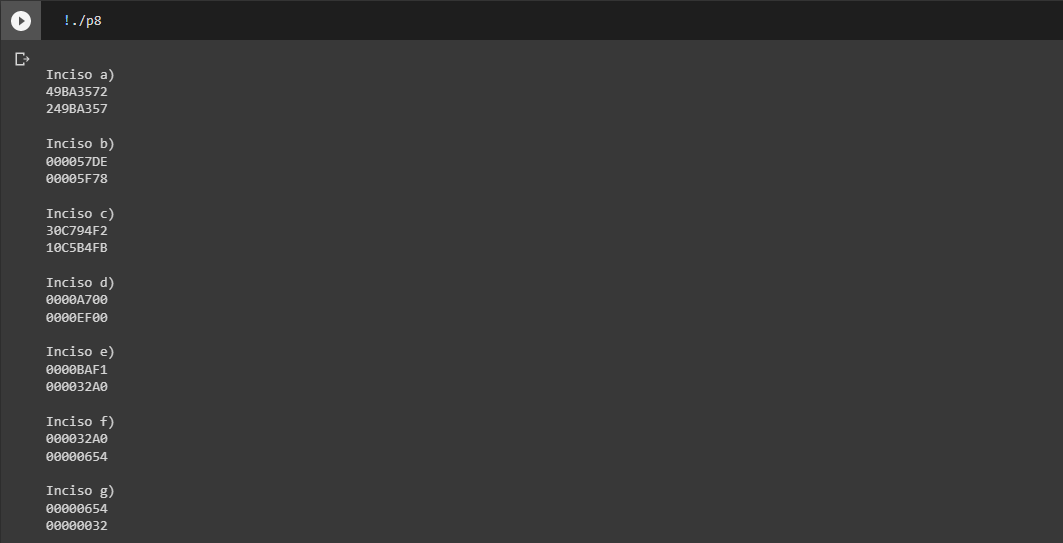
Seleccionar las instrucciones lógicas y de manipulación de bits adecuadas para desarrollar aplicaciones de sistemas basados en microprocesador mediante la distinción de su funcionamiento, de forma lógica y ordenada.

**Desarrollo**

1. Cree un programa llamado **P8.asm** que contenga la rutina *printHex* de la Práctica 5, 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:

1. Colocar en el registro EAX el valor 0x49BA3572 y por medio de rotaciones obtener 0x249BA357.
2. Colocar en el registro BX el valor 0x57DE y por medio de corrimientos obtener 0x5F78.
3. Colocar en el registro ESI el valor 0x30C794F2 y por medio de enmascaramiento invertir los bits 0, 3, 13, 17 y 29, sin modificar los demás.
4. Colocar en el registro CH el valor 0xA7 y por medio de enmascaramiento activar los bits 3 y 6, sin modificar los demás.
5. Colocar en el registro BP el valor 0xBAF1 y por medio de enmascaramiento desactivar los bits 0, 4, 6, 11 y 15, sin modificar los demás.
6. Dividir BP entre 8 usando operaciones de manipulación de bits.
7. Dividir EAX entre 32 usando operaciones de manipulación de bits.
8. Multiplicar BX por 8 usando operaciones de manipulación de bits.
9. Multiplicar ESI por 10 usando operaciones de manipulación de bits.

Por cada inciso, despliegue en pantalla el nuevo valor del registro modificado haciendo uso de la rutina *printHex*.

**Resultado en línea de comandos**

**Código**

section .data

   NL: db 13, 10

   NL\_L: equ $-NL

   ClauseText: db 10,13,"Inciso "

   ClauseText\_L: equ $-ClauseText

   ClauseLetter: db "a)",10,13

   ClauseLetter\_L: equ $-ClauseLetter

section .bss

   cad resb 64

section .text

global \_start:

\_start:

;  a) Colocar en el registro EAX el valor 0x49BA3572 y por medio de rotaciones obtener 0x249BA357

   call printClause

   mov eax,49BA3572h

   call printHex

   mov cl,4

   ror eax,cl

   call printHex

;  b)  Colocar en el registro BX el valor 0x57DE y por medio de corrimientos obtener 0x5F78

   call printClause

   xor bx,bx

   mov bx,57DEh

   mov eax,ebx

   rol bx,1

   shl bx,1

   call printHex

   mov eax,ebx

   call printHex

;  c) Colocar en el registro ESI el valor 0x30C794F2 y por medio de enmascaramiento invertir los bits 0, 3, 13, 17 y 29, sin modificar los demás.

   call printClause

   mov esi,30C794F2h

   mov eax,esi

   call printHex

   xor esi,20022009h

   mov eax,esi

   call printHex

;  d) Colocar en el registro CH el valor 0xA7 y por medio de enmascaramiento activar los bits 3 y 6, sin modificar los demás.

   call printClause

   xor ecx,ecx

   mov ch,0A7h

   mov eax,ecx

   call printHex

   xor ch,48h

   mov eax,ecx

   call printHex

;  e) Colocar en el registro BP el valor 0xBAF1 y por medio de enmascaramiento desactivar los bits 0, 4, 6, 11 y 15, sin modificar los demás.

   call printClause

   xor ebp,ebp

   mov bp,0BAF1h

   mov eax,ebp

   call printHex

   and bp,77AEh

   mov eax,ebp

   call printHex

;  f) Dividir BP entre 8 usando operaciones de manipulación de bits.

   call printClause

   mov cl,3

   mov eax,ebp

   call printHex

   shr bp,cl

   mov eax,ebp

   call printHex

;  g) Dividir EAX entre 32 usando operaciones de manipulación de bits.

   call printClause

   mov cl,5

   call printHex

   shr eax,cl

   call printHex

;  h) Multiplicar BX por 8 usando operaciones de manipulación de bits.

   call printClause

   mov eax,ebx

   call printHex

   mov cl,3

   shl bx,cl

   mov eax,ebx

   call printHex

;  i) Multiplicar ESI por 10 usando operaciones de manipulación de bits.

   call printClause

   mov eax,esi

   call printHex

   mov cl,4

   shl esi,cl

   mov eax,esi

   call printHex

   mov eax,1

   mov ebx,0

   int 80h

printHex:

   pushad

   mov esi,cad

   mov edx, eax

   mov ebx, 0fh

   mov cl, 28

.nxt: shr eax,cl

.msk: and eax,ebx

   cmp al, 9

   jbe .menor

   add al,7

.menor:add al,'0'

   mov byte [esi],al

   inc esi

   mov eax, edx

   cmp cl, 0

   je .print

   sub cl, 4

   cmp cl, 0

   ja .nxt

   je .msk

.print: mov eax, 4

   mov ebx, 1

   sub esi, 8

   mov ecx, esi

   mov edx, 8

   int 80h

   call printNl

   popad

   ret

printNl:

   pushad

   mov eax, 4

   mov ebx, 1

   mov ecx, NL

   mov edx, NL\_L

   int 80h

   popad

   ret

printClause:

   pushad

   ;Imprimir cadena

   mov eax, 4  ;Servicio

   mov ebx, 1  ;Salida

   mov ecx, ClauseText

   mov edx, ClauseText\_L

   int 80h

   ;Imprimir cadena

   mov eax, 4  ;Servicio

   mov ebx, 1  ;Salida

   mov ecx, ClauseLetter

   mov edx, ClauseLetter\_L

   int 80h

   add byte [ClauseLetter],1

   popad

   ret

**Conclusiones y comentarios**

Las operaciones bit a bit proveen bastante utilidad a la hora de realizar operaciones que a simple vista parecen complejas, sin embargo, pueden ser realizadas de una manera bastante sencilla.

**Dificultades en el desarrollo**

Lo que más se me complicó al momento de realizar el programa fue seguirle el ritmo a las rotaciones y desplazamientos, después de eso no tuve grandes complicaciones.