UNIVERSIDAD DE GUADALAJARA

CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERIAS

Actividad 10

Alumno

Alejandro Covarrubias Sánchez

Código

221350192

Profesor

José Juan Meza Espinoza

Carrera

Ingeniería en Computación

Materia

Seminario de Solución de Problemas de Traductores de Lenguajes I

Fecha de entrega

05 de noviembre de 2023

DESPLAZAMIENTOS Y CORRIMIENTOS

Inicio del programa: Declaración de variables

```
org 100h

jmp inicio
    uno db '00000001'
    dos db '00000010'
    cuatro db '00000100'
    ocho db '00001000'
    diez db '00010000'
    veinte db '00100000'
    cuarenta db '01000000'
    ochenta db '10000000'
    auxiliar db 0
    digito db ?
```

Primer segmento del código: Inicio del programa y segmento principal

```
inicio:
    xor ax,ax
    xor bx,bx

    mov al,1h    ;Carga el valor inicial
                 ;AX = 0000 0000 0000 0001 = 0001h

    mov cl,01h   ;CL = 01h

    mov auxiliar,al

    repetir:
        comparar:
            mov digito,al
            mov dl,0
            cmp digito,dl
            jne compararnum

        recorrido:
            xor ax,ax
            xor bx,bx
            mov al,auxiliar
            mov cl,1
            mov digito,al
            shl al,cl
            ;AX = 0000 0000 0000 0010 = 0028h
            mov auxiliar,al

        evaluar:
            mov bh,0
            cmp al,80h
            mov cx,ax
                jcxz repetir2
            jmp repetir
```

```
repetir2:
     comparar2:
          mov al,digito
          mov dl,1
          cmp digito,dl
          cmp al,80h
               je recorrido2
          cmp al,80h
               jne compararnum2

     recorrido2:
          xor ax,ax
          xor bx,bx
          mov al,digito
          mov cl,1
          shr al,cl
          mov digito,al

     evaluar2:
          mov bh,0
          cmp al,1h
          mov cx,ax
               jcxz fin
          jmp repetir2

 fin:
 mov ah, 0
 int 16h
```

Segundo segmento del código: Declaración del segmento ‘Comparar’

```
compararnum:
     mov al,digito
     cmp al,1h
     mov cx,8
     mov dx,offset uno
     mov si,dx
     je imprimenum

     mov al,digito
     cmp al,2h
     mov cx,8
     mov dx,offset dos
     mov si,dx
     je imprimenum

     mov al,digito
     cmp al,4h
     mov cx,8
     mov dx,offset cuatro
     mov si,dx
     je imprimenum
```

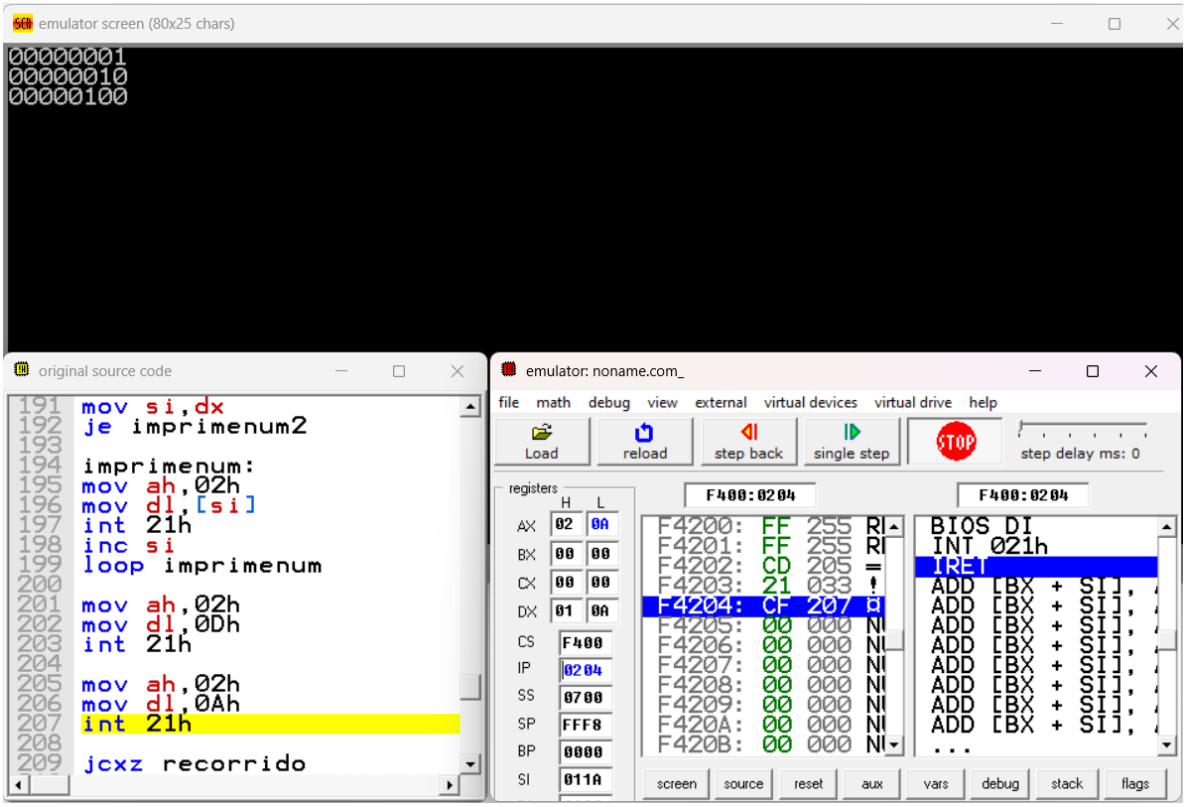
```
  mov al,digito
  cmp al,8h
  mov cx,8
  mov dx,offset ocho
  mov si,dx
  je imprimenum

  mov al,digito
  cmp al,10h
  mov cx,8
  mov dx,offset diez
  mov si,dx
  je imprimenum

  mov al,digito
  cmp al,20h
  mov cx,8
  mov dx,offset veinte
  mov si,dx
  je imprimenum

  mov al,digito
  cmp al,40h
  mov cx,8
  mov dx,offset cuarenta
  mov si,dx
  je imprimenum

  mov al,digito
  cmp al,80h
  mov cx,8
  mov dx,offset ochenta
  mov si,dx
  je imprimenum

compararnum2:
    mov al,digito
    cmp al,1h
    mov cx,8
    mov dx,offset uno
    mov si,dx
    je imprimenum2

    mov al,digito
    cmp al,2h
    mov cx,8
    mov dx,offset dos
    mov si,dx
    je imprimenum2
```

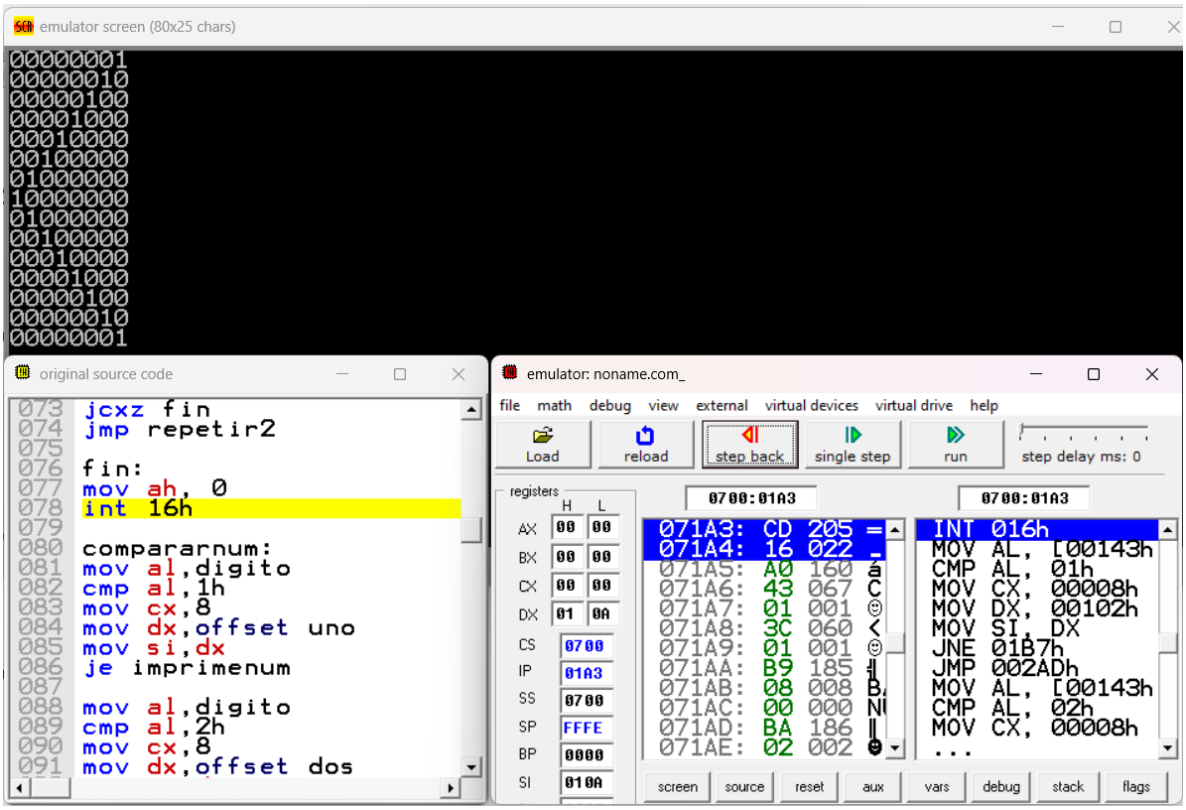
```
mov al,digito
cmp al,4h
mov cx,8
mov dx,offset cuatro
mov si,dx
je imprimenum2

mov al,digito
cmp al,8h
mov cx,8
mov dx,offset ocho
mov si,dx
je imprimenum2

mov al,digito
cmp al,10h
mov cx,8
mov dx,offset diez
mov si,dx
je imprimenum2

mov al,digito
cmp al,20h
mov cx,8
mov dx,offset veinte
mov si,dx
je imprimenum2

mov al,digito
cmp al,40h
mov cx,8
mov dx,offset cuarenta
mov si,dx
je imprimenum2

mov al,digito
cmp al,80h
mov cx,8
mov dx,offset ochenta
mov si,dx
je imprimenum2
```

Tercer segmento del código: Declaración del segmento ‘Imprimir’

```
imprimenum:
     mov ah,02h
     mov dl,[si]
     int 21h
     inc si
     loop imprimenum

     mov ah,02h
     mov dl,0Dh
     int 21h

     mov ah,02h
     mov dl,0Ah
     int 21h

     jcxz recorrido

imprimenum2:
     mov ah,02h
     mov dl,[si]
     int 21h
     inc si
     loop imprimenum2

     mov ah,02h
     mov dl,0Dh
     int 21h

     mov ah,02h
     mov dl,0Ah
     int 21h

     jcxz recorrido2

ret
```

EJECUCIÓN


emulator screen (80x25 chars)
00000001
00000010
00000100
original source code
191 mov si,dx
192 je imprimenum2
193
194 imprimenum:
195 mov ah,02h
196 mov dl,[si]
197 int 21h
198 inc si
199 loop imprimenum
200
201 mov ah,02h
202 mov dl,0Dh
203 int 21h
204
205 mov ah,02h
206 mov dl,0Ah
207 int 21h
208
209 jcxz recorrido
emulator: noname.com_
file math debug view external virtual devices virtual drive help
Load reload step back single step STOP step delay ms: 0
registers
AX 02 0A
BX 00 00
CX 00 00
DX 01 0A
CS F400
IP 0204
SS 0700
SP FFF8
BP 0000
SI 011A
F400:0204
F4200: FF 255
F4201: FF 255
F4202: CD 205 =
F4203: 21 033 !
F4204: CF 207
F4205: 00 000
F4206: 00 000
F4207: 00 000
F4208: 00 000
F4209: 00 000
F420A: 00 000
F420B: 00 000
F400:0204
BIOS DI
INT 021h
IRET
ADD [BX + SI],
screen source reset aux vars debug stack flags


emulator screen (80x25 chars)
00000001
00000010
00000100
00001000
00010000
00100000
01000000
10000000
01000000
00100000
00010000
00001000
00000100
00000010
00000001
original source code
073 jcxz fin
074 jmp repetir2
075
076 fin:
077 mov ah, 0
078 int 16h
079
080 compararnum:
081 mov al,digito
082 cmp al,1h
083 mov cx,8
084 mov dx,offset uno
085 mov si,dx
086 je imprimenum
087
088 mov al,digito
089 cmp al,2h
090 mov cx,8
091 mov dx,offset dos
emulator: noname.com_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers
AX 00 00
BX 00 00
CX 00 00
DX 01 0A
CS 0700
IP 01A3
SS 0700
SP FFFE
BP 0000
SI 010A
0700:01A3
071A3: CD 205 =
071A4: 16 022
071A5: A0 160 á
071A6: 43 067 C
071A7: 01 001
071A8: 3C 060 <
071A9: 01 001
071AA: B9 185
071AB: 08 008
071AC: 00 000
071AD: BA 186
071AE: 02 002
0700:01A3
INT 016h
MOV AL, [00143h
CMP AL, 01h
MOV CX, 00008h
MOV DX, 00102h
MOV SI, DX
JNE 01B7h
JMP 002ADh
MOV AL, [00143h
CMP AL, 02h
MOV CX, 00008h
screen source reset aux vars debug stack flags

CONCLUSIONES:

Para esta actividad se trabajó con el corrimiento y desplazamiento de números representados en binario, para lograr imprimir una ‘flecha’ con los 1. Gracias a esto, pude comprender mejor como trabajar con cadenas y bucles, además de complementar el uso de los saltos condicionales, con los que se trabajaron en actividades pasadas.

BIBLIOGRAFÍA

Brey, B. B. (2006). *Microprocesadores Intel* (Séptima ed.). Pearson.