

LABORATORIO I

Franz López Arredondo 2420191051@estudiantesunibague.edu.co

• ENUNCIADO

Para la elaboración del presente laboratorio se plantearon 4 puntos a desarrollar, a lo largo del documento se mostrarán los resultados obtenidos de dicho taller

• OBJETIVOS ESPECÍFICOS

1. Elaborar códigos que respondan correctamente a lo esperado del laboratorio.
2. Construir mediante el lenguaje ensamblador el código más óptimo posible.

• DESARROLLO

El primer objetivo del laboratorio es realizar un programa que imprima en pantalla un cuadrado de 10x10 con números del 0 al 9 usando un ciclo loop, ahora bien, para la elaboración de este código y asegurar el funcionamiento de el “Loop” se debió asegurar que el ciclo tuviese un límite de repeticiones, esto se aseguro dándole valores al “CX”

```

030  A30:
031  MOV     CX, 30H
032  MOV     CX, UCX
033  DEC     CX
034  LOOP    A20
035  CALL    F10READ
036  CALL    G10MODE
037  MOV     AX, 4C00H
038  INT     21H
039  MAIN  ENDP

```

Imagen.1 Fragmento del primer código

Como se observa en la función A30 se establece el “CX”, sin embargo se hace un llamado a “VCX “, esto con el fin de que cuando termine el primer ciclo, el “CX” cuente con su valor original y varíe con el paso del código, sin embargo, en la línea 033 del código se decrementa, esto se hace para que no siempre conserve el número de veces establecido desde

el principio, sino que, se reduzca con el fin de llegar a 0 y de esta manera deje de repetir su función.

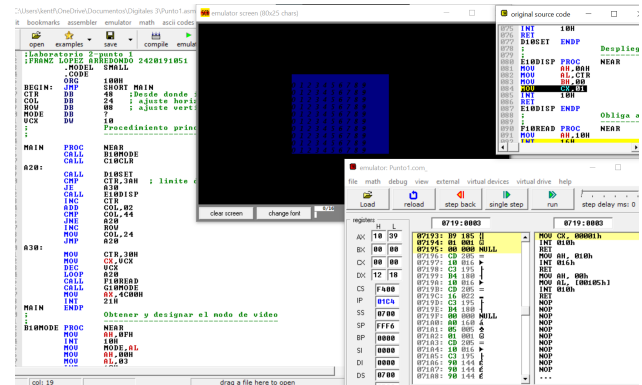


Imagen.2 Comprobación del código

Como se observa en la imagen.2 en la pantalla negra, se repite la serie de números del 0 al 9 en 10 ocasiones, sin embargo, el documento tendrá anexos en los cuales se presentará el código.

Para el segundo punto del taller, se nos pide centrar en pantalla de forma vertical los caracteres ingresados por teclado, haciendo uso de tamaños máximos menores a 15, al elaborar este código se creó una variable donde se guardará lo escrito por el usuario, no sin antes declarar un límite de valores, el cual es el 15, para que los caracteres ingresados no superen dicha cifra.

Ahora, para el ajuste vertical, se realizó un salto de línea y un borrar junto a este, con el fin de eliminar el carácter de espacio que se genera al saltar una línea como se muestra a continuación en la líneas de código 115 y 117.

```

108  CAMBIO
109  PROC    NEAR
110  MOV     AL, [SI]
111  MOV     DI, AL
112  INC     SI
113  INC     DI
114  MOV     DI, 00AH
115  INC     DI
116  MOV     DI, 00AH
117  INC     DI
118  MOV     CL, UARCX
119  DEC     UARCX
120  LOOP    UARCX
121  RET
122  CAMBIO  ENDP

```

Imagen.3 Fragmento de código 2

Sin embargo para que el código cumpla la función requerida se realizó el ajuste de la línea 82 y 83, donde se les denomina un valor vertical y otro horizontal a los caracteres que se registran, tal cual se presencia en la imagen 4

```

076 ;
077 ;----- Centrar y exhibir nombre:
078 F10CENT      PROC      NEAR
079              MOV      DH,NAMELEN      ;Para
080              SHR      DH,1
081              NEG      DH
082              ADD      DH,12            ;Mitad
083              MOV      DL,40            ;Mitad
084              CALL     Q20CURS
085              MOV      AH,09H
086              LEA      DX,SALIDA
087              INT      21H
088              RET

```

Imagen.4, ajuste de caracteres.

Teniendo como resultado lo mostrado en la siguiente imagen.

Imagen.5, resultado.

El tercer código es similar al anterior, con la diferencia de que el usuario elige si desea realizar nuevamente la opción o no, para este caso se asignó el número 1 como “Volver a repetir” y el número 2 haciendo referencia a la terminación del programa.

```

147 ;
148 ;----- Analisis
149 ANALISIS      PROC      NEAR
150              CMP      NAMEFLD,31H
151              JE       A20LOOP
152              CMP      NAMEFLD,32H
153              JE       A30
154              JMP      A30
155 ANALISIS      ENDP
156              END
157              BEGIN

```

Imagen.6, Fragmento código3

Observando la Imagen.6 en la línea 150 se compará los vales y busca si el valor registrado es el correcto, el 1 que en hexadecimal es equivalente a 31H, siendo correcto esto, se permite la repetición de las acciones, sin embargo al seleccionar el 2 (en hexadecimal el 32H) el código finalizará, como se muestra en el línea 152, como prueba superficial del resultado se presentan las siguientes 2 imágenes.

Imagen7, resultado.3,1

Imagen8, resultado.3,2

Por último, se asigna realizar un código en el cual el usuario establece los límites, tanto el máximo como el mínimo de los caracteres que se muestran en pantalla, esto en números Ascii, los cuales cuentan con un rango de 0 a 255.

Al realizar este punto fue necesario el transformar los números ingresados a un solo valor Ascii, ya que el programa lee los números como “1,2,3” en lugar de “123” , para esto se realizaron las operaciones de multiplicar por 100 el el primero de los 3 dígitos (si los hay) por 10 el segundo dígito y por 1 el último dígito para después sumarlos entre ellos y obtener el resultado esperado, con esto se establecen los

límites a gusto del usuario, a continuación se presentará los resultados, sin embargo, en los anexos quedarán los códigos para ser comprendidos con mayor información.

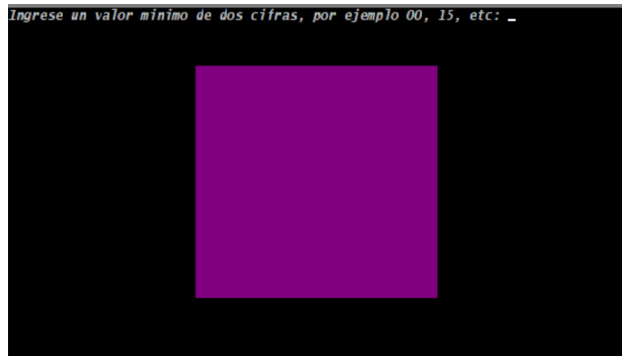


Imagen9, resultado.4,1

Como primer paso se pide ingresar el valor mínimo a establecer.



Imagen10, resultado.4,2

Una vez seleccionado el valor mínimo el código despliega una variable para establecer el rango máximo

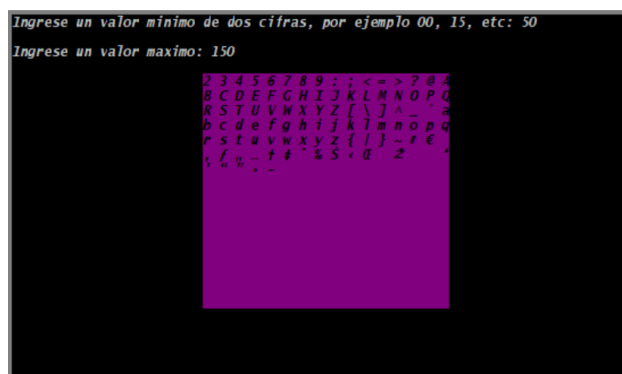


Imagen11, resultado.4,3

Ya establecidos los rangos y oprimiendo “enter” se mostrará en pantalla lo solicitado por el

usuario.

ascii codes

000: null	032: spa	064: @	096: `	128: €	160:	192: À	224: à
001: #	033: !	065: A	097: a	129:	161: i	193: Á	225: á
002: "	034: "	066: B	098: b	130: ,	162: j	194: Â	226: â
003: #	035: #	067: C	099: c	131: f	163: k	195: Ã	227: ã
004: \$	036: \$	068: D	100: d	132: " ' "	164: l	196: Ä	228: ä
005: %	037: %	069: E	101: e	133: -	165: m	197: Å	229: å
006: &	038: &	070: F	102: f	134: _	166: n	198: Æ	230: æ
007: beep	039: '	071: G	103: g	135: `	167: o	199: Ç	231: ç
008: back	040: (072: H	104: h	136: ~	168: p	200: È	232: è
009: tab	041:)	073: I	105: i	137: %	169: q	201: É	233: é
010: newl	042: *	074: J	106: j	138: S	170: r	202: Ê	234: ê
011: #	043: +	075: K	107: k	139: <	171: s	203: Ë	235: ë
012: #	044: ,	076: L	108: l	140: @	172: ~	204: Ì	236: ì
013: #	045: -	077: M	109: m	141:	173: -	205: Í	237: í
014: #	046: .	078: N	110: n	142: Z	174: o	206: Î	238: î
015: #	047: /	079: O	111: o	143:	175: ' "	207: Ï	239: ï
016: #	048: 0	080: P	112: p	144:	176:	208: Ð	240: ð
017: #	049: 1	081: Q	113: q	145: ' "	177: ±	209: Ñ	241: ñ
018: #	050: 2	082: R	114: r	146: ' "	178: º	210: Ò	242: ò
019: #	051: 3	083: S	115: s	147: " "	179: º	211: Ó	243: ó
020: #	052: 4	084: T	116: t	148: " "	180: ' "	212: Ô	244: ô
021: #	053: 5	085: U	117: u	149: ' "	181: µ	213: Õ	245: õ
022: #	054: 6	086: V	118: v	150: -	182: ¶	214: Ö	246: ö
023: #	055: 7	087: W	119: w	151: -	183: -	215: ×	247: ÷
024: #	056: 8	088: X	120: x	152: ' "	184: ' "	216: Ø	248: ø
025: #	057: 9	089: Y	121: y	153: ' "	185: ' "	217: Ù	249: ù
026: #	058: :	090: Z	122: z	154: S	186:	218: Ú	250: ú
027: #	059: ;	091: [123: {	155: >	187: º	219: Û	251: û
028: #	060: <	092: \	124:	156: e	188: º	220: Ü	252: ü
029: #	061: =	093:]	125: }	157:	189: º	221: Ý	253: ý
030: #	062: >	094: ^	126: ~	158: Z	190: º	222: Þ	254: þ
031: #	063: ?	095: _	127: `	159: Y	191: º	223: ß	255: res

Imagen12, tabla Ascii

Una vez se encuentre todo realizado comprobamos los caracteres desplegados con la tabla Ascii, esto para comprobar su correcto funcionamiento.

• Anexos

<https://github.com/franz-b11/Lab-2-DigIII>