



UNIVERSIDAD DE CARABOBO
FACULTAD EXPERIMENTAL DE CIENCIAS Y
TECNOLOGÍA
DEPARTAMENTO DE COMPUTACIÓN
ARQUITECTURA DEL COMPUTADOR



Practica 2

Icler Anaya — C.I. 24.450.482
Wilsen Hernandez — C.I. 24.993.998

Introducción

En este informe se encuentra plasmado la **Práctica n°2** de la materia Arquitectura del Computador, la cual consiste en desarrollar el método de ordenamiento burbuja en el lenguaje ensamblador, para una serie de caracteres dispuestos al azar para su posterior ordenamiento.

El algoritmo de Ordenamiento Burbuja funciona revisando cada elemento de una lista dada comparando un elemento **i** con el elemento siguiente **i+1**, si el primero es mayor que el segundo entonces **i** intercambia posiciones con **i+1**, y si el primer elemento es menor entonces se pasa al siguiente elemento de la lista para ser comparados con los restantes.

La forma metodológica de abordar el problema es la experimental, ya que de esta forma es posible evaluar los procesos, observar los criterios para el armado del programa y conocer el código que hace posible la resolución del problema, además conduce a lograr un mayor dominio en cuanto al lenguaje ensamblador del **microprocesador Intel 8051** mediante la programación de este método de ordenamiento.

Marco Teórico

El problema

Se desea elaborar en lenguaje ensamblador un programa que implemente en memoria el algoritmo de ordenamiento Burbuja, a través de la simulación del comportamiento del *micro-controlador 8051* como arquitectura para el lenguaje, el programa estará dado con un vector compuesto de **n** caracteres **ASCII** en el rango **A-Z** generado a través del método *congruencial lineal mixto* que genera una secuencia repetitiva de **n** números enteros que son una aproximación muy cercana a lo que puede ser una secuencia aleatoria de números. El primer elemento del vector a ordenar estará ubicado en la dirección **30H** de la memoria **RAM**, validando que la cantidad de elementos a ordenar **n** no supera la dirección **7FH**, es decir la cantidad máxima de elementos aceptados es **4FH** (hexadecimal).

Además, se tiene que validar que los elementos del vector se encuentren en el rango de letras mayúsculas “**A-Z**”. A su vez, se debe tener en cuenta que el parámetro **n** estará localizado en el registro **R5** y que el programa intercambie posiciones de los elementos del vector original en el mismo lugar donde el vector reside.

Por último, se debe guardar en las direcciones **2EH** y **2FH** el número total de intercambios realizados durante el proceso de ordenamiento como un número entero de **16 bits**.

Objetivos

- ◆ Implementar de manera eficiente el método de ordenamiento burbuja en el lenguaje ensamblador del *microprocesador Intel 8051*.
- ◆ Identificar cuáles son los modos de direccionamiento del *microprocesador Intel 8051* y listar los mismos con sus ejemplos.
- ◆ Identificar cómo se hacen las comparaciones y ejemplificar.

Además de estos objetivos puntuales y específicos resalta como objetivo realizar comparaciones entre los elementos del vector a ordenar para guardar en las direcciones **2E/2F** el número total de intercambios realizados durante el proceso de ordenamiento.

Aspectos teóricos importantes

El método utilizado para generar una secuencia de elementos aleatorios se ha implementado el *método de congruencia mixto lineal*. Aunque se han desarrollado básicamente tres métodos de congruenciales para generar números pseudoaleatorios, los cuales se derivan del empleo de diferentes versiones de la relación fundamental de congruencia. El objetivo de cada uno de los métodos es la generación en un tiempo mínimo, de sucesiones de números aleatorios con periodos máximos.

El *método congruencial lineal mixto*, esté al igual que otros métodos congruenciales, generan una secuencia de números pseudoaleatorios en la cual el próximo número pseudoaleatorio es determinado a partir del último número generado, es decir, el número pseudoaleatorio X_{n+1} es derivado a partir del número pseudoaleatorio X_n . La relación de recurrencia para el generador congruencial mixto es $X_{n+1} = (aX_n + c) \bmod m$, en donde:

- X_0 es la semilla, $X_0 > 0$.
- a es el multiplicador, $a > 0$.
- c es la constante aditiva $c > 0$.
- m es el modulo ($m > a, X_0, c$).

Esta relación de recurrencia nos dice que X_{n+1} es el residuo al dividir $X_n + c$ entre el modulo. Lo anterior significa que los valores posibles de X_{n+1} son $0, 1, 2, 3, \dots, m-1$, es decir que m representa el número de valores diferentes que pueden ser generados. Para entrar en acción vamos a darle valores arbitrarios a cada uno de estos parámetros y estudiar que reacción tienen en la relación de recurrencia.

Ejemplo: Supongamos que $a = 5$, $c = 7$, $X_0 = 7$ y $m = 8$. Entonces los resultados son:

n	X_n	X_{n+1}
0	7	2
1	2	1
2	1	4
3	4	3
4	3	6
5	6	5
6	5	0
7	0	7
...

Nótese que después de 8 pasadas el valor inicial de X se repite. Decimos entonces que el **periodo del generador** es 8, y efectivamente es igual al módulo. Eso no siempre es así. Viendo un caso donde el periodo es menor a m . El valor de los parámetros es $a = c = X_0 = 4$ y $m = 6$. Ahora los resultados son:

n	x_n	x_{n+1}
0	4	6
1	6	0
2	0	4
...

Como hemos podido observar, este método nos ayuda mucho en la generación de un vector con elementos aleatorios en el *microprocesador Intel 805*, y que efectivamente podremos comparar a cada uno de estos elementos generados con su equivalente al código **ASCII** en el rango **A-Z** en mayúsculas respectivamente.

Algoritmo de ordenamiento Burbuja

El ordenamiento burbuja es un algoritmo de clasificación simple. El ordenamiento burbuja funciona organizando elementos adyacentes repetidamente si no están en el orden correcto. La lista está ordenada, cuando no se necesita intercambio.

Haciendo esto, el elemento más pequeño burbujea al tope, por eso esta técnica ordenamiento se llama ordenamiento burbuja.

El ordenamiento burbuja es un algoritmo de clasificación sencillo que es fácil de entender y rápido de implementar. Sin embargo, en la práctica, no se recomienda.

La complejidad del algoritmo de ordenamiento burbuja es de $O(n^2)$. Su eficiencia disminuye drásticamente cuando aumenta el número de elementos en la lista sin clasificar.

Materiales, métodos, procedimientos

Plataforma de hardware-software

Se ha hecho uso de la herramienta **dosbox** y del simulador **AVSIM51** para la elaboración de la esta práctica, de igual manera de cómo se implementó en la practica 1.

Material de apoyo

- Microcontrolador 8051 – I. Scott MacKemzie & Raphael C.-W. Phan – Pearson.
- Manual del microcontrolador 8051 – Dr. Alejandro Vega.
- 8051 Cross Assembler User's Manual – Metalink Corporation.

Grupo de comandos utilizados

Se utilizaron el siguiente juego de instrucciones: **ADD, SUBB, MUL, DIV y XCH**, para poder elaborar el programa en lenguaje ensamblador.

Instrucción: SUBB

Función: resta con llevada.

Sintaxis: SUBB A,operando

Instrucción	Código de Operación	2º Byte	Bytes	Ciclos	Flags
SUBB A,Rn	1 0 0 1 1 r r r	-	1	1	C-AC-OV
SUBB A,direcc	0x95	direcc	2	1	C-AC-OV
SUBB A,@Ri	1 0 0 1 0 1 1 i	-	1	1	C-AC-OV
SUBB A,#dato	0x94	dato	2	1	C-AC-OV

Operación: SUBB A,operando

$(A) \leq (A) - (C) - \text{operando}$

Descripción: SUBB resta al acumulador el valor del operando y el bit de acarreo C. Deja el resultado en el Acumulador. El valor del operando no resulta afectado.

Instrucción: ADD**Función:** Suma el operando implicado al ACC y deja el resultado en ACC.**Sintaxis:** ADD A,operando

Instrucción	Código de Operación	2º Byte	Bytes	Ciclos	Flags
ADD A,Rn	0 0 1 0 1 r r r	-	1	1	C-AC-OV
ADD A,direcc	0x25	direcc	2	1	C-AC-OV
ADD A,@Ri	0 0 1 0 0 1 1 i	-	1	1	C-AC-OV
ADD A,#dato	0x24	dato	2	1	C-AC-OV

Operación: ADD A,operando $(A) \leq (A) + \text{operando}$ **Descripción:** ADD suma el valor del operando al valor del Acumulador, y deja el resultado en el Acumulador. El valor del operando no resulta afectado.**Instrucción: MUL****Función:** Multiplica el contenido del acumulador por el contenido del registro B.**Sintaxis:** MUL AB

Instrucción	Código de Operación	Bytes	Ciclos	Flags
MUL AB	0xA4	1	4	C, OV

Operación: MUL AB $(A) \leq \text{Byte bajo del producto } (A) \times (B)$ $(B) \leq \text{Byte alto del producto } (A) \times (B)$ **Descripción:** MUL AB multiplica el contenido del acumulador por el contenido del registro B. El byte bajo del resultado de 16 bits se deja en el acumulador, y el byte alto en el registro B. Si el producto es mayor que 255 (0xFF) el flag de **Overflow (OV)** se pone a 1. En caso contrario OV se pone a 0.**Instrucción: DIV****Función:** Divide el contenido del acumulador entre el contenido del registro B.**Sintaxis:** DIV AB

Instrucción	Código de Operación	Bytes	Ciclos	Flags
DIV AB	0x84	1	4	C, OV

Operación: DIV AB

(A) <= Cociente de la división entera (A)/(B)

(B) <= Resto de la división entera (A)/(B)

Descripción: Divide (división entera) el contenido del acumulador entre el contenido del registro B. El cociente se deja en el acumulador y el resto se deja en el registro B. Si inicialmente el registro B tiene valor 0, tras la división el contenido del acumulador y del registro B es indeterminado, y se activa el flag OV.

Instrucción: XCH

Función: Divide el contenido del acumulador entre el contenido del registro B.

Sintaxis: XCH A,operando

Instrucción	Código de Operación	2º Byte	Bytes	Ciclos	Flags
XCH A,Rn	1 1 0 0 1 r r r	-	1	1	-
XCH A,direcc	0xC5	direcc	2	1	-
XCH A,@Ri	1 1 0 0 0 1 1 i	-	1	1	-

Operación: XCH A,operando

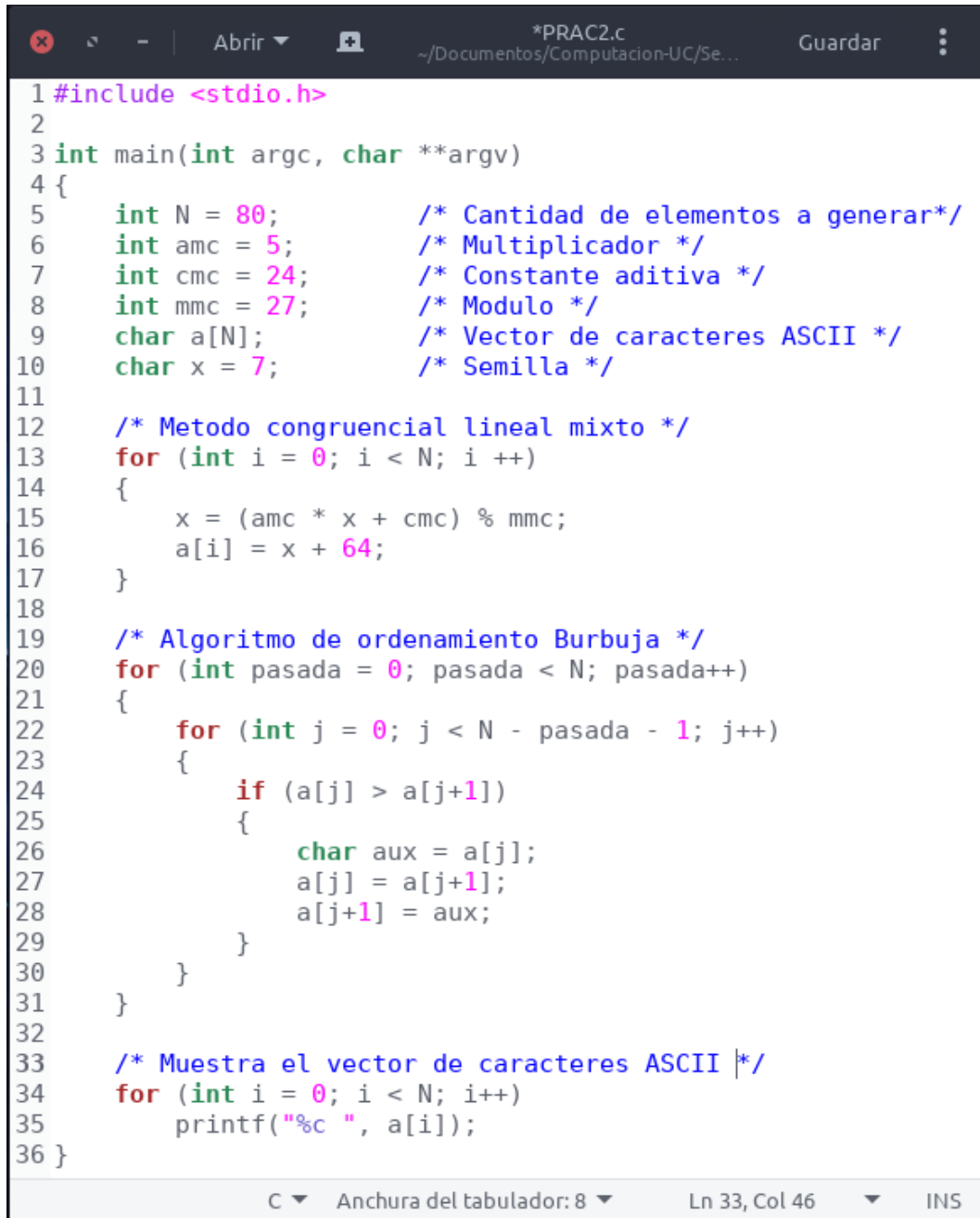
(A) <=> (operando)

Descripción: XCH intercambia los contenidos del Acumulador y del operando implicado.

Procedimientos

El primer paso que se realizó para el cumplimiento de esta práctica fue el de investigar el *método congruencial lineal mixto*, el cual se implementó para generar un vector con **n** elementos **pseudoaleatorios** los cuales posteriormente se ordenaran con el algoritmo de ordenamiento de Burbuja.

El segundo paso fue el elaborar una codificación del programa en un lenguaje de alto nivel como lo es el **lenguaje C**, para luego ser implementado en lenguaje ensamblador. El código elaborado fue el siguiente:



```
1 #include <stdio.h>
2
3 int main(int argc, char **argv)
4 {
5     int N = 80;           /* Cantidad de elementos a generar*/
6     int amc = 5;          /* Multiplicador */
7     int cmc = 24;         /* Constante aditiva */
8     int mmc = 27;         /* Modulo */
9     char a[N];            /* Vector de caracteres ASCII */
10    char x = 7;            /* Semilla */
11
12    /* Metodo congruencial lineal mixto */
13    for (int i = 0; i < N; i++)
14    {
15        x = (amc * x + cmc) % mmc;
16        a[i] = x + 64;
17    }
18
19    /* Algoritmo de ordenamiento Burbuja */
20    for (int pasada = 0; pasada < N; pasada++)
21    {
22        for (int j = 0; j < N - pasada - 1; j++)
23        {
24            if (a[j] > a[j+1])
25            {
26                char aux = a[j];
27                a[j] = a[j+1];
28                a[j+1] = aux;
29            }
30        }
31    }
32
33    /* Muestra el vector de caracteres ASCII */
34    for (int i = 0; i < N; i++)
35        printf("%c ", a[i]);
36 }
```

C Anchura del tabulador: 8 Ln 33, Col 46 INS

Resultados y análisis

Para comprobar que se ha elaborado una buena implementación del ***método congruencial lineal mixto y el algoritmo de ordenamiento Burbuja***, se ha elaborado 3 casos de prueba en los cuales se evaluaron la efectividad de estos dos métodos, el primero de ellos se ha colocado un valor **n=79**, y los resultados obtenidos fueron los siguientes:

Se almacenaron los elementos generados desde la dirección **30H** hasta la dirección **7FH**.

```

x - DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: AVSIM51

LABEL      OPERATION      8051/8751 AVSIM 8051 Simulator/Debugger      V1.31
GEN      MOV      A,R2      CPU REGISTERS      FLAGS      SCL SPD DSP SKP CURSOR
012BH    MOV      B,R7      C Accumulator      AC F0 OV P      OFF HI ON OFF HEX
012DH    MUL      AB      0 01010000:50:P      0 0 0 0      Cycles:
012EH    ADD      A,R3      addr      data
012FH    MOV      B,R4      PC:0139 » F7 09 05 00      Timers TH/TL TF/TR G/T/M1/M0
0131H    DIV      AB      SP: 07 » 21 60 05 18      T0: 00 00 0 0 0 0 0 0
0132H    MOV      A,B      1B 50 00 10      T1: 00 00 0 0 0 0 0 0
0134H    MOV      R7,A      DP:0000 » FF FF FF FF
0135H    ADD      A,#40H      R0:21:1 » 00: RB:05      Ints A S T1 X1 T0 X0 Edg IT IE
0137H    MOV      @R1,A      R1:60: » 00: B:10      En 0 0 0 0 0 0 X0: 0 0
0138H    INC      R1      R2:05 R4:1B R6:00      Pr 0 0 0 0 0 X1: 0 0
0139H    DJNZ     R0,GEN      R3:1B R5:50 R7:10      SBUF: In Out PCON:0:00000000
013CH    MOV      R1,#I      Data Space      00: 00: SCON:00000000
013EH    MOV      R2,#I      0000 21 60 05 18 1B 50 00 10 !^+P Ports
0140H    MOV      R3,#I      0008 00 00 00 00 00 00 00 00 P0 11111111
0142H    MOV      R4,#I      0010 00 00 00 00 00 00 00 00 FF: 11111111
0144H    MOV      R7,#I      0018 00 00 00 00 00 00 00 00 P1 11111111
BUBBLE   MOV      R1,R5      Data Space:R7\+11H      FF: 11111111
0148H    MOV      R2,#I      0020 00 00 00 00 00 00 00 00 P2 11111111
OUTERL   MOV      A,R5      0028 00 00 00 00 00 00 00 00 FF: 11111111
014CH    SUBB     A,R2      0030 45 56 5A 53 4B 59 4E 4D EUZSKYNM P3 11111111
014DH    DEC      A      0038 48 4A 54 50 57 44 51 41 HJTPWDQA FF: 11111111

>Select Command - or use arrow keys

ESC to menu

```

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: AVSIM51
8051/8751 AUSIM 8051 Simulator/Debugger V1.31
LABEL OPERATION CPU REGISTERS FLAGS SCL SPD DSP SKP CURSOR
GEN MOV A,R2 C Accumulator AC F0 OV P OFF HI ON OFF HEX
012BH MOV B,R7 0 01001101:4D:M 0 0 0 0 Cycles:
012DH MUL AB addr data
012EH ADD A,R3 PC:013C > 79 00 7A 00 Timers TH/TL TF/TR G/T/M1/M0
012FH MOV B,R4 SP: 07 > 00 80 05 18 T0: 00 00 0 0 0 0 0 0
0131H DIV AB 1B 50 00 0D T1: 00 00 0 0 0 0 0 0
0132H MOV A,B DP:0000 > FF FF FF FF
0134H MOV R7,A R0:00 > 00: RB:00 Ints A S T1 X1 T0 X0 Edg IT IE
0135H ADD A,#40H R1:00 > FF: B:00 En 0 0 0 0 0 0 0 X0: 0 0
0137H MOV @R1,A R2:05 R4:1B R6:00 Pr 0 0 0 0 0 0 0 X1: 0 0
0138H INC R1 R3:1B R5:50 R7:00 SBUF: In Out PCON:0xxxxxxx
0139H DJNZ R0,GEN Data Space 00: 00: SCON:00000000
013CH MOV R1,#I 0000 00 80 05 18 1B 50 00 0D C+P J Ports
013EH MOV R2,#I 0008 00 00 00 00 00 00 00 00 P0 11111111
0140H MOV R3,#I 0008 00 00 00 00 00 00 00 00 FF: 11111111
0142H MOV R4,#I 0010 00 00 00 00 00 00 00 00 P1 11111111
0144H MOV R7,#I 0018 00 00 00 00 00 00 00 00 FF: 11111111
BUBBLE MOV R1,R5 Data Space:R7\+60H FF: 11111111
0148H MOV R2,#I 0060 57 44 51 41 42 47 45 56 WQABGEV P2 11111111
OUTERL MOV A,R5 0068 5A 53 4B 59 4E 4D 48 4A ZSKYNMHJ FF: 11111111
014CH SUBB A,R2 0070 54 50 57 44 51 41 42 47 TPWLABG P3 11111111
014DH DEC A 0078 45 56 5A 53 4B 59 4E 4D EVZSKYNM FF: 11111111
>Select Command - or use arrow keys
ESC to menu

```

Al terminar la ejecución del programa con el primer caso se obtuvo el resultado deseado, que es tener el vector de elementos **ASCII** ordenamos.

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: AVSIM51
8051/8751 AUSIM 8051 Simulator/Debugger V1.31
LABEL OPERATION CPU REGISTERS FLAGS SCL SPD DSP SKP CURSOR
FIN NOP CPU REGISTERS FLAGS SCL SPD DSP SKP CURSOR
018DH no memory C Accumulator AC F0 OV P OFF HI ON OFF MENU
018EH no memory 0 00000000:00: 1 0 0 0 Cycles:
018FH no memory addr data
0190H no memory PC:018D > 00 FF FF Timers TH/TL TF/TR G/T/M1/M0
0191H no memory SP: 07 > 00 01 4F 00 T0: 00 00 0 0 0 0 0 0
0192H no memory 00 50 00 00 T1: 00 00 0 0 0 0 0 0
0193H no memory DP:0000 > FF FF FF FF
0194H no memory R0:00 > 00: RB:00 Ints A S T1 X1 T0 X0 Edg IT IE
0195H no memory R1:01 > 01: B:41 En 0 0 0 0 0 0 0 X0: 0 0
0196H no memory R2:4F R4:00 R6:00 Pr 0 0 0 0 0 0 0 X1: 0 0
0197H no memory R3:00 R5:50 R7:00 SBUF: In Out PCON:0xxxxxxx
0198H no memory Data Space 00: 00: SCON:00000000
0199H no memory 0000 00 01 4F 00 00 50 00 00 ED P Ports
019AH no memory 0008 31 32 00 00 00 00 00 12 P0 11111111
019BH no memory 0010 00 00 00 00 00 00 00 00 FF: 11111111
019CH no memory 0018 00 00 00 00 00 00 00 00 P1 11111111
019DH no memory Data Space:R7\+20H FF: 11111111
019EH no memory 0020 00 00 00 00 00 00 00 00 P2 11111111
019FH no memory 0028 00 00 00 00 00 00 06 01 FF: 11111111
01A0H no memory 0030 41 41 41 41 42 42 42 42 AAAABBBB P3 11111111
01A1H no memory 0038 44 44 44 44 45 45 45 45 DDDDEEEE FF: 11111111
>Select Command - or use arrow keys
Dump Expression commandFile Help IO Load --space-- ESC to screen

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: AVSIM51									
LABEL	OPERATION	8051/8751 AVSIM 8051 Simulator/Debugger V1.31							
FIN	NOP	CPU REGISTERS				FLAGS			
018DH	no memory	C	Accumulator	AC	F0	OV	P	OFF	HI ON OFF HEX
018EH	no memory	0	00000000:00:		1	0	0	0	Cycles:
018FH	no memory		addr	data					
0190H	no memory	PC:018D	>> 00	FF FF	FF	Timers	TH/TL	TF/TR	G/T/M1/M0
0191H	no memory	SP: 07	>> 00	01 4F	00	T0:	00 00	0 0	00 0 0 0
0192H	no memory			00 50	00	T1:	00 00	0 0	00 0 0 0
0193H	no memory	DP:0000	>> FF	FF FF	FF				
0194H	no memory	R0:00:	>> 00:	RB:00:		Ints	A S T1 X1	T0 X0	Edg IT IE
0195H	no memory	R1:01:	>> 01:	B:41		En	0 0 0 0	0 0	X0: 0 0
0196H	no memory	R2:4F	R4:00	R6:00		Pr	0 0 0 0	0 0	X1: 0 0
0197H	no memory	R3:00	R5:50	R7:00		SBUF:	In Out	PCON:0	XXXXXXXX
0198H	no memory	Data Space				00:	00:	SCON:0	00000000
0199H	no memory	0000	00 01	4F 00	00 50	00 00	EO P	Ports	
019AH	no memory	0008	31 32	00 00	00 00	00 00	12	P0	11111111
019BH	no memory	0010	00 00	00 00	00 00	00 00		FF:	11111111
019CH	no memory	0018	00 00	00 00	00 00	00 00		P1	11111111
019DH	no memory	Data Space:R7\+40H						FF:	11111111
019EH	no memory	0040	45 47	47 47	47 48	48 48	EGGGGHHH	P2	11111111
019FH	no memory	0048	48 4A	4A 4A	4A 4B	4B 4B	HJJJJKKK	FF:	11111111
01A0H	no memory	0050	4B 4B	4D 4D	4D 4D	4D 4E	KKMMMM	P3	11111111
01A1H	no memory	0058	4E 4E	4E 4E	50 50	50 50	NNNNPPPP	FF:	11111111
>Select Command - or use arrow keys									
								ESC to menu	

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: AVSIM51									
LABEL	OPERATION	8051/8751 AVSIM 8051 Simulator/Debugger V1.31							
FIN	NOP	CPU REGISTERS				FLAGS			
018DH	no memory	C	Accumulator	AC	F0	OV	P	OFF	HI ON OFF HEX
018EH	no memory	0	00000000:00:		1	0	0	0	Cycles:
018FH	no memory		addr	data					
0190H	no memory	PC:018D	>> 00	FF FF	FF	Timers	TH/TL	TF/TR	G/T/M1/M0
0191H	no memory	SP: 07	>> 00	01 4F	00	T0:	00 00	0 0	00 0 0 0
0192H	no memory			00 50	00	T1:	00 00	0 0	00 0 0 0
0193H	no memory	DP:0000	>> FF	FF FF	FF				
0194H	no memory	R0:00:	>> 00:	RB:00:		Ints	A S T1 X1	T0 X0	Edg IT IE
0195H	no memory	R1:01:	>> 01:	B:41		En	0 0 0 0	0 0	X0: 0 0
0196H	no memory	R2:4F	R4:00	R6:00		Pr	0 0 0 0	0 0	X1: 0 0
0197H	no memory	R3:00	R5:50	R7:00		SBUF:	In Out	PCON:0	XXXXXXXX
0198H	no memory	Data Space				00:	00:	SCON:0	00000000
0199H	no memory	0000	00 01	4F 00	00 50	00 00	EO P	Ports	
019AH	no memory	0008	31 32	00 00	00 00	00 00	12	P0	11111111
019BH	no memory	0010	00 00	00 00	00 00	00 00		FF:	11111111
019CH	no memory	0018	00 00	00 00	00 00	00 00		P1	11111111
019DH	no memory	Data Space:R7\+60H						FF:	11111111
019EH	no memory	0060	51 51	51 51	53 53	53 53	QQQQSSSS	P2	11111111
019FH	no memory	0068	53 54	54 54	54 56	56 56	TTTTUU	FF:	11111111
01A0H	no memory	0070	56 56	57 57	57 57	59 59	UUUUYY	P3	11111111
01A1H	no memory	0078	59 59	59 5A	5A 5A	5A 5A	YYYZZZ	FF:	11111111
>Select Command - or use arrow keys									
								ESC to menu	

En el segundo caso de prueba, se ha colocado un nuevo valor al $n=15$, el cual nos arrojó de igual manera que en el primer caso, que fue generar una cantidad n de elementos **pseudoaleatorios** y que estos fuesen ordenados efectivamente con el algoritmo de ordenamiento burbuja.

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: AVSIM51
LABEL  OPERATION  8051/8751 AVSIM 8051 Simulator/Debugger  U1.31
GEN    MOV    A,R2   CPU REGISTERS  FLAGS  SCL SPD DSP SKP CURSOR
012BH  MOV    B,R7   C Accumulator  AC F0 OV P  OFF HI ON OFF MENU
012DH  MUL    AB      0 01000001:41:A  0 0 0 0  Cycles:
012EH  ADD    A,R3    addr  data
012FH  MOV    B,R4   PC:0138 >> F7 05 D5 00 Timers TH/TL TF/TR G/T/M1/M0
0131H  DIV    AB      SP: 07 >> 01 3F 05 18  T0: 00 00 0 0 0 0 0 0
0132H  MOV    A,B      1B 10 00 01  T1: 00 00 0 0 0 0 0 0
0134H  MOV    R7,A    DP:0000 >> FF FF FF FF
0135H  ADD    A,#40H  R0:01:0 >> 3F:0 RB:00 Ints A S T1 X1 T0 X0 Edg IT IE
0137H  MOV    @R1,A   R1:0F:0 >> 41:0 B:01 En 0 0 0 0 0 0 0 0 X0: 0 0
0138H  INC    R1      R2:05 R4:10 R6:00 Pr 0 0 0 0 0 0 0 0 X1: 0 0
0139H  DJNZ   R0,GEN  R3:10 R5:10 R7:01 SBUF: In Out PCON:0:00000000
013CH  MOV    R1,#I   Data Space 00: 00: SCON:00000000
013EH  MOV    R2,#I   0000 01 3F 05 18 1B 10 00 01 07:01:0 Ports
0140H  MOV    R3,#I   0008 00 00 00 00 00 00 00 00 P0 11111111
0142H  MOV    R4,#I   0010 00 00 00 00 00 00 00 00 FF: 11111111
0144H  MOV    R7,#I   0018 00 00 00 00 00 00 00 00 P1 11111111
BUBBLE MOV    R1,R5   Data Space FF: 11111111
0148H  MOV    R2,#I   0020 00 00 00 00 00 00 00 00 P2 11111111
014BH  MOV    A,R5   0028 00 00 00 00 00 00 00 00 FF: 11111111
014CH  SUBB   A,R2    0030 45 56 5A 53 4B 59 4E 4D EUZSKYNM P3 11111111
014DH  DEC    A      0038 48 4A 54 50 57 44 51 41 HJTPWDQA FF: 11111111
>Select Command - or use arrow keys
Dump Expression commandFile Help IO Load --space-- ESC to screen

```

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: AVSIM51
LABEL  OPERATION  8051/8751 AVSIM 8051 Simulator/Debugger  U1.31
FIN    NOP          CPU REGISTERS  FLAGS  SCL SPD DSP SKP CURSOR
018DH  no memory    0 00000000:00: 1 0 0 0  OFF HI ON OFF MENU
018EH  no memory    addr  data  Cycles:
018FH  no memory    PC:018D >> 00 FF FF Timers TH/TL TF/TR G/T/M1/M0
0190H  no memory    SP: 07 >> 00 01 0F 00  T0: 00 00 0 0 0 0 0 0
0191H  no memory    00 10 00 00  T1: 00 00 0 0 0 0 0 0
0192H  no memory    DP:0000 >> FF FF FF FF
0193H  no memory    R0:00: >> 00: RB:00 Ints A S T1 X1 T0 X0 Edg IT IE
0194H  no memory    R1:01:0 >> 01:0 B:01 En 0 0 0 0 0 0 0 0 X0: 0 0
0195H  no memory    R2:0F R4:00 R6:00 Pr 0 0 0 0 0 0 0 0 X1: 0 0
0196H  no memory    R3:00 R5:10 R7:00 SBUF: In Out PCON:0:00000000
0197H  no memory    Data Space 00: 00: SCON:00000000
0198H  no memory    0000 00 01 0F 00 00 10 00 00 0A:0A: Ports
0199H  no memory    0008 31 32 00 00 00 00 00 00 12 P0 11111111
019AH  no memory    0010 00 00 00 00 00 00 00 00 FF: 11111111
019BH  no memory    0018 00 00 00 00 00 00 00 00 P1 11111111
019DH  no memory    Data Space FF: 11111111
019EH  no memory    0020 00 00 00 00 00 00 00 00 P2 11111111
019FH  no memory    0028 00 00 00 00 00 00 00 49 I FF: 11111111
01A0H  no memory    0030 41 44 45 48 4A 4B 4D 4E ADEHJKMN P3 11111111
01A1H  no memory    0038 50 51 53 54 56 57 59 5A PQSTUWYZ FF: 11111111
>Select Command - or use arrow keys
Trap: Undefined Address: C:018DH

```

Y en el último caso de prueba sea colocado el valor para n=21, obteniendo los mismo resultados del algoritmo de ordenamiento burbuja.

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: AVSIM51
8051/8751 AVSIM 8051 Simulator/Debugger V1.31
CPU REGISTERS  FLAGS  SCL SPD DSP SKP CURSOR
C Accumulator  AC F0 0U P OFF HI ON OFF MENU
0 01000001:41:A 0 0 0 0 Cycles:
addr data
PC:0138 > F7 05 D5 00 Timers TH/TL TF/TR G/T/M1/M0
SP: 07 > 07 3F 05 18 T0: 00 00 0 0 0 0 0 0
1B 16 00 01 T1: 00 00 0 0 0 0 0 0
DP:0000 > FF FF FF FF
R0:07: > 01:05 RB:00 Ints A S T1 X1 T0 X0 Edg IT IE
R1:3F: > 41:A B:01 En 0 0 0 0 0 0 0 X0: 0 0
R2:05 R4:1B R6:00 Pr 0 0 0 0 0 0 0 X1: 0 0
R3:1B R5:15 R7:01 SBUF: In Out PCON:0xxxxxxx
Data Space 00: 00: SCOM:00000000
0000 07 3F 05 18 1B 16 00 01 00 00 00 00 Ports
0008 00 00 00 00 00 00 00 00 P0 11111111
0010 00 00 00 00 00 00 00 00 FF: 11111111
0018 00 00 00 00 00 00 00 00 P1 11111111
Data Space FF: 11111111
0020 00 00 00 00 00 00 00 00 P2 11111111
0028 00 00 00 00 00 00 00 00 FF: 11111111
0030 45 56 5A 53 4B 59 4E 4D EUZSKYNM P3 11111111
0038 48 4A 54 50 57 44 51 41 HJTPWDQA FF: 11111111
>Select Command - or use arrow keys
Dump Expression commandFile Help IO Load --space-- ESC to screen

```

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: AVSIM51
8051/8751 AVSIM 8051 Simulator/Debugger V1.31
CPU REGISTERS  FLAGS  SCL SPD DSP SKP CURSOR
C Accumulator  AC F0 0U P OFF HI ON OFF MENU
0 01010110:56:U 0 0 0 0 Cycles:
addr data
PC:016C > E7 05 B5 F0 Timers TH/TL TF/TR G/T/M1/M0
SP: 07 > 00 16 00 15 T0: 00 00 0 0 0 0 0 0
00 16 00 00 T1: 00 00 0 0 0 0 0 0
DP:0000 > FF FF FF FF
R0:00: > 45:05 RB:01 Ints A S T1 X1 T0 X0 Edg IT IE
R1:31: > 56:U B:05 En 0 0 0 0 0 0 0 X0: 0 0
R2:00 R4:00 R6:00 Pr 0 0 0 0 0 0 0 X1: 0 0
R3:00 R5:00 R7:00 SBUF: In Out PCON:0xxxxxxx
Data Space 00: 00: SCOM:00000000
0000 00 16 00 15 00 16 00 00 00 00 00 01 Ports
0008 30 31 00 00 00 00 00 00 P0 11111111
0010 00 00 00 00 00 00 00 00 FF: 11111111
0018 00 00 00 00 00 00 00 00 P1 11111111
Data Space:R7\+31H FF: 11111111
0040 42 47 45 56 5A 53 00 00 BGEVZS P2 11111111
0048 00 00 00 00 00 00 00 00 FF: 11111111
0050 00 00 00 00 00 00 00 00 P3 11111111
0058 00 00 00 00 00 00 00 00 FF: 11111111
>Select Command - or use arrow keys
Dump Expression commandFile Help IO Load --space-- ESC to screen

```



```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: AVSIM51
LABEL OPERATION 8051/8751 AUSIM 8051 Simulator/Debugger U1.31
FIN NOP CPU REGISTERS FLAGS SCL SPD DSP SKP CURSOR
018DH no memory C Accumulator AC F0 OV P OFF HI ON OFF HEX
018EH no memory 0 00000000:00: 0 0 0 0 Cycles:
018FH no memory addr data
0190H no memory PC:018D » 00 FF FF Timers TH/TL TF/TR G/T/M1/M0
0191H no memory SP: 07 » 00 01 15 00 T0: 00 00 0 0 0 0 0 0
0192H no memory 00 16 00 00 T1: 00 00 0 0 0 0 0 0
0193H no memory DP:0000 » FF FF FF FF
0194H no memory R0:00: » 00: RB:00 Ints A S T1 X1 T0 X0 Edg IT IE
0195H no memory R1:01: » 01: B:41 En 0 0 0 0 0 0 X0: 0 0
0196H no memory R2:15 R4:00 R6:00 Pr 0 0 0 0 0 0 X1: 0 0
0197H no memory R3:00 R5:16 R7:00 SBUF: In Out PCON:0:00000000
0198H no memory Data Space 00: 00: SCON:00000000
0199H no memory 0000 00 01 15 00 00 16 00 00 ES - Ports
019AH no memory 0008 31 32 00 00 00 00 00 12 P0: 11111111
019BH no memory 0010 00 00 00 00 00 00 00 00 FF: 11111111
019CH no memory 0018 00 00 00 00 00 00 00 00 P1: 11111111
019DH no memory Data Space:R7\+31H FF: 11111111
019EH no memory 0040 56 56 57 59 5A 5A 00 00 UUYZZ P2: 11111111
019FH no memory 0048 00 00 00 00 00 00 00 00 FF: 11111111
01A0H no memory 0050 00 00 00 00 00 00 00 00 P3: 11111111
01A1H no memory 0058 00 00 00 00 00 00 00 00 FF: 11111111
>Select Command - or use arrow keys
ESC to menu

```

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: AVSIM51
LABEL OPERATION 8051/8751 AUSIM 8051 Simulator/Debugger U1.31
FIN NOP CPU REGISTERS FLAGS SCL SPD DSP SKP CURSOR
018DH no memory C Accumulator AC F0 OV P OFF HI ON OFF HEX
018EH no memory 0 00000000:00: 0 0 0 0 Cycles:
018FH no memory addr data
0190H no memory PC:018D » 00 FF FF Timers TH/TL TF/TR G/T/M1/M0
0191H no memory SP: 07 » 00 01 15 00 T0: 00 00 0 0 0 0 0 0
0192H no memory 00 16 00 00 T1: 00 00 0 0 0 0 0 0
0193H no memory DP:0000 » FF FF FF FF
0194H no memory R0:00: » 00: RB:00 Ints A S T1 X1 T0 X0 Edg IT IE
0195H no memory R1:01: » 01: B:41 En 0 0 0 0 0 0 X0: 0 0
0196H no memory R2:15 R4:00 R6:00 Pr 0 0 0 0 0 0 X1: 0 0
0197H no memory R3:00 R5:16 R7:00 SBUF: In Out PCON:0:00000000
0198H no memory Data Space 00: 00: SCON:00000000
0199H no memory 0000 00 01 15 00 00 16 00 00 ES - Ports
019AH no memory 0008 31 32 00 00 00 00 00 12 P0: 11111111
019BH no memory 0010 00 00 00 00 00 00 00 00 FF: 11111111
019CH no memory 0018 00 00 00 00 00 00 00 00 P1: 11111111
019DH no memory Data Space:R7\+31H FF: 11111111
019EH no memory 0040 56 56 57 59 5A 5A 00 00 UUYZZ P2: 11111111
019FH no memory 0048 00 00 00 00 00 00 00 00 FF: 11111111
01A0H no memory 0050 00 00 00 00 00 00 00 00 P3: 11111111
01A1H no memory 0058 00 00 00 00 00 00 00 00 FF: 11111111
>Select Command - or use arrow keys
ESC to menu

```

Con esto hemos podemos decir que, la solución al problema planteado ha sido efectiva para esta práctica.

Conclusión

El *método de ordenamiento burbuja*, ha demostrado su efectividad en esta práctica, al igual que el *método congruencial lineal mixto*, ya que se ha podido cumplir con el objetivo de la práctica, en la elaboración de un programa en lenguaje ensamblador.

Aunque el *método de ordenamiento burbuja* es de $O(n^2)$, no es recomendado para su uso ya que su tiempo de ejecución es lento. Entre los algoritmos de ordenamiento $O(n^2)$ se recomienda el algoritmo de inserción.

Por otra parte, como ya se mencionó, el *método congruencial lineal*, ha sido efectivo en esta práctica, y se obtuvo los resultados deseados con la implementación del mismo, ya que se pudo generar un vector con elementos **ASCII** de forma **pseudoaleatorio**, y poder ser almacenados a partir de la dirección **30H** en **RAM** del *micro-controlador Intel 8051*.

Lista de Referencias

Ehu. 8051 Set de instrucciones: XCH.
http://www.sc.ehu.es/sbweb/webcentro/automatica/web_8051/Contenido/set_8051/Instrucciones/51xch.htm

Ehu. 8051 Set de instrucciones: DIV.
http://www.sc.ehu.es/sbweb/webcentro/automatica/web_8051/Contenido/set_8051/Instrucciones/51div.htm

Ehu. 8051 Set de instrucciones: MUL.
http://www.sc.ehu.es/sbweb/webcentro/automatica/web_8051/Contenido/set_8051/Instrucciones/51mul.htm

Ehu. 8051 Set de instrucciones: ADD.
http://www.sc.ehu.es/sbweb/webcentro/automatica/web_8051/Contenido/set_8051/Instrucciones/51add.htm

Ehu. 8051 Set de instrucciones: SUBB.
http://www.sc.ehu.es/sbweb/webcentro/automatica/web_8051/Contenido/set_8051/Instrucciones/51subb.htm

Omar Trinida(Septiembre 19, 2010). 314159bits. Generar Números Aleatorios con Métodos Congruenciales. <https://314159bits.wordpress.com/2010/09/19/generar-numeros-aleatorios-con-metodo-congruenciales/>

Carlos Marquez Fernandez. Carlos Marquez .Método Congruencial Mixto. <https://carlosmarquez.files.wordpress.com/2012/02/unidad-4-generacion-de-numeros-pseudoaleatorios1.pdf>

Wikipedia. Ordenamiento de burbuja. https://es.wikipedia.org/wiki/Ordenamiento_de_burbuja

Zentut. C Bubble Sort. <http://www.zentut.com/c-tutorial/c-bubble-sort/>