

# Entrada/Salida (E/S) en Computadoras

- PROCEDIMIENTOS

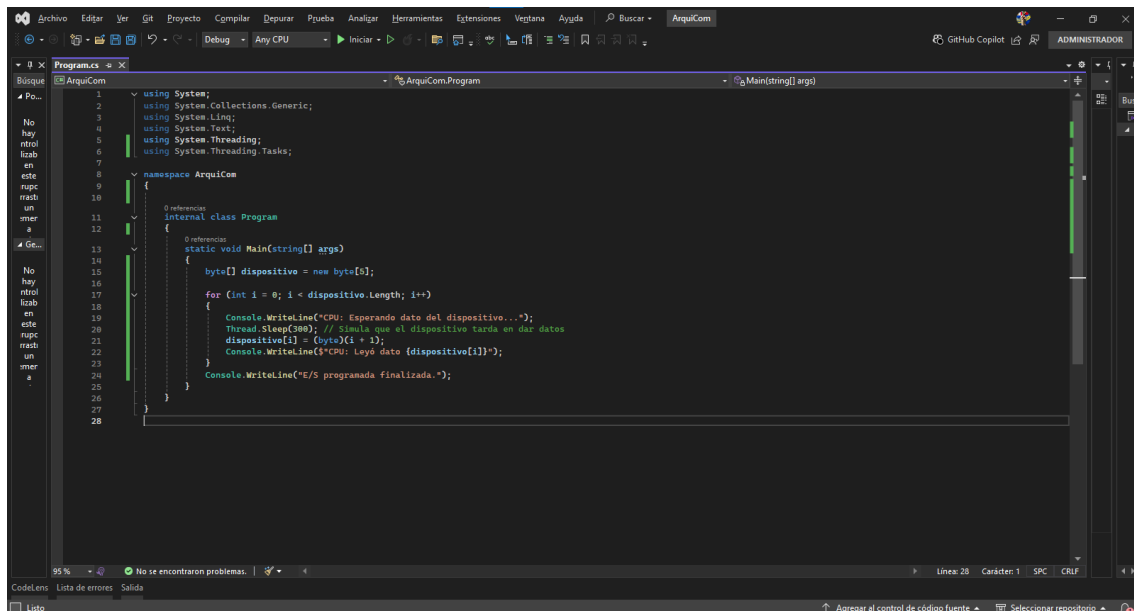
Operación de Entrada/Salida:

Implementar ejemplos de E/S programada y E/S mediante interrupciones usando simuladores.

Analizar el flujo de datos mediante DMA con un ejemplo práctico.

## 1. Ejemplo de E/S Programada (Polling)

En este método, la CPU consulta repetidamente el estado del dispositivo hasta que haya datos listos. Esto consume tiempo de CPU.



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading;
6 using System.Threading.Tasks;
7
8 namespace Arquicom
9 {
10     internal class Program
11     {
12         static void Main(string[] args)
13         {
14             byte[] dispositivo = new byte[5];
15             for (int i = 0; i < dispositivo.Length; i++)
16             {
17                 Console.WriteLine("CPU: Esperando dato del dispositivo...");
18                 Thread.Sleep(300); // Simula que el dispositivo tarda en dar datos
19                 dispositivo[i] = (byte)(i + 1);
20                 Console.WriteLine($"CPU: Leyó dato {dispositivo[i]}");
21             }
22             Console.WriteLine("E/S programada finalizada.");
23         }
24     }
25 }
26
27
28
```



```
1 CPU: Esperando dato del dispositivo...
2 CPU: Leyó dato 1
3 CPU: Esperando dato del dispositivo...
4 CPU: Leyó dato 2
5 CPU: Esperando dato del dispositivo...
6 CPU: Leyó dato 3
7 CPU: Esperando dato del dispositivo...
8 CPU: Leyó dato 4
9 CPU: Esperando dato del dispositivo...
10 CPU: Leyó dato 5
11 E/S programada finalizada.
12
13 C:\Users\GilmaR\source\repos\Arquicom\bin\Debug\Arquicom.exe (proceso 16944) se cerró con el código 0 (0x0).
14 Para cerrar automáticamente la consola cuando se detiene la depuración, habilite Herramientas -> Opciones -> Depuración ->
15 Cerrar la consola automáticamente al detenerse la depuración.
16 Presione cualquier tecla para cerrar esta ventana. . .
17
18
19
20
21
22
23
24
25
26
27
28
```

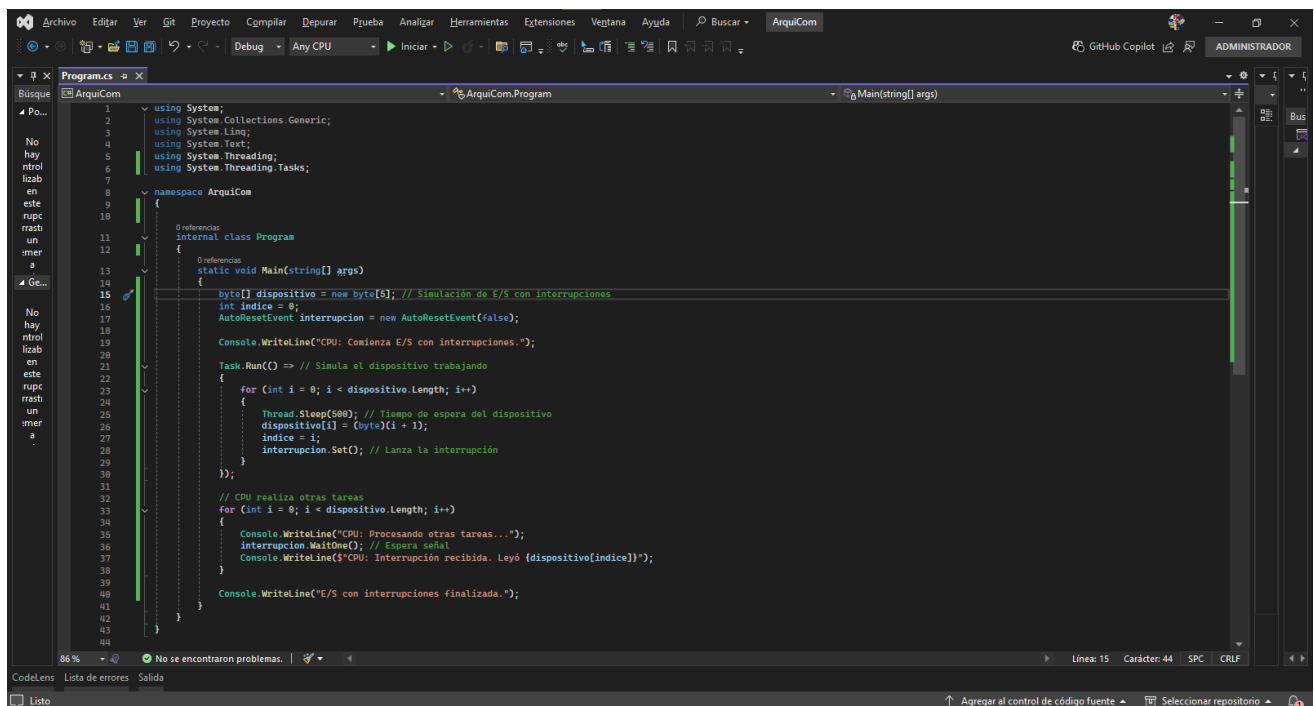
## Explicación:

Este programa simula cómo la CPU lee datos de un dispositivo usando **E/S programada** (polling). Se crea un arreglo de 5 bytes que representa los datos que la CPU debe recibir. En un ciclo, la CPU espera activamente (con una pausa de 300 ms para simular la demora del dispositivo), luego lee un dato (que se asigna como  $i + 1$ ) y lo muestra en pantalla. Este proceso se repite hasta leer los 5 datos. Al final, se indica que la operación de E/S programada terminó.

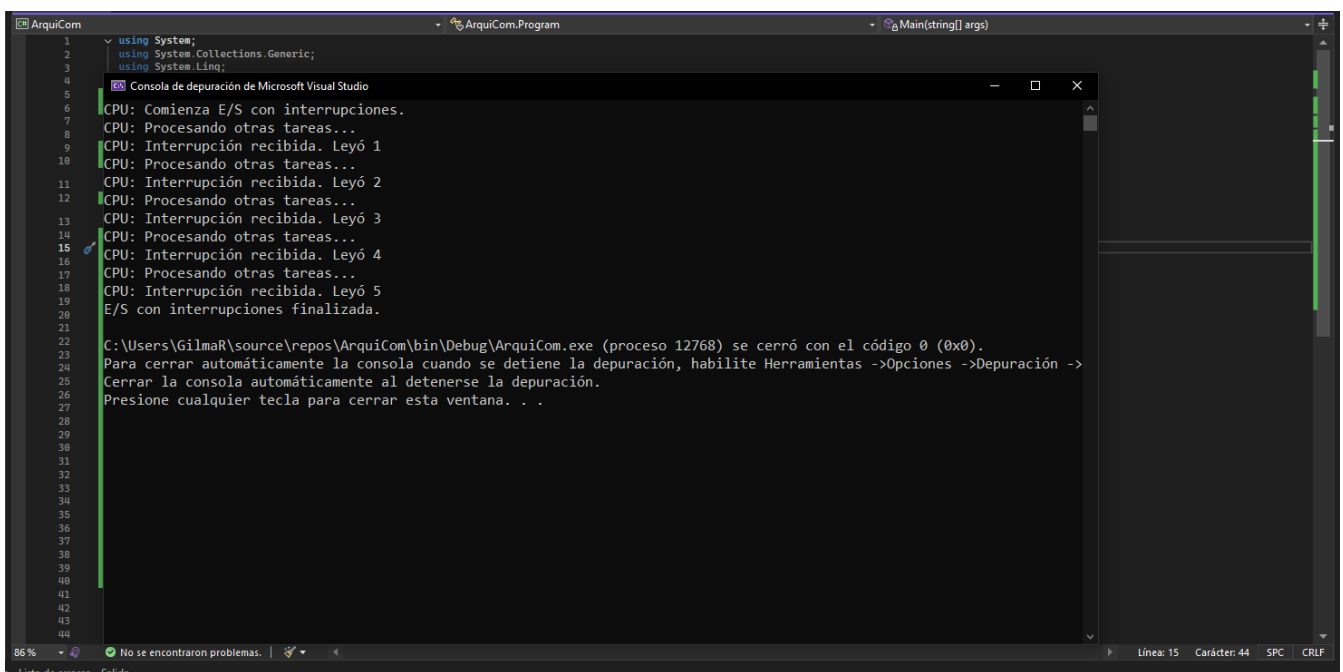
En resumen: el programa muestra cómo la CPU espera y obtiene datos secuencialmente de un dispositivo sin hacer otra cosa mientras espera.

## 2. Ejemplo de E/S mediante Interrupciones

Aquí la CPU puede realizar otras tareas mientras espera que el dispositivo “avise” que tiene datos listos.



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading;
6 using System.Threading.Tasks;
7
8 namespace Arquicom
9 {
10     internal class Program
11     {
12         static void Main(string[] args)
13         {
14             byte[] dispositivo = new byte[5]; // Simulación de E/S con interrupciones
15             int indice = 0;
16             AutoResetEvent interrupcion = new AutoResetEvent(false);
17
18             Console.WriteLine("CPU: Comienza E/S con interrupciones.");
19
20             Task.Run(() => // Simula el dispositivo trabajando
21             {
22                 for (int i = 0; i < dispositivo.Length; i++)
23                 {
24                     Thread.Sleep(500); // Tiempo de espera del dispositivo
25                     dispositivo[i] = (byte)(i + 1);
26                     indice = i;
27                     interrupcion.Set(); // Lanza la interrupción
28                 }
29             });
30
31             // CPU realiza otras tareas
32             for (int i = 0; i < dispositivo.Length; i++)
33             {
34                 Console.WriteLine("CPU: Procesando otras tareas...");
35                 interrupcion.WaitOne(); // Espera señal
36                 Console.WriteLine($"CPU: Interrupción recibida. Leyó {dispositivo[indice]}");
37             }
38
39             Console.WriteLine("E/S con interrupciones finalizada.");
40         }
41     }
42 }
```



```
1 CPU: Comienza E/S con interrupciones.
2 CPU: Procesando otras tareas...
3 CPU: Interrupción recibida. Leyó 1
4 CPU: Procesando otras tareas...
5 CPU: Interrupción recibida. Leyó 2
6 CPU: Procesando otras tareas...
7 CPU: Interrupción recibida. Leyó 3
8 CPU: Procesando otras tareas...
9 CPU: Interrupción recibida. Leyó 4
10 CPU: Procesando otras tareas...
11 CPU: Interrupción recibida. Leyó 5
12 E/S con interrupciones finalizada.
13
14 C:\Users\GilmaR\source\repos\ArquiCom\bin\Debug\ArquiCom.exe (proceso 12768) se cerró con el código 0 (0x0).
15 Para cerrar automáticamente la consola cuando se detiene la depuración, habilite Herramientas ->Opciones ->Depuración ->
16 Cerrar la consola automáticamente al detenerse la depuración.
17 Presione cualquier tecla para cerrar esta ventana. . .
```

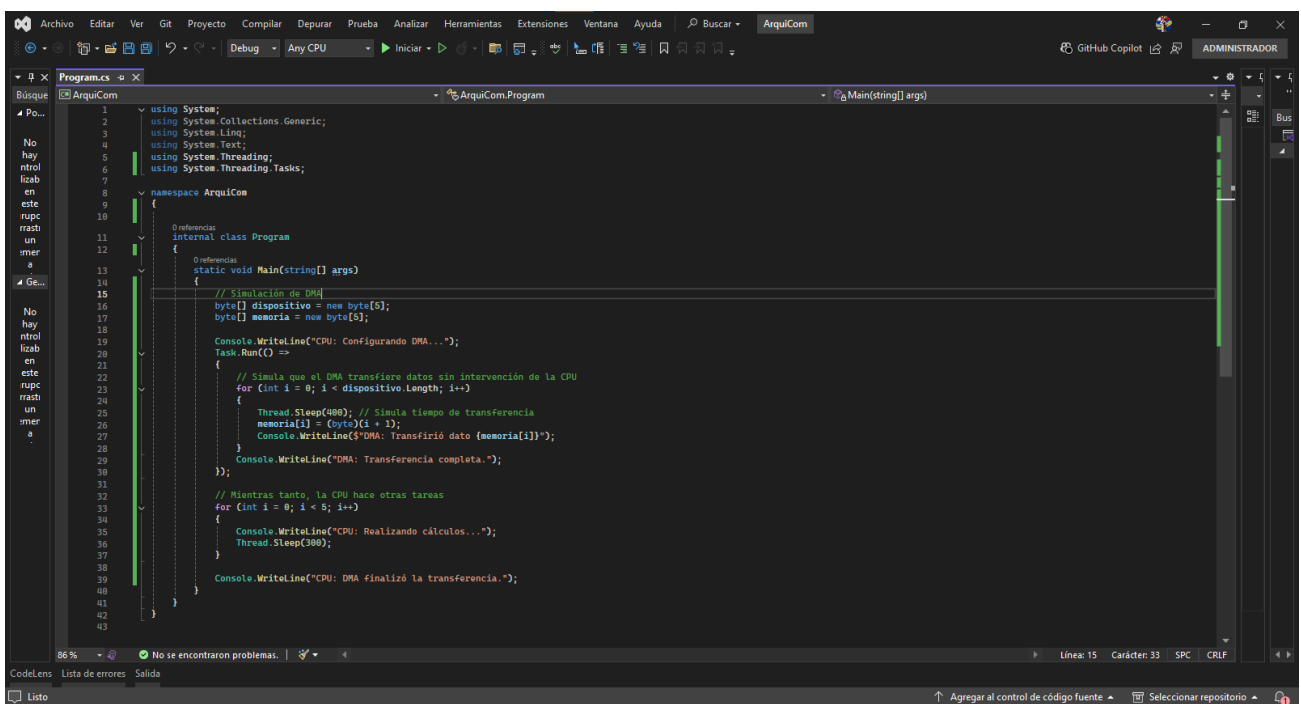
## Explicación:

Este programa simula la comunicación entre un dispositivo y la CPU usando interrupciones en vez de E/S programada. Aquí la CPU no está constantemente esperando, sino que realiza otras tareas hasta que el dispositivo le avisa que hay datos listos (interrupción).

1. Se crea un arreglo dispositivo de 5 bytes para almacenar datos que el dispositivo "envía".
2. Se usa un objeto AutoResetEvent llamado interrupcion, que sirve para que la CPU espere una señal
3. Se inicia una tarea en paralelo (Task.Run) que simula el trabajo del dispositivo:
  - Cada 500 ms, el dispositivo "genera" un dato (valores 1, 2, 3, 4 y 5).
  - Cuando el dato está listo, actualiza la posición actual índice y llama a interrupcion.Set() para notificar a la CPU (simula la interrupción).
4. Mientras tanto, en el hilo principal (CPU), se ejecuta un ciclo donde:
  - La CPU dice que está haciendo otras tareas (simulación).
  - Luego llama a interrupcion.WaitOne(), que hace que espere hasta recibir la señal de interrupción.
  - Cuando recibe la señal, lee el dato actual del arreglo y lo muestra.
5. Al final, se imprime que la operación de E/S con interrupciones ha terminado.

## 3. Ejemplo de DMA (Acceso Directo a Memoria)

En DMA, la CPU solo inicia la transferencia y un controlador especializado mueve los datos directamente a la memoria.



```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading;
6  using System.Threading.Tasks;
7
8  namespace Arquicom
9  {
10
11     [Serializable]
12     internal class Program
13     {
14         static void Main(string[] args)
15         {
16             // Simulación de DMA
17             byte[] dispositivo = new byte[5];
18             byte[] memoria = new byte[5];
19
20             Console.WriteLine("CPU: Configurando DMA...");
21             Task.Run(() =>
22             {
23                 // Simula que el DMA transfiere datos sin intervención de la CPU
24                 for (int i = 0; i < dispositivo.Length; i++)
25                 {
26                     Thread.Sleep(400); // Simula tiempo de transferencia
27                     memoria[i] = (byte)(i + 1);
28                     Console.WriteLine($"DMA: Transfirió dato {memoria[i]}");
29                 }
30                 Console.WriteLine("DMA: Transferencia completa.");
31             });
32
33             // Mientras tanto, la CPU hace otras tareas
34             for (int i = 0; i < 5; i++)
35             {
36                 Console.WriteLine("CPU: Realizando cálculos...");
37                 Thread.Sleep(300);
38             }
39
40             Console.WriteLine("CPU: DMA finalizó la transferencia.");
41         }
42     }
43 }
```

The screenshot shows the Visual Studio IDE with a C# file named 'Program.cs' open. The code is as follows:

```
1 using System;
2 using System.Collections.Generic;
3
4 using Console de depuración de Microsoft Visual Studio
5 using
6 using
7
8 namespace ArquiCom
9 {
10     CPU: Configurando DMA...
11     CPU: Realizando cálculos...
12     CPU: Realizando cálculos...
13     DMA: Transfirió dato 1
14     CPU: Realizando cálculos...
15     DMA: Transfirió dato 2
16     CPU: Realizando cálculos...
17     CPU: Realizando cálculos...
18     DMA: Transfirió dato 3
19     CPU: DMA finalizó la transferencia.
20
21     C:\Users\GilmaR\source\repos\ArquiCom\bin\Debug\ArquiCom.exe (proceso 6536) se cerró con el código 0 (0x0).
22     Para cerrar automáticamente la consola cuando se detiene la depuración, habilite Herramientas ->Opciones ->Depuración ->
23     Cerrar la consola automáticamente al detenerse la depuración.
24     Presione cualquier tecla para cerrar esta ventana. . .
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43 }
```

The debug console on the right displays the execution output, showing the CPU performing calculations and the DMA transferring data. The status bar at the bottom indicates 'No se encontraron problemas' and 'Línea: 15, Carácter: 33, SPC, CRLF'.

### Explicación:

Este programa simula cómo un controlador DMA transfiere datos directamente desde un dispositivo a la memoria sin intervención activa de la CPU. Mientras el DMA transfiere, la CPU puede hacer otras tareas sin bloquearse.

- 1) Se crean dos arreglos de bytes:
  - dispositivo (simula los datos que vienen del dispositivo),
  - memoria (simula la memoria principal donde se copiarán los datos).
- 2) La CPU "configura" el DMA (simulado con un mensaje).
- 3) Se lanza una tarea en paralelo (Task.Run) que representa al controlador DMA:
  - Recorre los 5 datos,
  - Simula un tiempo de transferencia de 400 ms por dato,
  - Copia cada dato al arreglo memoria (aquí simplemente asigna valores de 1 a 5),
  - Muestra en consola qué dato transfirió.
- 4) Mientras el DMA está transfiriendo, en el hilo principal la CPU realiza otras tareas, simuladas con un ciclo que imprime "Realizando cálculos..." y espera 300 ms en cada iteración.
- 5) Al terminar la tarea DMA y las tareas de la CPU, se muestra un mensaje que la transferencia ha finalizado.